TU0570
Tutorial
Implementing a SmartFusion2 and IGLOO2 SERDES EPCS Protocol Design

Microsemi
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.

50200570. 6.0 1/21

Contents

Figures

Tables

1 Revision History

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

1.1 Revision 6.0

Removed the references to Libero version numbers.

1.2 Revision 5.0

Updated the document for Libero v11.7 software release (SAR 75568).

1.3 Revision 4.0

Updated the document for Libero v11.6 software release (SAR 73206).

1.4 Revision 3.0

Updated the document for Libero v11.5 software release (SAR 64076).

1.5 Revision 2.0

Replaced the updated IGLOO2 design files (SAR 62650).

1.6 Revision 1.0

Initial release.

2 Preface

2.1 About this Document

This tutorial is for the SmartFusion®2 system-on-chip (SoC) Field Programmable Gate Array (FPGA) and IGLOO®2 FPGA devices. The tutorial demonstrates the use of SmartFusion2 and IGLOO2 serializer/deserializer (SERDES) in the Extended Physical Coding Sublayer (EPCS) mode. The EPCS mode and serializer/deserializer interface (SERDESIF) in the IGLOO2 device are common to the SmartFusion2 device. Therefore, the principles described in this tutorial can be applied to both the SmartFusion2 and IGLOO2 devices. It provides instructions on how to use the demo design. It also provides step-by-step instructions to create an EPCS based design using the Libero SoC software with the System Builder design flow.

2.2 Intended Audience

This tutorial is intended for:

- FPGA designers
- System-level designers

2.3 References

For more information about a complete and up-to-date listing of the IGLOO2 device documentation, refer *https://www.microsemi.com/product-directory/fpgas/1688-igloo2*. For more information about a complete and up-to-date listing of the SmartFusion2 device documentation, refer *https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2*

- *UG0451: SmartFusion2 and IGLOO2 Programming User Guide*
- *UG0447: SmartFusion2 and IGLOO2 FPGA High Speed Serial Interfaces User Guide*
- *SmartFusion2 and IGLOO2 High Speed Serial Interface Configuration Guide*

3 Implementing a SmartFusion2 and IGLOO2 SERDES EPCS Protocol Design

3.1 Introduction

The SmartFusion2 and IGLOO2 families of devices have embedded high-speed SERDES blocks. With EPCS protocols, these SERDES blocks can handle data rates from 1 Gbps to 2.5 Gbps with -STD speed-graded devices, and up to 3.2 Gbps with -1 speed-graded devices. The high-speed serial interface block, also known as SERDESIF, supports many serial communication standards. The SERDESIF module integrates several functional blocks to support multiple high-speed serial protocols within FPGAs.

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 the ten Gigabit attachment unit interface (XAUI) PCS logic in the SERDES block is bypassed. However, PCS logic can be implemented in the FPGA fabric, and the EPCS interface signals of the SERDES block can be connected. This allows any user-defined high-speed serial protocol to be implemented in the SmartFusion2 and IGLOO2 devices.

The CorePCS IP module supports programmable 8b/10b encoding and decoding. 8b/10b is commonly used in some protocols that are not included in the SERDESIF block by the Microsemi® SoC high-speed SERDES interface. Therefore, CorePCS is ideal to use with these protocols. It can be configured as a transmitter only, the 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, refer to the *CorePCS Handbook*.

This tutorial provides comprehensive step-by-step instruction of building an EPCS application design using the Libero® System-on-Chip (SoC) System Builder flow. It demonstrates the EPCS interface of SmartFusion2 and IGLOO2 FPGA devices and how this can be used for customized applications. It also provides a complete design flow starting from a new project to a working design on the IGLOO2 Evaluation Kit board.

After completing this tutorial, you will be able to perform the following tasks:

- Create a Libero SoC software project with a purposed EPCS interface
- Develop the Simulation Stimulus
- Simulate the design
- Generate the programming file
- Run the EPCS demo

3.2 Requirements

Table 1 shows the tutorial requirements for building SERDES EPCS Protocol Design.

Table 1 • **Tutorial Requirements**

Requirements	Version
Hardware	
Operating System	Windows 7, 8.1, or 10
SmartFusion2 and IGLOO2 FPGA Evaluation Kit	Rev C or later
FlashPro4 JTAG Programmer	1
SMA Male to SMA Male Loopback Cables	2
USB 2.0 A-male to mini-B for UART	1
12 V 2A wall-mounted power supply	1
Software	
Libero SoC	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Expres	
Host PC Drivers	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.

3.3 Demo Design

3.3.1 Introduction

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

- IGLOO2: *http://soc.microsemi.com/download/rsc/?f=m2gl_tu0570_df*
- SmartFusion2: *http://soc.microsemi.com/download/rsc/?f=m2s_tu0570_df*

The demo design files include:

- GUI_Installer
- Libero_Project
- Programming_Job
- Source_files
- Testbenches
- TCL_Script

Figure 1 shows the top-level structure of the IGLOO2 design files.

Figure 1 • **IGLOO2 Demo Design Files - Top-Level Structure**

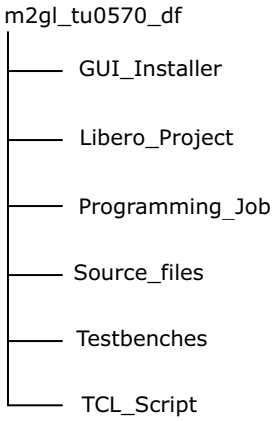
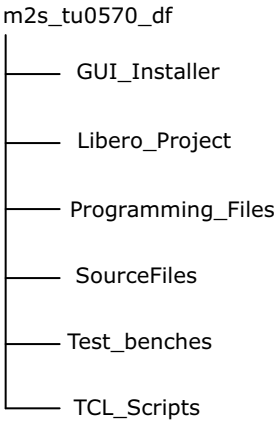


Figure 2 shows the top-level structure of the SmartFusion2 design files.

Figure 2 • **SmartFusion2 Demo Design Files - Top-Level Structure**



This example design demonstrates transmitting a Pseudo-Random Binary Sequence (PRBS) or counting pattern from the FPGA fabric to the SmartFusion2 and IGLOO2 high-speed SERDES interface. The SERDES block is configured for 2.5 Gbps operational speed. The PRBS pattern is sent over Lane 2, and a counting 8b/10b encoded pattern is used with Lane 1 of the SERDES block.

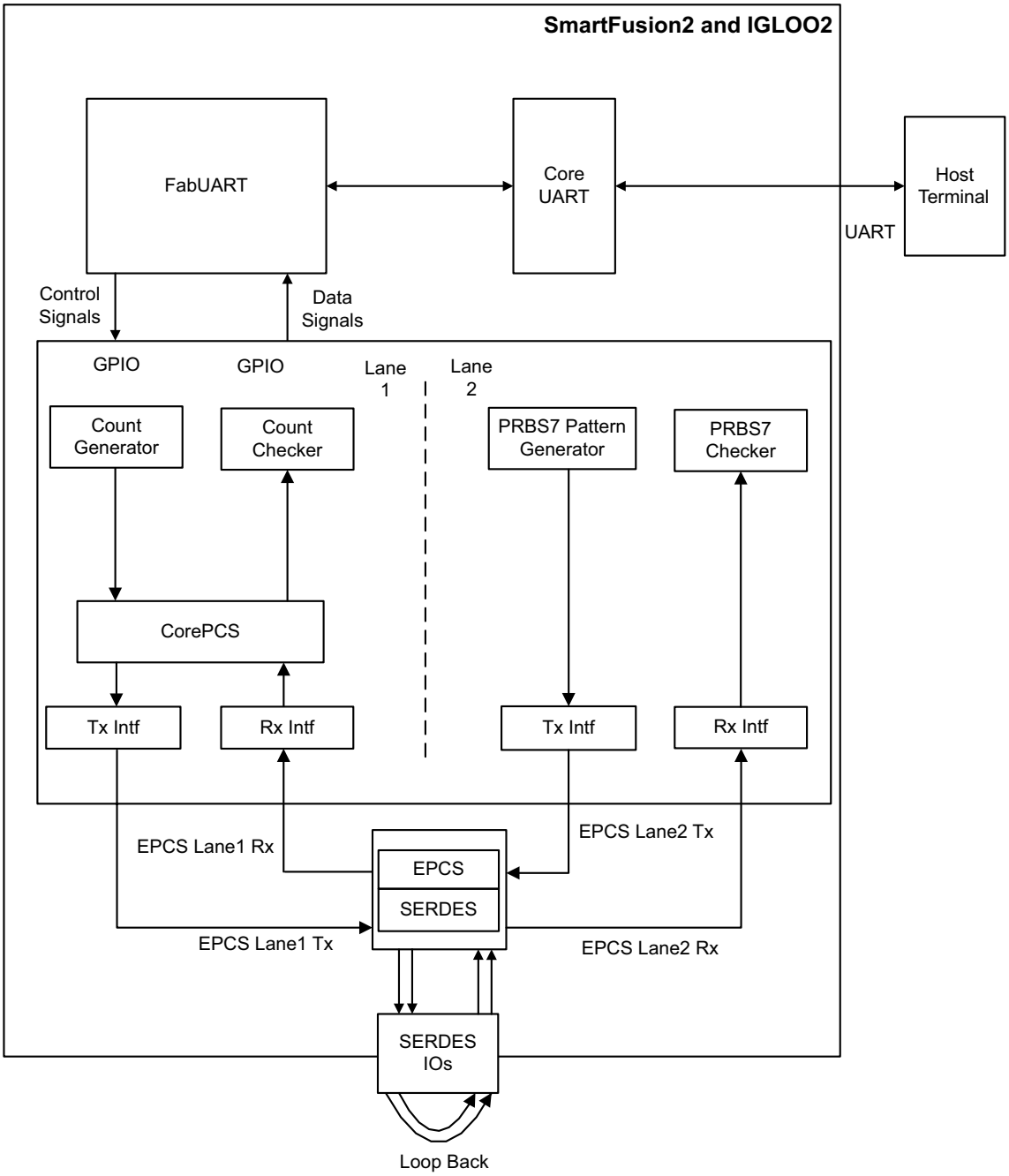
- Demo 1: Lane 2 traffic is sent directly from the fabric-based PRBS generator to the SERDES block and off-chip to test SMA connections. Input SMA connectors are routed to the SERDES receiver pins to bring the data back into the device to the fabric-based pattern checker.
- Demo 2: Lane 1 includes a pattern generator and checker that utilize the CorePCS IP module in the data path. The CorePCS module provides simple 8b/10b encoding and decoding functionality. This lane is routed off and on chip by looping the data on PCB trace connections.

Note: Lane 0 and Lane 3 are not used.

In both lane examples, the EPCS designing requires an understanding to meet the performance of the EPCS interface and FPGA fabric.

The system block diagram for the design implemented in the SmartFusion2 and IGLOO2 device is shown in Figure 3.

Figure 3 • **Demo Design Block Diagram**



Note:
For loopback
- Lane1 datapath is done on PCB with on-board traces.
- Lane2 datapath has SMA connectors and requires a cable to be connected to on-board SMA connectors.

3.3.2 Design Description

The hardware design for the implementation includes a PRBS, a count pattern generator, a PRBS sequence, a count pattern checker, an error counter, RX and TX fabric interface blocks, a delay line, a UART, an output select control, and a high-speed serial interface block connected to the SmartFusion2 and IGLOO2 SERDES block. Each block is explained in the following sections:

- PRBS7 Generator
- Count Generator
- PRBS7 Checker
- Count Checker
- Delay Line
- RX and TX Interface

3.3.2.1 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.

3.3.2.2 Count Generator

The count generator module implements a count pattern used to drive the CorePCS 8b/10b encoder. Packets created in the count generator are separated by a K28.5 character. The payload of the packet is a simple counting pattern.

3.3.2.3 PRBS7 Checker

The PRBS7 checker checks for valid PRBS sequences. If the received sequence does not match 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.

3.3.2.4 Count Checker

The count checker module checks for a valid count pattern received from the CorePCS 8b/10b decoder. The count checker checks each packet for the embedded count pattern used as the payload of the packet.

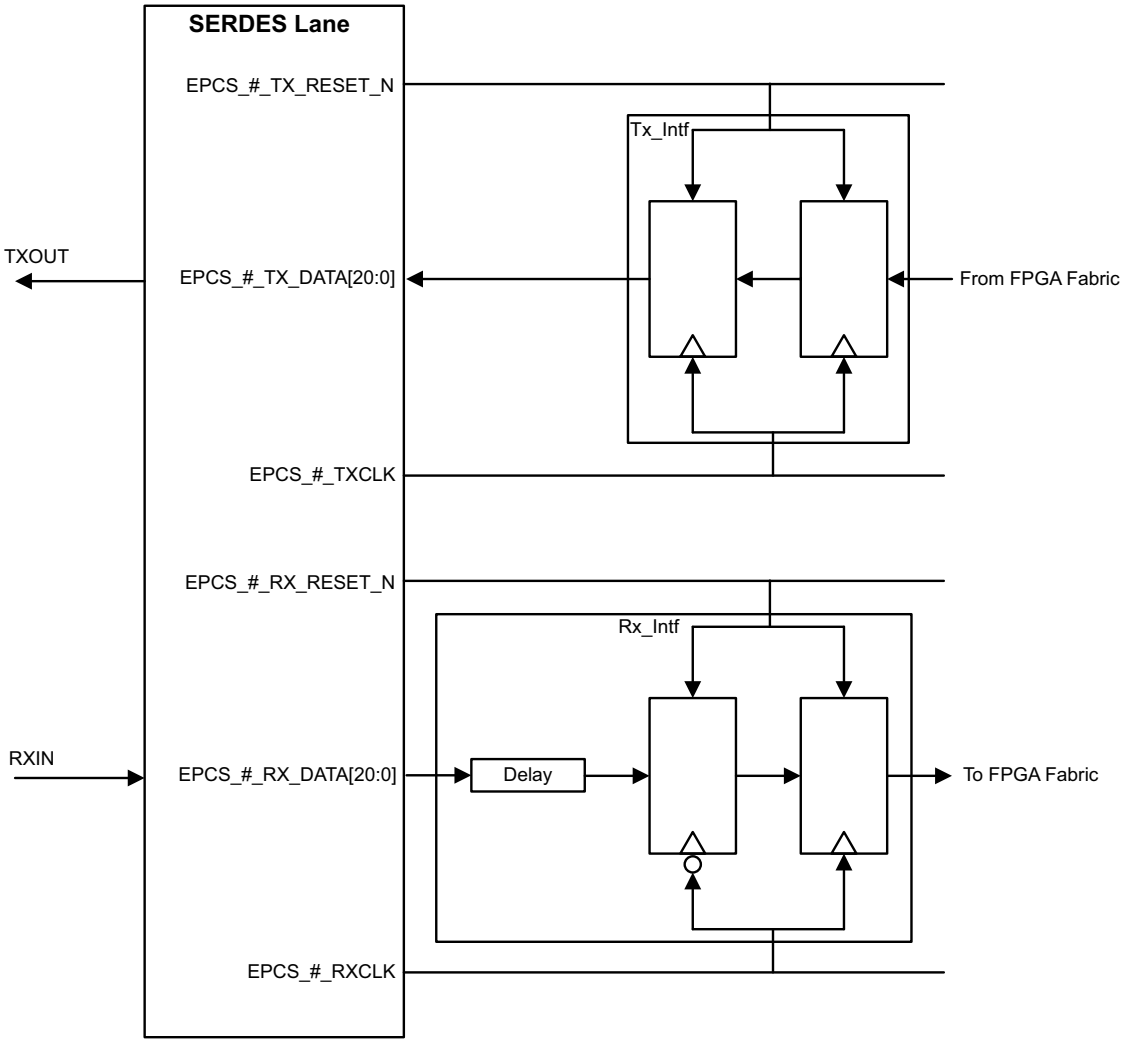
3.3.2.5 Delay Line

The delay line is used to balance the data delay to the first fabric register with the clock injection time of EPCS_RX_CLK to the fabric. This delay line uses a static delay value that can be used for any SERDES lane in SmartFusion2 and IGLOO2family devices.

3.3.2.6 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.

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



3.3.2.7 CoreUART, FabUART, and Output Select Modules

The COREUART module communicates with the UART interface on the IGLOO2 Evaluation Kit. FabUART and Output select modules are glue logic modules to connect the PRBS generator, 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 both Lane 1 and Lane 2. Depending on the lane selection, it feeds the corresponding status signals onto the UART.

3.3.2.8 SERDES

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 tutorial describes how to configure the SERDES block in the EPCS protocol. For more information about SERDES block, refer to the *UG0447: SmartFusion2 and IGLOO2 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.

3.3.2.9 Clocking

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

- Control Plane Clock: Used for HPMS or MSS, UART, and Output Select. The control plane clock is sourced by the SERDES REFCLK (using the REFCLK_OUT port of the SERDESIF) and passes through HPMS, which contains an embedded CCC (PLL) and divides the clock down to 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 remainder of the receive path.

3.4 Setting Up the Demo Design

3.4.1 Setting Up the Board

Use the following steps to set up the board:

1. Connect the FlashPro4 programmer to the programming header J5.
2. Connect the host PC or laptop to the J18 connector using the USB min-B cable.
3. Connect the 12 V 2 A-power jack to the board J6 power connector.
4. Install USB to UART drivers on the host PC. Ensure that the USB to UART bridge drivers are automatically detected.

Note: Download and install the drivers from: *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.*

5. Connect the jumpers on the board, as listed in Table 2. For more information about jumper locations, refer Appendix 6: Jumper Locations, page 51.

Note: Before making the jumper connections, switch OFF the power supply.

Table 2 • **Jumper Settings**

Jumper Number	Settings	Notes
J22	1-2 closed	Lineside output is enabled.
J23	2-3 closed	External clock is required to source SMA connectors to the lineside.
J3	1-2 closed	Manual power switching using the SW7 switch.
J8	1-2 closed	FlashPro4 for SoftConsole/FlashPro.

6. Connect the power supply to the J18 DC jack.

The design uses SERDES Lane 1 and Lane 2. Lane 2 must be looped back with SMA cables.

Figure 5 shows the IGLOO2 Evaluation Kit board with cable connections.

Figure 5 • **IGLOO2 Evaluation Kit Board**

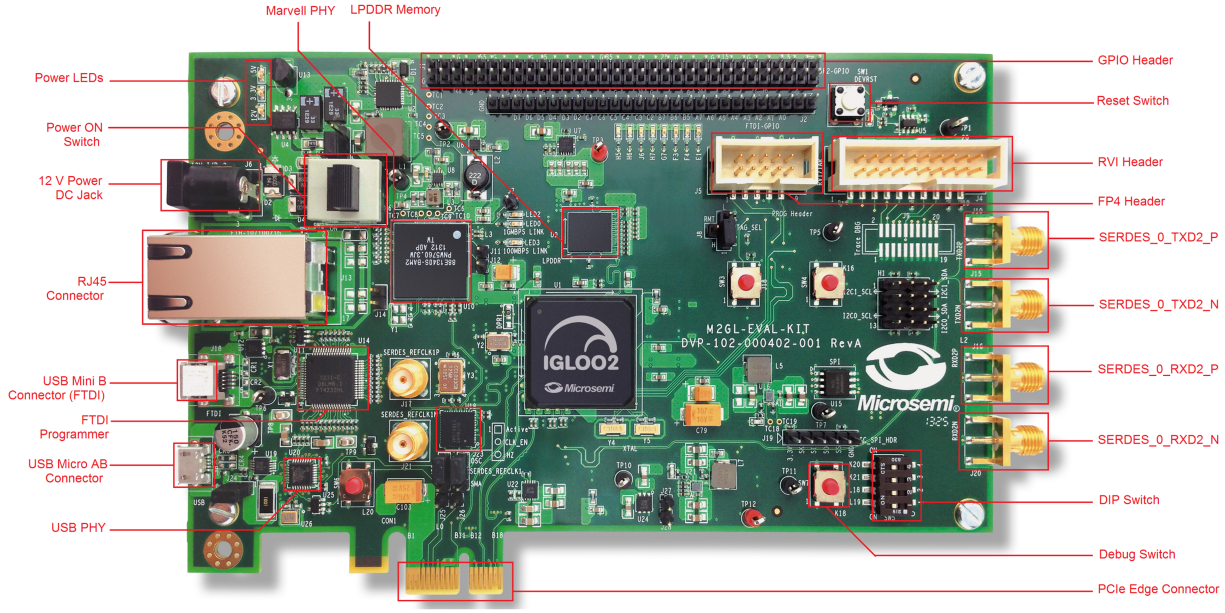
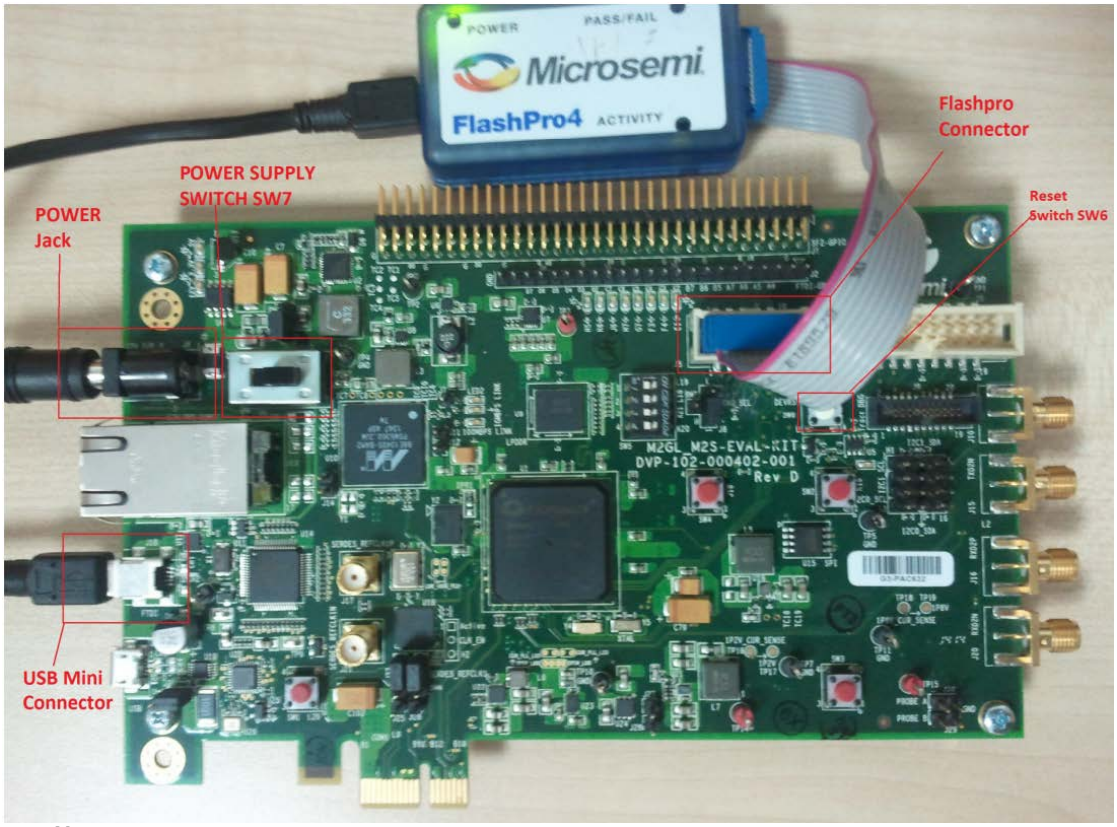


Figure 6 • **SmartFusion2 Security Evaluation Kit**



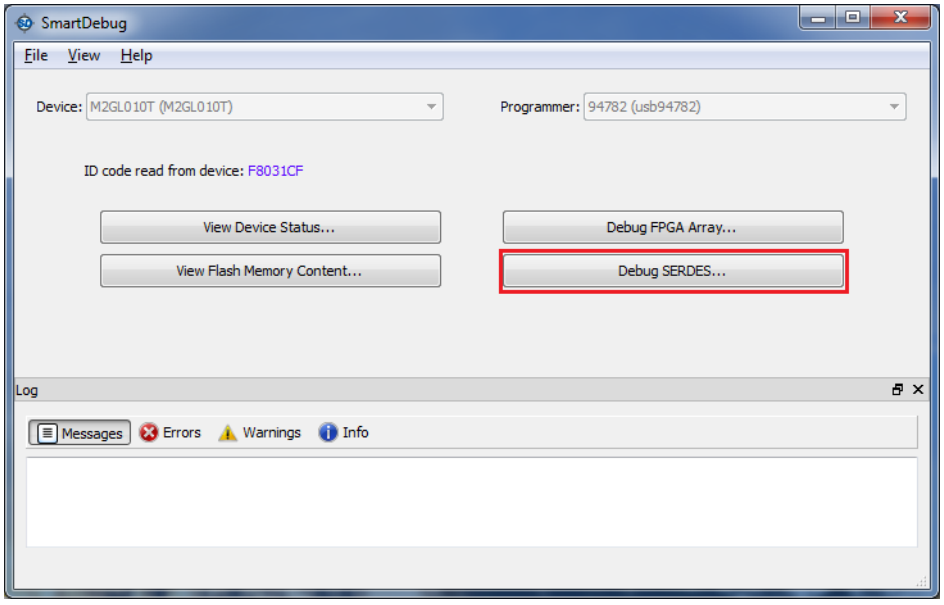
Note:

- SmartFusion2 and IGLOO2 Evaluation Kit board use common PCB design. Figure 5 shows SmartFusion2 and IGLOO2 Evaluation Kit board.
- SERDES Lane 1 is looped back from transmit to receive data on the board. Therefore, it is not required to connect external SMA Loopback cables on Lane 1.

3.4.2 Using SmartDebug with SERDES Design

1. Open **Libero Project**, go to **Design Flow > Debug Design** and double-click **SmartDebug Design**. The **SmartDebug** window is displayed, as shown in Figure 7.

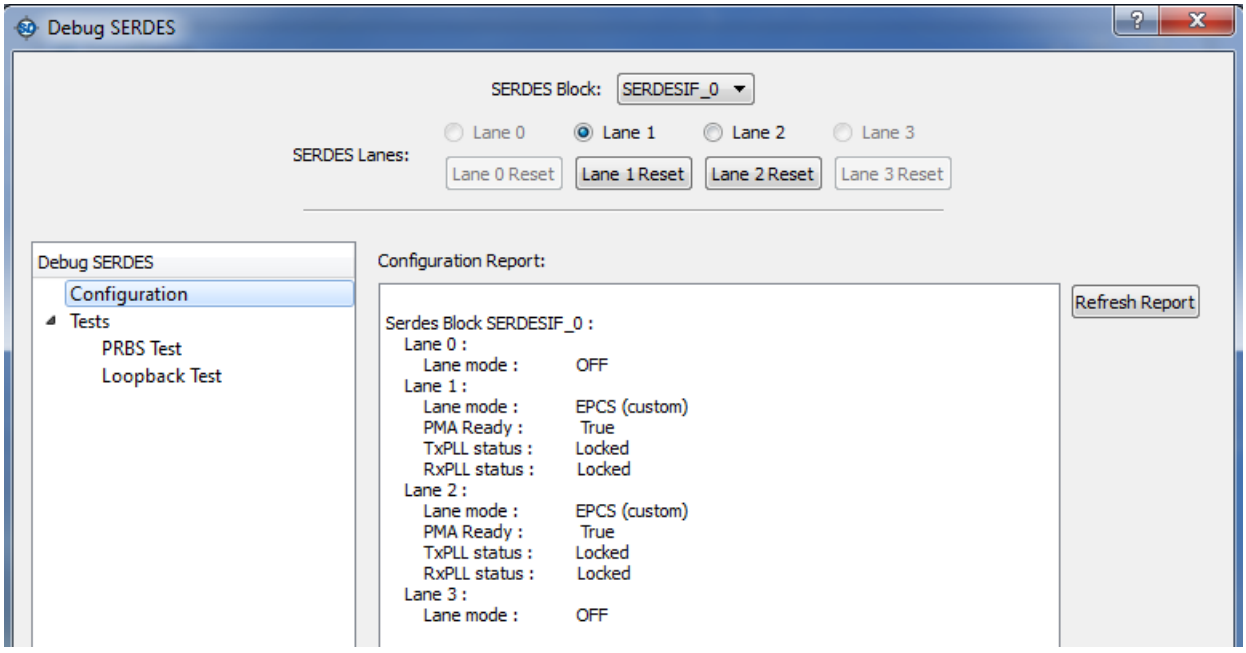
Figure 7 • **SmartDebug Window**



2. Click **Debug SERDES**.

The **Debug SERDES** window is displayed, as shown in Figure 8.

Figure 8 • **SmartDebug Window - Debug SERDES**



3. Select **Configuration** under **Debug SERDES**.
4. The **Debug SERDES** window displays the health information of the SERDES. It displays the Lanes that are in the design as well as the PLL status.

Note: For more information about the SmartDebug SERDES Debugger, refer to the *TU0530: SmartFusion2 and IGLOO2 SmartDebug – Hardware Design Debug Tools Tutorial*.

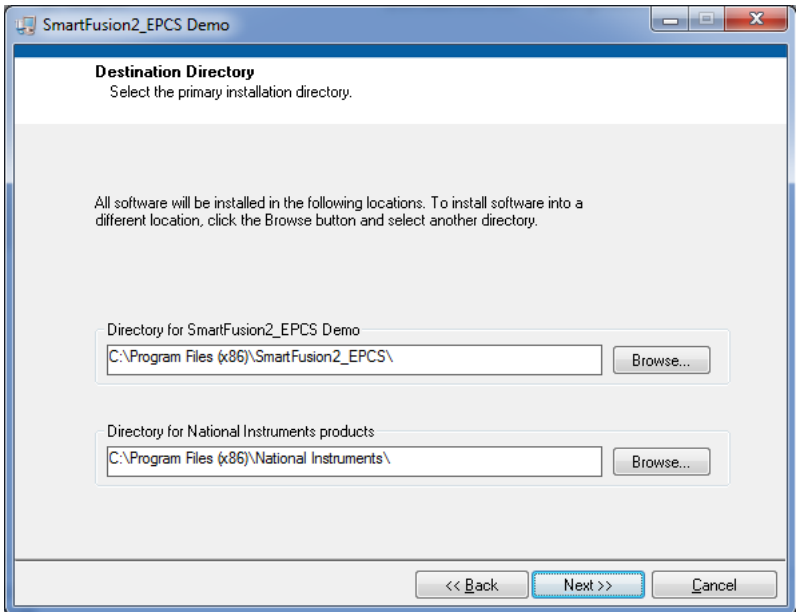
3.4.3 Installing the GUI

1. Run the installer, if the GUI is used for the first time.
2. Download the design files from:
   - IGLOO2: *http://soc.microsemi.com/download/rsc/?f=m2gl_tu0570_df*
   - SmartFusion2: *http://soc.microsemi.com/download/rsc/?f=m2s_tu0570_df*
3. Open **GUI_Installer > Volume > setup.exe**.
4. Click **Yes** for any message from **User Account Control**.

The Setup window appears and default locations are displayed, as shown in Figure 9.

5. Click **Next.**

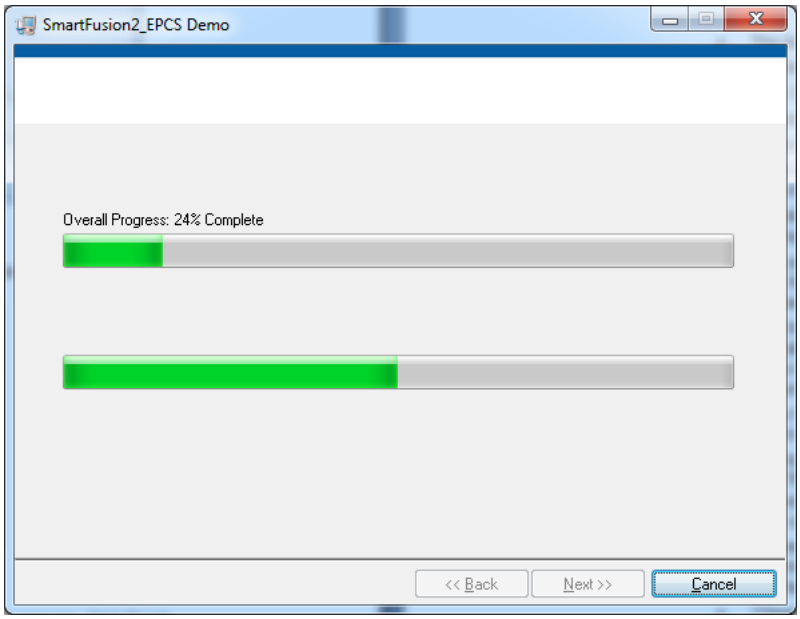
Figure 9 • **GUI Setup Window**



6. Follow the steps to begin the installation.

A progress bar appears, which shows the progress of installation, as shown in Figure 10.

Figure 10 • **GUI Setup Progress Bar**



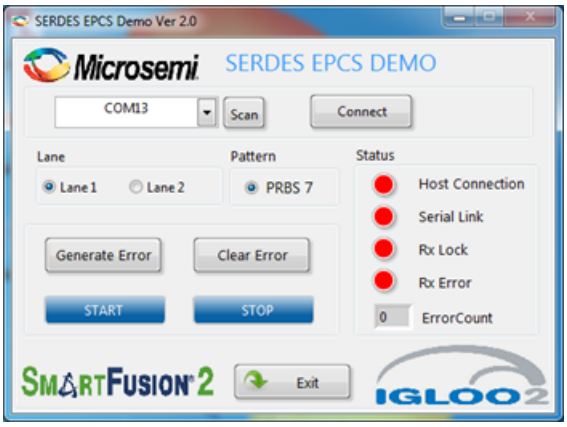
7. Wait for the installation to complete. It may take a few minutes.
8. After successful installation, **Installation Complete** message is displayed.
9. Restart the computer before using the installed GUI.

3.5 Running the Demo Design

1. Open **Programs > SERDES EPCS** Demo.

The SERDES EPCS Demo window is displayed, as shown in Figure 11.

Figure 11 • **EPCS Demo GUI Window**



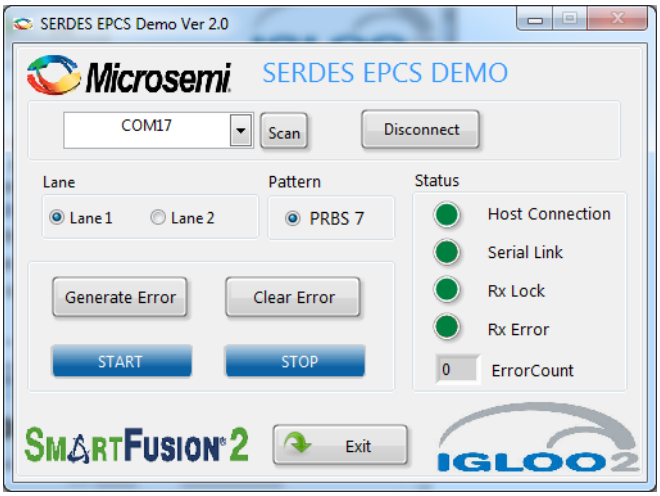
The drop-down list for the ports gives the list of serial ports available on the host PC. Only the working ports are enabled. The ports that are unavailable are grayed out.

Note: Default settings for the design are 9600 Baud, no flow control, one stop, and no parity.

2. Click **Connect** to connect the host PC to the hardware through the selected port.
3. Click **Start** to start the EPCS demo. The PRBS7 data starts getting generated and 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. For more information about the status signals, refer Appendix 5: Status Signals, page 50.
4. Click **Stop** to stop the EPCS demo.
5. Click **Exit** to exit.

Figure 12 shows a sample SERDES EPCS Demo window during an error free operation.

Figure 12 • **Sample SERDES EPCS Demo Window**



3.6 Libero Design Flow

This tutorial is described using the IGLOO2 device on the IGLOO2 Evaluation Kit. Follow the same procedure for the SmartFusion2 device.

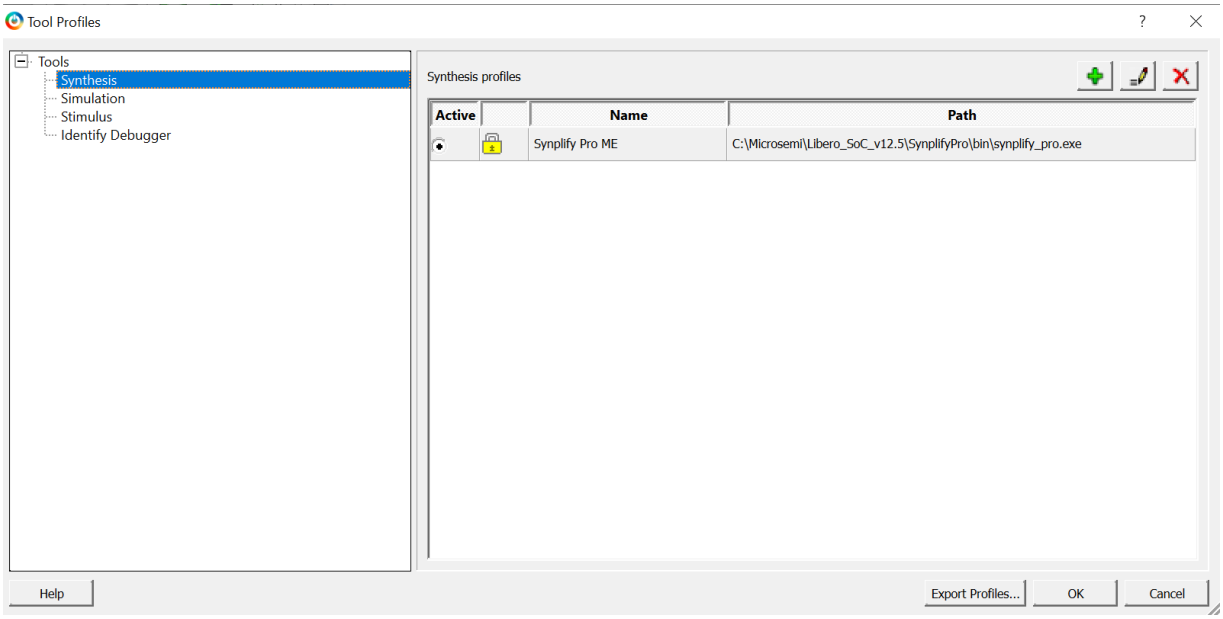
3.6.1 Step 1: Creating a Libero SoC Project

This section describes how to create an IGLOO2 EPCS demo design using the Libero software. It also highlights differences in the design flow when using the SmartFusion2 device.

3.6.1.1 Launching Libero SoC

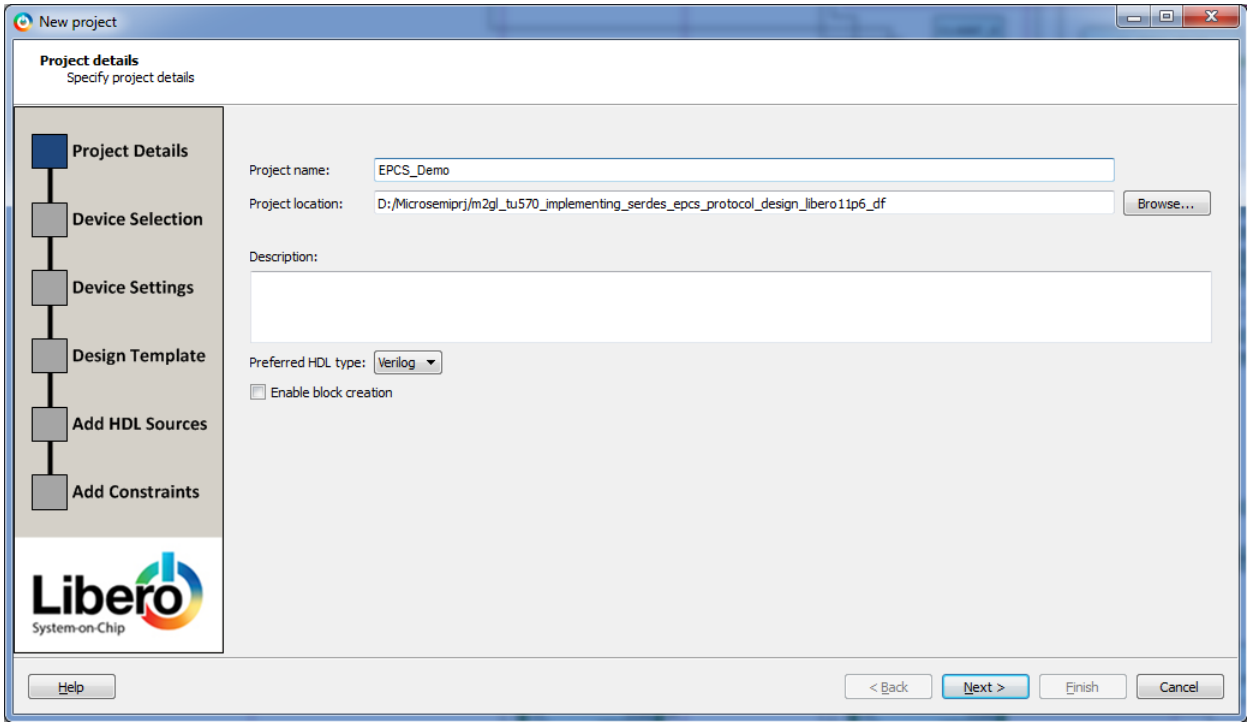
1. Click **Start > Programs > Microsemi Libero SoC > Libero SoC** , or double-click shortcut on the desktop to open the Libero SoC Project Manager.
2. Click **Edit Tool Profiles** to confirm the tool settings. Figure 13 shows the **Tool Profiles** window.
3. Ensure that the following tool settings are selected. For example:
    - Synthesis: synplify_Q202003M
    - Simulation: ModelsimPro 10.5c
    - Programming: FlashPro Express v12.5

Figure 13 • **Tool Profiles**



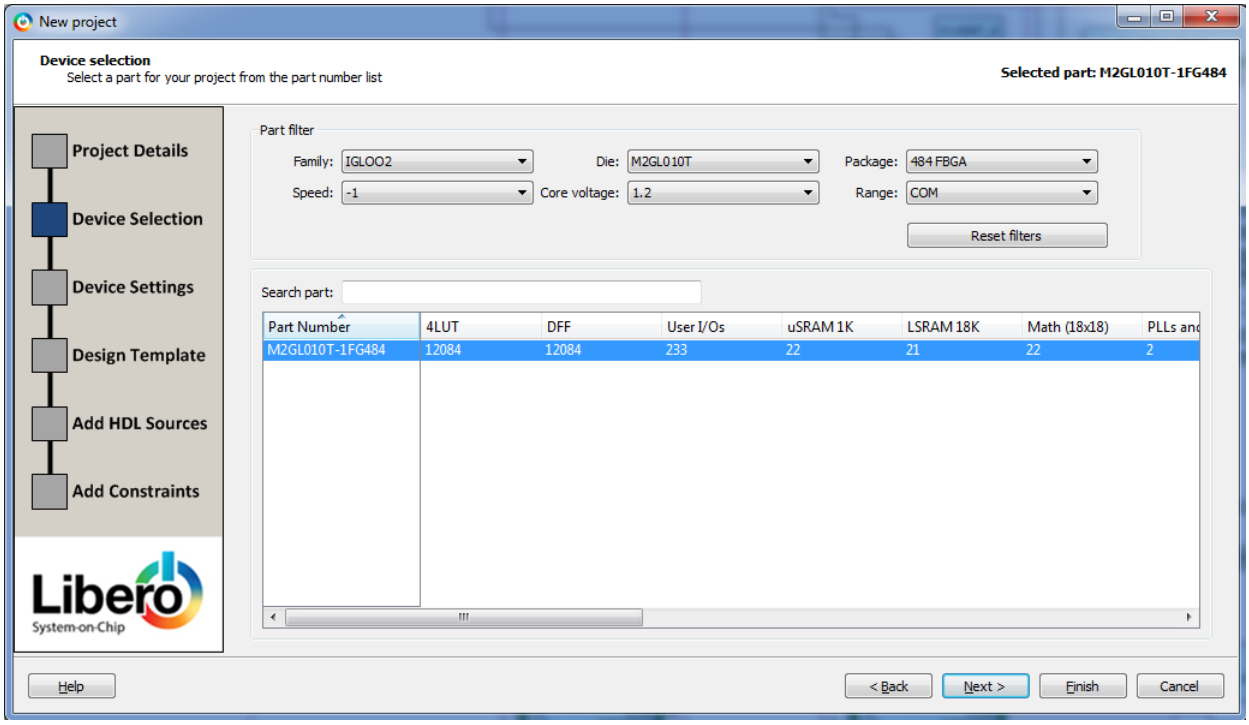
4. On the **Tool Profiles** window, click **OK**.
5. Select **New** on the **Start Page** tab to create a new project, or select **Project > New Project** from the Libero SoC menu. Figure 14 shows the IGLOO2 New Project window.
6. Enter the following details. For example:
    - Project name: EPCS_Demo
    - Project location: Select an appropriate location (for example, *D:/microsemi_prj*)
    - Preferred HDL type: Verilog

Figure 14 • **New Project Window - Project Details**



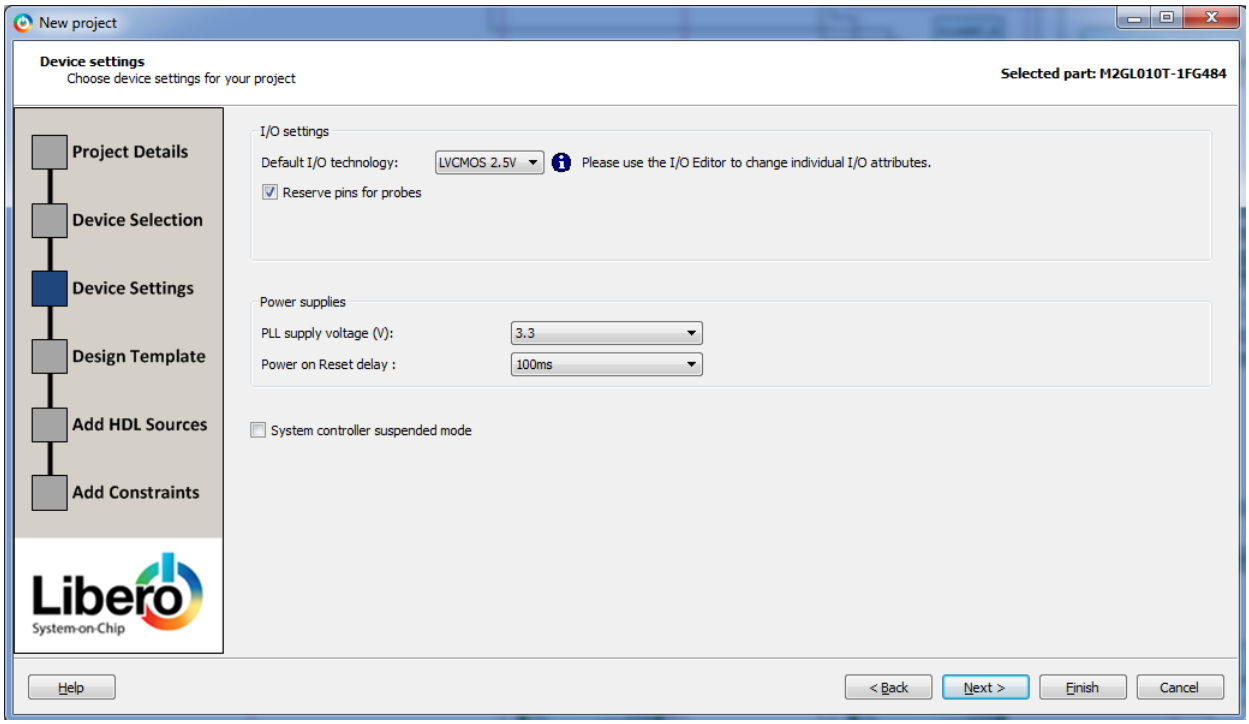
7. In the **Device Selection** window, enter the following details, as shown in Figure 15:
   - Family: SmartFusion2 and IGLOO2
   - Die: M2GL010T or M2S090T
   - Package: 484FBGA
   - Speed: -1
   - Core Voltage (V): 1.2
   - Range: COM

Figure 15 • **New Project Window - Device Selection**



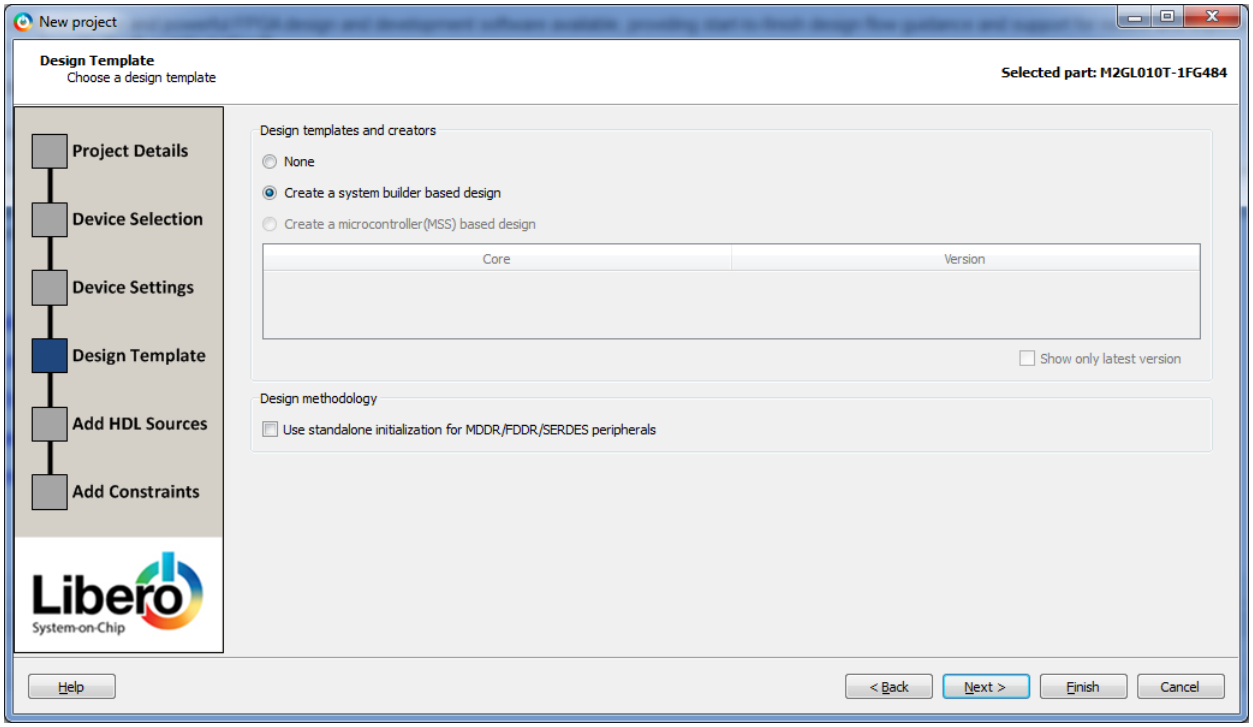
8. In the **Device Settings** window, enter the following details, as shown in Figure 16:
   - Default I/O technology: LVCMOS 2.5V
   - PLL supply voltage (V): 3.3 V
   - Power on Reset delay: 100 ms

Figure 16 • **New Project Window - Device Settings**



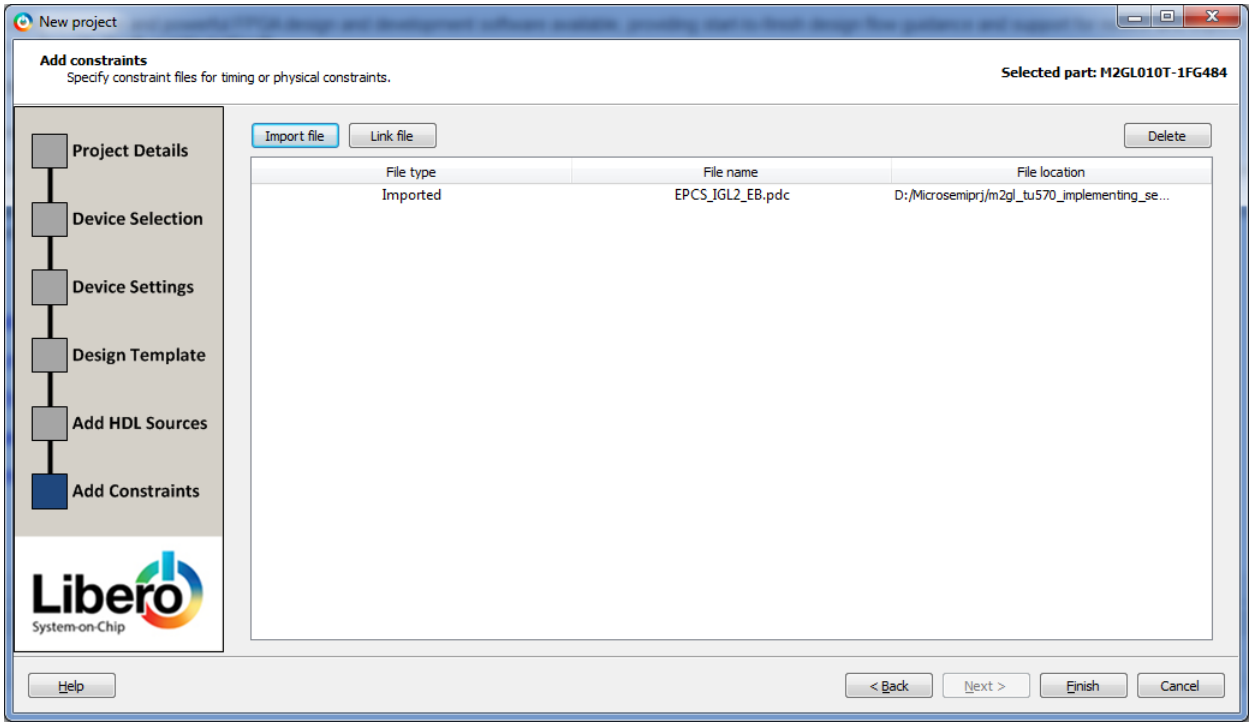
9. From the Design Template window, select **Create a system builder based design** and click **Next**.

Figure 17 • **Design Template window**



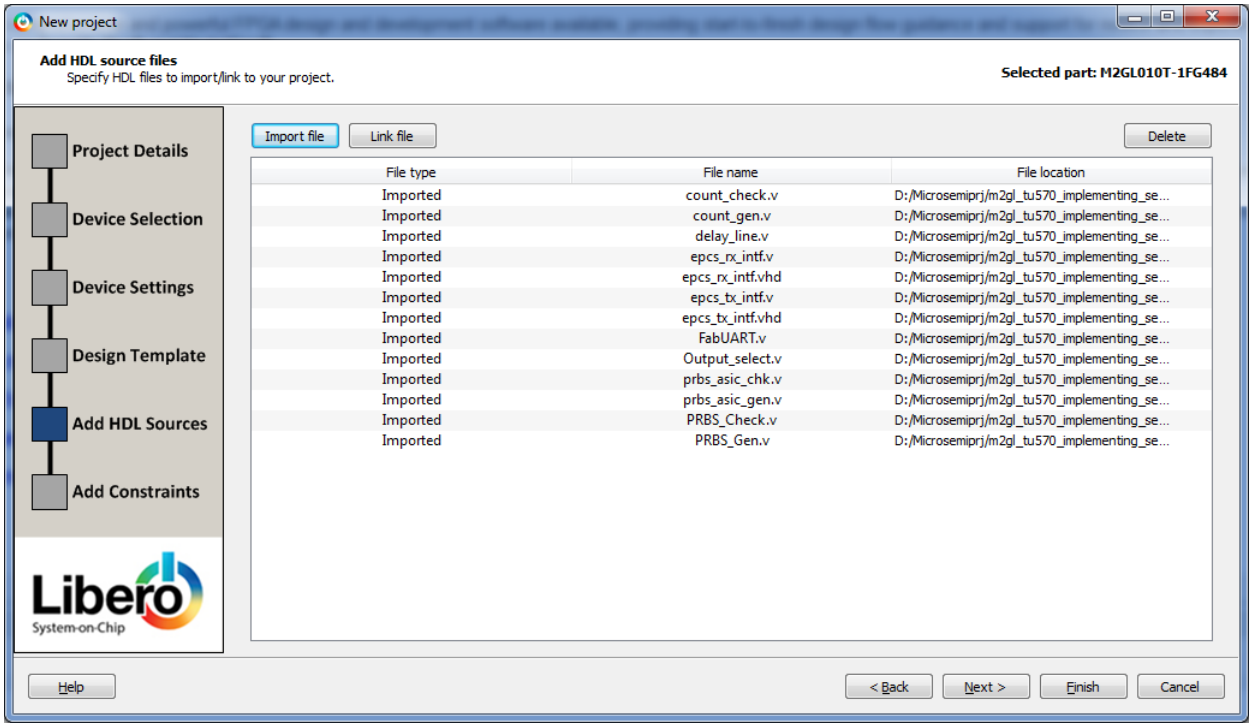
10. Click **Add HDL Sources** to import the RTL source files. HDL source files are included with the tutorial design files. Select all the files in the SourceFiles folder.

Figure 18 • **Add Constraints window**



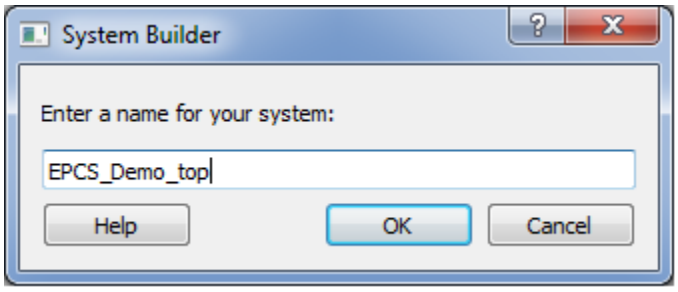
11. Click **Next**.
12. Click **Add Constraints**. Select the I/O constraint file located in the SourceFiles folder. This file is used for locking the pins for the demo using the SmartFusion2 and IGLOO2 Evaluation board.

Figure 19 • **Add HDL window**



13. Click **Finish**.
14. On the **New Project** window, click **OK**. The **System Builder** dialog box is displayed, as shown in Figure 20.

Figure 20 • **System Builder Name**



15. Enter **EPCS_Demo_top** as the name of the system and click **OK**. The **System Builder** window is displayed with the **Device Features** page.
16. In the **System Builder - Device Features** page, select the **SERDESIF_0** check box under **High Speed Serial Interfaces**, as shown in Figure 21 for IGLOO2 and Figure 22 for SmartFusion2

Figure 21 • **System Builder - IGLOO2 Device Features**

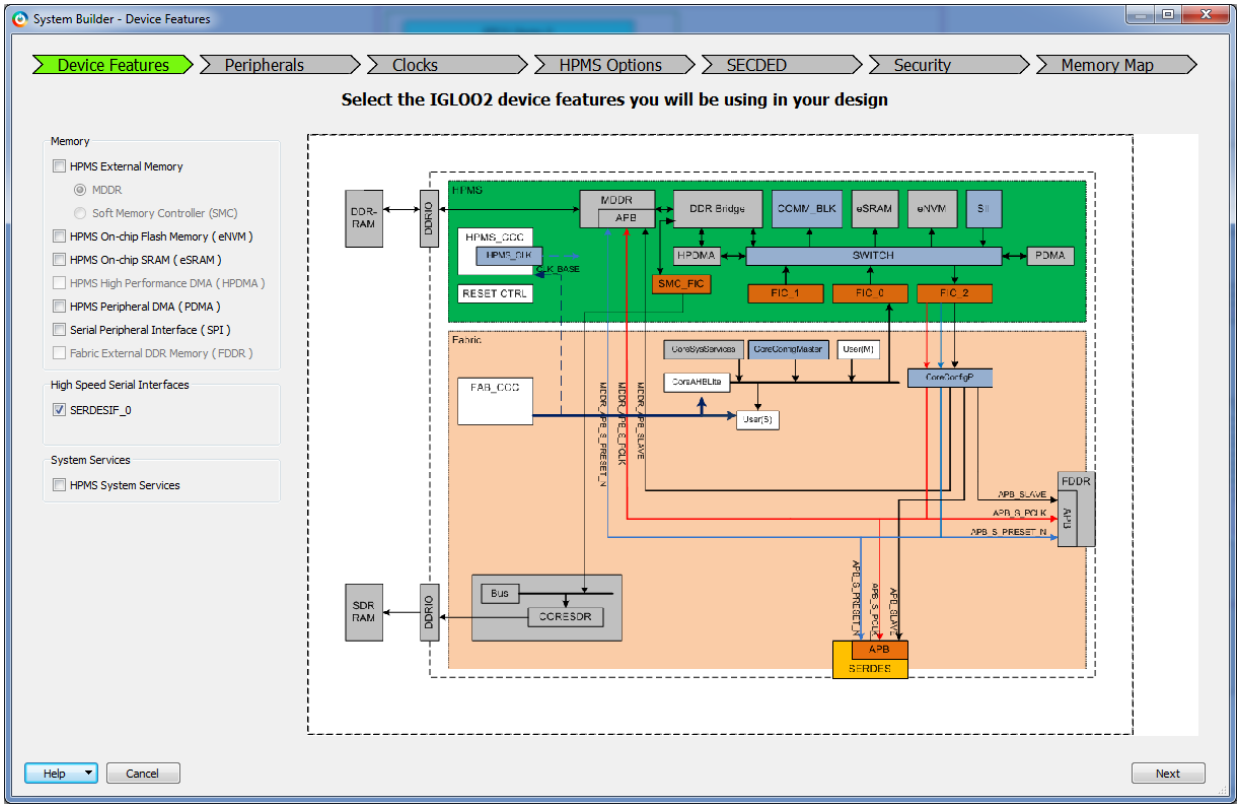
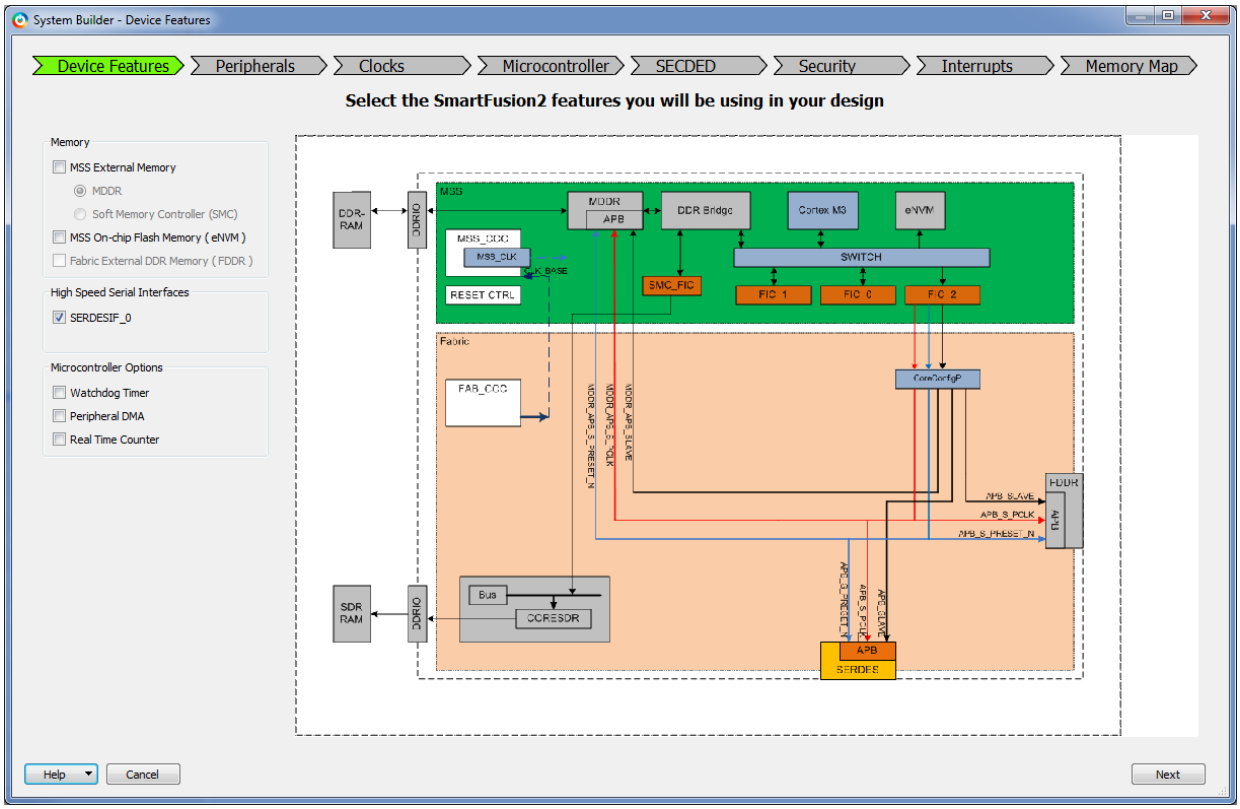
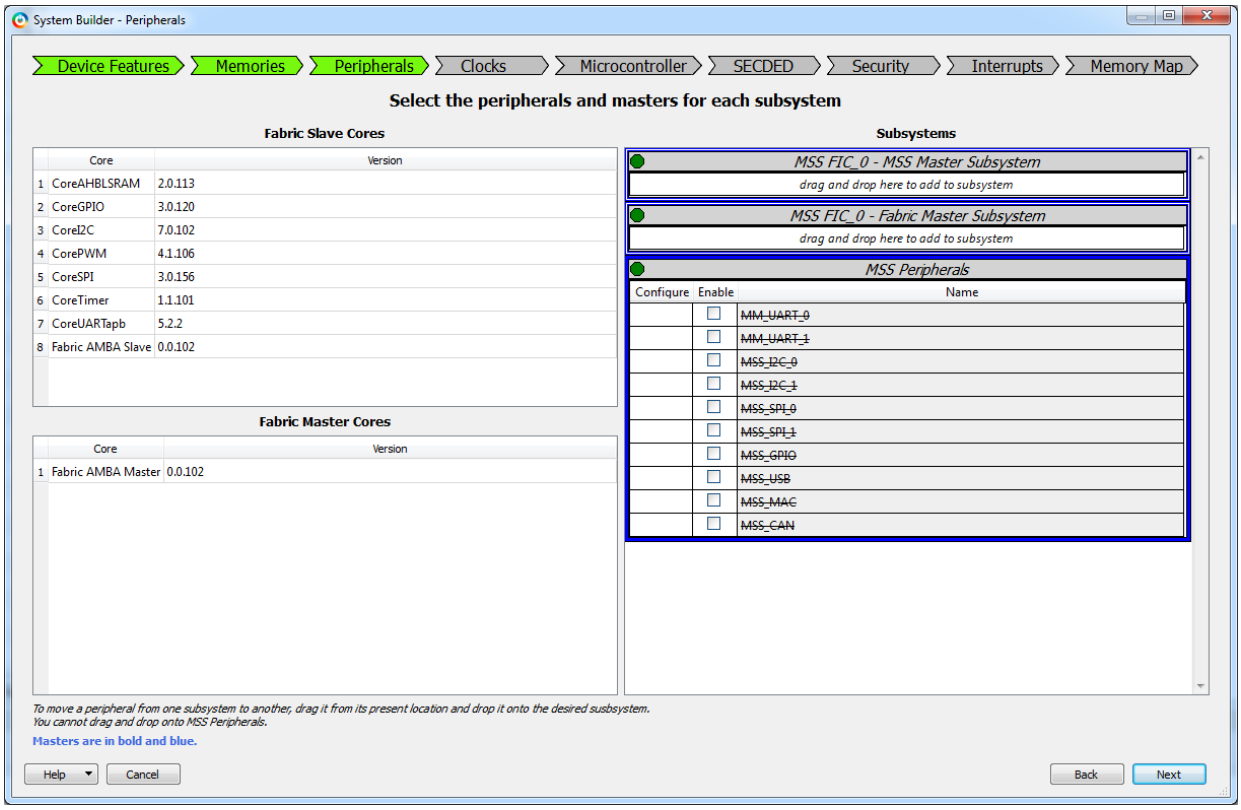


Figure 22 • **System Builder- SmartFusion2 Device Features**



17. Click **Next**. The **System Builder - Peripherals** page is displayed. Keep all the default selections for IGLOO2. For SmartFusion2, uncheck the default MSS peripherals and use the settings shown in Figure 23.

Figure 23 • **SmartFusion2 System Builder Peripherals**



18. Click **Next**. The **System Builder - Clock** page is displayed, as shown in Figure 24.

Figure 24 • **System Builder- IGLOO2 Clocks**

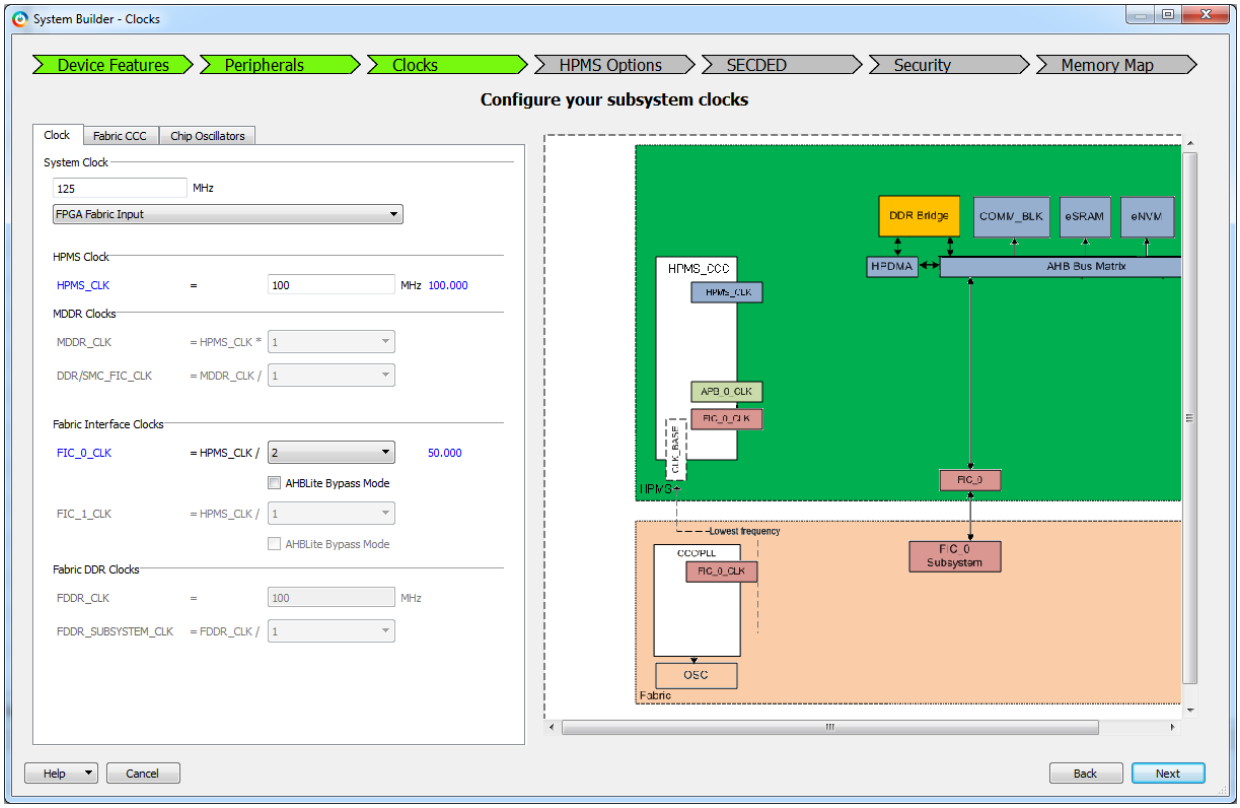
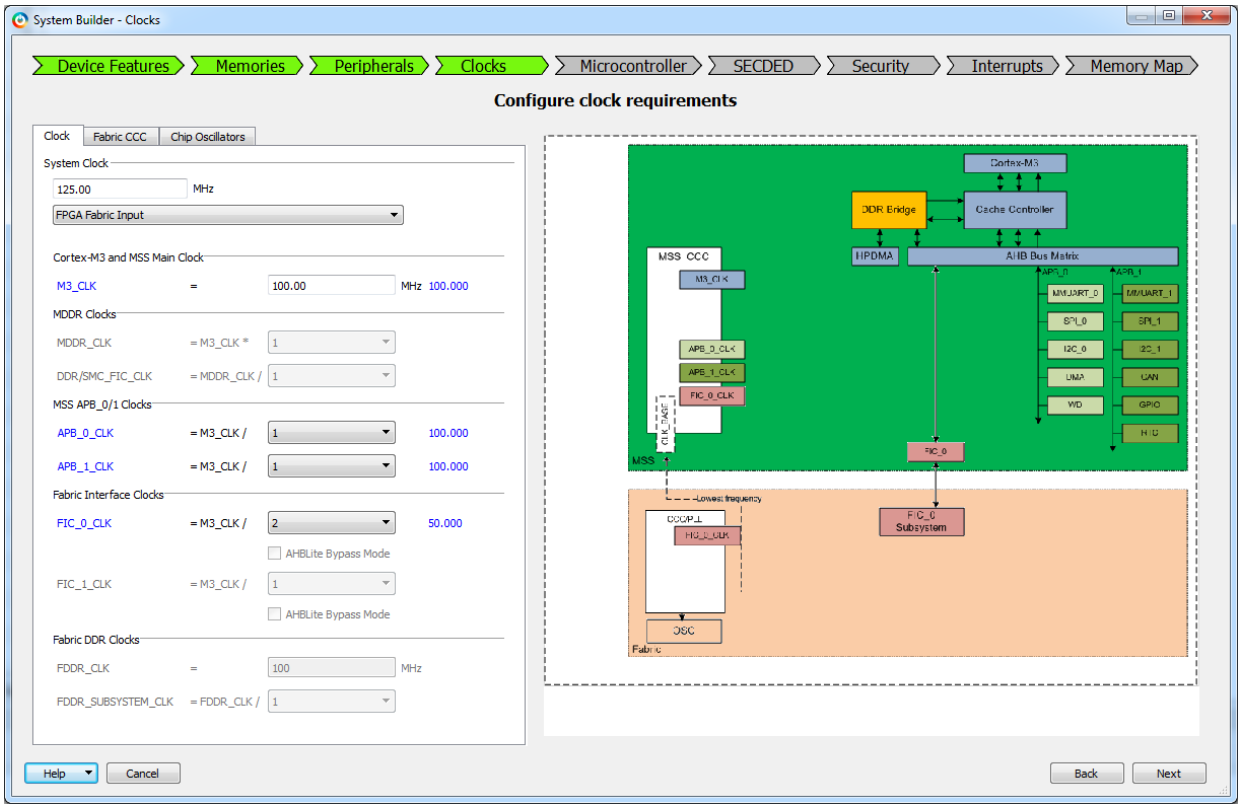


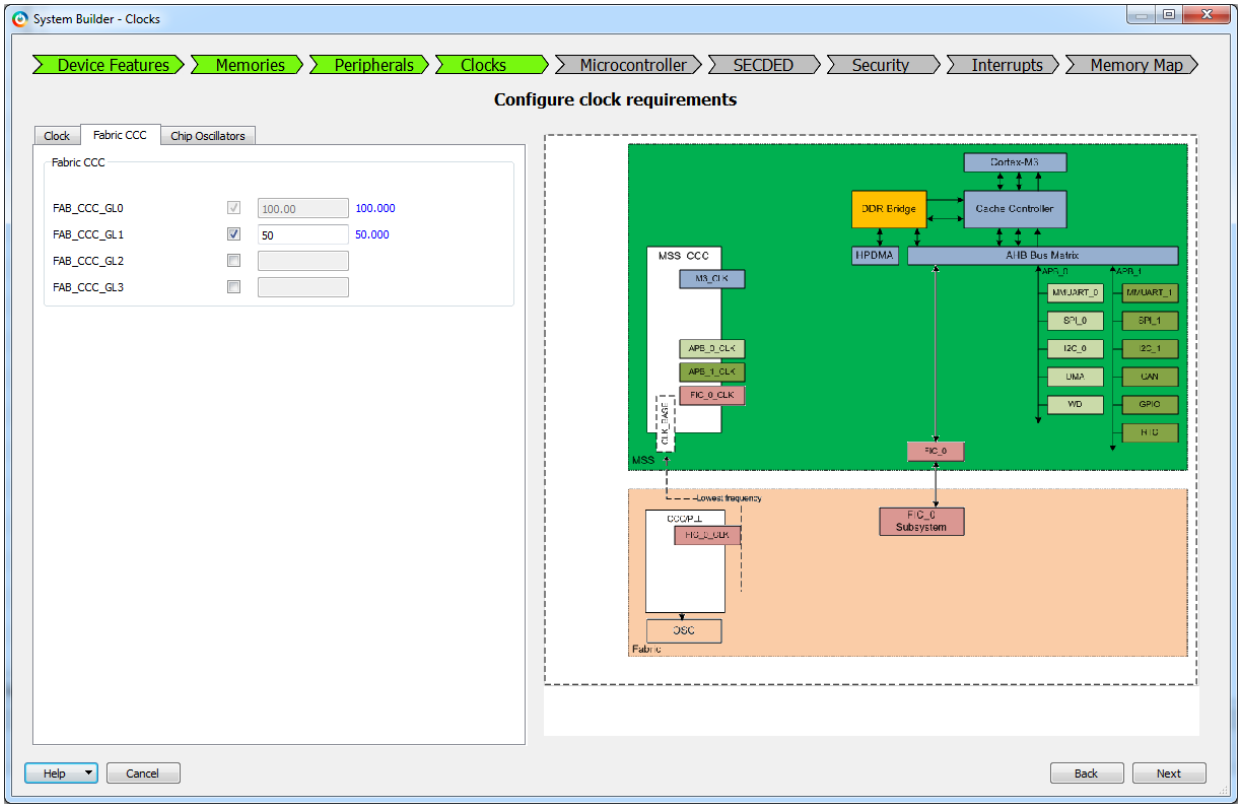
Figure 25 shows the SmartFusion2 clocks.

Figure 25 • **System Builder - SmartFusion2 Clocks**



19. Enter the following settings for SmartFusion2 and IGLOO2:
    - For IGLOO2, select **System Clock** source as **125.00** MHz, **FPGA Fabric Input** from the drop-down list, and **HPMS_CLK** as **100.00** MHz.
    - For SmartFusion2, select **System Clock** source as **125.00** MHz, **FPGA Fabric Input** from the drop-down list, and **M3_CLK** as **100.00** MHz. Select **Fabric CCC** tab, select the **FAB_CCC_GL1** check box and set to **50.00** MHz, as shown in Figure 26.

Figure 26 • **SmartFusion2 - Fabric CCC Tab**



20. Click **Next**. The **System Builder - Microcontroller Options** page is displayed. Keep the default selections.
21. Click **Next**. The **System Builder - SECDED** page is displayed. Keep the default selections.
22. Click **Next**. The **System Builder - Security** page is displayed. Keep the default selections.
23. Click **Next**. The **System Builder - Interrupts** page is displayed. Keep the default selections.
24. Click **Next**. The **System Builder - Memory Map** page is displayed. Keep the default selections.
25. Click **Finish**.

The **System Builder** generates a system based on the selected options. The System Builder block is created and added to the Libero SoC project automatically, as shown in Figure 27 for IGLOO2 and Figure 28 for SmartFusion2.

Figure 27 • **System Builder - IGLOO2 Component**

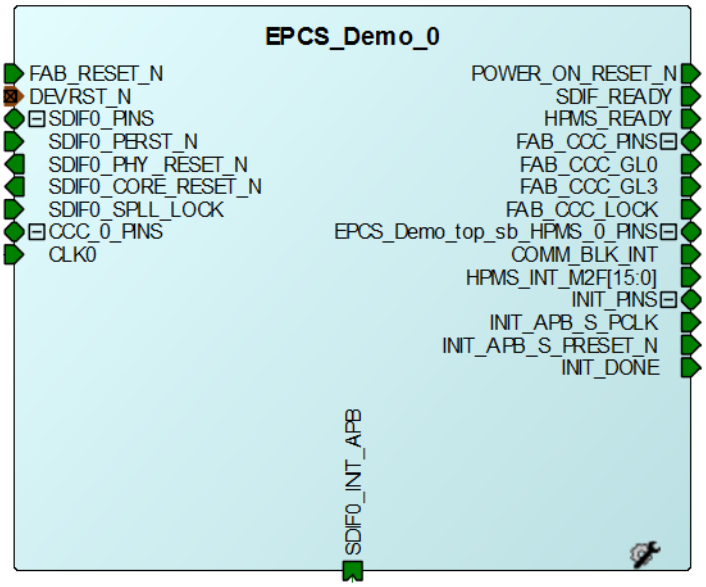
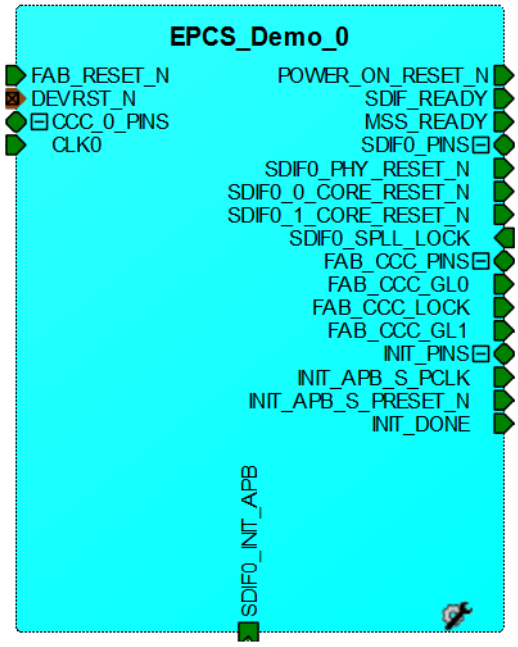


Figure 28 • **System Builder - SmartFusion2 Component**



Note: The symbol might be different when the buses are collapsed, such as the buses labeled **SDIF0_PINS, INT_PINS**, and **FAB_CCC_PINS**. Click on the + or - symbol to collapse or expand the buses.

If it is designed for the IGLOO2 family, the two soft cores, CoreResetP, and CoreConfigP are automatically instantiated and connected by the System Builder.

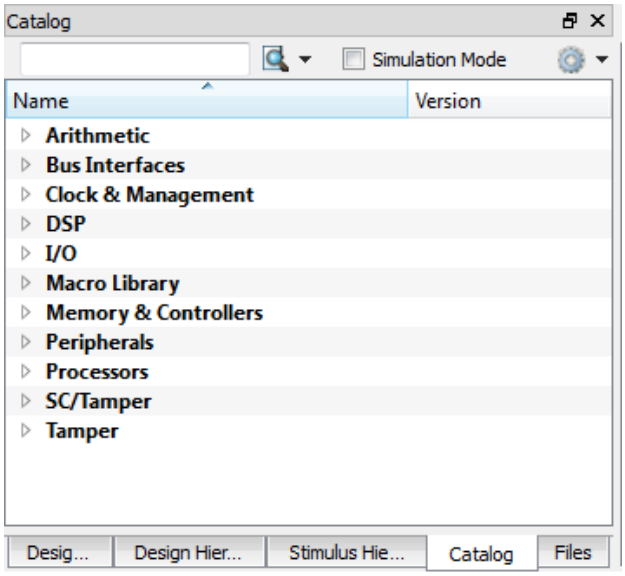
Note: CoreResetP and CoreConfigP reset and configure the peripherals. In this case, they are used to reset and configure the SERDESIF module. These modules are included in the System Builder generated component.

3.6.2 Step 2: Instantiating Libero SoC Catalog Components in SmartDesign

The Libero SoC catalog provides IP cores that can be dragged onto the SmartDesign canvas.

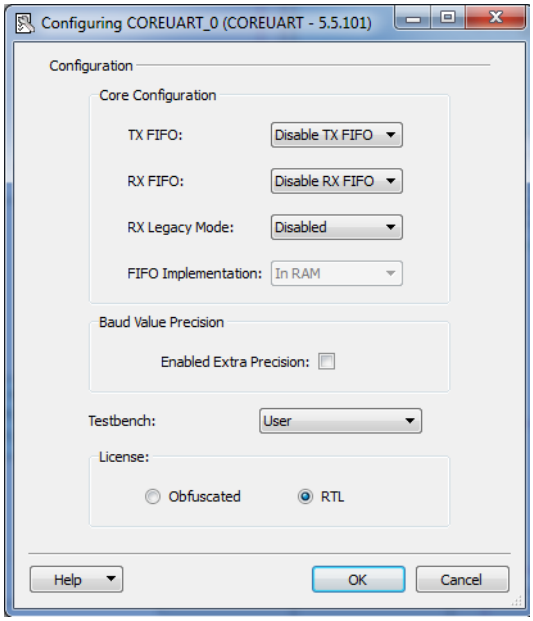
1. In the current EPCS_Demo_top canvas, go to the **Catalog** tab. Expand the **Peripherals** category, select **COREUART**, and drag it onto the `ECPS_Demo_top` canvas.

Figure 29 • **Catalog View**



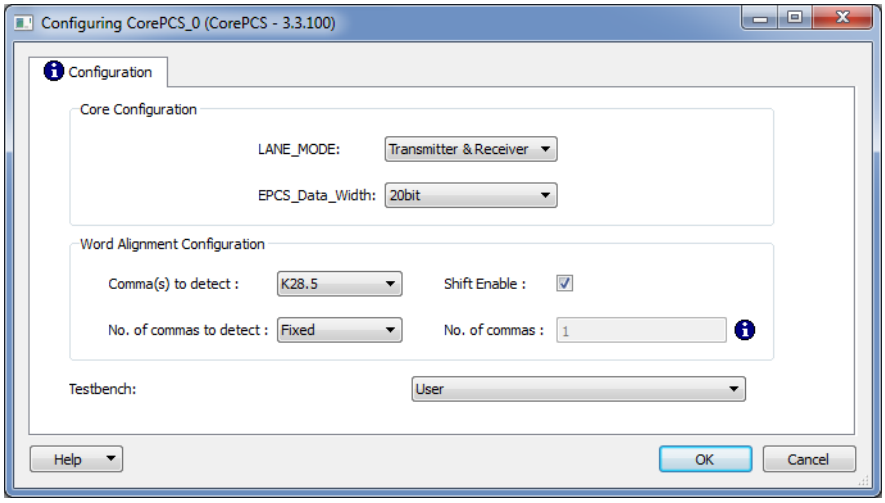
2. After dragging the **COREUART** onto the canvas, double-click **COREUART_0** module and set the parameters, as shown in Figure 30.

Figure 30 • **COREUART Settings**



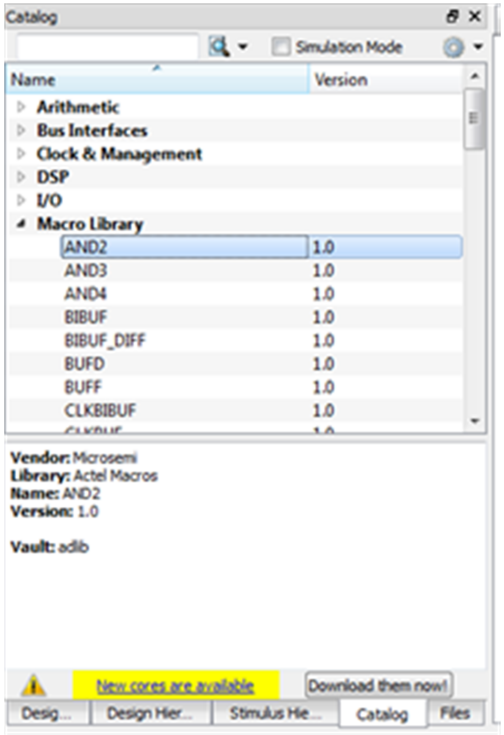
3. Click **OK**.
4. In the **Peripherals** tab, select the **CorePCS** module and drag it onto the `EPCS_Demo_top` SmartDesign canvas. Double-click **CorePCS_0** module and set the parameters, as shown in Figure 31.

Figure 31 • **CorePCS Configurator Settings**



5. Click **OK**.
6. In the **Macro Library** tab, select the **AND2** module, as shown in Figure 32. Drag two instances of the component onto the `EPCS_Demo_top` canvas.

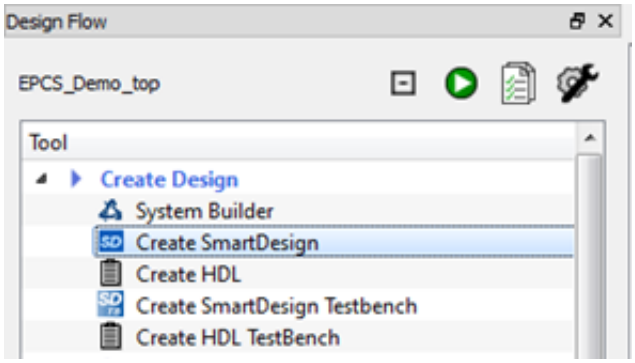
Figure 32 • **Macro Library Component**



3.6.3 Step 3: Creating SmartDesign Hierarchy and Adding a SERDESIF Component

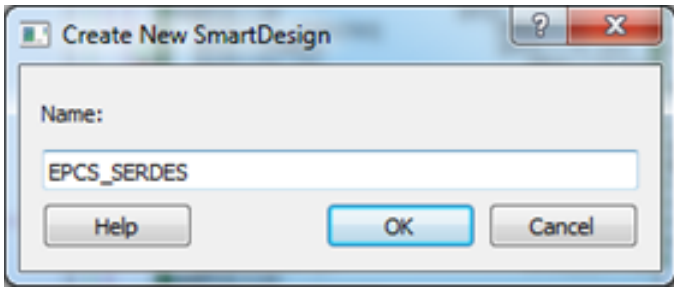
1. On the **Design Flow** tab, click **Create SmartDesign**.

Figure 33 • **Creating SmartDesign**



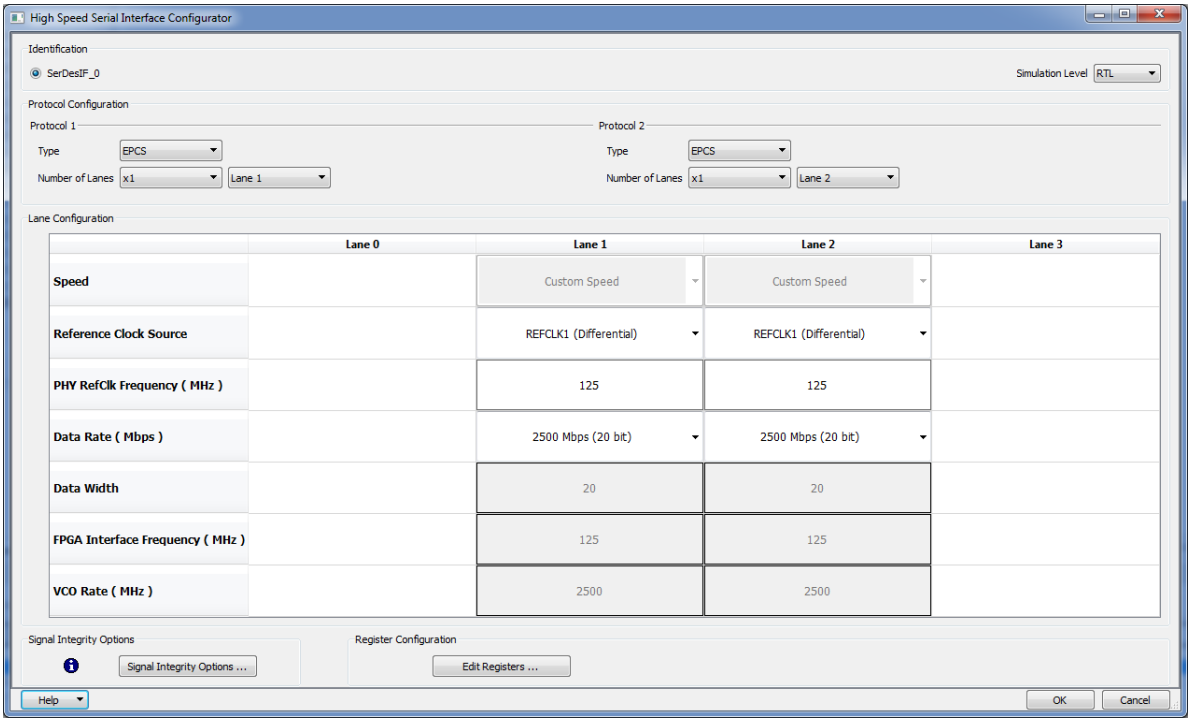
2. In the **Create New SmartDesign** dialog box, enter the **Name** as **EPCS_SERDES**.

Figure 34 • **New SmartDesign Setup**



3. Go to the **Catalog** tab, select the **Peripherals** category, and drag the **High Speed Serial Interface** onto the **EPCS_SERDES** SmartDesign canvas. If the component appears shadowed in the Vault, right-click the name and select **Download**.
4. Double-click **SERDES_IF_0** component in the SmartDesign canvas and open the **SERDES** Configurator. Configure the SERDES, as shown in Figure 35.

Figure 35 • **SERDESIF Configurator Settings**

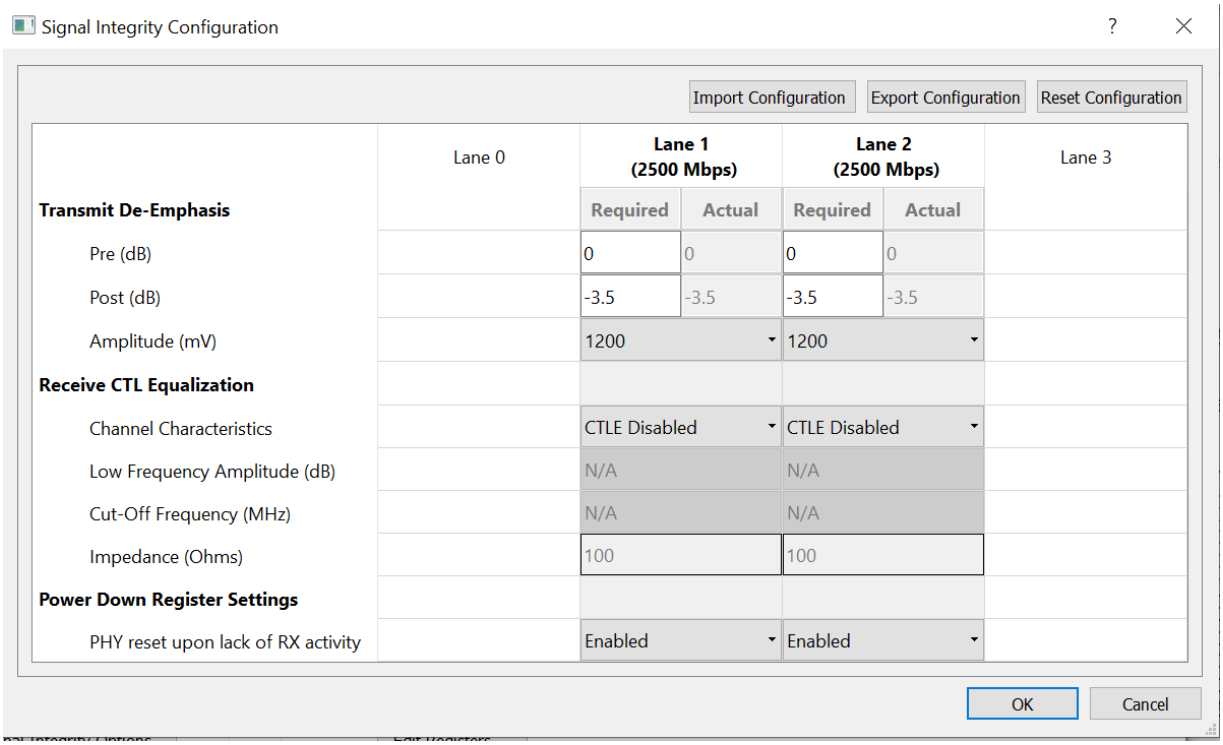


5. Enter the following information:

- Identification
  - SerDesIF_0
  - Simulation Level: RTL
- Protocol Configuration
  - Protocol 1: Type: EPCS, Number of Lanes: x1, Lane 1
  - Protocol 2: Type: EPCS, Number of Lanes: x1, Lane 2
- Lane Configuration
  - Speed: Lane[1:2] CUSTOM SPEED
  - Reference Clock Source: REFCLK1 (Differential)
  - PHY RefClk Frequency: 125 MHz
  - Data Rate: 2500 Mbps (20 bit) for all lanes

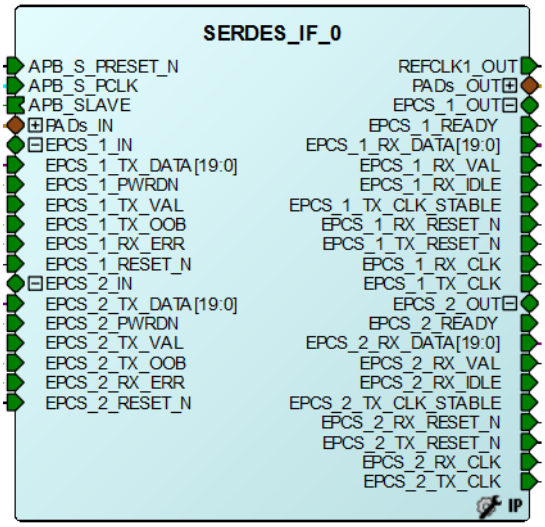
6. Click **Signal Integrity** option to adjust the programmable SerDes input and output parameters (refer Figure 35).

Figure 36 • **Signal Integrity Options window**



7. Click **OK**. Figure 37 shows the SERDESIF component.

Figure 37 • **SERDESIF Component**



8. From the **Design Hierarchy** tab, drag the appropriate imported HDL modules listed under the **work** subdirectory to the EPCS_SERDES canvas.
   - Add two epcs_tx_intf modules
   - Add two epcs_rx_intf modules
   - Add two delay_line modules

   You can modify the module names by selecting the module name. The module name appears in bold on top of the module.

3.6.3.1 Connecting Components in the EPCS_SERDES SmartDesign

Connect the EPCS_SERDES SmartDesign modules by using one of the following three methods:

1. In the first method, click **Connection Mode** on the **SmartDesign** window. The cursor changes from the normal arrow icon to the connection mode icon. Select a pin that you want to connect and drag it onto the pin that you want to connect.
2. In the second method, hold the Ctrl key and select the pins that you want to connect, right-click, and select **Connect**.

   Right-click the input source signal and select **Connect** to connect all the signals together. Similarly, select the input source signal, right-click, and select **Disconnect** to disconnect the signals that are already connected.
3. In the third method, click **Quick Connect** mode on the **SmartDesign** window. The Quick connect window is displayed. Select the pin in **Instance Pin** that you want to connect, select the pin in **Pins to Connect**, right-click, and select **Connect**.

Using any of the methods, connect the pins mentioned in Table 3.

Syntax: ModuleName:PortName

Table 3 • **EPCS_SERDES Interconnections**

From	To
SERDES_IF_0:EPCS_1_TX_DATA[19:0]	epcs_tx_intf_1:txdout[19:0]
SERDES_IF_0:EPCS_1_RX_VAL	epcs_rx_intf_1:rxvali
SERDES_IF_0:EPCS_1_RX_DATA[19:0]	delay_line1:in_data[19:0]
SERDES_IF_0:EPCS_1_RX_RESET_N	epcs_rx_intf_1:rstn
SERDES_IF_0:EPCS_1_TX_RESET_N	epcs_tx_intf_1:rstn
SERDES_IF_0:EPCS_1_RX_CLK	epcs_rx_intf_1:clk
SERDES_IF_0:EPCS_1_TX_CLK	epcs_tx_intf_1:clk
SERDES_IF_0:EPCS_2_TX_DATA[19:0]	epcs_tx_intf_2:txdout[19:0]
SERDES_IF_0:EPCS_2_RX_VAL	epcs_rx_intf_2:rxvali
SERDES_IF_0:EPCS_2_RX_DATA[19:0]	delay_line2:in_data[19:0]
SERDES_IF_0:EPCS_2_RX_RESET_N	epcs_rx_intf_2:rstn
SERDES_IF_0:EPCS_2_TX_RESET_N	epcs_tx_intf_2:rstn
SERDES_IF_0:EPCS_2_RX_CLK	epcs_rx_intf_2:I_clk
SERDES_IF_0:EPCS_2_TX_CLK	epcs_tx_intf_2:clk
epcs_rx_intf_1:rxdin[19:0]	delay1:out_data[19:0]
epcs_rx_intf_2:rxdin[19:0]	delay2:out_data[19:0]

4. Module pins can be promoted to the top-level. Right-click **[PIN]** and select **promote to top level**. To rename the promoted pins, right-click the pin and select **Rename Top Level Pin**.

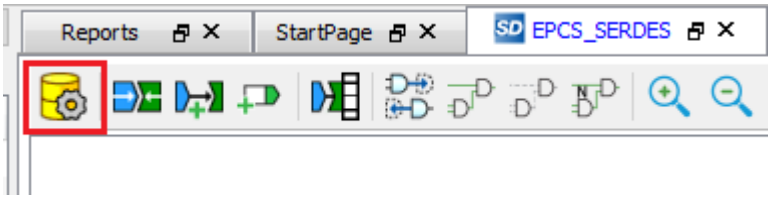
Table 4 lists the pins that must be promoted to top-level (as instructed above) and can be optionally renamed.

Table 4 • **EPCS_SERDES Connection Renames**

Module	Pin Name	Renamed
SERDES_IF_0	APB_S_PRESET_N	–
SERDES_IF_0	APB_S_PCLK	–
SERDES_IF_0	APB_SLAVE	–
SERDES_IF_0	REFCLK1_OUT	–
SERDES_IF_0	EPCS_1_READY	Lane1_READY
SERDES_IF_0	EPCS_1_RX_IDLE	Lane1_RX_IDLE
SERDES_IF_0	EPCS_1_RX_RESET_N	Lane1_RX_RESET_N
SERDES_IF_0	EPCS_1_TX_RESET_N	Lane1_TX_RESET_N
SERDES_IF_0	EPCS_1_RX_CLK	Lane1_RX_CLK
SERDES_IF_0	EPCS_1_TX_CLK	Lane1_TX_CLK
SERDES_IF_0	EPCS_2_RX_RESET_N	Lane2_RX_RESET_N
SERDES_IF_0	EPCS_2_TX_RESET_N	Lane2_TX_RESET_N
SERDES_IF_0	EPCS_2_RX_CLK	Lane2_RX_CLK
SERDES_IF_0	EPCS_2_TX_CLK	Lane2_TX_CLK
epcs_tx_intf_1	txdin[19:0]	Lane1_TX_data[19:0]
epcs_tx_intf_2	txdin[19:0]	Lane2_TX_data[19:0]
epcs_rx_intf_1	rxdout[19:0]	Lane1_RX_data[19:0]
epcs_rx_intf_2	rxdout[19:0]	Lane2_RX_data[19:0]
epcs_rx_intf_1	rxvalo	Lane1_RX_VAL
epcs_rx_intf_2	rxvalo	Lane2_RX_VAL
SERDES_IF_0	REFCLK1_OUT	REFCLK_OUT

5. Connect the following pins to the top-level EPCS_RESET_N pin:
   - EPCS_1_RESET_N
   - EPCS_2_RESET_N
6. Hold the Ctrl key, select the following SERDES_IF_0 ports, right-click the ports, and select **Mark Unused**:
   - EPCS_1_TX_CLK_STABLE
   - EPCS_2_READY
   - EPCS_2_RX_IDLE
   - EPCS_2_TX_CLK_STABLE
7. Hold the Ctrl key, select the following ports of **SERDES_IF_0**, right-click the ports, and select **Tie Low**:
   - EPCS_1_PWRDN
   - EPCS_1_TX_OOB
   - EPCS_1_RX_ERR
   - EPCS_2_PWRDN
   - EPCS_2_TX_OOB
   - EPCS_2_RX_ERR
8. Hold the Ctrl key, select the following **SERDES_IF_0** ports, right-click, and select **Tie High**.
   - EPCS_1_TX_VAL
   - EPCS_2_TX_VAL
9. Click **Generate Component to generate the EPCS_SERDES component**.

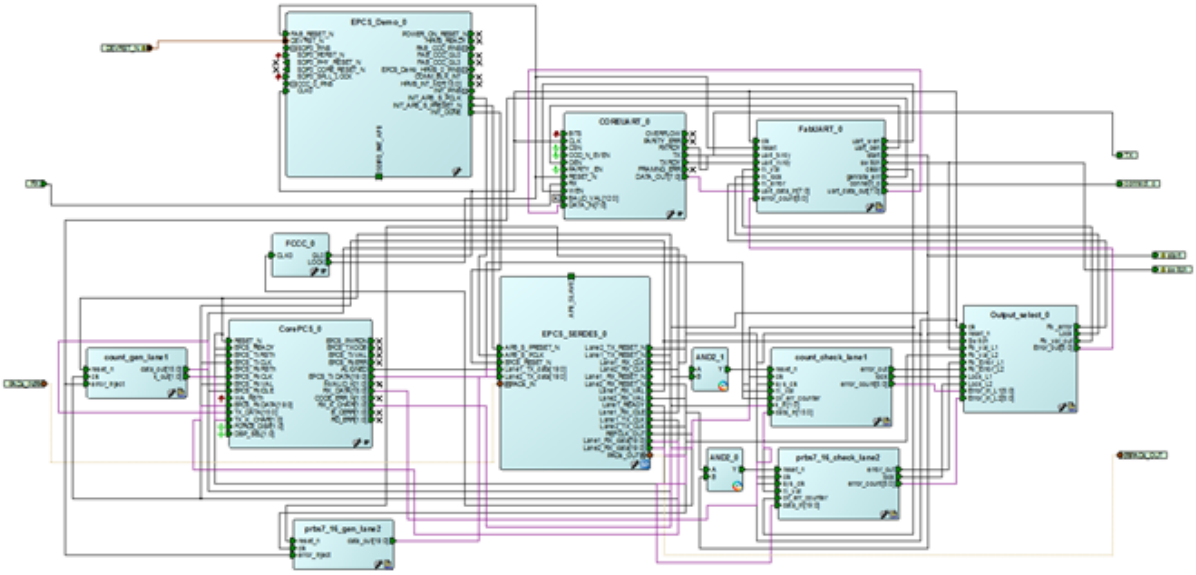
Figure 38 • **EPCS_SERDES Component Generation**



The EPCS_SERDES' was generated message is displayed on the **Libero SoC Log** window, if the design is generated without any errors. The Log window is displayed after generating the component successfully.

Figure 39 shows the EPCS_SERDES canvas in SmartDesign after configuring all the components

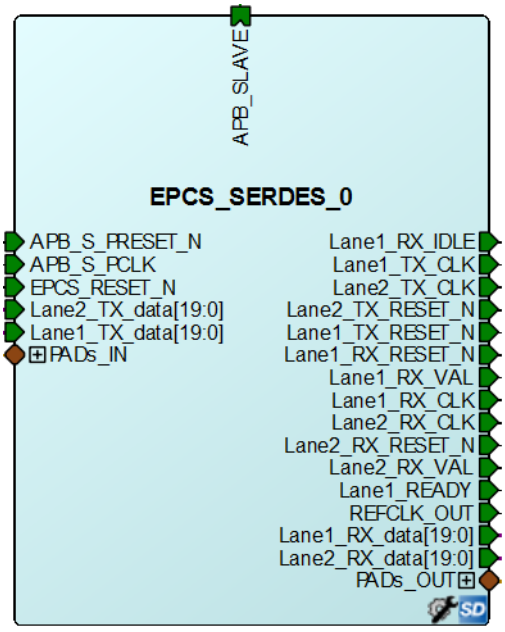
Figure 39 • **Complete EPCS_SERDES SmartDesign Canvas**



3.6.4 Step 4: Finalizing SmartDesign

1. Open the `EPCS_Demo_top` SmartDesign canvas and drag the EPCS_SERDES component from the **Design Hierarchy** onto the **EPCS_Demo_top** SmartDesign canvas. It is displayed with the top-level port names on the instance, as shown in Figure 40.

Figure 40 • **EPCS_SERDES Component**



2. To add the user HDL modules to the EPCS_Demo_top SmartDesign canvas, go to the **Design Hierarchy** tab, drag the appropriate imported HDL modules listed under the **work** subdirectory to the EPCS_SERDES canvas.
   - Add count_gen
   - Add count_check
   - Add Output_select
   - Add prbs7_16_check
   - Add prbs7_16_gen
   - Add FabUART

Note: You can modify the module names by selecting the module name.

Table 5 shows the EPCS_Demo_top ports that need to be connected on the SmartDesign canvas.

Syntax: ModuleName:PortName

Table 5 • **EPCS_Demo_top Interconnections**

From	To
CorePCS_0:RX_DATA[15:0]	count_check_lane1:data_in[15:0]
CorePCS_0:EPCS_TX_DATA[19:0]	EPCS_SERDES_0:Lane1_TX_data[19:0]
CorePCS_0:RX_K_CHAR[1:0]	count_check_lane1:k_in[1:0]
CorePCS_0:ALIGNED	count_check_lane1:rx_val
CorePCS_0:RESET_N:EPCS_TxRSTn	count_gen_lane1:reset_n
	EPCS_SERDES_0:Lane1_TX_RESET_N
CorePCS_0:EPCS_READY	EPCS_SERDES_0:Lane1_READY
CorePCS_0:EPCS_TxCLK	EPCS_SERDES_0:Lane1_TX_CLK
	count_gen_lane1:clk
CorePCS_0:EPCS_RxRSTn	EPCS_SERDES_0:Lane1_RX_RESET_N
	AND2_1:A
CorePCS_0:EPCS_RxCLK	EPCS_SERDES_0:Lane1_RX_CLK
	count_check_lane1:clk
CorePCS_0:EPCS_RxVAL	EPCS_SERDES_0:Lane1_RX_VAL
	Output_select_0:Rx_Val_L1
CorePCS_0:EPCS_RxIDLE	EPCS_SERDES_0:Lane1_RX_IDLE
CorePCS_0:EPCS_RxDATA[19:0]	EPCS_SERDES_0:Lane1_RX_data[19:0]
CorePCS_0:TX_DATA[15:0]	count_gen_lane1:data_out[15:0]
CorePCS_0:TX_K_CHAR[1:0]	count_gen_lane1:k_out[1:0]
EPCS_SERDES_0:Lane2_RX_RESET_N	AND2_0:A
count_gen_lane1:error_inject	prbs7_16_gen_lane2:error_inject
	FabUART_0:generate_error
prbs7_16_gen_lane2:reset_n	EPCS_SERDES_0:Lane2_TX_RESET_N
prbs7_16_gen_lane2:clk	EPCS_SERDES_0:Lane2_TX_CLK
prbs7_16_gen_lane2:Data_out[19:0]	EPCS_SERDES_0:Lane2_TX_data[19:0]
EPCS_Demo_0:INIT_APB_S_PCLK	EPCS_SERDES_0:APB_S_PCLK
EPCS_Demo_0:INIT_APB_S_PRESET_N	EPCS_SERDES_0:APB_S_RESET_N
EPCS_Demo_0:INIT_DONE	EPCS_SERDES_0:EPCS_RESET_N
COREUART_0:WEN	FabUART_0:uart_wen
COREUART_0:OEN	FabUART_0:uart_oen
COREUART_0:DATA_IN[7:0]	FabUART_0:uart_data_out[7:0]
COREUART_0:RXRDY	FabUART_0:uart_rxrdy
COREUART_0:TXRDY	FabUART_0:uart_txrdy
COREUART_0:DATA_OUT[7:0]	FabUART_0:uart_data_in[7:0]
FabUART_0:rx_val	Output_select_0:Rx_val_out
FabUART_0:rx_lock	Output_select_0:Lock

Table 5 • **EPCS_Demo_top Interconnections *(continued)***

From	To
FabUART_0:rx_error	Output_select_0:Rx_error
FabUART_0:error_count[5:0]	Output_select_0:Error_out[5:0]
FabUART_0:start	Output_select_0:reset_n
	AND2_0:B
	AND2_1:B
FabUART_0:switch	Output_select_0:switch
FabUART_0:clear	count_check_lane1:clr_err_counter
	prbs7_16_check_lane2:clr_err_counter
AND2_0:Y	prbs7_16_check_lane2:reset_n
AND2_1:Y	count_check_lane1:reset_n
prbs7_16_check_lane2:clk	EPCS_SERDES_0:Lane2_RX_CLK
Prbs7_16_check_lane2:data_in[19:0]	EPCS_SERDES_0:Lane2_RX_data[19:0]
count_check_lane1:error_out	Output_select_0:RX_Error_L1
count_check_lane1:lock	Output_select_0:Lock_L1
count_check_lane1:error_count[5:0]	Output_select_0:Error_In_L1[5:0]
prbs7_16_check_lane2:error_out	Output_select_0:RX_Error_L2
prbs7_16_check_lane2:lock	Output_select_0:Lock_L2
prbs7_16_check_lane2:error_count[5:0]	Output_select_0:Error_In_L2[5:0]

3. Right-click **[PIN]** and select **promote to top level to promote the following pins**. To rename the pin names, right-click the pin and select **Rename Top Level Pin**.
   - COREUART_0: RX
   - FabUART_0: switch
   - FabUART_0: start
   - FabUART_0: connect_o
   - COREUART_0: TX
4. Hold the Ctrl key, select the following **CorePCS_0** ports, right-click, and select **Tie Low**.
   - FORCE_DISP[1:0]
   - DSP_SEL[1:0]
5. Hold the Ctrl key, select the following **COREUART_0** ports, right-click, and select **Tie Low**.
   - CSN
   - ODD_N_EVEN
   - PARITY_EN
6. Hold the Ctrl key, select the following **CorePCS_0** ports, right-click, and select **Tie High**.
   - WA_RSTn
7. Hold the Ctrl key, select the following **EPCS_Demo_0** ports, right-click, and select **Tie High**.
   - SDIF0_PERST_N
   - SDIF0_SPLL_LOCK
8. Hold the Ctrl key, select the following **COREUART_0** ports, right-click, and select **Tie High**.
   - BIT8
9. Hold the Ctrl key, select the following **COREUART_0** ports, right-click, and select **Mark Unused**.
   - OVERFLOW
   - PARITY_ERR
   - FRAMING_ERR

10. For the M2GL010 device, hold the Ctrl key, select the following **EPCS_Demo_0** ports, right-click, and select **Mark Unused**.
    - SDIF0_PHY_RESET_N
    - SDIF0_CORE_RESET_N
    - POWER_ON_RESET
    - HPMS_READY
    - SDIF_READY
    - FAB_CCC_GL3
    - COMM_BLK_INT
    - HPMS_INTM2F[15:0]
11. For M2S090 device, hold the Ctrl key, select the following **EPCS_Demo_0** ports, right-click, and select **Mark Unused**.
    - POWER_ON_RESET_N
    - SDIF_READY
    - MSS_READY
    - SDIF0_PHY_REST_N
    - SDIF0_0_CORE_RESET_N
    - SDIF0_1_CORE_RESET_N
    - FAB_CCC_GL1
12. Hold the Ctrl key, select the following **CorePCS_0** ports, right-click, and select **Mark Unused**.
    - EPCS_PWRDN
    - EPCS_TXOOB
    - EPCS_TxVAL
    - EPCS_RxERR
    - INVALID_K[1:0]
    - CODE_ERR_N[1:0]
    - B_CERR[1:0]
    - RD_ERR[1:0]
13. Click **Generate Component to generate the EPCS_Demo_top component**.

Figure 41 • **EPCS_Demo_top Component Generation**

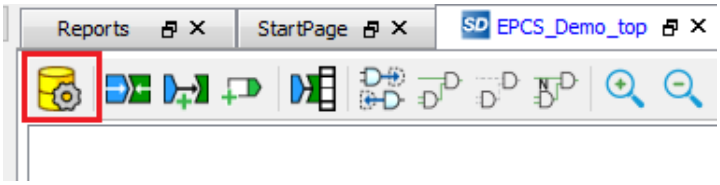
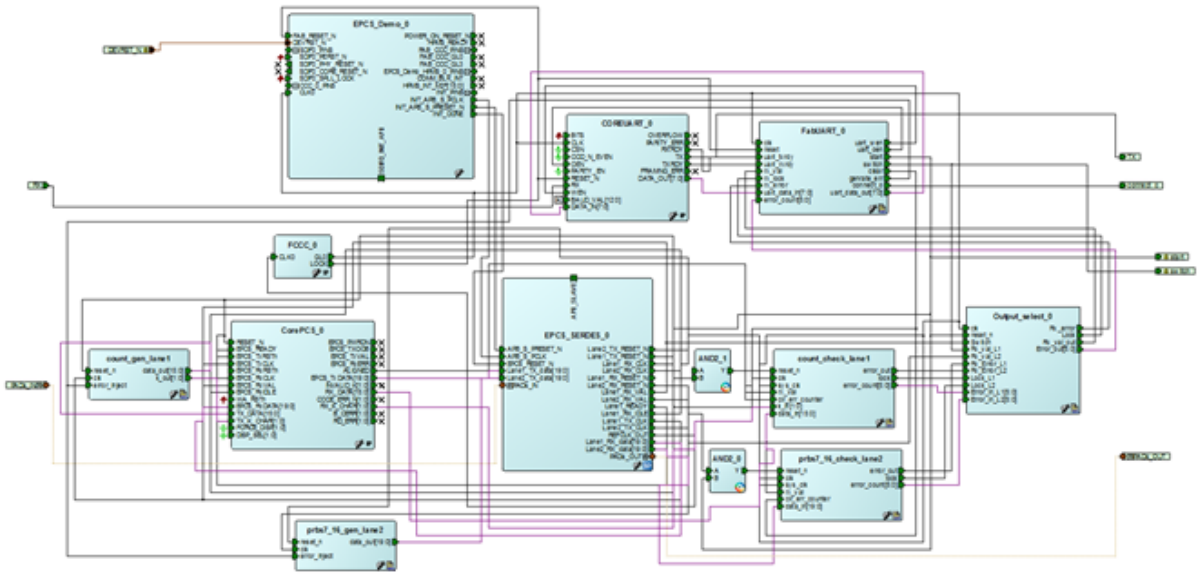


Figure 42 shows the Complete EPCS_Demo_top SmartDesign Canvas.

Figure 42 • **Complete EPCS_Demo_top SmartDesign Canvas**

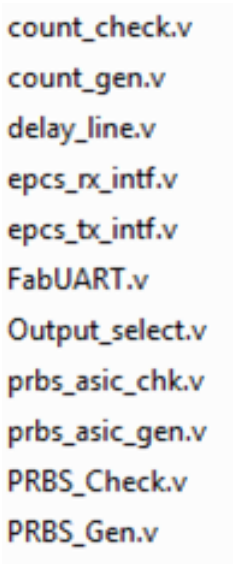


3.6.4.1 Design Files

The Libero project contains all the design files required to create this project. These files can downloaded from: *http://soc.microsemi.com/download/rsc/?f=m2gl_tu0570_df*

The Verilog HDL source files are located in the hdl directory.

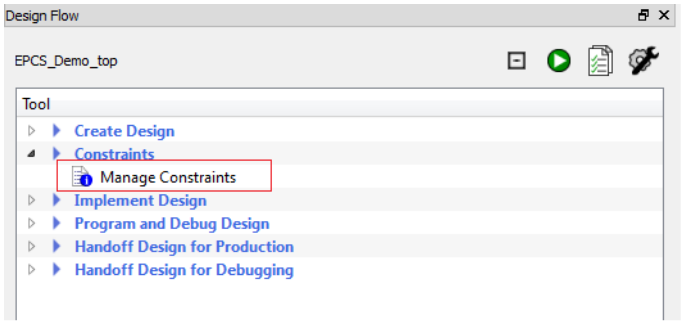
Figure 43 • **hdl Directory**



3.6.5 Step 5: Running Constraints Manager

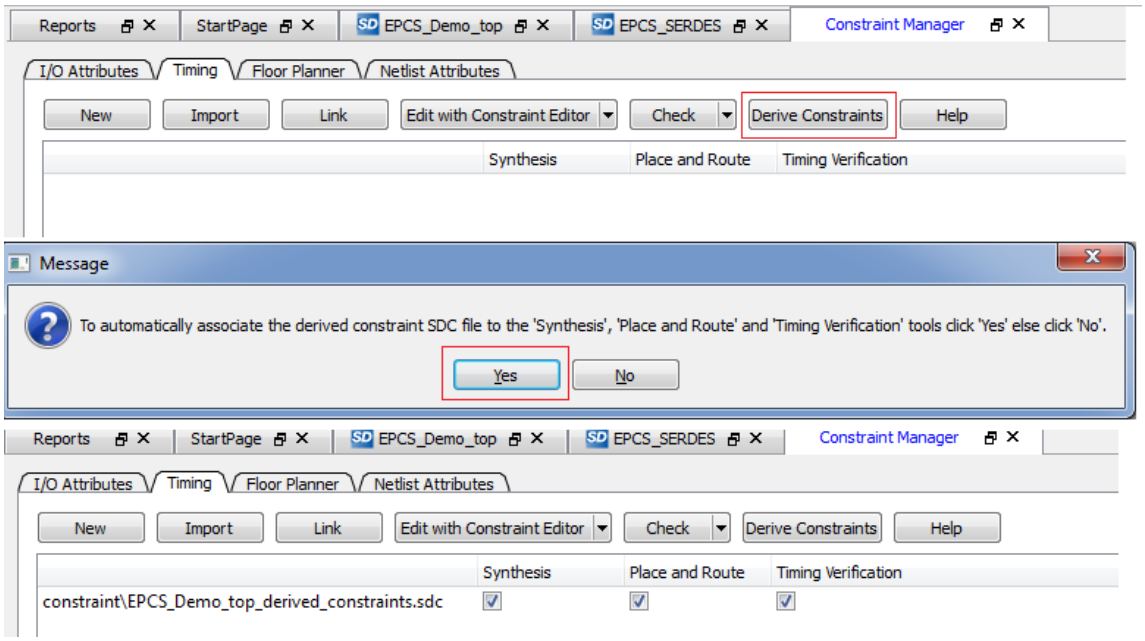
1. On the **Design Flow** window, double-click **Manage Constraints**.

Figure 44 • **Manage Constraints**



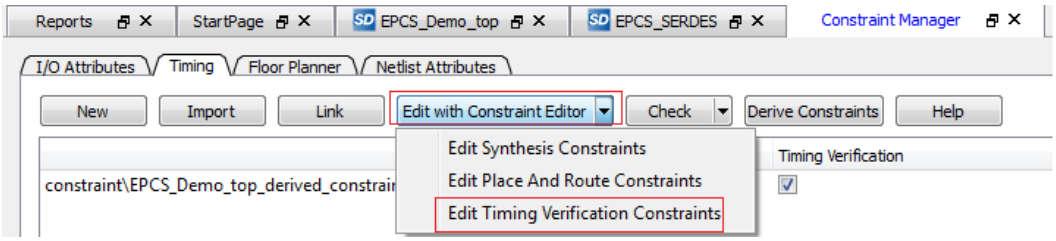
2. On the **Timing** tab, click **Derive Constraints** and select **Yes** to associate the derived SDC file to Synthesis, Place and Route, and Timing Verification.

Figure 45 • **Derive Constraints - Timing Tab**



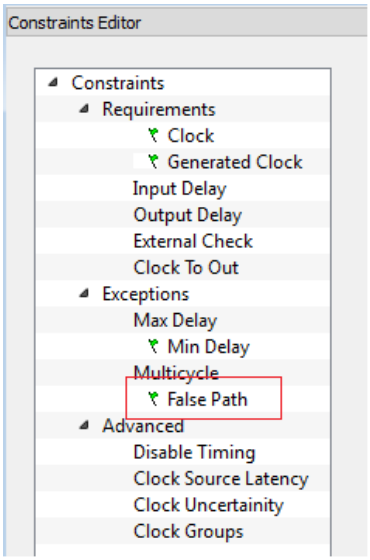
3. Click **Edit with Constraints Editor** and select **Edit Timing Verification Constraints** from the drop-down menu. This step is to edit timing paths that are not relevant when timing reports are generated.

Figure 46 • **Edit with Constraints - Timing Tab**



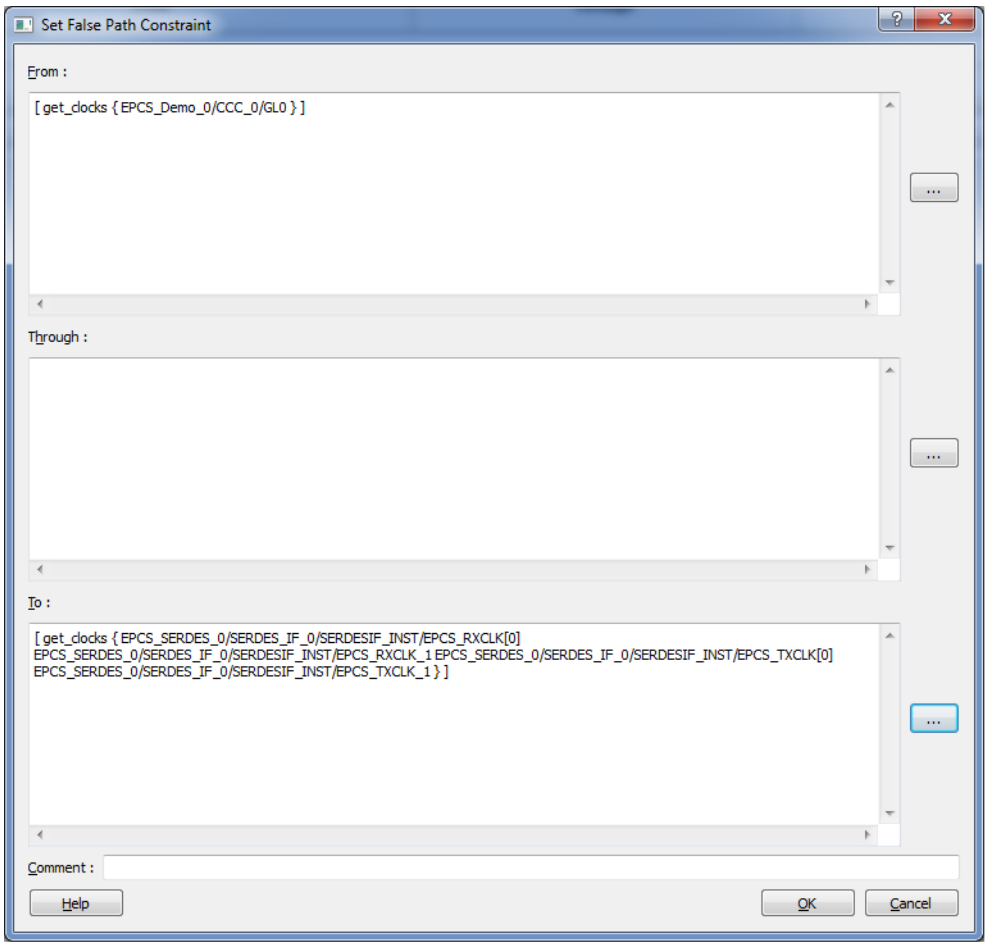
4. On the **Constraints Editor** window, double-click **False Path**.

Figure 47 • **Constraints Editor**



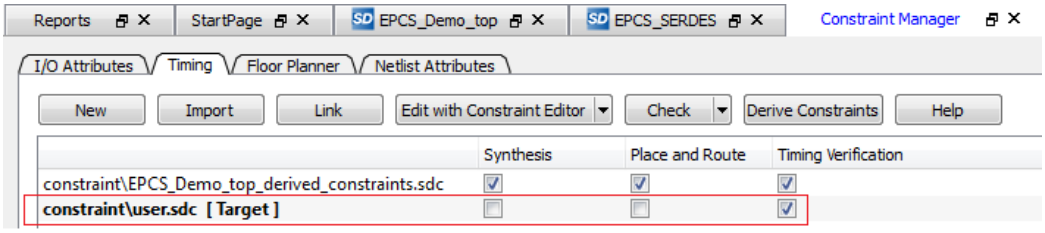
5. Specify the False Path between the system clock from GL0 to the high-speed SerDes RX and TX clocks and from the high-speed SerDes RX and TX clocks to GL0.

Figure 48 • **Set False Path Constraint**



6. Click **OK**.
7. Select the newly created user.sdc to be used with Timing Verification.

Figure 49 • **Selecting user.sdc File**



The I/O Editor can be opened to review the I/O placement imported from the I/O constraints in Step 1.

Figure 50 • **Invoke I/O Editor**

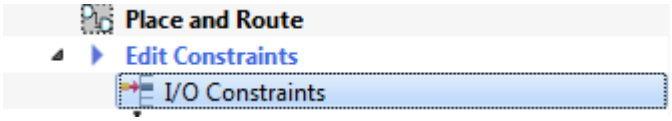
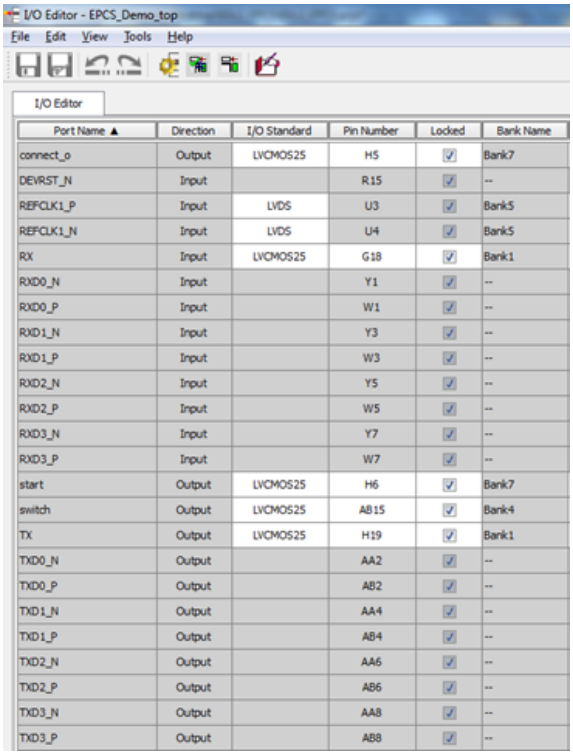


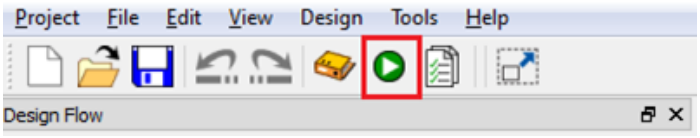
Figure 51 • **I/O Constraints Editor**



3.6.6 Step 6: Generate Program Data

Click **Generate Programming Data** to complete the place-and-route, verify timing, and generate the programming file.

Figure 52 • **Generate Programming Data**

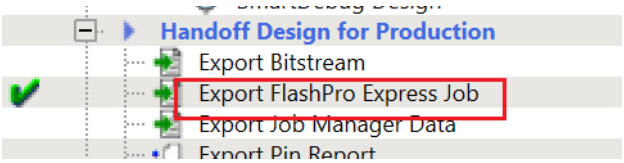


3.6.6.1 Export Programming File

Click **Export Programming File** to export a `.job` programming file to:

{LiberoProjectDirectory}\designer\top\export

Figure 53 • **Exporting Programming File**



4 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the SmartFusion2 and IGLOO2 devices with the programming job file using FlashPro Express.

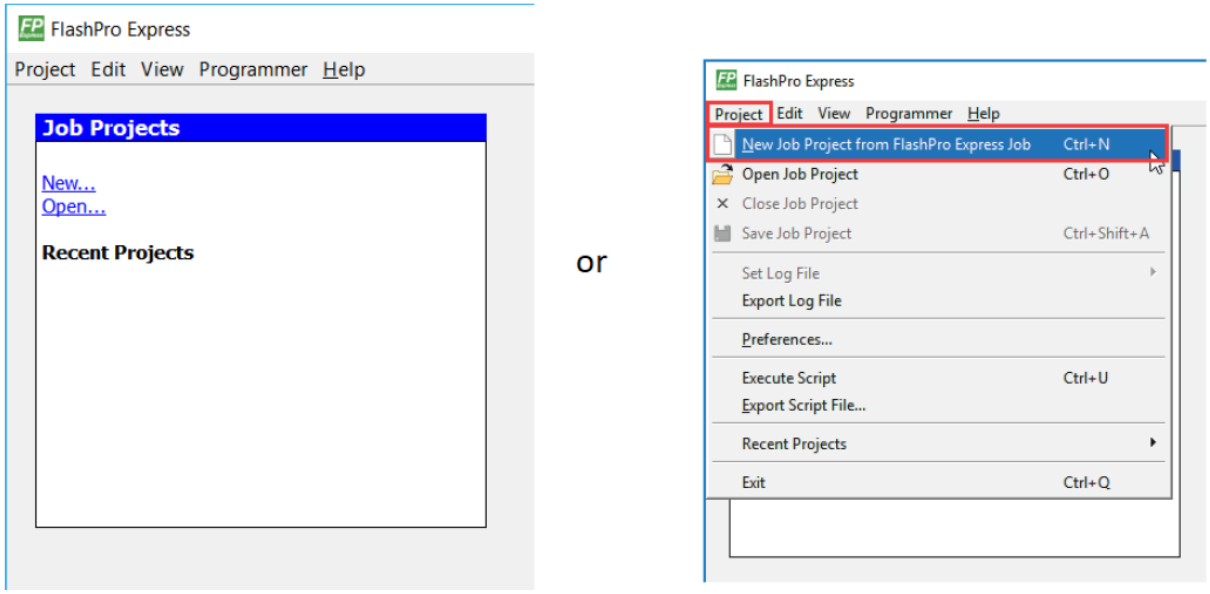
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 2, page 9.

Note: The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the **J6** connector on the board.
3. Power **ON** the power supply switch **SW7**.
4. On the host PC, launch the **FlashPro Express** software.
5. 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.

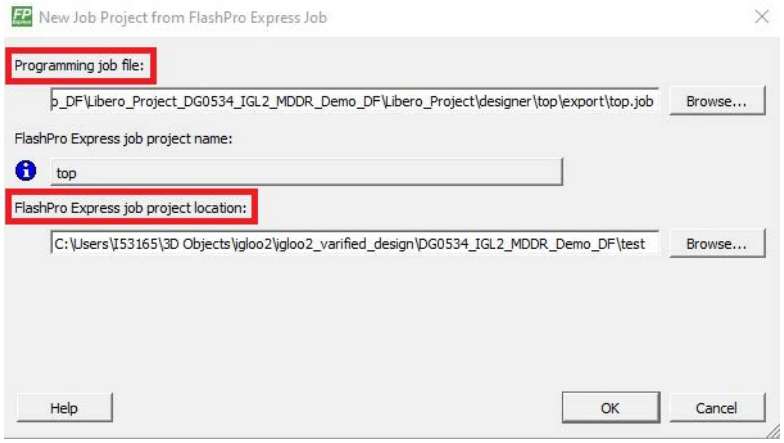
Figure 54 • **FlashPro Express Job Project**



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

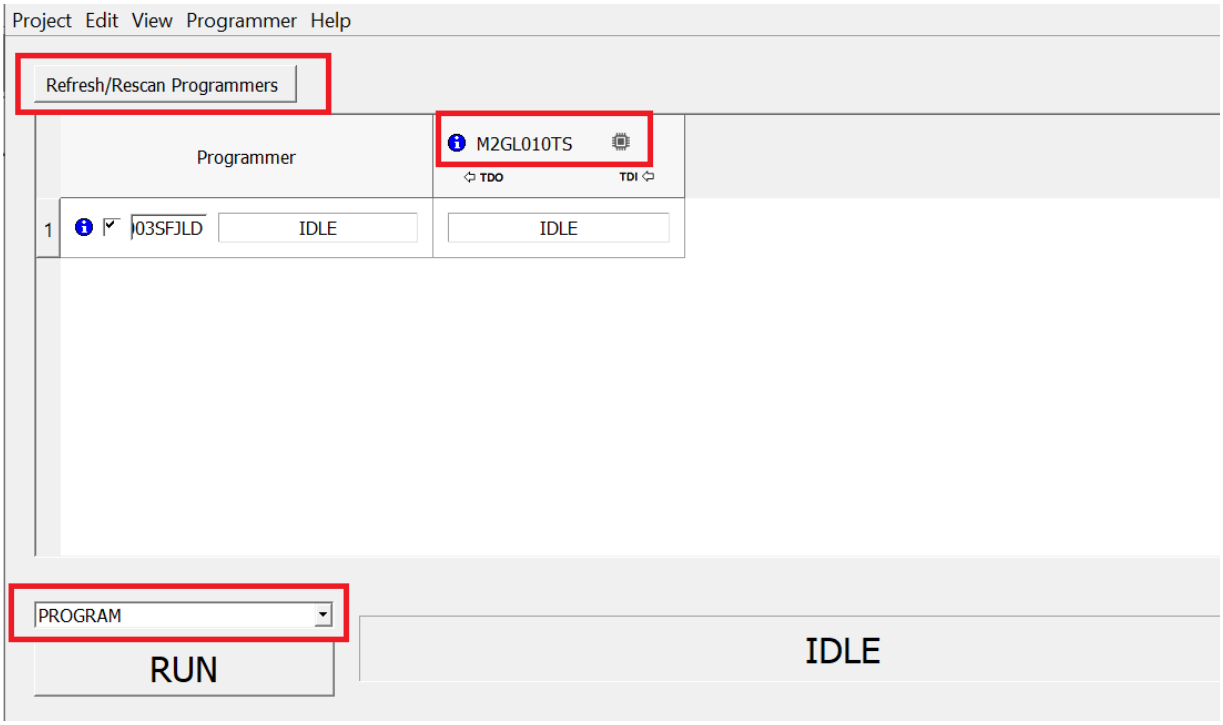
- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:
  <download_folder>\m2gl_tu0570_df\Programming_Job
- **FlashPro Express job project name:** Click **Browse** and navigate to the location where you want to save the project.

Figure 55 • **New Job Project from FlashPro Express Job**



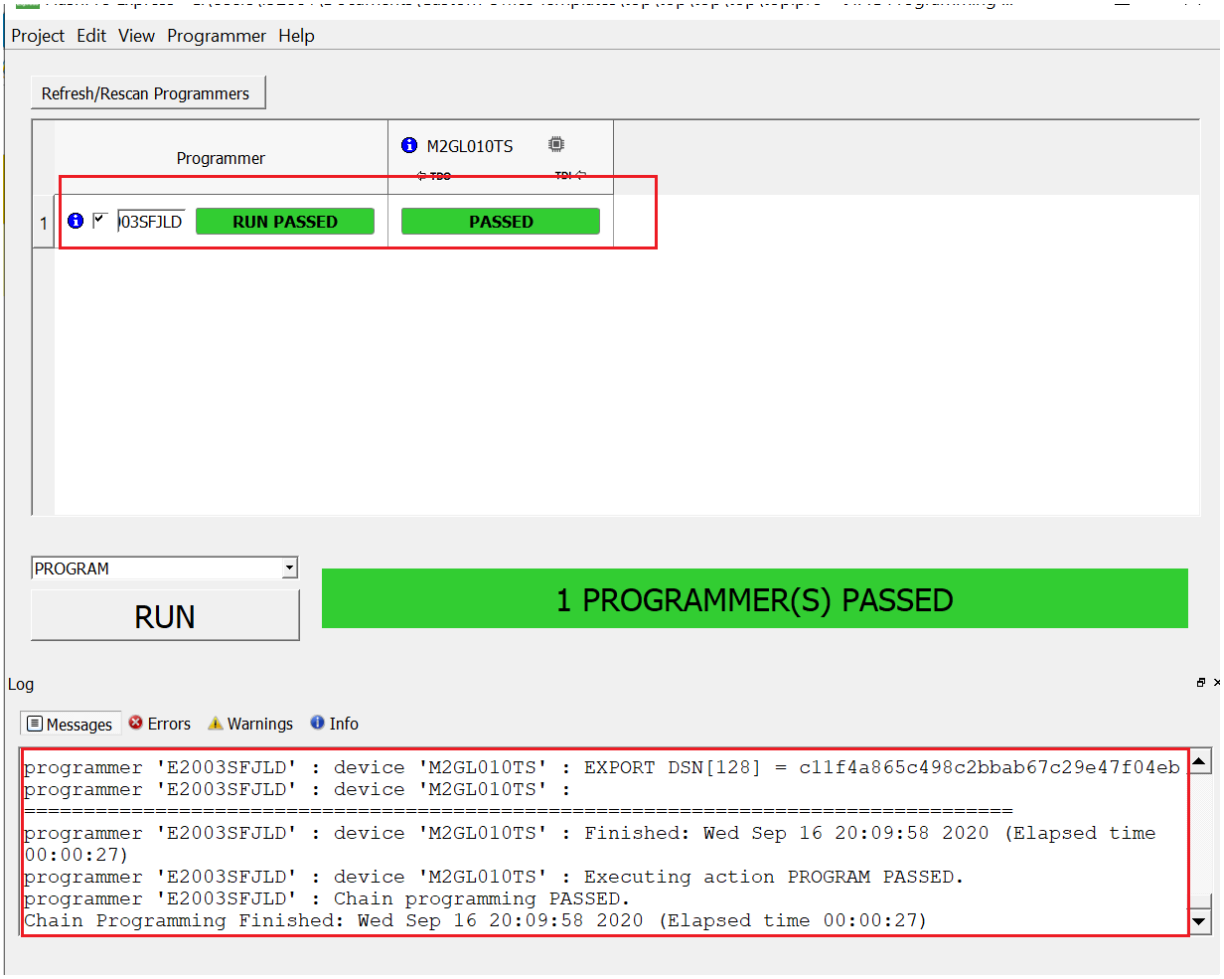
7. Click **OK**. The required programming file is selected and ready to be programmed in the device.
8. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan**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

Figure 56 • **Programming the Device**



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

Figure 57 • **FlashPro Express—RUN PASSED**



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

5 Appendix 2: Using SmartFusion2 and IGLOO2 for Customer Design

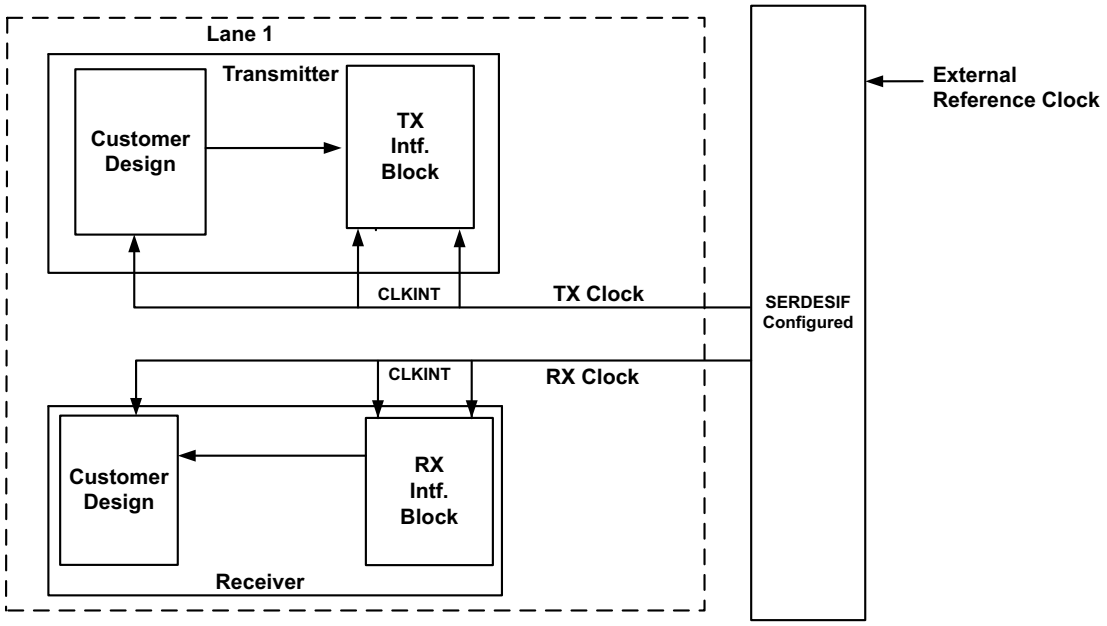
5.1 Transmitter Section

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 58.

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 block, as shown in Figure 58.

Figure 58 • **Replacing Demo Design with Customer Design**



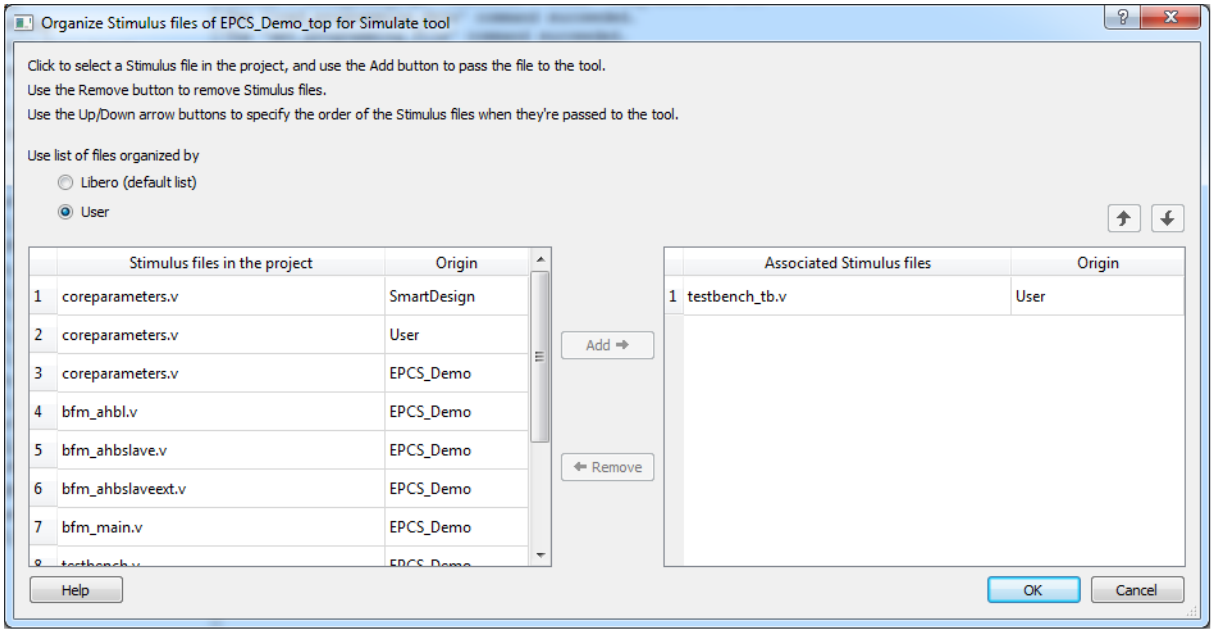
This demo is targeted to M2GL010T and M2S090T devices, it is not required to force the assignment of a clock resource. In the larger SmartFusion2 and IGLOO2 devices, Microsemi recommends using a regional clock to reduce the clock injection time into the fabric. This can be done by instantiating an RCLKINT library element on the clock outputs of SERDESIF EPCS TX_CLK and RX_CLK.

The interface blocks for the transmitter and receiver are also highly recommended to achieve timing closure. These blocks employ a scheme specifically designed for the SmartFusion2 and IGLOO2 family and optimizes the interface for both setups and hold. These blocks must be used exactly from this demo design into all EPCS designs. Verilog HDL is used in this tutorial. VHDL modules are available in the SOURCE Directory.

6 Appendix 3: Simulating the Design

Figure 59 shows the Organize Stimulus files window.

Figure 59 • **Organizing Simulation Testbench in Project**



Follow the steps to simulate the design:

1. Copy the testbench_tb.v file to the **Stimulus** folder of the Libero project. Right-click **Simulate** and select **Organize Input files > Organize Stimulus Files** to setup the `testbench_tb.v` file for simulation.
2. Copy the `wave.do` file to the Simulation folder of the Libero project.

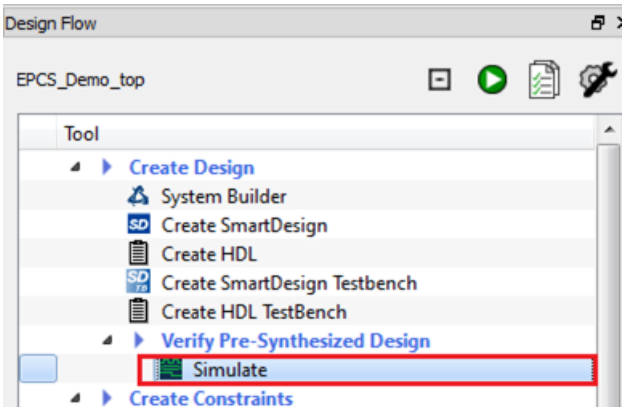
Note: The `wave.do` and `testbench_tb.v` files are available in the Test benches folder of the EPCS_Demo project.

3. Go to **Project > Project Settings > waveforms** and select the **Include DO file** checkbox.
4. Change the **Simulation runtime** to 2 µs using the **Do File** option under **Project Settings**.
5. Select `vsim command` under **Project Settings** and change the resolution to **1 ps**.
6. Click **Save** to save the settings.
7. Right-click **Simulate** in the **Libero Design Flow** and select **Open Interactively**.

The simulation for this design is done through the testbench. It simulates the high-speed serial interface block in the EPCS mode.

8. To run the simulation, double-click **Simulate** under **Verify Pre-Synthesized Design** in the **Design Flow** window of the Libero project, as shown in Figure 60.

Figure 60 • **Simulating the Design**

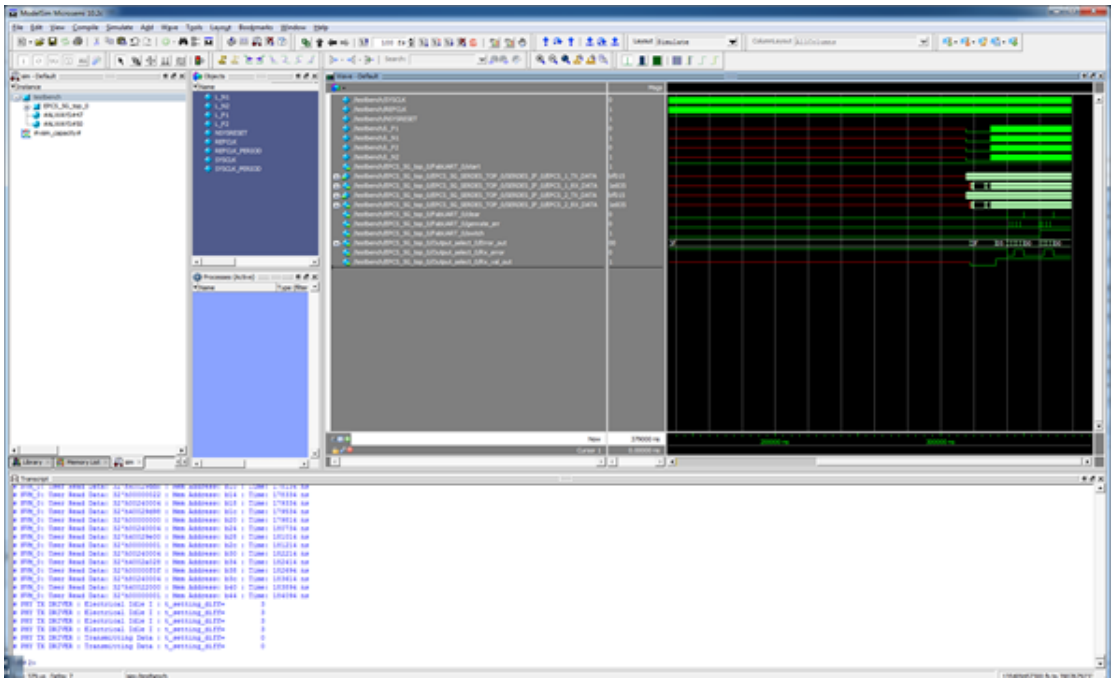


The simulation runs automatically, observe the simulation results for Lane1 and Lane2.

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 various steps of the SERDES initialization and operation. After 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 61.

Figure 61 • **Simulation Waveform Window**



6.1 Acceleration of SERDES EPCS Designs

The simulation example provided with this demo uses an acceleration technique to reduce the initialization time for the SERDES block. After the final SERDES register is written over the APB interface from HPMS, the SERDES block is held in reset mode for several hundred microseconds. Using a Verilog force command, the INIT_DONE output of the HPMS can be forced high after the SERDES peripheral is initialized. This code is found in the `testbench_tb.v` file and can be used in any EPCS design with a modification of the hierarchy naming.

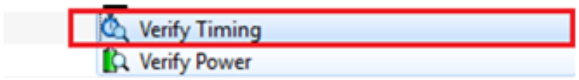
7 Appendix 4: Verifying Timing using SmartTime

SmartTime is a gate-level static timing analysis tool. With SmartTime, you can perform complete timing analysis of the design to ensure that all timing constraints are met and the design operates at the desired speed with the right amount of margin across all operating conditions. For more information, refer to the **SmartTime Users Guide** from the Libero SoC software Help.

7.1 Verify Timing

1. Right-click **Verify Timing** as shown in Figure 62. **Verify Timing** generates timing reports and violations reports for both Max and Min Delay Analysis.

Figure 62 • **Verify Timing**



2. Double-click **Open SmartTime** to **Launch SmartTime** as shown in Figure 63.

Figure 63 • **Open SmartTime**

The SmartTime window is displayed in **Max Delay Analysis View**, as shown in Figure 64.

Figure 64 • **SmartTime Session**

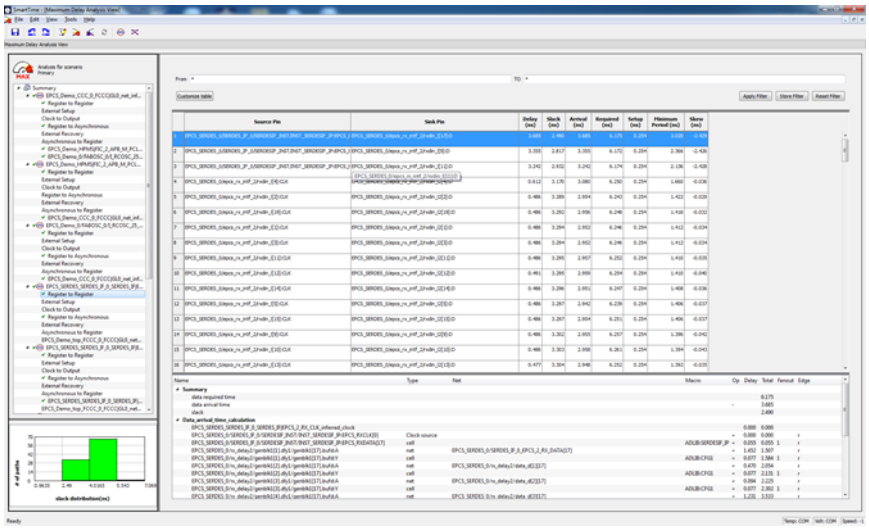


Figure 64 shows the SmartTime results. **Min Delay Analysis** is also available to generate timing information. Both the reports highlight any setup or hold time violations that cause design issues in the actual hardware.

8 Appendix 5: Status Signals

Table 6 describes the various status signals.

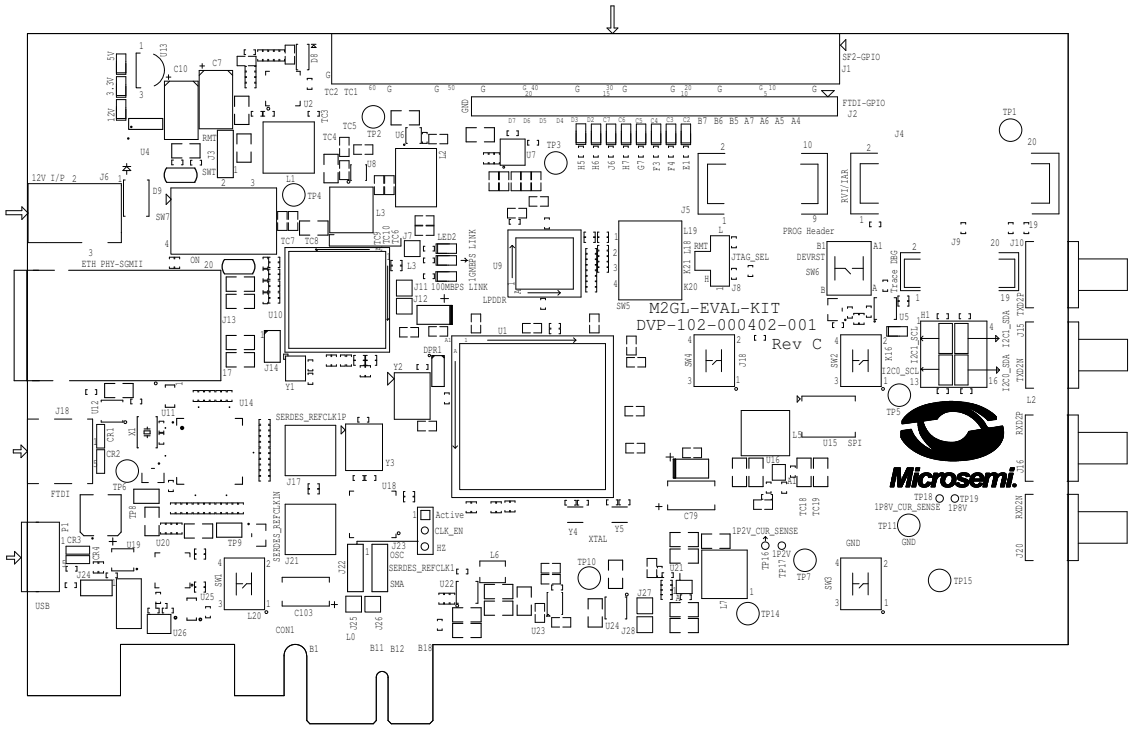
Table 6 • **Status Signals**

Status Signal	Description
Host Connection	Indicator of COM port connection on 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 is receiving valid and error-free data. It means that the receiver is locked to the PRBS7 sequences, and the subsequent transmitted sequences can be successfully received. RED: The receiver is receiving 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 sequences.
Error Count	Gives the count of errors detected in the received PRBS sequences.
Generate Error	Used to introduce errors in the transmission for debug purposes. Injects the error in the transmitted PRBS sequence, which increments the Error Count display.
Clear Error	Sets error count to zero.

9 Appendix 6: Jumper Locations

Figure 65 shows the jumper locations in the IGLOO2 Evaluation Kit.

Figure 65 • **IGLOO2 Evaluation Kit Silkscreen Top View**



Note: The location of the jumpers in Figure 65 is searchable.