PolarFire Family Clocking Resources

Introduction (Ask a Question)

This user guide describes the clocking resources and their components available in the PolarFire® family. The FPGA fabric is common to the PolarFire family, which consists of the following FPGA devices.

PolarFire FPGAs	Microchip's PolarFire FPGAs are the fifth-generation family of non-volatile FPGA devices, built on state-of-the-art 28 nm non-volatile process technology. PolarFire FPGAs deliver the lowest power at mid-range densities. PolarFire FPGAs lower the cost of mid-range FPGAs by integrating the industry's lowest power FPGA fabric, lowest power 12.7 Gbps transceiver lane, built-in low power dual PCI Express Gen2 (EP/RP), and on select data security (S) devices, an integrated low-power crypto co-processor.
PolarFire SoC FPGAs	Microchip's PolarFire SoC FPGAs are the fifth-generation family of non-volatile SoC FPGA devices, built on state-of-the-art 28 nm non-volatile process technology. The PolarFire SoC family offers the industry's first RISC-V based SoC FPGAs capable of running Linux®. It combines a powerful 64-bit 5x core RISC-V® Microprocessor Subsystem (MSS), based on SiFive's U54-MC family, with the PolarFire FPGA fabric in a single device.
RT PolarFire FPGAs	Microchip's RT PolarFire FPGAs combine our 60 years of space flight heritage with the industry's lowest-power PolarFire FPGA family to enable new capabilities for space and mission-critical applications. RT PolarFire FPGA family includes RTPF500T, RTPF500TL, RTPF500TS, RTPF500TLS, RTPF500ZT, RTPF500ZTL, RTPF500ZTS, and RTPF500ZTLS devices.
RT PolarFire SoC FPGAs	Designed to enable high-performance data processing, our radiation-tolerant PolarFire SoC FPGA is the industry's first embedded, real-time, Linux-capable, RISC-V-based Microprocessor Subsystem (MSS) on the flight-proven RT PolarFire FPGA fabric. With our extensive Mi-V ecosystem, designers can develop lower-power solutions for the challenging thermal environments seen in space. RT PolarFire SoC FPGA family includes RTPFS160ZT, RTPFS160ZTL, RTPFS160ZTS, RTPFS160ZTLS, RTPFS460ZT, RTPFS460ZTL, RTPFS460ZTS, and RTPFS460ZTLS devices.

The following table lists the clocking resources of the PolarFire family.

Table 1. Clocking Resources

Clocking Resources	PolarFire® FPGA (MPF)	PolarFire SoC FPGA (MPFS)	RT PolarFire FPGA (RTPF)	RT PolarFire SoC FPGA (RTPFS)
Clock Routing Resources	✓	✓	✓	✓
On-chip Oscillators	✓	✓	✓	✓
Clock Conditioning Circuitry	✓	✓	✓	✓
MSS Clock Controller	—	✓	—	✓

References (Ask a Question)

- For information about transceiver clock regions, see PolarFire Family Transceiver User Guide.
- For information about I/O interface modes and high-speed I/O clock networks, see PolarFire Family I/O User Guide.
- For information about device power-up, see PolarFire Family Device Power-Up and Resets User Guide.
- For information about the performance of the global clock for the transceiver quads, minimum setup and hold time for the clock gating enable signal, electrical characteristics of the on-chip oscillators, and PLL jitter

performance, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet

- For information about using Libero® SoC for PolarFire FPGA and PolarFire SoC FPGA, see Libero SoC Documentation.
- For information about MSS, see PolarFire SoC FPGA MSS Technical Reference Manual.

Table of Contents

Introduction....1
References....1

1. Clocking Features Overview....4
   1.1. Clock Routing Resources....4
   1.2. On-Chip Oscillators....4
   1.3. Clock Conditioning Circuitry....4
   1.4. MSS Clock Controller (For PolarFire SoC and RT PolarFire SoC FPGA Only)....4

2. Clock Routing Resources....6
   2.1. Global Clock Network....6
   2.2. Regional Clock Networks....9
   2.3. High-speed I/O Clock Networks....12
   2.4. Preferred Clock Inputs....13
   2.5. Interface Clock Block....15
   2.6. Clock Gating....18
   2.7. Clock Macros....19
   2.8. Managing Global Signals....20
   2.9. Global Net Clock Jitter....23

3. On-Chip Oscillators....25

4. Clock Conditioning Circuitry....27
   4.1. Features....27
   4.2. CCC Locations and Clocking Capabilities....28
   4.3. Phase-Locked Loops....29
   4.4. Delay-Locked Loops....42
   4.5. PLL/DLL Cascading....48
   4.6. CCC Configuration....49
   4.7. CCC Simulation Support....59
   4.8. PLL/DLL Placement....59
   4.9. Dynamic Configuration of CCC....60

5. MSS Clock Controller (For PolarFire SoC and RT PolarFire SoC FPGA Only)....64
   5.1. MSS Clocks....64
   5.2. FPGA Fabric Interface Clocks....65

6. Revision History....67

Microchip FPGA Support....70

Microchip Information....70
Trademarks....70
Legal Notice....70
Microchip Devices Code Protection Feature....71

1. Clocking Features Overview (Ask a Question)

Clocking capabilities are crucial to FPGA architectures. The PolarFire family includes an abundant selection of robust clocking resources, classified as:

- Clock routing resources
- On-chip oscillators
- Clock Conditioning Circuitry (CCC): PLLs and DLLs
- MSS Clock Controller (for PolarFire SoC and RT PolarFire SoC FPGA only)

1.1. Clock Routing Resources (Ask a Question)

To enable efficient clock distribution, the PolarFire family has a global clock network, regional clock networks, high-speed I/O clock networks, and preferred clock inputs and outputs.

- **Global Clock Network** is used to distribute high fan-out signals, such as clocks and resets across the FPGA fabric with low skew.
- **Regional Clock Networks** are low-latency networks that distribute clocks only to a specific designated area based on the driving source. Regional clock networks are used to move data in and out of the fabric.
- **High-Speed I/O Clock Networks** are used to distribute high-speed clocks along the edge of the device to service the I/Os. High-speed I/O clock networks are used to implement high-speed interfaces.
- **Preferred Clock Inputs** have access to the global clock network and/or CCCs through low-latency paths. Preferred clock input pins are recommended for connecting external clocks to the clock inputs of Phase-Locked Loops (PLLs), Delay-Locked Loops (DLLs), and fabric logic. While it is possible to use regular I/Os as clock inputs, doing so introduces high latency on the path.
- **Preferred Clock Outputs** are used to connect PLL clock outputs to external components. Preferred clock output pins have low-latency routing from the PLL clock outputs.

Important: The PolarFire family offers 24 full-chip or 48 half-chip global signals, up to 101 regional signals, and six high-speed I/O signals per I/O bank.

1.2. On-Chip Oscillators (Ask a Question)

The PolarFire family offers two independent on-chip RC oscillators (2 MHz and 160 MHz) for generating free-running clocks.

1.3. Clock Conditioning Circuitry (Ask a Question)

Embedded in each corner of a device is a CCC block containing two PLLs and two DLLs that provide flexible clock management and synthesis capabilities. The CCCs perform the following functions:

- Frequency synthesis (integer and fractional)
- Spread-spectrum clock generation
- Clock delay and phase adjustment
- Clock duty-cycle correction

1.4. MSS Clock Controller (For PolarFire SoC and RT PolarFire SoC FPGA Only) (Ask a Question)

The PolarFire SoC and RT PolarFire SoC FPGA MSS have a dedicated clock controller for generating clocks to all the MSS sub-blocks for correct operation and synchronous communication with the

user logic in the FPGA fabric. For information about MSS, see PolarFire SoC FPGA MSS Technical Reference Manual.

2. Clock Routing Resources (Ask a Question)

The PolarFire family contains low-skew clock networks and preferred clock inputs and outputs for efficient clock distribution.

2.1. Global Clock Network (Ask a Question)

The global clock network is used to distribute high fan-out signals such as clocks and resets across the FPGA fabric with low-skew, using a vertical and horizontal clock stripe architecture.

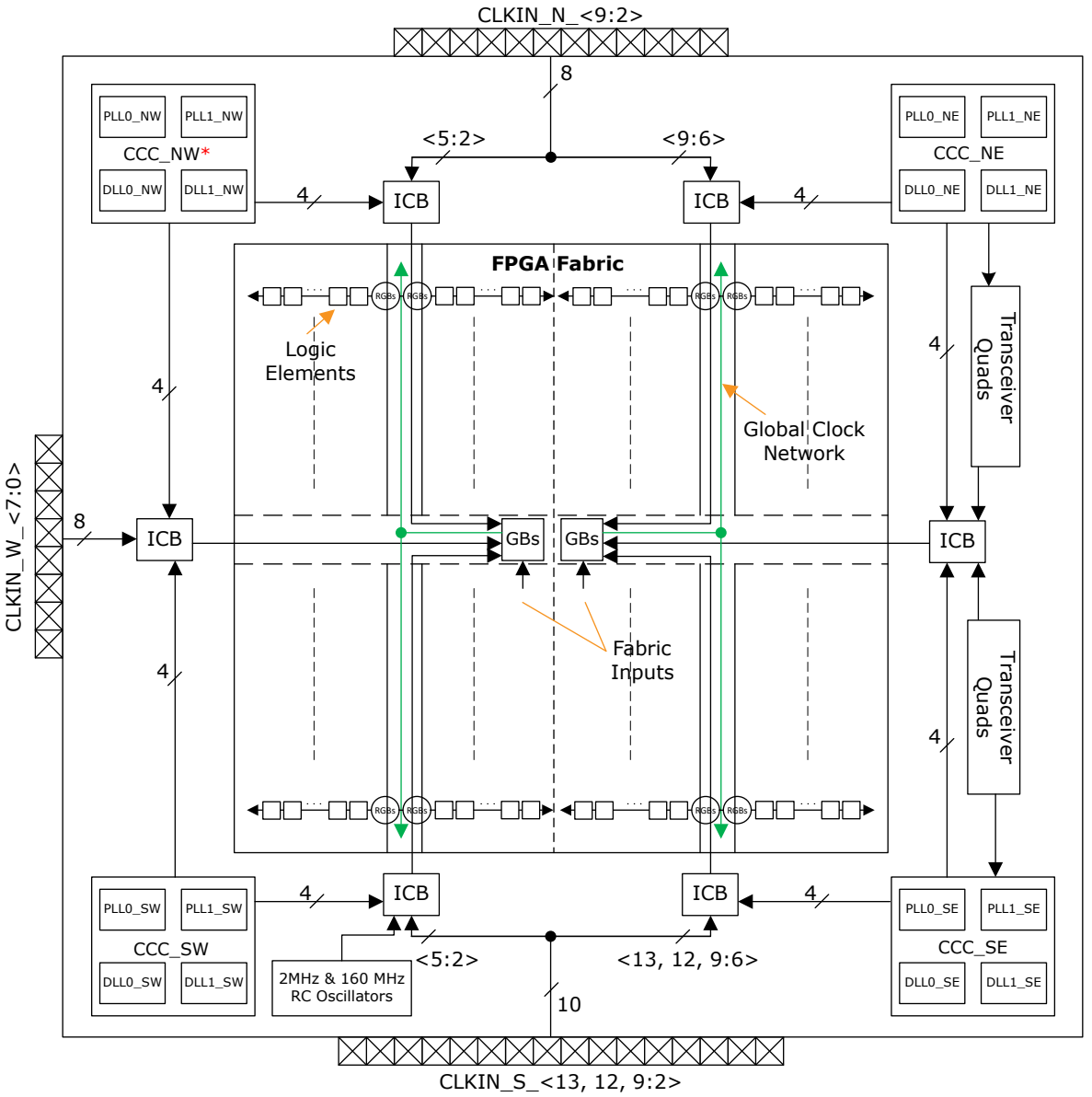
The following figure shows the global clock network architecture and the possible clocks that are routed on the network. Two vertical clock stripes are positioned at approximately one quarter and three quarters of the FPGA fabric width. A horizontal clock stripe runs across the middle of the FPGA fabric.

The global clock network can be driven by any of the following:

- Preferred clock inputs (CLKIN_z_w)
- On-chip oscillators
- CCC (PLL/DLL)
- Fabric routed signals
- Clock dividers
- NGMUXs
- Transceiver interface clocks

Important: Only one global clock is supported per transceiver quad. For more information about the performance of the global clock for the transceiver quads, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

Figure 2-1. Global Clock Network and Clock Sources



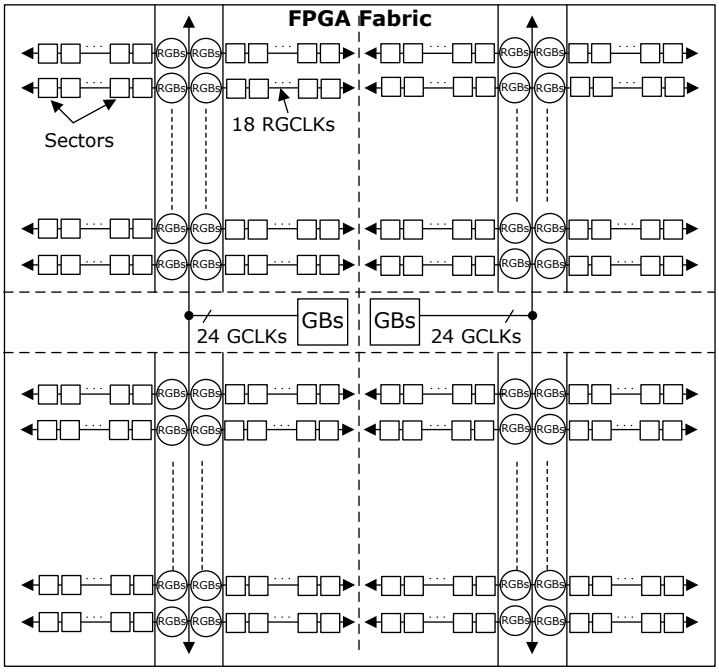
* *For PolarFire® SoC and RT PolarFire SoC FPGAs, there is no feeder line from CCC_NW going to west ICB. Instead the CCC_NW can generate reference clock to MSS.*

The global clock network is composed of Global Buffers (GBs) for clock distribution. As shown in the following figure, there are 48 GBs—24 GBs distribute clocks to the left half of the fabric and the remaining 24 GBs distribute clocks to the right half through vertical clock stripes.

Each GB drives an independent half-chip global clock (GCLK). Two GBs—one from each half—are instantiated by the Libero SoC to distribute a clock to the entire FPGA fabric. A design can have a maximum of 24 full-chip global signals or 48 half-chip global signals. Up to 24 fabric routed signals can drive the half-chip globals.

Clocks driven from regular I/Os, internally generated clocks, and high fan-out signals, such as resets can be routed to GBs using a CLKINT macro.

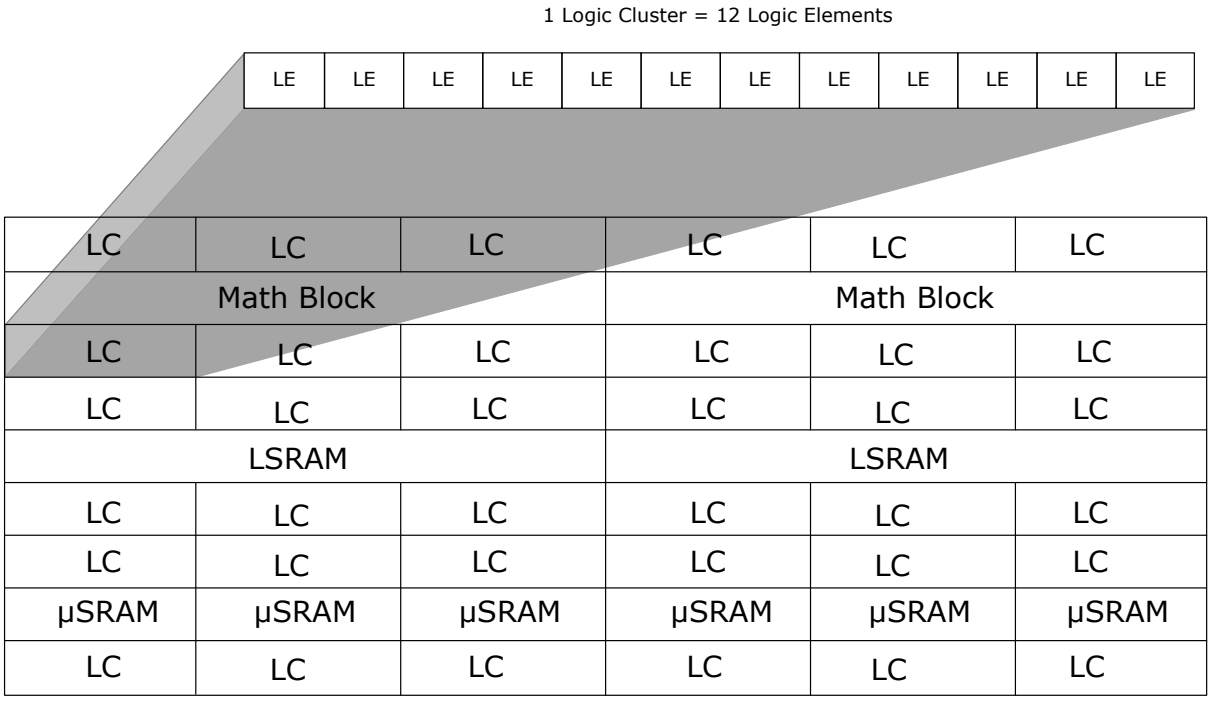
Figure 2-2. FPGA Fabric—Global Clock Routing Architecture



Each GB drives Row Global Buffers (RGB) present on the vertical clock stripes to reach the logic sectors. Each RGB selects a global clock, regional clock, or fabric routed clock to drive the logic sectors located on the left or right side of the vertical clock stripe.

As shown in Figure 2-2, the logic clusters are organized in a repeated pattern of sectors. A row of sectors is divided into four quarters. Each sector consists of six logic-cluster columns and nine logic-cluster rows including two Math blocks, two LSRAM blocks, and six µSRAM blocks, as shown in the following figure. Each logic-cluster contains 12 Logic Elements (LE), each LE consisting of a four-input LUT and D flip-flop.

Figure 2-3. Sector Representation



A set of 18 RGBs drive each quarter of every row of sectors on both sides of the vertical stripe. Each RGB generates a row global clock (RGCLK). Two RGBs—one from either side—must be driven from the same clock source to distribute the clock in both directions, to serve a region of half-row sectors. Up to eight fabric routed signals can drive the RGB resources that serve a half-row of sectors.

Each global and row global buffer has a gating option for glitch-free enabling and disabling of the clock, for more information, see Clock Gating. Up to seven fabric routed signals can gate the RGB resources that serve a half-row of sectors.

A RCLKINT or RGCLKINT macro is used to drive the RGB to create a local clock spanning a small fabric area.

If users do not specify the global clock network assignments, then synthesis tools assign a clock network based on the predetermined priorities. These priorities are primarily set by the fan-out of each signal. Synthesis tools attempt to assign the low-skew resources to clock signals with high fan-out. To improve the performance, users can take control of the clock network assignment by instantiating the required macros or attributes in the design. For a list of macros supported in the PolarFire family, see Clock Macros.

If there are more than 24 global clock signals in the design, the Libero SoC creates half-chip global clocks and splits the placement of the logic accordingly.

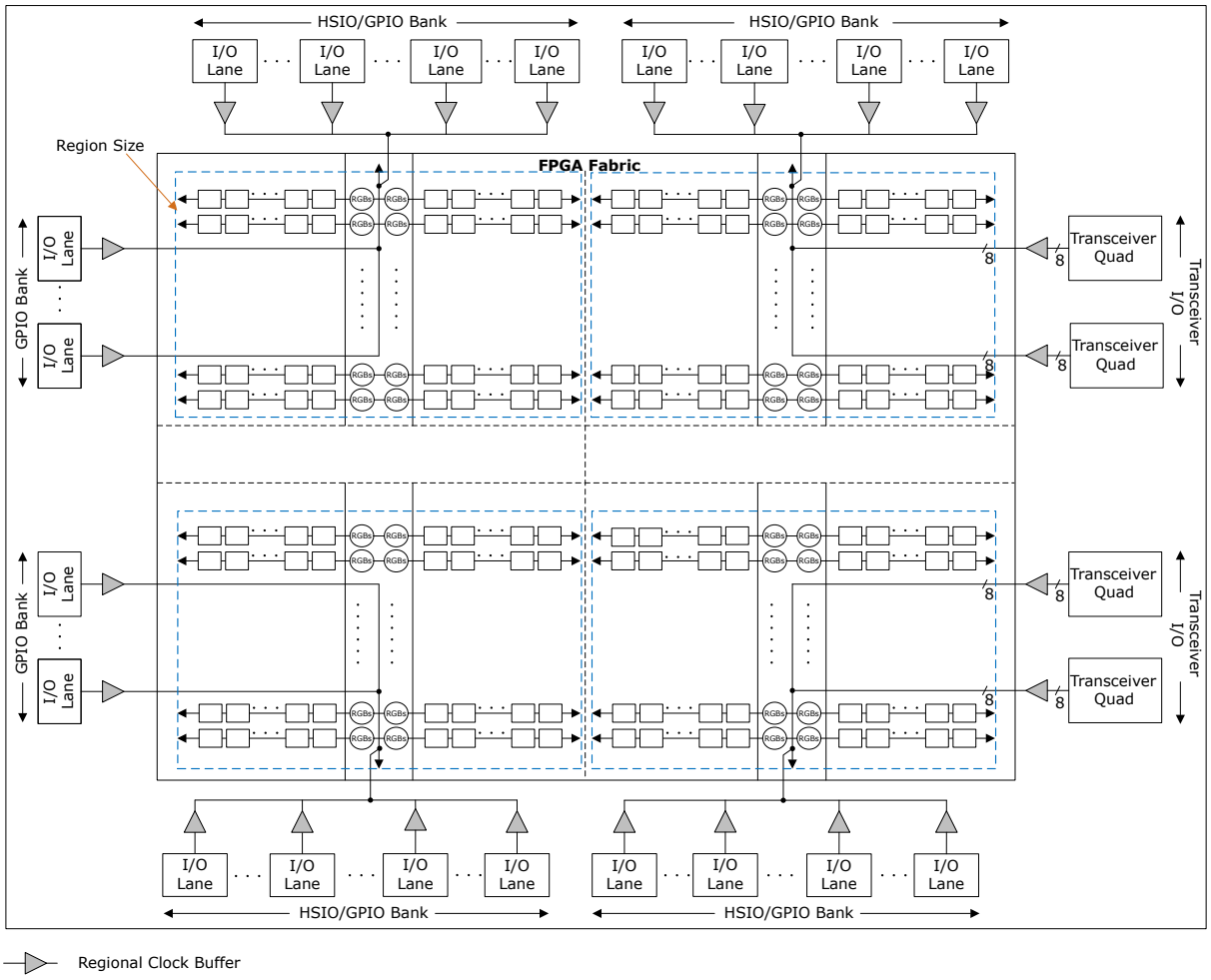
2.2. Regional Clock Networks (Ask a Question)

Regional clock networks exhibit lower latency than the global clock network. They comprise regional clock buffers (RCBs) that interface with the I/O and transceiver lanes at regular intervals as follows:

- One regional clock buffer per I/O lane (a grouping of 12 I/Os) on the north, south, and west edges.
- Two regional clock buffers per transceiver lane (eight per transceiver quad) on the east edge.

A regional clock buffer can be driven from either an I/O lane or the transceiver lane. The nets driven by the regional clock buffers are referred to as regional clocks (RCLKs). The following figure shows the location of the regional clock buffers.

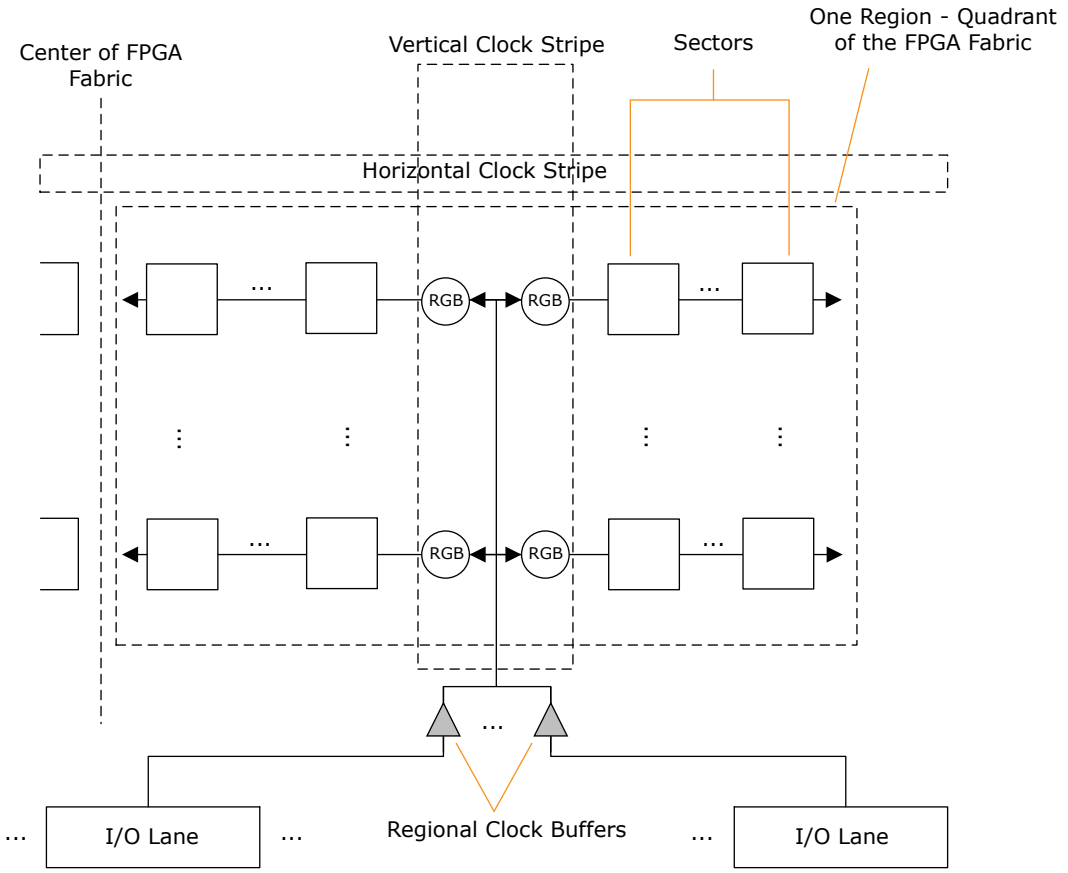
Figure 2-4. Regional Clock Buffers



Each regional clock buffer drives all the RGBs present within its region to distribute the clock. Each regional clock buffer serves a region of a fixed number of sectors. The region size depends on the location of the regional clock buffer and transceiver quad capabilities. The regions cover the entire fabric without overlapping. Regional clocks cannot be aggregated to span large regions. If a clock is required to span multiple regions, then a GCLK must be used.

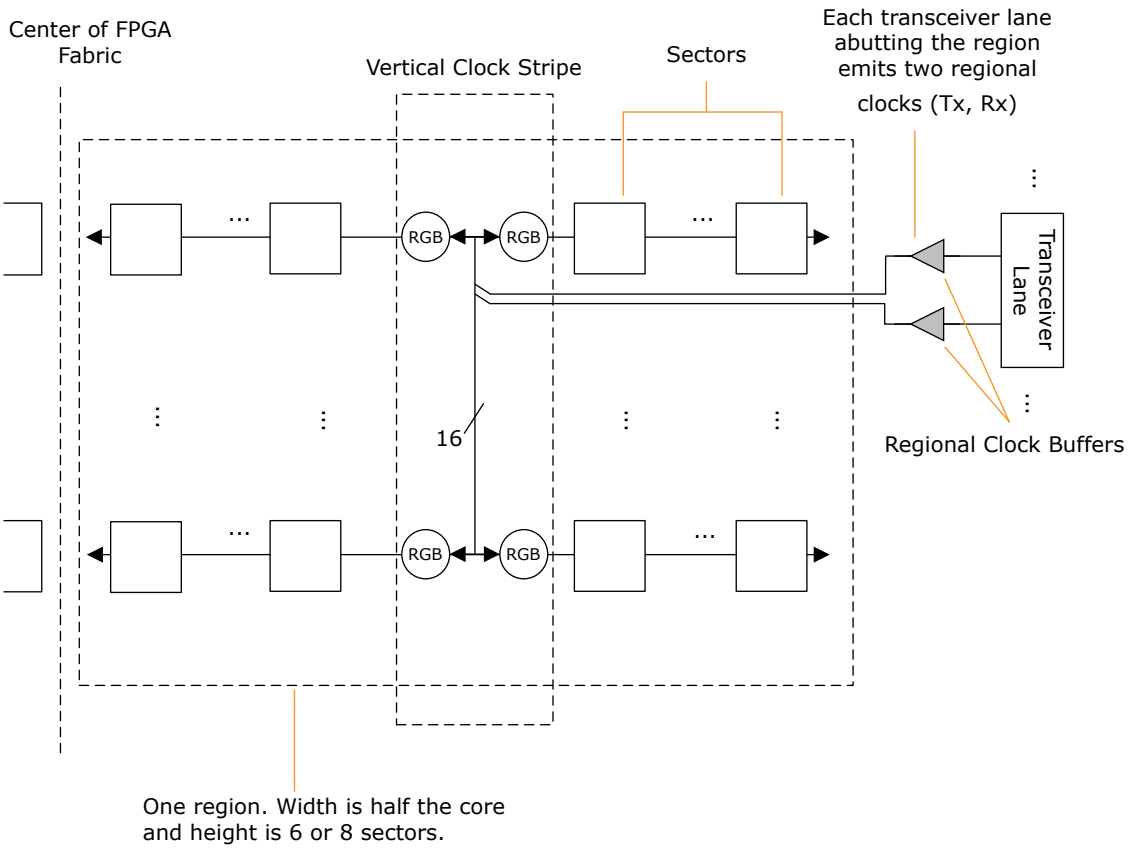
Each regional clock buffer on the north and south edge serves a quadrant of the FPGA fabric. The following figure shows the region served by the regional global buffers using the bottom-right corner of the FPGA fabric as an example.

Figure 2-5. Regional Clock Buffer Fan-outs from Bottom-Right I/O Lanes



Each regional clock buffer on the west and east edge serves a region as high as two transceiver quads and half the width of the FPGA fabric. The region size is six half-row sectors if the quads are without PCIe® controllers. The region size is eight half-row sectors, if one of the quads has PCIe controllers. The following figure shows the region served by the regional clock buffers from a transceiver lane. For more information about region resource details, see PolarFire Family Transceiver User Guide.

Figure 2-6. Regional Clock Buffer Fan-outs from Transceiver



The PolarFire family offers up to 101 regional clock buffers to move data in and out of the fabric.

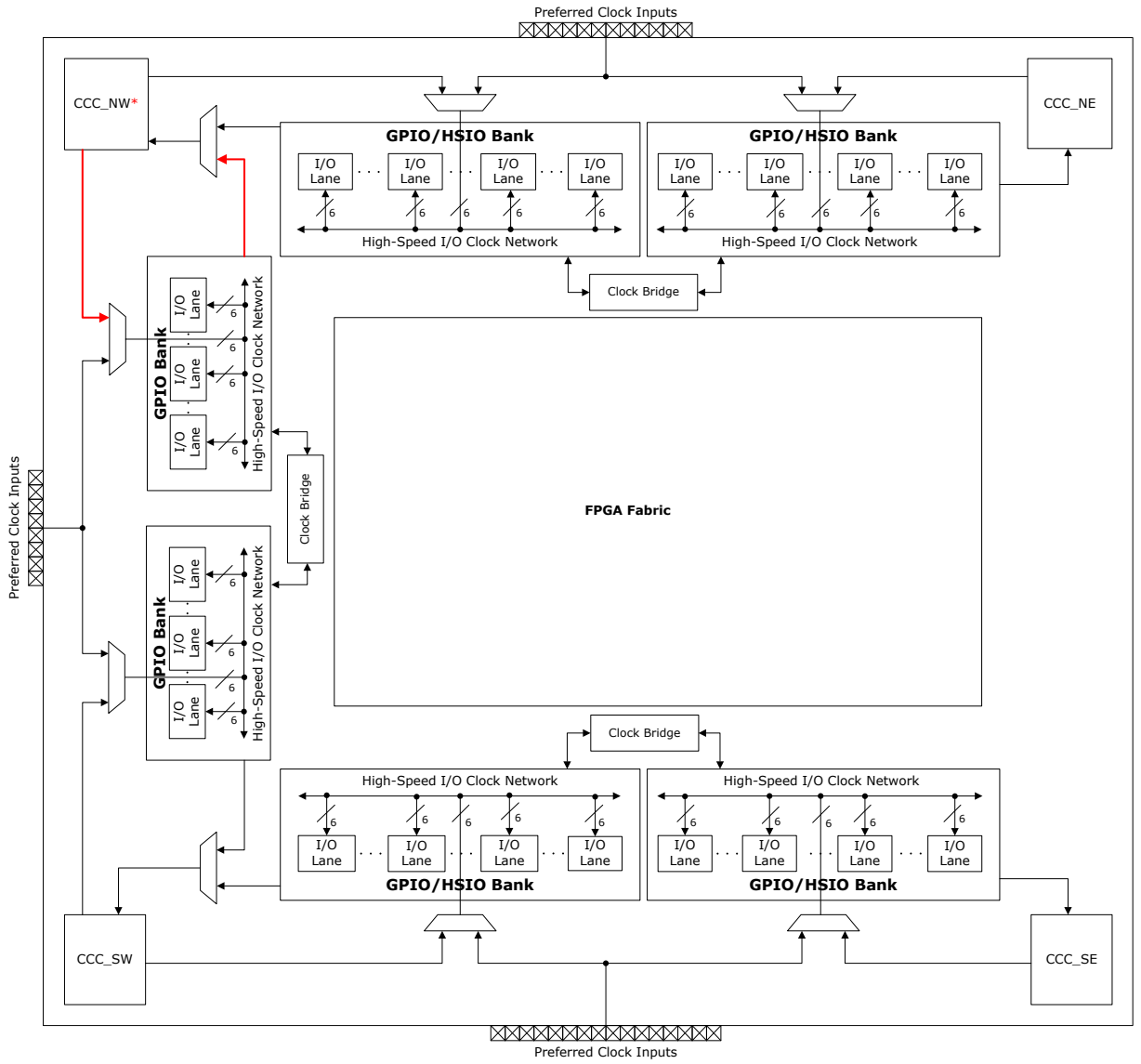
The transceiver interface is configurable to have the lane clocks (TX_CLK and RX_CLK) routed on regional or global clock networks. The I/Os support specific I/O interface modes (for example, DDRxn modes) that are designed to use regional clocks. For more information on I/O interface modes, see the I/O Interface Modes section of PolarFire Family I/O User Guide. When using the I/O interface modes that support regional clocks, Libero SoC automatically routes clocks coming from I/O lanes on the regional clock networks.

2.3. High-speed I/O Clock Networks (Ask a Question)

High-speed I/O clock networks are low-skew high-speed clocks distributed along the edge of the device to service the I/Os. High-speed I/O networks are used to clock data in and out of the I/O logic when implementing the high-speed I/O interfaces. There are no high-speed I/O clock networks located on the east side of the FPGA fabric. Each I/O bank has six high-speed I/O clocks. High-speed I/O clocks from adjacent banks on the same edge can be bridged to build large I/O interfaces.

High-speed I/O clock networks can be driven from I/Os or CCCs. The high-speed I/O clocks can feed reference clock inputs of adjacent CCCs through hardwired connections.

Figure 2-7. High-Speed I/O Clock Networks



* *For PolarFire® SoC and RT PolarFire SoC FPGAs, there is no high-speed I/O clock input going to CCC_NW from west edge and no feeder line from CCC_NW to high-speed I/O clock network on west edge as highlighted.*

The CCC can be configured to have a PLL or DLL clock output driving a high-speed I/O clock network. The I/Os support specific I/O interface modes (for example, DDRxn modes) that are designed to use the high-speed I/O clock networks. When using I/O interface modes that support high-speed I/O clock networks, the Libero SoC automatically routes the clocks coming from I/O lanes on the high-speed I/O clock networks. For more information about high-speed I/O clock networks, see PolarFire Family I/O User Guide.

2.4. Preferred Clock Inputs (Ask a Question)

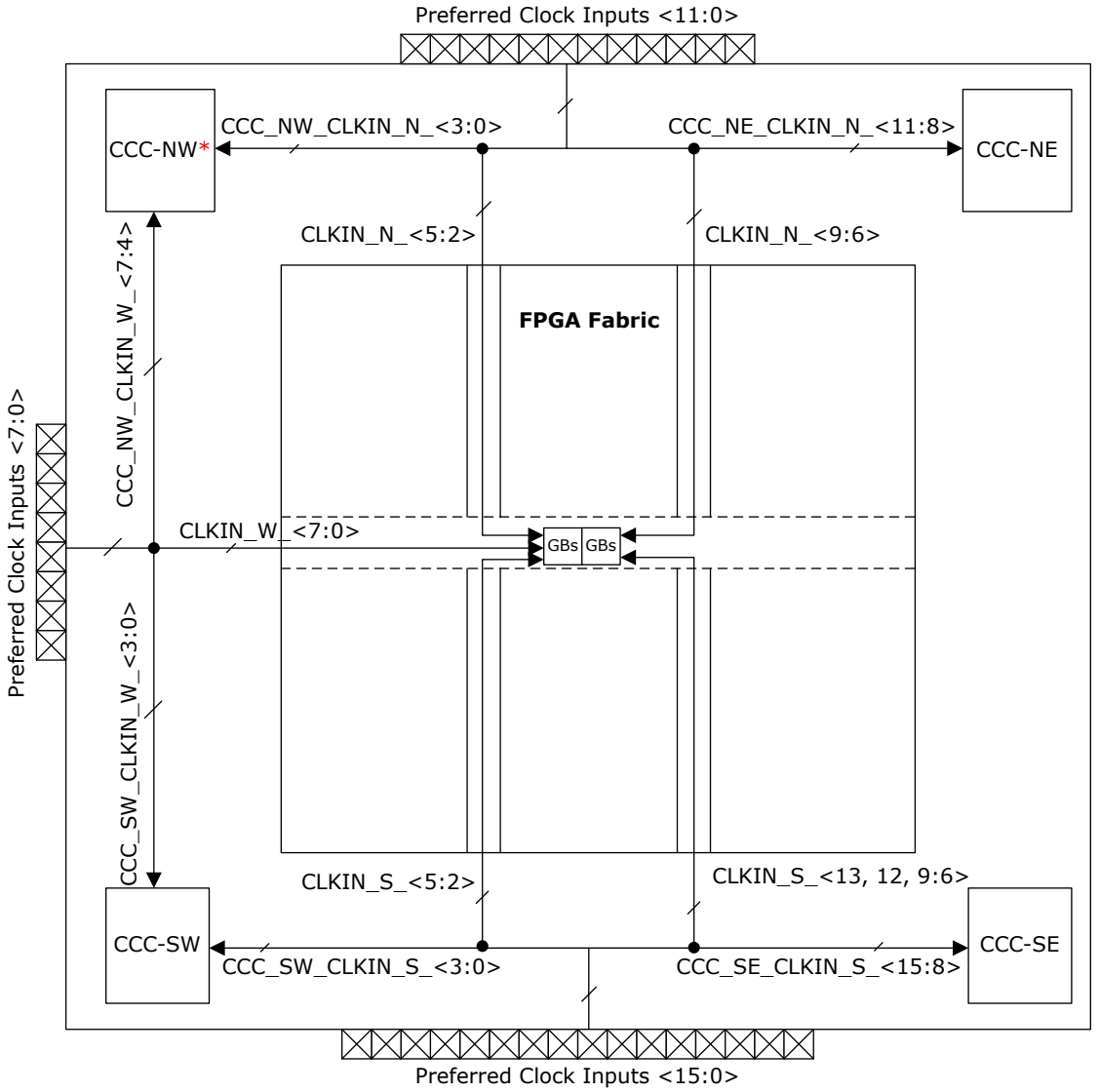
Preferred clock input I/Os connect external clock signals to the CCCs and/or the global clock network through low-latency paths. Use these preferred clock inputs for connecting external clocks to the clock inputs of PLLs, DLLs, and fabric logic. Using regular I/Os as clock inputs introduces high latency on the path.

Each preferred clock input can be used either as a single-ended clock input or be paired with an adjacent I/O to form a differential clock input. Preferred clock inputs, if not utilized for clocking, can be used as regular I/Os.

Preferred clock inputs are located on the north, south, and west sides of the device, with eight on the west side, 12 on the north side, and either 12 or 16 on the south side, depending on the package and device type as shown in the following figure. Some of the preferred clock inputs have access to both CCC and the global clock network and can be connected to either of them.

Note: For PolarFire SoC and RT PolarFire SoC FPGAs, there is no preferred clock input going to CCC_NW from the west edge.

Figure 2-8. Preferred Clock Inputs



** For PolarFire® SoC and RT PolarFire SoC FPGAs, there is no preferred clock input going to CCC_NW from west edge.*

2.4.1. Naming Convention (Ask a Question)

CLKIN_z_w and CCC_xy_CLKIN_z_w represent a preferred clock input that drives a global clock network directly, and an input of one of the CCCs respectively, where:

- xy—represents the CCC location: NE, SE, SW, or NW
- z—represents the location of the preferred clock input: N, W, or S
- w—represents the preferred clock input number: 0 to 15

2.4.2. Preferred Clock Inputs Connectivity in CCCs (Ask a Question)

Preferred clock inputs can be configured as reference clock or feedback clock to the PLLs and DLLs present in each CCC. The preferred clock inputs which are capable of driving CCCs have dedicated connections to clock inputs (reference clock or feedback clock) of PLLs and/or DLLs present in the CCCs. For information about the connectivity of preferred clock inputs to PLLs and DLLs present in a CCC, see the respective PolarFire FPGA Package Pin Assignment Tables, RT PolarFire FPGA Package Pin Assignment Tables, or PolarFire SoC FPGA Package Pin Assignment Tables.

Each CCC consists of two PLLs (labeled as PLL0 and PLL1) and two DLLs (labeled as DLL0 and DLL1). CCCs and their internal PLLs and DLLs are labeled according to their locations in the device. For example, the CCC located in the northeast corner is labeled as CCC_NE. Similarly, the PLLs and DLLs present in the CCC_NE are labeled as PLL0_NE, PLL1_NE, DLL0_NE, and DLL1_NE. Each PLL has two reference clock inputs—REF_CLK_0 and REF_CLK_1.

Important: For PolarFire family, some of the preferred clock inputs have connections to feedback clock input of the PLL/DLL present in the CCC. It is required to choose a preferred clock input, which has connection to the PLL reference clock input for clock frequency synthesis. For information about preferred clock inputs connectivity to PLLs/DLLs and global clock network, see the respective PolarFire FPGA Package Pin Assignment Tables, RT PolarFire FPGA Package Pin Assignment Tables, PolarFire SoC FPGA Package Pin Assignment Tables.

For example, in PolarFire FPGA, the package pin T9 of MPF300T-FCG1152 device has pin name as GPIO219PB4/CLKIN_W_3/CCC_SW_CLKIN_W_3. The T9 pin can be used as a preferred clock input, which can connect to global clock network or CCC_SW. In the CCC_SW, the T9 pin is connected to feedback clock inputs of PLLs and reference clock inputs of DLLs. Therefore, the PLL frequency synthesis cannot be performed on the external clock connected to T9 in the MPF300T-FCG1152 device.

As another example, in PolarFire SoC FPGA, the package pin J13 of MPFS250TS-FCG1152 device has pin name as GPIO9PB1/CLKIN_S_8/CCC_SE_CLKIN_S_8. The J13 pin can be used as a preferred clock input, which can connect to global clock network or CCC_SE. In the CCC_SE, the J13 pin is connected to reference clock inputs of PLLs.

2.5. Interface Clock Block (Ask a Question)

Interface Clock Block (ICB) multiplexes clock inputs from various clock sources (CCCs, preferred clock inputs, High-Speed I/O network, Oscillators, and FPGA fabric) and provides an entry into the global clock network. The east and west edges of the device have one ICB each. The north and south edges of the core have two ICBs each. Each ICB contains four clock dividers, two no-glitch clock multiplexers (NGMUXs), 12 clock gating circuits, and clock routing multiplexers to route clocks. Each ICB has two ICB_INT cells and 12 ICB_CLKINT cells. The ICB_INT cells are needed to route clocks from fabric to ICB. ICB_CLKINT cells are needed to route clocks from ICB to global buffers. If you encounter the limitation of the number of ICB_INT cells per ICB, use dedicated connections from CCCs, preferred clock inputs, and oscillators, which do not require ICB_INT cells. The following sections describe the clock dividers and NGMUXs present in the ICBs.

2.5.1. Clock Dividers (Ask a Question)

The PolarFire family provides 24 programmable clock dividers at the center of the edge on four sides of the device. Libero SoC selects the appropriate clock divider based on the input clock source. The clock divider input can come from any of the following:

- CCCs
- I/Os

- On-chip oscillators
- FPGA fabric

The divider clock outputs drive the global clock network.

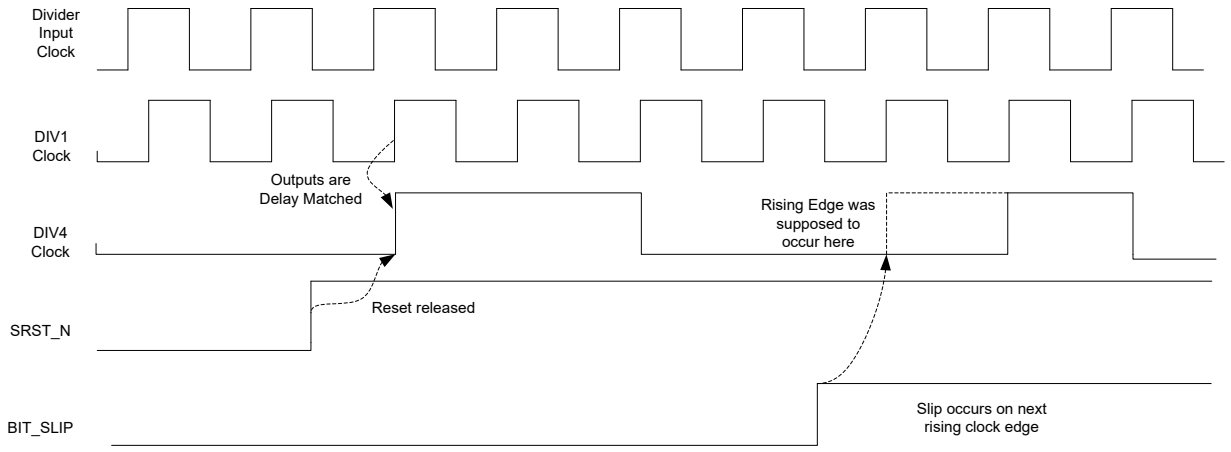
Clock dividers create divide-by-1, divide-by-2, divide-by-3.5, divide-by-4, or divide-by-5 clocks. The divide-by-3.5 and divide-by-5 modes do not generate 50% duty-cycle clock outputs.

Each divider has its own synchronous reset (SRST_N) from the fabric. Resetting the divider takes place on the next falling edge of the input clock after SRST_N is asserted. The dividers come out of reset on the first falling edge of the input clock after SRST_N is de-asserted.

Each divider has a bit-slip (BIT_SLIP) control signal. On the rising edge of the BIT_SLIP signal, one clock pulse is swallowed by the divider circuit. This function is used in various word alignment schemes needed for SGMII and video applications. When the BIT_SLIP signal arrives, it pushes one input clock cycle delay on the divided clock. Depending on the divide mode, bit-slip might happen on the rising edge or falling edge. And, it might happen on the 1st, 2nd, or 3rd divided clock.

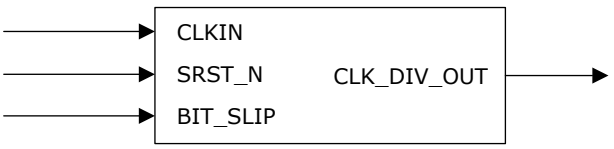
The following figure shows an example of using the reset and bit-slip operation of the dividers. The divide-by-1 clock is not affected by the reset or bit-slip operation.

Figure 2-9. Using the Reset and Slip Operations for the ICB Dividers



The following figure shows the clock divider inputs and outputs. The clock dividers are accessible using CLKDIV configurator. The values for the clock dividers are configurable using the Libero SoC and are programmed during the device programming.

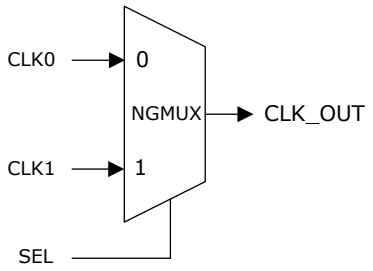
Figure 2-10. Clock Divider



2.5.2. Glitch-Free Clock Switching (Ask a Question)

The PolarFire family has 12 No-Glitch MUXs (NGMUXs) for glitch-free dynamic clock switching between two independent clocks. NGMUXs are located at the center of the edge on four sides of the device. Libero SoC selects the appropriate NGMUX based on the clock source. NGMUX is accessible by instantiating PF_NGMUX configurator in the design. The following figure shows the NGMUX symbol.

Figure 2-11. NGMUX Symbol



The CLK0 and CLK1 inputs to NGMUX can be driven from any of the following:

- Preferred clock inputs
- On-chip oscillators
- Global clock network
- Fabric routing
- CCC (PLL/DLL)
- Clock dividers

The selection control input for each NGMUX (SEL) is driven from the fabric or I/O and can be changed dynamically by the user logic. The selected clock (CLK_OUT) is driven onto the global clock network.

When both current and new clocks are active (Mode 0), glitch-free clock switching happens as quickly as three current clock cycles plus three new clock cycles. The NGMUX can also be configured to support the switch from an uncertain or inactive clock to an active clock (Mode 1). In Mode 1, the clock switching takes up to 50 new clock cycles with a minimal chance of glitch during the switching.

Figure 2-12. Mode 0: Clock Switching when Clocks are Active

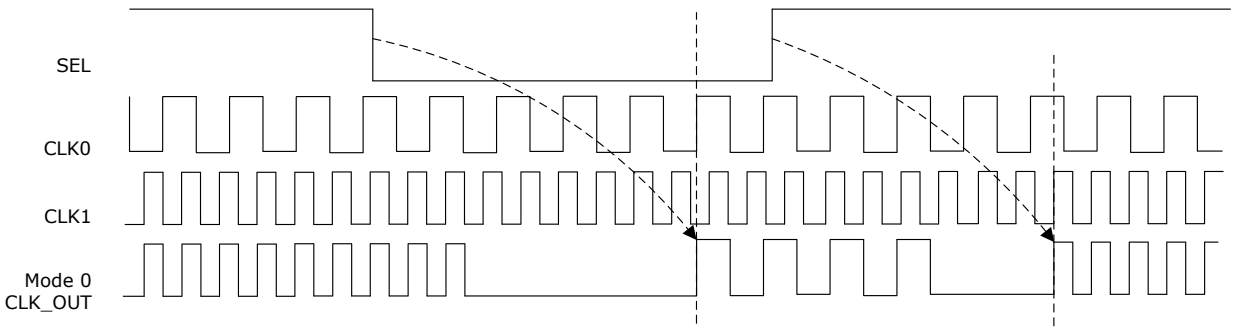


Figure 2-13. Mode 1: Clock Switching when Current Clock (CLK0) is not Active

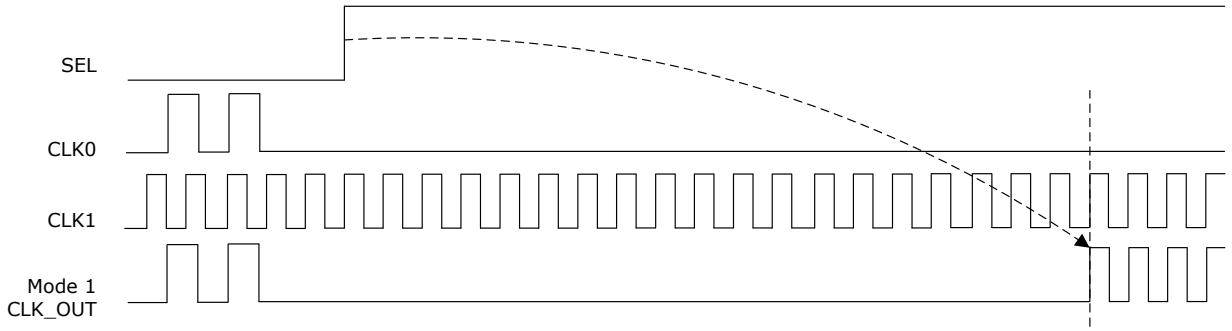
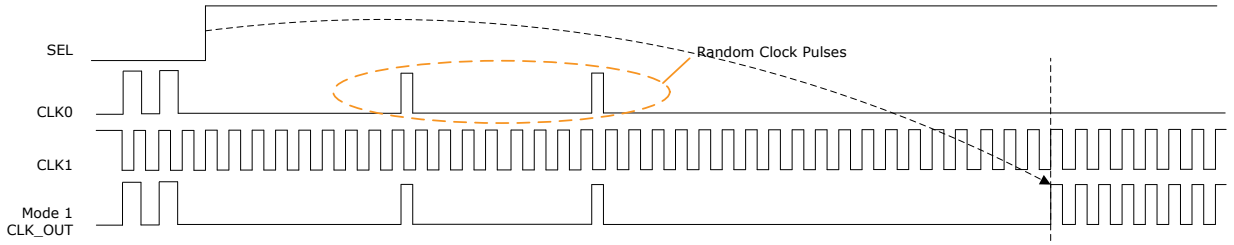


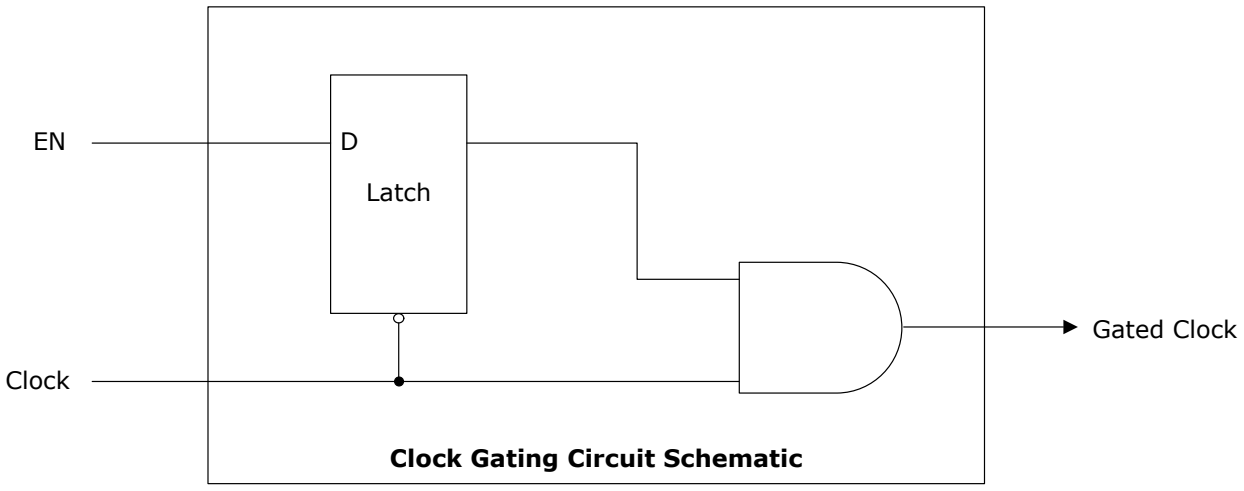
Figure 2-14. Mode 1: Clock Switching when Current Clock (CLK0) is Uncertain with Random Pulses



2.6. Clock Gating (Ask a Question)

Each global and row global buffer has a gating option for glitch-free enabling and disabling of the clock. The clock-gating enable port driven from the fabric logic can be changed dynamically. The clock gating feature is accessible by instantiating a clock buffer macro (GCLKINT or RGCLKINT) in the design. The GCLKINT macro provides clock gating at the global buffer level, and the RGCLKINT macro provides clock gating at the row global buffer level. The following figure shows a schematic of the clock-gating circuit.

Figure 2-15. Clock Gating Circuit Schematic

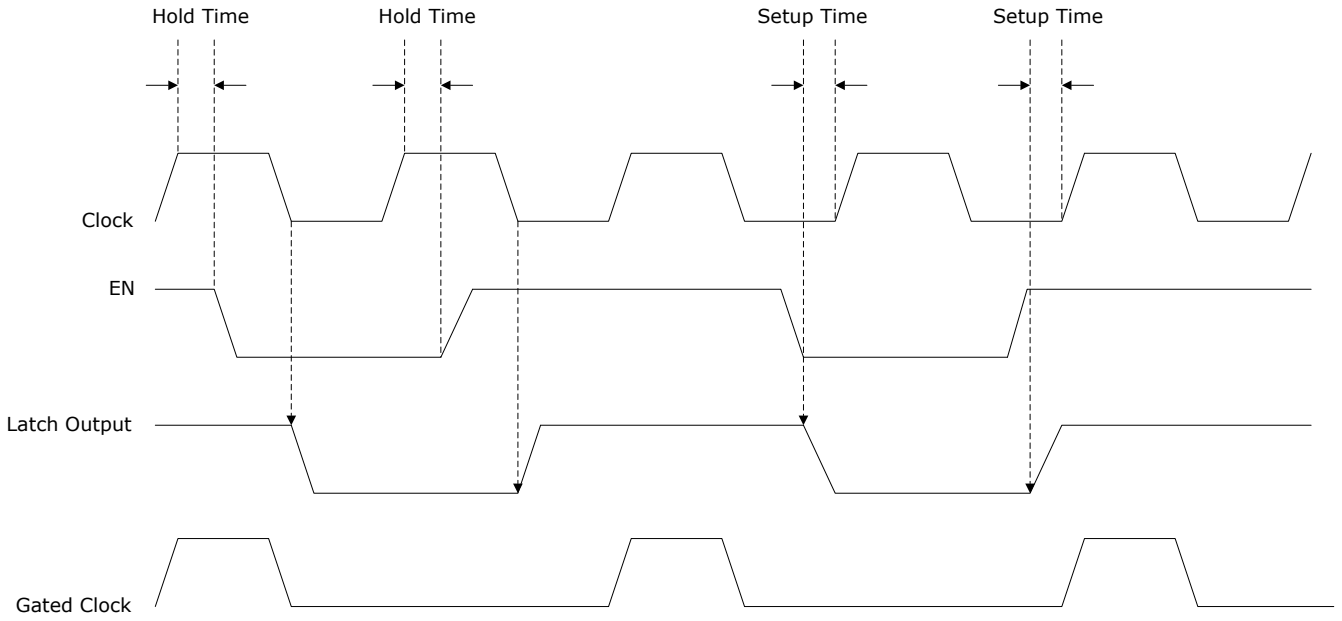


Clock gating is achieved using a latch and enable (EN) that is driven by the user logic implemented in the FPGA fabric. The latch is transparent when the clock input is in the low phase. The latch is

in a Hold state when the clock is in the high phase. The AND gate at the output allows enabling or disabling of the clock based on the latch output. The clock is active when the EN signal is HIGH, and it is gated off and driven LOW when the EN signal is LOW.

The following figure shows the timing waveforms for buffers with clock gating enabled. For the minimum setup and hold time for the clock gating enable signal, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

Figure 2-16. forms for the Clock Gating Circuitry



- If the EN signal changes during the clock high phase and the minimum hold time is met with respect to the prior rising clock edge, then the latch output changes after the falling clock edge.
- If the EN signal changes during the clock low phase and the minimum setup time is met with respect to the next rising clock edge, then the latch output changes immediately.
- If the EN signal violates either the setup or hold time with respect to the rising clock edge, then the output behavior is unknown.

For falling-edge clocks, the EN signal must arrive by the prior rising edge (earlier than usual). Unused global resources such as RGBs and GBs are tied off automatically to reduce dynamic power consumption.

2.7. Clock Macros (Ask a Question)

Clock macros provided in the Libero SoC macro catalog are used to assign signals to the global buffers and row global buffers.

Table 2-1. Clock Macros

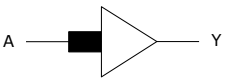
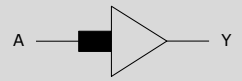
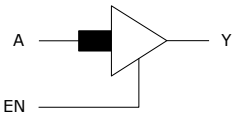
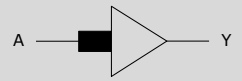
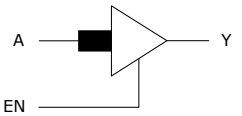
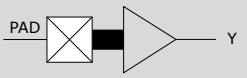
Macro Name	Description	Functional Symbol
CLKINT	Routes clock signal from the fabric routing or regular I/O to the global clock network.	

Table 2-1. Clock Macros (continued)

Macro Name	Description	Functional Symbol
CLKINT_PRESERVE	Similar to CLKINT, but the signals routed through the CLKINT_PRESERVE macro will never be demoted or optimized by the Libero Compile tool.	
GCLKINT	Gated version of CLKINT with an enable signal (EN) from the FPGA fabric to gate the clock.	
RCLKINT	Routes clock signal from global buffers, regional clock buffers, or fabric routing to RGBs.	
RGCLKINT	Gated version of RCLKINT with an enable signal (EN) from the FPGA fabric to gate the clock.	
CLKBUF	Drives a global buffer from a preferred clock input pad.	

2.8. Managing Global Signals (Ask a Question)

Assigning high fan-out nets, such as clocks and resets, to the global clock network is an effective way to reduce routing congestion and minimize skew. Due to its high propagation delays, the global clock network is not recommended for use in timing-critical data paths.

The clock macros can be used for assigning signals to the global clock network:

- The CLKBUF macro connects a preferred clock input to a GB. Preferred clock inputs have direct hardwired routing to GBs.
- The CLKINT macro connects a fabric-routed signal to a GB. The CLKINT macro must be used to connect a regular I/O to GB through the FPGA fabric.
- The RCLKINT macro connects a fabric-routed signal to RGB.

The CCCs, on-chip oscillators, clock dividers, NGMUXs, and transceivers drive GBs through hardwired routing.

Libero SoC supports automated global buffer allocation to minimize user intervention. The allocation strategy for global buffers employs the following priority:

- User-inserted global clock macros
- Clock nets
- Asynchronous reset/set nets
- Very high fan-out nets

In the Libero SoC, the default fan-out threshold for global net promotion is larger for data pins (pins involved in register-to-register paths) than asynchronous logic pins (pins involved in register-to-asynchronous paths).

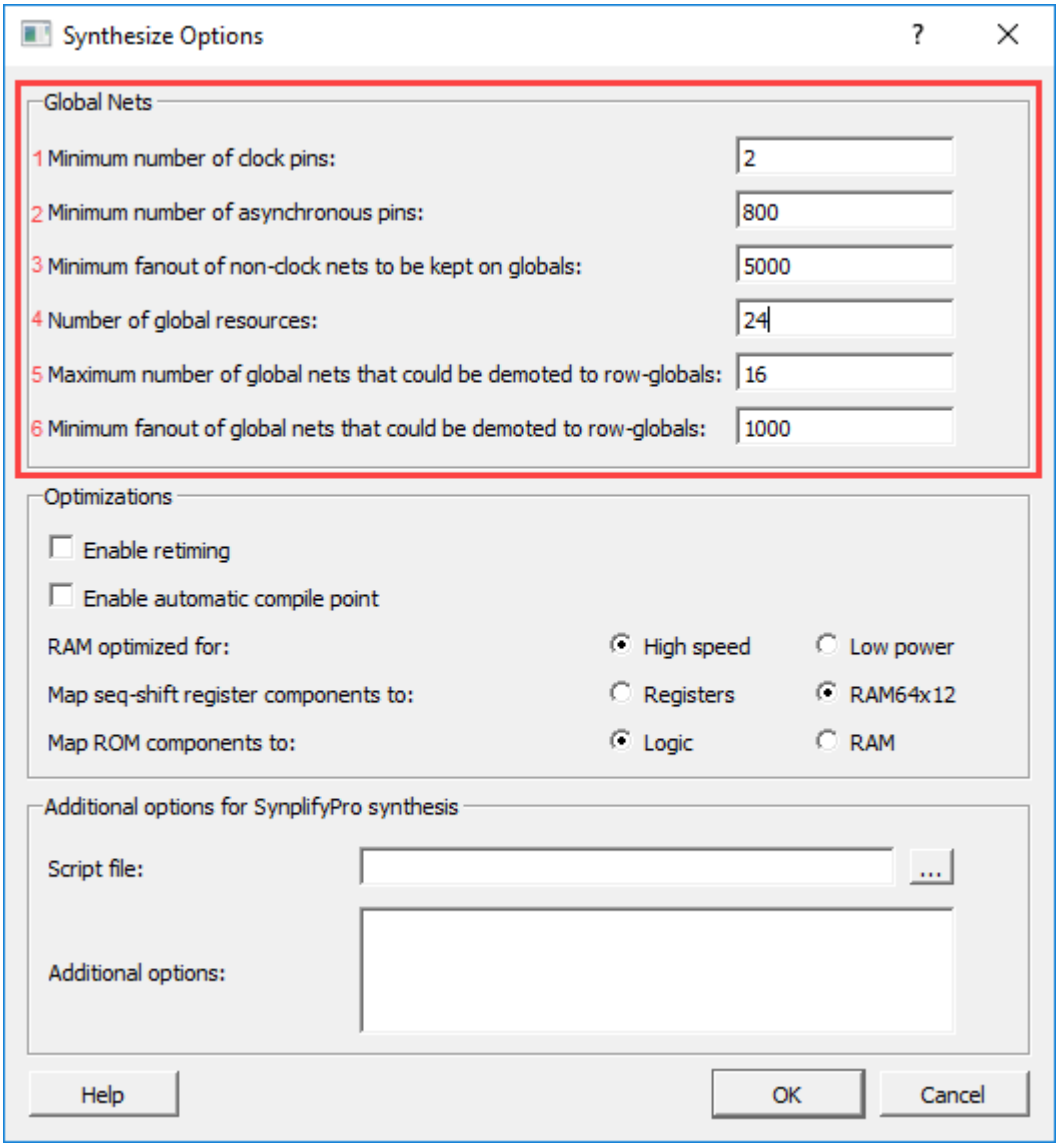
Due to this, the automated design flow is more likely to employ global nets on register-to-asynchronous paths than register-to-register paths. The reasoning for this is that asynchronous pins are not normally timing-critical, and routing them on global nets reduces routing congestion. However, register-to-asynchronous paths are functionally equivalent to register-to-register paths from the perspective of achieving timing closure. As a result, when designing register-to-asynchronous paths, ensure that timing-critical connections do not unnecessarily employ global nets.

If a design contains a failing register-to-asynchronous timing path, check if the path drives a global net in SmartTime. This is done by examining the path and looking for a GB between its launching and latching registers. If a GB is present, you may be able to improve the likelihood of timing closure by demoting the net to a fabric-routed net. An asynchronous net can be demoted by increasing the fan-out threshold of asynchronous pins above the fan-out of the asynchronous net. Alternatively, you can manually adjust the RTL by moving timing-critical pins from high-fan-out asynchronous nets to lower fan-out nets.

Users have the option of setting the minimum fan-out for automatic global assignment. The fan-out threshold values are set in the Libero SoC to automate clock pin promotion to global nets.

In the Libero Design Flow window, expand Implement Design, right-click Synthesize, and choose Configure Options. This opens the Synthesize Options dialog box, as shown in the following figure.

Figure 2-17. Synthesize Options Dialog Box



The following options specify the fan-out threshold value for net promotion and demotion:

1. **Minimum number of clock pins**: Specifies the fan-out threshold value for clock pin promotion. The default value is 2.
2. **Minimum number of asynchronous pins**: Specifies the fan-out threshold value for asynchronous pin promotion. The default value is 800.
3. **Minimum fan-out of non-clock nets to be kept on globals**: Specifies the fan-out threshold value for data pin promotion to global resources. It is the minimum fan-out of non-clock (data) nets to be kept on global nets (no demotion). The default value is 5,000 (must be between 1,000 and 200,000).
   If you run out of global resources for your design, increase this number. If a CLKINT net with fan-out less than this threshold value has data pins along with some clock or asynchronous reset/set pin, move all the data pins to the CLKINT driver net.
4. **Number of global resources**: Specifies the number of global resources to be used in the design. The default value is 24 and you can increase its value up to 48.

5. **Maximum number of global nets that could be demoted to row-globals**: Specifies the maximum number of global nets that could be demoted to RGB resources. The default value is 16 (must be between 0 and 50).
6. **Minimum fan-out of global nets that could be demoted to row-globals**: Specifies the minimum fan-out of global nets that could be demoted to RGB resources. The default value is 1,000 (must be between 25 and 5,000).
   It is undesirable to have high fan-out clock nets demoted using RGB resources because it may result in high skew. If you run out of global resources for your design, reduce this number to allow more globals to be demoted to RGB resources.

Note: Hardwired connections to global resources, such as connections from CCCs and preferred clock inputs, cannot be controlled by synthesize options.

After synthesis, the compiler tool performs the following steps to assign nets to global buffers:

1. Sorting all CLKINT nets in the following priority order:
   - Fan-out, only if fan-out ≥ threshold value specified by minimum fan-out of non-clock nets to be kept on globals
   - Number of clock pins
   - Number of asynchronous reset/set pins
   - Number of data pins
2. Determining the number of GB resources available for CLKINT nets after allocating them to any of the CLKBUF, CLKINT_PRESERVE, and GCLKINT nets.
3. Demoting CLKINT nets from the sorted list that are beyond the limit specified by the number of global resources.
   - If such a net has at least the number of pins specified by minimum fan-out of global nets that could be demoted to row-globals, replace the CLKINT with an RCLKINT macro. Limit the number of nets demoted to RCLKINT to the count specified by maximum number of global nets that could be demoted to row-globals.
   - Otherwise, merge the net with the driver of the CLKINT.

The HDL source file or SmartDesign schematic is the preferred place for defining which signals must be assigned to the global network using clock macro instantiation. A signal with high-fan-out may have logic replication if it is not promoted to a global during synthesis.

Note: The SmartTime tool automatically derives the clock jitter constraints based on the design fan-out and performs the timing analysis.

2.9. Global Net Clock Jitter (Ask a Question)

Global clock networks are susceptible to clock jitter induced by a design-specific simultaneous switching activity and the response of an internal device and board-level Power Distribution Network (PDN).

There are two distinct contributors to increased clock jitter:

- The PDN's inability to meet the large transient current demands under extreme design toggle rates. This can occur regardless of the clock frequencies in the user design.
- Design operation with a large amount of logic toggling simultaneously at the PDN resonance frequency, which typically ranges from 10 MHz to 40 MHz, depending on the specific board design. Depending on the aggressor clock frequency, either one of these two effects at a time can contribute to increased global net clock jitter.

The aggressor clock domain can induce jitter on unrelated victim clock domains. Therefore, the victim clock domains must also consider global net clock jitter. To address the first contributor, the PolarFire core voltage supply (VDD) must always be maintained in accordance with the data sheet specifications at the device package pins, during design operation. The operating voltage

specification includes the regulator DC variation plus any power supply ripple over the customer design frequencies, as measured at the device package pins. For information about operating voltage specification, see the respectivePolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet. The VDD ripple during design operation can be measured for specific designs and boards to ensure that it conforms to the data sheet specifications.

Typically, designs operate with a continuous noise floor and a relatively low simultaneous Flip-Flop (FF) toggle rate, which helps to spread the current demand over time. However, designs with abnormally high toggle rates, greater than 25% of the total device FFs toggling simultaneously in the same direction, increase the instantaneous current demand on the VDD supply. Furthermore, if that large quantity of simultaneously switching FFs periodically starts and stops with a low enable rate, it can cause a large transient current demand on the PDN as the logic is re-enabled. Depending on the amount of fabric logic that starts toggling simultaneously when the enable is asserted, the transient current demand on the PDN can cause the VDD to dip below the data sheet limits. To avoid extra global net clock jitter, ensure that the VDD is always maintained within the data sheet limits during design operation.

Regarding the second contributor to global net clock jitter, it is related to the utilization percentage of the FFs in the device that toggle simultaneously, in the same direction, at a given clock edge, when the clock domain operates within the 10 MHz to 40 MHz range. In typical designs, the average FF toggle rate per clock domain does not exceed ~25%. Specific boards and designs might require higher toggle rates, but support for that level of operation must be confirmed by design-specific VDD variation and clock jitter measurements taken on the user board with the design operating at its maximum switching activity.

For information about maximum period jitter specifications on the PolarFire Global Network for various FF toggle rates within the device, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet. This specification on PolarFire global net clock jitter is a worst-case maximum period jitter that applies across the supported silicon process, voltage, and temperature range.

Libero SoC v2022.1 and later design software flow has an enhanced capability that automatically analyzes, applies clock uncertainty constraints, and accounts for global net clock jitter during Layout and Static Timing Analysis (STA). The software clock jitter algorithms used during timing-driven Place & Route optimize and compensate for known clock jitter components of the design. This process generates a Timing Analysis Jitter Report that details the jitter calculated to clock domains. Clock uncertainty constraints can also be modified, and user additive constraints can be specified for external sources. For information, see Timing Constraints Editor User Guide.

3. On-Chip Oscillators (Ask a Question)

The PolarFire family offers two independent on-chip RC oscillators (2 MHz and 160 MHz) to generate free-running clocks. For information about the electrical characteristics of the on-chip oscillators, see the respectivePolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

On-chip oscillators are powered by device core supply VDD. They do not have any I/O pins and do not require external components for their operation.

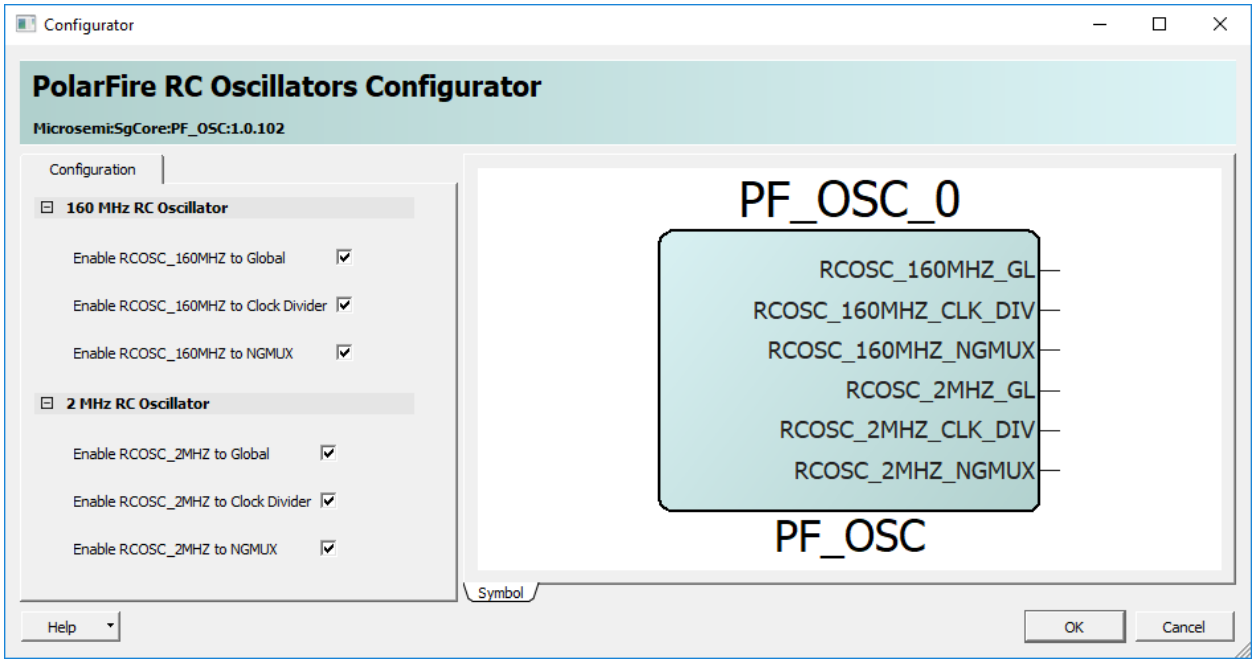
Important:

- The 2 MHz RC oscillator must not be used as the reference clock for PLLs. For better PLL input jitter immunity, the 160 MHz RC oscillator can be used. However, the usage of on-chip RC oscillators is intended for non-precision clocking applications. For RC oscillator frequency accuracy specifications, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.
- The on-chip oscillators are always running irrespective of whether they are used in the design or not. They cannot be disabled. Instantiating the on-chip oscillators makes the clock ports accessible. When they are not instantiated, the clock ports are inaccessible.

The on-chip oscillators' clocks are connected to the global clock network, clock dividers, and NGMUXs through hardwired routing. The on-chip oscillator clocks are used as CCC reference clocks through the global clock network. The on-chip oscillators are located at the lower left corner of the device, see Figure 2-1.

The following figure shows the RC oscillators configurator.

Figure 3-1. RC Oscillators Configurator



The following table lists the configurator ports of RC oscillators.

Table 3-1. RC Oscillators Configurator Ports

Ports	Description
RCOSC_160MHZ_GL	This port is used to drive the 160 MHz oscillator output to a global buffer.
RCOSC_160MHZ_CLK_DIV	This port is used to drive the 160 MHz oscillator output to a Clock Divider present in the Interface Clock Block (ICB).
RCOSC_160MHZ_NGMUX	This port is used to drive the 160 MHz oscillator output to a NGMUX present in the ICB.
RCOSC_2MHZ_GL	This port is used to drive the 2 MHz oscillator output to a global buffer.
RCOSC_2MHZ_CLK_DIV	This port is used to drive the 2 MHz oscillator output to a Clock Divider present in the ICB.
RCOSC_2MHZ_NGMUX	This port is used to drive the 2 MHz oscillator output to a NGMUX present in the ICB.

4. Clock Conditioning Circuitry (Ask a Question)

Each CCC, located in the corners of the device, contains two PLLs, two DLLs, and clock routing multiplexers to route clocks to and from PLLs and DLLs. CCCs provide flexible clock management and synthesis capabilities. PLLs are supported to allow low-jitter clocks for device outputs, and DLLs are supported to allow high-speed tracking of input periodic signals.

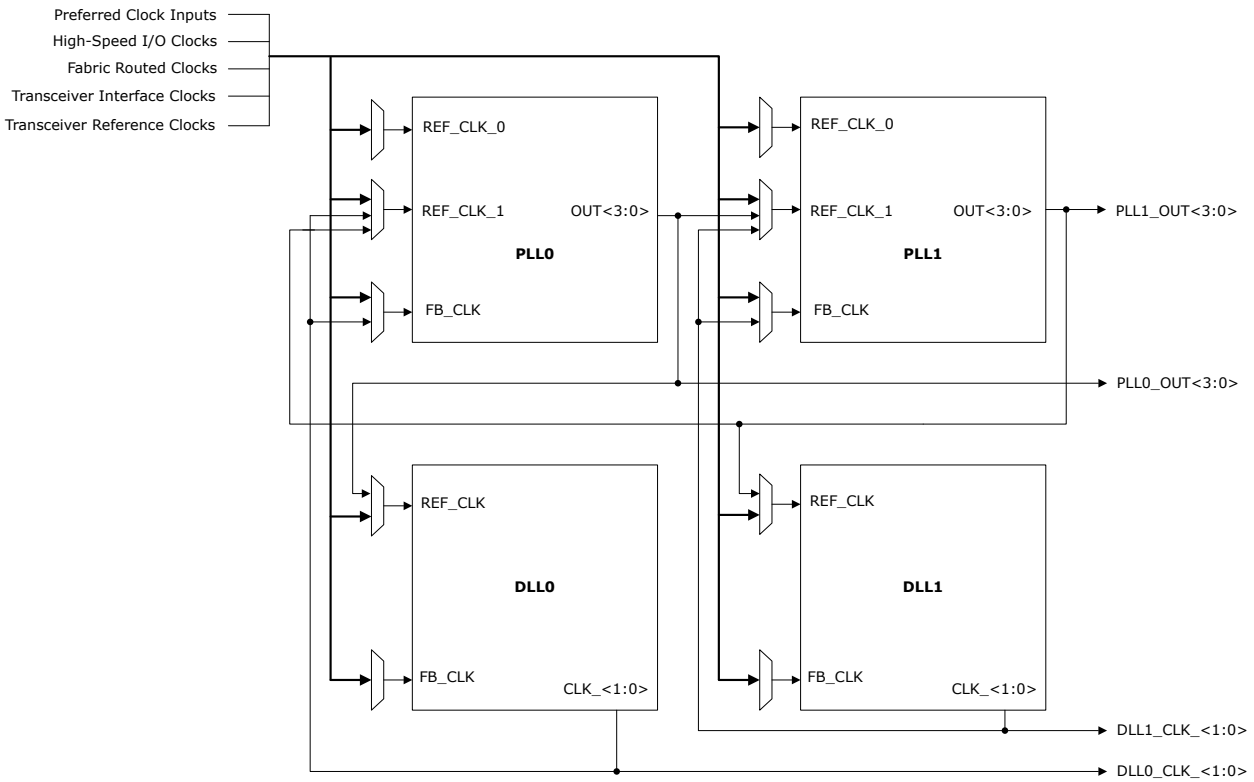
The Libero SoC software provides a CCC configurator with a visual configuration wizard for quick and easy configuration. For DC and switching characteristics of the CCCs, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

4.1. Features (Ask a Question)

Each CCC supports the following features:

- Two fractional PLLs, each supporting:
  - Integer or fractional mode
  - Dual-reference clock inputs with manual switchover
  - Four independent clock outputs
  - Phase selection and adjustment
  - Programmable delay cells
  - Internal feedback mode for low jitter
  - De-skew mode with external feedback clock input
  - Programmable bandwidth control
  - Spread-spectrum clock generation
  - Glitch-free start/stop operations for clock outputs
  - Power-down mode
- Two DLLs, each supporting:
  - Two independent clock outputs
  - Variable phase shift selection
  - Duty-cycle correction
  - Delay code generation
  - Clock division
  - Power-down mode
- PLL and DLL cascading

Figure 4-1. CCC Block Diagram



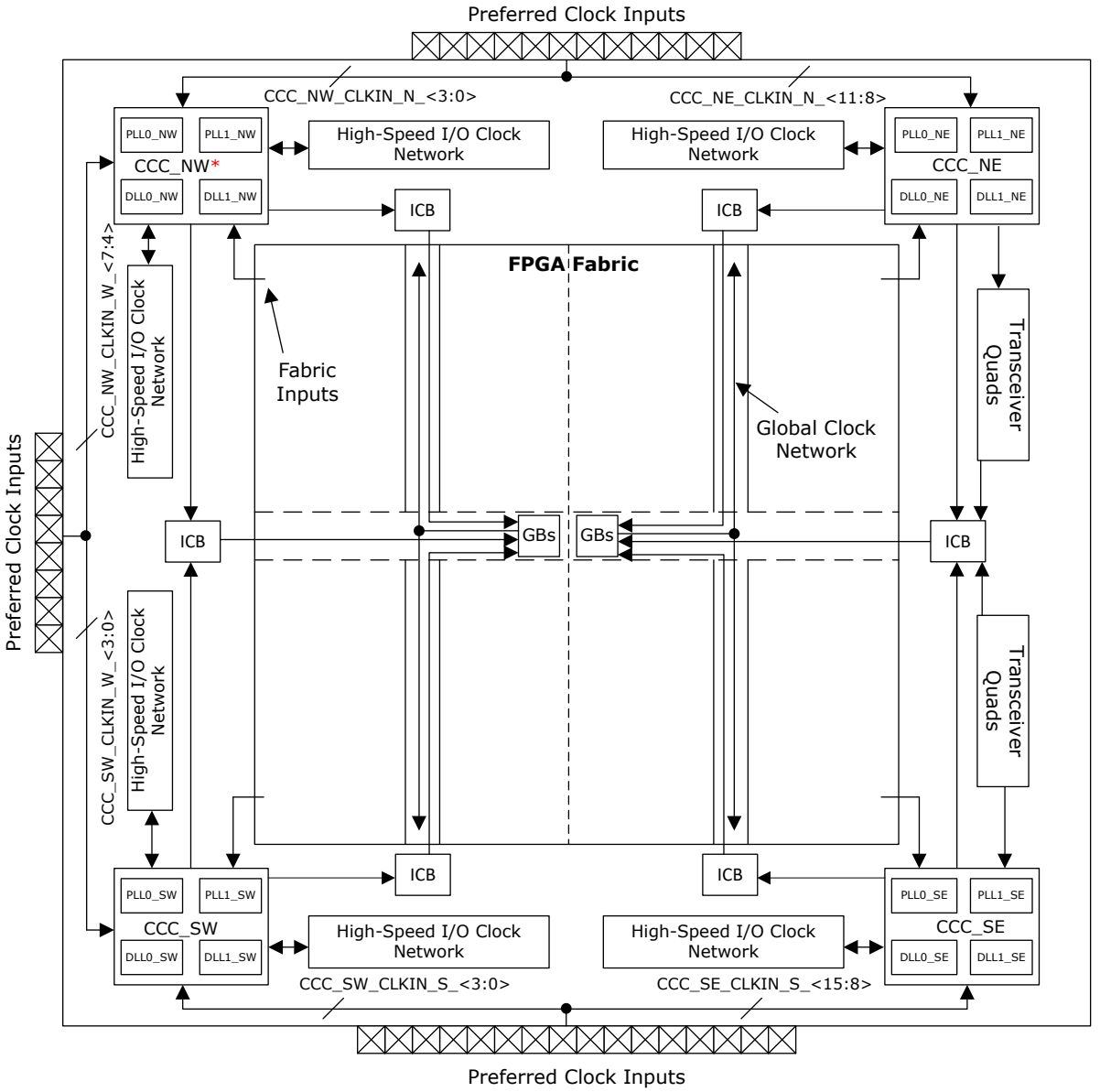
4.2. CCC Locations and Clocking Capabilities (Ask a Question)

The following figure shows the CCC locations and their clocking capabilities. CCCs are labeled according to their locations in the device. For example, the CCC located in the northeast corner is labeled as CCC_NE.

The source clocks for a CCC can come from preferred clock inputs, high-speed I/O clocks, and FPGA fabric. A pair of reference clock inputs from an adjacent transceiver block is present at the CCC_SE. The CCC clock outputs are driven to the global buffers, transceiver as reference clocks, and high-speed I/O clock networks.

Note: For PolarFire SoC and RT PolarFire SoC FPGAs, CCC_NW can generate reference clocks to MSS either from OUT2 or OUT3 clock outputs.

Figure 4-2. System-Level Block Diagram of CCCs



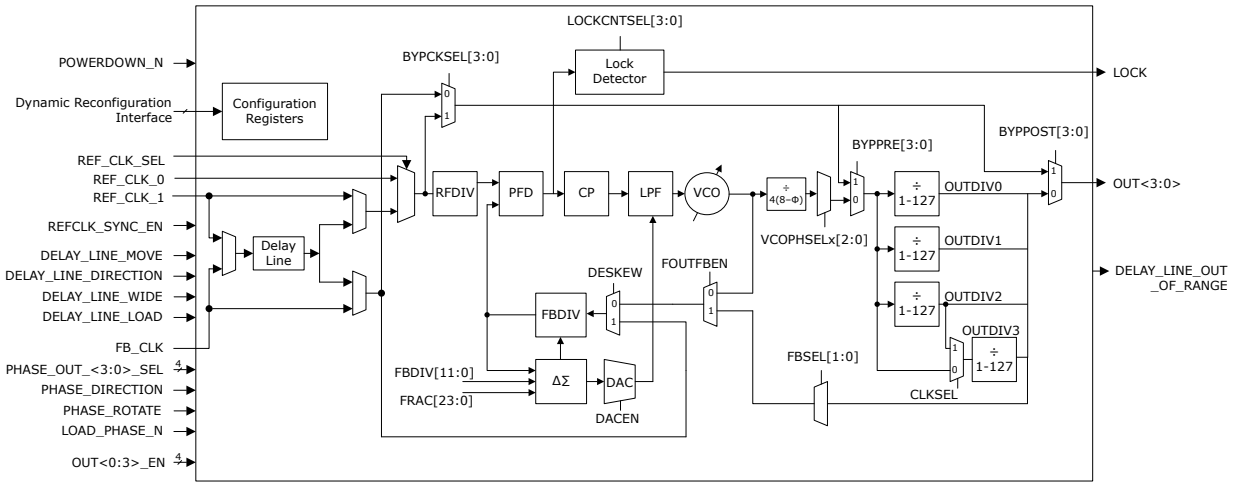
** For PolarFire® SoC and RT PolarFireSoC FPGAs, there is no preferred clock input going to CCC_NW from west edge and there is no feeder line from CCC_NW going to west ICB. Instead the CCC_NW can generate reference clock to MSS.*

** For PolarFire SoC and RT PolarFire SoC FPGAs, there is no high-speed I/O clock input going to CCC_NW from west edge and no feeder line from CCC_NW to high-speed I/O clock network on west edge.*

4.3. Phase-Locked Loops (Ask a Question)

PLLs are used in a variety of clock management applications, such as clock phase adjustment, jitter filter, and frequency synthesis. For systems where Electromagnetic Interference (EMI) is a significant factor, PLLs offer spread-spectrum capabilities to minimize the EMI. The following figure shows the PLL block diagram.

Figure 4-3. PLL Block Diagram



For RT PolarFire devices, the user must review the project-specific reliability requirements and the relevant Radiation Test Report corresponding to the intended operating environment. Reviewing the RT PolarFire radiation test reports for PLL upset rates in Heavy Ion (GEO) versus Proton (LEO) environments helps the user decide whether the CCC PLL can be used for the specific project requirements. The Heavy Ion PLL results are found in RT PolarFire® Heavy Ion Radiation Single Event Effect Test Report LBNL June 5, 2021. The Proton PLL results are found in the *PolarFire Proton Radiation Test Report CNL Oct 17, 2018*. RT PolarFire devices enable high-speed payload data path applications, as opposed to spacecraft control logic, for which RTG4™, RTAX™-S/DSP, and RTSX-SU FPGAs are better suited. It is a payload application-dependent decision whether the system tolerates the PLL upset rates in the test reports and the corresponding system-level impact, such as pauses or gaps in the data stream.

The following information is provided as a brief summary of PolarFire heavy-ion and proton test reports:

- **Heavy-Ion**: According to the RT PolarFire heavy ion test report, there are PLL loss of lock events at low LET levels that do not self-recover. In general, a user might decide to use the RTPF fabric CCC PLL in applications where loss of lock events occurring at the upset rates described in the test reports can be tolerated, where the clock is stopped until the user design can reset the PLL and re-acquire lock. The PF_CCC core does not include auto-reset logic to handle PLL loss of lock. Therefore, to react to a radiation-induced loss of lock, the user can add fabric logic to monitor the PLL lock output. The user logic would detect the loss of lock, gate off the clocks, toggle the PLL_POWERDOWN_N input to reset the PLL, then re-enable the gated clocks after the PLL has re-acquired lock. For more information on two methods to perform glitch-free clock gating using either the CCC Clock Output mode called Global Clock (Gated) or the glitch-free start and stop feature controlled by OUT#_EN, see Clock Start/Stop Input and CCC Configuration sections. The CCC clock output mode using Global Clock (Gated) provides access to the Clock Gating feature, which is described in Clock Gating.
- **Proton**: According to the PolarFire proton test report, there were zero PLL loss of lock events observed. The user must review whether this data is relevant for their mission environment, such as a LEO application, and then design with the PolarFire CCC/PLL as desired.

4.3.1. PLL Port Description (Ask a Question)

The following table lists PLL ports and their descriptions. These ports are accessible from FPGA fabric. The CCC configurator exposes the necessary ports based on the user configuration.

Table 4-1. PLL Port List

Port Name	Direction	Description
REF_CLK_0	Input	Reference clock.
REF_CLK_1	Input	Backup reference clock—Frequency of REF_CLK_0 and REF_CLK_1 must be the same. However, the clocks need to be sourced from different sources.
REF_CLK_SEL	Input	Reference clock select: 1'b0—REF_CLK_0 1'b1—REF_CLK_1
FB_CLK	Input	Feedback clock—Exposed in PLL external feedback mode only.
OUT<3:0>	Output	Clock outputs
PLL_LOCK	Output	Lock output
PLL_POWERDOWN_N	Input	Power down signal (active low): 1'b0—Power down 1'b1—PLL enabled
OUT0_EN	Input	OUT0 output enable
OUT1_EN	Input	OUT1 output enable
OUT2_EN	Input	OUT2 output enable
OUT3_EN	Input	OUT3 output enable
REFCLK_SYNC_EN	Input	Synchronizes the reference clock divider resets of both the PLLs in a CCC with the reference clock. This ensures the alignment of clock output edges from both the PLLs, synchronized with the reference clock.
DELAY_LINE_MOVE	Input	On the rising edge of DELAY_LINE_MOVE, delay line increments or decrements its delay taps based on DELAY_LINE_DIRECTION and DELAY_LINE_LOAD.
DELAY_LINE_DIRECTION	Input	• 0: Decrements the delay taps by 1 • 1: Increments the delay taps by 1
DELAY_LINE_WIDE	Input	• 0: Narrow mode with delay tap range 0–127 • 1: Wide mode with delay tap range 0–255
DELAY_LINE_LOAD	Input	Reloads the Libero SoC programmed delay settings. It must be set to 0 for dynamic delay tuning.
DELAY_LINE_OUT_OF_RANGE	Output	When the delay setting reaches the minimum or maximum value of the delay line, the delay line controller asserts DELAY_LINE_OUT_OF_RANGE to indicate that it has reached the end of the delay line. The delay setting stops at this minimum or maximum value, even if the DELAY_LINE_MOVE signal is still pulsing.
PHASE_OUT0_SEL	Input	Select the OUT0 for dynamic phase adjustment.
PHASE_OUT1_SEL	Input	Select the OUT1 for dynamic phase adjustment.
PHASE_OUT2_SEL	Input	Select the OUT2 for dynamic phase adjustment.
PHASE_OUT3_SEL	Input	Select the OUT3 for dynamic phase adjustment.
PHASE_DIRECTION	Input	Dynamic phase adjustment direction.
PHASE_ROTATE	Input	Rising edge on PHASE_ROTATE causes the phase adjustment to take place where the selected PLL outputs can either be rotated forward or backward by one VCO phase. This signal is shared for all four outputs of the PLL.
LOAD_PHASE_N	Input	A pulse from high to low reinitializes VCO phase shift information to the Libero® SoC programmed value.
DRI_CLK	Input	Clock for dynamic reconfiguration interface.
DRI_CTRL[10:0]	Input	Dynamic reconfiguration interface control bits.
DRI_WDATA[32:0]	Input	Multiplexed address and data bus.
DRI_ARST_N	Input	Active low asynchronous reset for dynamic reconfiguration interface.
DRI_RDATA[32:0]	Output	Multiplexed address and data bus.
DRI_INTERRUPT	Output	Interrupt to the dynamic reconfiguration interface master. It must be held active until the master is serviced the interrupt request.

4.3.1.1. Reference Clock Inputs (Ask a Question)

PLLs have two reference clock inputs (REF_CLK_0 and REF_CLK_1) and support dynamic clock switching. The REF_CLK_1 acts as a backup clock and must be operated at the same frequency as REF_CLK_0 but sourced from a different clock source. The reference clock frequency ranges from 1 MHz to 1250 MHz in integer mode, and 10 MHz to 1250 MHz in fractional mode. A stable reference clock must be supplied to the PLL. The POWERDOWN_N input can be used to keep the PLL in reset until the reference clock is stable.

If the reference clock is sourced from an external source through an I/O, then the PLL must be held in power-down state from the fabric logic until the input buffer of the I/O is known to be operational. This is when the later of the following two conditions occur:

- FABRIC_POR_N negates
- BANK_y_VDDI_STATUS asserts (where y is the number of the I/O bank containing the input buffer). Use the PF_INIT_MONITOR macro to get the status of an I/O bank and FABRIC_POR_N.

If the reference clock is sourced from the on-chip oscillator, then connect the POWERDOWN_N input to FABRIC_POR_N.

The reference clock must be sourced from one of the following:

- Preferred clock inputs
- High-speed I/O clocks
- FPGA fabric routed clocks
- Transceiver reference clocks (to CCC_SE only)
- Transceiver interface clocks (to CCC_SE and CCC_NE)

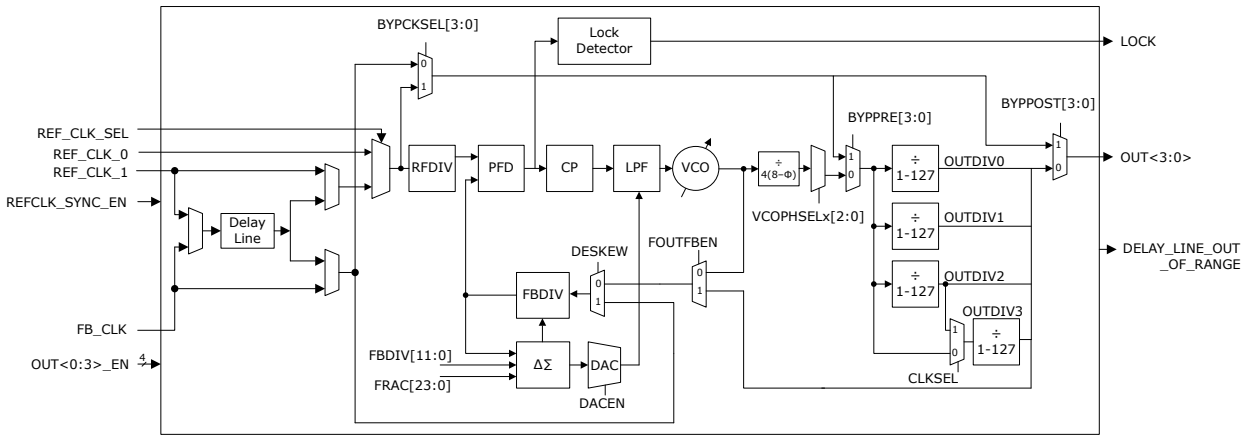
Important: The preferred clock inputs which are capable of driving CCCs have dedicated connections to clock inputs (reference clock or feedback clock) of PLLs and/or DLLs present in the CCCs. For the connectivity of preferred clock inputs to PLLs and DLLs present in a CCC, see Preferred Clock Inputs Connectivity in CCCs.

The clock switching feature is useful in applications that require a redundant reference clock when the primary reference clock stops running. The control signal (REF_CLK_SEL) for the clock switching comes from the FPGA fabric and it must be driven by the user logic to initiate clock switching.

- When REF_CLK_SEL = 1, the PLL selects REF_CLK_1 as the reference clock.
- When REF_CLK_SEL = 0, the PLL selects REF_CLK_0 as the reference clock.

The selected reference clock is passed through a reference divider (RFDIV) before it is fed into the Phase Frequency Detector (PFD). The division values for RFDIV range from 1 to 63. The valid operating range of PFD input frequency (FPFD) is 1 MHz to 312 MHz in integer mode, and 10 MHz to 250 MHz in fractional mode.

Figure 4-4. PLL Block Diagram—Clock Inputs and Outputs



4.3.1.2. Feedback Clock Input (Ask a Question)

The feedback clock input (FB_CLK) is available only when a PLL is configured in external feedback mode. The PLL clock output 0 must be connected to FB_CLK.

The preferred clock inputs which are capable of driving CCCs have dedicated connections to clock inputs (reference clock or feedback clock) of PLLs and/or DLLs present in the CCCs. For the connectivity of preferred clock inputs to PLLs and DLLs present in a CCC, see Preferred Clock Inputs Connectivity in CCCs.

Each PLL has a feedback divider (FBDIV), and a delta-sigma modulator in the feedback path for fractional frequency generation. The division values for FBDIV range from 8 to 10 and 12 to 4095 in integer mode, and 20 to 500 in fractional mode. The FBDIV (INTIN) represents the integer part of the PLL feedback value in fractional mode. The fractional part (FRAC/FRACIN) of the PLL feedback divide value is controlled by the delta-sigma modulator with 24-bit resolution. Total feedback divide value is equal to (FBDIV + FRAC/2^24).

4.3.1.3. Clock Outputs (Ask a Question)

Each PLL has four clock outputs (OUT<3:0>) that can drive global and high-speed I/O clock networks. The CCC_NE clock outputs can drive reference clock inputs of the adjacent transceiver block. The PLL clock output frequency ranges from 1 MHz to 1250 MHz. The OUTDIV2 and OUTDIV3 dividers can be cascaded to generate a slow clock on the OUT3 clock output. For PolarFire SoC and RT PolarFire SoC FPGAs, CCC_NW can generate a reference clock to MSS either from OUT2 or OUT3 clock outputs.

The clock outputs—OUT1 and OUT0 of each PLL can also be connected to preferred clock output pins through low-latency hardwired routing. These preferred clock output pins can be used to drive high-performance clocks in DDR3 and DDR4 applications. The PLL clock outputs are valid only when LOCK is asserted.

During device power-up and programming, the PLLs are powered down and the outputs are driven LOW. The PLL outputs are driven low in the absence of a reference clock. The VDDPLL supply for the PLLs must reach VDD minimum before the power-down of the PLLs is released. For more information about device power-up, see PolarFire Family Device Power-Up and Resets User Guide.

CCC_xy_PLLz_OUTw represents a preferred clock output of one of the PLLs present in a CCC, where:

- **xy**—represents the CCC location: NE, SE, SW, or NW
- **z**—represents the PLL identifier: 0 or 1
- **w**—represents the PLL clock output identifier: 0 or 1

Note: The SmartTime tool automatically derives the clock jitter constraints based on the design fan-out and performs the timing analysis.

4.3.1.4. Lock Output (Ask a Question)

The lock signal, PLL_LOCK, indicates that the PLL clock output is locked onto the reference clock. The lock signal is asserted HIGH to indicate that both frequency and phase lock are achieved.

Lock signal assertion and de-assertion depend on the quality of the reference clock signal (including its noise and jitter characteristics), presence or absence of the reference or feedback clock, and excessive noise in the PLL supply. For information about the maximum jitter tolerance, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

If the reference clock input of the PLL is disconnected, the user cannot see the PLL output, that is, the output is low with the lock signal becoming 0.

The Lock Delay feature of the PLL is used to avoid false toggling of the lock signal. The lock delay setting indicates the number of post-divided reference clock cycles to wait after the PLL lock is achieved, before asserting the lock signal. The lock delay is fixed and set to 256 PFD cycles.

4.3.1.5. Power-Down Input (Ask a Question)

The active-low power-down input (PLL_POWERDOWN_N) from the FPGA fabric forces the PLL to its lowest power state, and the clock outputs are driven low. The PLL_POWERDOWN_N is an asynchronous signal, which can be used to reset the PLL.

Important: When the PLL is configured in internal post-divider or the external feedback mode, the PLL_POWERDOWN_N input must be controlled and de-asserted synchronously. For an example circuit, see PolarFire Family FPGA Power-Up and Resets User Guide.

4.3.1.6. Clock Start/Stop Input (Ask a Question)

PLLs support glitch-free Start/Stop operations on the four clock outputs independently using clock Start/Stop signals (OUT#_EN). This capability allows the output divider values and phase selection to be modified without glitches during the time the clock is stopped. After the OUT#_EN signal is toggled from HIGH to LOW, the clock output is driven low after the second falling edge of the output divider clock output. The transition from low output to toggling clock and vice-versa is a glitch-free operation.

The OUT#_EN can transition at any time since it is re-timed internally. If multiple PLL outputs are enabled via the OUT#_EN signals and arrive at the PLL within 16 VCO cycles of each other, then the PLL outputs start up together and are phase aligned. The following table lists the truth table for enabling PLL outputs.

Table 4-2. Truth Table for Enabling of PLL Outputs

PLL_POWERDOWN_N	OUT#[1]_EN	OUT#[1]
0	X	0
1	0	0
1	1	Normal clock

Note:

1. # = 0, 1, 2, and 3

4.3.1.7. Reference Divider Synchronous Enable (Ask a Question)

Each CCC has an FPGA fabric input—REFCLK_SYNC_EN—to synchronously enable the RFDIV of both the PLLs. Internally, each PLL has its own enable for the RFDIV and is controlled through a PLL register map.

If the same reference clock is routed to both the PLLs in a CCC and needs to be divided, asserting REFCLK_SYNC_EN enables the reference divider of both PLLs on the same reference clock rising

edge. This ensures proper alignment of the output edges from both PLLs, synchronized with the reference clock.

Note: This feature is not supported in the current version of Libero SoC.

4.3.1.8. Bandwidth Adjustment (Ask a Question)

PLL loop bandwidth is the measure of the PLL's ability to track the reference clock and its jitter. The PLL filters the jitter present above the loop bandwidth. PLL passes the jitter (aka wander) below the loop bandwidth. PLLs provide a programmable bandwidth feature, which is configurable using the CCC configurator. The following table lists bandwidth parameter settings.

If the reference clock has a significant amount of jitter, use lower bandwidth to filter the noise. If a higher quality reference clock is used, fast lock time is achieved by using a higher bandwidth value. The CCC Configurator displays the computed bandwidth value based on the PLL bandwidth parameter configuration. For PLL jitter performance, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet.

Note: In the respective PolarFire Device Register Map, PolarFire SoC Register Map, or *RT PolarFire SoC Register Map* (will be available in a future release), on the PLL tab, the BWP field of the PLL_CTRL2 register (Proportional Path loop bandwidth control) controls the loop bandwidth. The BWP field is mapped to the configurator as shown in the following table.

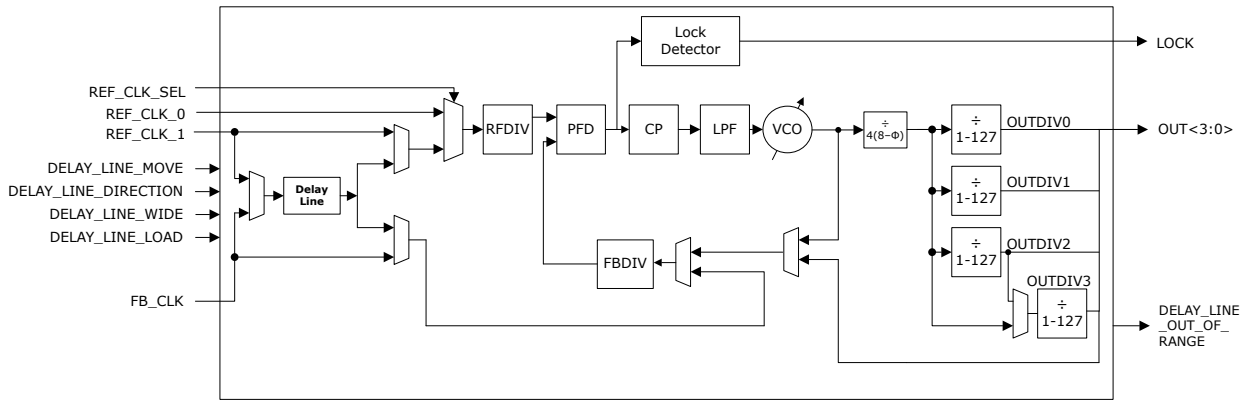
Table 4-3. PLL Bandwidth Parameter Settings

Bandwidth Settings	BWP Field Value	Description
Low	0x00	PLL with a low bandwidth; has better jitter rejection, but a slower lock time.
Medium-Low	0x01	PLL with a bandwidth between low to medium; has a balance between lock time and jitter rejection. This is the recommended setting.
Medium-High	0x10	PLL with a bandwidth between medium to high.
High	0x11	PLL with a high bandwidth; has a faster lock time but tracks more jitter.

4.3.1.9. Delay Line Controls (Ask a Question)

Each PLL has a programmable delay line that can be configured in the reference clock path or feedback clock path. For PLLs, adding delay in the reference clock path enables clock delay, and adding delay in the feedback clock path enables clock advancement with respect to the reference clock. The PLL must be configured in the external feedback mode to add the delay line in the feedback path.

Figure 4-5. PLL—Delay Line Controls



The delay line has 256 delay taps. Each delay tap is designed for ~25 ps steps. The delay taps are not process, voltage, and temperature (PVT) compensated. For characterized value, see the

respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet. Delay lines have two modes of operation—narrow and wide. In narrow mode, 128 delay taps can be programmed, and the resolution is 1 tap per each selection. In wide mode, 256 delay taps can be programmed, and the resolution is 2 taps per each selection. The values for the delay lines are configurable using the CCC configurator, and are programmed during device programming.

Delay lines can be dynamically fine-tuned by the fabric signals, DELAY_LINE_DIRECTION, and DELAY_LINE_MOVE. The dynamic tuning on clock outputs can be re-loaded to the pre-programmed value using the fabric signal, DELAY_LINE_LOAD. The mode of the delay line is configurable using the DELAY_LINE_WIDE signal. On the rising edge of DELAY_LINE_MOVE, the delay line increments or decrements its delay taps based on DELAY_LINE_DIRECTION and DELAY_LINE_LOAD. For more information on delay line control signals, see PLL Port Description.

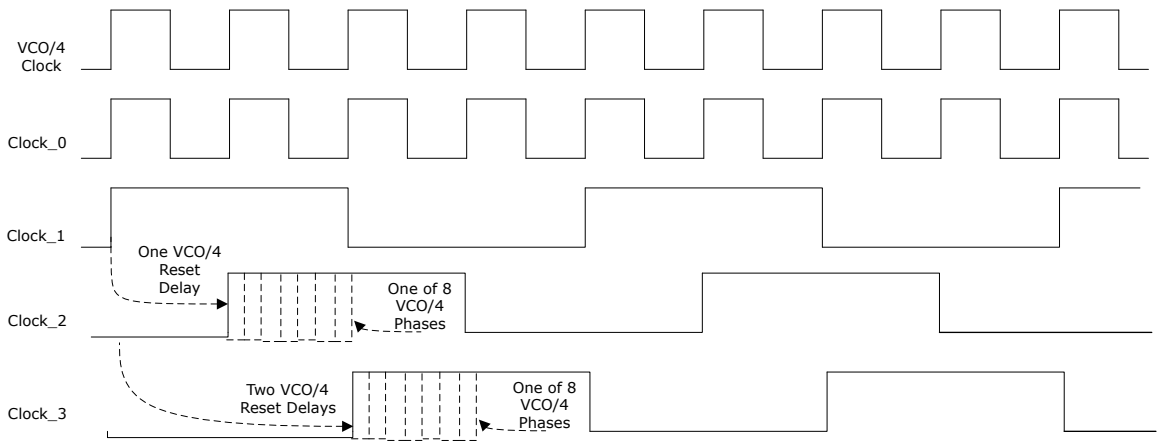
The total delay is a function of the number of delay taps configured and the time value of each delay tap.

4.3.1.10. Static Phase Shifting (Ask a Question)

The VCO/4 clock is available in eight phases with a phase difference of 45°. Each PLL clock output can select one of the eight VCO/4 clock phases, independently. This is called VCO phase select. In addition, each PLL clock output can be delayed further up to seven VCO/4 clock cycles, independently. This is accomplished by holding the output divider in reset for the number of specified VCO/4 clock cycles after a reset. This is called VCO reset delay. These two methods can be selected simultaneously for changing the phase of the clock output.

In the example shown in the following figure, Clock_0, Clock_1, Clock_2, and Clock_3 are PLL clock outputs. Clock_1, Clock_2, and Clock_3 are divided clocks of the VCO/4 clock. Clock_0 is the same as the VCO/4 clock. Clock_2 is delayed by one VCO/4 clock cycle and shows the eight possible VCO/4 phases to further delay Clock_2. Clock_3 is delayed by two VCO/4 clock cycles and shows the eight possible VCO/4 phases to further delay Clock_3. All phase delays shown in the following figure are PVT compensated by the PLL.

Figure 4-6. Example of Using VCO/4 Delay and VCO/4 Phase Select to Fine Tune PLL Output Phase



Phase shifts other than 45° are possible using output dividers. Each output divider is independently programmable, allowing each clock output to have a different phase shift based on the VCO frequency.

The phase shift for PLL clock outputs with respect to the reference clock is configurable using the CCC configurator. The CCC configurator configures the VCO frequency, VCO/4 phase and reset delay,

and output dividers based on the requested frequency and phase. If the configurator is not able to generate an exact match of the requested phase shift with respect to the reference clock, it gives two possible phases to select from—one above (actual higher) the requested phase and one below (actual lower) the requested phase.

Phase adjustment can further be made by placing a PLL delay line or DLL delay line in the reference (to push the phase out) or feedback (to pull the phase in) paths of the PLL. It is also possible to place the DLL delay line on the output clocks, but the jitter injected by the DLL delay line is not filtered by the PLL.

Note: Phase Shift is not allowed for clock output 0 in Post-Divider and External feedback modes.

4.3.1.11. Dynamic Phase Shifting (Ask a Question)

The dynamic phase shifting feature affects the phase of the PLL clock outputs without reconfiguring the device. All four PLL output clocks have the dynamic phase shifting feature. Each PLL has the following fabric ports for dynamic phase shifting.

Table 4-4. Dynamic Phase Shift Ports

Input	Description
PHASE_OUT0_SEL	Selects the OUT0 for dynamic phase adjustment.
PHASE_OUT1_SEL	Selects the OUT1 for dynamic phase adjustment.
PHASE_OUT2_SEL	Selects the OUT2 for dynamic phase adjustment.
PHASE_OUT3_SEL	Selects the OUT3 for dynamic phase adjustment.
PHASE_DIRECTION	Dynamic phase adjustment direction. This signal is shared for all four outputs of the PLL. 0—rotate phase backward 1—rotate phase forward
PHASE_ROTATE	Rising edge on PHASE_ROTATE causes the phase adjustment to take place where the selected PLL outputs can either be rotated forward or backward by one of 8 VCO/4 phases. This signal is shared for all four PLL outputs.
LOAD_PHASE_N	A pulse from high-to-low re-initializes the VCO phase shift information to the Libero® SoC programmed value.

The initial phase shift is static phase shift information set through the CCC configurator. The initial phase shift information is loaded into the PLL whenever the PLL is reset (PLL_POWERDOWN_N = 0) or pulsing the LOAD_PHASE_N signal from high-to-low.

User logic is required to vary the VCO phase settings from the initial value. This is achieved by setting the following signals:

1. Phase rotation direction: Set the PHASE_DIRECTION signal, this signal is shared for all four outputs of the same PLL.
2. Determine which PLL outputs have their phase modified: Select the PLL output clock(s) to have its phase modified via the bus PHASE_OUT<0:3>_SEL.

When the preceding setup signals are set, a rising edge on the PHASE_ROTATE signal (shared for all four outputs of each PLL) causes the phase adjustment to take place where the selected PLL outputs can either be rotated forward or backward by one of 8 VCO/4 phases.

It is required that any outputs that have their phase modified through either method must use the clock stop capability (see Clock Start/Stop Input) before phase modification. After performing the required phase shift configuration, the clocks must be started again. The Clock Start/Stop input is not available in post-divider and external feedback modes, use the Global Clock (Gated) option for the CCC outputs other than feedback clock. Each global buffer has a gating option for glitch-free enabling and disabling of the clock. For more information, see Clock Gating.

Important:

- The PLL requires reset using the PLL_POWERDOWN_N signal and must restart before dynamically shifting the phase.
- Dynamic phase shifting is not supported when OUTDIV > 1.

4.3.2. PLL Operational Modes (Ask a Question)

PLL operational modes depend on how feedback is received by the PLL. The VCO frequency varies based on the feedback mode that the PLL is currently in. The VCO operating frequency ranges from 800 MHz to 5000 MHz. The following are the three PLL operational feedback modes:

- Internal post-VCO
- Internal post-divider
- External

The frequency presented to the output dividers (OUTDIVx) is VCO/4. The CCC configurator configures all the internal dividers (RFDIV, FBDIV, and OUTDIVx) with appropriate values based on the reference clock frequency and the requested PLL output frequency. If the requested PLL output frequency is not achievable, the CCC configurator calculates the two nearest (lower and higher) possible output frequencies to select from, and sets the dividers accordingly.

The total feedback divide value (FBDIV) is equal to:

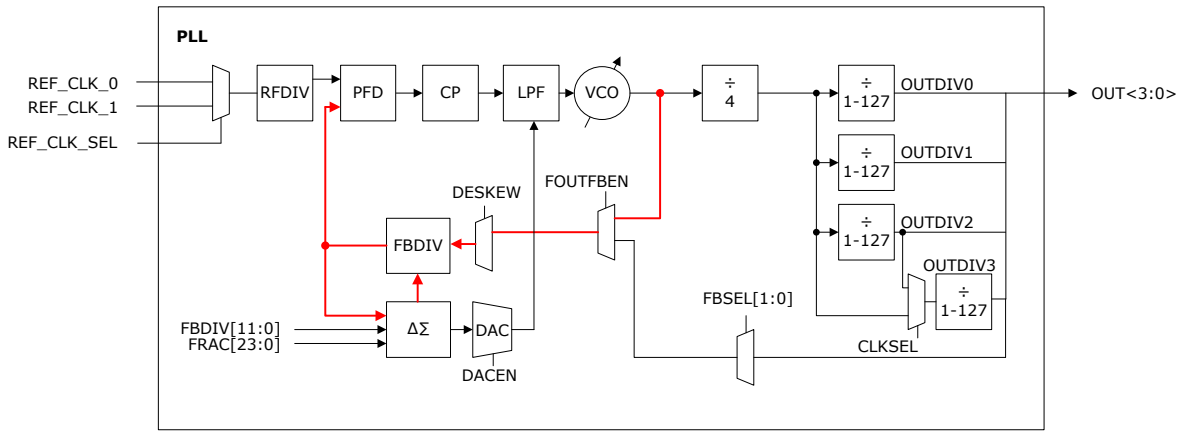
- FBDIV, when PLL is configured for Integer mode
- FBDIV + FRAC/2^24, when PLL is configured for Fractional mode

4.3.2.1. Internal Post-VCO Feedback Mode (Ask a Question)

In this mode, the VCO output is connected as a feedback clock, as highlighted in the following figure. The VCO operates at (REF_CLK × FBDIV)/RFDIV. The output frequency on OUT<3:0> is VCO/(4×OUTDIVx). The PLL outputs (OUT<3:0>) are held low until the PLL_LOCK is asserted. The PLL outputs get reset on the rising edge of PLL_LOCK (reset-on-lock feature) to ensure proper alignment of the PLL outputs. The PLL outputs are not in phase with the reference clock since the divide-by-4 phase generator and output dividers are not in the feedback loop.

Minimum jitter and phase error is achieved in this mode.

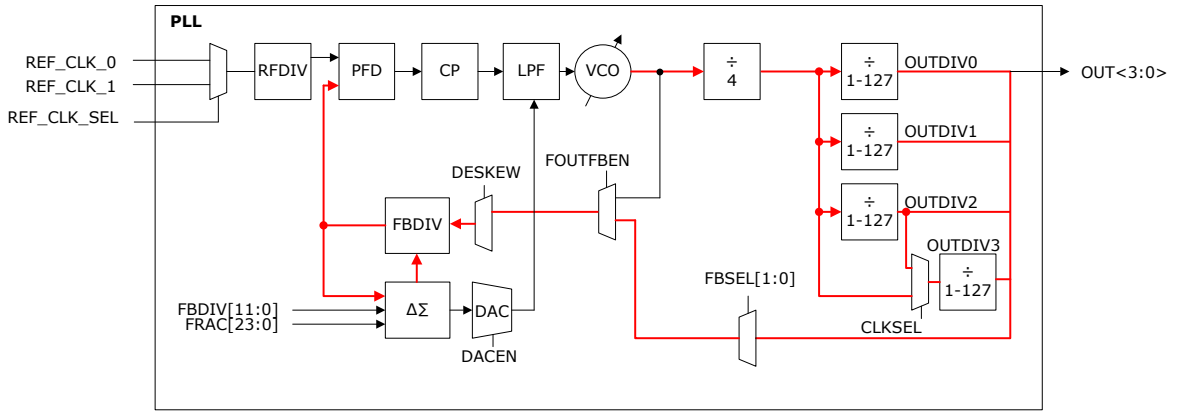
Figure 4-7. Internal Post-VCO Feedback Mode



4.3.2.2. Internal Post-Divider Feedback Mode (Ask a Question)

In this mode, one of the output dividers is placed into the feedback loop, as highlighted in the following figure. The VCO operates at (REF_CLK × 4 × OUTDIVx × FBDIV)/RFDIV. This mode supports a greater range of output frequencies than the range possible with internal Post-VCO feedback mode. The OUTDIV2 and OUTDIV3 can be cascaded to generate a clock up to 127 × 127 slower than the VCO clock. The CCC configurator adds soft-logic around the PLL to ensure proper alignment of the PLL outputs. In this mode, the PLL does not compensate for global clock network delay. For information about required PLL_POWERDOWN_N input control when configured in this mode, see Power-Down Input.

Figure 4-8. Internal Post-Divider Feedback Mode

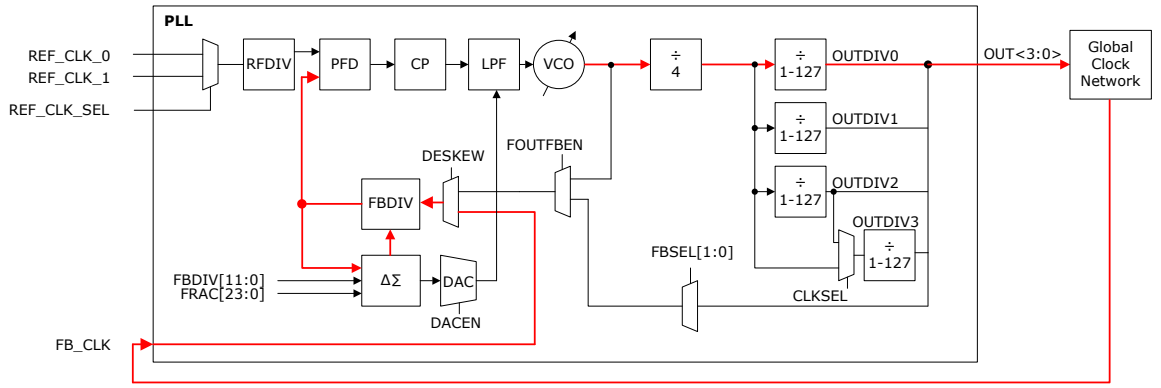


4.3.2.3. External Feedback Mode (Ask a Question)

In this mode, the feedback clock port (FB_CLK) is exposed to the user. The PLL clock output 0 must be connected to FB_CLK either through a global clock network or PCB routing. The following figure highlights the external feedback path routed through a global clock network. The external feedback mode allows users to adjust the clock automatically to compensate for clock network skew and/or PCB routing skew. This mode exhibits more jitter on the PLL outputs compared to the other modes.

In this mode, the VCO operates at (REF_CLK × 4 × OUTDIVx × FBDIV)/RFDIV. Any FBDIV value from 1 to 1250 is valid. In this mode, the PLL output, which is fed back as feedback clock (FB_CLK) is phase aligned with the PLL reference clock. The CCC configurator adds soft-logic around the PLL to ensure proper alignment of the PLL outputs. For information about required PLL_POWERDOWN_N input control when configured in this mode, see Power-Down Input.

Figure 4-9. External Feedback Mode—Feedback through Global Clock Network



4.3.3. Spread Spectrum Clock Generation (Ask a Question)

In the CCC, each PLL is integrated with a Spread Spectrum Modulator (SSMOD) for Spread Spectrum Clock Generation (SSCG). The SSMOD is enabled or disabled using a CCC configurator. The SSMOD modulates the PLL output to spread the fundamental clock signal energy to a wide band of frequencies for reducing ElectroMagnetic Interference (EMI). The lowering of EMI enables significant reduction in expensive shielding cost and reduce interference with other sensitive circuits.

The SSCG capability is supported only when the PLL is placed in Fractional-N mode. Programming options include selection of center spread or down spread, modulation depth, and modulation shape.

The modulation frequency is the rate at which the spreading signal sweeps from the minimum to the maximum PLL output frequency. The modulation depth is represented by percentage spread, which defines the frequency range of the modulated clock resulting from the spread spectrum modulation.

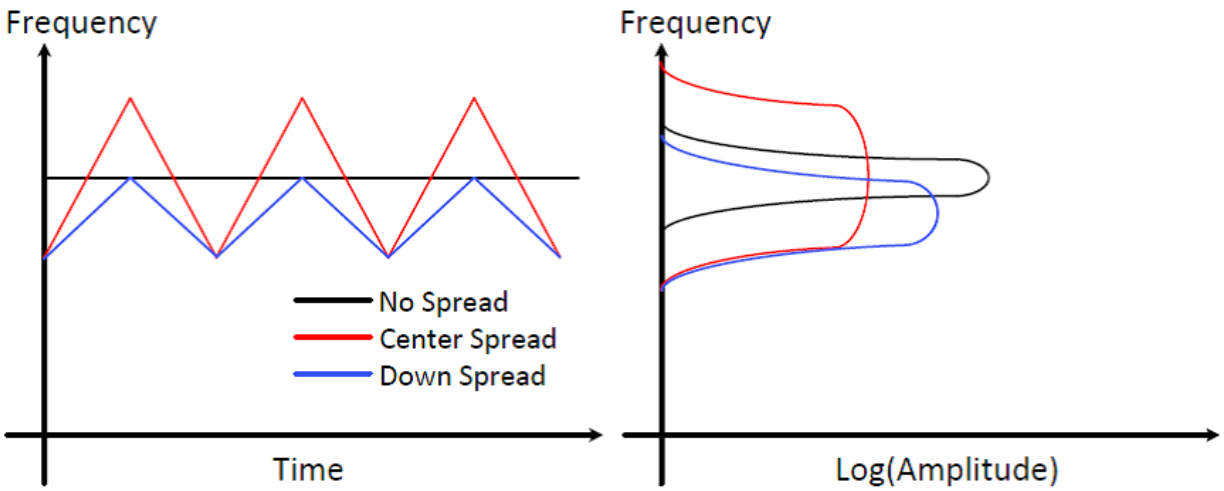
The modulation shape is selectable between a triangular modulation profile and a pseudo random noise source.

The SSMOD works by modulating the feedback divider value of the PLL, thus modulating the PLLs output frequency between minimum and maximum value. For example, consider the case of a PLL with a reference clock of 20 MHz, an output frequency of 1 GHz, and a center spread modulation of 1.5%. The feedback divider is programmed to 50. The SSMOD then modulates the integer and fractional bits so that the PLL is configured with:

- A nominal divide value of 50
- A maximum divide value of 50.75 (FBDIV = 12'b000000110010, FRAC = 24'b110000000000000000000000)
- A minimum divide value of 49.25 (FBDIV = 12'b000000110001, FRAC = 24'b010000000000000000000000)

The following figure shows the frequency versus time and the resulting amplitude in the frequency domain.

Figure 4-10. Spread Spectrum in Time and Frequency Domain Using Triangular-Modulated Waveform



The modulation frequency, modulation depth (spread), and spread mode are configurable in the CCC configurator. A SPREAD value of 0 turns off the modulation, a SPREAD value of 31 gives maximum modulation, and a value of 1 gives minimum modulation.

The following table lists the details of the modulation depth for a given SPREAD value, calculated as follows:

Modulation Depth = ±(SPREAD) × 0.1%

Table 4-5. Center and Down Spread Modulation Depths Based on SPREAD Value

SPREAD	Center Spread	Down Spread
0	0	0
1	±0.1%	−0.1%
2	±0.2%	−0.2%
3	±0.3%	−0.3%
4	±0.4%	−0.4%
5	±0.5%	−0.5%
6	±0.6%	−0.6%
...	...	...
29	±2.9%	−2.9%
30	±3.0%	−3.0%
31	±3.1%	−3.1%

The modulation mode (Center versus Down spread) and the modulation amplitude depends on the amount of EMI reduction desired and the timing margin for logic running on the spread clock domain. The larger the spread value, the greater the reduction in the EMI amplitude. The larger the spread value, the more timing margin needed for the correct logic operation.

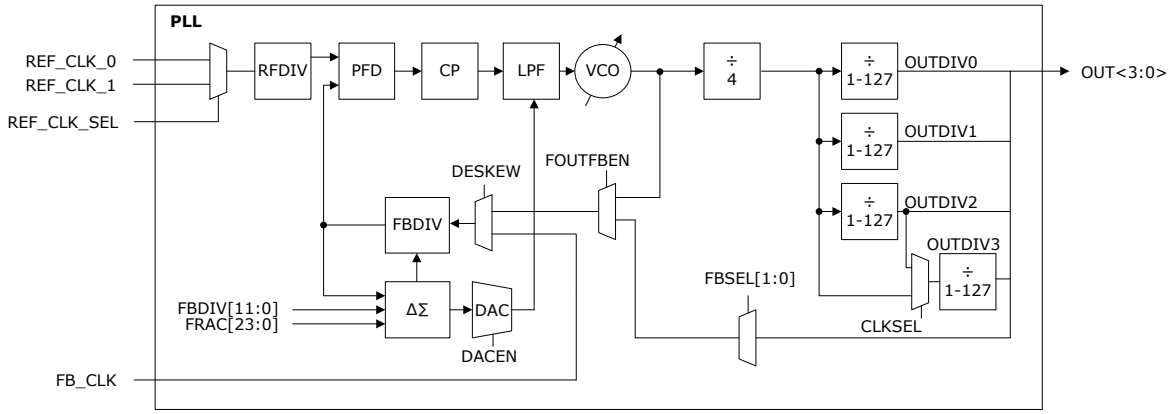
4.3.4. PLL Use Models (Ask a Question)

4.3.4.1. Clock Frequency Synthesis (Ask a Question)

PLLs are used to multiply or divide the reference clock with dividers such as RFDIV, FBDIV, and OUTDIVx. Each of the four PLL outputs offers independent dividers (OUTDIVx) with values between 1 to 127. OUTDIV2, and OUTDIV3 can be cascaded to generate a clock that is up to 127 × 127 slower than the VCO clock.

The following figure shows the frequency synthesis scheme in the PLL.

Figure 4-11. Frequency Synthesis



PLLs operate in Integer or Fractional mode. Fractional-N capability is added to the FBDIV so that the VCO frequency becomes a non-integer divide of the REF_CLK frequency.

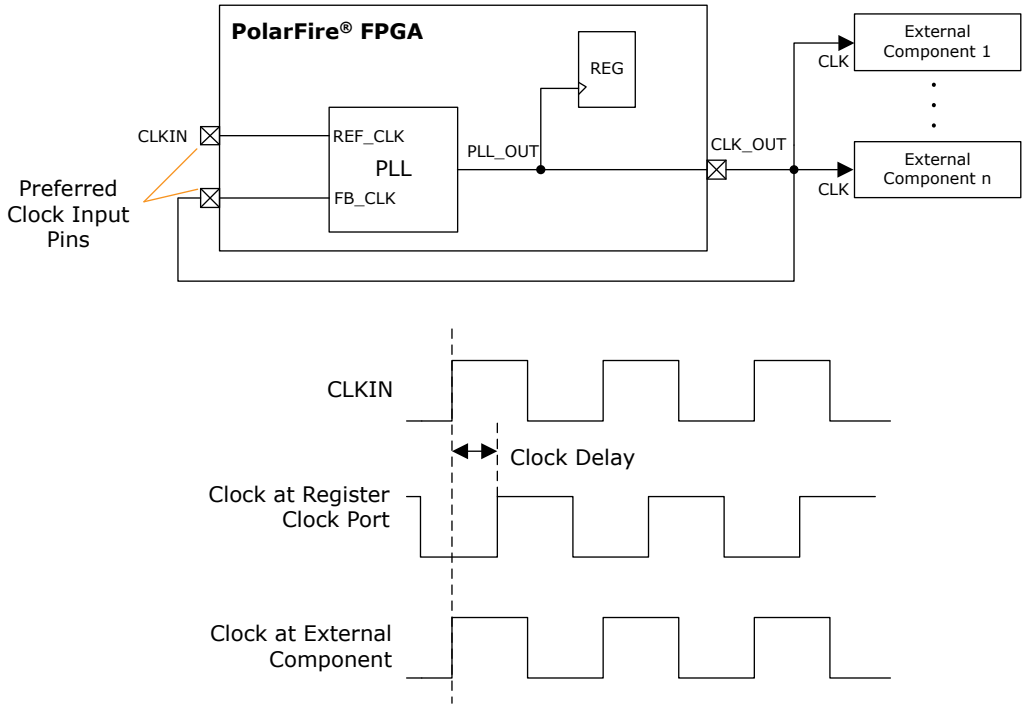
CAUTION Modifying the FBSEL register value is not recommended. The configurator always assumes the feedback path to be Output 0 of the PLL. Any modification might lead to unpredictable results.

4.3.4.2. Zero-Delay Buffer (Ask a Question)

Zero-delay buffer provides a phase-aligned copy of the input clock at the output pins and is useful for clock distribution applications that require a single clock to be fanned out to multiple external components with low skew between them.

As shown in the following figure, the PLL can be used to create a zero-delay clock buffer. The PLL is configured in external feedback mode and the feedback path is confined to the preferred PLL output pin. The zero-delay buffer aligns the clock driven off-chip with the clock input for a minimal delay between the clock input and the external clock output. Delay lines are provided in the CCC to allow the output clock to be pulled back in time.

Figure 4-12. Zero-Delay Buffer—Phase Relationship Between Clocks



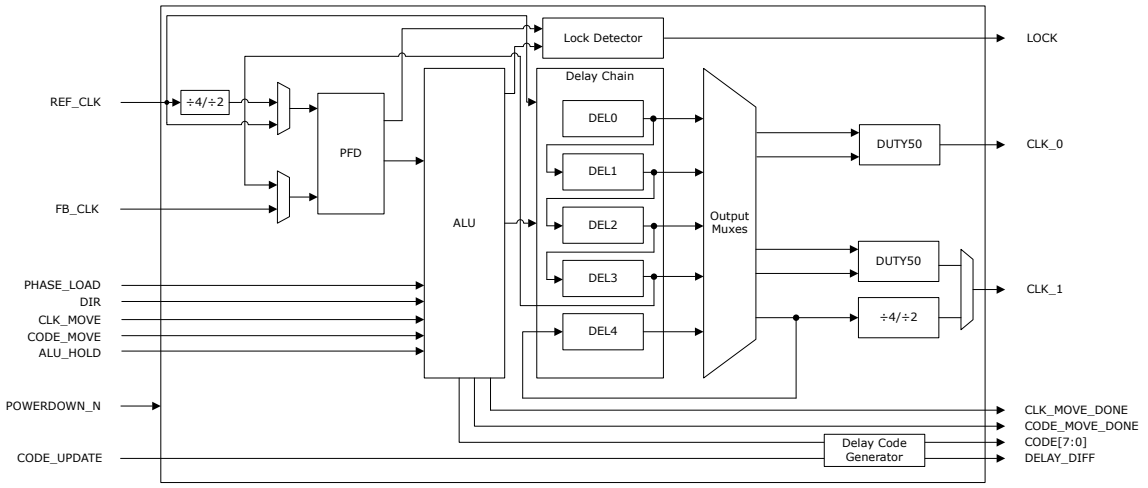
To create a zero-delay buffer, the routing delay between the CLK_OUT pin and the external component clock input pin must match the routing delay between the CLK_OUT pin and the PLL feedback clock pin. It is recommended to use the preferred PLL output pin to route the clock output off-chip and the preferred clock input pins to connect the PLL reference and feedback clock to reduce clock injection delay.

The clock at the internal register can lead or lag the external clock output.

4.4. Delay-Locked Loops (Ask a Question)

DLLs can be used in a wide range of applications, such as precise phase-shifted clock generation, clock insertion delay removal, phase reference delay (for 90° phase shift), and duty-cycle correction. DLLs add delay to the reference clock to create specific phase relationships. There are two types of DLL outputs—clock signals (CLK_0 and CLK_1) and a delay code vector (CODE[7:0]).

Figure 4-13. DLL Block Diagram



The key blocks of a DLL are PFD, arithmetic logic unit (ALU), and a delay chain including five delay cells with 128 delay taps each. Each delay tap is designed for a delay of ~25 ps steps. For characterized values, see the respective PolarFire FPGA Datasheet, RT PolarFire FPGA Datasheet, PolarFire SoC FPGA Datasheet, or RT PolarFire SoC FPGA Datasheet. The delay taps are not PVT compensated.

The reference clock must be sourced from one of the following:

- Preferred clock inputs
- High-speed I/O clocks
- FPGA fabric routed clocks
- Transceiver interface clocks (CCC_SE only)

Note: The preferred clock inputs which are capable of driving CCCs have dedicated connections to clock inputs (reference clock or feedback clock) of PLLs and/or DLLs present in the CCCs. For the connectivity of preferred clock inputs to PLLs and DLLs present in a CCC, see Preferred Clock Inputs Connectivity in CCCs.

The reference clock feeds the delay chain block and PFD. The reference clock can be divided before it is fed to the PFD.

The PFD detects the phase difference between the reference clock and the feedback clock, and produces an up or down signal to the ALU. The ALU increments or decrements the number of delay taps utilized by the delay cells until the rising edges of the feedback clock align with the reference clock. After the two clocks are in phase, the DLL is locked and the LOCK signal is asserted, thereby compensating for the delay in the clock distribution path. The phase lock range has four options to accommodate various cycle-to-cycle jitter tolerance requirements: ±200 ps–400 ps, ±350 ps–700 ps, ±500 ps–1000 ps, and ±750 ps–1500 ps.

The duty-cycle correction feature of DLLs corrects the duty cycle of the reference clock to create a 50% duty-cycle clock output. Clocks with a 50% duty cycle are important for implementing high-speed communication interfaces (for example, DDR applications).

4.4.1. DLL Operational Modes (Ask a Question)

The DLL can be operated in one of the following modes:

- Phase reference mode
- Phase generation mode

- Clock injection delay removal mode

4.4.1.1. Phase Reference Mode (Ask a Question)

In this mode, the DLL_REF_CLK feeds the PFD through two paths—one directly and another through four delay elements in a chain. The ALU increments or decrements delay taps in the delay elements to align the rising edges of the clock through two paths to the same phase. This alignment ensures that the clock delay through all the four delay blocks matches a whole clock period of the DLL_REF_CLK, with each delay block corresponding to a 90° phase shift.

In this mode, the DLL reports a delay code on DLL_CODE [7:0] that states how many delay taps are needed to generate a 90° phase shift with respect to reference clock. The delay code must be connected to a slave delay element located in the I/O logic to apply the same amount of delay to other inputs.

The reference clock frequency must be within the range of 133 MHz to 800 MHz.

Figure 4-14. DLL Port List—Phase Reference Mode

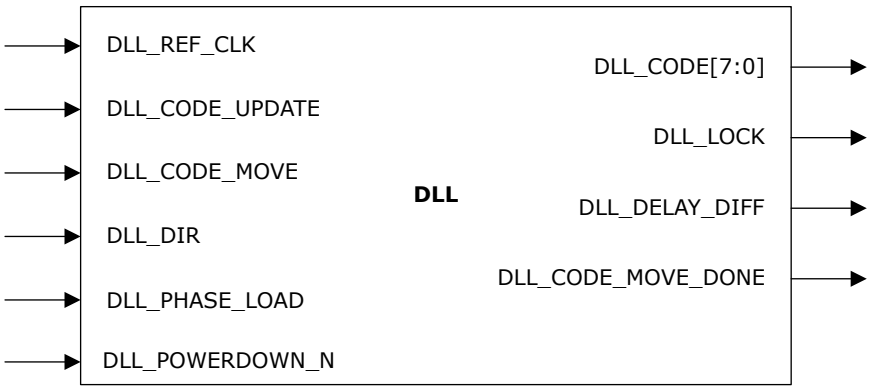


Table 4-6. DLL Port List—Phase Reference Mode

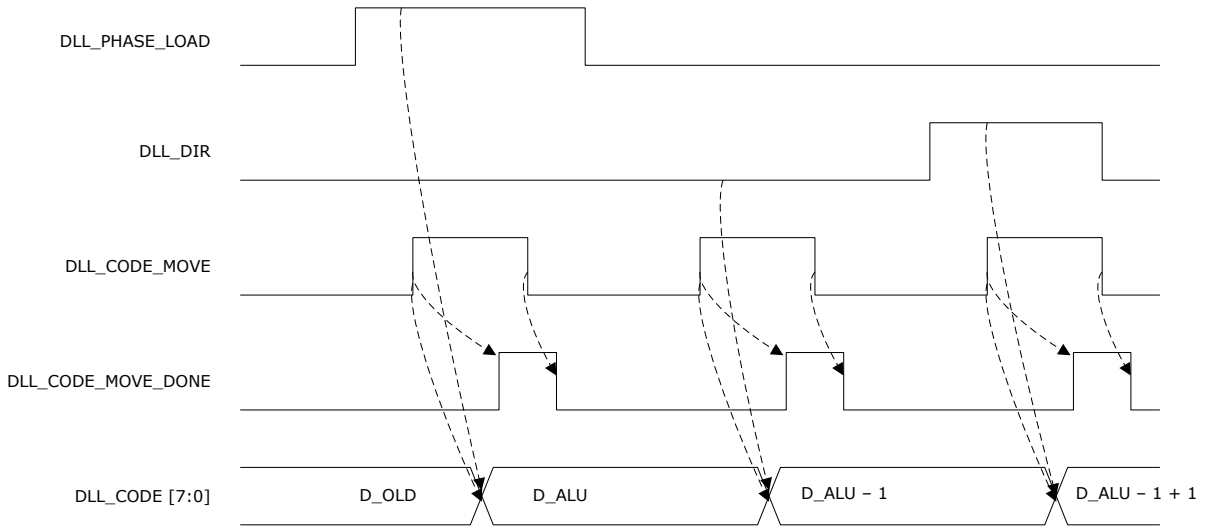
Port Name	Direction	Description
DLL_REF_CLK	Input	Reference clock
DLL_CODE_UPDATE	Input	Delay code update signal
DLL_CODE_MOVE	Input	Rising edge of DLL_CODE_MOVE adds or subtracts a delay tap on DLL_CODE[7:0] output based on DLL_PHASE_LOAD and DLL_DIR.
DLL_DIR	Input	Adds or subtracts a delay tap on DLL_CODE[7:0] when DLL_CODE_MOVE goes high. • 1'b0—Subtracts delay by one tap • 1'b1—Adds delay by one tap
DLL_PHASE_LOAD	Input	Reset the delay settings to the Libero SoC programmed values. It must be set to 0 for dynamic code tuning.
DLL_POWERDOWN_N	Input	DLL power-down input (active low): • 1'b0—Power-down state • 1'b1—DLL is enabled
DLL_CODE[7:0]	Output	Binary delay code output
DLL_LOCK	Output	Lock output
DLL_DELAY_DIFF	Output	Delay code difference indicator
DLL_CODE_MOVE_DONE	Output	The delay code tuning completes with DLL_CODE_MOVE_DONE going high. The falling edge on DLL_CODE_MOVE clears the DLL_CODE_MOVE_DONE and prepares for the next fine-tuning move.

The DLL_DELAY_DIFF output indicates when to update the delay code. The DLL_DELAY_DIFF output gets asserted when the delay code output is different than the ALU up-to-date calculation and an update is needed. The delay code gets updated by driving high on DLL_CODE_UPDATE signal for at least two reference clock cycles. If the DLL_CODE_UPDATE is driven high and held in that state, the delay code output is continuously updated.

The delay code can be adjusted statically by adding or subtracting a number of delay taps using CCC configurator. The user logic can further dynamically add or subtract one delay tap at a time using fabric input signals, DLL_DIR and DLL_CODE_MOVE. The dynamic fine tuning can be re-initialized to the Libero SoC programmed settings using the fabric input signals—DLL_PHASE_LOAD and DLL_CODE_MOVE.

Each rising edge on DLL_CODE_MOVE triggers one fine-tuning load or add/subtract move on delay code output depending on the DLL_PHASE_LOAD and DLL_DIR. The delay code tuning completes with DLL_CODE_MOVE_DONE going high. The falling edge on DLL_CODE_MOVE clears the DLL_CODE_MOVE_DONE and prepares for the next fine-tuning move.

Figure 4-15. Phase Reference Mode—Dynamic Configuration



4.4.1.2. Phase Generation Mode (Ask a Question)

In this mode, the DLL generates two independent clock outputs—DLL_CLK_0 and DLL_CLK_1. The clock outputs can be connected to global and high-speed I/O clock networks. The reference clock frequency ranges from 133 MHz to 800 MHz.

Figure 4-16. DLL Port List—Phase Generation Mode

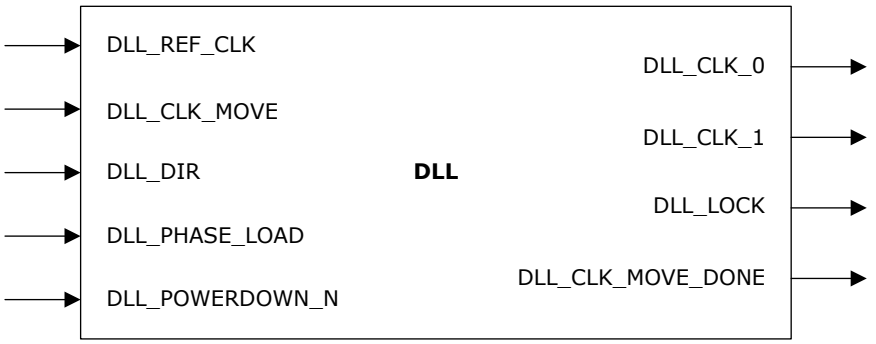


Table 4-7. DLL Port List—Phase Generation Mode

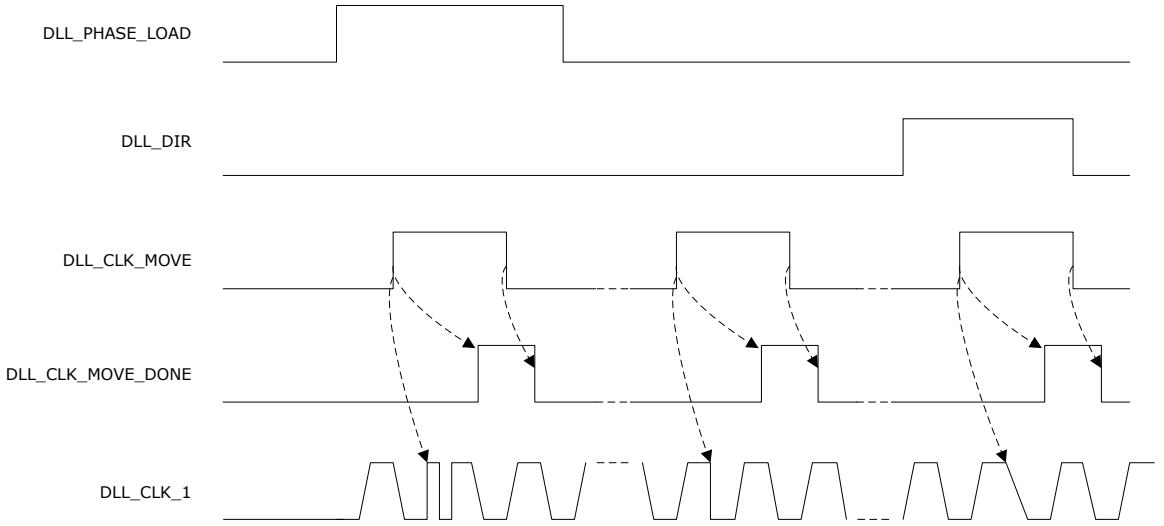
Port Name	Direction	Description
DLL_REF_CLK	Input	Reference clock.
DLL_CLK_MOVE	Input	The rising edge of DLL_CLK_MOVE adds or subtracts a delay tap on DLL_CLK_1 output based on DLL_PHASE_LOAD and DLL_DIR.
DLL_DIR	Input	Adds or subtracts a delay tap on DLL_CLK_1 when DLL_CLK_MOVE goes HIGH. • 1'b0—Subtracts delay by one tap • 1'b1—Adds delay by one tap
DLL_PHASE_LOAD	Input	Resets the delay settings to the Libero SoC programmed values. It must be set to 0 for dynamic clock tuning.
DLL_POWERDOWN_N	Input	DLL power-down input (active low): • 1'b0—Power-down state • 1'b1—DLL is enabled
DLL_CLK_0	Output	Primary clock output.
DLL_CLK_1	Output	Secondary clock output.
DLL_LOCK	Output	Lock output.
DLL_CLK_MOVE_DONE	Output	The clock tuning completes with DLL_CLK_MOVE_DONE going high. The falling edge on DLL_CLK_MOVE clears the DLL_CLK_MOVE_DONE and prepares for the next fine-tuning move.

DLL_CLK_0 can be statically shifted by 0°, 90°, 180°, 270°, or 360°, and can be optionally regulated with a 50% duty cycle.

DLL_CLK_1 can be statically shifted within 32 fine phase options from 0° to 360° in 11.25° steps. The user logic can further dynamically add or subtract one delay tap at a time using fabric input signals—DLL_DIR and DLL_CLK_MOVE. The dynamic fine-tuning on DLL_CLK_1 can be re-initialized to the Libero SoC programmed settings using the fabric input signals—DLL_PHASE_LOAD and DLL_CLK_MOVE.

Each rising edge on DLL_CLK_MOVE triggers one fine-tuning load or add/subtract move on DLL_CLK_1 depending on the DLL_PHASE_LOAD and DLL_DIR. The clock phase shift completes by asserting the DLL_CLK_MOVE_DONE. The falling edge on DLL_CLK_MOVE clears the DLL_CLK_MOVE_DONE and prepares for the next fine-tuning move.

Figure 4-17. Phase Generation Mode—DLL_CLK_1 Dynamic Configuration



Optional divider blocks are available to divide the DLL_CLK_1 output by 2 or 4. DLL_CLK_1 can be regulated with a 50% duty cycle while in 0°, 90°, 180°, 270°, or 360° shift.

The two clock outputs are glitch-free in static phase shift settings and dynamic fine-tuning. DLL_CLK_1 may not be glitch-free while it is re-initialized to the Libero SoC programmed settings.

4.4.1.3. Clock Injection Delay Removal Mode (Ask a Question)

In Clock Injection Delay Removal mode, the DLL is used to compensate for the clock injection delay associated with the source-synchronous receive interfaces. Clock injection delay is the delay from the input pin of the device to a destination element, such as a flip-flop.

Figure 4-18. DLL Port List—Clock Injection Delay Removal Mode

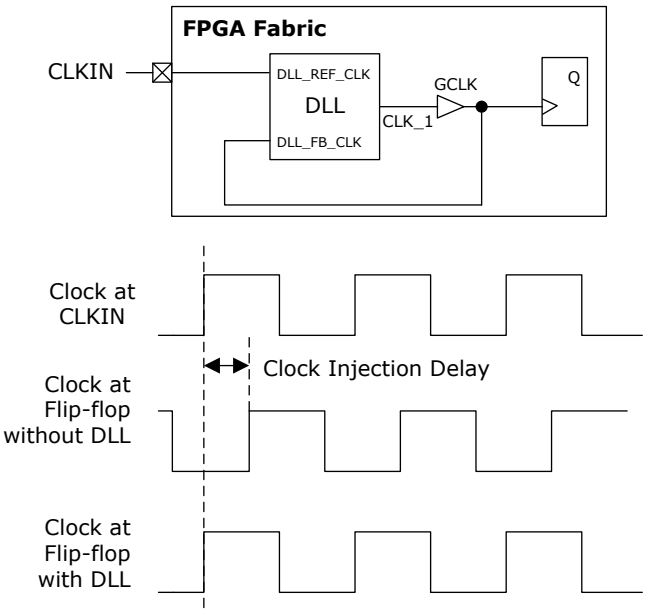
Table 4-8. DLL Port List—Clock Injection Delay Removal Mode

Port Name	Direction	Description
DLL_REF_CLK	Input	Reference clock
DLL_FB_CLK	Input	Feedback clock
DLL_POWERDOWN_N	Input	DLL power-down input (active low): 1'b0—Power-down state 1'b1—DLL is enabled
DLL_CLK_0	Output	Primary clock output
DLL_CLK_1	Output	Secondary clock output
DLL_LOCK	Output	Lock output

The external clock input is connected to the DLL reference clock and the DLL clock output is connected as DLL feedback clock through global clock routing, as shown in the following figure.

The DLL in the clock injection delay removal mode adds delay to the reference clock to align it with the feedback clock, thereby matching delays for the global clock network. When the reference and feedback clocks are phase-locked, the aggregate delay of one or more delay cells and the clock distribution network corresponds to an integer multiple of the clock cycle of the reference clock. This mode supports the distribution of DLL_CLK_0 and DLL_CLK_1 to the global and high-speed I/O clock networks. Only one can be fed back to DLL.

Figure 4-19. Clock Injection Delay Removal Mode



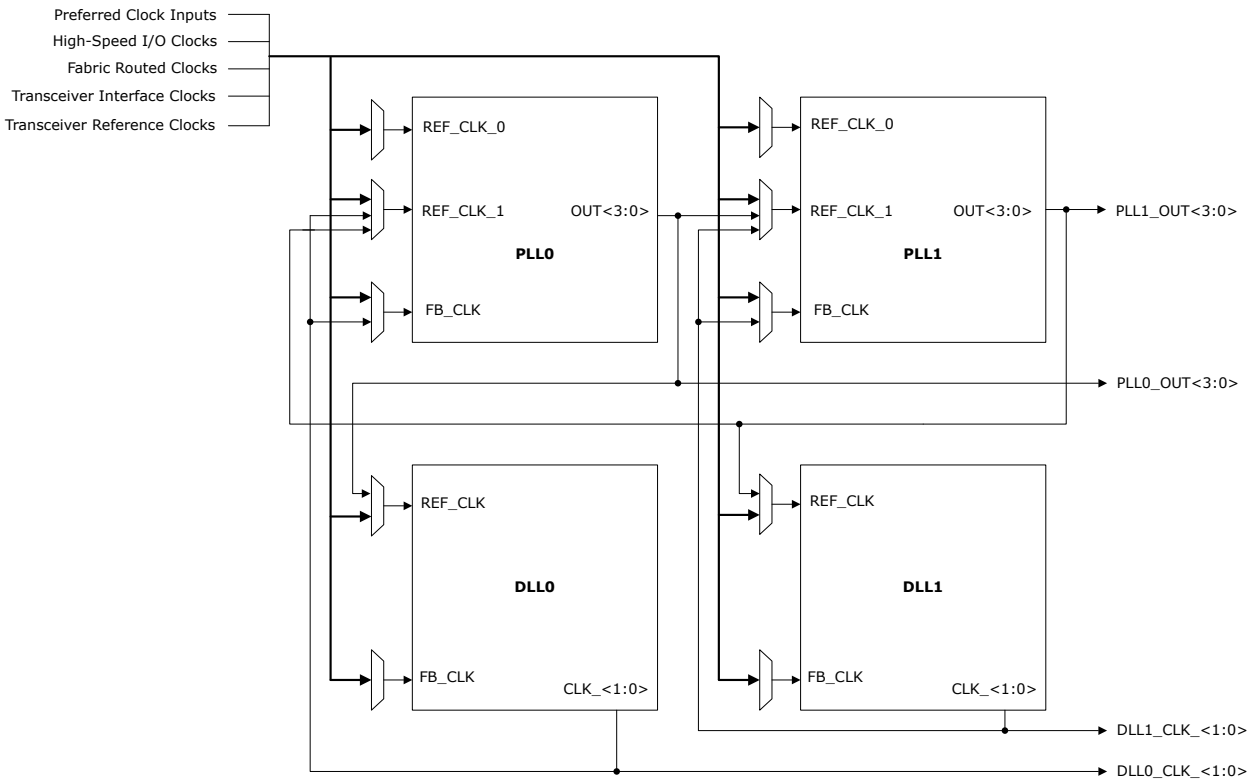
4.5. PLL/DLL Cascading (Ask a Question)

The clock routing MUXs in CCCs facilitate the cascading of PLLs and DLLs. There are two possible cascading schemes:

- PLL to PLL
- PLL to DLL

The following figure shows the reference and feedback clock connections in a CCC. The CCC configurator must be used for configuring PLL/DLL cascading schemes. The CCC configurator configures the MUXs based on the user inputs provided in the CCC configurator.

Figure 4-20. CCC Block Diagram



4.5.1. PLL-to-PLL Cascading (Ask a Question)

CCCs support PLL-to-PLL cascading for more precise clock generation and to allow a greater range of clock frequencies than the range possible with a single PLL.

During cascading, the output of the source PLL serves as the reference clock for the destination PLL. If one PLL drives another, the source PLL is required to use a lower PLL bandwidth setting than the PLL it is driving. This ensures that the dual-PLL combination does not amplify phase noise at any injected noise frequency.

4.5.2. PLL Driving DLL (Ask a Question)

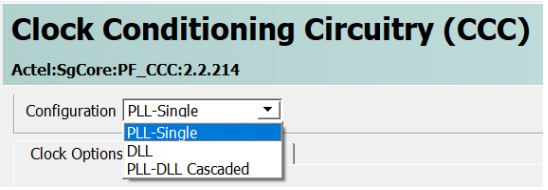
A DLL generates precise phase-shifted clocks. However, it cannot reduce the jitter on the reference clock. The solution is to use a PLL to clean up the jitter before driving it to one or more DLLs. This technique improves the output jitter of all the DLL outputs, but any jitter added by the DLL is still passed to the clock outputs. In this configuration, the PLL must be configured in the internal feedback mode.

4.6. CCC Configuration (Ask a Question)

The PLLs and DLLs present in each CCC are configurable statically using CCC configurator. The CCC configurator provides a visual configuration wizard for a quick and easy way to configure the CCC with desired settings. The CCC configuration set through the configurator defines the power-up state of the CCC. The CCC configurator must be instantiated into the design to use PLL or DLL. A single instantiation of the CCC configurator can be configured as a PLL or DLL. Multiple CCC configurators must be instantiated to use multiple PLLs and DLLs in a design. A design can have up to a maximum of eight PLLs and eight DLLs.

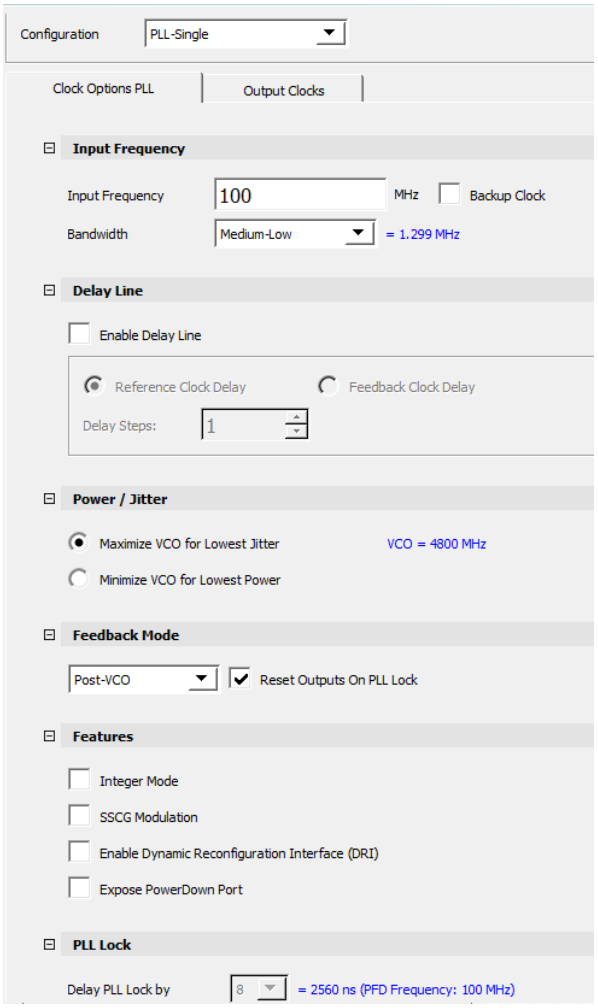
The following figure shows the available configuration options (PLL-Single and DLL) in the **CCC Configurator** window.

Figure 4-21. CCC Configurator—Configuration Options



The following figure shows the Clock Options PLL tab of PLL-Single configuration in the Configurator window.

Figure 4-22. PLL-Single Configuration—Clock Options PLL



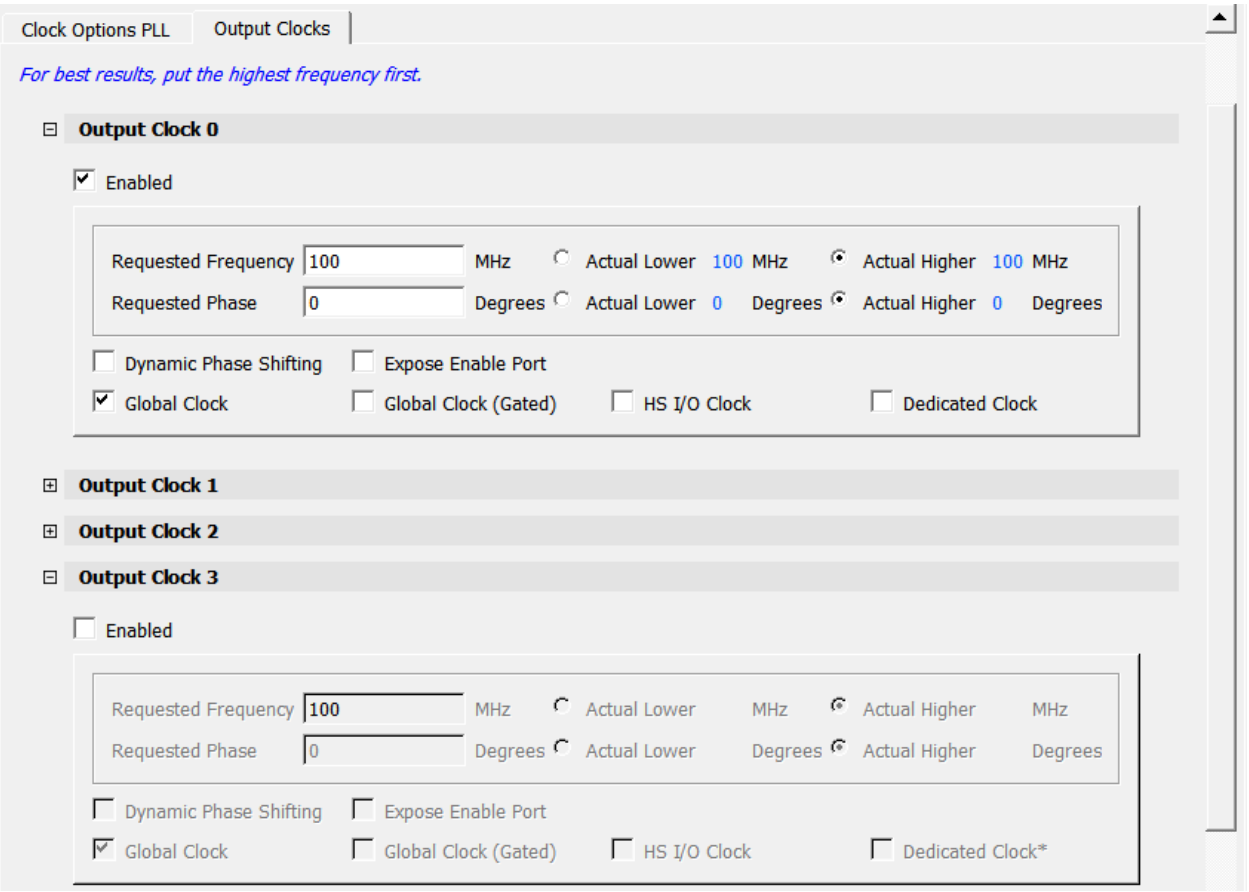
In the **Clock Options PLL** tab, configure the following parameters:

- **Input Frequency**: Enter clock frequency of PLL reference clock.
  - In Integer mode, frequency ranges from 1 MHz–1250 MHz.
  - In Fractional mode, frequency ranges from 10 MHz–1250 MHz.
- **Backup Clock**: If the design needs a redundant clock of the same frequency, enable the backup clock (REF_CLK_1). The clock switching must be done from the fabric using the REF_CLK_SEL signal.

- **Bandwidth**: Select the PLL loop bandwidth (Low, Medium-Low, Medium-High, and High) based on jitter and lock time requirements.
- **Delay Lines**: If the design needs clock delay/phase adjustment, enable delay line in the feedback clock path or backup clock path and select the number of delay steps or taps between 0 to 255.
- **Power/Jitter**: The VCO operating range can be set for minimum jitter at the output or minimum power consumption.
  - Select Maximize VCO for Lowest Jitter to use the highest VCO and FPFD frequencies.
  - Select Minimize VCO for Lowest Power to use minimum VCO and FPFD frequencies for low power consumption. In this mode, the lowest first enabled clock output frequency achievable is 200 MHz.
- **Feedback Mode**: Select PLL feedback mode (Internal, External, or Post-VCO), and feedback clock for the external feedback mode.
  - **Post-Divider**: The configurator uses one of the enabled output clocks as the feedback clock. The selected output clock is shown in the configurator and grayed out.
  - **External**: It exposes the feedback clock port to Fabric. The PLL clock output 0 is selected as the feedback clock.
  - **Post-VCO**: Selects the VCO output as the feedback clock.
    - **Reset Outputs On PLL Lock**: By default, reset on PLL lock is enabled. When this option is disabled, all enabled output clocks start toggling before the PLL is locked. There is no phase relationship between the outputs, and the output clocks are not synchronized with the input reference clock. This feature is available only in Post-VCO feedback mode.
- **Features**
  - Select **Integer Mode** to use the PLL. When **Integer Mode** is disabled, PLL is not constrained. PLL finds an Integer or Fractional solution depending on the input reference clock and requested output frequencies.
  - Select **SSCG Modulation** to enable the spread spectrum clock generation. Click the **SSCG Modulation** tab for more information.
  - Select **Enable Dynamic Reconfiguration Interface (DRI)** to expose the bus interface to the fabric.
  - Select **Export PowerDown** Port to expose the port to fabric.
- **PLL Lock**: The calculated values for PLL Lock Time and PFD frequency is reported in the GUI.
  - PLL Lock Time is calculated as: PFD period * $2^{Delay\_PLL\_Lock}$. The value for Delay_PLL_Lock is fixed at 8 and cannot be changed.
  - PFD frequency = PLL input reference clock frequency/PLL reference divider (RFDIV).

The following figure shows the **Output Clocks** tab of PLL-Single configuration in the **Configurator** window.

Figure 4-23. PLL-Single Configuration—Output Clocks



In the **Output Clocks** tab, configure the following parameters for Output Clock 0, Output Clock 1, Output Clock 2, and Output Clock 3:

- Select **Enabled** to enable the PLL output clocks.
- **Requested Frequency**: Ranges from 1 MHz–1250 MHz for both integer and fractional modes. If the configurator is not able to generate an exact match of the requested frequency, it gives two possible frequencies to select from—one above (actual higher) the requested frequency and the other below (actual lower) the requested frequency.
- **Requested Phase**: Enter the phase shift needed with respect to the reference clock. If the configurator is not able to generate an exact match of the requested phase, it gives two possible phases to select from—one above (actual higher) the requested phase and one below (actual lower) the requested phase.
- **Dynamic Phase Shifting**: Select to expose the fabric signals for dynamic phase shifting.

The CCC configurator calculates the internal divider and phase settings to generate the requested frequency and phase in the following priority:

1. OUT0 frequency
2. OUT0 phase
3. OUT1 frequency
4. OUT1 phase
5. OUT2 frequency
6. OUT2 phase

7. OUT3 frequency
8. OUT3 phase

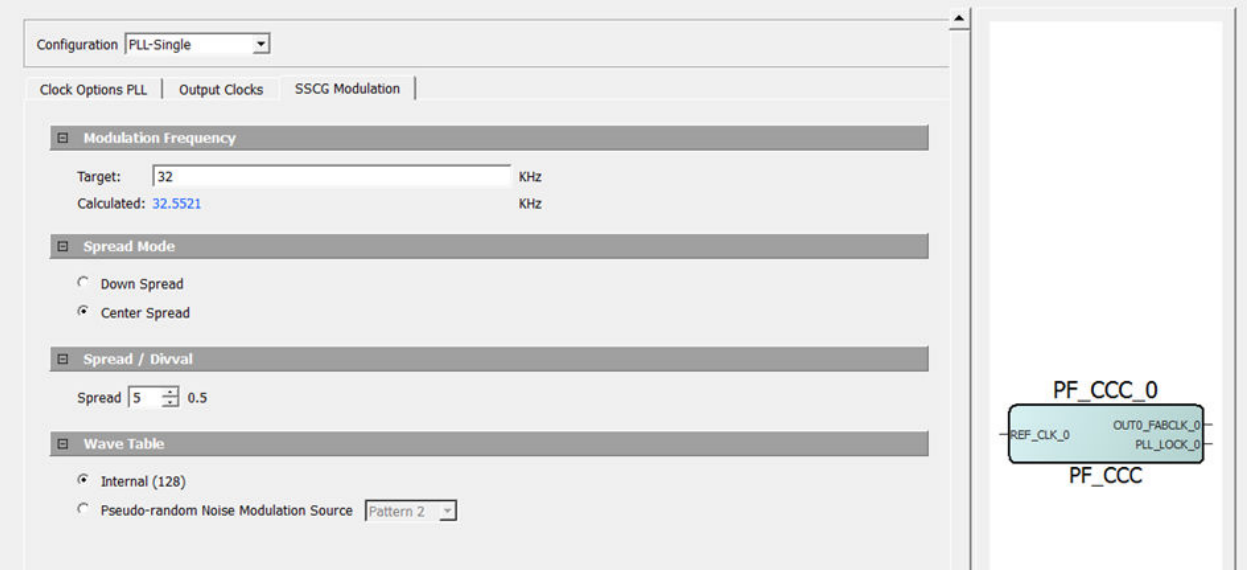
If the user has strict requirement for some frequency or phase, then they need to be allocated first. OUT0 and OUT1 are preferable for high-speed I/O clock as they offer low-latency and jitter.

- Select **Expose Enable** Port to expose the clock output enable port to the fabric.
- Select PLL output clock connectivity
  - **Global Clock**: To connect the PLL output clock to the global clock network.
  - **Global Clock (Gated)**: to connect the PLL output clock to GCLKINT to enable clock gating feature. It exposes OUTx_FABCLK_GATED_x_EN input port and OUTx_FABCLK_GATED_x output port.
  - **HS I/O Clock**: To connect the PLL output clock to a high-speed I/O clock network.
  - **Dedicated Clock**: To connect the PLL output clock to other PLL or DLL in the same CCC, clock dividers, NGMUXs, or preferred clock output pins through dedicated hardwired routing.

Important: The PLL output clocks—OUT0 and OUT1 can be connected to preferred clock output pins through dedicated hardwired routing. The PLL output clocks OUT2 and OUT3 cannot be directly connected to I/Os. They must be routed though fabric.

The following figure shows the **SSCG Modulation** tab of **PLL-Single** configuration in the **Configurator** window.

Figure 4-24. PLL-Single Configuration—SSCG Modulation



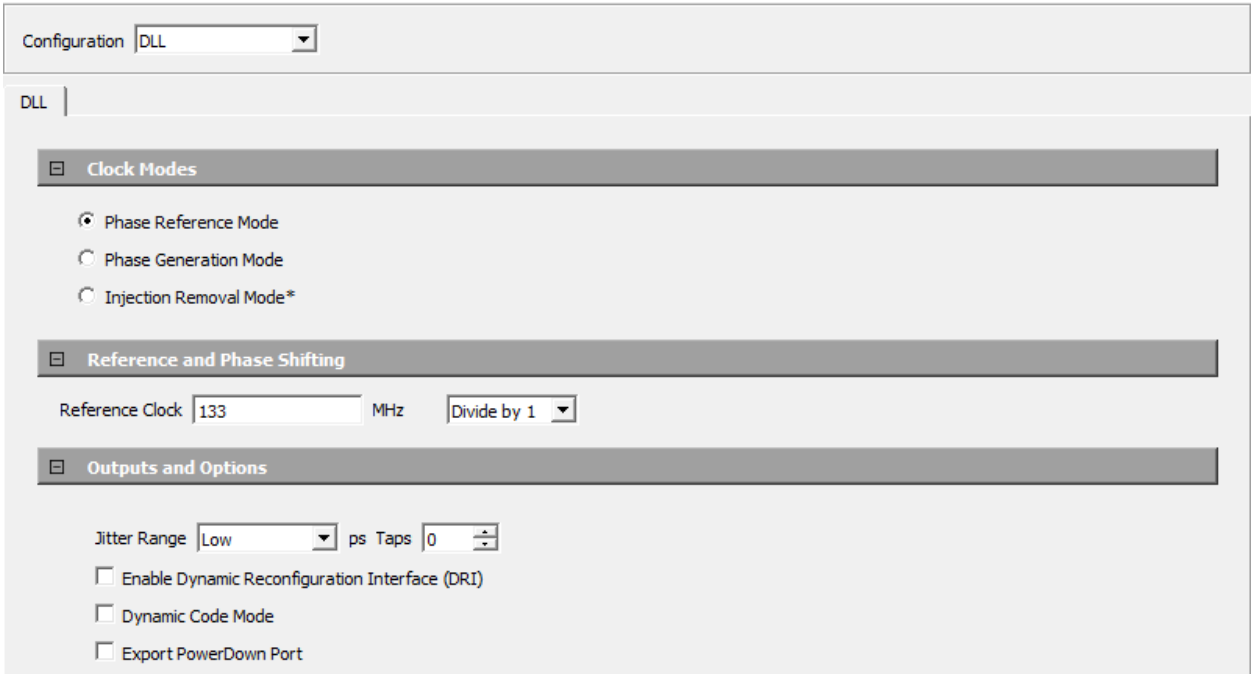
The following parameters are configurable in the **SSCG Modulation** tab if the SSCG modulation feature is enabled in the **Clock Options PLL** tab:

- **Modulation Frequency**: Enter the desired target modulation frequency. The configurator calculates the modulation frequency based on the PLL output frequency and desired modulation value. The calculated value is shown in the configurator.
- **Spread Mode**: Select **Down Spread** or **Center Spread**.
- **Spread/Divval**: Enter Spread value to compute frequency modulation.

- **Wave Table**: Select **Internal (128)** triangular modulation wave table or **Pseudo-random Noise Modulation Source** with an option to choose between 3 patterns.

The following figure shows **DLL** configuration settings in **Phase Reference Mode**.

Figure 4-25. DLL Configuration—Phase Reference Mode

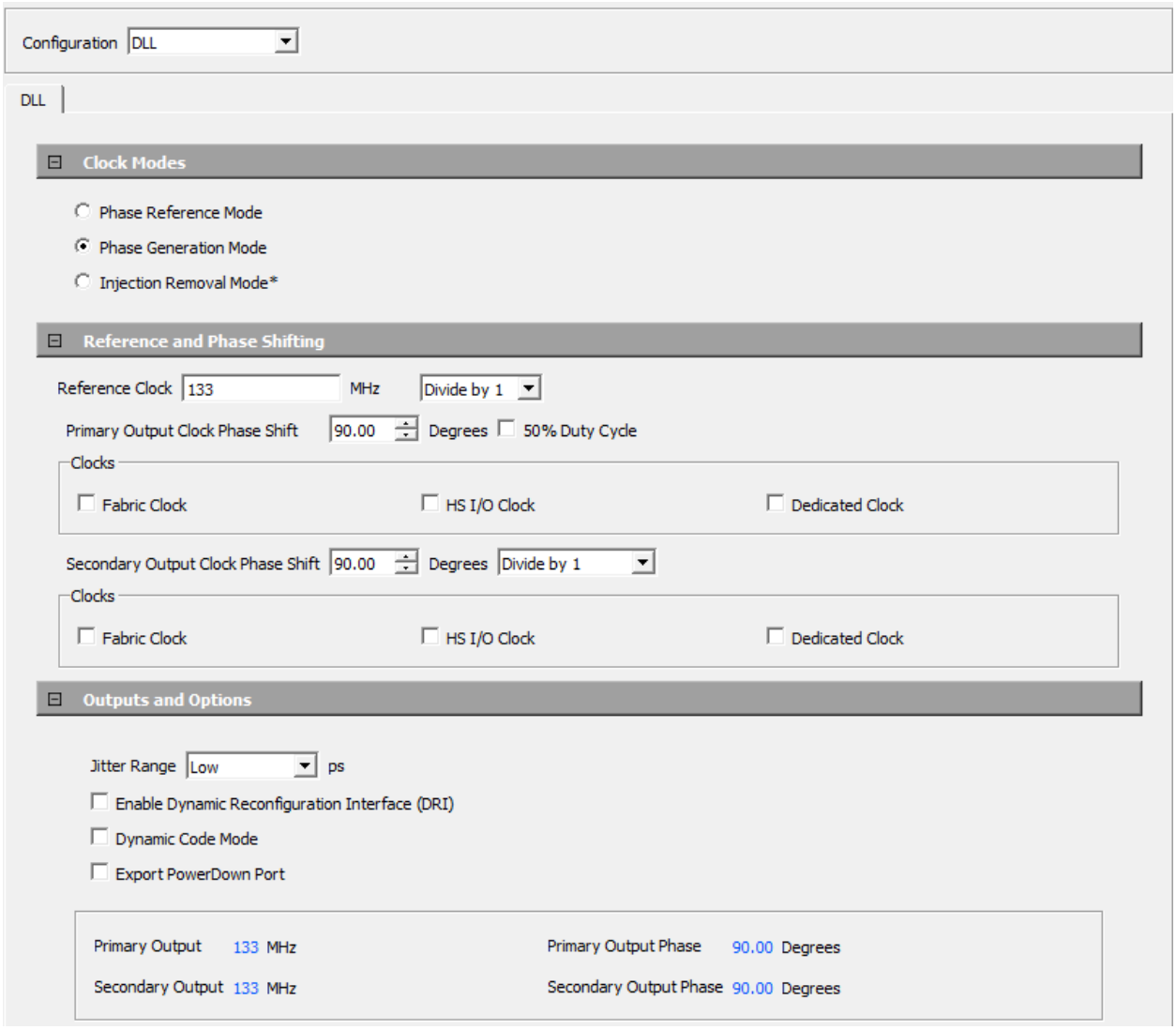


The following parameters are configurable in **Phase Reference Mode**:

- **Reference Clock**: Enter the frequency—ranges from 133 MHz–800 MHz.
- **Outputs and Options**:
  - **Jitter Range**: Select between Low, Medium Low, Medium High, and High.
  - **Delay Taps**: The delay code output can be further adjusted statically by adding or subtracting a number of delay taps. Select between −127 to 127.
  - Select **Enable Dynamic Reconfiguration Interface (DRI)** to expose the bus interface for DLL dynamic configuration.
  - Select **Dynamic Code Mode** to expose the ports for fine tuning the delay code output dynamically.
  - Select **Export PowerDown Port** to expose the port to the fabric.

The following figure shows **DLL** configuration settings in **Phase Generation Mode**.

Figure 4-26. DLL Configuration—Phase Generation Mode



The following parameters are configurable in **Phase Generation Mode**.

Reference Clock Options

- **Frequency**: Enter the reference clock frequency—ranges from 133 MHz–800 MHz.
- **Division**: Select reference clock division value between 1, 2, or 4 as per design requirements.

Primary Output Clock Options

- **Phase Shift**: If the primary output clock requires phase shifting in multiples of 90°, then select the phase shift from the drop-down list.
- **50% Duty Cycle**: Select **50% Duty cycle** to regulate the primary output clock for 50% duty cycle.
- Select primary output clock connectivity:
  - **Fabric Clock**: To connect the primary output clock to the global clock network.
  - **HS I/O Clock**: To connect the primary output clock to a high-speed I/O clock network.
  - **Dedicated Clock**: To connect the primary output clock to other DLL or PLL in the same CCC, clock dividers or NGMUXs through dedicated hardwired routing.

Secondary Output Clock Options

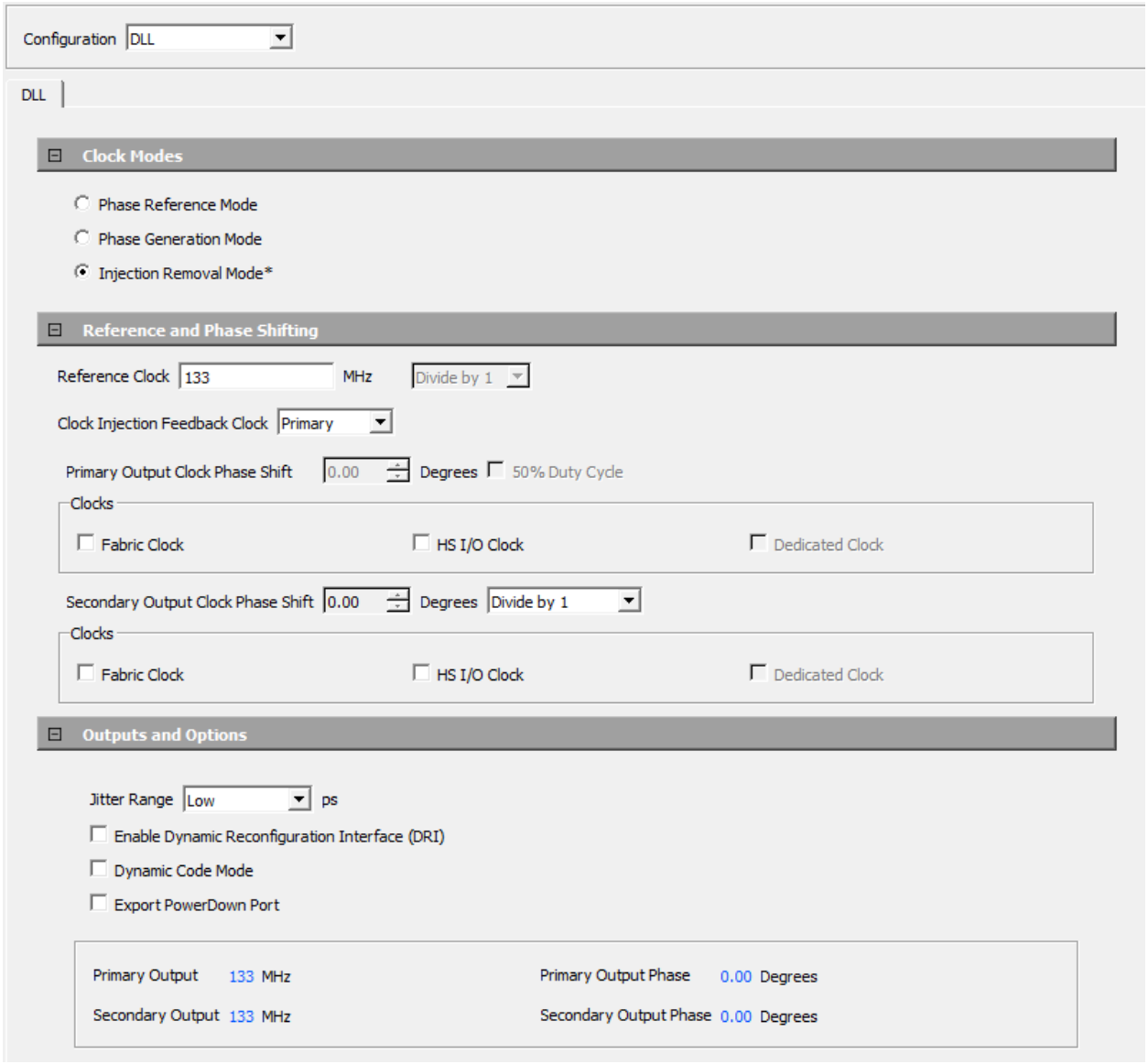
- **Phase Shift**: If the secondary output clock requires phase shifting in multiples of 11.25°, then select the phase shift from the drop-down list.
- **50% Duty Cycle and Clock Division**: The secondary clock output can be regulated with a 50% duty-cycle while in 0°, 90°, 180°, 270°, or 360° shift or when divided by 2 or 4.
- Select secondary output clock connectivity:
  - **Fabric Clock**: To connect the primary output clock to the global clock network.
  - **HS I/O Clock**: to connect the primary output clock to a high-speed I/O clock network.
  - **Dedicated Clock**: To connect the primary output clock to other DLL or PLL in the same CCC, clock dividers, or NGMUXs through dedicated hardwired routing.

Outputs and Options

- **Jitter Range**: Select between Low, Medium Low, Medium High, and High based on the design requirements.
- Select **Enable Dynamic Reconfiguration Interface** to expose the bus interface for DLL dynamic configuration.
- Select **Dynamic Clock Mode** to expose the ports for fine tuning the secondary clock output, dynamically. The secondary clock output can be adjusted dynamically by adding or subtracting a number of delay taps.
- Select **Export PowerDown Port** to expose the port to the fabric.

The following figure shows DLL configuration settings in **Injection Removal Mode**.

Figure 4-27. DLL Configuration—Injection Removal Mode



The following parameters are configurable in **Injection Removal Mode**:

Reference Clock frequency: Enter the reference clock frequency—ranges from 133 MHz–800 MHz.

Clock Injection Feedback Clock: Select the DLL feedback clock source between primary clock output and secondary clock output.

Primary Output Clock Options

- **Phase Shift**: This feature is not available in this mode.
- **50% Duty Cycle**: This feature is not available in this mode.
- Select primary output clock connectivity
  - **Fabric Clock**: Connects the primary output clock to the global clock network.
  - **HS I/O Clock**: Connects the primary output clock to a high-speed I/O clock network.
  - **Dedicated Clock**: This feature is not available in this mode.

Secondary Output Clock Options

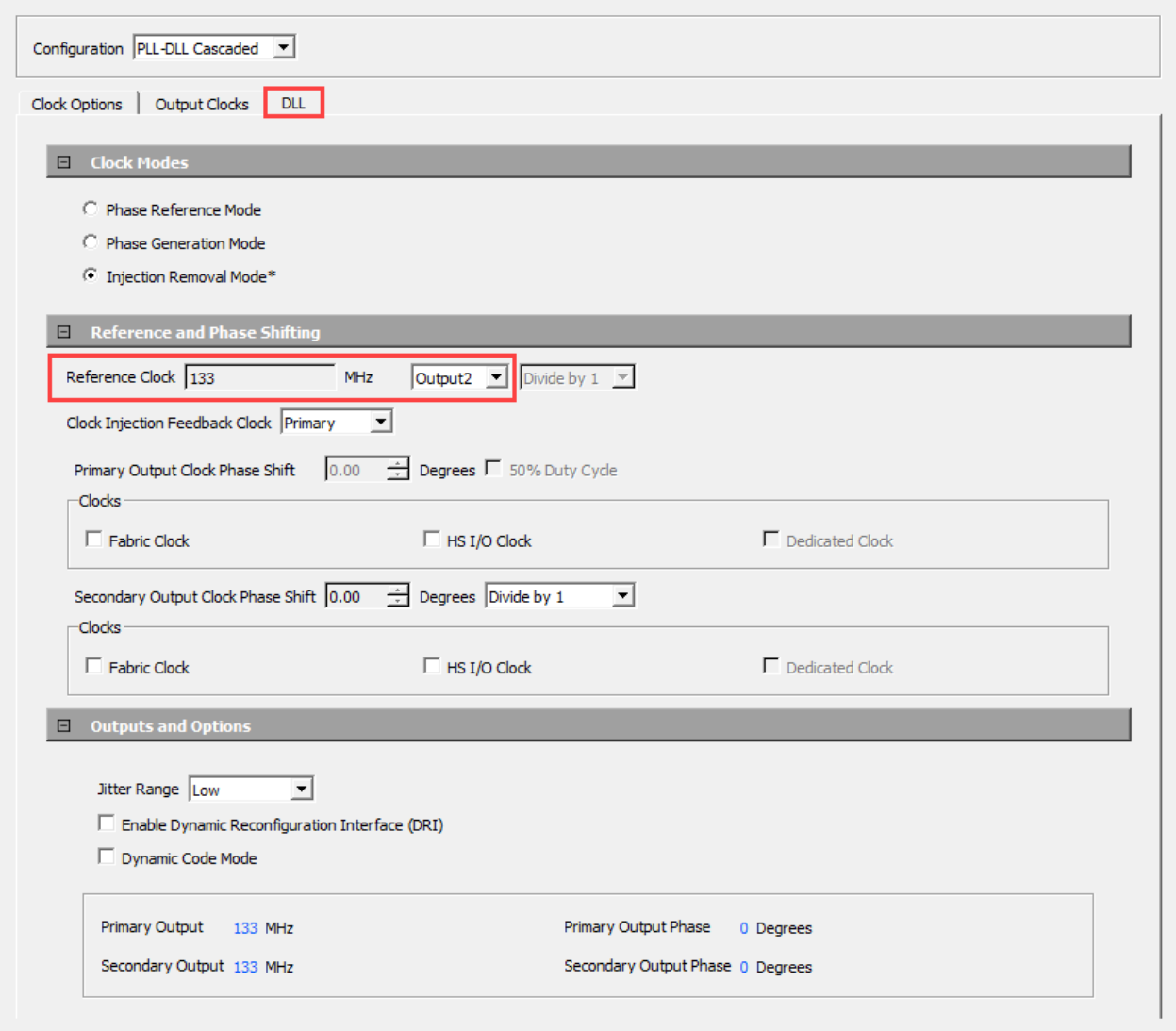
- **Phase Shift**: This feature is not available in this mode.
- **50% Duty Cycle**: This feature is not available in this mode.
- **Clock Division**: secondary clock output can be divided by 1, 2, or 4.
- Select secondary output clock connectivity:
  - **Fabric Clock**: Connects the primary output clock to the global clock network.
  - **HS I/O Clock**: Connects the primary output clock to a high-speed I/O clock network.
  - **Dedicated Clock**: This feature is not available in this mode.

Outputs and Options

- **Jitter Range**: Select between Low, Medium Low, Medium High, and High.
- Select **Enable Dynamic Reconfiguration Interface** to expose the bus interface for DLL dynamic configuration.
- Select **Export PowerDown Port** to expose the port to the fabric.

The following figure shows DLL reference clock selection under **PLL-DLL Cascaded** configuration. Select PLL **Output2** or **Output3** as reference clock to the DLL. Enter the frequency for selected reference clock (PLL Output2 or Output3) under **PLL Output Clocks** tab. The rest of PLL and DLL settings must be configured as explained in the preceding configurator.

Figure 4-28. PLL-DLL Cascading



4.7. CCC Simulation Support (Ask a Question)

Microchip Libero SoC provides pre-compiled simulation models for the CCC to show the functional behavior of the fabric CCC. The simulation steps include generating the top-level component, which instantiates CCC, performing simulation for verification with the ModelSim® tool, and performing static timing analysis with SmartTime in the Libero SoC.

Important:

- CCC simulation in post-divider mode does not work if POWERDOWN input is not driven or tied high.
- PLL is a fast simulation model and does not mimic the exact silicon behavior. The lock in simulation always comes after a fixed number of Phase Frequency Detector (PFD) cycles.

4.8. PLL/DLL Placement (Ask a Question)

Libero software automatically places the PLLs and DLLs as part of the Place and Route step.

To manually place the PLLs and DLLs, use the following PDC constraints for PLLs and DLLs placement:

PLL Placement

The `set_location` command has the following syntax:

```
set_location -inst_name <hierarchical inst name> -location <PLL location>
```

`-inst_name <hierarchical inst name>: specifies the hierarchical instance name.`

`-location <PLL location>: specifies the PLL location.`

The PLL location can be one of the following:

- PLL0_NW
- PLL1_NW
- PLL0_NE
- PLL1_NE
- PLL0_SW
- PLL1_SW
- PLL0_SE
- PLL1_SE

For example, `set_location -inst_name PF_CCC_0/pll_inst_0 -location PLL0_SE`

DLL Placement

The `set_location` command has the following syntax:

```
set_location -inst_name <hierarchical inst name> -location <DLL location>
```

`-inst_name <hierarchical inst name>: specifies the hierarchical instance name.`

`-location <DLL location>: specifies the DLL location.`

The DLL location can be one of the following:

- DLL0_NW
- DLL1_NW
- DLL0_NE
- DLL1_NE
- DLL0_SW
- DLL1_SW
- DLL0_SE
- DLL1_SE

For example, `set_location -inst_name PF_CCC_0/dll_inst_0 -location DLL0_SE`

4.9. Dynamic Configuration of CCC (Ask a Question)

Each CCC has a dynamic reconfiguration interface (DRI), which can be enabled to configure CCC parameters without reprogramming the device. The CCC configuration is controlled by volatile configuration registers that are loaded with values from the flash configuration bits at power-up. An APB bus master must be interfaced to the CCC using a DRI macro for dynamic configuration. The APB bus master is used to dynamically modify the CCC configuration register values as per design

needs. For more information on CCC configuration registers and their bit definitions, see PolarFire Device Register Map or PolarFire SoC Register Map.

To meet all the datasheet specifications, there are certain requirements that must be met when configuring the PLL/DLL parameters. The Libero CCC configurator implements all these requirements and creates a valid solution for the requested output clock frequencies and phases. Therefore, it is recommended that users generate the required configuration using the Libero CCC configurator and use the generated parameters in their dynamic configuration solution.

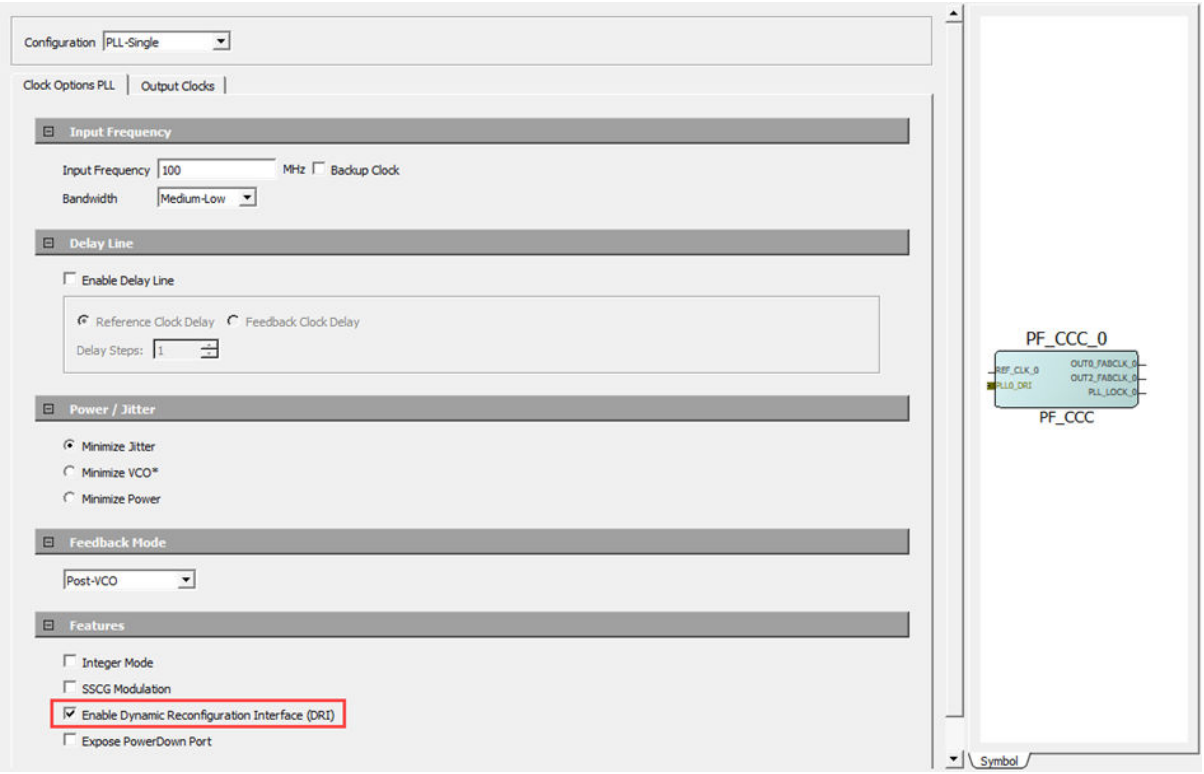
The PLL_POWERDOWN_N input must be asserted before making changes to the PLL configuration parameters. Asserting the PLL_POWERDOWN_N signal resets the PLL operation.

When the CCC is configured in the internal Post-VCO feedback mode, if the requirement is to change the phase or output divider configuration then the clock start/stop (OUT#_EN) signals can be used to stop the clock output before making the changes for glitchless configuration.

The following steps describe how to perform dynamic configuration of a CCC:

1. Select **Enable Dynamic Reconfiguration Interface** in the CCC configurator as shown in the following figure.

   Figure 4-29. Clock Conditioning Circuitry Window



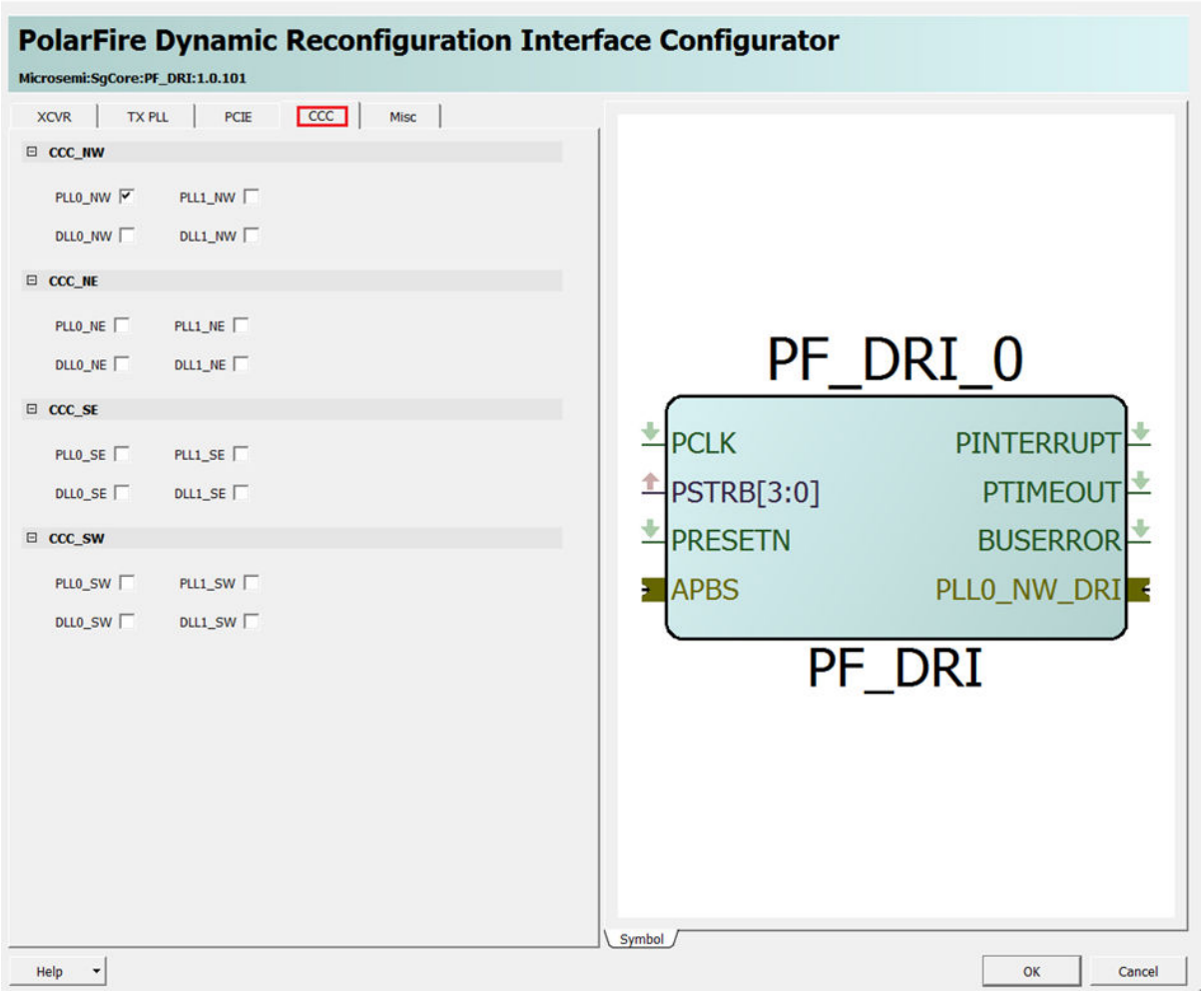
2. Instantiate a Dynamic Reconfiguration Interface macro into the SmartDesign. The dynamic reconfiguration interface macro converts the APB interface signals to CCC dynamic reconfiguration interface signals.
   The following table lists the ports for DRI. These ports are routed through hardwired connections to CCC. The DRI ports cannot be monitored or altered in the Libero design. These ports are used to facilitate HDL simulation of changes made to the CCC over the DRI. The DRI macro provided in the Libero Catalog takes care of converting standard APB3 read/writes to DRI transactions.

Table 4-9. DRI Port List

Port Name	Direction	Description
DRI_CLK	Input	Internal subsystem peripheral clock
DRI_WDATA[32:0]	Input	Write data
DRI_ARST_N	Input	Active low DRI asynchronous reset
DRI_CTRL[10:0]	Input	Control bits
DRI_RDATA[32:0]	Output	Read data
DRI_INTERRUPT	Output	Interrupt signal

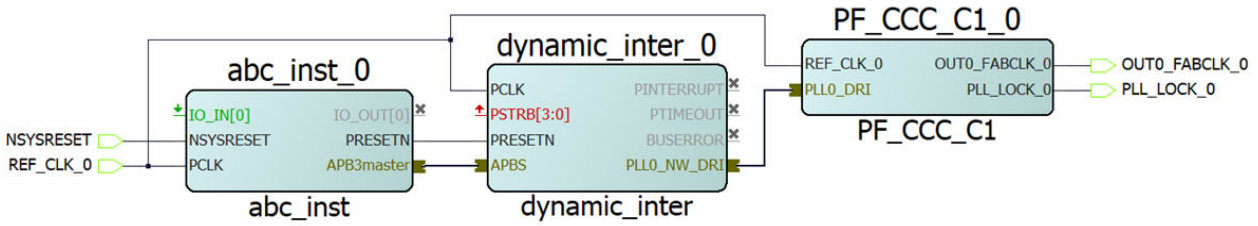
3. Double-click the **Dynamic Reconfiguration Interface** macro to configure.
4. In the macro configurator, under the **CCC** tab, select the PLLs and DLLs that need dynamic configuration. DRI macro interface is shown in the following figure.

Figure 4-30. Dynamic Reconfiguration Interface Configurator



5. Connect the APB master port from an APB master (for example, CoreABC) to the DRI macro's mirrored master port. See the following figure for connections.

Figure 4-31. CCC Dynamic Configuration System



Now, the APB master can dynamically configure the CCC configuration registers. For more information about how to use the DRI interface, see AN4592: PolarFire FPGA Dynamic Reconfiguration Interface Application Note (earlier AC475).

5. MSS Clock Controller (For PolarFire SoC and RT PolarFire SoC FPGA Only) (Ask a Question)

The PolarFire SoC and RT PolarFire SoC MSS have a dedicated clock controller for generating clocks to all the MSS sub-blocks for correct operation and synchronous communication with the user logic in the FPGA fabric. The MSS clock controller includes dedicated PLLs (MPLLs, DDR PLL, and SGMII PLL) for MSS clocking. The base clock for these PLLs comes either from a dedicated I/O (REFCLK) from Bank5 or one of the NW PLL outputs—OUT2 or OUT3. This dedicated I/O can be connected to a clock source, which can supply a 100 MHz or 125 MHz clock.

For MSS PLL radiation test results, refer to the Radiation Test Reports on the Microchip website (for example, the PolarFire SoC MSS Heavy Ion Test Results).

5.1. MSS Clocks (Ask a Question)

The DDR PLL is dedicated to MSS DDR operation, generating the necessary clocks required for the DDR controller and DDR PHY. The DDR memory clock frequency must be less than or equal to 666.66 MHz for DDR3 and 800 MHz for DDR4. The SGMII PLL is dedicated to SGMII operation, generating the necessary clocks required for an off-chip SGMII PHY.

The MPLL takes the input from one of two sources (REFCLK I/O or PLL_NW outputs) and generates a master input clock (`clk_in_mss`). The `clk_in_mss` clock is used to generate the MSS clocks as listed in the following table.

Table 5-1. MSS Clocks—1

Clock Name	Description	Maximum Operating Frequency	Possible MPLL Output Division Ratios
CPU core clock (clk_cpu)	Clocks all the user processor cores	625 MHz	1, 2, 4, or 8
MSS AXI clock (clk_axi)	Clocks MSS AXI buses and peripherals	312.5 MHz	1, 2, 4, or 8
MSS AHB and APB clock (clk_ahb)	Clocks MSS AHB and APB buses and peripherals	156.25 MHz	2, 4, or 8

All the clocks shown in the preceding table are synchronous to each other and divided from the MPLL output with appropriate division values such that:

- CPU core clock must be greater than or equal to MSS AXI clock
- MSS AXI clock must be greater than or equal to MSS AHB and APB clock

The MPLL also generates the following clocks listed in the table.

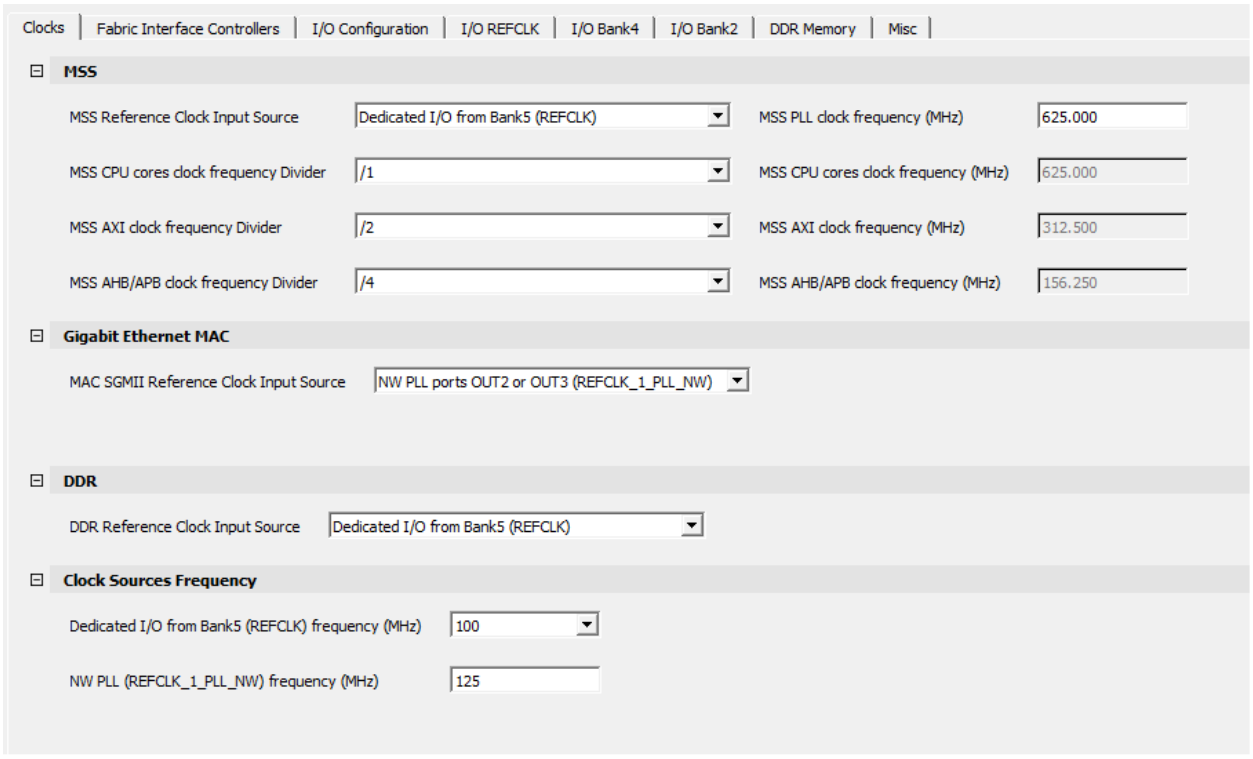
Table 5-2. MSS Clocks—2

Clock Name	Description	Maximum Operating Frequency	Notes
Crypto clock (clk_in_crypto)	Clocks User Cryptoprocessor in MSS mode	200 MHz	Configurable between 1 and 200 MHz
CAN clock (clk_in_clk)	Clocks MSS CAN controllers	80 MHz	Must be a multiple of 8 MHz
eMMC/SD/SDIO clock (clk_in_emmc)	Clocks MSS eMMC/SD/SDIO controller	200 MHz	Not configurable

At power-on and after MSS reset, the MSS is clocked from the on-chip 80 MHz RC oscillator with CPU/AXI dividers set to 1 and the AHB/APB dividers set to 2. Embedded software, running on the E51 processor core, switches the MSS clock source dynamically to the user configuration.

The MSS Configurator provides a single place where all clocks related to the MSS can be configured. For more information, see PolarFire SoC Standalone MSS Configurator User Guide.

Figure 5-1. MSS Configurator—Clocks Configuration



5.2. FPGA Fabric Interface Clocks (Ask a Question)

PolarFire SoC and RT PolarFire SoC FPGAs provide multiple Fabric Interface Controllers (FIC) to enable connectivity between user logic in the FPGA fabric and the MSS. FIC is part of the MSS and acts as a bridge between MSS and the fabric. There are three 64-bit AXI4 FICs, one 32-bit APB interface FIC, and one 32-bit AHB-Lite interface FIC, see the following table.

Table 5-3. FICs in PolarFire SoC and RT PolarFire SoC FPGA

FIC Interface	Description
FIC0 and FIC1	Each FIC provides two 64-bit AXI4 bus interfaces between the MSS and the Fabric. One of them is mastered by the MSS and has slaves in the fabric, the other is mastered by the fabric and has slaves in the MSS. Only **FIC1** can be used for data transfers to or from the PCIe Controller hard block in the FPGA.
FIC2	Provides a single 64-bit AXI4 bus interface between the MSS and the fabric. It is mastered by the fabric and has slaves in the MSS. It is only used to access non-cached DDR memory through the DDR controller inside the MSS block.
FIC3	Provides a single 32-bit APB bus interface between the MSS and the fabric. It is mastered by the MSS and has slaves in the fabric. It can be used to configure PCIe and XCVR hard blocks.
FIC4	This FIC is dedicated to interface with the User Crypto Processor. This provides two 32-bit AHB-Lite bus interfaces between the Crypto Processor and the fabric. One of them is mastered by the fabric and the crypto processor acts as a slave. The other is mastered by the DMA controller of the User Crypto Processor and has a slave in the fabric.

Each FIC can operate on a different clock frequency, defined as a ratio of the MSS main clock. The FIC is a hard block, which also contains a DLL, enabling or disabling it does not consume any user logic. If the frequency of the FIC block is greater than or equal to 125 MHz, then the DLL must be enabled for removing clock insertion delay. If the frequency of the FIC block is less than 125 MHz, then the DLL must be bypassed.

The Fabric side AXI interface of the FIC blocks; FIC0, FIC1, FIC2, and FIC3 can operate up to 250 MHz and the MSS side AXI interface of the FIC blocks can operate up to 312.5 MHz. The FIC4 can operate

up to 200 MHz. The MSS and Fabric clocks are asynchronous in nature and the FIC block takes care of clock domain crossing.

6. Revision History (Ask a Question)

The revision history table describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Table 6-1. Revision History

Revision	Date	Description
J	05/2025	The following is a summary of the changes made in this revision: • Updated the document for RT PolarFire® SoC support. The changes are made throughout the document. • Updated the descriptions of DELAY_LINE_ DIRECTION and DELAY_LINE_WIDE ports in Table 4-1.
H	11/2024	The following is a summary of the changes made in this revision: • Removed the following sentence from Global Clock Network: The skew is guaranteed to be less than the shortest possible propagation delay. • Added information about lock signal assertion and de-assertion in the Lock Output section. • Updated Figure 2-1 and Figure 4-2 by removing the arrow going from the Transceiver block to CCC NE for indicating that the Transceiver block can drive clock to CCC SE and not to CCC NE. • Added a note about the simulation support limitation in the CCC Simulation Support section.
G	05/2024	The following is a summary of the changes made in this revision: • Added a note about the unused condition of the on-chip oscillators in the On-Chip Oscillators section. • Added information about the radiation test reports which must be used to decide the CCC PLL usage in RT PolarFire devices. See the Phase-Locked Loops section. • Added a precautionary note in the Clock Frequency Synthesis section.
F	11/2023	Added a note about the usage of 2 MHz RC Oscillator. See On-Chip Oscillators.
E	05/2023	The following is a summary of the changes made in this revision: • Updated the document title and added RT PolarFire information. • Updated information about PLL_POWERDOWN_B control of CCC when using feedback modes. See Power-Down Input, Internal Post-Divider Feedback Mode, and External Feedback Mode.
D	12/2022	The following is a summary of the changes made in this revision: • Updated note about static phase shift in post-divider and external feedback modes. See Static Phase Shifting. • Added a note about post-VCO mode and post-divider mode. See Dynamic Phase Shifting. • Updated information about integer mode, See CCC Configuration.
C	08/2022	Added information about Resets Outputs On PLL Lock and PLL Lock. See CCC Configuration.
B	04/2022	The following is a summary of the changes made in this revision: • Added information about Global clock networks susceptible to clock jitter. See Global Net Clock Jitter. • Added a note about clock jitter constraints. See Managing Global Signals and Clock Outputs.

Table 6-1. Revision History (continued)

Revision	Date	Description
A	08/2021	The first publication of the document. This user guide was created by merging the following documents: • UG0684: PolarFire FPGA Clocking Resources User Guide • UG0913: PolarFire SoC FPGA Clocking Resources User Guide The revision history tables of both the user guides are retained for the future reference. See Table 6-2 and Table 6-3.

The following revision history table describes the changes that were implemented in the *UG0684: PolarFire FPGA Clocking Resources User Guide*. The changes are listed by revision.

Note: *UG0684: PolarFire FPGA Clocking Resources User Guide* is now obsolete and the information in the document has been migrated to *PolarFire® FPGA and PolarFire SoC FPGA Clocking Resources User Guide* (this document).

Table 6-2. Revision History of UG0684: PolarFire FPGA Clocking Resources User Guide

Revision	Date	Description
Revision 9.0	9/20	The following is a summary of the changes made in this revision: • Information about CCC Configuration was updated. • Changed PSTRB signal from low to high. See CCC Dynamic Configuration System figure. • Information about BYPASS_EN_N was removed.
Revision 8.0	4/20	The following is a summary of the changes made in this revision: • Information about External Feedback Mode was updated. • Information about CCC Configuration was updated. • Information about Dynamic Phase Shifting was updated. • Information about Preferred Clock Inputs Connectivity in CCCs was updated.
Revision 7.0	8/19	The following is a summary of the changes made in this revision: • Information about performance of global clock for the transceiver quads was removed and referred to *PolarFire FPGA Datasheet*. • Information about Requested Phase field of CCC configurator was added. See CCC Configuration. • Information about PLL Reference Clock Inputs was updated. See Reference Clock Inputs. • Information about RC Oscillators Configurator Ports was added. • Information about Reference Clock Inputs was updated. • Information related to Libero SoC v12.1 was updated.
Revision 6.0	4/19	Information about minimum input reference frequency for DLL was updated. See DLL Operational Modes.
Revision 5.0	10/18	The following is a summary of the changes made in this revision: • Information about PLL reference clock was added. See Reference Clock Inputs. • Port names were matched according to Libero SoC PolarFire v2.3. • Information about Internal Post-VCO Feedback Mode, Internal Post-Divider Feedback Mode, and External Feedback Mode was updated.
Revision 4.0	5/18	The following is a summary of the changes made in this revision: • Information about interface clock block was added. See Interface Clock Block. • Information about global clock networks was updated. See Global Clock Network. • Information about CCC_SW_CLKIN_W<3:0> was added. • Information about BIT_SLIP signal was added. See Clock Dividers. • Information about bandwidth adjustment was updated. See PLL Bandwidth Parameter Settings table.

Table 6-2. Revision History of UG0684: PolarFire FPGA Clocking Resources User Guide (continued)

Revision	Date	Description
Revision 3.0	12/17	The following is a summary of the changes made in this revision: • Information about global clock networks was updated. • The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v2.0 release.
Revision 2.0	7/17	The following is a summary of the changes made in this revision: • Information about PLL/DLL Placement was added. See PLL/DLL Placement. • Global Clock Network and Clock Sources figure, and System-Level Block Diagram of CCCs figure were updated.
Revision 1.0	2/17	The first publication of UG0684: PolarFire FPGA Clocking Resources User Guide.

The following revision history table describes the changes that were implemented in the UG0913: PolarFire SoC FPGA Clocking Resources User Guide document. The changes are listed by revision.

Note: UG0913: PolarFire SoC FPGA Clocking Resources User Guide document is now obsolete and the information in the document has been migrated to PolarFire® FPGA and PolarFire SoC FPGA Clocking Resources User Guide.

Table 6-3. Revision History of UG0913: PolarFire SoC FPGA Clocking Resources User Guide

Revision	Date	Description
Revision 3.0	7/21	The following is a summary of the changes made in this revision: • Information about Preferred Clock Inputs Connectivity in CCCs was updated. • Information about Glitch-Free Clock Switching was updated.
Revision 2.0	9/20	The following is a summary of the changes made in this revision: • Information about CCC Configuration was updated. • Changed PSTRB signal from low to high. See CCC Dynamic Configuration System figure. • Information about BYPASS_EN_N was removed.
Revision 1.0	4/20	The first publication of UG0913: PolarFire SoC FPGA Clocking Resources User Guide.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-1296-1

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.