DG0759
Demo Guide
PolarFire FPGA Transceiver

a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

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.

50200759. 11.0 1/21

Contents

Figures

Tables

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 11.0

Added Appendix 4: Running the TCL Script, page 43.

1.2 Revision 10.0

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

- Updated the document for Libero SoC v12.2.
- Removed the references to Libero version numbers.

1.3 Revision 9.0

The document was updated for Libero SoC v12.0.

1.4 Revision 8.0

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

1.5 Revision 7.0

The document was updated for Libero SoC PolarFire v2.2.

1.6 Revision 6.0

The document was updated for Libero SoC PolarFire v2.1.

1.7 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 2: How to Use SmartBert IP, page 36.

1.8 Revision 4.0

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

1.9 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 30.
- A list of reference documents was added. See Appendix 5: References, page 44.

1.10 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 16.

1.11 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, or embedded PCS with 8b10b, 64b66b, PIPE, and PCIe interface modes for connecting to the fabric. For an overview of PolarFire transceivers, see Appendix 1: PolarFire Transceiver Overview, page 34.

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 software.

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

- Using the job file: To program the device using the job file provided along with the design files, see Appendix 3: Programming the Device Using FlashPro Express, page 40.
- Using Libero SoC: To program the device using Libero SoC, see Libero Design Flow, page 22. 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 1 • **Design Requirements**

Requirement	Version
Operating system	64-bit Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT) PolarFire Splash Kit (MPF300TS-1FCG484EES)	Rev D or later Rev 2 or later
2 SMA-to-SMA cables with 10 Gbps support (not provided with the kit)	Required only for Evaluation kit.
Software	
Libero SoC	**Note:** Refer to the readme.txt file provided in the design files for the software versions used with this reference design.
ModelSim	
Synplify Pro	

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you begin:

1. Download the demo design files from the following location:
   For Evaluation Kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0759_eval_df
   For Splash Kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0759_splash_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location.
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc

2.3 Demo Designs

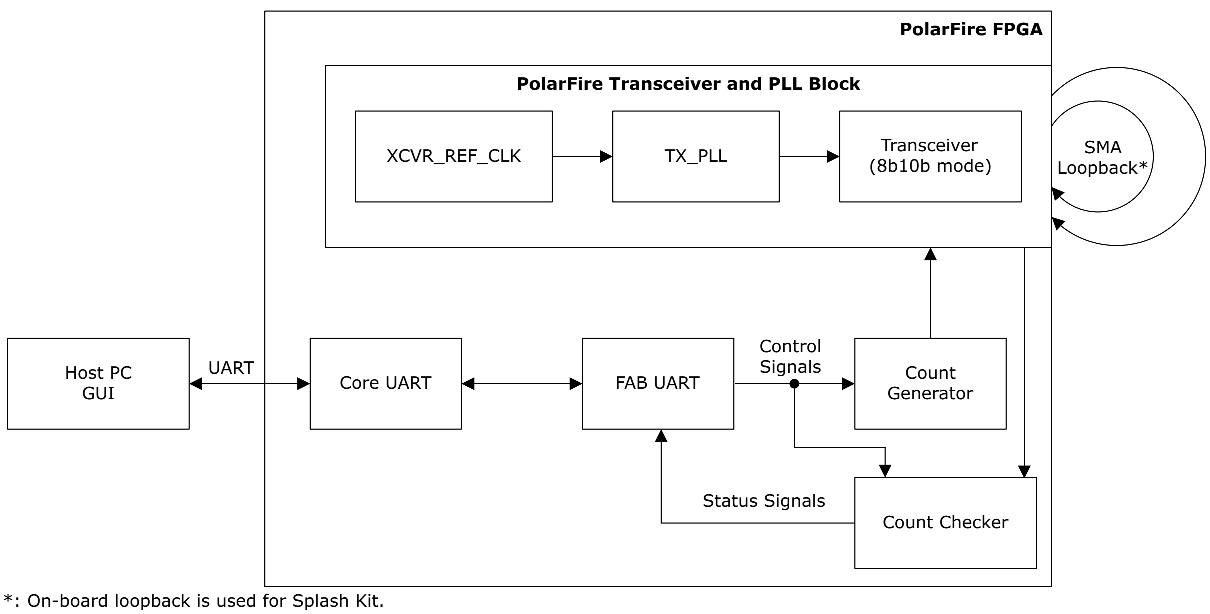
There are five reference designs associated with this guide. They are:

- 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.

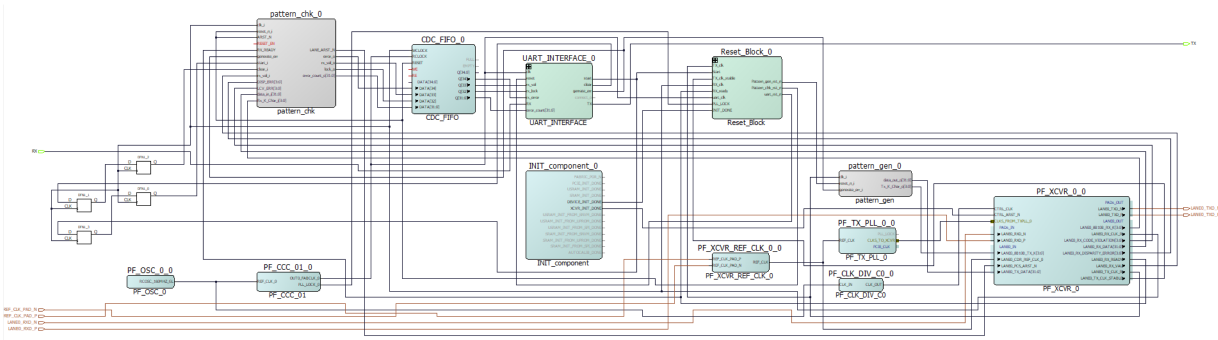
Figure 1 • **8b10b Design Block Diagram**



2.4.1 Design Implementation

The following figure shows the Libero SoC 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 with Enhanced Receiver management.

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 networks 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 that 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 the UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.4.1.5 CDC_FIFO

CoreFIFO block synchronizes Clock Domain Crossing (CDC) 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 a 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 2 • **8b10b Port List**

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

2.4.4 Simulating the Design

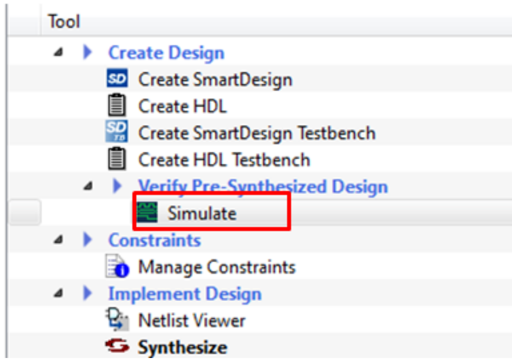
Before you begin:

1. Start Libero SoC.
2. In the Projects toolbar, click **Open Project**.
3. Browse the `PF_XCVR_8B10B` folder for the 8b10b design
4. Navigate to the `Libero_Project` folder, select Libero_Project.prjx and click **Open**.The PolarFire transceiver project opens in Libero SoC.
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 > 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**

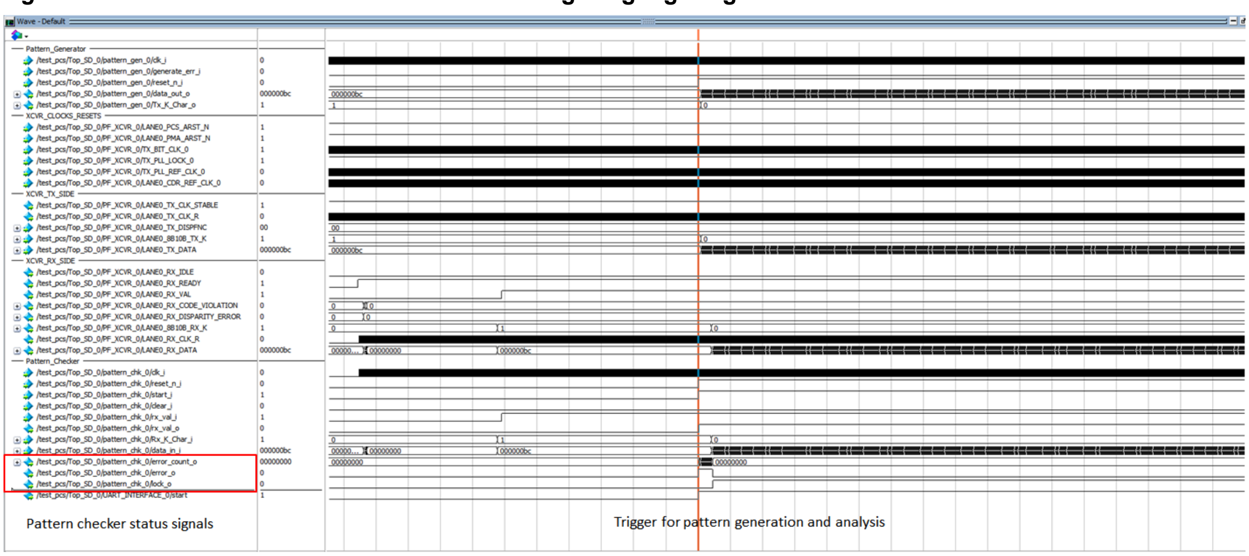
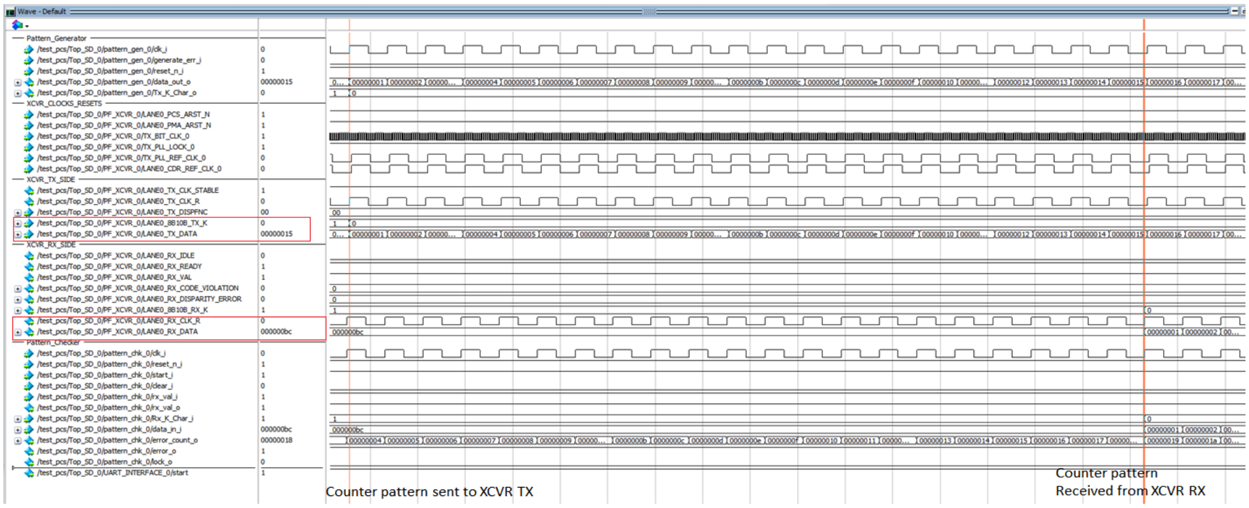


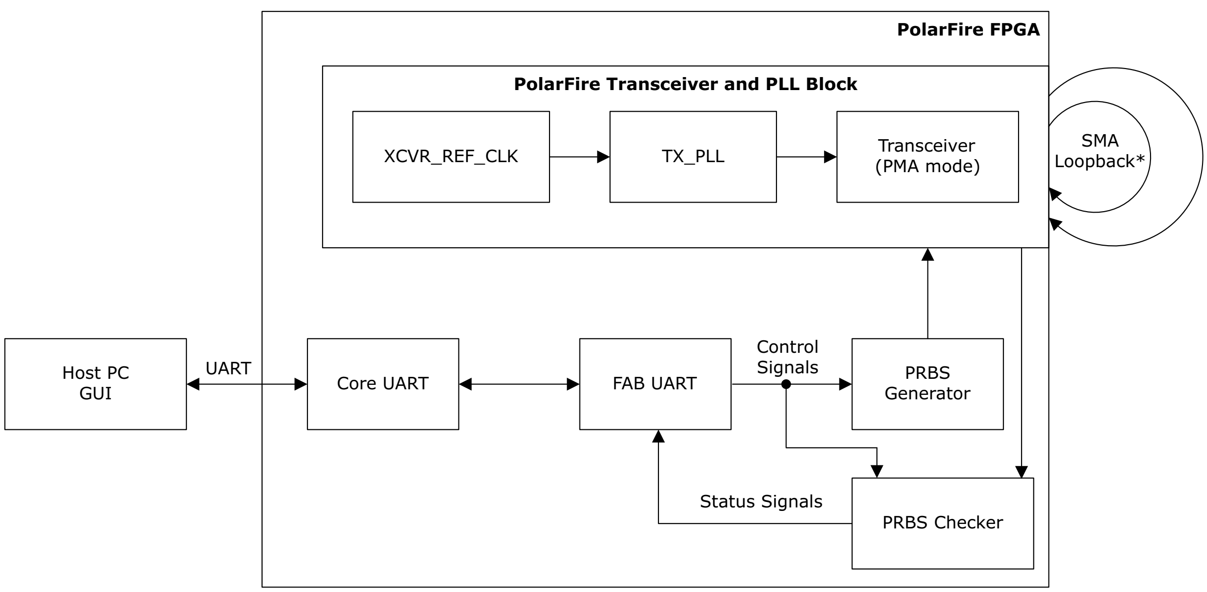
Figure 5 • Simulation Waveform for 8b10b Design Highlighting Tx and Rx Data Match



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, PRBS pattern checker, and the PolarFire transceiver in PMA mode.
The following figure shows the block diagram for the PMA design.

Figure 6 • PMA Design Block Diagram

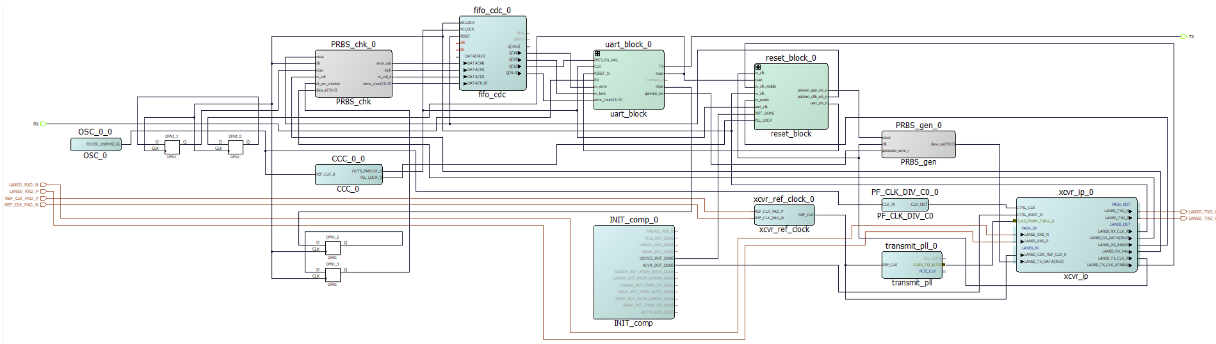


*: On-board loopback is used for Splash Kit.

2.5.1 Design Implementation

The following figure shows the Libero SoC 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 with Enhanced Receiver management.

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 networks 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 the UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.5.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC 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 3 • **Port List for the PMA Design**

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

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.

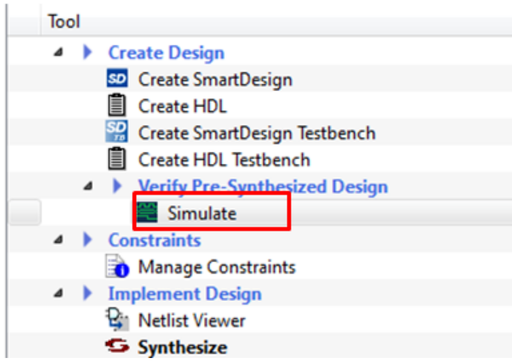
Before you begin:

1. Start Libero SoC.
2. In the Projects toolbar, click Open Project.
3. Browse the `PF_XCVR_PMA` folder for the PMA design.
4. Navigate to the `Libero_Project` folder, select Libero_Project.prjx and click **Open**
   The PolarFire transceiver project opens in Libero SoC.
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 > 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**

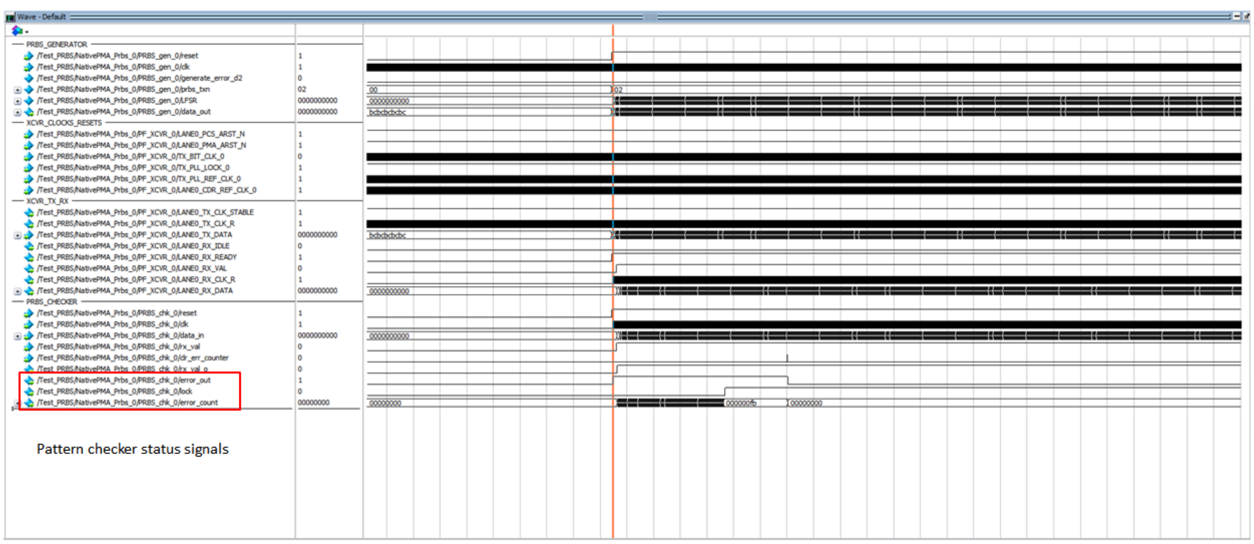
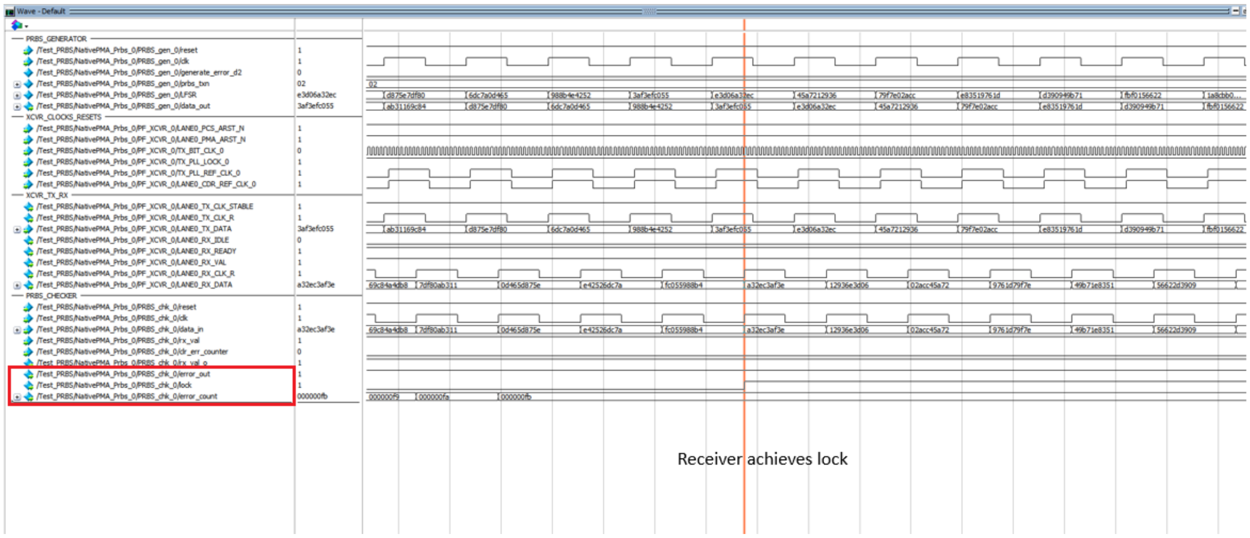


Figure 9 • **Simulation Waveform for PMA Design Highlighting PRBS Checker Lock**



2.6 Reference Design 2B: PMA with Bit-slip Feature

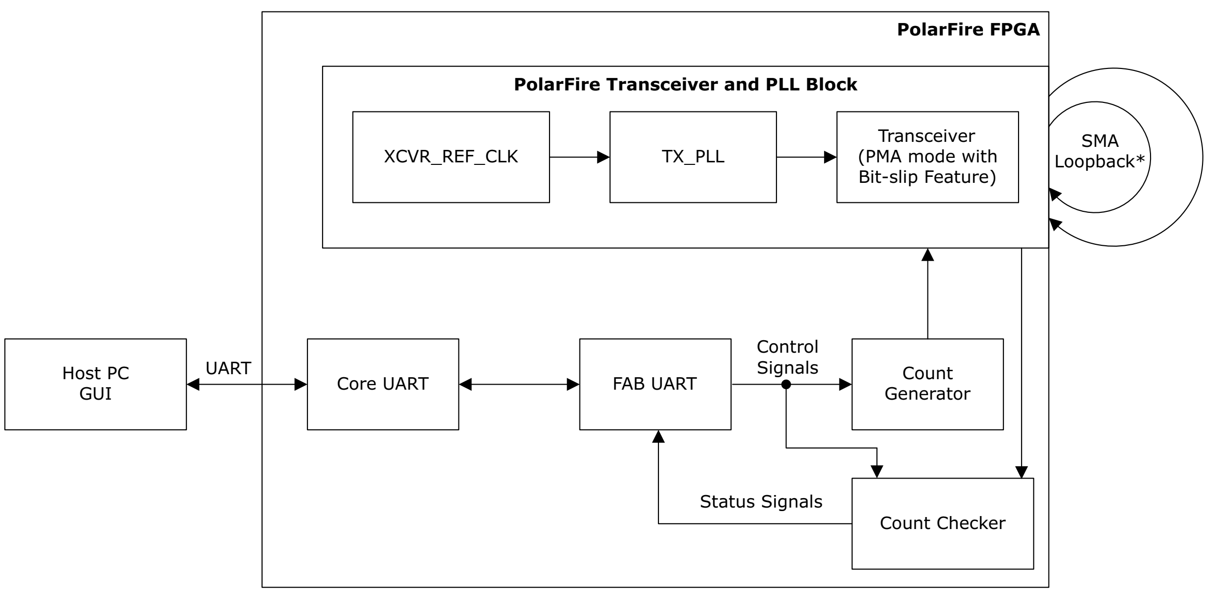
This reference design example implements the PolarFire transceiver in native PMA mode with bit-slip feature. It includes a pattern generator to generate a 40-bit counter. The pattern checker checks the received data and compares it with the expected counter. The PolarFire transceiver is configured in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

Figure 10 • **PMA with Bit-slip Feature Block Diagram**

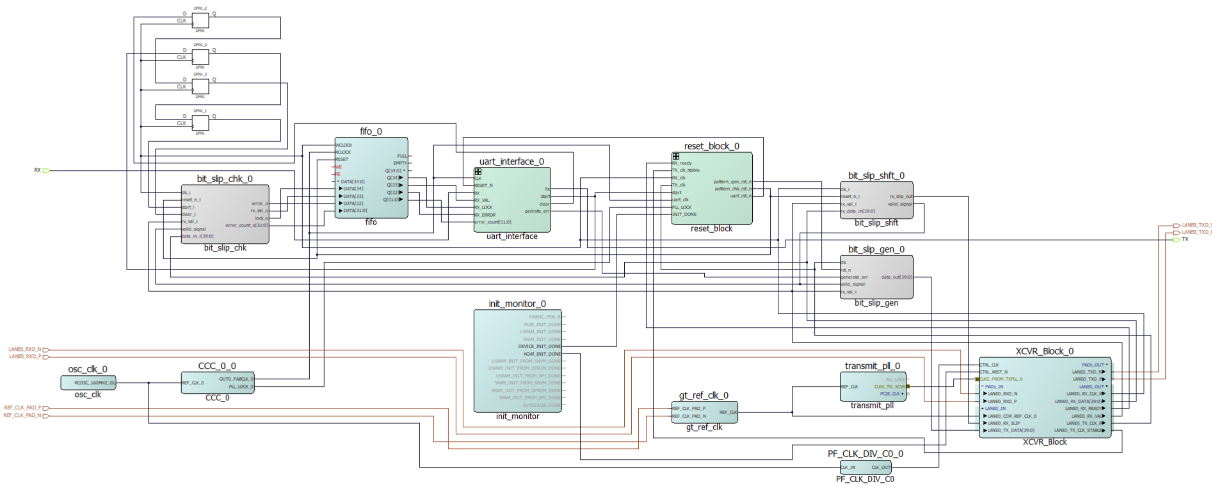


*: On-board loopback is used for Splash Kit.

2.6.1 Design Implementation

The following figure shows the Libero SoC 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 bit-slip mode with Enhanced Receiver management.

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 the UART interface. It receives 160 MHz input reference clock from on-board RC oscillator.

2.6.1.5 CDC_FIFO

CoreFIFO block synchronizes CDC 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 4 • **Port List for the PMA Design**

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

2.6.5 Simulating the Design

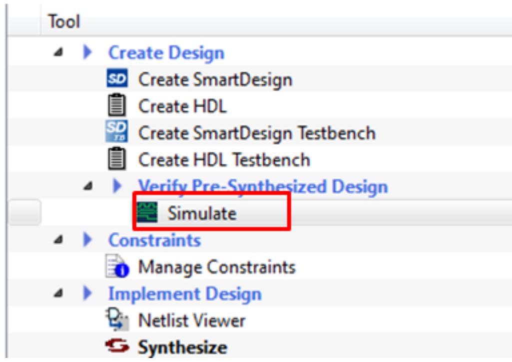
The pattern generator module generates a incremental counter pattern used to drive the transceiver in PMA mode. The pattern checker checks the received counter data and generates error flags if data mismatch occurs.

Before you begin:

1. Start Libero SoC.
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_Project` folder, select Libero_Project.prjx and click **Open**.
   The PolarFire transceiver project opens in Libero SoC.
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 > 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**

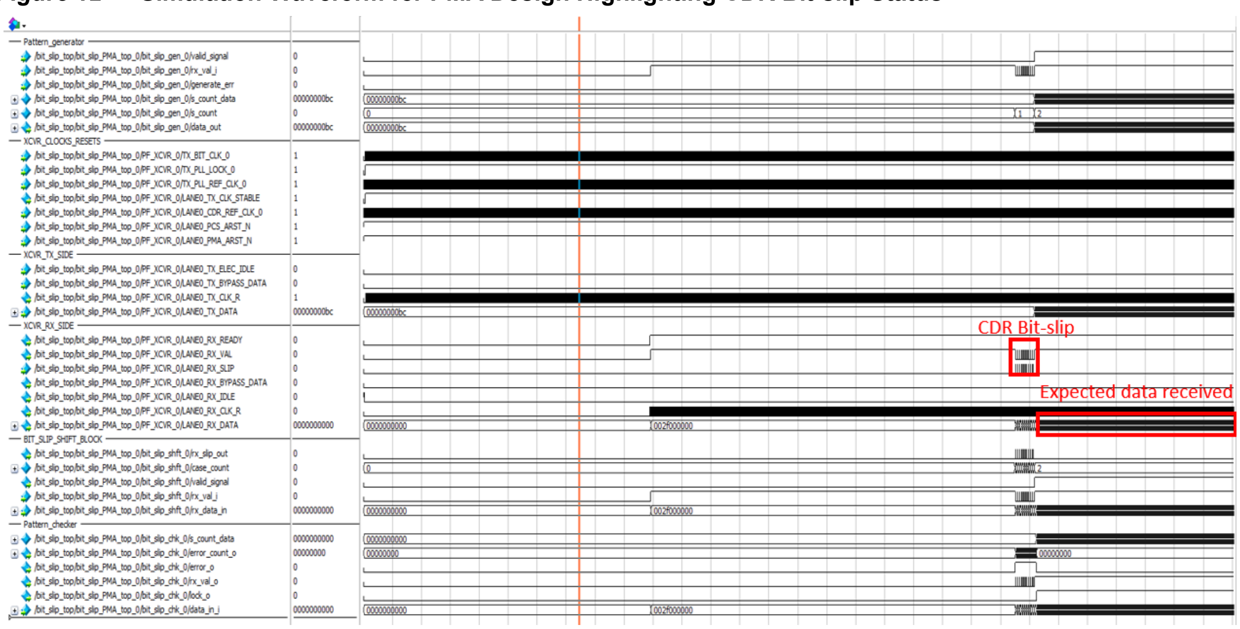
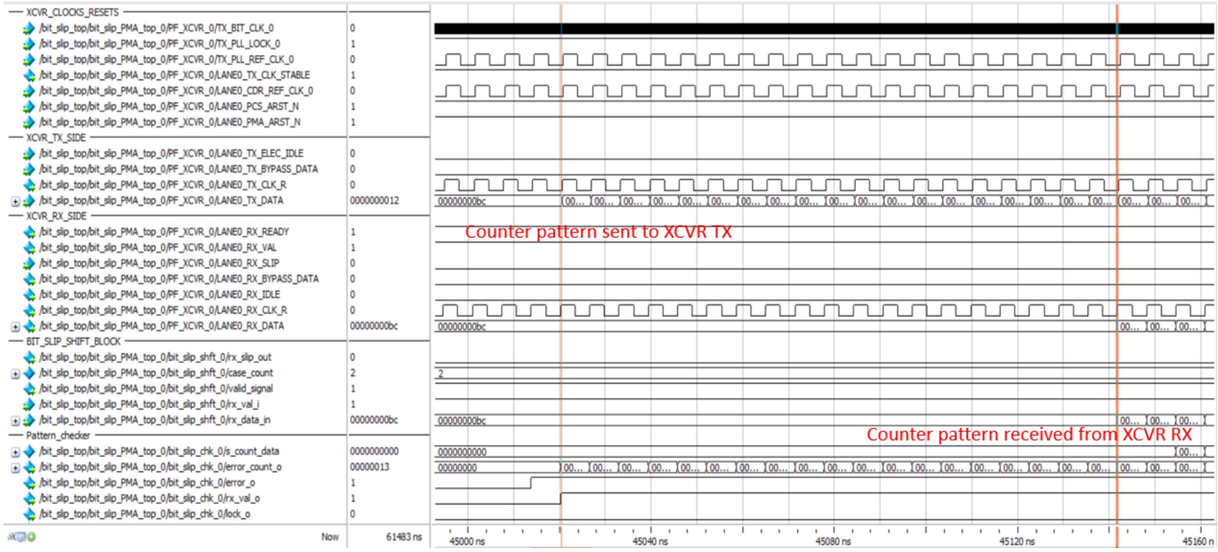


Figure 13 • **Simulation Waveform for PMA Design Highlighting Pattern Checker Lock**

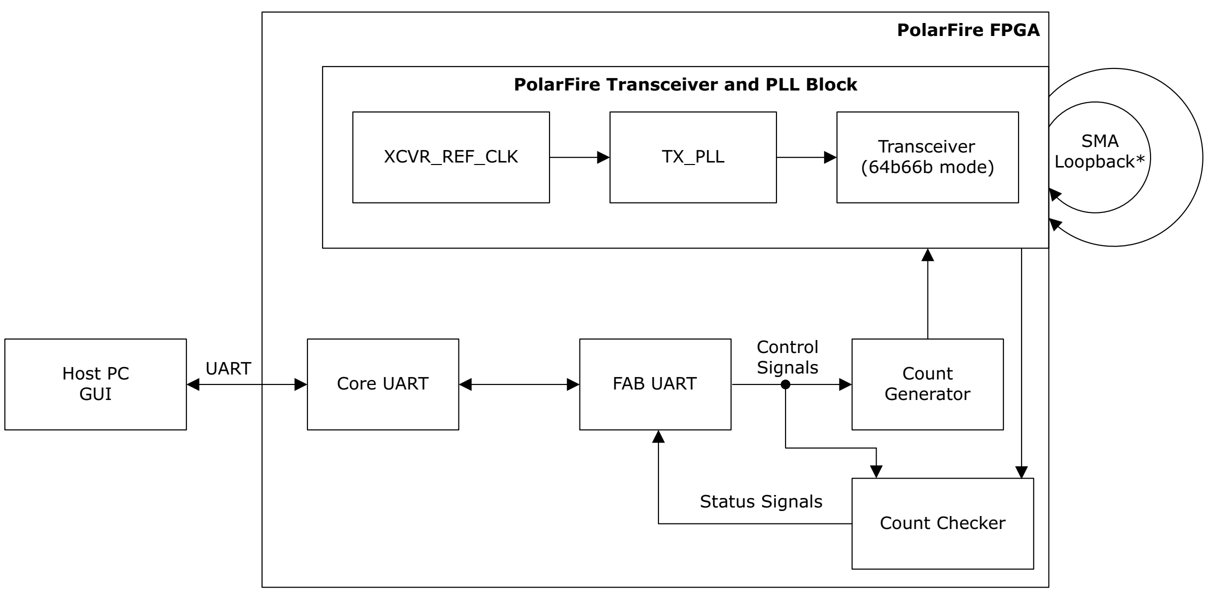


2.7 Reference Design 3: 64b66b Design

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.

Figure 14 • **64b66b Design Block Diagram**

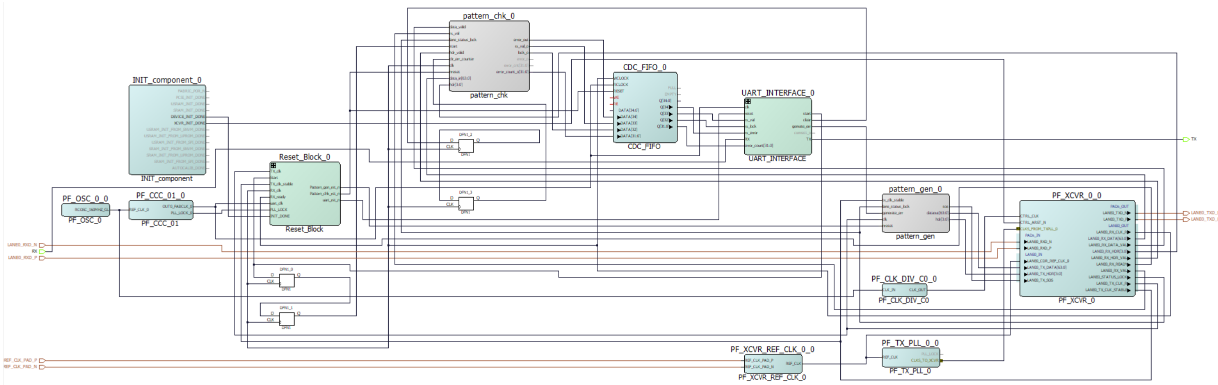


*: On-board loopback is used for Splash Kit.

2.7.1 Design Implementation

The following figure shows the Libero SoC 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 set to 10 Gbps, 64-bit PCS-Fabric interface width, and 64b66b mode with Enhanced Receiver management.

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**

Port	Direction	Description
LANE0_RXD_P	Input	Transceiver receiver differential input
LANE0_RXD_N	Input	Transceiver receiver differential input
REF_CLK_PAD_P	Input	Transmit PLL input clock from reference clock interface
REF_CLK_PAD_N	Input	Transmit PLL input clock from reference clock interface

Table 5 • **Port List for the 64b66b Design**

Port	Direction	Description
LANE0_PCS_ARST_N	Input	Asynchronous active-low reset for the PCS logic
LANE0_PMA_ARST_N	Input	Asynchronous active-low reset for the PMA logic
reset_n	Input	Active low reset for the fabric logic
clr_err_counter	Input	Error counter clear input
LANE0_TXD_P	Output	Transceiver transmitter differential output
LANE0_TXD_N	Output	Transceiver transmitter differential output
LANE0_STATUS_LOCK	Output	Indicates the lane status
LANE0_RX_VAL	Output	Receives data valid flag associated with a lane
error_out_o	Output	Error Flags
error_count_o[31:0]	Output	Error count Flags
Lock_o	Output	Data lock flag

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.

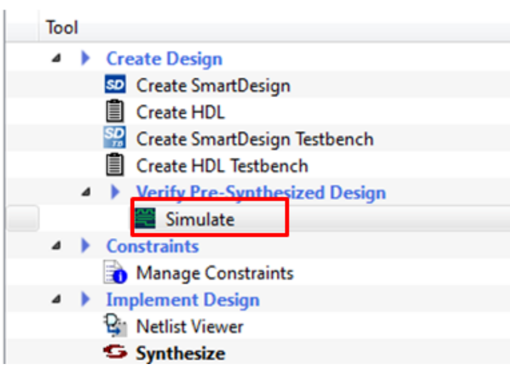
Before you begin:

1. Start Libero SoC.
2. In the Projects toolbar, click Open Project.
3. Browse the `PF_XCVR_64B66B` folder for the 64b66b design.
4. Navigate to the `Libero_Project` folder, select Libero_Project.prjx and click **Open**
   The PolarFire transceiver project opens in Libero SoC.
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 > 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**

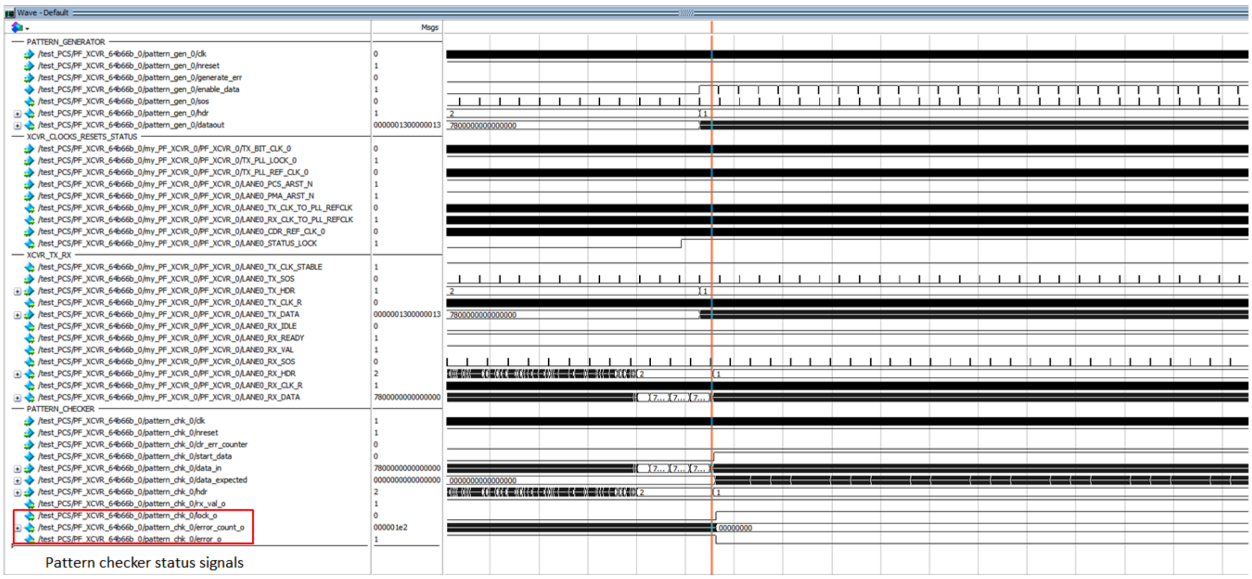
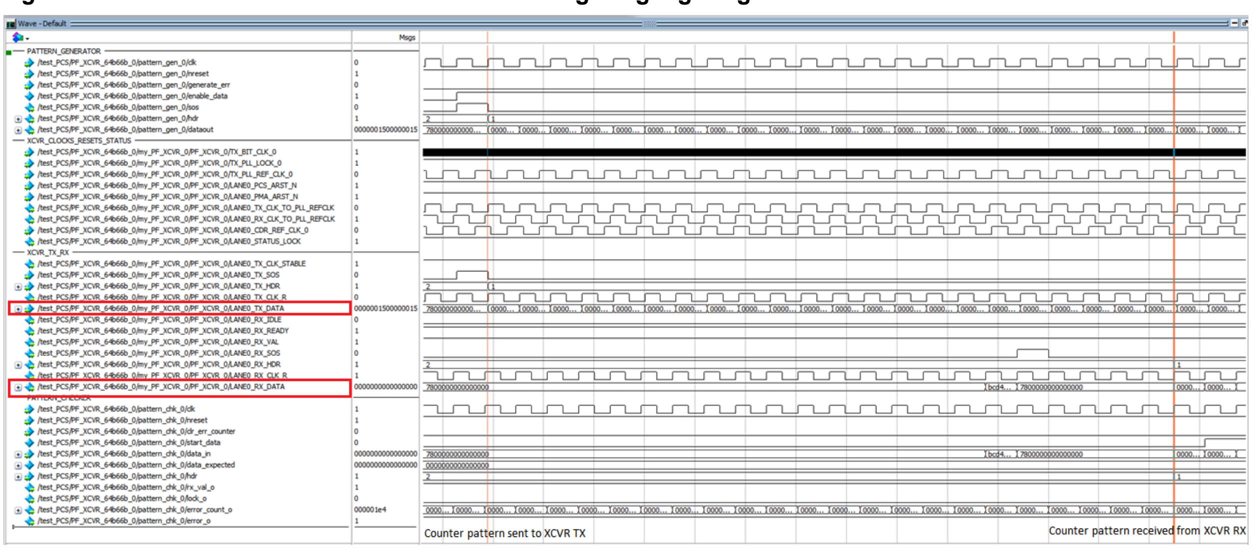


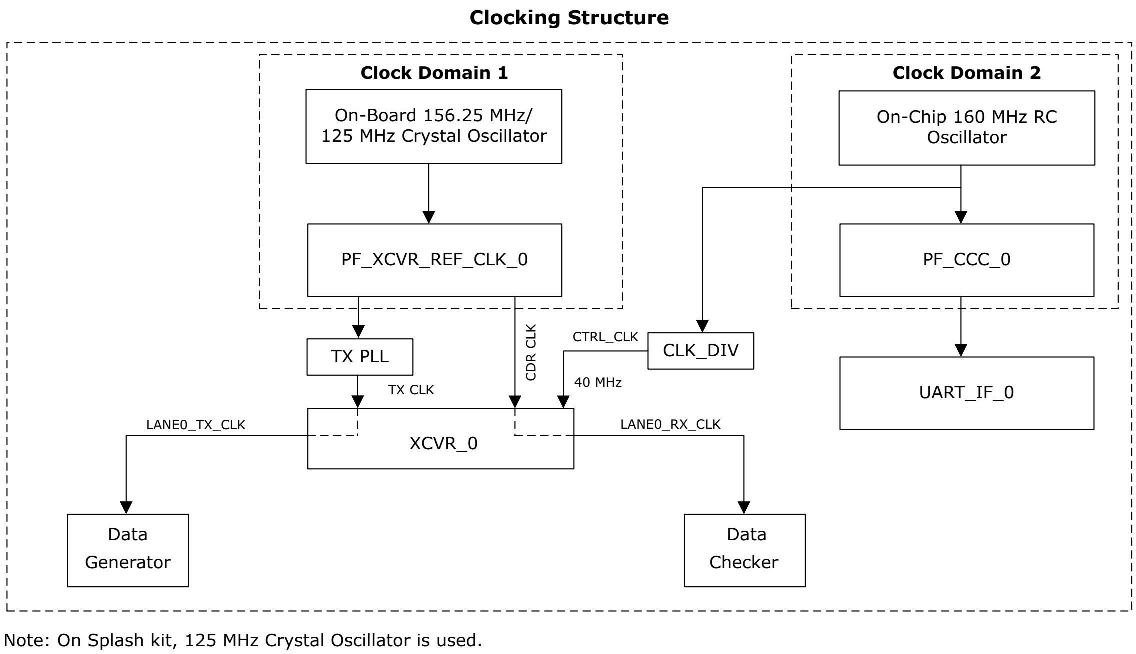
Figure 18 • **Simulation Waveform for 64b66b Design Highlighting Tx and Rx Data Match**



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.

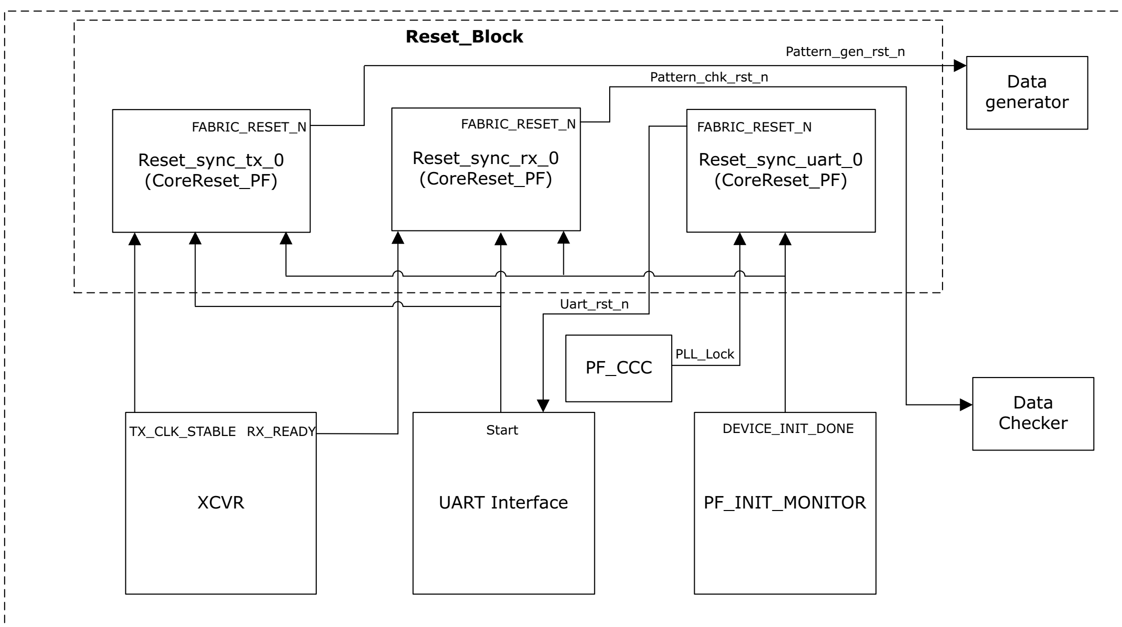
Reset_sync_uart_0 (CoreReset_PF) module releases active low reset of UART_INTERFACE when PLL_LOCK output from PF_CCC block and DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted.

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**



3 Libero Design Flow

This chapter describes the Libero design flow, which includes the following processes:

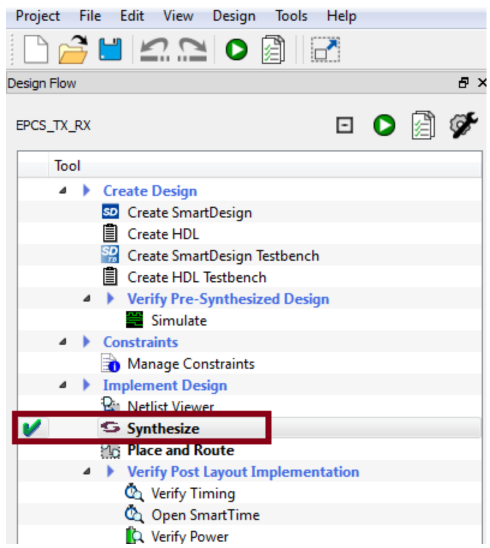
- Synthesize
- Place and Route
- Verify timing
- Design and Memory Initialization
- Generate Bitstream
- Run PROGRAM Action

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.

Figure 21 • **Synthesize**



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 6 • **8b10b Design Resource Utilization**

Type	Used	Total	Percentage
4LUT	633	299544	0.21
DFF	497	299544	0.17
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

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

Table 7 • **PMA Design Resource Utilization**

Type	Used	Total	Percentage
4LUT	692	299544	0.23
DFF	582	299544	0.19
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

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

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

Type	Used	Total	Percentage
4LUT	730	299544	0.24
DFF	804	299544	0.27
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

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**

Type	Used	Total	Percentage
4LUT	697	299544	0.23
DFF	535	299544	0.18
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

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

Table 10 • **SmartBert Design Resource Utilization**

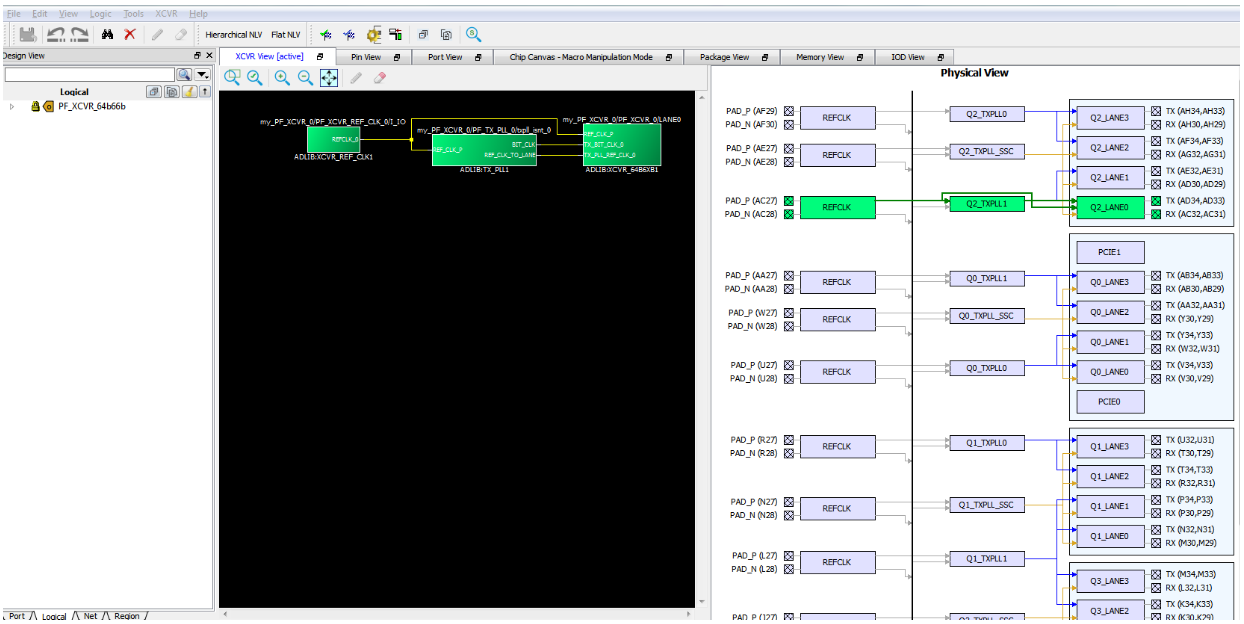
Type	Used	Total	Percentage
4LUT	5851	299544	1.95
DFF	1078	299544	0.36
Transceiver lanes	1	16	6.25
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09

3.3 Place and Route

3.3.1 XCVR Placement for Evaluation kit

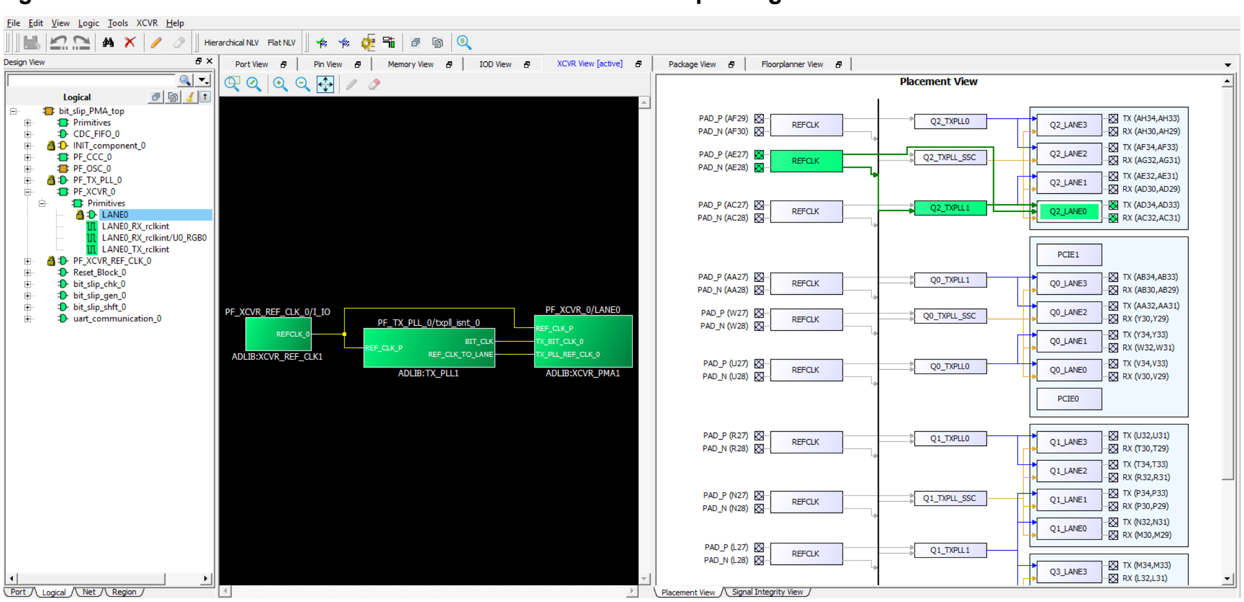
For 8b10b, Native PMA, 64b66b and SmartBert design, the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. Lane0 of Quad2 is used for on-board SMA loop-back of Tx and Rx. REFCLK is placed to use 156.25 MHz clock source for XCVR, as shown in following figure.

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.

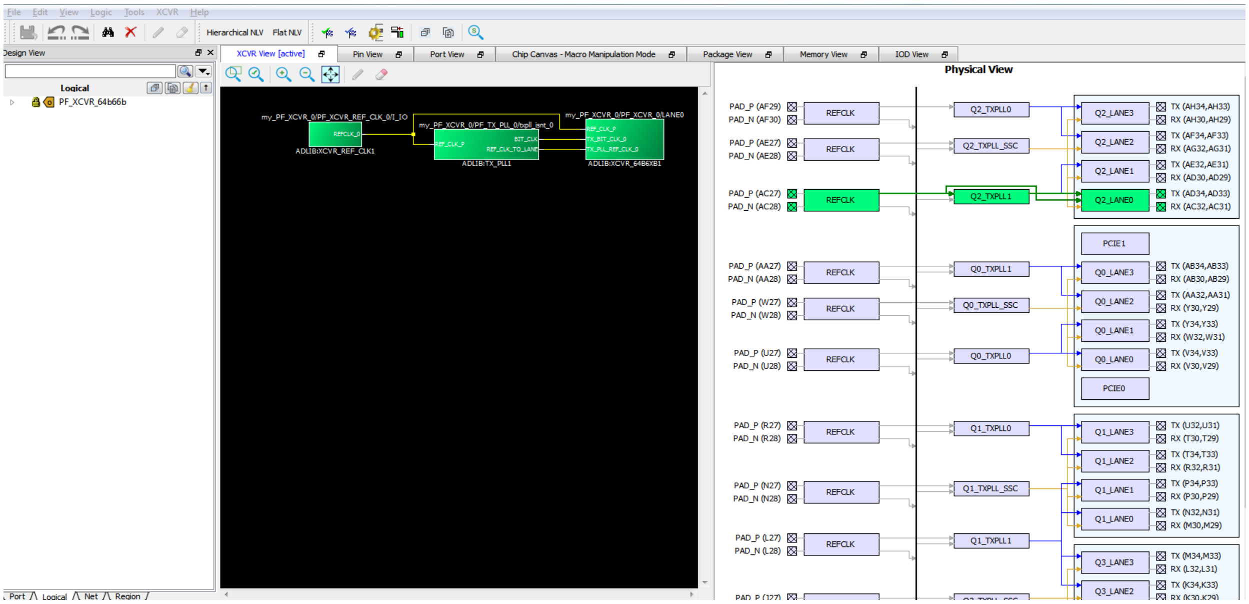
Figure 23 • **I/O Editor—Transceiver View for PMA with Bit-slip Design**



3.3.2 XCVR Placement for Splash Kit

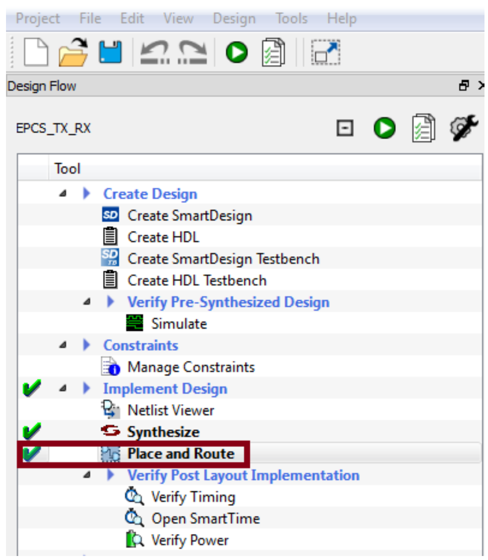
For 8b10b, Native PMA, PMA Bit-slip, 64b66b, and SmartBert design; the TX_PLL, XCVR_REF_CLK, and XCVR need to be constrained using the I/O Editor. Lane1 of Quad1 is used for on-board loopback of Tx and Rx. REFCLK is placed to use 125 MHz clock source for XCVR, as shown in following figure.

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.

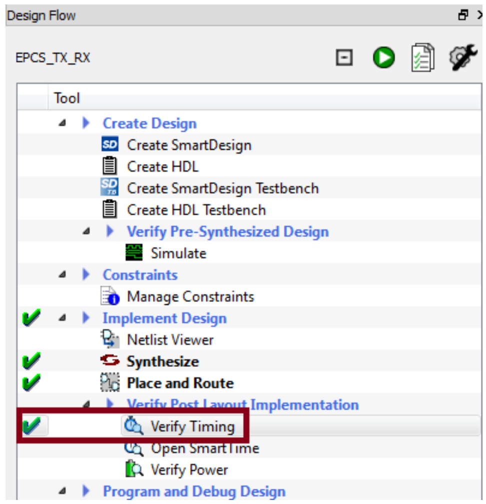
Figure 25 • **Place and Route**



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.

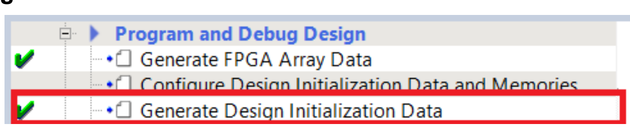
Figure 26 • **Design Flow**



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, perform the following steps:

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, on page 29.
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 27
- Programming the Device on the Splash Kit, page 28

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:

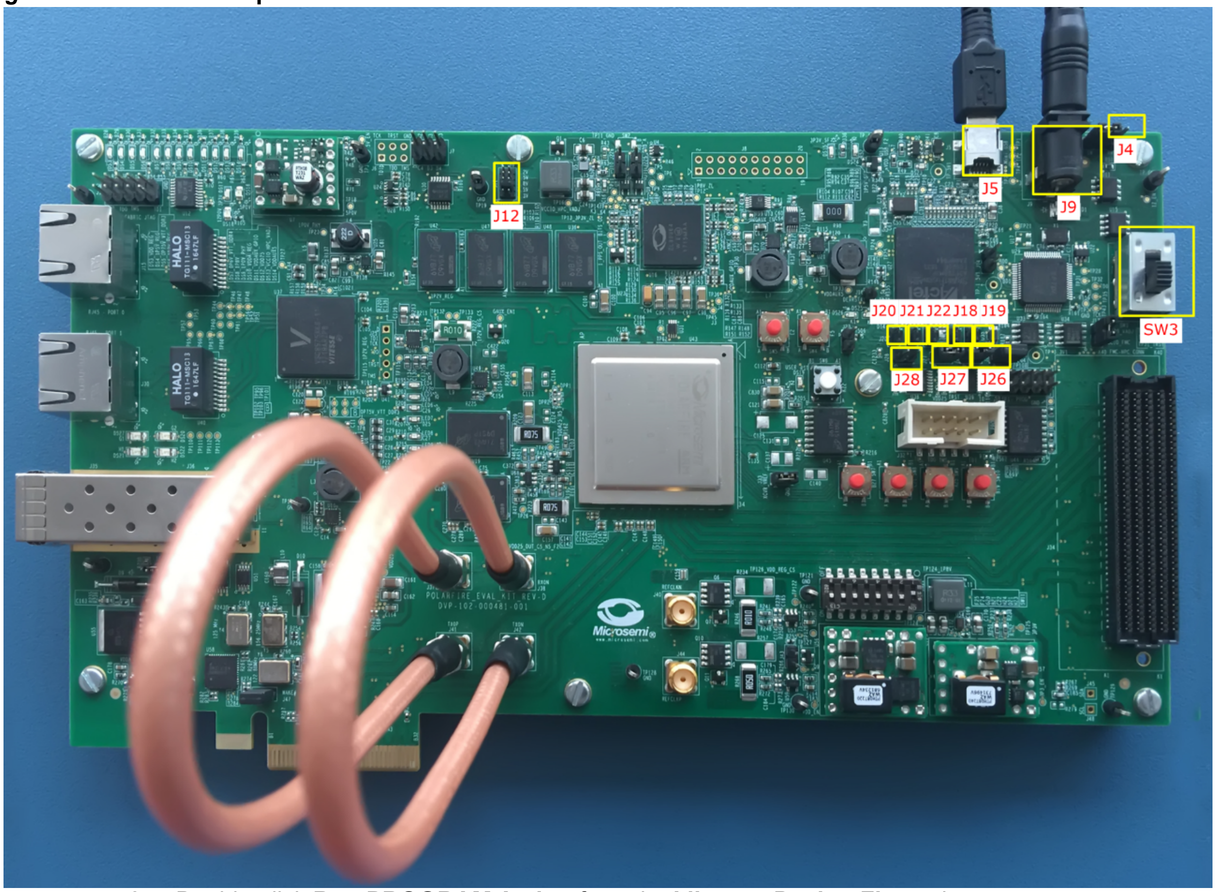
1. Ensure that the jumper settings on the board are same as listed in the following table.

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

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

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 two SMA to SMA cables as shown in the following figure. The following figure shows the board setup.

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



6. Double-click **Run PROGRAM Action** from the **Libero > Design Flow** tab.

   When the device is programmed successfully, a green tick mark appears as shown in the following figure. See Running the Demo, page 30 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 12 • **Jumper Settings on Splash Kit**

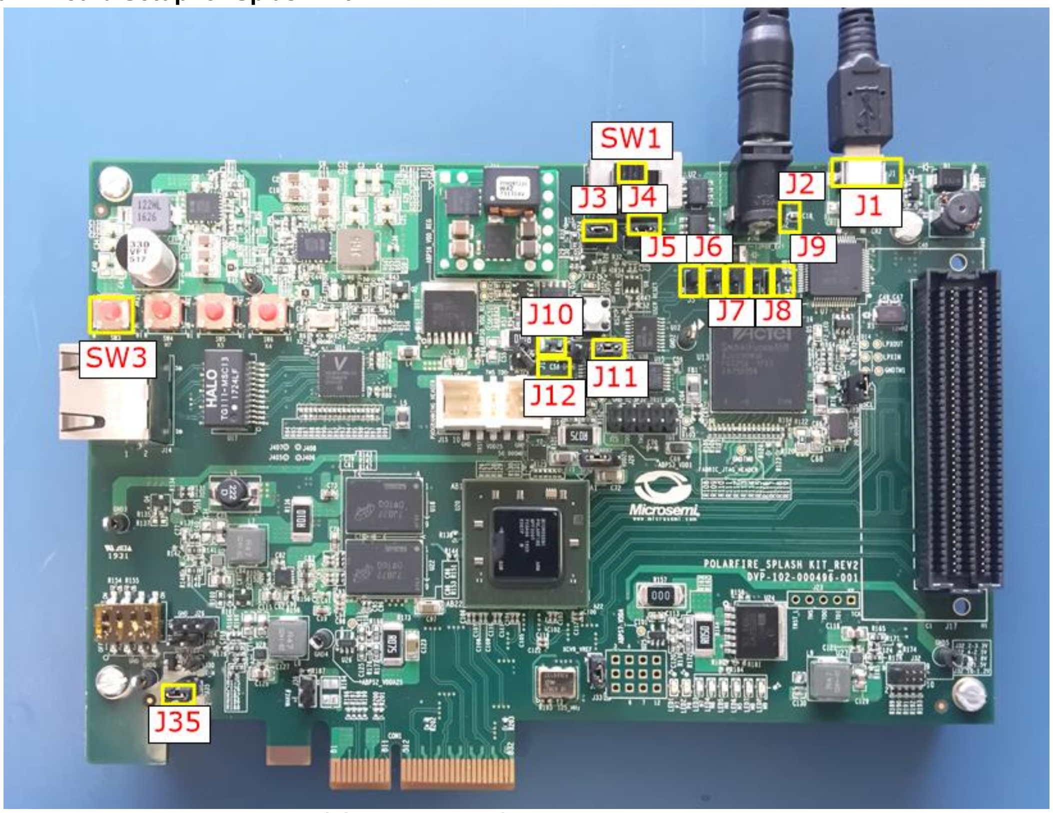
Jumper	Description
J5, J6, J7, J8, and J9	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI
J11	Close pin 1 and 2 for programming through FTDI chip
J10	Close pin 1 and 2 for programming through FTDI SPI
J4	Close pin 1 and 2 for manual power switching using SW1
J3	Open pin 1 and 2 for 1.0 V

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.

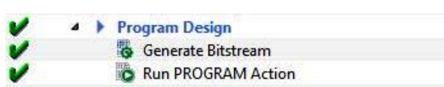
Figure 30 • **Board Setup for Splash Kit**



5. Double-click **Run PROGRAM Action** from the **Libero > Design Flow** tab.

When the device is programmed successfully, a green tick mark appears as shown in the following figure. See Running the Demo, page 30 to run the Multi-rate transceiver demo.

Figure 31 • **Programming the Device**



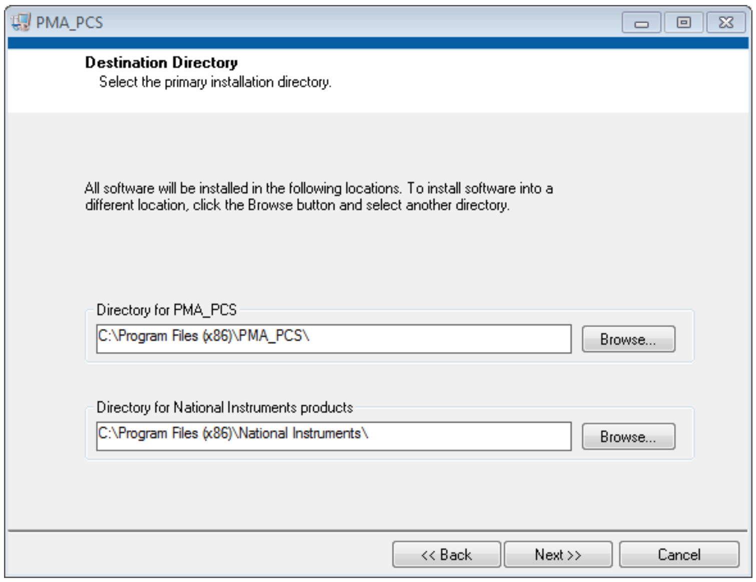
4 Running the Demo

This chapter describes how to install and use the GUI for selecting the test patterns and monitoring the loopback data.

Follow these steps:

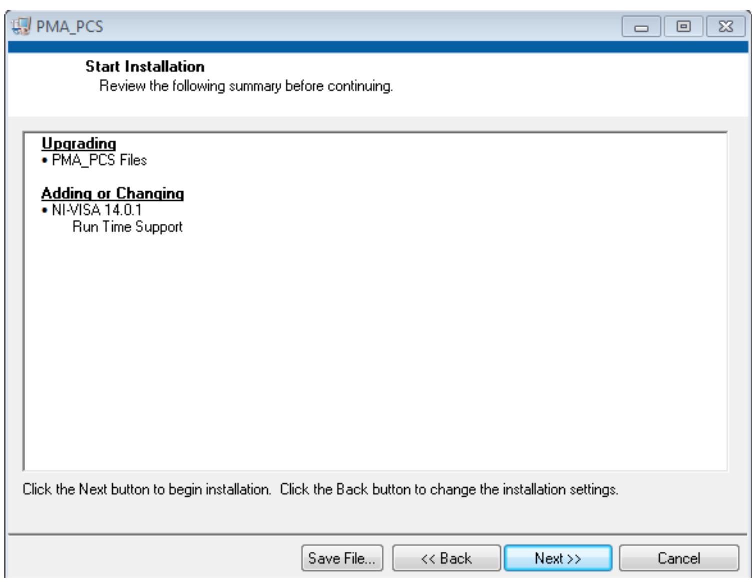
1. Install the **GUI_Installer** (setup.exe) from the following design files folder:
   `mpf_dg0759_eval\splash_df\GUI_Installer`
2. Apply default options as shown in the following figure.

Figure 32 • **Installing PMA_PCS Demo Application**



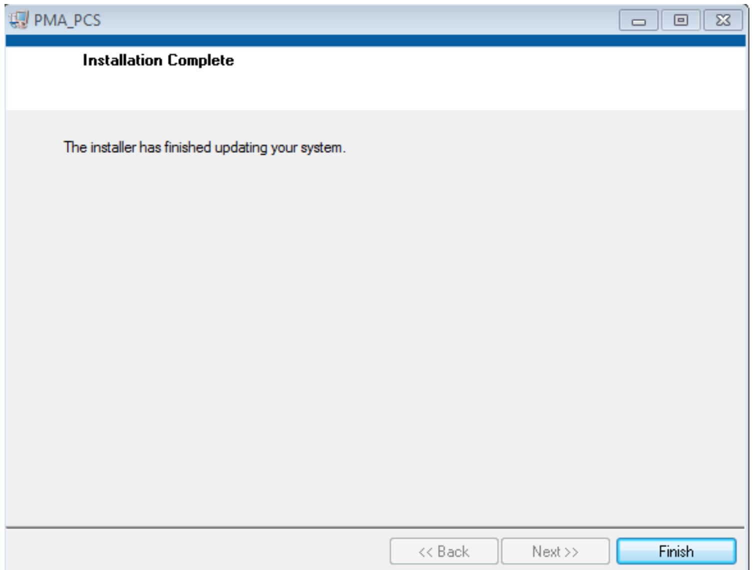
3. Click **Next.**

Figure 33 • **PMA_PCS Application Installation Steps**



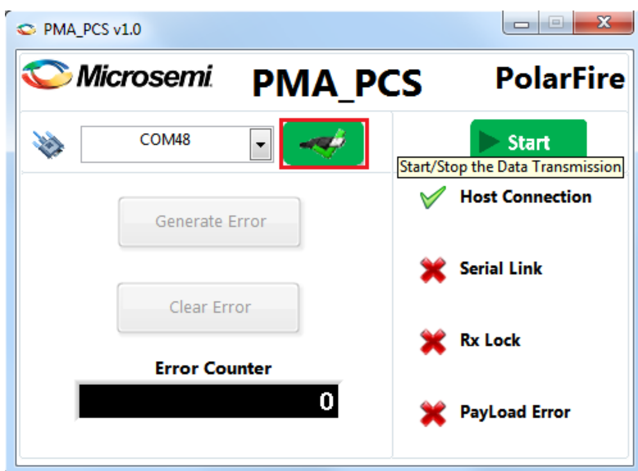
4. Click **Finish.**

Figure 34 • **Successful Installation of PMA_PCS Application**



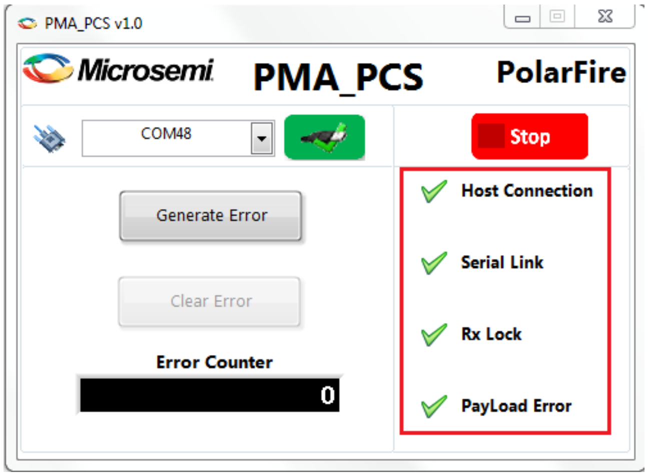
5. Go to **All Programs** > **PMA_PCS** > **PMA_PCS**. The PMA_PCS Demo window is displayed as shown in the following figure.
6. Select the COM port number that is detected to configure the serial port.
7. Click **Connect** to connect the GUI to the board through the selected port as shown in the following figure. After successfully connecting, the host connection status turns green as shown in the following figure.

Figure 35 • **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.

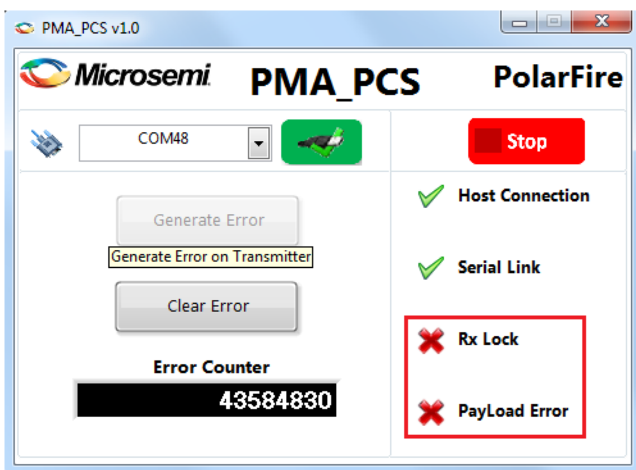
Figure 36 • **PMA PCS Status Signals**



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

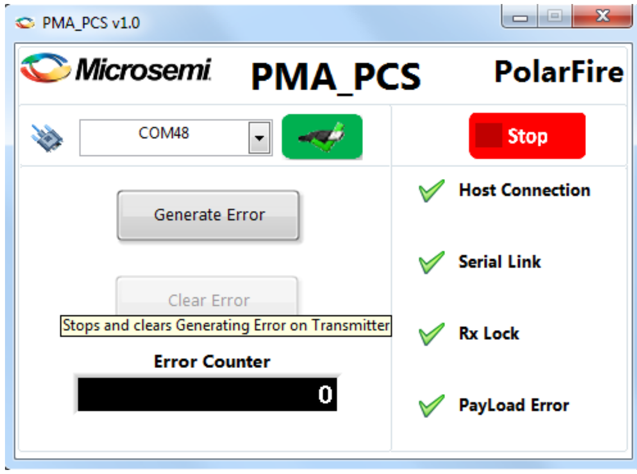
Note: Error counter displayed is a 32-bit counter running at a very high frequency clock. It keeps incrementing until injected error is cleared by clicking **Clear Error**.

Figure 37 • **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 38 • **Clear Data Error**



11. Click **Stop.**

The Multi-rate transceiver demo is successfully run.

5 Appendix 1: PolarFire Transceiver Overview

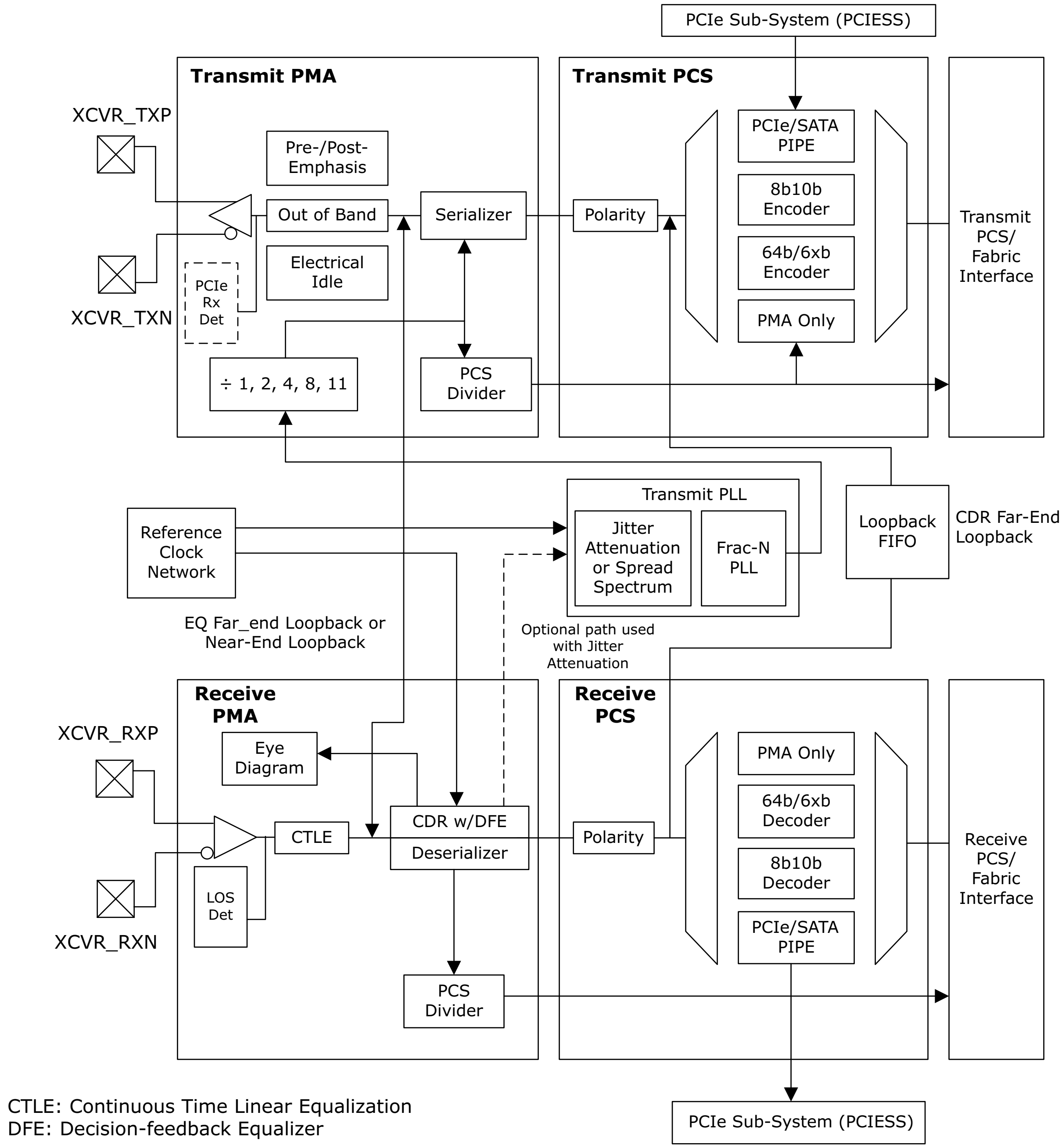
PolarFire FPGA transceivers include all required analog functions for high-speed data transmission between devices over Printed Circuit Boards (PCB) and high-quality cables. They are optimized for low-power operation and are suitable for a variety of device-to-device communication protocols.

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*.

Figure 39 • **PolarFire FPGA Transceiver**



The Microsemi Libero SoC design software supports configuring transceivers for various modes of operation. The Libero SoC 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.

6 Appendix 2: How to Use SmartBert IP

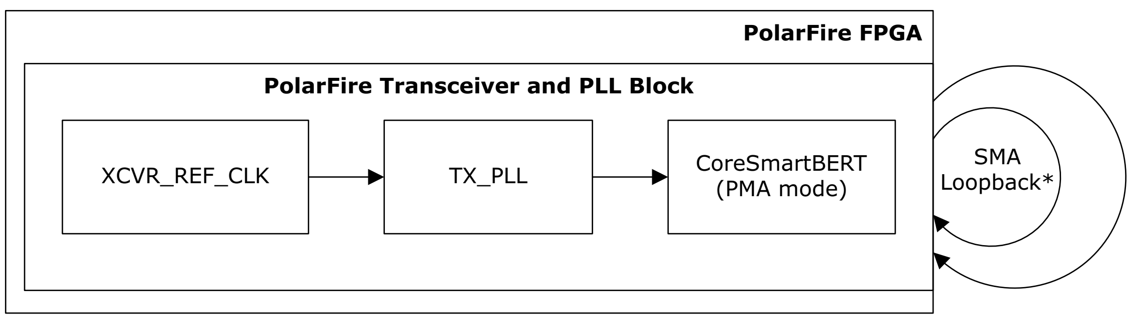
The CoreSmartBert core provides a demonstration platform for the PolarFire transceiver (PF_XCVR). It can be customized to use different line rates and reference clock rates. PRBS data pattern generators and checkers are included in the core. The pattern generator sends data out through the transmitter,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

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

6.1 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.

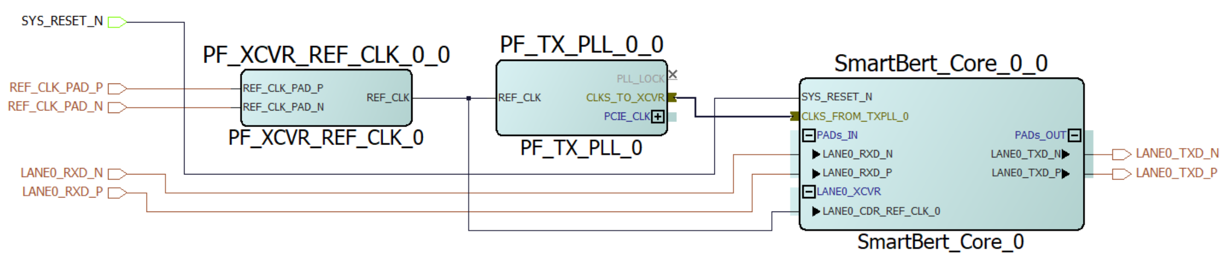
Figure 40 • **SmartBert Design Block Diagram**



6.1.1 Design Implementation

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

Figure 41 • **CoreSmartBert IP Design Implementation**



6.1.1.1 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.

6.1.1.2 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 networks in this design. The reference clock 0 is configured as a differential reference clock.

6.1.1.3 Transmit PLL

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

6.1.2 Port Description

The following table lists the important ports for the design.

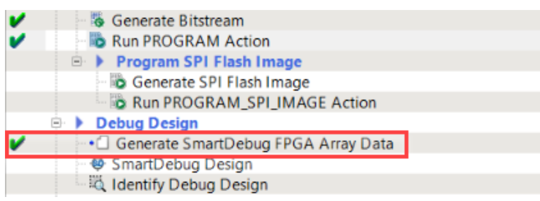
Table 13 • **Port List for the CoreSmartBert IP Design**

Port	Direction	Description
LANE0_RXD_P	Input	Transceiver Receiver differential input
LANE0_RXD_N	Input	Transceiver Receiver differential input
REF_CLK_PAD_P	Input	Transmit PLL input clock from reference clock interface
REF_CLK_PAD_N	Input	Transmit PLL input clock from reference clock interface
LANE0_TXD_P	Output	Transceiver Transmitter differential output
LANE0_TXD_N	Output	Transceiver Transmitter differential output

6.2 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.

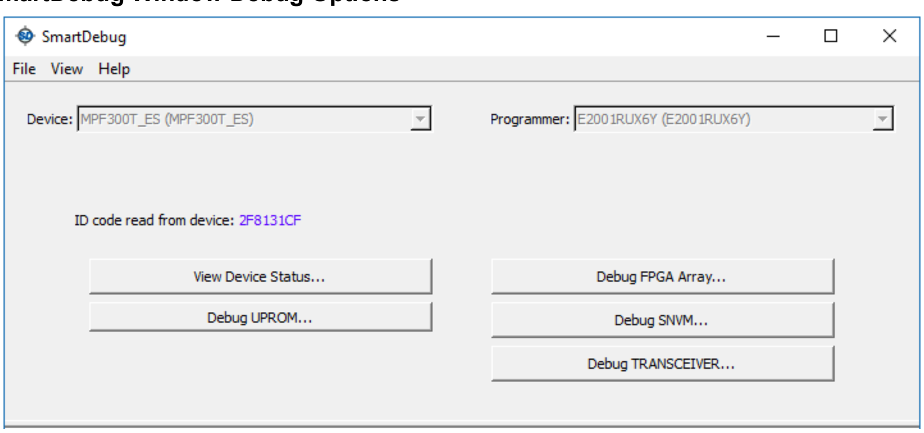
Figure 42 • **Launching SmartDebug Design Tools**



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

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

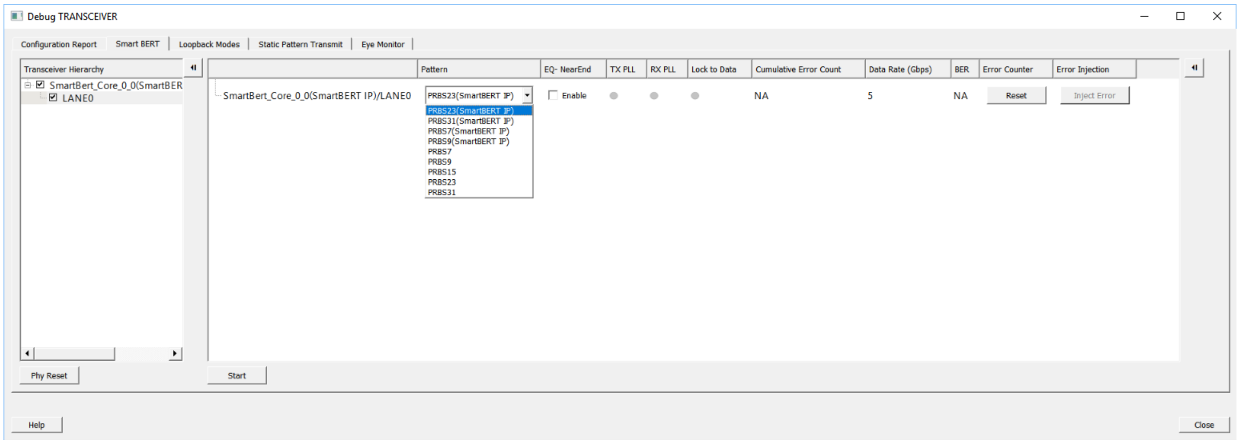
Figure 43 • **SmartDebug Window Debug Options**



To run SmartBert in Debug TRANSCEIVER, follow these steps:

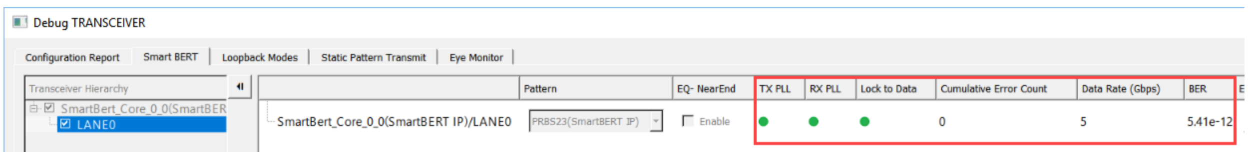
1. Select the **SmartBERT** tab in the **Debug TRANSCEIVER** window.
2. Select the **Pattern** from the drop-down list.

Figure 44 • **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 45 • **Debug TRANSCEIVER—Status**

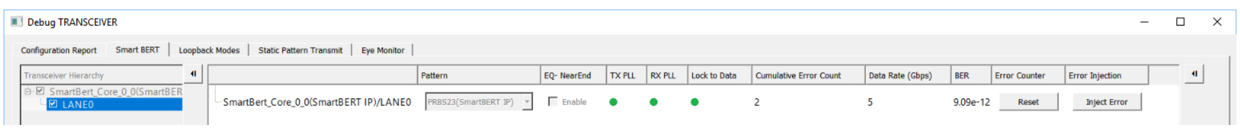


When a SmartBert IP lane is added, the **Error Injection** column is displayed in the right pane. The error injection feature is provided to inject an error while running a PRBS pattern.

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 46 • **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 5 Gbps transceiver data rate. Separate programming files (*.job) and corresponding design debug data container (DDC) for different transceiver data rates are exported from Libero and provided in the following locations:

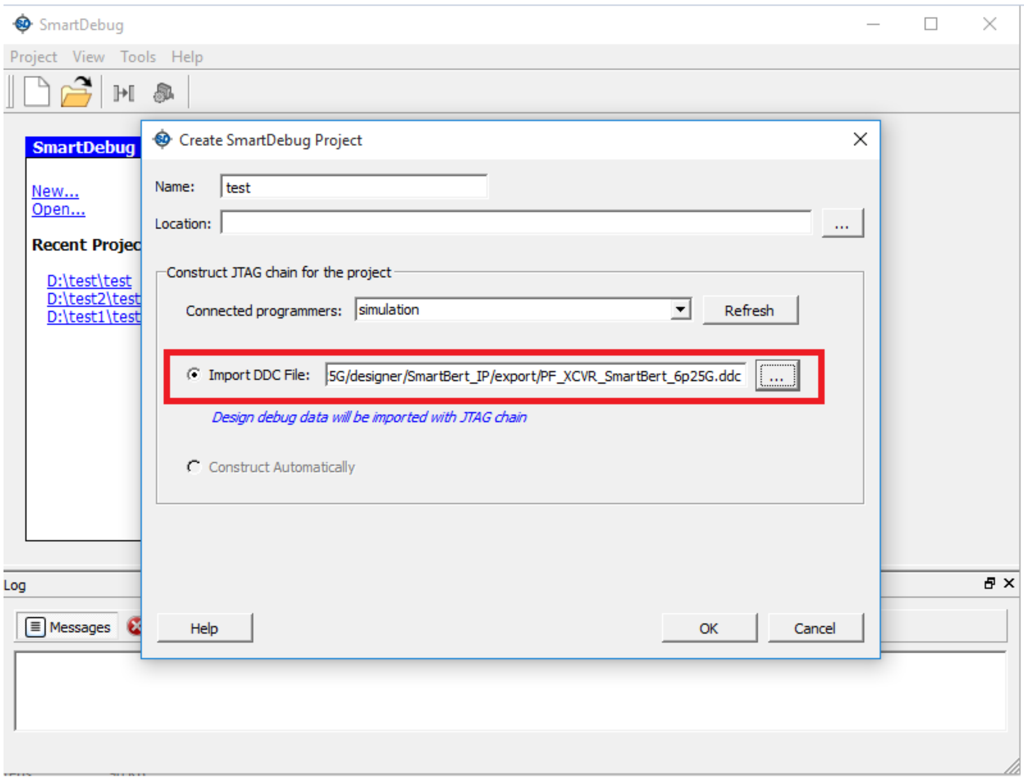
- *.job file: mpf_dg0759_eval/splash_df\PF_XCVR_SmartBert\Programming_Job
- *.ddc file: mpf_dg0759_eval/splash_df\PF_XCVR_SmartBert\Source_File

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 47 • **Create SmartDebug Project**



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

7 Appendix 3: Programming the Device Using FlashPro Express

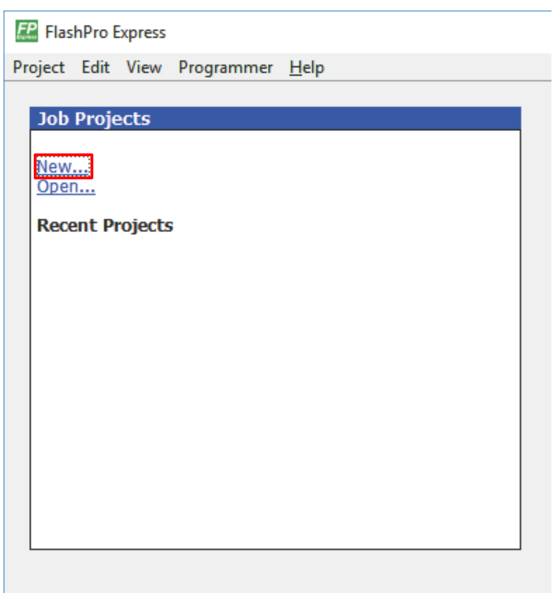
This section describes how to program the PolarFire device with the .job programming file using FlashPro Express. The .job file is available at the following design files folder location:

- **8b10b**: mpf_dg0759_eval\splash_df\PF_XCVR_8B10B\Programming_Job
- **64b66b**: mpf_dg0759_eval\splash_df\PF_XCVR_64B66B\Programming_Job
- **PMA**: mpf_dg0759_eval\splash_df\PF_XCVR_PMA\Programming_Job
- **PMA_WITH_BIT_SLIP**: mpf_dg0759_eval\splash_df\PF_XCVR_PMA_With_Bit_Slip\Programming_Job
- **SmartBert**: mpf_dg0759_eval\splash_df\PF_XCVR_SmartBert\Programming_Job

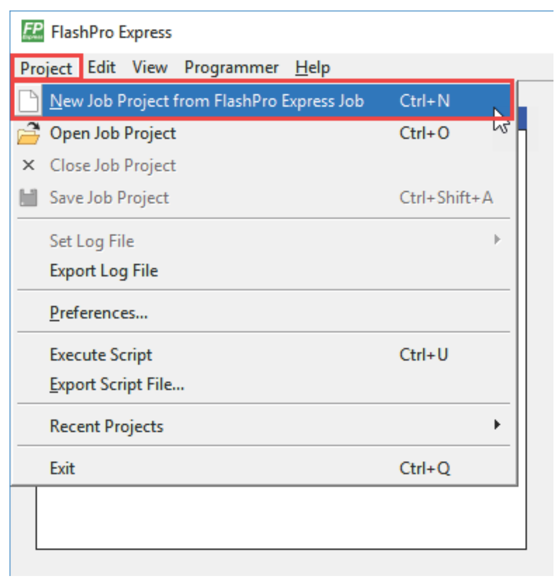
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are same as listed in Table 11, page 27 (for Evaluation board) and Table 12, page 28 (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 two SMA to SMA cables on the Evaluation board as shown in Figure 28, on page 28.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 48 • **FlashPro Express Job Project**

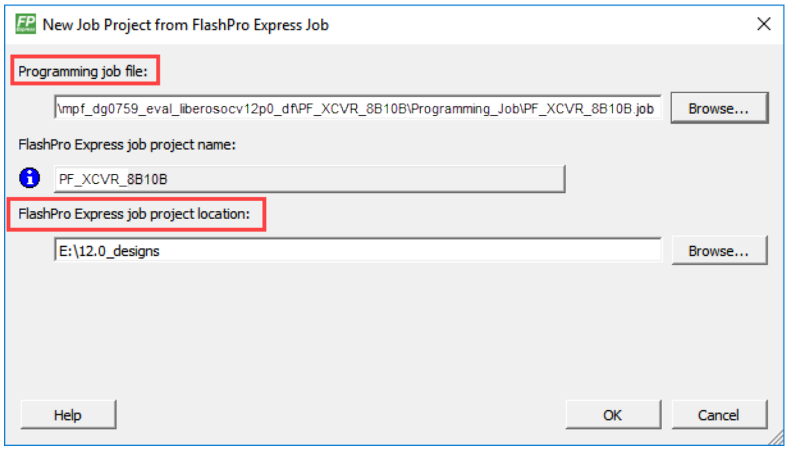


or



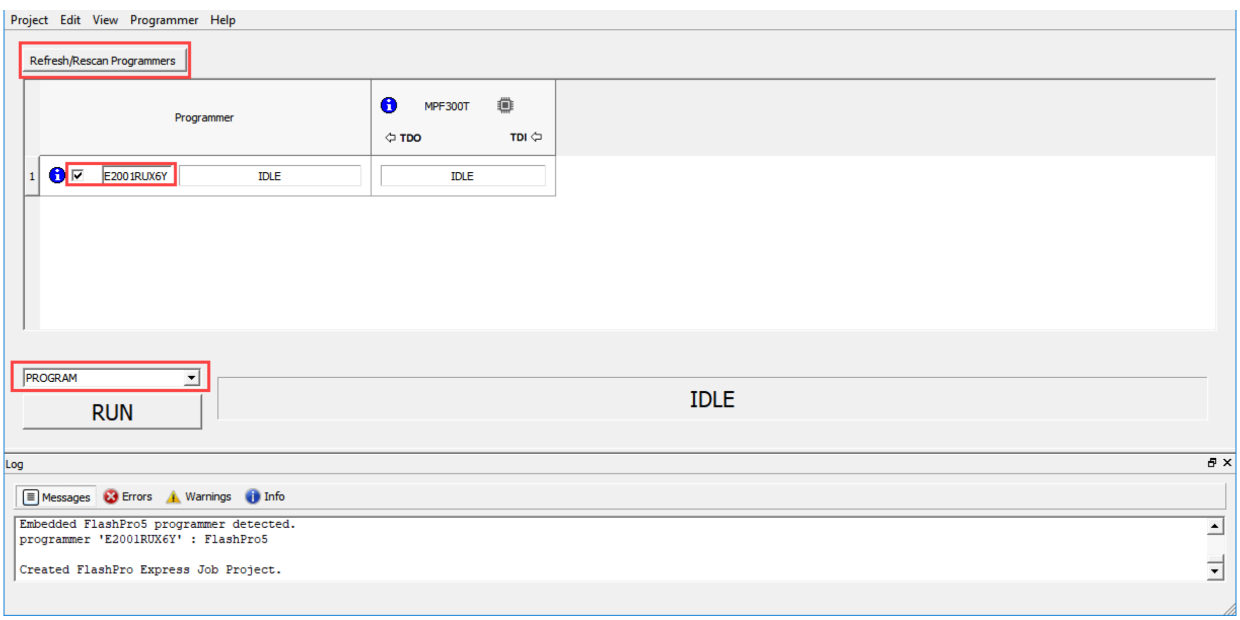
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
    - **Programming job file**: Click **Browse**, and navigate to the location where the `PF_XCVR_8B10B.job or PF_XCVR_64B66B.job or PF_XCVR_PMA_with_bit_slip.job or PF_XCVR_PMA.job PF_XCVR_SmartBert_5G.job` (for 5 Gbps data rate design) `or PF_XCVR_SmartBert_6P25G.job` (for 6.25 Gbps data rate design) `or PF_XCVR_SmartBert_10G.job` (for 10 Gbps data rate design) file is located and select the file.
    - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

Figure 49 • **New Job Project from FlashPro Express Job**



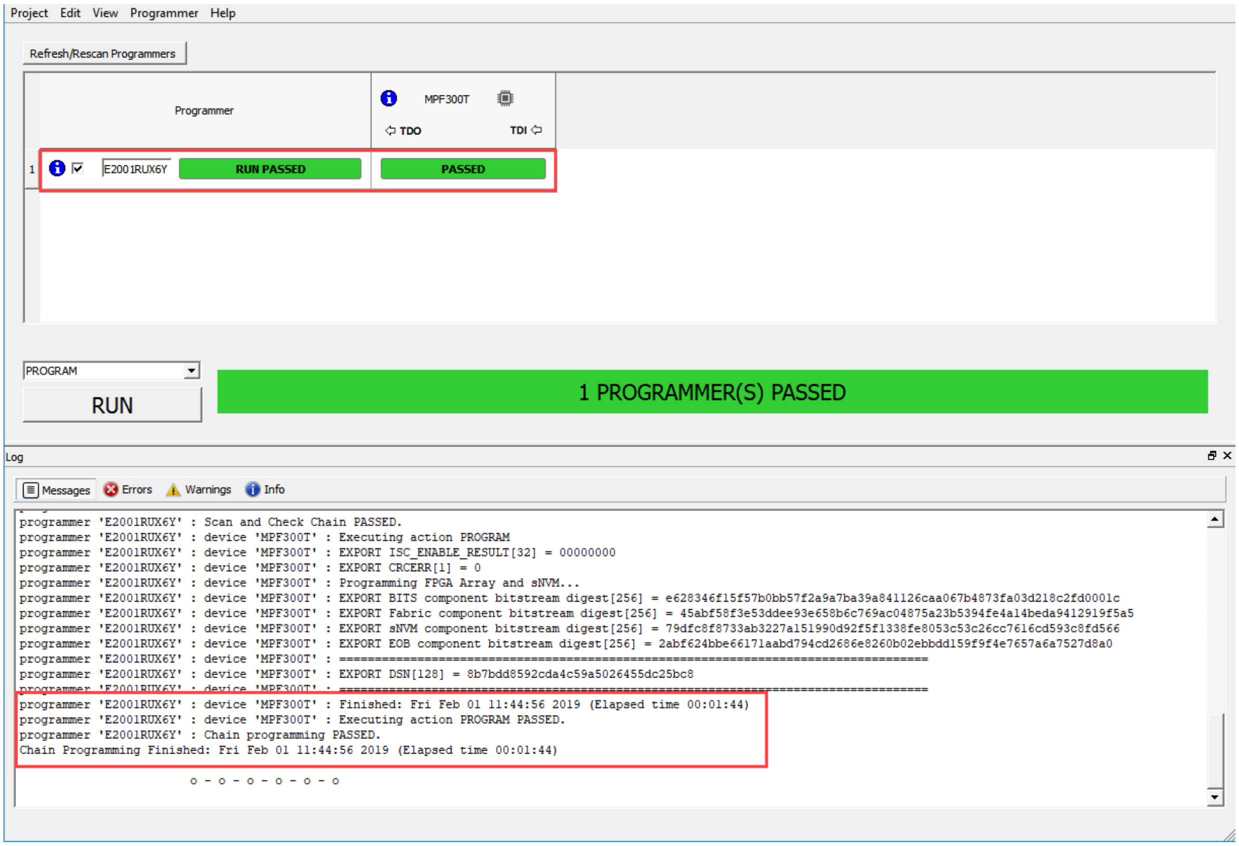
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**.

Figure 50 • **Programming the Device**



11. Click **RUN.** When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure. See Running the Demo, page 30 to run the demo.

Figure 51 • **FlashPro Express—RUN PASSED**



12. Close **FlashPro Express** or in the **Project** tab, click **Exit.**

8 Appendix 4: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to:

- **mpf_dg0759_eval_df/TCL_Scripts/readme.txt.**
- **mpf_dg0759_splash_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.

9 Appendix 5: 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*.