# Contents

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

2 Power-Up ................................................................. 2  
  2.1 Power-On Reset ........................................................... 2  
  2.2 Device Boot ............................................................... 3  
  2.3 Design Initialization ..................................................... 4  
    2.3.1 Secured Non-Volatile Memory (sNVM) .................... 5  
    2.3.2 μPROM ............................................................... 5  
    2.3.3 External SPI Flash ............................................... 5  
    2.3.4 How To Use Design and Memory Initialization ........ 5  
  2.4 HSIO/GPIO Bank Initialization ........................................ 11  
  2.5 Transceiver Initialization ............................................... 12  
  2.6 User PLLs and DLLs Initialization .................................... 12  
  2.7 PCIe Initialization ........................................................ 12  
  2.8 Fabric RAM Initialization ............................................... 12  
  2.9 State of Blocks During Power-Up .................................... 16  

3 Dynamic Reconfiguration Interface ................................. 17  
  3.1 DRI Support in Libero SoC PolarFire ............................. 17  
    3.1.1 DRI Configuration for XCVR ................................. 18  
    3.1.2 DRI Configuration for TX PLL ............................... 18  
    3.1.3 DRI Configuration for PCIe .................................. 19  
    3.1.4 DRI Configuration for CCC ................................. 20  
    3.1.5 DRI Configuration For CRYPTO ........................... 21  

4 Resets .............................................................................. 22  
  4.1 PolarFire Hard Resets ................................................. 22  
    4.1.1 DEVRST_N ......................................................... 22  
    4.1.2 Resets Initiated from POR Circuitry ....................... 22  
  4.2 Initialization Signals to Fabric ....................................... 23  
  4.3 Functional Example .................................................... 26  
  4.4 CoreReset_PF IP Core ................................................ 26  

5 Device Access Time ....................................................... 27  

6 Appendix: Power Supplies ............................................... 28
Figures

Figure 1  Programmed Device Boot-Up ................................................. 3
Figure 2  Design Initialization of a Programmed Device .......................... 4
Figure 3  The Design Initialization and Memory Initialization Tab ............... 6
Figure 4  Design Initialization - First Stage ........................................ 6
Figure 5  Design Initialization - Second Stage ....................................... 7
Figure 6  Design Initialization - Third Stage ......................................... 8
Figure 7  Custom Configuration .......................................................... 8
Figure 8  Generate Design Initialization Data ........................................ 9
Figure 9  All Three Initialization Clients Generated and Added in sNVM ........ 10
Figure 10 First and Second Initialization Clients Generated in sNVM ............. 10
Figure 11 Third Initialization Client on µPROM ..................................... 10
Figure 12 Third Initialization Client on SPI Flash .................................. 11
Figure 13 Program SPI Flash Image ...................................................... 11
Figure 14 PF Two-Port Large SRAM Configurator Window ......................... 13
Figure 15 Import Memory Content ...................................................... 14
Figure 16 Fabric RAMs Present in the Design ......................................... 15
Figure 17 Import Memory Content Dialog Box ....................................... 15
Figure 18 High-level Block Diagram of DRI Implementation ....................... 17
Figure 19 The DRI IP with APB Interface .............................................. 18
Figure 20 The XCVR Tab ................................................................. 18
Figure 21 The TX PLL Tab ............................................................... 19
Figure 22 The PCIE Tab ................................................................. 19
Figure 23 The CCC Tab ................................................................. 20
Figure 24 Enabling DRI on CRYPTO .................................................. 21
Figure 25 Simplified Block Diagram of Resets ...................................... 22
Figure 26 PolarFire Initialization Monitor ............................................. 23
Figure 27 PolarFire Initialization Monitor Configurator ............................. 24
Figure 28 PolarFire Initialization Monitor configurator - Simulation options .... 25
Figure 29 Example of PolarFire Initialization ....................................... 26
Figure 30 The CoreRest_PF IP ........................................................ 26
## Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Threshold Voltage Levels</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>Default State During Device Power-Up</td>
<td>16</td>
</tr>
<tr>
<td>Table 3</td>
<td>Port List for DRI</td>
<td>21</td>
</tr>
<tr>
<td>Table 4</td>
<td>Power Supplies in PolarFire Devices</td>
<td>28</td>
</tr>
</tbody>
</table>
1 Revision History

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

1.1 Revision 4.0

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

- Updated the document for Libero SoC PolarFire version v2.1.
- Updated Device Boot, page 3 and Figure 1, page 3.
- Updated Design Initialization, page 4 and Figure 2, page 4.
- Updated How To Use Design and Memory Initialization, page 5.
- Updated HSIO/GPIO Bank Initialization, page 11.
- Updated Transceiver Initialization, page 12.
- Updated State of Blocks During Power-Up, page 16.
- Added CoreReset_PF IP Core, page 26.
- Updated Initialization Signals to Fabric, page 23.

1.2 Revision 3.0

Revision 3.0 was published in January 2018. The following is a summary of changes made in this revision.

- Updated the screen shots as per the Libero SoC PolarFire v2.0 release throughout the document.
- A note about SPI Slave Programming mode is deleted from the Design Initialization, page 4 and the section is edited to add the usage of PolarFire Initialization Monitor.
- Added µPROM and External SPI Flash sections (see µPROM, page 5 and External SPI Flash, page 5).
- Edited HSIO/GPIO Bank Initialization, page 11 to describe BANK_##_CALIB_STATUS and BANK_##_VDDI_STATUS signals.
- A note is added in Power-Up to Functional Time, page 22 to give a reference to DS0141: PolarFire FPGA Datasheet.
- Figure 27, page 24 is replaced and a description about PolarFire Initialization Monitor is added below the figure.
- Deleted the Top-Level Device Power-Up figure from the Power-Up, page 2.
- Deleted GPIO_ACTIVE and HSIO_ACTIVE pins and added BANK_##_CALIB_STATUS and BANK_##_VDDI_STATUS pins in Figure 25, page 22.
- Deleted GPIO_ACTIVE and HSIO_ACTIVE pin descriptions and added BANK_##_CALIB_STATUS and BANK_##_VDDI_STATUS pin description in the Initialization Signals to Fabric, page 23.
- Recommendation on device reset usage was added in DEVRST_N, page 22.
- Figure 2, page 4 was updated.

1.3 Revision 2.0

Revision 2.0 was published in June 2017. The following was a summary of the changes made in revision 2.0 of this document.

- The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v1.1 SP1 release.
- Information about the use case of PolarFire Initialization Monitor was added. For more information, see Functional Example, page 26.

1.4 Revision 1.0

Revision 1.0 was the first publication of this document.
PolarFire FPGAs use advanced power-up circuitry to ensure reliable power-up. When the device is powered on, the Power-on Reset (POR) circuitry and the system controller ensure a systematic POR. Within the PolarFire device, the system controller is a dedicated microcontroller, which is responsible for the power-up and reset of the FPGA fabric and I/Os.

The entire process of powering up the device includes the following sequential steps:

- Power-on reset
- Device boot
- Design initialization

### 2.1 Power-On Reset

When the device is power cycled, the POR circuitry detects voltage ramp-up on the $V_{DD18}$, $V_{DD25}$, and $V_{DD}$ power supply rails using voltage detectors. For a list of power supplies, see Appendix: Power Supplies, page 28. The system controller remains in the reset state until the required voltage threshold levels are achieved.

The voltage detectors in the PolarFire devices are calibrated with a high-level of accuracy to ensure reliable monitoring of minimum threshold levels. The device boot starts with a delay of 10 ms after the following voltage supply rails reach their respective threshold voltage levels.

<table>
<thead>
<tr>
<th>Supply Rail</th>
<th>Threshold Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>95% (irrespective of whether VDD is 1.0V or 1.05V)</td>
</tr>
<tr>
<td>VDD18</td>
<td>90%</td>
</tr>
<tr>
<td>VDD25</td>
<td>90%</td>
</tr>
</tbody>
</table>

In PolarFire devices, separate voltage detectors monitor I/O bank supplies. During POR, the serial transceivers and the fabric are powered down, and HSIO/GPIO banks are tri-stated.

For more information on power supply sequencing requirements and recommendations, see the "Core Power Supply Operations" section, in UG0726: PolarFire FPGA Board Design User Guide.
2.2 Device Boot

After POR circuitry releases the reset signal to the system controller, the device boot-up procedure is executed. The system controller always executes the same device boot-up sequence irrespective of the user design. The following illustration shows the boot-up sequence for a programmed device.

Figure 1 • Programmed Device Boot-Up

<table>
<thead>
<tr>
<th>System controller</th>
<th>Executing boot-up sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>sNVM</td>
<td>Powered down</td>
</tr>
<tr>
<td></td>
<td>Powered up</td>
</tr>
<tr>
<td></td>
<td>Enabled for normal operation</td>
</tr>
<tr>
<td>HSIO/GPIO banks</td>
<td>Disabled, Outputs Tri-stated</td>
</tr>
<tr>
<td></td>
<td>Configured with flash config bits*</td>
</tr>
<tr>
<td>Transceiver I/O</td>
<td>Tri-stated</td>
</tr>
<tr>
<td>FPGA fabric</td>
<td>Powered down</td>
</tr>
</tbody>
</table>

Note: This figure does not show the exact time values. The exact time values will be shown in DS0141: PolarFire FPGA Datasheet.

The following events occur during device boot-up:

- sNVM is reset and subsequently powered up.
- Transceiver I/Os are tri-stated.
- Fabric is powered down.
- HSIO and GPIO banks are configured using flash configuration bits.
2.3 Design Initialization

During design initialization, user design blocks such as LSRAM, μSRAM, transceiver configurations, and PCIe can be optionally initialized using data stored in the non-volatile storage memory. The initialization data can be stored in μPROM, sNVM, or an external SPI flash. The storage location of the initialization data is selected during the Libero design flow. The initialization data can be encrypted if necessary for inclusion in the programming bit-stream.

**Note:** Encryption and authentication of initialization data are currently not supported in Libero.

The following illustration shows design initialization of the LSRAM/μSRAM and PCIe/transceiver blocks.

*Figure 2 • Design Initialization of a Programmed Device*

*Note:* This figure does not show the exact time values. The exact time values will be shown in DS0141: PolarFire FPGA Datasheet.
Figure 2, page 4 shows the sequence in which the fabric, PCIe, Transceiver, LSRAMs, and µSRAMs are automatically initialized. The sequence is customized depending on the resources instantiated in the user design. For example, the PCIe INIT DONE will not assert if the user design does not contain PCIe. As a result, the sequence skips PCIe INIT DONE and moves to the next step.

The user can monitor design initialization using the PolarFire Initialization Monitor. PolarFire Initialization Monitor outputs are asserted to indicate the current status. For more information about initialization signals and about the PolarFire Initialization Monitor, see Initialization Signals to Fabric, page 23.

Note: PolarFire devices have built-in tamper detection features to monitor voltage supplies and flags to detect minimum or maximum threshold values. These flags are valid only after design initialization, and not during POR.

2.3.1 Secured Non-Volatile Memory (sNVM)

Each PolarFire FPGA has 56 K Bytes of sNVM, organized into 221 pages of 236 or 252 bytes depending on whether the data is stored as plain text or encrypted/authenticated data. It can be accessed through system services calls to the PolarFire system controller. Pages within the sNVM can be marked as ROM during bit-stream programming. The sNVM content can be used to initialize LSRAMs and µSRAMs with secure data.

2.3.2 µPROM

PolarFire devices have a single user-programmable read-only memory (µPROM) row located at the bottom of the fabric, providing up to 513 Kb of non-volatile, read-only memory. The address bus is 16-bit wide and the read data bus is 9-bit wide. Fabric logic has access to the entire µPROM data.

2.3.3 External SPI Flash

The SPI flash memory interfaces with the system controller’s SPI and stores the programming images. The system controller supports devices from vendors like Micron, Winbond, Atmel, and Spansion.

2.3.4 How To Use Design and Memory Initialization

This section describes how to initialize PCIe, transceivers, and fabric RAM blocks during design time using the Configure Design Initialization Data and Memories option. This initialization is performed in two phases—configuring .mem files (initialization clients) and generating initialization data.

Complete the following steps to configure and generate the initialization clients:

1. Double-click the Configure Design Initialization Data and Memories option from Design Flow, Program and Debug Design.
   The Design Initialization tab appears in the right pane, as shown in the following figure.
2. The first stage initialization client is stored in the on-chip sNVM at the top of the address space. The initialization client asserts the FABRIC_POR_N signal available on the PolarFire Initialization Monitor macro, which can be instantiated in the design. For more information about PolarFire Initialization Monitor, see Initialization Signals to Fabric, page 23.

3. The second stage initialization client is also stored in the on-chip sNVM. The location of that script can be configured by the user. The initialization client initializes the PCIe and XCVR blocks present in the design and asserts PCIE_INIT_DONE available in the PolarFire Initialization Monitor macro when the initialization is complete. See Figure 5, page 7.
In design initialization, user design blocks such as LSRAM, μSRAM, transceivers, and PCIe can be initialized as an option using data stored in the non-volatile storage memory. The initialization data can be stored in μPROM, sNVM, or an external SPI Flash.

Follow the below steps to program the initialization data:
1. Set up your fabric RAMs initialization data, if any, using the Fabric RAMS tab.
2. Define the storage location of the initialization data.
3. Generate the initialization clients.
4. Generate or export the bitstream.
5. Program the device.

This step stores the PCIe and XCVR blocks initialization client from the specified start address, when the initialization client is generated. If there are other initialization clients added before, start address of the second stage initialization client can be part of any other available memory space.

4. Third stage initialization script can be stored in the on-chip μPROM, sNVM or an external SPI Flash memory in non-authenticated mode. The location of that script is configurable by the user.

5. The Initialization client initializes the fabric RAMs present in the design. The script asserts the ‘PolarFire Initialization Monitor’ macro SRAM_INIT_DONE (for LSRAM) USRAM_INIT_DONE (for μSRAM) signals, when the initialization is complete.

The following figure shows sNVM as the third stage chosen memory, in a similar way, μPROM or SPI flash can be selected.
Figure 6 • Design Initialization - Third Stage

6. Use text file as an option to override register settings for custom configuring XCVR/PCIe blocks present in the design.

7. Click Apply in the top-left corner of the window after selecting the required options. See the following figure.

Figure 7 • Custom Configuration
8. Double-click **Generate Design Initialization Data** under **Design Flow** tab. It automatically generates the first, second, and third stage initialization clients, which are automatically added to the non-volatile memory that the user chooses. The **Generate Design Initialization Data** button is highlighted as in the following figure.

*Figure 8 • Generate Design Initialization Data*
The following figures show examples of initialization clients automatically added.

**Figure 9 • All Three Initialization Clients Generated and Added in sNVM**

**Figure 10 • First and Second Initialization Clients Generated in sNVM**

**Figure 11 • Third Initialization Client on µPROM**
These steps ensure that PCIe, XCVR, and Fabric RAMs present in the design are initialized during power-up using initialization clients placed in the non-volatile memory based on the user selection.

9. If an external SPI-Flash is chosen for stage 3, before completing the Run PROGRAM Action, you should generate Generate SPI Flash Image and Run PROGRAM_SPI_Image Action from the Design Flow tab, as shown in the following figure.

For more information about sNVM, see UG0753: PolarFire FPGA Security User Guide. For more information about µPROM, see UG0680: PolarFire FPGA Fabric User Guide. For more information about SPI flash, see UG0714: PolarFire FPGA Programming User Guide.

2.4 HSIO/GPIO Bank Initialization

The GPIO and HSIO banks can be left powered down or powered up. During the device power-up, power up the used GPIO and HSIO banks simultaneously along with all the other power supplies. All of the banks are initialized automatically with flash configuration bits when fabric is powered up.

The time at which I/Os are functional depends on a combination of the following:

- Device boot
- Ramp-up time of the power applied to the I/O banks
- Calibration time of the high-speed I/Os (For example, DDR interfaces)

For low-speed operations below 400 MHz, I/Os are functional after the power applied to the I/O banks exceeds the threshold levels. For high-speed operations like DDR interfaces, the process of I/O calibration is necessary.

The I/O calibration process occurs automatically. For high-speed applications, the I/Os are functional after the I/O calibration completes. The status of the I/O calibration and bank power supply can be monitored using the status signals of the PolarFire Initialization Monitor IP.

The PolarFire Initialization Monitor asserts BANK_#_CALIB_STATUS and BANK_#_VDDI_STATUS signals to the fabric. BANK_#_CALIB_STATUS can be used by the user logic to determine if the calibration completes for each I/O bank. BANK_#_VDDI_STATUS signal can be used to monitor VDDI supply on specific I/O banks.

The Dynamic reconfiguration interface (DRI) does not block access until the I/Os are calibrated. The user design must wait for the completion of the I/O calibration process to release the DRI from reset. For more information about DRI, see Dynamic Reconfiguration Interface, page 17.
2.5 Transceiver Initialization

Transceiver power-up depends on $V_{DDA}$, $V_{DDA25}$, and $V_{DD\_XCVR\_CLK}$. $V_{DD\_XCVR\_CLK}$ is applicable if an external reference clock is used for transceivers. For a list of power supplies, see Appendix: Power Supplies, page 28. Glitches can occur in the reference clocks and the data bits during power-up.

The transceiver can be initialized by either of the following:

- Flash configuration bits valid after fabric power-up.
- Design initialization data stored in non-volatile memory

When the DEVICE_INIT_DONE from INIT component goes high, the transceiver is completely configured.

If global clock routes in the fabric are fed using the transceiver reference clock as a primary clock source, any fabric logic using this clock must be held in asynchronous reset using DEVICE_INIT_DONE signal until transceiver I/Os are enabled.

2.6 User PLLs and DLLs Initialization

Both PLLs and DLLs are initialized automatically with flash configuration bits when fabric is powered up.

2.7 PCIe Initialization

To achieve the PCIe initialization requirement, the physical layer is configured using flash configuration bits. The remainder of the configuration is done during design initialization with the user data stored in the non-volatile memory.

For more information about PCIe initialization process, see UG0685: PolarFire FPGA PCI Express User Guide.

2.8 Fabric RAM Initialization

After power-up, the state of LSRAM and µSRAM blocks are unknown. If required, these blocks can be pre-initialized with known values in Libero SoC PolarFire by:

- Importing the “.hex” file using the LSRAM and µSRAM Configurator before Place and Route.
- Importing “.hex” file using the Fabric RAMs tab of the Configure Design Initialization Data and Memories option after Place and Route is performed.

Follow these steps to initialize Fabric RAM using LSRAM and µSRAM configurator:
1. In the PF Two-Port Large SRAM Configurator window, select the Memory Initialization Settings tab. Then, select the Initialize RAM at Power-up check box as shown in the following figure.

   **Figure 14** • PF Two-Port Large SRAM Configurator Window

2. Select the Import File option and import memory content file (Intel-Hex) from the Import Memory Content dialog box, as shown in the following figure. File extensions are set to *.hex for Intel-Hex files during import. The imported memory content is displayed in the RAM Content Editor pane.
Figure 15 • Import Memory Content

For more information about LSRAM and µSRAM Configurators and user options, see the “Embedded Memory Blocks” section in UG0680: PolarFire FPGA Fabric User Guide.

Note: Follow the previous steps for importing memory file to initialize µSRAMs during Power-up.
Follow these steps to initialize Fabric RAM using the **RAM Initialization** tab of the **Design and Memory Initialization** option from the **Design Flow** tab:

1. The **RAM Initialization** tab shows Fabric RAMs present in the design as shown in the following figure.

*Figure 16 • Fabric RAMs Present in the Design*

![Fabric RAMs Present in the Design](image1)

2. Double-click on the required fabric RAM client and select the **Memory File** option and click **Browse** to import the memory file (Intel-Hex) from the **Import Memory Content** dialog box as shown in the following figure. File extensions are set to “*.hex” for Intel-Hex files during import.

*Figure 17 • Import Memory Content Dialog Box*

![Import Memory Content Dialog Box](image2)
Note: After the required hex file is chosen, generate the initialization client using the Generate Design Initialization Data option in the Design Flow tab. The generated initialization client is stored in non-volatile memories like µPROM, sNVM or external SPI flash as part of design initialization phase.

2.9 **State of Blocks During Power-Up**

The following table shows the state of different blocks during device power-up.

**Table 2 • Default State During Device Power-Up**

<table>
<thead>
<tr>
<th>Block</th>
<th>POR</th>
<th>Programmed Device Boot</th>
<th>Design Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>System controller</td>
<td>Under reset</td>
<td>Executing boot-up sequence</td>
<td>Execution from non-volatile memory</td>
</tr>
<tr>
<td>sNVM</td>
<td>Under reset</td>
<td>Powered up</td>
<td>Powered up</td>
</tr>
<tr>
<td>FPGA fabric array</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Powered up</td>
</tr>
<tr>
<td>LSRAM</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Initialized with user data</td>
</tr>
<tr>
<td>µSRAM</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Initialized with user data</td>
</tr>
<tr>
<td>µPROM</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Powered up</td>
</tr>
<tr>
<td>Math block</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Powered up</td>
</tr>
<tr>
<td>Transceiver and TX PLLs</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Initialized with user data</td>
</tr>
<tr>
<td>GPIO/HSIO - Low Speed</td>
<td>Disabled and tri-stated</td>
<td>Powered up but not functional</td>
<td>Functional after the I/O supply exceeds the threshold voltage level</td>
</tr>
<tr>
<td>GPIO/HSIO - High-speed</td>
<td>Disabled and tri-stated</td>
<td>Powered up but not functional</td>
<td>Functional after the I/O supply exceeds the threshold voltage level and the I/O calibration process completes</td>
</tr>
<tr>
<td>PCIe</td>
<td>Powered down</td>
<td>Powered down</td>
<td>Initialized with user data</td>
</tr>
<tr>
<td>Transceiver I/O</td>
<td>Tri-stated</td>
<td>Tri-stated</td>
<td>Termination Enabled, operational</td>
</tr>
</tbody>
</table>

Note: For more information about cold boot and warm boot power-up to functional time, see the “Power-Up to Functional Timing” section in *DS0141: PolarFire FPGA Datasheet.*
3. Dynamic Reconfiguration Interface

After initialization, run-time configuration of transceiver PMA/PCS, PCIe, TX PLLs, and CCC PLLs/DLLs can be performed using the Dynamic Reconfiguration interface (DRI). DRI is an APB slave that allows read and write operations.

Each embedded peripheral has its own address range on the APB slave. For more information about register set and base address, see the specific peripheral user guide.

**Note:** The APB wires in the device that connect to the APB Master cannot be monitored or altered in the Libero SoC PolarFire design. They are used to facilitate HDL simulation of changes made to the peripherals over the DRI.

The following figure shows the high-level block diagram of the DRI implementation in PolarFire devices.

*Figure 18 • High-level Block Diagram of DRI Implementation*

**Note:** For more information about user connections required while using DRI interface, see the Dynamic Configuration of CCC section in UG0684: PolarFire FPGA Clocking Resources User Guide.

### 3.1 DRI Support in Libero SoC PolarFire

PolarFire Dynamic Reconfiguration Interface (DRI) IP is available in the IP Catalog under Peripherals as shown in Figure 19, page 18. The following sections describe DRI configuration for XCVR, TX PLL, PCIe, and CCC blocks using the PolarFire Dynamic Reconfiguration Interface Configurator window.
3.1.1 DRI Configuration for XCVR

In the XCVR tab, individual lanes (LANE_0 - 3) can be selected for each quad lane (Qn Lanes, n = 0 - 5) as shown in the following figure. For example, when Q0_LANE0 and Q0_LANE1 lane check boxes are selected in Q0 Lanes, new slave ports are added to the DRI IP block.

3.1.2 DRI Configuration for TX PLL

In the TX PLL tab, the required options per quad to enable DRI for both spread spectrum generation capable transmit PLLs (Q#_TXPLL_SSC) and (Q#_TXPLLn) without spread spectrum capabilities can be selected as shown in the following figure. 

DRI option is provided for two Q#_TXPLLn (Q#_TXPLL0, Q#_TXPLL1) within transceiver quad locations. For more information, see the “Transmit PLL” section in UG0677: PolarFire FPGA Transceiver User Guide.
3.1.3 DRI Configuration for PCIe

In the PCIe tab, PCIE0 and PCIE1, or any one of them can be selected to enable DRI on the selected options as shown in Figure 22, page 19.
3.1.4 DRI Configuration for CCC

In the CCC tab, the required PLLs and DLLs (in all four corners NW, NE, SE, SW) can be selected to enable DRI on the selected options as shown in the following figure.

*Figure 23* • The CCC Tab

![Diagram showing DRI configuration interface](image)
3.1.5 DRI Configuration For CRYPTO

In the Misc tab, the CRYPTO option can be selected to enable DRI on CRYPTO as shown in the following figure.

Note: The Enable latency simulation option is currently not supported.

Figure 24 • Enabling DRI on CRYPTO

The following table shows the port list for Dynamic reconfiguration interface

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRI_CLK</td>
<td>input</td>
<td>Internal subsystem peripheral clock</td>
</tr>
<tr>
<td>DRI_WDATA[32:0]</td>
<td>input</td>
<td>Write data input of all APB peripheral slaves</td>
</tr>
<tr>
<td>DRI_ARST_N</td>
<td>input</td>
<td>Active low DRI asynchronous reset input</td>
</tr>
<tr>
<td>DRI_CTRL[10:0]</td>
<td>Input</td>
<td>Dynamic reconfiguration interface control bits</td>
</tr>
<tr>
<td>DRI_RDATA[32:0]</td>
<td>output</td>
<td>Read data output to all APB peripheral slaves</td>
</tr>
<tr>
<td>DRI_INTERRUPT</td>
<td>output</td>
<td>DRI interrupt signal</td>
</tr>
</tbody>
</table>

Note: The ports in the previous table use embedded resources within the device. No FPGA resources are used and have no timing requirements within the user design.
After device power-up, the PolarFire system controller manages resets to all device components, including the fabric, embedded memory blocks, transceiver, PLLs/DLLs, and peripherals. The following sections overview hard resets, soft resets, and the power down sequence.

### 4.1 PolarFire Hard Resets

PolarFire devices can be reset by any of the following sources:

- DEVRST_N
- Reset from POR circuitry

*Figure 25* • Simplified Block Diagram of Resets

#### 4.1.1 DEVRST_N

DEVRST_N (device reset) is powered through VDDI3. This dedicated pin is asserted to initiate a full device reset and re-boot through an external I/O pad. This signal is propagated as a non-maskable interrupt to the system controller. The system controller powers down the fabric when DEVRST_N is asserted. The device power up sequence restarts after the de-assertion of DEVRST_N. If DEVRST_N is asserted during programming, the system controller aborts programming.

It is not recommended to use DEVRST_N as a design reset. Design resets must be implemented using an HSIO or a GPIO pin of the FPGA. If unused, DEVRST_N must be kept asserted until the board stabilizes, for example, until the reference clocks to PLLs are received. After the board stabilizes, DEVRST_N must be de-asserted. DEVRST_N can be used when the device needs to be recovered from SEU or other unforeseen circumstances.

The DEVRST_N assertion results in full re-initialization of the device, including the loading of user configuration data to PCIe, transceivers, and the re-initialization of fabric LSRAMs and µSRAMs.

#### 4.1.1.1 Power-Down Sequence

Assertion of DEVRST_N or reset initiated from the fabric triggers the system controller to power down the device in the following sequence:

1. The reset signal propagates as a non-maskable interrupt to the system controller, which first disables all I/Os.
2. The fabric gets powered down.
3. Resets are issued to all peripherals, such as transceivers, PCIe, PLLs, and DLLs.

#### 4.1.2 Resets Initiated from POR Circuitry

POR circuitry releases the system controller from the Reset state when all voltage supplies ($V_{DD}$, $V_{DD18}$ and $V_{PD2}$) reach their minimum threshold levels. If any of the supplies fall below the minimum requirement, the device reset is issued and initiates the device power down sequence.
4.2 Initialization Signals to Fabric

The system controller provides initialization signals to the fabric logic using the INIT component. The following signals are asserted during the design initialization:

- DEVICE_INIT_DONE: asserted once the execution of design initialization is complete and remains high. For more information, see Figure 2, page 4.
- FABRIC_POR_N: indicates that the fabric is operational.
- PCIE_INIT_DONE: used by fabric logic to hold PCIe-related fabric logic in reset until the PCIe controller is initialized.
- SRAM_INIT_DONE: indicates that the LSRA M blocks are active.
- USRAM_INIT_DONE: indicates that the μSRAM blocks are active.
- BANK_#_CALIB_STATUS: This signal can be used by user logic to determine whether calibration got completed for each I/O bank.
- BANK_#_VDDI_STATUS: This signal can be used to monitor if there is VDDI power loss on specific I/O banks.

PolarFire Initialization Monitor (INIT component) IP is available in the IP Catalog under Clock and Management as shown in the following figure.

![PolarFire Initialization Monitor](image)

**Note:** The state of the DEVICE_INIT_DONE signal cannot be restored after exiting from the system controller suspend mode. For more information about the suspend mode, see PolarFire System Controller User Guide (to be released).
PolarFire Initialization Monitor provides simulation support. Use the Simulation Options tab to specify the time of releasing the output signals from the zero time instance. Figure 28, page 25 shows the Simulation Options tab.
Figure 28 • PolarFire Initialization Monitor configurator - Simulation options
### 4.3 Functional Example

PolarFire Initialization Monitor (PF_INIT_MONITOR component) can be used for issuing reset to the user logic. The following figure shows an example use case of PF_INIT_MONITOR. In this example, the DEVICE_INIT_DONE signal is connected to a logical AND with lock signal from PF_CCC macro to give a glitch free reset signal to the user logic. The DEVICE_INIT_DONE signal gets asserted after the execution of design initialization is complete.

For more information about the PF_CCC macro, see UG0684: PolarFire FPGA Clocking Resources User Guide.

*Figure 29 • Example of PolarFire Initialization*

### 4.4 CoreReset_PF IP Core

Libero SoC PolarFire provides the CoreReset_PF IP core as part of the IP catalog as shown in Figure 30, page 26. This IP core synchronously de-asserts the reset to the downstream logic in user-specified clock domain. As a result, the reset assertion is asynchronous but the negation is synchronous to the clock. This IP core ensures that the recovery time is met and that all of the flip-flops come out of reset in the same clock period. The CoreReset_PF IP also:

- Combines the resets from multiple sources like external GPIO, PLL lock, and PF_INIT_MONITOR blocks.
- Ensures that there are no inadvertent reset of fabric flip-flops when the fabric is powered down due to Flash*Freeze.

*Figure 30 • The CoreRest_PF IP*

**Note:** For more information about the CoreReset_PF IP, see the CoreReset_PF handbook available in Libero catalog.
This section discusses access time of the following listed individual devices.

- PCIe
- Transceivers
- DDR
- Fabric logic

**Note:** Information will be provided in future releases.
The following table lists the power supplies.

**Table 4 • Power Supplies in PolarFire Devices**

<table>
<thead>
<tr>
<th>Power Supply</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{DD}$</td>
<td>For fabric core and transceiver/PCIe blocks.</td>
</tr>
<tr>
<td>$V_{DD18}$</td>
<td>For fabric programming and RC oscillators.</td>
</tr>
<tr>
<td>$V_{DD25}$</td>
<td>For corner phase-locked loop (PLLs) and private non-volatile memory (sNVM).</td>
</tr>
<tr>
<td>$V_{DDIx}$</td>
<td>For I/O banks.</td>
</tr>
<tr>
<td>$V_{DDAUXx}$</td>
<td>For GPIO and HSIO banks.</td>
</tr>
<tr>
<td>$V_{DDA}$</td>
<td>For transceiver receiver, transmit, and common circuits.</td>
</tr>
<tr>
<td>$V_{DDA25}$</td>
<td>For transceiver PLLs.</td>
</tr>
<tr>
<td>$V_{DD_XCVR_CLK}$</td>
<td>For transceiver reference clock input buffers.</td>
</tr>
</tbody>
</table>

**Note:** These power supplies are expected to ramp monotonically from 0 V to the minimum recommended operating voltage. I/O banks can be kept in power-down state if not used in the design.