DG0624
Demo Guide
RTG4 FPGA SERDES EPCS Protocol Design

Microsemi
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

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.

50200624. 7.0 10/21

Contents

1 Revision History . . . 1
1.1 Revision 7.0 . . . 1
1.2 Revision 6.0 . . . 1
1.3 Revision 5.0 . . . 1
1.4 Revision 4.0 . . . 1
1.5 Revision 3.0 . . . 1
1.6 Revision 2.0 . . . 1
1.7 Revision 1.0 . . . 1

2 RTG4 FPGA SERDES EPCS Protocol Design . . . 2
2.1 Design Requirements . . . 3
2.2 Prerequisites . . . 3
2.3 Demo Design . . . 3
2.3.1 Introduction . . . 3
2.3.2 Description . . . 5
2.4 Setting Up the Demo Design . . . 7
2.4.1 Setting Up the Board . . . 7
2.4.2 Programming the Demo Design . . . 8
2.4.3 Installing the Demo GUI . . . 9
2.4.4 Running the Demo Design . . . 11
2.5 Conclusion . . . 12

3 Appendix 1: Programming the Device Using FlashPro Express . . . 13

4 Appendix 2: Running the TCL Script . . . 16

5 Appendix 3: Using RTG4 for Customer Design . . . 17
5.1 Transmitter Section . . . 17
5.2 Receiver Section . . . 17

6 Appendix 4: Simulating the Design . . . 18

7 Appendix 5: Verifying Timing using SmartTime . . . 20

8 Appendix 6: GUI Status Signal . . . 21

Figures

Figure 1 Design Files Top-Level Structure . . . 4
Figure 2 RTG4 Demo Design Files Top-Level Structure . . . 4
Figure 3 Demo Design Block Diagram . . . 5
Figure 4 TX and RX Interface RTL Blocks . . . 6
Figure 5 RTG4 Development Kit Board . . . 8
Figure 6 GUI Set Up Window . . . 9
Figure 7 GUI Setup Progress Bar . . . 10
Figure 8 EPCS Demo GUI Window . . . 11
Figure 9 Sample GUI Window . . . 12
Figure 10 FlashPro Express Job Project . . . 13
Figure 11 New Job Project from FlashPro Express Job . . . 14
Figure 12 Programming the Device . . . 14
Figure 13 FlashPro Express—RUN PASSED . . . 15
Figure 14 Replacing Demo Design with Customer Design . . . 17
Figure 15 Organizing Simulation Testbench in Project . . . 18
Figure 16 Simulating the Design . . . 19
Figure 17 Simulation Waveform Window . . . 19
Figure 18 Verify Timing . . . 20
Figure 19 SmartTime . . . 20
Figure 20 SmartTime Session . . . 20

Tables

Table 1 Design Requirements 3
Table 2 Jumper Settings 7
Table 3 GUI Status Signals 21

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 7.0

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

- Updated Figure 2 on page 4.
- Added a note in Appendix 4: Simulating the Design, page 18 after the Figure 15, page 18.

1.2 Revision 6.0

Updated "Setting Up the Board" section step 4, CDM_2.08.24_WHQL_Certified.zip web link.

1.3 Revision 5.0

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

- Added Appendix 1: Programming the Device Using FlashPro Express, page 13.
- Added Appendix 2: Running the TCL Script, page 16.
- Removed the references to Libero version numbers.

1.4 Revision 4.0

Updated the document for Libero v11.8 SP1 software release.

1.5 Revision 3.0

Updated the document for Libero v11.8 SP2 software release.

1.6 Revision 2.0

Updated the document for Libero v11.7 software release.

1.7 Revision 1.0

Revision 1.0 is the first publication of this document.

2 RTG4 FPGA SERDES EPCS Protocol Design

The RTG4™ devices have embedded high-speed SERDES blocks that can support data rates between 1 Gbps and 3.125 Gbps. The high-speed serial interface block supports several serial communication standards. The RTG4 SERDES block integrates several functional blocks to support multiple high-speed serial protocols within the FPGA.

The EPCS mode exposes the SERDES lanes directly to the fabric and configures the SERDES block in physical media attachment (PMA) only mode. In the EPCS mode, the peripheral component interconnect express (PCIe®) and ten Gigabit attachment unit interface (XAUI) PCS logic in the SERDES block is bypassed. However, the PCS logic can be implemented in the FPGA fabric, and the EPCS interface signals of the SERDES block can be connected to the user protocol. This allows any user-defined high-speed serial protocol to be implemented in the RTG4 device.

In conjunction with EPCS, the available CorePCS IP module supports programmable 8B10B encoding and decoding. 8B10B is commonly used in protocols that are not included in the SERDES block by the Microsemi system-on-chip (SoC) high-speed SERDES interface. Therefore, the CorePCS IP module can be used with these protocols. It can be configured as a transmitter only, receiver only, or both transmitter and receiver. Word alignment support is included in the receiver. It can also be configured to support 10-bit or 20-bit EPCS data. For more information about this, refer to *CorePCS Handbook*.

The SERDES blocks are completely configurable. Initial register settings of the SERDES blocks are required at run-time. This demonstration design initializes the SERDES configuration registers using the SERDES block that has a built-in initialization state machine. The state machine loads the SERDES block with the correct register settings on power up or assertion of DEVRST.

This demo describes the following:

- EPCS interface of the RTG4 device with High-Speed Serial Interface (PCIe, EPCS, XAUI) with initialization.
- Using CorePCS IP modules for customized applications.

2.1 Design Requirements

Table 1 lists the design requirements to run the design.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 Development Kit: • USB 2.0 cable • 12 V, 5A AC power adapter and cords	Rev B or later
SMA Male to SMA Male Loopback Cables	2- SMA Male-to-SMA Male Precision Cables, such as Pasternack Industries part number PE39429-12 (or equivalent)
Host PC or Laptop	64-bit Windows 7 and 10
Software	
Libero® System-on-Chip (SoC)	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
GUI Software	
Host PC Drivers for FlashPro5	USB to UART drivers
Framework	Microsoft .NET Framework 4 client for launching demo GUI

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you start:

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*

2.3 Demo Design

2.3.1 Introduction

The demo design files are available for download from the following path in the Microsemi website: *http://soc.microsemi.com/download/rsc/?f=rtg4_dg0624_df*

The demo design files include the following:

- GUI
- Libero_Project
- Programming_Job
- TCL_Scripts

Figure 1 • **Design Files Top-Level Structure**

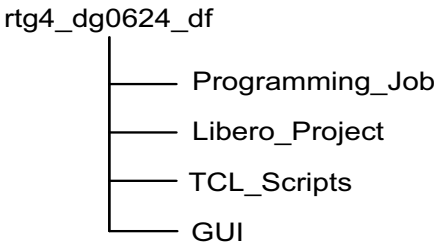
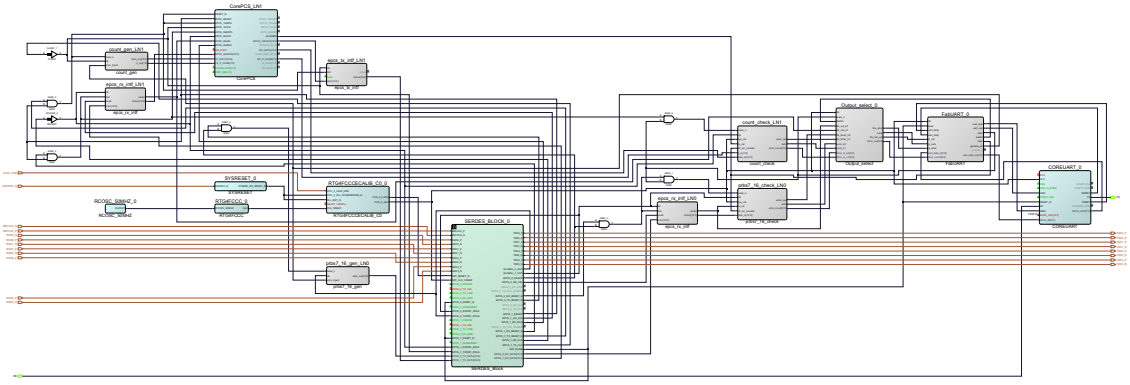


Figure 2 shows the top-level structure of the RTG4 design files.

Figure 2 • **RTG4 Demo Design Files Top-Level Structure**

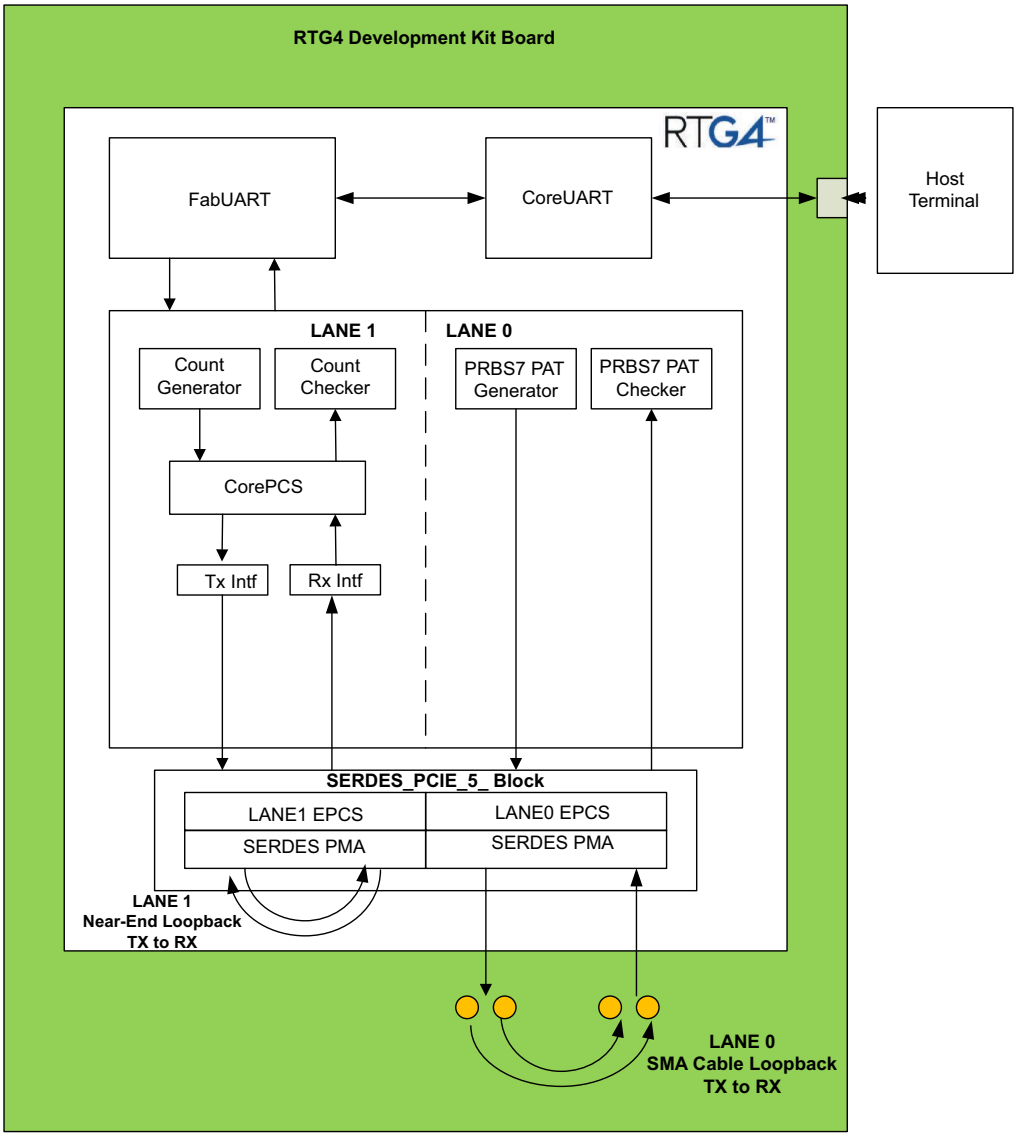


This example design demonstrates transmitting a pseudo-random binary sequence (PRBS) or counting pattern over the RTG4 high-speed SERDES interface. The SERDES block is configured for 2.5 Gbps operational speed. The PRBS pattern is sent over Lane 0, and a counting 8B10B encoded pattern is used with Lane 1 of the SERDES block.

- Demo 1: Lane 0 traffic is sent directly from the fabric based PRBS generator to the SERDES block and off-chip to test SMA connections. Input SMA connectors are routed to the SERDES receiver pins to bring the data back into the device to the fabric based pattern checker. External SMA cables complete the TX to RX loopback circuit.
- Demo 2: Lane 1 includes a pattern generator and checker that also utilizes the CorePCS IP module in the data path. The CorePCS IP module provides simple 8b/10b encoding and decoding functionality. The loopback is done on-chip.

Note: Lane 2 and Lane 3 are not used.

Figure 3 • **Demo Design Block Diagram**



2.3.2 Description

The hardware design for the implementation includes a PRBS and count pattern generator, PRBS sequence and count pattern checker, error counter, RX and TX fabric interface blocks, delay line, UART and output select control and high-speed serial interface block connected to the RTG4 SERDES block. Each of these blocks is explained in the following sections:

- PRBS7 Generator, page 6
- Count Generator, page 6
- PRBS7 Checker, page 6
- Count Checker, page 6
- RX and TX Interface, page 6

2.3.2.1 PRBS7 Generator

The generator implements the PRBS7 polynomial (x^7+x^6+1) and generates a continuous sequence of PRBS7 patterns of 10 bits each. Each 10-bit transmission from the generator occurs at a frequency of 39.3 MHz. The PRBS generator module runs at 125 MHz.

2.3.2.2 Count Generator

The count generator module implements a count pattern used to drive the CorePCS 8b10b encoder. Packets created in the count generator are separated by a K28.5 character. The payload of the packet is a simple counting pattern.

2.3.2.3 PRBS7 Checker

The PRBS7 checker checks for valid PRBS sequences. If the received sequence does not match with the one transmitted by the generator, the checker indicates an error. The checker also implements an error counter, which is incremented for each error in the received PRBS sequence.

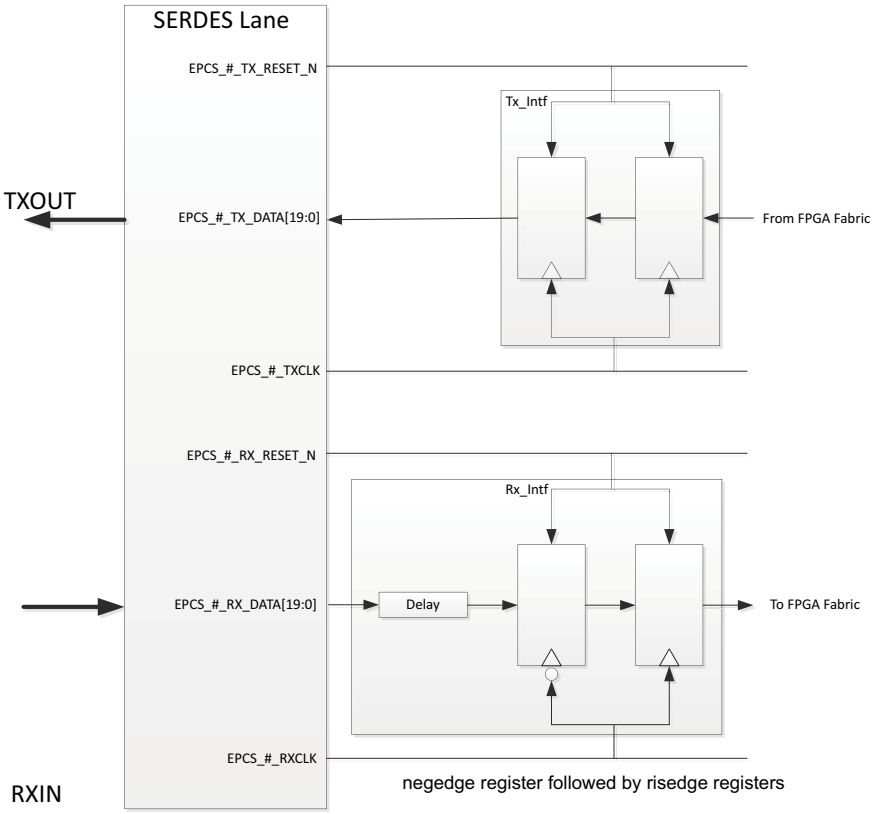
2.3.2.4 Count Checker

The count checker module checks for a valid count pattern received from the CorePCS 8b10b decoder. The count checker checks each packet for the embedded count pattern used as the payload of the packet.

2.3.2.5 RX and TX Interface

RX and TX interface modules manage the timing relationships of the clock and data from the EPCS interface to the FPGA fabric when the global clocks are not used.

Figure 4 • **TX and RX Interface RTL Blocks**



2.3.2.6 CoreUART, FabUART, and Output Select Modules

The COREUART module communicates with the UART interface on the RTG4 Development Kit. FabUART and Output select modules are glue logic modules to connect the PRBS generator and checker control and error reporting signals to the GUI that communicates to the device over UART. The Output Select block multiplexes status signals like Error, Error count, and Lock signals from Lane 0 and Lane 1. Depending on the Lane selection, it feeds the corresponding status signals onto the UART.

2.3.2.7 SERDES

The RTG4 high-speed SERDES is a hard IP block on-chip that supports rates up to 3.125 Gbps. The SERDES block offers embedded protocol support for PCIe and XAUI. The SERDES block also supports the EPCS interface, which can be used for custom protocols. This Demo uses the SERDES block in the EPCS protocol. For more information about the SERDES block, refer to *UG0567: RTG4 FPGA High Speed Serial Interfaces User Guide.* In this design, the SERDESIF block is configured to be 20-bit wide, 125 MHz REFCLK, and 2.5 Gbps.

2.3.2.8 Clocking

The two different types of clock domains in the EPCS demo design are:

- Control Plane Clock: Used for initialization of the SERDES block, UART, and Output Select. The control plane clock is sourced by the Fabric PLL and is passed to the control blocks at a 50 MHz rate.
- EPCS interface output Clock: Each SERDES lane provides an output clock for the transmitter and the receiver. The transmit clock is used to clock epcs_tx_intf and reminder of the transmit data path. The receive clock is used to clock epcs_rx_intf and the remainder of the receive path.

2.4 Setting Up the Demo Design

2.4.1 Setting Up the Board

The following steps describe how to set up the hardware demo for the RTG4 Development Kit:

1. Connect the jumpers on the board, as shown in Table 2.

The following table lists the jumper settings.

Table 2 • **Jumper Settings**

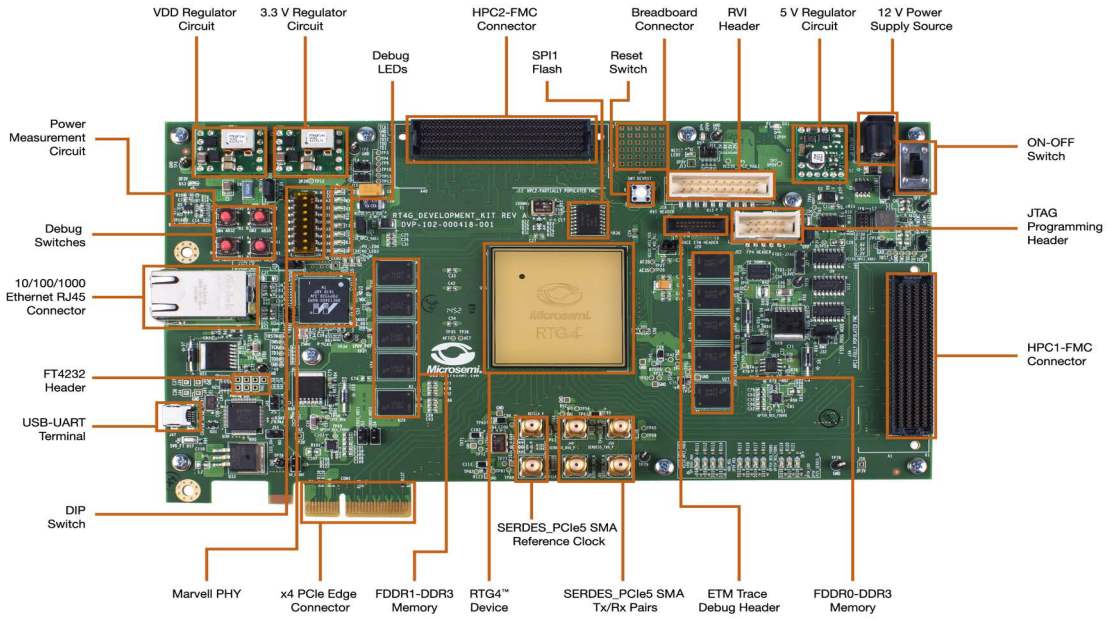
Jumper	Pin (From)	Pin (To)	Comments
J11, J17, J19, J23, J26, J21, J32, J27, J28	1	2	Default
J16	2	3	Default
J33	1 3	2 4	Default

Note: Ensure that the power supply switch, **SW6** is switched OFF while connecting the jumpers on the RTG4 Development Kit.

2. Connect the host PC or Laptop to the J47 connector using the USB min-B cable. This serves as both the FlashPro5 programmer interface and the UART control interface for the demo GUI.
3. Connect 12 V 6 A-power jack to the J9 power connector.
4. Ensure that the USB to UART bridge drivers are automatically detected. If USB to UART bridge drivers are not installed, download and install the drivers from: *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*
5. Install the SMA Male to SMA Male connectors (J49 to J50 and J58 to J59).

Figure 5 shows the RTG4 Development Kit board.

Figure 5 • **RTG4 Development Kit Board**



2.4.2 Programming the Demo Design

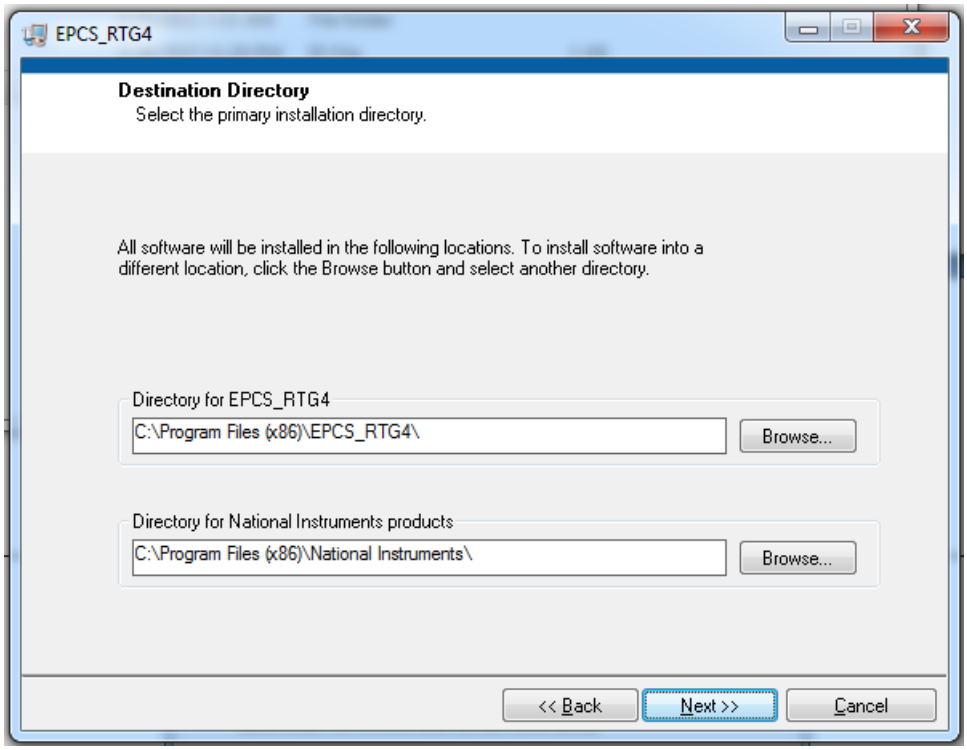
To program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 13.

2.4.3 Installing the Demo GUI

The following steps describe how to run the installer if the GUI is used for the first time:

1. Download the design files from: *http://soc.microsemi.com/download/rsc/?f=rtg4_dg0624_df*
2. Open **GUI_Installer** > **Volume** > **setup.exe**.
3. Click **Yes** for any message from **User Account Control**. The Destination Directory window is displayed with the default locations, as shown in Figure 6.
4. Click **Next**.

Figure 6 • **GUI Set Up Window**

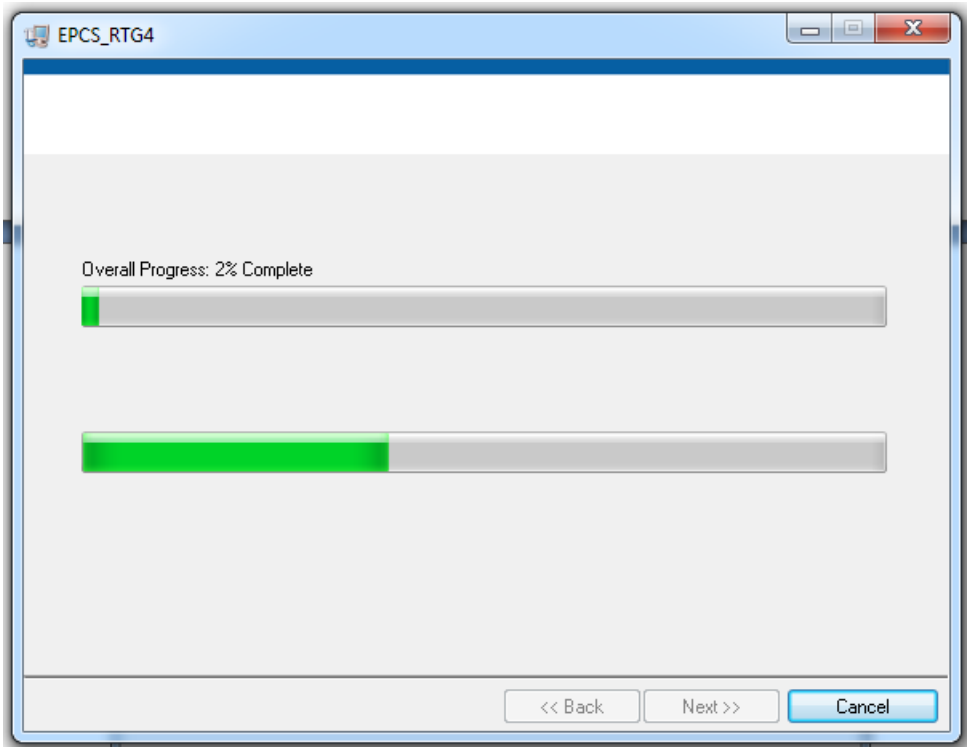


5. Follow the instructions in the GUI to start the installation the installation.

Note: Accept the license agreement and click **Next** in the Summary dialog box.

A progress bar shows the progress of installation, as shown in Figure 7.

Figure 7 • **GUI Setup Progress Bar**



6. Wait for the installation to complete. After successful installation, **Installation Complete** message is displayed.
7. Click **Finish**.
8. Restart the computer before using the installed GUI.

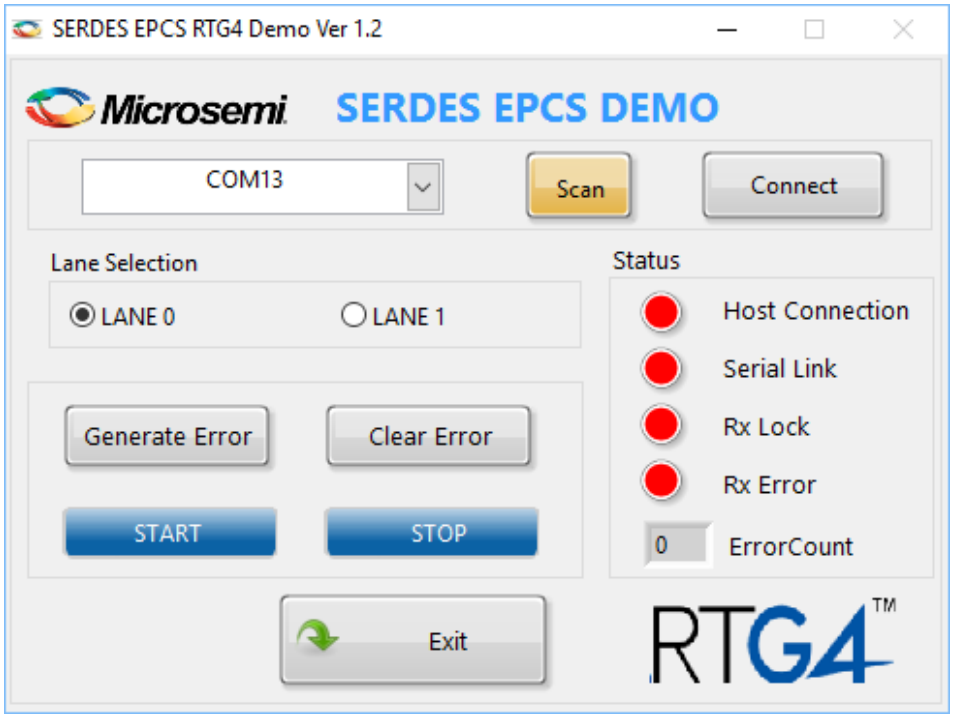
2.4.4 Running the Demo Design

The following steps describe how to run the demo design:

1. Open **Programs > EPCS_RTG4**.

Figure 8 shows the GUI window.

Figure 8 • **EPCS Demo GUI Window**

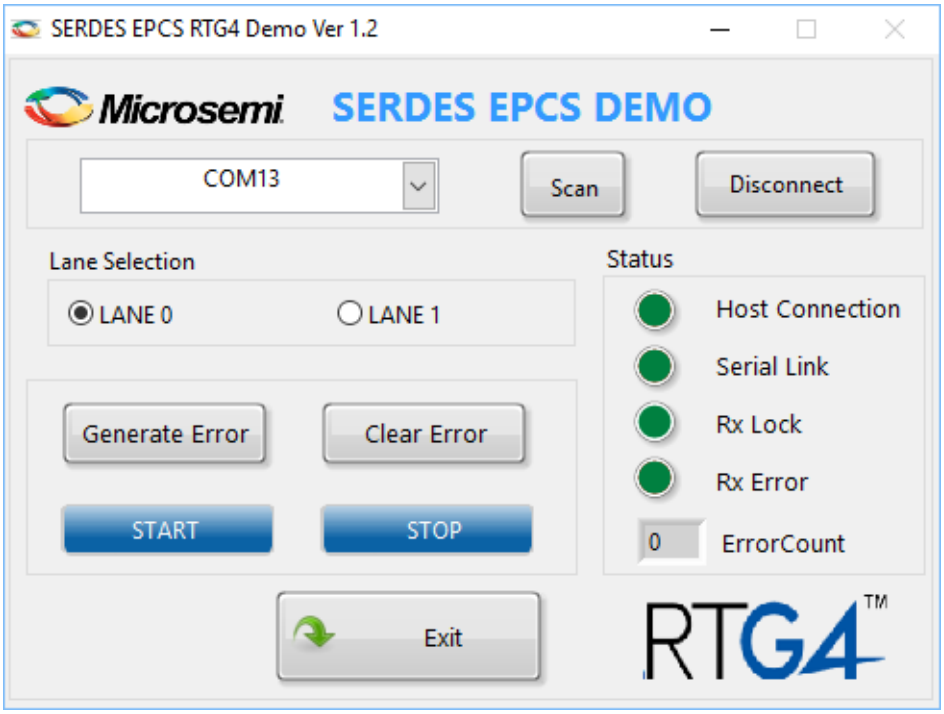


The drop-down list for ports has the list of serial ports available on the Host PC. The working ports are enabled and the unavailable ports are grayed out.

2. Click **Connect** to connect the Host PC to the hardware through the selected port. The status signals indicate the status of the complete system operation.
3. Select the desired lane. Either Lane 0 or Lane 1 can be individually tested.
4. Click **Start** to start the EPCS Demo. The PRBS7 data (or Count data) is sent over the serial transmit link. It is then received by the receiver and checked for any errors. The status can be monitored using the status signals in the GUI at any time. For more information about the status signals, refer to Appendix 6: GUI Status Signal, page 21.
5. Click **Stop** to stop the EPCS demo.
6. Click **Exit** to exit the GUI.

Figure 9 shows a sample GUI window during an error free operation of the SERDES EPCS demo.

Figure 9 • **Sample GUI Window**



2.5 Conclusion

This demo describes the EPCS interface of the RTG4 device, the use of the self-initializing SERDES block and CorePCS IP modules, and how to use them for customized applications. This demonstration design allows users to see an actual implementation of the RTG4 SERDES block on the development kit. The design shows data traffic into and out of the block. It can be used for signal quality analysis of the SERDES transmitters as well as demonstrate error free data looped between the RTG4 transmitter and receiver of the SERDES block.

3 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.

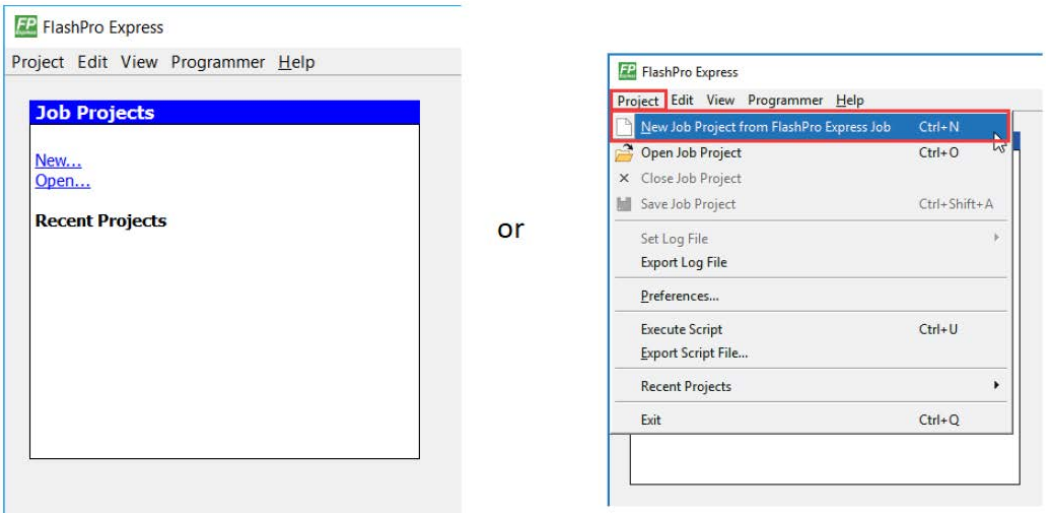
To program the device, perform the following steps:

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 Figure 10.

Figure 10 • **FlashPro Express Job Project**



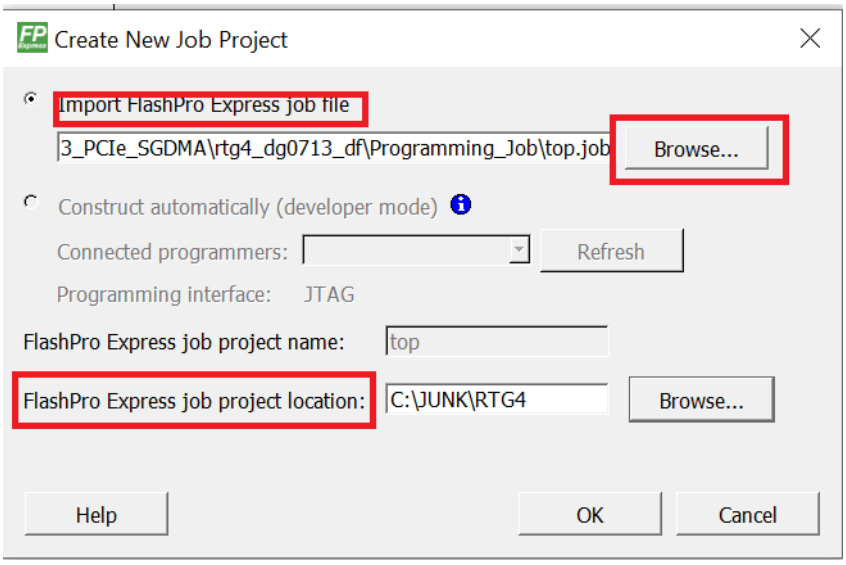
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **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_dg0624_df\Programming_Job`

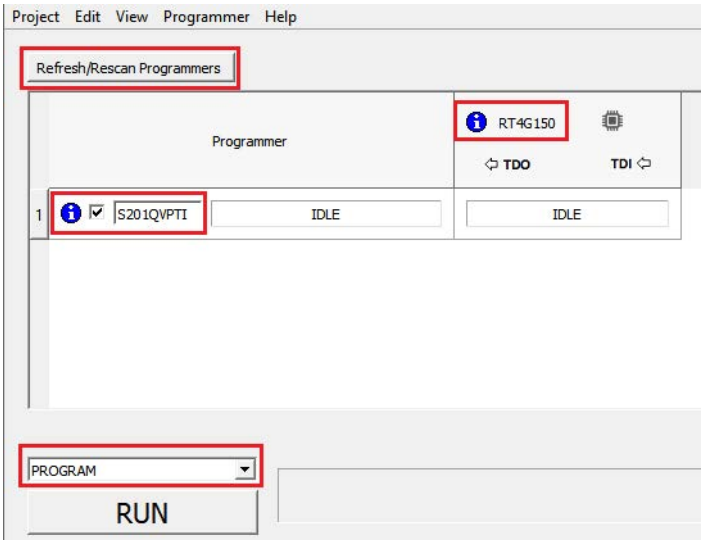
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 11 • **New Job Project from FlashPro Express Job**



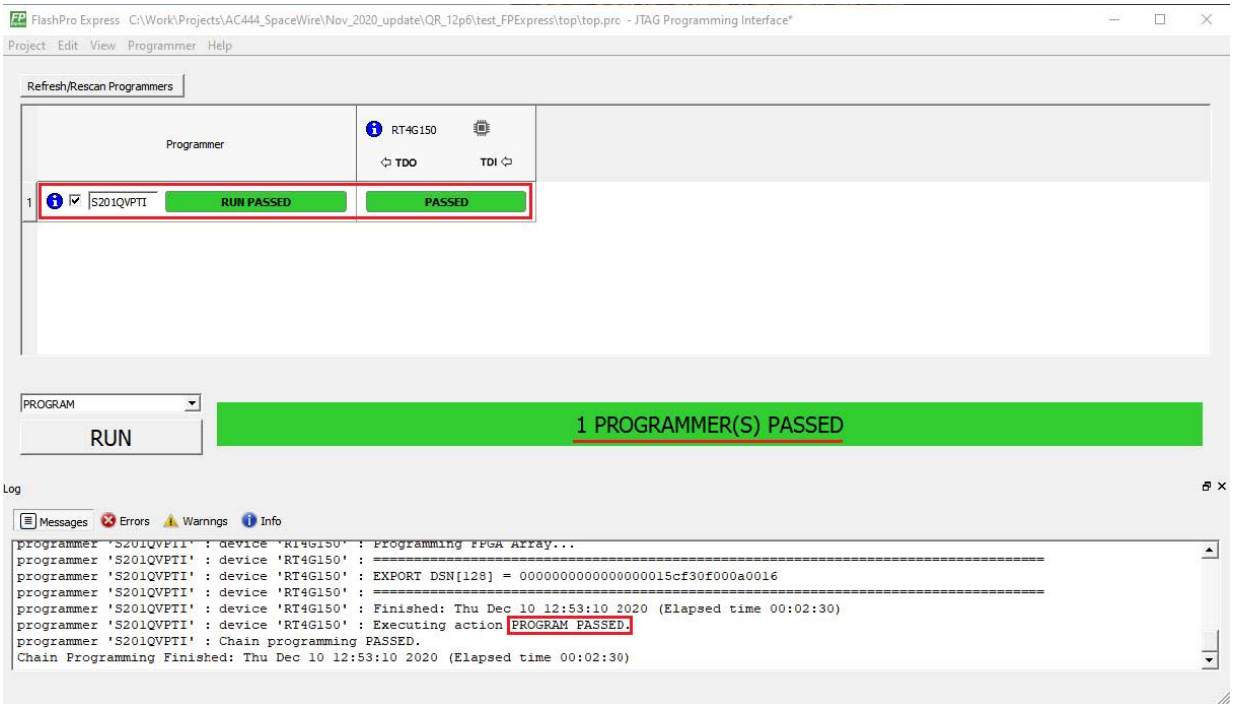
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in Figure 12. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 12 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in Figure 13.

Figure 13 • **FlashPro Express—RUN PASSED**



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

4 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_dg0624_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.

5 Appendix 3: Using RTG4 for Customer Design

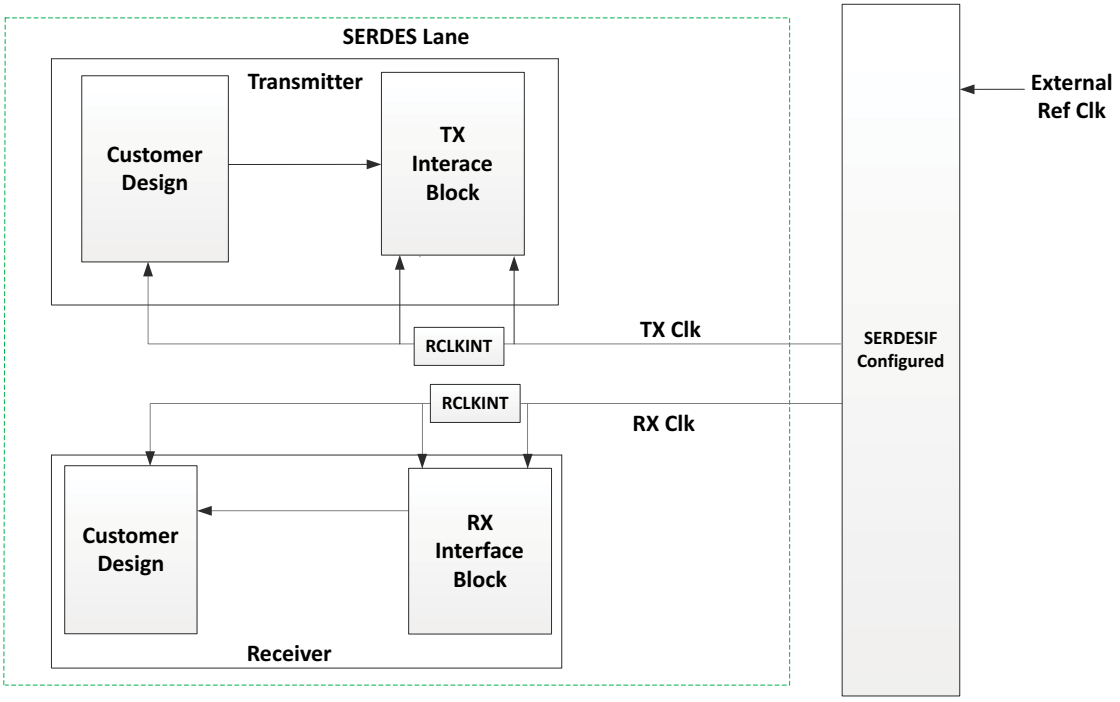
5.1 Transmitter Section

The PRBS7 generator in the transmitter section can be replaced with the customer data generator. The data generator is interfaced with the TX Interface block as shown in Figure 14.

5.2 Receiver Section

The PRBS7 checker in the receiver section can be replaced with the data receiver in the customer design. The data receiver takes input from the RX Interface, as shown in Figure 14.

Figure 14 • **Replacing Demo Design with Customer Design**

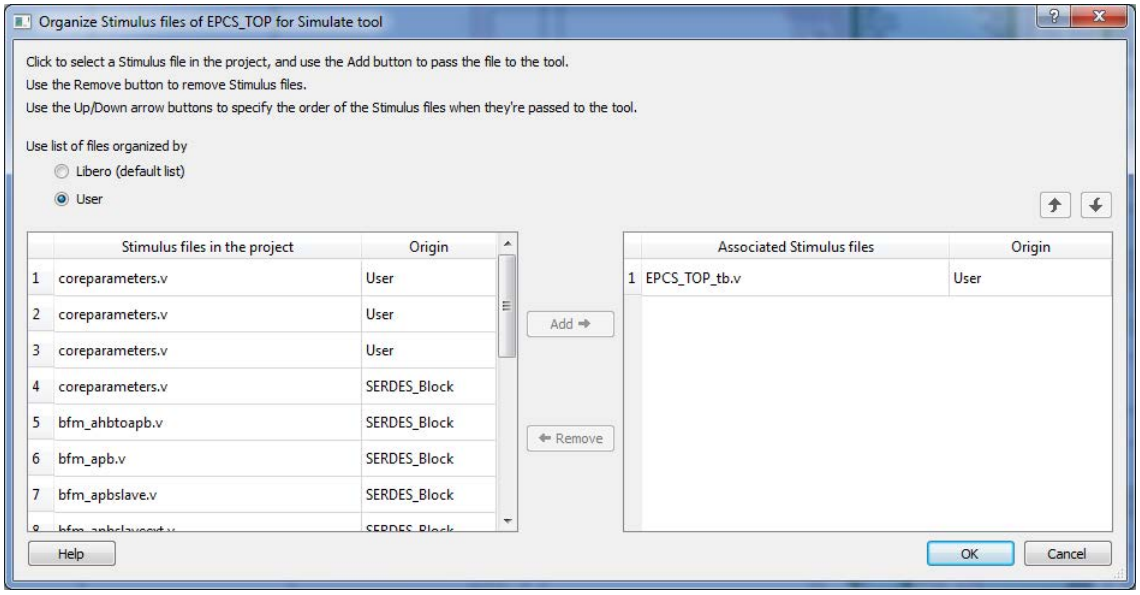


This demo is targeted for RTG4150. It is recommended to use a clock resource to reduce the clock injection time into the fabric. This is done by instantiating an RCLKINT library element on the clock outputs of SERDESIF EPCS TX_CLK and RX_CLK.

The interface blocks for the transmitter and receiver are also recommended to achieve timing closure. These blocks employ a scheme specifically designed for the RTG4 family and optimize the interface for both setups and hold when not using the global clocks. These blocks must be used exactly from this demo design in all EPCS designs. Verilog HDL is used in this tutorial. VHDL modules are available in the SOURCE Directory.

6 Appendix 4: Simulating the Design

Figure 15 • **Organizing Simulation Testbench in Project**



Note: In the `Sync_fsm.v` file, under corePCS change the value for the parameter TRIGGER to 0100 from 1010.

The following steps describe how to simulate the design:

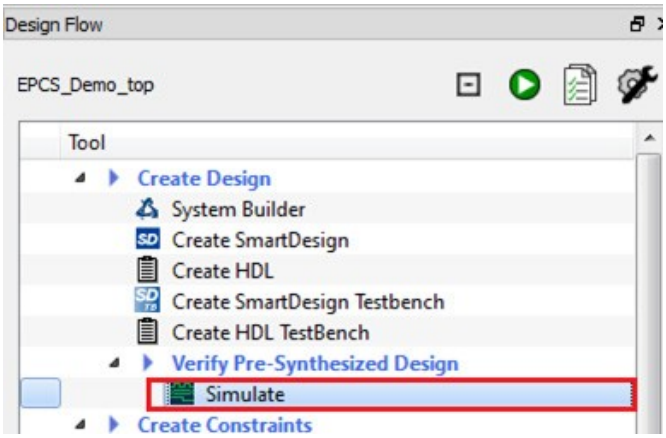
1. Import the `testbench_tb.v` file to the Libero project.
2. Right-click **Simulate** and select **Organize Input files** > **Organize Stimulus Files** to setup the `EPCS_TOP_TB.v` file for simulation, as shown in Figure 15.
3. Import the `wave.do` file to the **Simulation** folder of the Libero project.

Note: `wave.do` and `EPCS_TOP_TB.v` files are available in the **Test benches** folder of **EPCS_Demo** project.

4. Go to **Project** > **Project Settings** > **waveforms** and select **Include DO file** check box.
5. Change the **Simulation runtime** to **2 us** using the **Do File** option under **Project Settings**. In the **Simulation options > DO** file window, ensure the Testbench module name and Top level instance name are proper, according to the testbench file.
6. Select `vsim command` under **Project Settings** and change the resolution to **1 ps**.
7. Click **Save** to save the settings.
8. Right-click **Simulate** in the **Libero Design Flow** and select **Open Interactively**.

The design is simulated through a test bench. The testbench simulates the high-speed serial interface block in the EPCS mode. To run the simulation, double-click **Simulate** under **Verify Pre- Synthesized Design** in the **Design Flow** tab of the Libero project, as shown in Figure 16.

Figure 16 • **Simulating the Design**



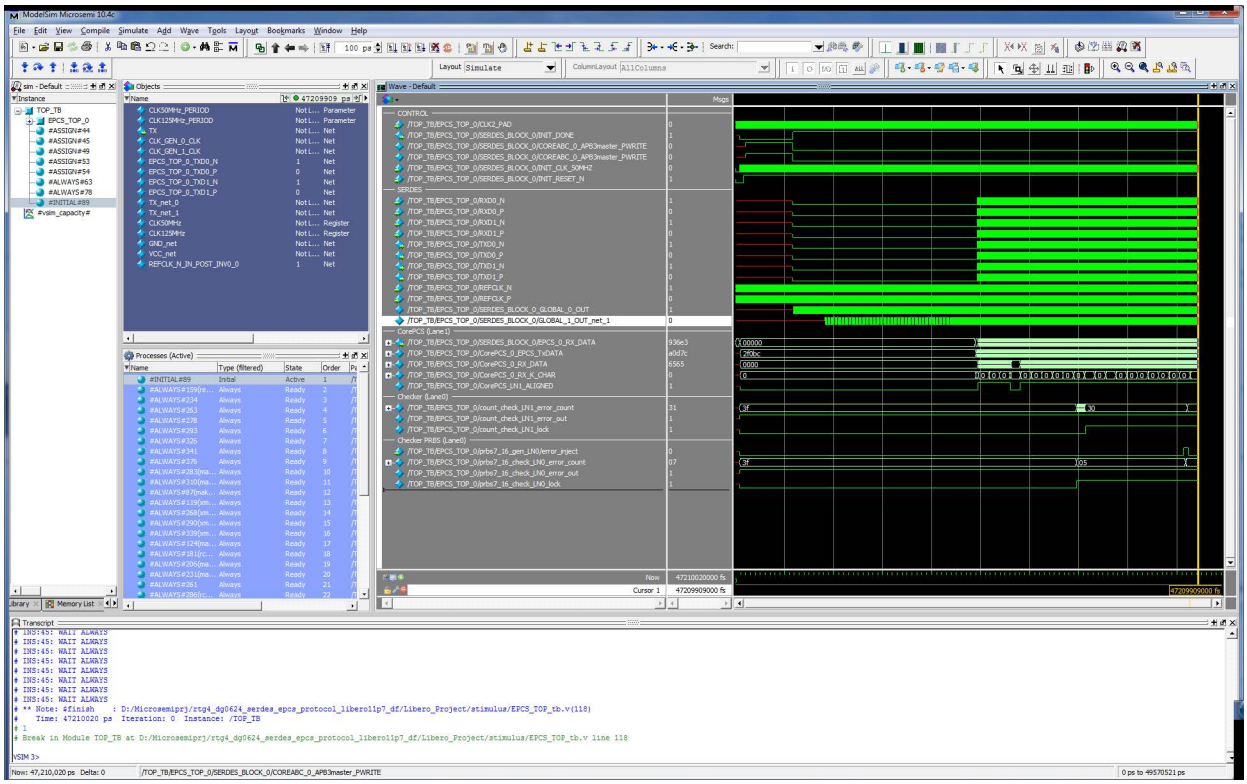
Observe the simulation results for Lane0 and Lane1.

The simulation automatically runs from the testbench and shows data on the lanes, generates and checks the results. The simulation posts messages to the log indicating the various steps of the SERDES initialization and operation.

Once the simulation moves to the operational phase, use **Run -all** to continue with the testbench.

After the simulation, the **Simulation Waveform window** is displayed, as shown in Figure 17.

Figure 17 • **Simulation Waveform Window**



7 Appendix 5: Verifying Timing using SmartTime

SmartTime is a gate-level static timing analysis tool. Using SmartTime, user can perform the following steps to complete a timing analysis of the design and ensure that all timing constraints are met. The design operates at the desired speed with the right amount of margin across all operating conditions. For more information, refer to *SmartTime for Libero SoC User's Guide*.

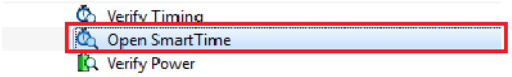
1. Double-click **Verify Timing**. The Libero software will perform a timing check and report any timing violations.

Figure 18 • **Verify Timing**

SmartTime is an interactive tool that can be invoked from the Design Flow pane.

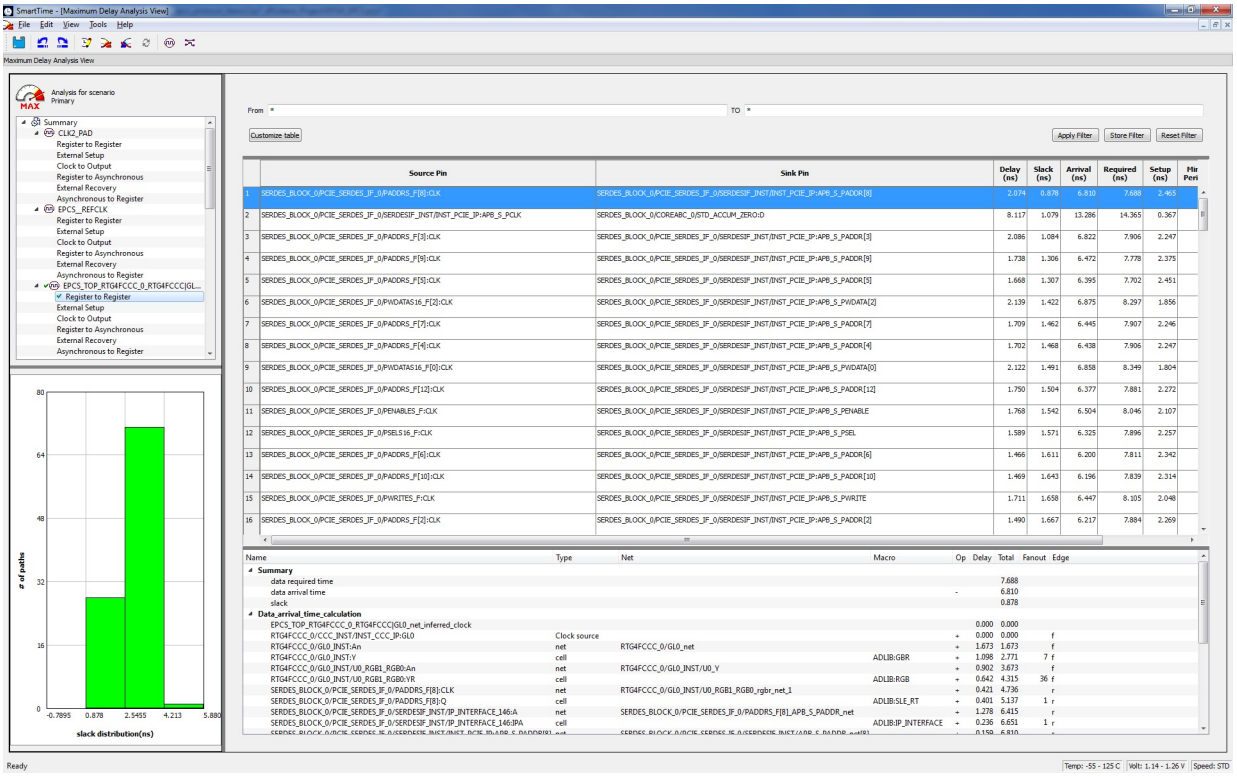
2. Double-click **Open SmartTime**.

Figure 19 • **SmartTime**



SmartTime window is displayed in Max Delay Analysis View, as shown in Figure 20, page 20. It shows the setup or hold time violations that cause design issues in the hardware.

Figure 20 • **SmartTime Session**



8 Appendix 6: GUI Status Signal

Table 3 shows the various status signals.

Table 3 • **GUI Status Signals**

Status Signal	Description
Host Connection	Indicator of COM port connection on the host PC. GREEN: COM port is connected. RED: COM port is disconnected.
Serial Link	Indicator of transmission link for serial data. GREEN: Link is up and running. RED: Link is down.
Rx Lock	Receiver lock. GREEN: The receiver receives a valid and error-free data. The receiver is locked to the PRBS7 or count sequences and the subsequently transmitted sequences can be successfully received. RED: The receiver receives an invalid data.
Rx Error	Indicates the status of the packets received. GREEN: Received packets are error-free. RED: A corrupted packet or any error is detected in the received PRBS7 or count sequences.
Error Count	Gives the count of errors detected in the received PRBS sequences.
Generate Error	Introduces errors in the transmission for debug purposes. Introduces error in the transmitted PRBS sequence, which increments the Error Count display.
Clear Error	Sets error count to zero.