Speedster7t Clock and Reset Architecture User Guide (UG083)

Speedster FPGAs

Preliminary Data

Achronix®
Data Acceleration

Copyrights, Trademarks and Disclaimers

Preliminary Data

This document contains preliminary information and is subject to change without notice. Information provided herein is based on internal engineering specifications and/or initial characterization data.

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

Achronix's new 7nm Speedster®7t FPGA family is 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 of high performance and complex designs, the clock network for Speedster7t FPGAs has been designed with numerous high performance clocks that allow for maximum routability. This document explains the architecture of the different clock networks in a Speedster7t FPGA, as well as information on how to use the clocks. It is intended to help designers understand and choose the best clocking options for their design on a Speedster7t FPGA.

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, machine learning processors (MLPs) that contain multipliers, adders, accumulators, and memory blocks. The tiles are distributed as columns in the Speedster7t FPGA, and 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.

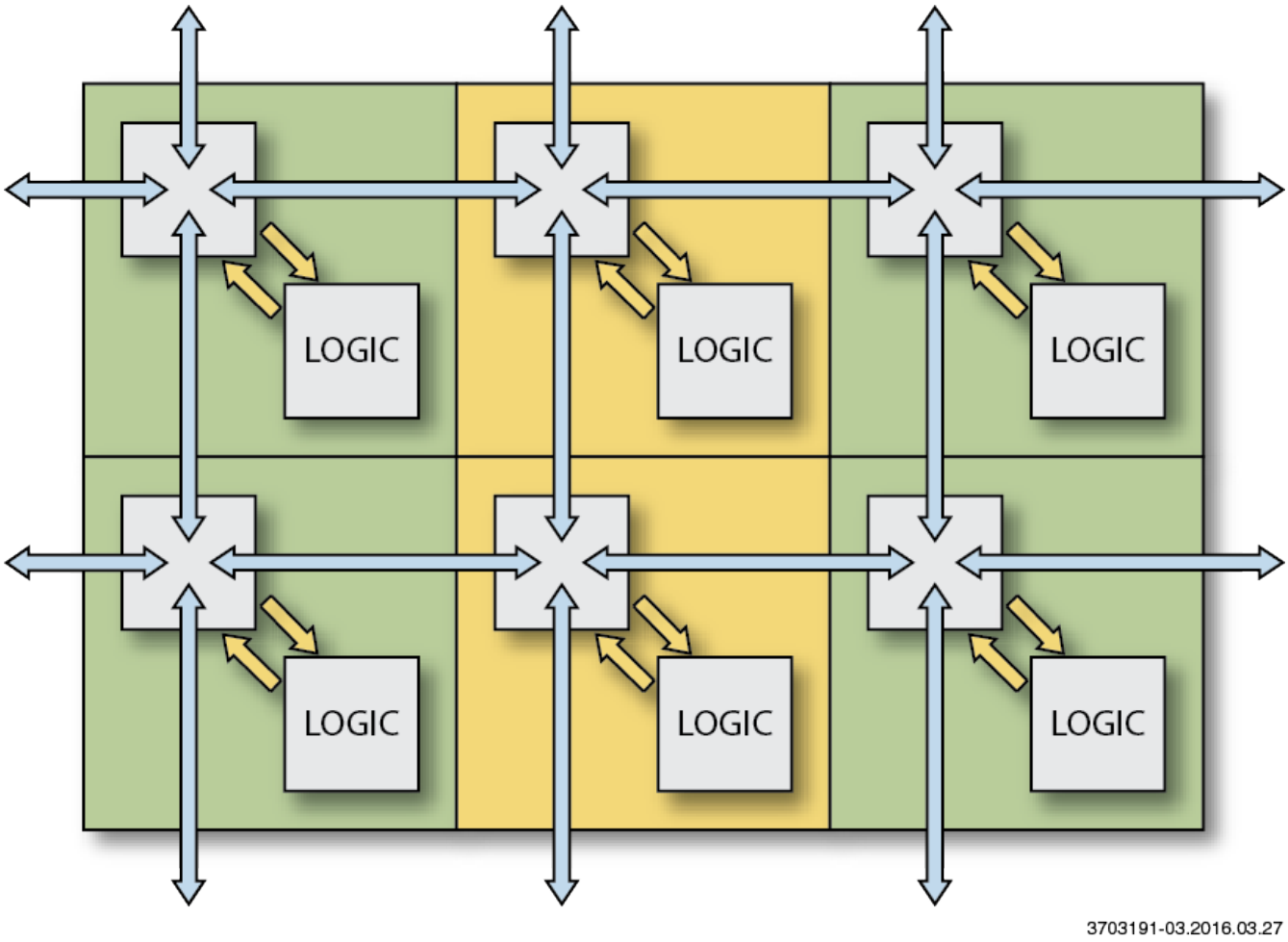


Figure 1: ***Tiles Assembled in Fabric***

These tiles are then grouped to form clusters. Clock signals can enter or exit from the top or bottom on the clock mini-trunk (vertical purple arrow), and from the left or right on the clock branch (horizontal purple arrow). The clock mini-trunk and branch are connected at the intersection via a crossbar hub. Clocks are distributed throughout the cluster using clock stems (shown in light blue in the following figure). Clusters are then grouped horizontally to form clock regions on either side of the global clock trunk.

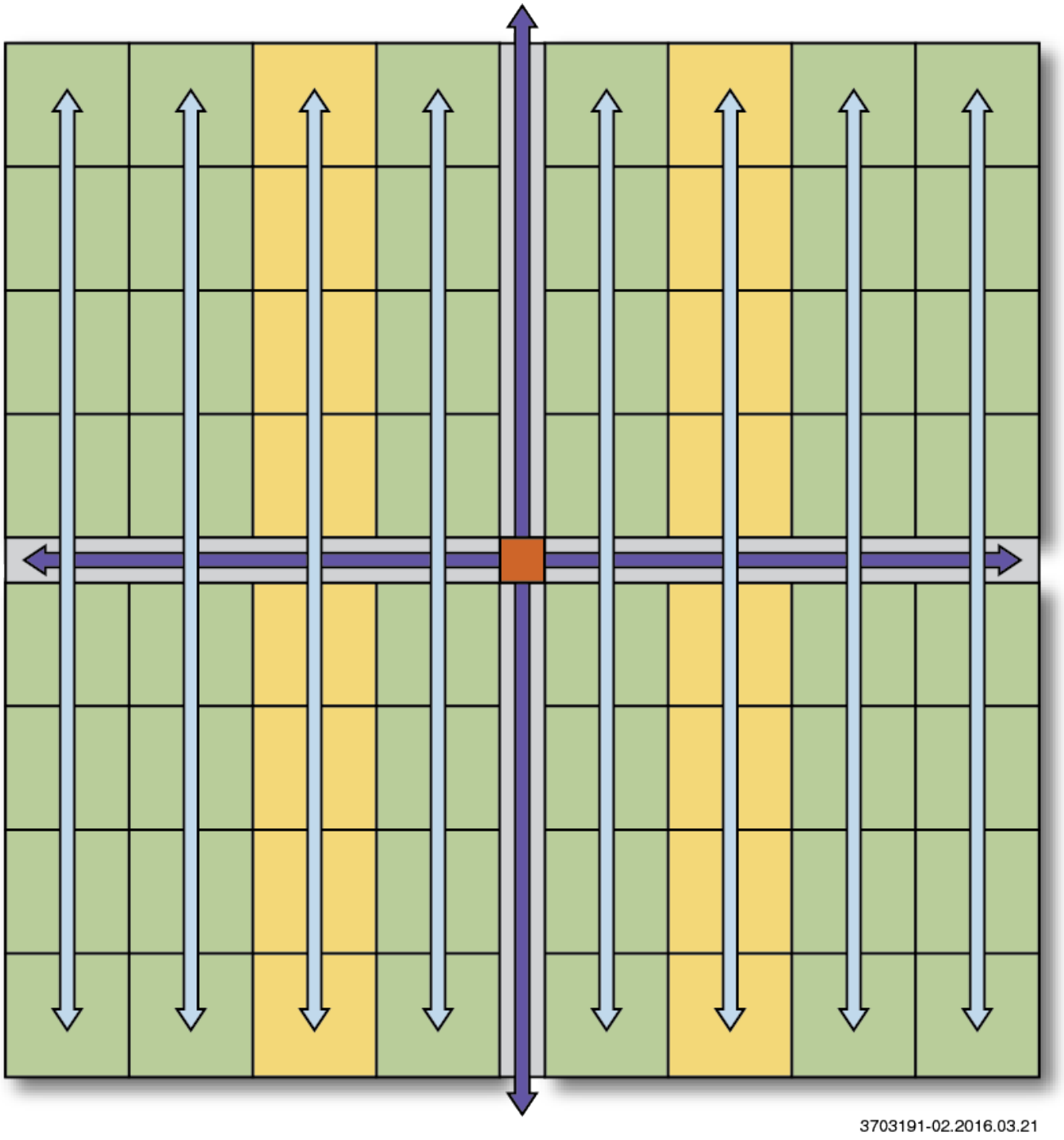


Figure 2: ***Clock Distribution within a Cluster***

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 sixteen PLLs which can be used for clock generation. Additionally, the Speedster7t FPGA includes an integrated Network on Chip (NoC) that can interact directly with the fabric.

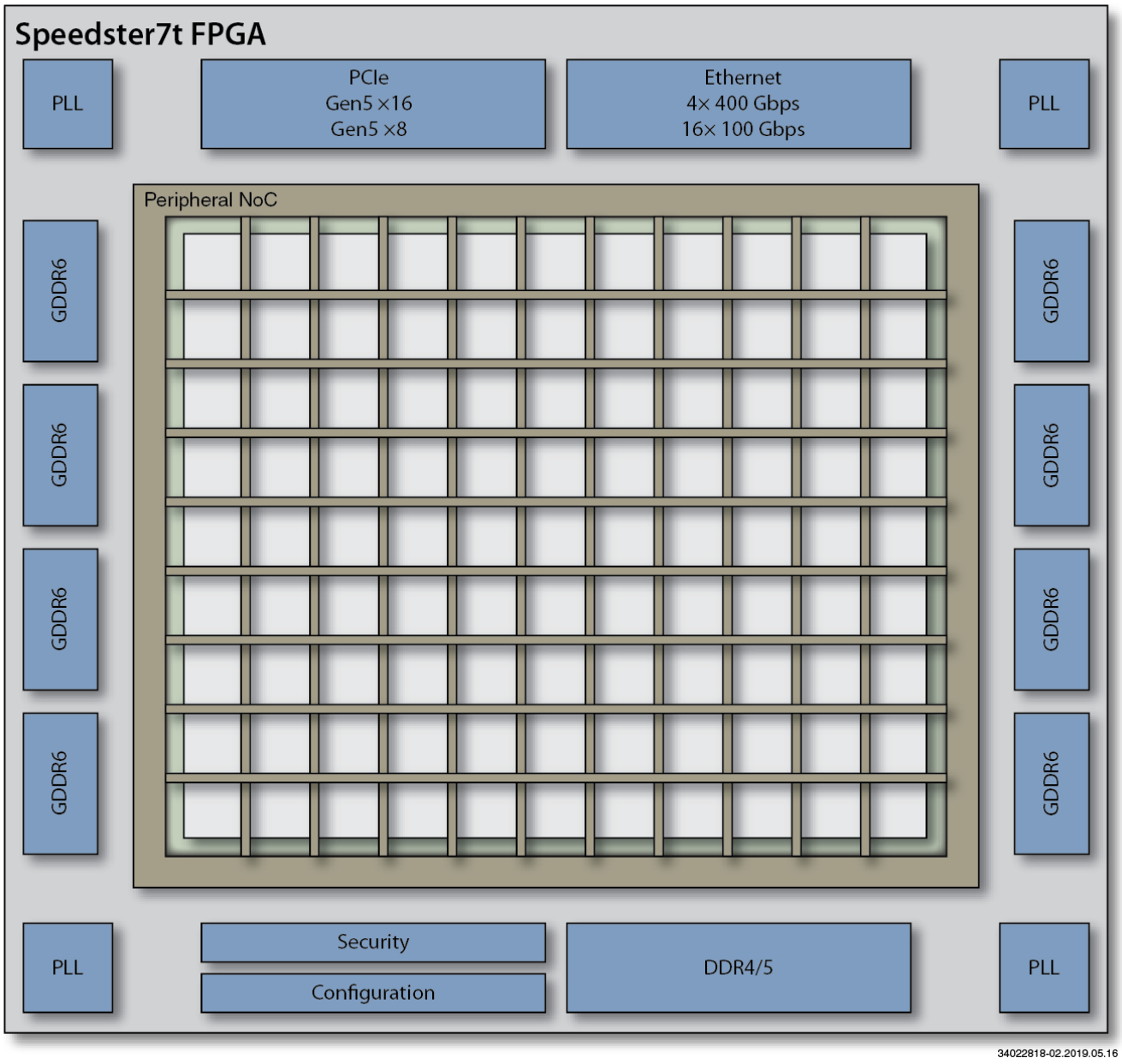


Figure 3: ***Speedster7t1500 Top-Level Block Diagram***

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

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

Chapter - 3: Speedster7t Clock Generation and PLLs

Clock Generation

The Speedster7t FPGA includes four on-chip oscillators in each corner of the device, for a total of sixteen, to provide clocks for user designs. The oscillators are 2 GHz and 100 MHz free-running clocks and include statically configured clock dividers for common values such as 1×, 2×, 4×, 8×, 16×, etc.

> **Note**
>
> On-chip oscillators are not currently configurable in ACE.

PLLs

There are sixteen general purpose PLLs, four in each corner of the Speedster7t FPGA. 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.

Below is a list of features available in the PLLs:

- 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 clock pins and PLLs from other device corners
- Up to four output clocks
- Reference clock and output clock dividers
- Output duty cycle 50%
- Low jitter
- Low power

> **Note**
>
> ACE currently supports the programmable fractional-N divide. Spread-spectrum is not configurable at this time.

Table 1: ***Details of PLL***

Parameter	Min	Max	Units
Reference frequency	10	600	MHz
Output frequency	7.5†	2000	MHz

Parameter	Min	Max	Units
Maximum long-term jitter	±2% divided reference clock		

† 0.93MHz minimum frequency after divider

Each PLL can generate up to four output clocks. Of those output clocks, 32 are available to route on the global clock network. 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 device.

Configuring PLLs

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.

To add a new configuration for a PLL, switch to the IP configuration perspective in an existing ACE project. First, create a clock input using the Programmable I/O configuration GUI. Here the customer can specify the name of the clock input, the I/O standard to use, the input frequency, and the placement of the clock input on a specific pad.

Once an input clock is available, the user can configure the PLL using the PLL configuration GUI in the I/O Designer Toolkit. Here the user can specify which of the sixteen PLLs to use, the input reference clock, and the total number of output clocks to produce. Each PLL allows for up to four output clocks. For the first output, the user specifies the desired output frequency, along with the output clock name and whether to expose the signal to the FPGA fabric. For each additional output clock, the user specifies the desired clock divider value with respect to the produced VCO frequency, output clock name, and whether to expose the clock for use in the FPGA fabric.

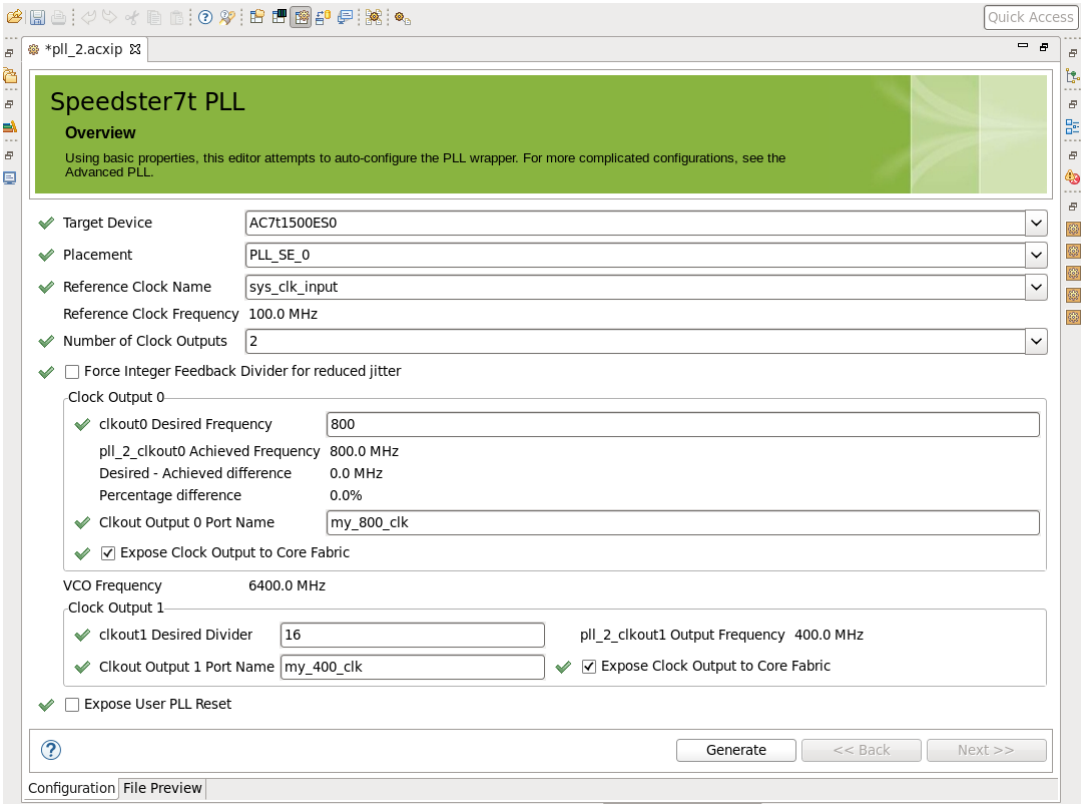


Figure 4: *PLL Configuration GUI in I/O Designer Toolkit*

In the above example, a 100 MHz input reference clock called `sys_clk_input` is chosen from a pull-down menu, as it is one of the input clocks available. 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 is set to 800 MHz and is called `my_800_clk`. The second output uses a divide by 16 of the calculated 6400 MHz VCO frequency, to achieve 400 MHz. Both clocks are exposed to the FPGA fabric, as indicated by checking the box "Expose Clock Output to Core Fabric".

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 user can choose to not expose the clock output 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. Below is a figure showing the placement of the differential clock input and PLL0 located in the south-east corner of the device, shown in green.

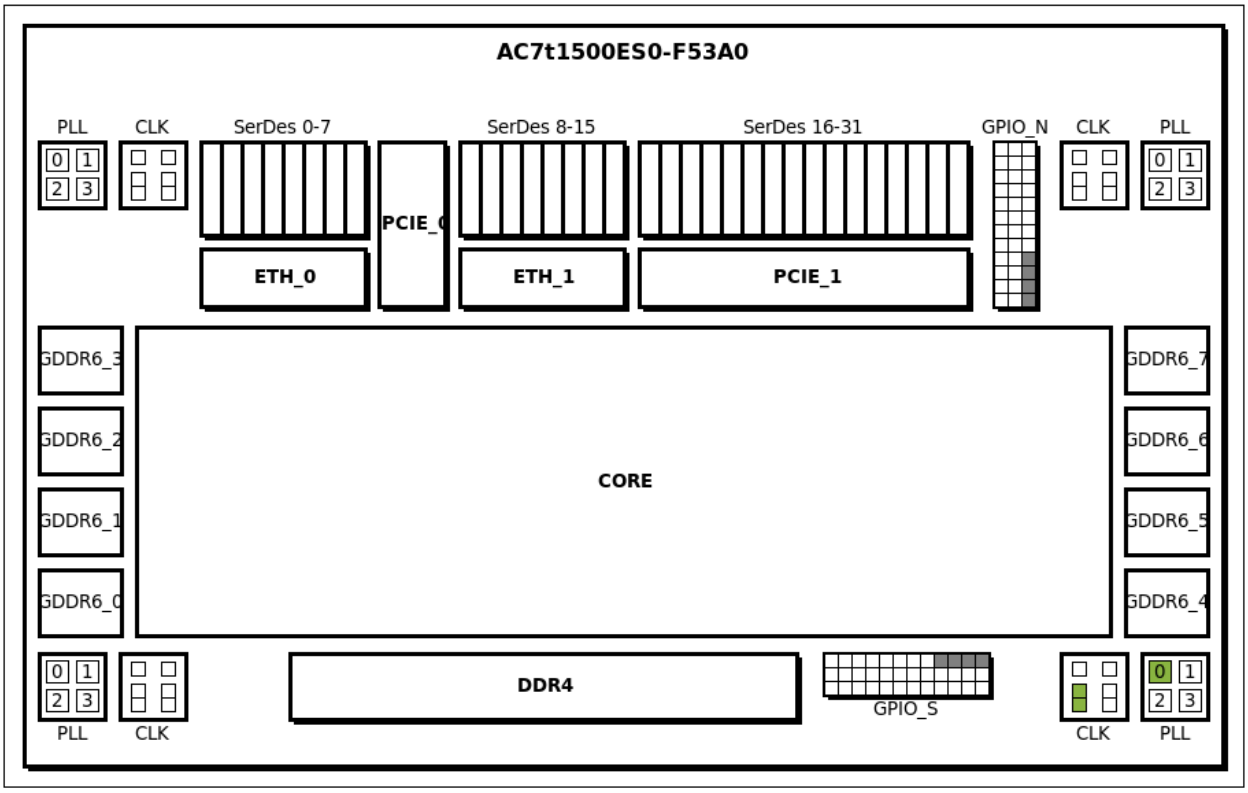


Figure 5: ***Differential Clock and PLL Placed on AC7t1500***

Dedicated PLLs

Along with the sixteen general-purpose PLLs, there are several dedicated PLLs for use by 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 2 GHz clock to the NoC. Below is a figure showing the NoC configuration GUI. The user can choose the appropriate 200 MHz clock from the pull-down menu to assign to the input of the NoC's PLLs.

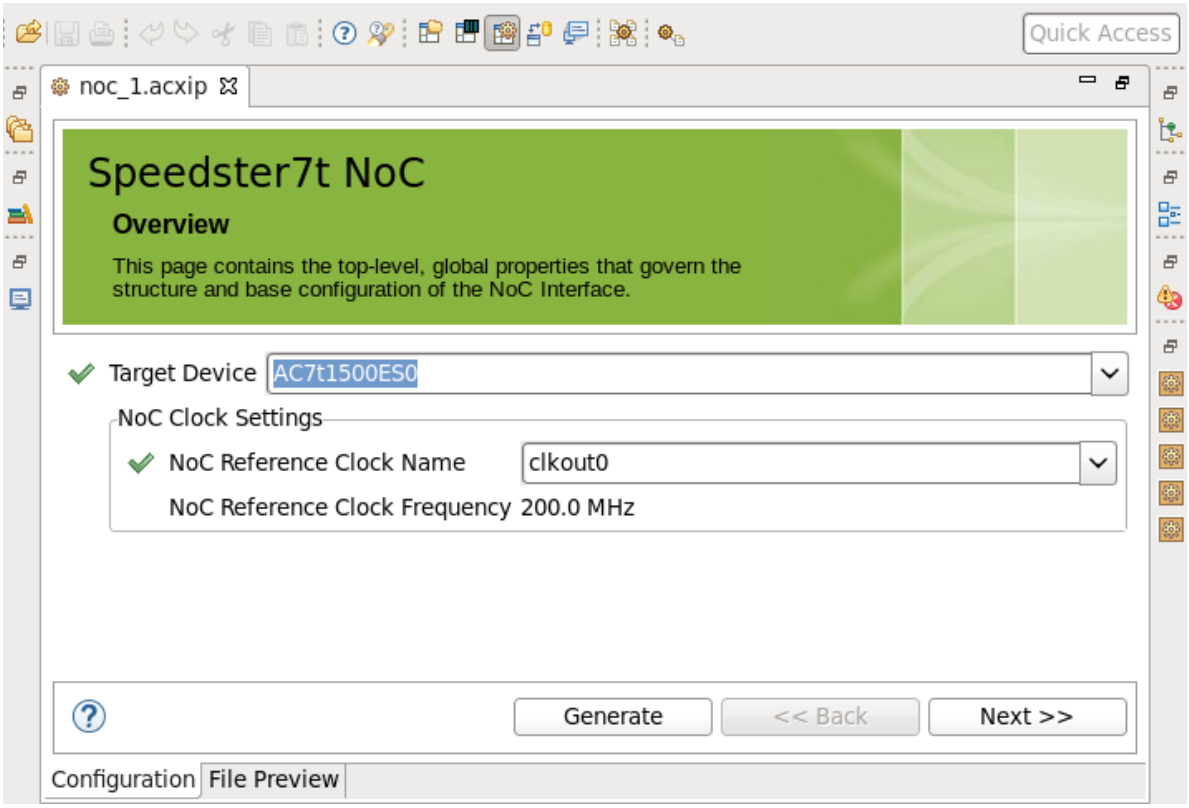


Figure 6: *NoC PLL Input Clock Configuration*

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. These dedicated PLLs are fed by outputs of the sixteen user PLLs.

The specific clock and frequencies are specified in the corresponding interface subsystem configuration GUI in the I/O Designer Toolkit. Below is a figure showing 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.

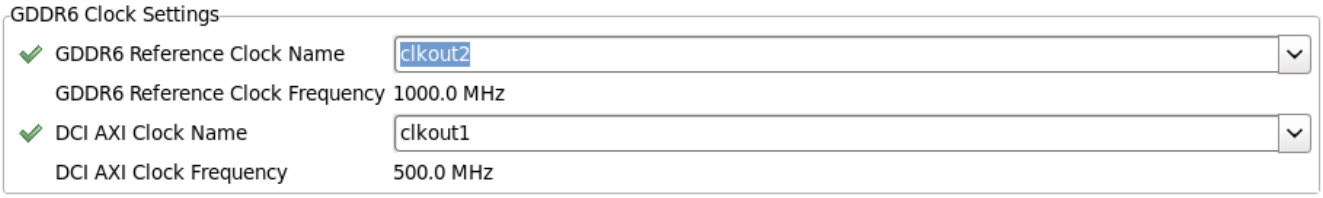


Figure 7: *Dedicated PLL Configuration for GDDR6*

Each SerDes quad has a PLL, which derives the receive clocks. These PLLs can be used for driving the receive and transmit logic in the SerDes, PCIe, Ethernet, as well as driving the global clock trunk and mini-trunk 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 *Speedster7t GDDR6 User Guide* (UG091), *Speedster7t DDR User Guide* (UG096), *Speedster7t Ethernet User Guide* (UG097), and *Speedster7t PCIe User Guide* (UG098).

DLLs

In each corner of a Speedster7t FPGA there is one master DLL with eight slaves available for the phase shifting of clocks. This arrangement allows for one master clock and up to eight slave clocks that can be phase shifted based on the master clock's frequency.

> **Note**
>
> DLLs for clocks are not currently configurable in ACE.

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 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.

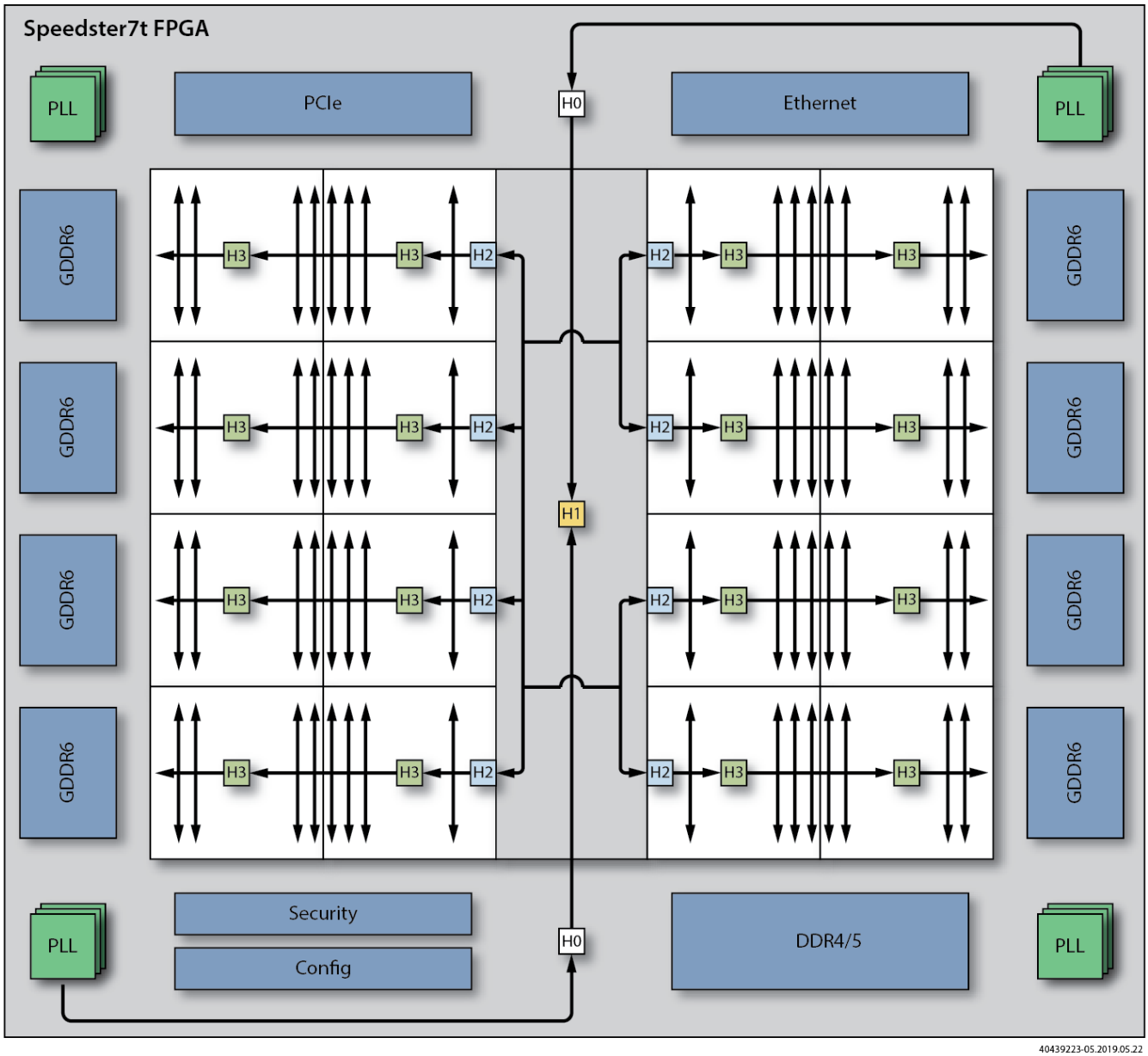


Figure 8: ***Global Core Clock Network***

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 (*N* varies depending on the Speedster7t device) arriving from the clock pins 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 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 they do not toggle during configuration. When this reset is de-asserted, these clocks are un-gated glitchlessly 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 be used as user resets after configuration but before normal user mode, and then as clocks during normal user mode. The remaining *N-3* bits are used as clock inputs. Bit *N-1* is the asynchronous user reset, and bit *N-2* is the synchronous reset. 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 on bits *N*-1 and *N-2*, and the remaining clock inputs (bits *N-3*), 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.

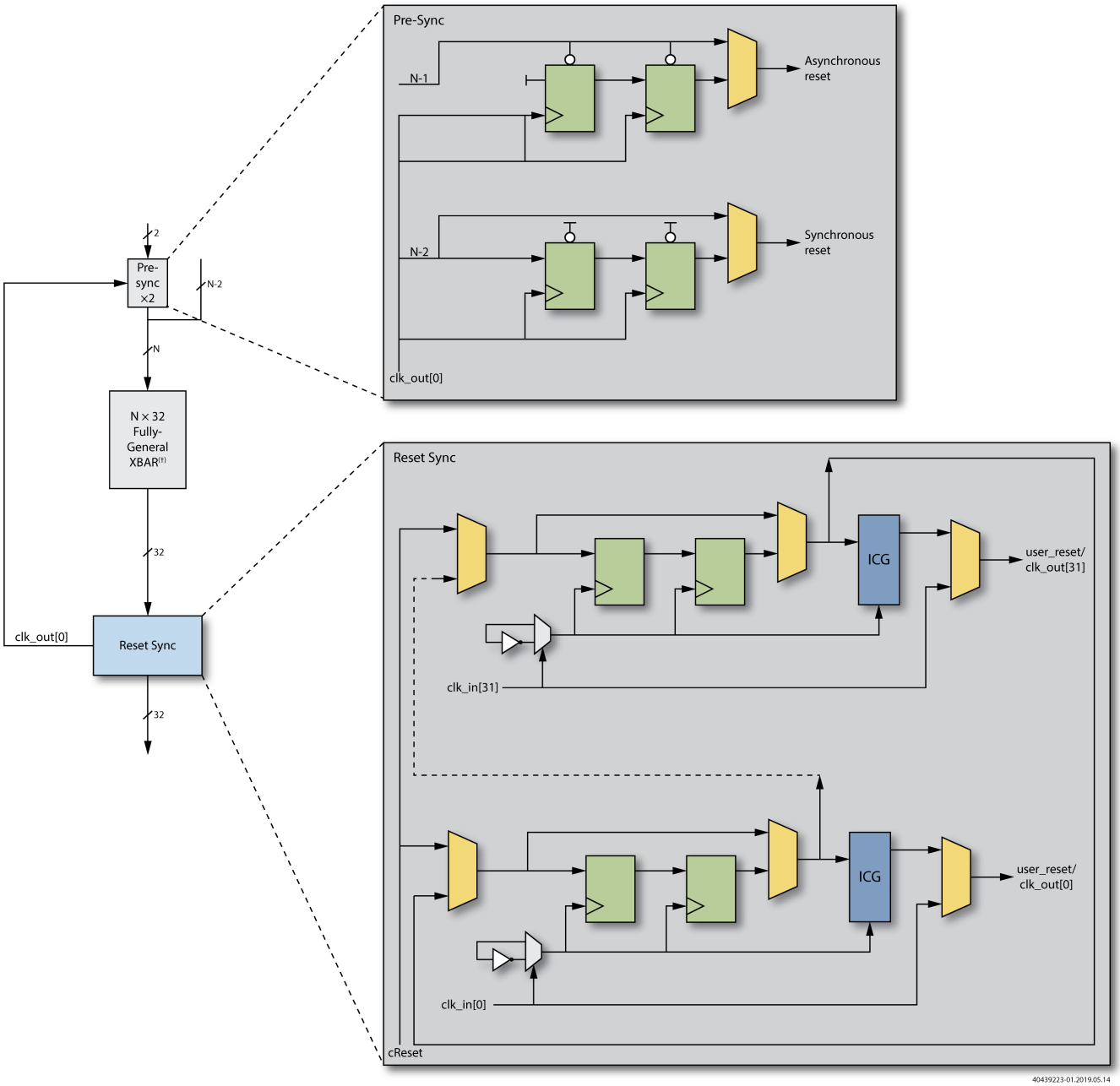


Figure 9: ***Clock Hub (H0)***

Figure Note

† 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.

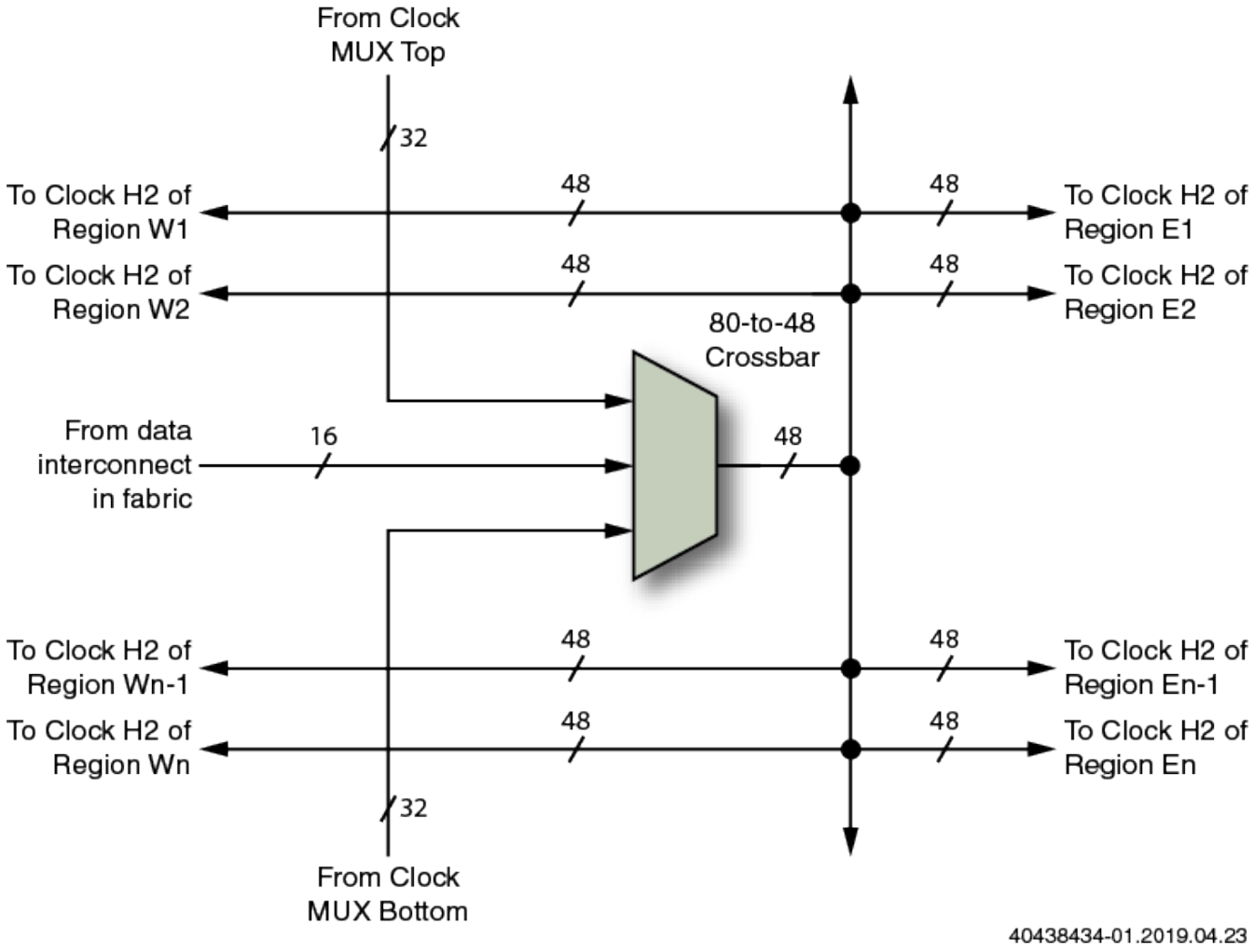


Figure 10: ***Detailed View of the Clock Hub 1***

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 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). Please note that 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.

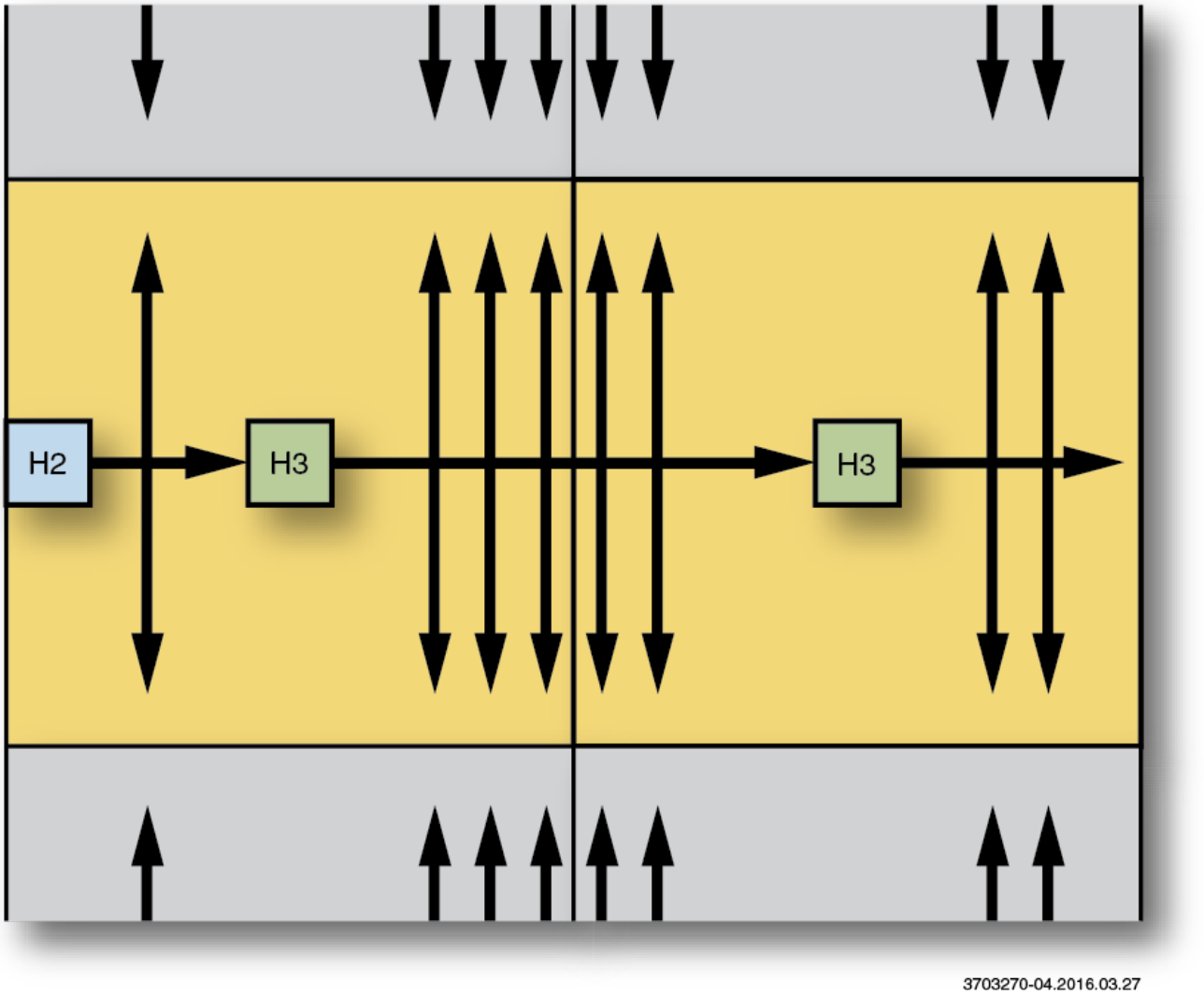


Figure 11: ***Detailed View a Clock Region***

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 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.

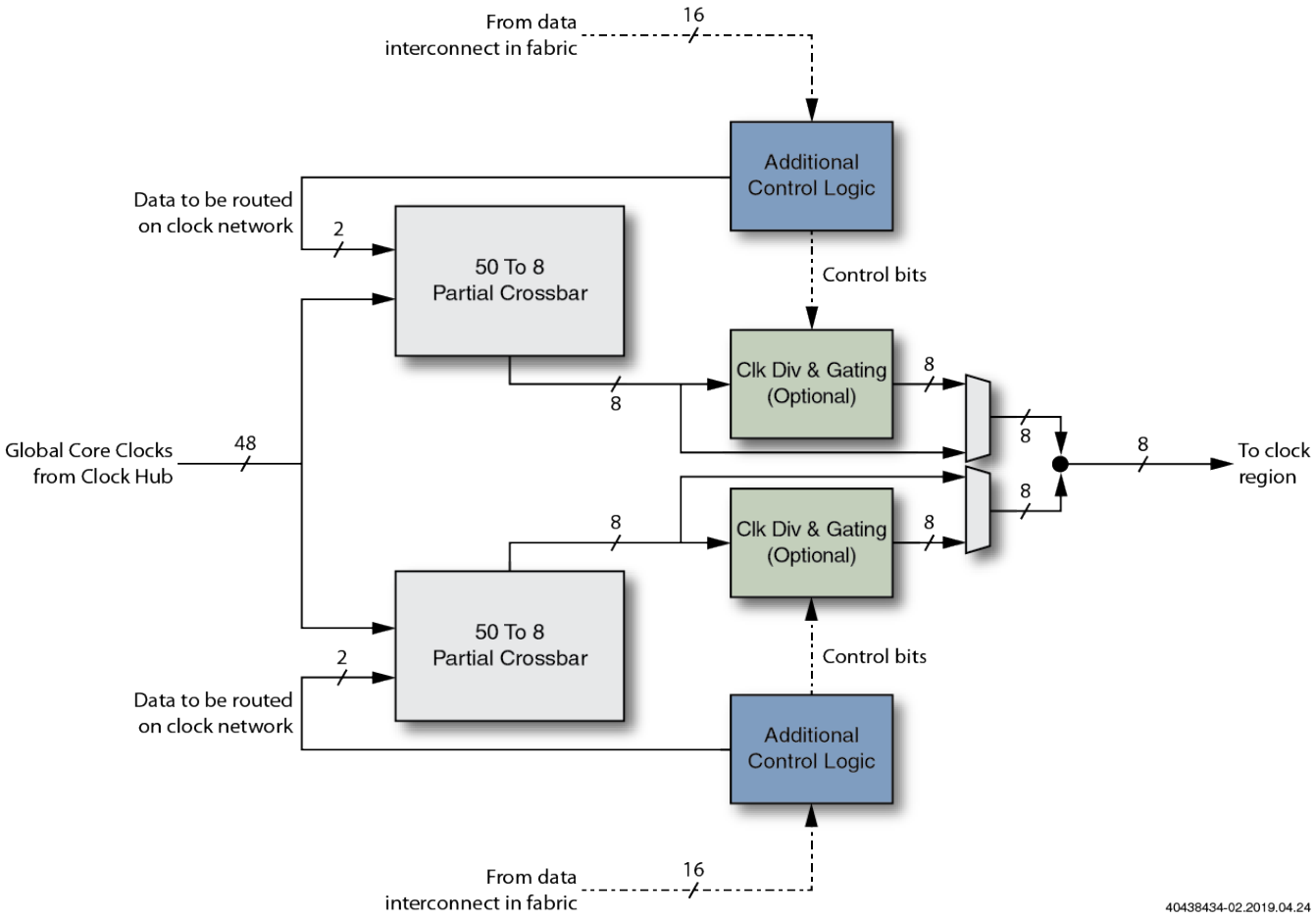


Figure 12: ***Detailed View of a Clock Hub 2***

The figure below 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 and these functions exist for all of the clock inputs.

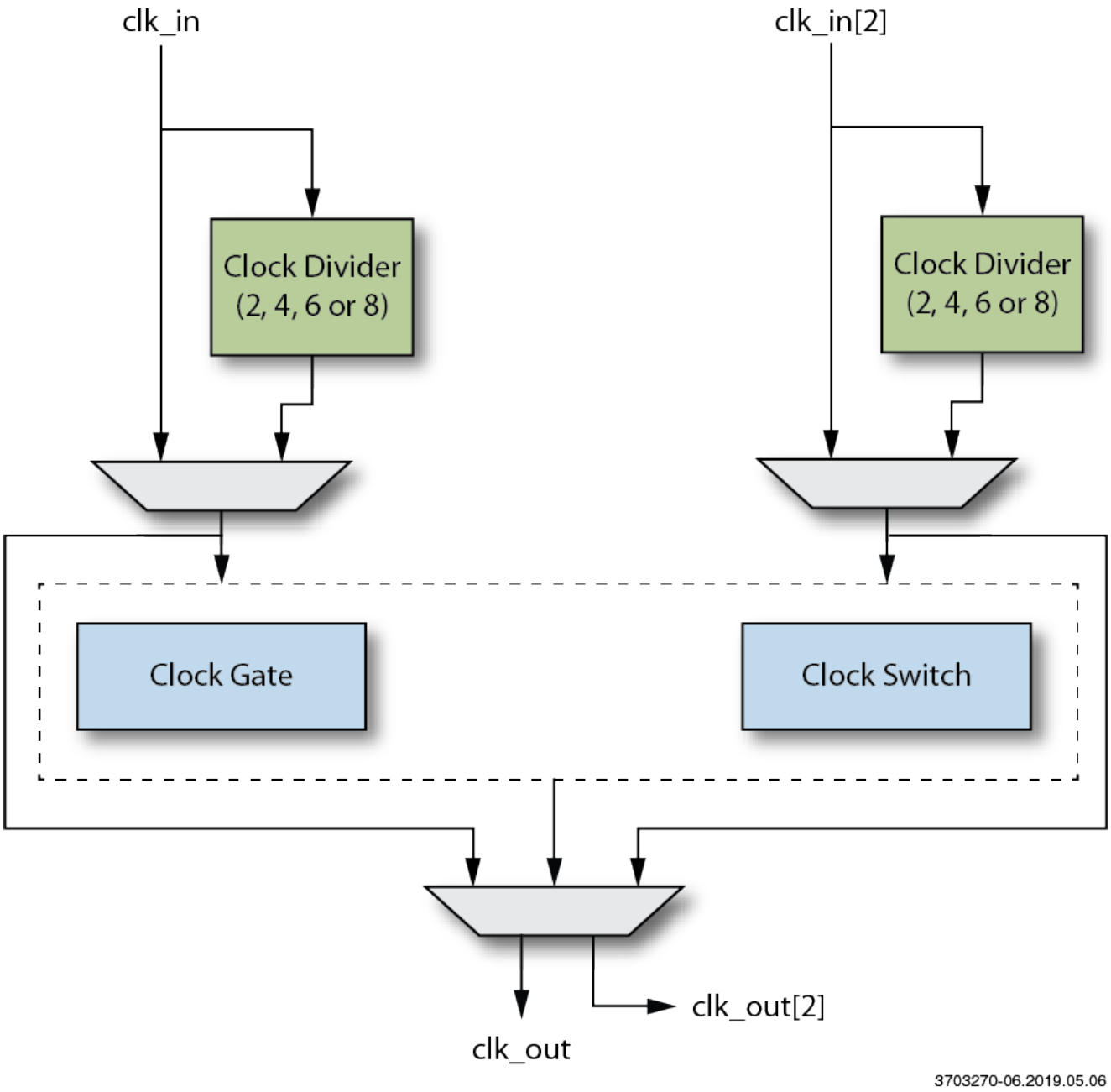


Figure 13: ***Internals of the Clock Division, Gating and Switching Circuitry***

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.

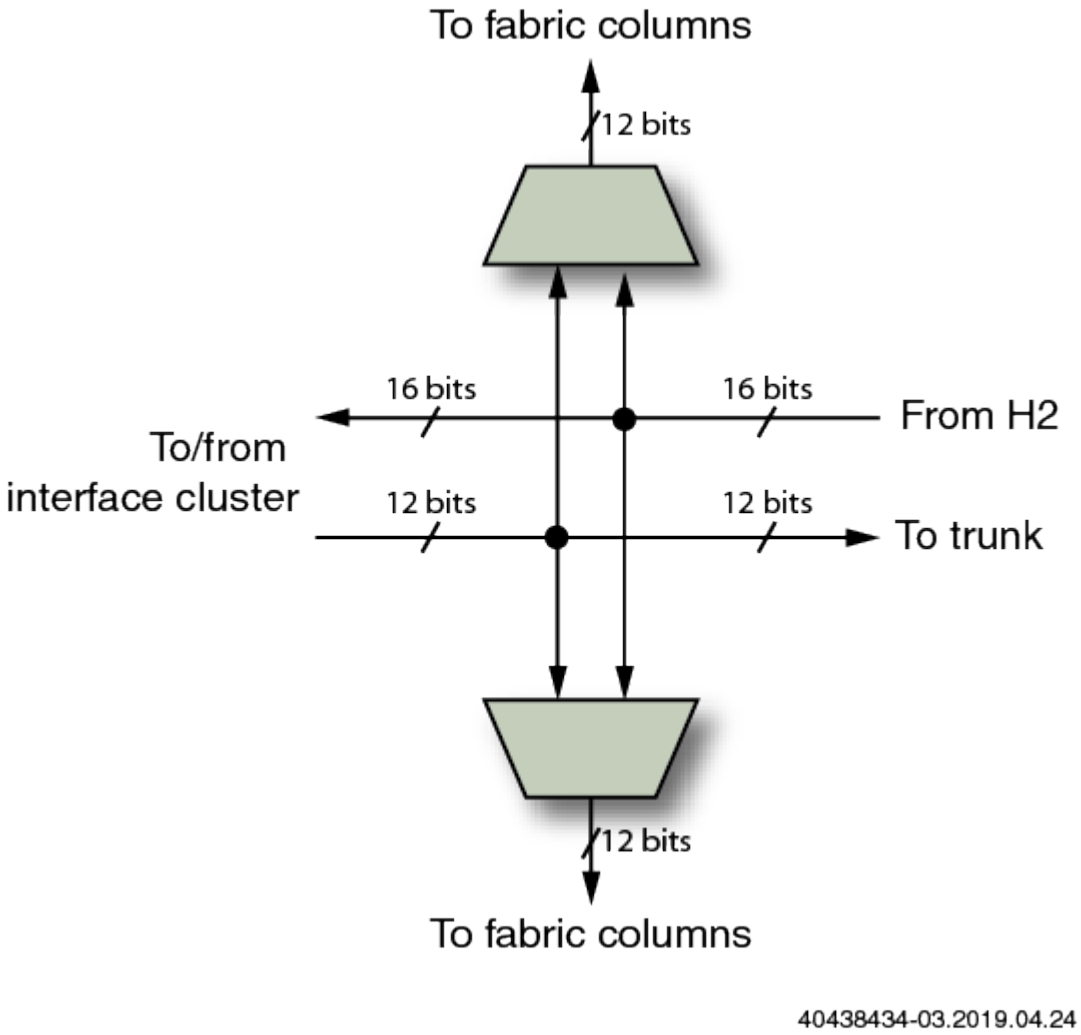


Figure 14: ***Detailed View of a Hub 3 on the West Side of the Device***

Mini-trunk and Branch Clock Hub

Similar to the clock hub 3 (H3), the mini-trunk and branch hub is the junction where the mini-trunk and branch clocks cross. This hub takes in 16 horizontal clocks from the trunk, 12 input interface clocks (branch clocks or mini-trunk 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 mini-trunk and branch hub.

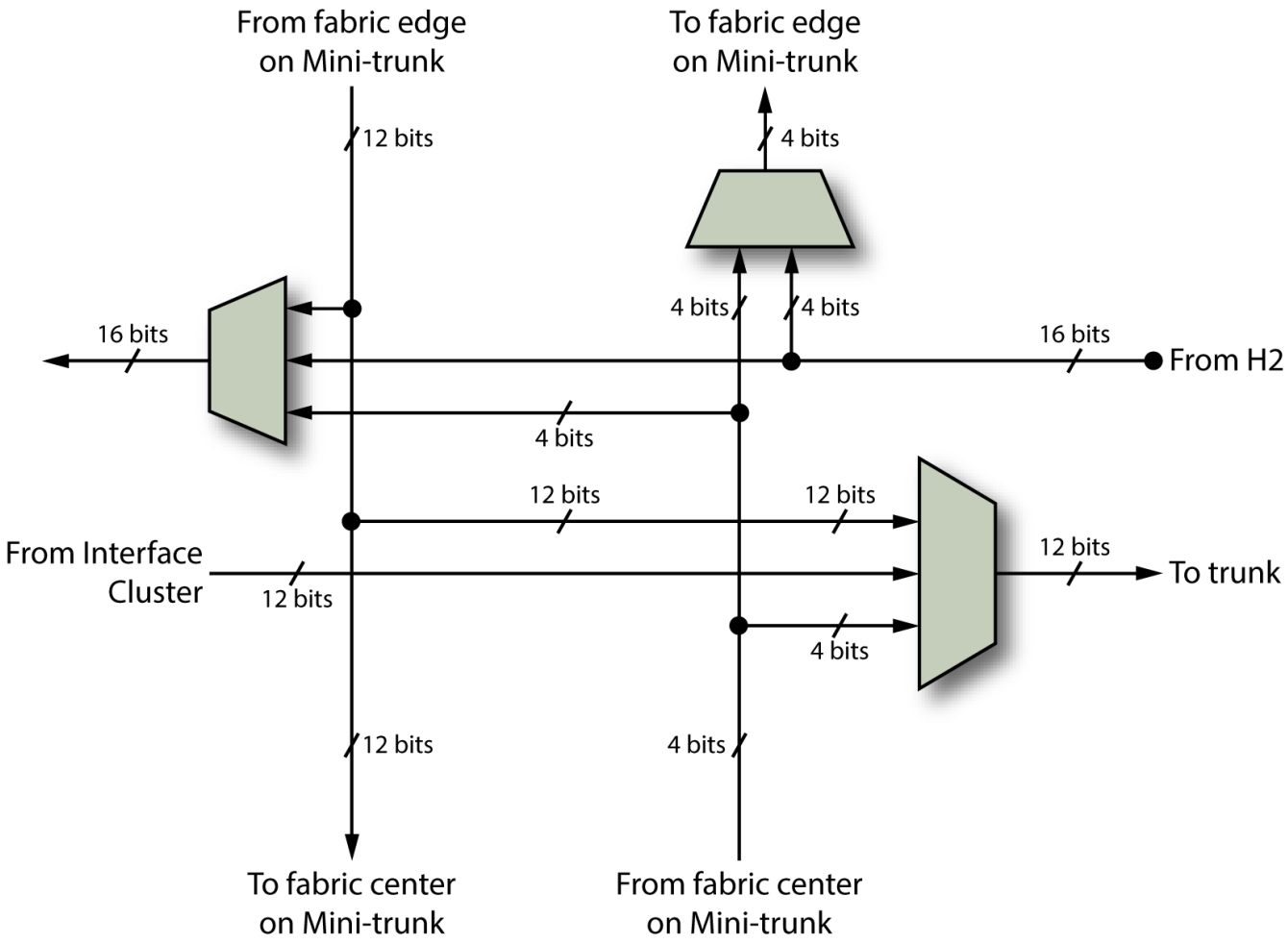


Figure 15: ***Detailed View of a Mini-trunk and Branch Hub on the North-West Side of the Device***

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, users must ensure that the data and clock input/output pins are used for their respective functionality only. In other words, users should ensure that:

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

The reason why these requirements are important is because there are differences in connectivity between data and clock pins. Connecting to those whose functions and connectivity are 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. Below is a table listing 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 2: *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

Location	Resets	Enables	Clocks
Clock region with clock routing to additional data pins	–	–	12 unique; up to 8 can be gated or switched
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 that users take advantage of the built-in clock dividers, clock switches, and clock gates.

Chapter - 5: Speedster7t Interface Clocks

Interface clocks are the clocks that enter the Speedster7t 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 mini-trunk 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 mini-trunk clocks, it is highly recommended designs drive logic *only* within the cluster where the mini-trunk clock enters. Keeping logic within the same cluster of the mini-trunk 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 mini-trunk. 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 light gray cluster and also the same clocks to the adjoining yellow cluster. The area shown in purple is reachable only by the south-east mini-trunk clocks.

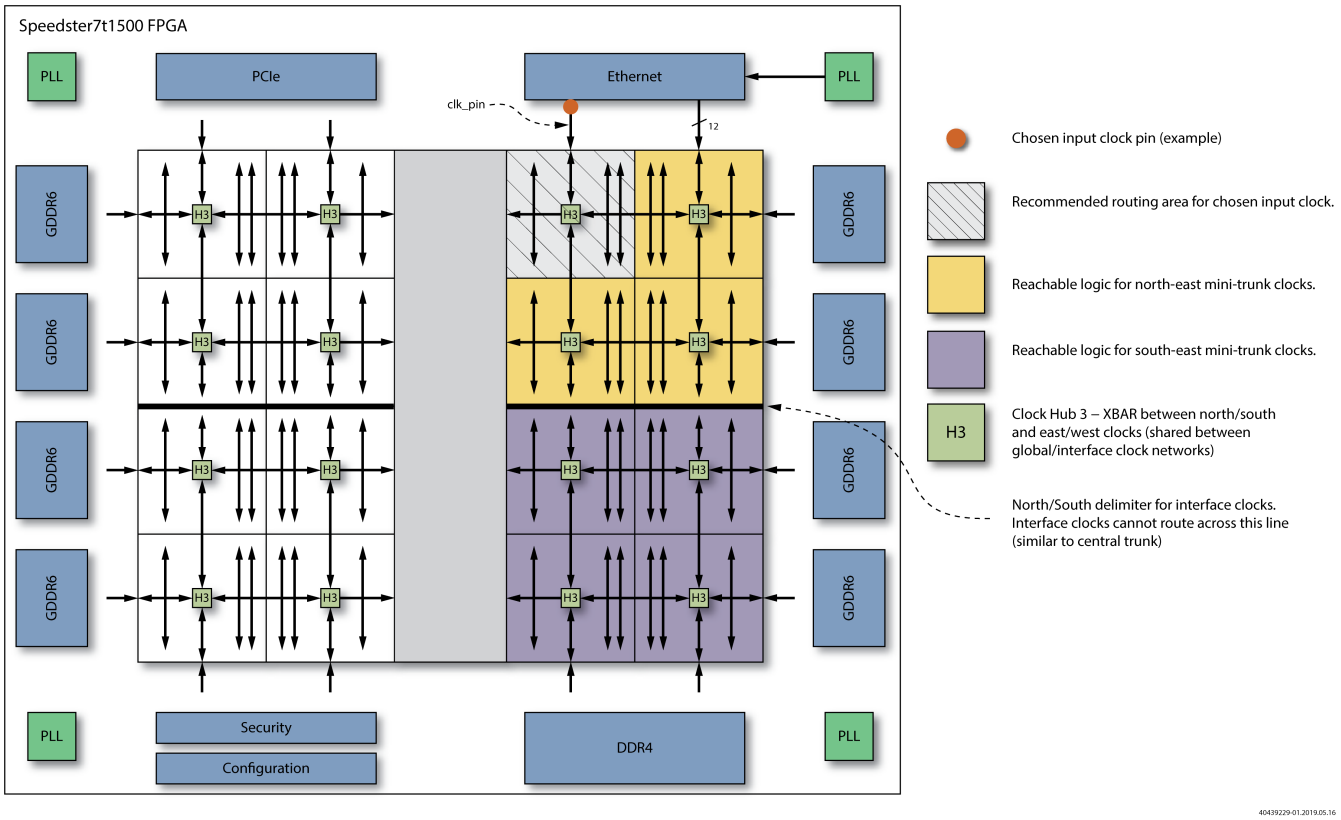


Figure 16: ***PLL Driving a Clock Mini-trunk***

Similar to mini-trunk 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 interface.

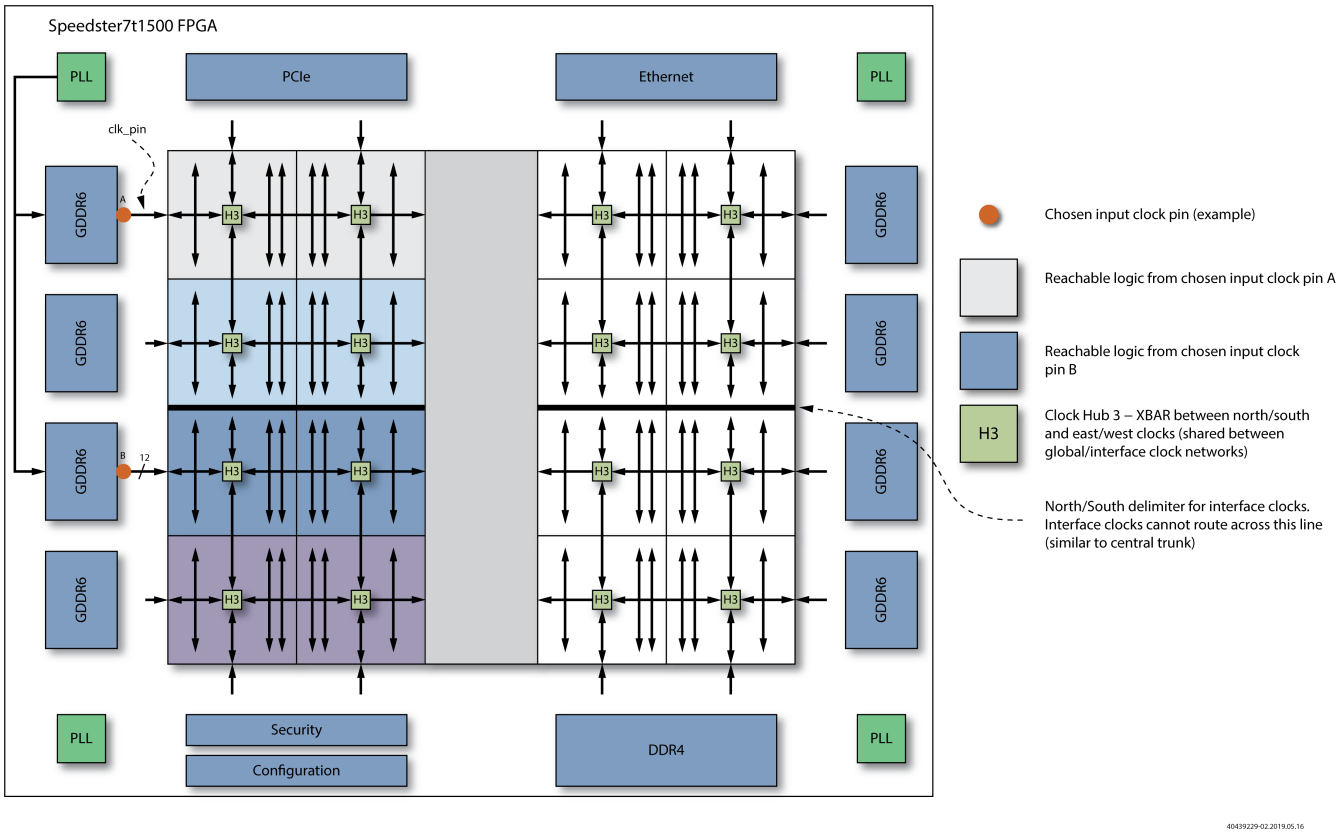


Figure 17: ***PLL Driving Multiple Branch Clocks***

Chapter - 6: Speedster7t Clock Setting and Reporting

Much of the decision making and optimization for clock selection is automatically done by ACE software to prevent no-routes. However, ACE does provide users with some options to specify the type of clock networks they wish 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 (boundary, mini-trunk, or trunk). For data-generated clocks, or data signals routed on the clock network, the designer can specify how to route the signal (local, regional, or center).

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

```
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 (see page 26).
-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 (see page 26).
-trunk	To specify a clock entering the core through a clock hub zero (H0) (see page 15). See Figure: Global Core Clock Network (see page 15).
-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) (see page 18) (drives an entire clock region).
-data_center	To specify a clock source coming from the fabric to drive a clock hub one (H1) (see page 18) 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 route 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.

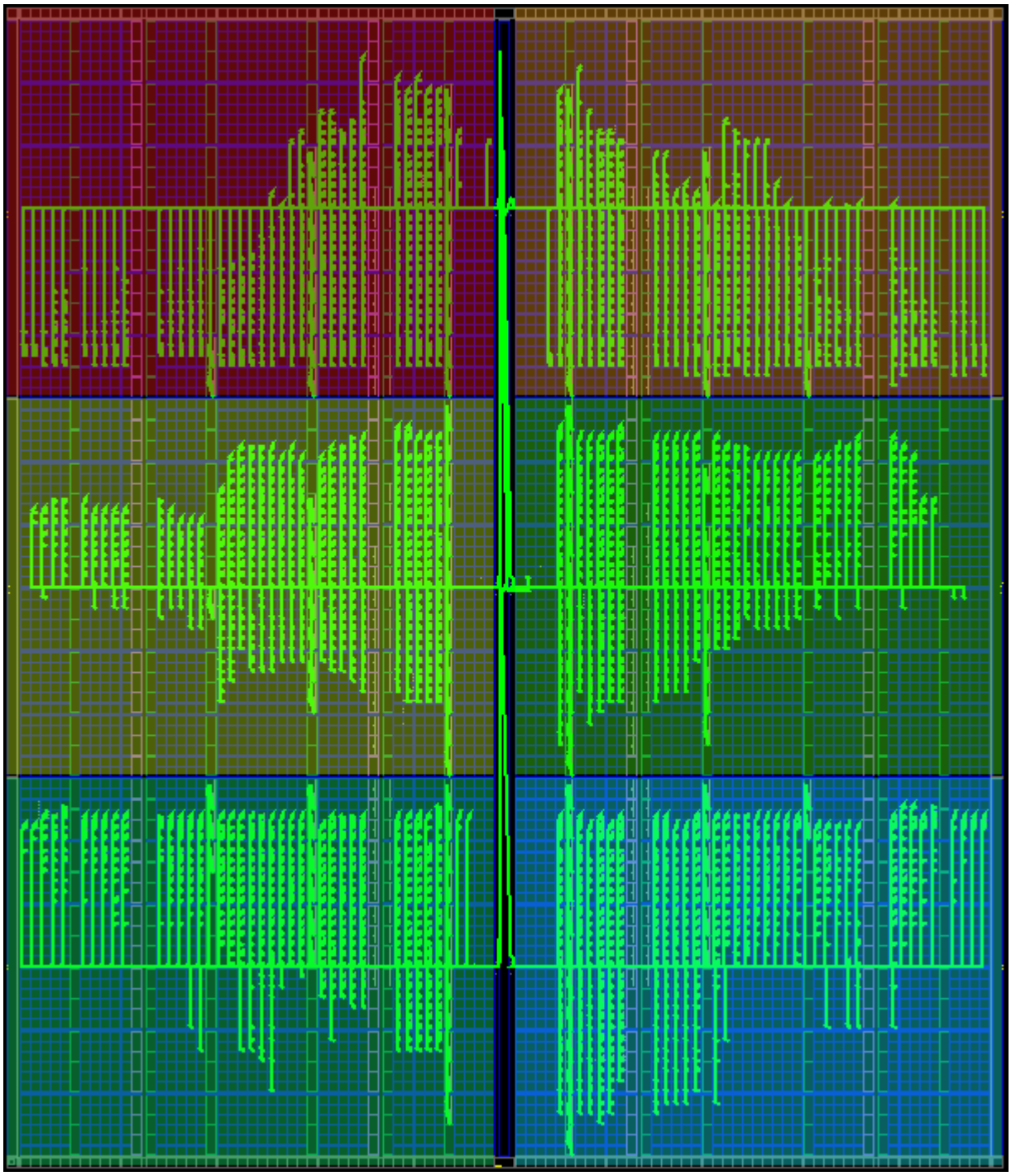


Figure 18: *Reset (Data Signal) Routed on a Clock Trunk*

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`.

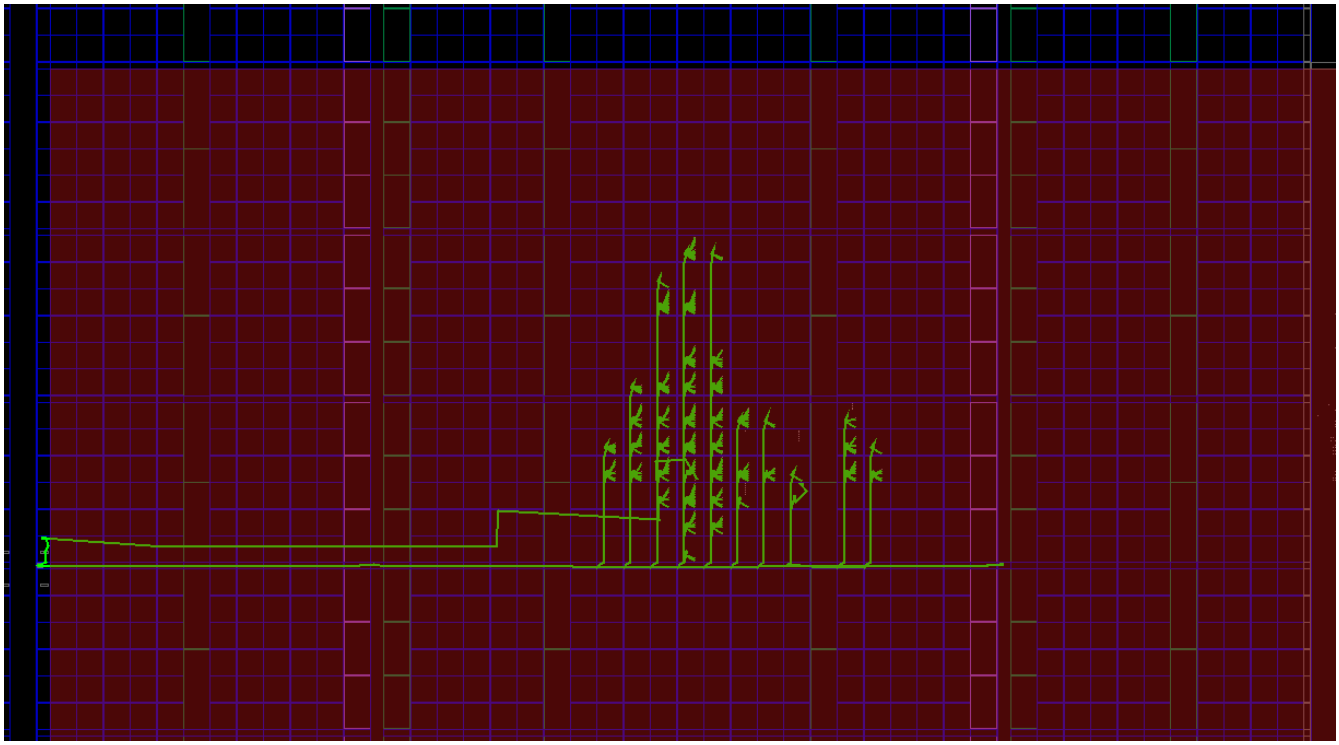


Figure 19: ***Reset (Data Signal) Routed on Regional Clock Network***

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.

Below is a figure showing a clock signal derived in the data fabric and routed on the clock network, using the option `-data_local`.

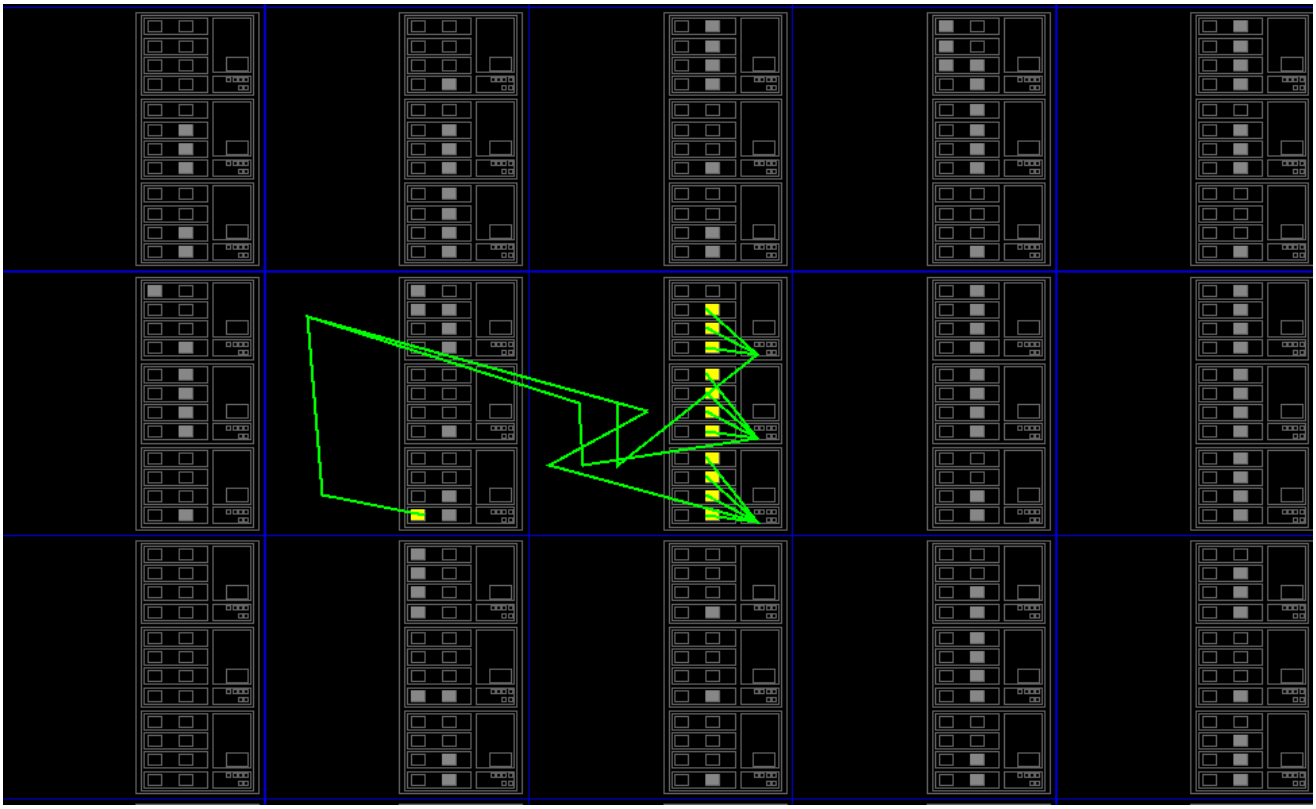


Figure 20: ***Data-Derived Clock Routed on Local Clock Network***

Clock Reporting

Once the clock constraints are specified in the design, and the design run through a full ACE compilation flow, the routing step outputs text and HTML files called `{design_name}_clocks_routed` describing 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

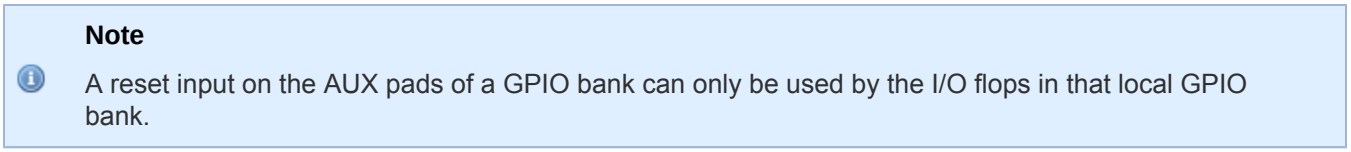
Each Speedster7t FPGA has a global reset network used to send reset to the entire device. There are three variations on how a design can reset the different portions of the device.

- 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.
- 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. If the user wishes to control their own reset for the 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. With six clock I/O available in banks in each corner, there are up to 24 I/O that can be used as either clock or reset signals. 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.

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. If the user wishes to send the same reset to multiple interface subsystems, the user must replicate the reset signal in the design and use the FPGA core to route the reset signal out to each interface subsystem.

The GPIO banks are a special case. For GPIO configurations that require a reset for a flopped I/O, the reset for the given bank comes from a reset input from one of the clock I/O banks, or from a local reset driven by one of the two AUX pads in the GPIO bank. The AUX pads in the GPIO bank can be used either for data, or as a local reset signal to reset the flops in that GPIO bank. If the AUX pad is configured as an input, it inputs a local reset that can be used by the GPIO flops. If the AUX pads are configured as an output, the reset is driven out by the FPGA fabric core.



Note

A reset input on the AUX pads of a GPIO bank can only be used by the I/O flops in that local GPIO bank.

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.

Revision History

Version	Date	Description
1.0	22 May 2019	• Initial Achronix release.
1.1	06 May 2020	• Added significant details to Speedster7t Clock Generation and PLLs (see page 9) . • Minor updates to Speedster7t Global Core Clock Network (see page 15). • Updated figures and text in Speedster7t Interface Clocks (see page 26). • Updates and additional details added to Speedster7t Reset Network (see page 33) .