
ACE User Guide (UG070)

Achronix Tool Suite



Copyrights, Trademarks and Disclaimers

Copyright © 2024 Achronix Semiconductor Corporation. All rights reserved. Achronix, Speedcore, Speedster, and ACE are trademarks of Achronix Semiconductor Corporation in the U.S. and/or other countries. All other trademarks are the property of their respective owners. All specifications subject to change without notice.

Notice of Disclaimer

The information given in this document is believed to be accurate and reliable. However, Achronix Semiconductor Corporation does not give any representations or warranties as to the completeness or accuracy of such information and shall have no liability for the use of the information contained herein. Achronix Semiconductor Corporation reserves the right to make changes to this document and the information contained herein at any time and without notice. All Achronix trademarks, registered trademarks, disclaimers and patents are listed at <http://www.achronix.com/legal>.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Table of Contents

Chapter 1 : Preface	1
Preface	1
About This Guide	1
Related Documents	1
Conventions Used in this Guide.....	2
Chapter 2 : Getting Started.....	3
Introduction.....	3
ACE Quickstart Tutorial.....	3
1. Create your Project.....	3
2. Add your Design Files and Set Implementation Options.....	4
3. Run the Flow	4
4. Analyze the Results.....	5
Congratulations!!!	5
Chapter 3 : Concepts	6
Workbench.....	6
Perspectives	6
Projects Perspective	6
Floorplanner Perspective	7
IP Configuration Perspective.....	7

2D NoC Performance Perspective	8
Programming and Debug Perspective.....	8
HW Demo Perspective.....	8
Editors.....	9
HTML Report Browser	9
Text Editor	10
VCD Waveform Editor.....	11
<i>VCD Viewer Preferences</i>	15
Views.....	16
Clock Domains View.....	20
<i>Organizing Table Data</i>	22
<i>Drag-and-Drop</i>	23
Clock Regions View	23
<i>Using the Table to Display Clock Regions in the Floorplanner View</i>	26
<i>Organizing Table Data</i>	27
<i>Partial Reconfig Cluster Value</i>	28
Clusters View.....	28
<i>Using the Table to Display Clusters in the Floorplanner View</i>	31
<i>Organizing Table Data</i>	31
<i>Partial Reconfig Cluster Value</i>	33
Critical Path Diagram View	33
<i>Fly-Out Palette</i>	35
Critical Paths View.....	37
Download View.....	40
Floorplanner View	43
<i>Panning and Zooming</i>	46

<i>Fly-Out Palette</i>	47
Flow View.....	53
HW Demo View	58
I/O Designer Toolkit Views	60
<i>I/O Designer Toolkit Views</i>	61
<i>I/O Ring Design File Generation</i>	61
<i>I/O Utilization View</i>	62
<i>I/O Package Diagram View</i>	63
<i>I/O Pin Assignment View</i>	64
<i>I/O Core Pin Assignment View</i>	65
<i>I/O Layout Diagram View</i>	67
IP Diagram View	69
IP Libraries View.....	71
IP Problems View	71
Multiprocess View	73
<i>Execution Queue Management</i>	75
<i>Multiprocess Flow Management</i>	76
<i>Select Implementations</i>	76
<i>Multiprocess Run Logs</i>	79
Netlist Browser View.....	79
<i>Filtering Displayed Instances</i>	86
<i>Drag-and-Drop</i>	87
NoC Performance View	87
<i>Loading Simulation Log Files</i>	88
<i>Browsing Time Slices</i>	88
NoC Time Slice View	93
<i>Adjusting View Properties</i>	94

Options View.....	96
Outline View.....	106
Partitions View.....	107
<i>Drag-and-Drop</i>	109
<i>Partial Reconfig Cluster Value</i>	110
Placement Regions View.....	110
<i>Using the Table to Display Placement Regions in the Floorplanner View</i>	114
<i>Organizing Table Data</i>	115
<i>Partial Reconfig Cluster Value</i>	116
Projects View.....	117
Properties View.....	123
<i>General Tab</i>	124
<i>Parameters Tab</i>	125
<i>Pins Tab</i>	125
<i>Diagram Tab</i>	126
<i>Save Properties</i>	127
Register Browser View.....	127
Search View.....	129
<i>Search Results and ACE Selection</i>	133
<i>Search Highlights</i>	133
Selection View.....	133
Snapshot Debugger View.....	137
Tcl Console View.....	142
Dialogs.....	146
Add Signals to Waveform Viewer Dialog.....	146
Add Source Files Dialogs.....	148
<i>Add IP Configuration Files Dialog</i>	148

Add RTL Files Dialog	148
Add Synthesis Constraint Files Dialog	149
Add Place and Route Constraint Files Dialog	149
Add Place and Route Netlist Files Dialog	150
Add Simulation Testbench Files Dialog	150
Assign Bussed Signal Names Dialog	151
Assign Bussed Values Dialog	153
Method 1	153
Method 2	154
Configure Clock Pre-Routes Dialog	155
Create a New Constraints File Dialog	158
Create a New Flow Step dialog	159
Create a New Text File Dialog	160
Create a SecureShare Zip File Dialog	160
Configuration	161
ACE Input Files	162
ACE Output Files	164
Create Implementation Dialog	166
Create Placement Region Dialog	167
Create Project Dialog	170
Generate I/O Ring Design Files Dialog	171
Generate IP Design Files Dialog	172
Load Acxdb Dialog	174
Load Project Dialog	175
New IP Configuration Dialog	176
Plot SerDes Diagram Dialog	178
MATLAB SerDes Eye Plot Y-axis Unit Formula	180

Restore Implementation Dialog.....	180
Save Implementation Dialog.....	181
Save Placement Dialog.....	182
Save Placement Regions Dialog.....	184
Search Filter Builder Dialog	186
Toolbars.....	188
Preferences.....	188
Configure DCC Connection Preference Page.....	189
Configure JTAG Connection Preference Page.....	190
<i>Multi-device JTAG Scan Chain (IEEE 1149.1) Example</i>	<i>192</i>
Critical Path Diagram View Preference Page.....	193
Floorplanner View Colors and Layers Preference Page	194
Floorplanner View Optimizations Preference Page	200
I/O Designer Preference Page	204
IP Diagram Color and Font Preferences Section.....	205
Multiprocess: Configure Custom Job Submission Tool Preference Page.....	206
Netlist Browser Preference Page.....	208
NoC Performance View Preference Page	209
Other Colors and Fonts Preference Page	211
Placement Regions Preference Page	213
Project Management Preference Page.....	214
Tcl Console View Preference Page.....	216
Text Editors Preference Page.....	218
<i>Quick Diff Preference Page</i>	<i>220</i>
Projects	222
Project File	222

Source Files	224
<i>IP Configuration Files</i>	225
<i>RTL Files</i>	226
<i>Synthesis Constraints Files</i>	226
<i>Place and Route Constraints Files</i>	226
<i>Place and Route Netlist Files</i>	227
<i>Simulation Testbench Files</i>	228
<i>Port Mapping Files</i>	228
Implementations	229
Active Project and Implementation.....	229
Project and Implementation Options	229
<i>Project Options</i>	230
<i>Implementation Options</i>	230
<i>Option Sets</i>	230
Output Files.....	231
<i>Log Files</i>	231
Flow	234
Flow Steps	234
<i>IP Configuration Steps</i>	237
<i>RTL Simulation Steps</i>	237
<i>Synthesis Steps</i>	237
<i>Place and Route Steps</i>	238
<i>Design Completion Steps</i>	240
<i>FPGA Programming Steps</i>	241
Flow Status.....	242
Flow Mode	242
Reports	244

Utilization Report.....	244
Pin Assignment Report	244
Clock Report	244
Timing Report.....	244
<i>Report Content</i>	245
Routing Report	245
Partitions Report	245
Power Dissipation Report	245
<i>Example Power Dissipation Report</i>	246
<i>Power Calculation Methodology</i>	251
<i>How to Generate the Power Dissipation Report</i>	251
<i>How to Generate a Simulation-Driven Power Dissipation Report</i>	252
Design Statistics Report	255
Multiprocess Summary Report.....	255
<i>Timing Results Summary Section</i>	256
<i>Runtime Results Summary Section</i>	257
Implementation Options Report	257
Advanced Concepts	259
ACE Verilog Attributes.....	259
<i>locked</i>	259
<i>fanout_limit</i>	259
<i>must_keep</i>	259
<i>do_not_rewire</i>	260
<i>do_not_clone</i>	260
<i>do_not_cluster</i>	260
<i>clock_type</i>	260
<i>local_clock_type</i>	260

<i>ace_useioff</i>	261
<i>ace_virtualize</i>	261
<i>ace_virtualize_clock_port</i>	261
<i>ace_virtualize_clock_net</i>	261
<i>async_capture</i>	261
<i>location</i>	261
Clock Regions.....	262
Instance States	262
Filter Properties	264
Timing Across All Temperature Corners	266
<i>Temperature Corners and Place and Route</i>	266
<i>Temperature Corners and Timing Analysis/Reports:</i>	266
ECO Commands.....	268
<i>ECO Use Cases</i>	268
<i>Net Legality Definition</i>	268
<i>Disclaimers</i>	268
<i>ECO Commands</i>	269
<i>GUI Support</i>	274
<i>Add Instance Pin Dialog</i>	275
<i>ECO Insert Instance Dialog</i>	276
<i>ECO Insert Net Dialog</i>	278
<i>ECO Rewire Instance Dialog</i>	279
<i>ECO Rewire Net Dialog</i>	280
Fabric Clusters	281

Chapter 4 : Tasks282

Running ACE.....	282
------------------	-----

GUI Mode	282
Command-line Mode	282
Batch Mode	283
Lab Mode (Reduced Functionality).....	283
ACE Initialization Script (ACE_INIT_SCRIPT)	284
<i>Relative paths</i>	284
<i>Example Log Output</i>	284
ACE Startup Arguments.....	285
Working With Perspectives	286
Switching Between Perspectives	287
Resetting Perspectives	287
Working with Views and Editors	287
Opening Views	287
Moving and Docking Views and Editors.....	287
Rearranging Tabbed Views and Editors.....	288
Detaching Views and Editors.....	288
Tiling Editors.....	289
Maximizing, Minimizing, and Restoring Views and Editors.....	289
<i>Maximize:</i>	289
<i>Minimize:</i>	290
Working with Projects and Implementations.....	293
Creating Projects.....	293
Saving Projects	294
Loading Projects.....	295
<i>Loading a Project Using the GUI</i>	295
<i>Loading a Project Using Tcl</i>	296

<i>Project Locking and Lock Files</i>	296
Removing Projects.....	297
Opening Project Files in an Editor.....	297
Adding Source Files.....	297
<i>Source File Load Order</i>	298
<i>Enabling/Disabling Netlists and Constraint Files for Implementations</i>	300
Removing Source Files.....	301
<i>Disabling Constraint Files</i>	301
Opening Source Files in an Editor.....	301
Creating Implementations.....	301
Saving Implementations.....	302
Restoring Implementations.....	302
Copying Implementations.....	303
Setting the Active Implementation.....	303
Removing Implementations.....	303
Configuring Project and Implementation Options.....	304
Opening Output Files in an Editor.....	304
Opening Report Files in an Editor.....	304
Cleaning Projects.....	304
Running the Flow.....	306
Running the Entire Flow.....	306
Running a Sub-Flow.....	307
<i>Run an Individual Flow Step</i>	307
<i>Run Remaining Enabled Flow Steps (Resume Flow)</i>	307
<i>Stopping the Flow</i>	308
Running Multiple Flows in Parallel.....	308

<i>Finding the Multiprocess View</i>	308
<i>Configuring the Execution Queues</i>	308
<i>Configuring the Desired Flow to be Followed by the Selected Implementations</i>	315
<i>Selecting the Implementations to be Run in Parallel</i>	316
<i>Starting Background Execution</i>	316
<i>Stopping/Canceling Background Execution</i>	317
<i>Viewing the Results</i>	317
<i>Multiprocess Batch Mode</i>	318
Detecting Changes to Project Source Files	324
<i>Files Open in the ACE Editor Area</i>	325
<i>Smart Change Detection Using Custom Checksums</i>	325
<i>Saving the Active Implementation</i>	325
<i>Restoring the Active Implementation</i>	325
<i>Caching the Project Source File State</i>	325
<i>Automatic Checking while Running the Flow</i>	325
<i>Warning Visualization in the Flow View</i>	326
<i>Pop-up Dialog Warnings</i>	326
<i>Managing Pop-up Preferences</i>	328
<i>Tcl Command Support</i>	330
Custom Flow Steps.....	330
Using the Tcl Console	332
Sending Commands from GUI Actions.....	332
Sending Commands from the Console	332
Command Highlighting.....	332
Command Auto-Completion.....	332
Command Help	333
Text Limit.....	335

Clearing the Console	335
Viewing the ACE Log File.....	335
Object Type Prefixes	335
Creating an IP Configuration	336
Creating and Naming an IP Configuration.....	337
Setting the IP Configuration	337
<i>Editable Fields</i>	338
<i>Calculated Fields</i>	338
<i>IP Editor Navigation</i>	338
Generating the IP Design Files.....	339
Adding Configuration Files to a Project	339
Viewing the Floorplanner.....	339
Opening and Closing the Floorplanner's Fly-Out Palette.....	339
Zooming the Floorplanner In and Out	339
Floorplanner Panning.....	340
Selecting Floorplanner Objects.....	341
Deselecting Floorplanner Objects.....	341
Toggling Floorplanner Mouse Tools	342
Filtering the Floorplanner View	342
<i>Filtering with Layers</i>	342
<i>Filtering with Selection</i>	342
Choosing Floorplanner Object Tooltips	342
Viewing Floorplanner Object Labels	343
Highlighting Objects in the Floorplanner View.....	343
<i>Selection vs. Highlighting</i>	344
<i>Objects May Be Both Selected and Highlighted Simultaneously</i>	345

<i>Batch Mode Highlighting</i>	346
Pre-Placing a Design.....	346
Placing an Object.....	347
Changing Between Fixed and Soft Placement.....	348
<i>Fixing placement of soft-placement objects</i>	348
<i>Un-fixing (softening) placement of fixed-placement objects</i>	348
Group Placement Mode.....	349
<i>Adjusting the Existing Placement of a Group of Selected Instances</i>	350
Removing Placement.....	351
Saving Pre-Placement Constraints	351
Using Pre-Placement in the Flow	352
<i>Types of Pre-Placement</i>	352
<i>Recommended Typical Flow</i>	352
Analyzing Critical Paths	353
Generating Timing Reports.....	353
Highlighting Critical Paths.....	354
Selecting Critical Path Objects	354
Zooming to Critical Paths	355
Printing Critical Path Details	355
Using Critical Path Diagrams.....	356
<i>Graph Elements</i>	356
<i>Critical Path Diagram Types</i>	357
<i>Adding Portions of the Graph to the ACE Selection Set</i>	357
Applying and Checking Properties	357
Applying Properties.....	357
<i>defparam</i>	357

<i>RTL Attribute</i>	357
<i>set_property Tcl Command</i>	358
Checking Whether Properties Were Applied	358
Configuring External Connections to Hardware	358
Configuring the DCC Connection	358
<i>Background Info</i>	359
<i>Installing DCC USB Drivers</i>	359
<i>Configuring the USB drivers</i>	359
<i>Configuring ACE</i>	360
Configuring the JTAG Connection	360
<i>JTAG Programmer Device Connection</i>	361
<i>JTAG Scan Chain</i>	362
Running the Snapshot Debugger	364
Snapshot Design Flow	364
Accessing the Snapshot Debugger	366
<i>Open the ACE GUI and Select the Project</i>	366
<i>Open the Snapshot Debugger</i>	366
Configuring the Trigger Pattern	367
<i>Configuring the Trigger Mode</i>	367
<i>Configuring Trigger Patterns</i>	368
Configuring the Monitor Signals	371
<i>Naming Captured Signal Data</i>	371
Configuring Test Stimulus	372
<i>Setting Stimuli Values Using the Table</i>	373
<i>Setting Multiple Stimuli Values as a Bus</i>	373
Configuring Advanced Options	374

<i>Pre-Store</i>	374
<i>Trigger Pattern Match Behavior</i>	375
<i>User Clock Frequency</i>	375
<i>Configure Output File Locations</i>	375
Collecting Samples of the User Design	376
<i>Arming the Snapshot Debugger</i>	376
<i>Using the Startup Trigger</i>	377
Viewing the Captured VCD Waveform	378
<i>Opening a VCD file</i>	379
<i>Viewing the Raw Text Content</i>	380
<i>Viewing the Signal Data as Waveforms</i>	380
Saving/Loading Snapshot Configurations	387
Snapshot in Batch Mode.....	387
Programming a Device using JTAG in the Download View.....	390
Selecting a Bitstream File	390
<i>Lab Mode</i>	391
Selecting Bitstream Programming Options.....	391
Downloading the Bitstream to the Target Device.....	391
Optimizing a Design.....	392
Attempting Likely Optimizations Using Option Sets.....	392
<i>Selecting the Implementations to be Generated and Run in Parallel</i>	392
<i>Generating Option Set Implementations and Starting Background Execution</i>	393
<i>Interpreting/Utilizing the Results</i>	393
Placement Regions and Placement Region Constraints	394
Placement Region Preferences.....	395
Creating a New Placement Region	395

Resizing an Existing Placement Region	397
Moving an Existing Placement Region	397
Assigning Placement Region Constraints	398
Listing all Objects Constrained to a Placement Region	400
Removing a Placement Region Constraint from an Object	400
Saving Placement Region Definitions and Placement Region Constraints	400
Deleting Placement Regions.....	401
Running the HW Demo	401
Installing HW Demo Designs	401
<i>HW Demo Installation Paths</i>	402
Selecting The Target Device And Demo	402
Loading The Demo JAM File.....	402
Displaying Board Status.....	403
Control of Running Demonstration Design	404
Using Incremental Compilation (Partitions)	404
Overview of Incremental Compilation and Partitions	404
<i>Defining Partitions</i>	404
<i>Enabling Incremental Compilation</i>	405
<i>Tracking Partition Status</i>	405
<i>Forcing an Unchanged Partition to Recompile</i>	406
<i>Viewing Instances In Partitions</i>	407
<i>Related Tcl Commands</i>	407
Incremental Compile Tutorial.....	408
<i>Overview</i>	408
<i>Tutorial Files</i>	408
Single-Process Incremental Compile Tutorial	408

Step 1: Obtaining the Files.....	409
Step 2: Set up the Synthesis Project	410
Step 3: Compile the Design in Synplify Pro	415
Step 4: Review Synplify Results	418
Step 5: Set up the ACE Project	422
Step 6: Compile the Design in ACE.....	425
Step 7: Review ACE Results	425
Step 8: Change the RTL (rtl_V1)	431
Step 9: Recompile the Design in Synplify Pro (rtl_V1).....	431
Step 10: Review Synplify Results (rtl_V1)	432
Step 11: Recompile the Design in ACE (rtl_V1).....	433
Step 12: Review ACE Results (rtl_V1)	434
Step 13: Additional Incremental Iterations	437
Step 14: Review ACE Results (rtl_V2).....	438
Step 15: Review ACE Results (rtl_V3).....	441
Step 16: Review ACE Results (rtl_V4)	444
Multiprocess Incremental Compile Tutorial	447
Step 1: Compile the Design in Synplify Pro or Clear the ACE Project.....	448
Step 2: Create Multiprocess Implementations and Run ACE	448
Step 3: Select the Implementation with Best Performance.....	449
Step 4: Change the RTL and Recompile the Design in Synplify Pro.....	453
Step 5: Recompile the Multiprocess Implementations in ACE.....	453
Automatic Flop Pushing into I/O Pins.....	457
Background.....	457
Capabilities.....	458
ACE Implementation Options	459
push_flops_into_pads.....	459

<i>pad_flop_pushing_clock_type</i>	459
ACE Attributes	460
Examples.....	461
Timing Analysis Implications.....	465
Working with Virtual I/O.....	466
Behavior	466
Implementation Options	467
Port Attributes.....	467
Runtime Messages	468
Schematic View.....	469
<i>Input Netlist</i>	469
<i>Output Netlist Styles</i>	470
Accessing Help	473
Accessing Context-Sensitive Help	473
Navigating Help Topics.....	473
<i>Using the Help Window</i>	474
<i>Using the Help View</i>	475
Searching Help.....	475
<i>Refining the Search Results</i>	475
<i>Highlighting Search Terms</i>	475
<i>Search Query Syntax</i>	475
Using the ACE SecureShare Tool to Create a Support Zip File.....	476
Importing and Exporting Preferences.....	477
Import Preferences.....	477
Export Preferences	479
Plotting SerDes Receiver Diagrams Using JTAG	482

Plotting a SerDes Diagram for a SerDes Lane	482
Using Partial Reconfiguration	484
Partial Reconfiguration Tutorial.....	484
<i>Partial Reconfiguration Flows</i>	484
<i>Flow Diagrams</i>	486
<i>Design Details</i>	490
<i>PR Flow 1: Multi-Project PR Flow Using Partition Export/Import</i>	494
<i>PR Flow 2: Multi-Project PR Flow Using Keep Out Regions</i>	520
Updating ROM, BRAM, or LRAM Memory Initialization Contents in the Bitstream	553
Initialized using a mem_init file.....	553
Initialized using initd* parameters.....	555
Plugging in Custom IP Generator Tools	560
File Location and Directory Organization	561
<i>ACE Installation Directory</i>	561
<i>Loading User-defined Directory Paths using Environment Variables</i>	561
IP Generator Plugin Development.....	562
<i>IP Creation</i>	562
<i>IP Property Definitions</i>	563
<i>IP GUI Layout definition</i>	565
<i>Optional Setup for Built-in Output File Generator Functions</i>	568
<i>IP Callback Functions</i>	568
<i>Add IP to GUI</i>	579
Chapter 5 : Tcl Command Reference	580
SDC Commands.....	580
all_clocks.....	580

<i>Description</i>	580
<i>Example</i>	580
<i>Also See</i>	580
all_inputs	581
<i>Description</i>	581
<i>Example</i>	581
<i>Also See</i>	581
all_outputs	581
<i>Description</i>	581
<i>Example</i>	581
<i>Also See</i>	581
create_clock.....	582
<i>Description</i>	582
<i>Example</i>	582
<i>Also See</i>	583
create_generated_clock	583
<i>Description</i>	584
<i>Example</i>	584
<i>Also See</i>	584
get_cells	585
<i>Description</i>	585
<i>Example</i>	585
<i>Also See</i>	585
get_clocks	586
<i>Description</i>	586
<i>Example</i>	586
<i>Also See</i>	586

get_nets	587
<i>Description</i>	587
<i>Example</i>	587
<i>Also See</i>	587
get_pins	587
<i>Description</i>	588
<i>Example</i>	588
<i>Also See</i>	588
get_ports	588
<i>Description</i>	589
<i>Example</i>	589
<i>Also See</i>	589
set_clock_groups	589
<i>Description</i>	590
<i>Example</i>	590
<i>Also See</i>	590
set_clock_latency	591
<i>Description</i>	591
<i>Example</i>	591
<i>Also See</i>	592
set_clock_uncertainty	592
<i>Description</i>	592
<i>Example</i>	593
<i>Also See</i>	593
set_data_check	593
<i>Description</i>	594
<i>Example</i>	594

<i>Also See</i>	595
set_disable_timing	595
<i>Description</i>	595
<i>Example</i>	596
<i>Also See</i>	596
set_false_path.....	596
<i>Description</i>	596
<i>Example</i>	597
<i>Also See</i>	597
set_input_delay	598
<i>Description</i>	598
<i>Example</i>	598
<i>Also See</i>	599
set_input_transition.....	599
set_load	599
set_max_delay	600
set_min_delay	600
set_multicycle_path.....	600
<i>Description</i>	601
<i>Example</i>	602
<i>Also See</i>	602
set_output_delay.....	602
<i>Description</i>	603
<i>Example</i>	603
<i>Also See</i>	603
Interactive Timing Commands	603
check_setup.....	604

<i>Example</i>	605
prepare_sta	605
<i>Example</i>	606
report_checks.....	606
report_clock_properties	608
<i>Example</i>	609
reset_sta.....	609
<i>Example</i>	609
ACE Tcl Commands.....	610
add_clock_preroute	610
add_project_constraints.....	610
<i>Also See</i>	611
add_project_ip	611
add_project_netlist.....	612
add_project_source_files.....	612
add_region_find_insts	613
add_region_insts.....	614
apply_highlights.....	615
apply_placement	615
check_project_status	615
clean_project	616
clear_arcs.....	616
clear_drawing	616
clear_flow.....	617
clear_lines.....	617
clear_ovals.....	617

clear_polygons	617
clear_rectangles	617
clear_strings.....	618
clock_info.....	618
clock_relation	619
create_boundary_pins.....	620
create_equivalent_regions	620
create_flow_step.....	621
create_impl.....	622
create_option_sets.....	622
create_path	623
create_project	623
create_region.....	624
deselect.....	625
disable_flow_step	626
disable_project_constraints	626
disable_project_source_file	627
display_file.....	627
display_netlist.....	628
display_properties.....	628
draw_arc.....	628
draw_line.....	629
draw_oval	630
draw_polygon.....	631
draw_rectangle.....	632
draw_string.....	633

enable_flow_step.....	634
enable_project_constraints.....	634
enable_project_source_file.....	635
export_all_partitions.....	636
export_partition.....	636
filter.....	637
find.....	638
generate_ioring_design_files.....	641
generate_ip_design_files.....	641
generate_route_delay_table.....	641
get_ace_cputime.....	642
get_ace_current_memory_usage.....	642
get_ace_ext_dir.....	642
get_ace_ext_lib.....	642
get_ace_peak_memory_usage.....	642
get_ace_version.....	643
get_active_impl.....	643
<i>Example</i>	643
<i>Also See</i>	643
get_active_project.....	643
get_best_multiprocess_impl.....	644
get_clock_region_bounds.....	644
get_clock_regions.....	644
get_clock_type.....	644
get_compatible_ordering_codes.....	644
get_compatible_placements.....	645

get_current_design	645
get_current_partname	645
get_efd_file_path.....	645
get_enabled_constraints.....	646
get_enabled_source_files.....	646
get_fabricdb_path	647
get_file_line.....	647
get_flow_steps.....	647
get_impl_names.....	647
get_impl_option.....	648
get_impl_option_is_supported.....	648
get_impl_option_is_supported_and_enabled	649
get_inst_partition.....	649
get_inst_region.....	650
get_installation_directory	650
get_location.....	650
get_package_names.....	650
get_part_names	650
get_partition_changed	651
get_partition_force_changed.....	651
get_partition_info.....	651
get_partition_insts.....	652
get_partition_names.....	652
get_partition_timestamp.....	652
get_partition_type	652
get_path_ids.....	653

get_path_property	653
get_placement.....	654
get_project_constraint_files.....	654
get_project_directory	654
get_project_ip_files	654
get_project_names.....	655
get_project_netlist_files.....	655
get_project_option.....	655
get_project_option_is_supported.....	656
get_project_option_is_supported_and_enabled	656
get_project_output_directory.....	656
get_project_source_files.....	657
get_properties	657
get_property.....	658
get_pvt_corners	658
get_region_bounds.....	659
get_region_insts.....	659
get_regions.....	659
get_report_sweep_temperature_corners.....	659
get_selection.....	660
get_synprj_from_netlist.....	661
get_synprj_from_project	661
get_techlib_name	661
get_techlib_path	661
get_techlibdb_path	662
get_techlib_name.....	662

get_techlibt_path.....	662
get_techlibx_name	663
get_techlibx_path	663
get_worst_path.....	663
has_ace_ext_lib.....	664
has_partitions.....	664
highlight	664
ignore_cancel.....	665
import_synplify_project_file	665
initialize_flow	665
insert_delay	665
is_impl_option_enabled	665
is_incremental_compile.....	666
is_labmode.....	666
is_project_option_enabled	666
is_timing_met	667
load_flowscripts.....	667
load_project.....	667
message.....	668
move_project_constraints.....	668
move_project_netlists.....	669
move_project_source_file	669
move_relative_paths	670
optimize_tile.....	670
redirect	671
refresh_drawing.....	671

regenerate_all_ip_design_files.....	671
remap_partial_bitstream.....	672
remove_clock_preroute.....	672
remove_flow_step.....	673
remove_impl.....	673
remove_path.....	673
remove_project.....	674
remove_project_constraints.....	674
remove_project_constraints_pvt.....	674
remove_project_ip.....	675
remove_project_netlist.....	675
remove_project_source_files.....	675
remove_region.....	676
remove_region_insts.....	676
rename_impl.....	677
report_clock_regions.....	677
report_clocks.....	678
report_coverage.....	678
report_design_stats.....	679
report_impl_options.....	679
report_partitions.....	680
report_performance.....	681
report_pins.....	682
report_placement.....	682
report_power.....	683
report_project_options.....	684

report_routing.....	685
report_utilization.....	686
reset_impl_option.....	686
reset_project_option.....	687
restore_impl.....	687
restore_project.....	688
run.....	689
run_fanout_control.....	690
run_final_drc_checks.....	690
run_fpga_download.....	690
run_generate_bitstream.....	691
run_generate_final_reports.....	691
run_generate_fullchip_sim.....	691
run_generate_netlist.....	691
run_insert_holdbuffers.....	692
run_multiprocess.....	692
run_multiprocess_iterator.....	695
run_place.....	697
run_post_process.....	697
run_prepare.....	697
run_route.....	697
run_secureshare.....	698
<i>Why is this command taking so long?</i>	698
<i>Example</i>	699
run_simulation.....	700
run_snapshot.....	700

<i>Cleansing the .snapshot file</i>	701
run_synthesis	701
run_timing_analysis	702
run_tool	702
run_un_post_process	703
run_unplace	703
run_unroute	704
save_clock_preroute	704
save_impl	705
save_partition_placements	705
save_placement	706
save_project	707
save_properties	707
save_regions	708
select	708
set_active_impl	709
set_clock_type	709
set_cluster	710
set_equivalent_pins	710
set_flyline_direction	710
set_impl_option	711
set_max_flyline_fanout	711
set_partition_force_changed	711
set_partition_info	712
set_placement	713
set_project_constraints_pvt	713

set_project_option.....	714
set_property.....	714
set_region_bounds.....	715
set_region_type.....	716
set_units.....	716
sleep.....	716
source_encrypted.....	716
source_secure.....	717
trace_connections.....	717
untar.....	718
write_aeskey_efuse_bitstream.....	718
write_bitstream.....	719
write_critical_paths_script.....	720
write_netlist.....	720
write_partition_blackbox.....	721
write_partition_db.....	721
write_tcl_history.....	721
TCL Commands for the IP Generator Plugin Framework.....	722
add_ip_binary_property.....	723
<i>Description</i>	724
<i>Example</i>	724
add_ip_bool_property.....	724
<i>Description</i>	725
<i>Example</i>	725
add_ip_enum_property.....	725
<i>Description</i>	725

<i>Example</i>	725
add_ip_file_path_property.....	726
<i>Description</i>	726
<i>Example</i>	726
add_ip_float_property.....	726
<i>Description</i>	727
<i>Example</i>	727
add_ip_gui_control.....	727
<i>Description</i>	728
<i>Example</i>	728
add_ip_gui_group.....	728
<i>Description</i>	729
<i>Example</i>	729
add_ip_gui_page.....	729
<i>Description</i>	729
<i>Example</i>	729
add_ip_hex_property.....	730
<i>Description</i>	730
<i>Example</i>	730
add_ip_include_path.....	730
<i>Description</i>	731
<i>Example</i>	731
add_ip_int_property.....	731
<i>Description</i>	732
<i>Example</i>	732
add_ip_label.....	732
<i>Description</i>	732

<i>Example</i>	732
add_ip_output_file	733
<i>Description</i>	733
<i>Example</i>	733
add_ip_output_file_group	733
<i>Description</i>	734
<i>Example</i>	734
add_ip_port.....	734
<i>Description</i>	735
<i>Example</i>	735
add_ip_rule_message	735
<i>Description</i>	735
<i>Example</i>	735
add_ip_signal_name_property	736
<i>Description</i>	736
<i>Example</i>	736
add_ip_string_property	736
<i>Description</i>	737
<i>Example</i>	737
add_ip_to_gui	737
<i>Description</i>	737
<i>Example</i>	738
create_ip.....	738
<i>Description</i>	738
<i>Example</i>	738
get_ip_include_paths.....	738
<i>Description</i>	738

<i>Example</i>	739
get_ip_macro_name.....	739
<i>Description</i>	739
<i>Example</i>	739
get_ip_ports.....	739
<i>Description</i>	739
<i>Example</i>	739
get_ip_property_attributes	740
<i>Description</i>	740
<i>Example</i>	741
get_ip_property_value	742
<i>Description</i>	742
<i>Example</i>	743
remove_ip_gui_control.....	743
<i>Description</i>	743
<i>Example</i>	743
remove_ip_gui_group	743
<i>Description</i>	744
<i>Example</i>	744
remove_ip_gui_page	744
<i>Description</i>	744
<i>Example</i>	744
remove_ip_label	744
<i>Description</i>	745
<i>Example</i>	745
remove_ip_property.....	745
<i>Description</i>	745

<i>Example</i>	745
send_ip_definitions	745
<i>Description</i>	745
<i>Example</i>	745
send_ip_gui_update	746
<i>Description</i>	746
<i>Example</i>	746
set_ip_category	746
<i>Description</i>	746
<i>Example</i>	746
set_ip_description	746
<i>Description</i>	747
<i>Example</i>	747
set_ip_macro_name.....	747
<i>Description</i>	747
<i>Example</i>	747
set_ip_property_value	747
<i>Description</i>	748
<i>Example</i>	748
set_ip_property_verilog_param_attributes	748
<i>Description</i>	749
<i>Example</i>	749
set_ip_resource_count.....	749
<i>Description</i>	749
<i>Example</i>	749
set_ip_resource_types	749
<i>Description</i>	750

<i>Example</i>	750
set_update_ip_configuration_callback.....	750
<i>Description</i>	750
<i>Example</i>	751
update_ip_binary_property.....	754
<i>Description</i>	754
<i>Example</i>	754
update_ip_bool_property.....	755
<i>Description</i>	755
<i>Example</i>	755
update_ip_configuration.....	755
<i>Description</i>	755
<i>Example</i>	756
update_ip_enum_property.....	756
<i>Description</i>	756
<i>Example</i>	756
update_ip_file_path_property.....	756
<i>Description</i>	757
<i>Example</i>	757
update_ip_float_property.....	757
<i>Description</i>	757
<i>Example</i>	757
update_ip_hex_property.....	757
<i>Description</i>	758
<i>Example</i>	758
update_ip_int_property.....	758
<i>Description</i>	758

<i>Example</i>	758
update_ip_label	759
<i>Description</i>	759
<i>Example</i>	759
update_ip_signal_name_property.....	759
<i>Description</i>	759
<i>Example</i>	759
update_ip_string_property.....	760
<i>Description</i>	760
<i>Example</i>	760

Chapter 6 : Troubleshooting..... 761

ACE Exit Error Codes.....	761
Duplicate Names for Arrays.....	763
Clock Definitions/Constraints.....	763
Asynchronous Reset of I/O from the Core	763
Multi-process Functionality License Requirements.....	763
Non-ASCII Characters in Path.....	764
Unable to Load Project: Project is Locked	764
Changing ACE Font Sizes	765
Fonts in Views.....	765
<i>Linux</i> :.....	766
<i>Windows</i> :	766
Fonts in HTML Reports	766
Unable to Initialize Reserved Module Name List.....	766

Startup Error — ACE is Unable to Connect on Port NNNN of Localhost.....	767
To Determine the Root Cause	768
<i>If it is a Firewall Problem</i>	768
<i>If it is a Licensing Problem</i>	769
<i>If it is a Filesystem or Network Performance Problem</i>	770
Unexpected ACE GUI problem: java.lang.NoClassDefFoundError: com/achronix/api/APIPlugin.....	770
If the problem persists after -clean, or if the command prompt is unfamiliar	771
<i>Detailed instructions:</i>	771
Multiprocess Summary Report Shows "No Timing Results Found" for Successfully Run Implementations with Existing Timing Reports.....	772
Windows: ACE Incorrectly Reports Read/Write File Permission Problems	772
Windows: ACE GUI Shown as "Not Responding"	772
Windows: Garbage sometimes appears in the Floorplanner View during panning operations (and remains after panning is completed)	773
Windows: ACE Startup Error Due to Missing DLL Component in Windows 10	773
Windows: The icons and buttons in ACE are too small.....	774
Asking Windows to upscale images and fonts for all applications.....	774
Asking just ACE to upscale images and fonts.....	774
<i>Asking Windows to alter the scaling settings for just ACE (which may make text and fonts blurry/blocky)</i>	774
<i>Enabling ACE's application framework's fractional scaling (fonts remain crisp, but icons may become blurry and ACE may experience instability with some video drivers)</i>	776
Linux: Resource Limits: ACE Reports an OutOfMemory Error, But There is Plenty of Free Memory Available.....	776
Linux: In the TWM Window Manager, the First Time the ACE GUI is Started After Installation, the ACE Window is So Small Users Might Not See it.....	777

Linux: Odd Behavior When Using X DISPLAY Forwarding if the X Client and X Server Are More than One Major Revision Apart.....	777
Linux: ACE Menus Do Not Show Icons Next to the Action Names	778
Linux: ACE Ignores LD_LIBRARY_PATH.....	778
Linux: ACE forces the use of the GDK X11 rendering engine even while running under Wayland.....	778
Linux: Incompatible Default Web Browser.....	779
Solution	780
<i>Details</i>	780
Additional Information.....	781
<i>RHEL/CentOS7</i>	781
<i>RHEL8</i>	782
<i>WebKit and WebKitGTK Technical Details</i>	782
<i>Other Linux Distros</i>	782
Linux: ACE Requires an Unusually Large Amount of Virtual Memory (Due to WebKit2)	783
Linux: ACE forces the use of the Adwaita GTK+3 Theme; Can I Change This?	783
Themes.....	783
<i>Changing the GTK+3 Theme</i>	784
Animations and Other Effects.....	785
Linux: Views and Editors Detach when Dragged Instead of Docking in the Workbench.....	785
Linux: CDE: Dialogs and Wizards Sometimes Appear Behind the Main ACE Window, Especially After Minimize/Maximize.....	786
Linux: "Failed to create the part's controls": Some Views and IP Editors may fail to initialize	786

Upgrading an ACE Installation.....	787
On Windows	787
On Linux	787
GUI Problems after Upgrading?	788
Chapter 7 : Revision History for ACE.....	789
Revision History	789

Chapter 1 : Preface

Preface

About This Guide

This guide is a reference manual for ACE, used for placing, routing, configuring, and debugging Achronix FPGAs. ACE works in conjunction with third-party synthesis and simulation tools to provide a complete design environment for Achronix FPGAs.

This guide consists of the following chapters:

- **Getting Started** (page 3) includes an Introduction to ACE and a quick Tutorial.
- **Concepts** (page 6) covers all the basic concepts of ACE, and can be considered a reference manual for the various GUI elements.
- **Tasks** (page 282) details how to complete various tasks within the GUI, plus provides the related TCL commands.
- **TCL Command Reference** (page 580) provides a complete TCL command reference, including syntax.
- **Troubleshooting** (page 761) shows a number of common problems and the recommended solutions.
- **Revision History** (page 789) lists the changes to each revision of this document

Related Documents

The following documents, as well as the latest version of this document (UG070), are always available for download at <https://www.achronix.com/support/docs>.

- *ACE Installation and Licensing Guide* (UG002)
- *JTAG Configuration User Guide* (UG004)
- *Snapshot User Guide* (UG016)
- *Synthesis User Guide* (UG018)

The following supplemental document, typically included in the software release download files, should also be consulted for the very latest information:

- *ACE (version) Release Notes*

Further documents are available for each Speedster device on both <https://www.achronix.com/support/docs> and <https://download.achronix.com> (login required).

Also see the Getting Started section on <https://support.achronix.com>.

Please consult your Achronix FAE for a complete list of documentation relevant to your Achronix products.

Conventions Used in this Guide

Item	Format	Examples
Command-line entries	Formatted with a bold fixed-width font, or in a special code block.	<pre>\$ Open top_level_name.log</pre> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Command-line code example</p> <pre>\$ Open top_level_name.log</pre> </div>
File names	Formatted with a fixed-width font.	<code>filename.ext</code>
GUI buttons, menus, menu or list choices, and radio buttons	Formatted with a variable-width bold font.	Select File → Open , select the desired file, then click OK to continue.
Variables	Formatted with italic emphasis and enclosed by the angle brackets "<>".	<code><design_dir> /output.log</code>
RTL Names	Formatted with italic emphasis.	<code>read_clk</code>
Window and dialog box headings and sub-headings	Heading formatted in quotation marks.	Under "Output Files", select ...
Window and dialog box names	Name uses initial caps.	From the Add Files dialog box, ...

Chapter 2 : Getting Started


Introduction

The Achronix implementation flow is an end-to-end solution which provides an intuitive user interface, batch mode support, advanced debug capabilities, and a simple push-button set of high-level flow steps:

- IP Configuration and Generation
- Simulation (uses standard 3rd party simulator tools)
- Synthesis (uses the industry standard Synopsys Synplify Pro)
- Placement
- Routing
- Timing Analysis
- Bitstream Generation
- FPGA Configuration
- On-chip Debugging

ACE Quickstart Tutorial

Start by copying all the files from `<ace_install_dir>/examples/quickstart/<device>` into a new empty directory (`<test_dir>`) in your file system. Use the `<device>` directory that matches the **Target Device** implementation option that you select in step 2. Before launching ACE, make sure you have also installed Synplify Pro and your Simulation tools, and configured the environment variables (such as `$ACX_SYNPLIFY_TOOL_PATH` and `$ACX_<sim_tool>_TOOL_PATH`) by following the instructions in the *ACE Installation and Licensing Guide* (UG002), or *Getting Started User Guide*.

Now click the () minimize icon in the upper right corner of the Welcome view to minimize these instructions. Follow these simple steps to complete your first design in ACE:

1. Create your Project





In the **Projects View** (page 117), click the () **Create Project** toolbar button. Follow these steps to create the project:

1. In the **Create Project Dialog** (page 170), enter (or browse to) the path to `<test_dir>` in the Project Directory field.
2. Enter `quickstart` in the **Project Name** field and click **OK**.

You should now see your new project show up in the Projects view. See **Creating Projects** (page 293) or **Working with Projects and Implementations** (page 293) for more details.

2. Add your Design Files and Set Implementation Options

In the Projects view, click the "quickstart" project to select it. Follow these steps to add the design source files for Simulation, Synthesis, and Place and Route:


1. Click the () **Add Source Files** toolbar button and select **Add RTL Files**.
2. In the Add RTL Files dialog, browse to the `<test_dir>/src/rtl` directory and select all of the files by holding down the **CTRL** key and clicking each file name.
3. Click the **Open** button to add the RTL files to your project.
4. Click the () **Add Source Files** toolbar button and select **Add Synthesis Constraint Files**.
5. In the Add Synthesis Constraint Files dialog, browse to the `<test_dir>/src/constraints` directory and click the `quickstart.sdc` file to select it.
6. Click the **Open** button to add the Synthesis Constraint files to your project.
7. Click the () **Add Source Files** toolbar button and select **Add Place and Route Constraint Files**.
8. In the Add Place and Route Constraint Files dialog, browse to the `<test_dir>/src/constraints` directory and select all of the files by holding down the **CTRL** key and clicking each file name.
9. Click the **Open** button to add the place and route constraint files to your project.
10. Click the () **Add Source Files** toolbar button and select **Add Simulation Testbench Files**.
11. In the Add Simulation Testbench Files dialog, browse to the `<test_dir>/src/tb` directory and select all of the files by holding down the **CTRL** key and clicking each file name.
12. Click the **Open** button to add the simulation testbench files to your project.

In the Options View, follow these steps to configure your project options:

1. Expand the **Project Options** section and select the **Target Device** that matches the set of design files that you copied earlier.
2. In the **Project Options** section, scroll down and enter the following semicolon-separated list for the **HDL Include Path**:
 - `<test_dir>/src/rtl;<test_dir>/src/tb`**Note:** The **HDL Include Path** applies to both **Synthesis** and **Simulation**.
3. Scroll down and expand the **Simulation** section of options.
4. Select the **Simulation Tool** you have installed from the drop-down list and enter `tb_quickstart` for the **Testbench Top Module**.



You now have a project that is ready to run through the flow! See [Adding Source Files \(page 297\)](#) or [Working with Projects and Implementations \(page 293\)](#) for more details.


3. Run the Flow

1. In the Flow view, expand the tree to view the detailed flow steps.
2. Under **RTL Simulation**, click the checkbox next to **Run RTL Simulation**.
3. Click the () **Run Flow** toolbar button.

Textual output from the flow is shown in the Tcl Console view. When the flow is finished running, you can see the completed flow steps in the Flow view updated with a green check mark (✓) to indicate success, and all newly generated reports are displayed in the editor area. See the [Flow \(page 234\)](#) concept or [Running the Flow \(page 306\)](#) for more details.

4. Analyze the Results

1. On the main toolbar, click the () **Floorplanner Perspective** toolbar button.
2. Within this perspective, use the [Critical Paths View \(page 37\)](#) to analyze critical paths and highlight them in the [Floorplanner View \(page 43\)](#). Clicking the () **Zoom To Path** toolbar button in the Critical Paths view, zooms the [Floorplanner View \(page 43\)](#) to the path currently selected in the Critical Paths view.
3. Use the [Search View \(page 129\)](#) and [Selection View \(page 133\)](#) to locate objects of interest.

Clicking the () **Zoom To Selection** toolbar button in the Selection view zooms the Floorplanner view to the objects in the current selection set. See the [Viewing the Floorplanner \(page 339\)](#) and [Analyzing Critical Paths \(page 353\)](#) tasks in the User Guide for more details.

Congratulations!!!



You have successfully completed a design in ACE!

Chapter 3 : Concepts

Workbench

The term "workbench" refers to the desktop development environment within ACE. The workbench aims to achieve seamless tool integration by providing a common platform for the creation, management, and navigation of project resources.

Each workbench window contains one or more [perspectives \(page 6\)](#). Perspectives contain [views \(page 16\)](#) and [editors \(page 9\)](#) and control what appears in certain menus and tool bars. More than one workbench window can exist on the desktop at any given time.

Perspectives


There are many different kinds of information that must be viewed within ACE. Perspectives are used to filter the information into usable, logically consistent groupings. A perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. A perspective defines the initial set and layout of [views \(page 16\)](#), [editors \(page 9\)](#), menus, and toolbars in the [Workbench \(page 6\)](#) window.

For example, the Projects perspective combines [views \(page 16\)](#) commonly used while managing project source files, while the Floorplanner perspective contains the views that are used while viewing chip layout and floorplanning information. Perspectives are frequently switched while working inside the [Workbench \(page 6\)](#).

Note

Within the [Workbench \(page 6\)](#) window, all perspectives share the same set of [editors \(page 9\)](#). All editors are usable/visible from all perspectives. Likewise, each of the [views \(page 16\)](#) may optionally be used within any perspective, but they are most useful when grouped with the other views from their native perspective. One of the views, the [Tcl console view \(page 142\)](#), is a member of all the perspectives.


Projects Perspective

The () Projects perspective allows selecting an active project and implementation, managing the contents and configuration of the active project/implementation, running the [flow \(page 234\)](#), and viewing the reports generated by the flow.

By default, this perspective contains the [Projects view \(page 117\)](#), [Flow view \(page 53\)](#), [Options view \(page 96\)](#), [Tcl console view \(page 142\)](#), and the editor area, which can contain any ACE editor or report. The [Multiprocess view \(page 73\)](#) is also part of this perspective, but is hidden by default.

For more information, see [Working with Projects \(page 293\)](#), [Running the Flow \(page 306\)](#), and [Using the Tcl Console \(page 332\)](#).

Floorplanner Perspective

The () Floorplanner perspective allows viewing and editing the placement and routing of the active project/implementation.

By default, this perspective contains the following:


- [Floorplanner View \(page 43\)](#)
- [Search View \(page 129\)](#)
- [Selection View \(page 133\)](#)
- [Critical Paths View \(page 37\)](#)
- [Critical Path Diagram View \(page 33\)](#)
- [Netlist Browser View \(page 79\)](#)
- [Clock Domains View \(page 20\)](#)
- [Clock Regions View \(page 23\)](#)
- [Placement Regions View \(page 110\)](#)
- [Partitions View \(page 107\)](#)
- [Tcl Console View \(page 142\)](#)

For more information on using the views in this perspective, see [Viewing the Floorplanner \(page 339\)](#), [Pre-Placing a Design \(page 346\)](#), and [Analyzing Critical Paths \(page 353\)](#).

Caution!

Unlike most other perspectives, the Floorplanner perspective hides the editor area. To view [editors \(page 9\)](#) and [reports \(page 244\)](#), a different perspective must be selected.

IP Configuration Perspective

The () IP configuration perspective is used to create and edit IP configuration files (.acxip) through the various IP configuration editors.


By default, this perspective contains the following:

- [Projects View \(page 117\)](#)
- [IP Libraries View \(page 71\)](#)
- [IP Diagram View \(page 69\)](#)
- [IP Problems View \(page 71\)](#)
- [Outline View \(page 106\)](#)
- [Tcl Console View \(page 142\)](#)
- [I/O Designer Toolkit Views \(page 60\)](#)

The IP configuration perspective also contains the editor area, which can contain any ACE editor or report.

See [Creating an IP Configuration \(page 336\)](#) for more details.

2D NoC Performance Perspective

The () 2D NoC Performance perspective is used to visualize the throughput or congestion of the 2D NoC network by loading a simulation log file produced by the device simulation model (DSM).


By default, this perspective contains the following:

- [NoC Performance View \(page 87\)](#)
- [NoC Time Slice View \(page 93\)](#)

Caution!

Unlike most other perspectives, the 2D NoC performance perspective hides the editor area. To view [editors \(page 9\)](#) and [reports \(page 244\)](#), a different perspective must be selected.

Programming and Debug Perspective

The () Programming and Debug perspective allows interaction with Achronix FPGAs via JTAG through a JTAG pod or embedded JTAG controller device. Downloading the device configuration and debugging is typically performed within this perspective.


By default, this perspective contains the following:

- [Snapshot Debugger View \(page 137\)](#)
- [Download View \(page 40\)](#)
- [Register Browser View \(page 127\)](#)
- [Tcl Console View \(page 142\)](#)

The Programming and Debug perspective also contains the editor area, which can contain any ACE editor or report.

For more information on using this perspective, see [Running the Snapshot Debugger \(page 364\)](#) and [Programming a Device using JTAG in the Download View \(page 390\)](#)

HW Demo Perspective

The () HW Demo perspective allows observing various aspects of a particular device, by selecting one of the provided demonstration designs from a list. When the demonstration is loaded into the attached board, LED states and DIP switch states (from the board) are displayed and updated in real-time. Internal device state information such as the temperature of the FPGA and power consumption are also displayed.

By default, this perspective contains the following:

- [Snapshot Debugger View \(page 137\)](#)
- [HW Demo View \(page 58\)](#)
- [Tcl Console View \(page 142\)](#)

The HW Demo perspective also contains the editor area, which can contain any ACE editor or report.

For more information on using this perspective, see [Running the HW Demo \(page 401\)](#).

Editors

Most [perspectives \(page 6\)](#) in the [workbench \(page 6\)](#) are comprised of an editor area and one or more [views \(page 16\)](#). Different editors are associated with different types of files. For example, when a file is opened by double-clicking in the [Projects view \(page 117\)](#), the associated editor opens in the Workbench. If there is no associated editor for a resource, the Workbench attempts to launch an external editor outside the Workbench. Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor.

Tabs in the editor area indicate the names of resources that are currently open for editing (usually the filename, and the tab tooltip provides the full path to the file). An asterisk (*) displayed in an editor tab indicates that an editor has unsaved changes. By default, editors are stacked in the editor area, but may be [tiled \(page 289\)](#) in order to view multiple editors simultaneously. The gray border at the left margin of the editor area may contain icons that flag errors, warnings, or problems detected by the system.

In ACE, the editor area is also used to view the [reports \(page 244\)](#) generated by ACE. By default, when ACE is running the [flow \(page 234\)](#) in single-process mode, ACE opens HTML versions of the reports in the [HTML report browser \(page 9\)](#) as soon as the report data is generated/updated. When ACE is in Multiprocess mode (via the [Multiprocess view \(page 73\)](#)), only the [Multiprocess summary report \(page 255\)](#) is automatically opened in the editor area — the other reports must be opened manually through the [Projects view \(page 117\)](#), or by following the Timing report hyperlinks for each [implementation \(page 229\)](#) found within the Multiprocess summary report.

ACE also provides a suite of IP Configuration editors, organized by fabric family/library, used to instantiate and configure the various IP surrounding the core fabric. See [Creating an IP Configuration \(page 336\)](#).

Note

The IP libraries and IP types displayed within ACE are dynamic and change based on which technology libraries and devices are installed and licensed at each customer site. Therefore, the individual IP configuration editors are not documented in this guide, and are instead documented in separate dedicated Achronix documentation. An example would be the *Speedster7t Soft IP User Guide (UG103)*.

HTML Report Browser

When HTML versions of generated [reports \(page 244\)](#) are opened within ACE, they are displayed within the Editor area using the HTML Report browser. This is a very limited form of a web browser — it only allows hyperlink traversal, refresh, forward, and back operations. The buttons for refresh, back, and forward are not displayed within the browser itself, but are instead shown in the main (topmost) ACE button-bar.

Note

The HTML Report Browser should not be used to browse the Internet — a dedicated web browser (i.e., Firefox) would be a much better choice, for both security and performance reasons.

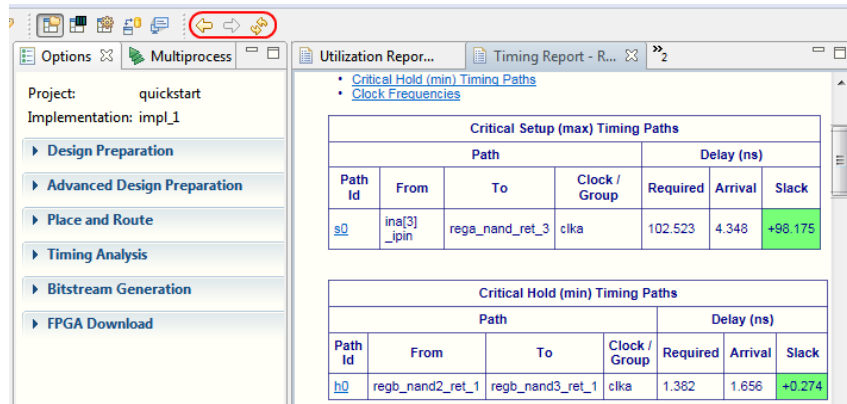


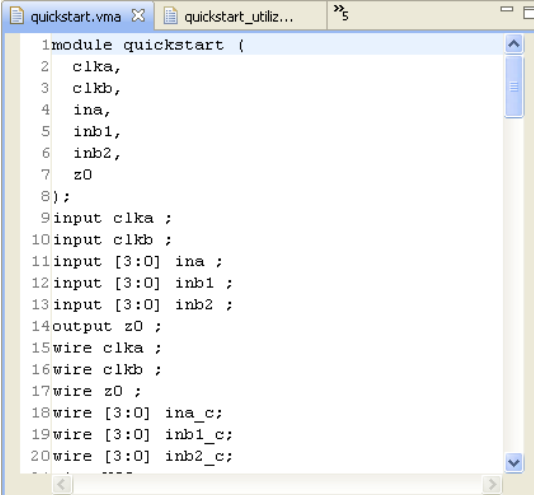
Figure 1 · HTML Report Browser, Toolbar Buttons

Table 1 · HTML Report Browser Toolbar Buttons

Icon	Action	Description
	Back	Returns to the last HTML location viewed.
	Forward	Returns to the HTML location viewed before the Back button was selected (the Forward button remains disabled until the Back button has been clicked).
	Refresh	Refreshes the displayed HTML report to show the current contents of the report file on disk.

Text Editor

Reports, source files, and scripts open in the text editor. The text editor supports typical editing functions, such as insert, delete, copy, cut, and paste.



```
1 module quickstart (
2   clk_a,
3   clk_b,
4   ina,
5   inb1,
6   inb2,
7   z0
8 );
9 input clk_a ;
10 input clk_b ;
11 input [3:0] ina ;
12 input [3:0] inb1 ;
13 input [3:0] inb2 ;
14 output z0 ;
15 wire clk_a ;
16 wire clk_b ;
17 wire z0 ;
18 wire [3:0] ina_c;
19 wire [3:0] inb1_c;
20 wire [3:0] inb2_c;
```

Figure 2 • Text Editor Example

VCD Waveform Editor

The VCD waveform editor does not allow editing a VCD file. It only allows viewing. But since it resides in the same location in the GUI as all the other **editors** (page 9), and it opens whenever a VCD file is selected, it can be thought of as an editor in read-only mode.

The waveform viewer allows examining VCD output in a familiar waveform visualization, displaying how signals change values over time. It is typically used to examine the VCD output that gets generated when **running the Snapshot debugger** (page 364) (see also: **Snapshot Debugger View** (page 137), **Viewing the Captured VCD Waveform** (page 378)).

As with familiar waveform editors, the placement of a vertical line marker (pink by default, managed by a user preference) can be manipulated with the mouse in the graphical waveform area so that the value of all signals at the same instant of time can be seen. Signals can also be re-ordered (individually moved vertically among peers in the table using buttons, or using drag-and-drop) and individual signals can be hidden or shown. Buses are collapsed by default (individual bits of the bus are hidden), but may be expanded to show all the bits, or re-collapsed to just the bus as desired. There are also buttons available to expand all buses at once, and collapse all buses at once. If necessary, bit and bus signals (but not the individual bits of a bus) may be duplicated in the display so that they can be displayed adjacent to multiple peers for ease of value comparisons. It is, of course, possible to change the zoom level of the graphical waveform area if desired using buttons or the mouse wheel in the waveform area.

For each VCD file, the editor remembers signal name ordering, zoom level, and the sample offset. These settings are remembered between Snapshot captures within a single session, as well as between ACE sessions. A button is also provided to return the file to defaults for all of these values at any time.

Note

1. In addition to the graphical waveform view, the raw text content of the VCD file may be viewed. To do so, select the **File Preview** tab at the bottom-left of the VCD waveform editor. To see the graphical waveform representation again, select the **Waveform** tab, also at the bottom-left of the VCD waveform editor.
2. None of the actions available in the VCD waveform editor change the content stored in the VCD file.

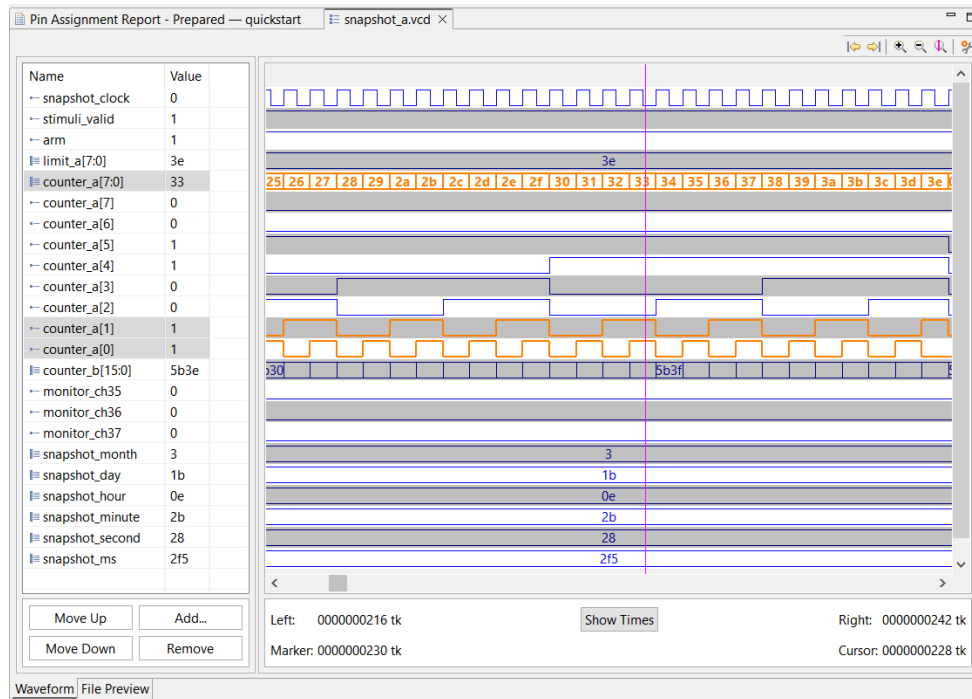


Figure 3 • VCD Waveform Editor Example

By default, the waveform area uses alternating foreground colors (black and blue by default) and background colors (light grey and white by default) for even and odd rows. When rows are selected in the signal name/value table, the corresponding rows in the waveform are highlighted with a specified selection foreground color (orange by default). Multi-bit bus values are also shown in hexadecimal in the waveform area when they fit in the available space. When the bus value is too wide for the available area, the value is not shown in the waveform bus row, though value edges are still visible. The available area may be made wider by zooming in, thus allowing larger bus values to be displayed (the marker can also be used to display the bus and bit value in the signal table Value column for a chosen point in time).

Table 2 • VCD Waveform Editor Options

Option	Description
Signal Value Table	
Name	The name of the signal as stored in the VCD file.
Value	The value of the signal at the indicated point in time of the marker.
Waveform Timing Info	
Left	The time in picoseconds (ps) or ticks (tk) indicated by the left edge of the viewable waveform area.
Right	The time (in ps or tk) indicated by the right edge of the viewable waveform area.
Marker	The time (in ps or tk) indicated by the vertical marker line (pink by default) shown in the viewable waveform area.
Cursor	The time (in ps or tk) indicated by the current (or last relevant) mouse cursor position over the viewable waveform area.

Note

Cursor Values – when the mouse moves away from the waveform area, the last position over the waveform is retained by the **Cursor** value. This (ps or tk) value does not change until the mouse cursor is again over the waveform, even if the view is scrolled or the zoom factor is changed.

Icons are shown to the left of each signal name as an indicator whether the name corresponds to a individual bit signal, a bus, or a bit within a bus.

Table 3 • VCD Waveform Editor Icons

Icon	Description
	Signal.
	Bus.
	Bus bit.



There are a number of buttons available below the signal name/value table area to allow altering which signals/ buses are visible, and where they are displayed in relation to their peers. Signals and buses are allowed to be displayed more than once within the table, if desired.


Table 4 • Signal Value Table Granular Actions

Action	Button	Context Menu	Description
Move Up	Y		Moves the currently-selected signals (or buses) one row higher in the table and the waveform area. Signals may also be dragged vertically to new locations with the mouse.
Move Down	Y		Moves the currently-selected signals (or buses) one row lower in the table and the waveform area. Signals may also be dragged vertically to new locations with the mouse.
Add...	Y		Opens the Add Signals to Waveform Viewer Dialog (page 146), which allows previously removed (hidden) signals to be displayed, and allows already-visible signals to be added to the signal list multiple times. (A signal could be duplicated and shown adjacent to multiple associated signals.)
Remove	Y	Y	Hides the currently-selected signal(s) (or bus(es), if any bus or any of its bus bits are selected), temporarily removing it/them from the table and the waveform area. The hidden signal/bus may be shown (added to the table and waveform) again via the Add... button.
Copy Signal Name		Y	When a single bus or signal is selected, this right-click context menu choice allows copying the current name of the selected bus or signal (this menu choice is not shown when multiple rows are selected in the table).
Copy Signal Value		Y	When a single bus or signal is selected, this right-click context menu choice allows copying the current value of the selected bus or signal (this menu choice is not shown when multiple rows are selected in the table).
Expand Bus		Y	This option (only visible when a single bus is selected) expands the bus to show all the bits making up the bus. The bus bits are shown below the bus, always in descending index order.
Collapse Bus		Y	This option (only visible when a single bus or single bus bit is selected) collapses the bus, hiding all the bits making up that bus.

There are also some supplemental buttons above the signal value table with a few additional actions, each of which impact the entire table contents at once.







Table 5 • Signal Value Table Bulk Actions

Icon	Action	Description
	Reset Signal Table	Resets the signals table content and order to the state when the file is first loaded. This is expected to generally match the order of the monitored bits as found in the *.snapshot file, with all buses collapsed.
	Expand All Buses	Expands all buses so that all bits are visible.

Icon	Action	Description
	Collapse All Buses	Collapses all buses so that zero bits are visible.


There are also a number of buttons that affect the waveform area in particular.

Table 6 - Waveform Buttons

Icon	Action	Description
	Move Marker to Previous Edge	Moves the vertical pink marker line in the waveform area to the previous edge for the signal/bus which is currently selected in the signal value table (has no effect when no signal is selected in the table).
	Move Marker to Next Edge	Moves the vertical pink marker line in the waveform area to the next edge for the signal/bus which is currently selected in the signal value table (has no effect when no signal is selected in the table).
	Zoom In	Increases the zoom factor in the waveform area, increasing the visible level of detail. This is also possible using the mouse wheel when the mouse cursor is over the waveform area.
	Zoom Out	Decreases the zoom factor in the waveform area, decreasing the visible level of detail. This is also possible using the mouse wheel when the mouse cursor is over the waveform area.
	Zoom to Marker Position	Without changing the zoom factor, scrolls the waveform area horizontally to make the marker visible.
	Configure Colors...	Opens the Preferences (page 188) dialog, switches to the General → Appearance → Colors and Fonts page, and scrolls to the VCD Editor section, allowing the colors and fonts used to be changed within the waveform area.
	Show Ticks/Show Times	Toggles the Waveform Timing info (left, right, marker, and cursor) between displaying values in ticks (tk) or picoseconds (ps).

As is typical, the scrollbars around the waveform area can be used to move the waveform area vertically (to see other signals) or horizontally (to see the same signals at different points in time). In most cases in Linux, scrolling the mouse wheel specifically over the scrollbars also moves the scrollbar in the expected direction, though this function may not be possible with all themes, desktop environments, and/or window managers.

VCD Viewer Preferences

The colors and fonts used within the VCD waveform viewer (and the font used within the Signal Value table) are managed by user-editable preferences. These preferences can easily be viewed and edited by selecting the  **Configure Colors...** button found to the upper-right of the waveform. These preferences allow selecting alternate waveform viewer color choices (for example, green-on-black colors similar to a more traditional oscilloscope). When fine-tuning colors, remember to use the **Apply** button near the bottom of the Colors and Fonts page in the Preference dialog so that choices can be tested immediately without closing the dialog, allowing immediate visual

feedback and continued iteration (in most desktop configurations, the Preferences dialog can be moved and made taller/wider to ensure visibility of all preferences without scrolling).

The following figure shows an example where most of the default colors have been altered in favor of a light-on-dark color configuration.

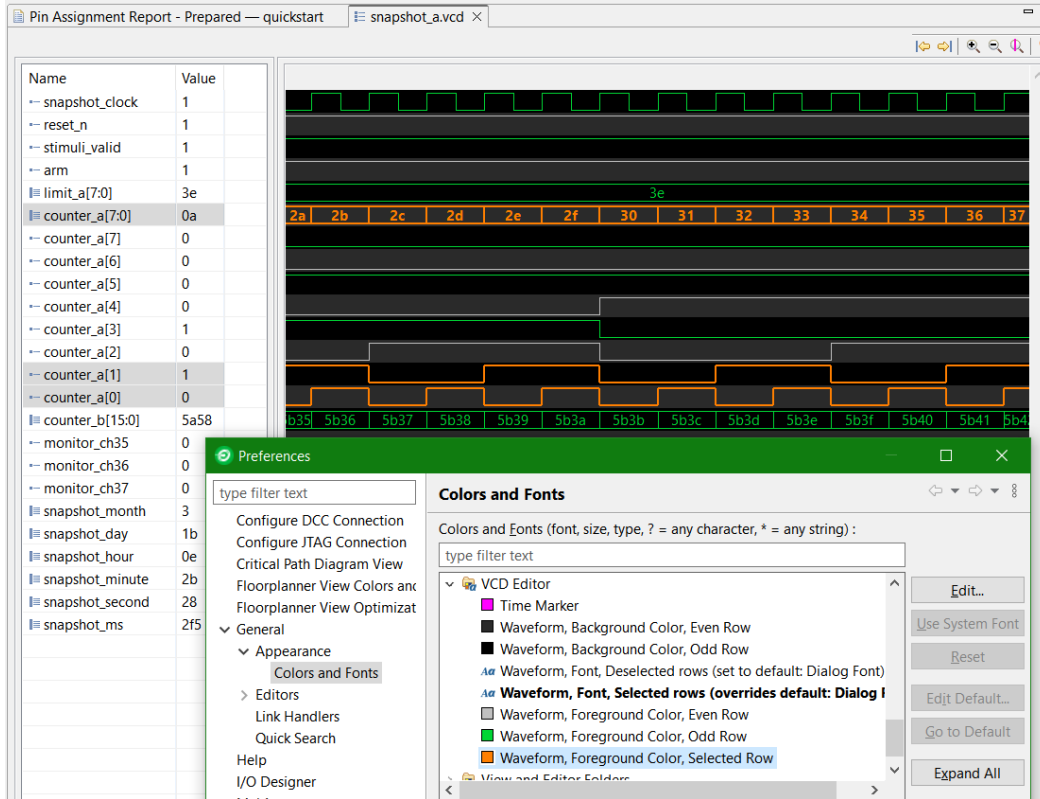


Figure 4 • VCD Waveform Color and Font Preferences

Views

Views support **editors** (page 9) and provide alternative presentations as well as ways to navigate the information in the **workbench** (page 6). For example, the **Projects view** (page 117) displays **projects** (page 222), **implementations** (page 229), and their related file-based resources.

All views have their own context menu showing ways to alter the location or presentation of the view. Simply right-click the view tab to display the menu.

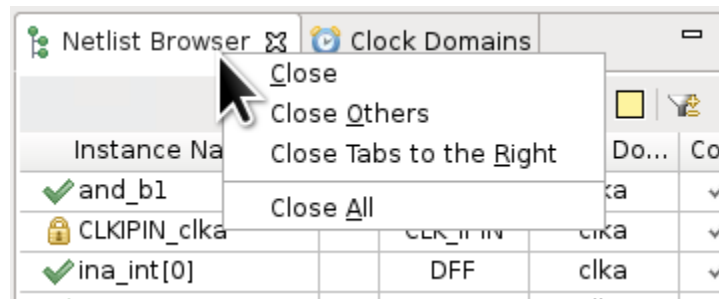


Figure 5 • Example of View Right-Click Context Menu

Some views have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view.

Some views also have their own menus to affect the content of the view. These menus often contain the same actions as in the toolbar, potentially an additional link to the relevant page(s) in the [preferences \(page 188\)](#) which are specific to this view, as well as additional actions/toggles specific to the view which are used less frequently. When such a menu is available, a () vertical ellipsis button appears at the far right of the view toolbar. To open the menu for a view, click the ellipsis button.

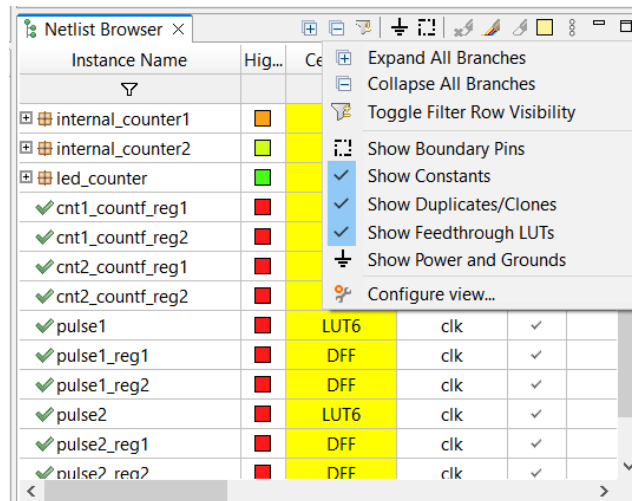


Figure 6 • Example of View Toolbar Including Opened View Menu

Views are grouped by shared context into [perspectives \(page 6\)](#). Within a perspective, a view might appear by itself, or stacked with other views in a tabbed notebook. The layout of a perspective can be changed by opening and closing views and by moving/docking them in different positions in the [Workbench \(page 6\)](#) window. See [Working with Views and Editors \(page 287\)](#) for more information.

The views contained by the Project perspective are:

- [Projects View \(page 117\)](#)
- [Flow View \(page 53\)](#)

- [Options View \(page 96\)](#)
- [Multiprocess View \(page 73\)](#)
- [Tcl Console View \(page 142\)](#)

The views within the Floorplanner perspective are:

- [Search View \(page 129\)](#)
- [Selection View \(page 133\)](#)
- [Critical Paths View \(page 37\)](#)
- [Critical Path Diagram View \(page 33\)](#)
- [Floorplanner View \(page 43\)](#)
- [Clock Regions View \(page 23\)](#)
- [Clock Domains View \(page 20\)](#)
- [Clusters View \(page 28\)](#)
- [Placement Regions View \(page 110\)](#)
- [Partitions View \(page 107\)](#)
- [Netlist Browser View \(page 79\)](#)
- [Properties View \(page 123\)](#)
- [Tcl Console View \(page 142\)](#)

The views contained within the IP Configuration perspective are as follows, with the subset specific to Speedster7t FPGAs (with I/O rings) also summarized at [I/O Designer Toolkit Views \(page 60\)](#):

- [Projects View \(page 117\)](#)
- [Outline View \(page 106\)](#)
- [IP Libraries View \(page 71\)](#)
- [IP Diagram View \(page 69\)](#)
- [IP Problems View \(page 71\)](#)
- [I/O Utilization View \(page 62\)](#)
- [I/O Package Diagram View \(page 63\)](#)
- [I/O Pin Assignment View \(page 64\)](#)
- [I/O Core Pin Assignment View \(page 65\)](#)
- [I/O Layout Diagram View \(page 67\)](#)
- [Tcl Console View \(page 142\)](#)

The views contained within the 2D NoC perspective are:

- [NoC Performance View \(page 87\)](#)
- [NoC Time Slice View \(page 93\)](#)

The views of the Programming and Debug perspective are:

- [Snapshot Debugger View \(page 137\)](#)
- [Download View \(page 40\)](#)
- [Register Browser View \(page 127\)](#)

- [Tcl Console View \(page 142\)](#)

The views of the HW Demo perspective are:

- [HW Demo View \(page 58\)](#)
- [Snapshot Debugger View \(page 137\)](#)
- [Tcl Console View \(page 142\)](#)

Finally, for ease of reference, a combined summary list of all available ACE views from all perspectives, sorted in alphabetical order:

- [Clock Domains View \(page 20\)](#)
- [Clock Regions View \(page 23\)](#)
- [Clusters View \(page 28\)](#)
- [Critical Path Diagram View \(page 33\)](#)
- [Critical Paths View \(page 37\)](#)
- [Download View \(page 40\)](#)
- [Floorplanner View \(page 43\)](#)
- [Flow View \(page 53\)](#)
- [HW Demo View \(page 58\)](#)
- [I/O Designer Toolkit Views \(page 60\)](#)
 - [I/O Core Pin Assignment View \(page 65\)](#)
 - [I/O Layout Diagram View \(page 67\)](#)
 - [I/O Package Diagram View \(page 63\)](#)
 - [I/O Pin Assignment View \(page 64\)](#)
 - [I/O Utilization View \(page 62\)](#)
- [IP Diagram View \(page 69\)](#)
- [IP Libraries View \(page 71\)](#)
- [IP Problems View \(page 71\)](#)
- [Multiprocess View \(page 73\)](#)
- [Netlist Browser View \(page 79\)](#)
- [NoC Performance View \(page 87\)](#)
- [NoC Time Slice View \(page 93\)](#)
- [Options View \(page 96\)](#)
- [Outline View \(page 106\)](#)
- [Partitions View \(page 107\)](#)
- [Placement Regions View \(page 110\)](#)
- [Projects View \(page 117\)](#)
- [Properties View \(page 123\)](#)
- [Register Browser View \(page 127\)](#)
- [Search View \(page 129\)](#)

- [Selection View \(page 133\)](#)
- [Snapshot Debugger View \(page 137\)](#)
- [Tcl Console View \(page 142\)](#)

Clock Domains View

The Clock Domains view provides a table listing all the clock domains found in the active design. Counts are also provided of the major logic types within each clock domain. Similar to the [Netlist Browser view \(page 79\)](#), filters are available for each column of the table, so that in cases where there are many clock domains in a design, the visible content of the table may be limited to just those clock domains meeting the chosen filter criteria. Filters are available for all columns of the table except the Highlight Color column.

By default, the Clock Domains view is included in the [Floorplanner perspective \(page 6\)](#). To add it to other perspectives, select **Window** → **Show View...** → **Other...** → **Achronix** → **Clock Domains**.

Clock Domain Name	Hig...	Flops	LUTs	ALUs	BRAMs	BMACCs	LRAMs	Others
= const		49	2726	9806	224	56	2870	40472
gclk	■	265274	534304	9946	224	56	2870	54044

Figure 7 - Clock Domains View Example With Filter Row Visible

Note

The various Achronix target devices contain different mixes of the possible resource types. Accordingly, the resource type columns in this view (e.g., flops, BRAMs, ALUs, etc.) are dynamic, and change to match the target device after the Prepare flow step has been run. The screenshots and example descriptions in this document section might not exactly reflect the current target device.







Table 7 - Clock Domains View Columns

Column Name	Description
Clock Domain Name	The name of the clock domain in the active design.
Highlight Color	If all instances within the clock domain have the same highlight color, this row shows a color square with that same highlight color. If even one contained instance has a differing highlight color, or no highlight at all, then the row displays no color square.

Column Name	Description
<i>Resource</i>	A different column is provided for each <i><resource></i> type (Flops, LUTs, etc.) within the target device. Each table cell in that column shows the sum count of all contained <i><resource></i> instances within the named clock domain for that row.

A number of actions are available in the view via buttons at the top of the view and context menus (right-click) on the rows of the table.

Table 8 • Clock Domains View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Add Instances to Selection	Y	Y	Adds the instances within the clock domain to the ACE selection set (as shown in the Selection view (page 133)).
	Choose Highlight Color	Y		Determines which color is applied to the objects chosen the next time the Highlight action is selected for this view.
	Highlight	Y	Y	Applies the currently active Highlight color to the instances within the chosen clock domain (see Highlighting Objects in the Floorplanner View (page 343)).
	Un-Highlight	Y	Y	Clears the Highlight for the instances within the chosen clock domain.
	Zoom To	Y	Y	Zooms the Floorplanner view (page 43) to an area containing all of the instances within the clock domain currently chosen in the tree.
	Search for Instances	Y	Y	Searches for instances belonging to the chosen clock domain. A Tcl <code>find</code> (page 638) command is issued, and the Search view (page 129) is populated with the results.


Icon	Action	Toolbar Button	Context Menu	Description
	Toggle Filter Row Visibility ⁽¹⁾	Y		Shows or hides the filter row (of filter icons).

Table Notes

- Toggle Filter Row Visibility** does not alter whether filters are active, it only shows or hides the icons.

Organizing Table Data

The following are ways to alter the presentation of the data in the Clock Domains table:

Column Resizing

The width of a column can be changed as follows:

1. place the mouse cursor over the boundary between columns (the mouse cursor changes to indicate resizing is possible).
2. Simply left-click and drag left or right to resize the column to the desired width, then release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold any column name.
2. Drag left or right to move the column between any other pair of columns.
3. Release the left mouse button to insert the column header at the new location.



While dragging, the dragged column header appears alongside the mouse cursor, plus a thick column header separator showing where the column insertion occurs when the mouse is released.


Filtering

Most columns of the table may filter the displayed clock domains by value. When filtering by column value, only clock domains with column values matching the filter are retained. Non-matching values are excluded from the table.


- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html>), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the **Overlay Color** column) lack a filter icon in the filter row.


To add a filter to a column:

1. The Filter row must first be visible. Select the () **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the () filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the () active filter icon.

To edit (or clear) an existing filter:


1. Click the () active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the () inactive version.

Drag-and-Drop

The Clock Domains view supports a limited set of drag-and-drop interactions with other views in the **Floorplanner perspective** (page 6). The view only acts as a drag-and-drop source. Items dropped on the Clock domains view are ignored.

Any row of the table may be dragged to the **Tcl console view** (page 142), and when dropped anywhere in the view, the clock domain name (with the appropriate object type prefix) is inserted at the beginning of the Tcl command-line.

Any clock domain in the table may be dragged to the **Placement regions view** (page 110) or the **Floorplanner view** (page 43) (when that view has the  **Placement Regions Tool** active) to **assign placement region constraints** (page 398). Dragging a clock domain is the equivalent of dragging all individual instances which are members of that clock domain. Be aware that since placement regions may only encompass the fabric core and boundary region, but not the I/O ring, any dragged I/O instances are not assigned to placement regions.

Clock Regions View

The Clock Regions view provides a tabular representation of the site type content of each of the **Clock regions** (page 262) in the currently selected Target Device, and allows toggling the visibility of the overlay within the **Floorplanner view** (page 43) for each Clock Region. The view table remains empty until the currently active Implementation has been prepared (i.e., the **Run Prepare** flow step has been completed).

By default, the Clock Regions view is included in the **Floorplanner perspective** (page 6). To add the view to the current perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Clock Regions**.

Visibility	Clock Region Name	Ov...	Clock Pre-Routes	LUTs	Flops	ALUs	LRAMs	BRAMs	IPINs
<input type="checkbox"/>	CLK_REGION_0_0			43200	86400	10800	160	160	4892
<input type="checkbox"/>	CLK_REGION_0_1		(clk_p: 2,4)	43200	86400	10800	160	160	1002
<input type="checkbox"/>	CLK_REGION_0_2			43200	86400	10800	160	160	1008
<input type="checkbox"/>	CLK_REGION_0_3			43200	86400	10800	160	160	1008
<input type="checkbox"/>	CLK_REGION_0_4			43200	86400	10800	160	160	1008
<input type="checkbox"/>	CLK_REGION_0_5		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	1008
<input type="checkbox"/>	CLK_REGION_0_6		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	1008
<input type="checkbox"/>	CLK_REGION_0_7		(clk_p: 1,3,5,7)	43200	86400	10800	160	160	4928
<input type="checkbox"/>	CLK_REGION_1_0			43200	86400	10800	160	160	4928
<input type="checkbox"/>	CLK_REGION_1_1			43200	86400	10800	160	160	1008

Figure 8 • Clock Regions View Example





⚠ Caution!

Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The resource type columns shown in the screenshot are examples only, and do not match all target devices.

Table 9 • Clock Regions Table Columns

Column	Editable	Description
Visibility	Y	When checked, the clock region overlay is painted in the Floorplanner view (page 43), using the translucent color shown in the Overlay Color column.
Name		The name of this clock region.
Overlay Color	Y	The color used to paint the location of the clock region as an overlay in the Floorplanner view. Right-click any row, then choose Change Overlay Color to choose an alternate overlay paint color for that clock region.
Clock Pre-Routes		If any clock pre-routes exist for a given partition, they are listed here.
Resource		The number of <resource> sites (Flops, LUTs, etc.) contained in this clock region.

Table 10 - Clock Regions View Actions

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show / Hide overlay		Y		Show or Hide the overlay for the chosen clock region in the Floorplanner view.
	Change Overlay Color		Y		Allows changing the translucent overlay color which is used to paint the chosen clock region in the Floorplanner view (when the visibility is enabled).
	Reset Overlay Color		Y		Reset the chosen clock region overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two clock regions are too similar for easy discernment, this action pseudo-randomly picks another color. Each time this action is chosen, another color is picked.
	Zoom To		Y		Pans and zooms the Floorplanner view to show the location of the selected clock region. ⁽¹⁾
	Reset All Overlay Colors			Y	Pseudo-randomly reassigns new overlay colors for all clock regions.
	Configure Clock Pre-Routes...		Y		Allows adding clock pre-route information for the selected partition(s).
	Show/Hide All Clock Regions	Y	Y	Y	Toggles the visibility checkboxes for all clock domains in the table, causing all to be alternately shown or hidden in the Floorplanner view.
	Toggle Filter Row Visibility	Y	Y	Y	Shows or hides the filter row (of filter icons) at the top of the table. ⁽²⁾

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
<p>Table Notes</p> <ol style="list-style-type: none"> 1. If the clock region visibility column checkbox is disabled, the clock region overlay is not painted and thus is not visible. 2. This toggle does not alter whether filters are active, it only shows or hides the row of filter icons. 					

Using the Table to Display Clock Regions in the Floorplanner View

Each clock region is automatically given a unique translucent overlay color to represent the clock region when painting the **Floorplanner view** (page 43). By default, no clock region overlays are painted in the Floorplanner view. Clock region overlays must be enabled if they are to be displayed. The overlay color may optionally be altered for each/all clock regions, but these color choices do not persist between ACE sessions.

Note


While alternate overlay colors are allowed to be chosen for each clock region, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each clock region.

The following are ways to alter the presentation of Clock region data in the **Floorplanner view** (page 43):

Enable/Disable Painting of Individual Clock Regions Within the Target Device:

When the checkbox in the **Visibility** column for a clock region is selected, the area of the target device (in the Floorplanner view) representing that clock region is painted in the displayed translucent overlay color. When the checkbox is unchecked, the Floorplanner view is redrawn with the chosen clock region overlay no longer painted.

Enable/Disable Painting of all Clock Regions Within the Target Device:

When the  **Show/Hide All Clock Regions** action is selected, the visibility of all clock regions are simultaneously either enabled or disabled, causing the Floorplanner view to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Clock Regions:

The overlay rendering color of each individual clock region may be chosen as follows:

1. Right-click the mouse pointer anywhere on the row of the desired clock region.
2. Select **Choose Overlay Color** from the popup context menu.
3. The Color Dialog may then be used to choose the desired color for the clock region.

Note

This is a temporary color change — colors are reverted to automatically chosen defaults if changes are made to the active design, the active implementation, the target device, or ACE is closed.

ACE automatically picks a different overlay color for an individual clock region if **Reset Overlay Color** is selected from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Clock Regions:

ACE automatically picks different overlay colors for all clock regions if the **Reset All Overlay Colors** action is selected from the Clock regions view local pull-down menu.

Organizing Table Data

The following are ways to alter the presentation of the data in the Clock regions table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns — at this point the mouse cursor should change to indicate resizing is possible.
2. Simply left-click and drag left or right to resize the column to the desired width.
3. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold on any column name
2. Drag left or right to move the column between any other pair of columns.
3. Release the left mouse button to insert the column header at the new location.



While dragging, the dragged column header is visible alongside the mouse cursor, and there is a thick column header separator showing where the column insertion occurs if the mouse is released at the present cursor location.

Filtering


Most columns of the table may filter the displayed clock regions by value. When filtering by column value, only clock regions with column values matching the filter are retained; non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html>), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the **Overlay Color** column) lack a filter icon in the filter row.


To add a filter to a column:


1. The Filter row must first be visible. Select the () **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the () filter icon for the desired column, which causes a data-appropriate filter dialog to appear.

3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the () active filter icon.

To edit (or clear) an existing filter:

1. Click the () active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the () inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all clock regions marked as "visible" in the view. Making more or fewer clock regions visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Clusters View

The Clusters view allows the toggling the visibility of overlays within the [Floorplanner view \(page 43\)](#) for each logic cluster. The view table remains empty until the currently active Implementation has been prepared (i.e. the **Run Prepare** flow step has been completed).

By default, the Clusters view is included in the [Floorplanner perspective \(page 6\)](#). To add the view to the current perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Clusters**.

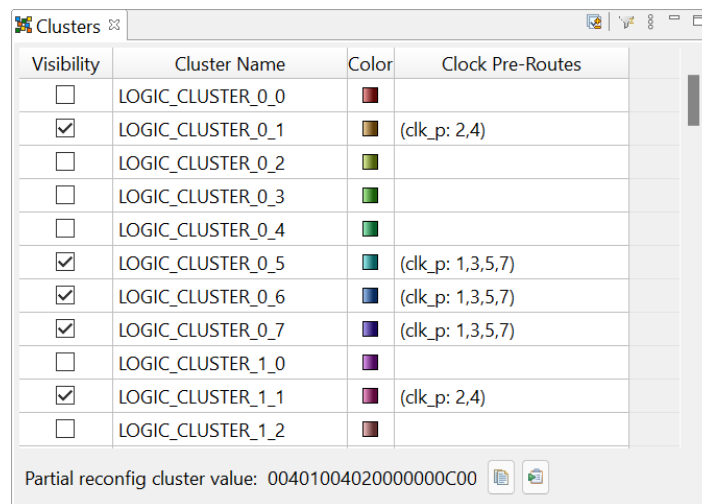


Figure 9 - Clusters View Example

Table 11 - Clusters Table Columns

Column	Editable	Description
Visibility	Y	When checked, the logic cluster overlay is painted in the Floorplanner view (page 43), using the translucent color shown in the Color column.
Cluster Name		The name of this cluster.
Color	Y	The color used to paint the location of the cluster as an overlay in the Floorplanner view. Right-click any row, then choose Change Overlay Color to choose an alternate overlay paint color for that cluster.
Clock Pre-Routes		If any clock pre-routes exist for a given partition, they are listed here.

Table 12 - Clusters View Actions

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show / Hide overlay		Y		Show or Hide the overlay for the chosen cluster in the Floorplanner view.






Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Change Overlay Color		Y		Allows changing the translucent overlay color which is used to paint the chosen cluster in the Floorplanner View (when the visibility is enabled).
	Reset Overlay Color		Y		Reset the chosen cluster overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two clusters are too similar for easy discernment, another color is pseudo-randomly selected. Each time this action is chosen, another color is selected.
	Zoom To		Y		Pans and zooms the Floorplanner view to show the location of the selected cluster. ⁽¹⁾
	Reset All Overlay Colors			Y	Pseudo-randomly reassigns new overlay colors for all clusters.
	Configure Clock Pre-Routes...		Y		Allows adding clock pre-route information for the selected partition(s).
	Show/Hide All Custers	Y	Y	Y	Shows or hides the checkboxes in the Floorplanner view for all logic clusters in the table.
	Toggle Filter Row Visibility	Y	Y	Y	Shows or hides the filter icons in the filter row at the top of the table. ⁽²⁾

Table Notes

1. If the cluster visibility column checkbox is disabled, the cluster overlay is not painted and thus is not visible.
2. This toggle does not alter whether filters are active, it only shows or hides the filter icons.

Using the Table to Display Clusters in the Floorplanner View

Each cluster is automatically given a unique translucent overlay color to represent the cluster when painting the **Floorplanner view** (page 43). By default, no cluster overlays are painted in the Floorplanner view. The cluster overlays must be enabled to be displayed. The overlay color for each/all clusters may optionally be altered, but these color choices are not persisted between ACE sessions.


 While choosing alternate overlay colors is allowed for each cluster, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each cluster.

The following are ways to alter the presentation of Cluster data in the **Floorplanner view** (page 43):

Enable/Disable Painting of Individual Clusters Within the Target Device:

When the checkbox in the **Visibility** column for a cluster is checked, the area of the target device (in the Floorplanner view) representing that cluster is painted in the displayed translucent overlay color. When unchecked, the Floorplanner view is redrawn with the chosen cluster overlay no longer painted.

Enable/Disable Painting of All Clusters Within the Target Device:

When the  **Show/Hide All Clusters** action is chosen, the visibility of all clusters are simultaneously either enabled or disabled, causing the Floorplanner view to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Clusters:

The overlay rendering color of each individual cluster may be chosen as follows:

1. Right-click the mouse pointer anywhere on the row of the desired cluster.
2. Select **Choose Overlay Color** from the popup context menu.
3. The Color Dialog can then be used to choose the desired color for the cluster.

 **Note**

This is a temporary color change — colors are reverted to automatically chosen defaults if changes are made to the active design, the active implementation, the target device, or ACE is closed.

ACE automatically picks a different overlay color for an individual cluster if **Reset Overlay Color** is chosen from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Clusters:

ACE automatically picks different overlay colors for all clusters if the **Reset All Overlay Colors** action is chosen from the clusters View local pull-down menu.

Organizing Table Data

The following are ways to alter the presentation of the data in the clusters table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns.
2. At this point, the mouse cursor should change to indicate resizing is possible.
3. Simply left-click and drag left or right to resize the column to the desired width.
4. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Left-click and hold on any column name.
2. Drag left or right to move the column between any other pair of columns.
3. Release the left mouse button to insert the column header at the new location.

While dragging, the dragged column header is visible alongside the mouse cursor and there is a thick column header separator showing where the column insertion is to occur if the mouse is released at the present cursor location.

Filtering

Most columns of the table may filter the displayed clusters by value. When filtering by column value, only clusters with column values matching the filter are retained. Non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html>), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the **Overlay Color** column) lack a filter icon in the filter row.

To add a filter to a column:


1. The Filter row must first be visible. Select the (🔍) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the (🔍) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the (🔍) active filter icon.

To edit (or clear) an existing filter:

1. Click the (🔍) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.

- Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the () inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all clusters marked as "visible" in the view. Making more or fewer clusters visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Critical Path Diagram View

The Critical Path Diagram view provides a graphical representation of a single critical path. Selecting a row in the associated [Critical Paths view \(page 37\)](#) table updates the diagram in the Critical Path Diagram view so that it contains a graphical representation of the selected critical path. The graphical representations consist of circular nodes (representing instances) connected by arrows (representing one or more nets). Similar to the [Floorplanner view \(page 43\)](#), the Critical Path Diagram view contains a fly-out palette of display options on the right of the diagram (collapsed/hidden by default), and a collection of buttons at the top. The colors used in the Critical Path Diagram view are configured via the [Critical Path Diagram view Preference page \(page 193\)](#). For details on usage of the diagram view, please see [Using Critical Path Diagrams \(page 356\)](#) and [Analyzing Critical Paths \(page 353\)](#).

By default, the Critical Path Diagram view is included in the [Floorplanner perspective \(page 6\)](#). To add it to the current perspective, select **Window** → **Show View** → **Other...** → **Critical Path Diagram**.

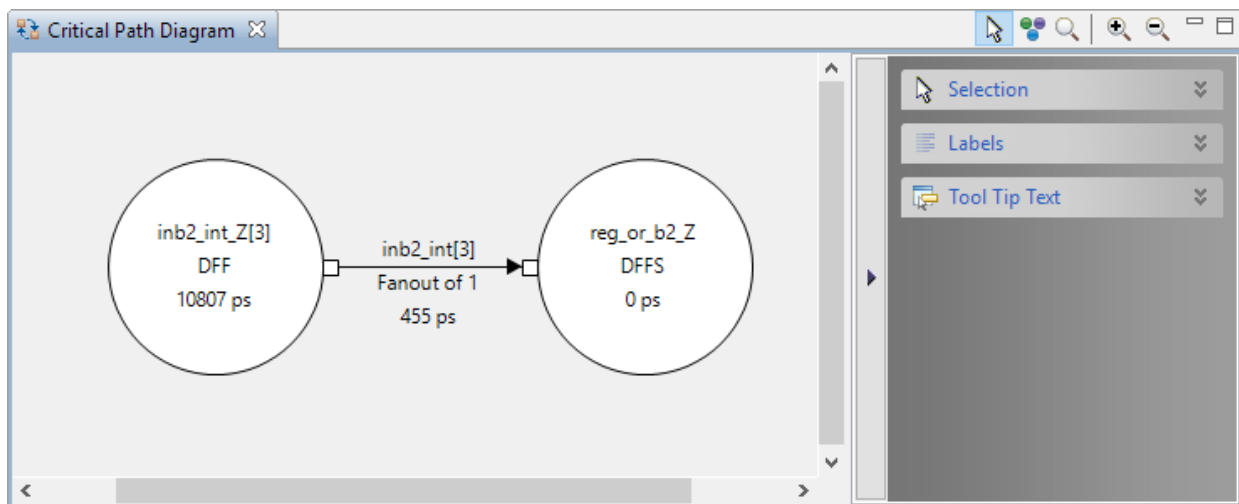


Table 13 - Critical Path Diagram View Example, Including Expanded Fly-Out Palette

When no critical path is selected in the [Critical Paths view \(page 37\)](#), the diagram displays this warning:

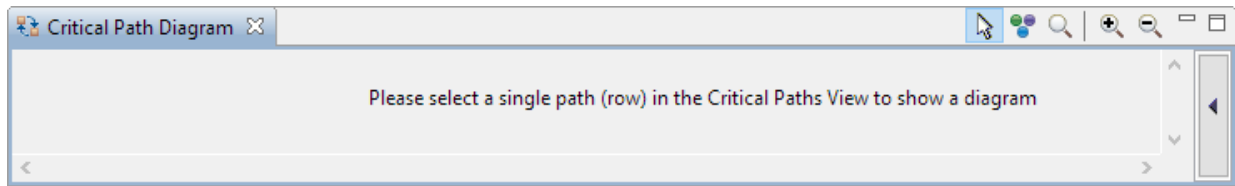


Figure 10 • No Critical Path Selected Warning, Including Collapsed Palette

Table 14 • Critical Path Diagram View Toolbar Buttons

Icon	Action	Description
	Selection tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The selection tool creates a selection rectangle when dragging with the left mouse button. Any objects in the selection rectangle are either added to or removed from the current ACE selection set, as configured in the fly-out palette.
	Movement tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The movement tool pans the view when dragging with the left mouse button.
	Zoom tool	Controls the behavior of the mouse while in the Critical Path Diagram view. The zoom tool creates a zoom-in rectangle when the left mouse button is pressed and held, then dragged to the lower-right. The zoom tool creates a zoom-out line when the left mouse button is pressed and held, then dragged to the upper-left.
	Zoom in	Increases the current zoom level in the Critical Path Diagram view by 200%.
	Zoom out	Decreases the current zoom level in the Critical Path Diagram view by 200%.

Right-clicking on an instance or net within the diagram also displays a context menu of additional actions.

Table 15 • Critical Path Diagram View Context Menu Actions

Icon	Action	Description
	Add to Selection	The instance or net under the mouse is added to the ACE selection set (and painted in the selection color).




Icon	Action	Description
	Remove from Selection	The instance or net under the mouse is removed from the ACE selection set if currently selected (and thus no longer is painted the selection color).
	Highlight	Sets the highlight color for the instance or net under the mouse to the currently-chosen view highlight color.
	Choose highlight color	Determines which color is applied to instances/nets the next time the highlight action is selected for this view.
	Un-Highlight	Turns off the highlight color for the instance or net under the mouse.
	Zoom To	Pans and zooms the Floorplanner view (not this view) to the closest zoom that still displays (centered) the entire instance or net currently under the mouse in the Critical Path diagram.
	Show in Netlist ⁽¹⁾	Attempts to open a text editor to the file and line number relevant to the instance or net under the mouse.
	Fix Placement of Instance	Causes the placement state of the instance under the mouse cursor to change from unfixed (or soft) to fixed.
	Unfix Placement of Instance	Causes the placement state of the instance under the mouse cursor to change from fixed to unfixed (or soft).
	Unplace Instance	Causes the placed instance under the mouse cursor to be unplaced, vacating the site.
	Unplace All Selected Instances	Causes all instances currently in the ACE Selection set (as listed in the Selection view (page 133)) to be unplaced at once (this is much more efficient than unplacing multiple instances individually).

Table Notes

1. This Early Access functionality might not always open the text editor to the expected location.

Fly-Out Palette

As with many other ACE diagram views, the Critical Path Diagram view includes a supplementary fly-out palette of additional settings which affect the interactions with the diagram, as well as the appearance of the diagram. The following options are available in the fly-out palette in the Critical Path Diagram view.

Selection



The () Selection Options control the selection of objects in the Critical Path Diagram view.

Table 16 - Selection Options

Option	Default	Description
Select	Enabled	This radio button controls the action applied to objects in the selection region. This setting causes the objects to be added to the current ACE selection set.
Deselect	Disabled	This radio button controls the action applied to objects in the selection region. This setting causes the objects to be removed from the current ACE selection set.

For more details about viewing and managing the ACE Selection set, see [Selection view \(page 133\)](#).

Label

The () Label options control the text labels on graph nodes and arrows (instances and net abstractions) in the Critical Path Diagram view.

Note

These labels are only displayed when there is enough room to print them on screen. It might be necessary to **Zoom In** to provide sufficient area for all the desired text to be displayed.

Table 17 - Label Options

Option	Description
Instance Names	Displays the instance name each graph node represents.
Instance Types	Displays the instance type (cell) for each graph node.
Net Names	Displays the net name represented by each arrow.
Delays	Displays the delay (in ps) to traverse each node or arrow.
Fanouts	Displays the fanout of the net represented by the arrow.

Tool Tip Text


The  ToolTip options control the tooltip content while hovering over graph nodes and arrows in the Critical Path Diagram view.

Table 18 • Tooltip Options

Option	Description
None	No tooltips are displayed for nodes or arrows.
Instance Names	Displays the instance name each graph node represents.
Instance Types	Displays the instance type (cell) for each graph node.
Net Names	Displays the net name represented by each arrow.
Pin Names	Displays source and sink pin names for nets.
Delays	Displays the delay (in ps) to traverse each node or arrow.
Fanouts	Displays the fanout of the net represented by the arrow.

Critical Paths View

The Critical Paths view provides a table of critical paths resulting from running timing analysis (the view contains no data unless or until one of the **Run ... Timing Analysis** flow steps is completed). This view (in cooperation with the [Critical Path Diagram view \(page 33\)](#)) displays critical path details, manages selection of objects on critical paths, and highlights critical paths in the [Floorplanner view \(page 43\)](#).

Note

The Critical Paths view is populated with the results from timing analysis, and thus remains empty unless or until one of the **Run ... Timing Analysis** flow steps is executed (timing analysis data is not stored within the *.acxdb files, thus timing analysis must be re-run every time the Critical Paths view is used).

Clicking a row in the table enables toolbar buttons for analyzing the associated critical path, and causes the display of a graphical diagram of the associated critical path in the [Critical Path Diagram view \(page 33\)](#). Clicking a column header sorts the table according to the data in that column.

By default, the Critical Paths view is included in the [Floorplanner perspective \(page 6\)](#). To add it to the current perspective, select **Window** → **Show View...** → **Critical Paths**.

Table 19 - Critical Path View Actions

Icon	Action	Description
	Select path	Adds the selected critical path in the table to the current ACE selection set.
	Select pins	Adds pins on the selected critical path in the table to the current ACE selection set.
	Select instances	Adds instances on the selected critical path in the table to the current ACE selection set.
	Select nets	Adds nets on the selected critical path in the table to the current ACE selection set.
	Zoom to path	Zooms the Floorplanner view to a region containing the selected critical path in the table.
	Print Path Details	Produces a detailed report of the selected critical path in the table to the text output in the Tcl Console view.
	Save Script File	Displays the Save Script File dialog which allows saving a Tcl script of find commands for use in the schematic viewer of the synthesis tool.

The view is primarily made up of a tree table, with each branch of the tree representing a separate clock domain. The most critical path of each clock domain is the branch node, with all other paths from that clock domain acting as leaves for that branch. Setup violations are considered "worse" than hold violations, thus any setup violation takes precedence over hold violations as the branch node, regardless of relative slack values.

Highlight	Clock Domain	Path	Slack	Type	Source	Destination
<input checked="" type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s29	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s31	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s34	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s35	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s36	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s37	-0.944	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r	pcie_complete_core_vc1.u_axi_fifo.genblk6.u_c2s0_r_fifo.u_fifo.last_contr
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s30	-0.658	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s32	-0.658	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s38	-0.658	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r	pcie_complete_core_vc1.u_axi_fifo.genblk8.u_c2s1_r_fifo.u_fifo.if_num_fu
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[0]	s33	-0.601	Setup Check Violated	pcie_complete_core_vc1.u_axi_fifo.genblk1.s2c_ars	pcie_complete_core_vc1.u_axi_fifo.genblk2.u_s2c0_w_fifo.u_fifo.x_fifo_co
<input type="checkbox"/>	u_pll1.SE_APLL_0_pcie_pll1.APLL.IACX.PLL/ogg_gm_clk[1]	s19	0.050	Setup Check Met	pcie_complete_core_vc1.u_axi_fifo.genblk2.u_s2c0_w	pcie_complete_core_vc1.u_axi_fifo.genblk2.u_s2c0_w_fifo.u_fifo.if_num_f
<input type="checkbox"/>	pcie_complete_core_vc1.x_pcie.pipe_ifinst_in0_serdes/o	s10	0.523	Setup Check Met	sac_mgmt_dl_link_up_o	mgmt_dl_link_up_o_reg_Z
<input type="checkbox"/>	apb_pclk	s0	6.720	Setup Check Met	sac_o_sbus_ack	u_sbus_if.rd_data_shift_in_Z[31]

Figure 11 - Critical Path View Example

Entries in the table are always grouped by clock domain, with individual paths sorted within a clock domain. The clock domains themselves are (by default) sorted from most critical to least critical.

The default sort order, from most critical to least critical, of the critical paths is as follows:

1. Setup violations, from the most negative slack value to zero.
2. Hold violations, from the most negative slack value to zero.
3. Setup met, from zero to the most positive slack.
4. Hold met, from zero to the most positive slack.

Table 20 - Critical Path View Table Columns

Column	Description
Highlight	Allows highlighting the path in the Floorplanner view, using the checkbox. Also allows configuring the highlight color of the path by clicking the color selector box.
Clock Domain	Displays the clock domain name of the path.
Path	Displays the unique path ID (used in the Timing report (page 244)).
Slack	Displays the slack for the path in ns.
Type	Displays the path type.
Source	Displays the source instance of the path.
Destination	Displays the destination instance of the path.

By default, the highlight colors for the Setup violation and Hold violation path types ranges from red (the worst slack values) through orange to yellow (any violation slack values close to zero). The default highlight color of Slack Met and Hold Met path types is always green, and does not vary by reported slack value.

The view local pull-down menu (found to the right of the view toolbar buttons) contains some additional controls for the view: four filters to control which types of paths are displayed in the table, as well as shortcuts to run the four stages of timing analysis. As mentioned previously, the most critical path within each clock domain is always displayed, regardless of the type filter settings. (Every clock domain is always represented in the tree table by at least one row of data.)

Table 21 - Critical Paths View Drop-down Menu Actions

Action	Description
Show Clock Paths	If checked, highlighted critical paths in the Floorplanner view show the clock routing segments as part of the critical path. If unchecked, only the data portion of the critical path is shown.

Action	Description
Show Setup Violations	If checked, Setup violation leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Hold Violations	If checked, Hold violation leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Setup Met	If checked, Setup Met leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Show Hold Met	If checked, Hold Met leaf nodes are shown in the treetable. If unchecked, these leaf nodes are hidden.
Run Prepared Timing Analysis	If selected, runs the Prepared Timing Analysis flow step.
Run Post-Place Timing Analysis	If selected, runs the Post-Place Timing Analysis flow step.
Run Post-Route Timing Analysis	If selected, runs the Post-Route Timing Analysis flow step.
Run Final Timing Analysis	If selected, runs the Final Timing Analysis flow step.

Download View

The Download view provides a graphical interface for choosing and downloading (via JTAG) a bitstream *.hex file to an Achronix FPGA or eFPGA connected to the workstation using USB via a Bitporter2 pod or FTDI FT232H or FT4232H device. Bitstream *.hex files are generated by ACE during the **Generate Bitstream flow step** (page 234).

By default, the Download view is included in the **Programming and Debug perspective** (page 6). To access the Download view if it is not visible in the current perspective, select **Window** → **Show View...** → **Others** → **Download View**.

When the Download view opens, the windows might need to be resized for optimal viewing.

⚠ Caution!**The JTAG connection must be configured before using the Download view!**

- ACE interacts with the FPGA or eFPGA using the JTAG interface through a USB Bitporter2 pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the Download view. The configuration is managed using the [Configure JTAG Connection preference page \(page 190\)](#), which is easily accessible by clicking the (🔧) **Configure JTAG Interface** button in the Download view. See [Configuring the JTAG Connection \(page 360\)](#) for more details. The (▶) **Download Bitstream** button and the (🔧) **Configure JTAG Interface** button each also provide a summary of the current JTAG configuration settings (for pod name and scan chain) in their tooltips for ease of reference.
- While using the Download view, it is strongly recommended that the [Tcl Console view \(page 142\)](#) be kept visible to display any status or error messages reported during the JTAG interactions. The Tcl Console view and Download view are both visible by default in the Programming and Debug perspective.

In addition to allowing the choice of a bitstream file and performing actual downloads, this view also allows toggling of some programming options, and some related utility interactions with JTAG-connected devices.

For all buttons and checkboxes, tooltips provide a more verbose description of each, if the summary statement is insufficient. The (▶) **Download Bitstream** button and the (🔧) **Configure JTAG Interface** button each also provide a summary of the current JTAG configuration settings (for pod name and scan chain) in their tooltips for ease of reference.

See also:

- [Programming a Device using JTAG in the Download View \(page 390\)](#)
- *Configuration User Guide (UG004)*
- Device family-specific user guide (i.e., *Speedster7t Configuration User Guide (UG094)*)

After downloading the bitstream, the JTAG connection status is retained, meaning that if the JTAG connection is open prior to programming, it stays open, and if the JTAG connection is closed prior to programming, it closes the connection when programming completes.

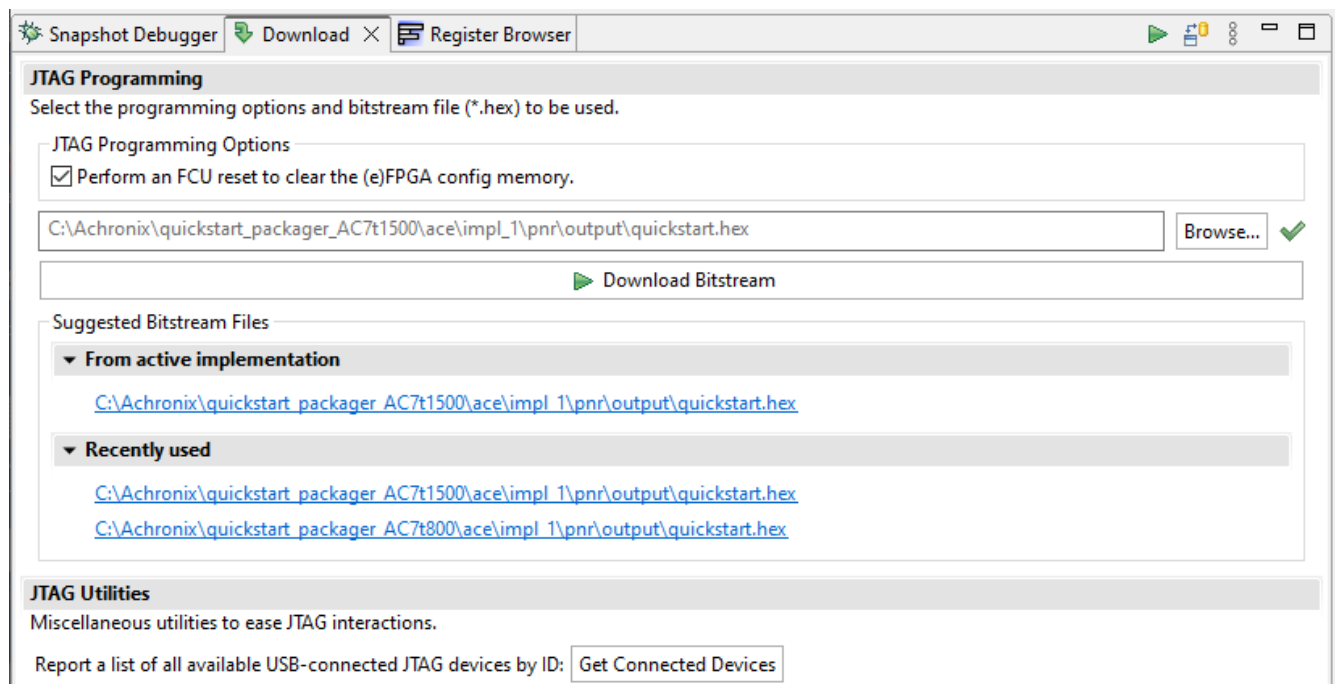



Figure 12 - Download View Example

Table 22 - Download View Options

Option	Description
JTAG Programming Options	
Perform an FCU reset to clear the (e)FPGA config memory.	When checked, performs a soft reset and clears all device configuration memory before beginning programming. This reset is typically only disabled for multi-stage programming (after stage 0 programming has completed, before programming later stages begins), or for "partial reconfig" when partial bitstreams are in use (see the chapter titled Partial Reconfiguration in the <i>Speedster7t Configuration User Guide</i> (UG094) for more details).
Browse	Allows choosing any *.hex bitstream file from the file system using a graphical file system browser.
 Download Bitstream.	Clicking this button performs the actual download by calling the appropriate Tcl commands in the <code>jtag::</code> namespace. See also: <i>Speedster7t Configuration User Guide</i> (UG094).
Suggested Bitstream Files	

Option	Description
From active implementation.	A list of all *.hex bitstream files (shown as hyperlinks) found in the output directory of the current Active Implementation (page 229). Select any of these hyperlinks to choose that file for download.
Recently used.	A list of the most recently used *.hex bitstream files (shown as hyperlinks). Select any of these hyperlinks to choose that file for download.
JTAG Utilities	
Report a list of all available USB-connected JTAG devices by ID.	Press the button to run a Tcl command (<code>jtag::get_connected_devices</code>) to report a list of all connected JTAG devices in the Tcl Console view.

Floorplanner View

The Floorplanner view provides a graphical view of the physical layout of the device. This view allows visualizing the device, place and route data, critical paths, and the current selection set. The view allows zooming out to see a general overview of the user design mapped onto the device, or zooming in to see specific details.

Clicking the tall narrow arrow button on the far right of the Floorplanner view shows or hides the **fly-out palette** (page 47) of display options.

By default, the Floorplanner view is included in the **Floorplanner perspective** (page 6). To add it to the current perspective, select **Window** → **Show View** → **Other...** → **Floorplanner**.

See also:

- [Viewing the Floorplanner](#) (page 339)
- [Pre-placing a design](#) (page 346)
- [Floorplanner View Colors and Layers Preference Page](#) (page 194)
- [Floorplanner View Optimizations Preference Page](#) (page 200).

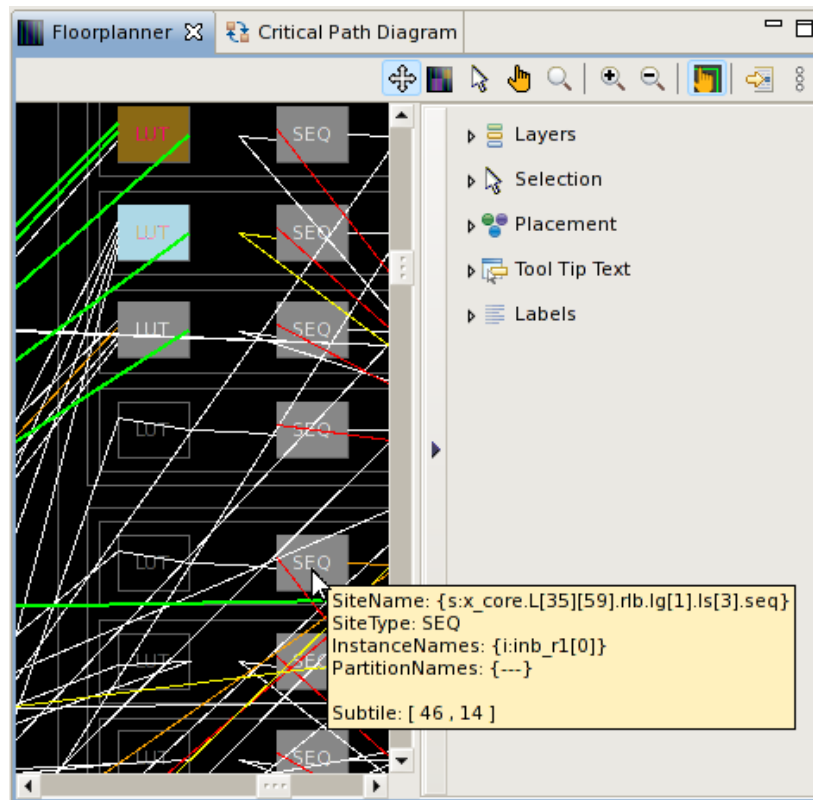









Figure 13 - Floorplanner View Example, Including Expanded Fly-Out Palette



Table 23 - Floorplanner View Toolbar Buttons

Icon	Action	Description
	Panning tool.	Controls the behavior of the mouse while in the Floorplanner view. The panning tool allows panning of the view when the left mouse button is pressed and held. Holding SPACE temporarily switches to the Panning tool. Releasing SPACE switches back to the previously selected tool. Tapping ALT cycles to the next tool.
	Placement Region tool.	Controls the behavior of the mouse while in the Floorplanner view. Allows the manipulation of placement regions and placement region constraints (page 394). When the Placement Region tool is active, the mouse may be used to create new placement regions (page 395), move (page 397) or resize (page 397) existing placement regions, and/or assign objects to placement region constraints (page 398). Pressing ALT cycles to the next tool.

Icon	Action	Description
	Selection tool.	Controls the behavior of the mouse while in the Floorplanner view. The selection tool creates a selection rectangle when the left mouse button is pressed and held. Any objects in the selection rectangle are applied with the current selection action, as configured in the fly-out palette. Pressing ALT cycles to the next tool.
	Placement tool.	Controls the behavior of the mouse while in the Floorplanner view. The placement tool allows drag-and-drop placement of instances when the left mouse button is pressed and held. Pressing ALT cycles to the next tool.
	Zoom tool.	Controls the behavior of the mouse while in the Floorplanner view. The zoom tool creates a zoom-in rectangle when the left mouse button is pressed and held, then dragged to the lower-right. The zoom tool creates a zoom-out line when the left mouse button is pressed and held, then dragged to the upper-left. Pressing ALT cycles to the next tool.
	Zoom in.	Increases the current zoom level in the Floorplanner view by 200%.
	Zoom out.	Decreases the current zoom level in the Floorplanner view by 200%.
	Toggle drag-scrolling.	Toggles drag-scrolling on and off. When drag-scrolling is enabled, dragging an instance into a defined margin around the edges of the view causes the view to automatically scroll in that direction. See the task Floorplanner Panning (page 340) for complete details. Pressing "Q" toggles this setting.
	Save pre-placement constraints.	Opens the Save Placement dialog (page 182) allowing the current placement to be saved to a pre-placement constraints (. pdc) file.

An instance or net within the Floorplanner may also be right-clicked to display a context menu of additional actions.

Table 24 • Floorplanner View Context Menu Actions

Icon	Action	Description
	Add to selection.	The instance or net under the cursor is added to the ACE selection set (and is painted in the selection color).
	Remove from selection.	The instance or net under the cursor is removed from the ACE selection set if currently selected (and thus is no longer painted the selection color).
	Highlight.	Sets the highlight color for the instance or net under the cursor to the currently-chosen Floorplanner view highlight color.





Icon	Action	Description
	Choose highlight color.	Determines which color is applied to instances or nets the next time the Highlight action is selected for this view.
	Remove highlight.	Turns off the highlight color for the instance or net under the cursor.
	Zoom to.	Pans and zooms the Floorplanner view to the closest magnification that displays the entire centered instance or net under the cursor.
	Show in netlist. ⁽¹⁾	Attempts to open a text editor to the file and line number relevant to the instance or net under the cursor.
	Fix instance placement.	Causes the placement state of the instance under the cursor to change from unfixed (or soft) to fixed.
	Unfix instance placement.	Causes the placement state of the instance under the cursor to change from fixed to unfixed (or soft).
	Unplace instance.	Causes the placed instance under the cursor to be unplaced, vacating the site.
	Unplace all selected instances.	Causes all Instances currently in the ACE selection set (as listed in the Selection view (page 133)) to be unplaced at once (this is much more efficient than unplacing multiple instances individually).

Table Notes

1. "Show in Netlist" is early access functionality and might not always open the text editor to the expected location.

Panning and Zooming

The Floorplanner view allows zooming in and out, to see more or less details respectively. There are several ways to change the zoom level:

1. Use the mouse scroll wheel.
2. Use the  **Zoom In** and  **Zoom Out** buttons in the toolbar.
3. Use keyboard shortcuts.

See the task [Zooming the Floorplanner In and Out \(page 339\)](#) for complete details.

Most of the other views within the Floorplanner perspective also include context-sensitive actions to **Zoom To** chosen individual objects or groups of objects — these actions cause the Floorplanner to center the chosen object(s)

in the Floorplanner, and to change the zoom level so that the chosen object(s) are as large/detailed as possible without overflowing the visible area.

When zoomed in, the FPGA requires more area than can easily fit in the view, making it necessary to pan the view around to see the different areas of the FPGA. Panning is most frequently performed using the arrow keys on the keyboard, mouse interactions with the scrollbars, the Panning Tool, or drag-scrolling during drag-and-drop interactions. See the [Floorplanner Panning \(page 340\)](#) task for complete details.

Note

When painting objects in the Floorplanner when the view is zoomed out, some objects become too small to be rendered with any detail. These objects are painted, at a minimum, as a single pixel of the appropriate color.

Empty sites (those without a placed instance) are a special case. Unless selected, sites that are too small are not painted at all, even if layer settings would otherwise allow them to be visible. Selected sites are always painted, with a minimum size of a single pixel.

When a single pixel represents multiple objects, as happens when zoomed all the way out, ACE paints only the most critical or most important object state at that pixel location, so the single pixel is the most critical or most important color. The relative priorities of the states are described in [Instance States \(page 262\)](#).

Fly-Out Palette

The following options are available in the fly-out palette in the Floorplanner view:

Layers


The () Layer Options control several layers of visible data in the Floorplanner view, allowing filtering the view so it contains a desired subset of all the available information.

Table 25 • Layer Options

Option	Default	Description
Instances	Enabled	Shows all placed instances.
Selected Instance Flylines ⁽¹⁾	Disabled	Shows flylines representing the net connections of selected instances (in the ACE selection set).
Clock Nets	Enabled	Shows all clock nets.
Non-clock Nets	Enabled	Shows all non-clock nets.
Routing Status		

Option	Default	Description
Open Connections ⁽²⁾	Disabled	Toggles the display of open portions of a net. Open connections are displayed in the same color as the routed portion of a net, but with a dotted line instead of a solid line. Open connections are a subset of a normal net, and are thus also managed by the layer options for Non-clock Routes , Clock Routes , and Route Drawing Mode .
Open Pins	Disabled	Toggles the display of squares highlighting the pins (red by default) at the endpoints of open connections.
Overflows	Disabled	Toggles the display of diamonds highlighting pins (orange by default) where route overflows occur (this is very rare).
Pins	Disabled	Shows all pins on each site.
Sites	Enabled	Shows all the sites on the device.

Table Notes

1. The displayed flylines are filtered by the **Non-clock Routes** and **Clock Routes** layer checkboxes. If only **Clock Routes** is checked, then only the flylines for clock nets of the selected instance(s) are displayed.
2. **Caution:** Dotted lines, as used for open connections, are much slower to render than solid lines. Thus, it is recommended that open connections remain disabled unless they are specifically needed for debugging purposes.

Note

Open Connections and Open Pins

When displaying an open connection for a placed instance, if the specific source and/or sink pins are not yet known (or not yet specified by the router), the connection is rendered to/from the center of the placed instance instead of to/from a specific pin. Likewise, when specific pins are not known, the **Open Pins** squares (red by default) are rendered in the center of the placed instance instead of on a specific pin (open connections and open pins are not, of course, rendered for unplaced instances).

Be aware that in a placed design that has not yet been routed, all nets are considered open connections.

Enabling **Open Pins** can make it much easier to find unrouted portions of a mostly routed design when zoomed out. But be aware this might be overwhelming on a large design that has been placed but not yet routed.

Every unrouted net is considered open. Thus, in an unrouted design, every endpoint of every net displays an open pin square, merging into a single large mass of color when zoomed out.

 **Tip**
Objects in the ACE Selection Set Are Always Visible

By default, any/all objects in the current ACE selection set (as shown in the [Selection view \(page 133\)](#)) are always visible in the Floorplanner, regardless of the chosen "Layers" filter settings. This means even if the **Instances** layer is disabled, any Instances in the current ACE selection set are still painted in the selected instances color (by default, bright green). Details of this behavior can be configured on the [Floorplanner view colors and layers preference page \(page 194\)](#).

In addition to the layers listed in the table, there are several other types of information displayed in the Floorplanner — enabling and disabling the display of these other types of information is controlled from other views. For example, the visibility of individual clock regions is controlled from the [Clock Regions view \(page 23\)](#); the visibility of individual Placement regions is controlled from the [Placement Regions view \(page 110\)](#); and the visibility of individual critical paths is controlled from the [Critical Paths view \(page 37\)](#).

Highlighting

Special colored highlighting of objects in the Floorplanner is possible via Tcl (see the [highlight \(page 664\)](#) Tcl command) and/or may be triggered via associated highlighting actions in most of the other views in the Floorplanner perspective. Highlighted objects are only visible in the Floorplanner if the appropriate layer is enabled, and the highlight color is only used if the object is not currently a member of the ACE selection set.

By default, the selection color takes precedence over the highlight color, which in turn takes precedence over the default color of the object. Further information about precedence of these states for instances can be found under [Instance states \(page 262\)](#), and can be partially reconfigured in the [Floorplanner view colors and layers preference page \(page 194\)](#). Additional information regarding highlighting can be found at [Highlighting Objects in the Floorplanner View \(page 343\)](#).

Selection


The () Selection Options control the selection of objects with the mouse in the Floorplanner view. Selected objects are added to the ACE selection set, and displayed appropriately in the [Selection view \(page 133\)](#).

Table 26 • Selection Options

Option	Default	Description
Instances	Enabled	Enables visible instances to be selected. If not checked, instances in the selection region are not added to the ACE selection set.
Nets	Enabled	Enables visible nets to be selected. If not checked, nets in the selection region are not added to the ACE selection set.
Pins	Disabled	Enables visible user design pins to be selected. If not checked, pins in the selection region are not added to the ACE selection set.

Option	Default	Description
Paths	Disabled	Enables visible paths to be selected. If not checked, paths in the selection region are not added to the ACE selection set.
Sites	Disabled	Enables visible sites to be selected. If not checked, sites in the selection region are not added to the ACE selection set.
Action		
Select	Enabled	Controls the action applied to objects in the selection region. Causes the objects to be added to the current ACE selection set.
Deselect	Disabled	Controls the action applied to objects in the selection region. Causes the objects to be removed from the current ACE selection set.
Remove Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes the placed instances to be un-placed.
Fix Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes the soft-placed instances to attempt to have fixed placement at the same site.
Un-fix Placement	Disabled	Controls the action applied to enabled objects in the selection region. Causes any fixed-placed instances to change to soft placement at the same site.

 **Note**

If **Instances** is checked under **Selection** but not under **Layers**, it is not possible to perform selection actions upon instances in the Floorplanner view using the mouse. For example, selection actions may be performed on only clock routes or only non-clock routes as desired, by simply setting the **Layers** filters appropriately.

Placement


The () Placement Options control the drag-and-drop placement behavior in the Floorplanner view.

Table 27 - Placement Options

Option	Default	Description
Group Placement	Disabled	[Expert Functionality] Controls whether single instances or groups of instances are placed with the drag-and-drop action of the Placement tool. Requires a group of instances to be in the ACE Selection Set prior to initiating drag and drop. The Group Placement option only succeeds in very specific circumstances, thus this setting should only be enabled by expert users who understand the caveats.
Fixed Placement	Enabled	Controls whether the drag-and-drop placement of an instance should be considered fixed or soft. Fixed placements are not changed by the placer. Soft placements are taken as a placement hint and might be changed by the placer.

⚠ Caution

When pre-placing objects (for a pre-placement constraints .pdc file), **Fixed Placement** should always be enabled.

Tool Tip Text

The () Tooltip options control the tooltip content while hovering over visible objects in the Floorplanner view.

Table 28 - Tooltip Options

Option	Default	Description
Allow Tooltips	Enabled	Allows enabling/disabling Tooltip support for the Floorplanner without needing to toggle all the individual checkboxes.
Instance Names	Enabled	Includes the names of all placed instances under the current mouse position in the tooltip text.
Port Names	Enabled	Includes the RTL port names of placed instances under the current mouse position in the tooltip text.
Net Names	Enabled	Includes all net names under the current mouse position in the tooltip text.
Pin Names	Disabled	Includes all user design pin names under the current mouse position in the tooltip text.

Option	Default	Description
Site Names	Enabled	Includes all leaf site names under the current mouse position in the tooltip text.
Site Types	Enabled	Includes the site cell type of each leaf site under the current mouse position in the tooltip text.
Site Pin Names	Disabled	Includes all site pin names under the current mouse position in the tooltip text.
Device Port Names	Enabled	Includes the top-level port names of the target device under the current mouse position in the tooltip text.
Subtile Coordinates ⁽¹⁾	Enabled	Includes the subtile coordinates under the current mouse position in the tooltip text.
Partition Names	Enabled	Includes all partition names under the current mouse position in the tooltip text.
Clusters	Enabled	Includes the name of the cluster under the current mouse position in the tooltip text.

Table Notes

1. Subtile Coordinates may be used with placement region commands on the Tcl command line.

Note

Tooltips Are Filtered by Layers Visibility

If **Instance Names** is checked under "Tool Tip Text" but **Instances** is not checked under "Layers", it is not possible to see instance names in the tooltips.

Label


The () Label options control the text labels on objects in the Floorplanner view.

Table 29 • Label Options

Option	Default	Description
None	Enabled	Disables the label display.
Instance Names	Disabled	Displays the instance names on placed instances.

Option	Default	Description
Port Names	Disabled	Displays the RTL port names on placed instances.
Site Names	Disabled	Displays the full site names on each leaf site.
Site Types	Disabled	Displays the site cell type on each leaf site.
Device Port Names	Disabled	Displays the top-level port names of the target device connected to the I/O site.

Flow View

The Flow view provides a hierarchical view of [Flow steps \(page 234\)](#) that can be performed on the [Active Project and Implementation \(page 229\)](#). From here, flow steps can be run and [flow status \(page 242\)](#) viewed. Flow steps are not able to run unless an active implementation is selected in the [Projects view \(page 117\)](#). When running flow steps, the [implementation options \(page 96\)](#) of the active implementation are used to govern the behavior of the flow. Be aware that altering the value of an implementation option clears the flow state of all downstream flow steps, changing them from the **Complete** state back to **Incomplete**.

By default, the Flow view is included in the [Projects perspective \(page 6\)](#). To add it to the current perspective, click **Window** → **Show View** → **Other...** → **Achronix** → **Flow**.

For more details, see the [Flow concept \(page 234\)](#) and the tasks for [Running the Flow \(page 306\)](#).

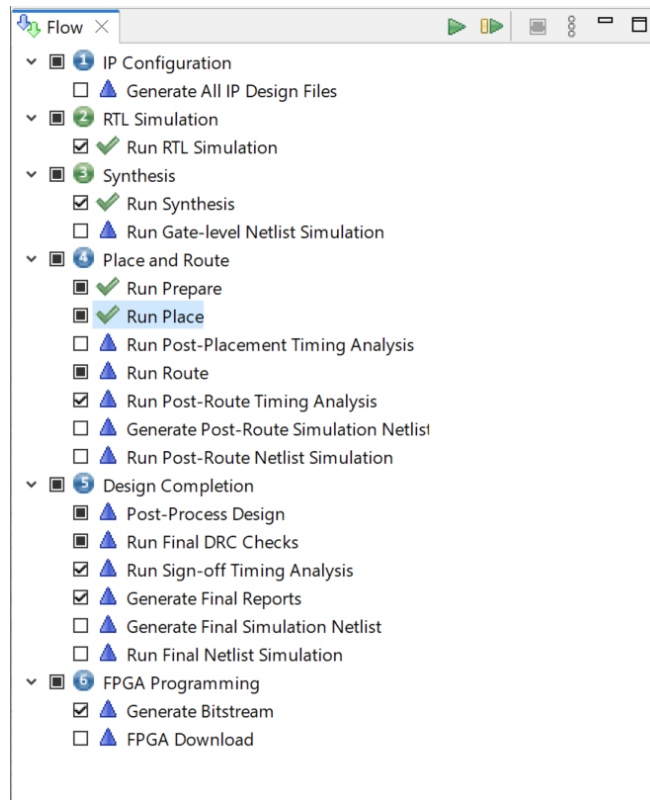


Figure 14 • Flow View Example

Table 30 • Flow View Icons

State	Flow Category	Flow Step
Incomplete		
Running		
Complete		
Disabled		
Warning		
Error		

Note









If the () icon appears on a Flow category or Flow step, this typically means ACE has detected changes to project source files, where the current source files on disk no longer match the design currently in memory. See [Detecting Changes to Project Source Files \(page 324\)](#).

Table 31 - Flow View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Run Flow	Y		Runs all the enabled flow steps sequentially from the beginning of the flow.
	Resume Flow	Y		Resumes running the flow from the last completed flow step. If no flow steps have been attempted yet, this action behaves identically to Run Flow .
	Show Multiprocess View	Y		Launches the Multiprocess View (page 73) , which allows managing multiple runs of the flow in parallel.
	Stop flow ⁽¹⁾	Y	Y	Stops the execution of any flow steps after the currently running flow step. Also attempts to interrupt the currently running flow step if possible.
	Run Selected Flow Step ⁽²⁾		Y	Runs the selected flow step, also running any required preceding flow steps that have not yet run. Preceding flow steps that are enabled but not required are skipped.
	Re-Run Flow		Y	Runs all the enabled flow steps sequentially from the beginning of the flow. Behavior is now identical to Run Flow .
	Re-Run Flow with <code>-i c init</code> ⁽³⁾		Y	Behaves identically to Re-Run Flow , unless Incremental Compilation is enabled. If Incremental Compilation is enabled, this additionally forces a full recompile of all partitions; any prior partition state is ignored (and overwritten).
	Clear Flow ⁽⁴⁾		Y	Issues a clear_flow (page 617) TCL command. All flow categories and flow steps with the state of Complete or Error are reset to the state of Incomplete. Additionally, the state of the current active project and implementation are cleared, as if they had not yet been run through the flow.
	Create Flow Step ⁽⁵⁾		Y	Displays an interactive dialog for creating a user-defined flow step (the dialog includes a prompt for which a single Tcl command should be invoked for this step) at the selected location within the flow.

Icon	Action	Toolbar Button	Context Menu	Description
	Remove Flow Step		Y	Removes a selected user-defined flow step. Only steps the user has created with Create Flow Step may be removed; this action cannot be used to remove "reserved" steps.

Table Notes

1. Some flow steps, such as FPGA Download, are currently unable to be interrupted while running.
2. Double-clicking a flow step is equivalent to this action.
3. **Caution:** This action is only relevant when using [Incremental Compilation \(Partitions\)](#) (page 404). If Incremental Compilation is disabled, this action behaves identically to **Re-Run Flow**.
4. Clear Flow does not remove any prior saved state from the hard drive. Any prior saved state may subsequently be (re-)loaded, including any partition state for incremental compilation.
5. **Caution:** Creation of user-defined flow steps is only recommended for advanced users.

Warning!**Current Flow Mode Setting Impacts Which Flow Steps Are Executed**

The implementation option for [Flow Mode](#) (page 242) affects which flow steps are executed during the **Run Flow**, **Resume Flow**, **Re-Run Flow**, and **Re-Run Flow** using `-ic init` actions (or related Tcl commands).

If warnings or errors are encountered while a flow step is running, they appear in the Flow view next to the name of the flow step where they occurred.

The colors used for the warning and error text are the same as those used to display warnings and errors in the [Tcl Console view](#) (page 142). They can be adjusted through the Colors and Fonts link on the [Tcl Console Preference Page](#) (page 216).

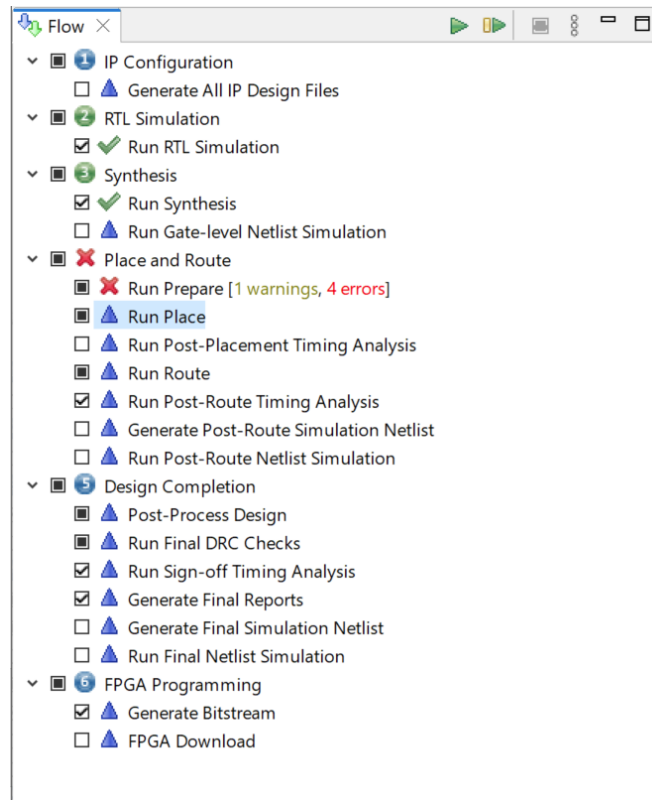


Figure 15 • Flow View Warnings and Errors Example

Clicking the colored "warning" or "error" text brings up a dialog listing all problems.

Clicking any of the problems in the list opens the corresponding log file in an editor, scrolled to the location of the problem in question.

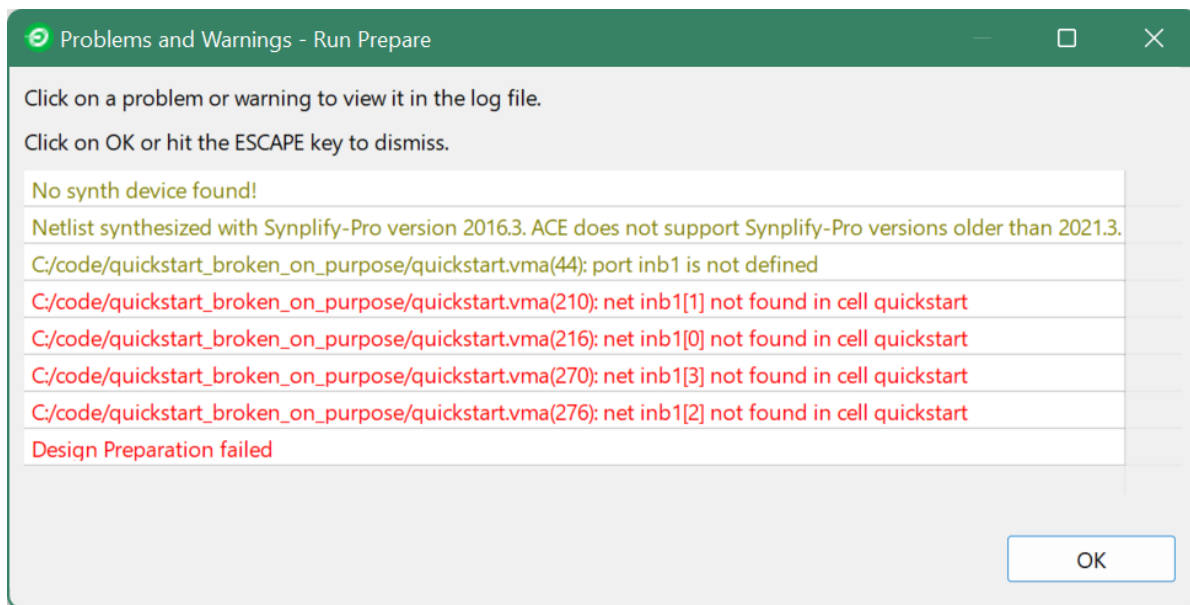


Figure 16 • Warning and Error List Dialog Example

HW Demo View

Warning!

- **The JTAG connection must be configured before using the HW Demos.**
ACE interacts with the FPGA using the JTAG interface through a Bitporter pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the HW Demo functionality. The configuration is managed using the [Configure JTAG Connection Preference Page \(page 190\)](#). See [Configuring the JTAG Connection \(page 360\)](#) for more details.
- **The DCC connection must be configured before using the HW Demos.**
ACE interacts with the HW Demo designs (and reference designs) using the DCC interface to the FPGA through a USB cable (not the Bitporter). This interface must be properly configured in ACE before using the HW Demo functionality. The configuration is managed using the [Configure DCC Connection Preference Page \(page 189\)](#). See [Configuring the DCC Connection \(page 358\)](#) for more details.

The HW Demo view provides a graphical interface for demonstrating particular aspects of a user selected device, using provided sample designs. These sample designs are typically provided as self-documenting overlays for the standard ACE installation.

By default, the HW Demo view is included in the [HW Demo Perspective \(page 6\)](#). To access the HW Demo view from any other perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **HW Demo**.

Before any demo overlays are installed, there are no demo designs available. In that case, the view will display minimal information:

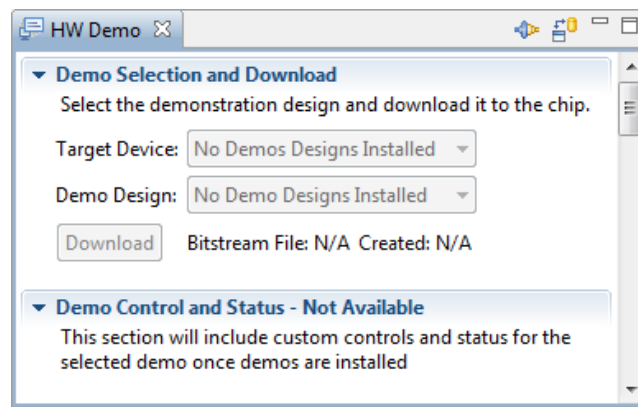


Figure 17 • HW Demo View with No Demo Designs Example

After demos are installed as ACE overlays, from this view the status of various board components can be observed updating in real-time as the demonstration design is running on the connected board. For example, in a basic fabric demonstration, when an associated DIP switch is changed, it is reflected in the view display; likewise when an LED on the board changes state (on/off) it is reflected in the view. Individual memory locations may be read and examined, and new values may be entered and "pushed" out to the target device.

Most hardware demos (including reference designs) are designed to show the features of the various hard IP blocks integrated into the target Achronix FPGA. Each demo or reference design installation package comes with associated documentation specific to the design.

Talk to your Achronix Marketing contact or FAE to request access to the demos appropriate to your development board.

See also: [Running the HW Demo \(page 401\)](#).

Table 32 • HW Demo View Toolbar Buttons

Icon	Action	Description
	Configure DCC Interface	Opens the preferences dialog to the Configure DCC Connection Preference Page (page 189) .
	Configure JTAG Interface	Opens the preferences dialog to the Configure JTAG Connection Preference Page (page 190) .

Table 33 • HW Demo View Options

Option	Description
Demo Selection and Download	

Option	Description
Target Device	List of FPGA devices that have demonstration designs.
Demo Design	List of demonstration designs for the currently selected device.
Download	Loads the currently selected demonstration design into the attached board.
Board Status	
LED State	Visually represents relevant LEDs from the attached board. When an LED changes state on the board, it is reflected in the view LED display. Clicking an individual LED in the view causes the corresponding LED on the attached board to toggle its state.
DIP Switch State	Visually represents the eight DIP switches from the attached board. When a switch changes state on the board, it is reflected in the DIP switch display. Clicking an individual switch in the view does <i>not</i> cause the corresponding switch on the attached board to toggle its state.
Device State	Displays DCC connection status and demo version number.
Demo Control and Status	
Each specific demonstration design has a simple user interface that is presented in the bottom section of the view. An example interface might provide a facility for reading and writing values to user specified addresses.	

Note

The Board Status section might not be present in all HW Demo (and reference) designs.

I/O Designer Toolkit Views

The I/O Designer Toolkit views provide a set of fully-integrated I/O ring design tools. With these tools it is possible to:

- Combine I/O ring IP configuration (.acxip) files into a complete I/O ring design
- View and update dynamic I/O ring resource utilization
- View and update (using drag and drop) dynamic I/O ring floorplan layout
- View dynamic I/O ring package ball layout and pin assignment report
- Automatically complete the following in real time:
 - Full I/O ring final DRC
 - Full I/O ring timing closure
 - Full I/O ring place and route

- Generate complete package ball pin assignment, power, and utilization reports
- Generate a full I/O ring simulation model that is 100% correctly configured, and has wrappers tailored specifically to the user design
- Generate pin placements PDC, Verilog wrappers, and port lists for the core user design
- Generate the full I/O ring bitstream
- Quickly and easily combine existing I/O ring IP configuration (.acxip) files from an existing ACE project into new ACE projects to create multiple designs

⚠ Caution!


The I/O Designer views and features are only applicable to specific Achronix Speedster7t FPGAs, such as the Speedster7t AC7t1500.

I/O Designer Toolkit Views

The views in the I/O Designer Toolkit are:

- [I/O Utilization View \(page 62\)](#)
- [I/O Package Diagram View \(page 63\)](#)
- [I/O Pin Assignment View \(page 64\)](#)
- [I/O Core Pin Assignment View \(page 65\)](#)
- [I/O Layout Diagram View \(page 67\)](#)

I/O Ring Design File Generation


Clicking the  **Generate I/O Ring Design Files** toolbar button opens the [Generate I/O Ring Design Files Dialog \(page 171\)](#), which allows selection of the output directory for all the customized I/O ring design files, including:

- Complete package ball pin assignment, power, and utilization reports
- Pin placements PDC, Verilog wrappers, and port lists for the core user design
- The full I/O ring bitstream, which is automatically combined with the core user design bitstream in ACE at the end of the normal place-and-route flow for the core user design
- Customized I/O ring simulation files, including Verilog wrappers for the top-level and I/O ring configuration data

i Note**Batch Mode Support**

I/O ring design files may also be generated in batch mode for a given ACE project by calling the [generate_ioring_design_files \(page 641\)](#) Tcl command. This command loads up all the I/O ring IP configuration (.acxip) files from an existing ACE project, and performs full design rule checks prior to generating the output files. I/O ring IP configuration files can also be edited in a text editor to support batch mode configuration prior to calling the `generate_ioring_design_files` Tcl command.

Table 34 · I/O Designer Toolbar Buttons

Icon	Description
	Opens the Generate I/O Ring Design Files Dialog (page 171), which allows selecting the directory into which the customized I/O ring design files are generated.

See also [Creating an IP Configuration](#) (page 336) and [Adding Source Files](#) (page 297).

I/O Utilization View

The I/O Utilization view provides a combined utilization summary of the active ACE project I/O ring IP configuration (.acxip) files, including shared resources such as clocks. Each resource type is summarized in a table inside an expandable section. These tables can be used for navigating between various IP configuration files in the project. Configuration errors are also summarized in the Status column for each row. Right-clicking a row brings up a context menu of actions that can be performed on the IP in that row. Double-clicking the table row opens the source IP configuration file for the data in that row.

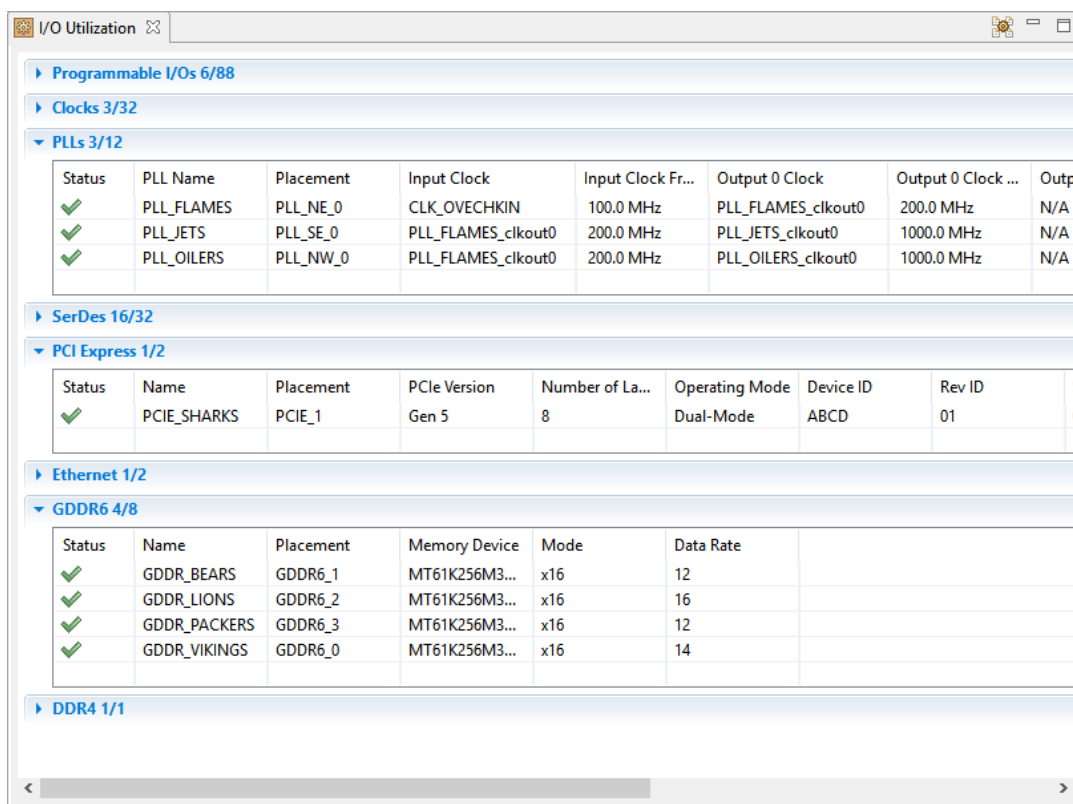






Figure 18 · I/O Designer View (Utilization Tab)

Table 35 - I/O Designer View Actions

Icon	Action	Description
	Open IP	Opens the selected IP file in an editor within ACE.
	Clone IP	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP	Renames the selected IP.
	Remove IP from project	Allows removal of the selected IP project. See also remove_project_ip (page 675).
	Show in file manager	Opens the operating system default file manager to the directory containing the IP file.

I/O Package Diagram View

The I/O Package Diagram view shows a live diagram of the target package balls and all I/O ring user design top-level pin ball assignments. Click and drag to pan the diagram and use the mouse wheel to zoom in and out. Package balls with yellow fill indicate placed user design pins on those package balls. Tooltip text provides extra information about each package ball location.

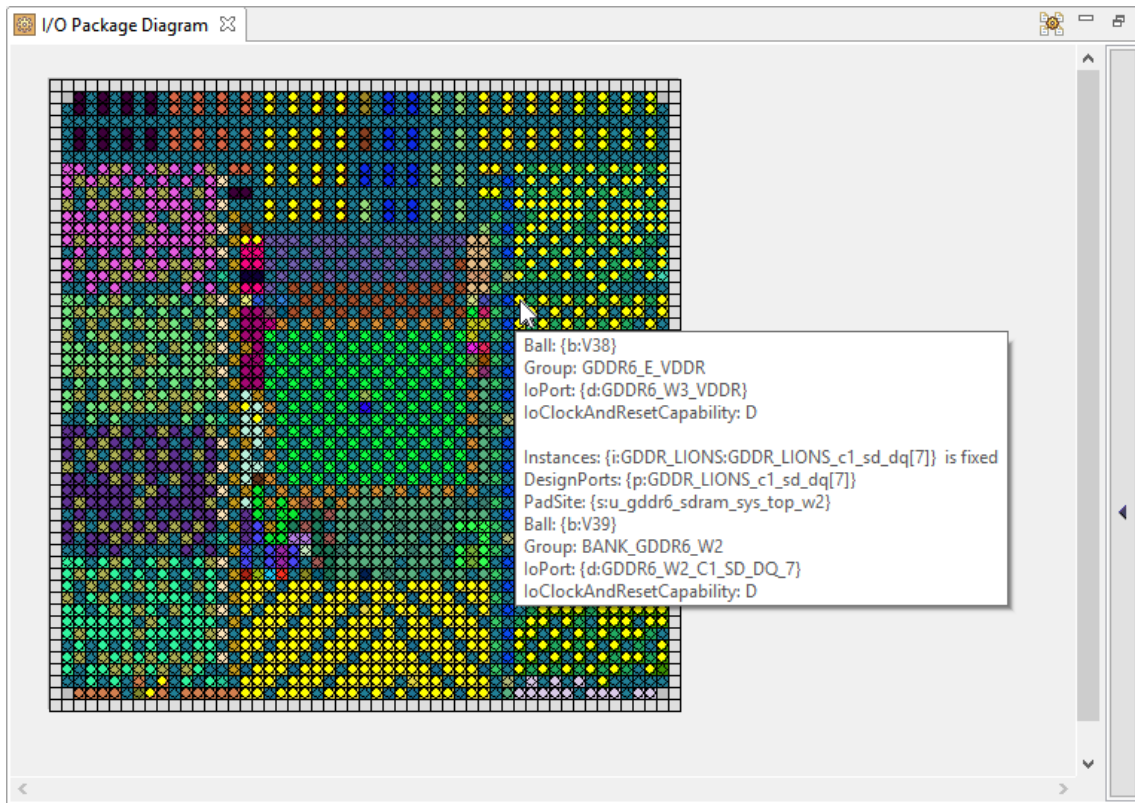


Figure 19 - I/O Package Diagram View

I/O Pin Assignment View

The I/O Pin Assignment view shows a live table of I/O ring user design top-level pin assignment information, including user design port name, I/O bank, package ball, top-level device port name, and pad/macro site name (for debugging in the full-chip simulation hierarchy). Columns can be sorted by left-clicking on the column headers. Columns can be filtered using the **Toggle Filter Row Visibility** button.

Used	Port Name	Remapped Name	Direction	Bank	Ball	Device Port	Pad/Macro Site	Site Polarity	Clock Capable Site	Reset Capable Site	Data Capable Site	Used as Clock	Used as Reset
✓	BACKSTROM_pad_n		IN	BANK_GPIO_S0_BY...	BL8	GPIO_S0_BYTE0_BIT_5	u_gpio_phy_h_36_top_s0_u...	N	Y	N	N	Y	N
✓	BACKSTROM_pad_p		IN	BANK_GPIO_S0_BY...	BK9	GPIO_S0_BYTE0_BIT_4	u_gpio_phy_h_36_top_s0_u...	P	Y	N	N	Y	N
✓	CLK_OVECHKIN_pad_n		IN	BANK_CLKIO_NE	N17	CLKIO_NE_REFIO_N_0	u_glb_clk_rst_gen_top_ne_u...	N	Y	N	N	Y	N
✓	CLK_OVECHKIN_pad_p		IN	BANK_CLKIO_NE	N16	CLKIO_NE_REFIO_P_0	u_glb_clk_rst_gen_top_ne_u...	P	Y	Y	N	Y	N
✓	DATA_HOLTBYP_pad_n		IN	BANK_GPIO_N0_B...	AH17	GPIO_N0_BYTE1_BIT_3	u_gpio_phy_v_36_top_n0_u...	N	N	N	Y	N	N
✓	DATA_HOLTBYP_pad_p		IN	BANK_GPIO_N0_B...	AG16	GPIO_N0_BYTE1_BIT_2	u_gpio_phy_v_36_top_n0_u...	P	N	N	Y	N	N
✓	DDR4_BRUINS_a17		INOUT	BANK_DDR4_S0	BB25	DDR4_S0_A17	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a10		INOUT	BANK_DDR4_S0	B127	DDR4_S0_A_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[11]		INOUT	BANK_DDR4_S0	B128	DDR4_S0_A_10	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[12]		INOUT	BANK_DDR4_S0	BK25	DDR4_S0_A_11	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[13]		INOUT	BANK_DDR4_S0	BE25	DDR4_S0_A_12	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a1		INOUT	BANK_DDR4_S0	BH30	DDR4_S0_A_13	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[2]		INOUT	BANK_DDR4_S0	BE27	DDR4_S0_A_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[3]		INOUT	BANK_DDR4_S0	BH26	DDR4_S0_A_2	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[4]		INOUT	BANK_DDR4_S0	BF26	DDR4_S0_A_3	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[5]		INOUT	BANK_DDR4_S0	BK26	DDR4_S0_A_4	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[6]		INOUT	BANK_DDR4_S0	B126	DDR4_S0_A_5	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[7]		INOUT	BANK_DDR4_S0	BE26	DDR4_S0_A_6	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[8]		INOUT	BANK_DDR4_S0	BL25	DDR4_S0_A_7	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_a[9]		INOUT	BANK_DDR4_S0	BG25	DDR4_S0_A_8	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_act_n		INOUT	BANK_DDR4_S0	BF25	DDR4_S0_A_9	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_act_p		INOUT	BANK_DDR4_S0	BL24	DDR4_S0_ACT_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ba[0]		INOUT	BANK_DDR4_S0	BK28	DDR4_S0_BA_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ba[1]		INOUT	BANK_DDR4_S0	BF28	DDR4_S0_BA_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bg[0]		INOUT	BANK_DDR4_S0	BG24	DDR4_S0_BG_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bg[1]		INOUT	BANK_DDR4_S0	BF24	DDR4_S0_BG_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bp_alert_n		INOUT	BANK_DDR4_S0	B125	DDR4_S0_BP_ALERT_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_bp_memreset_l		OUT	BANK_DDR4_S0	BH24	DDR4_S0_BP_MEMRESET_L	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cas_n		INOUT	BANK_DDR4_S0	BL29	DDR4_S0_CAS_N	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[0]		INOUT	BANK_DDR4_S0	B130	DDR4_S0_CID_0	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[1]		INOUT	BANK_DDR4_S0	BF30	DDR4_S0_CID_1	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_cid[2]		INOUT	BANK_DDR4_S0	BC28	DDR4_S0_CID_2	u_ddr4_sdram_sys_top_s0	--	N	N	Y	N	N
✓	DDR4_BRUINS_ck_n[0]		INOUT	BANK_DDR4_S0	BH27	DDR4_S0_CK_N_0	u_ddr4_sdram_sys_top_s0	N	Y	N	N	Y	N

Figure 20 • I/O Pin Assignment View

Table 36 • I/O Pin Assignment View Buttons

Icon	Description
	Generate I/O Ring Design Files.
	Clear Sorting.
	Toggle Filter Row Visibility.
	Remap Port/Signal Name (available in Remapped Name column right-click menu).

I/O Core Pin Assignment View

The I/O Core Pin Assignment view shows a live table of I/O ring user design top-level core pin assignment information, including user design signal name, direction, data type, group, and core pin name. Columns can be sorted by left-clicking the column headers. Columns can be filtered using the **Toggle Filter Row Visibility** button.

Signal Name	Remapped Name	Direction	Data Type	Group	Core Pin Name
BACKSTROM		IN	Clock	GPIO	i_user_10_00_mt_00[2]
DATA_HOLTBY		IN	Data	GPIO	i_user_11_09_lut_15[4]
ETH_WILD_m0_ff_clk_divby2		IN	Clock	Clocks and Resets	i_user_02_09_mt_00[0]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[27]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[25]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[1]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[0]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_14[24]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[4]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[2]
ETH_WILD_m0_pause_on		IN	Data	400G MAC 0 Flow Control	i_user_02_09_lut_15[5]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[26]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[25]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[24]
ETH_WILD_m0_rx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[23]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[20]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[19]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[21]
ETH_WILD_m0_rx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[18]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[3]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[4]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[5]
ETH_WILD_m0_rx_buffer2_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[6]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_18[26]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_18[27]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[0]
ETH_WILD_m0_rx_buffer3_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[1]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[13]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[14]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[15]
ETH_WILD_m0_tx_buffer0_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[16]
ETH_WILD_m0_tx_buffer1_at...		IN	Data	Buffer Levels	i_user_02_09_lut_19[11]

Figure 21 • I/O Core Pin Assignment View

Table 37 • I/O Core Pin Assignment View Buttons

Icon	Description
	Generate I/O Ring Design Files.
	Clear Sorting.
	Toggle Filter Row Visibility.
	Remap Port/Signal Name (available in Remapped Name column right-click menu).

I/O Layout Diagram View

The I/O Layout Diagram view shows an interactive floorplan of the target device. Empty IP Sites are shown in white. Sites with legally-placed IP are shown in green. Sites with IP placement overlap violations are shown in red.

IP can be moved from one site to another by dragging and dropping. IP can be cloned onto another compatible site by holding down the CTRL key while dragging and dropping. Double-clicking a placed IP opens an editor for that IP. Double-clicking an empty site opens the "Create new IP" dialog, preselecting the appropriate type of IP for that site, and placing the new IP at that site when the dialog is completed.

Changes to placement in the diagram result in updates to the source IP configuration (.acxip) files. Tooltip text provides extra information about each IP site. Right-clicking a site brings up a context menu of actions that can be performed on that site, or the IP placed on that site.

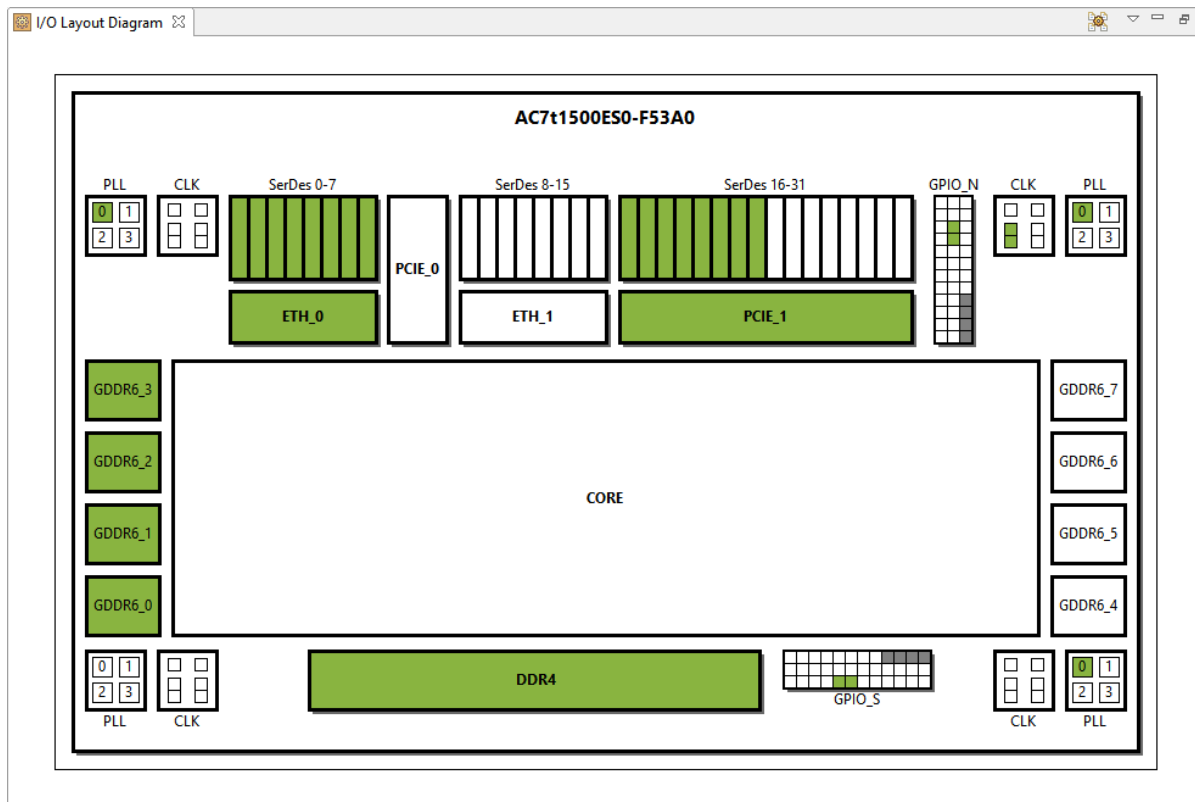






Figure 22 - I/O Designer View (Layout Tab)

Table 38 - I/O Designer View Actions

Icon	Action	Description
	Open IP	Opens the selected IP file in an editor within ACE.

Icon	Action	Description
	Create new IP here	Creates a new IP at the chosen site.
	Clone IP	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP	Renames the selected IP.
	Add IP to another project...	Adds the selected IP to another project in the ACE workspace.
	Add copies of IP to another project...	Adds a copy of the selected IP to another project in the ACE workspace.
	Remove IP from project	Allows removal of the selected IP project. See also remove_project_ip (page 675)

By default, if one or more IP editors are currently open, the diagram is configured to display a yellow highlight indicating the currently active IP editor.

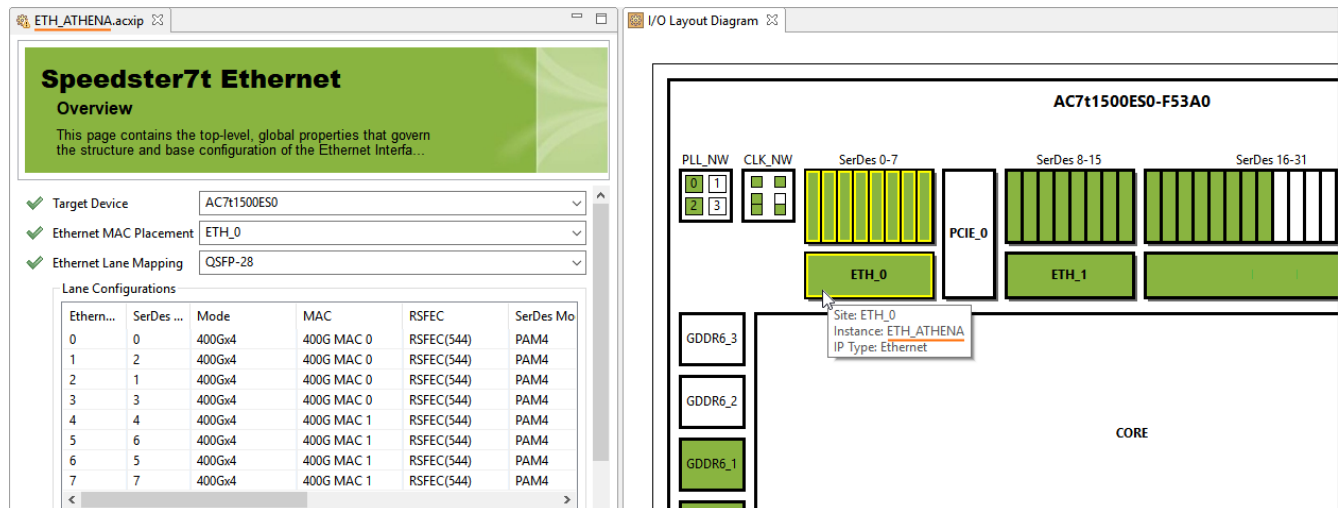


Figure 23 - I/O Layout Diagram View Currently Active Highlight Example

The **I/O Designer** page in the Preferences can be used to adjust the highlight color, or to hide it.

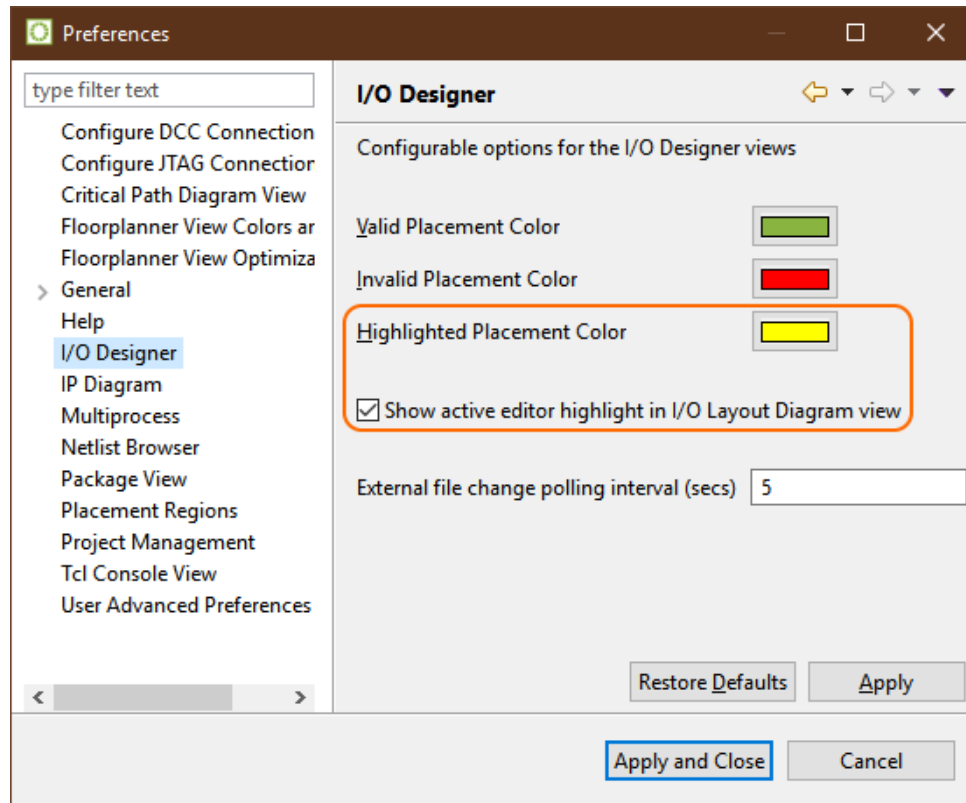


Figure 24 · I/O Designer Highlighted Placement Color Preference Example

IP Diagram View

The IP Diagram view provides a graphical visualization of the configuration of the IP currently being edited. As different IP configurations are selected via their editor, the IP Diagram view contents change to reflect the selected IP configuration.

Some IP supports multiple pages of diagrams (e.g., a logic block diagram page and a placement diagram page). In these cases, there are multiple labeled tabs at the bottom of the IP Diagram view to allow switching diagram pages. The following is an example of an IP diagram showing a `din` input and `dout` output indicating configuration errors:

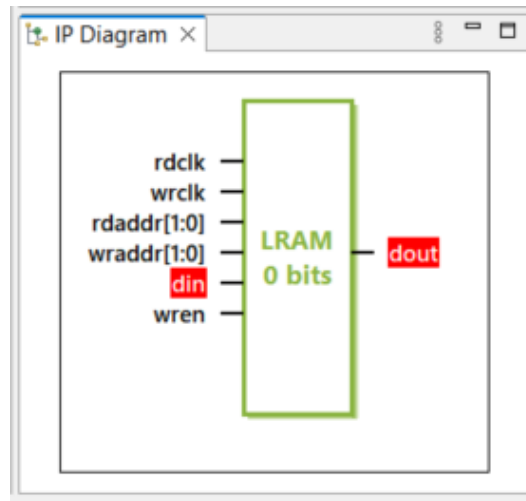




Figure 25 • Example IP Diagram Indicating Configuration Errors

When a supported IP Configuration editor is selected, the IP Diagram view shows a dynamic block diagram of the selected IP. Displayed labels change, and logic blocks may appear and disappear depending upon the configuration options currently selected in the IP editor. Tool tips are available on all text displayed in the IP Diagram. Text representing configuration options with warnings or errors are displayed with appropriate colors to indicate the condition.

By default, in the diagram, warnings have a yellow background and errors have a red background, though all colors and fonts used within the diagram may be overridden from the [IP Diagram Color and Font Preferences Section \(page 205\)](#), easily accessible from the () **Configure Colors and Fonts...** view menu item, found under the () vertical ellipsis menu button in the upper-right of the view.

Clicking any text label in the IP Diagram immediately turns the IP editor to the associated page, scrolls the associated option into view and gives it focus, so that the related configuration options may be edited.

See also: [Creating an IP Configuration \(page 336\)](#)

Note

If the selected editor is not an IP Configuration editor, or if the selected IP does not support a diagrammatic visualization, the IP Diagram view displays a notice that there is no diagram available for the selected editor as shown in the following example.

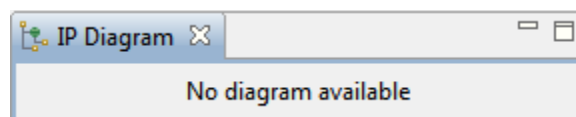


Figure 26 • Example IP Diagram View With Notice of No Available Diagram

IP Libraries View

The IP Libraries view provides an alternate method for creating IP configuration .acx ip files versus using the main menu, **File** → **New** → **IP Configuration....** The available IP listed varies depending on the current target device chosen for the active project or implementation. For example, targeted Speedster7t FPGAs have I/O ring IP choices displayed in their library tree, while Speedcore eFPGAs do not. Expanding a device family name (IP library) displays a list of available IP types for that family. Double-clicking the IP type or clicking the **Create New IP Configuration** button, opens the [new IP Configuration dialog \(page 176\)](#) to begin IP creation/configuration.

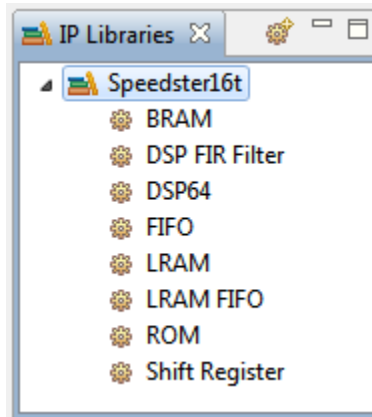


Figure 27 • IP Libraries View Example

Note

The displayed IP libraries and IP types are dynamic; these change based on which technology libraries and devices are installed and licensed, and on which **Target Device** was chosen for the **active project and implementation (page 229)** when **configuring project and implementation options (page 304)**. The screenshots and example descriptions in this section do not necessarily reflect the IP types of the actual target devices currently in use on the site.

Table 39 • IP Libraries Toolbar Buttons

Icon	Description
	Opens the new IP configuration dialog (page 176) to allow creating a new IP configuration file.

See also: [Creating an IP Configuration \(page 336\)](#).

IP Problems View

The IP Problems view displays all warnings and errors for all of the currently open IP Configuration editors.

The top half of the view displays a sorted tree table of all errors in order by IP configuration .acx ip file, then all warnings in order by file. When an IP problem is selected in this tree table, further details about the problem are displayed under the tree table in the bottom half of the view.

Double-clicking an error or warning opens the relevant IP Configuration editor to the appropriate page. See also [Creating an IP Configuration](#) (page 336).

Note

Unlike other IP-related views, this view shows information for all open IP Configuration editors, not just the top/active editor.

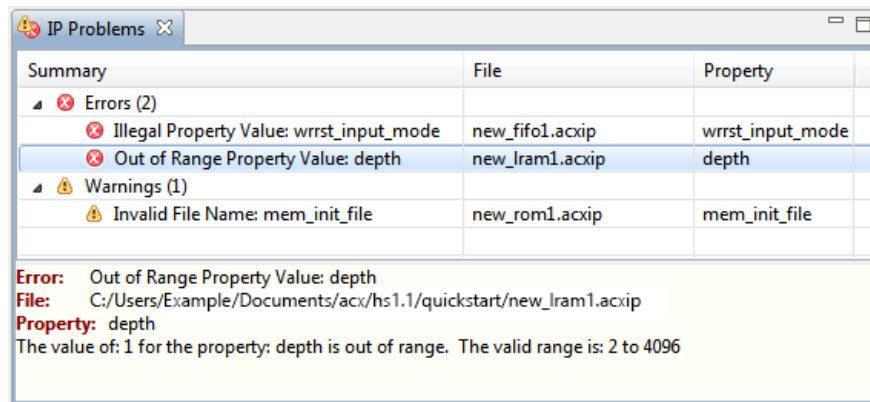


Figure 28 · IP Problems View Example

Table 40 · IP Problems View Icons


Icon	Description
	Warning
	Error

Table 41 · IP Problems View Table Columns

Column Name	Description
Summary	A brief summary statement of the IP configuration problem.
File	The name of the IP Configuration file in an open IP Configuration editor containing the error.

Column Name	Description
Property	The property which is part of the IP Configuration problem. Individual properties are usually similar to the field names shown in the IP Configuration editor. The raw properties and their values can be viewed in the editor by selecting the File Preview tab at the bottom of each editor. The Configuration tab shows a more user-friendly representation of the same data.

Multiprocess View

Similar to the Tcl command, [run_multiprocess \(page 692\)](#), the () Multiprocess view allows **running multiple flows in parallel (page 308)** and **attempting likely optimizations using option sets (page 392)**.

The Multiprocess view provides a means to select multiple [Implementations \(page 229\)](#) within a single [project \(page 222\)](#) for flow execution. Depending upon how this view is configured, the selected implementations may be queued for sequential flow execution, run all at the same time in parallel, or a combination of these in a configurable number of parallel sequential queues. The selected implementations may be executed in the background of the workstation running ACE, or optionally may be sent to an external cloud, grid or batch job system for execution.

The Multiprocess view may also help to explore the solution space provided by various ACE optimizations. The Multiprocess view can optionally generate new implementations derived from the current [active project and implementation \(page 229\)](#), where each newly generated implementation applies an overlay of likely [implementation option \(page 0\)](#) optimizations over the active implementation options. These collections of potentially optimized implementation options are termed "[option sets \(page 0\)](#)".

By default, the Multiprocess view is a part of the [Projects perspective \(page 6\)](#). To make the Multiprocess view visible from within any perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Multiprocess**.

This view is broken up into several sections:

- Execution Queue Management
- Multiprocess Flow Management
- Select Implementations
- Multiprocess Run Logs

Each section includes a brief descriptive paragraph describing its purpose. Each section may be collapsed and expanded by clicking the section title. Collapsing or expanding any section causes the other sections to be resized to fit the available data and view area.

For more detailed information on how to use this view, please see [Running Multiple Flows in Parallel \(page 308\)](#) and [Attempting Likely Optimizations Using Option Sets \(page 392\)](#).

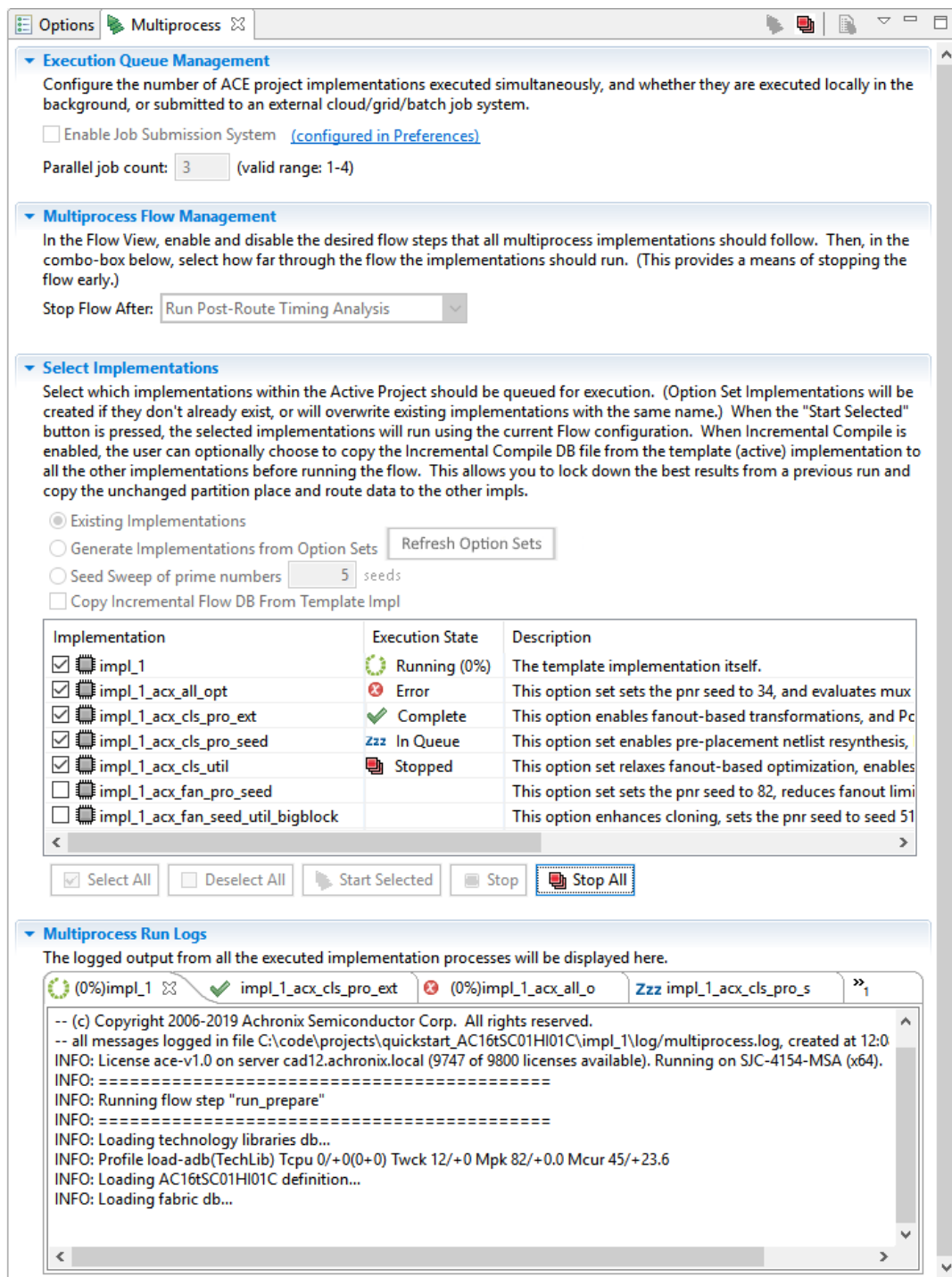





Figure 29 • Multiprocess View Example

Table 42 • Multiprocess View Toolbar Buttons

Icon	Action	Description
	Start Background Queue Execution	Starts execution of all implementations selected in the Select Implementations table in the number of parallel processes specified by Parallel Queue Count .
	Stop All Background Queue Execution	Stops or cancels execution of all currently running or queued implementations.
	Open Multiprocess Report	Opens the Multiprocess Summary report (page 255) for the selected project.

Execution Queue Management

This section configures the number of background processes allowed to run in parallel, and how/where they are executed.

Table 43 • Execution Queue Management Controls

Name	Description
Parallel Job Count	Sets the number of implementations allowed to execute in parallel. Defaults to 2 . When in background mode, the maximum allowed value is the number of available processor cores detected. When in Job Submission System mode, the maximum allowed value is 99.
Enable Job Submission System Support	When unchecked, background processes run locally on the workstation currently running the ACE GUI. When checked, ACE uses the cloud/grid/batch job submission system as configured in the preferences.
(configured in Preferences)	When selected, brings up the Multiprocess: Configure Custom Job Submission Tool preference page (page 206) to fully configure which cloud/grid/batch job submission system is used.

When the **Parallel Job Count** is set to the minimum value of **1**, all selected implementations are executed sequentially, one at a time. A value of **2** causes all selected implementations to be queued, and then the first two queued implementations are allowed to execute at the same time. As soon as an implementation completes its flow execution, the next queued implementation starts flow execution and the **Multiprocess Summary Report** (page 255) is updated with information gathered from the just-completed implementation.

By default, ACE executes implementations in parallel by starting a background process on the host workstation for each implementation (termed "background mode"). In this case, the effectiveness of parallel implementation

execution is naturally limited by the resources of the host workstation (i.e., the number of processor cores and the physical RAM).

Alternately, ACE may execute the implementations in processes distributed among multiple hosts via an external job submission system, which theoretically allows for far greater parallel compute resources. The job submissions are performed through a user-configured command line executable. This executable is configured via the **Multiprocess: Configure Custom Job Submission Tool preference page** (page 206), reached easily by following the <configured in Preferences> hyperlink.

See **Running Multiple Flows in Parallel** (page 308) for important details regarding parallel implementation execution, configuration, and external job submission tool support.

Multiprocess Flow Management

This section allows altering how far the flow is executed for the multiprocess implementations.

Table 44 • Multiprocess Flow Management Controls

Name	Description
Stop Flow After ⁽¹⁾	Allows overriding standard flow behavior and stopping the flow early — the flow step selected becomes the final flow step executed by all multiprocess implementations. Useful when steps late in the flow are known to fail with reported errors, but it is still desired to run multiple implementations through earlier parts of the flow.
<p>Table Notes</p> <p>1. The flow step chosen here is always enabled when the multiprocess run executes, regardless of whether it was enabled before the multiprocess run is launched.</p>	

See **Running Multiple Flows in Parallel** (page 308) for further details regarding multiprocess flow configuration.

Select Implementations



This section allows:


- Selecting which implementations to execute (implementations derived from **option sets** (page 0) are created if selected).
- Starting or stopping the execution of all selected implementations.
- Providing simple execution state feedback.

See **Running Multiple Flows in Parallel** (page 308) for further details regarding selecting the implementations to be run in parallel, starting/stopping/cancelling parallel execution, etc.

See **Attempting Likely Optimizations Using Option Sets** (page 392) for explanations of how to use option sets to achieve better QoR.

Table 45 • Select Implementations Controls

Name	Description
Existing Implementations	Updates the contents of the implementation table to show all existing implementations for the current active project (page 229).
Generate Implementations from Option Sets	Updates the contents of the implementation table to show the current active implementation (page 229) and a number of to-be-generated implementations, one per option set (page 0). The use of this radio button selection is covered in more detail in Attempting Likely Optimizations Using Option Sets (page 392). If the implementation table does not show any implementations besides the active implementation while in this mode, click the Refresh Option Sets button.
Refresh Option Sets ⁽¹⁾	Causes ACE to analyze the current active project and implementation (page 229) to (re-)generate customized option sets most likely to improve GoR. The implementation table is then updated with a list of to-be-generated implementations.
Seed Sweep of prime numbers	Updates the contents of the implementation table to show a number of to-be-generated implementations. Each of these implementations is identical to the currently active implementation, with the implementation option "seed" being automatically set to the next consecutive prime number. The seedcount text field beside this radio button can be used to choose how many such implementations should be created.
Implementation Table	A table containing implementation names along with their selection state and execution state. The implementations listed vary based upon the active project and implementation (page 229), in combination with the state of the radio buttons.
<input checked="" type="checkbox"/> Select All	Selects all implementations in the implementation table.
<input type="checkbox"/> Deselect All	Deselects all implementations listed in the implementation table.
 Start Selected	Queues all implementations selected in the implementation table and begins executing in the configured number of parallel processes.
 Stop All	If clicked, all currently queued implementations are removed from the queue(s) and all currently executing implementations are killed. The Multiprocess Summary Report (page 255) is updated with any and all captured information.

Name	Description
<p>Table Notes</p> <ol style="list-style-type: none"> 1. This button must be clicked prior to clicking the () Start Selected button whenever the active project or implementation has changed significantly, as well as the first time Generate Implementations from a new active project or implementation is chosen. 	

All the controls in this section center around what is in the table. The radio buttons change which implementations are listed in the table, and the push-buttons under the table change the selection state of the listed implementations, or alter the execution state of the implementations which is the purpose of the entire view.

The table contents are kept in sync with the current **active project and implementation** (page 229). Changing active projects (which implicitly changes active implementations) updates the implementation table contents according to the current radio button selection.

The following table describes the columns in the implementation table.

Table 46 • Implementation Table Columns

Column Name	Description
Implementation	Contains the implementation name, along with a checkbox indicating implementation selection, and an icon representing the implementation.
Execution State	Contains the execution state of the implementation.
Description	Blank when Existing Implementations is selected. When Generate Implementations from Option Sets is selected, contains a description of the option set (page 0) which is used as the overlay on the active implementation (page 229) when generating the new implementation.







 **Tip**

If the implementation table is not large enough (or is too large) for the full implementation list, click the section title and simply collapse and/or expand one of the other sections in this view. This causes the table to resize to exactly fit the entire current implementation list.

Implementation Execution States

There are a number of possible execution states, as listed in the second column, for the implementations in the table corresponding to the lifetime of a Multiprocess view background process. The icons from these states are also used on the tabs within the Multiprocess Run Logs section.

Table 47 • Implementation Execution States and Icons

Icon	Execution State	Description
	(blank)	Implementation has not been selected for execution.
	Selected	Implementation is currently selected for execution, and execution has not been started.
	In Queue	Execution of the selected implementations has been started, this implementation was selected for execution and is currently waiting in the queue for execution.
	Scheduled	Execution of the selected implementations has been started, this implementation was selected for execution, is at the head of the queue and is being prepared for execution. This state typically only lasts for a fraction of a second.
	Running	Implementation was selected for execution, and is currently executing. Log messages should be visible in the tabbed logging area.
	Complete	Implementation was (and still is) selected for execution, and its last execution was completed without flow errors, but does not mean that the design met timing. Log messages should be visible in the tabbed logging area. Summary information should be visible in the Multiprocess Summary Report (page 255) .
	Stopped	Implementation was (and still is) selected for execution, but its last execution was stopped or possibly canceled before it even started. If its execution had started, log messages should be visible in the tabbed logging area. If post-route timing analysis or sign-off timing analysis were completed for this implementation, the timing results should be visible in the Multiprocess Summary Report (page 255) .
	Error	Implementation was (and still is) selected for execution, but its last execution exited with reported errors. A tooltip for the error icon provides a summary of the detected error messages. Detailed log messages should be visible in the tabbed logging area. If post-route timing analysis or sign-off timing analysis were completed for this implementation, the timing results should be visible in the Multiprocess Summary Report (page 255) .

Multiprocess Run Logs

This section shows the logs for each selected implementation as they execute. A separate tab is provided for each individual implementation. The log info is updated live as background processes execute. Depending upon configuration, external cloud/grid/batch jobs may have their log info updated live, or it may not be updated until the job is completed. The displayed log info mirrors the information captured in the log file for each implementation.

Each tab includes the name of the implementation and the execution state, which updates live. If an implementation enters the error state, the tooltip for the tab title is updated to include a summary of the captured error messages. Error details are visible in the log shown in the tab, as well as within the [log files \(page 231\)](#) for each implementation.

Netlist Browser View

The Netlist Browser view provides a graphical, tree-based visualization of the user design hierarchy, as found in the netlist. The displayed netlist includes the results of any transformation, legalization, etc. that have happened through the current stage in the Flow.

For large designs, there are a tremendous number of objects in the netlist. To simplify the view, the Netlist Browser provides a number of ways to filter the flood of data down to just the most useful information (there are no filters active by default).

Each instance node in the tree includes the instance name and the cell type. Macros include the macro name, and the counts of the various major logic types contained within that macro. Be aware that these logic type counts are not affected by the filters. The numbers shown always represent the unfiltered total counts. Clock domain names and Partitions names also are listed when appropriate.

By default, the Netlist Browser view is included in the **Floorplanner perspective** (page 6). To add the Netlist Browser view to other perspectives, select **Window** → **Show View...** → **Other...** → **Achronix** → **Netlist Browser**.

As can be seen in the second column of the following example, three instances have been highlighted:

- inb1_ibuf[3].x_ipad_i_io_buff in cyan
- All members of the z0_obuf.* macro hierarchy in pink
- inb1_int_z[2] in dark blue

Instance Name	Hig...	Cell Type	Clock Dom...	Core	IORing	Partition	Flops	LUTs	ALUs	BRAMs	BMACCs	LRAMs	Others
inb1_ibuf[2]					✓		0	0	0	0	0	0	3
inb1_ibuf[3]					✓		0	0	0	0	0	0	3
x_ipad					✓		0	0	0	0	0	0	3
i_io_buff		io_buffer			✓		0	0	0	0	0	0	1
z0_obuf			clka		✓		0	0	0	0	0	0	3
inb1_int_Z[0]		DFF	clka	✓			1	0	0	0	0	0	0
inb1_int_Z[1]		DFF	clka	✓			1	0	0	0	0	0	0
inb1_int_Z[2]		DFF	clka	✓			1	0	0	0	0	0	0
inb1_int_Z[3]		DFF	clka	✓			1	0	0	0	0	0	0
reg_and_b1_RNO		LUT4	clka	✓			0	1	0	0	0	0	0
reg_and_b1_Z		DFFR	clka	✓			1	0	0	0	0	0	0
z0_PNO		LUT4	clka	✓			0	1	0	0	0	0	0

Figure 30 • Netlist Browser Example

Note

- The small colored square in the toolbar shows the active highlighting color. If highlighting is applied to a macro then all "child" instances within are also set to the current highlight color.
- Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The example images and descriptions in this section do not reflect the resource types of actual devices.





Table 48 • Netlist Browser Table Columns

Column Name	Description
Instance Name	The name of the instances in the netlist. Instances within a macro are grouped together as leaves under the macro branch. Additionally, an icon is used to indicate the placement state of the instance. The possible icons are shown in the following table.
Highlight Color	<ul style="list-style-type: none"> • For instances, shows a color square to indicate the instance highlight color, if any. • For macros, if all contained instances have the same highlight color, the macro shows a color square for that same highlight color. If even one contained instance has a different highlight color, or no highlight at all, the macro displays no color square. This value does not change for macros during filtering.
Cell Type	<ul style="list-style-type: none"> • For instances, shows the cell type of the instance. • For macros, this column is blank.
Clock Domain	<ul style="list-style-type: none"> • For instances, shows a list of all the clock domains of which the instance is a member. • For macros, shows a summary list of the clock domains for all the contained instances. This value does not change for macros during filtering.
Core	<ul style="list-style-type: none"> • For instances, this is checked if the instance is considered a member of the Core, or blank if it is not. • For macros, this is checked if any contained instances are considered a member of the Core, or blank if no contained instances are in the Core. This value does not change for macros during filtering.
IORing	<ul style="list-style-type: none"> • For instances, this is checked if the instance is considered a member of the IORing, or blank if it is not. • For macros, this is checked if any contained instances are considered a member of the IORing, or blank if no contained instances are in the IORing. This value does not change for macros during filtering.
Partition	<ul style="list-style-type: none"> • For instances, the name of the Partition to which the item belongs, if any. See Using Incremental Compilation (Partitions) (page 404). • For macros, (same).

Column Name	Description
Resource	<ul style="list-style-type: none"> For instances, this is one if the instance is of type <i>resource</i>, or zero otherwise. For macros, this is the sum count of all contained <i>resource</i> instances (regardless of filtering).

Icons decorate all the nodes in the tree in the Instance Name column.

Table 49 • Netlist Browser View Icons

Icon	Description
	Macro
	Unplaced Instance
	Placed Instance (Soft)
	Placed Instance (Fixed)



A number of actions are available in the view via:












- Buttons at the top of the view
- The (...) ellipsis view menu button
- Right-click context menus on the nodes of the tree



Note



If these actions are performed upon macros, all child leaf nodes, even those currently filtered to be hidden in the tree, are affected by the chosen action.

Table 50 • Netlist Browser View Actions

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Add to Selection		Y		Adds the item(s) to the ACE selection set (as shown in the Selection View (page 133)).
	Remove from Selection		Y		Removes the item(s) from the ACE selection set (as shown in the Selection View (page 133)).

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Choose Highlight Color	Y	Y		Determines which color is applied to the objects chosen from the tree the next time the highlight action is selected for this view.
	Highlight	Y	Y		Applies the currently active highlight color to the chosen item(s) in the tree. See Highlighting Objects in the Floorplanner View (page 343).
	Un-Highlight	Y	Y		Clears the Highlight for the chosen item(s) in the tree. When painted in the Floorplanner view, the chosen item(s) now use their default color(s) instead of a highlight color.
	Auto-Highlight	Y			Automatically applies unique highlight colors to all visible core hierarchy levels in the tree. IORing hierarchy levels are skipped.
	Zoom To		Y		Zooms the Floorplanner view to a region containing the instances currently chosen in the tree.
	Show in Netlist ⁽¹⁾		Y		Attempts to open a text editor to the file and line number relevant to the chosen instance. Available only when a single instance is chosen in the view.
	Unfix Placement of Instance		Y		Changes the state of an already-placed instance from fixed placement to soft placement. This choice is only available when an instance already has fixed placement.
	Fix Placement of Instance		Y		Changes the state of an already-placed instance from soft placement to fixed placement. This choice is only available when an instance already has soft placement.
	Unplace Instance		Y		Completely removes the site assignment for an instance, making it unplaced. This choice is only available when an instance is already placed.
	Expand All	Y			Expands all collapsed macro branches in the tree, making all leaf instances visible.
	Collapse All	Y			Collapses all expanded macro branches in the tree.



Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show Power and Grounds	Y		Y	<p>Disabled by default. When disabled, all instances of power and ground are hidden within the tree, effectively acting as a filter. Therefore, the Cell Type column gains the active filter indicator (yellow background by default) when power and ground are hidden.</p> <p>This filter is a higher priority than user custom filters applied to the Cell Type column. If a custom filter is created to expose only "GND" grounds, but Show Power and Grounds is disabled, the "GND" instances remain hidden.</p>
	Show Boundary Pins	Y		Y	<p>Disabled by default. When disabled, all instances of boundary pins are hidden within the tree, effectively acting as a filter. Therefore, the Cell Type column gains the active filter indicator (yellow background by default) when boundary pins are hidden.</p> <p>This filter is a higher priority than user custom filters applied to the Cell Type column. If a custom filter is created to expose only "OPIN" instances, but Show Boundary Pins is disabled, the "OPIN" instances remain hidden.</p>
	Show Feedthrough LUTs ⁽²⁾			Y	<p>Toggles the visibility of all feedthrough instances, most often created by Achronix optimizations. These always have "_ft_" plus some additional notation in the ACE-generated instance name.</p> <p>Enabled by default. When disabled, all feedthrough instances are hidden within the tree, effectively acting as a filter. Therefore, the Instance Name column gains the active filter indicator, a yellow background by default, when feedthroughs are hidden. In some cases, feedthrough instances might consist of more than just LUTs.</p> <p>This filter is a higher priority than the user custom filters applied to the Instance Name column. If a custom filter is created to expose only instances that contain "LUT" in the name, but Show Feedthrough LUTs is disabled, any instances that have a feedthrough-based name remain hidden, even if they have an explicit "LUT" in the name.</p>

Icon	Action	Toolbar Button	Context Menu	View Menu	Description
	Show Constants ⁽²⁾			Y	<p>Toggle the visibility of all instances with "const" somewhere in the name, most often created by Achronix optimizations.</p> <p>Enabled by default. When disabled, all instances with "const" anywhere in the name are hidden within the tree, effectively acting as a filter. Therefore, the Instance Name column gains the active filter indicator, a yellow background by default, when constants are hidden.</p> <p>This filter is a higher priority than the user custom filters applied to the Instance Name column. If a custom filter is created to expose only instances that contain "LUT" in the name, but Show Constants is disabled, any instances that have a "const" in their name remain hidden, even if they have an explicit "LUT" in the name.</p>
	Show Duplicates/Clones ⁽²⁾			Y	<p>Toggle the visibility of instances with various forms of "_DUP_" in the name, most often created by Achronix optimizations.</p> <p>Enabled by default. When disabled, instances with "_DUP_" or "_dup_" in the name (typically with some additional notation) are hidden within the tree, effectively acting as a filter. Therefore, the Instance Name column gains the active filter indicator (yellow background by default) when duplicates/clones are hidden.</p> <p>This filter is a higher priority than the user custom filters applied to the Instance Name column. If a custom filter is created to expose only instances that contain "LUT" in the name, but Show Duplicates/Clones is disabled, any instances that have a "_DUP_" in their name remain hidden, even if they have an explicit "LUT" in the name.</p>
	Toggle Filter Row Visibility ⁽³⁾	Y	Y		Changes whether the filter row (of filter icons) is visible or not.
	Configure view...		Y		Jumps to the Netlist Browser view in the Preferences dialog.


Warning!

- Be aware that when actions are performed upon macros, all the children of that macro, even the invisible/filtered nodes, are affected.
- With default preference settings, in the **Floorplanner View** (page 43), highlight colors of (placed) instances are only visible when the instances layer is enabled, and the instances are not members of the ACE selection set. This is because the instance selection color has a higher priority than the highlight color.

Filtering Displayed Instances

Some convenience filters for **Cell Type** are already present as visibility toggles in the Netlist browser. These filters are represented by the () **Show Boundary Pins** and () **Show Power and Grounds** toggle actions/buttons. Be aware that the "hide" functionality of these actions (when the Show toggle is disabled) is considered a higher-priority filter than any user custom filters. For example, when boundary pins are hidden due to this toggle, even if a custom filter tries to expose boundary pins, the toggled filter wins, and the pins remain hidden.

Additionally, some convenience filters for **Instance Name** are already present as visibility toggles. These filters are represented by the **Show Feedthrough Luts**, **Show Constants**, and **Show Duplicates/Clones** toggled menu items. Be aware that the "hide" functionality of these actions (when the Show toggle is disabled) is considered a higher-priority filter than any user custom filters. For example, when constant instances are hidden due to this toggle, even if a custom filter tries to expose constants, the toggled filter wins, and the constant instances remain hidden.

To enable custom instance filter manipulations, it might be necessary to click () **Toggle Filter Row Visibility** to cause the filter manipulation row to become visible. This toggle action is available in a context menu when right-clicking any table column header and is also available in the view supplemental menu (the small down arrow icon in the upper-right of the view, to the left of the **Minimize View** button).

Most columns of the table can filter the displayed instances (not the macros) by value. When filtering by column value, only instances with column values matching the filter are retained; non-matching values are excluded from the table.


Be aware that macro rows do not directly respond to filters, and remain visible as long as any single child instance remains visible. When all child instances of a macro are hidden, the parent macro is hidden as well. On a related note, macro summary counts in numeric columns (as when counting LUTs in a macro) do not change when filters are applied. The displayed counts are always the complete, unfiltered counts.


Warning!

- When using filters, the values being filtered are those of the individual instances, not the macros. Macros are filtered out only if all of their children are filtered out. As a result, when filtering by the logic types, the only possible filter numeric values in this table are 0 or 1, because these are the only legal values for an instance.
- Also, be aware that when filtering the **Instance Name** column, the parent macro names are considered part of the instance name — the prefix (the fully qualified instance name is used, not just the terminating leaf name).

Columns containing text can be filtered by string value (simple wildcard substring matching by default, but regular expression matching using Java rules is also available, see https://en.wikipedia.org/wiki/Regular_expression). Columns with checkmarks can be filtered by boolean value. Columns containing numbers can be filtered by numerical value.

To add a filter to a column:

1. Click the () filter icon, which causes a data-appropriate filter dialog to appear.
2. Fill in the desired filter values.
3. Click **Apply** to apply the filter to the instances in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the () active filter icon.

An example filter for the **Cell Types** column that uses regular expressions to block PWR, GND, ASC, SAC, and bit* cell types in the following image.

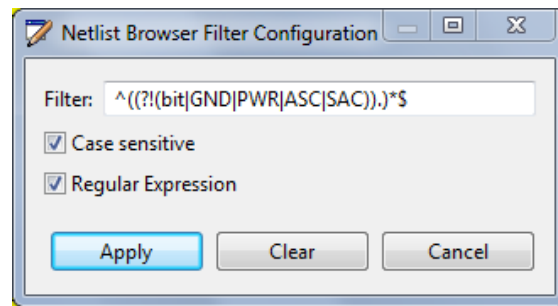




Figure 31 - Cell Types Column Filter Example

To edit (or clear) an existing filter:

1. Click the () active filter icon causing the data-appropriate filter dialog to appear, now pre-populated with the existing filter setting.
2. Change the filter value and click **Apply** again to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the () inactive version.

Drag-and-Drop

The Netlist browser supports a limited set of drag-and-drop interactions with other views in the **Floorplanner perspective** (page 6). The Netlist browser view only acts as a drag-and-drop source, not a destination; items dropped on the Netlist browser view are ignored.

Any node of the tree may be dragged to the **Tcl Console view** (page 142), and when dropped anywhere in the view, appropriate text is inserted at the beginning of the Tcl command-line.

Instance nodes may also be dragged to the **Floorplanner view** (page 43). When dropped on the Floorplanner view, the behavior depends upon the current Tool mode. When the Floorplanner **Placement/Panning Tool** is active, placement is attempted.

Any node of the tree may be dragged to the **Placement Regions view** (page 110) or the Floorplanner view (when that view has the **Placement Regions Tool** active) to **assign placement region constraints** (page 398). Dragging a macro is the equivalent of dragging all individual instances which are members of that macro.

NoC Performance View

The NoC Performance view shows an interactive diagram that includes the I/O ring and the 2D NoC resources for the target device. Loading a simulation log file produced by the device simulation model (DSM) into the view provides graphic visualization of traffic between Network Access Points (NAPs) for different periods of time ("time slices") during the simulation.

Loading Simulation Log Files

Load simulation log files by clicking **Browse** on the view control bar as shown:

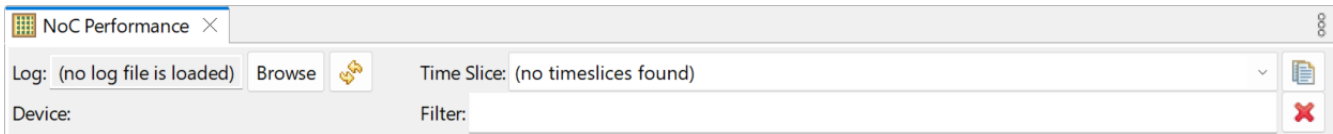


Figure 32 • Simulation Log File Loading Example

When a log file has been loaded, hover the mouse pointer over the **Log** text field to see the full path to the loaded file:



Figure 33 • Simulation Log File Path Example

Browsing Time Slices

Use the **Time Slice** combo box on the view control bar to choose a time slice to visualize. The statistics in the chosen time slice are used to colorize portions of the view diagram.

Use the **Filter** text control to enter a regular expression; the list of choices available in the **Time Slice** combo will be filtered down to only those choices that match the expression.

When the time slice list is being filtered, both the **Time Slice** combo box and the **Filter** text control will be colorized to remind you that filtering is taking place.

These colors can be adjusted in the **Colors and Fonts** preferences area. The Preferences dialog can be accessed through the main application menu (**Window → Preferences**), or by clicking the "three-dots" shortcut in the upper right of the view and then clicking **Configure view...**:

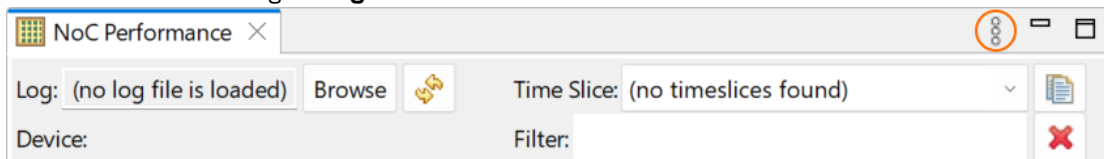


Figure 34 • Preferences Dialog Shortcut Location

There is a "tool box" of additional controls in a collapsible flyout panel to the right of the diagram, similar to the one found in the **Floorplanner view** (page 43):

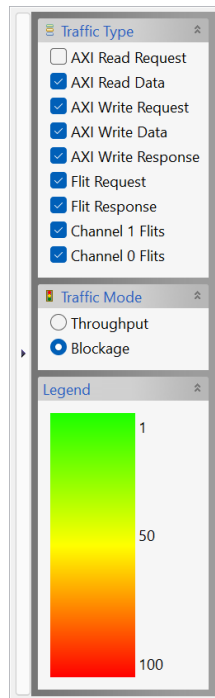


Figure 35 • Time Slice Tool Box Panel Example

The tool box contains three main sections:

- Traffic Type – toggles for choosing which types of traffic data to display in the diagram.
- Traffic Mode – two diagram display modes may be selected: **Throughput** or **Blockage**.
- Legend – displays the gradient range coloring for the selected traffic mode.

In Throughput mode, the diagram is colored using the following gradient range:

Table 51 • Throughput Mode Gradient Range

Throughput Gradient	Default Color
High	Green
Medium	Light blue
Low	Darker blue

The Throughput mode diagram coloring is illustrated in the following figure:

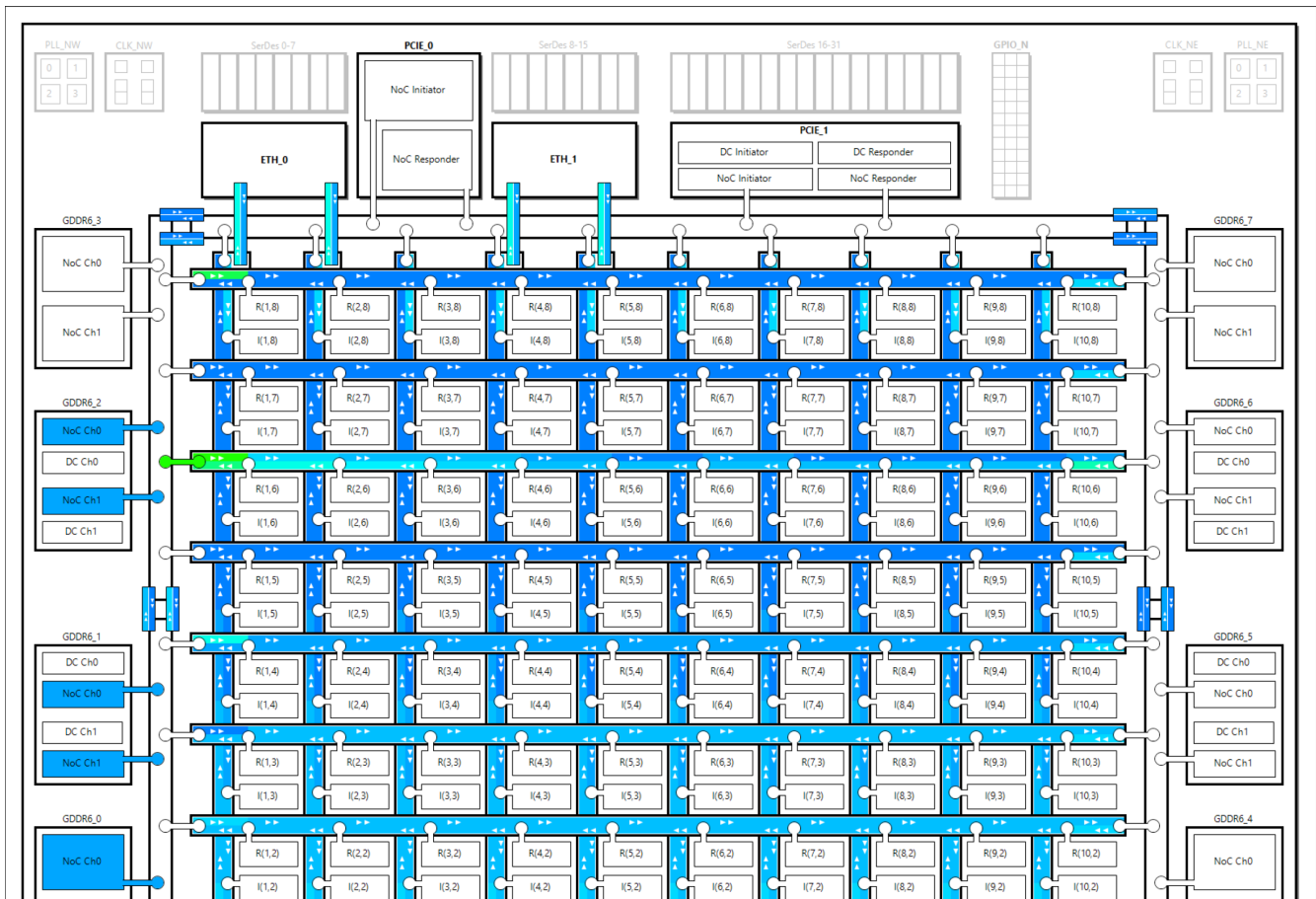


Figure 36 • Throughput Mode Diagram Coloring Example

In Blockage mode, the diagram is colored using the following gradient range:

Table 52 • Blockage Mode Gradient Range

Blockage Gradient	Default Color
Low	Green
Medium	Yellow
High	Red

The Blockage mode diagram coloring is illustrated in the following figure:

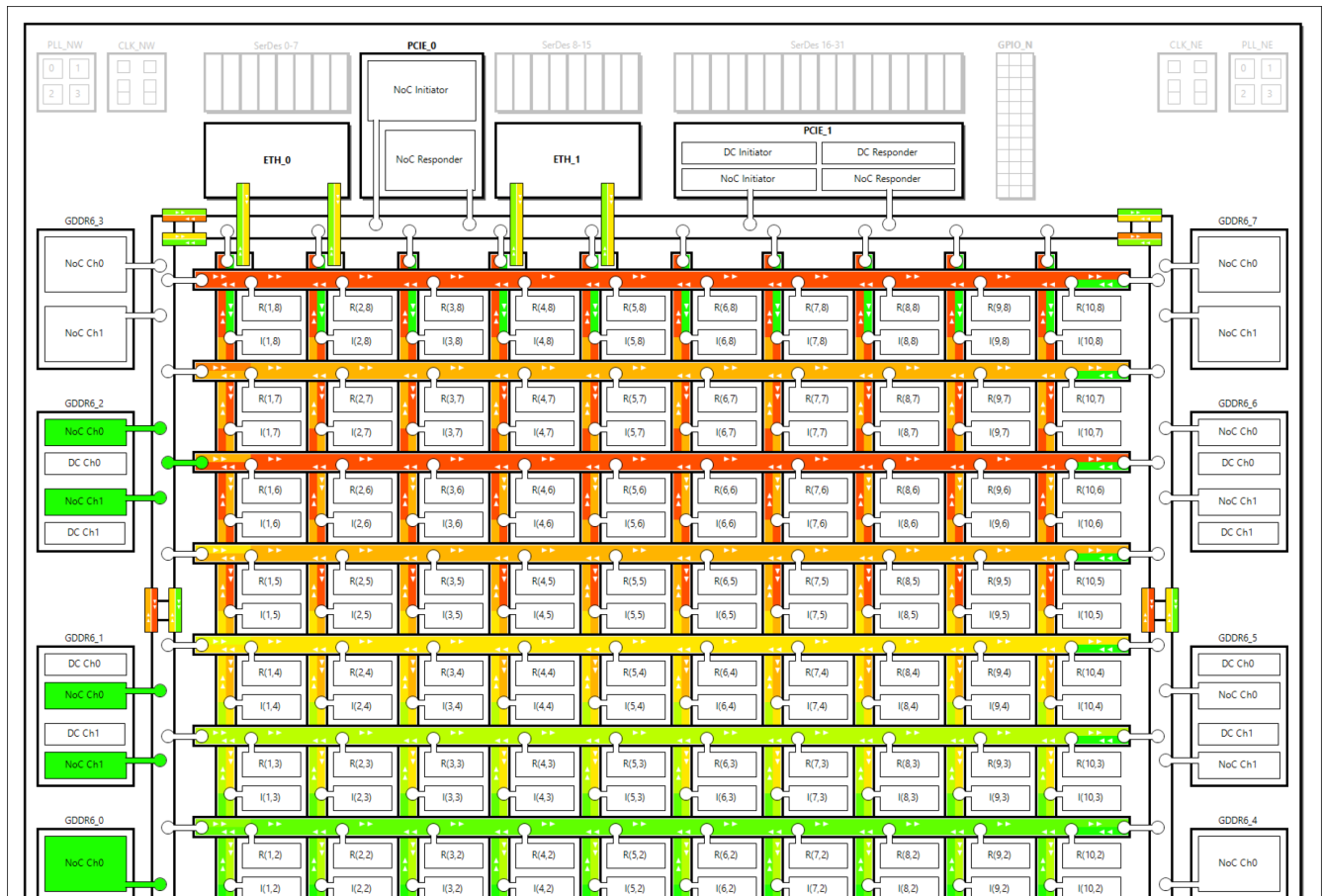


Figure 37 • Blockage Mode Diagram Coloring Example

Hover the mouse pointer over any colored portion of the diagram and a tool tip is displayed showing the raw data from the simulation log file used to determine the color.

There is also a breakdown of how much time was spent "idle" versus "trying," with the "trying" time further broken down into time spent "blocked" and time spent "transferring":

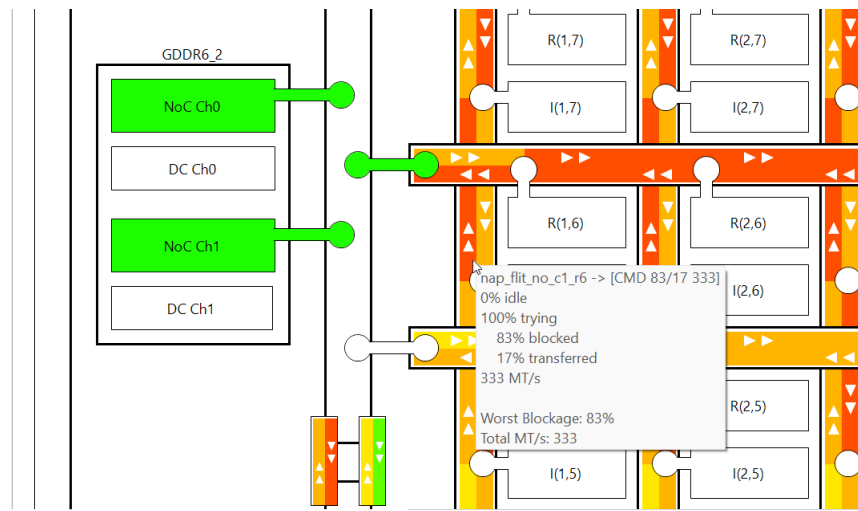


Figure 38 • Simulation Data Tool Tip Example

Zooming

The view can be zoomed in or out several levels with the mouse wheel while the mouse pointer is hovering over the diagram.

Drag-Scrolling

When zoomed in, the diagram can be scrolled in any direction by clicking and dragging the mouse pointer anywhere inside of the diagram.

Adjusting Diagram Properties

The font used in the diagram, as well as the colors used to generate the throughput and blockage gradients, can be configured in the Preferences dialog.

The Preferences dialog can be accessed through the main application menu (**Window** → **Preferences**), or by clicking the "three-dots" shortcut in the upper right of the view and then clicking **Configure view...**:

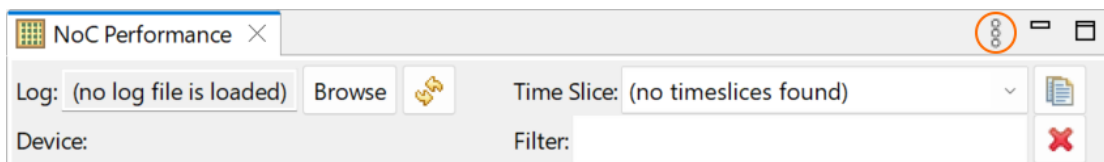


Figure 39 • Preferences Dialog Shortcut Location

The Preferences Dialog and the available settings are shown in the following figure.

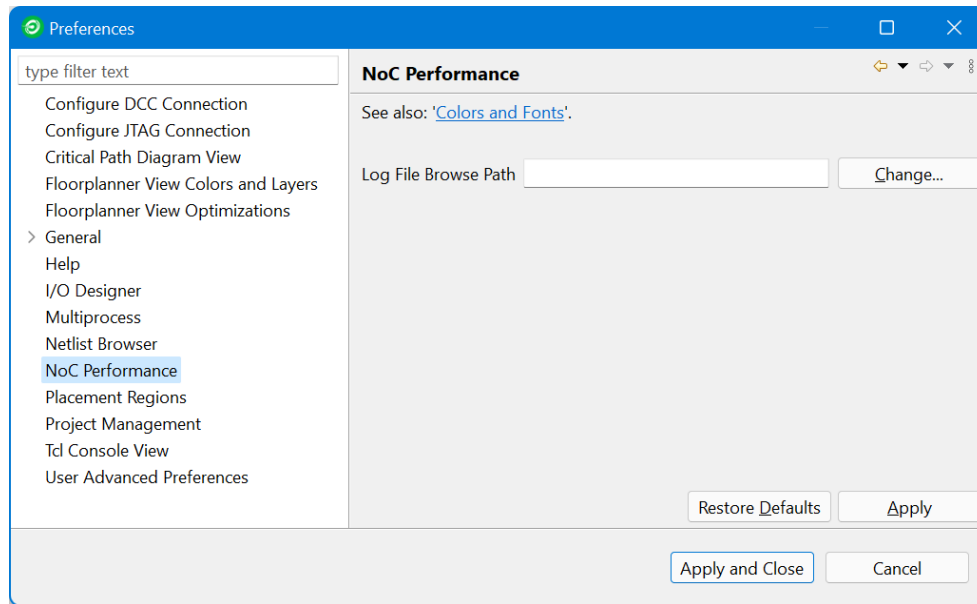


Figure 40 - NoC Performance Preferences

Table 53 - NoC Performance Preferences

Preference	Description
See Also: 'Colors and Fonts'	Clicking this hyperlink moves to the NoC Performance section of the Colors and Fonts preferences area.
Log File Browse Path	By default, the log file Browse button in the NoC Performance view begins browsing at the last path used. To always start browsing in a specific folder, enter that path here.

NoC Time Slice View

The NoC Performance time slice view displays information about the time slice currently selected in the **NoC Performance View** (page 87).

Statistics and "Notes" from the simulation log data can be rendered in different colors that can be specified in the application preferences.

Word-wrap can be toggled via a button in the upper-right of the view tool bar.

As with all views in ACE, this view can be dragged and dropped to any convenient location: left of the NoC Performance View, below it, etc.

```

NoC Time Slice
@1234 acx_perf> AXI Read Requests, Responses
@1234 acx_perf> nap_axi_slave_c1_r1 --> [AR 95/ 5 38] [ R 0/28 212]
@1234 acx_perf> nap_axi_slave_c2_r1 --> [AR 95/ 5 38] [ R 0/28 212]
@1234 acx_perf> nap_axi_slave_c3_r1 --> [AR 95/ 5 38] [ R 0/28 212]
@1234 acx_perf> nap_axi_slave_c4_r1 --> [AR 95/ 5 38] [ R 0/28 210]
@1234 acx_perf> nap_axi_slave_c5_r1 --> [AR 95/ 5 38] [ R 0/28 208]
@1234 acx_perf> nap_axi_slave_c6_r1 --> [AR 95/ 5 37] [ R 0/28 207]
@1234 acx_perf> nap_axi_slave_c7_r1 --> [AR 95/ 5 38] [ R 0/28 208]
@1234 acx_perf> nap_axi_slave_c8_r1 --> [AR 95/ 5 36] [ R 0/26 198]
@1234 acx_perf> fabric_axi_w_r1 --> [AR 87/13 251] [ R 0/84 1686]
@1234 acx_perf> AXI Write Requests/Data/Responses
@1234 acx_perf> ddr4_dc0_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_1_dc0_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_1_dc1_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_2_dc0_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_2_dc1_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_5_dc0_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_5_dc1_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_6_dc0_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> gddr6_6_dc1_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> pcie_x16_dc_slave_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> pcie_x16_dc_master_axi --> [AW 0/ 9 90] [ W 0/72 721] [ B 0/ 9 90]
@1234 acx_perf> Flit Requests/Responses
@1234 acx_perf> noc_ws_to_wn --> [Rq 10/18 10] [Rs 33/ 2 100]
@1234 acx_perf> noc_wn_to_ws --> [Rq 50/18 25] [Rs 75/ 2 80]
@1234 acx_perf> noc_wn_to_n --> [Rq 10/18 75] [Rs 33/ 2 60]
@1234 acx_perf> noc_n_to_wn --> [Rq 50/18 100] [Rs 75/ 2 40]
@1234 acx_perf> noc_en_to_n --> [Rq 10/18 75] [Rs 33/ 2 60]
@1234 acx_perf> noc_n_to_en --> [Rq 50/18 100] [Rs 75/ 2 40]
@1234 acx_perf> noc_es_to_en --> [Rq 10/18 10] [Rs 33/ 2 100]
@1234 acx_perf> noc_en_to_es --> [Rq 50/18 25] [Rs 75/ 2 80]
@1234 acx_perf> noc_es_to_s --> [Rq 10/18 75] [Rs 33/ 2 60]
@1234 acx_perf> noc_s_to_es --> [Rq 50/18 100] [Rs 75/ 2 40]
@1234 acx_perf> noc_ws_to_s --> [Rq 10/18 75] [Rs 33/ 2 60]
@1234 acx_perf> noc_s_to_ws --> [Rq 50/18 100] [Rs 75/ 2 40]
@1234 acx_perf> noc_es_to_s --> [Rq 10/18 75] [Rs 33/ 2 60]
@1234 acx_perf> noc_s_to_es --> [Rq 50/18 100] [Rs 75/ 2 40]
@1234 acx_perf> Channel 0/1 Flits
@1234 acx_perf> ethernet_flit_s_c1 --> [C0 0/37 745]
@1234 acx_perf> ethernet_flit_n_c2 --> [C0 0/37 745]
@1234 acx_perf> ethernet_flit_s_c2 --> [C0 0/37 745]
@1234 acx_perf> ethernet_flit_n_c4 --> [C0 0/37 745]
@1234 acx_perf> ethernet_flit_s_c4 --> [C0 0/37 745]
@1234 acx_perf> ethernet_flit_s_c1 --> [C1 10/100 1000]
@1234 acx_perf> ethernet_flit_n_c2 --> [C1 20/100 1000]
@1234 acx_perf> ethernet_flit_s_c2 --> [C1 30/100 1000]
@1234 acx_perf> ethernet_flit_n_c4 --> [C1 40/100 1000]
@1234 acx_perf> ethernet_flit_s_c4 --> [C1 50/100 1000]

```

Figure 41 • NoC Performance Time Slice View Example

Adjusting View Properties

The font used in the Time Slice View, as well as the colors used to differentiate statistics from notes, can be configured in the Preferences dialog.

The Preferences dialog can be accessed through the main application menu, **Window** → **Preferences**. The dialog can also be opened by double-clicking the gradient **Legend** section in the NoC Performance View toolbox.

The Preferences Dialog and the available settings are shown in the following example:

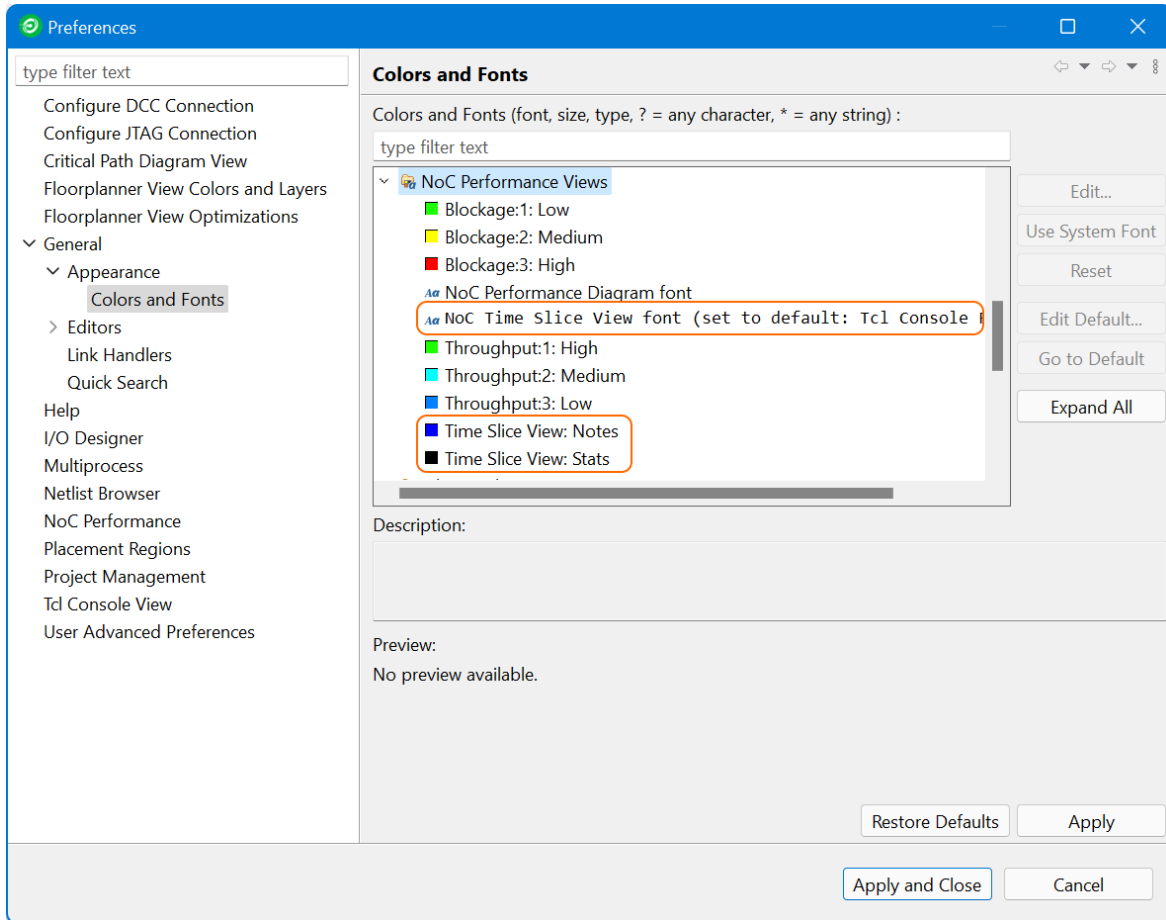


Figure 42 • NoC Time Slice View Colors and Fonts Preferences

Table 54 • NoC Performance Preferences

Preference	Description
NoC Time Slice View font	The font to be used in the NoC Time Slice view.
Time Slice View: Notes	The color used to render "notes" in the NoC Time Slice view.
Time Slice View: Stats	Color used to render "statistics" in the NoC Time Slice view.

Options View

The Options view displays project implementation **options** (page 229) for the active **implementation** (page 229). From this view, the **active project implementation** (page 229) can be configured for its run through the **flow** (page 234).

This view does not display any information unless an active implementation is selected in the **Projects view** (page 117). When **running the flow** (page 306), the implementation options of the active implementation are used to govern the flow.

By default, the Options view is included in the **Projects perspective** (page 6). To add the Options view to the current perspective, select **Window** → **Show View...** → **Options**.

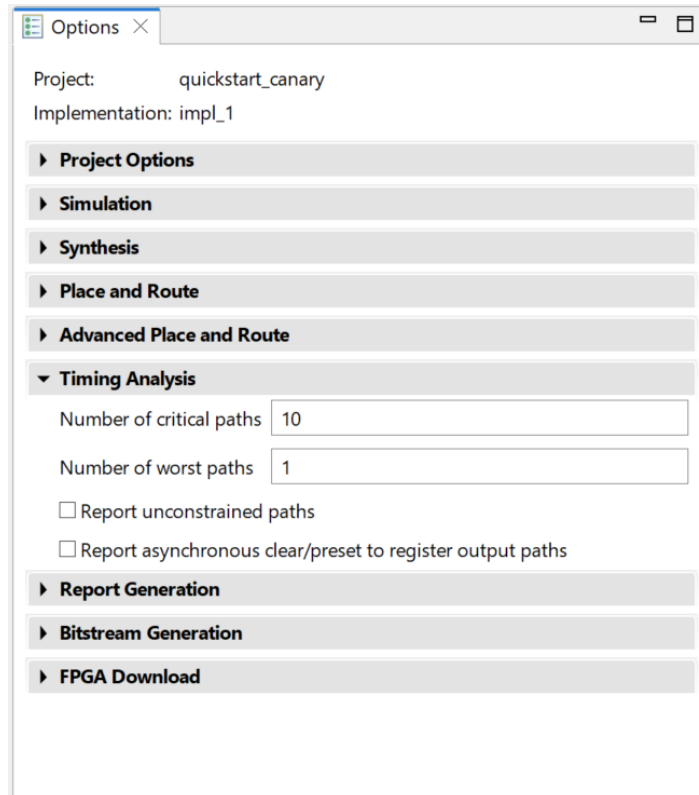


Figure 43 · Options View Example

 **Tip**
Tcl Equivalence

Each implementation option that can be configured via this graphical view may also be configured via the [set_impl_option \(page 711\)](#) Tcl command. The current value of each option can be retrieved with the [get_impl_option \(page 648\)](#) Tcl command. The values of options may be reset back to their default values with the [reset_impl_option \(page 686\)](#) Tcl command.

Likewise, each project-level option that can be configured via this graphical view may also be configured via the [set_project_option \(page 714\)](#) Tcl command. The current value of each option can be retrieved with the [get_project_option \(page 655\)](#) Tcl command. The values of options may be reset back to their default values with the [reset_project_option \(page 687\)](#) Tcl command.

 **Note**

The Options view does not show all available options.

- The options included in this view while ACE is running are the most used standard supported options, but are only a subset of all options available.
- The options shown in the tables below are the subset of non-advanced options in the view that are relevant to all libraries/devices. Library-specific or device-specific options are not listed within these tables.
- Power users of ACE may also configure the Options view to display all "advanced" options by setting the GUI preference under the main menu: **Window** → **Preferences** → **User Advanced Preferences** → **Display Advanced Impl Options**.
- A complete list (with descriptions) of all available library-specific, device-specific, and advanced options, along with default values and current values, is available in the [Implementation Options Report \(page 257\)](#), which can be generated with the [report_impl_options \(page 679\)](#) Tcl command. Be aware that the report content varies with the chosen active device/library and as new ACE releases alter the available options/values.

Table 55 • Project Options

Option	Tcl Option	Description
Target Device ⁽¹⁾	partname	Specifies the name of the FPGA part for this implementation.
Package	package	Specifies the FPGA package for the target device.
Speed Grade	speed_grade	Selects the desired speed grade for the target device.
Core Voltage	core_voltage	Selects the core voltage for the target device.
Junction Temperature	junction_temperature	Selects the junction temperature for the target device.

Option	Tcl Option	Description
Flow Mode ⁽²⁾	flow_mode	<p>Evaluation mode – ignores non-fatal DRCs as long as possible, allows I/O Virtualization, and ignores missing SDC constraints to get a post-route timing report quickly.</p> <p>Normal mode – enforces all DRC checks necessary to generate a correct bitstream. Some checks are flagged as warnings early on in the flow to provide an opportunity to fix the problems (for example, fixing the placement of I/Os). These same checks may change to report an error during final DRC checks.</p> <p>Strict mode – similar to Normal flow mode, but enforces all DRC checks and errors out as early in the flow as possible.</p> <p>See also: Flow Mode (page 242)</p>
Enable Incremental Compile	incremental_compile	Enables the ACE Incremental Compilation functionality. The upstream synthesis tool must also have incremental compilation enabled (Synplify Pro compile points), and the necessary constraint files must be included in the ACE project, otherwise ACE is unable to make use of this feature as intended. See Using Incremental Compilation (Partitions) (page 404) .
Incremental Compile Mode	incremental_compile_mode	Incremental Compile can either be in Strict or Smart mode. Strict mode – ensures that Placement of locked instances in unchanged partitions is completely preserved. Smart mode – allows ACE to attempt intelligently preserving placement in locked partitions for better design performance.
Export All Partitions	export_all_partitions	When enabled, all leaf-level partitions are exported.
Auto-Select Top Module	autoselect_top_module	Specifies whether the top module name for this implementation should be automatically selected. A value of 1 automatically selects the name. A value of 0 forces use of the <code>-top_module</code> implementation option value.
Top Module Name	top_module	Specifies the top module name for this implementation when the <code>-autoselect_top_module</code> implementation option is set to 0 .
Enable Final Timing Checks	check_final_timing	Causes the flow to stop and error out prior to Bitstream Generation if final sign off timing is not met across all temperature corners.
HDL Include Path	hdl_include_path	Specifies HDL include paths for Synthesis and Simulation, in semicolon separated string format.
HDL Defines	hdl_defines	Specifies HDL defines for Synthesis and Simulation, in space-separated string format.
Use Default Project Output Path	use_default_project_output_path	When enabled, uses the default output directory file path, which is the same as the ACE project file (ACXPRJ) source directory.
Project Output Path	project_output_path	Provides the output directory file path for output generation. Relative paths are relative to the ACXPRJ file.

Option	Tcl Option	Description
<p>Table Notes</p> <ol style="list-style-type: none"> 1. Caution: The chosen "Target Device" affects other options. Each target device can have unique options available within ACE, and may even have different default values for those options which are shared or common between devices. Changing the target device value may have a ripple effect upon other option values. Thus, reviewing the values of all other options after changing the target device value may be necessary. 2. Bitstream generation requires Normal or Strict flow mode. 		

Table 56 • Simulation Options

Option	Tcl Option	Description
Simulation Flow	<code>sim_flow</code>	Select Default to use the built-in simulation flow scripts. Select Custom to call your own custom simulation Tcl proc, which is defined in your ACE_INIT_SCRIPT setup.
Simulation Tool	<code>sim_tool</code>	Determines the simulation tool to use in the simulation flow steps.
Custom Simulation Command	<code>sim_custom_command</code>	Custom simulation Tcl command to run simulation. This Tcl procedure should be defined in your ACE_INIT_SCRIPT setup. ACE automatically calls this Tcl procedure and adds a <code>-sim_step <step></code> option to the command, which is set to one of <code>rtl</code> , <code>gate</code> , <code>routed</code> , or <code>final</code> .
Testbench Top Module	<code>sim_tb_top_name</code>	Specifies the name of the testbench top module to simulate.
Compile Simulation	<code>sim_<TOOL>_compile</code>	Runs the appropriate <TOOL> commands to compile your Verilog and/or VHDL.
Run Simulation	<code>sim_<TOOL>_run</code>	Run the appropriate <TOOL> commands to run the simulation.

Table 57 • Synthesis Options

Option	Tcl Option	Description
Generate Project File	<code>syn_use_default_project</code>	When enabled, the source Synthesis project file is automatically generated from the ACE project settings and managed by ACE in the <code>Project/Output/<impl>/syn</code> directory. When this option is disabled, the full file path to the synthesis project file must be specified, which then gets copied to the <code>Project/Output/<impl>/syn</code> directory in the <code>run_synthesis</code> flow step.
Project Override Path	<code>syn_project_override_path</code>	Synthesis project file override path. Use this option to specify the full file path to an existing synthesis project file to prevent ACE from automatically loading all ACXPRJ source files.

Option	Tcl Option	Description
Route Delay Model	syn_route_delay_model	The synthesis route delay model represents the instance and fanout-based net delays synthesis uses to approximate the post-place-and-route delays. Different route delay models map better to different designs. Sweeping over these options using the Multi-Process ACE feature helps to find the optimal setting for your specific design.
Fanout Limit	syn_fanout_limit	Specifies the maximum fanout any net can have when generating the gate level netlist. Nets that exceed the limit are cloned to reduce the fanout.
Enable Retiming	syn_retiming	Enables the running of re-timing during synthesis.
Default Frequency (MHz)	syn_default_frequency	Specifies the assumed default clock frequency in MHz for unconstrained clock domains.
Advanced Synplify Options	syn_advanced_options	Specifies a string of key value pairs for synthesis override options in Tcl list format (i.e., {{option1 val1} {option2 val2}}).

Table 58 • Place and Route Options

Option	Tcl Option	Description
PnR Mode	timing_driven_pnr	Timing Driven – data from timing analysis is used to optimize the design for high speed. Fast – placement and routing are optimized for runtime.
Multi-Threaded PnR	mt_pnr	Enables multi-threaded place and route. Enabling this can decrease compile time.
PnR Seed	seed	Initializes the random number state in the place and route algorithms.
Placement Effort	placement_effort	Low – placement has a shorter runtime, but may yield less design QoR than High effort placement. High – increases placement runtime to further optimize the design QoR if possible.
Router Hold-Violation Fix Limit (ps)	router_max_hold	Specifies the maximum hold-time violation (in picoseconds) that the router attempts to fix.
Post-PnR Buffer Limit	max_postpnr_buffer_limit	Specifies the maximum number of post-placement buffers that can be inserted.
Route Reset as Clock	route_rst_as_clock	Specifies the number of reset nets that should be routed on the clock network.
Physical Synthesis during Placement/ Routing	physical_synthesis	Specifies the use of physical synthesis during placement and routing.
Place and Route driven Rewiring	postpnr_rewire	If enabled, allows rewiring during place and route to improve performance and resource usage.
Rewiring Wide Gate Inputs	rewire_wide_gates	Enables wide gate input re-assignment.

Option	Tcl Option	Description
Placement-Driven Netlist Optimizations	post_place_netlist_optimization	Enables placement-driven netlist optimizations.
Clock Skew optimization	clock_skew_opt	Specifies whether to enable clock skew optimization to increase design performance. 0 – Disabled. 4 – After clock routing. 8 – After final routing. 12 – Both.
Place and Route Constraint Files ⁽¹⁾	enable_project_source_file, disable_project_source_file	Allows enabling or disabling an existing project place and route SDC/PDC constraint file for use in this implementation flow. All constraint files defined for the active project are listed.
Netlist Files	enable_project_source_file, disable_project_source_file	Allows enabling or disabling an existing project place and route netlist file for use in this implementation flow. All Netlist files defined for the active project are listed.

Table Notes

1. Constraint files are loaded in the order listed. To change the constraint file load order, see [Adding Source Files](#) (page 297).

Table 59 • Advanced Place and Route Options

Option	Tcl Option	Description
Timing-Driven Clustering	timing_driven_clustering	Specifies whether timing-driven clustering is enabled during placement.
Fanout Control	fanout_control	When enabled, nets with a fanout higher than the Fanout Limit are refactored.
Fanout Limit	fanout_limit	Specifies the maximum fanout any net can have when Fanout Control is enabled.
Fanout Limit for Critical Nets	critical_fanout_limit	Specifies the maximum fanout critical nets can have when Fanout Control is enabled.
Isolate Bigblk loads	isolate_bigblk_loads	When enabled, allows replication of LUTs driving NAPs, BRAMs, and FIFOs. 1 – enables. 0 – disables.

Option	Tcl Option	Description
Resynthesis Mode	synthesis_remap	<p>Specifies whether resynthesis should optimize for timing, area, or be disabled. Off – disables all resynthesis. Optimize for Area – can be used to reduce the total number of LUTs. Optimize for Timing – can be used to improve timing, but may increase area.</p> <p>The optimizations performed depend upon the following strategies chosen. If the "Place and Route" implementation option, Timing-Driven PnR, is disabled, resynthesis timing optimizations are also disabled.</p>
Rewrite Rule-1	resynthesis_rewrite_rule1	<p>When enabled, and Resynthesis Mode is set to Optimize for Timing, attempts to reduce the number of LUTs in series. In critical paths, rewrite looks at the LUT programs and the number of used inputs to determine where to apply the transformation. The following transformation is then applied when feasible:</p> <div data-bbox="987 800 1354 1297" style="text-align: center;"> <p>Rewrite Rule 1</p> <p style="font-size: small;">557362-01.2016.10.12</p> </div>
Move Flip-flop Reset	resynthesis_move_ff_reset	<p>Specifies whether resynthesis moves flip-flop reset logic to LUTs when Resynthesis Mode is set to Optimize for Timing.</p>
Period of Anti-Aging Oscillator (in ns) ⁽¹⁾	areafill_clock_period	<p>Period of areafill oscillator in ns (0 to disable). For anti-aging purposes, setting this option to a non-zero value causes ACE to automatically insert logic during Run Prepare to fill the area in the core fabric not consumed by the user design logic, driven by a ring oscillator which toggles at the specified period in nanoseconds. When 0, disables insertion of anti-aging area fill logic.</p>
Limit Anti-Aging to Clocks Paths	anti_aging_onlyclock	<p>Use anti-aging areafill only on clock nodes. Data paths are not filled.</p>

Option	Tcl Option	Description
Virtual IO Style	virtual_io_style	<p>Reduces the number of I/O pads by collapsing bussed ports. Only applies in Evaluation flow mode when I/O pad utilization exceeds the value of Virtual IO Utilization.</p> <p>Off – disables this feature.</p> <p>Stubout using Floating LUTs – converts the pads into unconnected LUTs.</p> <p>Serialize using LUTs – reduces the bus into a single pad feeding a scan chain made of LUTs, with a second pad used to select between "load" and "shift" modes.</p> <p>Serialize using DFFs – builds the scan chain from DFFs, allowing the boundary connections to be timed.</p> <p>Port buses to be virtualized can be specified manually in the RTL or PDC file with the port attribute, <code>ace_virtualize</code>, or automatically by ACE. See Working with Virtual I/O (page 466) for details.</p>
Virtual IO Utilization	virtual_io_utilization	<p>The I/O pad utilization percentage targeted by I/O virtualization. Must be an integer between 0 and 100. An error is returned if the given utilization cannot be met.</p> <p>0 – requests that all possible port buses and non-bused ports are virtualized to achieve the smallest possible number of pads.</p> <p>100 – requests that port buses and non-bussed ports are to be virtualized until the number of remaining ports fit in the target device.</p>
Push Flops Into Pads	push_flops_into_pads	<p>Control over whether the first level of flip-flops is to be automatically pushed into the I/O pins.</p> <p>Disabled – turns off pushing of flip-flops into the I/O pins.</p> <p>Automatic – enables full automatic pushing of all possible flip-flops into the I/O pins except for pins with the attribute "syn_useioff=0".</p> <p>Manual – push flip-flops only into the I/O pins which have the attribute "syn_useioff=1".</p> <p>See Automatic Flop Pushing into I/O Pins (page 457) for details.</p>
Pad Flop Pushing Clock Type	pad_flop_pushing_clock_type	<p>Controls flop pushing into I/O pins by clock type.</p> <p>Boundary – Only enable flop pushing into I/O pins clocked by a boundary clock.</p> <p>Trunk – Only enable flop pushing into I/O pins clocked by a trunk clock.</p> <p>All – Enable flop pushing into all I/O pins.</p> <p>See Automatic Flop Pushing into I/O Pins (page 457) for details.</p>

Table Notes

1. Setting the Period of Anti-Aging Oscillator option to a **non-zero** value increases the size of the user design in place and route and the corresponding bitstream to near maximum size.

Table 60 • Timing Analysis Options

Option	Tcl Option	Description
Number of critical paths	sync_timing_num_paths	Maximum number of critical paths per clock group.
Number of worst paths	sync_timing_num_worst	Maximum number of worst paths per end point.
Report unconstrained paths	report_unconstrained_timing_paths	When enabled, ACE includes unconstrained timing paths in the timing analysis reports.
Report asynchronous clear/preset to register output paths	report_clear_preset_timing_paths	When enabled, ACE includes asynchronous clear/preset to register output timing paths in the timing analysis reports.

Table 61 • Report Generation Options

Option	Tcl Option	Description
Output Obfuscated Timing Reports	report_obfuscate	Generates obfuscated timing reports in addition to standard reports.
Output Partition Reports	report_partitions	Enables partition report generation in the flow.
Output Pin Assignment Reports	report_pins	Enables pin assignment report generation in the flow.
Output Clock Reports	report_clocks	Enables clock analysis and report generation in the flow.
Output Placement Reports	report_placement	Enables placement report generation in the flow.
Output Routing Reports	report_routing	Enables routing report generation in the flow.
Output Power Reports	report_power	Enables power analysis and report generation in the flow.

Table 62 • Bitstream Generation Options

Option	Tcl Option	Description
Serial Flash (.flash)	bitstream_output_flash	Enables the generation of an additional serial flash formatted output file, named the same as the STAPL file, but with a .flash extension. Contains a binary image that can be directly loaded into a single serial flash memory.

Option	Tcl Option	Description
CPU Mode (.cpu)	bitstream_output_cpu	Enables the generation of an additional CPU Mode formatted output file, named the same as the STAPL file, but with a <code>.cpu</code> extension. Contains hexadecimal-formatted data organized in "CPU Bus Width" number of bits per file line. Data from this file is sent to the FCU CPU interface line by line (one line per clock cycle) from the top to the bottom of the file, where the left-most bit on each line is the MSB and the right-most bit is the LSB. In simulation, this file may be loaded using the <code>readmemh</code> function. For convenience, an additional binary representation of the CPU Mode output file is written, named the same as the STAPL file, but with a <code>_cpu.bin</code> extension. The file contains the same data in the same bit order as the <code>.cpu</code> file.
CPU Bus Width	bitstream_output_cpu_width	Controls the bit width of the CPU-mode formatted output file. When using the CPU interface in $\times 8$ mode, set this value to 8 . If using the CPU interface in $\times 128$ mode, set this to 128 . Determines how many bitstream bits are printed per line in the <code>.cpu</code> output file. The bit sequence required by the FCU (and output in the generated bitstream file) may be different for each CPU Bus Width setting. Therefore, it is important to set this option to match the actual CPU hardware interface width.

Table 63 - FPGA Download Options

Option	Tcl Option	Description
JTAG Device ID (jtag_id) ⁽¹⁾	download_jtag_id	Specifies the JTAG programming device name to which a connection is attempted during FPGA download. If not populated, auto-detection of JTAG programming devices is attempted, and the download fails if more than one JTAG device is auto-detected. The device naming schemes are described in the <i>Speedster7t Configuration User Guide</i> (UG094).
Single Device Scan Chain	download_single_device	Should be enabled when the target device is the only device on a single-device JTAG scan chain. If set to 0 , the pre-IR, post-IR, and chain offset options are used to configure the scan chain.
IR Bits Before Target FPGA Device	download_preir_padding	Specifies the total number of instruction register bits on the JTAG scan chain prior to the target device instruction register. Used for multi-device scan chains to pad the IR chain properly with ones placing other devices in bypass mode.
IR Bits After Target FPGA Device	download_postir_padding	Specifies the total number of instruction register bits on the JTAG scan chain after the target device instruction register. Used for multi-device scan chains to pad the IR chain properly with ones placing other devices in bypass mode.
Target FPGA Device Offset in Scan Chain	download_chain_offset	Specifies the offset of the target device on the JTAG scan chain for multi-device chains. 0 – selects the first device on the chain. 1 – selects the second device on the chain, etc.

Option	Tcl Option	Description
<p>Table Notes</p> <ol style="list-style-type: none"> This option is used only by the FPGA Download flow step, and does not affect Bitporter pod selection for the Download view (page 40) or Snapshot Debugger view (page 137). The pod selection for those views is a user preference, which is managed by the Configure JTAG Connection preference page (page 190), and is not a per-project setting. See Configuring the JTAG Connection (page 360) for more details. 		

Outline View

The Outline view displays a tree of all pages in the currently active IP Configuration [Editor \(page 9\)](#). Each page has its own title, and an icon to indicate the cumulative validity of all IP configuration contained on that page.

The information in the tree is dynamic, and changes as corresponding values are changed in the active IP Configuration Editor. As pages in the Editor are added or removed, entries in the tree are added or removed accordingly. As values in the Editor page change validity, the validity of the corresponding page in the Outline view tree also change.

In addition to showing the page validity, the Outline view provides an alternate method for navigating between the various dynamic pages of the [IP Configuration Editors \(page 9\)](#). Selecting (clicking) an item in the Outline view causes the IP Configuration Editor to turn to the associated page.

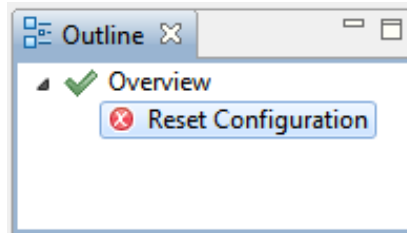


Figure 44 • Outline View Example

Table 64 • Outline View Icons

Icon	Description
✓	No errors or warnings on the page.
⚠	At least one warning on the page, but no errors.
✗	At least one error on the page.

See also: [Creating an IP Configuration \(page 336\)](#)

Partitions View

Note

The Partitions view is only relevant when Incremental Compilation is enabled and partitions (compile points) have been defined in the project constraints files. Otherwise, this view table is empty.

The Partitions view displays the state of the active implementation partitions, and allows (through interactions with the [Floorplanner view \(page 43\)](#), [Search view \(page 129\)](#), and [Selection view \(page 133\)](#)) visualizations of the partitions and their relationships with the rest of the active implementation.

The Partitions view is a default member of the Floorplanner perspective, and can be added to any other perspective by selecting **Window** → **Show View** → **Other...** → **Partitions View**.

See also: [Using Incremental Compilation \(Partitions\) \(page 404\)](#)

Partition Name	Hi..	Timestamp	Clock Pre-Routes	Anchor Instance	Re-compiled	Force Re-compile	Imported	LUTs	Flops	ALUs	LRAMs	BRAMs	IPIN
/partial_reconfig_partition_top		Tue Oct 11 17:53:07 CDT 2022			✓			5	39	4	0	0	2
/partial_reconfig_partition_top/pr_col_1_pr_row_1_i_partial_reconfig_core		Tue Oct 11 17:53:07 CDT 2022 (i_clk_pr_1_ipin_net:2) (i_reset_n_pr_1_ipin_net:1)		pr_col_1_pr_row_1_i_partial_reconfig_core/req_row_cs_block_i_axi_master_i_axi_master		✓	✓	521	357	19	0	0	0
/partial_reconfig_partition_top/pr_col_1_pr_row_2_i_partial_reconfig_core		Tue Oct 11 17:53:07 CDT 2022 (i_clk_pr_1_ipin_net:2) (i_reset_n_pr_1_ipin_net:1)		pr_col_1_pr_row_2_i_partial_reconfig_core/req_row_cs_block_i_axi_master_i_axi_master		✓	✓	521	357	19	0	0	0
/partial_reconfig_partition_top/pr_col_1_pr_row_3_i_partial_reconfig_core		Tue Oct 11 17:53:07 CDT 2022 (i_clk_pr_1_ipin_net:2) (i_reset_n_pr_1_ipin_net:1)		pr_col_1_pr_row_3_i_partial_reconfig_core/req_row_cs_block_i_axi_master_i_axi_master		✓	✓	521	357	19	0	0	0
/partial_reconfig_partition_top/pr_col_1_pr_row_4_i_partial_reconfig_core		Tue Oct 11 17:53:07 CDT 2022 (i_clk_pr_1_ipin_net:2) (i_reset_n_pr_1_ipin_net:1)		pr_col_1_pr_row_4_i_partial_reconfig_core/req_row_cs_block_i_axi_master_i_axi_master		✓	✓	521	357	19	0	0	0
/partial_reconfig_partition_top/pr_col_1_pr_row_5_i_partial_reconfig_core		Tue Oct 11 17:53:07 CDT 2022 (i_clk_pr_1_ipin_net:2) (i_reset_n_pr_1_ipin_net:1)		pr_col_1_pr_row_5_i_partial_reconfig_core/req_row_cs_block_i_axi_master_i_axi_master		✓	✓	521	357	19	0	0	0

Figure 45 - Partitions View Example

Caution!

Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic and change to match the target device after running the Prepare flow step. The screenshots and example descriptions in this section do not reflect the resource types of the actual device being used.








Table 65 - Partitions View Columns

Column	Description
Partition Name	The name of the partition as specified in the design constraints file(s).
Highlight Color	If all instances within the partition have the same highlight color, the row shows a color square with that same highlight color. If even one contained instance has a differing highlight color, or no highlight at all, then the row displays no color square.
Time Stamp	The timestamp of the last compile for this partition (compile point) in the upstream synthesis tool.
Clock Pre-Routes	If any clock pre-routes exist for a given partition, they are listed here.
Anchor Instance	The instance in the partition to "anchor" drag-and-drop operations.

Column	Description
Re-compiled	Contains a checkmark if the partition was recompiled, requiring placement and routing to be re-run.
Force Re-compile on Next Run	Indicates whether the partition is forced to re-compile during the next pass through ACE. This checkbox may be toggled on or off using the right-click Context Menu choices Force Partition Changed and Un-Force Partition Changed .
Resource	The sum count of all <i>resource</i> instances contained in this partition and no other partitions.
Cumulative Resource	The sum count of all contained <i>resource</i> instances, including in child partitions (below this partition in the RTL hierarchy).

A number of actions are available within the view, available in the toolbar at the top of the view and/or in right-click context menus for each partition in the table.

Table 66 • Partitions View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Search for Instances	Y	Y	Issues a <code>find Tcl</code> command that returns the names of all the instances in the selected partition(s).
	Add Instances to Selection	Y	Y	Adds the instances within the partition to the ACE Selection Set (as shown in the Selection view (page 133)).
	Highlight	Y	Y	Applies the currently active Highlight color to the instances within the chosen partition (see Highlighting Objects in the Floorplanner View (page 343)).
	Un-Highlight	Y	Y	Clears the Highlight for the instances within the chosen partition.
	Choose Highlight Color	Y	Y	Determines which color is applied to the objects chosen the next time the Highlight action is selected for this view.
	Auto-Highlight	Y		Automatically assigns a unique highlight color to each partition.
	Zoom To	Y	Y	Zooms the Floorplanner view to a region containing the instances within the partition currently chosen in the tree.



Icon	Action	Toolbar Button	Context Menu	Description
	Toggle Filter Row Visibility ⁽¹⁾	Y		Changes whether the filter row (of filter icons) is visible.
	Configure Clock Pre-Routes		Y	Allows adding clock pre-route information for the selected partition(s).
	Force Partition Changed		Y	Override the partition timestamp during the next pass through Ace: A check mark appears in the Force Re-compile on Next Run column, and the partition is replaced and re-routed the next time the flow is run, even if there were no RTL changes and it was not recompiled in the upstream synthesis tool. Un-Force Partition Changed removes the check mark in the column.
	Un-Force Partition Changed		Y	Removes the check mark from the Force Re-compile on Next Run column, so the compilation timestamp is no longer overridden. This is essentially an undo operation for the Force Partition Changed action.
	Export Partition		Y	Exports the selected partition to an .epdb file under {impl_name}/output/partitions, and exports a blackbox netlist for the partition to a .v file under {impl_name}/output/blackboxes.

Table Notes

1. Toggle Filter Row Visibility does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Note

All actions upon a partition act only upon the members of that partition, not upon the members of any child partitions.

Drag-and-Drop

The Partitions view supports a limited set of Drag-and-Drop interactions with other views in the **Floorplanner perspective** (page 6). The view only acts as a Drag-and-Drop source; items dropped on the Partitions view are ignored.

Any row of the table may be dragged to the **Tcl Console view** (page 142), and when dropped anywhere in the view the partition name (with the appropriate object type prefix) is inserted at the beginning of the Tcl command-line.

Any partition in the table may be dragged to the **Placement Regions view** (page 110) or the **Floorplanner View** (page 43) (when that view has the **Placement Regions Tool** active) to **assign placement region constraints** (page 398). Dropping in the Placement Regions view uses the

`add_region_find_insts` (page 613)

Tcl command. Dropping onto the Floorplanner view uses the

`set_placement -partition` (page 713)

Tcl command. Dragging a partition is the equivalent of dragging all individual instances which are members of that partition.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all selected partition rows in the view. Selecting more or fewer rows in the view updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Placement Regions View

The Placement Regions view provides a tabular representation of the content of all user-created **Placement Regions** (page 394) for the design. The view allows the manipulation of the visibility of the Placement Region itself as painted (as a colored overlay) in the **Floorplanner view** (page 43), and the content (the instances constrained to the region) of each Placement Region. The view table remains empty until the currently active implementation has completed the **Run Prepare** flow step.

Because Placement Regions are manually defined, and Instances are manually constrained to the Placement Regions, it is necessary to track the total number of sites and associated instances for each Resource type. Accordingly, based upon the chosen target device Implementation Option, there are columns for each available resource type found within the device. If more instances are ever assigned to a Placement Region than there are available sites of that type within that Placement Region, the view displays an error for that placement region and resource type combination.

The **Placement Regions and Placement Region Constraints** (page 394) section describes the creation and usage of Placement Regions in greater detail.

By default, the Placement Regions view is included in the **Floorplanner perspective** (page 6). To add the view to another perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Placement Regions**.

In the following example, error icons are shown for the CLK_IPIN assignments of `region_2`, indicating that too many instances (1) are assigned for the available sites (0) within the region. The region itself also shows an error icon to indicate when an erroneous resource type is scrolled offscreen.

Visibility	Name	Ove_	Clock Pre-Routes	Soft	Keep out	Include Routing	PR Zone	LUTs	Flops	ALUs	LRAMs
<input checked="" type="checkbox"/>	PR_ZONE_pr_col_1_pr_row_1_i_partial_reconfig_core					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	521/8,640	357/17,280	19/2,160	0/32
<input checked="" type="checkbox"/>	PR_ZONE_pr_col_1_pr_row_2_i_partial_reconfig_core		(i_clk_pr_1_ipin_net: 2) (i_clk_ipin_net: 1,3) (i_reset_n_pr_1_ipin_net: 1)			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	521/8,640	357/17,280	19/2,160	0/32
<input type="checkbox"/>	PR_ZONE_pr_col_1_pr_row_3_i_partial_reconfig_core					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	521/8,640	357/17,280	19/2,160	0/32
<input type="checkbox"/>	PR_ZONE_pr_col_1_pr_row_4_i_partial_reconfig_core					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	521/8,640	357/17,280	19/2,160	0/32
<input checked="" type="checkbox"/>	PR_ZONE_pr_col_1_pr_row_5_i_partial_reconfig_core					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	521/8,640	357/17,280	19/2,160	0/32

Partial reconfig cluster value: 000000001000000401

Figure 46 • Placement Regions View Example

i Note

Resource type columns, such as Flops, BRAMs, ALUs, etc. are dynamic. When the Run Prepare flow step has completed, the columns appropriate to the target device are chosen and values are updated. The resource type columns shown in the screenshots should be considered examples only — they might not match the exact resources of any particular target device.

Table 67 - Placement Regions Table Columns

Column	Description
Visibility	When selected (this is user-editable), this placement region overlay is painted in the Floorplanner view (page 43), using the chosen translucent Overlay Color .
Name	The name of this placement region.
Overlay Color	The (user editable) translucent color used to paint an overlay in the Floorplanner view, showing the location of the placement region.
Clock Pre-Routes	If any clock pre-routes exist for a given partition, they are listed here.
Soft	When the placement region is created, it can be defined as a Soft region. Soft regions contain a checkmark in this column. The "soft" placement region status cannot be changed after creation.
Keep Out	When the placement region is created, it can be defined as a Keep Out region. Keep Out regions contain a checkmark in this column. The "keep out" placement region status cannot be changed after creation.
Include Routing	Treats the region as a routing constraint as well as a placement constraint, keeping all routing wires and instances inside the region boundary box.
PR Zone	When the placement region is created, it can be defined as a Partial Reconfiguration (PR) zone. PR Zone regions contain a checkmark in this column. The "PR zone" placement region status cannot be changed after creation.
Resource	The number of <i>Resource</i> instances constrained to this placement region or the number of <i>Resource</i> sites contained within the bounds of this placement region.

Table 68 - Placement Regions View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Show/Hide overlay: region_name		y	Toggles the Visibility checkbox for this placement region, showing or hiding the colored translucent overlay for the placement region in the Floorplanner View (page 43).













Icon	Action	Toolbar Button	Context Menu	Description
	Show or Hide All	Y		Toggles the Visibility checkbox for all placement regions, showing or hiding their colored translucent overlays in the Floorplanner View.
	Select Constrained Instances		Y	Adds all Instances constrained within this placement region to the ACE Selection Set (see Selection View (page 133)).

Table 69 - Placement Regions View Actions (cont.)

Icon	Action	Toolbar Button	Context Menu	Description
	Deselect Constrained Instances		Y	Removes all Instances constrained within this placement region from the ACE Selection Set.
	Change Overlay Color		Y	Allows the choice of which translucent Overlay Color is used to represent this placement region in the Floorplanner View.
	Reset Overlay Color		Y	Resets the chosen placement region overlay color, allowing ACE to automatically pick a new color. If the overlay colors of two placement regions are too similar for easy discernment, another color is pseudo-randomly picked. Each time this action is chosen, another color is picked.
	Reset All Overlay Colors		Y	Pseudo-randomly reassigns new overlay colors for all placement regions.
	Un-Highlight Constrained Instances	Y	Y	Clears the opaque Highlight color for all instances constrained to the selected Placement Region.
	Highlight Constrained Instances	Y	Y	Highlights all instances constrained to the currently selected placement region with the currently-selected opaque highlight color. The highlighted results are visible in the Floorplanner view (see Highlighting Objects in the Floorplanner View (page 343)).
	Choose Highlight Color for next Highlight command	Y	Y	Allows the changing of the current placement region constrained instances opaque highlight color (which is different from the placement region translucent overlay color). This opaque highlight color is used in the Floorplanner view when the Highlight Constrained Instances () action is chosen in the Placement Regions view.

Icon	Action	Toolbar Button	Context Menu	Description
	Zoom to: region_name	Y	Y	Zooms and pans the Floorplanner view to show the location of the selected Placement Region (the Placement Region itself is not visible as an overlay unless the appropriate Visibility checkbox is enabled in the Placement Region View table).
	Print Instances: region_name		Y	Causes a list of all Instances constrained to the selected placement region to be printed in the Tcl Console. Issues the <code>get_region_insts</code> (page 659) Tcl command.
	Remove All Instance Constraints		Y	Removes all Instances from this Placement Region, thus clearing their placement region constraints.
	Delete Placement Region		Y	Removes the selected Placement Region and all associated placement region constraints from the design.
	Configure Clock Pre-Routes...		Y	Allows adding clock pre-route information to the selected partition(s).
	Save Placement Regions	Y		Brings up the Save Placement Regions Dialog (page 184) to save one or all of the Placement Regions definitions and all associated instance placement region constraints.
	Toggle Filter Row Visibility	Y		Changes whether the filter row (of filter icons) at the top of the table is visible or not. This does not alter whether filters are active, it only changes the visibility of the row of filter icons.

Using the Table to Display Placement Regions in the Floorplanner View

Each Placement Region is automatically given a unique translucent overlay color to represent the Placement Region when painting the **Floorplanner view** (page 43). By default, no Placement Region overlays are painted in the Floorplanner view. The Placement Region overlays to be displayed must be enabled. The overlay color may optionally be altered for each or all Placement Regions, but these color choices are not persistent between ACE sessions.

Note

While alternate overlay colors may be chosen for each Placement Region, these overlay colors are not saved between sessions. Each time a design is loaded, new overlay colors are automatically chosen for each Placement Region.


The following are ways to alter the presentation of Placement Region data in the **Floorplanner view** (page 43):

Enable/Disable Painting of Individual Placement Regions Within the Target Device

When the checkbox in the **Visibility** column for a Placement Region is selected, the area of the target device (in the Floorplanner view) representing that Placement Region is painted in the displayed translucent overlay color. When

the checkbox is unchecked, the Floorplanner view is redrawn with the chosen Placement Region overlay no longer painted.

Enable/Disable Painting of all Placement Regions Within the Target Device

When the  **Show/Hide All Placement Regions** action is chosen, the visibility of all Placement Regions is simultaneously either enabled or disabled, causing the Floorplanner view to be repainted appropriately.

Temporarily Alter the Overlay Rendering Color of Individual Placement Regions

The overlay rendering color of each individual Placement Region may be chosen as follows:

1. Right-click anywhere on the row of the desired Placement Region.
2. Select **Choose Overlay Color** from the popup context menu.
3. Use the Color Dialog to choose the desired color for the Placement Region.

This is a temporary color change — colors are reverted to automatically chosen defaults every time ACE starts. During a session, colors are also reverted to the defaults if any of the following are changed:

- The active design
- The active implementation
- The target device

ACE automatically picks a different overlay color for an individual Placement Region if **Reset Overlay Color** is chosen from the right-click popup content menu.

Temporarily Alter the Overlay Rendering Color for All Placement Regions

ACE automatically picks different overlay colors for all Placement Regions if the **Reset All Overlay Colors** action is chosen.

Organizing Table Data

The following are ways to alter the presentation of the data in the Placement Regions table:

Column Resizing

The width of a column can be changed as follows:

1. Place the mouse cursor over the boundary between columns. The mouse cursor changes to indicate resizing is possible.
2. Click and drag left or right to resize the column to the desired width.
3. Release the mouse button.

Column Reordering

The order of the columns in the table can be changed as follows:

1. Click and hold any column name.
2. Drag left or right to move the column between any other pair of columns.
3. Release the mouse button to insert the column header at the new location.

While dragging, the dragged column header appears alongside the mouse cursor. A thick column header separator appears at the present cursor location to show where the column insertion occurs if the mouse button is released.

Filtering

Most table columns can filter the displayed Placement Regions by value. When filtering by column value, only Placement Regions with column values matching the filter are retained; non-matching values are excluded from the table.

- Columns containing text can be filtered by string value. Simple substring text matching (with optional wildcard) is used by default, but Regular Expression matching, also known as RegEx (see https://en.wikipedia.org/wiki/Regular_expression), is also available. The ACE GUI RegEx matching follows Java rules (see <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html>), which are extremely similar to Perl rules.
- Columns with checkmarks can be filtered by boolean value.
- Columns containing numbers can be filtered by numerical value.
- Columns which may not be filtered (i.e., the **Overlay Color** column) lack a filter icon in the filter row.

To add a filter to a column:

1. The Filter row must first be visible. Select the (🔍) **Toggle Filter Row Visibility** action to show the row if necessary.
2. Click the (🔍) filter icon for the desired column, which causes a data-appropriate filter dialog to appear.
3. Fill in the desired filter values and click **Apply** to apply the filter to the rows in the table.

All values matching that filter are retained, and all other values are excluded. Additionally, the background color of the filtered column changes to a bright yellow to indicate the filter is active, and the filter icon at the head of the column also changes to the (🔍) active filter icon.

To edit (or clear) an existing filter:

1. Click the (🔍) active filter icon, which causes the data-appropriate filter dialog to appear pre-populated with the current column filter setting.
2. Change the filter value and click **Apply** to edit the filter.
3. Click **Cancel** to leave the filter unchanged.
4. Click **Clear** to remove the filter from the column.

If the filter is cleared, the background color of the column returns to the default background color, and the filter icon also changes to the (🔍) inactive version.

Partial Reconfig Cluster Value

The partial reconfig cluster value display at the bottom of the view shows a value representing the set of all placement regions marked as "visible" in the view. Making more or fewer placement regions visible updates this value accordingly.

The **Copy hex value to clipboard** button copies the current value to the system clipboard.

The **Send Tcl command** button automatically issues an appropriate `set_impl_option bitstream_prc_cluster_map {partial_reconfig_value}` command.

Projects View

The Projects view provides a hierarchical view of the [projects \(page 222\)](#) in the [workbench \(page 6\)](#). From here, projects can be added and removed, project configurations edited, the active [implementation \(page 229\)](#) can be chosen, saved, or restored, files may be opened for editing, etc.

Clicking an implementation [activates \(page 229\)](#) it. Similarly, clicking a project activates the first implementation in the project definition. The active project and active implementation are both displayed in a bold font.

The various [source files \(page 224\)](#) making up a project are added and removed from this view. Source files of the project are listed in the order in which they are loaded. To change the order the source files are listed or loaded, see [Adding Source Files \(page 297\)](#).

By default, the Projects view is included in the [Projects perspective \(page 6\)](#). To add it to the current perspective, click **Window** → **Show View...** → **Projects**.

For detailed information about managing projects, implementations, source files, etc., see [Working with Projects and Implementations \(page 293\)](#). See also: [Project Management Preference Page \(page 214\)](#).

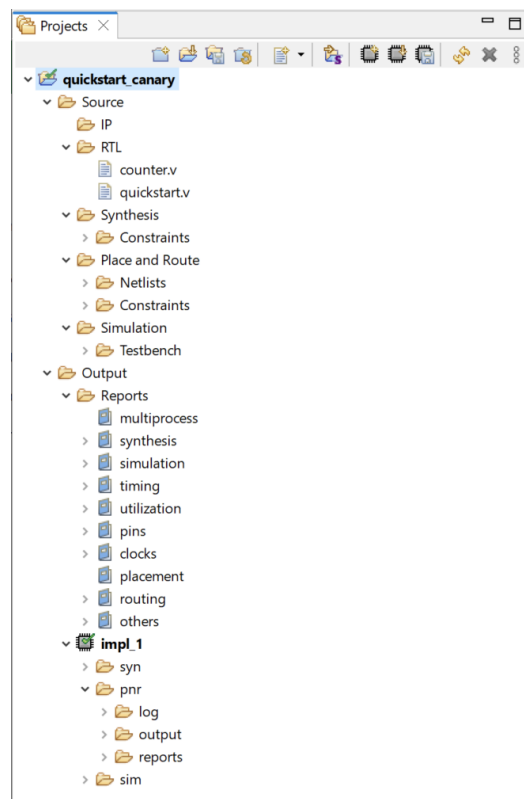
























Figure 47 • Projects View Example

Table 70 • Projects View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Open File		Y	Opens the selected file in a text editor within ACE. See also: display_file (page 627) .
	Create project	Y		Opens the Create Project dialog (page 170) to allow creating a new project definition in the tool. See also: create_project (page 623) .
	Load project	Y		Opens the Load Project dialog (page 175) to allow loading an existing ACE Project file into the tool. See also: load_project (page 667) .
	Save project	Y	Y	Saves the changes to the selected ACE Project to its ACE Project file on disk. See also: save_project (page 707) .
	Save Project As...		Y	Saves the selected ACE project to a newly-chosen filename and location on disk.
	Reload project	Y	Y	Reloads the selected ACE Project. See also: restore_project (page 688) .
	Add source file	Y	Y	This is a parent action, which allows selecting between the file category specific Add Source Files actions that follow.
	Add IP Configuration Files	Y	Y	Opens the Add Source Files Dialogs (page 148) to allow adding source IP Configuration (page 336) files (.acx ip) to the project for use in generation of configurable IP block RTL wrappers or I/O Ring design files. See also: add_project_source_files (page 612)
	Add RTL Files ⁽¹⁾	Y	Y	Opens the Add Source Files Dialogs (page 148) to allow adding source RTL files to the selected project in the Projects view for use in the Synthesis and Simulation flow steps. See also: add_project_source_files (page 612)

Icon	Action	Toolbar Button	Context Menu	Description
	Add Synthesis Constraint Files ⁽²⁾	Y	Y	Opens the Add Source Files Dialogs (page 148) to allow adding source synthesis constraint files to the selected project in the Projects view for use in the Synthesis flow steps. See also: add_project_source_files (page 612)
	Add Place and Route Constraint Files ⁽³⁾	Y	Y	Opens the Add Source Files Dialogs (page 148) to allow adding source place and route constraint files to the selected project in the Projects view for use in the place and route flow steps. See also: add_project_source_files (page 612)
	Add Place and Route Netlist Files ⁽⁴⁾	Y	Y	<p>Opens the Add Source Files Dialogs (page 148) to allow adding source place and route netlist files to the selected project in the Projects view for use in the place and route flow steps.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>The Run Synthesis flow step automatically adds the generated synthesized netlist file to the ACE project place and route netlists after it successfully completes, so there is no need to manually add the synthesized netlist file if running the Run Synthesis flow step. See also: add_project_source_files (page 612).</p> </div>
	Add Simulation Testbench Files ⁽⁵⁾	Y	Y	Opens the Add Source Files Dialogs (page 148) to allow adding source simulation files to the selected project in the Projects view for use in the Simulation flow steps. See also: add_project_source_files (page 612)

Icon	Action	Toolbar Button	Context Menu	Description
	Import Synplify Project File	Y	Y	Opens the Import Synplify Project File Dialog to allow browsing to an existing Synplify Pro project file (*.prj), and import all of its settings into the ACE project. This imports the source RTL, synthesis constraints, HDL include path, and all other project options, and sets them in the ACE project. See also: import_synplify_project_file (page 665) .
	Create implementation	Y	Y	Opens the Create Implementation Dialog (page 166) to allow creating a new project implementation definition for the selected project in the Projects view. See also: create_impl (page 622) Tcl command.
	Restore implementation	Y	Y	Opens the Restore Implementation Dialog (page 180) to allow restoring the active project implementation from a .acxdb Archive file. See also: restore_impl (page 687) Tcl command.
	Rename implementation		Y	Allows renaming the Implementation. See also: rename_impl (page 677) Tcl command.
	Save implementation	Y	Y	Opens the Save Implementation Dialog (page 181) to allow saving the active project implementation to a .acxdb Archive file. See also: save_impl (page 705) Tcl command.
	Open Multiprocess Report		Y	Opens the Multiprocess Summary Report (page 255) for the selected project, if the project has one. See also: display_file (page 627) Tcl command.
	Refresh contents	Y	Y	Refreshes the listing of supporting files contained within the selected project or implementation.

Icon	Action	Toolbar Button	Context Menu	Description
	Remove	Y	Y	Allows removing the selected items from the Projects view. Removing items from a project does not delete the corresponding resources from the file system, except for removing implementation output and report files. See also: remove_project (page 674), remove_project_constraints (page 674), remove_project_netlist (page 675), remove_project_ip (page 675) and remove_impl (page 673) Tcl commands.
	Clone IP		Y	Creates a duplicate of the selected IP and adds it to the project.
	Rename IP		Y	Renames the selected IP.
	Add IP to another project...		Y	Adds the selected IP to another project in the ACE workspace.
	Add copies of IP to another project...		Y	Adds a copy of the selected IP to another project in the ACE workspace.
	Regenerate All IP Design Files		Y	Regenerates HDL (Verilog and, optionally, VHDL), constraint files (*.pdc and *.sdc), etc. for all IP Design files contained in the project (as found in the IP folder of the project in the Projects view). See also: generate_ip_design_files (page 641) Tcl command.
	Group reports by type		Y	If enabled, reports are grouped into subfolders in the Projects view tree.
	Only show HTML reports		Y	If enabled, only HTML reports are shown in the Projects view tree.
	Only show the most recent report		Y	If enabled, only the most recent report is shown in any subfolder in the Projects view tree.
	Expand all children		Y	Recursively expand all tree nodes that are children of the selected tree node















Icon	Action	Toolbar Button	Context Menu	Description
	Collapse all children		Y	Recursively collapse all tree nodes that are children of the selected tree node

Table Notes

1. RTL files are loaded by the Synthesis and Simulator tools in the same order they are displayed within this view. For details on how to change this display/load order, see [Adding Source Files \(page 297\)](#).
2. Synthesis Constraint files are loaded by the Synthesis tool in the same order they are displayed within this view. For details on how to change this display/load order, see [Adding Source Files \(page 297\)](#).
3. Place and Route Constraint files are loaded by ACE in the same order they are displayed within this view. For details on how to change this display/load order, see [Adding Source Files \(page 297\)](#).
4. Place and Route Netlist files are loaded by ACE in the same order they are displayed within this view. For details on how to change this display/load order, see [Adding Source Files \(page 297\)](#).
5. Simulation Testbench files are loaded by the Simulator tool in the same order they are displayed within this view. For details on how to change this display/load order, see [Adding Source Files \(page 297\)](#).

Table 71 - Project View Icons

Icon	Description
	Project.
	Project (Active).
	Project (Save Needed).
	Project (Active, Save Needed).
	Implementation.
	Implementation (Active).
	File.
	Folder.
	Folder containing unsaved changes.
	IP file.

Icon	Description
	IP file with warnings.
	IP file with errors.
	IP file with unsaved changes.

Note

Some files in the "Constraints" or "Netlists" sections of the tree may appear greyed-out to indicate that those constraint files are not enabled in the **Active implementation** (page 229) as shown in the following figure. Various constraint files in a project can be enabled or disabled for an implementation in the **Options view** (page 96), under **Design Preparation** → **Constraint Files**.

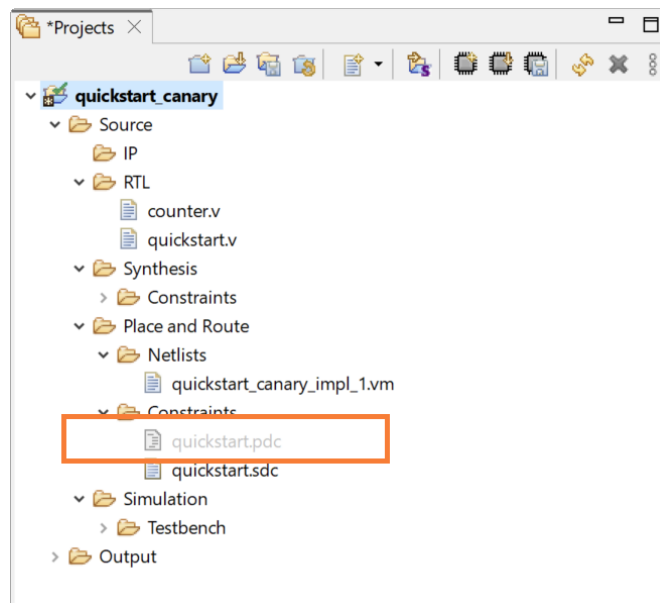


Figure 48 • Disabled Constraints File Example

Properties View

The Properties view can provide in-depth specifics about many types of pin, net, and instance items on demand, and the view then allows navigating many of the relationships between connected items.

To initialize the Properties view with desired information, use the **Display Properties For...** choices found on the right-click context menus of many of the views within the Floorplanner Perspective. The `display_properties` (page 628) Tcl command can also be used to populate the Properties view.

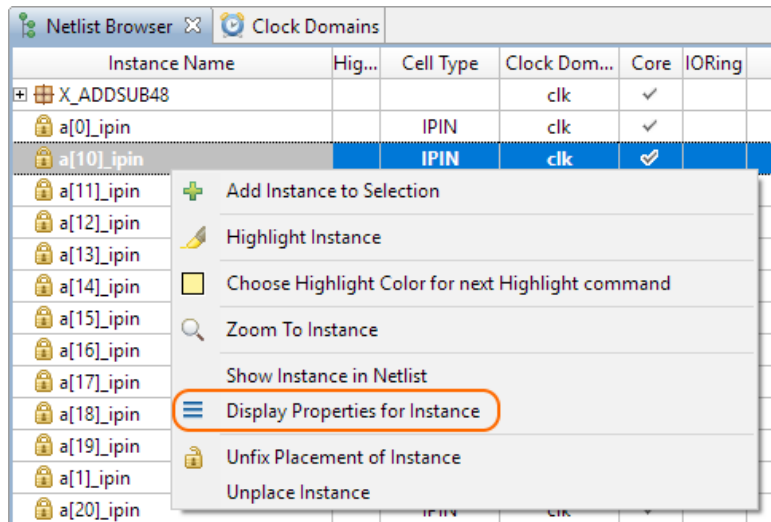


Figure 49 • Properties View Example

General Tab

The **General** tab shows the basic, top-level information about the item.

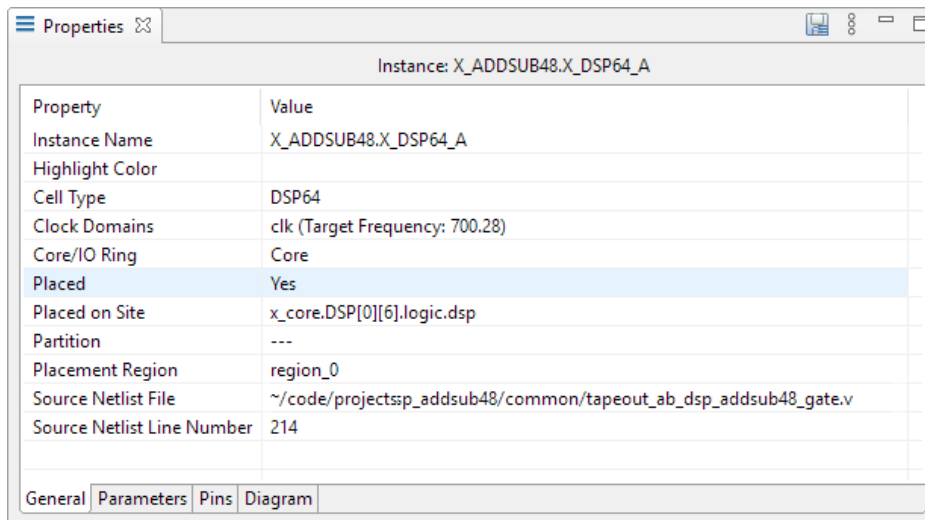


Figure 50 • General Tab Example

Note

Double-click a **Source Netlist File** filename to immediately open that file in the Editor area, or double-click a **Source Netlist Line Number** to immediately open the source netlist file, showing the given line number, in the Editor area.

Parameters Tab

The **Parameters** tab shows the type, name, and value of all of the configurable parameters for the item.

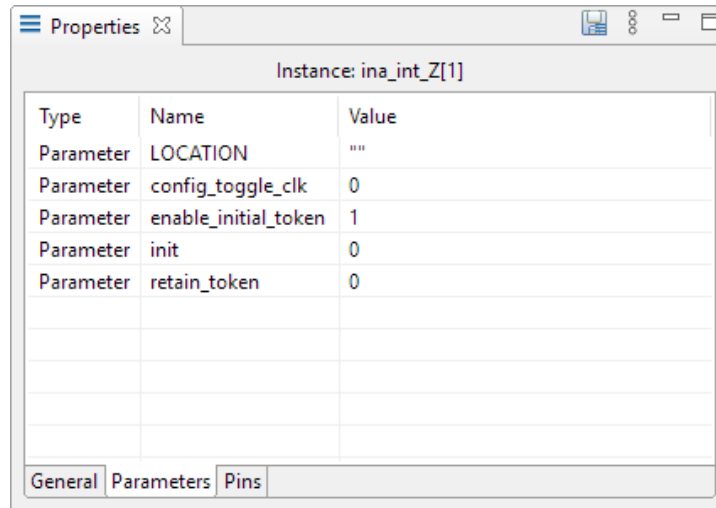


Figure 51 • Parameters Tab Example

Pins Tab

The **Pins** tab shows a variety of information about the item pins.

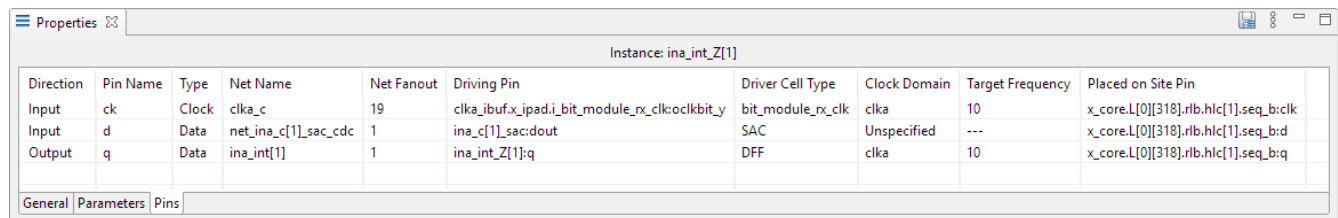


Figure 52 • Pins Tab Example

A number of actions are available on the Pins tab right-click menus as shown in the following table.

Table 72 • Properties View Pins Tab Actions

Icon	Action	Advanced	Description
	Copy Cell Text		Copies the text onto the system clipboard.
	Add to Selection		Adds the item to the ACE selection set (as shown in the Selection view (page 133)).





Icon	Action	Advanced	Description
	Remove from Selection		Removes the item from the ACE selection set (as shown in the Selection view (page 133)).
	Highlight		Applies the currently active highlight color to the chosen item (see Highlighting Objects in the Floorplanner View (page 343)).
	Choose Highlight Color		Determines which color is applied the next time the Highlight action is selected.
	Zoom To		Zooms the Floorplanner view to an area containing the item.
	Show in Netlist ⁽¹⁾		Attempts to open a text editor to the file and line number relevant to the chosen item (available only when a single item is chosen in the view).
	Display Properties		Display properties for the chosen item in the Properties view.

Table Notes

1. **Caution:** The Show in Netlist feature is early access functionality and might not always open the text editor to the expected location.

Note

Reminder: Instances Selection color vs Highlight color priority

- With default preference settings, in the [Floorplanner view \(page 43\)](#), highlight colors of (placed) instances are only visible when the Instances layer is enabled, and the instances are not members of the ACE selection set. This behavior is due to the instance selection color having a higher priority than the highlight color.

Properties Navigation

- Move from one item to another by using the **Display Properties** right-click menu items in the Pins tab.
- Use **Display Properties** to move between related pins, nets, and instances.

Diagram Tab

Some items are complicated or interesting enough to warrant a supplemental diagram. For these types of items, a diagram showing the current item configuration can be found on the **Diagram** tab. Tooltips over the diagram can provide supplemental information where useful.

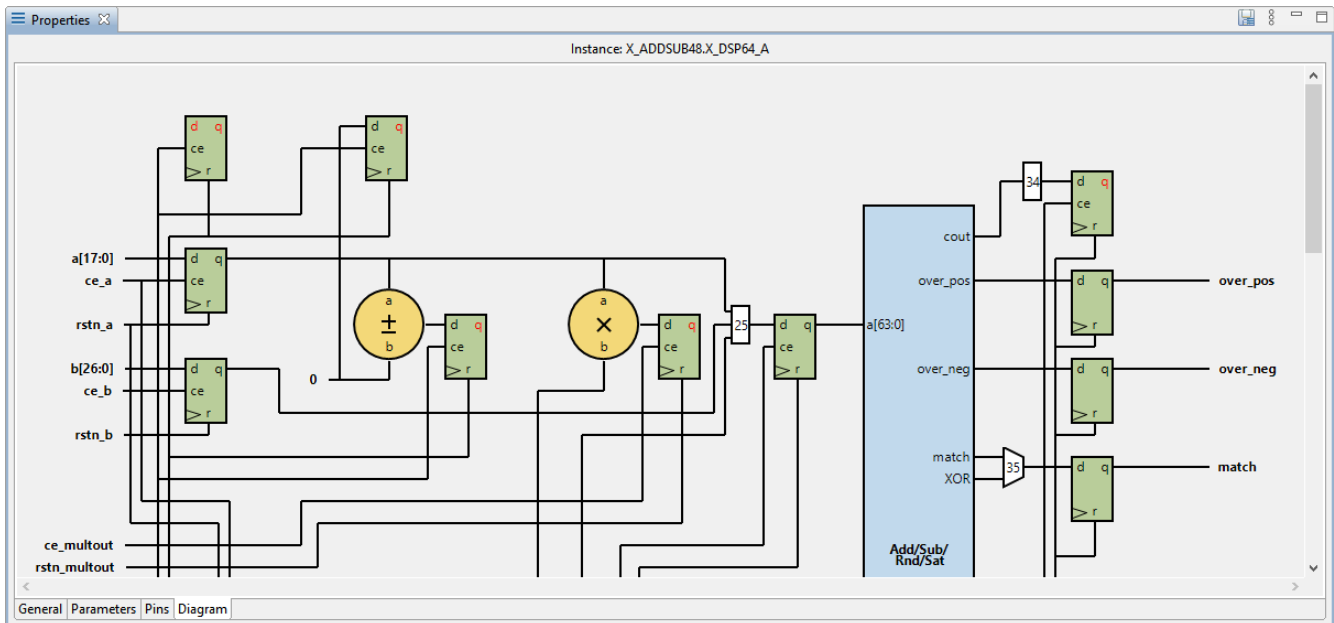


Figure 53 - Diagram Tab Example

Save Properties

The **Save Properties** action can be used to save all changed properties on objects in the database after Prepare has been run. See the [save_properties](#) (page 707) Tcl command reference for details.

This action can be performed by clicking **Save Properties** on the Properties view tool bar:

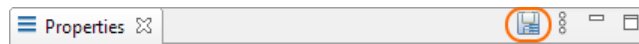


Figure 54 - Save Properties Example

Register Browser View

The Register Browser view can be used to read/write values from/to actual hardware using the (e)FPGA JTAG interface with Tcl commands (see the *JTAG Configuration User Guide* (UG004) for details about the available Tcl commands). The view is typically used for very targeted interactions with a connected FPGA. This tool is not designed for efficient bulk reads of block RAM.

⚠ Caution!

JTAG Connection Requirement

To use the Register Browser view, a valid JTAG connection is required. See [Configuring the JTAG Connection](#) (page 360) and the [Configure JTAG Connection Preference Page](#) (page 190).

Be aware that not all Achronix FPGAs support Register Browser functionality. If the target FPGA for the active implementation does not support this functionality, the view is not interactive.

Values can be read from or written to hardware by right-clicking any row in the table and choosing **Read Values From Chip** or **Write Values To Chip** in the popup context menu. All child rows are read from or written to as well.

The Register Browser interacts with the FPGA at the register level with all reads and writes happening at that level. When writing register field values to hardware, the entire register value is updated via a read-modify-write operation, combining the current hardware register value with the updated register field values.

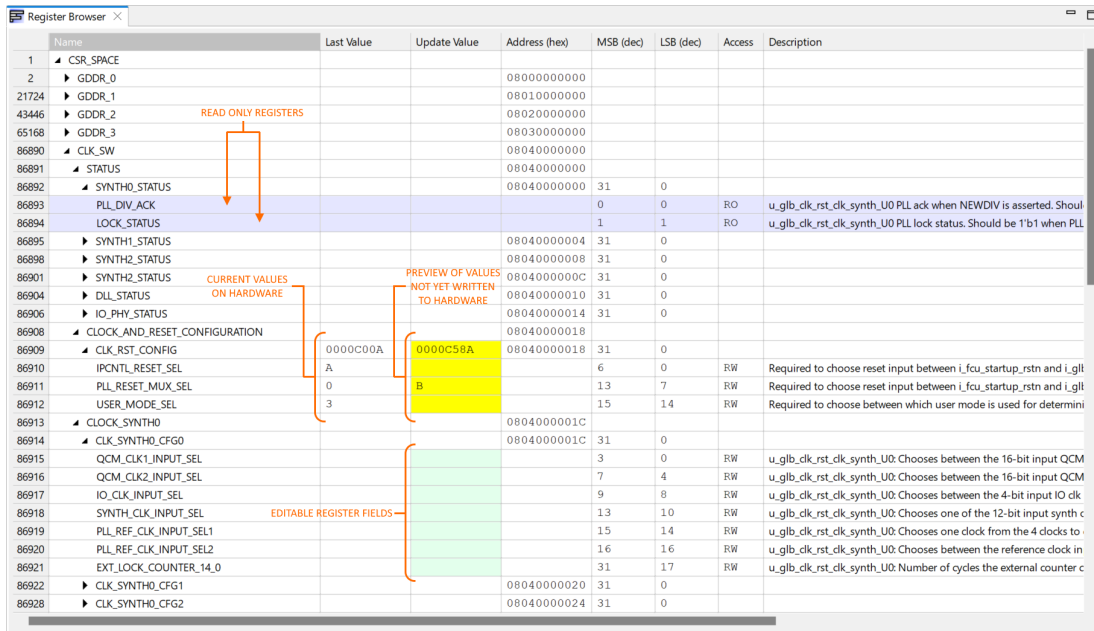


Figure 55 - Register Browser View Example

When the view first appears, even if a compatible device is connected and running, no current register values are pre-populated in the table. Values are only read from the device when commanded to do so. Drill down to just the specific register values needed to query. Be aware that the more rows read or written at a time, the longer it takes. Be careful not to read or write more than is needed.

The tree of register namespaces is organized hierarchically by memory addresses and by conceptual relationships. Fields which are read-only have their entire row colored with a blue background.

Fields with a green background in the **Update Value** column are writable but have not been given a value to be written. Click in any of these cells to enter a hexadecimal value to be written to that field. Invalid values or values that are too large for the given register field are rejected automatically.

When any field within a register has its **Update Value** populated, the background of the field being edited plus those of all other fields within that same register turn yellow, indicating a write is pending but has not yet been committed. The **Update Value** column shows a register value combining the current hardware value of the register (as of the last time one of the fields of that register was edited) combined with any register field values just entered. Put another way, the current register value is read from hardware, and then any values in the yellow register fields for that register are applied to it.

When using **Write Values To Chip**, the current register value at that time is read into ACE, any register field values entered are applied to it, and the updated register value is written back to the device. The **Last Value** column is updated for the register and all fields within that register. All register writes are always performed as a read-modify-write, including when all fields within the register are being edited.

Search View

The Search view provides an interface for searching the ACE design database for design objects (instances, nets, ports, pins, sites, and paths), displaying the results of a search in a list, organized by object type. Optionally, all or part of the results of a search can be added to the current ACE Selection Set, as displayed in the [Selection view \(page 133\)](#). The Search view is a graphical interface to the [find \(page 638\)](#) Tcl command.

Instances and Ports in the results list may be dragged and dropped onto the [Floorplanner view \(page 43\)](#) to assign placement or add [placement region \(page 394\)](#) constraints (the behavior depends upon the Floorplanner active tool/mode). Instances and paths in the results list may be dragged to the [Placement Regions view \(page 110\)](#) to add placement region constraints.

By default, the Search view is included in the [Floorplanner perspective \(page 6\)](#). To add it to the current perspective, select **Window** → **Show View...** → **Search**.

See also: [Object Type Prefixes \(page 335\)](#)

Note

A maximum of 200 objects are displayed in the Search view at a time. Use the arrow buttons (← and →) on the view toolbar to page through the full set of search results.

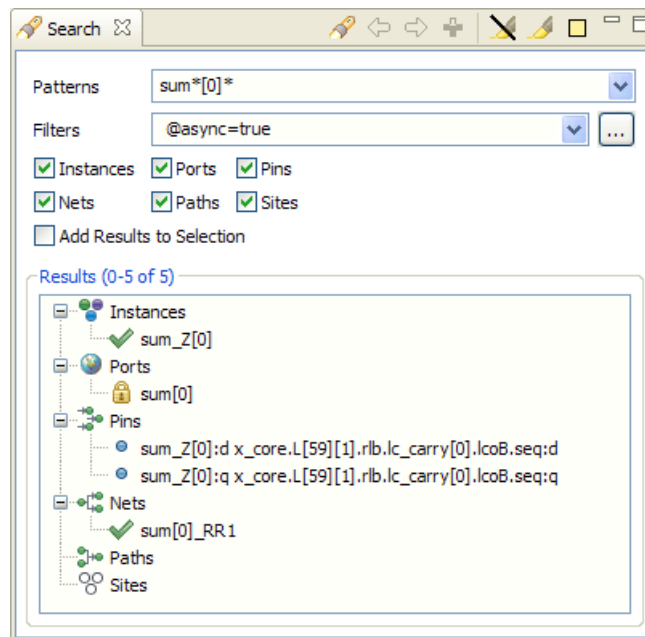















Figure 56 - Search View Example









Table 73 • Search View Icons

Icon	Description
	Object (unplaced instances and ports; all pins, nets, and paths).
	Placed Object (applies to instances and ports).
	Fixed-Placed Object (applies to instances and ports).
	I/O Macro (applies to ports).
	Instances (all instances are under this branch of the search results).
	Ports (all ports are under this branch of the search results).
	Pins (all pins are under this branch of the search results).
	Nets (all nets are under this branch of the search results).
	Paths (all paths are under this branch of the search results).
	Sites (All sites are under this branch of the search results).

Many of the actions in the Selection view are available as both toolbar buttons and right-click context menu choices. Toolbar buttons typically act upon all the listed Search results items, while context menu actions only affect the subset of items currently chosen within the Results list.

Table 74 • Search View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Find objects	Y		Searches for objects in the ACE design database using the search criteria from the Search view.
	Display next 200 results	Y		Displays the next 200 objects in the search results list.
	Display previous 200 results	Y		Displays the previous 200 objects in the search results list.

Icon	Action	Toolbar Button	Context Menu	Description
	Add to selection	Y	Y	Adds all objects that are currently chosen in the Search view "Results" list to the current selection set (as displayed in the Selection View).
	Remove from selection		Y	Removes all objects that are currently chosen in the Search view "Results" list from the current selection set (as displayed in the Selection View).
	Un-highlight Results ⁽¹⁾	Y	Y	Turns off the highlight color for objects.
	Highlight Results	Y	Y	Highlights objects with the currently-selected search highlight color. The highlighted results are visible in the Floorplanner view (other views are not affected by highlights).
	Choose Highlight Color	Y		Allows changing the current highlight color for search result highlighting. This color is used in the Floorplanner view when the Search view () Highlight Results button is clicked.
	Zoom To Object		Y	Zooms the Floorplanner view to a region containing the item currently chosen in the results list.
	Show in Netlist ⁽²⁾		Y	If relevant data exists, opens a text editor to the file and line number relevant to the chosen result item (available only when a single item is chosen in the results list, and that item is an Instance or Net).
	Fix Placement of Instance		Y	Causes the placement state of the chosen Instance to change from unfixed (or soft) to Fixed.
	Unfix Placement of Instance		Y	Causes the placement state of the chosen Instance to change from Fixed to unfixed (soft).
	Unplace Instance(s)		Y	Unplaces all Instances currently chosen in the results list.


Icon	Action	Toolbar Button	Context Menu	Description
	Save Placement Using Search Results	Y		Opens the Save Placement Dialog (page 182) , pre-populating its Specific List of Instances field with the current search query.

Table Notes


1. The Un-highlight Results button stops highlighting the search results in the Floorplanner view. Other views are not affected by highlighting.
2. The Show in Netlist action is early access functionality and might not always open the text editor to the expected location.

Table 75 • Search View Options

Option	Description
Patterns	Enter a Tcl regular-expression pattern which is used to perform a name-based search. Previously used search patterns may be selected from the drop-down menu.
Filters	Enter a search filter to further restrict the search results by properties other than name. Previously used search filters may be selected from the drop-down menu. See Filter Properties (page 264) .
... (Search Filter Builder)	This button opens the Search Filter Builder Dialog (page 186) providing a guide through the options available for search filters.
Instances ⁽¹⁾	When checked, includes Instances in the search results.
Ports ⁽¹⁾	When checked, includes Ports in the search results.
Pins ⁽¹⁾	When checked, includes Pins in the search results.
Nets ⁽¹⁾	When checked, includes Nets in the search results.
Paths ⁽¹⁾	When checked, includes Paths in the search results.
Sites ⁽¹⁾	When checked, includes Sites in the search results.
Add Results to Selection	If selected when a search is performed, all the results of that search are added to the ACE selection set.

Option	Description
<p>Table Notes</p> <ol style="list-style-type: none"> Caution: If none of the object-type option checkboxes are checked, the search is performed as if all types were checked. 	

Search Results and ACE Selection

The complete results of a search may be added to the current ACE Selection Set (and thus rendered in a special color, by default a bright green, in the Floorplanner view) by checking the **Add Results to Selection** checkbox before starting the search. A subset of the search results may be added to the current ACE selection set by selecting the desired additions in the search "Results" list and pressing the () **Add to Selection** button, or, double-clicking a single entry in the "Results" list adds it to the current selection.

Search Highlights

There is typically a tremendous amount of visualization data available in the Floorplanner view. The granular highlighting allowed by the Search view, the Selection view, and the Highlight functionality (see [Highlighting Objects in the Floorplanner View \(page 343\)](#)) is an attempt to help turn this data into useful information, to find and focus on specific information within the user designs.

By highlighting multiple search result sets in the same or different colors, desired information is made more visible in the graphical views. By selectively un-highlighting or re-highlighting smaller (more specific) result sets (which are a subset of already-highlighted objects), focus may be directed to just the objects of interest.

When used in combination with the layering functionality (see the Layers portion of the toolbox for the Floorplanner view) and Selection functionality (see the [Selection view \(page 133\)](#) as well as the [select \(page 708\)](#) and [deselect \(page 625\)](#) Tcl commands), a graphical visualization can be achieved at whatever granularity is desired.

Caution!



The selection color takes precedence over the highlight color by default. If design objects are both highlighted and selected, they are painted the selection color (bright green by default) in the Floorplanner view. To see the design objects painted in the highlight color (with default precedence settings), the objects must first be removed from the current selection set (as shown in the Selection view). The Floorplanner view settings (including precedence) for highlight and selection colors can be manipulated on the [Floorplanner View Colors and Layers Preference Page \(page 194\)](#).

Selection View

The Selection view provides an interface that allows viewing and managing the current selection set. A selection set consists of a collection of ACE design database objects. The selection set may also be manipulated with the [select \(page 708\)](#) and [deselect \(page 625\)](#) Tcl commands.

The Selection view displays the current selection set in a list, organized by object type. The object type groupings are Instances, Ports, Pins, Nets, Paths, and Sites; these are the only object types which may be Selected.

Note

A maximum of 200 objects are displayed in the Selection view at a time. Use the arrow buttons ( and ) on the view toolbar to page through the full content of the ACE selection set.

The current page of selected objects in the Selection view is also displayed with special coloration (by default a bright green) in the **Floorplanner view** (page 43).

Objects may be added to the selection set from the **Search view** (page 129) if **Add Results to Selection** is checked when a search is issued, or by choosing individual objects from the search results and selecting **Add to Selection**, or from the Floorplanner view (see **Selecting Floorplanner Objects** (page 341)).

Various drag-and-drop operations may be initiated by dragging single or (in some cases) multiple items from the selection list to other views in the Floorplanner perspective. If dragging all selected objects of a given type (i.e., Instances), including those not in the current page of 200 selected objects, the node with that type name (i.e., Instances) may be dragged. Be aware that some drag-and-drop operations, such as pre-placement assignment, does not work with multiple, simultaneously selected objects.

Instances and Ports in the selection list may be dragged and dropped onto the Floorplanner view to assign pre-placement or add **placement region** (page 394) constraints. The behavior depends upon the active Floorplanner tool or mode. Instances and Paths in the results list may be dragged to the **Placement Regions view** (page 110) to add placement region constraints.

By default, the Selection view is included in the **Floorplanner perspective** (page 6). To add it to the current perspective, select **Window** → **Show View...** → **Other...** → **Selection**.

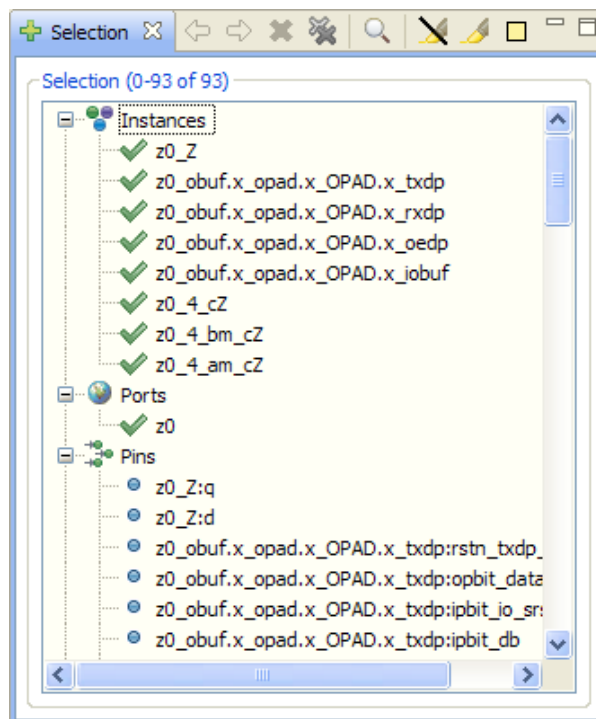
















Figure 57 • Selection View Example

Table 76 • Selection View Icons

Icon	Description
	Object.
	Placed Object (applies to instances and ports).
	Fixed-Placed Object (applies to instances and ports).
	I/O Macro (applies to ports).
	Instances (all instances are under this branch of the selection).
	Ports (all ports are under this branch of the selection).
	Pins (all pins are under this branch of the selection).
	Nets (all nets are under this branch of the selection).
	Paths (all paths are under this branch of the selection).
	Sites (all sites are under this branch of the selection).

Many of the actions available in the Selection view are available as both toolbar buttons and right-click context menu choices. Toolbar buttons act upon all the listed Selection items, while context menu actions only affect the subset of items currently chosen within the Selection list. Be aware that available right-click context menu choices vary depending upon the context: the number and the type of the items alter the available actions.

Table 77 • Selection View Actions

Icon	Action	Toolbar Button	Context Menu	Description
	Zoom to Full Selection Set	Y		Zooms the Floorplanner view to a region containing the current list of chosen objects in the Selection view.
	Zoom to Object		Y	Zooms the Floorplanner view to a region containing the currently chosen object in the Selection view.
	Display next 200 objects	Y		Displays the next 200 objects in the selection set.
	Display Previous 200 objects	Y		Displays the previous 200 objects in the selection set.










Icon	Action	Toolbar Button	Context Menu	Description
	Deselect object	Y	Y	Deselects objects in the Selection view list, removing them from the current selection set in ACE.
	Deselect all objects	Y		Deselects all objects in the current selection set in ACE, resulting in an empty selection set.
	Un-Highlight Selection ⁽¹⁾	Y	Y	Turns off the highlight color for objects in the current selection.
	Highlight Selection	Y	Y	Sets the highlight color for objects in the current ACE selection set to the currently-chosen highlight color (the highlight coloring is only visible in the Floorplanner view after the objects are no longer Selected, since the Selection color overrides the highlight color).
	Choose Highlight Color	Y		Allows changing the current ACE selection set highlight color (which is different from and overridden by the Selection color). This color is used in the Floorplanner view when the  Highlight Selection action is chosen in the Selection view.
	Show in Netlist ⁽²⁾		Y	If relevant data exists, opens a text editor to the file and line number relevant to the chosen Selection item (available only when a single item is chosen in the Selection list, and that item is an Instance or Net).
	Fix Instance Placement		Y	Causes the placement state of the Instance under the mouse cursor to change from unfixed (or soft) to Fixed.
	Unfix Instance Placement		Y	Causes the placement state of the Instance under the mouse cursor to change from Fixed to unfixed (or soft).
	Unplace Instance(s)		Y	Unplaces the Instances chosen in the view.
	Unplace All Instances in ACE Selection Set		Y	Unplaces all Instances that are members of the current ACE selection set.
	Save Placement of Selection Set	Y		Opens the Save Placement Dialog (page 182), pre-populating its Specific List of Instances field with the query to obtain the active Selection set.

Table Notes

1. Stops highlighting the ACE selection set in the Floorplanner view. Other views are not affected by highlighting.
2. The Show in Netlist action is early access functionality and may not always open the text editor to the expected location.



For more information about the interaction between Selection and Highlighting, please see [Search Highlights \(page 133\)](#) as well as [Highlighting Objects in the Floorplanner View \(page 343\)](#).

See also: [Object Type Prefixes \(page 335\)](#).

Snapshot Debugger View

The Snapshot Debugger view provides a graphical interface for controlling an embedded Snapshot IP block in a programmed Achronix device. By default, the Snapshot Debugger view is included in the [Programming and Debug Perspective \(page 6\)](#). To access the Snapshot Debugger view from any other perspective, select **Window** → **Show View** → **Other...** → **Achronix** → **Snapshot Debugger**.

This view allows [running the Snapshot Debugger \(page 364\)](#) embedded in the design. A simple button press [collects live sample data \(page 376\)](#) in a VCD file. This view also allows [configuring the debug capture trigger pattern\(s\) \(page 367\)](#), [configuring a test stimulus \(page 372\)](#), and [configuring the data capture ranges \(page 374\)](#) before and after the trigger(s).

For convenience, favorite Snapshot configurations can be () [saved \(page 387\)](#) and () [loaded \(page 387\)](#) via the view toolbar buttons. Saved configurations may also be used to drive the [snapshot in Batch mode \(page 387\)](#) via Tcl.

Tip


When a user design containing the ACX_SNAPSHOT macro completes the [Flow step \(page 234\) Run Prepare](#), a names .snapshot configuration file, is automatically generated. This file contains harvested information from the design including the widths, depths and signal names for the monitor, trigger, and stimuli busses, user clock frequency, and default log and .vcd file path settings. When an [active Project and Implementation \(page 229\)](#) is available, the Snapshot view automatically loads the names .snapshot file to pre-populate the relevant fields of the view.

Note

When the names .snapshot file is generated, the file contains only a subset of a complete Snapshot configuration, and thus a generated names .snapshot file should not be used to drive [Snapshot in batch mode \(page 387\)](#) via Tcl.

See also: [Running the Snapshot Debugger \(page 364\)](#), [Assign Bussed Values Dialog \(page 153\)](#), [Assign Bussed Signal Names Dialog \(page 151\)](#), [VCD Waveform Editor \(page 11\)](#), and the [Snapshot User Guide \(UG016\)](#).

Table 78 • Snapshot Debugger View Toolbar Buttons

Icon	Action	Description
	Arm Snapshot	Performs the following steps: <ol style="list-style-type: none"> 1. Sends the trigger conditions configuration to the Snapshot Debugger core. 2. Send the Stimulus value to the design-under-test. 3. Waits for the trigger condition to be met. 4. Retrieves the trace buffer contents. 5. Outputs a VCD file.

Icon	Action	Description
	Cancel Snapshot	Cancels the () Arm Snapshot by stopping the polling process and then resetting the ACX_SNAPSHOT macro.
	Save Snapshot Configuration	Saves the current settings of the Snapshot view to a text file. See Saving/Loading Snapshot Configurations (page 387) .
	Load Snapshot Configuration	Loads a previously saved configuration file. See Saving/Loading Snapshot Configurations (page 387) .
	Capture Snapshot Startup Trigger	Requires that the initial startup trigger parameters on the ACX_SNAPSHOT macro have been configured to enable the Startup Trigger feature, and that the Arm Snapshot action has not been executed since the bitstream has been programmed. Performs the following steps: <ol style="list-style-type: none"> 1. Waits for the startup trigger condition to be met. 2. Retrieves the trace buffer contents. 3. Outputs a VCD file.
	Configure JTAG Interface	Opens the preferences dialog with the Configure JTAG Connection Preference Page (page 190) visible. See Configuring the JTAG Connection (page 360) .

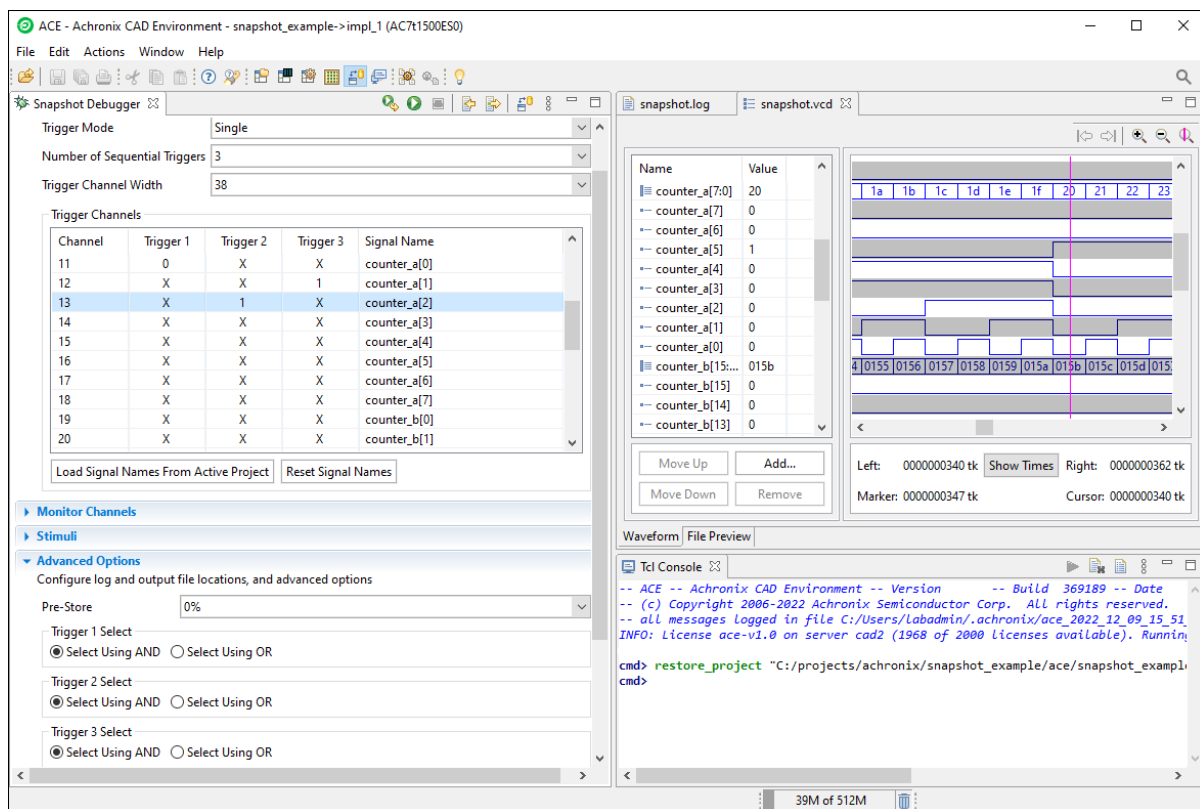





Figure 58 - Snapshot Debugger View Example

Table 79 - Snapshot Debugger View Options

Option	Description
Trigger Conditions	
Trigger Mode	<p>Allows selecting the trigger mode to use when the Arm Snapshot action is run. The default trigger mode is Single:</p> <ol style="list-style-type: none"> 1. The trigger conditions are programmed into the ACX_SNAPSHOT macro. 2. The GUI waits for a single trigger event to occur which matches those trigger conditions. 3. A single VCD file is recorded. <p>If Immediate trigger mode is selected, pressing the Arm button results in the same behavior as Single trigger mode, except that all 3 trigger patterns are treated as "Don't Care" (X) so that the trigger event occurs as soon as the Arm button is pressed.</p> <p>If Repetitive trigger mode is selected:</p> <ol style="list-style-type: none"> 1. The trigger conditions are programmed into the ACX_SNAPSHOT macro. 2. Samples are captured repetitively until the upper limit of trigger event records is reached. <p>When Repetitive trigger mode is selected, an additional set of repetitive trigger mode options appear allowing the configuration of:</p> <ul style="list-style-type: none"> • The number of sequential times Snapshot should be armed repetitively using the configured trigger conditions • The way in which the output VCD files are managed
Number of Sequential Triggers	<p>Allows selecting the use of either 1, 2, or 3 sequential triggers:</p> <p>1 – Trigger 2 and Trigger 3 are ignored during the match</p> <p>2 – Trigger 3 is ignored during the match and Snapshot triggers when Trigger 1 is matched, followed on any subsequent clock by a match on Trigger 2</p> <p>3 – snapshot triggers after a match on Trigger 1, followed by Trigger 2, followed by Trigger 3</p> <p>See Configuring the Trigger Pattern (page 367), Configuring Test Stimulus (page 372), and Configuring the Monitor Signals (page 371).</p>
Trigger Channel Width	<p>The Snapshot debugger module can be configured to trigger channel widths of 1 to 40 channels. The Trigger Channel Width must be set to the value that corresponds with the configured Snapshot RTL instantiation. The trigger width is automatically extracted from the user design and saved in the generated <code>names.snapshot</code> file, which can be loaded and edited.</p>
Channel	<p>The trigger channel number connected to the Snapshot Debugger core.</p>
Trigger 1	<p>Sets the Trigger 1 value for each channel. Valid options are:</p> <p>X – don't care</p> <p>R – rising edge</p> <p>F – falling edge</p> <p>0 – level 0</p> <p>1 – level 1</p> <p>See Configuring the Trigger Pattern (page 367).</p>
Trigger 2	<p>Sets the trigger 2 value for each channel. Valid options are:</p> <p>X – don't care</p> <p>R – rising edge</p> <p>F – falling edge</p> <p>0 – level 0</p> <p>1 – level 1</p> <p>This column is only editable if 2 or 3 triggers are selected. See Configuring the Trigger Pattern (page 367).</p>

Option	Description
Trigger 3	<p>Sets the trigger 3 value for each channel. Valid options are:</p> <ul style="list-style-type: none"> X – don't care R – rising edge F – falling edge 0 – level 0 1 – level 1 <p>This column is only editable if 3 triggers are selected. See Configuring the Trigger Pattern (page 367).</p>
Signal Name	Sets the user-defined name for the trigger channel. This signal name is automatically extracted from the user design and saved in the generated <code>names.snapshot</code> file, which can be loaded and edited.
Load Signal Names From Active Project	When clicked, loads the <code>names.snapshot</code> file generated during design preparation (the Run Prepare flow step), which renames all signals with their project-specific names and loads other harvested project-specific settings.
Reset Signal Names	When clicked, renames all signals back to their default names, which is "signal" with a suffix corresponding to the channel number.
Repetitive Trigger Settings	
Record Limit	The repetitive trigger Record Limit setting determines how many times (number of records) the GUI repeatedly Arms the Snapshot debugger and captures samples. This may be set to automatically run Snapshot up to 128 times.
VCD Record Limit	Determines how many repetitively triggered Snapshot records to capture in a single VCD file. Essentially concatenates the VCD files from consecutive runs of Snapshot (records) into a single VCD file. The VCD file waveform contains a set of virtual signals to indicate the system timestamp at which each Snapshot record was captured. Up to 10 Snapshot records may be concatenated in a single VCD file.
Overwrite VCD File	<p>When selected, the VCD Waveform File name specified in the Advanced Options section is used to store the output VCD file. The file is overwritten with the new VCD file each time the VCD record limit is reached. If not selected, multiple VCD files are written out and a unique VCD record number is added to the VCD Waveform File name specified in the Advanced Options section for each VCD.</p> <p>For example, if the Record Limit is set to 8, the VCD Record Limit is set to 2, and the VCD Waveform file path set to <code>./snapshot.vcd</code>, Snapshot outputs 4 VCD files:</p> <ol style="list-style-type: none"> 1. <code>./snapshot1.vcd</code> 2. <code>./snapshot2.vcd</code> 3. <code>./snapshot3.vcd</code> 4. <code>./snapshot4.vcd</code> <p>Each file contains 2 Snapshot capture records.</p>
Monitor Channels	
Monitor Channel Width	The Snapshot debugger module can be configured to monitor channel widths of 1 to 4087 channels. The Monitor Channel Width must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The monitor width is automatically extracted from the user design and saved in the generated <code>names.snapshot</code> file, which can be loaded and edited.
Number of Samples	The Snapshot debugger module can be configured to capture between 512 and 16384 samples. The Number of Samples must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The number of samples is automatically extracted from the user design and saved in the generated <code>names.snapshot</code> file, which can be loaded and edited.
Channel	The monitor channel number connected to the Snapshot Debugger core.

Option	Description
Signal Name	Sets the user-defined name for the monitor channel. This signal name is automatically extracted from the user design and saved in the generated <code>names . snapshot</code> file, which can be loaded and edited. This signal name is used in the VCD file waveform output.
Load Signal Names From Active Project	When clicked, loads the <code>names . snapshot</code> file generated during design preparation (the Run Prepare flow step), which renames all signals with their project-specific names and loads other harvested project-specific settings.
Reset Signal Names	When clicked, renames all signals back to their default names, which are "signal" with a suffix corresponding to the channel number.
Stimuli	
Stimuli Channel Width	The Snapshot debugger module can be configured to stimuli channel widths of 0 (no stimuli) to 512 channels. The Stimuli Channel Width must be set to the value that corresponds with the parameterized Snapshot RTL instantiation. The stimuli width is automatically extracted from the user design and saved in the generated <code>names . snapshot</code> file, which can be loaded and edited.
Channel	The stimuli channel number connected to the Snapshot Debugger core.
Value	The value to drive out on this stimuli channel ARM_DELAY cycles before Snapshot is Armed (when the Arm button is pressed).
Signal Name	Sets the user-defined name for the stimuli channel. This signal name is automatically extracted from the user design and saved in the generated <code>names . snapshot</code> file, which can be loaded and edited.
Advanced Options	
Pre-Store	Controls the ratio of samples collected before and after the trigger. See Configuring Advanced Options (page 374) .
Trigger 1 Select	When set to Select Using AND , Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that ALL signal values not masked must match the specified pattern in order to generate a trigger match event. When set to Select Using OR , Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if ANY of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (page 367) .
Trigger 2 Select	When set to Select Using AND , Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that ALL signal values not masked must match the specified pattern in order to generate a trigger match event. When set to Select Using OR , Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if ANY of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (page 367) .
Trigger 3 Select	When set to Select Using AND , Snapshot ANDs the values within the active Trigger to determine a match. This setting indicates that ALL signal values not masked must match the specified pattern in order to generate a trigger match event. When set to Select Using OR , Snapshot ORs the values within the active Trigger to determine a match. This setting indicates the trigger match event is generated if ANY of the non-masked signal values match the specified pattern. See Configuring the Trigger Pattern (page 367) .
Frequency (MHz)	Must be configured to match the the <code>user_clk</code> timing constraint set in the SDC file of the design being debugged. This is automatically set according to the values captured in the <code>names . snapshot</code> file when an active implementation is available. See Configuring Advanced Options (page 374) .

Option	Description
File Paths Relative To	(Only relevant when the log and VCD file paths provided are relative paths, not absolute paths.) Chooses whether the Log File and Waveform File paths are understood to be relative to the Active Project directory or to the Working Directory . (When using the GUI, the working directory corresponds to the <code>pwd</code> according to the Tcl interpreter.) See Configuring Advanced Options (page 374) .
Log File	File name for the Snapshot log file, where raw Snapshot output (including warning and error messages) is logged. The default file name and path can be overwritten, and the accompanying Browse button may be used to graphically navigate to the desired directory or file. See Configuring Advanced Options (page 374) .
Waveform File	File name for the Snapshot VCD waveform output file, where the Snapshot sampled values (the trace buffer) is stored. The default file name and path can be overwritten, and the accompanying Browse button may be used to graphically navigate to the desired directory or file. See Configuring Advanced Options (page 374) .
Startup Trigger	This is the same as the () Capture Snapshot Startup Trigger button in the view toolbar. See Collecting Samples of the User Design (page 376) .
Arm	This is the same as the () Arm Snapshot button in the view toolbar. See Collecting Samples of the User Design (page 376) .
Cancel	This is the same as the () Cancel Snapshot button in the view toolbar. See Collecting Samples of the User Design (page 376) .

Tcl Console View

The Tcl Console view provides an interactive Tcl console for ACE. All user interactions that change design and project data go through the Tcl command interface, including all commands executed while in the GUI. From the Tcl Console view, executed commands and their information are displayed, including any warning and error messages. The Tcl console can also be used interactively by typing Tcl commands directly into the console to manipulate projects or the current design.

The following highlights are used:

- Bold Green** – valid ACE commands.
- Blue Text** – informational messages.
- Yellow Text** – warning messages.
- Red Text** – error messages.

The Tcl Console view consists of two main areas: a "readout" area at the top, and an "entry" area at the bottom.

- When the cursor is in the entry area, the up and down arrow keys can be used to move forward and backward through the history of recently-issued commands.
- When typing in a command or file path, pressing the **TAB** key causes an auto-completion window to pop up. The up and down arrow keys can be used to move through entries in the content-assist list. Pressing **Enter** chooses the selected entry in the list. Entries in the list may also be chosen with the mouse.
- When the focus is in the entry area:
 - Pressing **ESCAPE** clears the currently entered command text. If there is no command text in the entry area, pressing **ESCAPE** clears any row selection from the readout area.

- Pressing **CTRL+V** pastes the contents of the system clipboard into the entry area.
- Pressing **CTRL+SPACE** jumps into the multi-line entry dialog, useful to execute more complicated Tcl commands.
- When the focus is in the multi-line entry dialog, Pressing **ALT+UP ARROW** and **ALT+DOWN ARROW** can be used to move forward and backward through the history of recently-used commands.
- When the focus is in the readout area:
 - Pressing **ESCAPE** moves the focus to the entry area.
 - Pressing **CTRL+C** copies the contents of the currently selected row(s) to the system clipboard.
 - Pressing **CTRL+V** pastes the contents of the system clipboard into the entry area.
 - Pressing **CTRL+F** performs a "find" operation. A dialog appears to permit searching through all of the text in the readout area.
 - Typing characters other than navigation keys (arrows, home, end, page up, page down, etc.) moves the focus to the readout area.
- When the cursor is anywhere over the readout or entry areas, **CTRL+MOUSEWHEEL** can be used to quickly increase or decrease the size of the font.

The colors of all the different elements of the view can be configured by selecting the **Configure View...** action and choosing **Colors and Fonts**.

By default, the Tcl Console view is included in all [perspectives \(page 6\)](#). If it is not visible, add it with **Window** → **Show View** → **Tcl Console**.

For more details, see [Using the Tcl Console \(page 332\)](#), check the available preferences on the [Tcl Console View Preference Page \(page 216\)](#), and see the available commands in the [Tcl Command Reference \(page 580\)](#).

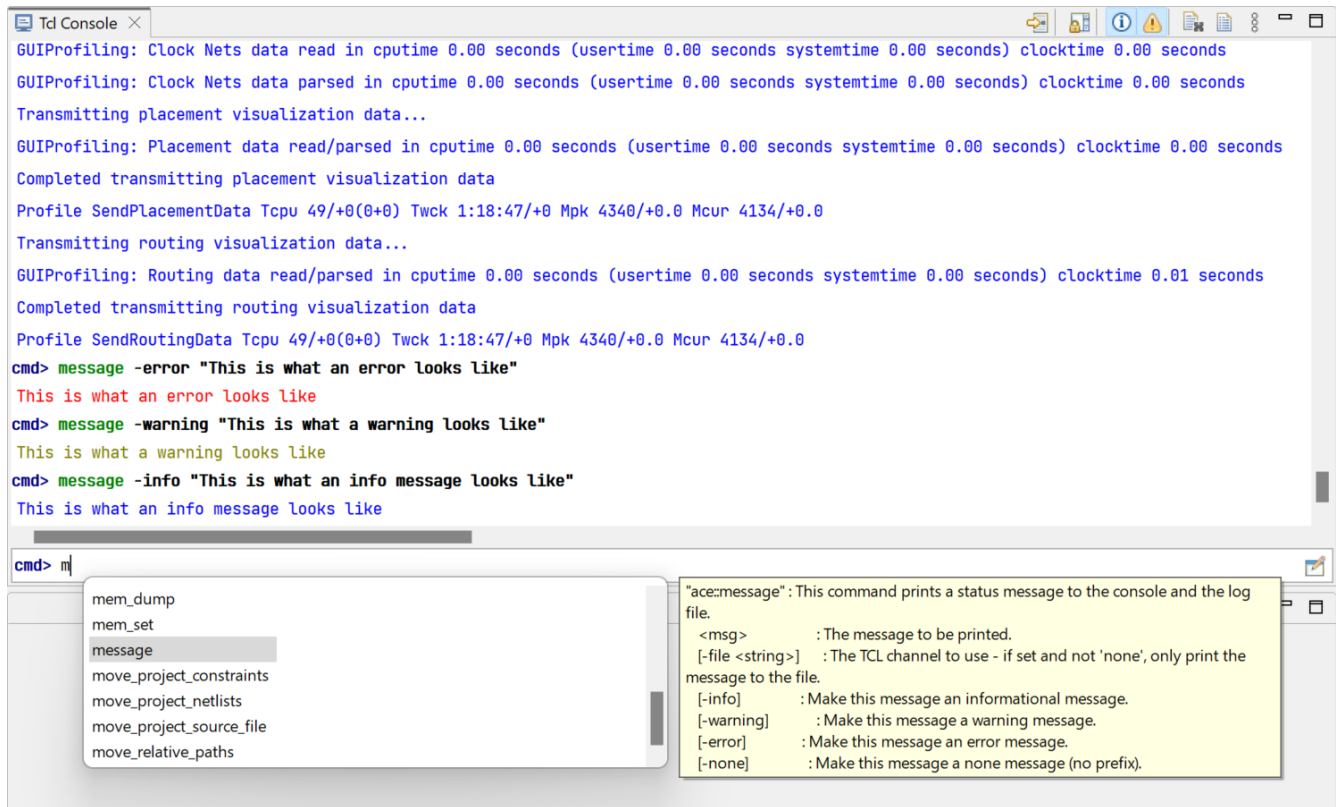




Figure 59 • Tcl Console View Example

Table 80 • Tcl Console View Actions

Icon	Action	Description
	Find...	Find instances of specified text in the readout area.
	Scroll Lock	When enabled, the readout area does not automatically scroll as new lines are added.
	Toggle Info Messages	Toggles the display of info messages in the readout area.
	Toggle Warning Messages	Toggles the display of warning messages in the readout area.
	Clear Console	Clears the readout area.
	Open Log File	Opens the log file on disk in the ACE editor area.
	Configure view	Opens the Configure View... menu item.

Icon	Action	Description
	Multi-line entry	Opens the multi-line entry dialog.
	View entire line	When one or more lines are selected in the readout area, displays the lines in a floating dialog including a "word wrap" toggle to assist with very-long-lines.

Warning!

When Tcl command return values are displayed in the Tcl Console, any long returned values are visually truncated to 500 characters in the console. The actual returned value is not truncated, only the visual representation in the console. Thus, scripts using long return values still behave properly.

Right-click anywhere in the readout area to access a context menu of actions.

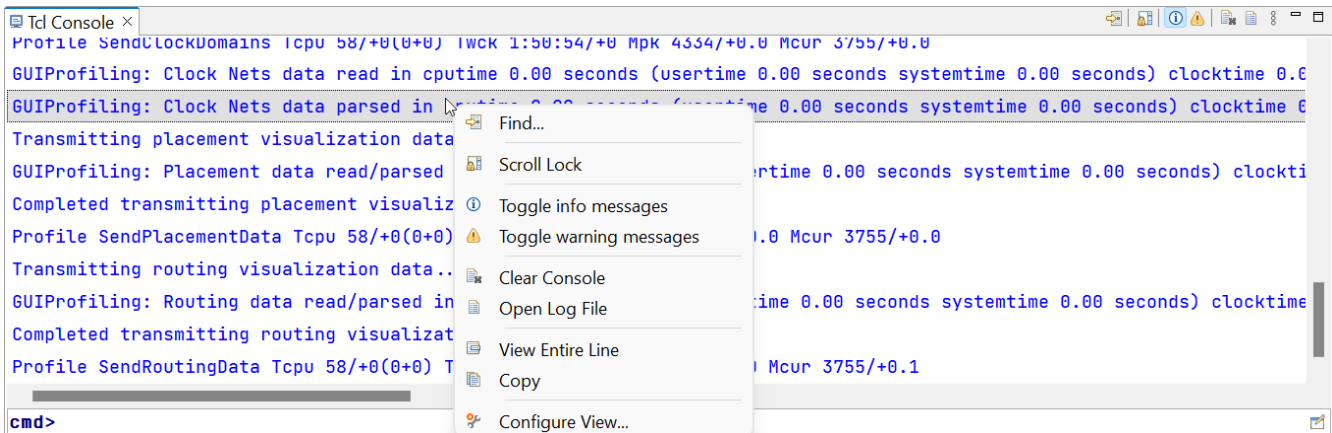


Figure 60 • Readout Area Context Menu Example

Dialogs

Several dialogs are used within ACE. These dialogs are typically shown in response to specific menu choices or button clicks.

Add Signals to Waveform Viewer Dialog

The Add Signals to Waveform Viewer dialog appears when the **Add...** button is clicked in the **VCD Waveform editor** (page 11).

This dialog allows:

1. Making signals visible which were previously hidden by clicking the **Remove** button in the VCD Waveform editor.
2. Adding duplicates of already-shown signals to the table and waveform.

The choice of which rows (if any) are selected in the VCD Waveform editor signal value table when the **Add** button is pressed affects where the to-be-added signals appear in the signal values table. The signals chosen in the dialog are added or inserted using the **Append** and **Insert** buttons. When the dialog **Insert** button is pressed, signals chosen in the dialog are inserted above the first row currently selected in the VCD signal value table, or at the top of the table if no rows are currently selected. Alternatively, when the **Append** button is pressed, signals are added to the bottom of the table, regardless of which rows are selected. Remember that the **Move Up** and **Move Down** buttons on the VCD Waveform editor may be used to move the added/inserted signals to a different location in the table after this dialog is closed.

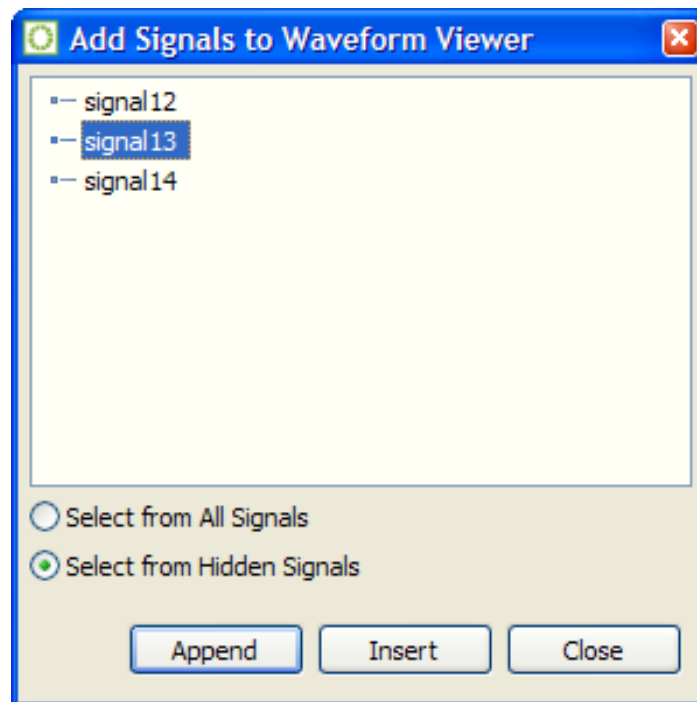


Figure 61 - Add Signals to Waveform Viewer Dialog Example

The majority of the dialog is taken up by an area listing the signals. The listed signals are filtered depending upon the radio-button currently selected in the dialog.

Table 81 - Add Signals to Waveform Viewer Dialog Actions

Action	Description
Select from All Signals	Causes the list to be populated with all signals contained in the current VCD file.
Select from Hidden Signals	Causes the list to be populated with all signals found in the VCD file which are currently hidden (includes signals removed from the VCD Waveform editor signal table). If no signals are currently hidden, the list of hidden signals is empty. When collapsed, the bits of buses are not considered hidden, and thus do not appear in this listing.
Append ⁽¹⁾	Appends the currently-selected signal to the bottom of the VCD Waveform editor signal table.
Insert ⁽¹⁾	Inserts the signal(s) currently selected in the dialog immediately above the signal currently selected in the VCD Waveform editor signal table. If no signal was selected when the Add... button was pressed to open this dialog, the signal selected in the dialog is inserted at the top of the VCD Waveform editor signal table.
Close	Closes the dialog.

Table Notes

- The **Append** and **Insert** buttons have the following characteristics:
 - Each button may be clicked multiple times for a given signal, adding the signal selected in the dialog list to the VCD Waveform editor signal table multiple times.
 - The buttons are disabled if no signal is currently selected in the dialog signal list.
 - If either button is used to un-hide a previously-hidden signal, the signal is removed from the list of hidden signals (if that filter is active) since it is no longer considered hidden.

There are also some icons used by content displayed in the dialog signal list as shown in the following table.

Table 82 - Add Signals to Waveform Viewer Dialog Icons

Icon	Description
	Signal

Icon	Description
☰	Bus

Note
 The dialog does not show individual bus bits, so the bus bit icon is not used.

Add Source Files Dialogs

There are several categories of **source files** (page 224) that can be added to the ACE project. Under the parent (📁) **Add Project Source Files** action, there are category-specific actions to add files to each category. Each category has its own dialog as shown in the following examples.

After selecting the files to add, click **Open** (in Windows) or **OK** (in Linux) to add them to the project.

Add IP Configuration Files Dialog

The Add IP Configuration Files dialog browses for IP Configuration (.acxip) **source files** (page 224) to add to the selected **project** (page 222).

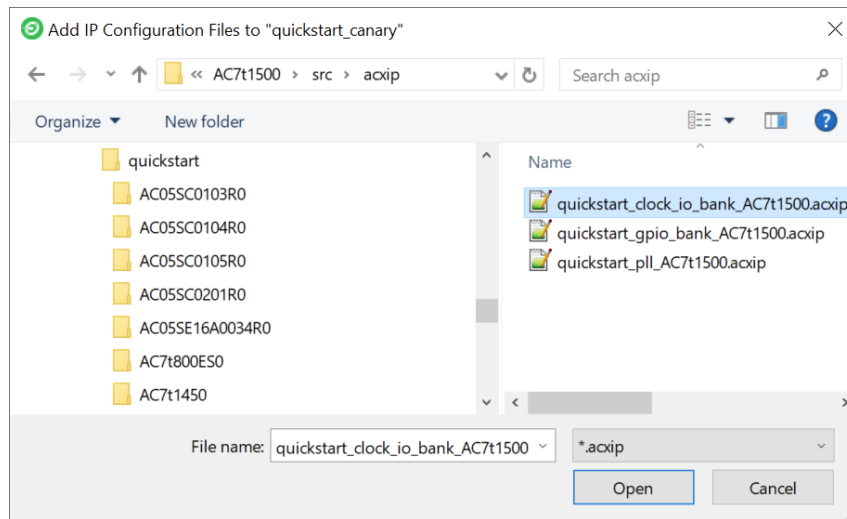


Figure 62 • Add IP Configuration Files Dialog Example

Add RTL Files Dialog

The Add RTL Files dialog browses for Verilog and VHDL RTL (.v, .sv, .vhd, and .vhd\l) **source files** (page 224) to add to the selected **project** (page 222).

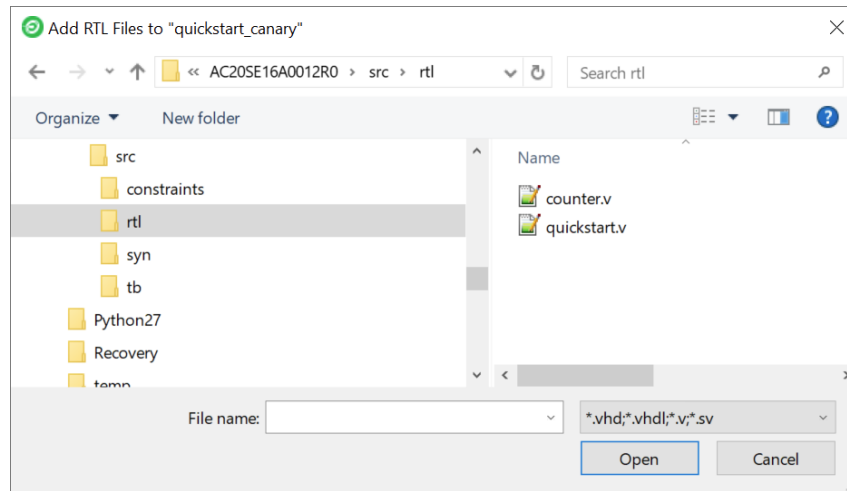


Figure 63 • Add RTL Files Dialog Example

Add Synthesis Constraint Files Dialog

The Add Synthesis Constraint Files dialog browses for Synthesis constraints (. fdc, . sdc, and . scf) **source files** (page 224) to add to the selected **project** (page 222).

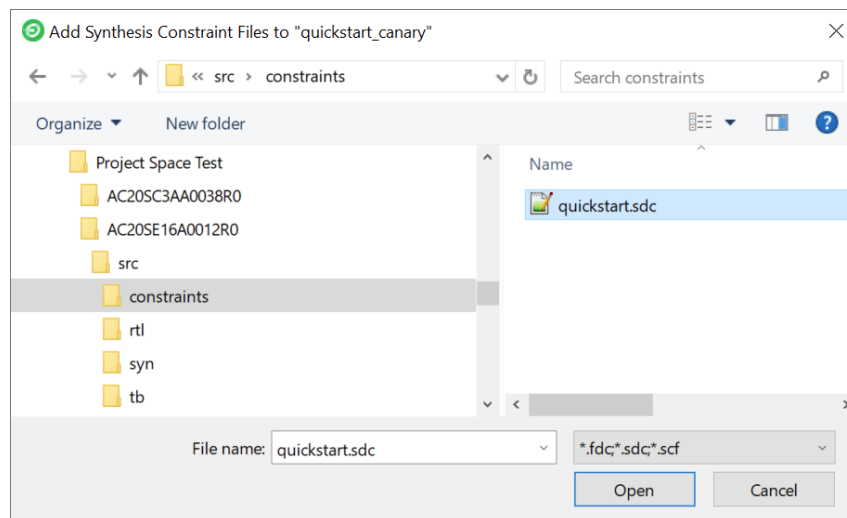


Figure 64 • Add Synthesis Constraint Files Dialog Example

Add Place and Route Constraint Files Dialog

The Add Place and Route Constraint Files dialog browses for Place and Route constraints (. pdc, . sdc, . prt, . scf, . xml, and . hex) **source files** (page 224) to add to the selected **project** (page 222).

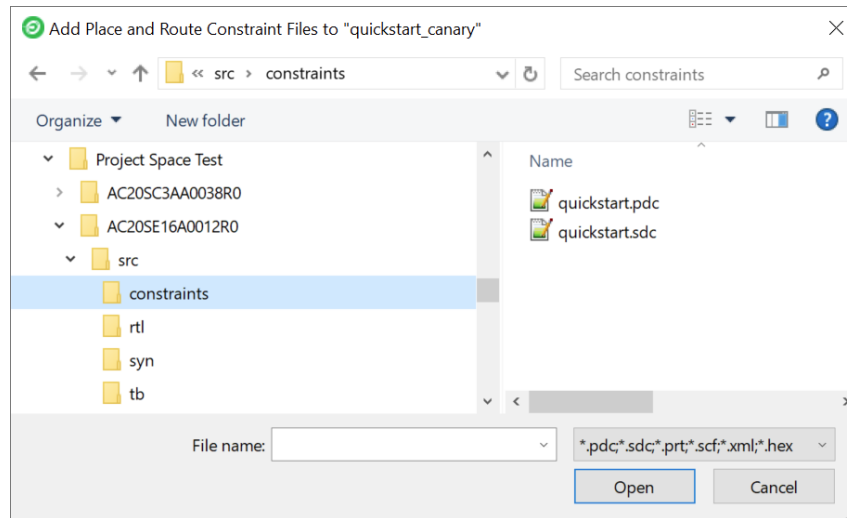


Figure 65 • Add Place and Route Constraint Files Dialog Example

Add Place and Route Netlist Files Dialog

The Add Place and Route Netlist Files dialog browses for source Place and Route Netlist (.vm and .vma) **source files** (page 224) to add to the selected **project** (page 222).

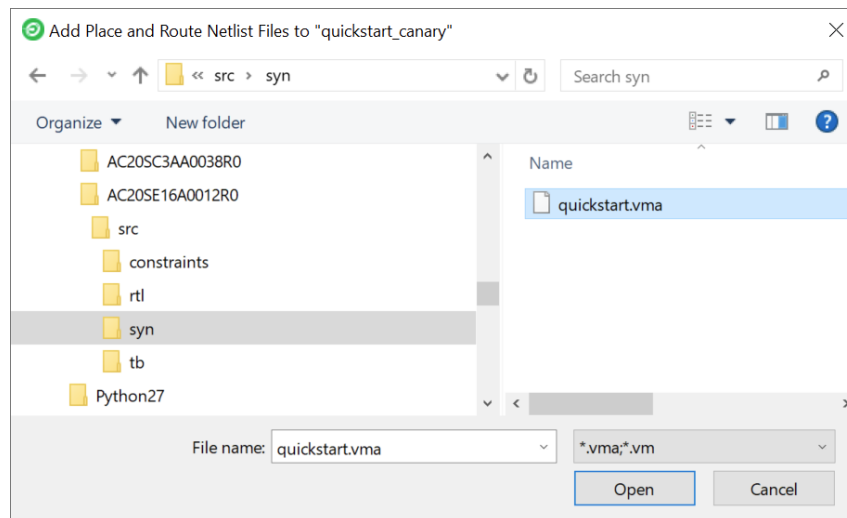


Figure 66 • Add Place and Route Netlist Files Dialog Example

Add Simulation Testbench Files Dialog

The Add Simulation Testbench Files dialog browses for source Simulation Testbench (.v, .sv, .vhd, .vhdl, .f, .txt, .dat, and .mem) **source files** (page 224) to add to the selected **project** (page 222).

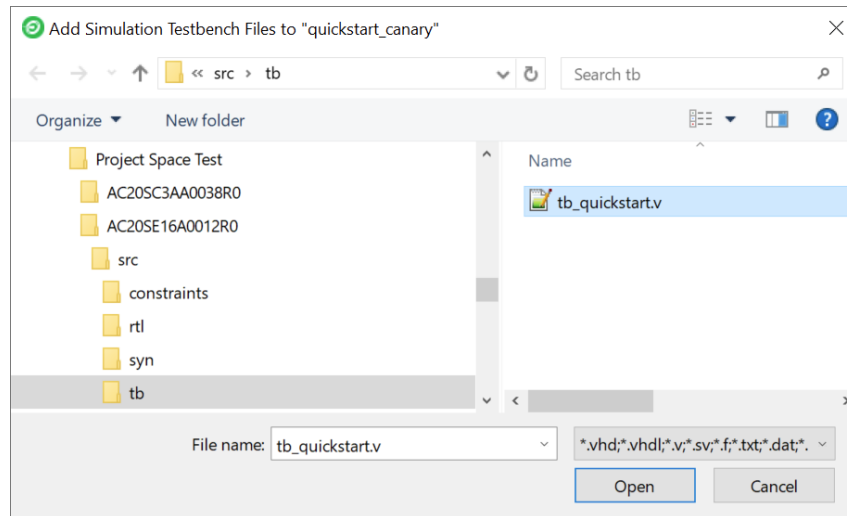


Figure 67 • Add Simulation Testbench Files Dialog Example

See also: [Adding Source Files \(page 297\)](#) and [Adding Configuration Files to a Project \(page 339\)](#).

Assign Bussed Signal Names Dialog

The Assign Bussed Signal Names dialog wizard allows the assigning multiple signal names from the **SnapShot Debugger view (page 137)** "Monitor Channels", "Trigger Channels", or "Stimuli Channels" tables using bus notation. After configuring the bus in the dialog, the bus name and indices are propagated to all the selected signals, changing the signal names appropriately. Monitor channel signal names are then used in the Snapshot sampled output, visible in the **VCD Waveform editor (page 11)**.

Note

This dialog is only useful when creating a Snapshot configuration from scratch. Typically, this dialog is not needed since ACE automatically outputs all signal names from the user design into the names . snapshot file as part of the normal ACE place and route flow.

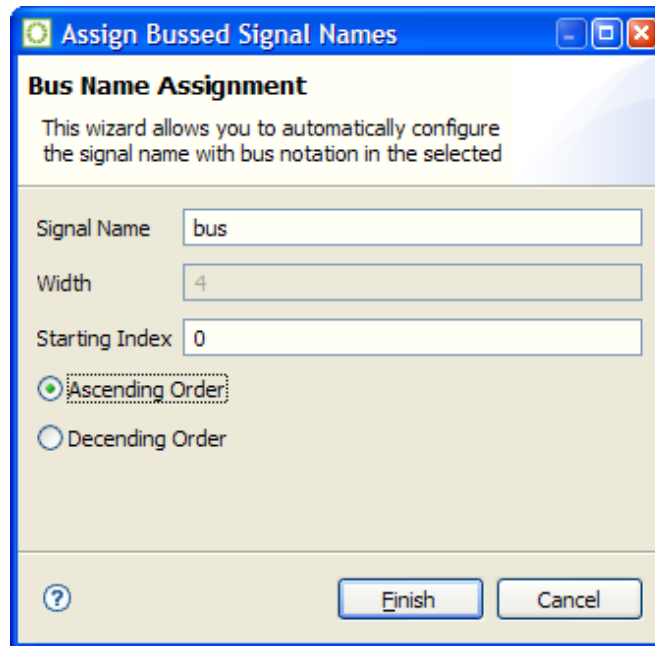


Figure 68 • Assign Bussed Signal Names Dialog Example

Table 83 • Assign Bussed Signal Names Dialog Options

Option	Description
Signal Name	The desired name of the bus.
Width	The width (in bits) of the bus. This value is not editable. It reflects the number of signals which were selected from the table in the Snapshot Debugger view (page 137) .
Starting Index	The desired starting index of the bus, sometimes also called the offset into the bus.
Ascending Order	When selected, the bus indices start at Starting Index and increment Width times.
Descending Order	When selected, the bus indices start at Starting Index and decrement Width times.
Finish	Accepts the specified bus configuration, closes the dialog, and applies the changes to the SnapShot Debugger view (page 137) table.
Cancel	Discards the specified bus configuration and closes the dialog. No changes are applied to the SnapShot Debugger view (page 137) table.

Assign Bussed Values Dialog

The Assign Bussed Values Dialog allows assigning a value to multiple signals from the [SnapShot Debugger view \(page 137\)](#) "Trigger Channels" or "Stimuli Channels" tables as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all of the selected signals in the [SnapShot Debugger View \(page 137\)](#). Two methods are available to launch this dialog allowing bus value assignments:

Method 1

1. Click to select a single row in the [SnapShot Debugger View \(page 137\)](#) table which has a bussed signal name (i.e., `limit_a[2]`).
2. Right click to edit the **Value by Bus**.

This method automatically finds all other bits in the bus with the same signal name (e.g., `limit_a[0]`, `limit_a[1]`, `limit_a[2]`, etc.) and opens the dialog to allow editing of the entire bus of signals.

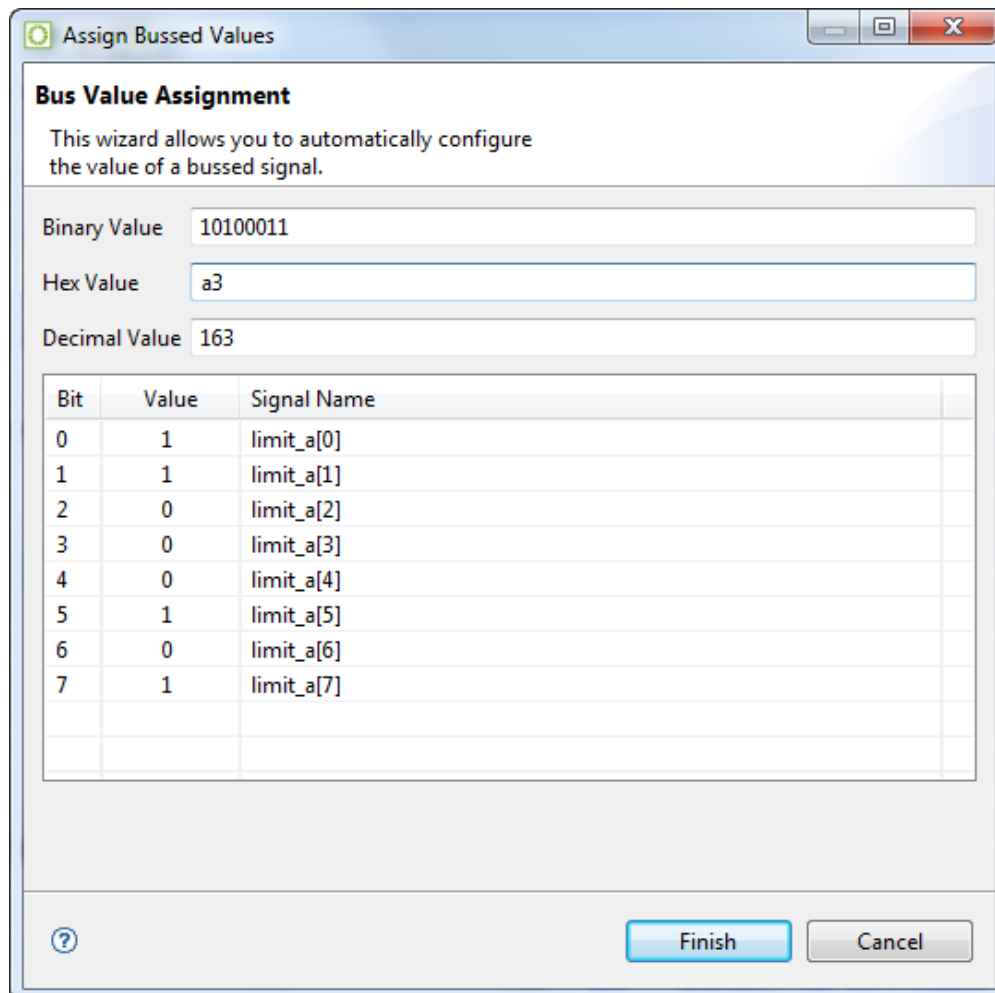


Figure 69 • Value By Bus Assignment Example

Method 2

1. Hold CTRL or SHIFT and click to select multiple rows in the [SnapShot Debugger View \(page 137\)](#) table.
2. Right-click to edit the **Value by Selection**.

This method opens the dialog to allow editing all of the selected signals as a bussed value.

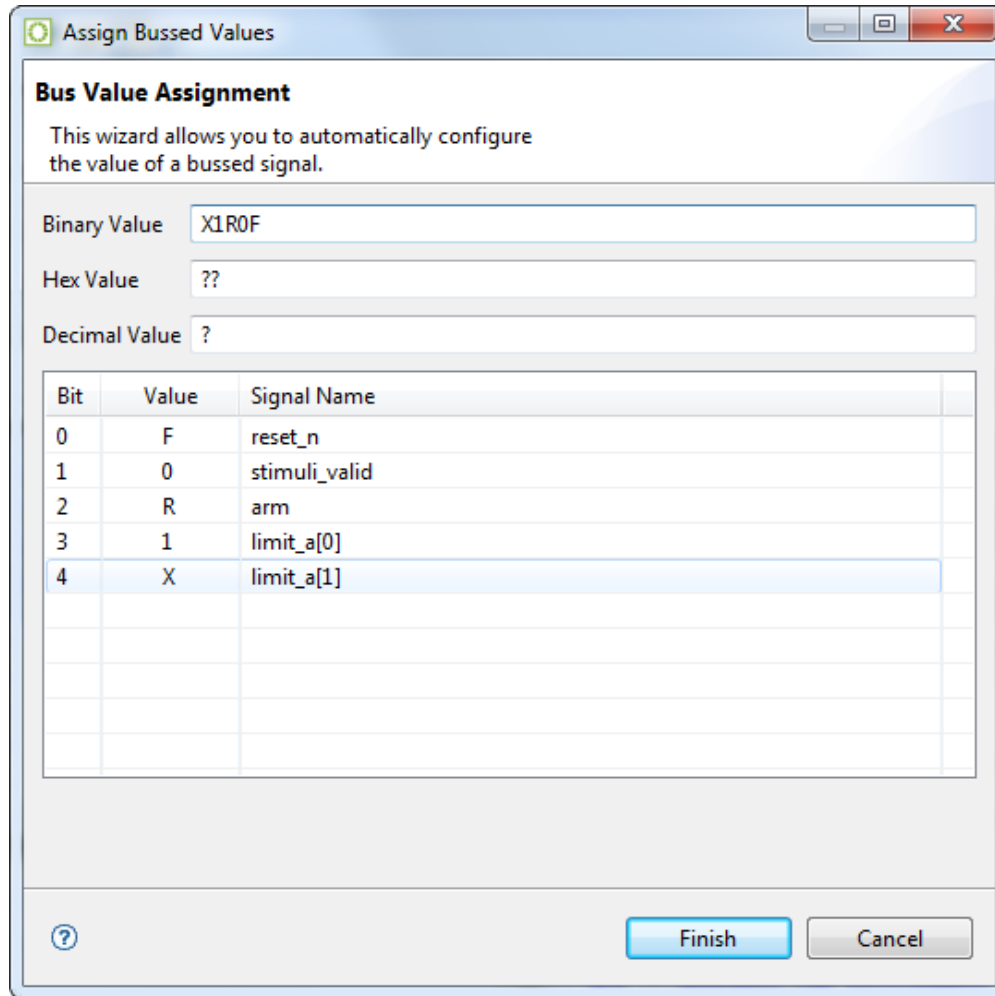


Figure 70 • Bus Value By Selection Assignment Example

See also: [Configuring the Trigger Pattern \(page 367\)](#).

Table 84 - Assign Bussed Values Dialog Options

Option	Description
Binary Value	The desired value for the bus in binary. Valid values for each bit for Trigger Channels are X (don't care), R (rising edge), F (falling edge), 1 (level 1), and 0 (level 0). Valid values for each bit for Stimuli Channels are 1 (level 1), and 0 (level 0). The right-most bit corresponds to bit 0 in the table of signal names, and the left-most bit corresponds to the MSb in the table.
Hex Value	The desired value for the bus in hexadecimal. This field is only capable of representing level (1 or 0) values for each channel. X (don't care), R (rising edge), and F (falling edge) binary values result in a "?" character in this field.
Decimal Value	The desired value for the bus in decimal. This field is only capable of representing level (1 or 0) values for each channel. X (don't care), R (rising edge), and F (falling edge) binary values result in a "?" character in this field.
Bit	The bit offset into the bus value being edited.
Value	The bit value at the bit offset into the bus value being edited.
Signal Name	The signal name at the bit offset into the bus value being edited.
Finish	Accepts the specified bus configuration, closes the dialog, and applies the changes to the corresponding SnapShot Debugger view (page 137) table.
Cancel	Discards the specified bus configuration and closes the dialog. No changes are applied to the corresponding SnapShot Debugger view (page 137) table.

Configure Clock Pre-Routes Dialog

The Configure Clock Pre-Routes dialog allows the creation of new clock pre-route constraints.

This dialog appears after the **Configure Clock Pre-Routes...** action is selected from a context menu in the **Clock Regions view** (page 23), **Clusters view** (page 28), **Partitions view** (page 107), or **Placement Regions view** (page 110).

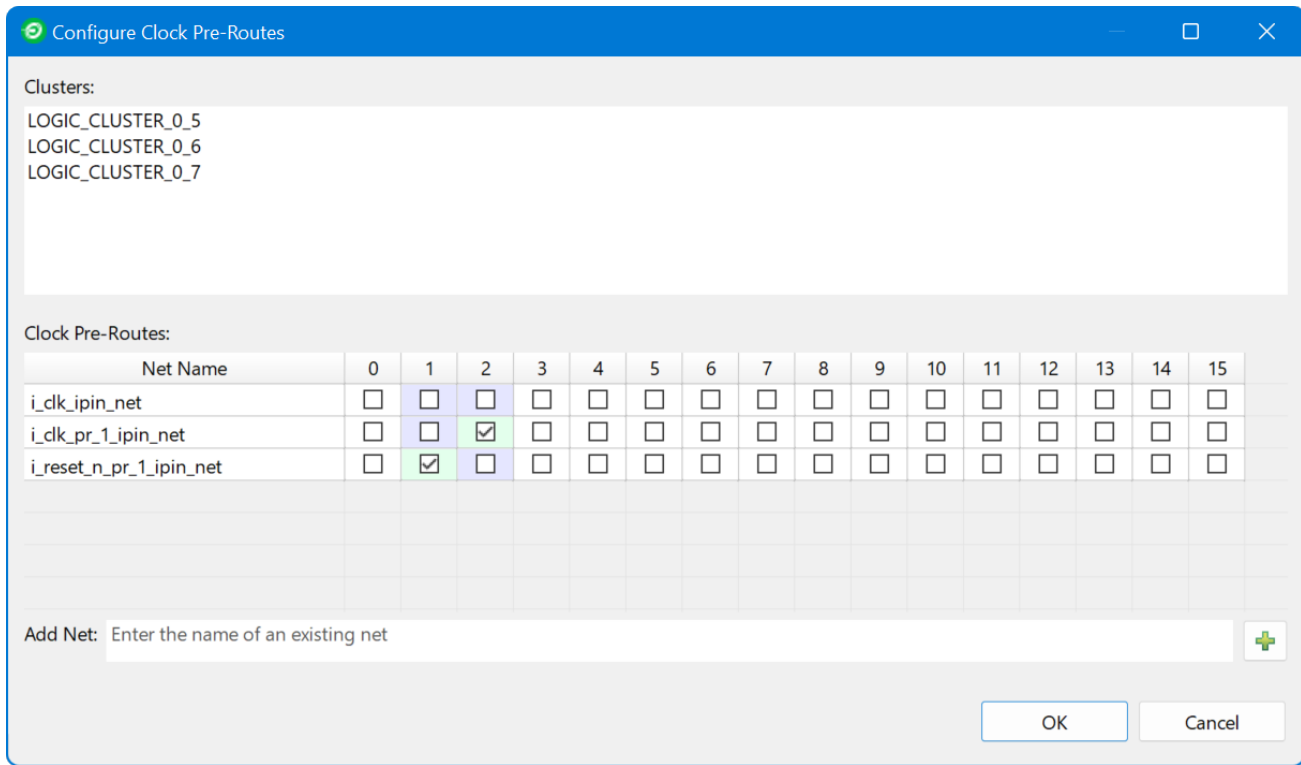


Figure 71 - Configure Clock Pre-Routes Dialog Example

Table 85 - Configure Clock Pre-Routes Dialog Options

Option	Description
Clock Regions/Clusters/ Partitions/Placement Regions	A list of the targets to which pre-route table changes are to be applied.
Net Name	Each row in the table contains the name of a net to be constrained.
Numbered columns	Each numbered column represents a numbered clock track. Checking a cell in the table constrains the given net to the given clock track.
Add Net	Type the name of any net in the design into the text field, press ENTER or click the (+) Add button to add that net to the table.

The table in this dialog uses background colors to indicate the actions performed since the dialog appeared:

- A red background indicates that a check box was removed. Red cells indicate that one or more `remove_clock_preroute` commands are to be issued if the **OK** button is clicked.

- A green background indicates that a check box was added. Green cells indicate that one or more `add_clock_preroute` commands are to be issued if the **OK** button is clicked.
- A purple background indicates that a check box exists somewhere in the given clock track column. Only one net at a time may be associated with any clock track. Checking any cell in the table automatically unchecks all other cells in that column.
- Hovering over a green or red cell shows the `add_clock_preroute` or `remove_clock_preroute` commands to be issued for that row in the table when the **OK** button is clicked.

Note

If some, but not all, targets have a given net associated with a given clock track, *all* targets can be assigned that net/clock track association by unchecking and then re-checking the appropriate check box. When the dialog was first shown, the check box had a purple background. After unchecking and then re-checking, the box has a green background to indicate that new `add_clock_preroute` commands are to be executed to cover any additional targets.

For example, in the following example, the previous example was changed as follows:

- `(i_reset_n_pr_1_ipin_net : 1)` was clicked, unchecking it
- `(i_clk_ipin_net : 2)` was clicked, checking it and automatically unchecking `(i_clk_pr_1_ipin_net : 2)`
- `(i_clk_pr_1_ipin_net : 4,5,6)` were all clicked, checking them

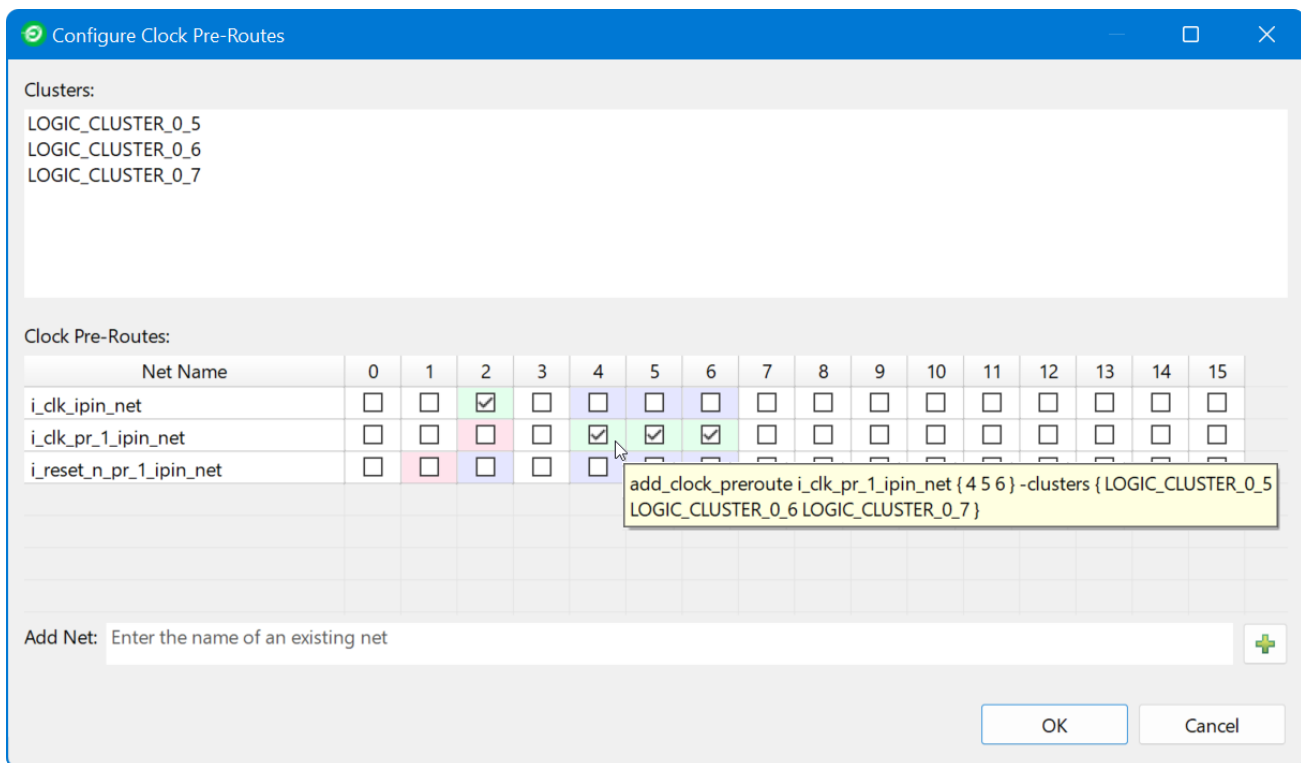


Figure 72 - Configure Clock Pre-Routes Dialog Update Indicators

Create a New Constraints File Dialog

The Create a New Constraints File dialog easily creates a new, empty constraints file and optionally adds it to the currently active project. The dialog is available in all perspectives, and can be accessed by selecting **File** → **New** → **SDC Constraints File...**

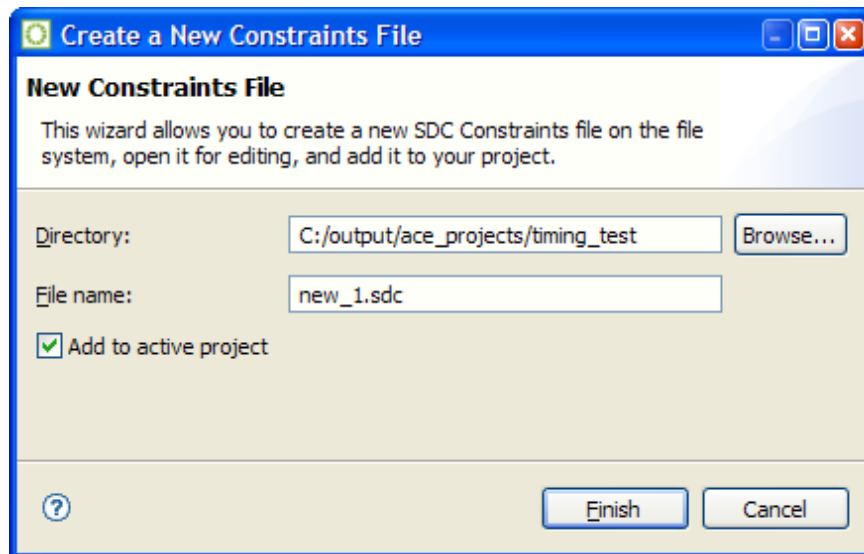


Figure 73 - Create a New Constraints File Dialog Example

The dialog allows typing the file destination Directory, or selecting it graphically using the **Browse...** button. The Directory name provided must already exist. If selected, the **Browse...** button displays a Directory Selection dialog, which also allows creating a new directory and then selecting it.

The File Name must be unique — a file with that name must not already exist in the destination Directory.

If there is currently an active project in ACE, the **Add to active project** checkbox is enabled and checked by default. If there is no project active, the checkbox is disabled and deselected.

When **Finish** is selected, the text file is created, and a **text editor** (page 10) is opened in ACE for the new text file.

Note

This dialog may be used to create PDC files as well as SDC files. Simply use `.pdc` instead of `.sdc` for the file extension.

Create a New Flow Step dialog

The Create a New Flow Step dialog (available from the [Flow view \(page 53\)](#)) allows creating a custom step to add to the flow.

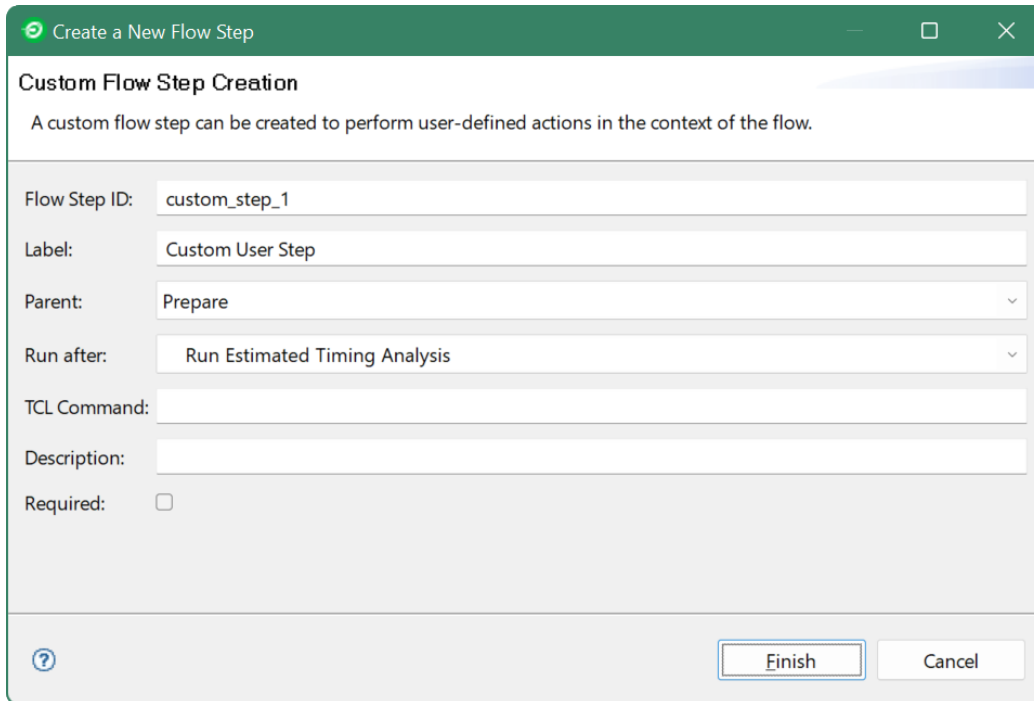


Figure 74 - Create a New Flow Step Dialog Example

Table 86 - Create a New Flow Step Options

Field	Description
Flow Step ID	A unique ID for the new flow step.
Label	The label to appear in ACE.
Parent	The parent flow step. Leave blank to create a new "top level" flow step.
Run after	Name of the flow step after which the new step should be run. Leave blank to create a new "first" flow step.
Tcl Command	The Tcl command to run for the new flow step.
Description	A description of the new step to be displayed in a tool tip when the cursor hovers over the step in the Flow view (page 53) .

Field	Description
Required	Checked if this step must always be run, cleared if the step is optional.

See also: [create_flow_step](#) (page 621), [Custom Flow Steps](#) (page 330), and [ACE_INIT_SCRIPT](#) (page 284) under [Running ACE](#) (page 282)

Create a New Text File Dialog

The Create a New Text File dialog simply allows creating a new text file and opening the file in the ACE text editor in a single action. The dialog is available in all [Perspectives](#) (page 6), and can be selected via **File** → **New** → **Text File**.

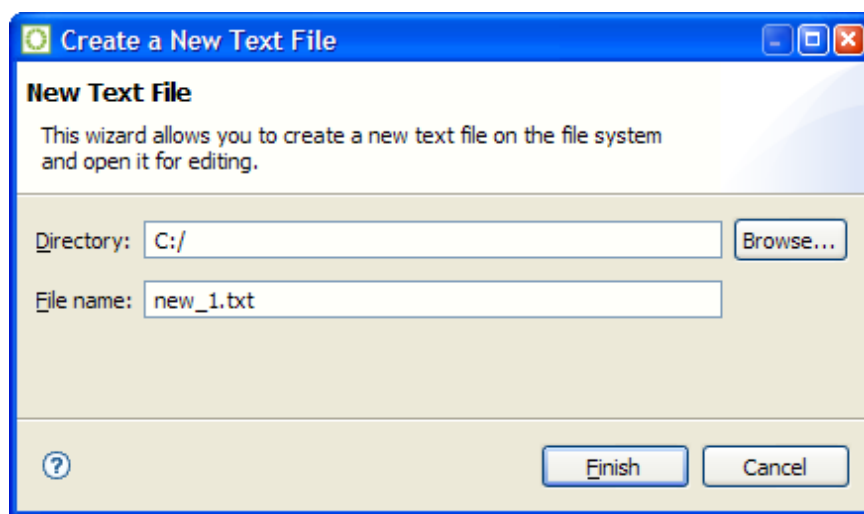


Figure 75 - Create a New Text File Dialog Example

The dialog allows typing the file destination directory, or selecting it graphically using the **Browse...** button. The **Directory** name provided must already exist. If selected, the **Browse...** button displays a Directory Selection dialog, which also allows creating a new directory and then selecting it.

The **File name** must be unique — a file with that name must not already exist in the destination Directory.

When **Finish** is selected, the text file is created, and the ACE [text editor](#) (page 10) is opened for the new text file.

Create a SecureShare Zip File Dialog

The Create a SecureShare ZIP File dialog allows choosing or refining the contents of a SecureShare file, which is then zipped and placed in the chosen directory, ready for delivery to Achronix Technical Support at <https://support.achronix.com/hc/en-us/>. By default, the ZIP file contains all of the important details of the design, enabling Achronix support engineers to examine the design and all output files created during the ACE flow.

Optionally, files may be removed from the default lists of chosen files if it is preferred that those files are not sent to Achronix. In addition, the information may be optionally encrypted in the SecureShare ZIP file.

There are presently three main sections of information in the dialog:

1. The Configuration section
2. The ACE input files section
3. The ACE output files section

More sections are planned to appear in future ACE releases, covering additional support categories, such as synthesis.

This dialog is accessed by selecting **Help** → **Start SecureShare**, or by using the keyboard shortcut **Ctrl + Alt + Shift + S**. When the dialog is shown, it is completely populated by default based upon information gathered from the [Active Project and Implementation](#) (page 229). See also [Using the ACE SecureShare Tool to Create a Support Zip File](#) (page 476).

Configuration

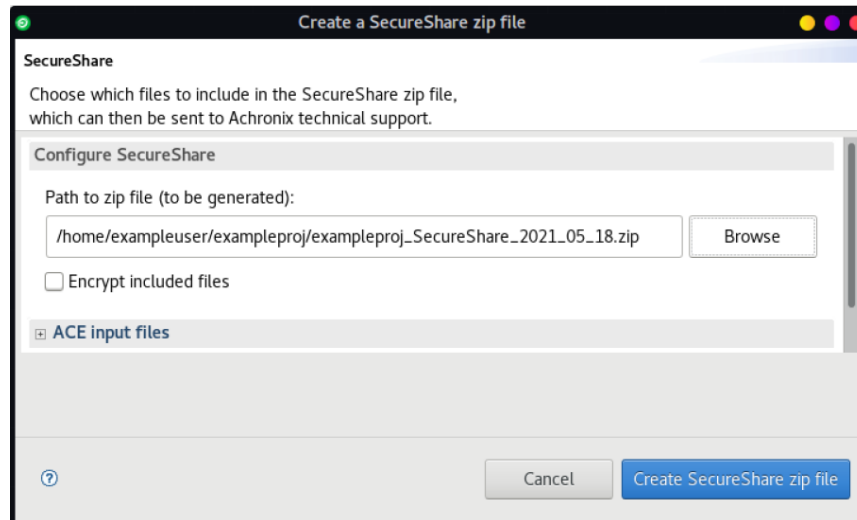


Figure 76 · Create A SecureShare ZIP File Dialog

Table 87 · Create A SecureShare ZIP File Options

Control	Description
Path to ZIP file (to be generated)	<p>This is the SecureShare file to be generated by ACE when the Finish button is clicked.</p> <p>By default, the ZIP file is written to the output subdirectory of the active implementation, with the file name being the design name as a prefix, followed by "SecureShare", then a suffix comprised of the date the file is generated. For example,</p> <pre><active_impl_output_directory>/ <active_project_name>_SecureShare_<yyyy>_<mm>_<dd> .zip.</pre>
Encrypt included files	<p>Defaults to false/unchecked. When checked, the ZIP file (above) is encrypted, and placed in an additional file with the extension <code>.zip.encrypted</code>.</p>

ACE Input Files

Each file category (other than the project file) can hold an arbitrary number of files. Files can be added to (or removed from) any of these categories as desired. Clicking any **Add** button pops up a file selection dialog which defaults to the correct file extensions for filtering and allows choosing one or more files to be added to the associated file list.

Note

Be aware that every file selection dialog also allows setting the file extension filter to `*.*`, allowing any filename to be chosen.

Selecting one or more files from a file list category enables the associated **Remove** button which, when clicked, removes the selected files from the list.



Figure 77 • ACE Input Files Dialog Example

Table 88 • SecureShare Input File Categories

File Category	Description
Project file (*.acxprj)	Defaults to the .acxprj file for the active project.
Netlist source files	Defaults to all the netlist source files (*.v, *.vm, *.vma, *.sv, *.vhdl) found within the active project.
Constraint files ⁽¹⁾	Defaults to all the enabled constraint files (*.hex, *.pdc, *.prt, *.scf, *.sdc, *.xml) for the active implementation within the active project.
*.acxip files	Defaults to all of the .acxip files found within the active project.
Other	Defaults to empty. Allows the addition of any other arbitrary ACE input files which are not covered by the earlier categories.



Table Notes

1. Constraint files within the active project which are not part of the active implementation are not included by default.

ACE Output Files

Similar to input files, each output file category (other than the project file) can hold an arbitrary number of files. Add files to (or remove files from) any of these categories as desired. Clicking any **Add** button pops up a file selection dialog which defaults to the correct file extensions for filtering and allows choosing one or more files to be added to the associated file list.

 **Note**

Be aware that every file selection dialog also allows setting the file extension filter to "*.*", allowing any filename to be chosen.

Selecting one or more files from a file list category enables the associated **Remove** button which, when clicked, removes the selected files from the list.

ACE output files

All files generated by ACE as output, including acxip-generated RTL, logs, reports, acxdb files, generated netlists, bitstreams, and potential debug output.

Include log files

C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\log\impl.log
C:\Users\-example- \achronix\ace_2020_12_03_18_35_01.log
C:\Users\-example- \achronix\rlm_diagnostics.log
C:\Users\-example- \achronix\workspace_8.3\e4_2018_12\metadata\log

Include reports

C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_checker_post_import.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_checker_post_prepare.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_clocks_prepared.html
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_clocks_prepared.txt
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_linefile_prepared.txt
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_pins_prepared.html
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_pins_prepared.txt
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_recondition_post_elaborate.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_recondition_post_import.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_recondition_pre_flatten.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_timing_prepared_C2_0p85V_0C.csv
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_timing_prepared_C2_0p85V_0C.html
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_timing_prepared_C2_0p85V_0C.txt
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_utilization_prepared.html
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\reports\quickstart_utilization_prepared.txt

Include acxdb files

C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_placed.acxdb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_placed_higheffort.acxdb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_placed_loweffort.acxdb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_prepared.acxdb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_prepared_initial.acxdb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\quickstart_routed.acxdb

Include output files

C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\output\fastc_max.lib
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\output\fastc_min.lib
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\output\fv_spec.adb
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\output\slowc_max.lib
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\output\slowc_min.lib

Include debug files

C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_analysis.txt
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_directed_retiming.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_mlp_merge_attribute.pdc
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_optimize_brms.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_optimize_mpls.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_fanout_3.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_fanout_4.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_fanout_6.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_map.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_prepare.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_remap.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_rewire_1.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_recondition_post_rewrite_1.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_rewire_info.tcl
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_verify.log
C:\-example- \projects\rel-v8.3\quickstart_example\impl_1\debug\quickstart_verify_guidefile

Include other

Figure 78 • ACE Output Files Dialog Example

Table 89 - SecureShare Input File Categories

File category	Description
Log files	Defaults to a list of all the files found within the active implementation log subdirectory which might include several multiprocess logs, plus: <ul style="list-style-type: none"> Any detected GUI log files The latest ACE session log file The latest ACE <code>r\lm_diagnostics.log</code> file to help track down any ACE license-related issues.
Reports	Defaults to a list of all files found within the active implementation reports subdirectory, plus the latest Multiprocess Summary Report for the active project.
*.acxdb files	Defaults to a list of all .acxdb files found within the active implementation directory.
Output files	Defaults to a list of all files found within the active implementation output subdirectory.
Debug files	Defaults to a list of all files found within the active implementation .debug subdirectory.
Other	Defaults to empty. Allows adding any other arbitrary ACE-related output files which are not covered by the earlier categories.

Create Implementation Dialog

The Create Implementation dialog is used to create a new implementation in the selected project. After indicating a new name for the implementation and whether to copy option values from the active implementation, click **Finish** to create the implementation in the selected project.

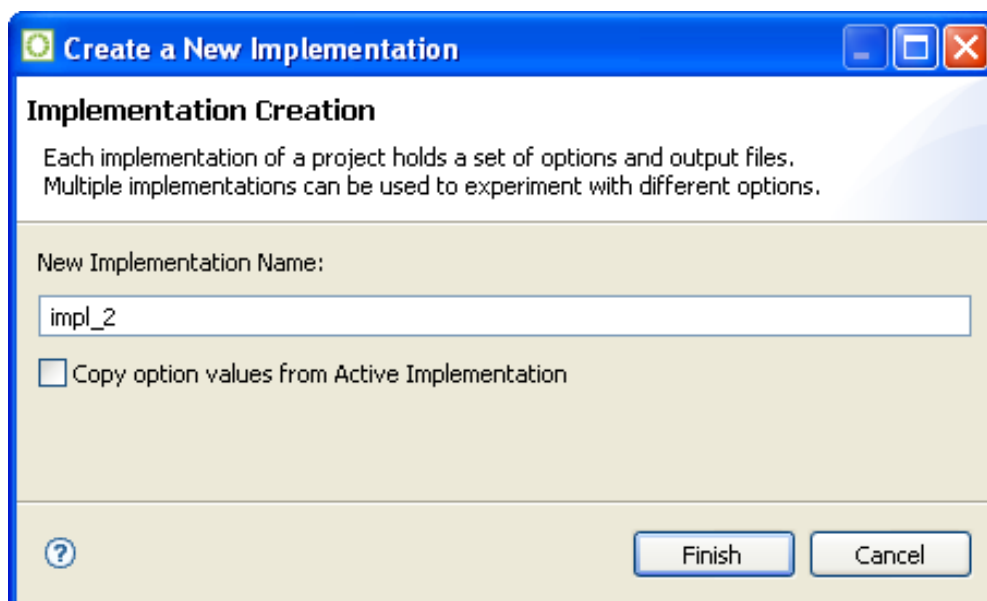
**Figure 79** - Create Implementation Dialog Example

Table 90 · Create Implementation Dialog Fields

Field	Default	Description
New Implementation Name	impl_	The name of the new project implementation to be created. This name must be unique among existing implementations in the selected project. The new name is used to create a new directory under the project directory for the selected project.
Copy Option Values from Active Implementation	Unchecked	If this field is checked, option values from the current Active Implementation are copied into the new implementation.

Create Placement Region Dialog

This wizard dialog appears after a drag-and-drop to define a rectangular area in the **Floorplanner view** (page 43) while the Placement Region tool is active. This dialog allows naming the new Placement Region, and defining its bounds.

See also: [Creating a New Placement Region](#) (page 395).

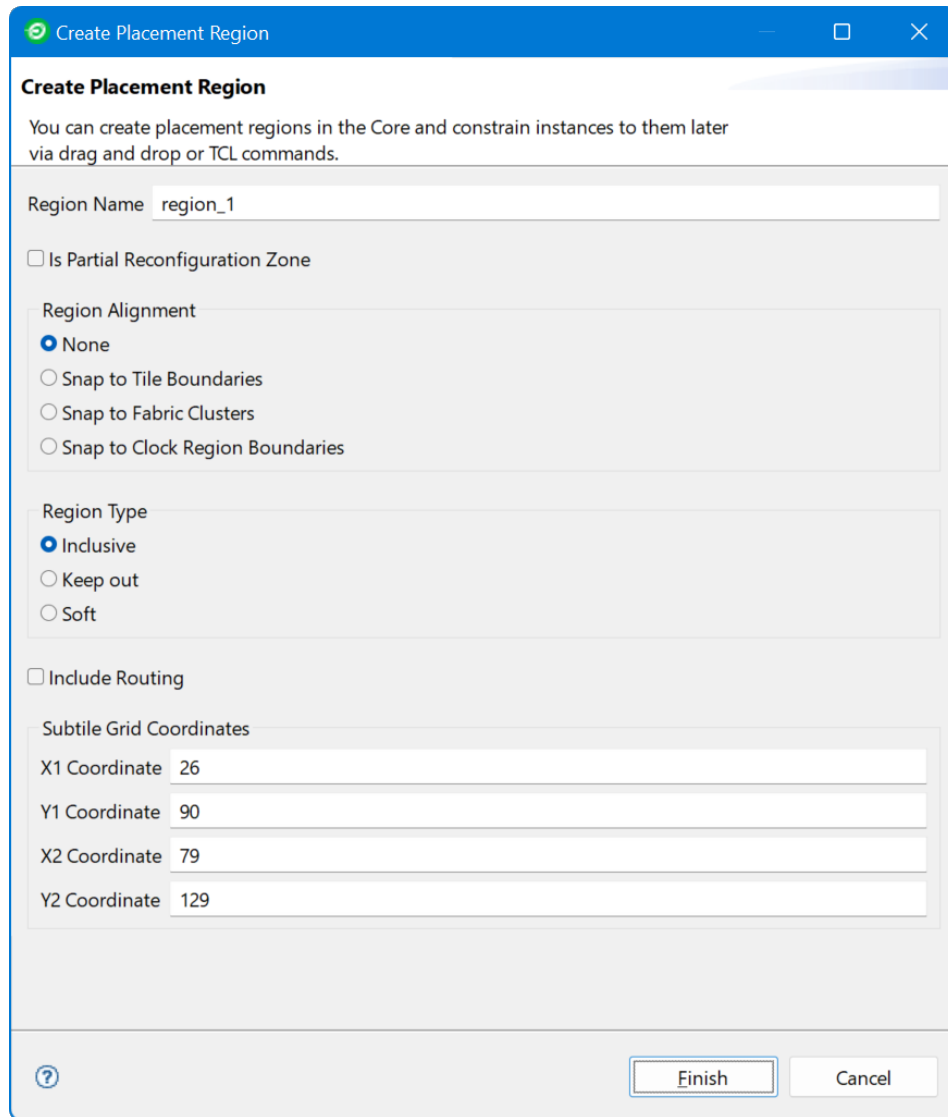


Figure 80 · Create Placement Region Dialog Example

Table 91 · Create Placement Region Dialog Options

Option	Description
Region Name	The name for the new Placement region. ACE pre-populates this field with a default incrementing value.
Include Routing	Treats the region as a routing constraint as well as a placement constraint, keeping all routing wires and instances inside the region boundary box.

Option	Description
Is Partial Reconfiguration Zone	Indicates the region is intended to be used for partial reconfiguration.
Region Alignment	
None	No snapping.
Snap to Tile Boundaries ⁽¹⁾	If selected, ACE creates a Placement region that encompasses all tiles selected within the drag-and-drop rectangle.
Snap to Fabric Clusters ⁽²⁾	If selected, ACE creates a Placement region that encompasses all fabric clusters within the drag-and-drop rectangle.
Snap to Clock Region Boundaries ⁽³⁾	If selected, ACE creates a Placement region that encompasses all Clock regions which contain any of the selected tiles.
Region Type	
Inclusive	Instances added to the region are placed within the region bounding box. Permits instances to be placed inside the region even if they do not belong to the region.
Keep out	Prevents any instances from being placed inside the region. No instances may be added to a Keep Out region.
Soft	Instances added to the region are pulled toward the region center during placement, but instances are permitted to overflow the bounds of the soft region. Soft Placement regions are rendered as ellipses in the Floorplanner View (page 43), and the center of the ellipse acts as a center-of-gravity for placement. Soft regions do not limit the contained number of constrained instances, nor do they have a true count of contained sites of each resource. See Placement Regions and Placement Region Constraints (page 394) for more details.
Subtile Grid Coordinates	
X1 Coordinate	The upper-left X coordinate within the subtile grid, corresponds to the left edge.
Y1 Coordinate	The upper-left Y coordinate within the subtile grid, corresponds to the top edge.
X2 Coordinate	The lower-right X coordinate within the subtile grid, corresponds to the right edge.
Y2 Coordinate	The lower-right Y coordinate within the subtile grid, corresponds to the bottom edge.

Table Notes

1. Since Placement regions can only contain entire sites (no partial sites), the Placement region can potentially grow larger than the outline rectangle.
2. In this mode, since Placement regions can only contain entire fabric clusters (no partial fabric clusters), the Placement region almost certainly grows larger than the outline rectangle.
3. If selected, ACE creates a Placement region that encompasses all Clock regions which contain any of the selected tiles.

Note

When using Subtile grid coordinates, the 0,0 coordinate maps to the upper-left of the Core+Boundary in the Floorplanner view. The exact coordinates of the lower-right corner coordinate limits of the Core+Boundary vary by device.

Create Project Dialog

The Create Project dialog helps create a new project in the Workbench. After indicating a name and location for the project, click **Finish** to create the project.

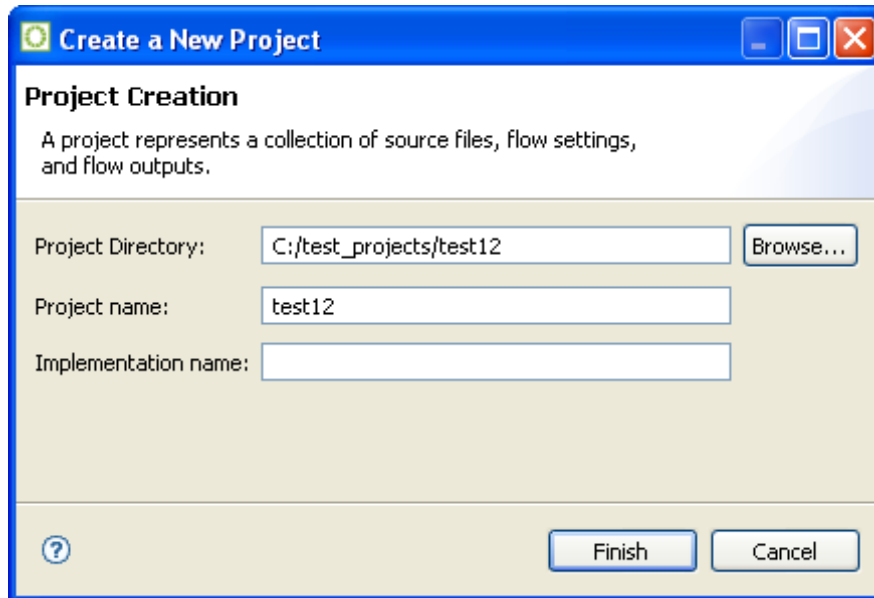


Figure 81 • Create Project Dialog Example

Table 92 • Create Project Dialog Fields

Field	Description
Project Directory	The location in the file system where the project is created. Either type the new location or browse to select a file system location for the new project.
Project Name	The name of the new project to be created. This is the base name of the .acxprj file created in the project directory.
Implementation Name	The name of the new implementation to be created with the project. Leaving this blank causes a name to be chosen automatically.

Generate I/O Ring Design Files Dialog

The Generate I/O Ring Design Files dialog selects the directory into which all the customized I/O Ring design files are output, including:

- Complete package ball pin assignment, power, and utilization reports
- Pin placements PDC, Verilog wrappers, and port lists for the Core user design
- The full I/O Ring bitstream, which is automatically combined with the Core user design bitstream in ACE at the end of the normal place-and-route flow for the Core user design
- Customized I/O Ring simulation files, including Verilog wrappers for the top-level and I/O Ring configuration data

The files generated are based upon the I/O Ring IP Configuration (.acxip) files in the active ACE project created via the active IP Configuration [editor \(page 9\)](#). See also: [Creating an IP Configuration \(page 336\)](#) and [I/O Designer Toolkit Views \(page 60\)](#).

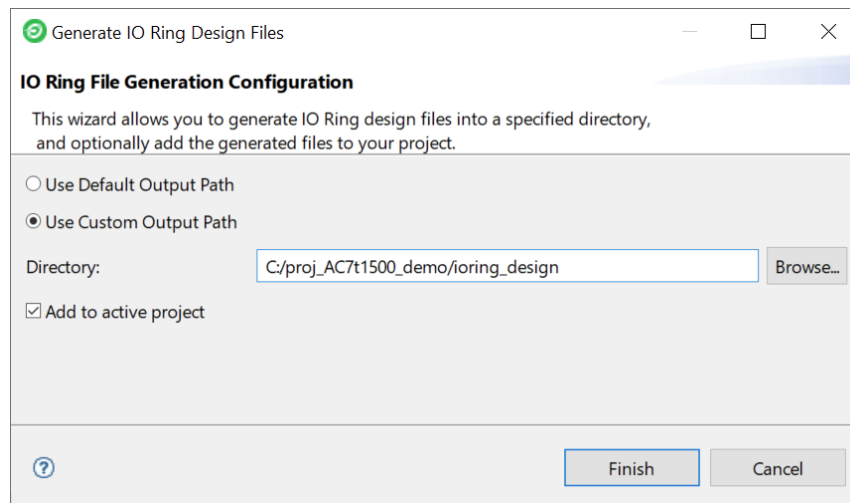


Figure 82 • Generate I/O Ring Design Files Dialog Example

Table 93 • Generate I/O Ring Design Files Dialog Fields

Field	Default	Description
Use Default Output Path	Selected	When selected, the I/O Ring design file output path is configured in your ACE Project Options using the <code>use_default_ioring_design_generation_path</code> and <code>ioring_design_generation_path</code> project options. When selected, the Directory path is grayed out and shows the file path as configured in your ACE project options.

Field	Default	Description
Use Custom Output Path	Not Selected	When selected, the I/O Ring design file output path can be customized by setting the Directory path. The ACE project option settings are ignored in this case.
Directory	<active_ace_project_dir >/ioring_design	Selects the target directory path for the I/O Ring design files when generated.
Add to active project	Selected	When selected, the necessary generated I/O Ring design files (e.g., PDC pin placement constraints, Verilog wrappers, and SDC files for the core user design logic) are automatically added to the active ACE project file.

Generate IP Design Files Dialog

The Generate IP Design Files dialog creates the necessary RTL models, timing constraints and bitstream files for configuring embedded core IP. The files generated are based upon the configuration (.acxip) file created via the active core IP Configuration [editor](#) (page 9).

See also: [Creating an IP Configuration](#) (page 336).

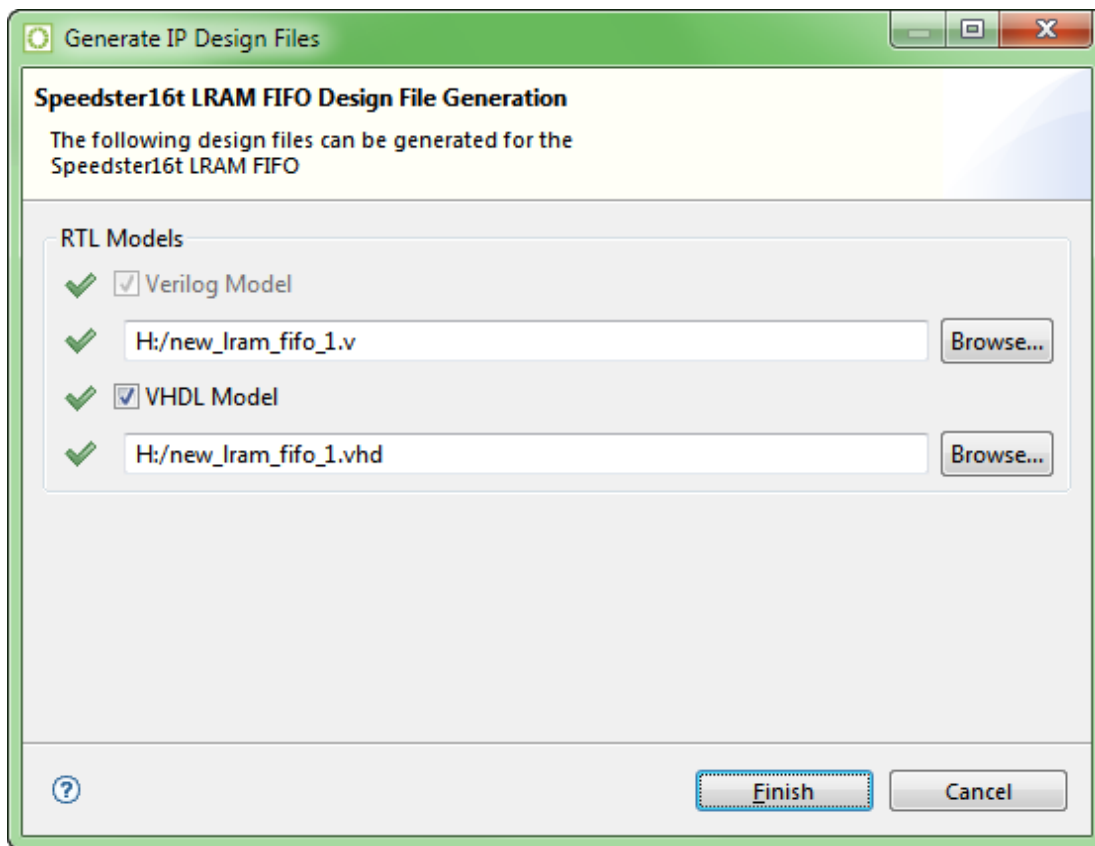


Figure 83 - Generate IP Design Files Dialog Example

Note

Each IP Configuration editor has its own set of output files specific to the type of core IP being configured. For example, some types of IP require PDC or SDC constraints files, while other types of IP do not. The dialog fields presented are dynamic, and change according to the needs of the core IP being generated.

The following table of dialog fields describes the common output files for most types of core IP. The fields presented for each individual core IP are typically a subset of the fields listed.

Table 94 - Generate IP Design Files Dialog Fields

Field	Default	Description
RTL Models		
Verilog Model ⁽¹⁾	Selected	Selects whether to generate a Verilog model for the configuration.
VHDL Model ⁽¹⁾⁽²⁾	Deselected	Selects whether to generate a VHDL model for the configuration.

Field	Default	Description
Timing Constraints		
SDC Constraints ⁽¹⁾	Selected	Selects whether to generate an SDC constraints file for the configuration.
Placement Constraints ⁽¹⁾	Selected	Selects whether to generate a placement constraints file for the configuration.
<p>Table Notes</p> <ol style="list-style-type: none"> 1. The default file path is displayed under the option. An alternate path can be selected via the Browse... button. 2. Because the VHDL RTL model is a simple wrapper around the Verilog RTL model, when using the VHDL RTL model, the Verilog RTL model must also be generated and included in the design. 		

Load Acxdb Dialog

After cancelling the **Flow** (page 234) in the **Flow view** (page 53), the Load Acxdb dialog allows restoring the database of the active implementation to the previously saved state found in the `.acxdb` archive file. This restoration should be performed after canceling a running flow to prevent leaving the database in a partially processed state.

After populating the **Acxdb File** field with the desired archive file, which defaults to the `.acxdb` file with the most recent timestamp, click **Finish** to restore the implementation to the state preserved in the file. To avoid loading a file and to proceed with the database in the current, possibly incomplete flow state, click **Cancel**.

Note

When canceling the flow, this dialog only appears if the run is cancelled after the **Run Prepare** stage is complete.

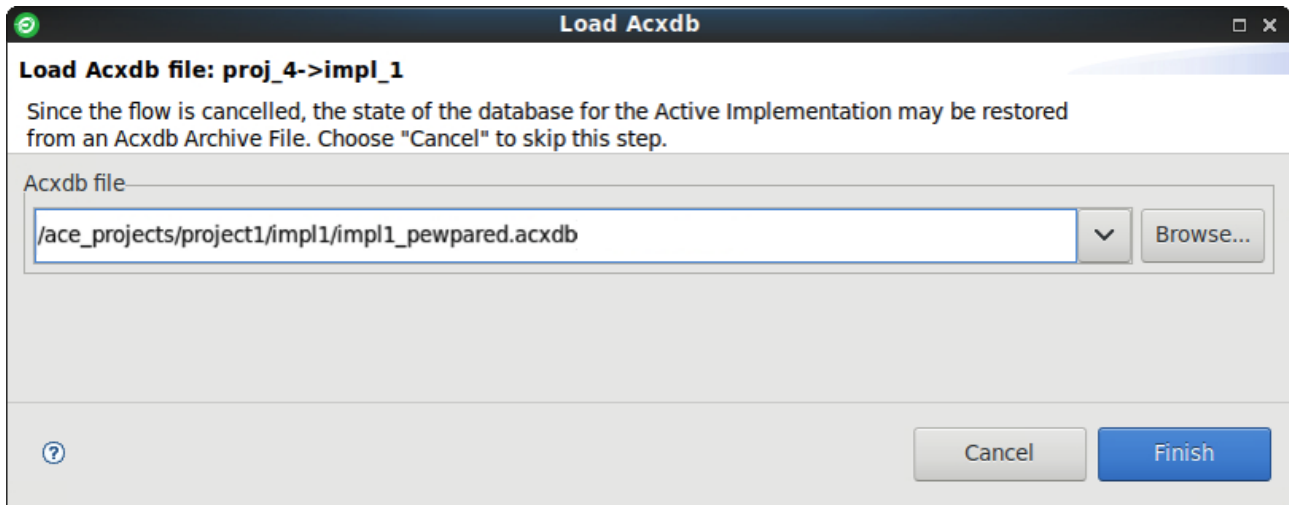


Figure 84 - Load Acxdb Dialog Example

Table 95 - Load Acxdb Dialog Fields

Field	Default	Description
Acxdb File	The latest archive file file found in the active implementation directory.	The file path to the desired .acxdb archive file, which is used to restore the implementation to an earlier saved state. Enter the desired path directly into this field. A drop-down list provides easy access to all archive files in the default directory for the active implementation. The Browse... button allows graphically navigating to alternate files.

Load Project Dialog

The Load Project Dialog allows browsing to find an existing project file to load into the Workbench. After selecting the project file and choosing to activate an implementation, click **Finish** to load the project.

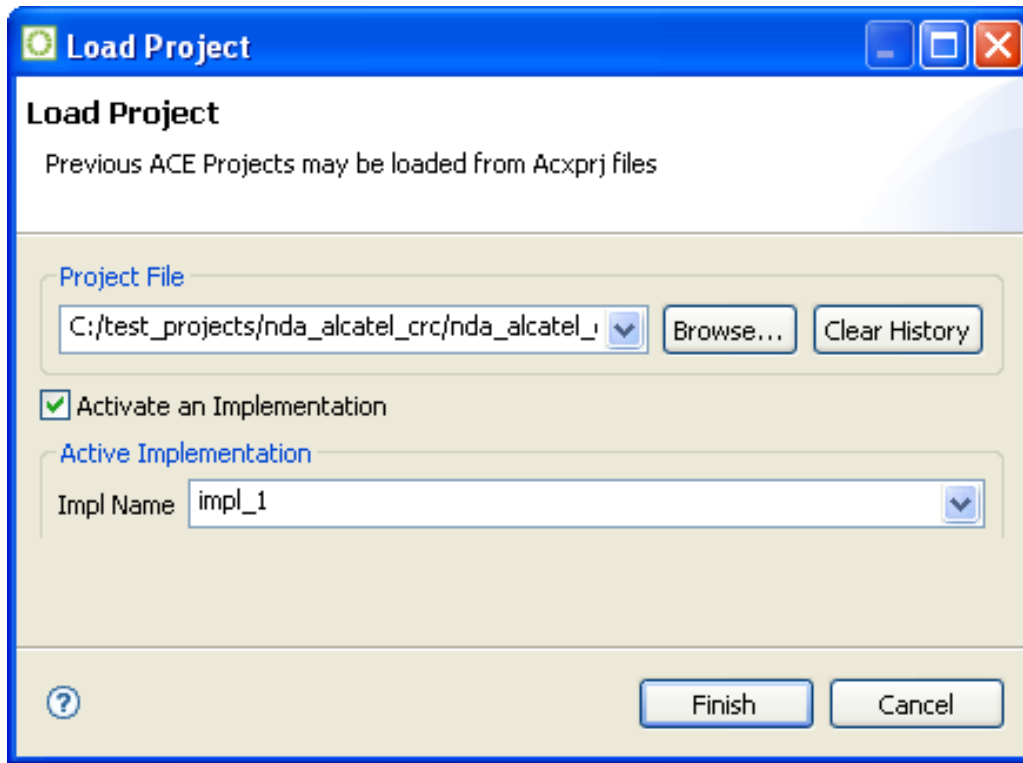


Figure 85 - Load Project Dialog Example

Table 96 - Load Project Dialog Fields

Field	Description
Project File	The path to the ACE Project File (.acxprj). Either type the new location or browse to select a path to the project file. A history of previously loaded projects can be accessed via the drop-down list. This previous project history may be cleared at any time by clicking the Clear History button.
Activate an Implementation	Check to activate an implementation upon loading the project. If another project is already loaded, this field can be cleared to preserve the current active implementation in the ACE session.
Impl Name	The name of the implementation to activate after loading the project. A drop-down list allows selecting from any implementation defined within the specified project file.

New IP Configuration Dialog

The New IP Configuration Dialog helps create a new IP configuration (.acxip) file. After indicating a name and location for the configuration file, click **Finish** to create the file and open the relevant IP Configuration [editor \(page 9\)](#). See also: [Creating a New IP Configuration \(page 336\)](#).

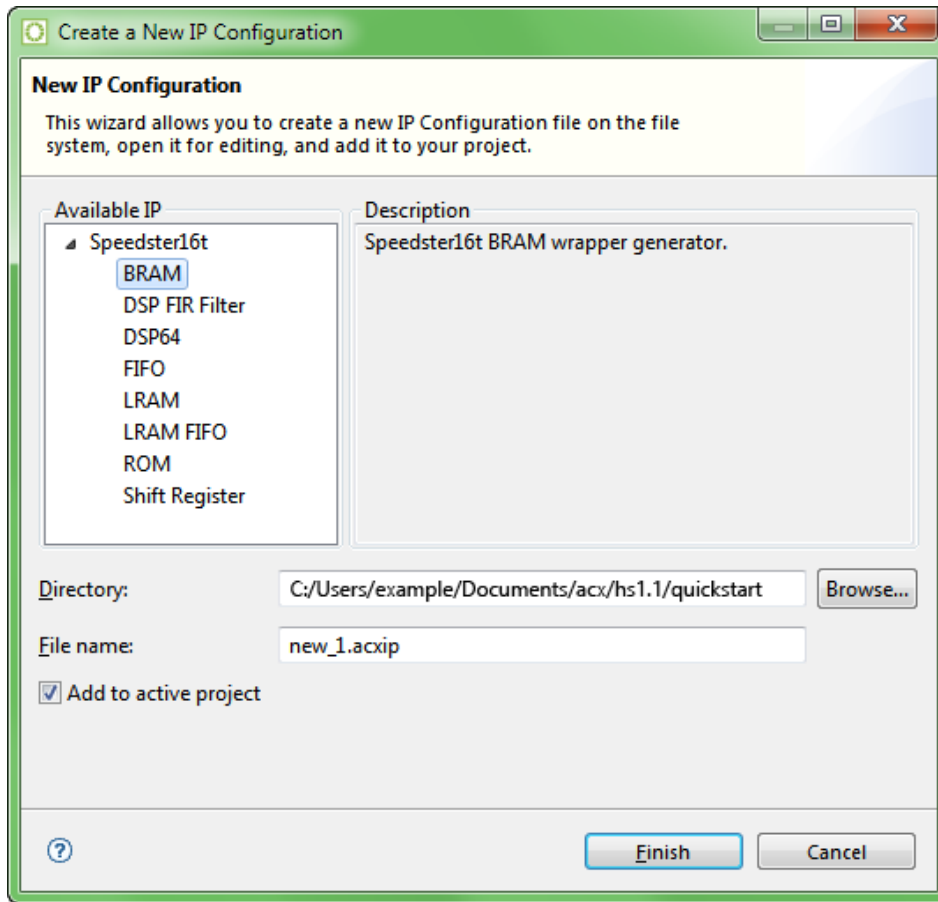


Figure 86 • New IP Configuration Dialog Example

Note

The displayed IP Libraries and IP types are dynamic and change based on which technology libraries and devices are installed and licensed. The screenshots and example descriptions in this section do not necessarily reflect the IP types of the actual device being used.

New IP Configuration Dialog Fields

Field	Default	Description
Available IP	-	Lists the available IP blocks by FPGA family.
Description	-	Provides a description of the IP block.
Directory	Current active project directory	The location in the file system where the configuration file is created. Either type the new location or browse to select a file system location for the new configuration.
File Name ⁽¹⁾	new_.acxip	The name of the new configuration file to be created. The file name (without the .acxip suffix) also becomes the module name in the generated IP.
Add to active project	Enabled	This check box allows the IP configuration file to be added to the current project (this option is only available if a project is active). If unchecked, the IP configuration file is created at the chosen path but not automatically added to any project. Because it is not a member of a project, the new .acxip file is not visible in the Projects view.

Table Notes

- Names that collide with Achronix reserved module names are prohibited.

Plot SerDes Diagram Dialog

The Plot SerDes Diagram dialog appears after selecting **Plot SerDes Diagram** from the option menu when right-clicking a SerDes Lane in the **I/O Layout Diagram view** (page 67). The dialog allows plotting a diagram (Eye plot, Histogram, or Bathtub) for the selected SerDes RX lane using the built-in capture hardware.

Note

This tool requires the free Matlab runtime executable to be installed in addition to ACE with an open JTAG connection in your ACE session to access the SerDes logic on the chip.

See also: [Plotting SerDes Receiver Diagrams Using JTAG](#) (page 482).

Plot a Serdes Rx Diagram using JTAG

Serdes Diagram Plot Configuration

Using an existing open JTAG connection, capture and plot internal Rx diagrams from the chosen SerDes lane

IP Block Name:

JTAG Device ID:

Number of Samples:

Capture Point:

Diagram Type:

Figure Name:

Figure Number:

Capture Data File:

Figure 87 • Plot SerDes Diagram Dialog Example

Table 97 • Plot SerDes Diagram Dialog Fields

Option	Description
IP Block Name	The name of the SerDes IP block selected from the right-click option menu in the I/O Layout Diagram view.
JTAG Device ID	The ID of the connected JTAG device (an FTDI USB chip or a Bitporter2 pod). This ID can be specified with a hard-coded string such as "AC12345" or with a Tcl variable.
Number of Samples	The number of samples to capture in the SerDes RX capture hardware. The default for an eye plot is 500 samples. Over JTAG, this requires approximately 10 minutes to read back and plot. More samples produces a more accurate diagram, while less samples can reduce runtime.
Capture Point	Selects the internal SerDes block from which to capture data.
Diagram Type	Select an eye plot, histogram, or bathtub plot.

Option	Description
Figure Name	Specifies a label to appear on the generated diagram image.
Figure Number	Specifies a figure number label to appear on the generated diagram image.
Capture Data File	Enter the file path to the diagram capture data retrieved from the SerDes hardware via JTAG. Click Browse... to graphically select the file. This file can be used to plot multiple diagram types from the same data.

MATLAB SerDes Eye Plot Y-axis Unit Formula

ACE 9.2+ uses MATLAB Runtime 2021b (9.11) for the SerDes eye plot. With this update, the Y-axis units are updated from voltage (V) to the more accurate digital codes (codes). Full scale is +/-128. To convert from codes to voltage, assume the FFE and DFE gain is 1 and the ADC is 8-bit with a full scale of 0.55V. Thus, the conversion from codes to voltage is:

$$\text{estimate_voltage} = \text{codes} \times (0.55 / 256)$$

Restore Implementation Dialog

The Restore Implementation dialog restores the database state of the active implementation from a .acxdb archive file. After indicating the path to the archive file from which to restore the implementation, click **Finish** to restore the active implementation.

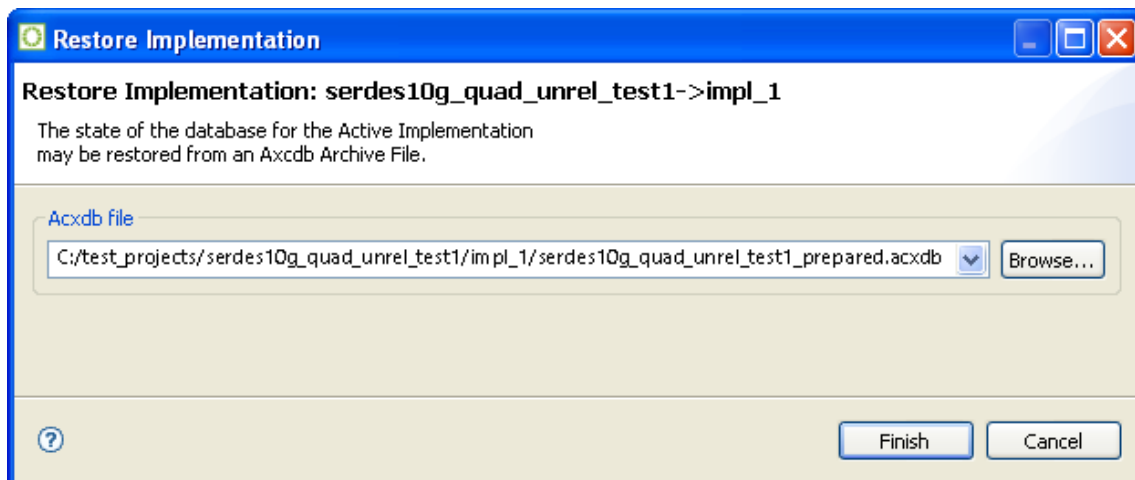


Figure 88 - Restore Implementation Dialog Example

Table 98 · Restore Implementation Dialog Fields

Field	Default	Description
Acxdb File	The newest .acxdb file in the directory	The path to the .acxdb archive file from which to restore the implementation. A drop-down list provides easy access to all other archive files in the directory.

Save Implementation Dialog

The Save Implementation dialog saves the database state of the active implementation to an .acxdb archive file. After indicating the path to the file in which to save the implementation, and whether to include the log file, click **Finish** to save the active implementation.

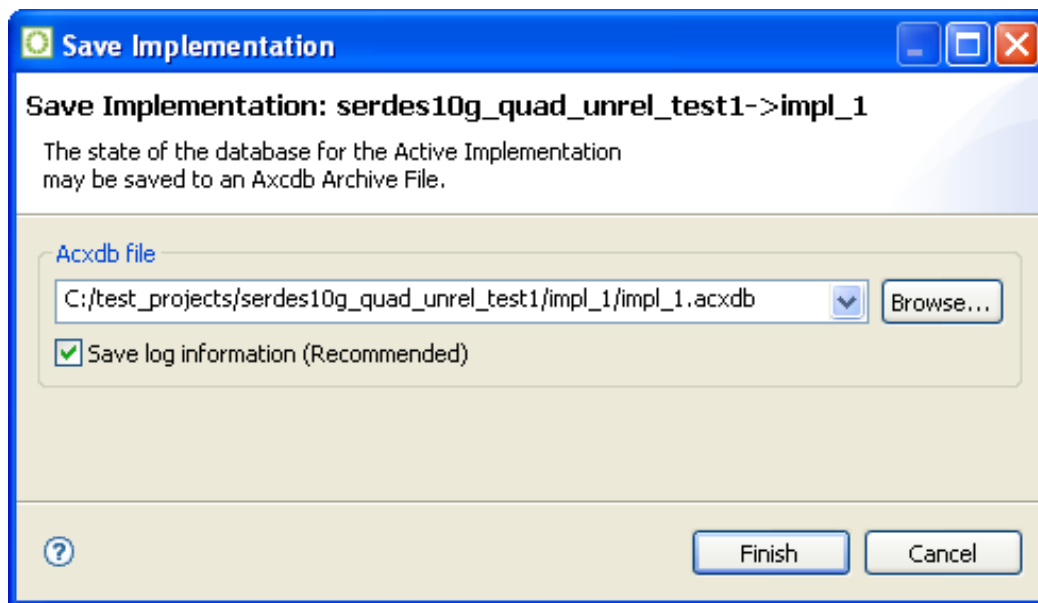


Figure 89 · Save Implementation Dialog Example

Note

Implementations may only be saved after the Run Prepare flow step has been completed. Prior to that, there is no meaningful content in the database to save.

Table 99 • Save Implementation Dialog Fields

Field	Default	Description
Acxdb File	.acxdb	The path to .acxdb archive file in which the implementation is saved.
Save Log Information	On	If this field is checked, the log file for the current active implementation is included in the archive file.

Save Placement Dialog

The Save Placement dialog saves the current placement to pre-placement constraints file(s). After selecting the appropriate options one page one, click **Next** and continue on page two. Click **Finish** to save the placement. This dialog can be triggered from the **Floorplanner view** (page 43), the **Search view** (page 129), and the **Selection view** (page 133).

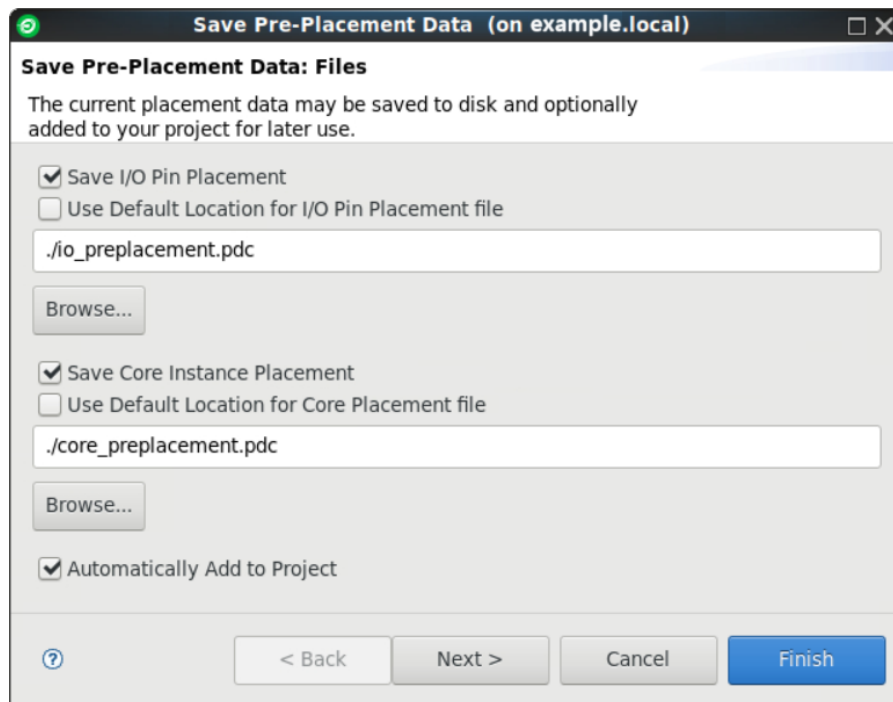


Figure 90 • Save Placement Page One Dialog Example

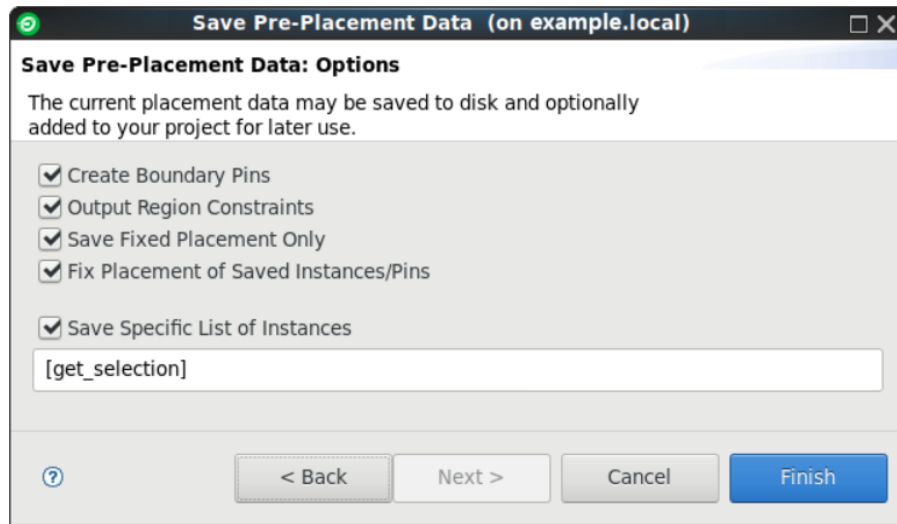


Figure 91 - Save Placement Page Two Dialog Example

Table 100 - Save Placement Dialog Fields

Field	Default	Description
Save I/O Pin Placement	Enabled	Indicates whether placement of instances in the Boundary ring should be saved.
Use Default Location for I/O Pin Placement File	Enabled	Specifies using the default I/O pin placement file path.
(Text Box)	(blank)	Specify the I/O pin placement file path if the above option is disabled.
Save Core Instance Placement	Enabled	Indicates whether placement of instances in the core fabric should be saved.
Use Default Location for Core Placement File	Enabled	Specifies using the default core placement file path.
(Text Box)	(blank)	Specify the core placement file path if the above option is disabled.
Automatically Add to Project	Enabled	Controls whether the output placement files are automatically added to the current project as pre-placement constraints files.

Field	Default	Description
Create Boundary Pins	Enabled	Indicates whether the placement of boundary pin instances should be saved.
Output Region Constraints	Enabled	Controls whether Placement regions and Placement region constraints (page 394) are exported to the PDC file.
Save Fixed Placement Only ⁽¹⁾	Enabled	Controls whether only the current fixed-placement constraints are saved to the output files. If set, all other placement information (such as that generated by the placer) is ignored.
Fix Placement of Saved Instances/ Pins	Enabled	Forces all saved placements to be "fixed" placement, to allow lock down of all placed instances.
Save Specific List of Instances	Contextual	Enabled by default when created from the Search view (page 129) or Selection view (page 133). Disabled by default when created from the Floorplanner view (page 43).
(Text Box)	(blank)	Enter a Tcl list of instance names, or a Tcl statement that returns a list of instance names. This list is then used (instead of all instances in the design) in combination with the other dialog settings when choosing what to save.

Table Notes

1. It is recommended to always use this option when saving pre-placement constraints.

Save Placement Regions Dialog

The Save Placement Regions Dialog appears after selecting the **Save Placement Regions** action in the Placement Regions view. This dialog allows saving placement region definitions, including the instance constraints for those placement regions.

See also: [Saving Placement Region Constraints](#) (page 400).

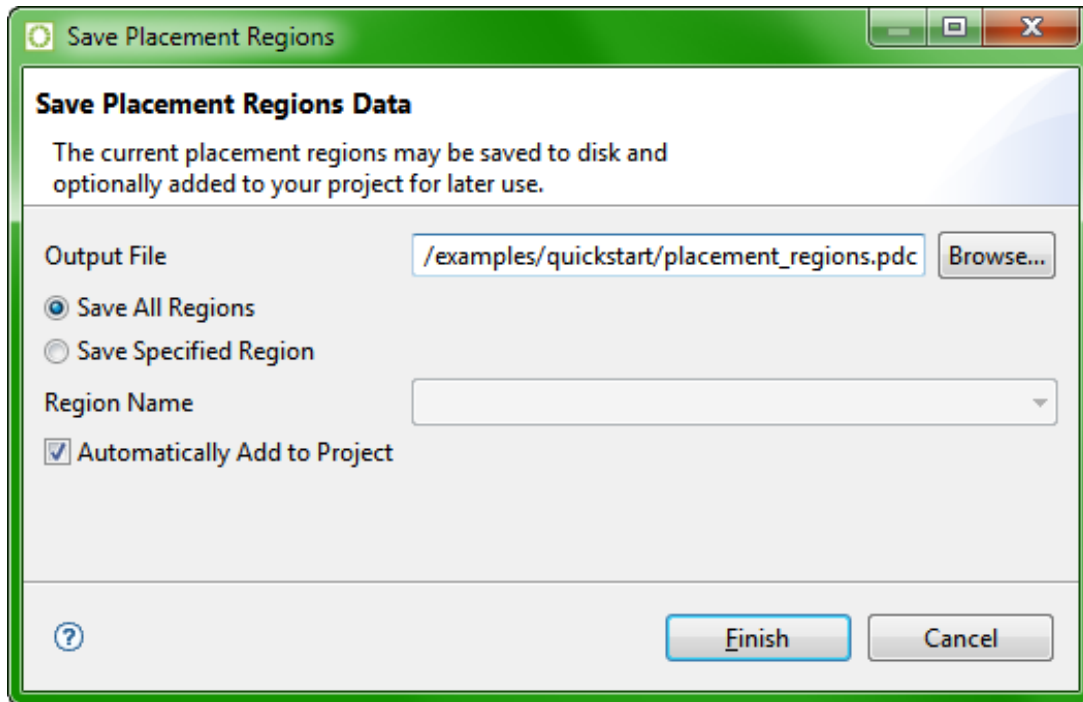


Figure 92 · Save Placement Regions Dialog Example

Table 101 · Save Placement Regions Dialog Options

Option	Description
Output File	The full path to the .pdc file containing the region definitions and constraints.
Save all Regions	Saves the data for all regions listed in the Placement Regions view.
Save Specified Region	Saves the data for a single region specified by Region Name .
Region Name	The name of the Placement Region to be saved. This field is disabled unless Save Specified Region is selected.
Automatically Add to Project	When selected, if the Output File is not already a member of the constraints for the active project, the file is added to the project as a constraint file.

Search Filter Builder Dialog

The Search Filter Builder dialog allows building simple or compound search filters for the [Search view \(page 129\)](#) (the dialog is accessed from the ... button in the **Filters** row of the Search view).

Search filters find objects in the design based upon properties other than object name. Simple filters may be combined into a compound filter by joining them with Boolean operators. See [Filter Properties \(page 264\)](#) for a table of the available filter properties with descriptions.

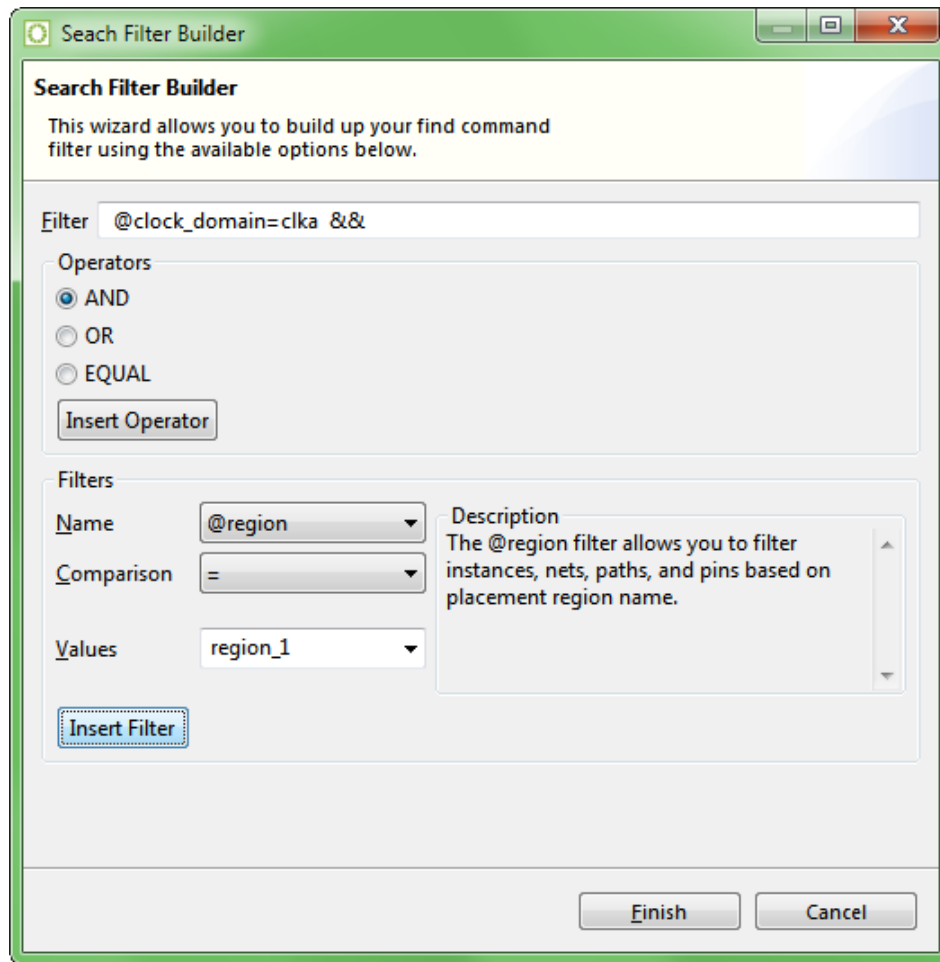


Figure 93 - Search Filter Builder Dialog Example

Table 102 - Search Filter Builder Dialog Options

Option	Description
Filter	The filter string itself. Type directly into this field, or populate this field by using the Insert Operator and Insert Filter buttons.

Option	Description
Operators	
AND	Select this radio button to join two filters into a compound filter where both sub-filters must be true.
OR	Select this radio button to join two filters into a compound filter where either sub-filter is true.
EQUAL	Select this radio button to join two filters into a compound filter where the Boolean value of both sub-filters is identical.
Insert Operator	Click this button to insert the selected Boolean operator into the Filter field at the current cursor position within that field.
Filters	
Name	A combo box showing all choices of supported filter parameter names. The value of this field affects the content of the Description areas, as well as the possible values for the Comparison , and Values options. For a list of all available types of filters, see Filter Properties (page 264) .
Description	Provides a textual description of the current filter parameter selected in the Name field. This field may also provide hints or details on how the Comparison or Values fields may be populated.
Comparison	Selects from the set of possible comparisons relevant to the selected filter parameter Name . The contents of this combo box change according to the current Name selection.
Values	Enter the desired value(s) to compare against. For some filter parameter Name values, the combo box may contain possible values. The contents of this combo box change according to the current Name selection.
Insert Filter	Click this button to insert the selected filter specification into the Filter field at the current cursor position within that field.
Finish	Click this button to close the dialog, copying the current value of the Filter text field into the Filters field of the Search view.
Cancel	Click this button to close the dialog without changing the current value of the Filters field of the Search view.

Toolbars

There are three kinds of toolbars in the **Workbench** (page 6):

- **Main** – sometimes called the Workbench toolbar, is displayed at the top of the Workbench window directly beneath the menu bar. The contents of this toolbar change based on the active perspective. Items in the toolbar might be enabled or disabled based on the state of either the active **view** (page 16) or **editor** (page 9). Sections of the main toolbar can be rearranged using the mouse.
- **View** – individual view toolbars, which appear in the title bar of a **view** (page 16). Actions in a view toolbar apply only to the view in which they appear. Some view toolbars include a Menu button, shown as an inverted triangle, which contains additional actions for that view.
- **Trim** – minimizing a view/editor tab stack also produces a toolbar in the trim at the outer edge of the workbench window (a Trim Stack). This bar contains an icon for each of the views in the stack and/or a single icon for each stack of editors. Clicking one of these icons results in the view/editor being displayed as an overlay onto the workbench window.

In all cases, when the cursor is positioned over a toolbar button, a tooltip describing its function appears.

Preferences

The Preferences dialog sets user preferences. Click the desired page title to view its content and make changes.

The dialog pages may be searched using the filter function. To filter by matching the page title, simply type the name of the page being sought in the "type filter text" box, and the available pages are listed below the text box. The filter also searches on keywords such as "appearance" and "text".

The history controls, (←) Previous and (→) Next, allow navigation through previously viewed pages. To step back or forward several pages at a time, click the desired control button to see a drop-down list of the most recently viewed preference pages.

The Preferences dialog can be found from the main workbench menu by selecting **Window** → **Preferences...**

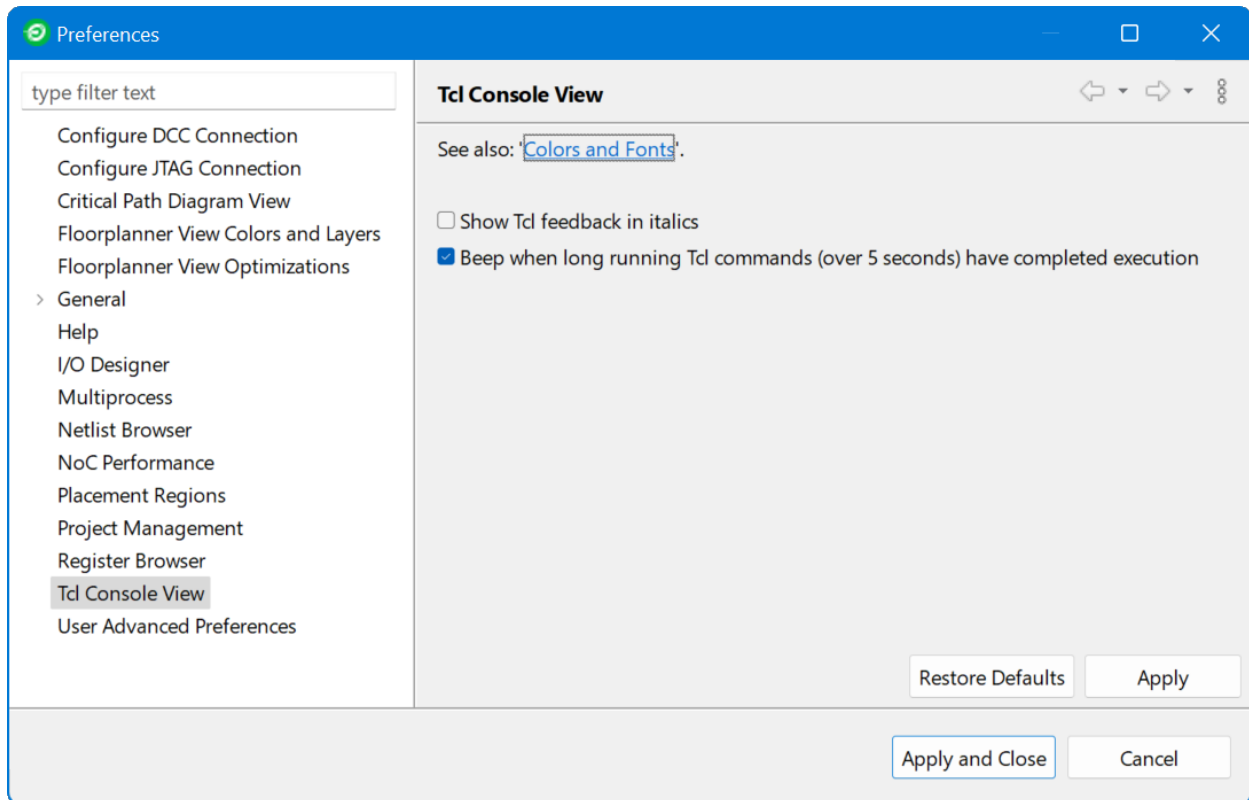


Figure 94 · Preferences Dialog Example

Configure DCC Connection Preference Page

The Configure DCC Connection Preference page configures the [Preferences](#) (page 188) for the Demo Command and Control (DCC) connection, as used with the [HW Demo](#) (page 58) view.

See also: [Configuring the DCC Connection](#) (page 358) and [Running the HW Demo](#) (page 401).

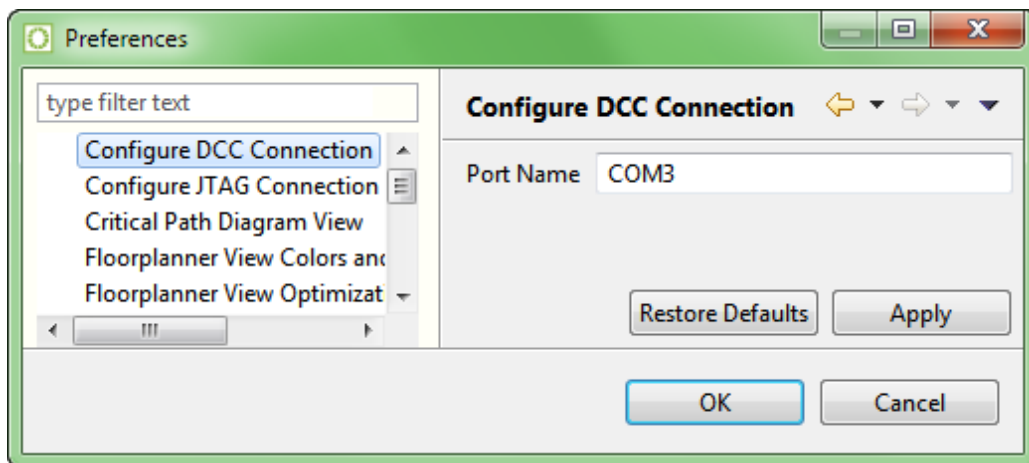


Figure 95 · Configure DCC Connection Preference Page Example

Table 103 · Configure DCC Connection Preference Page Options

Option	Description
Port Name	Enter the serial port name used for the DCC connection. For further information about determining which serial port should be used, please see Configuring the DCC Connection (page 358) .

Configure JTAG Connection Preference Page

The Configure JTAG Connection preference page configures which Bitporter2 or FTDI FT2232H or FT4232H device is used to communicate via JTAG to the desired device under test (the test board). The page also specifies where the Achronix device is found in the JTAG scan chain, which might potentially contain multiple Achronix and non-Achronix devices.

Warning!

Bitporter2 JTAG pods and test boards can be damaged if connected improperly. Always be aware of ESD safety concerns. Before attempting to use a USB JTAG connection, with or without a Bitporter2 pod, please consult the *JTAG Configuration User Guide (UG004)* at <https://www.achronix.com/documentation/jtag-configuration-user-guide-ug004>.

Multiple views use these configuration preferences, including the [Download view \(page 40\)](#), the [Snapshot Debugger view \(page 137\)](#), the [Register Browser view \(page 127\)](#), and the [HW Demo view \(page 58\)](#). Specialized functionality for some IP configuration editors might also use these JTAG preferences. Some Tcl commands may also use these JTAG preferences by default, though these can always be overridden with Tcl command arguments/flags.

The section [Configuring the JTAG Connection \(page 360\)](#) explains the proper use of all fields of this page in detail.

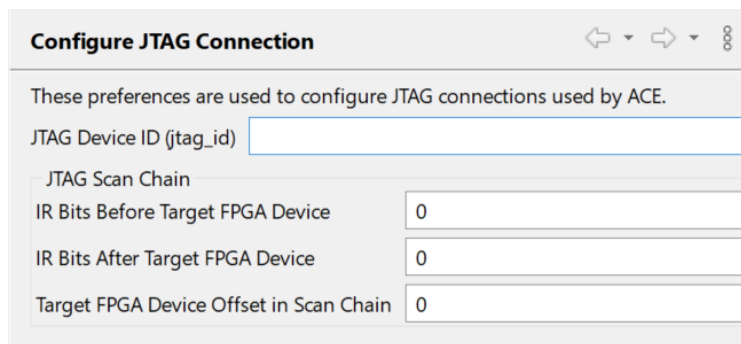


Figure 96 - Configure JTAG Connection Preference Page Example

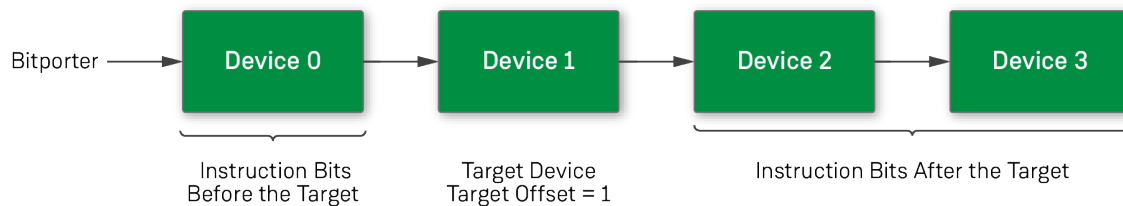
Table 104 - Configure JTAG Interface Preference Page Options

Option	Description
JTAG Programmer Device Name ⁽¹⁾	<p>The name of the JTAG device which should be used for all ACE JTAG interactions with the chosen FPGA or eFPGA. If the name is not specified, auto-detection of JTAG devices is attempted.</p> <div style="border: 1px solid green; padding: 5px; margin-top: 10px;"> <p> Performance Tip</p> <p>Even if only one JTAG device is connected, specifying the JTAG device by name (instead of relying upon auto-detection) can save up to several seconds of initialization time on every JTAG connection.</p> </div>
JTAG Scan Chain	
IR Bits Before the Target FPGA Device ⁽²⁾	<p>Sets the (decimal) number of instruction register bits between the board JTAG TDI pin and the target device.</p>
IR Bits After the Target FPGA Device ⁽²⁾	<p>Sets the (decimal) number of instruction register bits between the target device and the board JTAG TDO pin.</p>
Target FPGA Device Offset in Scan Chain ⁽²⁾	<p>Sets the device count (in decimal) between the board JTAG TDI pin and target FPGA device.</p>

Option	Description
<p>Table Notes</p> <ol style="list-style-type: none"> 1. Auto-detection can only be used safely when just one JTAG pod/device is connected. If more than one pod/device is automatically detected while no name is specified, JTAG interactions fail, stating that it is required to specify which pod/device to use. The Tcl command <code>jtag::get_connected_devices</code> provides a list of connected JTAG device names. See the <i>Speedster7t Configuration User Guide (UG094)</i> for more information. 2. The default value of zero is always correct for single-device JTAG scan chains. For multi-device scan chains, the default values of all zeros never work. 	

Multi-device JTAG Scan Chain (IEEE 1149.1) Example

The following high-level diagram summarizes how ACE must be configured for JTAG daisy-chains. For an explanation of daisy-chained JTAG scan chains, visit https://en.wikipedia.org/wiki/Jtag#Daisy-chained_JTAG_.28IEEE_1149.1.29.



557324-01.2023.05.20

Figure 97 - Example High-Level Diagram of a Multi-device JTAG Scan Chain

When multiple FPGA devices are attached to the same JTAG scan chain, the target FPGA device must be specified. The FPGA device closest to the Bitporter2 (more accurately, closest to the board JTAG TDI pin) has a target offset of 0.

Because different FPGA devices have different instruction register (IR) sizes, the total IR bit length before and after the target must be specified as well. Achronix devices have a JTAG instruction register size of 23 bits. Hence, in the above example diagram, if all devices were Achronix FPGA devices, there would be 23 IR bits before the target FPGA device, (23 IR bits within the target FPGA device,) and 46 IR bits after the target FPGA device.

Note

For a more thorough explanation regarding ACE multi-device JTAG scan chain configurations, refer to [Configuring the JTAG Connection \(page 360\)](#).

Critical Path Diagram View Preference Page

The Critical Path Diagram View Preference Page configures the display [Preferences \(page 188\)](#) for the [Critical Path Diagram \(page 33\)](#) view.

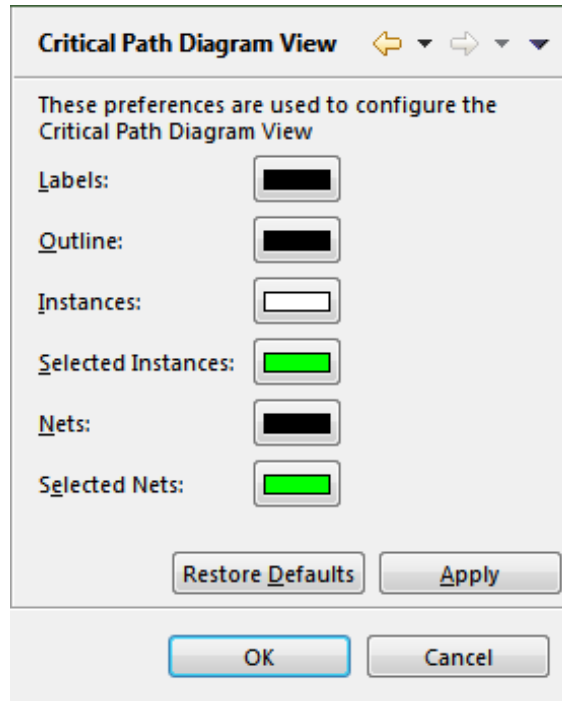


Figure 98 - Critical Path Diagram View Preference Page Example

Table 105 - Critical Path Diagram View Preference Page Options

Option	Description
Labels	Configures the color of the label text printed for graph nodes and arrows in the diagram.
Outline	Configures the color of the outline of the graph nodes in the diagram.
Instances	Configures the background color of graph nodes in the diagram.
Selected Instances	Configures the background color of graph nodes representing instances in the ACE Selection Set in the diagram.
Nets	Configures the color of the arrows in the diagram.

Option	Description
Selected Nets	Configures the color of arrows representing nets in the ACE Selection Set in the diagram.
Restore Defaults	Returns all preferences on this page to their ACE default values.
Apply	Immediately applies any configuration changes on this page to the current diagram in the Critical Path Diagram view (page 33) . These config values are also saved and are used in all future ACE sessions. The Preferences dialog stays open to allow other preference configuration changes, if desired.
OK	Immediately applies any preference configuration changes (including on other preference pages). These config values are also saved and are used in all future ACE sessions.
Cancel	Discards any preference configuration changes made since the dialog was opened (or since the last time the Apply button was clicked in the dialog, whichever was most recent).

Floorplanner View Colors and Layers Preference Page

The Floorplanner View Colors and Layer Preference page configures the colors of multiple layers (and states within the layers) for the [Floorplanner view \(page 43\)](#). Additionally, options are provided allowing a degree of control over the display priorities of the possible [Instance states \(page 262\)](#) and Net/Route highlighting vs. selection. For more about Highlighting, see [Highlighting Objects in the Floorplanner View \(page 343\)](#). For more about Selection, see the [Selection view \(page 133\)](#).

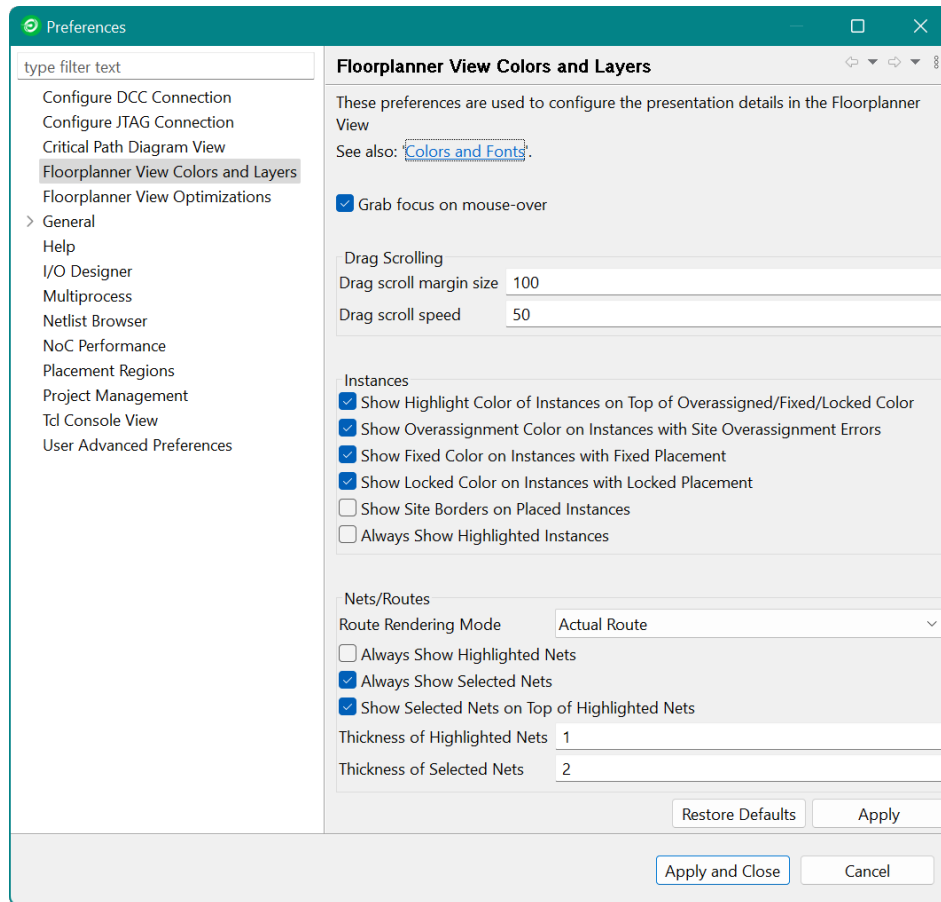


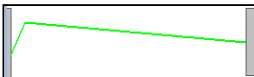
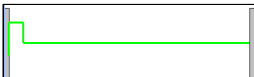
Figure 99 • Floorplanner View Colors and Layers Dialog

The meanings of the various **Instance states** (page 262) are defined elsewhere. While Instances can have multiple states at once, they can only show a single state at a time, thus there is some ability to alter the display priority of the various Instance states.

Similarly, the display of the Floorplanner Nets/Routes layers can be adjusted. (Both clock and non-clock nets are affected identically by these settings.)

Table 106 • Instances and Nets/Routes Preferences

Preference	Description
Instances	
Grab focus on mouse-over	If enabled, the Floorplanner view grabs focus whenever the cursor moves into the view.

Preference	Description
Drag scroll margin size	The size, in pixels, of the "drag scroll margins." Defines how close the cursor must get to an edge of the Floorplanner view to start drag scrolling.
Drag scroll speed	Defines How far, in pixels, the view moves on each drag scroll.
Show Highlight Color of Instances on top of Overassigned/Fixed/Locked Color	When enabled, the Instance Highlight color has a higher priority than all other Instance states except Selection.
Show Overassignment Color on Instances with Site Overassignment Errors	Allows toggling whether site over-assignment errors are displayed visually on the Floorplanner view.
Show Fixed Color on Instances with Fixed Placement	If disabled, both fixed placement and non-fixed placement instances are shown in the same color, grey by default.
Show Locked Color on Instances with Locked Placement	If disabled, locked and non-locked instances are shown with the same color.
Show Site Borders on Placed Instances ⁽¹⁾	If enabled, all placed instances are rendered with the site border color as an outline around the instance. If disabled, placed instances are rendered without a site border.
Always Show Highlighted Instances	If enabled, highlighted instances are always displayed, even if their layer (Instances/Selected Instances) is otherwise disabled. If disabled, highlighted instances are only displayed when their layer (Instances/Selected Instances) is enabled.
Nets/Routes	
Route Rendering Mode	<p>Allows altering how Non-clock Routes and Clock Routes are drawn (when the Non-clock Routes or Clock Routes layers are enabled, respectively, in the Floorplanner palette). Choices are Actual Route and 1 Corner.</p> <ul style="list-style-type: none">  Actual Route mode draws a single straight line from wire sources to wire sinks for each route segment.  1 Corner mode draws two lines for each wire: from each wire source, draws a vertical line to the wire sink Y coordinate, then a horizontal line to the wire sink; no diagonal lines are used, which speeds rendering in complex designs.

Preference	Description
Always Show Highlighted Nets	If enabled, Highlighted nets are always displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Highlighted nets are only displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Always Show Selected Nets	If enabled, Selected nets are always displayed, even if their layer (Clock Routes or Non-clock Routes) is otherwise disabled. If disabled, Selected nets are only displayed when their layer (Clock Routes or Non-clock Routes) is enabled.
Show Selected Nets on Top of Highlighted Nets	Allows changing whether the Selection or Highlight color takes priority, and is painted "on top" of the other state during rendering.
Thickness of Highlighted Nets	The highlight color of a net is rendered this many pixels wide.
Thickness of Selected Nets	The selection color of a net is rendered this many pixels wide.

Table Notes

1. **Caution:** When **Show Site Borders on Placed Instances** is enabled while the view is extremely zoomed out, the site border render color might actually hide the placement state color. Thus, this preference is disabled by default.

Note

- The **1 Corner** route drawing mode is significantly faster (up to 5×) than the **Actual Route** drawing mode, but it can make individual routes harder to differentiate from each other. When Floorplanner performance is a concern, use **1 Corner** mode when possible, and only switch to **Actual Route** mode when needed.
- It is possible to show both the selection and highlight state of a single net simultaneously. Simply ensure the higher priority (top) state has a narrower thickness than the lower priority (bottom) state, and the bottom state color is rendered as a "halo" effect around the top state color.
- Having the "Always Show..." preferences enabled might be most useful in designs with massive numbers of nets/routes. Both the Clock and Non-clock route layers can be disabled, and then Select or Highlight the interesting routes, and only the Selected and/or Highlighted routes are displayed, without the distractions of the less-interesting routes cluttering up the view.

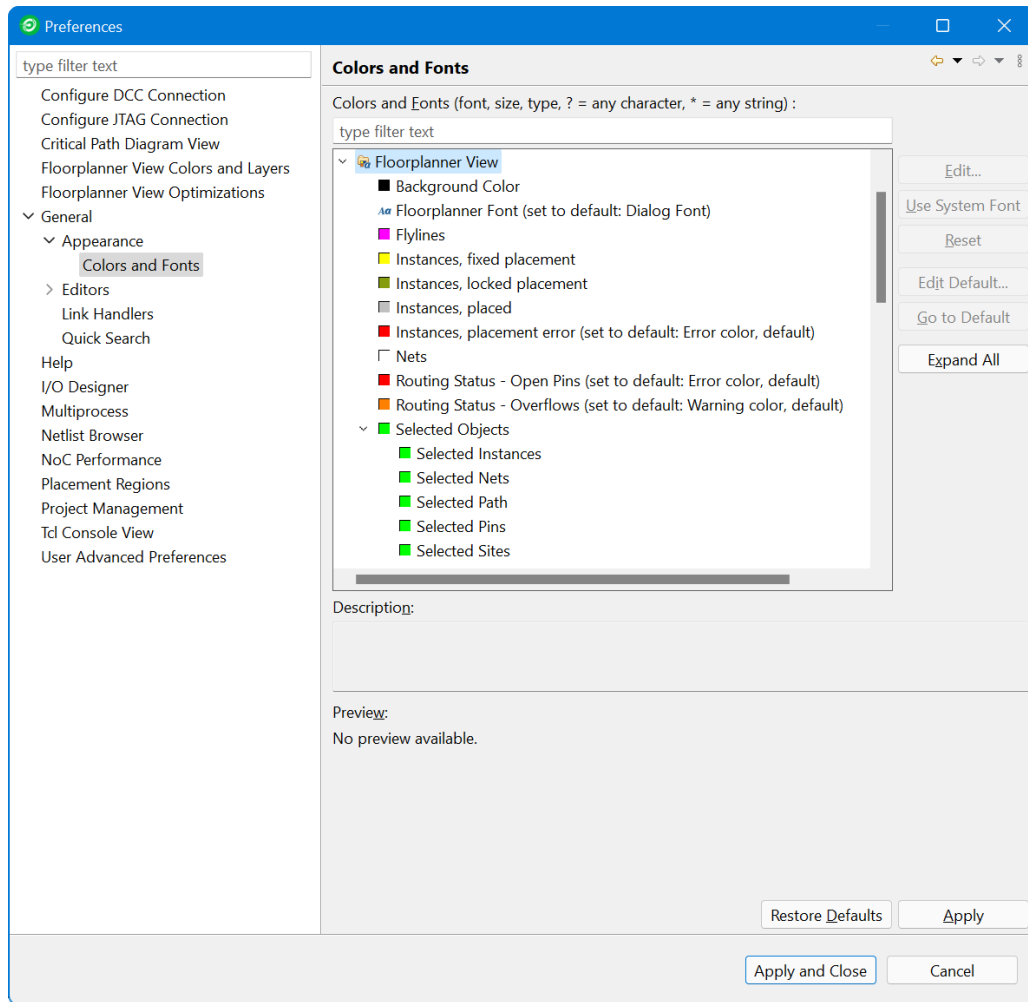


Figure 100 - Colors and Fonts | Floorplanner View section

Table 107 - Color Preferences

Color Preference	Description
Background Color	Used to render the background of the device.
Instances, placed	Represents instances with default (Soft) placement.
Instances, fixed placement	Represents instances with Fixed placement.
Instances, locked placement	Represents instances with Locked placement.

Color Preference	Description
Instances, placement error	Represents an instance that shares a site assignment with another instance. Since a site can only legally contain a single instance, this is an error state.
Nets	Represents all net Routes for both clock and non-clock nets.
Flylines	Flylines are only rendered for Selected Instances, and only when the Layer called Selected Instance Flylines is enabled. These are straight single-segment lines directly connecting a net source instance and sink instance, where either the source or sink is a Selected Instance.
Selected Instances	Represents any Instance that is a member of the Selection Set (and is thus also visible in the Selection View (page 133)).
Selected Nets	Represents any Net that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Pins	Represents any Pin that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Paths	Represents any Path that is a member of the Selection Set (and is thus also visible in the Selection View).
Selected Sites	Represents any Site that is a member of the Selection Set (and is thus also visible in the Selection View).
Routing Status - Open Pins	Represents Open Pins, the endpoints of Open Connections (which are the unrouted portions of a net). Open Pin squares are only visible when enabled in the Layers section of the Floorplanner view Palette.
Routing Status - Overflows	Represents Overflow pins, a rare routing error state. Overflow diamonds on pins are only visible when enabled in the Layers section of the Floorplanner view Palette.

Floorplanner View Optimizations Preference Page

The Floorplanner View Optimizations Preference page configures rendering optimizations for the **Floorplanner view** (page 43). The following Floorplanner View Optimizations page example includes default values for Windows:

Floorplanner View Optimizations

These preferences are used to configure the Floorplanner View's performance optimizations. (Windows users should be especially careful when changing these options, since some combinations can make the application appear non-responsive to the Windows OS, which can then cause an infinite repainting loop.)

The Floorplanner will choose different optimizations based upon the complexity of the active design. The complexity of a design is currently measured by the total number of routing segments.

Optimization settings which may vary with design complexity

	High	Med	Low
When panning, show only background layer:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Enable incremental rendering:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Show intermediate render stages:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Render large areas as smaller tiled areas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Max re-quartering recursion:	<input type="text" value="2"/>	<input type="text" value="2"/>	<input type="text" value="1"/>
Max unsegmented area:	<input type="text" value="100"/>	<input type="text" value="400"/>	<input type="text" value="800"/>
Reduce overdraw with route preprocessing and caching:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Max zoom level to be preprocessed/cached:	<input type="text" value="5"/>	<input type="text" value="5"/>	<input type="text" value="5"/>

Current Design's Complexity: N/A (Please activate a project/implementation)

Settings used for all complexities

Route segment count complexity cutoff, High/Med:

Route segment count complexity cutoff, Med/Low:

Nets processed per incremental render work unit:

Route segments processed per incremental render work unit:

Enable polyline rendering (Early Access):

Enable temp collection route rendering:

Force faster classic rendering APIs when possible:

Figure 101 • Floorplanner View Optimizations Preference Page Example

Designs on modern FPGAs are continuing to increase in complexity. To maintain acceptable Floorplanner performance, highly complex designs require significant rendering optimizations. The set of preferences on this page allows advanced users, FAEs, and tech support to adjust the Floorplanner optimization settings in rare cases when it proves necessary.

ACE tracks three levels of design complexity (High, Med, and Low) and, by default, enables or disables individual optimization settings based upon the design complexity. Because drawing the routing has the most significant impact upon Floorplanner render performance, design complexity is measured in terms of the route segment count. The cutoffs between complexity levels are user configurable.

Note

By default, all optimization features are disabled for the simplest designs. As design complexity increases, more optimizations are enabled by default. Some optimizations have overhead, and actually hinder render performance on small designs — for this reason, all optimizations are typically disabled for the simplest (Low complexity) designs.

The current (active) design Floorplanner complexity is always reported in this preference page as **Current Design's Complexity:** (near the middle of the page). This detail helps to determine which column of optimization settings impacts the rendering of the current design in the Floorplanner. Be aware that the route segment count used to compute the design complexity only includes route segments of routed nets. The same design often reports a Low complexity before it is routed, and a High complexity after routing is complete.

Table 108 • Optimization Settings

Option	Technical Description	Usability Notes
Optimization settings which may vary with design complexity		
When panning, show only background layer:	Reduces the amount of rendering performed while panning / scrolling; the detailed render only occurs after panning / scrolling is completed.	Enable this if panning / scrolling the Floorplanner feels too slow.
Enable incremental rendering:	The render work is broken up into small chunks within each render layer and performed asynchronously, instead of performing the entire render of all layers at once. Because the work chunks are performed asynchronously, the application has a chance to respond to subsequent mouse and keyboard input earlier, instead of waiting for the entire render to complete.	If enabled, each Floorplanner render is slightly (5% to 10%) slower overall, but the Floorplanner Perspective becomes significantly more responsive to mouse and keyboard actions. Obsolete renders might then be interrupted (allowing faster renders when quickly changing zoom levels with the mouse wheel, for example). In some cases (for example, if a great deal of Floorplanner panning/scrolling occurs), there might be noticeable rendering latency/lag.
Show intermediate render stages:	When renders are slow, it can be frustrating to stare at an empty grey/black window waiting for something to change. When this setting is enabled, the user can observe as render layers are built up into the final rendered image.	Provides more frequent visual feedback during rendering, (so it can feel faster, because something is visibly happening,) but renders are actually slightly slower overall.

Option	Technical Description	Usability Notes
Render large areas as smaller tiled areas:	The total render area, if greater than the Max unsegmented area , are broken into quadrants which are rendered individually. Rendered areas are checked for (re-)quartering up to Max re-quartering recursion times. Because the quadrant area is smaller, it completes rendering faster than the whole.	Large render areas are each broken into four smaller chunks for increased visual feedback. Enabling this increases total render times, but it might feel faster, because something is visibly happening.
<ul style="list-style-type: none"> • Max re-quartering recursion: 	The maximum number of times a render area may be broken into smaller (and smaller) pieces. Only relevant when Render large areas as smaller tiled areas is enabled.	Relevant at the outermost zoom levels. Increasing this value might increase total render times, but can provide more frequent visual feedback.
<ul style="list-style-type: none"> • Max unsegmented area: 	Areas larger than this are broken into smaller chunks up to Max re-quartering recursion times. Only relevant when Render large areas as smaller tiled areas is enabled.	Relevant at the outermost zoom levels. Decreasing this can increase total render times, but can provide more frequent visual feedback.
Reduce overdraw with route preprocessing and caching:	Significantly improves render speeds by reducing route line overdraw via culling. Routing data is pre-processed and cached at multiple zoom levels when the routing data is loaded/updated. Due to memory constraints, only the outermost zoom levels may be cached.	By pre-processing routes at multiple zoom levels when the routes are loaded, we reduce the number of route lines we draw over preexisting routes lines of the same color. This slightly increases the memory footprint and load times, but significantly reduces render times for large designs (when overdraw is most frequent).
<ul style="list-style-type: none"> • Max zoom level to be preprocessed/cached: 	0=zoomed all the way out, 1=zoomed in one step, etc.	This number must be kept small to avoid running out of memory. (Floorplanner route render cache sizes can more than double at each increasing zoom level.)
Settings used for all complexities		
Route segment count complexity cutoff, High/Med:	Designs with a route segment count greater than this number use the optimization settings for High complexity designs (the first column of checkboxes/fields).	-
Route segment count complexity cutoff, Med/Low:	Designs with a route segment count less than (or equal to) this number use the optimization settings for Low complexity designs (the third column of checkboxes/fields).	-
Nets processed per incremental render work unit:	The number of nets to process per discrete work unit when rendering the routing layers. Used when the current zoom is NOT in the route overdraw reduction cache. Only relevant when Enable incremental rendering is on.	Increasing this number might very slightly improve rendering performance, but decrease the frequency of visual feedback.

Option	Technical Description	Usability Notes
Route segments processed per incremental render work unit:	The number of route segments to process per discrete work unit when rendering cached route segments. Used when the current zoom is in the route overdraw reduction cache. Only relevant when both Enable incremental rendering and Reduce overdraw with route preprocessing and caching are on.	Increasing this number might very slightly improve rendering performance, but decrease the frequency of visual feedback.
Enable polyline rendering (Early Access) ⁽¹⁾	-	Polyline rendering of the routes is a known major Floorplanner performance advantage in Windows, but only minor advantages were seen in tested Linux configurations. The advantages are most noticeable in the largest designs.
Enable temp collection route rendering: ⁽²⁾	-	Might slow rendering when using route overdraw reduction cache, but other route rendering might speed up slightly.
Force faster classic rendering APIs when possible:	-	The classic rendering APIs are significantly faster in all tested cases, but might produce slightly cruder visual output due to a lack of anti-aliasing. (The modern/advanced render APIs are still automatically used when absolutely required.)

Table Notes

1. **Caution: Enable polyline rendering** is new, Early Access functionality. While Achronix found no negative stability or performance impacts when testing this optimization, customer hardware and software have a wider variety of configurations than in our test labs. If any new performance or stability issues are observed in the GUI, please contact Achronix Technical Support. Achronix notes the specifics of your configuration (to reproduce and correct the problem), and might then suggest disabling this new polyline rendering functionality to determine if performance or stability improves.
2. **Caution: Enable temp collection route rendering** requires significantly more ACE GUI memory when enabled!

Note

Technical Note for Windows Users

The Windows operating system requires that applications check-in every five seconds, or the application is deemed non-responsive. Non-responsive applications are given a figurative kick-in-the-pants by Windows, and asked to repaint the screen. When the screen paint itself is taking more than five seconds, as might happen with poor Floorplanner Optimization settings, an application can be forced into an effective infinite-loop of paint requests from the operating system.

If a Windows user ever notices ACE being called non-responsive by Windows (check the application title bar), ACE has most likely entered this looped painting state. To escape, change back to the Project perspective (or any other perspective without the Floorplanner view visible), then on this Floorplanner View Optimizations Preference page, ensure that incremental rendering and tiled rendering are both enabled for the current design complexity level. If both are already enabled, please call Achronix Technical Support for guidance on further Floorplanner optimization adjustments.

Warning!
Disabling optimizations that are enabled by default is not recommended.

I/O Designer Preference Page

The I/O Designer Preference page contains a number of preferences for the I/O Designer views.

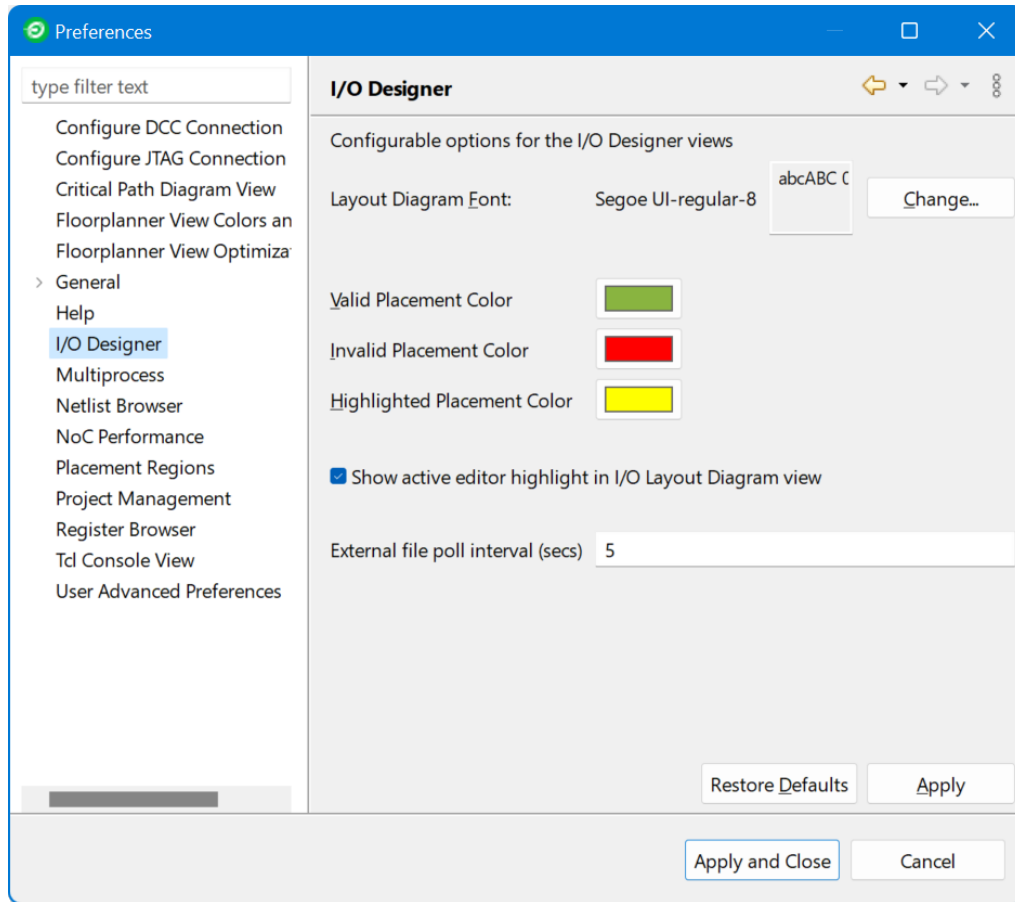


Figure 102 · I/O Designer Preference Page Example

Table 109 · I/O Designer Preferences Options

Option	Description
Layout Diagram Font	The font used on shapes in the I/O Layout diagram.
Valid Placement Color	The color used for sites with valid placements.

Option	Description
Invalid Placement Color	The color used for sites with invalid placements.
Highlighted Placement Color	The color used to highlight the currently selected site.
Show active editor highlight...	Toggles whether the currently selected site is drawn with a highlight.
External file poll interval (secs)	Specifies how often IP files are checked for external modification.

IP Diagram Color and Font Preferences Section

While there is no longer a single preference page dedicated only to IP Diagram Preferences, there are still a number of color and font preferences related to the **IP Diagram view** (page 69) which are available for user manipulation, now located under **General** → **Appearance** → **Colors and Fonts** in the preference tree under the **IP Diagram View** branch.

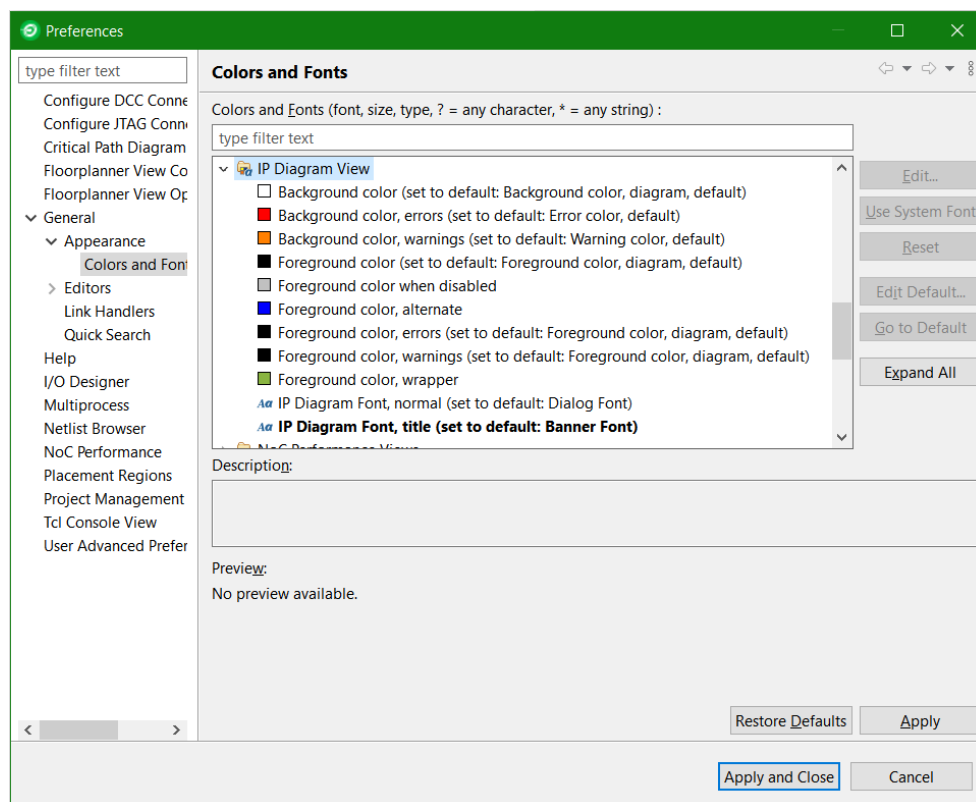


Figure 103 - IP Diagram Preferences Found on the Colors and Fonts Page

Table 110 - IP Diagram Color and Font Preference Options

Option	Description
Background Color	The default background color for the entire diagram.
Background Color, errors	Text representing IP Options with errors have their backgrounds painted this color.
Background Color, warnings	Text representing IP Options with warnings have their backgrounds painted this color.
Foreground Color	The color used for enabled logic blocks and signals (and their textual labels).
Foreground Color when disabled	The color used for disabled logic blocks and signals (and their textual labels).
Foreground Color, alternate	The color used to highlight portions of the diagram.
Foreground Color, errors	Text representing IP Options with errors have their foregrounds painted this color.
Foreground Color, warnings	Text representing IP Options with warnings have their foregrounds painted this color.
Foreground Color, wrapper	For core IP, the color used to represent the IP RTL wrapper itself. Everything enclosed by this color is within the wrapper.
IP Diagram Font, normal	The font used for all diagram text except the logic block titles.
IP Diagram Font, title	The font used to title logic blocks in the diagram.

Multiprocess: Configure Custom Job Submission Tool Preference Page

The Multiprocess: Configure Custom Job Submission Tool Preference Page allows configuring the ACE **Multiprocess View** (page 73) to submit Multiprocess jobs to a third-party cloud/grid/job submission system using a command-line tool. As a useful example, by default, ACE is configured to use an Oracle Grid Engine derivative via the `qsub` command (see https://en.wikipedia.org/wiki/Oracle_Grid_Engine). Be aware that the Oracle Grid Engine `qsub` arguments are not 100% compatible with the `qsub` standard (documented at <http://pubs.opengroup.org/onlinepubs/9699919799/utilities/qsub.html>).

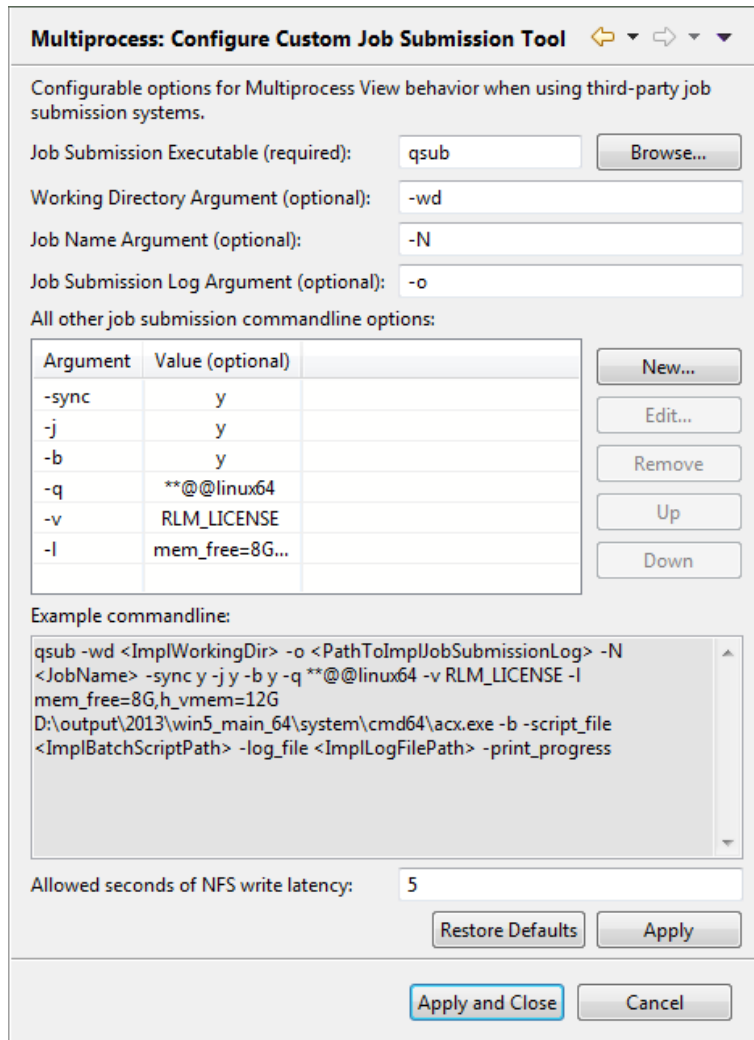


Figure 104 - Multiprocess: Configure Custom Job Submission Tool Preference Page Example

When the third-party job submission system support is enabled, ACE calls the specified executable, using the specified command-line arguments, to submit ACE Multiprocess jobs.

See [Configuring ACE to use an External Job Submission System \(page 311\)](#) for more information.

Warning!**Debugging job submission configurations:**

- If the job submission system is properly configured on the host machine (meaning it is possible to successfully submit non-ACE jobs from the command-line) and ACE is still unable to successfully submit jobs, please contact Achronix technical support.
- **DO NOT** copy the text of the attempted command and manually attempt to submit from the command-line.

Netlist Browser Preference Page

The Netlist Browser preference page contains a number of preferences for the Netlist Browser view.

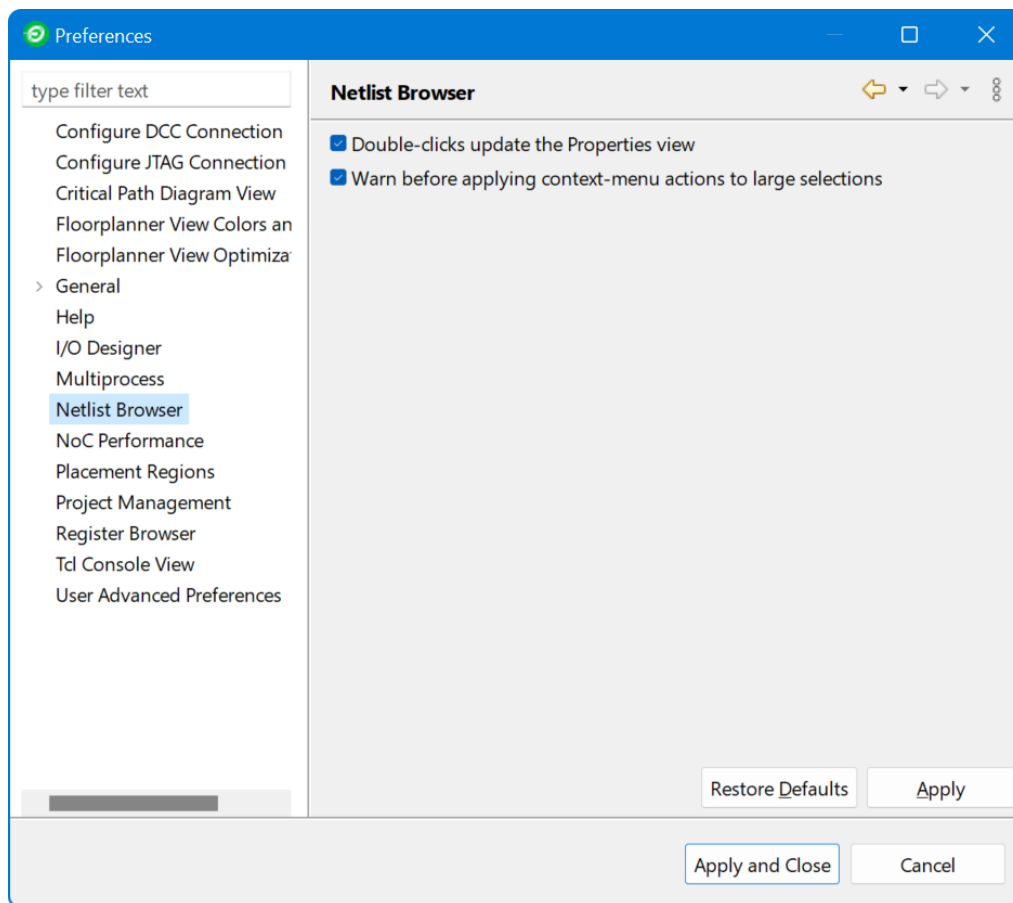


Figure 105 • Netlist Browser Preference Page Example

Table 111 - Netlist Browser Preferences Options

Option	Description
Double-clicks update the Properties view	Toggles whether double-clicking an item in the Netlist browser view shows that item in the Properties view.
Warn before applying context-menu actions to large selections	Applying actions to very large selections can take a very long time. Toggling this option causes a warning message to appear before an action is run against a large selection.

NoC Performance View Preference Page

The NoC Performance view preference page offers preferences related to the 2D NoC Performance view.

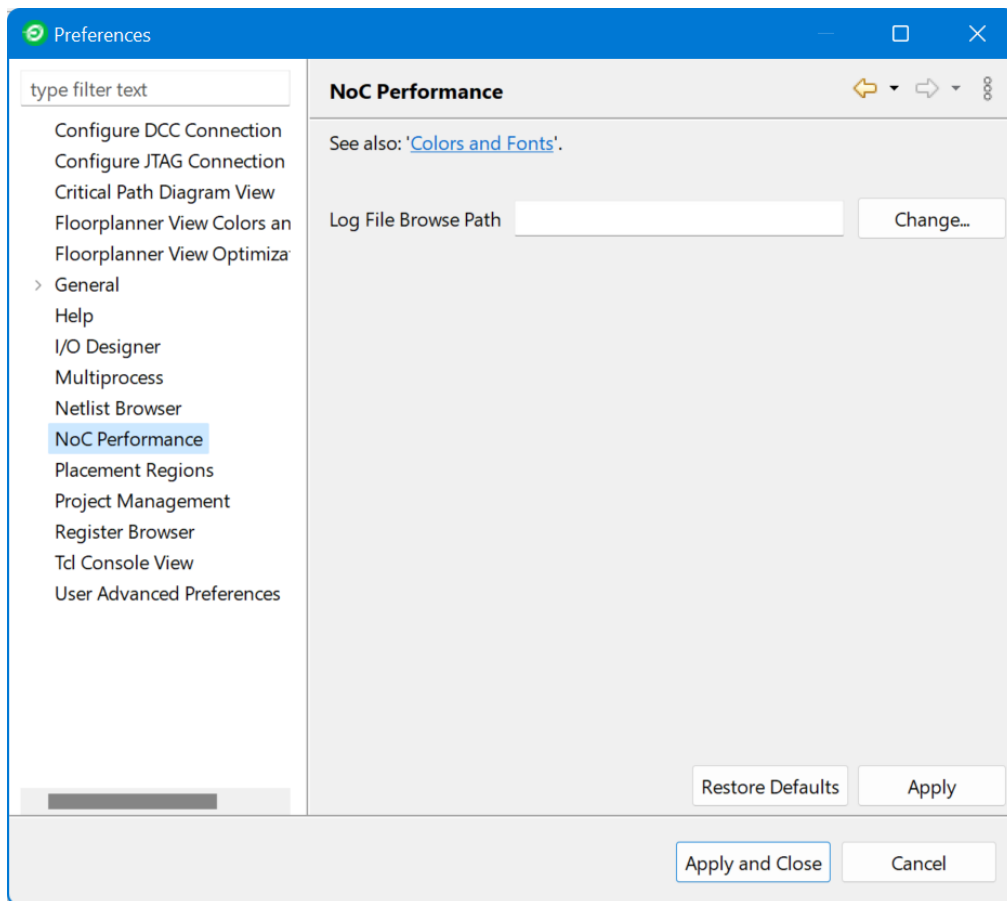


Figure 106 - NoC Performance View Preference Page

Table 112 · NoC Performance View Preferences

Preference	Description
Log File Browse Path	By default, browsing for simulation log files starts in whatever directory was last browsed. Setting the log file directory path causes browsing to always start at the specified path.

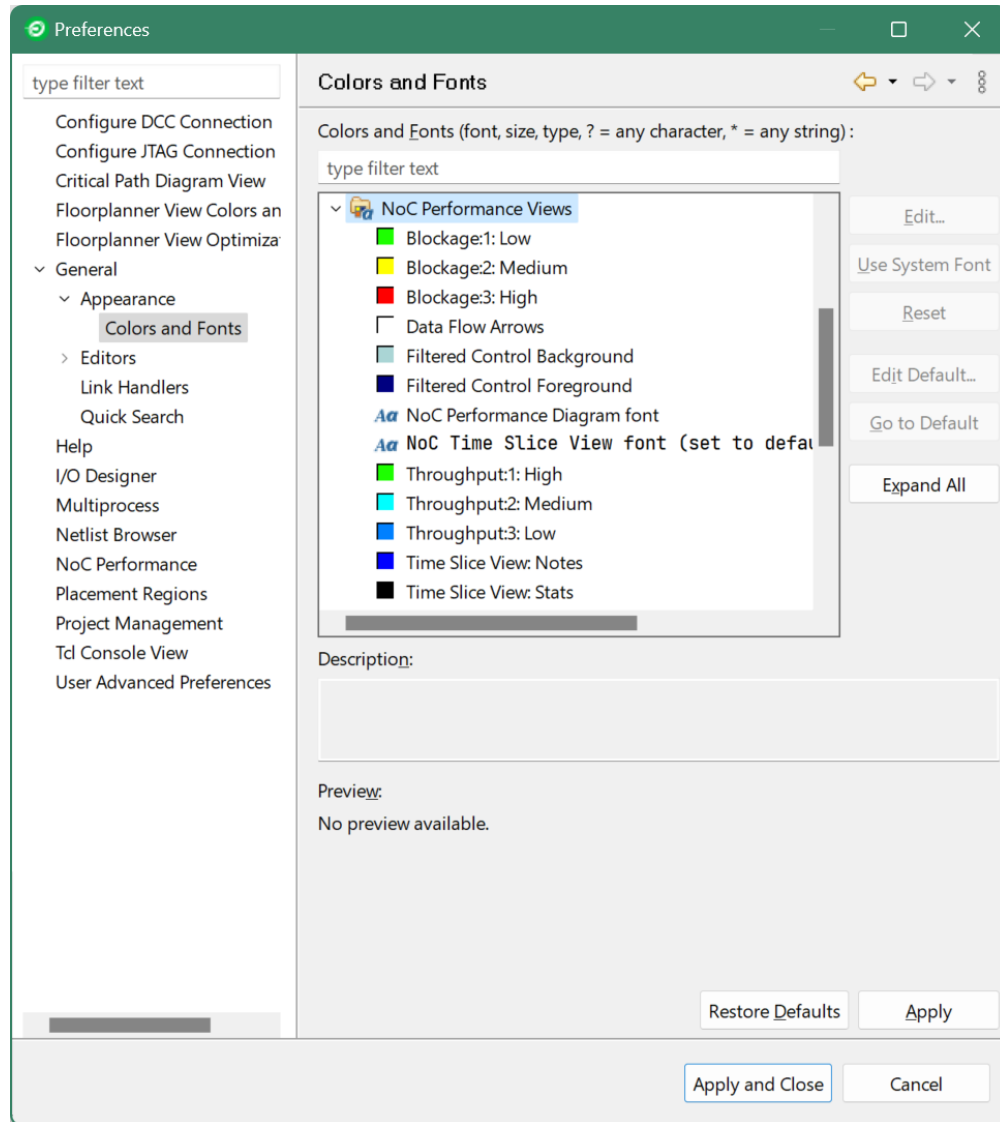


Figure 107 · Colors and Fonts | NoC Performance View Section

Table 113 · Color Preferences

Color Preference	Description
Blockage:1:Low	Selects the color to render shapes with "low" blockage.
Blockage:2:Medium	Selects the color to render shapes with "medium" blockage.
Blockage:3:High	Selects the color to render shapes with "high" blockage.
Data Flow Arrows	Selects the color to render "data flow direction" arrows.
Filtered Control Background	Color for the background of "filtered" controls in the NoC Performance view
Filtered Control Foreground	Color for the foreground of "filtered" controls in the NoC Performance view
NoC Performance Diagram font	Selects the font used to label diagram shapes.
Noc Time Slice view font	Selects the font used to render text in the NoC Time Slice view
Throughput:1:High	Selects the color to render shapes with "high" throughput.
Throughput:2:Medium	Selects the color to render shapes with "medium" throughput.
Throughput:3:Low	Selects the color to render shapes with "low" throughput.
Time Slice view: Notes	Selects the color to render "notes" in the Time Slice view.
Time Slice view: Stats	Selects the color to render "stats" in the Time Slice view.

Other Colors and Fonts Preference Page

The Other Colors and Fonts Preference Page allows setting many of the fonts, colors and components used by ACE. The page is accessed by selecting **General** → **Appearance** → **Other Colors and Fonts**.

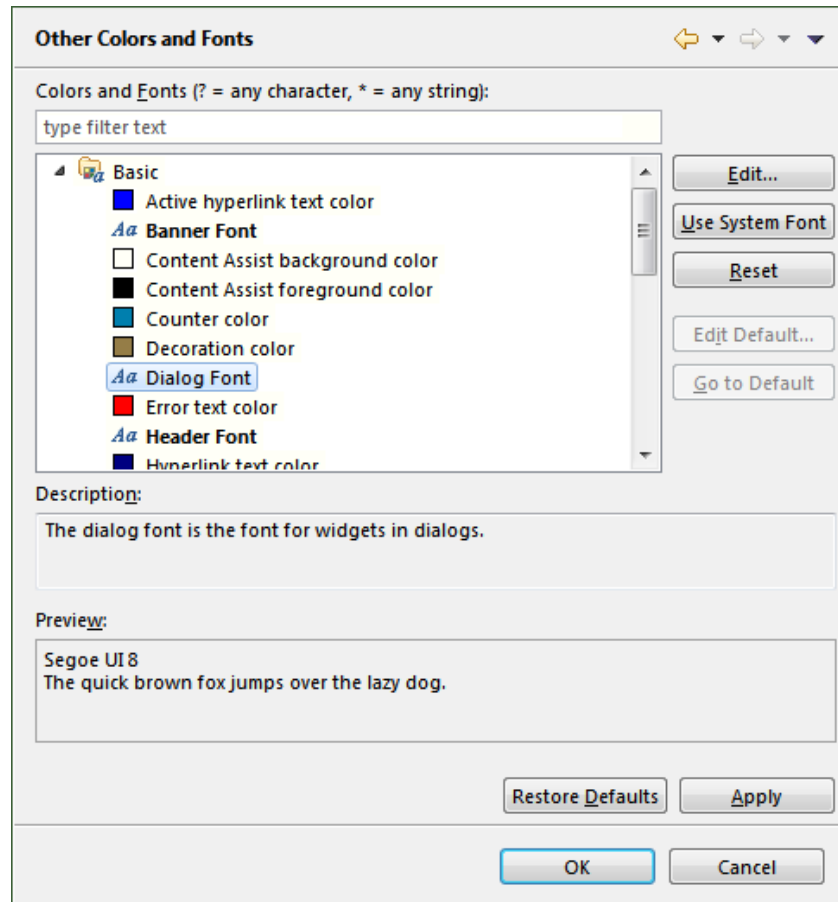


Figure 108 • Other Colors and Fonts Preference Page Example

A tree is used to navigate among and show a short preview of the various colors and fonts. The current face (but not size) of any font is previewed in its label. Colors are previewed in the icon associated with its label. Additionally, some categories (Workbench in particular) provide a more detailed preview of their contributions (shown below the description area, if available).

- Font settings can be changed either by selecting the font from the list and clicking **Use System Font** to use the operating system font, or by clicking **Edit** to open up a font selection dialog. **Reset** can be used to return to the default value.
- Color settings can be changed by clicking **Edit** to the right of the tree area when a color is selected. **Reset** can be used to return to the default value.
- The **Colors and Fonts** text field can be used to filter the contents. Simply type in an entry and any matching results remain in the tree view.
- The **Description:** and **Preview:** sections provide details when the Workbench colors and font settings are selected.

Placement Regions Preference Page

The Placement Regions Preference Page determines how **Placement Regions** (page 394) are handled in the **Placement Regions view** (page 110) and the **Floorplanner view** (page 43) (when the Floorplanner Placement Region Tool is active).

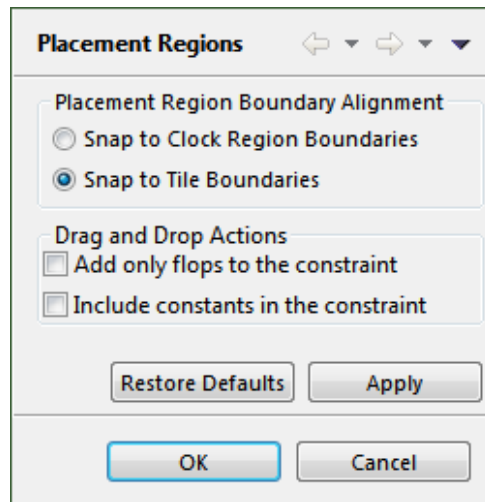


Figure 109 • Placement Regions Preference Page Example

Table 114 • Placement Region Preferences

Option	Description
Placement Region Boundary Alignment	
Snap to Clock Region Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries are forced to align with Clock Region Boundaries. Use this for a simple, coarse-grained approach to Placement Regions. This setting is recommended for most use cases.
Snap to Tile Boundaries	When creating, resizing, or moving Placement Regions, the Placement Region boundaries are forced to align with Tile Boundaries, for a very fine-grained region. This setting is only recommended for advanced users.
Drag and Drop Actions	

Option	Description
Add only flops to the constraint	When using drag-and-drop to assign Placement Region Constraints, this setting ensures only flops are assigned to the region constraint. All other dropped items are excluded from the constraint.
Include constants in the constraint ⁽¹⁾	When using drag-and-drop to assign Placement Region Constraints, this setting includes constants in the constraint.

Table Notes

1. The **Include constants in the constraint** setting is ignored if **Add only flops to the constraint** is enabled.

Project Management Preference Page

The Project Management Preference Page sets the behavior of [Editors \(page 9\)](#) and [Reports \(page 244\)](#).

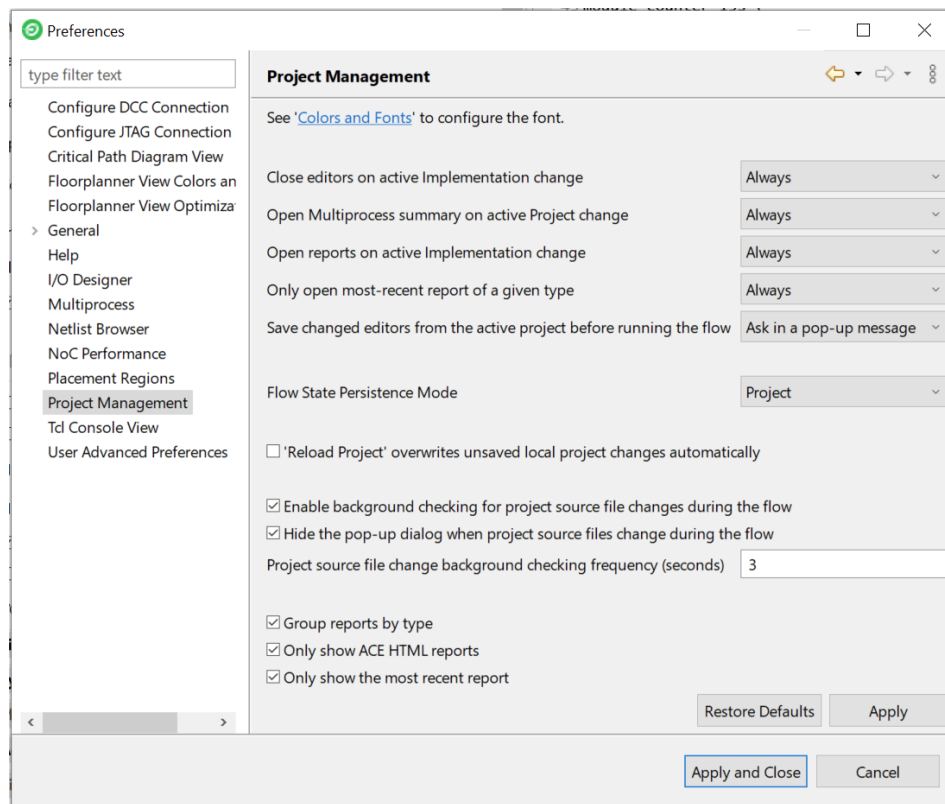


Figure 110 - Project Management Preference Page Example

Table 115 · Project Management Preferences

Option	Description
See also:	Link to Colors and Fonts preferences allowing choice of the font used in the Projects view.
Close editors on active implementation change	Can be used to automatically close open editors when the active implementation changes.
Open Multiprocess summary on active project change	Can be used to automatically open the Multiprocess summary report when the active project changes.
Open reports on active implementation change	Can be used to automatically open implementation-specific reports when the active implementation changes.
Only open most-recent report of a given type	Can be used to automatically open only the most-recent report of a given type.
Save changed editors from the active project before running the flow	Can be used to automatically save all open editors before running the flow.
Flow State Persistence Mode	Choose whether or not the enabled/disabled flow step state is persisted across ACE sessions per Session, per Project, or is Disabled (not persisted)
Reload Project overwrites unsaved local project changes automatically	Can be used to automatically overwrite unsaved local project changes when Reload Project is used (without a confirmation prompt).
Enable background checking for project source file changes during the flow	If enabled, project source files are periodically polled in the background to look for any changes made outside of ACE.
Hide the pop-up dialog when project source files change during the flow	If enabled, source file change notification is suppressed until the flow has finished running.
Project source file change background checking frequency (seconds)	Determines how often to poll for background source file changes.
Group reports by type	If enabled, reports are grouped into subfolders in the Projects view tree.
Only show HTML reports	If enabled, only HTML reports are shown in the Projects view tree.
Only show the most recent report	If enabled, only the most recent report is shown in any subfolder in the Projects view tree.

Tcl Console View Preference Page

The Tcl Console View Preference Page contains settings that alter the behavior and/or presentation of information in the [Tcl Console View](#) (page 142).

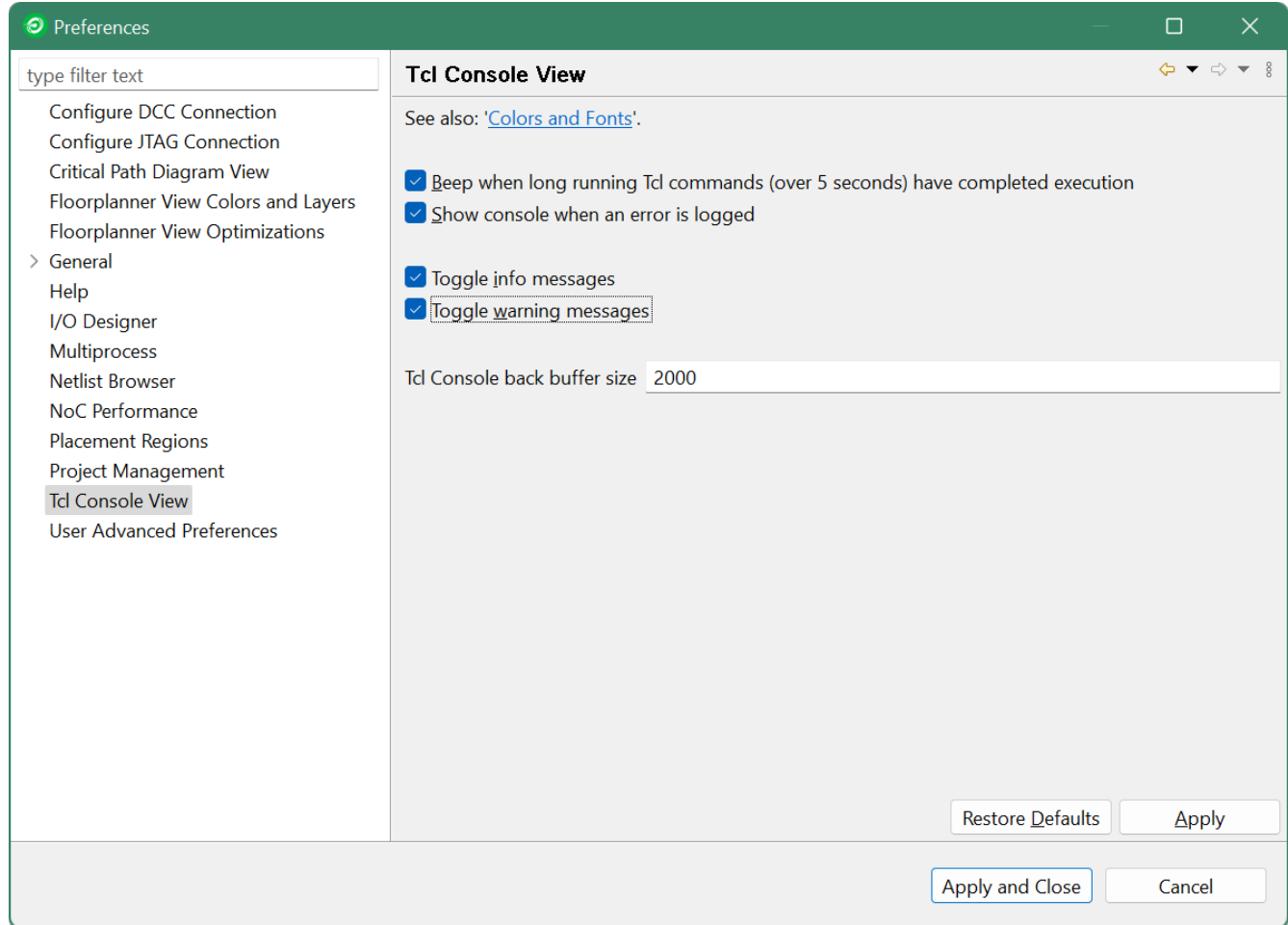


Figure 111 • Tcl Console View Preference Page Example345px345px640

Table 116 • Tcl Console View Preferences

Option	Description
See also: 'Colors and Fonts'	Link to Colors and Fonts Preferences allowing choice of the font used in the Tcl Console.

Option	Description
Beep when long running Tcl commands (over 5 seconds) have completed execution	Enabling this option provides audible feedback (the default system beep/bell sound) upon completion of long-running commands.
Show Console when an error is logged	When enabled, any time an error is logged, the Tcl Console will be shown, even if it was minimized, or beneath another view in a tab stack
Toggle info messages	Toggles the visibility of info messages in the Tcl Console's readout area
Toggle warning messages	Toggles the visibility of warning messages in the Tcl Console's readout area
Tcl Console back buffer size	Defines the maximum number of lines shown in the Tcl Console's readout area. Lowering this value will decrease memory usage.

Text Editors Preference Page

The Text Editors Preference Page sets the behavior and appearance of the text editor.

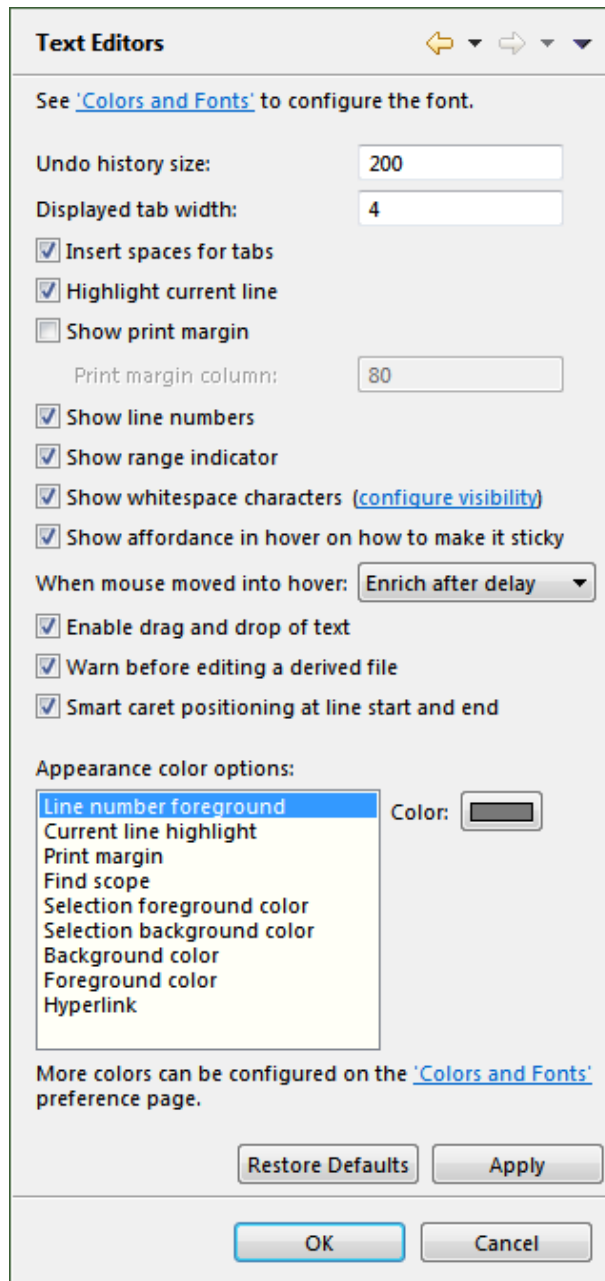


Figure 112 • Text Editors Preference Page Example

Table 117 · Text Editor Options

Option	Default	Description
Undo history size	200	Sets the number of undo events in the history queue.
Display tab width	4	Sets the editor tab width in spaces.
Insert spaces for tabs	Deselected	Enables insertion of spaces in place of tab characters.
Highlighting current line	Selected	Enables/disables the highlighting of the current line. The highlight color is set in Appearance color options .
Show print margin	Deselected	Selects whether the print margin is visible.
Print margin column	80	Sets the print margin column position.
Show line numbers	Selected	Enables/disables the display of line numbers in the Editor view.
Show range indicator	Selected	Enables the display of range indicators in the text editor.
Show whitespace characters	Deselected	Enables the display of whitespace characters (·) in text editors.
Show affordance in hover on how to make sticky	Selected	Enable the affordance (visual clue) in the hover text and make it sticky.
When mouse moved into hover	Enrich after delay	Sets the hover display mode.
Enable drag and drop of text	Selected	Enables/disables the ability to drag and drop selected text.
Warn before editing a derived file	Selected	Enables warning if a derived file is going to be edited.
Smart caret position at line start and end	Selected	Controls whether the editor automatically positions the caret and the start or end of a line.
Appearance color options	Various	Sets custom colors for various aspects of the text editor.

Quick Diff Preference Page

The Quick Diff Preference Page, enables the Quick Diff option and configures its appearance. The page is accessed via **Text Editors** → **Quick Diff**.

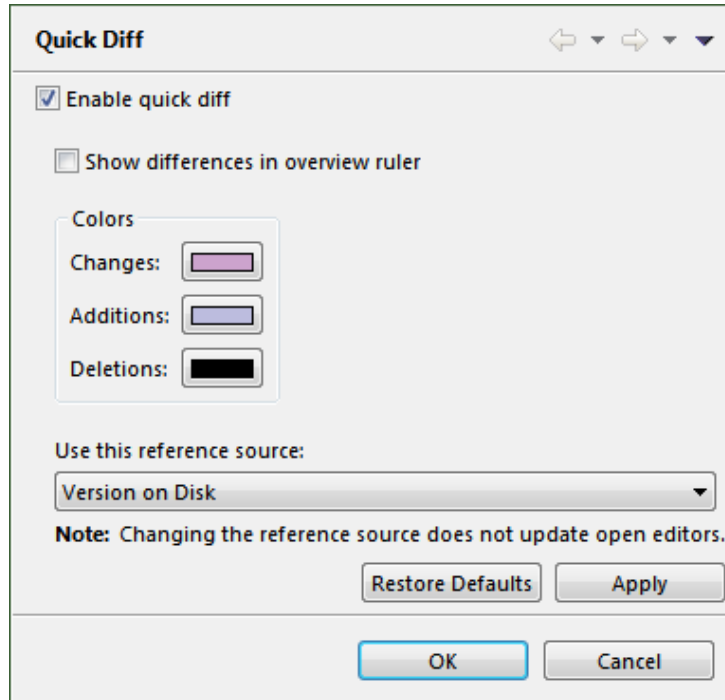


Figure 113 • Quick Diff Preference Page Example

Table 118 • Quick Diff Preference Page

Option	Default	Description
Enable quick diff	Selected	Enables/disables the quick diff option.
Show differences in overview ruler	Deselected	Shows differences in the overview ruler.
Colors		
Changes	-	Sets the color used to indicate changes. ⁽¹⁾
Additions	-	Sets the color used to indicate additions. ⁽¹⁾
Deletions	-	Sets the color used to indicate deletions. ⁽¹⁾

Option	Default	Description
Use this reference source	Version on disk	This option sets which reference to use as the base for generating quick diff comparisons. Options are: Version on Disk: Current file is compared against the last saved version on disk.

Table Notes

1. The button to the right of the option allows changes to the display color (refer to "Changing Color Coding").

Projects

Projects represent the collection of [Source Files \(page 224\)](#), flow options, IP configuration files, and output files for a particular design.

Project File

Projects are persisted in `.acxprj` project files created automatically by the tool whenever a project is saved. A project file is actually just a Tcl script supporting only a defined subset of Tcl commands. Project files can be edited manually and then loaded into the tool to use as a script or for running regressions.

Note

When ACE loads a project file, it locks that file to prohibit other ACE sessions from loading the same file. This prevents project data corruption, which could occur if two sessions attempt to work with the same project at the same time.

In the GUI, loaded project file contents are displayed in a tree structure in the [Projects View \(page 117\)](#). Project file contents may also be viewed in a [Text Editor \(page 10\)](#) in the GUI by double-clicking the project name in the Projects view (example file contents follow):

Example Project file contents

```
# quickstart_canary
# AUTOMATICALLY GENERATED FILE
# MAY BE OVERWRITTEN AT ANY TIME DURING USE OF TOOL

# Created by: ACE -- Achronix CAD Environment -- Version 10.0 -- Build xxxxxx- -- Date
2024-04-10 08:26
# Release Version: 0.X (Main)
set_project_option -project quickstart_canary -- partname "AC20SE16A0012R0"
set_project_option -project quickstart_canary -- package "CORE"
set_project_option -project quickstart_canary -- speed_grade "C2"
set_project_option -project quickstart_canary -- core_voltage "0.80"
set_project_option -project quickstart_canary -- junction_temperature "0"

set_project_option -project quickstart_canary flow_mode "evaluation"
set_project_option -project quickstart_canary hdl_include_path ""
set_project_option -project quickstart_canary use_default_project_output_path "1"

set_project_option -project quickstart_canary sim_flow "Default"
set_project_option -project quickstart_canary sim_tb_top_name "tb_quickstart"
set_project_option -project quickstart_canary sim_tool "Aldec Riviera"

set_project_option -project quickstart_canary bitstream_output_cpu "0"
set_project_option -project quickstart_canary bitstream_output_cpu_width "8"

# Synthesis RTL Files
add_project_source_files -rtl -project quickstart_canary {{./AC20SE16A0012R0/src/rtl/
counter.v}}
add_project_source_files -rtl -project quickstart_canary {{./AC20SE16A0012R0/src/rtl/
quickstart.v}}

# Synthesis Constraint Files
add_project_source_files -syn_constraint -project quickstart_canary {{./AC20SE16A0012R0/
src/constraints/quickstart.sdc}}

# Netlist Files
add_project_source_files -pnr_netlist -project quickstart_canary {{./impl_1/syn/rev_acx/
quickstart_canary_impl_1.vm}}

# PnR Constraint Files
add_project_source_files -pnr_constraint -project quickstart_canary {{./AC20SE16A0012R0/
src/constraints/quickstart.pdc}}
add_project_source_files -pnr_constraint -project quickstart_canary {{./AC20SE16A0012R0/
src/constraints/quickstart.sdc}}

# IP Settings Files
```

```
# Simulation Testbench Files
add_project_source_files -sim_tb -project quickstart_canary {{./AC20SE16A0012R0/src/tb/
tb_quickstart.v}}

# Implementations
# impl_1
create_impl -not_active -project quickstart_canary impl_1

set_impl_option -project quickstart_canary -impl impl_1 fanout_control "1"
set_impl_option -project quickstart_canary -impl impl_1 fanout_limit "95"

set_impl_option -project quickstart_canary -impl impl_1 syn_advanced_options ""
set_impl_option -project quickstart_canary -impl impl_1 syn_default_frequency "200"
set_impl_option -project quickstart_canary -impl impl_1 syn_fanout_limit "200"
set_impl_option -project quickstart_canary -impl impl_1 syn_retiming "1"
set_impl_option -project quickstart_canary -impl impl_1 syn_route_delay_model
"acx_custom_route_delay_3"
set_impl_option -project quickstart_canary -impl impl_1 syn_use_default_project "1"

enable_project_source_file -syn_constraint -project quickstart_canary -impl impl_1
"../../../../../Project Space Test/AC20SE16A0012R0/src/constraints/quickstart.sdc"
enable_project_source_file -pnr_netlist -project quickstart_canary -impl impl_1 "./
impl_1/syn/rev_acx/quickstart_canary_impl_1.vm"
disable_project_source_file -pnr_constraint -project quickstart_canary -impl impl_1
"../../../../../Project Space Test/AC20SE16A0012R0/src/constraints/quickstart.pdc"
enable_project_source_file -pnr_constraint -project quickstart_canary -impl impl_1
"../../../../../Project Space Test/AC20SE16A0012R0/src/constraints/quickstart.sdc"

set_active_impl "impl_1" -project "quickstart_canary"

# End of file
```

Source Files

A project contains source files used as inputs to the ACE tools flow. There are 6 categories of source files:

1. IP Configuration Files
2. RTL Files
3. Synthesis Constraints files
4. Place and Route Constraints files
5. Place and Route Netlist Files
6. Simulation Testbench Files

In the GUI, source files may be browsed in the [Projects View \(page 117\)](#) and viewed or edited in the built-in [Editors \(page 9\)](#) by double clicking the file name in the Projects View, or by using the right-click context menus.

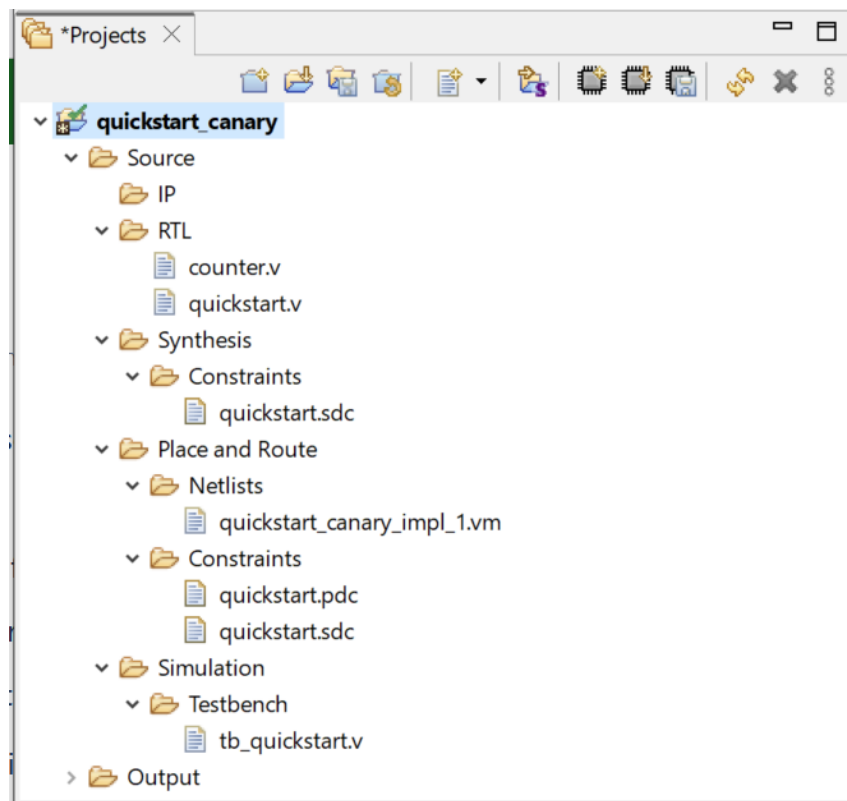


Figure 114 · P

IP Configuration Files

ACE provides GUI support to ease configuration of the most complicated embedded IP in Achronix FPGAs. The ACXIP source data files used by the IP Configuration [Editors \(page 9\)](#) (files with the `.acxip` extension) may optionally be associated with a project. When associated with a project, these IP Configuration files may then be browsed in the [Projects view \(page 117\)](#) under the project IP folder, and the associated editor may be started by double clicking the file name in the Projects View.

There are 2 main categories of IP Configuration files:

1. Soft IP macro configurations, which generate Verilog/SystemVerilog/VHDL modules that are instantiated in the user design RTL and mapped to the Core FPGA fabric. These IP Configuration tools are available for both Speedster FPGAs and Speedcore eFPGAs.
2. I/O Ring configurations, which generate a single configuration for the entire I/O Ring including the I/O Ring bitstream, pin assignments, etc. I/O Ring IP Configuration tools do not output any RTL modules to instantiate in the end user design. They provide a configuration outside the scope of the end user design RTL (which maps to the Core FPGA fabric). These IP Configuration tools are only available for Speedster FPGAs.

For more details, see [Creating an IP Configuration \(page 336\)](#), or one of the many IP Configuration Editors.

RTL Files

The project source RTL (Verilog/SystemVerilog/VHDL) files represent the end user design logic, which is intended to be simulated, synthesized, placed and routed, and mapped to the Core FPGA fabric. The RTL files added to your ACE project should not include Simulation Testbench RTL files. The list of RTL files specified in the ACE project is used to generate the input file list to the Synthesis tool, and is also used in the file list for Simulation (in addition to the Simulation Testbench source files).

Note

Adding RTL source files to your ACE project is optional if you plan to disable the Synthesis and Simulation flow steps and run Synthesis and Simulation outside of ACE. This will require you to manually add your synthesized gate level netlist to the ACE project as a Place and Route Netlist file.

Synthesis Constraints Files

The project source Synthesis Constraint files are used as an input to the Synthesis tool. The file types supported by Synplify Pro are *.sdc, *.scf, and *.fdc. The files added to the ACE project will be used when generating the underlying Synthesis tool project file.

Synthesis Constraint files which have been added to the ACE project can be enabled or disabled per [Implementation \(page 229\)](#). See also: [enable_project_source_file \(page 635\)](#) and [disable_project_source_file \(page 627\)](#)

Note

Adding Synthesis Constraint source files to your ACE project is optional if you plan to disable the Synthesis flow steps and run Synthesis outside of ACE. This will require you to manually add your synthesized gate level netlist to the ACE project as a Place and Route Netlist file.

Place and Route Constraints Files

Place and Route Constraints files are used to control certain aspects of the static timing analysis of the user design, placement of instances in the user design, design partitioning, reporting, and bitstream generation. The constraints file types are defined in the following table.

Place and Route Constraint files which have been added to the ACE project can be enabled or disabled per [Implementation \(page 229\)](#). See also: [enable_project_source_file \(page 635\)](#) and [disable_project_source_file \(page 627\)](#)

Table 119 • Place and Route Constraint File Types

File Type	Description
*.sdc, *.scf	Timing constraints files in SDC format. These files are read by the timer in ACE. All timing constraints must be placed in these files.
*.pdc	Placement constraints files for ACE. ACE can support many functions in this type of file, including placement and insertion of boundary pins. No SDC timing constraints can be used in these files.

File Type	Description
*.prt	Partition definition file for incremental compile and partition-based design with import/export capability in ACE. There should be only one *.prt file per project in ACE. This file is typically generated by Synplify and controls which partitions will be re-compiled in the next ACE run.
*.xml	ACE reporting extensions files, used to augment the power, utilization, or other reports with information from the surrounding ASIC or I/O Ring.
*.hex	Hexadecimal-formatted bitstream extension files, used to augment the Core FPGA fabric bitstream with the bitstream for the surrounding ASIC or I/O Ring.

Note

All *.sdc, *.scf, and *.pdc constraints files are read in and executed in the order specified in the ACE project file. Constraints files that are added to the project first are executed first, and likewise, constraints files which are added to the project last are executed last. If there is any order dependency between commands in your constraints, please make sure to add the constraints files in the correct order for execution in ACE. See also: [move_project_source_file \(page 669\)](#)

At a minimum, your design will need to include SDC timing constraints for all clocks, and PDC constraints for all boundary pin placements prior to generating a bitstream.

Place and Route Netlist Files

In simple design flows, there is only 1 Place and Route Netlist file enabled for the active implementation: the synthesized gate-level Verilog netlist output of the Synthesis tool. If you are using the ACE Synthesis flow steps to generate the netlist, then you do not need to manually add the synthesized gate-level netlist to your ACE project. The Run Synthesis flow step will add it to your ACE project automatically upon successful design synthesis. If you are disabling the Synthesis flow steps and running Synthesis outside of ACE, then you will need to manually add the synthesized gate-level netlist to your ACE project.

In more complex design flows, there may be multiple Place and Route Netlist files enabled for the active implementation. For example, you might need to add some exported Partition blackbox netlist files, or encrypted gate level netlists for 3rd party IP cores.

Place and Route Netlist files which have been added to the ACE project can be enabled or disabled per [Implementation \(page 229\)](#). See also: [enable_project_source_file \(page 635\)](#) and [disable_project_source_file \(page 627\)](#)

When running the [Multiprocess \(page 308\)](#) flow with the Synthesis flow step enabled, multiple gate level netlists will be synthesized and added to the ACE project (one per Implementation). Keep in mind, that as you are updating your ACE project and dealing with multiple gate level netlists for the same design, you must have only 1 gate level netlist file "enabled" per implementation. Otherwise you will see duplicate module definition errors during the Run Prepare flow step.

Note

All Place and Route Netlist files are parsed in the order specified in the ACE project file. Place and Route Netlist files that are added to the project first are parsed first, and likewise, Place and Route Netlist files which are added to the project last are parsed last. If there is any order dependency between Place and Route Netlist module definitions, please make sure to add the Place and Route Netlist files in the correct order for parsing in ACE. See also: [move_project_source_file \(page 669\)](#)

At a minimum, your design will need to include SDC timing constraints for all clocks, and PDC constraints for all boundary pin placements prior to generating a bitstream.

Simulation Testbench Files

The project source Simulation Testbench files represent the Testbench for the end user design logic, not the end user design logic itself (which must be added as RTL source files). The list of Simulation Testbench files specified in the ACE project is used to generate the file list for Simulation (in addition to the RTL source files). Simulation Testbench RTL files are always compiled after the RTL source files in the simulator.

Simulation Testbench files can include RTL (Verilog/SystemVerilog/VHDL) and any other file type supported by the simulator.

Filelist (or simulator command) files with the *.f extension can be added to the ACE project, and will be passed to the simulator tool with the -f option.

Memory initialization files (mem_init_files), test vector files, or any other type of file referenced in the testbench or user design RTL may be added to the ACE project as Simulation Testbench files. When ACE runs the Simulation flow step, it first creates a new simulation run/output directory in <project_output_path>/<impl>/<sim_step>/<tool> (for example C:/test_project/output/impl_1/rtl/riviera). ACE will generate the necessary simulator input files, and copy all the mem_init_files, test vector files, etc into the new simulator run directory. ACE will change working directories into this simulator run directory to run the simulation. This way, all the mem_init_files, test vector files, etc are available in the same local directory the simulator tool is running in, which avoids several relative file path issues.

Note

Adding Simulation Testbench source files to your ACE project is optional if you plan to disable the Simulation flow steps and run Simulation outside of ACE.

Port Mapping Files

The I/O Designer [I/O Pin Assignment View \(page 64\)](#) and [I/O Core Pin Assignment View \(page 65\)](#) provide the option to remap port names from the values used in the IP editors to the name desired for use in the user RTL (top-level port list). This action creates an "alias" to be used in all files created when the **Generate IORing Design Files** button in the IO Designer is clicked (or similarly, calling the [generate_ioring_design_files \(page 641\)](#) Tcl command).

These remapped names are placed in the board.acxpm (remapped pin ports) and core.acxpm (remapped core pin ports) files, stored in the project directory (the directory containing the *.acxprj [Project file \(page 222\)](#)). These files are later read when generating the design files and are not needed within the design/netlist, nor are they explicitly named in the *.acxprj [Project file \(page 222\)](#).

These are simple text files, with the contents automatically managed by the [I/O Pin Assignment View \(page 64\)](#) and [I/O Core Pin Assignment View \(page 65\)](#). Each line in the port mapping file contains a key/value pair, of the form:

```
original_port_name=remapped_name_alias
```

See [Creating an IP Configuration \(page 336\)](#) for more details on handling IP configurations and the I/O Designer.

Implementations

A [Project \(page 222\)](#) may have multiple implementations. Each implementation contains the set of flow options (also called implementation options) configuring the run of that project through the [Flow \(page 234\)](#), and the flow outputs for this particular configuration. With this capability, the same design (RTL/Netlist) can be implemented (run through the flow) with different sets of synthesis options, placement and routing optimizations, or even timing constraints, just by creating multiple implementations for the same project.

Each implementation is associated with an implementation output directory located under the Project Output Directory. The Project Output Directory is configurable with ACE project-level options, and can be set to:

1. The default (the project source directory where the ACE [Project File \(page 222\)](#) is located), or
2. A custom project output path

Implementation directories are named with the implementation name and contain flow output files. [Output Files \(page 231\)](#) are divided into three main sub-directories under the implementation directory: `syn` (for Synthesis output), `pnr` (for Place and Route output), and `sim` (for Simulator output). The output directories contain reports, log files, and output files that are intended to be consumed by other tools later in the flow, such as netlists for simulation or the FPGA bitstream for programming.

Implementation definitions are *not* individually saved to their own files. Instead they are stored as part of the [Project File \(page 222\)](#). In the GUI, project implementations can be browsed in the [Projects View \(page 117\)](#). Selecting an implementation [activates \(page 229\)](#) it and displays its implementation options in the [Options View \(page 96\)](#).

When an implementation has been run through the flow, the state of the database (netlist, constraints, placement, and routing data) may be saved to an `.acxdb` file (see [Saving Implementations \(page 302\)](#)). Implementations may later be restored from previously saved `.acxdb` files (see [Restoring Implementations \(page 302\)](#)).

Active Project and Implementation

The **active** project is the project containing the **active** implementation in the current tool session. The active implementation is the project implementation on which flow and project management commands are operating. **Only one** implementation can be active at a time. The active state applies across all projects and only in the context of the current tool session.

In the ACE GUI, the active project name, active implementation name, and target device name are all shown on the ACE titlebar in the format "Project -> Implementation (Device)". Additionally, within the [Projects View \(page 117\)](#) tree, the active project and its active implementation are both shown in a bold font.

Project and Implementation Options

There are 2 types of Options that can be configured for a given ACE project:

1. Project Options: see [set_project_option \(page 714\)](#) and [get_project_option \(page 655\)](#)
2. Implementation Options: see [set_impl_option \(page 711\)](#) and [get_impl_option \(page 648\)](#)

Project Options

Project Options apply globally across all implementations within a Project. These options do not change from implementation to implementation, and typically govern the overall structure of the project and flow. Some examples are **Target Device**, **HDL Include Path**, and **Flow Mode**.

Project options may have parent-child dependencies with other Project Options or Implementation Options. When changing Project Options, keep in mind that changing a parent option, like **Target Device**, will cause downstream child options to be enabled/disabled, have their valid range of values updated, or may change the child option's value. To query whether a child option is currently enabled (usable) or disabled (hidden and not applicable), see: [get_project_option_is_supported_and_enabled](#) (page 656) and [get_impl_option_is_supported_and_enabled](#) (page 649).

If a change to a Project Option impacts a child Implementation option, that Implementation option will be updated for **all** Implementations in the project (not just the active implementation).

Both Project Options and Implementation Options are displayed in the [Options View](#) (page 96) .

Implementation Options

There are a wide variety of configurable implementation options which alter the implementation of the end user design in Synthesis and Place and Route as it moves through the [flow](#) (page 234). The the most-commonly used option settings are displayed in the [Options View](#) (page 96) for the current [Active Project and Implementation](#) (page 229). Within the Options view, implementation options are grouped in categories, which loosely correlate to which flow step the option affects. Changing the value of an option causes the current results of its associated flow step (if any) to become invalid or cleared, and that flow step (and all later flow steps) must be rerun, making use of the newly-changed option.

An [Implementation Options Report](#) (page 257) of all available project and implementation options may be generated via the Tcl command [report_impl_options](#) (page 679). This command may also be used to compare the current options configuration of an implementation with the default values for all options.

The values of implementation options may be set with the Tcl command [set_impl_option](#) (page 711), or reset back to the default values with [reset_impl_option](#) (page 686).

Option Sets

Because some implementation options have a large impact upon runtime, and because the QoR benefits of these implementation options may vary significantly by design (often a QoR gain, but sometimes a slight QoR loss), many of the performance-related implementation options are disabled by default for newly created projects and implementations.

Achronix QoR experts have compiled subsets of implementation options known to optimize a wide variety of design types based upon design details. These "option sets" are made available (with description) to users in the [Multiprocess View](#) (page 73), and through that view, may be used to generate new implementations with the indicated implementation options enabled.

Each Option Set shown in the Multiprocess View consists of override values for a small subset of all the implementation options. These overriding values are applied to newly generated implementations over the existing implementation option values inherited from a user-selected template implementation.

It is worth repeating that the Option Sets do not contain a complete assignment of all the implementation options. Each Option Set only contains a small subset of option values, which override the implementation options inherited from the template implementation. The overriding implementation options in each set are **subsets** of the entire set

of QoR oriented implementation options. They only change *some* of the implementation options, and all the rest of the values are inherited from the template implementation. The Option Sets only enable performance-related implementation options, and (currently) never disable any already-enabled implementation options. So each generated implementation starts with the exact same implementation options as the template implementation, and then just the few implementation options named in the Option Set description are overwritten with the described values.

Achronix broke up the Option Sets into small granular chunks because of QoR/runtime trade-offs. Some of the options have a large runtime cost, and on some designs, there is a minimal performance gain. Based upon the observed runtimes reported in the [Multiprocess Summary Report \(page 255\)](#), hours of runtime may be saved while only losing 0.01% frequency by using one Option Set over another Option Set as a design is iterated.

Note

Currently, the Option Set overrides only *enable* optimization-oriented implementation options, not disable them. Thus, if the implementation options in the template implementation are already the same values as those in the Option Set, the results from the two implementations (the implementation generated from the Option Set, and the template implementation) are identical.

It is expected that among all the Option Sets, at least one can be found that provides the necessary QoR gain for an acceptable runtime impact, allowing the fastest possible design iteration. See the [Multiprocess View \(page 73\)](#) and [Attempting Likely Optimizations Using Option Sets \(page 392\)](#) for more information.

Output Files

Output files are generated for project [implementations \(page 229\)](#) by running certain steps in the [Flow \(page 234\)](#). By default, output files are automatically written to the project implementation directory.

Each implementation is associated with an implementation output directory located under the Project Output Directory. The Project Output Directory is configurable with ACE project-level options, and can be set to:

1. The default (the project source directory where the ACE [Project File \(page 222\)](#) is located), or
2. A custom project output path

Implementation directories are named with the implementation name and contain flow output files. Output Files are divided into three main sub-directories under the implementation directory: `syn` (for Synthesis output), `pnr` (for Place and Route output), and `sim` (for Simulator output). The output directories contain reports, log files, and output files that are intended to be consumed by other tools later in the flow, such as netlists for simulation or the FPGA bitstream for programming.

In the GUI, output files can be browsed in the [Projects view \(page 117\)](#) under their implementation and viewed in the editor area by double clicking the file name in the Projects view.

Log Files

A number of log files are automatically generated while ACE is running. They include the following:

- ACE Session Log
- Implementation Log
- Multiprocess Log
- SnapShot Log
- ACE GUI Log

- Synthesis Log
- Simulation Log

The contents of these logs are typically the series of Tcl commands and resulting return values occurring during the execution of ACE. Achronix support may sometimes request one or more of these log files to assist in requested support efforts.

ACE Session Log

Every time ACE is started, a new ACE session log is created in the directory `<user_home_dir>/.achronix/`. This file is named `ace_<date_timestamp>.log`, where `<date_timestamp>` is `<year>_<month>_<day>_<hour>_<minute>_<second>`, in 24-hour format. For example, if ACE was started in Linux with a username of `example_user`, on January 11th of 2023 at 2:34:56 PM, the complete log file name would be:

```
/home/example_user/.achronix/ace_2023_01_11_14_34_56.log
```

ACE session log messages are also sent to the [Tcl Console View \(page 142\)](#). For more information, see [Viewing the ACE Log File \(page 335\)](#).

Implementation Log

In addition to the session log, each [project \(page 222\)](#) [implementation \(page 229\)](#) has a log maintained for the complete life of the implementation. All changes to the implementation, including running the [Flow \(page 234\)](#) for that implementation, are appended to the implementation log. This log is stored in the directory `<project_dir>/<impl_name>/pnr/log/impl.log`.

Multiprocess Log

Unlike normal flow executions, implementation runs initiated from the [Multiprocess View \(page 73\)](#) do not have their log information appended to the ACE Session Log. The reason is because multiple processes would be appending info to the log file simultaneously, leaving log entries interleaved in an unreadable mess. Instead, each implementation executed in the background creates a new log file named `multiprocessImpl.log` in the log directory for that implementation, overwriting any prior multiprocess log created for that implementation. Each implementation executed via the Multiprocess View does still append information to its lifetime implementation log file.

When running Multiprocess with an external job submission system (i.e., the example GridEngine default configuration), two additional log files may be created:

- The `jobExecution.log` contains a raw (unfiltered) copy of the implementation log for that multiprocess session, plus any additional info that the user custom job submission system might choose to inject. This file only exists if the (optional) job submission logging functionality is configured appropriately.
- The `jobScheduler.log` is only used during [Multiprocess Batch Mode \(page 318\)](#) (when in GUI mode instead of batch mode, similar info is captured in the individual implementation feedback tabs within the Multiprocess View). This file captures the standard output and error streams from the job submission system itself. This log file is particularly useful when initially configuring ACE to work with the user job submission system as it captures submission configuration errors (such as typos in job queue names).

Snapshot Log

The Snapshot log file is discussed in [Collecting Samples of the User Design \(page 376\)](#).

ACE GUI Log

On rare occasions, Achronix Support may request the ACE GUI Log. This log file may be found by selecting: **Help** → **About Ace** → **Installation Details** → **Configuration** → **View Error Log**, which opens the GUI log in a non-ACE text file editor or HTML browser. The editor/browser typically reports the full path of the opened file.

Synthesis Log

When the Synthesis flow step is run, a separate synthesis log files is generated. This log is stored in the directory `<project_dir>/<impl_name>/syn/rev_acx/<project_name>_<impl_name>.srr`.

Simulation Log

When the Simulation flow step is run, a separate simulation log files is generated. This log is stored in the directory `<project_dir>/<impl_name>/sim/<project_name>-<tool>-<sim_step>.log`.

Flow

The flow is the set of steps that must be run to complete a design in ACE. These steps are listed, in order, within the **Flow view** (page 53). The current Flow Mode implementation option (selected in the **Options view** (page 96)) affects which Flow Steps may be executed.

A flow can only be run on a single **project** (page 222), and within that project a single **implementation** (page 229) at a time (these are the **active project and implementation** (page 229)) during an interactive ACE session.

To run multiple implementations from the same project through the flow simultaneously, use the **Multiprocess view** (page 73).

To run multiple separate projects through the flow simultaneously, multiple sessions of ACE must be run.

The flow can be customized by creating **custom flow steps** (page 330).

Additional details may be found in the section, **Running the Flow** (page 306).

- **Flow Steps** (page 234)
 - **IP Configuration Steps** (page 237)
 - **RTL Simulation Steps** (page 237)
 - **Synthesis Steps** (page 237)
 - **Place and Route Steps** (page 238)
 - **Design Completion Steps** (page 240)
 - **FPGA Programming Steps** (page 241)
- **Flow Status** (page 242)
- **Flow Mode** (page 242)

Flow Steps

The **Flow** (page 234) is composed of a series of flow steps, each representing a command operating on the design in the **Active Project and Implementation** (page 229). Some flow steps are required (such as preparing the design), and some are optional (such as writing out a netlist for simulation). The user must enable or disable any optional flow steps prior to running the flow (see: **enable_flow_step** (page 634) and **disable_flow_step** (page 626)).

Flow steps are order-dependent. The default order of flow steps is displayed in the ACE GUI **Flow View** (page 53), with flow steps grouped into categories for organizational purposes. Users can customize the flow to add user-defined flow steps (see: **create_flow_step** (page 621)).

The complete flow can be run from start to finish, executing all required and enabled optional flow steps in-order by simply clicking the **Run** (▶) toolbar button in the **Flow View** (page 53), or by call the **run** (page 689) command.

Individual flow steps can be run by double-clicking on the flow step in the **Flow View** (page 53), or by calling the **run** (page 689) command with the `-step <step_id>` option. Attempting to run a flow step out of order will cause one of the following behaviors:

1. You are trying to run a flow step that is further down the flow from the last completed flow step, and there are incomplete flow steps in between. In this case, ACE will search back to the last completed flow step (or the beginning of the flow if no completed flow steps), then will walk forward down the list of flow steps in-order, and will automatically run any incomplete required (or enabled optional) pre-requisite flow steps. If all the pre-requisite flow steps complete successfully, the flow step you specified with the `-step <step_id>` option will be run last.

- a. For example: Let's say you have run **Run Prepare** and it has completed successfully. **Run Prepare** was the last completed flow step. Now, you double-click on the **Run Route** flow step without first running the required **Run Place** flow step. All of the other optional flow steps in between are disabled. ACE will then automatically run the **Run Place** flow step, and if it is successful, will continue to to run the desired **Run Route** flow step.
2. You are trying to run a flow step that is further up the flow from the last completed flow step. This could be because you want to re-run a previously completed flow step, or because you want to enable a previous optional flow step and then go back and run it before moving on. In this case, ACE will clear the status (mark as Incomplete) of all the flow steps from the targeted flow step onward, and then will attempt to run the targeted flow step.
 - a. For example: Let's say you have completed running all the flow steps up through the **Run Route** flow step. The **Run Prepare**, **Run Place**, and **Run Route** flow steps are all marked as complete. But now you want to change your PDC constraints source file and re-run only **Run Prepare**. So you double-click on **Run Prepare** to run it. This will cause **Run Prepare**, **Run Place**, and **Run Route** to all be marked as Incomplete. Then ACE will start running the **Run Prepare** step and will stop after it finishes.

The implementation option for **Flow Mode** (page 242) (typically configured through the **Options View** (page 96)) affects which flow steps can be executed.

Table 120 • Achronix Default Flow Steps and IDs

Name	ID
IP Configuration	ip_configuration
– Generate All IP Design Files	generate_all_ip_design_files
RTL Simulation	rtl_simulation
– Run RTL Simulation	run_simulation_rtl
Synthesis	synthesis
– Run Synthesis	run_synthesis
– Run Gate-level Netlist Simulation	run_simulation_gate
Place and Route	place_and_route
– Run Prepare	run_prepare
– Run Place	run_place
– Run Post-Placement Timing Analysis	report_timing_placed
– Run Route	run_route

Name	ID
- Run Post-Route Timing Analysis	report_timing_routed
- Generate Post-Route Simulation Netlist	write_netlist_routed
- Run Post-Route Netlist Simulation	run_simulation_routed
Design Completion	design_completion
- Post-Process Design	post_process
- Run Final DRC Checks	final_drc_checks
- Run Sign-off Timing Analysis	report_timing_final
- Generate Final Reports	write_reports_final
- Generate Final Simulation Netlist	write_netlist_final
- Run Final Netlist Simulation	run_simulation_final
FPGA Programming	fpga_program
- Generate Bitstream	write_bitstream
- FPGA Download	fpga_download

Table Notes

- All flow step IDs can be executed at the ACE GUI Tcl console (see [Tcl Console View \(page 142\)](#)) or as part of the user Tcl script that can be invoked when [running ACE \(page 282\)](#) in batch mode. The following Tcl command allows executing the various flow steps IDs listed:

```
run (page 689) [-step <string>] [-stop_at_step
<string>] [-resume] [-ic <string>]
```

- Because advanced users are allowed to create their own flow steps ([create_flow_step \(page 621\)](#)), this list may be a subset of the flow steps available to users. To see a complete list of current flow step IDs, use the Tcl command [get_flow_steps \(page 647\)](#).

IP Configuration Steps

Generate All IP Design Files (Optional)

This is the first step in the flow, as it generates IP design files which may contain RTL files that define IP modules for the user to instantiate in the end user design RTL. This step generates IP design files for each Core fabric Soft IP configuration (*.acxip file) in the ACE project (both Speedcore eFPGAs and Speedster FPGAs), and also generates I/O Ring design files when applicable (for Speedster FPGAs). This step is not run by default when **Run Flow** is executed.

This flow step has the id `generate_all_ip_design_files` for Tcl commands.

RTL Simulation Steps

Run RTL Simulation (Optional)

This flow step runs RTL simulation using the configuration set in the ACE project options. There are 2 simulation flows: Default and Custom. Select Default to use the built-in simulation flow scripts. Select Custom to call your own custom simulation TCL proc, which is defined in your ACE_INIT_SCRIPT setup. The actual simulation is performed in a 3rd party simulator. The ACE default simulation flow generates simulation project files targeting the given simulation tool, passes in simulator command arguments, and runs the simulator using the testbench files provided by the end user in the ACE project. The simulator HDL include path and HDL Defines are set in the Project Options section of the ACE project options, which apply to both Synthesis and Simulation flow steps. This step is not run by default when **Run Flow** is executed.

This flow step has the id `run_simulation_rtl` for Tcl commands.

Synthesis Steps

Run Synthesis (Optional)

This flow step runs Synthesis using the configuration set in the ACE project options. There are 2 main synthesis flows:

1. Using ACE as the master of the Synthesis project (the `syn_use_default_project` project option is set to 1), or,
2. Using Synplify Pro as the master of the Synthesis project file outside of ACE (the `syn_use_default_project` project option is set to 0) .

When the `syn_use_default_project` project option is set to 1, the source Synthesis project file will be automatically generated from the ACE project settings and managed by ACE in the Project->Output->(impl)->syn directory. When this option is disabled, the user must specify the full file path to the synthesis project file, which then gets copied to the Project->Output->(impl)->syn directory when the `run_synthesis` flow step is run.

The synthesis flow #1 is the simplest approach and is useful if you want a push-button way to run Synthesis from within ACE using a consolidated ACE project file that specifies the RTL and Synthesis Constraints source files.

The synthesis flow #2 is useful if you want to use the Synplify GUI to manage the Synplify project file and use it for interactive debug. In this flow, you do not need to specify RTL source files or Synthesis Constraints source files in your ACE project.

While the actual synthesis is performed in a 3rd party tool, the ACE synthesis flow step generates synthesis project files targeting the given synthesis tool, passes in synthesis options, and runs synthesis on the end user design in the

ACE project. The simulator HDL include path and HDL Defines are set the the Project Options section of the ACE project options, which apply to both Synthesis and Simulation flow steps. This step is run by default when **Run Flow** is executed.

This flow step has the id `run_synthesis` for Tcl commands.

Run Gate-Level Netlist Simulation (Optional)

This flow step runs gate-level netlist simulation using the configuration set in the ACE project options. There are 2 simulation flows: Default and Custom. Select Default to use the built-in simulation flow scripts. Select Custom to call your own custom simulation TCL proc, which is defined in your ACE_INIT_SCRIPT setup. The actual simulation is performed in a 3rd party simulator. The ACE default simulation flow generates simulation project files targeting the given simulation tool, passes in simulator command arguments, and runs the simulator using the testbench files provided by the end user in the ACE project. The simulator HDL include path and HDL Defines are set the the Project Options section of the ACE project options, which apply to both Synthesis and Simulation flow steps. This step is not run by default when **Run Flow** is executed.

This flow step has the id `run_simulation_gate` for Tcl commands.

Place and Route Steps

Run Prepare

The first flow step required for any design is **Run Prepare**. This step (in order):

1. Clears all previously loaded netlists and constraints.
2. Loads and compiles the device.
3. Loads all the design files for the active implementation into ACE.
4. Runs design checks.
5. Transmutes the design into an Achronix design.

The active project is saved to disk automatically when this step is successfully completed. In addition, this step automatically generates a pin assignment and an utilization report.

When the active implementation is prepared, the design is ready to be placed or analyzed for timing results. An encrypted Verilog netlist can also be generated for the prepared implementation for simulation. I/O pre-placement can also be performed when the design is prepared (see [Pre-Placing a Design \(page 346\)](#)).

This flow step has the id `run_prepare` for Tcl commands.

Run Place

After **Run Prepare** has successfully completed on an implementation, the **Run Place** step must be run in order to place the design. If place and route has already been run on this implementation, this step may be skipped, and the place and route data from the previous run may be loaded by using the **File** → **Load Place and Route Data** menu option. When the design is successfully placed, the encrypted placement data is stored to disk and is ready to be loaded again later.

This flow step has the id `run_place` for Tcl commands.

Run Post-Placement Timing Analysis (Optional)

After **Run Place** has successfully completed on an implementation, the **Run Post-Placement Timing Analysis** step can be run. This step generates and writes a timing report file for the placed design, without requiring all final DRC checks to pass. This timing report will only be run against the single active temperature corner, even if multiple temperature corner reporting has been requested. The generated report is automatically displayed in the editor area upon successful completion. This step is not run by default when **Run Flow** is executed.

This flow step has the id `report_timing_placed` for Tcl commands.

Run Route

After **Run Place** has successfully completed on an implementation, the **Run Route** step must be run in order to route the design. If place and route has already been run on this implementation, this step may be skipped, and the place and route data from the previous run may be loaded by using the **File** → **Load Place and Route Data** menu option. When the design is successfully routed, the encrypted placement data is stored to disk and is ready to be loaded again later.

This flow step has the id `run_route` for Tcl commands.

Run Post-Route Timing Analysis (Optional)

After **Run Place** and **Run Route** have successfully completed on an implementation, the **Run Post-Route Timing Analysis** step can be run. This step generates and writes a timing report file for the placed and routed design, without requiring all final DRC checks to pass. This timing report will only be run against the single active temperature corner, even if multiple temperature corner reporting has been requested. The generated report is automatically displayed in the editor area upon successful completion. This step is not run by default when **Run Flow** is executed.

This flow step has the id `report_timing_routed` for Tcl commands.

Generate Post-Route Simulation Netlist (Optional)

After **Run Route** has successfully completed on an implementation, the **Generate Post-Route Simulation Netlist** step can be run. This step generates and writes an encrypted, post-place-and-route Verilog simulation netlist file from the design. This netlist may be used to simulate the post-place-and-route design. This step is not run by default when **Run Flow** is executed.

This flow step has the id `write_netlist_routed` for Tcl commands.

Run Post-Route Netlist Simulation (Optional)

This flow step runs post-route simulation using the configuration set in the ACE project options. There are 2 simulation flows: Default and Custom. Select Default to use the built-in simulation flow scripts. Select Custom to call your own custom simulation TCL proc, which is defined in your `ACE_INIT_SCRIPT` setup. The actual simulation is performed in a 3rd party simulator. The ACE default simulation flow generates simulation project files targeting the given simulation tool, passes in simulator command arguments, and runs the simulator using the testbench files provided by the end user in the ACE project. The simulator HDL include path and HDL Defines are set the the Project Options section of the ACE project options, which apply to both Synthesis and Simulation flow steps. This step is not run by default when **Run Flow** is executed.

This flow step has the id `run_simulation_routed` for Tcl commands.

Design Completion Steps

Caution!

All **Flow Steps** (page 234) under the Design Completion category are skipped by default when the flow mode is set to **Evaluation**. See **Flow Mode** (page 242) for more details.

Post-Process Design

After **Run Place** and **Run Route** have successfully completed (or a .acxdb file containing place and route data has been loaded) on an implementation, the **Post-Process Design** step must be run. This step post-processes the design by inserting Achronix-specific technology (such as reset, compensation block, and vmode insertion) that relies on final placement and routing information. This step should not affect timing results.

This flow step has the id `post_process` for Tcl commands.

Run Final DRC Checks

After **Post-Process Design** has successfully completed on an implementation, the **Run Final DRC Checks** step must be run. This step performs all final DRC checks in order to ensure that the bitstream, final timing, and final simulation netlist can be generated without errors. If a design fails final DRC checks, a **Post-Route** timing report can still be generated for experimental purposes. However, no bitstream may be generated to run the design on the hardware unless all final DRC checks pass.

This flow step has the id `final_drc_checks` for Tcl commands.

Run Sign-off Timing Analysis (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Run Sign-Off Timing Analysis** step can be run. This step generates and writes a final sign-off timing report file for the placed and routed design, after all final DRC checks have passed. Unlike earlier timing analysis flow steps, the timing report generated by this flow step may be run against all temperature corners if **Timing Across All Temperature Corners** (page 266) has been enabled. Each generated report is automatically displayed in the editor area upon successful completion. This step is run by default when **Run Flow** is executed (and the Flow Mode is not set to **Evaluation**).

This flow step has the id `report_timing_final` for Tcl commands.

Generate Final Reports (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Generate Final Reports** step can be run. This step generates various report files, including clocks, pins, power, etc. Implementation options are used to control which report files are generated. The generated report is automatically displayed in the editor area upon successful completion. This step is run by default when **Run Flow** is executed.

This flow step has the id `write_reports_final` for Tcl commands.

Generate Final Simulation Netlist (Optional)

After **Run Final DRC Checks** has successfully completed on an implementation, the **Generate Final Simulation Netlist** step can be run. This step generates and writes an encrypted, post-place-and-route Verilog simulation netlist file from the final DRC-free design. This netlist may be used to simulate the post-place-and-route design. This step is not run by default when **Run Flow** is executed.

This flow step has the id `write_netlist_final` for Tcl commands.

Run Final Netlist Simulation (Optional)

This flow step runs final netlist simulation using the configuration set in the ACE project options. There are 2 simulation flows: Default and Custom. Select Default to use the built-in simulation flow scripts. Select Custom to call your own custom simulation TCL proc, which is defined in your ACE_INIT_SCRIPT setup. The actual simulation is performed in a 3rd party simulator. The ACE default simulation flow generates simulation project files targeting the given simulation tool, passes in simulator command arguments, and runs the simulator using the testbench files provided by the end user in the ACE project. The simulator HDL include path and HDL Defines are set the the Project Options section of the ACE project options, which apply to both Synthesis and Simulation flow steps. This step is not run by default when **Run Flow** is executed.

This flow step has the id `run_simulation_routed` for Tcl commands.

FPGA Programming Steps

⚠ Caution!

The **Generate Bitstream** flow step fails if attempted in **Evaluation** flow mode. Additionally, all **Flow Steps** (page 234) under the FPGA Programming category are skipped by default when the flow mode is set to **Evaluation**. See **Flow Mode** (page 242) for more details.

Generate Bitstream

After a design is placed and routed, the **Generate Bitstream** step can be run. This step generates a bitstream (*.hex file) for the current implementation based on the settings in the **Options view** (page 96) (see the Options settings for Bitstream Generation). This step is run by default when **Run Flow** is executed. The flow mode must be set to **Normal** (or **Strict**) before bitstream generation can complete successfully (while it typically produces much shorter flow runtimes, the **Evaluation** flow mode relaxes the DRCs too much to produce a reliable bitstream).

This flow step has the id `write_bitstream` for Tcl commands.

FPGA Download

After the bitstream is generated, it is ready for downloading to the FPGA via either the Achronix Bitporter 2 pod or an FTDI FT2232H (or FT4232H) device as specified under the Options view settings (see the Options settings for FPGA Download). This step is not run by default when **Run Flow** is executed.

This flow step has the id `fpga_download` for Tcl commands.

A bitstream can also be downloaded to the FPGA via the **Download view** (page 40) (see **Programming a Device using JTAG in the Download View** (page 390))

After downloading the bitstream, the JTAG connection status is retained, meaning that if the JTAG connection is open prior to programming, it will stay open, and if the JTAG connection is closed prior to programming, it will close the connection when programming completes.













For more details, refer to the *Configuration User Guide* (UG004).

Flow Status

Flow Steps (page 234) each have a status associated with them. The current status of each flow step for the **Active Project and Implementation** (page 229) can be seen in the GUI in the **Flow View** (page 53).

Each step can be in one of the following flow states:

Table 121 • Flow State Icons

State	Flow Category	Flow Step
Incomplete		
Running		
Complete		
Disabled		
Error		
Complete (but out of sync with source files)		

Be aware that changing or **Configuring Project and Implementation Options** (page 304) which affect a flow step can cause the status of a flow step to be reset back to Incomplete.

See the concepts for the **Flow** (page 234) and **Flow View** (page 53), as well as the task for **Running the Flow** (page 306) for more details.

Flow Mode

The flow mode is managed as an **Implementation** (page 229) Option, typically through the **Options View** (page 96).

The chosen flow mode determines which DRCs are executed at different points in the **Flow** (page 234), affects timing analysis, and in some configurations prohibits the final **Flow Steps** (page 234) from executing successfully.

- **Evaluation** – this mode ignores non-fatal DRCs as long as possible, allows IO Virtualization, and to get a post-route timing report quickly, ignores missing SDC constraints and only runs timing against a single target temperature corner, even if timing across all corners has been requested. This mode allows iterating more quickly during preliminary or early design stages.
- **Normal** – this flow mode enforces all DRC checks prior to generating a bitstream. Some checks are flagged as warnings early on in the flow to provide an opportunity to correct the problems (i.e., fixing the placement of I/Os). These same checks may change to report an error during final DRC checks. This mode should be used when developing a real design for a product, and enables bitstream generation.
- **Strict** – this mode is similar to **Normal** flow mode, but to reduce runtime, strictly enforces all DRC checks, erroring out as early in the flow as possible. This more restrictive mode should be used during the later, more mature design iterations.

When in **Evaluation** flow mode, the **Run Flow** and **Re-run Flow** actions in the **Flow View** (page 53) stop after the **Place and Route** flow step category is completed. By default, the flows steps under **Design Completion** and **FPGA Programming** are not run unless the implementation is in **Normal** or **Strict** flow mode.

Note

The **Generate Bitstream** flow step fails if attempted while in **Evaluation** flow mode. Bitstream generation requires **Normal** or **Strict** flow mode, so that ACE may ensure all DRCs have passed.

Some examples of error checks in **Strict** flow mode, applicable to Speedcore devices, are as follows:

1. If any Speedcore boundary pins (IPIN/OPIN/CLK_IPIN/CLK_OPIN) are not explicitly instantiated in the user design RTL or PDC files, ACE errors out in the Run Prepare flow step.
2. If any Speedcore boundary pins (IPIN/OPIN/CLK_IPIN/CLK_OPIN) are not explicitly placed with "fixed" placement in the project PDC files, ACE errors out at the beginning of the Run Place flow step.

In other flow modes, these checks do not happen until Final DRC Checks prior to bitstream generation.

Reports

ACE generates a number of reports to provide information on how user designs are being handled in the selected Achronix device. These reports are meant to assist in making design decisions.

Each of the listed reports is generated in HTML format by default, and with the noted Tcl command, each report can optionally (unless otherwise noted) be generated in plain text format, or in CSV format for easy import into spreadsheet programs. As soon as the reports are generated, they are opened for viewing within the ACE Editor area.

Utilization Report

The Utilization Report shows a summary and details of the utilization of the device resources for the current design.

This report is automatically generated as part of the **Run Prepare flow step** (page 234).

To generate this report manually, see the `report_utilization` (page 686) Tcl command.

Pin Assignment Report

The Pin Assignment Report shows detailed information on each of the user design top-level ports, including placement and configuration details.

Typically, the report is automatically generated by the **Flow** (page 234) at multiple times (as part of the **Run Prepare**, **Run Place**, and **Post-Process Design flow steps** (page 234)).

To generate this report manually, see the `report_pins` (page 682) Tcl command.

Clock Report

The Clock Report shows all clocks used in the design, their frequencies/periods, their relationships, and their constraints. Related information regarding device Clock Regions is also included in the report.

The report is automatically generated by the **Flow** (page 234) at multiple stages of the flow.

To generate this report manually, see the `report_clocks` (page 678) Tcl command.

Timing Report

The Timing Report provides details on how well the current design is meeting timing on the selected device.

Timing analysis can be performed at several stages in the **Flow** (page 234), each stage generating a different report. If the report is generated before the design has been placed or routed, placement and/or routing will be estimated.

This report is automatically generated by the flow during any of the (optional) **Run ... Timing Analysis Flow Steps** (page 234).

To generate this report manually, see the `run_timing_analysis` (page 702) Tcl command.

If **Timing Across All Temperature Corners** (page 266) is enabled through the relevant implementation options, this will only affect the (final) timing report, generated during the **Run Sign-Off Timing Analysis** flow step, and only when the **Flow Mode** (page 242) is set to Normal or Strict mode. All other timing reports will only include a single temperature

corner. When multiple corner reporting occurs, a separate timing report is generated for each corner, with the file name of the report noting the corner being reported.

i Generating timing reports for multiple temperature corners is only possible when in the Normal or Strict flow modes, and only for the final (sign-off) timing analysis flow step. In all other cases, only a single temperature corner will be reported.

When multiple-corner reporting occurs, each corner will generate a separate report file, with the report file's name noting the corner being analyzed/reported.

See also: [Multiprocess Summary Report \(page 255\)](#)

Report Content

The report contains a Summary section and a Details section.

The Summary section contains three tables. There is a table for Critical Setup (max) Timing Paths, one for Critical Hold (min) Timing Paths, and one for the resulting Clock Frequencies. Each summary section table contains a single row for each Clock/Group, showing the most critical path for that Clock/Group.

The Details section contains a configurable maximum number of critical setup paths and critical hold paths for each Clock/Group, and each of those critical paths includes a configurable maximum number of worst paths for the critical path endpoint.

The number of critical paths and worst paths are [Implementation Options \(page 0\)](#) configured in the [Options View \(page 96\)](#) under "Timing Analysis".

Routing Report

The Routing Report collects a number of routing-related statistics and any related errors into an easily readable report format.

This report is automatically generated by the [Flow \(page 234\)](#) during the **Run Route** and **Post-Process Design flow steps (page 234)**.

To generate this report manually, see the [report_routing \(page 685\)](#) Tcl command.

Partitions Report

The Partitions Report collects a number of partition-related statistics into an easily readable report format.

This report is automatically generated by the [Flow \(page 234\)](#) (and opened in the GUI) during the **Run Prepare flow step (page 234)** when [Using Incremental Compilation \(Partitions\) \(page 404\)](#).

To generate this report manually, see the [report_partitions \(page 680\)](#) Tcl command.

Power Dissipation Report

The Power Dissipation report shows an estimate for the amount of power, in Watts, dissipated by the current design on the selected Achronix device under normal operating conditions. This report is typically generated after place-and-route, and therefore includes the effects of all transistor and wiring parasitics in the power calculation. This is in

contrast to the Speedster7t Power Estimator and Speedcore Power Estimator tools, which are used early in the design process, prior to actual design activity.

Total power dissipation is reported so that thermal requirements can be estimated for the user design. Total power is also broken down into several different categories, such as power dissipated for each power rail, for each cell type, and instance power vs. interconnect power. Therefore the report can also be used for the design and sizing of the power supplies, and the selection of hard IP blocks such as DDR4 vs. GDDR6.

This User Guide provides an example power dissipation report, with an explanation of each section of the report, as well as detailed information about how power dissipation is calculated, and how to generate the report.

Example Power Dissipation Report

The Power Dissipation Report consists of eight sections, each of which presents the power dissipation from a different perspective.

- Report Header
- Power Summary
- Hard IP Power
- Dynamic Power Per Cell
- Power Per Rail
- Core Dynamic Power Details
- Core Dynamic Instance Power Details
- Core Dynamic Interconnect Power Details section

The following describes each section in more detail.

Section: Report Header

This section, common to all ACE reports, provides information about how and when the report was generated. It can be seen that the design name is `pcie_gddr6_ddr4_vp_demo_top`, which is a VectorPath® card demonstration design that configures the PCIe, GDDR6, and DDR4 interfaces. The operating conditions assumed for this report can also be seen: C2 speed grade, 0.85 volts, and 0 degrees Celsius.

Power Dissipation Report

```
ACE -- Achronix CAD Environment -- Version 9.2 -- Build 463545 -- Date 2023-09-20 15:15
Design: pcie_gddr6_ddr4_vp_demo_top - impl_1 - pcie_gddr6_ddr4_vp_demo_top
Device: AC711500ES0
Generated on Wed Sep 20 15:28:29 PDT 2023
Host: bob.achronix.local
```

Disclaimer

This is an estimation, based on the design characteristics and frequencies specified below. Power consumption is reported as worst case across all operating modes, including bitstream programming and user mode. Actual power results may vary from the estimates.

Operating conditions: C2, 0 deg. C, 0.85 V (Core), Fast Corner

Figure 115 • Example of Power Dissipation Report Header Section

Section: Power Summary

This section reports the total power dissipation, which is about 82 Watts in the example design, and then breaks that total down into four major sub-categories:

- The relative contributions of dynamic (switching) vs. static (leakage) power to the total
- The relative contributions of core (programmable) logic power vs. hardened IP power, both static and dynamic
- The relative contributions of instance vs. interconnect power in the programmable core, not including clock power
- Dynamic clock network power, which includes clock interconnect and clock instances, such as clock gates, clock dividers, and clock switches

Power Summary

Total Power:	81.7951 W
Total Dynamic Power:	63.5311 W
Total Static Power:	18.2640 W
Core Dynamic Power:	0.7087 W
Core Static Power:	1.5786 W
Hard IP Dynamic Power:	62.8223 W
Hard IP Static Power:	16.6854 W
Dynamic Instance Power:	0.5354 W
Dynamic Interconnect Power:	0.0848 W
Dynamic Clock Network Power:	0.0885 W

Figure 116 - Example of Power Dissipation Report Power Summary Section

Section: Hard IP Power

This section reports details of the power contributed by each hard IP block in the design. The example design contains many of the available hard IP blocks, so the report contains many rows. For example, it can be seen that more than half of the total power dissipation in the design, which contains very little core logic, is used by the GDDR6 controller.

In order to include the power contribution of hard IP blocks, those IPs must have been configured in the [IP Configuration editors \(page 336\)](#), and the associated [IP Configuration Files \(page 224\)](#) must be included in their ACE project.

Hard IP Power

CLKIO_NE:	0.4635 W
CLKIO_NW:	0.5251 W
CLKIO_SE:	0.4812 W
CLKIO_SW:	0.4507 W
DDR4:	2.2601 W
Ethernet:	3.2886 W
GDDR6:	43.1043 W
GPIO_N_B0:	0.3118 W
GPIO_N_B1:	0.3101 W
GPIO_N_B2:	0.2349 W
GPIO_S_B0:	0.3052 W
GPIO_S_B1:	0.1381 W
GPIO_S_B2:	0.2284 W
NoC:	15.2879 W
PCIe:	7.3981 W
PLL:	0.3166 W
SERDES_LN_0:	0.2221 W
SERDES_LN_1:	0.2221 W
SERDES_LN_10:	0.2221 W
SERDES_LN_11:	0.2221 W
SERDES_LN_12:	0.2221 W
SERDES_LN_13:	0.2221 W
SERDES_LN_14:	0.2221 W
SERDES_LN_15:	0.2221 W
SERDES_LN_2:	0.2221 W
SERDES_LN_3:	0.2221 W
SERDES_LN_4:	0.2221 W
SERDES_LN_5:	0.2221 W
SERDES_LN_6:	0.2221 W
SERDES_LN_7:	0.2221 W
SERDES_LN_8:	0.2221 W
SERDES_LN_9:	0.2221 W
SoC:	0.8493 W
iNoC:	6.3650 W

Figure 117 • Example of Power Dissipation Report Hard IP Power Section

Section: Dynamic Power Per Cell

This section reports details of the dynamic power contributed by instances of each cell type, such as Reconfigurable Logic Blocks (LUT6s, DFFs, MUXes and ALUs), Block RAMs, Network Access Points, etc. This design contains very little programmable core logic, so these numbers are quite small relative to the Hard IPs.

Dynamic Power Per Cell

RLB Power: 0.0062 W
 BRAM Power: 0.0026 W
 LRAM Power: 0.0016 W
 MLP Power: 0.0000 W
 NAP Power: 0.5250 W
 Hard IP Power: 85.8727 W
 Other Power: 0.0000 W

Figure 118 - Example of Power Dissipation Report Dynamic Power Per Cell Section

Section: Power Per Rail

This section reports the power dissipation for each power rail in the design (rows), breaking it down into categories (columns) such as dynamic vs. static power, and core logic vs. hard IPs. For example, the VDDL power rail represents the programmable logic core of the design, and it contributes all of the core static and dynamic power. The rest of the power rails are used exclusively by the hard IPs.

Power Per Rail

Power Rail	Total Power (W)	Total Dynamic Power (W)	Total Static Power (W)	Core Dynamic Power (W)	Core Static Power (W)	Hard IP Dynamic Power (W)	Hard IP Static Power (W)
CLKIO_NE_VDDIO	0.4124	0.3592	0.0532	0.0000	0.0000	0.3592	0.0532
CLKIO_NW_VDDIO	0.4124	0.3592	0.0532	0.0000	0.0000	0.3592	0.0532
CLKIO_SE_VDDIO	0.4124	0.3592	0.0532	0.0000	0.0000	0.3592	0.0532
CLKIO_SW_VDDIO	0.4124	0.3592	0.0532	0.0000	0.0000	0.3592	0.0532
DDR4_S0_VAA	0.0129	0.0108	0.0021	0.0000	0.0000	0.0108	0.0021
DDR4_S0_VDDQ	0.7491	0.7098	0.0393	0.0000	0.0000	0.7098	0.0393
FCU_VDDIO	0.8452	0.1690	0.6762	0.0000	0.0000	0.1690	0.6762
GCG_NE_PLL_VDDA	0.0120	0.0015	0.0105	0.0000	0.0000	0.0015	0.0105
GCG_NW_PLL_VDDA	0.0120	0.0015	0.0105	0.0000	0.0000	0.0015	0.0105
GCG_SE_PLL_VDDA	0.0120	0.0015	0.0105	0.0000	0.0000	0.0015	0.0105
GCG_SW_PLL_VDDA	0.0166	0.0061	0.0105	0.0000	0.0000	0.0061	0.0105
GDDR6_E_VDDA	8.4480	8.1413	0.3067	0.0000	0.0000	8.1413	0.3067
GDDR6_E_VDDIO	4.2780	3.8947	0.3833	0.0000	0.0000	3.8947	0.3833
GDDR6_E_VDDP	4.6350	4.2197	0.4153	0.0000	0.0000	4.2197	0.4153
GDDR6_E_VDDR	3.3300	2.8442	0.4858	0.0000	0.0000	2.8442	0.4858
GDDR6_W_VDDA	8.4480	8.1413	0.3067	0.0000	0.0000	8.1413	0.3067
GDDR6_W_VDDIO	4.2780	3.8947	0.3833	0.0000	0.0000	3.8947	0.3833
GDDR6_W_VDDP	4.6350	4.2197	0.4153	0.0000	0.0000	4.2197	0.4153
GDDR6_W_VDDR	3.3300	2.8442	0.4858	0.0000	0.0000	2.8442	0.4858
GPIO_N0_VDDIO	0.4905	0.4888	0.0017	0.0000	0.0000	0.4888	0.0017
GPIO_S0_VDDIO	0.3250	0.3234	0.0017	0.0000	0.0000	0.3234	0.0017
SRDS_N_PA_VDDH	5.9922	0.9466	5.0456	0.0000	0.0000	0.9466	5.0456
SRDS_N_PA_VDDL	7.1711	0.7401	6.4309	0.0000	0.0000	0.7401	6.4309
TS_VDDA	0.0004	0.0001	0.0004	0.0000	0.0000	0.0001	0.0004
VCC	20.8372	19.7863	1.0509	0.0000	0.0000	19.7863	1.0509
VDDL	2.2874	0.7087	1.5786	0.7087	1.5786	0.0000	0.0000

Figure 119 - Example of Power Dissipation Report Power Per Rail Section

Section: Core Dynamic Power Details

This section reports the programmable core dynamic power by power rail and clock domain (rows), breaking it down into categories (columns) for non-clock instance power, non-clock interconnect power, and clock instance and interconnect power.

Core Dynamic Power Details

Power Rail	Clock	Frequency (MHz)	Total Dynamic Power (W)	Instance Power (W)	Interconnect Power (W)	Clock Power (W)
VDDL	i_nap_clk	499.88	0.2035	0.1447	0.0252	0.0336
VDDL	i_reg_clk	249.94	0.4460	0.3530	0.0464	0.0466
VDDL	i_adm_clk	124.97	0.0575	0.0376	0.0127	0.0072
VDDL	tck	25.00	0.0015	0.0001	0.0004	0.0011
VDDL	v_acx_sc_GPIO_H_JOB1_GLB_SER_GEN_CLK	10.00	0.0000	0.0000	0.0000	0.0000
VDDL	Combinational	----	0.0001	0.0000	0.0001	0.0000

Figure 120 • Example of Power Dissipation Report Core Dynamic Power Details Section

Section: Per-Tile Core Dynamic Instance Power Details

This section reports the programmable core dynamic instance power by power rail and clock domain (rows), breaking it down into categories (columns) for each type of logic tile, such as clock tiles, boundary IO pin tiles, Machine Learning Processor (MLP) tiles, 2D Network on Chip (NoC) tiles, and Reconfigurable Logic Block (RLBs) tiles.

Per-Tile Core Dynamic Instance Power Details

Power Rail	Clock	Frequency (MHz)	Total Instance Power (W)	Clock Power (W)	IO Pin Power (W)	MLP Power (W)	NOC Power (W)	RLB Power (W)
VDDL	i_nap_clk	499.88	0.1447	0.0000	0.0000	0.0016	0.1416	0.0016
VDDL	i_reg_clk	249.94	0.3530	0.0000	0.0000	0.0016	0.3480	0.0034
VDDL	i_adm_clk	124.97	0.0376	0.0000	0.0000	0.0011	0.0354	0.0012
VDDL	tck	25.00	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
VDDL	v_acx_sc_GPIO_H_JOB1_GLB_SER_GEN_CLK	10.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
VDDL	Combinational	----	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 121 • Example of Power Dissipation Report Per-Tile Core Dynamic Instance Power Details Section

Section: Per-Tile Core Dynamic Interconnect Power Details

This section reports the programmable core dynamic interconnect power by power rail and clock domain (rows), breaking it down into categories (columns) for each type of logic tile, such as clock tiles, boundary IO pin tiles, Machine Learning Processor (MLP) tiles, 2D Network on Chip (NoC) tiles, and Reconfigurable Logic Block (RLBs) tiles. Interconnect power includes the actual routing wires, as well as the statically programmable FPGA routing MUXes used to connect different wire segments to form a complete route.

Per-Tile Core Dynamic Interconnect Power Details

Power Rail	Clock	Frequency (MHz)	Total Interconnect Power (W)	Clock Power (W)	IO Pin Power (W)	MLP Power (W)	NOC Power (W)	RLB Power (W)
VDDL	i_nap_clk	499.88	0.0252	0.0194	0.0000	0.0017	0.0012	0.0366
VDDL	i_reg_clk	249.94	0.0464	0.0256	0.0000	0.0077	0.0035	0.0563
VDDL	i_adm_clk	124.97	0.0127	0.0027	0.0000	0.0010	0.0003	0.0158
VDDL	tck	25.00	0.0004	0.0007	0.0000	0.0000	0.0000	0.0007
VDDL	Combinational	----	0.0001	0.0000	0.0000	0.0000	0.0000	0.0001

Figure 122 • Example of Power Dissipation Report Per-Tile Core Dynamic Interconnect Power Details Section

Power Calculation Methodology

Power dissipation is calculated using the following basic equation:

$$power_{total} = \sum_{nets} (energy_{net} \times frequency_{net} \times activity_{net})$$

Here total power is summed up over the individual contribution of every net in the design, including nets internal to logic gates and hard IP blocks. Power for each net is calculated as the product of the total *energy* dissipated in each zero-to-one or one-to-zero transition, measured in units of joules, which is derived from extracted layout data and present in the technology library being used; the target *frequency* of that net clock domain, measured in units of 1/seconds; and a unitless quantity called *activity*, which is defined as the average number of transitions *per clock cycle*.

Activity can be larger than one if a net experiences a lot of glitching. The activity factor for clock nets is, by definition, 2.0, since there are two transitions per cycle. ACE typically uses a default value of 0.125 for the activity value of non-clock nets. This value reflects an assumption that non-clock nets switch, on average, about once every eight clock cycles. This has been shown to be a reasonable assumption across a wide variety of designs, but it is, of course, only an approximation and not always correct.

In order to produce the most accurate power report possible, per-net activity values can be specified in a Switching Activity Interchange Format (SAIF) file. SAIF files are typically produced by a gate-level logic simulator. However, if necessary, the file can be written by hand. The accuracy of the activity values in a SAIF file depend on the quality of the simulation testbench, and are only reflective of the operating mode(s) exercised in that testbench. See the section [How to Generate a Simulation-Driven Power Dissipation Report](#) (page 252) below for more information.

How to Generate the Power Dissipation Report

This report is generated automatically when running the ACE [Flow](#) (page 234) as part of the [write_reports_final](#) (page 240)

flow step. However, it is not generated by default. The `report_power` [implementation option](#) (page 304) must be set to "1" in the project before running the `write_reports_final` flow step.

This report can also be generated on-demand by [restoring the routed implementation](#) (page 302) and then calling the [report_power](#) (page 683) Tcl command.

As power dissipation varies across different process corners and operating conditions, a separate power dissipation report is generated for each temperature corner, with the file name of the report noting the corner being reported. See the page [Timing Across All Temperature Corners \(page 266\)](#) for more information.

The Power Dissipation Report can optionally include the contribution of hard IP blocks, such as the PCIe controller and the 2D Network on Chip. In order to include those contributions (which can often dominate the contribution of the programmable logic), those hard IP blocks must have been configured in the [IP Configuration editors \(page 336\)](#), and also have included the associated [IP Configuration Files \(page 224\)](#) in the ACE project.

How to Generate a Simulation-Driven Power Dissipation Report

As discussed above in the Power Calculation Methodology section, per-net *activity* values can be specified in a text file called a Switching Activity Interchange Format (SAIF) file. SAIF files are typically produced by a gate-level logic simulator such as Synopsys VCS, Cadence Incisive, Siemens QuestaSim, or Aldec Riviera. This section covers the format of a SAIF file, how to use a SAIF file in ACE, and how to generate a SAIF file in the VCS and QuestaSim simulators. The documentation for your simulator of choice should be consulted for definitive instructions.

Format of a SAIF File

SAIF files are written in a clear-text format defined in IEEE Standard 1801-2009 (see <https://ieeexplore.ieee.org/document/8686430>). Therefore, if desired, small SAIF files can be generated by hand or with a script. The standard defines both *backward* and *forward* SAIF file formats. Here we are using a *backward* SAIF file which is intended for back-annotation into EDA tools. Below is an example of a complete SAIF file that is accepted by ACE.

Sample SAIF File

```
(SAIFILE
  (SAIFVERSION "2.0")
  (DIRECTION "backward")
  (DURATION 5950100.00)
  (INSTANCE tb_test1
    (INSTANCE DUT
      (NET
        (ina\[0\]
          (T0 3750100) (T1 2200000) (TX 0)
          (TC 22) (IG 0)
        )
        (clk
          (T0 3000100) (T1 2950000) (TX 0)
          (TC 118) (IG 0)
        )
      )
    )
  )
)
```

The file must:

1. Begin with the SAIFILE keyword.

2. Give the version number.
3. Specify that it is a backward SAIF file.
4. Give the duration of the simulation as the number of verilog timesteps.

Most important are the `INSTANCE` and `NET` sections of the file. It can be seen that this design has two levels of hierarchy. The instance `tb_test1` is the testbench containing behavioral verilog that applies test vectors to the device under test. The instance `DUT` is the Device Under Test, which is an instance of the top level design being compiled in ACE. The design contains two nets, an input net, `ina[0]`, and a clock net, `clk`. Square brackets and other reserved characters must be escaped with `"\"` in the SAIF file.

Under each net are five values, `T0`, `T1`, `TX`, `TC`, and `IG`. The `T0`, `T1`, and `TX` values are the number of timesteps that the net spent at the logic values 0, 1, and X respectively. The `IG` value is the number of inertial glitch values — for details see the documentation. The only important value for ACE is the `TC` value, which is the total number of zero-to-one and one-to-zero transitions over the duration of the simulation. Since the clock net had 118 transitions during simulation, and the `ina[0]` net had 22 transitions, ACE calculates the activity value, which is defined as the average number of transitions per clock cycle, as $22/118 = 0.1864$ (not very different from the default activity value of 0.125).

How to Use a SAIF File in ACE

To use a SAIF file, two [implementation options \(page 304\)](#) must be specified to ACE:

1. The pathname to the SAIF file
2. The name of the top-level instance in the simulation testbench

The Tcl interpreter normalizes the pathname, so it can be an absolute or relative path, a direct pathname or a symbolic link, and in Windows or Linux format. The name of the top-level instance in the SAIF file must be specified because ACE only knows the name of the top-level *module* in the RTL. Since the testbench is not being compiled, ACE does not know the name of the *instance* of that module in the behavioral simulation testbench module. The default SAIF file pathname is the empty string (no file specified). The default top-level instance name is "DUT", as in this example.

When generating the Power Dissipation Report during the normal ACE flow, specify the pathname to the SAIF file in the ACE project file using the `report_power_saif_file` implementation option, and the name of the top-level instance in the `report_power_saif_top_level` implementation option:

```
set_impl_option -project test1 -impl impl_1 report_power_saif_file "../vcs-routed/
sim_output.saif"
set_impl_option -project test1 -impl impl_1 report_power_saif_top_level "DUT"
```

When generating the Power Dissipation Report from the ACE command line, simply call the `report_power` command with the following two options:

```
report_power -saif_file ../vcs-routed/sim_output.saif -saif_top_level DUT
```

How to Generate a SAIF File Using Synopsys VCS

In this section, for convenience, we provide instructions for how to generate a SAIF file using Synopsys VCS. However, note that VCS behavior may change at any time, and these instructions may be incorrect or

incomplete. More than one method of generating a SAIF file may be supported. Consult your simulator documentation for definitive instructions.

There are four steps required:

1. Generate a Final Simulation Netlist (a post-routing simulation netlist that matches the netlist used by the power report) using ACE. The [Generate Final Simulation Netlist \(page 240\)](#) flow step is an optional flow sub-step that runs under the Design Completion flow step, so that sub-step must be enabled before running the ACE flow.
2. From within the verilog testbench, several additional system functions must be called to specify which part of the design to be observed, and to start and stop the recording of data. The following code can be used as a rough template for a simulation testbench:

```
initial begin
    $set_gate_level_monitoring("on");
    $set_toggle_region(mixedHdlScope);
    // initialization of Verilog signals, and then:
    $toggle_start;
    // testbench
    $toggle_stop;
    $toggle_report("sim_output.saif", $timeUnit, mixedHdlScope);
end
```

3. Call the VCS compiler to generate the `simv` executable. To generate a SAIF file, the `-debug_access+pp` flag must be added:

```
vcs <input_files> <testbench> -debug_access+pp
```

4. Call the simulation executable. To generate a SAIF file, the `-lcs` (limited customer access) flag must be added:

```
simv -lcs
```

How to Generate a SAIF File Using Siemens QuestaSim

In this section, for convenience, we provide instructions for how to generate a SAIF file using Siemens QuestaSim. However, note that QuestaSim behavior may change at any time, and these instructions may be incorrect or incomplete. More than one method of generating a SAIF file may be supported. Consult your simulator documentation for definitive instructions.

There are seven steps required:

1. Generate a Final Simulation Netlist (a post-routing simulation netlist that matches the netlist used by the power report) using ACE. The [Generate Final Simulation Netlist \(page 240\)](#) flow step is an optional flow sub-step that runs under the Design Completion flow step, so that sub-step must be enabled before running the ACE flow.
2. Launch QuestaSim in interactive mode, while specifying the additional option `-voptargs="+acc"`. This option prevents QuestSim from compressing the netlist and throwing away nodes that must be reported.

```
vsim -voptargs="+acc"
```

3. After the simulator returns to the prompt, run the following command to request the simulator to stop rather than exiting after the simulation completes:

```
onfinish stop
```

4. Specify the scope of the data to be collected, which should include all nets in the DUT. Using the example SAIF file above, this would be as follows:

```
power add -in -inout -internal -out tb_test1.DUT.*
```

5. Run the simulation:

```
run -all
```

6. After the simulation completes, request the simulator to write out the SAIF file:

```
power report -al -bsaif sim_output.saif
```

7. Quit QuestaSim:

```
quit
```

Design Statistics Report

The Design Statistics Report is meant to show various statistics about the current design.

Presently, the only statistics being reported are a histogram showing LUT Function usage counts. This information is primarily meant as a tool for Achronix Technical Support to assist ACE users unable to share their full design. It allows Achronix to better understand the nature of the design and thus provide advice on QoR improvements.

This report is not automatically generated by the [Flow \(page 234\)](#), but can be generated manually with the [report_design_stats \(page 679\)](#) Tcl command.

Multiprocess Summary Report


The Multiprocess Summary Report provides a comparative summary of the achieved frequencies and worst-case slacks (if the "Run Post-Route Timing Analysis" or "Run Sign-off Timing Analysis" [Flow Steps \(page 234\)](#) are enabled), as well as process execution times, for selected [Implementations \(page 229\)](#) of a single Project when executed from the [Multiprocess View \(page 73\)](#). This report is automatically shown (and refreshed) in the ACE Editor Area as new results become available during Multiprocess execution.

The Multiprocess Summary Report is generated/updated multiple times during a Multiprocess execution as new data becomes available from the executing [Implementations \(page 229\)](#). While the Multiprocess flow is still executing, the report contains incomplete results — rows containing incomplete data are marked as such. Similarly, if errors are encountered during flow execution for one of the selected [Implementations \(page 229\)](#), the row(s) of data for that implementation are marked accordingly.

Because the Multiprocess Summary Report summarizes the results of multiple [Implementations \(page 229\)](#) (unlike the other reports, which are generated for a single implementation), the HTML file containing the report is not placed in any implementation `report` subfolder. Instead, this report is placed in the directory containing the `.acxprj` ACE [Project File \(page 222\)](#), and is named `multiprocess_summary.html`.

 **Tip**

If closed, the Multiprocess Summary Report can be re-opened at any time.

To re-open the Multiprocess Summary Report for the active project, select the () **Open Multiprocess Report** button in the upper-right of Multiprocess view, or from the context menu when right-clicking the project in the [Projects View \(page 117\)](#).

This report is only available in HTML format. There is no Tcl command available to generate this report manually.

For more information on how this report is used, see [Running Multiple Flows in Parallel \(page 308\)](#) and [Attempting Likely Optimizations Using Option Sets \(page 392\)](#).

Timing Results Summary Section

The Timing Results Summary section of the report is only generated if either the "Run Post-Route Timing Analysis" or "Run Sign-off Timing Analysis" [Flow Steps \(page 234\)](#) are executed during the Multiprocess Flow.

 **Caution!**

Any desired optional flow steps in the [Flow View \(page 53\)](#) must be enabled before Multiprocess execution is started.

The completed implementations in this section are sorted in approximate QoR order by their timing results. The implementation with the best results is at the top (also see the [get_best_multiprocess_impl \(page 644\)](#) Tcl command.)

While an implementation is waiting to be executed or is currently executing, the Clock Domain column of the report contains a message to indicate the implementation execution status.

When an implementation has completed execution, the timing results for that implementation are populated in the Report. Much of the time, each implementation provides timing results (Upper-Limit Frequency, Worst Setup Slack, and Worst Hold Slack) for a single PVT combination (named in a column group header). However, when multiple PVT combinations are reported, each PVT will be shown under its own column group header in the summary report. The table cells containing results for the active PVT combination (the specific combination chosen for an implementation, which would otherwise be the only one reported) are shown in bold to differentiate from other PVT combinations shown for that implementation. (See [Timing Across All Temperature Corners \(page 266\)](#) for additional details about when multiple PVT corners will be reported for an implementation.)

The Upper-Limit Frequency, Worst Setup Slack, and Worst Hold Slack cells are independently color-coded to indicate whether the timing constraints were met for each of the defined clock domain/PVT combinations for that implementation. If the timing constraints were met for all clocks and all reported PVT combinations in an implementation, the Implementation Name cell is also color coded green to indicate success. If errors are encountered during flow execution, the appropriate Implementation Name cells are colored red, and "**(Flow Error)**" appears in that row.

Hyperlinks to each detailed [Timing Report \(page 244\)](#) are made available under the first clock domain Upper-Limit Frequency column for each PVT (one is created for each implementation for each reported PVT).

In some cases, some implementations provide Sign-Off timing results but other implementations do not. This difference can happen most often when some implementations are in Evaluation **Flow Mode** (page 242) while others are not. It may also happen when the multi-process flow is cancelled, or in rare flow error cases. When this mix of timing results from differing flow steps occurs, the column headings indicate that the report contains Sign-Off data. All earlier Post-Route timing results are footnoted to indicate that they are not showing Sign-Off data.

Runtime Results Summary Section

This section of the report is always generated and indicates the total flow execution time and peak memory utilization for each implementation. The flow runtimes reported are wall-clock runtimes (not CPU times) harvested from the run logs. Rows in the table indicate whether an implementation flow execution is incomplete (running or still waiting to run), or has encountered errors.

Many factors affect runtimes:

- Selected **Implementation Options** (page 0) for an implementation (and thus the **Option Sets** (page 0)) can have a significant impact.
- The available processors, available memory, and total load on the workstation executing ACE have a significant impact.
 - On multi-user workstations, workstation load may vary widely over the multiprocess execution period.
 - Even on single-user workstations, if using a **Parallel Queue Count** greater than one, be aware that the last-executed implementation(s) likely can be executing (at least partially) with a reduced machine load compared to the first-executed implementations. As the parallel execution queue is emptied, new implementation processes are not started, thus fewer processes are executing, meaning that the last-executed implementations have the lowest contention for processing cores, caches, memory, I/O, etc.
- Additionally, when using external cloud/grid/batch Job Submission systems:
 - The reported times do not include the time spent waiting in the external system queue(s). The runtimes reported only include time spent actually running the ACE flow.
 - The processor speeds and system loads may vary widely for the nodes running each job, making runtime comparisons difficult (at best).
 - If meaningful runtime comparisons are required, extra care must be taken to ensure that each node system load and hardware is identical for all jobs for the duration of the multiprocess session.

Caution!

Do not compare runtimes unless workstation hardware and system loads are identical for all implementations. Because of all of the limitations listed above, the reported implementation flow execution wall-clock times are intended to be used only as general guidelines, not as benchmarking times.

Implementation Options Report

The Implementation Options Report provides information about available options for the currently active implementation. This report is not automatically generated by the **Flow** (page 234), but can be generated on demand using the `report_impl_options` (page 679) Tcl command.

By default, the report only displays the names, descriptions and default values of the most commonly used implementation options (meaning the subset which is shown within the [Options View \(page 96\)](#)). In addition, by default, the report displays the options applicable to the target device of the currently active implementation.

The `-project` and `-impl` arguments can be used to show the options applicable to a different project implementation.

The `-show_values` argument can be used to include the current implementation value of each option in the report.

The `-show_all` argument can be used to include all possible options for the specified implementation in the report, instead of only the most-commonly-used subset shown within the [Options View \(page 96\)](#).

 **Note**

The values of hidden options (those which are not shown in the GUI Options View) should only be altered after guidance from Achronix support.

Advanced Concepts

The following are advanced concepts intended primarily for extremely experienced users, or users being actively guided by Achronix FAEs.

ACE Verilog Attributes

This page lists various attributes that can be applied to instances, nets, pin, ports, or other objects in the ACE data model. Each attribute is listed with the type of object or objects to which it may be applied, and a description of how it effects the ACE synthesis, placement, and routing flow.

locked

Applies to nets only.

Nets marked with this attribute are not unrouted by the `run_unroute` command, or when the `run_route` flow step is re-run on a routed design.

fanout_limit

Applies to nets only.

This is a dual-use property and can be applied globally and also individually to nets.

The global limit is specified with the `fanout_limit` impl option.

When applied to an individual net, this attribute overrides the global limit. Net drivers are cloned, or buffer trees inserted, to keep each (non clock/reset) net fanout under this limit. Applies only when the `fanout_control` impl option is enabled.

 **Note**

ACE never inserts more fanout buffers than the maximum specified by the `max_buffer_limit` impl_option.

In order to find the correct net name, if the name of the driving register is known, the following Tcl command can be used:

```
set_property fanout_limit 50 [ find {*} -nets -filter @driving_pin=[get_pins *<source reg>*q] ]
```

must_keep

Applies to instances or nets.

The instance or net cannot be deleted by any of the netlist optimization flow commands.

Note

If the instance or net is logically redundant, it might be left dangling with no input and no output connections.

do_not_rewire

Applies to instances, nets, pins.

Rewiring permutes input pins among equivalent nets (i.e., nets in a tree of fanout-control buffers) to minimize wirelength and improve timing. This attribute disables rewiring for a given instance, net, or pin. Some rewiring can happen during `run_prepare`, but the majority of changes are made after `run_place` and `run_route`. Subject to the `prepnr_rewire` and `postpnr_rewire` implementation options.

do_not_clone

Applies to instances only.

Prevents a given instance from being cloned during fanout control optimization. However, fanout buffers may still be inserted for nets above the `fanout_limit`.

do_not_cluster

Applies to instances only.

Prevents an instance from being clustered during structural or timing-driven clustering. Structural clustering (enabled with the `structural_clustering_mode` implementation option) creates clusters from groups of LUTs and Flops when certain pre-defined structures are encountered. Timing-driven clustering (enabled by the `timing_driven_clustering` implementation option) creates larger clusters from LUT-to-LUT and ALU-to-LUT connections to keep timing-critical net routing short. Instances in a cluster are placed together.

clock_type

Applies to nets or driver (output) pins.

Normally, this property is set on a net or driver (output) pin using the `set_clock_type` TCL command in the SDC constraints, but it can also be specified using this attribute. Applies globally to all target pins driven by that net or pin. For more information see the documentation for the `set_clock_type` (page 709) Tcl command. The attribute must be a comma-separated list of the following strings: {"boundary", "trunk", "direct_trunk", "minitrunk" (Speedcore only), "blocked", "data_region", "data_center", "data_local", "branch_fast", "branch_nominal", "none", "low_jitter", "local", "global", "immediate"}.

Note

Not all attribute combinations make sense.

local_clock_type

Applies to pins only.

Has all of the same properties as the `clock_type` attribute above, but use this attribute when the type is being specified for a specific target (input) pin, and the routing type needs to be different than the global value specified for the driving net/pin. Useful for certain kinds of data-generated clocks: parts of the clock that feed back should be "data" but the rest should be "data_region" or "data_center".

ace_useioff

Applies to ports, nets, I/O pin instances, or flop-flop instances.

Depending on the value of the `push_flops_into_pads` implementation option, the presence of this property causes boundary flop-flop instances to be pushed into the attached input or output pin instance. For more information see the [Automatic Flop Pushing into I/O Pins \(page 457\)](#) section.

ace_virtualize

Applies to ports only.

Allows specifying which ports and port busses are virtualized when ACE is run in evaluation flow mode, and when the design contains more top level ports than available I/O pads/pins. For more information see the [Working with Virtual I/O \(page 466\)](#) section.

ace_virtualize_clock_port

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows specifying the top-level port name to be connected to the clock input of the new serialization flop instances. Cannot be used with the `ace_virtualize_clock_net` attribute (they are mutually exclusive). For more information see the [Working with Virtual I/O \(page 466\)](#) section.

ace_virtualize_clock_net

Applies to ports only.

When the `virtual_io_style` implementation option is set to the value `serialize_dff`, allows specifying the top-level net name to be connected to the clock input of the new serialization flop instances. Cannot be used with the `ace_virtualize_clock_port` attribute (they are mutually exclusive). For more information see the [Working with Virtual I/O \(page 466\)](#) section.

async_capture

Applies to a DFF "d" input pin only.

Suppresses setup-and-hold checks at the input pin of a DFF instance during Standard Delay Format (SDF) export for the Timing Annotated Gate Level Simulation flow. This attribute is used on the capture flop of user-supplied clock domain crossing synchronizer macros. For more information, refer to the relevant IP Component Library User Guide for that product family.

location

Applies to instances only.

A string attribute giving the site name on which the instance is to be pre-placed. Equivalent to the "set_placement -fixed" PDC constraint.

Clock Regions

Device fabrics deal with numerous clocks. Due to physical routing limitations, only a finite number of clocks can be routed to each individual site within the fabric. To keep the placement/routing problem space manageable for the most complex designs, a fabric is divided up into Clock Regions of a relatively coarse granularity, where each Clock Region as a whole allows a finite number of clocks to be routed within that clock region.

Each fabric is divided up into a number of Clock Regions of roughly similar area. The exact numbers of clock regions, the dimensions of each region, the number and types of sites within the region, the allowed sources of the clocks routed to each region, and the differences (like skew) of clock behavior between clock regions all are specified by the chosen target device. A subset of this information is presented in the [Clock Regions View \(page 23\)](#). See the technical specification of the target device for complete details.

For designs with an extremely large number of clocks, the use of [Placement Regions and Placement Region Constraints \(page 394\)](#) may be necessary to guide placement decisions regarding Clock Regions. This should be discussed thoroughly with an Achronix FAE first, as improper use of Placement Region Constraints can lower QOR or even cause Placement or Routing to become unsolvable.





Instance States

An individual Instance can have a variety of states simultaneously in ACE; only the highest priority state is used to color the Instance in the [Floorplanner View \(page 43\)](#). The states are listed below in the default render priority order. Higher priority states (earlier in the table) take precedence over lower priority states (later in the table). For example, if an instance has Fixed Placement and is also Selected, the Selection color has priority, and the Selection color is used to paint the instance.

Additionally, several of the states have associated icons, which normally are displayed alongside the instance when the instance appears in tables and lists, as in the [Search View \(page 129\)](#), [Selection View \(page 133\)](#), and [Netlist Browser View \(page 79\)](#). Similar to the colors, when multiple simultaneous states are involved, the highest priority icon available is used. Thus, an instance that is both Fixed and Locked uses the Fixed icon, while an instance that is both Selected and Fixed also uses the Fixed icon (because the higher priority Selected state has no corresponding icon).

Table 122 - Instance States By Priority Order

Priority	State	Default Color	Icon	Description
1	Selected	bright green	-	This instance has been added to the ACE Selection Set, typically either by using the Selection View (page 133) or the Tcl select (page 708) command. For more information, see Selecting Floorplanner Objects (page 341) .

Priority	State	Default Color	Icon	Description
2	Highlighted	(user-defined)	-	An instance chosen to Highlight, typically by using the Highlight Instance commands in one of the views within the Floorplanner Perspective, or by the <code>highlight</code> (page 664) Tcl command. See Highlighting Objects in the Floorplanner View (page 343) for more information
3	Overassigned Site	bright red	-	An instance that shares a site assignment with another instance. Since a site can only legally contain a single instance, this is an error state.
4	Fixed Placement	bright yellow		An instance whose site assignment has been marked as "fixed". As long as an instance is defined with hard fixed placement, ACE does not change the site assignment for that instance during the Placement phase of PnR. For more information, see Placing an Object (page 347).
5	Locked Placement	dark yellow		An instance that is a member of a Locked Partition that has remained unchanged since the prior incremental compilation. ACE does not change the site assignment for that instance during the Placement phase of PnR. For more information, see Using Incremental Compilation (Partitions) (page 404).
6	Default (Soft) Placement	dark grey		A placed instance with no other specially defined state is displayed in this manner.
7	Unplaced	-		An instance without a current site assignment (placement).

 **Note**

The colors mentioned are the default colors. These colors may be modified on the [Floorplanner View Colors and Layers Preference Page](#) (page 194). On that same preference page, it is possible to toggle whether some states are shown at all, and to partly alter the render priority of some states.

Filter Properties

Several Tcl commands [[find](#) (page 638), [filter](#) (page 637)] allow Tcl command-line filtering of object lists. Additionally, the [Search Filter Builder dialog](#) (page 186) performs a similar function for the [Search view](#) (page 129) in the ACE GUI.

The allowed filtering properties, operators, and values (where applicable) are as follows:

Table 123 - Supported Filter Properties

Property Name	Operators	Values	Description
@async_reset	=	true, false	Allows filtering ports, nets, and pins based on whether they connect to an async reset.
@attribute	=	<i>property : value</i>	Allows filtering instances, nets, and ports based on verilog attribute/defparam values. Values of the @attribute filter must be a property-value pair separated by a semicolon (i.e., prop:value).
@clock	=	true, false	Allows filtering ports, nets, and pins based on whether they connect to a clock net.
@clock_as_data	=	true, false	Allows filtering instances and nets based on whether they have any data pins being driven by a clock signal.
@clock_domain ⁽¹⁾	=	<i>clockDomainName</i>	Allows filtering instances, nets, paths, and pins based on clock domain name.
@clock_region	=	<i>clockRegionName</i>	Allows filtering instances, nets, paths, sites, and pins based on clock region name. Some nets and paths may be part of multiple clock regions.
@data_as_clock	=	true, false	Allows filtering nets based on whether they have any clock pins being driven by a data signal.
@direction	=	in, out, inout	Allows filtering ports and pins based on direction.
@driver_type	=	(device-dependent)	Allows filtering nets and pins based on the type (cell name) of the driving instance.
@driving_net	=	<i>netName</i>	Allows filtering instances based on a net name that is driving them.

Property Name	Operators	Values	Description
@driving_pin	=	<i>pinName</i>	Allows filtering instances and nets based on the name of a pin that is driving them.
@enable	=	true, false	Allows filtering ports, nets, and pins based on whether they connect to an enable signal.
@fanout	=, <, >	(integers > 0)	Allows filtering nets and pins based on the number of fanout connections. Valid @fanout values must be integers greater than 0.
@fixed_placement	=	true, false	Allows filtering instances based on whether placement is fixed.
@partition	=	<i>partitionName</i>	Allows filtering instances, nets, paths, and pins based on partition name.
@placed	=	true, false	Allows filtering instances, nets, and pins based on whether they are placed. In this context, "placed" means the net is routed.
@power	=, <, >	(floating point numbers > 0.0)	Allows filtering nets based on power consumption. Valid @power values must be a floating point number greater than 0.0.
@power_rank	=, <, >	(integers ≥ 0)	Allows filtering nets based on the level of power consumption relative to other nets. The most power-consuming net is ranked 1 while the least power consuming net is ranked n. Valid @power_rank values must be integers greater than or equal to 0.
@region	=	<i>placementRegionName</i>	Allows filtering instances, nets, paths, and pins based on placement region name.
@reset	=	true, false	Allows filtering ports, nets, and pins based on whether they connect to a reset signal.
@sink_type	=	(device-dependent)	Allows filtering nets and pins based on the type (cell name) of instance(s) the net is driving.
@type	=	(device-dependent)	Allows filtering instances based on the type (cell name) of the instance.

Property Name	Operators	Values	Description
Table Notes			
1. Some instances may be part of multiple clock domains, such as a CDC instance.			

Timing Across All Temperature Corners

ACE supports place and route across all temperature corners, as well as reporting timing across all temperature corners of interest for a given place-and-route result at a specific junction temperature. This allows the designer to confirm whether the optimized placed-and-route result is able to close timing at the desired frequency (F_{MAX}) at all temperature corners of interest.

Temperature Corners and Place and Route

Note

By default, ACE only optimizes place and route for a single user-chosen temperature corner per implementation. If it is desired to optimize place and route for multiple temperatures, the impl-option `pnr_optimize_corners` should be enabled (set to 1).

If `pnr_optimize_corners` is zero (disabled, the default), ACE requires setting the desired junction temperature at which the design is placed and routed for the target F_{MAX} . This selected junction temperature is used to load the corresponding timing libraries and to optimize the place and route to close timing at the requested frequency.

If `pnr_optimize_corners` is not zero (and is thus enabled), the timing libraries for all corners will be used to optimize the place and route, though ACE still requires that a single junction temperature be chosen (for use during Timing Analysis and the generated timing reports for the early flow steps).

Be warned that the value of `pnr_optimize_corners` has no impact upon the generated [Timing Report \(page 244\)](#)s. The number of corners included in the generated timing reports is influenced only by the [Flow Steps \(page 234\)](#) and the chosen [Flow Mode \(page 242\)](#).

Temperature Corners and Timing Analysis/Reports:

When the Flow Mode is set to **Evaluation**, this will force all timing analysis to report only a single temperature corner, even for the Sign Off Timing Report, regardless of other settings. (The single corner will be the one corresponding to the implementation option `junction_temperature`, shown as **Design Preparation** → **Junction Temperature** in the [Options View \(page 96\)](#).)

When the Flow Mode is set to **Normal** or **Strict**, after all DRC checks have passed (see the **Run Final DRC Checks** flow step), results are sufficiently accurate that timing analysis results will be generated for all relevant temperature corners. This is needed because any given corner might be worse than any other given the underlying hardware technology used (due to the effects of temperature inversion). Keep in mind that prior to passing the DRC checks, timing analysis results will still only be generated for a single temperature corner, the one corresponding to the

implementation option `junction_temperature`, shown as **Design Preparation** → **Junction Temperature** in the Options View.

Table 124 - Number of temperature corner timing reports generated (as influenced by Flow Mode and Flow Step)

	Estimated Timing Analysis Report	Post-Placement Timing Analysis Report	Post-Route Timing Analysis Report	Sign-off (Final) Timing Analysis Report
Evaluation Flow Mode	1 corner	1 corner	1 corner	1 corner
Normal Flow Mode	1 corner	1 corner	1 corner	all corners
Strict Flow Mode	1 corner	1 corner	1 corner	all corners



The value of `pnr_optimize_corners` does not influence the number of temperature corners generated for any of the timing reports.

When reports are generated for multiple corners, the report file names are extended with a suffix describing the PVT corner contained within the report. The suffix includes the speed grade, the voltage, and the temperature, in that order, separated by underscores: `_speed_grade_voltage_junction_temp`

For example:

- `_C1_1p00V_85C` corresponds to: speed grade = C1; voltage = 1.00V; junction temperature = 85°C
- `_C2_0p70V_n40C` corresponds to: speed grade = C2; voltage = 0.70V; junction temperature = -40°C

Note

The **Power Dissipation Report** (page 245)s are generated for each temperature corner as well.

Currently, the only way to see a single report summary for multiple temperature corners at once is through the **Multiprocess Summary Report** (page 255). Keep in mind that even when using the Multiprocess feature, each individual implementation being run will still follow all the temperature corner processing rules mentioned above. Using Multiprocess does not change any of the above temperature corner optimization/analysis rules.

ECO Commands

ECO Use Cases

The ECO Command Tool Chain is a set of useful tools that allow editing a design with a high level of granularity. These tools can be used to achieve highly specific goals, such as manually adding logic blocks to the fabric, improving timing, performing analysis on the FPGA itself, and more.

Net Legality Definition

Throughout this documentation, the concept of *net legality* is mentioned frequently. For a net to be legal, all of the following must be true:

- There is exactly one and only one driving/output pin connected to the net.
- There is at least one input/sink pin connected to the net.
- All instances to which the net connects must be placed (with the exception of their respective site pins which do not need to be placed).

When a net is *legal*, the router may route the net. An *illegal* net causes the router to silently exit. When performing ECO changes, it is necessary to tie-off and/or correct any nets that become illegal. ECO commands indicate illegal nets caused by each ECO command, including why such nets were deemed illegal. It is necessary to manually keep track of such nets and correct them along the way.

Disclaimers

 **Caution!**

ECO commands are intended for advanced users only. **Use at your own risk!**

ECO commands are intended to be performed at the end of the place-and-route [Flow Steps \(page 234\)](#). Performing ECO before or during place-and-route is possible, but this requires more caution.

Although optional, it is highly recommended to specify the name of a site when inserting an instance, or else ACE refuses to perform net-routing with nets connected to unplaced instances. If desired, optionally execute `run_place` after instance insertion to let ACE handle placement automatically. The flow step `run_place`:

- May perform automatic placement of an ECO-inserted instance, but if the new instance is deemed redundant or useless by the ACE reconditioner, it is removed from the design.
- Is not incremental; it always places all instances in the design, which can be time consuming. As such, when using `ace_eco::insert_instance`, consider placing the new instance manually.

While performing ECO, certain nets may become partially routed (i.e., calling `rewire_net -connect` without specifying `-reroute`) or derouted (i.e., the nets lost a pin connection that kept them legal). In these circumstances, a warning is displayed on the Tcl console. All such illegal nets must be identified and resolved before proceeding.

When any illegal nets have been made legal (a driver added to a floating net, a sink pin added to a dangling net, drivers removed from a multi-driven net), `rewire_net <netName> -reroute` must be called to physically create the connections specified in the ECO-modified netlist.

If it is decided to perform `run_prepare` after performing ECO, all changes made are overwritten since `run_prepare` sources the original netlist(s) specified in the ACE project, but not any ECO modifications. As such, keep in mind that all work might be accidentally undone by calling `run_prepare`.

Running ECO commands but failing to resolve issues created by the resulting changes may potentially lead to errors within ACE if certain functions are subsequently called. For instance, a set of ECO commands could disconnect an output pin from a net, but fail to connect a new one. If another (non-ECO) command is then called that assumes that each net has a driving pin, ACE reports errors.

ECO commands modify the gate-level netlist generated from `run_prepare`, and rerunning `run_place` or `run_route` with a modified netlist may lead to unintended consequences to the design.



Tip

It is recommended to save often when performing ECO as inadvertently executing a wrong command may either break the design beyond repair or cause ACE to report errors, forcing a restart from scratch.

ECO Commands

The ECO commands are all in a special `ace_eco::Tcl` Namespace; they are not included in the global namespace. These commands are

- `ace_eco::delete_instance`
- `ace_eco::delete_net`
- `ace_eco::get_instance_pins`
- `ace_eco::insert_instance`
- `ace_eco::insert_net`
- `ace_eco::rewire_instance`
- `ace_eco::rewire_net`

delete_instance

Command Syntax

```
ace_eco::delete_instance {<i:instance_name> <i:instance_name> ...} [-reroute]
```

The `delete_instance` command deletes the named instances and disconnects the pins of those instances from their respective nets. Any nets left with no connections are not automatically deleted. These nets must be deleted manually using `ace_eco::delete_net`.

Table 125 • delete_instance Arguments

Arg Name	Optional	Description
<instances>	No	List of user design instances to be deleted. The "i:" prefix is optional on each instance.

Arg Name	Optional	Description
<code>[-reroute]</code>	Yes	The optional <code>-reroute</code> argument is used to re-route nets after the instances have been deleted.

delete_net

Command Syntax

```
ace_eco::delete_net {<n:net_name> <n:net_name> ...}
```

The `delete_net` command deletes the named nets.

Table 126 - delete_net Arguments

Arg Name	Optional	Description
<code><nets></code>	No	List of user design nets to be deleted. The "n:" prefix is optional on each net.

get_instance_pins

Command Syntax

```
ace_eco::get_instance_pins {<i:instance_name> <i:instance_name> ...}
```

The `get_instance_pins` command returns a list of all pins (and nets, if connected) for the named instances. The returned lists takes the form of:

```
{{t:instance1:pin1 n:net1} {t:instance1:pin2 n:net2} {t:instance1:disconnected_pin3}}
{{t:instance2:pin1 n:net1} ...} ...
```

Table 127 - get_instance_pins Arguments

Arg Name	Optional	Description
<code><instances></code>	No	List of user design instances to be queried. The "i:" prefix is optional on each instance.

insert_instance**Command Syntax**

```
ace_eco::insert_instance <i:instance_name> <cell_type_name> [-site <s:site_name>] [-pins
{{<p:pin_name> <n:net_name>} {<p:pin_name> <n:net_name>} ...}] [-parameters
{{<param_name> <param_value>} {<param_name> <param_value>} ...}] [-fixed] [-reroute]
```

The `insert_instance` command generates a new instance of the specified cell type and inserts it into the netlist. If `-site` is specified, the command places the new instance on the named site (given that the site is legal to use).

Table 128 • insert_instance Arguments

Arg Name	Optional	Description
<instance_name>	No	The name which is given to the newly inserted instance. The "i:" prefix is optional.
<cell_type_name>	No	Type of instance. This must be a valid cell type for the current fabric.
-site	Yes	The optional <code>-site</code> argument names the site where the new instance is placed. The site named must be compatible with the cell type, or the command aborts before the instance is created. The "s:" prefix is optional.
-pins	Yes	The optional <code>-pins</code> argument specifies a list of pin name/net name pairs. Each named pin is connected to the associated named net. The "p:" and "n:" prefixes are optional.
-parameters	Yes	The optional <code>-parameters</code> argument specifies a list of user parameters and values for the new instance. The parameters must be compatible with the cell type, or the command aborts before the instance is created.
-fixed	Yes	The optional <code>-fixed</code> argument indicates that given a <code>-site</code> parameter, the newly created instance is fixed to that site. By default, Soft Placement is performed (no fixing).
-reroute	Yes	The optional <code>-reroute</code> argument is used to re-route nets after the instance has been inserted.

Warning!

- The instance name is currently not verified to follow Verilog/VHDL identifier standards.
- Although optional, it is highly recommended to specify the name of a site when inserting an instance; otherwise, ACE refuses to route pins with an unplaced instance.

insert_net**Command Syntax**

```
ace_eco::insert_net <n:net_name> {<t:instance_name:pin_name>
<t:instance_name:pin_name> ...} [-route]
```

The `insert_net` command creates a new net and insert it into the netlist. This new net must have at least two connections specified, and these connections must make the net legal.

Table 129 - insert_net Arguments

Arg Name	Optional	Description
<net_name>	No	The name to be given to the newly inserted net. The "n:" prefix is optional.
<pins>	No	List of fully-qualified (including instance) pin names. Each named pin is connected to the new net. The "t:" pin prefixes are optional.
-route	Yes	The optional <code>-reroute</code> argument is used to automatically route the new net after it has been inserted.

Warning!

- The new net name is currently not verified to follow Verilog/VHDL identifier standards.
- The `insert_net` command does not connect to pins already connected to a net; those pins must be disconnected first before calling this command.
- The user-specified net must be legal (has at least one input pin and exactly one driving pin) upon creation, or else the net is not created. However, instances which the new net connects to do not have to be placed.

rewire_instance**Command Syntax**

```
ace_eco::rewire_instance <i:instance_name> "{<p:pin_name> [n:net_name]} {<p:pin_name>
[n:net_name]} ..." [-reroute] [-disconnect]
```

The `rewire_instance` command allows connecting or disconnecting the pins of an instance to/from specific nets. Both of these operations can be performed at the same time. Connections from user-specified nets to user-specified pins may be created, deleted, or changed.

Table 130 • rewire_instance Arguments

Arg Name	Optional	Description
<instance_name>	No	The name of the instance whose wiring is to be changed. The "i:" prefix is optional.
<pin_net_pairs>	No	List of user design pins/ports paired with the optional nets to which each are to be connected. The "p:" and "n:" prefixes are optional. Example: '{pin1 net1} {disconnected_pin2}'
-reroute	Yes	The optional <code>-reroute</code> argument is used to re-route nets after the instance has been inserted.
-disconnect	Yes	The optional <code>-disconnect</code> argument causes all existing connections to be disconnected before applying new pin/net connections.

Note

- If a named pin/net connection already exists as specified, that argument is safely ignored.
- If a pin name is provided without a net name, that pin is disconnected from any currently connected net.
- If `-disconnect` is used with an empty `pin_net` list, all prior connections are removed without any new connections being created.

rewire_net

Command Syntax

```
ace_eco::rewire_net <n:net_name> [-connect "<p:pin_name> | <n:net_name> ... "] [-
disconnect "<p:pin_name> ..." ] [-clocktype <type>] [-reroute] [-verbose]
```

The `rewire_net` command allows connecting/disconnecting pins from the specific net. The same action could also be accomplished with `ace_eco::rewire_instance` commands, but that can quickly become very cumbersome if most of the pins on one net should now connect to another net. A single `ace_eco::rewire_net` command can do the work that would require hundreds of `ace_eco::rewire_instance` commands, each specifying the same net.

Note

The command `rewire_net` may be used on a net with no arguments except for `-reroute`, to perform routing on the specified net. This option is useful for cleanup work as it is necessary to route any remaining partially or unrouted nets.

Table 131 • *rewire_net* Arguments

Arg Name	Optional	Description
<net_name>	No	The name of the net whose wiring is to be changed. The "n:" prefix is optional.
-connect	Yes	The optional <code>-connect</code> argument specifies a list of pins to be connected to a net. If already connection, they are first disconnected from their original nets.
-disconnect	Yes	The optional <code>-disconnect</code> argument specifies a list of pins that should be disconnected from net.
-clocktype	Yes	The optional <code>-clocktype</code> argument indicates the clock type of the pins to be connected.
-reroute	Yes	The optional <code>-reroute</code> argument is used to re-route nets after the instance has been inserted.
-verbose	Yes	The optional <code>-verbose</code> argument generates additional feedback as the command is running.

GUI Support

GUI support for ECO functionality is hidden by default. To enable ECO actions in the GUI, enable the checkbox found at: **Window** → **Preferences** → **User Advanced Preferences** → **Enable ECO Functionality**.

When enabled, ECO actions for the ECO commands appear in right-click context menus available on most views in the Floorplanner Perspective. For example, right-click a net to display the available ECO net commands; right-click an instance to display available ECO instance commands, etc.

Warning!

The GUI ECO wizards do not provide extra safety checks or guidance at this time. Errors, warnings, and success feedback from the ECO changes are only shown in the **Tcl Console View** (page 142) as the ACE ECO Tcl commands themselves are executed by the wizards.

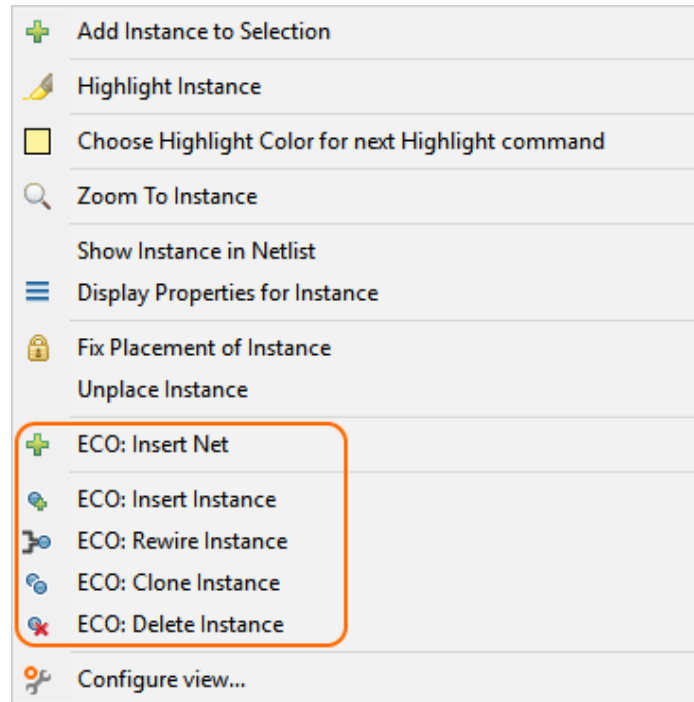


Figure 123 • Example of ECO Instance Commands in Context Menu

Add Instance Pin Dialog

The Add Instance Pin Dialog selects an existing pin to add to an existing instance.

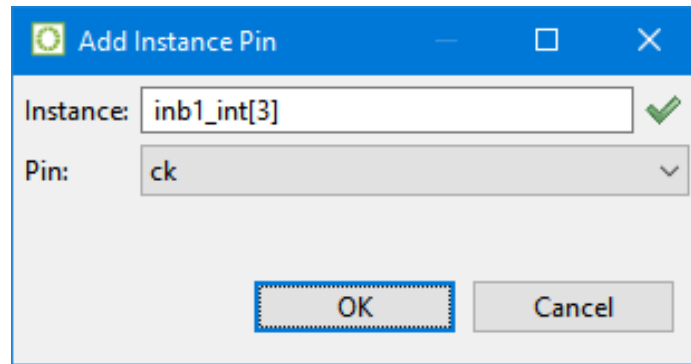


Figure 124 - Add Instance Pin Dialog Example

Table 132 - Add Instance Pin Dialog Fields

Field	Description
Instance	A valid instance name. If an invalid name is specified, the green check mark next to the Instance field changes to indicate the error.
Pin	A list of the instance pins.

ECO Insert Instance Dialog

The ECO Insert Instance Dialog allows the addition of a new instance.

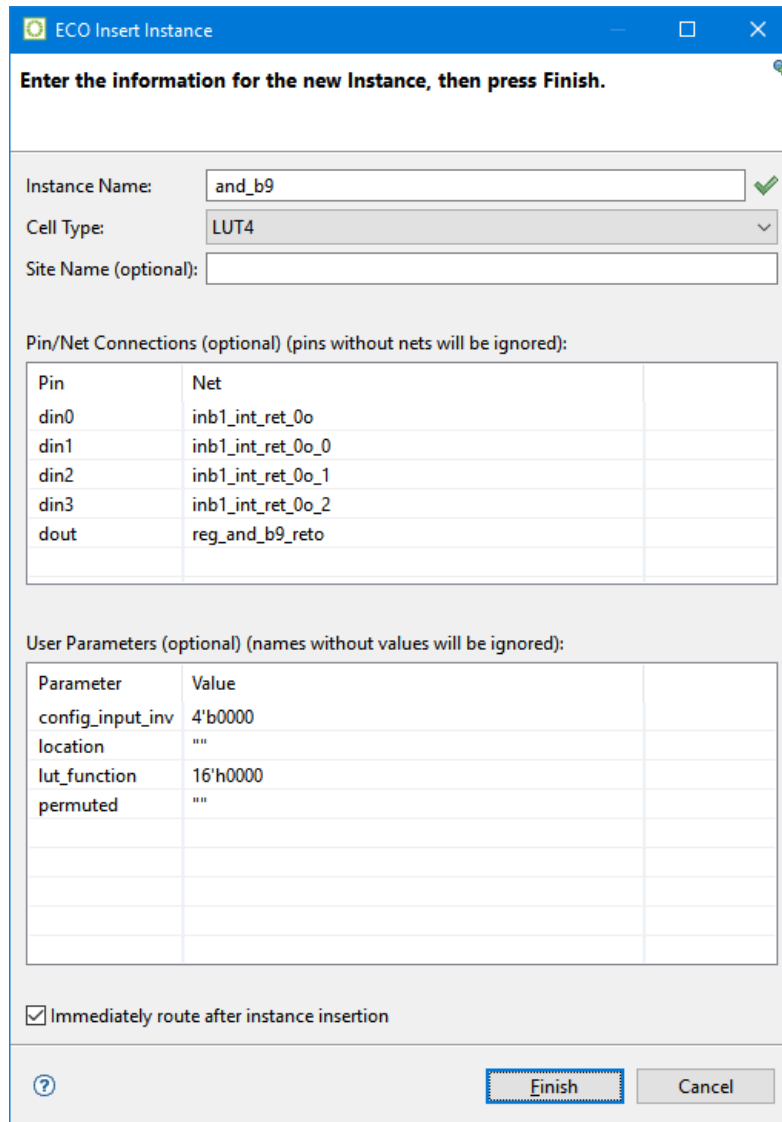


Figure 125 • ECO Insert Instance Dialog Example

Table 133 • ECO Insert Instance Dialog Fields

Field	Description
Instance Name	The name for the new instance. The name must be unique to the design. If an instance with the given name already exists, the green check mark next to the Instance Name field changes to indicate the error.

Field	Description
Cell Type	The cell type for the new instance.
Site Name (optional)	The name of the site where the new instance should be placed.
Pin/Net Connections	Click a cell in the Net column to connect a pin to an existing net.
User Parameters (optional)	Click a cell in the Value column to specify a parameter value. Parameters without values are ignored.
Immediately route after instance insertion	If enabled, the design is rerouted immediately after the new instance is inserted.

ECO Insert Net Dialog

The ECO Insert Net Dialog allows the addition of a new net.

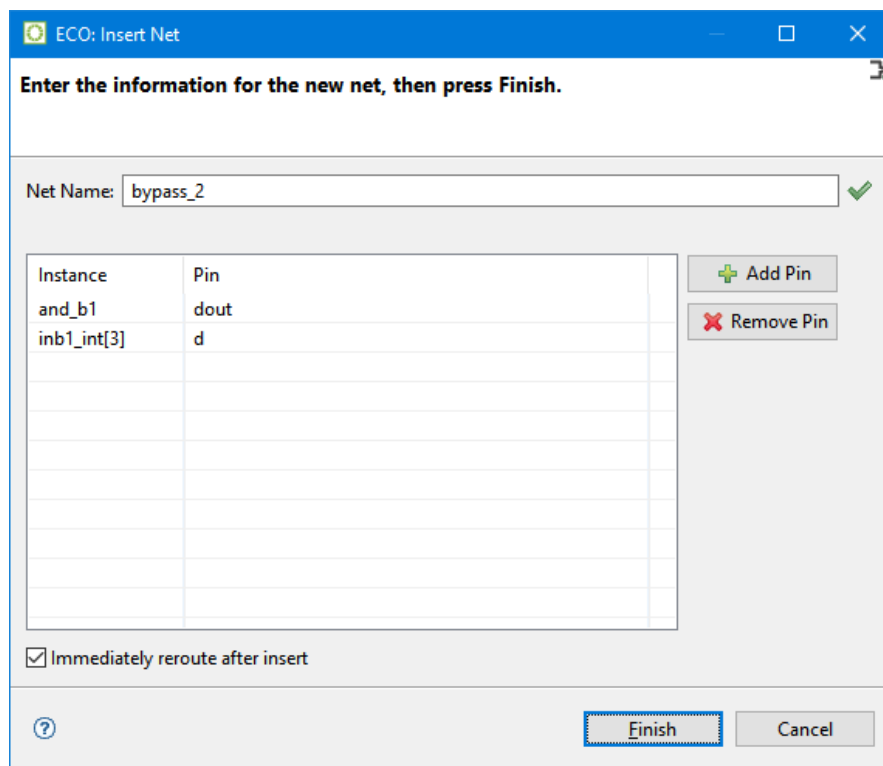


Figure 126 - ECO Insert Net Dialog Example

Table 134 - ECO Insert Net Dialog Fields

Field	Description
Net Name	The name for the new net. The name must be unique to the design. If a net with the given name already exists, the green check mark next to the Net Name field changes to indicate the error.
Add Pin	The Instance Pins table in the dialog lists the pins to which the new nets are connected. Use the Add Pin button to add pins to this table.
Remove Pin	Use the Remove Pin button to remove all currently selected pins from the Instance Pins table.
Immediately reroute after insert	If enabled, the design is rerouted immediately after the new net is inserted.

ECO Rewire Instance Dialog

The ECO Rewire Instance Dialog allows adjusting the properties of an existing instance.

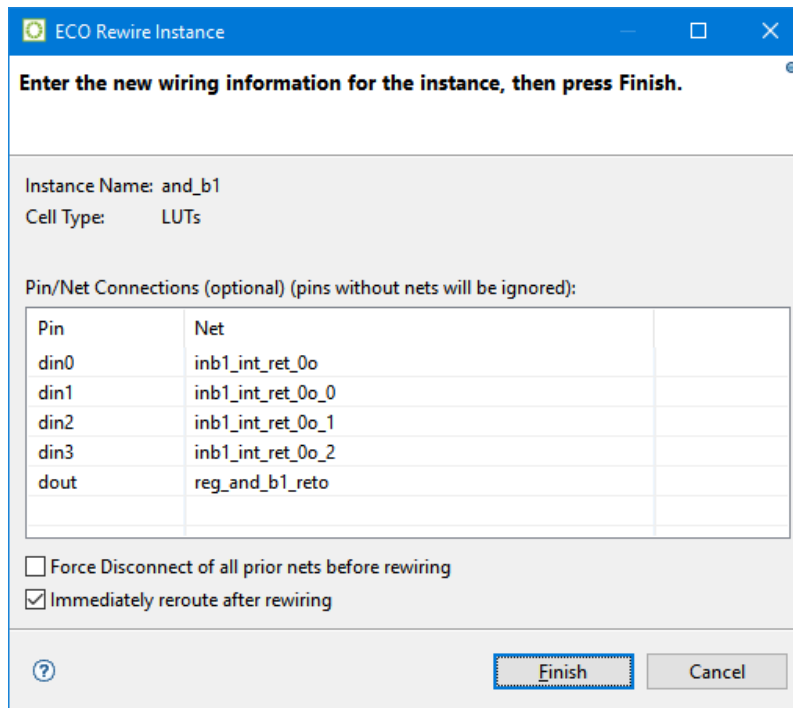


Figure 127 - ECO Rewire Instance Dialog Example

Table 135 - ECO Rewire Instance Dialog Fields

Field	Description
Pin/Net Connections	Click a cell in the Net column to connect the pin to a different net.
Force disconnect of all prior nets before rewiring	Causes all existing connections to be disconnected before applying new pin/net connections.
Immediately route after instance insertion	If enabled, the design is rerouted immediately after the new instance is inserted.

ECO Rewire Net Dialog

The ECO Rewire Net Dialog allows an existing net's pin connections to be adjusted.

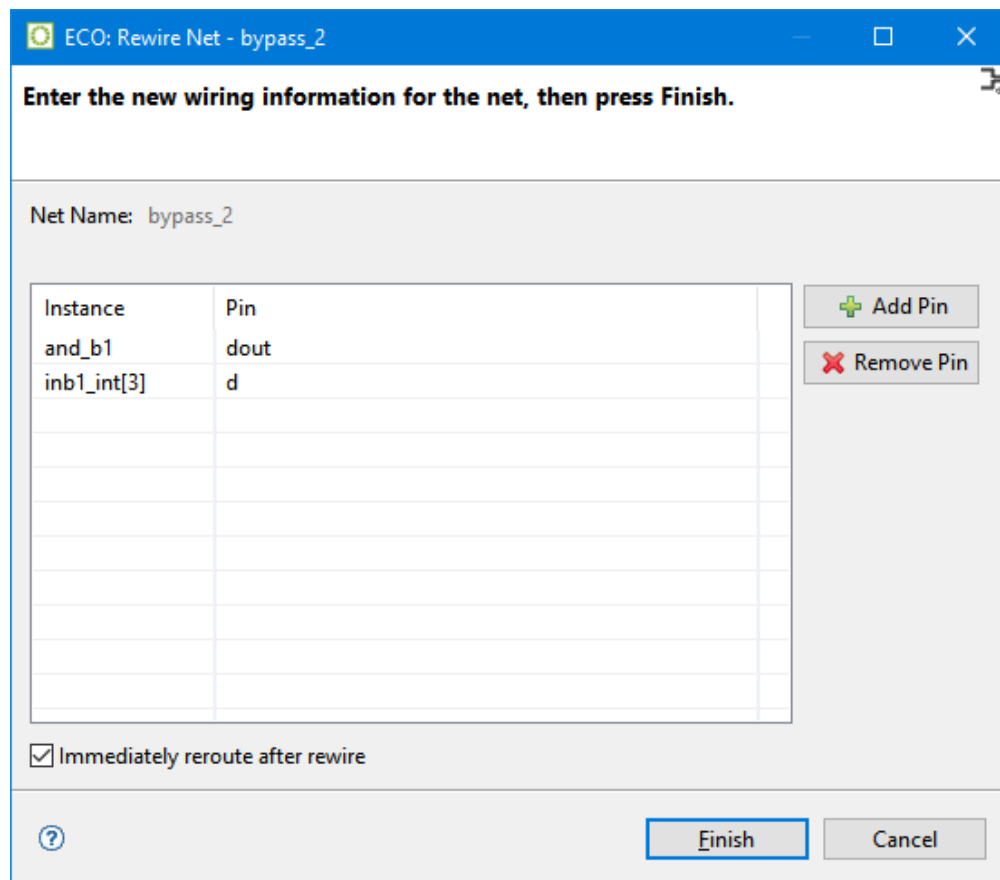


Figure 128 - ECO Rewire Net Dialog Example

Table 136 - ECO Rewire Net Dialog Fields

Field	Description
Add Pin	The Instance Pins table in the dialog lists the pins to which the new net is connected. Click this button to add pins to the table.
Remove Pin	Click this button to remove all currently selected pins from the Instance Pins table.
Immediately reroute after insert	If enabled, the design is rerouted immediately after the Insert Net dialog is completed.

Fabric Clusters

Device fabrics in the Achronix Speedster and Speedcore families consist of a device-specific pattern of functional resources called tiles. For example, reconfigurable logic block (RLB) tiles contain LUT, DFF, and ALU sites. BRAM tiles contain block RAM sites. The tiles are arranged in rows and columns, with all columns in the device core (not including the I/O ring) consisting of identical tile types. The arrangement of tiles into columns may look somewhat irregular (i.e., 5 columns of LUTs, a BRAM column, 5 more columns of LUTs, and an LRAM column). However, at a coarse level, those tile arrangements do follow a regular pattern. All fabrics are constructed by arranging rows and columns of tiles into a basic unit of layout called a fabric cluster. A complete device is created by replicating the fabric clusters into a larger grid of rows and columns. For example, the Speedster7t AC7t1500 has 8 rows and 10 columns of fabric clusters, for a total of 80. See the *Speedster7t FPGA Datasheet (DS015)* for additional details.

All fabric clusters are identical, containing exactly the same pattern of tile rows and columns. The exact numbers of fabric clusters, the dimensions of each cluster, and the arrangement and types of tiles within each cluster, are all specified by the chosen target device. A designer can make use of this regularity if the design consists of a number of identical cores. If the cores are designed to fit exactly within one or more fabric clusters, a complete design can be created by replicating that core multiple times across the device using the fabric cluster grid.

The use of **placement regions and placement region constraints** (page 394) might be necessary to guide the placement of multi-core designs in order to align them with the fabric cluster grid. This should be discussed thoroughly with an Achronix FAE first, as improper use of placement region constraints can lower QOR or even cause placement or routing to become unsolvable.

Chapter 4 : Tasks

While the [Concepts \(page 6\)](#) section was primarily concerned with which features exist in ACE, this Tasks section is concerned with how features in ACE may be best utilized.

Running ACE

ACE can be run with full functionality in three different modes:

- [GUI Mode \(page 282\)](#)
- [Command-line Mode \(page 282\)](#)
- [Batch Mode \(page 283\)](#)

A fourth mode, [Lab Mode \(page 283\)](#), is also available, with reduced functionality.

An optional [ACE_INIT_SCRIPT \(page 284\)](#) can be configured to load a user-defined Tcl script during ACE startup, prior to running any other Tcl commands or scripts. This enables users, or groups of users, to customize ACE settings, define custom Tcl procs, or define custom ACE flow steps.

Finally, a table of typical supported [ACE startup arguments \(page 285\)](#) is provided at the end of this section.

GUI Mode

To run in GUI mode, invoke the `ace` executable either with no options or with the `-gui` option. GUI mode launches the interactive GUI, from which all commands are issued.

Starting ACE in GUI Mode, implicit

```
% ./ace
```

or

Starting ACE in GUI Mode, explicit

```
% ./ace -gui
```

Command-line Mode

To run in command-line mode, invoke the ACE executable with the `-b` option from a console. Command-line mode takes control of the console and allows interactively entering Tcl commands at a command prompt.

Starting ACE in Command-line Mode

```
% ./ace -b
-- ACE -- Achronix CAD Environment -- Version 5.4 -- Build 84486- -- Date 2015-02-11
19:58
-- (c) Copyright 2006-2015 Achronix Semiconductor Corp. All rights reserved.
-- all messages logged in file /home/username/.achronix/ace_2015_02_13_11_00_11.log,
created at 11:00:11 on 02/13/2015
INFO: License ace-v1.0 on server acxlicense (9 of 10 licenses available). Running on
docs.achronix.local (x86_64).
ACE>
```

Batch Mode

To run in batch mode, invoke the ACE executable with the `-b` option and the `-script_file` option.

Starting ACE in Batch Mode

```
% ./ace -b -script_file path_to_script_file.tcl
```

Lab Mode (Reduced Functionality)

ACE also supports a reduced functionality mode, intended for use in Lab environments. The primary purpose of this mode is to allow a lighter-weight tool for chip programming and debugging, and/or for demonstrating hardware functionality with demo designs. In this mode, a license is not required (no license check occurs), and thus most Tcl functionality is unavailable (because most ACE Tcl functionality requires an appropriate license). In lab mode, the user is unable to work with project files, run the Flow, view the Floorplanner, configure IP, etc.

When the ACE GUI is in lab mode, only the default views within the Programming and Debug Perspective and HW Demo Perspective are usable. Only the subset of ACE Tcl commands needed to support those views is functional. The views within the Projects Perspective, Floorplanner Perspective, IP Configuration Perspective, and NoC Performance Perspective are non-functional (and inaccessible), since these views and editors require licensed ACE functionality.

Starting ACE GUI in Lab Mode

```
% ./ace -lab_mode
```

or

Starting ACE in Command-line Lab Mode

```
% ./ace -b -lab_mode
-- ACE -- Achronix CAD Environment -- Version 8.8 -- Build xxxx -- Date 2022-xx-xx xx:xx
-- (c) Copyright 2006-2022 Achronix Semiconductor Corp. All rights reserved.
-- all messages logged in file /home/username/.achronix/ace_2022_08_09_10_11_12.log,
created at 10:11:12 on 08/09/2022
ACE>
```

or

Starting ACE in Batch Lab Mode

```
% ./ace -b -lab_mode -script_file path_to_jtag_script_file.tcl
```

ACE Initialization Script (ACE_INIT_SCRIPT)

An optional ACE_INIT_SCRIPT can be configured to load a user-defined Tcl script during ACE startup, prior to running any other Tcl commands or scripts. This enables users, or groups of users, to:

- Customize ACE settings
- Define custom Tcl procedures/commands
- Define custom ACE flow steps
- Extend ACE features and functions

There are three ways to enable the ACE_INIT_SCRIPT feature in ACE, which use the following order of precedence:

1. Use the `-ace_init_script` commandline. This is useful when creating wrapper scripts to call ACE with an explicit ACE_INIT_SCRIPT, or when debugging or testing different ACE_INIT_SCRIPTs.
2. Define the `$ACE_INIT_SCRIPT` environment variable as the file path to your ACE initialization Tcl script. This can be useful for development teams or organizations wanting to share and standardize the use of the ACE_INIT_SCRIPT across multiple ACE users.
3. Define the `$ACE_INIT_SCRIPT` environment variable as an empty string. This causes ACE to look for the ACE_INIT_SCRIPT in `$HOME/.achronix/ace_init.tcl`. This is useful for an individual developer to customize ACE specifically for that individual, without impacting other users.

Relative paths

If a relative path is used to specify the file path to the ACE_INIT_SCRIPT Tcl file, that path is evaluated relative to the current working directory from which ACE was launched (which need not be the ACE installation directory).

Example Log Output

To confirm the ACE_INIT_SCRIPT was loaded in your ACE session properly, look for the "Running initialization script" message near the beginning of the ACE session log file.

Starting ACE in Command-line Lab Mode

```
% ./ace -b -ace_init_script ./my_init_script.tcl
-- ACE -- Achronix CAD Environment -- Version x.x -- Build xxxx -- Date 2023-xx-xx xx:xx
-- (c) Copyright 20xx-20xx Achronix Semiconductor Corp. All rights reserved.
-- all messages logged in file /home/username/.achronix/ace_20xx_xx_xx_xx_x_xx.log,
created at xx:xx:xx on xx/xx/xxxx
INFO: Running initialization script /full/path/to/my_init_script.tcl
ACE>
```

ACE Startup Arguments

The most common startup arguments are listed in the following table. Other arguments exist, but should only be used under the guidance of an Achronix FAE (or as specified in one of the Troubleshooting sections of an Achronix User Guide).

Table 137 • Common ACE Startup Arguments

Argument	Description
-b or -batch	Starts ACE in non-GUI mode, also known as Batch or Command-line mode. See also: the <code>-script_file</code> argument
-lab_mode	Starts ACE in (unlicensed) Lab Mode, with limited functionality. May be used in combination with ACE GUI mode, Command-line mode, and/or Batch mode. See the prior section in this chapter for more details about ACE Lab Mode.
-log_file <path_to_log_file>	Use this to override default ACE session logging behavior, which would otherwise default to creating a session log file at <user_home_dir>/ .achronix/ace_<datestamp>_<timestamp>.log. A session log file covers all ACE activity across all projects and implementations touched during the process lifetime of ACE. Keep in mind that the <code>-log_file</code> argument has no impact upon implementation log files; individual implementation log files are always created under each implementation directory whenever that implementation is being accessed. There is currently no way to override the path of the implementation log files.
-print_progress	Enables verbose progress messages to be logged during execution of the Flow. Verbose progress logging is disabled by default.
-project_file <path_to_acxprj_file>	Allows pre-specifying which *.acxprj file should be loaded by ACE at startup. This is similar to immediately calling the following ACE Tcl command at the Tcl command prompt after ACE has started (after the <code>-ace_init_script</code> , but before the <code>-script_file</code>): <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <pre>load_project <path_to_acxprj_file></pre> </div>

Argument	Description
<code>-ace_init_script</code> <code><path_to_ace_init_script_file></code>	An optional ACE_INIT_SCRIPT can be configured to load a user-defined Tcl script during ACE startup, prior to running any other Tcl commands or scripts. This option overrides the \$ACE_INIT_SCRIPT environment variable and any init script in the user home workspace directory.
<code>-script_file</code> <code><path_to_script_file></code> <code>[-script_args <string>]</code> ⁽¹⁾	After the ACE session is initialized (after any ACE_INIT_SCRIPT is executed, and after any designated <code>-project_file</code> is loaded), ACE executes the script contained in the script file. If the optional <code>-script_args</code> <code><string></code> is also used, then the arguments are passed to the script file when the script is executed. To access these arguments in a TCL script, use the special TCL variables <code>\$argc</code> and <code>\$argv</code> . These variables are analogous to the top-level arguments to the C main function, but with the application name itself instead stored in the special TCL variable <code>\$argv0</code> .
<code>-snapshot_version</code> <code><int></code>	Allows specifying which version of Achronix Snapshot should be used during the ACE session. Version 3 was the old *.jam/STAPL support, no longer used starting in ACEv9.2. Version 4 is the new Tcl support available starting in ACEv9.2.
<code>-version</code> ⁽²⁾	Intended for use in ACE Batch mode. Requests that ACE report its version number and then immediately exit. Example: <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p style="text-align: center;">Using the -version argument (in Linux)</p> <pre>% ./ace -b -version -- ACE -- Achronix CAD Environment -- Version 8.8 -- Build xxxx -- Date 2022-xx-xx xx:xx -- (c) Copyright 2006-2022 Achronix Semiconductor Corp. All rights reserved. %</pre> </div>

Table Notes

1. When using script files in Microsoft Windows, starting ACE in batch mode with a script file opens a new console that only survives for the life of the script file, and then immediately closes. This typically makes it difficult to read any logged results which might have appeared in that console. For best results, be sure your script logs its output to a file, or review the session log file after script execution. See also: the `-log_file` argument.
2. This argument has no effect in GUI Mode. ACE always logs the version number to the Tcl Console immediately at ACE startup in all modes. As when using the `-script_file` argument in Windows, ACE starts a separate console window for the session. Because ACE immediately exits after reporting/logging the version banner, this console window is only visible for a fraction of a second. The requested version information can be found in the session log file. See also: `-log_file` argument.

Working With Perspectives

Perspectives define the initial set and layout of views in the Workbench window, providing a set of functionality aimed at accomplishing a specific type of task or working with specific types of resources.

Switching Between Perspectives

Each perspective has an associated icon on the main toolbar. Switch between perspectives by clicking one of these icons.

It is also possible to view the available choices as a menu by selecting **Window** → **Open Perspective** from the menu bar.

Descriptions of the available perspectives are found in the section describing the [Perspectives \(page 6\)](#) concept.

Resetting Perspectives

Often, when altering positions of [Editors \(page 9\)](#) and [Views \(page 16\)](#) within a perspective, this can result in an arrangement that is no longer appealing. Rather than try to manually move the Views and Editors back to the original positions, it is much faster and simpler to reset the perspective.

To restore a perspective to its original layout:

1. Select **Window** → **Reset Perspective** from the menu bar.
2. Click **OK** on the resulting dialog.

Working with Views and Editors

Views and editors are the main visual entities appearing in the Workbench. In any given perspective there is a single editor area which can contain multiple editors with a number of surrounding views providing context.

Opening Views

Perspectives offer pre-defined combinations of views and editors. To open a view not included in the current perspective, select **Window** → **Show View** from the menu bar.

Moving and Docking Views and Editors

To change the location of a view or editor in the current perspective:

1. Without releasing the left mouse button, drag the view or editor by its tab.


 **Note**

A group of stacked views or editors can be dragged using the empty space to the right of the tabs.

2. While dragging the tab (or tab stack), as the mouse is moved around the Workbench, the area under the mouse changes to display (sometimes subtle) feedback indicating where the tab (or stack) can dock if the left mouse button is released at the current location.
 - Drag the tab near the left, right, top, or bottom border of another view or editor to see how that view/editor splits its available area with the dragged tab.
 - Drag the tab near the tabs of another tab stack to see where the dragged tab can be inserted/appended in the existing stack.
 - A tab may be dragged outside of the Workbench area to turn it into a detached view (a view shown in its own separate window).

- When the view is in the desired location relative to the view or editor area under the cursor, release the left mouse button.

Table 138 • View and Editor Tab Docking Feedback

Feedback	Description
Vertical bar between tabs	Marks the insertion point between other tabs.
Translucent rectangles overlaid upon existing view/editor	Shows the positioning of the dragged view/editor alongside the pre-existing views/editors already in that docking location.
Translucent rectangle floating outside the ACE window	Shows the position where the detached (page 288) view/editor can appear.
	No changes. This docking location is either identical to the present layout, or is an illegal position. If the left mouse button is released, no change occurs.

⚠ Caution!

In Linux, there is currently a known bug, in the application frameworks underlying ACE, that may cause view/editor tab movements to **detach** (page 288) instead of docking when the Help Window is open. See the **Troubleshooting** (page 761) section for more details, including several workarounds.

Rearranging Tabbed Views and Editors

In addition to dragging and dropping (docking) views/editors inside the Workbench, the order of views/editors can be rearranged within a tabbed stack:

- Click the tab of the view/editor to be moved and drag it to the desired location. As the tab is dragged across other tabs, a vertical bar insertion cursor appears.
- Release the mouse button when the insertion cursor is in the desired location.

i Note

A group of stacked views/editors can be moved by starting the drag using the empty space to the right of the tabs.

Detaching Views and Editors

Detached views and editors are shown in a separate window with a smaller trim. These views work like other views and editors, except that they are always shown in front of the Workbench window. To detach views/editors:

- If the Workbench window is maximized, resize it so that it does not fill the entire screen.
- Click and hold the tab of the view/editor to be detached.

3. Drag the tab (or tab group) outside of the Workbench window and release the mouse button. The tab can also be dragged into the window of a previously detached view/editor to have multiple detached views/editors together.

To restore the view/editor to appear inside of the Workbench window, drag its tab into the Workbench window.

Tiling Editors

The Workbench allows multiple files to be open in multiple editors. Unlike views, editors cannot be dragged outside the Workbench to create new windows. However, editor sessions can be tiled within the editor area in order to view source files side by side:

1. With two or more files open in the editor area, select one of the editor tabs.
2. While holding down the left mouse button, drag that editor tab over the left, right, top or bottom border of the editor area. The mouse pointer changes to a drop cursor, indicating where the editor session is to be moved.
3. Release the mouse button.
4. Optionally, Drag the borders of the editor area, or each editor, to resize as desired.

Note

This operation is a similar to moving and docking views inside the Workbench, except that all editor sessions must be contained within the editor area.

Maximizing, Minimizing, and Restoring Views and Editors

ACE provides a rich environment which, in its basic form, consists of the following:

- An Editor Area which contains one or more stacks showing the open editors
- One or more View Stacks which surround the Editor Area and contain one or more views

These elements compete for valuable screen area and correctly managing the amount of area given to each can greatly enhance your productivity within ACE.

The two most common mechanisms for managing this issue are "minimize" (use as little space as possible) and "maximize" (provide as much space as possible). ACE provides two ways to access these operations:

1. Use the minimize and maximize buttons provided on a stack border.
2. Double-click an individual tab or the blank area to the right of the tabs.

Maximize:

It is desirable at times to focus attention on one particular view or editor to the exclusion of the others. The most popular candidates for this are maximizing the editor area in order to view a report, or maximizing the **Floorplanner View** (page 43) to make as much of the display available for floorplanning as possible.

ACE implements the maximize behavior by minimizing all stacks *except* the one being maximized. This allows the maximized stack to completely occupy the window while still allowing access to any open views in the perspective by using the icons located in the area around the edges of the window which are called the "Trim Stack".

Editor maximization operates on a complete Editor Area which includes all Editor Stacks, rather than simply maximizing the particular Editor Stack. This allows for "compare" workflows which require the ability to see multiple editor files in a split editor area at the same time.

Minimize:

Another way to optimize the use of the screen area is to directly minimize stacks that are of no current interest. Minimizing a stack causes it to be moved into the trim area at the edges of the workbench window, creating a Trim Stack.

Note

The first time a stack is minimized, the Trim Stack may end up on any edge of the window. If the Trim Stack is manually moved to a particular window edge, that same edge is typically reused when that stack is again minimized.

View Stacks get minimized into a trim representation that contains the icons for each view in the stack:

Clock Domain Name	Hig...	Core IO ...	ALUs	BRAMs	CLK_IPI...	CLK_O...	DSPs
clka		✓	0	0	1	0	0

Figure 129 - Example View Stack Before Minimization

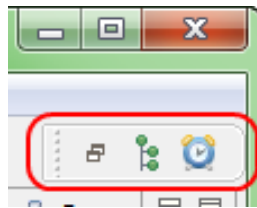


Figure 130 - Example Trim Stack After View Stack is Minimized

The minimize behavior for the Editor Area is somewhat different. Minimizing the Editor Area results in a trim stack containing only a placeholder icon representing the entire editor area rather than icons for each open editor (since in most cases all of the icons would be the same, making them essentially useless).

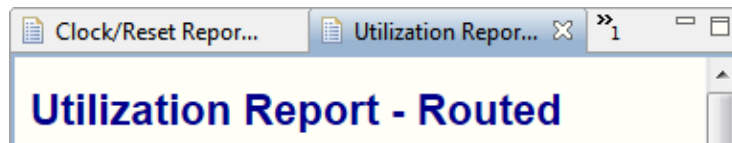


Figure 131 - Example of Editor Area Before Minimization



Figure 132 - Example of Minimized Editor Area Trim Stack

For workflows needing more than one element visible (i.e., having the Editor Area *and* a View Stack in the presentation at the same time) additional screen space can still be gained by minimizing the stacks that are not of current interest. This removes them from the main presentation and places them on the Trim Stack, allowing more space for the remaining stacks in the window.

Note

There are two ways to end up with a stack in the trim:

1. Directly by minimizing the stack.
2. Indirectly by maximizing another stack.

Depending on how the Trim Stack was created, its behavior is different: when restoring a stack from a maximized state, only those trim stacks that were automatically minimized during the initial maximize are restored to the main presentation, while stacks that were manually minimized, stay minimized.

Tip

This difference is important in that it allows fine-grained control over the presentation. While using maximize is a one-click operation, it is an "all or nothing" paradigm (i.e., no other stack is allowed to share the presentation with a maximized stack). While adequate for most tasks, it may be desired to have the presentation show more than one stack. In this case, instead of maximizing, minimize all the other stacks *except* the ones wanted in the presentation. When set up, the editor area can still be subsequently maximized, but the subsequent restore only restores the particular stack(s) that were sharing the presentation, not the ones explicitly/manually minimized.

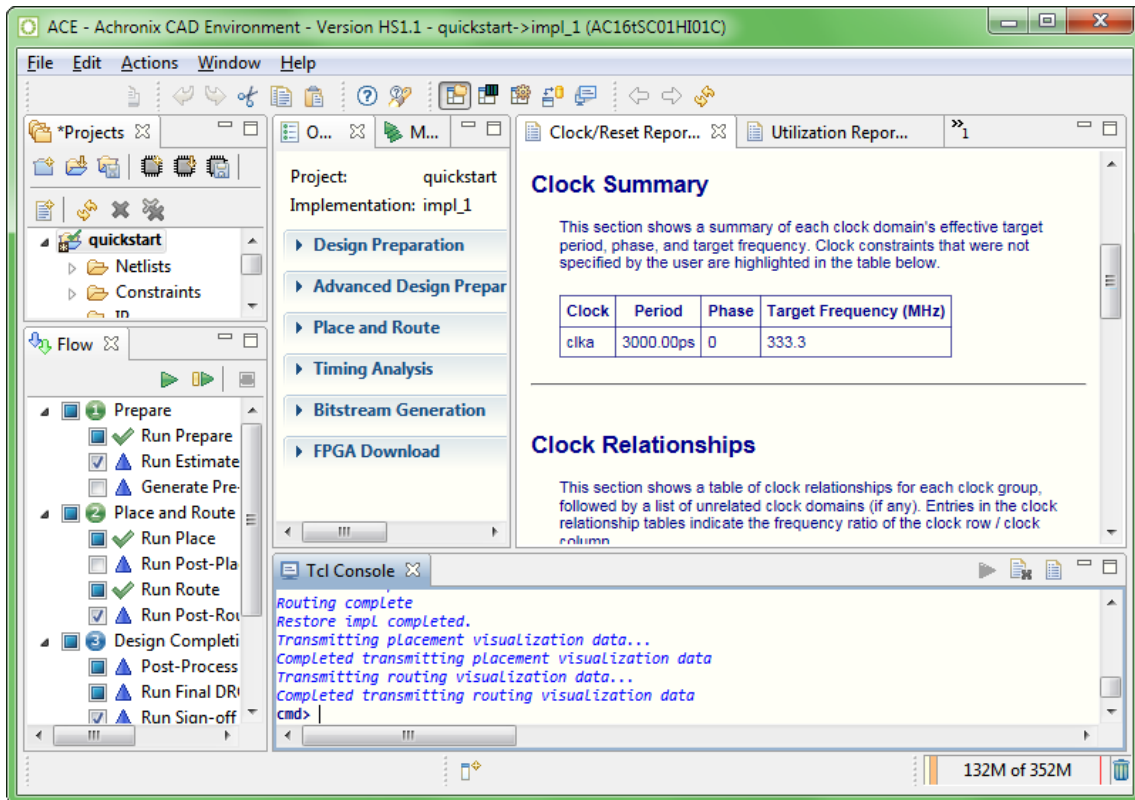


Figure 133 • Example Default Presentation of the Projects Perspective

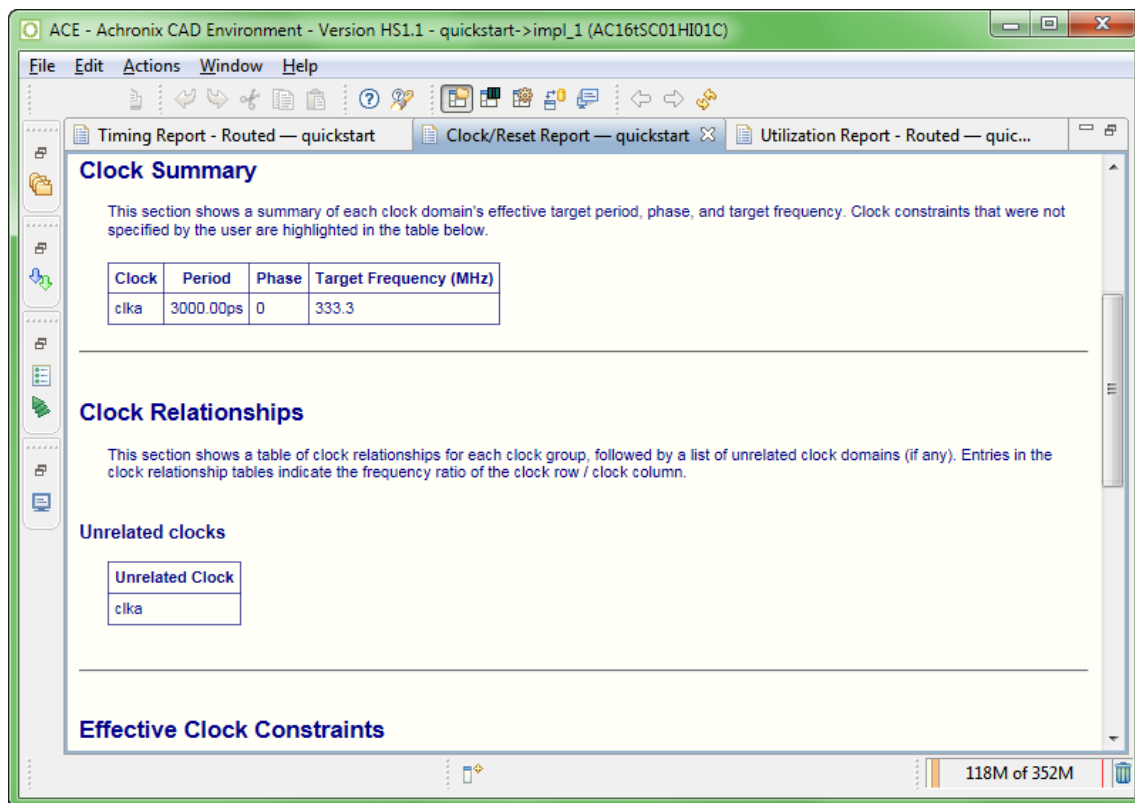



Figure 134 • Example of the Projects Perspective With The Editor Area Maximized

Working with Projects and Implementations

Creating Projects

To create a new project in the workspace:

1. Click the () **Create Project** toolbar button in the Projects view.
2. In the Create Project dialog, type in or browse to the location of the new project directory.

Caution!

Directories in the path that do not exist are created.

3. Type in the new project name and click **Finish**.

After clicking **Finish**, the new project appears in the Projects view. The new project contains a default implementation named `imp1_1`, which is set as the new active implementation. A project file is also created and saved in the new project directory.

Saving Projects

Some project operations cause changes to a project to be saved to the project file automatically, while others change project data without saving. Each Project with unsaved changes is marked in the GUI with an asterisk on the lower left corner of its project icon (📁). If any project in the workspace has unsaved changes, the Projects view title is also marked with an asterisk:



Figure 135 - Projects View Title Unsaved Changes Example

To save the changes to a project:

1. Select the project in the Projects view.
2. Either press **CTRL+S** on the keyboard, select the (📁) **File Save** toolbar button on the main toolbar, or select the **File** → **Save** menu option.

To save a project to a different file:

1. Select the project in the Projects view.
2. Select the **File** → **Save As...** menu option.
3. Browse to a new file location.
4. Enter a project name and click **Save**.

When exiting, ACE prompts to save changes to any projects with unsaved changes:

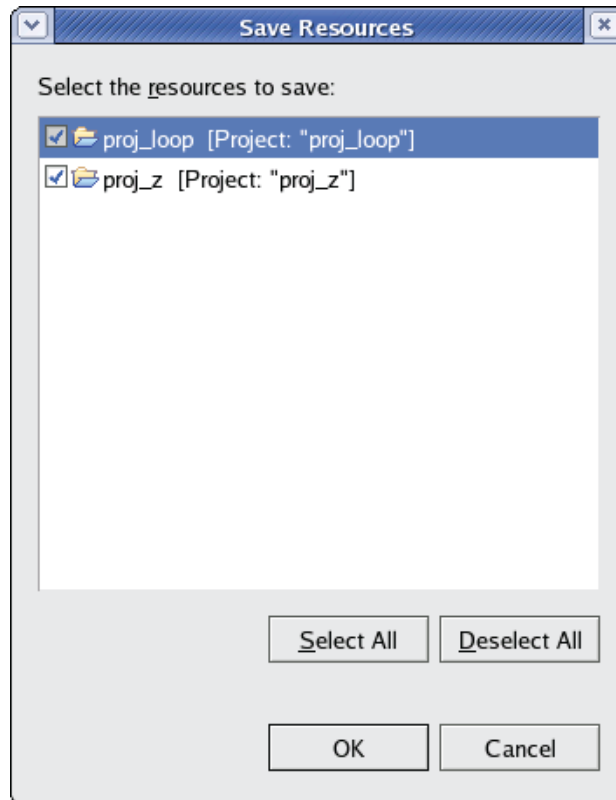


Figure 136 • Project Unsaved Changes Prompt

Loading Projects


By default, when the ACE GUI starts, it attempts to automatically re-load all projects which were open in the prior ACE GUI session.

Caution!

Be aware that any projects which are still locked by another ACE session are not automatically re-loaded, nor is any related error reported. Additionally, project files from the prior session which are no longer found in the file system are not loaded, nor are any related errors reported.

Loading a Project Using the GUI

To load existing [Projects \(page 222\)](#) into the workspace:

1. Click the () **Load Project** toolbar button in the [Projects View \(page 117\)](#).
2. In the [Load Project Dialog \(page 175\)](#), **Browse** to the location of the project directory. Or, if the project has been opened by ACE previously, find the previously opened `.acxprj` project file in the list of choices in the drop-down combo box within the dialog.

3. Select the project file and click **Open**.

After clicking **Open**, the `load_project` (page 667) Tcl command is issued and the project now appears in the Projects view. This project is restored from its previous state, and its last implementation is set as the new active implementation. Any place-and-route data for the active implementation is not loaded by default. See [Restoring Implementations](#) (page 302) for details on loading a prior place-and-route state.

 **Note**

Default Implementation Options Change Over Time

The default [Implementation Options](#) (page 0) for ACE change over time as new optimizations become available and existing optimizations are refined. When a project is loaded from an earlier version of ACE, a "Project Version Mismatch" popup dialog is shown offering to reset all Implementation Options of all implementations to the latest default values.

To avoid risking the loss of old optimizations saved in implementation options, say no to the offered reset. To see how the new default implementation options could affect the design, simply create a new implementation for the project. The new implementation contains all the new default values for implementation options, but contains no place-and-route data. Be aware that since no constraint files from the project are enabled by default in a new implementation, it is necessary to choose which constraint files to enable for the new implementation before running the flow.

Loading a Project Using Tcl

The Tcl commands `load_project` (page 667) and `restore_project` (page 688) may be used to open projects (and potentially also the most recent project implementation) in ACE:

- The `load_project` command is simple, and only opens the specified project for later use, without loading any additional place-and-route state of an implementation.
- The `restore_project` command is capable of much more and by default, attempts to load the most recent `.acxdb` file (potentially containing place-and-route data) for the most recent implementation in the specified project.

Project Locking and Lock Files

 **Warning!**

Project locks protect users from data corruption

ACE uses project locks and lock files to protect user data. Do not attempt to bypass ("-force") the project locks or lock files.

Achronix does not support running multiple ACE sessions on the same project (directory) simultaneously. Having a single project open in multiple ACE sessions is known to cause problems.

Every project opened by ACE is locked by that ACE session for as long as the project is open. Locking is primarily used to prevent file corruption, which could occur if multiple ACE sessions attempt to operate within the same project simultaneously. If another ACE session attempts to open a project while the project is still locked, ACE reports an error in the Tcl Console. The error message mentions the username and hostname of the session which created the project lock, allowing the coordination of sequential (not simultaneous!) project access.

Example error message for a locked project

```
cmd> load_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1"
Project: "~/output/quickstart/quickstart.acxprj" is locked by another ACE session and
cannot be loaded. This project is locked by user: TestUser1 on host: TestStation1. [...]
```

Instead of forcing project lock overrides or deleting lock files, when needing simultaneous access to a design, consult your Achronix FAE. Some potential options include [Using Incremental Compilation \(Partitions\) \(page 404\)](#), or using version control tools to store the project.


Caution!

While it is possible to keep multiple copies of the same project in separate project directories, this method is extremely difficult to coordinate, and is thus not recommended by Achronix.

In the unlikely occurrence of an ACE crash, a project may mistakenly remain locked after ACE has closed. Because the project is still locked, subsequent attempts to load the project fail with an error message similar to the one above. To recover from such situations, see ["Unable to Load Project: Project is Locked \(page 764\)"](#) under [Troubleshooting \(page 761\)](#).

Removing Projects

To remove a project from the workspace:

1. Select a project in the Projects view.
2. Click the () **Remove** toolbar button in the Projects view.


After clicking **Remove**, the project no longer appears in the Projects view. The project files are not deleted from the file system during this operation, and it is the responsibility of the user to clean up unwanted files on the disk. The project is left untouched on the disk so that it can be loaded again, later, if desired.

Opening Project Files in an Editor

To open a project file in the editor area, double-click the project in the Projects view. The project file now appears in a text editor in the editor area. Editing a project file in the workspace does not affect the project unless the project is removed and then re-loaded from the changed project file.

Adding Source Files

To add source netlist and constraint files to a project in the workspace:

1. Select the [Project \(page 222\)](#) in the [Projects View \(page 117\)](#) to which the source files are to be added.
2. Click the () **Add Source Files** toolbar button in the Projects view.
3. In the [Add Source Files Dialogs \(page 148\)](#), browse to the location of the source files.
4. Select the desired file in the dialog, and click **Open**.

⚠ Caution!

By default, ACE loads source files in the same order they were added to the project. If ACE is loading files in an incorrect order, drag and drop them into the desired order within the project Netlists and/or Constraints nodes in the [Projects View \(page 117\)](#).

After clicking **Open**, the source files appear in the appropriate virtual folder under the selected project in the Projects view. The source files are not actually consumed until the corresponding flow step is run (such as **Run Synthesis**, **Run RTL Simulation**, or **Run Prepare**). Adding a source file to a project simply creates a link to the file so that it may be loaded during flow execution.

See also:

- [add_project_source_files \(page 612\)](#)
- [Removing Source Files \(page 301\)](#)
- [enable_project_source_file \(page 635\)](#)
- [disable_project_source_file \(page 627\)](#)

Source File Load Order

ℹ Note

Source file load order is shared by all [Implementations \(page 229\)](#) within a given [Project \(page 222\)](#). Enabling of constraint files (choosing which constraint files are actually loaded) or netlist files is allowed to differ in each implementation and is managed by the checkboxes in the [Options View \(page 96\)](#).

To assist with the understanding of the load order of the source files, the RTL, netlist, and constraint files are listed in the [Projects View \(page 117\)](#) in the same order in which they are loaded (the constraint files are additionally listed in order within the [Options View \(page 96\)](#)).

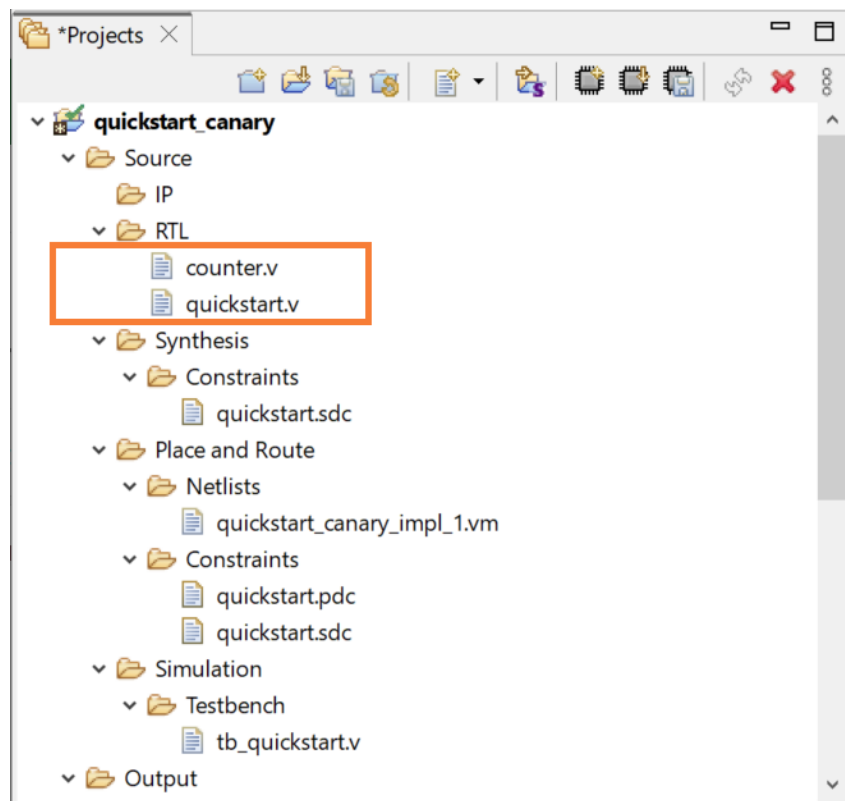


Figure 137 - Projects View File Order Example

By default, ACE loads source files in the same order they were added to the Project. Frequently, the order in which source files are loaded is important. For example, the creation of a clock may occur in the source file `create_clocks.sdc`, while operations upon that created clock may happen in the source file `generated_clocks.sdc`. To avoid errors, first add `create_clocks.sdc` to the project as a source file, then add `generated_clocks.sdc` as a source file. If attempting to add all the files to the project in a single operation, the results are platform dependent, but often the operating system "helpfully" sorts the bulk-added files alphabetically behind the scenes, which causes ACE to add them to the project in a potentially incorrect order (and thus later try loading them in that same incorrect order).

When the displayed source file load order is incorrect, there are a few ways to alter the load order:

- Changing the order of existing RTL files, netlist source files, and constraint source files can be performed quickly using mouse drag-and-drop operations in the [Projects View \(page 117\)](#) (or by using Tcl commands created explicitly for this purpose). The tree should be expanded to show all the netlist and/or constraint files, then drag-and-drop the files to re-order them within the appropriate Project View node until they achieve the desired order. The next time the [Flow is Run \(page 306\)](#), the constraint files are loaded in the chosen order.

See also:

- [get_project_source_files \(page 657\)](#)
- [move_project_source_file \(page 669\)](#)
- A more tedious way to alter the order (but possibly the easiest way to script) is by removing all the source files from the project.

see:

- [Removing Source Files \(page 301\)](#)
- [remove_project_source_files \(page 675\)](#)

Add them to the project again, one at a time, in the desired order.

See:

- [add_project_source_files \(page 612\)](#)

Enabling/Disabling Netlists and Constraint Files for Implementations

Implementations (page 229) are allowed to individually enable and disable the loading of netlists and constraint files within their parent **Project** (page 222). This selective loading is managed through the **Options View** (page 96), under the **Design Preparation** category of Implementation Options. Simply uncheck the checkbox next to the constraint files which should not be loaded for the implementation.

See also:

- [disable_project_source_file \(page 627\)](#)
- [enable_project_source_file \(page 635\)](#)

Constraint files which are disabled (unchecked) for the current **Active Project and Implementation** (page 229) are displayed in grey (instead of black) within the **Projects View** (page 117).

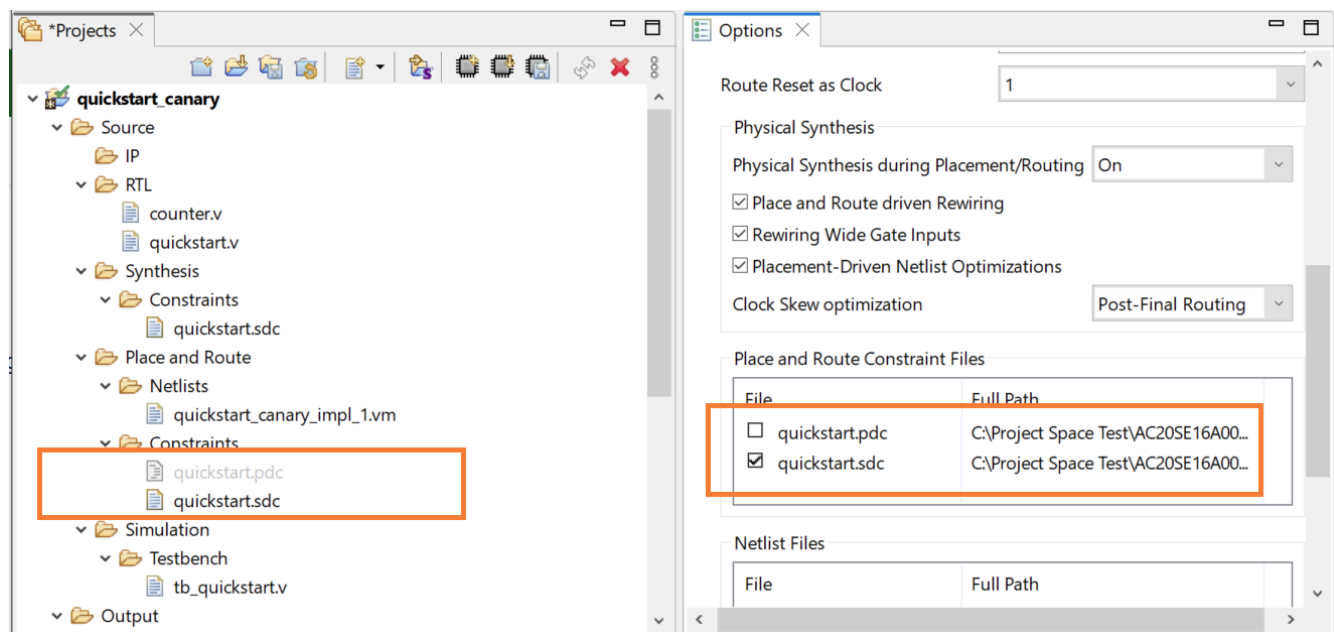




Figure 138 • Projects View Disabled Constraints File Example

Removing Source Files

To remove a source file from a [Project \(page 222\)](#) in the workspace:

1. Select a source file in the [Projects View \(page 117\)](#).
2. Click the () **Remove** toolbar button in the Projects view.

Or:

1. Right-click the source file in the Projects View.
2. Choose () **Remove** in the popup context menu.

After clicking **Remove**, the source file no longer appears in the Projects view. Source files are not deleted from the file system during this operation, and it is the responsibility of the user to clean up unwanted files on the disk. The source file is left on the disk so that it can be loaded again later, if desired.

See also:

- [remove_project_source_files \(page 675\)](#)
- [Adding Source Files \(page 297\)](#)

Disabling Constraint Files

It is often not necessary to completely remove constraint files from a project. Instead, constraint files can be individually disabled for any [Implementations \(page 229\)](#) within a project.

1. Select/activate an implementation within the Projects View. The [Options View \(page 96\)](#) is updated to show the implementation options for that implementation. At the bottom of the **Place and Route** implementation options category, there is a list displayed of the Constraint Files for the project.
2. In the Options View, expand the **Place and Route** implementation options category. At the bottom of the category, there is a list displayed of the Constraint Files in the project.
3. Deselect (clear) the checkbox(es) of constraint files which should not be loaded for the implementation.

See also:

- [disable_project_source_file \(page 627\)](#)
- [enable_project_source_file \(page 635\)](#)

Opening Source Files in an Editor


To open a source file in the editor area, double-click the source file in the Projects view. The source file now appears in a text editor in the editor area. Editing a source file in the workspace does not affect the results of the flow unless the flow is re-run on the affected project implementations.

Note

IP (.acxip) files that are not part of the currently active project cannot be opened.

Creating Implementations


To create a new [implementation \(page 229\)](#) in a [project \(page 222\)](#) in the workspace:

1. Select a project in the [Projects view \(page 117\)](#).
2. Click the  **Create Implementation** toolbar button in the Projects view.
3. In the [Create Implementation dialog \(page 166\)](#), type in the name of the new implementation and click **Finish**.

After clicking **Finish**, the new implementation appears under the selected project in the Projects view. The new implementation is set to be the [active implementation \(page 229\)](#) and contains default values for all [implementation options \(page 0\)](#). A new implementation directory structure is also created under the project directory if it does not already exist.

Saving Implementations

To save the state of the database (options, netlist, constraints, placement, and routing data) for an implementation in a project in the workspace:

1. Activate an implementation in the Projects View.
2. Run the flow (at least through Run Prepare).
3. Optionally edit placement or routing information.
4. Click the  **Save Implementation** toolbar button in the Projects view.
5. In the [Save Implementation Dialog \(page 181\)](#), type the file path to the .acxdb Archive File to which the implementation data is to be saved and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation is stored in the .acxdb Archive file, which can be restored again later.

See also:

- [Restoring Implementations \(page 302\)](#)
- [save_impl \(page 705\)](#)
- [restore_impl \(page 687\)](#)

Some Flow Steps Automatically Save the Implementation State

A subset of the [Flow Steps \(page 234\)](#) automatically save the current state in .acxdb files. These files are called:

- <implementation_name>_prepared.acxdb
- <implementation_name>_placed.acxdb
- <implementation_name>_routed.acxdb

These files are created at the end of the Run Prepare, Run Place, and Run Route flow steps, respectively.

Restoring Implementations

To restore the state of the database (options, netlist, constraints, placement, and routing data) for an implementation in a project in the workspace:

1. Activate an implementation in the Projects View.
2. Click the  **Restore Implementation** toolbar button in the Projects view.

3. In the **Restore Implementation Dialog** (page 180), enter the file path to the `.acxdb` Archive File from which to restore the implementation data and click **Finish**.

After clicking **Finish**, the state of the database (options, netlist, constraints, placement, and routing data) for the implementation is restored from the `.acxdb` Archive file.


 The Run Prepare, Run Place, and Run Route **Flow Steps** (page 234) automatically save checkpoint `.acxdb` files (by default) that may be restored later.

See also:

- [restore_impl](#) (page 687)
- [save_impl](#) (page 705)

Copying Implementations

To create a new **implementation** (page 229) that is a copy of an existing implementation,

1. In the **Projects View** (page 117), select (**activate** (page 229)) the implementation to be copied.
2. Select the () **Create Implementation** toolbar button in the Projects View.
3. In the pop-up **Create Implementation dialog** (page 166):
 - a. Enter the name of the new implementation.
 - b. Check the **Copy Option Values from Active Implementation** checkbox.
 - c. Click **Finish**.

After clicking **Finish**, the new implementation appears under the selected project in the Projects view. The new implementation is set to be the **active implementation** (page 229) and contains **implementation options** (page 0) values copied from the source implementation. A new implementation directory structure is also created under the project directory if it does not already exist.

Setting the Active Implementation


To change the **active implementation** (page 229) in the GUI, do one of the following:

- Click an implementation in the Projects view, which activates the selected implementation
- Click a project in the Projects view, which activates the first implementation of the selected project

Changing the active implementation causes the flow status to be cleared and changes the target for all flow operations to the new active implementation.

Removing Implementations

To remove an implementation from a project in the workspace:

1. Select an implementation in the Projects view.
2. Click the () **Delete** toolbar button in the Projects view.

After clicking **Delete**, the implementation no longer appears in the Projects view. Removing an implementation from a project causes all settings for the implementation to be deleted from the project file when the project is saved.

Configuring Project and Implementation Options

To configure project and implementation options in the workspace:

1. Select an implementation in the **Projects view** (page 117), changing the active implementation to the selection.
2. In the **Options view** (page 96), use the controls to configure the available implementation options for the active implementation.

After changing implementation options in the **Options view** (page 96), the **flow status** (page 242) is optionally cleared based on the options that changed. A change to an implementation option requires the **flow** (page 234) to be re-run for that implementation in order for the changes to affect the results of the flow. The changes to the implementation options are not saved until the affected project is saved.

Opening Output Files in an Editor

To open an output file in the editor area, double-click the output file in the Projects view. The output file appears in a text editor in the editor area.

Note

Editing an output file is *NOT* recommended.

Opening Report Files in an Editor

To open a report file in the editor area, double-click the report file in the Projects view. The report file now appears in a web browser in the editor area.

Note

Editing a report file is *NOT* recommended.

Cleaning Projects

In the **Projects** (page 222) Perspective, while running the **Flow** (page 234) for active **Implementations** (page 229), ACE generates output files and sub-directories under the active implementation directory. When the configuration is changed and the entire flow or a sub-flow is re-run, any previously generated output files with the same filenames are overwritten. However, some organizations prefer to clean the active implementation directory (and sub-directories) before every flow run to avoid having any lingering stale files.

ACE provides a simple and easy way to delete (clean) these sub-directories and output files through the **clean_project** (page 616)

Tcl command and/or related actions in the **Projects View** (page 117). Specifically, the implementation sub-directories which are cleaned include:

- .debug/
- output/
- reports/

Additionally, the *.acxdb files for the selected implementation(s) are deleted.

 **Warning**

Cleaning projects (or implementations) is an irreversible action – the files are deleted from the file system.

In contrast, see also [Removing Implementations \(page 303\)](#), which removes an implementation from a project without deleting any files from the file system.

 **Note**

Multiprocess Reports are a special case

The [Multiprocess Summary Report \(page 255\)](#) files are a special case. These files are not stored at the Implementation level, but at the Project level. Thus, these reports are not deleted when individual Implementations are deleted (**Clean Implementation**), but are deleted when an entire Project is cleaned (**Clean Project**).

 **Note**


Log files are never cleaned/deleted

To ensure the full history of an implementation is always maintained, cleaning a project or implementation never deletes *.log files (or any other files found within the <implementation_directory>/log/subdirectories).

To clean the implementations of a project from the workspace:

1. Select a project in the Projects view.
2. Right click the selected project and select () **Clean Project** from the menu.
3. This option selects all implementations of the selected project by default. Choose one or more implementations to clean from the **Clean project** dialog.
4. Click **OK**.

This operation can also be performed on an implementation:

1. Select an implementation from a project in the Projects view.
2. Right click the selected implementation and select () **Clean Implementation** from the menu.
3. This option selects only the current implementation by default while the rest of the implementations for the selected project are not selected. Choose one or more implementations to clean from the **Clean project** dialog.
4. Click **OK**.

See also:

- [Projects View \(page 117\)](#)
- [clean_project \(page 616\)](#)
- [remove_project \(page 674\)](#)
- [remove_impl \(page 673\)](#)
- [Removing Projects \(page 297\)](#)



- [Removing Implementations \(page 303\)](#)

Running the Flow

A flow can only be run on the current [Active Implementation \(page 229\)](#). If no active implementation is set in the [Projects View \(page 117\)](#), then the [Flow Steps \(page 234\)](#) in the [Flow View \(page 53\)](#) are disabled. Some flow steps are optional while others are required. Optional flow steps may be enabled or disabled in the Flow view by checking or clearing the checkbox to the left of each flow step label.

Running the Entire Flow

To run the current [Active Project and Implementation \(page 229\)](#) through the entire flow (sequentially run each of the [Flow Steps \(page 234\)](#) in order):

1. Enable the desired optional flow steps (and disable the unwanted optional flow steps) by clicking the checkboxes next to the flow steps. Required flow steps cannot be disabled.
2. Choose the () **Run Flow** or **Re-Run Flow** action in the [Flow View \(page 53\)](#), either as a toolbar button or context menu choice.
3. (Optionally) Stop the flow from continuing to the next flow step at any time by clicking the () **Stop Flow** toolbar button in the Flow view. When this occurs, ACE displays a dialog allowing the optional restoration of a prior flow state for the implementation by loading an .acxdb file.

See also:

- [restore_impl \(page 687\)](#)
- [Load Acxdb Dialog \(page 174\)](#)

Disabled flow steps are skipped (not executed) during this operation.

As each individual flow step is run, its [Flow Status \(page 242\)](#) changes from incomplete, to running, to either error or complete. If an error occurs during the execution of a flow step, the flow is stopped, and no further steps are attempted.

See also:

- [run \(page 689\)](#)
- [enable_flow_step \(page 634\)](#)
- [disable_flow_step \(page 626\)](#)

Note

Special note regarding the Flow and Incremental Compilation:

When Incremental Compilation is enabled, it might be necessary sometimes to recompile all partitions. This operation can be performed by managing the individual partitions using the [Partitions View \(page 107\)](#), but an easier way to trigger the recompile is to select the Flow View context menu choice **Re-Run Flow with "-ic init"**, which re-initializes the state of all partitions before starting the full flow.

See [Using Incremental Compilation \(Partitions\) \(page 404\)](#) for more details. See also:

```
run -ic init (page 689).
```

Note**Special note regarding Evaluation Flow Mode:**

When the **Flow Mode** implementation option is set to **Evaluation**, the flow steps under **Design Completion** and **FPGA Programming** are not executed. See [Flow Mode \(page 242\)](#) for more details.

Running a Sub-Flow

When using the [Flow View \(page 53\)](#), there are several ways to run a subset of the available [Flow Steps \(page 234\)](#) on the [Active Project and Implementation \(page 229\)](#).

It is possible to run individual flow steps one-at-a-time, to run all required flow steps up to a specified step (stopping when the specified step is completed), and to resume running a partial flow to flow completion.

As each flow step is run, its [Flow Status \(page 242\)](#) (as displayed in the Flow View) visibly changes from incomplete, to running, to either complete or error. If an error occurs during the execution of a flow step, the flow stops running any further steps. Disabled Flow steps are not executed during these operations.

Run an Individual Flow Step

Simply right-click the chosen flow step, and select the **Run Selected Flow Step** context menu item. Alternately, double-click the chosen flow step.

If any prerequisite required flow steps have not yet been executed, they are run in standard order prior to the chosen step. Any preceding optional steps are not run, even if they are enabled.

After any prerequisite required steps are complete, the chosen flow step is executed.

Warning!


Run Selected Flow Step runs the selected step even if that step is optional and not currently enabled (its checkbox is unchecked). Also, this action executes not only the selected step, but any preceding required steps.


Again, only the preceding *required* flow steps are run, not any preceding optional steps, even if they were selected (had their checkboxes checked).

See also:

- [run -step <id> \(page 689\)](#)

Run Remaining Enabled Flow Steps (Resume Flow)

When the flow has been stopped before completion, or when a partial flow state has been [loaded \(page 295\)](#) from a saved `.acxdb` file, ACE can continue the flow if the () **Resume Flow** action is chosen. This action causes ACE to start running at the first enabled flow step which follows the latest successfully completed flow step.

1. Ensure the desired optional steps are enabled (checked) in the Flow View.
2. Choose the () **Resume Flow** action from the view toolbar, or from the right-click context menu.


Note


If the current flow mode is set to **Evaluation**, the flow stops after the **Place and Route** category completes. See [Flow Mode \(page 242\)](#) for details.


See also:

- [run -resume \(page 689\)](#)
- [enable_flow_step \(page 634\)](#)
- [disable_flow_step \(page 626\)](#)

Stopping the Flow

At any time while a flow step is running, it is possible to ask ACE to stop running the flow with the Flow View () **Stop Flow** action.


Some flow steps might respond by stopping immediately, while others need to perform some additional work before exiting the flow step. In both cases, the [Flow Status \(page 242\)](#) of that step typically is changed to the () Error status to indicate that the flow step did not complete successfully.

It is frequently the case that when the flow is interrupted in this manner, the Tcl Console shows many logged error messages for the interrupted flow step. Typically the () **Resume Flow** or **Run Selected Flow Step** can be selected and ACE resumes normal work from the last successfully completed flow step.

Running Multiple Flows in Parallel

Normally, ACE only allows a single [project \(page 222\)](#) [implementation \(page 229\)](#) to be run through the [flow \(page 234\)](#) at a time. Using the [Multiprocess View \(page 73\)](#), ACE allows running multiple implementations *within a single project* through the flow in parallel, via a configurable number of parallel processes. Executing multiple implementations in this manner allows ACE to provide a [Multiprocess Summary Report \(page 255\)](#) of the resulting frequencies, permitting QOR performance comparisons between implementations utilizing different starting clock constraints, placement constraints, and potential optimizations.

Finding the Multiprocess View

The Multiprocess view is, by default, present in the () [Projects perspective \(page 6\)](#), in a tab group shared with the [Options view \(page 96\)](#). If it is not visible for some reason, the Multiprocess view may be displayed within any perspective by selecting **Window** → **Show View** → **Other...** → **Achronix** → **Multiprocess**.

Configuring the Execution Queues

Within the Multiprocess view, the [Execution Queue Management \(page 75\)](#)" section allows the configuring the desired number of parallel processes used to consume the queue of selected implementations. Simply set the value of **Parallel Job Count** to the desired number of parallel processes. Using the minimum value of **1** causes all queued implementations to be executed sequentially, one after another.

ACE may be configured to execute the parallel processes in the background on the host workstation running the ACE GUI, or ACE may submit each implementation as an independent executable job to an external cloud/grid/batch job

submission system. Detailed configuration of the external job submission command is handled on the [Multiprocess: Configure Custom Job Submission Tool Preference Page](#) (page 206).

License Management Considerations with Multiprocess

Warning

Each parallel ACE process needs access to an ACE software license.

Floating licenses:

When running using the Multiprocess View in the ACE GUI, to run N parallel execution queues, $N+1$ ACE licenses are needed (the extra license is for the GUI itself, as it is managing all the queues running in the background). Talk to your Achronix FAE to ensure that your site has enough licenses to enable running with Multiprocess functionality.

Node-locked licenses:

When running using the Multiprocess View in the ACE GUI, provide a node-locked license for every host machine running ACE. When running local/background execution, a single node-locked license is sufficient for all ACE sessions running on that host. When running using an external job submission system, every execution host needs its own node-locked license installed on that execution host. Talk to your Achronix FAE to ensure that your site has enough licenses to enable running with Multiprocess functionality.

The following is a common best practice when determining the needed ACE floating license counts to support multiprocess runs at a specific site, as well as choosing the best value for **Parallel Job Count** based upon the available floating license count.

- Start with the number of ACE users (U).
- Determine the maximum number (P) of parallel job execution hosts available to the job submission system. Alternately, if job execution hosts are each allowed to run more than one job at a time, determine the maximum number (P) of ACE multiprocess jobs the system could theoretically handle in parallel, which is usually determined by ACE memory requirements.

Note

Remain aware that ACE memory requirements vary widely based upon design size/complexity, target device, and other factors. Remember that ACE logs its peak memory consumption at the completion of every flow step — this peak memory value is a useful guideline when determining expected multiprocess memory consumption.

Sites trying to minimize license usage, or where users must share the available execution hosts equally, the minimum number of required ACE licenses (L_{min}) would then be $L_{min} = U + P$. Each user is then allowed to consume up to L_{user} licenses during their multiprocess sessions, where $L_{user} = 1 + (P \div U)$.

Sites that want to maximize job throughput, where individual users may be allowed to completely saturate the execution hosts, the maximum number of required ACE licenses (L_{max}) would then be $L_{max} = U + (P \times U)$. Each user is then allowed to consume up to L_{user} licenses during their multiprocess sessions, where $L_{user} = 1 + (P \times U)$.

Each user must then set their **Parallel Job Count** to their personal value of $L_{user} - 1$ (one license is reserved for the ACE session coordinating Multiprocess), which should then ensure that no multiprocess jobs run out of licenses.

Important Considerations When Using Background Execution on the Local Host Workstation

Be aware that if the configured number of parallel processes is too high, total execution time actually takes longer than it would at lower values. The constraints are available memory and available processor cores, as well as the load from other processes running on the host workstation.

When choosing how many parallel background implementations to allow, it is very important that users ensure they do not exhaust the physical memory (RAM) available on the executing workstation, otherwise flow execution times quickly increase (due to the OS swapping memory pages to disk). Do take into account any other users on the same workstation, as well as the memory currently in use by the already-running ACE GUI and associated back-end `acx` process.

Each additional background ACE process takes multiple Gigabytes (GB) of memory — the exact amount varies depending upon the size of the design and the size of the target Achronix device (smaller designs and smaller devices, of course, take less memory). An estimate for large designs on a very large FPGA device is around 16GB of memory used for each background process. Again, this is only an estimate — designs nearing 100% device utilization may require more memory.

Be aware that with modern multi-core hyper-threading workstations, memory limits are usually the reason to constrain the parallel process count. It is not unusual to find workstations capable of running 8 simultaneous threads while only having 32GB of RAM. While on this example workstation, if the ACE user is running the flow on a very large FPGA design (where our estimate was around 16GB per background process), the most efficient parallel process count would likely be **1** or **2**; it would depend upon the Operating System, how much memory ACE and other currently-running processes are already using, and whether the user planned to continue using the workstation interactively while the background processes were executing. Since multiple iterations through the flow are likely, it may be worthwhile to track the total multiprocess duration at multiple parallel process counts, so as the user continues working, they can use the most efficient settings for that workstation.

In the majority of cases, the parallel process count should *at most* be the *lesser* of the following two values (remaining aware that lower values may be even faster):

- **processor constraint: $1 + T$**

where:

T = the total number of simultaneous threads supported by the workstation,

$T = (P \times (C \times H))$, where

P = the total number of processors in the workstation

C = the number of physical cores per processor

H = 2 if the cores are hyper-threaded, 1 if not

- **memory constraint: A / D**

where:

D = amount of memory needed by the design, as reported in ACE log files (or the Tcl Console) during a prior flow execution

A = the total available (unused) RAM memory,

$A = R - (O + G + B + U)$, where

R = total RAM installed in the workstation

O = amount of memory required by the Operating System

G = amount of memory required by the currently-running ACE GUI

B = amount of memory required by the currently-running ACE backend process (named `acx` or `acx.exe` in process lists)

U = amount of memory required by all other user processes expected to execute while the background processes are running

Continuing the example of the 8 thread 32GB workstation: If the workstation is running Linux, estimate the OS requires 0.5GB, the ACE GUI process requires 1GB, the GUI backend process (`acx`) requires 3GB, and no other user processes are running; the available memory $A = (32\text{GB} - (0.5\text{GB} + 1\text{GB} + 3\text{GB} + 0\text{GB})) = 27.5\text{GB}$. If the log files of a prior run report the user design requiring a peak memory usage of 7GB, then the memory constraint value is $(27.5\text{GB} / 7\text{GB}) \approx 3.9$. The processor constraint would be $(8 \text{ threads} + 1) = 9$. The lesser of the two values is the 3.9 for the memory constraint. So following the guidelines, the ideal parallel process count would be between **3** and **4**. To completely balance the two constraints for the design, the example user would need $7\text{GB} \times 9 \text{ threads} = 63\text{GB}$ of available memory before they could expect optimal performance running 9 parallel processes.

**Tip****ACE Memory Utilization**

ACE logs the amount of memory (RAM) used by the backend as a design proceeds through the flow. This number is reported at the end of every [flow step \(page 234\)](#) in the log files and (when the GUI is running the flow in single process mode) in the Tcl Console. It is also possible to directly query ACE at any time to find out the peak backend memory usage in KB with the [get_ace_peak_memory_usage \(page 642\)](#)

Tcl command. These features should allow an educated decision to be made as to how much memory each parallel background process requires for the design, and thus how many processes may be executed in parallel within the current memory constraints.

Example from log

```
Flow step "report_timing_final" completed in 1 seconds. Peak memory usage is 4917 MB.
```

Example from Tcl Console View query showing peak memory use in KB

```
cmd> get_ace_peak_memory_usage
5035008
```

Configuring ACE to Use an External Job Submission System

Due to the wide variety of grid, batch, queue and cloud job submission systems available, it is not possible for ACE to support each individual product specifically. Instead ACE Multiprocess can be configured to interface with whatever job submission system is available at the user site.

Minimum Requirements

Currently, the following are required for the minimum functionality:

- The name of the job submission executable or script (providing a full directory path to the executable or script is recommended, though it may not be necessary in some PATH configurations).
- The job must be submitted in synchronous/blocking mode (the job submission process must not complete until the ACE child process/job has completed execution). ACE itself currently has no support for the tracking of job status through periodic queries as would be necessary with asynchronous/non-blocking jobs.

Warning!

A non-blocking/asynchronous job submission system currently risks data corruption, because ACE can no longer guarantee it knows when the job is complete, so ACE cannot properly manage data locking states across the simultaneously executing implementations.

- An exit code of zero from the job submission process indicates success.
- A non-zero exit code from the job submission process indicates failure. The Multiprocess system simply reports success/failure based upon the exit code value.

Presently, if the job system at the user site is not already a synchronous/blocking system, then it is necessary for the user to write their own script or executable which approximates synchronous/blocking functionality.

In theory, this should be possible:

1. Submit the job.
2. Capture the unique identifier for that job.
3. Loop while querying the job status (using the previously captured unique job identifier from the job system) until completion is indicated.
4. Capture the job exit code.
5. Return the appropriate exit code (to ACE) indicating the success or failure of the job status.

After the job submission request completes, and after any network files have been written, the ACE Multiprocess GUI reads the output files from the submitted job, gathering the information needed for the Multiprocess Summary Report. The read of the result files only happens once per job.

If the user job submission process finishes before the submitted ACE job is complete (as would happen with a non-blocking job submission system), the ACE implementation output files are either missing or incomplete when queried, and the Multiprocess Summary Report shows that no results were found for that ACE job.

Optional Improvements

When external job submission systems are properly configured, the following features are also available within ACE Multiprocess:

- Support for killing or cancelling submitted jobs
- Assignment of the job working directory
- Assignment of a job name
- Streaming real-time log output for each Job

Killing or Cancelling Already-submitted Jobs

For simplicity, the ACE Multiprocess system only manages the job through the (blocking) job submission process. The Multiprocess system currently does not track job identifiers or any special job status logged by the job submission process itself. When ACE needs to cancel or kill the job, it essentially sends a "kill" (technically a "SIGINT" in Linux) to the (blocking) job submission process. It is expected that this also kills/cancels the underlying ACE job. If this does not actually kill the underlying ACE job (or remove it from the appropriate job queue, etc.), then it becomes the responsibility of the ACE user to manually kill the job with the job submission tool.

Job Working Directory

In some cases it might be necessary to specify the working directory of the ACE job as a command line argument to the job submission process. While ACE jobs lacking an explicit working directory assignment are known to run without errors in most situations, some job submission systems may require the explicit assignment of a working directory. The working directory specified by ACE for a job changes for each implementation and, typically, is the implementation directory itself.

Job Name

It is extremely convenient for ACE Multiprocess to have a way to pass in the job name as a command line argument to the job submission process.

The job name does not aid ACE directly, but is intended to assist external users of the job submission system in tracking job status, job lifetime, queue management, etc. through other (non-ACE) tools.

The job name is currently made unique by concatenating the following information, with items in brackets replaced by their logical values:

- ACE_Multiprocess_[user name]_[project name]_[implementation name]

Additionally, special characters found in the variable values are replaced by the underscore (`_`) character.

Streaming Real-Time Job Log Output

ACE logs all of its normal output in a log file, which gets post-processed after job completion to verify how far ACE went through the flow, and to harvest the reported timing information for inclusion in the Multiprocess Summary Report. However, properly configuring the following can help the user track the progress of the ACE jobs as they run.

If the job submission process redirects or pipes the standard output and standard error streams from the underlying ACE job, so that the job submission process re-transmits that same data on its own standard output and standard error streams, then ACE may be able to show the streamed job output during the Multiprocess run.

If the underlying ACE job standard output and standard error streams are redirected to a file, preferably through user-managed command line options for the job submission process itself, then ACE may be able to show the streamed job output from the file during the Multiprocess run.

Note

Due to various concerns such as network file write caching and the occasional complexity of shell redirection in spawned processes, this job submission log file option may be difficult to get working properly.

Configuring ACE

The external job submissions are performed via a user-configurable command-line executable. The configuration is managed through the [Multiprocess: Configure Custom Job Submission Tool Preference Page \(page 206\)](#), reached by following the **(configured in Preferences)** hyperlink in the Multiprocess View. As a potentially useful example, by default ACE is configured to use GridEngine (see http://en.wikipedia.org/wiki/Oracle_Grid_Engine) through the `qsub` command. (When using a system other than the GridEngine, users need to clear all fields on that preference page and provide the values which are appropriate for their own system.) For the configuration to work, the job submission command must be in the path (or have its path fully specified), and the ACE executable must be reachable from the job system execution hosts.

ACE is able to optionally provide some values to the job submission system if the related argument fields are populated. The optional values ACE may provide are:

- The working directory for the ACE Multiprocess job

- The job name
- The path and filename to be used by the job submission log file

It is extremely likely that additional command-line arguments are required by the job submission executable in order to meet the ACE minimum requirements. Additional arguments are also typically needed to assign execution queues, memory limits, etc. These additional arguments (and any argument values) should be specified on the preference page as well.

⚠ Caution!

Command-line arguments must not be specified in the **Job Submission Executable** field. Attempts to do so fail.

Debugging Job Submission System Configurations:

If the job submission system is properly configured on the host machine, (meaning it is possible to successfully execute non-ACE tasks using the job submission executable from the command line), and ACE is still unable to successfully submit jobs to the system, please contact Achronix technical support.

⚠ Warning!

Potential for File Corruption

Attempting to manually run the logged command on the command line (without the Multiprocess View additional automated safety locks in place) might cause ACE data file corruption.

While ACE does provide the complete attempted job submission command in the "Multiprocess Run Logs" section of the Multiprocess view, *DO NOT* copy the text of the attempted command and manually attempt execution from the command line. A large number of assumptions are made (including bypassing the normal project-level and implementation-level safety checks which prohibit file corruption) when ACE is executed using the provided command options and Tcl batch script — these assumptions are violated during manual execution attempts.

Network File System Latency Concerns

When dealing with external job submission systems, network drive latency becomes a concern. The ACE multiprocess system waits for each external process to complete before it harvests the timing information for that implementation. To avoid potential hangs (where the multiprocess system mistakenly waits forever for a file to appear, or for a file to be completely written), there is a configurable timeout setting, which is by default 5 seconds. If, after the external process for an implementation has completed, the timing summary information cannot be found within the allowed number of seconds, then the **Multiprocess Summary Report** (page 255) shows the message "No Timing Results Found" for that implementation.

i Note

A "No Timing Results Found" message for an implementation in the summary report means the timing information needed for the summary was not available within the allotted time. The allotted time may be increased using the **Allowed seconds of NFS write latency** setting in the **Multiprocess: Configure Custom Job Submission Tool Preference Page** (page 206), as shown in the image below.

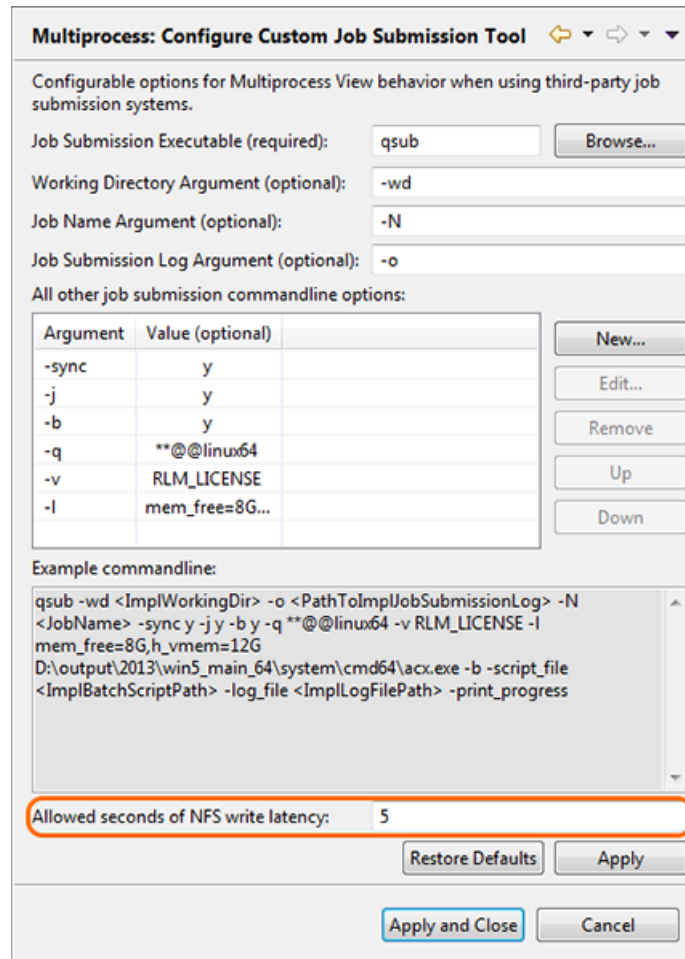


Figure 139 · Configure Custom Job Submission Tool Dialog

Configuring the Desired Flow to be Followed by the Selected Implementations

All the implementations run through the Multiprocess View follow the same **flow steps** (page 234) through the **flow** (page 234), as configured in the **Flow View** (page 53). Thus, ensure that all optional flow steps are enabled/disabled as desired before starting multiprocess execution.

Additionally, in the section of the Multiprocess View labeled "**Multiprocess Flow Management** (page 76)", it may be chosen to stop the multiprocess flows early, prior to traditional "completion". For example, when designs are known to be incomplete, and thus known to fail the **Run Final DRC Checks** flow step, users often choose to stop the flow prior to running that failing flow step.

To stop all the multiprocess flows at a given flow step, simply select that flow step in the **Stop Flow After:** drop-down list. No subsequent flow steps are executed for any of the selected multiprocess implementations.

As a convenience, since optional flow steps are frequently chosen to be the final multiprocess flow step, if the selected final flow step is optional and not enabled, then as the multiprocess implementations are scheduled, the selected flow step is enabled for all the multiprocess implementations before they begin execution. For example, if

Stop Flow After is set to **Run Post-Route Timing Analysis** (an optional step), but this flow step is disabled in the Flow View, then the multiprocess flows will all enable that step and stop only after the **Run Post-Route Timing Analysis** flow step has completed.

✔ **Multiple PVT corners**

If it is desirable for the summary report to contain results for multiple PVT corners at once for any implementation, then the **Flow Mode** (page 242) must be set appropriately for the implementation, and the flow must include/enable the **Run Sign-off Timing Analysis** flow step. See **Timing Across All Temperature Corners** (page 266) for more details.

Selecting the Implementations to be Run in Parallel

To select the implementations to be run in parallel:



1. In the **Projects View** (page 117), select the desired **project** (page 222). The Implementation Table within the Multiprocess view **Select Implementations** (page 76) section is updated to display data for the **active project and implementation** (page 229).
2. In the Multiprocess View, ensure the **Existing Implementations** radio button within the "Select Implementations" section is selected. This limits the contents of the Implementation Table to just the implementations which already exist for the active project (generating and executing new implementations using **option sets** (page 0) is covered in **Attempting Likely Optimizations Using Option Sets** (page 392)).
3. In the Implementation Table, all listed implementations are selected (the checkbox in the Implementation column will be checked) by default. Implementations may be selected/deselected in bulk with the **Select All** and **Deselect All** buttons. Individual implementations may have their selection toggled by clicking their checkboxes in the first column of the Implementation Table.

✔ **Tip**

If the implementation table is not large enough (or is too large) for the full implementation list, simply collapse and/or expand one of the other sections in this view (left-click the section title). This causes the table to resize to exactly fit the current implementation list.

Starting Background Execution

To start background execution:

1. When the parallel count has been set, the flow has been configured, and the desired implementations have been selected, click the  **Start Selected** button, or the equivalent () **Start Background Queue Execution** action in the Multiprocess view local button-bar or menu, to begin background multiprocess execution.
2. After multiprocess execution has been started, the **Parallel Queue Count** and Implementation Table is disabled. They are not re-enabled until multiprocess execution is completed. In the **Multiprocess Run Logs** (page 79) section, a new tab is created for the logged output of each selected implementation. The log info in each tab is updated live as the corresponding implementation process executes (the displayed log info mirrors the information captured in the **log files** (page 231) for each implementation).
3. As implementations are queued, start execution, and complete execution, the implementations **execution states** (page 78) are updated in the implementation table, and each implementation log tab icon is also updated to show the current execution state.



Note




Presently, it is not possible to control the order of implementation execution.

Caution!

For safety, all ACE Tcl commands (i.e., most ACE GUI interactions) are blocked while multiprocess execution is underway. Blocked Tcl commands are queued and allowed to run when multiprocess execution is completed. Similarly, multiprocess execution is blocked until all in-process and already-queued ACE Tcl commands (including running the Flow in the foreground) are completed.

Stopping/Canceling Background Execution

All queued and executing background implementations may be quickly cancelled by selecting the  **Stop All** button below the Implementation Table, or the equivalent () **Stop All Background Queue Execution** action in the Multiprocess view local button bar or menu.

It is also possible to cancel execution of individual implementations. This may only be done via the () Progress View. During multiprocess execution, a () button to show this view is visible in the lower-right of the ACE status bar. This view is also available by selecting **Window** → **Show View** → **Other...** → **General** → **Progress**. The Progress View displays all queued and currently-executing background tasks, including the tasks for the background implementation processes. To the right of each listed incomplete background task is a () stop icon, which cancels/stops execution of that task. Because the Progress View can list more tasks than just the background multiprocess implementations, caution should be used to avoid cancelling/stopping the wrong task.

Viewing the Results

After the first implementation completes execution, an HTML **Multiprocess Summary Report** (page 255) file is created and opened in ACE (the report file is created in the project directory, and is named `multiprocess_summary.html`. This action automatically overwrites previous multiprocess summary reports without prompting). As each subsequent implementation completes execution, the multiprocess summary report is updated with the latest data.

As implementations complete execution, their **execution states** (page 78) change appropriately. If an implementation encounters errors while running the flow, the execution state for that implementation becomes the Error state, which is reflected by the icon shown both in the log tab and the Implementation Table. In addition, the tooltip for the appropriate log tab and Implementation Table entry is updated to include a summary of the captured error messages. Error details are visible in the log messages shown in the tab, as well as within the **Implementation Log** (page 232) and **Multiprocess Log** (page 232) for that implementation.

⚠ Caution!

There is a known sequence whereby all multiprocess results are identical. If there is an existing project with previously generated option sets, and ACE has been upgraded to a newer version, it prompts when opening the existing project to reset the [implementation \(page 229\)](#) options to the defaults for the new version of ACE. The recommendation is to accept this reset as a new version of ACE may include new implementation options which are only applied by accepting this reset. At the same time, older implementation options that have been deprecated are removed.

The issue is that currently ACE resets all of the [option sets \(page 0\)](#) to the same default values. Subsequently, when a multiprocess flow is run with the new project, all results are identical. The workaround is, after having upgraded ACE to the new version and accepting the [implementation \(page 229\)](#) option reset, delete all the implementations other than the original base implementation and then regenerate the [option sets \(page 0\)](#).

Multiprocess Batch Mode

Overview

To obtain the highest QoR, ACE supports running multiple different implementations in parallel using Multiprocess. Multiprocess is available from the ACE GUI and is described in:

- [Multiprocess View \(page 73\)](#)
- [Running Multiple Flows in Parallel \(page 308\)](#)
- [Attempting Likely Optimizations Using Option Sets \(page 392\)](#)

Multiprocess batch mode provides the same functionality as the GUI, but can be run from the ACE Tcl console command line or by using an external Tcl script. The relevant Tcl command is [run_multiprocess \(page 692\)](#).

Modes

Similar to running Multiprocess from the GUI, Multiprocess batch mode must be run in the context of a currently [Active Project and Implementation \(page 229\)](#). The current active project is used as the basis for all the implementations that are run, with the current active implementation used as the basis for any newly-generated implementations.

Multiprocess batch mode supports three modes of operation:

- Generate implementations from option sets (default setting).
- Seed sweep (`-seed_sweep`).
- Use existing implementations (`-use_existing_impls`).

A full list of all options is given in the [run_multiprocess \(page 692\)](#) manual page.

Generate Implementations From Option Sets

Running from option sets is the default mode of operation for Multiprocess batch mode and is used when neither `-use_seeds` nor `-use_existing_impls` is specified. This mode generates fresh implementations for

every available option set definition. Previously existing implementations with the same name are overwritten. See [Attempting Likely Optimizations Using Option Sets \(page 392\)](#) for additional details.

The currently active implementation is always included as one of the executed flows when this mode is used.

Seed Sweep

The seed sweep mode generates fresh implementations (based upon the active implementation) for every specified seed value. Previously existing implementations with the same name are overwritten.

Seed sweep mode is selected by use of the `-use_seeds` argument:

```
run_multiprocess -use_seeds {5 7 13}
```

The current active project and implementation forms the basis of each generated implementation, with the implementation option "seed" set to the given seed value as an override of the seed inherited from the active implementation.

Implementations created during seed sweep are named `{active_impl_name}_seed{value}` (e.g., `impl_1_seed18` for a seed value of 18 and an active implementation name of `impl_1`).

The currently active implementation is always included as one of the executed flows when this mode is used.

Use Existing Implementations

The Use Existing Implementations mode does not generate any new implementations, but simply runs each of the named implementations. Use Existing Implementations mode is selected by use of the `-use_existing_impls` argument:

```
run_multiprocess -use_existing_impls {impl_1 impl_1_improved impl_1_experimental}
```

To run all existing implementations, specify:

```
run_multiprocess -use_existing_impls [get_impl_names]
```

Unlike the other modes, the currently active implementation is *NOT* included as one of the flows run unless it is explicitly named in the `-use_existing_impls` list.

Flow Steps

An important principle to understand is that the enabled or disabled [Flow steps \(page 234\)](#) of the currently active implementation are inherited by all implementations executed during Multiprocess batch mode. Therefore, before commencing the Multiprocess batch mode, ensure that the currently active implementation has the desired flow steps enabled (and has the unwanted flow steps disabled).

In addition, Multiprocess batch mode can be configured to stop at an explicit flow step via the `-stop_flow_at` argument. This argument can be used to terminate each flow at a particular step. For example, if `report_timing_routed` is specified as the stop step, then none of the DRC or bitstream flow steps are performed (because those flow steps occur later than the chosen stop step; see the [Flow Steps \(page 234\)](#) page for the ordered complete listing of all default flow steps). Using the `-stop_flow_at` argument reduces the overall time taken for Multiprocess batch mode, as each implementation runs a reduced number of flow steps. After

Multiprocess batch mode has completed and an implementation has been found which achieves the desired QoR, then that implementation can be loaded into ACE, and the final flow steps executed. With the aforementioned example, which was stopped at `report_timing_routed`, the routed `.acxdb` can be loaded into ACE, and the DRC and bitstream generation flow steps subsequently executed to produce the required bitstream.

Note

If the flow step specified by `-stop_flow_at` is disabled in the active implementation when the multiprocess run begins, (as long as the current **Flow mode** (page 242) allows it) the step is explicitly enabled for the active implementation and all executed implementations.

Getting Started

The commands to start Multiprocess batch mode vary according to whether ACE is running in command-line mode, batch mode (using a script file), or from within the ACE GUI.

Command-line Mode (Interactive)

1. Open ACE in command line mode (`ace -b`). See **Running ACE** (page 282).
2. Use **restore_project** (page 688) to load the project.
3. **Set the active implementation** (page 709).
4. (Optional) Use **disable_flow_step** (page 626) and **enable_flow_step** (page 634) to configure any flow steps desired/needed or bypassed for all of the implementations that are to be run.
5. Issue **run_multiprocess** (page 692) command. See **examples** (page 321) below.

Batch Mode (Script File)

Open ACE in command-line mode, passing in a script file. Use script arguments to specify the project name (`ace -b -script_file <my_mp_batch_script.tcl>`). See **Running ACE** (page 282).

Code

```
$ ace -batch -script_file <my_mp_script> -script_args <my_project_name>
```

An example script file, using the project names as the first argument is shown below.

Code

```
# Script file to run Multiprocess batch mode
set my_proj [lindex $argv 0]

# 1. Restore the project
restore_project $my_proj

# 2. Set active implementation (to the default)
```

```

set_active_impl impl_1

# 3. (Optional) Ensure Run Estimated Timing Analysis flow step is enabled. Disable
Generate Bitstream
enable_flow_step report_timing_routed
disable_flow_step write_bitstream

# 4. Run Multiprocess batch mode generating new implementations from option sets
#   Set to a maximum of 8 jobs
#   Stop after Post-Route Timing Analysis.
run_multiprocess -parallel_job_count 8 -stop_flow_at report_timing_routed

```

Note

When running in the command-line mode, or batch mode, in order to cancel a Multiprocess batch mode run, CTRL+C must be used.

ACE GUI

1. Open ACE GUI.
2. [Load the project \(page 295\)](#).
3. [Set the active implementation \(page 303\)](#).
4. (Optional) Using the [Flow View \(page 53\)](#), select or deselect any flow steps desired/needed or bypassed for all of the implementations that are to be run.
5. In the Tcl console window, issue the [run_multiprocess \(page 692\)](#) command. See [examples \(page 321\)](#) below.

Examples

Running all Option Sets

To run all option sets, using the existing maximum job count as specified in your ACE GUI preferences:

```
run_multiprocess
```

To run all option sets, limiting concurrent jobs to 8:

```
run_multiprocess -parallel_job_count 8
```

Running a Seed Sweep

To run a seed sweep using preferred seed values:

```
run_multiprocess -use_seeds {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31}
```

Re-running Four Existing Implementations

To re-run four existing implementations all at the same time:

```
run_multiprocess -use_existing_impls {impl_1 impl_1_acx_mux_utl_seed impl_1_acx_seed21  
impl_1_acx_seed33} -parallel_job_count 4
```

Re-running an Existing Implementation

To re-run an existing implementation, stopping just after running "write netlist final":

```
run_multiprocess -use_existing_impls {impl_1_seed88} -stop_flow_at write_netlist_final
```

Running all Option Sets

To run all option sets on the grid, with custom job submission parameters:

```
run_multiprocess -use_job_submission 1 -jobs_wd my_jobs_working_dir -jobs_name  
my_job_name -jobs_log my_jobs_logfile -jobs_args {{-sync y} {-j y} {-b y}}
```

Progress Monitoring

Within the Tcl console or shell, `run_multiprocess` checks each of the input arguments to ensure they are correct (including checking that any specified existing implementations exist) and then launches the requested number of parallel implementations runs. Within the Tcl console or shell, `run_multiprocess` indicates the start and completion (success or failure) of each implementation run.

To monitor progress, use the [Multiprocess Summary Report \(page 255\)](#) (`multiprocess_summary_report.html`) file that indicates which implementations have completed and their timing summary. This report is generated regardless of whether the Multiprocess batch mode was run from a command shell, or from within the ACE Tcl console. This report can either be viewed external to ACE using a web browser, or within ACE as detailed below.

Viewing Multiprocess Summary Report within ACE

Open the Multiprocess Summary Report from the Projects view, by right-clicking the project.

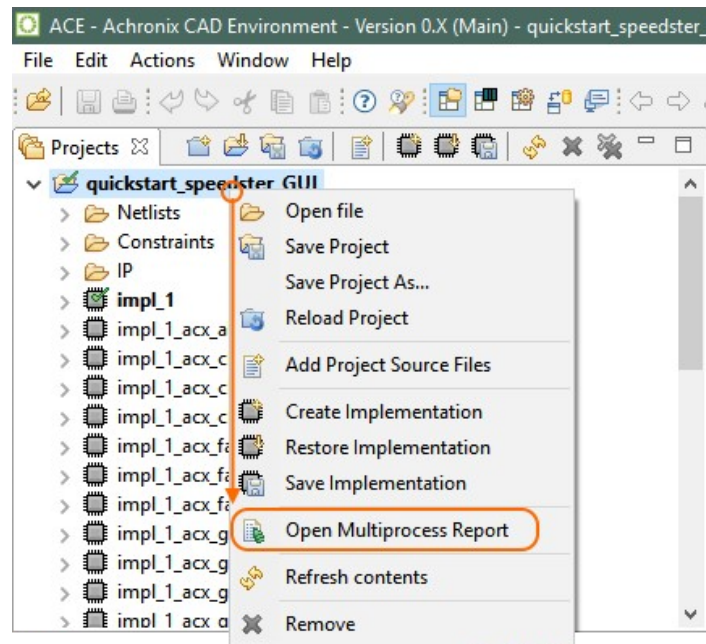


Figure 140 - Open Multiprocess Report

The report can be refreshed by right-clicking the view and selecting **Refresh**, or by pressing the refresh hotkey, **F5**, when the report tab has focus (click the report first). The report view does not automatically update when using Multiprocess batch mode. This behavior differs from when Multiprocess is run directly from the GUI Multiprocess view, where the report view is automatically updated after each implementation completes execution.

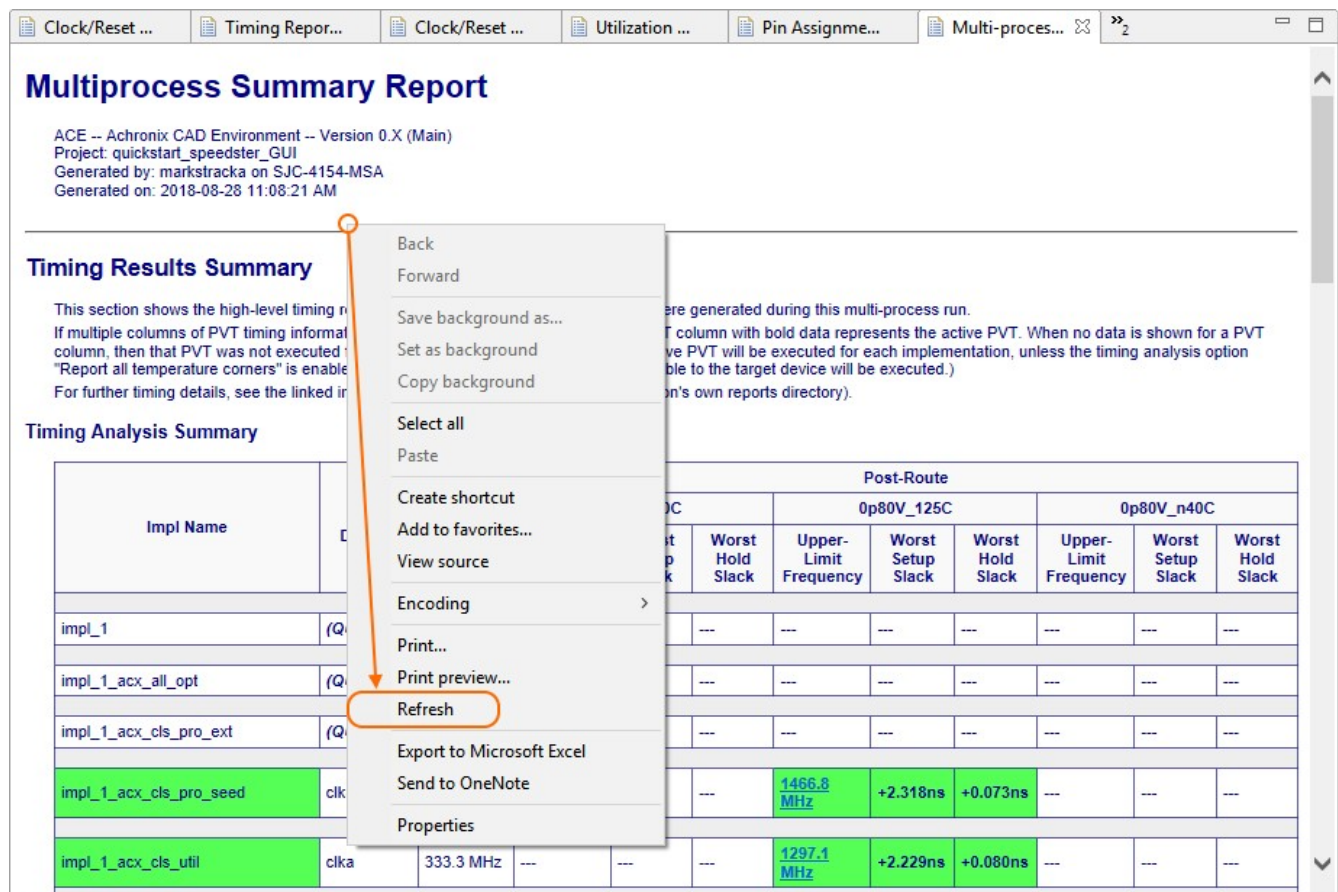


Figure 141 • Refresh Multiprocess Report View

Furthermore, progress monitoring can be achieved if a job submission system is used — completed jobs might be able to be monitored using the job submission system tool suite. Finally, to see the status of an individual implementation, open the <implementation>/log/multiprocessImpl.log file, and monitor updates to the individual implementation progress.

Stopping the Running Implementations

When running in command-line or batch mode, use CTRL+C to cancel a Multiprocess batch mode run. When calling run_multiprocess in the ACE GUI **Tcl Console view** (page 142), (as with all other Tcl commands,) the command can be cancelled using the Progress view (**Window** → **Show View...** → **Other** → **Progress**).

Detecting Changes to Project Source Files

ACE provides a rich set of features to enable detecting changes to project source files against the state of the project files loaded into the ACE database during the Run Prepare flow step, as described in the following sections.

Files Open in the ACE Editor Area

If a project source file is open in a text editor window, a pop-up dialog box appears offering to refresh the contents of the stale file in the ACE editor tab if the source file is changed on disk.

Smart Change Detection Using Custom Checksums

Instead of caching timestamps for files to perform the checking, ACE caches custom file checksum values. The checksums are computed in a robust way that ignores comment lines and whitespace lines, so that only the actual Verilog netlist or SDC/PDC constraints commands are used in the checksum. This allows generated files, such as the Synplify netlist, to be regenerated from the same RTL which produces the same gate level netlist, but with different comments at the top, to be treated as an unchanged file. Timestamps are inherently fragile and change when a project is copied from one directory to another. This checksum approach is much more robust and does not flag a source file as changed unless its content is meaningfully changed.

Saving the Active Implementation

When ACE saves an ACXDB file (when the state of the ACE database is saved for the active implementation), it caches the checksums of all project source files used to create that state in the DB. The project source file checksums are saved inside the `.acxdb` file.

Restoring the Active Implementation

When ACE loads/restores an ACXDB file (when saved place and route data is loaded from the `.acxdb` file on disk into the ACE database for the active implementation), ACE checks all project source files and checksums on disk against the cached project source files and checksums inside the ACXDB file. If a project source file has been added to the current ACE project, removed from the current ACE project, or if its file checksum has changed, ACE prints a warning message to the Tcl Console and ACE log file. This alerts users that the saved ACXDB is out of sync with the current project source files.

Caching the Project Source File State

The **Run Prepare** flow step is the first flow step, and is where all project source files are loaded into the ACE database from disk. Whenever the **Run Prepare** flow step is run, ACE caches the project source files and checksums for all files used in the active ACE project implementation.

Automatic Checking while Running the Flow

Each flow step (**Run Place**, **Run Route**, Timing Analysis, Final DRC checks, etc.) checks the project source files (including the `.acxprj` ACE project file itself) and checksums on disk against the files and checksums cached at the beginning of **Run Prepare**. If any file is missing, added, or out of sync, ACE reports a warning at the end of each flow step to the Tcl Console and ACE log file:

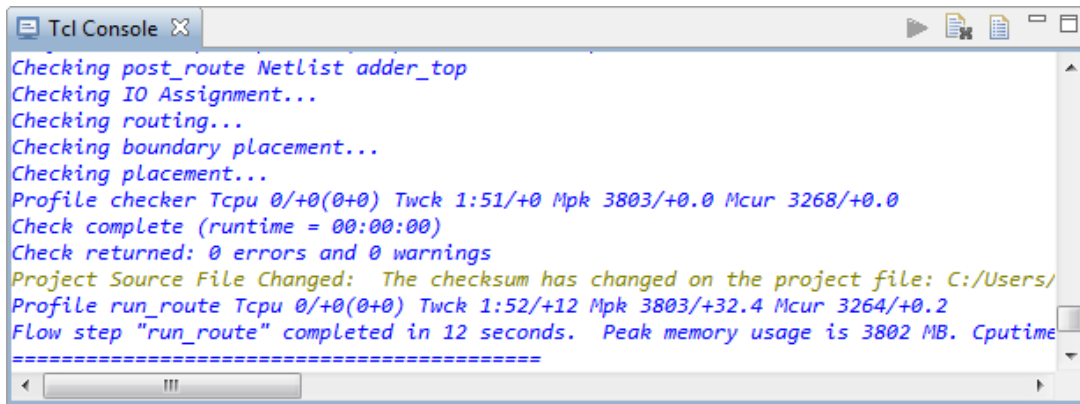


Figure 142 · Missing, Added, or Out of Sync File Warning Example

Warning Visualization in the Flow View

If any flow step reports a warning about out-of-sync files, all completed flow steps in the **Flow View** (page 53) are marked with a yellow warning icon instead of the green checkmark icon to indicate that the step is complete, but is out of sync with the source files on disk. The tooltip text in the Flow View shows all the warning messages.

Even when no flow step Tcl commands are running, the GUI checks the project source files and checksums every 5 seconds (by default) in a background thread. If any file becomes out-of-sync, all completed flow steps in the Flow View are marked with a yellow warning icon instead of the green checkmark icon to indicate that the step is complete, but is out of sync with the source files on disk.

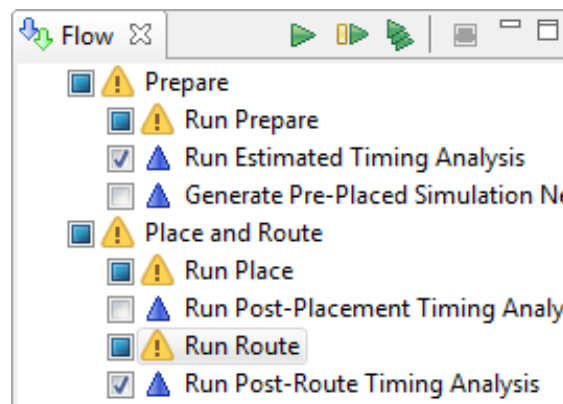


Figure 143 · Out of Sync File Warning Example

Pop-up Dialog Warnings

If the flow is running, a “Project Source Files Changed” dialog appears if a change to project source files is detected during the built-in check at the end of each flow step. The same warning messages that are printed to the Tcl console are displayed in the dialog. Optionally choose to cancel running the rest of the flow, or let the flow continue.

The flow continues to run in the background until the choice is made. The pop-up dialog has a preference checkbox which allows disabling the pop-up from being shown in the future.

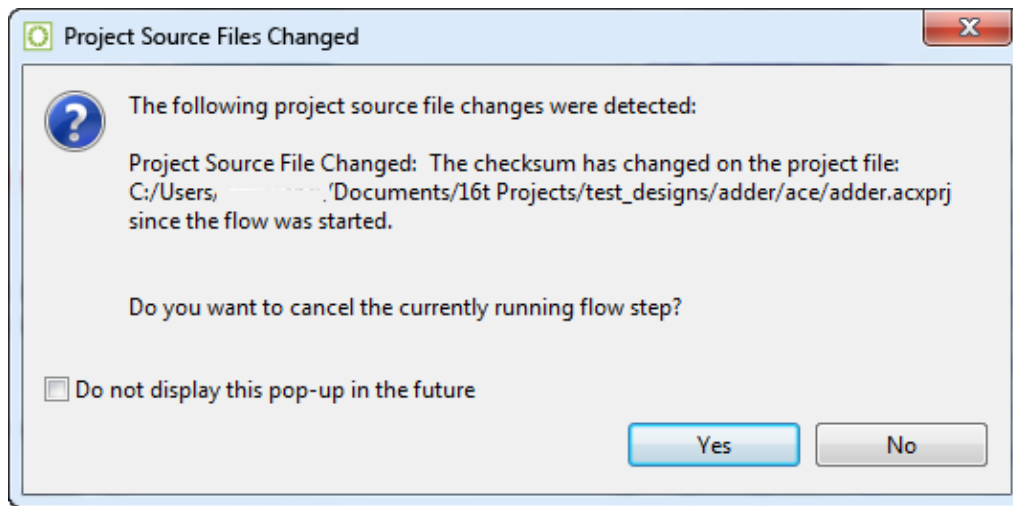


Figure 144 • Project Source Files Changed Dialog Example

If the flow is not running, a different “Project Source Files Changed” dialog appears if a change to project source files is detected during the built-in check at the end of each flow step. The same warning messages that are printed to the Tcl console are displayed in the dialog. There is no choice for cancelling the running flow, since the flow is not running. The pop-up dialog has a preference checkbox which allows disabling the pop-up from being shown in the future.

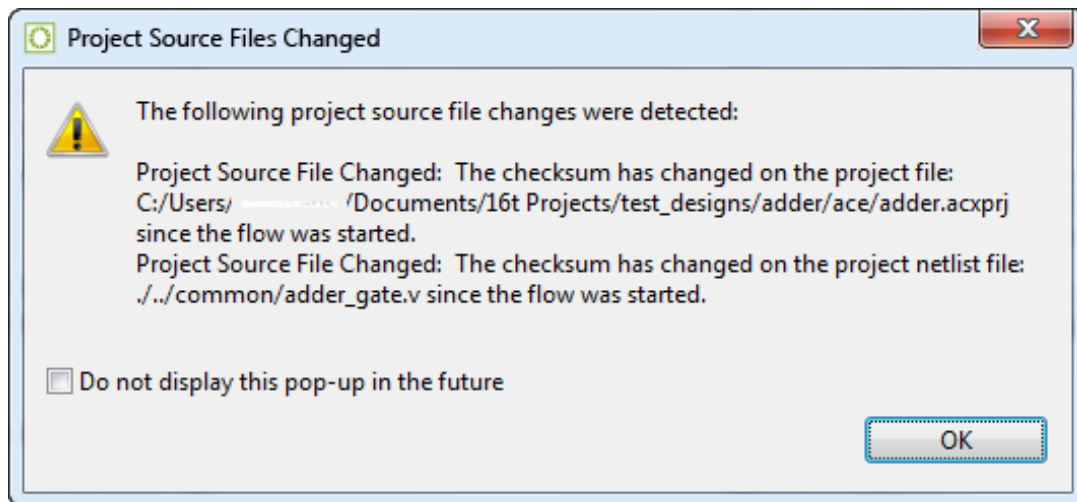


Figure 145 • Project Source Files Changed Dialog Example When Flow Is Stopped

Note**Minimal Pop-up Interruptions**

In general, pop-ups are minimized to only alert new changes. The Project Source Files Changed dialog appears only when a new change is detected. So if the gate level netlist file is changed while the flow is running, the pop-up (if enabled by the user preference) appears. If the same gate level netlist file is then changed several more times, no further pop-up appears since there has already been a notification that the file is different than the original source file. However, if a project constraints file (in addition to the gate level netlist) is then changed, the Project Source Files Changed dialog appears again to alert that now 2 source files are changed.

Managing Pop-up Preferences

There is a user preference to enable/disable the Project Source File Changed pop-up at the bottom of the **Project Management Preference Page** (page 214). This preference setting is the same as that controlled with a checkbox in the Project Source File Changed Dialog. From the main menu bar, select **Window** → **Preferences** and select **Project Management** on the left-hand side of the Preferences dialog. This preference page can be used to re-enable the pop-up if the checkbox in the dialog is enabled.

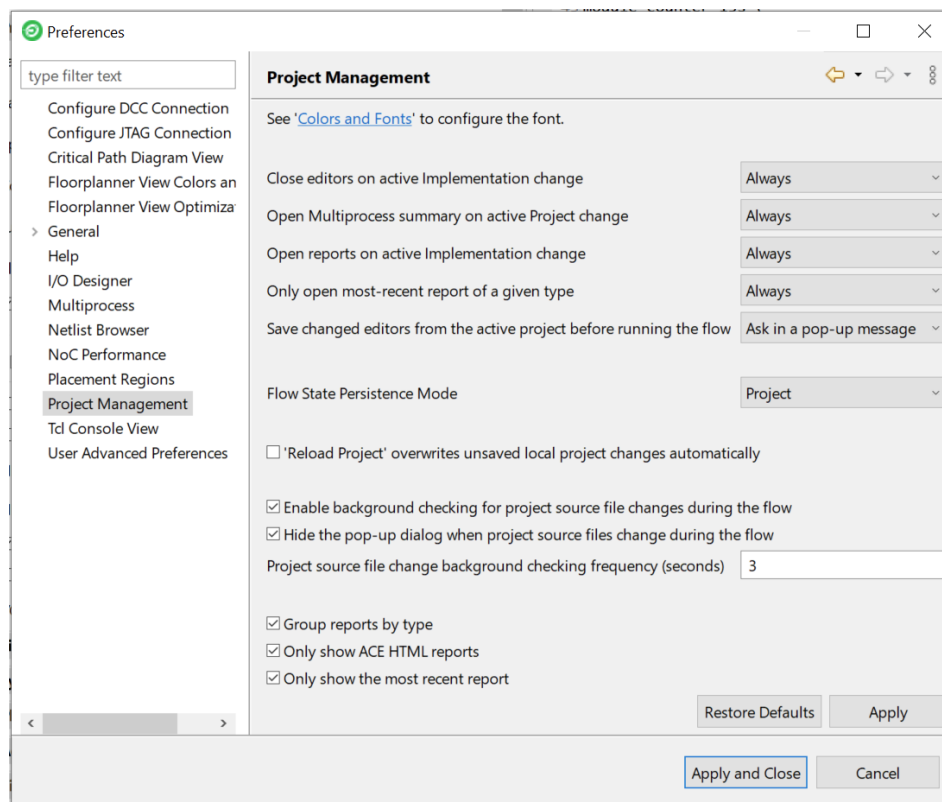


Figure 146 • Project Management Preference Page Example

Table 139 - Project Management Preferences

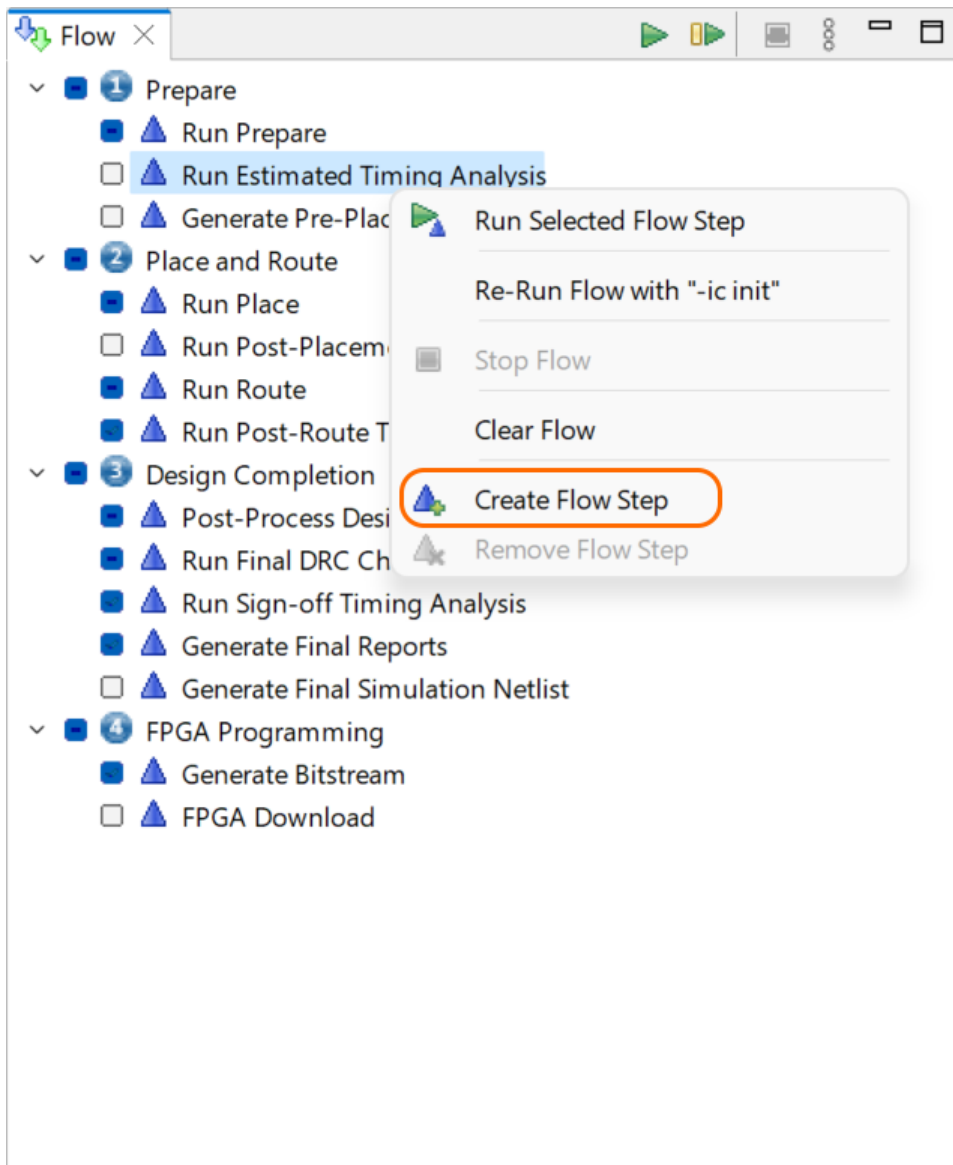
Option	Description
See also:	Link to Colors and Fonts preferences allowing choice of the font used in the Projects view.
Close editors on active implementation change	Can be used to automatically close open editors when the active implementation changes.
Open Multiprocess summary on active project change	Can be used to automatically open the Multiprocess summary report when the active project changes.
Open reports on active implementation change	Can be used to automatically open implementation-specific reports when the active implementation changes.
Only open most-recent report of a given type	Can be used to automatically open only the most-recent report of a given type.
Save changed editors from the active project before running the flow	Can be used to automatically save all open editors before running the flow.
Flow State Persistence Mode	Choose whether or not the enabled/disabled flow step state is persisted across ACE sessions per Session, per Project, or is Disabled (not persisted)
Reload Project overwrites unsaved local project changes automatically	Can be used to automatically overwrite unsaved local project changes when Reload Project is used (without a confirmation prompt).
Enable background checking for project source file changes during the flow	If enabled, project source files are periodically polled in the background to look for any changes made outside of ACE.
Hide the pop-up dialog when project source files change during the flow	If enabled, source file change notification is suppressed until the flow has finished running.
Project source file change background checking frequency (seconds)	Determines how often to poll for background source file changes.
Group reports by type	If enabled, reports are grouped into subfolders in the Projects view tree.
Only show HTML reports	If enabled, only HTML reports are shown in the Projects view tree.
Only show the most recent report	If enabled, only the most recent report is shown in any subfolder in the Projects view tree.

Tcl Command Support

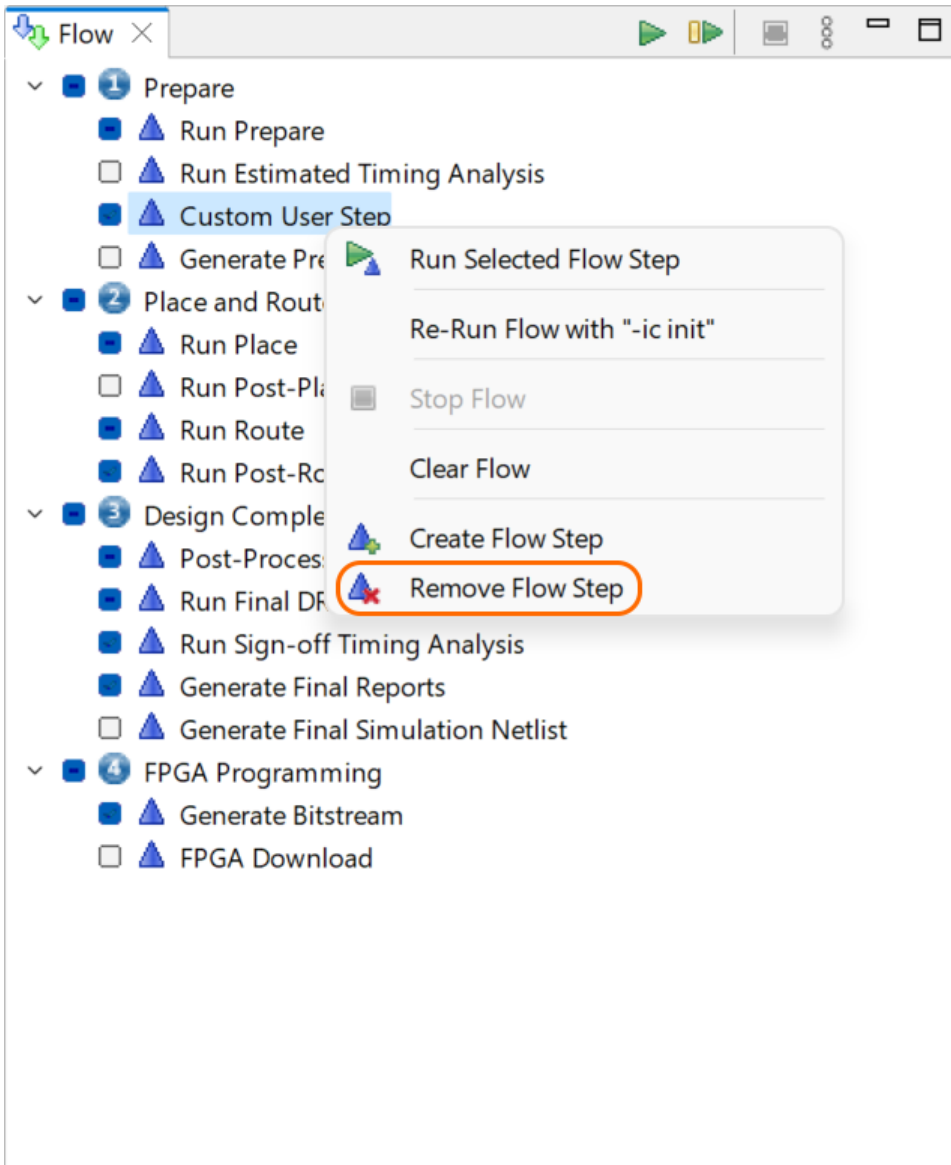
The `check_project_status` Tcl command can be called to manually check file and checksum consistency outside of the built-in checks performed at the end of each flow step. If any file is missing, added, or out of sync, ACE reports a warning to the Tcl Console and ACE log file. This command only applies if the flow has run at least through the **Run Prepare** flow step and there is a design loaded in the ACE DB.

Custom Flow Steps

Custom flow steps can be created by right-clicking the **Flow view** (page 53) tree and selecting **Create Flow Step**. The **Create a new Flow Step dialog** (page 159) appears:



Remove a custom flow step by right-clicking the Flow view tree and selecting **Remove Flow Step**.



⚠ Persistent Custom Flow Steps

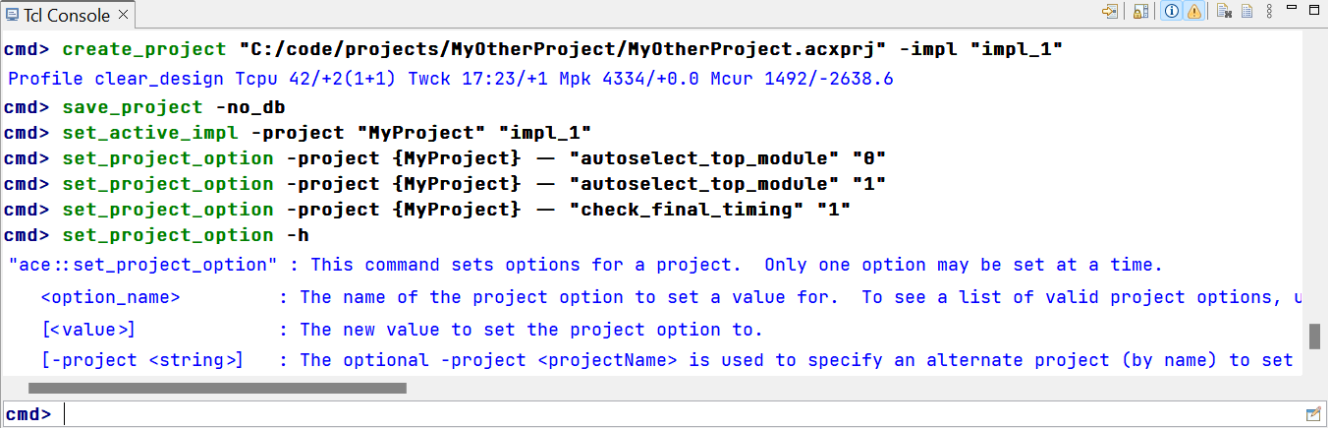
Custom flow steps can be defined in an [ACE_INIT_SCRIPT](#) (page 282) to make them persistent across ACE sessions, so they are defined every time you launch ACE.

Using the Tcl Console

Any operation that changes project or design data can be performed from the command line via a Tcl command (see the [Tcl Command Reference \(page 580\)](#)). The [Tcl Console view \(page 142\)](#) provides an interface from within the GUI for viewing and executing Tcl commands.

Sending Commands from GUI Actions

Any action in the GUI that changes project or design data automatically sends a Tcl command through the [Tcl Console view \(page 142\)](#) to do the work. All Tcl commands generated by GUI actions are displayed in the Tcl console along with any output from the command.



```

Tcl Console x
cmd> create_project "C:/code/projects/MyOtherProject/MyOtherProject.acxprj" -impl "impl_1"
Profile clear_design Tcpu 42/+2(1+1) Twck 17:23/+1 Mpk 4334/+0.0 Mcur 1492/-2638.6
cmd> save_project -no_db
cmd> set_active_impl -project "MyProject" "impl_1"
cmd> set_project_option -project {MyProject} -- "autoselect_top_module" "0"
cmd> set_project_option -project {MyProject} -- "autoselect_top_module" "1"
cmd> set_project_option -project {MyProject} -- "check_final_timing" "1"
cmd> set_project_option -h
"ace::set_project_option" : This command sets options for a project. Only one option may be set at a time.
<option_name>           : The name of the project option to set a value for. To see a list of valid project options, u
[<value>]                : The new value to set the project option to.
[-project <string>]      : The optional -project <projectName> is used to specify an alternate project (by name) to set
cmd>

```

Figure 147 • Tcl Console Example

Sending Commands from the Console

To send a command from the Tcl console, enter or paste the command text in the entry area in the Tcl console view and press ENTER. Valid commands are highlighted in bold green.

All output from the command is displayed in the Tcl Console view under the command prompt. Informational messages are displayed in **blue text**. Warning messages are displayed in **yellow text**. Error messages are displayed in **red text**.

Command Highlighting

Text entered in the Tcl console is checked against the valid set of user Tcl commands. Valid commands are highlighted in bold green.

Command Auto-Completion

When typing into the Tcl console, pressing the **TAB** key pops up a Tcl command auto-completion dialog. If no auto-complete suggestions are found, an error beep sounds.

Pressing the **TAB** key at an empty **cmd>** prompt pops up the full list of available commands. When the command auto-completion dialog is open, use the arrow keys to navigate up and down the list of choices and press the **Enter** key on a selected command to complete it at the command prompt. Typing while the command auto-completion dialog is open shortens or lengthens the list of valid commands, depending on the cursor position in the Tcl console view.

The screenshot shows a Tcl Console window with the following text:

```

cmd> create_project "C:/code/projects/MyOtherProject/MyOtherProject.acxprj" -impl "impl_1"
Profile clear_design Tcpu 42/+2(1+1) Twck 17:23/+1 Mpk 4334/+0.0 Mcur 1492/-2638.6
cmd> save_project -no_db
cmd> set_active_impl -project "MyProject" "impl_1"
cmd> set_project_option -project {MyProject} -- "autoselect_top_module" "0"
cmd> set_project_option -project {MyProject} -- "autoselect_top_module" "1"
cmd> set_project_option -project {MyProject} -- "check_final_timing" "1"
cmd> set_project_option -h
"ace::set_project_option" : This command sets options for a project. Only one option may be set at a time.
  <option_name>          : The name of the project option to set a value for. To see a list of valid project options, u
  [<value>]              : The new value to set the project option to.
  [-project <string>]    : The optional -project <projectName> is used to specify an alternate project (by name) to set

cmd> set
set
set_active_impl
set_clock_groups
set_clock_latency
set_clock_type
set_clock_uncertainty
set_cluster
set_csr_clock
set_data_check
set_disable_timing

```

When the **set** command is entered, an auto-completion dialog box appears, listing the following commands:

- set
- set_active_impl
- set_clock_groups
- set_clock_latency
- set_clock_type
- set_clock_uncertainty
- set_cluster
- set_csr_clock
- set_data_check
- set_disable_timing

To the right of the list, help text for the selected command (**set_active_impl**) is displayed:

```

"ace::set_active_impl" : This command sets the active implementation for
the current ACE session. The active implementation controls which
implementation the flow and project management commands are
operating on.
  <implName>          : The required <implName> argument is used to
specify the name of the implementation to set as the active
implementation.
  [-project <string>] : The optional -project <projectName> option is
used to specify an alternate project (by name) for the active
implementation to be set in.

```

Figure 148 • Tcl Command Auto Completion Dialog Example

Command Help

When the command auto-completion dialog is open, help text appears to the right of the command list for the selected command.

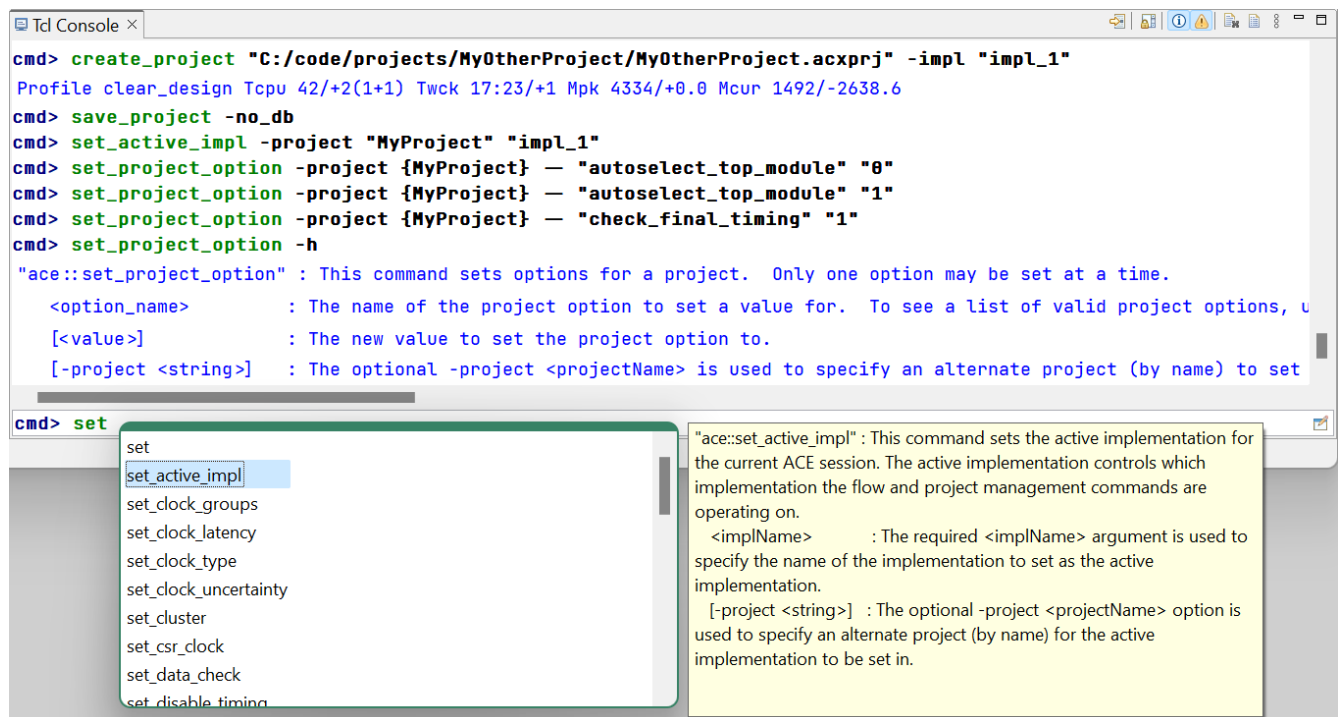


Figure 149 • Tcl Command Auto Completion Dialog Help Text Example

To view help text for commands, either bring up the command auto-completion dialog and select the desired command, or enter the command name at the **cmd>** prompt and use the **-help** argument to output the help text to the Tcl console.



Figure 150 • Tcl Command Help Option Example

Text Limit


The Tcl console view has a default limit of 2000 lines. When this limit is reached, any new lines entered via commands or message text causes the text at the top of the Tcl Console to be pruned.

This line limit can be adjusted on the Tcl Console Preference page. Increase it as much as you'd like, but keep in mind that the greater the value, the more memory the Tcl Console view will potentially use.

Note

Hitting the text limit does not affect the contents of the ACE log file. All messages continue to be logged in the log file and earlier messages are not removed.


Clearing the Console

Text in the Tcl console view can be cleared by clicking the () **Clear Console** toolbar button in the Tcl console view. This action clears out all of the text currently in the readout area.

Note

Clearing the console does not clear the contents of the ACE log file.

Viewing the ACE Log File

All Tcl commands and messages issued during an ACE session are recorded in the ACE log file. If the **Text Limit** ([page 335](#)) is reached from excessive messages, it is sometimes useful to browse the log file for previous messages. To open the ACE log file in the editor area, simply click the () **Display Log File** toolbar button in the Tcl Console view.

Object Type Prefixes

There are a variety of different object types supported by ACE. Most of these object types have a special single-letter prefix designating the type. These type prefixes are useful to avoid name collisions (i.e., between a net and a pin with the same name).

Many Tcl commands (i.e., [select \(page 708\)](#)) require that these prefixes be used when commands are issued. Other commands (i.e., [find \(page 638\)](#)), by default, include these prefixes on the return values.

Table 140 • Object Type Prefixes Used in ACE Tcl Commands (Sorted Alphabetically by Prefix)

Prefix	Object Type
c:	Critical Path
d:	Device Port
f:	Fabric Pin
i:	Instance
k:	Clock Domain
n:	Net
p:	Port
s:	Site
t:	Pin

Table 141 • Object Type Prefixes Used in ACE Tcl Commands (Sorted Alphabetically by Object Type)

Object Type	Prefix
Clock Domain	k:
Critical Path	c:
Device Port	d:
Fabric Pin	f:
Instance	i:
Net	n:
Pin	t:
Port	p:
Site	s:

Creating an IP Configuration


Achronix FPGAs feature a wide variety of embedded IP. These highly flexible IP blocks require configuration for proper operation.

ACE includes a number of IP [editors \(page 9\)](#) and [views \(page 16\)](#) which work together to provide a guide through the process of correctly configuring IP. The data for these IP configuration editing sessions is stored in `.acxip` files, which may be saved and loaded for future reuse or modification.

Using the data stored in the `.acxip` files, ACE generates RTL wrappers (Verilog and VHDL) containing the specified configuration parameters around the appropriate Achronix macro cells, as well as appropriate `.sdc` and `.pdc` files to complete the IP timing and pre-placement configuration. These generated files may then be incorporated into the user design for synthesis and simulation.


Note

Use of a generated VHDL wrapper also requires the generated Verilog wrapper (the VHDL simply wraps the Verilog instantiation).

Creating and editing IP configurations is typically performed from the () [IP configuration perspective \(page 6\)](#). In addition to the IP Configuration editors, this perspective incorporates supporting views allowing:

- Creating new IP configurations ([IP libraries view \(page 71\)](#))
- Viewing a graphical diagram of the IP configuration currently being edited ([IP diagram view \(page 69\)](#)); the diagram may show the macro interface, the dataflow, and/or the placement of the IP instance within the chip
- Navigating instantly to any page of the active IP configuration editor, while displaying the names and validity of each page ([outline view \(page 106\)](#))
- Viewing a detailed list of all the errors and warnings pertaining to all IP configuration files currently opened ([IP problems view \(page 71\)](#))
- Navigating directly to the source of the problem in the relevant IP configuration editor ([IP problems view \(page 71\)](#))

Creating and Naming an IP Configuration

Switch to the IP configuration perspective either by clicking the  IP configuration perspective icon or selecting **Open Perspective** → **IP Configuration** from the main menu. Select **File** → **New** → **IP Configuration...** from the main menu, or use the IP libraries view (see [IP libraries view \(page 71\)](#)) to open the [New IP Configuration Dialog \(page 176\)](#). After setting the location and name for the `.acxip` configuration file, click **Finish** to complete the process and activate the appropriate IP editor.

Caution!

The file name chosen for the `.acxip` configuration file is used as the module name for the generated Verilog module. A name must be chosen for the `.acxip` configuration file that is both a valid file name and also a valid Verilog module name which does not conflict with any other module names defined in the Achronix libraries.




For example:

- If the `.acxip` configuration file is named `foo.acxip`, the generated Verilog module is named `module foo`.
- If the `.acxip` configuration file is named `LRAM.acxip`, the generated Verilog module is named `module LRAM`.

If `LRAM` is a primitive in the Achronix libraries, the user design errors out in simulation or synthesis with module name conflicts.

Setting the IP Configuration

From the IP Editor, use either the **<< Back** and **Next >>** or the [Outline view \(page 106\)](#) to navigate the editor pages, setting the appropriate values needed for the desired configuration. Any errors and warnings are displayed in the [IP Problems view \(page 71\)](#). Some IP editors also display supplemental graphical information in the [IP Diagram view \(page 69\)](#).

In addition to the list of problems within the IP problems view, to the left of most fields (sometimes also called properties) there is a button with an icon indicating the validity of the value in that field. The green checkmark () indicates the value in the field has no problems. A warning () or error () icon is shown when the field value does have one or more problems. The tooltip for the button then shows the problem being reported for the associated field. Clicking the button transfers the application focus to the IP problems View, and within that view, selects all the problems associated with that field.

Note

In complicated IP, where there are many interactions between fields, there might be more than one problem entry associated with a single field.

Editable Fields

Most of the properties/fields within the IP Editor pages are editable and correspond (sometimes loosely) to parameters in the underlying Verilog macros.

Editable fields are meant to be modified in a top-down, left-to-right order. This order is recommended because some of the field values affect the validity of downstream values, and ACE often tries to help keep configurations legal by automatically changing downstream values when they are incompatible with newly-edited upstream values.

Often, editable fields may be temporarily disabled when upstream choices cause the field to become irrelevant, or to have only a single legal value. Disabled editable fields become read-only and are shown with an alternate background color, typically grey.

Tip

Modifications should be made to fields within the IP configurations editors in a top-down, left-to-right order. When editing an upstream value, it often causes downstream values to be overwritten without warning.

Calculated Fields

Some of the fields in the IP editor pages are never editable. These fields contain calculated values based upon the current contents of user-editable fields. These calculated fields are provided for informational purposes.

Many of these calculated values have limited ranges of legal values — when the calculated value falls outside the legal range, the calculated value color changes to indicate a problem. As when user-editable IP configuration properties fall outside a legal range, an IP problem entry (see [IP problems view \(page 71\)](#)) is created. But an IP problem created by a calculated value field does not "blame" the calculated value field, it instead blames one of the user-editable properties involved in its calculation. While only one field is blamed in an IP problem entry, be aware that all active fields that might be involved in the calculation are listed in the IP problem entry as potential fields which, when changed, might fix the IP problem

Note

While only one field is allowed to be blamed in an IP problem entry, remain aware that all active fields that might be involved in the calculation are listed in the IP Problem entry. Any one of these listed fields, when changed, might fix the IP Problem.

IP Editor Navigation


Navigate between sequential IP editor pages by using the **<< Back** and **Next >>** buttons. When one of these buttons becomes disabled, it means there are no further pages of configuration information in the indicated direction.

The currently active page is always selected in the [outline view \(page 106\)](#). Navigate directly to a given IP configuration page simply by selecting the desired page name in the **Outline View**. Be aware that pages may be created or removed from the outline view based upon user changes to the IP configuration editable fields.

Left-click any text in the **IP diagram view** (page 69) to turn to the IP configuration editor page containing the settings for that text.

Double-click a table entry in the **IP problems view** (page 71) to turn to the IP configuration editor page containing the property being blamed for the selected IP problem.

Generating the IP Design Files

After setting the IP configuration, click the  **Generate IP Design Files** icon to open the **Generate IP Design Files dialog** (page 172). Select the desired options such as whether to generate the Verilog wrapper, VHDL wrapper, timing constraints, placement constraints, etc. After selecting the desired options and file paths, click **Finish** to create the selected files.

Note

The generated VHDL RTL wrapper is not standalone. It requires the generated Verilog RTL file.

Adding Configuration Files to a Project


Existing configuration files (from other projects, or that were removed from the current project at some point) can be manually added to the active project. Use the procedure under **adding source files** (page 297) to add the configuration file and its related source files to the active project.

Viewing the Floorplanner

This section covers working with the floorplanner.

Opening and Closing the Floorplanner's Fly-Out Palette

To open and close the Floorplanner view fly-out palette of view options:

1. Click the  **Fly-out** button on the far right side of the **Floorplanner View** (page 43) to open the fly-out palette.

Note

While the fly-out palette is open, it may be resized by clicking and dragging its left border.

2. When the view options are configured, click the  **Fly-in** button on the left side of the fly-out palette to close the fly-out palette.

Zooming the Floorplanner In and Out

There are several ways to zoom in and out in the **Floorplanner View** (page 43).

Note

Zoom levels are always in powers of 2 (i.e., zoom in is at 200% and zoom out is at 50%). Therefore, it might not be possible to zoom in to perfectly fit a given area.


To zoom in and out with the mouse wheel:

1. Hover the mouse cursor over the desired point from which to zoom in or out in the Floorplanner view.
2. Move the mouse wheel forward to zoom in or backward to zoom out.


To zoom in and out using keystrokes:

1. Hover the mouse cursor over the center of the desired area from which to zoom in or out in the Floorplanner view.
2. Type either "Z" or "+" on the keyboard to zoom in or "z" or "-" to zoom out.



To zoom in and out using the **Zoom Tool**:

1. Select the () **Zoom Tool** from the view toolbar.
2. To zoom in on an area, click in the upper left corner of the area desired and drag the mouse to the lower right until the zoom rectangle encloses the area desired. To zoom out, click the point on the Floorplanner view from which to zoom out and drag the mouse to the upper left until the zoom out label indicates the desired zoom level.

To zoom in and out with the **Placement Tool**:

1. Select the () **Placement Tool** from the view toolbar.
2. Hover the mouse cursor over the point from which to zoom in or out in the Floorplanner view. Click the left mouse button to zoom in or the right mouse button to zoom out.

To zoom in and out with the **Zoom In** and **Zoom Out** buttons:

1. Pan to the area from which to zoom in to or out in the Floorplanner view.
2. Click the () **Zoom In** button to zoom in or the () **Zoom Out** button to zoom out.

Floorplanner Panning


To pan with the scroll bars:

1. Click and drag the vertical scroll bar to pan up and down or click and drag the horizontal scroll bar to pan left and right.
2. In Linux, place the mouse cursor over a scroll bar, then roll the mouse wheel.



To pan with key-strokes:

1. Use the arrow keys on the keyboard to pan left, right, up and down.
2. To scroll faster, press the CTRL key while pressing the arrow keys.

To pan with the **Panning** tool:

1. Select the () **Panning** tool from the view toolbar.
2. Click and drag the view with the mouse to pan around.

To pan with the **Placement** tool:

1. Select the () **Placement** tool from the view toolbar.
2. Ensure that drag-scrolling is enabled. This setting can be toggled by tapping the "Q" key, or by clicking the () **Toggle Drag-Scrolling** tool bar button.


3. Begin dragging an instance. When close to any of the view edges, the view automatically scrolls in that direction. The size of the view "drag scroll margins" as well as the speed the view drag-scrolls, can be adjusted on the [Floorplanner view Colors and Layers Preference page \(page 194\)](#).

 **Tip**


If panning the view is too slow, in the [Floorplanner view Optimizations Preference page \(page 200\)](#), there is a setting, **When panning, show only background layer**, to improve panning/scrolling performance by reducing the amount of graphic rendering performed during the pan/scroll operation.

Selecting Floorplanner Objects

To select objects with keystrokes:

1. In the () **Selection** section of the view fly-out palette, check the object types to select.
2. Press and hold the S key on the keyboard to start a selection rectangle at the current mouse cursor position (clearing/replacing the previous selection).
Optionally, press and hold the SHIFT+S keys to start a selection rectangle at the current mouse cursor position (adding to the current selection instead of replacing it).
3. Drag the mouse while holding down the key or keys on the keyboard to create a selection rectangle which includes the objects desired.
4. Release the key(s) to apply the selection.

To select objects with the Selection Tool:


1. Click the () **Selection Tool** on the view toolbar.
2. From the **Selection** section of the view fly-out palette, check the object types you wish to select.
3. Also, ensure the **Action** control in the fly-out palette is set to **Select**.
4. Click the desired object, or click and drag with the left mouse button in the view to create a selection area rectangle (clearing/replacing any previous selection).
Optionally, hold the CTRL key when clicking/dragging (adding to the previous selection instead of replacing it).
5. Release the mouse button to apply the selection.

Additionally, right-click individual objects and select **Add to Selection** from the context menu popup.


Further details about the ACE selection set are available on the [Selection View \(page 133\)](#) page.

Deselecting Floorplanner Objects

To deselect objects with key strokes:

1. Select the () **Selection Tool** from the view toolbar.
2. From the **Selection** section fly-out palette, check the object types to deselect.
3. Press and hold the D key on the keyboard to start a selection rectangle at the current mouse position.
4. Drag the mouse while holding down the key to create a selection rectangle including the objects to deselect.
5. Release the D key to remove the objects within the rectangle from the current selection set.

To deselect objects with the **Selection Tool**:

1. Select the () **Selection Tool** from the view toolbar.
2. From the **Selection** section of the fly-out palette, check the object types to deselect. Also, ensure the Action control is set to **Deselect**.
3. Click and drag with the left mouse button in the view to create a selection rectangle.
4. Release the mouse button to remove the objects from the current selection set.

Toggling Floorplanner Mouse Tools

To toggle the mouse tools:

1. Press the ALT key on the keyboard to switch between tools, or simply click the desired mouse tool on the view toolbar.

Filtering the Floorplanner View

It is often useful to filter the **floorplanner view** (page 43) graphics to see only objects of interest.

Filtering with Layers

Simple filtering of the view by object type is accomplished with the **Layer** options in the view fly-out palette.

By checking or unchecking the individual layers, all members of a given object type may be shown or hidden. Sites, Instances, Clock Routes, and Non-clock Routes may be manipulated in this way.

Because routes are always painted on top of instances, which themselves are painted on top of sites, it is often necessary to disable the painting of the topmost layers when they obscure the lower layers.


Note

The hiding of individual objects of a given object type layer may be overridden if that object is selected or highlighted, depending upon the current preference settings. See the **floorplanner view colors and layers preference page** (page 194) for more information about changing these preferences.

Filtering with Selection

Selection overrides the layer filters. When a layer is turned off, selected objects (those in the current ACE selection set) remain visible. So, for example, to see just the selected instances, turn off the instances and routes layers. The selected instances remain visible, while all other placed instances (and all non-selected nets) are hidden.

To filter with selection in the floorplanner view:

1. Add the desired objects to the current ACE selection set.
2. In the () **Layers** section of the fly-out palette, un-check the object types to hide.

Choosing Floorplanner Object Tooltips

For instant feedback on instance, net, or site names in the **floorplanner view** (page 43), a tooltip (hover text) can be enabled. In addition, the contents of the tooltip can be printed to the **Tcl Console View** (page 142) for easy copy and paste.

To get object tooltip text:

1. In the () **Tool Tip Text** section of the **fly-out palette** (page 47), enable the checkboxes for the object types with data that should be contained in the tool tip text.
2. In the floorplanner view, hover the mouse cursor over objects to display the tool tip text.



Tip


Capturing Tooltip Content:

Optionally, press the P key on the keyboard while tooltip text is visible to print the tooltip text to the TCL console view, allowing easy copy and pasting to create TCL commands or scripts.

Viewing Floorplanner Object Labels

A variety of object labels are available when displaying objects in the **floorplanner view** (page 43) (see "Fly-Out Palette").

To display object labels in the Floorplanner view:

1. In the () **Labels** section of the fly-out palette, select which object labels to display.
2. Pan and zoom to objects of interest to view the object labels.



Note

Some labels are not painted unless the view is zoomed in far enough to display the full extent of the text.

Highlighting Objects in the Floorplanner View

There is typically a tremendous amount of visualization data available in the **floorplanner view** (page 43). Because viewing all of the data simultaneously can be overwhelming, ACE provides tools such as selection and highlighting so that a particular subset of the entire design may be visualized within the floorplanner view. For simple, short-term subset visualizations, this functionality is provided by the ACE selection set as managed in the **selection view** (page 133). For longer-term visualizations, or to simultaneously compare and contrast multiple design subsets, ACE provides the

highlight (page 664)




Tcl command. As with the ACE selection set, applied highlights are visibly displayed on placed/routed objects in the floorplanner view. Highlight colors are also shown in several tabular views such as the **netlist browser view** (page 79), where the highlight colors of instances are displayed in their own table column.




Note

Only instances, nets, and paths may be highlighted.

Currently highlights are only supported for individual instance, net, and path object types (remember to use the correct **object type prefixes** (page 335) when using the Tcl commands).

Most of the views within the floorplanner perspective provide context-sensitive functionality to manage highlights, through buttons in each supporting view to () **Highlight** chosen objects, () **Un-highlight** chosen objects, or to () **Choose Highlight Color** which is next used from that view (each view tracks an active highlight color

independently of the other views, allowing one color for a selection highlight, and an alternate color for a netlist browser highlight). Additionally, some of the views (typically those representing multiple aggregations of objects) include a button to () **Auto-highlight** all objects within that view, with each aggregation automatically using a different color.

 **Caution!**

In tabular views (such as the **clock domains view** (page 20)), highlights may be shown and manipulated for aggregations of objects (as with the highlight color of a clock domain row representing the shared highlight color of all instances within that clock domain).

Be aware that if there are multiple highlight colors within the aggregation, then no highlight color is shown for the aggregation row in the table. The aggregation only displays a highlight color in the table if every single object within the aggregation has the exact same highlight color.

Additionally, when a new highlight (or un-highlight) is applied to an aggregation, it affects all individual members of that aggregation. The highlight color of all contained individual objects are overwritten with the new highlight value.

The following Tcl commands are available to manage highlights:

- **highlight** (page 664)
- **apply_highlights** (page 615)



 **Note**

There is no specific Tcl command to remove existing highlights. Instead, exclude the `-rgb` flag when calling `highlight`, which effectively applies a non-highlight to the specified object(s).

Selection vs. Highlighting

Despite some similarities, selection and highlighting serve two different purposes in ACE. They are compared and contrasted in the table below.

Table 142 - Selection and Highlighting Differences

Selection	Highlighting
There is a single ACE Selection Set.	Each object in a design may have its own unique highlight color, or a single highlight color may be applied to multiple objects, even if they are different object types.
The Selection color (a very bright green by default) is managed globally for each object type, through the floorplanner view colors and layers preference page (page 194).	Each Tcl call to <code>highlight</code> an object must specify which color to use for that call. When using the () Highlight action from within a view, the () Choose Highlight Color for that view is used.
A selection is very short-term, and is never saved/loaded between sessions.	A highlight is expected to be long-term, and highlight colors on objects are saved in <code>.acxdb</code> files when Implementations (page 229) are saved. As a result, prior highlights are restored when an implementation <code>.acxdb</code> file is loaded.

Selection	Highlighting
Selection may be applied (through the appropriate view) to aggregations of objects	Highlights may also be applied (through the appropriate view) to aggregations of objects.
Selection membership may be managed through the selection view (page 133)	There is presently no special view to manage highlights.
The selection color of an object (or aggregation) is only rendered within the floorplanner view.	The highlight color is rendered not only within the floorplanner view, but also (sometimes as individual objects, sometimes in aggregate) as a color tile in most of the tabular supporting views of the floorplanner perspective. When the highlight color is displayed in a tabular view, it is typically shown within its own column of color tile values.

Objects May Be Both Selected and Highlighted Simultaneously

It is possible for objects to be both selected and highlighted at the same time. When this occurs, the exact precedence of the color used to render the object is handled differently depending upon the object type, and which view is being rendered.

When paths are displayed in the floorplanner view, the selection color for a path takes precedence over the highlight color (the selection color for a path is managed on the [floorplanner view colors and layers preference page \(page 194\)](#)).

When instances are displayed in the floorplanner view, the selection color of an instance takes precedence over every other color (the selection color for an instance within the floorplanner view is managed on the [floorplanner view colors and layers preference page \(page 194\)](#)).

Note

It is possible to change the relative render priorities of some of the states of an instance on that same floorplanner preference page. The [instance states \(page 262\)](#) section discusses this in further detail.

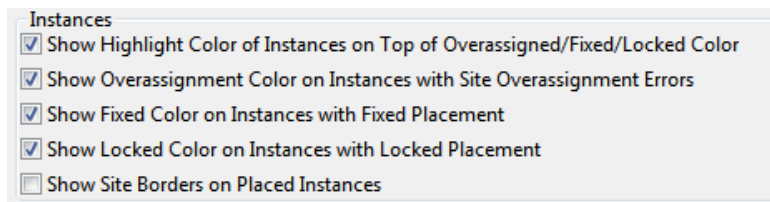


Figure 151 - Floorplanner Preferences Instance State Render Management

Uniquely among the object types, nets may display both their selection and highlight states simultaneously in the floorplanner view, though this is disabled by default (for performance reasons). By default, when nets are displayed in the floorplanner view, the selection color takes precedence over the highlight color. But nets, being simple lines, are handled specially, as configured in the [floorplanner view colors and layers preference page \(page 194\)](#). There, it is possible to choose to have the net highlight rendered on top of the net selection, or vice versa. It is also possible to choose which line thickness (width) shall be used to render both the selection line and the highlight line for the net. By making the bottom line thicker than the top line, it is thus possible to have a "halo" effect of one color outlining the other color for the same net(s).

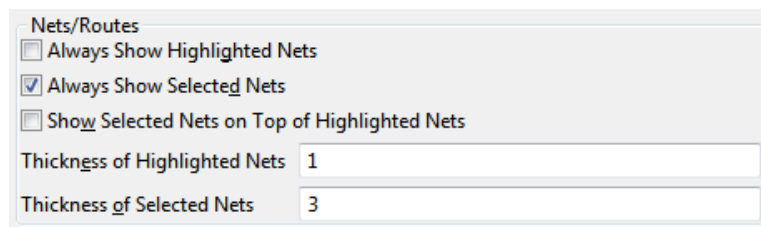


Figure 152 • Non-Default Preference Configuration Showing Both Selection and Highlights for Nets

Warning

Floorplanner Performance:

Choosing to render highlighted nets thicker than a single pixel wide might have a significant negative performance impact upon floorplanner rendering speeds of large designs at some customer sites. Exact details depend upon specifics of the workstation hardware, OS/kernel version, and the active desktop graphics rendering library.

Rendering selected nets at a thickness greater than one is expected to have little-to-no performance impact upon floorplanner rendering speeds.

Batch Mode Highlighting

Batch-mode highlights in Tcl are supported through the use of the optional `-batch` command-line argument for the [highlight](#) (page 664) command. This flag blocks the transmission of incremental highlight updates to the ACE GUI, significantly speeding up execution times when applying multiple scripted highlight changes in sequence. When all batched changes to the highlights have completed, [apply_highlights](#) (page 615) must then be called to send the highlight updates to the GUI for display.

Example Tcl sequence to highlight all instances orange, except instance names starting with "temp"

```
highlight [find {*} -insts] -rgb {255 128 0} -batch ;# applies orange highlight to all
insts
highlight [find {temp*} -insts] -batch ;# removes highlights from all insts
starting with "temp"
apply_highlights ;# sends pending highlight changes
to GUI
```

Pre-Placing a Design

This section details manual pre-placement of instances in ACE.

Placing an Object

Currently in ACE, there are two types of objects that can be placed: instances and ports. Placing a port is equivalent to placing the instance to which the port is connected in the design.

There are two types of manual pre-placement in ACE: soft and fixed. Fixed placement locks the placement of an instance to a site such that the placer is not allowed to move the instance to another site. Fixed placement is the only type of pre-placement command recommended. Soft placement is used as a global placement hint to the placer.

Caution!

Soft placement is not fully functional.

To begin pre-placement activity, the **Run Prepare** flow step for the active implementation must be run. Placing an object automatically resets the flow status to start over from the **Run Prepare** step.

To place an object from the [Search View \(page 129\)](#), [Selection View \(page 133\)](#), or [Netlist Browser View \(page 79\)](#):

1. In the **Placement** () section of the [Floorplanner View \(page 43\)](#) fly-out palette, check the placement options desired.

Note

Fixed placement is recommended for all pre-placement.

2. Pan and zoom the Floorplanner view until the destination placement site is visible.
3. In the starting view (Search view, Selection view, or Netlist Browser view), click and drag the desired object from the starting view onto the desired destination placement site in the Floorplanner view.

Note


The icon on the now-placed object in the Search view, Selection view, and/or Netlist Browser view is now updated to reflect the placement status.

To re-place an object within the Floorplanner view (move from one placement site to another):

Note


This operation alters an existing site assignment (placement) for an instance. Sometimes it is helpful to start with a fully placed and routed design, and then fine-tune the placement using this operation.

Be aware that changing the placement of an already-placed object clears the current routing data for all connections to and from that instance.

1. In the **Placement** () section of the fly-out palette, check the placement options desired.

Note

Fixed placement is recommended for all pre-placement.

2. Click the **Placement Tool** () on the view toolbar to change the mouse behavior within the view to drag and drop placement mode.
3. Pan and zoom the Floorplanner view until both the to-be-moved instance and its intended destination placement site are visible.
4. In the Floorplanner view, click and drag the placed instance from its current placement site onto the new placement site.

Changing Between Fixed and Soft Placement


There are two types of placement in ACE: soft and fixed. Fixed placement locks the placement of an instance to a site such that the placer is not allowed to move the instance to another site. Fixed placement is the only type of pre-placement command recommended. Soft placement is used as a global placement hint to the placer.

Caution!


Soft placement is not fully functional.

Fixing placement of soft-placement objects

To fix placement with key-strokes:

1. In the **Selection** () section of the **Floorplanner View** (page 43) fly-out palette, check the object types having placement that should be fixed.
2. Press and hold the **f** key on the keyboard to start a selection rectangle at the current mouse position.
3. Drag the mouse while holding down the **f** key on the keyboard to create a selection rectangle which includes the objects desired.
4. Release the key to fix the placement of the enclosed objects.

To select objects with the Selection Tool:


1. Select the **Selection Tool** () from the **Floorplanner View** (page 43) toolbar.
2. From the **Selection** section of the fly-out palette, check the object types having placement that should be fixed. Also, ensure the **Action** control is set to **Fix Placement**.
3. Click and drag with the left mouse button in the view to create a selection rectangle. Optionally, hold **CTRL** while dragging to add to the selection.
4. Release the mouse button to fix the placement of the activated objects in the selection.

Note


Not using **CTRL** clears the previous selection!

Un-fixing (softening) placement of fixed-placement objects

To un-fix placement with key-strokes:

1. In the **Selection** () section of the **Floorplanner View** (page 43) fly-out palette, check the object types with placement that should be un-fixed.
2. Press and hold the **u** key on the keyboard to start a selection rectangle at the current mouse position.
3. Drag the mouse while holding down the **u** key to create a selection rectangle which includes the desired objects.
4. Release the key to un-fix the placement of the enclosed objects.

To select objects with the Selection Tool:

1. Select the **Selection Tool** () from the **Floorplanner View** (page 43) toolbar.
2. From the **Selection** section of the fly-out palette, check the object types with placement to be fixed. Also, ensure the **Action** control is set to **Un-fix Placement**.
3. Click and drag with the left mouse button in the view to create a selection rectangle. Optionally, hold **CTRL** down to add to the selection.
4. Release the mouse button to un-fix the placement of the activated objects in the selection.

 **Note**

Not using **CTRL** clears the previous selection!

Group Placement Mode

 **Caution!**

Advanced Functionality


Group placement mode is advanced functionality, and has multiple failure cases. Group placement should only be attempted by expert users who understand all the caveats.

During normal drag-and-drop placement operations (when **Group Placement** is disabled), only the placement of a single instance is altered.

When **Group Placement** is enabled, ACE attempts to *shift* the placement of all instances in the current ACE selection set. Group placement cannot be used on instances that are not already placed – attempting to perform group placement on unplaced instances results in failure.

When group placement is attempted, the placement shift is based upon the relative change in placement site coordinates of a single *anchor* instance. The anchor instance is the instance which is dragged-and-dropped. The relative change is calculated based upon the coordinates of the anchor instance initial site and the destination site.



If the starting site coordinates of the anchor instance are at (X=15000, Y=30000) and the destination site coordinates for the anchor instance are at (X=20000, Y=40000), the coordinate shift is (X=+5000, Y=+10000). For each instance in the selection set, this coordinate shift is applied to that instance starting site coordinates; the resulting X and Y values are the coordinates where the destination site is sought for that instance. If no destination site is found at those adjusted coordinates, the entire group placement adjustment is aborted for all instances, and none of the instances are moved. All instances are left untouched in their initial placements, including the dragged-and-dropped anchor instance.

 **Tip!**

Reminder: The anchor instance must already be a member of the ACE selection set when the drag is initiated. All other instances in the selection set must also already be placed before the group placement drag is initiated.

When choosing a drop location for the anchor instance, keep in mind that all other instances in the selection set must have sites at the same relative coordinate offsets from both the anchor instance starting placement and ending placement. If an ending placement site is not found for any instance at the expected coordinate offsets, the entire group placement adjustment operation is aborted, and all instances remain in their initial placements.

Adjusting the Existing Placement of a Group of Selected Instances

1. Empty the ACE Selection Set (as seen in the [Selection View \(page 133\)](#)).
2. Select all the instances that should take part in the group placement adjustment, so that they are added to the ACE Selection Set.
3. Ensure that all instances in the ACE Selection Set are already placed (the icon for the instances in the Selection View should be either  for Soft Placement or  for Fixed Placement).
4. Ensure that the **Fixed Placement** checkbox (within the [Floorplanner View \(page 43\)](#) fly-out palette) is in the desired state.
5. Enable **Group Placement** mode by clicking the checkbox (within the Floorplanner View fly-out palette).
6. Choose which placed, selected instance is to be the anchor instance, then scroll and zoom the Floorplanner until both the initial site and destination site for the anchor instance are plainly visible.
7. Double-check that all other instances in the selection set have corresponding destination sites.
8. Drag the anchor instance from its initial placement to the destination site. If all initial requirements are met, and if destination sites were found for all selected instances at the calculated offsets, the GUI will issue a bulk Tcl [set_placement \(page 713\)](#) command for all selected instances, and the instances should move to the new sites.
9. Disable **Group Placement** mode by clicking the checkbox (within the Floorplanner View fly-out palette).

 **Caution!****Group Placement pays attention to the Fixed Placement checkbox:**

Be aware that when a group placement adjustment succeeds, when the new placement is applied, the current state of the Floorplanner View **Fixed Placement** checkbox affects all adjusted placements.


If **Fixed Placement** is checked, all adjusted placements are fixed, regardless of whether they initially had soft or fixed placement. If **Fixed Placement** is unchecked, all adjusted placements are soft, regardless of whether they initially had soft or fixed placement.

All routing to and from the moved instances will be cleared after the placement.



See also: [Floorplanner View \(page 43\)](#), [Selection View \(page 133\)](#), [Selecting Floorplanner Objects \(page 341\)](#), [Deselecting Floorplanner Objects \(page 341\)](#), [select \(page 708\)](#), [deselect \(page 625\)](#), [set_placement \(page 713\)](#)

Removing Placement

To un-place objects with key-strokes in the [Floorplanner View \(page 43\)](#):

1. Ensure the **Instances** checkbox is checked in the **Selection** () section of the view Fly-out Palette.
2. With the mouse positioned in the Floorplanner view, press and hold the **r** key on the keyboard to start a selection rectangle at the current mouse position to remove placement of objects.
3. Drag the mouse while still holding down the **r** key to create a selection rectangle including the objects to be un-placed.
4. Release the key to un-place all the objects contained by the selection rectangle.

To un-place objects with the **Selection Tool** () in the Floorplanner view:

1. Select the **Selection Tool** () from the view toolbar.
2. In the **Selection** () section of the view fly-out palette:
 - a. Set the Action control to **Remove Placement**.
 - b. Ensure the **Instances** checkbox is checked.
3. Click and drag with the left mouse button in the view to create a selection rectangle.
4. Release the mouse button to un-place all the objects within the selection rectangle.

It is also possible to un-place objects using the right-click context menu in the Floorplanner view, [Search View \(page 129\)](#), and [Selection View \(page 133\)](#).

The fastest way to un-place multiple objects is to add them all to the ACE Selection Set (as shown in the Selection view; see [Selecting Floorplanner Objects \(page 341\)](#)), and then un-place all of them at once by performing any one of the following:


- In the Selection view, right-click the mouse on the **Instances** node in the tree, then choose **Unplace All Instances in ACE Selection Set**.
- In the Floorplanner view, right-click the mouse anywhere on the floorplan, then choose **Unplace All Selected Instances**.
- In the [Tcl Console View \(page 142\)](#), type: `run_unplace -insts [get_selection]` ". (See [run_unplace \(page 703\)](#).)

Performance Tip:

It is always faster to un-place multiple objects at once instead of individually, especially when a very complex net (like a clock net) is affected.

Saving Pre-Placement Constraints

To save the current placement to disk as pre-placement constraints in .pdc files:

1. Place objects in the design as described in [Placing an Object \(page 347\)](#).
2. In the Floorplanner view or Package view, click the **Save Pre-placement Constraints** toolbar button () to bring up the [Save Placement Dialog \(page 182\)](#).
3. Configure the dialog with appropriate options and click **Finish**.

After clicking **Finish**, the pre-placement files are saved to disk. If the option was selected to automatically add the files to the current project, the Projects view shows the new files under the active project Constraints folder.

Using Pre-Placement in the Flow

Types of Pre-Placement

There are three ways to pre-place instances in ACE:

- After the **Run Prepare** flow step, interactively pre-place instances using the ACE GUI drag-and-drop placement features, or use the `set_placement -fixed` TCL command in the TCL Console
- Include a PDC constraints file in your ACE project that uses `set_placement -fixed` TCL commands to pre-place the instances
- Set the location parameter on the instance primitive in the user design RTL (not recommended; the location parameter is effectively the same as using the `set_placement -fixed` Tcl command on that instance)

Using the `set_placement -fixed` Tcl commands in a PDC constraints file is recommended. If using the location parameter in the user design RTL, the RTL must be changed and re-synthesized to update the pre-placement.

ACE applies pre-placement from user design RTL and PDC constraints at the end of **Run Prepare** in two stages:

1. ACE loops over all instances that have the location parameter set and internally calls `set_placement -fixed` on each instance and then prints the log file message “Applying defparam placement of <instance> to location <location>”. The log file or TCL console shows messages for each instance placed with the location parameter.
2. The **Run Prepare** step applies all of the `set_placement` commands in the PDC files. If there is a `set_placement` command in your PDC for an instance that is already placed with the location parameter, the PDC `set_placement` overrides the placement set in the location parameter. There is no warning message. It is the same as having two `set_placement` commands in the same PDC file that place the same instance. The last `set_placement` command always wins.

It is recommended to use only the PDC method.

Recommended Typical Flow

To use pre-placement in the flow, it is recommended to first create the pre-placement constraints:

1. Run the **Run Prepare** flow step on the active implementation.
2. Switch to the Floorplanner perspective and place all the objects for pre-placement (using fixed placement). See [Placing an Object \(page 347\)](#) for details.
3. Save the pre-placement and automatically add it to the project (see [Saving Pre-Placement Constraints \(page 351\)](#)).
4. Optionally, a pin assignment report can be generated for the current placement with the `report_pins` (page 682) Tcl command.
5. Resume running the flow. The pre-placement data is used in the place and route solution.

When the pre-placement constraints are in place, it is recommended to include them in the project for future runs:

1. The next time the flow is run with this implementation, ensure that in the [Options View \(page 96\)](#), the new pre-placement constraints files are enabled.

2. Simply run the **Run Prepare** flow step. The pre-placement constraints are automatically applied. A pin assignment report is also automatically generated during **Run Prepare**.
3. Optionally, to see that the objects are pre-placed, switch to the Floorplanner perspective to view the placement.

Analyzing Critical Paths

Critical paths are computed by timing analysis. Timing analysis can be run at several points in the **flow** (page 234), as indicated in the **flow view** (page 53). Timing analysis can be repeated with different **implementation** (page 229) options without having to re-run the rest of the flow, by double-clicking the appropriate **Run Timing Analysis flow step** (page 234).

The results of timing analysis are shown in a **timing report** (page 244), which is automatically displayed as timing analysis completes. The most recently generated version of each timing report file is always available in every implementation **reports** sub-directory.

The active critical paths may also be viewed in the **critical paths view** (page 37) and the **critical path diagram view** (page 33). Unlike the reports, the views only show the critical paths for the **active project and implementation** (page 229). When the active implementation changes, the two views are cleared. Also, the views are only populated when timing analysis is run for the active implementation during that same ACE session. The timing analysis data is not saved in the `.acxdb` file, and must be re-created every session to guarantee correctness.



Tip

While a generated timing report may be viewed from an implementation **reports** directory at any time, including in later ACE sessions, the two critical path views only show data from the most recent timing analysis within the current ACE session.

Generating Timing Reports

A **timing report** (page 244) is generated and displayed in the GUI whenever one of the Run ... Timing Analysis **flow steps** (page 234) is run. Timing reports may also be generated at any time from Tcl by running the appropriate flow step (`run` (page 689) -step <flow_step_name>) or with the `run_timing_analysis` (page 702) Tcl command.

The accuracy of the timing analysis results will improve as more of the flow steps are executed. (The relative placement and routing may only be estimated until the corresponding flow steps are completed.)

Timing reports can be found in the **implementation** (page 229)'s reports directory, available for browsing via the **Projects view** (page 117). In addition to the HTML report files displayed in the GUI, there are equivalent report files in `.txt` (text) and `.csv` (spreadsheet) formats, generated simultaneously with the HTML reports by default.

The Timing Analysis **implementation options** (page 0) in the **options view** (page 96) determine how timing analysis is run and the amount of report information which is generated. Additionally, while the **Flow Mode** (page 242) is set to **Evaluation**, the timing reports will always be limited to a single temperature corner to minimize runtimes.

Critical paths detected during timing analysis are also displayed in the **Critical Paths View** (page 37), using the same information included within the timing reports. (If the critical paths view is empty, it can be repopulated by re-running timing analysis.) The same path ID can be used to cross-reference between the critical paths view and the timing reports.

For details about how temperature corners are handled timing analysis and reporting, see **Timing Across All Temperature Corners** (page 266).

Highlighting Critical Paths

Note

The Floorplanner can only display routed paths. Paths which are not routed cannot be displayed in the Floorplanner.

To highlight a routed critical path in the **Floorplanner View** (page 43):

1. First, run one of the timing analysis flow steps to generate critical path data.
2. Then, in the **Critical Paths View** (page 37), browse through all reported critical paths.
 - a. By default, highlight colors of setup/hold violations are arranged in a gradient from red to yellow according to the slack's distance from zero.
 - b. Paths with a positive slack (setup/hold met) are colored green by default.
3. To highlight a path in the Floorplanner, simply check the box for the desired path in the Highlight column of the table within the **Critical Paths View** (page 37). To un-highlight a path, simply uncheck the box.

Tip: Critical Path Highlight Colors May Be Changed

The highlight color of each individual critical path can be changed by clicking on the color chooser box in the Highlight column of the **Critical Paths View** (page 37) table. In the color chooser dialog, select the desired color for that path and click **OK**.


Selecting Critical Path Objects

In order to manipulate objects (for example, by pre-placing them) on a critical path, it is convenient to add them to the current ACE selection set (as displayed in the **selection view** (page 133)) for easy access.


Note

When objects are in the ACE selection set, they change to the selection color which overrides all other colors, including highlight colors.

To add a critical path to the current selection:


1. In the **critical paths view** (page 37), click the table row containing the data for the path for which objects are to be selected.
2. To add the path to the current ACE selection set, click the  **Select Path** toolbar button on the **critical paths view** (page 37) toolbar.
3. The path is now added to the selection in the **selection view** (page 133) and is shown with the selection color in the **floorplanner view** (page 43).

To add the pins of a critical path to the current selection:


1. In the **critical paths view** (page 37), click the table row containing the data for the path for which objects are to be selected.
2. To add the path pins to the current ACE selection set, click the  **Select Pins** toolbar button on the **critical paths view** (page 37) toolbar.

3. The pins are now added to the selection in the [selection view \(page 133\)](#) and are shown with the selection color in the [floorplanner view \(page 43\)](#).

To add the instances of a critical path to the current selection:


1. In the [critical paths view \(page 37\)](#), click the table row containing the data for the path for which objects are to be selected.
2. To add the path instances to the current ACE selection set, click the () **Select Instances** toolbar button on the [critical paths view \(page 37\)](#) toolbar.
3. The instances are now added to the selection in the [selection view \(page 133\)](#) and are shown with the selection color in the [floorplanner view \(page 43\)](#).

To add the nets of a critical path to the current selection:

1. In the [critical paths view \(page 37\)](#), click the table row containing the data for the path for which objects are to be selected.
2. To add the path nets to the current ACE selection set, click the () **Select Nets** toolbar button on the [critical paths view \(page 37\)](#) toolbar.
3. The nets are now added to the selection in the [selection view \(page 133\)](#) and are shown with the selection color in the [floorplanner view \(page 43\)](#).

Zooming to Critical Paths

To zoom the [floorplanner view \(page 43\)](#) to the region of a critical path:


1. In the [critical paths view \(page 37\)](#), click the table row containing the desired critical path data.
2. To zoom to the path in the [floorplanner view \(page 43\)](#), click the () **Zoom to Path** toolbar button on the [critical paths view \(page 37\)](#) toolbar.

Note

This action only applies to routed designs.

Printing Critical Path Details

To print the details of a critical path to the [Tcl console view \(page 142\)](#):

1. In the [critical paths view \(page 37\)](#), click the table row containing the data for the path for which details are to be printed.
2. To print the details text, click the () **Print Path Details** toolbar button on the [critical paths view \(page 37\)](#) toolbar.

Tip

Critical path details are also available in the [timing report \(page 244\)](#)

Using Critical Path Diagrams

The **Critical Path Diagram View** (page 33) provides a graphical representation of a single critical path. These paths are each selected from the table in the **Critical Paths View** (page 37). The graphical representations consist of circular nodes (representing instances) connected by arrows (representing one or more nets).

Tip

To quickly look at the diagrams for all the critical paths:

1. Make sure both the **Critical Paths View** (page 37) and **Critical Path Diagram View** (page 33) are visible.
2. Click a row in the Critical Paths view table.
3. Use the keyboard up arrow and down arrow keys to change which row is selected in the table.
4. The Critical Path view diagram is updated to graph the relevant critical path.

Graph Elements

The graphical diagram is made up of nodes and arrows. The information represented by the nodes, arrows, and their supporting text, can vary depending upon the current settings in the diagram **fly-out palette** (page 35).

Nodes

The larger circles in the diagram are the primary graph nodes which represent the key instances or turn points on the critical path. Intermediate nodes, when enabled, are smaller circles, representing instances the data passes through while flowing between turn points. Several useful pieces of information are available for each graph node. These may be enabled and disabled via the fly-out palette.

Note

Some information is hidden when the graph node circle is too small to contain it. To see all enabled information, the diagram must be zoomed in. Configurable tooltips can be used to see information that would otherwise be hidden due to insufficient drawing area.

Arrows

The arrows connecting the graph nodes in the diagram represent the nets connecting the object instances and can also display various pieces of information that may be enabled and disabled via the fly-out palette. In addition to the direction of the arrow, the line types making up the arrow also represent important information:

- Bold arrows, visible when the **Intermediate Nodes** fly-out palette setting is disabled, represent one or more nets and any hidden intermediate nodes, lumped into a single abstraction. Bold arrows, since they potentially represent multiple nets and hidden intermediate instances, may only display text for the cumulative time in picoseconds of their **Delays**. **Net Names** and **Fanouts** are never displayed for bold arrows, since they make no sense in this context.
- Thinner arrows are shown when the **Intermediate Nodes** fly-out palette setting is enabled. Each of these represent an individual net. Because these thinner arrows each represent individual nets connecting the instances, the individual net **Fanouts** and **Net Names** may also be displayed for each arrow, in addition to the **Delays**.

Critical Path Diagram Types

Different types of critical paths may have different visual representations. The **Type** column in the [Critical Paths View \(page 37\)](#) table provides the critical path type of each row.

All path types are displayed as a straight line of objects connected by arrows.

Adding Portions of the Graph to the ACE Selection Set

When the graph has helped track down which nets and/or instances to adjust for timing purposes, it can be helpful to find those objects in the [Floorplanner View \(page 43\)](#). To do so, use the techniques described in [Selecting Critical Path Objects \(page 354\)](#), or add specific nodes and arrows from the graph to the ACE selection set, and use the **Zoom to Selection** button in the [Selection View \(page 133\)](#) to cause the [Floorplanner View \(page 43\)](#) to scroll and zoom so that all the selected objects are visible.

Applying and Checking Properties

Applying Properties

There are three methods to apply properties, detailed below. More information and examples can be found in the Synthesis Optimizations chapter of the *Synthesis User Guide* (UG071).

defparam

Properties can be applied as defparams on a module or module port in the RTL black-box library. Applying defparams is an internal only method and sets the default value of the property for all instances of that module, or instance pins of a port.

RTL Attribute

Attributes can be set in the RTL to show the design intent and to guide both Synplify and ACE. Synplify attributes are detailed in the Synplify help manual. ACE attributes can also be set in the RTL, and these are passed by Synplify to ACE. Attributes can be applied in both Verilog-2001 and SystemVerilog formats, as shown below:

Equivalent methods for applying properties

```

reg my_reg /* synthesis syn_preserve=1, must_keep=1
*/;
(* must_keep=1 *) reg my_reg /* synthesis syn_preserve=1 */;
(* must_keep=1, syn_preserve=1 *) reg my_reg;

```

In certain cases it is necessary to set both a Synplify and an ACE attribute when if, without the Synplify attribute, the object may be optimized or reduced. For example, if it is required to keep a register, then Synplify will require the `syn_preserve` attribute to ensure the register is in the netlist output to ACE, and ACE will require the `must_keep` attribute to keep the subsequent register. Similar situations arise with directing and controlling fanout, where both

`syn_maxfan` (Synplify) and `fanout_limit` (ACE) may be required. This requirement is a result of the fact that Synplify Pro does not propagate its own `syn_*` attributes on to ACE in the gate-level netlist.

set_property Tcl Command

The `set_property` command can be applied in a PDC file to an object. See [set_property \(page 714\)](#) for full details.

```
set_property IOSTANDARD {"LVCMOS18"} {p:led_anode[0]}
```

Checking Whether Properties Were Applied

It is recommended to use the Synplify technology viewer to verify that properties were applied during Synplify. Selecting the object and then selecting "properties" will list the properties which will be passed to ACE. In addition, the netlist can be searched for the appropriate object, and the properties checked.

Within ACE, examine the object using the [Properties View \(page 123\)](#), or use the [display_properties \(page 628\)](#), [get_properties \(page 657\)](#), or [get_property \(page 658\)](#) Tcl commands.

Configuring External Connections to Hardware

ACE includes features supporting interactions with running hardware through both the JTAG and DCC interfaces.

The ACE JTAG connection utilizes a Bitporter2 pod or FTDI FT2232H (or FT4232H) device various Tcl commands to interact with an Achronix FPGA. The JTAG interface is used by the:

- [Download view \(page 40\)](#)
- [Snapshot Debugger view \(page 137\)](#)
- [HW Demo view \(page 58\)](#)

The automated SerDes link tuning functionality available for some FPGAs also uses JTAG.

Note

For more details on managing the physical connection between the workstation, the Bitporter2 pod or FTDI device, and the FPGA board, see the *JTAG Configuration User Guide* (UG004), as well as any documentation specific to the development board.

The ACE DCC connection utilizes a direct serial connection (over a dedicated USB cable, kept separate from the JTAG connection) to interact with the Achronix development board. The DCC interface is used by the [HW Demo view \(page 58\)](#).

Configuring the DCC Connection

The ACE DCC (Demo Command and Control) connection allows the ACE software to communicate with demo and/or reference designs running on Achronix hardware. The DCC connection is managed using the [Configure DCC Connection Preference Page \(page 189\)](#).

Background Info

The ACE DCC connection utilizes a direct serial RS232 connection through a simple on-board 2-pin interface, over a dedicated FTDI USB Serial Port cable. The DCC interface does not make use of the Bitporter. The DCC protocol is currently not a published standard, and is only meant to be used with Achronix demo/reference designs running on Achronix FPGAs on Achronix boards. The DCC interactions perform individual register reads and writes, and are executed using a handful of simple Tcl commands.

Installing DCC USB Drivers

The necessary USB drivers for the DCC cable are included in the ACE download. They will need to be installed before the DCC cable is connected.

Windows

When running the ACE installer, simply make sure the setting **FTDI CDM USB drivers for the Development Board DCC interface** is checked. When the installer completes, the DCC cable may be connected.

Linux (not currently supported)

The necessary FTDI USB drivers are already included in supported Linux distros. If the distro-supplied USB drivers do not work correctly, the necessary drivers may be downloaded from the FTDI website. Contact Achronix support if you need help finding the Linux drivers.

Configuring the USB drivers

The DCC cable USB drivers automatically choose a serial port to be redirected to the USB cable. It may be necessary to view and/or change which serial port is chosen.

Windows

To see which serial (COM) port is being used by the driver, go to the Windows Device Manager, and look under **Ports (COM and LPT)**. The correct COM port is shown as **USB Serial Port (COM*)**, with the actual COM port number included.

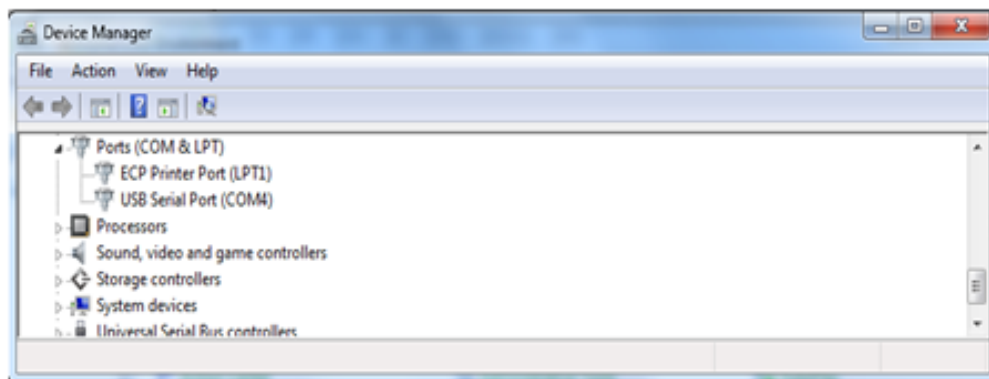


Figure 153 • Windows Device Manager COM Ports Example

To change which COM port is being used by the USB cable, right-click the USB Serial Port entry, and then choose **Port Settings** → **Advanced**, and select the alternate COM port to be used.

Linux (not currently supported)

Contact Achronix support for the latest information about finding or changing the DCC cable serial port assignment.

Configuring ACE

When it is determined which serial port is being used by the DCC cable, ACE needs to be configured to use that same serial port.

Open the [Configure DCC Connection Preference Page \(page 189\)](#) (**Window** → **Preferences** → **Configure DCC Connection**), then populate the **Port Name** field with the serial port name the DCC cable is using. The exact port names used vary according to the operating system in use.

In Windows, this is a COM port, typically one of COM1 through COM9. COM3 is the most frequent (and thus default) choice.

In Linux, the serial ports are named `/dev/ttyS*` with the `*` being replaced by a number. This is typically one of `/dev/ttyS0` through `/dev/ttyS9`.

Configuring the JTAG Connection

Achronix currently supports two JTAG programmer device types:

- Bitporter2
- FTDI FT2232H (or FT4232H)

While in the simplest cases, a test bench may only have a single JTAG programmer device connected to a JTAG scan chain containing a single Achronix FPGA (or eFPGA), many may have multiple JTAG programmer devices, and/or multiple devices within the connected JTAG scan chain. For these reasons, the ACE tools must specify which JTAG programmer device connection to use, and which JTAG scan chain member is relevant.

The various tools within ACE obtain their choice of JTAG programmer device and JTAG scan chain configuration from a few different locations, mostly based upon the usage model of the ACE tool itself.

The primary sources of JTAG programmer device and JTAG configuration information include:

- The GUI [Configure JTAG Connection preference page \(page 190\)](#)
- Command-line overrides
- The [implementation options \(page 0\)](#), as managed within the [Options view \(page 96\)](#)

The GUI views and editors responsible for live FPGA interactions retrieve their JTAG programmer device and JTAG connection details from the [Configure JTAG Connection preference page \(page 190\)](#).

The various Tcl commands that perform JTAG interactions typically require the JTAG device name to be provided as an argument, and there are specific Tcl commands to list available device connections, open and close individual device connections, and initialize the JTAG scan chain for a device with the desired configuration details. See the *JTAG Configuration User Guide* (UG004) for a complete listing of these Achronix Tcl commands in the `jtag::` and device-specific namespaces.

The JTAG scan chain implementation options are currently only used during the [FPGA Download flow step \(page 234\)](#).

Note

The `acx_stapl_player`, deprecated starting with ACE 9.2, is currently still being shipped (for backwards compatibility) as a part of the ACE software, though it is to be removed from future releases. While `acx_stapl_player` has previously been used behind-the-scenes by ACE for any JTAG interactions involving the now-obsolete STAPL `*.jam` files, it remains available for manual use from the operating system shell/command-line. The use of this tool is covered in detail within the *Configuration User Guide* (available from www.achronix.com/support/docs).

Caution!**Special note for sites with more than one connected FPGA/eFPGA:**

The interactive members of the ACE tools currently assume that only a single FPGA/eFPGA is of interest at a time, and those interactive tools all share a single configuration on the [Configure JTAG Connection preference page \(page 190\)](#). This single configuration is stored per-user, and is not unique per-design or per-implementation.

At sites with more than a single connected FPGA/eFPGA, remember to change the JTAG programmer device and JTAG scan chain values stored on that preference page every time when alternating between FPGAs/eFPGAs.

JTAG Programmer Device Connection

It is often possible that multiple JTAG programmer devices are visible from a single workstation. Because of this, ACE software must be configured to know which JTAG programmer device is intended to be used. The ACE GUI JTAG programmer device and JTAG connection details are managed primarily through the [Configure JTAG Connection preference page \(page 190\)](#) (for interactive GUI tools) and the implementation options within the [Options view \(page 96\)](#) (for any JTAG tools called within the [flow \(page 234\)](#)).

Warning!**Bitporter2 pods may be damaged if improperly connected!**

Read the *Configuration User Guide*, as well the user guide(s) specific to your development kit, before physically connecting the Bitporter2 JTAG ribbon cable to the JTAG header on the development board.

Important!

This section describes how to configure ACE to communicate with an already-connected JTAG programmer device. For more details on managing the physical connection between the workstation, the JTAG programmer device, and the FPGA board, see the *Configuration User Guide*, as well the user guide(s) specific to the development kit, and any related release notes.

The *Configuration User Guide* also covers additional details about testing JTAG programmer device connections, JTAG programmer device naming/addressing, and managing connections to multiple devices.

Starting in ACE 9.2, ACE uses the Achronix `jtag : Tcl` library to perform all ACE JTAG operations. High-level JTAG Tcl commands may automatically detect the presence of available JTAG devices over USB, though all allow specifying JTAG devices by name. If multiple JTAG devices are detected, and ACE has not been informed which of

the JTAG devices (by name) should be used, the Tcl command execution fails because, for safety, ACE does not pick a JTAG device randomly.

It is strongly recommended to specify the chosen JTAG device by name in all cases, instead of relying upon auto-detection. Not only does this avoid problems if additional JTAG devices are connected at a later date, but connections to named devices are faster to initiate.

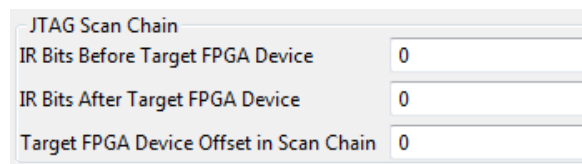
Tip

Several seconds of initialization time can be saved on every JTAG connection if the JTAG programmer device is specified by name instead of using auto-detection.

The details of JTAG device naming and name retrieval are covered thoroughly in the *Configuration User Guide*. As a simple summary, the device names for supported JTAG devices are derived from their device serial number. The Tcl command `jtag::get_connected_devices` provides a list of connected JTAG device names.

JTAG Scan Chain

The JTAG specification (see en.wikipedia.org/wiki/JTAG) supports multiple devices being connected in sequence, sharing a single set of JTAG pins on the board. These devices are treated as being on the same JTAG scan chain. In order for ACE to successfully communicate with any target device in a scan chain, ACE must be told the scan chain configuration on the [Configure JTAG Connection preference page \(page 190\)](#).



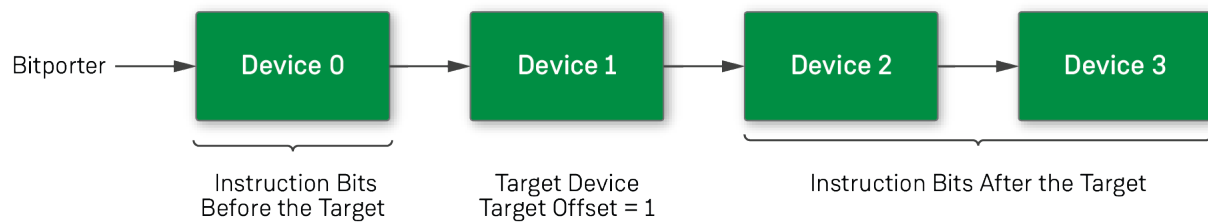
JTAG Scan Chain	
IR Bits Before Target FPGA Device	0
IR Bits After Target FPGA Device	0
Target FPGA Device Offset in Scan Chain	0

Figure 154 • JTAG Scan Chain Fields, Configure JTAG Connection Preference Page

Note

The default JTAG scan chain preference values, all zeros, is always correct for single-device scan chains. For multi-device scan chains, the default values never work.

When multiple FPGA devices are attached to the same JTAG scan chain, the target FPGA device must be specified. Because different FPGA devices have different instruction sizes, the total instruction length before and after the target must be specified as well.



557324-01.2023.05.20

Figure 155 • Multi-device Scan Chain with Bitporter Example

The **Target FPGA Device Offset in Scan Chain** specifies the ordinal position relative to the Bitporter. The device closest to the Bitporter (technically, the device closest to the board JTAG TDI pin) has a target offset of 0.

The number of instruction register (IR) bits before the target FPGA device is specified under **IR Bits Before Target FPGA Device**, while **IR Bits After Target FPGA Device** specifies the number of IR bits that follow the target FPGA device in the chain. Achronix FPGA devices have an instruction size of 23 bits. Hence, in the above example, if all devices were Achronix FPGA devices, there would be 23 instruction bits before the target, (23 instruction bits in the target), and 46 instruction bits after the target.

In JTAG, the least significant bit enters the scan chain first, while the most significant bit enters the scan chain last. From the perspective of ACE, *before* refers to the more significant bits in the scan chain, and *after* refers to the less significant bits. Instruction bits before the target are not scanned through the target FPGA. Instruction bits after the target instruction bits are scanned through the target FPGA before arriving at their scan chain destination. The key detail is that ACE regards the before/after terminology from the perspective of where the bits ultimately land in the instruction registers, NOT in terms of when the bits pass through the JTAG TDI pin, and NOT in terms of the sequence in which the bits pass through the target device.

Thus, in a chain of four Achronix FPGAs (each FPGA instruction register consists of 23 bits, the total IR bits = $4 \times 23 = 92$ IR bits), to specify the device closest to the TDI pin, the initial device (IR scan chain bits [91:69]) requires an entry of 0:69:0 for the three ACE scan chain configuration values. The first 0 for **IR Bits Before Target FPGA Device**, 69 for **IR Bits After Target FPGA Device**, and 0 for **Target FPGA Device Offset in Scan Chain**. The second Achronix FPGA in a chain of four would be 23:46:1, the third would be 46:23:2, and the fourth would be 69:0:3.

Table 143 • Example Scan Chain Values: Four Achronix FPGAs in the Same Scan Chain

Device (IR bit Range Within 92-bit IR Scan Chain)	IR Bits Before Target FPGA Device	IR Bits After Target FPGA Device	Target FPGA Device Offset in Scan Chain
0 (bits [91:69])	0	69	0
1 (bits [68:46])	23	46	1
2 (bits [45:23])	46	23	2
3 (bits [22:0])	69	0	3

Specifying a single-device chain (where there is nothing in the chain except the solo target device) would always require an entry of 0:0:0. There are zero IR bits before the target device, zero IR bits after the target device, and it is the device closest to the TDI pin in the JTAG scan chain. This single-device scan chain configuration is the default configuration.

Note

Instruction register lengths vary by vendor and device.

For those new to JTAG, it might be worth mentioning that if non-Achronix devices are in the scan chain, it is extremely likely that their instruction registers are not 23 bits long, thus the before/after bit counts required would not be multiples of 23.

The scan chain Offset number is independent of the IR bit numbers, and is used to derive data register pre- and post-padding, since according to the JTAG specifications, devices being bypassed each always have a DR length of one.

Running the Snapshot Debugger

The following sections describe how to configure and use the snapshot debugger in an end-user design.

Note


Snapshot hardware architecture details and use of the `ACX_SNAPSHOT` user macro can be found in the *Snapshot User Guide* (UG016), though there may also be supplemental details specific to some Achronix device families. These documents are available from www.achronix.com/support/docs, download.achronix.com (account required), or from an Achronix FAE.

The Snapshot content here in the ACE User Guide provides a general overview of functions which are common to all Achronix devices, and is focused on the snapshot debugger user interface for real time in-system debugging.

Snapshot Design Flow

Warning!

The JTAG connection must be configured before using the snapshot debugger.

ACE interacts with the FPGA using the JTAG interface through a Bitporter2 pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the Snapshot Debugger view. The configuration is managed using the [Configure JTAG Connection preference page \(page 190\)](#), which is easily accessible by clicking the () **Configure JTAG Interface** button in the Snapshot Debugger view. See [Configuring the JTAG Connection \(page 360\)](#) for more details.

Snapshot is the real-time design debugging tool for Achronix FPGAs and eFPGAs. Snapshot, which is embedded in the ACE software, delivers a practical platform to evaluate the signals of a user design in real-time and optionally send stimuli to the user design.

To utilize the Snapshot debugger tool, the Snapshot macro must be instantiated inside the RTL for the design under test (DUT). After instantiating the macro and programming the device, the design can be debugged in the ACE GUI

using the [Snapshot Debugger view \(page 137\)](#) and the [VCD Waveform Editor \(page 11\)](#), found within the [Programming and Debug perspective \(page 6\)](#).

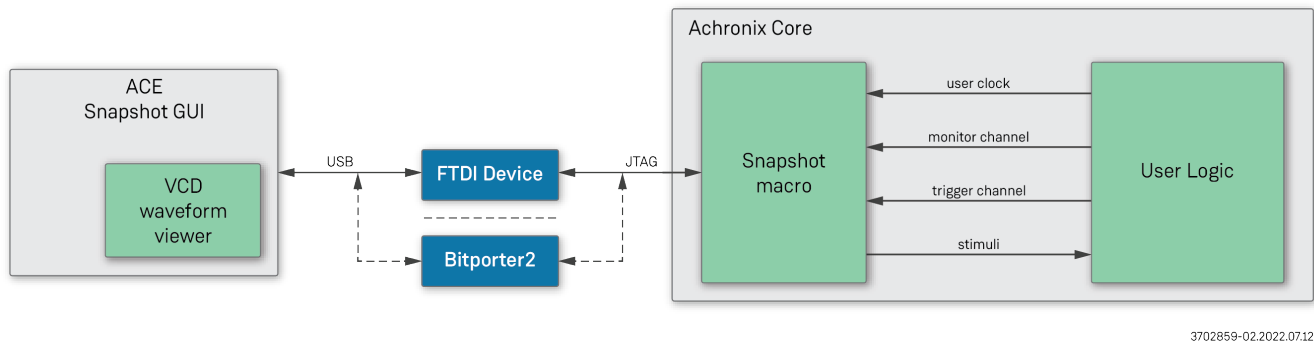


Figure 156 • Snapshot Communication with the Snapshot Debugger View within ACE

When instantiated in a design, the Snapshot macro can be used to interface with any logic mapped to the Achronix FPGA core. The Snapshot macro provides a JTAG/JTAP interface to control/observe debug logic mapped to the core. This interface allows the ACE Snapshot Debugger view, which drives the JTAG interface, to control/observe the signals associated with the debug logic.

Within the ACE GUI, the Snapshot Debugger view allows configuring an embedded Snapshot Debugger core, interactively arm the core, and generate a VCD waveform output of the collected samples. By default, the generated VCD waveform output is displayed in the ACE editor area using the [VCD Waveform Editor \(page 11\)](#). The VCD output can also be read into a third-party waveform viewer.

At a high level, to utilize Snapshot, first:

1. Instantiate the Snapshot macro `ACX_SNAPSHOT` in the user design.
2. Set the required constraints in the `.sdc` files.
3. Synthesize the design.
4. Place and route the design in ACE.
5. Generate the Bitstream for the design in ACE.
6. Configure the ACE JTAG connection to the FPGA (see [Configuring the JTAG Connection \(page 360\)](#))
7. Program the Achronix device with the Bitstream.
 - Use of the ACE GUI [Download view \(page 40\)](#) is documented in the section [Programming a Device using JTAG in the Download View \(page 390\)](#)
 - Use of the library of JTAG commands executable on the Tcl command-line is documented in the [JTAG Configuration User Guide \(UG004\)](#)

When these prerequisite steps are complete, the ACE GUI [Snapshot Debugger view \(page 137\)](#) allows the evaluation/interaction with the running design in real-time.

The following sections further explain Snapshot and provide a guide through the process.

Accessing the Snapshot Debugger

Open the ACE GUI and Select the Project

Open the ACE GUI tool, and load or activate the selected project in the Projects View as shown below. See:

- [Loading Projects, \(page 295\)](#)
- [Setting the Active Implementation \(page 303\)](#)
- [Working with Projects and Implementations \(page 293\)](#)

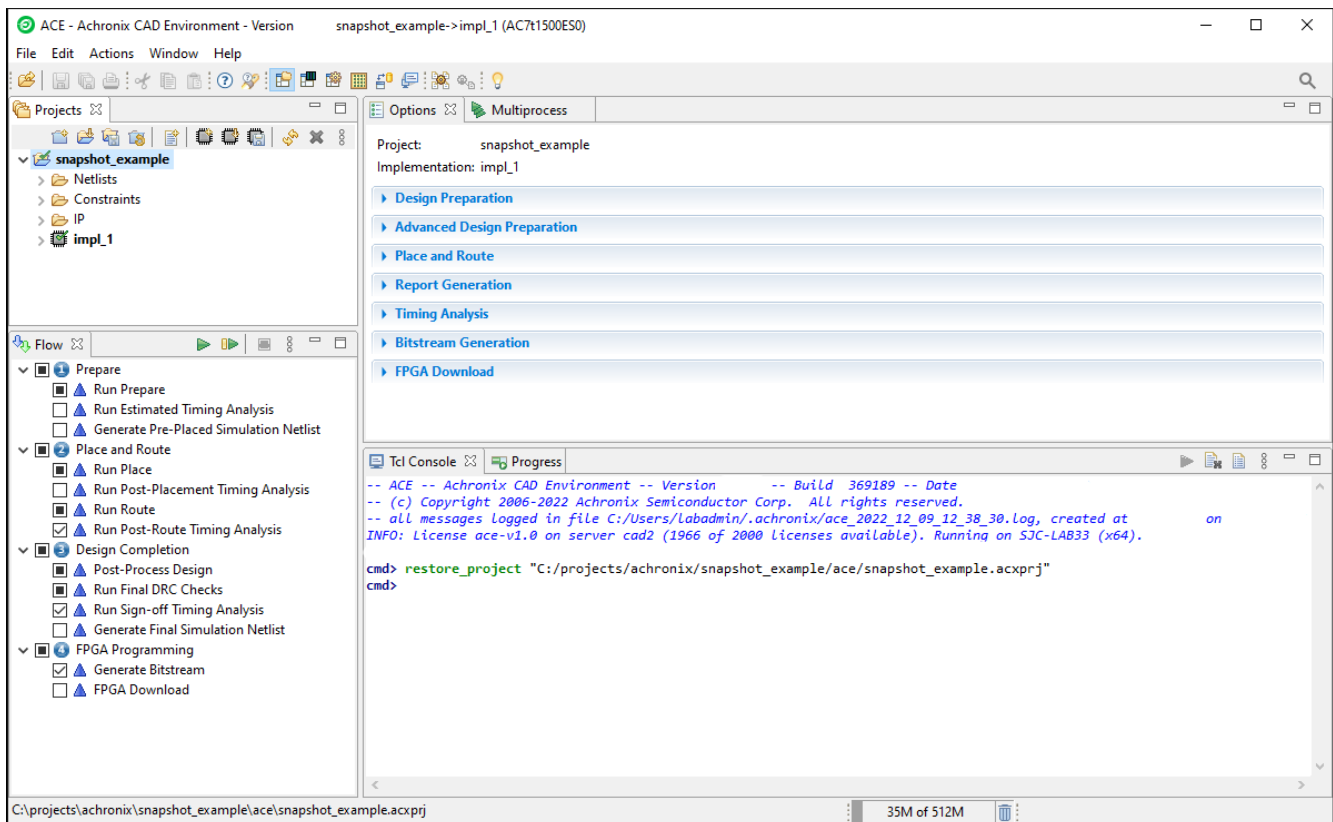




Figure 157 • ACE Tool Load Project

Open the Snapshot Debugger

Click the toolbar button to change to the  Programming and Debug Perspective as described in the [Working with Perspectives \(page 286\)](#) section. The [Snapshot Debugger view \(page 137\)](#) should be visible by default, as shown below. If not, select **Window** → **Show View** → **Snapshot Debugger** from the main menu bar.

The [Snapshot Debugger view \(page 137\)](#) should have automatically loaded the default Snapshot configuration file for the project, generated when the design ran through place and route, located in `<ace_project_dir>/<active_impl_dir>/output/names.snapshot`. If the file loaded, the correct signal names from the user

design appear in the **Trigger Channels**, **Monitor Channels**, and **Stimuli** tables. If the file did not automatically load, click the  **Load Snapshot Configuration** toolbar button in the **Snapshot Debugger view** (page 137) to browse to the location of the preferred *.snapshot configuration file, or manually enter the signal names, channel widths, etc. to match the design.

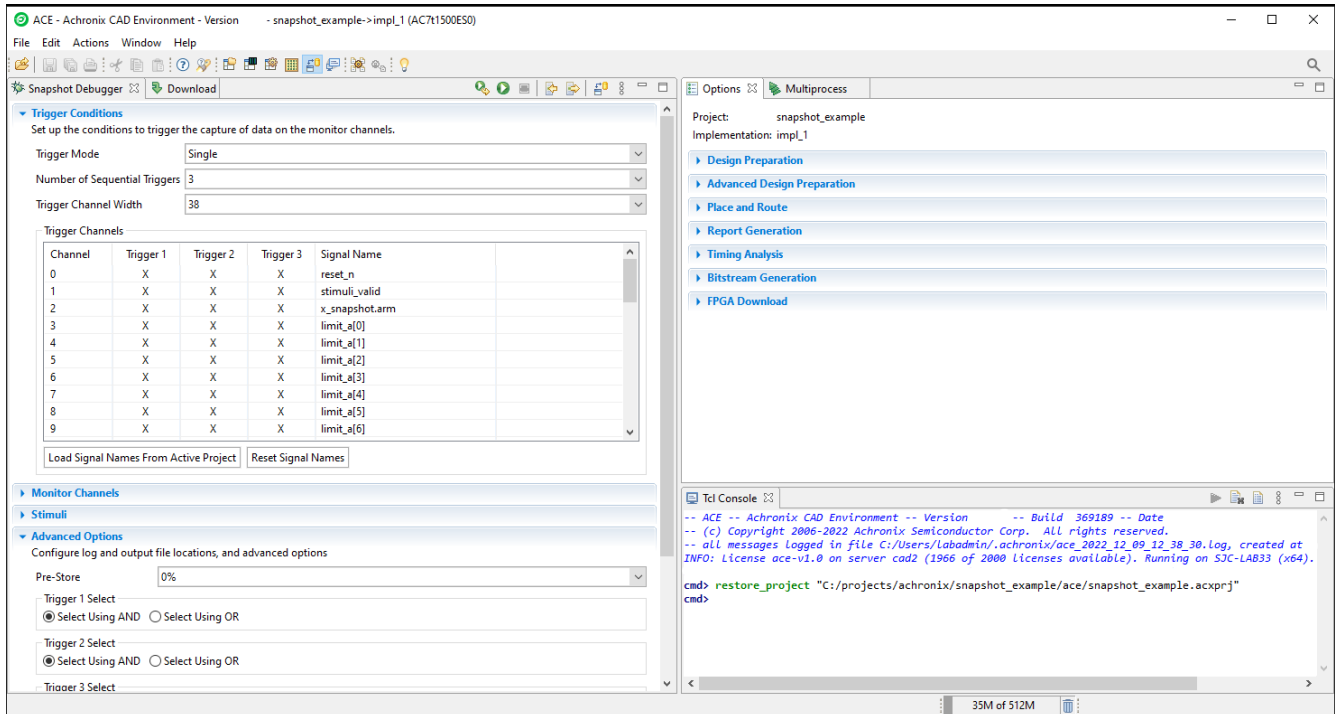


Figure 158 • Snapshot Debugger View

Configuring the Trigger Pattern

Note

The Trigger Channel signal names are automatically configured to the correct values when the names.snapshot file is loaded. The names.snapshot file is generated during design preparation (the **Run Prepare Flow Step** (page 234)), which contains the user design signal names connected to Snapshot, along with the trigger width and the maximum number of sequential triggers.

Configuring the Trigger Mode

The **Trigger Mode** option allows the user to select the trigger mode to use when the Arm action is run.

Single

The default trigger mode is **Single**, which means the trigger conditions are programmed in to the ACX_SNAPSHOT macro and then the GUI waits for a single trigger event to occur which matches those trigger conditions, and then a single VCD file is recorded. This option arms Snapshot and captures data only once.

Immediate

If **Immediate** trigger mode is selected, pressing the Arm button results in the same behavior as **Single** trigger mode, except that all 3 trigger patterns are treated as "Don't Care" (X's) so that the trigger event will occur as soon as the Arm button is pressed. This mode is useful to quickly capture the state of the running design without waiting for any trigger pattern to be met.

Repetitive

If **Repetitive** trigger mode is selected, the trigger conditions are programmed in to the ACX_SNAPSHOT macro and samples are captured repetitively until the upper limit of trigger event records is reached. When **Repetitive** trigger mode is selected, an additional set of repetitive trigger mode options will appear to allow the user to configure the number of sequential times Snapshot should be armed repetitively using the configured trigger conditions, and the way in which the output VCD files are managed. This mode is useful when the trigger conditions do not narrow in on the exact data pattern and the pattern you intend to observe occurs sporadically at the trigger conditions. You can let the repetitive trigger mode run for a long period of time, taking several capture records at the trigger conditions, to help find the pattern you are interested in. The user can optionally cancel the remaining Snapshot session once the desired data is captured.

The repetitive trigger Record Limit setting determines how many times (number of records) the GUI will repeatedly Arm the Snapshot debugger and capture samples. The user may set this to automatically run Snapshot up to 128 times.

The repetitive trigger VCD Record Limit setting determines how many Snapshot records to capture in a single VCD file. This essentially concatenates the VCD files from consecutive runs of Snapshot (records) into a single VCD file. The VCD file waveform contains a set of virtual signals to indicate the system timestamp at which each Snapshot record was captured. The user may concatenate up to 10 Snapshot records in a single VCD file.

If the Overwrite VCD File option is selected, the VCD Waveform File name specified in the Advanced Options section will be used to store the output VCD file. The file will be overwritten with the new VCD file each time the VCD record limit is reached. If the Overwrite VCD File option is not selected, then multiple VCD files will be written out and a unique VCD record number will be added to the VCD Waveform File name specified in the Advanced Options section for each VCD. For example, if you set the Record Limit to 8 and set the VCD Record Limit to 2, and set the VCD Waveform file path the `./snapshot.vcd`, then Snapshot would output 4 VCD files to `./snapshot1.vcd`, `./snapshot2.vcd`, `./snapshot3.vcd`, `./snapshot4.vcd`, each containing 2 Snapshot capture records.

Configuring Trigger Patterns

The Snapshot Debugger can be configured to use a **Trigger Channel Width** of 1 to 40 bits. The value entered in the Snapshot Debugger View must match the value of the `TRIGGER_WIDTH` parameter set on the ACX_SNAPSHOT module in the user design RTL. (This will be the width of the `i_trigger` bus.)

The Snapshot Debugger is capable of handling one to three sequential trigger patterns. The post-trigger data is sampled once the last trigger pattern in the sequence is matched.

The user may specify the number of desired sequential trigger patterns using the **Number of Sequential Triggers** option in the **Snapshot Debugger View** (page 137). If **1** is selected, Trigger 2 and Trigger 3 are ignored. If **2** is selected,

Trigger 3 is ignored and Snapshot will trigger when Trigger 1 is matched, followed (on any subsequent clock) by a match on Trigger 2. If **3** is selected, then Snapshot will trigger after a match on Trigger 1, followed (on any subsequent clock) by a match on Trigger2, followed (on any subsequent clock) by a match on Trigger3.

Each sequential trigger is hooked up to the trigger channels on the Snapshot Debugger core. The LSB of the trigger pattern is hooked to trigger channel 0, and the MSB is hooked to upper most trigger channel bit (TRIGGER_WIDTH - 1).

Each sequential trigger is made up of three parts: the pattern mask, the edge mask, and the don't care mask. In the Snapshot Debugger View, these 3 masks are combined for ease of use into a single trigger pattern value, which allows each bit to be specified as **X** (don't care), **R** (rising edge), **F** (falling edge), **0** (level 0), or **1** (level 1). The trigger pattern defines the trigger channel signal conditions that are required to detect a match. If a given trigger channel value is set to X (don't care), then this trigger channel is ignored when computing a match. If a given trigger channel value is set to R (rising edge), then this trigger channel is evaluated as a match when a rising edge of this signal is seen by Snapshot. If a given trigger channel value is set to F (falling edge), then this trigger channel is evaluated as a match when a falling edge of this signal is seen by Snapshot. If a given trigger channel value is set to 1 (level 1), then this trigger channel is evaluated as a match as long as this signal's level is seen as a 1 by Snapshot (it is not edge sensitive). If a given trigger channel value is set to 0 (level 0), then this trigger channel is evaluated as a match as long as this signal's level is seen as a 0 by Snapshot (it is not edge sensitive).

 **Warning!**

If any active Trigger is configured with as all X's (don't care), the trigger pattern will be a match on the first clock cycle that trigger is evaluated.

The values within a trigger pattern may cause a trigger match event either by AND'ing or OR'ing. If AND'ing, then **all** signal values not masked (set to X) must match their pattern for the trigger match event to occur. If OR'ing, then the trigger match event will occur if **any** of the non-masked (not set to X) signal values match the specified pattern. The AND/OR configuration is set per sequential trigger using the **Select using AND** or **Select using OR** radio buttons. This selection can be different for each sequential trigger.

In the "Trigger Channels" table of the Snapshot Debugger View, the trigger patterns can be viewed and edited.

Setting Pattern Values Using the Table

For each channel, a value of **X** (don't care), **R** (rising edge), **F** (falling edge), **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under each "Trigger" column as shown below.

Trigger Channels

Channel	Trigger 1	Trigger 2	Trigger 3	Signal Name
0	X	X	X	reset_n
1	X	X	X	stimuli_valid
2	X	X	X	arm
3	X	X	X	limit_a[0]
4	X	X	X	limit_a[1]
5	0	X	X	limit_a[2]
6	1	X	X	limit_a[3]
7	R	X	X	limit_a[4]
8	F	X	X	limit_a[5]
9	X	X	X	limit_a[6]

Load Signal Names From Active Project Reset Signal Names

Figure 159 • Trigger Channels Setting Example

Setting Multiple Pattern Values as a Bus

The Assign Bussed Values Dialog wizard allows assigning a value to multiple signals from the **Snapshot Debugger view** (page 137) "Trigger Channels" or "Stimuli Channels" tables as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all the selected signals in the **Snapshot Debugger View** (page 137). There are 2 ways to launch this dialog to allow bus assignment of values:

1. With your mouse, left click to select a single row in the **Snapshot Debugger View** (page 137) table which has a bussed signal name (i.e. din[2]). Then right mouse click to edit the **Value by Bus**. This method will automatically find all the other bits in the bus with the same signal name (i.e. din[0], din[1], din[2], etc.) and open the dialog to allow editing of the entire bus of signals.
2. With your mouse, hold CTRL or SHIFT and left click to select multiple rows in the **Snapshot Debugger View** (page 137) table. Then right mouse click to edit the **Value by Selection**. This method will open the dialog to allow editing of all selected signals as a bussed value.

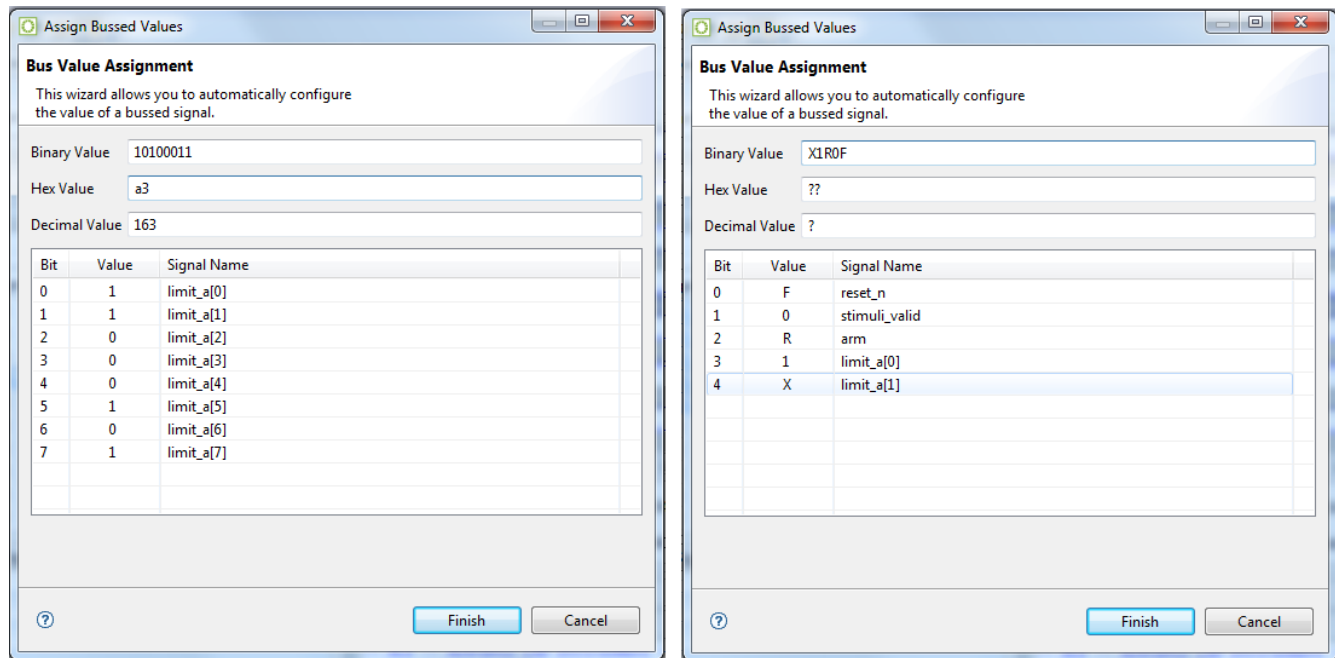


Figure 160 - Assign Bussed Values Dialog Example

See [Assign Bussed Values Dialog \(page 153\)](#) for more information on this dialog.

Configuring the Monitor Signals

Note


The Monitor Signals are automatically configured to the correct values when the names . snapshot file is loaded. The file is generated during design preparation (the **Run Prepare flow Step (page 234)**) which contains the user design signal names connected to Snapshot, along with the monitor width and number of samples.

The value of **Monitor Channel Width** in the [SnapShot Debugger view \(page 137\)](#) must be configured to match the value of the MONITOR_WIDTH parameter of the ACX_SNAPSHOT instance inside the RTL of the design being debugged (this is the width of the i_monitor bus).

The value of **Number of Samples** in the [SnapShot Debugger view \(page 137\)](#) should be configured to match the value of the MONITOR_DEPTH parameter of the ACX_SNAPSHOT instance inside the RTL of the design being debugged. If the value in the GUI does not match the value in the RTL, the value from the RTL is used and a warning is entered into the Snapshot log file.

Naming Captured Signal Data

Custom signal names for each channel can be entered under the **Signal Name** heading within the "Monitor Channels" table. The signal/bus names in the table are then used as labels on the captured signal data in the VCD waveform output, and are visible in the [VCD Waveform Editor \(page 11\)](#).

Multiple signals can be combined into a bus by selecting multiple rows in the "Monitor Channels" table, right-clicking a selected signal row to bring up a popup context menu, and selecting () **Assign Bus Name** from the context menu to bring up the **Assign Bussed Signal Names dialog** (page 151). After configuring the bus in the dialog, the bus name and indices are propagated to all the previously-selected signals.

To select a contiguous range of rows:

1. Select the first signal.
2. hold the Shift key and select the last signal.

To select a non-contiguous set of rows:

1. Select the first signal.
2. While holding down the Ctrl key, select the other signals.


Signal names may be returned to their defaults by clicking the **Reset Signal Names** button under the "Monitor Channels" table.

Note

Reset Signal Names resets all signal names in the table at once, not just the currently selected rows/signals.

The **Load Signal Names From Active Project** button loads the `names.snapshot` file generated during design preparation (the **Run Prepare flow step** (page 234)) which renames all signals with their project-specific names, and also loads the project-specific default settings for monitor width, user clock frequency, default `.log` and `.vcd` file path, etc.

Configuring Test Stimulus

 The stimuli channel signal names are automatically configured to the correct values when the `names.snapshot` file is loaded. The `names.snapshot` file is generated during design preparation (the **Run Prepare Flow Step** (page 234)), which contains the user design signal names connected to Snapshot, along with the stimuli width.

Snapshot has the capability to send 0 to 512 bits of test stimuli (the `ACX_SNAPSHOT` macro output signal `o_stimuli`) to the Design Under Test (DUT). This data is sent once per arming session, is only valid while the `o_stimuli_valid` signal is high.

This `o_stimuli` output is optional, and need not be connected to the DUT — it may safely be left floating when Snapshot is used to only read signals.

The value of **Stimuli Channel Width** in the **SnapShot Debugger view** (page 137) must be configured to match the value of the `STIMULI_WIDTH` parameter of the `ACX_SNAPSHOT` instance inside the RTL of the design being debugged (this is the width of the `o_stimuli` bus).

In the **Stimuli Channels** table of the Snapshot Debugger View, the stimuli values can be viewed and edited.

Setting Stimuli Values Using the Table

For each channel, an output value of **0** (level 0), or **1** (level 1) can be specified via a pull-down menu under the **Value** column as shown.

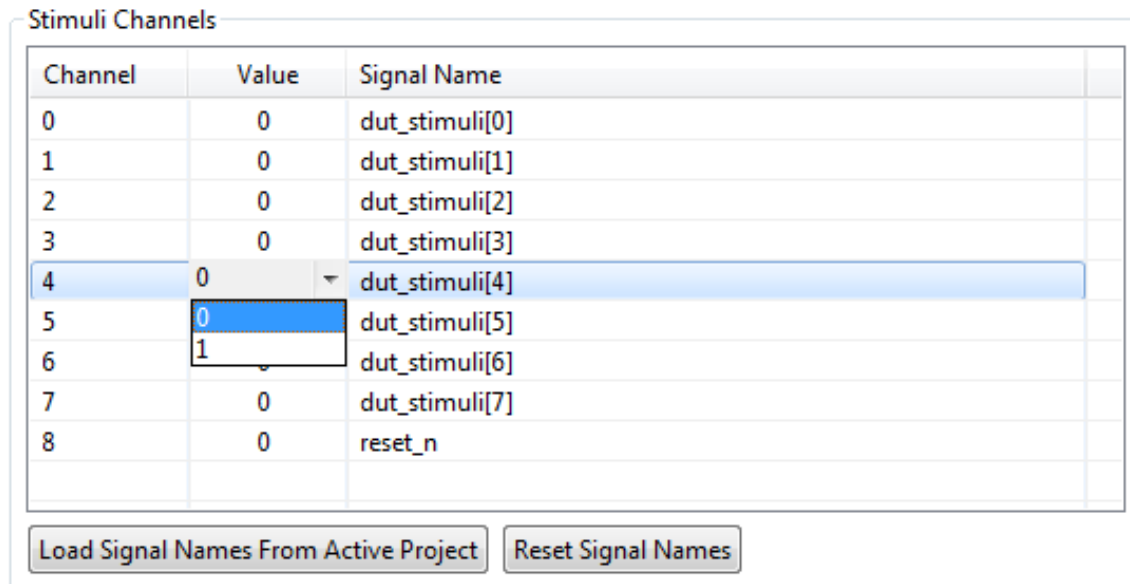


Figure 161 • Stimuli Channels Value Setting Example

Setting Multiple Stimuli Values as a Bus

The Assign Bussed Values Dialog wizard allows assigning a value to multiple signals from the **SnapShot Debugger view** (page 137) **Stimuli Channels** table as a bus. After configuring the bus in the dialog, the values of each signal are propagated to all the selected signals in the **SnapShot Debugger View** (page 137). There are two ways to launch this dialog to allow bus assignment of values:

1. Left click to select a single row in the **SnapShot Debugger View** (page 137) table which has a bussed signal name (i.e., `din[2]`).
Right click to edit the **Value by Bus**. This method automatically finds all other bits in the bus with the same signal name (i.e., `din[0]`, `din[1]`, `din[2]`, etc.) and opens the dialog to allow editing of the entire bus of signals.
2. Hold **CTRL** or **SHIFT** and left click to select multiple rows in the **SnapShot Debugger View** (page 137) table.
Right click to edit the **Value by Selection**. This method opens the dialog to allow editing of all selected signals as a bussed value.

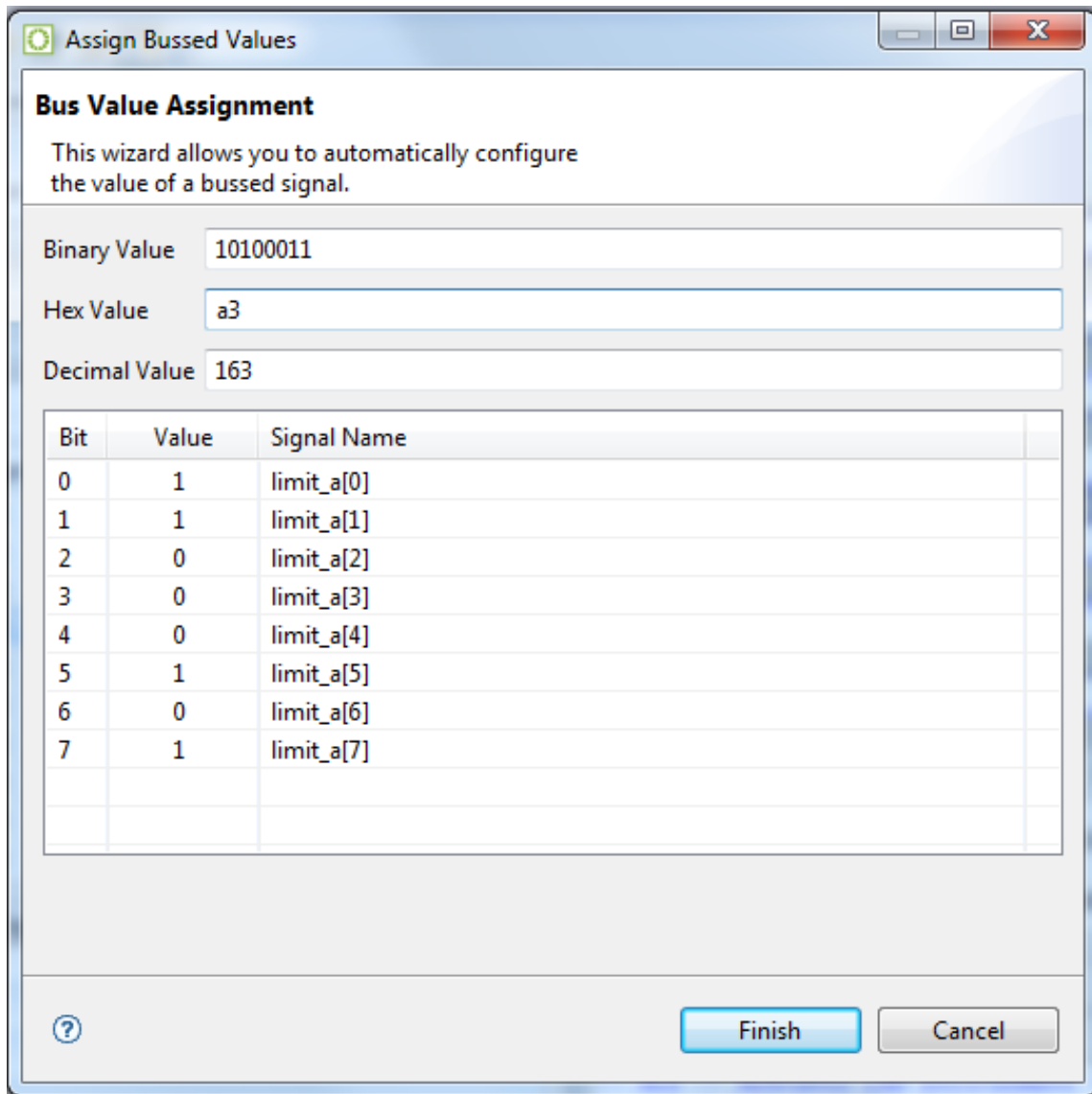


Figure 162 • Assigned Bus Values Dialog Wizard Example

See [Assign Bussed Values Dialog](#) (page 153) for more information on this dialog.

Configuring Advanced Options

Pre-Store

In the [Snapshot Debugger View](#) (page 137), the **Pre-Store** setting configures the portion of samples that are collected before the trigger, and (indirectly) how many are collected after the trigger.

For example, assume that Snapshot is configured to use a monitor depth of 1024 samples. See the table below:

Table 144 · Effect of "Pre-store" on samples collected before and after the trigger event

"Pre-Store" value	Samples collected before trigger	Samples collected after trigger
0%	0	1024
25%	256	768
50%	512	512
75%	768	256

When a **Pre-Store** value other than **0%** is selected, the `.vcd` file contains a signal `snapshot_pre_store` that transitions (goes low) at the point where the (last sequential) trigger event occurred. Thus, the trigger event may easily be found without needing to actually count the samples.

Trigger Pattern Match Behavior

The values within a trigger pattern may cause a trigger match event either by ANDing or ORing. If ANDing, then *all* signal values not masked (set to X) must match their pattern for the trigger match event to occur. If OR'ing, the trigger match event occurs if *any* of the non-masked (not set to X) signal values match the specified pattern. The AND/OR configuration is set per sequential trigger using the **Select using AND** or **Select using OR** radio buttons. This selection can be different for each sequential trigger.

User Clock Frequency

The **Frequency** field must be configured to match the `user_clk` frequency in the target user design, which typically matches the timing constraint set in the SDC file of the design being debugged. The value from the user design SDC file is set automatically in the `names.snapshot` file when an active implementation is available. The frequency value entered in the Snapshot GUI (or `.snapshot` configuration file) determines the time (in picoseconds) for all signals shown in the captured VCD file. All samples are captured at the rising edge of the Snapshot `user_clk` signal.

Configure Output File Locations

The final Snapshot configuration steps specify the locations of the output files which contain the log messages and sample data collected by Snapshot.

File Paths Relative To Chooses whether the **Log File** and **Waveform File** paths are understood to be relative to the **Active Project** directory or to the **Working Directory**. (This setting only matters when the Log and Waveform file paths provided are relative paths, and not absolute paths). When the **Working Directory** option is chosen, keep in mind that the working directory will be the path returned by the Tcl command `pwd`. (See <https://www.tcl.tk/man/tcl8.6/TclCmd/pwd.html>)

Log File configures the file name and path for the log file generated by the Snapshot Debugger run. The associated **Browse** button provides a directory/file selection dialog for the selection of a location different than the default (the



default is `<active_impl_dir>/log/snapshot.log`, or if there is no [Active Project and Implementation](#) (page 229), `<user_home>/snapshot.log`. If an error occurs during setup or while reading back the sample information, the Snapshot log file contains the error messages.

Waveform File configures the file name and path for storing downloaded sample waveform information from the SnapShot Debugger core in VCD format. The **Browse** button allows for the selection of a location different than the default (the default is `<active_impl_dir>/output/snapshot.vcd`, or if there is no active implementation, `<user_home>/snapshot.vcd`).

Collecting Samples of the User Design



Caution!

The JTAG connection must be configured before collecting Snapshot samples!

ACE interacts with the FPGA or eFPGA using the JTAG interface through a USB Bitporter2 pod or FTDI FT2232H device. This JTAG interface must be properly configured in ACE before using the [Snapshot Debugger View](#) (page 137). The configuration is managed using the [Configure JTAG Connection Preference Page](#) (page 190), which is easily accessible by pressing the  **Configure JTAG Interface** button in the upper-right of the view. See [Configuring the JTAG Connection](#) (page 360) for more details. The  **Configure JTAG Interface** button provides a summary of the current JTAG configuration settings (for pod name and scan chain) in its tooltips for ease of reference.



Starting in ACE 9.2, Snapshot uses Tcl JTAG commands instead of STAPL commands for all JTAG interactions (see [JTAG Configuration User Guide](#) (UG004) and [Snapshot User Guide](#) (UG016) for details). If taking advantage of the Tcl JTAG libraries, this slightly complicates things, because now it must be ensured that the JTAG device is not connected when starting to use Snapshot (at present, using the Snapshot Debugger view requires sole ownership of the JTAG device connection, and errors are reported if Snapshot fails to open the device connection on its own. Achronix plans to relax this limitation in a future release of ACE, ensuring that Snapshot can share an already-open JTAG connection with others). What this means is that `jtag::close` must be called on the connection (if already open) before Snapshot can use the connection.

Caution! The JTAG connection must be closed before collecting Snapshot samples!

Snapshot must be the sole owner of the Tcl JTAG connection to the JTAG device for the lifetime of the polling/collection. This means `jtag::close` must be called (if the Tcl JTAG connection was previously opened) before starting the  **Arm** or  **Capture Startup Trigger** actions. If the Tcl JTAG connection to the JTAG device name remains open when either action is started, the attempted action fails, and an error message is placed in the `.log` file reporting that Snapshot was unable to open the named JTAG connection.

Arming the Snapshot Debugger


When all the fields in the [Snapshot Debugger view](#) (page 137) are configured, and the design is running on the target device, Snapshot is ready to be armed.

Select the  **Arm** button (or the  **Arm Snapshot** button in the SnapShot Debugger view toolbar), and the ACE Snapshot Debugger sends the configuration data (including the optional stimulus) to the `ACX_SNAPSHOT`

(or `ACX_SNAPSHOT_JTAG_UNIT`) circuit running on the Achronix device, waits for the trigger condition(s) to be met, retrieves the trace buffer contents, and outputs a VCD file as well as a LOG file.

When Armed, Snapshot begins to analyze the already-executing design in real-time.

The Snapshot log file and Snapshot waveform file are populated with the captured results, and the files are opened in ACE (the log file opens in an ACE **text editor** (page 10), while the waveform (.vcd) file opens in the ACE **VCD waveform editor** (page 11)). If an error occurs during Snapshot Debugger configuration or while reading back the sampled information (trace buffer), the Snapshot log file contains the relevant error messages, and the Snapshot waveform file is not created/updated.

The () **Cancel** button aborts the Snapshot arming process. The Snapshot log file is updated, but the Snapshot waveform file is not created/updated if the cancel button is clicked. Cancel is useful if accidentally sending in trigger conditions that are never matched.

If using **Repetitive** trigger mode, Snapshot repetitively executes the arm action for the number of records specified, or until the cancel button is clicked. See **Configuring the Trigger Pattern** (page 367) for details on the Repetitive Trigger feature.

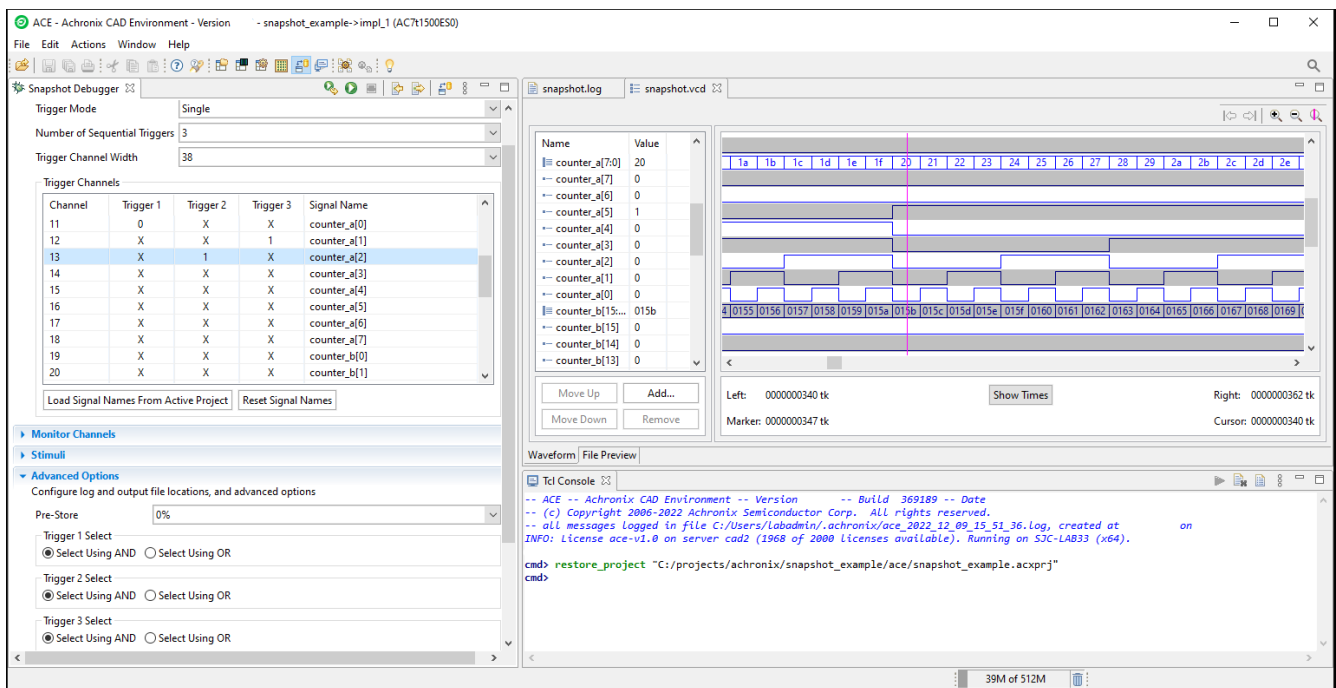



Figure 163 - Snapshot Debugger Arming Example

Using the Startup Trigger

The Startup Trigger feature is a very special case of Snapshot sample collection, intended specifically to fill the trace buffer based upon captured trigger events that happen very soon after the Achronix FPGA enters user mode. When Snapshot is configured to use the Startup Trigger feature, unlike typical interactive Arming, this means the macro automatically arms as soon as the FPGA enters user mode, so that trigger analysis (and potentially sample collection) can begin immediately. When the () **Capture Startup Trigger** action is selected, ACE does not arm the

Snapshot circuit (and does not send any stimuli), but simply starts polling the Snapshot circuit already running on the Achronix device, waits for the trigger conditions to be met (if not already met prior to the button being pressed), and retrieves the populated trace buffer contents. It is not unusual for the trace buffer to already be filled, ready and waiting (because the trigger conditions had already been met), when the user starts the (👉) **Capture Startup Trigger** action in ACE. After the trace buffer data has been retrieved, Snapshot creates the VCD file based on the collected trace buffer contents. Of course, a Snapshot .log file (with the name and path chosen in the **Advanced Options** section in the **Log File** field) is also always populated to provide additional success/failure details, and is opened in the ACE GUI when the action is complete, along with the VCD file.

Be aware that before the Startup Trigger functionality can be used in the Snapshot Debugger view, the Startup Trigger feature must have already been specifically enabled using the initial startup trigger parameters on the `ACX_SNAPSHOT` (or `ACX_SNAPSHOT_JTAG_UNIT`) macro, which includes the needed circuit configuration in the bitstream used to program the Achronix FPGA/eFPGA (see the *Snapshot User Guide* (UG016) for details). This of course means that any time the macro parameters are changed, the bitstream needs to be regenerated and the Achronix device need to be reprogrammed.

Because the Startup Trigger functionality can only be used at chip startup, the FPGA needs to be reset and reprogrammed between Startup Trigger attempts.

Be aware that as soon as the (👉) **Arm** action is triggered in any session where the Startup Trigger function was configured in the Snapshot circuit, (before the FPGA has been reset/reprogrammed) the startup trigger conditions and any pre-existing trace buffer contents are cleared when the interactive Arm action re-initializes the Snapshot circuit.

Viewing the Captured VCD Waveform

ACE includes a built-in simple VCD waveform viewer, called the **VCD Waveform Editor** (page 11), which is used whenever any *.vcd file is opened within ACE. This is typically only used when viewing VCD files generated by Snapshot, typically from the **Snapshot Debugger View** (page 137) when **Running the Snapshot Debugger** (page 364) or after using the `run_snapshot` (page 700) Tcl command in ACE batch mode.



If the **VCD Waveform Editor** (page 11) has insufficient functionality for some use cases, be aware that the *.vcd file created by Snapshot uses the industry standard format, thus any third party VCD viewer can also be used to examine the ACE-generated VCD files.

The VCD waveform viewer does not allow editing the content of a *.vcd file, but it does allow altering which of the signals captured within the VCD are displayed, to change the order of the displayed signals, and to compare/contrast the captured values graphically as waveforms.

The example screenshot below shows an example dataset from a Snapshot session where relatively few signals were captured, but with a higher sample count. Three signal names are visibly selected in the name/value table (the rows in the table with a grey background), with their waveforms displayed in bold orange. The marker line is also visible in the waveform area, in pink, with the values of each of the signals at that exact marker time displayed in the table on the left (in the **Value** column). Bussed signals also have their values displayed in the table in hex format, and bus values are also displayed in the waveform area (when the visible area between bus edges is wide enough to fit the bus value), again in hex format. Buttons can be seen above the table on the left to operate on the table contents as a whole. Buttons can be seen below the table on the left to change the table content in a more granular manner. Buttons can be seen in the upper-right above the waveform to scroll and zoom the content of the waveform area. Below the waveform area are live feedback labels describing the tick (or time) counts for the onscreen area, the

marker position, and the mouse cursor position. At the very bottom left of the screenshot, tabs can be shown that allow switching between the **Waveform** view of the file data, and the **File Preview** (raw textual) view of the VCD file.

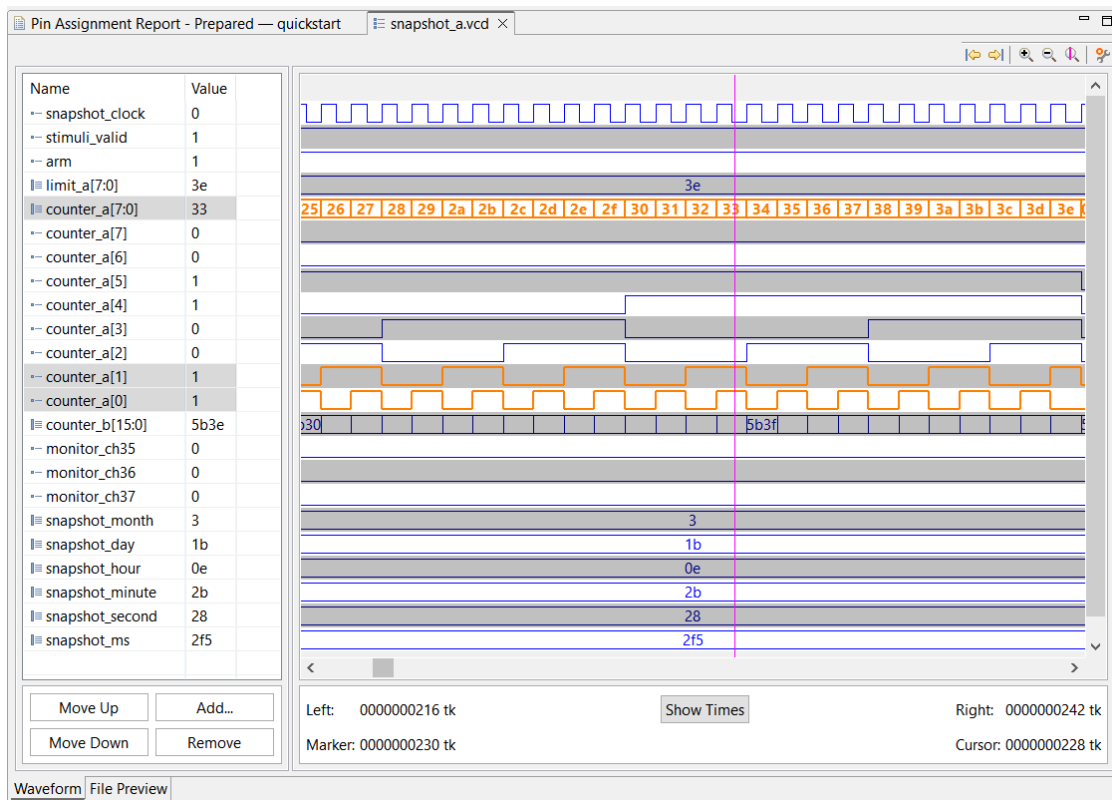


Figure 164 - VCD Waveform Editor Example

The **VCD Waveform Editor** (page 11) page describes each of these features at a high level, while the following sections describe in more detail how each feature may be utilized.

Opening a VCD file

When using the **Snapshot Debugger View** (page 137), Snapshot automatically opens any VCD files as soon as they are created (or updated) during a Snapshot session.

When ACE starts, ACE also automatically re-opens any VCD files that were previously open the last time ACE was closed.

There are also several ways to manually open previously-generated VCD files:

- With the drop-down application menu choice **File** → (📁) **Open Files** and/or the (📁) **Open Files** button (in the upper left, in the application button bar), arbitrary files can be browsed to and opened graphically, including VCD files.

- If a Snapshot VCD file was previously output to one sub-directory of an implementation (for example, `output` or `reports`), the file can be opened by selecting it from the corresponding file listings in the **Projects View** (page 117).
- The Tcl command `display_file` (page 627) can be used (from the **Tcl Console View** (page 142)) to open VCD files (or any other files) in ACE.

Re-Opening the Same Filename Restores Prior State

ACE includes the ability to remember many details, by filename/path, of the chosen configuration in the VCD editor. When a previously-used filename/path is re-opened, either manually or automatically, ACE tries to restore many details from the prior VCD viewing session, like the exact signals which were chosen to be visible, and their relative ordering in the signals table. Restoring the exact prior configuration is not always possible, of course, if certain details (like the signal names being captured) are changed between sessions (as may occur when VCD files are overwritten by a later capture).

Viewing the Raw Text Content

While ACE always defaults to showing the VCD file content as waveforms, the raw textual content of the file can always be viewed by selecting the **File Preview** tab in the lower-leftmost corner of the view.

There are no special features available when viewing the VCD file textual content in this manner.

Viewing the Signal Data as Waveforms

When opening a VCD file, ACE always defaults to showing the VCD file content as waveforms. If using the **File Preview** to view the raw text, user may always return to viewing the waveform data by selecting the **Waveform** tab in the lower-leftmost corner of the view.

There are a wide variety of ways that ACE allows altering which portion of the captured data is visible and in what order. These are mentioned in detail in the following sections.

What is Displayed by Default

When opening a VCD file for the first time, ACE defaults to showing every captured signal, and when encountering a bus signal, shows only the consolidated bus. The individual captured bits within the bus are available, but will be collapsed by default.

The order of the signals in the table (and accompanying waveforms) initially match the order of the signals as found in the VCD file.

The initial position of the waveform marker is the first moment of the trace, at tick 0. Thus, in the signals table, the values shown are also from tick 0. (The values column in the signals table always represents the value found for the signal at the time of the marker.)

Though ACE remembers for future sessions all changes regarding which signals are displayed, and in which order, the (🔧) **Reset Signal Table** button in the upper left above the signals table will forget prior setting and restore the displayed signals to defaults, as if the file were being loaded for the first time.

Adjusting the Table and Table Column Widths



The width of the entire signals table may be adjusted by dragging the separator between the table and the waveform to the right/left. If the table width is too narrow to display all the contained columns, a horizontal scrollbar allows scrolling the table content to the right or left.

As with most tables in ACE, the column widths within the signals table can be adjusted by dragging the vertical column separator (the vertical grey line between the column header titles) to the right/left to widen/narrow each column. When a column is too narrow to display its full content, the column content is truncated and an ellipsis suffix is appended. On some operating systems/desktop environments/themes, it is also possible to double-click any table column separator to cause the column (to the left of the separator) to automatically resize its width to accommodate its widest content for all the rows which are visible on screen. On some desktop configurations (and table widths), the right edge of the rightmost (value) column may not be easily distinguishable/visible; in these cases, scroll the entire table as far right as possible, then treat the right edge of the table as the rightmost column separator.

Panning/Scrolling the Table and Waveform Contents


The signals table can be scrolled vertically using the table vertical scrollbar or the waveform vertical scrollbar. Both the table and the waveform scroll vertically in lockstep, so that the signal name/value and the waveform for that signal are always aligned (as if the waveform was just another column in the table). The mouse scroll wheel can be used over the signals table (and the table vertical scrollbar) to scroll vertically. On some operating systems/desktop environments/themes, using the scroll wheel over the waveform area vertical scrollbar also scrolls the table and waveform area vertically (though on others, it changes the zoom level of the waveform area).


Scrolling horizontally, the signals table and waveform are not in lockstep, and are independent. Horizontal scrolling of each can be done clicking/dragging their horizontal scrollbars independently. On some operating systems/desktop environments/themes, using the mouse scroll wheel over the waveform area horizontal scrollbar also scrolls the waveform area horizontally (though on others, it changes the zoom level of the waveform area).

The () **Move marker to previous edge** and () **Move marker to next edge** buttons can also be used to move the marker, and the waveform is panned to center horizontally on the marker position. When a single row is selected in the table, the marker finds the previous/next edge for that signal. When multiple rows are selected in the table, the marker finds the previous/next edge detected for any of the signals at those rows in the table. (When there are no signals selected in the table, these buttons become disabled.)

The up/down arrow keys on the keyboard may also be used to change the selected table row vertically, and when the top/bottom of the table are reached, pan/scroll the table vertically (with the waveform area moving vertically in lockstep). After clicking the mouse in the waveform area to give the waveform area focus (which also moves the marker to the clicked position), the up/down/left/right arrow keys on the keyboard can also be used to pan/scroll the waveform area vertically or horizontally, with the signals table moving vertically in lockstep with the waveforms.

Changing the Waveform Zoom Level

The () **Zoom in** button (found in the upper-right of the waveform area) causes the horizontal space used to represent one tick (or unit of time) to get wider. The rescaled visible area is based around the center of the current visible area, not around the mouse position nor the marker position.

The  **Zoom out** button (found in the upper-right of the waveform area) causes the horizontal space used to represent one tick (or unit of time) to get wider. The rescaled visible area is based around the center of the current visible area, not around the mouse position nor the marker position.

Using the mouse vertical scroll wheel over the waveforms similarly changes the zoom level in the waveform area. Be aware that when using the mouse wheel, the mouse position is taken into account, so the tick/time under the mouse stays visible as the zoom changes. On some operating systems/desktop environments/themes, using the mouse scroll wheel over the waveform area's scrollbars may pan/scroll the waveform area instead of changing the zoom level.

There are also several keyboard shortcuts that may be used when/if the waveform area has application focus. (Click the mouse in the waveform area, including either waveform scrollbar, to move application focus so that the keyboard shortcuts work.) The "**Z**" and "**+**" (plus) keys zoom in based on the mouse cursor position, and the "**z**" and "**-**" (minus) keys zoom out, also based on the mouse cursor position.

Changing the Order of Displayed Signals

There are two ways to change the position of a single signal within the signals table:

- Select one (or more, as long as they are adjacent) row(s) in the table, then press the **Move Up** and **Move Down** buttons (found below the table) to move the selected rows up and down with respect to their neighbors.
- Select one row in the table, then vertically drag that row to the desired destination position in the table. Visible feedback is provided (a horizontal bar, bold and/or colored, the details vary based on the OS/desktop environment/theme) to indicate the drop/insertion point between the current rows of the table.

Multiple adjacent selected signals in the table can be moved together as a unit using the same actions. To select multiple rows, click the first row, press and hold the **Shift** key on the keyboard, then click and release the mouse button over the second row, causing all intervening rows to be selected at once. (Note that when using drag-and-drop, the **Shift** key on the keyboard must remain pressed during the drag-and-drop operation, and should not be released until the mouse button has already been released at the drop/insertion location.)

Be aware that contiguous bus bits are not allowed to be separated/split from their parent bus. When moving a bus, the bus and all its contiguous bits will move as a single unit. Similarly, when moving anything past a bus, the moved row(s) will move over the entire bus (and all contiguous bits) in one step.



Keep in mind that if a signal is going to be moved a significant distance, it may be faster to hide/remove the signal, and then re-add/insert the signal directly at the new desired location.

Adding VCD File Signals to the Display

With the Add button and the accompanying [Add Signals to Waveform Viewer Dialog \(page 146\)](#), one or more signals can be added and placed anywhere within the signals table.

To insert a signal at the top of the table:

1. Ensure that no rows of the table are selected. If necessary, deselect any selected rows by holding the keyboard **Ctrl** key while clicking a selected row.
2. Press the Add... button to bring up the [Add Signals to Waveform Viewer Dialog \(page 146\)](#),

3. Find and select the desired signal(s) in the list. (If your desired signals are not in the list of hidden signals, which is shown by default, first select the **Select from All Signals** mode in the dialog.)
To select more than one signal, press and hold the **Shift** key when clicking the mouse to select multiple adjacent signals. Press and hold the **Ctrl** key on the keyboard when clicking to select multiple non-adjacent signals.
4. Press the **Insert** button in the dialog to insert the chosen signals at the top of the table.

To append a signal at the bottom of the table:

1. (The selection state in the table does not matter when appending signals to the bottom of the table).
2. Press the Add... button to bring up the **Add Signals to Waveform Viewer Dialog** (page 146),
3. Find and select the desired signal(s) in the list. (If your desired signals are not in the list of hidden signals, which is shown by default, first select the **Select from All Signals** mode in the dialog.)
To select more than one signal, press and hold the **Shift** key when clicking the mouse to select multiple adjacent signals. Press and hold the **Ctrl** key on the keyboard when clicking to select multiple non-adjacent signals.
4. Press the **Append** button in the dialog to append the chosen signals to the bottom of the table.

To insert a signal in the middle of the table:

1. Select the one row immediately below the desired row where the signals should be inserted.
2. Press the Add... button to bring up the **Add Signals to Waveform Viewer Dialog** (page 146),
3. Find and select the desired signal(s) to be added from the list in the dialog. (If your desired signals are not in the list of hidden signals, which is shown by default, first select the **Select from All Signals** mode in the dialog.)
To select more than one signal, press and hold the **Shift** key when clicking the mouse to select multiple adjacent signals. Press and hold the **Ctrl** key on the keyboard when clicking to select multiple non-adjacent signals.
4. Press the **Insert** button in the dialog to insert the chosen signals before the row that was selected.
Recall that buses are not allowed to be split, so if the insertion point attempted (the selected row in the signals table) is a bus bit, the actual insertion will occur before the entire bus.

Duplicating Signals

It is often desirable to have a signal included in the table (and waveform area) multiple times to ease value comparisons. Simply choose your insertion point in the table as mentioned above, then use the **Add..** button to add another copy of the desired signal to the table. (Make sure the **Select from All Signals** mode in the **Add Signals to Waveform Viewer Dialog** (page 146) is chosen, because the signal is already displayed, and, thus, is not in the list of currently hidden signals.) If necessary, then use the **Move Up** and **Move Down** buttons (or drag-and-drop signals in the table using the mouse) to better align the copied signal in relation to its peers.

Removing/Hiding Displayed Signals

Signals can also be hidden/removed from the table (and simultaneously the waveform area).

Simply select the row(s) in the signals table containing the signal(s) to be hidden/removed, then press the **Remove** button below the table, or right-click and select **Remove** from the context menu. The chosen signal(s) are hidden. Keep in mind, this does not affect the content of the VCD file, and the signal(s) may still be revealed/added again at any time (and at any desired position in the table).




Examining Signal Values

While it is possible to see the values of all the signal traces by simply examining the waveform, proper use of the signals table in combination with the waveform area can make things much easier.

Bits Versus Buses

Icons are shown to the left of each signal name as an indicator whether the name corresponds to a individual bit signal, or a bus, or a bit within a bus.


Table 145 • Signal Name/Value Table Icons

Icon	Description
	Signal
	Bus
	Bus bit

In the table, the values (in the Values table column) of bit signals and bus bit signals are shown as a simple 0 or 1. The Values column for a bus row will be shown in (unsigned) hexadecimal format.


Additionally, bus hex values are displayed within the waveform area when there is sufficient horizontal space between edges of the value to contain the rendered value string. Zooming in on the waveform area increases the horizontal distance between bus edges, allowing wider bus values to become visible.




When dealing with wide buses, it might make sense to make the waveform font choice as small a point size as remains legible, allowing wider bus values to be shown in the same amount of horizontal space. Use the () **Configure colors...** button (in the upper-right above the waveforms) to quickly access these color and font preferences.

Expanding and Collapsing Buses and Bus Bits

Buses will be shown collapsed by default, with none of the individual bits of the bus being visible.

Buses can optionally be expanded to expose the individual bits. All bits of all buses can be expanded at once by using the () **Expand All Buses** button, in the upper left above the signals table. A single bus may be expanded by selecting the bus row in the table, then right clicking the mouse on that row, and selecting **Expand Bus** from the context menu.

After a bus has been expanded to show its bus bits, it can also be collapsed again. All bits of all buses can be collapsed at once by using the () **Collapse All Buses** button, in the upper left above the signals table. A single bus may be collapsed by selecting the bus row or the row for any of its bits in the table, then right clicking the mouse on that row, and selecting **Collapse Bus** from the context menu.

Highlighting Signal Rows

When rows are selected in the signal table, the corresponding waveforms are rendered with a special selection foreground color (orange by default). Optionally, using font preference settings (see below), selected bus values in the waveform area may also be rendered in a user-selected font to make them stand out even more, if the selection foreground color change is insufficient.

Multiple rows in the table may be selected (and thus highlighted in the waveforms) simultaneously by holding the CTRL key on the keyboard while clicking on each desired signal row in the table.

Understanding How the Waveform Area Represents Time

The Snapshot signal traces are captured with respect to both a snapshot clock (ticks, abbreviated "tk") and an elapsed time (shown in s, ms, us, ns, ps, or fs, whichever is most appropriate). Ticks and elapsed time may be toggled between by pressing the button centered below the waveform area (which alternates between the labels **Show Ticks** and **Show Times** each time it gets pressed).

There are four labels below the waveform area that show the times for various portions of the interface:

Label	Description
Left	The tick or elapsed time value at the leftmost visible part of the waveform area.
Right	The tick or elapsed time value at the rightmost visible part of the waveform area.
Marker	The tick or elapsed time value at the marker bar, which may be offscreen, and may thus have a value less than the left value or greater than the right value.
Cursor	The tick or elapsed time value at the cursor horizontal position.

Note about Cursor values

When the mouse moves away from the waveform area, the last position over the waveform is retained by the **Cursor** value. This ps or tk value does not change until the mouse cursor is again over the waveform, even if the view is scrolled or the zoom factor is changed.

Using the Marker

The waveform area includes support for a user-managed marker, represented as a pink colored vertical bar by default, to allow examining signal values at chosen points in time/ticks.

When the marker is moved, the **Values** column in the signals table are immediately updated to contain the values of all signals at the point in time/ticks represented by the marker location. Additionally, the value of the "marker" label below the waveform area is updated to show the exact time/tick value of the marker.

Targeted Marker Placement

Clicking the primary mouse button anywhere within the waveform area moves the vertical marker bar to the horizontal position that was clicked. (When zoomed far enough out, the marker may even be placed after the trace has ended.)

The "m" (for marker) and "s" (for select) keyboard shortcuts also causes the marker to be placed under the current cursor position. (If the cursor is over the signals table, the marker is moved to the left edge of the visible waveform area, such that the tick/time value shown for the "Left" and "Marker" values below the waveform area matches.)

Using the Marker to Find the Next or Previous Signal Edge

The (←) **Move marker to previous edge** and (→) **Move marker to next edge** buttons (found in the upper-right of the waveform area) can also be used to move the marker from its current position to a new signal edge position, and the waveform is panned to center horizontally on the marker position. When a single signal is selected in the table, the marker finds the previous/next edge for that single signal. When multiple signals are selected in the table, the marker finds the previous/next edge detected for any of the signals selected in the table. (When there are no signals selected in the table, these buttons become disabled.)

Editing Waveform Color and Font Preferences

All of the colors and fonts used within the waveform area (as well as the font used in the table) are allowed to be adjusted as [Preferences \(page 188\)](#). These can be reached quickly by pressing the (⚙️) **Configure colors...** button in the upper-right of the waveform area, or by navigating to **Window** → **Preferences** → **General** → **Appearance** → **Colors and Fonts** → **VCD Editor**.

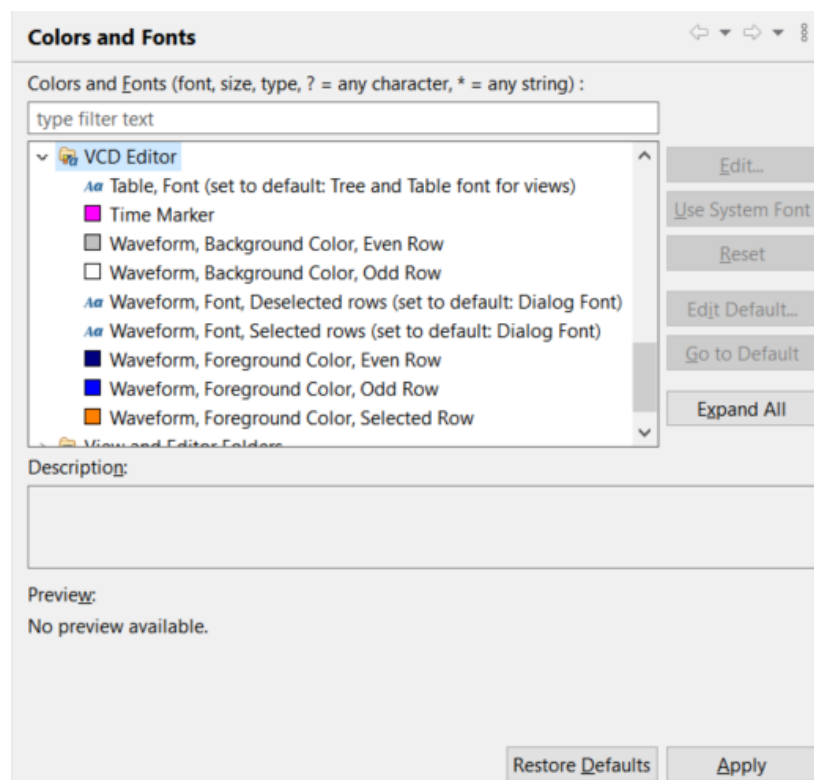



Figure 165 • Default Colors and Fonts Used by the VCD Editor Area


These preferences can be altered to, for example, provide a green-on-black color scheme, as would be seen on a traditional lab scope. Some prefer to have a bold font for the selected signals (for hexadecimal bus values). An

example of this sort of color and font scheme is visible in the screenshot at the end of the [VCD Waveform Editor \(page 11\)](#) description.

Saving/Loading Snapshot Configurations

An existing known-good Snapshot configuration (the collection of settings in the [Snapshot Debugger View \(page 137\)](#)) may be re-used at a later date, or in batch mode.

Snapshot configurations may be saved to a Snapshot configuration file (with the `.snapshot` file extension) using the () **Save SnapShot Configuration** button found in the [Snapshot Debugger view \(page 137\)](#) toolbar.

These Snapshot configurations may then be loaded later by using the () **Load SnapShot Configuration** button, found in the [Snapshot Debugger view \(page 137\)](#) toolbar.

Note

Previously saved Snapshot configuration files are necessary to run [Snapshot in Batch mode \(page 387\)](#).

Tip

When a user design containing the `ACX_SNAPSHOT` macro completes the [flow step \(page 234\)](#) **Run Prepare**, a `names.snapshot` configuration file is automatically generated. This file contains harvested information from the design including the monitor width, monitor depth, monitored signal names, trigger width, maximum number of triggers, trigger signal names, stimuli width, stimuli signal names, and user clock frequency. When an [active project and implementation \(page 229\)](#) is available, the Snapshot Debugger view automatically loads the implementation `names.snapshot` file to pre-populate the relevant fields of the view. When generated, the file contains only a subset of a complete Snapshot configuration, and thus a generated `names.snapshot` file should not be used to drive [Snapshot in batch mode \(page 387\)](#) via Tcl.

The `names.snapshot` configuration file can be loaded as a starting point to map the Snapshot RTL configuration into the Snapshot Debugger View. The Snapshot settings can be further customized and saved as custom Snapshot configuration files for later use.

Snapshot in Batch Mode

It is also possible to run Snapshot from ACE in batch mode or from the GUI Tcl Console. To do so, use the TCL command [run_snapshot \(page 700\)](#). Note that [run_snapshot \(page 700\)](#) requires the use of a [previously-saved \(page 387\)](#) Snapshot configuration file (`.snapshot`), and allows some values to be overridden by Tcl options. It can run Snapshot through either the Arm or Startup Trigger mode. The latter is selected with the `-startup_trigger` option. See the [run_snapshot \(page 700\)](#) command in the TCL Command Reference section for further details about the Tcl options. Note, [run_snapshot \(page 700\)](#) functions the same as the [Snapshot Debugger view \(page 137\)](#). After running Snapshot, the JTAG connection status is retained, meaning that if the JTAG connection is open prior to calling [run_snapshot \(page 700\)](#), it will stay open, and if the JTAG connection is closed prior to calling [run_snapshot \(page 700\)](#), it will close the connection when programming completes.

The Snapshot configuration file may be edited manually in a text editor, or by configuring the [Snapshot Debugger view \(page 137\)](#) in the ACE GUI and [saving the Snapshot configuration \(page 387\)](#).

Example Snapshot Configuration File

```
#Snapshot Configuration File
#Tue Sep 12 13:52:54 PDT 2017
files_relative_to_project=1
frequency=322.0
log_file=./impl_1/log/snapshot.log
monitor_ch0.name=reset_n
monitor_ch1.name=stimuli_valid
monitor_ch10.name=limit_a[7]
monitor_ch11.name=counter_a[0]
monitor_ch12.name=counter_a[1]
monitor_ch13.name=counter_a[2]
monitor_ch14.name=counter_a[3]
monitor_ch15.name=counter_a[4]
monitor_ch16.name=counter_a[5]
monitor_ch17.name=counter_a[6]
monitor_ch18.name=counter_a[7]
monitor_ch19.name=counter_b[0]
monitor_ch2.name=arm
monitor_ch20.name=counter_b[1]
monitor_ch21.name=counter_b[2]
monitor_ch22.name=counter_b[3]
monitor_ch23.name=counter_b[4]
monitor_ch24.name=counter_b[5]
monitor_ch25.name=counter_b[6]
monitor_ch26.name=counter_b[7]
monitor_ch27.name=counter_b[8]
monitor_ch28.name=counter_b[9]
monitor_ch29.name=counter_b[10]
monitor_ch3.name=limit_a[0]
monitor_ch30.name=counter_b[11]
monitor_ch31.name=counter_b[12]
monitor_ch32.name=counter_b[13]
monitor_ch33.name=counter_b[14]
monitor_ch34.name=counter_b[15]
monitor_ch4.name=limit_a[1]
monitor_ch5.name=limit_a[2]
monitor_ch6.name=limit_a[3]
monitor_ch7.name=limit_a[4]
monitor_ch8.name=limit_a[5]
monitor_ch9.name=limit_a[6]
monitor_width=38
num_samples=4096
num_triggers=3
pre_store=0%
repetitive_trigger.overwrite_vcd=0
repetitive_trigger.record_limit=10
```



```
repetitive_trigger.vcd_record_limit=10
snapshot_version=3
stimuli=110010100
stimuli_ch0.name=stimuli[0]
stimuli_ch1.name=stimuli[1]
stimuli_ch2.name=stimuli[2]
stimuli_ch3.name=stimuli[3]
stimuli_ch4.name=stimuli[4]
stimuli_ch5.name=stimuli[5]
stimuli_ch6.name=stimuli[6]
stimuli_ch7.name=stimuli[7]
stimuli_ch8.name=do_reset
stimuli_ch9.name=stimuli_ch9
stimuli_width=9
trigger1=XXXXXXXXXXXXXXXXXXXX00110101XXXXXXXXXXXX
trigger1.select_using_and=1
trigger2=XXXXXXXXXXXXXXXXXXXX1111R000XXXXXXXXXXXX
trigger2.select_using_and=1
trigger3=XXXXXXXXXXXXXXXXXXXXFXXXXXXXXXXXXXXXXX
trigger3.select_using_and=1
trigger_ch0.name=reset_n
trigger_ch1.name=stimuli_valid
trigger_ch10.name=limit_a[7]
trigger_ch11.name=counter_a[0]
trigger_ch12.name=counter_a[1]
trigger_ch13.name=counter_a[2]
trigger_ch14.name=counter_a[3]
trigger_ch15.name=counter_a[4]
trigger_ch16.name=counter_a[5]
trigger_ch17.name=counter_a[6]
trigger_ch18.name=counter_a[7]
trigger_ch19.name=counter_b[0]
trigger_ch2.name=arm
trigger_ch20.name=counter_b[1]
trigger_ch21.name=counter_b[2]
trigger_ch22.name=counter_b[3]
trigger_ch23.name=counter_b[4]
trigger_ch24.name=counter_b[5]
trigger_ch25.name=counter_b[6]
trigger_ch26.name=counter_b[7]
trigger_ch27.name=counter_b[8]
trigger_ch28.name=counter_b[9]
trigger_ch29.name=counter_b[10]
trigger_ch3.name=limit_a[0]
trigger_ch30.name=counter_b[11]
trigger_ch31.name=counter_b[12]
trigger_ch32.name=counter_b[13]
trigger_ch33.name=counter_b[14]
trigger_ch34.name=counter_b[15]
```


```
trigger_ch4.name=limit_a[1]
trigger_ch5.name=limit_a[2]
trigger_ch6.name=limit_a[3]
trigger_ch7.name=limit_a[4]
trigger_ch8.name=limit_a[5]
trigger_ch9.name=limit_a[6]
trigger_mode=Single
trigger_width=38
vcd_file=./impl_1/output/snapshot.vcd
```


Programming a Device using JTAG in the Download View

This section describes the GUI features and end user tasks for programming a bitstream using a JTAG connection. Other modes of bitstream programming such as CPU, Serial Flash, and 2-stage over PCIe are covered in the *Configuration User Guide*.

NOTE

The JTAG connection must be configured before using the Download View!

ACE interacts with the FPGA using the JTAG interface through a Bitporter2 pod, or a FTDI FT2232H or FT4232H device. This JTAG interface must be properly configured in ACE before using the Download view. The configuration is managed using the [Configure JTAG Connection preference page \(page 190\)](#), which is easily accessible by clicking the  **Configure JTAG Interface** button in the Download view. See [Configuring the JTAG Connection \(page 360\)](#) for more details.

A bitstream (*.hex) file can be downloaded to the FPGA or eFPGA device from the [Download view \(page 40\)](#). To access the Download view, change to the  Programming and Debug perspective, or select **Window** → **Show View...** → **Others** → **Download View**.

From this view, bitstream files can be selected for downloading, programming options can be adjusted, utility functions can be called against connected devices, and the actual download to the target device may be triggered.

Selecting a Bitstream File

By default, the [Download View \(page 40\)](#) automatically remembers the last *.hex bitstream file opened by the current user, and pre-populates the filename/path with the last-used value when ACE is started.

When a different bitstream file is desired, it can be chosen by either selecting a hyperlink from the provided lists of bitstream files, or by pressing the **Browse** button.

As elsewhere in ACE, pressing the **Browse** button allows searching for *.hex files in the filesystem using the graphical file browser native to the operating system desktop environment.

The view also remembers the last dozen or so files the chosen, and lists them in the Suggested Recently Used lists as hyperlinks. The list only displays files which are still valid choices. Previously used files which have been deleted from the file system are not be displayed in the list of suggestions. Select any listed hyperlink to make that file the active *.hex bitstream file to be downloaded.

ACE also includes a suggestion list of all the *.hex files that exist for the [Active Project and Implementation \(page 229\)](#). ACE updates this suggested file list whenever new files are created or when files are deleted. Unlike in prior

versions, ACE no longer attempts to automatically pick which bitstream file should be used. A hyperlink must now be manually chosen when switching to a file to be the active bitstream used. Recent configuration feature improvements, such as support for Partial Reconfiguration and Multi-Stage Programming, make it impossible for the Download view to always display the bitstream file that is most needed.

Lab Mode

The **Browse** button continues to work normally when in Lab Mode. The list of suggested **Recently used** files continues to be populated and usable. This list is updated if any new files (not already in the list) are chosen using the **Browse** button. But when ACE is in Lab Mode, it is impossible to load projects/implementations meaning there is never a "current" active project/implementation. Thus the list of suggested file hyperlinks from the active implementation is empty in Lab Mode.

Selecting Bitstream Programming Options

While getting ready to use the [Download view \(page 40\)](#) to program a device with a bitstream *.hex file, there are some configuration options available to be adjusted by toggling checkboxes in the view. Any adjustments should be made prior to each bitstream file download.

Table 146 · JTAG Programming Options

Option	Description
Perform an FCU reset to clear the (e)FPGA config memory	When checked, performs a soft reset and clears all device configuration memory before beginning programming. This reset is typically only disabled for multi-stage programming after stage 0 programming has completed but before programming later stages begins, or for Partial Reconfiguration when partial bitstreams are in use (see the chapter titled Partial Reconfiguration in the <i>Speedster7t Configuration User Guide (UG094)</i> for more details).
Close the JTAG connection/device when programming is complete	By default, the chosen JTAG connection (see Configuring the JTAG Connection (page 360)) is opened at the beginning of programming (if it was not already open) and is closed when programming is completed. Uncheck this box to leave the named JTAG connection open when programming is complete, allowing additional JTAG interactions to occur over the same open connection.

Downloading the Bitstream to the Target Device

When using the [Download view \(page 40\)](#), after finishing [Selecting a Bitstream File \(page 390\)](#), and [Selecting Bitstream Programming Options \(page 391\)](#), perform the download to the target device by pressing the **Download Bitstream** button.

This causes the Tcl command `jtag::download_bitstream` to be executed in the [Tcl Console view \(page 142\)](#) using the appropriate arguments. All feedback from the command (informational, warnings, and errors) appears in the Tcl Console view, and also is saved in the ACE session [Log file \(page 231\)](#).

For more information about the JTAG Tcl interface commands, and for troubleshooting information regarding common programming problems, please see the *Speedster7t Configuration User Guide* (UG094).

Optimizing a Design

There are numerous methods of design optimization available in ACE.

Many optimizations can be performed automatically by ACE, at the cost of additional runtime. These automatic optimizations are managed at a granular level through the **implementation options** (page 0), which may be configured from the **Options view** (page 96) and/or the **set_impl_option** (page 711) Tcl command.

Achronix optimization experts have also collected together into **option sets** (page 0) the implementation options which are known to work well together. These option sets may be used to create new implementations for user designs, to allow comparing/contrasting how various optimizations affect achieved frequencies and required runtimes.

Other optimizations must be performed manually, typically by editing the design source RTL or `.sdc` timing constraints. **Analyzing critical paths** (page 353) is an important part of this process. Optimization through RTL changes is currently beyond the scope of this document — ask your Achronix Field Applications Engineer for more information regarding source optimization possibilities.

Attempting Likely Optimizations Using Option Sets

In addition to **running multiple flows in parallel** (page 308), the **Multiprocess view** (page 73) allows generating new implementations with auto-generated combinations of **implementation options** (page 0). These known-good subsets of implementation options are called **option sets** (page 0).

ACE can generate customized option sets based on design details (such as the target device) found within the **active project and implementation** (page 229). These customized option sets are only generated by request, and are specific to the details of each implementation. To generate the customized option sets for the active implementation, use the Refresh Option Sets button in the **Multiprocess view** (page 73), or the option `-create_option_sets` when calling the **run_multiprocess** (page 692) Tcl command.

The Multiprocess view allows selecting a starting template **implementation** (page 229) and then generating new implementations using the template implementation as a base. Each generated implementation overrides the implementation options found in the template implementation with the specified option set configuration (an overriding subset of the full collection of implementation options). The majority of the implementation options within the generated implementation are left with the same settings as existed in the template implementation. Only the options specified in the option set are overridden to take on new values. The newly generated implementation is given a name which includes the option set name for clarity (the generated name uses the template implementation name as a prefix, with the option set name as the suffix).

See the information in **Running Multiple Flows in Parallel** (page 308), which discusses the basic use of the **Multiprocess view** (page 73) and **Multiprocess Summary report** (page 255) — the rest of this section builds upon those descriptions.

Selecting the Implementations to be Generated and Run in Parallel


After **finding the Multiprocess view** (page 308), **configuring the execution queues** (page 308), and **configuring the desired flow to be followed by the selected implementations** (page 315), the implementations to be generated may be selected:

1. In the **Projects view** (page 117), select (activate) the desired **project** (page 222) and **implementation** (page 229).

2. Select the radio button labeled **Generate Implementations from Option Sets**, found within the Multiprocess view **Select Implementations** (page 76) section.
3. Click the **Refresh Option Sets** button to generate new option sets particular to the details of the active project/implementation. All of these custom option sets include "auto" in their names.
4. The Implementation Table within the Multiprocess view **Select Implementations** (page 76) section is then updated to display a collection of potential implementations based upon the **active implementation** (page 229), with names derived from the refreshed option sets.

The first entry in the Implementation Table is the active implementation itself. This implementation is the template from which all the generated implementations are derived. All other implementations in the table are generated, one for each option set, if they are selected (their checkbox is checked) when background execution is started. The Description column of the table indicates succinctly what implementation option changes are caused by each option set (thus describing how each generated implementation differs from the template, the active implementation).


Generating Option Set Implementations and Starting Background Execution

After the () **Start Selected** button has been pressed, but before the behavior described in **Starting Background Execution** (page 316) commences, ACE:

1. removes implementations in the active project with the same name as to-be-generated implementations
2. creates new implementations (exact copies of the template implementation) with the required names
3. applies the appropriate option set implementation options to the new implementations (overriding the inherited implementation option values with the subset making up the option set)
4. adds all selected (checked) implementations to the background processing queue(s), to be run through the flow

From this point on, the available functionality and behavior is identical to that described in **Running Multiple Flows in Parallel** (page 308), beginning with **starting background execution** (page 316).

Warning!

Each generated implementation which is selected overwrites *without prompting* any already-existing implementation with the same name in the active project. The template (active) implementation is not changed/overwritten. If a previously-existing implementation with a to-be-generated name collision is kept, the previously-existing implementation must be renamed to avoid the name collision *before* the () **Start Selected** button is pressed.

Interpreting/Utilizing the Results

After **viewing the results** (page 317), the final step of an optimization pass is usually comparing the results and choosing which generated option set implementation provides the best QoR in comparison to the template implementation.

By default, the implementations in the **multiprocess summary report** (page 255) are sorted approximately by QoR, though it is likely still preferred to analyze the results in detail to choose which implementation is the best by preferred criteria.

Note**Early Access Functionality**

The automatic QoR sorting of the Multiprocess Summary report should be considered early access functionality. The sort details are likely to change (and improve) in future ACE releases.

That best generated implementation could then be renamed, so that it does not get overwritten by future multiprocess runs (e.g., it might be named "fastest1", "lowestpower1", etc.).

With the newly-renamed implementation selected in the Projects view (making it the active implementation), it also becomes the new template implementation in the Multiprocess view, ready for another multiprocess iteration through the option sets.

By iterating through several best template implementations (perhaps each with a new implementation name for "breadcrumb" purposes), the desired QoR may be reached.

Caution!

Ensure **Generate Implementations from Option Sets** is selected for each optimization iteration, otherwise any changed implementation options in the template implementation are not inherited by the option set implementations.

Also, there is a scenario where all multiprocess results can be identical. The cause and a workaround are described in [Running Multiple Flows in Parallel \(page 317\)](#).

Placement Regions and Placement Region Constraints

Warning!

Placement Regions and Placement Region Constraints are an advanced feature, and should only be used under the guidance of an Achronix FAE. Unguided use of placement region constraints can cause loss of QOR, or may make a design impossible for the Placer or Router to solve.

Note**ACE automated placement often produces better QOR than user-defined placement regions/constraints**

When attempting to use Placement Regions and Placement Region Constraints, it is highly recommended that a parallel implementation of the project lacking the user-defined Placement Regions be kept. In a number of tested cases, completely automated placement in ACE was able to produce better QOR than with user-defined Placement Regions and Placement Region Constraints. This can easily be achieved by keeping the placement region constraints in a separate pdc file, which can then be enabled or disabled for the place and route flow.

Placement Regions are user-defined rectangular areas of the core fabric (*not* the IO Ring), to which the user can inclusively constrain the placement of multiple instances from their design, without needing to manually assign instances to specific sites within that region.

Because of clock distribution limitations, only a finite number of clocks can be routed to each of the [Clock Regions \(page 262\)](#) in the fabric. Placement regions allow advanced users to ensure that those constraints are met if

the automated tools need guidance. When necessary, clocked instances (flops, BRAMs, etc) may be constrained to placement regions to guarantee ACE does not attempt routing more clocks into a region than the region can support.

Placement Regions and the associated instance placement constraints may be manipulated through Tcl, or via the ACE GUI using the [Floorplanner View \(page 43\)](#) and [Placement Regions View \(page 110\)](#). The [Search View \(page 129\)](#), [Selection View \(page 133\)](#), [Critical Paths View \(page 37\)](#), and [Netlist Browser View \(page 79\)](#) may also be used to assign instance placement constraints by using drag-and-drop operations.

Be aware that Placement Regions are not treated as distinct objects in the ACE design database. Thus, they do not have their own [object type prefix \(page 335\)](#), nor are they directly searchable in the [Search View \(page 129\)](#) or with the `Tcl find` [\(page 638\)](#) command.

Placement Region Preferences

There are a number of user preferences which may be configured to alter how the mouse creates placement regions and assigns placement region constraints. These preferences are found on the [Placement Regions Preference page \(page 213\)](#).


Creating a New Placement Region

Placement regions may be created/defined by using the mouse in the Floorplanner View, or by directly calling the Tcl command

```
create_region (page 624)
```

. In both cases, the bounds of the created region may "snap to" (grow to encompass) the entirety of all enclosed Clock Region boundaries or tile boundaries.

To create a Placement Region using the mouse in the Floorplanner view:

1. Ensure the  Floorplanner Placement Region Tool is active.
2. (Optional) If the Placement region is meant to align with one or more [Clock regions \(page 262\)](#), enable the overlays for those regions from the [Clock Regions view \(page 23\)](#). This does not affect the functionality in any way, but makes it easier to know where to define the region bounds.
3. Press and hold the left mouse button with the cursor at one of the corners of the area to be defined as the new Placement region.
4. While still holding the left mouse button, drag the cursor to the opposite corner of the desired Placement region area. Release the left mouse button when the cursor reaches the desired location.
5. ACE calculates the enclosed subtile grid coordinates, growing the grid as necessary to ensure all partially-enclosed tiles are fully enclosed.
6. The [Create Placement Region dialog \(page 167\)](#) pops up pre-populated with the calculated subtile coordinates:

Create Placement Region

You can create placement regions in the Core and constrain instances to them later via drag and drop or TCL commands.

Region Name

Include Routing

Region Alignment

None

Snap to Tile Boundaries

Snap to Fabric Clusters

Snap to Clock Region Boundaries

Region Type

Inclusive

Keep out

Soft

Subtile Grid Coordinates

X1 Coordinate

Y1 Coordinate

X2 Coordinate

Y2 Coordinate

Figure 166 - Pre-Populated Create Placement Region Dialog


7. Fill in the desired Placement region name.
8. Select whether the Placement region should be snapped to align with the edges of the **Clock regions** (page 262), or the **fabric clusters** (page 281), or with the more granular grid of basic resource tiles.
9. Select whether the Placement region should be an inclusive region, a "keep out" region, or a soft region (see the **create_region** (page 624) Tcl command documentation for more information on these options).
10. Click the **Finish** button to create the new Placement Region.

11. ACE adds the new Placement Region to the table in the [Placement Regions View \(page 110\)](#) and displays it as a translucent overlay within the Floorplanner (at this point, the region contains no constraints).

Resizing an Existing Placement Region

Existing Placement Regions may be resized with the [set_region_bounds \(page 715\)](#) Tcl command, or with the mouse in the Floorplanner view. Any existing Placement Region Constraints for that region are kept — only the enclosed area is updated.

To resize a Placement Region with the mouse in the [Floorplanner View \(page 43\)](#):


1. Ensure the  Floorplanner Placement Region tool is active.
2. In the [Placement regions view \(page 110\)](#), ensure the checkbox in the first column is selected for the desired Placement region. This makes the Placement region overlay visible within the Floorplanner view.
3. Ensure the **Snap To:** option in the [Placement Regions Preference page \(page 213\)](#) is configured as desired.
4. (Optional) If the Placement region is meant to align with (snap to) one or more [Clock regions \(page 262\)](#), enable the overlay for those regions from the [Clock Regions view \(page 23\)](#). This action does not affect the functionality during the resize in any way, but makes it easier to know where to define the region bounds.
5. Move the mouse over any of the four corners of the placement region to be resized. The mouse cursor changes to a diagonal resize cursor when in a potential resize location.
6. Press and hold the left mouse button and drag the mouse to expand or shrink the Placement region area as desired.
7. Release the left mouse button when the mouse is at the desired location.
8. ACE calculates the enclosed subtile grid coordinates, growing as necessary to ensure all partially-enclosed subtiles (or Clock regions) are fully enclosed.
9. The Placement Region View table content is updated to show the new site counts enclosed by the Placement region, and the Floorplanner is updated to show the new Placement region overlay.

Moving an Existing Placement Region

Existing Placement Regions may be moved with the [set_region_bounds \(page 715\)](#) Tcl command, or with the mouse in the [Floorplanner view \(page 43\)](#). Any existing Placement Region Constraints for that region will be kept — only the enclosed area will be updated.

Be aware that the **Snap To** setting is enforced during the move — the enclosed area might not stay the same dimensions after the move. As with creating/resizing a region, the area will grow to ensure there are no partial sites in the enclosed area. It is frequently desired to resize (shrink) the Placement Region after a move, as it can easily grow larger than expected if sites/Clock Regions were partially enclosed at the ending mouse location.

To move a Placement Region with the mouse in the Floorplanner:

1. Ensure the  Floorplanner Placement Region Tool is active.
2. In the Placement Regions view, ensure the checkbox in the first column is selected for the desired Placement Region. This selection makes the Placement Region visible within the Floorplanner view.
3. Ensure the **Snap To** option in the [Placement Regions Preference Page \(page 213\)](#) is configured as desired.
4. (Optional) If the Placement Region is meant to align with (snap to) one or more [Clock Regions \(page 262\)](#), enable the overlay for those regions from the [Clock Regions view \(page 23\)](#). Enabling the overlay does not affect the functionality during the resize in any way, but makes it easier to know where to define the region bounds.

5. Move the mouse over the placement region to be moved. The mouse pointer changes to a "move" cursor when the mouse is over any placement region.
6. Hold the left mouse button while dragging the mouse to the desired new location for the placement region.
7. Release the left mouse button when the upper-left corner of the dragged region is at the desired location.
8. ACE calculates the enclosed subtile grid coordinates, growing as necessary to ensure all partially-enclosed subtiles (or Clock regions) are fully enclosed.
9. The Placement Region View table content is updated to show the new site counts enclosed by the Placement region, and the Floorplanner is updated to show the Placement region overlay at the new location (and with the latest dimensions).

Assigning Placement Region Constraints

Placement region constraints may only be assigned to core and boundary instances (not I/O pads). Instances may be assigned placement region constraints interactively from the Tcl console, or from a PDC constraint file, with the [add_region_insts](#) (page 614) and [add_region_find_insts](#) (page 613) Tcl commands, or interactively with drag-and-drop mouse actions in the ACE GUI.

When using the [add_region_insts](#) (page 614) or [add_region_find_insts](#) (page 613) Tcl commands, the instances to constrain may be specified using an explicit list of instance names, or by clock domain name or critical path ID.

If specified as an explicit list of instance names the list may be formatted explicitly, or it may be the output of a [find](#) (page 638) Tcl command.

```
add_region_insts "region_1" {i:inst1 i:inst2}
add_region_insts "region_1" [find -insts inst*]
add_region_insts "region_1" [find -insts inst* -filter {@type=DFF && @clock_domain=clk1}]
]
```

If specified by critical path ID, ACE determines which instances are part of that critical path, and assigns the placement region constraint to those instances.

```
add_region_insts "region_1" {c:sc_s0}
```

Likewise, if specified by clock domain name, ACE determines which instances are part of that clock domain, and assigns the placement region constraint to all of those instances.

```
add_region_insts "region_1" {k:clka}
```

When writing PDC constraint files, the recommended practice is to use the [add_region_find_insts](#) (page 613) command instead of the [add_region_insts](#) (page 614) command. This is because:

1. When the instance list is specified with a [find](#) (page 638) command, or by critical path ID or clock domain name expression, the command/expression is evaluated and expanded into a list at the time at which the [add_region_insts](#) (page 614) command is evaluated (which happens at the beginning of the [run_prepare](#) (page 697) flow step), not at the time at which it is applied with the [apply_placement](#) (page 615) command (which happens at the end of the [run_prepare](#) (page 697) flow step).

2. New instances which may be created during the [run_prepare](#) (page 697) flow step, even if they would have matched the command/expression, are not included. Therefore, the [add_region_insts](#) (page 614) command is best reserved for interactive use.
3. The [add_region_find_insts](#) (page 613) command, on the other hand, specifies the [find](#) (page 638) command as a string argument to be batched and evaluated later during the [apply_placement](#) (page 615) command.

```
add_region_find_insts "region_1" "find -insts inst*"
add_region_find_insts "region_1" "find -insts inst* -filter {@type=DFE &&
@clock_domain=clk1}"
```

Note

Saving Critical Path or Clock Domain Constraints

When critical paths or clock domains are used to specify the constraint, they are immediately expanded into a list of the corresponding instances within ACE at the time at which the

[add_region_insts](#) (page 614)

command is evaluated. If the placement region constraints are later exported from ACE (and saved into a pdc file), they are exported as explicit lists of instance names and the original association with a critical path or clock domain is lost. More concise constraints for user designs may be created by manually entering the placement region constraint in the PDC file using the clock domain name instead of the list of explicit instances.

If any instance which was previously assigned a placement region constraint is assigned a new placement region, the prior constraint is overridden and discarded.

Optionally, placement region constraints may be restricted to allow only flops, in which case all other instances are excluded. (Setting these inclusion/exclusion preferences for mouse actions is done on the [Placement Regions Preference page](#) (page 213).)

```
add_region_insts -flops_only "region_1" [find -insts * -filter {@clock_domain=clk1}]
add_region_find_insts -flops_only "region_1" "find -insts * -filter
{@clock_domain=clk1}"
```

When placement region constraints are assigned to instances interactively using drag-and-drop mouse actions in the ACE GUI, the mouse drag-assign actions can start from:

- the Search view, where individual Instances and/or Paths, groups of Instances and/or Paths, or all Instances and/or Paths in the search results (if the titled branch nodes themselves are dragged, even the Instances/Paths not in the current set of 200 on the visible page of results) may be drag-assigned.
- the Selection view, where individual Instances and/or Paths, groups of Instances and/or Paths, or all Instances and/or Paths in the selection set (if the titled branch nodes themselves are dragged, even the Instances/Paths not in the current set of 200 on the visible page of results) may be drag-assigned.
- the Critical Paths view, where individual Paths or groups of paths may be drag-assigned.
- the Clock Domains view, where clock domains may be drag-assigned to include all applicable Instances from that clock domain in the assignment.
- the Netlist Browser view, where any node of the tree may be dragged, and all child nodes are included.

Mouse drag-assign actions can end at:

- An individual Placement Region row in the table within the Placement Regions View. After the assignment of the dropped Core/Boundary Instances completes, the site utilization counts are updated.
- A visible Placement Region overlay in the Floorplanner view, if the Placement Region Tool is active in the Floorplanner. After the assignment of the dropped Core/Boundary Instances completes, the site utilization counts in the Placement Regions view for that region are updated.

Note

Overlapped Placement Regions

If multiple placement regions overlap visibly in the Floorplanner view, any Instances dropped within the visibly overlapping area are ignored. In such cases, instances must either be dropped in the Placement Regions view, or dropped in the Floorplanner view where there is no visible overlap (Placement Region overlays may be disabled from the Placement Regions view to eliminate visible overlaps — in these cases, constraint assignment occurs to whichever placement region remains visible at the Floorplanner drop location).

Listing all Objects Constrained to a Placement Region

The count of total sites of each type within each placement region is listed in the [Placement Regions view \(page 110\)](#), along with the count of each Instance type for the sites.

If there are more instances constrained to a region than there are sites for that region, the corresponding cell in the Placement Regions view table turns red to indicate the problem.

To view a list in the [Tcl Console view \(page 142\)](#) of all objects constrained to a placement region, do one of the following:

- Use the [get_region_insts \(page 659\)](#) Tcl command.
- Right-click the desired Placement region in the Placement Regions view and select **Print Instances**.

Removing a Placement Region Constraint from an Object

Placement region constraints may be removed from individual core/boundary instances, or from all instances assigned to a region at once.

To un-assign a placement region constraint for individual core instances, use the [remove_region_insts \(page 676\)](#) Tcl command.

To remove all instance constraints from a placement region, use the same Tcl command, or in the [Placement Regions view \(page 110\)](#), right-click the desired placement region, and select **Clear Placement Region**.

Saving Placement Region Definitions and Placement Region Constraints

Placement region constraints may be saved:

- From the [Floorplanner view \(page 43\)](#) with the "Save Pre-placement Constraints" action (which displays the [Save Placement dialog \(page 182\)](#))
- From the [Placement Regions view \(page 110\)](#) with the "Save Placement Regions" action (which displays the [Save Placement Regions dialog \(page 184\)](#))

- By using the [save_regions](#) (page 708) Tcl command directly

Note

Important Consideration When Saving Placement Region Constraints

Only the final list of all individual instances being constrained is saved. The individual Tcl commands which built up the final list of constraints (including "find" commands, the extraction of instances from critical paths, or from clock domains) is lost. The saved PDC file may be edited to replace explicit lists of instances with `find` commands or clock domain names.

Deleting Placement Regions

Unwanted Placement Regions may be deleted from the [Placement Regions view](#) (page 110) by right-clicking the region in the table and selecting **Remove Placement Region**.

Alternately, the [remove_region](#) (page 676) Tcl command may be called directly.

Running the HW Demo

The HW Demo facility is primarily intended as an aide to Achronix field application engineers (FAEs) that allows them to conveniently demonstrate particular features of Achronix FPGAs. Demonstration designs built into the ACE GUI software can easily be loaded into the attached board/device and executed. As the demonstration design is executing, the status of the design can be monitored in real-time, and visually represented within the HW Demo display.

The HW Demo facility uses fully functional designs (not included within an ACE installation, but provided as directory overlays) to demonstrate the real world application of hardened IP blocks. A given design may consist of a single IP block type, but typically they combine several IP block types working in a coordinated manner. These prebuilt designs are also useful to new ACE users as a way to gain experience setting up the Bitporter and prototyping environments.

Installing HW Demo Designs

Each HW demo or reference design (including bitstreams, additional software, documentation, and source files when possible) is packaged into and delivered in a single tarball, ZIP, or Windows installer file, downloadable from the Achronix support site. A set of installation instructions is provided as a separate document (not here) as the details may vary for each design. Installation might require several steps, depending on the software tools and drivers needed.

There are expected to be two types of designs available. Reference designs are meant to be modified, while demo designs are black boxes. Reference designs typically are installed into the user home directory (to encourage editing), while demo designs may be installed into the `<ace_install>` directory (which often has read-only permissions to discourage accidental overwrites). Both design types use the same framework within ACE, and both are presented through the HW Demo view in the ACE GUI.

Ask your FAE for further details about acquiring and installing the HW demo and reference designs for your specific development kit.

HW Demo Installation Paths

By default, when ACE starts up, it searches for installed HW demos in the following paths:

- <userhome>/achronix/ref_designs/
- <ace_install>/ref_designs/

After downloading a design tarball or zip, the design should be unpacked into either of those directories.

Selecting The Target Device And Demo

At the top of the [HW Demo view \(page 58\)](#) are controls for selecting the target device and an associated demonstration design. After selecting a target device (or the default device matches the device you are working with) the list of available designs is accessible in the **Demo Design** control.

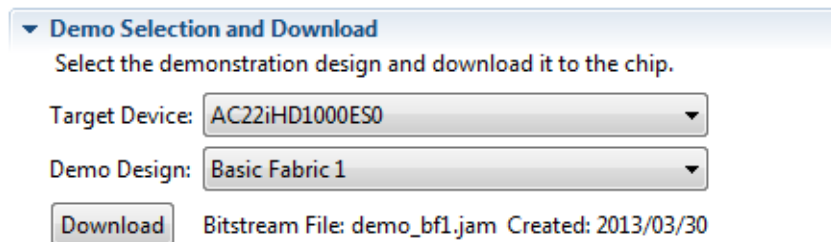


Figure 167 • Demo Design Control Example

Note

If no demos are installed, these controls remain disabled (which indicates the lack of installed designs).

Loading The Demo JAM File

After selecting the target device and demonstration design, the name of the *.jam file appears to the right of the **Download** button. Click the **Download** button to initiate loading of the the design into the attached FPGA device. Any designs that are running when **Download** is clicked are terminated without warning. If there are any errors or problems during the download process, a pop up dialog will be displayed with an explanatory message. When the selected design has been loaded and started, monitoring of the attached FPGA device is initiated using the DCC connection.

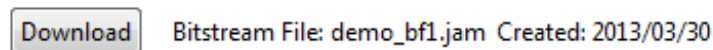


Figure 168 • Demonstration Design Download Example

Displaying Board Status

After a design has been loaded and started running, ACE may monitor the status of the demonstration board LEDs and DIP switches, as well as key internal conditions such as core voltage, temperature, etc. Clicking the visualization of an LED in the HW Demo view causes the corresponding actual LED (on the demonstration board) to toggle state.

Note

The visualized DIP switches are only used for reporting the state of the corresponding actual switch on the demonstrations board. The physical DIP switch cannot be set by clicking its image in ACE.

Board Status

Once a demonstration design is downloaded, this section will display the status of the boards LEDs, DIP switches, temperature, current, etc.

















<p>LED State</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 01234567 </div> <p>DIP Switch State</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> <div style="text-align: center;"></div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 01234567 </div>	<p>Device State</p> <p>Status: Disconnected</p> <p>Demo Version: 0.0</p> <p>Core Voltage: 0.0 V</p> <p>Current: 0.0 mA</p> <p>Power: 0.0 W</p> <p>Temperature: 0.0 C</p>
--	---

Figure 169 - Rudimentary Demo Design Example

Control of Running Demonstration Design

While the Snapshot Debugger has extensive facilities for collecting data samples from a running design, it does not currently provide any direct mechanisms for controlling or interacting with a design. The [HW Demo view \(page 58\)](#) may provide a simple set of on-screen controls for reading and writing register values in some demo designs. In a demo similar to the example shown below, to read a register value, enter its address and click the **Read** button. The current value of the specified register appears in the **Data:** field to the right. Likewise, to modify a register value, enter its address and new value in the provided fields, and click the **Write** button.

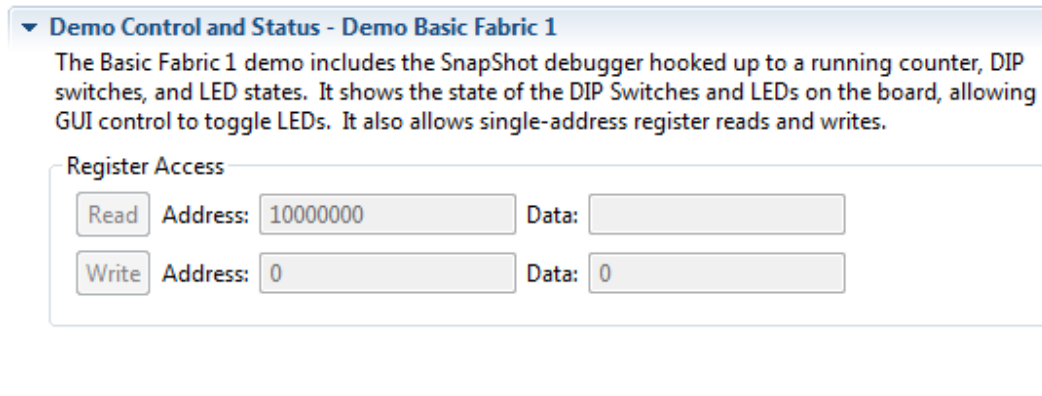


Figure 170 • HW Demo View Example

Using Incremental Compilation (Partitions)

This section begins with a high-level overview, and then continues with detailed tutorials.

Overview of Incremental Compilation and Partitions

Upstream synthesis tools have the ability to break a design up into smaller logical units (see: *Synplify Pro for Achronix User Guide*, Chapter 11: Working with Compile Points). Within ACE these smaller logical units are called "Partitions". These partitions can each be thought of as a nearly independent block — each partition can potentially be synthesized, optimized, placed, and routed independently. Because of this independence, when only one partition changes, only that partition needs to be re-run through the flow, leading to a significant runtime savings.

Defining Partitions

It is expected that partitions are defined primarily by the upstream synthesis tool. The synthesis tool typically exports a partition definition/constraint file. For example, the file below is an example of a *.prt file exported by Synplify Pro for Achronix.

Example partition definition (*.prt) file

```
set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1476335212" -cp_type
"hard"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out/fpu_out_ctl" -view
"fpu_out_ctl" -timestamp "1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out" -view "fpu_out" -timestamp
"1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div/fpu_div_ctl" -view
"fpu_div_ctl" -timestamp "1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div" -view "fpu_div" -timestamp
"1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_exp_dp" -view
"fpu_mul_exp_dp" -timestamp "1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_ctl" -view
"fpu_mul_ctl" -timestamp "1476337611" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul" -view "fpu_mul" -timestamp
"1476337611" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in/fpu_in_ctl" -view
"fpu_in_ctl" -timestamp "1476335212" -cp_type "locked"
set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in" -view "fpu_in" -timestamp
"1476335212" -cp_type "locked"
```

Enabling Incremental Compilation

Enabling incremental compilation support within ACE is quite easy, assuming the partitions are already defined through the upstream synthesis tool:

1. In the **Projects view** (page 117), add the partition definition file(s) to the ACE project (see **Adding Source Files** (page 297)). The new partition definition file appears in the Projects view as a **Constraints** file and, in the **Options view** (page 96), in the **Design Preparation** section in the list of **Constraints Files** (and should already have its checkbox selected).
2. In the **Options View** (page 96), within the **Design Preparation** section, select the checkbox labeled **Enable Incremental Compile**.
3. In the Projects View, **save the current project** (page 294).

From this point forward, this Project/Implementation uses incremental compilation when running the flow.

Note

The presence of the partition definition constraint file in the project, plus the checked **Enable Incremental Compile** implementation option, are the only configuration changes that distinguish the incremental compile flow from the standard non-incremental flow.

Tracking Partition Status

ACE provides two main tools for checking the compilation state, timestamps, and other statistics of each partition.

Partitions Report

The **Partitions report** (page 245), automatically generated (and opened in the GUI) during the **Run Prepare flow step** (page 234), shows the current status of each of the partitions, including resource counts and re-compilation states.

Partitions View

Similar to the Partitions report, the **Partitions view** (page 107) shows the status of each partition and a variety of other statistics. Additionally, the view allows for ease of visualization of the partitions and their relationships to the instances and each other.

Forcing an Unchanged Partition to Recompile

When using the Partitions view, ACE provides a mechanism to override the partition timestamp during the next pass through the **flow** (page 234). The column named **Force Re-compile on Next Run** displays the status of this override mechanism.

To mark a partition as needing forced compilation:

1. Click the partition in the Partitions view.
2. Right-click anywhere in the partition row to open the context menu, and choose **Force Partition Changed**.

A check mark appears in the **Force Re-compile on Next Run** column of the view in the row containing the partition. The next time the flow is executed, the partition is re-placed and re-routed, even if there were no RTL changes and it was not re-compiled in the upstream synthesis tool.

To remove the mark for forced recompilation:

1. Click the partition in the Partitions View.
2. Right-click anywhere in the partition row to open the context menu, and choose **Un-Force Partition Changed**.

The check mark disappears in the **Force Re-compile on Next Run** column of the view in the row containing the partition. The next time the Flow is executed, the partition is only re-placed and re-routed if the partition was re-compiled in the upstream synthesis tool.

Note

The ACE forced recompilation flag is a one-time trigger. When compilation is completed, any force flags for that implementation are cleared.

 **Tip****Forcing all Partitions to Re-compile**

The easiest way to force all partitions to immediately be recompiled (run through the entire flow) is:

- Enter the following Tcl command in the [Tcl Console view \(page 142\)](#):

```
run -ic init
```

- Alternately:
 - a. Change to the Projects Perspective.
 - b. In the [Flow view \(page 53\)](#), enable and disable the optional [Flow steps \(page 234\)](#) as desired.
 - c. Right-click any flow step, and select the context menu item **Re-Run Flow with "-ic init"**.

See [Running the Entire Flow \(page 306\)](#) for additional details.

Viewing Instances In Partitions

There are multiple ways to quickly see which instances belong to a given partition:

- The [Search view \(page 129\)](#) and the [find \(page 638\)](#) Tcl command can both be used to list all the instances within a partition or list of partitions, using the @partition filter. The [Search Filter Builder dialog \(page 186\)](#) might ease the building of the filter for the Search view.
- Adding all the instances within a partition to the ACE Selection set (using the [Selection view \(page 133\)](#), especially when populated with search results) is an easy way to see where members of a partition are within the [Floorplanner view \(page 43\)](#). When the Floorplanner layer for **Selected Instance Flylines** is enabled, the connectivity of the selected partition is also visible.
- The [Netlist Browser view \(page 79\)](#) is a table of the instances (and enclosing macros) making up the design, with a column indicating the partition for each instance. This table can be filtered by column values, thus the table can be filtered to include only the instances within a given partition.
- Using highlight colors assigned from any of the above views (especially using the Partitions view **Auto-Highlight** functionality) can make it easy to see how members of various partitions are placed in relation to each other in the Floorplanner.

The Floorplanner view also includes a new color in the [Instance states \(page 262\)](#) for the new "Locked" state relating to partitions. Instances that are locked are a member of a locked partition that has remained unchanged since the prior incremental compilation. ACE does not change the site assignment for that instance during the Placement phase of place-and-route.

Related Tcl Commands

The following Tcl commands were created specifically to interact with partitions:

- [get_partition_changed \(page 651\)](#)
- [get_partition_force_changed \(page 651\)](#)
- [get_partition_info \(page 651\)](#)
- [get_partition_insts \(page 652\)](#)

- [get_partition_timestamp \(page 652\)](#)
- [get_partition_type \(page 652\)](#)
- [is_incremental_compile \(page 666\)](#)
- [report_partitions \(page 680\)](#)
- [set_partition_force_changed \(page 711\)](#)
- [set_partition_info \(page 712\)](#)

Additionally, the following Tcl commands were enhanced with additional options specific to incremental compilation and/or partitions:

- [run \(page 689\)](#)
- [filter \(page 637\)](#)
- [find \(page 638\)](#)

Incremental Compile Tutorial

Overview

This tutorial demonstrates the process of running incremental design compile within ACE. This tutorial consists of two parts:

- [Single-Process Incremental Compile Tutorial \(page 408\)](#) – covers how to process a single-pass incremental compile. This first tutorial must be run before the Multiprocess Incremental Compile tutorial
- [Multiprocess Incremental Compile Tutorial \(page 447\)](#) – details how to run a set of changes in order to select an optimal implementation. This second tutorial expands upon concepts from the first and cannot be run standalone.

Tutorial Files

Note

The files needed for this tutorial are no longer available due to their use of obsolete components. Despite this fact, the procedure outlined in the tutorial remains valid and is presented here to provide a detailed overview of the incremental compile procedure.

This is an advanced tutorial. It assumes that both Synplify Pro and ACE are installed in your system search path and that you are already familiar with the use of both tools. If that is not the case, start with an introductory tutorial for those tools.

Single-Process Incremental Compile Tutorial

The goal of this tutorial is to illustrate the incremental compile flow from an initial version of RTL, through the following steps:

1. Synthesis in Synplify Pro.
2. ACE place and route.

3. A modification of the original RTL.
4. Back through the flow.

The goal of the flow is to help minimize the time it takes to make incremental changes to existing RTL and get those changes through ACE with the minimum amount of time and perturbation to the existing design implementation in ACE.

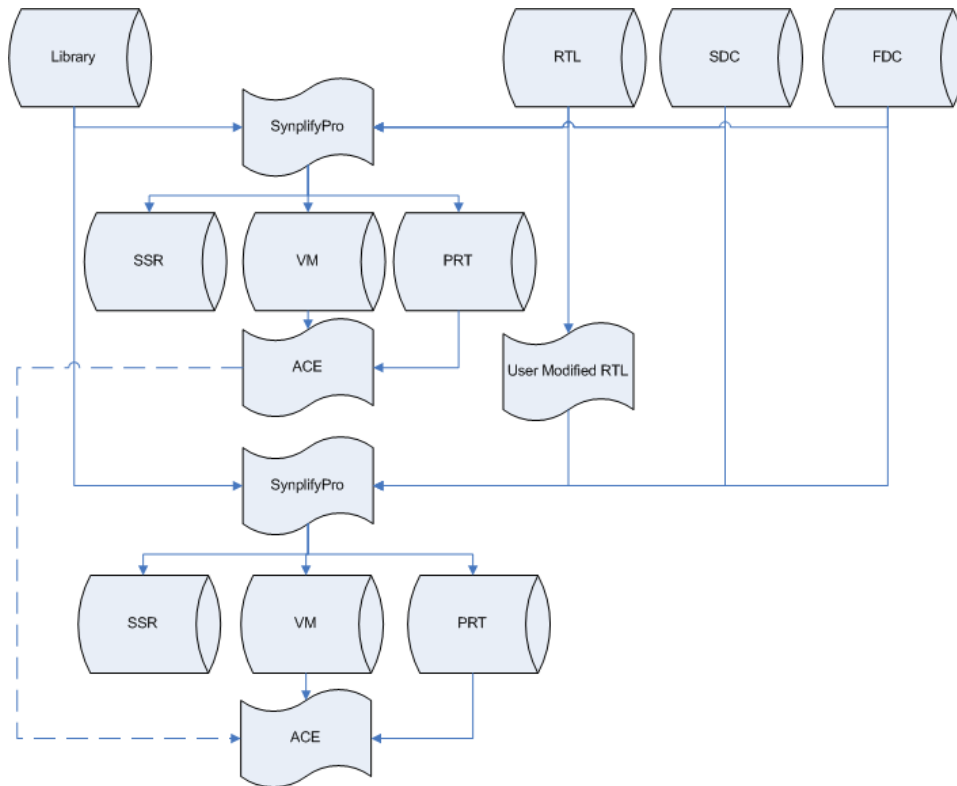


Figure 171 • Incremental Compile Flow Chart

Note

Legend

- ⊕ = Step that is integral to the running of this tutorial.
- ✔ = Items that should be checked at this point in the tutorial to gain insight in to how the flow works and the feedback the tools are providing.
- Bold Text** = Text that can be found as a label to some GUI component including report table headings.

Step 1: Obtaining the Files

The tutorial reference design ZIP file is no longer available. Please refer to the following tutorials for procedural details using your own design files.










Directory	Description
 <ZIP root>	Root directory of ZIP file.
 /ace	ACE generated project, report and log files (empty).
 /constraints	SDC, PDC and FDC constraint files.
 /rtl	Synplify Pro RTL project files.
 /rtl_v1	Synplify Pro RTL project files.
 /rtl_v2	Synplify Pro RTL project files.
 /rtl_v3	Synplify Pro RTL project files.
 /rtl_v4	Synplify Pro RTL project files.
 /syn	Synplify project, log and output files (empty).


Figure 172 • Tutorial Directory Structure

Step 2: Set up the Synthesis Project

- Start the Synplify Pro GUI. For Linux:

```
% cd <your work area>/Speedcore_Incremental_Compile_RefDesign_RD012/synplify
% synplify_pro
```

For windows, double-click the Synplify Pro icon.

- Create a new project with () **Open Project** → **New Project** (or **File** → **New Project** in Windows). Windows users need to ensure that the project is saved to the chosen work area (**File** → **Save As**). To follow the directory structure used in this tutorial, use `<your_work_area>/Speedcore_Incremental_Compile_RefDesign_RD012/synplify`. The Synplify Pro home screen appears with an empty project named `proj_1` and an implementation named `rev_1`, as in the following screen shot:

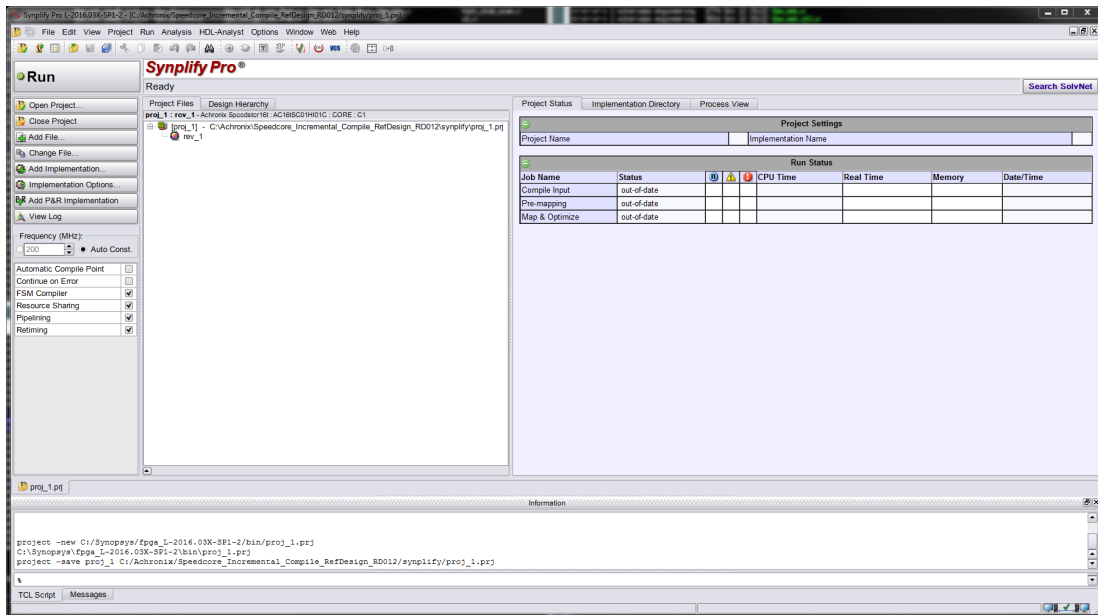


Figure 173 • Synplify Pro Home Screen

➕ Select **Project** → **Add Source File** to bring up the **Add Files to Project** dialog box. click the blue (↑) button to navigate up to the parent directory. In the "Files of type:" combo-box, select **All Files (*)**. Then double-click constraints to navigate to that directory and click the **<-Add All** button to add all of the constraint files. Click the blue up-arrow to navigate up one level and add all of the Verilog files in the rtl directory; click **OK**.

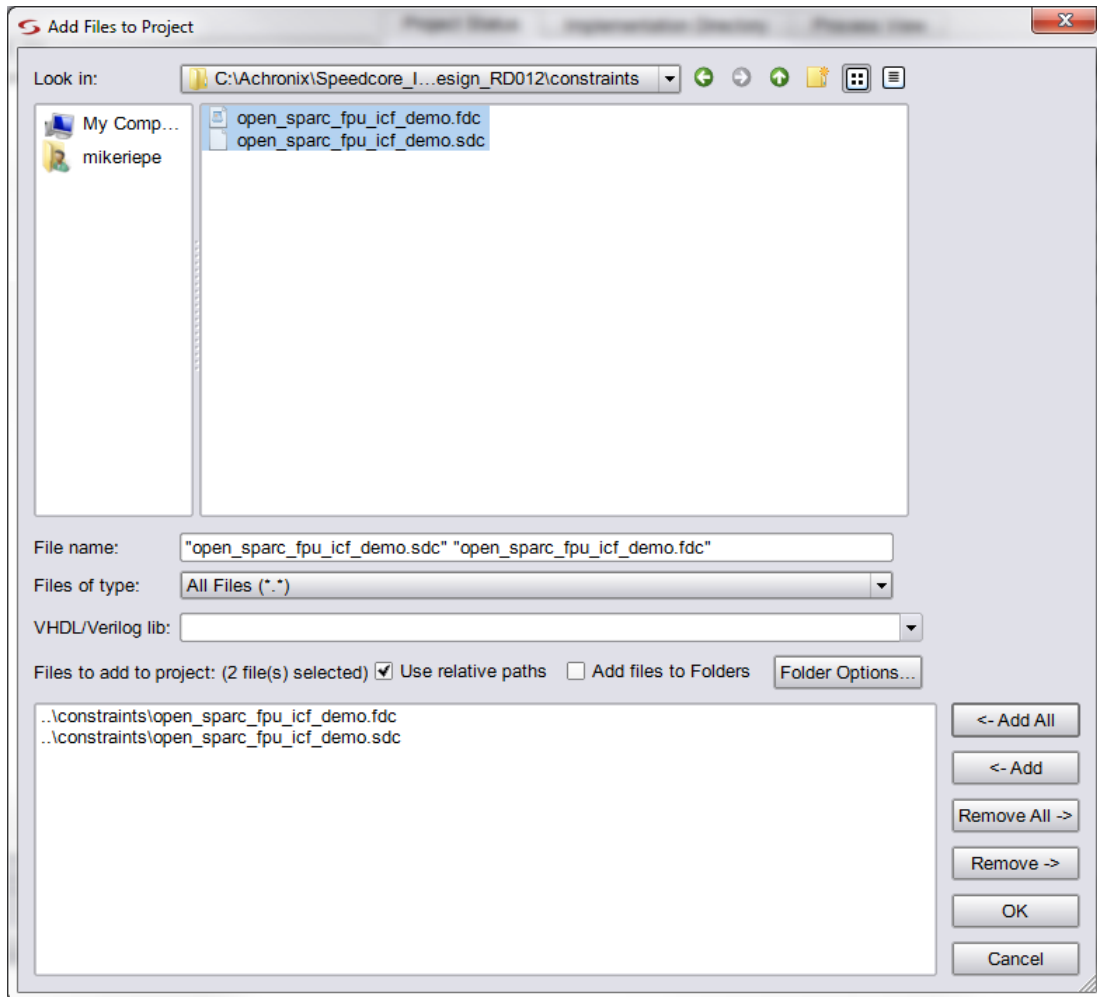


Figure 174 • Add Files to Project Dialog Box

- ✓ All of the files just added are now listed under the `proj_1` project in the Project Files tab (click **+** to expand each file type).
- ⊕ Ensure that the file `fpu_top.v` (the top-level module) appears last in the list of Verilog files. If it does not, correct the order by using the mouse to select the file name and drag it to the end of the list. For Windows users, the Result Base Name is set to the project name used earlier, the default being `Proj_1`. To have the file names match this document exactly, manually change the **Project** → **Implementation Options** → **Implementation Results** → **Result Base Name** to `"open_sparc_fpu_icf_demo"`, and click **OK**.

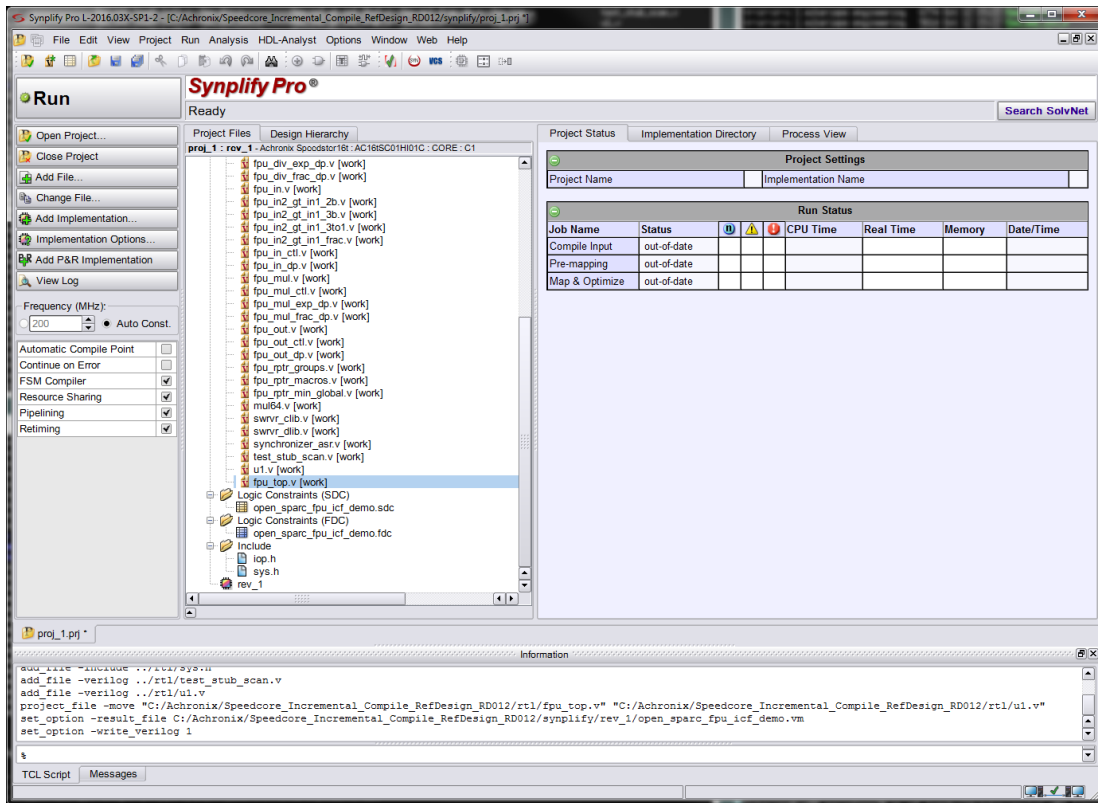


Figure 175 • Synplify Pro Home Screen After File Additions

⊕ Depending on how Synplify Pro is installed, the locations of the Achronix macro libraries may need to be specified. Open the Implementation Options window and then click the Verilog tab (**Project** → **Implementation Options** → **Verilog**). Ensure that *<ACE install location>/libraries* is present in the Include Path Order box. If not, add them by clicking the green + and navigating to the *<ACE install location>/libraries* directory. Click **Choose**, then click **OK**.

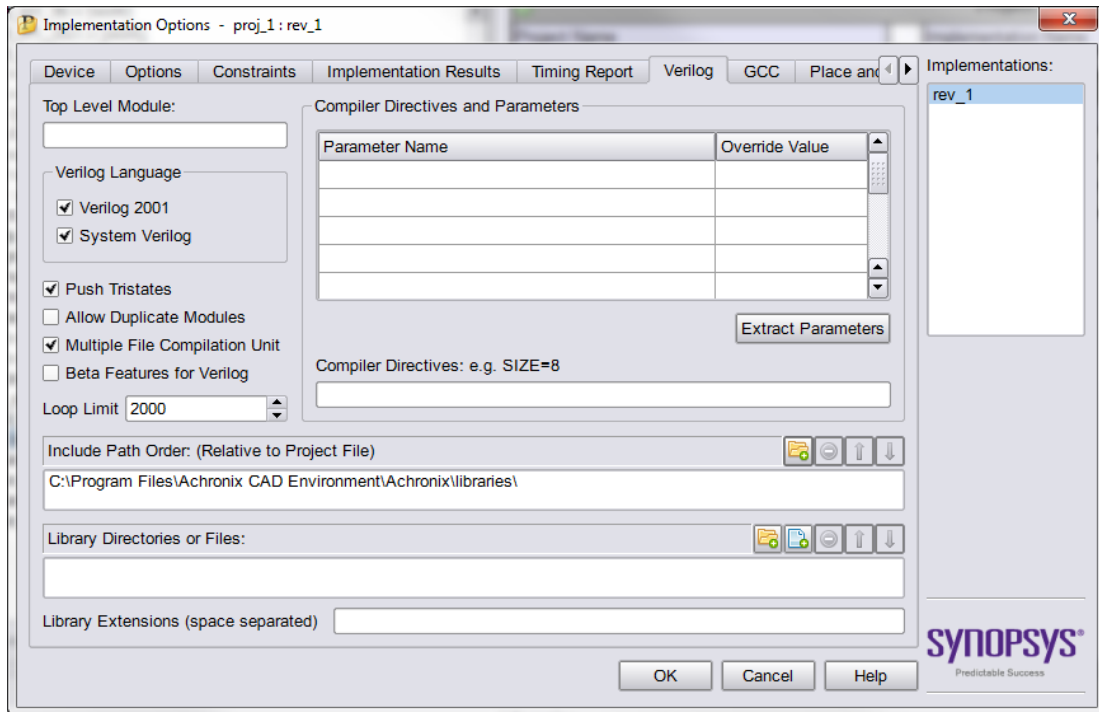


Figure 176 • Implementation Options Window

- Ensure that the Verilog file `<ACE install location>/libraries/device_models/16t_synplify.v` has been added to the project. If it has not, use the **Project** → **Add Source File** dialog box again to add it. Then drag this file to the top of the Verilog files listed to ensure that it is the first one read in.
- Finally, select the Achronix technology and part name. Open the Implementation Options Device tab (**Project** → **Implementation Options** → **Device**) and select the **Technology:** and **Part:** name.

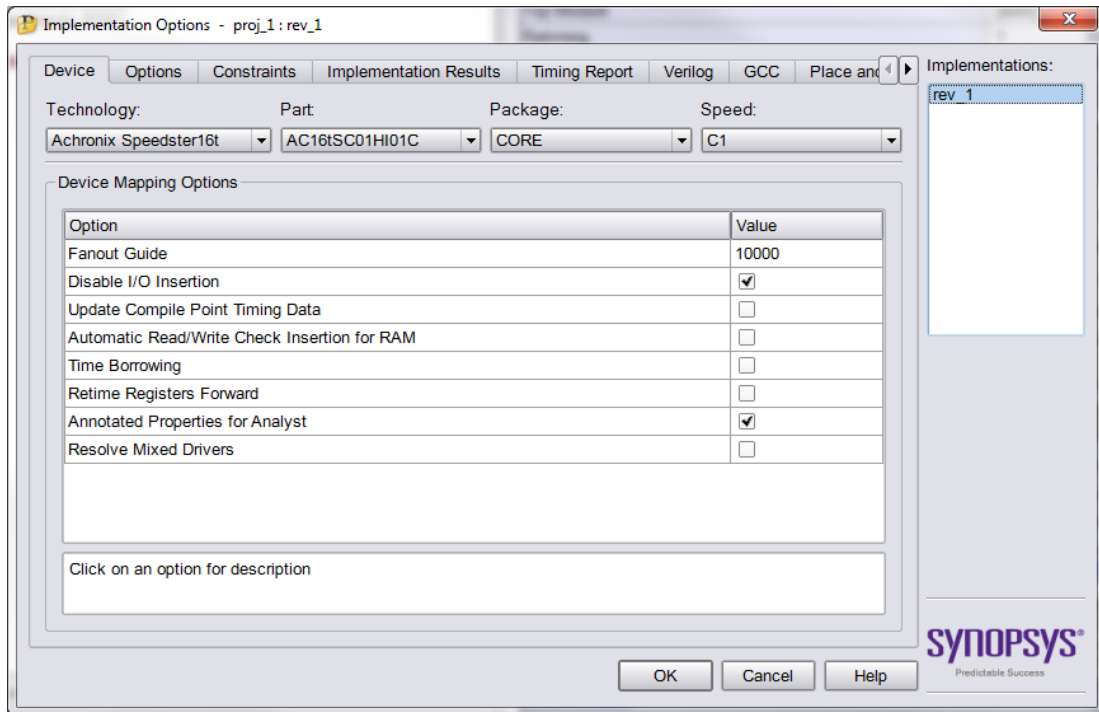


Figure 177 • Implementation Option Device Tab

Step 3: Compile the Design in Synplify Pro

➕ Select **Run** → **Compile Only** (or click **F7**) to parse the Verilog and constraints files and enable viewing. If this is the first time the design was compiled and the project has not been saved yet, Synplify Pro may ask to save the project file. Click **Save** to continue.

✔ In order to see the nine compile point constraints defined for this project, open the `constraints/open_sparc_fpu_icf_demo.fdc` file as a text file by navigating to the **Project Files** tab, expand the **Logic Constraints (FDC)** section, and left-mouse click the `open_sparc_fpu_icf_demo.fdc` file, and **Open as Text**. All nine of them are of type locked, as in the example below:

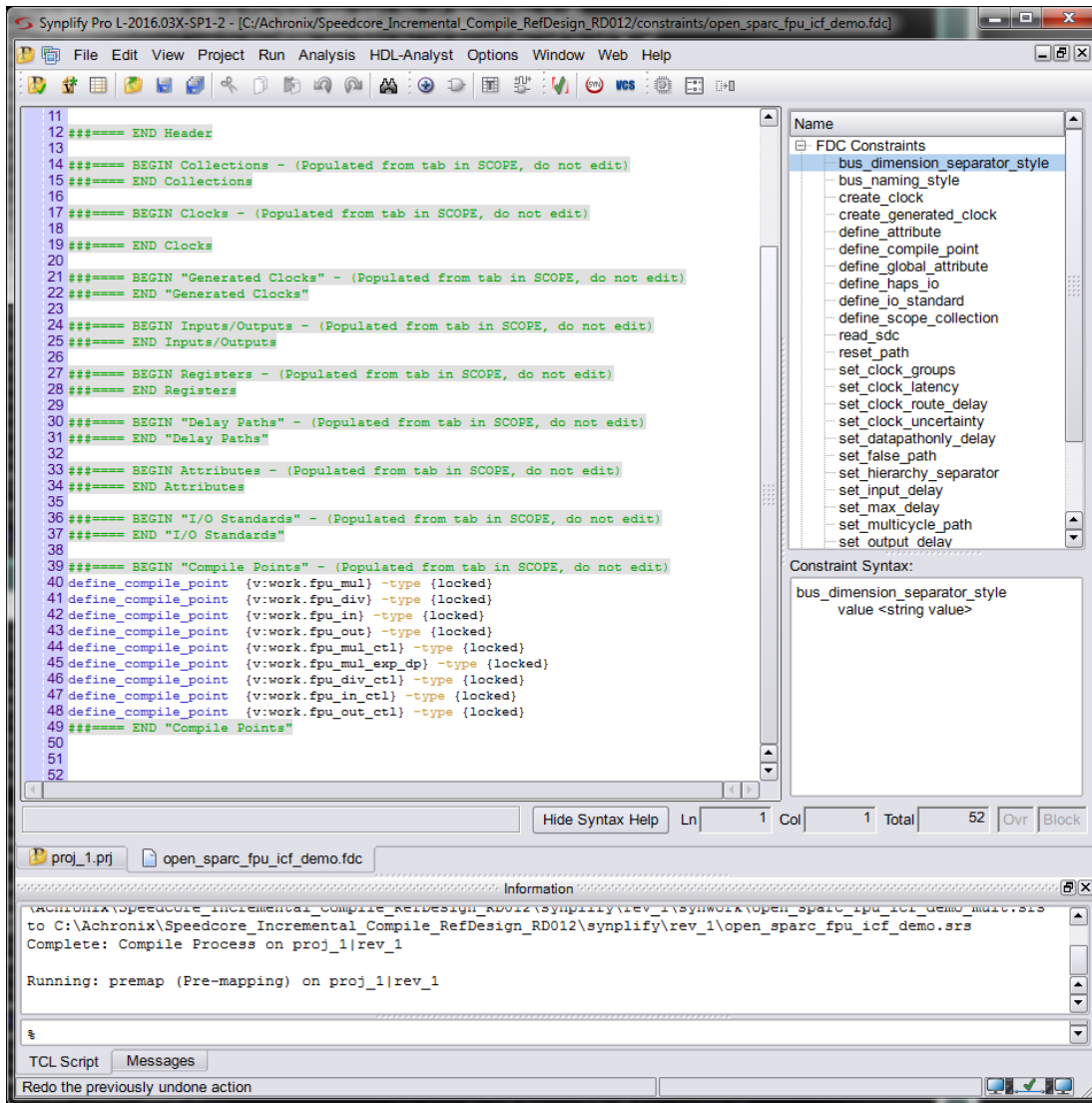



Figure 178 · Contents of the

Each compile point becomes a partition in the ACE tool. If one or more RTL source files are later edited and changed, Synplify Pro and ACE only need to recompile the partitions that have changed, rather than the whole design.

Optionally; instead of adding the `open_sparc_fpu_icf_demo.fdc` file to define the compile points, constraints can also be created or edited using the SCOPE tool. To manually add a new constraint file, click the () **New constraint file** button, and then click the **Compile Points** tab. To open the existing `open_sparc_fpu_icf_demo.fdc` file in the SCOPE tool, double-click the file name in the **Project Files** tab, and then click the **Compile Points** tab. Then, to add a new Compile Point, select the first blank row in the table, double-click in the View field to bring up a drop-down list of available view names, and select the one desired. Then double-click in the Type field to set the compile-point type. ACE treats all compile points as locked for purposes of placement and routing, but soft or hard compile points can be used in synthesis if locked results in poor QoR.

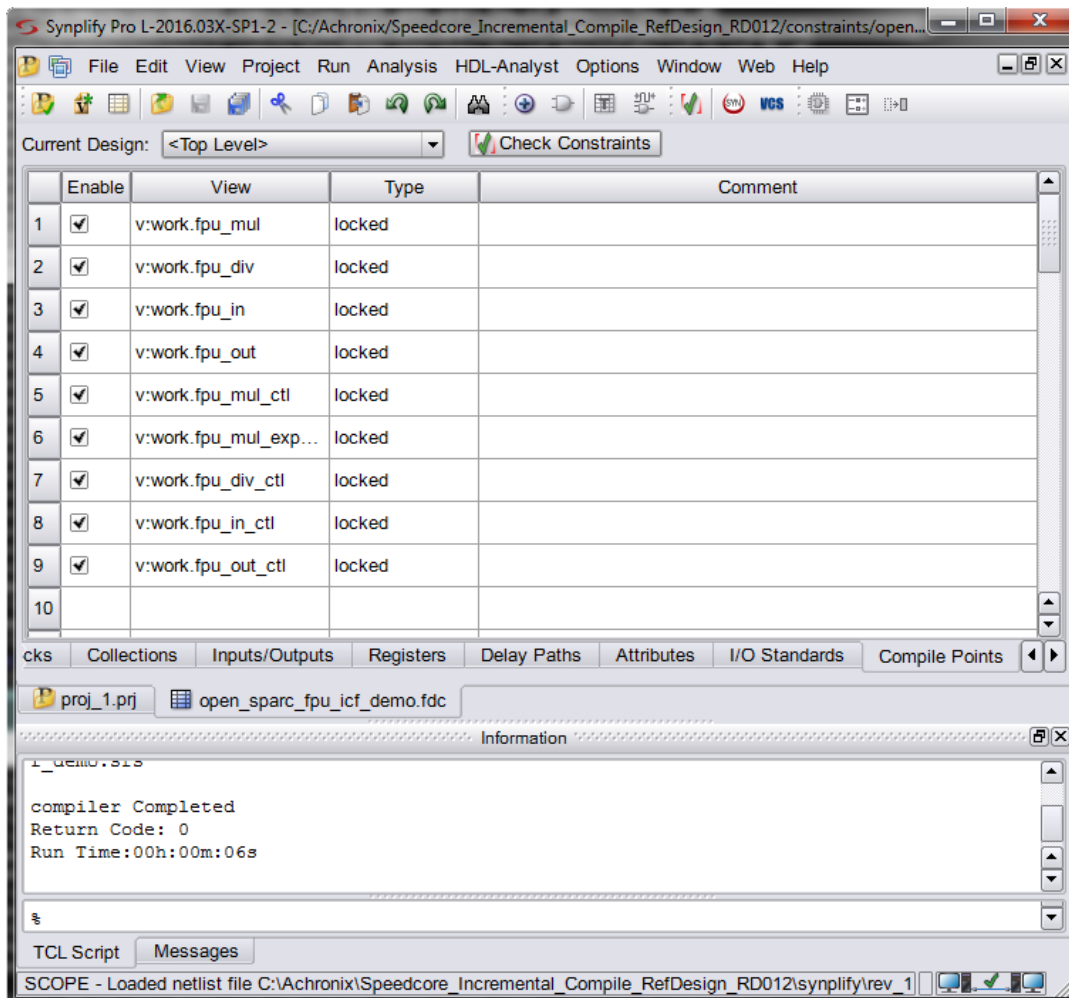


Figure 179 - Compile Points Tab Within the Synplify Pro SCOPE View

Close the SCOPE View window and do not save any changes.

Note

Synplify-Pro can be configured to create the compile points automatically. To experiment with this option, open the Implementation Options window, select the Options tab, and check the **Auto Compile Point** option. This option uses various heuristics (such as the sizes of the modules, the number of pins and the presence of timing constraints) to select a set of module views as compile points. These may be in addition to any compile points manually specified as constraints.

For this tutorial, ensure that the **Auto Compile Point** option is un-checked, then click **OK** to close the Implementation Options window.

+ Lastly, click **Run** to complete the mapping of the design.

For more information see Chapter 11, "Working with Compile Points" in the document *Synopsys FPGA Synthesis Synplify Pro for Achronix User Guide*, located in the Synplify Pro installation directory under `/doc`.

Caution!

Windows users may encounter an error with `m_generic.exe` while using compile points. This condition is caused by an issue with parallel synthesis jobs in the current version of Synplify Pro for Achronix. This situation is being addressed by Synopsys and is expected to be rectified in an upcoming release. If Synplify encounters this error, select **Options** → **Configure Compile Point Process**, change the "4" in the box to "1" as shown below.

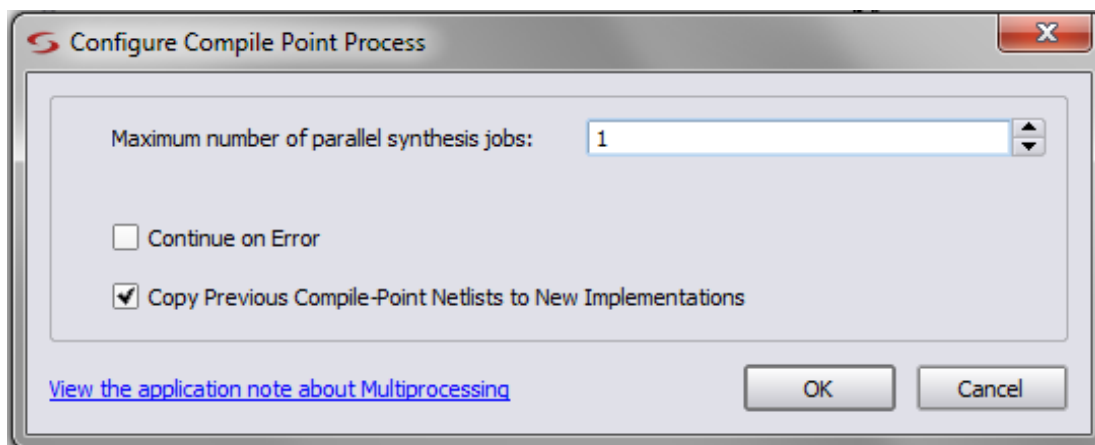


Figure 180 • Configure Compile Point Process Dialog Box

Step 4: Review Synplify Results

This step reviews some of the files and features available to better understand the behavior of Synplify Pro with compile-point constraints.

Synplify-Pro Log File

Using either the Synplify Pro GUI or another text editor, open the Synplify Pro log file `Speedcore_Incremental_Compile_RefDesign_RD012/synplify/rev_1/open_sparc_fpu_icf_demo.srr` and search for the section titled "Summary of Compile Points".

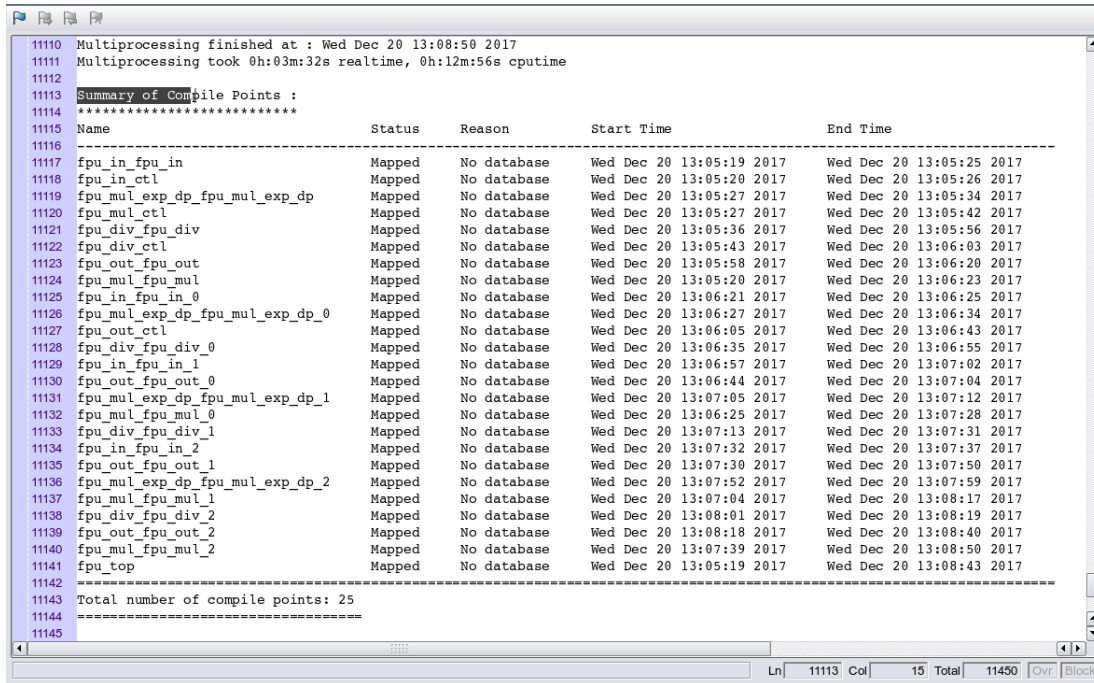


Figure 181 • Synplify-Pro Log File Showing Summary of Compile Points Section

The log file `rev_1/synlog/open_sparc_fpu_icf_demo_fpga_mapper.srr` lists a summary line for each of the defined compile points. Each compile point is an instance of the modules defined in the FDC file. All of these compile points are marked as **Mapped** (in this case "No database" because the design is being mapped for the first time). The timestamp of the last compile for each indicate they are all mapped at about the same time. Immediately below that section is a reference to a separate `.srr` log file file for each compile point.

Note

The "Summary of Compile Points" section may contain different **Name** entries than those that were defined in the FDC file. These can be instances of those modules.

The log file may contain the following warnings:

```
@N: MF104 :|Found compile point of type locked on View view:work.fpu_in_ctl(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_in(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul_ctl(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul_exp_dp(verilog)
@N: MF104 :|Found compile point of type locked on View view:work.fpu_mul(verilog)
```

```

@N: MF104 : |Found compile point of type locked on View view:work.fpu_div_ctl(verilog)
@N: MF104 : |Found compile point of type locked on View view:work.fpu_div(verilog)
@N: MF104 : |Found compile point of type locked on View view:work.fpu_out_ctl(verilog)
@N: MF104 : |Found compile point of type locked on View view:work.fpu_out(verilog)

```

These warnings are due to a caveat when using attributes with compile points. Attributes can be used when setting constraints for compile points. However, when using `syn_hier` on a compile point, the only valid value is `flatten`. All other values of this attribute (e.g., `hard`) are ignored for compile points. The `syn_hier` attribute behaves normally for all other module boundaries not defined as compile points.

ACE Partitioning Constraints File

✔ The file `synplyfy/rev_1/open_sparc_fpu_icf_demo_partition.prt` is written by Synplyfy for inclusion in the ACE project. This file contains TCL commands that define the Synplyfy-Pro compile points as partitions in ACE. Each command contains both the instance and view names of each partition, as well as its timestamp and compile-point type. There are many more partitions (37) listed in the `.prt` file than there were compile points listed in the Synplyfy log file because the partitions represent instances, while the compile points represent modules (many of the modules are instantiated multiple times in this design). The number of these instances match the number of the compile points found in the "Summary of Compile Points" section.


```

1 set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1513803914" -cp_type "hard"
2 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
3 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
4 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
5 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
6 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
7 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
8 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
9 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
10 set_partition_info -name "/fpu_top/fpu_inst[0].i.fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
11 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
12 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
13 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
14 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
15 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
16 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
17 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
18 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
19 set_partition_info -name "/fpu_top/fpu_inst[1].i.fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
20 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
21 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
22 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
23 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
24 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
25 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
26 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
27 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
28 set_partition_info -name "/fpu_top/fpu_inst[2].i.fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc
29 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803
30 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "l
31 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803
32 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "l
33 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1
34 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513803
35 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "l
36 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914
37 set_partition_info -name "/fpu_top/fpu_inst[3].i.fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "loc

```

Figure 182 • Contents of the `open_sparc_fpu_icf_demo_partition.prt` File

Technology View

➤ Click the  **Technology View** button to open the design schematic, then select one of the fpu_inst instances. These instances can be identified by expanding the Instances/Groups folder and then left-mouse click one to select it. The selected instance is highlighted with a red boundary in the Tech popup view.

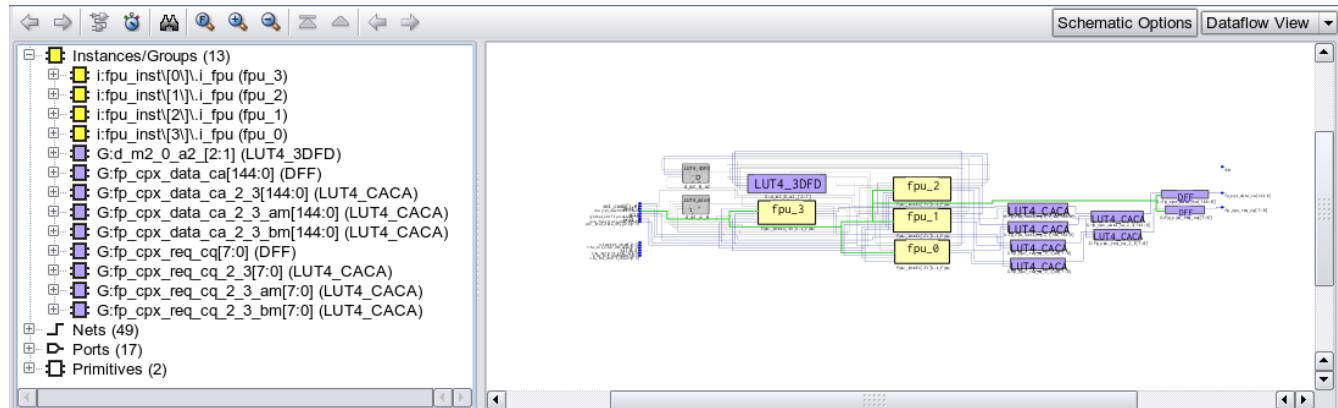


Figure 183 • Synplify Pro Initial Technology View Top-level Schematic, with no instance selected

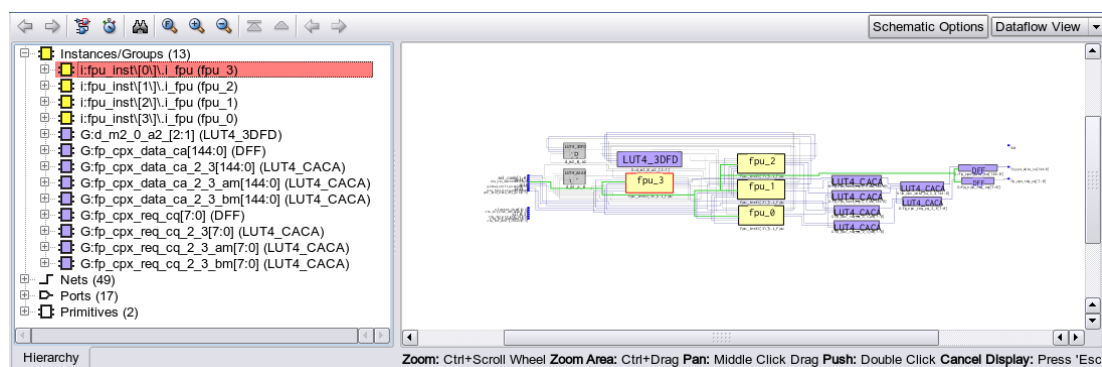


Figure 184 • Note how the selected instance (fpu_3) is highlighted in red

➤ Use the right mouse button to push into that level of the hierarchy. The schematic then updates. The locked and hard partitions have a green background color while the default instance background color is yellow. In the schematic area, use the right mouse button now to either push or pop hierarchy levels, depending on where the mouse is located when clicked.

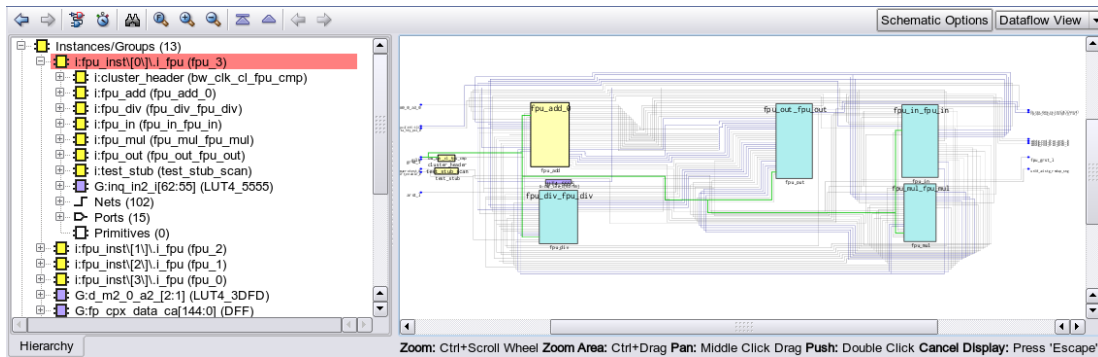


Figure 185 • Synplify Pro Technology View

- Exit from Synplify Pro. Next, the design must be placed and routed in ACE.

Step 5: Set up the ACE Project

- Start the ACE GUI. Under Linux, execute:

```
% cd <your work area>/Speedcore_Incremental_Compile_RefDesign_RD012/ace
% ace
```

Under Windows, double click the ACE icon.

- Then create a new project with **File** → **Create Project**. Click **Browse** to navigate through the filesystem to ensure that the project is created under the subdirectory `Speedcore_Incremental_Compile_RefDesign_RD012/ace` and click **OK**. Use `proj_1` for the project name, and `impl_1` for the implementation name. Click **Finish**.

The ACE home screen appears with an empty project named `proj_1` and an implementation named `impl_1`, as in the following screen shot:

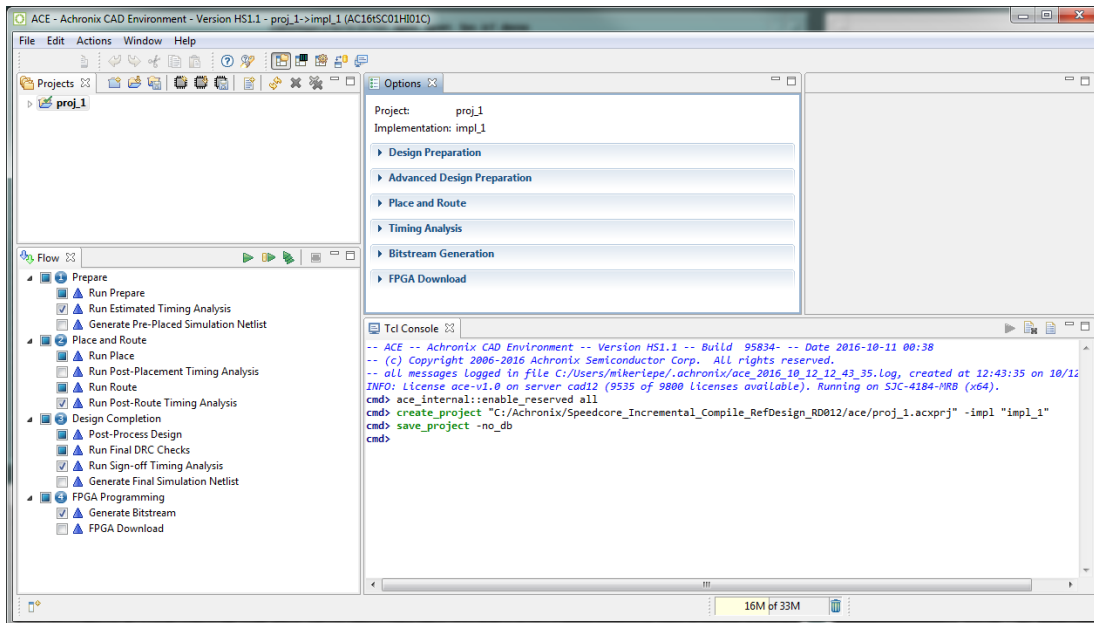


Figure 186 - ACE Home Screen

+ Select **File** → **Add Project Source Files...**, then click **Speedcore_Incremental_Compile_RefDesign_RD012** in the pathname bar to locate the source files and open the following dialog box:

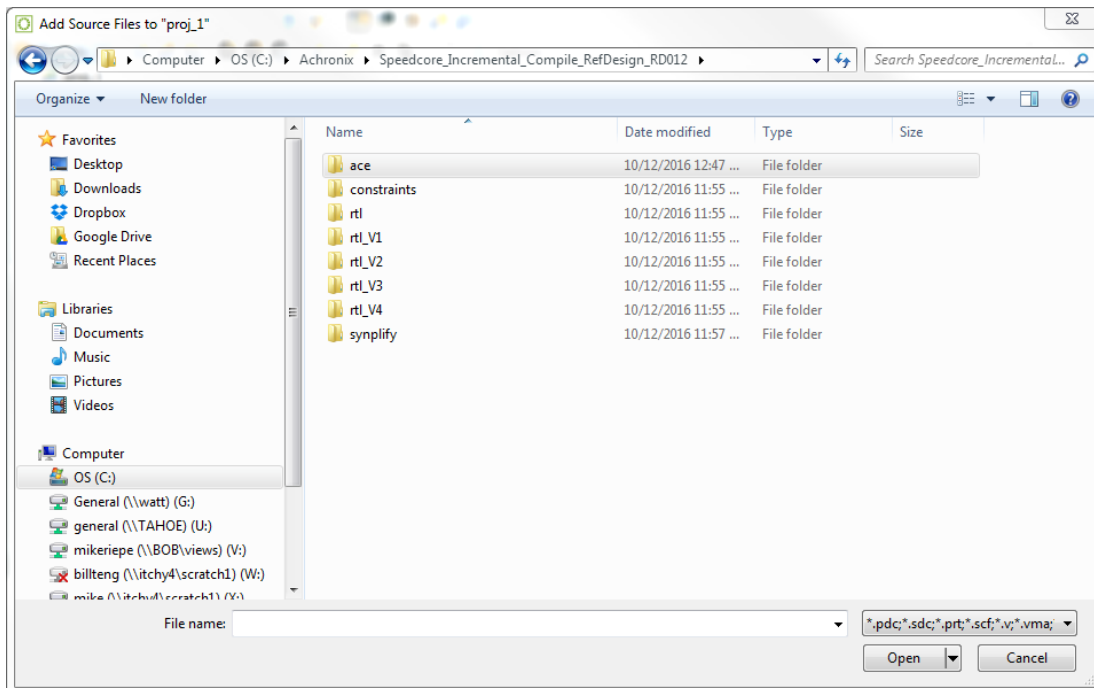


Figure 187 • Design Source Files Dialog

- ➕ Navigate to the constraints directory, select the `open_sparc_fpu_icf_demo.sdc` file, and then click **OK**. Bring up the Add Project Source Files dialog box again, navigate to the directory `synplify/rev_1/`, control-click to select the files `open_sparc_fpu_icf_demo.vm` and `open_sparc_fpu_icf_demo_partition.prt`, and click **OK** to add them to the project.
- ➕ Finally, in the Options tab under "Design Preparation", verify that the **Target Device** is the same device name used in Synplify, and verify that the **Enable Incremental Compile** implementation option checkbox is checked. All of the files just added appear under the `ace/Netlists` and `ace/Constraints` folders of the Projects tab.



Tip

In the **Projects Perspective** → **Projects View** click the triangle next to Constraints to list out the constraint files, etc.

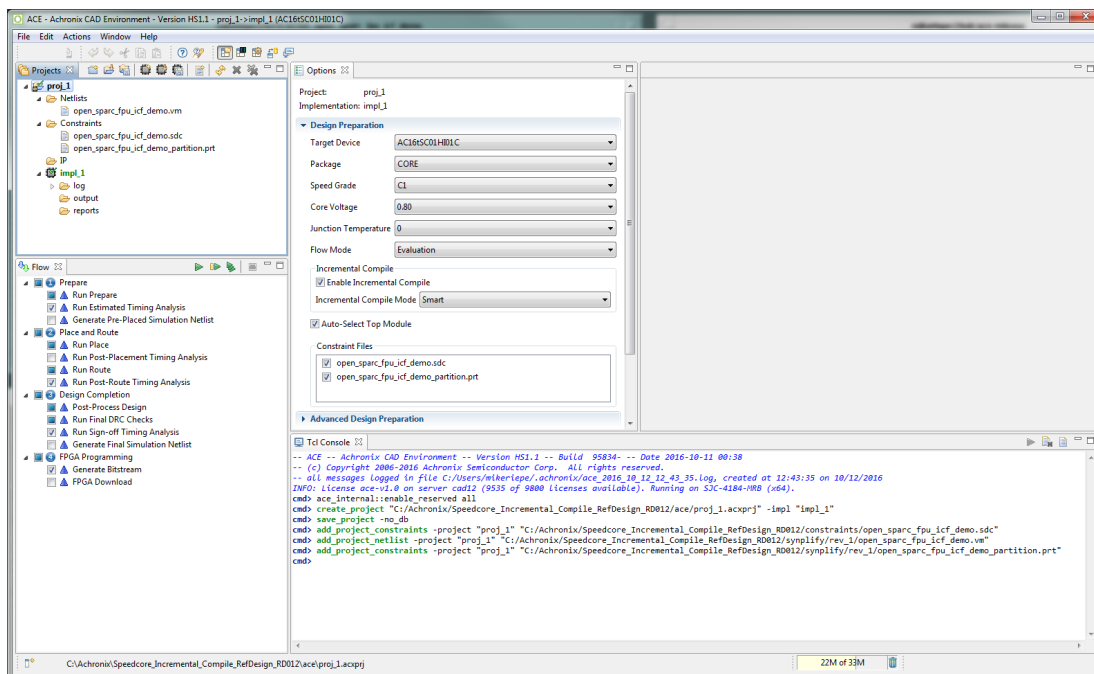


Figure 188 • ACE Projects Tab

- ✔ Recall from **Step 4** (page 418) that the `open_sparc_fpu_icf_demo_partition.prt` file (added to the project above) contains the partition definitions exported from Synplify Pro. The presence of this constraint file and the Enable Incremental Compile implementation option are the only configuration changes that distinguish the incremental compile flow from the standard non-incremental flow.
- ✔ Immediately under the **Enable Incremental Compile** implementation option checkbox is a drop-down box for the **Incremental Compile Mode** implementation option. Available values are strict and smart. Strict mode ensures that

placement of locked instances in unchanged partitions are completely preserved. Smart mode (the default) allows ACE to try to intelligently preserve placement in locked partitions for better design performance.

Step 6: Compile the Design in ACE

➕ In the Flow tab, uncheck the **Run Sign-off Timing Analysis** and **Generate Bitstream** flow step checkboxes to save some runtime. Then click the green triangle (▶) in the upper-right corner of the Flow view to run the prepare, placement, and routing flow. When the ACE flow completes, a green check mark appears by the **Run Final DRC Checks** flow step in the Flow view.

Step 7: Review ACE Results

Next is a review of some of the features available to help in understanding and optimizing the partition constraints.

Partition Report

✔ While the ACE flow is running, the Partition Report can be viewed at any time after the completion of the Run Prepare flow step. In the ACE GUI, the report opens automatically in the Editor Area of the Project perspective.

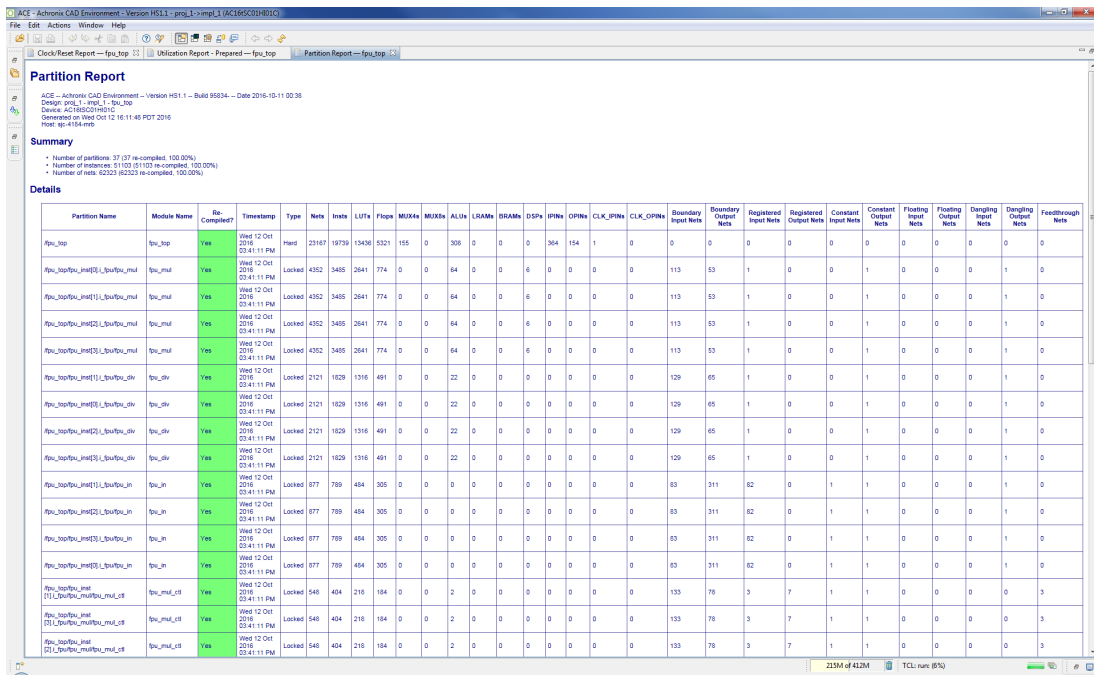


Figure 189 - ACE Partition Report After First Incremental Compile Iteration

✔ First look at the Summary section. The report shows the total number of partitions and the number that were recompiled (in this case 100% because this was the first pass through the flow). Also listed are the number of instances and nets that are owned by a partition, plus the number that were recompiled by ACE (again, 100% of each). Placement runtimes are proportional to the number of recompiled instances, and routing runtimes are proportional to the number of recompiled nets.

✔ The Details section displays a table with one row for each partition. Columns are printed as follows:

- Partition Name (which are the same as the instance name in the unflattened RTL)
- Module Name (derived from the module name in the RTL)
- Re-Compiled? column ("yes" or "no")
- Timestamp (time and date when it was last compiled in Synplify Pro)
- Type (only "Hard" and "Locked" are supported by ACE)
- The number of nets and instances owned by the partition
- A series of columns with instance counts for LUTs, Flops (DFFs), ALUs, LRAMs, BRAMs, etc.

The final eleven columns in the Details section provide information about the boundary nets of each partition. This information is useful in analyzing the suitability of each module for partitioning and to suggest ways in which the design may be improved to make it more amenable to partitioning. These include:

- Two columns counting the number of input nets and output nets crossing the boundary. These correspond to the ports of the original RTL module that have been flattened away. An input net has its driver outside the partition, while an output net has its driver inside the partition. The larger the ratio of boundary nets to instances in a partition, the more likely it is that the placement and routing of the partition is disturbed when neighboring partitions are recompiled (this is a corollary of Rent's Rule).
- Two columns with the number of input and output boundary nets that are registered. Registering boundary ports is always a good idea as it can be harder to maintain timing closure of cross-boundary paths when a partition or its parent needs to be recompiled.
- Two columns with the number of input and output boundary nets driven by a constant. Designers often tie-off unused inputs or outputs of a block and assume that those constants are optimized away by the logic synthesis tools. However, logic synthesis must assume that those constants may change in the future, so constant propagation cannot be performed across locked partition boundaries. Locking can result in a netlist that is much larger than expected. It is better to define a compile point on an RTL wrapper module that encloses input pin constants inside the partition and output-pin constants outside the partition. This method provides logic synthesis with the freedom to eliminate gates made redundant by the constants.
- Four columns with the number of input and output boundary nets that are floating and dangling, respectively. Floating nets have input pin loads with no driver, and dangling nets have a driver with no input pin loads. Similarly to constant boundary nets, logic synthesis is not able to optimize away floating and dangling logic across locked partition boundaries. Again, if a design has pins on a module that can logically be left unconnected, it is usually best to create a wrapper module so that unconnected inputs are enclosed within the partition and unconnected outputs are outside the partition, and then define the wrapper module as the partition instead.
- One column with the number of feedthrough nets. A feedthrough net enters an input pin of a module and exits through an output pin without driving any logic inside the partition. Again because logic synthesis cannot optimize logic across Locked partition boundaries, and it cannot eliminate pins on either Locked or Hard partition boundaries, it is best to eliminate feedthrough nets from the design. Feedthroughs impose constraints on synthesis, placement, and routing that can result in unnecessary delay and routing congestion.

Note

The table is sorted so that partitions with a recompiled state of "Yes" appear first, then sorted by number of instances.

Floorplan View

After the Routing flow step has completed, switch to the Floorplanner perspective to view the results. In the **Floorplanner View** (page 43) flyout palette, under the Layers section, turn on visibility for Instances, but turn off visibility for Sites, Clock Routes, and Non-clock Routes.

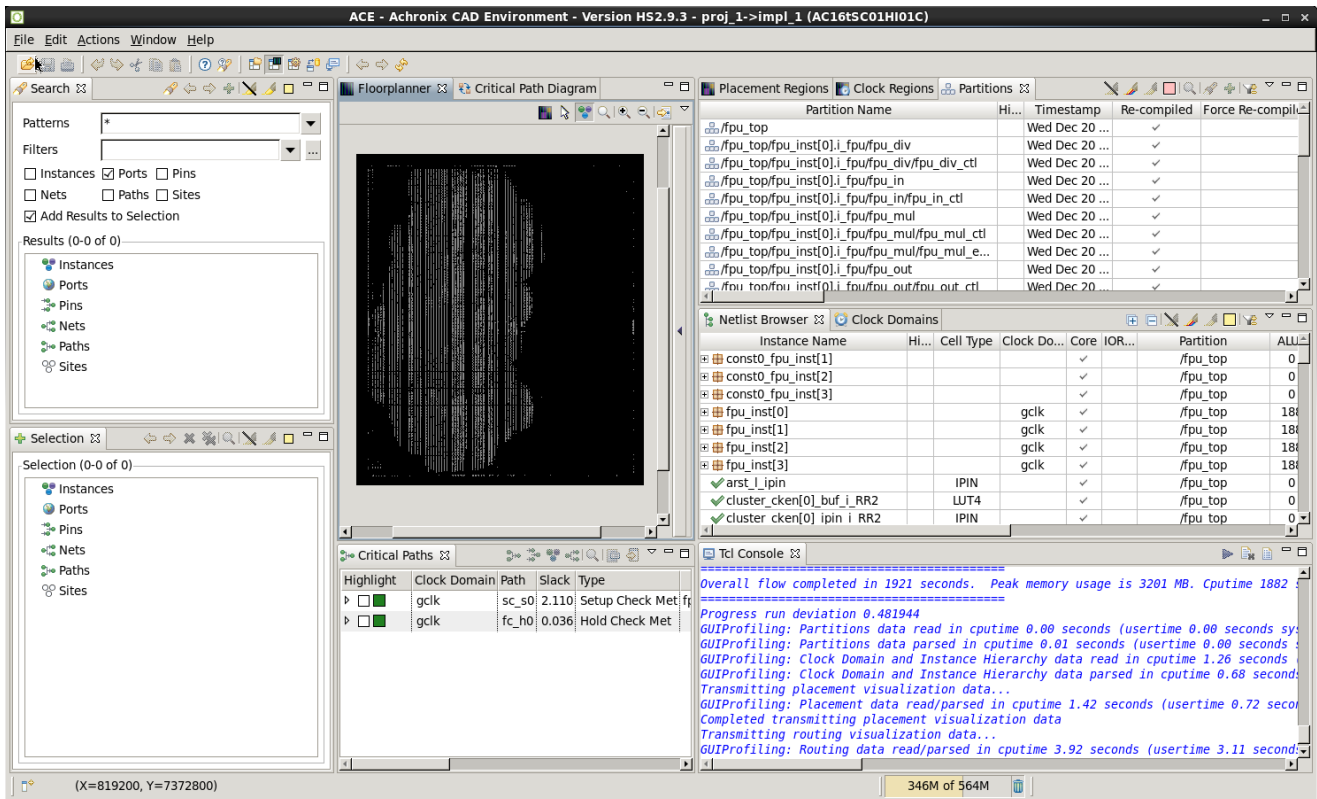






Figure 190 • ACE Floorplanner Perspective After First Incremental Compile Iteration

Switch to the **Partitions View** (page 107) to see a table with one row for each partition name. Columns exist for the Partition Timestamp; Re-Compiled status (a check-mark or not); the number of Flops, LUTs, ALUs, BRAMs, LRAMs, and Others (IOs, sources, etc); and the number of Cumulative Flops, LUTs, ALUs, BRAMs, LRAMs, and Others. The number of instances of each type includes only those instances directly owned by the given partition. The number of cumulative instances of each type includes instances owned by the given partition, as well as all child partitions below that partition in the RTL hierarchy.

The column named Force Recompile on Next Run provides a mechanism to override the partition timestamp during the *next* pass through ACE. Right-click anywhere in the row for a partition and select **Force Partition Changed**. A check mark appears in the Force Recompile on Next Run column, and the partition is re-placed and re-routed the next time the flow is run during this ACE session, even if there were no RTL changes nor recompilation in Synplify-Pro. Right-click again and select **Un-Force Partition Changed** to remove the check mark, or else the checkmark is removed automatically when the flow is run later in this ACE session.

- The column named Highlight Color displays a box with the partition highlight color. It should currently be empty. Right-click the row for a partition and select Highlight Partition to highlight the partition instances in the Floorplanner view with the current highlight color for the Partition tab. In the toolbar above the table, click the () Choose Highlight Color tool to change the highlight color to be used in the next Highlight command. Right-click the partition row and select **Un-Highlight Partition** to disable highlighting of the partition instances. Alternatively, the () **Highlight Partition** and () **Un-Highlight Partition** tools in the toolbar can be used to highlight the currently selected partition in the table.
- Finally, ensure that none of the partitions are highlighted, and click the () Auto-Highlight Partitions tool in the toolbar to highlight all of the partitions with an automatically selected color:

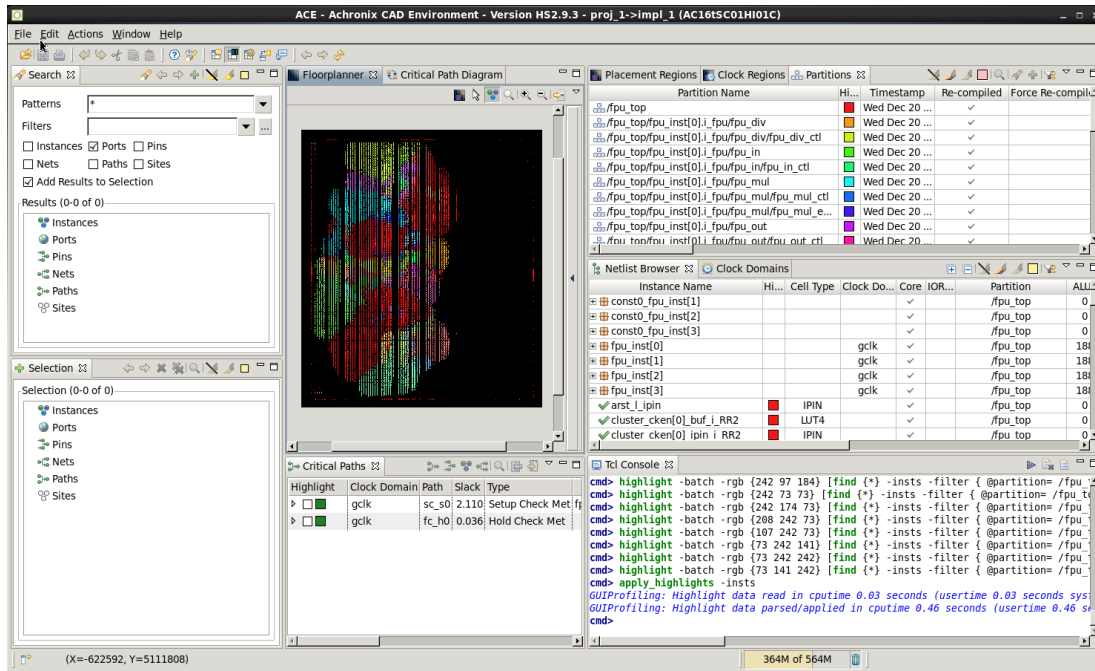






Figure 191 - Highlighting Partitions in ACE After First Incremental Compile Iteration

The  auto-highlight feature is an extremely powerful tool in understanding the logical and physical connectivity relationships between partitions. Generally, instances in a partition are placed in close proximity to each other if the connectivity within the partition is stronger than the connectivity outside (i.e., Rent's Rule: the number of ports on the partition being much smaller than the number of instances). The instances in one partition are generally placed close to the instances of other partitions with strong connectivity between them, and farther away from other partitions with weak connectivity. Specifically, if a partition has instances scattered over a wide area, it means that the instances are more strongly connected to other partitions than they are to each other. It may be better to remove that partition from the partition constraints. Placement and routing QoR may improve if that partition is recompiled every time the RTL is recompiled, allowing those instances to adapt to changes in their neighbors. This behavior may even be a sign that the RTL should be re-architected to absorb those glue logic instances into the top-level block or their neighboring instances.

Tools also exist in the toolbar to zoom (see [Zooming the Floorplanner In and Out \(page 339\)](#)) to the instances of the selected partition, search for the instances of the selected partition (this generates the appropriate Tcl [find \(page 638\)](#) command to populate the [Search View \(page 129\)](#)), and add the instances of the selected partition to the selection set (done with the Tcl [select \(page 708\)](#) command, with results displayed in the [Selection View \(page 133\)](#)).

➕ The partition table includes filtering functionality, which can be used to control visibility of the partition rows. To enable filtering, click the  **Toggle Filter Row Visibility** button. The filter row allows a content-appropriate filter to be applied to one or more table columns. Numeric columns filter based upon number ranges, while name columns filter based upon string matching or even regular expressions. For example, clicking the  filter icon above the LUTs column allows viewing only partitions with, for example, greater than 3,000 LUTs. This filter functionality is most useful when there are a large number of partitions.

✓ Tip

After using the  Placement Region Tool in the Floorplanner view to create a new placement region, if using partitions, drag-and-drop a row from the Partitions view onto the newly created placement region (either the appropriate row of the table in the [Placement Regions View \(page 110\)](#), or directly into the placement region painted in the Floorplanner view). This action generates the correct [add_region_find_insts \(page 613\)](#) command to add all instances in the dragged partition to the designated placement region.

Netlist Browser

✓ The [Netlist Browser View \(page 79\)](#) also contains several features dedicated to the Incremental Compile Flow. There is a column (Partition) naming which partition owns all instances represented by that row in the table. No partition name is given if the instances are not owned by a partition, or if they are split between two or more partitions. When a partition is highlighted in the Partitions view, that highlight color is also present in the Highlight Color column of the Netlist Browser in all rows owned by the given partition.

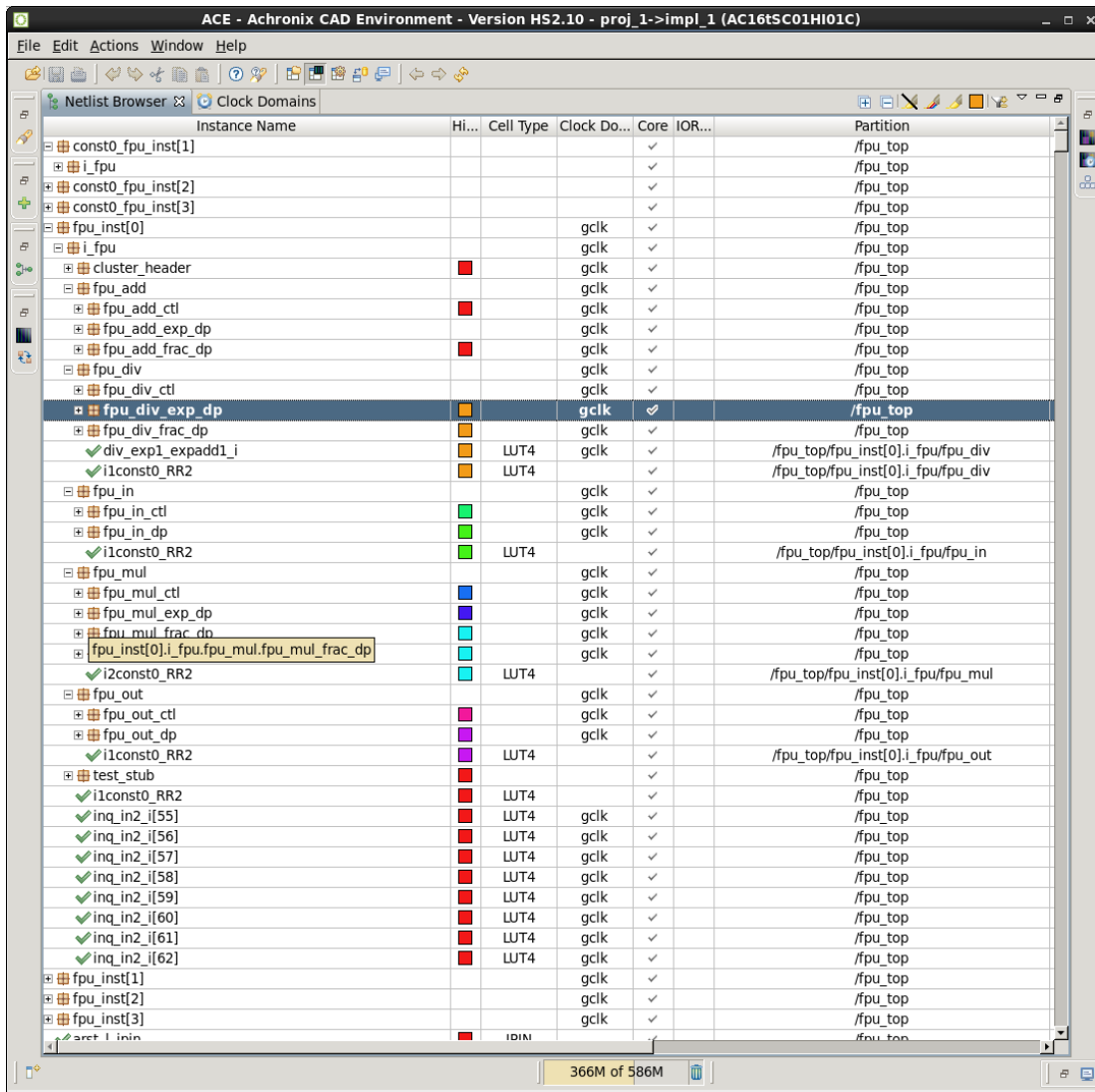


Figure 192 • ACE Netlist Browser

End of the First Pass of ACE Place and Route

- + At this point, exit out of ACE if desired.

Note

ACE can stay in memory and "incremental" runs can be implemented in the same ACE session because ACE recognizes that inputs to the ACE flows have changed, and runs incremental flows appropriately.

Step 8: Change the RTL (rtl_V1)

It is at this point where the utility of the Incremental Compile Flow begins to become apparent. The next steps simulate the actions of a product development team in the middle of a design iteration by modifying the RTL for one of the partitions, and then rerunning Synthesis, Prepare, Placement, and Routing (SPP&R).

- + First, navigate to the top-level directory of the tutorial project and start Synplify-Pro if it is no longer running. Under Linux, execute:

```
% cd <your work area>/ Speedcore_Incremental_Compile_Reference_Design_RD012
% synplify_pro
```

Under Windows, double-click the Synplify Pro icon.

- + This step simulates an RTL change by replacing the source file `rtl/fpu_mul_ctl.v` with the version in the directory `rtl_V1`. To do this in Synplify Pro, in the Project Files tab, navigate to the Verilog folder and left click the file to be changed to select it by clicking its name, in this case `fpu_mul_ctl.v`, and then right-click to bring up popup menu followed by **Change File...** or select **Project** → **Change File**. In the dialog box that appears, use the drop-down box of the Look in field and navigate from the directory `rtl` to the directory `rtl_V1`. Then double-click the file `fpu_mul_ctl.v`, or select it and click **OK**. The old version of the file is then removed from the project and replaced by the modified version. Diff the old and new versions of the file to see the changes:

```
$ diff rtl_V1/fpu_mul_ctl.v rtl
```

Note

A flop has been added to the 5-bit wire `mul_exc_out`.

Step 9: Recompile the Design in Synplify Pro (rtl_V1)

- + Click **Run** on the Synplify Pro home screen to recompile and remap the design.

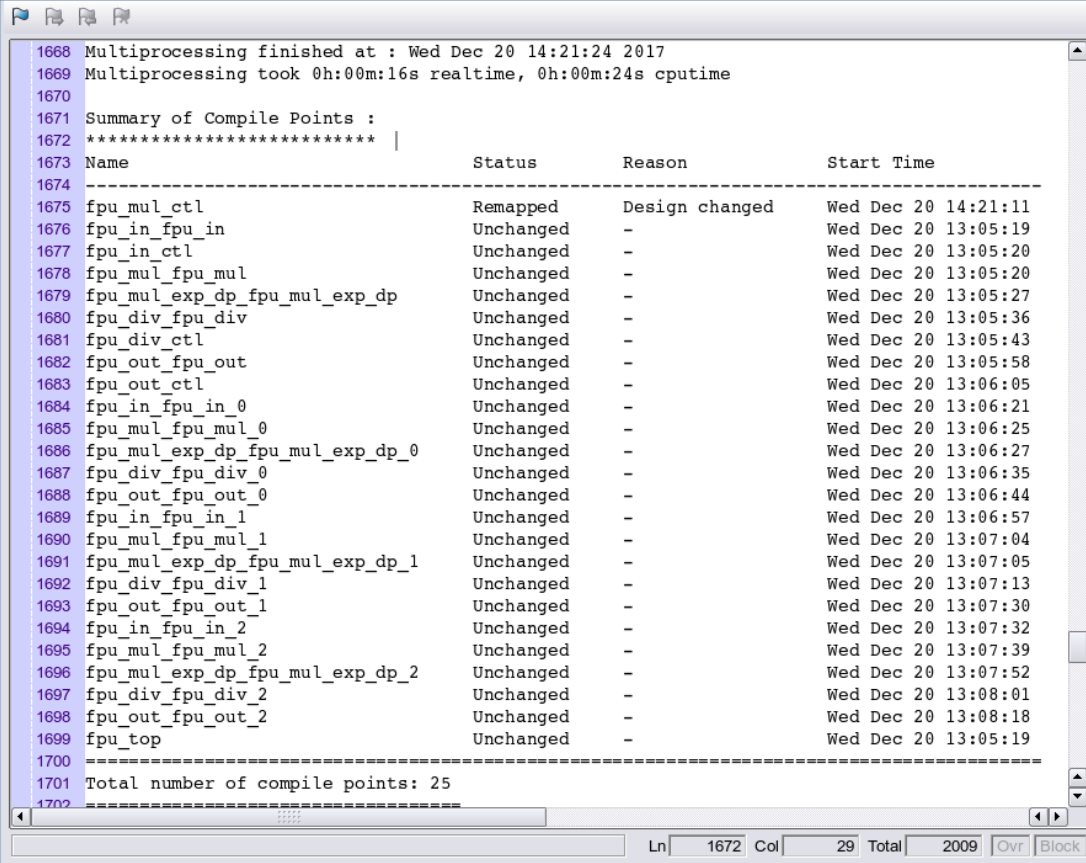
Note

Runtime is much faster than in the first iteration because only the changed module needs to be recompiled.

Step 10: Review Synplify Results (rtl_V1)

Synplify Pro Log File (rtl_v1)

Once again, open the Synplify Pro log file `Speedcore_Incremental_Compile_Reference_Design_RD012/synplify/rev_1/open_sparc_fpu_icf_demo.srr` and search for the section titled "Summary of Compile Points". All of the partitions have a status of "unchanged" except for the partition `fpu_mul_ctl` and its parent partition, which have status "remapped" and a reason of "design changed". The timestamp of the remapped partitions have also been advanced.



```

1668 Multiprocessing finished at : Wed Dec 20 14:21:24 2017
1669 Multiprocessing took 0h:00m:16s realtime, 0h:00m:24s cputime
1670
1671 Summary of Compile Points :
1672 *****
1673 Name                               Status      Reason      Start Time
1674 -----
1675 fpu_mul_ctl                          Remapped    Design changed  Wed Dec 20 14:21:11
1676 fpu_in_fpu_in                        Unchanged  -            Wed Dec 20 13:05:19
1677 fpu_in_ctl                            Unchanged  -            Wed Dec 20 13:05:20
1678 fpu_mul_fpu_mul                      Unchanged  -            Wed Dec 20 13:05:20
1679 fpu_mul_exp_dp_fpu_mul_exp_dp        Unchanged  -            Wed Dec 20 13:05:27
1680 fpu_div_fpu_div                      Unchanged  -            Wed Dec 20 13:05:36
1681 fpu_div_ctl                          Unchanged  -            Wed Dec 20 13:05:43
1682 fpu_out_fpu_out                      Unchanged  -            Wed Dec 20 13:05:58
1683 fpu_out_ctl                          Unchanged  -            Wed Dec 20 13:06:05
1684 fpu_in_fpu_in_0                     Unchanged  -            Wed Dec 20 13:06:21
1685 fpu_mul_fpu_mul_0                   Unchanged  -            Wed Dec 20 13:06:25
1686 fpu_mul_exp_dp_fpu_mul_exp_dp_0     Unchanged  -            Wed Dec 20 13:06:27
1687 fpu_div_fpu_div_0                   Unchanged  -            Wed Dec 20 13:06:35
1688 fpu_out_fpu_out_0                   Unchanged  -            Wed Dec 20 13:06:44
1689 fpu_in_fpu_in_1                     Unchanged  -            Wed Dec 20 13:06:57
1690 fpu_mul_fpu_mul_1                   Unchanged  -            Wed Dec 20 13:07:04
1691 fpu_mul_exp_dp_fpu_mul_exp_dp_1     Unchanged  -            Wed Dec 20 13:07:05
1692 fpu_div_fpu_div_1                   Unchanged  -            Wed Dec 20 13:07:13
1693 fpu_out_fpu_out_1                   Unchanged  -            Wed Dec 20 13:07:30
1694 fpu_in_fpu_in_2                     Unchanged  -            Wed Dec 20 13:07:32
1695 fpu_mul_fpu_mul_2                   Unchanged  -            Wed Dec 20 13:07:39
1696 fpu_mul_exp_dp_fpu_mul_exp_dp_2     Unchanged  -            Wed Dec 20 13:07:52
1697 fpu_div_fpu_div_2                   Unchanged  -            Wed Dec 20 13:08:01
1698 fpu_out_fpu_out_2                   Unchanged  -            Wed Dec 20 13:08:18
1699 fpu_top                              Unchanged  -            Wed Dec 20 13:05:19
1700 -----
1701 Total number of compile points: 25
1702 -----

```

Figure 193 • Synplify Pro Log File Showing Changed Compile Points

ACE Partitioning Constraints File (rtl_V1)

Also re-open file `Speedcore_Incremental_Compile_RefDesign_RD012/synplify/rev_1/open_sparc_fpu_icf_demo_partition.prt` and observe that the same changes are also reflected in the constraints file written out for ACE. Again, the timestamp has advanced when compared with the unmodified partitions.

```

1 set_partition_info -name "/fpu_top" -view "fpu_top" -timestamp "1513803914" -cp_type "hard"
2 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
3 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
4 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
5 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
6 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
7 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
8 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
9 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
10 set_partition_info -name "/fpu_top/fpu_inst[0].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
11 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
12 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
13 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
14 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
15 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
16 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
17 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
18 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
19 set_partition_info -name "/fpu_top/fpu_inst[1].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
20 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
21 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
22 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
23 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
24 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
25 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
26 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
27 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
28 set_partition_info -name "/fpu_top/fpu_inst[2].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"
29 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out/fpu_out_ctl" -view "fpu_out_ctl" -timestamp "1513803914" -cp_type "locked"
30 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_out" -view "fpu_out" -timestamp "1513803914" -cp_type "locked"
31 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div/fpu_div_ctl" -view "fpu_div_ctl" -timestamp "1513803914" -cp_type "locked"
32 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_div" -view "fpu_div" -timestamp "1513803914" -cp_type "locked"
33 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_exp_dp" -view "fpu_mul_exp_dp" -timestamp "1513803914" -cp_type "locked"
34 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul/fpu_mul_ctl" -view "fpu_mul_ctl" -timestamp "1513808464" -cp_type "locked"
35 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_mul" -view "fpu_mul" -timestamp "1513803914" -cp_type "locked"
36 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in/fpu_in_ctl" -view "fpu_in_ctl" -timestamp "1513803914" -cp_type "locked"
37 set_partition_info -name "/fpu_top/fpu_inst[3].i_fpu/fpu_in" -view "fpu_in" -timestamp "1513803914" -cp_type "locked"

```

Figure 194 • ACE Partitioning Constraints File: `open_sparc_fpu_icf_demo_partition.prt`

- Use **File** → **Close** to close Synplify Pro, and click **Save changes to project proj_1**.

Step 11: Recompile the Design in ACE (rtl_V1)

- If ACE was exited earlier, navigate to the same ace directory as before, and start ACE again. Otherwise, continue on from the current ACE session.
- Ensure that this tutorial project is the active project. Again, uncheck the **Run Sign-off Timing Analysis** and **Generate Bitstream** flow steps to save some runtime.
- Click the green triangle (▶) in the upper-right corner of the Flow view to rerun the Prepare, Placement, and Routing flow.

Note

The runtime of the flow is also significantly reduced over the first non-incremental compile.

In this second pass, ACE reads the new `open_sparc_fpu_icf_demo.vm` netlist file and the new `open_sparc_fpu_icf_demo_partition.prt` constraints file from the synplify directory. During the `run_prepare` flow step, ACE then executes an operation called Tear & Stitch. Each partition which has not been recompiled during synthesis is *torn* out of the database, and a copy from the previous pass is *stitched* back in. The copy from the previous run contains the complete set of placement and routing data. The placement of all stitched

instances are locked, and all routes are marked as preroutes to prevent their modification when the remainder of the netlist is placed and routed.

Step 12: Review ACE Results (rtl_V1)

Partition Report (rtl_V1)

✔ Maximize the Partition Report tab in the Editor Area of the Projects perspective. In the summary section, only 4 of the 37 partitions (10.81%) were recompiled by ACE, resulting in 3.05% of the instances being re-placed and 3.33% of the nets being rerouted. Also note in the Details section that only 4 fpu_mul_ctl partitions were recompiled, and their new timestamps are displayed. The counts of instances and nets in those partitions have changed by a small amount.

The screenshot shows the 'Partition Report' window in the ACE environment. The window title is 'ACE - Achronix CAD Environment - Version HS2.9.3 - proj_1->impl_1 (AC16t5C01H01C)'. The report is for 'fpu_top'. The summary section indicates that 37 partitions were recompiled (10.81%), 33647 instances were re-placed (3.05%), and 66245 nets were re-routed (3.33%). The details section contains a table with the following columns: Partition Name, Module Name, Re-Compiled?, Imported?, Top?, Leaf?, Timestamp, Type, Nets, Insts, LUTs, Flops, MUXs, ALUs, LRAMs, BRAMs, DSPs, IPNs, CLK_IPNs, CLK_OPNs, Boundary Input Nets, Boundary Output Nets, Registered Input Nets, Registered Output Nets, and Constant Input Nets.

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUXs	ALUs	LRAMs	BRAMs	DSPs	IPNs	CLK_IPNs	CLK_OPNs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	
fpu_top	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	133	78	3	7	1
fpu_top	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	133	78	3	7	1
fpu_top	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	133	78	3	7	1
fpu_top	fpu_mul_ctl	Yes	No	No	Yes	Wed 20 Dec 2017 02:21:04 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	133	78	3	7	1
fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0
fpu_top	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2633	772	0	0	69	0	0	9	0	0	0	113	53	1	0	0
fpu_top	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2633	772	0	0	69	0	0	9	0	0	0	113	53	1	0	0
fpu_top	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4898	3642	2792	772	0	0	69	0	0	9	0	0	0	113	53	1	0	0
fpu_top	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4868	3612	2762	772	0	0	69	0	0	9	0	0	0	113	53	1	0	0
fpu_top	fpu_div	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0	22	0	0	0	0	0	0	129	65	1	0	0

Figure 195 • ACE Partition Report After the rtl_V1 Incremental Compile Iteration

Floorplanner View (rtl_V1)

➕ Switch to the () Floorplanner perspective. In the Floorplanner view, compared to figure above (see figure 190) from the first iteration, about 3% fewer instances are now drawn with a locked fill color (dark yellow by default).

The locked placement state is just one of several potential placement states (see Instance States (page 262)) that an instance can have when painted in the Floorplanner. This locked placement state is somewhat similar in concept to the fixed placement state. (Instances with fixed placement status, shown in the Floorplanner with a light yellow fill color by default, are instances with user-assigned placement constraints, usually defined in a .pdc file. These fixed

instances are not allowed to be moved from their constrained placements during the flow.) The locked placement state indicates instances that are locked in place because they are in a partition that was not recompiled. Only instances with the default (or soft) placement state (light-grey fill color by default) were allowed to have their placement changed during the flow.

- ✓ The visibility of the instance locked placement state and the associated locked fill color can be chosen within the ACE GUI preferences. To see/edit these, select **Window** → **Preferences** → **Floorplanner View Colors**. In particular, ensure the **Instances** → **Show Locked Color on Instances with Locked Placement** box is checked, and that the **Locked Instances View Color** is set to the desired color (dark yellow by default).

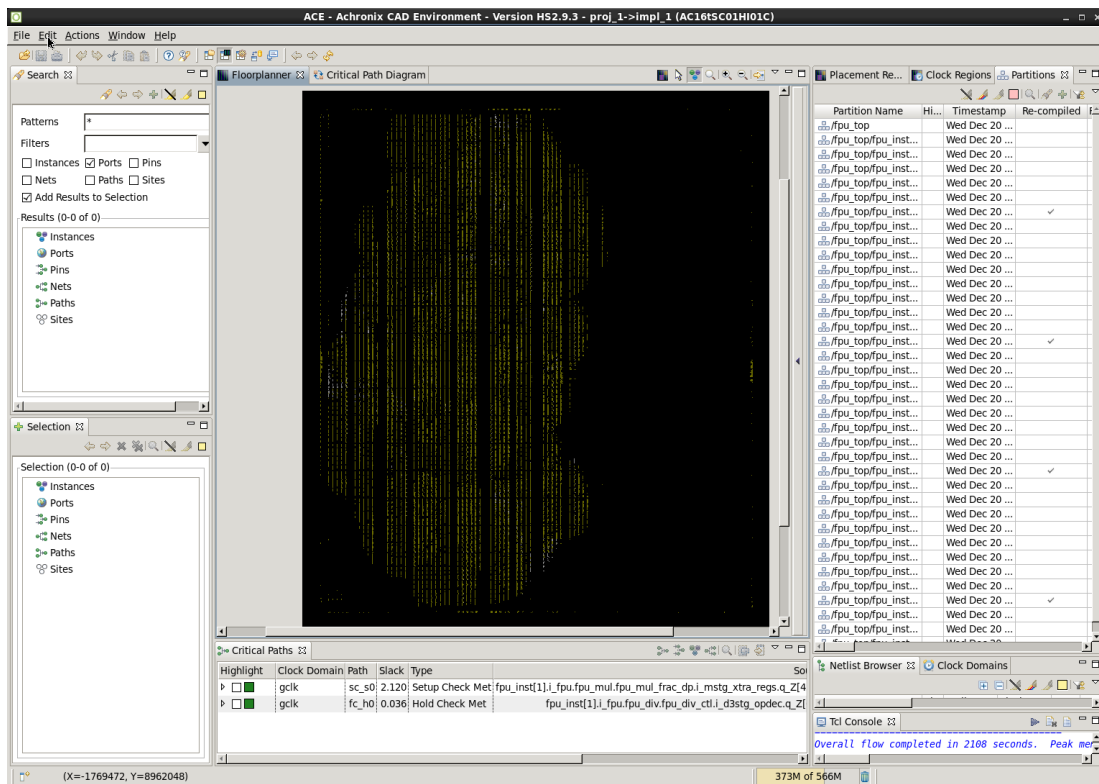


Figure 196 • ACE Floorplanner Perspective After the rtl_V1 Incremental Compile Iteration

➕ To get a feel for the amount of re-routing that was required, select (add to the ACE Selection Set) the instances in the partitions that were recompiled and then view the flylines representing the nets for those instances. To do this, check the **Selected Instance Flylines** box in the Floorplanner view fly-out palette. Then in the Partitions view, choose the rows for the partitions that were recompiled, and click (➕) **Add Instances to Selection** in the Partitions View toolbar. Blue flylines are drawn for the nets of the selected instances (or rather for the first 200 selected instances, since the selection set contains more than 200 objects). In the **Selection View** (page 133) click the gold left-arrow (←) and right-arrow (→) buttons in the view header to cycle visibility between different subsets of 200 of the selection set instances at a time.

Note

There are many more than 200 nets that are actually re-routed. However, by using the selection set to filter the visible connectivity in this manner, the amount of visible change to the design has been minimized to just the instances of the module that was changed.

This information is helpful in understanding the effect of any placement and routing changes during the second pass. Instances in a failing critical path with wayward placement could indicate that changes in the RTL were too extensive for effective incremental recompilation. This situation can occur when one of the partitions grows significantly in size and no longer fits in the area between locked neighboring partitions, for example. The placement for this recompiled partition may be squeezed into an undesirable aspect ratio, forcing long routing detours. In situations such as this, it may be best to force the entire design to be recompiled by enabling **Force Recompile on Next Run** for all of the partitions in the Partitions View.

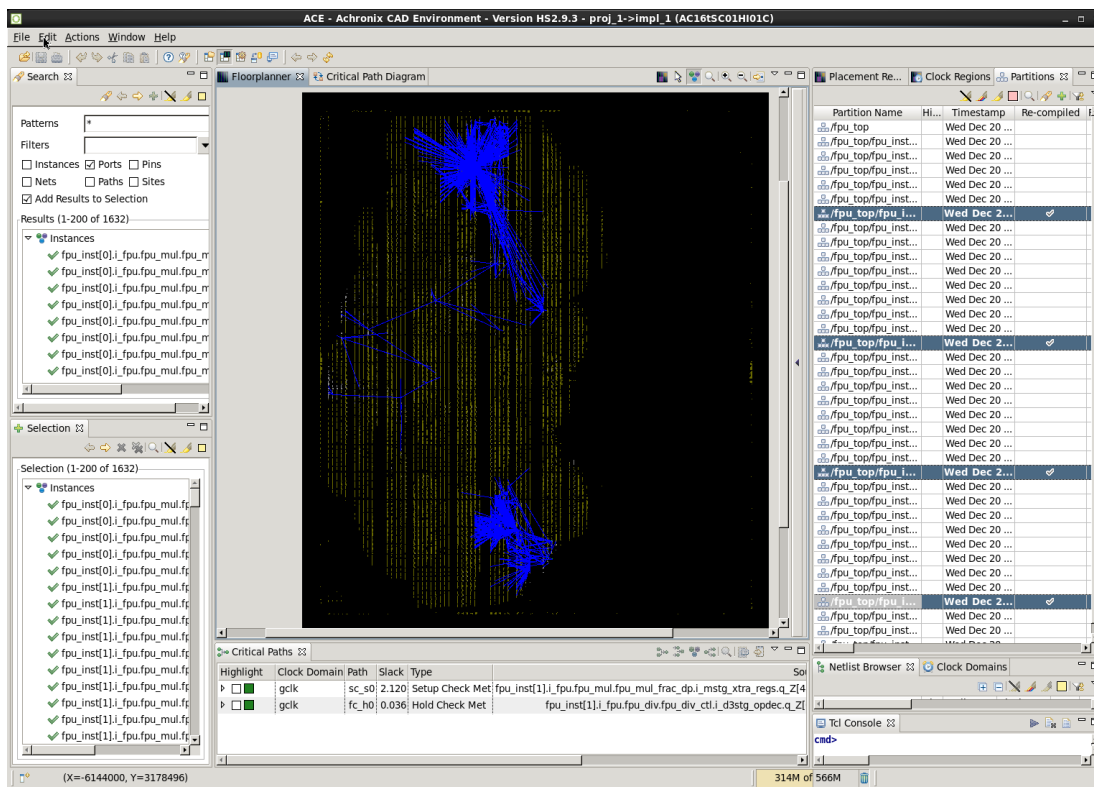





Figure 197 - ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V1)

Partitions View (rtl_V1)

✓ In the **Partitions View** (page 107) of the () Floorplanner Perspective, note that only 4 of the 37 partitions have a check mark in the Re-Compiled column.

Click () **Deselect all** in the Selection view to remove the routing flylines, and then click () **Auto-Highlight Partitions** in the Partitions View to observe any changes in the placement of the changed partitions. Unchanged

partitions can also be manually unhighlighted to only see (highlight) those that were not re-placed. This technique can help in understanding the effectiveness of incremental compilation on the design.

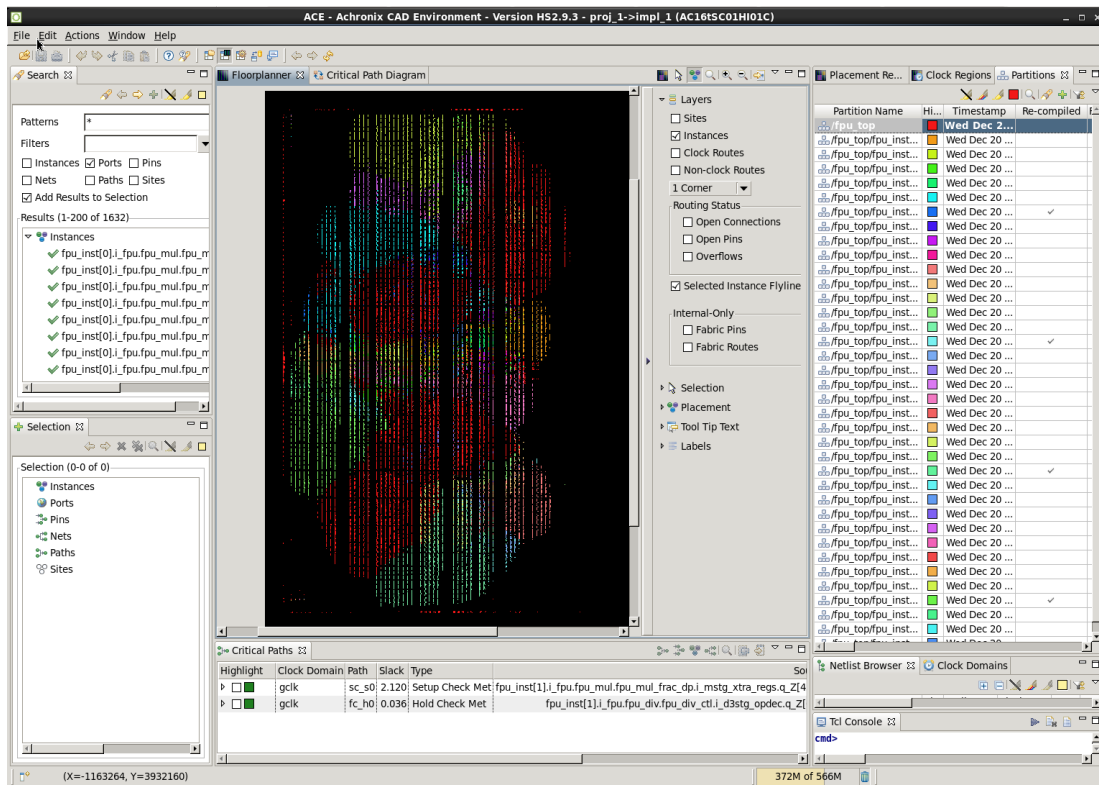


Figure 198 - Highlighting Partitions in ACE After Second Incremental Compile Iteration

Step 13: Additional Incremental Iterations

Steps 8–12 can be rerun with additional RTL changes if desired. This tutorial design contains additional directories with additional RTL change examples. Or modify the existing RTL files by adding or deleting module pins, add or remove partitions from the .fdc file, or rename module instances, to further explore the incremental compile flow.

Table 147 - Additional Tutorial Examples

Directory	Changes
rtl_v1	Flopped the signal mul_exc_out in fpu_mul_ctl.v
rtl_v2	Reverts the changes from rtl_v1, adds a new 6-bit counter in place of a constant
rtl_v3	Modifies the partition fpu_div_ctl by adding an enable check
rtl_v4	Modifies the top-level partition fpu.v by inverting one net

Step 14: Review ACE Results (rtl_V2)

After rerunning the synplify_pro and ace flows, review the perturbation of the design in ace by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V2)

✔ Maximize the Partition Report tab in the Editor Area of the Projects perspective. In the summary section, only 4 of the 37 partitions (10.81%) were recompiled by ACE, resulting in 3.12% of the instances being re-placed and 3.54% of the nets being rerouted. Also note in the Details section that only 4 fpu_mul_ctl partitions were recompiled, and their new timestamps are displayed. The counts of instances and nets in those partitions have changed by a small amount.

ACE - Achronix CAD Environment - Version HS2.9.3 - proj_1->impl_1 (AC16TSC01H01C)

File Edit Actions Window Help

Clock/Reset Report - fpu_top Utilization Report - Routed - fpu_top Power Dissipation Report - fpu_top Timing Report - Routed - fpu_top Partition Report - fpu_top

Partition Report

ACE - Achronix CAD Environment - Version HS2.9.3 - Build 110956 - Date 2017-12-07 18:46
 Design: fpu_top - impl_1 - fpu_top
 Device: AC16TSC01H01C
 Generated on Thu Dec 21 07:45:10 PST 2017
 Host: f62.achronix.local

Summary

- Number of partitions: 37 (4 re-compiled, 10.81%)
- Number of instances: 53483 (1668 re-compiled, 3.12%)
- Number of nets: 66386 (2348 re-compiled, 3.54%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPNs	OPNs	CLK_IPNs	CLK_OPNs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Const Input Nets	
fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0	
fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4998	3642	2792	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4868	3612	2762	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0	22	0	0	0	0	0	0	0	0	129	65	1	0	0

388M of 573M

Figure 199 - ACE Partition Report After the rtl_V2 Incremental Compile Iteration

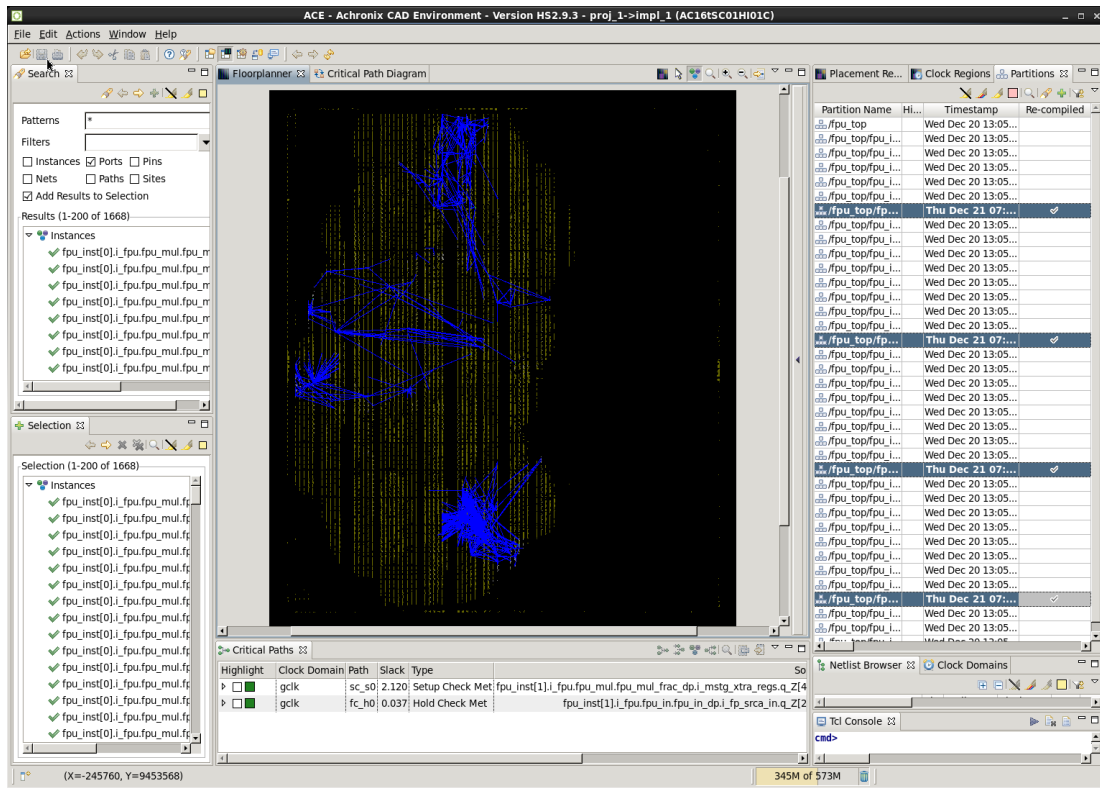


Figure 201 · ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V2)

Step 15: Review ACE Results (rtl_V3)

After rerunning the synplify_pro and ace flows, review the perturbation of the design in ace by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V3)

ACE - Achronix CAD Environment - Version HS2.9.3 - proj_1->impl_1 (AC16t5C01H101C)

Partition Report — fpu_top ⌵ Clock/Reset Report — fpu_top ⌵ Utilization Report - Routed — fpu_top ⌵ Power Dissipation Report — fpu_top ⌵ Timing Report - Routed — fpu_top ⌵

Partition Report

ACE - Achronix CAD Environment - Version HS2.9.3 - Build 110956 - Date 2017-12-07 18:46
 Design: fpu_1 - impl_1 - fpu_top
 Device: AC16t5C01H101C
 Generated on Thu Dec 21 08:48:05 PST 2017
 Host: 623.achronix.local

Summary

- Number of partitions: 37 (6 re-compiled, 21.62%)
- Number of instances: 53478 (1504 re-compiled, 2.81%)
- Number of nets: 65381 (1388 re-compiled, 2.99%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUXs	MUXBs	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Consta Input Nets	
fpu_top fpu_top[0]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	0	73	80	3	0	0
fpu_top fpu_top[1]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	0	73	80	3	0	0
fpu_top fpu_top[2]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	0	73	80	3	0	0
fpu_top fpu_top[3]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	415	330	209	119	0	0	2	0	0	0	0	0	0	0	0	73	80	3	0	0
fpu_top fpu_top[4]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[5]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[6]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[7]_fpu fpu_top fpu_top	fpu_top	Yes	No	No	Yes	Thu 21 Dec 2017 08:32:32 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0	0
fpu_top fpu_top[8]_fpu fpu_top	fpu_top	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0

300M of 559M

Figure 202 - ACE Partition Report After the rtl_V3 Incremental Compile Iteration

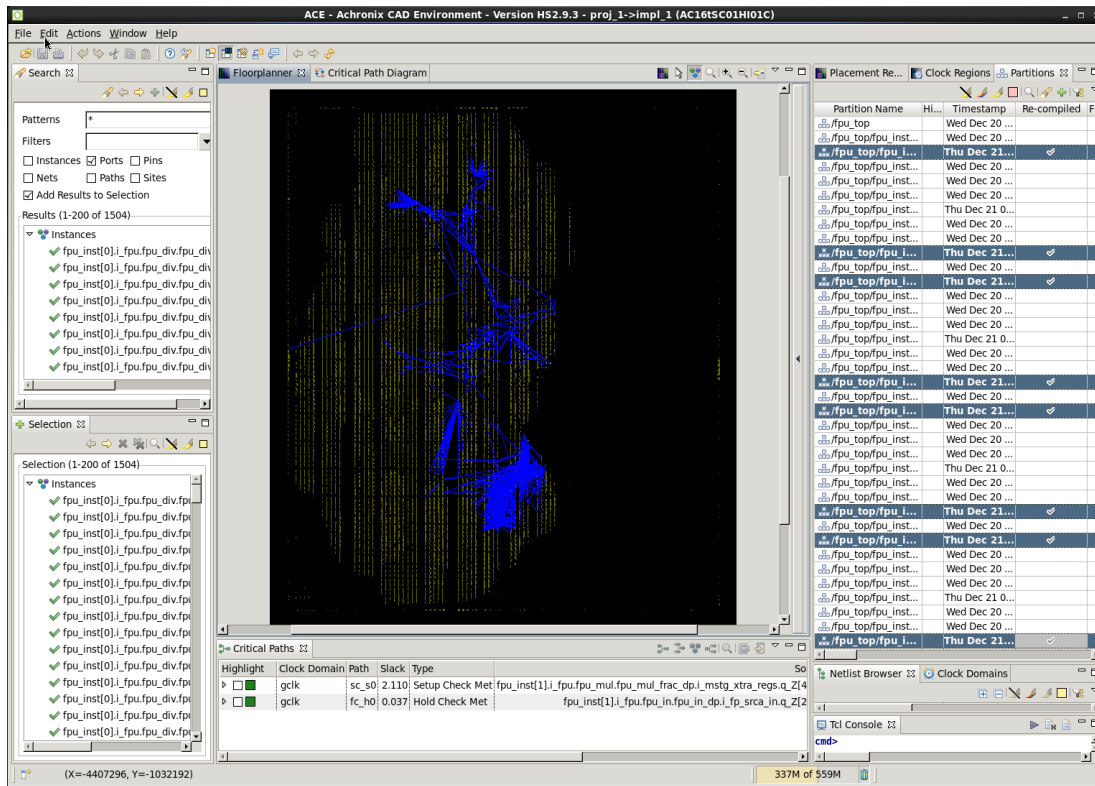


Figure 204 - ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V3)

Step 16: Review ACE Results (rtl_V4)

After rerunning the synplify_pro and ACE flows, review the perturbation of the design in ACE by looking at the Partition Report and the Floorplanner views with and without flylines.

Partition Report (rtl_V4)

ACE - Achronix CAD Environment - Version HS2.9.3 - proj_1->impl_1 (AC16TSC01H101C)

File Edit Actions Window Help

Clock/Reset Report - fpu_top Utilization Report - Routed - fpu_top Power Dissipation Report - fpu_top Timing Report - Routed - fpu_top Partition Report - fpu_top

Partition Report

ACE - Achronix CAD Environment - Version HS2.9.3 - Build 110956 - Date 2017-12-07 18:46
 Design: proj_1 - impl_1 - fpu_top
 Device: AC16TSC01H101C
 Generated on Thu Dec 21 09:37:43 PST 2017
 Host: h023.achronix.local

Summary

- Number of partitions: 37 (4 re-compiled, 10.81%)
- Number of instances: 53479 (184 re-compiled, 0.34%)
- Number of nets: 66381 (328 re-compiled, 0.49%)

Details

Partition Name	Module Name	Re-Compiled?	Imported?	Top?	Leaf?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX2s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Const Input Nets	
fpu_top fpu_top[0]_fpu fpu_out fpu_out_cfl	fpu_out_cfl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[1]_fpu fpu_out fpu_out_cfl	fpu_out_cfl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[2]_fpu fpu_out fpu_out_cfl	fpu_out_cfl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top fpu_top[3]_fpu fpu_out fpu_out_cfl	fpu_out_cfl	Yes	No	No	Yes	Thu 21 Dec 2017 09:30:41 AM	Locked	82	46	28	18	0	0	0	0	0	0	0	0	0	0	0	36	17	3	0	0
fpu_top	fpu_top	No	No	Yes	No	Wed 20 Dec 2017 01:05:14 PM	Hard	24704	21254	13675	6426	154	0	308	0	0	0	536	154	1	0	0	0	0	0	0	0
fpu_top fpu_top[0]_fpu fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top fpu_top[1]_fpu fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4939	3683	2833	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top fpu_top[2]_fpu fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4896	3642	2792	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top fpu_top[3]_fpu fpu_mul	fpu_mul	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	4866	3612	2762	772	0	0	69	0	0	9	0	0	0	0	0	113	53	1	0	0
fpu_top fpu_top[0]_fpu fpu_div	fpu_div	No	No	No	No	Wed 20 Dec 2017 01:05:14 PM	Locked	2141	1849	1285	542	0	0	22	0	0	0	0	0	0	0	0	129	65	1	0	0

334M of 524M

Figure 205 • ACE Partition Report After rtl_V4 Incremental Compile Iteration

Floorplanner View (rtl_V4)

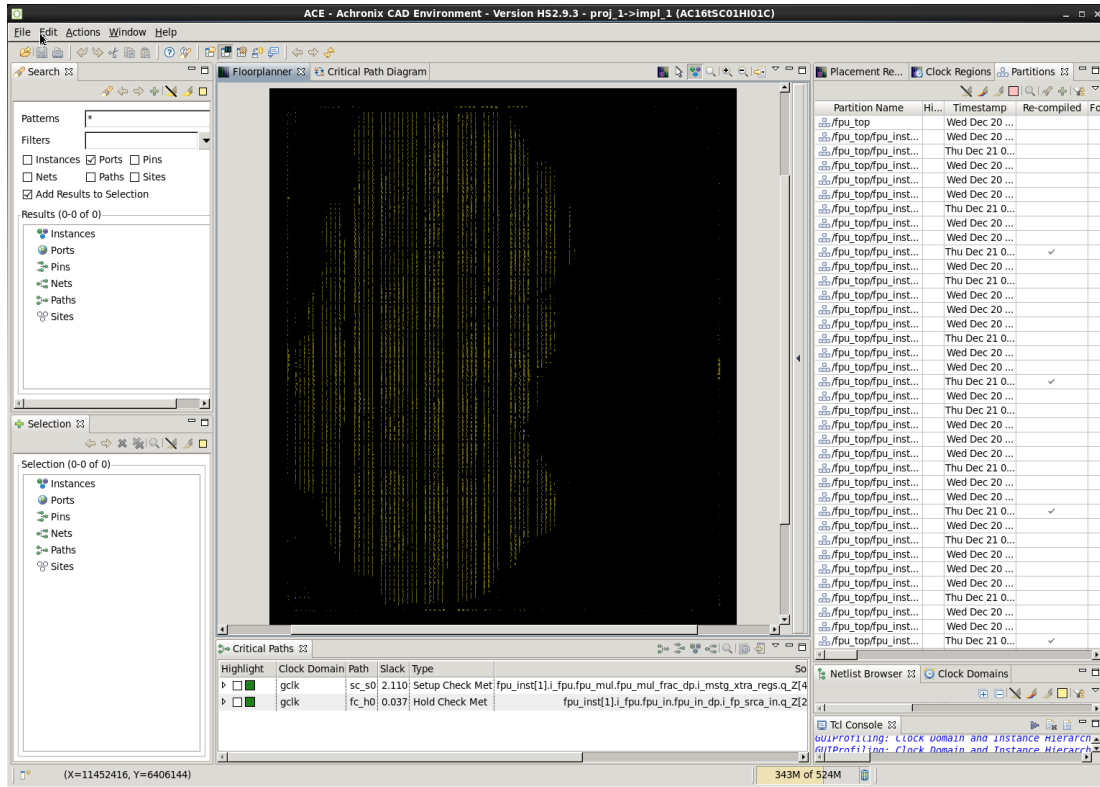


Figure 206 • ACE Floorplanner Perspective After rtl_V4 Incremental Compile Iteration

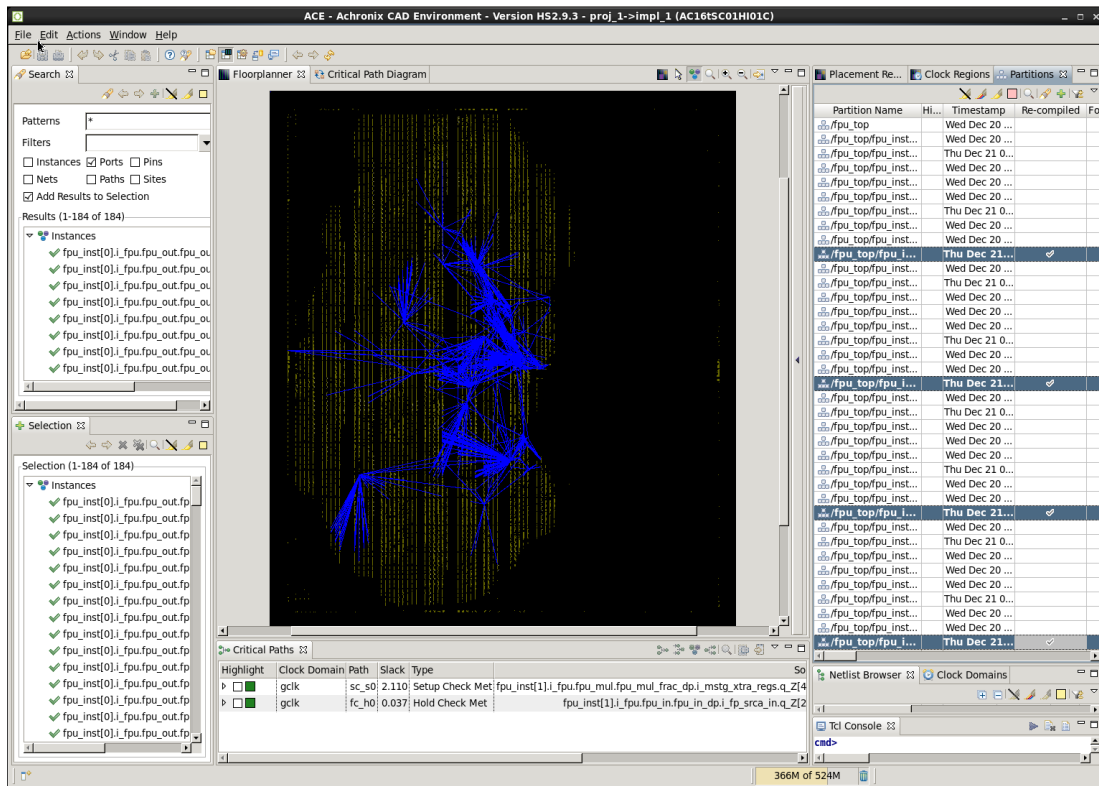


Figure 207 · ACE Floorplanner Perspective with Selected Instance Flylines (rtl_V4)

Note

For details how to run a set of changes in order to select an optimal implementation, continue on the [Multiprocess Incremental Compile Tutorial \(page 447\)](#). This second tutorial expands upon concepts from this tutorial.

Multiprocess Incremental Compile Tutorial

Using the incremental compile flow with the multiprocess GUI can be a powerful combination to help with timing closure. The Multiprocess GUI is used to try multiple experiments with different implementation options and/or sets of design constraints. The best implementation can then be selected to be the source for unchanged partitions in a subsequent incremental run. Across all implementations, the locked placement and routing data for all unchanged partitions is then merged from that best implementation. This merging is accomplished by copying the `best_impl/output/<design>.icdb` file from the best implementation to all other implementations before starting the next incremental iteration. All file copying is handled automatically by the multiprocess GUI. If you have not yet completed the [Single-Process Incremental Compile Tutorial \(page 408\)](#), please complete Steps 1 through 5 of that tutorial now before proceeding.

Below are the step-by-step actions of using the multiprocess GUI inside the incremental compile flow.

Step 1: Compile the Design in Synplify Pro or Clear the ACE Project

If you have previously completed the [Single-Process Incremental Compile Tutorial \(page 408\)](#), clear the ACE project and begin again from the beginning. Clear the project by deleting all of the files and subdirectories under the Ace directory of the tutorial work area. Otherwise, the first time ACE is run, it performs an incremental compile and the results do not match those described below.

Step 2: Create Multiprocess Implementations and Run ACE

From the ACE Home Screen, Use **Window** → **Show View** → **Multiprocess** to open the multiprocess GUI. Then select the **Generate Implementations from Option Sets** radio button. This action generates a large set of implementations automatically using a number of predefined implementation option variables. Optionally, set the “Parallel Job Count” option and configure the job submission system in the Ace preferences. Next click the **RUN** button (three stacked green triangles) to start running all implementations in parallel in the background. See the following screenshot of the Multiprocess View menus.

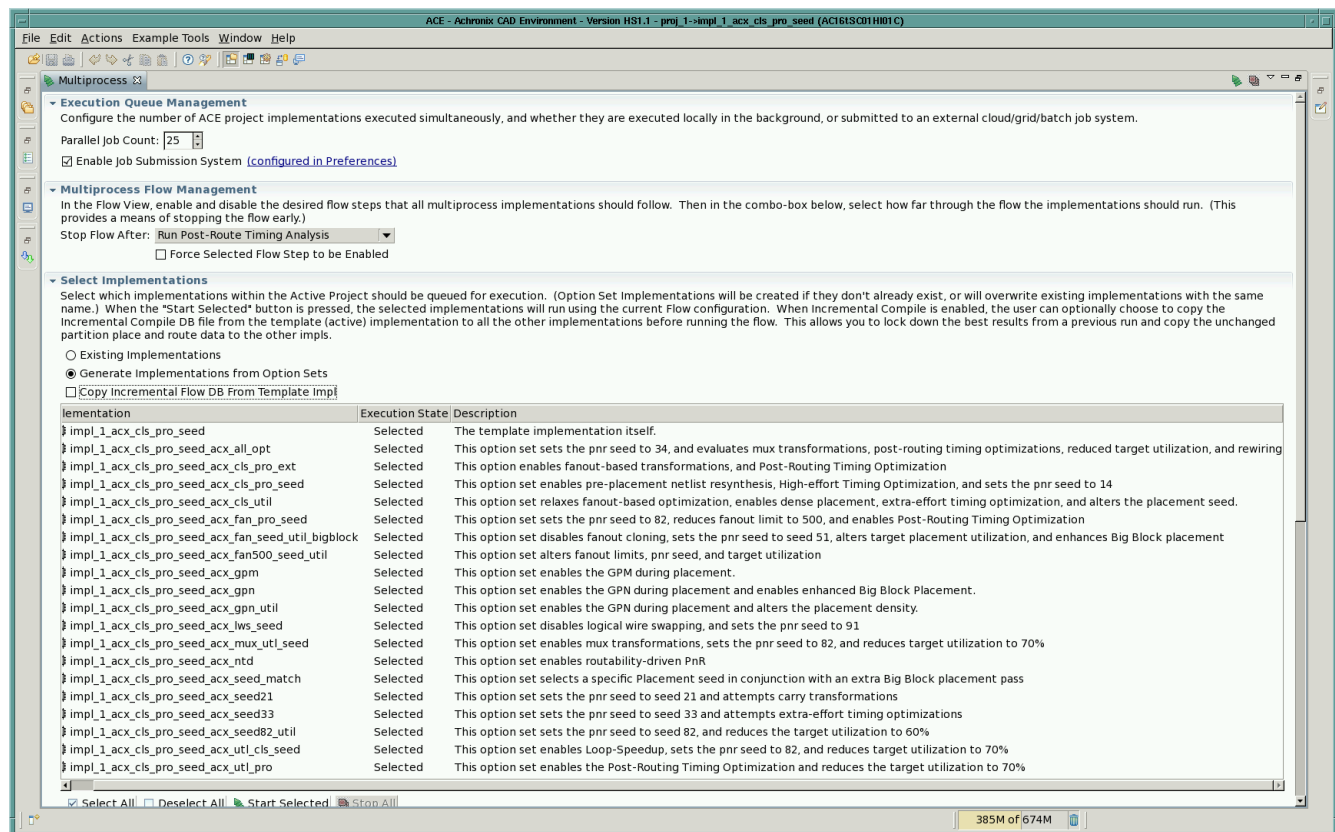


Figure 208 • Multiprocess View Menus

As in the single-process incremental compile flow, during the first pass through ACE, all implementations have their partitions compiled from scratch, as seen in the following screenshot of the partition report from one of the completed implementations.

ACE - Achronix CAD Environment - Version HS1.1 - proj_1->impl_1_acc_cls_pro_seed (AC16TSC01H01C)

File Edit Actions Example Tools Window Help

Partition Report - fpu_top Timing Report - Routed - fpu_top Clock/Reset Report - fpu_top Utilization Report - Routed - fpu_top

Partition Report

ACE - Achronix CAD Environment - Version HS1.1 - Build 95908* - Date 2016-10-12 14:50
 Design: proj_1_impl_1_acc_cls_pro_seed - fpu_top
 Device: AC16TSC01H01C
 Generated on Wed Oct 12 22:19:11 PDT 2016
 Host: cad43.achronix.local

Summary

- Number of partitions: 37 (37 re-compiled, 100.00%)
- Number of instances: 51107 (51107 re-compiled, 100.00%)
- Number of nets: 62327 (62327 re-compiled, 100.00%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthrough Nets
fpu_top	fpu_top	Yes	Wed 12 Oct 2016 10:06:52 PM	Hard	23135	19707	13406	5320	154	0	308	0	0	0	364	154	1	0	0	0	0	0	0	0	0	0	0	0	0
fpu_top fpu_inst01_fpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst11_fpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst21_fpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst31_fpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst11_fpu fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst01_fpu fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst21_fpu fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst31_fpu fpu_div	fpu_div	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst01_fpu fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
fpu_top fpu_inst21_fpu fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
fpu_top fpu_inst11_fpu fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0
fpu_top fpu_inst31_fpu fpu_in	fpu_in	Yes	Wed 12 Oct 2016 10:06:52 PM	Locked	877	789	484	305	0	0	0	0	0	0	0	0	0	0	83	311	82	0	1	1	0	0	0	1	0

375M of 674M | TCL: run-step rep..._tim... (70%)

Figure 209 • Partition Report from First Completed Multiprocess Implementation

Step 3: Select the Implementation with Best Performance

After all of the parallel runs have completed, select the implementation with the best performance on the most timing-critical clock domain. A summary of the frequency, setup slack, and hold slack for each implementation is provided in the Multiprocess Summary Report (see the following screenshot).

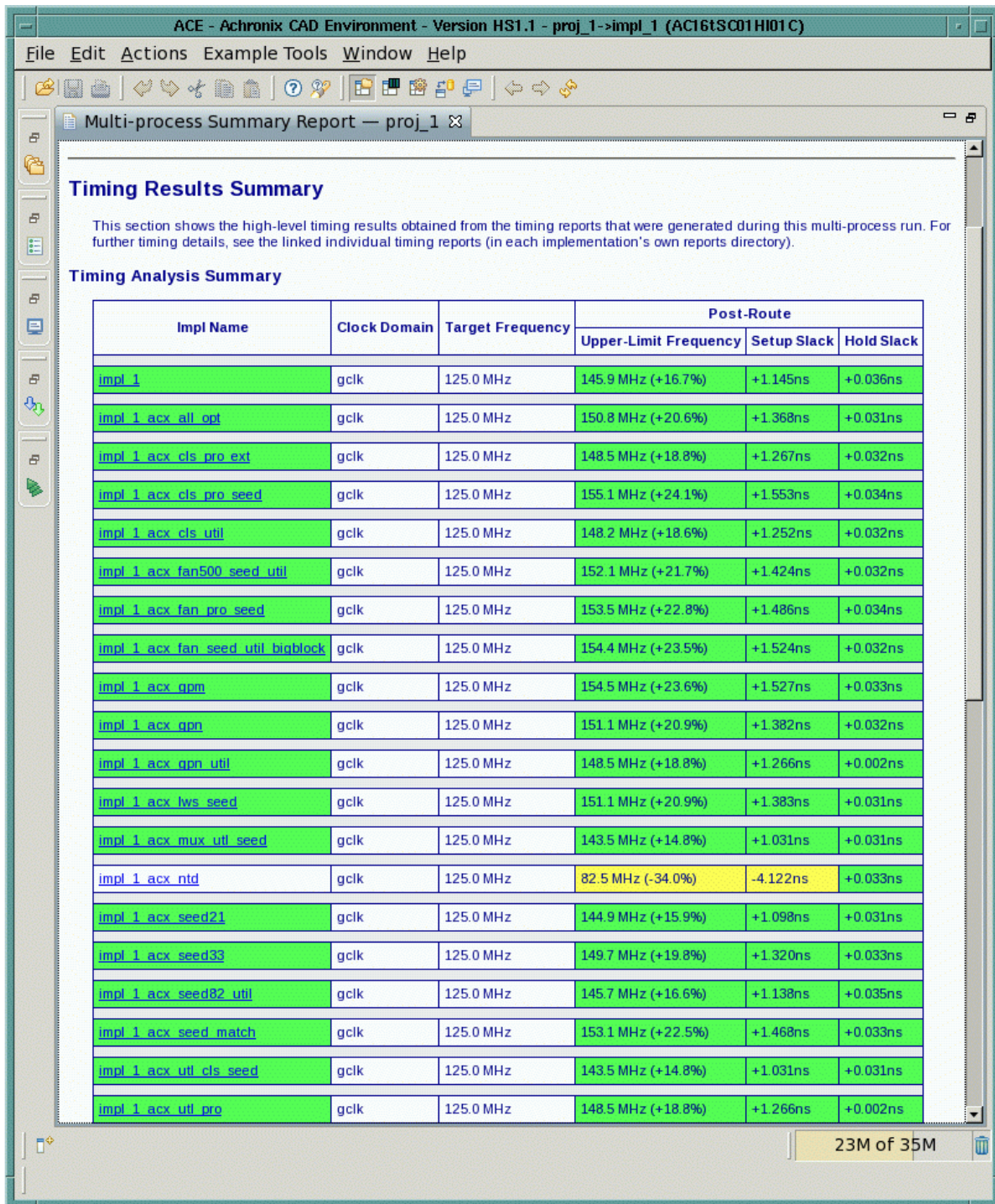


Figure 210 • Multiprocess Summary Report from the First Incremental Run

The following screenshot shows the critical path in the best implementation of the run, impl_1_acx_cls_pro_seed (actual results may differ).

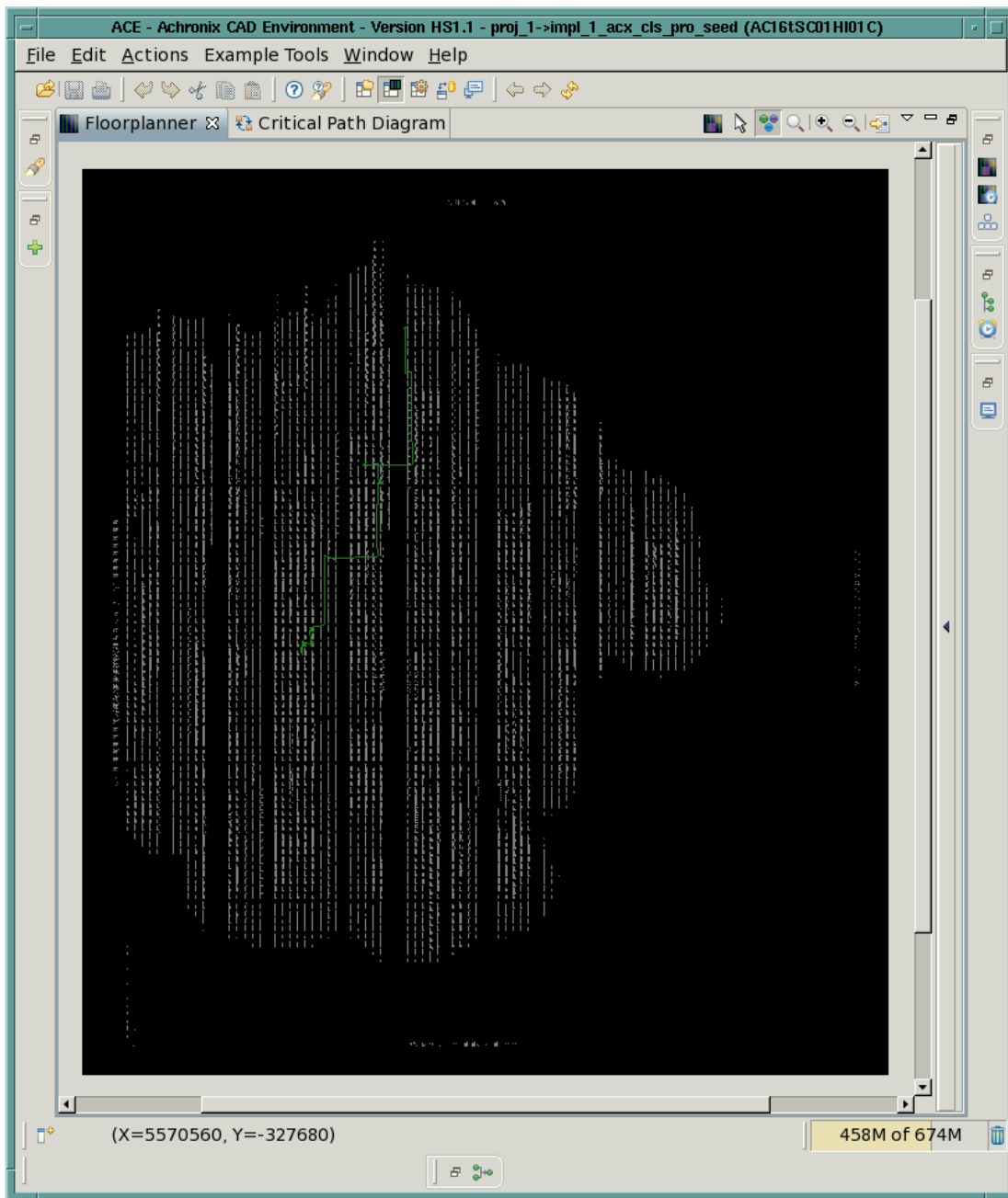


Figure 211 • Critical Path (green) in the Best Multiprocess Implementation

Return to the Multiprocess View and check the **Copy Incremental Flow DB from Template Impl** checkbox. Optionally, change the radio button to **Existing Implementations** if desired (though this is not necessary). See the following screenshot.

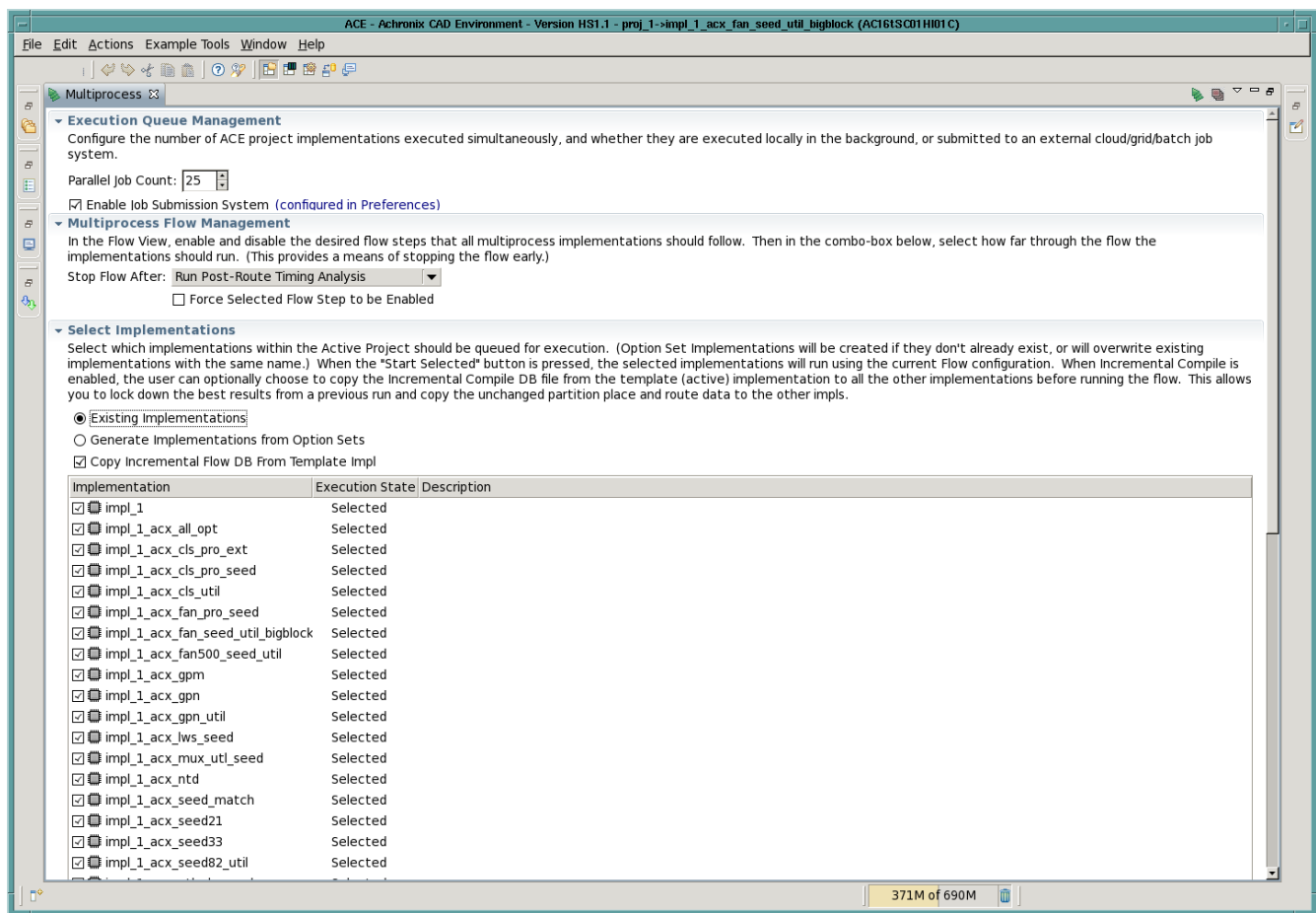


Figure 212 • Multiprocess View with the

The Template implementation referred to in the checkbox is the implementation to be used as the source for all unchanged partitions in the next incremental compile. The Template implementation is the same as the Active implementation. From the Projects View of the Projects Perspective, click the triangle to the left of the Project name to expand the list of implementations. Then left-click on the desired implementation (the one with the best performance) to make it the Active implementation. The implementation name of the active implementation turns bold and is highlighted as in the following screenshot.

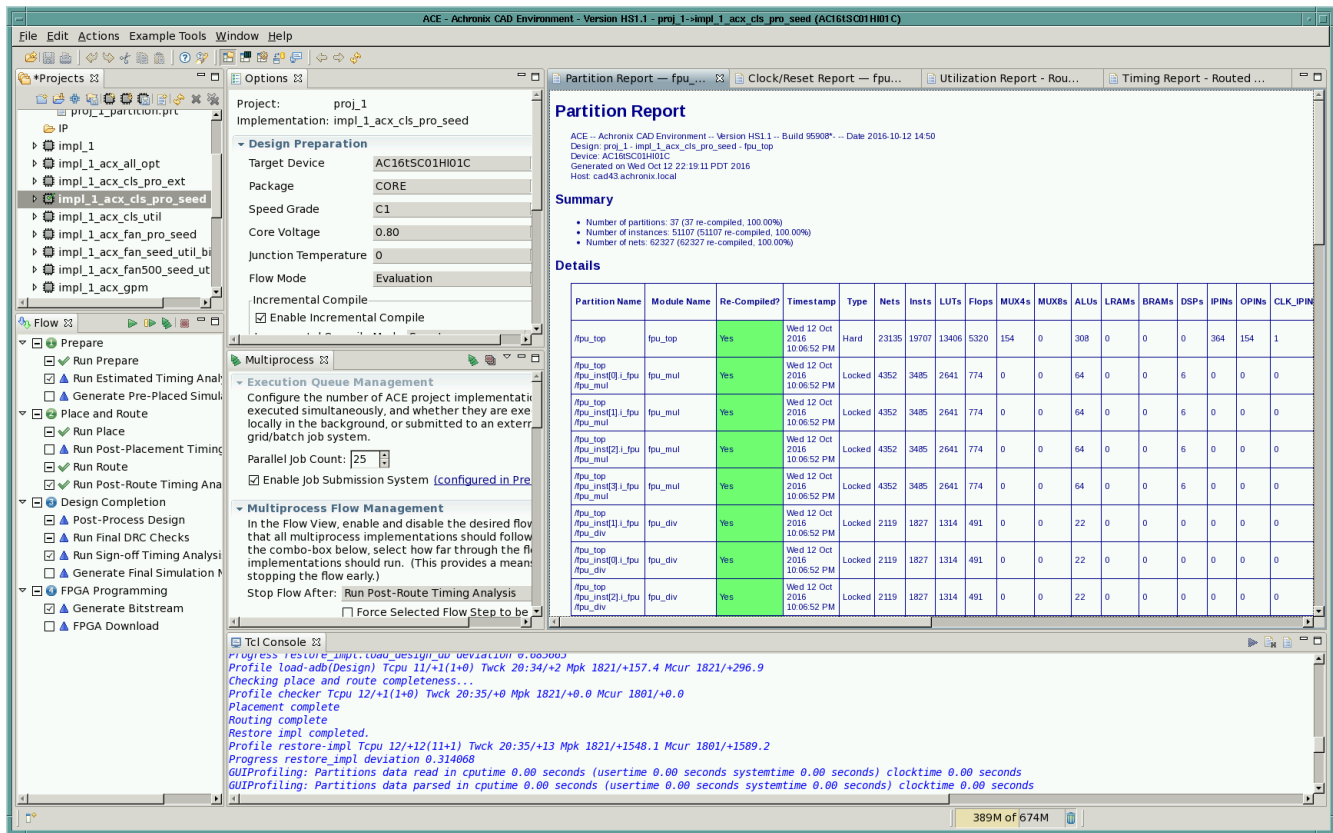


Figure 213 • Selection of the Template (Active) Implementation

Step 4: Change the RTL and Recompile the Design in Synplify Pro

Repeat Steps 8-10 of the [Single-Process Incremental Compile Tutorial \(page 408\)](#) to modify the RTL and force Synplify Pro to recompile the partitions in at least one of the defined compile points.

Step 5: Recompile the Multiprocess Implementations in ACE

Click the three stacked green triangle icon in the Multiprocess view to start a new incremental compile iteration on all implementations in parallel. ACE automatically copies the output /<design>.icdb file from the Template implementation into all other implementations and uses that as the source for the tear-and-stitch operation on all unchanged partitions during the run_prepare flow step.

As seen in the following screenshot from the impl_1_acx_mux_utl_seed implementation, only 8 of the 37 partitions have been recompiled in this iteration.

Partition Report
 ACE - Achronix CAD Environment - Version HSL1.1 - Build 95908* - Date 2016-10-12 14:50
 Design: proj_1_impl_1_acx_clk_pro_seed - fpu_top
 Device: AC16SC01H01C
 Generated on Wed Oct 12 23:04:38 PDT 2016
 Host: cae60.achronix.local

Summary

- Number of partitions: 37 (8 re-compiled, 21.62%)
- Number of instances: 51123 (15572 re-compiled, 30.46%)
- Number of nets: 62242 (19616 re-compiled, 31.47%)

Details

Partition Name	Module Name	Re-Compiled?	Timestamp	Type	Nets	Insts	LUTs	Flops	MUX4s	MUX8s	ALUs	LRAMs	BRAMs	DSPs	IPINs	OPINs	CLK_IPINs	CLK_OPINs	Boundary Input Nets	Boundary Output Nets	Registered Input Nets	Registered Output Nets	Constant Input Nets	Constant Output Nets	Floating Input Nets	Floating Output Nets	Dangling Input Nets	Dangling Output Nets	Feedthrough Nets	
fpu_top fpu_inst[1]_l_tpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0	
fpu_top fpu_inst[0]_l_tpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0	
fpu_top fpu_inst[2]_l_tpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0	
fpu_top fpu_inst[3]_l_tpu fpu_mul	fpu_mul	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	4352	3485	2641	774	0	0	64	0	0	6	0	0	0	0	113	53	1	0	0	1	0	0	0	1	0	
fpu_top fpu_inst[9]_l_tpu fpu_mul_cff	fpu_mul_cff	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
fpu_top fpu_inst[1]_l_tpu fpu_mul_cff	fpu_mul_cff	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
fpu_top fpu_inst[2]_l_tpu fpu_mul_cff	fpu_mul_cff	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
fpu_top fpu_inst[3]_l_tpu fpu_mul_cff	fpu_mul_cff	Yes	Wed 12 Oct 2016 10:46:51 PM	Locked	552	408	218	188	0	0	2	0	0	0	0	0	0	0	0	133	78	3	7	1	1	0	0	0	0	3
fpu_top	fpu_top	No	Wed 12 Oct 2016 10:06:52 PM	Hard	23134	19707	13406	5320	154	0	308	0	0	0	364	154	1	0	0	0	0	0	0	0	0	0	0	0	0	0
fpu_top fpu_inst[1]_l_tpu fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst[0]_l_tpu fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0
fpu_top fpu_inst[2]_l_tpu fpu_div	fpu_div	No	Wed 12 Oct 2016 10:06:52 PM	Locked	2119	1827	1314	491	0	0	22	0	0	0	0	0	0	0	0	129	65	1	0	0	1	0	0	0	1	0

Figure 214 - Partition Report from Second Incremental Multiprocess Compile

After all of the parallel runs complete, return to the updated Multiprocess Summary Report in the Multiprocess View to examine the critical path reports for each implementation.

As seen in the example below, the impl_1_acx_clk_pro_seed implementation achieved 155.1 MHz in the first iteration and 148.2 MHz in the second iteration. One of the incremental compiles, for impl_acx_fan_pro_seed, failed to route and returned a Flow Error.

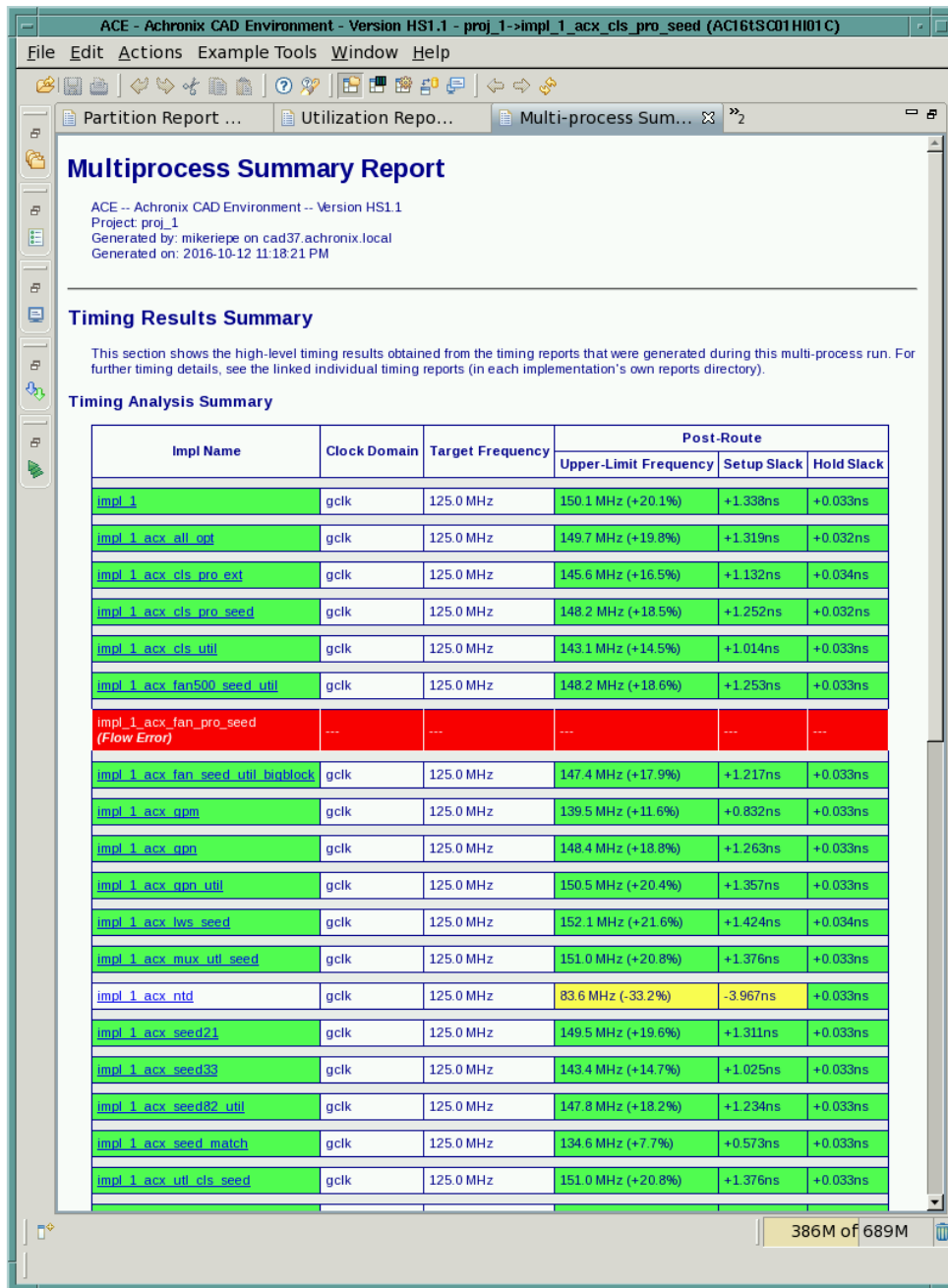


Figure 215 • Multiprocess Summary Report from the Second Incremental Run

Finally, select **File** → **Restore Implementation** to restore the routed.acxdb database for the Template implementation, and select **Actions** → **Timing** → **Run Post-Route Timing Analysis** to observe the new critical path. The following screenshot shows the critical path in the Template implementation after recompiling all of the changed partitions.

As usual, the instances of all unchanged partitions are highlighted in a dark yellow color to indicate that they are locked.

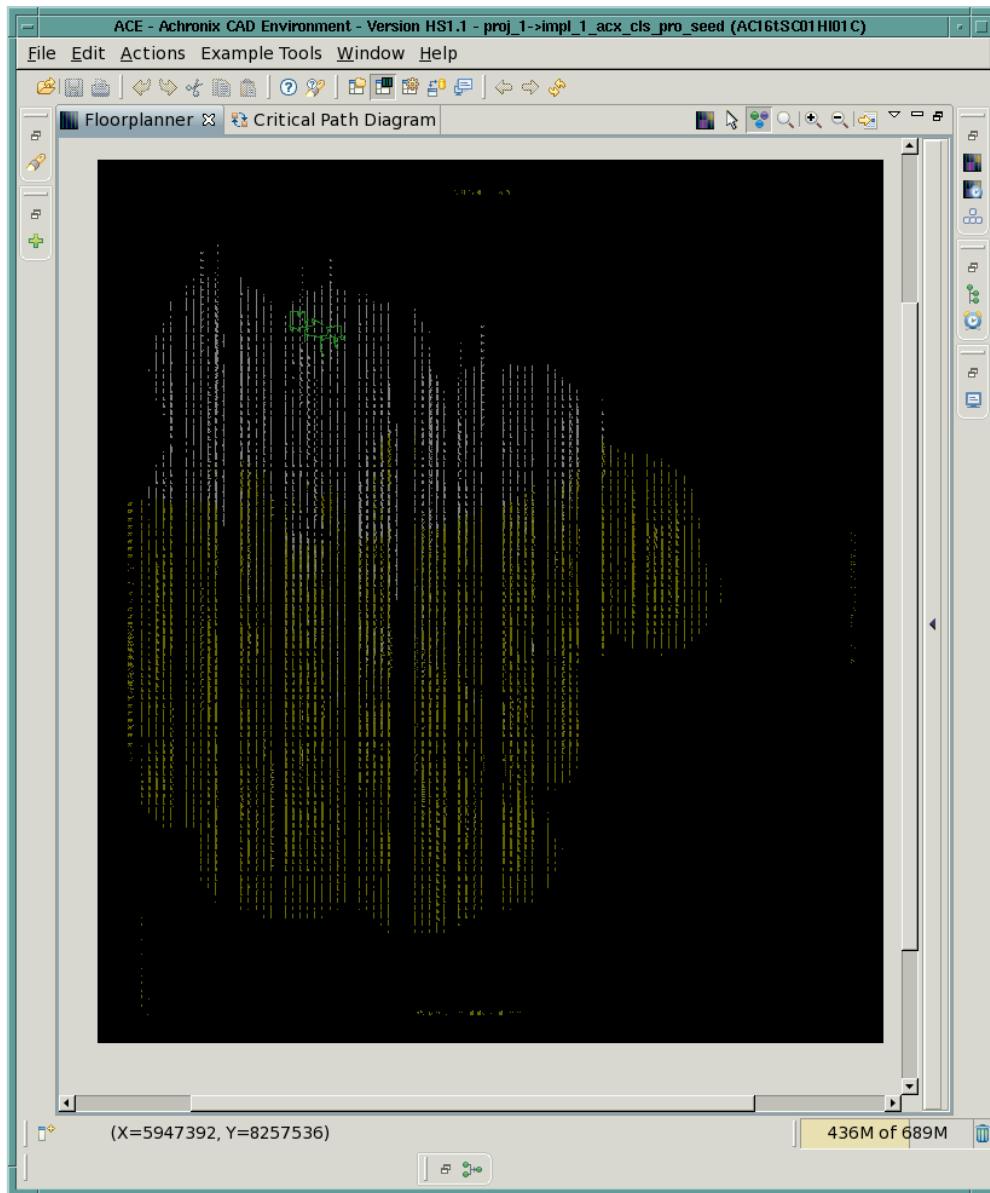


Figure 216 · Critical Path (Green) from the Best Multiprocess Implementation of the Second Incremental Run

All timing paths inside the unchanged partitions can be observed to remain the same. Once timing closure of a critical block in at least one of the multiprocess implementations is achieved, this flow allows timing closure to be maintained until that block must be recompiled.

Automatic Flop Pushing into I/O Pins

The term “flop pushing” refers to the process of converting an unregistered I/O pin, plus one or more attached DFFs, into a registered I/O pin. The purpose of flop pushing is to help with chip I/O timing closure. By avoiding the pin-to-flop, or flop-to-pin, delays, extra margin is achieved for off-chip timing paths.

Background

The flop-pushing feature is necessary in ACE because Synplify-Pro does not support inferencing of registered I/O pins. The following Verilog source code describes a simple design consisting of two flip-flops, plus a set of IPIN and OPIN instances that have been behaviorally instantiated in a top-level wrapper module.

Verilog Design File

```

1  module TEST_TOP (in, rst, ce, clk, out)
2      input in, rst, ce, clk;
3      output out;
4
5      IPIN    ipin_in  (.din(in),  .dout(in_p) );
6      IPIN    ipin_rst (.din(rst), .dout(rst_p));
7      IPIN    ipin_ce  (.din(ce),  .dout(ce_p) );
8      CLK_IPIN ipin_clk (.din(clk), .dout(clk_p));
9      OPIN    opin_out (.din(out),  .dout(out_p));
10
11     TEST test (.in(in_p), .rst(rst_p), .ce(ce_p), .clk(clk_p), .out(out_p));
12
13 endmodule
14
15 module TEST (in, rst, ce, clk, out)
16     input in, rst, ce, clk;
17     output out;
18     reg out;
19
20     wire dff_q;
21
22     ACX_DFFER dff (.d(in), .clk(clk), .ce(ce), .rn(rst), .q(dff_q));
23
24     always @(posedge clk or negedge rst)
25     begin
26         if (!rst)
27             out <= 0;
28         else
29             if (ce) out <= dff_q;
30     end
31 endmodule

```

Alternatively, the IPINs and OPINs could be instantiated in the netlist with a set of create_boundary_pins Tcl commands in a separate .pdc constraints file, as follows.

PDC Constraints File

```

1 create_boundary_pins {p:in} {ipin_in}
2 create_boundary_pins {p:rst} {ipin_rst}
3 create_boundary_pins {p:ce} {ipin_ce}
4 create_boundary_pins {p:clk} {ipin_clk}
5 create_boundary_pins {p:out} {opin_out}

```

When structurally instantiating the IPINs and OPINs, one could register the I/Os by connecting the clk, ce, and rstn inputs. But the `create_boundary_pins` constraints will not register the I/Os.

By placing the flops in the device core as separate instances, extra delay from the pin to the flop is incurred on the I/O-ring-to-core routing path. If the device's I/O timing is tight, that could result in an external setup timing failure. On the other hand, the presence of separate DFFER instances allows the flops to be placed in the core near their fan-in/fan-out logic, possibly reducing the routing delay from the flop to intermediate logic in the design. Flop pushing can therefore be viewed as a form of optional retiming, allowing the designer the ability to trade off-chip for on-chip delays between the I/O pins and the boundary flops.

In ACE, flop pushing is performed by a command called the *reconditioner* during the `run_prepare` flow step. Flop pushing happens very early in the prepare flow, after flattening and elaboration. You will see messages like the following in the ACE logfile.

```

INFO: Running netlist reconditioner...
INFO: Reconditioning post_elaborate TEST_TOP
INFO: For detailed information see the file: ace/impl_1/pnr/log/
test_recondition_post_elaborate.log
INFO: Pushed DFFs into 1 of 1 candidate IPIN output pins
INFO: Pushed DFFs into 1 of 1 candidate OPIN input pins
INFO: Reconditioning complete (runtime = 00:00:00)

```

Capabilities

In the simplest case, ACE performs flop pushing by executing the following algorithm:

1. Finding an IPIN that drives a DFF, or a DFF that drives an OPIN
2. Deleting the DFF
3. Converting the IPIN/OPIN into a flopped IPIN/OPIN by making the following changes:
 - a. Connecting the DFF clock input to the IPIN/OPIN clock input
 - b. Optionally connecting the DFF reset and enable inputs to the IPIN/OPIN reset and enable inputs
 - c. Changing the IPIN/OPIN's mode parameter from "0" to "1"
 - d. Moving the value of the `init` and `sr_assertion` attributes from the DFF to the IPIN/OPIN

Flop pushing is supported for flops connected to IPIN data output pins, and OPIN data input pins. For obvious reasons it is not supported for CLK_IPINs or CLK_OPINs. ACE also supports more complex cases:

- IPINs that drive more than one DFF
- Chains of buffer LUTs and inverter LUTs between the pad and the DFF

- OPINs in which the data input and the clock-enable input are both driven by DFFs

The reconitioner checks for many possible scenarios that prevent flops pushing, especially in the above complex cases. A partial list of such exceptions includes:

- The IPIN or OPIN is already registered
- DFFs in the IPIN fanout do not all share the same clock input nets
- DFFs in the IPIN fanout do not all share the same set, reset, or enable input nets
- DFFs in the IPIN fanout are a mixture of DFF, DFFC, DFFP, DFFR, and/or DFFS instances
- DFFs in the IPIN fanout have a mixture of synchronous and asynchronous resets
- DFFs in the IPIN fanout are a mixture of positive and negative edge triggered
- DFFs in the IPIN fanout have different `init` parameter values
- A DFF in the IPIN fanout is driven by more than one input pin, or drives more than one output pin
- A DFF clock is driven by a generated clock or reset that can only be routed in the core (not the I/O ring)
- LUTs between the IPIN/OPIN and DFF are configured as anything but a buffer or inverter
- Nets on the path between the IPIN/OPIN and DFF (including intermediate buffers or inverters) have a fanout greater than one
- DFFs driven by a IPIN/OPIN through a chain of buffers and/or inverters that have different inversion (odd vs. even number of inverters)

ACE Implementation Options

The behavior of ACE with respect to flop pushing is controlled globally by the implementation options `push_flops_into_pads` and `pad_flop_pushing_clock_type`, described below. See the [Options View \(page 96\)](#) section for more information about implementation options. Flop pushing can also be controlled on a per-instance basis using the `ace_use_ioff` attribute, as described in the next section [ACE Attributes \(page 460\)](#).

push_flops_into_pads

The implementation option `push_flops_into_pads` controls whether flop pushing is performed automatically or manually. This implementation option has the following legal settings:

- "0" - flop pushing is completely disabled
- "1" - (manual mode) push flops into pads that have the `ace_use_ioff` attribute set to "1"
- "15" - (automatic mode) push flops into all pads *except* those that have the `ace_use_ioff` attribute set to "0"

Example Setting

```
set_impl_option push_flops_into_pads 15
```

pad_flop_pushing_clock_type

The implementation option `pad_flop_pushing_clock_type` enables automatic flop pushing to be controlled by the routing type of the pushed clock. The option only applies when the implementation option

`push_flops_into_pads` has the value "15" (automatic mode). This implementation option has the following legal settings:

- "boundary" – automatically push flops into pads only when the flops are clocked by a boundary clock
- "trunk" – automatically push flops into pads only when the flops are clocked by a trunk clock
- "all" – automatically push flops into all pads regardless of the clock routing type

The routing type of a clock net is controlled by the `set_clock_type` command.

set_clk_type Examples

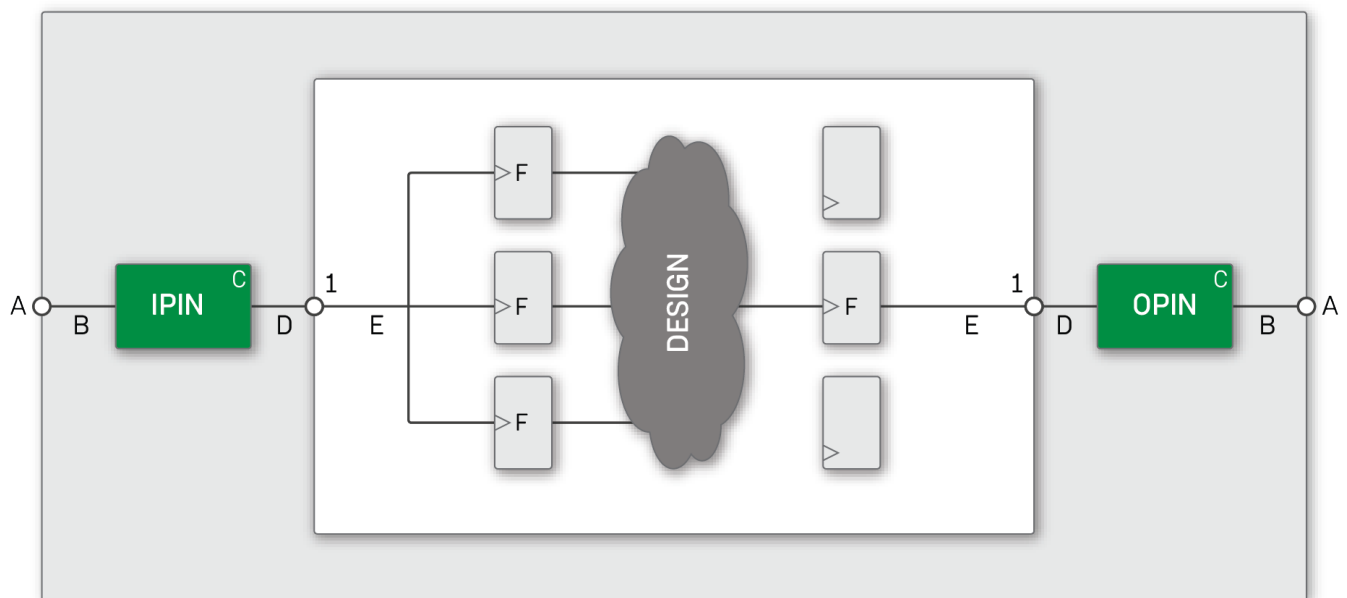
```
set_clock_type clk1 -boundary
set_clock_type clk2 -trunk
set_impl_option pad_flop_pushing_clock_type "boundary"
```

In the above example, only flops clocked by the boundary clock `clk1` are automatically pushed in the pads.

ACE Attributes

The behavior of ACE with respect to flop pushing can be controlled for individual input/output pads with the `ace_use_ioff` synthesis attribute. The attribute has the following semantics:

- If the `ace_use_ioff` attribute associated with an I/O pin has a non-zero value, ACE pushes flip-flops into that pin when possible. This behavior is useful when the `push_flops_into_pads` implementation option has the value 1 (manual mode).
- If the `ace_use_ioff` attribute associated with an I/O pin has the value 0, ACE prevents flip-flops from being automatically pushed into that pin. This behavior is useful when the `push_flops_into_pads` implementation option has a value of 15 (automatic mode).



7079642-01.2024.04.13

Figure 217 • Valid Locations to Place the `ace_useioff` Attribute in an eFPGA Design Hierarchy

The `ace_useioff` attribute can be placed in several different locations in your RTL code, as shown in the figure above, and as described below.

- On a top-level module port (A) connected to an IPIN/OPIN
- On a net (B) connecting an IPIN/OPIN to a top-level module port
- On an IPIN/OPIN instance (C)
- On a net (D) connecting an IPIN/OPIN to a hierarchical module pin
- On a net (E) connecting a hierarchical module port to one or more flip-flop instances
- On one or more flip-flop instances (F) driven by an IPIN instances or driving an OPIN instance

If there is more than one flip-flop driven by the same IPIN, all DFF instances must have a `ace_useioff` attribute with the same value.

The `ace_useioff` *cannot* be placed in the following location:

- On the an intermediate port (1) separating two levels of hierarchy. Synplify-Pro does forward annotate attributes on intermediate *output* ports, but when placed on intermediate *input* ports the attributes are lost during flattening, so this method is not recommended

Be careful not to add the `ace_useioff` attributes in more than one location relative to a given I/O pin. A conflict occurs when some attributes associated with particular I/O pin have the value "1", and some have the value "0". When a conflict is detected, ACE issues a warning message, the value "0" is assumed, and flop pushing is disabled for that I/O. Be especially careful to avoid conflicts when the attributes are placed on DFF instances, and an IPIN drives more than one DFF.

The `ace_useioff` attributes can be specified by the user in the Verilog/VHDL source code, or in the physical design constraints (`.pdc`) file, as shown in the following examples. An advantage of specifying the attribute in the `.pdc` file is that it is not necessary to re-compile the design in order to experiment with different flop-pushing strategies. An advantage of specifying the attribute in the HDL source code is that the intent of the designer is more self-documenting, as readers do not have to refer to a separate `.pdc` file and cross-reference the port/net/instance names between the files.

Examples

Several examples follow demonstrating how to set the `ace_useioff` attribute on:

- Top-level ports
- I/O pin instances
- Boundary wires
- DFF instances

Also demonstrated is whether to use either Verilog, VHDL, or a PDC constraint. For more information about the use of attributes in Synplify, see the section "Forward Annotation of RTL Attributes to Netlist" in the Synthesis Optimization Recommendations chapter of the *Synthesis User Guide* (UG018).

Verilog Example of a Port attribute

```

module flop_push_test1 (ina, inb, sel, clk, z0);
    input [3:0] ina /* synthesis ace_useioff=1 */;
    input [3:0] inb /* synthesis ace_useioff=0 */;
    input sel      /* synthesis ace_useioff=1 */;
    input clk;
    output z0      /* synthesis ace_useioff=1 */;
endmodule

```

VHDL Example of a Port Attribute

```

entity flop_push_test1 is
port(
    ina    : in signed( 3 downto 0 );
    inb    : in signed( 3 downto 0 );
    sel    : in std_logic;
    clk    : in std_logic;
    z0     : out std_logic
);

attribute ace_useioff : boolean;
attribute ace_useioff of ina : signal is TRUE;
attribute ace_useioff of inb : signal is FALSE;
attribute ace_useioff of sel : signal is TRUE;
attribute ace_useioff of z0  : signal is TRUE;

end entity;

```

Physical Design Constraints (.pdc) File of Port Attributes

```

set_property ace_useioff "1" [find -ports {ina\[*\]}]
set_property ace_useioff "0" [find -ports {inb\[*\]}]
set_property ace_useioff "1" {p:sel p:z0}

```

Note

In the above three examples, the input port bus `ina`, the input port `sel`, and the output port `z0` are selected for flop pushing. The input port bus `inb`, the input port `clk`, are not.

If the `ace_useioff` attribute has the value "1", ACE tries to push a flip-flop into the pad connected to the given port even when flop pushing is disabled by default. If the attribute has the value "0", ACE prevents flop pushing on that pad, even if flop pushing is enabled by default. Of course flop pushing may be prevented by any of the exceptions listed [above \(page 459\)](#).

It is not possible, using RTL attributes, to apply different values of the `ace_useioff` attribute to different ports in a port bus (e.g, giving `in_a[2]` a `ace_useioff` value of "0"). It also cannot be done by applying an attribute to a wire that is assigned the value of the bus (see the following example, `flop_push_test5`). The solution requires bit-blasting the bus into separate ports or assigning different values of `ace_useioff` to different bus ports using PDC constraints.

Verilog Example of an IPIN Instance Attribute

```
module flop_push_test2 (in, clk);
  input [38:0] in;
  input clk;

  wire ipin_dout_37;
  IPIN ipin_37( .din(in[37]) , .dout(ipin_dout_37) ) /* synthesis ace_useioff = 0 */;

  reg data_37 = 1'b0;
  always @(posedge clk)
    begin
      data_37 <= ipin_dout_37;
    end
endmodule
```

Physical Design Constraints (.pdc) File of IPIN Instance Attributes

```
set_property ace_useioff "1" [find -insts {ipin_*}]
set_property ace_useioff "0" {i:ipin_37}
```

Verilog Example of a Boundary Wire Attribute

```
module flop_push_test3 (in, clk);
  input [38:0] in;
  input clk;
```

```
(* syn_keep *) wire ipin_dout_37 /* synthesis ace_useioff=0 */;
IPIN ipin_37 (.pad(in[37]), .dout(ipin_dout_37));

reg level1_37 = 1'b0;
always @(posedge clk[0])
begin
    level1_37 <= ipad_dout_37;
end
endmodule
```

Physical Design Constraints (.pdc) File of Wire Attributes

```
set_property ace_useioff "1" [find -nets {ipin_dout_*}]
set_property ace_useioff "0" {n:ipin_dout_37}
```

Verilog Example of a DFF Instance Attribute

```
module flop_push_test4 (in, clk);
    input [38:0] in;
    input clk;

    wire ipin_dout_37;
    IPIN ipin_37(.din(in[37]), .dout(ipin_dout_37) );

    wire dff1_q, dff2_q;
    ACX_DFF dff1 (.d(ipin_dout_37), .clk(clk), q(dff1_q)) /* synthesis ace_useioff = 0
*/;
    ACX_DFF dff2 (.d(ipin_dout_37), .clk(clk), q(dff2_q)) /* synthesis ace_useioff = 0
*/;
endmodule
```

Physical Design Constraints (.pdc) File of DFF Instance Attributes

```
set_property ace_useioff "1" [find -insts {dff*}]
set_property ace_useioff "0" {i:dff1 i:dff2}
```

The following is an example of a boundary wire attribute that *does not* work as expected.

Verilog Example of a Boundary Wire Attribute that DOES NOT work

```
module flop_push_test5 (in, clk);
    input [38:0] in;
```

```

input clk;

(* syn_keep *) wire ipin_din_37 /* synthesis ace_useioff=0 */;
assign ipin_din_37 = in[37];

wire ipin_dout_37;
IPIN ipin_37 (.pad(ipin_din_37) , .dout(ipin_dout_37) );

reg data_37 = 1'b0;
always @(posedge clk[0])
    begin
        data_37 <= ipad_dout_37;
    end
endmodule

```

Caution!

As noted above, it is not possible using RTL attributes to apply different values of the `ace_useioff` attribute to different ports in a port bus. The above example, `flop_push_test5`, appears to be a clever way to apply the `ace_useioff` attribute to the net connecting the UCM port to a single IPIN in the 39-bit port bus `in`. Unfortunately, this technique does not work. Synplify Pro optimizes away the wire `ipin_din_37`, despite the presence of the `syn_keep` attribute, and the `ace_useioff` attribute is not forward-annotated into the ACE input netlist. One can use PDC, however, to apply the `ace_useioff` attribute to the IPIN instance as in the following example.

Physical Design Constraints (.pdc) For Example `flop_push_test5` That DOES Work

```

set_property ace_useioff "1" [find -nets {in\[*\}}]
set_property ace_useioff "0" {n:in\[37\}}

```

Timing Analysis Implications

As discussed above, flop pushing can be viewed as a form of retiming. Pushing a flop into a boundary pin reduces the off-chip timing path at the flop input by reducing the wiring delay. But it increases the on-chip timing path at the flop output, possibly by a large amount depending on how closely the driven logic is placed to the boundary pin. On small designs, especially when the logic is placed near the center of the chip, the increased delay can be significant.

Enabling flop pushing by default across a suite of designs often causes QoR to appear to degrade significantly. This apparent degradation happens because the off-chip delays are often not modeled well in the design timing constraints. The off-chip delay must be modeled using a `set_input_delay` or `set_output_delay` timing constraint. If those delays are zero, the improvement in off-chip delay may not be evident, and the increase in on-chip delay may be dominant.

When `set_input_delay` and `set_output_delay` constraints are given at all, this causes the timer to completely ignore timing paths that start or end off-chip. Pushing a flop into a pad with unspecified input/output delay could cause new setup/hold violations to appear that were not previously modeled.

Working with Virtual I/O

The role of I/O virtualization is to take a design with too many I/O pads (or boundary pins in the case of a Speedcore fabric) and reduce the number of I/O until the design fits in the given fabric. This option is only run in evaluation flow mode.

Behavior

I/O virtualization is performed automatically as part of the `run_prepare` flow step in evaluation flow mode. It is not permitted to export a bitstream for a design with virtualized I/Os. If there are a sufficient number of boundary pin sites to place the design, then the command is a no-op. If the design has more boundary pin instances than available sites, I/O virtualization modifies the netlist by reducing the number of boundary pins until the design fits. When virtualizing I/O, no attempt is made to maintain logical equivalency with the original netlist. Rather, the goal is to perturb the behavior of the placement and routing tools as little as possible and, therefore, make an evaluation run correspond as closely as possible to a production run of a similar design.

I/O virtualization operates by collapsing multi-bit bused boundary pins as well as single non-bused boundary pins. By default, pins are selected automatically for virtualization, starting with the widest pin bus until enough boundary pin sites are available to fit the remaining number of boundary pin instances. If that number is insufficient, individual non-bused pins are also virtualized. Pin buses and individual pins can also be manually selected for virtualization through top-level port attributes in the RTL or PDC constraints (see [Port Attributes \(page 467\)](#), below). Depending on the virtualization mode, some serialization boundary pins may be inserted, so it is possible for the process to fail and leave too many boundary pins in the design.

The pins are collapsed using one of three user-specified styles:

- **stubout** – each IPIN is replaced with a "stub" LUT that drives a constant zero onto the IPIN output net. Similarly, each OPIN is replaced with a "stub" LUT, driven by the OPIN input net, with a floating output pin. These stub LUTs are given `must_keep` attributes so that ACE does not optimize them away.
- **serialize_dff** – bused IPINs are replaced with a single IPIN that drives a scan chain implemented with DFFs. The output of each DFF drives the output net from its original corresponding IPIN as well as the next stage in the scan chain. Bused OPINs are replaced with a single OPIN that is driven by a scan chain implemented with DFFs. The input of each DFF is driven by a 2-input MUX, one input of which is driven by the input net from its original corresponding OPIN. The other input of the MUX is driven by the output of the DFF in the previous stage of the scan chain. One additional IPIN per port bus is also added which drives the select line of the MUXes.
- **serialize_lut** – this style is the same as the `serialize_dff` style, except that the scan chain is implemented with LUTs instead of DFFs.

The stubout style is the simplest and has the greatest rate of pad compression. However, it is the least realistic since there are no connections pulling the stub LUTs toward the edge of the chip. The placer pulls them into the chip core, placing them at the center of gravity of the loads that they drive. The two serialize styles keep one representative boundary pin and are, therefore, more realistic, though of course the strength of the placement forces pulling the scan chain toward to chip edge are significantly reduced. The main reason to use the `serialize_dff` style over the `serialize_lut` style is that, in the former style, the timer only sees a path that starts or ends at the last stage of the scan chain, while in the former the entire scan chain (possibly hundreds of LUT delays) contributes to the length of the timing path.

For the `serialize_dff` style, a clock must be connected to the DFFs that make up the scan chain. By default, that clock is selected automatically to be the clock in the chip core with the largest number of load pins. The clock can be user-specified either through a global implementation option, or on a per-port basis through an attribute on the port. These options are discussed in [Implementation Options \(page 467\)](#) and [Port Attributes \(page 467\)](#).

Implementation Options

The behavior of I/O virtualization can be controlled on a global basis by the following implementation options:

- **virtual_io_style** – controls the style, or method, used to virtualize excess I/O pad or boundary pin buses in the top-level netlist. Legal enumeration values are:
 - `stubout` (the default)
 - `serialize_dff`
 - `serialize_lut`
 See above for a definition of the behavior of each of these styles.
- **virtual_io_utilization** – sets the I/O pad or boundary pin utilization percentage targeted by I/O virtualization. Legal values are integers between 0 and 100. An error is returned if the given utilization cannot be met. A target utilization of zero percent requests that all possible port buses and non-bused ports are to be virtualized to achieve the smallest possible number of pins. A target utilization of 100 percent requests that port buses and non-bused ports are to be virtualized until the number of remaining ports fit into the target fabric. This option is mutually exclusive with the `virtual_io_num_pads` option (both cannot be specified).
- **virtual_io_num_pads** – sets the final number of I/O pad or boundary pin instances targeted by I/O virtualization. Legal values are 0 or larger. A target pad number of zero requests that all possible port buses and non-bused ports are to be virtualized to achieve the smallest possible number of pins. If the specified value is larger than the number of available I/O pad or boundary pin sites in the selected fabric, the number of available I/O pad or boundary pin sites are targeted. This option is mutually exclusive with the `virtual_io_utilization` option (both cannot be specified).
- **virtual_io_clock_port** – specifies the name of the clock, by its top-level port name, to be used by I/O virtualization to clock serialization flops. Only applies for the `serialize_dff` virtualization style. This option can also be specified individually for a given port with the RTL or PDC port attribute, `ace_virtualize_clock_port`, which overrides this option if given. If not specified, the virtualization clock is derived automatically as the core clock net driving the largest number of loads. This option is mutually exclusive with the `virtual_io_clock_net` option (both cannot be specified).
- **virtual_io_clock_net** – specifies the name of the clock, by its net name, to be used by I/O virtualization to clock serialization flops. Only applies for the `serialize_dff` virtualization style. This option can also be specified individually for a given port with the RTL or PDC port attribute, `ace_virtualize_clock_net`, which overrides this option if given. If not specified, the virtualization clock is derived automatically as the core clock net driving the largest number of loads. This option is mutually exclusive with the `virtual_io_clock_port` option (both cannot be specified).

Port Attributes

By default, I/O virtualization selects port buses for virtualization automatically. They are virtualized in order of decreasing size until the netlist meets the given target boundary pin utilization. However, port buses which are virtualized through the use of the RTL port attribute `ace_virtualize` can be manually controlled. When the virtualization style is set to `serialize_dff`, either a top-level port name or net name to be connected to the clock input of the new serialization flop instances can also be specified. Use the RTL port attribute `ace_virtualize_clock_port` or `ace_virtualize_clock_net` respectively.

The attribute can be set in the Verilog/VHDL source code, or in the physical design constraints (`.pdc`) file, as follows. An advantage of setting the property in the `.pdc` file is that the design does not have to be re-synthesized in order to experiment with different virtualization strategies.

Verilog Example

```

module pds (
    input
        clk_i,
    (* ace_virtualize="1", ace_virtualize_clock_port="clk_i" *)
    output [63:0]
        tx_data_o,
    (* ace_virtualize="1", ace_virtualize_clock_net="clk_i_c" *)
    output [ 7:0]
        tx_ifg_delay_o
endmodule

```

VHDL Example

```

entity pds is
port(
    clk_i      : in std_logic;
    tx_data_o  : out signed( 63 downto 0);
    tx_ifg_delay_o : out std_logic_vector(7 downto 0)
);

attribute ace_virtualize : boolean;
attribute ace_virtualize of tx_data_o : signal is TRUE;
attribute ace_virtualize of tx_ifg_delay_o : signal is TRUE;

attribute ace_virtualize_clock_port : string;
attribute ace_virtualize_clock_net : string;
attribute ace_virtualize_clock_port of tx_data_o : signal is "clk_i";
attribute ace_virtualize_clock_net of tx_ifg_delay_o : signal is "clk_i_c";

end entity;

```

If the target boundary pin utilization is not met after all user-specified ports are virtualized, additional ports are selected automatically until the target boundary pin utilization is met.

Physical Design Constraints (.pdc) File

```

set_property ace_virtualize "1" [find -ports {sample_src\[*\}}]
set_property ace_virtualize_clock_port "clk" [find -ports {sample_src\[*\}}]

```

Runtime Messages

Below are the output messages from I/O virtualization using the example above with user-specified port buses and the `serialize_dff` virtualization style.

Runtime Messages

```
INFO: Virtualize IO: Serializing user-specified 512-bit output PortBus tx_data_o using
clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 64-bit output PortBus tx_data_valid_o
using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 64-bit output PortBus tx_ifg_delay_o
using clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 512-bit input PortBus rx_data_i using
clock clk_i_c
INFO: Virtualize IO: Serializing user-specified 128-bit output PortBus pause_val_o using
clock clk_i_c
INFO: Virtualize IO: Serializing remaining 6 auto-selected input ports using clock clk_c
INFO: Virtualize IO: Serializing remaining 13 auto-selected output ports using clock
clk_c

WARNING: Virtualize IO: Netlist pds had too many IOs to fit in the selected device.
Merged and deleted 1280 of 1498 IO ports. Final number of ports is 227. This is for
evaluation purposes only and will cause simulation mismatches.
```

Schematic View

The following images illustrate each of the available virtualization styles with schematic diagrams showing 4-bit busses of input pads and output pads. First shown is the input netlist before pad virtualization, followed by the output netlist for the stubout, serialize_dff, and serialize_lut styles.

Input Netlist

The following figure illustrates the input netlist for a 4-bit bus of input pads, and a 4-bit bus of output pads. The output pads have an output-enable driver.

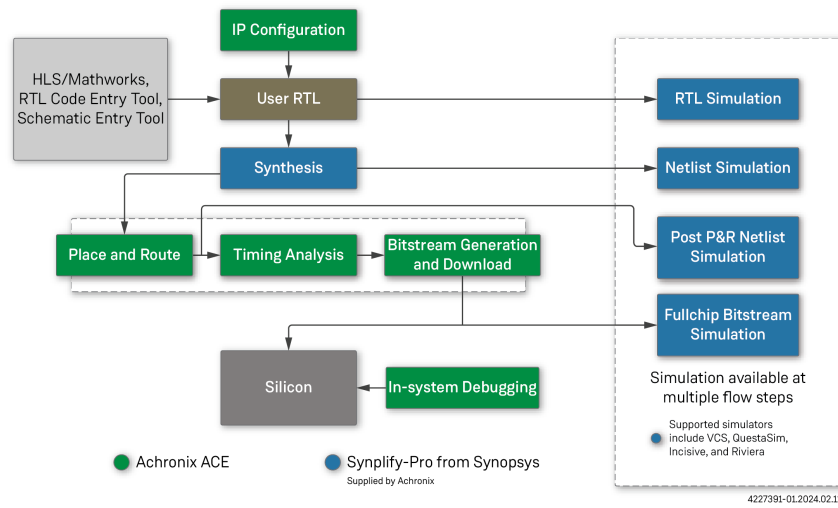


Figure 218 • Input and Output Pads

Output Netlist Styles

stubout

The following two schematics illustrate the output of I/O virtualization when using the stubout style. Notice that none of the IPIN or OPIN instances remain. The new LUTs replacing the IPIN instances are all driven by constant zeros, and the new LUTs replacing the OPIN instances have unconnected outputs.

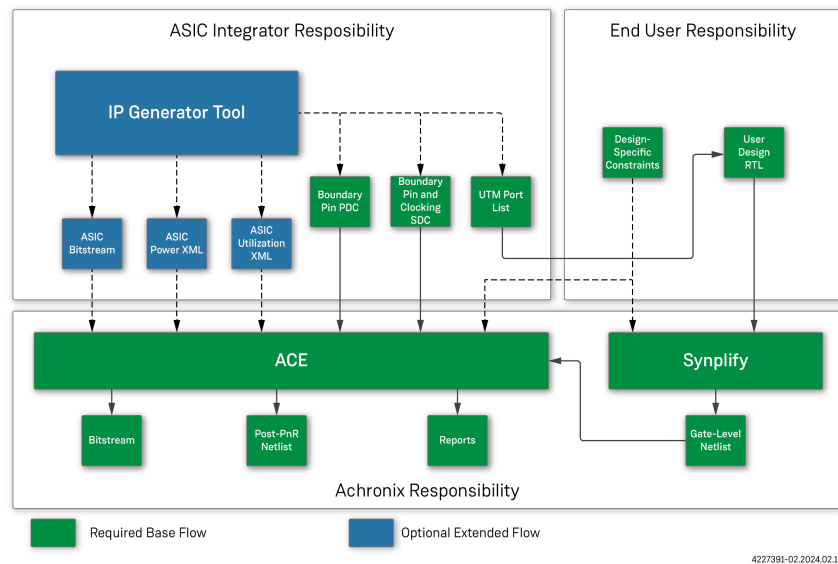


Figure 219 • Stubout Style Input Pad

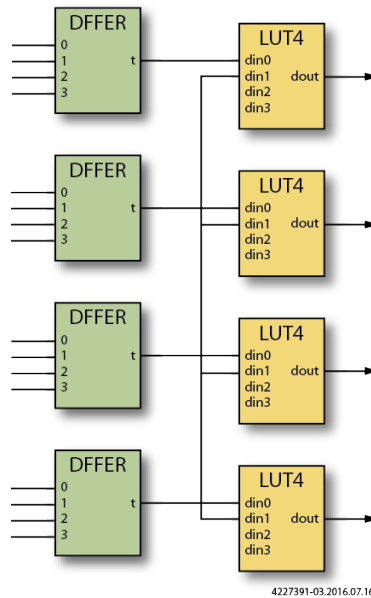


Figure 220 • Stubout Style Output Pad

serialize_dff

The following two schematics illustrate the output of I/O virtualization using the `serialize_dff` style. Notice that the 4-bit IPIN and OPIN buses have been replaced by a single IPIN or OPIN instance. On the input side, the input pads were previously driving a bus of four DFFs. Those DFFs are now driven by the intermediate outputs of a 4-bit shift chain built from DFFs. On the output side, observe that the 4-bit output DFF shift chain is driven by four 2-to-1 MUXes. One input of the MUX comes from the flops originally driving the outputs, while the other input of each MUX is driven by the output of the previous stage of the shift chain. A new IPIN instance has been created to drive the select pin if these MUXes.

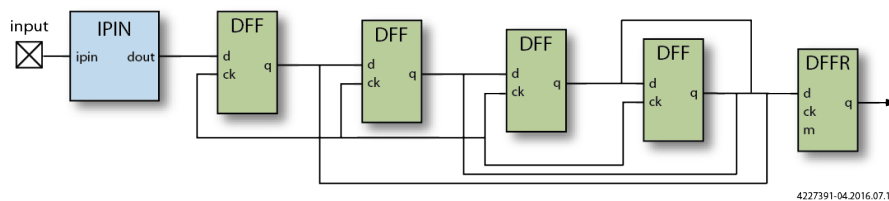


Figure 221 • serialize_dff Style Input Pad

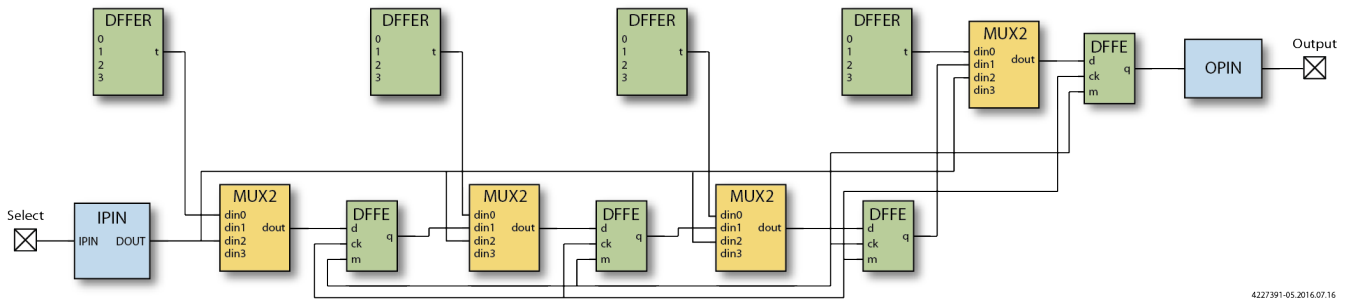


Figure 222 • *serialize_dff* Style Output Pad

serialize_lut

The following two schematics illustrate the output of I/O virtualization using the *serialize_lut* style. Notice that the 4-bit IPIN and OPIN buses have been replaced by a single IPIN or OPIN instance. On the input side, the input pads were previously driving a bus of four DFFs. Those DFFs are now driven by the intermediate outputs of a 4-bit shift chain built from LUTs. On the output side, observe that the 4-bit output LUT shift chain is driven by four 2-to-1 MUXes. One input of the MUX comes from the flops originally driving the outputs, while the other input of each MUX is driven by the output of the previous stage of the shift chain. A new IPIN instance has been created to drive the select pin if these MUXes.

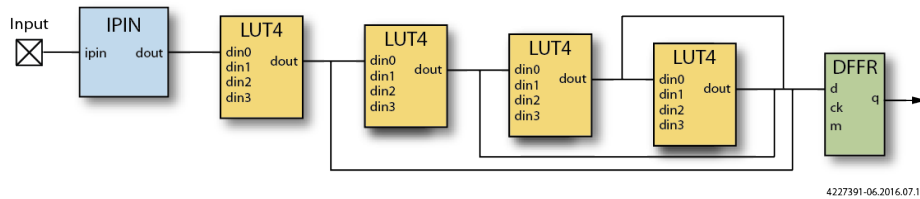


Figure 223 • *serialize_lut* Style Input Pad

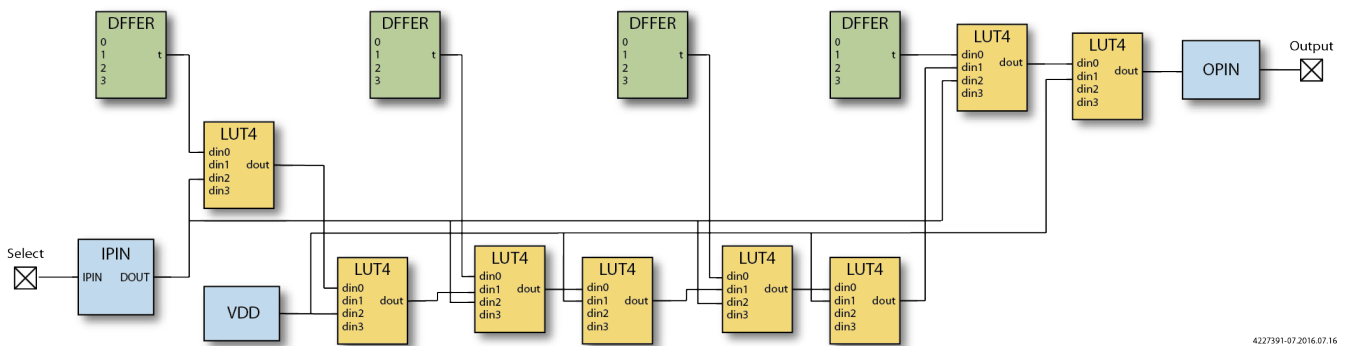


Figure 224 • *serialize_lut* Style Output Pad

Accessing Help

ACE provides a number of ways to access help information, including context-sensitive help and a built-in copy of this user guide document.

Accessing Context-Sensitive Help

ACE provides brief context-sensitive help for most parts of the application. This contextual help typically contains a brief description of the view, dialog, etc., followed by a list of hyperlinks to relevant sections within the ACE User Guide.

To cause the context-sensitive help to be shown, simply press the **F1** key in Windows, or **Shift+F1** in Linux, and the contextual help appears in a view on the right.

The following is an example of what appears when contextual help is opened while the **Projects view** (page 117) has focus:

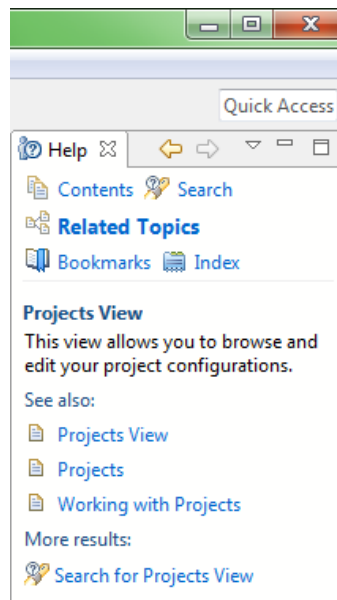



Figure 225 • Context-Sensitive Help Example

Navigating Help Topics

Help topics (corresponding to sections within the ACE User Guide) can be browsed using the Help window or Help view. Choosing which to use is a matter of preference; the Help view is displayed within the workbench like any other view, and is good for quick help lookups. The Help window is (as the name implies) a separate window from the rest of ACE and can be individually maximized, thus allowing easier reading of larger quantities of content.

Using the Help Window






The Help window is separate from the workbench, used exclusively for browsing and searching help content. To open the window, select **Help** → **Help Contents** from the main menu. This action opens the help window with the () **Contents** tab visible in the left frame, which shows the table of contents.

Caution!


There is a known bug on the Linux platform in the application frameworks underlying ACE that might cause view/editor tab movements to detach instead of docking when the Help window is open. See the [Troubleshooting \(page 761\)](#) section for more details, including several workarounds.

Navigating the Help Window



Table of Contents

1. In the left frame, select the () **Contents** tab.
2. To find the topic to be read in the table of contents:
 - a. Click to expand the subtopics.
 - b. Click in the desired topic to have it displayed in the frame on the right.
3. Some topics provide links to additional related topics within (of after) their content. Click these links to learn more.
4. Use the () **Back** and () **Forward** buttons (above the right frame) to navigate back and forth among the recently viewed topics. These buttons behave the same way as in Web browsers.
5. Use the () **Home** button (above the right frame) to return to the help home page in the () **Contents**.



Searching

To quickly locate topics on a particular subject in the documentation, enter a query in the **Search** field at the top of the window. Search results are displayed in the left frame on the () **Search Results** tab. For more details, see [Searching Help \(page 475\)](#).

Synchronizing

Clicking the () **Show in Table of Contents** button above the right frame selects that topic for that page in the **Contents** tree in the left frame (useful when navigating search results when the tree may be out of sync). The () **Link with Contents** button above the left frame in the **Contents** tab keeps the navigation tree synchronized to the current topic shown in the right frame.

Maximizing and Restoring Help Frames

The two main frames of the Help window can each be maximized to take up the entire window. To maximize a frame, click the () **Maximize** button in the frame toolbar, or double-click any blank part of the toolbar for that frame. To return the frame to its original size, click the () **Restore** button or double-click the toolbar again.

Using the Help View

The Help view provides the same features as the Help window, but does it in a single view panel within the **Workbench** (page 6) instead of in a separate window.

Searching Help

The help system includes a search engine that can run simple or complex queries on the documentation to help locate the desired information. To search help:

1. From the main menu, select **Help** → **Search**.
2. Type the word or phrase for the search subject.
3. Click **GO** or press **Enter**. The list of results are displayed below in the left frame (within the **Search Results** tab).
4. Click the topic in the list of results to view the content.

Alternately, searches can be initiated within the Help window using the **Search** field at the top left of the window.



Refining the Search Results

Reducing the Scope of the Search


Sites licensed for both Speedcore and Speedster devices can narrow the **Scope** of a search by restricting the scope to only the preferred user guide(s).

Changing the Appearance of the Search Results

Two buttons on the search results toolbar can be used to change the way results are displayed:

- The () **Show result categories** button, when clicked, causes the results to be grouped by book (this action only has a noticeable effect at sites licensed for both Speedcore and Speedster devices, i.e., when both ACE User Guides are available).
- The () **Show result descriptions** button, when clicked, causes a brief description of each result to be shown.

Highlighting Search Terms

By default, when a search result is selected, the search terms used to find the document are highlighted in the document content. Clicking the () **Highlight Search Terms** toolbar button toggles this feature on and off. This button is available in both the help window and the help view. The state of this button is remembered in both views when displaying subsequent search results.

Search Query Syntax

Follow these expression rules for searching local help content:

- The following stop words are common English words which are ignored (not searched for) if they appear in the search expression:
 - a, and, are, as, at, be, but, by, in, into, is, it, no, not, of, on, or, s, such, t, that,

the, their, then, there, these,
they, to, was, will, with

- The search engine ignores character case (i.e., "Workbench" returns topics that contain "workbench", "Workbench", "WorkBench", and "WORKBENCH")
- Unless otherwise stated, there is an implied AND between all search terms so that topics that contain all the search terms are returned (e.g., "verilog module" returns topics that contain the word "verilog" and the word "module" but does not return topics that contain only one of these words)
- Use "OR" before optional terms (e.g., "project OR implementation" returns topics that contain the word "project", or the word "implementation", or both)
- Use "NOT" before terms to exclude from search results (e.g., "verilog NOT module" returns topics that contain the word "verilog" and do not contain the word "module")

Note

The word "NOT" only works as a binary operator (e.g., "NOT module" is an illegal search query by itself).

- Use "?" for a single-character wildcard and "*" for a multi-character wildcard (e.g., "par?" returns topics that contain "part" or "park", but not "participate", while "par*" returns topics that contain "part", "park", "participate", "pardon", etc.)

Note

The search engine does not accept terms with a wild card at first character position.

- Use double quotation marks around terms which should be treated as a phrase (e.g., "creating projects" returns topics that contain the entire phrase "creating projects" while topics where the words "creating" and "projects" are not consecutive are not returned)
- Punctuation acts as term delimiters (e.g., "plugin.xml" returns hits on topics that contain "plugin" and "xml", so to limit the search to the item, include the double quotes as shown)

Note

The search engine automatically performs "fuzzy" searches and word stemming. Entering "create" returns results including hits on topics that contain "creates", "creating", "creator", etc. To prevent the search engine from stemming a term, enclose the term in double quotes as shown.

Using the ACE SecureShare Tool to Create a Support Zip File

When encountering some problems, the [Troubleshooting \(page 761\)](#) chapter and/or opening a case with Achronix Technical Support (at support.achronix.com/hc/en-us) might not be enough to find a solution. For these instances, ACE includes the SecureShare tool.

The [Create a SecureShare Zip File dialog \(page 160\)](#) gathers all the important information from a user design and collects it into a single ZIP file. Sensitive files may be optionally excluded, or additional files included, before the ZIP file is created by the SecureShare tool. Optionally, the SecureShare tool can even encrypt the information in the ZIP file. Achronix technical support engineers can decrypt any files encrypted by the tool as they help track down and solve the problem.

To use the SecureShare tool:

1. Load (and activate) the **project** (page 222) and **implementation** (page 229) for which help is desired. (See also **Active Project and Implementation** (page 229).) The SecureShare tool gathers the relevant files for whichever project and implementation are active when the tool is started.

 **Note**

If help is needed for multiple projects or implementations, each need to be handled using separate SecureShare ZIP files.

2. Open the **Create a SecureShare Zip File dialog** (page 160) by selecting **Help** → **Start SecureShare**, or by using the keyboard shortcut **Ctrl+Alt+Shift+S**.
3. Examine each file category and **Remove** any files containing sensitive information which should not be transmitted to Achronix.
4. **Add** any additional files which might help Achronix track down the problem. Ideally, add the files to the appropriate categories. If no appropriate category exists, add the files to the "Other" category.
5. Make sure the ZIP file (under the **Configure SecureShare** heading at the top) is pointing to an appropriate directory/filename.
6. Select the **Encrypt included files** checkbox if desired.
7. Click the **Finish** button at the bottom of the dialog.

ACE then creates a ZIP file with the chosen name in the chosen directory. If the Encrypt option was chosen, an additional file with the `.zip.encrypted` file extension is created alongside the (not-encrypted) ZIP file. The resulting `.zip` or `.zip.encrypted` file can be attached to the support request ticket.

Importing and Exporting Preferences

Preference files can be both imported to and exported from ACE, allowing individual or group preferences to be shared or migrated from an existing version of ACE to a newer version when upgrading.

Import Preferences

The Import wizard can be used to import preferences from the file system into ACE. To import a preference file:

1. Select **File** → **Import...**
2. In the Import wizard, select **Preferences** and click **Next**.

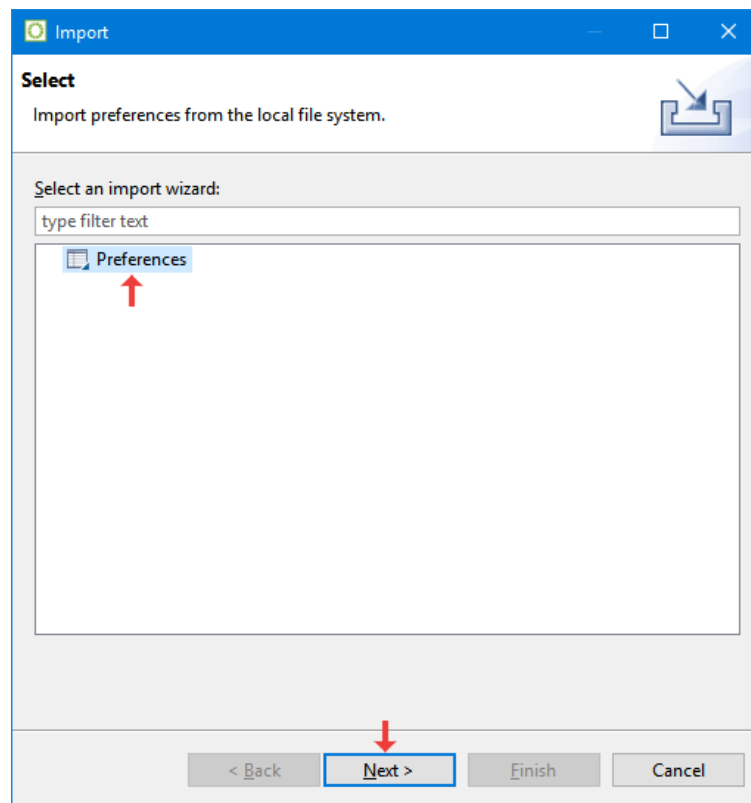


Figure 226 • Import Wizard Select Preferences Example

3. Click **Browse...** and locate the Preferences file on the file system.
4. Select **Import all** to accept all of the preferences defined in the file.
5. Click **Finish**.

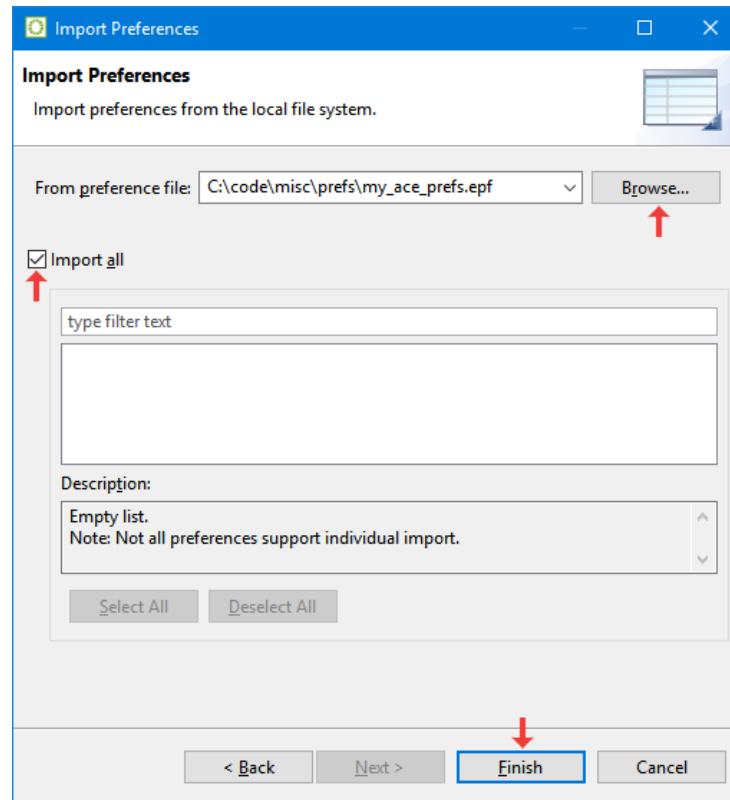


Figure 227 · Import Wizard Locate File Example

Export Preferences

The Export wizard can be used to export preferences from ACE to the file system. To export a preference file:

1. Select **File** → **Export...**
2. In the Export wizard select **Preferences** and click **Next**.

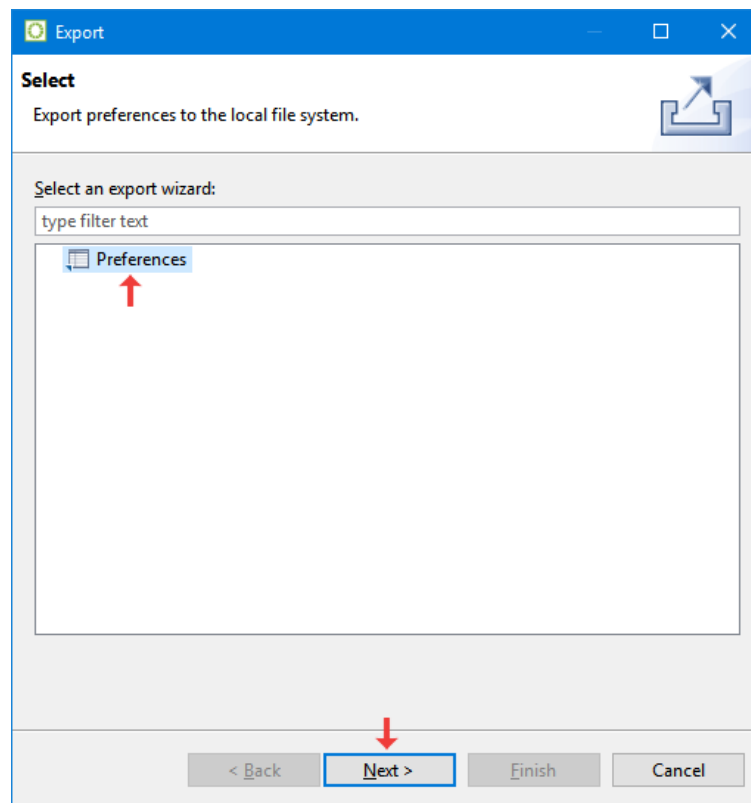


Figure 228 - Export Wizard Select Preferences Example

3. Select **Export all** to add all of the preferences to the file.
4. Click **Browse...** and locate the preferences file on the file system.
5. Click **Finish**.

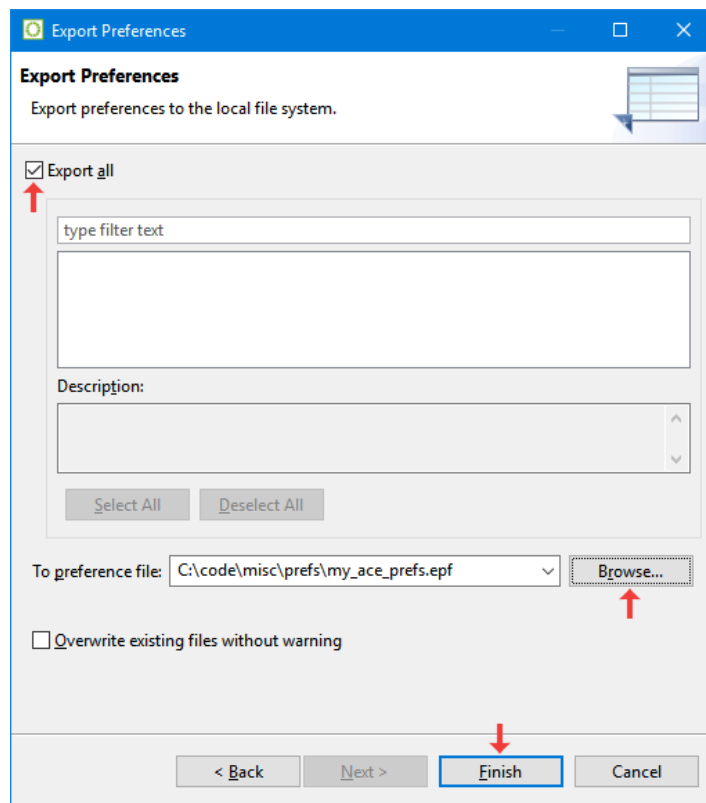


Figure 229 • Export Wizard Locate File Example

Plotting SerDes Receiver Diagrams Using JTAG

Using an open JTAG connection to the SerDes hardware, it is possible to capture Rx diagnostic data and plot Eye diagrams, Histograms, and Bathtub plots.

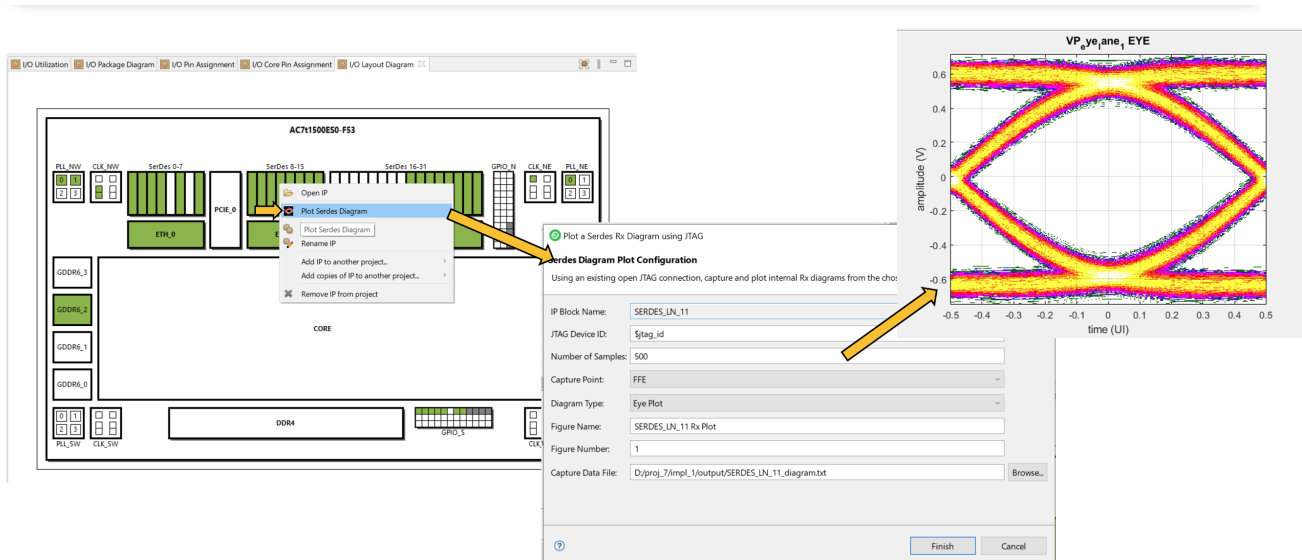


Figure 230 • SerDes Rx Eye Diagram Plot Example

Plotting a SerDes Diagram for a SerDes Lane

Diagrams can be plotted using the following simple steps. For advanced users, the `jtag::capture_serdes_diagram_data` and `jtag::plot_serdes_diagram_data_matlab` Tcl commands may be used directly.

1. Open an ACE project containing the I/O Ring design configuration, and switch to the IP Configuration perspective.
2. Open a valid JTAG connection to the FPGA using the JTAG Tcl commands in the [Tcl Console view \(page 142\)](#):

```
set jtag_id "AC12345"
jtag::open $jtag_id
jtag::ac7t1500_initialize_fcu $jtag_id
```

3. With an open JTAG connection, right-click any configured (green) SerDes Lane in the [I/O Layout Diagram view \(page 67\)](#):

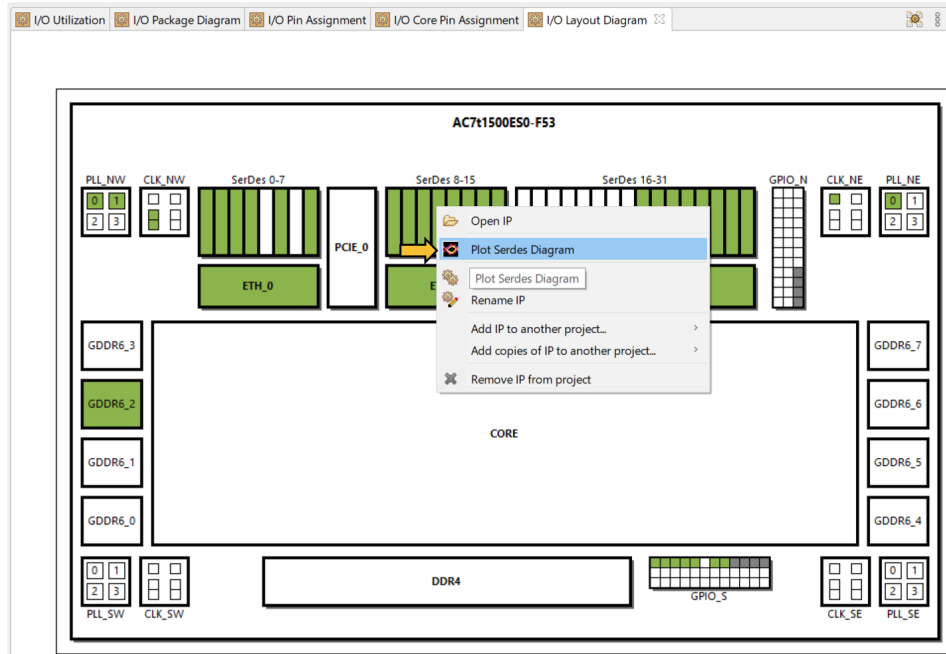


Figure 231 • Plot SerDes Diagram Lane Selection Example

4. Configure the diagram plotting options in the **Plot Serdes Diagram** dialog (page 178):

Plot a Serdes Rx Diagram using JTAG

Serdes Diagram Plot Configuration

Using an existing open JTAG connection, capture and plot internal Rx diagrams from the chosen SerDes lane

IP Block Name:

JTAG Device ID:

Number of Samples:

Capture Point:

Diagram Type:

Figure Name:

Figure Number:

Capture Data File:

Figure 232 • Plot SerDes Diagram Dialog Configuration Example

5. Click **Finish**.

Note

Multiple diagrams can be plotted from the same capture data file using the `jtag::plot_serdes_diagram_data_matlab` Tcl command after completing these steps.

The diagram plotting capture over JTAG takes a long time. For 500 samples, it typically takes upwards of 10 minutes to run.

Using Partial Reconfiguration

This section begins with a high-level overview, and then continues with detailed tutorials.

Partial Reconfiguration Tutorial

Partial Reconfiguration tutorial examples are provided for two different flows. ACE currently supports Partial Reconfiguration for the Speedster7t AC7t1500ES0 FPGA only.

Partial Reconfiguration Flows

- [PR Flow 1: Multi-Project PR Flow Using Partition Export/Import \(page 494\)](#)
- [PR Flow 2: Multi-Project PR Flow Using Keep Out Regions \(page 520\)](#)

ACE currently supports 2 different partial reconfiguration flows. PR flow 1 is the recommended flow.

Overview of PR Flow 1

PR Flow 1 uses the ACE partitions feature. Familiarity with the ACE partition flow is recommended before proceeding with PR flow 1.

The partition import/export flow requires separate projects for the top-level design as well as each PR Core. The partition export flow must first be run in its own Synplify and ACE project for the PR core to be exported as a partition. After the partition export flow is run, ACE generates a placed and routed partition for the PR core. The partition import flow is next run in a separate Synplify and ACE project. In the partition import flow, the top-level design instantiates the previously exported PR core blackbox module and ACE imports the placed and routed data for the PR core into the top-level design.

Note

In the partition import flow, ACE only places and routes the top-level design and not the imported PR cores.

The base bitstream is created in the first partition import run where the previously exported PR cores are imported into the top-level design. On Silicon, the base bitstream is programmed onto the device first. The partial bitstream for the second set of PR Cores is subsequently programmed onto the device. The base bitstream includes the top-level logic as well as the PR Core logic. The partial bitstream is generated in the second partition import run where a different set of exported PR Cores are imported in the same top-level design.

Overview of PR Flow 2

The Multi-Project PR flow using keepout regions is another method to create designs that can be partially reconfigured. This flow involves using separate ACE projects to generate separate base and partial bitstreams for the top-level design and each PR core. On silicon, the base bitstream consisting of only the top-level logic is programmed first. The partial bitstreams for the other PR core can be programmed subsequently. Using this flow, the user design is stitched together at the bitstream level. There is no integration in ACE between the the top-level design and each PR core since the designs are stitched at the bitstream level. As such, this flow comes with a few pitfalls so that only high-level users are recommended to use this flow. PR flow 1 is therefore considered an improvement to PR flow 2.

Flow Comparisons

Table 148 - Pros and Cons For Each PR Flow

Item	PR Flow 1	PR Flow 2
Turn-around Time	<p>Con: PR flow 1 has a slower turn-around time. The base bitstream consists of the top-level design as well as the PR cores. PR Cores must be exported in separate ACE sessions before being imported into the top-level design.</p>	<p>Pro: PR Flow 2 has a faster turn-around time. The base bitstream consists only of the top-level design. PR cores place-and-route can be bypassed. Only a keep-out region is needed.</p>
Integrating Third-Party Accelerators	<p>Pro: PR flow 1, allows integrating third party accelerator cores at a bitstream level and also within ACE. The partition binary file for third-party accelerators can be integrated in an ACE project which allows running checks and timing analysis for the integrated design to ensure that the accelerator is compatible with the top-level design.</p>	<p>Con: PR flow 2 enables integrating third party accelerator cores only at the bitstream level. Third party accelerators cannot be tested until programmed at the bitstream level.</p>
Bitstream Stitching	<p>Pro: In PR flow 1, ACE integrates the PR core project together with the top-level project using the partitions feature. A series of checks are performed to ensure that the base and partial bitstreams can be stitched together without issue.</p>	<p>Con: In PR Flow 2, bitstreams might not be stitched together correctly if any of the ACE projects are configured incorrectly. The base bitstream and partial bitstreams are created in separate independent ACE projects so if all user projects are not set up correctly, ACE cannot catch the errors.</p> <ol style="list-style-type: none"> 1. Scenario 1 – if clock nets used in the partial bitstream are not pre-routed in the base bitstream, the clock signal cannot be correctly connected to the PR cores after partial reconfiguration. 2. Scenario 2 – if clock nets on different clock tracks are pre-routed in the partial and base bitstream, the clock signals cannot be correctly connected to the PR Cores after partial reconfiguration. 3. Scenario 3 – if keep-out regions are not set on clusters in the base bitstream that are used by PR cores, the top-level logic might be lost after partial reconfiguration.

Item	PR Flow 1	PR Flow 2
Closing Timing	<p>Pro: In PR flow 1, ACE integrates the PR core project together with the top-level project during the partition import run and also performs timing analysis for the integrated design.</p>	<p>Con: In PR Flow 2, the design might not meet timing on paths between the top-level and PR core after the partial bitstream is programmed because timing analysis is performed separately in projects for the top-level design and PR cores (timing delays are different on the east and west sides of the Speedster7t AC7t1500ES0 FPGA.)</p>
Simulation	<p>Pro: Simulating the entire design is possible in the partition import run.</p>	<p>Con: Simulating the entire design within an ACE project is difficult because there is no integration between separate ACE projects. Though not impossible, a lot of manual work is required.</p>

Flow Diagrams

The following sequence outlines the high-level steps needed to partially reconfigure PR core A as PR core B.

PR Flow 1: Multi-project PR Flow Using Partition Export/Import

In the operating procedure for this example, the base bitstream in step 3 is programmed onto silicon first. This base bitstream includes the top-level logic and the PR Core A logic. To partially reconfigure PR Core A with PR Core B, the partial bitstream created in step 6 is programmed subsequently.

1. Export PR core A as a partition using ACE.

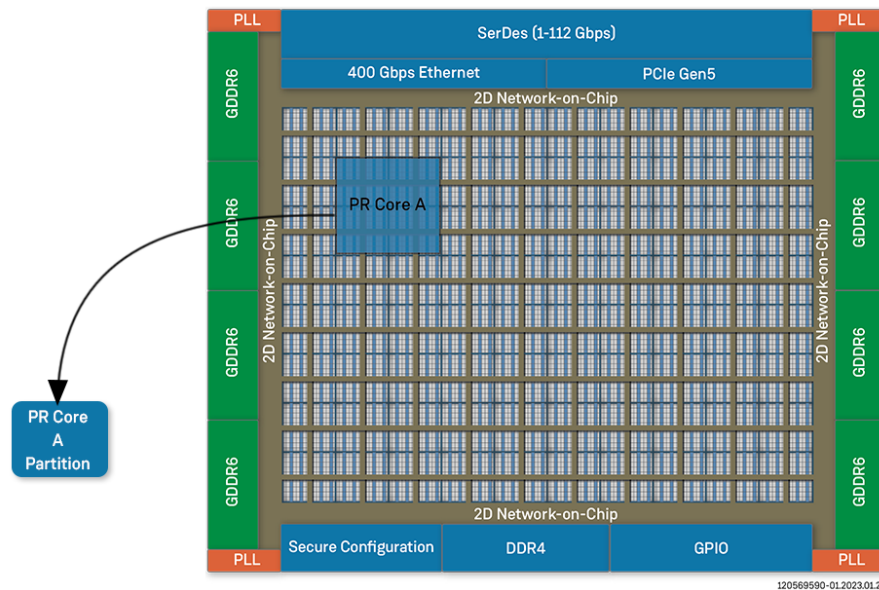


Figure 233 • Export PR Core as a Partition Example

2. Import PR core A partition into top-level design.

3. Generate a base bitstream for the entire device.

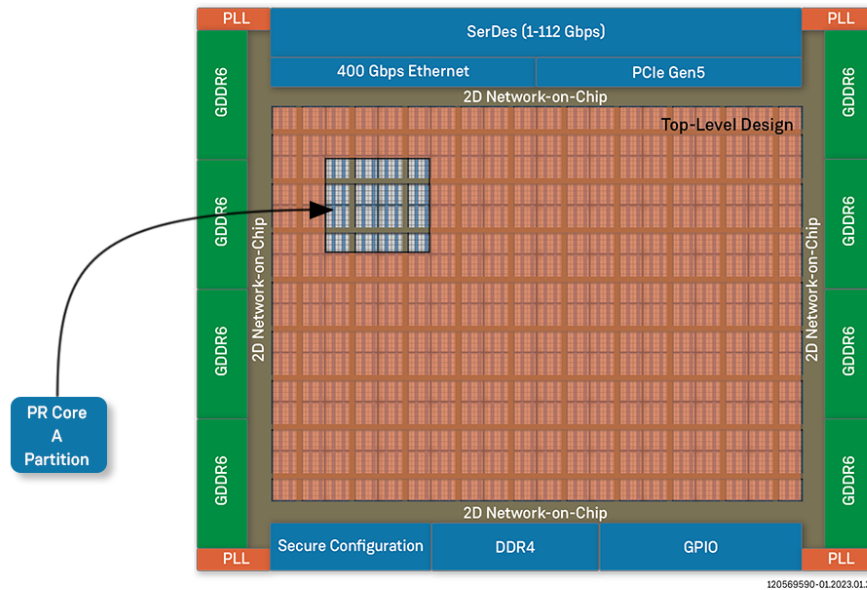


Figure 234 · Import PR Core to Top Level Example

4. Export PR core B as a partition using ACE.

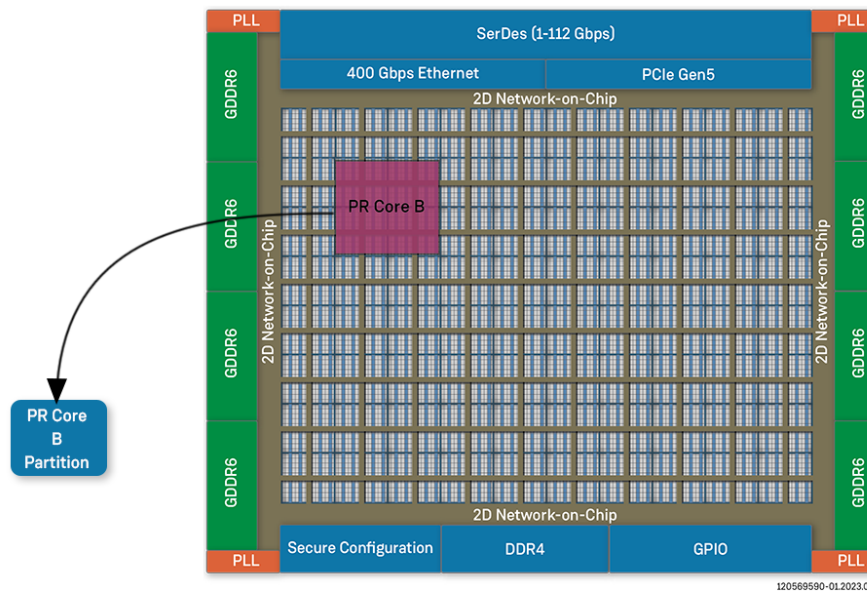


Figure 235 · Export PR Core as a Partition Example

5. Import the PR core B partition into the same top-level design.
6. Generate a partial bitstream only for PR core B.

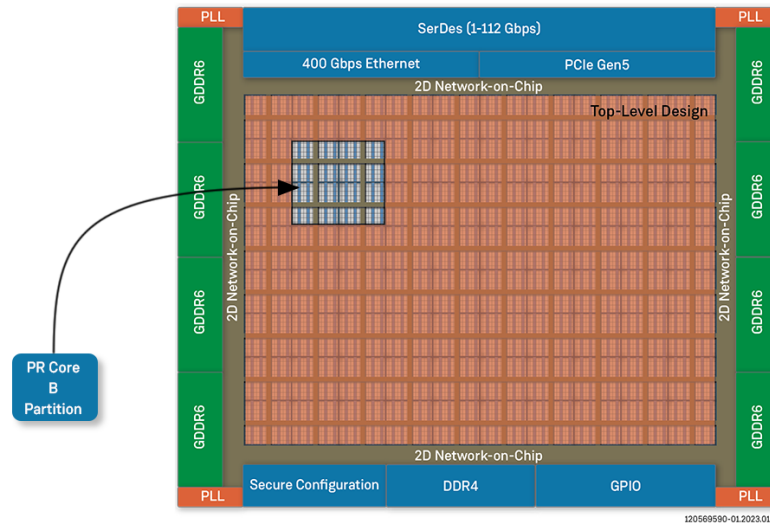


Figure 236 - Generate Partial Bitstream Example

PR Flow 2: Multi-project PR Flow Using Keep Out Regions

In the operating procedure for this example, the base bitstream created in step 3 is programmed onto silicon first. Unlike PR Flow 1, the base bitstream only contains the top-level logic (not the PR Core A logic). The partial bitstream for PR Core A (created in step 1) is programmed next. To partially reconfigure PR Core A with PR Core B, the partial bitstream for PR Core B (created in step 2) is programmed subsequently.

1. Generate a partial bitstream for PR core A using ACE.

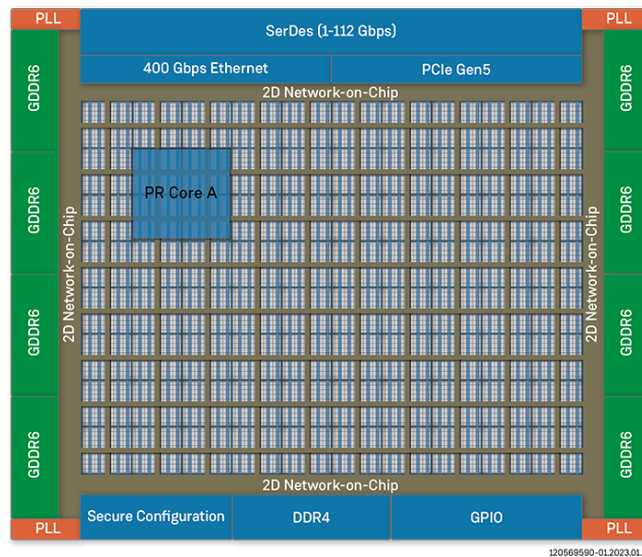


Figure 237 - Generate Partial Bitstream Example

- 2. Generate a partial bitstream for PR core B using ACE.

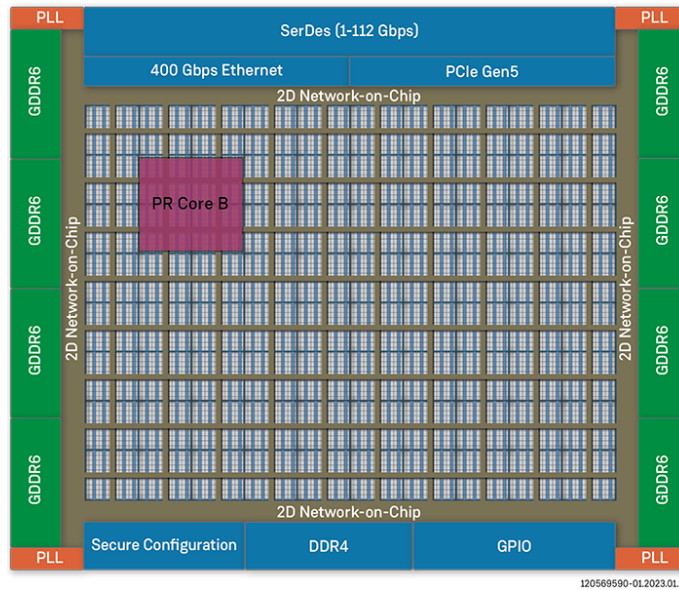


Figure 238 • Generate Partial Bitstream Example

- 3. Generate a base bitstream for the top-level design using ACE with keep out regions for the PR cores.

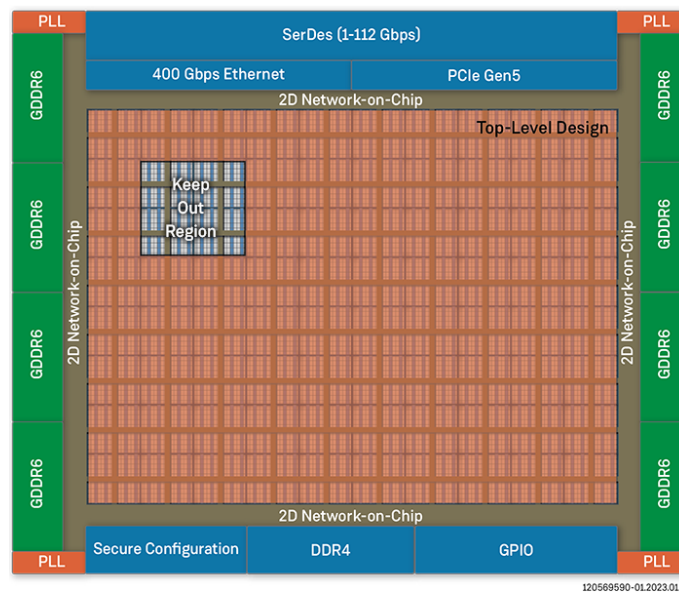


Figure 239 • Generate Bitstream Example

Design Details

This tutorial uses a simple example design to illustrate running a design through PR Flow 1. In this tutorial, three separate bitstreams are created:

1. Base Bitstream (Top-level design)
2. Partial Bitstream 1 (PR Core 1)
3. Partial Bitstream 2 (PR Core 2)

Base Bitstream (Top-level design)

RTL files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/rtl/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src/rtl/...

Constraint files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/constraints/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src/constraints/...

The top-level design instantiates the user static logic. Static Logic refers to any user design logic which is not intended to be reconfigured in the PR flow. The base bitstream is generated using the top-level design. The top-level design in this tutorial demonstrates a 2-bit binary up-counting LED display to indicate that the board and FPGA are operating properly upon powerup. The RTL along with the device-specific netlists and constraints are available under the <ACE_install_dir>/Achronix/examples/partial_reconfiguration directory.

Other than just instantiating the user static logic, the top-level design also must instantiate any shared system controller logic, I/O Ring, and I/O interfaces used in the other PR cores. Clock nets used in each PR core to be partially reconfigured must also be carefully pre-routed since clock nets are also routed on the core fabric HW clock network, which is not reconfigurable. Also, a keep-out region for each PR Core must be created to prevent any logic and routing from the top-level design from entering the cluster(s) used by the PR Cores in the partial bitstreams.

Partial Bitstream 1 (PR Core 1A)

RTL files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/rtl/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/rtl/...

Constraint files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/constraints/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/constraints/..

PR core 1 consists of a 32-bit adder with outputs connected to a 32-bit register. This 32-bit register is connected to the NAP initiator in the cluster. The NAP Initiator can be used to read the 32-bit adder output register via the 2D NoC with the JTAG interface or with PCIe. In this tutorial, the JTAG interface is used to read the user registers. There is also a reset register that can be written to in this design. The reset register drives all resets in PR core 1.

Partial bitstreams must not contain any I/O ring programming. Clock nets also must be pre-routed on the clock network on a specified track number. The top-level design must also pre-route the same clock net on the same track number to the same cluster where PR core 1 is placed. A cluster map also must be set to identify which core fabric cluster(s) are to be re-programmed. The following sections in this tutorial cover this in more detail.

Partial Bitstream 1 (PR Core 1B)

RTL files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b/src/rtl/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1b/src/rtl/...

Constraint files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/constraints/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/constraints/..

PR core 1B is identical to PR core 1A except that the 32-bit adder increments by 2 instead of 1 during each read.

Partial Bitstream 2 (PR Core 2A)

RTL files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/src/rtl/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/src/rtl/...

Constraint files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/src/constraints/...
2. PR fFlow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/src/constraints/...

PR core 2A is identical to PR core 1A except that the 32-bit adder instead is implemented as a 32-bit subtractor. The 32-bit subtractor decrements by 1 at each NAP read.

Partial Bitstream 2 (PR Core 2B)

RTL files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/src/rtl/...

2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b/src/rtl/...

Constraint files:

1. PR flow 1 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/src/constraints/...
2. PR flow 2 - <ACE_INSTALL_DIR>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b/src/constraints/...

PR core 2B is identical to PR core 2A except that the 32-bit subtractor now decrements by 2 instead of 1 at each NAP read.

Reset Configuration Options

This tutorial uses option 1, below, but there are several ways resets for a partial reconfiguration design can be configured. The different methods are listed below, ordered by recommendation (1 being the most and 4 being the least recommended method).

1. Use a NAP to control reset via the 2D NoC. This can be performed using host software (e.g., `jtag::nap_axi_write` or PCIe), or top-level control logic. This reset scheme is utilized by the demo in this tutorial. All nets driven by reset are self contained within the PR zone.
2. Use a DFF chain to de-assert reset when the cluster enters user mode.
 - Use init values on DFFs to toggle the reset when a cluster enters user mode.
 - This is an example showing how to instantiate a DFF chain for an active-low reset design:

```

wire rst_wire_1;
wire rst_wire_2;
wire i_rst;

// First DFF in chain
(* must_keep=1 *) ACX_DFF #(
    .init    (1'b1)
) reset_dff_1 (
    .q      (rst_wire_1),
    .d      (),
    .ck     (i_clk)
);

// Second DFF in chain
(* must_keep=1 *) ACX_DFF #(
    .init    (1'b0)
) reset_dff_2 (
    .q      (rst_wire_2),
    .d      (rst_wire_1),
    .ck     (i_clk)
);

```



```
// Third DFF in chain (q-pin drives the PR Core reset)
(* must_keep=1 *) ACX_DFF #(
    .init    (1'b0)
) reset_dff_3 (
    .q      (i_rst),
    .d      (rst_wire_2),
    .ck     (i_clk)
);
```

- In this example, `reset_dff_3` drives the reset with `1'b0` for 2 clock cycles before being de-asserted to `1'b1`.
 - The disadvantage of this reset scheme is that the resets cannot be toggled by the user.
3. Use top-level CLK I/O pads to drive the reset to the global clock trunk using clock-preroute constraints.
- Pre-route the reset net on a specified track on the global clock trunk.
 - PDC constraints:

```
#Create a clock IPIN for your reset and place it on a clock IPIN tile
create_boundary_pins {p:i_reset_n} {i_reset_n_ipin} -clock
set_placement -fixed -batch {p:i_reset_n} {d:i_user_06_00_trunk_00[16]}

#Pre-route the reset net on clock track 1 to the PR Zone defined for your PR
Core.
#add_clock_preroute "i_reset_n_ipin_net" 1 -placement_regions "PR_ZONE_2_7"
```

- Disadvantage: The reset net is routed over the clock network so there is one less clock track available.
4. Use top-level Data I/O pads to drive the reset and use the `data2clk` path to route the reset to the global clock trunk using clock-preroute constraints.
- Pre-route the reset net on a specified track on the global clock trunk.
 - PDC constraints

```
#Create a data IPIN for your reset and place it on a data IPIN tile
create_boundary_pins {p:i_reset_n} {i_reset_n_ipin}
set_placement -fixed -batch {p:i_reset_n} {d:i_user_11_09_lut_13[15]}

#Pre-route the reset net on clock track 1 to the PR Zone defined for your PR
Core.
#add_clock_preroute "i_reset_n_ipin_net" 1 -placement_regions "PR_ZONE_2_7"
```

- Disadvantages: There must be a path available from the data IPIN to the central clock trunk. If there are no paths available because all clusters are blocked off and in use, the router cannot pre-route this reset net.

PR Flow 1: Multi-Project PR Flow Using Partition Export/Import

Introduction

The Multi-Project flow with partition export/import is the recommended flow for partial reconfiguration.

The partition import/export flow requires separate projects for the top-level design as well as for each PR core. The partition export flow must first be run in its own Synplify and ACE project in order for a PR core to be exported as a partition. After the partition export flow is run, ACE generates a placed-and-routed partition for the PR core. The partition import flow is next run in a separate Synplify and ACE project in which the top-level design instantiates the PR core blackbox module that was previously exported. ACE imports the placed-and-routed data for the PR core into the top-level design.

Note

In the partition import flow, ACE does not place-and-route the imported PR cores, only the top-level design.

Moving Partitions

Partitions are also very useful for replicating and moving logic. Each PR core can be placed-and-routed, and the timing-closed, once in one location of the fabric, and then stamped down multiple times and relocated throughout the fabric. This is accomplished by instantiating multiple copies of the exported PR core partition in the top-level design. ACE then imports the timing-closed PR core partition database. Initially, each instance of the PR core is placed on top of each other but ACE has a special feature which allows partitions to be moved/re-mapped to other locations in the fabric. This is made possible using relative placement and routing to map to the repetitive cluster-based architecture of the fabric.

High-Level Design Flow

The bottom-up flow must be used for PR flow 1. This involves exporting a partition for each of the PR cores and importing them into the top-level design. The base bitstream can be generated for the top-level design to contain each of the PR cores. Subsequent partial bitstreams can be generated from the top-level design project or from each PR core project.

Please follow this high-level design flow carefully:

1. [Export Partition for PR Core 1A \(page 495\)](#)
 - a. [Synthesize PR Core 1A Using Synplify \(page 495\)](#)
 - b. [Run PR Core 1A Through Run Prepare in ACE \(page 497\)](#)
 - c. [Set Up Region Constraints for PR Core 1A \(page 498\)](#)
 - d. [Set Up Clock Pre-Routing Constraints for PR Core 1A \(page 499\)](#)
 - e. [Run PR Core 1A Through Place-and-Route in ACE \(page 501\)](#)
 - f. [Run Final Sign-Off Timing Analysis for PR Core 1A \(page 501\)](#)
 - g. [Save Region/Clock Pre-Routing Constraints Back to PDC File \(page 502\)](#)
 - h. [Ensure ACE Generated Blackbox File and Exported Partition for PR Core 1A \(page 503\)](#)
2. [Export Partition for PR Core 2A \(page 503\)](#)
 - a. [Repeat Procedure for PR Core 2A \(page 503\)](#)
3. [Importing PR Core 1A and 2A Partitions and Generating Base Bitstream for Top-Level Design \(page 504\)](#)

- a. [Create I/O Ring Configuration for Target Board \(page 504\)](#)
 - b. [Synthesize Top-Level Design Using Synplify \(page 508\)](#)
 - c. [Run Top-Level Design Through Run Prepare in ACE \(page 509\)](#)
 - d. [Set Up Clock Pre-Routing Constraints \(page 510\)](#)
 - e. [Set Up Region Constraints \(Optional\) \(page 511\)](#)
 - f. [Run Top-Level Design Through Place-and-Route in ACE \(page 513\)](#)
 - g. [Run Final Sign-Off Timing Analysis in ACE \(page 514\)](#)
 - h. [Generate Base Bitstream for Top-level Design \(page 515\)](#)
 - i. [Save Region/Clock Pre-Routing and Partition Placement Constraints Back to PDC File \(page 516\)](#)
4. [Generate Partial Bitstream for PR Core 1B and 2B \(page 516\)](#)
 - a. [Export PR Core 1B as Partition \(page 516\)](#)
 - b. [Export PR Core 2B as Partition \(page 517\)](#)
 - c. [Import PR Core 1B and 2B Into Top-Level Design \(page 517\)](#)
 - d. [Generate Partial Bitstream for PR Core 1B and 2B in Top-level Design \(page 518\)](#)
 5. [Bitstream Programming Sequence \(page 519\)](#)
 - a. [Apply Power to Board \(page 519\)](#)
 - b. [Program Top-Level Base Bitstream Containing PR Core 1A and 2A \(page 519\)](#)
 - c. [Run PR Core 1A and 2A Test Scripts \(page 520\)](#)
 - d. [Program Partial Bitstream for PR Core 1B and 2B \(page 520\)](#)
 - e. [Run PR Core 1B and 2B Test Scripts \(page 520\)](#)

Export Partition for PR Core 1A

Synthesize PR Core 1A Using Synplify

1. Navigate to the following directory:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_1/pr_core_1a/src
```

Note

<ace_install_dir> is the directory path where ACE was installed.

Below this directory should be the following two directories:

- rtl
 - constraints
2. In the rtl directory, create a new Synplify project and add all of the RTL files.
 3. Add the pr_core_1a_timing.sdc file to the constraints directory.
 4. Create a compile point for the PR core 1 module to be exported as a partition.

Note

The compile point cannot be set for the top-level module.

5. Add the `pr_core_1a_export_partitions.fdc` file from the `constraints` directory to the project. The compile point is set in the file as follows:

```
define_compile_point {v:work.pr_core_1a} -type {locked}
```

6. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
7. Set **Part** to **AC7t1500ES0**.
8. Set **Package** to **F53**.
9. Set **Speed** to **C2**.
10. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_core_1a_top**.
11. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_core_1a_top**.
12. Set **Include Path Order** to `<ace_install_dir>/libraries`.
13. Add the `AC7t1500ES0_synplify.sv` file to the project, this file is under `<ace_install_dir>/libraries/device_models`.

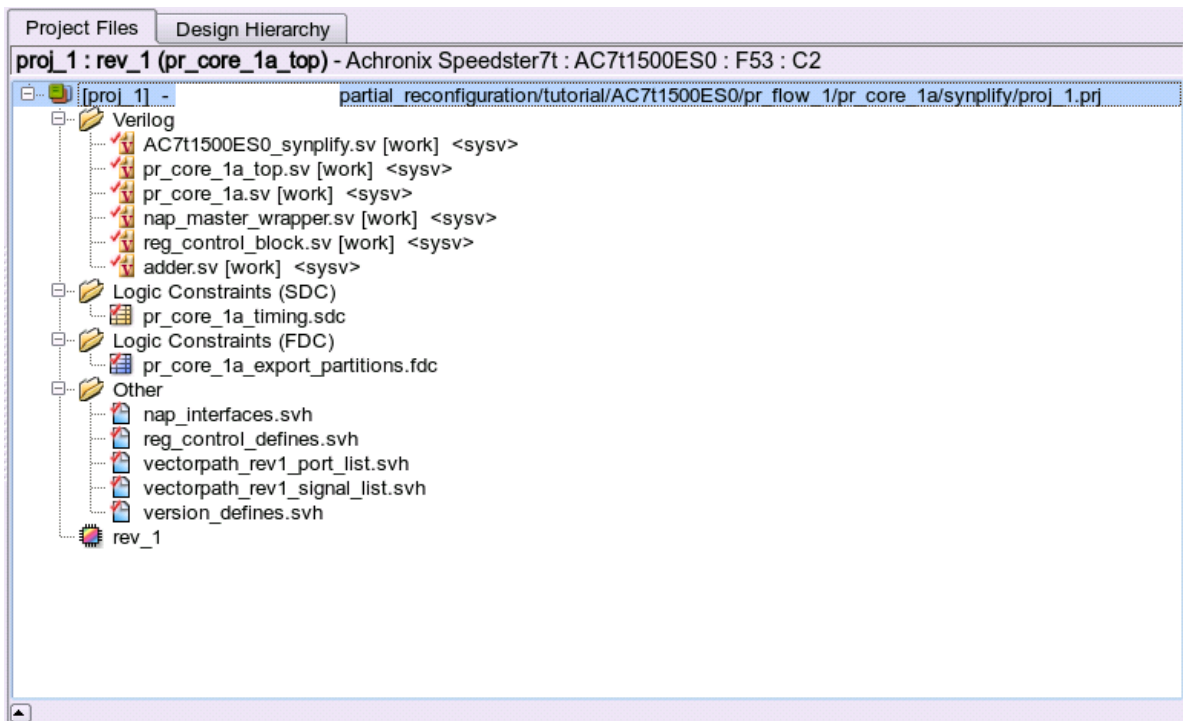


Figure 240 • Project Example

14. Run and compile the design using Synplify.
15. After successful completion, Synplify generates a gate-level netlist under `<synplify_out_dir>/rev_1/pr_core_1a_top.vm`.
16. Synplify also creates a `.prt` file indicating that the compile point was successfully set for PR core 1A. This file is generated under `<synplify_out_dir>/rev_1/pr_core_1a_top_partition.prt`.

Run PR Core 1A Through Run Prepare in ACE

1. In the **Projects** view, click the () **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the **Projects** view, click the () **Add Source Files to Project** button.
4. Add the following files to the project:
 - `pr_core_1a_top_partition.prt` (the partition info file created by Synplify in step 1a)
 - `pr_core_1a_top.vm` (the synthesized netlist created by Synplify in step 1a)
 - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/ioring_design/pr_core_1a_ioring.pdc`
 - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/src/ioring_design/pr_core_1a_ioring.sdc`
5. In the **Options** view, navigate to the **Design Preparation** section and ensure that the **Export All Partitions** option is checked and all other options are correctly set.
6. Copy the option values as shown in the following figure.
7. In the **Flow** view, right-click **Run Prepare** and select **Run Selected Flow Step**:

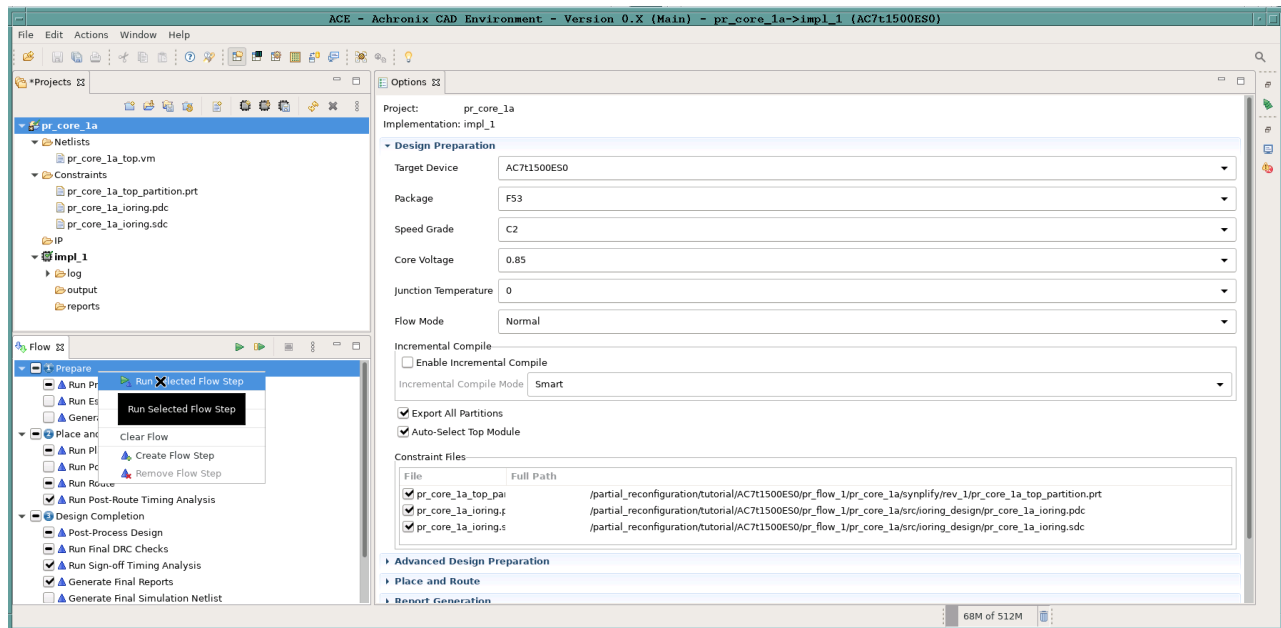


Figure 241 • Run Prepare Flow Step Example

Set Up Region Constraints for PR Core 1A

1. Ensure that the run_prepare flow step has completed successfully.
2. On the **Floorplanner** view, click the **Placement Region Tool** button (highlighted in red in the following image).
3. Click and drag within the Floorplanner to create a PR zone for the PR core.

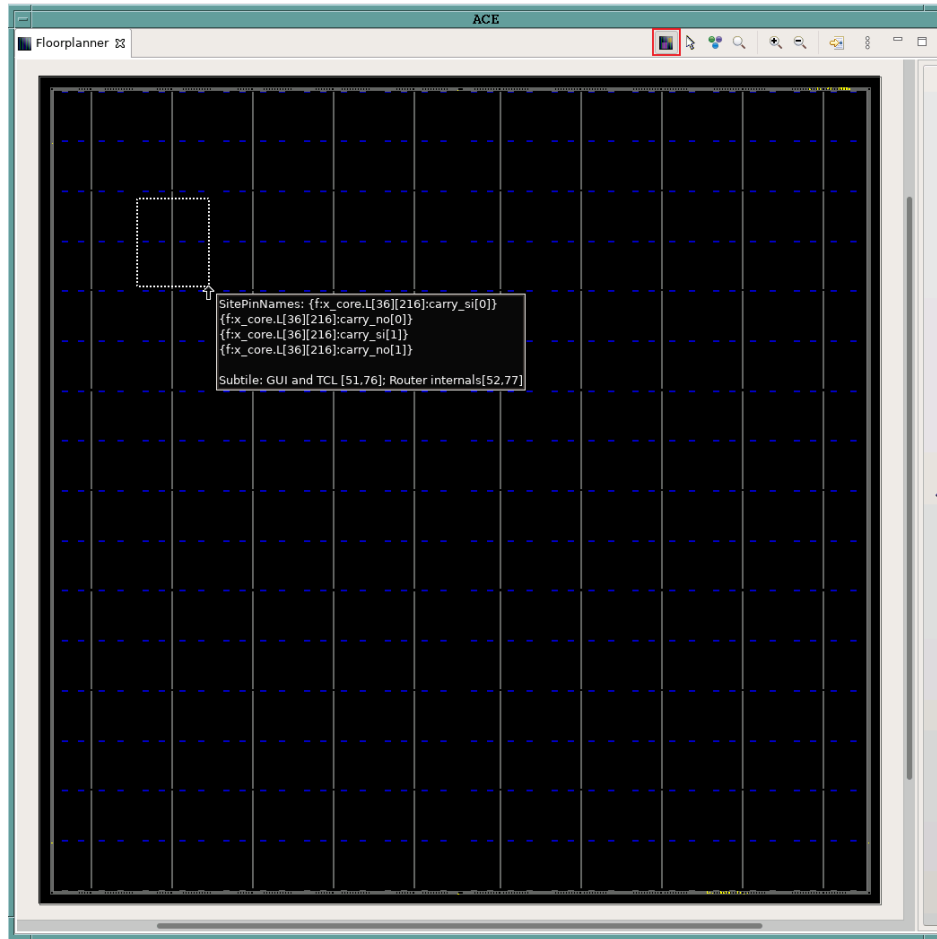


Figure 242 • Creating a Placement Region Zone Example

4. After releasing the left mouse button, the **Create Placement Region** dialog appears.
5. In **Region Name**, enter a name for the PR zone.
6. Set **Region Alignment** to **Snap to Fabric Clusters** (a "fabric cluster" represents the lowest level of granularity at which the core fabric can be partially reconfigured).
7. Set the **Region Type** to **Inclusive** (ensures that all instances are constrained to the region).
8. Select **Include Routing** (ensures that all routed nets are constrained to the region).
9. Select **Is Partial Reconfiguration Zone** (indicates this region is a PR zone).

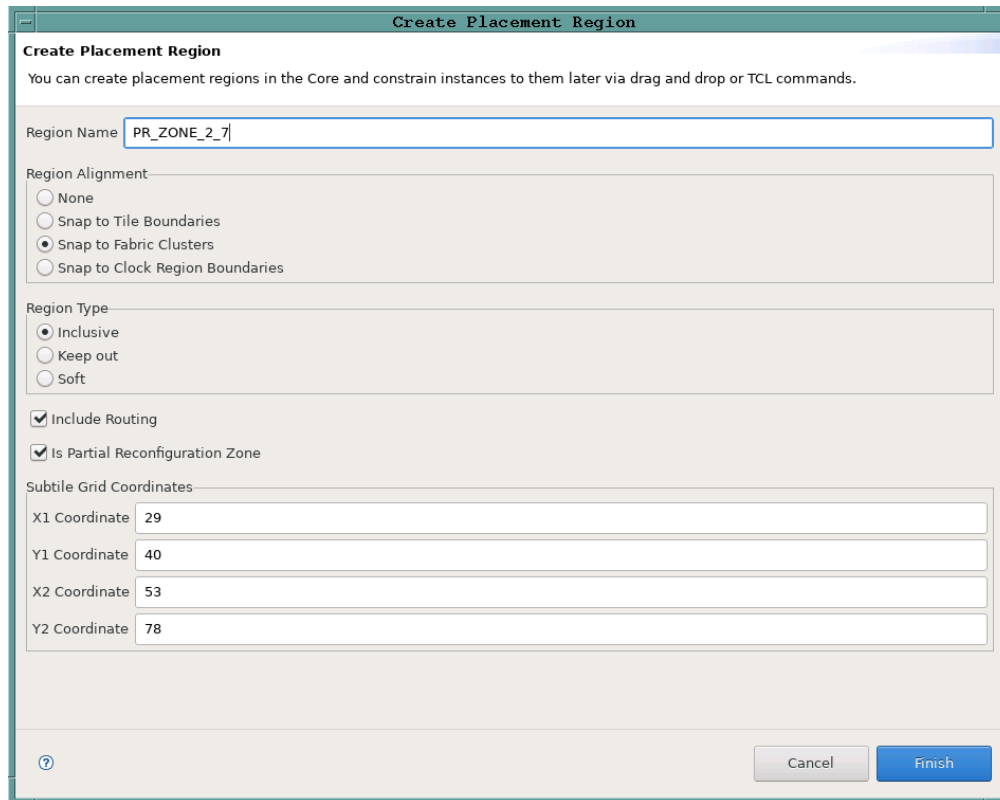


Figure 243 • Create Placement Region Dialog Example

10. Click **Finish**. The Tcl console displays the following:

```
create_region "PR_ZONE_2_7" {29 40 53 78} -snap fabric_clusters -type inclusive
-include_routing -pr_zone
```

11. Add all instances in the PR core to the region. A recommended quick way of doing so is to use the `add_region_find_insts` Tcl command nested with the `find` Tcl command. Run the following example in the Tcl Console to add all instances in PR core 1A to the PR zone (`i_pr_core_1a` is the instance name for the top-level partial reconfigurable module).

```
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts}
```

Set Up Clock Pre-Routing Constraints for PR Core 1A

1. In the **Partitions** view, right-click the cell below the **Clock Pre-Routes** column heading and select **Configure Clock Pre-Routes** as shown:

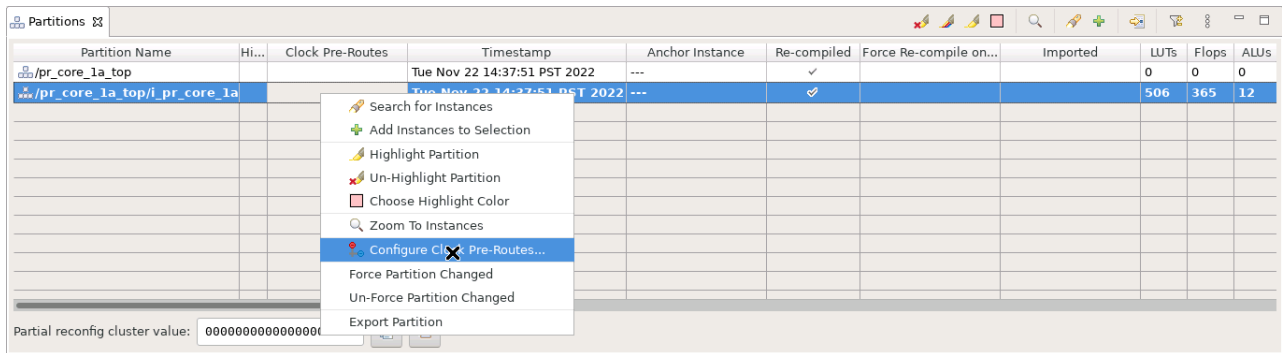


Figure 244 • Configure Clock Pre-Routes Example

- The **Configure Clock Pre-Routes** dialog appears with the `i_clk_pr_1_ipin_net` signal that drives the PR core 1A clock pins pre-routed on clock track 1.

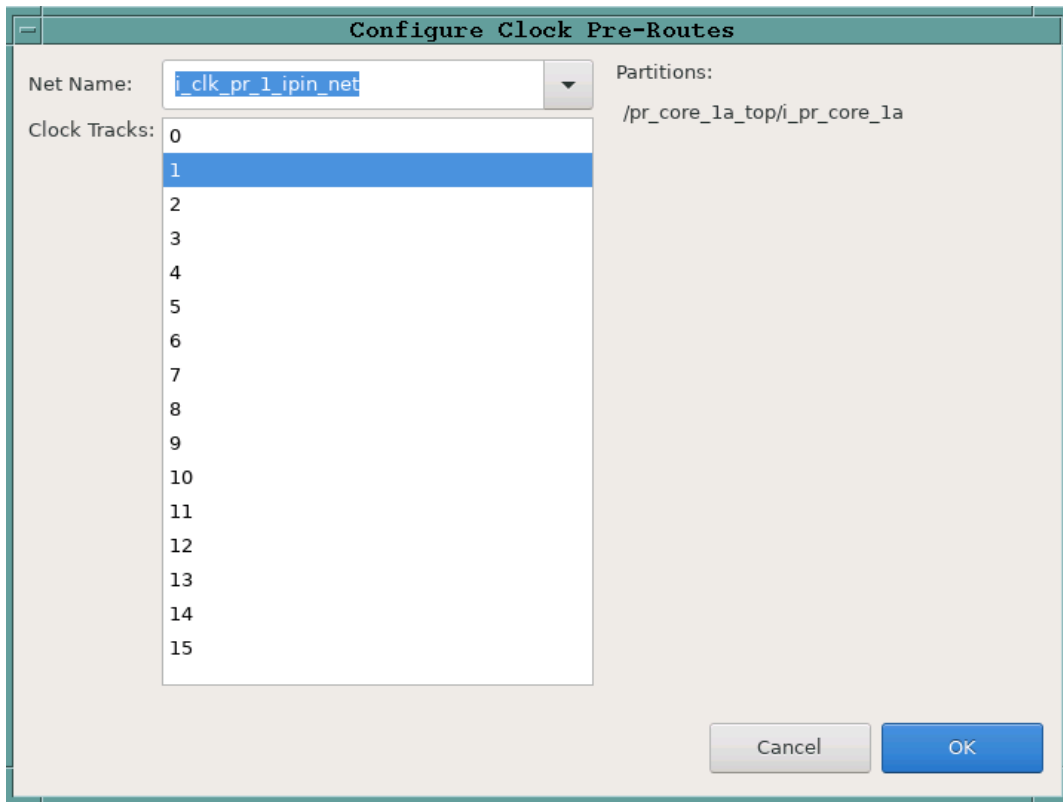


Figure 245 • Configure Clock Pre-Routes Dialog Example

- Click **OK**. The Tcl console displays the following:


```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -partitions { /pr_core_1a_top/
i_pr_core_1a }
```

Run PR Core 1A Through Place-and-Route in ACE

1. In the **Flow** view, right-click **Place and Route** and select **Run Selected Flow Step** as shown:

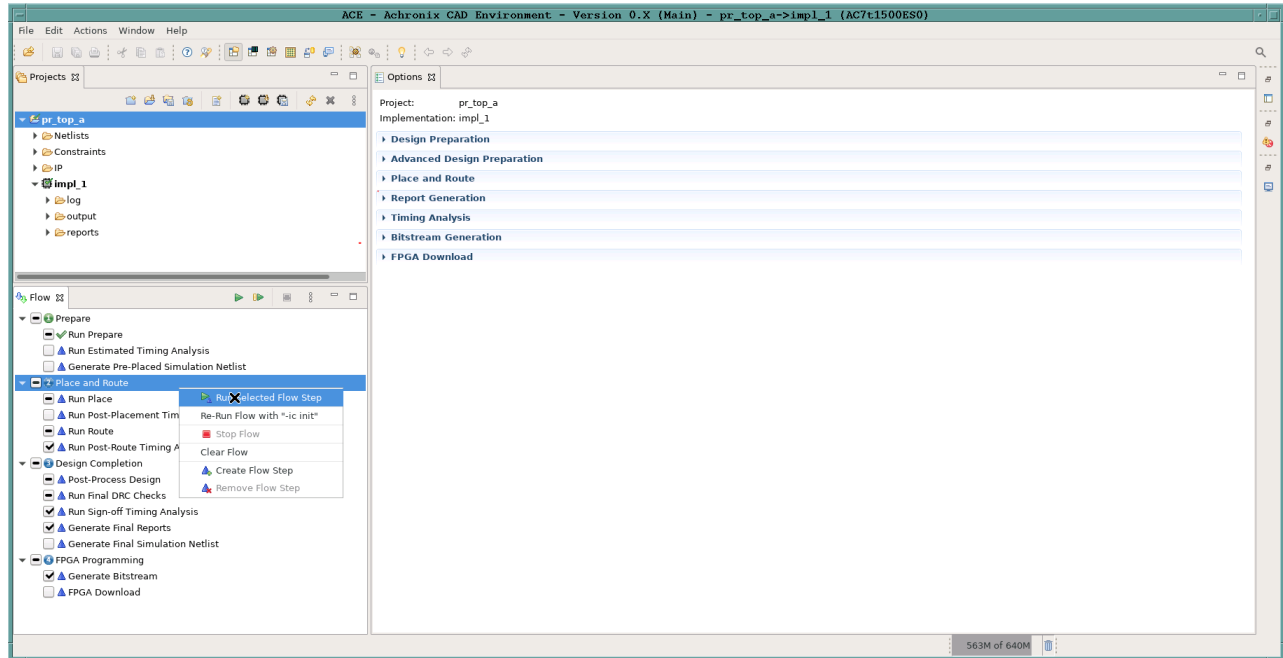


Figure 246 • Run Place and Route Flow Step Example

Run Final Sign-Off Timing Analysis for PR Core 1A

1. Ensure that the place-and-route flow step completed successfully.
2. In the **Flow** view, right-click **Design Completion** and select **Run Selected Flow Step** as shown:

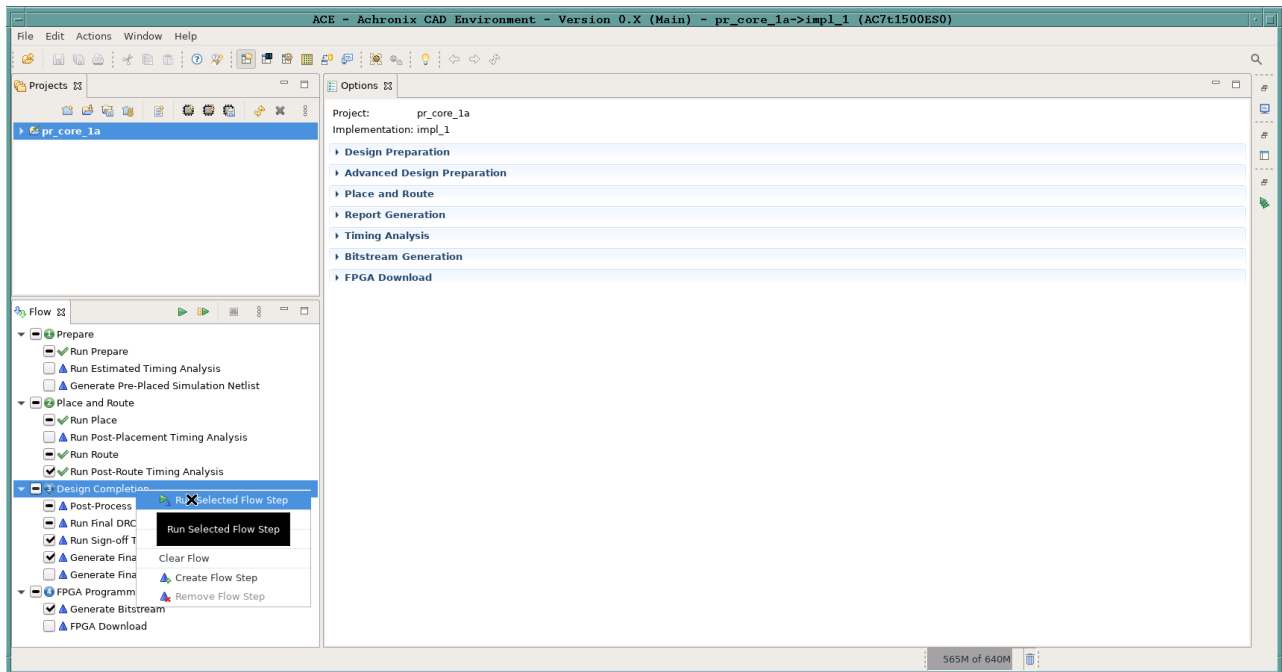



Figure 247 • Run Design Completion Flow Step Example

Save Region/Clock Pre-Routing Constraints Back to PDC File

1. Ensure that the Design Completion flow step completed successfully.
2. In the **Placement Regions** view, click the  **Save Placement Regions** button.

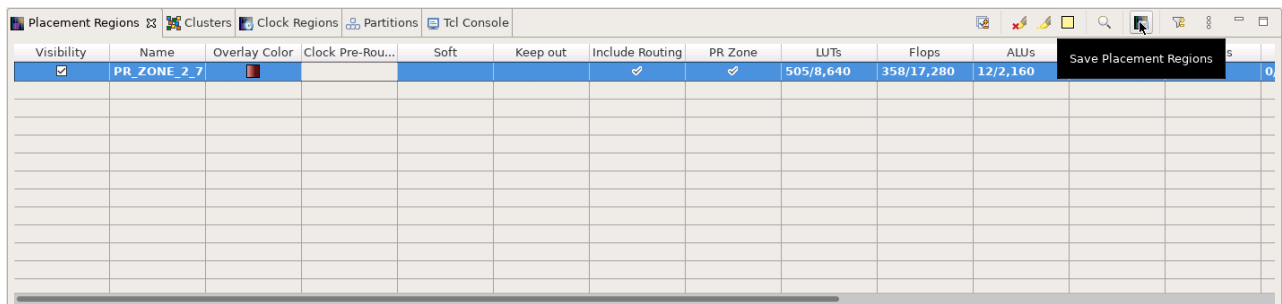


Figure 248 • Save Placement Regions Example

3. The **Save Placement Regions** dialog appears.
4. Click **Finish** to save the PR zone region constraints.
5. The Tcl console displays the following and the `placement_regions.pdc` file is created.

```
create_region "PR_ZONE_2_7" {32 41 87 92} -snap fabric_clusters -type inclusive
-include_routing -pr_zone
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

6. ACE automatically generates a clock pre-route file to the output area under `<ace_output_dir>/ace/impl_1/output/partial_reconfig_core_1_top_clock_preroutes.pdc`.

Note

This file should not be added to the ACE project as it is regenerated during each run. Make a copy of this file before adding it to the project.

Alternately, use the `save_clock_preroute` Tcl command to generate a `.pdc` file with the clock pre-route constraints.

Ensure ACE Generated Blackbox File and Exported Partition for PR Core 1A

ACE generates a blackbox file for the exported partition (PR core 1A). Ensure that this file exists in the following directory:

```
<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/
ace/impl_1/output/blackboxes/pr_core_1a_bb.v
```

ACE also exports a partition database file for the exported partition (PR core 1a). Ensure this file exists in the following directory:

```
<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/
ace/impl_1/output/partitions/pr_core_1a_top.i_pr_core_1a.epdb
```

Export Partition for PR Core 2A

Repeat Procedure for PR Core 2A

1. Repeat the steps in section 1 to export a partition for PR core 2A.
PR core 2A is placed on a different cluster (row=7, col=6) than PR core 1A. Steps 1c, 1d and 1g are skipped by directly adding the following file to the project:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/
pr_core_2a/pr_core_2a_ace_placements.pdc
```

This file contains all the correct region and clock pre-route constraints.

Importing PR Core 1A and 2A Partitions and Generating Base Bitstream for Top-Level Design

Create I/O Ring Configuration for Target Board

In this design, a simple clock input is driving a PLL to create clocks for the core fabric. Also, a simple GPIO interface to display LEDs is in the top-level design.

1. Navigate to the **IP Libraries** view.
2. Right-click the **Clock I/O Bank** IP and select **New IP Configuration**:

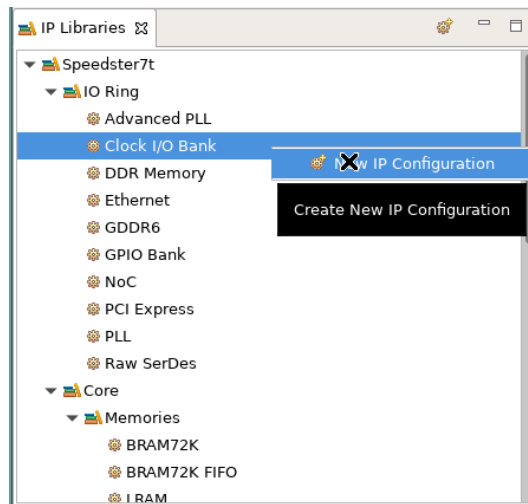


Figure 249 • New IP Configuration Example

3. Configure the Clock I/O Bank for the PR cores using the options shown in the following image:

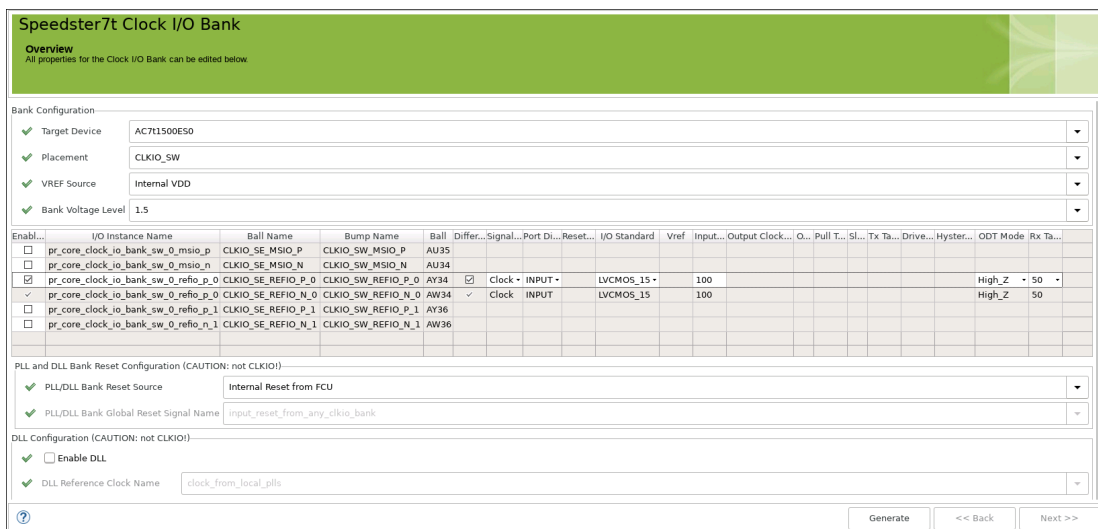


Figure 250 • Clock I/O Bank Configuration Example

4. In the **IP Libraries** view, right-click the **PLL** IP and select **New IP Configuration**.
5. Configure the PLL for the PR core using the options shown in the following image:

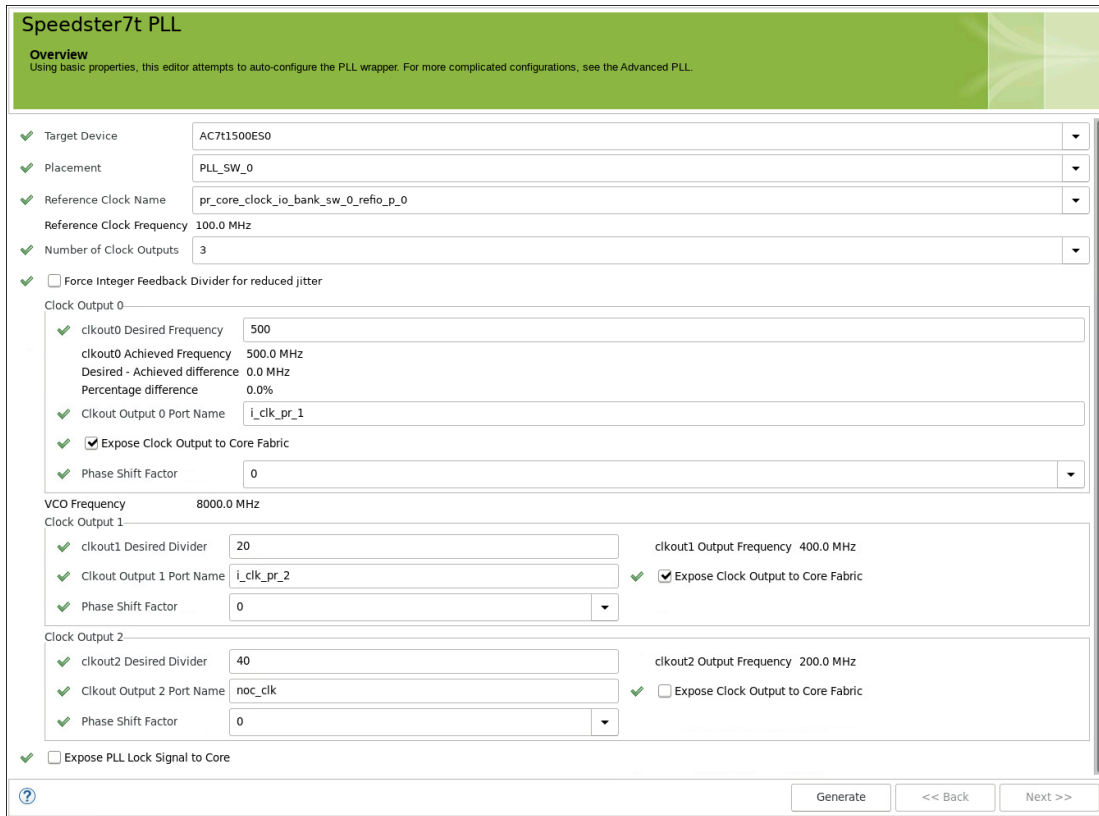


Figure 251 • PLL IP Configuration Example

6. In the **IP Libraries** view, right-click the **NoC** IP and select **New IP Configuration**.
7. Configure the 2D NoC IP for the PR core using the options shown in the following image:

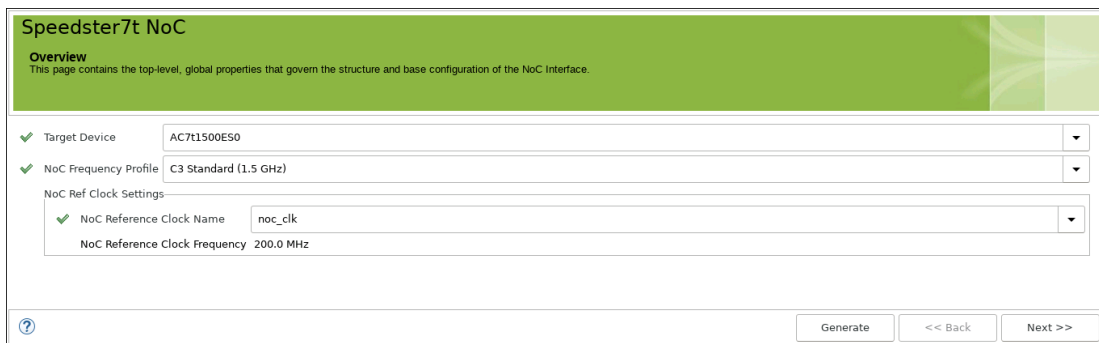


Figure 252 • 2D NoC IP Configuration Example

8. In the **IP Libraries** view, right-click the **Clock I/O Bank** IP and select **New IP Configuration**.
9. Configure the Clock I/O Bank for the top-level design using the options shown in the following image:

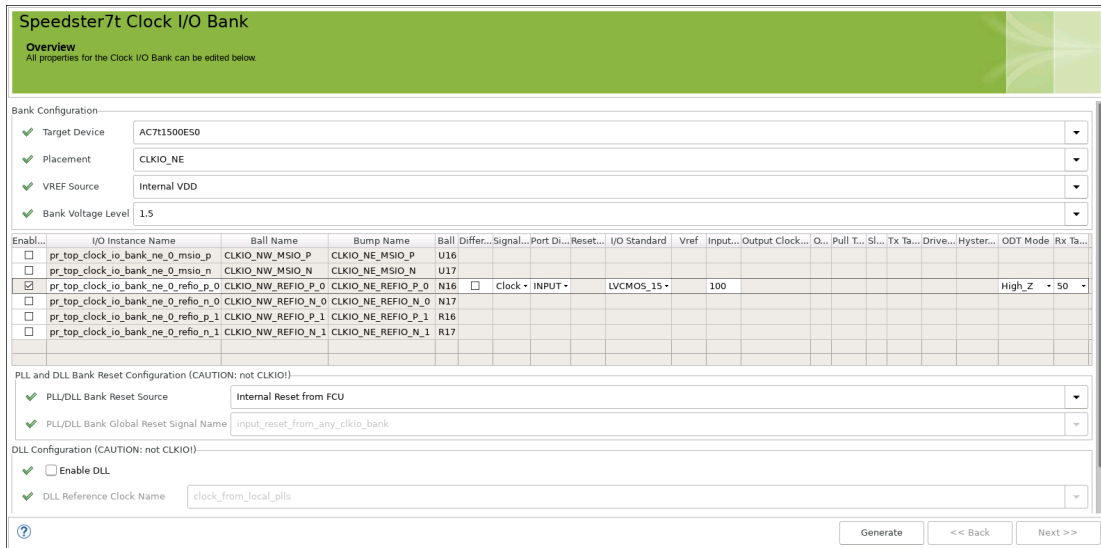


Figure 253 • Clock I/O Bank IP Configuration Example

10. In the **IP Libraries** view, right-click the **PLL** IP and select **New IP Configuration**.
11. Configure the PLL for the top-level design using the options shown in the following image:

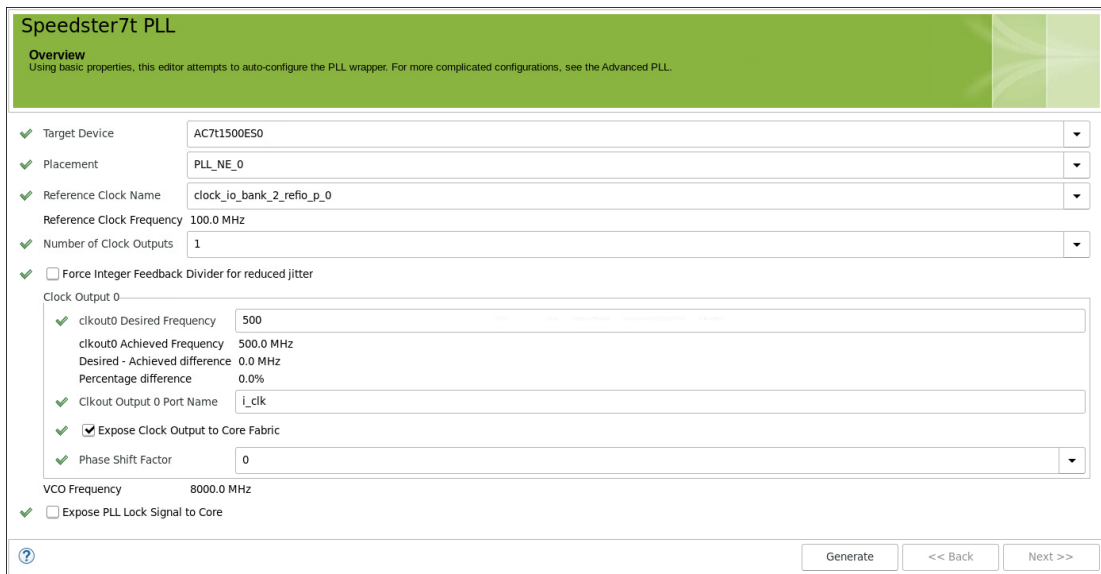


Figure 254 • PLL IP Configuration Example

12. In the **IP Libraries** view, right-click the **GPIO Bank** IP and select **New IP Configuration**.

13. Configure the GPIO Bank for the top-level design using the options shown in the following image:

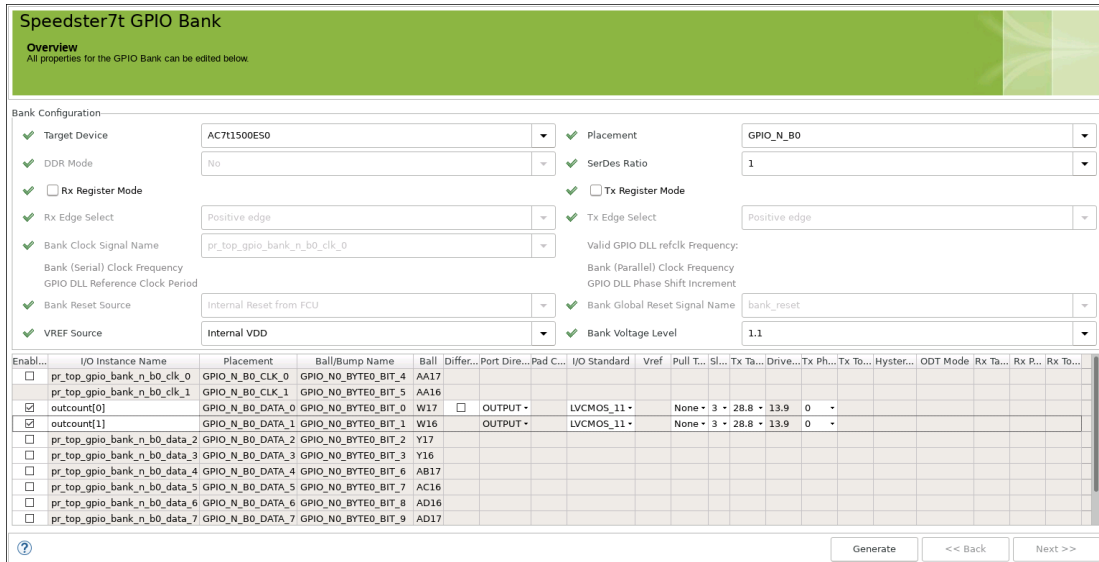


Figure 255 • GPIO Bank IP Configuration Example

14. Click **Generate** to display the **Generate IO Ring Design Files** dialog and select **Add to active project**.

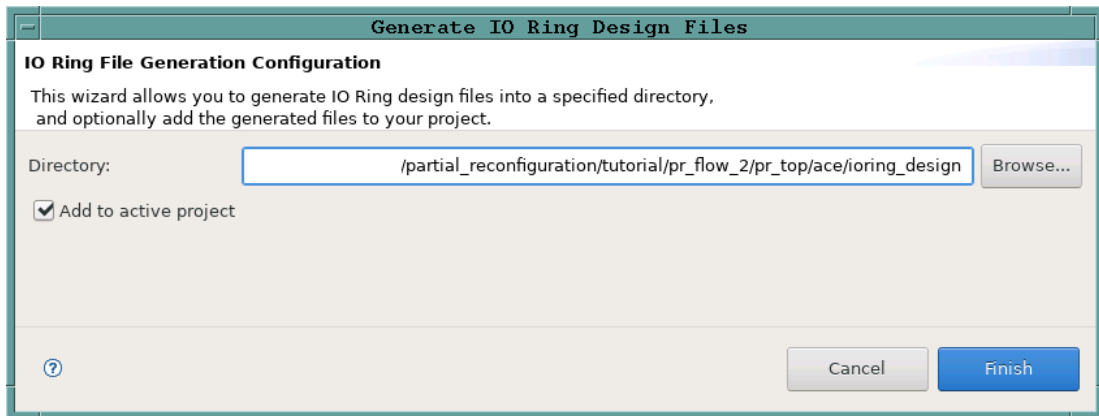


Figure 256 • Generate IO Ring Design Files Dialog Example

15. Click **Finish** to generate the I/O ring design files.

16. When complete, the following files should be added to the ACE project:

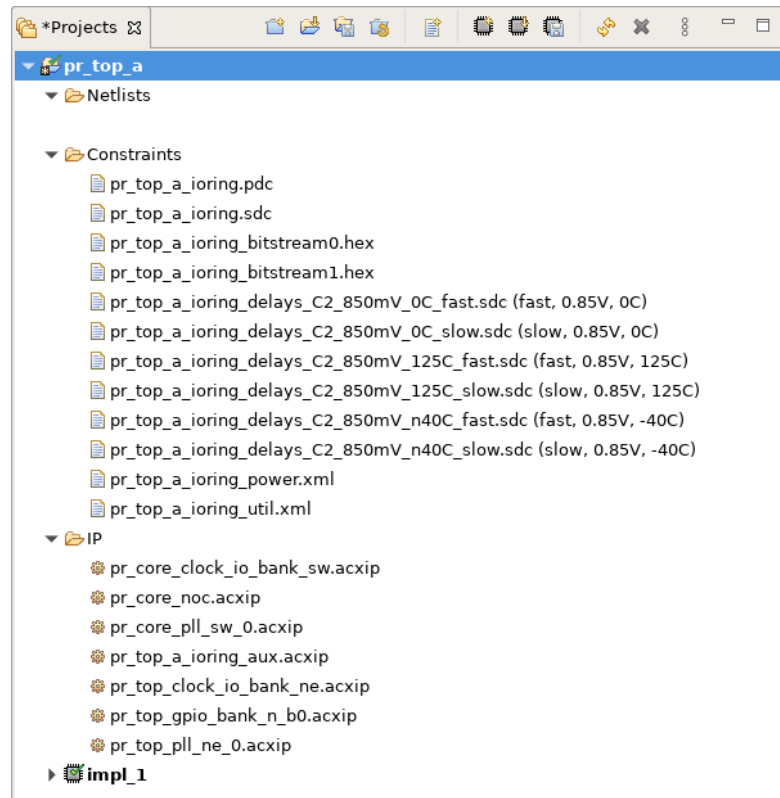


Figure 257 • I/O Ring Design Files

Synthesize Top-Level Design Using Synplify

1. Create a new Synplify project and add all RTL files from the <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/src/rtl directory.
2. Add the pr_top_a_timing.sdc file from the <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top/src/constraints directory.
3. Add the PR core 1A and PR core 2A blackboxes generated by ACE in the previous 2 steps as follows:
 - PR core 1A blackbox - <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output/blackboxes/pr_core_1a_b.v
 - PR core 2A blackbox - <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/ace/impl_1/output/blackboxes/pr_core_2a_bb.v
4. From the main menu, select **Project** → **Implementation Options** → **Device**.
5. Set **Technology** to **Achronix Speedster7t**.
6. Set **Part** to **AC7t1500ES0**.
7. Set **Package** to **F53**.

8. Set **Speed** to **C2**.
9. From the main menu, select **Project** → **Implementation Options** → **Implementation Results**.
10. Set **Result Base Name** to "pr_top_a".
11. Select **Project** → **Implementation Options** → **Verilog**.
12. Set **Top Level Module** to "pr_top_a".
13. Set **Include Path Order** to "<ace_install_dir>/libraries".
14. Add the AC7t1500ES0_synplify .sv file to the project from the <ace_install_dir>/libraries/device_models directory.
15. Verify that the Synplify project matches the following configuration:

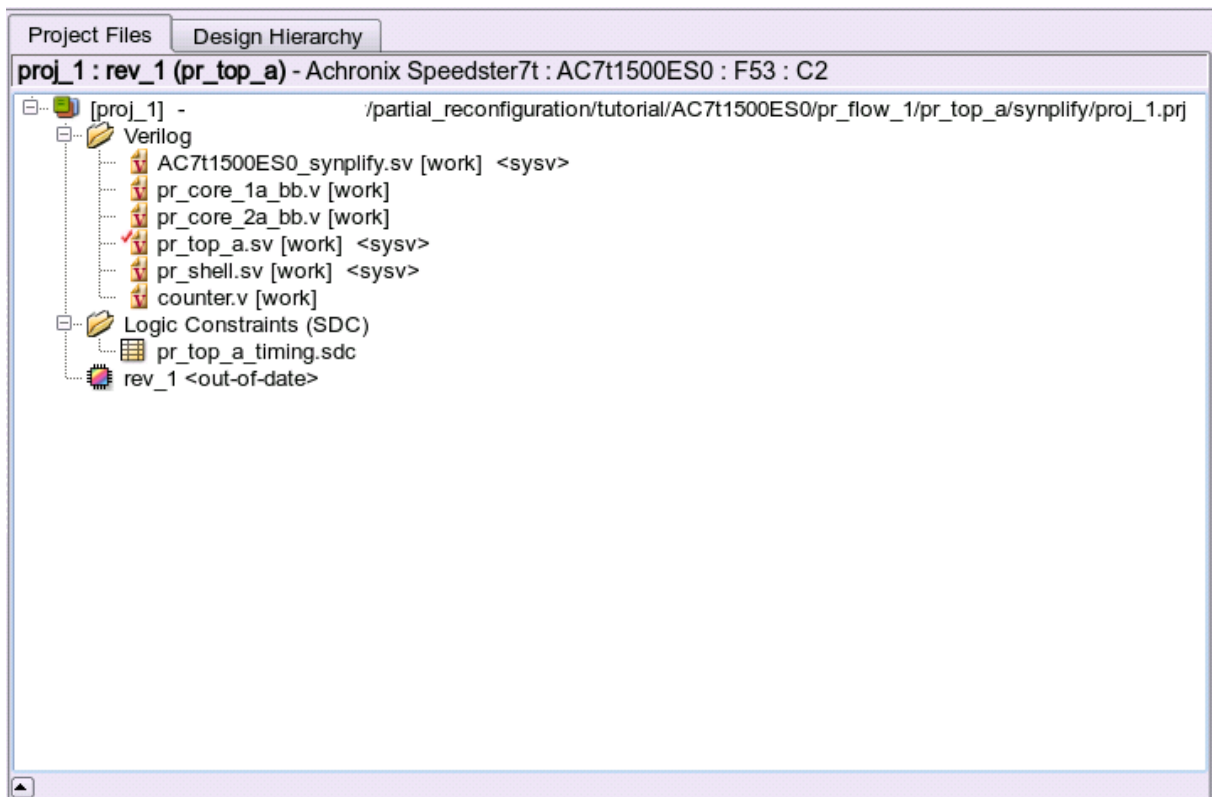



Figure 258 • Synplify Project Example

16. Run and compile the design in Synplify.
17. When successfully completed, Synplify generates a gate-level netlist as <synplify_out_dir>/rev_1/pr_top_a .vm.

Run Top-Level Design Through Run Prepare in ACE

1. Navigate to the Projects view and click the (📁) **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.

3. In the Projects view, click the () **Add Source Files to a Project** button.
4. Add the following files to your project:
 - Blackbox file generated by ACE in step 1j –
`<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1a/ace/impl_1/output/blackboxes/pr_core_1a_bb.v`
 - Blackbox file generated by ACE in step 2j –
`<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2a/ace/impl_1/output/blackboxes/pr_core_2a_bb.v`
 - Synthesized netlist created by Synplify in step 3a –
`<ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/synplify/rev_1/pr_top_a.v`
5. Add all ioring files created in step 3a to the project.
6. Ensure that the blackbox files `pr_core_1a_bb.v` and `pr_core_2a_bb.v` are both above the `pr_top_a.v` file (click and drag the files to change the order).
7. In the **Options** view, navigate to the **Design Preparation** section and ensure that the **Export All Partitions** option is cleared and all other options are correctly set.
8. In the **Flow** view, right-click **Run Prepare** and select **Run Selected Flow Step**.
9. Ensure that the `run_prepare` flow step completed successfully.

Set Up Clock Pre-Routing Constraints

1. In the **Partitions** view, right-click the cell below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes** as shown in the following image:

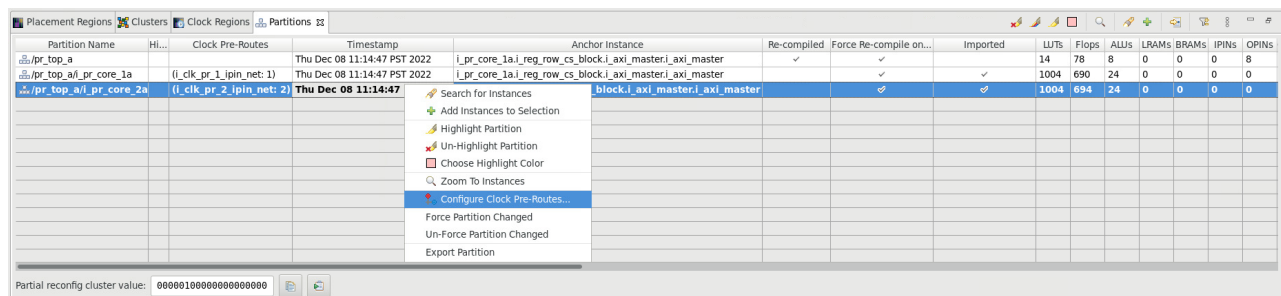


Figure 259 • Configure Clock Pre-routes Example

2. In the **Configure Clock Pre-Routes** dialog, ensure that the clock net is pre-routed into the same track specified in step 1d:

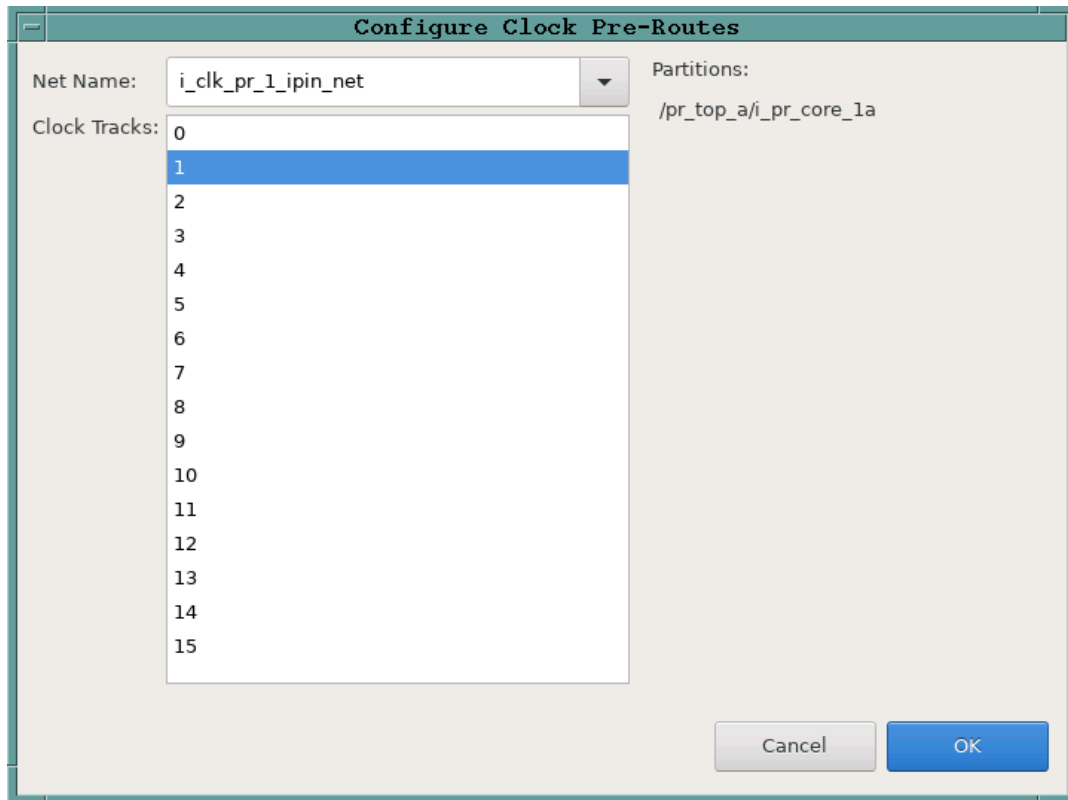



Figure 260 • Configure Clock Pre-Routes Dialog Example

3. Click **OK**. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net 2 -partitions {/pr_top_a/i_pr_core_1a}
```

Set Up Region Constraints (Optional)

1. In the Floorplanner view, navigate to the top section and click the () **Placement Region Tool** button.
2. Click and drag in the Floorplanner to draw a region bounding box for the top-level logic:

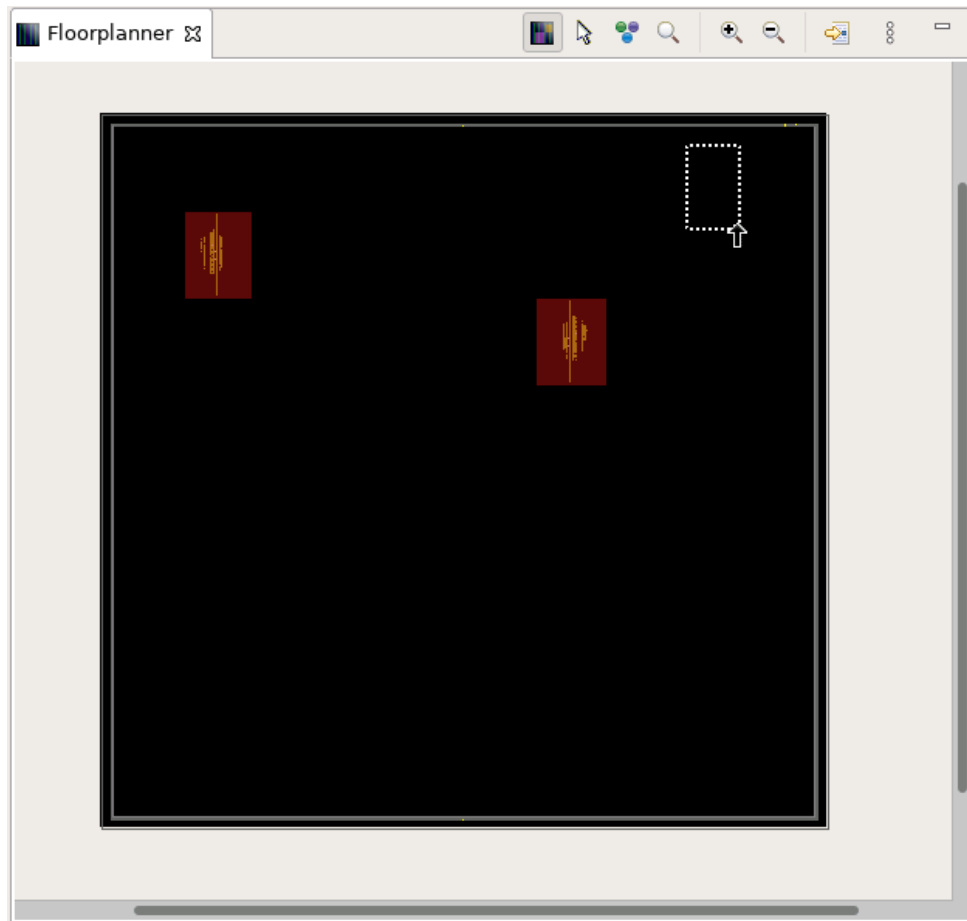


Figure 261 • Floorplanner Region Bounding Box Example

Note

ACE automatically creates a PR Zone for the imported partition.

3. In the **Create Placement Region** dialog, set **Region Alignment** to **Snap to Fabric Clusters**.
4. Set **Region Type** to **Inclusive**.
5. Clear the **Is Partial Reconfiguration Zone** checkbox.

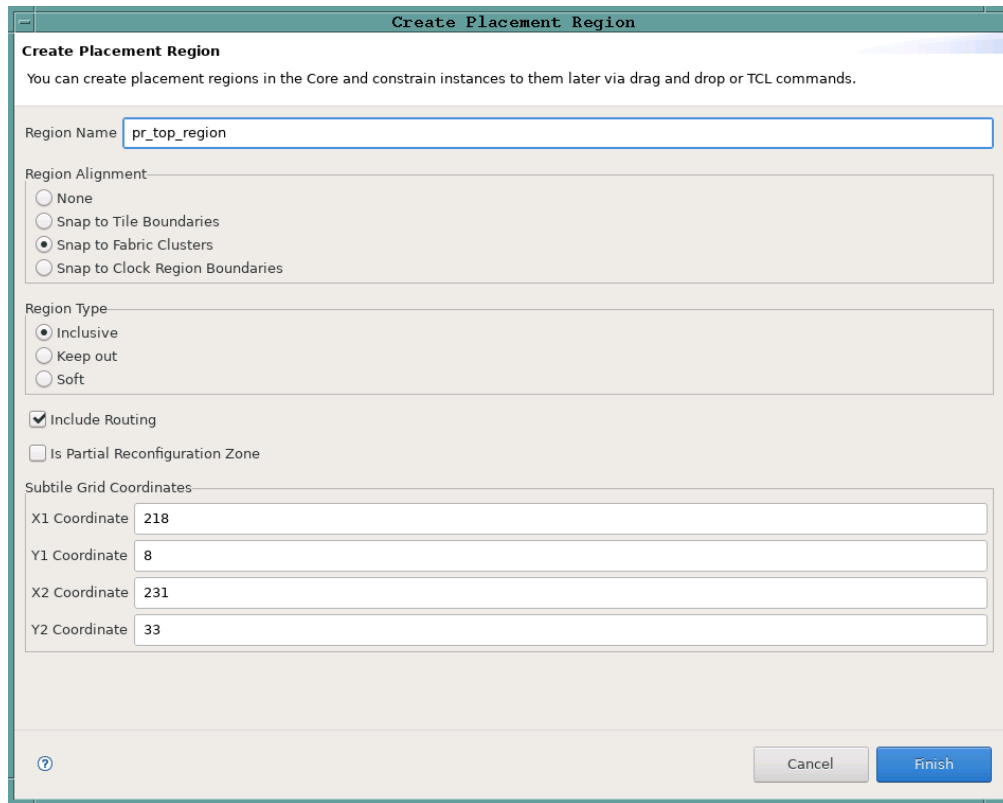


Figure 262 • Create Placement Region Dialog Example

- Click **Finish**. The Tcl console displays the following:

```
create_region "pr_top_region" {218 8 231 33} -snap fabric_clusters -type inclusive
-include_routing
```

- As before, add all instances in the PR core to the region using the `add_region_find_insts` Tcl command nested with the `find` command. `i_pr_shell` is the instance name for the top-level partial reconfigurable module:

```
add_region_find_insts "pr_top_region" {find {*i_pr_shell*} -insts}
```

Run Top-Level Design Through Place-and-Route in ACE

- In the **Flow** view, right-click **Place and Route** and select **Run Selected Flow Step**:

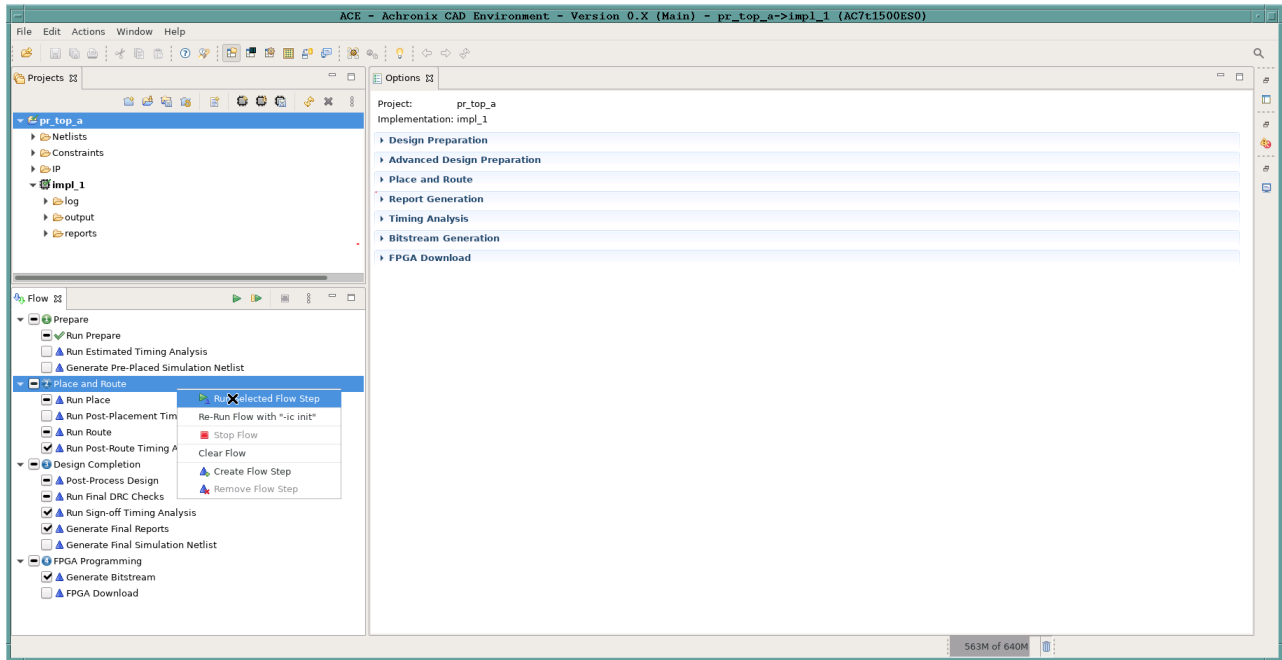


Figure 263 • Place and Route Flow Step Example

Run Final Sign-Off Timing Analysis in ACE

1. In the **Flow** view, right click the **Design Completion** step and select **Run Selected Flow Step**:

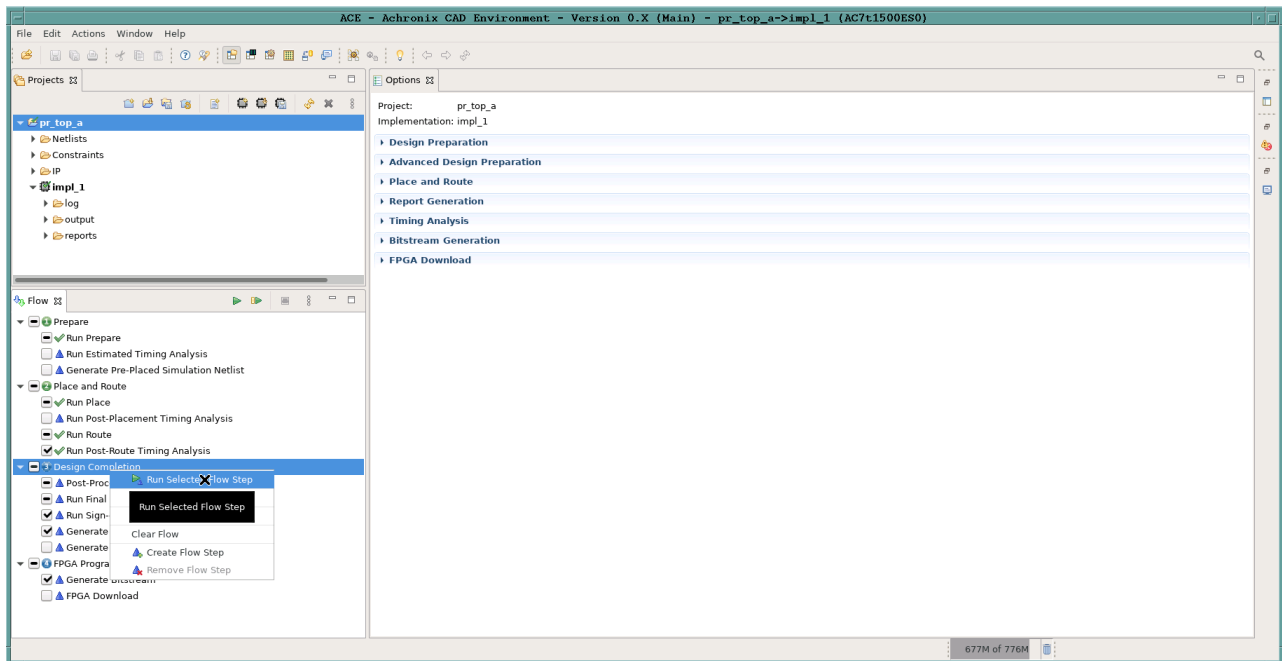


Figure 264 • Run Design Completion Flow Step Example

Generate Base Bitstream for Top-level Design

Note

The top-level base bitstream contains the top-level design plus the two imported PR cores.

1. In the "Options" view, click the **Bitstream Generation** tab.
2. In the **FCU Configuration** section, check the **Lock FCU After Programming** checkbox:

FCU Configuration	
4-bit Speedcore Instance ID (hex)	0
Memory Scrubbing Mode	Background Scan and Repair
CRC Checking Mode	Fully Enabled
<input checked="" type="checkbox"/> Lock FCU After Programming	

Figure 265 • Bitstream Generation FCU Configuration Example

3. In the Flow view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**:

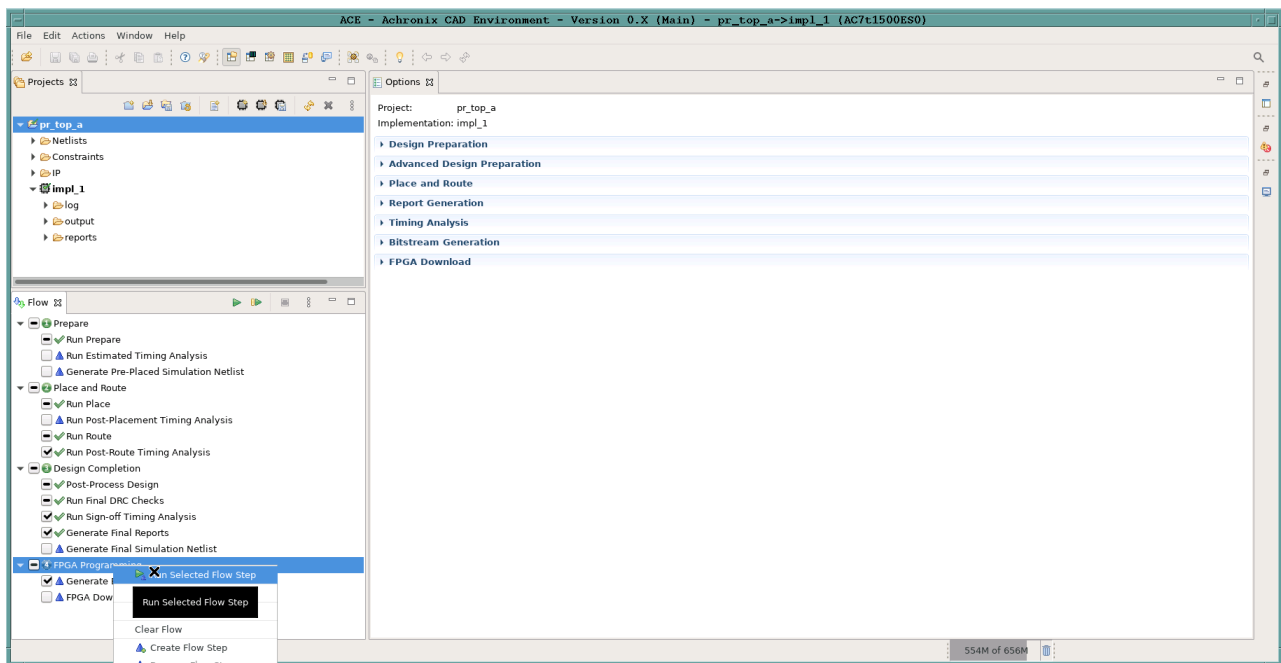


Figure 266 • Run FPGA Programming Flow Step Example

Save Region/Clock Pre-Routing and Partition Placement Constraints Back to PDC File

1. Repeat step 1g to save the placement region and clock pre-routing constraints to a pdc file.
2. In the **Partitions** view, click the  **Save Partition Pre-Placement** button:

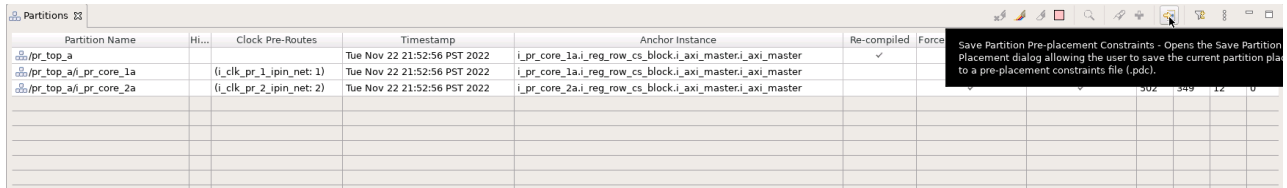


Figure 267 · Save Partition Pre-Placement Example

3. In the **Save Partition Pre-Placement Data** dialog, ensure that the **Save Specific List of Partitions** checkbox is cleared in order to save the partition placement constraints for each partition.

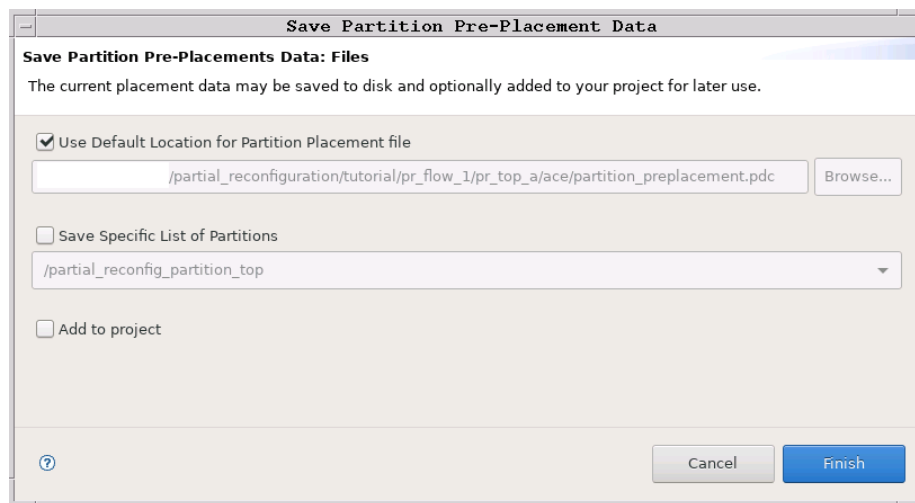


Figure 268 · Save Partition Pre-Placement Data Dialog Example

Generate Partial Bitstream for PR Core 1B and 2B

Export PR Core 1B as Partition

1. Repeat all steps in section 1 to export PR core 1B as a partition.
PR core 1B is identical to PR core 1A except that the 32-bit adder increments by 2 instead of 1 at each NAP read. The source files for PR core 1B are located in the following directory:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b/src/...
```

Export PR Core 2B as Partition

1. Repeat all steps in section 1 to export PR core 2B as a partition.

PR core 2B is identical to PR core 2A except that the 32-bit subtractor decrements by 2 instead of 1 at each NAP read.

The source files for PR core 2B are located in the following directory:

```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/  
pr_core_2b/src/...
```

Import PR Core 1B and 2B Into Top-Level Design


1. Create an ACE Project named "pr_top_b".
2. In the **Options** view, set the same options used in step 3b.
3. In the **Projects** view, click the () **Add Source Files to Project** button.
4. Add the following files to the project:
 - All ioring config files generated in step 3a (the I/O in pr_top_b is the same as pr_top_a so the files can be reused)
 - <ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_1b/ace/impl_1/output/blackboxes/pr_core_1b_bb.v (blackbox file generated by ACE in step 4a).
 - <ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_core_2b/ace/impl_1/output/blackboxes/pr_core_2b_bb.v (blackbox file generated by ACE in step 4b).
 - <ace_install_dir>/examples/partial_reconfig/tutorial/AC7t1500ES0/pr_flow_1/pr_top_b/synplify/rev_1/pr_top_b.vm (synthesized netlist created by Synplify in step 3a)
5. Ensure that the blackbox files pr_core_1b_bb.v and pr_core_2b_bb.v are both above the pr_top_b.vm file (drag and drop files to change the order):



Figure 269 - PR Core Source Files Example

6. Run the design through "Place-and-Route" (remember to configure the clock pre-routes by repeating the step 3d).
7. Ensure that the clock nets are pre-routed on the same clock track.

Generate Partial Bitstream for PR Core 1B and 2B in Top-level Design

1. Ensure that the Place-and-Route flow step completed successfully.
2. Set the bitstream cluster mask and partial reconfiguration option for PR core 1B and 2B (set the mask for both partitions by selecting both in the **Partitions** view):

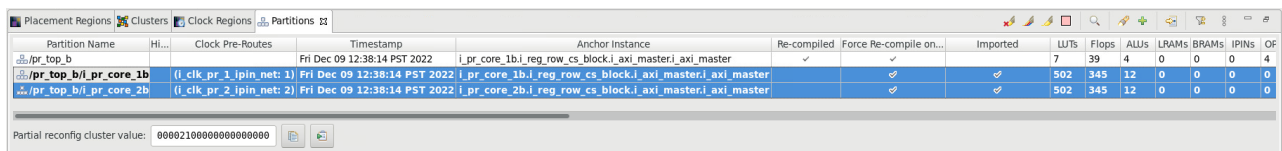



Figure 270 - Selecting PR Core 1B and 2B Example

3. Click the  **Send Tcl Command** button to set this implementation option. The Tcl Console displays the following:

```
set_impl_option bitstream_prc_cluster_map 00002100000000000000
```

4. In the **Options** view, click the **Bitstream Generation** tab.
5. In the **Partial Reconfiguration** section, Check the **Enable Partial Reconfiguration** checkbox.
6. Ensure that the **Partial Reconfig Cluster Map (hex)** is set to the same value as in the **Placement Regions** view:

Partial Reconfiguration

Enable Partial Reconfiguration

Partial Reconfig Cluster Map (hex) 00002100000000000000

Figure 271 - Partial Reconfiguration Settings Example

7. In the **Options** view, click the **Bitstream Generation** tab.
8. In the **Partial Reconfiguration** section, check the **Lock FCU After Programming** checkbox:

FCU Configuration

4-bit Speedcore Instance ID (hex) 0

Memory Scrubbing Mode Background Scan and Repair

CRC Checking Mode Fully Enabled

Lock FCU After Programming

Figure 272 - Partial Reconfiguration Settings Example

9. Run the **FPGA Programming** flow step to generate a single partial bitstream for PR core 1B and 2B.

Bitstream Programming Sequence

Apply Power to Board

1. Apply power to the board and ensure cables are properly connected.

Program Top-Level Base Bitstream Containing PR Core 1A and 2A

1. Run the following commands.

Remember that the `pr_top_a` base bitstream contains PR core 1A, 2A and the top-level logic.

```
set jtag_id [jtag::get_connected_devices]
jtag::open $jtag_id
jtag::configure_scan_chain $jtag_id AC7t1500ES0 0 0 0 -single_device
jtag::ac7t1500_initialize_fcu $jtag_id -reset
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_a/ace/impl_1/output/
pr_top_a.hex
```

Run PR Core 1A and 2A Test Scripts

1. Run the following commands.

The scripts contain NAP reads from PR core 1A and 2A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_1/test_scripts/read_pr_core_1a.tcl  
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_1/test_scripts/read_pr_core_2a.tcl
```

Program Partial Bitstream for PR Core 1B and 2B

1. Run the following command.

PR core 1B and 2B can be partially reconfigured onto the board at the same time since a single partial bitstream was generated for both PR core 1B and 2B in step 4d.

```
#Program partial bitstream  
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/  
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_1/pr_top_b/ace/impl_1/output/  
pr_top_b.hex
```

Run PR Core 1B and 2B Test Scripts

1. Run the following commands.

The scripts contain NAP reads from PR core 1B and 2B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_1/test_scripts/read_pr_core_1b.tcl  
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_1/test_scripts/read_pr_core_2b.tcl
```

PR Flow 2: Multi-Project PR Flow Using Keep Out Regions

Introduction

The Multi-project PR flow using keepout regions is an alternate method to create designs that can be partially reconfigured.

This flow involves using separate ACE projects to generate separate base and partial bitstreams for the top-level design and each PR core. On silicon, the base bitstream is programmed first. Partial bitstreams can be programmed subsequently. Using this flow, the user design is stitched together at a bitstream level. There is no integration in ACE between the the top-level design and each PR core since the designs are stitched at the bitstream level. As such, this flow has a few pitfalls and can only be recommended for advanced users.

Advantages of Using This PR Flow

- Very fast turn-around time in that the top-level design bypasses place-and-route of the PR cores requiring only a keep-out region
- Enables integrating third party accelerator cores at the bitstream level while not requiring any knowledge of the logic
- Instead of simulating the entire design together, partial bitstreams can be run/tested on the Achronix Virtual Lab which has a faster turn-around time than simulation

Possible Issues Encountered When Using This PR Flow

- The bitstream might not be stitched together correctly at the bitstream level if any of the ACE projects are configured incorrectly as in the following scenarios:
 - a. In the base bitstream, clock nets to be used in the partial bitstream are not pre-routed. The clock signal is not correctly connected to the PR cores after partial reconfiguration.
 - b. Clock nets are pre-routed on different clock tracks in the partial and base bitstreams. The clock signals are incorrectly connected to the PR cores after partial reconfiguration.
 - c. In the base bitstream, keep-out regions are not set on clusters to be used by PR cores. The top-level logic might be lost after partial reconfiguration.
- The design might not meet timing on paths between the top-level and PR core after the partial bitstream is programmed since timing analysis is done separately in each ACE project.

Note

The timing delays on the east side are different than those on the west side of the AC7t1500ES0 device.

- Simulating the entire design within an ACE project is difficult as no integration exists between separate ACE projects. It is not impossible but requires a lot of manual work.

To avoid these issues, consider using the multi-project flow with partition export/import (PR flow 1) instead. In that flow, ACE can integrate separate projects created for both the top-level design and for each PR core by taking advantage of the partition export/import feature. This feature enables ACE to run checks to ensure that the top-level design and each PR core can be stitched together correctly. Timing can be closed and simulation run on the integrated design.

High-level design flow

The bottom-up flow is recommended for PR flow 2. This involves generating the partial bitstreams for each of the PR cores before generating the full bitstream for the top-level design. In the the top-level design, keep-out regions and clock pre-routes on each cluster where the PR cores are partially reconfigured, must be set. As such, it is easier to finish floor planning and generating the partial bitstream for each PR core before working on the top-level design used to create the full bitstream.

Please follow this high-level design flow carefully:

1. [Partial Bitstream Generation for PR Core 1A \(page 522\)](#)
 - a. [Synthesize PR Core 1A Using Synplify \(page 522\)](#)
 - b. [Run PR Core 1A Through Run Prepare in ACE \(page 524\)](#)
 - c. [Set Up Region Constraints for PR Core 1A \(page 525\)](#)

-
- d. [Set Up Clock Pre-Routing Constraints for PR Core 1A \(page 528\)](#)
 - e. [Run PR Core 1A Through Place-and-Route in ACE \(page 530\)](#)
 - f. [Run Final Sign-Off Timing Analysis for PR Core 1A \(page 530\)](#)
 - g. [Set Bitstream Cluster Mask and Partial Reconfiguration Option for PR Core 1A \(page 531\)](#)
 - h. [Generate Partial Bitstream for PR Core 1A \(page 532\)](#)
 - i. [Save Region and Clock Pre-Routing Constraints to PDC File \(page 533\)](#)
2. [Generate Partial Bitstream for PR Cores 1B 2A and 2B \(page 534\)](#)
 - a. [Generate Partial Bitstream for PR Core 1B \(page 534\)](#)
 - b. [Generate Partial Bitstream for PR Core 2A \(page 534\)](#)
 - c. [Generate Partial Bitstream for PR Core 2B \(page 535\)](#)
3. [Generate Full Bitstream for Top-Level Design \(page 535\)](#)
 - a. [Create I/O Ring Configuration for Target Board \(page 535\)](#)
 - b. [I/O Ring Configuration Steps \(page 535\)](#)
 - c. [Synthesize Top-Level Design Using Synplify \(page 541\)](#)
 - d. [Run Top-Level Design Through Run Prepare in ACE \(page 542\)](#)
 - e. [Set Up Keep Out Regions and Placement Region Constraints for Static Logic \(page 543\)](#)
 - f. [Set Up Clock Pre-Routing Constraints \(page 546\)](#)
 - g. [Run Place-and-Route in ACE for Top-Level Design \(page 547\)](#)
 - h. [Run Final Sign-Off Timing Analysis in ACE for Top-Level Design \(page 548\)](#)
 - i. [Generate Base Bitstream for Top-Level Design \(page 549\)](#)
 - j. [Save Placement Region and Clock Pre-Routing Constraints to PDC File \(page 550\)](#)
4. [Bitstream Programming Sequence \(page 550\)](#)
 - a. [Apply Power to Board \(page 550\)](#)
 - b. [Program Top-Level Design Full Bitstream \(page 551\)](#)
 - c. [Program PR Core 1A Partial Bitstream \(page 551\)](#)
 - d. [Run PR Core 1A Test Scripts \(page 551\)](#)
 - e. [Program PR Core 2A Partial Bitstream \(page 551\)](#)
 - f. [Run PR Core 2A Test Scripts \(page 551\)](#)
 - g. [Program PR Core 1B Partial Bitstream \(page 552\)](#)
 - h. [Run PR Core 1B Test Scripts \(page 552\)](#)
 - i. [Program PR Core 2B Partial Bitstream \(page 552\)](#)
 - j. [Run PR Core 2B Test Scripts \(page 552\)](#)

Partial Bitstream Generation for PR Core 1A

Synthesize PR Core 1A Using Synplify

1. Navigate to the following directory:

<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/
pr_flow_2/pr_core_1a/src

 **Note**

<ace_install_dir> is the directory path where ACE was installed.

Below this directory should be the following two directories:

- rtl
 - constraints
2. In the rtl directory, create a new Synplify project and add all of the RTL files.
 3. Add the pr_core_1a_timing.sdc file to the constraints directory.
 4. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
 5. Set **Part** to **AC7t1500ES0**.
 6. Set **Package** to **F53**.
 7. Set **Speed** to **C2**.
 8. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_core_1a_top**.
 9. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_core_1a_top**.
 10. Set **Include Path Order** to **<ace_install_dir>/libraries**.
 11. In the <ace_install_dir>/libraries/device_models directory, add the AC7t1500ES0_synplify.sv file to the project

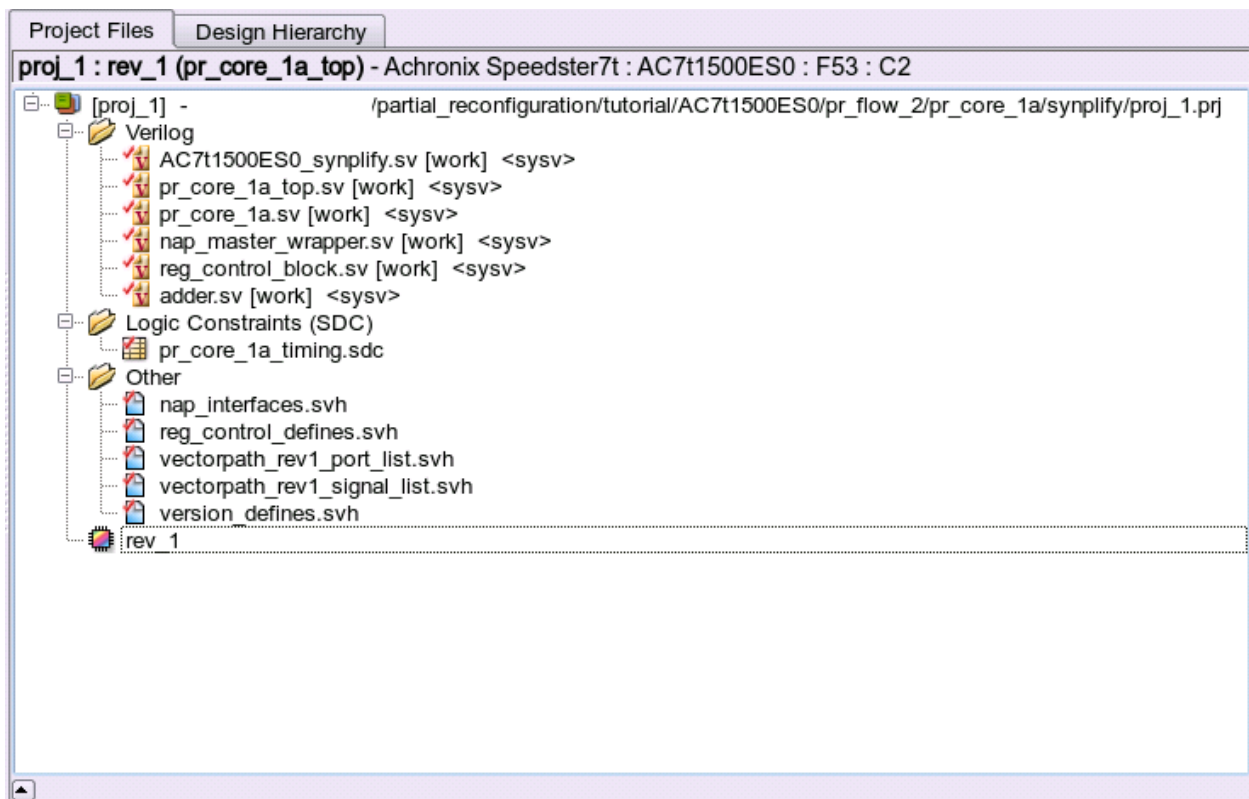




Figure 273 • Project Example

12. Run and compile the design using Synplify.
13. After successful completion, Synplify generates a gate-level netlist under `<synplify_out_dir>/rev_1/pr_core_1a_top.vm`.

Run PR Core 1A Through Run Prepare in ACE

1. In the **Projects** view, click the () **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the **Projects** view, click the () **Add Source Files to a Project** button.
4. Add the following files to your ACE project:
 - `<synplify_out_dir>/rev_1/pr_core_1a_top.vm`. (synthesized netlist created by Synplify in step 1a)
 - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/ioring_design/pr_core_1a_top_ioring.pdc`

Note

This file is required to generate a bitstream because all top-level core fabric pins need to be pre-placed.

For partial bitstream projects, either create a dummy .pdc file for the top-level pins, or, use the .pdc file generated from the top-level design through the I/O Designer.

◦ <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/src/ioring_design/pr_core_1a_top_ioring.sdc

5. In the **Options** view, navigate to the **Design Preparation** tab and make sure all options are correctly set.
6. Copy the option values shown in the following example.
7. In the **Flow** view, right click **Run Prepare** and select **Run Selected Flow Step**.

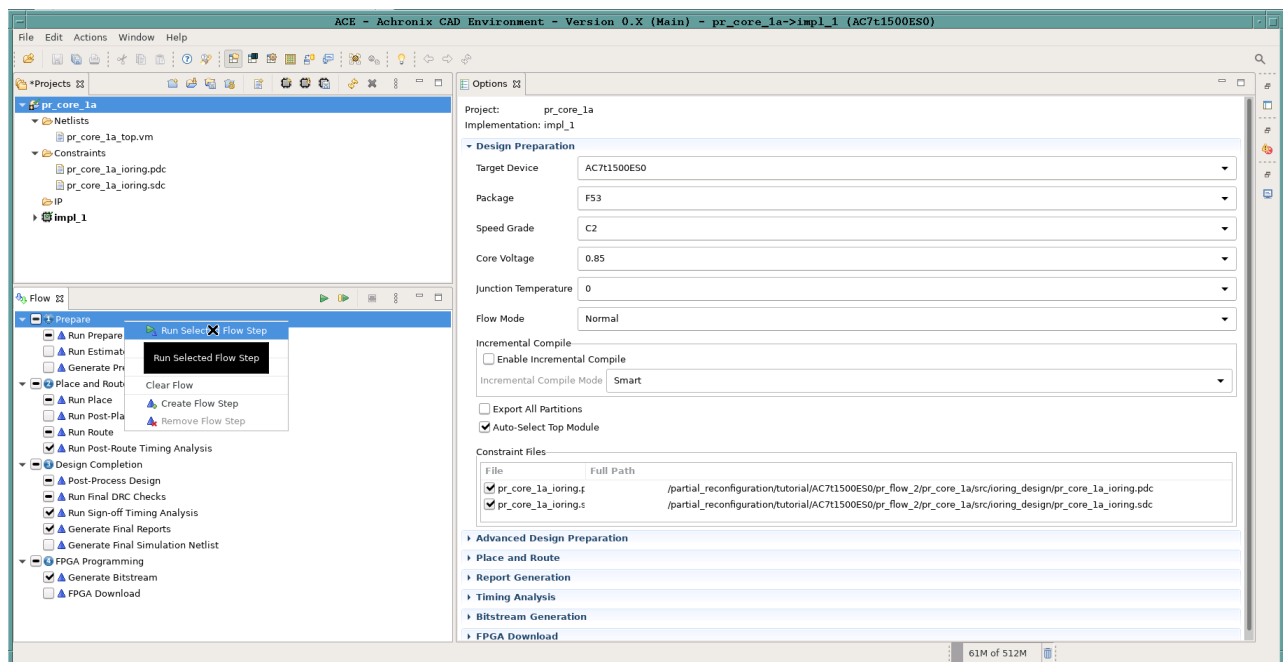



Figure 274 • Run Prepare Flow Step Example

Set Up Region Constraints for PR Core 1A

1. Ensure that the run_prepare flow step has completed successfully.
2. In the **Floorplanner** view, click the () **Placement Region Tool** button.
3. Click and drag on the Floorplanner to draw a PR Zone for the PR Core.

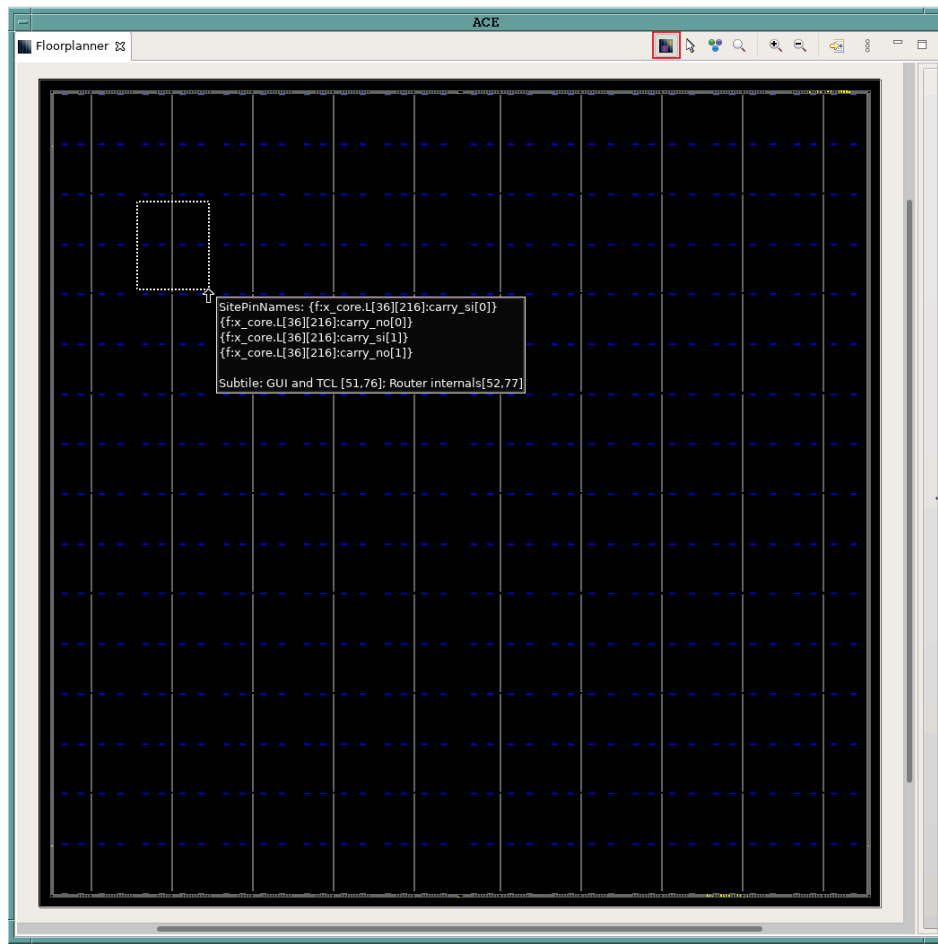


Figure 275 • Creating a Placement Region Zone Example

4. After releasing the left mouse button, the **Create Placement Region** dialogue appears.
5. In **Region Name**, enter a name for the PR zone
6. Set **Region Alignment** to **Snap to Fabric Clusters** (a fabric cluster represents the lowest level of granularity at which the core fabric can be partially reprogrammed).
7. Set **Region Type** to **Inclusive** (ensures that all instances are constrained to the region).
8. Select **Include Routing** (ensures that all routed nets are constrained to the region).
9. Select **Is Partial Reconfiguration Zone** (must be selected to indicate this region is a PR Zone).

Figure 276 • Create Placement Region Dialog Example

10. Click **Finish**. The Tcl console displays the following:

```
create_region "PR_ZONE_2_7" {32 41 48 77} -snap fabric_clusters -type inclusive
-include_routing -pr_zone
```

11. Add all the instances in the PR Core to the region. A quick and recommended way of doing so is to use the `add_region_find_insts` Tcl command nested with the `find` Tcl command. Run the following example in the Tcl console to add all instances in PR Core 1A to the PR zone (`i_pr_core_1a` is the instance name for the top-level partial reconfigurable module).
12. In the GUI, drag the PR core 1A module in the netlist browser view to the `PR_ZONE_2_7` to issue this Tcl command:

```
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

13. In the **Floorplanner** tab, the image should resemble the following (the red box represents the bounding box of the PR zone just created):

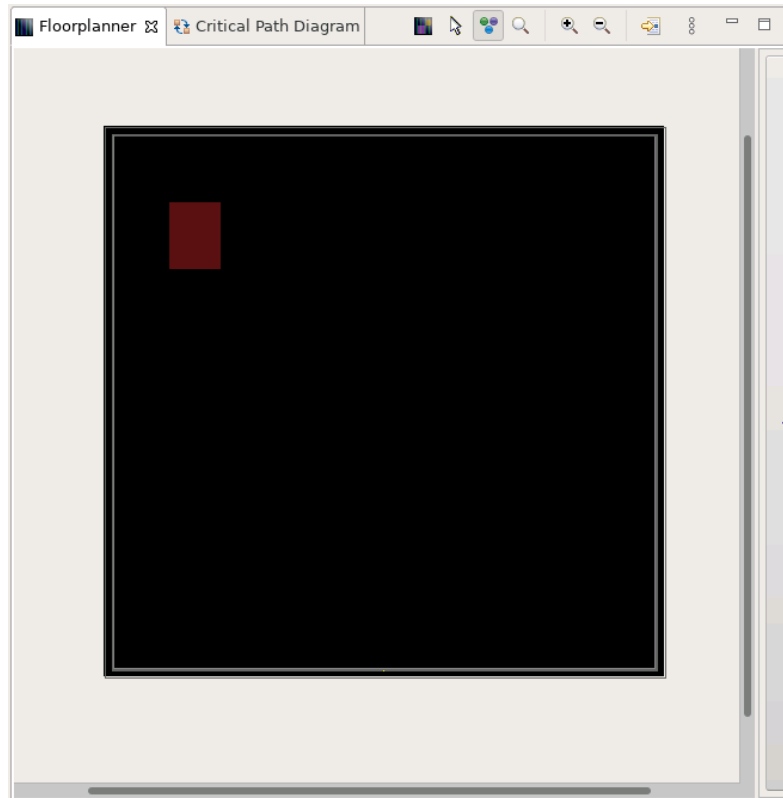


Figure 277 • PR Zone Bounding Box Example

Set Up Clock Pre-Routing Constraints for PR Core 1A

There are several different methods for setting up clock pre-routing. For PR flow 2, the easiest way to configure clock pre-routes is to use the **Placement Regions** view.

1. In the **Placement Regions** view, right-click the box below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes** as shown:

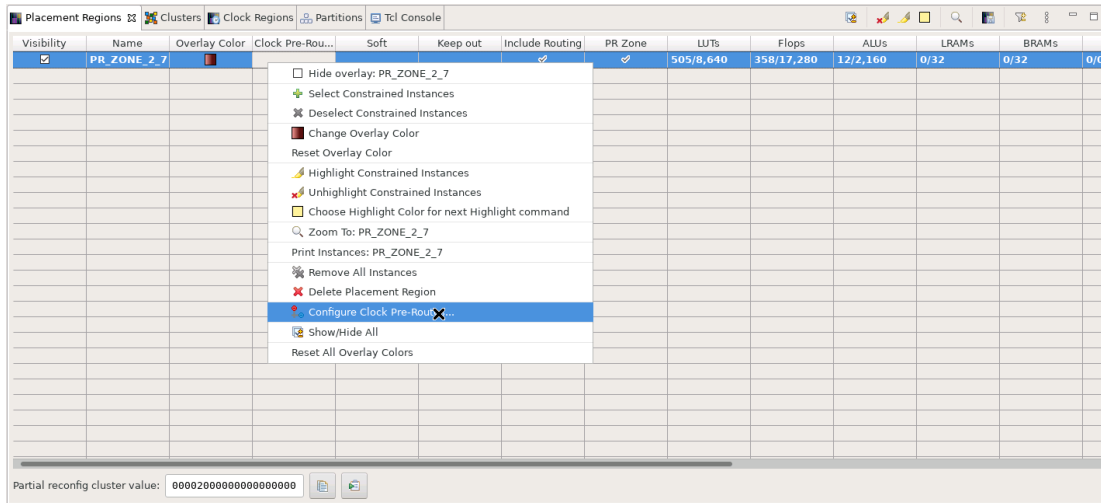


Figure 278 • Configure Clock Pre-Routes Example

2. The **Configure Clock Pre-Routes** dialog appears with the `i_clk_pr_1_ipin_net` signal that drives PR core 1 clock pins pre-routed on clock track 1.

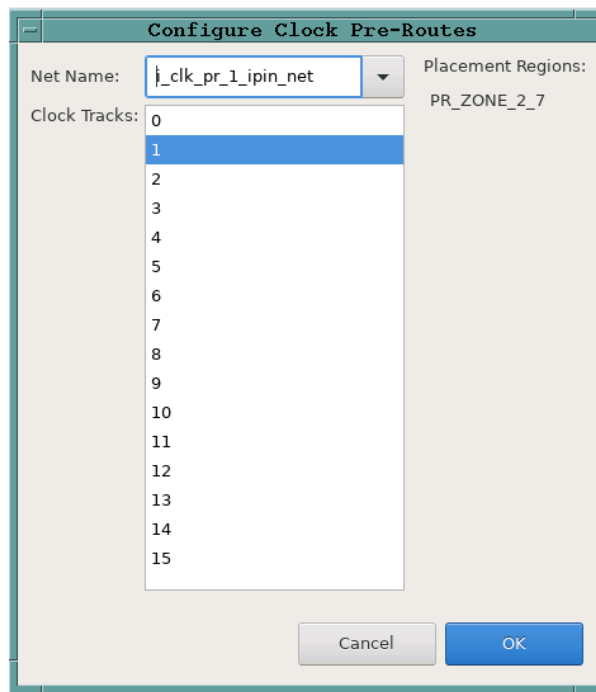


Figure 279 • Configure Clock Pre-Routes Dialog Example

3. Click **OK**. The Tcl console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -placement_regions { PR_ZONE_2_7 }
```

Note

Each clock net used in the top-level design must be routed on a different clock track.

Run PR Core 1A Through Place-and-Route in ACE

1. In the **Flow** view, right-click the **Place and Route** step and select **Run Selected Flow Step**.

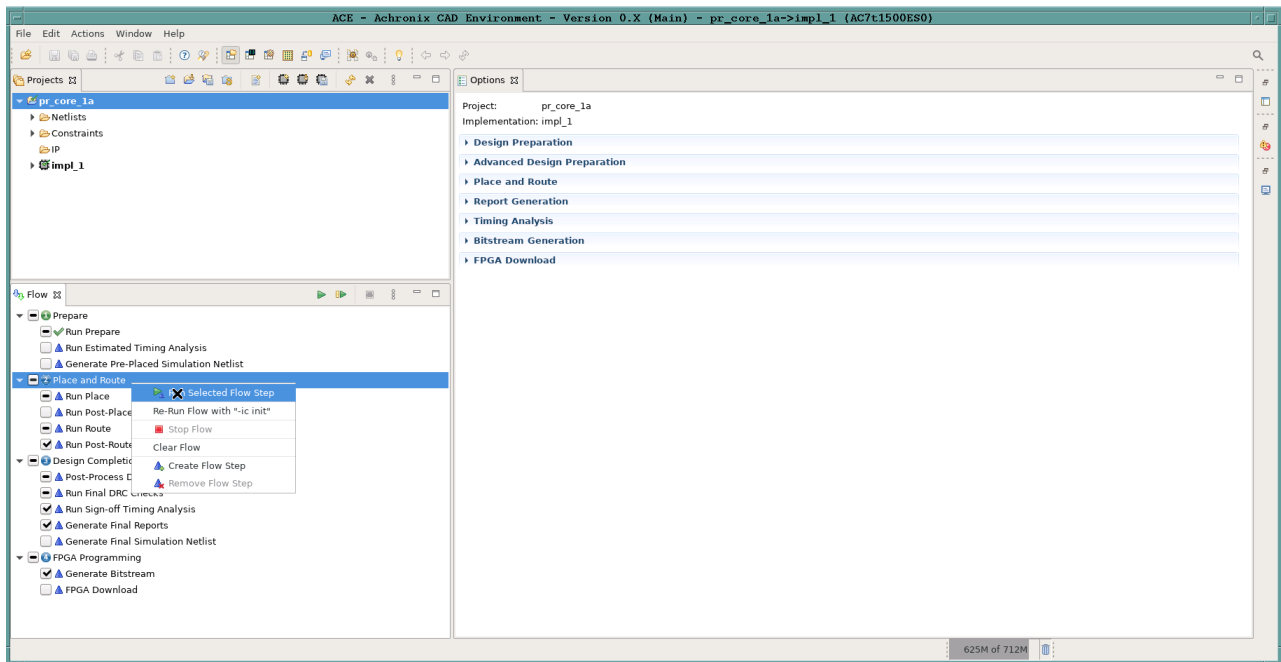


Figure 280 • Run Place and Route Flow Step Example

Run Final Sign-Off Timing Analysis for PR Core 1A

1. In the **Flow** view, right-click **Design Completion** and select **Run Selected Flow Step**.

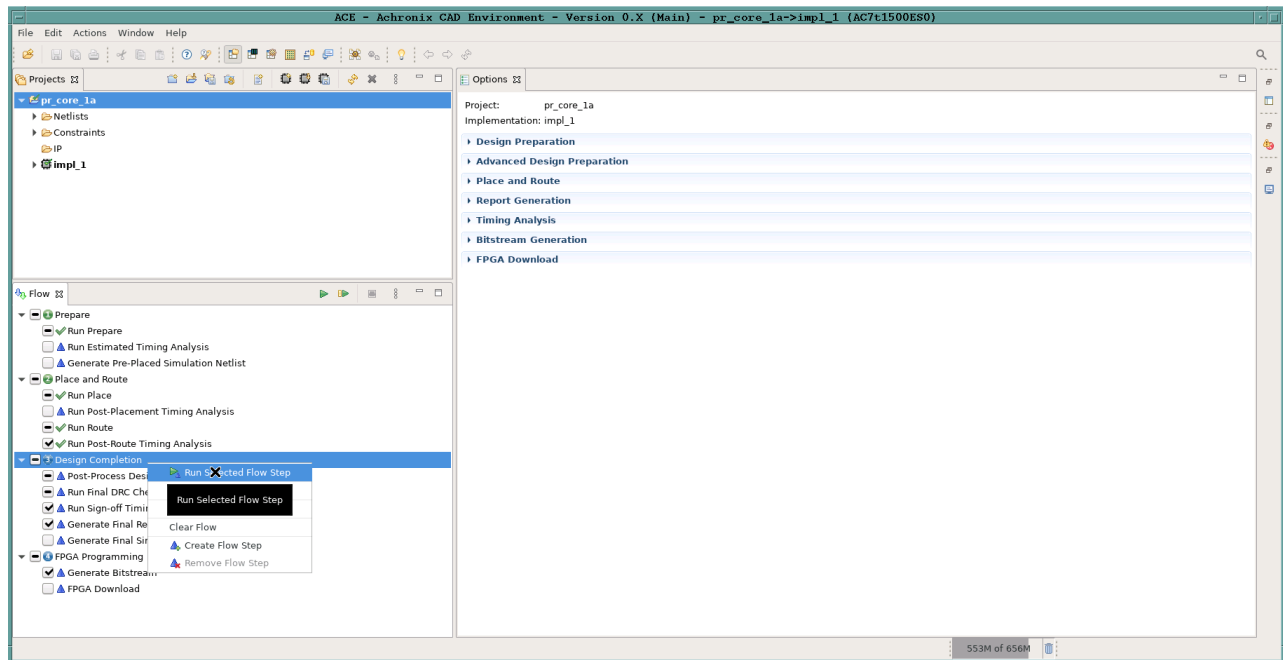


Figure 281 • Run Design Completion Flow Step Example

Set Bitstream Cluster Mask and Partial Reconfiguration Option for PR Core 1A

1. Ensure that the **Design Completion** flow step has run successfully.
2. In the **Placement Regions** view, check the **Visibility** checkbox for the PR_ZONE_2_7 region to be partially reconfigured (the partial reconfig cluster value is computed based on the placement regions having the **Visibility** item checked).

At the bottom of the **Placement Regions** view, in the **Partial reconfig cluster value** section, the bitstream cluster mask value appears for the PR zone region created in step 1c. This value indicates which clusters are occupied by the PR zone. When the `bitstream_prc_cluster_map` implementation option is set to this value, a partial bitstream is generated for the clusters within the PR Zone.

3. Click the (📄) **Send Tcl Command** button to set this implementation option.

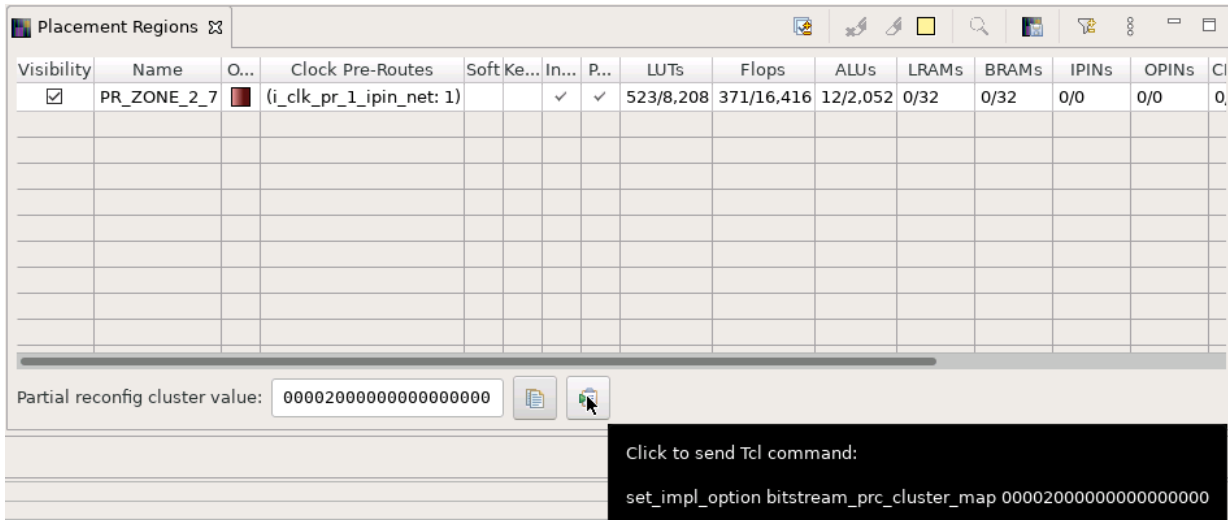


Figure 282 - Send Tcl Command Example

- In the **Options** view, click the **Bitstream Generation** tab.
- In the **Partial Reconfiguration** section, check the **Enable Partial Reconfiguration** checkbox and ensure that the **Partial Reconfig Cluster Map (hex)** is set to the value from the **Placement Regions** view:

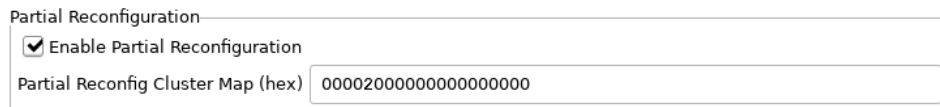


Figure 283 - Partial Reconfiguration Settings Example

- In the **FCU Reconfiguration** section, ensure that the **Lock FCU After Programming** item is checked:



Figure 284 - FCU Configuration Settings Example

Generate Partial Bitstream for PR Core 1A

- In the **Flow** view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**.

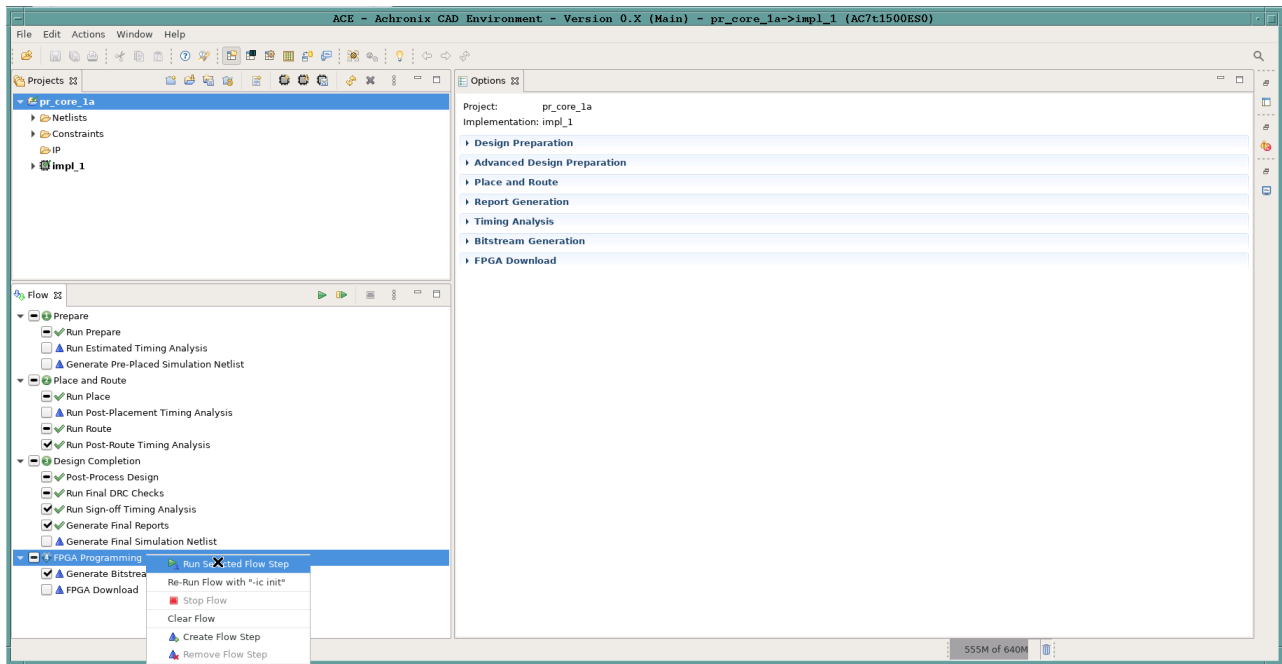



Figure 285 • Run FPGA Programming Flow Step Example

Save Region and Clock Pre-Routing Constraints to PDC File

Subsequent runs must use the same clock pre-routing and placement region constraints. These constraints must be saved to .pdc files and added to the ACE project.

Warning!

Adding .pdc files to an ACE project invalidates the flow and requires re-running the flow steps beginning with Run Prepare.

1. In the **Placement Regions View**, click the  **Save Placement Regions** button:

The screenshot shows the 'Placement Regions View' table. The table has columns for Visibility, Name, Overlay Color, Clock Pre-Rou..., Soft, Keep out, Include Routing, PR Zone, LUTs, Flops, ALUs, and a 'Save Placement Regions' button. The first row is highlighted in blue and contains the following data:



Visibility	Name	Overlay Color	Clock Pre-Rou...	Soft	Keep out	Include Routing	PR Zone	LUTs	Flops	ALUs	Save Placement Regions
<input checked="" type="checkbox"/>	PR_ZONE_2_7					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	505/8,640	358/17,280	12/2,160	

Figure 286 • Save Placement Regions Example

- The **Save Placement Regions** dialog appears. Click **Finish** to save the PR zone region constraints:

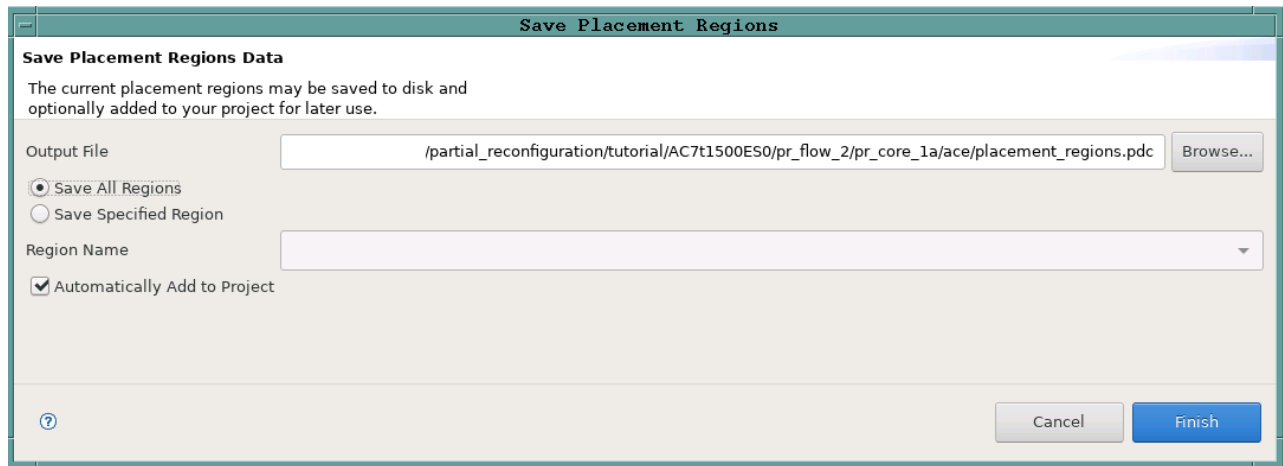


Figure 287 • Save Placement Regions Dialog Example

- The Tcl console displays the following and the `placement_regions.pdc` file is created.

```
create_region "PR_ZONE_2_7" {32 41 48 77} -snap fabric_clusters -type inclusive
-include_routing -pr_zone
add_region_find_insts "PR_ZONE_2_7" {find {*i_pr_core_1a*} -insts
```

- ACE automatically generates a clock pre-route file to the output area under `<ace_output_dir>/ace/impl_1/output/pr_core_1a_top_clock_preroutes.pdc`.

Note

This file should not be added to the ACE project as it is regenerated during each run. Make a copy of this file before adding it to the project.

Alternately, use the `save_clock_preroute` Tcl command to generate a `.pdc` file with the clock pre-route constraints.

Generate Partial Bitstream for PR Cores 1B 2A and 2B

Generate Partial Bitstream for PR Core 1B

- Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:
 - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1b`

Generate Partial Bitstream for PR Core 2A

- Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:

- `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a`

Generate Partial Bitstream for PR Core 2B

1. Repeat the same steps used in section 1 to generate the partial bitstream for PR Core 1 but use the following file:
 - `<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2b`

Generate Full Bitstream for Top-Level Design

Create I/O Ring Configuration for Target Board

In this design, a simple clock input is driving a PLL to create clocks for the core fabric. Also, a simple GPIO interface to display LEDs is in the top-level design.

I/O Ring Configuration Steps

1. In the IP Libraries view, under **IO Ring**, right-click **Clock I/O Bank** and select **New IP Configuration**.

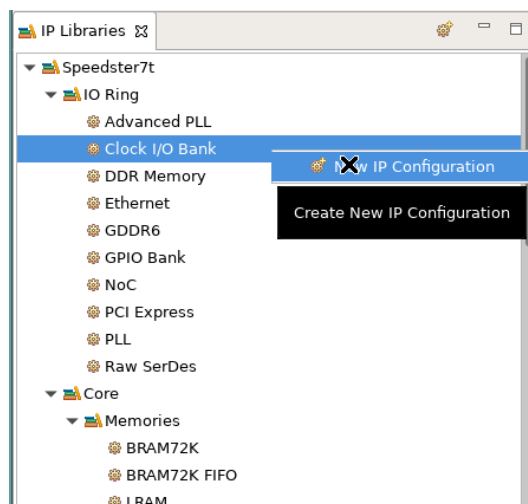


Figure 288 • Clock I/O Bank New IP Configuration Example

2. Configure the Clock I/O Bank for the PR Cores using the options shown in the following example:

Speedster7t Clock I/O Bank

Overview
All properties for the Clock I/O Bank can be edited below.

Bank Configuration

- ✓ Target Device: AC7t1S00E50
- ✓ Placement: CLKIO_SW
- ✓ VREF Source: Internal VDD
- ✓ Bank Voltage Level: 1.5

Enabl...	I/O Instance Name	Ball Name	Bump Name	Ball	Differ...	Signal...	Port Di...	Reset...	I/O Standard	Vref	Input...	Output Clock...	O...	Pull T...	Sl...	Tx Ta...	Drive...	Hyster...	ODT Mode	Rx Ta...
<input type="checkbox"/>	pr_core_clock_io_bank_sw_0_msio_p	CLKIO_SE_MSIO_P	CLKIO_SW_MSIO_P	AU35																
<input type="checkbox"/>	pr_core_clock_io_bank_sw_0_msio_n	CLKIO_SE_MSIO_N	CLKIO_SW_MSIO_N	AU34																
<input checked="" type="checkbox"/>	pr_core_clock_io_bank_sw_0_refio_p_0	CLKIO_SE_REFIO_P_0	CLKIO_SW_REFIO_P_0	AY34	<input checked="" type="checkbox"/>	Clock	INPUT		LVCMOS_15		100								High_Z	50
<input checked="" type="checkbox"/>	pr_core_clock_io_bank_sw_0_refio_n_0	CLKIO_SE_REFIO_N_0	CLKIO_SW_REFIO_N_0	AW34	<input checked="" type="checkbox"/>	Clock	INPUT		LVCMOS_15		100								High_Z	50
<input type="checkbox"/>	pr_core_clock_io_bank_sw_0_refio_p_1	CLKIO_SE_REFIO_P_1	CLKIO_SW_REFIO_P_1	AY36																
<input type="checkbox"/>	pr_core_clock_io_bank_sw_0_refio_n_1	CLKIO_SE_REFIO_N_1	CLKIO_SW_REFIO_N_1	AW36																

PLL and DLL Bank Reset Configuration (CAUTION: not CLKIO!)

- ✓ PLL/DLL Bank Reset Source: Internal Reset from FCU
- ✓ PLL/DLL Bank Global Reset Signal Name: input_reset_from_any_clkio_bank

DLL Configuration (CAUTION: not CLKIO!)

- ✓ Enable DLL
- ✓ DLL Reference Clock Name: clock_from_local_plls

Generate << Back Next >>

Figure 289 • SW Clock I/O Bank IP Configuration Example

3. In the **IP Libraries** view, under **IO Ring**, right-click the **PLL** and select **New IP Configuration**.
4. Configure the PLL for the PR core using the options shown in the following example:

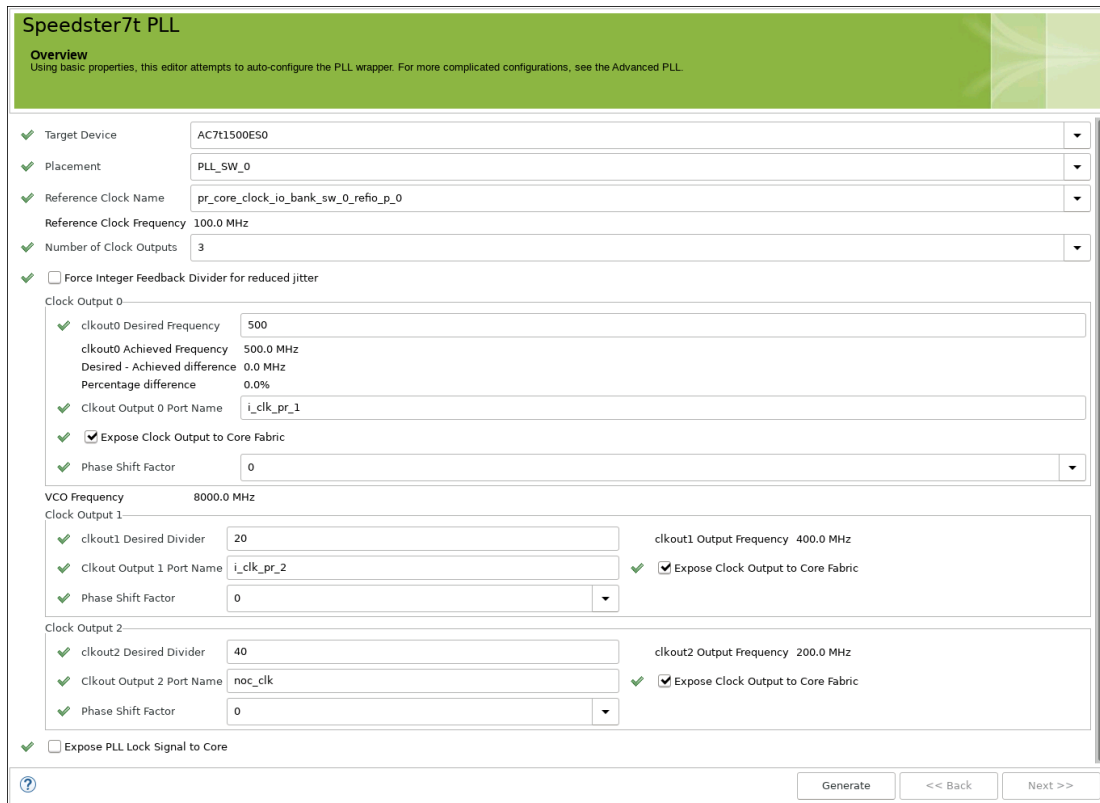


Figure 290 • SW PLL IP Configuration Example

5. In the **IP Libraries** view, under **IO Ring**, right-click **NoC** and select **New IP Configuration**.
6. Configure the 2D NoC IP for the PR Core using the options shown in the following example:

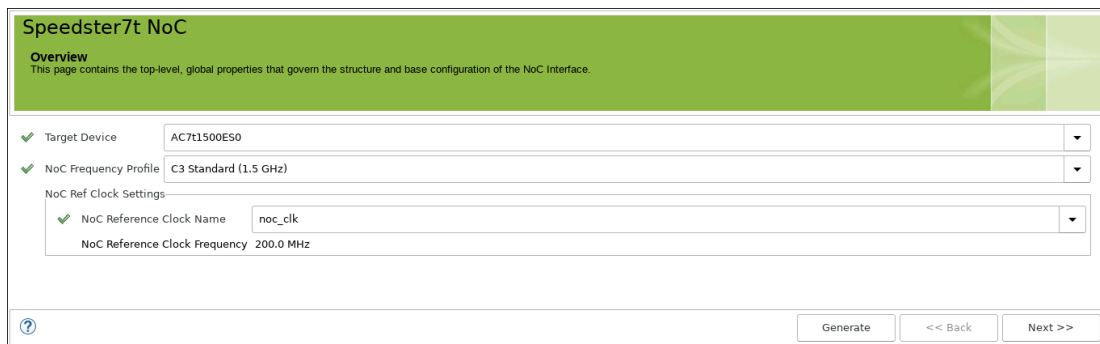


Figure 291 • 2D NoC IP Configuration Example

7. In the **IP Libraries** view, under **IO Ring**, right-click **Clock I/O Bank** and select **New IP Configuration**.
8. Configure the Clock I/O Bank for the top-level design using the options shown in the following example:

Speedster7t Clock I/O Bank

Overview
All properties for the Clock I/O Bank can be edited below.

Bank Configuration

- ✓ Target Device: AC711500E50
- ✓ Placement: CLKIO_NE
- ✓ VREF Source: Internal VDD
- ✓ Bank Voltage Level: 1.5

Enabl...	I/O Instance Name	Ball Name	Bump Name	Ball	Differ...	Signal...	Port Di...	Reset...	I/O Standard	Vref	Input...	Output Clock...	O...	Pull T...	Sl...	Tx Ta...	Drive...	Hyster...	ODT Mode	Rx Ta...
<input type="checkbox"/>	pr_top_clock_io_bank_ne_0_msio_p	CLKIO_NW_MSIO_P	CLKIO_NE_MSIO_P	U16																
<input type="checkbox"/>	pr_top_clock_io_bank_ne_0_msio_n	CLKIO_NW_MSIO_N	CLKIO_NE_MSIO_N	U17																
<input checked="" type="checkbox"/>	pr_top_clock_io_bank_ne_0_refio_p_0	CLKIO_NW_REFIO_P_0	CLKIO_NE_REFIO_P_0	N16	<input type="checkbox"/>	Clock	INPUT		LVCMS0_15		100								High_Z	50
<input type="checkbox"/>	pr_top_clock_io_bank_ne_0_refio_n_0	CLKIO_NW_REFIO_N_0	CLKIO_NE_REFIO_N_0	N17																
<input type="checkbox"/>	pr_top_clock_io_bank_ne_0_refio_p_1	CLKIO_NW_REFIO_P_1	CLKIO_NE_REFIO_P_1	R16																
<input type="checkbox"/>	pr_top_clock_io_bank_ne_0_refio_n_1	CLKIO_NW_REFIO_N_1	CLKIO_NE_REFIO_N_1	R17																

PLL and DLL Bank Reset Configuration (CAUTION: not CLKIO!)

- ✓ PLL/DLL Bank Reset Source: Internal Reset from FCU
- ✓ PLL/DLL Bank Global Reset Signal Name: input_reset_from_any_clkio_bank

DLL Configuration (CAUTION: not CLKIO!)

- ✓ Enable DLL
- ✓ DLL Reference Clock Name: clock_from_local_pll

Figure 292 • NE Clock I/O Bank IP Configuration Example

9. In the **IP Libraries** view, under **IO Ring**, right-click **PLL** and select **New IP Configuration**.
10. Configure the PLL for the top-level design using the options shown in the following example:

Speedster7t PLL

Overview
Using basic properties, this editor attempts to auto-configure the PLL wrapper. For more complicated configurations, see the Advanced

Target Device: AC7t1500ES0

Placement: PLL_NE_0

Reference Clock Name: pr_top_clock_io_bank_ne_0_refio_p_0
 Reference Clock Frequency: 100.0 MHz

Number of Clock Outputs: 1

Force Integer Feedback Divider for reduced jitter

Clock Output 0

clkout0 Desired Frequency: 500
 clkout0 Achieved Frequency: 500.0 MHz
 Desired - Achieved difference: 0.0 MHz
 Percentage difference: 0.0%

Clkout Output 0 Port Name: clk

Expose Clock Output to Core Fabric

Phase Shift Factor: 0

VCO Frequency: 8000.0 MHz

Expose PLL Lock Signal to Core

?
Generate
<< Back
Next >>

Figure 293 • NE PLL IP Configuration Example

11. In the **IP Libraries** view, under **IO Ring**, right-click **GPIO Bank** and select **New IP Configuration**.
12. Configure the GPIO Bank for the top-level design using the options shown in the following example:

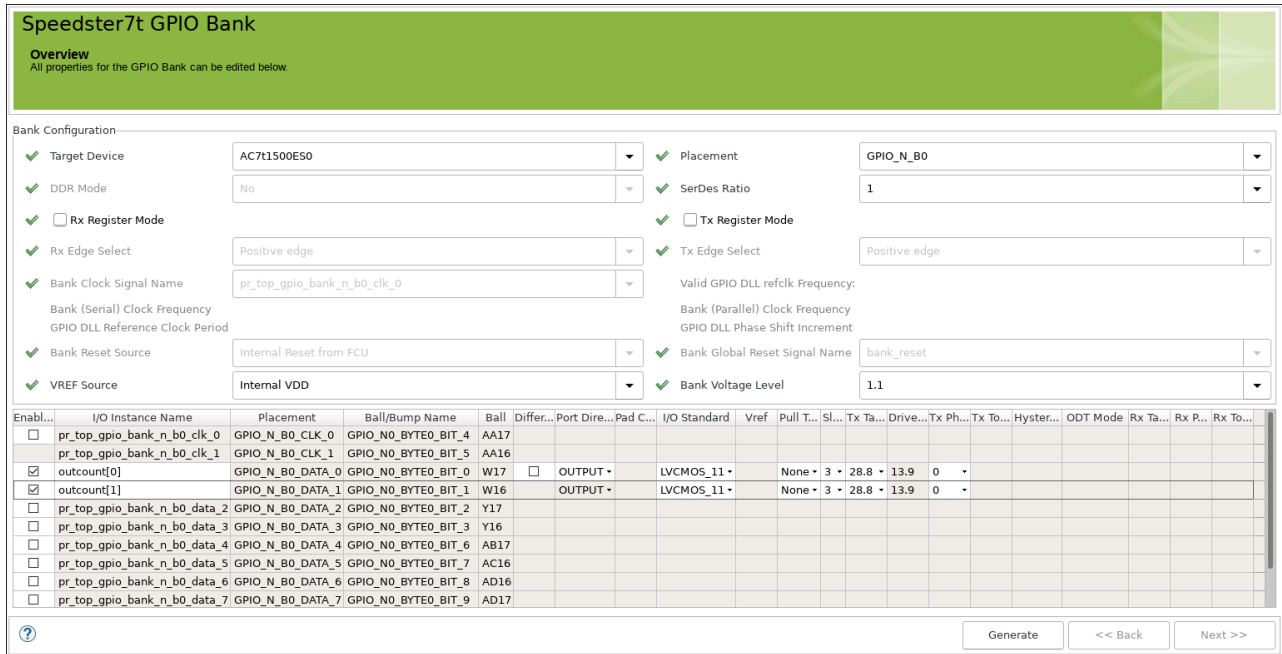


Figure 294 • GPIO Bank IP Configuration Example

- Click **Generate** to display the **Generate IO Ring Design Files** dialog and select **Add to active project**:

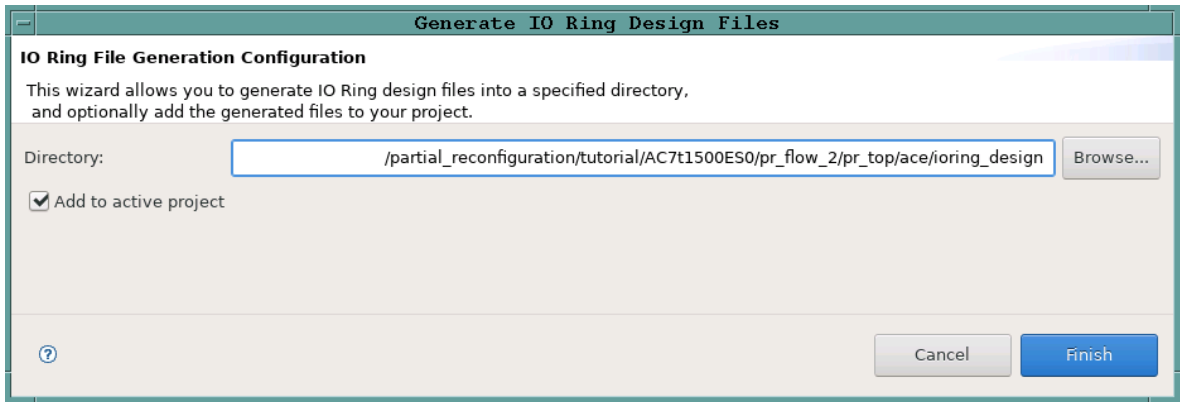


Figure 295 • Generate IO Ring Design Files Dialog Example

- Click **Finish** to generate the I/O ring design files.
- When complete, the following files should be added to the ACE project:

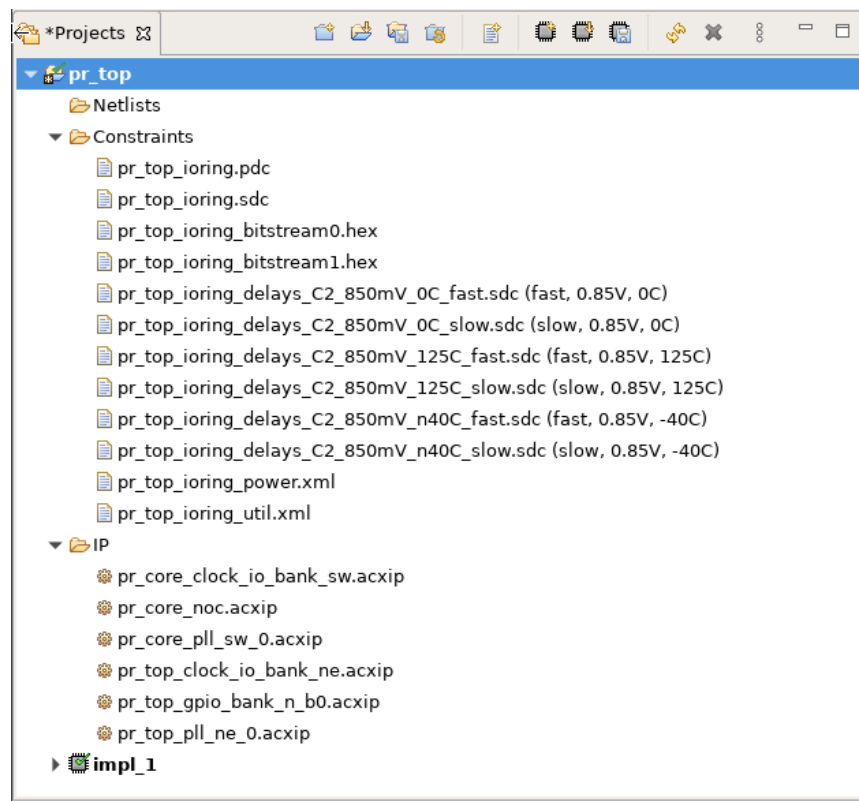


Figure 296 • I/O Ring Design Files

Synthesize Top-Level Design Using Synplify

1. Navigate to the following directory:


```
<ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/src
```

 Below this directory should be the following two directories:
 - rtl
 - constraints
2. Create a new Synplify project and add all RTL files from the `rtl` directory.
3. Add the `pr_top_timing.sdc` file from the `constraints` directory
4. In the **Project** → **Implementation Options** → **Device** tab, set **Technology** to **Achronix Speedster7t**.
5. Set **Part** to **AC7t1500ES0**.
6. Set **Package** to **F53**.
7. Set **Speed** to **C2**.
8. In the **Project** → **Implementation Options** → **Implementation Results** tab, set **Result Base Name** to **pr_top**.
9. In the **Project** → **Implementation Options** → **Verilog** tab, set **Top Level Module** to **pr_top**.

10. Set **Include Path Order** to `<ace_install_dir>/libraries`.
11. Add the `AC7t1500ES0_synplify .sv` file from the following directory:
 - `<ace_install_dir>/libraries/device_models`

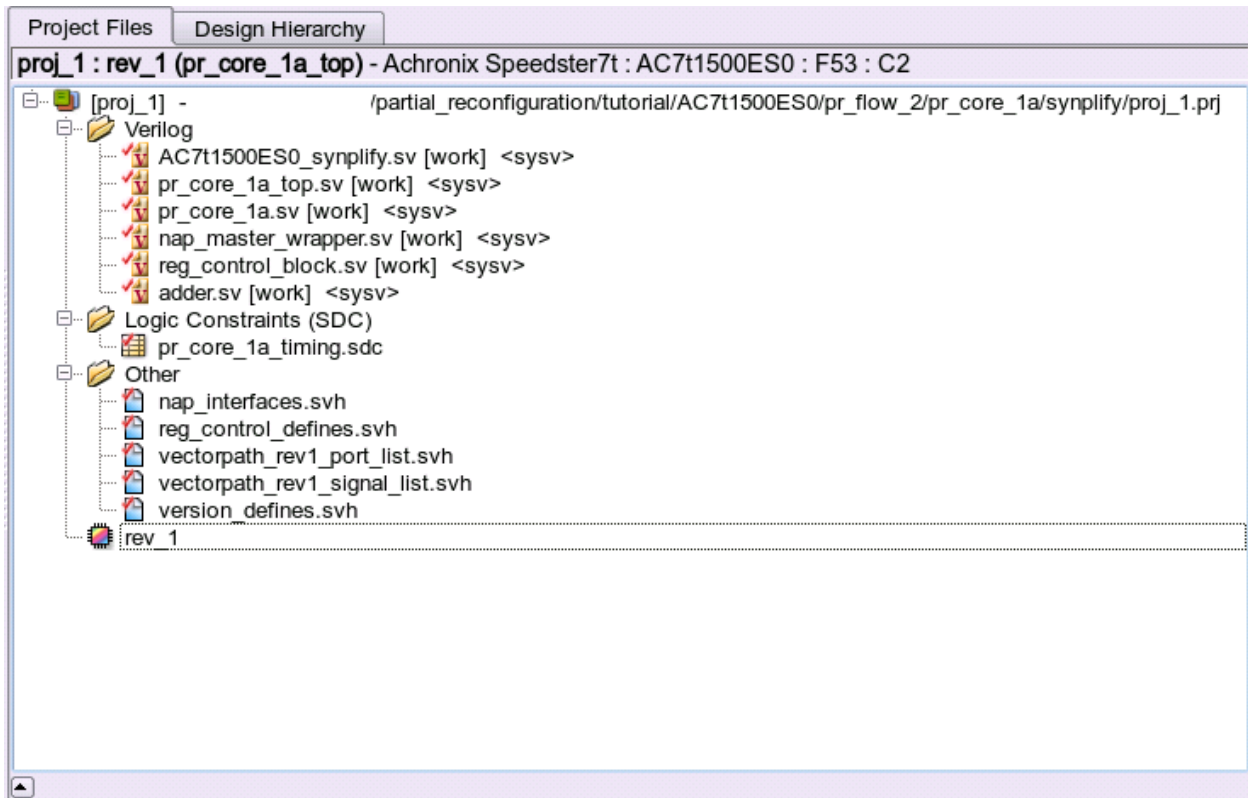




Figure 297 • Synplify Project Files Example

12. Run and compile the design using Synplify.
13. When the compile completes successfully, Synplify generates a gate-level netlist in the file:
 - `<synplify_out_dir>/rev_1/pr_top.vm`.

Run Top-Level Design Through Run Prepare in ACE

1. In the **Projects** view, click the () **Create a New Project** button.
2. Create a project using the pop-up dialog and click **Finish** when complete.
3. In the **Projects** view and click the () **Add Source Files to a Project** button.
4. Add the following files to the ACE project:
 - All the ioring files created in step 3
 - `<synplify_out_dir>/rev_1/pr_top.vm` (synthesized netlist created by Synplify in step 3)
5. In the **Options** view, select **Design Preparation** and make sure all options are correctly set.
6. Copy the option values in the following example:

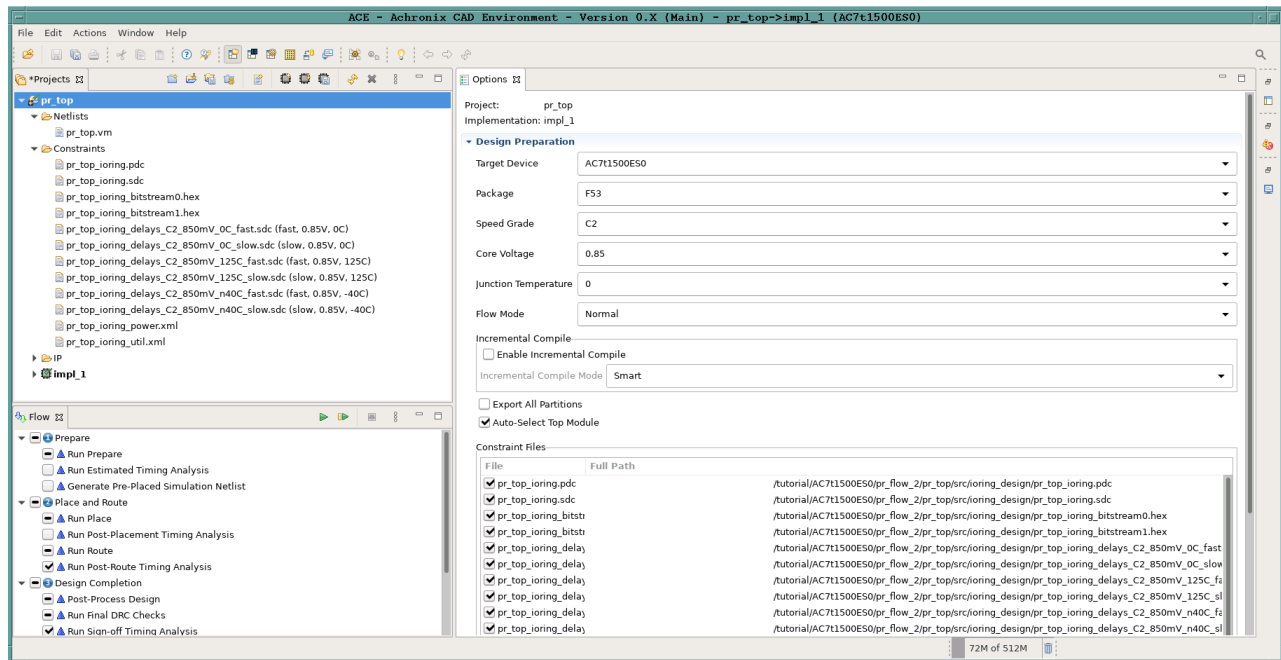



Figure 298 • Design Preparation Options Example

7. In the Flow view, right-click the **Run Prepare** step and select **Run Selected Flow Step**.

Set Up Keep Out Regions and Placement Region Constraints for Static Logic

Keep out regions must be created in order to define "holes" in the core fabric that the top-level logic should not enter. These holes are later filled by partial reconfiguration bitstreams.

1. Ensure that the Run Prepare flow step has completed successfully.
2. In the **Floorplanner** view, click the () **Placement Region Tool** button.
3. Click and drag in the Floorplanner to draw a PR Zone for the PR Core.

Note

The bounding box of this keep out region must be the same as the one set for PR core 1A in section 1d.

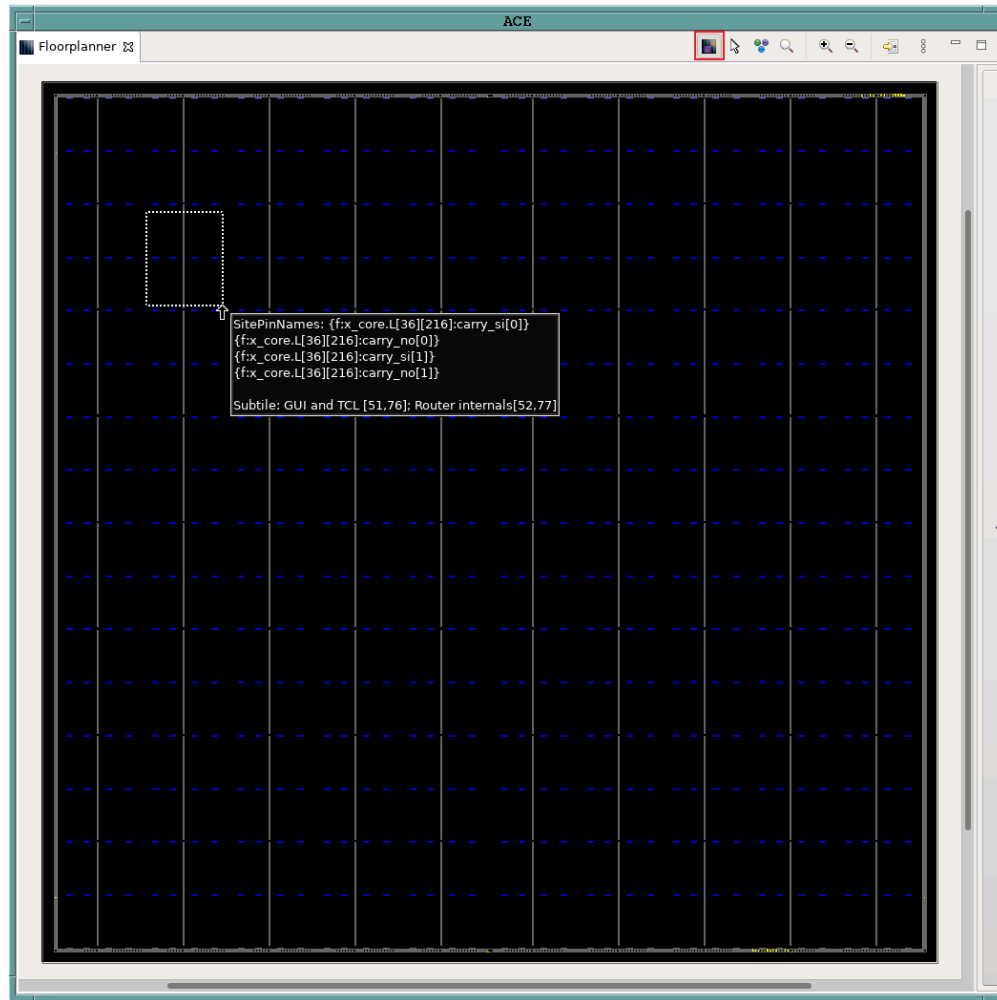


Figure 299 - Creating a Placement Region Zone Example

4. After releasing the left mouse button, the **Create Placement Region** dialog appears.
5. Set **Region Alignment** to **Snap to Fabric Clusters**.
6. Set **Region Type** to **Keepout**.

Note

The bounding box of this region has to match that of the PR Zone for PR Core 1A created in step 1c.

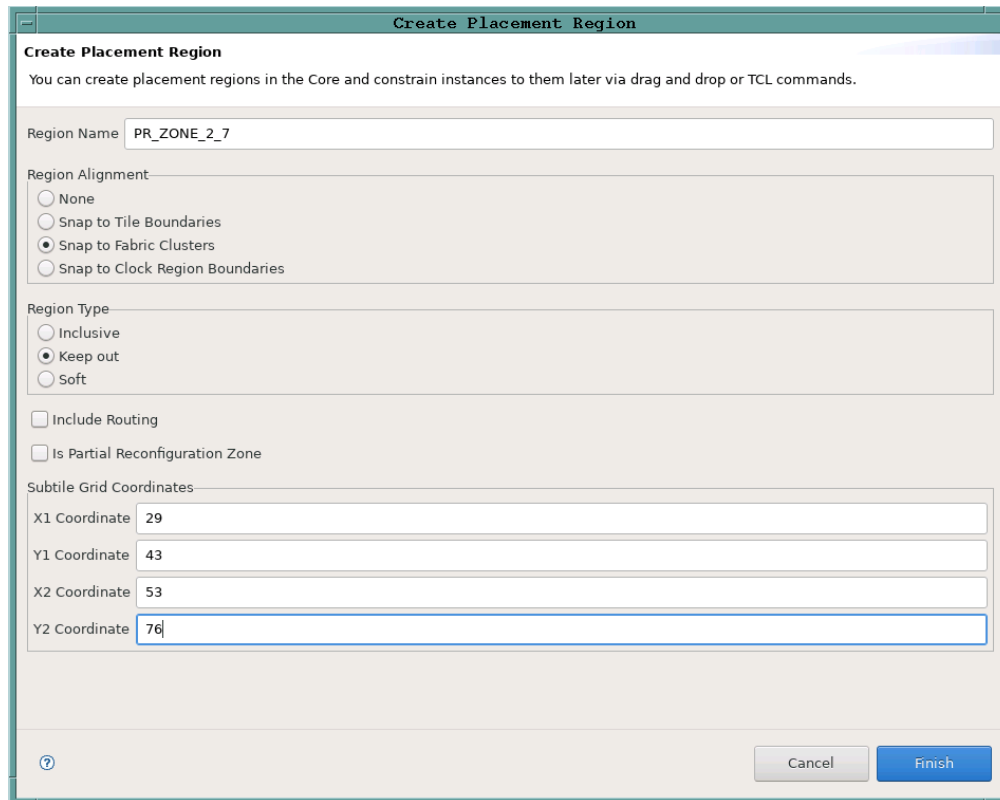


Figure 300 - Create Placement Region Dialog Example

7. Repeat steps 2 through 6 to create a keep out region for the PR zone for PR core 2A.
8. Optionally, create a region for the top-level logic. The floorplanner view should resemble the following example after performing these steps:
 - Red region – keep out region for PR core 1A.
 - Light blue region – keep out region for PR core 2A.
 - Yellow region – region for top-level logic.

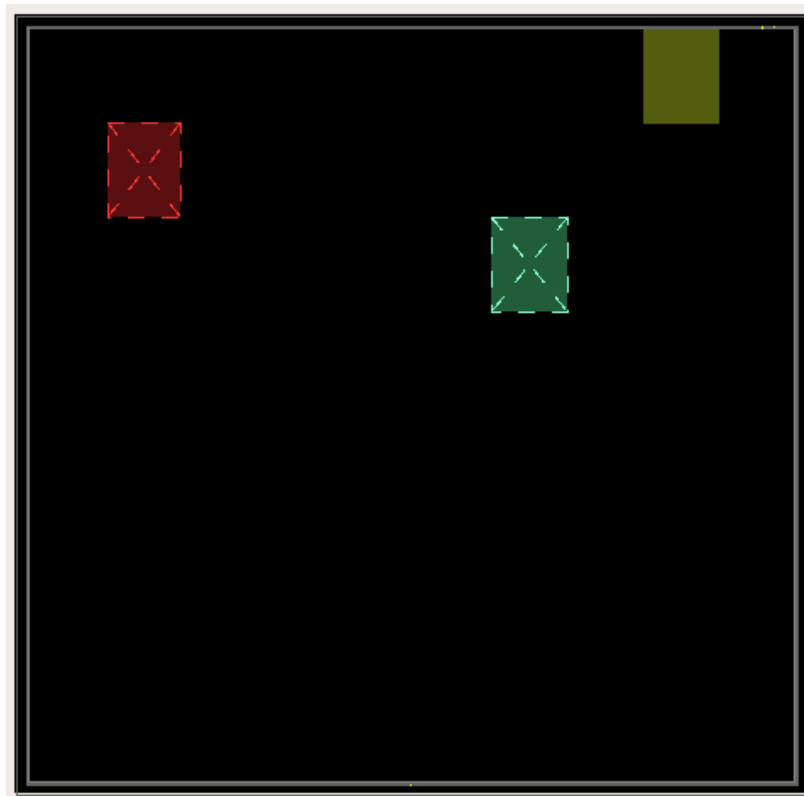


Figure 301 • Floor Planner Regions Example

Set Up Clock Pre-Routing Constraints

1. In the **Placement Regions** view, right-click the box below the **Clock Pre-Routes** heading and select **Configure Clock Pre-Routes**:

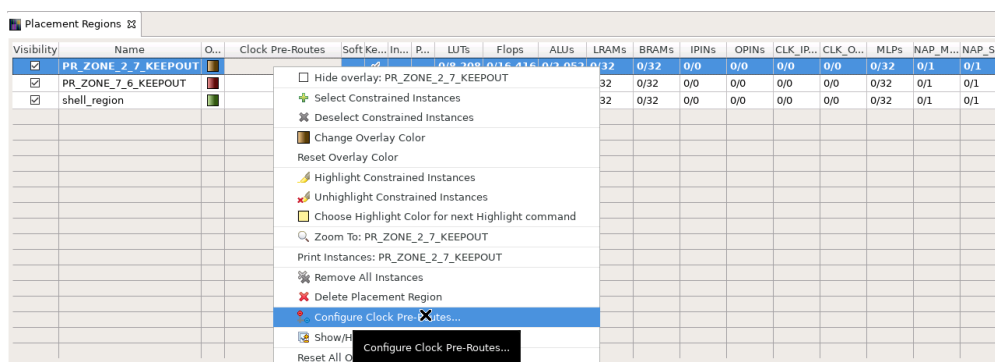


Figure 302 • Configure Clock Pre-Routes Example

2. The **Configure Clock Pre-Routes** dialog appears.

3. Ensure that the clock net is pre-routed into the same track specified in step 1d.

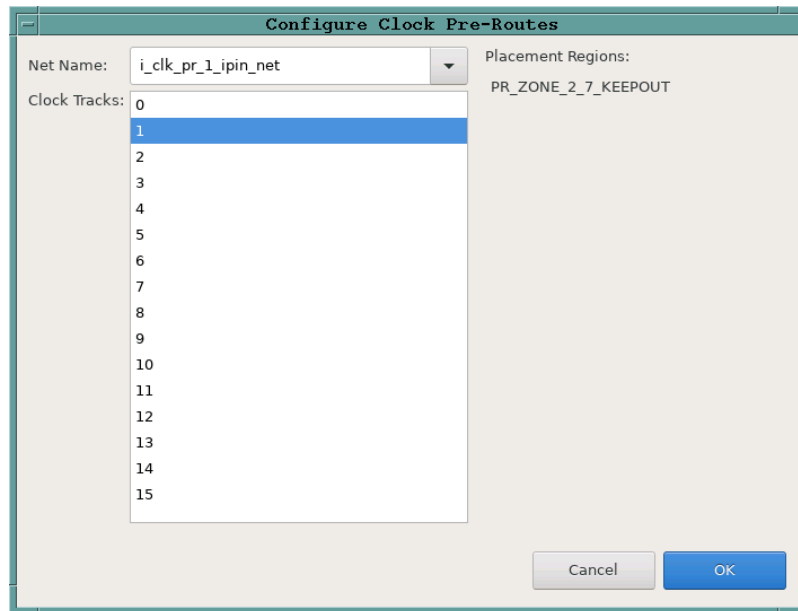


Figure 303 • Configure Clock Pre Routes Dialog Example

4. Click **OK**. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_1_ipin_net { 1 } -placement_regions { PR_ZONE_2_7 }
```

5. Repeat steps 1 and 2 above to route the clock net for PR core 2 to the PR core 2A keep out region.
6. The Tcl Console displays the following:

```
add_clock_preroute i_clk_pr_2_ipin_net { 2 } -placement_regions { PR_ZONE_7_6 }
```

Run Place-and-Route in ACE for Top-Level Design

1. In the **Flow** view, right-click the **Place and Route** step and select **Run Selected Flow Step**.

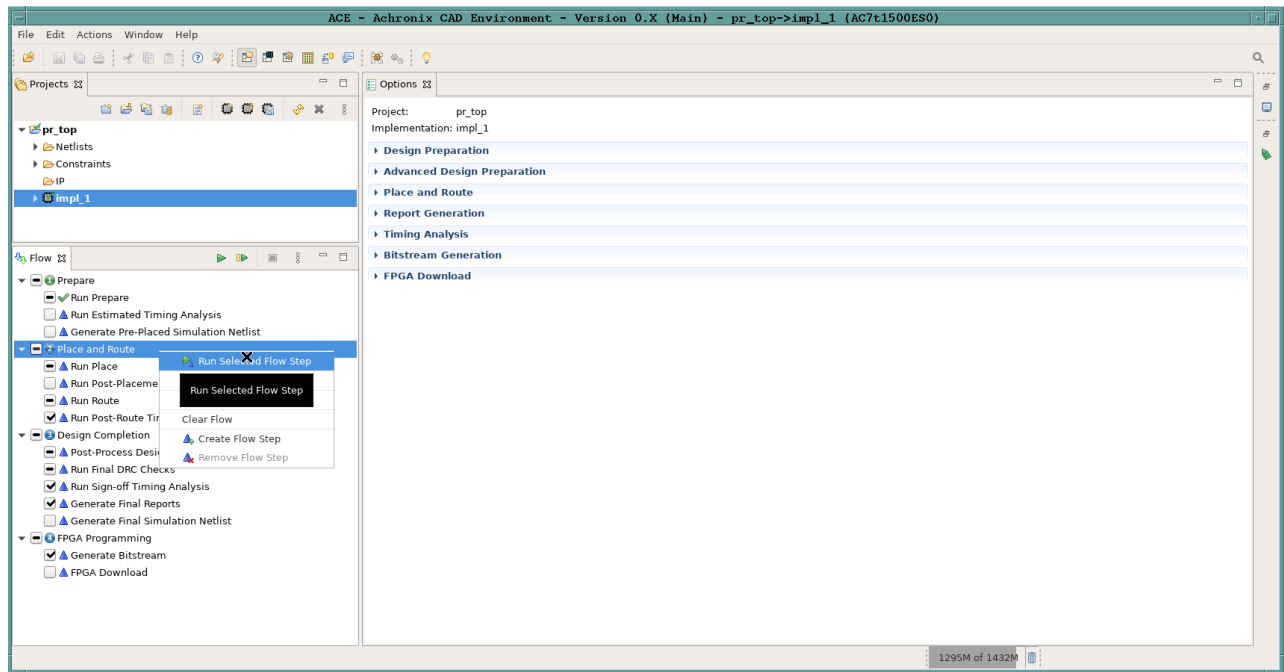


Figure 304 - Run Place and Route Flow Step Example

Run Final Sign-Off Timing Analysis in ACE for Top-Level Design

1. Ensure that the Place and Route flow step completed successfully.
2. In the **Flow** view, right-click the **Design Completion** step and select **Run Selected Flow Step**.

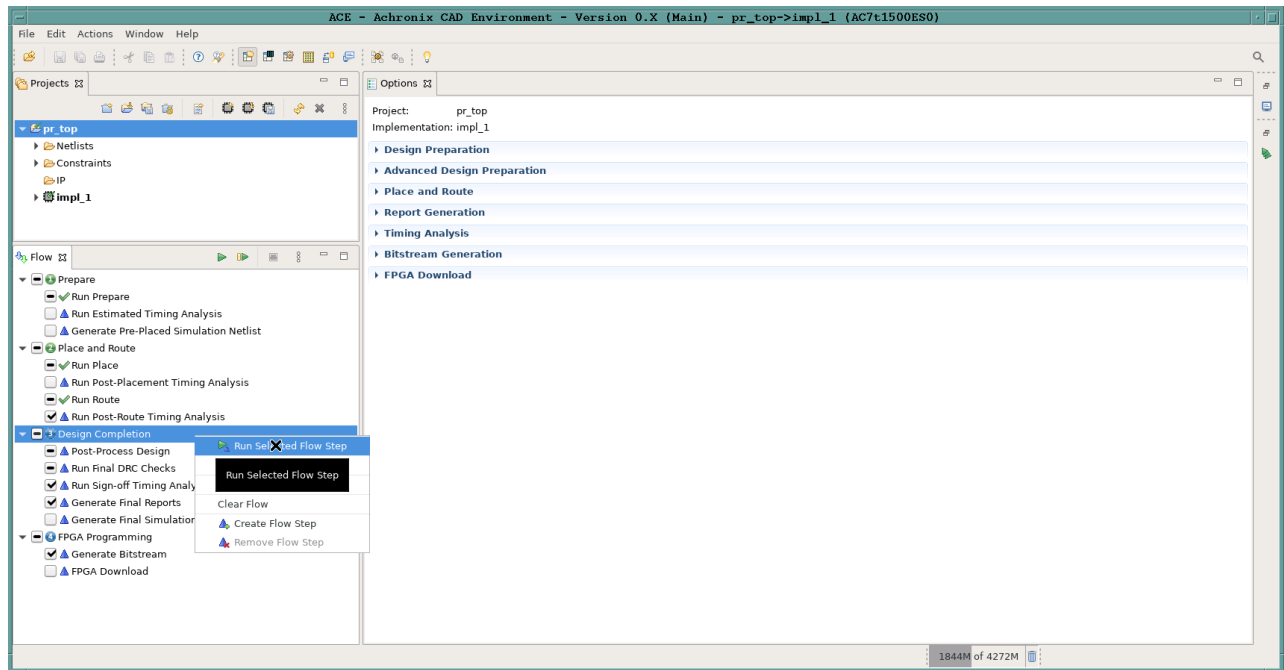


Figure 305 - Run Design Completion Flow Step Example

Generate Base Bitstream for Top-Level Design

1. Ensure that the Design Completion flow step completed successfully.
2. In the Flow view, right-click the **FPGA Programming** step and select **Run Selected Flow Step**.
3. Ensure that the **Enable Partial Reconfiguration** option is NOT selected in order to create a base bitstream.

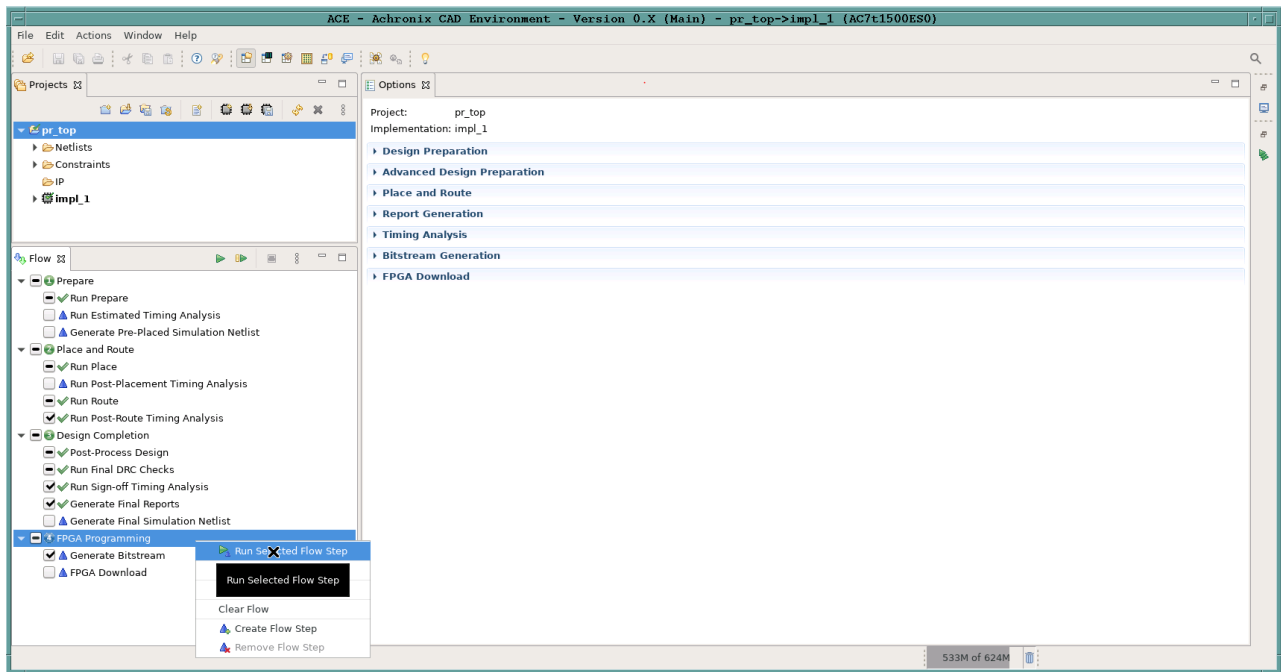


Figure 306 - Run FPGA Programming Flow Step Example

4. In the **Options** view, click the **Bitstream Generation** tab.
5. Scroll down to the **Partial Reconfiguration** section and ensure that the **Lock FCU After Programming** item is checked.



Figure 307 - Lock FCU After Programming Option Example

Save Placement Region and Clock Pre-Routing Constraints to PDC File

1. Repeat the steps in section 1i to save the region and clock pre-routing and placement region constraints to a .pdc file.

Bitstream Programming Sequence

Apply Power to Board

1. Apply power to the board and ensure cables are properly connected.

Program Top-Level Design Full Bitstream

1. Run the following commands.

```
set jtag_id [jtag::get_connected_devices]
jtag::open $jtag_id
jtag::configure_scan_chain $jtag_id AC7t1500ES0 0 0 0 -single_device
jtag::ac7t1500_initialize_fcu $jtag_id -reset
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_top/ace/impl_1/output/
pr_top.hex
```

Program PR Core 1A Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1a/ace/impl_1/output/
pr_core_1a.hex
```

Run PR Core 1A Test Scripts

1. Run the following command.
The script contains NAP reads from PR core 1A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/
pr_flow_2/test_scripts/read_pr_core_1a.tcl
```

Program PR Core 2A Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream
jtag::ac7t1500_initialize_fcu $jtag_id
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/ace/impl_1/output/
pr_core_2a.hex
```

Run PR Core 2A Test Scripts

1. Run the following command.
The script contains NAP reads from PR core 2A.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_2/test_scripts/read_pr_core_2a.tcl
```

Program PR Core 1B Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream  
jtag::ac7t1500_initialize_fcu $jtag_id  
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/  
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_1b/ace/impl_1/output/  
pr_core_1b.hex
```

Run PR Core 1B Test Scripts

1. Run the following command.
The script contains NAP reads from PR core 1B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_2/test_scripts/read_pr_core_1b.tcl
```

Program PR Core 2B Partial Bitstream

1. Run the following commands.

```
#Program partial bitstream  
jtag::ac7t1500_initialize_fcu $jtag_id  
jtag::ac7t1500_program_bitstream $jtag_id <ace_install_dir>/examples/  
partial_reconfiguration/tutorial/AC7t1500ES0/pr_flow_2/pr_core_2a/ace/impl_1/output/  
pr_core_2b.hex
```

Run PR Core 2B Test Scripts

1. Run the following command.
The script contains NAP reads from PR core 2B.

```
source <ace_install_dir>/examples/partial_reconfiguration/tutorial/AC7t1500ES0/  
pr_flow_2/test_scripts/read_pr_core_2b.tcl
```

Updating ROM, BRAM, or LRAM Memory Initialization Contents in the Bitstream

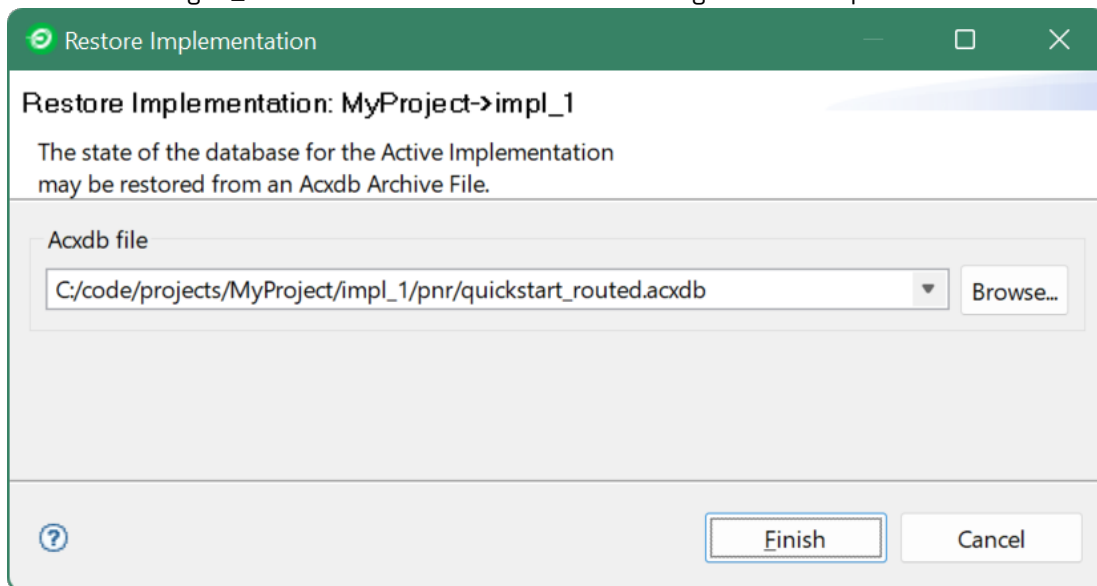
QUESTION: Once we've synthesized a design with ROMs and run through ACE P&R, is there a way to generate a new bitstream with updated contents for the ROMs without having to go through resynthesize or ACE P&R again?

ANSWER: Yes. There are 2 ways, depending on the method you used to initialize the ROM memory contents.

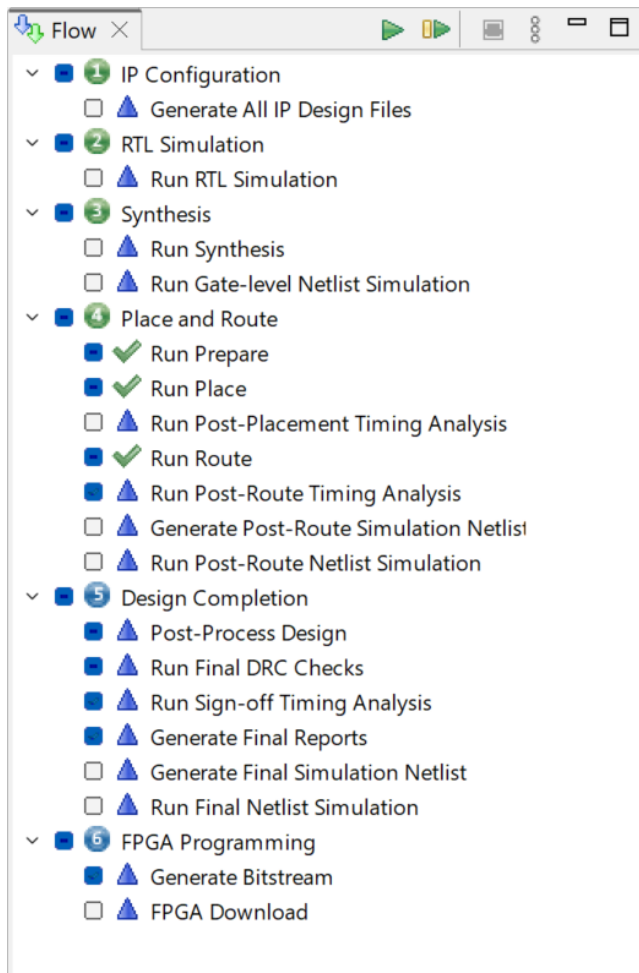
Initialized using a mem_init file

If you used a mem_init_file, then:

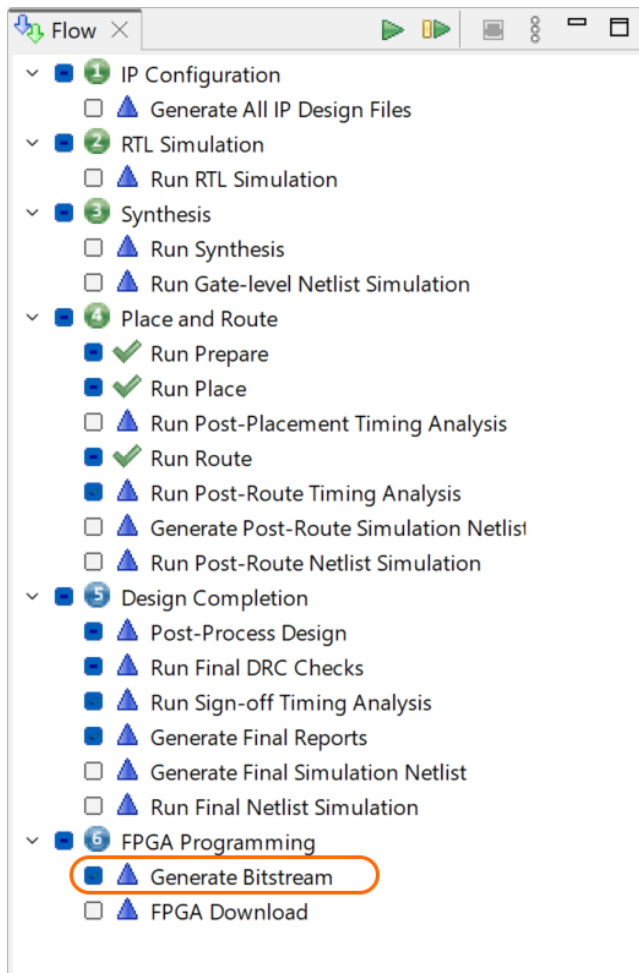
1. Launch ACE.
2. Load your ACE project (restore_project).
3. Load your post-route ACXDB database (restore_impl). In the GUI, this can be done using the 'Restore Implementation' action.
 - a. Select the <design>_routed.acxdb file. This will load the design and all the place and route information.



You will see the green checkboxes on the Run Prepare, Run Place, and Run Route flow steps in the Flow View in the GUI.



4. Now, you can keep your ACE session open and update the `mem_init_file` contents on disk as many times as you want, and generate new bitstreams using the 2 steps below:
 - a. Update the `mem_init_file` and save to disk.
 - b. Double-click the 'Generate Bitstream' flow step in the Flow View (or call “run -step write_bitstream”) to generate the new bitstream with only the ROM contents updated.



Initialized using initd* parameters

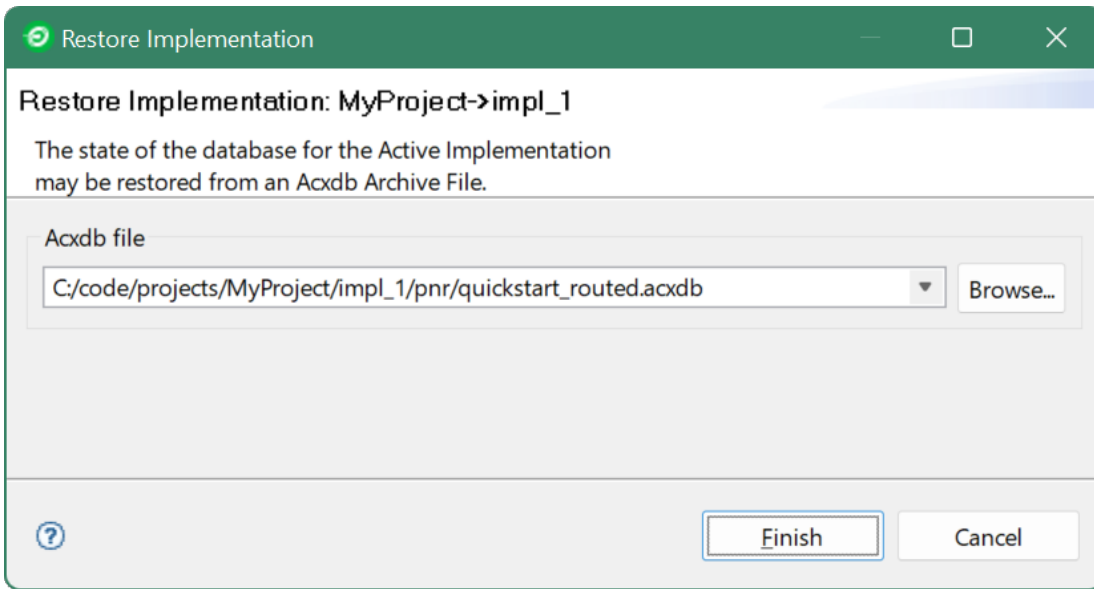
If you used the `initd*` parameters on the BRAM/LRAM instance, then:

1. Launch ACE.
2. Load your ACE project (`restore_project`).
3. Load your post-route ACXDB database (`restore_impl`).

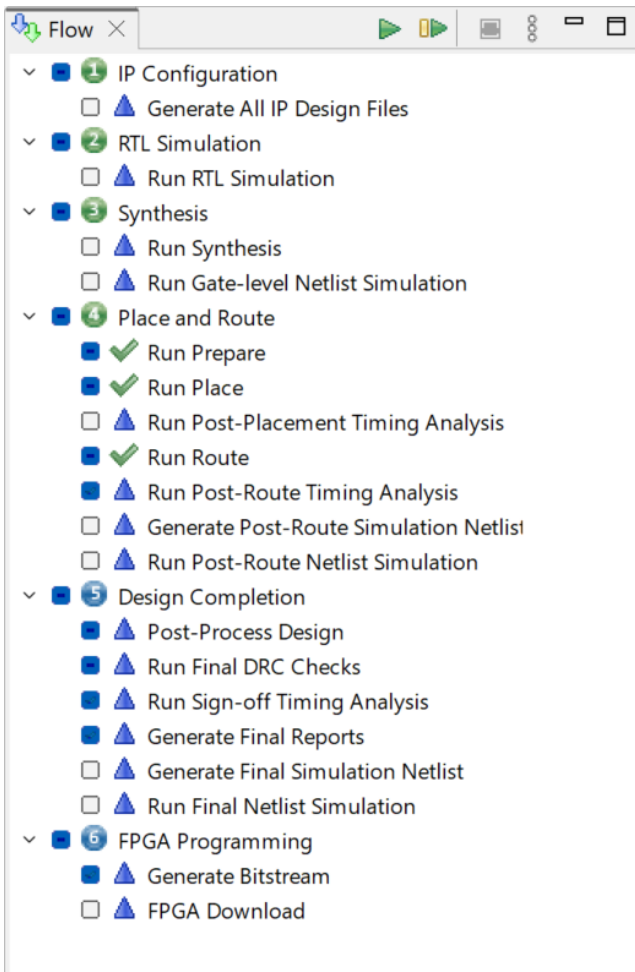
In the GUI, this can be done using the 'Restore Implementation' action.

Select the `<design>_routed.acxdb` file.

This will load the design and all the place and route information.



You will see the green checkboxes on the Run Prepare, Run Place, and Run Route flow steps in the Flow View in the GUI.

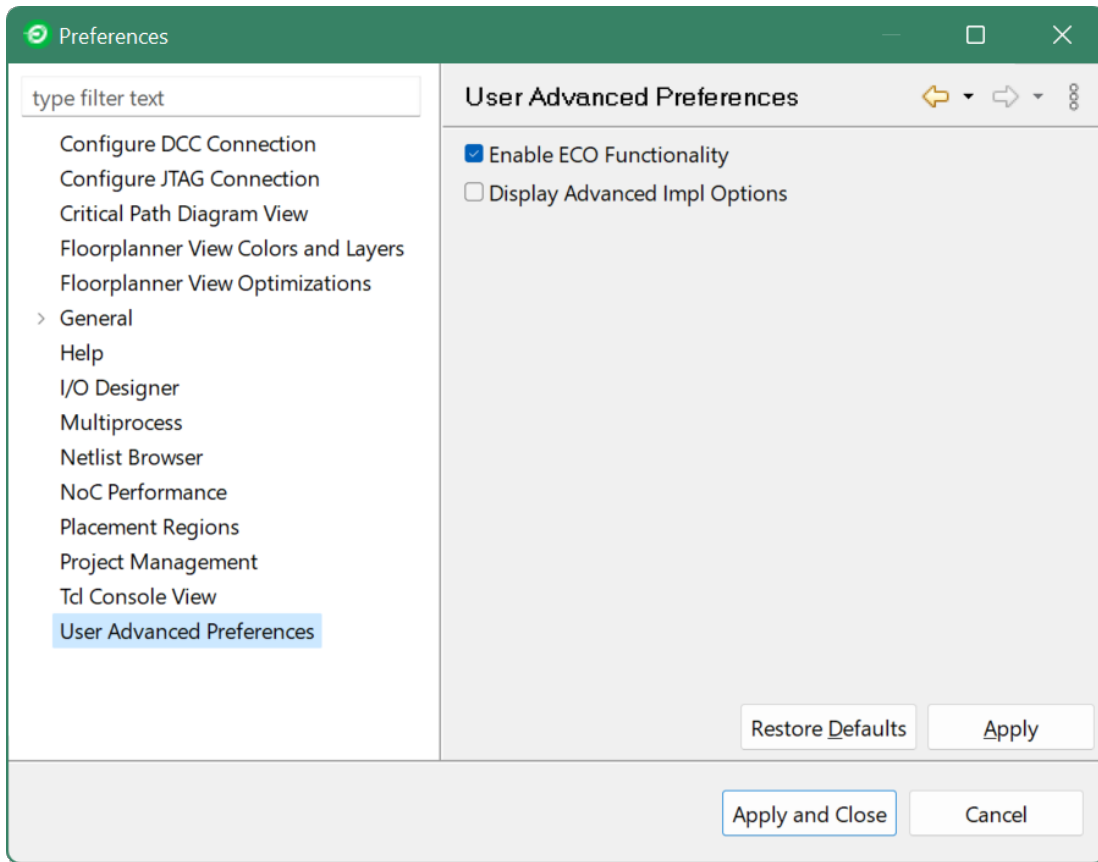


4. Now, you can keep your ACE session open and use ECO commands to update the `initd*` parameters as many times as you want, and generate new bitstreams using the 2 steps below:
 - a. Use the `set_property` command to override the `initd*` parameters on the BRAM/LRAM instance.

Example: `set_property {initd_12} {72'hABCD1234ABCD12345} {i:design.hierarchy.path.to1.ram.instance}`.

In the GUI, you can also go to the menu **Window | Preferences**, and go to **User Advanced Preferences**, and check the box to 'Enable ECO Functionality'.

¹ <http://design.hierarchy.path.to>

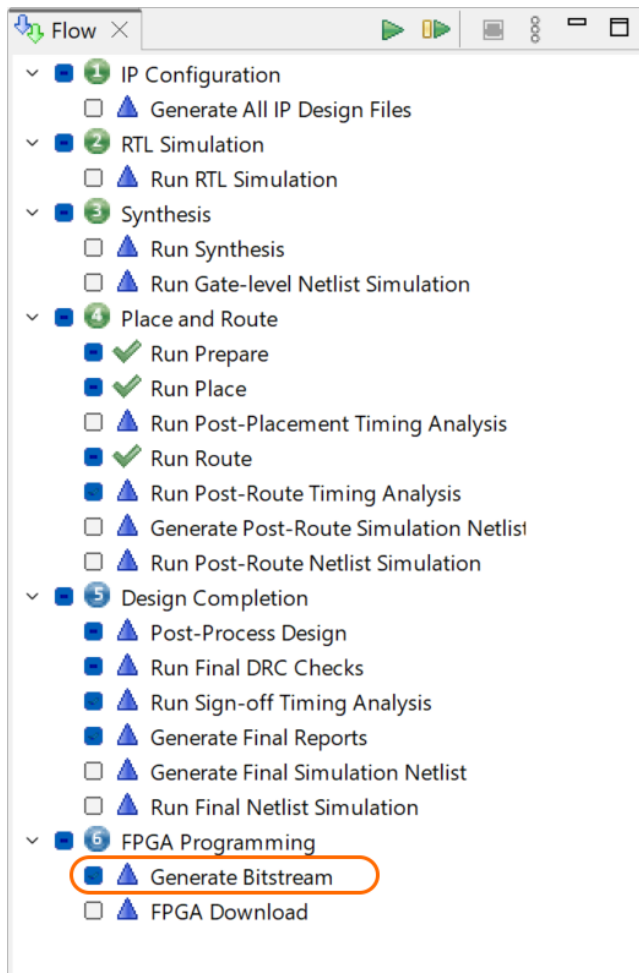


Right-click on the ROM instance in the Floorplanner, Search, Selection, or Netlist Browser views, and select 'Display Properties'. This updates the contents of the Properties view. You can now edit the `initd*` parameters on the Parameters tab of the wizard.

The screenshot shows a 'Properties' window for an instance named 'rd_data_2_0_0'. The window contains a table with three columns: 'Type', 'Name', and 'Value'. The 'Type' column for all entries is 'Parameter'. The 'Name' column lists various configuration parameters, and the 'Value' column shows their corresponding values in hexadecimal or binary notation. The 'Parameters' tab is selected at the bottom of the window.

Type	Name	Value
Parameter	acx_memtest	1'b0
Parameter	byte_width	9
Parameter	cfgbram_assist	6'h5
Parameter	cfgbram_clkpwma	3'h0
Parameter	cfgbram_clkpwmb	3'h0
Parameter	cfgbram_clock_pulse_widtha_first	15'h0007
Parameter	cfgbram_clock_pulse_widtha_second	15'h0007
Parameter	cfgbram_clock_pulse_widthb_first	15'h0007
Parameter	cfgbram_clock_pulse_widthb_second	15'h0007
Parameter	cfgbram_rda	3'h2
Parameter	ecc_decoder_enable	0
Parameter	ecc_encoder_enable	0
Parameter	initd_0	72'h0000000000000000
Parameter	initd_1	72'h0000000000000000
Parameter	initd_2	72'h0000000000000000
Parameter	initd_3	72'h0000000000000000
Parameter	initd_4	72'h0000000000000000
Parameter	initd_5	72'h0000000000000000
Parameter	initd_6	72'h0000000000000000
Parameter	initd_7	72'h0000000000000000
Parameter	initd_8	72'h0000000000000000
Parameter	initd_9	72'h0000000000000000
Parameter	initd_10	72'h0000000000000000

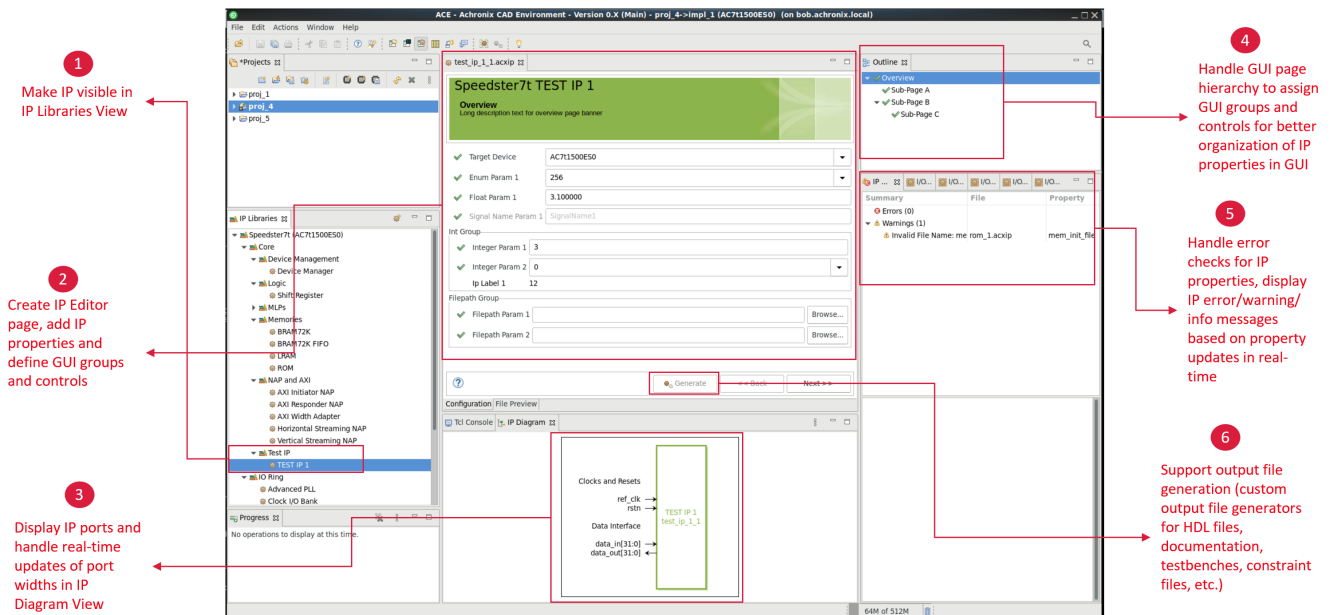
- b. Double-click the Generate Bitstream flow step in the Flow View (or call “run -step write_bitstream”) to generate the new bitstream with only the ROM contents updated



Plugging in Custom IP Generator Tools

The IP Generator Plugin Framework enables the creation of IP Generation tools in the ACE GUI by defining the specification for IP properties, GUI controls and callbacks for property updates, checks and output file generation in TCL. This is particularly useful for Speedcore eFPGA customer who want to provide custom IP or ASIC configuration tools to their end users which are fully integrated into ACE (including the GUI).

This document section describes the features, usage and setup for IP Generator Plugin Framework.



File Location and Directory Organization

ACE Installation Directory

During ACE start-up, the built-in ACE soft IP libraries in the ACE install directory are searched. The TCL files that must be sourced will be located in `<install>/libraries/<library>/macros/<ip_name>/src/gui/ip_plugin.tcl`

Each IP base directory must follow this directory and file naming convention: `src/gui/ip_plugin.tcl`



ACE loads the following TCL files automatically at initialization: `<install>/libraries/*/macros/*/src/gui/ip_plugin.tcl`

Loading User-defined Directory Paths using Environment Variables

IP Generator Plugin definitions can also be loaded by specifying base directory paths using environment variables. The user-defined search paths are defined in `ACE_IPGEN_PLUGIN_PATH` environment variable. Multiple base directories are separated by `'.'`.

For each base directory, the file `<base_dir>/src/gui/ip_plugin.tcl` will be sourced.

For example, for following file paths:

```
/home/$USER/my_ip_1/src/gui/ip_plugin.tcl
```

```
/home/$USER/my_ip_2/src/gui/ip_plugin.tcl
```

Loading user-defined directory paths using environment variable

```
1 export ACE_IPGEN_PLUGIN_PATH=/home/$USER/my_ip_1:/home/$USER/my_ip_2
```

IP Generator Plugin Development

The ip_plugin.tcl must follow the Model-View-Controller architecture.

The IP Creation and IP property definitions define the Model.

The IP GUI layout definition defines the View.

The Callback functions work as the Controller.

Resources:

- [TCL Commands for IP Generator Plugin Framework \(page 722\)](#)

IP Creation

The first phase of IP creation involves defining the IP name, library, version description and resource types needed by this IP.

IP Creation

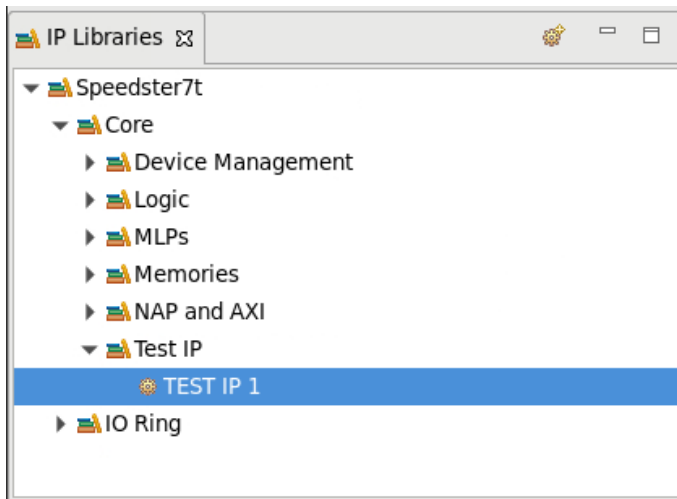
```
1 package require ipgen
2 #
3 #####
4 #####
5 # test example
6 #
7 #####
8 #####
9 # Create the top-level IP Gen tool definition
10 # if this is called a second time, it wipes out the existing IP Gen tool
11 # definition. This lets you "source" this TCL script multiple times in an ACE
12 # session when you are developing the IP Gen tool
13 ace_ip::create_ip "TEST IP 1" "Speedster7t" "1.0"
14 # Determines where the IP shows up in the IP Libraries View tree structure,
15 # and possibly in documentation structure
16 ace_ip::set_ip_category { "Core" "Test IP" "TEST IP 1" }
17 # Top level description of the IP. Shows up in tool-tip in IP Libraries
18 # View, and as overview text in the documentation
19 ace_ip::set_ip_description "This page contains the top-level, global
20 # properties that govern the structure and base configuration of the Test Soft
21 # IP wrapper."
```

```

15 |
16 | #
   | #####
   | #####
17 | # Define Resource Types needed by this IP
18 | # Determines which devices are supported. ACE looks at the primitive
   | resource types supported (NAP_M, BRAM72K, etc) by a given device to determine
   | if that device can support this IP.
19 | # If no resource type is defined, then we assume it is built out of pure
   | synthesizable logic that is supported on all devices
20 |
21 | # set_ip_resource_types <list_of_cell_types> [-ip_name <ip_name>]
22 | ace_ip::set_ip_resource_types {"NAP_I"}

```

The IP library view in ACE shows the Test IP as follows:



IP Property Definitions

IP Property Definitions

```

1 | #
   | #####
   | #####
2 | # Define the IP Properties
3 |
4 | ace_ip::add_ip_enum_property "enumParam1" "Enum Param 1" "256" {"32" "64"
   | "128" "256"} -description "Enum Param 1 description." -enabled "true"
5 | ace_ip::set_ip_property_verilog_param_attributes "enumParam1" -verilog_param
   | "false"
6 |

```

```
7 ace_ip::add_ip_enum_property "enumParam2" "Enum Param 2" "X" {"X" "1" "2" "3"
  "4" "5" "6" "7" "8"} -description "Enum Param 2 description." -enabled
  "true"
8 ace_ip::set_ip_property_verilog_param_attributes "enumParam2" -verilog_param
  "true" -format "sized_hex" -to_upper "false" -num_bits 4
9
10 ace_ip::add_ip_enum_property "enumParam3" "Enum Param 3" "X" {"X" "1" "2" "3"
  "4" "5" "6" "7" "8" "9" "10"} -description "Enum Param 3 description."
  -enabled "true"
11 ace_ip::set_ip_property_verilog_param_attributes "enumParam3" -verilog_param
  "true" -format "sized_hex" -to_upper "false" -num_bits 4
12
13 ace_ip::add_ip_enum_property "enumParam4" "Enum Param 4" "1/2" {"0" "1/2"
  "1/3" "1/4" "1/5" "1/6" "1/7" "1/8" "1/9" "1/10" "1" "CUSTOM"} -description
  "Enum Param 4 description." -enabled "true"
14 ace_ip::set_ip_property_verilog_param_attributes "enumParam4" -verilog_param
  "true" -format "quoted_string" -to_upper "false"
15
16 ace_ip::add_ip_enum_property "enumParam5" "Enum Param 5" "1/2" {"0" "1/2"
  "1/3" "1/4" "1/5" "1/6" "1/7" "1/8" "1/9" "1/10" "1" "CUSTOM"} -description
  "Enum Param 5 description." -enabled "true"
17 ace_ip::set_ip_property_verilog_param_attributes "enumParam4" -verilog_param
  "true" -format "quoted_string" -to_upper "false"
18
19 ace_ip::add_ip_binary_property "binaryParam1" "Binary Param 1"
  "00010101010101010101010101010101" 32 -min "000000000000000000000000000000"
  -max "11111111111111111111111111111111" -description "Binary Param 1
  description." -enabled "false"
20 ace_ip::set_ip_property_verilog_param_attributes "binaryParam1"
  -verilog_param "true" -to_upper "false" -num_bits 32
21
22 ace_ip::add_ip_hex_property "hexParam1" "Hex Param 1" "15555555" 32 -min
  "00000000" -max "FFFFFFFF" -description "Hex Param 1 description." -enabled
  "false"
23 ace_ip::set_ip_property_verilog_param_attributes "hexParam1" -verilog_param
  "false" -to_upper "false" -num_bits 32
24
25 ace_ip::add_ip_string_property "stringParam1" "String Param 1" "my Default"
  -min_characters "2" -max_characters "32" -valid_characters_regex "\[a-zA-
  Z0-9\]\.\\*" -description "string param 1 description" -enabled "true"
26 ace_ip::set_ip_property_verilog_param_attributes "stringParam1"
  -verilog_param "true" -to_upper "true"
27
28 ace_ip::add_ip_int_property "intParam1" "Integer Param 1" "3" -min "2" -max
  "32" -description "Integer param 1 description" -enabled "true"
29 ace_ip::set_ip_property_verilog_param_attributes "intParam1" -verilog_param
  "true" -to_upper "true" -num_bits 5
30
31 ace_ip::add_ip_int_property "intParam2" "Integer Param 2" "0" -min "0" -max
  "16" -description "Integer param 2 description" -enabled "true"
```



```

32 ace_ip::set_ip_property_verilog_param_attributes "intParam2" -verilog_param
   "true" -to_upper "true" -num_bits 4
33
34 ace_ip::add_ip_float_property "floatParam1" "Float Param 1" "3.1" -min "2.7"
   -max "32.4" -description "Float param description" -enabled "true"
35 ace_ip::set_ip_property_verilog_param_attributes "floatParam1" -verilog_param
   "true" -to_upper "true" -num_bits 6
36
37 ace_ip::add_ip_bool_property "boolParam1" "Bool Param 1" "Yes" -description
   "Bool param 1 description" -enabled "true"
38 ace_ip::set_ip_property_verilog_param_attributes "boolParam1" -verilog_param
   "true" -to_upper "true" -num_bits 1
39
40 ace_ip::add_ip_bool_property "boolParam2" "Bool Param 2" "Yes" -description
   "Bool param 2 description" -enabled "true"
41 ace_ip::set_ip_property_verilog_param_attributes "boolParam2" -verilog_param
   "true" -to_upper "true" -num_bits 1
42
43 ace_ip::add_ip_label "label1" "Ip Label 1" "my label" -description "Ip Label
   1 description" -enabled "false"
44
45 ace_ip::add_ip_file_path_property "filepathParam1" "Filepath Param 1" ""
   -description "Filepath Param 1 description" -enabled "true" -file_must_exist
   "false" -is_directory "false" -allow_blank_file "true"
46 ace_ip::add_ip_file_path_property "filepathParam2" "Filepath Param 2" ""
   -description "Filepath Param 2 description" -enabled "true" -file_must_exist
   "false" -is_directory "false" -allow_blank_file "true"
47
48 ace_ip::add_ip_signal_name_property "signalNameParam1" "Signal Name Param 1"
   "SignalName1" -description "Signal Name Param 1 description" -enabled "false"

```

IP GUI Layout definition

IP GUI Layout Definition

```

1 #
   #####
   #####
2 # Define the GUI layout here
3 # First define the page structure
4 ace_ip::add_ip_gui_page "Overview" "" "Long description text for overview
   page banner"
5 ace_ip::add_ip_gui_page "Sub-Page A" "Overview" "Long description text for
   page A banner"
6 ace_ip::add_ip_gui_page "Sub-Page B" "Overview" "Long description text for
   page B banner"

```

```
7 ace_ip::add_ip_gui_page "Sub-Page C" "Sub-Page B" "Long description text for
page C banner"
8
9
10 # Add groups, IP property controls, and Calculated value controls to each
page
11 # Control type can be:
12 # List: combo
13 # Int: combo, text
14 # Float: text
15 # Bool: checkbox
16 # Hex, Binary, Signal: text
17 # Filepath: file
18 # label (calculated value): text
19
20 # Overview page:
21 # add_ip_gui_control <property_name> <page> [-parent <parent>] [-control_type
<type>] [-span <column_span>] [-ip_name <ip_name>]
22 ace_ip::add_ip_gui_control "enumParam1" "Overview" -control_type "combo"
23 ace_ip::add_ip_gui_control "floatParam1" "Overview" -control_type "text"
24 ace_ip::add_ip_gui_control "signalNameParam1" "Overview" -control_type "text"
25
26 # add_ip_gui_group <name> <label> <page> [-parent <parent_group>] [-columns
<layout_columns>] [-span <column_span>] [-ip_name <ip_name>]
27 ace_ip::add_ip_gui_group "int_group" "Int Group" "Overview" -columns 3 -span
3
28 ace_ip::add_ip_gui_control "intParam1" "Overview" -control_type "text"
-parent "int_group"
29 ace_ip::add_ip_gui_control "intParam2" "Overview" -control_type "combo"
-parent "int_group"
30 ace_ip::add_ip_gui_control "label1" "Overview" -control_type "text" -parent
"int_group"
31
32 ace_ip::add_ip_gui_group "filepath_group" "Filepath Group" "Overview"
-columns 3 -span 3
33 ace_ip::add_ip_gui_control "filepathParam1" "Overview" -control_type "file"
-parent "filepath_group"
34 ace_ip::add_ip_gui_control "filepathParam2" "Overview" -control_type "file"
-parent "filepath_group"
35
36 # Sub-Page A:
37 ace_ip::add_ip_gui_control "enumParam2" "Sub-Page A" -control_type "text"
38 ace_ip::add_ip_gui_group "group_1" "Group 1" "Sub-Page A" -columns 3 -span 3
39 ace_ip::add_ip_gui_control "enumParam4" "Sub-Page A" -control_type "combo"
-parent "group_1"
40 ace_ip::add_ip_gui_control "binaryParam1" "Sub-Page A" -control_type "text"
-parent "group_1"
41
42 # Sub-Page B:
43 ace_ip::add_ip_gui_control "enumParam3" "Sub-Page B" -control_type "combo"
```

```

44 ace_ip::add_ip_gui_group "group_1" "Group 1" "Sub-Page B" -columns 3 -span 3
45 ace_ip::add_ip_gui_control "enumParam5" "Sub-Page B" -control_type "text"
   -parent "group_1"
46 ace_ip::add_ip_gui_control "hexParam1" "Sub-Page B" -control_type "text"
   -parent "group_1"
47
48 # Sub-Page C:
49 ace_ip::add_ip_gui_group "bool_group" "Bool Group" "Sub-Page C" -columns 3
   -span 3
50 ace_ip::add_ip_gui_control "boolParam1" "Sub-Page C" -control_type "checkbox"
   -parent "bool_group"
51 ace_ip::add_ip_gui_control "boolParam2" "Sub-Page C" -control_type "checkbox"
   -parent "bool_group"

```

test_ip_1_1.acxip

Speedster7t TEST IP 1

Overview
Long description text for overview page banner

✓ Target Device

✓ Enum Param 1

✓ Float Param 1

✓ Signal Name Param 1

Int Group

✓ Integer Param 1

✓ Integer Param 2

Ip Label 1

Filepath Group

✓ Filepath Param 1

✓ Filepath Param 2

Figure 308 • Overview Page in ACE

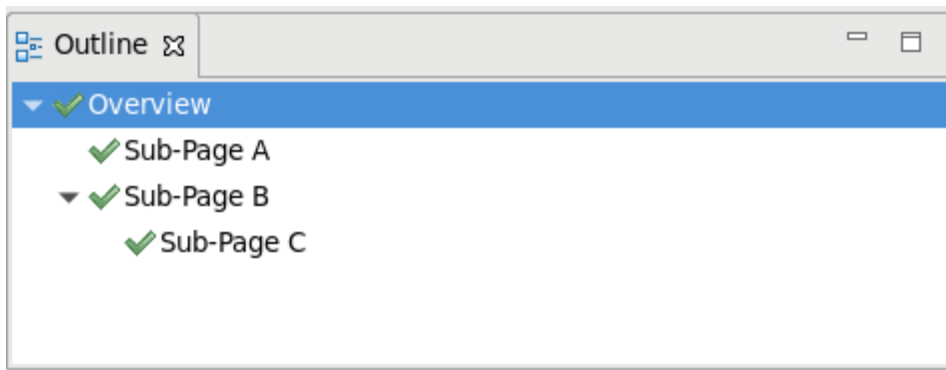


Figure 309 - Page Outline in GUI

Optional Setup for Built-in Output File Generator Functions

Setup for built-in Verilog generator functions

```

1  # This is optional, and is used by any built-in output file generator TCL
   procs
2  ace_ip::set_ip_macro_name "ACX_EXAMPLE_SOFT_IP"
3
4
5  # This is optional, and is used by any built-in output file generator TCL
   procs
6  ace_ip::add_ip_include_path "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/rtl/
   ACX_EXAMPLE_SOFT_IP.sv"
7  ace_ip::add_ip_include_path "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/
   include/example_include_file.svh"

```

IP Callback Functions

IP Plugin framework uses callbacks for instance-level updates to properties based on conditions, add custom checks, define ports and also generate output files.

Update Callback

Each IP definition must contain one update callback function to define instance-level property updates, add custom checks and define ports. Properties can be enabled/disabled but cannot be added/removed at the instance-level dynamically.

⚠ Caution!

Every IP project (ACXIP file) created by user in ACE is an instance-level IP which has a unique ACXIP file path.

⚠ Caution!

Properties that are added/removed inside update callback are dynamic properties. GUI support for dynamic properties is not yet added to the current release of ACE.

Update Callback

```

1  proc ip1_update_ip_configuration_proc {} {
2      # Run rule checks
3      # Example:
4      # Rule check
5      if {[ace_ip::get_ip_property_value "intParam1"] <=
6  [ace_ip::get_ip_property_value "intParam2"]} {
7          ace_ip::add_ip_rule_message "intParam1" "Rule check example" "Error"
8          "Integer Param 1 must be greater than Integer Param 2. Please fix this."
9      }
10
11     set binaryParam1 "00010101010101010101010101010101"
12     set enumParam4 [ace_ip::get_ip_property_value "enumParam4"]
13
14     switch $enumParam4 {
15         "0" {
16             set binaryParam1 "00000000000000000000000000000000"
17         }
18         "1/2" {
19             set binaryParam1 "00010101010101010101010101010101"
20         }
21         "1/3" {
22             set binaryParam1 "00001001001001001001001001001001"
23         }
24         "1/4" {
25             set binaryParam1 "010000001000100010001000100010001"
26         }
27         "1/5" {
28             set binaryParam1 "00000010000100001000010000100001"
29         }
30         "1/6" {
31             set binaryParam1 "00000001000001000001000001000001"
32         }
33         "1/7" {
34             set binaryParam1 "01000000001000000100000010000001"

```

```
34     }
35     "1/8" {
36         set binaryParam1 "11000000000000010000000100000001"
37     }
38     "1/9" {
39         set binaryParam1 "10000000000001000000001000000001"
40     }
41     "1/10" {
42         set binaryParam1 "00000000000100000000010000000001"
43     }
44     "1" {
45         set binaryParam1 "00111111111111111111111111111111"
46     }
47     default {
48         set binaryParam1 "00010101010101010101010101010101"
49     }
50 }
51
52 ace_ip::set_ip_property_value "binaryParam1" $binaryParam1
53 if { [ace_ip::get_ip_property_value "enumParam4"] == "CUSTOM" } {
54     ace_ip::update_ip_binary_property "binaryParam1" -enabled "true"
55 } else {
56     ace_ip::update_ip_binary_property "binaryParam1" -enabled "false"
57 }
58
59 set hexParam1 "00010101010101010101010101010101"
60 set enumParam5 [ace_ip::get_ip_property_value "enumParam5"]
61 switch $enumParam5 {
62     "0" {
63         set hexParam1 "00000000"
64     }
65     "1/2" {
66         set hexParam1 "15555555"
67     }
68     "1/3" {
69         set hexParam1 "09249249"
70     }
71     "1/4" {
72         set hexParam1 "40888891"
73     }
74     "1/5" {
75         set hexParam1 "02108421"
76     }
77     "1/6" {
78         set hexParam1 "01041041"
79     }
80     "1/7" {
81         set hexParam1 "40204081"
82     }
83     "1/8" {
```

```

84         set hexParam1 "c0010101"
85     }
86     "1/9" {
87         set hexParam1 "80040201"
88     }
89     "1/10" {
90         set hexParam1 "00100401"
91     }
92     "1" {
93         set hexParam1 "3fffffff"
94     }
95     default {
96         set hexParam1 "15555555"
97     }
98 }
99 ace_ip::set_ip_property_value "hexParam1" $hexParam1
100 if {[ace_ip::get_ip_property_value "enumParam5"] == "CUSTOM"} {
101     ace_ip::update_ip_hex_property "hexParam1" -enabled "true"
102 } else {
103     ace_ip::update_ip_hex_property "hexParam1" -enabled "false"
104 }
105
106 set enumParam1 [ace_ip::get_ip_property_value "enumParam1"]
107 if {$enumParam1 <= 64} {
108     ace_ip::update_ip_enum_property "enumParam2" -items {"X" "1" "2" "3"
109 "4"}
110     ace_ip::update_ip_enum_property "enumParam3" -items {"X" "1" "2" "3"
111 "4"}
112 } else {
113     # do not forget the else conditions for updates as the GUI does not
114     # reset automatically when conditions are not met
115     ace_ip::update_ip_enum_property "enumParam2" -items {"X" "1" "2" "3"
116 "4" "5" "6" "7" "8"}
117     ace_ip::update_ip_enum_property "enumParam3" -items {"X" "1" "2" "3"
118 "4" "5" "6" "7" "8" "9" "10"}
119 }
120
121 # Update Calculated Values
122 set label1 [expr {$enumParam1 * [ace_ip::get_ip_property_value
123 "intParam1"] / 64}]
124 ace_ip::update_ip_label "label1" -value $label1
125
126 # Update resource counts
127 ace_ip::set_ip_resource_count "NAP_I" [expr {$enumParam1 / 64}]
128
129 # #
130 #####
131 #####
132 # # Define Ports for generated Verilog/VHDL and interactive diagram

```

```

126     ace_ip::add_ip_port "ref_clk" "Clocks and Resets" "in" 1 "West" "ref_clk
description Text for tooltip and documentation.."
127     ace_ip::add_ip_port "rstn" "Clocks and Resets" "in" 1 "West" "rstn
description Text for tooltip and documentation.."
128
129     ace_ip::add_ip_port "data_in" "Data Interface" "in" 32 "West" "data_in
description Text for tooltip and documentation.."
130     ace_ip::add_ip_port "data_out" "Data Interface" "out" 32 "West" "data_out
description Text for tooltip and documentation.."
131
132     set w 0
133     if {[ace_ip::get_ip_property_value "boolParam1"] == "No"} {
134         set w [expr {$enumParam1 / 32}]
135     }
136     if {$w > 0} {
137         ace_ip::add_ip_port "port_a" "Bidi Interface" "inout" $w "East"
"port_a description Text for tooltip and documentation.."
138         ace_ip::add_ip_port "port_b" "Bidi Interface" "inout" $w "East"
"port_b description Text for tooltip and documentation.."
139     }
140 }

```

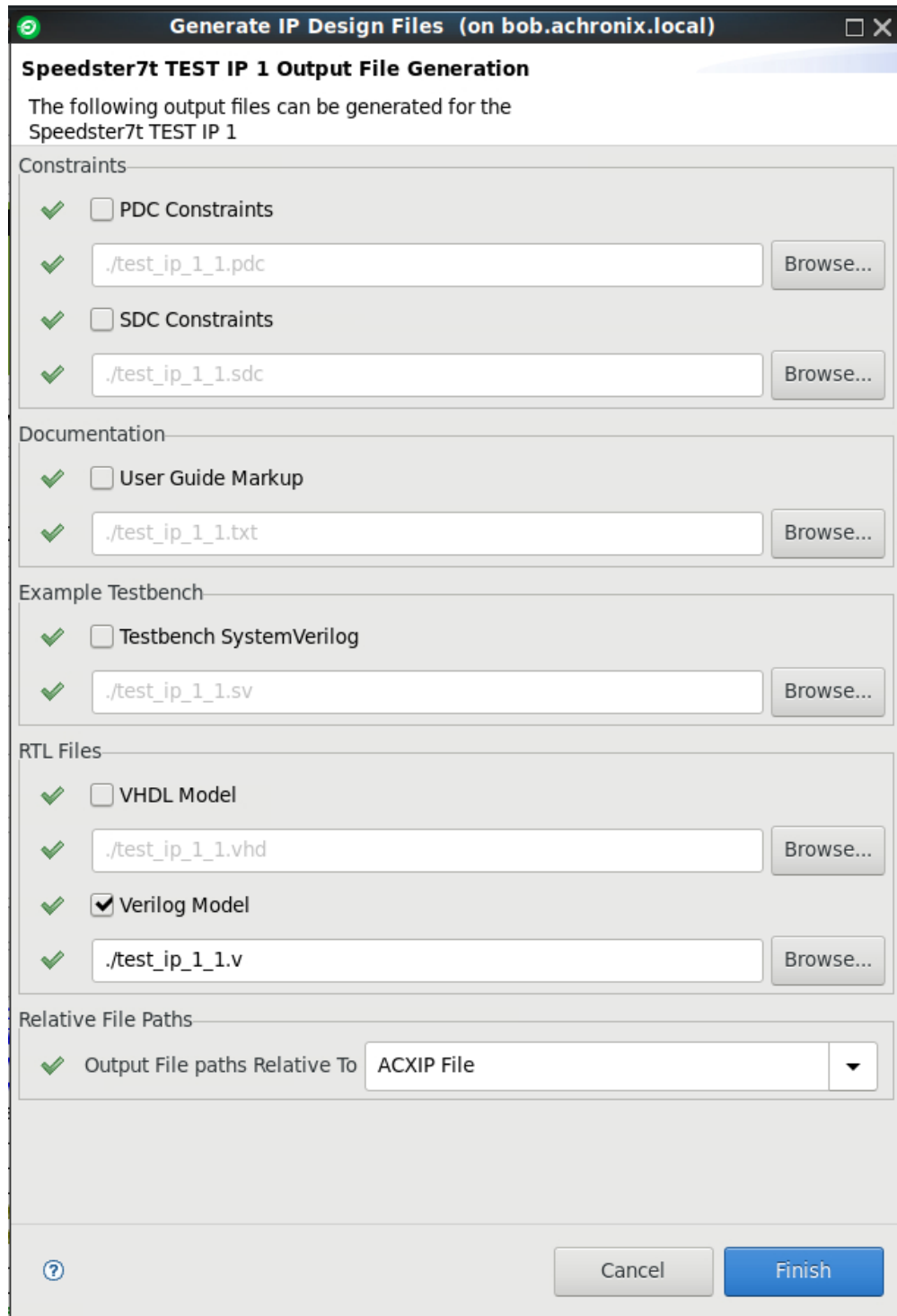
Output File Generation Callback

The output file generation callback can be used to add callback functions for output file generation. Each of the output file generator must contain TCL procedure that is used as a callback to generate output file. There are three arguments to this callback which ACE passes internally when running them during output file generation:

- `output_file_path`: Path to the output file which is specified by user during generation.
- `acxip_file_path`: Path to ACXIP file. This can be passed to the output file handler to be printed in the generated output file.
- `output_file`: Output file handler. All data passed to this handler is passed to the generated output file whose path is the `output_file_path`.

No properties can be added or updated in these callbacks. Output file generators can be grouped using an output file group.

The following wizard is generated for the output file options defined in example TCL in sections 3.5.2.1 and 3.5.2.2. The filenames are set by default and the path is relative to the ACXIP file path.



Built-in Output File Generation

The current framework supports built-in output file generation for Verilog and VHDL only. The built-in callback name must be specified as part of the output file. Both these built-in file generators are constructed using helper TCL functions.

⚠ Caution!

All these built-in file generators and helper functions is defined in a TCL package "**ipgen**". It is therefore important to require this package in plugin TCL files that need to use these functions.

Built-in output file generation

```

1  #
   #####
   #####
2  # Define supported output file types (Verilog, VHDL, SystemVerilog, SDC, PDC,
   Custom, etc)
3  ace_ip::add_ip_output_file_group "RTL Files" "RTL files output description"
4
5  # If there is a built-in output file generator already written in TCL, then
   set the callback to that built-in function, or write your own custom function
6  # These callbacks are called from inside generate_ip_design_files
7
8  # This command adds two properties i.e. output file enable property:
   "output.verilog_file_en" and output file path property: "output.verilog_file"
   to property definitions automatically
9  ace_ip::add_ip_output_file "Verilog Model"
   "ace_ip::builtin_generate_ip_verilog_file" "output.verilog_file_en"
   "output.verilog_file" "RTL Files" "Description text here..." -enabled "true"
   -file_ext "*.v"
10
11 ace_ip::add_ip_output_file "VHDL Model"
   "ace_ip::builtin_generate_ip_vhdl_file" "output.vhdl_f_en" "output.vhdl_f"
   "RTL Files" "Description text here..." -enabled "false" -file_ext "*.vhd"

```

i Built-in TCL helper functions

Verilog Built-in and helper functions

```
ace_ip::builtin_generate_ip_verilog_file {output_file_path acxip_file_path
output_file}
ace_ip::outputVerilogIpPorts {output_file}
ace_ip::outputVerilogMacroParameterList {output_file}

ace_ip::outputModuleDirectives {output_file}

ace_ip::writeVerilogParam {output_file val name label includeComma}

ace_ip::outputVerilogModulePortMapBlock {output_file moduleName}
```

VHDL Built-in and helper functions

```
ace_ip::builtin_generate_ip_vhdl_file {output_file_path acxip_file_path
output_file}
ace_ip::outputVhdlIpPorts {output_file}
ace_ip::outputVhdlPortMapBlock {output_file moduleName}
```

Miscellaneous

```
ace_ip::get_common_header_comment {output_file acxip_file_path file_type}
ace_ip::getFileNameWithoutExtension {output_file_path}
```

Test IP 1 Built-in Verilog output file generation

```
1 #####
2 # Generated Verilog file
3 # IP Generator Tool: TEST IP 1
4 # IP Generator Library: Speedster7t
5 # IP Generator Version: 1.0
6 # Source ACXIP File: /userbuilds3/allanlobo/main/proj_4/test_ip_1_1.acxip
7
8 `include "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/rtl/
ACX_EXAMPLE_SOFT_IP.sv"
9 `include "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/include/
example_include_file.svh"
10 `timescale 1 ps / 1 ps
11
12 module test_ip_1_1
13 (
```

```

14   input [31:0] data_in, //data_in description Text for tooltip and
documentation...
15   output [31:0] data_out, //data_out description Text for tooltip and
documentation...
16   input ref_clk, //ref_clk description Text for tooltip and documentation...
17   input rstn //rstn description Text for tooltip and documentation...
18 );
19 ACX_EXAMPLE_SOFT_IP #
20 (
21   .enumParam2 (4'hX), //Enum Param 2
22   .enumParam3 (4'hX), //Enum Param 3
23   .enumParam4 ("1/2"), //Enum Param 4
24   .INTPARAM1 (3), //Integer Param 1
25   .INTPARAM2 (0), //Integer Param 2
26   .FLOATPARAM1 (3.100000), //Float Param 1
27   .STRINGPARAM1 (my Default), //String Param 1
28   .BOOLPARAM1 (1'h1), //Bool Param 1
29   .binaryParam1 (32'b000101010101010101010101010101), //Binary Param 1
30   .BOOLPARAM2 (1'h1), //Bool Param 2
31 )
32 i_test_ip_1_1
33 (
34   .data_in (data_in), //data_in description Text for tooltip and
documentation...
35   .data_out (data_out), //data_out description Text for tooltip and
documentation...
36   .ref_clk (ref_clk), //ref_clk description Text for tooltip and
documentation...
37   .rstn (rstn) //rstn description Text for tooltip and documentation...
38 );
39
40 endmodule // test_ip_1_1

```

Test IP 1 Built-in VHDL output file generation

```

1   #####
2   # Generated VHDL file
3   # IP Generator Tool: TEST IP 1
4   # IP Generator Library: Speedster7t
5   # IP Generator Version: 1.0
6   # Source ACXIP File: /userbuilds3/allanlobo/main/proj_4/test_ip_1_1.acxip
7
8   library IEEE ;
9   use IEEE.STD_LOGIC_1164.all ;
10
11  entity acx_test_ip_1_1_wrapper is
12
13  port (

```

```

14   data_in : in std_logic_vector(31 downto 0); --data_in description Text for
      tooltip and documentation...
15   data_out : out std_logic_vector(31 downto 0); --data_out description Text
      for tooltip and documentation...
16   ref_clk : in std_logic; --ref_clk description Text for tooltip and
      documentation...
17   rstn : in std_logic --rstn description Text for tooltip and documentation...
18   );
19
20   end entity acx_test_ip_1_1_wrapper ;
21
22   architecture acx_test_ip_1_1_wrapper_arch of acx_test_ip_1_1_wrapper is
23
24     component test_ip_1_1 is
25
26     port (
27     data_in : in std_logic_vector(31 downto 0); --data_in description Text for
      tooltip and documentation...
28     data_out : out std_logic_vector(31 downto 0); --data_out description Text
      for tooltip and documentation...
29     ref_clk : in std_logic; --ref_clk description Text for tooltip and
      documentation...
30     rstn : in std_logic --rstn description Text for tooltip and documentation...
31     );
32
33     end component test_ip_1_1 ;
34
35     begin
36     test_ip_1_1_inst : test_ip_1_1
37     port map (
38     data_in => data_in,
39     data_out => data_out,
40     ref_clk => ref_clk,
41     rstn => rstn
42     );
43
44   end architecture acx_test_ip_1_1_wrapper_arch ;

```

Custom Output File Generation

Custom output file generation

```

1   ace_ip::add_ip_output_file_group "Constraints" "Some description text for
      tool-tip and docs"
2   ace_ip::add_ip_output_file "PDC Constraints"
      "ip1_custom_generate_ip_pdc_file" "output.pdc_file_en" "output.pdc_file"
      "Constraints" "Description text here..." -enabled "false" -file_ext "*.pdc"

```

```

3  proc ip1_custom_generate_ip_pdc_file {output_file_path acxip_file_path
   output_file} {
4      upvar $output_file out_file
5
6      set name [ace_ip::get_ip_property_value "name"]
7      set library [ace_ip::get_ip_property_value "library"]
8      set version [ace_ip::get_ip_property_value "acxip_version"]
9
10     puts $out_file "#####"
11     puts $out_file "# Generated PDC output file"
12     puts $out_file "# IP Generator Tool: $name"
13     puts $out_file "# IP Generator Library: $library"
14     puts $out_file "# IP Generator Version: $version"
15     puts $out_file "# Source ACXIP File: $acxip_file_path\n"
16 }
17
18 ace_ip::add_ip_output_file "SDC Constraints"
   "ip1_custom_generate_ip_sdc_file" "output.sdc_file_en" "output.sdc_file"
   "Constraints" "Description text here..." -enabled "false" -file_ext "*.sdc"
19 proc ip1_custom_generate_ip_sdc_file {output_file_path acxip_file_path
   output_file} {
20     upvar $output_file out_file
21
22     ace_ip::get_common_header_comment out_file $acxip_file_path "SDC output"
23 }
24
25 ace_ip::add_ip_output_file_group "Example Testbench" "Some description text
   for tooltip and docs"
26 ace_ip::add_ip_output_file "Testbench SystemVerilog"
   "ip1_custom_generate_ip_testbench_sv_file" "output.testbench_sv_file_en"
   "output.testbench_sv_file" "Example Testbench" "Description text here..."
   -enabled "false" -file_ext "*.sv"
27 proc ip1_custom_generate_ip_testbench_sv_file {output_file_path
   acxip_file_path output_file} {
28     upvar $output_file out_file
29
30     ace_ip::get_common_header_comment out_file $acxip_file_path "Testbench"
31 }
32
33 ace_ip::add_ip_output_file_group "Documentation" "Some description text for
   tooltip and docs"
34 ace_ip::add_ip_output_file "User Guide Markup"
   "ip1_custom_generate_ip_ug_docs_file" "output.ug_docs_file_en"
   "output.ug_docs_file" "Documentation" "Description text here..." -enabled
   "false" -file_ext "*.txt"
35 proc ip1_custom_generate_ip_ug_docs_file {output_file_path acxip_file_path
   output_file} {
36     upvar $output_file out_file
37
38     ace_ip::get_common_header_comment out_file $acxip_file_path "UG Doc"

```

```
39 | }
```

Add IP to GUI

Setup for built-in Verilog generator functions

```
1 | # Finally send this definition to the GUI  
2 | ace_ip::add_ip_to_gui "TEST IP 1"
```

Chapter 5 : Tcl Command Reference

The Tcl commands supported by ACE are broken into three subsets in this document:

- The **SDC Commands** (page 580), timing constraints which are also supported by upstream tools like Synplify, These commands are used in the SDC project constraints files.
- The **Interactive Timing Commands** (page 603), which are used to interact with the ACE STA timer. The commands are not constraints and can be used interactively in the ACE Tcl console.
- The **ACE Tcl Commands** (page 610), which are unique to ACE.

Additionally, ACE ships with a Tcl version 8.6 interpreter; the commands supported by Tcl itself are documented at: <https://www.tcl-lang.org/man/tcl8.6/TclCmd/contents.htm>

SDC Commands

The following are the Tcl commands which are used to define timing constraints in both ACE and upstream tools like Synplify.

all_clocks

```
all_clocks
```

Returns a collection of all clocks in the design.

Description

The `all_clocks` command will return a list of all of the clocks that have already been defined using either `create_clock` or `create_generated_clock`. It is often used in SDC files as an argument to commands that need a list of all of the clocks.

Example

To set all of the clocks to have the same setup timing uncertainty value, enter:

```
cmd> set_clock_uncertainty -setup .05 [all_clocks]
```

Also See

[create_clock](#) (page 582)

[create_generated_clock](#) (page 583)

[get_clocks](#) (page 586)

[set_clock_uncertainty](#) (page 592)

all_inputs

all_inputs

Returns a collection of all input ports (ports marked "in" and "inout") at the top level of the design.

Description

The `all_inputs` command returns a list of all of the input ports in the design. It is sometimes used as a command line argument to other SDC commands when a list of all of the input ports are needed.

Example

To set all of the inputs to have the same minimum input delay from the same clock, enter:

```
cmd> set_input_delay -min .01 [all_inputs] -clock clk
```

Also See

[set_input_delay \(page 598\)](#)

all_outputs

all_outputs

Returns a collection of all output ports (ports marked "out" and "inout") at the top level of the design.

Description

The `all_outputs` command returns a list of all of the output ports in the design. It is sometimes used as a command line argument to other SDC commands when a list of all of the output ports are needed.

Example

To set output delay constraint to all outputs with respect to the same clock, enter:

```
cmd> set_output_delay -min .01 [all_outputs] -clock clk
```

Also See

[set_output_delay \(page 602\)](#)

create_clock

```
create_clock [<clock>] [-period <string>] [-name <string>] [-waveform <list>]
```

Define a clock

Argument	Optional	Description
[<clock>]	Y	nets, ports or pins (as 'inst/pin')
[-period <string>]	Y	clock period in ns (required)
[-name <string>]	Y	alternate name
[-waveform <list>]	Y	list of edges for clock rise and fall timings in the period

Description

The `create_clock` command is the main SDC constraint input to static timing analysis. In its simplest form it can define a clock and its associated period. This definition is used by the timer to start timing paths. The timing paths will take one of four possible paths:

1. Traverse from a `create_clock` statement, through the clock IPIN, through the clock tree to a source DFF/LRAM/BRAM clock pin, through the source device, through the whatever logic is between the source, to the capture logic or `set_output_delay` (page 602) constraint.
2. Traverse from the `create_clock` statement, through a `set_input_delay` (page 598) constraint defined on a given port, through the path from that port to the DFF/LRAM/BRAM data input pin. These paths are often referred to as I/O timing paths, and specifically, input timing paths.
 - a. A `create_clock` statement can also be used strictly for IO timing, and not actually be placed and routed in the design. These are often referred to as "virtual clocks".
3. Sometimes a `create_clock` statement will be assigned to an input port that will traverse to a data input pin of a DFF. If this is done, the arrival time of the rising and falling edges will be separated in time by the definitions of the first asserted edge and deasserted edge of the clock.

Example

To define a clock on an input clock port and assign it a period of 2 ns, enter:

```
cmd> create_clock -period 2 [get_ports clock_in[0]]
```

To define a clock with a non-default (50/50) duty cycle, the `create_clock` `-waveform` option can be used:

```
set clock_period 10
set clock_asserted_edge 0
set clock_deasserted_edge [expr $clock_period / 5]
create_clock -name my_clock_name -period $clock_period -waveform
"$clock_asserted_edge $clock_deasserted_edge" [get_ports my_clock_port_name]
```

To define a virtual clock you must use the `-name` option so that the clock name can be referenced by other SDC commands. Otherwise, the `-name` option is optional.

```
create_clock -name virtual_my_clock -period 10
```

The `virtual_my_clock` can then be used in IO timing constraints to define the arrival time of data at input ports based on a clock that will not have any design specific latency:

```
set_input_delay 1 -clock virtual_my_clock [get_ports i_user_data*]
```

Also See

[create_generated_clock](#) (page 583)

[get_ports](#) (page 588)

[report_clock_properties](#) (page 608)

[report_clocks](#) (page 678)

[report_checks](#) (page 606)

[set_clock_latency](#) (page 591)

[set_clock_groups](#) (page 589)

[set_clock_uncertainty](#) (page 592)

[set_input_delay](#) (page 598)

[set_output_delay](#) (page 602)

create_generated_clock

```
create_generated_clock <clock> [-source <string>] [-divide_by <int>] [-multiply_by
<int>] [-name <string>]
```

Define a generated clock

Argument	Optional	Description
<clock>		nets or pins (as 'inst/pin')
[-source <string>]	Y	(required argument) master clock source pin
[-divide_by <int>]	Y	factor
[-multiply_by <int>]	Y	factor
[-name <string>]	Y	alternate name for the generated clock

Description

The `create_generated_clock` defines a clock which is applied to an output pin of an instance, internal to the design, or an output port of the design. This SDC command must follow a previously defined `create_clock` (page 582) definition, of which the port used to define that `create_clock` (page 582) statement would be used as the argument to the `-source` option. There must be a valid timing path between the source clock node and the generated clock node, so that latency between these two nodes can be calculated. A generated clock can have one of three characteristics of the source clock:

1. The same period as the source clock (`-divide_by 1`)
2. A period less than the source clock (`-divide_by` integer value greater than 1)
3. A period greater than the source clock (`-multiply_by` integer value greater than 1)

The generated clock will typically have a positive latency (delay) from the source clock as there is typically logic between the source clock and the generated clock. Therefore, the arrival time of a generated clock pin at a clock leaf node is calculated taking into account the latency from the source clock to the generated clock, plus the latency of the logic between the generated clock node and the generated clock leaf pin in the timing path. If frequency division is done (`-divide_by` or `-multiply_by`) than the deasserted edge and the second asserted edge of the generated clock's arrival times will be adjusted to the period calculated by the specified `-multiply_by` or `-divide_by` the respective values.

Example

To create a generated clock on an top level output port in a design, which is derived from a top level input port where the path is non-inverting and not divided, the following can be used:

```
create_generated_clock [get_ports o_user_clkout_001_003] -name out_clk
-divide_by 1 -source [get_ports i_user_clkkin_001_003]
```

Also See

[create_clock](#) (page 582)

[get_ports](#) (page 588)

[report_clock_properties](#) (page 608)

[report_clocks](#) (page 678)

[report_checks](#) (page 606)

[set_clock_groups](#) (page 589)

[set_clock_latency](#) (page 591)

[set_clock_uncertainty](#) (page 592)

get_cells

`get_cells pattern`

Returns a collection of cells (instances) in the design. All cell names match the specified pattern. Wildcards may be used to select multiple cells at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_cells` command can be use in conjunction with other SDC commands when those commands need a list of cell instance names as input. It accepts the use of the "*" wildcard will return a list of all cell instance names that match.

Example

To get all cell instances with the string "reg" in their name, enter;

```
get_cells *reg*
```

The output of the `get_cells` command can be passed to other commands, such as `set_multicycle_path`:

```
set_multicycle_path -from [get_cells top.*my_module.module_reg*] -setup -end 2
```

Also See

[set_multicycle_path](#) (page 600)

[set_false_path](#) (page 596)

get_clocks

```
get_clocks patterns [-nocase]
```

Returns a collection of clocks in the design. All clock names in the collection match the specified pattern. Wildcards may be used to select multiple clocks at once.

Argument	Optional	Description
patterns		The required <patterns> option is used to filter returned node names (string patterns are matched using Tcl string matching)
[-nocase]	Y	The optional -nocase option specifies the matching of node names to the patterns should be case-insensitive

Description

The `get_clocks` command can be used after the `create_clock` (page 582) command is used to define clocks. Additionally, if the `create_generated_clock` (page 583) command is used, the `get_clocks` command will include them as well. The `get_clocks` command is used to get a sub-set of what would be returned by the `all_clocks` (page 580) command. Typically, this command is used in conjunction with other SDC commands when a specific clock or specific group of clocks is needed as a command line argument.

Example

To get all of the clocks with the string "in" in their name, enter:

```
get_clocks *in*
```

To define clock to clock relationships, such as an asynchronous relationships between two clocks the following can be done:

```
set_clock_groups -asynchronous -group [get_clocks system_clock] -group [get_clocks test_clk]
```

Also See

[create_clock](#) (page 582)
[create_generated_clock](#) (page 583)
[report_clock_properties](#) (page 608)
[report_clocks](#) (page 678)
[set_clock_groups](#) (page 589)

[set_clock_latency](#) (page 591)

[set_clock_uncertainty](#) (page 592)

get_nets

`get_nets pattern`

Returns a collection of nets in the design. All net names in the collection match the specified pattern. Wildcards may be used to select multiple nets at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_nets` command can be use in conjunction with other SDC commands when those commands need a list of net names as input. It accepts the use of the "*" wildcard will return a list of all net names that match.

Example

To get all nets that have a name containing the string "data" and ending with "[0]", enter:

```
get_nets *data*[0]
```

To define a false path through a net the following command style can be used:

```
set_multicycle_path 2 -setup -through [get_nets *reset_sync_n] -to [get_clocks sys_clk]
```

Also See

[create_generated_clock](#) (page 583)

[get_clocks](#) (page 586)

[set_false_path](#) (page 596)

[set_multicycle_path](#) (page 600)

get_pins

`get_pins pattern`

Returns a collection of pins in the design. All pin names match the specified pattern. Wildcards may be used to select multiple pins at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_pins` command can be use in conjunction with other SDC commands when those commands need a list of cell instance pin names as input. It accepts the use of the "*" wildcard will return a list of all net names that match.

Example

In order to define a pin as an argument to another SDC command such as `create_generated_clock`, you can do the following, where the pin "clk_out" of the CLKDIV instance "sub-module top.first_sub_module.second_sub_module" is the location of the generated clock:

```
create_generated_clock -divide_by 2 [get_pins
top.first_sub_module.second_sub_module/clk_out]
```

Likewise, the same method can be used for any SDC command that takes a pin argument:

```
set_multicycle_path -from [get_pins top.first_sub_module/*reg*/q] -to [get_pins
top.second_sub_module/*reg*/d] -setup 4
```

Also See

[create_generated_clock](#) (page 583)

[set_false_path](#) (page 596)

[set_multicycle_path](#) (page 600)

get_ports

`get_ports` pattern

Returns a collection of ports (design inputs and outputs) in the design. All port names match the specified pattern. Wildcards may be used to select multiple ports at once.

Argument	Optional	Description
pattern		The required <pattern> option is used to filter returned node names (string pattern is matched using Tcl string matching)

Description

The `get_ports` command can be used in conjunction with other SDC commands when those commands need a list of top level port names as input. It accepts the use of the "*" wildcard which will return a list of all port names that match.

Example

To get all ports with the string "[0]" in their name, enter:

```
get_ports *[0]*
```

This command can also be used in an argument of other SDC commands that take a port as input. One of the most common is in the definition of a clock as it comes into the design:

```
create_clock -period 0.9 [get_ports {sys_clk}]
```

Also See

[create_clock](#) (page 582)

[create_generated_clock](#) (page 583)

[set_false_path](#) (page 596)

[set_input_delay](#) (page 598)

[set_multicycle_path](#) (page 600)

[set_output_delay](#) (page 602)

set_clock_groups

```
set_clock_groups [-name <string>] [-group <list>] [-asynchronous]
```

Define clock groups. With one `-group`, the clocks in that group have a `false_path` from/to all other clocks. With multiple `-group` options, the clocks in each group have a `false_path` from/to the clocks in the other groups. The groups have no meaning outside this command.

Argument	Optional	Description
<code>[-name <string>]</code>	Y	Name of clock group
<code>[-group <list>]</code>	Y	set of clocks
<code>[-asynchronous]</code>	Y	clocks are unrelated (default)

Description

The `set_clock_groups` command is defined in the SDC files after the `create_clock` (page 582) and `create_generated_clock` (page 583) statements have been defined. This command can be used to quickly define asynchronous relationships between clocks. This methodology replaced the older `set_false_path` (page 596) based STA/SDC description methodology and is more efficient to write the SDC as well as enabling the timer to be more efficient.

Example

To assume a design has clocks `system_clock` and `test_clk` are asynchronous to all other clocks, enter:

```
set_clock_groups -asynchronous -group [get_clocks system_clock] -group
[get_clocks test_clk]
```

This command specifies that A1 and B are unrelated to C. For instance, a path between A2 and C will not be timed. A path between A1 and A2, on the other hand, will be timed (unless there are other commands specifying a false path between them).

```
set_clock_groups -asynchronous -group [get_clocks {A B}] -group [get_clocks C]
```

Also See

[create_clock](#) (page 582)

[create_generated_clock](#) (page 583)

[get_clocks](#) (page 586)

[report_checks](#) (page 606)

[set_false_path](#) (page 596)

set_clock_latency

```
set_clock_latency delay port_pin_list [-clock <string>] [-rise] [-fall] [-min] [-max] [-late] [-early] [-source]
```

Set latency of clock network

Argument	Optional	Description
delay		delay_value
port_pin_list		port_pin_list (one or more ports)
[-clock <string>]	Y	clock list
[-rise]	Y	rise
[-fall]	Y	fall
[-min]	Y	min
[-max]	Y	max
[-late]	Y	late
[-early]	Y	early
[-source]	Y	source

Description

The `set_clock_latency` command is used to describe the arrival time of a clock at the top level port where it is defined using the `create_clock` (page 582) command. The off-design latency of the clock will modify the timing of the IO logic. The impact of this can be seen in `report_checks` (page 606) reports of IO timing where it will be reflected on both the reference clock path as well as the input data arrival times. It is common for there to be more than one `set_clock_latency` definitions for each clock in order to model the off design latency of the clock for both edges of the clock as well as the early and late arrival times of the clock. Care should be taken to ensure that early and late arrival times, which model the range of arrival times that can occur due to off design events such as crosstalk or varying paths to the clock input port, are not replicated (double counted) in the use of the `set_clock_uncertainty` (page 592) command.

Example

In order to define off chip clock latency, which will impact clock to IO and clock to other clock timing, use the following command, for "late" arriving clock edges at the port where the `acx_sc_i_user_clk_in_000_001[0]_1` clock is defined:

```
set_clock_latency -source -late -rise 0.169006 [get_clocks
acx_sc_i_user_clkkin_000_001[0]_1]
```

Also See

[create_clock](#) (page 582)

[get_clocks](#) (page 586)

[report_clock_properties](#) (page 608)

[report_clocks](#) (page 678)

[report_checks](#) (page 606)

[set_clock_uncertainty](#) (page 592)

set_clock_uncertainty

```
set_clock_uncertainty <uncertainty> [<objects>] [-from <string>] [-to <string>] [-
setup] [-hold]
```

Set uncertainty of clock network

Argument	Optional	Description
<uncertainty>		clock uncertainty in ns
[<objects>]	Y	one or more clocks, ports, or pins
[-from <string>]	Y	source clock
[-to <string>]	Y	destination clock
[-setup]	Y	uncertainty applies to setup check
[-hold]	Y	uncertainty applies to hold check

Description

Caution!

The `set_clock_uncertainty` command must be used with either the `<objects>` option or a pair of `-from` and `-to` options.

The `set_clock_uncertainty` command is generally used to model off design PLL jitter. This jitter is typically defined as "cycle to cycle" jitter, meaning that for one asserted edge to the next asserted edge there is some amount of +/- "uncertainty" in the arrival time of the second asserted edge. This uncertainty is used to shorten (-) the clock

insertion delay to a capture device in setup paths, and to increase the clock insertion delay to a capture device in hold timing paths. Since setup timing is typically measured from the first asserted edge to the next asserted edge, this PLL jitter based clock uncertainty is directly applicable. However, typically hold timing is done with respect to the same clock edge. Therefore, the `set_clock_uncertainty` command has both `-setup` and `-hold` options to enable the user to use different constraint values as the `-hold` value is not modeling PLL jitter, but instead can be used to add general timing guard band, which is typically referred to as modeling "known unknowns" as well as "unknown unknowns". The values of these constraints work in conjunction with the values defined in both the `create_clock` (page 582) definitions, as well as the `set_clock_latency` (page 591) values.

Care should be taken to ensure that uncertainty defined in those other constraint commands are not duplicated in the `set_clock_uncertainty` command. The effect that `set_clock_uncertainty` has on timing is global. All timing paths, both core and IO, will be impacted by this constraint and will be visible in the `report_checks` (page 606) reports in the capture or "reference" clock timing path on the "clock uncertainty" line. For setup paths, the value will be subtracted from the clock arrival time, and for hold timing the value will be added to the clock arrival time.

Example

To model PLL cycle to cycle jitter specification of 0.02nS for all of the clocks, the following command can be used:

```
set_clock_uncertainty -setup .02 [all_clocks]
```

To define extra hold timing guard band the following command can be used:

```
set_clock_uncertainty -hold .005 [all_clocks]
```

Also See

[all_clocks](#) (page 580)

[create_clock](#) (page 582)

[create_generated_clock](#) (page 583)

[report_clock_properties](#) (page 608)

[report_clocks](#) (page 678)

[report_checks](#) (page 606)

[set_clock_latency](#) (page 591)

set_data_check

```
set_data_check value [-clock <string>] [-setup] [-hold] [-from <string>] [-to <string>]
```

Set data-to-data check values of setup and hold

Argument	Optional	Description
value		check value
[-clock <string>]	Y	clock
[-setup]	Y	setup
[-hold]	Y	hold
[-from <string>]	Y	from_list (one or more clocks)
[-to <string>]	Y	to_list (one or more clocks)

Description

The `set_data_check` command can be used to add timing constraint between two data signals arriving to different pins/ports. This added timing constraint is analogous to the standard setup/hold timing constraints between a clock and data modeled for a DFF/LRAM/BRAM, but in this case the `-from` related pins is defined as the reference or clock pin, and the `-to` related pin is the data pin. The command supports unique `-setup` and `-hold` values, but if neither `-setup` nor `-hold` options are used, the values are applied only to setup. Often this command is used to define "data skew" which is typically defined as a +/- delta between data bus arrival times. Therefore, both the `-setup` and `-hold` options must be used, in different command instantiations, to define one data bit in a bus as the reference to N number of other data bits. Both the `-from` pin and the `-to` pin must be singular. For a bus that is 16 bits wide, there needs to be 15 constraints.

The `-from` and `-to` nodes defined in these commands must have existing valid timing paths to them for the `set_data_check` command to function. Therefore, if these constraints are applied to output ports, there must also be a `set_output_delay` (page 602) constraint applied to them. Additionally, if both `-setup` and `-hold` data checks are to be performed, there must be both `-min` and `-max` `set_output_delay` (page 602) constraints defined.

Example

In order to constrain the delay between two or more data pins, such as a "data skew" constraint, the following command can be used. This will define both a setup and hold timing relationship between the `reference_pin_name` and all of the the other `_pin_names`.

```
set_data_check -from [get_pins top.sub-module.instance1/reference_pin_name] -to
[get_pins top.sub-module.instance1/another_pin_name] .1
```

If more than one clock can drive a signal to the `-from` related pin, than the command can be made more specific by using the `-clock` options

```
set_data_check -from [get_pins top.sub-module.instance1/pin_name] -to [get_pins
top.sub-module.instance1/another_pin_name] .1 -clock [get_clocks my_clock]
```

To model a data skew constraint, a Tcl loop such as this can be used:

```
set first_port 0
set plus_minus_constraint 0.05
foreach port_name [get_ports dout_gpio[*]] {
    if { $first_port == 0 } {incr first_port;set reference_port
$port_name;continue}
    set_data_check -from $reference_port -to $port_name $plus_minus_constraint
    -hold
    set_data_check -from $reference_port -to $port_name $plus_minus_constraint
    -setup
}
```

Also See

[get_pins](#) (page 587)

[get_clocks](#) (page 586)

[report_checks](#) (page 606)

[set_output_delay](#) (page 602)

set_disable_timing

```
set_disable_timing <objects> [-from <string>] [-to <string>]
```

Disable timing arcs in a circuit

Argument	Optional	Description
<objects>		one or more instances, ports, or pins
[-from <string>]	Y	input pin name of instance <object>
[-to <string>]	Y	output pin name of instance <object>

Description

The `set_disable_timing` command is typically used to disable an existing timing arc. This is somewhat analogous to `set_false_path` (page 596), but with a much more limited scope. The `set_disable_timing` command requires a `-from` and a `-to` option to be used together, to bound the scope of its effect. Often, this

command is used to break timing loops from an input pin to an output pin of the same cell instance. The pin names used in the `-from` and `-to` options must be just the pin name, as found in the cell library.

Example

In order to break all of the timing arcs for an instance (`top.sub-module.instance1`), the following command can be used:

```
set_disable_timing [get_cells top.sub-module.instance1]
```

To break a given timing arc between two pins of a given cell instance, the following can be done:

```
set_disable_timing -from input_pin_name -to input_pin_name [get_cells top.sub-module.instance1]
```

Also See

[get_cells](#) (page 585)

[set_false_path](#) (page 596)

set_false_path

```
set_false_path [-from <list>] [-to <list>] [-through <list>]
```

Define a false path exception (this declares that the clocks are unrelated)

Argument	Optional	Description
<code>[-from <list>]</code>	Y	from_list (one or more clocks)
<code>[-to <list>]</code>	Y	to_list (one or more clocks)
<code>[-through <list>]</code>	Y	through_list (one or more clocks)

Description

The `set_false_path` command is used to create "timing exception" to the general STA paradigm that all timing paths are analyzed as a one cycle setup and zero cycle hold timing path. Another timing exception syntax is [set_multicycle_path](#) (page 600) and the user must be sure if a path is truly never used, or if it is a multicycle path. Timing paths that are analyzed by STA, but found to not be valid for whatever reason can be removed from the analysis by using the `set_false_path` statement. This command has several options, and can define very wide ranging paths so care should always be taken to limit the scope of these commands in order to ensure that only the paths known to be false are affected by these commands. To enable users to focus these statements on a finite

number of paths there is syntax to define the path start point (`-from`), path intermediate points (`-through`) as well as path end points (`-to`). It is advisable to be as explicit in the timing path definition as possible to ensure that real or valid timing paths are not being suppressed. Often the process of defining timing exceptions is like "peeling an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next-worst path becomes the new worst case path. Therefore, the most effective timing exceptions are typically constructed after all of the timing paths have been validated and all of the resulting exceptions combined to minimize the number of exceptions.

The definition of the path can be very narrow or very broad depending on how it is constructed. Each statement can contain only one `-from` and one `-to` statement, but each of them can reference many "from" nodes and many "to" nodes. If there are more than one node for these, all combinations apply. All "from" nodes are applied to all "to" nodes. Additionally, the `-through` option can have multiple nodes defined in it. Each of the `-through` nodes apply independently so if a path goes through any of the matching nodes it applies. However, multiple `-through` statements can be used in series with each other. They are order dependent, so `'-through a'` and `'-through b'` implies that the path must first go through "a" and then go through "b". If the `-through` command has multiple nodes in it, then that one statement is define as an 'OR'; `'-through a -through "b c" -through d'` implies that the path must go through "a" and then go through either "b" or "c", and then go through "d".

Example

In order to remove a timing path between an "instance/clock_pin_name" to an "instance/input_pin" from being analyzed by the timer, the following command can be used:

```
set_false_path -from [get_pins top.module1.reg_instance_name/clock_pin_name]
               -through [get_nets some_applicable_net] -to [get_pins
top.module2.instance_name/input_pin_name]
```

In order to remove all timing from a given timing node such as an input port, the following can be done:

```
set_false_path -from [get_ports my_port_which_I_do_not_want_to_time]
```

A false path can also be define `-through` a net, as well as instances and pins:

```
set_false_path -through [get_nets my_net_of_interest_name]
```

Also See

[get_pins](#) (page 587)

[get_ports](#) (page 588)

[get_nets](#) (page 587)

[set_multicycle_path](#) (page 600)

set_input_delay

```
set_input_delay delay port_pin_list [-clock <string>] [-rise] [-fall] [-max] [-min] [-add_delay] [-clock_fall]
```

Specify an input delay constraint or clock

Argument	Optional	Description
delay		delay_value
port_pin_list		port_pin_list (one or more ports)
[-clock <string>]	Y	clock_name
[-rise]	Y	rise
[-fall]	Y	fall
[-max]	Y	max
[-min]	Y	min
[-add_delay]	Y	add delay
[-clock_fall]	Y	delay with reference to falling edge of clock

Description

The `set_input_delay`, as well as the `set_output_delay` (page 602) command, is fundamental to validating the correctness of a design's timing. It is defined after the clocks are define and it references the related clock using the `-clock` option. Typically, there will be four (4) definitions of `set_input_delay` for each data/clock combination, at each timing corner. The arrival time of the data at its design input port is relative to this clock's arrival time. Therefore, the `set_clock_latency` (page 591) constraint will impact the arrival time of data related to that clock. The value of the `set_input_delay` constraint is used in the timing path which receives the data signal. This value can be seen in a `report_checks` (page 606) report in the data path section under "input external delay".

Example

In order to constrain a design's input port, an arrival time for a signal at the input port, relative to the asserted edge of a specified clock, for both min and max data path timing, as well as having different values for rise and fall edges, can be defined using this command:

```
set_input_delay .1 -rise -max -clock [get_clocks my_clock_name] [get_ports my_input_port_name]
set_input_delay .13 -fall -max -clock [get_clocks my_clock_name] [get_ports
```

```

my_input_port_name]
set_input_delay .05 -rise -min -clock [get_clocks my_clock_name] [get_ports
my_input_port_name]
set_input_delay .055 -fall -min -clock [get_clocks my_clock_name] [get_ports
my_input_port_name]

```

Also See

[create_clock](#) (page 582)

[get_clocks](#) (page 586)

[get_ports](#) (page 588)

[report_checks](#) (page 606)

[set_clock_latency](#) (page 591)

[set_output_delay](#) (page 602)

set_input_transition

```
set_input_transition slew port_pin_list [-clock <string>]
```

Specify an input slew/transition constraint

Argument	Optional	Description
slew		slew_value
port_pin_list		port_pin_list (one or more ports)
[-clock <string>]	Y	clock_name

set_load

```
set_load load port_pin_list
```

Specify an output load/capacitance constraint

Argument	Optional	Description
load		cap_value
port_pin_list		port_pin_list (one or more ports)

set_max_delay

```
set_max_delay delay [-from <list>] [-to <list>] [-through <list>]
```

Set a maximum delay for a path

Argument	Optional	Description
delay		delay
[-from <list>]	Y	from_list (one or more clocks)
[-to <list>]	Y	to_list (one or more clocks)
[-through <list>]	Y	through_list (one or more clocks)

set_min_delay

```
set_min_delay delay [-from <list>] [-to <list>] [-through <list>]
```

Set a minimum delay for a path

Argument	Optional	Description
delay		delay
[-from <list>]	Y	from_list (one or more clocks)
[-to <list>]	Y	to_list (one or more clocks)
[-through <list>]	Y	through_list (one or more clocks)

set_multicycle_path

```
set_multicycle_path multiplier [-setup] [-hold] [-start] [-end] [-from <list>] [-to <list>] [-through <list>]
```

Define multicycle path

Argument	Optional	Description
multiplier		integer path multiplier
[-setup]	Y	Use the specified path multiplier for setup/max delay calculations. This is the default behavior

Argument	Optional	Description
<code>[-hold]</code>	Y	Use the specified path multiplier for hold/min delay calculations
<code>[-start]</code>	Y	Use the start/source clock period in the calculations. This is the default behavior for hold checks
<code>[-end]</code>	Y	Use the end/target clock period in the calculations. This is the default behavior for setup checks
<code>[-from <list>]</code>	Y	list of path startpoints containing clock, primary input/inout port, sequential-cell instance, clock pin of sequential-cell instance, or pin with input delay constraint
<code>[-to <list>]</code>	Y	list of path endpoints containing clock, primary output/inout port, sequential-cell instance, data pin of sequential-cell instance, or pin with output delay constraint
<code>[-through <list>]</code>	Y	list of pins, ports, nets that the path must pass through

Description

The `set_multicycle_path` command is used to create "timing exception" to the general STA paradigm that all timing paths are analyzed as a one cycle setup and zero cycle hold timing path. Another timing exception syntax is `set_false_path` and the user must be sure if a path is truly used but in more than one cycle, or never used. Timing paths that are analyzed by STA, but found to not be only valid on more than one cycle, for whatever reason, can have its analysis adjusted by using the `set_multicycle_path` statement. This command has several options, and can define very wide ranging paths so care should always be taken to limit the scope of these commands in order to ensure that only the paths known to be false are effected by these commands. To enable users to focus these statements on a finite number of paths there is syntax to define the path start point (`-from`), path intermediate points (`-through`) as well as path end points (`-to`). It is advisable to be as explicit in the timing path definition as possible to ensure that real or valid timing paths are not being suppressed. Often the process of defining timing exceptions is like "peeling an onion". The timing typically only shows the "worst case path", so as you eliminate that path, the next-worst path becomes the new worst case path. Therefore, the most effective timing exceptions are typically constructed after all of the timing paths have been validated and all of the resulting exceptions combined to minimize the number of exceptions.

The definition of the path can be very narrow or very broad depending on how it is constructed. Each statement can contain only one `-from` and one `-to` statement, but each of them can reference many "from" nodes and many "to" nodes. If there are more than one node for these, all combinations apply. All "from" nodes are applied to all "to" nodes. Additionally, the `-through` option can have multiple nodes defined in it. Each of the `-through` nodes apply independently so if a path goes through any of the matching nodes it applies. However, multiple `-through` statement can be used in series with each other. They are order dependent, so `'-through a'` and `'-through b'` implies that the path must first go through "a" and then go through "b". If the `-through` command has multiple nodes in it, than that one statement is define as an 'OR'; `'-through a -through "b c" -through d'` implies that the path must go through "a" and then go through either "b" or "c", and then go through "d".

Example

To change the default STA one cycle timing paradigm for all paths between two DFFs, to being a two cycle path, the following can be done:

```
set_multicycle_path 2 -from [get_pins top.sub_module_name.register_name/ck] -to
[get_pins top.some_module_name.some_register_name/q] -setup
```

It is important to understand, that the changing of the default one cycle path for setup, usually requires a modification from the default zero (0) cycle hold timing constraint (but not always). In this case, the hold timing is changed to be a one (1) cycle hold check:

```
set_multicycle_path 1 -from [get_pins top.sub_module_name.register_name/ck] -to
[get_pins top.some_module_name.some_register_name/q] -hold
```

Also See

[get_pins](#) (page 587)

set_output_delay

```
set_output_delay delay port_pin_list [-clock <string>] [-rise] [-fall] [-max] [-min]
[-add_delay] [-clock_fall]
```

Specify an output delay constraint or clock

Argument	Optional	Description
delay		delay_value
port_pin_list		port_pin_list (one or more ports)
[-clock <string>]	Y	clock_name
[-rise]	Y	rise
[-fall]	Y	fall
[-max]	Y	max
[-min]	Y	min
[-add_delay]	Y	add delay

Argument	Optional	Description
<code>[-clock_fall]</code>	Y	delay with reference to falling edge of clock

Description

The `set_output_delay`, as well as the `set_input_delay` (page 598) command, is fundamental to validating the correctness of a design's timing. It is defined after the clocks are define and it references the related clock using the `-clock` option. Typically, there will be four (4) definitions of `set_output_delay` for each data/clock combination, at each timing corner. The required time of the data at its design output port is relative to this clock's arrival time. Therefore, the `set_clock_latency` (page 591) constraint will impact the required time of data related to that clock. The value of the `set_output_delay` constraint is used in the timing path which drives the output data signal. This value can be seen in a `report_checks` (page 606) report in the data path section under "output external delay".

Example

In order to constrain a design's output port, a required time for a signal at the output port, relative to the asserted edge of a specified clock, can be defined using this command:

```
set_output_delay .1 -rise -max -clock [get_clocks my_clock_name] [get_ports
my_output_port_name]
set_output_delay .13 -fall -max -clock [get_clocks my_clock_name] [get_ports
my_output_port_name]
set_output_delay .05 -rise -min -clock [get_clocks my_clock_name] [get_ports
my_output_port_name]
set_output_delay .055 -fall -min -clock [get_clocks my_clock_name] [get_ports
my_output_port_name]
```

Also See

[get_ports](#) (page 588)

[get_clocks](#) (page 586)

[report_checks](#) (page 606)

[set_clock_latency](#) (page 591)

[set_output_delay](#) (page 602)

Interactive Timing Commands

These commands are used to query the ACE Static Timing Analyzer (STA) interactively, from the ACE command prompt. To use these commands, interactive timer mode must be enabled by calling `prepare_sta` (page 605). To exit

interactive timer mode, call [reset_sta](#) (page 609). While in interactive timer mode, regular ACE commands remain available, but the placement and routing of the design should not be changed.

check_setup

The `check_setup` command performs sanity checks on the design. Individual checks can be performed with the keywords. If no check keywords are specified all checks are performed.

Command Syntax

```
check_setup [-verbose] [-unconstrained_endpoints] [-multiple_clock] [-no_clock] [-no_input_delay] [-no_output_delay] [-loops] [-generated_clocks]
```

Table 149 • Command-line Options for check_setup

Argument	Optional	Description
-verbose	✓	Show offending objects rather than just error counts.
-unconstrained_endpoints	✓	Check path endpoints for timing constraints (timing check or set_output_delay).
-multiple_clock	✓	Check register/latch clock pins for multiple clocks.
-no_clock	✓	Check register/latch clock pins for a clock.
-no_input_delay	✓	Check for inputs that do not have a set_input_delay command.
-no_output_delay	✓	Check for outputs that do not have a set_output_delay command.
-loops	✓	Check for combinational logic loops.
-generated_clocks	✓	Check that generated clock source pins have been defined as clocks.

Argument	Optional	Description
> filename	✓	Write output to file.
>> filename	✓	Append output to file.

Example

To check the effectiveness of the timing constraint to fully constrain all of the design's timing endpoints, the `check_setup` command can be run after `[run_prepare]`:

The following generates standard output summarizing any checks that violate:

```
check_setup
```

The following reports only a summary of the "unconstrained endpoints":

```
check_setup -unconstrained_endpoints
```

The following will create an "check_setup.rpt" file, and will indicate all of the information available for each violations:

```
check_setup -unconstrained_endpoints -verbose > check_setup.rpt
check_setup -multiple_clock -verbose >> check_setup.rpt
check_setup -no_clock -verbose >> check_setup.rpt
check_setup -no_input_delay -verbose >> check_setup.rpt
check_setup -no_output_delay -verbose >> check_setup.rpt
check_setup -loops -verbose >> check_setup.rpt
check_setup -generated_clocks -verbose >> check_setup.rpt
```

prepare_sta

The `prepare_sta` command prepares the ACE Static Timing Analyzer (STA) for interactive use. This step is required before other interactive timer commands can be used. Typically, `prepare_sta` is only used after `place` and `route`. If

the netlist, or the placement or routing, is modified, run this command again before interactive timing commands are used.

Command Syntax

```
prepare_sta (-slowc | -fastc) [-unconstrained]
```

Table 150 - Command-line Options for *prepare_sta*

Argument	Optional	Description
-slowc		Use delays for the slow timing corner. This option is often used for verifying setup time requirements. ⁽¹⁾
-fastc		Use delays for the fast timing corner. This option is often used for verifying hold time requirements. ⁽¹⁾
-unconstrained	✓	Enable reporting of unconstrained paths.

Table Notes

- Exactly one timing corner must be specified.

Example

To change the Tcl interface to be in STA interactive mode to analyze the slow timing arcs, enter the following from the ACE Tcl window, while in the ACE Tcl interface mode:

```
prepare_sta -slowc
```

When ACE is in STA "slowc" mode, to look at the fast timing arcs, enter the following:

```
reset_sta
prepare_sta -fastc
```

report_checks

The `report_checks` command reports the timing results for paths in the design.

Command Syntax

```
report_checks [-from <list>] [-to <list>] [-rise_to <list>] [-fall_to <list>] [-
path_delay <min,max>] [-group_count <int>] [-endpoint_count <int>] [-through <list>] [-
rise_through <list>] [-fall_through <list>] [-slack_max <float>] [-slack_min <float>] [-
sort_by_slack] [-path_group <list>] [-format <end,full,short,summary>] [-fields <list:
input_pins,nets>] [-digits <int>] [-no_line_split]
```

Table 151 • Command Line Options for report_checks

Argument	Optional	Description
-from <list>	✓	Report only paths starting at the specified objects: clocks, instances, ports, or pins.
-to <list>	✓	Report only paths ending at the specified objects: clocks, instances, ports, or pins.
-rise_to <list>	✓	Report only paths ending rising edge at the specified objects: clocks, instances, ports, or pins.
-fall_to <list>	✓	Report only paths ending falling edge at the specified objects: clocks, instances, ports, or pins.
-path_delay <min,max>	✓	The type of timing analysis. Currently only <i>max</i> (for setup analysis) and <i>min</i> (for hold time analysis) are supported. The default is <i>max</i> .
-group_count <int>	✓	The maximum number of paths to report, per clock group.
-endpoint_count <int>	✓	The number of paths to report per endpoint (default 1).
-through <list>	✓	Report only paths through the specified objects: instances, pins, or nets.
-rise_through <list>	✓	Report only paths through a rising edge at the specified objects: instances, pins, or nets.
-fall_through <list>	✓	Report only paths through a falling edge at the specified objects: instances, pins, or nets.
-slack_max <float>	✓	Report only paths with slack less than this number.

Argument	Optional	Description
-slack_min <float>	✓	Report only paths with slack larger than this number.
-sort_by_slack	✓	Sort paths by slack across all path groups rather than by slack grouped for each path group (default).
-path_group <list>	✓	Report only paths in these groups.
-format <type>	✓	Specifies which format to report for each path. [end, full, short, summary].
-fields <list>	✓	Report extra fields to the path report: List of input_pins nets.
-digits <int>	✓	Number of digits to print after the decimal point.
-no_line_split	✓	Do not break long lines.
-unconstrained	✓	Report unconstrained paths.
> filename	✓	Write output to file.
>> filename	✓	Append output to file.

report_clock_properties

The report_clock_properties command reports the clock defined for the timer in the design.

Command Syntax

```
report_clock_properties
```

Table 152 • Command-line Options for report_clock_properties

Argument	Optional	Description
> filename	✓	Write output to file.
>> filename	✓	Append output to file.

Example

```
cmd> report_clock_properties
```

```
Clock  Period Waveform
```

```
-----
clk[0] 2.500 0.000 1.250
clk[1] 2.500 0.000 1.250
clk[2] 2.500 0.000 1.250
clk[3] 2.500 0.000 1.250
clk[4] 2.500 0.000 1.250
clk[5] 2.500 0.000 1.250
clk[6] 2.500 0.000 1.250
clk[7] 2.500 0.000 1.250
clk[8] 2.500 0.000 1.250
clk[9] 2.500 0.000 1.250
```

reset_sta

The `reset_sta` command exits interactive timer mode. This command should be used before changing placement or routing, otherwise the STA might use stale data.

Command Syntax

```
reset_sta
```

Example

When in ACE interactive timing mode (`slowc` or `fastc`), to run non-timing related commands, enter the following:

```
cmd> reset_sta
```

To switch from interactive timing mode (`slowc`) to (`fastc`), enter the following:

```
reset_sta
prepare_sta -fastc
```

ACE Tcl Commands

The following commands are used only within ACE. These are not available within Synplify, etc.

add_clock_preroute

```
add_clock_preroute <net_name> <track_list> [-clock_regions <list>] [-clusters <list>] [-placement_regions <list>] [-partitions <list>] [-data_region]
```

This command will take a clock or reset net, and constrain it to be routed it over the clock tracks specified into the clock regions, fabric clusters, partition bounding boxes, or placement regions depending on what the user specifies.

Argument	Optional	Description
<net_name>		The name of the clock or reset net to be pre-routed.
<track_list>		The list of integer clock track numbers to pre-route this net on. Valid clock track numbers are device-specific.
[-clock_regions <list>]	Y	The list of clock region names to pre-route this net into. Valid clock region names are device-specific.
[-clusters <list>]	Y	The list of cluster names to pre-route this net into. Valid cluster names are device-specific.
[-placement_regions <list>]	Y	The list of placement region names to pre-route this net into. Valid placement region names are device-specific.
[-partitions <list>]	Y	The list of partition names to pre-route this net into. Valid partition are device-specific.
[-data_region]	Y	The -data_region option is only valid for non-clock nets. You can optionally use -data_region to route the net using region resources. "data_center" is applied by default if this option isn't used.

add_project_constraints

```
add_project_constraints <file> [-project <string>] [-corner <string>] [-temperature <string>] [-voltage <string>]
```

This command adds a link to an SDC, PDC, or TCL constraint file to a project.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the SDC, PDC, or TCL constraint file.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the SDC constraint file to be added to.
[-corner <string>]	Y	The optional -corner <corner> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given process corner. Valid values are "fast" and "slow"
[-temperature <string>]	Y	The optional -temperature <temp> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given temperature corner. Valid values are device-specific and must match a value from the junction_temperature impl option list.
[-voltage <string>]	Y	The optional -voltage <v> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given core voltage corner. Valid values are device-specific and must match a value from the core_voltage impl option list.

After a project has been created, you can point to a constraint file (SDC or PDC) using the following command. In this example, there is an existing file located at `../constraints/top.sdc`:

```
add_project_constraints -project [get_active_project] "../constraints/top.sdc"
```

Also See

[create_project](#) (page 623)

[get_active_project](#) (page 643)

add_project_ip

```
add_project_ip <list_of_files> [-project <string>]
```

This command associates one or more IP settings files with a project.

Argument	Optional	Description
<list_of_files>		The required <list_of_files> argument is used to specify the file paths to the IP settings files. The file paths may be absolute, or may be relative to the acxprj file's directory.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be added to. The named project must already be opened in ACE.

add_project_netlist

```
add_project_netlist <file> [-project <string>] [-impl <string>]
```

This command adds a link to a verilog netlist file to a project.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the verilog netlist.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be added to.
[-impl <string>]	Y	The optional -impl <implName> options is used to select an alternate netlist for a different impl, i.e. due to varied synthesis tools or options.

add_project_source_files

```
add_project_source_files <list_of_files> [-project <string>] [-ip] [-rtl] [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-sim_tb] [-impl <string>] [-corner <string>] [-temperature <string>] [-voltage <string>]
```

This command adds a link to one or more source files to a project.

Argument	Optional	Description
<list_of_files>		The required <list_of_files> argument is used to specify the file paths to source files.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the source file to be added to.

Argument	Optional	Description
<code>[-ip]</code>	Y	The optional <code>-ip</code> option is used to specify the source file to be added is an IP settings file.
<code>[-rtl]</code>	Y	The optional <code>-rtl</code> option is used to specify the source file to be added is an RTL file.
<code>[-syn_constraint]</code>	Y	The optional <code>-syn_constraint</code> option is used to specify the source file to be added is a synthesis constraint file.
<code>[-pnr_constraint]</code>	Y	The optional <code>-pnr_constraint</code> option is used to specify the source file to be added is a place and route constraint file.
<code>[-pnr_netlist]</code>	Y	The optional <code>-pnr_netlist</code> option is used to specify the source file to be added is a place and route netlist file.
<code>[-sim_tb]</code>	Y	The optional <code>-sim_tb</code> option is used to specify the source file to be added is a simulation testbench file.
<code>[-impl <string>]</code>	Y	The optional <code>-impl <implName></code> option is used to select an alternate netlist for a different impl, i.e. due to varied synthesis tools or options. This option can only be used with <code>"-pnr_netlist"</code>
<code>[-corner <string>]</code>	Y	The optional <code>-corner <corner></code> option is used to mark a place and route SDC constraints file (containing only delays) as being applicable only to the given process corner. Valid values are "fast" and "slow"
<code>[-temperature <string>]</code>	Y	The optional <code>-temperature <temp></code> option is used to mark a place and route SDC constraints file (containing only delays) as being applicable only to the given temperature corner. Valid values are device-specific and must match a value from the <code>junction_temperature</code> impl option list.
<code>[-voltage <string>]</code>	Y	The optional <code>-voltage <v></code> option is used to mark a place and route SDC constraints file (containing only delays) as being applicable only to the given core voltage corner. Valid values are device-specific and must match a value from the <code>core_voltage</code> impl option list.

add_region_find_insts

```
add_region_find_insts <region> <find_command> [-flops_only] [-clocks_only] [-include_constants] [-batch] [-verbose]
```

Add user design instances to a placement region constraint using a find command

Argument	Optional	Description
<region>		Name of the region
<find_command>		Find command used to get list of user design instances
[-flops_only]	Y	When adding instances, filter out all instances except flops
[-clocks_only]	Y	When adding instances, filter out all instances with no connected clock
[-include_constants]	Y	When adding instances, do not filter out power/ground constants
[-batch]	Y	Postpone application of this constraint until apply_placement is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.
[-verbose]	Y	Print additional command status messages.

add_region_insts

```
add_region_insts <region> <insts> [-flops_only] [-clocks_only] [-include_constants] [-batch] [-verbose]
```

Add user design instances to a placement region constraint

Argument	Optional	Description
<region>		Name of the region
<insts>		List of user design instances
[-flops_only]	Y	When adding instances, filter out all instances except flops
[-clocks_only]	Y	When adding instances, filter out all instances with no connected clock
[-include_constants]	Y	When adding instances, do not filter out power/ground constants
[-batch]	Y	Postpone application of this constraint until apply_placement is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.

Argument	Optional	Description
[-verbose]	Y	Print additional command status messages.

apply_highlights

apply_highlights [-insts] [-nets] [-paths]

This command updates the GUI with highlighting information on the present design.

Argument	Optional	Description
[-insts]	Y	Update highlighting of all instances in the design
[-nets]	Y	Update highlighting of all nets in the design
[-paths]	Y	Update highlighting of all paths in the design

apply_placement

apply_placement [-batch] [-defparams] [-partition] [-nocomplain]

Apply batched pre-placement commands

Argument	Optional	Description
[-batch]	Y	Specifies that placement should be applied from batched placement commands
[-defparams]	Y	Specifies that placement should be applied from defparams
[-partition]	Y	Specifies that placement should be applied on partition anchor instances before calling move_partition
[-nocomplain]	Y	Batched placement that does not yet apply is kept in the batch list and ignored

check_project_status

check_project_status [-rtl_sim] [-syn] [-prepare]

This command checks if any project source files have changed since running the prepare flow step on the active implementation. If the source files are consistent, no message is printed. Otherwise, warnings are printed for each out of sync file.

Argument	Optional	Description
<code>[-rtl_sim]</code>	Y	Check cached source files during rtl simulation flowstep.
<code>[-syn]</code>	Y	Check cached source files during synthesis flowstep.
<code>[-prepare]</code>	Y	Check cached source files during prepare flowstep.

clean_project

```
clean_project [-project <string>] [-impl_names <list>]
```

This command deletes output files from multiple implementations on the file system. The implementations' output directories on the file system are not deleted.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) from which the implementations' output files will be removed. If no <code>projectName</code> is specified, the active project will be used by default.
<code>[-impl_names <list>]</code>	Y	The optional <code>-impl_names <list></code> argument is used to specify the names of the implementations to remove. If no list is specified, the output files for all implementations under the project will be deleted.

clear_arcs

```
clear_arcs [-id <int>]
```

This command allows you to clear a custom arc or all the arcs on the GUI's Floorplanner view.

Argument	Optional	Description
<code>[-id <int>]</code>	Y	The optional <code>-id <id></code> option specifies a unique id for a single arc to clear. If this option is not used, all arcs will be cleared.

clear_drawing

```
clear_drawing
```

This command clears the current custom drawing on the GUI's Floorplanner view.

clear_flow

```
clear_flow
```

This command clears user design DB and the completion status of all flow steps.

clear_lines

```
clear_lines [-id <int>]
```

This command allows you to clear a custom line or all the lines on the GUI's Floorplanner view.

Argument	Optional	Description
[-id <int>]	Y	The optional -id <id> option specifies a unique id for a single line to clear. If this option is not used, all lines will be cleared.

clear_ovals

```
clear_ovals [-id <int>]
```

This command allows you to clear a custom oval or all the ovals on the GUI's Floorplanner view.

Argument	Optional	Description
[-id <int>]	Y	The optional -id <id> option specifies a unique id for a single oval to clear. If this option is not used, all ovals will be cleared.

clear_polygons

```
clear_polygons [-id <int>]
```

This command allows you to clear a custom polygon or all the polygons on the GUI's Floorplanner view.

Argument	Optional	Description
[-id <int>]	Y	The optional -id <id> option specifies a unique id for a single polygon to clear. If this option is not used, all polygons will be cleared.

clear_rectangles

```
clear_rectangles [-id <int>]
```

This command allows you to clear a custom rectangle or all the rectangles on the GUI's Floorplanner view.

Argument	Optional	Description
<code>[-id <int>]</code>	Y	The optional <code>-id <id></code> option specifies a unique id for a single rectangle to clear. If this option is not used, all rectangles will be cleared.

clear_strings

```
clear_strings [-id <int>]
```

This command allows you to clear a custom string or all the strings on the GUI's Floorplanner view.

Argument	Optional	Description
<code>[-id <int>]</code>	Y	The optional <code>-id <id></code> option specifies a unique id for a single string to clear. If this option is not used, all strings will be cleared.

clock_info

```
clock_info [-domain <string>] [-pin <string>] [-net <string>] [-all] [-unique] [-multi] [-freq] [-period] [-phase] [-edge_type] [-routing_props] [-core] [-driver] [-clock_net] [-is_clock] [-info] [-group] [-equal] [-names] [-sdc]
```

Return information from the clock database. If a domain is specified, by default the name of the domain is returned. If no domain is specified, by default a list of domains is returned. Options may modify the type of value that is returned.

Argument	Optional	Description
<code>[-domain <string>]</code>	Y	specifies name of domain
<code>[-pin <string>]</code>	Y	use the domain of this pin
<code>[-net <string>]</code>	Y	use the domain of this net
<code>[-all]</code>	Y	even report uninteresting domains
<code>[-unique]</code>	Y	use a unique domain for domains with the same frequency
<code>[-multi]</code>	Y	with <code>-net</code> or <code>-pin</code> : report a list of domains instead of a single domain
<code>[-freq]</code>	Y	return the frequency (MHz); requires a domain

Argument	Optional	Description
[-period]	Y	return the period (ps); requires a domain
[-phase]	Y	return the phase; requires a domain
[-edge_type]	Y	1 for pos-edge, -1 for neg-edge, 0 for combinational; requires a domain
[-routing_props]	Y	list of strings denoting the routing properties (if set); requires a domain
[-core]	Y	1 if used in the core, otherwise 0; or list of domains used in core
[-driver]	Y	name of the driving pin or port; requires a domain
[-clock_net]	Y	name of the clock net; requires a domain
[-is_clock]	Y	true if net is a clock net; requires a pin or net
[-info]	Y	list as for 'array set'; requires a domain
[-group]	Y	list of related domains, or list of groups
[-equal]	Y	list of domains with same frequency; requires a domain
[-names]	Y	list of all names for the domain; requires a domain
[-sdc]	Y	return list of sdc commands

clock_relation

```
clock_relation <domain1> <domain2> [-default] [-group] [-sdc]
```

Return relation between clocks. For related clocks the return is a list with 5 values: the word "related" followed by T1 T2 e1 e2. T is the abstract period: $T1/T2 = \text{period1}/\text{period2}$. e is an abstract offset (in the same units as T). By default the numbers are as small as possible, but with -group all related clocks use the same units.

Argument	Optional	Description
<domain1>		first domain
<domain2>		second domain

Argument	Optional	Description
[-default]	Y	apply current default_relation rule
[-group]	Y	values are in group units
[-sdc]	Y	return list of sdc commands

create_boundary_pins

```
create_boundary_pins <name> <boundary_pin_names> [-clock] [-data] [-purpose <string>]
```

This command instantiates IPIN/OPINs at the Core/IO Ring boundary

Argument	Optional	Description
<name>		A reference to a net in the design where the boundary pins should be inserted (<p:toplevel_portname> <t:user_pin_name> <n:net_name>)
<boundary_pin_names>		A list of one or more instance names for the boundary pins to be inserted on the given net
[-clock]	Y	Create a clock pin even if the specified net is a data net
[-data]	Y	Create a data pin even if the specified net is a clock net
[-purpose <string>]	Y	Set the PURPOSE property on the created boundary pin instances. Legal values are: 'USER' (the default), 'JTAG', 'CFG'.

create_equivalent_regions

```
create_equivalent_regions <source>
```

Create non-overlapping placement regions which have the same tiles in the same order as the provided <source>

Argument	Optional	Description
<source>		Name of the source. May be a region or a partition name.

create_flow_step

```
create_flow_step <id> <label> [-command <string>] [-parent_id <string>] [-required] [-skip_for_eval_mode] [-offset <int>] [-description <string>]
```

This command creates a flow step definition, which is basically a wrapper around an existing command or script that manages flow status and dependencies.

Argument	Optional	Description
<id>		The required <id> string argument specifies the identifier of the flow step to create. The <id> argument must be unique among all flow step ids in ACE. Run <code>get_flow_steps</code> for a list of existing flow step IDs.
<label>		The required <label> argument specifies the label string to display in the GUI for this flow step. The label should be as short as possible.
[-command <string>]	Y	The optional -command <command> option specifies the TCL command to run when this flow step is invoked.
[-parent_id <string>]	Y	The optional -parent_id <parentId> option specifies the flow step id of an existing flow step (which does not have a command of its own) that this new flow step will be grouped under in the flow hierarchy. Run <code>get_flow_steps</code> for a list of existing flow step IDs.
[-required]	Y	The optional -required option specifies whether or not this flow step is required for further processing of the flow. If this option is not used, the user may optionally enable or disable this flow step for use in running the flow with <code>run_flow</code> .
[-skip_for_eval_mode]	Y	The optional -skip_for_eval_mode option specifies whether or not this flow step will be skipped when <code>flow_mode</code> is set to evaluation.
[-offset <int>]	Y	The optional -offset <offset> option specifies the position (as a positive integer) under the parent flow step (or top level) at which this flow step should be inserted. Without this option, the flow step will be appended to the end of the flow steps under the parent flow step (or top level).
[-description <string>]	Y	The optional -description <description> option specifies the description text to display in the GUI for this flow step.

create_impl

```
create_impl <implName> [-project <string>] [-copy] [-not_active]
```

This command creates a new implementation in a project. This command causes the new implementation to become the active implementation.

Argument	Optional	Description
<implName>		The required <implName> argument is used to specify the name for the new implementation.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the implementation to be added to.
[-copy]	Y	The optional -copy option is used to copy the implementation options of the active implementation into the newly created implementation.
[-not_active]	Y	If this option is set, the new project impl will not be activated and the active impl in the current ACE session will not be changed.

create_option_sets

```
create_option_sets [-path <string>] [-device <string>] [-max_option_sets <int>] [-prepare_impl_options <int>] [-pnr_impl_options <int>] [-exclude_synthesis] [-init] [-verbose]
```

This command creates new option sets for a project implementation.

Argument	Optional	Description
[-path <string>]	Y	The path to write to
[-device <string>]	Y	The Achronix device name
[-max_option_sets <int>]	Y	The number of autogen option sets to create. Default is 26.
[-prepare_impl_options <int>]	Y	The number of prepare impl options to add per option set.
[-pnr_impl_options <int>]	Y	The number of place and route impl options to add per option set.
[-exclude_synthesis]	Y	Do not sweep over synthesis options

Argument	Optional	Description
<code>[-init]</code>	Y	re-create the <path> to write option sets to
<code>[-verbose]</code>	Y	print exactly what is being done

create_path

```
create_path <pins> [-id <string>] [-rgb <list>]
```

This command creates a user-defined pin path that may be used for selection or highlighting.

Argument	Optional	Description
<pins>		The required <pins> list argument specifies the ordered list of instance pins.
<code>[-id <string>]</code>	Y	The optional <code>-id <id></code> option specifies the string id to use for this path. If an id is not specified, a unique id will be automatically generated.
<code>[-rgb <list>]</code>	Y	The optional <code>-rgb <rgb></code> option is used to specify the RGB (Red-Green-Blue) color value to use for highlighting the specified objects as a 3 element list of integers {red green blue}. If the <code>-rgb</code> option is not used, then the objects in the list will be un-highlighted.

create_project

```
create_project <projectFile> [-impl <string>] [-not_active]
```

This command creates a new project in ACE.

Argument	Optional	Description
<projectFile>		The required <projectFile> argument is used to specify the project file location for the new project. The file name is used as the project's name in ACE.
<code>[-impl <string>]</code>	Y	The optional <code>-impl <implName></code> option is used to specify the name of the initial implementation for this new project.
<code>[-not_active]</code>	Y	If this option is set, the new project impl will not be activated and the active impl in the current ACE session will not be changed.

create_region

```
create_region <name> <bounds> [-find_insts <string>] [-insts <list>] [-
snap_to_clock_regions] [-snap_to_fabric_clusters] [-snap <string>] [-soft] [-type
<string>] [-flops_only] [-clocks_only] [-include_routing] [-include_constants] [-
pr_zone] [-batch] [-verbose]
```

This command creates a placement region in the Core with the given name and bounding box of tiles. Instances may be added to this placement region to create a region constraint for the placer.

Argument	Optional	Description
<name>		Name of the region
<bounds>		List of bounding box coordinates {x1 y1 x2 y2}. x1 and y1 are the upper left corner of the box. x2 and y2 are the lower right corner of the box. Alternatively, a list of sites.
[-find_insts <string>]	Y	Pass in a find command string to create the list of user design instances to constrain into this region's bounding box for the placer.
[-insts <list>]	Y	List of user design instances to constrain into this region's bounding box for the placer.
[-snap_to_clock_regions]	Y	Snap the bounding box to clock region boundaries (deprecated, use '-snap clock_regions')
[-snap_to_fabric_clusters]	Y	Snap the bounding box to fabric cluster boundaries (deprecated, use '-snap fabric_clusters')
[-snap <string>]	Y	How to snap the region bounding box coordinates. Legal values are: 'none', 'tiles' (the default), 'fabric_clusters', 'clock_regions'.
[-soft]	Y	Create a 'soft' placement region, which attempts to pull instance placement to its center, but allows instance placement to overflow the bounds of the region (deprecated, use '-type soft')

Argument	Optional	Description
<code>[-type <string>]</code>	Y	What type of placement region this is. Legal values are: 'inclusive' (the default), 'keepout', 'soft'. Instances added to an 'inclusive' region (and attached routing wires when '-include_routing' is set) will be placed within the region bounding box. An 'inclusive' region permits instances to be placed inside the region even if they do not belong to the region. A 'keepout' region prevents any instances (and routing wires when '-include_routing' is set) from being placed inside the region. No instances may be added to a 'keepout' region. Instances added to an 'soft' region will be pulled toward the region's center during placement, but instances are permitted to overflow the bounds of the 'soft' region.
<code>[-flops_only]</code>	Y	When adding instances, filter out all instances except flops
<code>[-clocks_only]</code>	Y	When adding instances, filter out all instances that do not have a connected clock pin
<code>[-include_routing]</code>	Y	Constrain routing wires, as well as instances, to stay within the region boundary box
<code>[-include_constants]</code>	Y	When adding instances, do not filter out power/ground constants
<code>[-pr_zone]</code>	Y	This will indicate that the placement region is intended to be used for partial reconfiguration.
<code>[-batch]</code>	Y	Postpone application of this constraint until <code>apply_placement</code> is called (this avoids frequent GUI updates). This option is only relevant if you manually apply placement constraints after the design has been prepared.
<code>[-verbose]</code>	Y	Print additional command status messages.

deselect

`deselect [-objects <list>]`

This command removes objects from the current list of selected objects.

Argument	Optional	Description
<code>[-objects <list>]</code>	Y	The optional <code>-objects <objects></code> option is used to specify a list of objects to remove from the current selection. Objects must be prepended with object type prefixes (see "find" command). Objects in the <code><objects></code> list that are not in the current selection are silently ignored. Without this option, the <code>deselect</code> command will remove all objects from the current selection.

disable_flow_step

`disable_flow_step <id>`

This command disables an existing optional flow step from being run during a "run" command.

Argument	Optional	Description
<code><id></code>		The required <code><id></code> argument specifies the id of the flow step to disable.

disable_project_constraints

`disable_project_constraints [-project <string>] [-impl <string>] <file>`

This command disables project constraints files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to disable constraints for.
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to disable constraints for.
<code><file></code>		The project constraints file to disable for a project implementation.

disable_project_source_file

```
disable_project_source_file <file> [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-project <string>] [-impl <string>]
```

This command disables project source files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
<file>		The project constraints file to disable for a project implementation.
[-syn_constraint]	Y	The optional -syn_constraint option is used to specify the source file to be disabled is a synthesis constraint file.
[-pnr_constraint]	Y	The optional -pnr_constraint option is used to specify the source file to be disabled is a place and route constraint file.
[-pnr_netlist]	Y	The optional -pnr_netlist option is used to specify the source file to be disabled is a place and route netlist file.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to enable source file for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to enable source file for.

display_file

```
display_file <file> [-line_number <int>] [-search <string>]
```

This command automatically opens a file in the GUI. This command has no effect in batch mode.

Argument	Optional	Description
<file>		The required <file> argument specifies the path to the file to automatically open in the GUI (when in -gui mode).
[-line_number <int>]	Y	The optional -line_number option allows you to open a text file to a particular line.

Argument	Optional	Description
<code>[-search <string>]</code>	Y	The optional <code>-search</code> option allows you to open a text file to the first line where the searchtext appears.

display_netlist

```
display_netlist <object>
```

This command attempts to open the gate level netlist file in the GUI for the given user design instance or net. This command has no effect when ACE is running in batch mode.

Argument	Optional	Description
<code><object></code>		The required <code><object></code> argument specifies the instance (i:) or net (n:) name for which the netlist file will be opened in the GUI.

display_properties

```
display_properties <object> [-print]
```

This command displays detailed properties of the specified object in the GUI, and optionally prints the details to the console.

Argument	Optional	Description
<code><object></code>		The required <code><object></code> argument specifies the object to get properties for.
<code>[-print]</code>	Y	The <code>-print</code> option will print all the object property details to the TCL console in addition to sending the data to the GUI.

draw_arc

```
draw_arc <x> <y> <width> <height> <startAngle> <arcAngle> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]
```

This command allows you to draw a custom arc on the GUI's Floorplanner view.

Argument	Optional	Description
<code><x></code>		The required <code><x></code> argument specifies the upper-left x coordinate for the arc.

Argument	Optional	Description
<y>		The required <y> argument specifies the upper-left y coordinate for the arc.
<width>		The required <width> argument specifies the width of the arc.
<height>		The required <height> argument specifies the height of the arc.
<startAngle>		The required <startAngle> argument specifies the starting angle of the arc.
<arcAngle>		The required <arcAngle> argument specifies the angle of the arc.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the arc. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the arc. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the arc as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the arc thickness in pixels. If this option is not used, a thickness of 1 will be used.
[-fill]	Y	The optional -fill option specifies whether the arc should be filled with color or not. If this option is not used, the arc will be hollow.

draw_line

```
draw_line <x1> <y1> <x2> <y2> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>]
```

This command allows you to draw a custom line on the GUI's Floorplanner view.

Argument	Optional	Description
<x1>		The required <x1> argument specifies the first x coordinate for the line.
<y1>		The required <y1> argument specifies the first y coordinate for the line.
<x2>		The required <x2> argument specifies the second x coordinate for the line.
<y2>		The required <y2> argument specifies the second y coordinate for the line.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the line. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the line. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the line as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the line thickness in pixels. If this option is not used, a thickness of 1 will be used.

draw_oval

```
draw_oval <x> <y> <width> <height> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]
```

This command allows you to draw a custom oval on the GUI's Floorplanner view.

Argument	Optional	Description
<x>		The required <x> argument specifies the upper-left x coordinate for the oval.
<y>		The required <y> argument specifies the upper-left y coordinate for the oval.
<width>		The required <width> argument specifies the width of the oval.
<height>		The required <height> argument specifies the height of the oval.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the oval. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the oval. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the oval as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the oval thickness in pixels. If this option is not used, a thickness of 1 will be used.
[-fill]	Y	The optional -fill option specifies whether the oval should be filled with color or not. If this option is not used, the oval will be hollow.

draw_polygon

```
draw_polygon <points> [-layer <int>] [-id <int>] [-rgb <list>] [-batch] [-thickness <int>] [-fill]
```

This command allows you to draw a custom polygon on the GUI's Floorplanner view.

Argument	Optional	Description
<points>		The required <points> argument specifies the list of x-y coordinates for polygon, starting with the x coordinate and alternating. For example: {1 1 2 2 3 5 1 6}.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the arc. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the arc. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the polygon as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the arc thickness in pixels. If this option is not used, a thickness of 1 will be used.
[-fill]	Y	The optional -fill option specifies whether the arc should be filled with color or not. If this option is not used, the arc will be hollow.

draw_rectangle

```
draw_rectangle <x> <y> <width> <height> [-layer <int>] [-id <int>] [-rgb <list>]
[-batch] [-thickness <int>] [-fill]
```

This command allows you to draw a custom rectangle on the GUI's Floorplanner view.

Argument	Optional	Description
<x>		The required <x> argument specifies the upper-left x coordinate for the rectangle.
<y>		The required <y> argument specifies the upper-left y coordinate for the rectangle.

Argument	Optional	Description
<width>		The required <width> argument specifies the width of the rectangle.
<height>		The required <height> argument specifies the height of the rectangle.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the rectangle. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the rectangle. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the rectangle as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.
[-thickness <int>]	Y	The optional -thickness <pixels> option specifies the rectangle thickness in pixels. If this option is not used, a thickness of 1 will be used.
[-fill]	Y	The optional -fill option specifies whether the rectangle should be filled with color or not. If this option is not used, the rectangle will be hollow.

draw_string

```
draw_string <x> <y> <string> [-layer <int>] [-id <int>] [-rgb <list>] [-batch]
```

This command allows you to draw a custom string on the GUI's Floorplanner view.

Argument	Optional	Description
<x>		The required <x> argument specifies the x coordinate for the string.

Argument	Optional	Description
<y>		The required <y> argument specifies the y coordinate for the string.
<string>		The required <string> argument specifies the string text.
[-layer <int>]	Y	The optional -layer <layer> option specifies the drawing layer for the string. If this option is not used, the top layer (5) will be used. Using a value of 0 will draw on the background.
[-id <int>]	Y	The optional -id <id> option specifies a unique id for the string. If this option is not used, a unique id will be automatically generated and returned by the command.
[-rgb <list>]	Y	The optional -rgb <rgb> option specifies the rgb color value for the string as a 3 element list of integers {red green blue}. If this option is not used, the color blue will be used.
[-batch]	Y	The optional -batch option causes the GUI to not refresh after this command. This is useful when running many draw commands in a row. Afterwards, refresh_drawing can be called.

enable_flow_step

```
enable_flow_step <id>
```

This command enables an existing optional flow step to be run during a "run" command.

Argument	Optional	Description
<id>		The required <id> argument specifies the id of the flow step to enable.

enable_project_constraints

```
enable_project_constraints [-project <string>] [-impl <string>] <file>
```

This command enables project constraints files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to enable constraints for.
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to enable constraints for.
<code><file></code>		The project constraints file to enable for a project implementation.

enable_project_source_file

```
enable_project_source_file <file> [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-project <string>] [-impl <string>]
```

This command enables project source files for a project implementation. If no project or impl names are specified, the currently active project implementation is used.

Argument	Optional	Description
<code><file></code>		The project constraints file to enable for a project implementation.
<code>[-syn_constraint]</code>	Y	The optional <code>-syn_constraint</code> option is used to specify the source file to be enabled is a synthesis constraint file.
<code>[-pnr_constraint]</code>	Y	The optional <code>-pnr_constraint</code> option is used to specify the source file to be enabled is a place and route constraint file.
<code>[-pnr_netlist]</code>	Y	The optional <code>-pnr_netlist</code> option is used to specify the source file to be enabled is a place and route netlist file.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to enable source file for.

Argument	Optional	Description
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to enable source file for.

export_all_partitions

```
export_all_partitions [-info_list]
```

Command to export the place-and-route database and blackbox Verilog model for all leaf-level partitions in the design

Argument	Optional	Description
<code>[-info_list]</code>	Y	Return a Tcl list containing {<partition> <view> <epdb filename> <blackbox filename>} for each partition

export_partition

```
export_partition <partition> [-dboutputfile <string>] [-bboutputfile <string>] [-info_list]
```

Command to export the place-and-route database and blackbox Verilog model for a partition

Argument	Optional	Description
<code><partition></code>		Export the place-and-route database and blackbox Verilog model for the specified partition
<code>[-dboutputfile <string>]</code>	Y	Specifies the output file name for the partition database (default is <code><active_impl_dir>/pnr/output/partitions/<cellname>_<partition>.epdb</code>)
<code>[-bboutputfile <string>]</code>	Y	Specifies the output file name for the partition blackbox Verilog model (default is <code><active_impl_dir>/pnr/output/blackboxes/<cellname>_bb.v</code>)
<code>[-info_list]</code>	Y	Return a Tcl list containing {<partition> <view> <epdb filename> <blackbox filename>} for each partition

filter

```
filter <objects> [-patterns <list>] [-insts] [-nets] [-ports] [-pins] [-paths] [-sites] [-filter <string>] [-no_prefix]
```

This command takes a TCL list of DB objects and returns a filtered TCL list of objects that match the filter options passed in. Each object name in the returned list is prepended with an object type indicator (unless `-no_prefix` is used). Object types prefixes are: p: = port (top level user design), t: = pin, i: = instance, n: = net. Find results may contain a mixture of object types. The `-insts`, `-nets`, `-ports`, and `-pins` object type options may be used to filter the results to just those object types. Specifying no object type options will result in a search of all object types.

Argument	Optional	Description
<objects>		The required <objects> argument specifies a list of object names to filter.
[-patterns <list>]	Y	The optional <code>-patterns</code> argument specifies a list of pattern strings to match object names against. Each pattern string in the list may use '*' and '?' wildcard characters for matching.
[-insts]	Y	The optional <code>-insts</code> object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-insts</code> option is not used, then the results will not contain any instance objects.
[-nets]	Y	The optional <code>-nets</code> object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-nets</code> option is not used, then the results will not contain any net objects.
[-ports]	Y	The optional <code>-ports</code> object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-ports</code> option is not used, then the results will not contain any top level user design port objects.
[-pins]	Y	The optional <code>-pins</code> object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-pins</code> option is not used, then the results will not contain any pin objects.

Argument	Optional	Description
<code>[-paths]</code>	Y	The optional <code>-paths</code> object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-paths</code> option is not used, then the results will not contain any path objects.
<code>[-sites]</code>	Y	The optional <code>-sites</code> object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-sites</code> option is not used, then the results will not contain any site objects.
<code>[-filter <string>]</code>	Y	The optional <code>-filters</code> option may be used to specify a boolean expression of object properties to filter the results with. Each property filter in the expression must follow the filter syntax of <code>@<propertyname><operator><value></code> . Multiple property filters may be used in the expression by using boolean operators. For example: "find * -filter {@type=DFF @type=LUT4}". The supported filter property names are currently: <code>@async_reset</code> , <code>@attribute</code> , <code>@clock</code> , <code>@clock_as_data</code> , <code>@clock_domain</code> , <code>@clock_region</code> , <code>@data_as_clock</code> , <code>@direction</code> , <code>@driver_type</code> , <code>@driving_net</code> , <code>@driving_pin</code> , <code>@enable</code> , <code>@fanout</code> , <code>@fixed_placement</code> , <code>@limit</code> , <code>@partition</code> , <code>@placed</code> , <code>@power</code> , <code>@power_rank</code> , <code>@region</code> , <code>@reset</code> , <code>@sink_type</code> , or <code>@type</code> . The supported filter operators are currently: <code>></code> , <code><</code> , <code>!</code> , and <code>=</code> . The supported boolean operators (when using multiple filters) are currently: <code>&&</code> , <code> </code> , and <code>==</code> .
<code>[-no_prefix]</code>	Y	The optional <code>-no_prefix</code> option is used to remove the object type prefix from the names returned in the results.

See also: [Object Type Prefixes \(page 335\)](#), [Search Filter Builder Dialog \(page 186\)](#), [Filter Properties \(page 264\)](#), [find \(page 638\)](#), [Search View. \(page 129\)](#)

find

```
find <patterns> [-insts] [-nets] [-ports] [-pins] [-paths] [-sites] [-filter
<string>] [-sort <string>] [-sort_order <string>] [-no_prefix] [-no_refresh] [-
handle] [-warning] [-error]
```

This command returns a TCL list of object names that match any of the pattern strings passed in. Each object name in the returned list is prepended with an object type indicator (unless `-no_prefix` is used). Object types prefixes are: `p` = port (top level user design), `t` = pin, `i` = instance, `n` = net. Find results may contain a mixture of object types. The `-insts`, `-nets`, `-ports`, and `-pins` object type options may be used to filter the results to just those object types. Specifying no object type options will result in a search of all object types.

Argument	Optional	Description
<patterns>		The required <patterns> argument specifies a list of pattern strings to match object names against. The pattern for matching with specific pins must have separator '/' in the form of <instance pattern>/<pin pattern>. Each pattern string in the list may use '*' and '?' wildcard characters for matching.
[-insts]	Y	The optional -insts object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -insts option is not used, then the results will not contain any instance objects.
[-nets]	Y	The optional -nets object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -nets option is not used, then the results will not contain any net objects.
[-ports]	Y	The optional -ports object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -ports option is not used, then the results will not contain any top level user design port objects.
[-pins]	Y	The optional -pins object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -pins option is not used, then the results will not contain any pin objects.
[-paths]	Y	The optional -paths object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -paths option is not used, then the results will not contain any path objects.

Argument	Optional	Description
<code>[-sites]</code>	Y	The optional <code>-sites</code> object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-sites</code> option is not used, then the results will not contain any site objects.
<code>[-filter <string>]</code>	Y	The optional <code>-filters</code> option may be used to specify a boolean expression of object properties to filter the results with. Each property filter in the expression must follow the filter syntax of <code>@<propertyname><operator><value></code> . Multiple property filters may be used in the expression by using boolean operators. For example: <code>"find * -filter {@type=DFF @type=LUT4}"</code> . The supported filter property names are currently: <code>@async_reset</code> , <code>@attribute</code> , <code>@clock</code> , <code>@clock_as_data</code> , <code>@clock_domain</code> , <code>@clock_region</code> , <code>@data_as_clock</code> , <code>@direction</code> , <code>@driver_type</code> , <code>@driving_net</code> , <code>@driving_pin</code> , <code>@enable</code> , <code>@fanout</code> , <code>@fixed_placement</code> , <code>@limit</code> , <code>@partition</code> , <code>@placed</code> , <code>@power</code> , <code>@power_rank</code> , <code>@region</code> , <code>@reset</code> , <code>@sink_type</code> , or <code>@type</code> . The supported filter operators are currently: <code>></code> , <code><</code> , <code>!</code> , and <code>=</code> . The supported boolean operators (when using multiple filters) are currently: <code>&&</code> , <code> </code> , and <code>==</code> .
<code>[-sort <string>]</code>	Y	The <code>-sort</code> option allows the user to specify the type of sort performed on the find results list. The default is "dictionary". Other options are "ascii" or "none"
<code>[-sort_order <string>]</code>	Y	The <code>-sort_order</code> option allows the user to specify the direction of sort performed on the find results list. You may specify either "increasing" or "decreasing". The default is "increasing".
<code>[-no_prefix]</code>	Y	The optional <code>-no_prefix</code> option is used to remove the object type prefix from the names returned in the results
<code>[-no_refresh]</code>	Y	The optional <code>-no_refresh</code> option is used to prevent sending an update to the GUI Search View to optimize speed
<code>[-handle]</code>	Y	The optional <code>-handle</code> option is used to return the reserve string "@@FindResults" instead of the TCL list of object names. This handle can then be used in the highlight command to speed up processing by avoiding extra name parsing
<code>[-warning]</code>	Y	Print warning message if the find command does not find any objects.

Argument	Optional	Description
<code>[-error]</code>	Y	Print message and error out if the find command does not find any objects.

The ACE GUI provides a graphical interface for the `find` command through the [Search View \(page 129\)](#). See also: [Object Type Prefixes \(page 335\)](#), [Search Filter Builder Dialog \(page 186\)](#), [Filter Properties \(page 264\)](#), [filter \(page 637\)](#), [select \(page 708\)](#), [Selection View \(page 133\)](#), [trace_connections. \(page 717\)](#)

generate_ioring_design_files

```
generate_ioring_design_files <outputDir> [-add_to_project]
```

This command generates the all IO Ring design files for the active ACE project, using all ACXIP files that have been added to the active project.

Argument	Optional	Description
<code><outputDir></code>		The required <code><outputDir></code> argument allows the user to specify the directory path to output the IO Ring design files into.
<code>[-add_to_project]</code>	Y	Using the <code>-add_to_project</code> option automatically adds the required generated IO Ring design files to your ACE project. This includes utilization XML, SDC constraints, PDC constraints, and IO Ring bitstream files.

generate_ip_design_files

```
generate_ip_design_files <acxipFile>
```

This command generates the enabled design files for a given IP configuration (.acxip file).

Argument	Optional	Description
<code><acxipFile></code>		The required <code><acxipFile></code> argument specifies the IP configuration (.acxip file) to generate design files for.

generate_route_delay_table

```
generate_route_delay_table [-outputfile <string>]
```

This command extracts route delay numbers on nets for estimating the cell-cell route delays vs fanout.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile <file></code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation <code>.debug</code> directory and is named <code><design_name>_route_delay.log</code> .

get_ace_cputime

```
get_ace_cputime
```

This command returns the cumulative cpu time of this ACE process

get_ace_current_memory_usage

```
get_ace_current_memory_usage
```

This command returns the current memory usage (in kB) of this ACE process

get_ace_ext_dir

```
get_ace_ext_dir
```

This command returns the path to the ACE Extensions directory if one has been enabled.

get_ace_ext_lib

```
get_ace_ext_lib <partName>
```

This command returns the blackbox library path in the ACE Extensions directory for the specified partname if one has been enabled.

Argument	Optional	Description
<code><partName></code>		The required <code><partName></code> argument is used to specify the name of the target device to find the blackbox library file for. The part name specified must exist among the valid part names in the ACE installation.

get_ace_peak_memory_usage

```
get_ace_peak_memory_usage
```

This command returns the maximum memory usage (in kB) of this ACE process during the current session

get_ace_version

```
get_ace_version [-buildid] [-builddate] [-full]
```

This command returns the version of ACE

Argument	Optional	Description
[-buildid]	Y	The optional -buildid option will return the buildid.
[-builddate]	Y	The optional -builddate option will say when ace was built.
[-full]	Y	The optional -full option will return the full ACE version and build designation.

get_active_impl

```
get_active_impl [-quiet]
```

This command returns the name of the active implementation in the current ACE session.

Argument	Optional	Description
[-quiet]	Y	do not print a message if there is no active project

Example

To automatically set the value of the `set_impl_option -impl` option, after the `create_project -impl` command has been run, which defines the name of the active impl, the following command can be used:

```
set_impl_option -project [get_active_project] -impl [get_active_impl]
"partname" "AC16tSC01HI01C"
```

Also See

[create_project](#) (page 623)

[get_active_project](#) (page 643)

[set_impl_option](#) (page 711)

get_active_project

```
get_active_project [-quiet] [-path]
```

This command returns the name of the active project (which contains the active implementation) in the current ACE session.

Argument	Optional	Description
<code>[-quiet]</code>	Y	do not print a message if there is no active project
<code>[-path]</code>	Y	Return the file path to the active project's acxprj project file, instead of the project name

get_best_multiprocess_impl

```
get_best_multiprocess_impl
```

This command finds the best impl from the MultiProcess Summary Report in the active project.

get_clock_region_bounds

```
get_clock_region_bounds <region>
```

Returns the bounding box for a clock region

Argument	Optional	Description
<code><region></code>		Name of the region

get_clock_regions

```
get_clock_regions
```

Returns the list of clock region names for the device

get_clock_type

```
get_clock_type <clock>
```

Get properties of a clock. For a non-driving (target) clock pin, this is a combination of local properties and properties of the clock domain.

Argument	Optional	Description
<code><clock></code>		net or pin ('inst/pin')

get_compatible_ordering_codes

```
get_compatible_ordering_codes
```


This command returns a list of compatible ordering codes for the active project based on it's device, package, and speed grade.

get_compatible_placements

```
get_compatible_placements <source> [-anchor <string>] [-outputfile <string>]
```

Get a list of compatible placements for the given <source> partition

Argument	Optional	Description
<source>		Name of the source partition
[-anchor <string>]	Y	Name of the anchor instance. An anchor instance will be chosen automatically if not given.
[-outputfile <string>]	Y	Name of optional output file. If given, the compatible placements will be written out as a series of set_placement commands.

get_current_design

```
get_current_design [-quiet]
```

This command returns the name of the top module in the current design. This command returns an error if no current design is loaded.

Argument	Optional	Description
[-quiet]	Y	do not print a message if there is no active project

get_current_partname

```
get_current_partname [-quiet]
```

This command returns the name of the currently loaded device.

Argument	Optional	Description
[-quiet]	Y	do not warn if there is no current part

get_efd_file_path

```
get_efd_file_path <partName>
```

This command returns the path to the efd file for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the efd file for. The part name specified must exist among the valid part names in the ACE installation.

get_enabled_constraints

```
get_enabled_constraints [-project <string>] [-impl <string>]
```

This command returns a list of all the enabled constraint files for an implementation.

Argument	Optional	Description
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.

get_enabled_source_files

```
get_enabled_source_files [-project <string>] [-impl <string>] [-syn_constraint] [-pnr_constraint] [-pnr_netlist]
```

This command returns a list of all the enabled source files for an implementation.

Argument	Optional	Description
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get enabled constraints for.
[-syn_constraint]	Y	The optional -syn_constraint option is used to return a list of enabled synthesis constraint file paths.

Argument	Optional	Description
<code>[-pnr_constraint]</code>	Y	The optional <code>-pnr_constraint</code> option is used to return a list of enabled place and route constraint file paths.
<code>[-pnr_netlist]</code>	Y	The optional <code>-pnr_netlist</code> option is used to return a list of enabled place and route netlist file paths.

get_fabricdb_path

```
get_fabricdb_path <partName>
```

This command returns the path to the fabric db file for the given part.

Argument	Optional	Description
<code><partName></code>		The required <code><partName></code> argument is used to specify the name of the part to find the fabric db for. The part name specified must exist among the valid part names in the ACE installation.

get_file_line

```
get_file_line <object>
```

This command returns the file path and line offset into to the source netlist for the given user design instance or net.

Argument	Optional	Description
<code><object></code>		The required <code><object></code> argument specifies the instance (i:) or net (n:) name for which the file line will be retrieved.

get_flow_steps

```
get_flow_steps
```

This command returns a list of all the currently defined flow step id strings.

get_impl_names

```
get_impl_names [-project <string>]
```

This command returns a list of all the implementation names for an existing project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the implementation names from.

get_impl_option

```
get_impl_option <option_name> [-project <string>] [-impl <string>] [-syn] [-type]
[-range] [-default]
```

This command returns the current value of a project implementation option. Only one option value may be retrieved at a time.

Argument	Optional	Description
<code><option_name></code>		The name of the impl option to retrieve a value for. To see a list of valid impl options, use the <code>report_impl_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to get options for.
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to get options for.
<code>[-syn]</code>	Y	option is synthesis option
<code>[-type]</code>	Y	return the type of this option - bool,list,string,integer
<code>[-range]</code>	Y	return the permissible range of this option if bounded
<code>[-default]</code>	Y	return the default value

get_impl_option_is_supported

```
get_impl_option_is_supported <option_name> [-project <string>] [-impl <string>]
```

Returns '1' if the impl option is supported on the current device, otherwise returns '0'

Argument	Optional	Description
<option_name>		The name of the impl option to retrieve a value for. To see a list of valid impl options, use the report_impl_options TCL command.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.

get_impl_option_is_supported_and_enabled

```
get_impl_option_is_supported_and_enabled <option_name> [-project <string>] [-impl <string>]
```

Returns '1' if the impl option is supported on the current device, otherwise returns '0'

Argument	Optional	Description
<option_name>		The name of the impl option to retrieve a value for. To see a list of valid impl options, use the report_impl_options TCL command.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to get options for.

get_inst_partition

```
get_inst_partition <instance>
```

Returns the Partition associated with a user design instance

Argument	Optional	Description
<instance>		Name of the instance

get_inst_region

```
get_inst_region <instance>
```

Returns the region associated with a user design instance

Argument	Optional	Description
<instance>		Name of the instance

get_installation_directory

```
get_installation_directory
```

This command returns the path to the root of the ACE installation.

get_location

```
get_location <object>
```

This command allows you to get the location of an object (i:instance, s:site, or t:pin) on the GUI's Floorplanner view.

Argument	Optional	Description
<object>		The required <object> argument specifies the object to get coordinates for. The correct object type prefix is required.

get_package_names

```
get_package_names [<partName>] [-default]
```

This command returns the list of valid package names for the specified part.

Argument	Optional	Description
[<partName>]	Y	Name of part
[-default]	Y	Returns the default package value

get_part_names

```
get_part_names [-default]
```

This command returns the list of valid part names in the installed library.

Argument	Optional	Description
[-default]	Y	Returns the default partname value

get_partition_changed

```
get_partition_changed <name>
```

Get the changed flag of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition

get_partition_force_changed

```
get_partition_force_changed <name>
```

Get the force changed flag of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition

get_partition_info

```
get_partition_info <name> [-timestamp] [-comment] [-view] [-type] [-is_import] [-import_from] [-changed] [-id] [-disabled] [-parent] [-is_top] [-is_parent]
```

Get info of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition
[-timestamp]	Y	get timestamp
[-comment]	Y	get comment
[-view]	Y	get view name
[-type]	Y	get partition type
[-is_import]	Y	was the partition imported (0 or 1)?

Argument	Optional	Description
[-import_from]	Y	file the partition was imported from
[-changed]	Y	has the partition timestamp changed
[-id]	Y	unique partition id number
[-disabled]	Y	is the partition disabled (0 or 1)?
[-parent]	Y	name of the partition's parent partition
[-is_top]	Y	is this the top partition (0 or 1)?
[-is_parent]	Y	is this partition a parent of another (0 or 1)?

get_partition_insts

```
get_partition_insts <name>
```

Returns the list of user design instances in a partition

Argument	Optional	Description
<name>		Name of the Partition

get_partition_names

```
get_partition_names
```

Returns the list of user design partition names

get_partition_timestamp

```
get_partition_timestamp <name>
```

Get the timestamp of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition

get_partition_type

```
get_partition_type <name>
```


Get the type of a partition with the given name

Argument	Optional	Description
<name>		Name of the Partition

get_path_ids

```
get_path_ids
```

This command returns path IDs from the last timing run.

get_path_property

```
get_path_property <id> [-pins] [-insts] [-nets] [-frequency] [-type] [-rgb] [-text] [-slack]
```

This command returns path properties.

Argument	Optional	Description
<id>		The required <id> argument specifies the id of the path to get properties for.
[-pins]	Y	The optional -pins option returns the list of pin names that make up this path.
[-insts]	Y	The optional -insts option returns the list of instance names on this path.
[-nets]	Y	The optional -nets option returns the list of net names on this path.
[-frequency]	Y	The optional -frequency option returns the frequency of this path in MHz. If no frequency is defined, -1 is returned.
[-type]	Y	The optional -type option returns the type of path: Setup Check Met, Setup Check Violated, Hold Check Met, Hold Check Violated, Hardware Limit, or User-Defined.
[-rgb]	Y	The optional -rgb option returns the integer rgb highlight color value for the path. A value of -1 means it is not highlighted.
[-text]	Y	The optional -text option returns the details text for this path.

Argument	Optional	Description
<code>[-slack]</code>	Y	The optional <code>-slack</code> option returns the slack for this path.

get_placement

```
get_placement <objName>
```

This command returns the site of the specified placed instance

Argument	Optional	Description
<code><objName></code>		The required <code><objName></code> argument is used to specify the instance or port to get placement for.

get_project_constraint_files

```
get_project_constraint_files [-project <string>]
```

This command returns a list of all the constraint file paths for a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the constraint file paths from.

get_project_directory

```
get_project_directory [-project <string>]
```

This command returns the path to a project file's parent directory

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the directory path from.

get_project_ip_files

```
get_project_ip_files [-project <string>]
```

This command returns a list of all the IP settings file paths for a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the IP settings file paths from.

get_project_names

```
get_project_names
```

This command returns a list of all the project names loaded in the current ACE session.

get_project_netlist_files

```
get_project_netlist_files [-project <string>] [-noimpl]
```

This command returns a list of all the netlist file paths for a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the netlist file paths from.
<code>[-noimpl]</code>	Y	Do not list impl-specific files, even with active impl.

get_project_option

```
get_project_option <option_name> [-project <string>] [-type] [-range] [-default]
[-quiet]
```

This command returns the current value of a project option. Only one option value may be retrieved at a time.

Argument	Optional	Description
<code><option_name></code>		The name of the project option to retrieve a value for. To see a list of valid project options, use the <code>report_project_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get options for.
<code>[-type]</code>	Y	return the type of this option - bool,list,string,integer
<code>[-range]</code>	Y	return the permissible range of this option if bounded

Argument	Optional	Description
<code>[-default]</code>	Y	return the default value
<code>[-quiet]</code>	Y	quietly return default if no project active.

get_project_option_is_supported

```
get_project_option_is_supported <option_name> [-project <string>]
```

Returns '1' if the project option is supported on the current device, otherwise returns '0'

Argument	Optional	Description
<code><option_name></code>		The name of the project option to retrieve a value for. To see a list of valid project options, use the <code>report_project_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get options for.

get_project_option_is_supported_and_enabled

```
get_project_option_is_supported_and_enabled <option_name> [-project <string>]
```

Returns '1' if the project option is supported on the current device, otherwise returns '0'

Argument	Optional	Description
<code><option_name></code>		The name of the project option to retrieve a value for. To see a list of valid project options, use the <code>report_project_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get options for.

get_project_output_directory

```
get_project_output_directory [-project <string>]
```

This command returns the path to a project output directory specified by project option "project_output_path". By default, this command returns the project file's parent directory

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the directory path from.

get_project_source_files

```
get_project_source_files [-project <string>] [-ip] [-rtl] [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-sim_tb]
```

This command returns a list of all the source file paths for a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to get the constraint file paths from.
<code>[-ip]</code>	Y	The optional <code>-ip</code> option is used to return a list of IP settings file paths.
<code>[-rtl]</code>	Y	The optional <code>-rtl</code> option is used to return a list of RTL file paths.
<code>[-syn_constraint]</code>	Y	The optional <code>-syn_constraint</code> option is used to return a list of synthesis constraint file paths.
<code>[-pnr_constraint]</code>	Y	The optional <code>-pnr_constraint</code> option is used to return a list of place and route constraint file paths.
<code>[-pnr_netlist]</code>	Y	The optional <code>-pnr_netlist</code> option is used to return a list of place and route netlist file paths.
<code>[-sim_tb]</code>	Y	The optional <code>-sim_tb</code> option is used to return a list of simulation testbench file paths.

get_properties

```
get_properties <object>
```

This command returns the list of option-value pairs for the specified object.

Argument	Optional	Description
<object>		The required <object> argument specifies the object to get properties for.

get_property

```
get_property <object> <propName> [-object_type <string>]
```

This command returns the specified property value for the specified object.

Argument	Optional	Description
<object>		The required <object> argument specifies which object will be queried.
<propName>		The required <propName> argument specifies the name of the property whose value will be retrieved.
[-object_type <string>]	Y	type of object: cell pin net port clock

get_pvt_corners

```
get_pvt_corners [<partName>] [-default] [-speed_grade_only] [-speed_grade <string>] [-core_voltage_only] [-core_voltage <string>] [-junction_temperature_only]
```

This command returns a list of the valid PVT corners for the specified part

Argument	Optional	Description
[<partName>]	Y	Name of part
[-default]	Y	Returns the default PVT corner.
[-speed_grade_only]	Y	Returns a list of valid speed_grade values for a specified part.
[-speed_grade <string>]	Y	Speed grade value.
[-core_voltage_only]	Y	Returns a list of valid core_volatge values for a specified part and speed_grade.
[-core_voltage <string>]	Y	Core voltage value.

Argument	Optional	Description
<code>[-junction_temperature_only]</code>	Y	Returns a list of valid junction_temperature values for a specified part, speed_grade and core_voltage.

get_region_bounds

`get_region_bounds <region>`

Returns the bounding box for a placement region constraint

Argument	Optional	Description
<code><region></code>		Name of the region

get_region_insts

`get_region_insts <region>`

Returns the list of user design instances in a placement region constraint

Argument	Optional	Description
<code><region></code>		Name of the region

get_regions

`get_regions [-verbose]`

Returns the list of placement region constraint names

Argument	Optional	Description
<code>[-verbose]</code>	Y	print region information with each region

get_report_sweep_temperature_corners

`get_report_sweep_temperature_corners [-quiet]`

This command returns a list of the valid junction temperatures for the target device at the given speed grade and core voltage level

Argument	Optional	Description
[-quiet]	Y	do not print a message if there is no active project

get_selection

`get_selection [-insts] [-nets] [-ports] [-pins] [-paths] [-sites] [-handle]`

This command returns the current list of selected objects.

Argument	Optional	Description
[-insts]	Y	The optional -insts object type option is used to specify that the results may include instance object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -insts option is not used, then the results will not contain any instance objects.
[-nets]	Y	The optional -nets object type option is used to specify that the results may include net object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -nets option is not used, then the results will not contain any net objects.
[-ports]	Y	The optional -ports object type option is used to specify that the results may include top level user design port object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -ports option is not used, then the results will not contain any top level user design port objects.
[-pins]	Y	The optional -pins object type option is used to specify that the results may include pin object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -pins option is not used, then the results will not contain any pin objects.
[-paths]	Y	The optional -paths object type option is used to specify that the results may include path object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the -paths option is not used, then the results will not contain any path objects.

Argument	Optional	Description
<code>[-sites]</code>	Y	The optional <code>-sites</code> object type option is used to specify that the results may include site object types. If no other object type option is used, all object types will be included in the results by default. If another object type option is used, and the <code>-sites</code> option is not used, then the results will not contain any site objects.
<code>[-handle]</code>	Y	The optional <code>-handle</code> option is used to return the reserve string " <code>@@Selection</code> " instead of the TCL list of object names. This handle can be used in the <code>highlight</code> command to speed up processing by avoiding extra name parsing

get_synprj_from_netlist

```
get_synprj_from_netlist [<filename>]
```

This command returns the name the of the synthesis prj-file associated with the ace project, if available.

Argument	Optional	Description
<code>[<filename>]</code>	Y	Name of file whose prj-file you want to find [any]

get_synprj_from_project

```
get_synprj_from_project
```

This command returns the name the of the synthesis prj-file associated with the ace project, if available.

get_techlib_name

```
get_techlib_name <partName>
```

This command returns the name the of black box verilog library for the given part.

Argument	Optional	Description
<code><partName></code>		The required <code><partName></code> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlib_path

```
get_techlib_path <partName>
```

This command returns the path to the black box verilog library file for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibdb_path

```
get_techlibdb_path <partName>
```

This command returns the path to the techlib db file for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the techlib db for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibt_name

```
get_techlibt_name <partName>
```

This command returns the name of the transmuted black box verilog library for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibt_path

```
get_techlibt_path <partName>
```

This command returns the path to the transmuted black box verilog library file for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibx_name

```
get_techlibx_name <partName>
```

This command returns the name the of the expanded black box verilog library for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_techlibx_path

```
get_techlibx_path <partName>
```

This command returns the path to the expanded black box verilog library file for the given part.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the part to find the library for. The part name specified must exist among the valid part names in the ACE installation.

get_worst_path

```
get_worst_path [-type <string>]
```

This command returns the path ID with the worst slack from the last timing run.

Argument	Optional	Description
[-type <string>]	Y	The optional -type <type> option returns the worst slack path ID for the specified type of path: Setup Check Met, Setup Check Violated, Hold Check Met, Hold Check Violated, Hardware Limit, or User-Defined.

has_ace_ext_lib

has_ace_ext_lib <partName>

This command returns a 1 if the blackbox library path is configured in the ACE Extensions directory for the specified partname.

Argument	Optional	Description
<partName>		The required <partName> argument is used to specify the name of the target device to find the blackbox library file for. The part name specified must exist among the valid part names in the ACE installation.

has_partitions

has_partitions

Check if partitions have been defined on the design via the *.prt file

highlight

highlight <objects> [-rgb <list>] [-batch] [-clear]

This command is used to highlight or un-highlight a list of objects in the GUI's physical view.

Argument	Optional	Description
<objects>		The required <objects> argument is used to specify a list of objects which will have their highlight color set. Highlight of instance, net, and path object types is currently supported. All other object types passed in will be silently ignored. Objects must be prepended with object type prefixes (see "find" command).
[-rgb <list>]	Y	The optional -rgb <rgb> option is used to specify the RGB (Red-Green-Blue) color value to use for highlighting the specified objects as a 3 element list of 8-bit (0-255) integers {red green blue}. If the -rgb option is not used, then the objects in the list will be un-highlighted.
[-batch]	Y	The optional -batch option is used to suppress the refresh of highlighting information to the GUI. This can be useful (faster) if highlighting multiple groups of nets in a loop, since each highlight command that affects a net will otherwise refresh the entire routing data set in the GUI.
[-clear]	Y	The optional -clear option is used to clear all prior highlighting

ignore_cancel

```
ignore_cancel <script>
```

Temporarily ignore cancel button. Useful to execute cleanup commands in a flow step after a cancel has been caught.

Argument	Optional	Description
<script>		commands to execute

import_synplify_project_file

```
import_synplify_project_file [<project_file_path>]
```

This command will allow the user to import all of the synthesis project options, RTL files, and Constraints into their ACE project automatically from an existing Synplify PRJ file.

Argument	Optional	Description
[<project_file_path>]	Y	The <project_file_path> is used to specify the path to a synplify PRJ file

initialize_flow

```
initialize_flow
```

This command clears the current flow model, then sources the master flow.tcl script. The master flow.tcl script uses these flow TCL commands to define the default flow.

insert_delay

```
insert_delay <pinlist>
```

This command parses the user directive to add extra delays for paths that require additional delay for alleviating timing violations.

Argument	Optional	Description
<pinlist>		The required {pinlist} option is used to specify the load pins of a net that need to be driven by inserted gates. For each pin, you can optionally specify an integer delay value by using the format <delay>,<pin_name>. The default delay value is 1.

is_impl_option_enabled

```
is_impl_option_enabled <option_name> [-project <string>] [-impl <string>]
```

This command returns if an implementation option is enabled.

Argument	Optional	Description
<code><option_name></code>		The name of the impl option to retrieve its enabled state. To see a list of valid impl options, use the <code>report_impl_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to get options for.
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to get options for.

is_incremental_compile

```
is_incremental_compile
```

Check if the Incremental Compile Impl Option is set to 1, and that partitions have been defined on the design via the *.prt file

is_labmode

```
is_labmode
```

This command returns 1 if we are in labmode .

is_project_option_enabled

```
is_project_option_enabled <option_name> [-project <string>]
```

This command returns if a project option is enabled.

Argument	Optional	Description
<code><option_name></code>		The name of the project option to retrieve its enabled state. To see a list of valid project options, use the <code>report_project_options</code> TCL command.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project implementation (by name) to get options for.

is_timing_met

```
is_timing_met [-type <string>]
```

This command returns 1 if there is positive slack on all critical paths from the last timing run. Otherwise, it returns 0

Argument	Optional	Description
[-type <string>]	Y	The optional -type <type> option returns 1 if there is positive slack for all paths of the specified type of path: Setup Check Met, Setup Check Violated, Hold Check Met, Hold Check Violated, Hardware Limit, or User-Defined. Otherwise, it returns 0

load_flowscripts

```
load_flowscripts [-bitstream]
```

This command loads all of the encrypted flow scripts.

Argument	Optional	Description
[-bitstream]	Y	load only scripts relevant for bitstream

load_project

```
load_project <projectFile> [-not_active] [-force]
```

This command loads a project file into ACE. Loading a project file does not load the design files, it just sets up a project for later use.

Argument	Optional	Description
<projectFile>		The required <projectFile> argument specifies the path to a project file.
[-not_active]	Y	If this option is set, no impl in the project will be activated and the active impl in ACE will not be changed.
[-force]	Y	The -force option can be used to override a project lock that has been set by another ACE session. Using -force causes the current ACE session to take ownership of the project lock for the project being restored. DO NOT use this option to run multiple ACE sessions on the same project at the same time, or else output files (acxprj, acxdb, icdb, jam, etc) and log files may become corrupted!

message

```
message <msg> [-file <string>] [-info] [-warning] [-error] [-none] [-console_off]
[-console_on]
```

This command prints a status message to the console and the log file.

Argument	Optional	Description
<msg>		The message to be printed.
[-file <string>]	Y	The TCL channel to use - if set and not 'none', only print the message to the file.
[-info]	Y	Make this message an informational message.
[-warning]	Y	Make this message a warning message.
[-error]	Y	Make this message an error message.
[-none]	Y	Make this message a none message (no prefix).
[-console_off]	Y	turn off console io
[-console_on]	Y	turn on console io

The -file argument, if specified, and not *none*, can be any TCL-channel - it is not limited to a file. For example:

```
set outfile [open "m.txt" w+]
message -file $outfile -info "Hello World to a file!"
message -file $outfile "Another info message : Hello World2 to a file!"
message -file $outfile -none "This is a message without prefix."
close $outfile
message -file none "This message goes to the console (and your ace-logfile)"
message "And so does this one !"
```

You can specify only one of the -info, -none, -warn, -error options.

move_project_constraints

```
move_project_constraints [-project <string>] <file> <offset>
```

This command moves a project constraints file to the specified offset to allow re-ordering of constraints within a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to move constraints for.
<code><file></code>		The project constraints file to move.
<code><offset></code>		The offset to move the project constraints file to. Other constraints files will be moved down automatically.

move_project_netlists

```
move_project_netlists [-project <string>] <file> <offset>
```

This command moves a project netlist file to the specified offset to allow re-ordering of netlists within a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to move netlists for.
<code><file></code>		The project netlist file to move.
<code><offset></code>		The offset to move the project netlist file to. Other netlist files will be moved down automatically.

move_project_source_file

```
move_project_source_file [-project <string>] <file> <offset> [-rtl] [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-sim_tb]
```

This command moves a project RTL file to the specified offset to allow re-ordering of RTL files within a project.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to move RTL file for.
<code><file></code>		The project RTL file to move.
<code><offset></code>		The offset to move the project RTL file to. Other RTL files will be moved down automatically.

Argument	Optional	Description
[-rtl]	Y	The optional -rtl option is used to specify the source file to be moved is an RTL file.
[-syn_constraint]	Y	The optional -syn_constraint option is used to specify the source file to be moved is a synthesis constraint file.
[-pnr_constraint]	Y	The optional -pnr_constraint option is used to specify the source file to be moved is a place and route constraint file.
[-pnr_netlist]	Y	The optional -pnr_netlist option is used to specify the source file to be moved is a place and route netlist file.
[-sim_tb]	Y	The optional -pnr_netlist option is used to specify the source file to be moved is a simulation testbench file.

move_relative_paths

```
move_relative_paths <line> <source> <target>
```

Change relative paths in input string; starting out relative to parameter source ,returned as relative to parameter target.

Argument	Optional	Description
<line>		line containing a quoted string of paths.
<source>		The location of the original script containing <line>
<target>		The script that will contain <line> and still work.

optimize_tile

```
optimize_tile [-verbose] [-timing <int>] [-optimize_fixed]
```

optimize placement of single tile or tiles seperately

Argument	Optional	Description
[-verbose]	Y	use verbose option reporting
[-timing <int>]	Y	timing-mode: 1 = timing-driven

Argument	Optional	Description
<code>[-optimize_fixed]</code>	Y	change placement of fixed instances - use with caution !

redirect

```
redirect <command> [-variable <string>] [-noprint] [-printinfo]
```

redirect messages to variable.

Argument	Optional	Description
<code><command></code>		command to run
<code>[-variable <string>]</code>	Y	name of variable
<code>[-noprint]</code>	Y	just add to variable, do not print
<code>[-printinfo]</code>	Y	just print info messages

refresh_drawing

```
refresh_drawing
```

This command refreshes the current custom drawing on the GUI's Floorplanner view.

regenerate_all_ip_design_files

```
regenerate_all_ip_design_files [-ioringOutputDir <string>] [-add_to_project]
```

This command re-generates design files the all ACXIP files in the active ACE project, for both Core fabric IP and IO Ring design.

Argument	Optional	Description
<code>[-ioringOutputDir <string>]</code>	Y	The <code>-ioringOutputDir</code> option allows the user to specify a non-default directory path to output the IO Ring design files into.
<code>[-add_to_project]</code>	Y	Using the <code>-add_to_project</code> option automatically adds the required generated IO Ring design files to your ACE project. This includes utilization XML, SDC constraints, PDC constraints, and IO Ring bitstream files.

remap_partial_bitstream

```
remap_partial_bitstream <hex_file> <original_clusters> <new_clusters> [-  
output_file <string>]
```

This command returns a list of compatible ordering codes for the active project based on it's device, package, and speed grade.

Argument	Optional	Description
<hex_file>		The path to the bitstream hex file.
<original_clusters>		Provide a list of clusters from your original design (i.e. {{LOGIC_CLUSTER_1_1} {LOGIC_CLUSTER_1_2} {LOGIC_CLUSTER_2_1} {LOGIC_CLUSTER_2_2}}).
<new_clusters>		Provide a list of clusters from your partially reconfigured design (i.e. {{LOGIC_CLUSTER_6_4} {LOGIC_CLUSTER_6_5} {LOGIC_CLUSTER_7_4} {LOGIC_CLUSTER_7_5}}).
[-output_file <string>]	Y	Optional output file. If no output file is specified, the command outputs the file to <original_hex_file>_remapped.hex

remove_clock_preroute

```
remove_clock_preroute <net_name> <track_list> [-clock_regions <list>] [-clusters  
<list>] [-placement_regions <list>] [-partitions <list>]
```

This command will remove the pre-routing constraints from a clock or reset net on the clock tracks and regions specified.

Argument	Optional	Description
<net_name>		The name of the clock or reset net to remove from pre-routing.
<track_list>		The list of integer clock track numbers to remove from pre-route on this net. Valid clock track numbers are device-specific.
[-clock_regions <list>]	Y	The list of clock region names to remove from pre-route on this net. Valid clock region names are device-specific.
[-clusters <list>]	Y	The list of cluster names to remove from pre-route on this net. Valid cluster names are device-specific.

Argument	Optional	Description
<code>[-placement_regions <list>]</code>	Y	The list of placement region names to remove from pre-route on this net. Valid placement region names are device-specific.
<code>[-partitions <list>]</code>	Y	The list of partition names to remove from pre-route on this net. Valid partition are device-specific.

remove_flow_step

```
remove_flow_step <id>
```

This command removes an existing flow step from ACE only if the flow step is a user-defined flow step.

Argument	Optional	Description
<code><id></code>		The required <code><id></code> argument specifies the id of the flow step to remove.

remove_impl

```
remove_impl <implNames_list> [-project <string>]
```

This command removes multiple implementations from a project. The implementations' output directories on the file system are not deleted.

Argument	Optional	Description
<code><implNames_list></code>		The required <code><implNames_list></code> argument is used to specify the names of the implementations to remove.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) for the implementation to be removed from.

remove_path

```
remove_path [<id>] [-all]
```

This command removes a user-defined pin path.

Argument	Optional	Description
<code>[<id>]</code>	Y	Specifies the id of the path to remove

Argument	Optional	Description
[-all]	Y	Removes all paths

remove_project

```
remove_project <projectName>
```

This command removes a project from ACE. The project file on disk is not deleted.

Argument	Optional	Description
<projectName>		The required <projectName> argument is used to specify the project to be removed (by name).

remove_project_constraints

```
remove_project_constraints <files> [-project <string>]
```

This command removes the link to an SDC, PDC, or TCL constraint file from a project. The SDC constraint file on disk is not deleted.

Argument	Optional	Description
<files>		The required <files> argument is used to specify the SDC, PDC, or TCL constraint files (by file path).
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the SDC constraint file to be removed from.

remove_project_constraints_pvt

```
remove_project_constraints_pvt <file>
```

This command allows the user to remove all PVT conditions from an SDC constraint file.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the SDC constraint file.

remove_project_ip

```
remove_project_ip <list_of_files> [-project <string>]
```

This command removes the association from a project to one or more IP settings files. The IP settings files on disk are not deleted.

Argument	Optional	Description
<list_of_files>		The required <list_of_files> argument is used to specify the IP settings files (by file path). The file paths may be absolute, or may be relative to the acxprj file's directory.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the IP settings file to be removed from. The named project must already be opened in ACE.

remove_project_netlist

```
remove_project_netlist <files> [-project <string>] [-impl <string>]
```

This command removes the link to a verilog netlist file from a project. The verilog netlist file on disk is not deleted.

Argument	Optional	Description
<files>		The required <file> argument is used to specify the verilog netlists (by file path).
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the verilog netlist to be removed from.
[-impl <string>]	Y	If a file belongs to an impl and should be removed only from that impl, specify that impl here.

remove_project_source_files

```
remove_project_source_files <list_of_files> [-ip] [-rtl] [-syn_constraint] [-pnr_constraint] [-pnr_netlist] [-sim_tb] [-project <string>]
```

This command removes the link to source files from a project. The source files on disk is not deleted.

Argument	Optional	Description
<list_of_files>		The required <list_of_files> argument is used to specify a list of source files (by file path).

Argument	Optional	Description
[-ip]	Y	The optional -ip option is used to specify the source file to be removed is an IP settings file.
[-rtl]	Y	The optional -rtl option is used to specify the source file to be removed is an RTL file.
[-syn_constraint]	Y	The optional -syn_constraint option is used to specify the source file to be removed is a synthesis constraint file.
[-pnr_constraint]	Y	The optional -pnr_constraint option is used to specify the source file to be removed is a place and route constraint file.
[-pnr_netlist]	Y	The optional -pnr_netlist option is used to specify the source file to be removed is a place and route netlist file.
[-sim_tb]	Y	The optional -pnr_netlist option is used to specify the source file to be removed is a simulation testbench file.
[-project <string>]	Y	The optional -project <projectName> option is used to specify an alternate project (by name) for the RTL file to be removed from.

remove_region

```
remove_region [-region <string>] [-all]
```

This command removes a placement region constraint specification

Argument	Optional	Description
[-region <string>]	Y	Name of the region to delete
[-all]	Y	Remove all region constraints

remove_region_insts

```
remove_region_insts <region> [-insts <list>] [-all] [-flops_only] [-clocks_only] [-verbose]
```

Remove user design instances from an existing placement region constraint

Argument	Optional	Description
<region>		name of the region to clear
[-insts <list>]	Y	List of user design instances to remove from this placement region constraint.
[-all]	Y	Clear all user design instances from this region constraint
[-flops_only]	Y	When removing instances, filter out all instances except flops
[-clocks_only]	Y	When removing instances, filter out all instances with no connected clock
[-verbose]	Y	Print additional debug messages.

rename_impl

```
rename_impl <newImplName> [-project <string>] [-impl <string>]
```

This command renames an implementation. Changing the name of an implementation also changes the name of the implementation output directory on disk (even without calling "save_project").

Argument	Optional	Description
<newImplName>		The required <newImplName> argument is used to specify the new implementation name.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to change the name for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to change the name for.

report_clock_regions

```
report_clock_regions [-outputfile <string>] [-text] [-csv]
```

This command generates and writes a formatted report showing which clock nets are routed in each clock region

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The <code>-outputfile <file></code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation debug directory and is named <code><design_name>_regions.html</code> .
<code>[-text]</code>	Y	The <code>-text</code> option is used to specify whether the file should be output to the console as plain text
<code>[-csv]</code>	Y	The <code>-csv</code> option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets

report_clocks

`report_clocks`

Report clocks in the current design

report_coverage

`report_coverage [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>] [-verbose]`

Generate and write a coverage report for pins.

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The optional <code>-outputfile <file></code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <code><design_name>_pins.html</code> .
<code>[-text]</code>	Y	The optional <code>-text</code> option is used to specify whether the file should be output as plain text.
<code>[-csv]</code>	Y	The optional <code>-csv</code> option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
<code>[-html]</code>	Y	The optional <code>-html</code> option is used to specify whether the file should be output as html-file.
<code>[-columns <list>]</code>	Y	The optional <code>-columns</code> option is used to specify a customized ordered list of columns names to output in the pin assignment report.

Argument	Optional	Description
<code>[-verbose]</code>	Y	The optional <code>-verbose</code> option is used to specify whether the file should report ALL attributes and parameters for each IO port/pad.

report_design_stats

```
report_design_stats [-outputfile <list>] [-html] [-csv] [-text]
```

This command generates and writes a formatted report about various design statistics

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The <code>-outputfile <file></code> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the <code>-text</code> , <code>-html</code> , and <code>-csv</code> options. If <code>-text</code> is given the output is written to the GUI console and Ace logfile. If <code>-html</code> or <code>-csv</code> is given, the output is written to the default implementation reports directory in a file named <code><design_name>_design_stats</code> , with the extension <code>.html</code> or <code>.csv</code> (respectively).
<code>[-html]</code>	Y	The <code>-html</code> option specifies that the output file(s) are written in HTML format (this is the default)
<code>[-csv]</code>	Y	The <code>-csv</code> option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
<code>[-text]</code>	Y	The <code>-text</code> option specifies that the output file(s) are written in plain text format

report_impl_options

```
report_impl_options [-outputfile <string>] [-text] [-csv] [-project <string>] [-impl <string>] [-hide_values] [-show_standard] [-show_project_options] [-diff_options]
```

Output a report of the current impl options defined in ACE. If no `-project` and/or `-impl` options are specified, the active impl will be reported.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile</code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <code><design_name>_impl_options.html</code> .
<code>[-text]</code>	Y	The optional <code>-text</code> option is used to specify that the file should be output as plain text.
<code>[-csv]</code>	Y	The optional <code>-csv</code> option is used to specify that the file should be output as a CSV file for use in Excel spreadsheets.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option must be used with <code>-impl <implName></code> . These options are used to specify an alternate project implementation (by name).
<code>[-impl <string>]</code>	Y	The optional <code>-impl <implName></code> option can be used with or without the <code>-project <projectName></code> option to specify an impl. Without <code>-project <projectName></code> , <code>-impl <implName></code> finds the impl in the active project.
<code>[-hide_values]</code>	Y	If the <code>-hide_values</code> flag is used, the report will NOT output an additional column to display the current values for the given impl.
<code>[-show_standard]</code>	Y	If the <code>-show_standard</code> flag is used, the report and optional diff options Tcl file will ONLY output standard options that show in the GUI. If this flag is not set, by default, all available options for the active impl will be output.
<code>[-show_project_options]</code>	Y	If the <code>-show_project_options</code> flag is used, the options report table will contain an additional column specifying the option type.
<code>[-diff_options]</code>	Y	The optional <code>-diff_options</code> flag is used to create a Tcl file containing the full set of implementation options with values which vary from the default values. Each impl option's default value is listed as a comment.

report_partitions

```
report_partitions [-outputfile <list>] [-html] [-csv] [-text]
```

This commands generates and writes a formatted partition report

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The <code>-outputfile <file></code> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the <code>-text</code> , <code>-html</code> , and <code>-csv</code> options. If <code>-text</code> is given the output is written to the GUI console and Ace logfile. If <code>-html</code> or <code>-csv</code> is given, the output is written to the default implementation reports directory in a file named <code><design_name>_partitions</code> , with the extension <code>.html</code> or <code>.csv</code> (respectively).
<code>[-html]</code>	Y	The <code>-html</code> option specifies that the output file(s) are written in HTML format (this is the default)
<code>[-csv]</code>	Y	The <code>-csv</code> option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
<code>[-text]</code>	Y	The <code>-text</code> option specifies that the output file(s) are written in plain text format

report_performance

```
report_performance [-outputfile <list>] [-html] [-csv] [-text] [<num_paths>] [-setup_pass_only]
```

Generate a report detailing the estimated design performance and the quality of the mapping in ACE, including a critical path breakdown

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The <code>-outputfile <file></code> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the <code>-text</code> , <code>-html</code> , and <code>-csv</code> options. If <code>-text</code> is given the output is written to the GUI console and Ace logfile. If <code>-html</code> or <code>-csv</code> is given, the output is written to the default implementation reports directory in a file named <code><design_name>_performance_report</code> , with the extension <code>.html</code> or <code>.csv</code> (respectively).
<code>[-html]</code>	Y	The <code>-html</code> option specifies that the output file(s) are written in HTML format (this is the default)
<code>[-csv]</code>	Y	The <code>-csv</code> option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
<code>[-text]</code>	Y	The <code>-text</code> option specifies that the output file(s) are written in plain text format

Argument	Optional	Description
[<num_paths>]	Y	The number of critical paths to analyze in each clock domain (default is 3)
[-setup_pass_only]	Y	If set, only paths that meet timing are logged

report_pins

```
report_pins [-outputfile <list>] [-text] [-csv] [-html] [-columns <list>]
```

Generate and write a pin to package assignment report

Argument	Optional	Description
[-outputfile <list>]	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <design_name>_pins.html.
[-text]	Y	The optional -text option is used to specify whether the file should be output as plain text.
[-csv]	Y	The optional -csv option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
[-html]	Y	The optional -html option is used to specify whether the file should be output as html-file.
[-columns <list>]	Y	The optional -columns option is used to specify a customized ordered list of columns names to output in the pin assignment report.

report_placement

```
report_placement [-outputfile <list>] [-html] [-csv] [-text]
```

This command generates and writes a formatted placement QoR report

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The <code>-outputfile <file></code> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the <code>-text</code> , <code>-html</code> , and <code>-csv</code> options. If <code>-text</code> is given the output is written to the GUI console and Ace logfile. If <code>-html</code> or <code>-csv</code> is given, the output is written to the default implementation reports directory in a file named <code><design_name>_placement</code> , with the extension <code>.html</code> or <code>.csv</code> (respectively).
<code>[-html]</code>	Y	The <code>-html</code> option specifies that the output file(s) are written in HTML format (this is the default)
<code>[-csv]</code>	Y	The <code>-csv</code> option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
<code>[-text]</code>	Y	The <code>-text</code> option specifies that the output file(s) are written in plain text format

report_power

```
report_power [-outputfile <list>] [-html] [-csv] [-text] [-temperature <string>]
[-clocks <string>] [-achieved] [-saif_file <string>] [-saif_top_level <string>]
```

This command generates and writes a formatted power dissipation report

Argument	Optional	Description
<code>[-outputfile <list>]</code>	Y	The <code>-outputfile <file></code> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the <code>-text</code> , <code>-html</code> , and <code>-csv</code> options. If <code>-text</code> is given the output is written to the GUI console and Ace logfile. If <code>-html</code> or <code>-csv</code> is given, the output is written to the default implementation reports directory in a file named <code><design_name>_power</code> , with the extension <code>.html</code> or <code>.csv</code> (respectively).
<code>[-html]</code>	Y	The <code>-html</code> option specifies that the output file(s) are written in HTML format (this is the default)
<code>[-csv]</code>	Y	The <code>-csv</code> option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
<code>[-text]</code>	Y	The <code>-text</code> option specifies that the output file(s) are written in plain text format

Argument	Optional	Description
<code>[-temperature <string>]</code>	Y	Report power at the given junction temperature (in degrees C) instead of the currently specified operating condition
<code>[-clocks <string>]</code>	Y	The <code>-clocks <{ clk1 freq } { clk2 freq } .. { clkN freq } ></code> option may be used to specify the operating frequency (in MHz) of all the clocks in the design. If this option is not present, the frequencies from the design constraints will be used by default. If no constraints are found, the best achieved frequency will be used.
<code>[-achieved]</code>	Y	The <code>-achieved</code> option may be used to specify that achieved static timing results be used to calculate the power dissipation report for each clock
<code>[-saif_file <string>]</code>	Y	The <code>-saif_file</code> option may be used to specify the path to a .saif file. Switching Activity Interchange Format (saif) files can be used to provide a more accurate power profile of a design.
<code>[-saif_top_level <string>]</code>	Y	The <code>-saif_top_level</code> option may be used to specify the name of the top level instance in the saif file. Default name used is "DUT".

report_project_options

```
report_project_options [-outputfile <string>] [-text] [-csv] [-project <string>]
[-hide_values] [-show_standard] [-diff_options]
```

Output a report of the current Project options defined in ACE. If no `-project` option is specified, the active project will be reported.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile</code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation reports directory and is named <code><design_name>_impl_options.html</code> .
<code>[-text]</code>	Y	The optional <code>-text</code> option is used to specify that the file should be output as plain text.
<code>[-csv]</code>	Y	The optional <code>-csv</code> option is used to specify that the file should be output as a CSV file for use in Excel spreadsheets.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name).
<code>[-hide_values]</code>	Y	If the <code>-hide_values</code> flag is used, the report will NOT output an additional column to display the current values for the given project.
<code>[-show_standard]</code>	Y	If the <code>-show_standard</code> flag is used, the report and optional diff options Tcl file will ONLY output standard options that show in the GUI. If this flag is not set, by default, all available options for the active impl will be output.
<code>[-diff_options]</code>	Y	The optional <code>-diff_options</code> flag is used to create a Tcl file containing the full set of project options with values which vary from the default values. Each project option's default value is listed as a comment.

report_routing

`report_routing [-outfile <list>] [-text] [-csv] [-html] [-nonterse] [-terse] [-wL] [-overflowreportlimit <int>]`

This command generates and writes a formatted routing report.

Argument	Optional	Description
<code>[-outfile <list>]</code>	Y	The optional <code>-outfile <file></code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation debug directory and is named <code><design_name>_routing.html</code> .
<code>[-text]</code>	Y	The optional <code>-text</code> option is used to specify whether the file should be output as plain text (Default is autodetect)
<code>[-csv]</code>	Y	The optional <code>-csv</code> option is used to specify whether the file should be output as a CSV file for use in Excel spreadsheets.
<code>[-html]</code>	Y	The optional <code>-html</code> option is used to specify whether the file should be output as html-file.
<code>[-nonterse]</code>	Y	normal information level.
<code>[-terse]</code>	Y	terse info level.

Argument	Optional	Description
[-wl]	Y	wire length report.
[-overflowreportlimit <int>]	Y	limit on the number of overflows. Defaults to 11.

report_utilization

report_utilization [-outputfile <list>] [-html] [-csv] [-text]

This command generates and writes a formatted device utilization report

Argument	Optional	Description
[-outputfile <list>]	Y	The -outputfile <file> option specifies a Tcl list of one or more output file names or file path names. If this option is not present, the output depends on the -text, -html, and -csv options. If -text is given the output is written to the GUI console and Ace logfile. If -html or -csv is given, the output is written to the default implementation reports directory in a file named <design_name>_utilization, with the extension .html or .csv (respectively).
[-html]	Y	The -html option specifies that the output file(s) are written in HTML format (this is the default)
[-csv]	Y	The -csv option specifies that the output file(s) are written in CSV format for import into Excel spreadsheets
[-text]	Y	The -text option specifies that the output file(s) are written in plain text format

reset_impl_option

reset_impl_option [<option_name>] [-all] [-project <string>] [-impl <string>]

This command resets a project implementation option to its (device-specific) default value. Only one option may be reset at a time.

Argument	Optional	Description
[<option_name>]	Y	The name of the impl option to reset to its default. To see a list of valid impl options, use the report_impl_options TCL command.

Argument	Optional	Description
<code>[-all]</code>	Y	The optional <code>-all</code> option resets all impl options to their device-specific default values.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to set options for.
<code>[-impl <string>]</code>	Y	The optional <code>-project <projectName></code> and <code>-impl <implName></code> options are used to specify an alternate project implementation (by name) to set options for.

reset_project_option

```
reset_project_option [<option_name>] [-all] [-project <string>]
```

This command resets a project option to its (device-specific) default value. Only one option may be reset at a time.

Argument	Optional	Description
<code>[<option_name>]</code>	Y	The name of the project option to reset to its default. To see a list of valid project options, use the <code>report_project_options</code> TCL command.
<code>[-all]</code>	Y	The optional <code>-all</code> option resets all project options to their device-specific default values.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) to set options for.

restore_impl

```
restore_impl <filename> [-project <string>] [-impl <string>]
```

The `restore_impl` command loads an ACXDB file to restore the state of the DB and impl options for a given impl. Restoring an impl automatically makes that impl the active impl. If no `-impl` and `-project` options are specified, then the ACXDB file is loaded for the current active impl. By default, the DB will be restored using all saved information in the ACXDB file, including placement and routing information. Restoring an impl overrides the current impl option values with the impl option values saved in the ACXDB file. Restoring an impl clears the current state of the DB for the current active impl, so be sure to save your active impl before restoring an impl.

Argument	Optional	Description
<code><filename></code>		Specifies the ACXDB file path to restore the state of the active impl from

Argument	Optional	Description
<code>[-project <string>]</code>	Y	Specifies an alternate project name to use instead of the active project when using the <code>-impl <implname></code> option
<code>[-impl <string>]</code>	Y	Specifies an alternate impl name to restore instead of restoring the current active impl

This functionality is also accessible through buttons/menus in the ACE GUI as described in [Restoring Implementations \(page 302\)](#).

Restoring an Implementation clears current data

Restoring an [Implementation \(page 229\)](#) will first clear all data in memory before beginning the restore process. Any data that has not been saved will be lost.

The restored implementation (and project) will become the Active Project and Active Implementation, and all Implementation Options will also be restored from file, overwriting any values currently in memory.

See also: [save_impl \(page 705\)](#) and [Saving Implementations \(page 302\)](#).

restore_project

```
restore_project <projectFile> [-reload] [-not_active] [-no_db] [-activeimpl
<string>] [-acxdb <string>] [-force]
```

The `restore_project` command loads an ACE project (`.acxprj`) file and restores the project's implementation options. The `-acxdb` option can be used to specify the file path of the ACXDB file to restore the DB state for the active implementation. The `-activeimpl` option can be used to specify the impl name to activate and restore. If `-not_active` is used, no impl in the project will be activated or restored from its ACXDB file and the active impl in ACE will not be changed.

Argument	Optional	Description
<code><projectFile></code>		Specifies the ACE project (<code>.acxprj</code>) file path to load and restore.
<code>[-reload]</code>	Y	Use this option to re-load the ACE project (<code>.acxprj</code>) file from disk. This will clear the design DB and flow status, requiring the design to be re-run from the beginning of the flow.
<code>[-not_active]</code>	Y	If this option is set, no impl in the project will be activated or restored from its ACXDB file and the active impl in ACE will not be changed.
<code>[-no_db]</code>	Y	If this option is set, the DB state of the active impl for the project will not be restored.

Argument	Optional	Description
<code>[-activeimpl <string>]</code>	Y	The <code>-activeimpl</code> option can be used to specify an alternate impl name to activate and restore. By default, the last active impl during the session the project was saved in will be activated.
<code>[-acxdb <string>]</code>	Y	Specifies an ACXDB file path from which to restore the state of the active impl.
<code>[-force]</code>	Y	The <code>-force</code> option can be used to override a project lock that has been set by another ACE session. Using <code>-force</code> causes the current ACE session to take ownership of the project lock for the project being restored. DO NOT use this option to run multiple ACE sessions on the same project at the same time, or else output files (<code>acxprj</code> , <code>acxdb</code> , <code>icdb</code> , <code>jam</code> , etc) and log files may become corrupted!

run

```
run [-step <string>] [-stop_at_step <string>] [-resume] [-ic <string>]
```

This command runs the steps of the design flow. It can be used to run the entire flow from the beginning, run a specific flow step, or resume the flow from the last incomplete step. Using no options will run the entire flow from the beginning. The default Achronix flow step IDs (for those options requiring them) are: {prepare run_prepare report_timing_prepared write_netlist_prepared place_and_route run_place report_timing_placed run_route report_timing_routed design_completion post_process final_drc_checks report_timing_final write_netlist_final fpga_program write_bitstream fpga_download} Because advanced users may create their own flow steps, a complete list of all flow step IDs can be retrieved with the Tcl command 'get_flow_steps'.

Argument	Optional	Description
<code>[-step <string>]</code>	Y	The optional <code>-step <id></code> option is used to run the specified flow step, by ID, (along with any incomplete pre-requisite steps,) and all of its children. See 'get_flow_steps' for a list of all IDs.
<code>[-stop_at_step <string>]</code>	Y	The optional <code>-stop_at_step <id></code> option is used to stop the flow after running the specified flow step, by ID. See 'get_flow_steps' for a list of all IDs.
<code>[-resume]</code>	Y	The optional <code>-resume</code> option is used to run the entire flow from the last successfully completed flow step.

Argument	Optional	Description
<code>[-ic <string>]</code>	Y	The optional <code>-ic init continue</code> option specifies incremental compilation flow modes. 'init' implies beginning of the flow without using previous state of compiled design and 'continue' implies incrementally compiling of previous state of design.

run_fanout_control

```
run_fanout_control [-physical <int>] [-fanout_limit <int>] [-fanout_limit_clone <int>]
```

This command does fanout control for high fanout control nets

Argument	Optional	Description
<code>[-physical <int>]</code>	Y	Clone Critical Instances that have slack lower than this limit
<code>[-fanout_limit <int>]</code>	Y	Apply fanout control on nets with fanout greater than this limit
<code>[-fanout_limit_clone <int>]</code>	Y	Apply fanout cloning on nets with fanout greater than this limit

run_final_drc_checks

```
run_final_drc_checks
```

This command performs final DRC checks on the active design. If there is currently no active project/implementation, the `reportsdir` and `debugdir` must be specified.

run_fpga_download

```
run_fpga_download [-hex_file <string>]
```

This command downloads the generated bitstream to the target device for the active implementation.

Argument	Optional	Description
<code>[-hex_file <string>]</code>	Y	Optional hex file to download. The default hex file in your output directory will be used if this is not specified

run_generate_bitstream

```
run_generate_bitstream [-outputdir <string>] [-aeskey <string>]
```

This command generates a bitstream file for programming the target device.

Argument	Optional	Description
[-outputdir <string>]	Y	Output directory name
[-aeskey <string>]	Y	Key used for encryption. If not given, key is taken from impl. If not active in impl, the bitstream is not encrypted.

run_generate_final_reports

```
run_generate_final_reports [-name_postfix <string>] [-format <string>]
```

This command generates various report files, including clocks, pins, power, etc. Implementation options are used to control which report files are generated.

Argument	Optional	Description
[-name_postfix <string>]	Y	Postfix added to report file name (e.g., to distinguish multiple 'placed' reports)
[-format <string>]	Y	Specify report formats; default is { text html csv }

run_generate_fullchip_sim

```
run_generate_fullchip_sim [-debugdir <string>] [-modelsdir <string>]
```

This command generates the files necessary for fullchip simulation.

Argument	Optional	Description
[-debugdir <string>]	Y	The -debugdir <dir> option is used to override the default location for debug files during this step.
[-modelsdir <string>]	Y	The -modelsdir <dir> option is used to override the default location for the fullchip sim top-level models.

run_generate_netlist

```
run_generate_netlist [-outputfile <string>] [-final] [-compress]
```

This command generates a verilog netlist for simulation.

Argument	Optional	Description
[-outputfile <string>]	Y	Output netlist file name.
[-final]	Y	Output DRC-free final netlist
[-compress]	Y	Compress output file with gzip

run_insert_holdbuffers

```
run_insert_holdbuffers [-margin <int>] [-io_buffers <int>] [-typebased_buffers <int>]
```

This command generates extra gate delays by inserting a buffer per target pin if that pin has a hold time slack value that is less than the margin specified.

Argument	Optional	Description
[-margin <int>]	Y	Insert a delay buffer per target pin whose hold time slack is less than the specified value
[-io_buffers <int>]	Y	Insert a delay buffer per target pin when driven directly by a flopped IO Pad/Pin
[-typebased_buffers <int>]	Y	Insert a delay buffer per target pin according to the given cell-type bitfield specification (2 ⁰ = DFF inputs, 2 ¹ = DFF outputs, 2 ² = clocked OPIN inputs, 2 ³ = clocked IPIN outputs, 2 ⁴ =BRAM inputs, 2 ⁵ = BRAM outputs, 2 ⁶ = LRAM inputs, 2 ⁷ = LRAM outputs, 2 ⁸ = DSP inputs, 2 ⁹ = DSP outputs, 2 ¹⁰ = MLP inputs, 2 ¹¹ = MLP outputs, 2 ¹² = NAP inputs, 2 ¹³ = NAP outputs). All other values are reserved for future use and should be unset.)

run_multiprocess

```
run_multiprocess [-use_existing_impls <list>] [-use_seeds <list>] [-parallel_job_count <int>] [-use_job_submission <int>] [-stop_flow_at <string>] [-copy_icdb <int>] [-jobs_exec <string>] [-jobs_wd <string>] [-jobs_name <string>] [-jobs_log <string>] [-jobs_args <list>] [-jobs_nfs_latency <int>] [-create_option_sets] [-exclude_synthesis] [-remove_nonbest]
```

This command runs the ACE multiprocess flow for the active implementation. To generate new implementations from option sets and run multiprocess, use -create_option_sets. NOTE: For any optional arguments that are not specified, the current Multiprocess configuration from the ACE GUI User Preferences will be used as defaults.

Argument	Optional	Description
<code>[-use_existing_impls <list>]</code>	Y	The <code>use_existing_impls</code> option allows the user to specify a list of existing impl names to run in multiprocess, instead of using seed sweep or generating impls from option sets. To run all existing impls, you can specify <code>-use_existing_impls [ace::get_impl_names]</code>
<code>[-use_seeds <list>]</code>	Y	The <code>use_seeds</code> option allows the user to specify a list of PnR seed values to run in multiprocess, instead of using existing impls or generating impls from option sets.
<code>[-parallel_job_count <int>]</code>	Y	(optional) Sets the number of implementations to run in parallel. If not specified, defaults to the GUI's preference setting.
<code>[-use_job_submission <int>]</code>	Y	(optional) Set to a 0 to run background jobs on the local machine, or set to a 1 to submit jobs to a cloud/grid/batch submission system. If not specified, defaults to the GUI's preference setting.
<code>[-stop_flow_at <string>]</code>	Y	(optional) If a valid flow step ID is specified, that flow step will be enabled and will be the last flow step executed by all implementations. If not specified, ACE will run the entire flow (ignores user's GUI preference setting).
<code>[-copy_icdb <int>]</code>	Y	(optional) Set to a 1 to copy the incremental flow DB from the template impl, or set to a 0 to not copy. Defaults to 0 (no file copy) if not specified (ignores user's GUI preference setting).
<code>[-jobs_exec <string>]</code>	Y	(optional) Specify the job submission system executable. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_wd <string>]</code>	Y	(optional) Specify the job submission system working directory argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_name <string>]</code>	Y	(optional) Specify the job submission system job name argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_log <string>]</code>	Y	(optional) Specify the job submission system job log argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.

Argument	Optional	Description
<code>[-jobs_args <list>]</code>	Y	(optional) Specify the job submission system list of additional arguments, each with at most one optional value, formatted as a list of TCL lists in the form: <code>{{arg1 val1} {arg2} {arg3 val3} ... }</code> . Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_nfs_latency <int>]</code>	Y	(optional) Specify the allowed seconds of NFS write latency (how long to wait between process completion and reading the files generated by the just-finished process). This is relevant both to local background jobs, and job submission systems. If not specified, defaults to the GUI's preference setting.
<code>[-create_option_sets]</code>	Y	(optional) Auto-generate option sets relevant to the active implementation, which will appear the active impl's <code>option_sets</code> directory. Note that this is only relevant when neither <code>-use_existing_impls</code> nor <code>-use_seeds</code> are active (meaning we're in the default mode, generating new impls for option sets).
<code>[-exclude_synthesis]</code>	Y	(optional) Auto-generate option sets excluding synthesis options relevant to the active implementation, which will appear the active impl's <code>option_sets</code> directory. This option is ignored if 'run_synthesis' flow step is disabled. Note that this is only relevant when neither <code>-use_existing_impls</code> nor <code>-use_seeds</code> are active (meaning we're in the default mode, generating new impls for option sets).
<code>[-remove_nonbest]</code>	Y	(optional) Removes all recently generated implementations from the Projects View, except for the base impl and best impl, at the end of the Multiprocess run.

Default Values

For any parameters that are not specified, the ACE GUI user preferences will be used.

For example, if `-parallel_job_count` is not explicitly specified, and if your ACE GUI user preferences are currently configured to use four (4) jobs, that value will be used for your batch multiprocess run.

A more detailed description of the use of `run_multiprocess` can be found in the [Multiprocess Batch Mode \(page 318\)](#) section.

The GUI provides a graphical interface for multiprocess through the [Multiprocess View \(page 73\)](#). See also: [Running Multiple Flows in Parallel \(page 308\)](#), [Attempting Likely Optimizations Using Option Sets \(page 392\)](#).

run_multiprocess_iterator

```
run_multiprocess_iterator [-use_existing_impls <list>] [-use_seeds <list>] [-parallel_job_count <int>] [-use_job_submission <int>] [-stop_flow_at <string>] [-copy_icdb <int>] [-jobs_exec <string>] [-jobs_wd <string>] [-jobs_name <string>] [-jobs_log <string>] [-jobs_args <list>] [-jobs_nfs_latency <int>] [-create_option_sets] [-exclude_synthesis] [-remove_nonbest] [-iterations <int>]
```

This command runs the ACE multiprocess flow for the active implementation for 5 (default) iterations, each time selecting the 'best' impl option for the subsequent multiprocess run. To run the same multiprocess flow as run_multiprocess, this command is exposed to all options from run_multiprocess. To run multiprocess with option sets, use -create_option_sets.

Argument	Optional	Description
[-use_existing_impls <list>]	Y	The use_existing_impls option allows the user to specify a list of existing impl names to run in multiprocess, instead of using seed sweep or generating impls from option sets. To run all existing impls, you can specify -use_existing_impls [ace::get_impl_names]
[-use_seeds <list>]	Y	The use_seeds option allows the user to specify a list of PnR seed values to run in multiprocess, instead of using existing impls or generating impls from option sets.
[-parallel_job_count <int>]	Y	(optional) Sets the number of implementations to run in parallel. If not specified, defaults to the GUI's preference setting.
[-use_job_submission <int>]	Y	(optional) Set to a 0 to run background jobs on the local machine, or set to a 1 to submit jobs to a cloud/grid/batch submission system. If not specified, defaults to the GUI's preference setting.
[-stop_flow_at <string>]	Y	(optional) If a valid flow step ID is specified, that flow step will be enabled and will be the last flow step executed by all implementations. If not specified, ACE will run the entire flow (ignores user's GUI preference setting).
[-copy_icdb <int>]	Y	(optional) Set to a 1 to copy the incremental flow DB from the template impl, or set to a 0 to not copy. Defaults to 0 (no file copy) if not specified (ignores user's GUI preference setting).
[-jobs_exec <string>]	Y	(optional) Specify the job submission system executable. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.

Argument	Optional	Description
<code>[-jobs_wd <string>]</code>	Y	(optional) Specify the job submission system working directory argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_name <string>]</code>	Y	(optional) Specify the job submission system job name argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_log <string>]</code>	Y	(optional) Specify the job submission system job log argument. Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_args <list>]</code>	Y	(optional) Specify the job submission system list of additional arguments, each with at most one optional value, formatted as a list of TCL lists in the form: <code>{{arg1 val1} {arg2} {arg3 val3} ... }</code> . Relevant only when a job submission system is in use. If not specified, defaults to the GUI's preference setting.
<code>[-jobs_nfs_latency <int>]</code>	Y	(optional) Specify the allowed seconds of NFS write latency (how long to wait between process completion and reading the files generated by the just-finished process). This is relevant both to local background jobs, and job submission systems. If not specified, defaults to the GUI's preference setting.
<code>[-create_option_sets]</code>	Y	(optional) Auto-generate option sets relevant to the active implementation, which will appear the active impl's <code>option_sets</code> directory. Note that this is only relevant when neither <code>-use_existing_impls</code> nor <code>-use_seeds</code> are active (meaning we're in the default mode, generating new impls for option sets).
<code>[-exclude_synthesis]</code>	Y	(optional) Auto-generate option sets excluding synthesis options relevant to the active implementation, which will appear the active impl's <code>option_sets</code> directory. This option is ignored if 'run_synthesis' flow step is disabled. Note that this is only relevant when neither <code>-use_existing_impls</code> nor <code>-use_seeds</code> are active (meaning we're in the default mode, generating new impls for option sets).
<code>[-remove_nonbest]</code>	Y	(optional) Removes all recently generated implementations from the Projects View, except for the base impl and best impl, at the end of the Multiprocess run.

Argument	Optional	Description
<code>[-iterations <int>]</code>	Y	The iterations option allows the user to specify the number of iterations between 1 and 5. Default is 5.

⚠ Warning: Early Access Functionality

This multiprocessing iterator (along with the QoR sorting of the [Multiprocess Summary Report \(page 255\)](#)) should currently be considered early-access functionality. Future ACE releases may improve QoR, reduce runtimes, and require fewer iterations to achieve similar results.

See also: [run_multiprocess \(page 692\)](#), [Multiprocess View \(page 73\)](#), [Running Multiple Flows in Parallel \(page 308\)](#), [Attempting Likely Optimizations Using Option Sets \(page 392\)](#), [Multiprocess Batch Mode \(page 318\)](#)

run_place

`run_place`

This command clears all routing and places the design.

run_post_process

`run_post_process`

This command post-processes the routed design to insert reset and other Achronix-specific technologies.

run_prepare

`run_prepare [-ic <string>]`

This command clears the current netlist and constraints data, then loads all the design files for the active implementation, runs design checks, and compiles the design into an Achronix design.

Argument	Optional	Description
<code>[-ic <string>]</code>	Y	The optional <code>-ic init continue</code> option specifies incremental compilation flow modes. 'init' implies beginning the flow without using previous state of compiled design and 'continue' implies incrementally compiling of previous state of design.

run_route

`run_route`

This command routes the design.

run_secureshare

```
run_secureshare [-read_manifest <string>] [-generate_manifest <string>] [-archive_path <string>] [-encrypt] [-wizard] [-force]
```

Read or generate a SecureShare Manifest File, and create a zipped (and optionally encrypted) archive containing project source, database, log, report, and debug files. To receive better support, please attach this zip to your Achronix support ticket.

Argument	Optional	Description
<code>[-read_manifest <string>]</code>	Y	Manifest file path read to generate an archive (unless <code>-no_archive</code> is specified). Cannot be used with <code>-generate_manifest</code> .
<code>[-generate_manifest <string>]</code>	Y	Manifest file output path, and default archive output path (unless <code>-archive_path</code> is specified). Cannot be used with <code>-read_manifest</code> .
<code>[-archive_path <string>]</code>	Y	Archive output path, which can override the output directory within a manifest when using <code>-read_manifest</code>
<code>[-encrypt]</code>	Y	Encrypt the archive. With <code>-read_manifest</code> , the archive is always encrypted (unless <code>-no_archive</code> is specified). With <code>-generate_manifest</code> , the manifest is marked for encryption, and the generated archive is encrypted (unless <code>-no_archive</code> is specified).
<code>[-wizard]</code>	Y	Generate a manifest and then open the GUI file-picker.
<code>[-force]</code>	Y	Overwrite generated output files.

Why is this command taking so long?

In some cases, such as when the home directory is mounted to a network drive, reading the log files can be quite slow.

To skip this step run, add the following flag to the `run_secureshare` command:

Argument	Optional	Description
<code>[-skip_home_directory_logs]</code>	Y	When generating a manifest, do not search the home directory for log files.


Example

Running the command with no arguments will automatically generate a SecureShare manifest and archive using the current active impl:

```
run_secureshare
```


If you'd like to create the archive in a specific location, rather than the project default, use:

```
run_secureshare -archive_path <output_path>
```

 The file extension for a regular SecureShare archive will always be .zip

To encrypt the archive, use:

```
run_secureshare -encrypt
```

 The file extension for an encrypted SecureShare archive will always be .zip.encrypted

To generate an archive from a previously created manifest, use:

```
run_secureshare -read_manifest <manifest_path.acxssm>
```

This will read the .acxssm manifest file, which includes the archive output directory and encryption status, and generate an archive using these options.


You can override the manifest with the `-archive_path <output_path>` and `-encrypt` options.

If you'd like to automatically create a manifest without generating an archive, use:

```
run_secureshare -no_archive
```

To specify the manifest path, use:

```
run_secureshare -generate_manifest <manifest_path>
```

 The file extension for a SecureShare manifest will always be .acxssm

To overwrite manifest and archive outputs, as well as create missing directories, use:

```
run_secureshare -force
```

run_simulation

```
run_simulation [-rtl] [-gate] [-routed] [-final]
```

This command runs custom simulator commands to simulate user design.

Argument	Optional	Description
[-rtl]	Y	Run RTL Simulation (this is the default)
[-gate]	Y	Run Gate-level Netlist Simulation
[-routed]	Y	Run Post-Route Netlist Simulation
[-final]	Y	Run Final Netlist Simulation

run_snapshot

```
run_snapshot <snapshotFile> [-jtag_id <string>] [-ir_bits_before <int>] [-ir_bits_after <int>] [-target_offset <int>] [-startup_trigger] [-timeout <int>] [-cleanse_snapshot_file] [-verbose]
```

This command runs the Snapshot Debugger for a given Snapshot configuration (.snapshot file). This command outputs a VCD file and a Log file. The file paths are specified in the Snapshot configuration file.

Argument	Optional	Description
<snapshotFile>		The required <snapshotFile> argument specifies the Snapshot configuration (.snapshot file) to be used by Snapshot debugger.
[-jtag_id <string>]	Y	The unique ID of JTAG device to communicate with

Argument	Optional	Description
<code>[-ir_bits_before <int>]</code>	Y	Number of instruction register bits in the scan chain before the target device. Default: set to 0 for single device scan chains
<code>[-ir_bits_after <int>]</code>	Y	Number of instruction register bits in the scan chain after the target device. Default: set to 0 for single device scan chains
<code>[-target_offset <int>]</code>	Y	Target device offset in the scan chain. A value of 0 indicates the first device on the scan chain that receives TDI from the JTAG programmer device. Default: 0
<code>[-startup_trigger]</code>	Y	Run Snapshot Startup Trigger to capture the startup trigger in the Snapshot Debugger. Default: Snapshot Arm
<code>[-timeout <int>]</code>	Y	Specifies the timeout (in seconds) before Snapshot is cancelled while waiting for all trigger conditions to occur. If not specified, Snapshot will not timeout.
<code>[-cleanse_snapshot_file]</code>	Y	Cleanse the Snapshot (.snapshot) file content before beginning normal work. This is typically only needed when a *.snapshot file fails to load, and at most is needed once per file.
<code>[-verbose]</code>	Y	Exposes additional log info when running Snapshot.

See also: [Running the Snapshot Debugger \(page 364\)](#), *Snapshot User Guide (UG016)*

Cleansing the .snapshot file

This is normally not necessary, but if any *.snapshot file fails to load, try using this functionality once per file (cleansing a file a second time with the same version of ACE will have no effect). When enabled, this opens the file, performs a validate-cleanse-and-version-upgrade pass on the content, and re-saves the cleansed content to the same filename/path, overwriting the prior content. Cleansing occurs before beginning normal (arm or startup_trigger) Snapshot processing. Cleansing will only be necessary with files that fail to load successfully; these are typically files which have been hand-edited, or with files which are several years old. Note that cleansing adds 10s-30s of runtime, thus cleansing is disabled by default.

run_synthesis

```
run_synthesis
```

This command synthesizes the user design RTL using the specified synthesis constraints.

run_timing_analysis

```
run_timing_analysis [-prepared] [-placed] [-routed] [-final] [-name_postfix
<string>] [-format <string>] [-temperature <string>] [-voltage <string>] [-grade
<string>]
```

This command runs timing analysis on the design.

Argument	Optional	Description
[-prepared]	Y	Indicates that the design has only been prepared (this is the default)
[-placed]	Y	Indicates that the design has been placed but not routed
[-routed]	Y	Indicates that the design has been placed and routed
[-final]	Y	Indicates that this is sign-off timing (this involves some extra checks)
[-name_postfix <string>]	Y	Postfix added to report file name (e.g., to distinguish multiple 'placed' reports)
[-format <string>]	Y	Specify report formats; default is { text html csv }
[-temperature <string>]	Y	The temperature selection to do timing analysis at a PVT corner
[-voltage <string>]	Y	The voltage selection to do timing analysis at a PVT corner
[-grade <string>]	Y	The speed grade selection to do timing analysis at a PVT corner

See also: [Generating Timing Reports \(page 353\)](#), [Timing Report \(page 244\)](#), [Timing Across All Temperature Corners \(page 266\)](#)

run_tool

```
run_tool <id> [-args <string>]
```

This command runs a registered tool executable by tool ID, as specified in the ACE extensions config.xml file.

Argument	Optional	Description
<id>		The required <id> argument must match a registered tool executable by tool ID, as specified in the ACE extensions config.xml file.
[-args <string>]	Y	The optional -args <tool_args> option is used to pass commandline arguments to the underlying tool.

run_un_post_process

```
run_un_post_process [-reportsdir <string>] [-debugdir <string>] [-reroute]
```

This command removes design post-processing.

Argument	Optional	Description
[-reportsdir <string>]	Y	The optional -reportsdir <dir> option is used to override the default location for report files during this step.
[-debugdir <string>]	Y	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
[-reroute]	Y	The optional -reroute option is used to re-route the affected nets after un_post_process

run_unplace

```
run_unplace [-fixed] [-boundary] [-core] [-constants] [-insts <list>]
```

Unplace instances in the design

Argument	Optional	Description
[-fixed]	Y	Unplace instances with fixed placement constraints as well as movable instances
[-boundary]	Y	Only unplace boundary instances
[-core]	Y	Only unplace core elements
[-constants]	Y	Only unplace constant sources
[-insts <list>]	Y	Only unplace the instances specified in this list

run_unroute

```
run_unroute [-net <string>] [-pin <string>] [-nets <list>] [-regions <list>] [-regionarea <list>] [-nocore] [-clock_only] [-core] [-keepsametile] [-uniqify] [-consts] [-keepconsts]
```

Remove all or parts of a routing.

Argument	Optional	Description
[-net <string>]	Y	Only remove routing for given net
[-pin <string>]	Y	Only remove routing for given pin
[-nets <list>]	Y	unroute only the nets specified in this list
[-regions <list>]	Y	unroute only the nets in the given regions
[-regionarea <list>]	Y	unroute only the nets in the given regions
[-nocore]	Y	unroute only nets in the I/O-ring
[-clock_only]	Y	only unroute clock nets
[-core]	Y	only unroute core nets
[-keepsametile]	Y	keep all same-tile - connections
[-uniqify]	Y	make constants unique again
[-consts]	Y	only unroute constants
[-keepconsts]	Y	do NOT unroute constants

save_clock_preroute

```
save_clock_preroute [-outputfile <string>] [-add_to_project]
```

This command will save the pre-routing constraints to a PDC file.

Argument	Optional	Description
[-outputfile <string>]	Y	The optional -outputfile <file> option is used to specify the file path to which the add_clock_preroute commands will be saved. If this option is not used, the file will be saved to <project_dir>/<impl_dir>/pnr/output/<project_name>_clock_preroutes.pdc

Argument	Optional	Description
<code>[-add_to_project]</code>	Y	The optional <code>-add_to_project</code> option is used to specify that the file should be automatically added to the active project. If this is omitted, the generated PDC file is not automatically added to any project.

save_impl

```
save_impl <filename> [-no_log]
```

The `save_impl` command always uses the active impl, since only the active impl is connected to the live DB state. All other impls have no live DB state. The `save_impl` command saves the state of an impl (impl options and db state) to a `.acxdb` file. By default, the `.acxdb` file will save the entire state of the current DB, including placement and routing information. If an impl is saved before running the prepare flow step, a warning message will be printed and only the impl options will be saved, since the DB has not been prepared.

Argument	Optional	Description
<code><filename></code>		Specifies the ACXDB file path where the active impl state should be saved
<code>[-no_log]</code>	Y	If the <code>-no_log</code> option is set, no additional debug information will be saved in the ACXDB file, including log files from the current ACE session

This functionality is also accessible through buttons/menus in the ACE GUI – see [Saving Implementations \(page 302\)](#). See also: [restore_impl \(page 687\)](#) and [Restoring Implementations \(page 302\)](#).

save_partition_placements

```
save_partition_placements [-outputfile <string>] [-partition <string>] [-add_to_project]
```

Save the partition placement constraints to a PDC file which can be added to the active project.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The name of the output file
<code>[-partition <string>]</code>	Y	The name of the specific partition to save.
<code>[-add_to_project]</code>	Y	The optional <code>-add_to_project</code> option is used to specify whether the generated PDC file should automatically be added to the active project or not.

save_placement

```
save_placement [-iofile <string>] [-corefile <string>] [-add] [-output_regions] [-create_iopins] [-io_only] [-iopins_only] [-core_only] [-fixed_only] [-fix] [-device_port_names] [-instances <list>]
```

Save the current placement to one or more files as a set of pre-placement commands

Argument	Optional	Description
[-iofile <string>]	Y	The -iofile <file> option is used to specify the file path to save the IO Ring Boundary instance pre-placement commands to. If this option is not used, the file will be saved to the active project's directory as io_preplacement.pdc.
[-corefile <string>]	Y	The -corefile <file> option is used to specify the file path to save the Core instance pre-placement commands to. If this option is not used, the file will be saved to the active project's directory as core_preplacement.pdc
[-add]	Y	The -add option specifies that the outputfile should be automatically added to the active project's constraints. (It will be added to the end of the constraints list, and will thus be the last constraints file loaded.)
[-output_regions]	Y	The -output_regions option is used to enable output of region constraints into the Core PDC file (or IO Ring Boundary PDC file if -io_only is used)
[-create_iopins]	Y	The -create_iopins option is used to enable output of create_boundary_pin commands into the IO Ring Boundary PDC file
[-io_only]	Y	The -io_only option is used to specify whether only the IO Ring Boundary pre-placement file is output or not
[-iopins_only]	Y	The -iopins_only option allows you to save only the placement of the boundary pin instances
[-core_only]	Y	The -core_only option is used to specify whether only the Core pre-placement file is output or not
[-fixed_only]	Y	The -fixed_only option is used to specify whether only Fixed Instance placement data is output or not
[-fix]	Y	The -fix option forces all saved placements to be "fixed" placement, allowing you to lock down all of your placed instances

Argument	Optional	Description
<code>[-device_port_names]</code>	Y	The <code>-device_port_names</code> option is used to specify whether device port names should be output for IPINs/OPINs instead of site names
<code>[-instances <list>]</code>	Y	The <code>-instances <instances></code> option is used to specify a specific list of design instances you want to save placement for. You can call the <code>find</code> command to pass its results to this option.

save_project

```
save_project [-project <string>] [-outputfile <string>] [-acxdb <string>] [-no_log]
```

The `save_project` command saves the state of an ACE project to a `.acxprj` project file. By default, the active project is saved, unless the `-project` option is specified. The project is saved to the original project file path unless `-outputfile` is used. If the project contains the current active impl, the state of the DB can optionally be saved to an `.acxdb` file using the `-acxdb` option.

Argument	Optional	Description
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option may be used to specify a project to write out.
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile <projectFile></code> option may be used to specify an output file location.
<code>[-acxdb <string>]</code>	Y	Enables output of an ACXDB file for the active implementation and requires a file path to save the state of the active impl to
<code>[-no_log]</code>	Y	If the <code>-no_log</code> option is set, no additional debug information will be saved in the ACXDB file, including log files from the current ACE session

save_properties

```
save_properties [-outputfile <string>] [-add]
```

This command is used to save all changed properties on objects in the DB after `prepare` has been run.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile <file></code> option is used to specify the file path to which the <code>set_property</code> commands will be saved. If this option is not used, the file will be saved to the active project's directory as <code>properties.sdc</code>
<code>[-add]</code>	Y	The optional <code>-add</code> option is used to specify that the file should be automatically added to the active project. If this is omitted, the file of changed properties is not automatically added to any project.

save_regions

```
save_regions [-outputfile <string>] [-region <string>] [-all] [-explicit] [-add]
```

Save the placement region constraints to a file, using a TCL command history list approach by default.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The name of the output file
<code>[-region <string>]</code>	Y	The name of the region to save. Saves explicit instance list.
<code>[-all]</code>	Y	Save all regions (default)
<code>[-explicit]</code>	Y	Save explicit lists of instances constrained to each region, as opposed to saving the sequence of TCL commands used to build up the region constraints
<code>[-add]</code>	Y	The optional <code>-add</code> option is used to specify whether the file automatically added to the active project or not.

select

```
select <objects> [-clear]
```

This command is used to control the current object selection.

Argument	Optional	Description
<code><objects></code>		The required <code><objects></code> argument specifies a list of objects to append to the current selection. Objects must be prepended with object type prefixes (see "find" command).

Argument	Optional	Description
<code>[-clear]</code>	Y	The optional <code>-clear</code> flag is used to deselect all objects before performing the select action.

The ACE GUI provides a graphical interface for this command through the [Selection View \(page 133\)](#). See also: [Object Type Prefixes \(page 335\)](#), [Search View \(page 129\)](#), [find \(page 638\)](#), [Selecting Objects in the Floorplanner \(page 341\)](#), [trace_connections. \(page 717\)](#)

set_active_impl

```
set_active_impl <implName> [-project <string>]
```

This command sets the active implementation for the current ACE session. The active implementation controls which implementation the flow and project management commands are operating on.

Argument	Optional	Description
<code><implName></code>		The required <code><implName></code> argument is used to specify the name of the implementation to set as the active implementation.
<code>[-project <string>]</code>	Y	The optional <code>-project <projectName></code> option is used to specify an alternate project (by name) for the active implementation to be set in.

set_clock_type

```
set_clock_type <clock> [-batch] [-boundary] [-trunk] [-minitrunk] [-data_region]
[-data_center] [-data_local] [-data_column]
```

Set properties of a clock. If a non-driving (target) clock pin is specified, a local property is set for that pin only.

Argument	Optional	Description
<code><clock></code>		net or pin ('inst/pin')
<code>[-batch]</code>	Y	Postpone application of this constraint until <code>apply_placement</code> is called at the end of <code>run_prepare</code> . This is useful for nets that are created or renamed by ACE during <code>run_prepare</code>
<code>[-boundary]</code>	Y	boundary clock (not routed through the trunk)
<code>[-trunk]</code>	Y	trunk clock

Argument	Optional	Description
[-minitrunk]	Y	routed via minitrunk instead of main trunk
[-data_region]	Y	data as clock, using region resources
[-data_center]	Y	data as clock, using center resources
[-data_local]	Y	data as clock, using local resources
[-data_column]	Y	data as clock, using branch base resources

set_cluster

```
set_cluster <cluster> [-id <int>] [-wt <int>]
```

Pre-placement command to generate user-defined clusters. This clustering command directs the ACE Placer to keep the specified instances together.

Argument	Optional	Description
<cluster>		The required <cluster> list argument is a list of instances which should be clustered in an RLB half
[-id <int>]	Y	Optional 2nd level cluster id
[-wt <int>]	Y	Optional 2nd level cluster weight {2, ..., 5} - higher weights signify cluster importance

set_equivalent_pins

```
set_equivalent_pins <equivalent_pin_names>
```

This command marks multiple nets or instance pins as functionally equivalent

Argument	Optional	Description
<equivalent_pin_names>		A list of pin names or net names that are to be marked as functionally equivalent (<p:toplevel_port_name> <t:user_pin_name> <n:net_name>)

set_flyline_direction

```
set_flyline_direction <direction>
```

This command is used to control whether selected instance flylines are shown for loads of the selected instance, drivers, or both.

Argument	Optional	Description
<direction>		The required <direction> argument specifies whether selected instance flylines are shown for loads of the selected instance, drivers, or both. Valid values are loads_only, drivers_only, both and all

set_impl_option

```
set_impl_option <option_name> [<value>] [-project <string>] [-impl <string>]
```

This command sets options for a project implementation. Only one option may be set at a time.

Argument	Optional	Description
<option_name>		The name of the impl option to set a value for. To see a list of valid impl options, use the report_impl_options TCL command.
[<value>]	Y	The new value to set the impl option to.
[-project <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.
[-impl <string>]	Y	The optional -project <projectName> and -impl <implName> options are used to specify an alternate project implementation (by name) to set options for.

set_max_flyline_fanout

```
set_max_flyline_fanout <limit>
```

This command is used to hide selected instance flylines for nets with fanout greater than the limit passed in.

Argument	Optional	Description
<limit>		The required <limit> argument specifies the maximum fanout for flylines to be displayed for a net.

set_partition_force_changed

```
set_partition_force_changed <name> <changed>
```

Set the force changed flag of a partition with the given name. This command should be called at the end of the incremental compile flow. Setting the flag will force the partition to be recompiled the next time that the flow is run, even if the partition timestamp has not changed.

Argument	Optional	Description
<name>		Name of the Partition
<changed>		Set to 1 to mark this partition as force changed, set to 0 to reset the flag and mark the partition as not forced changed

set_partition_info

```
set_partition_info [-name <string>] [-view <string>] [-cp_type <string>]
-timestamp <string> -import <string> [-comment <string>] [-exclusive_placement]
```

Set partition (compile point) information on the design. This command creates a new partition definition if one does not already exist in the design. This usually comes from the synthesis tool in <design>_partition.tcl

Argument	Optional	Description
[-name <string>]	Y	Hierarchical pathname of the partition in the netlist (e.g. /top_module/instance1/instance2)
[-view <string>]	Y	Module name of the partition in the original RTL. Will be used as the name of the blackbox module if the partition is exported.
[-cp_type <string>]	Y	Compile point type (hard, locked, or soft)
-timestamp <string>		Timestamp in seconds to indicate when this partition was last synthesized. Default is current time in seconds. The options -timestamp and -import are mutually exclusive
-import <string>		Pathname to the .epdb file containing the placed and routed partition database to be imported. The options -timestamp and -import are mutually exclusive
[-comment <string>]	Y	User specified comment string describing the partition
[-exclusive_placement]	Y	Sets the given partition with an exclusive placement property. Instances belonging to the current partition will not be placed together with non partition instances

set_placement

```
set_placement <objName> <siteName> [-fixed] [-batch] [-partition] [-warning] [-auto_place_neighbors <int>]
```

This command assigns the placement of an instance to a site

Argument	Optional	Description
<objName>		The required <objName> argument is used to specify the name of an instance (i:) or port (p:) to be placed. If multiple objects are to be placed at once, a TCL list of objectnames may be specified. NOTE: Ports (p:) may not be used if the port is connected to a black box IO Ring instance. The port (p:) may only be used to place boundary pins (IPIN, OPIN, CLK_IPIN, CLK_OPIN) that are connected to the port.
<siteName>		The required <siteName> argument is used to specify the site (s:) to place the instance on. In the case of IO pin placement, a device port name (d:) may be specified instead of a site name. If multiple objects are to be placed at once, a TCL list of sitenames may be specified.
[-fixed]	Y	The -fixed option specifies that the placement of the instance should be fixed to this site and not movable by the placer
[-batch]	Y	Postpone application of this constraint until apply_placement is called at the end of run_prepare. This option is useful for instances that are created or renamed by Ace during run_prepare.
[-partition]	Y	Use this instance as an anchor to specify the placement of an entire partition relative to the specified site
[-warning]	Y	Instead of erroring out, only print a warning message if the instance(s) specified with <objName> do not exist at the time this constraint is applied
[-auto_place_neighbors <int>]	Y	This option can be applied to PINs and takes the number of logic levels to process as an argument. It currently only supports placing LUTs driving OPINs for 1 level of logic.

set_project_constraints_pvt

```
set_project_constraints_pvt <file> [-corner <string>] [-temperature <string>] [-voltage <string>]
```

This command allows the user to set the specific PVT conditions in which an SDC constraint file is applied.

Argument	Optional	Description
<file>		The required <file> argument is used to specify the file path to the SDC constraint file.
[-corner <string>]	Y	The -corner <corner> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given process corner. Valid values are "fast" and "slow"
[-temperature <string>]	Y	The -temperature <temp> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given temperature corner. Valid values are device-specific and must match a value from the junction_temperature impl option list.
[-voltage <string>]	Y	The -voltage <v> option is used to mark an SDC constraints file (containing only delays) as being applicable only to the given core voltage corner. Valid values are device-specific and must match a value from the core_voltage impl option list.

set_project_option

```
set_project_option <option_name> [<value>] [-project <string>]
```

This command sets options for a project. Only one option may be set at a time.

Argument	Optional	Description
<option_name>		The name of the project option to set a value for. To see a list of valid project options, use the report_project_options TCL command.
[<value>]	Y	The new value to set the project option to.
[-project <string>]	Y	The optional -project <projectName> is used to specify an alternate project (by name) to set options for.

set_property

```
set_property <propName> <propValue> <objects> [-warning] [-quiet]
```

This command is used to set properties on objects in the DB.

Argument	Optional	Description
<propName>		The required <propName> argument specifies property name to set on the objects passed in.
<propValue>		The required <propValue> argument specifies property value to set on the objects passed in.
<objects>		The required <objects> argument specifies a list of objects to set the property for. Objects must be prepended with object type prefixes (see "find" command).
[-warning]	Y	The optional -warning option allows you to downgrade error messages about missing netlist objects to warning messages
[-quiet]	Y	The optional -quiet option allows you to disable printing of info messages

set_region_bounds

```
set_region_bounds <region> <bounds> [-snap_to_clock_regions] [-
snap_to_fabric_clusters] [-snap <string>]
```

This command updates a placement region's bounding box of tiles

Argument	Optional	Description
<region>		Name of the region
<bounds>		List of bounding box coordinates {x1 y1 x2 y2}. x1 and y1 are the upper left corner of the box. x2 and y2 are the lower right corner of the box.
[-snap_to_clock_regions]	Y	Snap the bounding box to clock region boundaries (deprecated, use '-snap clock_regions')
[-snap_to_fabric_clusters]	Y	Snap the bounding box to fabric cluster boundaries (deprecated, use '-snap fabric_clusters')
[-snap <string>]	Y	How to snap the region bounding box coordinates. Legal values are: 'none', 'tiles' (the default), 'fabric_clusters', 'clock_regions'

set_region_type

```
set_region_type <region> [-type <string>] [-soft] [-include_routing]
```

This command updates a placement region's type

Argument	Optional	Description
<region>		Name of the region
[-type <string>]	Y	What type of placement region this is. Legal values are: 'inclusive' (the default), 'keepout', 'soft'. Instances added to an 'inclusive' region (and attached routing wires when '-include_routing' is set) will be placed within the region bounding box. An 'inclusive' region permits instances to be placed inside the region even if they do not belong to the region. A 'keepout' region prevents any instances (and routing wires when '-include_routing' is set) from being placed inside the region. No instances may be added to a 'keepout' region. Instances added to an 'soft' region will be pulled toward the region's center during placement, but instances are permitted to overflow the bounds of the 'soft' region.
[-soft]	Y	A 'soft' placement region attempts to pull instance placement to its center, but allows instance placement to overflow it's bounds
[-include_routing]	Y	Constrain routing wires, as well as instances, to stay within the region boundary box

set_units

```
set_units
```

Set the default units for timing constraints.

sleep

```
sleep <seconds>
```

sleep for number seconds.

Argument	Optional	Description
<seconds>		number of seconds to sleep

source_encrypted

```
source_encrypted <tclfile> [-nodigest] [-debug] [-sig] [-untar]
```


source encrypted tcl file.

Argument	Optional	Description
<tclfile>		encrypted tcl file to source
[-nodigest]	Y	no digest is also OK
[-debug]	Y	internal use only
[-sig]	Y	only require a signature
[-untar]	Y	encrypted file is tar archive, untar first.

source_secure

source_secure <tclfile> [-encoding <string>] [-nodigest] [-sig] [-untar]
source encrypted tcl file.

Argument	Optional	Description
<tclfile>		encrypted tcl file to source
[-encoding <string>]	Y	compatibility with 'source'
[-nodigest]	Y	no digest is also OK
[-sig]	Y	only require a signature
[-untar]	Y	encrypted file is tar archive, untar first.

trace_connections

trace_connections <insts> [-drivers_only] [-targets_only] [-include_clocks] [-include_resets]

This command is used to find instances that are connected to the list of instances passed in. By default, this command traces to find all (drivers and targets) connected instances, except for those connected via clock or reset pins.

Argument	Optional	Description
<insts>		The required <insts> argument specifies a list of instance objects to trace connectivity from. Objects must be prepended with object type prefixes (see "find" command).
[-drivers_only]	Y	If you want to select only upstream logic that drives the instances in the current selection, use -drivers_only
[-targets_only]	Y	If you want to select only downstream logic driven by the currently selected instances, use -targets_only.
[-include_clocks]	Y	To include tracing connectivity on clock nets, use -include_clocks.
[-include_resets]	Y	To include tracing connectivity on reset nets, use -include_resets.

Similar to the [find](#) (page 638) command, this functionality is especially useful when creating lists as input to other Tcl commands, like [select](#) (page 708) and [highlight](#) (page 664).

untar

```
untar <tarfile> [-member <string>] [-list] [-lz]
```

extract content from tar file.

Argument	Optional	Description
<tarfile>		tarfile from which to extract
[-member <string>]	Y	member to extract, default: none
[-list]	Y	list files in archive
[-lz]	Y	uncompress first

write_aeskey_efuse_bitstream

```
write_aeskey_efuse_bitstream aeskey_file aeskey_index [-outputfile <string>]
```

This command generates a .hex file used to burn the AES key eFuse. It can run after the run_prepare flow step.

Argument	Optional	Description
aeskey_file		The path to the aes key file containing your 256-bit AES key.
aeskey_index		The AES key index to burn. Must be either 1,2,3.
[-outputfile <string>]	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation output directory and is named <design_name>_efuse.hex.

write_bitstream

```
write_bitstream [-outputfile <string>] [-debugdir <string>] [-reportsdir <string>]
[-flash] [-pcie] [-pcie_txt] [-cpu] [-cpu_width <int>] [-two_stage] [-nocompress]
[-compress] [-chainfile <string>]
```

This command generates a programming bitstream for a fully placed and routed design in .hex format.

Argument	Optional	Description
[-outputfile <string>]	Y	The optional -outputfile <file> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation output directory and is named <design_name>.hex.
[-debugdir <string>]	Y	The optional -debugdir <dir> option is used to override the default location for debug files during this step.
[-reportsdir <string>]	Y	The optional -reportsdir <dir> option is used to override the default location for report files during this step.
[-flash]	Y	The optional -flash option may be used to output an additional serial flash binary file format output
[-pcie]	Y	The optional -pcie option may be used to output an additional PCIe binary file format output
[-pcie_txt]	Y	The optional -pcie_txt option may be used to output an additional PCIe file format output. This is a text file format used for debug and must be used with -pcie or bitstream_output_pcie

Argument	Optional	Description
<code>[-cpu]</code>	Y	The optional <code>-cpu</code> option may be used to output an additional CPU Mode file format output
<code>[-cpu_width <int>]</code>	Y	This option controls the bit width of the CPU Mode formatted output file. If you are using the CPU interface in x8 mode, set this value to 8. If you are using the CPU interface in x128 mode, set this to 128.
<code>[-two_stage]</code>	Y	The optional <code>-two_stage</code> option may be used to output bitstream files divided by user mode
<code>[-nocompress]</code>	Y	write output as plain-text file
<code>[-compress]</code>	Y	compress output-file
<code>[-chainfile <string>]</code>	Y	The optional <code>-chainfile <file></code> may be used to override the chainfile set in the active Impl

write_critical_paths_script

```
write_critical_paths_script [-outputfile <string>]
```

This command writes a tcl script that may be used for viewing critical paths in the synthesis tool.

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	The optional <code>-outputfile <file></code> option may be used to specify an output file name or file path. If this option is not present, the output is written to the default implementation output directory and is named <code><design_name>_critical_paths.tcl</code> .

write_netlist

```
write_netlist [-outputfile <string>] [-debugdir <string>] [-final] [-compress]
```

This command generates a verilog netlist for simulation (same as `run_generate_netlist`).

Argument	Optional	Description
<code>[-outputfile <string>]</code>	Y	Output netlist file name.
<code>[-debugdir <string>]</code>	Y	The optional <code>-debugdir <dir></code> option is used to override the default location for debug files during this step.

Argument	Optional	Description
<code>[-final]</code>	Y	Output DRC-free final netlist
<code>[-compress]</code>	Y	Compress output file with gzip

write_partition_blackbox

```
write_partition_blackbox <partition> [-library <string>] [-outputfile <string>]
```

Command to write a verilog blackbox model for a partition

Argument	Optional	Description
<code><partition></code>		Export a blackbox netlist for the specified partition as a verilog file
<code>[-library <string>]</code>	Y	Specifies the name of the blackbox model library (default is "blackbox")
<code>[-outputfile <string>]</code>	Y	Specifies the output file name for the blackbox netlist (default is <code><cellname>_bb.v</code> in the implementation output directory)

write_partition_db

```
write_partition_db <partition> [-outputfile <string>] [-include_pr_zone]
```

Command to export the place-and-route database for a partition

Argument	Optional	Description
<code><partition></code>		Export the place-and-route database for the specified partition into an .epdb file
<code>[-outputfile <string>]</code>	Y	Specifies the output file name for the partition database (default is <code>partitions/<instname>.epdb</code> in the implementation output directory)
<code>[-include_pr_zone]</code>	Y	Search for Placement Region that corresponds to partition and use it in splitting routing for nets

write_tcl_history

```
write_tcl_history <outputfile>
```

Dump the TCL command history for this ACE session to a file

Argument	Optional	Description
<outputfile>		The file to save the TCL script to

TCL Commands for the IP Generator Plugin Framework

These commands are in the `ace_ip::` namespace.

- [add_ip_binary_property](#) (page 723)
- [add_ip_bool_property](#) (page 724)
- [add_ip_enum_property](#) (page 725)
- [add_ip_file_path_property](#) (page 726)
- [add_ip_float_property](#) (page 726)
- [add_ip_gui_control](#) (page 727)
- [add_ip_gui_group](#) (page 728)
- [add_ip_gui_page](#) (page 729)
- [add_ip_hex_property](#) (page 730)
- [add_ip_include_path](#) (page 730)
- [add_ip_int_property](#) (page 731)
- [add_ip_label](#) (page 732)
- [add_ip_output_file](#) (page 733)
- [add_ip_output_file_group](#) (page 733)
- [add_ip_port](#) (page 734)
- [add_ip_rule_message](#) (page 735)
- [add_ip_signal_name_property](#) (page 736)
- [add_ip_string_property](#) (page 736)
- [add_ip_to_gui](#) (page 737)
- [create_ip](#) (page 738)
- [get_ip_include_paths](#) (page 738)
- [get_ip_macro_name](#) (page 739)
- [get_ip_ports](#) (page 739)
- [get_ip_property_attributes](#) (page 740)
- [get_ip_property_value](#) (page 742)
- [remove_ip_gui_control](#) (page 743)
- [remove_ip_gui_group](#) (page 743)
- [remove_ip_gui_page](#) (page 744)
- [remove_ip_label](#) (page 744)

- [remove_ip_property](#) (page 745)
- [send_ip_definitions](#) (page 745)
- [send_ip_gui_update](#) (page 746)
- [set_ip_category](#) (page 746)
- [set_ip_description](#) (page 746)
- [set_ip_macro_name](#) (page 747)
- [set_ip_property_value](#) (page 747)
- [set_ip_property_verilog_param_attributes](#) (page 748)
- [set_ip_resource_count](#) (page 749)
- [set_ip_resource_types](#) (page 749)
- [set_update_ip_configuration_callback](#) (page 750)
- [update_ip_binary_property](#) (page 754)
- [update_ip_bool_property](#) (page 755)
- [update_ip_configuration](#) (page 755)
- [update_ip_enum_property](#) (page 756)
- [update_ip_file_path_property](#) (page 756)
- [update_ip_float_property](#) (page 757)
- [update_ip_hex_property](#) (page 757)
- [update_ip_int_property](#) (page 758)
- [update_ip_label](#) (page 759)
- [update_ip_signal_name_property](#) (page 759)
- [update_ip_string_property](#) (page 760)

add_ip_binary_property

```
add_ip_binary_property <name> <gui_label> <default_value> <number_of_bits> [-min
<number>] [-max <number>] [-description <text>] [-enabled <state>] [-ip_name
<ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value (binary)
<number_of_bits>		Integer	Number of allowed bits

Argument	Optional	Type	Description
<code>[-min <number>]</code>	Y	String	Minimum allowed value (binary)
<code>[-max <number>]</code>	Y	String	Maximum allowed value (binary)
<code>[-description <text>]</code>	Y	String	Tool-tip / documentation description
<code>[-enabled <state>]</code>	Y	Boolean	Flag to enable/disable property
<code>[-ip_name <ip_name>]</code>	Y	String	IP name associated with this property

Description

The `add_ip_binary_property` is used to define an IP binary property.

Example

```
add_ip_binary_property "example_binary_property_1" "Example Binary Range Property 1" "0101" 4 -min "0000" -max "1111" -description "Long description text for tool tip" -enabled "false"
```

add_ip_bool_property

```
add_ip_bool_property <name> <gui_label> <default_value> [-description <text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<code><name></code>		String	ACXIP property name
<code><gui_label></code>		String	Property label to display on the GUI editor
<code><default_value></code>		String	Default property value (Yes or No)
<code>[-description <text>]</code>	Y	String	Tool-tip / documentation description
<code>[-enabled <state>]</code>	Y	Boolean	Flag to enable/disable property
<code>[-ip_name <ip_name>]</code>	Y	String	IP name associated with this property

Description

The `add_ip_bool_property` is used to define an IP boolean property.

Example

```
add_ip_bool_property "example_bool_property_1" "Example Boolean Property 1" "Yes"
-description "Long description text for tooltip" -enabled "false"
```

add_ip_enum_property

```
add_ip_enum_property <name> <gui_label> <default_value> <list_of_valid_values> [-
description <text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value
<list_of_valid_values>		Array	List of valid values
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_enum_property` is used to define an IP enum property.

Example

```
add_ip_enum_property "example_enum_property_1" "Example Enum Property 1"
"Item2" {"Item1" "Item2" "Item3"} -description "Long description text for tool tip"
-enabled "false"
```

add_ip_file_path_property

```
add_ip_file_path_property <name> <gui_label> <default_value> [-description <text>]
[-enabled <state>] [-file_must_exist <exists>] [-isDirectory <is_dir>] [-ip_name
<ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value
[-file_must_exist <exists>]	Y	Boolean	If set to "true", the file must exist in the filesystem, or else the rule check will fail
[-isDirectory <is_dir>]	Y	Boolean	Set to "true" if the path is a directory instead of a file
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_file_path_property` is used to define an IP file path property.

Example

```
add_ip_file_path_property "example_filepath_property_1" "Example Filepath Property
1" "./test_file.txt" -description "Long description text for tool-tip" -enabled
"false" -file_must_exist "true"
```

add_ip_float_property

```
add_ip_float_property <name> <gui_label> <default_value> [-min <number>] [-max
<number>] [-description <text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		Double	Default property value
[-min <number>]	Y	Double	Minimum allowed value
[-max <number>]	Y	Double	Maximum allowed value
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_float_property` is used to define an IP float property.

Example

```
add_ip_float_property "example_float_property_1" "Example Floating Point Range Property 1" "3.1" -min "2.7" -max "32.4" -description "Long description text for tooltip" -enabled "false"
```

add_ip_gui_control

```
add_ip_gui_control <property_name> <page> [-parent <parent>] [-control_type <type>] [-span <column_span>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<property_name>		String	ACXIP property name
<page>		String	IP GUI page name to add the control into
[-parent <parent>]	Y	String	The parent IP Group box to add the control into (if applicable)

Argument	Optional	Type	Description
<code>[-control_type <type>]</code>	Y	String	GUI control type. Valid options are combo, radio, text, checkbox, file
<code>[-span <column_span>]</code>	Y	Integer	GUI page layout column span
<code>[-ip_name <ip_name>]</code>	Y	String	IP name associated with this page

Description

The `add_ip_gui_control` is used to define IP property GUI controls. Control types for each of the property type can be as follows: combo, radio or text for List, combo or text for Int, text for Float, Hex, Binary and Signal and file for Filepath.

Example

```
add_ip_gui_control "some_int_property" "Overview" -control_type "combo"
add_ip_gui_control "another_int_property" "Overview" -control_type "text"
```

add_ip_gui_group

```
add_ip_gui_group <name> <label> <page> [-parent <parent_group>] [-columns
<layout_columns>] [-span <column_span>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<code><name></code>		String	Group name
<code><label></code>		String	GUI visible group label
<code><page></code>		String	IP GUI page name to add the group to
<code>[-parent <parent_group>]</code>	Y	String	The parent IP Group box to add the group into (if applicable)
<code>[-columns <layout_columns>]</code>	Y	Integer	Number of columns to use for layout inside this group
<code>[-span <column_span>]</code>	Y	Integer	Column span
<code>[-ip_name <ip_name>]</code>	Y	String	IP name associated with this page

Description

The `add_ip_gui_group` is used to define groups for GUI control/s in IP editor GUI page.

Example

```
add_ip_gui_group "group_x" "Example Group X" "Overview" -columns 2 -span 1
add_ip_gui_control "some_list_property" "Overview" -parent "group_x" -control_type
"combo"
```

add_ip_gui_page

```
add_ip_gui_page <name> <description> [-parent <parent_page_name>] [-columns
<layout_columns>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	IP GUI page name
<description>		String	IP description text for page banner
[-columns <layout_columns>]	Y	Integer	Number of GUI page layout columns
[-parent <parent_page_name>]	Y	String	The parent IP page (if applicable)
[-ip_name <ip_name>]	Y	String	IP name associated with this page

Description

The `add_ip_gui_page` is used to define GUI page structure as part of the GUI page layout definition.

Example

```
add_ip_gui_page "Overview" "Long description text for page banner"
add_ip_gui_page "Sub-Page A" -parent "Overview" "Long description text for page
banner"
add_ip_gui_page "Sub-Page B" -parent "Overview" "Long description text for page
banner"
add_ip_gui_page "Sub-Page C" -parent "Sub-Page B" "Long description text for page
banner"
```

add_ip_hex_property

```
add_ip_hex_property <name> <gui_label> <default_value> <number_of_bits> [-min
<number>] [-max <number>] [-description <text>] [-enabled <state>] [-ip_name
<ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value (hex)
<number_of_bits>		Integer	Number of allowed bits
[-min <number>]	Y	String	Minimum allowed value (hex)
[-max <number>]	Y	String	Maximum allowed value (hex)
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_hex_property` is used to define an IP hexadecimal property.

Example

```
add_ip_hex_property "example_hex_property_1" "Example Hex Range Property 1"
"012345" 24 -min "000000" -max "FFFFFF" -description "Long description text for
tooltip" -enabled "false"
```

add_ip_include_path

```
add_ip_include_path <include_path> [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<include_path>		String	Include path
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_include_path` is used to include path in output file. This is optional, and is used by any built-in output file generator TCL procedures.

Example

```
add_ip_include_path "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/rtl/
ACX_EXAMPLE_SOFT_IP.sv"
add_ip_include_path "speedster7t/macros/ACX_EXAMPLE_SOFT_IP/src/include/
example_include_file.svh"
```

add_ip_int_property

```
add_ip_int_property <name> <gui_label> <default_value> [-min <number>] [-max
<number>] [-description <text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		Integer	Default property value (decimal)
[-min <number>]	Y	Integer	Minimum allowed value (decimal)
[-max <number>]	Y	Integer	Maximum allowed value (decimal)
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_int_property` is used to define an IP integer property.

Example

```
add_ip_int_property "example_int_property_1" "Example Integer Range Property 1" "3"
-min "2" -max "32" -description "Long description text for tool tip" -enabled
>false"
```

add_ip_label

```
add_ip_label <name> <gui_label> <default_value> [-description <text>] [-enabled
<state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_label` is used to define IP labels.

Example

```
add_ip_label "examplelabel_1" "Example Dynamic Label 1" "Hello World" -description
"Long description text for tool tip" -enabled "true"
```


add_ip_output_file

```
add_ip_output_file <label> <enable_property> <file_property> <group> <description>
[-file_ext <supported_extensions>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<label>		String	IP GUI page editor name
<enable_property>		String	ACXIP property name for file output enable
<file_property>		String	Output file ACXIP property name
<group>		String	Output generation group
<description>		String	Description text
[-file_ext <supported_extensions>]	Y	String	Supported extension
[-enabled <state>]	Y	Boolean	Enable setting in GUI
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_output_file` is used to define output file types (Verilog, VHDL, SystemVerilog, SDC, PDC, Custom, etc).

Example

```
add_ip_output_file_group "RTL Files" "Some description text for tool-tip and docs"
add_ip_output_file "Verilog Model" "output.verilog_file_en" "output.verilog_file"
"RTL Files" "Description text here..." -enabled "Yes" -file_ext "*.v"
```

add_ip_output_file_group

```
add_ip_output_file_group <name> <description>
```

Argument	Optional	Type	Description
<name>		String	Output file group name

Argument	Optional	Type	Description
<description>		String	Output file group description for tool-tip

Description

The `add_ip_output_file_group` is used to define output file group for supported output file types (Verilog, VHDL, SystemVerilog, SDC, PDC, Custom, etc).

Example

```
add_ip_output_file_group "RTL Files" "Some description text for tool-tip and docs"
```

add_ip_port

```
add_ip_port <port_name> <signal_group> <direction> <width> <diagram_side>
<description> [-exposed <exposed>] [-tieoff <tieoff>] [-typedef <typedef>] [-
clock_domain <domain>]
```

Argument	Optional	Type	Description
<port_name>		String	Port name
<signal_group>		String	Port signal group
<direction>		String	Port direction (in/out/inout)
<width>		Integer	Port width
<diagram_side>		String	GUI diagram side (East/West)
<description>		String	Port description for tooltip
[-exposed <exposed>]	Y	Boolean	Indicates whether to tie off this port on the underlying soft IP macro, or expose it in the generated wrapper
[-tieoff <tieoff>]	Y	String	If a port is not "exposed", use this value to tie it off inside the generated wrapper
[-typedef <typedef>]	Y	String	Optionally add a typedef string to the generated Verilog

Argument	Optional	Type	Description
<code>[-clock_domain <domain>]</code>	Y	String	Optionally indicate what clock domain this port is on (for generating SDC files, etc)

Description

The `add_ip_port` is used to define ports for generated Verilog/VHDL and interactive diagram.

Example

```
add_ip_port "ref_clk" "Clocks and Resets" "in" 1 "West" "Some description Text for tool-tip and documentation..."
```

add_ip_rule_message

```
add_ip_rule_message <ip_property_name> <title> <status> <message>
```

Argument	Optional	Type	Description
<code><ip_property_name></code>		String	ACXIP property name
<code><title></code>		String	message title
<code><status></code>		String	Error, Warning, Info
<code><message></code>		String	message to be displayed

Description

The `add_ip_rule_message` is used to report ip rule messages.

Example

```
add_ip_rule_message "read_data_width" "Data width mismatch example" "Error" "Write width must be set to the same value as Read Width. The current values are different. Please fix this."
```

add_ip_signal_name_property

```
add_ip_signal_name_property <name> <gui_label> <default_value> [-description
<text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor
<default_value>		String	Default property value
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_signal_name_property` is used to define an IP signal name property.

Example

```
add_ip_signal_name_property "example_signal_name_property_1" "Example Signal Name
Property 1" "Yes" -description "Long description text for tool tip" -enabled
>false"
```

add_ip_string_property

```
add_ip_string_property <name> <gui_label> <default_value> [-min_characters
<number>] [-max_characters <number>] [-valid_characters_regex <regex>] [-
description <text>] [-enabled <state>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<gui_label>		String	Property label to display on the GUI editor

Argument	Optional	Type	Description
<default_value>		String	Default property value
[-valid_characters_regex <regex>]	Y	String	Regular expression for checking validity of the property value
[-min_characters <number>]	Y	Integer	Minimum number of characters allowed
[-max_characters <number>]	Y	Integer	Maximum number of characters allowed
[-description <text>]	Y	String	Tool-tip / documentation description
[-enabled <state>]	Y	Boolean	Flag to enable/disable property
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `add_ip_string_property` is used to define an IP string property.

Example

```
add_ip_string_property "example_string_property_1" "Example String Property 1
Label" "Default Value" -min_characters "2" -max_characters "32"
-valid_characters_regex "[a-zA-Z0-9].*" -description "Long description text for
tooltip" -enabled "true"
```

add_ip_to_gui

```
add_ip_to_gui <ip_name>
```

Argument	Optional	Type	Description
<ip_name>		String	IP generator tool name

Description

The `add_ip_to_gui` is required to be called once, at the end of the IP generator tool definition TCL file. This sends the new definition to the GUI.

Example

```
add_ip_to_gui "Example Soft IP"
```

create_ip

```
create_ip <ip_name> <library> <version>
```

Create the top-level IP generator tool definition.

Argument	Optional	Type	Description
<ip_name>		String	IP generator tool name
<library>		String	IP Library
<version>		String	IP definition version

Description

The `create_ip` creates the top-level IP generator tool definition. If this is called a second time, it wipes out the existing IP generator tool definition. This lets you "source" this TCL script multiple times in an ACE session when you are developing the IP generator tool.

Example

```
create_ip "Example Soft IP" "Speedster7t" "1.3"
```

get_ip_include_paths

```
get_ip_include_paths [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `get_ip_include_paths` is used to obtain a list of include paths defined for the output file. This is optional, and is used by any built-in output file generator TCL procedures..

Example

```
set includePaths [get_ip_include_paths]
foreach p $includePaths {
    puts $out_file "`include \"$p\""
```

get_ip_macro_name

```
get_ip_macro_name [-ip_name <ip_name>]
```

Argument	Optional	String	Description
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `get_ip_macro_name` is used to get the declared macro name. This is optional, and is used by any built-in output file generator TCL procedures.

Example

```
set macroName [get_ip_macro_name]
```

get_ip_ports

```
get_ip_ports
```

Description

The `get_ip_ports` returns a TCL associative array of all attributes for all IP generator ports. The following attributes are returned for each port: `port_name`, `signal_group`, `direction`, `width`, `diagram_side`, `description`, `exposed`, `tieoff`, `typedef` and `clock_domain`.

Example

```
proc ace_ip::outputVerilogIpPorts {output_file} {
    upvar $output_file out_file
    array set ipgenPorts [ace_ip::get_ip_ports]
    set numPorts [array size ipgenPorts]
    set idx 0
    foreach key [array names ipgenPorts] {
        array set ipPort $ipgenPorts($key)
        set portName $ipPort(port_name)
        set direction $ipPort(direction)
```

```

set width $ipPort(width)
set desc $ipPort(description)
set l ""
if {$direction == "in"} {
    if {$width > 1} {
        set w [expr $width - 1]
        set l " input \[$w:0\] $portName"
    }
    if {$width == 1} {
        set l " input $portName"
    }
}
if {$direction == "out"} {
    if {$width > 1} {
        set w [expr $width - 1]
        set l " output \[$w:0\] $portName"
    }
    if {$width == 1} {
        set l " output $portName"
    }
}
}
if {$idx < [expr $numPorts - 1]} {
    puts $out_file " $l, \/\/$desc"
} else {
    puts $out_file " $l \/\/$desc"
}
incr idx
}
}

```

get_ip_property_attributes

get_ip_property_attributes [-property <property_name>]

Argument	Optional	Type	Description
[-property <property_name>]	Y	String	ACXIP property name

Description

The `get_ip_property_attributes` is used to get a TCL associative array of all the attributes of an IP property. For every property, the following common attributes are returned: `name`, `label`, `default_value`, `description`, `enabled`, `verilog_param`, `to_upper`, `num_bits`, `format` and `alt_param_name`. Additional property attributes are also returned based on property type. If no property name is passed, a TCL associative array of all properties with their respective attributes is returned.

Example

```
proc ace_ip::outputVerilogMacroParameterList {acxip output_file} {
    upvar $output_file out_file
    upvar $acxip acxip_data

    array set ipProperties [ace_ip::get_ip_property_attributes]

    set vCounter 0
    foreach key [array names ipProperties] {
        array set ipProp $ipProperties($key)
        set isVerilogParam $ipProp(verilog_param)
        if {$isVerilogParam == "true"} {
            incr vCounter
        }
    }

    if {$vCounter > 0} {
        set idx 0
        foreach key [array names ipProperties] {
            array set ipProp $ipProperties($key)
            set isVerilogParam $ipProp(verilog_param)
            set altParamName $ipProp(alt_param_name)
            set toUpper $ipProp(to_upper)
            set numBits $ipProp(num_bits)
            set verilogFormat $ipProp(format)
            if {$isVerilogParam == "true"} {
                set paramName $key
                if {[string trim $altParamName] != 0} {
                    set paramName $altParamName
                }
                if {$toUpper == "true"} {
                    set paramName [string toupper $paramName]
                }

                set val $acxip_data($key)
```


Example

```
if {[ace_ip::get_ip_property_value "intParam1"] <= [ace_ip::get_ip_property_value
"intParam2"]} {
    ace_ip::add_ip_rule_message "intParam1" "Rule check example" "Error" "Integer
Param 1 must be greater than Integer Param 2. Please fix this."
}
```

remove_ip_gui_control

```
remove_ip_gui_control <name> <page>
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<page>		String	IP GUI page name

Description

The `remove_ip_gui_control` is used to remove a dynamic GUI IP control. A GUI IP control is dynamic if it is added inside the update IP configuration callback. The remove function only removes a control if it exists, otherwise it silently succeeds.

Example

```
for {set i $acxip(num_lanes)} {$i < 32} {incr $i} {
    remove_ip_gui_control "top_level_int_a_property_index$i" "Overview"
}
```

remove_ip_gui_group

```
remove_ip_gui_group <acxip_file_path> <name>
```

Argument	Optional	Type	Description
<name>		String	Group name
<page>		String	IP GUI page name

Description

The `remove_ip_gui_group` is used to remove a dynamic GUI IP group. A GUI IP group is dynamic if it is added inside the update IP configuration callback. The remove function only removes a group if it exists, otherwise it silently succeeds. It also removes any GUI controls that are mapped to the removed group.

Example

```
for {set i $acxip(num_lanes)} {$i < 32} {incr $i} {
    remove_dynamic_ip_gui_group "group_x" "Overview"
}
```

remove_ip_gui_page

```
remove_ip_gui_page <name>
```

Argument	Optional	Type	Description
<name>		String	Page name

Description

The `remove_ip_gui_page` is used to remove a dynamic GUI IP page. A GUI IP page is dynamic if it is added inside the update IP configuration callback. The remove function only removes a page if it exists, otherwise it silently succeeds. It also removes any GUI groups and controls that are mapped to the removed page.

Example

```
for {set i $acxip(num_lanes)} {$i < 32} {incr $i} {
    remove_ip_gui_page "Page For Lane $i"
}
```

remove_ip_label

```
remove_ip_label <name>
```

Argument	Optional	Type	Description
<name>		String	Label name

Description

The `remove_ip_label` is used to remove calculated values of dynamic IP labels. An IP label is dynamic if it is added inside the update IP configuration callback. The remove function only removes a property if it exists, otherwise it silently succeeds.

Example

```
remove_ip_label "some_calculated_value"
```

remove_ip_property

```
remove_ip_property <acxip_file_path> <name>
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name

Description

The `remove_ip_property` is used to remove a dynamic IP property. An IP property is dynamic if it is added inside the update IP configuration callback. The remove function only removes a property if it exists, otherwise it silently succeeds. It also removes any GUI controls that are mapped to the removed property.

Example

```
remove_ip_property "some_property"
```

send_ip_definitions

```
send_ip_definitions
```

Description

The `send_ip_definitions` is called by the GUI after it has established the socket connection with the backend. When this command is invoked, all the IP Generator definitions present in the backend datamodel are sent to the GUI.

Example

```
send_ip_definitions
```

send_ip_gui_update

```
send_ip_gui_update <acxip_file_path> <acxip_data>
```

Argument	Optional	Type	Description
<acxip_file_path>		String	ACXIP filepath
<acxip_data>		String	The updated set of ACXIP property name-value pairs after the update function is run

Description

The `send_ip_gui_update` sends the refresh of the C++ datamodel from the backend, over the socket to refresh the GUI.

Example

```
send_ip_gui_update $acxip_file_path $acxip_data
```

set_ip_category

```
set_ip_category <category_list> [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
[-ip_name <ip_name>]	Y	String	IP name
<category_list>		Array	List of IP Library View categories

Description

The `set_ip_category` determines where the IP shows up in the IP Libraries View tree structure, and possibly in documentation structure.

Example

```
set_ip_category {"Core" "Memories" "Example Sub-Category"}
```

set_ip_description

```
set_ip_description <description> [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<code>[-ip_name <ip_name>]</code>	Y	String	IP name
<code><description></code>		String	IP description

Description

The `set_ip_description` sets the top level description of the IP. Shows up in tool tip in IP Libraries View, and as overview text in the documentation.

Example

```
set_ip_description "Description of Example Soft IP for Tooltip in IP Library View, etc"
```

set_ip_macro_name

```
set_ip_macro_name <macro_name> [-ip_name <ip_name>]
```

Argument	Optional	String	Description
<code><macro_name></code>		String	IP macro name
<code>[-ip_name <ip_name>]</code>	Y	String	IP name associated with this property

Description

The `set_ip_macro_name` is used to declare macro name. This is optional, and is used by any built-in output file generator TCL procedures.

Example

```
set_ip_macro_name "ACX_EXAMPLE_SOFT_IP"
```

set_ip_property_value

```
set_ip_property_value <name> <value>
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
<value>		String	ACXIP property value

Description

The `set_ip_property_value` is used to set ACXIP property value. This command can only be used inside the update IP configuration callback.

Example

```
ace_ip::set_ip_property_value "binaryParam1" $binaryParam1
```

set_ip_property_verilog_param_attributes

```
set_ip_property_verilog_param_attributes <property> [-format <format>] [-verilog_param <is_verilog_param>] [-to_upper <convert>] [-alt_param_name <alt_name>] [-num_bits <num_bits>] [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
<property>		String	IP property name
[-format <format>]	Y	String	Property representation format in verilog file. Valid formats are quoted_string, unquoted_string, sized_hex, sized_binary, sized_decimal
[-verilog_param <is_verilog_param>]	Y	Boolean	Indicates if the IP property is a verilog parameter
[-to_upper <convert>]	Y	Boolean	Convert verilog parameter name to uppercase
[-alt_param_name <alt_name>]	Y	String	Alternate verilog parameter name. Is set to IP Property name by default
[-num_bits <num_bits>]	Y	Integer	Number of bits to represent the value
[-ip_name <ip_name>]	Y	String	IP name associated with this property

Description

The `set_ip_property_verilog_param_attributes` is used to specify verilog parameter attributes for an IP property.

Example

```
ace_ip::set_ip_property_verilog_param_attributes "hexParam1" -verilog_param "false"
-to_upper "false" -num_bits 32
```

set_ip_resource_count

```
set_ip_resource_count <cell> <count>
```

Argument	Optional	Type	Description
<cell>		String	Resource type name
<count>		Integer	Resource count

Description

The `set_ip_resource_count` is used to update resource counts. This can be set dynamically inside the auto-correct/rule check function. This is not directly exposed to the user by default (you could expose it in a Calculated Value label). This is used by ACE to rule check if the current configuration consumes more resources than the target device has available.

Example

```
set_ip_resource_count BRAM72K [expr {$acxip_data(read_width) *
$acxip_data(addr_depth) / 72000}]
set_ip_resource_count MLP72 [expr {$acxip_data(read_width) *
$acxip_data(addr_depth) / 72000 / 2}]
```

set_ip_resource_types

```
set_ip_resource_types <list_of_cell_types> [-ip_name <ip_name>]
```

Argument	Optional	Type	Description
[-ip_name <ip_name>]	Y	String	IP name

Argument	Optional	Type	Description
<list_of_cell_types>		Array	List of Resource Types needed by this IP

Description

The `set_ip_resource_types` defines Resource Types needed by this IP. It determines which devices are supported. ACE looks at the primitive resource types supported (NAP_M, BRAM72K, etc) by a given device to determine if that device can support this IP.

If no resource type is defined, then we assume it is built out of pure synthesizable logic that is supported on all devices.

Example

```
set_ip_resource_types {"BRAM72K" "MLP72"}
```

set_update_ip_configuration_callback

```
set_update_ip_configuration_callback <tcl_proc>
```

Argument	Optional	Type	Description
<tcl_proc>		String	TCL proc name to be run

Description

The `set_update_ip_configuration_callback` is used to run a TCL callback procedure. This callback can be used to

- perform version migrations
- auto-correct property values
- add/remove, update enabled state or update calculated values of IP properties
- run rule checks
- update resource counts
- define Ports for generated Verilog/VHDL and interactive diagram
- update GUI page list for dynamic pages, and add/remove controls

Example

```
set_update_ip_configuration_callback example_ip_update_ip_configuration_proc
```

```
proc example_ip_update_ip_configuration_proc {target_device acxip_data
acxip_file_path} {
    # Version migration
    # Example:
    if {$acxip_data(version) == "1.0"} {
        set $acxip_data(some_property) "migrated_value"
        set $acxip_data(version) "1.1"
    }
    if {$acxip_data(version) == "1.1"} {
        message "Migrating ACXIP data from version 1.1 to 1.3..."
        $acxip_data(some_property) = "migrated_value"
    } elseif {$acxip_data(version) == "1.2"} {
    } elseif {$acxip_data(version) == "1.3"} {
    } else {
        # set_ip_rule_message <ip_property_name> <title> <status> <message> [-
ip_name <ip_name>]
        set_ip_rule_message "version" "ACXIP Version Compatibility Mismatch"
"Error" "Incompatible ACXIP version detected. The version in the ACXIP file is:
$acxip_data(version). This version of the ACE software supports version 1.0 to
1.3"
    }

    # Auto-correct property values
    # Example:
    if($acxip_data(some_property) > $some_value) {
        $acxip_data(other_property) = $new_value
    }

    # Run rule checks
    # Example:
    # Rule check #2
    if($acxip_data(write_width) != $acxip_data(read_width)) {
        # set_ip_rule_message <ip_property_name> <title> <status> <message> [-
ip_name <ip_name>]
        set_ip_rule_message "read_data_width" "Data width mismatch example" "Error"
"Write width must be set to the same value as Read Width. The current values are
different. Please fix this."
    }

    # Update enabled state of properties
    # Example:
    if($acxip_data(some_property) > $some_value) {
        set_ip_property_enabled_state "some_list_property" "true"
        set_ip_property_enabled_state "some_int_property" "false"
    }
}
```

```

# Update enum item allowed values
if($acxip_data(some_property) == "Yes") {
    set_ip_property_enum_items "some_list_property" {"NewItem" "Item1" "Item7"}
}

# Update Calculated Values
# set_ip_label <name> <value> [-ip_name <ip_name>]
set_ip_label "some_calculated_value" "New Value"

# set_ip_label_enabled_state <name> <state> [-ip_name <ip_name>]
set_ip_label_enabled_state "some_calculated_value" "false"

# Update resource counts
# This can be set dynamically inside the autocorrect/rule check function
# This is not directly exposed to the user by default (you could expose it in a
Calculated Value label). This is used by ACE to rule check if the current
configuration consumes more resources than the target device has available.
# set_ip_generator_resource_count <cell> <count> [-ip_name <ip_name>]
set_ip_generator_resource_count BRAM72K [expr {$acxip_data(read_width) *
$acxip_data(addr_depth) / 72000}]
set_ip_generator_resource_count MLP72 [expr {$acxip_data(read_width) *
$acxip_data(addr_depth) / 72000 / 2}]

#
#####
#####
# Define Ports for generated Verilog/VHDL and interactive diagram
# add_ip_generator_port <name> <signal_group> <direction> <width>
<diagram_side> <description> [-exposed <exposed>] [-tieoff <tieoff>] [-typedef
<typedef>] [-clock_domain <domain>] [-ip_name <ip_name>]

    add_ip_generator_port "ref_clk" "Clocks and Resets" "in" 1 "West" "Some
description Text for tooltip and documentation..."
    add_ip_generator_port "rstn" "Clocks and Resets" "in" 1 "West" "Some
description Text for tooltip and documentation..."

    add_ip_generator_port "data_in" "Data Interface" "in" 32 "West" "Some
description Text for tooltip and documentation..."
    add_ip_generator_port "data_out" "Data Interface" "out" 32 "East" "Some
description Text for tooltip and documentation..."
    if($condition){
        add_ip_generator_port "port_a" "Bidi Interface" "inout" 8 "West" "Some
description Text for tooltip and documentation..."
        add_ip_generator_port "port_b" "Bidi Interface" "inout" 8 "East" "Some
description Text for tooltip and documentation..."
    }

```

```

# Add/Remove IP Properties and calculated values as necessary
# Add function only adds the property if it doesn't already exist, otherwise it
silently succeeds
# Remove function only removes a property if it exists, otherwise it silently
succeeds
# Remove function also removes any GUI controls that are mapped to the removed
property
for {set i 0} {$i < $acxip(num_lanes)} {incr $i} {
    add_ip_bool_property "some_bool_property_index$i" "Boolean Property Example
Index $i" "Yes" -description "Long description text for tooltip" -enabled "false"
    # add_ip_calculated_value <name> <gui_label> <default_value> [-description
<text>] [-enabled <state>] [-ip_name <ip_name>]
    add_ip_calculated_value "some_calculated_value_index$i" "Example Dynamic
Label Index $i" "Hello World" -description "Long description text for tooltip"
-enabled "true"
}
for {set i $acxip(num_lanes)} {$i < 32} {incr $i} {
    remove_ip_bool_property "some_bool_property_index$i"
    remove_ip_calculated_value "some_calculated_value_index$i"
}

# Update GUI page list for dynamic pages, and add/remove controls
# Add function only adds the page if it doesn't already exist, otherwise it
silently succeeds
# Remove function only removes a page if it exists, otherwise it silently
succeeds
for {set i 0} {$i < $acxip(num_lanes)} {incr $i} {
    # You can put this into a separate TCL proc
    # add_ip_gui_page <name> <parent_page_name> <description> [-columns
<layout_columns>] [-ip_name <ip_name>]
    add_ip_gui_page "Page For Lane $i" "Overview" "Long description text for
page banner"
    add_ip_gui_control "top_level_int_a_property_index$i" "Page For Lane $i"
-control_type "combo"
    add_ip_gui_control "top_level_int_b_property_index$i" "Page For Lane $i"
-control_type "text"

    # add_ip_gui_group <name> <label> <page> [-parent <parent_group>] [-columns
<layout_columns>] [-span <column_span>] [-ip_name <ip_name>]
    add_ip_gui_group "group_x" "Example Group X" "Page For Lane $i" -columns 2
-span 1
    add_ip_gui_control "another_list_property_index$i" "Page For Lane $i"
-parent "group_x" -control_type "combo"
    add_ip_gui_group "group_a" "Example Group A" "Page For Lane $i" -parent
"group_x" -columns 1 -span 1
    add_ip_gui_group "group_b" "Example Group B" "Page For Lane $i" -parent
"group_x" -columns 1 -span 1

```

```

        # add_ip_gui_control <property_name> <page> [-parent <parent>] [-
control_type <type>] [-span <column_span>] [-ip_name <ip_name>]
        add_ip_gui_control "some_bool_property_index$i" "Page For Lane $i" -parent
"group_a"
        add_ip_gui_control "some_list_property_index$i" "Page For Lane $i" -parent
"group_b" -control_type "radio"

    }
    for {set i $acxip(num_lanes)} {$i < 32} {incr $i} {
        # You can put this into a separate TCL proc
        # add_ip_gui_page <name> <parent_page_name> <description> [-ip_name
<ip_name>]
        remove_ip_gui_page "Page For Lane $i"
    }

    # This sends the refresh of the C++ datamodel from the backend, over the socket
to refresh the GUI
    send_ip_generator_gui_update $acxip_file_path
}

```

update_ip_binary_property

```

update_ip_binary_property <name> [-number_of_bits <number_of_bits>] [-min <number>]
[-max <number>] [-enabled <state>]

```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-number_of_bits <number_of_bits>]	Y	Integer	Number of allowed bits
[-min <number>]	Y	String	Minimum allowed value
[-max <number>]	Y	String	Maximum allowed value
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_binary_property` is used to update an IP binary property constraints.

Example

```

update_ip_binary_property "example_binary_property_1" -number_of_bits 3 -min "000"
-max "111" -enabled "true"

```

update_ip_bool_property

```
update_ip_bool_property <name> [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_bool_property` is used to update an IP boolean property constraints.

Example

```
update_ip_bool_property "example_bool_property_1" -enabled "true"
```

update_ip_configuration

```
update_ip_configuration <ip_name> <acxip_data> <acxip_file_path>
```

Argument	Optional	Type	Description
<acxip_file_path>		String	ACXIP file path
<ip_name>		String	IP name
<acxip_data>		Array	List of ACXIP property-value pairs

Description

The `update_ip_configuration` is called by the GUI when the user changes a property. This passes data back to the GUI via the socket to update the GUI on rule check messages, the new state of ACXIP property values, list item updates, resource counts, enabled state, etc. This function also clears all the rule check messages prior to running the callback function (including on the GUI side). This function also clears all the port definitions, and you need to add the ports each time this is called, based on configuration.

Example

```
update_ip_configuration "Example Soft IP" {{prop1 val1} {prop2 val2} {prop3
val3}} $acxip_file_path
```

update_ip_enum_property

```
update_ip_enum_property <name> [-items <list_of_valid_values>] [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-items <list_of_valid_values>]	Y	Array	List of valid values
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_enum_property` is used to update valid values for IP enum property. It also allows enabling/disabling properties.

Example

```
update_ip_enum_property "some_list_property" -items {"NewItem" "Item1" "Item7"}
```

update_ip_file_path_property

```
update_ip_file_path_property <name> [-enabled <state>] [-file_must_exist <exists>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-file_must_exist <exists>]	Y	Boolean	Determine if file exists in the filesystem
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_file_path_property` is used to update an IP file path property constraints.

Example

```
update_ip_file_path_property "example_filepath_property_1" -enabled "true"
```

update_ip_float_property

```
update_ip_float_property <name> [-min <number>] [-max <number>] [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-min <number>]	Y	Double	Minimum allowed value
[-max <number>]	Y	Double	Maximum allowed value
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_float_property` is used to update an IP float property constraints.

Example

```
update_ip_float_property "example_float_property_1" -enabled "true"
```

update_ip_hex_property

```
update_ip_hex_property <name> [-number_of_bits <number_of_bits>] [-min <number>] [-max <number>] [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name

Argument	Optional	Type	Description
<code>[-number_of_bits <number_of_bits>]</code>	Y	Integer	Number of allowed bits
<code>[-min <number>]</code>	Y	String	Minimum allowed value
<code>[-max <number>]</code>	Y	String	Maximum allowed value
<code>[-enabled <state>]</code>	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_hex_property` is used to update an IP hexadecimal property constraints.

Example

```
update_ip_hex_property "example_hex_property_1" -number_of_bits 32 -min "00000000"
-max "FFFFFFFF" -enabled "true"
```

update_ip_int_property

```
update_ip_int_property <name> [-min <number>] [-max <number>] [-enabled <state>]
```

Argument	Optional	Type	Description
<code><name></code>		String	ACXIP property name
<code>[-min <number>]</code>	Y	Integer	Minimum allowed value
<code>[-max <number>]</code>	Y	Integer	Maximum allowed value
<code>[-enabled <state>]</code>	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_int_property` is used to update an IP integer property constraints.

Example

```
update_ip_int_property "example_int_property_1" -min "16" -max "24" -enabled "true"
```

update_ip_label

```
update_ip_label <name> [-value <value>] [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-value <value>]	Y	String	Calculated property value
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_label` is used to update an IP property calculated value.

Example

```
update_ip_label "some_calculated_value" -value "New Value" -enabled false
```

update_ip_signal_name_property

```
update_ip_signal_name_property <name> [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_signal_name_property` is used to update an IP signal name property constraints.

Example

```
update_ip_signal_name_property "example_signal_name_property_1" -enabled "true"
```

update_ip_string_property

```
update_ip_string_property <name> [-min_characters <number>] [-max_characters
<number>] [-valid_characters_regex <regex>] [-enabled <state>]
```

Argument	Optional	Type	Description
<name>		String	ACXIP property name
[-valid_characters_regex <regex>]	Y	String	Regular expression for checking validity of the property value
[-min_characters <number>]	Y	Integer	Minimum number of characters allowed
[-max_characters <number>]	Y	Integer	Maximum number of characters allowed
[-enabled <state>]	Y	Boolean	Flag to enable/disable property

Description

The `update_ip_string_property` is used to update an IP string property.

Example

```
update_ip_string_property "example_string_property_1" -min_characters "8"
-max_characters "24" -valid_characters_regex "[0-9].*" -enabled "false"
```

Chapter 6 : Troubleshooting

This chapter is intended to cover some areas where users frequently report problems, along with solutions. Your Achronix FAE likely can point you to a more recent FAQ.

This chapter will also cover some known ACE issues, with current workarounds where possible.

ACE Exit Error Codes

The following are some of the known exit codes that can be reported by ACE.

Code	Description	Solution
1	Generic error code; catchall for unexpected problem states.	Contact Achronix Technical Support
2	Invalid command line options for ACE or acx	See Running ACE (page 282) for a list of valid user command-line options. Contact Achronix Technical Support if problems persist.
3	License failure. ACE was unable to obtain a required license.	See the <i>ACE License & Installation Quickstart Guide (UG002)</i> for more details about configuring ACE license management. Contact Achronix Technical Support if problems persist.
5	GUI startup failure: incomplete installation: compatible Java release not found. This error typically indicates that the included version of Java used by the ACE GUI is improperly configured.	Contact Achronix Technical Support
13	GUI startup failure. This typically indicates that there's a problem in Linux with the LD_LIBRARY_PATH environment variable, though it can also occur due to crashes when loading the 64-bit WebKit2GTK+3 HTML browser.	Linux users: Unset the LD_LIBRARY_PATH environment variable and try running ACE again. Also, ensure there is a compatible 64-bit HTML browser (WebKit2 for GTK+3) installed. See Linux: Incompatible Default Web Browser (page 779) for more details, and if that does not help, contact Achronix Technical Support.

Code	Description	Solution
100 / 101	The GUI is attempting to workaround a number of known startup issues through automated forced restarts (which display these exit codes). If restarts fail and the GUI did not start successfully, this is an error.	Contact Achronix Technical Support if the GUI did not start.
values ≤ 136	Various license management error codes from RLM.	See the <i>ACE License & Installation Quickstart Guide (UG002)</i> for more details on ACE license management. Contact Achronix Technical Support if problems persist.
201	The GUI detected a socket communication error with the acx backend.	Contact Achronix Technical Support
202	When the GUI was attempting to exit gracefully (due either to user request or a fatal error), critical errors occurred, forcing the GUI to perform a hard kill of itself.	Contact Achronix Technical Support if problems persist.
203	The GUI is unable to start due to underlying framework errors.	Contact Achronix Technical Support. (This is most likely due to an attempt to execute ACE in an unsupported OS.)
404	ACE was attempted to run on an obsolete OS; the GUI is known to be incompatible, so this is disallowed.	Run ACE on a supported operating system.
504	An error occurred while interpreting the Tcl script file passed to ACE.	The Tcl script contains errors or encountered an unhandled error condition. Either fix the bug in the Tcl script, or enhance the error handling in the Tcl script to better handle/report the error condition. Contact Achronix Technical Support if problems persist.
505	No home directory is defined for the current user. By default, ACE places log files and occasional temp files in locations under the user's home directory, so when no home directory is defined, ACE is unable to proceed safely.	Define a home directory for the userid which starts ACE, or change to a userid with a valid home directory before starting ACE. Consult a local system administrator if necessary.

Code	Description	Solution
506	ACE is unable to open a socket connection between the GUI and the acx backend. (Specifically, the acx backend is unable to bind a socket port needed for communications with the GUI.)	See the Troubleshooting section below titled: Startup Error — ACE is Unable to Connect on Port NNNN of Localhost (page 767)
507	The ACE acx backend detected an unexpected GUI socket closure, likely due to a fatal GUI error, when then has caused the acx backend to exit.	Contact Achronix Technical Support

Duplicate Names for Arrays

If the following error message is seen during the prepare stage of ACE, it indicates the occurrence of a duplicate net name in the RTL. The RTL must be modified to clear the error.

```
ERROR: int_cnt[1] is already declared (VNLR-1044)
```

Note

This situation only occurs when one of the duplicates is a single-dimensional array, and the other is a two-dimensional array.

Clock Definitions/Constraints

At least one clock must be defined. Clocks should not be redefined.

Asynchronous Reset of I/O from the Core

If an I/O is not clocked by a boundary clock, use synchronous reset only.

Multi-process Functionality License Requirements

Multi-process functionality requires a license for each background process; therefore users with a single license cannot access this functionality. When encountering this limit, contact your FAE for current workarounds.

Note

Node-locked licenses support multi-process flows without issues. Floating licenses require a new license for each process.

Non-ASCII Characters in Path

Do not use non-ASCII characters in paths. For example, if the username includes German extended characters (e.g. umlauts), Chinese, etc., ACE might function incorrectly. To remedy this, ensure that all paths only contain ASCII characters.

Unable to Load Project: Project is Locked

Example error message for locked project

```
cmd> restore_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1"
Project: "~/output/quickstart/quickstart.acxprj" is locked by another ACE session and
cannot be loaded. This project is locked by user: Docs on host: hostname. You can use
restore_project -force to override the lock. To manually unlock this project, delete the
lock file: ~/output/quickstart/quickstart.lock
cmd>
```

ACE locks the **Project File** (page 222)s every time it loads a project to prohibit corruption of the project's data. If project locks were not used, and more than one ACE session was allowed to open the same files (or write to the same files) simultaneously, the results would be inconsistent and project data files could become corrupted.

⚠ Caution!

Do not use this forced unlock procedure to run multiple ACE sessions on the same project simultaneously, or file corruption might occur!

Project definitions, Implementation definitions, saved implementation states (for both normal flow and incremental compilation), log files, and output directories might get corrupted from having two or more ACE sessions writing to the same files.

Most notably, do not start another ACE session on a project while Multiprocess is already running on that same project; the Multiprocess session must own the project lock (which also locks all implementations) to ensure consistent results and avoid file corruption.

Achronix does not support simultaneously running multiple ACE sessions on the same project (directory). This is known to cause problems. Do not do this! The project lock files are there to protect users; do not attempt to bypass them.

In rare cases, ACE can crash, mistakenly leaving a project in the locked state. The easiest way to unlock a mistakenly-still-locked project (which has just failed to load in the GUI) is the following:

1. Double-check that there are no other legitimate users of the project file, including yourself in another desktop!
2. In the **Tcl Console View** (page 142), click on the empty **cmd>** line.
3. Click the up arrow (↑) on the keyboard. Each click of the up arrow moves one step backwards through the Tcl command history. Keep moving backwards through the Tcl command history until the Tcl command which attempted to load the locked project is displayed. The failed command should be a call to **load_project** (page 667) or **restore_project** (page 688).

Note

If you regularly load multiple Projects into ACE at the same time, you might have to go back several commands to reach the one that failed.

4. Move to the end of the Tcl command line (press the **End** key on the keyboard), press the space bar, then add the argument **"-force"** to the end of the command. This argument forces ACE to load (or restore) the project despite the presence of a lock, and this session of ACE re-locks the project, taking ownership using a new lock.

Example workaround

```
cmd> restore_project "~/output/quickstart/quickstart.acxprj" -activeimpl "impl_1"
-force
```

5. Press the **Enter** key to issue the Tcl command to load or restore the project.

Changing ACE Font Sizes

Fonts in Views

Some views in ACE (particularly diagrams) allow users to specify the exact font/size which will be used. These are typically defined in the ACE preferences, under **Window** → **Preferences** → **General** → **Appearance** → **Colors and Fonts**. There are categorical groupings for each such view allowing its own font and color definitions.

Views that lack their own specific font definition will inherit the font definitions in the **Basic** category, almost always using the **Dialog Font**. A very few views may also use the **Banner Font** and **Header Font** (also found in the **Basic** category) for limited rendering.

The **Dialog Font** used in most ACE views is directly inherited from the underlying GUI application framework stack, (typically called the **Application Font** in Linux/GTK, or **Message Box Font** in Windows).

The various views within ACE often accept font changes immediately, but, in some cases, it might be necessary to restart ACE to see the font changes inherited from the OS propagate completely as the new **Dialog Font** value throughout the application.

⚠ Caution!

It is highly recommended that a plain font (not bold, not italics) be chosen as the **Dialog Font**. There is no font naming standard followed across all OS flavors (or even within a single OS), so these plain font varieties are often called "Regular", sometimes "Book", sometimes "Medium", and sometimes don't have a particular designation.

Linux:

The config location can vary in every Linux desktop / version / distro. In Linux, ACE uses the GTK+ widgets and fonts, so changes should be restricted to those settings.

For example, in the RHEL/CentOS 7.x KDE Plasma desktop, it can be configured from the main desktop (not ACE) menu under **System Settings** → **Application Appearance** → **GTK+ Appearance** → **GTK+ Fonts**.

As another example, in the RHEL8 Gnome3 desktop, it can be configured from the Gnome Tweaks tool (gnome-tweaks), under **Tweaks** → **Fonts** → **Interface**.

Windows:

In Windows 10 and 11, users are no longer allowed direct control of the font at a system-wide level through the **Control Panel**. (Though some third-party tools still allow this to be configured). Windows users will be best served by changing the **Basic** → **Dialog Font** value directly in the ACE **Colors and Fonts** preferences, as mentioned in the beginning of this troubleshooting entry.

Fonts in HTML Reports

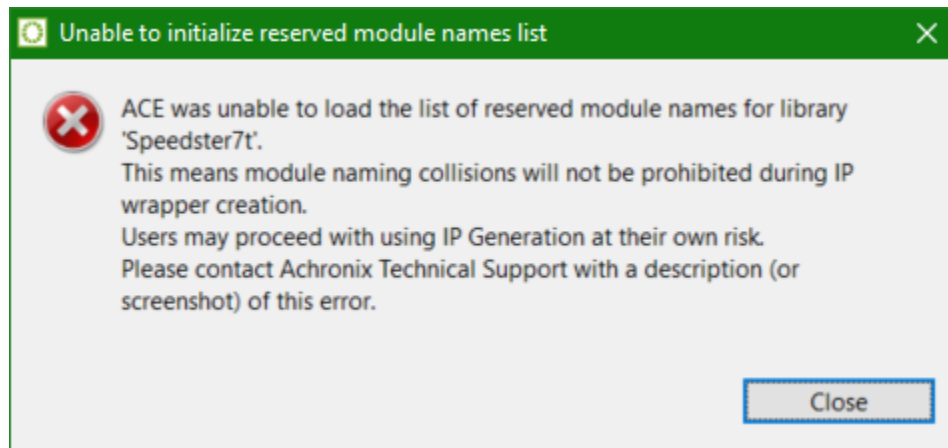
The font used in the HTML Reports is directly inherited from the system's HTML browser's font settings. The system HTML browser can be Internet Explorer (Windows) or WebKit2GTK3 (Linux). The HTML reports typically use the "Proportional" (sometimes called "Web Page") font and Monospace (also called "Plain Text") font types — change these settings in the operating system's HTML browser settings to make the fonts change in the ACE HTML Report viewer.

Unable to Initialize Reserved Module Name List

This problem is reported by the ACE GUI at startup with the following dialog.

i Note

The exact library name, "Speedster7t" in the screenshot, differs based upon the customer's specific license/installation.



Typically this error indicates that the device overlay files have been installed incorrectly. The list of reserved module names is contained in the overlay files for each Achronix device/library. In very rare cases, the file may exist in the expected location but ACE might lack read permissions for the file.

The file ACE is attempting to read should be located at:

```
<ace_install_directory>/libraries/<library_name>/reserved_module_names.txt
```

where *<ace_install_directory>* is the directory where ACE has been installed and the *ace* executable is found, and *<library_name>* is the name of the library from the error dialog (though forced to all lowercase letters).

Thus, if the Linux user "tester" has installed ACE in their home directory under

```
/home/tester/ace/Achronix-linux
```

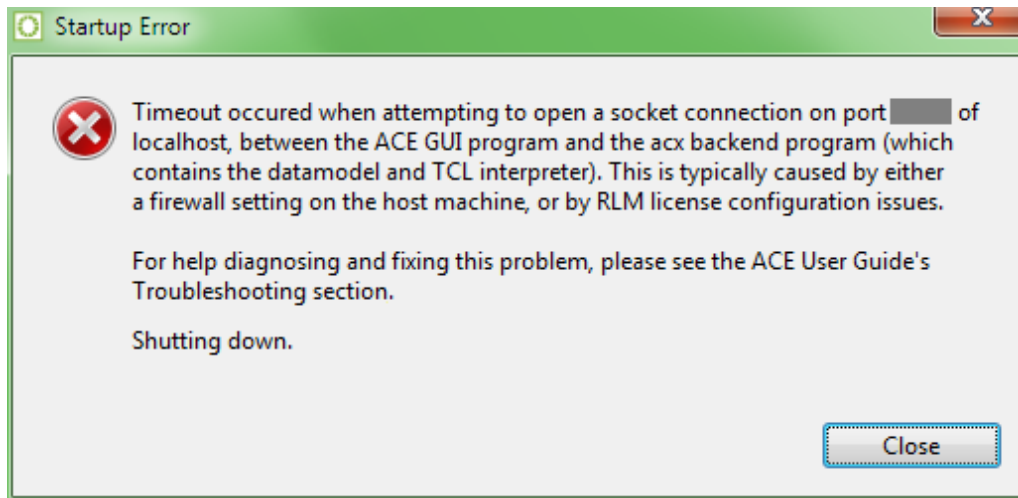
and saw the error dialog captured above about the Speedster7t library, then ACE was unable to find (or potentially experienced permission issues when attempting to read) the file:

```
/home/tester/ace/Achronix-linux/libraries/speedster7t/reserved_module_names.txt
```

To solve the problem, contact Achronix technical support for help on correctly installing both ACE and the associated device overlay file(s) in the appropriate location.

Startup Error — ACE is Unable to Connect on Port NNNN of Localhost

This problem is reported by the ACE GUI at startup with the following (rather verbose) "Startup Error" dialog:



To Determine the Root Cause

Attempting to run ACE in batch mode (with `ace -batch`, see [Running ACE \(page 282\)](#)) can help determine the source of the problem. In batch mode, the GUI is not used; therefore, no socket connection is needed between the GUI and the acx backend.

If batch mode ACE does not start, or takes more than 60 seconds before providing the Tcl command prompt (regardless of whether or not ACE reports license errors), then there's typically a licensing configuration problem.

If batch mode ACE starts successfully in less than 60 seconds, then there is a firewall configuration problem.

In very rare cases, usually when ACE is installed in a remote NFS location with many device overlays, poor filesystem (HDD) or network (NFS) performance may also cause these timeouts. In these rare cases, simply trying to start ACE in GUI mode a second time will often succeed where the first attempt had a timeout, because for the second attempt the information will now be found in the filesystem cache in memory, which is much faster to access.

⚠ Caution!

Though unlikely, realize it is possible for multiple causes to occur for the same user. It might be necessary to first fix a firewall problem, and then fix a licensing problem. After attempting one kind of fix, if the dialog continues to appear, re-diagnose the problem to verify whether the same issue is still occurring, or whether a new issue is occurring but showing the same symptom.



When batch mode has started, you can exit ACE batch mode by typing "exit" or "quit" and pressing Enter.

If it is a Firewall Problem

ACE always uses localhost (IPv4 address 127.0.0.1) TCP sockets to communicate between the ACE GUI program and the ACE backend program `acx` (the pure textual TCL interface when ACE is started with '`ace -b`' or '`ace -batch`'; see [Running ACE \(page 282\)](#) for details). In most cases, the required localhost sockets are already configured to be available, but in highly restricted network environments, the required localhost sockets might need special

permissions to be added to the firewall of the workstation running ACE. Obtain help from your local system administrator and/or network administrator to properly configure the firewall to allow the necessary network traffic.

Note

If the system/network administrator needs to know exactly which executables require firewall permissions, tell them:

(Windows)

- `<ace_install_dir>\system\gui\ACE_GUI_Launcher.exe`
- `<ace_install_dir>\system\cmd64\acx.exe`

(Linux)

- `<ace_install_dir>/system/gui/ACE_GUI_Launcher`
- `<ace_install_dir>/system/classic_gui/ACE_GUI_Launcher`
- `<ace_install_dir>/system/cmd64/acx`

By default, ACE uses an automatically chosen unused (free) TCP port socket somewhere in the range 1024-65535. This automatically chosen port number can change every time ACE is started.

To force ACE to use a specific TCP port number (and thus require only a single hole poked in the firewall for each executable), add the `"-acxport <portnumber>"` commandline option when starting ACE (where `<portnumber>` is the desired TCP port number in decimal). Ask your network or system administrator exactly which port number should be specified — this is the same port number the administrator opened for each executable in the firewall.

Please contact Achronix Technical Support if problems persist.



In some licensing configurations, additional firewall ports might need to be opened specifically for the licensing software. Refer to the *ACE Installation and Licensing Guide* (UG002) for details.

The following executables may require firewall access for the licensing software:

(Windows)

- `<ace_install_dir>\system\cmd64\acx.exe`
- `<ace_install_dir>\ace.exe`

(Linux)

- `<ace_install_dir>/system/cmd64/acx`
- `<ace_install_dir>/ace`

If it is a Licensing Problem

A socket connection timeout can also occur when the GUI is attempting to open a connection to the acx backend before the acx backend is ready. This situation can occur if the acx backend is having trouble finding licenses for ACE itself, or for the FPGA/eFPGA devices currently installed for ACE. This long, slow license search is most often seen when ACE is configured to find its license on one or more license servers, and one of those specified servers is not actually a license server (typos and license server migrations being the frequent causes).

Refer to the *ACE Installation and Licensing Guide* (UG002) for details on how to determine the workstation's current RLM license configuration, and how to diagnose and fix what might be going wrong. Frequently, it is simply a matter of correcting or removing the incorrect license server setting in the "RLM_LICENSE" environment variable. Contact Achronix Technical Support if problems persist.

Workaround: Extending the Timeout

This workaround is not generally recommended, but in cases where there are transient networking or license server problems, it might be necessary to extend the ACE socket initialization timeout value to be more permissive. To extend the timeout, set the environment variable "ACX_GUI_INIT_TIMEOUT_SECONDS" to a decimal number > 60. Invalid numeric values cause a non-fatal warning to be logged, and the default value of 60 seconds will be used instead. Please work with your IT department for help configuring the environment variable such that it will be set for every ACE startup.

If it is a Filesystem or Network Performance Problem

Again, this is rarely the case – licensing configuration problems are much more likely. But if it proves to be a performance problem, installing ACE locally, particularly on an SSD (instead of an HDD), will eliminate network filesystem performance problems. If a local installation is impossible, there are some potential workarounds available.

Workaround: Extending the Timeout

It might be necessary to extend the ACE socket initialization timeout value to be more permissive. To extend the timeout, set the environment variable "ACX_GUI_INIT_TIMEOUT_SECONDS" to a decimal number > 60. Invalid numeric values cause a non-fatal warning to be logged, and the default value of 60 seconds will be used instead. Please work with your IT department for help configuring the environment variable such that it will be set for every ACE startup.

Alternate Workaround: Multiple Parallel ACE Installs

Alternately, if there are too many device overlays installed at the same time (the exact value of "too many" will vary based on drive and network performance, but so far this problem has only been reported with more than six device overlays installed on a slow NFS mount), users could install ACE in two (or more) separate locations, with a subset of the device overlays applied in each install location. This would reduce the amount of device (and related license) processing that occurs at ACE startup, to the extent that a timeout will no longer occur.

Unexpected ACE GUI problem: java.lang.NoClassDefFoundError: com/achronix/api/APIPlugin

This typically indicates that a local file cache (maintained by the frameworks underlying ACE) has become corrupted. There are a variety of potential causes, with the most frequent being if network file system (NFS) problems occur when programs (like ACE) access the user's home directory.

Thankfully the fix for ACE is relatively simple; start ACE with an additional command-line argument: "-clean". This will cause ACE to immediately discard the old file cache as it starts, and rebuild the cache from scratch.

Note that it is not recommended to use the -clean option every time ACE is started, because rebuilding the cache can significantly increase ACE startup times.

```
ace -clean
```

If the user normally starts ACE from an application/start menu or a desktop shortcut (instead of from a command prompt), and/or the user does not know how to start ACE from the command prompt, attempt the next recommended fix, described below.

If the problem persists after -clean, or if the command prompt is unfamiliar

There's a more drastic and thorough solution, manually deleting a sub-directory under the user's home directory. Be warned, this not only clears the cache, but also resets all ACE GUI preferences to their default values, so this is not recommended unless the "-clean" option has already been attempted.

The ACE GUI filecache is stored in the user's home directory under a sub-directory named ".achronix", with a separate sub-directory used by each version of ACE, named with the prefix "workspace_" and the suffix being the version number of ACE. For example, when using ACE v9.1.1, the sub-directory will be named "workspace_9.1.1".

The full path for our example would be "<user_home_directory>/.achronix/workspace_9.1.1". Delete this subdirectory, and ACE should then be able to start normally.

Detailed instructions:

1. Open your preferred graphical file manager.
 - a. Linux: The exact program name will vary by the user's choice of Desktop Environment, but the five most commonly used program names would probably be Files, nautilus, dolphin, thunar, or caja.
 - b. Windows: the program provided by Microsoft for all versions of Windows is File Explorer
2. Navigate to your home directory.
 - a. Linux: For example username "acxuser", this would typically be the directory "/home/acxuser/". (Replace "acxuser" with your own username.)
 - b. Windows: For example username "acxuser", this would typically be:

 , also known as "C:\Users\acxuser". (Replace "acxuser" with your own username.)
3. Double-click (or expand) the sub-directory (under your home directory) named ".achronix".
4. Find the sub-directory under ".achronix" with the prefix "workspace_" and the suffix matching the version of ACE that fails to start. Continuing our prior example, for ACEv9.1.1, this would be the directory name "workspace_9.1.1". Right-click that workspace directory and in the resulting context menu, choose "Delete".

The ACE workspace directory (containing the corrupted file cache) has now been deleted/cleaned. The next time ACE is started, the file cache (and the rest of the workspace) will be recreated from scratch, with default values.

(If the same error message occurs the next time ACE is started, even after you've deleted the workspace directory, contact Achronix technical support.)

Multiprocess Summary Report Shows "No Timing Results Found" for Successfully Run Implementations with Existing Timing Reports

In cases of high network file system read/write latency, it is possible that the multiprocess system might not find the required timing information within the allowed period after the external process has completed execution (most likely to occur when using external job submission systems). Sometimes the file writes occurring on the remote machine might be cached for a while, and not immediately written to the NFS drive, so that when the ACE Multiprocess system notices that the spawned process has completed, it does not find the needed timing information on the NFS drive. Therefore, the file read attempt times out, and Multiprocess gives up looking.

For these cases, there is a user preference on the [Multiprocess: Configure Custom Job Submission Tool Preference Page \(page 206\)](#) called **Allowed seconds of NFS write latency**. To fix the problem, increase the value of this preference to allow more time between the completion of the implementation's flow process and when ACE gives up looking for the expected timing information for that completed implementation.

Windows: ACE Incorrectly Reports Read/Write File Permission Problems

In some cases, the ACE GUI might report a file permissions error when attempting to read or write a file on a network drive, when the permissions should actually allow the attempted read or write. As a workaround, please move the affected project to a local, non-network drive location.

Windows: ACE GUI Shown as "Not Responding"

In rare cases, when the Floorplanner Perspective is first changed, the ACE GUI window can become solid grey, and the title bar can change to read something like "**ACE - Achronix CAD Environment - designName - implementationName - (deviceName) - (Not Responding)**". When this problem has been reported, it has always been the case that the Floorplanner is taking too long to repaint.

The Windows operating system requires that applications check-in every five seconds, or the application is deemed non-responsive. Non-responsive applications are given a figurative kick-in-the-pants by Windows, and asked to repaint the screen. When the screen paint itself is taking more than five seconds, as can happen with poor Floorplanner Optimization settings, an application can be forced into an effective infinite-loop of paint requests from the operating system.

If, in Windows, it is ever noticed that the ACE GUI is being called non-responsive by Windows (check the application title bar), ACE has most likely entered this looped painting state. To escape this state, change back to the Project perspective (or any other perspective without the Floorplanner view visible), then navigate to the [Floorplanner View Optimizations Preference Page \(page 200\)](#), (**Window** → **Preferences** → **Floorplanner View Optimizations**) and ensure that **Enable Incremental Rendering** and **Render large areas as smaller tiled areas** are both enabled for the current design's complexity level.

Note

Both are enabled by default for everything except trivial designs. Press the **Restore Defaults** button to return to the default settings. If both are already enabled, and the non-responsive state still occurs, please call Achronix Technical Support for guidance on further Floorplanner Optimization tweaks.

Windows: Garbage sometimes appears in the Floorplanner View during panning operations (and remains after panning is completed)

In some cases customers are reporting Floorplanner render errors during/after panning operations.

While Achronix works to fix these rendering errors, there are two potential workarounds to mitigate the render problem.

- Whenever render errors appear in the Floorplanner, first ensure the Floorplanner View has the application focus, then press the backtick (`) key on the keyboard to force the Floorplanner to perform a complete repaint of the full render area.
- In the Floorplanner preferences (**Window** → **Preferences** → **Floorplanner View Optimizations**) enable the **When panning, show only background layer** checkboxes for both the **High** and **Medium** complexity columns. Now, when the panning operation completes, the full view will be automatically re-rendered, eliminating any mid-pan render errors.

Windows: ACE Startup Error Due to Missing DLL Component in Windows 10

In some Windows 10 configurations, the following error dialog might appear when invoking the ACE GUI. This error occurs due to a missing DLL component from the Visual Studio redistributable installer. This situation can be resolved by reinstalling the `vc_redist.x64.exe` executable. This executable can be downloaded from the following link: <https://www.microsoft.com/en-ca/download/details.aspx?id=48145>.

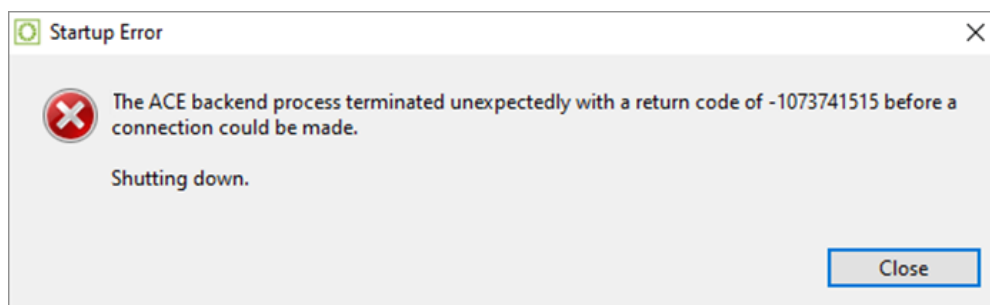


Figure 310 - ACE Startup Error

Windows: The icons and buttons in ACE are too small

There are two main ways to deal with this. You can ask Windows to scale everything, which will affect all applications (not just ACE), and if that's not good enough, you can additionally make just ACE further alter the icon (and potentially font) scaling.

Asking Windows to upscale images and fonts for all applications

See the Microsoft documentation here for related information: <https://support.microsoft.com/en-us/windows/make-text-and-apps-bigger-c3095a80-6edd-4779-9282-623c4d721d64>

Asking just ACE to upscale images and fonts

ACE already scales fonts (and to a lesser extent images) according to what Windows tells it to do, following the settings from the OS, as described above. But by default ACE uses a very coarse scaling granularity for images, and only doubles or triples icon sizes (like 200% or 300%), and does not upscale images/icons by smaller fractions (like 125% or 150%). This is because the fractional scaling can reportedly cause the (Eclipse) frameworks underlying ACE to crash with some video drivers. Thus to maximize stability, ACE disables framework-based fractional scaling by default.

Windows itself has more advanced scaling in the Compatibility Settings area which some users may prefer, but this can also make fonts and images blocky or blurry, so ACE does not enable these settings by default either.

Users having trouble due to the icons/buttons being too small may choose to try either alternate scaling option below (or both in combination) **at their own risk** .

Asking Windows to alter the scaling settings for just ACE (which may make text and fonts blurry/blocky)

Close ACE if it is running. Open **Windows Explorer** and navigate to the directory where ACE was installed. (By default this will be C:\Program Files\Achronix CAD Environment) In that directory find the `ace` executable, right-click that file, and select **Properties**. A dialog titled **ace.exe Properties** will open. Turn to the **Compatibility** tab, and near the bottom of the dialog press the **Change high DPI settings** button, which will cause a new smaller dialog to appear, also titled **ace.exe Properties**. Near the bottom of this dialog select the checkbox **Override high DPI scaling behavior. Scaling performed by:** and then in the combobox below, select the value **System (Enhanced)** instead of the default value of **Application**. Press the **OK** button to close the small properties dialog, and then press the **OK** button to close the larger properties dialog as well. Start ACE, and observe that image and font scaling may now occur in a more granular/fractional fashion, though things may now be blurry and/or blocky.

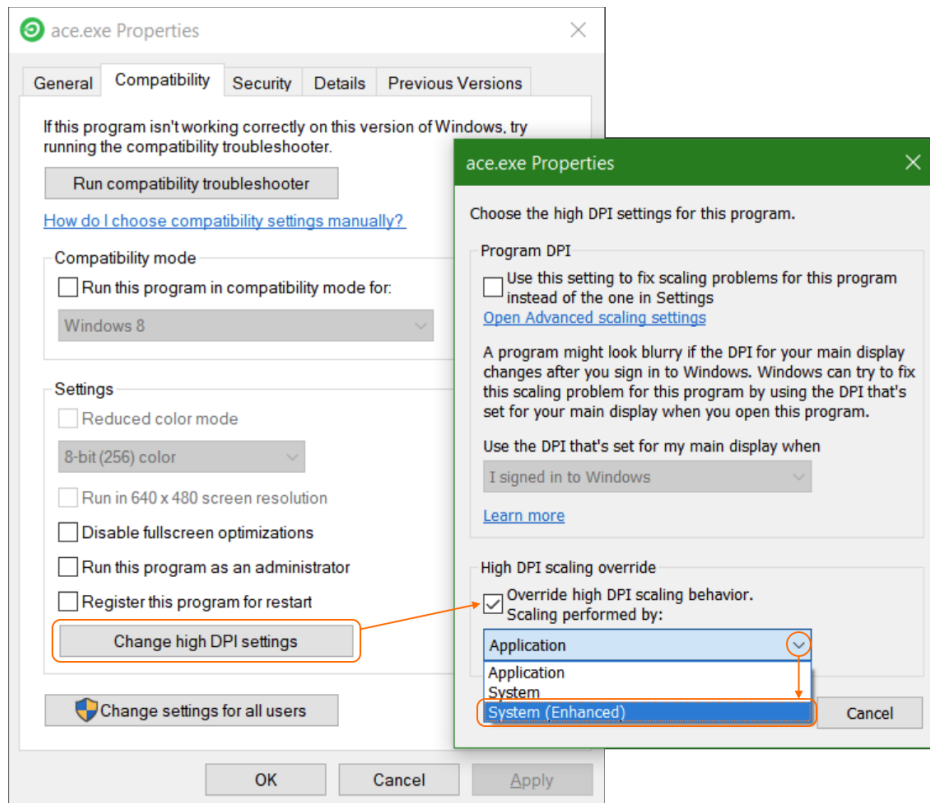


Figure 311 • Screenshot Showing the Sequence to Change the DPI Settings

To revert this change, disable this supplemental scaling functionality and return ACE to the default behavior:

1. Close ACE if it is running.
2. Open **Windows Explorer** and navigate to the directory where ACE was installed. (By default this will be C : \Program Files\Achronix CAD Environment)
3. In that directory find the `ace` executable, right-click that file, and select **Properties**. A dialog titled **ace.exe Properties** will open.
4. Turn to the **Compatibility** tab, and near the bottom of the dialog press the **Change high DPI settings** button, which will cause a new smaller dialog to appear, also titled **ace.exe Properties**. Near the bottom of this dialog change the setting **System (Enhanced)** back to the default of **Application**, and then deselect (uncheck) the checkbox **Override high DPI scaling behavior. Scaling performed by:**
5. Press the **OK** button to close the small properties dialog, and then press the **OK** button to close the larger properties dialog as well.

The next time ACE is started the scaling will be back to the default behavior.

Enabling ACE's application framework's fractional scaling (fonts remain crisp, but icons may become blurry and ACE may experience instability with some video drivers)

Close all running instances of ACE. Open the file

`<ace_installation_directory>\system\gui\ACE_GUI_Launcher.ini` in a text editor (like notepad), and at the bottom of the file add a new line:

Add the following new line to the bottom of the file ACE_GUI_Launcher.ini

```
-Dswt.autoscale=quarter
```

And then save the changes to the file. Note that saving the file may require Administrator access; work with IT or MIS personnel to make this change if required.

The next time ACE is started, this will allow ACE to scale images and icons to the more granular fractional values requested by Windows, like 125%, 150%, 175%, 200%, 225%, etc.

If after making this change, some images within ACE vanish, or if ACE experiences crashes (or even silently vanishes), simply remove that new line from the bottom of the `ACE_GUI_Launcher.ini` file to return to the prior (much more stable) behavior.

Linux: Resource Limits: ACE Reports an OutOfMemory Error, But There is Plenty of Free Memory Available

Note

When an OutOfMemory error is reported by ACE, always verify that there is sufficient physical and virtual memory available to run ACE. Running out of physical or virtual memory is the true cause of the error message in >95% of the cases reported.

Consult an Achronix FAE if the ACE memory requirements are unknown for the available licensed Achronix target devices.

There are several types of resource exhaustion in Linux that can be reported as an OutOfMemoryError. Insufficient thread and/or file resources may have similar error reports in some cases.

In some OS configurations, Linux can create a new thread for each file opened, so even when thread resource limits are mentioned in the detailed error message, it could be a case where the file limits (max files open simultaneously) are set too low for the user.

A quick fix attempt would be, before starting ACE, close all other running programs which could have files open. If this works, then the file limits are very likely the problem. But even if this doesn't help on the first attempt, the root problem could still be due to file limits.

In the bash shell, (other shells use different, but often similar, commands,) this is typically managed with the 'ulimit' command. All current ulimit settings can be queried using 'ulimit -a', or query just the open file limit with 'ulimit -n'.

To see if a higher file limit helps, if (for example) the current open files value is 1000, try raising it to 2000, then run ACE. To increase the file limit in this way using the bash shell, use the command 'ulimit -n 2000'. It might be possible to remove the limit entirely with 'ulimit -n unlimited'. (Again, users of other shells need to use a different command.)

It is highly recommended that these file limit changes be performed under the supervision of the system administrator. System administrators often apply upper bounds for such ulimit assignments, and individual users cannot exceed those upper bounds without system administrator assistance.

If raising the open files limit does allow ACE to launch correctly, the new raised limit should be applied to the user's '~/.bashrc' (or similar) files loaded at shell startup, again with help from the system administrator if necessary. (Alternately, if permissions allow it, create a script used to start ACE. In that script, the file limit could be temporarily raised before starting ACE, and then lowered again when ACE completes execution.)

Linux: In the TWM Window Manager, the First Time the ACE GUI is Started After Installation, the ACE Window is So Small Users Might Not See it

Currently, twm is ignoring the ACE GUI's attempts to set its own initial application window size and location. After ACE is installed (and until the window is moved and resized), the ACE window is in the upper-left corner of the screen, with tiny dimensions (we've seen it as small as 7 pixels wide by 7 pixels tall). This tiny window is often not noticed, especially if there already is a minimized application icon in that region of the screen.

When ACE is running in that tiny window, the ACE window can be moved and enlarged in the same manner as with any other running application window in twm.

ACE does not support the twm standard of choosing the application window's position and dimensions at startup with command-line arguments. Instead, ACE remembers the position and dimensions at application shutdown, and the next time the application is started, ACE returns to that same position and dimension from the last ACE session.

Linux: Odd Behavior When Using X DISPLAY Forwarding if the X Client and X Server Are More than One Major Revision Apart

When running the ACE GUI, the host workstation and display workstation must be at most one OS major revision apart (RHEL7 can talk to RHEL8, but RHEL7 should not talk to RHEL9).

RHEL only supports X DISPLAY redirection across adjacent major operating system revisions. There are known problems when (for example) applications like the ACE GUI are running on RHEL7 but having their X DISPLAY redirected to a RHEL9 workstation, or vice-versa. Users attempting to bridge multiple OS revisions in this way see GUI painting errors and mouse handling errors, especially for drag-and-drop operations. Some users have also reported hung GUIs and application crashes when they attempted to host ACE on RHEL7 and display on RHEL9.

Because the operating system vendor does not support this behavior, Achronix is unable to support it.

Linux: ACE Menus Do Not Show Icons Next to the Action Names

Most actions within ACE are intended to have an associated graphical icon. This icon is able to be displayed in the drop-down menus within ACE. If no icons are displayed next to actions in menus, this behavior is caused by a GTK+ configuration that has disabled the icons.

To re-enable icons for all GTK+ applications (not just ACE), the following command should reset the display. As this issue is the result of GTK+ functionality, and not ACE functionality, this tweak is not officially supported by Achronix.

```
gconftool-2 --type boolean --set /desktop/gnome/interface/menus_have_icons true
```

Linux: ACE Ignores LD_LIBRARY_PATH

In the majority of cases, the ACE GUI crashes when it encounters custom/obsolete libraries through an assigned LD_LIBRARY_PATH environment variable.

Because of this, by default ACE intentionally ignores preassigned LD_LIBRARY_PATH values when it starts.

Caution!

Achronix does not support ACE when forced to run using LD_LIBRARY_PATH.

At your own risk, add the command-line argument "`-enable_ld_library_path`" to force ACE to keep the preassigned LD_LIBRARY_PATH value at startup.

```
./ace -enable_ld_library_path
```

Linux: ACE forces the use of the GDK X11 rendering engine even while running under Wayland

Technically this is not a bug, but an intentional design choice. But because some users prefer the new features in Wayland despite the known stability of X11 on all Achronix-supported versions of Linux, we'll include an entry for this detail.

At present, the Eclipse framework underlying ACE has a high number of bugs when rendering natively under Wayland. Mouse actions, particularly relating to drag-and-drop, are especially problematic. Custom diagram rendering performance (as used in the Floorplanner, for example) has been observed to be much slower under Wayland.

To work around these limitations while the Eclipse framework developer community learns how to best use GTK/GDK under Wayland, Achronix currently forces ACE to render in GTK/GDK using X11 programming interfaces at all times, even while running on Wayland.

If users really want to allow the Eclipse frameworks underlying ACE to render GTK/GDK natively under Wayland, they do so at their own risk – this is unsupported by Achronix due to all the known framework issues.

UNSUPPORTED call to allow ACE to use the known-buggy Wayland GTK/GDK rendering

```
./ace -allow_wayland
```

Note that using the above startup command does not force ACE to use Wayland. There's a large number of details that must already be in place before using Wayland rendering is even possible, which are outside the scope of this document. When ACE is started in this manner, the Eclipse frameworks underlying ACE will automatically choose whether to use X11 or Wayland based on a number of factors, also beyond the scope of this document. This command simply ensure ACE will no longer enforce the use of X11 rendering in all cases.

To help convince any optimists that it truly is a bad idea to start ACE this way, the list of the Eclipse framework's Wayland bugs can be viewed online at the links below.

(recently reported issues:)

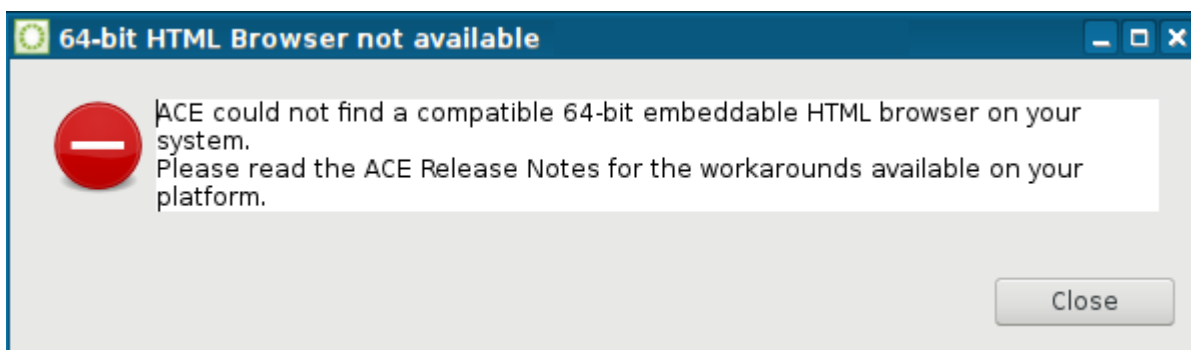
<https://github.com/eclipse-platform/eclipse.platform.swt/issues?q=is%3Aissue+label%3AWayland+>

(older issues:)

https://bugs.eclipse.org/bugs/buglist.cgi?classification=Eclipse%20Project&component=SWT&f0=OP&f1=OP&f2=product&f3=component&f4=alias&f5=short_desc&f6=status_whiteboard&f7=content&f8=CP&f9=CP&j1=OR&o2=substring&o3=substring&o4=substring&o5=substring&o6=substring&o7=matches&order=changeddate%20DESC%2Cbug_status%2Cpriority%2Cassigned_to%2Cbug_id&product=Platform&query_format=advanced&v2=wayland&v3=wayland&v4=wayland&v5=wayland&v6=wayland&v7=%22wayland%22

When viewing the issues lists, readers should keep in mind that the current ACE release is using Eclipse framework version 4.28, also known as version 2023-06.

Linux: Incompatible Default Web Browser



At startup, ACE tries to find a compatible 64-bit embeddable HTML browser already installed on the system for ACE to use to display HTML reports and help content. If no such embeddable browser is detected within the Linux installation, ACE shows the warning dialog above and reverts to a primitive fallback HTML browser (which has slightly reduced functionality and known stability issues on some platforms, but is still better than nothing)

RHEL/CentOS v7.9 and RHEL/Rocky v8.x , SUSE15, Ubuntu 20.04LTS, and Ubuntu22.04LTS customers are not expected to experience any problems with web browser support. Customers running on unsupported Linux distros might need to perform additional steps to ensure ACE has a compatible web browser framework available if the fallback (reduced-functionality) browser proves to be unstable.

Solution

To solve reported web browser incompatibility problems, work with your IT department to install an ACE-compatible 64-bit web browser in your distribution.

For all Achronix-supported Linux operating systems, any WebKit2GTK packages compiled for GTK+3 support are expected to work, regardless of version number, though the latest versions from the official distro repositories are expected to be the fastest and most stable.

Caution!

ACE Linux web browser support requires GTK+3 and WebKit2

Starting in ACE v8.4, ACE is no longer compatible with GTK+2. In Linux ACE requires GTK+3. GTK+4 is not yet supported.

Starting in ACE v8.4, ACE is no longer compatible with WebKit (unofficially also known as WebKit1). In Linux ACE requires WebKit2 (the successor to WebKit).

Details

GTK+2

Starting in ACEv8.4, ACE no longer supports GTK+2.

GTK+3

For GTK+3 Linux distros (like RHEL/CentOS7 and later, SUSE15 and later, and Ubuntu 20.04LTS and later), with the assistance of a system administrator, install a GTK+3 compatible version of WebKit2GTK (*not* WebKitGTK).

These are expected to be the following package names, all of which are available within the official distribution repositories and are often installed by default.

- (RHEL/CentOS7) a package named "webkitgtk4.x86_64", which contains WebKit2 compiled for GTK+3
- (RHEL/Rocky8) a package named "webkit2gtk3", which contains WebKit2 compiled for GTK+3
- (SUSE15) a package with a name starting with "libwebkit2gtk-4.0", which contains WebKit2 compiled for GTK+3
- (Ubuntu, all supported versions) a package with a name starting with "libwebkit2gtk-4.0", which contains WebKit2 compiled for GTK+3

GTK+4

ACE does not support GTK+4 at this time.

When Nothing Else Works

When nothing else works, or when all the available GTK+3 versions of WebKit2 fail to be found or start (due to crashes), one more choice exists. An ancient fallback HTML3 browser is shipped within ACE. This fallback browser appears to be stable on supported Linux distros (while using X11), but it has reduced functionality and performance, renders without antialiasing (the fonts often look ugly), the fonts can be illegible (too small) on HiDPI monitors, it occasionally might not show the entire report on the first try (though scrolling the report or resizing the report usually causes it to fully paint), and it does not directly support the Wayland display server (instead of X11).

ACE automatically tries this fallback browser when it cannot find any compatible embeddable browser within the OS. But if it finds a compatible browser (that subsequently crashes), ACE might not get a chance to automatically try the fallback browser.

It is possible to force the use of the fallback browser, which can get around crashes in the various WebKit and/or GTK frameworks. Start ACE with the appropriate argument to force the use of the fallback browser.

Starting ACE while forcing the use of the (reduced functionality) fallback HTML browser

```
./ace -force_fallback_html_browser
```

Note

This fallback browser is NOT an ideal solution. It is always a much better idea to work with the local IT/MIS support team to get the latest stable version of WebKit2 and GTK+3 installed and working together on the Linux workstation instead.

Caution!

Do not use both `-allow_wayland` and `-force_fallback_html_browser` at the same time

The fallback html browser is known to have multiple display and mouse interaction errors if you try using both `-allow_wayland` and `-force_fallback_html_browser` at the same time. (ACE defaults to disallowing Wayland rendering in favor of X11 rendering, so the `-force_fallback_html_browser` argument is expected to work in all cases, as long as `-allow_wayland` is not used at the same time.)

Additional Information

RHEL/CentOS7

RHEL/CentOS7 includes versions of WebKit2GTK+3 (the package "webkit4.x86_64") within the official release repositories. These are compatible with ACE when running on RHEL/CentOS7. These packages are often installed by default, and must be installed for full functionality to be available in ACE.

RHEL8

RHEL8 includes versions of WebKit2GTK+3 (the package "webkit2gtk3.x86_64") within the official release repositories. These are compatible with ACE when running on RHEL8. These packages are often installed by default, and must be installed for full functionality to be available in ACE.

WebKit and WebKitGTK Technical Details

There are two main APIs for WebKit development. These are known as WebKit2 (the latest edition), and its predecessor known as WebKit (also sometimes referred to for clarity as WebKit(1) or WebKit[1]). There are many released versions of each, including (confusingly) WebKit(1) version 2.x.y, which is incompatible with WebKit2.

Versions of WebKit and WebKit2 can be compiled to support GTK+2 or GTK+3, and recent versions of WebKit2 may also support GTK+4, which are then all grouped under the WebKitGTK name/prefix in some distros like RHEL7. (Other Linux distributions including RHEL8 have chosen a clearer delineation by using the names WebKitGTK and WebKit2GTK.)

Over time, WebKit for GTK has been made available as a multitude of different package names on Linux, corresponding to the various combinations of support for GTK+2/GTK+3/GTK+4 and WebKit(1)/WebKit2, and recently (on SUSE and Ubuntu) with differing versions of an underlying package called libsoup. The most common known package/library names are listed below for supported versions of Linux. Names which are ~~struck~~ are known to not work with ACE.

- ~~webkitgtk~~: WebKit(1) compiled for GTK+2 (found in non-official community repositories for RHEL7)
- ~~webkitgtk2~~: WebKit2 compiled for GTK+2 (only briefly available for RHEL7; this appears to have been a short-lived option during the very early development of WebKit2)
- ~~webkitgtk3~~: WebKit(1) compiled for GTK+3 (found in the official RHEL7 repositories)
- ~~webkitgtk4~~: WebKit2 compiled for GTK+3 (found in the official RHEL7 repositories; this is the WebKitGTK package most actively developed/supported for RHEL7/CentOS7)
- ~~webkit2gtk3~~: WebKit2 compiled for GTK+3 (found in the official RHEL8 and RHEL9 repositories; this is the WebKitGTK package most actively developed/supported for RHEL/Rocky8 and RHEL/Rocky9)
- ~~libwebkit2gtk-4_0~~ and ~~libwebkit2gtk-4.0~~: WebKit2 compiled for GTK+3 using libsoup2 (found in the official SUSE 15 and Ubuntu repositories). This is the preferred version to be used with ACE on these distros.
- ~~libwebkit2gtk-4_1~~ and ~~libwebkit2gtk-4.1~~: WebKit2 compiled for GTK+3 using libsoup3 (found in the official SUSE 15 and Ubuntu 22.04LTS repositories). ACE has not yet been tested with this library.
- ~~libwebkit2gtk-5.0~~: WebKit2 compiled for GTK+4 (found in the official SUSE 15 repositories)

Other Linux Distros

Note

For unsupported Linux distros:

Alternate distributions of Linux may use different package naming schemes than those shown above.

The naming standards for WebKitGTK packages targeting GTK+2, GTK+3, GTK+4, WebKit, and WebKit2 are non-obvious, and are thus difficult to understand. It is unfortunately very easy to confuse versions of WebKitGTK(1) v2.x with versions of WebKit2GTK, even though they're incompatible.

A package management tool (like 'yum' on RHEL, 'zypper' on SUSE, or 'apt' on Ubuntu) is the best way to research these versions/dependencies, as well as perform the eventual package installation.

A website that can assist with the navigation of the confusing WebKitGTK names and versions is <https://pkgs.org/download/webkitgtk>. At that site, search for "webkit2gtk" in your chosen Linux distribution, find one that works with GTK+3, and then install it with the local package management tool appropriate for your Linux distro.

Linux: ACE Requires an Unusually Large Amount of Virtual Memory (Due to WebKit2)

In Linux, the Eclipse framework underlying ACE uses the WebKit2GTK+3 HTML browser framework. This framework uses large amounts of virtual memory (sometimes more than 100GB), apparently as part of the browser's javascript security model. At the present time, there appears to be no way for Achronix to ask/force WebKit2 to use less virtual memory in ACE.

Linux: ACE forces the use of the Adwaita GTK+3 Theme; Can I Change This?

The Eclipse GUI framework underlying ACE only guarantees correct behavior and appearance on GTK+3 when the Adwaita GTK+3 theme is in use. Thus ACE is only officially supported running on the Adwaita GTK+3 theme. Because the Adwaita GTK+3 theme is available by default on all supported Linux distributions, this restriction is not expected to be a problem in most cases.

Themes

ACE currently always uses the GTK+3 widgets library in Linux, regardless which desktop environment and/or window manager is being used. The GTK+3 widgets support theme customization. Thus the look and feel of ACE can be modified by changing what is called the "application GTK widgets theme/style/appearance" (the exact name varies based on distro/version/desktop) setting in Linux. By default, to ensure accurate rendering and behavior, as well as optimal rendering performance, ACE forces itself to use the Adwaita theme, even if another theme has been explicitly chosen for other GTK+3 applications running on that Linux desktop.

⚠ Caution!

The default GTK+3 theme "Adwaita" is the only theme supported by ACE.

The Eclipse GUI framework underlying ACE only guarantees correct behavior and appearance in Linux when the Adwaita GTK+3 theme is in use. Thus ACE is also only officially supported in Linux when running on the Adwaita GTK+3 theme. At startup, ACE overrides the user/system GTK+3 theme choice and forces itself to use Adwaita, even if the user has chosen a different theme to be used by all GTK+3 applications. Because Adwaita is the default GTK+3 theme provided by the GTK team themselves, and because Adwaita ships as a part of GTK+3 by default, this restriction is not expected to be a problem for ACE users on Achronix-supported Linux distros.

At this time, the default GTK+3/Gnome theme "Adwaita" is the only supported GTK+3 Theme choice for ACE.

Customers running Linux distributions other than RHEL and/or SUSE (and their respective clones), which all use Adwaita by default, might also have good results with whatever GTK+3 Theme is enabled by default with their distro, but this is only recommended if the preferred Adwaita theme is not available.

When trying alternate themes, Achronix requires that GTK+ v3.22 or later be used with ACE, because those GTK+ versions are reportedly the "final, stable" versions of GTK+3, and are thus the most stable. All releases of GTK+3 prior to v3.22 were considered by the GTK+ team to be (unstable) development releases.

By starting ACE with the command-line argument `-keep_user_gtk_theme`, ACE stops enforcing the usage of the Adwaita theme, and allows the use of alternate system GTK+3 themes. Be warned that the use of alternate GTK+3 themes can cause illegible foreground/background color combinations, graphical performance issues, drag-and-drop issues, loss of accessibility functionality (like screen readers), and in rare cases can even cause ACE to crash. In all cases that were backtracked to the Themes frameworks, when the user reverted back to allowing ACE to use the (default) Adwaita theme, these problems did not reoccur in ACE.

Changing the GTK+3 Theme

i "Dark" themes are not supported in ACE.

At this time, ACE does not support any dark themes. Customers desiring ACE support for dark themes should request "dark theme support" as an ACE feature enhancement so it can be prioritized appropriately.

The exact setting to change the GTK application widgets theme varies by Linux distribution and version, as well as by desktop manager and version.

Under CentOS7 Gnome, for example:

Gnome Tweak Tool → **Appearance** → **Theme** → **GTK+**

Under CentOS7 KDE, for example:

System Settings → **Application Appearance** → **GTK+ Appearance** → **GTK+ Styles** → **Widget Style**

Under CentOS7 MATE, for example:

System → **Preferences** → **Look and Feel** → **Appearance** → **Theme** → **Customize...** → **Controls**

Consult with your local System Administrator for additional details regarding the configuration of GTK+3 and themes for your local Linux installation.

Animations and Other Effects

While desktop, application/window, and widget animations can improve the feel of applications for some users, other users want to avoid the negative performance impacts.

Animations and special effects are not managed by ACE itself. Often these are controlled within the Desktop Environment, Compositor, and/or Window Manager (exact locations of these settings vary significantly, with some settings available only through the manual editing of Linux configuration files), and some animations may vary even with the user's choice of GTK Theme.

Under CentOS7 Gnome, for example, animations settings can be found at:

Tweak Tool → **Appearance** → **Enable animations**

Under CentOS7 KDE, for example, multiple animation settings can be found at:

System Settings → **Desktop Effects**

Consult with your local System Administrator for additional details regarding the configuration of desktop, application, and widget animations and special effects for your local Linux installation.

Linux: Views and Editors Detach when Dragged Instead of Docking in the Workbench

There is currently a known GTK theme bug (**Linux-only**) in the Eclipse application frameworks underlying ACE that causes view/editor tab docking (including tab re-arrangement) to fail when the Help Window is used. This bug can occur even when the Help Window is minimized; some part of the underlying frameworks is remembering the window size/location despite minimization.

This bug currently appears to be dependent upon which GTK Theme is being used by the Linux window manager. (This theme choice is configured outside ACE.) We have not yet heard any reports of the bug being observed when the system default GTK themes (Adwaita on RHEL/CentOS7 and RHEL8) were in use.

There are three workarounds to allow docking when this bug occurs:

- Close the Help Window while performing the view/editor tab movement operations, and then re-open the Help Window when the movements are completed. (Minimizing the Help Window is not enough. The Help Window must be closed.)
- Shrink and move both the ACE window and the Help Window to a size/location where they do not intersect, then change the view/editor tab locations within the Workbench, then restore the desired Workbench and Help window sizes/locations.
- Work with the local Linux system administrator to change the GTK theme being used, or try to update to a newer/patched version of the chosen GTK theme.
Some versions of the Clearlooks and Glider themes seem most likely to exhibit the problem.

See the previous section titled [Themes \(page 783\)](#) for more information on choosing an alternative theme in Linux.

Linux: CDE: Dialogs and Wizards Sometimes Appear Behind the Main ACE Window, Especially After Minimize/Maximize

This problem has only been observed at sites running an X server on RHEL/CentOS and the X client on CDE running within SunOS/Solaris. **This configuration is not officially supported**. (Achronix does not support running ACE where either the X server or the X client are on anything except supported operating systems. See the release notes accompanying ACE to determine the exact supported OS versions for a given ACE release.)

CDE has known inter-window focus issues when displaying GTK applications using the default CDE configuration. This problem is not unique to ACE, nor is it something over which ACE has any control whatsoever.

As an example, IBM tools also experience similar problems. A good description of a fix for the issue is in the IBM support forums at: <http://www-1.ibm.com/support/docview.wss?uid=swg21124274>.

Note

Paraphrased workaround from the IBM support forums:

Basically, the problem is caused by an awkward default setting of CDE that allows modal dialogs to be hidden behind other (parent) windows.

To replace this default setting with a more sane one, the following line needs to be added to `$HOME/.Xdefaults`:

```
Dtwm*secondariesOnTop: True
```

After that, reload the `Xdefaults` and restart the window manager.

Finally, it might be necessary to also update **CDE Style Manager** → **Windows** where "**Allow Primary windows on top**" should not be enabled (uncheck the checkbox).

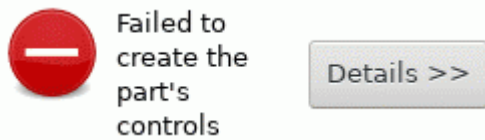
If this specific workaround from IBM tech support does not solve the problem in the local CDE configuration, please perform a web search (using similar terminology) with the assistance of a local system administrator to find and apply the fix/workaround for the local Linux window manager configuration.

⚠ Caution!

Achronix does not support running ACE on any combinations of Solaris/SunOS/CDE. Consult with a local System Administrator before making these or similar changes on SunOS or Solaris or CDE.

Linux: "Failed to create the part's controls": Some Views and IP Editors may fail to initialize

When this problem occurs, an error message will appear similar to the following screenshot:



The problem has only been observed on older versions of RHEL/CentOS7 (versions prior to RHEL/CentOS7.7), and only with specific Desktop or Window Managers, and only at very high resolutions (when GTK+3 font scaling may become enabled). The error currently appears to be related to bugs in the various graphical library dependencies provided by Linux itself (these are not shipped within ACE).

All customers experiencing the problem have reported that upgrading the version of their preferred Desktop or Window Manager, or changing to another Desktop or Window Manager, or upgrading to newer releases of RHEL/CentOS7 (versions 7.7 or later) has fixed their problem and allowed all ACE Views and IP Editors to once again work as expected.

Upgrading an ACE Installation

Note

This is also supposed to be covered in the *ACE Installation Guide (UG002)*.

On Windows

Each version of ACE should ideally be installed into a new, empty directory! Never install ACE in the same directory as a prior install, unless that prior version has already been uninstalled first.

1. Disconnect any USB Bitporters.
2. (If a node-locked license is being used for ACE:) Copy the `license/*.lic` file from the ACE installation directory to another location (somewhere not under the ACE installation directory).
3. Optionally, uninstall the prior version of ACE.
4. Install the desired version of ACE to a new directory.
5. (If a node-locked license is being used for ACE:) Copy the `license/*.lic` file back to the proper location within the new ACE installation directory.
6. Re-connect any USB Bitporters.
7. Run ACE.

On Linux

Each version of ACE must be installed into a new, empty directory! Never install ACE in the same directory as a prior install.

1. Create a new directory to contain the new version of ACE.
2. Untar ACE into the new directory.
3. Run ACE.

GUI Problems after Upgrading?

In rare cases after an upgrade, (almost always when a different version of ACE is mistakenly installed on top of an existing prior installation,) ACE GUI errors or crashes might occur, especially in the IP Editors.

If you do not wish to perform an uninstall/reinstall of ACE, the following steps often solve the problem:

1. Delete the ".eclipse" subdirectory (the leading '.' is important!) in the your home directory.
 - (Windows:) typically "C:\Users\username\.eclipse\"
 - (Linux:) typically "/home/username/.eclipse/"

Caution!

This subdirectory is hidden in Linux; if unsure what this means or how to find it, please ask your system administrator for help.

2. Try running ACE again

If problems persist:

1. Contact Achronix Technical Support, providing the following log files along with a description of the problem encountered:
 - (Windows):
 - i. "C:\Users\username\.achronix\ace_timestamp.log"
where *timestamp* is *year_month_day_hour_minute_second*. (Typically the crash occurred in the most recent log file.) For example:
"C:\Users\patsmith\.achronix\ace_2018_12_02_16_06_58.log"
 - ii. "C:\Users\username\.achronix\workspace_version\framework\.metadata\.log"
where *version* is the version of ACE which is being run, and *framework* is the Eclipse framework version underlying ACE. For example:
"C:\Users\patsmith\.eclipse\workspace_7.1\46\.metadata\.log"
 - (Linux)
 - i. "\home\username\.achronix\ace_timestamp.log"
where *timestamp* is *year_month_day_hour_minute_second*. (Typically the crash occurred in the most recent log file.) For example:
"\home\patsmith\.achronix\ace_2018_12_02_16_06_58.log"
 - ii. "\home\username\.achronix\workspace_version\framework\.metadata\.log"
where *version* is the version of ACE which is being run, and *framework* is the Eclipse framework version underlying ACE. For example:
"\home\patsmith\.eclipse\workspace_7.1\46\.metadata\.log"
2. Delete the ".achronix" subdirectory (the leading '.' is important!) in your home directory.
3. Run ACE

Chapter 7 : Revision History for ACE

Revision History

Version	Date	Description
1.0	02 Jul 2016	Initial Speedcore document release.
1.1	30 Oct 2016	<p>Additions:</p> <ul style="list-style-type: none"> Added new tasks detailing Automatic Flop Pushing into I/O Pins (page 457) and Working with Virtual I/O (page 466). Added the page Filter Properties (page 264) showing all supported filters for the Search Filter Builder Dialog (page 186), and the find (page 638), and filter (page 637) Tcl commands. Added new Speedster16t IP Configuration Editor sections Added JTAG support for FTDI FT2232H (in addition to Bitporter) Added Strict Flow Mode in addition to Evaluation and Normal modes to the Options View (page 96) Added new Task content with additional info about Highlighting Objects in the Floorplanner View (page 343), and updated cross-references for Views providing Highlight functionality. Added the page Detecting Changes to Project Source Files (page 324) Integrated the formerly standalone Incremental Compile Tutorial into this guide, under Using Incremental Compilation (Partitions) (page 404) <p>Updates:</p> <ul style="list-style-type: none"> Updated the Options View (page 96) to reflect the latest implementation options. The Clock Domains View (page 20), Clock Regions View (page 23), Partitions View (page 107), and Placement Regions View (page 110) have all been updated to mention support for new actions, and dynamic per-device site type columns Updated the Floorplanner View (page 43) to reflect the device awareness for Labels and Tooltips in the fly-out palette. Split the Tcl Command Reference (page 580) into two sections: SDC Commands (page 580) and ACE Tcl Commands (page 610). Renamed all occurrences of "SnapShot" to "Snapshot"


Version	Date	Description
1.1	30 Oct 2016	<p>Removals:</p> <ul style="list-style-type: none"> In the Critical Path Diagram View (page 33), the Layers choices specific to obsolete fabrics have been removed. Associated screenshots and text were be updated. Obsolete commands were removed from the ACE Tcl Commands. (page 610)
1.2	30 Nov 2016	<p>Additions:</p> <ul style="list-style-type: none"> Concepts (page 6): Added ECC support to the Speedster16t BRAM Configuration Editor and Speedster16t FIFO Configuration Editor. Tcl Command Reference (page 580): Added a new Tcl commands category, Interactive Timing Commands. (page 603) <p>Updates:</p> <ul style="list-style-type: none"> Concepts (page 6): <ul style="list-style-type: none"> In the Floorplanner View (page 43), the choice "IO Port Names" has been re-added to the Labels and Tooltips options for eFPGA/Speedcore products. Misc flop pushing clarifications in the Options View (page 96) and Automatic Flop Pushing into I/O Pins (page 457) pages. Enhanced the description of strict mode error checking on the Flow Mode (page 242) page. Tcl Command Reference (page 580): Clarified when the "p:" port name may be used with the set_placement (page 713) Tcl command.
1.2.1	23 Dec 2016	<p>Updates:</p> <ul style="list-style-type: none"> Tasks (page 282): <ul style="list-style-type: none"> Minor updates for clarity on Automatic Flop Pushing into I/O Pins (page 457). Removed references to obsolete <i>Bitporter and acx_stapl_player Software Release Notes (RN007)</i>. Tcl Command Reference (page 580): <ul style="list-style-type: none"> Added details to get_pins (page 587). Corrected error in set_clock_latency (page 591). Add details for the option <code>-cpu_width <int></code> to write_bitstream (page 719). Concepts (page 6): Removed references to obsolete <i>Bitporter and acx_stapl_player Software Release Notes (RN007)</i>. Views (page 16): Added details on the CPU Bus Width option under the Options View. (page 96)

Version	Date	Description
2.0_beta	01 Feb 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Troubleshooting (page 761): added section about ACE startup errors regarding localhost TCP ports.
2.0_beta2	28 Feb 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): <ul style="list-style-type: none"> ◦ Added seventeen new Bitstream generation Implementation Options to the Options View (page 96) for Speedcore eFPGAs (two options specific to Speedster FPGAs were hidden). ◦ Added new preferences to the Project Management Preference Page (page 214) to disable and/or change the frequency of project source file consistency checks. <p>Updates:</p> <ul style="list-style-type: none"> • Tasks (page 282): Updated Automatic Flop Pushing into I/O Pins (page 457): the port attribute <code>syn_useiff</code> should no longer be used. • Tcl Command Reference (page 580): <ul style="list-style-type: none"> ◦ add_project_constraints (page 610) now supports filtering constraints by corner, temperature, and voltage. ◦ run_stapl_action (page 789) now supports STAPL variable initialization overrides with the new <code>-defines</code> option. ◦ write_bitstream (page 719) now supports configurable bit widths for CPU Mode formatted output files. ◦ Updated help text for get_pins (page 587), set_clock_latency (page 591), set_equivalent_pins. (page 710)
2.0	02 May 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Added details regarding Target Device property to the following IP configuration pages: Speedster16t Shift Register Overview Page, Speedster16t ROM Overview Page, Speedster16t DSP FIR Filter Overview Page, Speedster16t BRAM Overview Page, Speedster16t LRAM Overview Page, Speedster16t LRAM FIFO Overview Page, and Speedster16t FIFO Overview Page • Tcl Command Reference (page 580): <ul style="list-style-type: none"> ◦ added set_project_constraints_pvt (page 713)

Version	Date	Description
2.5	01 Jul 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Views (page 16): Added new Implementation Options "Move Flip-flop Reset", "Pad Flop Pushing Clock Type", and "Report all temperature corners" to the Options View (page 96). • Tcl Command Reference (page 580): Added create_boundary_pins (page 620), export_partition (page 636), get_ace_ext_dir (page 642), get_ace_ext_lib (page 642), get_flow_steps (page 647), get_report_sweep_temperature_corners (page 659). <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): New IP Configuration Dialog (page 176) now prohibits IP module name collisions with Achronix's reserved module names. • Concepts (page 6): The Flow Steps (page 234) page now includes a table of flow step names and IDs. • Concepts (page 6): The Speedster16t BRAM Configuration Editor now supports "Simple Dual Port with Soft ECC".
2.9	24 Sep 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Tcl Command Reference (page 580): Added get_partition_names (page 652), move_project_netlists (page 669), write_partition_blackbox (page 721), write_partition_db (page 721). <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): <ul style="list-style-type: none"> ◦ The Multiprocess Summary Report (page 255) now supports the Timing Analysis "Report all temperature corners" implementation option, and shows additional columns of timing data for each reported PVT combination. Peak memory usage is now reported alongside the runtimes. Clarity of results is improved in cases where the report contains a mix of Sign-Off timing data for some implementations and Post-Route timing data for other implementations. ◦ The Snapshot Debugger View (page 137) has been significantly updated to reflect the latest Snapshot Version 3 enhancements. ◦ Configure JTAG Connection Preference Page (page 190) updated with details regarding multi-device scan chains. • Tasks (page 282): The entire section regarding Running the Snapshot Debugger (page 364) has been updated to reflect the latest Snapshot Version 3 enhancements. These include new features like: Startup Trigger; Edge Triggering; configurable monitor, trigger, and stimuli widths; Repetitive Trigger mode; etc. For complete details, see the <i>Snapshot User Guide</i> appropriate to each Achronix device family.

Version	Date	Description
2.10	24 Dec 2017	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): Added supplemental information describing Timing Across All Temperature Corners (page 266) • Tcl Command Reference (page 580): Added new commands: export_all_partitions (page 636), generate_route_delay_table (page 641), insert_delay (page 665) <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): <ul style="list-style-type: none"> ◦ On the Configure JTAG Connection Preference Page (page 190), corrected misleading information regarding multi-device scan chains, and removed now-obsolete configuration settings (for pod type and connection type). ◦ Removed now-obsolete information from the Options View (page 96) regarding JTAG configuration settings. ◦ In the Flow View (page 53), added the Warning icon to the icons table. The icon is primarily used to indicate out-of-sync files, as described in Detecting Changes to Project Source Files (page 324). • Tasks (page 282): <ul style="list-style-type: none"> ◦ The Configuring the JTAG Connection (page 360) page was updated to further clarify details regarding multi-device scan chains, and to remove mention of now-obsolete configuration settings (for pod type and connection type). ◦ The Generating Timing Reports (page 353) page was updated with additional information about multiple temperature corners and related Tcl commands. ◦ The Running Multiple Flows in Parallel (page 308) page was updated to clarify why external job submission systems must use synchronous/blocking commands, and why asynchronous/non-blocking commands can potentially cause serious problems. ◦ The Single-Process Incremental Compile Tutorial (page 408) was updated for clarity ◦ The Snapshot Design Flow (page 364) received an updated diagram • Tcl Command Reference (page 580): Updated create_boundary_pins (page 620), create_flow_step (page 621), export_partition (page 636), run_fpga_download (page 690), save_properties (page 707), write_partition_blackbox (page 721), write_partition_db (page 721) • Troubleshooting (page 761): enhanced information regarding ACE error codes

Version	Date	Description
3.0	19 Aug 2018	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): <ul style="list-style-type: none"> ◦ Added new page describing various supported ACE Verilog Attributes (page 259) that can be applied to instances, nets, pin, ports, or other objects in the ACE datamodel. ◦ Described the new Properties View (page 123) ◦ Added a description of the Implementation Options Report (page 257) • Tasks (page 282) <ul style="list-style-type: none"> ◦ Added content regarding Applying and Checking Properties (page 357) • Tcl Command Reference (page 580): <ul style="list-style-type: none"> ◦ Added new Tcl commands get_clock_regions (page 644), get_clock_region_bounds (page 644), get_file_line (page 647), get_regions (page 659), get_region_bounds (page 659), report_coverage (page 678) ◦ Added new Interactive Timing Commands (page 603) pages for check_timing and report_clock. Reminder: these are stand-alone timer commands, enabled only after prepare_sta (page 605) is run, and disabled after reset_sta (page 609) is run. <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): <ul style="list-style-type: none"> ◦ The JTAG Browser View was updated to include a new "Word Step" field. ◦ Corrected the Critical Path Diagram View (page 33) to reflect the removal of the now-obsolete Layers section of the palette ◦ The configuration of the Floorplanner View (page 43)'s Route Rendering Mode has been moved from the view's fly-out palette to the Floorplanner View Colors and Layers Preference Page (page 194) ◦ Clarified how the  icon in the Flow View (page 53) relates to Detecting Changes to Project Source Files (page 324) ◦ Documented the new ease-of-use feature/button () on the Multiprocess View (page 73) allowing users to re-open a pre-existing Multiprocess Summary Report (page 255). ◦ For the Options View (page 96), updated the listings of common options, and improved the content describing the Tcl interactions for viewing/changing implementation options in general ◦ Improved the description of the Power Dissipation Report (page 245), including clarification of the ramifications of Timing Across All Temperature Corners (page 266)

Version	Date	Description
3.0	19 Aug 2018	<p>Updates (cont.):</p> <ul style="list-style-type: none"> • The Projects View (page 117) gained a new action: Reload Project (); added a screenshot and description showing how disabled constraints will be greyed out in this view; clarified details regarding the load order of constraint files. • Tasks (page 282): <ul style="list-style-type: none"> ◦ Clarified details regarding Adding Source Files (page 297) to ACE projects, how the source file load order may be changed, and the ways in which constraint files may be enabled/disabled (instead of added/removed) for implementations ◦ Mentioned that it is also possible to disable constraint files in implementations, instead of completely Removing Source Files (page 301) from projects ◦ Enhanced the instructions for Assigning Placement Region Constraints (page 398) ◦ Renamed the section 'Getting Floorplanner Object Tooltips' to Choosing Floorplanner Object Tooltips (page 342) for clarity ◦ Enhanced the descriptions of Loading Projects (page 295) to cover both GUI and Tcl interactions, with new explanations of project locking and lock files. ◦ Added more thorough discussion of the license ramifications of Running Multiple Flows in Parallel (page 308), and added a cautionary note describing how multiple implementations can unexpectedly generate identical results after upgrading ACE, along with the recommended fix. • Tcl Command Reference (page 580): <ul style="list-style-type: none"> ◦ Updated <code>report_timing</code> with newly supported command line options. ◦ Renamed '<code>display_rtl</code>' to <code>display_netlist</code> (page 628) which will better reflect the command's actual functionality • Troubleshooting (page 761): updated discussion of project locking and when forcing locked projects to load is acceptable; updated descriptions of font management; updated content for ACE startup error diagnosis (firewall vs license issues); added new content regarding missing icons for actions in menus.

Version	Date	Description
7.0	07 Dec 2018	<p>ACE v7.0 is the first combined release, supporting both Speedster FPGA devices and Speedcore eFPGA devices. Rather than parallel releases, there is now a single release, thus the change in numbering schemes (to be more in sync with the higher-versioned Speedster software, which was in the 6.x release sequence).</p> <p>Additions:</p> <ul style="list-style-type: none"> • Tasks (page 282): added several pages describing the ACE help system: Accessing Help (page 473); added a new page about Detaching Views and Editors (page 288); added a new page describing Multiprocess Batch Mode (page 318) (using the new run_multiprocess (page 692) Tcl command), including the new seed sweep functionality • ACE Tcl Commands (page 610): added new command run_multiprocess (page 692) • Troubleshooting (page 761): added a section regarding changing Linux GTK theme and animation settings to affect the render performance as well as the look and feel of the ACE GUI; added a section describing the known (GTK theme-dependent) bug where views/editors may detach (instead of move) while the Help Window is open; added a section about the impacts of Linux resource limits; added a section about the twm Window Manager in Linux; added a section regarding Linux LD_LIBRARY_PATH concerns; added a section about dialogs in the CDE Window Manager <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): The Floorplanner View (page 43) has changed the presentation of routing errors (open connections, open pins, and route overflows); the Critical Path Diagram View (page 33) now has right-click context menu items similar to the other views within the Floorplanner Perspective; the Options View (page 96) section was updated to match the latest lists of options, but now includes only those options which are common to all target devices – implementation options which are unique to specific libraries or devices will now be documented elsewhere, including within the on-demand Implementation Options Report (page 257). • Tasks (page 282): The Moving and Docking Views and Editors (page 287) page and Rearranging Tabbed Views and Editors (page 288) page have been re-titled and have had their content updated to reflect that Editors can now be moved around just like Views. The user feedback has changed during movement and docking, and the descriptions/tables have been updated accordingly.

Version	Date	Description
7.0	07 Dec 2018	<p>Updates (cont.):</p> <ul style="list-style-type: none"> • ACE Tcl Commands (page 610): added "-verbose" option to: add_region_find_insts (page 613), add_region_insts (page 614), create_region (page 624), and remove_region_insts (page 676); find (page 638) added "-warning" and "-error" options; run_generate_fullchip_sim (page 691) added "-modelsdir" option; set_false_path (page 596) has improved description of various options; add_region_find_insts (page 613), add_region_insts (page 614), create_region (page 624), and remove_region_insts (page 676) added a new "-clocks_only" option; remove_region_insts (page 676) added a new "-flops_only" option; run_insert_holdbuffers (page 692) added a new option "-io_buffers". • Troubleshooting: (page 761) the Linux web browser section has been updated to reflect the change from the Mozilla XulRunner to the WebKitGTK+ HTML browser framework <p>Removals:</p> <ul style="list-style-type: none"> • Deleted Concepts and Tasks pages made obsolete by the updated GUI frameworks: "Fast Views", "Opening Perspectives"
7.1	27 Mar 2019	<p>Additions:</p> <ul style="list-style-type: none"> • Advanced Concepts (page 259): added a new page describing ECO Commands (page 268) <p>Updates:</p> <ul style="list-style-type: none"> • Reports (page 244): updated information on the Implementation Options Report (page 257) page to improve clarity and accuracy • Views (page 16): added info to the Properties View (page 123) page covering double-click shortcut gestures and context-menu actions; updated the table of Actions to match the latest functionality on the Projects View (page 117) page • Tasks (page 282): updated content on the Multiprocess Batch Mode (page 318) page for improved clarity • Troubleshooting (page 761): removed obsolete content and updated content for new potential problems and workarounds, now that ACE has added official support for GTK3 and WebKit2 (both new as of this release, see new content for technical details)

Version	Date	Description
7.2	06 Jun 2019	<p>Updates:</p> <ul style="list-style-type: none"> • Tcl Command Reference (page 580): Updated the descriptions for all SDC Commands (page 580), and provided/updated usage examples where applicable. To avoid command naming collisions with other tools, updated the names and descriptions for the Interactive Timing Commands (page 603). • Views (page 16): updated the Netlist Browser view (page 79) actions table, adding missing actions and updating icons that have changed; updated the Floorplanner View (page 43) to mention the new Allow Tooltips toggle checkbox; updated the Critical Paths View (page 37) to mention the new Show Clock Paths action; updated the Multiprocess view (page 73) page, updating the screen shot and the text to include the new Seed Sweep functionality; updated the Flow View (page 53) page, updating the screen shot and icon images to match recent updates. • Advanced Concepts (page 259): updated the ECO Commands (page 268) section, moving each related dialog into its own page and updating screen shots • Troubleshooting (page 761): updated the section about Themes to explain that ACE will now enforce the usage of the Adwaita theme when running on GTK3.14+ (to maximize performance and stability).
8.0	17 Sep 2019	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): added pages for the new I/O Designer Toolkit Views (page 60) and Generate I/O Ring Design Files Dialog (page 171). • ACE Tcl Commands (page 610): added new command: <code>generate_soc_design_files</code> (renamed <code>generate_ip_design_files</code> (page 641) in the 8.1 release). <p>Updates:</p> <ul style="list-style-type: none"> • Concepts: The Projects View (page 117) page removed the obsolete Remove All Projects action and added the new actions to Clone IP and Rename IP. • Tasks (page 282): The Single-Process Incremental Compile Tutorial (page 408) content has had its formatting tweaked. • ACE Tcl Commands (page 610): The interactive timing command <code>report_checks</code> (page 606) removed the unsupported argument <code>"-input_pins"</code>. The interactive timing command <code>reset_sta</code> (page 609) had a typo in the examples.

Version	Date	Description
8.1	24 Jan 2020	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): What was previously a single I/O Designer View (with multiple inner tabs) has now been split into multiple independent-but-related views, grouped under the I/O Designer Toolkit Views (page 60) page. The new pages are for the I/O Core Pin Assignment View (page 65), I/O Layout Diagram View (page 67), I/O Package Diagram View (page 63), I/O Pin Assignment View (page 64), and I/O Utilization View (page 62). (As a reminder, these I/O Designer Toolkit views are only relevant for Achronix Speedster7t FPGAs such as the AC7t1500.) • Troubleshooting (page 761): Added a section dealing with the most common symptom of improperly installed device overlays: Unable to initialize reserved module names list. <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): updated I/O Designer Toolkit Views (page 60) page with info about cloning and double-clicking. • ACE Tcl Commands (page 610): The command <code>generate_soc_design_files</code> was renamed <code>generate_ip_design_files</code> (page 641). The command <code>run_unroute</code> (page 704) has gained arguments to deal with regions.
8.1.1	21 Feb 2020	<p>Updates:</p> <ul style="list-style-type: none"> • ACE Tcl Commands (page 610): The command <code>run_insert_holdbuffers</code> (page 692) was updated to support a new optional argument: <code>"-typebased_buffers"</code>.

Version	Date	Description
8.2	17 Jul 2020	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): Added info about 'Add [copies of] IP to another project...' to Projects View (page 117). Added info about "active editor highlighting" to I/O Layout Diagram View (page 67). Added info about 'Remap Port/Signal Name' to I/O Pin Assignment View (page 64), I/O Core Pin Assignment View (page 65). • ACE Tcl Commands (page 610): added new commands: get_fanout (page 789), run_multiprocess_iterator (page 695) <p>Updates:</p> <ul style="list-style-type: none"> • ACE Tcl Commands (page 610): the run_multiprocess (page 692) command has a new flag <code>-create_option_sets</code>; the save_placement (page 706) command can now save placement of just a subset of the design, using the new option <code>-instances</code>; the run_insert_holdbuffers (page 692) command has a new option <code>-typebased_buffers</code>. • Concepts (page 6): The Multiprocess Summary Report (page 255) is now automatically sorted by quality of results, instead of alphabetically. The Multiprocess View (page 73) (and run_multiprocess (page 692)) can now optionally generate customized option sets for any chosen template implementation. The Netlist Browser View (page 79) now has convenience filters to hide instances of the cell types of boundary pins, power, and ground. The Save Placement Dialog (page 182) can now optionally accept Tcl lists (or Tcl statement that generate lists) of Instance names whose placements shall be saved, and can be pre-populated from the Search View (page 129) and the Selection View (page 133). • Tasks (page 282): The Single-Process Incremental Compile Tutorial (page 408) has been updated to improve phrasing/clarity.

Version	Date	Description
8.3	16 Dec 2020	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): created a page for the new Create a SecureShare Zip File Dialog (page 160) • Tasks (page 282): created a page for Using the ACE SecureShare Tool to Create a Support Zip File (page 476) • ACE Tcl Commands (page 610): added get_best_multiprocess_impl (page 644), report_performance (page 681), <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): The old "global" option sets have been removed in favor of the improved custom-generated option sets based on design analysis; the Multiprocess View (page 73) and Active Project and Implementation (page 229) pages have been updated accordingly. Added more details about Log Files (page 231) generated during Multiprocess. • Tasks (page 282): Misc clarifications for Multiprocess Batch Mode (page 318) , option set updates for the Attempting Likely Optimizations Using Option Sets (page 392). • ACE Tcl Commands (page 610): remove_impl (page 673) now supports working on a list of impls instead of just a single impl; clarified help text for run_multiprocess (page 692) and run_multiprocess_iterator (page 695);
8.3.2	11 Mar 2021	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts (page 6): added a page to describe the new NoC Performance View (page 87) • ACE Tcl Commands (page 610): added new commands: get_compatible_placements (page 645), move_partition, process_move_partition, redirect (page 671) <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): Under ACE Verilog Attributes (page 259), renamed "ace_useioff" to "syn_useioff"; on the Perspectives (page 6) page described the new NoC Performance Perspective; • Tasks (page 282): The Automatic Flop Pushing into I/O Pins (page 457) page was updated to use the latest Verilog attributes. • ACE Tcl Commands (page 610): renamed "report_regions" to report_clock_regions (page 677), the add_project_ip (page 611) and remove_project_ip (page 675) commands now require Tcl lists of acxip filenames to improve behavior with paths containing spaces; apply_placement (page 615) and set_placement (page 713) now accept partition names (to be used in combination with move_partition); the report_checks (page 606) command now supports a new option "-unconstrained"; report_impl_options (page 679) now supports new arguments "-hide_values", "-show_standard", and "-diff_options";

Version	Date	Description
8.5	03 Aug 2021	<p>Additions:</p> <ul style="list-style-type: none"> • ACE Tcl Commands (page 610): Added new commands create_equivalent_regions (page 620), remove_project_constraints_pvt (page 674), and untar (page 718). • Concepts (page 6): Added a new Advanced Concept description for Fabric Clusters (page 281). <p>Updates:</p> <ul style="list-style-type: none"> • SmartSupport™ has been renamed SecureShare™, affecting several pages. • The Create Placement Region Dialog (page 167) concept and Creating a New Placement Region (page 395) task have been updated to reflect new functionality, allowing (not just instance placement, but also) routing to be restricted to the placement region. • Concepts (page 6): The NoC Performance View (page 87) now supports drag-scrolling. • ACE Tcl Commands (page 610): <ul style="list-style-type: none"> ◦ The get_property (page 658) command has been updated with a new "-object_type" argument. ◦ The message (page 668) command now allows console logging support to be toggled on and off. ◦ The report_checks (page 606) command now warns that the report generated by this command may include less information than the regular Timing Report. ◦ The restore_project (page 688) description has been updated for clarity. ◦ The set_clock_type (page 709) command now supports a new "-batch" argument. ◦ The set_partition_info (page 712) command arguments and descriptions have been updated for clarity. ◦ The write_bitstream (page 719) command has new arguments "-max_size" and "-two_stage". • Troubleshooting (page 761): Commentary regarding now-obsolete/unsupported Windows7 and CentOS6 (along with the associated GTK2 details) have been removed.

Version	Date	Description
8.6	20 Oct 2021	<p>Additions:</p> <ul style="list-style-type: none"> • ACE Tcl Commands (page 610): added new commands optimize_tile (page 670) , regenerate_all_ip_design_files (page 671) <p>Updates:</p> <ul style="list-style-type: none"> • Lots of pages throughout the document have been updated with very minor edits to improve grammar and clarity of phrasing, as well as fixing typos. • Concepts (page 6): The Views (page 16) page has been updated to note that the view context menu icon used by all Views has changed from a tiny down-arrow to a vertical elipsis – (most View screenshots have not yet been updated to reflect this icon change). • ACE Tcl Commands (page 610): The set_partition_info (page 712) command gained a new argument -exclusive_placement. The set_placement (page 713) command gained a new argument -auto_place_neighbors. The run_secureshare (page 698) command removed the argument -no_archive and gained the argument -wizard. <p>Deletions:</p> <ul style="list-style-type: none"> • Removed remaining content regarding the (no longer supported) 22i and 16t IP Configuration Editors. • Removed remaining content regarding the (no longer supported) JTAG Browser View and JTAG Diagram View.

Version	Date	Description
8.7	25 May 2022	<p>Additions:</p> <ul style="list-style-type: none"> • A new Clusters View (page 28) has been added to the Floorplanner Perspective. <p>Updates:</p> <ul style="list-style-type: none"> • Concepts (page 6): The Properties View (page 123) has gained an action/dialog to Save Changed Properties. The Create Placement Region Dialog (page 167) has been updated with new groupings for the available Region Alignments and Region Types, with new options for each. The Floorplanner View (page 43) can now show Cluster information in the tooltips. The NoC Performance View (page 87) now includes tooltip information reporting the comparative "blocked" and "transferred" percentages for the "trying" times, and includes a brief table describing the Preferences for this view. • Tasks (page 282): The Creating a New Placement Region (page 395) page has been updated to show the latest Region Alignment (-snap) and Region Type (-type) choices for the associated dialog. • ACE Tcl Commands (page 610): <ul style="list-style-type: none"> ◦ The create_region (page 624) command has been updated with new options <code>-snap [none tiles fabric_clusters clock_regions]</code> and <code>-type [inclusive keepout soft]</code>. The former <code>-snap_to*</code> and <code>-soft</code> options are now deprecated. ◦ The highlight (page 664) command has added a new option <code>-clear</code>. ◦ The set_region_bounds (page 715) command has been updated with new options <code>-snap [none tiles fabric_clusters clock_regions]</code> and <code>-type [inclusive keepout soft]</code>.

Version	Date	Description
8.8	18 Jul 2022	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts: Added a page for the new Load Acxdb Dialog (page 174). Added a page for the new NoC Time Slice View (page 93). Added a page for the new Plot SerDes Diagram Dialog (page 178). Added a page for the Configure Clock Pre-Routes Dialog (page 155). • Tasks: Added a new page for Cleaning Projects (page 304) under Working with Projects and Implementations (page 293). Added a new page for Plotting SerDes Receiver Diagrams Using JTAG (page 482) • ACE Tcl Commands: added new commands add_clock_preroute (page 610), remove_clock_preroute (page 672), save_clock_preroute (page 704), clean_project (page 616), get_compatible_ordering_codes (page 644), get_pvt_corners (page 658), is_labmode (page 666), set_region_type (page 716) • Note: Early access support for Partial Reconfiguration has been added, relevant documentation will be included in a future release. <p>Updates:</p> <ul style="list-style-type: none"> • Concepts: The Netlist Browser View (page 79) has three new boolean toggle filters for the Instance Names column for Constants, Duplicates/Clones, and Feedthroughs. The Clock Regions View (page 23), Clusters View (page 28), Partitions View (page 107), and Placement Regions View (page 110) now display and allow configuration of Clock Pre-Route information when relevant. The NoC Performance View (page 87) page was updated with the latest changes to the related preferences. The Floorplanner View Colors and Layers Preference Page (page 194) was updated to reflect the Always Show Highlighted Instances option, as well as the latest locations/names of several preferences. • Tasks: Updated the Running ACE (page 282) page to reflect the changes to Lab Mode functionality (lab mode now allows a restricted subset of Tcl functionality, specifically the commands necessary for JTAG operations). Added a section describing ACE Startup Arguments to the Running ACE (page 282) page. Updated Running the Entire Flow (page 306) to mention the new Load Acxdb Dialog (page 174) which can appear when a run is canceled. • Troubleshooting: Added a new section describing a "Failed to create the part's controls" error message, along with a workaround. Added a new section about font and image scaling (for high resolution/DPI monitors) in Windows. • ACE Tcl Commands: The create_region (page 624) command has a new <code>-pr_zone</code> argument to designate partial reconfiguration zones. The get_regions (page 659) commands has a new <code>-verbose</code> argument to print region information. The load_flowscripts (page 667) command has a new <code>-bitstream</code> argument to load only those scripts relevant to bitstream manipulations. The <code>move_partition</code> command's arguments have changed.

Version	Date	Description
8.8	18 Jul 2022	<p>Deletions:</p> <ul style="list-style-type: none"> ACE Tcl Commands: The obsolete process_move_partition command has been removed; use set_placement (page 713) -partition instead. <p>Minor enhancements to document style added throughout.</p>
9.0	10 Feb 2023	<p>Additions:</p> <ul style="list-style-type: none"> Tasks: Added new section for the Partial Reconfiguration Tutorial (page 484)(s) ACE Tcl Commands: Added new commands get_synprj_from_project (page 661), move_relative_paths (page 670), run_generate_final_reports (page 691), save_partition_placements (page 705) <p>Updates:</p> <ul style="list-style-type: none"> Concepts: On the Flow Steps (page 234) page, added the new flow step named 'Generate Final Reports'. Updated Design Completion Steps (page 240) page to add description for the new flow step 'Generate Final Reports'. Updated Clock Domains View (page 20), Clock Regions View (page 23), Clusters View (page 28), Partitions View (page 107), and Placement Regions View (page 110) with refreshed screenshots and descriptions for the new functionality to support Partial Reconfiguration. Updated the Save Placement Dialog (page 182) page to reflect the latest options. Tasks: Updated the Multiprocess task page Running Multiple Flows in Parallel (page 308) to mention node-locked license management; the page previously only mentioned floating license concerns. ACE Tcl Commands: add_clock_preroute (page 610) has a new argument -data_region; add_project_netlist (page 612) and remove_project_netlist (page 675) have a new argument -impl; create_boundary_pins (page 620) has a new argument -purpose; find (page 638) has improved help text describing pin pattern matching; get_impl_option (page 648) has a new argument -syn; the help content for report_checks (page 606) has been tweaked to improve clarity; report_power (page 683) now has added support for saif files; report_routing (page 685) has a new argument -wl; source_encrypted (page 716) has a new argument -untar; write_bitstream (page 719) has a new argument -pcie; write_partition_db (page 721) has a new argument -include_pr_zone; Throughout the document, hyperlinks that point outside the document have been updated to static text to prevent outside content from appearing in the GUI help. Throughout the document, content has been edited to improve clarity and consistency. <p>Deletions:</p> <ul style="list-style-type: none"> ACE Tcl Commands: The obsolete move_partition has been removed; use set_placement (page 713) -partition instead.

Version	Date	Description
9.1	26 Apr 2023	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts: Created a new page describing the Project concept of Port Mapping Files (page 228) (as used in IO Designer). Created info covering the new I/O Designer Preference Page (page 204), new Netlist Browser Preference Page (page 208), and new NoC Performance View Preference Page (page 209). • ACE Tcl Commands (page 610): Added new command remap_partial_bitstream (page 672) <p>Updates:</p> <ul style="list-style-type: none"> • Concepts: Updated the Save Placement Dialog (page 182) page to reflect the latest options. Updated the Configure Clock Pre-Routes Dialog (page 155) to reflect the latest functionality. • Troubleshooting (page 761): Removed info related to obsolete versions of Windows. Updated Linux usage directions for the fallback html browser (to improve functionality under Wayland). Updated Windows info related to upgrading ACE and multiple parallel installs of ACE. • ACE Tcl Commands (page 610): Updated the arguments for run_timing_analysis (page 702); Clarified the usage description of set_partition_force_changed (page 711). • Throughout the document, content has been edited to improve clarity and consistency.

Version	Date	Description
9.2	27 Oct 2023	<p>Additions:</p> <ul style="list-style-type: none"> • Concepts: Created a page describing the Create a New Flow Step dialog (page 159). Created a page describing the new Register Browser View (page 127). • Tasks: Created a new page Viewing the Captured VCD Waveform (page 378) to better describe the uses of the VCD Waveform Editor (page 11). Created a new page describing the creation (and removal) of Custom Flow Steps (page 330). <p>Updates:</p> <ul style="list-style-type: none"> • Concepts: Updated the Download View (page 40) page to reflect the changes that occurred as the bitstream files migrate from STAPL *.jam files to simpler *.hex files: the lists of procedures and actions are no longer relevant. Updated the Configure JTAG Connection Preference Page (page 190) to remove mentions of the obsolete Bitporter pod, now mentioning the newer Bitporter2 pod and FTDI FT2232H/FT4232H connectivity options. Updated Save Placement Dialog (page 182) page to reflect the latest options, and to ensure the field names in the dialog are consistent with the arguments for the Tcl command save_placement (page 706). Updated the VCD Waveform Editor (page 11) page with the new actions to copy signal name or value to the clipboard, as well as mentioning all the new waveform color and font preferences, with an updated screenshot showing the current default colors and the new signal row highlight color, as well as a new screenshot example showing users how to achieve a green-on-black color configuration (similar to a traditional oscilloscope) using the new color preferences. Updated the Flow View (page 53) page to mention the new per-flow-step error count tracking and filtered error message dialog. Updated the Projects View (page 117) page to include the new context menu actions for report grouping and filtering. Updated the Project Management Preference Page (page 214) with a new screenshot and descriptions of each available setting. Floorplanner View (page 43) was updated with a new dedicated Panning Tool, a quick toggle between placement/panning modes, and automatic panning during drag-and-drop operation. Floorplanner View Colors and Layers Preference Page (page 194) describes the new "drag scroll" settings. I/O Layout Diagram View (page 67) now mentions the support for double-click and drag-and-drop operations. The list of supported filters (used in the find (page 638) and filter (page 637) Tcl commands, and in the Search View (page 129)) has been updated on the Filter Properties (page 264) page.

Version	Date	Description
9.2	27 Oct 2023	<p>Updates (cont.):</p> <ul style="list-style-type: none"> • Tasks: Updated the Programming a Device using JTAG in the Download View (page 390) page and child pages (Selecting a Bitstream File (page 390), Selecting Bitstream Programming Options (page 391), and Downloading the Bitstream to the Target Device (page 391)) to reflect the changes/simplifications to the Download view's functionality as part of the migration from STAPL *.jam files to *.hex files. Updated the Configuring the JTAG Connection (page 360) page now that STAPL files, the <code>acx_stapl_player</code>, and the classic Bitporter(1) pod are deprecated, mentioning the replacements instead where relevant. Updated the Snapshot-related Collecting Samples of the User Design (page 376) page to mention the need for JTAG preferences to be pre-configured before use, and to reflect the migration from using STAPL to the new <code>Tcl jtag :</code> libraries, and the related impact to manual connection management. Floorplanner Panning (page 340) was updated to mention the new Pan/Place toggle behavior, the new dedicated Panning Tool, and the new support for panning during drag-and-drop. The Running ACE (page 282) page now describes how to use the optional ACE_INIT_SCRIPT functionality. • Tcl Command Reference (page 580): The display_file (page 627) command now supports an optional <code>-search</code> argument. The get_project_netlist_files (page 655) command supports a new argument <code>-noimpl</code>. The logging command message (page 668) supports a new argument <code>-none</code>. The report_checks (page 606) command clarified the description of the <code>-sort_by_slack</code> argument. The run_fpga_download (page 690) command has been updated to work with *.hex files instead of STAPL *.jam files, and no longer produces a separate log file (the standard ACE log file is used instead during the download). The run_timing_analysis (page 702) command supports new arguments for <code>-voltage</code> and <code>-grade</code>. The set_clock_type (page 709) command supports a new option <code>-data_column</code>. The write_bitstream (page 719) command now creates *.hex files instead of STAPL *.jam files. • Troubleshooting (page 761): Added a new entry describing how to deal with a rare file cache corruption problem, seen as the symptom <code>NoClassDefFoundError</code>. Tweaked several entries to acknowledge that ACE now supports additional Linux distros/versions beyond RHEL. • Throughout the document, content has been edited and some screenshots have been updated to improve clarity and consistency.

Version	Date	Description
10.0	16 Apr 2024	<p>Additions:</p> <ul style="list-style-type: none"> • Tcl Command Reference: A new suite of commands under the <code>ace_ip::</code> namespace has been added under the heading TCL Commands for the IP Generator Plugin Framework (page 722). New commands in the <code>ace::</code> namespace have also been added in several categories, including for project source files: <code>add_project_source_files</code> (page 612), <code>get_project_source_files</code> (page 657), <code>enable_project_source_file</code> (page 635), <code>disable_project_source_file</code> (page 627), <code>remove_project_source_files</code> (page 675), <code>import_synplify_project_file</code> (page 665), <code>move_project_source_file</code> (page 669) ; for implementation options and (new) project options: <code>create_option_sets</code> (page 622), <code>get_impl_option_is_supported_and_enabled</code> (page 649), <code>get_package_names</code> (page 650), <code>get_project_option</code> (page 655), <code>get_project_option_is_supported</code> (page 656), <code>get_project_option_is_supported_and_enabled</code> (page 656), <code>get_project_output_directory</code> (page 656), <code>is_impl_option_enabled</code> (page 665), <code>is_project_option_enabled</code> (page 666), <code>report_project_options</code> (page 684), <code>reset_project_option</code> (page 687), <code>set_project_option</code> (page 714) ; for implementation source files: <code>get_enabled_source_files</code> (page 646) ; for the flow: <code>run_synthesis</code> (page 701), <code>run_simulation</code> (page 700) ; for timing analysis <code>get_path_ids</code> (page 653), <code>get_worst_path</code> (page 663), <code>is_timing_met</code> (page 667) ; and other misc: <code>source_secure</code> (page 717), <code>write_aeskey_efuse_bitstream</code> (page 718). <p>Updates:</p> <ul style="list-style-type: none"> • Major updates throughout the document to reflect the changes for the new Unified ACE flow: new Synthesis and Simulation Flow Steps (page 234), restructured flow and related changes to the Flow View (page 53), new Project Options/Impl Options infrastructure with related changes to the Options View (page 96) and Configuring Project and Implementation Options (page 304), new ACE project file structure with related changes to the Projects View (page 117), related new features and actions throughout ACE, etc. • The ACE Quickstart Tutorial (page 3) was updated to reflect the major changes for the Unified ACE flow, particularly to the Projects View.

Version	Date	Description
10.0	16 Apr 2024	<p>Updates (cont.):</p> <ul style="list-style-type: none"> • Concepts: The Tcl Console View (page 142) has been rebuilt to significantly improve performance, and has gained many new features. The Snapshot Debugger View (page 137) is now aware of changes to the <code>pwd</code> from the Tcl console when calculating paths relative to the Working Directory. The VCD Waveform Editor (page 11) now has additional abilities to expand and collapse buses, as well as a button to reset the table and waveform to default contents. The associated Add Signals to Waveform Viewer Dialog (page 146) was also updated with new bus and bus bit handling details. The pages for Flow Mode (page 242), IP Configuration Steps (page 237), Place and Route Steps (page 238), Design Completion Steps (page 240), Timing Report (page 244), and Multiprocess Summary Report (page 255) were all updated to reflect the current handling of PVT corners. • Tasks: Many new features are described When Configuring Advanced Options (page 374) for the Snapshot Debugger View, paths relative to the Working Directory now take the Tcl interpreter's <code>pwd</code> into account. The Viewing the Captured VCD Waveform (page 378) information has been updated with the changes to bus and bit handling. The pages for Timing Across All Temperature Corners (page 266), Generating Timing Reports (page 353), and Running Multiple Flows in Parallel (page 308) were updated to reflect the current handling of PVT corners. • Throughout the document, content has been edited and some screenshots have been updated to improve clarity and consistency. Additionally, the documentation framework has been updated, so there are likely to be minor formatting changes throughout the document as well. <p>Deletions:</p> <ul style="list-style-type: none"> • Tcl Command Reference: The following commands have been removed: <code>get_fanout</code>, <code>get_pod_names</code>, <code>get_stapl_actions</code>, <code>run_stapl_actions</code>,