

DG0729
Demo Guide
LiteFast IP on RTG4 Device



a  **MICROCHIP** company



a  MICROCHIP company

Microsemi Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Sales: +1 (949) 380-6136

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

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

1	Revision History	1
1.1	Revision 4.0	1
1.2	Revision 3.0	1
1.3	Revision 2.0	1
1.4	Revision 1.0	1
2	Demo Design Features	2
2.1	Device Family Support	3
2.2	Hardware Design	4
2.2.1	CorePCS	4
2.3	Module Level Descriptions	4
2.3.1	LiteFast Transmitter Module	4
2.3.2	LiteFast Receiver Module	5
2.3.3	SerDes	7
2.3.4	UART	8
2.3.5	CoreABC	8
3	Validating the Design on RTG4 Development Kit	9
3.1	Design Requirements	9
3.2	Prerequisites	9
3.3	Demo Design	9
3.4	Setting Up the Demo Design	10
3.5	Building the System	11
3.6	Programming the Device	11
3.7	Executing the Demo Design	12
3.7.1	LiteFast Demo GUI Installation	12
3.7.2	LiteFast Demo GUI Description and Usage	14
4	Using LiteFast in Customer Application	18
4.1	LiteFast Transmitter Section	18
4.2	LiteFast Receiver Section	18
4.3	Guidelines for Libero Design Flow	19
4.4	System Resource Utilization for Demo Design	19
5	Appendix 1: Programming the Device Using FlashPro Express	20
6	Appendix 2: Running the TCL Script	23

Figures

Figure 1	RTG4 Typical Application Block Diagram	3
Figure 2	LiteFast SmartDesign Top-Level Diagram	4
Figure 3	LiteFast Transmitter SmartDesign	4
Figure 4	LiteFast IP Transmitter Configurator	5
Figure 5	LiteFast Receiver SmartDesign	6
Figure 6	LiteFast IP Receiver Configurator	6
Figure 7	SerDes Configurator window	8
Figure 8	Design Structure	10
Figure 9	RTG4 Development Kit Board Overview	10
Figure 10	Board Setup	11
Figure 11	LiteFast GUI Setup Window	12
Figure 12	LiteFast GUI Installation	13
Figure 13	LiteFast GUI Setup Progress Bar	13
Figure 14	LiteFast GUI Window	14
Figure 15	Connected LiteFast GUI	17
Figure 16	GUI Payload Error Injection	17
Figure 17	GUI with CRC Error Injection	17
Figure 18	Custom LiteFast Demo Diagram	18
Figure 19	FlashPro Express Job Project	20
Figure 20	New Job Project from FlashPro Express Job	21
Figure 21	Programming the Device	21
Figure 22	FlashPro Express—RUN PASSED	22

Tables

Table 1	LiteFast IP Support Mode	3
Table 2	Design Requirements	9
Table 3	Main Sections of the GUI	14
Table 4	Push Buttons in the GUI	14
Table 5	Status Signals on the GUI	15
Table 6	GUI Data Window options	16
Table 7	Slider Bars Available on GUI	16
Table 8	System Resource Utilization for Demo Design	19

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 4.0

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

- Updated the document for Libero SoC v2021.2.
- Updated [Figure 2](#), page 4.
- Updated [Figure 5](#), page 6.
- Updated [Figure 8](#), page 10.

1.2 Revision 3.0

This document is updated to include features and enhancements introduced in the LiteFast IP v1.0.5.

- Added [Appendix 1: Programming the Device Using FlashPro Express](#), page 20.
- Added [Appendix 2: Running the TCL Script](#), page 23.
- Removed the references to Libero version numbers.

1.3 Revision 2.0

The document was updated to include features and enhancements introduced in the Libero v11.8 SP2 release.

1.4 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Demo Design Features

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 the 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 a data frame and initiates the data transmission. LiteFast receiver extracts application data from the 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 the targeted application, the received data extracted from the data frame is written into a receiver buffer. If the 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 the available storage space of the receiver buffer is greater than a threshold value, the LiteFast receiver would notify the remote LiteFast transmitter to resume data frame transmission.

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

The following are the main features of LiteFast IP:

- Supports x1, x2, or x4 lanes per SerDes
- Supports cumulative speeds from 4 to 10 Gbps for x4 lanes, 20 Gbps for x2 lanes (10 Gbps per lane), and 12.7 Gbps for x1 lane
- 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 the 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 a data frame and generates an 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 the data frame. LiteFast receiver monitors token field in the data and idle frames and provides the remote token value (used for flow control) to the 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 the LiteFast IP.

Table 1 • LiteFast IP Support Mode

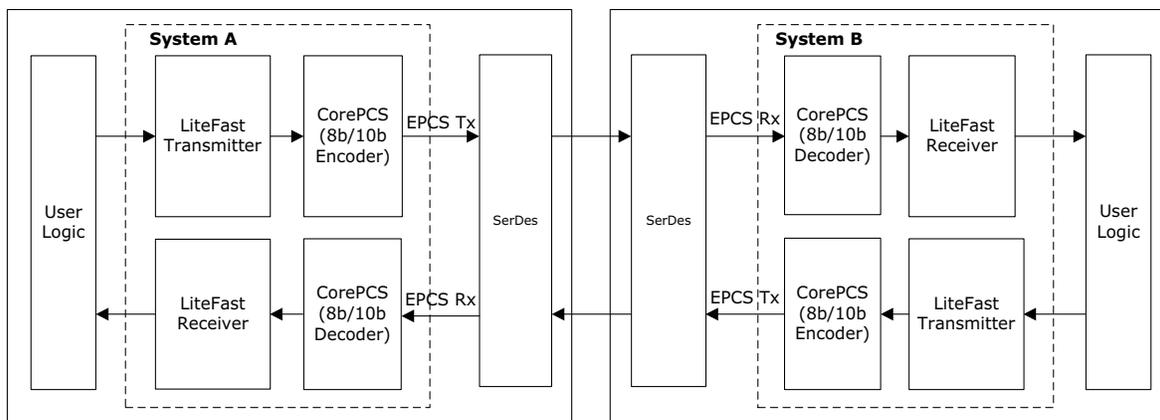
User Application Data Width	1 Lane	2 Lanes	4 Lanes
8 bits	Support	No	No
16 bits	Support maximum 16-bit per lane	Support maximum 8-bit per lane	No
32 bits	Support maximum 32-bit per lane	Support maximum 16-bit per lane	Support maximum 8-bit per lane
64 bits	No	Support maximum 32-bit per lane	Support maximum 16-bit per lane

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 a maximum of 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:

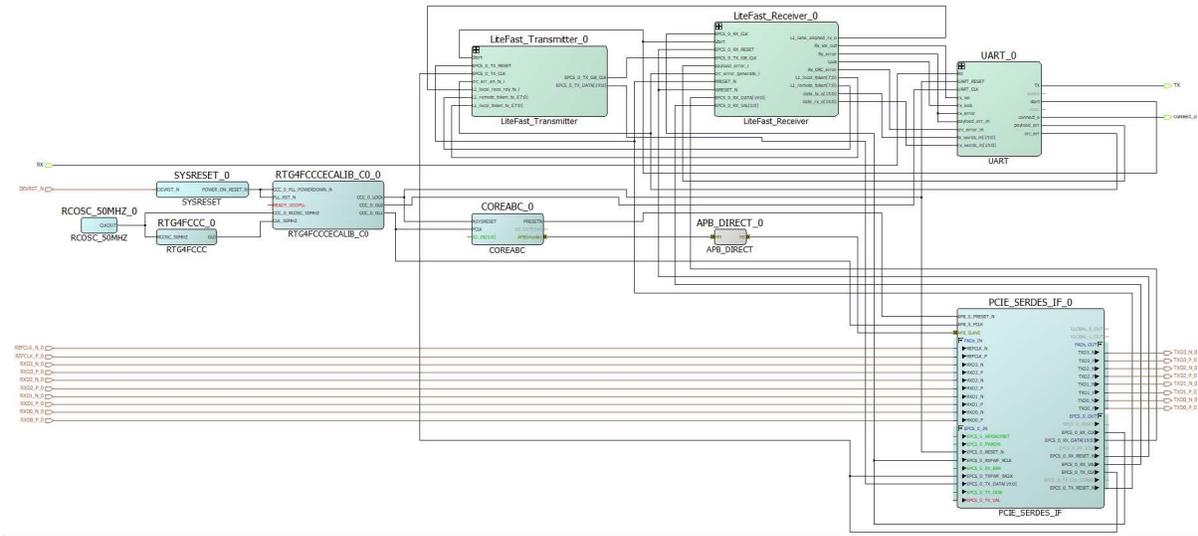
- PolarFire® (All devices with transceivers in this family)
- IGLOO®2 (All devices with transceivers in this family)
- SmartFusion®2 (All devices with transceivers in this family)
- RTG4™

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 SmartDesign diagram for the design is shown in the following figure.

Figure 2 • LiteFast SmartDesign Top-Level Diagram



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

2.2.1 CorePCS

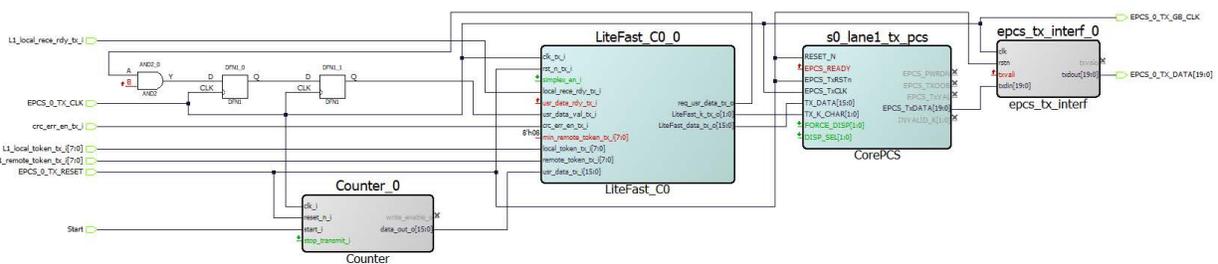
The CorePCS is implemented in the Fabric and supports programmable 8b10b encoding/decoding. This Core can be configured as a transmitter, receiver, or both transmitter and receiver. Word alignment support is included in the receiver. The Core can be configured to support 10-bit or 20-bit external physical coding sublayer (EPCS) data. For more information about CorePCS block, refer to [CorePCS 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 a transmitter mode, CorePCS block configured in transmitter only mode, and transmitter interface for SerDes. The following figure shows the LiteFast SmartDesign transmitter block.

Figure 3 • LiteFast Transmitter SmartDesign



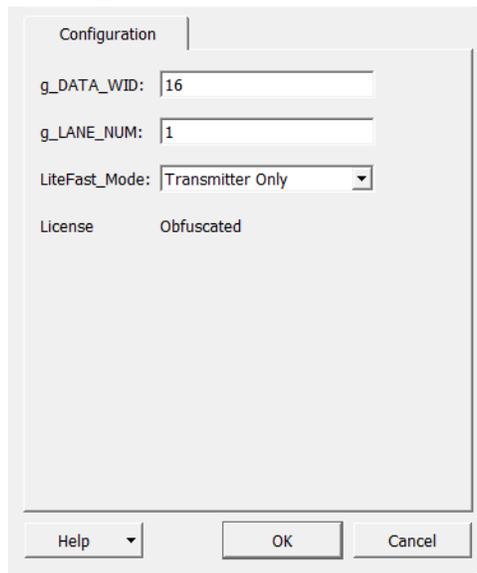
2.3.1.1 Counter Module

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 about exact number of lane, refer 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 the running disparity technique for 8B/10B encoding. The CorePCS output transmits interface is connected to the SerDes EPCS transmit interface.

2.3.1.4 EPCS Transmit Interface Module

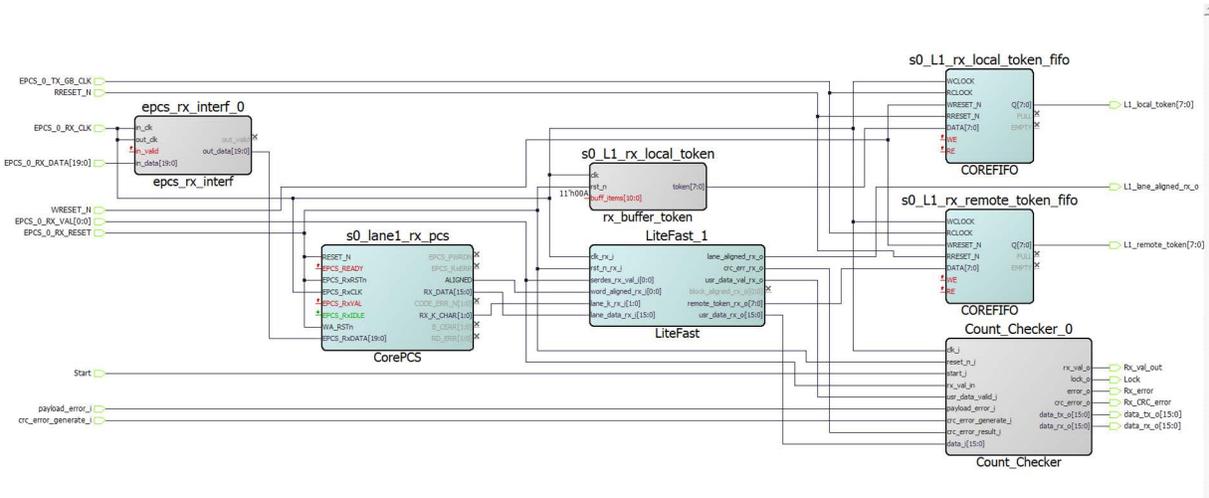
The EPCS transmit interface block is a fabric interface that synchronizes data between SerDes block and fabric modules.

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 SmartDesign diagram for the LiteFast receiver block is shown in the following figure.

Figure 5 • LiteFast Receiver SmartDesign



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.

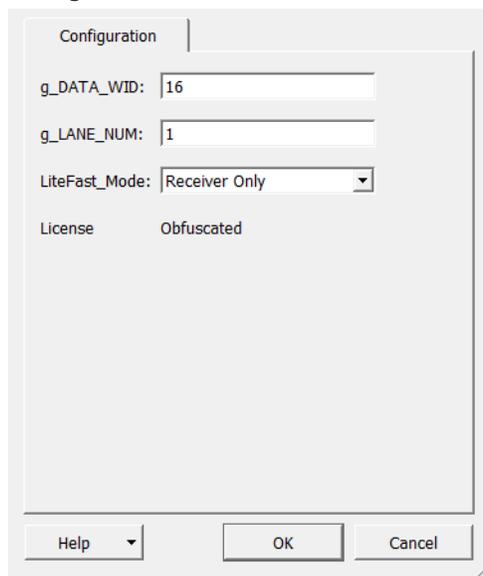
2.3.2.2 CorePCS Receiver Module

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, refer Table 1, page 3.

Figure 6 • LiteFast IP Receiver Configurator



2.3.2.4 Counter Checker Module

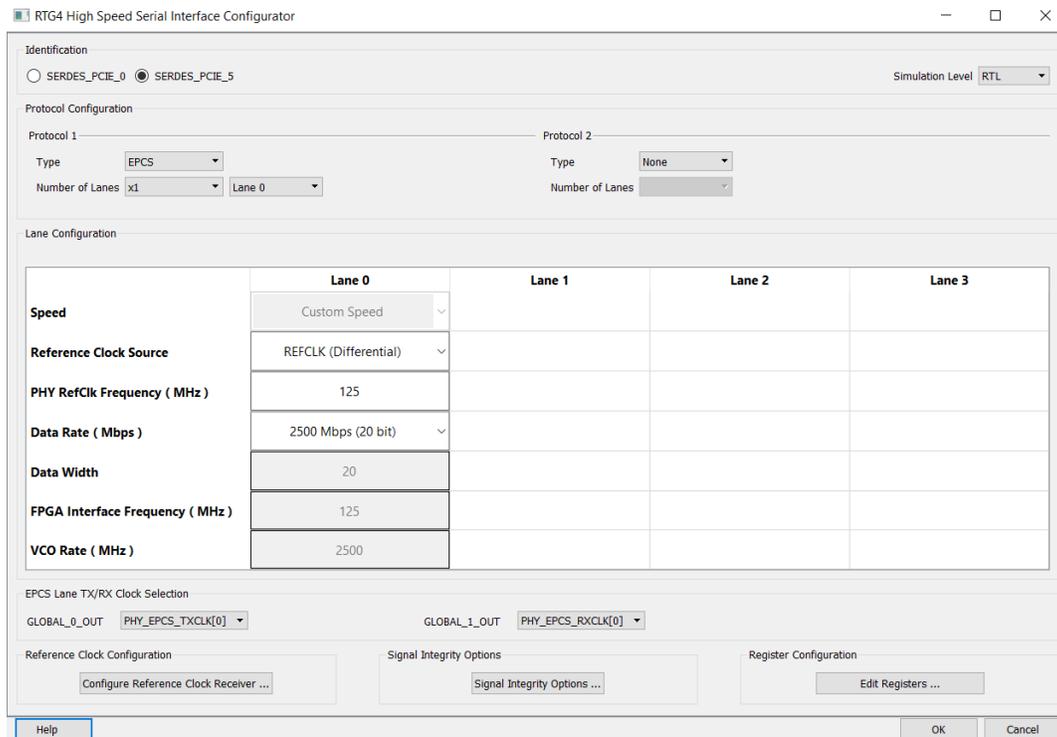
The counter checker block contains a 16-bit count generator and a 16-bit checker that checks the incoming data with the self-generated data. Error counter is incremented whenever there is a mismatch between estimated data and captured data.

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, refer to [UG0567: RTG4 FPGA High Speed Serial Interfaces User Guide](#).

In this demo design, the PCIE_SERDES_IF_0 block is configured for EPCS mode on Lane0 with the 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 SmartDesign 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 about CoreUART block, refer to [CoreUART 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, refer to [CoreABC Handbook](#).

3 Validating the Design on RTG4 Development Kit

3.1 Design Requirements

The following table lists the design requirements to run the demo:

Table 2 • Design Requirements

Requirement	Version
Hardware	
RTG4 Development Kit	-
SMA M to SMA M loopback cables	-
Mini to Micro USB cable	-
Host PC or laptop	64-bit Windows 7 and 10
Software	
JOB file	Included with project
GUI Software	Included with project
Libero [®] System-on-Chip (SoC)	Note: Refer to the <code>readme.txt</code> file provided in the design files for the software versions used with this reference design.
FlashPro Express	

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.

3.2 Prerequisites

Before you start:

- Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location: <https://www.microsemi.com/product-directory/design-resources/1750-libero-soc>

3.3 Demo Design

Download the demo design files from the Microsemi website at:

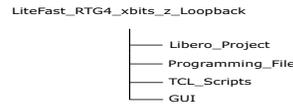
http://soc.microsemi.com/download/rsc/?f=rtg4_dg0729_df

The demo design files include:

- Libero project
- Programming files
- TCL scripts
- GUI

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

Figure 8 • Design Structure



Note: x can be 8 or 16 and z can be external or Internal.

3.4 Setting Up the Demo Design

The following steps describe how to set up the 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 about RTG4 Development Kit board, refer to [UG0617: RTG4 FPGA Development Kit User Guide](#).

Figure 9 • RTG4 Development Kit Board Overview

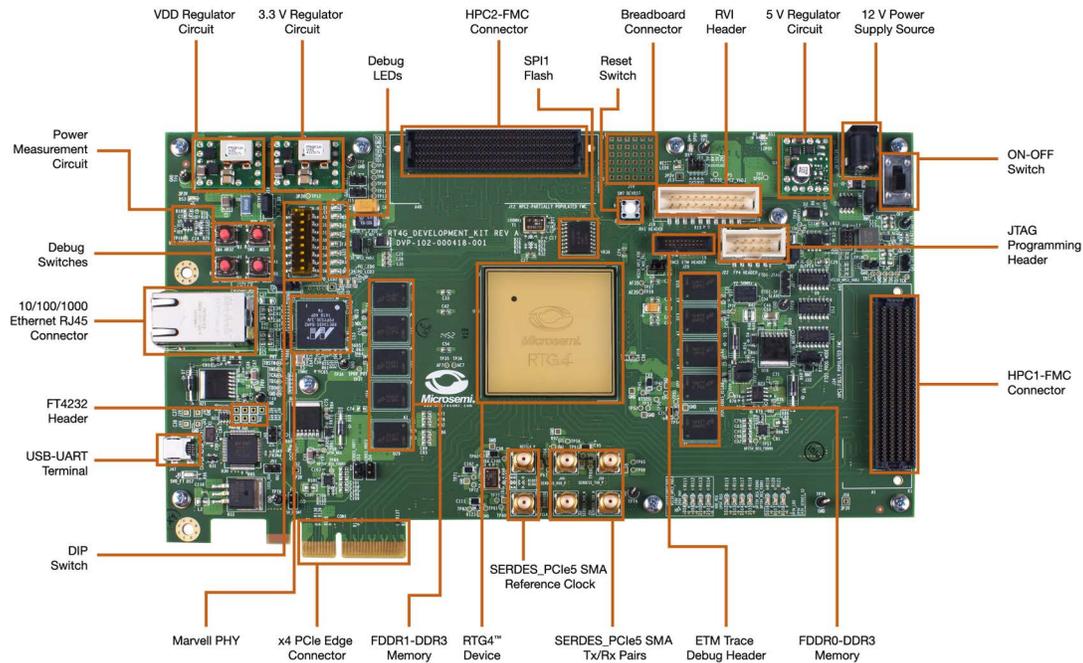
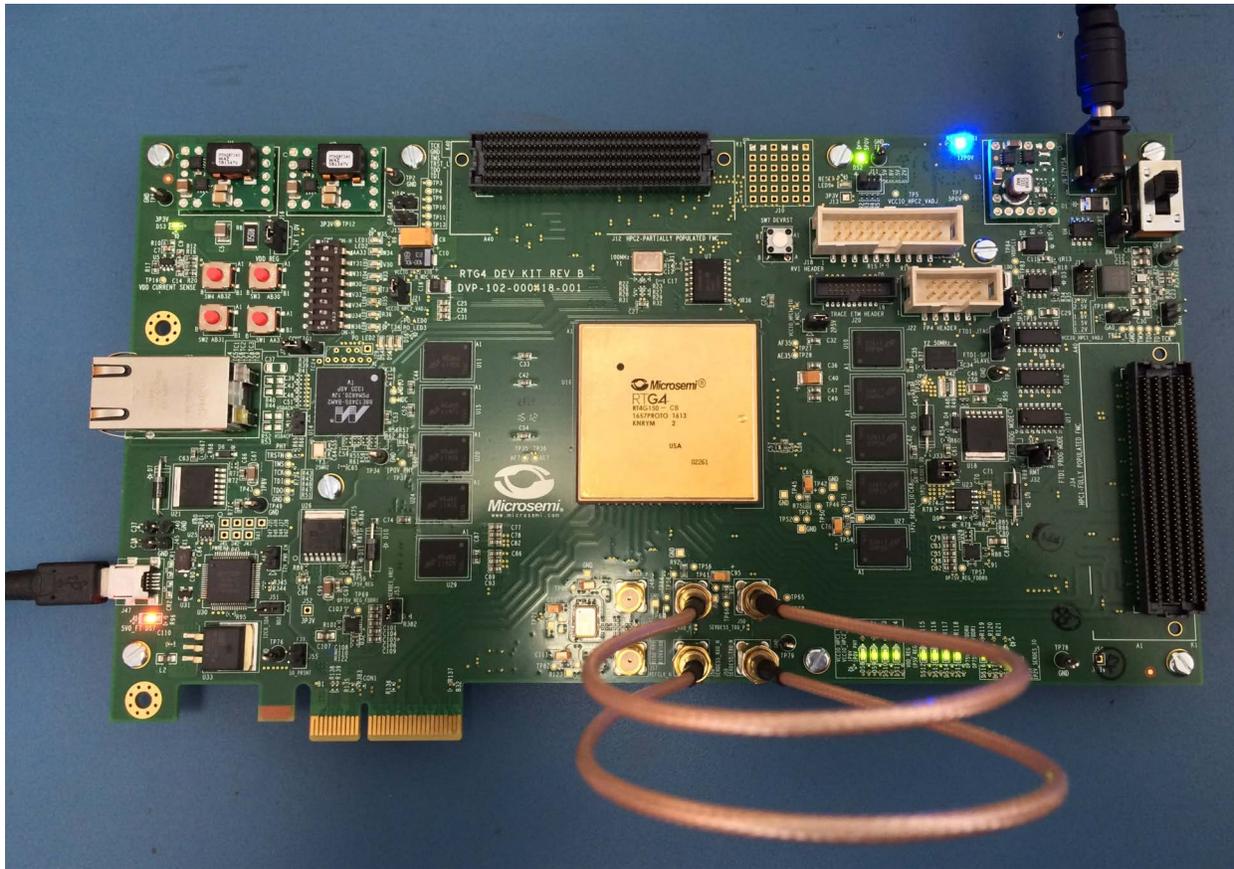


Figure 10 • Board Setup



4. Install USB-UART drivers. USB-UART driver files - FT232R is available on the Microsemi website.

3.5 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 the local disk and unzip it, ensure that the Libero version is v(xx.x).
3. Open the Libero project using the Libero tool.
4. Click generate bitstream option in the Libero tool.

Observe the synthesis, map, and implementation reports.

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

3.6 Programming the Device

To program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to [Appendix 1: Programming the Device Using FlashPro Express](#), page 20.

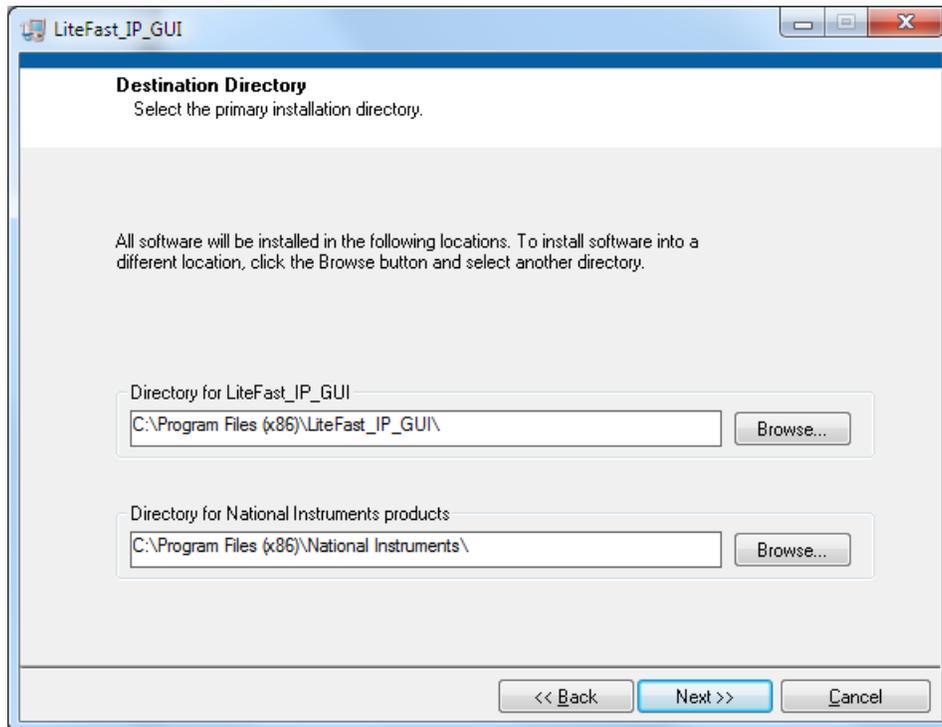
3.7 Executing the Demo Design

3.7.1 LiteFast Demo GUI Installation

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

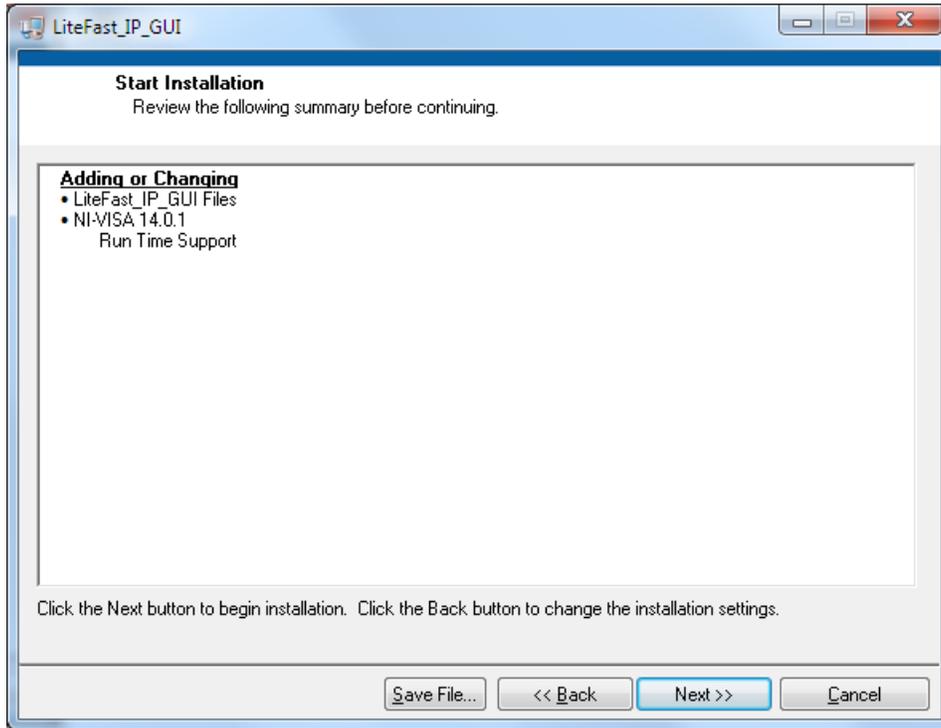
1. Download the GUI installation files:
http://soc.microsemi.com/download/rsc/?f=rtg4_dg0729_df/GUI
2. Open **GUI_Installer>Volume>setup.exe**.
3. When prompted, click **Yes**.
4. On the LiteFast Destination Directory window, click **Next**, as shown in the following figure. Default locations are displayed.

Figure 11 • LiteFast GUI Setup Window



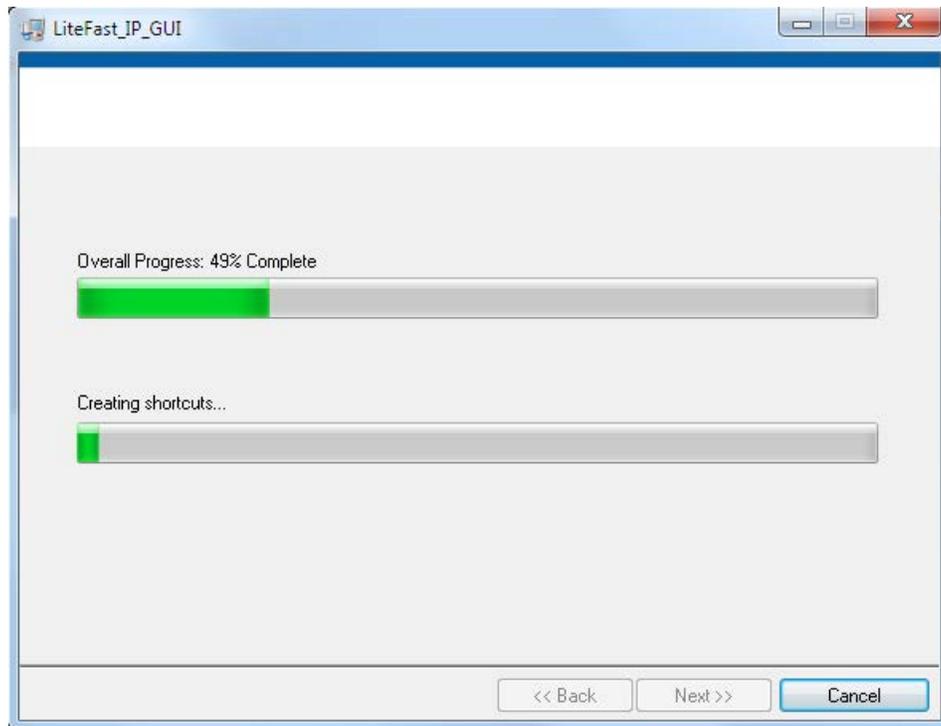
5. On the LiteFast Start Installation window, click **Next**, as shown in the following figure.

Figure 12 • LiteFast GUI Installation



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

Figure 13 • LiteFast GUI Setup Progress Bar



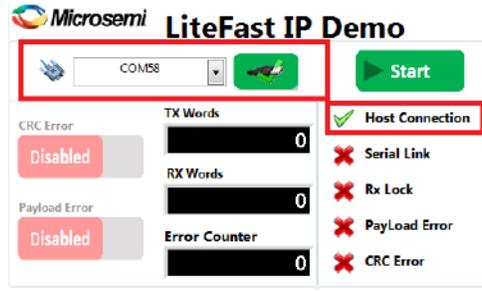
- If the installation is successful, Installation Complete message appears. Click **Finish**. Restart the computer before using the installed GUI.

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

3.7.2 LiteFast Demo GUI Description and Usage

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

Figure 14 • LiteFast GUI Window



Note: Hover the mouse pointer on each option 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

GUI Options	Description	Expectation from user
Drop-down COM port list	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 the dropdown list and clicking connect.	Unless the GUI is not auto connecting, the user doesn't have to perform any activity. This functionality is auto-updated by the GUI.

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

The following table describes the Push Buttons in the GUI.

Table 4 • Push Buttons in the GUI

GUI Options	Description	Expectation from user
Start/Stop	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.	Unless this button is pressed, the demo design will not be operative. User needs to monitor the core functionality.

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

Table 5 • Status Signals on the GUI

GUI Options	Description	Expectation from user
Host Connection	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.	User needs to check for the Green Mark before starting any functionality check on the GUI.
Serial Link	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.	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.
Rx Lock	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 subsequently 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.	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.
Payload Error	This indicates the errors received. Green mark: This indicates that there are no pay load errors detected. The data received is same as the data transmitted. Red cross mark: This indicates that the data received has injected errors.	User needs to monitor the functionality. Debug is not expected from the user. This error is injected into the data packets when the Payload Error slide bar is enabled. It increments the Error Count display.
CRC Errors	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.	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.

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

GUI Data Window	Description	Expectation from user
RX Words	Shows the number of received data words. This number rolls over after 65535 words.	User needs to monitor the number of Rx words received.
TX Words	Shows the number of transmitted data words. This number rolls over after 65535 words.	User needs to monitor the number of Tx words transmitted.
Error Counter	It indicates the packet loss and rolls over after 65535 words.	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 because the internal counters are read slowly, but the internal logic is accurate.

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

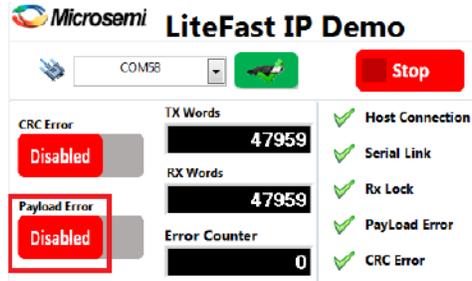
The following table lists down the slider bars available on GUI.

Table 7 • Slider Bars Available on GUI

GUI Slider Bars	Description	Expectation from user
Payload Error Slider	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.	User needs to monitor the functionality of the error injection and error detection by the core on GUI.
CRC Error Slider	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.	User needs to monitor the functionality of the CRC error injection and error detection by the core.

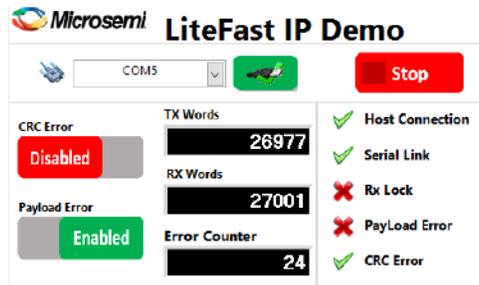
The following figure shows how the GUI looks like during an error free operation of the LiteFast demo system.

Figure 15 • Connected LiteFast GUI



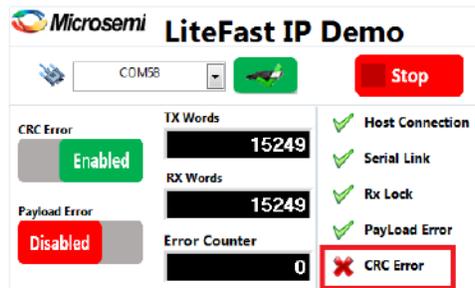
The following figure shows how the GUI looks like during a payload error injection of the LiteFast demo system.

Figure 16 • GUI Payload Error Injection



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

Figure 17 • 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.

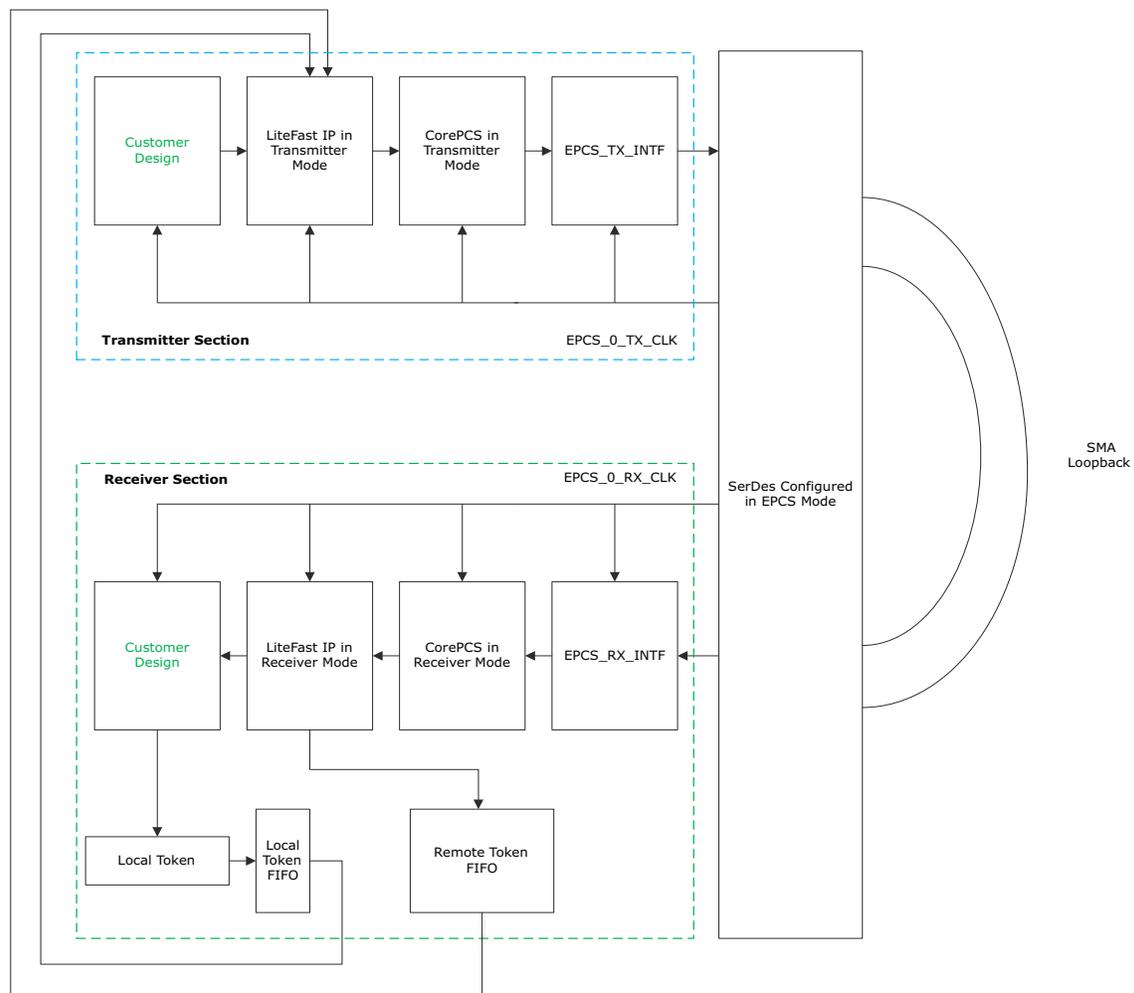
4.1 LiteFast Transmitter Section

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 18 • 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 SmartDesign in the Libero v(xx.x) SP2.
2. Add the LiteFast IP from the IP catalog.
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 the 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 bitstream.
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

Type	Used	Total	Percentage
4LUT	2848	151824	1.88
DFF	2209	151824	1.45
I/O Register	0	2154	0.00
User I/O	3	720	0.42
-- Single-ended I/O	3	720	0.42
-- Differential I/O Pairs	0	360	0.00
RAM64x18	2	210	0.95
RAM1K18	2	209	0.96
MACC	0	462	0.00
H-Chip Globals	7	48	14.58
CCC	1	8	12.50
RCOSC_50MHz	1	1	100.00
SYSRESET	1	1	100.00
SERDESIF Blocks	1	6	16.67
FDDR	0	2	0.00
GRESET	1	1	100.00
RGRESET	2	206	0.97

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

5 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

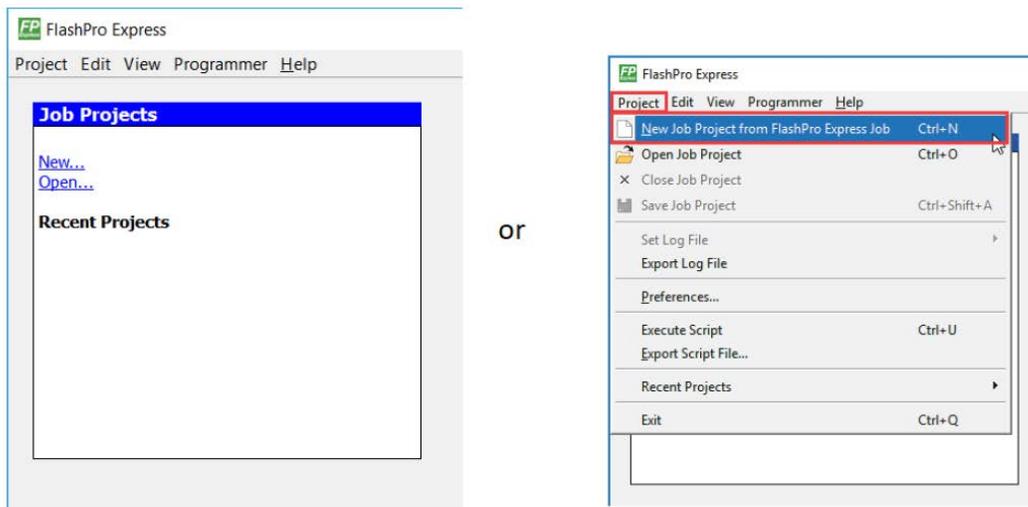
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide*
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

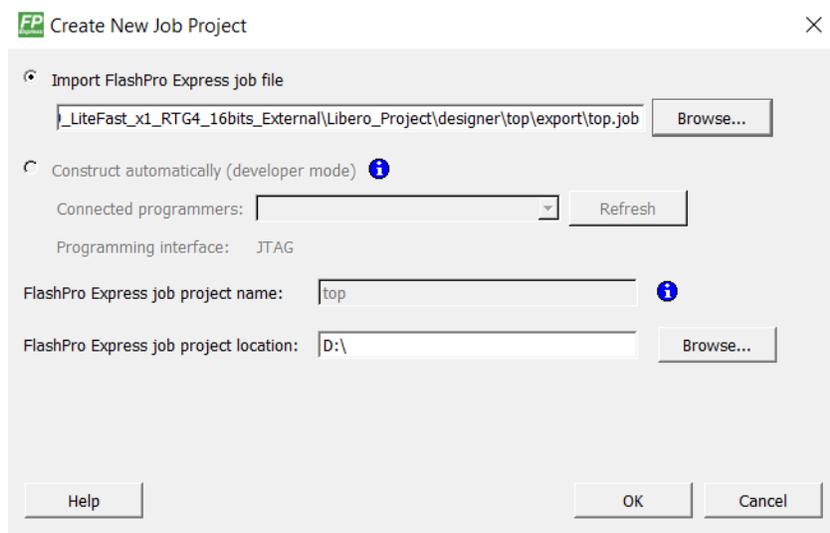
3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 19 • FlashPro Express Job Project

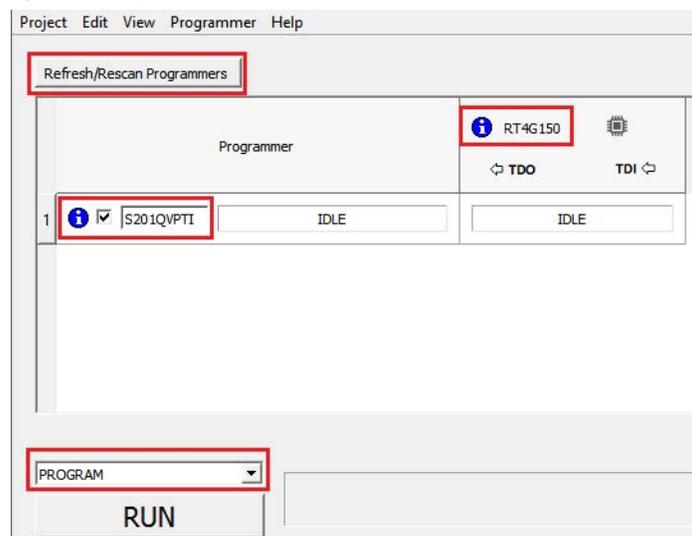


8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
 - **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:

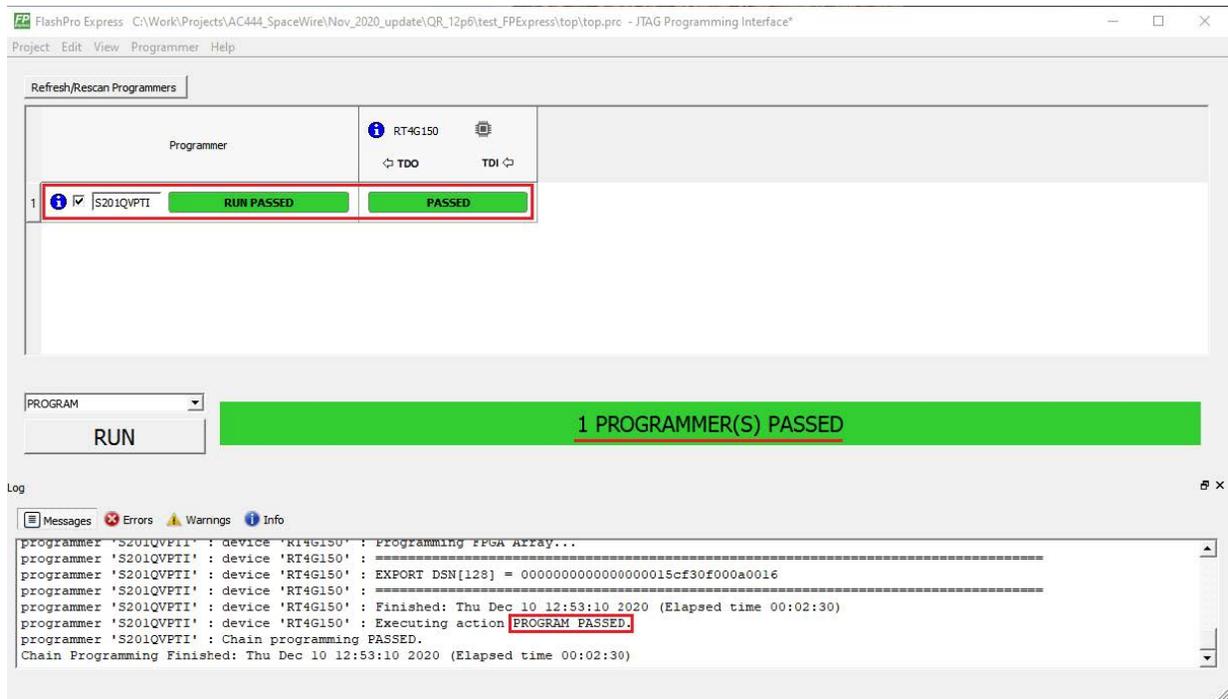
```
<download_folder>\rtg4_dg0729_df\Programming_Job
```
 - **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 20 • New Job Project from FlashPro Express Job

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

Figure 21 • Programming the Device

11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 22 • FlashPro Express—RUN PASSED

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

6 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:

- `rtg4_dg0729_df/DG729_LiteFast_RTG4_8bits_Ext_Loopback/TCL_Scripts/readme.txt`.
- `rtg4_dg0729_df/DG729_LiteFast_RTG4_8bits_Int_Loopback/TCL_Scripts/readme.txt`.
- `rtg4_dg0729_df/DG729_LiteFast_x1_RTG4_16bits_External/TCL_Scripts/readme.txt`.
- `rtg4_dg0729_df/DG729_LiteFast_x1_RTG4_16bits_Internal/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.