PolarFire FPGA Transceiver Application Note

AN4662

Introduction (Ask a Question)

PolarFire® FPGA family includes embedded low-power, and 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 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.

Table of Contents

1. PolarFire FPGA Transceiver Demo (Ask a Question)

This document presents five designs that demonstrate the use of PolarFire® transceivers in 8b10b, 64b66b, PMA modes, PMA bit slip feature, and use of SmartBERT IP The current version of this document contains designs that provide Libero® design flow examples, HDL simulation, and transceiver validation on a PolarFire Evaluation board. These reference designs show 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 with the design files, see Appendix 3: Programming the Device Using FlashPro Express.
- Using Libero SoC: To program the device using Libero SoC, see Programming the Device on the Evaluation Kit. Use this option when the reference design is modified.

Important: To evaluate the performance and features of the transceiver to the design requirements, use the reference designs.

1.1 Design Requirements (Ask a Question)

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

Table 1-1. Design Requirements

Requirement	Version
Operating System	64-bit Windows® 10 and 11
Hardware	
PolarFire® Evaluation Kit (MPF300TS-EVAL-KIT)	Rev D or later
2 SMA-to-SMA cables with 10 Gbps support (not provided with the kit)	—
Software	
Libero® SoC [1]	See the `readme.txt` file provided in the design files for the software versions used with this reference design.
Microchip FPGA_GUI_Pack[2]	Version 1.0

Important:

1. Libero® SmartDesign and configuration screenshots shown in this document are for illustration purpose only. Open the Libero design to see the latest updates.
2. Microchip FPGA_GUI_Pack includes the necessary run-time engine and drivers for the operation of Microchip FPGA Demo GUI applications.

1.2 Prerequisites (Ask a Question)

Before you begin, perform the following steps:

1. Download the demo design files from the following location: www.microchip.com/en-us/application-notes/AN4662
2. Download and install Libero SoC on the host PC from the following location: Libero SoC Documentation

3. Install the PMA_PCS GUI application by running the `setup.exe` file available in the design files folder: `<$Design_Files_Directory>\mpf_an4662_df\GUI\GUI_Installer`
   At the end of the installation, you may be prompted to download and install the FPGA_GUI_Pack, if it is not already available on your system.
4. Alternatively, you can manually download and install the Microchip FPGA_GUI_Pack from the following location: ww1.microchip.com/downloads/aemDocuments/documents/FPGA/SOCDesignFiles/FPGA_GUI_Pack_Installer.zip.

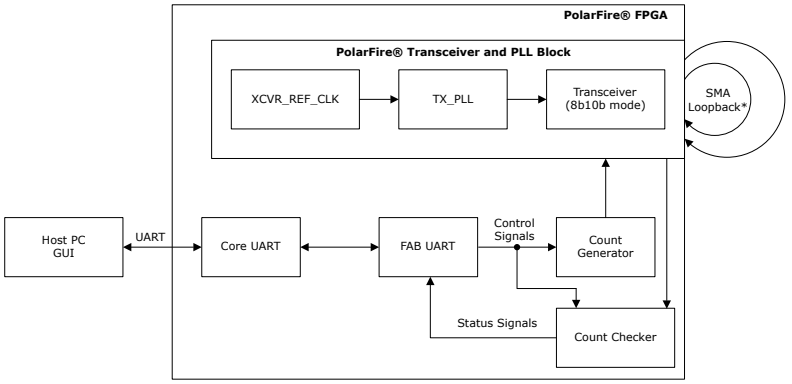
1.3 Demo Designs (Ask a Question)

This document presents five designs: PMA, PMA bitslip, 8b10b, 64b66b, and SmartBERT IP reference design. All five designs are located under `mpf_an4662_df/HW`, and the programming files are located under the `mpf_an4662_df/Programming_files` folder.

1.4 Reference Design 1: 8b10b (Ask a Question)

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 figure shows the design block diagram.

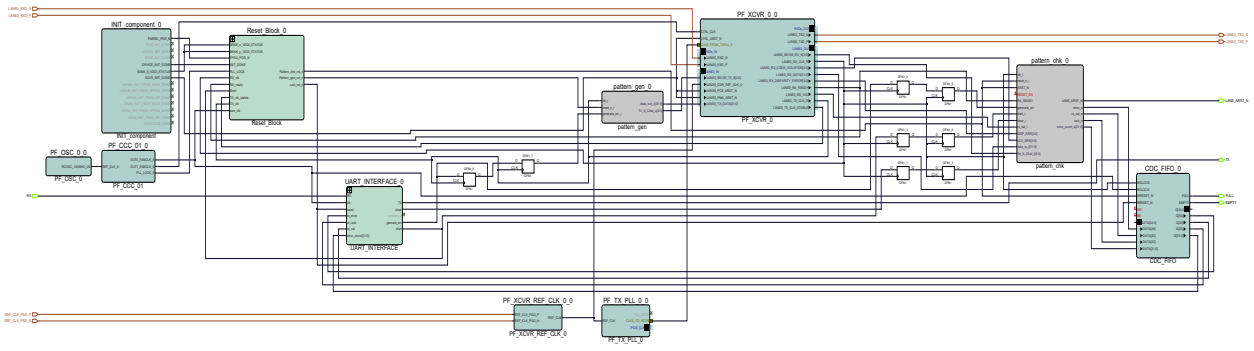
Figure 1-1. 8b10b Design Block Diagram



1.4.1 Design Implementation (Ask a Question)

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

Figure 1-2. 8b10b Design Implementation



1.4.1.1 PolarFire Transceiver Configurator (Ask a Question)

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

1.4.1.2 PolarFire Transceiver Reference Clock (Ask a Question)

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.

1.4.1.3 Transmit PLL (Ask a Question)

The transmit PLLs reference clock and 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. Therefore, the clock runs at half the data rate, consuming less dynamic power.

1.4.1.4 Clock Conditioning Circuitry (Ask a Question)

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

1.4.1.5 CDC_FIFO (Ask a Question)

CoreFIFO block synchronizes Clock Domain Crossing (CDC) data signals between fabric and UART interface.

1.4.2 Pattern Generator and Checker (Ask a Question)

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.

1.4.3 Port Description (Ask a Question)

The following table lists the important ports for the design.

Table 1-2. 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

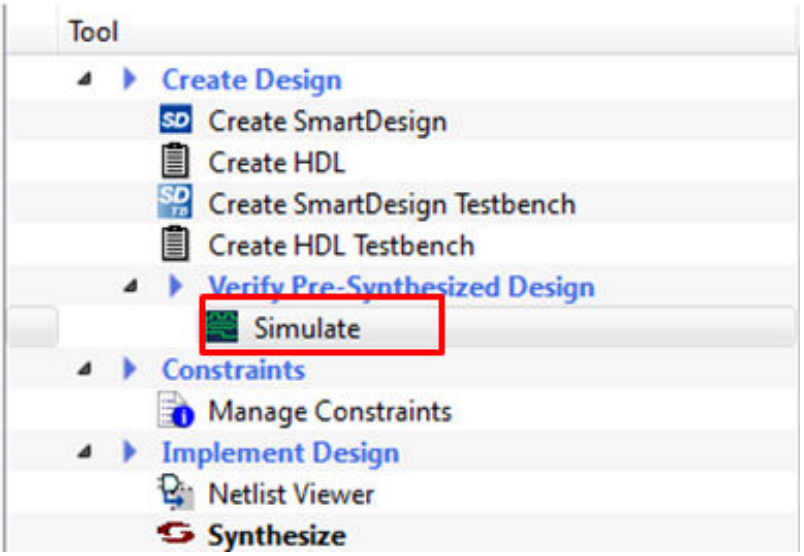
1.4.4 Simulating the Design (Ask a Question)

For simulation, perform the following the steps:

1. Open Libero® SoC design built through the TCL scripts.
2. To run the simulation, navigate to **Stimulus Hierarchy**, right-click on the `top.v` file, and select **Set as active stimulus**.

3. In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design, as shown in the following figure.

Figure 1-3. Simulating the 8b10b Design



1.4.5 Simulation Flow (Ask a Question)

The following steps describe the simulation flow:

1. Initially, the transceiver is 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 checks 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 show the simulation waveform for the 8b10b design highlighting pattern checker status signals and Tx/Rx data match. The simulation run time will be 58 us approximately.

Figure 1-4. Simulation Waveform for 8b10b Design Highlighting Pattern Checker Status

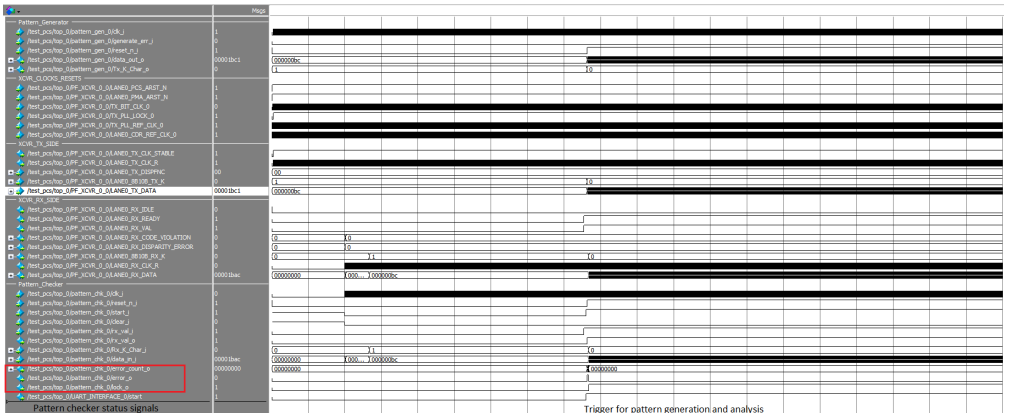
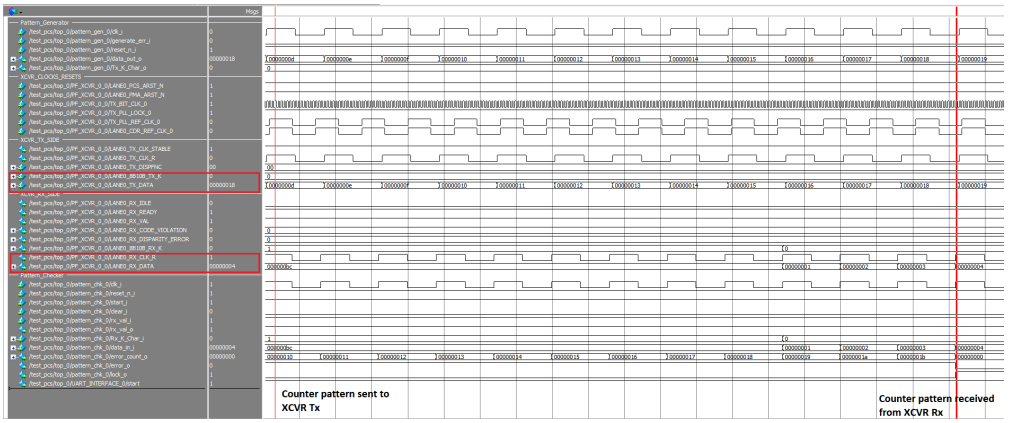


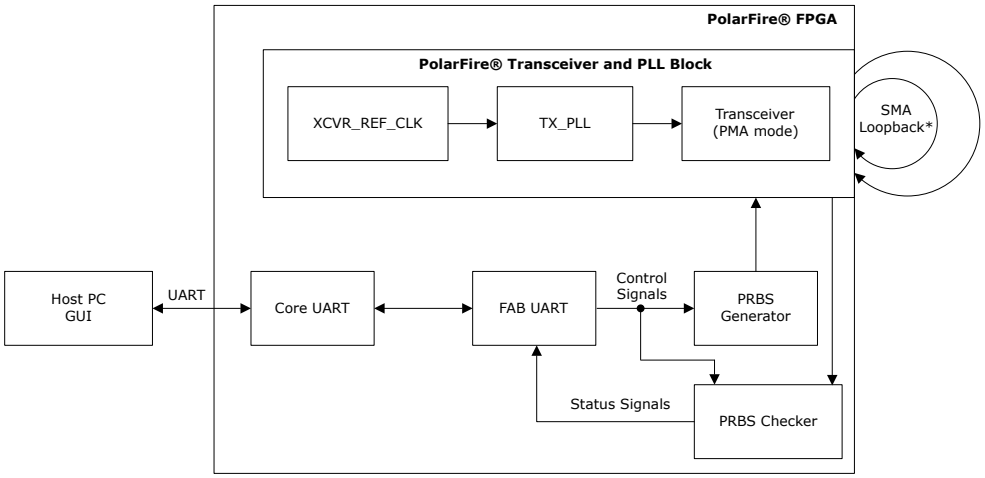
Figure 1-5. Simulation Waveform for 8b10b Design Highlighting Tx and Rx Data Match



1.5 Reference Design 2: PMA Design (Ask a Question)

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 of the PMA design.

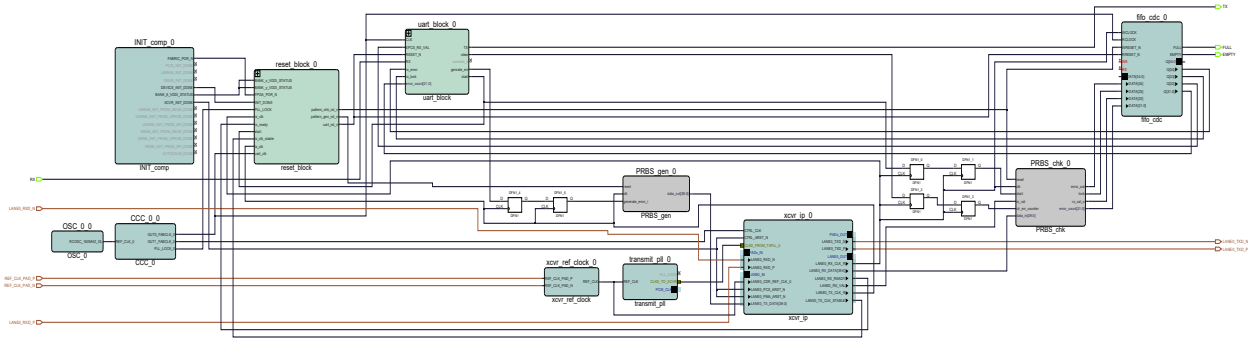
Figure 1-6. PMA Design Block Diagram



1.5.1 Design Implementation (Ask a Question)

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

Figure 1-7. PMA Design Implementation



1.5.1.1 PolarFire Transceiver Configurator (Ask a Question)

The PolarFire Transceiver Interface configurator is set to 5 Gbps, 40-bit PCS-Fabric interface width, and PMA mode with Enhanced Receiver management.

1.5.1.2 PolarFire Transceiver Reference Clock (Ask a Question)

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.

1.5.1.3 Transmit PLL (Ask a Question)

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

1.5.1.4 Clock Conditioning Circuitry (Ask a Question)

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

1.5.1.5 CDC_FIFO (Ask a Question)

CoreFIFO block synchronizes CDC data signals between fabric and UART interface.

1.5.2 PRBS Generator and Checker (Ask a Question)

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.

1.5.3 Port Description (Ask a Question)

The following table lists the important ports for the design.

Table 1-3. 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

Table 1-3. Port List for the PMA Design (continued)

Port	Direction	Description
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

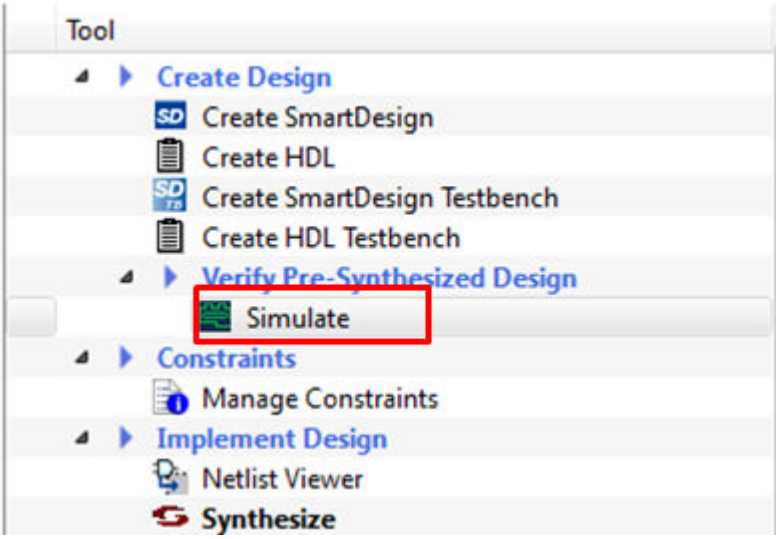
1.5.4 Simulating the Design (Ask a Question)

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.

For simulation, perform the following the steps:

1. Open Libero SoC design built through the TCL scripts.
2. To run the simulation, navigate to **Stimulus Hierarchy**, right-click the `top.v` file, and select **Set as active stimulus**.
3. In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design, as shown in the following figure.

Figure 1-8. Simulating the PMA Design



1.5.5 Simulation Flow (Ask a Question)

The following steps describe the simulation flow:

1. Initially, the transceiver is 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. The simulation run time will be 79 us approximately.

Figure 1-9. Simulation Waveform for PMA Design Highlighting PRBS Checker Status

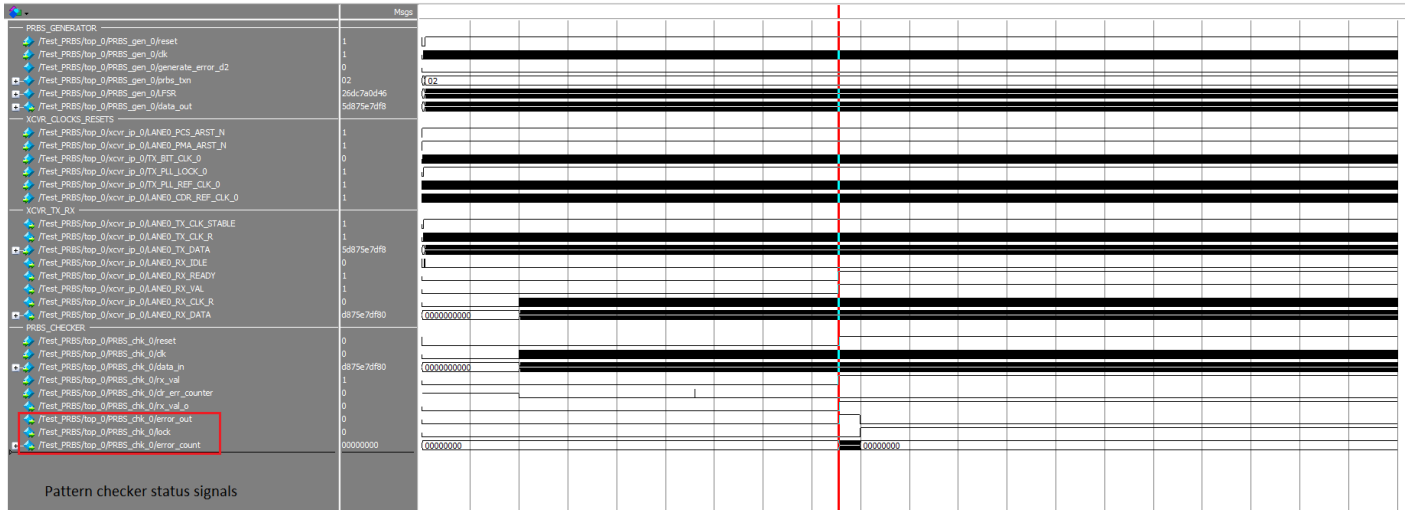
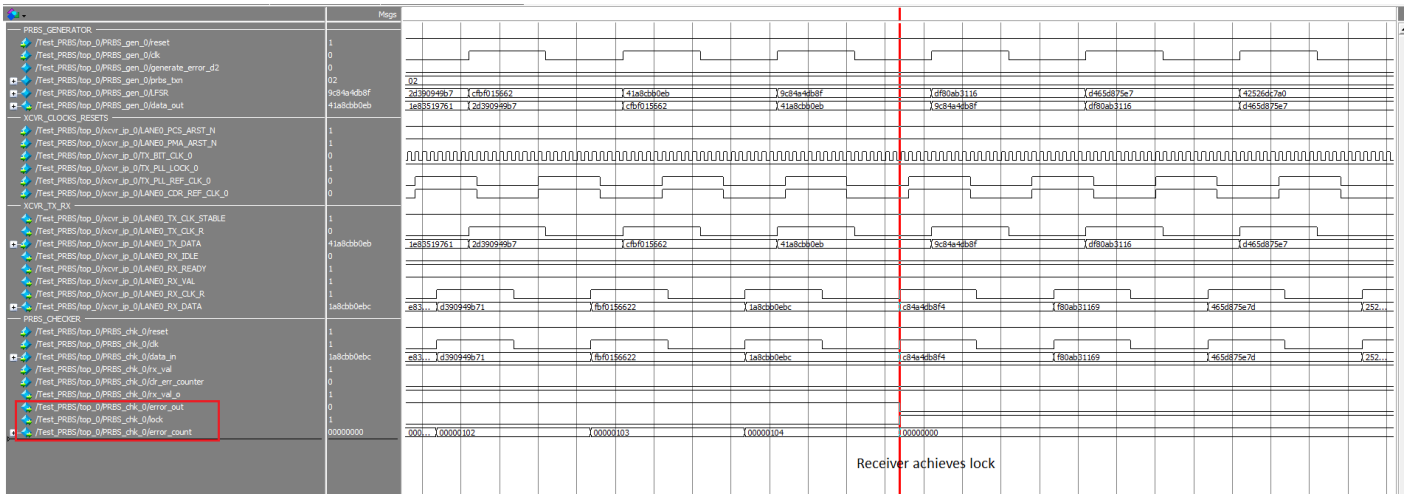


Figure 1-10. Simulation Waveform for PMA Design Highlighting PRBS Checker Lock



1.6 Reference Design 2B: PMA with Bit-slip Feature (Ask a Question)

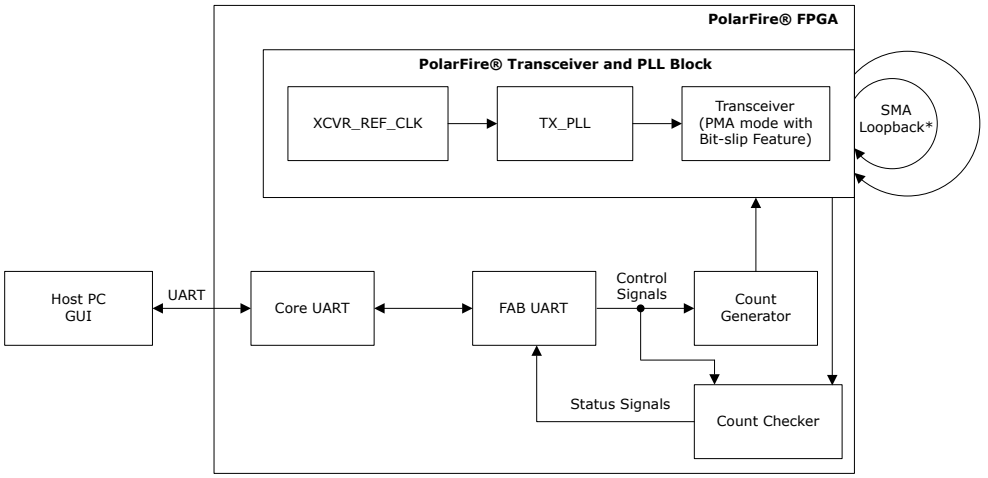
This reference design example implements the PolarFire transceiver in native PMA mode with Bit-slip feature. It includes a pattern generator to generate a 40-bit counter. The pattern checker checks the received data and compares it with the expected counter. The PolarFire transceiver is configured in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

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

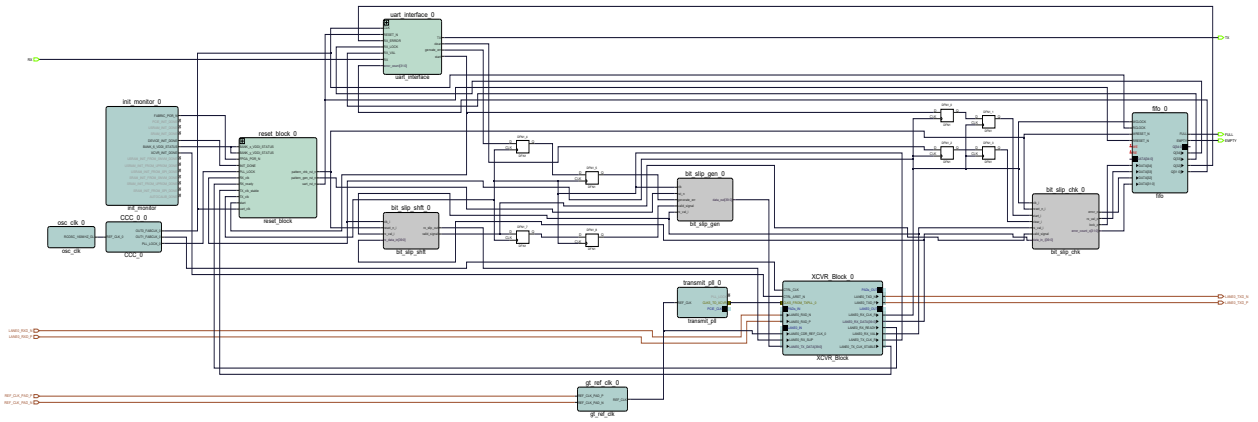
Figure 1-11. PMA with Bit-slip Feature Block Diagram



1.6.1 Design Implementation (Ask a Question)

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

Figure 1-12. Design Implementation of PMA with Bit-slip Feature



1.6.1.1 PolarFire Transceiver Configurator (Ask a Question)

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.

1.6.1.2 PolarFire Transceiver Reference Clock (Ask a Question)

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.

1.6.1.3 Transmit PLL (Ask a Question)

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

Important: A few PolarFire Evaluation boards might have an inconsistent 125 MHz oscillator that does not consistently supply 125 MHz. Due to this behavior, there might 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.

1.6.1.4 Clock Conditioning Circuitry (Ask a Question)

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

1.6.1.5 CDC_FIFO (Ask a Question)

CoreFIFO block synchronizes CDC data signals between fabric and UART interface.

1.6.2 Pattern Generator and Checker (Ask a Question)

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.

1.6.3 Bit-slip Shift (Ask a Question)

Bit-slip shift module receives data from transceiver. Using a finite state machine, it compares if the transceiver's 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 then used to trigger finite state machines in pattern_gen_0 and pattern_chk_0 modules.

1.6.4 Port Description (Ask a Question)

The following table lists the important ports for the PMA design.

Table 1-4. 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

Table 1-4. Port List for the PMA Design (continued)

Port	Direction	Description
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

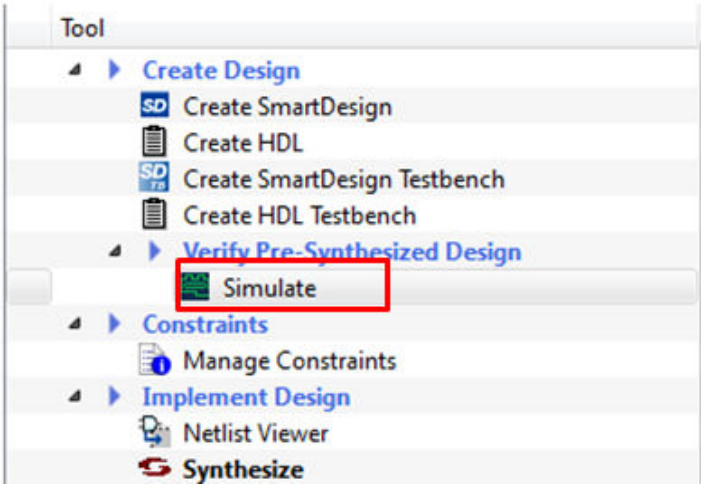
1.6.5 Simulating the Design (Ask a Question)

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

For simulation, perform the following the steps:

1. Open Libero SoC design built through the TCL scripts.
2. To run the simulation, navigate to **Stimulus Hierarchy**, right-click the `top.v` file, and select **Set as active stimulus**.
3. In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design, as shown in the following figure.

Figure 1-13. Simulation in Design Flow



1.6.6 Simulation Flow (Ask a Question)

The following steps describe the simulation flow:

1. Initially, the transceiver is 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. The `Bit_slip_shift` module receives data from transceiver and asserts 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. The simulation run time will be 72 us approximately.

Figure 1-14. Simulation Waveform for PMA Design Highlighting CDR Bit-slip Status

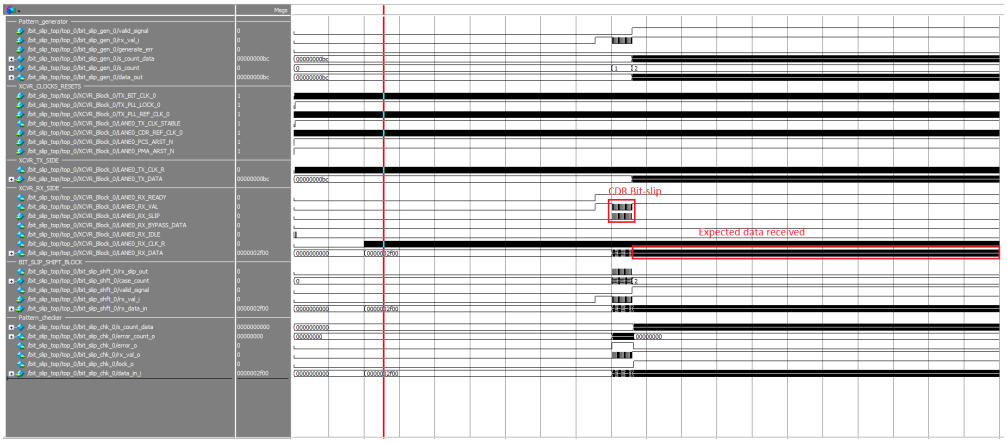
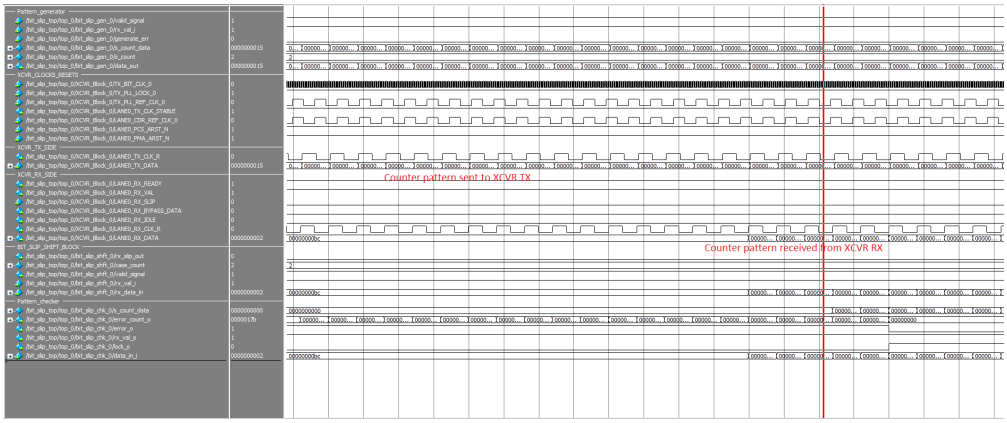


Figure 1-15. Simulation Waveform for PMA Design Highlighting Pattern Checker Lock

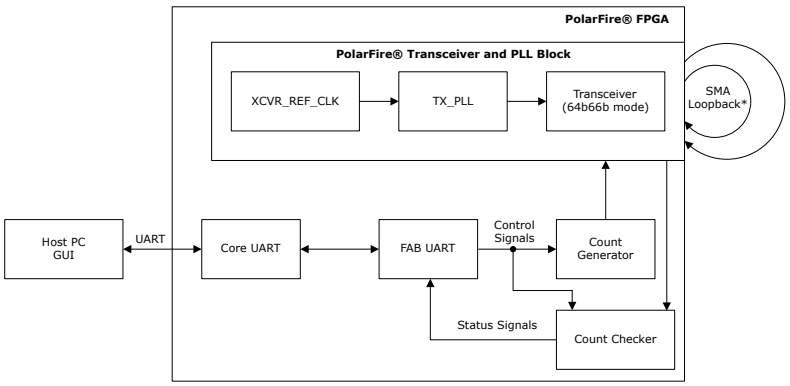


1.7 Reference Design 3: 64b66b Design (Ask a Question)

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 of the 64b66b design.

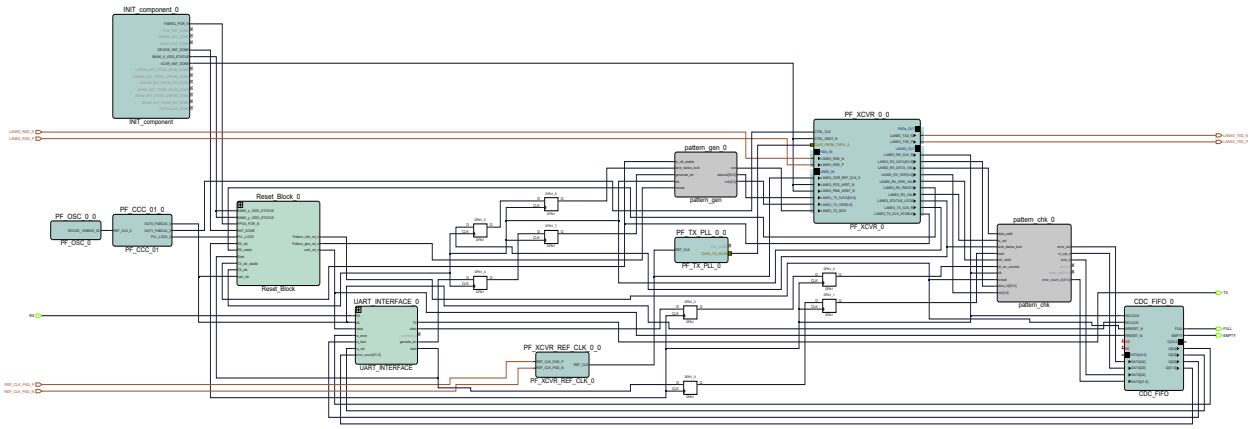
Figure 1-16. 64b66b Design Block Diagram



1.7.1 Design Implementation (Ask a Question)

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 1-17. 64b66b Design Implementation



1.7.1.1 PolarFire Transceiver Configurator (Ask a Question)

The PolarFire Transceiver configurator is set to 10 Gbps, 64-bit PCS-Fabric interface width, and 64b66b mode with Enhanced Receiver management.

1.7.1.2 PolarFire Transceiver Reference Clock (Ask a Question)

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

1.7.1.3 Transmit PLL (Ask a Question)

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

1.7.1.4 Clock Conditioning Circuitry (Ask a Question)

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

1.7.1.5 CDC_FIFO (Ask a Question)

CoreFIFO block synchronizes Clock Domain Crossing (CDC) data signals between fabric and UART interface.

1.7.2 Pattern Generator and Checker (Ask a Question)

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.

1.7.3 Port Description (Ask a Question)

The following table lists the important ports for the design.

Table 1-5. Port List for the 64b66b Design

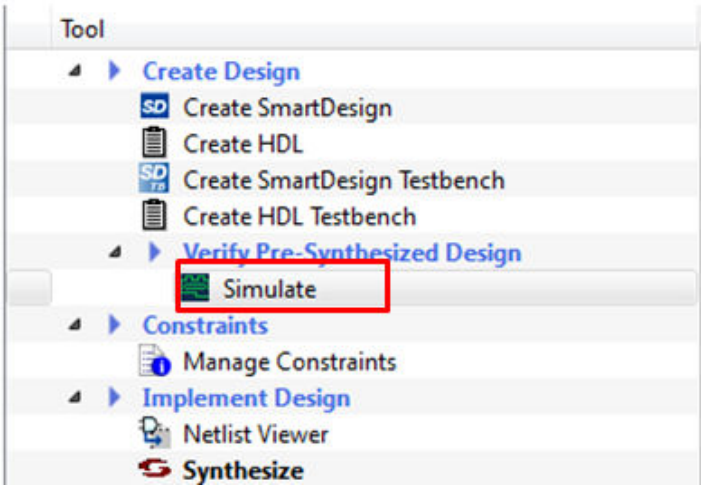
Port	Direction	Description
LANE0_RXD_P	Input	Transceiver receiver differential input
LANE0_RXD_N	Input	Transceiver receiver differential input
REF_CLK_PAD_P	Input	Transmit PLL input clock from reference clock interface
REF_CLK_PAD_N	Input	Transmit PLL input clock from reference clock interface
LANE0_PCS_ARST_N	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

1.7.4 Simulating the Design (Ask a Question)

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. Open Libero SoC design built through the TCL scripts.
2. To run the simulation, navigate to **Stimulus Hierarchy**, right-click the `top.v` file, and select **Set as active stimulus**.
3. In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design, as shown in the following figure.

Figure 1-18. Simulation in Design Flow



1.7.5 Simulation Flow (Ask a Question)

The following steps describe the simulation flow:

1. Initially, the transceiver is 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 figures show the simulation waveform for the 64b66b design highlighting pattern checker status signals and Tx/Rx data match. The simulation run time will be 78 us approximately.

Figure 1-19. Simulation Waveform for 64b66b Design Highlighting Pattern Checker Status

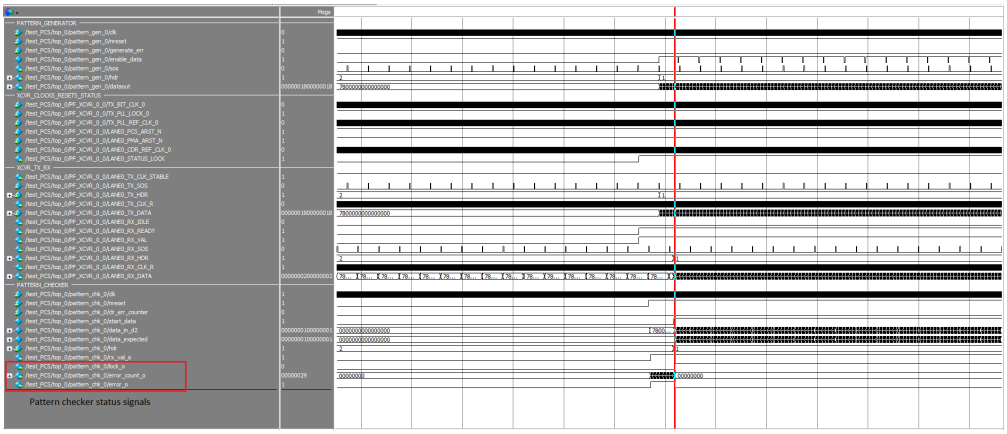
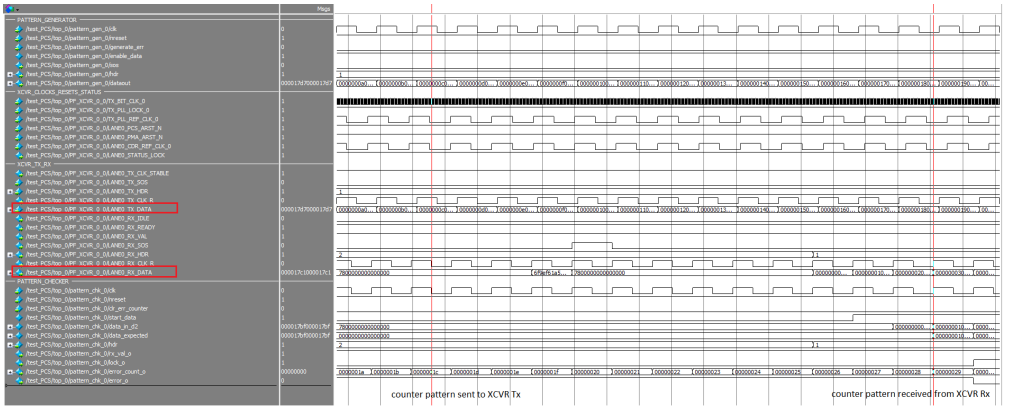


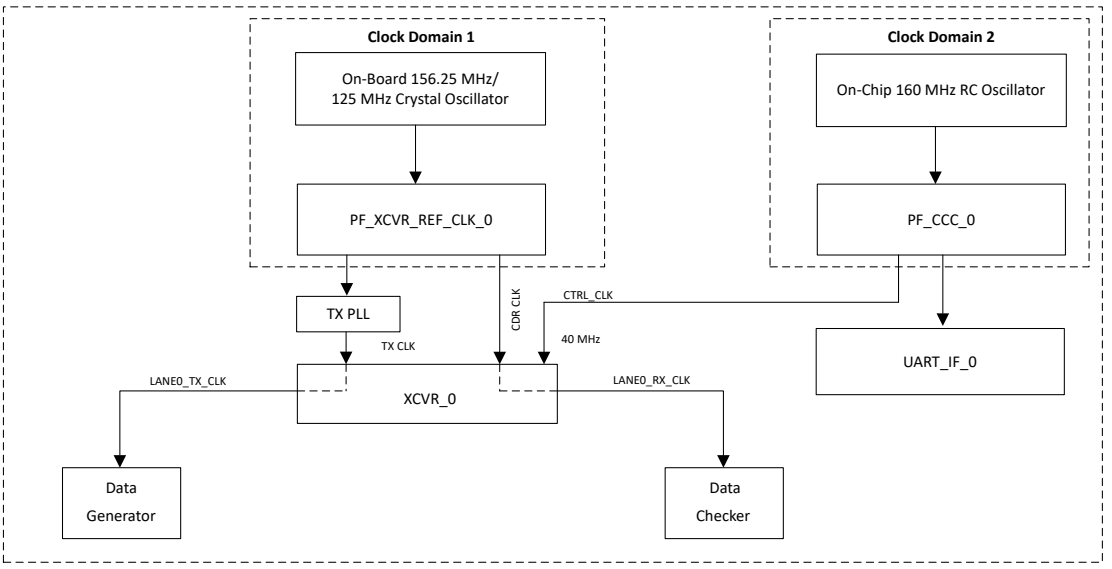
Figure 1-20. Simulation Waveform for 64b66b Design Highlighting Tx and Rx Data Match



1.8 Clocking Structure (Ask a Question)

Evaluation kit's reference designs have 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.

Figure 1-21. Clocking Structure



1.9 Reset Structure (Ask a Question)

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.

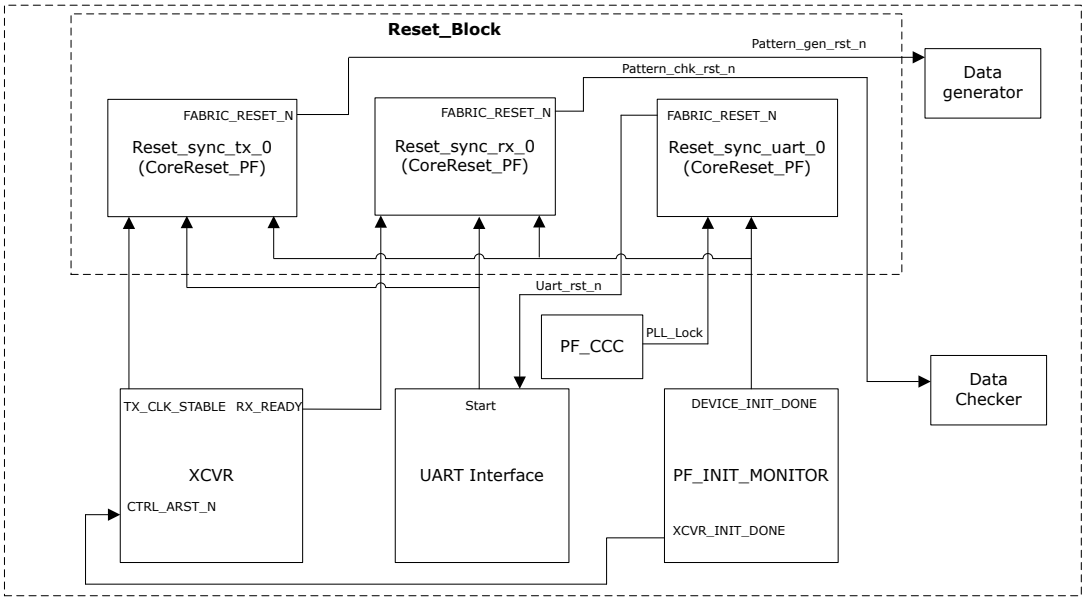
Reset_sync_uart_0 (CoreReset_PF) module releases active-low reset of UART_INTERFACE when PLL_LOCK output from PF_CCC block and DEVICE_INIT_DONE signal from PF_INIT_MONITOR block are asserted.

DEVICE_INIT_DONE signal is asserted when the device initialization is complete. For more information about device initialization, see PolarFire Family 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 1-22. Reset Structure



2. Programming the Device on the Evaluation Kit (Ask a Question)

Once the design is created through TCL scripts as mentioned in Appendix 4: Running the TCL Script, open the design in Libero® SoC software. To program the PolarFire® device, perform the following steps:

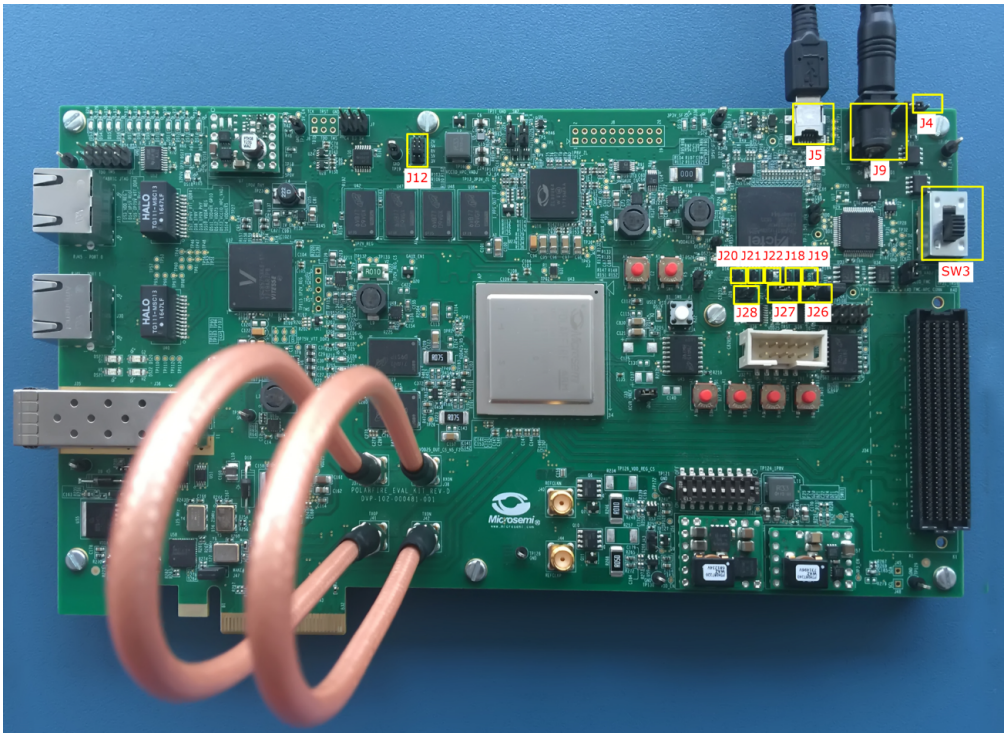
1. Ensure that the jumper settings on the board are same as listed in the following table.

Table 2-1. 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 2.5V
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, see the following figure. The following figure shows the board setup.

Figure 2-1. Board Setup for Evaluation Kit



6. Double-click **Run PROGRAM Action**.

When the device is programmed successfully, a green tick mark appears, see the following figure. See Running the Demo to run the multi-rate transceiver demo.

Figure 2-2. Programming the Device

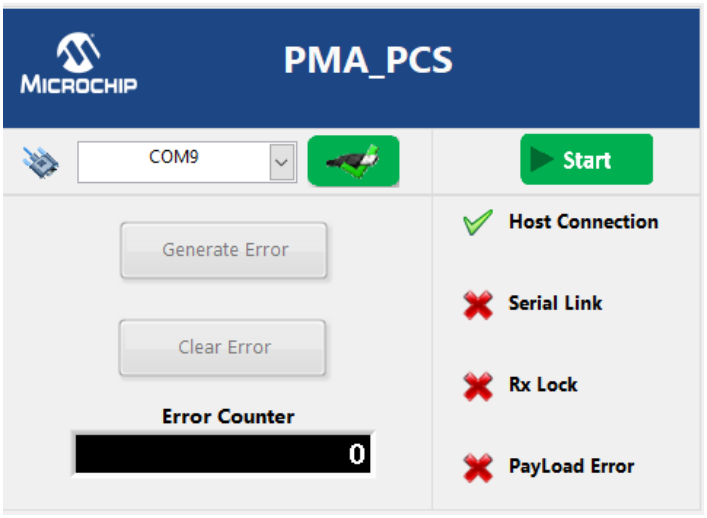
3. Running the Demo (Ask a Question)

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

To install the GUI, perform the following steps:

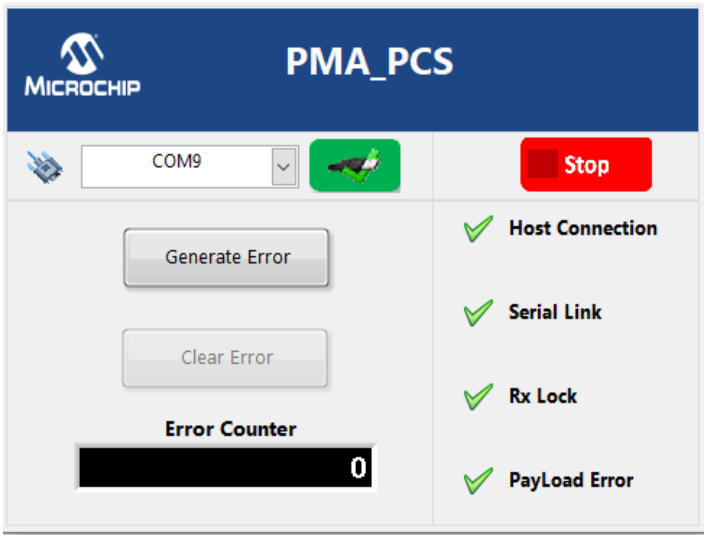
1. Launch the **PMA_PCS GUI** application. If the GUI application is not yet installed, see Prerequisites section.
2. Go to **All Programs** > **PMA_PCS > PMA_PCS**. The PMA_PCS Demo window is displayed.
3. Select the COM port number that is detected to configure the serial port.
4. Click the icon to 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, see the following figure.

Figure 3-1. Selecting COM Port and Connecting



5. 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

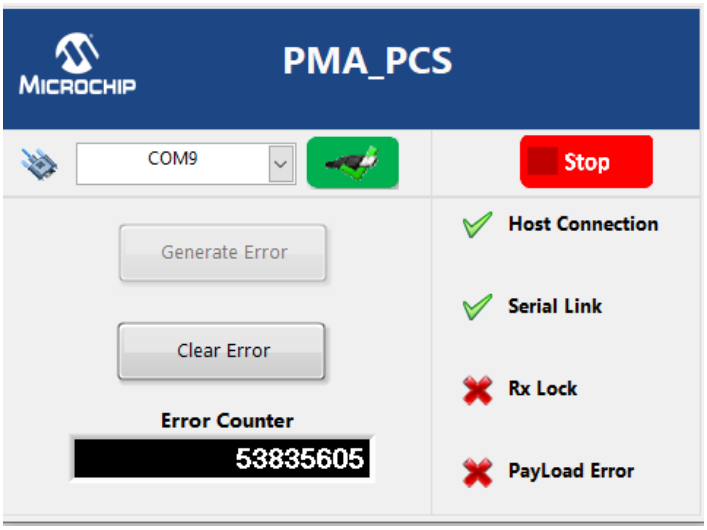
Figure 3-2. PMA PCS Status Signals



6. Click **Generate Error** to generate error in the data and observe the error status, see the following figure.

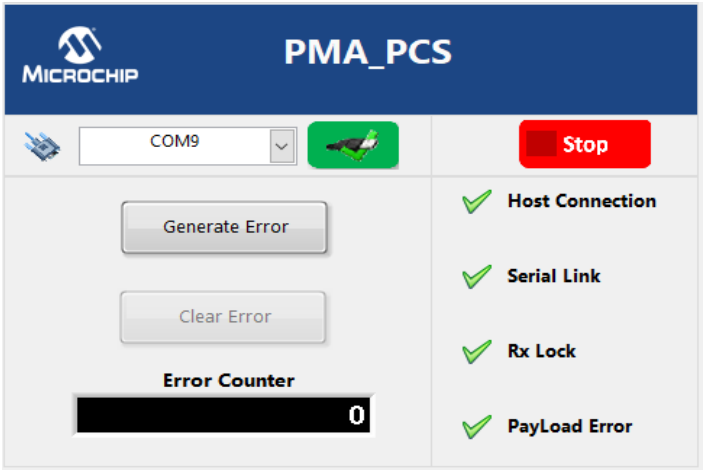
> **Important:** The error counter displays a 32-bit counter running at a very high frequency clock. It keeps incrementing until injected error is cleared by clicking **Clear Error**.

Figure 3-3. Generate Data Error



7. Click **Clear Error** to stop generating error and observe that **Rx Lock** and **PayLoad Error** turns green, and Error Counter is 0, see the following figure.

Figure 3-4. Clear Data Error



8. Click **Stop**.

The Multi-rate transceiver demo is successfully run.

4. Appendix 1: PolarFire Transceiver Overview (Ask a Question)

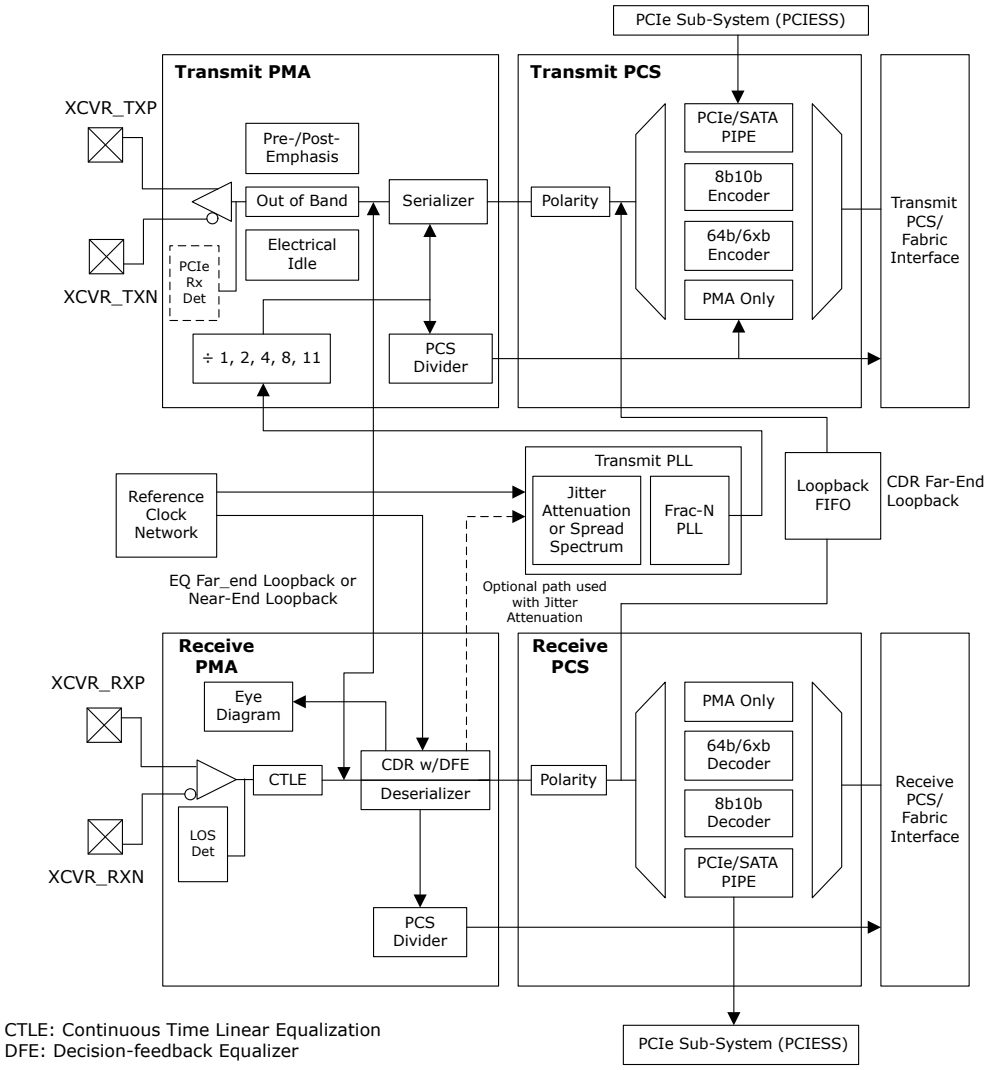
PolarFire FPGA transceivers include all required analog functions for high-speed data transmission between devices over Printed Circuit Boards (PCB) and high-quality cables. They are optimized for low-power operation and are suitable for a variety of device-to-device communication protocols.

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 bitstream. 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 to 10GBASE-R 66-bit control or data blocks in accordance with Clause 49 of the IEEE® 802.3-2008 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 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 1 GbE interfaces. The CoreTSE suite of 1 GbE 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 PolarFire Family Transceiver User Guide.

Figure 4-1. PolarFire FPGA Transceiver



The Microchip 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.

5. Appendix 2: How to Use SmartBERT IP (Ask a Question)

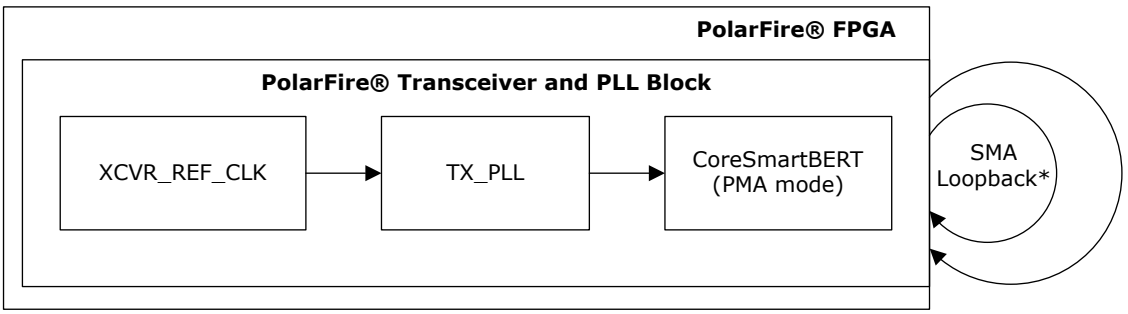
The CoreSmartBERT provides a demonstration platform for the PolarFire transceiver (PF_XCVR). It can be customized to use different line rates and reference clock rates. PRBS data pattern generators and checkers are included in the core. The pattern generator sends data out through the transmitter, accepts 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.

5.1 Reference Design: SmartBERT IP Design (Ask a Question)

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 of the SmartBERT design.

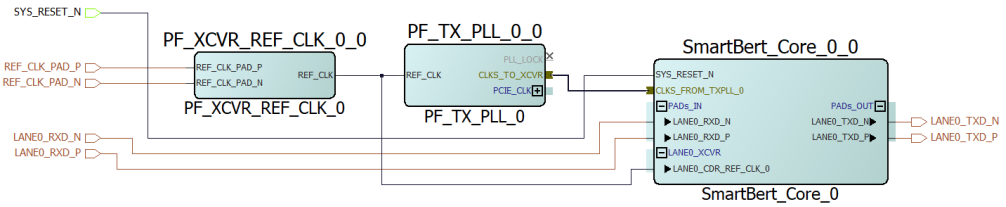
Figure 5-1. SmartBERT Design Block Diagram



5.1.1 Design Implementation (Ask a Question)

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

Figure 5-2. CoreSmartBERT IP Design Implementation



5.1.1.1 PolarFire CoreSmartBERT IP Configurator (Ask a Question)

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.

5.1.1.2 PolarFire Transceiver Reference Clock (Ask a Question)

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.

5.1.1.3 Transmit PLL (Ask a Question)

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

5.1.2 Port Description (Ask a Question)

The following table lists the important ports for the design.

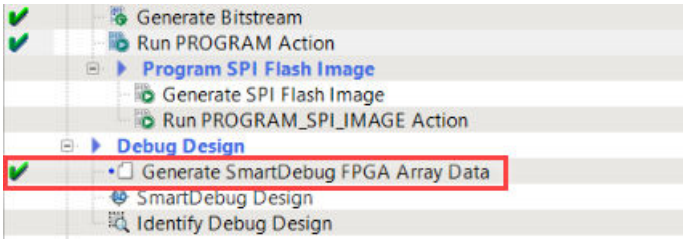
Table 5-1. Port List for the CoreSmartBERT IP Design

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

5.2 How to Use SmartBERT (Ask a Question)

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, see the following figure.

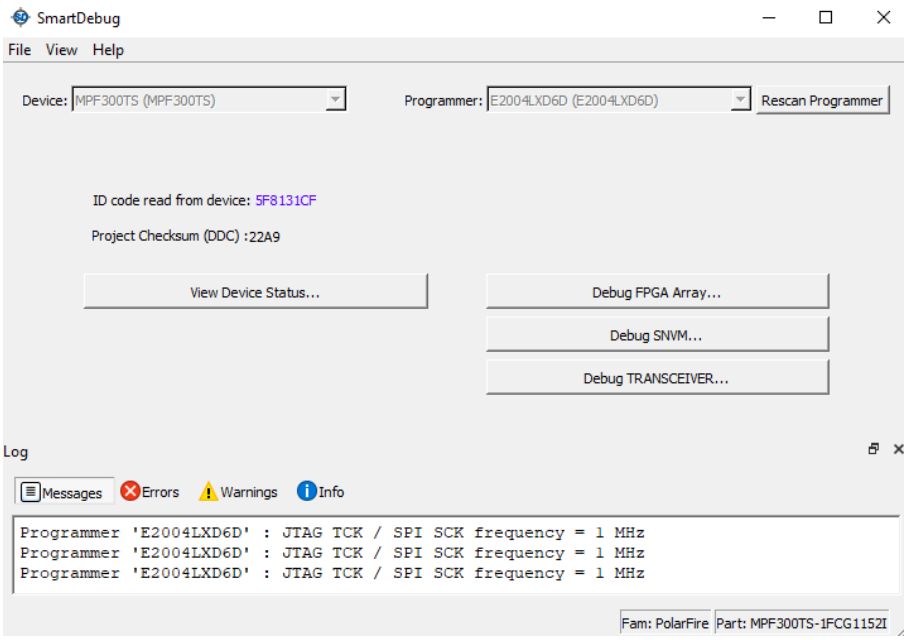
Figure 5-3. Launching SmartDebug Design Tools



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

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

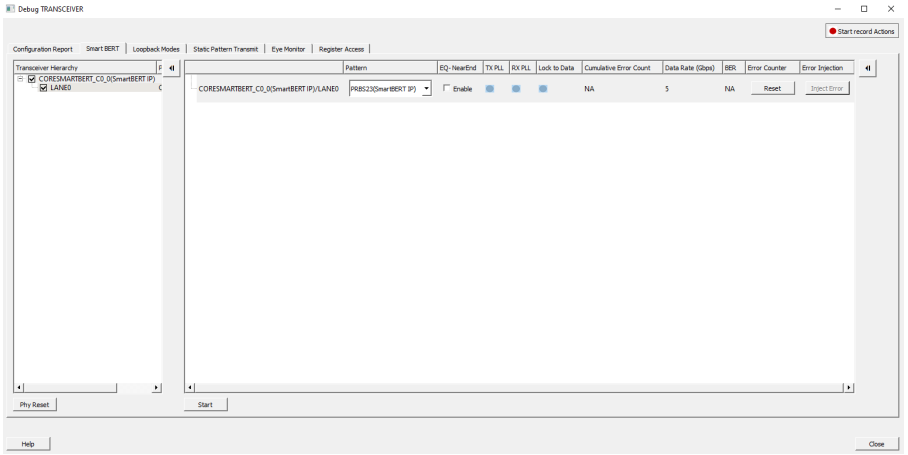
Figure 5-4. SmartDebug Window Debug Options



To run **Smart BERT** in **Debug TRANSCEIVER**, perform the following steps:

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

Figure 5-5. 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.

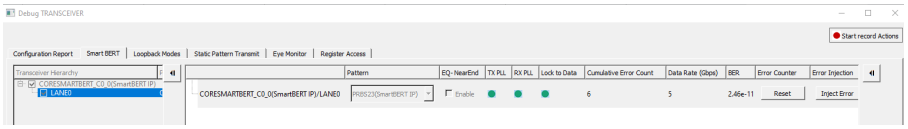
Figure 5-6. Debug TRANSCEIVER—Status

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 5-7. Smart BERT—Cumulative Error Count



For more information about Debug transceiver features, see AN4594: Debugging PolarFire FPGA Using SmartDebug.

Important: If any of the probe points in SmartBERT tab are not working as expected, see Appendix: Known Issues section in AN4594: Debugging PolarFire FPGA Using SmartDebug.

The SmartBERT reference design is configured for 5 Gbps transceiver data rate. Separate programming files (`*.job`) and corresponding design Debug Data Container (DDC) are exported from Libero and provided in the following locations:

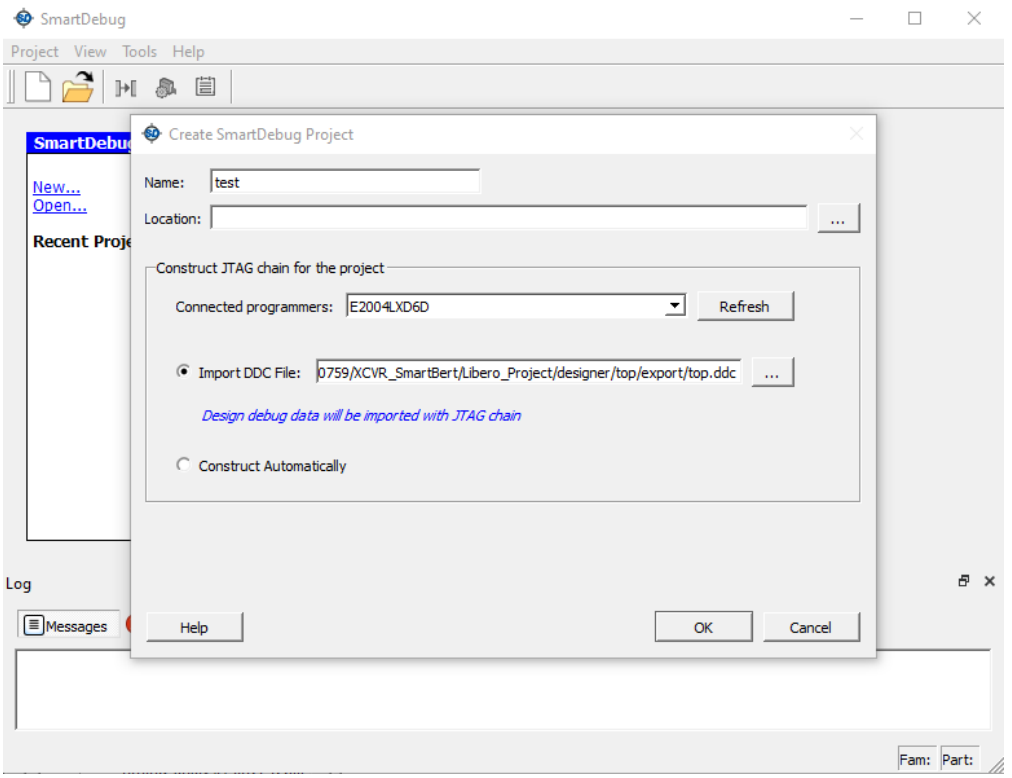
- `*.job file: mpf_an4662_df/Programming_File`
- `*.ddc file: mpf_an4662_df/Source_File`

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

To import `*.ddc` file in standalone SmartDebug, perform the following steps.

1. Launch SmartDebug in standalone mode.
2. In the **SmartDebug** window, click **Project > New Project**. The **Create SmartDebug Project** dialog box opens, see 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 5-8. Create SmartDebug Project



For more information about how to export DDC file from Libero, see AN4594: Debugging PolarFire FPGA Using SmartDebug.

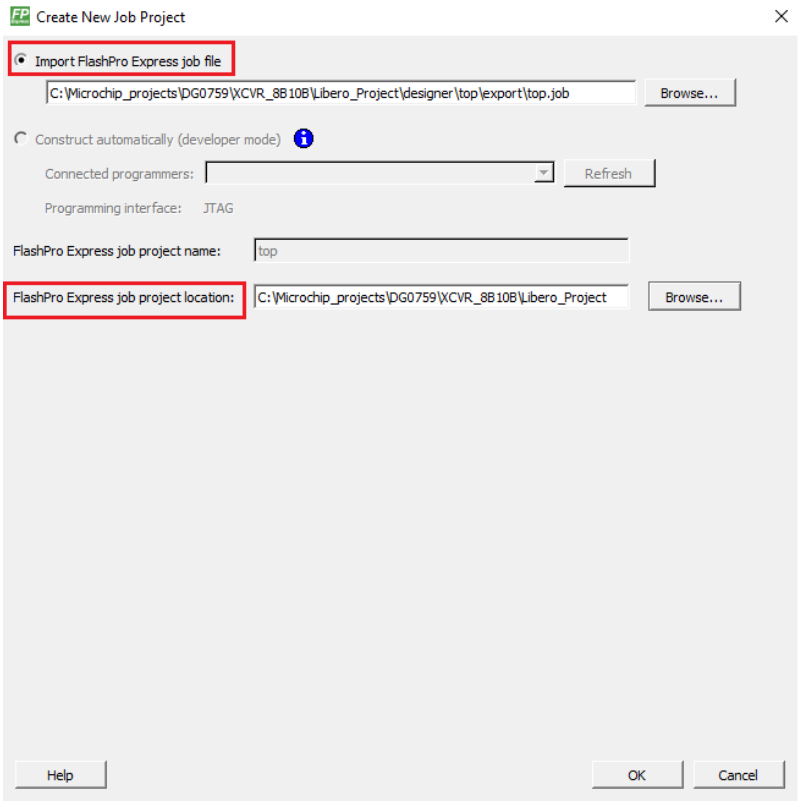
6. Appendix 3: Programming the Device Using FlashPro Express (Ask a Question)

This section describes how to program the PolarFire device with the `.job` programming file using FlashPro Express. The `.job` files for all the 5 designs are available at the following location: `mpf_an4662_df/Programming_Files`

To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are same as listed in Table 2-1 (for Evaluation board).
2. Connect the power supply cable to the J9 connector on the Evaluation board.
3. Connect the USB cable from the Host PC to J5 (FTDI port) on the Evaluation board.
4. Power on the board using the SW3 slide switch on the Evaluation board.
5. Connect **TXN** to **RXN** and **TXP** to **RXP** using the two SMA to SMA cables on the Evaluation board, see Figure 2-1.
6. On the host PC, launch the **FlashPro Express** software.
7. To create a new job project, click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu.
8. In the **New Job Project from FlashPro Express Job** dialog box, enter the following:
   - **Programming job file**: Click **Browse**, and navigate to the location where the `.job` 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.

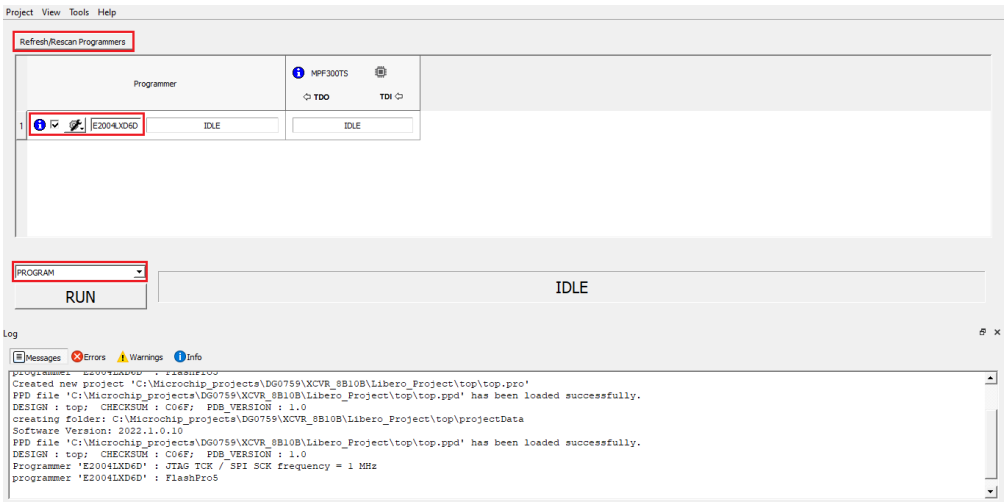
Figure 6-1. New Job Project from FlashPro Express Job



9. Click **OK**. The required programming file is selected and ready to be programmed in the device.

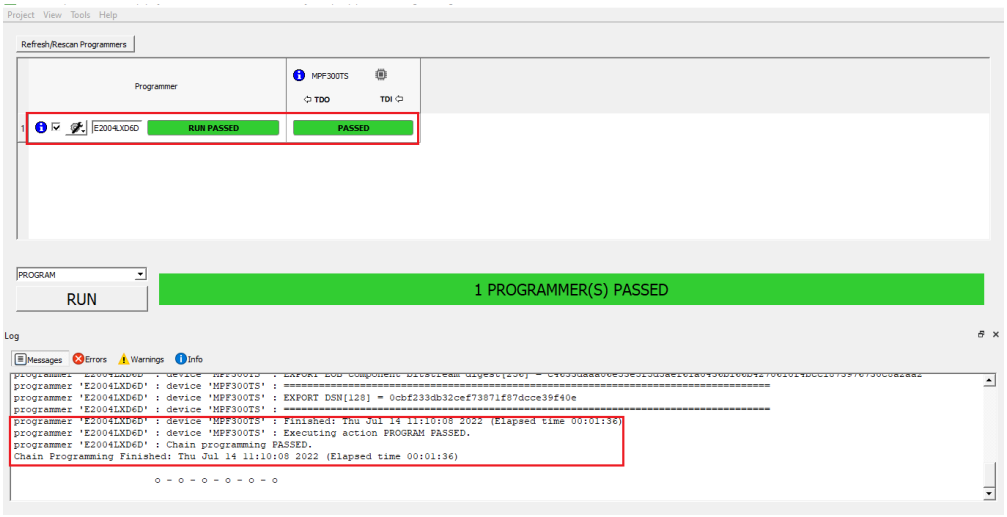
10. The FlashPro Express window appears, as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**, as shown in the following figure.

Figure 6-2. Programming the Device



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

Figure 6-3. FlashPro Express—RUN PASSED



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

7. Appendix 4: Running the TCL Script (Ask a Question)

TCL scripts are provided in the design files folder under directory `mpf_an4662_df/HW/<8b10b/64b66b/PMA/PMA_bitslip/Smartbert>`. If required, the design flow can be reproduced from Design Implementation till the generation of job file.

To run the TCL, perform the following steps:

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

After successful execution of TCL script, Libero project is created within `HW` directory. For more information about TCL scripts, see `mpf_an4662_df/TCL_Script_readme.txt`.

For more details on TCL commands, see Tcl Commands Reference Guide. Contact Technical Support for any queries encountered while running the TCL script.

Important: While running the scripts, simulation runs in the Batch mode by default, which takes more time for the design to complete the build and compilation. If you do not want to run simulation, then you can comment the simulation flow command in the `script.tcl`.

8. Appendix 5: References (Ask a Question)

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 AN4592: PolarFire FPGA Dynamic Reconfiguration Interface Application Note.
- For information about PolarFire transceiver blocks, PF_XCVR, PF_TX_PLL, and PF_XCVR_REF_CLK, see PolarFire Family Transceiver User Guide.
- For more information about CCC, see PolarFire Family Clocking Resources User Guide.
- For more information about Libero, ModelSim, and Synplify, see the Libero SoC Documentation web page.
- For more information about device and memory initialization, see PolarFire Family Power-Up and Resets User Guide.
- For more information about SmartDebug features, see AN4594: Debugging PolarFire FPGA Using SmartDebug Application Note.
- For more information about PolarFire FPGA Evaluation Kit, see UG0747: PolarFire FPGA Evaluation Kit User Guide.
- For more information about CoreUART, see *CoreUART Handbook*. This user guide can be downloaded from Libero SoC Catalog.

9. Revision History (Ask a Question)

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

Table 9-1. Revision History

Revision	Date	Description
B	03/2025	The following is the summary of the changes made in B revision. • Updated the document for Libero v2024.2. • Updated the `.job` filepath and TCL script filepath throughout the document. • Updated Table 1-1 and note in the Design Requirements section. • Added information regarding Microchip FPGA_GUI_PACK in the Prerequisites section. • Updated Demo Designs section. • Updated Figure 1-21 in the Clocking Structure section. • Updated Figure 1-22 in the Reset Structure section. • Updated Design Implementation figures throughout the document. • Updated GUI application figures in the Running the Demo section.
A	08/2022	The following is the summary of the changes made in A revision. • The document was migrated to the Microchip template. • The document number was updated to DS00004662A from 50200759. • The document ID was updated to AN4662 from DG0759. • Removed Splash Kit content throughout the document. • Updated figures in Figure 1-1. • Updated Design Implementation figures throughout the document. • Updated content of Simulating Design section throughout the document. • Updated Simulation Flow figures throughout the document. • Updated Figure 1-1, Figure 1-6, Figure 1-11, Figure 1-16, and Figure 5-1. • Updated all table of Resource Utilization section. • Updated all figures of Running the Demo. • Updated all figures of How to Use SmartBERT. • Updated all design file locations throughout the document. • Updated all figures of Appendix 3: Programming the Device Using FlashPro Express.
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.
9.0	—	The document was updated for Libero SoC v12.0.
8.0	—	Merged Splash kit related content and updated the document for Libero SoC PolarFire v2.3.
7.0	—	The document was updated for Libero SoC PolarFire v2.2.
6.0	—	The document was updated for Libero SoC PolarFire v2.1.

Table 9-1. Revision History (continued)

Revision	Date	Description
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.
4.0	—	The design files were updated to include enhancements to the GUI logic.
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 and Running the Demo. • A list of reference documents was added. See Appendix 4: References.
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.
1.0	—	The first publication of this document.

Microchip FPGA Support

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

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

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

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

Microchip Information

Trademarks

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

ISBN: 979-8-3371-0771-4

Legal Notice

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

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

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

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

Microchip Devices Code Protection Feature

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

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