Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer’s responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.
# Contents

1 Revision History ............................................................... 1
  1.1 Revision 8.0 ................................................................. 1
  1.2 Revision 7.0 ................................................................. 1
  1.3 Revision 6.0 ................................................................. 1
  1.4 Revision 5.0 ................................................................. 1
  1.5 Revision 4.0 ................................................................. 1
  1.6 Revision 3.0 ................................................................. 1
  1.7 Revision 2.0 ................................................................. 1
  1.8 Revision 1.0 ................................................................. 1

2 PolarFire FPGA Transceiver Demo ........................................... 2
  2.1 Design Requirements ..................................................... 2
  2.2 Prerequisites ............................................................... 3
  2.3 Demo Designs ............................................................. 3
  2.4 Reference Design 1: 8b10b ............................................... 3
    2.4.1 Design Implementation ............................................... 4
    2.4.2 Pattern Generator and Checker ..................................... 5
    2.4.3 Port Description ....................................................... 5
    2.4.4 Simulating the Design ................................................ 5
    2.4.5 Simulation Flow ....................................................... 6
  2.5 Reference Design 2: PMA Design ....................................... 7
    2.5.1 Design Implementation ............................................... 8
    2.5.2 PRBS Generator and Checker ......................................... 9
    2.5.3 Port Description ....................................................... 9
    2.5.4 Simulating the Design ................................................ 9
    2.5.5 Simulation Flow ....................................................... 10
  2.6 Reference Design 2B: PMA with Bit-slip Feature .................... 11
    2.6.1 Design Implementation ............................................... 12
    2.6.2 Pattern Generator and Checker ..................................... 12
    2.6.3 Bit-slip Shift .......................................................... 13
    2.6.4 Port Description ....................................................... 13
    2.6.5 Simulating the Design ................................................ 13
    2.6.6 Simulation Flow ....................................................... 14
  2.7 Reference Design 3: 64b66b Design .................................... 15
    2.7.1 Design Implementation ............................................... 16
    2.7.2 Pattern Generator and Checker ..................................... 16
    2.7.3 Port Description ....................................................... 16
    2.7.4 Simulating the Design ................................................ 17
    2.7.5 Simulation Flow ....................................................... 18
  2.8 Clocking Structure ....................................................... 19
  2.9 Reset Structure .......................................................... 20

3 Libero Design Flow .......................................................... 21
  3.1 Synthesize ................................................................. 21
  3.2 Resource Utilization ..................................................... 21
  3.3 Place and Route .......................................................... 23
    3.3.1 XCVR Placement for Evaluation kit .................................. 23
    3.3.2 XCVR Placement for Splash Kit ..................................... 24
  3.4 Verify Timing ............................................................. 25
Figures

Figure 1  8b10b Design Block Diagram ................................................................. 3
Figure 2  8b10b Design Implementation ............................................................... 4
Figure 3  Simulating the 8b10b Design ................................................................. 6
Figure 4  Simulation Waveform for 8b10b Design Highlighting Pattern Checker Status 6
Figure 5  Simulation Waveform for 8b10b Design Highlighting Tx and Rx Data Match 7
Figure 6  PMA Design Block Diagram ................................................................. 7
Figure 7  PMA Design Implementation ................................................................. 8
Figure 8  Simulation Waveform for PMA Design Highlighting PRBS Checker Status 10
Figure 9  Simulation Waveform for PMA Design Highlighting PRBS Checker Lock 11
Figure 10  PMA with Bit-slip Feature Block Diagram ........................................... 11
Figure 11 Design Implementation of PMA with Bit-slip Feature ............................ 12
Figure 12 Simulation Waveform for PMA Design Highlighting CDR Bit-slip Status 14
Figure 13 Simulation Waveform for PMA Design Highlighting Pattern Checker Lock 15
Figure 14  64b66b Design Block Diagram ........................................................... 15
Figure 15  64b66b Design Implementation ........................................................... 16
Figure 16 Simulating the 64b66b Design .............................................................. 17
Figure 17 Simulation Waveform for 64b66b Design Highlighting Pattern Checker Status 18
Figure 18 Simulation Waveform for 64b66b Design Highlighting Tx and Rx Data Match 18
Figure 19 Clocking Structure ................................................................................. 19
Figure 20 Reset Structure ...................................................................................... 20
Figure 21 Synthesize ............................................................................................ 21
Figure 22 I/O Editor—Transceiver View .............................................................. 23
Figure 23 I/O Editor—Transceiver View for PMA with Bit-slip Design ................. 23
Figure 24 I/O Editor .............................................................................................. 24
Figure 25 Place and Route ................................................................................... 24
Figure 26 Design Flow ......................................................................................... 25
Figure 27 Generate Design Initialization Data ....................................................... 25
Figure 28 Board Setup for Evaluation Kit ............................................................. 26
Figure 29 Programming the Device .................................................................... 26
Figure 30 Board Setup for Splash Kit .................................................................. 27
Figure 31 Programming the Device .................................................................... 27
Figure 32 Programming the Device with FlashPro5 ............................................ 28
Figure 33 Installing PMA_PCS Demo Application .............................................. 29
Figure 34 PMA_PCS Application Installation Steps ............................................. 29
Figure 35 Successful Installation of PMA_PCS Application ................................. 30
Figure 36 Selecting COM Port and Connecting .................................................. 30
Figure 37 PMA_PCS Status Signals .................................................................... 31
Figure 38 Generate Data Error ............................................................................ 31
Figure 39 Clear Data Error .................................................................................. 32
Figure 40 PolarFire FPGA Transceiver .............................................................. 34
Figure 41 SmartBert Design Block Diagram ....................................................... 35
Figure 42 CoreSmartBert IP Design Implementation ........................................ 35
Figure 43 Launching SmartDebug Design Tools ............................................... 36
Figure 44 SmartDebug Window Debug Options ................................................. 36
Figure 45 Debug TRANSCEIVER—Pattern Selection ....................................... 37
Figure 46 Debug TRANSCEIVER—Status ........................................................... 37
Figure 47 SmartBert—Cumulative Error Count .................................................. 37
Figure 48 Create SmartDebug Project ................................................................. 38
# Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Design Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>8b10b Port List</td>
<td>5</td>
</tr>
<tr>
<td>Table 3</td>
<td>Port List for the PMA Design</td>
<td>9</td>
</tr>
<tr>
<td>Table 4</td>
<td>Port List for the PMA Design</td>
<td>13</td>
</tr>
<tr>
<td>Table 5</td>
<td>Port List for the 64b66b Design</td>
<td>16</td>
</tr>
<tr>
<td>Table 6</td>
<td>8b10b Design Resource Utilization</td>
<td>21</td>
</tr>
<tr>
<td>Table 7</td>
<td>PMA Design Resource Utilization</td>
<td>22</td>
</tr>
<tr>
<td>Table 8</td>
<td>64b66b Design Resource Utilization</td>
<td>22</td>
</tr>
<tr>
<td>Table 9</td>
<td>PMA with Bit-slip Design Resource Utilization</td>
<td>22</td>
</tr>
<tr>
<td>Table 10</td>
<td>SmartBert Design Resource Utilization</td>
<td>22</td>
</tr>
<tr>
<td>Table 11</td>
<td>Jumper Settings on Evaluation Kit</td>
<td>26</td>
</tr>
<tr>
<td>Table 12</td>
<td>Jumper Settings on Splash Kit</td>
<td>27</td>
</tr>
<tr>
<td>Table 13</td>
<td>Port List for the CoreSmartBert IP Design</td>
<td>36</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 current publication.

1.1 Revision 8.0
Merged Splash kit related content and updated the document for Libero SoC PolarFire v2.3.

1.2 Revision 7.0
The document was updated for Libero SoC PolarFire v2.2.

1.3 Revision 6.0
The document was updated for Libero SoC PolarFire v2.1.

1.4 Revision 5.0
The following is a summary of the changes made in revision 5.0 of this document.
• The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v2.0 release.
• Information about the usage of the SmartBert IP was added. See Appendix: How to Use SmartBert IP, page 35.

1.5 Revision 4.0
The design files were updated to include enhancements to the GUI logic.

1.6 Revision 3.0
The following is a summary of the changes made in revision 3.0 of this document.
• The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v1.1 SP1 release.
• Information about programming the device and running the demo was added. See Programming the Device Using FlashPro, page 28 and Running the Demo, page 29.
• A list of reference documents was added. See Appendix: References, page 39.

1.7 Revision 2.0
Revision 2.0 of this document included the demonstration of PolarFire transceivers used in 64b66b mode. See Reference Design 3: 64b66b Design, page 15.

1.8 Revision 1.0
The first publication of this document.
2 PolarFire FPGA Transceiver Demo

Each PolarFire® FPGA family includes embedded low-power, performance-optimized transceivers. The transceivers include the physical media attachment (PMA), the associated logic of the protocol physical coding sub-layer (PCS), and interfaces to the FPGA fabric. Each lane in the multi-lane architecture natively supports serial data transfer rates ranging from 250 Mbps to 12.7 Gbps. The transceivers can be configured either with PMA only, or embedded PCS with 8b10b, 64b66b, PIPE, and PCIe interface modes for connecting to the fabric. For an overview of PolarFire transceivers, see Appendix: PolarFire Transceiver Overview, page 33.

This guide presents five designs that demonstrate the use of PolarFire transceivers in PMA, 8b10b, 64b66b modes and SmartBert IP. The current version of this document includes designs that provide Libero® design flow examples, HDL simulation, and transceiver validation on a PolarFire Evaluation board/Splash board. These reference designs shows how to configure and create simple PolarFire FPGA transceiver designs using Libero SoC PolarFire software.

The Multi-rate transceiver reference design can be programmed using any of the following options:

- Using the stp file: To program the device using the stp file provided along with the design files, see Programming the Device Using FlashPro, page 28.
- Using Libero SoC PolarFire: To program the device using Libero SoC PolarFire, see Libero Design Flow, page 21. Use this option when the reference design is modified.

Note: You can use the reference designs to evaluate the performance and features of the transceiver to your design requirements.

2.1 Design Requirements

The following table lists the hardware and software required to run the demo.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>64-bit Windows 7, 8.1, or 10</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td></td>
</tr>
<tr>
<td>PolarFire Evaluation Kit (MPF300-EVAL-KIT)</td>
<td>Rev D or later</td>
</tr>
<tr>
<td>PolarFire Splash Kit (MPF300TS-1FCG484EES)</td>
<td>Rev 2 or later</td>
</tr>
<tr>
<td>– PolarFire Evaluation Board/Splash Board</td>
<td></td>
</tr>
<tr>
<td>– 12 V/5 A power adapter and cord</td>
<td></td>
</tr>
<tr>
<td>– USB 2.0 A-male to mini-B cable for UART and programming</td>
<td></td>
</tr>
<tr>
<td>2 SMA-to-SMA cables with 10 Gbps support (not provided with the kit)1</td>
<td></td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>Libero SoC PolarFire</td>
<td>v2.3</td>
</tr>
<tr>
<td>ModelSim</td>
<td>10.5c Pro</td>
</tr>
<tr>
<td>Synplify Pro</td>
<td>L2017.09M-SP1-1</td>
</tr>
</tbody>
</table>

1. Applicable only for PolarFire Evaluation Kit.
2.2 Prerequisites

Before you start:

1. Download and unzip the design files from the following links:
   For Evaluation Kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0759_eval_liberosocpolarfirev2p3_df
   For Splash Kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0759_splash_liberosocpolarfirev2p3_df
2. Download and install Libero SoC PolarFire v2.3 on the host PC from the following location.
   https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc-polarfire#downloads
   The latest versions of ModelSim, Synplify Pro, and FTDI drivers are included in the Libero SoC PolarFire installation package.

2.3 Demo Designs

There are five reference designs associated with this guide. They are:

- PMA reference design and programming file, located in the PF_XCVR_PMA folder
- PMA with bit-slip design and programming file, located in the PF_XCVR_PMA_with_Bit_Slip folder
- 8b10b reference design and programming file, located in the PF_XCVR_8B10B folder
- 64b66b reference design and programming file, located in the PF_XCVR_64B66B folder
- SmartBert IP reference design and programming file, located in the PF_XCVR_SmartBert folder

2.4 Reference Design 1: 8b10b

This design implements the PolarFire FPGA transceiver in 8b10b mode. The design includes a counter 8b10b pattern generator, PolarFire transceiver in 8b10b mode, and a counter sequence checker.

The following illustration shows the block diagram of the design.

Figure 1 • 8b10b Design Block Diagram
2.4.1 Design Implementation

The following figure shows the Libero SoC PolarFire software design implementation of the transceiver 8b10b design.

Figure 2 • 8b10b Design Implementation

2.4.1.1 PolarFire Transceiver Configurator

The PolarFire transceiver interface configurator is set to 5 Gbps, 32-bit PCS-Fabric interface width and 8b10b mode.

2.4.1.2 PolarFire Transceiver Reference Clock

The transceiver reference clock can be configured either as a differential clock, or two single-ended REFCLKs. This demo requires a single REFCLK. The REFCLK can source transceivers, and global clock network in this design. The reference clock 0 is configured as a differential reference clock.

2.4.1.3 Transmit PLL

The transmit PLLs reference clock and desired output clock are set to 156.25 MHz and 5000 Mbps, respectively. The PolarFire transceiver is a half-rate architecture, that is, the internal high-speed path uses both edges of the clock to keep the clock rates down. The clock thus runs at half of the data rate, thereby consuming less dynamic power.

2.4.1.4 Clock Conditioning Circuitry

PF_CCC block provides 125 MHz output clock to UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.4.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC (clock domain crossing) data signals between fabric and UART interface.
2.4.2 Pattern Generator and Checker

The pattern generator module implements a 32-bit counter pattern used to drive the transceiver in 8b10b mode. Packets created in the pattern generator are separated by a K28.5 character. The packet payload is a simple incremental counting pattern. The pattern checker checks the packet and generates error flags if mismatch occurs, which can be monitored in the GUI application running in the host PC.

2.4.3 Port Description

The following table lists the important ports for the design.

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANE0_RXD_P</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_RXD_N</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>REF_CLK_PAD_P</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface.</td>
</tr>
<tr>
<td>REF_CLK_PAD_N</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface.</td>
</tr>
<tr>
<td>LANE0_PCS_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PCS lane.</td>
</tr>
<tr>
<td>generate_err_i</td>
<td>Input</td>
<td>Injecting Error, active high.</td>
</tr>
<tr>
<td>clear_i</td>
<td>Input</td>
<td>Error counter clear input, active high</td>
</tr>
<tr>
<td>LANE0_TXD_P</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_TXD_N</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_RX_CLK_R</td>
<td>Output</td>
<td>Regional receive clock to fabric.</td>
</tr>
<tr>
<td>LANE0_TX_CLK_R</td>
<td>Output</td>
<td>Regional transmit clock to fabric.</td>
</tr>
<tr>
<td>LANE0_RX_VAL</td>
<td>Output</td>
<td>Receiver data valid flag associated with a lane.</td>
</tr>
<tr>
<td>error_o</td>
<td>Output</td>
<td>Error Flags.</td>
</tr>
<tr>
<td>lock_o</td>
<td>Output</td>
<td>Data lock flag.</td>
</tr>
<tr>
<td>error_count_o[31:0]</td>
<td>Output</td>
<td>Error count Flags.</td>
</tr>
</tbody>
</table>

2.4.4 Simulating the Design

Before you start:

1. Start Libero SoC PolarFire.
2. In the Projects toolbar, click Open Project.
3. Browse the PF_XCVR_8B10B folder for the 8b10b design.
4. Navigate to the libero_Design folder, select PF_XCVR_8B10B.prjx and click Open. The PolarFire transceiver project opens in Libero SoC PolarFire.
5. Navigate to the Design Hierarchy tab and double-click the top level component. The SmartDesign page opens on the right pane and displays the high-level design.

Note: If not already installed, download the PF_XCVR_REF_CLK, PF_TX_PLL and PF_CCC, and PF_XCVR cores under Libero SoC PolarFire > Catalog.
In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design as shown in the following figure. The ModelSim tool takes around five minutes to complete the simulation.

**Figure 3** • Simulating the 8b10b Design

### 2.4.5 Simulation Flow

The following steps describe the simulation flow:

1. At the start, the transceiver is kept at reset.
2. The **pattern_gen_0** block sends incremental counter pattern with K28.5 character to the transceiver.
3. Transmitter lanes are connected to receiver lanes internally in the testbench stimulus.
4. The **pattern_chk_0** block waits for valid data and starts checking the receiver data.
5. The **Generate_err_i** input can be pulsed to send 32'hFFFFFFEF instead of the counter pattern. This increments the **error_count_o[31:0]**.

The following figures shows the simulation waveform for the 8b10b design highlighting pattern checker status signals and Tx/Rx data match.

**Figure 4** • Simulation Waveform for 8b10b Design Highlighting Pattern Checker Status
2.5 Reference Design 2: PMA Design

The second reference design implements the PolarFire transceiver in PMA mode. The design includes a PRBS pattern generator and PRBS pattern checker, and the PolarFire transceiver in PMA mode. The following figure shows the block diagram for the PMA design.

*On-board loopback is used for Splash Kit.*
2.5.1 Design Implementation

The following figure shows the Libero SoC PolarFire software design implementation of the transceiver PMA design.

Figure 7 • PMA Design Implementation

2.5.1.1 PolarFire Transceiver Configurator

The PolarFire Transceiver block includes the transceiver. The PolarFire Transceiver Interface configurator is set to 5 Gbps, 40-bit PCS-Fabric interface width, and PMA mode.

2.5.1.2 PolarFire Transceiver Reference Clock

The transceiver reference clock can be configured either as a differential, or two single-ended REFCLKs. This demo requires a single REFCLK. The REFCLK can source transceivers, and global clock network in this design. The reference clock 0 is configured as a differential reference clock.

2.5.1.3 Transmit PLL

The transmit PLLs reference clock and desired output clock are set to 156.25 MHz and 5000 Mbps, respectively.

2.5.1.4 Clock Conditioning Circuitry

PF_CCC block provides 125 MHz output clock to UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.5.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC (clock domain crossing) data signals between fabric and UART interface.
2.5.2 PRBS Generator and Checker
The generator implements the PRBS polynomial and generates a continuous sequence of PRBS7 patterns of 40 bits each. The PRBS checker uses input PRBS data from the receiver side of the transceiver to generate PRBS data locally, the two are then compared for data integrity. An error flag is raised if data mismatch occurs.

2.5.3 Port Description
The following table lists the important ports for the design.

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANE0_RXD_P</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_RXD_N</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_PCS_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PCS lane.</td>
</tr>
<tr>
<td>LANE0_PMA_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PMA lane</td>
</tr>
<tr>
<td>LANE0_TXD_P</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_TXD_N</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_RX_CLK_R</td>
<td>Output</td>
<td>Regional receive clock to fabric.</td>
</tr>
<tr>
<td>LANE0_TX_CLK_R</td>
<td>Output</td>
<td>Regional transmit clock to fabric.</td>
</tr>
<tr>
<td>LANE0_TX_CLK_STABLE</td>
<td>Output</td>
<td>Transmits transceiver/PCS lane ready flag.</td>
</tr>
<tr>
<td>LANE0_RX_READY</td>
<td>Output</td>
<td>Receives transceiver/PCS lane ready flag.</td>
</tr>
<tr>
<td>LANE0_RX_VAL</td>
<td>Output</td>
<td>Receives data valid flag associated with a lane.</td>
</tr>
<tr>
<td>LANE0_RX_IDLE</td>
<td>Output</td>
<td>Receives electrical-idle detection flag.</td>
</tr>
</tbody>
</table>

2.5.4 Simulating the Design
The pattern generator module generates a PRBS pattern used to drive the transceiver in PMA mode, and the pattern checker checks the received PRBS data and generates error flags if data mismatch occurs.

Before you start:
1. Start Libero SoC PolarFire.
2. In the Projects toolbar, click Open Project.
3. Browse the PF_XCVR_PMA folder for the PMA design.
4. Navigate to the libero_Design folder, select PF_XCVR_PMA.prjx and click Open. The PolarFire transceiver project opens in Libero SoC PolarFire.
5. Navigate to the Design Hierarchy tab and double-click the top level component. The SmartDesign page opens on the right pane and displays the high-level design.
Now, you can view all of the design blocks and IP cores instantiated in the design.

Note: If not already installed, download the PF_XCVR_REF_CLK, PF_TX_PLL and PF_CCC, and PF_XCVR cores under Libero SoC PolarFire > Catalog.
In the **Design Flow** tab, double-click **Simulate under Verify Pre-Synthesized Design** to simulate the design as shown in the following figure. The ModelSim tool takes about 5 minutes to complete the simulation. Simulating the PMA Design

---

### 2.5.5 Simulation Flow

The following steps describe the simulation flow:

1. At the start, the transceiver is kept at reset.
2. The `PRBS_gen` block sends 40-bit wide PRBS7 pattern to the transceiver.
3. Transmitter lanes are connected to receiver lanes internally in the testbench stimulus.
4. The `PRBS_chk` block waits for the valid data and starts checking the receiver data.

The following figures show the simulation waveform for the PMA design highlighting PRBS checker status signals and Tx/Rx PRBS data lock.

**Figure 8**  •  **Simulation Waveform for PMA Design Highlighting PRBS Checker Status**
2.6 Reference Design 2B: PMA with Bit-slip Feature

This reference design example implements the PolarFire transceiver in native PMA mode with bit-slip feature. It includes a pattern generator to generate a 40-bit counter. The pattern checker checks the received data and compares it with the expected counter. The PolarFire transceiver is configured in native PMA mode.

The deserializer has a bit-slip feature for word alignment. In this mode, the CDR slips to the next bit from the deserializer. This feature helps with building word alignment logic in the fabric. It is available for PMA only applications using fabric-based alignment. The configurator enables this using RX_SLIP input port.

For more information about bit-slip feature, see the UG0677: PolarFire FPGA Transceiver User Guide.

The following figure shows the block diagram for the PMA design with bit-slip feature.

*On-board loopback is used for Splash Kit.
2.6.1 Design Implementation

The following figure shows the Libero SoC PolarFire software design implementation of the transceiver PMA design with bit-slip feature.

Figure 11 • Design Implementation of PMA with Bit-slip Feature

2.6.1.1 PolarFire Transceiver Configurator

The PolarFire Transceiver block includes the transceiver. The PolarFire Transceiver Interface configurator is set to 5 Gbps, 40-bit PCS-Fabric interface width, and PMA mode.

2.6.1.2 PolarFire Transceiver Reference Clock

The transceiver reference clock can be configured either as a differential, or two single-ended REFCLKs. This demo requires a single REFCLK. The REFCLK can source transceivers, and global clock network in this design. The reference clock 0 is configured as a differential reference clock.

2.6.1.3 Transmit PLL

The transmit PLLs reference clock and desired output clock are set to 125 MHz and 5000 Mbps, respectively.

Note: A few PolarFire Evaluation boards may have an inconsistent 125 MHz oscillator that does not consistently supply 125 MHz. Due to this behavior, there may be a mismatch between Tx and Rx words, resulting in payload error and a negative Rx_Lock status. If this occurs, change the clock source to the on-board 122.88 MHz oscillator by opening the J46 jumper pins. This changes the data rate to 4.9152 Gbps. There is no other impact to the design.

2.6.1.4 Clock Conditioning Circuitry

PF_CCC block provides 125 MHz output clock to UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.6.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC (clock domain crossing) data signals between fabric and UART interface.

2.6.2 Pattern Generator and Checker

The pattern generator transmits K28.5 character and waits for the symbol alignment to occur with the help of CDR bit-slip feature. When symbol alignment occurs on the receiver side, valid_signal gets asserted from Bit_slip_shift_0 module. Transmitter starts generating 40-bit incremental counter pattern when valid_signal is asserted. The pattern checker checks the received data and compares it with the expected counter pattern. An error flag is raised if data mismatch occurs. Error flags can be monitored using GUI application running in the host PC.
2.6.3 Bit-slip Shift

Bit-slip shift module receives data from transceiver. Using a finite state machine, it compares if transceiver receiver data is equal to K28.5 character value. If this value is not received, it asserts RX_SLIP port of the transceiver interface until the expected K28.5 character is received. When expected K28.5 character is received, valid_signal is asserted, which is used to trigger finite state machines in pattern_gen_0 and pattern_chk_0 modules.

2.6.4 Port Description

The following table lists the important ports for the design.

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANE0_RXD_P</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_RXD_N</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_PCS_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PCS lane.</td>
</tr>
<tr>
<td>LANE0_PMA_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PMA lane</td>
</tr>
<tr>
<td>LANE0_RX_SLIP</td>
<td>Input</td>
<td>Rising-edge requests for the transceiver lane to CDR slip the parallel boundary by one bit.</td>
</tr>
<tr>
<td>LANE0_TXD_P</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_TXD_N</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_RX_CLK_R</td>
<td>Output</td>
<td>Regional receive clock to fabric.</td>
</tr>
<tr>
<td>LANE0_TX_CLK_R</td>
<td>Output</td>
<td>Regional transmit clock to fabric.</td>
</tr>
<tr>
<td>LANE0_TX_CLK_STABLE</td>
<td>Output</td>
<td>Transmits transceiver/PCS lane ready flag.</td>
</tr>
<tr>
<td>LANE0_RX_READY</td>
<td>Output</td>
<td>Receives transceiver/PCS lane ready flag.</td>
</tr>
<tr>
<td>LANE0_RX_VAL</td>
<td>Output</td>
<td>Receives data valid flag associated with a lane.</td>
</tr>
<tr>
<td>LANE0_RX_IDLE</td>
<td>Output</td>
<td>Receives electrical-idle detection flag.</td>
</tr>
</tbody>
</table>

2.6.5 Simulating the Design

The pattern generator module generates a incremental counter pattern used to drive the transceiver in PMA mode. The pattern checker checks the received counter data and generates error flags if data mismatch occurs.

Before you start:
1. Start Libero SoC PolarFire.
2. In the Projects toolbar, click Open Project.
3. Browse PF_XCVR_PMA_With_Bit_Slip folder for PMA design with bit slip feature.
4. Navigate to the libero_Design folder, select PF_XCVR_PMA_BIT_SLIP.prjx and click Open. The PolarFire transceiver project opens in Libero SoC PolarFire.
5. Navigate to the Design Hierarchy tab and double-click the top level component. The SmartDesign page opens on the right pane and displays the high-level design. Now, you can view all of the design blocks and IP cores instantiated in the design.

Note: If not already installed, download the PF_XCVR_REF_CLK, PF_TX_PLL, PF_CCC, and PF_XCVR cores under Libero SoC PolarFire > Catalog.
In the Design Flow tab, double-click **Simulate under Verify Pre-Synthesized Design** to simulate the design as shown in the following figure. The ModelSim tool takes about five minutes to complete the simulation.

2.6.6 Simulation Flow

The following steps describe the simulation flow:

1. At the start, the transceiver is kept at reset.
2. The Pattern_gen block sends 40-bit wide K28.5 pattern to the transceiver.
3. Transmitter lanes are connected to receiver lanes internally in the testbench stimulus.
4. Bit_slip_shift module receives data from transceiver and keeps asserting RX_SLIP port until symbol alignment occurs and then asserts valid_signal.
5. After symbol alignment, pattern_gen block starts sending incremental counter pattern and pattern_chk block starts checking the receiver data.

The following figures show the simulation waveform for the PMA design highlighting pattern_chk block status signals and Tx/Rx PRBS data lock.

*Figure 12 • Simulation Waveform for PMA Design Highlighting CDR Bit-slip Status*
2.7 Reference Design 3: 64b66b Design

This reference design example implements the PolarFire transceiver in 64b66b mode. It includes a pattern generator to generate a 64-bit counter. The pattern checker checks the received data and compares it with the expected counter. The PolarFire transceiver is configured in 64b66b mode.

The following figure shows the block diagram for the 64b66b design.

*: On-board loopback is used for Splash Kit.
2.7.1 Design Implementation

The following figure shows the Libero Soc PolarFire software design implementation of the transceiver in 64b66b mode, 10 Gbps and PCS-Fabric interface width of 64.

Figure 15 • 64b66b Design Implementation

2.7.1.1 PolarFire Transceiver Configurator

The PolarFire Transceiver block includes the transceiver block. The PolarFire Transceiver configurator is 64b66b mode.

2.7.1.2 PolarFire Transceiver Reference Clock

The reference clock 0 is configured as a differential reference clock.

2.7.1.3 Transmit PLL

The transmit PLLs reference clock and desired output clock are set to 156.25 MHz and 10312.5 Mbps, respectively.

2.7.1.4 Clock Conditioning Circuitry

PF_CCC block provides 125 MHz output clock to UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.7.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC (clock domain crossing) data signals between fabric and UART interface.

2.7.2 Pattern Generator and Checker

The pattern generator generates a 64-bit counter. The pattern checker checks the received data and compares it with the expected counter. If the received sequence does not match the one transmitted by the generator, the checker raises an error flag.

2.7.3 Port Description

The following table lists the important ports for the design.

Table 5 • Port List for the 64b66b Design

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANE0_RXD_P</td>
<td>Input</td>
<td>Transceiver receiver differential input.</td>
</tr>
<tr>
<td>LANE0_RXD_N</td>
<td>Input</td>
<td>Transceiver receiver differential input.</td>
</tr>
<tr>
<td>REF_CLK_PAD_P</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface</td>
</tr>
<tr>
<td>REF_CLK_PAD_N</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface</td>
</tr>
<tr>
<td>LANE0_PCS_ARST_N</td>
<td>Input</td>
<td>Asynchronous active-low reset for the PCS logic.</td>
</tr>
</tbody>
</table>
**2.7.4 Simulating the Design**

The pattern generator module generates a counter/prbs-31 pattern used to drive the transceiver in 64b66b mode, and the pattern checker checks the packet and generator error flags.

1. Before you start:
2. Start Libero SoC PolarFire.
3. In the Projects toolbar, click Open Project.
4. Browse the `PF_XCVR_64B66B` folder for the 64b66b design.
5. Navigate to the `libero_Design` folder, select `PF_XCVR_64B66B.prjx` and click `Open`.

The PolarFire transceiver project opens in Libero SoC PolarFire.

6. Navigate to the Design Hierarchy tab and double-click the top level component.
   
   The SmartDesign page opens on the right pane and displays the high-level design. Now, you can view all of the design blocks and IP cores instantiated in the design.

**Note:** If not already installed, download the `PF_XCVR_REF_CLK`, `PF_TX_PLL`, `PF_CCC`, and `PF_XCVR` cores under Libero SoC PolarFire > Catalog.

In the Design Flow tab, double-click Simulate under Verify Pre-Synthesized Design to simulate the design as shown in the following figure. The ModelSim tool takes about five minutes to complete the simulation.

**Figure 16 • Simulating the 64b66b Design**
2.7.5 Simulation Flow

The following steps describe the simulation flow:

1. At the start, the transceiver is kept at reset.
2. The pattern generator sends 64b66b start of sequence (“78 00 00 00 00 00 00 00”) using sync header “10”.
3. Once the lane_status_lock is asserted and transceiver is ready, the pattern generator sends counter pattern using the sync header “01”.
4. Transmitter lanes are connected to receiver lanes internally in the testbench stimulus.
5. The pattern checker waits for the valid data and starts checking the received data.

The following figure shows the simulation waveform for the 64b66b design highlighting pattern checker status signals and Tx/Rx data match.

**Figure 17** • Simulation Waveform for 64b66b Design Highlighting Pattern Checker Status

**Figure 18** • Simulation Waveform for 64b66b Design Highlighting Tx and Rx Data Match
2.8 Clocking Structure

In the reference designs for Evaluation kit, there are two clock domains. The on-board 156.25 MHz crystal oscillator drives the XCVR reference clock in 8b10b, PMA, and 64b66b designs and on-board 125 MHz crystal oscillator drives the XCVR reference clock in PMA with Bit Slip design. This provides clock source to the Data Counter and Data Checker blocks. The on-chip 160 MHz RC oscillator drives the UART_IF_0 block. The following figure shows the clocking structure in the reference design. For Splash kit, 125 MHz crystal oscillator drives the XCVR reference clock for all the designs.

Figure 19 • Clocking Structure
2.9 Reset Structure

In the 8b10b, PMA, PMA with bit slip and 64b66b mode reference designs, the reset signal of data generator, data checker, and UART blocks are issued using Reset_Block module. Reset_sync_tx_0 (CoreReset_PF) module releases active low reset of data generator block when TX_CLK_STABLE from PF_XCVR interface, DEVICE_INIT_DONE signal from PF_INIT_MONITOR block, and start signal from UART_INTERFACE module are asserted.

Similarly, Reset_sync_rx_0 (CoreReset_PF) module releases active low reset of data checker when RX_READY from PF_XCVR interface, DEVICE_INIT_DONE signal from PF_INIT_MONITOR block, and start signal from UART_INTERFACE module are asserted. This is to ensure that data generation and analysis starts only after transceiver Tx and Rx links are ready and independent.

Reset_sync_uart_0 (CoreReset_PF) module releases active low reset of UART_INTERFACE when PLL_LOCK output from PF_CCC block and DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted.

DEVICE_INIT_DONE signal is asserted when the device initialization is complete. For more information about device initialization, see UG0725: PolarFire FPGA Device Power-Up and Resets User Guide.

For more information on CoreReset_PF IP core, see CoreReset_PF handbook from the Libero catalog.

The following figure shows the reset structure in the reference design.

Figure 20 • Reset Structure
This chapter describes the Libero design flow, which includes the following steps:

- Synthesize
- Place and Route
- Verify timing
- Design and Memory Initialization
- Generate Bitstream
- Run PROGRAM Action

The Libero SoC software design flow is similar for all reference designs.

### 3.1 Synthesize

In the Design Flow window, double-click **Synthesize**. When the synthesis is successful, a green check mark appears as shown in the following figure.

![Synthesize](image)

### 3.2 Resource Utilization

The resource utilization values are with respect to Evaluation Kit.

The following table lists the resource utilization of the 8b10b design after synthesis.

**Note:** These values may vary slightly for different Libero runs, settings, and seed values.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>535</td>
<td>299544</td>
<td>0.18</td>
</tr>
<tr>
<td>DFF</td>
<td>422</td>
<td>299544</td>
<td>0.14</td>
</tr>
<tr>
<td>Transceiver lanes</td>
<td>1</td>
<td>16</td>
<td>6.25</td>
</tr>
<tr>
<td>TX_PLL</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
<tr>
<td>XCVR_REF_CLK</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
</tbody>
</table>
The following table lists the resource utilization of the PMA design after synthesis.

**Table 7 • PMA Design Resource Utilization**

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>640</td>
<td>299544</td>
<td>0.21</td>
</tr>
<tr>
<td>DFF</td>
<td>508</td>
<td>299544</td>
<td>0.17</td>
</tr>
<tr>
<td>Transceiver lanes</td>
<td>1</td>
<td>16</td>
<td>6.25</td>
</tr>
<tr>
<td>TX_PLL</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
<tr>
<td>XCVR_REF_CLK</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
</tbody>
</table>

The following table lists the resource utilization of the 64b66b design after synthesis.

**Table 8 • 64b66b Design Resource Utilization**

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>690</td>
<td>299544</td>
<td>0.23</td>
</tr>
<tr>
<td>DFF</td>
<td>729</td>
<td>299544</td>
<td>0.24</td>
</tr>
<tr>
<td>Transceiver lanes</td>
<td>1</td>
<td>16</td>
<td>6.25</td>
</tr>
<tr>
<td>TX_PLL</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
<tr>
<td>XCVR_REF_CLK</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
</tbody>
</table>

The following table lists the resource utilization of the PMA with bit-slip design after synthesis.

**Table 9 • PMA with Bit-slip Design Resource Utilization**

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>574</td>
<td>299544</td>
<td>0.19</td>
</tr>
<tr>
<td>DFF</td>
<td>421</td>
<td>299544</td>
<td>0.14</td>
</tr>
<tr>
<td>Transceiver lanes</td>
<td>1</td>
<td>16</td>
<td>6.25</td>
</tr>
<tr>
<td>TX_PLL</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
<tr>
<td>XCVR_REF_CLK</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
</tbody>
</table>

The following table lists the resource utilization of the SmartBert design after synthesis.

**Table 10 • SmartBert Design Resource Utilization**

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>6103</td>
<td>299544</td>
<td>2.04</td>
</tr>
<tr>
<td>DFF</td>
<td>1078</td>
<td>299544</td>
<td>0.36</td>
</tr>
<tr>
<td>Transceiver lanes</td>
<td>1</td>
<td>16</td>
<td>6.25</td>
</tr>
<tr>
<td>TX_PLL</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
<tr>
<td>XCVR_REF_CLK</td>
<td>1</td>
<td>11</td>
<td>9.09</td>
</tr>
</tbody>
</table>
3.3 Place and Route

3.3.1 XCVR Placement for Evaluation kit

For 8b10b, Native PMA, 64b66b and SmartBert design, the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. Lane0 of Quad2 is used for on-board SMA loop-back of Tx and Rx. REFCLK is placed to use 156.25 MHz clock source for XCVR, as shown in following figure.

Figure 22 • I/O Editor—Transceiver View

For PMA with Bit-slip design, the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. Lane0 of Quad2 is used for on-board SMA loop-back of Tx and Rx. REFCLK is placed to use 125 MHz clock source for XCVR, as shown in following figure.

Figure 23 • I/O Editor—Transceiver View for PMA with Bit-slip Design
### 3.3.2 XCVR Placement for Splash Kit

For 8b10b, Native PMA, PMA Bit-slip, 64b66b, and SmartBert design; the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. Lane1 of Quad1 is used for on-board loopback of Tx and Rx. REFCLK is placed to use 125 MHz clock source for XCVR, as shown in following figure.

**Figure 24 • I/O Editor**

In the Design Flow window, double-click Place and Route. When place and route is successful, a green check mark appears as shown in the following figure.

**Figure 25 • Place and Route**
3.4 Verify Timing

In the Design Flow window, double-click Verify Timing. When the design successfully meets the timing requirements, a green tick mark appears as shown in the following figure.

Figure 26 • Design Flow

3.5 Design and Memory Initialization

This option is used to create the XCVR initialization client, which is used in the demo design. When the PolarFire device powers up, the transceiver block is initialized by the initialization client generated during the Configure Design Initialization Data and Memories stage in the design flow. For more information about device power-up, see UG0725: PolarFire FPGA Device Power-up and Resets User Guide.

Figure 27 • Generate Design Initialization Data

3.6 Generate Bitstream

To generate the bitstream:

1. Double-click Generate Bitstream from the Design Flow tab. When the bitstream is successfully generated, a green tick mark appears as shown in Figure 31, page 27.
2. Right-click Generate Bitstream and select View Report to view the corresponding log file in the Reports tab.

3.7 Programming the Device

To program the device, see any of the following sections based on the kit used.

• Programming the Device on the Evaluation Kit, page 26
• Programming the Device on the Splash Kit, page 27
### 3.7.1 Programming the Device on the Evaluation Kit

After generating the bitstream, the PolarFire device can be programmed. Follow these steps to program the PolarFire device:

1. Ensure that the jumper settings on the board are same as listed in the following table.

#### Table 11 • Jumper Settings on Evaluation Kit

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J18, J19, J20, J21, J22</td>
<td>Close pin 2 and 3 for programming the PolarFire FPGA through FTDI</td>
</tr>
<tr>
<td>J28</td>
<td>Close pin 2 and 3 for programming through the on-board FlashPro5</td>
</tr>
<tr>
<td>J26</td>
<td>Close pin 1 and 2 for programming through the FTDI SPI</td>
</tr>
<tr>
<td>J27</td>
<td>Close pin 1 and 2 for programming through the FTDI SPI</td>
</tr>
<tr>
<td>J4</td>
<td>Close pin 1 and 2 for manual power switching using SW3</td>
</tr>
<tr>
<td>J12</td>
<td>Close pin 3 and 4 for 2.5 V</td>
</tr>
<tr>
<td>J46</td>
<td>Close pin 1 and 2 for routing 125 MHz differential clock oscillator output to the line side.</td>
</tr>
<tr>
<td></td>
<td>Open pin 1 and 2 for routing 122.88 MHz differential clock oscillator output to the line side.</td>
</tr>
</tbody>
</table>

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to J5 (FTDI port) on the board.
4. Power on the board using the SW3 slide switch.
5. Connect TXN to RXN and TXP to RXP using the 2 SMA to SMA cables as shown in the following figure. The following figure shows the board setup.

#### Figure 28 • Board Setup for Evaluation Kit

When the device is programmed successfully, a green tick mark appears as shown in the following figure. See Running the Demo, page 29 to run the Multi-rate transceiver demo.

#### Figure 29 • Programming the Device
3.7.2 Programming the Device on the Splash Kit

After generating the bitstream, the PolarFire device can be programmed. Follow these steps to program the PolarFire device:

1. Ensure that the jumper settings on the board are same as listed in the following table.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J5, J6, J7, J8, J9</td>
<td>Close pin 2 and 3 for programming the PolarFire FPGA through FTDI</td>
</tr>
<tr>
<td>J11</td>
<td>Close pin 1 and 2 for programming through FTDI chip</td>
</tr>
<tr>
<td>J10</td>
<td>Close pin 1 and 2 for programming through FTDI SPI</td>
</tr>
<tr>
<td>J4</td>
<td>Close pin 1 and 2 for manual power switching using SW1</td>
</tr>
<tr>
<td>J3</td>
<td>Open pin 1 and 2 for 1.0 V</td>
</tr>
</tbody>
</table>

2. Connect the power supply cable to the J2 connector on the board.
3. Connect the USB cable from the Host PC to J1 (FTDI port) on the board.
4. Power on the board using the SW1 slide switch.

The following figure shows the board setup.

Figure 30 • Board Setup for Splash Kit


When the device is programmed successfully, a green tick mark appears as shown in the following figure. See Running the Demo, page 29 to run the Multi-rate transceiver demo.

Figure 31 • Programming the Device
4 Programming the Device Using FlashPro

This chapter describes how to program the PolarFire device with the .stp programming file using FlashPro. The .stp file is available at the following design files folder location:

- **8b10b**: mpf_dg0759_eval/splash_liberosocpolarfirev2p3_df\PF_XCVR_8B10B\Programming_File
- **64b66b**: mpf_dg0759_eval/splash_liberosocpolarfirev2p3_df\PF_XCVR_64B66B\Programming_File
- **PMA**: mpf_dg0759_eval/splash_liberosocpolarfirev2p3_df\PF_XCVR_PMA\Programming_File
- **PMA_WITH_BIT_SLIP**: mpf_dg0759_eval/splash_liberosocpolarfirev2p3_df\PF_XCVR_PMA_With_Bit_Slip\Programming_File
- **SmartBert**: mpf_dg0759_eval/splash_liberosocpolarfirev2p3_df\PF_XCVR_SmartBert\Programming_File

Follow these steps:

1. Ensure that the jumper settings on the board are same as listed in Table 11, page 26 (for Evaluation board) and Table 12, page 27 (for Splash board).
2. Connect the power supply cable to the J9 connector on the Evaluation board or J2 connector on the Splash board.
3. Connect the USB cable from the Host PC to J5 (FTDI port) on the Evaluation board or J1 (FTDI port) on the Splash board.
4. Power on the board using the SW3 slide switch on the Evaluation board or SW1 slide switch on the Splash board.
5. Connect TXN to RXN and TXP to RXP using the 2 SMA to SMA cables on the Evaluation board as shown in Figure 28, page 26.
6. On the host PC, start the FlashPro software.
7. Click New Project to create a new project. In the New Project window, enter project name.
8. Click Browse and navigate to the location where you want to save the project.
9. Select Single device as the programming mode and click OK to save the project.
10. Click Configure Device.
11. Click Browse, and navigate to the location where the PF_XCVR_8B10B.stp or PF_XCVR_64B66B.stp or PF_XCVR_PMA_with_bit_slip.stp or PF_XCVR_PMA.stp PF_XCVR_SmartBert_5G.stp (for 5 Gbps data rate design) or PF_XCVR_SmartBert_6P25G.stp (for 6.25 Gbps data rate design) or PF_XCVR_SmartBert_10G (for 10 Gbps data rate design) file is located and select the file.
12. Click Program to program the device.

Figure 32 • Programming the Device with FlashPro5

When the device is programmed successfully, a Run PASSED status is displayed. See Running the Demo, page 29 to run the Multi-rate transceiver demo.
This chapter describes how to install and use the GUI for selecting the test patterns and monitoring the loopback data.

Follow these steps:

1. Install the **GUI_Installer** (setup.exe) from the following design files folder:
   ```plaintext
   mpf_dg0759_eval/splash_liberoscopolarfirev2p3_df\GUI_Installer
   ```
2. Apply default options as shown in the following figure.

   **Figure 33 • Installing PMA_PCS Demo Application**

   ![Installing PMA_PCS Demo Application](image)

   3. Click **Next** to start the installation.

   **Figure 34 • PMA_PCS Application Installation Steps**

   ![PMA_PCS Application Installation Steps](image)
4. Click **Finish** to complete the installation.

*Figure 35 • Successful Installation of PMA_PCS Application*

5. Go to **All Programs > PMA_PCS > PMA_PCS**. The PMA_PCS Demo window is displayed as shown in the following figure.

6. Select the COM port number that is detected to configure the serial port.

7. Click **Connect** to connect the GUI to the board through the selected port as shown in the following figure. After successfully connecting, the host connection status turns green as shown in the following figure.

*Figure 36 • Selecting COM Port and Connecting*
8. Click **Start** to start the PMA_PCS demo. The data starts getting generated and sent over the serial transmit link. It is then received by the receiver and checked for any errors. The status can be monitored using the status signals on the GUI at any time as shown in the following figure. The following are the status signals:

- Host connection: indicates UART connection status
- Serial Link: indicates transceiver link status
- Rx Lock: indicates if the transmitter and receiver data got locked
- PayLoad error: indicates if there is a data mismatch between pattern generator and checker.

*Figure 37 • PMA PCS Status Signals*

9. Click **Generate Error** to generate error in the data and observe the error status as shown in the following figure.

**Note:** Error counter displayed is a 32-bit counter running at a very high frequency clock. It keeps incrementing until injected error is cleared by clicking **Clear Error**.

*Figure 38 • Generate Data Error*
10. Click **Clear Error** to stop generate error and observe that **Rx Lock** and **Payload Error** turns green, and Error Count is displayed as 0 as shown in the following figure.

**Figure 39 • Clear Data Error**

11. Click **Stop** to stop the PMA_PCS demo.

The Multi-rate transceiver demo is successfully run.
Appendix: PolarFire Transceiver Overview

PolarFire FPGA transceivers include all required analog functions for high-speed data transmission between devices over printed circuit boards (PCB) and high-quality cables. They are optimized for low-power operation and are suitable for a variety of device-to-device communication protocols.

The transceiver supports the following embedded PCS:

- **8b10b**—The 8b10b mode supports only encodes and decodes for interface widths of 16, 32, and 64 bits at the PMA. The 8b10b trans-coders are protocol independent; in other words, they do not include a protocol-specific word aligner or word alignment state machine. Comma-detection is supported in this mode. The serial data must be aligned to comma-alignment boundaries before being used as parallel data. Without proper alignment, the incoming 8b10b data does not decode correctly. The comma character (K28.5) is usually used for alignment, as its 10-bit code is guaranteed not to occur elsewhere in the encoded bit stream. The bit-slip functions in the FPGA fabric can be used to implement the word align or word alignment state machine as required.

- **64b6xb**—The 64b66b/64b67b (64b6xb) interface modes are used for mainly 10 Gbps-based protocols, 10G base interface over Ethernet (10GBASE-R/KR), and common public radio interface (CPRI) rates of 9.830 Gbps, and 40GBASE-R standards. The 64b/66b encoder is used to achieve DC balance and sufficient data transitions for clock recovery. It encodes 64-bit XGMII data and 8-bit XGMII control into 10GBASE-R 66-bit control or data blocks in accordance with Clause 49 of the IEEE802.3-2008 specification.

- **PIPE**—The standard PIPE interface provides a standard interface between the PMA lane and the higher link-level of the PHY. The PHY interface for the PCI Express supports both, PCIe Gen1/2 and SATA 1.0/2.0/3.0.

- **PMA only**—direct access to the PMA without any encoding. The transceiver PMA mode is useful in supporting protocols such as SDI-HD. The PMA Only mode is also used for 1GbE interfaces. The CoreTSE suite of 1GbE IPs contain a soft 8b10b encoder/decoder that allows the use of either the transceiver, or the I/O CDR for implementing this standard.

- **PCIe**—Fully embedded PCIe Gen1/Gen2 root-port or endpoint subsystem (PCIESS) with AXI4 user interfaces with built-in DMA.

For more information, see the UG0677: PolarFire FPGA Transceiver User Guide.
The Microsemi Libero SoC PolarFire design software supports configuring transceivers for various modes of operation. The Libero SoC PolarFire software design tools allow designers to set the configuration needed for a specific operational mode for each transceiver lane. The software correctly provisions and generates all of the required programming and configuration data used to initialize and bring the transceiver into operation. The transceiver configuration registers are set automatically by the Libero Transceiver Interface configurator. These registers must be left at the default values set by the configurator, except for use cases that explicitly request different values.
Appendix: How to Use SmartBert IP

The CoreSmartBert core provides a demonstration platform for PolarFire transceiver (PF_XCVR). It can be customized to use different line rates and reference clock rates. PRBS data pattern generators and checkers are included in the core. The pattern generator sends data out through the transmitter, accepts data through the receiver, and checks it against an internally generated pattern. These patterns are optimized for the logic width that are selected at run time.

Test the PMA functionality of PF_XCVR interface on board and SmartDebug provides the user interface to this core.

7.1 Reference Design: SmartBert IP Design

The reference design implements the PolarFire transceiver in PMA mode. The design includes a CoreSmartBert core along with TX_PLL, and XCVR_REF_CLK macros. The following figure shows the block diagram for the SmartBert design.

![SmartBert Design Block Diagram](image)

*On-board loopback is used for Splash Kit.

7.1.1 Design Implementation

The following figure shows the Libero SoC PolarFire software design implementation of the transceiver PMA design using CoreSmartBert IP.

![CoreSmartBert IP Design Implementation](image)

7.1.1.1 PolarFire CoreSmartBert IP Configurator

The PolarFire CoreSmartBert IP block includes the transceiver along with the in-built PRBS data pattern generators and checkers. The PolarFire Transceiver Interface is set to 5 Gbps and PMA settings are selected.
7.1.1.2 PolarFire Transceiver Reference Clock
The transceiver reference clock can be configured either as a differential, or two single-ended REFCLks. This demo requires a single REFCLK. The REFCLK can source transceivers, and global clock network in this design. The reference clock 0 is configured as a differential reference clock.

7.1.1.3 Transmit PLL
The transmit PLLs reference clock and desired output clock are set to 156.25 MHz and 5000 Mbps, respectively.

7.1.2 Port Description
The following table lists the important ports for the design.

<table>
<thead>
<tr>
<th>Port</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANE0_RXD_P</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>LANE0_RXD_N</td>
<td>Input</td>
<td>Transceiver Receiver differential input.</td>
</tr>
<tr>
<td>REF_CLK_PAD_P</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface.</td>
</tr>
<tr>
<td>REF_CLK_PAD_N</td>
<td>Input</td>
<td>Transmit PLL input clock from reference clock interface.</td>
</tr>
<tr>
<td>LANE0_TXD_P</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
<tr>
<td>LANE0_TXD_N</td>
<td>Output</td>
<td>Transceiver Transmitter differential output.</td>
</tr>
</tbody>
</table>

7.2 How to Use SmartBert
After programming the SmartBert IP design, double-click **Generate SmartDebug FPGA Array Data** to generate SmartDebug data and double-click **SmartDebug Design** in the **Design Flow** window as shown in the following figure.

**Figure 43** • Launching SmartDebug Design Tools

The **SmartDebug** window is displayed, as shown in the following figure.

To access the debug transceiver feature, select **Debug TRANSCEIVER** in the SmartDebug window.

**Figure 44** • SmartDebug Window Debug Options
To run SmartBert in Debug TRANSCEIVER, follow these steps:

1. Select the **SmartBERT** tab in the **Debug TRANSCEIVER** window.
2. Select the **Pattern** from the drop-down list.

**Figure 45 • Debug TRANSCEIVER—Pattern Selection**

3. Click **Start**. It enables both transmitter and the receiver for a particular lane and for a particular PRBS pattern. The following figure shows the status of the TXPLL, RXPLL, Lock to Data, Data rate, and the BER.

**Figure 46 • Debug TRANSCEIVER—Status**

When a SmartBert IP lane is added, the **Error Injection** column is displayed in the right pane. The error injection feature is provided to inject an error while running a PRBS pattern.

4. Click **Reset** to clear the error count under **Cumulative Error Count**. Error Count is displayed when the lane is added.

The following figure shows the **Smart BERT** tab with error count incremented using **Inject Error** in the **Debug TRANSCEIVER** window.

**Figure 47 • SmartBert—Cumulative Error Count**

For more information about Debug transceiver features, see **TU0804: PolarFire FPGA SmartDebug Hardware Design Tools Tutorial**.

**Note:** If any of the probe points in SmartBert tab are not working as expected, see Appendix: Known Issues section in **TU0804: PolarFire FPGA SmartDebug Hardware Design Tools Tutorial**.

The SmartBert reference design is configured for 5Gbps transceiver data rate. Separate programming files (*.stp) and corresponding design debug data container (DDC) for different transceiver data rates are exported from Libero and provided in the following locations:

- **.stp file:** mpf_dg0759_eval/splash_liberosc_polarfirev2p3_dfPF_XCVR_SmartBert/Programming_File
- **.ddc file:** mpf_dg0759_eval/splash_liberosc_polarfirev2p3_dfPF_XCVR_SmartBert/Source_File

Launch SmartDebug in standalone mode and import the DDC file to access all debug features.
Follow the steps to import *.ddc file in standalone SmartDebug.

1. Launch SmartDebug in standalone mode.
2. In the SmartDebug window, click Project > New Project. The Create SmartDebug Project dialog box opens as shown in the following figure.
3. Select the Import from DDC File in the Create SmartDebug Project dialog box and browse the *.ddc file. The design debug data of the target device, all hardware, and JTAG chain information present in the DDC file exported from Libero are automatically inherited by the SmartDebug project.

*Figure 48 • Create SmartDebug Project*

For more information about how to export DDC file from Libero, see TU0804: PolarFire FPGA SmartDebug Hardware Design Tools Tutorial.
8 Appendix: References

This section lists documents that provide more information about the Multi-rate Transceiver and IP cores used in the reference design.

- For more information about dynamic rate change of transceivers, see AC475: PolarFire FPGA Dynamic Reconfiguration Interface Application Note.
- For information about PolarFire transceiver blocks, PF_XCVR, PF_TX_PLL, and PF_XCVR_REF_CLK, see UG0677: PolarFire FPGA Transceiver User Guide.
- For more information about CCC, see UG0684: PolarFire FPGA Clocking Resources User Guide.
- For more information about Libero, ModelSim, and Synplify, see the Microsemi Libero SoC PolarFire web page.
- For more information about device and memory initialization, see UG0725: PolarFire FPGA Device Power-Up and Resets User Guide.
- For more information about SmartDebug features, see TU0804: PolarFire FPGA SmartDebug Hardware Design Tools Tutorial.
- For more information about PolarFire FPGA Evaluation Kit, see UG0747: PolarFire FPGA Evaluation Kit User Guide.
- For more information about PolarFire FPGA Splash Kit, see UG0786: PolarFire FPGA Splash Kit User Guide.
- For more information about CoreUART, see CoreUART Handbook.