# Contents

1 Revision History .......................................................... 1
  1.1 Revision 5.0 .......................................................... 1
  1.2 Revision 4.0 .......................................................... 1
  1.3 Revision 3.0 .......................................................... 1
  1.4 Revision 2.0 .......................................................... 1
  1.5 Revision 1.0 .......................................................... 1

2 Clocking Features Overview ............................................. 2
  2.1 Clock Routing Resources ............................................... 2
  2.2 On-Chip Oscillators .................................................. 2
  2.3 Clock Conditioning Circuitry ........................................ 2

3 Clock Routing Resources ................................................ 3
  3.1 Global Clock Network ............................................... 3
  3.2 Regional Clock Networks ........................................... 6
  3.3 High-Speed I/O Clock Networks ..................................... 8
  3.4 Preferred Clock Inputs .............................................. 9
      3.4.1 Naming Convention ........................................... 10
      3.4.2 Preferred Clock Inputs Connectivity in CCCs ............ 10
  3.5 Interface Clock Block ............................................... 11
      3.5.1 Clock Dividers ............................................... 12
      3.5.2 Glitch-Free Clock Switching ................................. 13
  3.6 Clock Gating ........................................................ 14
  3.7 Clock Macros ...................................................... 15
  3.8 Managing Global Signals ............................................ 16

4 On-Chip Oscillators ..................................................... 19

5 Clock Conditioning Circuitry ........................................... 20
  5.1 Features ............................................................ 20
  5.2 CCC Locations and Clocking Capabilities .......................... 22
  5.3 Phase-Locked Loops ................................................ 23
      5.3.1 PLL Port Description ........................................ 23
      5.3.2 PLL Operational Modes ...................................... 29
      5.3.3 Spread Spectrum Clock Generation ......................... 31
      5.3.4 PLL Use Models .............................................. 33
  5.4 Delay-Locked Loops ................................................ 34
      5.4.1 DLL Operational Modes ...................................... 35
  5.5 PLL/DLL Cascading ................................................ 39
      5.5.1 PLL-to-PLL Cascading ....................................... 39
      5.5.2 PLL Driving DLL ............................................. 39
  5.6 CCC Configuration ................................................ 40
  5.7 CCC Simulation Support ............................................ 47
  5.8 PLL/DLL Placement ................................................ 48
  5.9 Dynamic Configuration of CCC ..................................... 48
      5.9.1 CCC Configuration Registers ............................... 51
      5.9.2 PLL Configuration Registers Bit Definitions ............. 52
      5.9.3 DLL Configuration Registers Bit Definitions ............ 55
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Global Clock Network and Clock Sources</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>FPGA Fabric—Global Clock Routing Architecture</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Sector Representation</td>
<td>5</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Regional Clock Buffers</td>
<td>6</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Regional Clock Buffer Fan-outs from Bottom-Right I/O Lanes</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Regional Clock Buffer Fan-outs from Transceiver</td>
<td>7</td>
</tr>
<tr>
<td>Figure 7</td>
<td>High-Speed I/O Clock Networks</td>
<td>8</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Preferred Clock Inputs</td>
<td>9</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Using the Reset and Slip Operations for the ICB Dividers</td>
<td>12</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Clock Divider</td>
<td>12</td>
</tr>
<tr>
<td>Figure 11</td>
<td>NGMUX Symbol</td>
<td>13</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Mode 0: Clock Switching when Clocks are Active</td>
<td>13</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Mode 1: Clock Switching when Current Clock (CLK0) is not Active</td>
<td>13</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Mode 1: Clock Switching when Current Clock (CLK0) is Uncertain with Random Pulses</td>
<td>14</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Clock Gating Circuit Schematic</td>
<td>14</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Timing Waveforms for the Clock Gating Circuity</td>
<td>14</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Synthesize Options Dialog Box</td>
<td>17</td>
</tr>
<tr>
<td>Figure 18</td>
<td>RC Oscillators Configurator</td>
<td>19</td>
</tr>
<tr>
<td>Figure 19</td>
<td>CCC Block Diagram</td>
<td>21</td>
</tr>
<tr>
<td>Figure 20</td>
<td>System-Level Block Diagram of CCCs</td>
<td>22</td>
</tr>
<tr>
<td>Figure 21</td>
<td>PLL Block Diagram</td>
<td>23</td>
</tr>
<tr>
<td>Figure 22</td>
<td>PLL Block Diagram—Clock Inputs and Outputs</td>
<td>25</td>
</tr>
<tr>
<td>Figure 23</td>
<td>PLL—Bypass Controls</td>
<td>26</td>
</tr>
<tr>
<td>Figure 24</td>
<td>PLL—Delay Line Controls</td>
<td>27</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Example of Using VCO/4 Delay and VCO/4 Phase Select to Fine Tune PLL Output Phase</td>
<td>28</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Internal Post-VCO Feedback Mode</td>
<td>30</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Internal Post-Divider Feedback Mode</td>
<td>30</td>
</tr>
<tr>
<td>Figure 28</td>
<td>External Feedback Mode—Feedback through Global Clock Network</td>
<td>31</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Spread Spectrum in Time and Frequency Domain Using Triangular-Modulated Waveform</td>
<td>32</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Frequency Synthesis</td>
<td>33</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Zero-Delay Buffer—Phase Relationship Between Clocks</td>
<td>33</td>
</tr>
<tr>
<td>Figure 32</td>
<td>PolarFire DLL Block Diagram</td>
<td>34</td>
</tr>
<tr>
<td>Figure 33</td>
<td>DLL Ports—Phase Reference Mode</td>
<td>35</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Phase Reference Mode—Dynamic Configuration</td>
<td>36</td>
</tr>
<tr>
<td>Figure 35</td>
<td>DLL Ports—Phase Generation Mode</td>
<td>36</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Phase Generation Mode—DLL_CLK_1 Dynamic Configuration</td>
<td>37</td>
</tr>
<tr>
<td>Figure 37</td>
<td>DLL Ports—Clock Injection Delay Removal Mode</td>
<td>38</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Clock Injection Delay Removal Mode</td>
<td>38</td>
</tr>
<tr>
<td>Figure 39</td>
<td>CCC Block Diagram</td>
<td>39</td>
</tr>
<tr>
<td>Figure 40</td>
<td>CCC Configurator—Configuration Options</td>
<td>40</td>
</tr>
<tr>
<td>Figure 41</td>
<td>PLL-Single Configuration—Clock Options PLL</td>
<td>40</td>
</tr>
<tr>
<td>Figure 42</td>
<td>PLL-Single Configuration—Output Clocks</td>
<td>42</td>
</tr>
<tr>
<td>Figure 43</td>
<td>PLL-Single Configuration—SSCG Modulation</td>
<td>43</td>
</tr>
<tr>
<td>Figure 44</td>
<td>DLL Configuration—Phase Reference Mode</td>
<td>44</td>
</tr>
<tr>
<td>Figure 45</td>
<td>DLL Configuration—Phase Generation Mode</td>
<td>45</td>
</tr>
<tr>
<td>Figure 46</td>
<td>DLL Configuration—Injection Removal Mode</td>
<td>46</td>
</tr>
<tr>
<td>Figure 47</td>
<td>PLL-DLL Cascading</td>
<td>47</td>
</tr>
<tr>
<td>Figure 48</td>
<td>Clock Conditioning Circuity Window</td>
<td>49</td>
</tr>
<tr>
<td>Figure 49</td>
<td>Dynamic Reconfiguration Interface Configurator</td>
<td>50</td>
</tr>
<tr>
<td>Figure 50</td>
<td>CCC Dynamic Configuration System</td>
<td>51</td>
</tr>
</tbody>
</table>
Tables

Table 1  Preferred Clock Inputs Connectivity to PLLs and DLLs .............................................. 10
Table 2  Clock Macros ............................................................................................................. 15
Table 3  PLL Port List ............................................................................................................. 23
Table 4  Truth Table for Enabling of PLL Outputs ................................................................. 26
Table 5  PLL Bandwidth Parameter Settings ........................................................................... 27
Table 6  Dynamic Phase Shift Ports ......................................................................................... 29
Table 7  Center and Down Spread Modulation Depths Based on SPREAD Value .................... 32
Table 8  DLL Port List—Phase Reference Mode ...................................................................... 35
Table 9  DLL Port List—Phase Generation Mode ..................................................................... 36
Table 10 DLL Port List—Clock Injection Delay Removal Mode .............................................. 38
Table 11 PLL Register Map .................................................................................................... 51
Table 12 SOFT_RESET—0x00 ............................................................................................... 52
Table 13 PLL_CTRL—0x04 ..................................................................................................... 52
Table 14 DLL Register Map ................................................................................................... 52
Table 15 PLL_REF_FB—0x08 ............................................................................................... 53
Table 16 PLL_DIV_0_1—0x10 .............................................................................................. 53
Table 17 PLL_DIV_2_3—0x14 .............................................................................................. 54
Table 18 PLL_CTRL2—0x18 .................................................................................................... 54
Table 19 PLL_PHADJ—0x20 ................................................................................................... 54
Table 20 DLL Control Register 0—0x00 .................................................................................. 55
Table 21 DLL Control Register 1—0x04 .................................................................................. 57
Table 22 DLL Status Register 0—0x08 ..................................................................................... 57
1 Revision History

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

1.1 Revision 5.0

The following is a summary of the changes made in revision 5.0 of this document.

- Information about PLL reference clock was added. See Reference Clock Inputs, page 24.
- Port names were matched according to Libero SoC PolarFire v2.3. See Table 3, page 23, Table 8, page 35, Table 9, page 36, and Table 10, page 38.
- Information about Internal Post-VCO Feedback Mode, page 30, Internal Post-Divider Feedback Mode, page 30, and External Feedback Mode, page 31 was updated.

1.2 Revision 4.0

The following is a summary of the changes made in revision 4.0 of this document.

- Information about interface clock block was added. See Interface Clock Block, page 11.
- Information about global clock networks was updated. See Global Clock Network, page 3.
- Information about CCC_SW_CLKIN_<3:0> was added. See Table 1, page 10.
- Information about BIT_SLIP signal was added. See Clock Dividers, page 12.
- Information about bandwidth adjustment was updated. See Table 5, page 27.
- Updated Figure 1, page 3, Figure 4, page 6, Figure 7, page 8, and Figure 19, page 21.

1.3 Revision 3.0

The following is a summary of the changes made in revision 3.0 of this document.

- Information about global clock networks was updated. See Global Clock Network, page 3.
- The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v2.0 release.

1.4 Revision 2.0

The following is a summary of the changes made in revision 2.0 of this document.

- Information about PLL/DLL Placement was added. See PLL/DLL Placement, page 48.
- Figure 1, page 3, Figure 8, page 9, and Figure 20, page 22 were updated.

1.5 Revision 1.0

The first publication of this document.
Clocking features are crucial to FPGA architectures. PolarFire® FPGAs include an abundant selection of robust clocking resources, classified as:

- Clock routing resources
- On-chip oscillators
- Clock conditioning circuitry (CCC): PLLs and DLLs

### 2.1 Clock Routing Resources

To enable efficient clock distribution, PolarFire FPGAs have 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.

**Note:** PolarFire FPGAs offer 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.

### 2.2 On-Chip Oscillators

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

### 2.3 Clock Conditioning Circuitry

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

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

PolarFire FPGAs contain low-skew clock networks, and preferred clock inputs and outputs for efficient clock distribution.

3.1 Global Clock Network

The global clock network is used to distribute high fanout signals such as clocks and resets across the FPGA fabric with low-skew, using a vertical and horizontal clock stripe architecture. The skew is guaranteed to be less than the shortest possible propagation delay.

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

**Note:** Only one global clock is supported per transceiver quad and the maximum frequency supported is 325 MHz for both STD and -1 devices. See DS0141: PolarFire FPGA Datasheet for more information.

**Figure 1 • Global Clock Network and Clock Sources**
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 PolarFire 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 • 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, page 4, 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 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, see Clock Gating, page 14 for more information. 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 designers 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 fanout of each signal. Synthesis tools attempt to assign the low-skew resources to clock signals with high fanout. To improve the performance, designers can take control of the clock network assignment by instantiating the required macros or attributes in the design. See Clock Macros, page 15 for a list of macros supported in PolarFire devices.

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

Regional clock networks exhibit lower latency than the global clock network. They comprise of 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.

Each regional clock buffer drives all of 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 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.

**Figure 6 • Regional Clock Buffer Fan-outs from Transceiver**

PolarFire FPGAs offer 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 PolarFire I/Os support specific I/O interface modes (for example, DDRxn modes) that are designed to use regional clocks. See the PolarFire I/O Interface Modes section of UG0686: PolarFire FPGA User I/O User Guide for more information. When using the I/O interface modes that support regional clocks, Libero SoC PolarFire automatically routes clocks coming from I/O lanes on the regional clock networks.

3.3 High-Speed I/O Clock Networks

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 into 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 7 • High-Speed I/O Clock Networks
The CCC can be configured to have a PLL or DLL clock output driving a high-speed I/O clock network. The PolarFire I/Os support specific I/O interface modes (for example, DDRxn modes) that are designed to use the high-speed I/O clock networks. See PolarFire I/O Interface Modes section of UG0686: PolarFire FPGA User I/O User Guide for more information. When using I/O interface modes that support high-speed I/O clock networks, the Libero SoC PolarFire automatically routes the clocks coming from I/O lanes on the high-speed I/O clock networks.

### 3.4 Preferred Clock Inputs

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 side 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 following figure. Some of the dedicated clock inputs have access to both CCC and the global clock network and can be connected to either.

*Figure 8 • Preferred Clock Inputs*
3.4.1 Naming Convention

CLKIN\_z\_w and CCC\_xy\_CLKIN\_z\_w represents 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 location of the Preferred clock input: N, W, or S.
- \(w\)—represents the preferred clock input number: 0 to 15.

3.4.2 Preferred Clock Inputs Connectivity in CCCs

Preferred clock inputs can be configured as reference clock or feedback clock to the PLLs and DLLs present in each CCC.

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—RFCLK0 and RFCLK1.

The following table lists the connectivity of preferred clock inputs to PLLs and DLLs.

<table>
<thead>
<tr>
<th>Preferred Clock Input</th>
<th>PLL Connectivity</th>
<th>DLL Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RFCLK0</td>
<td>RFCLK1</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_W_4</td>
<td>PLL0_NW, PLL1_NW</td>
<td>PLL0_NW, PLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_W_5</td>
<td>PLL0_NW, PLL1_NW</td>
<td>PLL0_NW, PLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_W_6</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_W_7</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_W_0</td>
<td>PLL0_SW, PLL1_SW</td>
<td>PLL0_SW, PLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_W_1</td>
<td>PLL0_SW, PLL1_SW</td>
<td>PLL0_SW, PLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_W_2</td>
<td>PLL0_SW, PLL1_SW</td>
<td>DLL0_SW, DLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_W_3</td>
<td>PLL0_SW, PLL1_SW</td>
<td>DLL0_SW, DLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_S_0</td>
<td>PLL0_SW, PLL1_SW</td>
<td>PLL0_SW, PLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_S_1</td>
<td>PLL0_SW, PLL1_SW</td>
<td>PLL0_SW, PLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_S_2</td>
<td>PLL0_SW, PLL1_SW</td>
<td>DLL0_SW, DLL1_SW</td>
</tr>
<tr>
<td>CCC_SW_CLKIN_S_3</td>
<td>PLL0_SW, PLL1_SW</td>
<td>DLL0_SW, DLL1_SW</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_8</td>
<td>PLL0_SE, PLL1_SE</td>
<td>PLL0_SE, PLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_9</td>
<td>PLL0_SE, PLL1_SE</td>
<td>PLL0_SE, PLL1_SE</td>
</tr>
</tbody>
</table>
Table 1 • Preferred Clock Inputs Connectivity to PLLs and DLLs (continued)

<table>
<thead>
<tr>
<th>Preferred Clock Input</th>
<th>PLL Connectivity</th>
<th>DLL Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCC_SE_CLKIN_S_10</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_11</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_12</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_13</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_14</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_SE_CLKIN_S_15</td>
<td>PLL0_SE, PLL1_SE</td>
<td>DLL0_SE, DLL1_SE</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_0</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_1</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_2</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_3</td>
<td>PLL0_NW, PLL1_NW</td>
<td>DLL0_NW, DLL1_NW</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_8</td>
<td>PLL0_NE, PLL1_NE</td>
<td>DLL0_NE, DLL1_NE</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_9</td>
<td>PLL0_NE, PLL1_NE</td>
<td>DLL0_NE, DLL1_NE</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_10</td>
<td>PLL0_NE, PLL1_NE</td>
<td>DLL0_NE, DLL1_NE</td>
</tr>
<tr>
<td>CCC_NW_CLKIN_N_11</td>
<td>PLL0_NE, PLL1_NE</td>
<td>DLL0_NE, DLL1_NE</td>
</tr>
</tbody>
</table>

3.5 Interface Clock Block

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 are hit with the limitation of number of ICB_INT cells per ICB, use dedicated connections from CCCs, preferred clock inputs, and oscillators, which does not require ICB_INT cells. The following sections describes the clock dividers and NGMUXs present in the ICBs.
3.5.1 Clock Dividers

PolarFire FPGAs provide 24 programmable clock dividers at the center of the edge on four sides of the device. Libero SoC PolarFire 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. 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 control signal (BIT_SLIP). 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 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 9 • Using the Reset and Slip Operations for the ICB Dividers](image)

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 PolarFire and are programmed during device programming.

![Figure 10 • Clock Divider](image)
### 3.5.2 Glitch-Free Clock Switching

PolarFire FPGAs have 12 No-Glitch MUXes (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 PolarFire selects the appropriate NGMUX based on the clock source. NGMUX is accessible by instantiating NGMUX configurator in the design. The following figure shows the NGMUX symbol.

![NGMUX Symbol](image)

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 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.

![Mode 0: Clock Switching when Clocks are Active](image)

![Mode 1: Clock Switching when Current Clock (CLK0) is not Active](image)
3.6 Clock Gating

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.

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 low phase. The latch is in a hold state when the clock is in 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. See DS0141: PolarFire FPGA Datasheet, for the minimum setup and hold time for the clock gating enable signal.

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

Figure 15 • Clock Gating Circuit Schematic

Figure 16 • Timing Waveforms 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, 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, EN signal must arrive by prior rising edge (earlier than usual). Unused global resources such as: RGBs and GBs are tied-off automatically to reduce the dynamic power consumption.

### 3.7 Clock Macros

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

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Description</th>
<th>Functional Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKINT</td>
<td>Routes clock signal from the fabric routing or regular I/O to global clock network.</td>
<td>![Functional Symbol](A --- Y)</td>
</tr>
<tr>
<td>CLKINT_PRESERVE</td>
<td>Similar to CLKINT; but the signals routed through the CLKINT_PRESERVE macro will never be demoted or optimized by the Libero Compile tool.</td>
<td>![Functional Symbol](A --- Y)</td>
</tr>
<tr>
<td>GCLKINT</td>
<td>Gated version of CLKINT with an enable signal (EN) from the FPGA fabric to gate the clock.</td>
<td>![Functional Symbol](A --- Y)</td>
</tr>
<tr>
<td>RCLKINT</td>
<td>Routes clock signal from global buffers, regional clock buffers, or fabric routing to RGBs.</td>
<td>![Functional Symbol](A --- Y)</td>
</tr>
<tr>
<td>RGCLKINT</td>
<td>Gated version of RCLKINT with an enable signal (EN) from FPGA fabric to gate the clock.</td>
<td>![Functional Symbol](A --- Y)</td>
</tr>
<tr>
<td>CLKBUF</td>
<td>Drives a global buffer from preferred clock input pad.</td>
<td>![Functional Symbol](PAD --- Y)</td>
</tr>
</tbody>
</table>
3.8 Managing Global Signals

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 fabric routed signal to 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 PolarFire supports automated global buffer allocation to minimize the 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 PolarFire, 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 reduce 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 fanout threshold of asynchronous pins above the fanout of the asynchronous net. Alternatively, you can manually adjust the RTL by moving timing-critical pins from high-fanout asynchronous nets to lower fanout 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 PolarFire 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 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 could 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 fanout 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 fanout 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-fanout may have logic replication, if it is not promoted to a global during synthesis.
4 On-Chip Oscillators

PolarFire FPGAs offer 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 DS0141: PolarFire 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.

The on-chip oscillators' clocks can be connected to the global clock network, clock dividers, and NGMUXs through hardwired routing. The on-chip oscillator clocks can be used as CCC reference clock through global clock network. The on-chip oscillators are located at lower left corner of the device, see Figure 1, page 3. The on-chip oscillators are accessible using the RC oscillators configurator in Libero SoC PolarFire. Unused on-chip oscillators are automatically disabled.

Figure 18 • RC Oscillators Configurator
5 Clock Conditioning Circuitry

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 PolarFire software provides a CCC configurator with a visual configuration wizard for quick and easy configuration. For DC and switching characteristics of the CCCs, see DS0141: PolarFire FPGA Datasheet.

5.1 Features

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 19 • CCC Block Diagram
5.2 **CCC Locations and Clocking Capabilities**

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.

*Figure 20 • System-Level Block Diagram of CCCs*
5.3 Phase-Locked Loops

PolarFire PLLs can be 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, PolarFire PLLs offer spread-spectrum capabilities to minimize the EMI.

Figure 21 • PLL Block Diagram

5.3.1 PLL Port Description

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 3 • PLL Port List

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF_CLK_0</td>
<td>Input</td>
<td>Reference clock.</td>
</tr>
<tr>
<td>REF_CLK_1</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>REF_CLK_SEL</td>
<td>Input</td>
<td>Reference clock select:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b0—REF_CLK_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b1—REF_CLK_1</td>
</tr>
<tr>
<td>FB_CLK</td>
<td>Input</td>
<td>Feedback clock—Exposed in PLL external feedback mode only.</td>
</tr>
<tr>
<td>OUT&lt;3:0&gt;</td>
<td>Output</td>
<td>Clock outputs</td>
</tr>
<tr>
<td>PLL_LOCK</td>
<td>Output</td>
<td>Lock output</td>
</tr>
<tr>
<td>PLL_POWERDOWN_</td>
<td>Input</td>
<td>Power down signal (active low):</td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>1'b0—Power down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b1—PLL enabled</td>
</tr>
<tr>
<td>OUT0_EN</td>
<td>Input</td>
<td>OUT0 output enable</td>
</tr>
<tr>
<td>OUT1_EN</td>
<td>Input</td>
<td>OUT1 output enable</td>
</tr>
<tr>
<td>OUT2_EN</td>
<td>Input</td>
<td>OUT2 output enable</td>
</tr>
<tr>
<td>OUT3_EN</td>
<td>Input</td>
<td>OUT3 output enable</td>
</tr>
<tr>
<td>BYPASS_EN_N</td>
<td>Input</td>
<td>Bypass MUXes enable (active low)—Enables bypass MUXes present at output dividers. The bypass MUXes configuration must be done through CCC configurator.</td>
</tr>
</tbody>
</table>
5.3.1.1 Reference Clock Inputs

PolarFire PLLs have two reference clock inputs (REF.CLK.0 and REF.CLK.1) and supports 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.

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)

Table 3 • PLL Port List (continued)

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFCLK_SYNC_EN</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>DELAY_LINE_MOVE</td>
<td>Input</td>
<td>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.</td>
</tr>
</tbody>
</table>
| DELAY_LINE_DIRECTION| Input     | 0—Decrements the delay taps by 1 or 2 based on DELAY_LINE_WIDE  
1—Increments the delay taps by 1 or 2 based on DELAY_LINE_WIDE  |
| DELAY_LINE_WIDE     | Input     | 0—Increments or Decrement the delay taps by 1  
1—Increments or Decrement the delay taps by 2  |
| DELAY_LINE_LOAD     | Input     | Reloads the Libero SoC PolarFire 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 PolarFire 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 should be held active until the master is serviced the interrupt request. |
• Transceiver interface clocks (to CCC_SE and CCC_NE)
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 reference clock.
• When REF_CLK_SEL = 0, the PLL selects REF_CLK_0 as 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 ranges from 1 to 63. The valid operating range of PFD input frequency ($F_{PFD}$) is 1 MHz to 312 MHz in integer mode, and 10 MHz to 250 MHz in fractional mode.

Figure 22 • PLL Block Diagram—Clock Inputs and Outputs

5.3.1.2 Feedback Clock Input
The feedback clock input (FB_CLK) is available only when a PLL is configured in external feedback mode. One of the PLL clock outputs must be connected to FB_CLK.

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 ranges from 8 to 10, 12 to 4095 in integer mode, and 20 to 500 in fractional mode. The fractional part (FRAC) of the PLL feedback divide value is controlled by the delta-sigma modulator with 24-bit resolution.

5.3.1.3 Clock Outputs
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 slow clock on OUT3 clock output.

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 VDDPPLL 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 **UG0725: PolarFire FPGA 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.
5.3.1.4 Lock Output

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.

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 has been achieved, before asserting the lock signal. The lock delay must be set between 2 and 32768 cycles in the power of 2. The default setting in PLL register map is 2^8. Note that if this value is too low, the lock signal toggles.

5.3.1.5 Power-Down Input

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.

5.3.1.6 Clock Start/Stop Input

PolarFire 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 glitchlessly 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>
<thead>
<tr>
<th>PLL_POWERDOWN_N</th>
<th>OUT#1_EN</th>
<th>OUT#1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Normal clock</td>
</tr>
</tbody>
</table>

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

5.3.1.7 Bypass Input

Each PLL output has its own bypass MUXes—BYPPRE and BYPPOST—that select REF_CLK or FB_CLK (depending on BYPCKSEL MUX) to be passed to the post-divider input or directly to the output, respectively.

Figure 23 • PLL—Bypass Controls
The bypass MUXes must be configured through the CCC configurator. Each PLL has a bypass enable signal (BYPASS_EN_N) to cause all previously set up bypass MUXes settings of the PLL to occur simultaneously. The BYPASS_EN_N signal enables the bypass MUXes.

5.3.1.8 Reference Divider Synchronous Enable

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 current version of Libero SoC PolarFire.

5.3.1.9 Bandwidth Adjustment

PLL loop bandwidth is the measure of the PLLs ability to track the reference clock and its jitter. PLL filters the jitter present above the loop bandwidth. PolarFire 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. For PLL jitter performance and bandwidth values, see DS0141: PolarFire FPGA Datasheet.

Table 5 • PLL Bandwidth Parameter Settings

<table>
<thead>
<tr>
<th>Bandwidth Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>PLL with a low bandwidth; has better jitter rejection, but a slower lock time.</td>
</tr>
<tr>
<td>Medium-Low</td>
<td>PLL with a bandwidth between low to medium; has a balance between lock time and jitter rejection. This is the recommended setting.</td>
</tr>
<tr>
<td>Medium-High</td>
<td>PLL with a bandwidth between medium to high.</td>
</tr>
<tr>
<td>High</td>
<td>PLL with a high bandwidth; has a faster lock time, but tracks more jitter.</td>
</tr>
</tbody>
</table>

5.3.1.10 Delay Line Controls

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 should be configured in external feedback mode to add the delay line in feedback path.

Figure 24 • 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. See DS0141: PolarFire FPGA Datasheet for characterized value. 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, delay line increments or decrements its delay taps based on DELAY_LINE_DIRECTION and DELAY_LINE_LOAD. See PLL Port Description, page 23 for more information on delay line control signals.

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

### 5.3.1.11 Static Phase Shifting

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. Both of these 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 25* • Example of Using VCO/4 Delay and VCO/4 Phase Select to Fine Tune PLL Output Phase

The 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 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.

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.
5.3.1.12 Dynamic Phase Shifting

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

Table 6  Dynamic Phase Shift Ports

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHASE_OUT0_SEL</td>
<td>Selects the OUT0 for dynamic phase adjustment.</td>
</tr>
<tr>
<td>PHASE_OUT1_SEL</td>
<td>Selects the OUT1 for dynamic phase adjustment.</td>
</tr>
<tr>
<td>PHASE_OUT2_SEL</td>
<td>Selects the OUT2 for dynamic phase adjustment.</td>
</tr>
<tr>
<td>PHASE_OUT3_SEL</td>
<td>Selects the OUT3 for dynamic phase adjustment.</td>
</tr>
</tbody>
</table>
| 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 VCO phase. 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 PolarFire 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 have been 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 VCO phase.

Note: For any output that is divided, requests for many rotations in the same direction will rotate the output through VCO/8 phase delays through all of phases in the entire divided frequency. For example, if the output clock is divided by 1 (no output division) then continued forward rotations of the phase will provide a selection of up to 8 VCO phases, but if the output clock is divided by 2, continued forward rotations trace all 16 possible VCO phases in two clock periods.

It is required that any outputs that have their phase modified through either method must use the clock stop capability to glitchlessly stop the clock.

5.3.2 PLL Operational Modes

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

5.3.2.1 Internal Post-VCO Feedback Mode

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 26 • Internal Post-VCO Feedback Mode

5.3.2.2 Internal Post-Divider Feedback Mode

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 reset-on-lock feature (as explained in the internal post-VCO feedback mode) is not supported in the internal post-divider feedback mode. Hence, the PLL outputs may not be phase aligned. In this mode, the PLL does not compensate for global clock network delay.

Figure 27 • Internal Post-Divider Feedback Mode
5.3.2.3 **External Feedback Mode**

In this mode, the feedback clock port (FB_CLK) is exposed to the user. One of the four PLL clock outputs 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 designers 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 \((\text{REF_CLK} \times 4 \times \text{OUTDIVx} \times \text{FBDIV})/\text{RFDIV}\). 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 reset-on-lock feature requires user logic to be implemented in the fabric to align the PLL outputs.

*Figure 28 • External Feedback Mode—Feedback through Global Clock Network*

5.3.3 **Spread Spectrum Clock Generation**

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 spread spectrum modulator 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 spread spectrum modulator then modulates the integer and fractional bits so that the PLL is configured with a:

- nominal divide value of 50
- maximum divide value of 50.75 (FBDIV = 12'b000000110010, FRAC = 24'b11000000000000000000000000000000)
- minimum divide value of 49.25 (FBDIV = 12'b000000110001, FRAC = 24'b01000000000000000000000000000000)
The following figure shows the frequency versus time and the resulting amplitude in the frequency domain.

*Figure 29*: 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:

\[
\text{Modulation Depth} = \pm (\text{SPREAD}) \times 0.1\%
\]

*Table 7*: Center and Down Spread Modulation Depths Based on SPREAD Value

<table>
<thead>
<tr>
<th>SPREAD</th>
<th>Center Spread</th>
<th>Down Spread</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>±0.1%</td>
<td>-0.1%</td>
</tr>
<tr>
<td>2</td>
<td>±0.2%</td>
<td>-0.2%</td>
</tr>
<tr>
<td>3</td>
<td>±0.3%</td>
<td>-0.3%</td>
</tr>
<tr>
<td>4</td>
<td>±0.4%</td>
<td>-0.4%</td>
</tr>
<tr>
<td>5</td>
<td>±0.5%</td>
<td>-0.5%</td>
</tr>
<tr>
<td>6</td>
<td>±0.6%</td>
<td>-0.6%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>29</td>
<td>±2.9%</td>
<td>-2.9%</td>
</tr>
<tr>
<td>30</td>
<td>±3.0%</td>
<td>-3.0%</td>
</tr>
<tr>
<td>31</td>
<td>±3.1%</td>
<td>-3.1%</td>
</tr>
</tbody>
</table>

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 EMI amplitude. The larger the spread value, the more timing margin needed for the correct logic operation.
5.3.4 PLL Use Models

5.3.4.1 Clock Frequency Synthesis

PolarFire PLLs can be used to multiply or divide the reference clock with dividers—RFDIV, FBDIV, and OUTDIVx. Each of the four PLL outputs offer independent dividers (OUTDIVx) with values between 1 to 127. For OUT3, OUTDIV2, and OUTDIV3 can be cascaded to generate a clock that is up to $127 \times 127$ slower than the VCO clock.

Figure 30 • Frequency Synthesis

PolarFire 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.

5.3.4.2 Zero-Delay Buffer

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 PolarFire PLL can be used to create a zero-delay clock buffer. The PLL is configured in external feedback mode and 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 31 • 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 with 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 internal register can lead or lag the external clock output.

### 5.4 Delay-Locked Loops

PolarFire DLLs can be used in a variety 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]).

The key blocks of a PolarFire DLL are PFD, arithmetic logic unit (ALU), and a delay chain including five delay cells with 128 delay taps each. Each delay tap is design for a delay of ~25 ps steps. See DS0141: PolarFire FPGA Datasheet for characterized values. 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)

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 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 PolarFire DLLs corrects the duty-cycle of the reference clock to create 50% duty-cycle clock output. Clocks with 50% duty-cycle are important for implementing high-speed communication interfaces (for example, DDR applications).
5.4.1 DLL Operational Modes

The DLL can be operated in one of the following:

- Phase reference mode
- Phase generation mode
- Clock injection delay removal mode

5.4.1.1 Phase Reference Mode

In this mode, the DLL_REF_CLK feeds the PFD through two paths—one directly, and one 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 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 1 MHz to 800 MHz.

Figure 33 • DLL Ports—Phase Reference Mode

Table 8 • DLL Port List—Phase Reference Mode

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLL_REF_CLK</td>
<td>Input</td>
<td>Reference clock</td>
</tr>
<tr>
<td>DLL_CODE_UPDATE</td>
<td>Input</td>
<td>Delay code update signal</td>
</tr>
<tr>
<td>DLL_CODE_MOVE</td>
<td>Input</td>
<td>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</td>
</tr>
<tr>
<td>DLL_DIR</td>
<td>Input</td>
<td>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</td>
</tr>
<tr>
<td>DLL_PHASE_LOAD</td>
<td>Input</td>
<td>Reset the delay settings to the Libero SoC PolarFire programmed values. It must be set to 0 for dynamic code tuning.</td>
</tr>
<tr>
<td>DLL_POWERDOWN_N</td>
<td>Input</td>
<td>DLL power-down input (active low): 1'b0—Power-down state 1'b1—DLL is enabled</td>
</tr>
<tr>
<td>DLL_CODE[7:0]</td>
<td>Output</td>
<td>Binary delay code output</td>
</tr>
<tr>
<td>DLL_LOCK</td>
<td>Output</td>
<td>Lock output</td>
</tr>
<tr>
<td>DLL_DELAY_DIFF</td>
<td>Output</td>
<td>Delay code difference indicator</td>
</tr>
<tr>
<td>DLL_CODE_MOVE_DONE</td>
<td>Output</td>
<td>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</td>
</tr>
</tbody>
</table>
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 PolarFire 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 34 • Phase Reference Mode—Dynamic Configuration

5.4.1.2 Phase Generation Mode

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 1 MHz to 800 MHz.

### Figure 35 • DLL Ports—Phase Generation Mode

### Table 9 • DLL Port List—Phase Generation Mode

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLL_REF_CLK</td>
<td>Input</td>
<td>Reference clock</td>
</tr>
<tr>
<td>DLL_CLK_MOVE</td>
<td>Input</td>
<td>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</td>
</tr>
</tbody>
</table>
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 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 PolarFire 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.

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 PolarFire programmed settings.

**Table 9 • DLL Port List—Phase Generation Mode (continued)**

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLL_DIR</td>
<td>Input</td>
<td>Adds or subtracts a delay tap on DLL_CLK_1 when DLL_CLK_MOVE goes high.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b0—Subtracts delay by one tap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b1—Adds delay by one tap</td>
</tr>
<tr>
<td>DLL_PHASE_LOAD</td>
<td>Input</td>
<td>Resets the delay settings to the Libero SoC PolarFire programmed values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It must be set to 0 for dynamic clock tuning.</td>
</tr>
<tr>
<td>DLL_POWERDOWN_N</td>
<td>Input</td>
<td>DLL power-down input (active low):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b0—Power-down state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1'b1—DLL is enabled</td>
</tr>
<tr>
<td>DLL_CLK_0</td>
<td>Output</td>
<td>Primary clock output</td>
</tr>
<tr>
<td>DLL_CLK_1</td>
<td>Output</td>
<td>Secondary clock output</td>
</tr>
<tr>
<td>DLL_LOCK</td>
<td>Output</td>
<td>Lock output</td>
</tr>
<tr>
<td>DLL_CLK_MOVE_DONE</td>
<td>Output</td>
<td>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</td>
</tr>
</tbody>
</table>

**Figure 36 • 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 PolarFire programmed settings.
5.4.1.3 Clock Injection Delay Removal Mode

In Clock Injection Delay Removal mode, the PolarFire 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 37** • DLL Ports—Clock Injection Delay Removal Mode

![Diagram of DLL Ports](image)

**Table 10** • DLL Port List—Clock Injection Delay Removal Mode

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

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 feedback clock, thereby matching delays for the global clock network. When the reference and feedback clock are phase locked, the aggregate delay of the one or more delay cells and the clock distribution network correspond to an integer multiple of the clock cycle of reference clock. This mode supports distribution of DLL_CLK_0 and DLL_CLK_1 to the global and high-speed I/O clock networks. Note that only one can be fed back to DLL.

**Figure 38** • Clock Injection Delay Removal Mode

![Diagram of Clock Injection Delay](image)
5.5 PLL/DLL Cascading

The clock routing MUXes 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 MUXes based on the user inputs provided in the CCC configurator.

**Figure 39 • CCC Block Diagram**

![CCC Block Diagram](image.png)

5.5.1 PLL-to-PLL Cascading

CCC supports 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.

5.5.2 PLL Driving DLL

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 internal feedback mode.
5.6 CCC Configuration

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 Configurator window.

Figure 40 • CCC Configurator—Configuration Options

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

Figure 41 • 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 **Minimize Jitter** to use the highest VCO and FPFD frequencies.
  - Select **Minimize VCO*** to use the lowest VCO and highest FPFD frequencies. It is a balance between minimum jitter and minimum power consumption. In this mode, the lowest first enabled clock output frequency achievable is 200 MHz as this sets corresponding output divider’s division value to one.
  - Select **Minimize Power** to use minimum VCO and FPFD frequencies.
- **Feedback Mode**: select PLL feedback mode (Internal, External, or Post-VCO), and feedback clock for the external feedback mode.
  - **Internal**: 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. One of the output clocks must be selected as the feedback clock.
  - **Post-VCO**: Selects the VCO output as the feedback clock.
- **Features**
  - Select **Integer Mode** to use the PLL in integer mode, otherwise the PLL operates in Fractional Mode.
  - Select SSCG modulation to enable the spread spectrum clock generation. Click the **SSCG Modulation** tab for more information.
  - Select **Enable Dynamic Reconfiguration Interface** to expose the bus interface to the fabric.
  - Select **Export PowerDown Port** to expose the port to fabric.
The following figure shows the **Output Clocks** tab of PLL-Single configuration in the **Configurator** window.

*Figure 42 • 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 one below (actual lower) the requested frequency.
- **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 **Enable Bypass** to configure the bypass multiplexers present at input and output of the output dividers. Each clock output has its own bypass multiplexers to bypass PLL and its output divider. Each PLL has a BYPASS_EN_N control signal to dynamically enable the bypass multiplexers:
  - REF_PREDIV—Bypasses the PLL and selects the reference clock as an input to the output divider.
  - FB_PREDIV—Bypasses the PLL and selects the feedback clock as an input to the output divider.
  - REF_POSTDIV—Bypasses the PLL and output divider—selects the reference clock as the output clock.
  - FB_POSTDIV—Bypasses the PLL and output divider—selects the feedback clock as the output clock.

- Select PLL output clock connectivity
  - **Fabric Clock**: to connect the PLL output clock to the global clock network.
  - **Fabric 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.
  - **High-speed 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, NOMUXs, or preferred clock output pins through dedicated hardwired routing.

**Note:** 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 through fabric.

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

**Figure 43** • 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 44 • DLL Configuration—Phase Reference Mode**

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

- **Reference Clock**: Enter the frequency—ranges from 1 MHz–800 MHz.
- **Outputs and Options**:
  - **Jitter Range**: Select between ±200 ps–400 ps, ±350 ps–700ps, ±500 ps–1000ps, and ±750 ps–1500 ps.
  - **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 45 • DLL Configuration—Phase Generation Mode*

<table>
<thead>
<tr>
<th>Reference Clock Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frequency</strong>: Enter the reference clock frequency—ranges from 1 MHz–800 MHz.</td>
</tr>
<tr>
<td><strong>Division</strong>: Select reference clock division value between 1, 2, or 4 as per design requirements.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Secondary Output Clock Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase Shift</strong>: If the secondary output clock requires phase shifting in multiples of 11.25°, then select the phase shift from the drop-down list.</td>
</tr>
<tr>
<td><strong>50% Duty Cycle and Clock Division</strong>: 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.</td>
</tr>
<tr>
<td><strong>Select secondary output clock connectivity</strong>:</td>
</tr>
<tr>
<td>• <strong>Fabric Clock</strong>: to connect the primary output clock to the global clock network.</td>
</tr>
<tr>
<td>• <strong>HS I/O Clock</strong>: to connect the primary output clock to a high-speed I/O clock network.</td>
</tr>
<tr>
<td>• <strong>Dedicated Clock</strong>: to connect the primary output clock to other DLL or PLL in the same CCC, clock dividers or NGMUXs through dedicated hardwired routing.</td>
</tr>
</tbody>
</table>
Outputs and Options

- **Jitter Range**: Select between ±200 ps–400 ps, ±350 ps–700ps, ±500 ps–1000ps, and ±750 ps–1500 ps 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**.

![DLL Configuration—Injection Removal Mode](image)

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

- **Reference Clock** frequency: Enter the reference clock frequency—ranges from 1 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 ±200 ps–400 ps, ±350 ps–700 ps, ±500 ps–1000 ps, and ±750 ps–1500 ps.
- 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 need to be configured as explained in the preceding configurator.

![PLL-DLL Cascading](image)

**Figure 47 • PLL-DLL Cascading**

### 5.7 CCC Simulation Support

Microsemi Libero SoC PolarFire 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 PolarFire.
5.8 PLL/DLL Placement

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 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 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`

5.9 Dynamic Configuration of CCC

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. See CCC Configuration Registers, page 51 for more information on CCC configuration registers and their bit definitions.
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 48 • Clock Conditioning Circuitry Window*

2. Instantiate a PolarFire Dynamic Reconfiguration Interface macro into the SmartDesign. The dynamic reconfiguration interface macro converts the APB interface signals to CCC dynamic reconfiguration interface signals.
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 49 • 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 50** • CCC Dynamic Configuration System

![CCC Dynamic Configuration System Diagram](image)

Now, the APB master can dynamically configure the CCC configuration registers.

### 5.9.1 CCC Configuration Registers

The following tables show the CCC configuration registers.

**Table 11** • PLL Register Map

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFT_RESET</td>
<td>0x00</td>
<td>PLL reset register, see Table 13, page 52.</td>
</tr>
<tr>
<td>PLL_CTRL</td>
<td>0x04</td>
<td>PLL control register 1, see Table 14, page 52.</td>
</tr>
<tr>
<td>PLL_REF_FB</td>
<td>0x08</td>
<td>PLL reference and feedback register, see Table 15, page 53.</td>
</tr>
<tr>
<td>PLL_FRACN</td>
<td>0x0C</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PLL_DIV_0_1</td>
<td>0x10</td>
<td>PLL output 0/1 division register, see Table 16, page 53.</td>
</tr>
<tr>
<td>PLL_DIV_2_3</td>
<td>0x14</td>
<td>PLL output 2/3 division register, see Table 17, page 54.</td>
</tr>
<tr>
<td>PLL_CTRL2</td>
<td>0x18</td>
<td>PLL control register 2, see Table 18, page 54.</td>
</tr>
<tr>
<td>RESERVED</td>
<td>0x1C</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PLL_PHADJ</td>
<td>0x20</td>
<td>PLL phase register, see Table 19, page 54.</td>
</tr>
<tr>
<td>SSCG_REG_0</td>
<td>0x24</td>
<td>Reserved.</td>
</tr>
<tr>
<td>SSCG_REG_1</td>
<td>0x28</td>
<td>Reserved.</td>
</tr>
<tr>
<td>SSCG_REG_2</td>
<td>0x2C</td>
<td>Reserved.</td>
</tr>
<tr>
<td>SSCG_REG_3</td>
<td>0x30</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

**Note:** Do not write to reserved registers and bit fields.
### Table 12 • DLL Register Map

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLL_CTRL0</td>
<td>0x00</td>
<td>DLL control register 0, see Table 20, page 55.</td>
</tr>
<tr>
<td>DLL_CTRL1</td>
<td>0x04</td>
<td>DLL control register 1, see Table 21, page 57.</td>
</tr>
<tr>
<td>DLL_STAT0</td>
<td>0x08</td>
<td>DLL status register 0, see Table 22, page 57.</td>
</tr>
<tr>
<td>DLL_STAT1</td>
<td>0x0C</td>
<td>Reserved.</td>
</tr>
<tr>
<td>DLL_STAT2</td>
<td>0x0F</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

### 5.9.2 PLL Configuration Registers Bit Definitions

The following tables show the bit definitions of PLL configuration registers.

### Table 13 • SOFT_RESET—0x00

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7:0]</td>
<td>RESERVED</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>8</td>
<td>PERIPH</td>
<td>RW</td>
<td>Asserts the functional reset of the CCC.</td>
</tr>
<tr>
<td>[31:16]</td>
<td>RESERVED</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

### Table 14 • PLL_CTRL—0x04

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>REG_POWERDOWN_B</td>
<td>RW</td>
<td>Powers-down the PLL for the lowest quiescent current and the PLL outputs are low. This is an active low bit.</td>
</tr>
</tbody>
</table>
| 1          | REG_RFDIV_EN       | RW | Reference divider enable.  
0—Reference divider is reset, reference clock is divided by 1.  
1—Reference divider is active.  
The reference divide value is loaded on the rising edge of reference clock. |
| 2          | REG_DIVQ0_EN       | RW | PLL output divider 0 enable bit. |
| 3          | REG_DIVQ1_EN       | RW | PLL output divider 1 enable bit. |
| 4          | REG_DIVQ2_EN       | RW | PLL output divider 2 enable bit. |
| 5          | REG_DIVQ3_EN       | RW | PLL output divider 3 enable bit. |
| 6          | REG_RFCLK_SEL      | RW | Reference clock select  
0—Selects REF_CLK_0 as reference clock for PLL.  
1—Selects REF_CLK_1 as reference clock for PLL. |
| 7          | RESETONLOCK        | RW | Control signal to force a post divide reset on the rising edge of LOCK:  
0—Outputs are enabled based only on POSTDIVEN.  
1—Outputs are held low if LOCK is low and they get reset on the rising edge of LOCK. |
| [11:8]     | BYPCK_SEL          | RW | Select reference clock or feedback clock for bypass mode. One-hot MUX selection for each post divider:  
0—Select feedback clock for bypass mode.  
1—Select reference clock for bypass mode. |
Table 14 • PLL_CTRL—0x04 (continued)

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>REG_BYPASS_GO_B</td>
<td>RW</td>
<td>Bypass MUXes enable (active low)—Enables bypass multiplexers present at output dividers. The bypass multiplexers configuration is set through CCC configurator or dynamically through register configuration.</td>
</tr>
<tr>
<td>[19:16]</td>
<td>REG_BYPASSPRE</td>
<td>RW</td>
<td>Bypass MUX control for input to post divider. One-hot MUX selection for each post divider: 0—Select VCO output as input to post divider 1—Select REF_CLK or FB_CLK (depending on BYPCKSEL) as input to post divider.</td>
</tr>
<tr>
<td>[23:20]</td>
<td>REG_BYPASSPOST</td>
<td>RW</td>
<td>Bypass MUX control for post divider output. One-hot MUX selection for each clock output: 0—Select post divider output to drive PLL clock output. 1—Select RF_CLK or FB_CLK (depending on BYPCKSEL) to drive PLL clock output.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
<td>Reserved.</td>
</tr>
<tr>
<td>25</td>
<td>LOCK</td>
<td></td>
<td>Status of PLL lock.</td>
</tr>
<tr>
<td>26</td>
<td>LOCK_INT_EN</td>
<td>RW</td>
<td>Enable PLL lock interrupt.</td>
</tr>
<tr>
<td>27</td>
<td>UNLOCK_INT_EN</td>
<td>RW</td>
<td>Enable PLL unlock interrupt.</td>
</tr>
<tr>
<td>28</td>
<td>LOCK_INT</td>
<td></td>
<td>PLL lock interrupt signal.</td>
</tr>
<tr>
<td>29</td>
<td>UNLOCK_INT</td>
<td></td>
<td>PLL unlock interrupt signal</td>
</tr>
<tr>
<td>30</td>
<td>RESERVED</td>
<td></td>
<td>Reserved.</td>
</tr>
<tr>
<td>31</td>
<td>LOCK_N</td>
<td></td>
<td>Status of PLL lock. Active low signal.</td>
</tr>
</tbody>
</table>

Table 15 • PLL_REF_FB—0x08

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FSE_B</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FOUTFB_SELMUX_EN</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>[13:8]</td>
<td>RFDIV</td>
<td>RW</td>
<td>Reference divider value (1 to 63). Frequency into PFD is RF_CLK/REFDIV.</td>
</tr>
<tr>
<td>[27:16]</td>
<td>RESERVED</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>[31:28]</td>
<td>RESERVED</td>
<td>Reserved.</td>
<td></td>
</tr>
</tbody>
</table>

Table 16 • PLL_DIV_0_1—0x10

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2:0]</td>
<td>VCOPHSEL0</td>
<td>RO</td>
<td>Selects one of eight VCO phases for post divider 0 input.</td>
</tr>
<tr>
<td>[5:3]</td>
<td>DIV0_START</td>
<td>RW</td>
<td>VCO/4 reset delay counter for post divider 0.</td>
</tr>
<tr>
<td>[14:8]</td>
<td>POST0DIV</td>
<td>RW</td>
<td>Post divide value for post divider 0 (1 to 127).</td>
</tr>
</tbody>
</table>
Clock Conditioning Circuitry

Table 16 • PLL_DIV_0—0x10

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>RESERVED</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>[18:16]</td>
<td>VCOPHSEL1</td>
<td>RO</td>
<td>Selects one of eight VCO phases for post divider 1 input.</td>
</tr>
<tr>
<td>[30:24]</td>
<td>POST1DIV</td>
<td>RW</td>
<td>Post divide value for post divider 1 (1 to 127).</td>
</tr>
<tr>
<td>31</td>
<td>RESERVED</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

Table 17 • PLL_DIV_2—0x14

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2:0]</td>
<td>VCOPHSEL2</td>
<td>RO</td>
<td>Selects one of eight VCO phases for post divider 2 input.</td>
</tr>
<tr>
<td>[14:8]</td>
<td>POST2DIV</td>
<td>RW</td>
<td>Post divide value for post divider 2 (1 to 127).</td>
</tr>
<tr>
<td>15</td>
<td>RESERVED</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>[18:16]</td>
<td>VCOPHSEL3</td>
<td>RO</td>
<td>Selects one of eight VCO phases for post divider 3 input.</td>
</tr>
<tr>
<td>31</td>
<td>CKPOST3_SEL</td>
<td>RW</td>
<td>Selects input clock for post divide 3:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0—Selects 8-phase output or RF_CLK, depending on BYPASSPRE setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1—Selects FOUTDIV[2] as input to post divide 3 (cascaded dividers).</td>
</tr>
</tbody>
</table>

Table 18 • PLL_CTRL2—0x18

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1:0]</td>
<td>BWI</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>4</td>
<td>IREF_EN</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>5</td>
<td>IREF_TOGGLE</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>[12:9]</td>
<td>LOCKCNT</td>
<td>RW</td>
<td>Lock count selects the number of PFD edges after the last cycle slip before the lock signal goes high. Count is defined as 2^LOCKCOUNT (for example, if LOCKCOUNTSEL = 4'd10, LOCK goes high to 1024 PFD periods after the last cycle slip).</td>
</tr>
</tbody>
</table>

Table 19 • PLL_PHADJ—0x20

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PLL_REG_SYNCREFDIV_EN_</td>
<td>RW</td>
<td>Synchronous enable signal of both reference dividers in a CCC.</td>
</tr>
</tbody>
</table>
### 5.9.3 DLL Configuration Registers Bit Definitions

The following tables show the bit definitions of DLL configuration registers.

#### Table 19 • PLL_PHADJ—0x20 (continued)

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PLL_REG_ENABLE_SYNCREFDIV</td>
<td>RW</td>
<td>Asserting PLL_REG_SYNCREFDIV_EN enables synchronization of PLL reference divider enable with reference clock rising edge.</td>
</tr>
<tr>
<td>[4:2]</td>
<td>REG_OUT0_PHSINIT</td>
<td>RW</td>
<td>Initial VCO phase shift value for PLL output 0.</td>
</tr>
<tr>
<td>14</td>
<td>REG_LOADPHS_B</td>
<td>RW</td>
<td>Pulse REG_LOADPHS_B high then low initializes the VCO phase shift to initial values.</td>
</tr>
</tbody>
</table>

#### Table 20 • DLL Control Register 0—0x00

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1:0]</td>
<td>REG_PHASE_P</td>
<td>RW</td>
<td>Primary clock output phase selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b00 ≥ 0 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b01 ≥ pi/2 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b10 ≥ pi phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b11 ≥ 2*pi phase shift.</td>
</tr>
<tr>
<td>[3:2]</td>
<td>REG_PHASE_S</td>
<td>RW</td>
<td>Secondary clock output phase selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b00 ≥ pi/2 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 01 ≥ pi phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b10 ≥ 3*pi/2 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 11 ≥ 2*pi phase shift.</td>
</tr>
<tr>
<td>[5:4]</td>
<td>REG_SEL_P</td>
<td>RW</td>
<td>Primary clock output selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 00 ≥ normal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 01 ≥ normal through dummy delay to match divider on CLKO_S.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 10 ≥ duty cycle 50 when REF_SEL = 0 and FB_SEL = 1 for clock phase reference mode (REG_DIV_SEL = 0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 11 ≥ duty cycle 50 inversion.</td>
</tr>
<tr>
<td>[7:6]</td>
<td>REG_SEL_S</td>
<td>RW</td>
<td>Secondary clock output selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 00 ≥ normal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 01 ≥ divide by 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 10 ≥ divide by 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 11 ≥ duty cycle is 50 when REF_SEL = 0 and FB_SEL = 1 for clock phase reference mode (REG_DIV_SEL = 0).</td>
</tr>
<tr>
<td>8</td>
<td>REG_REF_SEL</td>
<td>RW</td>
<td>reference clock selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b0 ≥ divide by 1 or divide by 2 with 0 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b1 ≥ divide by 4 with 0 phase shift.</td>
</tr>
<tr>
<td>9</td>
<td>REG_FB_SEL</td>
<td>RW</td>
<td>feedback clock selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b 0 ≥ direct feedback or through dummy delay to match divider on reference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b 1 ≥ 2*pi shift for clock phase reference mode or feedback through dummy delay inversion.</td>
</tr>
<tr>
<td>Bit Number</td>
<td>Field Name</td>
<td>RW</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>----</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>REG_DIV_SEL</td>
<td>RW</td>
<td>Divided feedback clock select in clock injection removal mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b0 ≥ M4/M5 selects 0/1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1'b1 ≥ M4/M5 selects 2/3.</td>
</tr>
<tr>
<td>[13:11]</td>
<td>REG_FPHASE_CLK</td>
<td>RW</td>
<td>clk_o_1 fine phase selection in DEL4:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 000 ≥ pi/4 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 001 ≥ 5×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 010 ≥ 3×pi/8 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 011 ≥ 7×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 100 ≥ pi/2 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 101 ≥ 9×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 110 ≥ 5×pi/8 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 111 ≥ 11×pi/16 phase shift.</td>
</tr>
<tr>
<td>[15:14]</td>
<td>REG_ALU_UPD</td>
<td>RW</td>
<td>ALU update frequency:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 00 ≥ ALU update after 16 accumulated cycles of add or subtract.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 01 ≥ ALU update after 8 accumulated cycles of add or subtract.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 10 ≥ ALU update after 32 accumulated cycles of add or subtract.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 11 ≥ ALU update after 4 accumulated cycles of add or subtract.</td>
</tr>
<tr>
<td>[18:16]</td>
<td>REG_FPHASE_CODE</td>
<td>RW</td>
<td>Delay code fine phase selection:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 000 ≥ pi/4 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 001 ≥ 5×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 010 ≥ 3×pi/8 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 011 ≥ 7×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 100 ≥ pi/2 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 101 ≥ 9×pi/16 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 110 ≥ 5×pi/8 phase shift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b 111 ≥ 11×pi/16 phase shift.</td>
</tr>
<tr>
<td>19</td>
<td>REG_LOCK_FRC</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>[21:20]</td>
<td>REG_LOCK_FLT</td>
<td>RW</td>
<td>Phase lock tolerance options on glitch filter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 00 ≥ ±200 ps–400 ps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 01 ≥ ±350 ps–700 ps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 10 ≥ ±500 ps–1000 ps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2'b 11 ≥ ±750 ps–1500 ps.</td>
</tr>
</tbody>
</table>
### Table 21 • DLL Control Register 1—0x04

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7:0]</td>
<td>REG_SET_ALU</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>[14:8]</td>
<td>REG_ADJ_DEL4</td>
<td>RW</td>
<td>Adjust fine phase shift in DEL4, when MSB = 1, subtract REG_ADJ_DEL4[5:0] or add REG_ADJ_DEL4[5:0].</td>
</tr>
<tr>
<td>15</td>
<td>REG_TEST_S</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>23</td>
<td>REG_TEST_RING</td>
<td>RW</td>
<td>Reserved.</td>
</tr>
<tr>
<td>30</td>
<td>REG_RELOCK_FAST</td>
<td>RW</td>
<td>Keep locking without counting down to 0 in clock injection removal mode.</td>
</tr>
<tr>
<td>31</td>
<td>REG_POWERDOWN_EN</td>
<td>RW</td>
<td>Enable POWERDOWN_B as reset input.</td>
</tr>
</tbody>
</table>

### Table 22 • DLL Status Register 0—0x08

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Field Name</th>
<th>RW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>REG_RESET</td>
<td>RW</td>
<td>Active high reset of DLL</td>
</tr>
<tr>
<td>1</td>
<td>REG_ALU_HOLD</td>
<td>RW</td>
<td>Freeze ALU from updating</td>
</tr>
<tr>
<td>2</td>
<td>REG_CODE_UPD</td>
<td>RW</td>
<td>Delay code update when set</td>
</tr>
<tr>
<td>3</td>
<td>REG_PHASE_LOAD</td>
<td>RW</td>
<td>Fine phase reset in DEL4 or delay code to system register settings, when each time REG_PHASE_SHIFT_CLK and REG_PHASE_SHIFT_CODE go high</td>
</tr>
<tr>
<td>4</td>
<td>REG_PHASE_DIRECTION</td>
<td>RW</td>
<td>Fine phase shift going 1 = up or 0 = down one tap in DEL4 or delay code, when each time REG_PHASE_SHIFT_CLK or REG_PHASE_SHIFT_CODE go high</td>
</tr>
<tr>
<td>5</td>
<td>REG_DIFF_RANGE</td>
<td>RW</td>
<td>The range (up to 8) of difference for DELAY_DIFF assertion</td>
</tr>
<tr>
<td>8</td>
<td>REG_LOCK_INT_EN</td>
<td>RW</td>
<td>Enable lock interrupt</td>
</tr>
<tr>
<td>9</td>
<td>REG_UNLOCK_INT_EN</td>
<td>RW</td>
<td>Enable unlock interrupt</td>
</tr>
<tr>
<td>10</td>
<td>REG_LOCK_INT</td>
<td>RO</td>
<td>Lock interrupt</td>
</tr>
<tr>
<td>11</td>
<td>REG_UNLOCK_INT</td>
<td>RO</td>
<td>Unlock interrupt</td>
</tr>
<tr>
<td>12</td>
<td>REG_PHASE_MOVE_CLK</td>
<td>RW</td>
<td>Fine phase shifting on CLKO_S when going high, has to go low before the next shifting</td>
</tr>
<tr>
<td>13</td>
<td>REG_PHASE_MOVE_CODE</td>
<td>RW</td>
<td>Fine phase shifting on DELAY_CODE when going high, has to go low before the next shifting</td>
</tr>
<tr>
<td>14</td>
<td>RESERVED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[23:16]</td>
<td>SRO_DELAY_CODE</td>
<td>RO</td>
<td>Same as DELAY_CODE output</td>
</tr>
<tr>
<td>[31:24]</td>
<td>SRO_ADJ_CODE</td>
<td>RO</td>
<td>Dynamic delay code adjustment, when MSB = 1, subtract SRO_ADJ_CODE[6:0] or add SRO_ADJ_CODE[6:0]</td>
</tr>
</tbody>
</table>