

Designers need a way to protect their source IP, and that is often achieved by way of encrypted RTL. The Achronix tool flow using ACE and Synplify Pro supports the use of encrypted IP to enable protection of all or part of an IP's RTL.

Achronix supports the IEEE 1364-2005 standard of Verilog for encryption in ACE. This standard of encryption is also supported by Synplify Pro, Modelsim/Questasim, VCS, Incisive, and Riviera.

Achronix provides a script called `encrypt_ACE_IP.PL` in each ACE installation that can be used to encrypt some or all of the RTL files in a design. This script uses a common encryption method where a random AES-128 session key is used to encrypt the file(s) in question to create an encrypted data block. The same session key is then encrypted within an RSA keyblock for each specific tool vendor, using the tool vendor's public key. The resulting encrypted data block and encrypted key block are bundled together into an encryption envelope. Each tool vendor uses its own private key to handle decryption of the vendor-specific keyblock to retrieve the session key, and subsequently decrypt the file(s). The decrypted file is never written back to disk. Instead, after synthesis or place and route, the design is re-encrypted before being written back out to a file.

This tutorial demonstrates how to encrypt source RTL, and use that encrypted RTL through the Achronix tool flow using Synplify Pro and ACE.

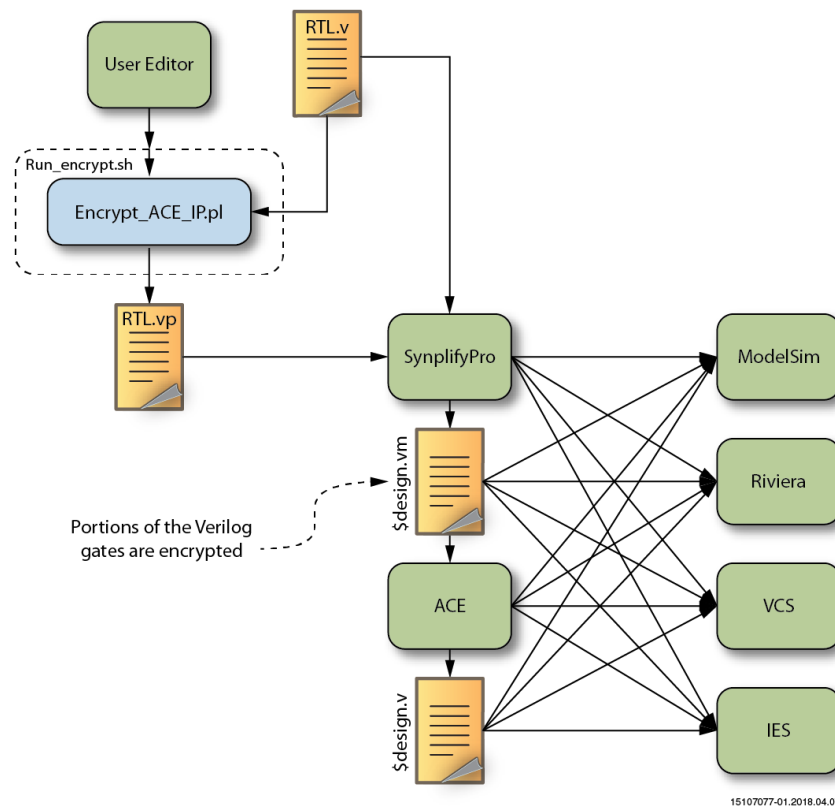


Figure 1 • Encrypted IP Flow Chart

Step 1: Request the Tutorial Files

For this tutorial, two files are needed:

- Tutorial archive, `Speedcore_Encrypted_Compile_RefDesign_RD016.zip`
- Updated script `encrypt_ACE_IP.PL`

To request these files, please file a **document request** ticket at support.achronix.com, with the subject "AN008 Tutorial Files".

Note

In order to file a support ticket, you will need to register for a support account. Details can be found in the support article, [How Do I Register for an Achronix Support Account?](#).

Step 2: Unzip the Tutorial Files

Unzip `Speedcore_Encrypted_Compile_RefDesign_RD016.zip` into a suitable work area. This archive contains the following files and directories:

Table 1 • Tutorial Archive Structure

Directory		Description
<code>run_encrypt.sh</code>		Script to encrypt RTL source files
<code>src</code>		Source file directory
	<code>./ace</code>	Contains example ACE project file, Tcl script to load files and run flow
	<code>./constraints</code>	Contains the SDC constraint file
	<code>./rtl</code>	Contains the source RTL files
	<code>./syn</code>	Contains example Synplify project, Tcl to load design files, log and output files

Step 3: Encrypt Selected RTL Files

A designer can encrypt all or only some of their source RTL files by using the Achronix-supplied utility, `encrypt_ACE_IP.PL`. This utility needs to be run only once.

In the top level of the work area there is a shell script `run_encrypt.sh` which needs to be modified to point to the appropriate versions of files and directories used in this tutorial:

1. Verify the list of files in the directory `src/rtl` that need to be encrypted:

```
rtl_list_of_files_to_encrypt="picorv32.v"
```

2. Change the path to the Achronix release to an absolute path that is appropriate for your site.

```
path_to_achronix=<path_to_ACE_install_location>
```

3. Verify the path to the Perl encryption script, `encrypt_ACE_IP.PL` :

```
path_to_encrypt=$path_to_achronix/Achronix-linux/doc/encrypt_ACE_IP.PL
```

4. After modifying this script as needed, run it to create encrypted RTL file(s). The RTL files defined in the variable `$rtl_list_of_files_to_encrypt` are moved to the directory `src/rtl/encrypt_ACE_IP_dir/`. The script encrypts the entire RTL file and then writes the encrypted files in the same directory. The file extension of the encrypted versions is `.vp`.

```
% ./run_encrypt.sh
```

If a user wishes to encrypt only a section of the Verilog code in a file, the user needs to add pragma statements surrounding the desired encrypted section of the files.

Partially encrypted Verilog is supported in ACE, and can be useful when interfaces for modules are to be exposed, but not the modules themselves.

```
'pragma protect begin  
// Code to encrypt  
'pragma protect end
```

⚠ Caution!

Partially encrypted Verilog is no longer supported in Synplify Pro, as of release 2022.09 and newer. Per Synopsis: "This is an enhancement done to avoid the risk of exposing key details when RTL is encrypted using multiple keys." . Attempting to partially encrypted Verilog in a later version of Synplify Pro will result in the following error written to the .srr-file

```
@E: CG217 : "out.v":10:0:10:8|Detected partially encrypted module in out.v. This is not supported.
```

To work around this restriction, simply separate the blackbox-module definitions that need to be exposed into a separate file.

Step 4: Run Through Flow with Synplify Pro and ACE

ACE 10.0 introduced a unified ACE tool flow. This new streamlined ACE project structure and flow fully integrates synthesis, simulation, and place-and-route into a single ACE project (`*.acxprj`) file and a single graphical user interface. This tutorial will cover both flows:

- **Discrete ACE tool flow**
- **Unified ACE tool flow**

Select which ever flow is appropriate to the project's needs.

Discrete ACE Tool Flow

Set up the Synthesis Project

1. Change to the Synplify directory of the work area:

```
% cd <your_work_area>/src/syn
```

2. In the script `run_synplifyPro_load_verilog.tcl` , modify the variable `set path_to_achronix` to point to the Achronix release area as was done for the shell script `run_encrypt.sh` as shown above.
3. Start Synplify Pro:

```
% synplify_pro
```

If Synplify Pro comes up with a project loaded, click on the "Close Project" button to close it

4. Load the example project by selecting **File** → **Open Project...** then click on **Existing Project** and choose the file `<work_area>/src/syn/partial_encrypted_ref_design.prj` . Alternatively, a Tcl script is available to create the project. Use the Tcl command window to type:

```
% source run_synplifyPro_load_verilog.tcl
```

The Synplify Pro home screen should now appear similar to the [figure](#) below:

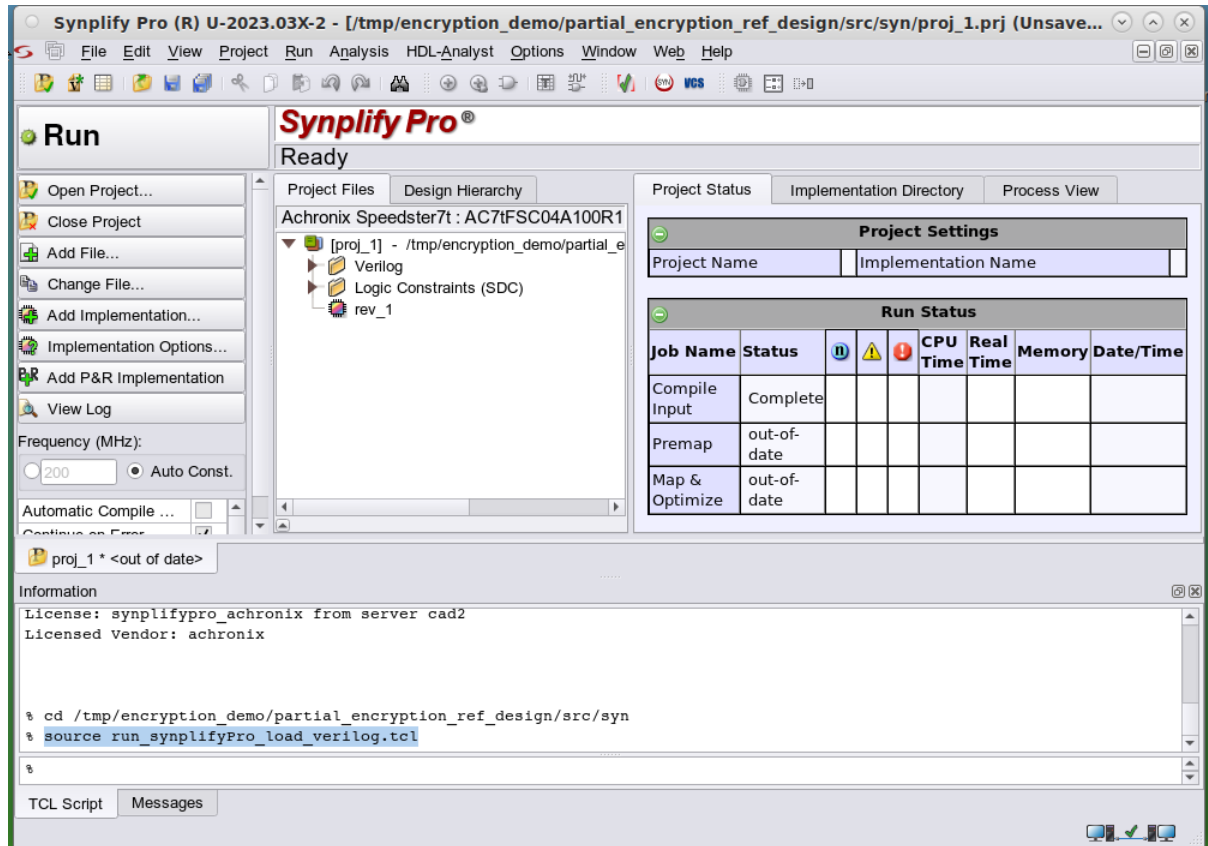


Figure 2 • Synplify Pro Home Screen After Loading/Creating the Project

- Expand the Project File tab's Logic Constraints and Verilog folders, and if necessary, scroll to the bottom. Notice that the encrypted RTL file (`.vp` file extension) is listed just above the top-level RTL file (`picorv32_axi_top.v`), which is always listed last.

Compile the Design in Synplify Pro

- Click on the **Run** button to compile the project. The synthesis is complete when the status reports Done.

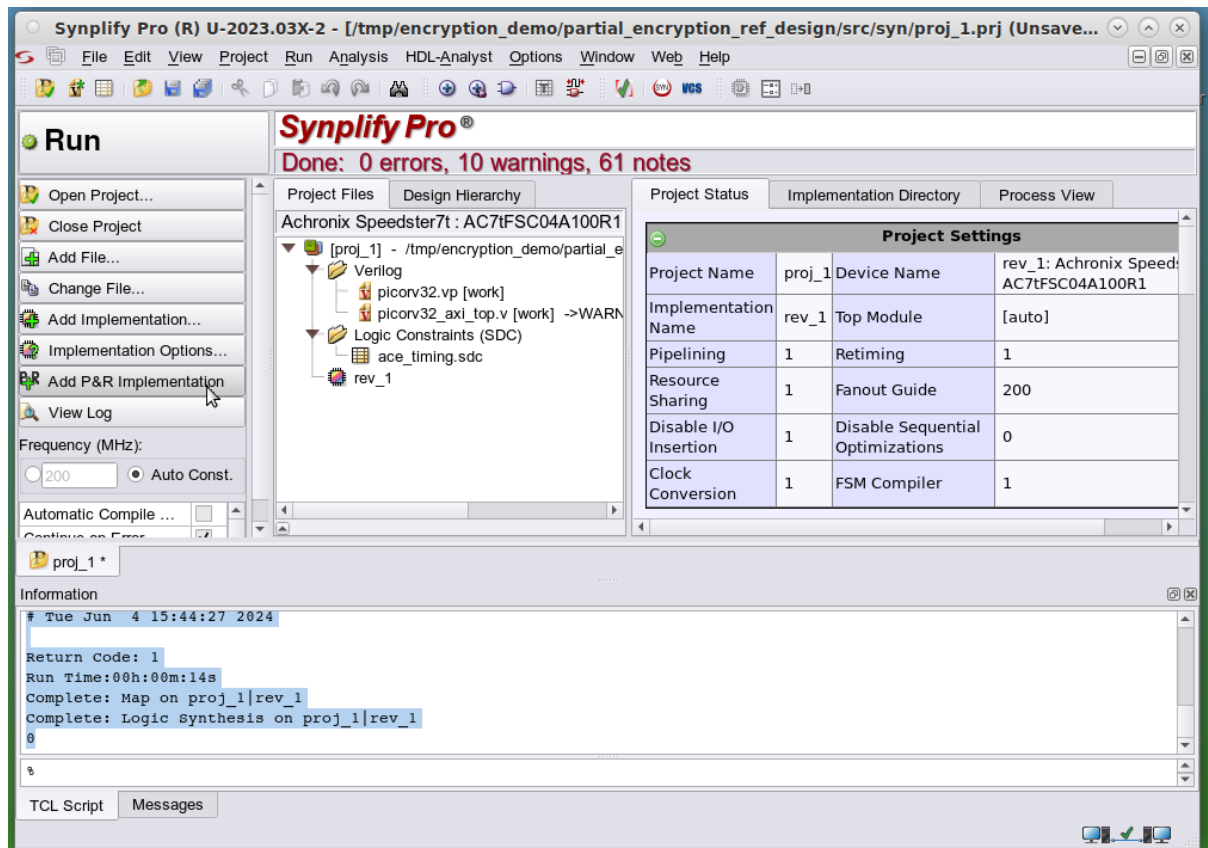
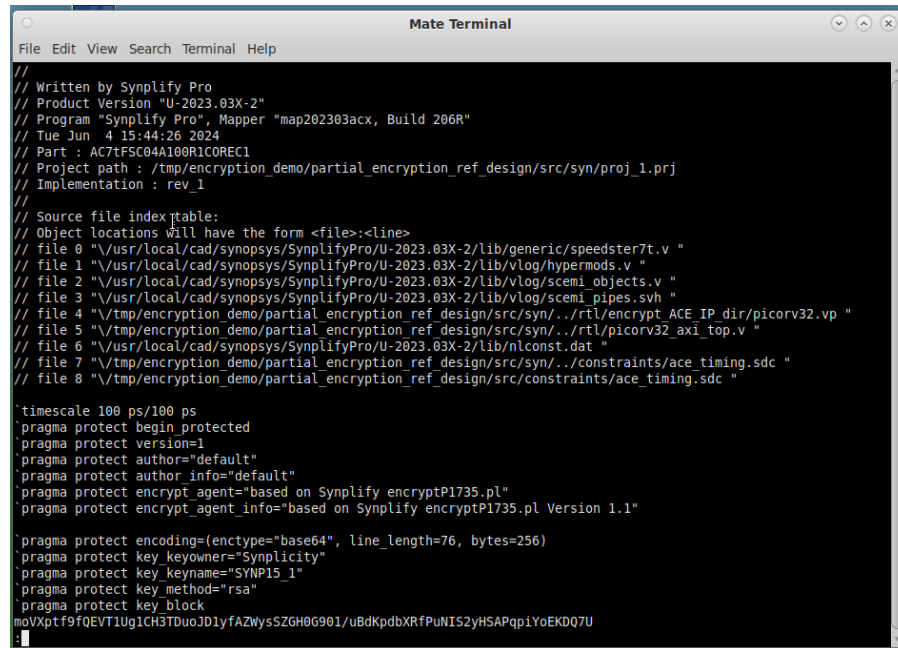


Figure 3 - SynplifyPro Home Screen After Project is Compiled

2. Exit the Synplify Pro session by selecting **File** → **Save and File** → **Exit**.
3. Confirm that the modules defined in `run_encrypt.sh` are not readable in the `rev_1/picorv32_axi_top.v` output Verilog file from Synplify Pro.



```
mate@mate:~$ ace
// Written by Synplify Pro
// Product Version "U-2023.03X-2"
// Program "Synplify Pro", Mapper "map202303acx, Build 206R"
// Tue Jun  4 15:44:26 2024
// Part : AC7tFSC04A100R1COREC1
// Project path : /tmp/encryption_demo/partial_encryption_ref_design/src/syn/proj_1.prj
// Implementation : rev_1
//
// Source file index table:
// Object locations will have the form <file>:<line>
// file 0 "\usr/local/cad/synopsys/SynplifyPro/U-2023.03X-2/lib/generic/speedster7t.v "
// file 1 "\usr/local/cad/synopsys/SynplifyPro/U-2023.03X-2/lib/vlog/hypermods.v "
// file 2 "\usr/local/cad/synopsys/SynplifyPro/U-2023.03X-2/lib/vlog/scemi_objects.v "
// file 3 "\usr/local/cad/synopsys/SynplifyPro/U-2023.03X-2/lib/vlog/scemi_pipes.svh "
// file 4 "\tmp/encryption_demo/partial_encryption_ref_design/src/syn/./rtl/encrypt ACE_IP_dir/picorv32.vp "
// file 5 "\tmp/encryption_demo/partial_encryption_ref_design/src/syn/./rtl/picorv32_axi_top.v "
// file 6 "\usr/local/cad/synopsys/SynplifyPro/U-2023.03X-2/lib/nlconst.dat "
// file 7 "\tmp/encryption_demo/partial_encryption_ref_design/src/syn/./constraints/ace_timing.sdc "
// file 8 "\tmp/encryption_demo/partial_encryption_ref_design/src/constraints/ace_timing.sdc "
//
timescale 100 ps/100 ps
pragma protect begin_protected
pragma protect version=1
pragma protect author="default"
pragma protect author_info="default"
pragma protect encrypt_agent="based on Synplify encryptP1735.pl"
pragma protect encrypt_agent_info="based on Synplify encryptP1735.pl Version 1.1"
//
pragma protect encoding=(enctype="base64", line_length=76, bytes=256)
pragma protect key_keyowner="Synplicity"
pragma protect key_keyname="SYNP15_1"
pragma protect key_method="rsa"
pragma protect key_block
moVXptf9fQEVt1Ug1CH3TDuoJDlyfAZWysSZGH0G901/uBdkpDbXRfPuNIS2yHSAPqpiYoEKDQ7U
:
```

Figure 4 • Sample

Set up the ACE Project

1. Change directories to the ACE work area, and launch ACE:

```
% cd <your_work_area>/src/ace
% ace
```

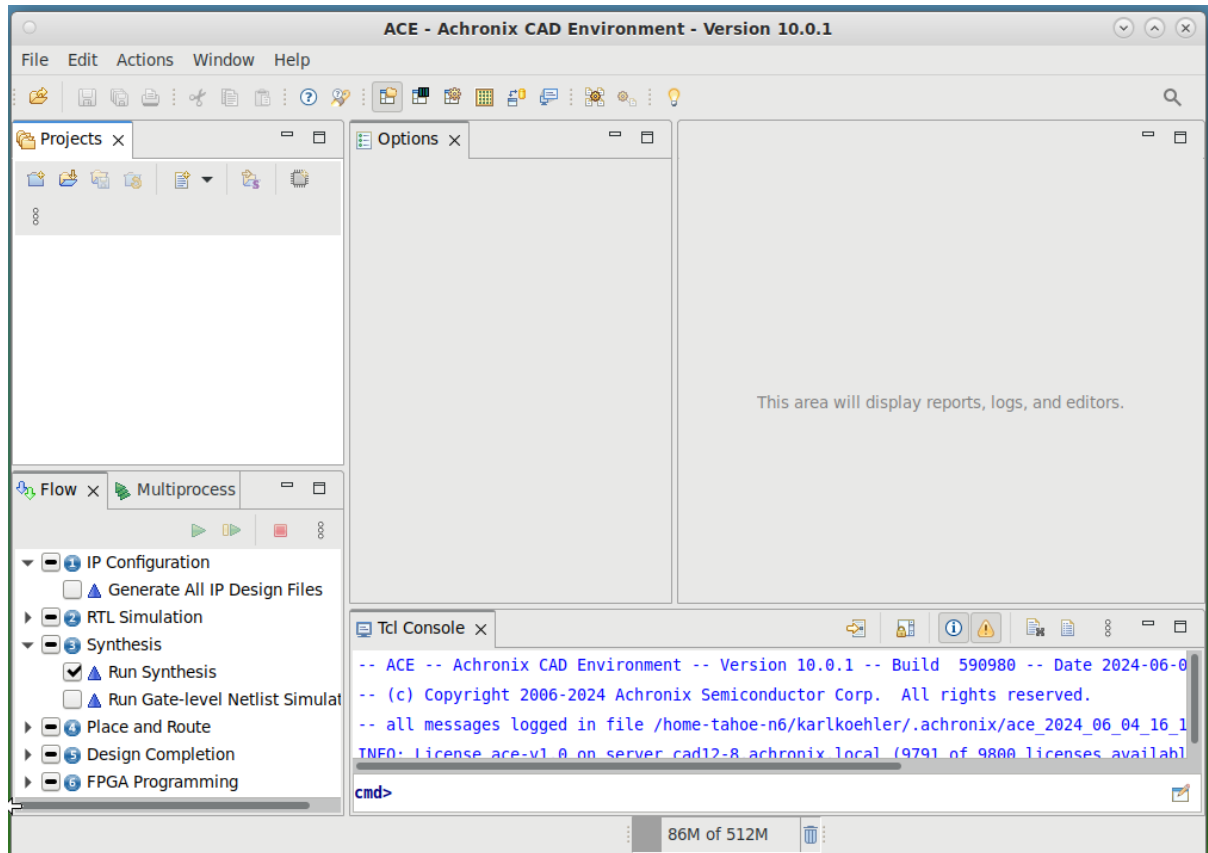


Figure 5 • ACE GUI before Loading a Project

Note

Ensure to remove any existing project from the Project Perspective's Project tab.

- Load the example project by clicking on **File** → **Load Project...** and selecting `<your_work_area>/src/ace/partial_encryption_ref_design.acxprj`, then click **Finish**. Once the project has loaded, double-click **Generate Final Simulation Netlist** in the Flow window to run the project through the flow.

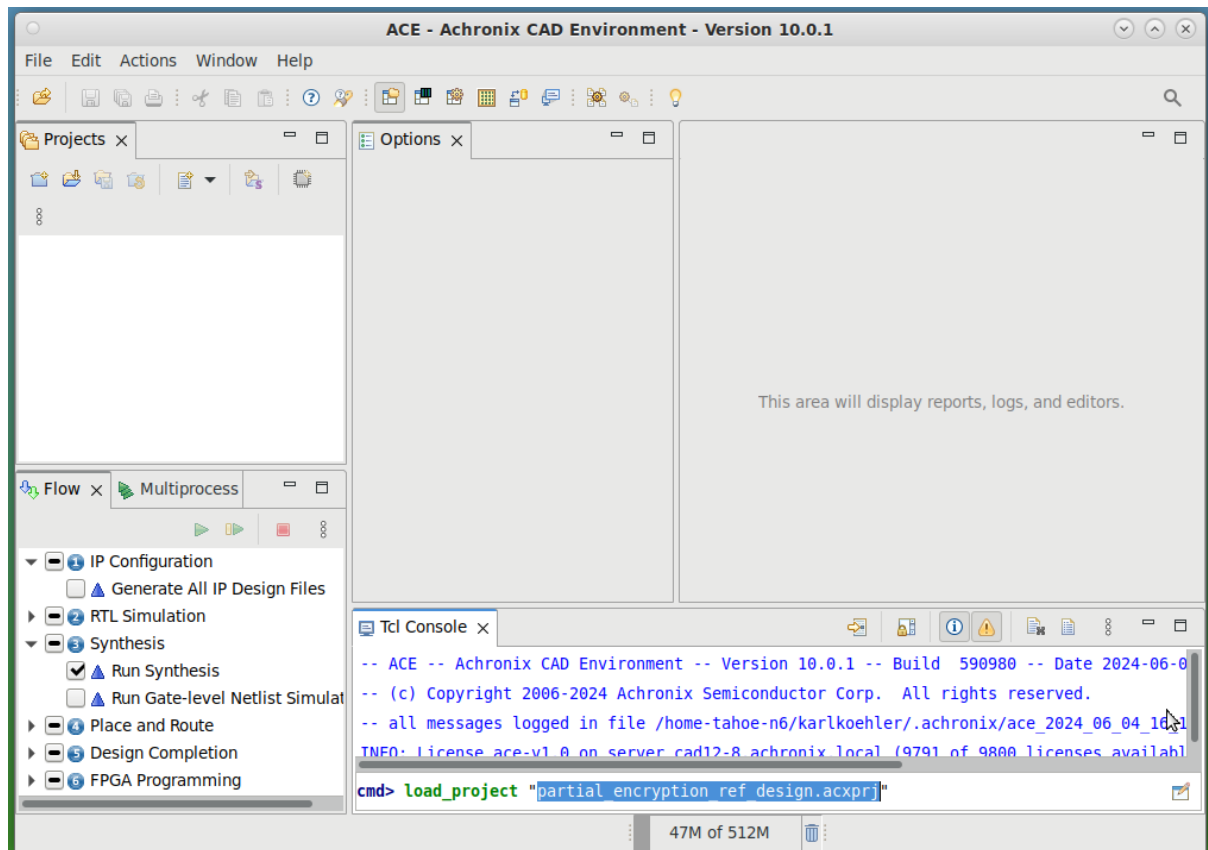


Figure 6 • ACE GUI Loading

3. After loading the project, the screen should appear as follows:

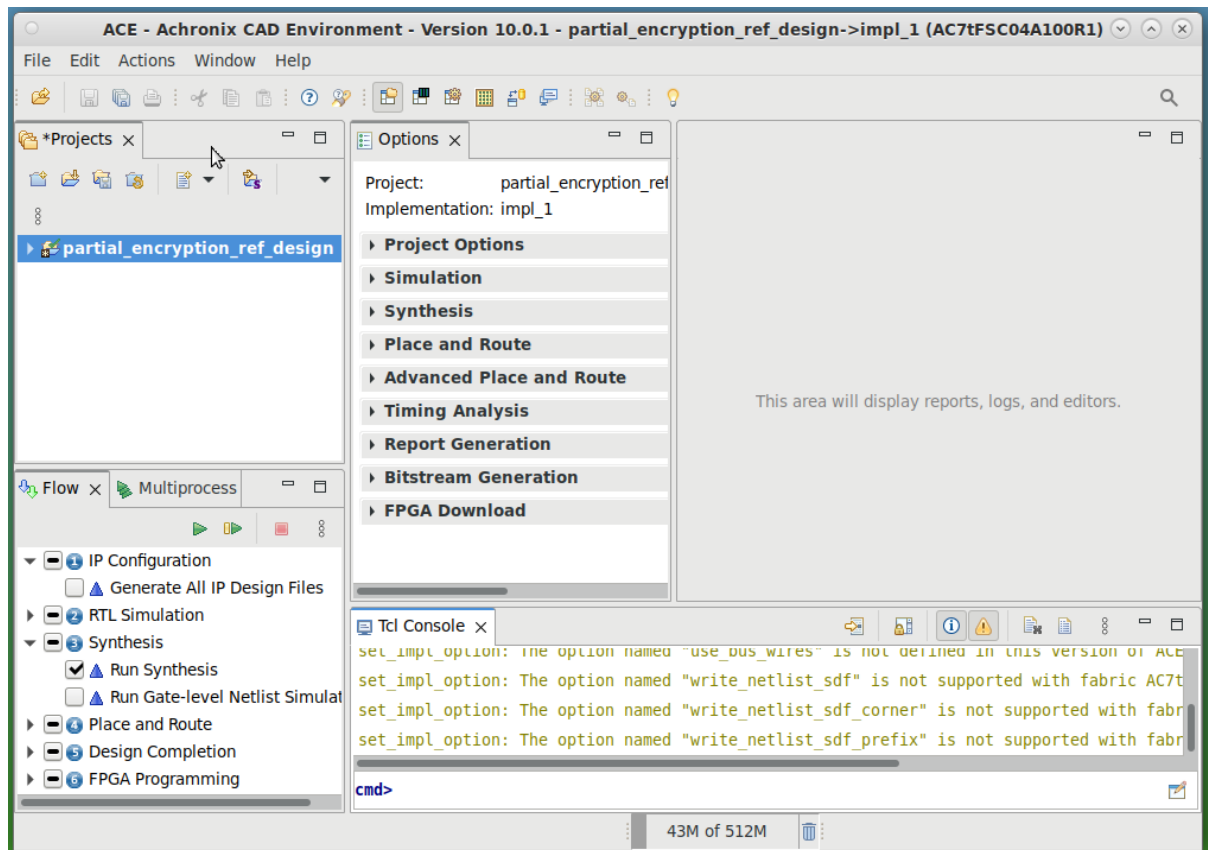


Figure 7 • ACE GUI Project before Running

4. Because synthesis has already been run, uncheck the "run Synthesis" flow step.

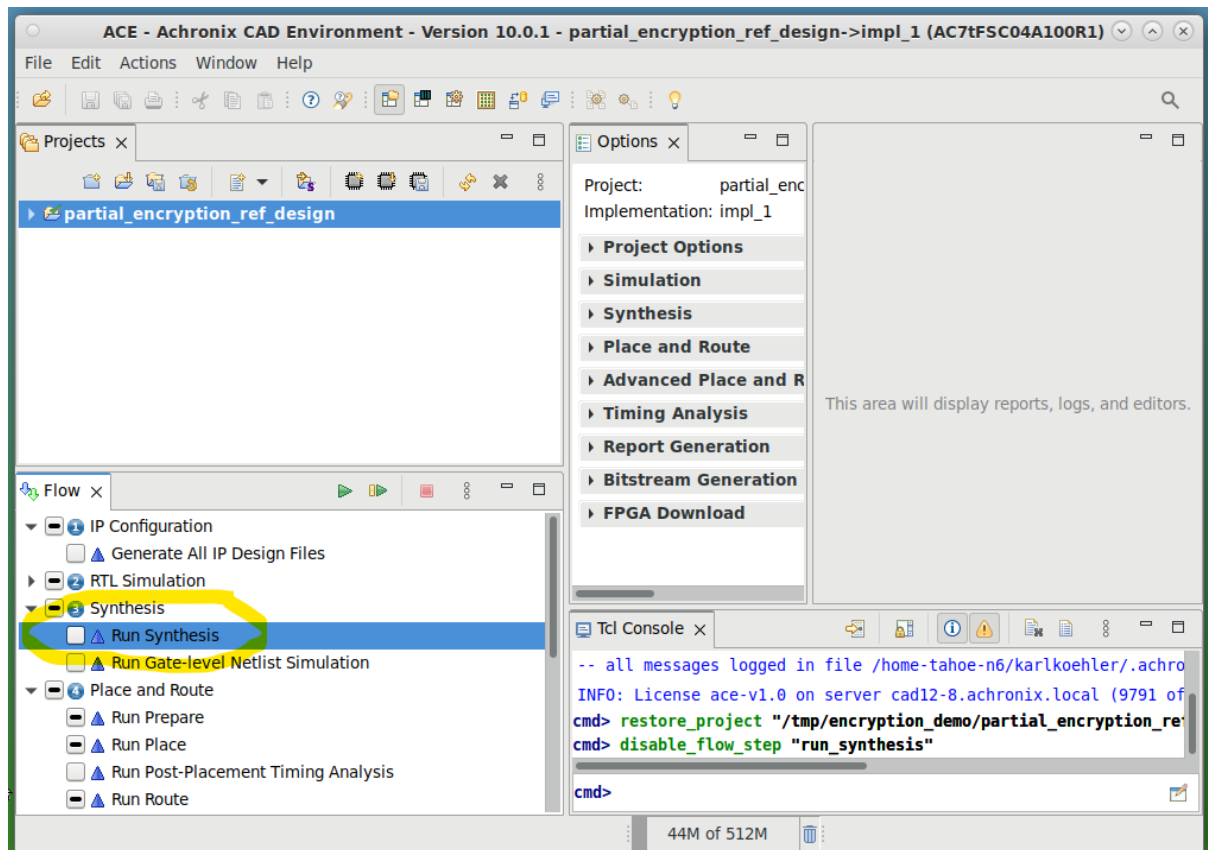


Figure 8 • Unchecking

Alternatively, in the Tcl Console type following command to set up the project and run the flow:

```
source run.tcl
```

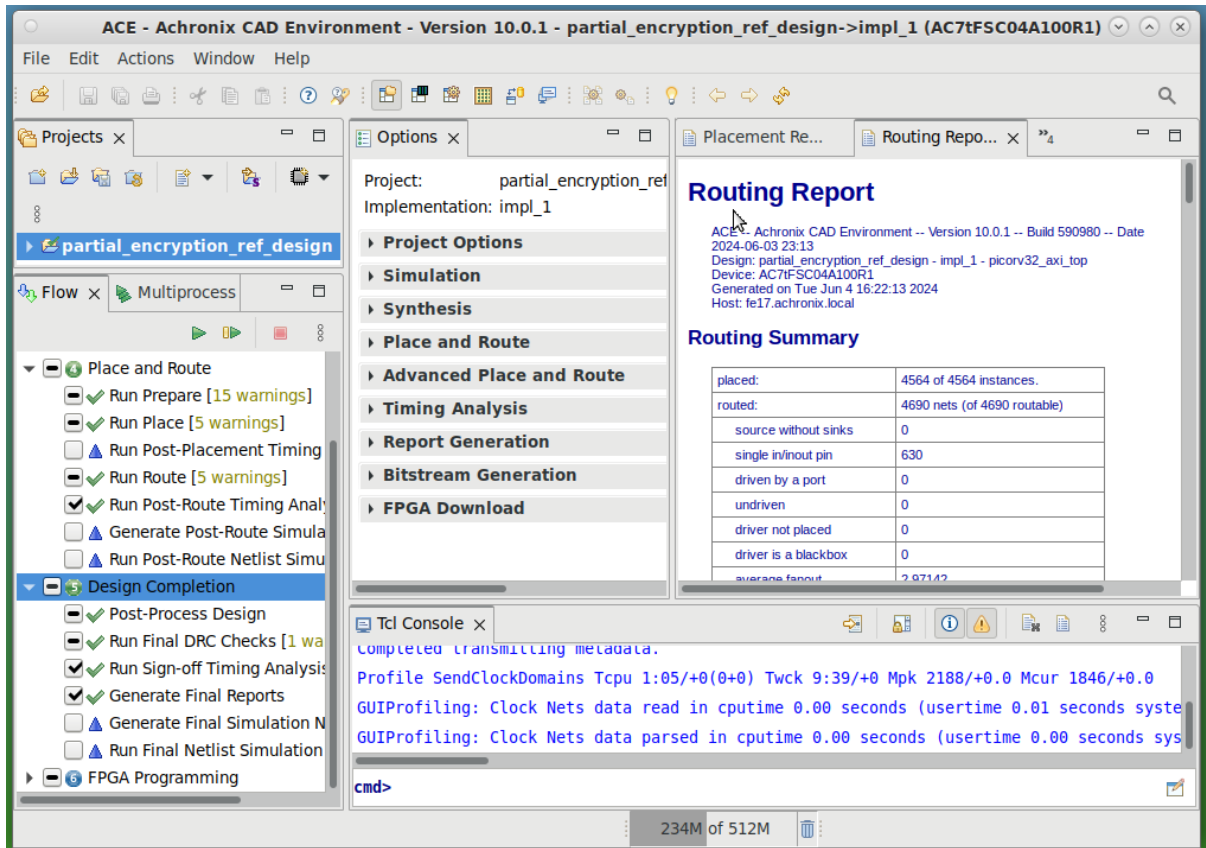


Figure 9 - ACE Screen After Project Compilation

- Exit from ACE. Inspect the resulting Verilog output file in `impl_1/output/picorv32_axi_top.v` to confirm that the modules defined in `run_encrypt.sh` are encrypted in the ACE output.

Unified ACE Tool Flow

Set up the ACE Project

- Ensure that Synplify Pro is available. Set the "ACX_SYNPLIFY_TOOL_PATH" environment variable. On the bash shell:

```
export ACX_SYNPLIFY_TOOL_PATH=`which synplify_pro`
```

If using C-shell, then the following will do the same:

```
setenv ACX_SYNPLIFY_TOOL_PATH `which synplify_pro`
```

- If ACE was already started but the the synthesis step did not run, this situation can still be easily fixed by entering:

```
set ::env(ACX_SYNPLIFY_TOOL_PATH) /path/to/synplify_pro
```

- Then add the files to the project:

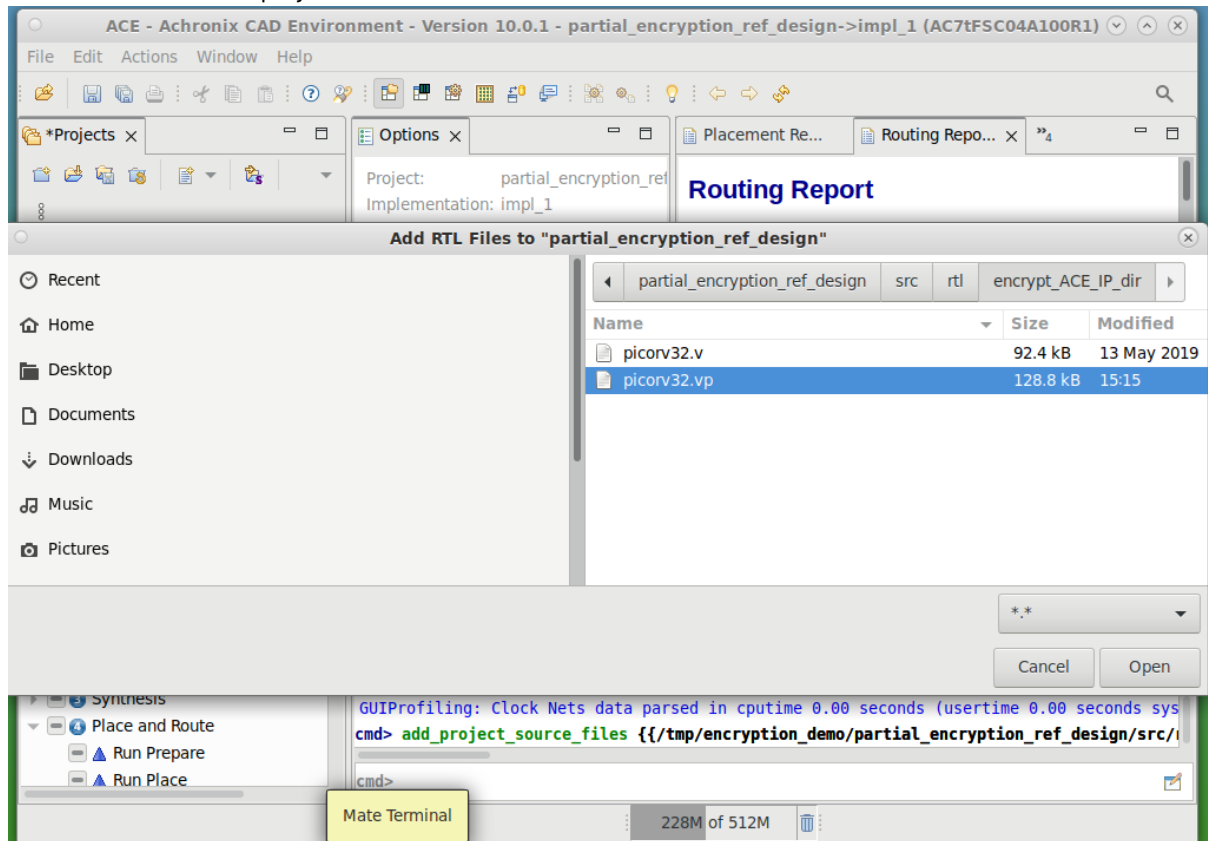


Figure 10 - Adding Encrypted RTL Source File to the ACE Project

- Add both the unencrypted top-file and the just encrypted file to ACE as RTL source file (using '*' as filter). The project should now appear as follows:

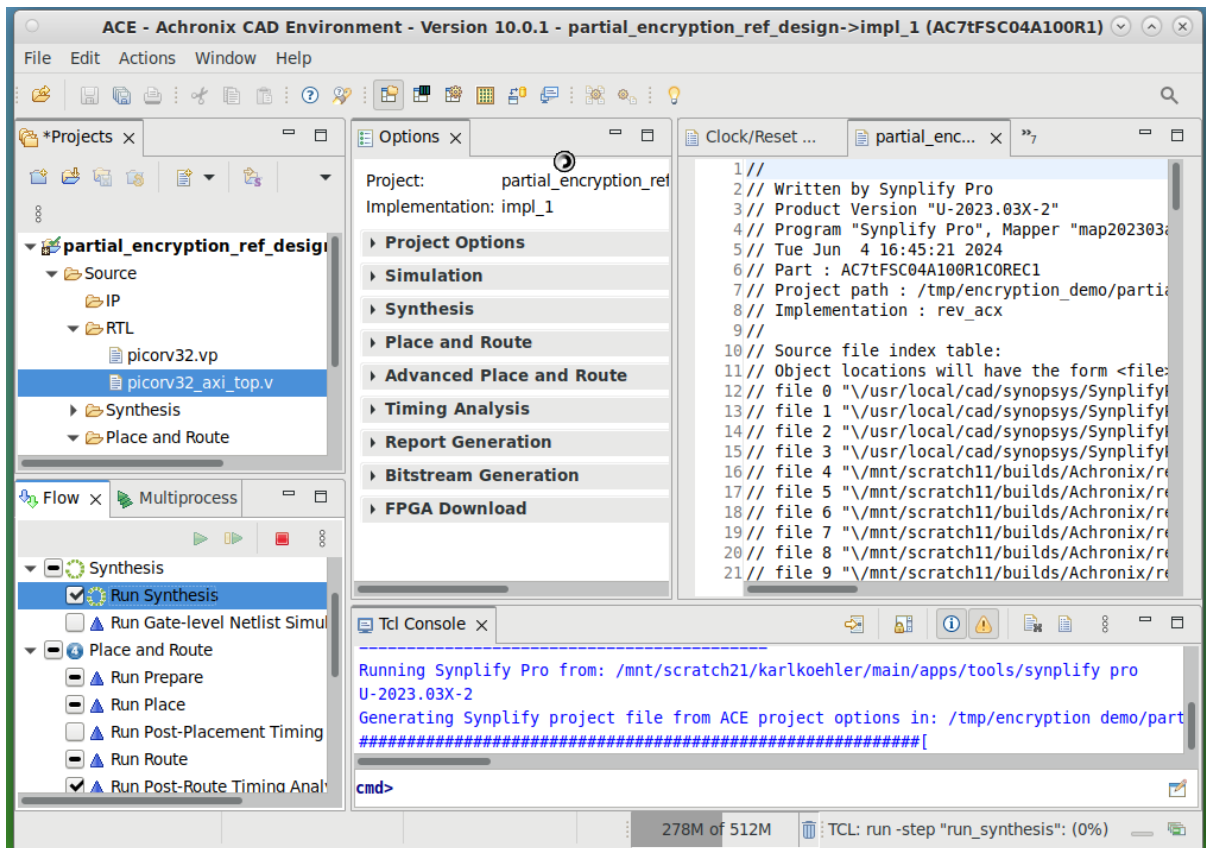


Figure 11 - Running the Synthesis Flow Step in ACE

- Just as with Synplify, source files can be dragged and dropped into the correct order. Ensure that the file containing the **top** module is at the bottom. Now click the **Run** button to complete the flow.

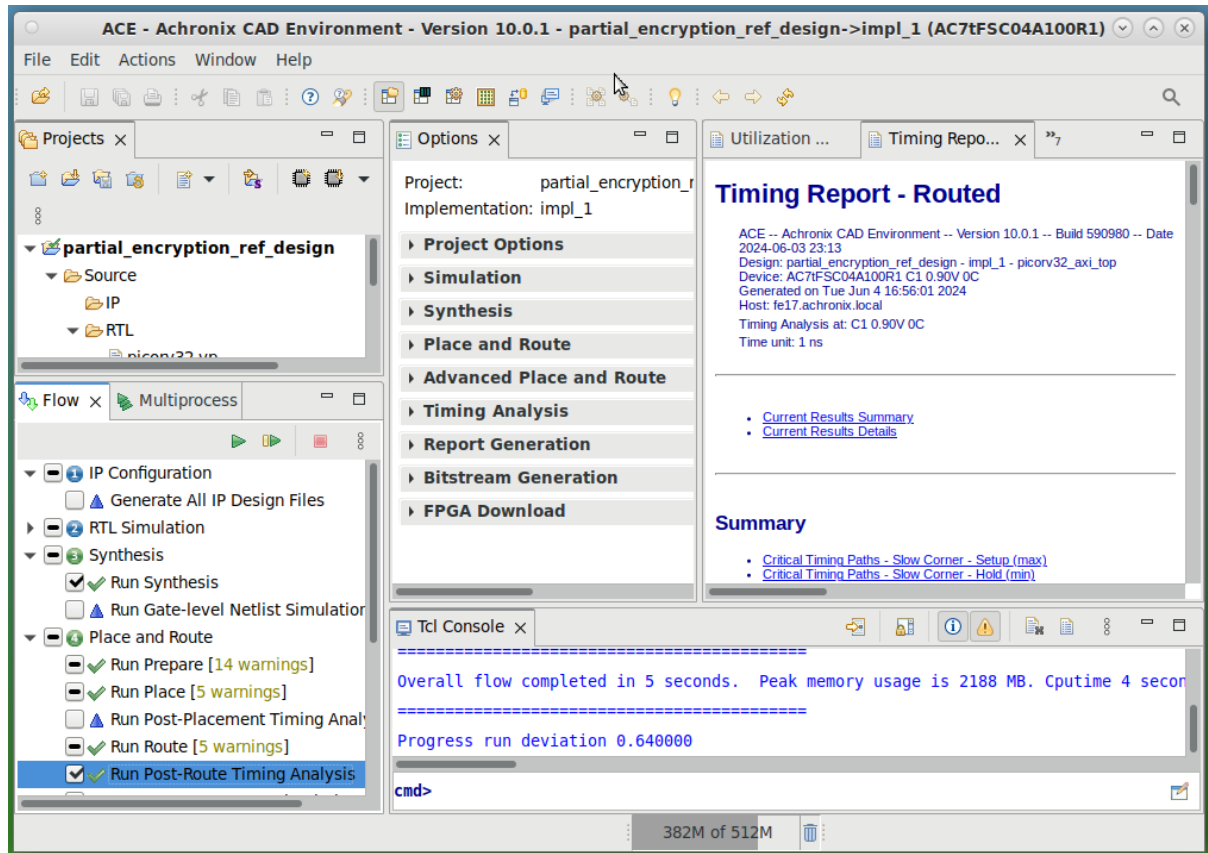


Figure 12 • Successful Synthesis Run in ACE

- Once synthesis completes, the place-and-route source files now contains the fully-encrypted netlist. Verify the fact that the file is indeed encrypted by double-clicking the file name in the GUI. If the place-and-route netlist is not encrypted, double-check that the unencrypted `.v` file was not added by mistake.

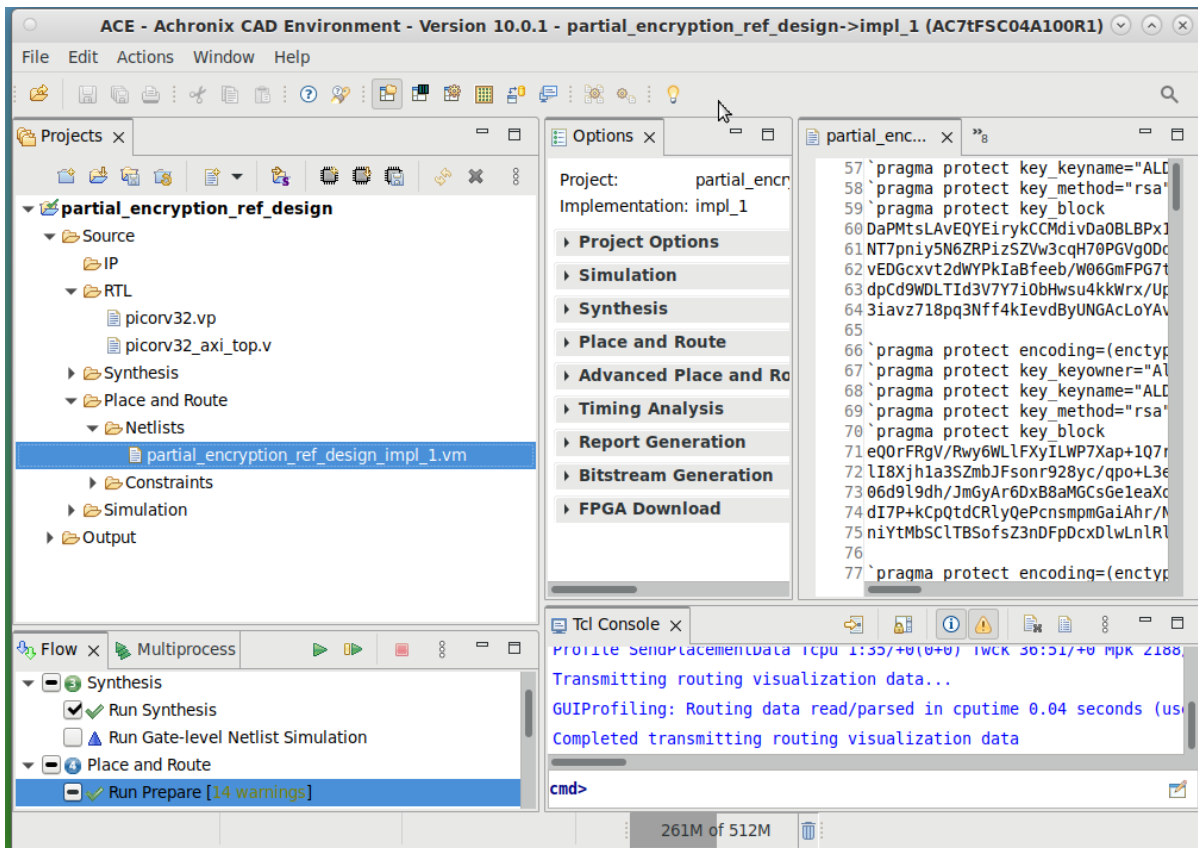


Figure 13 • Viewing the Automatically Generated Encrypted Gate-Level Netlist in ACE

Supported Simulation Tool Versions

This tutorial does not take users through running the encrypted Verilog files in a simulation; however, below lists the simulation tool versions that support encrypted netlist files:

Simulator	Support Version
ModelSim/Questasim	v10.1c or newer
VCS	G-2012.09 or new
Riviera	Riviera-pro-2011.10 or newer
IES	15.20-s008 or newer

Achronix[®]

Data Acceleration

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

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

Copyright © 2024 Achronix Semiconductor Corporation. All rights reserved. Achronix, Speedster and VectorPath are registered trademarks, and Speedcore and Speedchip are trademarks of Achronix Semiconductor Corporation. All other trademarks are the property of their prospective 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 www.achronix.com/legal.