Speedster7t Clock and Reset Architecture User Guide (UG083)

Speedster FPGAs

Achronix
Data Acceleration

Copyrights, Trademarks and Disclaimers

Notice of Disclaimer

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

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

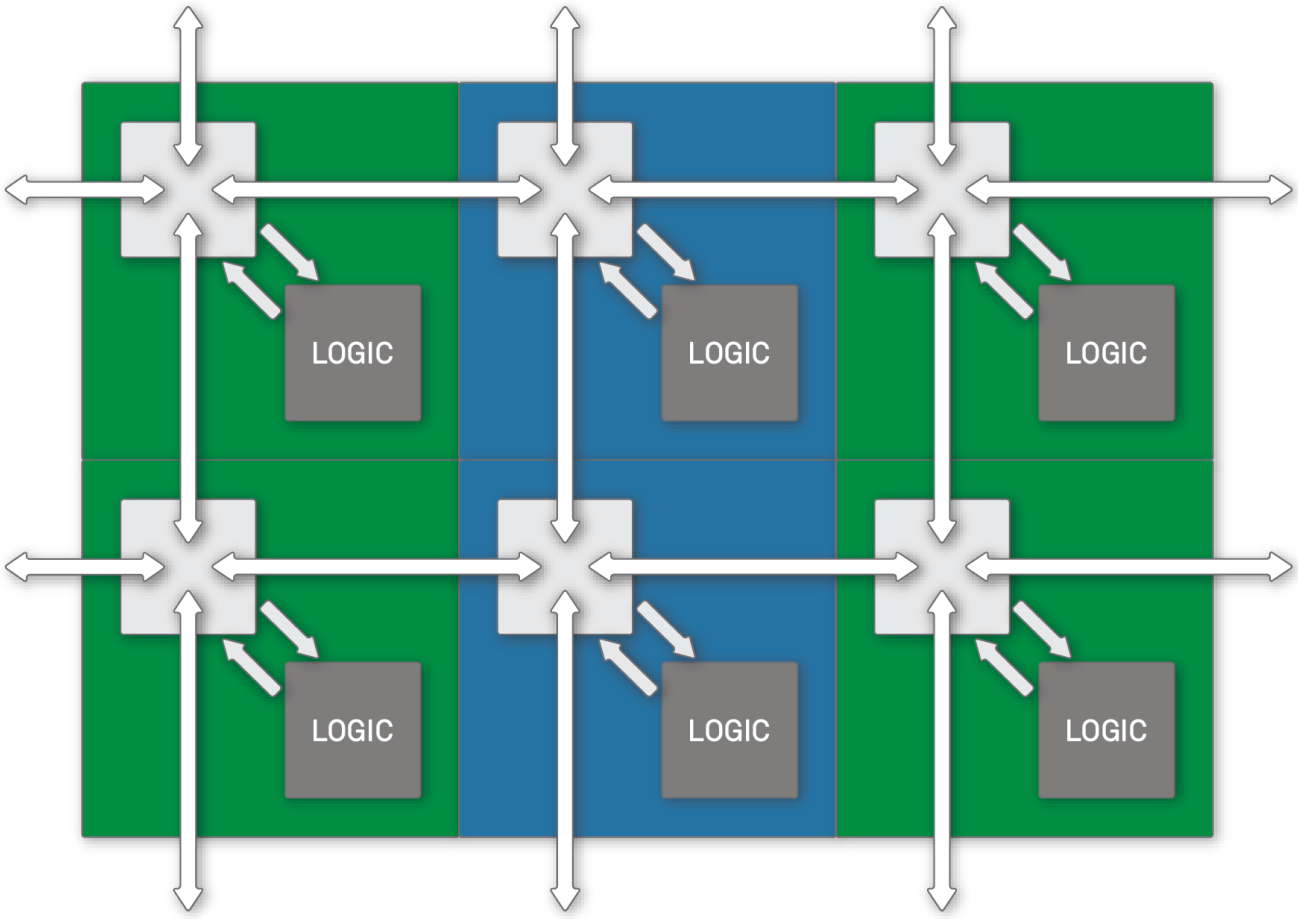
Table of Contents

Chapter 1 : Introduction

The Achronix Speedster7t AC7t700/800 and AC7t1400/1500 FPGAs are specifically designed to deliver extremely high performance for demanding applications including data-center workloads and networking infrastructure. The processing tasks associated with these high-performance applications, specifically those associated with artificial intelligence and machine learning (AI/ML) and high-speed networking, represent some of the most demanding processing workloads in the data center. In order to meet the demand for high-performance and complex designs, the clock network for Speedster7t FPGAs has been designed with numerous high-performance clocks that allow for maximum route-ability. This document explains the architecture of the different clock networks in Speedster7t FPGA, as well as provides information on how to use the clocks. It is intended to help designers understand and choose the best clocking options for their Speedster7t FPGA designs.

Chapter 2 : Fabric Architecture

The Speedster7t FPGA fabric is comprised of two main tile types: reconfigurable logic blocks (RLBs) that contain look-up tables, flip-flops, and ALUs, and machine learning processors (MLPs) that contain multipliers, adders, accumulators, and memory blocks. The tiles are distributed as columns in the Speedster7t FPGA fabric, with each tile consists of a routing switch box plus a logic block. Each type of block is designed to snap together in a grid, where abutting routing networks connect. The figure below illustrates this concept.



3703191-03.2022.07.11

Figure 1 • Tiles Assembled in Fabric

The tiles are grouped to form clusters. Clock signals can enter or exit from the top or bottom on the clock minitrunk (large vertical arrows), and from the left or right on the clock branch (large horizontal arrows). The clock minitrunk and branch are connected at the intersection via a crossbar hub. Clocks are distributed throughout the cluster using clock stems (smaller vertical arrows in the following figure).

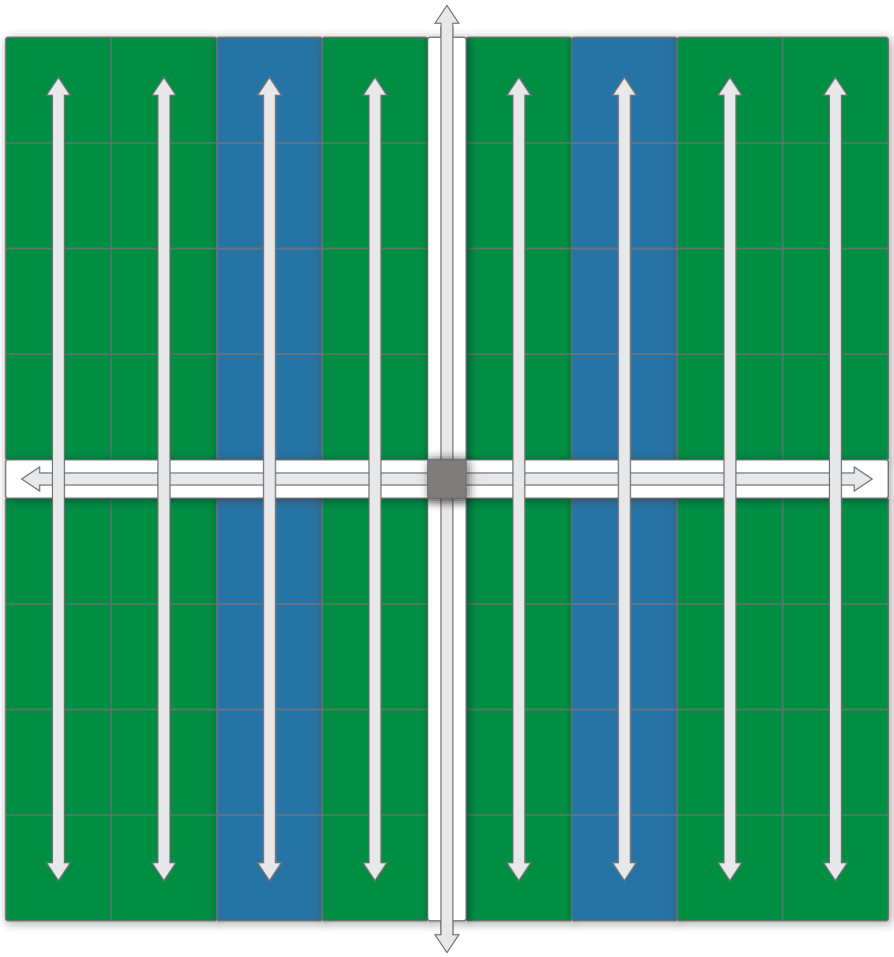


Figure 2 • Clock Distribution Within a Cluster

Clock regions are assembled together to form the fabric core, which is surrounded by high performance and high bandwidth interface subsystems. Speedster7t FPGAs include on-chip oscillators, as well as PLLs which can be used for clock generation. Additionally, the Speedster7t FPGA includes an integrated 2D Network on Chip (NoC) that can interact directly with the fabric.

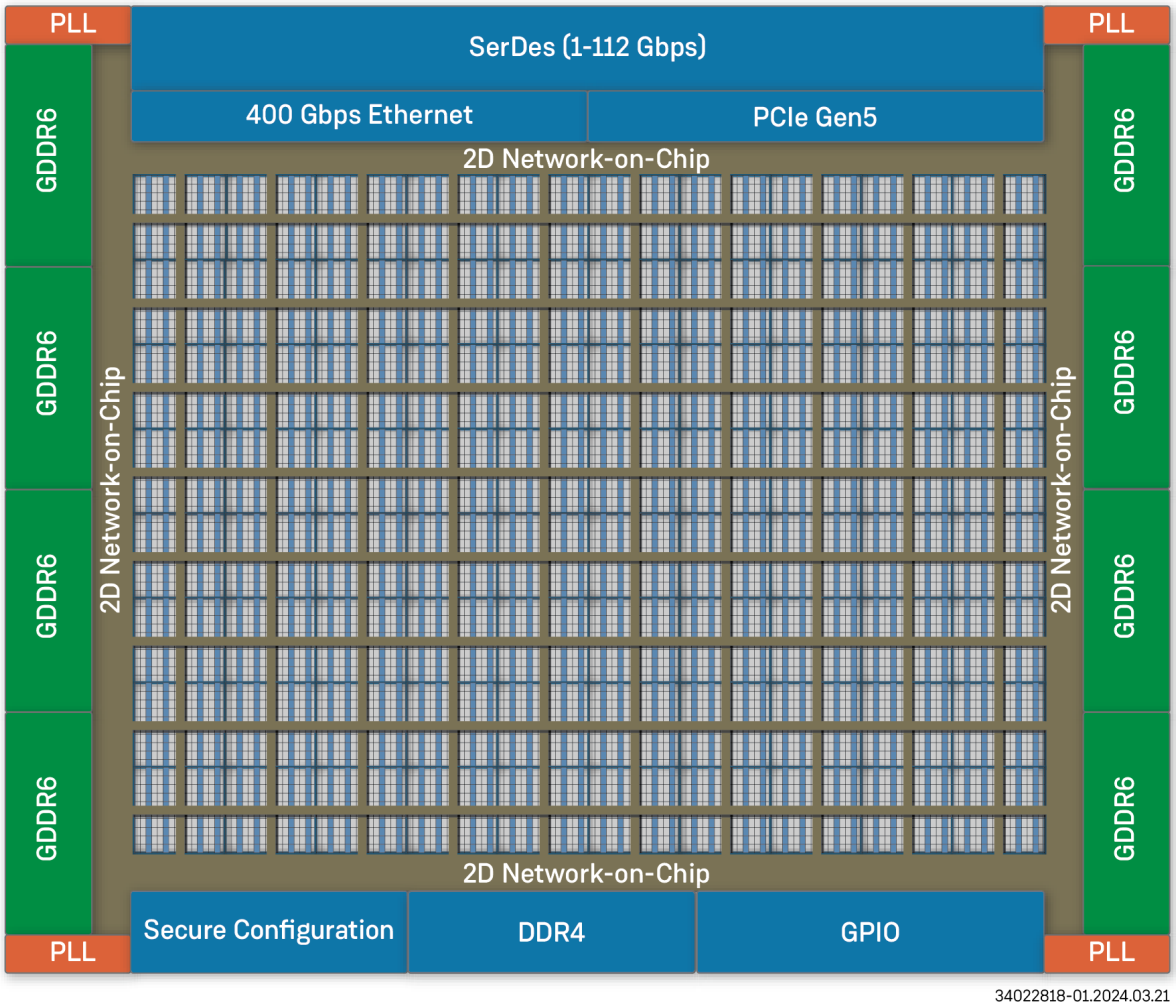


Figure 3 • Speedster AC7t1400/AC71500 Top-Level Floorplan

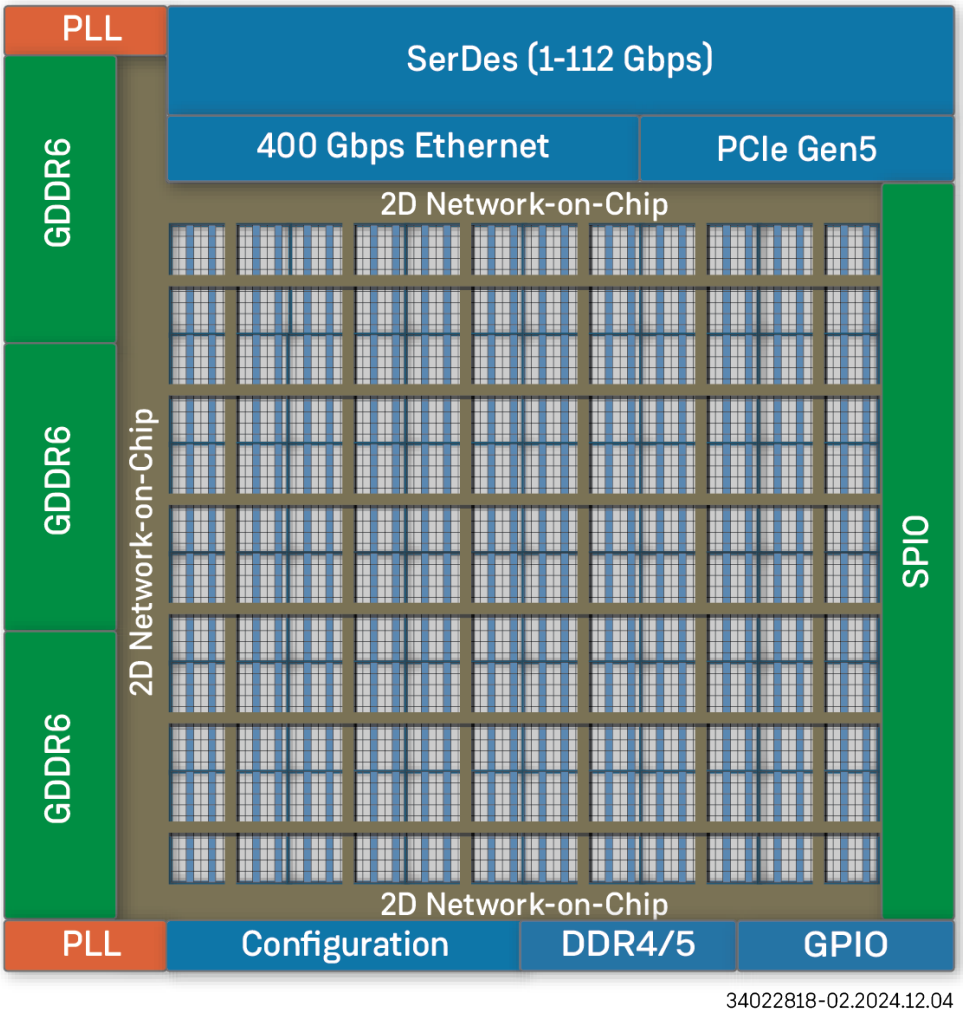


Figure 4 • Speedster AC7t700 AC7t800 Top-Level Floorplan

The Speedster7t FPGA fabric clock network consists of four main parts:

- The clock input pins and on-chip oscillators which can drive PLLs to generate clocks.
- The global core clock network, which is used to drive the majority of logic in the Speedster7t FPGA fabric.
- The interface clock network, which is used to drive logic for the interface subsystems and any associated fabric logic surrounding the corresponding interface.
- The dedicated clocks for the 2D NoC and memory interfaces.

Chapter 3 : Speedster7t Clock and Reset Generation

Clock I/O

There are two types of clock I/O pins located in the corners of the Speedster7t FPGA. Each clock I/O bank contains the following:

- One pair of MSIO, consisting of `MSIO_P` and `MSIO_N`. The pair may be configured as a differential clock input/output or a differential reset input. When not configured as a pair, both the positive and the negative signals can be used as either single-ended clock inputs/outputs or single-ended reset inputs. The MSIO can handle frequencies from 0.75MHz to 500MHz.
- Two pairs of REFIOs, `REFIO_0` and `REFIO_1`, consisting of `REFIO_P_0`/`REFIO_N_0` and `REFIO_P_1`/`REFIO_N_1` respectively. Each pair may be configured as a differential clock input/output or a differential reset input. When the REFIOs are not configured as a pair, the positive signals can be used as either single-ended clock inputs/outputs or single-ended reset inputs but the negative signals can only be used as single-ended reset inputs. The REFIOs can handle frequencies from 10 MHz to 1000 MHz.

Warning!

- If the positive and negative signals in the REFIOs are both enabled to be single-ended, they must have the same port direction. This restriction does not apply to the MSIO.
- The PLL reference clock inputs can only handle frequencies from 5 MHz to 600 MHz. Thus, any MSIO clock inputs going to the PLL reference clock input must be limited to a minimum of 5 MHz or use the **inner** (page 17)/**outer** (page 14) PLL bypass paths. Similarly, any REFIO clock inputs going to the PLL reference clock input must be limited to a maximum of 600 MHz or use the **inner** (page 17)/**outer** (page 14) PLL bypass paths.

The clock I/O bank can handle both reset inputs and clock inputs/outputs, but not data signals.

Configuring Clock I/O

Configuration of all I/O ring interface subsystems, including clocks and PLLs, takes place in simple `.acxip` text files, which can be managed in batch mode or via the I/O Designer Toolkit. Each interface subsystem is configured through its own ACE configuration GUI in the I/O Designer Toolkit, which outputs all required files and data for integration of the I/O ring configuration, including output files for simulation, synthesis, place and route, and bitstream generation.

A clock I/O can be configured as an input or output using the Clock I/O Bank configuration editor. The following can be specified:

- The I/O standard
- The signal type
- The port direction
- The input frequency

- The pull type
- The hysteresis type
- The ODT mode
- The receiver target impedance
- The placement of the clock input on a specific pad

An example of ACE configuration GUI of the I/O Designer Toolkit is shown here

> **Note**
>
> The configurable fields shown below may differ depending on the "Target Device" selected.

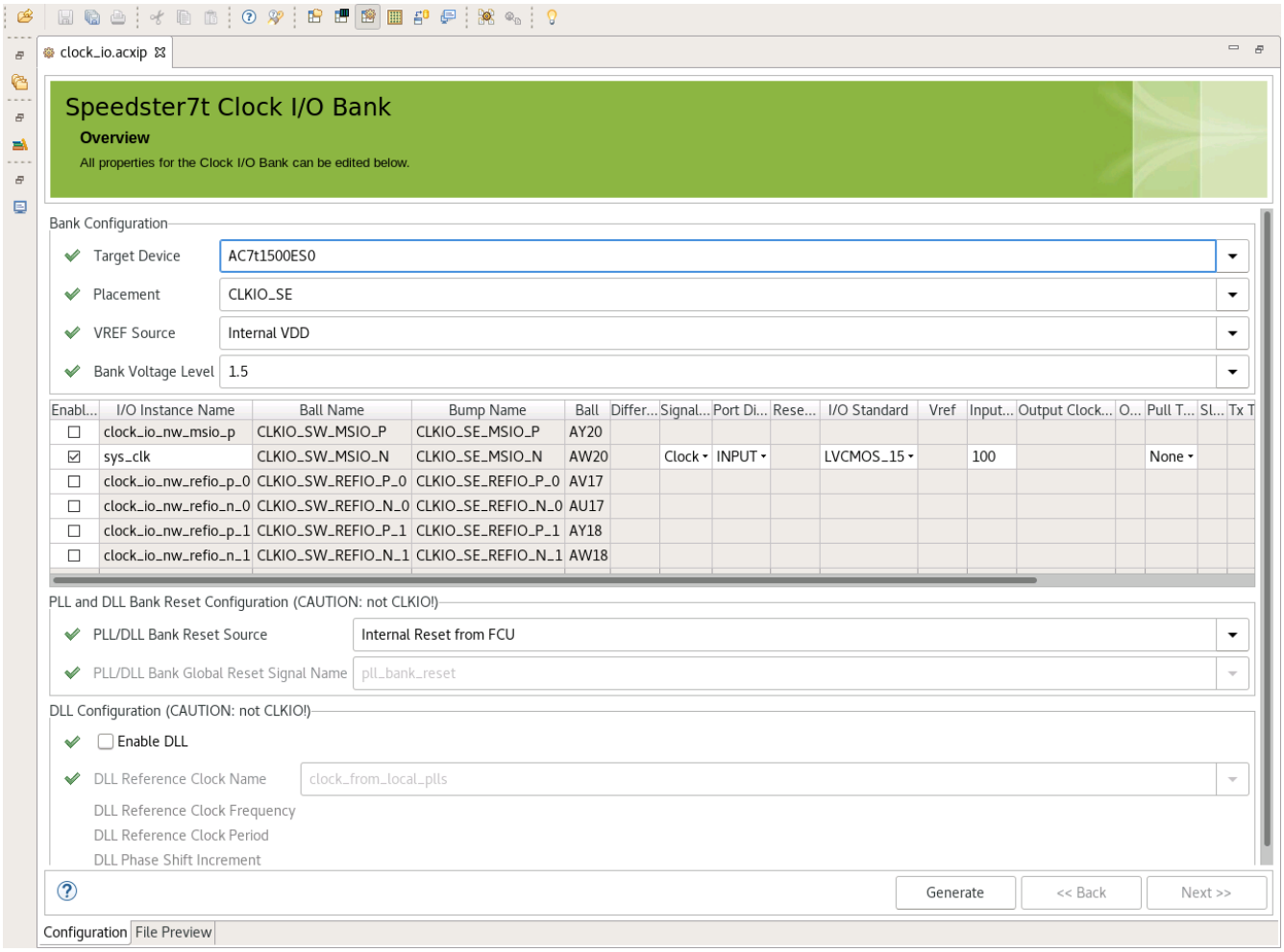


Figure 5 • Example Clock I/O Designer Toolkit I/O Bank Configuration Editor

Before selecting clock I/O banks and pins for a design, the following restrictions must be understood.

Enabling CLKIO as Inputs to the Interface Subsystems

If it is intended to route a clock or a reset to a particular interface subsystem, it is recommended to use clock I/O banks in specific corners as described in the table below. This recommendation is made not because inputs from specific clock I/Os can be used to drive particular subsystems, but because clocks to particular subsystems can only be routed from PLLs in specific corners. It is possible to drive a PLL in a corner with a clock input from another corner by using PLL clock outputs from one corner as the reference clock of a PLL in another corner, but it is up to the designer to decide whether that configuration is necessary for their design.

Table 1 • Recommended Clock I/O Banks for the Different Interface Subsystems

Interface Subsystem	Recommended Clock I/O Bank
PCIe 0/1	CLKIO_NE/NW
Ethernet 0/1	CLKIO_NE/NW
GDDR6 0/1/2/3	CLKIO_NW/SW
GDDR6 4/5/6/7	CLKIO_NE/SE
DDR4/5	CLKIO_SE/SW

Enabling a CLKIO as a Global Reset Input to the Core Fabric

REFIOs and MSIOs can be used as a reset input to distribute an asynchronous global reset over the clock network. To enable this feature:

1. Enable `REFIO_*_*` or `MSIO_*` pins in the Clock I/O Bank configuration editor.
2. Set the **"**Port Direction" to "INPUT" and "Signal Type" to "Reset".
3. Check the **Reset Connects to Core** checkbox.

> **Note**
>
> In AC7t700/800 devices, a total of 4 resets can be supported using any of the 12 REFIO and MSIO pins.
>
> In AC7t1400/1500 Devices, a total of 2 resets can be supported, only NE/NW `REFIO_P_0` pins can be used as reset inputs.

Routing a PLL Clock Output to a Clock I/O Pin

To route the clock output of a PLL to a clock I/O pin, choose a pin in the same corner as the source PLL. Also, the source PLL must be placed according to the selected destination clock I/O pin as described in the table below.

Table 2 • Corner Available Clock I/O Destination Determined by Source PLL Placement in the Same Corner

Corner Source PLL Placement	Allowed Destination Clock I/O Pin
PLL_0	REFIO_P_0
PLL_1	REFIO_P_1
PLL_2	MSIO_P
PLL_3	MSIO_N

Note

In AC7t1400/1500 devices, it is possible to enable 4 clock outputs for any PLL but only PLL clock outputs 0–2 can route to a clock output pin. ACE does not allow PLL clock output 3 to be routed to a clock output pin.

There is no such limitation for AC7t700/800 devices.

For example, to route a clock output from PLL_SE_0 to a clock I/O pin, according to the above table, this PLL clock output must be routed to the CLKIO_SE_REFIO_P_0 pin as follows:

1. Enable the CLKIO_SE_REFIO_P_0 pin in the Clock I/O configuration editor
2. Set the "Port Direction" as "Output".
3. select the appropriate PLL clock output using the pull-down menu under "Output Clock Source".

PLLs

A group of four general purpose PLLs are placed at the corners of the Speedster7t FPGA. For AC7t1400/1500, each of the four corners is populated with a 4-PLL group, giving a total of 16 PLLs. For AC7t700/800, only the northwest (NW) and southwest (SW) corners are populated, giving a total of 8 PLLs.

They are fractional-N divide and spread-spectrum PLLs, supporting a wide range of frequencies with excellent jitter performance. The general-purpose PLLs can be used to drive low-skew, high-speed clocks to nearby I/O, the global clock network, and interface clocks in the FPGA fabric.

Available features in these low-power, low-jitter PLLs are:

- Programmable PLL with fractional-N divide and spread-spectrum clock generation
- Wide range of output frequencies supported: 7.5 MHz to 2 GHz
- Reference clock from dedicated clock I/O, adjacent PLLs (for cascading PLLs), as well as PLLs from other device corners
- Up to four output clocks
- Reference clock and output clock dividers
- Inner and outer bypass paths
- Output duty cycle 50%

Table 3 • PLL Details

Parameter	Min	Max	Units
Reference frequency	5	600	MHz
Output frequency	7.5	2000	MHz
Maximum long-term jitter	±1% divided reference clock		

Each PLL can generate up to four output clocks. Each output can route to the global clock network for the FPGA fabric, to interface subsystems, to nearby GPIO, or to a PLL in a different corner of the device. By cascading PLLs, a design can route a high-quality, high-speed clock around to the entire Speedster7t FPGA.

Configuring PLLs

To add a new configuration for a PLL:

1. Switch to the IP configuration perspective in an existing ACE project.
2. Create a clock input using the Clock I/O Bank configuration editor.

When an input clock is available, the PLL can be configured using either the PLL or the Advanced PLL configuration editor in the I/O Designer toolkit. Both of these editors allow specification of the following:

- Which PLL to use
- The input reference clock
- The total number of output clocks to produce

Each PLL allows for up to four output clocks. The PLL configuration editor allows specifying:

- The desired output frequency for the first output
- The output clock name
- Whether to expose the signal to the FPGA fabric

For each additional output clock, the desired clock divider value is specified with respect to:

- The produced VCO frequency
- The output clock name
- Whether to expose the clock for use in the FPGA fabric
- Whether to expose the PLL lock signal to the FPGA fabric

Unlike the PLL configuration editor, the Advanced PLL configuration editor exhibits these differences:

- Does not automatically calculate the divider values for PLL clock output 0
- Allows the configuration of the optional output synthesizer (OS)
- Allows the configuration of the output divider (OD) for all the PLL clock outputs
- Allows enabling the spread spectrum functionality
- Allows enabling the external or internal PLL bypass

> **Note**
>
> In the PLL and Advanced PLL configuration editors, the PLL clock output 3 always goes through the OS unless using either the ring oscillator or the outer bypass path for PLL clock output 3.

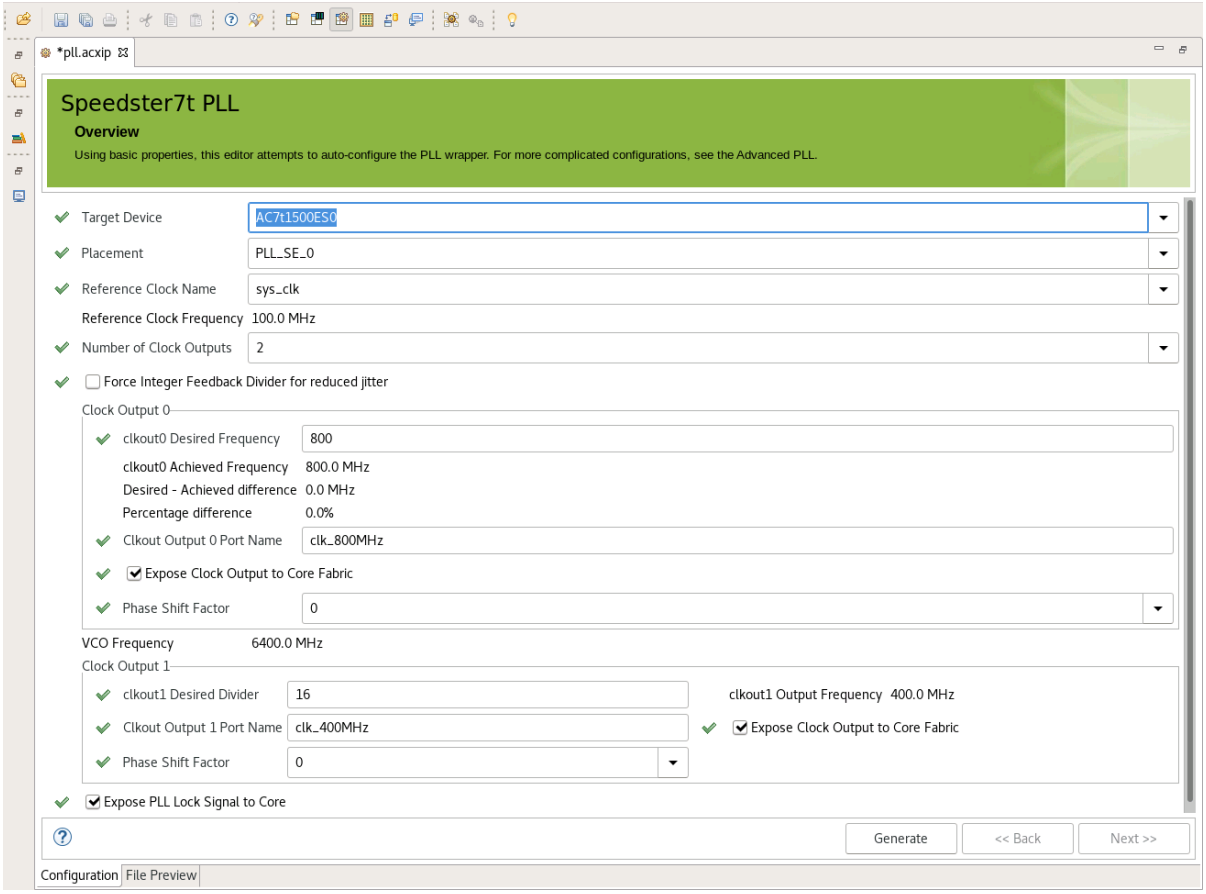


Figure 6 • PLL Configuration Editor in the I/O Designer Toolkit

In the above example, a 100 MHz input reference clock called `sys_clk` is chosen from a pull-down menu, as it is one of the input clocks available in the allowed frequency range and visible at this PLL placement site. The PLL is placed in the south-east corner of the device and uses PLL0 via a pull-down menu, as indicated by `PLL_SE_0`. This example shows two output clocks. The first clock is set to 800 MHz and is called `clk_800MHz`. The second output, `clk_400MHz`, uses a divide-by-16 output of the calculated 6400MHz VCO frequency to achieve 400MHz. Both clocks are exposed to the FPGA fabric, as indicated by checking the box **Expose Clock Output to Core Fabric**. The PLL lock signal is also exposed to the FPGA fabric, as indicated by checking the box **Expose PLL Lock Signal to Core** . As a result, the two PLL output clocks are routed on the global clock trunk and are available for use by logic inside the FPGA fabric. Alternatively, if a clock is only to be used by the NoC, or some other subsystem in the I/O ring, the clock output can be set to not be exposed to the FPGA fabric by leaving the box unchecked. In that case, the clock output is available for use as a clock that routes directly to another subsystem in the I/O ring and does not consume a clock resource inside the FPGA fabric.

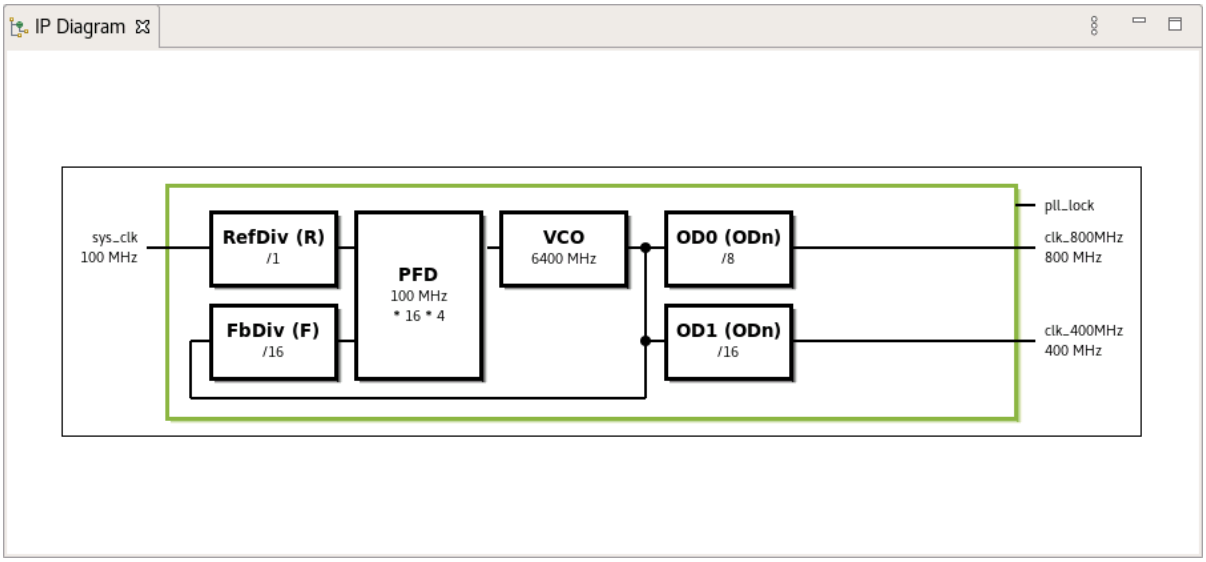


Figure 7 • IP Diagram of the Enabled PLL Configuration

The figure below shows the placement of the differential clock input at REFIO_0 and PLL0 located in the south-east corner of an AC7t1500 device, shown in green.

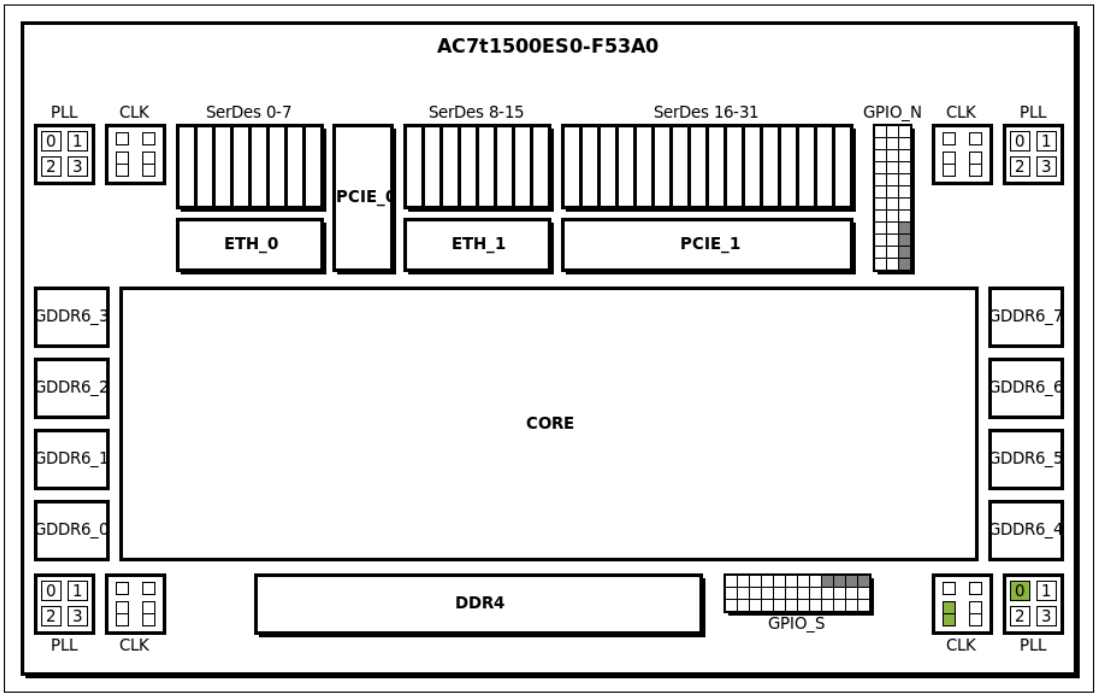


Figure 8 • Differential Clock and PLL Placed on Speedster AC7t1500

Enabling the Spread Spectrum Functionality

To enable the Spread Spectrum functionality:

1. Create a clock input (page 6).
2. Ensure that the value in the **Divided Reference Frequency** field (yellow highlight in the figure below) is between 5–7.5MHz.
3. Select the **Enable Spread Spectrum** box (green highlight) as shown below.

   The Advanced PLL configuration editor tracks whether the divided reference frequency value is in the allowed range by displaying a Yes/No value in the **Allows fractional fbdiv(F) and Spread Spectrum support** field (pink highlight below). The **Enable Spread Spectrum** checkbox is disabled if the divided reference frequency value is out of the allowed range.

   In this example, `clk_1000MHz`, is a spread spectrum clock.

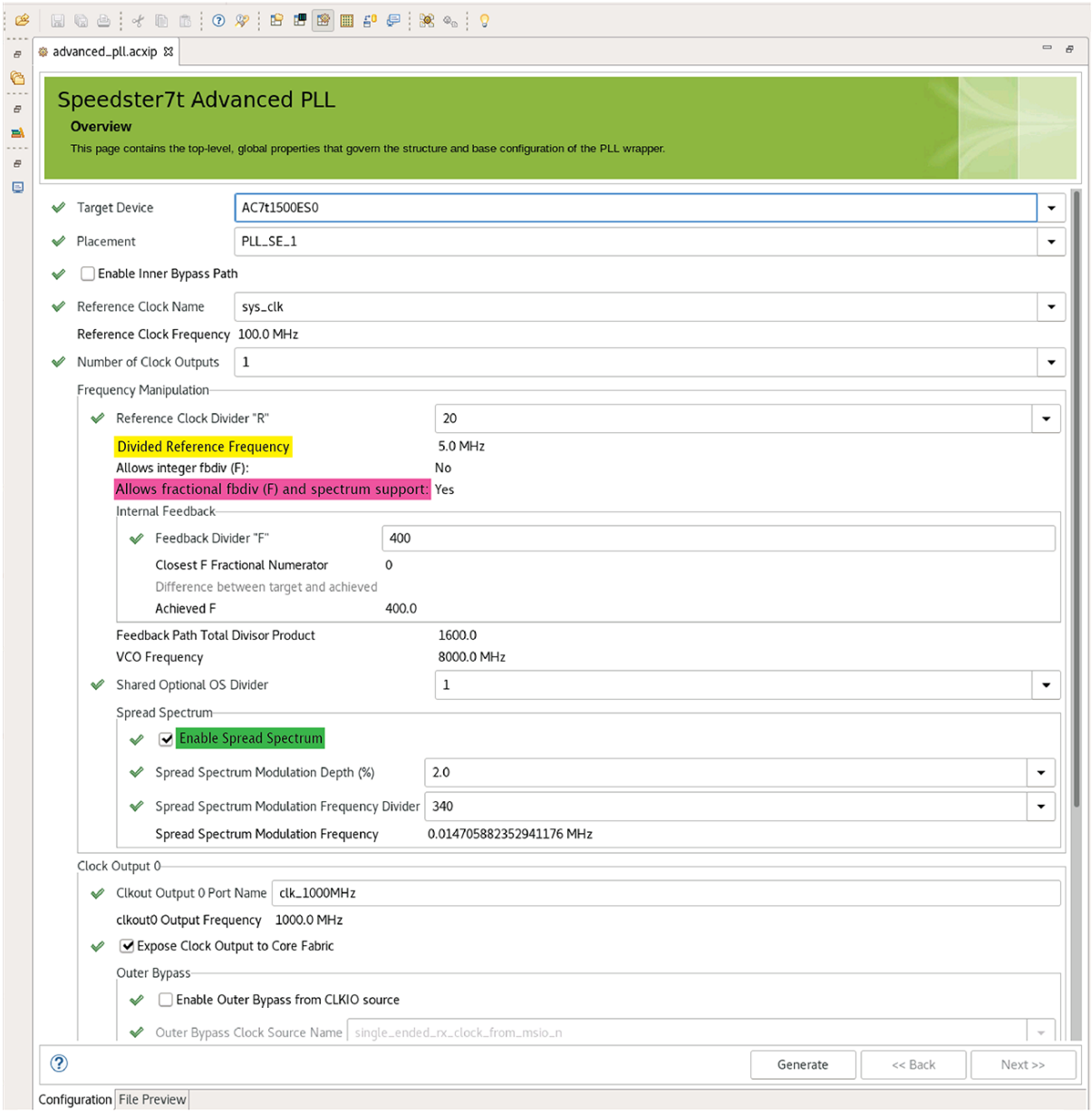


Figure 9 • I/O Designer Toolkit Advanced PLL Configuration Editor Example

Enabling the Outer PLL Bypass for a Single PLL Clock Output

A PLL clock output can be bypassed using the outer PLL bypass path. Any clock outputs can be overridden with their own outer bypass path, in any combination of outputs, including the extreme case of all four outputs being overridden with four (outer-) bypass paths. Unlike the inner PLL bypass (page 17) path, which overrides all the PLL

clock outputs, enabling the outer PLL bypass for a particular PLL clock output does not affect the functionality of the remaining outputs.

> **Note**
>
> It is possible to override any PLL clock output using the outer bypass path for that particular PLL clock output and it is also possible to override multiple PLL clock outputs simultaneously using their respective outer bypass paths.

If it is planned to enable the outer PLL bypass, keep in mind that only an input from a specific source clock I/O pin can be used to override a particular PLL clock output as described in the following table.

> **Warning!**
>
> The outer PLL bypass can only be enabled using the Advanced PLL configuration editor.

Table 4 • Available CLKIO Source to Override a PLL Clock Output, Depending on the PLL Site Placement When Using the Outer PLL Bypass

Source Clock I/O	Destination PLL Placement			
	PLL_[NE,NW,SW,SE]_0	PLL_[NE,NW,SW,SE]_1	PLL_[NE,NW,SW,SE]_2	PLL_[NE,NW,SW,SE]_3
MSIO_P	PLL Clock Output 2	PLL Clock Output 3 (1)	PLL Clock Output 0 (3)	PLL Clock Output 1
MSIO_N (2)	PLL Clock Output 3 (1)	PLL Clock Output 0 (3)	PLL Clock Output 1	PLL Clock Output 2
REFIO_0	PLL Clock Output 0 (3)	PLL Clock Output 1	PLL Clock Output 2	PLL Clock Output 3 (1)
REFIO_1	PLL Clock Output 1	PLL Clock Output 2	PLL Clock Output 3 (1)	PLL Clock Output 0 (3)

Table Notes

1. The Clock Output 3 signals can be overridden by the Ring Oscillator output.
2. The MSIO_N signals must be in single-ended mode. When in differential mode, MSIO_N is not a valid clock source.
3. This is the same source Clock I/O used by this PLL placement site for the inner bypass path.

In the following example, a 100MHz input reference clock from the South-East `MSIO_N` pin, `sys_clk`, is routed to `PLL_SE_0`. According to the above table, this input can only override PLL clock output 3 of `PLL_SE_0` via the outer bypass.

This configuration is enabled as follows:

1. Set the **Number of Clock Outputs** (green highlight below) to **4** which enables PLL clock output 3.

2. Check the **Enable Outer Bypass from CLKIO source** box (yellow highlight, second figure) in the section for PLL clock output 3.

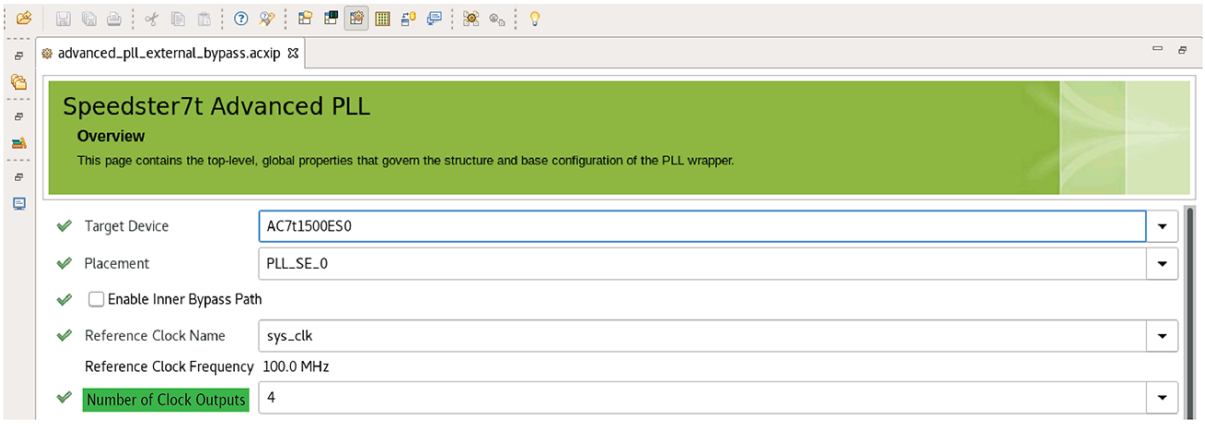


Figure 10 • Number of Clock Outputs Example

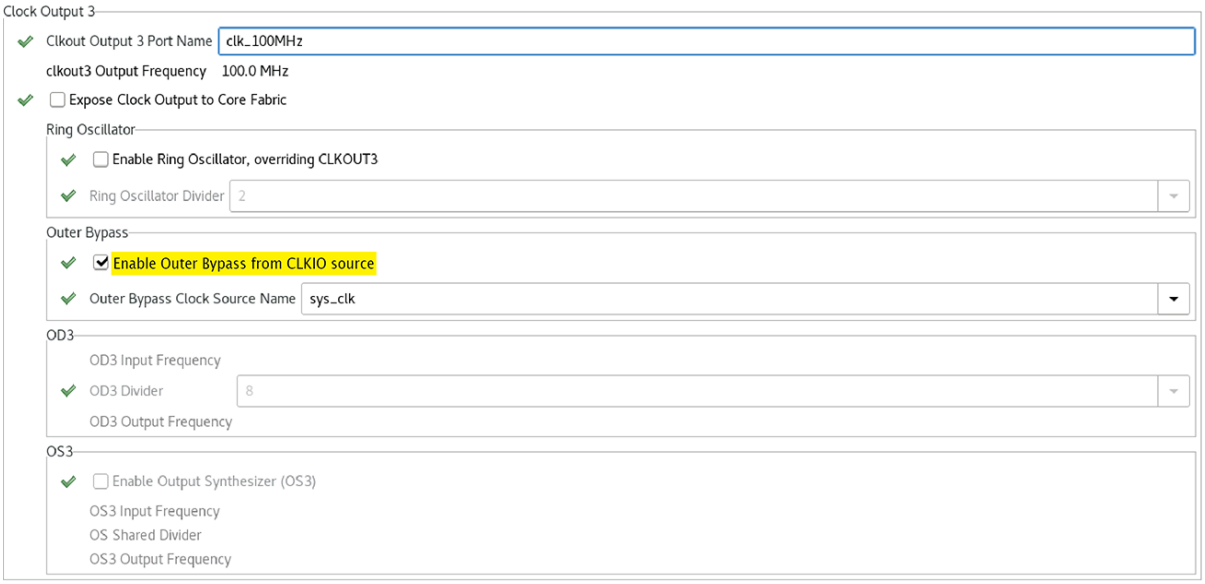


Figure 11 • Enable the Outer Bypass From CLKIO Source Example

The IP diagram below shows `sys_clk` overriding the PLL clock output 3 using the outer bypass path.

> **Note**
>
> In the enabled configuration, `sys_clk` (or any other clock), can still be used as the reference clock of the PLL. Unlike the inner PLL bypass path, which overrides all the PLL clock outputs, enabling the outer PLL bypass for a particular PLL clock output does not affect the functionality of the remaining outputs.

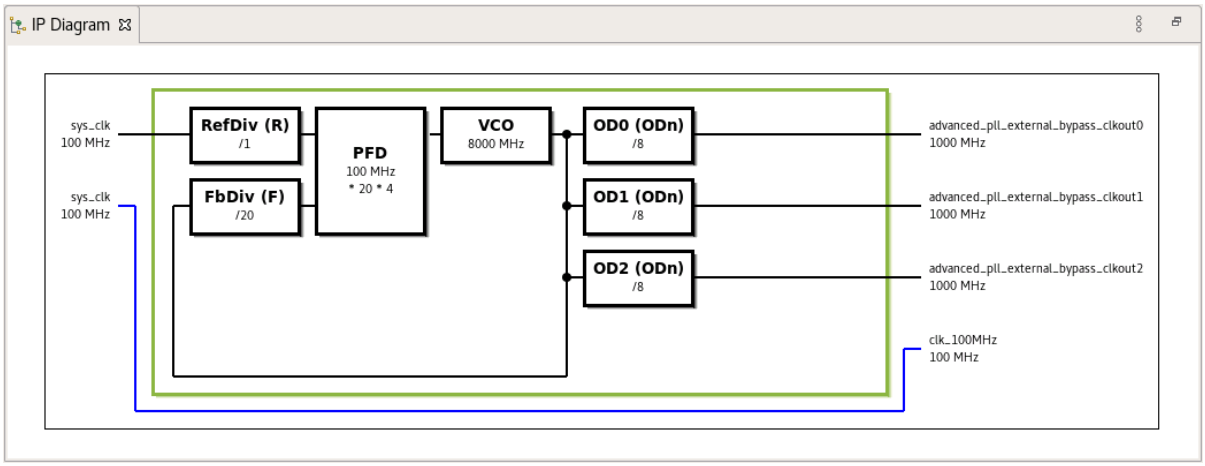


Figure 12 • A Diagram of the Enabled Outer PLL Bypass Configuration (Highlighted in Blue)

Enabling the Inner PLL Bypass

The PLL can be bypassed using the inner PLL bypass path. This path overrides all of the PLL clock outputs and restricts them to using the following resources:

- The optional output synthesizer (OS) divider
- The ring oscillator
- The outer PLL bypass paths for all of the PLL clock outputs

To enable the inner PLL bypass, choose the destination PLL placement according to the selected source clock I/O pin as shown in the table below.

> **Warning!**
>
> The inner PLL bypass can only be enabled using the Advanced PLL configuration editor.

Table 5 • Available CLKIO Source Depending on the Destination PLL Site Placement Using Inner PLL Bypass

Destination PLL Placement	Source Clock I/O Pin
PLL_[NE,NW,SW,SE]_0	REFIO_0
PLL_[NE,NW,SW,SE]_1	MSIO_N
PLL_[NE,NW,SW,SE]_2	MSIO_P
PLL_[NE,NW,SW,SE]_3	REFIO_1

In the following example, a 100MHz input reference clock from the South-East `MSIO_N` pin, `sys_clk`, is required to override a PLL using the inner bypass. From the above table, this input can only be routed around `PLL_SE_1` via the internal bypass path. This configuration is enabled by checking the **Enable Inner Bypass Path** box (yellow highlight, below).

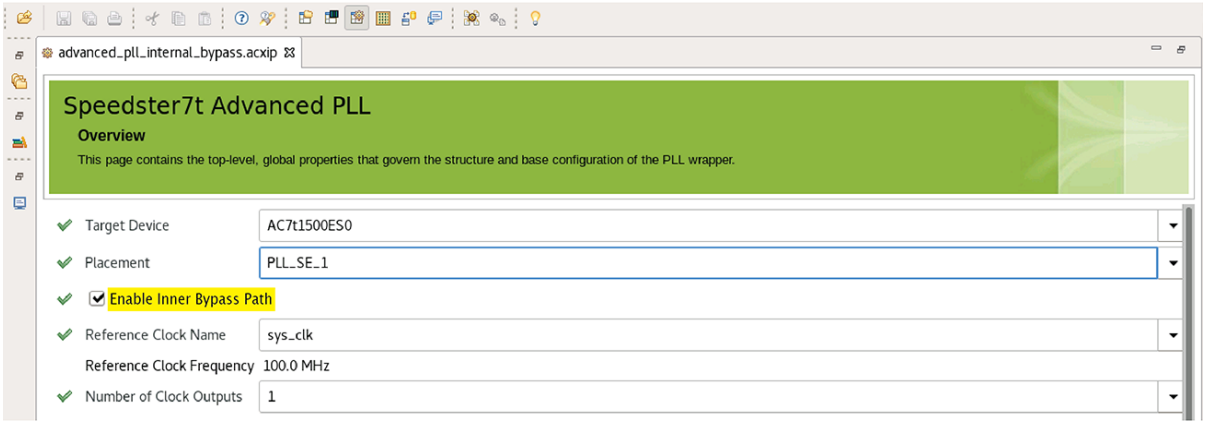


Figure 13 • Enable the Inner PLL Bypass Example

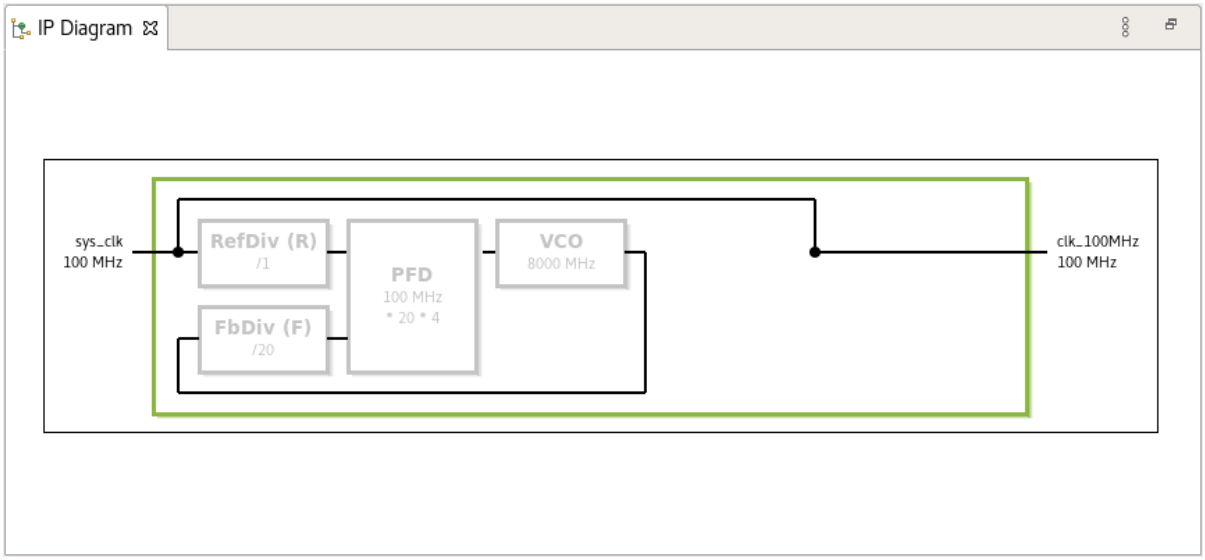


Figure 14 • Diagram of the Enabled Inner PLL Bypass Configuration

Dedicated PLLs

Along with the general-purpose PLLs located in the corners of the FPGA, there are several dedicated PLLs for use by the interface subsystems surrounding the fabric portion of the FPGA. There are twelve dedicated PLLs to provide clocks to the external network on chip (NoC). The NoC expects a single 200 MHz input reference clock, and the dedicated PLLs automatically generate the configured clock, up to 2 GHz, to the NoC. The figure below shows the

NoC configuration GUI. Choose the appropriate 200 MHz clock from the pull-down menu to assign to the input of the NoC PLLs.

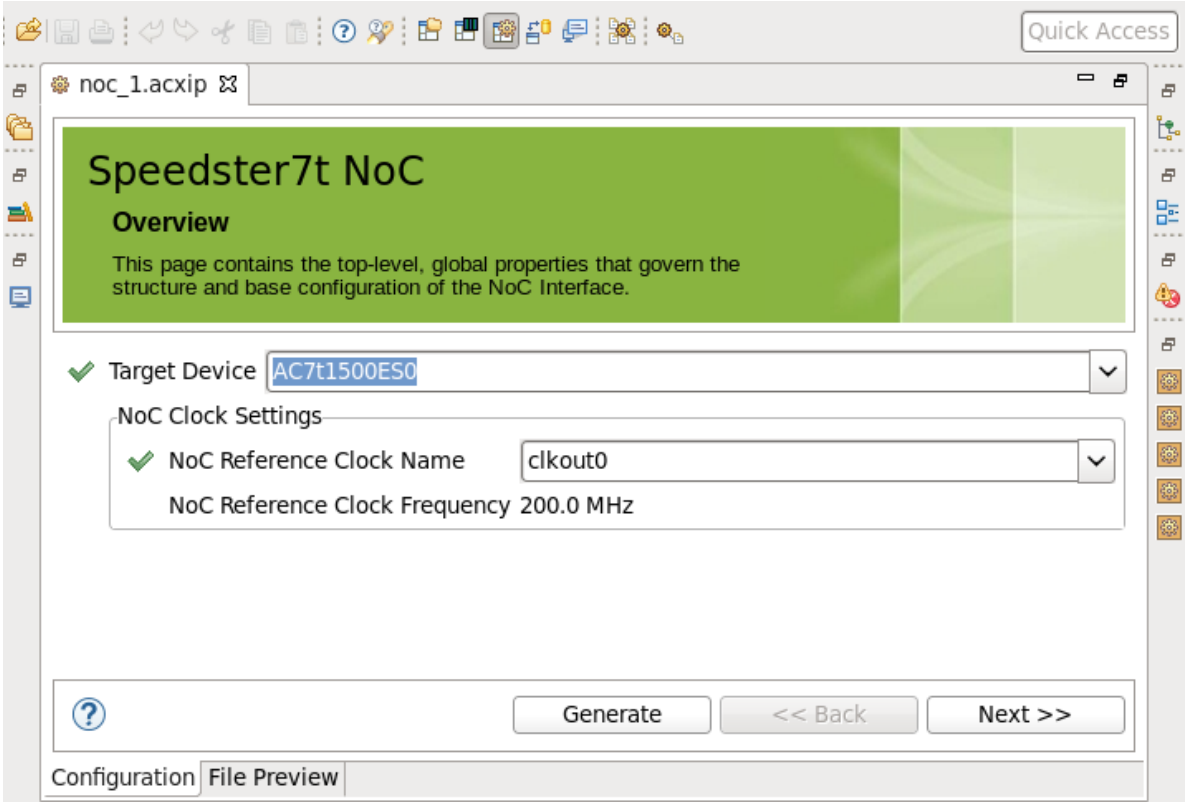


Figure 15 • NoC PLL Input Clock Configuration Example

Additionally, the GDDR6 and DDR4 PHYs also have built-in PLLs which can be used for protocol-specific clocking. These dedicated PLLs cannot route to the fabric, but are used with their specific interface subsystem. They are fed by outputs from the general purpose user PLLs.

The specific clock and frequencies are specified in the corresponding interface subsystem configuration GUI in the I/O Designer Toolkit. The figure below shows the PLL configuration of a GDDR6 channel. The first pull-down menu chooses a clock output as the reference clock for the GDDR6 dedicated PLL. The second pull-down menu shows possible clocks that can be used for the direct connect interface (DCI) of the GDDR6 subsystem.

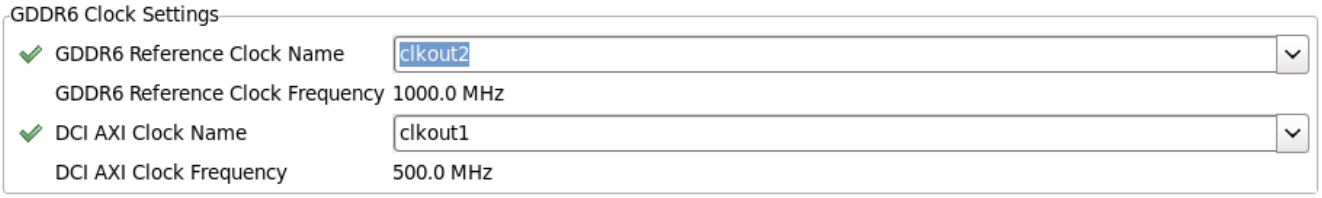


Figure 16 • Dedicated PLL GDDR6 Configuration Example

Each SerDes quad has a PLL which derives the receive clocks. These PLLs can be used to drive the receive and transmit logic in the SerDes, PCIe, and Ethernet. They can also drive the global clock trunk and minitrunk clocks in the fabric, enabling logic in the fabric to easily communicate with the SerDes-based interfaces.

> **Note**
>
> Currently, ACE supports SerDes PLL settings only though the PCIe and Ethernet configuration GUIs.

For more details on the required clock frequencies for each interface subsystem, refer to the following documents:

- *Speedster7t GDDR6 User Guide* (UG091)[1]
- *Speedster7t DDR User Guide* (UG096)[2]
- *Speedster7t Ethernet User Guide* (UG097)[3]
- *Speedster7t PCIe User Guide* (UG098, Achronix support account required for access)

On-Chip Oscillators

A group of four oscillators are placed at the corners of the Speedster7t FPGA. For AC7t1400/1500, each of the four corners is populated with a 4-oscillator group, giving a total of 16 oscillators. For AC7t700/800, only the northwest (NW) and southwest (SW) corners are populated, giving a total of 8 oscillators.

The oscillators are 4 GHz free-running clocks. The frequency can vary from device to device, as well as on the same device due to runtime temperature changes. They include statically configured clock dividers for common values of /2, /4, /8, /16, /32, /64, and /128.

> **Warning!**
>
> It is not recommended to use the on-chip oscillators for timing-critical production systems as the frequencies of the output clocks from these oscillators can be unpredictable.

Configuring On-Chip Oscillators

On-chip oscillators can only be accessed through clock output 3 of any given PLL. To enable the on-chip oscillator, follow these steps:

1. Create a clock input and choose a PLL.
2. Set the **Number of Clock Outputs** to **4** as shown below.

1 https://www.achronix.com/documentation/speedster7t-gddr6-user-guide-ug091
2 https://www.achronix.com/documentation/speedster7t-ddr-user-guide-ug096
3 https://www.achronix.com/documentation/speedster7t-ethernet-user-guide-ug097

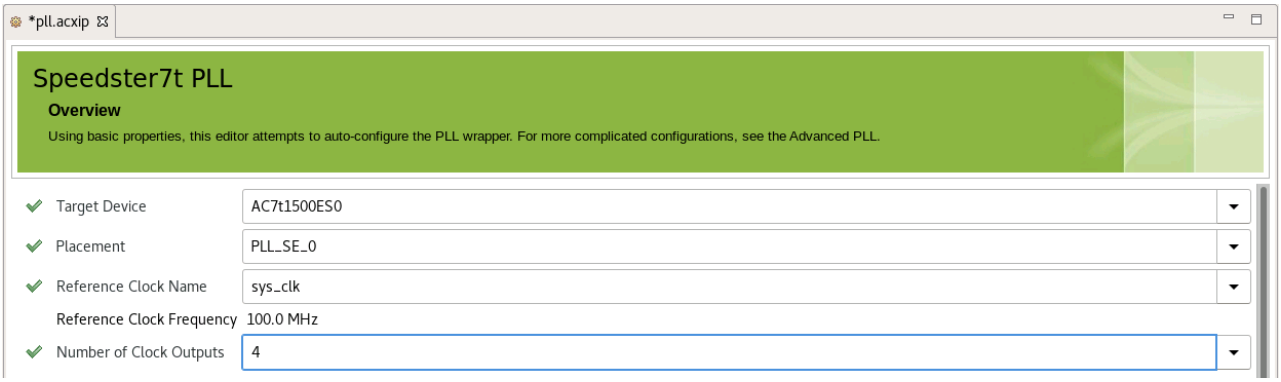


Figure 17 • Number of Clock Outputs Example

3. Under **Clock Output 3**, check the **Enable Ring Oscillator, overriding CLKOUT3** checkbox.
4. Set the desired divider value from the **Ring Oscillator Divider** drop-down menu as shown below.

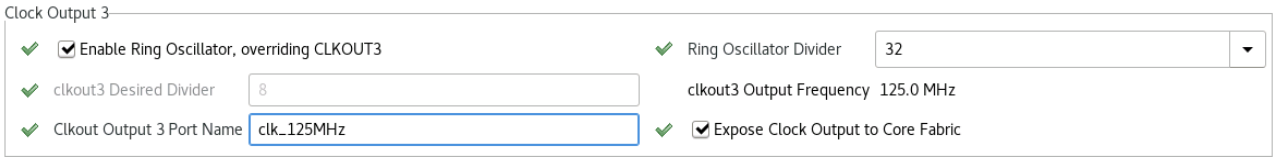


Figure 18 • Enable On-Chip Oscillator Example

Chapter 4 : Speedster7t Global Core Clock Network

The global core clock network is a balanced and low-skew H-tree that enables clock distribution to all parts of the Speedster7t FPGA fabric. Clock signals coming in from the top and bottom are routed through clock hubs (H0) and aggregated at the center of the device (clock hub H1). These are then provided to all clock regions on both the West and East sides.

A total of 48 global clock signals are generated in the clock hub H1. These signals are then distributed to all clock regions. Every clock region supports up to a maximum of 16 clocks, and it is able to select, among other sources, any of the 48 available global clocks.

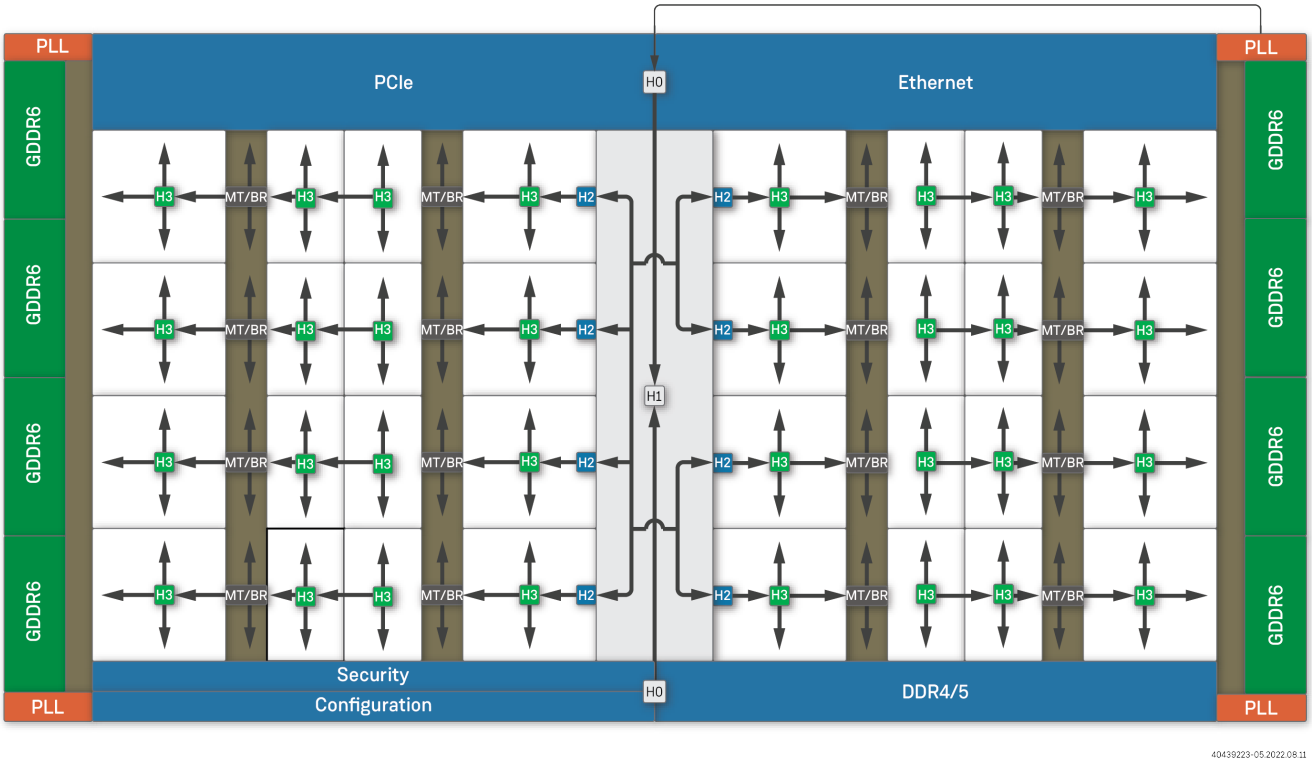


Figure 19 • Global Core Clock Network

Clock Hub 0 (H0)

There are two clock hubs (H0) in the FPGA boundary ring, one at the top center of the device and one at the bottom center. Each one of these H0 hubs takes the *N* clock signals arriving from the clock pins (*N* varies depending on the Speedster7t device) and outputs 32 clock signals to the central clock trunk, driving the clock hub (H1) at the center of the core. The H0 is an *N* × 32 fully general crossbar driving a reset sync block which then provides clocks to the

balanced clock trunk. The reset sync block is used to ensure that clocks are enabled synchronously with respect to the reset coming from the FPGA configuration unit (FCU).

The global fabric reset that arrives from the FCU is distributed asynchronously throughout the entire fabric via a reset distribution tree. This reset keeps the fabric in reset during configuration and is de-asserted some time before entry into user mode. Additionally, `cReset` is used to gate the clocks so that they do not toggle during configuration. When this reset is de-asserted, these clocks are glitchlessly released and start driving into the fabric.

As shown in the figure below, for *N* clock inputs, the two most significant bits (*N-1* and *N-2*) can serve as user resets or as clocks during normal user mode. The remaining *N-2* bits serve as clock inputs. Both Bit *N-1* and bit *N-2* can be used as either asynchronous assert, synchronous de-assert, or purely asynchronous resets. In the pre-sync block, both of these resets can be synchronized with `clk_out[0]` from the reset sync block. The fully-general crossbar takes the two resets from the pre-sync block, `user_reset[1]` and `user_reset[0]`, on bits *N-1* and *N-2* respectively, along with the remaining clock inputs (*N-2* bits), and routes them to any of the thirty-two output bits.

Bits *N-1* and *N-2* can route to any bit except the least significant bit, bit[0], when being used as user resets. In the reset sync block, the MUX before each output selects between the gated clock from the output of the crossbar (specifically the output of the integrated clock gate, or ICG) or one of the user resets, depending on which of the thirty-two bits the user reset signal was routed.

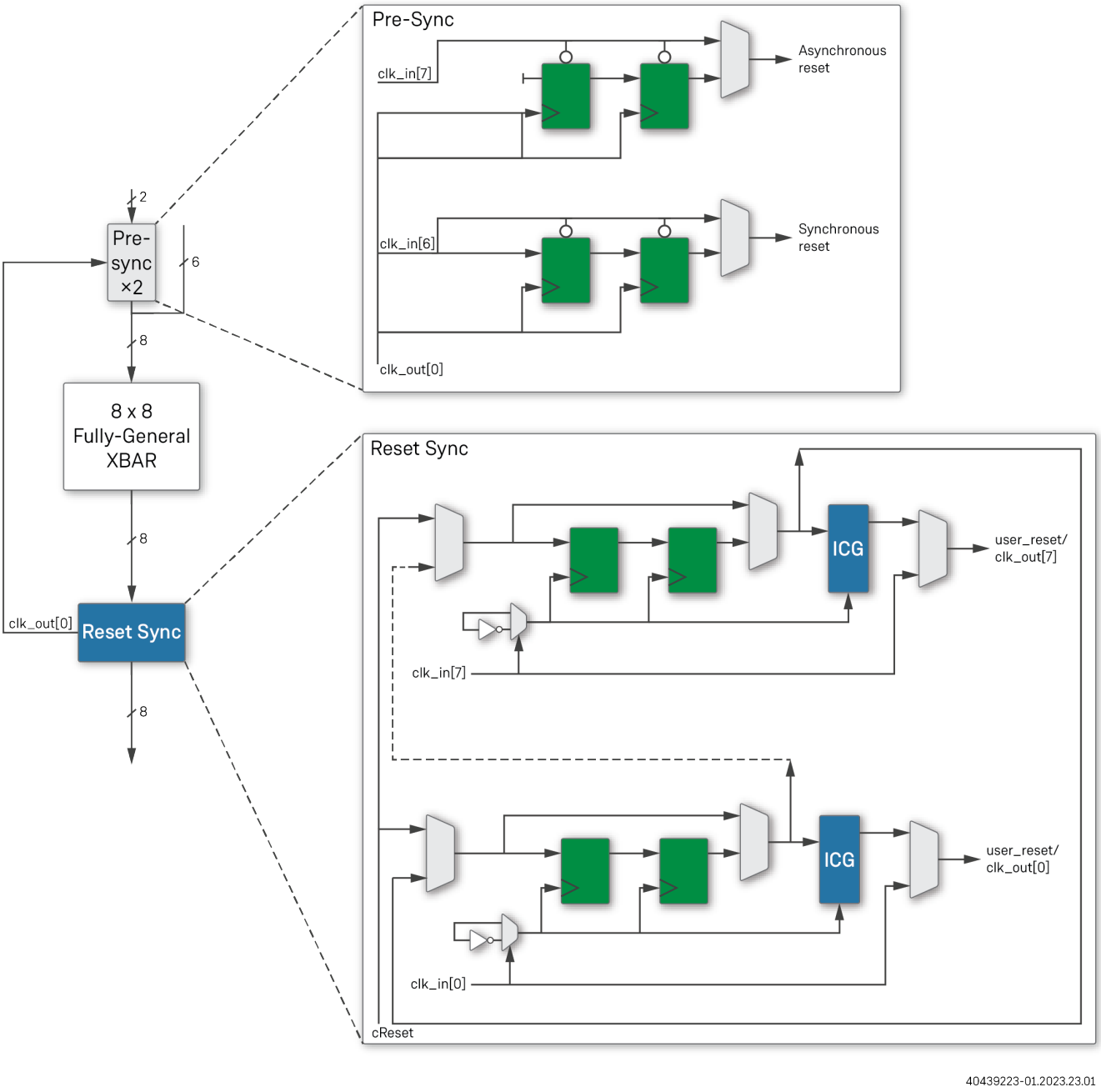


Figure 20 • Clock Hub (H0)

Figure Notes

1. The width of the crossbar depends upon the Speedster7t device.

Clock Hub 1 (H1)

The clock hub (H1) in the center of the device collects the two sets of 32 clocks coming from the top and bottom clock hubs (H0), as well as 16 data signals from the data interconnect to generate the 48 global clocks. The global clock bus travels up and down the length of the clock trunk to drive clock hubs (H2) at the root of each set of the clock regions (see the figure below). The 16 data signals coming from the data interconnect provides a path for bringing a signal generated within the fabric onto the global clock network. This arrangement allows for a total of 16 data signals and 64 clocks to route to 48 total clocks on the global clock network.

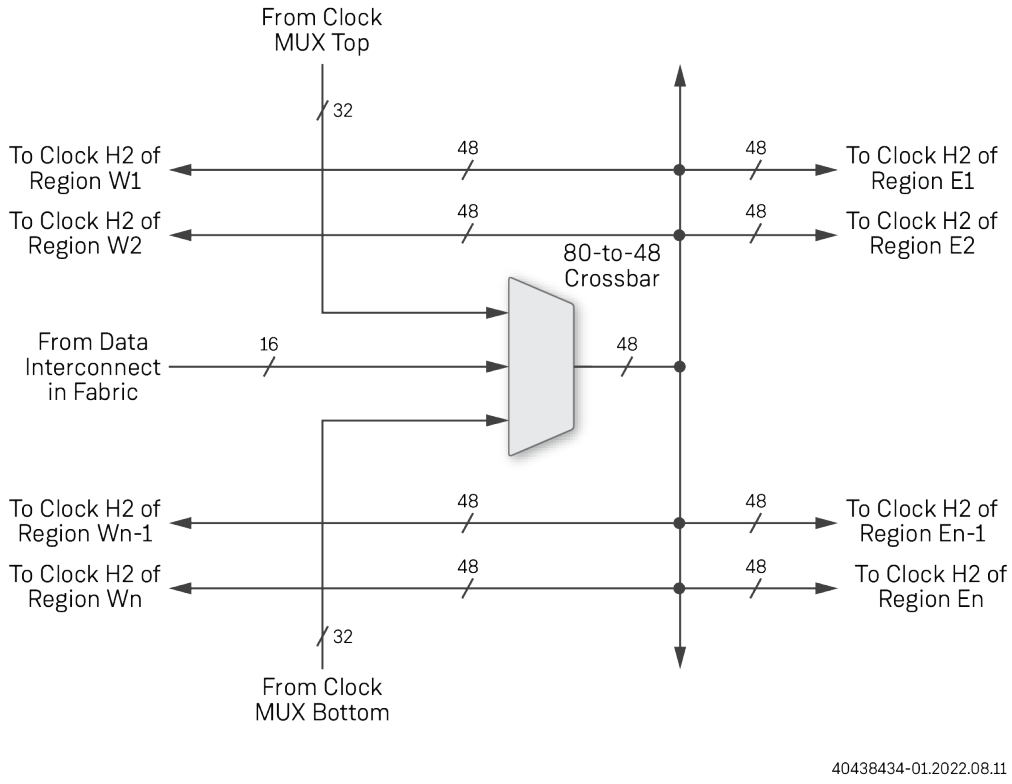


Figure 21 • Detailed View of the Clock Hub 1

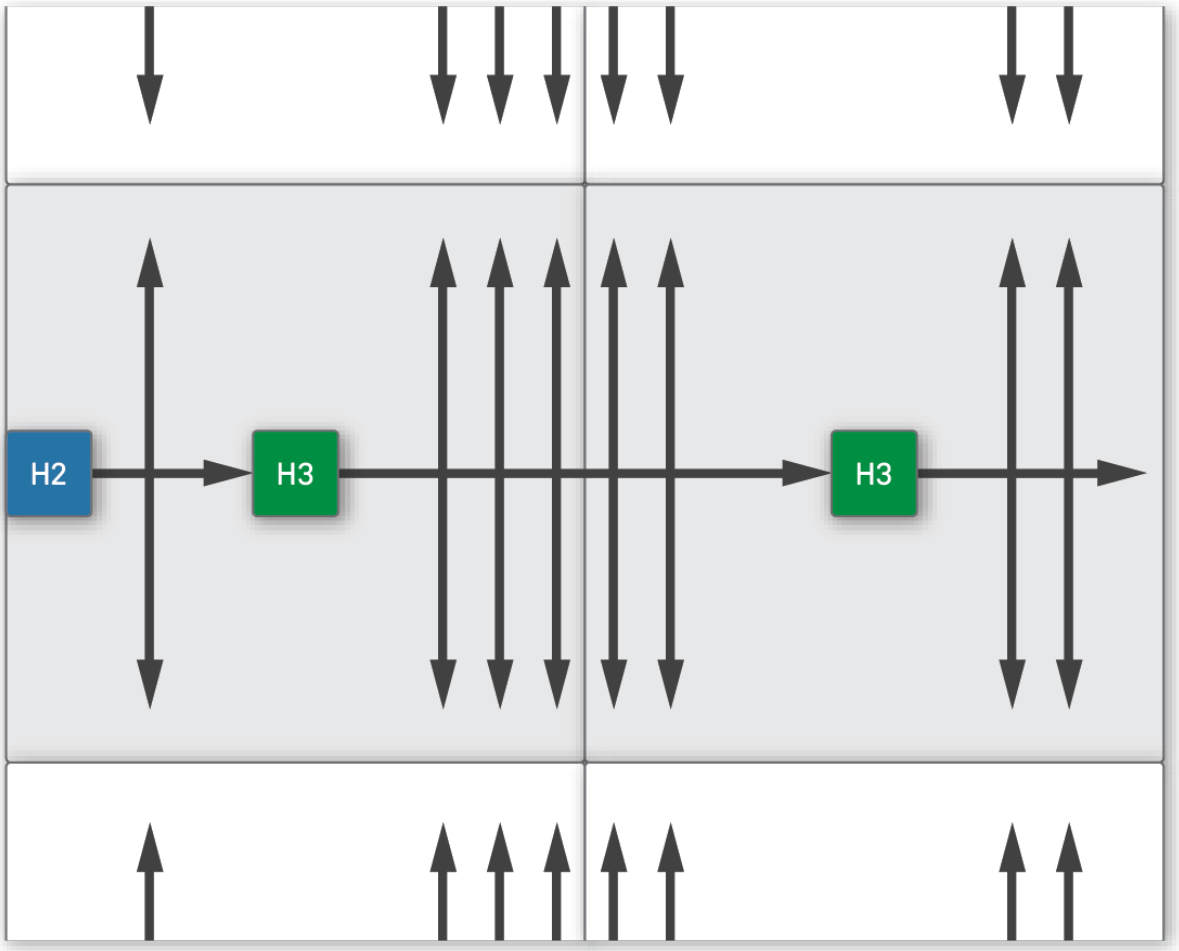
Clock Hub 2 (H2)

Clock regions in Speedster7t FPGAs have fixed heights, but the width, and consequently, number of IP columns is variable. This means that both the size and number of clock regions vary based on the device in the Speedster7t FPGA family. There are an equal number of clock regions on the left and right halves of the core. At the root of each half row of clock regions is the clock Hub 2 (H2), fed with 48 global clocks from H1, plus 16 data signals from the data interconnect to feed the clock network of a half row of clock regions.

All of these inputs are fed into a clock hub 2 (H2) which performs an appropriate clock selection and outputs 16 clocks which then fan out to all the clock hub 3s (H3s) at the root of every clock region (see the figure below).

Note

While 16 signals from the data interconnect feed into the H2, only 4 signals from the data interconnect can drive out of the H2 on the regional clock network.



3703270-04.2022.08.11

Figure 22 • Detailed View of a Clock Region

Each H2 has two clock crossbars. Each partial crossbar takes in 50 signals total, 48 from the incoming global clock bus, plus 2 signals from data interconnect, and drives out an 8-bit clock bus. These 8-bit buses are then aggregated to form the 16-bit clock output bus which is provided to all of the H3 clock hubs in the row. The result is a total of 48 incoming global clocks plus 4 incoming data signals to drive a 16-bit clock bus to the clock region.

Each of the 8 clocks from the crossbar can optionally be divided. The clock division and gating logic is controlled by signals coming in from the data interconnect bus (see the figure below). As stated previously, each H2 receives 32 inputs from data interconnect (16 to each partial crossbar shown with dashed lines). While only a total of four of

these signals can drive out on the clock network (2 for each partial crossbar), the full 32 (16 for each partial crossbar) can be used as control logic to the clock division, clock gating, and clock switching logic.

A summary of the features available in the clock hub 2 (H2) are as follows:

- Clock divider, with four (static) divide-by settings: 2, 4, 6, 8. There is an option to not use the clock divider if the clock simply needs to be propagated.
- Dynamic clock gate, allowing real-time clock gating (for power management).
- Glitchless clock switch, allowing dynamic muxing between different clock sources for that particular clock region.

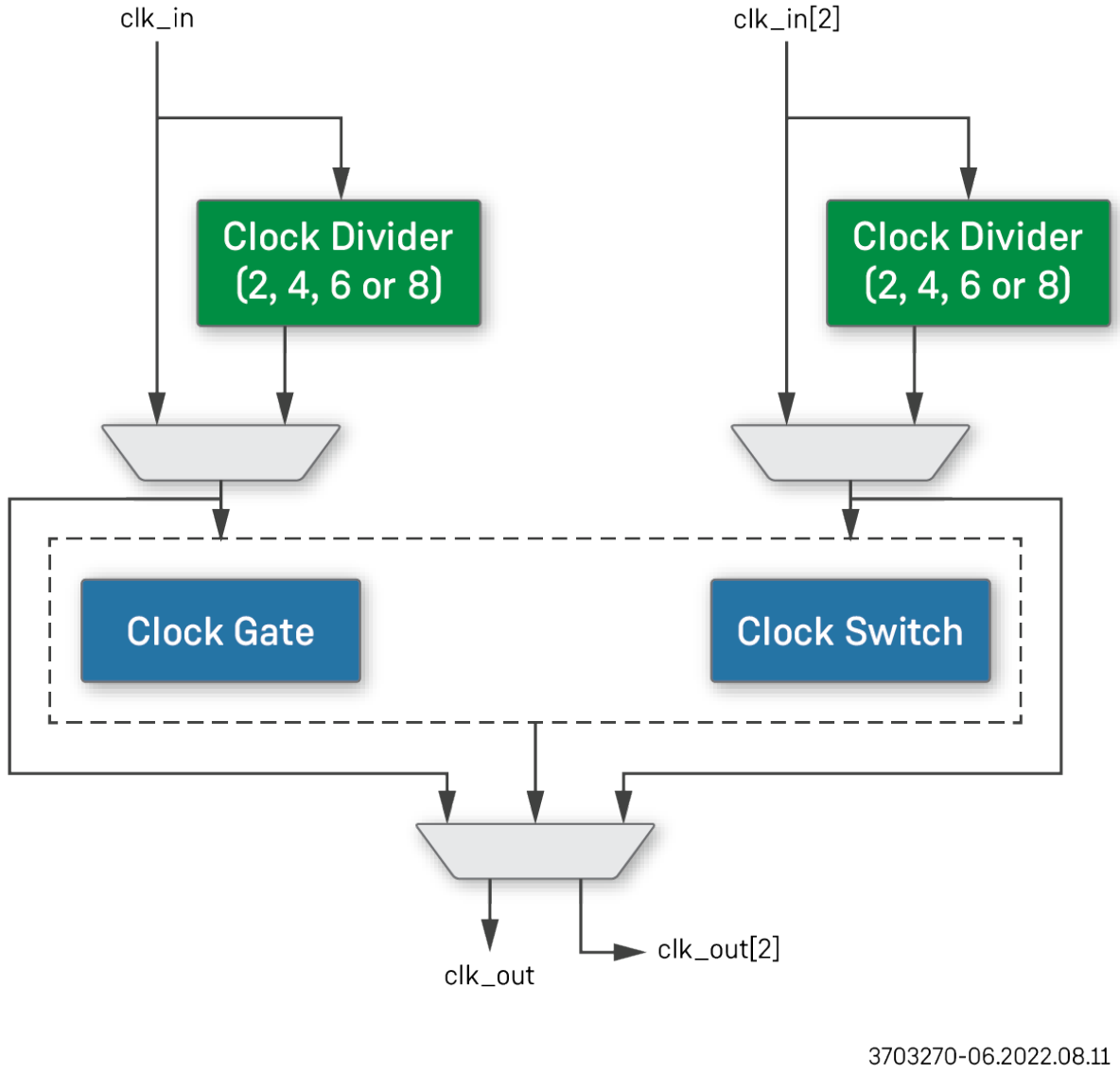


Figure 23 • Detailed View of Clock Hub 2

The following figure provides a more in-depth look at the three features available in the H2 clock hub. Clock division logic is controlled by static configuration memory bits, again with a static mux selection bit to determine whether or not the clock should be propagated as-is or divided down. The clock gate relies on an enable input to the logic, while the glitchless clock switch module uses select/deselect logic to select between two different clock inputs. The divider and gating or switching logic can also be cascaded as shown. This figure is a conceptual view of the circuitry for one instance only — these functions exist for all of the clock inputs.

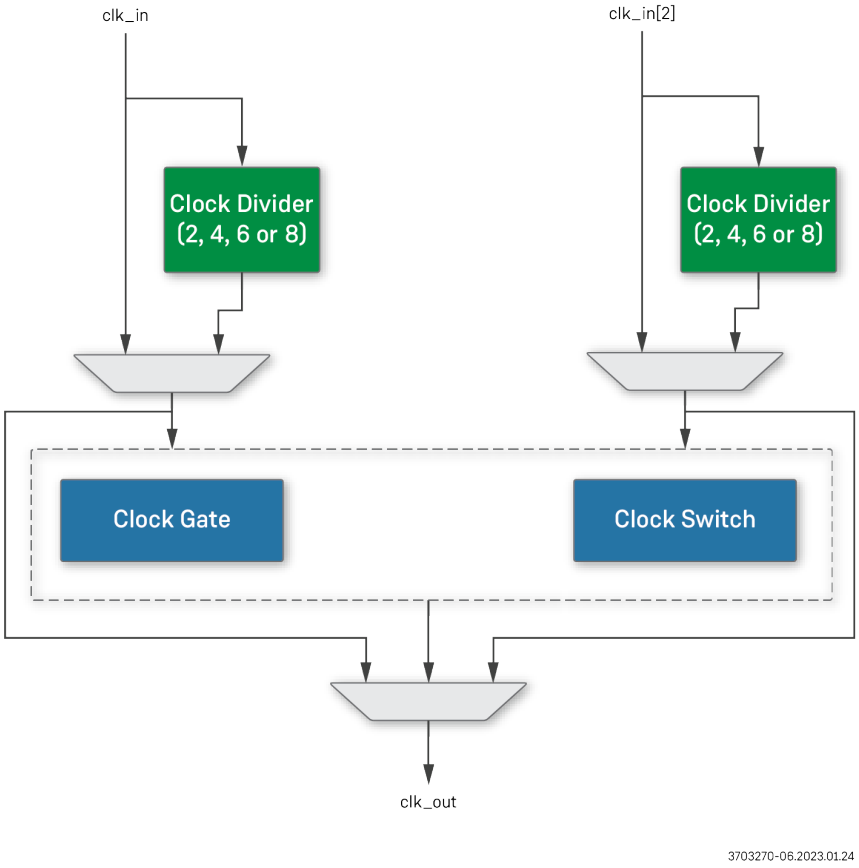


Figure 24 • Internals of Clock Division, Gating and Switching Circuitry

Clock Hub 3 (H3)

At the center of each clock region is a clock hub 3 (H3). The H3 drives 16 horizontal clocks from the trunk, 12 horizontal clocks towards the trunk, and drives out 12 clocks to the north and south. The H3 can select from any of the clocks to drive all the clock stems within the clock region, and north and south to the adjacent H2s.

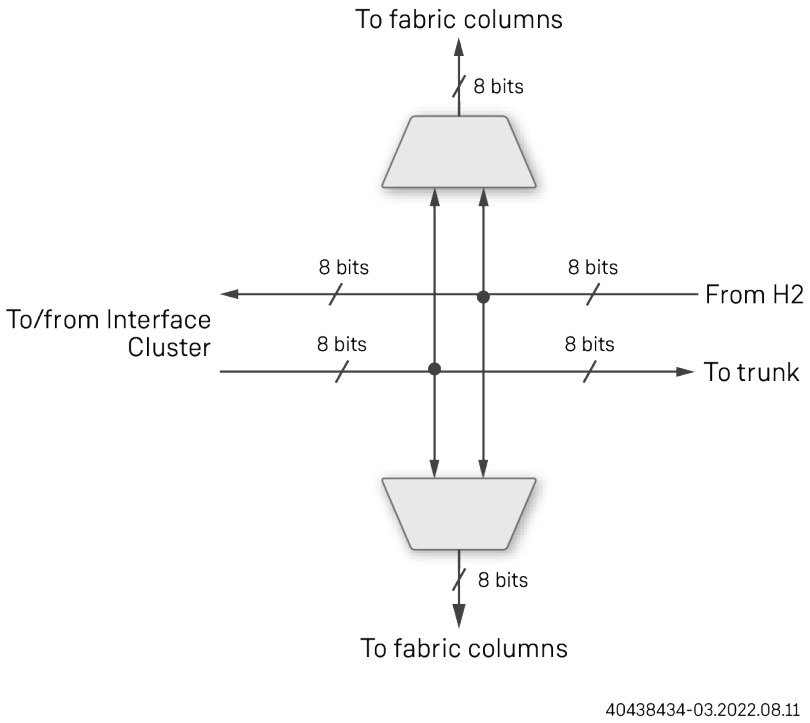


Figure 25 • Detailed View of Hub 3 on the West Side of the Device

Minitrunk and Branch Clock Hub (MT/BR)

Similar to the clock hub 3 (H3), the minitrunk and branch hub is the junction where the minitrunk and branch clocks cross. This hub takes in 16 horizontal clocks from the trunk, 12 input interface clocks (branch clocks or minitrunk clocks), and drives out 12 clocks to the trunk and into the fabric, as well as driving 4 clocks out of the fabric. Below is a figure showing the details of the minitrunk and branch hub.

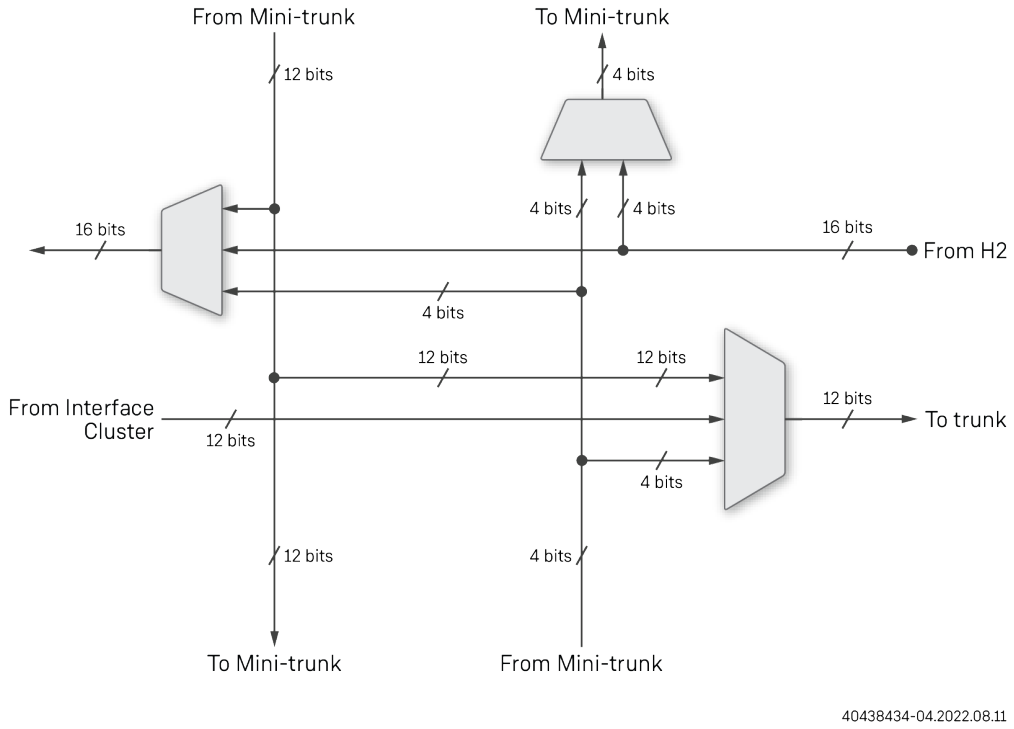


Figure 26 • Detailed View of a Minitrunk and Branch Hub on the North-West Side of the Device

Junctions

Data-to-Clock Junctions

There are multiple junction points in the fabric where a data signal can drive a clock network:

- Clock hub 1 – 16 data inputs
- Clock hub 2 – 16 data inputs (4 data inputs can drive the clock network)
- RLB input – any logic cluster clock can be driven by a selected data signal
- Selected inputs for the BRAM, LRAM, or multiplier/adders input of MLPs

Based on the fanout and other requirements of the clock signal generated in the data interconnect of fabric, an appropriate junction point is selected by ACE software.

Clock-to-Data Junctions

Switching elements can allow some LUT inputs and inputs to BRAMs, LRAMs, and/or multiplier/adders in MLPs to be driven by a signal output by an H2. Specifically, register resets and clock enables with high fan-outs can be driven on the clock network. This allows for a single reset or clock enable signal to route on the clock network to reach various endpoints without using up significant clocking resources (consumes a clock track).

Important!

While there are data-to-clock and clock-to-data junctions inside of the Speedster7t FPGA fabric, it must be ensured that the data and clock input/output pins are used for their respective functionality only. In other words, ensure that:

- Only *data input* signals used as *data* through the Speedster7t FPGA routing interconnect in the fabric are connected to *data* input and output pins.
- Only *clock input* signals routed through the Speedster7t FPGA trunk, minitrunk and branch clock networks are connected to the *clock* input and output pins.

These requirements are important is because there are differences in connectivity between data and clock pins. Connecting to those pins with functions and connectivity not suited for the desired implementation can lead to routing challenges and QoR degradation.

Limitations on the Clock Network

There are some limitations in the Speedster7t FPGA architecture on how many clocks and signals can route to different areas and destinations. The table below lists the limitations on signals to a logic cell, a logic group, an RLB, and within a clock region. Pay close attention to the limitations when reset is routed on the clock network, and when clocks are routed to data pins (a data pin is considered anything that is not a clock, reset, or enable pin, and can be either an input or output pin of a DFF, LUT, ALU, etc.).

Table 6 • Limitation on Clocks, Resets, and Enables

Location	Resets	Enables	Clocks
RLB	3	3	3
Logic group	2	2	1
Logic cell	1	1	1
Clock region without reset or clock enable routed on clock	–	–	16 unique; up to 8 can be gated or switched
Clock region with reset routed on clock	up to 4 resets on clock network	–	(16 – *x* resets) unique; up to 8 can be gated or switched
Clock region with clock enable routed on clock	–	up to 4 clock enables on clock network	(16 – *y* clock enables) unique; up to 8 can be gated or switched
Clock region with clock routing to additional data pins	–	–	12 unique; up to 8 can be gated or switched

Location	Resets	Enables	Clocks
Clock region with reset, clock enable, and/or data-to-clock	–	–	(*x* resets + *y* clock enables + *z* data signals + unique clocks) ≤ 12; up to 8 can be gated or switched

Note

Special care must be taken when using multiple unique clocks that route to data pins. This scenario can occur when there are many derived clocks in a design. It is suggested to take advantage of the built-in clock dividers, clock switches, and clock gates.

Chapter 5 : Speedster7t Interface Clocks

Interface clocks enter the Speedster7t FPGA fabric clusters directly from the four sides to facilitate the construction of interface logic operating on the same clock domain as the interface subsystems. Interface clocks that enter from the north or south drive the minitrunk clocks. Similarly, interface clocks that enter from the east or west drive branch clocks.

Unlike the global core clock network, a single interface clock cannot reach all locations in a device. Instead, as the name implies, interface clocks are intended to be used for clocking logic in the fabric along with the associated interface subsystems. Additionally, interface clocks cannot route across the central trunk or the north/south delimiter. For minitrunk clocks, it is highly recommended designs drive logic *only* within the cluster where the minitrunk clock enters. Keeping logic within the same cluster of the minitrunk clock is imperative to meeting tight performance requirements.

The following figure illustrates how a PLL generates a clock signal for the Ethernet MACs. In turn, the dedicated PLL inside the Ethernet subsystem generates clocks for the direct connect (DC) interface to the fabric and delivers a clock via a CLK_IPIN to the cluster's minitrunk. In this scenario, the designer can assume low clock-insertion delay to any register in that cluster (shaded light gray area), providing the best performance for the adjacent fabric logic to communicate with the Ethernet interface.

The interface subsystems generate separate clocks for each cluster in the fabric. For example, the figure below shows the Ethernet driving a clock to the shaded cluster and also the same clocks to the adjoining light grey cluster. The area shown in blue is reachable only by the south-east minitrunk clocks.

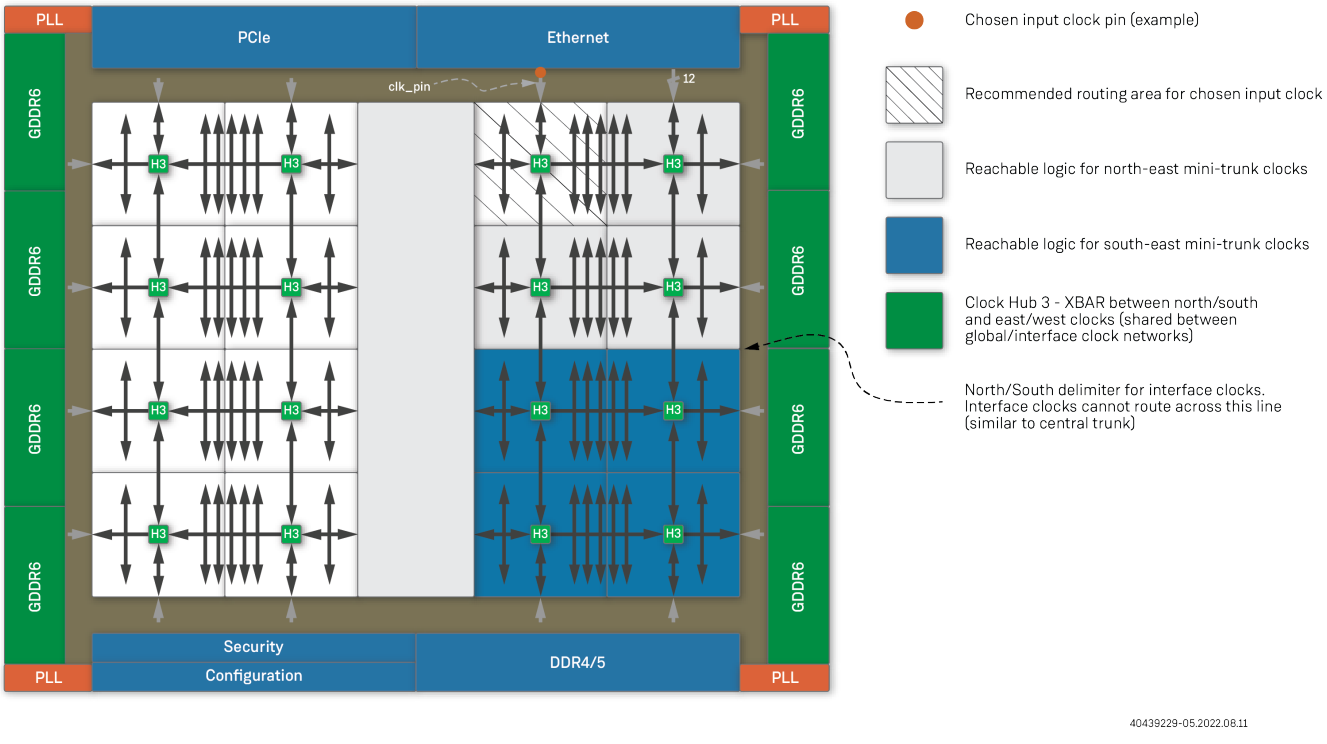


Figure 27 • PLL Driving a Clock Minitrunk

Similar to minitrunk clocks, branch clocks are intended to drive logic close to the interface subsystem where the clock enters. Additionally, branch clocks can only drive logic within the same clock region where they enter. The figure below shows a PLL driving a clock into two GDDR6 controllers. In turn, the dedicated PLLs in the GDDR6 controllers drive separate clocks to the fabric logic in two different clock regions via their direct connect interfaces.

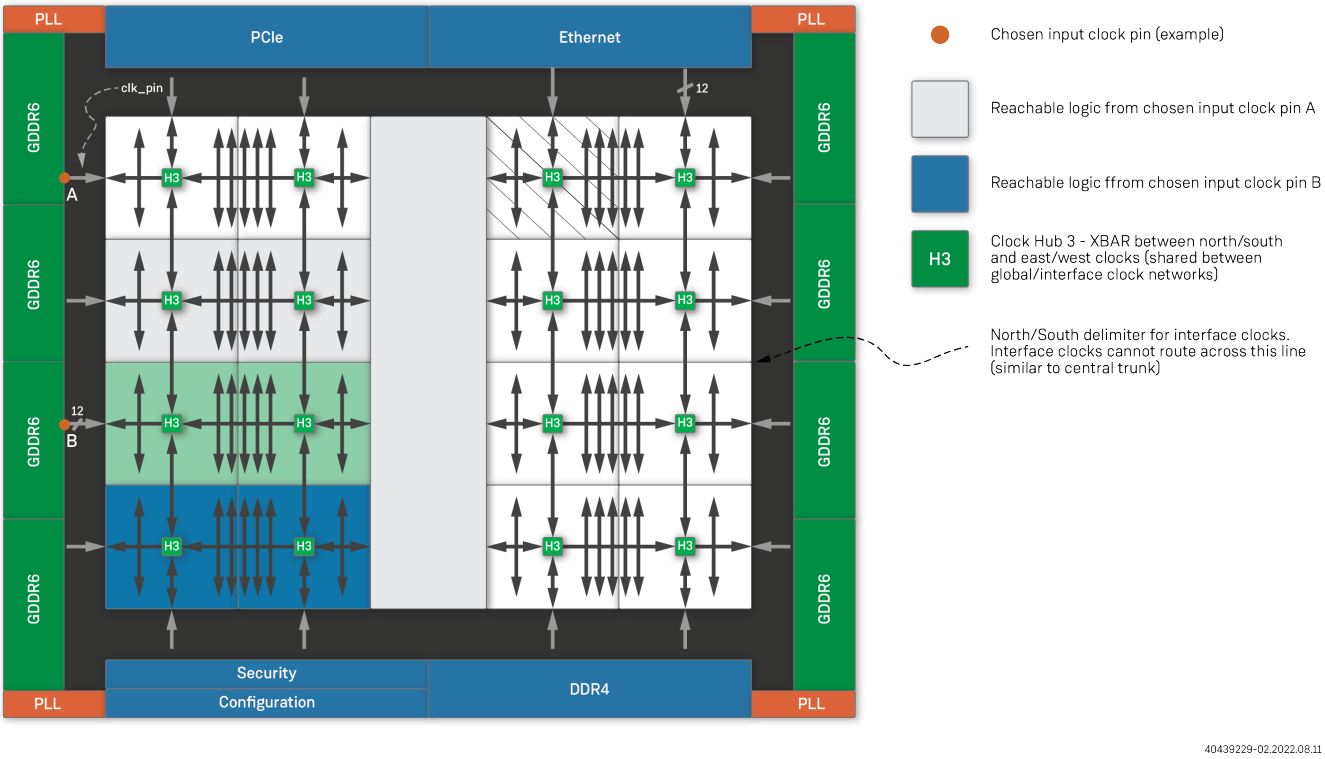


Figure 28 • PLL Driving Multiple Branch Clocks

Chapter 6 : Speedster7t Clock Setting and Reporting

Much of the decision making and optimization for clock selection is automatically performed by ACE software to prevent no-routes. However, ACE does provide some options to specify the type of clock networks to use for particular implementations. Generally, a clock type is determined based on the location of the CLK_IPIN on which it enters the device. If pin locations are assigned for clocks, there is no need to specify the type of clock (e.g., boundary, minitrunk, or trunk). For data-generated clocks, or data signals routed on the clock network, the designer can specify how to route the signal (e.g., local, regional, or center).

The clock type can be specified for a particular clock pin or logic net in the PDC file as follows:

```
set_clock_type -argument <clock or net name>
```

Argument	Usage
-boundary	To specify a clock entering a cluster from either the east or west through an interface cluster. See **Figure: Interface Clock Driving Multiple Branch Clocks** (page 33).
-minitrunk	To specify a clock entering a cluster from either the north or south through an interface cluster. See **Figure: Interface Clock Driving a Clock Minitrunk** (page 33).
-trunk	To specify a clock entering the core through a **clock hub zero (H0)** (page 22). See **Figure: Global Core Clock Network** (page 22).
-data_local	To specify a clock driving an RLB sourced in the fabric.
-data_region	To specify a clock source coming from the fabric to drive a **clock hub two (H2)** (page 25) (drives an entire clock region).
-data_center	To specify a clock source coming from the fabric to drive a **clock hub one (H1)** (page 25) to drive the balanced network across the entire core.

Data Signals Routed on the Clock Network

In general, data signals should always be routed on the data interconnect. For certain very high fan-out signals, such as reset or enable, it can be advantageous to route the signal on the clock network. Additionally, a design may generate clocks in the data fabric, which then need to be routed on the clock network in order to drive clock pins. However, there are trade-offs with this technique in a design. Routing a data signal on the clock network can produce balanced skew to endpoints, but at the same time consumes valuable clocking resources.

Using set_clock_type for Data Signals and Derived Clocks

Some designs need to route data signals on the clock network, or the design creates a generated clock in the fabric that needs to be routed in the clock network. The designer can use the `set_clock_type` constraint in a PDC file to direct ACE to route signals generated in the data interconnect onto the clock network. This constraint has three different options that direct ACE to route the signal in different ways. These options should not be chosen simply based on the fan-out of a signal, but rather the location of its endpoints. It is not advised to use this constraint on a clock originating in the clock network, but rather a data signal that is to be routed on the clock network, or a clock that is originally derived in the data fabric. Below are the different options a designer can use and examples of when to use those options.

Using set_clock_type -data_center

Data routed on clock example 1

```
set_clock_type -data_center {data_signal}
```

The option of `-data_center` should be used when the signal routes to endpoints in more than one clock region. This option tells ACE to route the signal on the clock network through the main clock crossbar hub (Hub1) in the main clock trunk. From there it can reach endpoints in multiple clock regions. This option is used for a reset signal that reaches endpoints all over the device. If the design uses a large number of clocks as well, the designer needs to keep in mind the limitations on the number of clocks that can be routed out of the main clock crossbar (48 total clocks). Additionally, while this solution provides balanced delays to endpoints, it does cause larger insertion delays on the signal because it routes on the main trunk. The figure below shows a data signal routed on the clock network using the `-data_center` option.

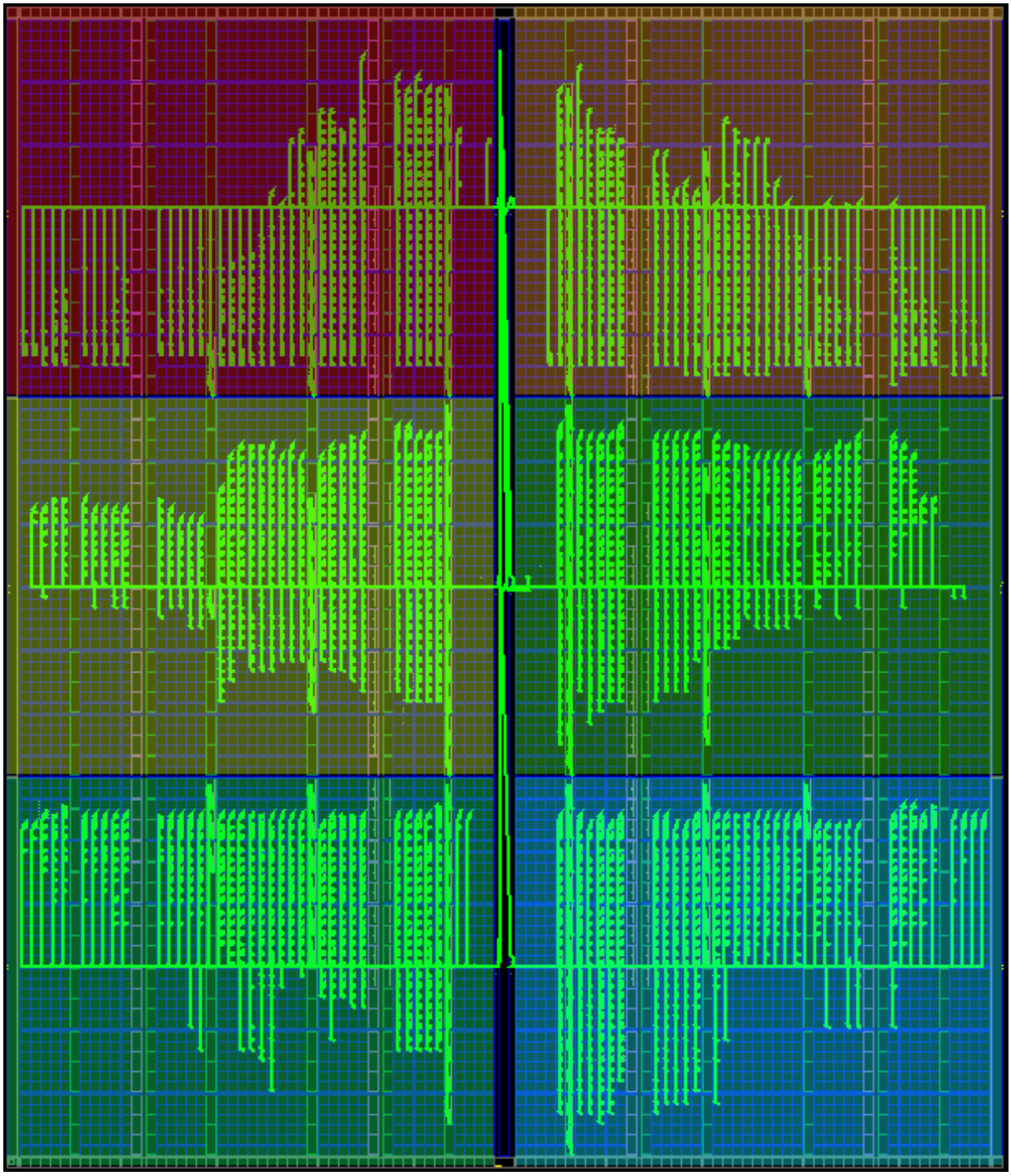


Figure 29 • Reset (Data Signal) Routed on a Clock Trunk

Using set_clock_type -data_region

Data routed on clock example 2

```
set_clock_type -data_region {data_signal}
```

The option `-data_region` should be used when routing a signal to endpoints in only one clock region. This option tells ACE to route the signal on the clock network through the clock crossbar for the particular clock region (Hub2). From there it can reach endpoints in only that specific clock region. A designer selects this option when there are a limited number of endpoints within a clock region. This option has the added benefit of only affecting the number of clocks available within that one clock region.

Below is a figure showing a data signal routed on the clock network using the option `-data_region`.

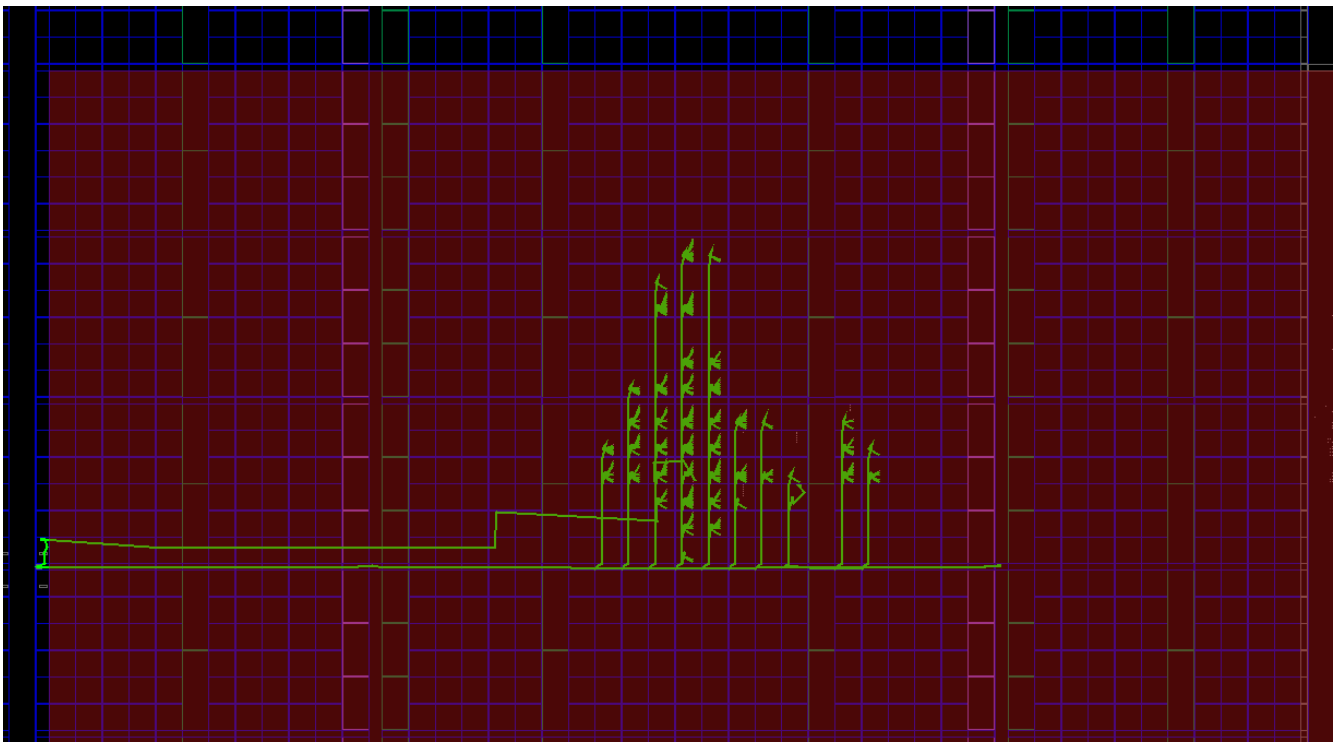


Figure 30 • Reset (Data Signal) Routed on Regional Clock Network

Using set_clock_type -data_local

Data routed on clock example 3

```
set_clock_type -data_local {data_signal}
```

The option `-data_local` should only be used in very specific cases. Generally, it is *not* recommended to use this option if trying to route a data signal such as reset or enable on the clock network, because these signals tend to have high fan-out. This option should only be used with very low fan-out. The `-data_local` option can be beneficial when using a clock derived in the data fabric that drives a small number of clock pins.

The option `-data_local` results in very low insertion delay compared to clocks routed on the main trunk or the regional clock network and does not consume precious clocking resources in the clock region (will not route through the Hub2 clocking element). However, the resulting clock may not be well balanced and can have large skew when fan-out is high. If the designer knows the clock signal is only reaching a small handful of endpoints (less than 12), and those endpoints are all immediately near each other, ideally within the same tile, the designer can set this option. The designer should consider hand-placing the endpoints to ensure they are placed close enough to make timing successful with the `-data_local` option.

The figure below shows a clock signal derived in the data fabric and routed on the clock network, using the option `-data_local`.

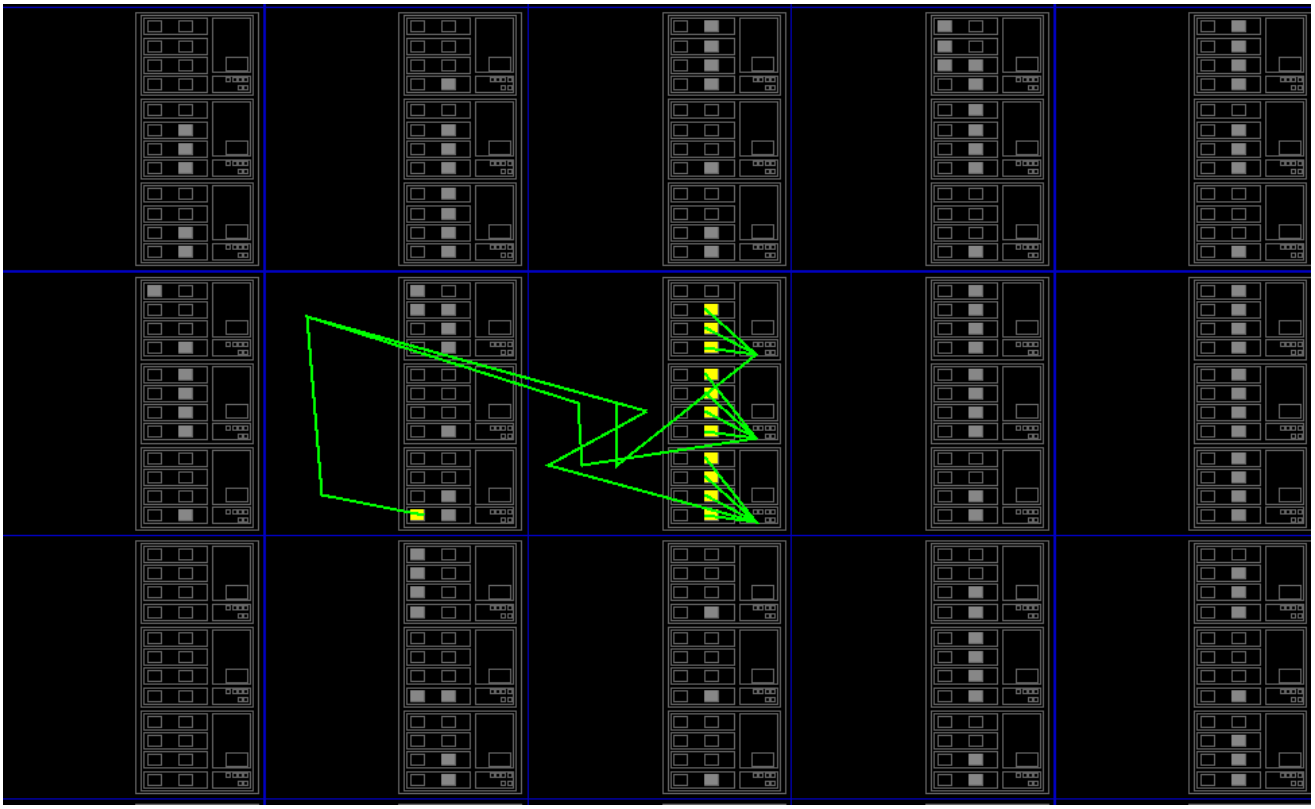


Figure 31 • Data-Derived Clock Routed on Local Clock Network

Clock Reporting

When the clock constraints are specified in the design, and the design is run through a full ACE compilation flow, the routing step outputs text and HTML files called `<design_name>_clocks_routed` which describe the clock relationships, clock constraints and clock regions in the design. The clock regions section provides detailed information on how each of the clocks in the design were routed and how they are distributed in the used clock regions, including the boundary. A legend is also provided in the file to help decipher the routing details.

Chapter 7 : Speedster7t Reset Network

Each Speedster7t FPGA has a global reset network used to send reset signals to the entire device. There are three variations on how a design can reset the different portions of the device.

- Global reset signal from the FPGA configuration unit (FCU) which resets the FPGA fabric and all interface subsystems after the device is configured.
- Reset signals that enter the device as **inputs** (page 6).
- Reset signals generated in the FPGA fabric.

For the first method, this reset occurs automatically — no action is required on the part of the user. To control reset within the user design, adding options two or three above allow for an easy solution. The global reset network receives external reset inputs from Clock I/O banks, as well as resets generated internally in the device from fabric logic.

A clock I/O bank is located at each of the four corners of a AC7t1400/1500 device and the northwest (NW) and southwest (SW) corners of a AC7t700/800 device. Each clock I/O Bank has six pins: two MSIO, and four REFIO. While these clock I/O can be used to drive reset signals in to the global reset network on the Speedster7t device, these I/O cannot be used to drive reset signals out to a board.

Each MSIO pin can be configured independently as either a clock input, a clock output, a reset input, or they can be configured together as a differential pair (with the same choices). The two MSIO pins are allowed to be configured in opposing directions, unlike the REFIO pairs.

A REFIO pair, when configured independently, must have both ports going in the same direction (either in or out). Each REFIO pair can be configured as follows:

- one differential clock input
- one differential clock output
- one differential reset input
- one or two single-ended reset inputs
- one single-ended clock input (must use P)
- one single-ended clock output (must use P)
- one single-ended clock input (must use P) and one single-ended reset input (must use N)

Essentially, the REFIO P is like an MSIO in that it can handle any of the options. The REFIO N can follow the P as any of the differential options, or can be an independent reset input only if the P is also an input (clock or reset).

Therefore:

- The maximum CLKIO clock inputs for a clock I/O bank is 4.
- The maximum CLKIO clock outputs for a clock I/O bank is 4.
- The maximum CLKIO reset inputs for a clock I/O bank is 6.

Note

- In an AC7t1400/1500 device, the core fabric can only receive reset signals from the NE and NW REFIO_0 clock I/O pins.
- In an AC7t700/800 device, the core fabric can only receive up to four reset signals from any of the NW and SW MSIO and REFIO clock I/O pins.

The interface subsystems can receive reset signals from the global reset network via the FCU reset, a global reset from an input of a clock I/O bank, or they can receive reset locally as an output from the FPGA fabric. A reset from a clock I/O bank can reach multiple interface subsystems. If using a local reset signal, each interface subsystem receives a separate local reset signal from the adjoining fabric cluster. To send the same reset to multiple interface subsystems, the reset signal must be replicated in the design and the FPGA core used to route the reset signal out to each interface subsystem. For reset de-assertion, each subsystem synchronizes the reset signals, providing synchronous reset de-assertion across subsystems. Additionally, each interface subsystem ignores inputs and drives outputs appropriately until the subsystem comes out of reset. This delay helps isolate the blocks while in reset and prevents accidental random transactions.

Chapter 8 : Revision History

Version	Date	Description
1.0	22 May 2019	• Initial Achronix release.
1.1	06 May 2020	• Added significant details to **Speedster7t Clock and Reset Generation** (page 6). • Minor updates to **Speedster7t Global Core Clock Network** (page 22). • Updated figures and text in **Speedster7t Interface Clocks** (page 33). • Updates and additional details added to **Speedster7t Reset Network** (page 41).
1.2	18 Jul 2022	• Provide additional details in **Speedster7t Clock and Reset Generation** (page 6). • Provide clarification to description of Clock Hub 0 in **Speedster7t Global Core Clock Network** (page 22). • Provide additional information in **Speedster7t Reset Network** (page 41).
1.3	30 Oct 2025	• Deprecate references to Speedster AC7t1550 device. • Updated to cover both AC7t700/800 and AC7t1400/1500 devices. • De-featured the PLL's DLL functionality, including screenshot updates. • Other minor updates.