PolarFire FPGA: High-Speed Data Transfer in 8b10b Mode Using the LiteFast IP
# Contents

1 Revision History .................................................................................. 1  
  1.1 Revision 1.1 .................................................................................. 1  
  1.2 Revision 1.0 .................................................................................. 1  

2 High-Speed Data Transfer Using the LiteFast IP ........................................ 2  
  2.1 Design Requirements ...................................................................... 2  
  2.2 Prerequisites .................................................................................. 3  
  2.3 Demo Design .................................................................................. 3  
    2.3.1 Design Implementation ............................................................ 4  
    2.3.2 IP Configuration ...................................................................... 6  
  2.4 Clocking Structure .......................................................................... 13  
  2.5 Simulating the Design .................................................................... 13  
    2.5.1 Simulation Flow ....................................................................... 15  

3 Libero Design Flow .............................................................................. 18  
  3.1 Synthesize ..................................................................................... 18  
  3.2 Place and Route ............................................................................ 19  
    3.2.1 Resource Utilization ................................................................. 19  
  3.3 Verify Timing ................................................................................ 20  
  3.4 Generate FPGA Array Data ............................................................. 20  
  3.5 Design and Memory Initialization .................................................... 20  
  3.6 Generate Bitstream ........................................................................ 21  
  3.7 Run PROGRAM Action ................................................................... 22  

4 Programming the Device Using FlashPro ............................................ 23  

5 Running the Demo ............................................................................... 25  

6 Using LiteFast For Board-to-Board Data Transfer ................................. 28  

7 Appendix: References ......................................................................... 29
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Hardware Implementation Block Diagram</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2</td>
<td>LiteFast Top-Level Design</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>UART_IF_0 Sub-Blocks</td>
<td>4</td>
</tr>
<tr>
<td>Figure 4</td>
<td>LiteFast_Transmitter_0 Sub-Blocks</td>
<td>5</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Transceiver_IF_0 Sub-Blocks</td>
<td>5</td>
</tr>
<tr>
<td>Figure 6</td>
<td>LiteFast_Receiver_0 Sub-Blocks</td>
<td>5</td>
</tr>
<tr>
<td>Figure 7</td>
<td>PF_OSC_0 Configurator</td>
<td>6</td>
</tr>
<tr>
<td>Figure 8</td>
<td>PF_CCC_0 Clock Options PLL</td>
<td>7</td>
</tr>
<tr>
<td>Figure 9</td>
<td>PF_CCC_0 Output Configuration</td>
<td>7</td>
</tr>
<tr>
<td>Figure 10</td>
<td>COREUART_0 Configurator</td>
<td>8</td>
</tr>
<tr>
<td>Figure 11</td>
<td>LiteFast_tx_0 Configurator</td>
<td>8</td>
</tr>
<tr>
<td>Figure 12</td>
<td>COREFIFO_0 Configurator</td>
<td>9</td>
</tr>
<tr>
<td>Figure 13</td>
<td>PF_TX_PLL Configurator</td>
<td>9</td>
</tr>
<tr>
<td>Figure 14</td>
<td>PF_XCVR_REF_CLK Configurator</td>
<td>10</td>
</tr>
<tr>
<td>Figure 15</td>
<td>PF_XCVR_0 Configurator</td>
<td>10</td>
</tr>
<tr>
<td>Figure 16</td>
<td>LiteFast_rx_0 Configurator</td>
<td>11</td>
</tr>
<tr>
<td>Figure 17</td>
<td>COREFIFO_0 and COREFIFO_1 Configurator</td>
<td>11</td>
</tr>
<tr>
<td>Figure 18</td>
<td>COREFIFO_2 Configurator</td>
<td>12</td>
</tr>
<tr>
<td>Figure 19</td>
<td>COREFIFO_3 Configurator</td>
<td>12</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Clocking Structure</td>
<td>13</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Testbench and LiteFast Reference Design Interaction</td>
<td>14</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Simulating the Design</td>
<td>15</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Simulation Waveform with No Errors</td>
<td>16</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Simulation Waveform With Payload Error</td>
<td>16</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Simulation Waveform With CRC Error</td>
<td>17</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Libero Design Flow Options</td>
<td>18</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Starting I/O Editor</td>
<td>19</td>
</tr>
<tr>
<td>Figure 28</td>
<td>I/O Editor XCVR View</td>
<td>19</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Design and Memory Initialization</td>
<td>20</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Generating XCVR Initialization Client</td>
<td>21</td>
</tr>
<tr>
<td>Figure 31</td>
<td>XCVR Initialization Client Created</td>
<td>21</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Board Setup</td>
<td>22</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Board Setup</td>
<td>23</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Selecting the on-board FlashPro5</td>
<td>24</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Detecting the COM Port and Host Connection Status</td>
<td>25</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Starting the LiteFast Demo</td>
<td>25</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Overall Status</td>
<td>26</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Enabling Payload Error</td>
<td>26</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Checking Payload Error Status</td>
<td>26</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Enabling CRC Error</td>
<td>26</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Checking CRC Error Status</td>
<td>27</td>
</tr>
<tr>
<td>Figure 42</td>
<td>Data Transfer Between Board A and B</td>
<td>28</td>
</tr>
</tbody>
</table>
### Tables

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Design Requirements</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2</td>
<td>I/O Signals</td>
<td>6</td>
</tr>
<tr>
<td>Table 3</td>
<td>Simulation Signals</td>
<td>14</td>
</tr>
<tr>
<td>Table 4</td>
<td>Resource Utilization</td>
<td>19</td>
</tr>
<tr>
<td>Table 5</td>
<td>Jumper Settings</td>
<td>22</td>
</tr>
<tr>
<td>Table 6</td>
<td>Jumper Settings</td>
<td>23</td>
</tr>
</tbody>
</table>
1 Revision History

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

1.1 Revision 1.1
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.

1.2 Revision 1.0
Revision 1.0 was 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 12.5 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 .stp file: To program the device using the .stp file provided along with the reference design, see Programming the Device Using FlashPro, page 23.
- Using Libero SoC PolarFire: To program the device using Libero SoC PolarFire, see Libero Design Flow, page 18.

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 28.

2.1 Design Requirements

The following table lists the resources required to run the demo.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows 7, 8.1, or 10</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td></td>
</tr>
<tr>
<td>PolarFire Evaluation Kit (MPF300-EVAL-KIT)</td>
<td>Rev B or later</td>
</tr>
<tr>
<td>– PolarFire evaluation board</td>
<td></td>
</tr>
<tr>
<td>– 12 V/5 A wall-mounted power adapter</td>
<td></td>
</tr>
<tr>
<td>– USB 2.0 A-male to mini-B cable for UART and programming</td>
<td></td>
</tr>
<tr>
<td>2 SMA-to-SMA cables (not provided with the kit)</td>
<td></td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>FlashPro</td>
<td>12.100.9.13</td>
</tr>
<tr>
<td>Libero SoC PolarFire</td>
<td>v1.1SP1</td>
</tr>
<tr>
<td>ModelSim</td>
<td>10.5c Pro</td>
</tr>
<tr>
<td>Synplify Pro</td>
<td>L-2016.09M-G5</td>
</tr>
</tbody>
</table>
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_liberosocpolarfirev1p1_sp1_df
2. Download and install Libero SoC PolarFire v1.1 SP1 on the host PC from the following location.
   https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc-polar-fire#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>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PADs_IN</td>
<td>Transceiver LANE0_RXD_P and LANE0_RXD_N on the board</td>
</tr>
<tr>
<td>REF_CLK_PAD_P_0 and</td>
<td>This is the differential reference clock generated from the on-board 125 MHz oscillator</td>
</tr>
<tr>
<td>REF_CLK_PAD_N_0</td>
<td></td>
</tr>
<tr>
<td>RX</td>
<td>This is the input signal received by the UART interface from the GUI</td>
</tr>
<tr>
<td>TX</td>
<td>This is the output data received by the GUI from the UART interface</td>
</tr>
</tbody>
</table>

Output Signals

| PADs_OUT                | LANE0_TXD_P and LANE0_TXD_N looped back to LANE0_RXD_P and LANE0_RXD_N using SMA cables |
| LANE0_TX_CLK_STABLE     | This signal asserts high after TX_PLL locks to 125 MHz                      |

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

### 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_OSC_0 (on-chip RC160MHz oscillator), PF_CCC, CoreUART, Reset_Synchronizer_0, and FabUART modules. These modules are described in the following sections.

#### 2.3.2.1.1 PF_OSC_0

This on-chip oscillator provides clock to the PF_CCC_0 block. In the reference design, PF_OSC_0 is configured to provide a 160 MHz clock through a global buffer as shown in the following figure.

![PF_OSC_0 Configurator](image)
2.3.2.1.2 PF_CCC_0

The PF_CCC_0 block provides 125 MHz output fabric clock to COREUART_0 and FabUART_0 modules, which are fabric blocks. The following figures shows the input and output configurations of PF_CCC_0.

Figure 8 • PF_CCC_0 Clock Options PLL

![PF_CCC_0 Clock Options PLL](image)

Figure 9 • PF_CCC_0 Output Configuration

![PF_CCC_0 Output Configuration](image)

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

2.3.2.1.3 COREUART_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_0 and FabUART_0 modules run at the same frequency. Hence, the TX and RX FIFO options are disabled in the COREUART_0 configurator as shown in Figure 10, page 8. For more information about CoreUART, see HB0095: CoreUART Handbook.
2.3.2.1.4 Reset_Synchronizer_0
The Reset_Synchronizer_0 block is a two stage synchronizer, which synchronizes the PLL_LOCK_0 signal.

Figure 10 • COREUART_0 Configurator

2.3.2.1.5 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_tx_0
LiteFast_tx_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 11 • LiteFast_tx_0 Configurator

2.3.2.2.3 Synchronizer_0
The Synchronizer_0 block is a two stage synchronizer, which synchronizes the req_usr_data_tx_o signal.
2.3.2.4 COREFIFO_0

The COREFIFO_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_0 configuration. For more information about CoreFIFO, see HB0379: CoreFIFO Handbook.

Figure 12 • COREFIFO_0 Configurator

2.3.2.3 Transceiver_IF_0

This block contains the PF_TX_PLL_0, PF_XCVR_REF_CLK_0, and the PF_XCVR_0 modules. These modules are described in the following sections.

2.3.2.3.1 PF_TX_PLL

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 13 • PF_TX_PLL Configurator
2.3.2.3.2 PF_XCVR_REF_CLK

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 14 • PF_XCVR_REF_CLK Configurator

2.3.2.3.3 PF_XCVR_0

The PolarFire high-speed transceiver (PF_XCVR) is a hard IP block that supports high-speed data rates ranging from 250 Mbps to 12.5 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 5 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 5 Gbps data rate. For more information about the PolarFire Transceiver, see UG0677: PolarFire FPGA Transceiver User Guide.

The following figure shows the PF_XCVR_0 configuration.

Figure 15 • PF_XCVR_0 Configurator
2.3.2.4 **LiteFast_Receiver_0**

This block contains the LiteFast_rx_0, COREFIFO, and Count_Checker_0 modules. These modules are described in the following sections.

2.3.2.4.1 **LiteFast_rx_0**

LiteFast_rx_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_rx_0 Configurator**

![LiteFast Configurator](image)

2.3.2.4.2 **COREFIFO**

The LiteFast_Receiver block contains the following instances of COREFIFO:

- COREFIFO_0 stores the storage capacity of the LiteFast_rx_0 block instantiated on the receiving board, and sends this data to the LiteFast_tx_0 block on the transferring board for preventing excess transfer of frames.
- COREFIFO_1 stores the storage capacity of the LiteFast_rx_0 block instantiated on the transferring board for preventing excess transfer of frames. The following figure shows the COREFIFO configuration.

*Figure 17* • **COREFIFO_0 and COREFIFO_1 Configurator**

![COREFIFO Configurator](image)

- COREFIFO_2 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_2 configuration.
Figure 18 • COREFIFO_2 Configurator

- COREFIFO_3 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_3 configuration.

Figure 19 • COREFIFO_3 Configurator

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.
2.4 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.

![Clocking Structure Diagram]

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_liberosocpolarfirev1p1_sp1_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 21** • Testbench and LiteFast Reference Design Interaction

**Table 3** • Simulation Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>From Testbench to DUT</strong></td>
<td></td>
</tr>
<tr>
<td>SYSCLK</td>
<td>125 MHz system clock</td>
</tr>
<tr>
<td>NSYSRESET</td>
<td>This is the reset pulse that resets the PF_XCVR_0 block</td>
</tr>
<tr>
<td>start_o</td>
<td>This signal starts the Counter and Checker</td>
</tr>
<tr>
<td>payload_err_o</td>
<td>This signal injects payload error in the Counter generated data</td>
</tr>
<tr>
<td>crc_err_o</td>
<td>This signal injects CRC error in the LiteFast frames generated by the LiteFast_tx_0 module</td>
</tr>
<tr>
<td>clear_o</td>
<td>This signal disables payload error and CRC error</td>
</tr>
<tr>
<td><strong>From DUT to Testbench</strong></td>
<td></td>
</tr>
<tr>
<td>data_out_o</td>
<td>This is the Counter generated data passed to LiteFast_tx_0</td>
</tr>
<tr>
<td>LANE0_RX_VAL</td>
<td>This output signal of the PF_XCVR indicates that the XCVR has received and validated the LiteFast idle frames</td>
</tr>
<tr>
<td>LANE0_TX_CLK_STABLE</td>
<td>This output signal of the PF_XCVR indicates that the LANE0_TX_CLK is locked to transmitter frequency (TX PLL)</td>
</tr>
<tr>
<td>LANE0_TX_DATA</td>
<td>This data bus transmits LiteFast frames to Transceiver</td>
</tr>
<tr>
<td>LANE0_RX_DATA</td>
<td>This data bus receives LiteFast frames from Transceiver</td>
</tr>
</tbody>
</table>
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 22** • 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. After 10 clock cycles (80 ns), the testbench asserts the NSYSRESET signal to the DUT.
3. At ~172 ns, the testbench asserts the LANE0_TX_CLK_STABLE signal. This indicates that PF_TX_PLL in the DUT has locked to 125 MHz.
4. The LiteFast IP sends IDLE character (K28.5) to XCVR.
5. The K28.5 character (0x000000BC) is looped back from TX lane to RX lane.
6. The receiver PLL locks (LANE0_RX_CLK_R) to 125 MHz and then, the K28.5 character asserts the LANE0_RX_VAL signal high at ~26939 ns.
7. The Counter_0, LiteFast_tx_0, LiteFast_rx_0, and Count_Checker_0 come out of reset because of the LANE0_RX_VAL assertion.
8. At 27000 ns, the testbench drives the start_o (UART_IF_0 output) signal high.
9. The Counter_0 starts sending incremental data starting from 0 to LiteFast_tx_0, which sends that data to PF_XCVR_0.
10. The LiteFast_rx_0 receives the data from the external loopback and the Count_Checker_0 compares the received data. **Figure 23**, page 16 shows the simulation waveform with no errors.

### Table 3 • Simulation Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rx_val_o</td>
<td>This is the registered version of LANE0_RX_VAL signal in LANE0_RX_CLK domain</td>
</tr>
<tr>
<td>lock_o</td>
<td>This signal asserts high when the counter and checker data match</td>
</tr>
<tr>
<td>error_o</td>
<td>This signal asserts high when the counter and checker data mismatch</td>
</tr>
<tr>
<td>data_tx_o</td>
<td>This output bus of the LiteFast_rx_0 contains the de-framed data passed to the Count_Checker_0</td>
</tr>
<tr>
<td>data_rx_o</td>
<td>This is the Count_Checker_0 generated data for comparing with the data_tx_o</td>
</tr>
</tbody>
</table>

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 22** • 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. After 10 clock cycles (80 ns), the testbench asserts the NSYSRESET signal to the DUT.
3. At ~172 ns, the testbench asserts the LANE0_TX_CLK_STABLE signal. This indicates that PF_TX_PLL in the DUT has locked to 125 MHz.
4. The LiteFast IP sends IDLE character (K28.5) to XCVR.
5. The K28.5 character (0x000000BC) is looped back from TX lane to RX lane.
6. The receiver PLL locks (LANE0_RX_CLK_R) to 125 MHz and then, the K28.5 character asserts the LANE0_RX_VAL signal high at ~26939 ns.
7. The Counter_0, LiteFast_tx_0, LiteFast_rx_0, and Count_Checker_0 come out of reset because of the LANE0_RX_VAL assertion.
8. At 27000 ns, the testbench drives the start_o (UART_IF_0 output) signal high.
9. The Counter_0 starts sending incremental data starting from 0 to LiteFast_tx_0, which sends that data to PF_XCVR_0.
10. The LiteFast_rx_0 receives the data from the external loopback and the Count_Checker_0 compares the received data. Figure 23, page 16 shows the simulation waveform with no errors.
11. 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.

12. At 34000 ns, the testbench drives the payload_err_o signal low.
13. 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 25 • Simulation Waveform With CRC Error**

14. 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.
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 26 • Libero Design Flow Options**

### 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 26, page 18.

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 27 • 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 28 • 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 26, page 18.

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>1818</td>
<td>299544</td>
<td>0.61</td>
</tr>
<tr>
<td>DFF</td>
<td>1433</td>
<td>299544</td>
<td>0.48</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>1530</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Element</td>
<td>2074</td>
<td>299544</td>
<td>0.69</td>
</tr>
</tbody>
</table>
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 26, page 18.
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 26, page 18.

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.
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.

![Generating XCVR Initialization Client](image)

**Figure 30 • Generating XCVR Initialization Client**

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

![XCVR Initialization Client Created](image)

**Figure 31 • XCVR Initialization Client Created**

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 26, page 18.

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

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

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to 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 32 • Board Setup


When the device is successfully programmed, a green tick mark appears as shown in Figure 26, page 18. See Running the Demo, page 25 to run the LiteFast standalone demo.
4 Programming the Device Using FlashPro

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

\texttt{mpf\_dg0783\_liberoscopolarfirev1p1\_spl\_df\Programming\_File}

Follow these steps:

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

\begin{table}[h]
\centering
\begin{tabular}{|c|p{10cm}|c|}
\hline
Jumper & Description & Default \\
\hline
J18, J19, J20, J21, J22 & Close pin 2 and 3 for programming the PolarFire FPGA through FTDI & Closed \\
\hline
J28 & Close pin 2 and 3 for programming through the on-board FlashPro5 & Open \\
\hline
J26 & Close pin 1 and 2 for programming through the FTDI SPI & Closed \\
\hline
J27 & Close pin 1 and 2 for programming through the FTDI SPI & Closed \\
\hline
J4 & Close pin 1 and 2 for manual power switching using SW3 & Closed \\
\hline
J12 & Close pin 3 and 4 for 2.5 V & Closed \\
\hline
J46 & Close pin 1 and 2 for setting the Reference Clock to 125 MHz on board oscillator & Closed \\
\hline
\end{tabular}
\end{table}

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

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{board_setup.png}
\caption{Board Setup}
\end{figure}
6. On the host PC, start the FlashPro software.
7. Click **New Project** to create a new project.
   In the New Project window, do the following, and click OK:
8. Enter a project name.
9. Select **Single device** as the programming mode.
10. Click **Configure Device**.
11. Click **Browse**, and select the *PolarFire_LiteFast_8b10b_Demo.stp* file from the **Load Programming File** window.
12. From the **View Programmer** pane, select the on-board FlashPro5 programmer as shown in the following figure.

   **Figure 34 • Selecting the on-board FlashPro5**

   ![Selecting the on-board FlashPro5](image)

13. Click **Program** to program the device.
   The Programmer List Window in the FlashPro, shows the Programmer Name, Programmer Type, Port, Programmer Status, and the Programmer Enabled information.
   When the device is programmed successfully, a Run Program PASSED status is displayed. The device is successfully programmed. See **Running the Demo**, page 25 to run the LiteFast standalone demo.
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_liberosocpolarfirev1p1_sp1_df.zip` file.
2. From the `mpf_dg0783_liberosocpolarfirev1p1_sp1_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 35 • Detecting the COM Port and Host Connection Status**

   ![Detecting the COM Port and Host Connection Status](image)

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

   **Figure 36 • Starting the LiteFast Demo**

   ![Starting the LiteFast Demo](image)
The Serial Link, Rx Lock, PayLoad Error, and CRC Error changes to a green tick mark as shown in the following figure.

**Figure 37 • Overall Status**

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 38 • Enabling Payload Error**

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

**Figure 39 • Checking Payload Error Status**

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

**Figure 41 • 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.
6 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 42 • 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.
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.