Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>RTG4 Typical Application Block Diagram</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>LiteFast Smart Design Top-Level Diagram</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>LiteFast Transmitter Smart Design</td>
<td>4</td>
</tr>
<tr>
<td>Figure 4</td>
<td>LiteFast IP Transmitter Configurator</td>
<td>5</td>
</tr>
<tr>
<td>Figure 5</td>
<td>LiteFast Receiver Smart Design</td>
<td>5</td>
</tr>
<tr>
<td>Figure 6</td>
<td>LiteFast IP Receiver Configurator</td>
<td>6</td>
</tr>
<tr>
<td>Figure 7</td>
<td>SerDes Configurator window</td>
<td>7</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Design Structure</td>
<td>8</td>
</tr>
<tr>
<td>Figure 9</td>
<td>RTG4 Development Kit Board Overview</td>
<td>9</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Board Setup</td>
<td>10</td>
</tr>
<tr>
<td>Figure 11</td>
<td>New Project Window</td>
<td>11</td>
</tr>
<tr>
<td>Figure 12</td>
<td>FlashPro GUI Window</td>
<td>11</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Single Device Configuration Window</td>
<td>12</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Action List Window</td>
<td>12</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Select Action and Procedures</td>
<td>12</td>
</tr>
<tr>
<td>Figure 16</td>
<td>LiteFast GUI Setup Window</td>
<td>13</td>
</tr>
<tr>
<td>Figure 17</td>
<td>LiteFast GUI Installation</td>
<td>14</td>
</tr>
<tr>
<td>Figure 18</td>
<td>LiteFast GUI Setup Progress Bar</td>
<td>14</td>
</tr>
<tr>
<td>Figure 19</td>
<td>LiteFast GUI Window</td>
<td>15</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Connected LiteFast GUI</td>
<td>18</td>
</tr>
<tr>
<td>Figure 21</td>
<td>GUI Payload Error Injection</td>
<td>18</td>
</tr>
<tr>
<td>Figure 22</td>
<td>GUI with CRC Error Injection</td>
<td>19</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Custom LiteFast Demo Diagram</td>
<td>20</td>
</tr>
</tbody>
</table>
## Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>LiteFast IP Support Mode</td>
<td>3</td>
</tr>
<tr>
<td>Table 2</td>
<td>Hardware Requirements</td>
<td>8</td>
</tr>
<tr>
<td>Table 3</td>
<td>Main Sections of the GUI</td>
<td>15</td>
</tr>
<tr>
<td>Table 4</td>
<td>Push Buttons in the GUI</td>
<td>16</td>
</tr>
<tr>
<td>Table 5</td>
<td>Status Signals on the GUI</td>
<td>16</td>
</tr>
<tr>
<td>Table 6</td>
<td>GUI Data Window options</td>
<td>17</td>
</tr>
<tr>
<td>Table 7</td>
<td>Slider Bars Available on GUI</td>
<td>17</td>
</tr>
<tr>
<td>Table 8</td>
<td>System Resource Utilization for Demo Design</td>
<td>21</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 2.0
The document was updated to include features and enhancements introduced in the Libero v11.8 SP2 release.

1.2 Revision 1.0
Revision 1.0 was the first publication of this document.
Microsemi LiteFast IP is a scalable, lightweight in terms of utilization, high data rate protocol for applications based upon high-speed serial communication. LiteFast has an inbuilt flow control scheme and physical link is maintained when there is no application data for transmission.

Microsemi LiteFast IP has two distinct sections as LiteFast transmitter and LiteFast receiver. The LiteFast demo design includes the LiteFast transmitter and receiver sections along with example design as part of the user application for traffic generation and frame checking and other components to demonstrate as validation suit. LiteFast transmitter packs the application data into data frame and initiates the data transmission. LiteFast receiver extracts application data from data frame and delivers the application data to the user interface. An idle frame is transmitted when there is no application data for transmission, the physical link between systems is maintained by idle frames.

For a given system of targeted application, the received data extracted from the data frame is written in to a receiver buffer. If available storage space of receiver buffer approaches to zero, LiteFast receiver notifies the remote LiteFast transmitter to pause data frame transmission, to prevent the overflow of receiver buffers. If available storage space of receiver buffer is greater than a threshold value, the LiteFast receiver would notify the remote LiteFast transmitter to resume data frame transmission.

Threshold value must be at least 128 bytes and upper limit of threshold is fixed by user application.

The following are the main features of LiteFast IP:

- Supports x1, x2, or 4x lanes per SerDes
- Supports cumulative speeds from 4 to 10 Gbps for x4 lanes per SerDes
- Serial full duplex or serial simplex operation
- Data packet size: 1 to 128 bytes of application data. The length of the payload must be a multiple of eight, otherwise K28.4 bytes are filled to meet the requirement
- Idle packet: 8 bytes
- Supports 8b10b encoding mechanism
- Supports CRC-32
- Supports hot plug
- Idle frame for establishing and maintaining the link and data frame for user data
- Flow control through token exchange
- Word alignment, block alignment, and lane alignment for the receive chain
- Independent of user application and the device
- Supports for little endian

LiteFast transmitter module packs user data into data frame and generates idle frame. Frame data are striped on multiple lanes if multiple lanes are configured.

In multiple lanes LiteFast receiver, multiple lanes are aligned according to /A/ order sets, then LiteFast receiver module un-strips all lanes together to get LiteFast data frame and idle frame. LiteFast receiver module drops idle frame and extracts payload in data frame. LiteFast receiver monitors token field in the data and idle frames and provides the remote token value (used for flow control) to transmitter.

LiteFast IP data width supports 8bits/16bits/32bits/64bits and supports x1, x2, or x4 lanes.
The following table lists the working mode supported by LiteFast IP.

<table>
<thead>
<tr>
<th>User Application Data Width</th>
<th>1 Lane</th>
<th>2 Lanes</th>
<th>4 Lanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>Support</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>16 bits</td>
<td>Support maximum 16-bit per lane</td>
<td>Support maximum 8-bit per lane</td>
<td>No</td>
</tr>
<tr>
<td>32 bits</td>
<td>No</td>
<td>Support minimum 16-bit per lane</td>
<td>Support maximum 8-bit per lane</td>
</tr>
<tr>
<td>64 bits</td>
<td>No</td>
<td>No</td>
<td>Support maximum 16-bit per lane</td>
</tr>
</tbody>
</table>

Because the 8b10b IP (CorePCS) in Libero supports 8 bits or 16 bits data width, each lane data width can only be 8 bits or 16 bits. LiteFast IP supports maximum four lanes per SerDes.

**Note:** User has to instantiate CorePCS from Libero catalog and configure SerDes through SerDes configurator, for limitation on data width and serial lane bandwidth.

The following figure shows the typical application block diagram of the LiteFast demo.

*Figure 1 • RTG4 Typical Application Block Diagram*

---

### 2.1 Device Family Support

The following FPGA families are supported by the IP core:

- SmartFusion®2 (All devices with transceivers in this family)
- IGLOO®2 (All devices with transceivers in this family)
- RTG4™

**Note:** User must ensure that, the selected device in the family has transceivers.
2.2 Hardware Design

The hardware design for the implementation includes a LiteFast transmitter and LiteFast receiver blocks connected to the RTG4 FPGA SerDes. UART block communicates with the GUI to send data and receive control signals. The top-level Smart Design diagram for the design is shown in the following figure.

Figure 2 • LiteFast Smart Design Top-Level Diagram

Note: All the figures shown in this demo guide are from 16-bit external loopback demo design.

2.2.1 CorePCS

The CorePCS is implemented in the Fabric and supports programmable 8b10b encoding/decoding. This Core can be configured as a transmitter, receiver, or both transmitter and receiver. Word alignment support is included in receiver. The Core can be configured to support 10-bit or 20-bit external physical coding sublayer (EPCS) data. For more information on CorePCS block, see the CorePCS v3.3 Handbook.

2.3 Module Level Descriptions

2.3.1 LiteFast Transmitter Module

The LiteFast transmitter module contains a counter to generate data, LiteFast IP in transmitter mode, CorePCS block configured in transmitter only mode, and transmitter interface for SerDes. The following figure shows LiteFast smart design transmitter block.

Figure 3 • LiteFast Transmitter Smart Design

2.3.1.1 Counter Module

The counter block contains a 16-bit counter that transmits incremental data, each clock cycle.
2.3.1.2 LiteFast IP in Transmitter Module
The Lite Fast IP is configured in transmitter only mode. The IP can be configured from IP’s configurator window, as shown in the following figure. The g_DATA_WID indicates the data width, g_LANE_NUM indicates the number of lanes to be configured. LiteFast_Mode contains the drop down in which Transmitter Only mode needs to be selected. For more information on exact number of lane, see Table 1, page 3.

Figure 4 • LiteFast IP Transmitter Configurator

2.3.1.3 CorePCS Transmitter Module
The CorePCS transmitter block multiplexes the control and data inputs on EPCS_TX_DATA. The CorePCS uses running disparity technique for 8B/10B encoding. The CorePCS output transmit interface is connected to SerDes EPCS transmit interface.

2.3.1.4 EPCS Transmit Interface Module
The EPCS transmit interface block is a fabric interface that synchronizes data between SerDes block and fabric modules.

It receives data from the fabric modules and transmits onto the SerDes.

2.3.2 LiteFast Receiver Module
The LiteFast receiver block contains a receiver interface from SerDes, LiteFast IP in receiver mode, CorePCS block configured in receiver only mode, and count checker block. It contains the local token generation block, remote token first in first out (FIFO), and local token FIFO. The smart design diagram for the LiteFast receiver block is shown in the following figure.

Figure 5 • LiteFast Receiver Smart Design

2.3.2.1 EPCS Receive Interface Module
The EPCS receive interface block is a fabric interface that synchronizes data between SerDes block and Fabric modules.

It receives the data from SerDes and sends it to the fabric modules.
2.3.2.2 **CorePCS Receiver Module**

The CorePCS in the Rx chain implements 10B/8B decoding. RX_K_CHAR control signal output of CorePCS is used to indicate a control or data character.

2.3.2.3 **LiteFast IP in Receiver Module**

The LiteFast IP is configured in the receiver only mode. In this mode it only receives data. The IP can be configured by double clicking the IP so that the configurator window opens, as shown in the following figure. The g\_DATA\_WID indicates the data width, g\_LANE\_NUM indicates the number of lanes to be configured and LiteFast Mode contains the drop down in which Receiver Only mode needs to be selected. For exact number of lane and data width choice, see Table 1, page 3.

![LiteFast IP Receiver Configurator](image)

2.3.2.4 **Counter Checker Module**

The counter checker block contains a 16-bit count generator and 16-bit checker that checks the incoming data with the self-generated data. Error counter is incremented whenever there is a mismatch between estimated data and captured data.

2.3.3 **SerDes**

RTG4 FPGA high-speed SerDes is a hard IP block on chip that supports rates up to 5 Gbps. The SerDes block offers embedded protocol support for PCIe, SRIO, XAUI, SGMII and so on. The SerDes block also supports EPCS interface which can be used for custom protocols. For more information, see the UG0567: RTG4 FPGA High Speed Serial Interfaces User Guide.
In this demo design, the SERDES_PCIE_5 block is configured for EPCS mode on Lane0 with 20-bit parallel interface on both transmit and receive side, and external reference clock from on-board Oscillator. The configurator window for SERDES in the smart design is shown in the following figure.

**Figure 7 • SerDes Configurator window**

### 2.3.3.1 Reference Clock Source

For the given demo, when line speed is working at 2.5 Gbps, the reference Clock to SerDes is 125 MHz and it is given from differential pads (REFCLKP and REFCLKN). In RTG4 Development Kit, the REFCLK differential pads are driven from 125 MHz on board Oscillator.

### 2.3.4 UART

The UART block contains the CoreUART and FabUART modules. The FabUART module is the wrapper interface that sends/receives commands and data to the GUI through CoreUART block. For more information on CoreUART block, see the [CoreUART v5.4 Handbook](#).

### 2.3.5 CoreABC

CoreABC (ABC = APB bus controller) is a simple, configurable, low-gate count, programmable state machine/controller primarily targeted toward the implementation of advanced microcontroller bus architecture (AMBA) advanced peripheral bus (APB) based designs. CoreABC is used for SerDes initialization in this demo design. For more information, see the [CoreABC v3.5 Handbook](#).
3 Validating the Design on RTG4 Development Kit

3.1 Requirements

The following are the hardware requirements to run the demo:

<table>
<thead>
<tr>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTG4 Development Kit</td>
<td>One</td>
</tr>
<tr>
<td>SMA M to SMA M loopback cables</td>
<td>Two</td>
</tr>
<tr>
<td>Mini to Micro USB cable</td>
<td>One</td>
</tr>
<tr>
<td>STAPL/PDB file</td>
<td>Included with project</td>
</tr>
<tr>
<td>GUI Software</td>
<td>Included with project</td>
</tr>
<tr>
<td>Libero</td>
<td>v11.8 SP2</td>
</tr>
</tbody>
</table>

3.2 Design Files Directory Structure

The demo design files are available for download from the following path in the Microsemi website:

- The internal loop back demo files for 8 bits: [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Int_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Int_Loopback)
- The internal loop back demo files for 16 bits: [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Int_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Int_Loopback)
- The external loop back demo files for 8 bits: [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Ext_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Ext_Loopback)
- The external loop back demo files for 16 bits: [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Ext_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Ext_Loopback)

The following figure shows the top-level structure of the design files. Which are delivered along with this demo guide:

![Design Structure](LiteFast_RTG4_xbits_z_Loopback)

- Libero
  - Project
    - Programming_File

Note: x can be 8 or 16 and z can be external or Internal.
### 3.3 Setting the Board

The following steps describe how to set the board for demo:

1. The RTG4 150 device consists of six SerDes with four lanes each. Lane0 of SERDES_PCIE_5 can be used for external loopback testing on the RTG4 Development Kit board.
2. A pair of SMA-SMA cables are required to loopback SERDES5_TXP_0 to SERDES5_RXP_0 and SERDES5_TXN_0 to SERDES5_RXN_0.
3. Connect the micro-USB to USB-A cable to the board, as shown in the following figure. The USB-A connector needs to be connected to the computer. For more information on RTG4 Development Kit board, see the [UG0617: RTG4 FPGA Development Kit User Guide](#).

![Figure 9 • RTG4 Development Kit Board Overview](image-url)

Figure 9 • RTG4 Development Kit Board Overview
3.4 Building the System

The following steps describe how to rebuild the design:

1. Download the demo design files from the design file links.
2. Save the design files at local disk and unzip it, ensure that the Libero version is 11.8 SP2.
3. Open the Libero project using the Libero tool.
4. Click generate bit stream option in Libero tool.

Observe the synthesis, map, and implementation reports.

**Note:** User need not perform any other activities other than the above ones.

3.5 Programming the Device

The following steps helps in programming the device:

1. Download the design files:
   - The internal loop back demo files for 8 bits:  
     [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Int_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Int_Loopback)
   - The internal loop back demo files for 16 bits:  
   - The external loop back demo files for 8 bits:  
     [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Ext_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_8bits_Ext_Loopback)
   - The external loop back demo files for 16 bits:  
     [http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Ext_Loopback](http://soc.microsemi.com/download/rsc/?f=LiteFast_RTG4_16bits_Ext_Loopback)

   Programming file (STAPL/PDB) is located in the Programming_File folder.

2. Connect USB-UART Cable to RTG4 Development Kit board.
3. Open FlashPro v11.8 SP2 (installed as part of the Libero IDE).
4. Click **New Project** in FlashPro.
5. Enter the project name as **LiteFast_RTG4_Demo** in the **New Project** window.

**Figure 11 • New Project Window**

If necessary, change the default location in the Project Location.

7. Click **OK**. The FlashPro GUI window is displayed. The Programmer List Window updates with programmer information.

**Figure 12 • FlashPro GUI Window**

After the project is connected the program is ready to load the STAPL/PDB file downloaded.

8. Click **Configure Device**. The **Single Device Configuration** window is displayed in FlashPro.
9. Click Browse to find the programming file.
10. On the **Load Existing Programming File** window, select the required programming file and click **Open**.
11. The **Single Device Configuration** window updates the programming file information and the actions available with the programming file in the Action list box.

*Figure 13 • Single Device Configuration Window*

![Single Device Configuration Window](image)

**Note:** Microsemi recommends using the default settings.

12. After loading the programming file, select **Program**. Click **Procedures...**.

*Figure 14 • Action List Window*

![Action List Window](image)

**Select Action and Procedures** window is displayed, showing the procedures for the programming action.

**Note:** Microsemi recommends using the default settings.

13. Click **Restore Default Procedures**.

*Figure 15 • Select Action and Procedures*

![Select Action and Procedures](image)

14. Click **Program** to program the device.
The programmer list window updates the programmer status column with run passed indicating that the device is successfully programmed.

**Note:** The status indicator is updated during programming to show the programming progress, then it changes to a pass or fail result when the operation is completed.

15. View the Log window for the details about of the programmed device.
16. Power Cycle the board.

### 3.6 Executing the Demo Design

#### 3.6.1 LiteFast Demo GUI Installation

Perform the following steps to install the LiteFast IP Demo GUI:

1. Download the GUI installation files: [http://soc.microsemi.com/download/rsc/?f=LiteFast_GUI_1_0_1](http://soc.microsemi.com/download/rsc/?f=LiteFast_GUI_1_0_1)
2. Open `GUI_Installer>Volume>setup.exe`.
3. When prompted, click Yes.
4. On the LiteFast Destination Directory window click Next, as shown in the following figure. Default locations are displayed.

*Figure 16 • LiteFast GUI Setup Window*
5. On the LiteFast Start Installation window click **Next**, as shown in the following figure.

*Figure 17 • LiteFast GUI Installation*

![LiteFast GUI Installation](image1.png)

A progress bar appears which shows the progress of installation, as shown in the following figure. Wait for the installation to complete. It may take few minutes.

*Figure 18 • LiteFast GUI Setup Progress Bar*

![LiteFast GUI Setup Progress Bar](image2.png)
6. If the installation is successful, Installation Complete message appears. Click Finish. Restart computer before using the installed GUI.

**Note:** The GUI given along with this demo design is applicable only for this demo design.

### 3.6.2 LiteFast Demo GUI Description and Usage

1. Open **Programs>LiteFast_IP_Demo**.
2. The LiteFast GUI window appears, as shown in the following figure.

**Figure 19** • LiteFast GUI Window

Note: Hover the mouse pointer on each options to access tool tips and know more information about the specific options.

The following table lists down the main sections of the GUI.

**Table 3** • Main Sections of the GUI

<table>
<thead>
<tr>
<th>GUI Options</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop-down COM port list</td>
<td>As soon as the GUI is opened, the internal logic of the GUI, tries to communicate with the system as well as the board and check the number of available COM ports and the COM port to which the board is connected. If there is any issue during the auto-connect of the host PC with the board, then the user can manually connect by selecting the appropriate COM port from drop down list and clicking connect.</td>
<td>Unless the GUI is not auto connecting, the user doesn't have to perform any activity. This functionality is auto-updated by the GUI.</td>
</tr>
</tbody>
</table>

**Note:** Default settings for the design are 9600 Baud, no flow control, one stop, and no parity. User must not modify any settings.
Validating the Design on RTG4 Development Kit

The following table describes the Push Buttons in the GUI.

### Table 4 • Push Buttons in the GUI

<table>
<thead>
<tr>
<th>GUI Options</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start/Stop</td>
<td>This is dual functionality button. Click the Start button to start the LiteFast demo. The internal counter transmits the count data which is sent over to the serial link. The count data is received by the receiver and checked for any errors. The status at any time can be monitored using the status signals in the GUI. Stop is exactly opposite of the Start and embedded in the same button. On clicking the Stop button, the example design stops the counter and the core link becomes down.</td>
<td>Unless this button is pressed, the demo design will not be operative. User needs to monitor the core functionality.</td>
</tr>
</tbody>
</table>

The following table lists down the status signals on the GUI.

### Table 5 • Status Signals on the GUI

<table>
<thead>
<tr>
<th>GUI Options</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Connection</td>
<td>This indicates if the host is properly connected to the board or not. Green tick mark: It indicates that the communication channel is established Red cross mark: It indicates that there is a communication problem with the host COM port.</td>
<td>User needs to check for the Green Mark before starting any functionality check on the GUI.</td>
</tr>
<tr>
<td>Serial Link</td>
<td>This indicates the transmission link for the serial LiteFast data. Green mark: It indicates that the link is up and running. Red cross mark: It indicates that there is some issue with the board, connection or any other place as the link is not up.</td>
<td>User needs to check for the green mark before starting any functionality check on the GUI. In case of a red cross mark, the user needs to check for the board and cable connections and may have to perform debug.</td>
</tr>
<tr>
<td>Rx Lock</td>
<td>This indicates the receiver lock. Green mark: It indicates that the receiver is in sync with the incoming data. The receiver has locked to the count sequences and the subsequent transmitted sequences are successfully received Red cross mark: It indicates that there is some problem with the receiver as it is not able to receive the data correctly.</td>
<td>User needs to check for the Green Mark before starting any functionality check on the GUI. In case of a Red Cross mark, the user need to check for the board and cable connections and may have to perform debug.</td>
</tr>
<tr>
<td>Payload Error</td>
<td>This indicates the errors received. Green mark: This indicates that there are no payload errors detected. The data received is same as the data transmitted Red cross mark: This indicates that the data received has injected errors.</td>
<td>User needs to monitor the functionality. Debug is not expected from the user. This error is injected in the data packets when the Payload Error slide bar is enabled. It increments the Error Count display.</td>
</tr>
</tbody>
</table>
Validating the Design on RTG4 Development Kit

Table 5 • Status Signals on the GUI (continued)

<table>
<thead>
<tr>
<th>GUI Options</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC Errors</td>
<td>This indicates that the received packet has CRC errors. Green mark: This indicates that the data received is with the correct CRC. Red cross mark: This indicates that the received data has CRC errors</td>
<td>User needs to monitor the functionality. Debug is not expected from user. This error is injected in the data packets when the CRC Error slide bar is enabled.</td>
</tr>
</tbody>
</table>

Note: Default settings for the design are 9600 Baud, no flow control, one stop, and no parity. User need not modify any settings.

The following table lists down the GUI Data Window options.

Table 6 • GUI Data Window options

<table>
<thead>
<tr>
<th>GUI Data Window</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Words</td>
<td>Shows the number of the received data words. This number rolls over after 65535 words.</td>
<td>User needs to monitor the number of Rx words received.</td>
</tr>
<tr>
<td>TX Words</td>
<td>Shows the number of the transmitted data words. This number rolls over after 65535 words.</td>
<td>User needs to monitor the number of Tx words transmitted.</td>
</tr>
<tr>
<td>Error Counter</td>
<td>It indicates the packet loss and rolls over after 65535 words.</td>
<td>User needs to monitor, when the payload errors are injected. This text box shows the count of the errored packets received. The number displayed over here could be different from Tx and Rx packet numbers and may not match the actual subtraction between Tx and Rx words. This is due to the fact that the internal counters are read slowly but the internal logic is accurate.</td>
</tr>
</tbody>
</table>

Note: Error counter may not exactly co-relate to TX words and RX words due to the rollover.

The following table lists down the slider bars available on GUI.

Table 7 • Slider Bars Available on GUI

<table>
<thead>
<tr>
<th>GUI Slider Bars</th>
<th>Description</th>
<th>Expectation from user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload Error Slider</td>
<td>It is used to introduce errors in the transmission for debugging. Enabling this slider injects an error in the transmitted count sequence, which as a result, increments the Error Count display and the Payload Error indicator turns Red</td>
<td>User needs to monitor the functionality of the error injection and error detection by the core on GUI</td>
</tr>
<tr>
<td>CRC Error Slider</td>
<td>It is used to introduce CRC errors in the transmission for debugging. Enabling this slider injects the CRC error in the transmitted count sequence, which as a result, increments the Error Count display and the CRC Error indicator turns red.</td>
<td>User needs to monitor the functionality of the CRC error injection and error detection by the core.</td>
</tr>
</tbody>
</table>
The following figure shows how the GUI looks like during an error free operation of the LiteFast demo system.

**Figure 20 • Connected LiteFast GUI**

![Connected LiteFast GUI](image1)

The following figure shows how the GUI looks like during payload error injection of the LiteFast demo system.

**Figure 21 • GUI Payload Error Injection**

![GUI Payload Error Injection](image2)
The following figure shows how the GUI looks like during CRC error injection of the LiteFast demo system.

*Figure 22 • GUI with CRC Error Injection*
4 Using LiteFast in Customer Application

The demo design consists of LiteFast IP, traffic generator, checker, and other modules. This system is specific to the demo. In case, user likes to use the LiteFast IP in the design, some modifications are needed to be done.

Replace the counter and checker modules with customer design.

4.1 LiteFast Transmitter Section

The count generator in the transmitter section can be replaced with the user data generator module. The data generator is interfaced with the LiteFast transmit module.

4.2 LiteFast Receiver Section

The count checker in the receiver section can be replaced with the data receiver in the user design. The data receiver takes input from the LiteFast receiver module, as shown in the following figure.

Figure 23 • Custom LiteFast Demo Diagram
4.3 Guidelines for Libero Design Flow

In a system, the LiteFast IP can be added as per the system design requirements. Once the IP is added, the following are the steps to be followed for Libero design completion:

1. Create the design using smart design in the Libero v11.8 SP2.
2. Add the LiteFast IP from the IP catalogue.
3. Configure the transmitter and receiver sections, as per system requirements.
4. Go through the synthesis and check for any warnings.
5. Adjust the DELAY value of the delay line module in EPCS RX module to avoid any hold violations.
6. Give proper LiteFast TX and RX clock constraints.
7. Proper care for clock crossing needs to be taken for local and remote token handling.
8. Complete the implementation and generate the bit stream.
9. Program the device.
10. Execute the design and check the functionality.

4.4 System Resource Utilization for Demo Design

Table 8 • System Resource Utilization for Demo Design

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>2495</td>
<td>151824</td>
<td>1.64</td>
</tr>
<tr>
<td>DFF</td>
<td>1957</td>
<td>151824</td>
<td>1.29</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>2160</td>
<td>0.00</td>
</tr>
<tr>
<td>User I/O</td>
<td>3</td>
<td>720</td>
<td>0.42</td>
</tr>
<tr>
<td>-- Single-ended I/O</td>
<td>3</td>
<td>720</td>
<td>0.42</td>
</tr>
<tr>
<td>-- Differential I/O Pairs</td>
<td>0</td>
<td>360</td>
<td>0.00</td>
</tr>
<tr>
<td>RAM64x18</td>
<td>2</td>
<td>210</td>
<td>0.95</td>
</tr>
<tr>
<td>RAM1K18</td>
<td>2</td>
<td>209</td>
<td>0.96</td>
</tr>
<tr>
<td>MACC</td>
<td>0</td>
<td>462</td>
<td>0.00</td>
</tr>
<tr>
<td>H-Chip Globals</td>
<td>7</td>
<td>48</td>
<td>14.58</td>
</tr>
<tr>
<td>CCC</td>
<td>1</td>
<td>8</td>
<td>12.50</td>
</tr>
<tr>
<td>RCOSC_50MHZ</td>
<td>1</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>SYSRESET</td>
<td>1</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>SERDESIF Blocks</td>
<td>1</td>
<td>6</td>
<td>16.67</td>
</tr>
<tr>
<td>FDDR</td>
<td>0</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>GRESET</td>
<td>1</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>RGRESET</td>
<td>2</td>
<td>206</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Note: Resource utilization is given with respect to 16 bit demo design.