DG0783
Demo Guide
PolarFire FPGA: High-Speed Data Transfer in 8b10b Mode Using the LiteFast IP

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

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

About Microsemi

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

Contents

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 most current publication.

1.1 Revision 3.0

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

- Updated the document for Libero v2021.2.
- Updated Table 1, page 2, Table 2, page 6, and Table 4, page 29.
- The following list of figures are replaced.
    - Figure 2, page 4 through Figure 6, page 6
    - Figure 7, page 7
    - Figure 8, page 8
    - Figure 10, page 10
    - Figure 12, page 12
    - Figure 13, page 13
    - Figure 16, page 15
    - Figure 17, page 17
    - Figure 19, page 19
    - Figure 20, page 20
    - Figure 45, page 38
- Removed information about PF_OSC_0, see PF_OSC_0 from this document.
- Added information about PF_CLK_DIV_C0_0, see PF_CLK_DIV_C0_0, page 14.
- Updated information about COREFIFO, see COREFIFO, page 16.
- Added Figure 18, page 18.
- Added information about LiteFast_XCVR_Top, see LiteFast_XCVR_Top, page 21.
- Updated information in chapter Programming the Device Using FlashPro Express, see Appendix 1: Programming the Device Using FlashPro Express, page 37.
- Added Appendix 2: Running the TCL Script, page 39.

1.2 Revision 2.0

A note about an inconsistent 125 MHz oscillator that does not supply 125 MHz constantly on few PolarFire Evaluation boards was added, see Table 2, page 6.

1.3 Revision 1.0

The first publication of this document.

2 High-Speed Data Transfer Using the LiteFast IP

This document describes how to run the LiteFast IP demo on the PolarFire Evaluation Board using the LiteFast GUI application. The GUI application is packaged along with the design files. The reference design is built using the PolarFire high-speed transceiver block in 8b10b mode and the LiteFast IP core. It operates in loopback mode because the TX and RX transceiver lanes are manually looped back on the board. This setup facilitates a standalone demo that does not require another board.

Microsemi's LiteFast IP core implements a serial, point-to-point, and light-weight protocol for high-speed serial communication. LiteFast IP creates a high-speed serial link by connecting to the transceiver block available in Microsemi's PolarFire® device. The high-speed transceiver block handles data rates ranging from 250 Mbps to 12.7 Gbps. The transceiver (PF_XCVR) module integrates several functional blocks to support high-speed serial data transfer within the FPGA.

The LiteFast IP supports data widths of 16, 32, and 64 bits and supports multiple transceiver lanes. In the reference design, the LiteFast IP is configured to 32-bit data width and single lane.

For more information about the LiteFast design implementation, and the necessary blocks and IP cores instantiated in Libero SoC PolarFire, see Demo Design, page 3.

The reference design can be programmed using any of the following options:

- Using the pre-generated .job file: To program the device using the .job file provided along with the reference design, see Appendix 1: Programming the Device Using FlashPro Express, page 37.
- Using Libero SoC PolarFire: To program the device using Libero SoC PolarFire, see Libero Design Flow, page 28.

The reference design can be used on two Microsemi PolarFire boards to implement a full-duplex data transfer application. For more information about the implementation of LiteFast IP for data transfer between two boards, see Using LiteFast For Board-to-Board Data Transfer, page 36.

2.1 Design Requirements

The following table lists the hardware and software design requirements for running this demo design.

Table 1 • **Design Requirements**

Requirement	Version
Operating System	Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT) – PolarFire evaluation board – 12 V/5 A wall-mounted power adapter – USB 2.0 A-male to mini-B cable for UART and programming	Rev B or later
2 SMA-to-SMA cables (not provided with the kit)	
Software	
FlashPro Express	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
Libero SoC PolarFire	
ModelSim	
Synplify Pro	

Table 1 • **Design Requirements *(continued)***

IP
LiteFast IP core
PF XCVR IP core
PF_TX_PLL
PF_XCVR_REF_CLK
CoreUART
COREFIFO

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 start:

1. Download the reference design files from the following location: *http://soc.microsemi.com/download/rsc/?f=mpf_dg0783_df*
2. Download and install Libero SoC PolarFire v2021.1 on the host PC from the following location. https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc-polarfire#downloads

   The latest versions of ModelSim and Synplify Pro are included in the Libero SoC PolarFire installation package.

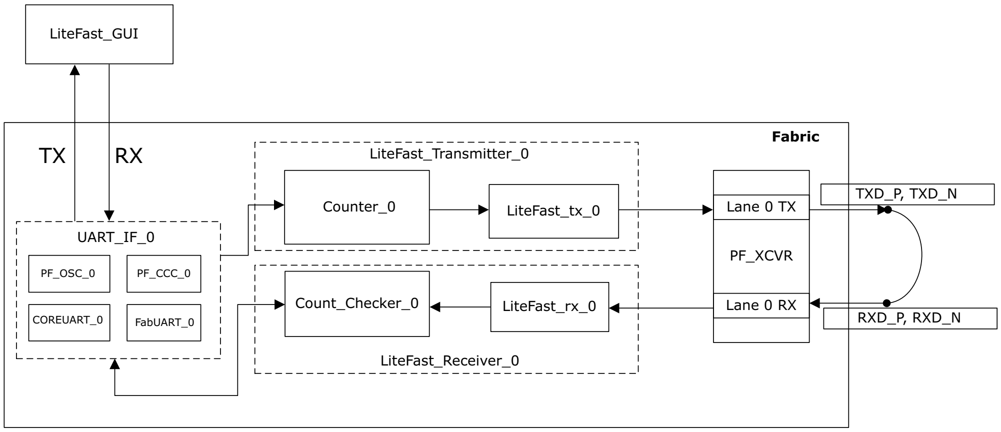
2.3 Demo Design

In the reference design:

1. The UART_IF_0 block interfaces with the GUI. This block receives RX signals to start and stop the LiteFast demo. This block drives the Counter_0 and the Count_Checker_0 blocks when the start signal is received. When a CRC error or payload error is selected on the GUI, the UART_IF_0 block receives that RX signal and passes it to the Counter_0 block for error injection.
2. The Counter_0 block acts as the application that transfers 32-bit parallel data to the LiteFast_tx_0 block.
3. The LiteFast_tx_0 block is the instantiation of LiteFast IP, configured as transmitter. It receives the 32-bit data, converts the data to LiteFast frames and forwards the data to the PF_XCVR_0 block.
4. The PF_XCVR_0 (transceiver) IP block receives the data on its TX lane, encodes the data in 8b10b format, and serializes the data. The encoded and serialized data is looped back to the RX lane.
5. PF_XCVR_0 decodes the data in 8b10b format, deserializes the data on its RX lane, and then sends the decoded data to the LiteFast_rx_0 block, which is the instantiation of LiteFast IP configured as receiver.
6. The Count_Checker_0 block generates 32-bit data in sync with Counter_0, and compares this data with the 32-bit data received from the LiteFast_rx_0 block. This block also sends the number of TX words transmitted, status of serial link, CRC error, and payload error to the UART_IF_0 block.
7. The UART_IF_0 block forwards these status and error information on its TX interface to the GUI for display.

The following figure shows the hardware implementation of the high-speed data transfer using the LiteFast IP.

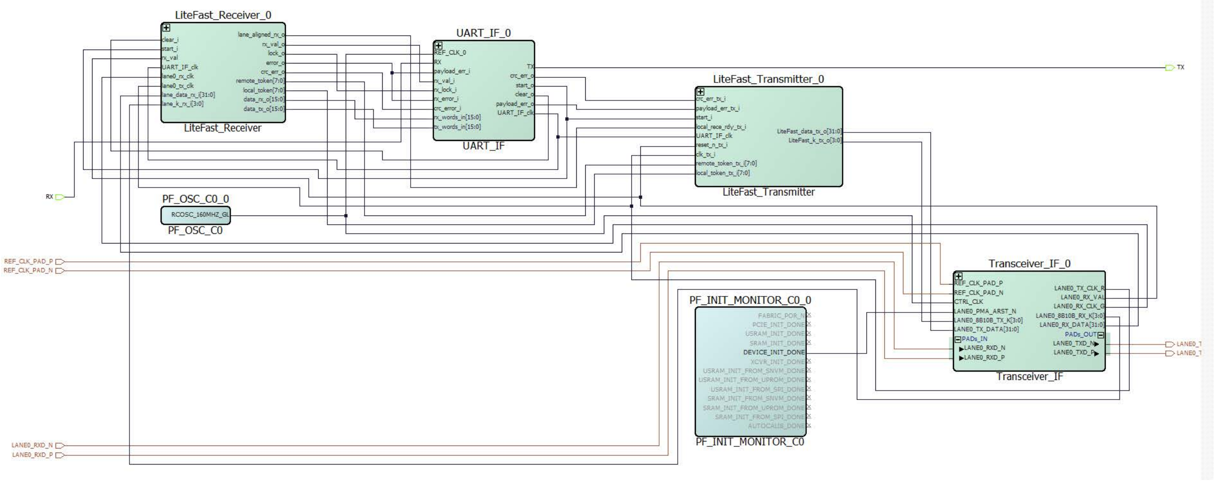
Figure 1 • **Hardware Implementation Block Diagram**



2.3.1 Design Implementation

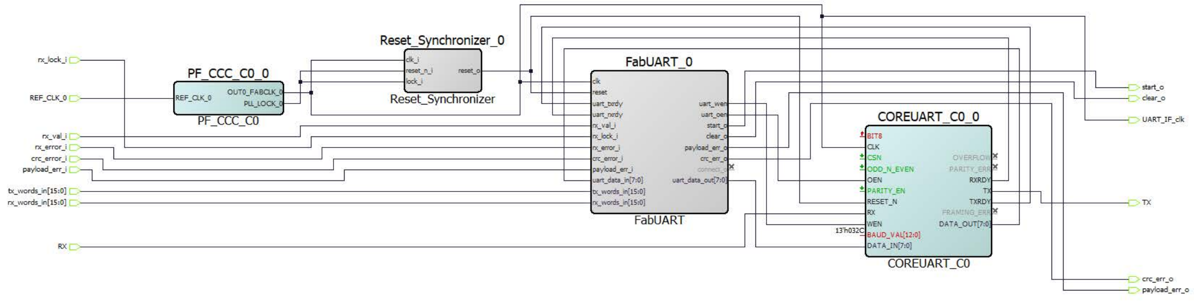
The following figure shows the top-level Libero design of the high-speed data transfer using LiteFast IP.

Figure 2 • **LiteFast Top-Level Design**



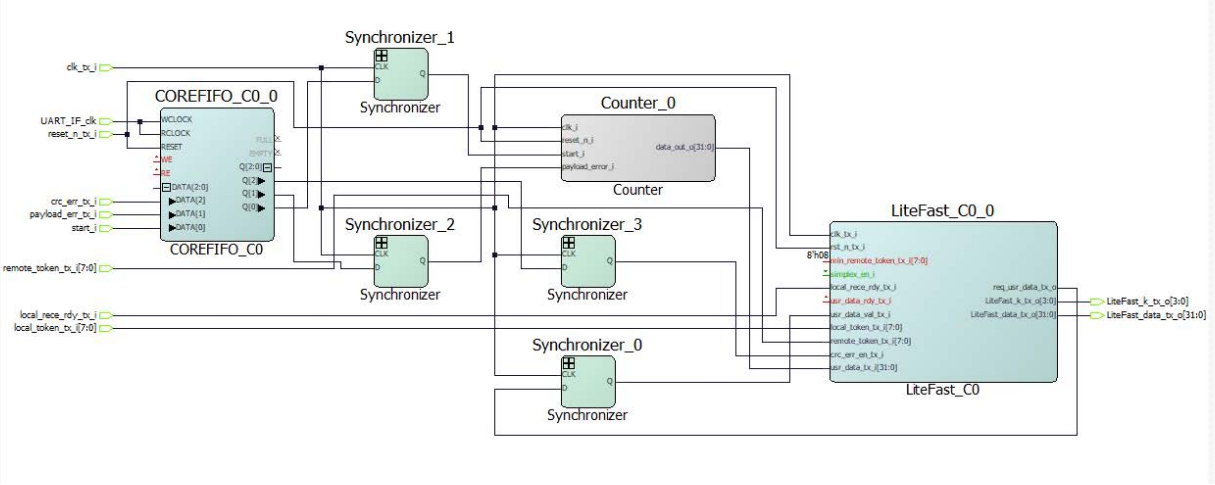
The sub-blocks of UART_IF_0 block are shown in the following figure.

Figure 3 • **UART_IF_0 Sub-Blocks**



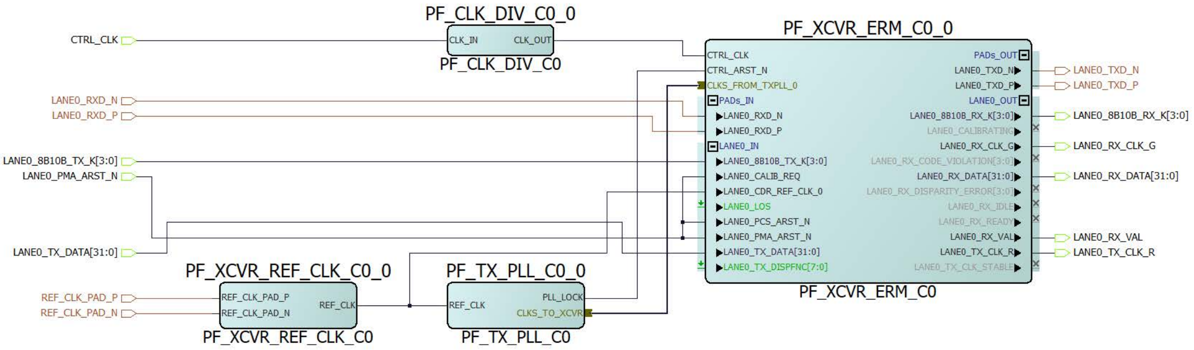
The sub-blocks of LiteFast_Transmitter_0 block are shown in the following figure.

Figure 4 • **LiteFast_Transmitter_0 Sub-Blocks**



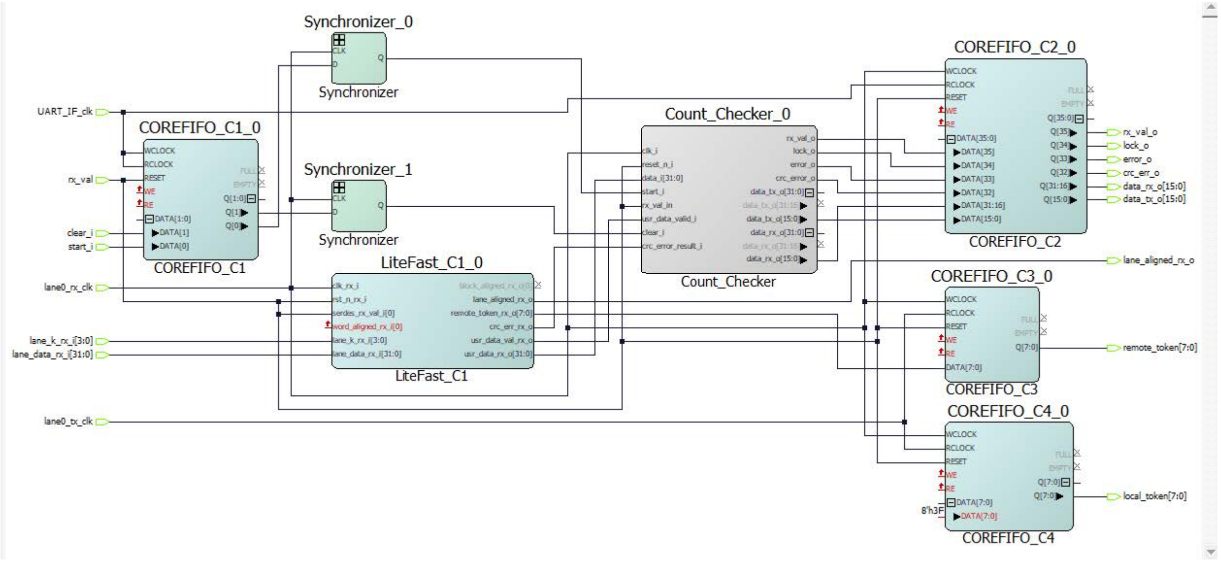
The sub-blocks of Transceiver_IF_0 block are shown in the following figure.

Figure 5 • **Transceiver_IF_0 Sub-Blocks**



The sub-blocks of LiteFast_Receiver_0 block are shown in the following figure.

Figure 6 • **LiteFast_Receiver_0 Sub-Blocks**



The following table lists the important I/O signals of the design.

Table 2 • **I/O Signals**

Signal	**Description**
Input Signals	
PADs_IN	Transceiver LANE0_RXD_P and LANE0_RXD_N on the board
REF_CLK_PAD_P_0 and REF_CLK_PAD_N_0	This is the differential reference clock generated from the on-board 125 MHz oscillator
RX	This is the input signal received by the UART interface from the GUI
TX	This is the output data received by the GUI from the UART interface
Output Signals	
PADs_OUT	LANE0_TXD_P and LANE0_TXD_N looped back to LANE0_RXD_P and LANE0_RXD_N using SMA cables

2.3.2 IP Configuration

The following sections describe the user-defined blocks, IP blocks, and their configurations for each top-level block.

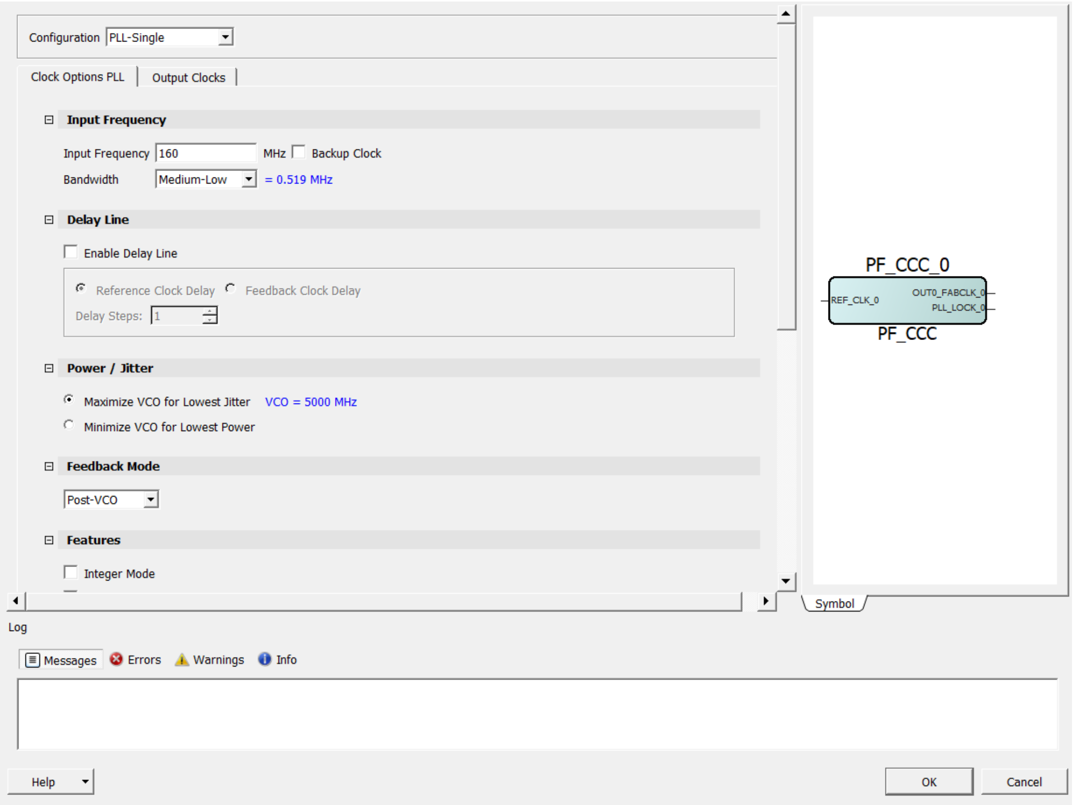
2.3.2.1 UART_IF_0 block

The UART_IF_0 block contains the PF_CCC, CoreUART, Reset_Synchronizer_0, and FabUART modules. These modules are described in the following sections.

2.3.2.1.1 PF_CCC_C0_0

The PF_CCC_C0_0 block provides 125 MHz output fabric clock to COREUART_C0_0 and FabUART_0 modules, which are fabric blocks. The following figures shows the input and output configurations of PF_CCC_C0_0.

Figure 7 • **PF_CCC_C0_0 Clock Options PLL**



Configuration PLL-Single

Clock Options PLL | Output Clocks

Input Frequency

Input Frequency 160 MHz □ Backup Clock

Bandwidth Medium-Low = 0.519 MHz

Delay Line

□ Enable Delay Line

● Reference Clock Delay ○ Feedback Clock Delay

Delay Steps: 1

Power / Jitter

● Maximize VCO for Lowest Jitter VCO = 5000 MHz

○ Minimize VCO for Lowest Power

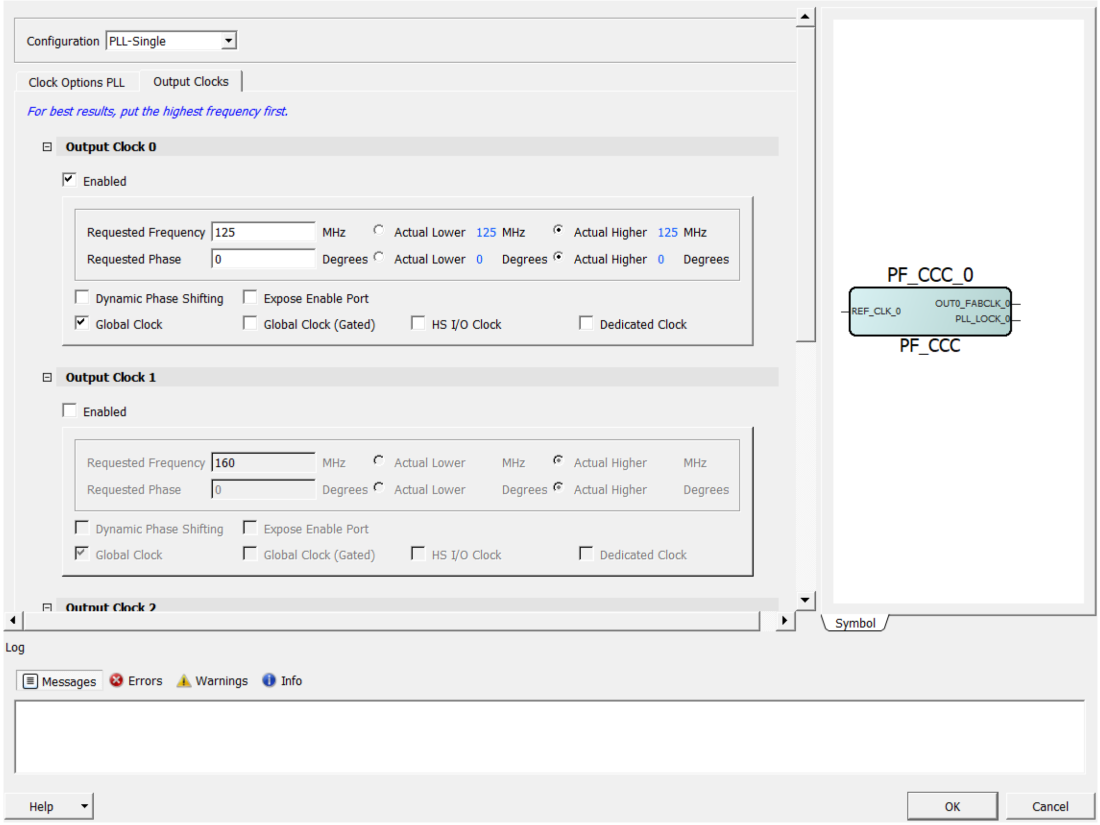
Feedback Mode

Post-VCO

Features

□ Integer Mode

Figure 8 • **PF_CCC_C0_0 Output Configuration**



Configuration PLL-Single

Clock Options PLL | Output Clocks

For best results, put the highest frequency first.

⊟ **Output Clock 0**

☑ Enabled

Requested Frequency 125 MHz ○ Actual Lower 125 MHz ● Actual Higher 125 MHz

Requested Phase 0 Degrees ○ Actual Lower 0 Degrees ● Actual Higher 0 Degrees

☐ Dynamic Phase Shifting ☐ Expose Enable Port

☑ Global Clock ☐ Global Clock (Gated) ☐ HS I/O Clock ☐ Dedicated Clock

⊟ **Output Clock 1**

☐ Enabled

Requested Frequency 160 MHz ○ Actual Lower MHz ● Actual Higher MHz

Requested Phase 0 Degrees ○ Actual Lower Degrees ● Actual Higher Degrees

☐ Dynamic Phase Shifting ☐ Expose Enable Port

☑ Global Clock ☐ Global Clock (Gated) ☐ HS I/O Clock ☐ Dedicated Clock

⊟ **Output Clock 2**

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

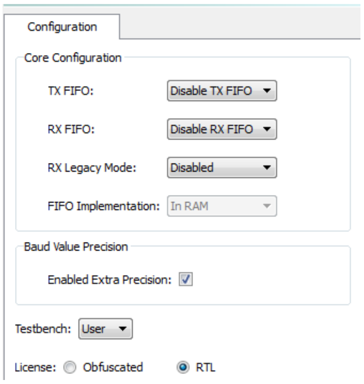
2.3.2.1.2 COREUART_C0_0

User inputs from the GUI are received by the COREUART module, which converts this serial input data to parallel data and forwards the data to the FabUART module for further processing. Both COREUART_C0_0 and FabUART_0 modules run at the same frequency. Hence, the TX and RX FIFO options are disabled in the COREUART_C0_0 configurator as shown in Figure 9, page 9. For more information about CoreUART, see *HB0095: CoreUART Handbook*.

2.3.2.1.3 Reset_Synchronizer_0

The Reset_Synchronizer_0 block is a two stage synchronizer, which synchronizes the PLL_LOCK_0 signal.

Figure 9 • **COREUART_0 Configurator**



Configuration

Core Configuration

TX FIFO: Disable TX FIFO

RX FIFO: Disable RX FIFO

RX Legacy Mode: Disabled

FIFO Implementation: In RAM

Baud Value Precision

Enabled Extra Precision: ☑

Testbench: User

License: ○ Obfuscated ● RTL

2.3.2.1.4 FabUART_0

The FabUART module drives the Counter_0 and Count_Checker_0 modules, and receives the data and error information from the Count_Checker_0 module. The FabUART module passes this data to the CoreUART, which converts this parallel data to serial and forwards the data to the GUI. It is a user-defined module.

2.3.2.2 LiteFast_Transmitter_0

This block contains the Counter_0, LiteFast_tx_0, Synchronizer_0, and the COREFIFO_0 modules. These modules are described in the following sections.

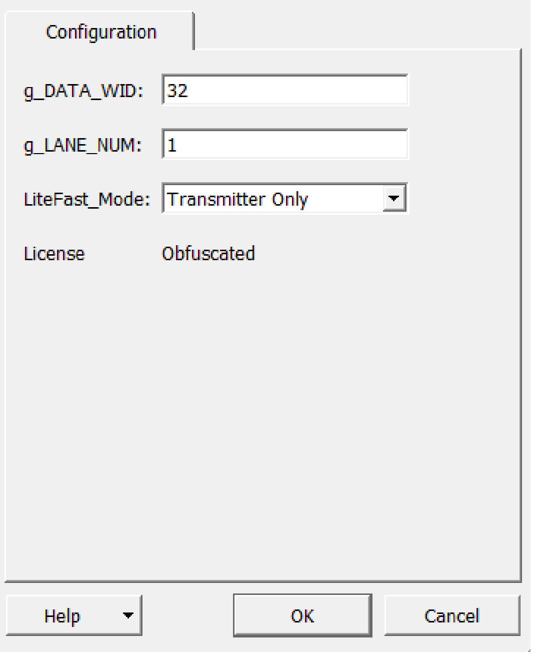
2.3.2.2.1 Counter_0

The counter_0 module implements a 32-bit counter that transmits incremental data to the LiteFast_tx_0 module at each clock cycle.

2.3.2.2.2 LiteFast_C0_0

LiteFast_C0_0 is the instantiation of the LiteFast IP configured as transmitter. This IP core receives the incremental data from Counter_0 and converts that data to LiteFast frames. These frames are sent to the Transceiver_IF_0 block over a single lane. The following figure shows this data width and lane configuration.

Figure 10 • **LiteFast_C0_0 Configurator**



g_DATA_WID: 32

g_LANE_NUM: 1

LiteFast_Mode: Transmitter Only

License Obfuscated

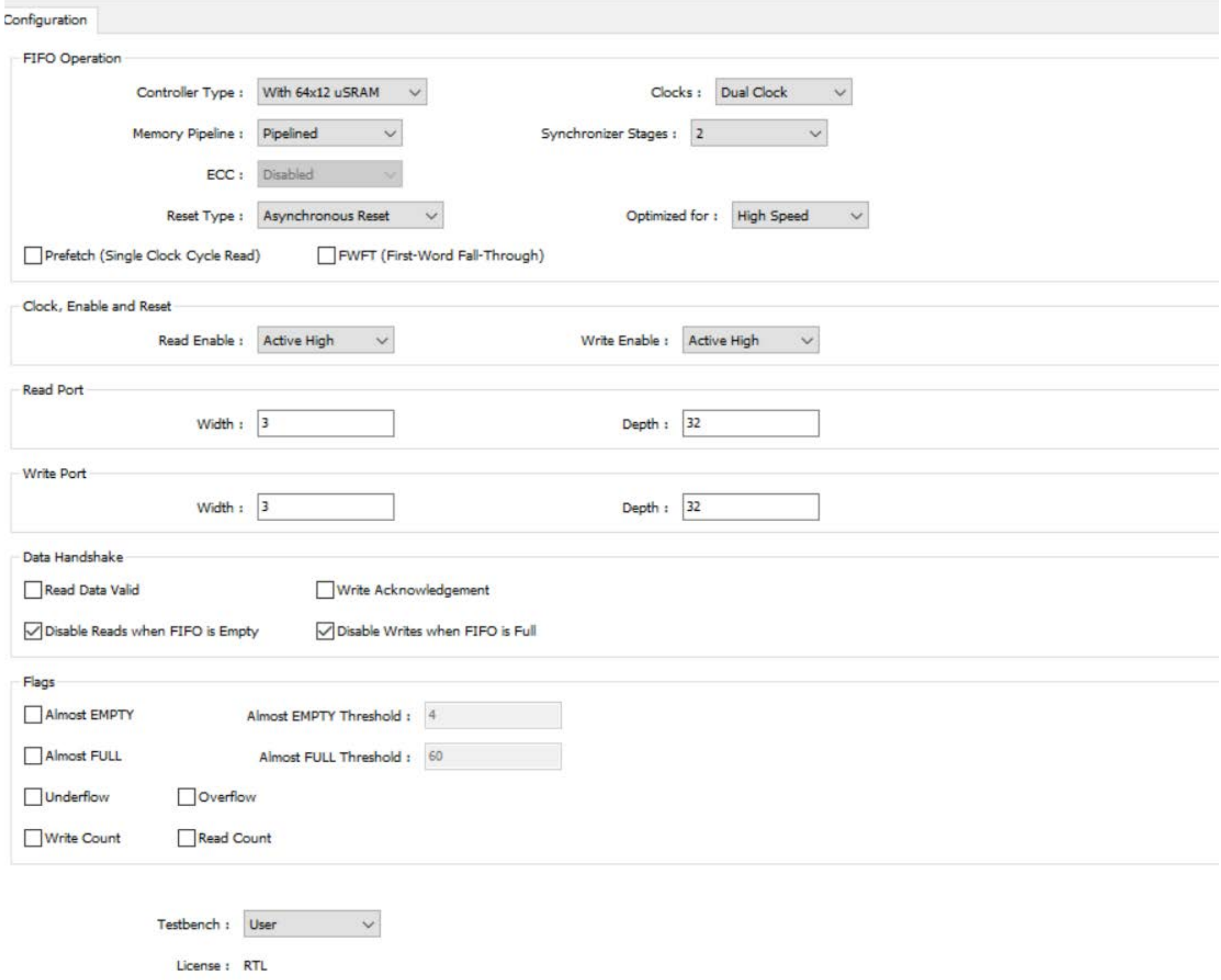
2.3.2.2.3 Synchronizer

The Synchronizer block is a two stage synchronizer, which synchronizes the clock domain crossing signals.

2.3.2.2.4 COREFIFO_C0_0

The COREFIFO_C0_0 IP is used for clock domain crossing of the crc_err_en_tx_i, payload_error_i, and start_i signals from UART_IF_CLK domain to LANE0_TX_CLK. The following figure shows the COREFIFO_C0_0 configuration. For more information about CoreFIFO, see *HB0379: CoreFIFO Handbook*.

Figure 11 • COREFIFO_C0_0 Configurator



Configuration

FIFO Operation

Controller Type : With 64x12 uSRAM

Clocks : Dual Clock

Memory Pipeline : Pipelined

Synchronizer Stages : 2

ECC : Disabled

Reset Type : Asynchronous Reset

Optimized for : High Speed

□ Prefetch (Single Clock Cycle Read)

□ FWFT (First-Word Fall-Through)

Clock, Enable and Reset

Read Enable : Active High

Write Enable : Active High

Read Port

Width : 3

Depth : 32

Write Port

Width : 3

Depth : 32

Data Handshake

□ Read Data Valid

□ Write Acknowledgement

☑ Disable Reads when FIFO is Empty

☑ Disable Writes when FIFO is Full

Flags

□ Almost EMPTY

Almost EMPTY Threshold : 4

□ Almost FULL

Almost FULL Threshold : 60

□ Underflow

□ Overflow

□ Write Count

□ Read Count

Testbench : User

License : RTL

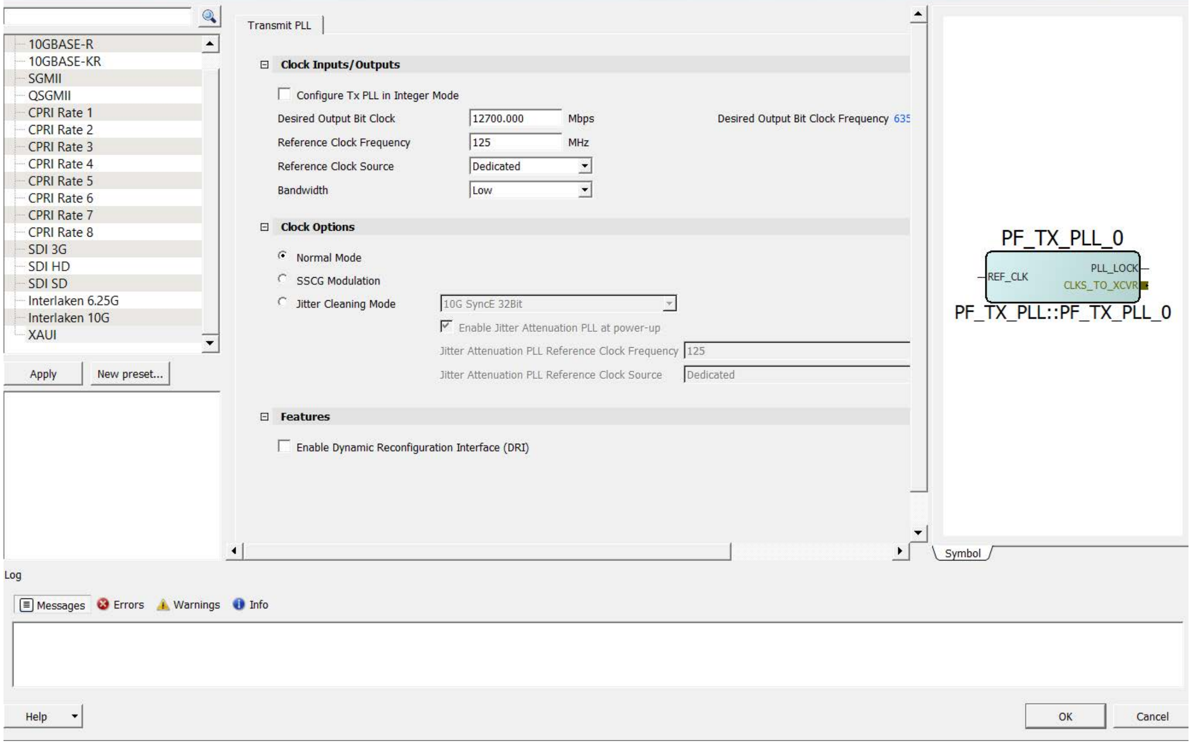
2.3.2.3 Transceiver_IF_0

This block contains the PF_TX_PLL_C0_0, PF_XCVR_REF_CLK_C0_0, and the PF_XCVR_C0_0 modules. These modules are described in the following sections.

2.3.2.3.1 PF_TX_PLL_C0_0

The PF_TX_PLL IP block provides the reference clock to the transceiver lane. This block is configured as shown in the following figure.

Figure 12 • **PF_TX_PLL Configurator**



Clock Inputs/Outputs

□ Configure Tx PLL in Integer Mode

Desired Output Bit Clock: 12700.000 Mbps — Desired Output Bit Clock Frequency 635

Reference Clock Frequency: 125 MHz

Reference Clock Source: Dedicated

Bandwidth: Low

Clock Options

● Normal Mode

○ SSCG Modulation

○ Jitter Cleaning Mode: 10G SyncE 32Bit

☑ Enable Jitter Attenuation PLL at power-up

Jitter Attenuation PLL Reference Clock Frequency: 125

Jitter Attenuation PLL Reference Clock Source: Dedicated

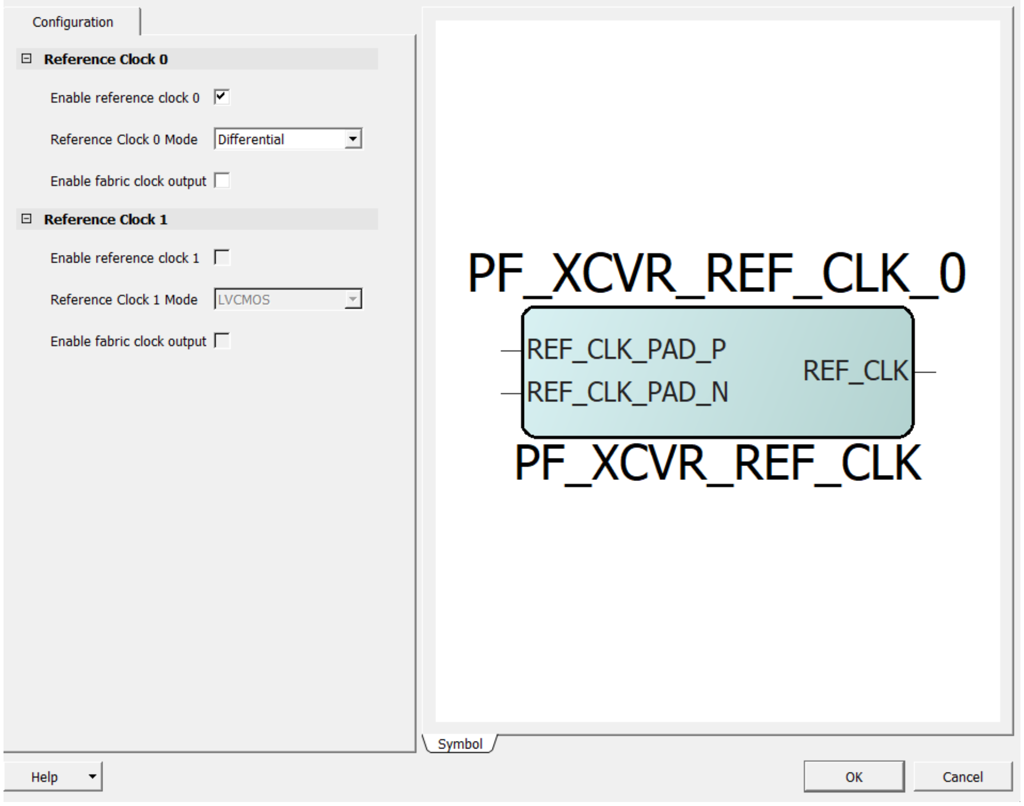
Features

□ Enable Dynamic Reconfiguration Interface (DRI)

2.3.2.3.2 PF_XCVR_REF_CLK_C0_0

The PF_XCVR_REF_CLK IP block provides the reference clock to PF_TX_PLL and CDR PLL. This block is configured as shown in the following figure.

Figure 13 • **PF_XCVR_REF_CLK Configurator**



Configuration

⊟ Reference Clock 0

Enable reference clock 0 ☑

Reference Clock 0 Mode: Differential

Enable fabric clock output ☐

⊟ Reference Clock 1

Enable reference clock 1 ☐

Reference Clock 1 Mode: LVCMOS

Enable fabric clock output ☐

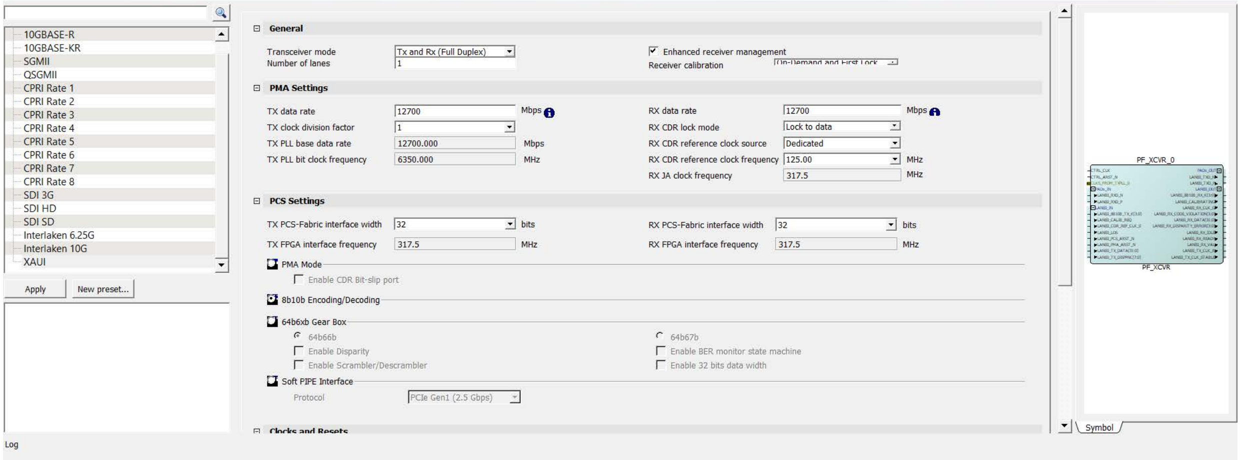
2.3.2.3.3 PF_XCVR_C0_0

The PolarFire high-speed transceiver (PF XCVR) is a hard IP block that supports high-speed data rates ranging from 250 Mbps to 12.7 Gbps. In this demo, the transceiver block (PF_XCVR) is configured in 8b10b mode on lane 0 with a CDR reference clock of 125 MHz to support 12.7 Gbps data rate.

PolarFire Transmit PLL (PF_TX_PLL) is used to send the reference clock feed to the transceiver. The dedicated reference clock (PF_XCVR_REF_CLK) drives the PF_TX_PLL to generate the desired output clock for the 12.7 Gbps data rate. For more information about the PolarFire Transceiver, see *UG0677: PolarFire FPGA Transceiver User Guide*.

The following figure shows the PF_XCVR_C0_0 configuration.

Figure 14 • **PF_XCVR_0 Configurator**



10GBASE-R
10GBASE-KR
SGMII
QSGMII
CPRI Rate 1
CPRI Rate 2
CPRI Rate 3
CPRI Rate 4
CPRI Rate 5
CPRI Rate 6
CPRI Rate 7
CPRI Rate 8
SDI 3G
SDI HD
SDI SD
Interlaken 6.25G
Interlaken 10G
XAUI

Apply | New preset...

Log

⊟ **General**

Transceiver mode: Tx and Rx (Full Duplex)
Number of lanes: 1
☑ Enhanced receiver management
Receiver calibration: On-Demand and First Lock

⊟ **PMA Settings**

TX data rate: 12700 Mbps
TX clock division factor: 1
TX PLL base data rate: 12700.000 Mbps
TX PLL bit clock frequency: 6350.000 MHz
RX data rate: 12700 Mbps
RX CDR lock mode: Lock to data
RX CDR reference clock source: Dedicated
RX CDR reference clock frequency: 125.00 MHz
RX JA clock frequency: 317.5 MHz

⊟ **PCS Settings**

TX PCS-Fabric interface width: 32 bits
TX FPGA interface frequency: 317.5 MHz
RX PCS-Fabric interface width: 32 bits
RX FPGA interface frequency: 317.5 MHz

● PMA Mode
□ Enable CDR Bit-slip port

● 8b10b Encoding/Decoding

● 64b6xb Gear Box
● 64b66b
□ Enable Disparity
□ Enable Scrambler/Descrambler
○ 64b67b
□ Enable BER monitor state machine
□ Enable 32 bits data width

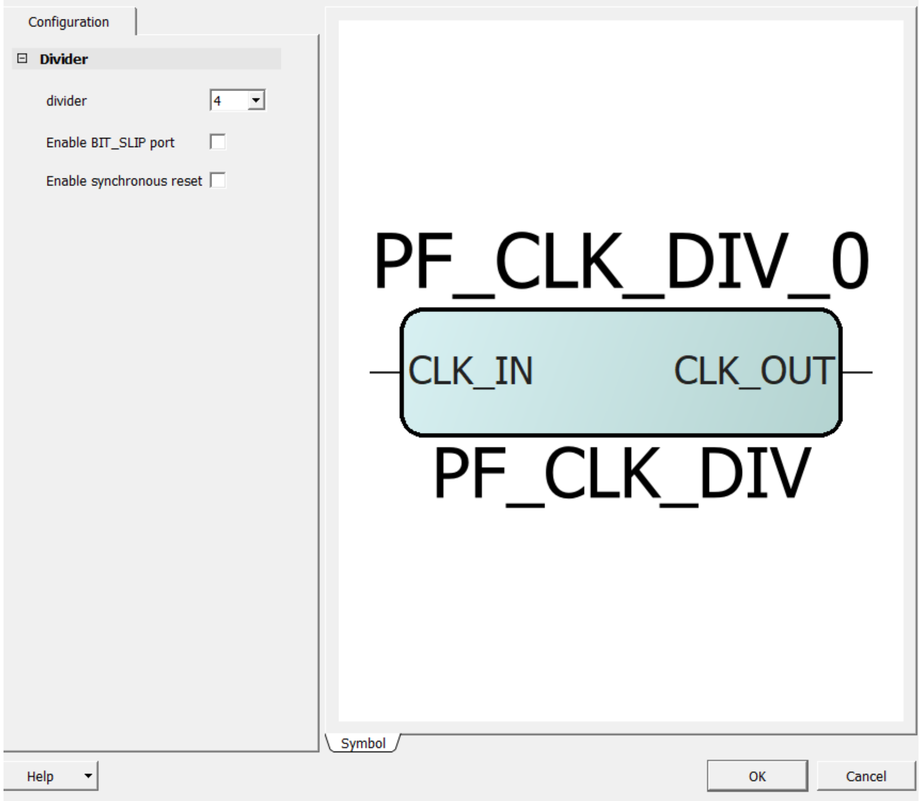
● Soft PIPE Interface
Protocol: PCIe Gen1 (2.5 Gbps)

⊟ **Clocks and Resets**

2.3.2.3.4 PF_CLK_DIV_C0_0

This block divides the input clock from PF_OSC_C0_0 by a factor of 4 and provides 40 MHz clock which is used as cntrl_clk to the PF_XCVR_C0_0.

Figure 15 • **PF_CLK_DIV_Configurator**



Configuration

⊟ **Divider**

divider: 4
Enable BIT_SLIP port □
Enable synchronous reset □

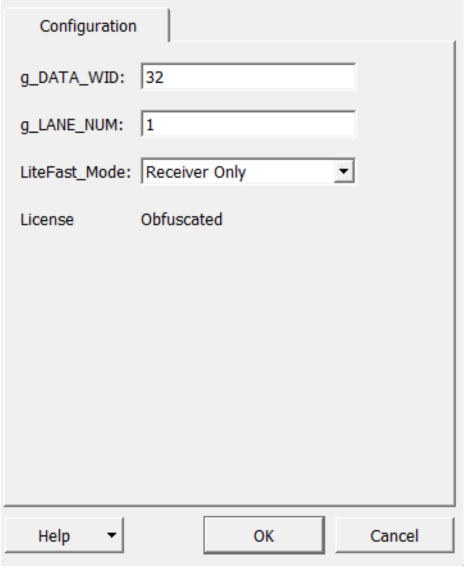
2.3.2.4 LiteFast_Receiver_0

This block contains the LiteFast_C1_0, COREFIFO, and Count_Checker_0 modules. These modules are described in the following sections.

2.3.2.4.1 LiteFast_C1_0

LiteFast_C1_0 is the instantiation of the LiteFast IP configured as receiver. This block recognizes the LiteFast frame, and extracts the user data (payload) from that frame and forwards it to the Count_Checker_0 block.

Figure 16 • **LiteFast_C1_0 Configurator**



g_DATA_WID: 32

g_LANE_NUM: 1

LiteFast_Mode: Receiver Only

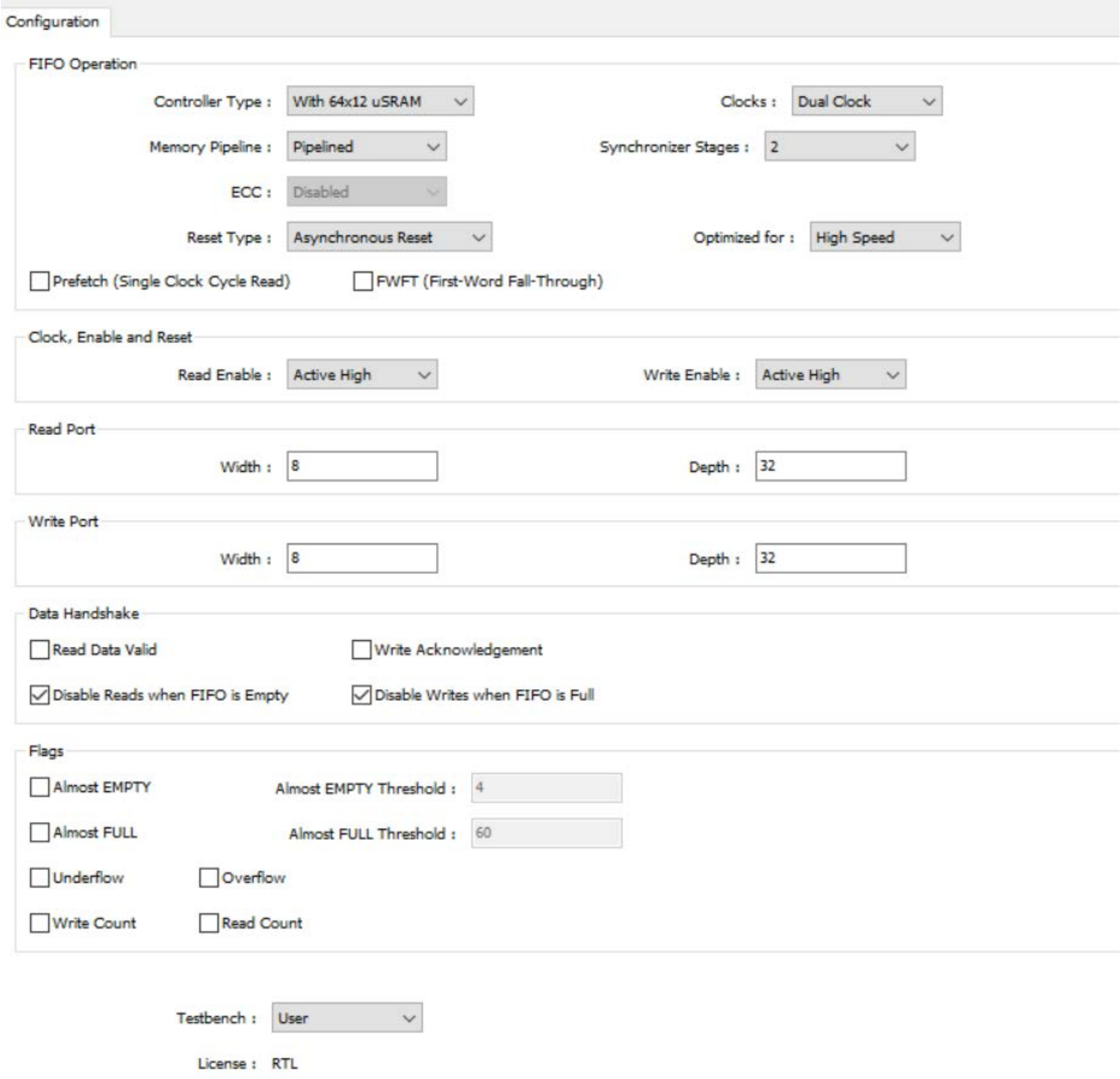
License Obfuscated

2.3.2.4.2 COREFIFO

The LiteFast_Receiver block contains the following instances of COREFIFO:

- COREFIFO_C3_0 stores the storage capacity of the LiteFast_C1_0 block instantiated on the receiving board, and sends this data to the LiteFast_C0_0 block on the transferring board for preventing excess transfer of frames.
- COREFIFO_C4_0 stores the storage capacity of the LiteFast_C1_0 block instantiated on the transferring board for preventing excess transfer of frames. The following figure shows the COREFIFO configuration.

Figure 17 • **COREFIFO_C3_0 Configurator**



Configuration

FIFO Operation

Controller Type : With 64x12 uSRAM

Clocks : Dual Clock

Memory Pipeline : Pipelined

Synchronizer Stages : 2

ECC : Disabled

Reset Type : Asynchronous Reset

Optimized for : High Speed

☐ Prefetch (Single Clock Cycle Read) ☐ FWFT (First-Word Fall-Through)

Clock, Enable and Reset

Read Enable : Active High

Write Enable : Active High

Read Port

Width : 8

Depth : 32

Write Port

Width : 8

Depth : 32

Data Handshake

☐ Read Data Valid ☐ Write Acknowledgement

☑ Disable Reads when FIFO is Empty ☑ Disable Writes when FIFO is Full

Flags

☐ Almost EMPTY Almost EMPTY Threshold : 4

☐ Almost FULL Almost FULL Threshold : 60

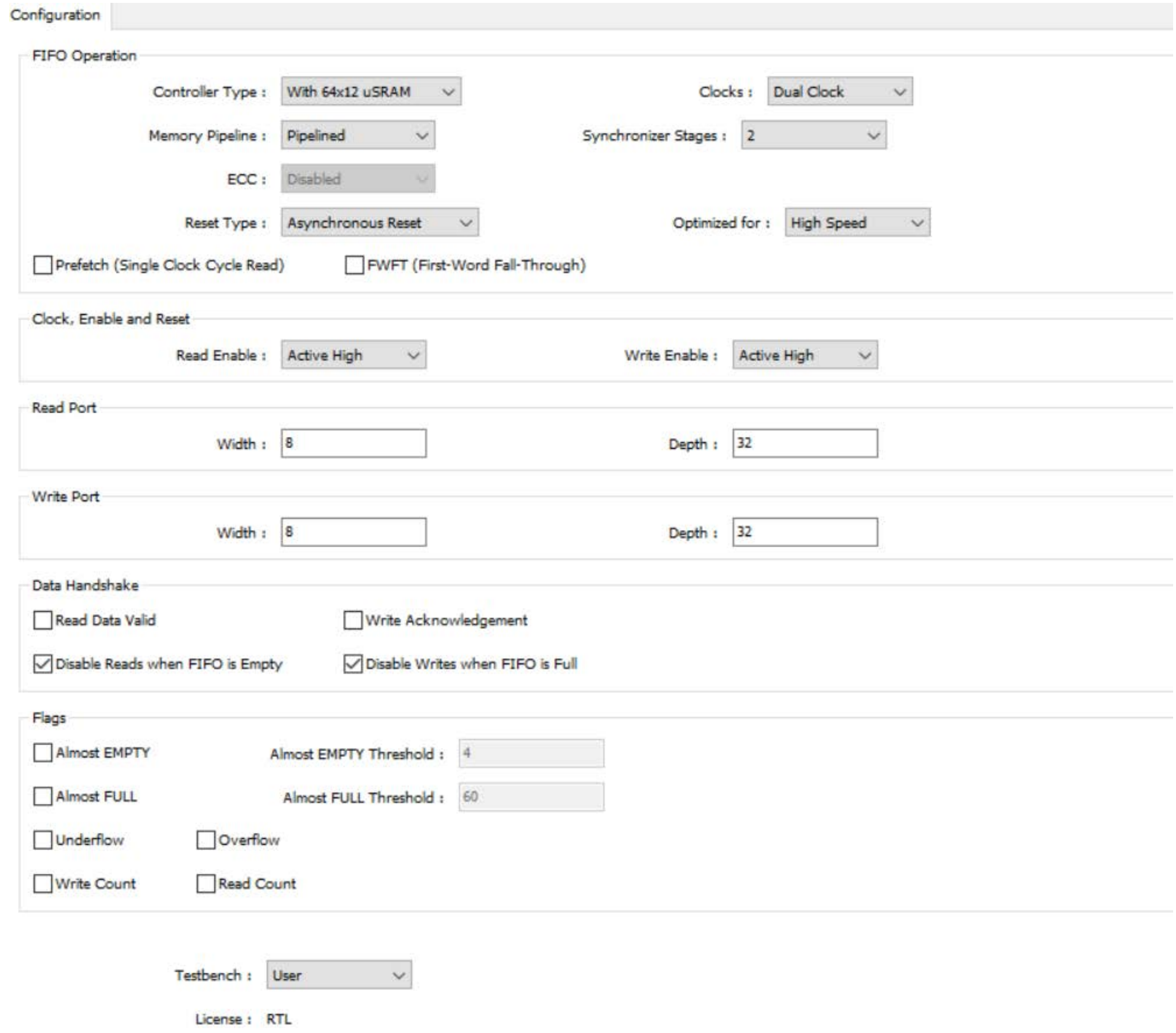
☐ Underflow ☐ Overflow

☐ Write Count ☐ Read Count

Testbench : User

License : RTL

Figure 18 • **COREFIFO_C4_0 Configurator**



Configuration

FIFO Operation

Controller Type : With 64x12 uSRAM

Clocks : Dual Clock

Memory Pipeline : Pipelined

Synchronizer Stages : 2

ECC : Disabled

Reset Type : Asynchronous Reset

Optimized for : High Speed

□ Prefetch (Single Clock Cycle Read)

□ FWFT (First-Word Fall-Through)

Clock, Enable and Reset

Read Enable : Active High

Write Enable : Active High

Read Port

Width : 8

Depth : 32

Write Port

Width : 8

Depth : 32

Data Handshake

□ Read Data Valid

□ Write Acknowledgement

☑ Disable Reads when FIFO is Empty

☑ Disable Writes when FIFO is Full

Flags

□ Almost EMPTY

Almost EMPTY Threshold : 4

□ Almost FULL

Almost FULL Threshold : 60

□ Underflow

□ Overflow

□ Write Count

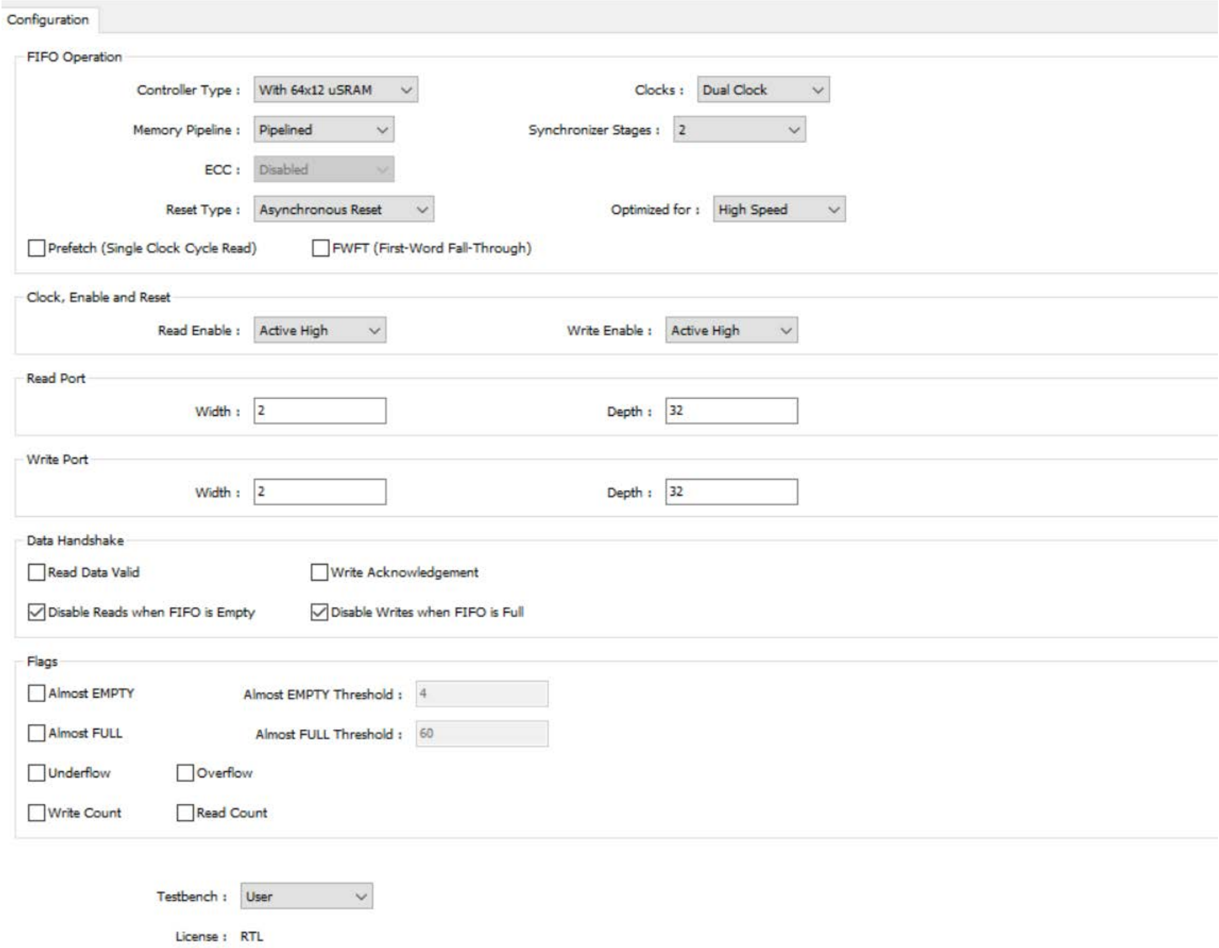
□ Read Count

Testbench : User

License : RTL

- COREFIFO_C1_0 is used for clock domain crossing of the clear_i and start_i signals from UART_IF_CLK domain to LANE0_RX_CLK. The following figure shows the COREFIFO_C1_0 configuration.

Figure 19 • **COREFIFO_C1_0 Configurator**



Configuration

FIFO Operation

Controller Type : With 64x12 uSRAM

Clocks : Dual Clock

Memory Pipeline : Pipelined

Synchronizer Stages : 2

ECC : Disabled

Reset Type : Asynchronous Reset

Optimized for : High Speed

□ Prefetch (Single Clock Cycle Read)

□ FWFT (First-Word Fall-Through)

Clock, Enable and Reset

Read Enable : Active High

Write Enable : Active High

Read Port

Width : 2

Depth : 32

Write Port

Width : 2

Depth : 32

Data Handshake

□ Read Data Valid

□ Write Acknowledgement

☑ Disable Reads when FIFO is Empty

☑ Disable Writes when FIFO is Full

Flags

□ Almost EMPTY

Almost EMPTY Threshold : 4

□ Almost FULL

Almost FULL Threshold : 60

□ Underflow

□ Overflow

□ Write Count

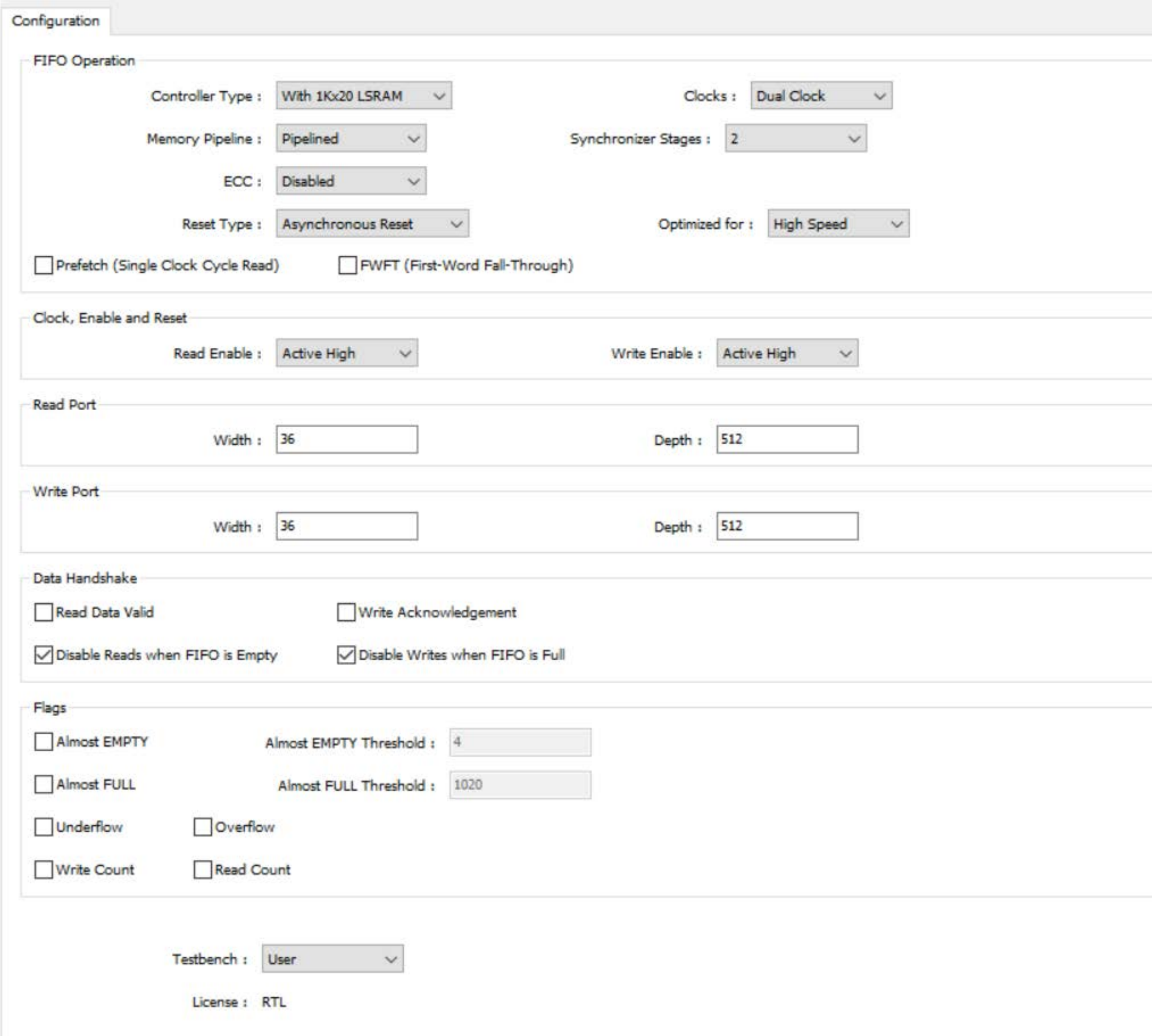
□ Read Count

Testbench : User

License : RTL

- COREFIFO_C2_0 is used for clock domain crossing of the data_tx_o bus, data_rx_o bus, crc_error_o, error_o, lock_o, and rx_val_o signals from LANE0_RX_CLK domain to UART_IF_CLK. The following figure shows the COREFIFO_C2_0 configuration.

Figure 20 • **COREFIFO_C2_0 Configurator**



Configuration

FIFO Operation

Controller Type : With 1Kx20 LSRAM

Clocks : Dual Clock

Memory Pipeline : Pipelined

Synchronizer Stages : 2

ECC : Disabled

Reset Type : Asynchronous Reset

Optimized for : High Speed

□ Prefetch (Single Clock Cycle Read)

□ FWFT (First-Word Fall-Through)

Clock, Enable and Reset

Read Enable : Active High

Write Enable : Active High

Read Port

Width : 36

Depth : 512

Write Port

Width : 36

Depth : 512

Data Handshake

□ Read Data Valid

□ Write Acknowledgement

☑ Disable Reads when FIFO is Empty

☑ Disable Writes when FIFO is Full

Flags

□ Almost EMPTY

Almost EMPTY Threshold : 4

□ Almost FULL

Almost FULL Threshold : 1020

□ Underflow

□ Overflow

□ Write Count

□ Read Count

Testbench : User

License : RTL

2.3.2.4.3 Count_Checker_0

The Count_Checker_0 block contains a 32-bit checker that checks the incoming data with the self-generated data. Error signal is asserted whenever there is a mismatch between the estimated data and captured data. It is a user-defined module.

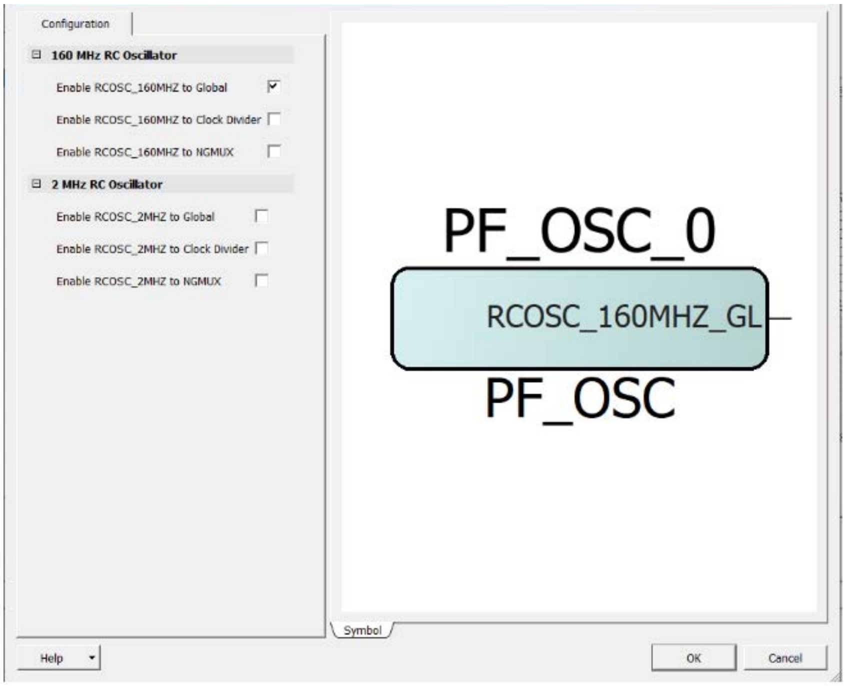
2.3.2.5 LiteFast_XCVR_Top

All the above blocks are instantiated in this module. This block also contains PF_OSC_C0_0 and PF_INIT_MONITOR_C0_0. These blocks are described in the following section.

2.3.2.5.1 PF_OSC_C0_0

This on-chip oscillator provides a clock to the PF_CCC_C0_0 block in the UART_IF and to the PF_CLK_DIV_C0_0 in the Transceiver_IF. In the reference design, PF_OSC_C0_0 is configured to provide a 160 MHz clock through a global buffer, as shown in the following figure.

Figure 21 • **PF_OSC Configurator**



Configuration

⊟ 160 MHz RC Oscillator

Enable RCOSC_160MHZ to Global ☑

Enable RCOSC_160MHZ to Clock Divider ☐

Enable RCOSC_160MHZ to NGMUX ☐

⊟ 2 MHz RC Oscillator

Enable RCOSC_2MHZ to Global ☐

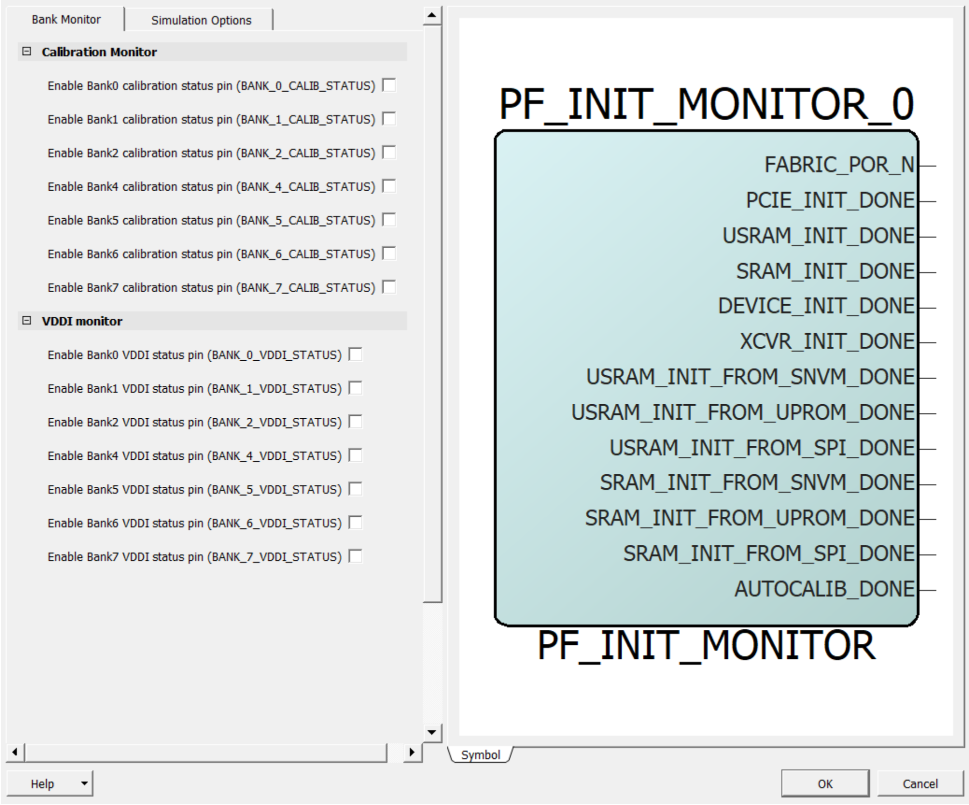
Enable RCOSC_2MHZ to Clock Divider ☐

Enable RCOSC_2MHZ to NGMUX ☐

2.3.2.5.2 PF_INIT_MONITOR_C0_0

This initialization monitor asserts the DEVICE_INIT_DONE signal when the device initialization is done, and this signal is used to reset the transceiver.

Figure 22 • **PolarFire Initialization Monitor Configurator**



Calibration Monitor

- Enable Bank0 calibration status pin (BANK_0_CALIB_STATUS) □
- Enable Bank1 calibration status pin (BANK_1_CALIB_STATUS) □
- Enable Bank2 calibration status pin (BANK_2_CALIB_STATUS) □
- Enable Bank4 calibration status pin (BANK_4_CALIB_STATUS) □
- Enable Bank5 calibration status pin (BANK_5_CALIB_STATUS) □
- Enable Bank6 calibration status pin (BANK_6_CALIB_STATUS) □
- Enable Bank7 calibration status pin (BANK_7_CALIB_STATUS) □

VDDI monitor

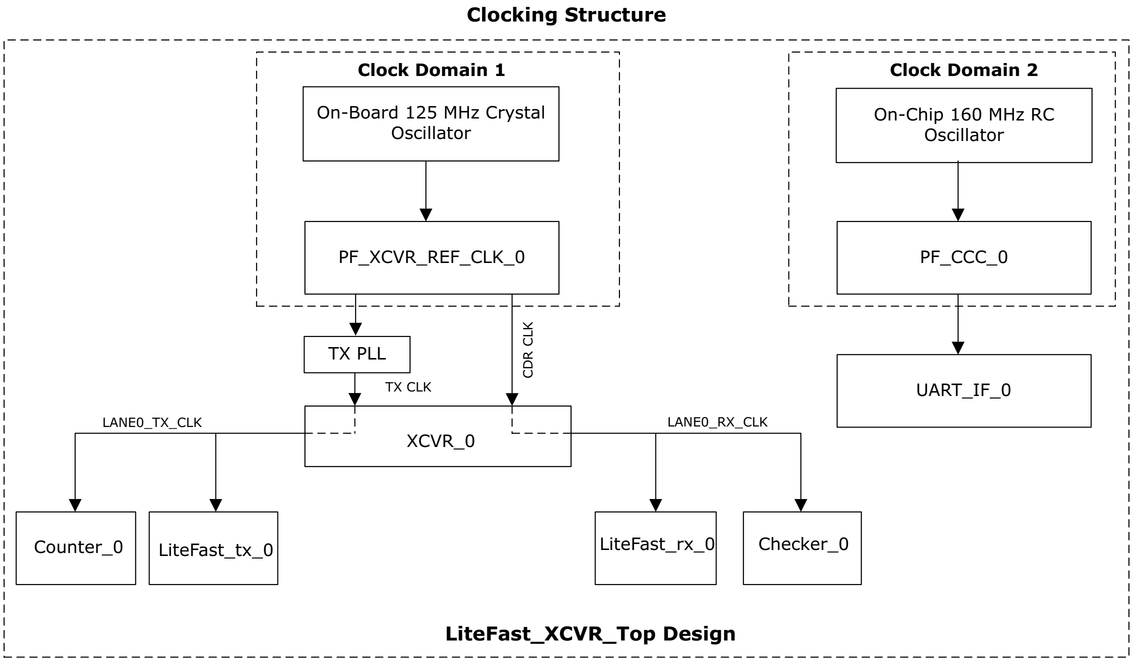
- Enable Bank0 VDDI status pin (BANK_0_VDDI_STATUS) □
- Enable Bank1 VDDI status pin (BANK_1_VDDI_STATUS) □
- Enable Bank2 VDDI status pin (BANK_2_VDDI_STATUS) □
- Enable Bank4 VDDI status pin (BANK_4_VDDI_STATUS) □
- Enable Bank5 VDDI status pin (BANK_5_VDDI_STATUS) □
- Enable Bank6 VDDI status pin (BANK_6_VDDI_STATUS) □
- Enable Bank7 VDDI status pin (BANK_7_VDDI_STATUS) □

2.4 Clocking Structure

In the reference design, there are two clock domains. The on-board 125 MHz crystal oscillator drives the XCVR reference clock, which provides clock source to the Counter_0, LiteFast_tx_0, XCVR_0, LiteFast_rx_0, and Checker_0 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 23 • **Clocking Structure**



2.5 Simulating the Design

Before you start:

1. Start Libero SoC PolarFire and select **Project** -> **Tool Profiles...**.
2. In the **Tool Profiles** window, select **Synthesis** and **Simulation** on the **Tools** panes and select the latest active installation directory paths for these two tools.
3. In the **Project** menu, click **Open Project**.
   The **Open Project** dialog box opens.
4. Browse the `mpf_dg0783_df\Libero_Project\PF_LiteFast_8b_10b` design files folder and select the `PF_LiteFast_8b_10b` PRJX file. Then, click **Open**.
   The PolarFire LiteFast project opens in Libero SoC PolarFire.
5. Download the following IP cores from **Libero SoC PolarFire**->**Catalog**:
   - LiteFast
   - PF_XCVR
   - PF_TX_PLL
   - PF_XCVR_REF_CLK
   - COREUART

The following figure shows the interaction between testbench and the design. Table 3 lists the simulation signals.

Figure 24 • **Testbench and LiteFast Reference Design Interaction**

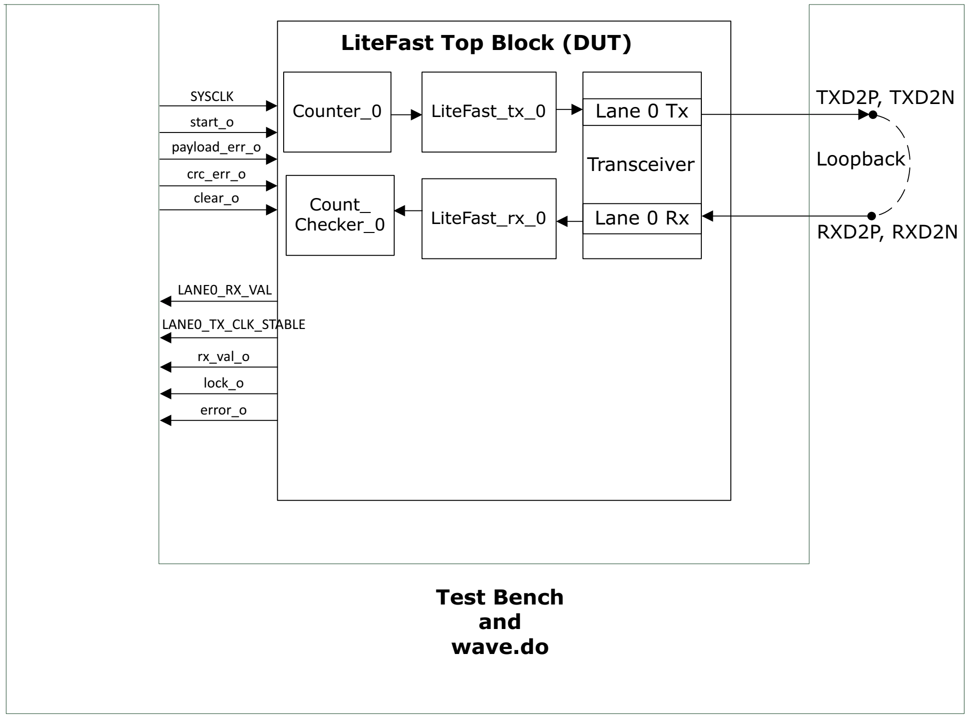
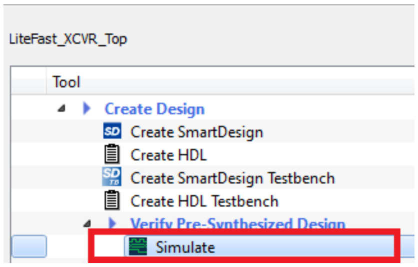


Table 3 • **Simulation Signals**

Signal	Description
From Testbench to DUT	
SYSCLK	125 MHz system clock
start_o	This signal starts the Counter and Checker
payload_err_o	This signal injects payload error in the Counter generated data
crc_err_o	This signal injects CRC error in the LiteFast frames generated by the LiteFast_tx_0 module
clear_o	This signal disables payload error and CRC error
From DUT to Testbench	
data_out_o	This is the Counter generated data passed to LiteFast_tx_0
LANE0_RX_VAL	This output signal of the PF_XCVR indicates that the XCVR has received and validated the LiteFast idle frames
LANE0_TX_CLK_STABLE	This output signal of the PF_XCVR indicates that the LANE0_TX_CLK is locked to transmitter frequency (TX PLL)
LANE0_TX_DATA	This data bus transmits LiteFast frames to Transceiver
LANE0_RX_DATA	This data bus receives LiteFast frames from Transceiver
rx_val_o	This is the registered version of LANE0_RX_VAL signal in LANE0_RX_CLK domain
lock_o	This signal asserts high when the counter and checker data match
error_o	This signal asserts high when the counter and checker data mismatch
data_tx_o	This output bus of the LiteFast_rx_0 contains the de-framed data passed to the Count_Checker_0
data_rx_o	This is the Count_Checker_0 generated data for comparing with the data_tx_o

In the **Design Flow** tab, double-click **Simulate** under **Verify Pre-Synthesized Design** to simulate the design. The **Simulate** option is highlighted in the following figure.

Figure 25 • **Simulating the Design**



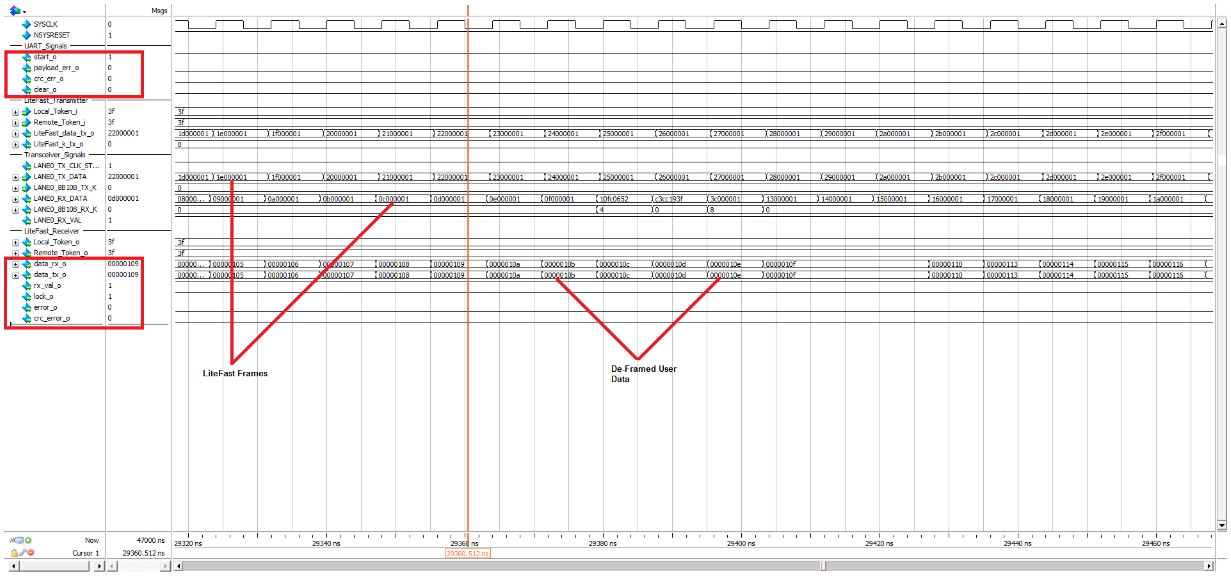
When the Simulation is initiated, ModelSim compiles all the design source files, testbench, and the stimulus, and launches the waveform window to show the simulation signals.

2.5.1 Simulation Flow

The following steps describe the LiteFast testbench simulation flow:

1. At 0 ns, the testbench drives the 125 MHz system clock to the DUT.
2. At ~172 ns, the testbench asserts the LANE0_TX_CLK_STABLE signal. This indicates that PF_TX_PLL in the DUT has locked to 317.5 MHz.
3. The LiteFast IP sends IDLE character (K28.5) to XCVR.
4. The K28.5 character (0x000000BC) is looped back from TX lane to RX lane.
5. The receiver PLL locks (LANE0_RX_CLK_R) to 317.5 MHz and then, the K28.5 character asserts the LANE0_RX_VAL signal high at ~26939 ns.
6. The Counter_0, LiteFast_tx_0, LiteFast_rx_0, and Count_Checker_0 come out of reset because of the LANE0_RX_VAL assertion.
7. At 27000 ns, the testbench drives the start_o (UART_IF_0 output) signal high.
8. The Counter_0 starts sending incremental data starting from 0 to LiteFast_tx_0, which sends that data to PF_XCVR_0.
9. The LiteFast_rx_0 receives the data from the external loopback and the Count_Checker_0 compares the received data. Figure 26, page 26 shows the simulation waveform with no errors.

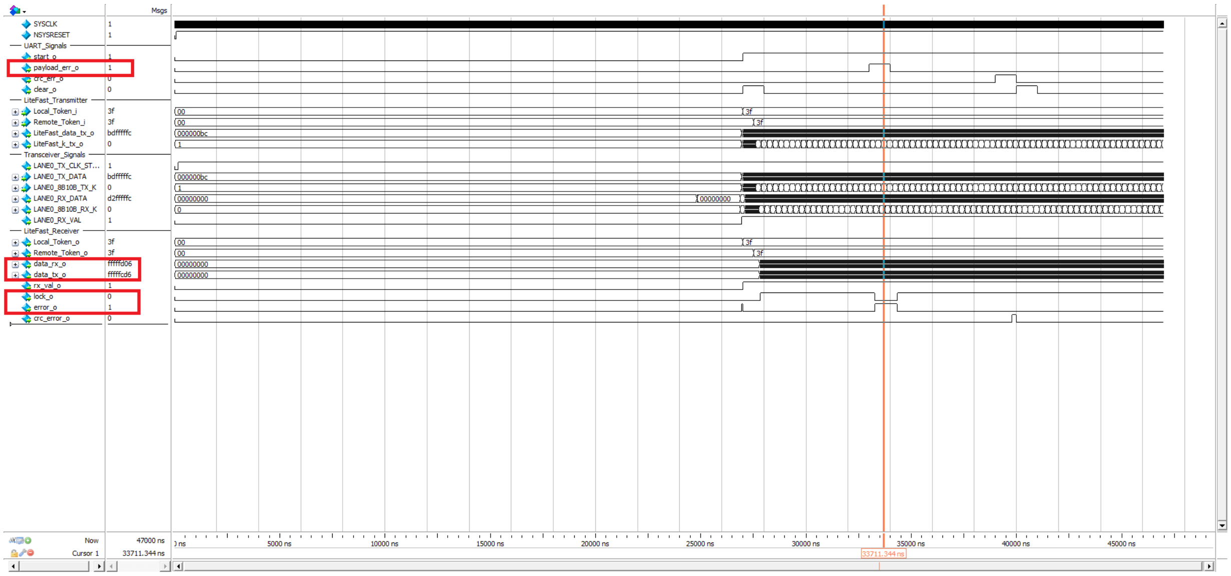
Figure 26 • **Simulation Waveform with No Errors**



	Msgs
SYSCLK	0
NSYSRESET	1
UART_Signals	
start_o	1
payload_err_o	0
crc_err_o	0
clear_o	0
LiteFast_Transmitter	
Local_Token_i	3f
Remote_Token_i	3f
LiteFast_data_tx_o	22000001
LiteFast_k_tx_o	0
Transceiver_Signals	
LANE0_TX_CLK_ST...	1
LANE0_TX_DATA	22000001
LANE0_8B10B_TX_K	0
LANE0_RX_DATA	0d000001
LANE0_8B10B_RX_K	0
LANE0_RX_VAL	1
LiteFast_Receiver	
Local_Token_o	3f
Remote_Token_o	3f
data_rx_o	00000109
data_tx_o	00000109
rx_val_o	1
lock_o	1
error_o	0
crc_error_o	0

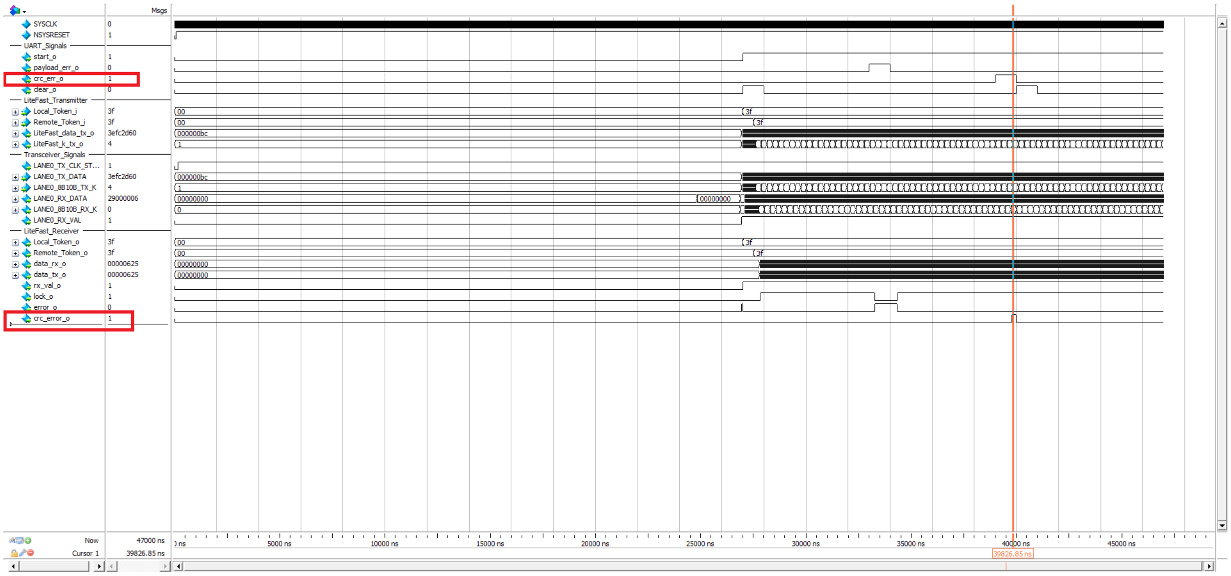
10. At 33000 ns, the testbench drives the payload_err_o high. As a result, lock_o goes low and error_o goes high, which indicates that the Counter_0 generated data does not match the Count_Checker_0 generated data. The following figure shows these signals.

Figure 27 • Simulation Waveform With Payload Error



11. At 34000 ns, the testbench drives the payload_err_o signal low.
12. At 39000 ns, the testbench drives the crc_err_o high for simulating the CRC error. As a result, the crc_error_o signal goes high, which indicates the presence of CRC checksum error. The following figure shows these signals.

Figure 28 • Simulation Waveform With CRC Error



13. At 40000 ns, the testbench drives the crc_err_o signal low and the clear pulse high, which clears all the errors.

Note: The data_rx_o, data_tx_o, rx_val_o, lock_o, error_o, and crc_error_o signals are uninitialized when COREFIFO_3 is in reset. As a result, these uninitialized states appear as glitches in the waveform. This behavior does not impact the design.

This concludes the testbench simulation flow of the LiteFast reference design.

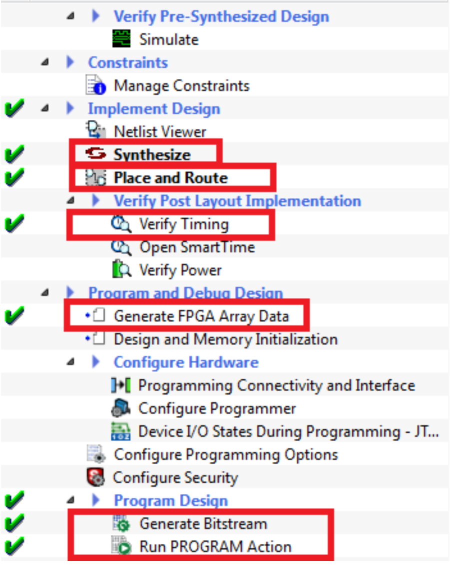
3 Libero Design Flow

This chapter describes the Libero design flow, which involves the following steps:

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

The following figure shows these options in the **Design Flow** tab.

Figure 29 • **Libero Design Flow Options**



- Verify Pre-Synthesized Design
  - Simulate
- Constraints
  - Manage Constraints
- ✓ Implement Design
  - Netlist Viewer
  - ✓ **Synthesize**
  - ✓ **Place and Route**
  - Verify Post Layout Implementation
    - ✓ Verify Timing
    - Open SmartTime
    - Verify Power
- Program and Debug Design
  - ✓ Generate FPGA Array Data
  - Design and Memory Initialization
  - Configure Hardware
    - Programming Connectivity and Interface
    - Configure Programmer
    - Device I/O States During Programming - JT...
  - Configure Programming Options
  - Configure Security
  - ✓ Program Design
    - ✓ Generate Bitstream
    - ✓ Run PROGRAM Action

3.1 Synthesize

To synthesize the design:

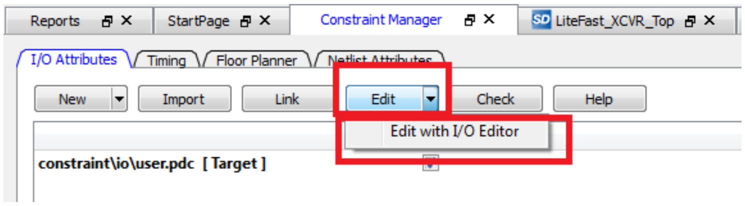
1. Double-click **Synthesize** from the **Design Flow** tab.
   When the synthesis is successful, a green tick mark appears as shown in Figure 29, page 28.
2. Right-click **Synthesize** and select **View Report** to view the synthesis report and log files in the **Reports** tab.
   We recommend viewing the `LiteFast_XCVR_Top.srr` and the `LiteFast_XCVR_Top_compile_netlist.log` files for debugging synthesis and compile errors.

3.2 Place and Route

To place and route the design:

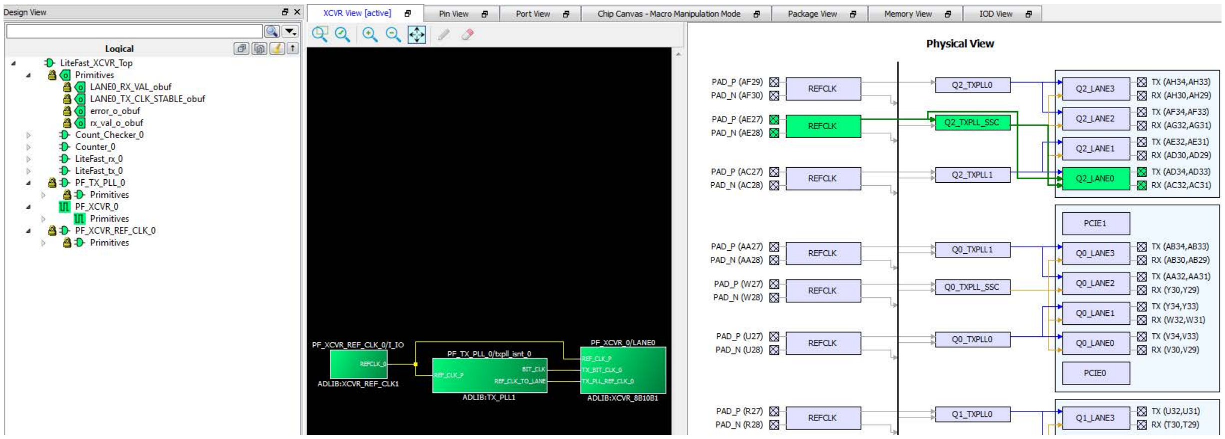
1. From **Design Flow,** double-click **Manage Constraints** and click the **Edit with I/O Editor** option from the **I/O Attributes** tab as shown in the following figure.

Figure 30 • **Starting I/O Editor**



2. Place TX_PLL, XCVR_REF_CLK, and PF_XCVR TX and RX lane using the **I/O Editor** as shown in the following figure.

Figure 31 • **I/O Editor XCVR View**



3. Double-click **Place and Route** from the **Design Flow** tab.
   When place and route is successful, a green tick mark appears as shown in Figure 29, page 28.
4. Right-click **Place and Route** and select **View Report** to view the place and route report and log files in the **Reports** tab.
   We recommend viewing the `LiteFast_XCVR_Top_place_and_route_constraint_coverage.xml` file for place and route constraint coverage.

3.2.1 Resource Utilization

The resource utilization report is written to the `LiteFast_XCVR_Top_layout_log.log` file in the **Reports** tab -> **LiteFast_XCVR_Top report** -> **Place and Route**. The following table lists the resource utilization of the design after place and route. These values may vary slightly for different Libero runs, settings, and seed values.

Table 4 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	1961	299544	0.65
DFF	1586	299544	0.53
I/O Register	0	1530	0.00
Logic Element	2314	299544	0.77

3.3 Verify Timing

To verify timing:

1. Double-click **Verify Timing** from the **Design Flow** tab.
   When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 29, page 28.
2. Right-click **Verify Timing** and select **View Report** to view the verify timing report and log files in the **Reports** tab.

3.4 Generate FPGA Array Data

Double-click **Generate FPGA Array Data** from the Design Flow window.

A green tick mark is displayed after the successful generation of the FPGA array data as shown in Figure 29, page 28.

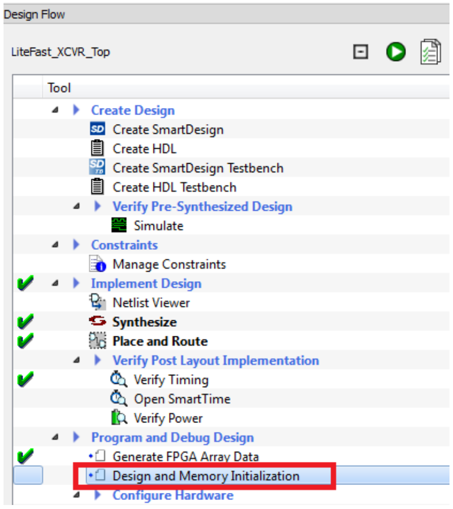
3.5 Design and Memory Initialization

This option is used to create the non-PCIe XCVR initialization client, which is used in the reference design. When the PolarFire device powers up, the transceiver block is initialized by the initialization client generated during the design and memory initialization stage in the design flow.

To create the XCVR initialization client:

1. Double-click **Design and Memory Initialization** from the **Design Flow** window. The Design and Memory Initialization window opens as shown in the following figure.

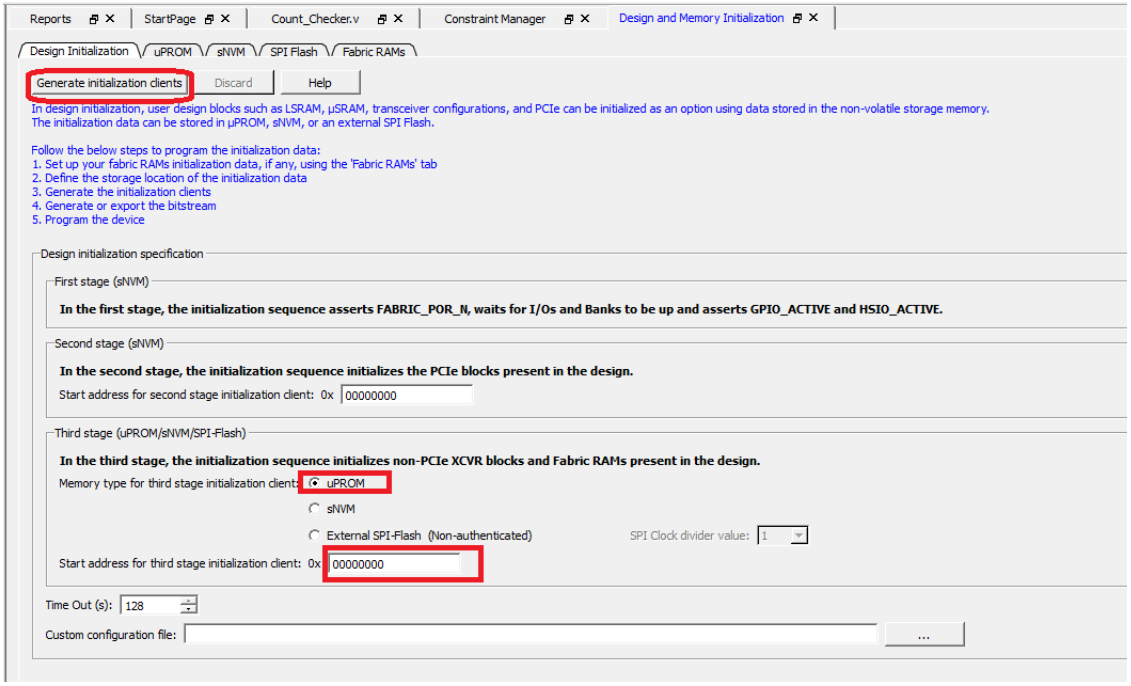
Figure 32 • **Design and Memory Initialization**



2. In the Third Stage pane, select uPROM as the non-volatile memory, retain the default start address (0x00000000), and then click **Generate initialization clients** as shown in the following figure.

Note: The default start address 0x00000000 is retained because there are no other initialization clients specified in the uPROM.

Figure 33 • **Generating XCVR Initialization Client**



In design initialization, user design blocks such as LSRAM, µSRAM, transceiver configurations, and PCIe can be initialized as an option using data stored in the non-volatile storage memory.
The initialization data can be stored in µPROM, sNVM, or an external SPI Flash.

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

Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence asserts FABRIC_POR_N, waits for I/Os and Banks to be up and asserts GPIO_ACTIVE and HSIO_ACTIVE.

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (uPROM/sNVM/SPI-Flash)

In the third stage, the initialization sequence initializes non-PCIe XCVR blocks and Fabric RAMs present in the design.

Memory type for third stage initialization client: ● uPROM
○ sNVM
○ External SPI-Flash (Non-authenticated) SPI Clock divider value: 1

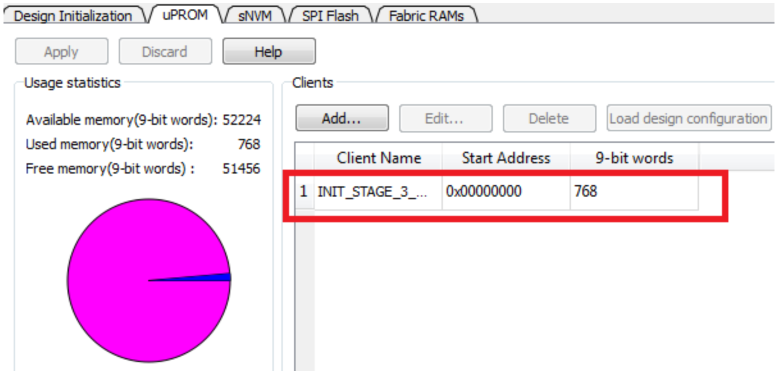
Start address for third stage initialization client: 0x 00000000

Time Out (s): 128

Custom configuration file:

3. The XCVR initialization client is created in the uPROM tab as shown in the following figure.

Figure 34 • **XCVR Initialization Client Created**



Available memory(9-bit words): 52224
Used memory(9-bit words): 768
Free memory(9-bit words) : 51456

	Client Name	Start Address	9-bit words
1	INIT_STAGE_3_...	0x00000000	768

The generation of XCVR client was successful. When the device is programmed the XCVR initialization client is written to the uPROM.

3.6 Generate Bitstream

To generate the bitstream:

1. Double-click **Generate Bitstream** from the **Design Flow** tab.
   When the bitstream is successfully generated, a green tick mark appears as shown in Figure 29, page 28.
2. Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.

3.7 Run PROGRAM Action

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

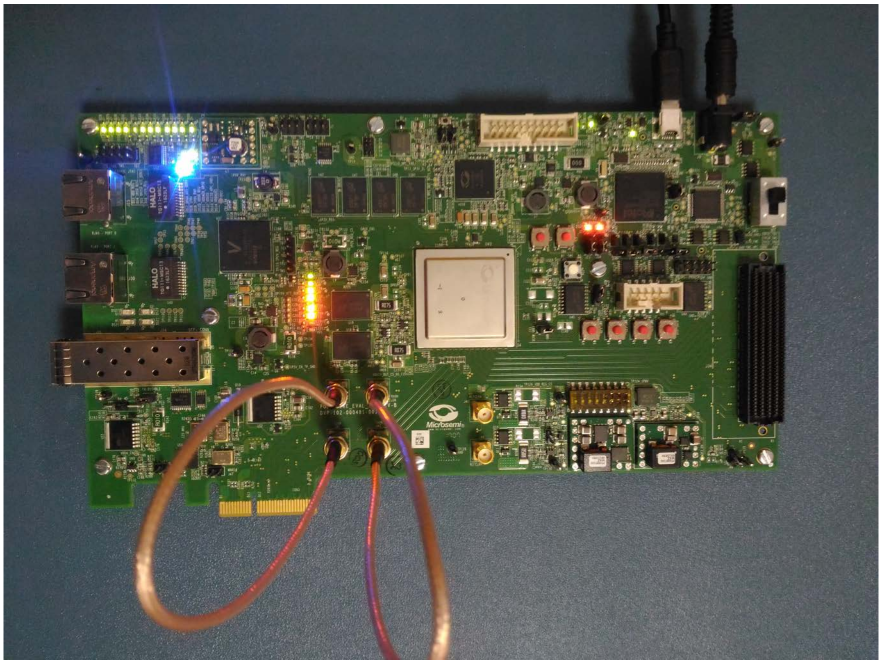
1. Ensure that the jumper settings on the board are same as listed in the following table.

Table 5 • **Jumper Settings**

Jumper	Description	Default
J18, J19, J20, J21, J22	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI	Closed
J28	Close pin 2 and 3 for programming through the on-board FlashPro5	Open
J26	Close pin 1 and 2 for programming through the FTDI SPI	Closed
J27	Close pin 1 and 2 for programming through the FTDI SPI	Closed
J4	Close pin 1 and 2 for manual power switching using SW3	Closed
J12	Close pin 3 and 4 for 2.5 V	Closed
J46	Close pin 1 and 2 for setting the Reference Clock to 125 MHz on board oscillator	Closed

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

Figure 35 • **Board Setup**



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

Right-click **Run Program Action** and select **View Report** to view the corresponding log file in the **Reports** tab.

When the device is successfully programmed, a green tick mark appears as shown in Figure 29, page 28. See Running the Demo, page 33 to run the LiteFast standalone demo.

4 Running the Demo

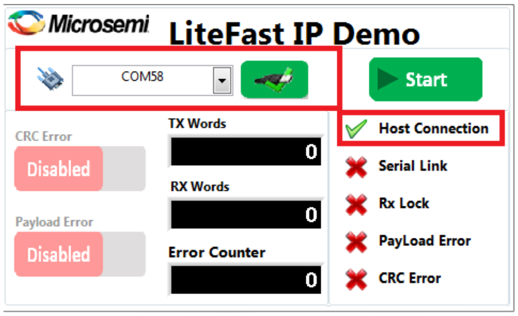
This chapter describes how to install and use the GUI to run the LiteFast demo. The following procedure assumes that:

1. The PolarFire Evaluation board is connected
2. The PolarFire FPGA is programmed with the LiteFast design

To run the LiteFast demo:

1. Extract the contents of the `mpf_dg0783_df.zip` file.
2. From the `mpf_dg0783_df\GUI_Installer` folder, double-click the `setup.exe` file.
3. Follow the instructions displayed on the installation wizard.
4. After successful installation, LiteFast_IP_Demo appears on the Start menu of the host PC desktop.
5. From the Start menu, click LiteFast_IP_Demo.
6. Double-click the LiteFast_GUI application from the installation directory to start the GUI application.
7. The GUI detects the COM port number and automatically connects to the board, as shown in the following figure. Port numbers may vary.
   The Host Connection status changes to a green tick mark as shown in the following figure.

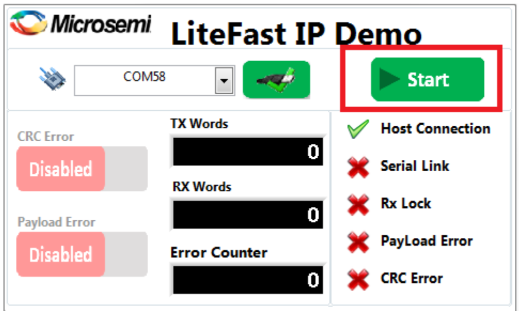
Figure 36 • **Detecting the COM Port and Host Connection Status**



[

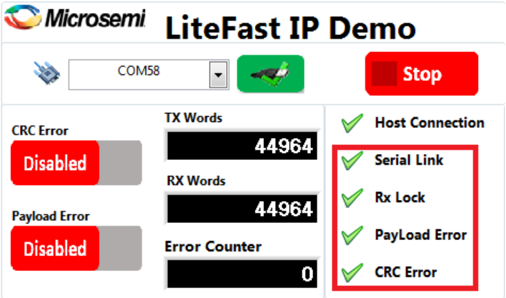
8. Click the **Start** button to start the LiteFast demo.

Figure 37 • **Starting the LiteFast Demo**



The Serial Link, Rx Lock, PayLoad Error, and CRC Error changes to a green tick mark as shown in the following figure.

Figure 38 • **Overall Status**



COM58

CRC Error: Disabled

Payload Error: Disabled

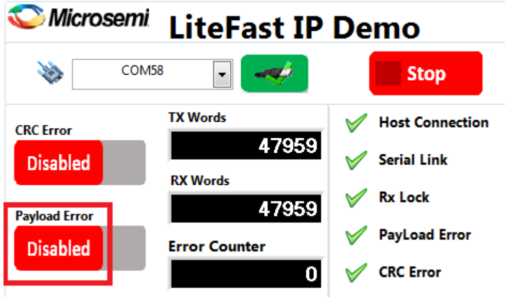
TX Words: 44964

RX Words: 44964

Error Counter: 0

9. Click the Disabled button in the **Payload Error** area as shown in the following figure to inject payload error in the Counter generated data.

Figure 39 • **Enabling Payload Error**



COM58

CRC Error: Disabled

Payload Error: Disabled

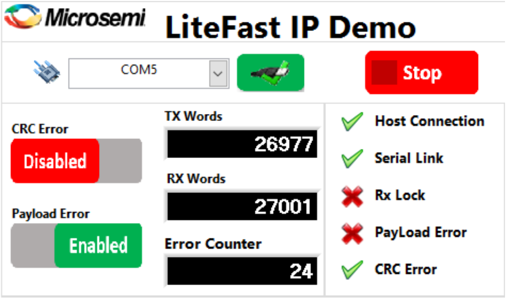
TX Words: 47959

RX Words: 47959

Error Counter: 0

The Rx Lock and Payload Error status changes to a red cross mark as shown in the following figure.

Figure 40 • **Checking Payload Error Status**



COM5

CRC Error: Disabled

Payload Error: Enabled

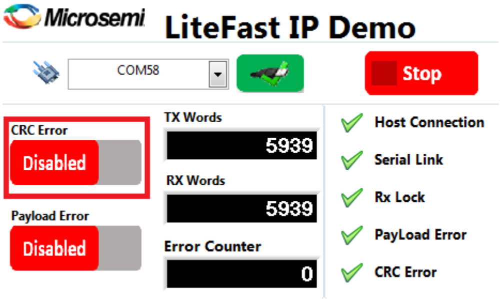
TX Words: 26977

RX Words: 27001

Error Counter: 24

10. Click Disabled button in the **CRC Error** area as shown in the following figure to inject CRC error in the Counter generated data.

Figure 41 • **Enabling CRC Error**



COM58

CRC Error: Disabled

Payload Error: Disabled

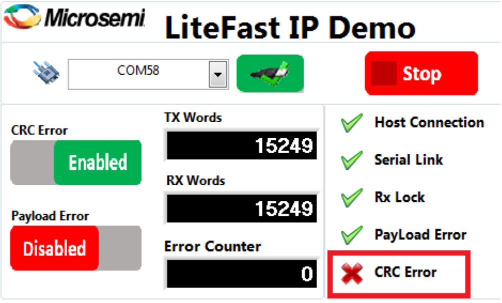
TX Words: 5939

RX Words: 5939

Error Counter: 0

The CRC Error status changes to a red cross mark as shown in the following figure.

Figure 42 • **Checking CRC Error Status**



11. **TX Words** displays the number of transmitted data words. This number rolls over after 65535 words.
12. **RX Words** displays the number of received data words. This number rolls over after 65535 words.
13. **Error Counter** displays the packet loss.

Note: Error counter may not exactly co-relate to TX words and RX words due frequency gap between LiteFast transmitter IP and the CoreUART IP.

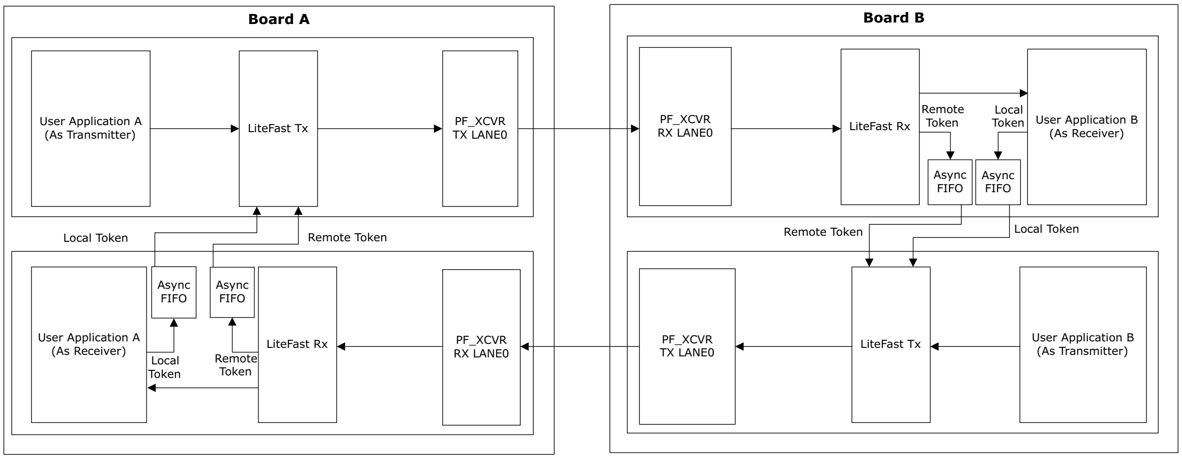
This concludes running the LiteFast IP Demo.

5 Using LiteFast For Board-to-Board Data Transfer

This section describes how to implement the LiteFast IP for board-to-board data transfer.

Suppose, there are two Microsemi boards (Board A and B) running different applications (Application A and B). The following figure outlines the LiteFast transmitter and receiver designs in Board A and B for implementing a full-duplex data transfer.

Figure 43 • **Data Transfer Between Board A and B**



Note: The depth of the FIFO must be defined based on the user application.

The following steps describe how to implement full-duplex data transfer between board A and B:

1. Program the reference design on both board A and B.
2. Connect TXDP and TXDN of board A to RXDP and RXDN of board B using two SMA cables.
3. Connect TXDP and TXDN of board B to RXDP and RXDN of board A using two SMA cables.
4. Install the LiteFast GUI on two separate Host PCs (A and B).
5. Connect Host PC A to board A and Host PC B to board B using USB cables.
6. Power-up board A and B.
7. Start the LiteFast GUI on Host PC A and B.
8. Click **Start** on both the instances of the GUI running on Host PC A and B.
   The number of TX Words displayed on the GUI running on Host PC A matches the number of RX Words displayed on the GUI running on Host PC B. Similarly, the number of TX Words displayed on the GUI running on Host PC B matches the number of RX Words displayed on the GUI running on Host PC A.

This concludes the full-duplex data transfer demo.

6 Appendix 1: Programming the Device Using FlashPro Express

This chapter 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:

`mpf_dg0783_df\Programming_File`

Follow these steps:

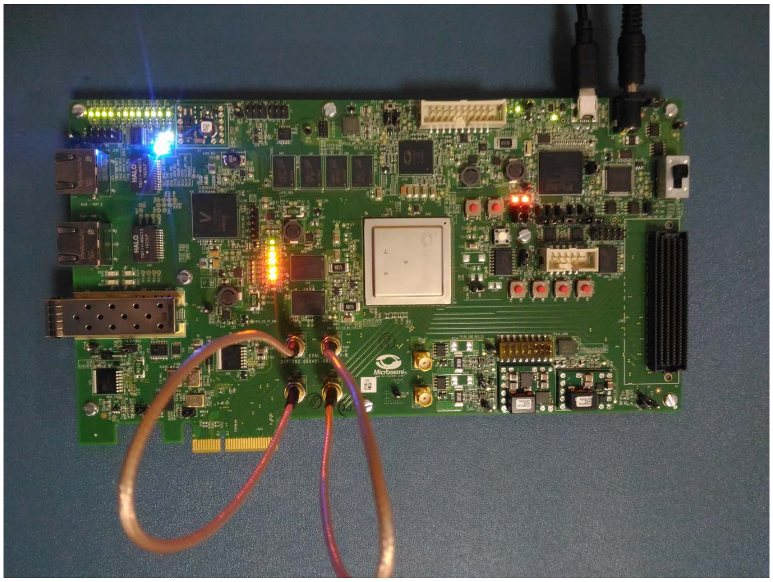
1. Ensure that the jumper settings on the board are same as listed in the following table.

Table 6 • **Jumper Settings**

Jumper	Description	Default
J18, J19, J20, J21, J22	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI	Closed
J28	Close pin 2 and 3 for programming through the on-board FlashPro5	Open
J26	Close pin 1 and 2 for programming through the FTDI SPI	Closed
J27	Close pin 1 and 2 for programming through the FTDI SPI	Closed
J4	Close pin 1 and 2 for manual power switching using SW3	Closed
J12	Close pin 3 and 4 for 2.5 V	Closed
J46	Close pin 1 and 2 for setting the Reference Clock to 125 MHz on board oscillator	Closed

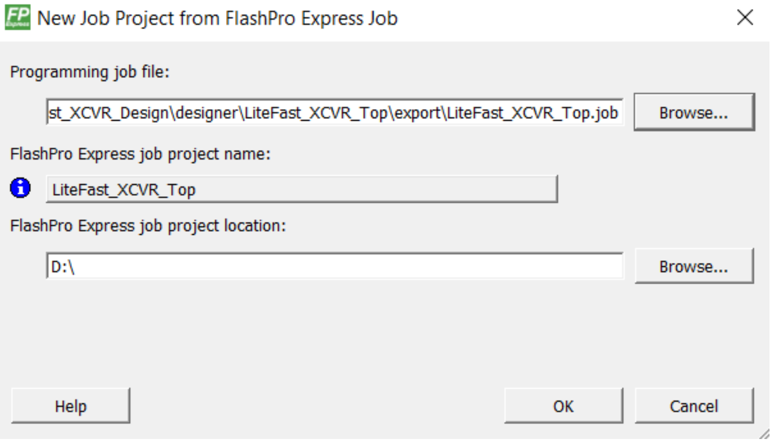
2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the Host PC to **J5** (FTDI port) on the board.
4. Power on the board using the **SW3** slide switch.
5. Connect **TXN** to **RXN** and **TXP** to **RXP** using the 2 SMA to SMA cables as shown in the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

Figure 44 • **Board Setup**



6. On the host PC, start the FlashPro Express software.
7. Click **New Project** to create a new project.
   In the New Project window, do the following, and click OK:
8. Browse the job file in the design files.
9. Choose the location of the FlashPro Express project.

Figure 45 • **New project window**



New Job Project from FlashPro Express Job

Programming job file: st_XCVR_Design\designer\LiteFast_XCVR_Top\export\LiteFast_XCVR_Top.job

FlashPro Express job project name: LiteFast_XCVR_Top

FlashPro Express job project location: D:\

10. Click **OK** to program the device.

FlashPro Express programs the .job file through the programmer type connected to the board.

7 Appendix 2: 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_dg0783_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.

8 Appendix 3: References

This section lists documents that provide more information about the LiteFast standard and IP cores used in the reference design.

- For more information about LiteFast IP, see *UG0701: LiteFast IP User Guide*.
- For more information about PF_CCC, see *UG0684: PolarFire FPGA Clocking Resources User Guide*.
- For more information about CoreFIFO, see *HB0379: CoreFIFO Handbook*.
- For more information about CoreUART, see *HB0095: CoreUART Handbook*.
- For information about PolarFire transceiver blocks, PF_TX_PLL, and PF_XCVR_REF_CLK, see *UG0677: PolarFire FPGA Transceiver User Guide.*
- For more information about Libero, ModelSim, and Synplify, see the Microsemi Libero SoC PolarFire web page.