Implementing a SmartFusion2/IGLOO2 SERDES EPCS Protocol Design - Libero SoC v11.4

Tutorial

January 2015

Microsemi

Table of Contents

Preface

About this document

This demo is for the SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) or IGLOO®2 FPGA devices. The demo demonstrates the use of SmartFusion2/IGLOO2 serializer/de-serializer (SERDES) in the extended physical code sublayer (EPCS) mode. The EPCS and serializer/deserializer interface (SERDESIF) in the IGLOO2 device are common to the SmartFusion2 device. Therefore the principles described in this demo can be applied to both the SmartFusion2 and IGLOO2 devices. It provides instructions on how to use the corresponding demo design.

Intended Audience

SmartFusion2/IGLOO2 devices are used by:

- FPGA designers
- System-level designers

References

Microsemi Publications

- SmartFusion2 SoC FPGA Programming User Guide
- SmartFusion2 SoC FPGA High Speed Serial Interfaces User Guide
- IGLOO2 FPGA Programming User Guide
- IGLOO2 FPGA High Speed Serial Interfaces User Guide

Refer to the following web page for a complete and up-to-date listing of SmartFusion2 device documentation: http://www.microsemi.com/products/fpga-soc/soc-fpga/sf2docs

Refer to the following web page for a complete and up-to-date listing of IGLOO2 device documentation: http://www.microsemi.com/products/fpga-soc/fpga/igloo2docs#documents

Implementing an SmartFusion2 and IGLOO2 SERDES EPCS Protocol Design

Introduction

The SmartFusion2 and IGLOO2 family of devices have embedded high speed SERDES blocks that can handle data rates from 1 Gbps to 5 Gbps. The high speed serial interface block, also known as SERDESIF, supports many serial communication standards. This module integrates several functional blocks to support multiple high speed serial protocols within the FPGA.

The extended physical code sublayer (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, but 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 IGLOO2/SmartFusion2 device.

The CorePCS IP module supports programmable 8B10B encoding and decoding. 8B10B is commonly used in some protocols that are not included in the SERDESIF block by the Microsemi SoC high speed SERDES interface, so CorePCS is ideal to use 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. Refer to *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 IGLOO2/SmartFusion2 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

Design Requirements

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
M2GL/M2S-EVAL-KIT IGLOO2/SmartFusion2 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
12V 2A wall-mounted power supply	1
STAPL/PDB file	Generated after running through the Libero SoC design flow
GUI Software	Provided in the design files archive file
Host PC or Laptop	Windows 7 64-bit Operating System
Software Requirements	
Libero SoC	v11.4 SP1
FlashPro Programming Software	v11.4 SP1
Host PC Drivers	USB to UART drivers
Framework	Microsoft .NET Framework 4 client for launching demo GUI

Demo Design

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=IGLOO2_EPCS_Demo_11p4_sp1_DF

SmartFusion2: http://soc.microsemi.com/download/rsc/?f=SmartFusion2_EPCS_Demo_11p4sp1_DF

The demo design files include:

- GUI installer
- Libero project
- Programming file
- SourceFiles
- Test benches
- TCL script

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

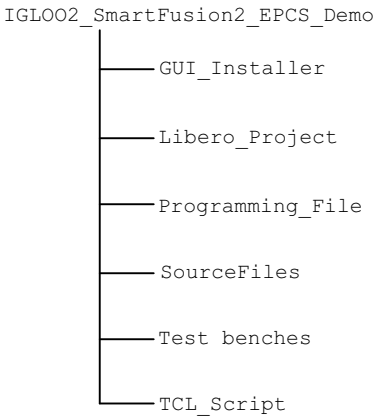


Figure 1 • **Demo Design Files Top-Level Structure**

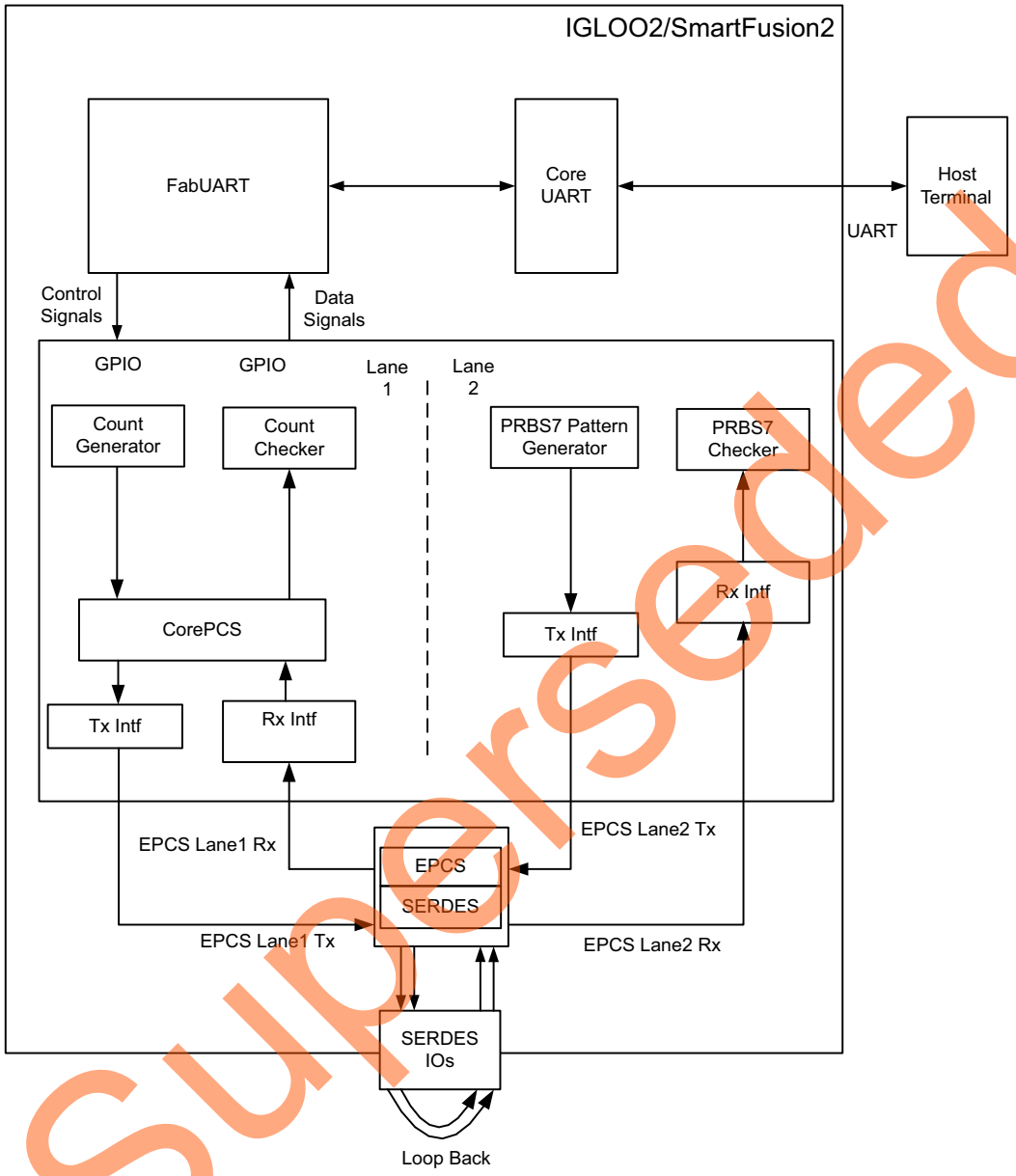
This example design demonstrates transmitting a pseudorandom binary sequence (PRBS) or counting pattern over the IGLOO2/SmartFusion2 high speed SERDES interface. The SERDES block is configured for 2.5 Gbps operational speed. The PRBS pattern is sent over Lane 2 and a counting 8B10B encoded pattern is 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 also utilizes 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 design require a general understanding to correctly meet performance between the EPCS interface and the FPGA fabric.

The system block diagram for the design implemented in an IGLOO2/SmartFusion2 device is as shown in Figure 2.



Note:
For loop back
- 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.

Figure 2 • **Demo Design Block Diagram**

Demo Design 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 IGLOO2/SmartFusion2 SERDES block. Each of these blocks is further explained in the following sections:

- PRBS7 Generator
- Count Generator
- PRBS7 Checker
- Count Checker
- Delay Line
- RX Intf/TX Intf

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.

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.

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.

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.

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 IGLOO2 or SmartFusion2 family devices.

RX Intf/TX Intf

RX and TX interfaces modules manage the timing relationships of the clock and data from the EPCS interface to the FPGA fabric.

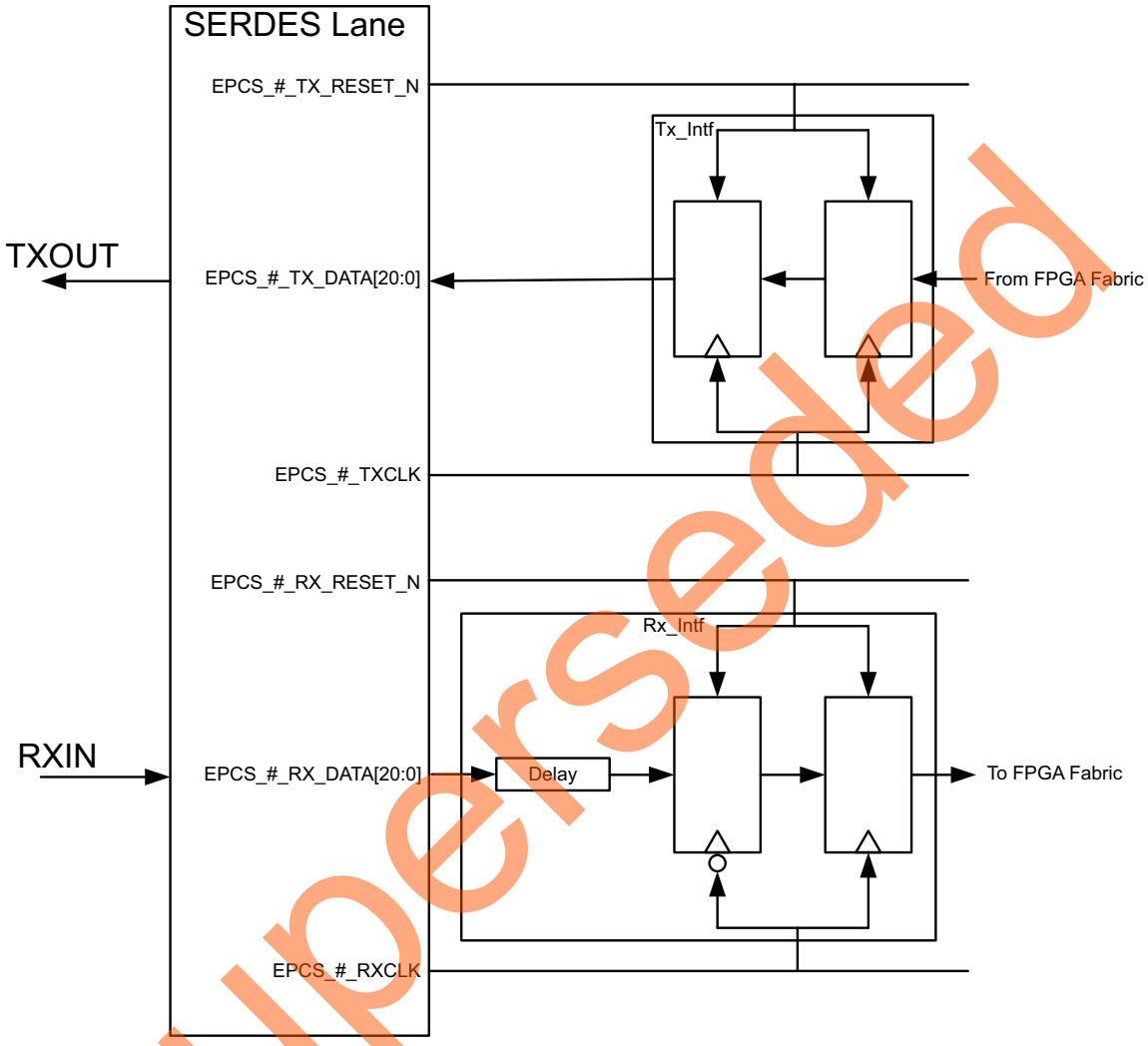


Figure 3 • **TX and RX Interface RTL Blocks**

CoreUART, FabUART, and Output Select Modules

The COREUART module is used to communicate with UART interface on the IGLOO2 Evaluation 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 both Lane 1 and Lane 2. Depending on the Lane selection, it feeds the corresponding status signals onto the UART.

SERDES

The IGLOO2/SmartFusion2 high speed SERDES is a hard IP block on chip that supports rates upto 5 Gbps. The SERDES block offers embedded protocol support for PCIe and XAUI. The SERDES block also supports EPCS interface, which can be used for custom protocols. This Tutorial configures the SERDES block in the EPCS protocol. Refer to the *IGLOO2 FPGA High Speed Serial Interfaces User Guide*/*SmartFusion2 SoC FPGA High Speed Serial Interfaces User Guide* for more information on

SERDES block. In this design, the SERDESIF block is configured to be 20-bit wide, 125 MHz REFCLK, and 2.5 Gbps.

Clocking

There are two different types of clock domains in the EPCS demo design. There is a control plane clock which is 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.

The other type of clock domain is the EPCS interface output clocks. Each SERDES lane provides an output clock for the transmitter and the receiver. The transmit clock is used to clock epcs_tx_intf and remainder of the transmit data path. The receive clock is used to clock epcs_rx_intf and remainder of the receive path.

Setting Up the Demo Design

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 one end of the USB 2.0 mini-B to the USB mini connector marked J18 on the board.
3. Connect the other end of the USB 2.0 A-male connector to the Host PC or laptop.
4. Connect 12 V 2 A-power jack to the board J6 power connector.
5. Install USB to UART drivers on the Host PC. Make sure 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.

6. Connect the jumpers on the board, as listed in Table 2. For more information on jumper locations, refer to "Appendix 5: Jumper Locations" on page 69.

   Caution: Before making the jumper connections, turn off the power supply switch.

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.

7. 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 4 shows the IGLOO2/SmartFusion2 Evaluation Kit board with cable connections.

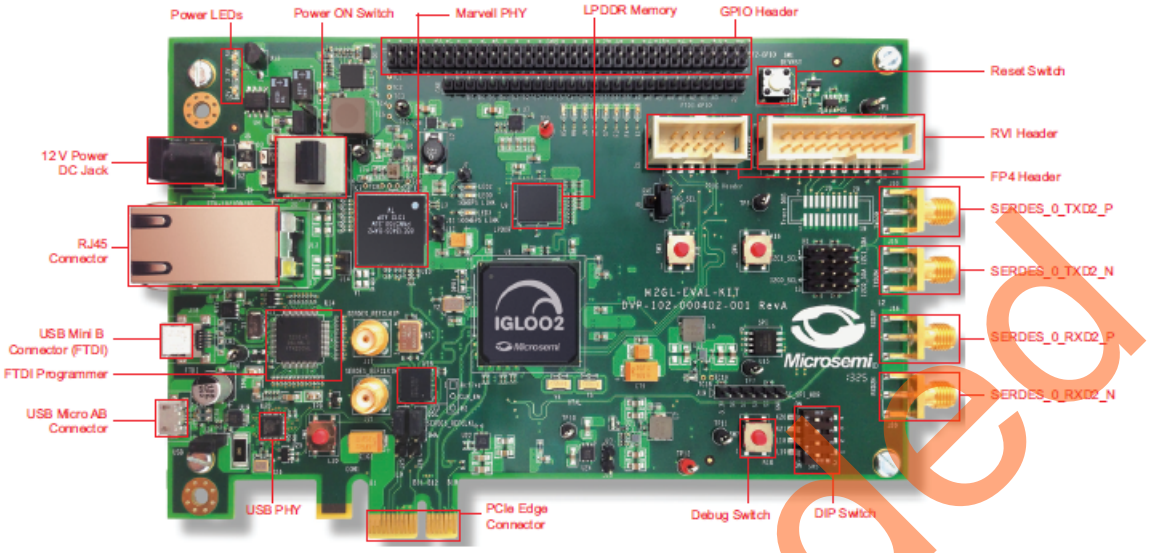


Figure 4 • **IGLOO2/SmartFusion2 Evaluation Kit Board**

Notes:

1. *IGLOO2/SmartFusion2 Evaluation Kit board use common PCB design. Figure 4 shows IGLOO2 Evaluation Kit board.*
2. *SERDES Lane 1 is looped back from Transmit to receive data on the board. Hence, it is not required to connect external SMA Loopback cables on Lane 1.*

Programming the Device

1. Download the design files from:
   IGLOO2: http://soc.microsemi.com/download/rsc/?f=IGLOO2_EPCS_Demo_11p4_sp1_DF
   SmartFusion2:
   http://soc.microsemi.com/download/rsc/?f=SmartFusion2_EPCS_Demo_11p4sp1_DF
2. Programming file (STAPL/PDB) is located in the *Programming_File* folder.
3. Connect the **FlashPro4** programmer to the IGLOO2/SmartFusion2 Evaluation Kit.
4. Open **FlashPro v11.4**, which is installed as part of the Libero SoC software.
5. Click **New Project** in FlashPro.

6. In the **New Project** dialog box, type *EPCS_Demo* in the **Project Name** field, as shown in Figure 5.

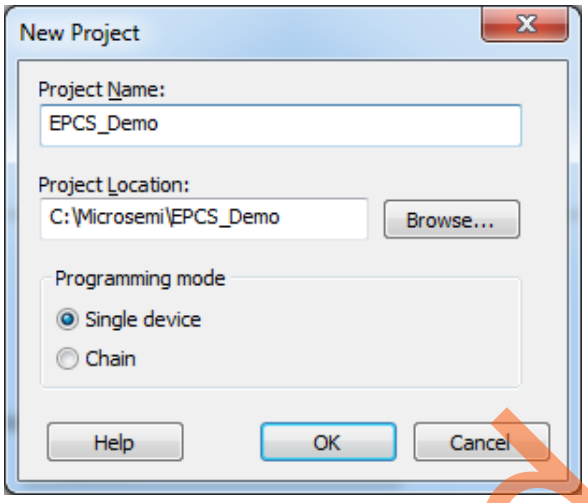


Figure 5 • **New Project Window**

7. If required, change the default location of the project in the **Project Location** field.
8. Select **Single device** as **Programming mode**.
9. Click **OK**. The FlashPro GUI is displayed. The Programmer List Window is updated with the programmer information.

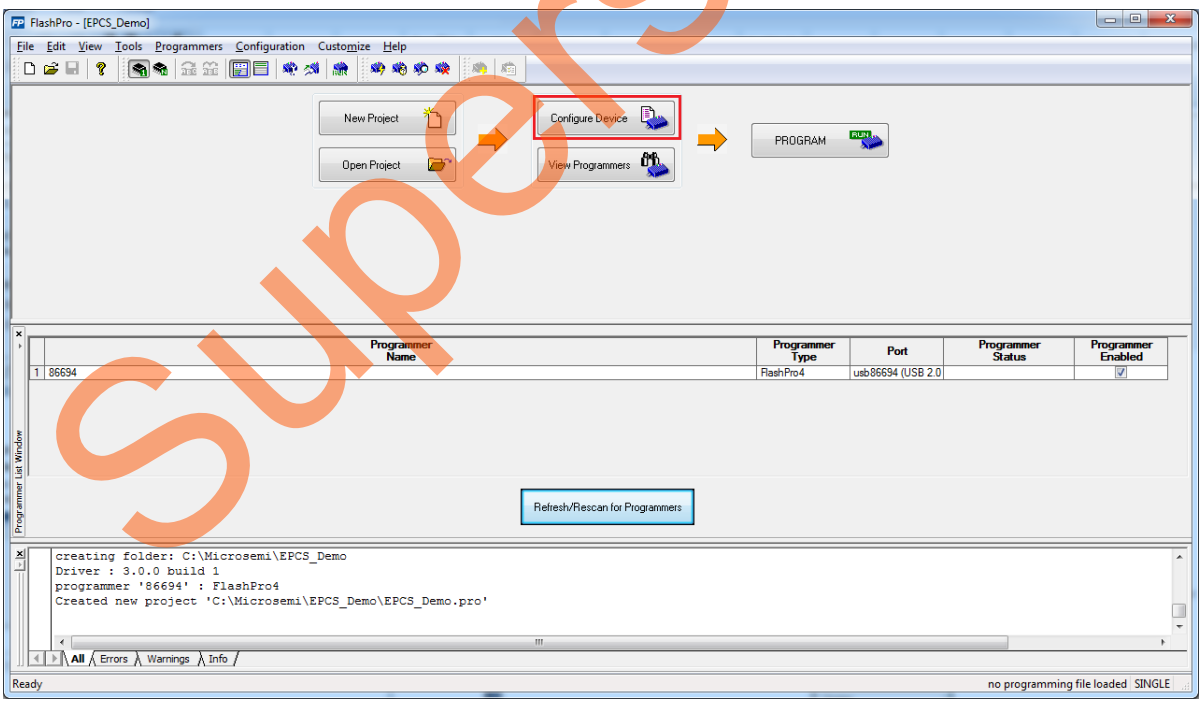


Figure 6 • **FlashPro GUI Window**

After the project is created and the programmer is connected, the STAPL/PDB file downloaded in Step 1. is ready to be loaded.

10. Click **Configure Device** (highlighted in red in Figure 6 on page 13). The **Single Device Configuration** window is displayed in FlashPro, as shown in Figure 7.
11. Click **Browse** to find the programming file. The **Load Existing Programming File** dialog box is displayed.
12. Select the programming file and click **Open**.

    The **Single Device Configuration Window** is updated to list the Programming file information and the actions available with your Programming file in the Action list box Figure 7. Program is the default action displayed in the Action list box.
13. The **Single Device Configuration Window** is updated to list your Programming file information and the actions available with the Programming file in the **Action** drop-down menu**,** as shown in Figure 7. **PROGRAM** is the default action displayed in the **Action** drop-down menu.

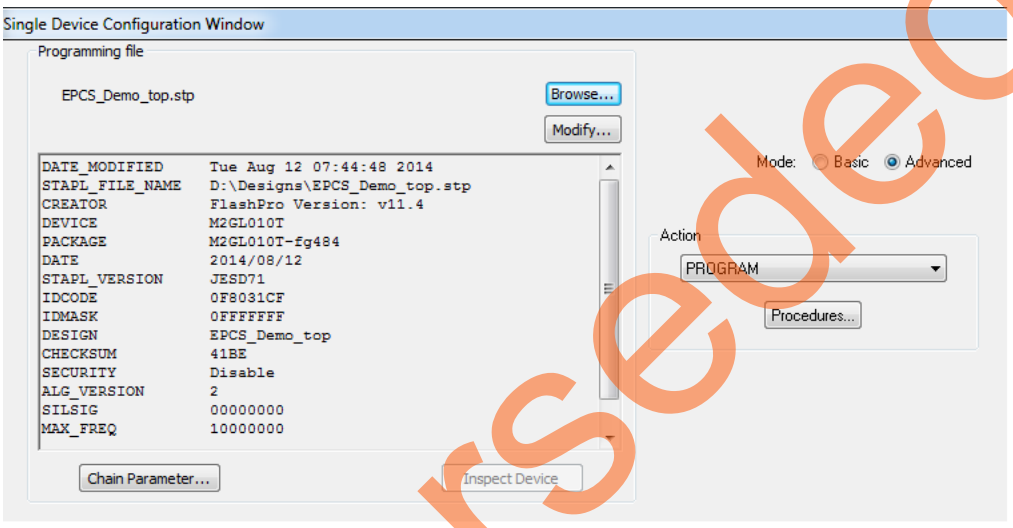


Figure 7 • **Single Device Configuration Window**

Note: It is recommended to use the default settings.

14. Click **Procedures**, as shown in Figure 8.

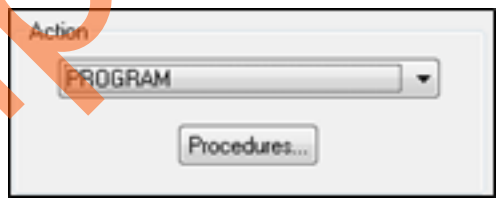


Figure 8 • **Action Window**

The **Select Action And Procedures** dialog box is displayed, showing the procedures for the **PROGRAM** action, as shown in Figure 9 on page 15.

Note: Microsemi recommends using the default settings.

15. Click **Restore Default Procedures**.

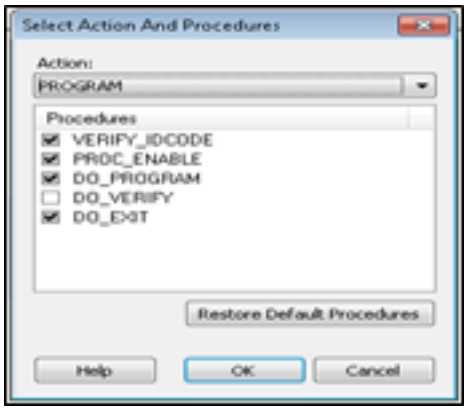


Figure 9 • **Select Action and Procedures Window**

16. In FlashPro, click **PROGRAM** to program the device.

    The Programmer List Window is displayed with **Run Passed** in the **Programmer Status** column, indicating that the device is programmed successfully, as shown in Figure 10.

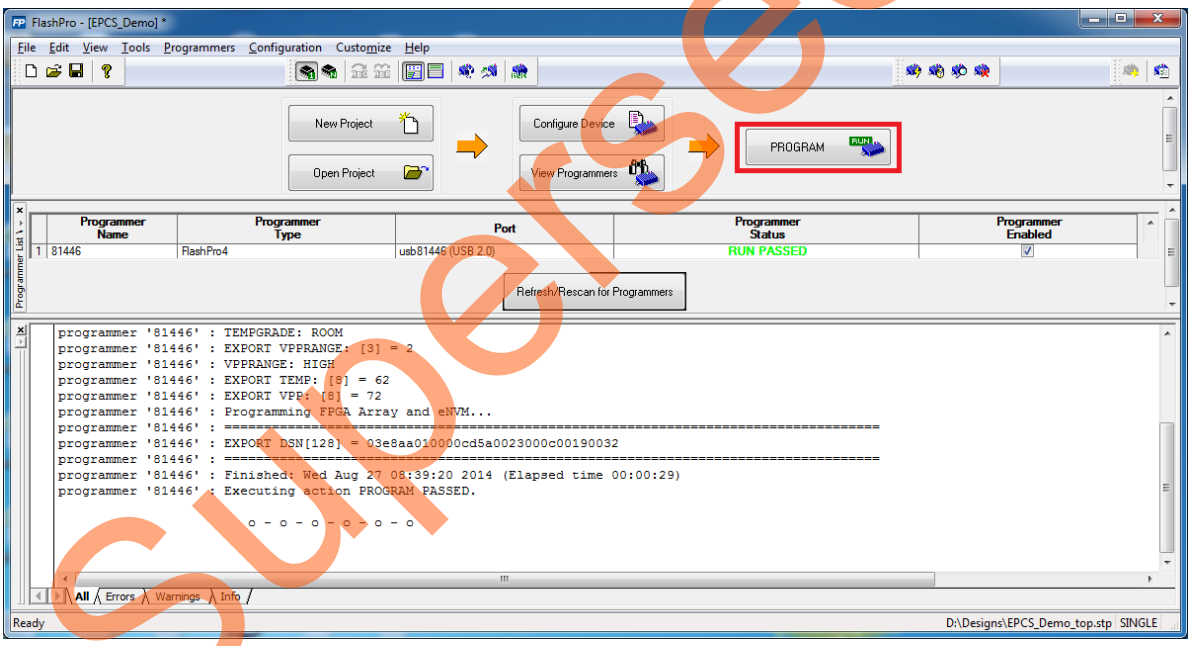


Figure 10 • **Successful Programming Session**

Note: The status indicator updates during programming to show the programming progress and changes to a pass or fail result when the operation is complete.

17. View the **Log** window and make note of the details about the programmed device.
18. Power cycle the board.

Using SmartDebug with SERDES Design

1. Open the **Libero Project** and double-click SmartDebug Design under Debug Design of the **Design Flow** tab, as shown in Figure 11.

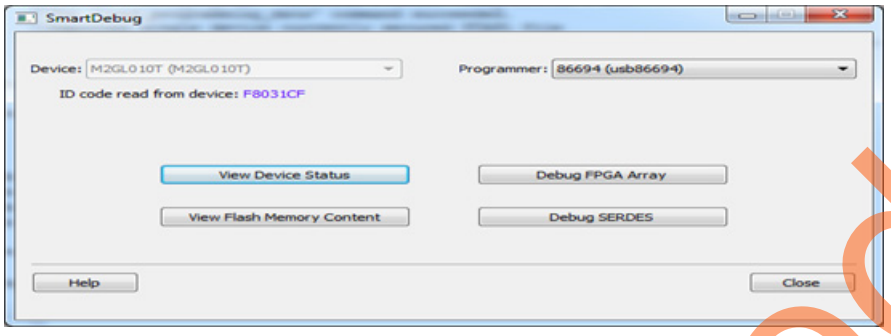


Figure 11 • **SmartDebug Window**

2. Click **Debug SERDES**.
3. The **Debug SERDES** window is displayed as shown in Figure 12.

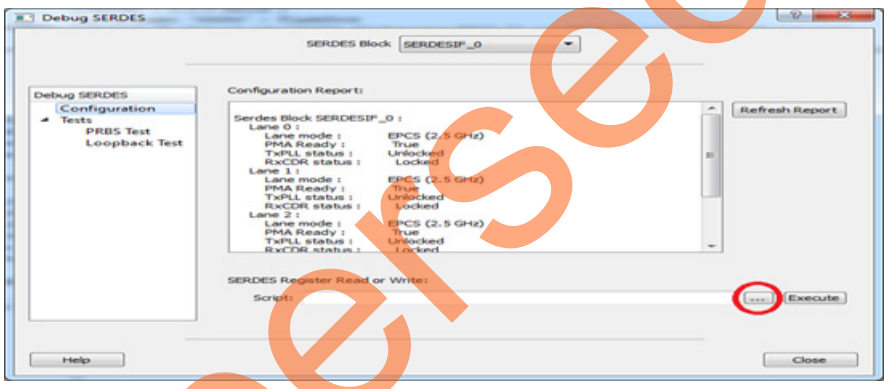


Figure 12 • **SmartDebug Window - Debug SERDES**

4. Select **Configuration** under **Debug SERDES**.
5. Click **Browse** (highlighted in Figure 12) and load the TCL script provided along with the project under *TCLScript* folder.
6. After loading the script, click **Execute**.

Note: The TCL scripts write/read to the control and status registers of the SERDES blocks. More details on the SmartDebug SERDES Debugger can be found in the *SmartFusion2 and IGLOO2 SmartDebug – Hardware Design Debug Tools Tutorial*.

Installing the GUI

1. Run the installer, if the GUI is being used for the first time.
2. Download the design files from:

   IGLOO2: http://soc.microsemi.com/download/rsc/?f=IGLOO2_EPCS_Demo_11p4_sp1_DF

   SmartFusion2: http://soc.microsemi.com/download/rsc/?f=SmartFusion2_EPCS_Demo_11p4sp1_DF
3. Open **GUI_Installer>Volume>setup.exe**.
4. Click **Yes** for any message from **User Account Control**.

   Setup window appears and default locations are displayed, as shown in Figure 13 on page 17.

5. Click **Next**.

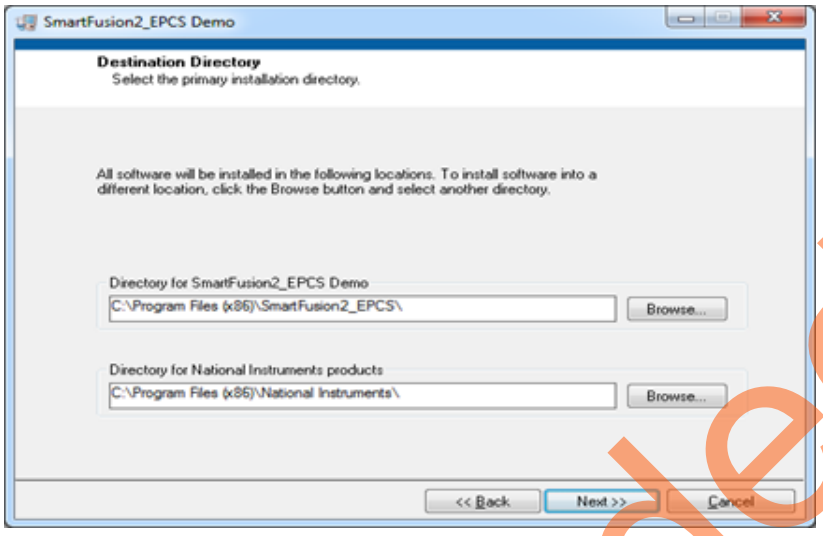


Figure 13 • **GUI Setup Window**

6. Follow the steps to begin the installation.
   A progress bar appears which shows the progress of installation, as shown in Figure 14.

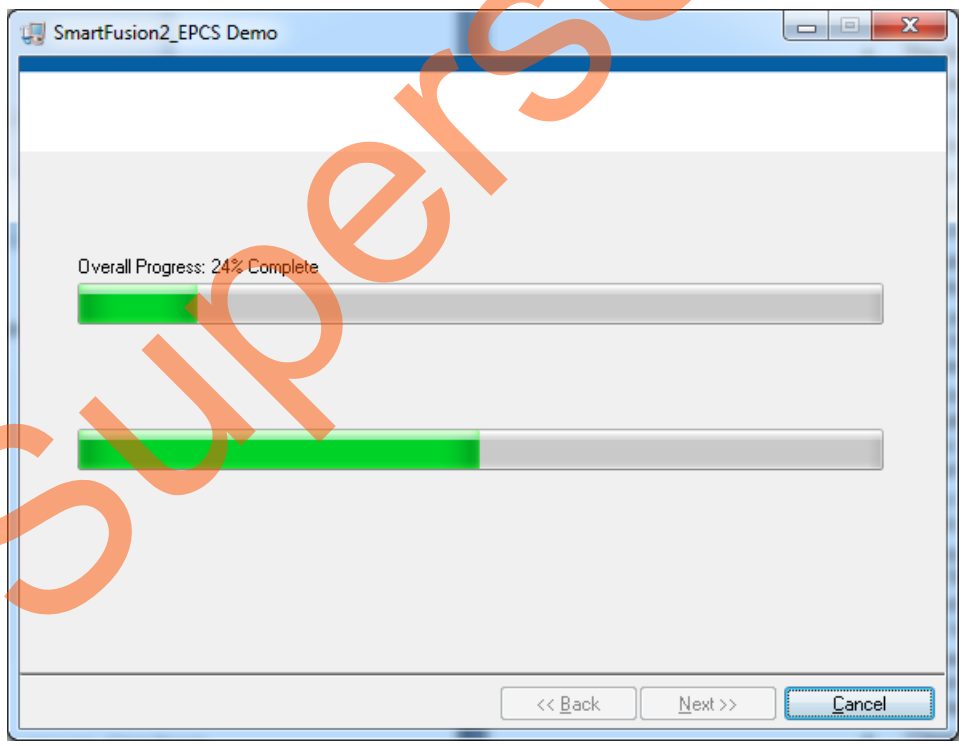


Figure 14 • **GUI Setup Progress Bar**

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

Running the Demo Design

1. Open **Programs**>**SERDES EPCS Demo**.
   The GUI window is displayed as shown in Figure 15.

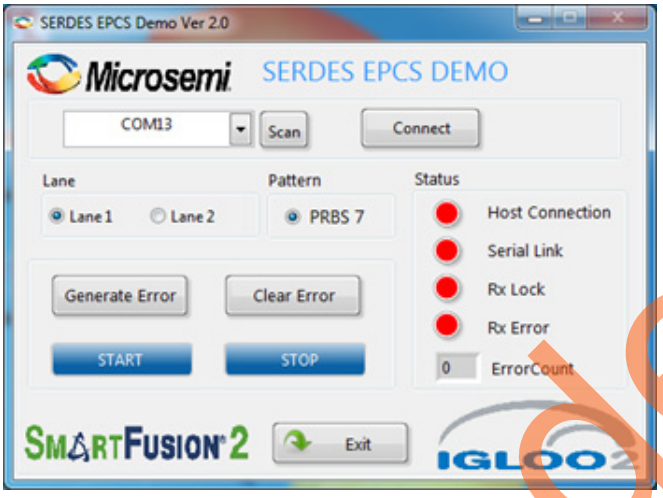


Figure 15 • **EPCS Demo GUI Window**

The drop-down menu for 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.
   Status signals indicate the status of the complete system operation.
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 at any time can be monitored using the status signals in the GUI. For more information on the status signals, refer to "Appendix 4: Status Signals" on page 68.
4. Click **Stop** to stop the EPCS demo.
5. Click **Exit** to exit the GUI.

Figure 16 shows a sample GUI window during an error free operation of the SERDES EPCS demo.

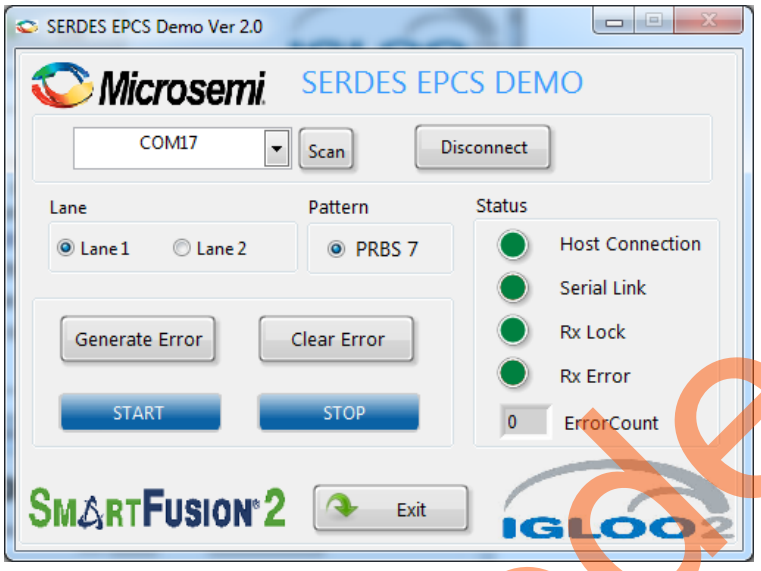


Figure 16 • **Sample GUI Window**

Libero Design Flow

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

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.

Launching Libero SoC

1. Click **Start > Programs > Microsemi Libero SoC v11.4 > Libero SoC v11.4**, or click shortcut on the desktop to open the Libero SoC Project Manager.
2. Select **New** on the **Start Page** tab to create a new project, or select **Project > New Project** from the Libero SoC menu. Figure 17 on page 20 shows the New Project window.

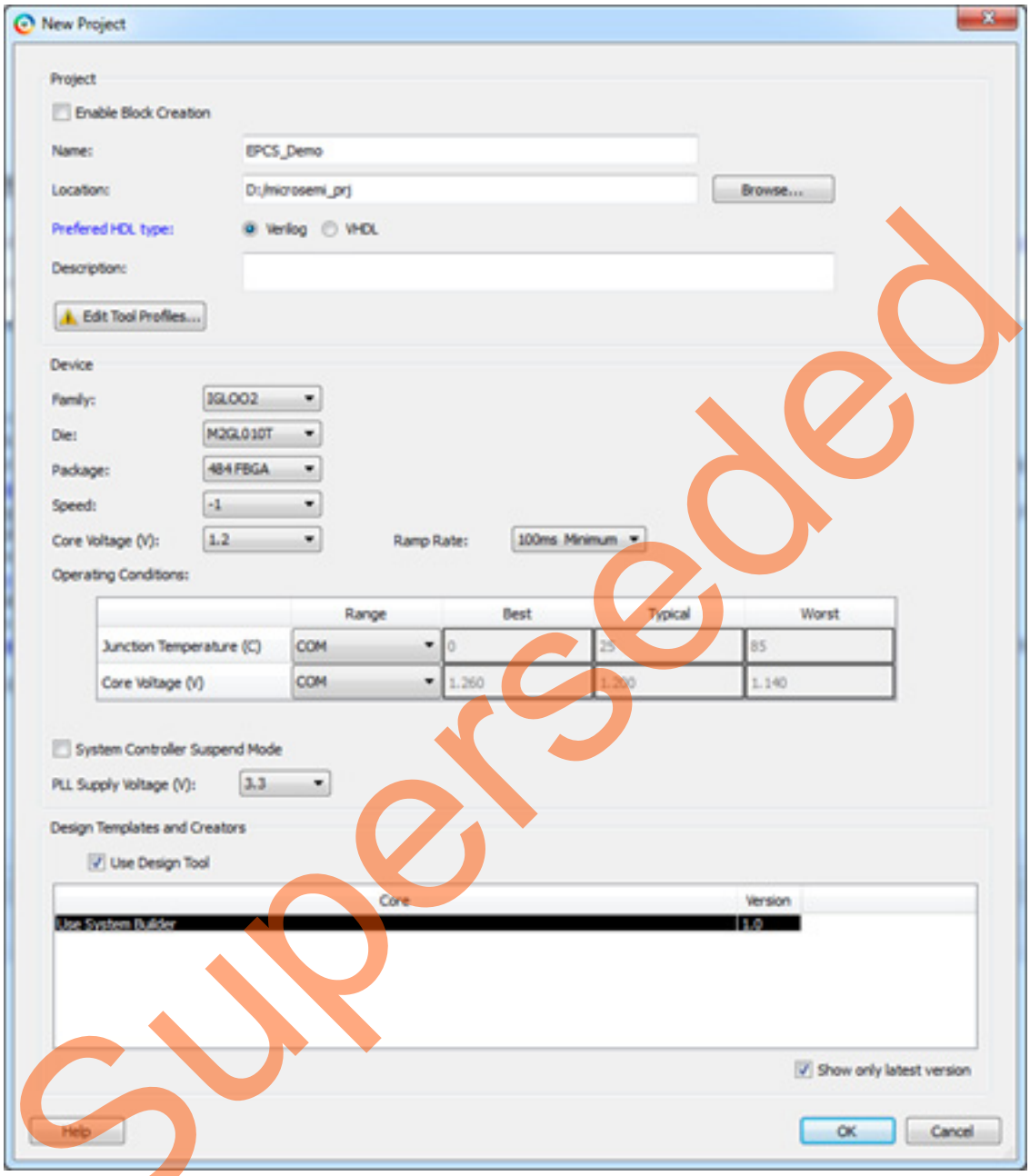


Figure 17 • **IGLOO2 Project Settings**

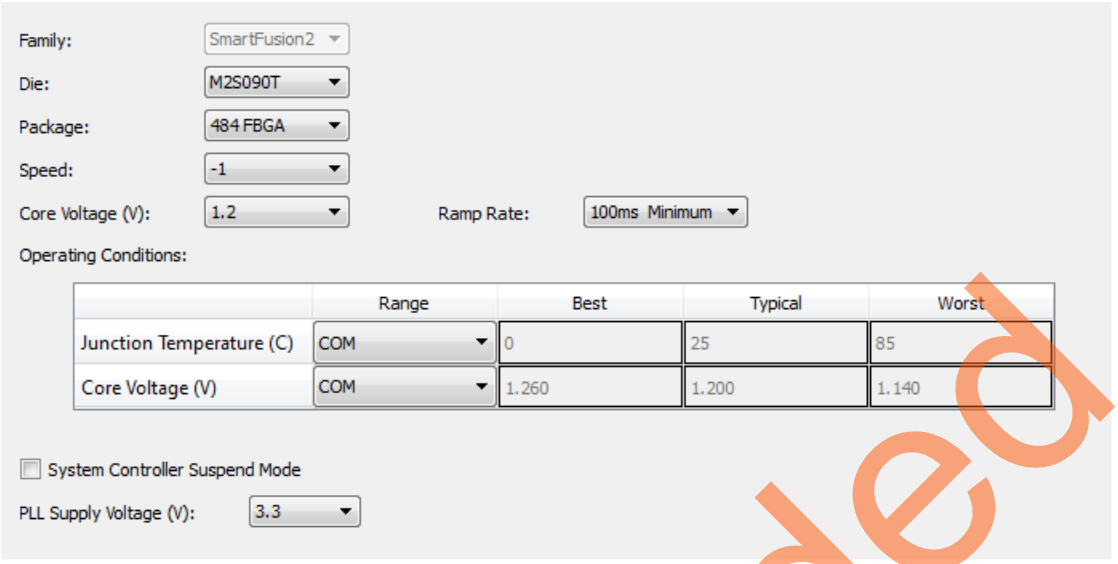


Figure 18 • **SmartFusion2 Project Settings**

3. Enter the following details:
   a. Project:
   – Name: EPCS_Demo
   – Location: Select an appropriate location (for example, *D:/microsemi_prj*)
   – Preferred HDL type: Verilog
   b. Device (select the following values using the drop-down list provided):
   – Family: IGLOO2 or SmartFusion2
   – Die: M2GL010T or M2S090T
   – Package: 484FBGA
   – Speed: -1
   – Core Voltage (V): 1.2
   – Ramp Rate: 100ms Minimum
   c. Operating Conditions: COM
   – PLL Supply Voltage (V): 3.3
4. Click **Edit Tool Profiles** to confirm the tool settings. Figure 19 on page 22 shows the **Tool Profiles** window.
5. Check the below tool settings:
   – Synthesis: Synplify Pro ME I-2013.09M-SP1
   – Simulation: ModelSim 10.3a

– Programming: FlashPro 11.4

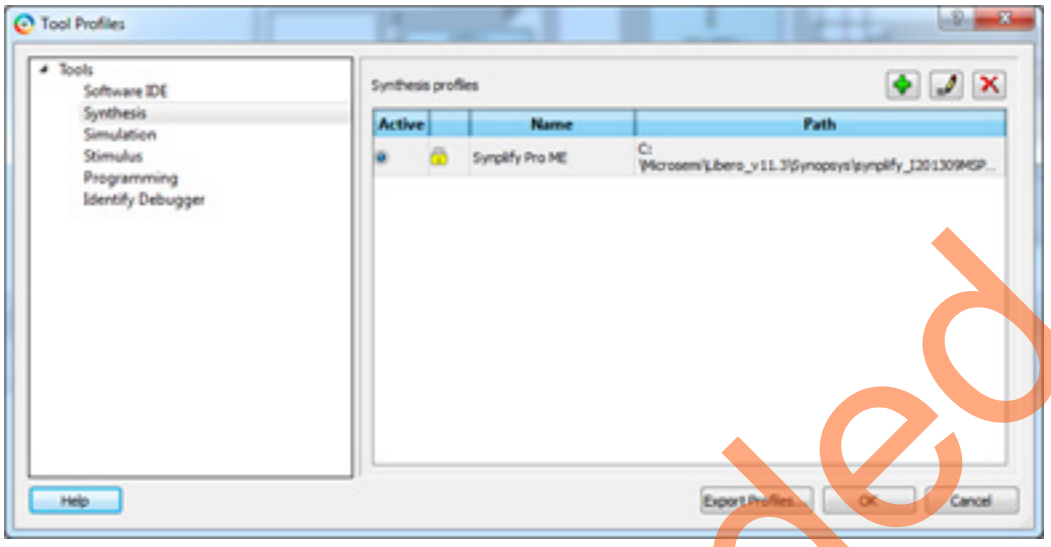


Figure 19 • **Tool Profiles**

6. Click **OK** on the **Tool Profiles** window.
7. Click **OK** on the **New Project** window. The **System Builder** dialog box is displayed as shown in Figure 20.

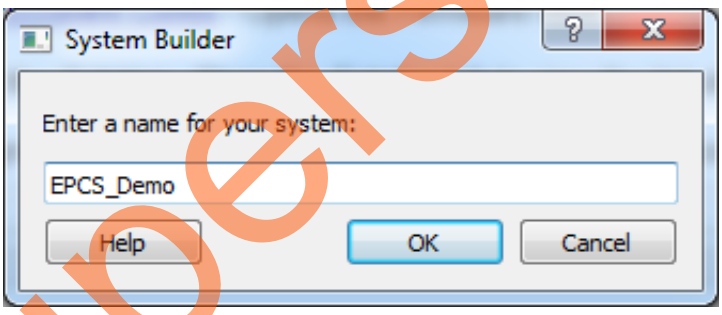


Figure 20 • **System Builder Name**

8. Enter **EPCS_Demo** as the name of the system and click **OK**. The System Builder window is displayed with the **Device Features** page.

9. In the **System Builder - Device Features** page, select **SERDESIF_0** check box under **High Speed Serial Interfaces** as shown in Figure 21.

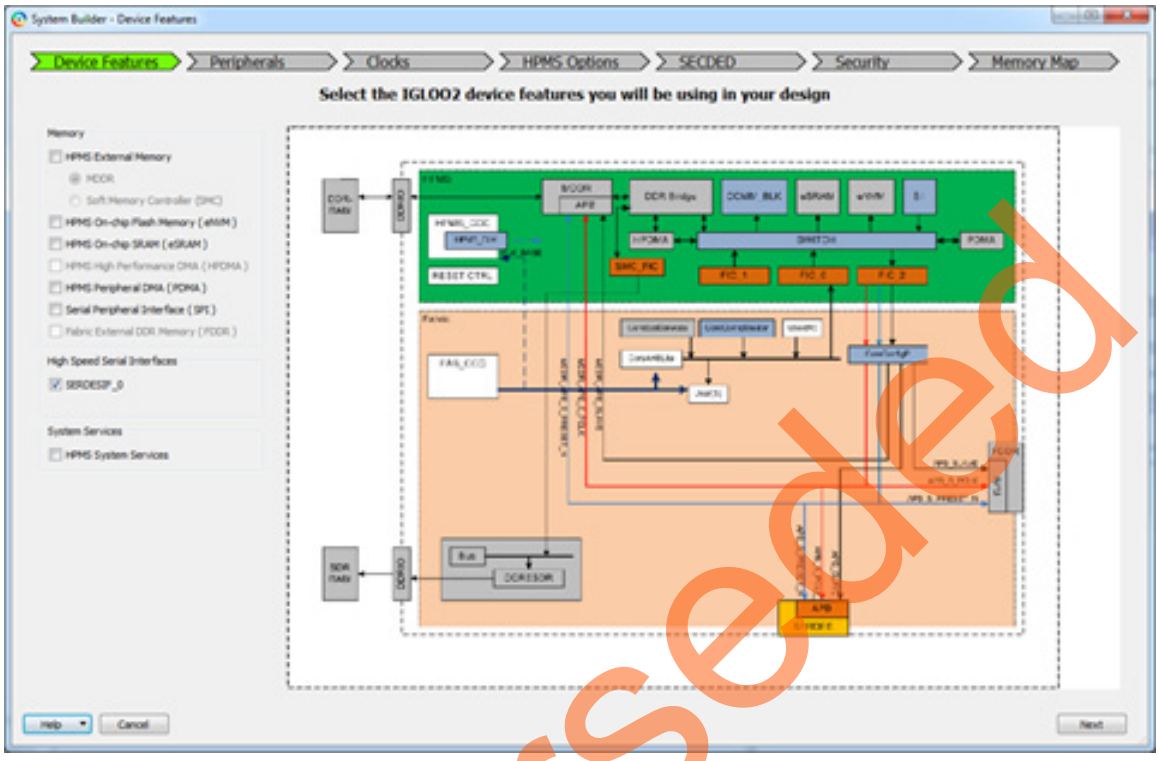


Figure 21 • **System Builder - IGLOO2 Device Features**

10. Select the **MSS On-chip Flash Memory (eNVM)** check box.

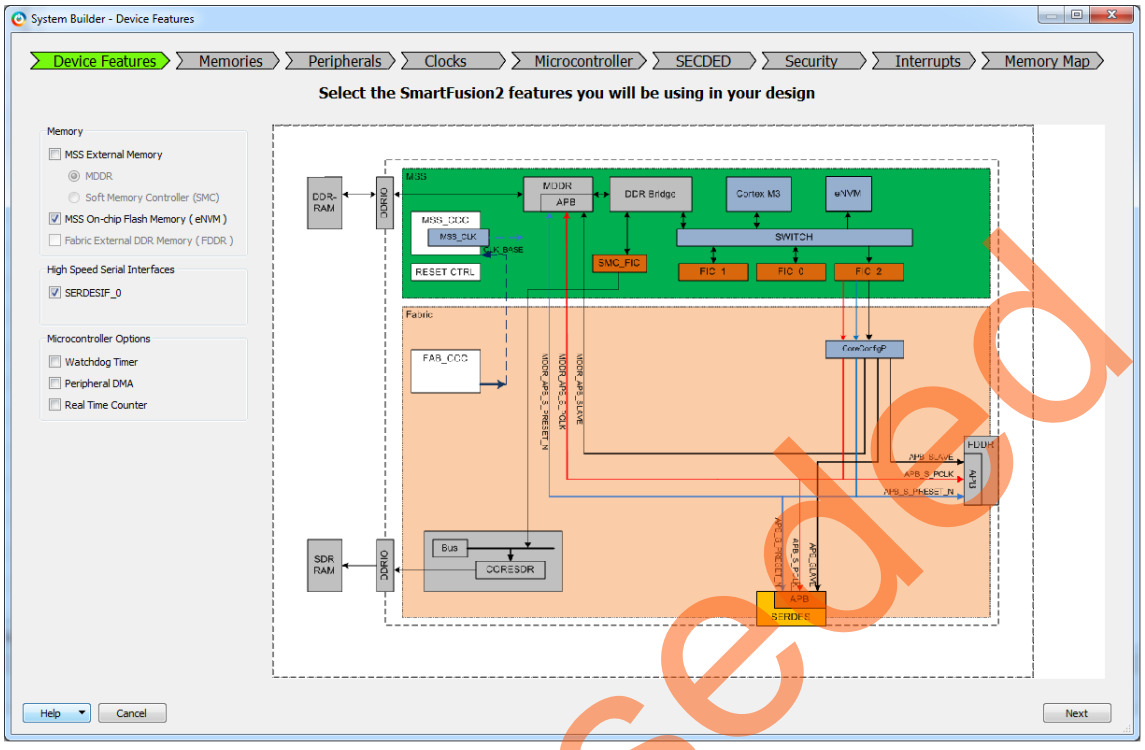


Figure 22 • **System Builder- SmartFusion2 Device Features**

11. Click **Next**. The **System Builder - Peripherals** page is displayed. Keep all the default selections for IGLOO2. For SmartFusion2 use the settings show in Figure 23.

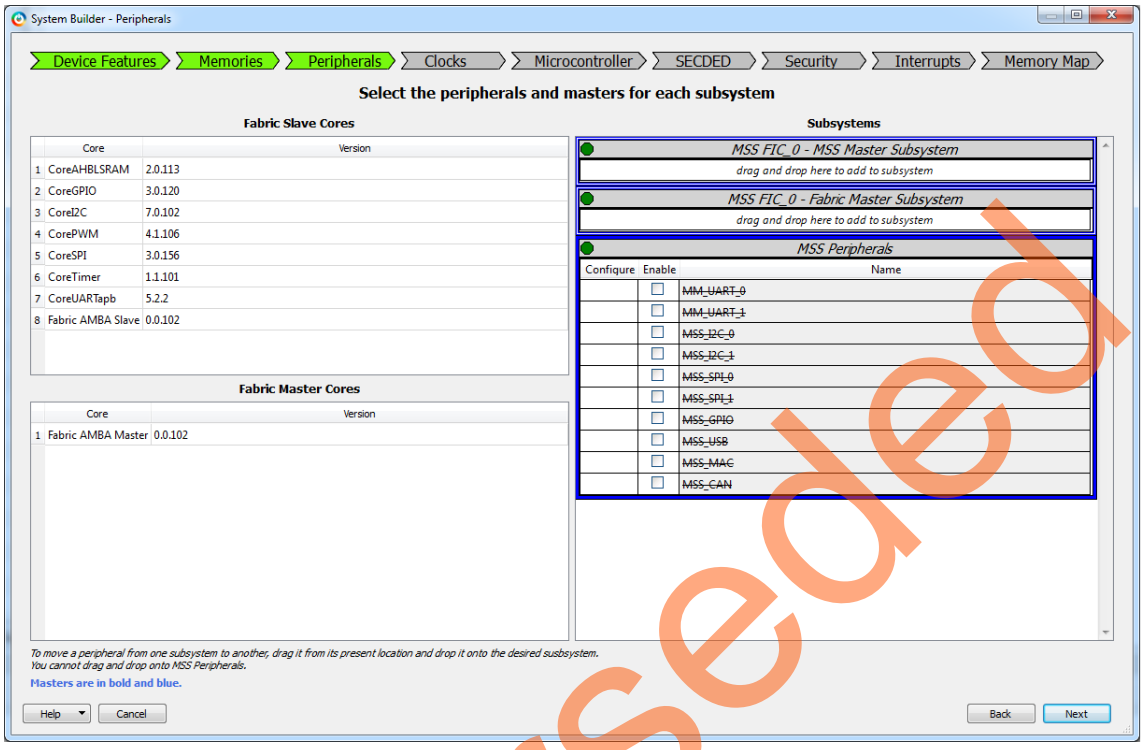


Figure 23 • **SmartFusion2 System Builder Peripherals**

12. Click **Next**. The **System Builder - Clock** page is displayed, as shown in Figure 24 on page 26.

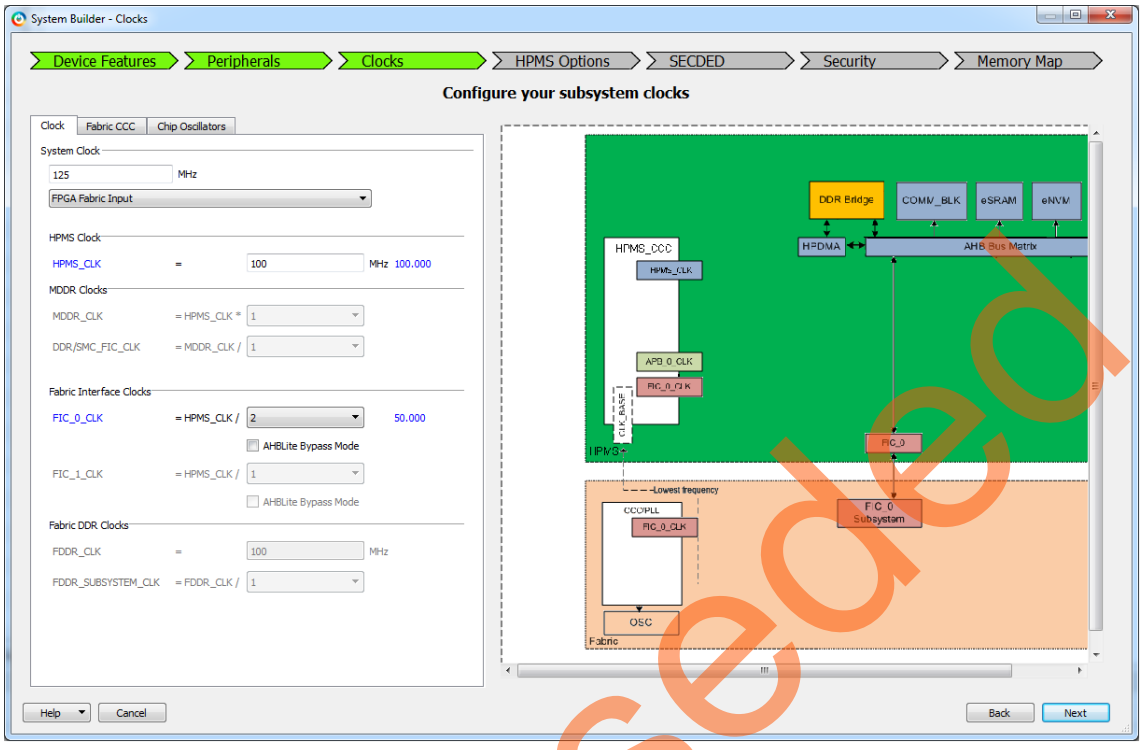


Figure 24 • **System Builder- IGLOO2 Clocks**

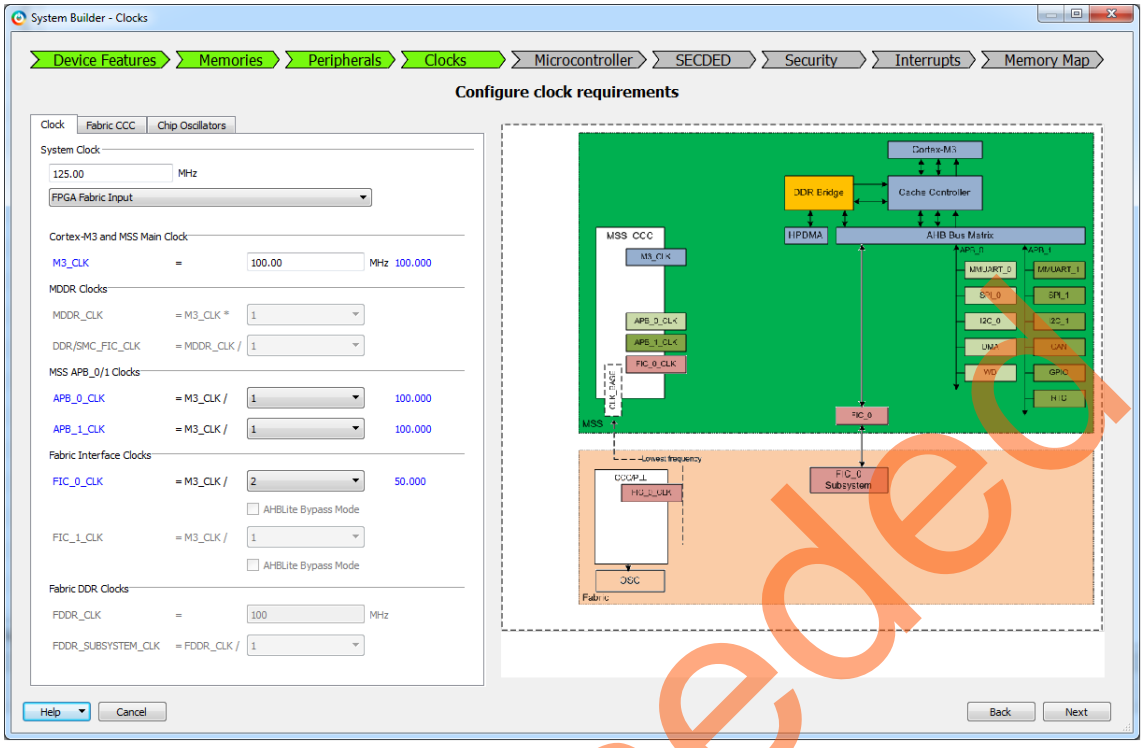


Figure 25 • **System Builder- SmartFusion2 Clocks**

13. Enter the following settings for IGLOO2 and Smart Fusion2:

    a. 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.

b. 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 **FAB_CCC_GL1** check box and set to **50.00** MHz.

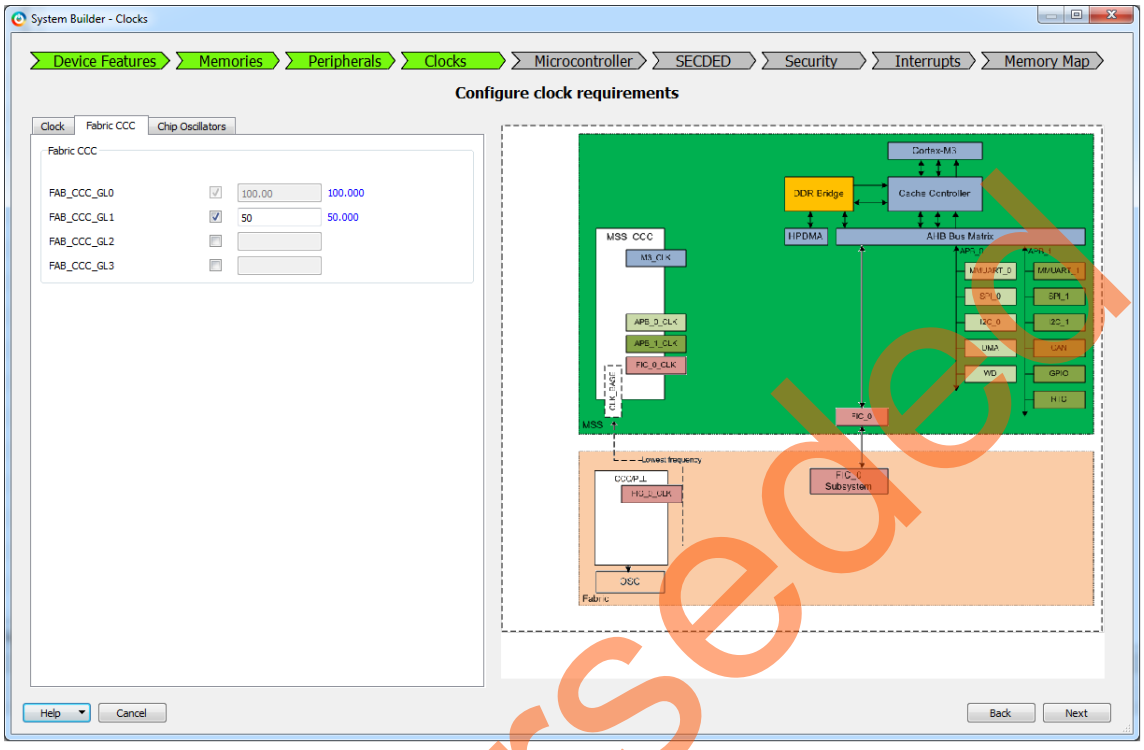


Figure 26 • **SmartFusion2 - Fabric CCC Tab**

14. Click **Next**. The **System Builder - Microcontroller Options** page is displayed. Keep all the default selections. For SmartFusion2, continue through default settings.
15. Click **Next**. The **System Builder - SECDED** page is displayed. Keep all the default selections.
16. Click **Next**. The **System Builder - Security** page is displayed. Keep all the default selections.
17. Click **Next**. The **System Builder - Interrupts** page is displayed. Keep all default selections.
18. Click **Next**. The **System Builder - Memory Map** page is displayed. Keep all the default selections.
19. Click **Finish**.

The **System Builder** generates the 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.

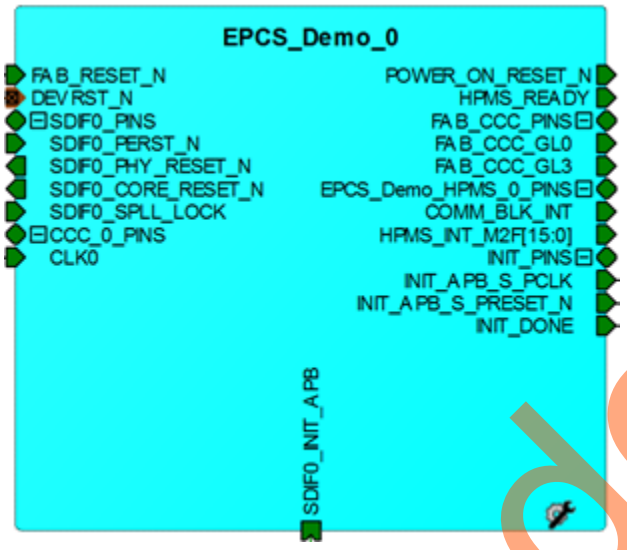


Figure 27 • **System Builder - IGLOO2 Component**

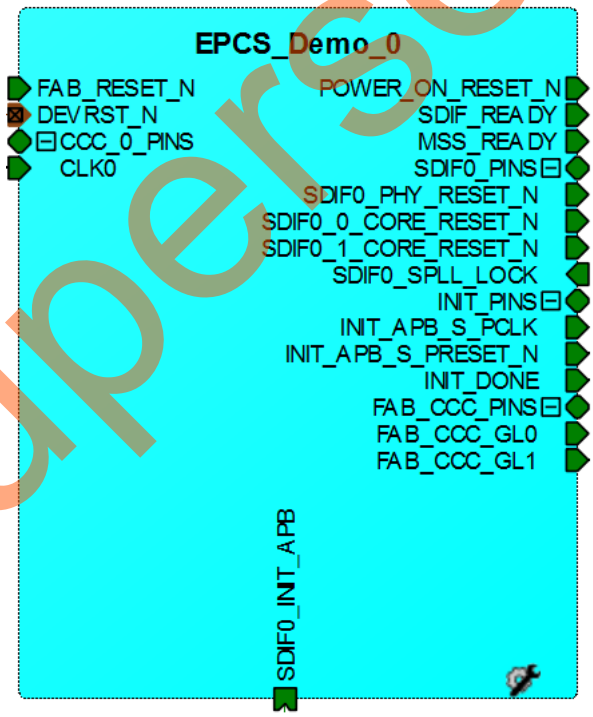


Figure 28 • **System Builder - SmartFusion2 Component**

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 resets and configures 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.

Step 2: Importing User Logic into the Project

The tutorial provides the user logic that needs to be included in the design.

1. To add the user logic to the EPCS demo design, click **File > Import > HDL Source files**.
2. Browse to the `xxxx.v` file location in the design files folder: *<download_folder>/IGLOO2_EPCS_Demo/Source Files*

Figure 29 on page 31 shows the component in the Design Hierarchy window.

Select the following HDL source files:

- count_check.v
- count_gen.v
- delay_line.v
- epcs_rx_intf.v
- epcs_tx_intf.v
- FabUART.v
- prbs_asic.chk.v
- prbs_asic_gen.v
- PRBS_Check.v
- PRBS_Gen.v
- Output_Select.v

The selected HDL modules are used throughout the design.

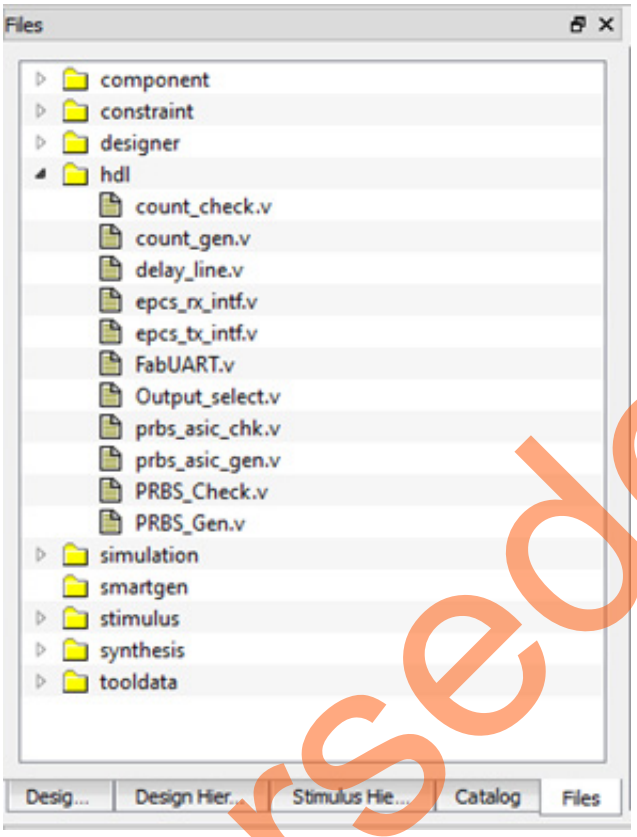


Figure 29 • **Imported User HDL Source Files**

Step 3: Instantiating Libero SoC Catalog Components in SmartDesign

The Libero SoC catalog provides IP cores that can be easily dragged and dropped into the SmartDesign canvas.

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

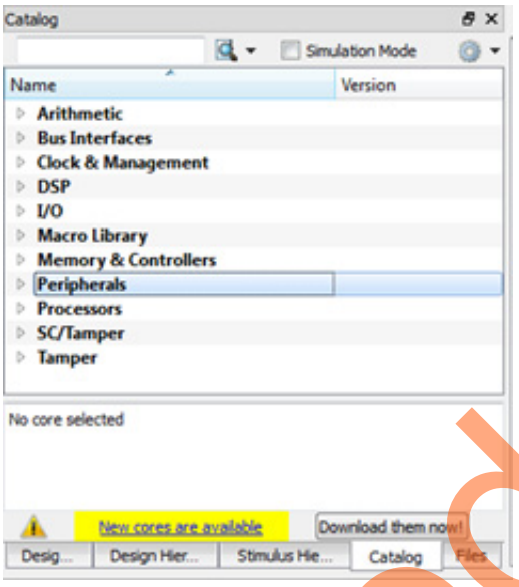


Figure 30 • **Catalog View**

2. After dragging the **COREUART** onto the canvas, double-click the **COREUART_0** module and set the parameters, as shown in Figure 31.

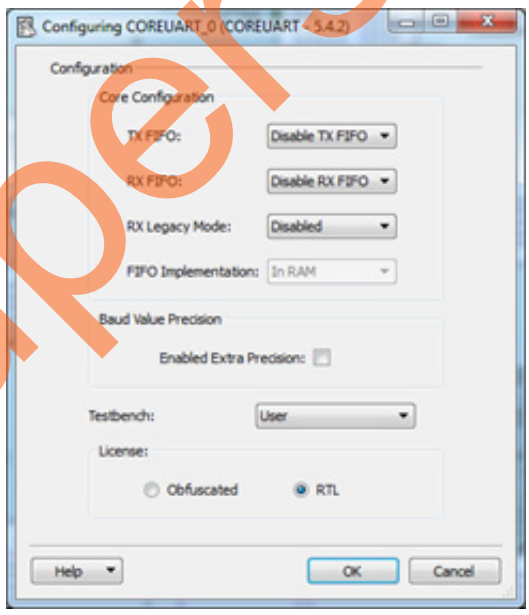


Figure 31 • **COREUART Settings**

3. Click **OK**.

4. In the **Peripherals** tab, select the **CorePCS** module and drag-and-drop onto the `EPCS_Demo_top` SmartDesign canvas. Double-click the **CorePCS_0** module and set the parameters, as shown in Figure 32.

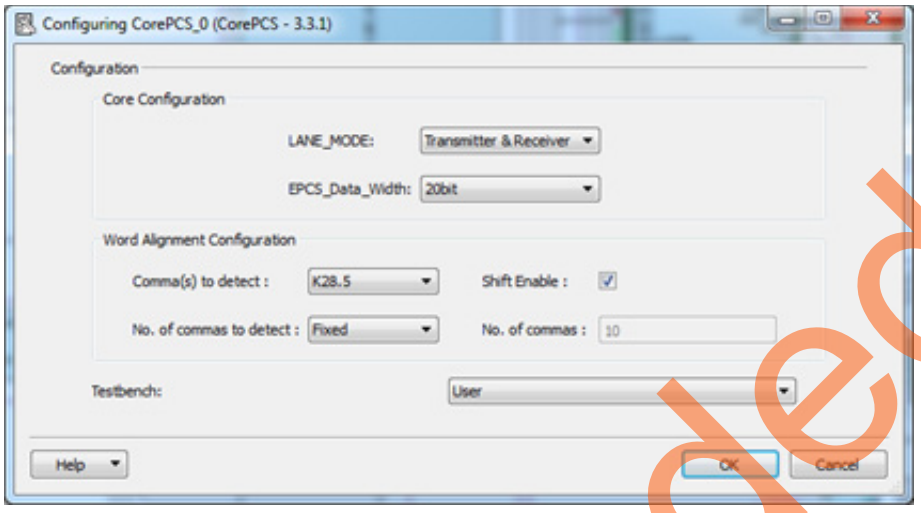


Figure 32 • **CorePCS Configurator Settings**

5. Click **OK**.
6. In the **Macro Library** tab, select the **AND2** module. Drag-and-drop two instances of the component onto the `EPCS_Demo_top` canvas.

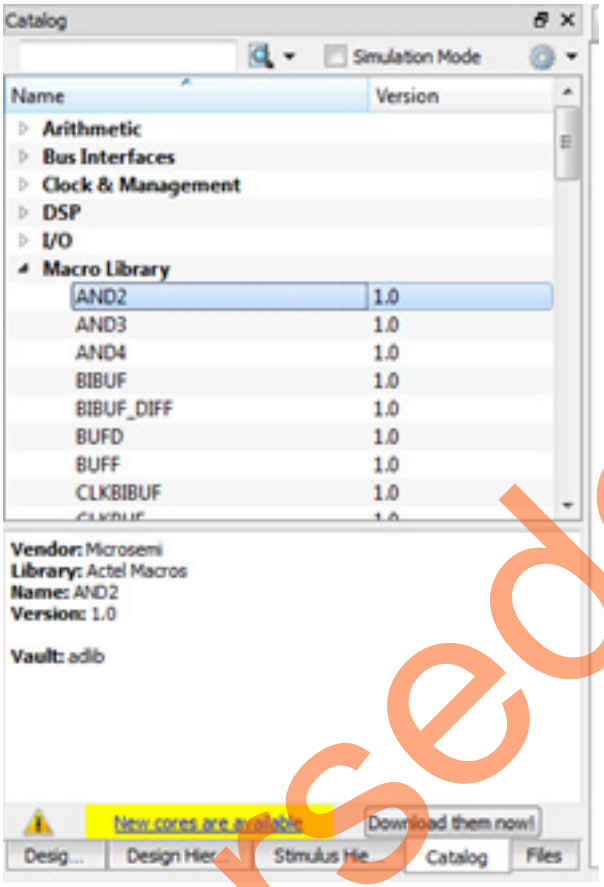


Figure 33 • **Macro Library Component**

Step 4: Creating SmartDesign Hierarchy and Adding a SERDESIF Component

1. Go to the **Design Flow** tab and select **Create SmartDesign**.

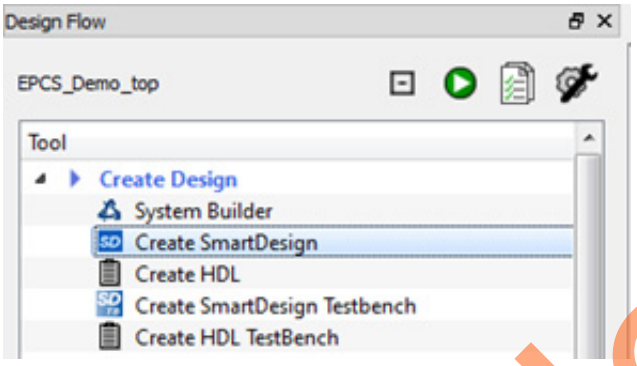


Figure 34 • **Creating SmartDesign**

2. In the **Create New SmartDesign** dialog box, enter a Name for the SmartDesign module (EPCS_SERDES).

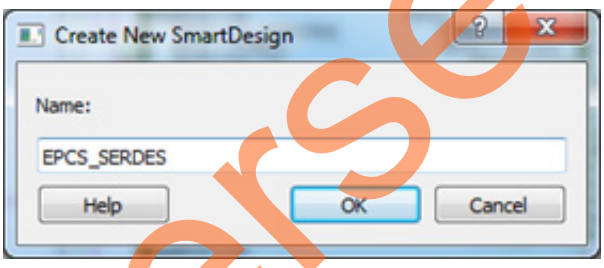


Figure 35 • **New SmartDesign Setup**

3. Go to the **Catalog** tab, select the **Peripherals** category, and drag-and-drop 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 the **SERDES_IF_0** component in the SmartDesign canvas to open the **SERDES** configurator. Configure the SERDES, refer Figure 36.

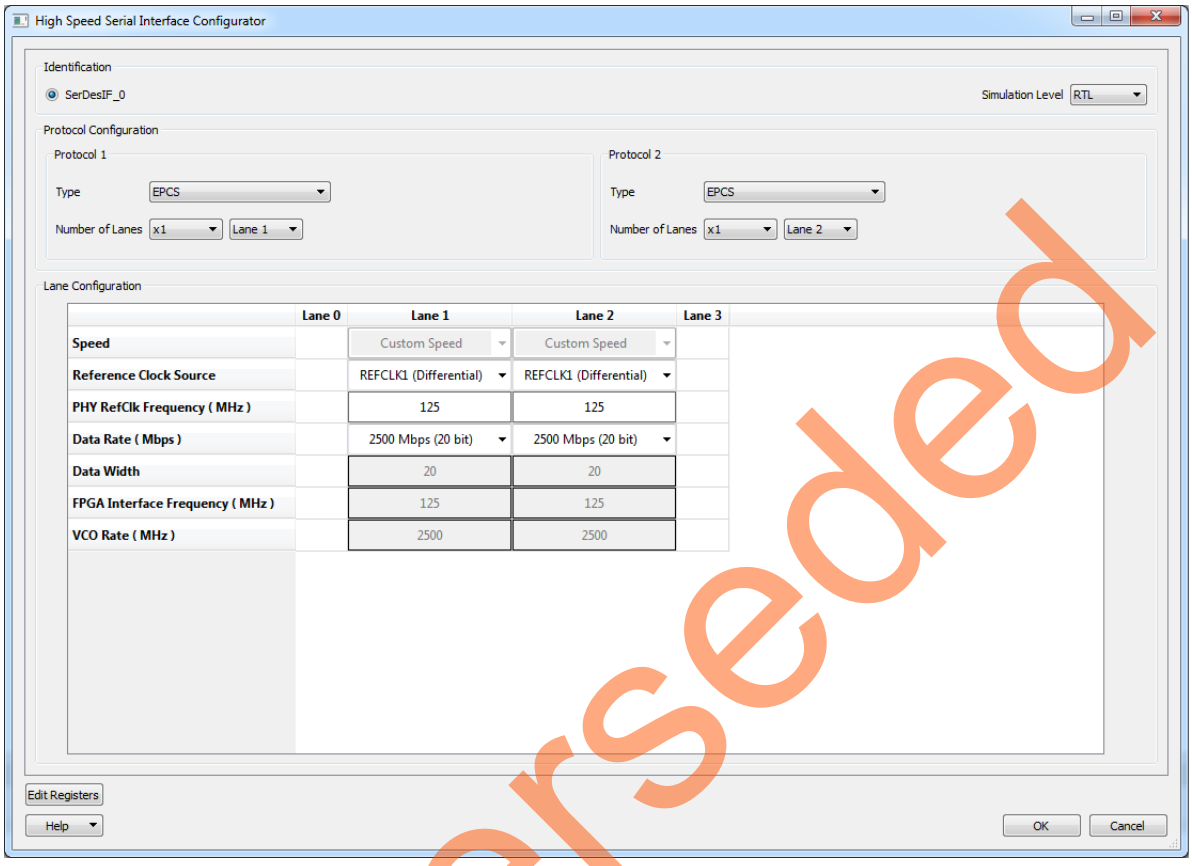


Figure 36 • **SERDESIF Configurator Settings**

– Identification
  - SerDesIF_0
  - Simulation Level: RTL
– Protocol Configuration
  - Protocol1: Type: EPCS
    Number of Lanes: x1, Lane 1
  - Protocol2: EPCS
    Number of Lanes: x1, Lane 2
– Lane Configuration
  - Speed: Lane[1:2] CUSTOM SPEED
  - Reference Clock Source: REFCLK1 (Differential)
  - PHY RefClk Frequency (MHz): 125
  - Data Rate (Mbps): 2500 Mbps (20 bit) for all Lanes

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

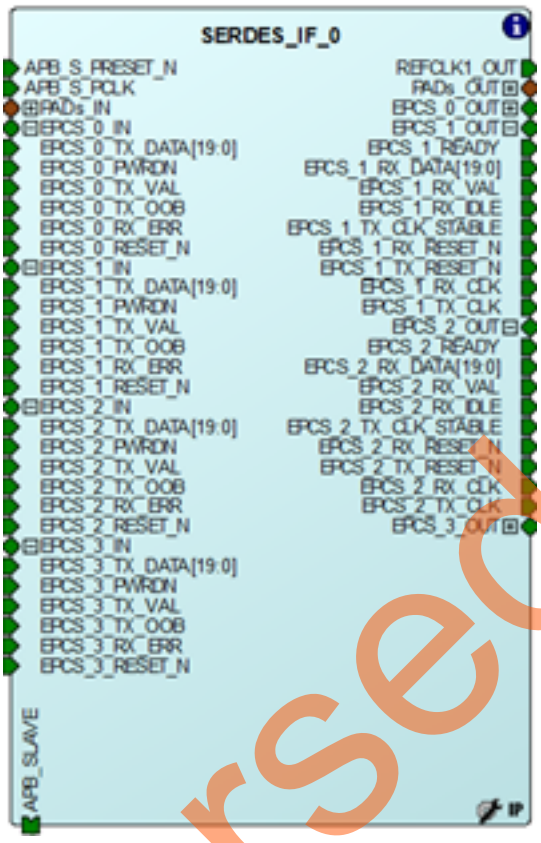


Figure 37 • **SERDESIF Component**

6. Select and drag-and-drop the appropriate imported HDL modules listed under the **work** subdirectory to add to the EPCS_SERDES canvas from the **Design Hierarchy** tab.
   - 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.

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-and-drop onto the pin that you want to connect with.
2. In the second method, hold 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:l_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]

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

Table 4 lists the pins that must be promoted to top level (as instructed above) and 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

Table 4 • **EPCS_SERDES Connection Renames (continued)**

Module	Pin Name	Renamed
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

Connect the following pins to the top-level EPCS_RESET_N pin:

- EPCS_1_RESET_N
- EPCS_2_RESET_N

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

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

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

Click **Generate Component** to generate the EPCS_SERDES component.

Figure 38 • **EPCS_SERDES Component Generation**

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.

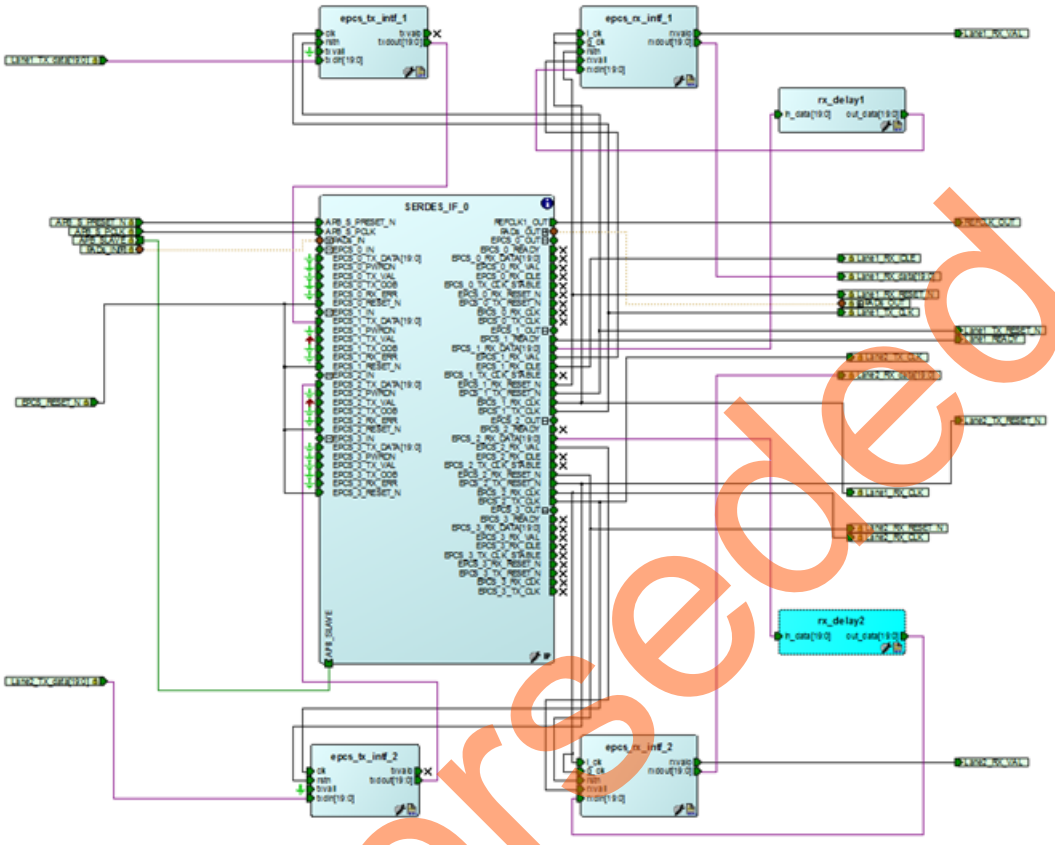


Figure 39 • **Complete EPCS_SERDES SmartDesign Canvas**

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

Step 5: Finalizing SmartDesign

Open the `EPCS_Demo_top` SmartDesign canvas, drag-and-drop 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.

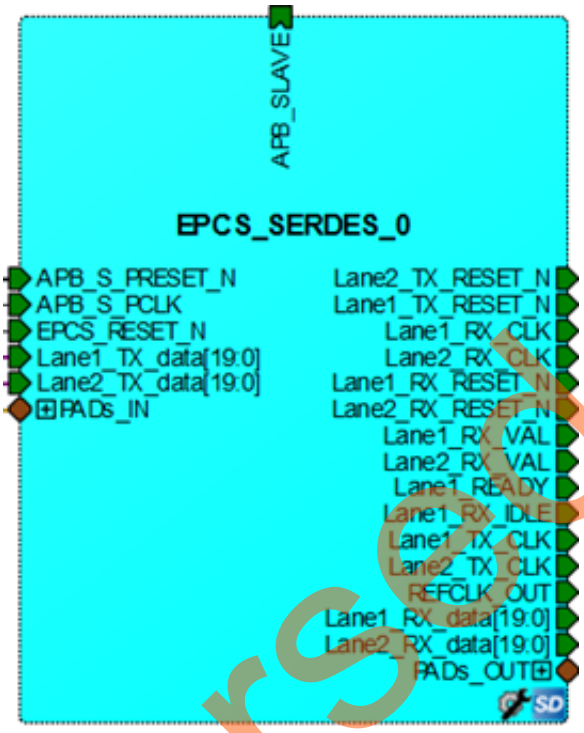


Figure 40 • **EPCS_SERDES Component**

To add the user HDL modules to the EPCS_Demo_top SmartDesign canvas, go to the **Design Hierarchy** tab, drag-and-drop the appropriate imported HDL modules listed under the **work** subdirectory.

- 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

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

From	To
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
FabUART_0:rx_error	Output_select_0:Rx_error
FabUART_0:error_count[5:0]	Output_select_0:Error_out[5:0]

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

From	To
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]

Right-click the **[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

Hold the CTRL key, select the following **CorePCS_0** ports, right-click, and select **Tie Low**.

- FORCE_DISP[1:0]
- DSP_SEL[1:0]

Hold the CTRL key, select the following **COREUART_0** ports, right-click, and select **Tie Low**.

- CSN
- ODD_N_EVEN
- PARITY_EN

Hold the CTRL key, select the following **CorePCS_0** ports, right-click, and select **Tie High**.

- WA_RSTn

Hold the CTRL key, select the following **EPCS_Demo_0** ports, right-click, and select **Tie High**.

- SDIF0_PERST_N
- SDIF0_SPLL_LOCK

Hold the CTRL key, select the following **COREUART_0** ports, right-click, and select **Tie High**.

- BIT8

Hold the CTRL key, select the following **COREUART_0** ports, right-click, and select **Mark Unused**.

- OVERFLOW
- PARITY_ERR

- FRAMING_ERR

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]

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

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]

Click **Generate Component** to generate EPCS_Demo_top component.

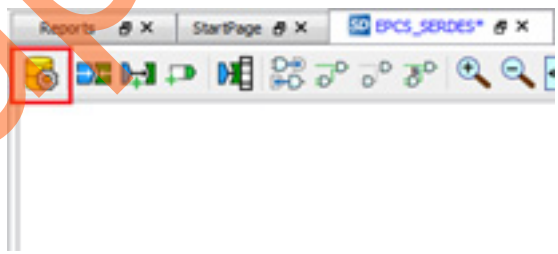


Figure 41 • **EPCS_Demo_top Component Generation**

Figure 42 shows the Complete EPCS_Demo_top SmartDesign Canvas.

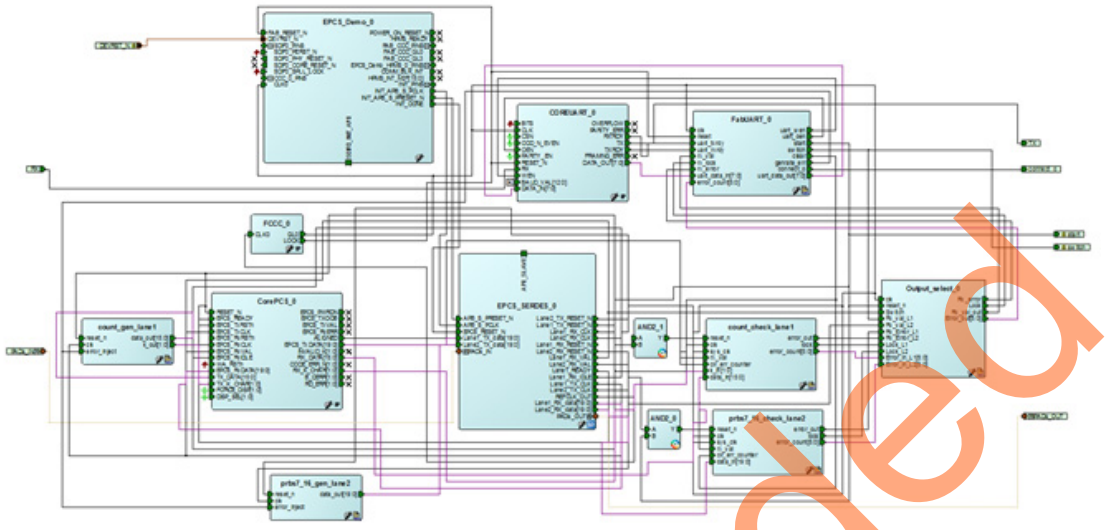


Figure 42 • **Complete EPCS_Demo_top SmartDesign Canvas**

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=IGLOO2_EPCS_Demo_11p4_sp1_DF

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

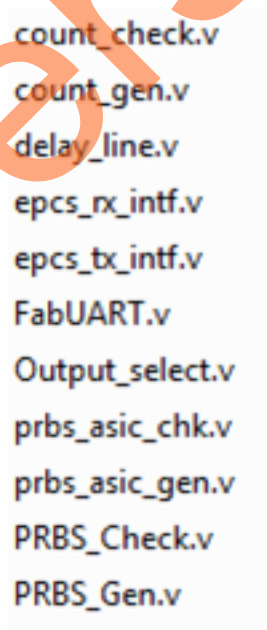


Figure 43 • **hdl Directory**

2. The compile and place-and-route constraints are located in the constraint or `EPCS_Demo_top_compile.sdc` file.

```
##############################################################################
#
```

```
# All SERDESIF Tx and Rx clocks are constrained for 125MHz (2.5Gbps)
#
# Clocks for Lanes 1 and 2
create_clock -period 8.000 -name {EPCS_2_TX_CLK}\
  [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK[0]}]
create_clock -period 8.000 -name {EPCS_2_RX_CLK}\
  [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK[0]}]
create_clock -period 8.000 -name {EPCS_1_TX_CLK}\
  [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_TXCLK_1}]
create_clock -period 8.000 -name {EPCS_1_RX_CLK}\
  [get_pins {EPCS_SERDES_0/SERDES_IF_0/SERDESIF_INST:EPCS_RXCLK_1}]
#
##############################################################################
##

##############################################################################
##
# Constraints for the remainder control plane clocks in the design
#
create_clock -period 20.000 -name {SDIF0_CLK}\
  [get_pins {EPCS_Demo_0/EPCS_Demo_HPMS_0/MSS_ADLIB_INST:CLK_CONFIG_APB}]
create_clock -period 20.000 -name {HPMS_CLK_OUT}\
  [get_pins {EPCS_Demo_0/CCC_0/GL0_INST:Y}]
#
##############################################################################
##
# False path on clock domain crossing from data plane to control plane clocks.
#
set_false_path -from {HPMS_CLK_OUT} -to {EPCS_1_TX_CLK
                                         EPCS_2_TX_CLK\
                                         EPCS_1_RX_CLK\
                                         EPCS_2_RX_CLK}
set_false_path -from {EPCS_1_TX_CLK} -to {HPMS_CLK_OUT}
set_false_path -from {EPCS_2_TX_CLK} -to {HPMS_CLK_OUT}
set_false_path -from {EPCS_1_RX_CLK} -to {HPMS_CLK_OUT}
set_false_path -from {EPCS_2_RX_CLK} -to {HPMS_CLK_OUT}

set_false_path -from {EPCS_1_TX_CLK} -to {EPCS_1_RX_CLK}
#
##############################################################################
###
```

3. The I/O constraints are located in `constraint/io/EPCS_IGL2_EB.pdc` file. This file includes the pin locations required to run this demo on the M2S_M2GL Evaluation Kit.

```
# Microsemi I/O Physical Design Constraints file
# Auto Generated User I/O Constraints file
# Version: v11.2 11.2.0.26
# Family: IGLOO2 , Die: M2GL010T , Package: 484 FBGA
# Date generated: Fri Jan 24 18:53:30 2014
#
# User Locked I/O Bank Settings
#
#
# Unlocked I/O Bank Settings
# The I/O Bank Settings can be locked by directly editing this file
# or by making changes in the I/O Attribute Editor
#
#
# User Locked I/O settings
#
#set_io CLK0 \
# -pinname K1 \
# -fixed yes \
# -DIRECTION INPUT
set_io RX \
-pinname G18 \
```

```
-fixed yes \
-DIRECTION INPUT
set_io TX \
-pinname H19 \
-fixed yes \
-DIRECTION OUTPUT
set_io connect_o \
-pinname H5 \
-fixed yes \
-DIRECTION OUTPUT
set_io start \
-pinname H6 \
-fixed yes \
-DIRECTION OUTPUT
set_io switch \
-pinname AB15 \
-fixed yes \
-DIRECTION OUTPUT
#
# Dedicated Peripheral I/O Settings
#
#
# Unlocked I/O settings
# The I/Os in this section are unplaced or placed but are not locked
# the other listed attributes have been applied
```

After the pins are assigned, the I/O Editor can be invoked to review the assignments.

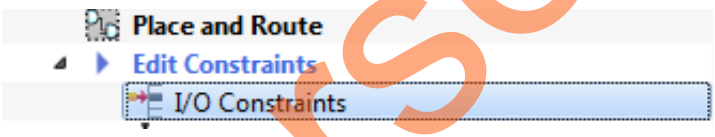
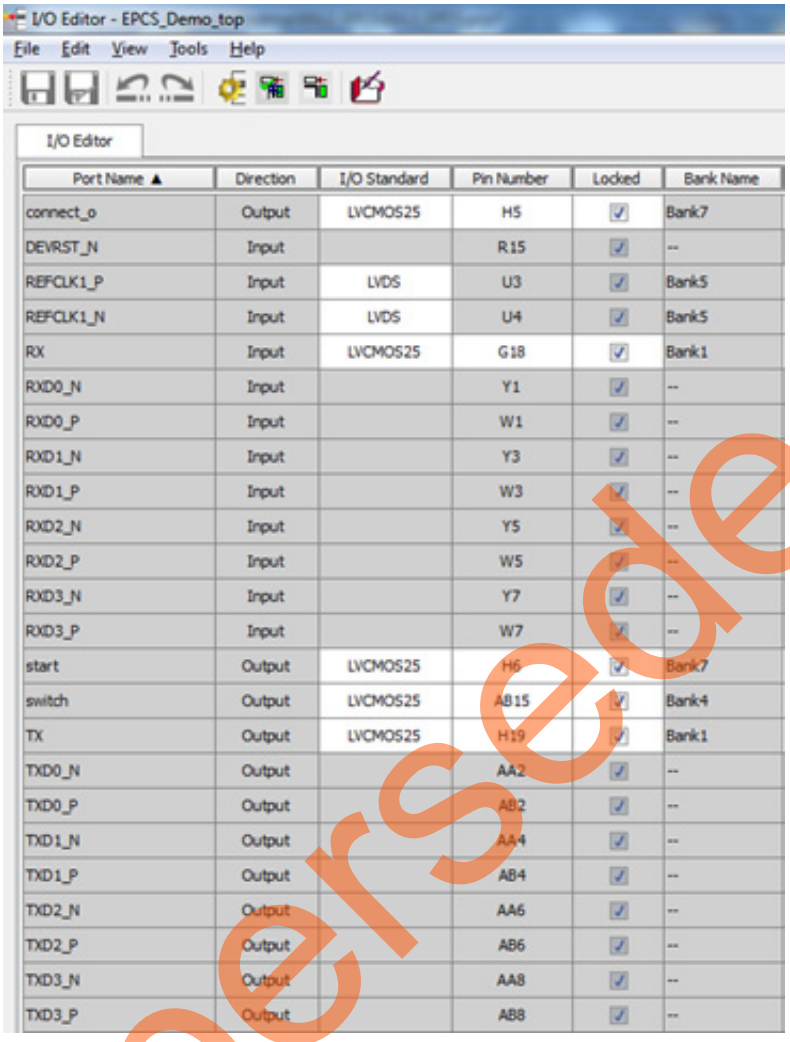


Figure 44 • **Invoke I/O Editor**



Port Name ▲	Direction	I/O Standard	Pin Number	Locked	Bank Name
connect_o	Output	LVCMOS25	H5	☑	Bank7
DEVRST_N	Input		R15	☑	--
REFCLK1_P	Input	LVDS	U3	☑	Bank5
REFCLK1_N	Input	LVDS	U4	☑	Bank5
RX	Input	LVCMOS25	G18	☑	Bank1
RXD0_N	Input		Y1	☑	--
RXD0_P	Input		W1	☑	--
RXD1_N	Input		Y3	☑	--
RXD1_P	Input		W3	☑	--
RXD2_N	Input		Y5	☑	--
RXD2_P	Input		W5	☑	--
RXD3_N	Input		Y7	☑	--
RXD3_P	Input		W7	☑	--
start	Output	LVCMOS25	H6	☑	Bank7
switch	Output	LVCMOS25	AB15	☑	Bank4
TX	Output	LVCMOS25	H19	☑	Bank1
TXD0_N	Output		AA2	☑	--
TXD0_P	Output		AB2	☑	--
TXD1_N	Output		AA4	☑	--
TXD1_P	Output		AB4	☑	--
TXD2_N	Output		AA6	☑	--
TXD2_P	Output		AB6	☑	--
TXD3_N	Output		AA8	☑	--
TXD3_P	Output		AB8	☑	--

Figure 45 • **I/O Constraints Editor**

Step 6: Generate Program Data

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

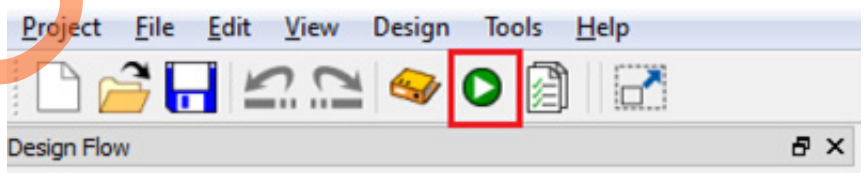


Figure 46 • **Generate Programming Data**

Export Programming File

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

{LiberoProjectDirectory}\designer\EPCS_Demo_top\export

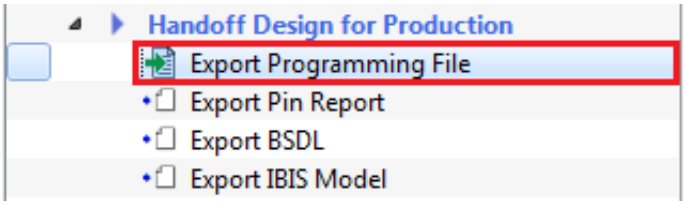


Figure 47 • **Exporting Programming File**

Step 7: Creating ENVM Client Using SmartFusion2

The following sections describe how to create the ARM® Cortex®-M3 firmware used to initialize the MSS and SERDESIF. The eNVM client must be uploaded with the firmware application to initialize the SERDESIF through **CoreConfigP**. The Cortex-M3 processor executes the code in eNVM after the SmartFusion2 device is reset. In this design, the eNVM client is created with the firmware application code to initialize the SERDESIF.

1. To build the firmware eNVM client, invoke the standalone SoftConsole IDE. The **SoftConsoleIDE Project Workspace** window is displayed as shown in Figure 48.

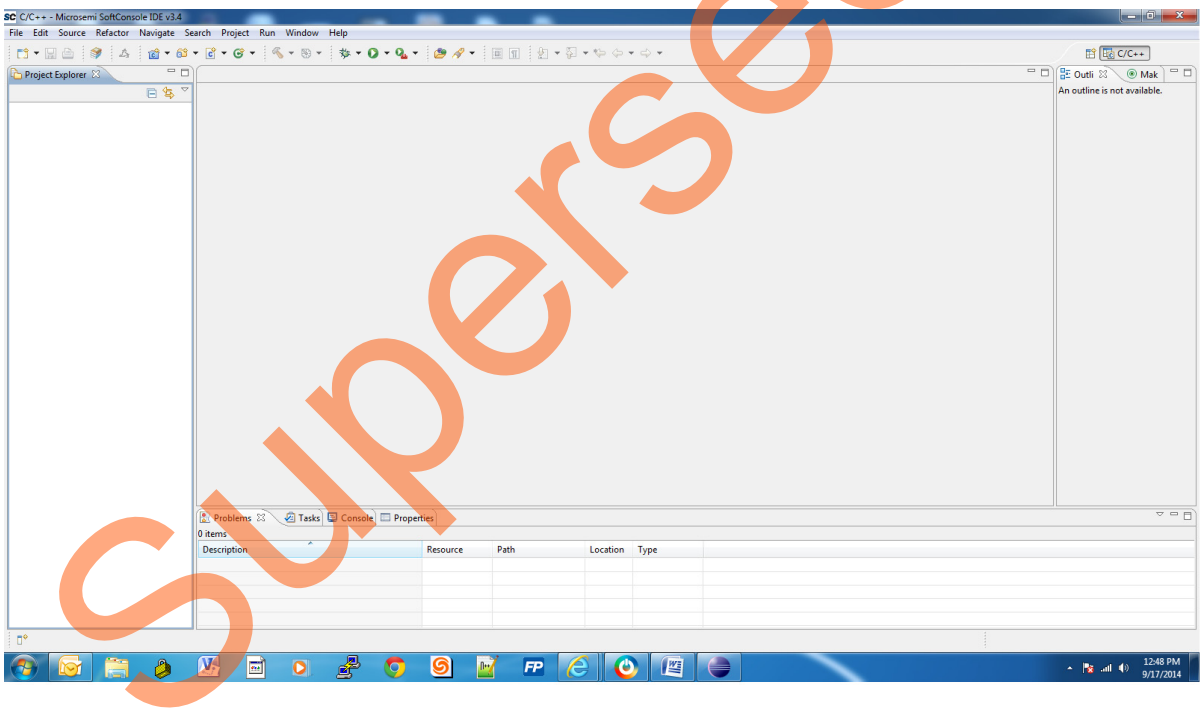


Figure 48 • **SoftConsole IDE Project Workspace**

2. Libero builds an initial SoftConsole workspace to begin the building of an eNVM client. Start by selecting **Export Firmware** from Libero Manager.

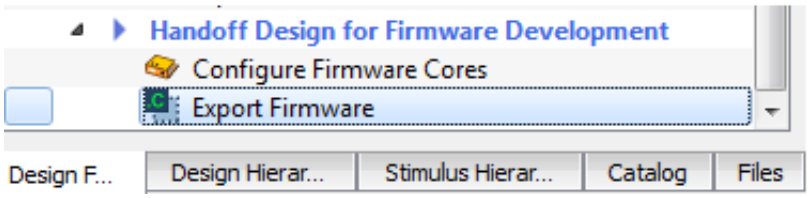


Figure 49 • **Export Firmware**

3. Libero prompts you to build a SoftConsole workspace. Select the location to save the project, select **Create project** check box, and click **OK** to create workspace.

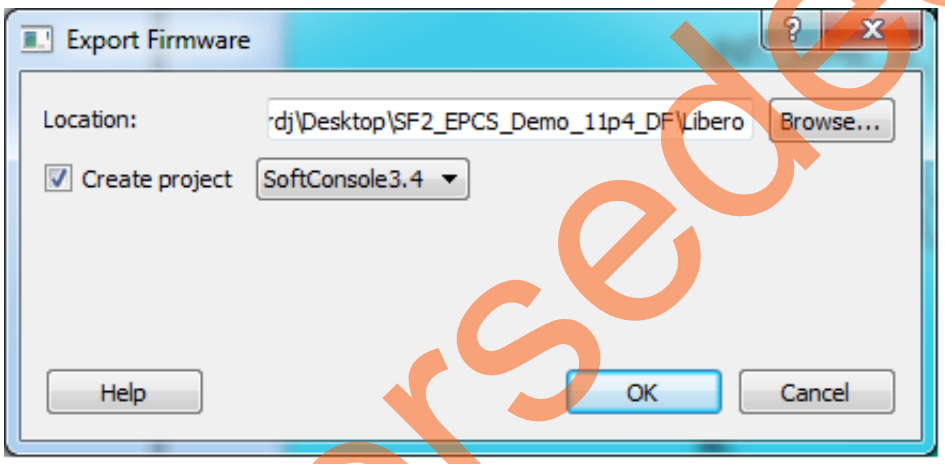


Figure 50 • **Create New SoftConsole Workspace**

4. Import the existing project into workspace as shown in Figure 51.

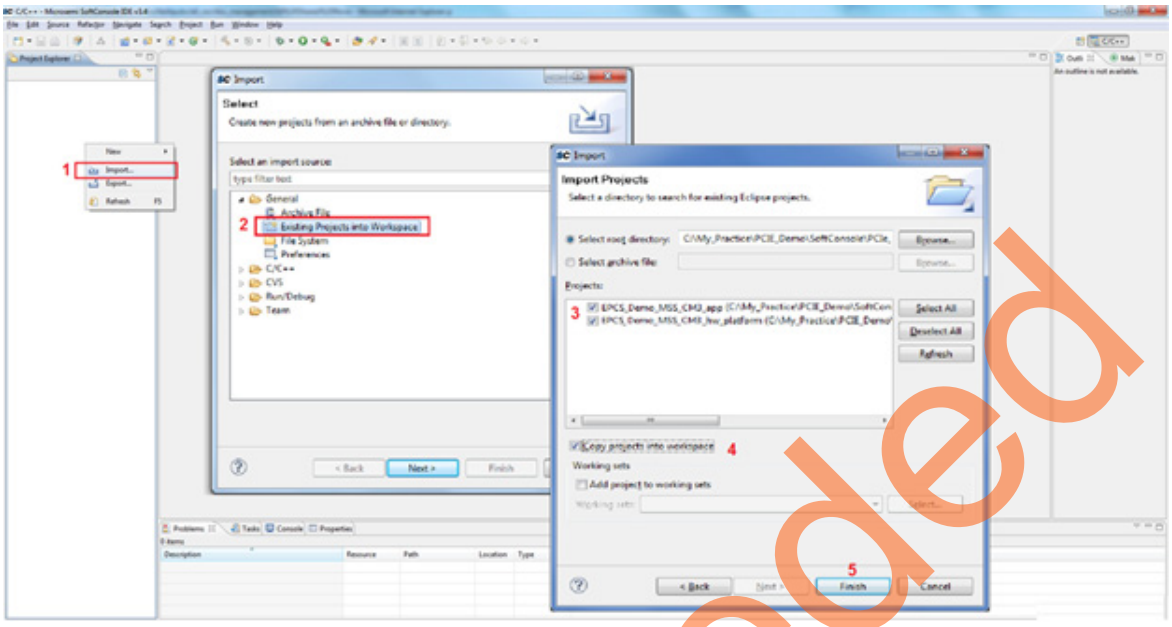


Figure 51 • **Import the Existing Project into Workspace**

a. Right-click **Project Explorer** tab on the left pane and select **Import…**. The **Import** dialog box is displayed.

b. Select **Existing Project into Workspace** under **General** and click **Next**. The **Import Projects** dialog box is displayed.

c. Click **Browse** to navigate to the SoftConsole project folder.

d. Select **EPCS_Demo_MSS_CM3_app** and **EPCS_Demo_MSS_CM3_hw_platform** check boxes under **Projects**.

e. Select the **Copy projects into workspace** check box.

f. Click **Finish**. The **SoftConsole Workspace** window is displayed as shown in Figure 52 on page 52.

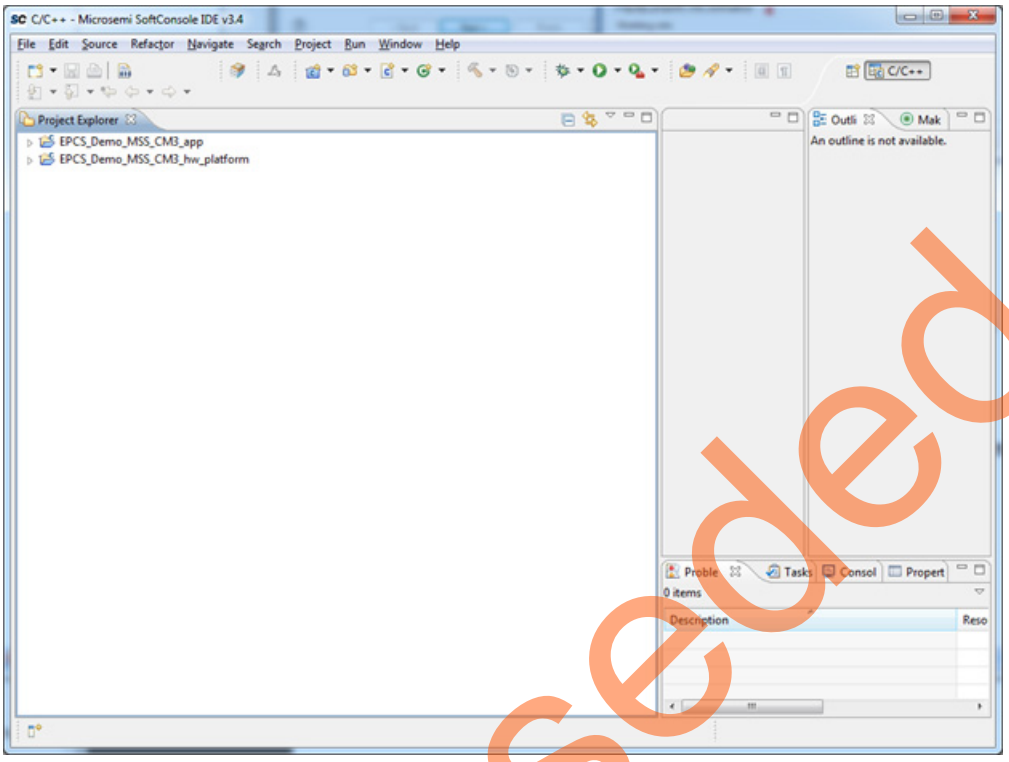


Figure 52 • **SoftConsole Workspace**

5. Hold the **CTRL** key and select **EPCS_Demo_MSS_CM3_app** and **EPCS_Demo_MSS_CM3_hw_platform** projects in the Project Explorer tab. Select **Build Configurations => Set Active => Release**.

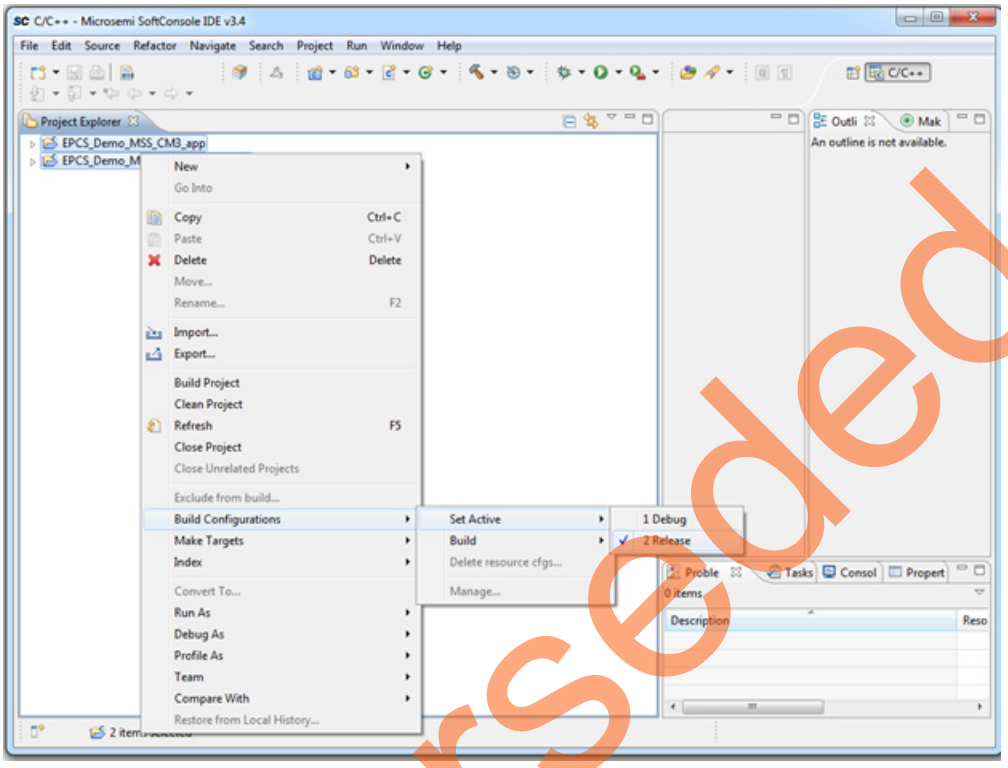


Figure 53 • **Release Mode Option**

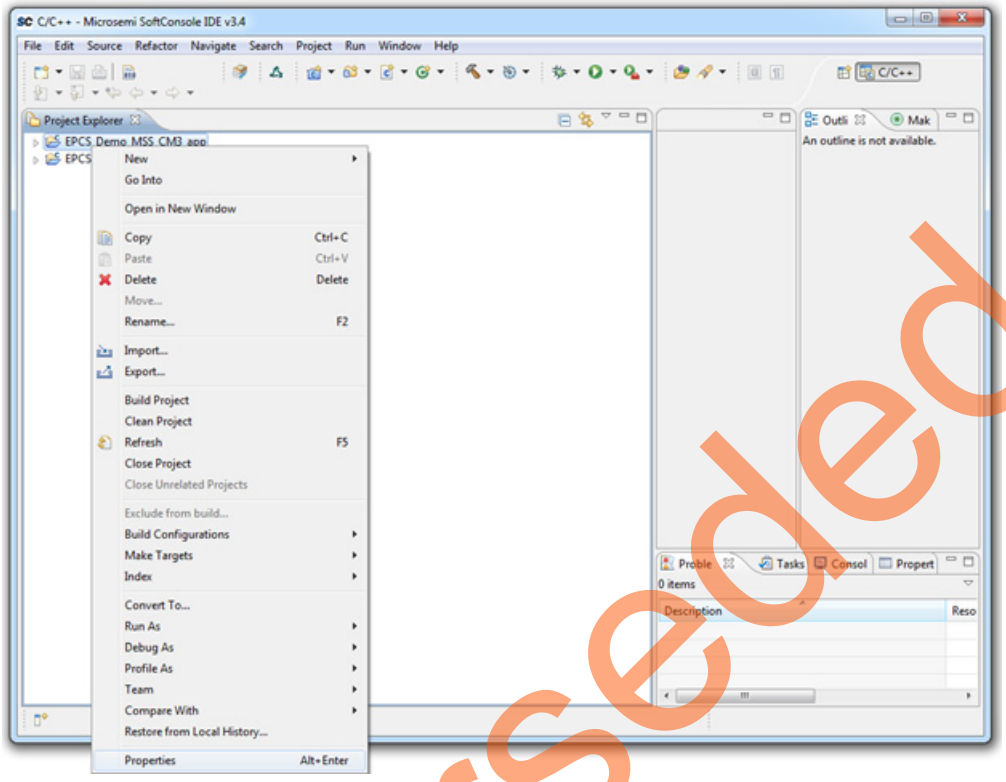


Figure 54 • **Properties Option**

6. The **Properties for EPCS_Demo_MSS_CM3_app** window is displayed as shown in Figure 55.

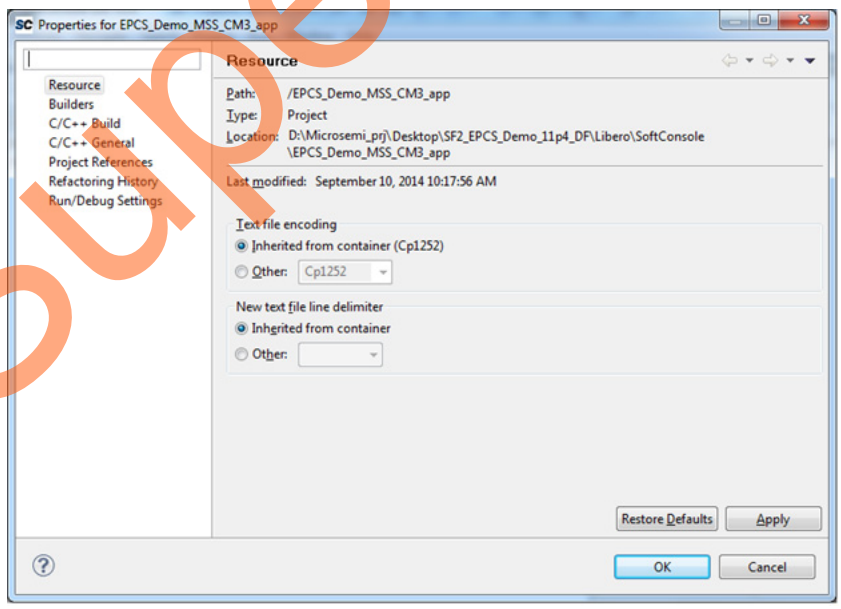


Figure 55 • **Properties Window**

7. In the **Properties for EPCS_Demo_MSS_CM3_app** window, expand **C/C++ Build** and select **Settings**.
8. Select **Miscellaneous** and provide the release mode linker script file to the linker by changing the 'Linker flags' field to *-T../../EPCS_Demo_MSS_CM3_hw_platform/CMSIS/startup_gcc/productionsmartfusion2-execute-in-place.ld* as shown in Figure 56.

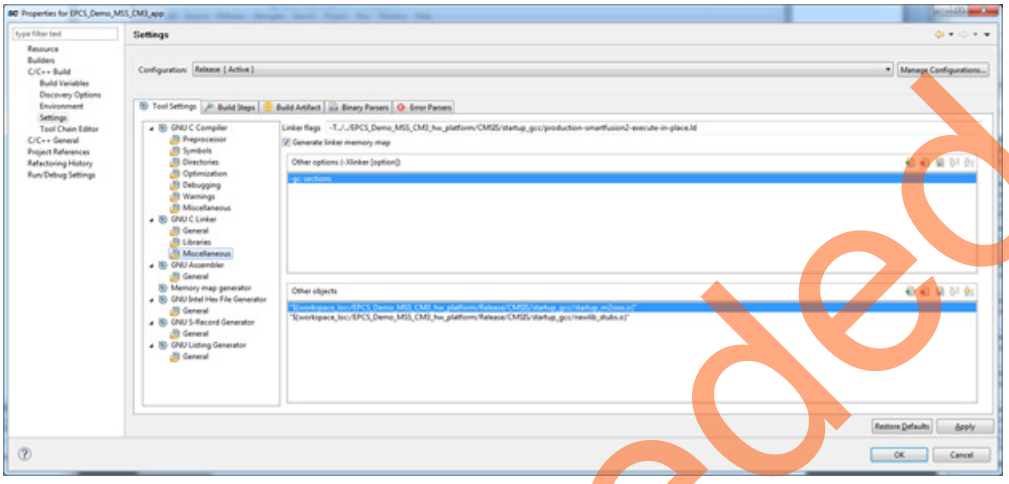


Figure 56 • **LD File Option**

9. Click **OK** to close the **Properties for EPCS_Demo_MSS_CM3_app** window.
10. To clean and build the project, select **Project > Clean** as shown in Figure 57.

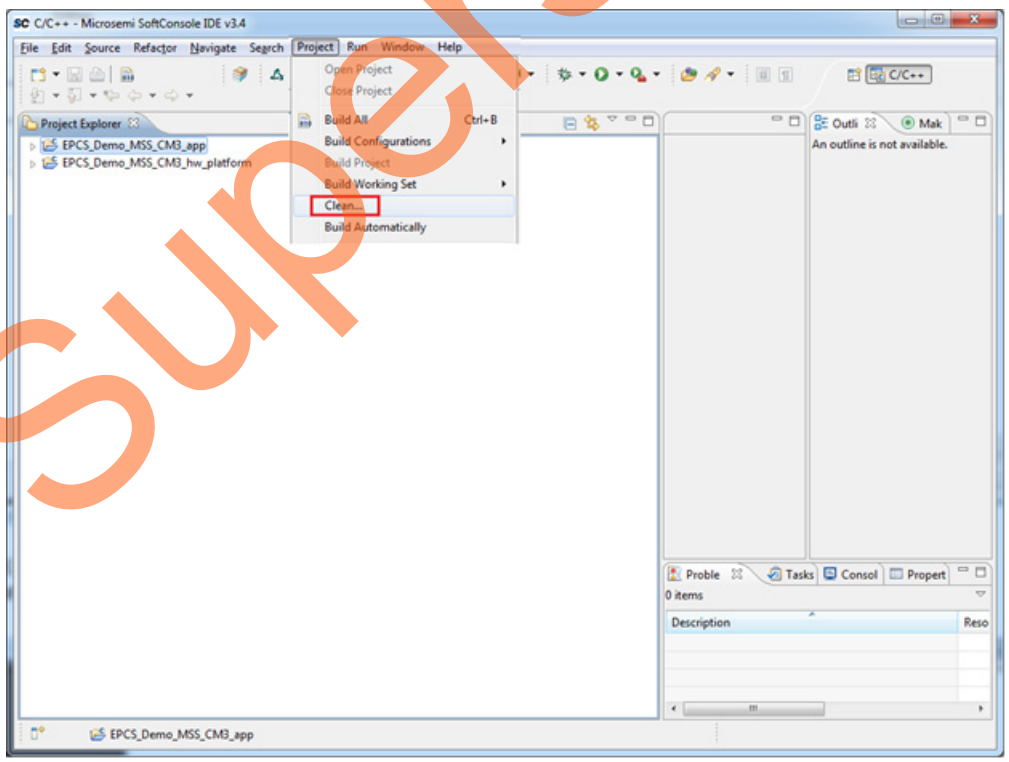


Figure 57 • **Building the SoftConsole Project**

11. The **Clean** window is displayed. Click **OK** to build the SoftConsole projects as shown in Figure 58.

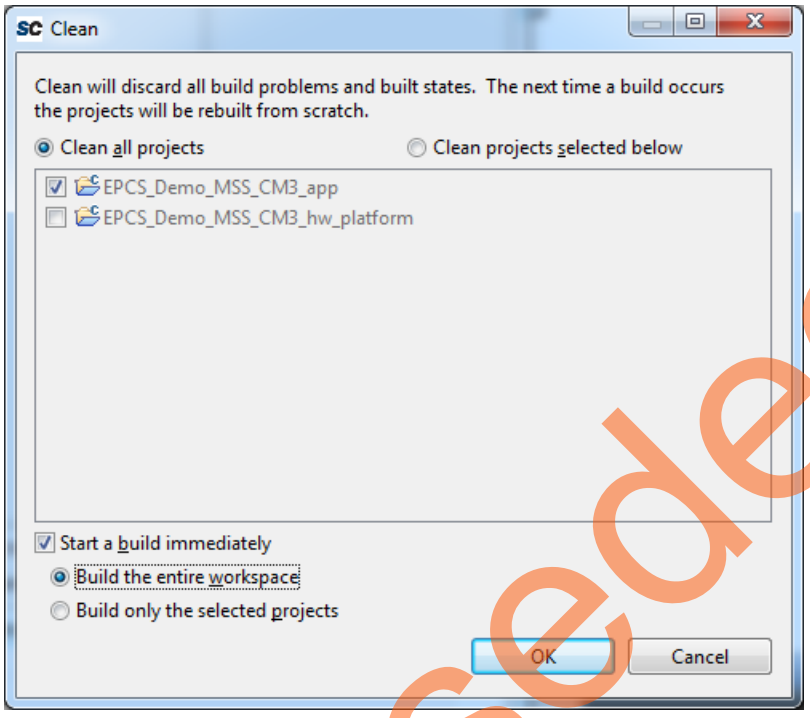


Figure 58 • **Clean and Build SoftConsole Project**

12. SoftConsole creates a hex file in the **Release** folder under the **EPCS_Demo_MSS_CM3_appproject** as shown in Figure 59.

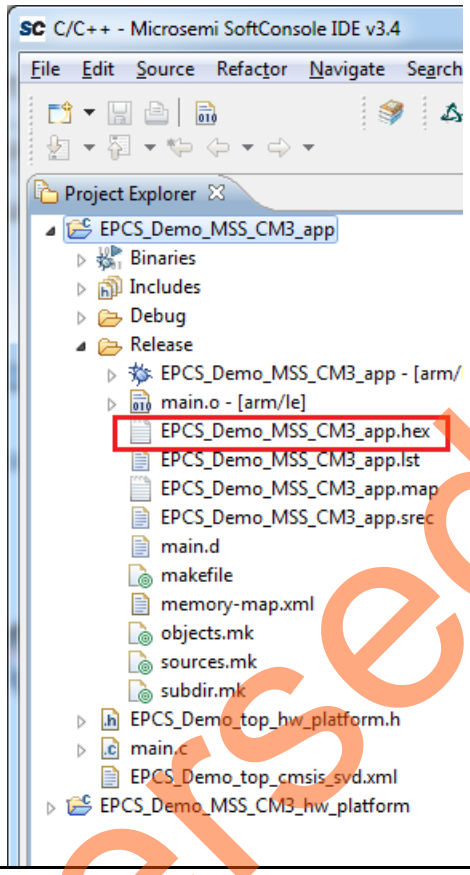


Figure 59 • **Generated Hex File**

13. Close the **SoftConsole** project window.
14. Open the Libero project and **EPCS_Demo_top** tab. Double-click **EPCS_Demo_0** and go to **System Builder - Memories** tab to add the eNVM data storage client.

    The eNVM configurator window is displayed as shown in Figure 60 on page 58.

15. Select **Data Storage** under the **Available client types** tab and click **Add to System**.

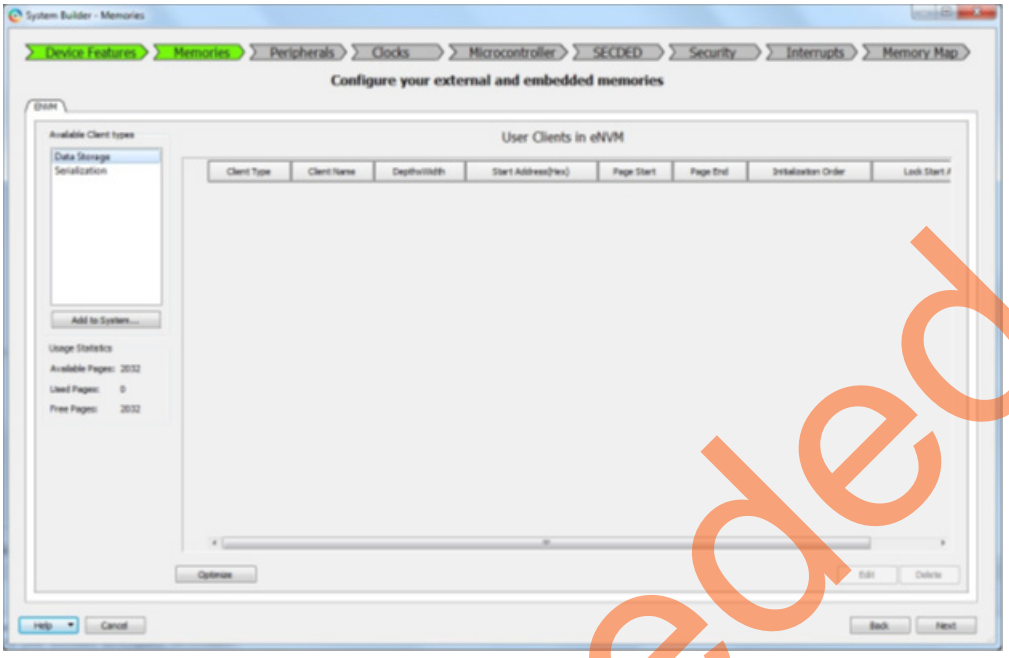


Figure 60 • **System Builder - Memory eNVM**

The **Add Data Storage Client** window is displayed as shown in Figure 61.

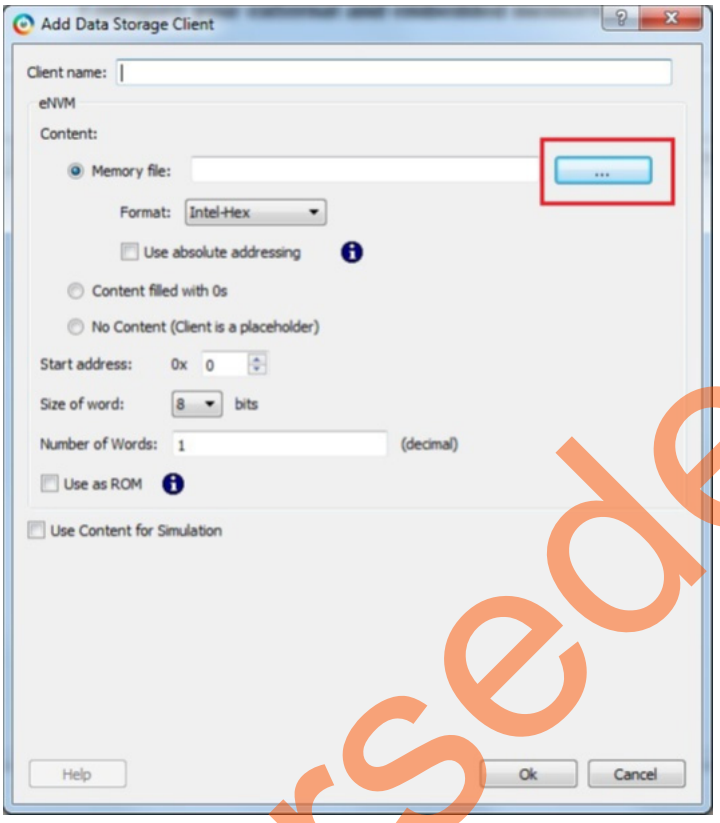


Figure 61 • **Add Data Storage Client**

16. Enter a data storage **Client Name** as eNVM in the **Add Data Storage Client** window.

17. Browse for the `.hex` file generated. The generated executable image can be found in the **Release** folder under the SoftConsole project workspace as shown in Figure 62.

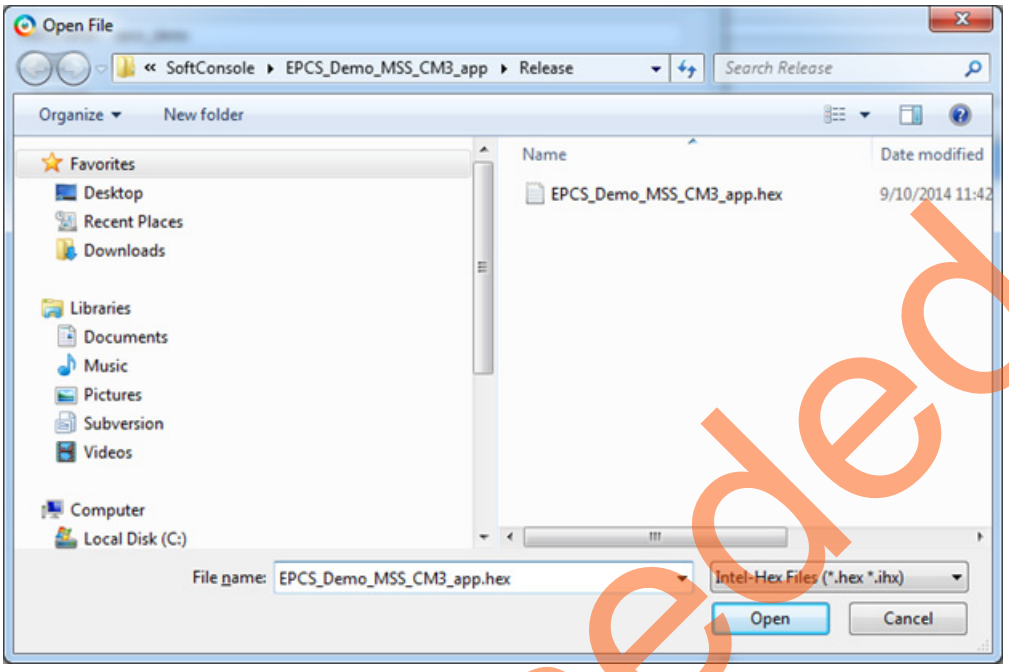


Figure 62 • **Selecting .hex File**

18. Click **Ok** to add the data storage client.

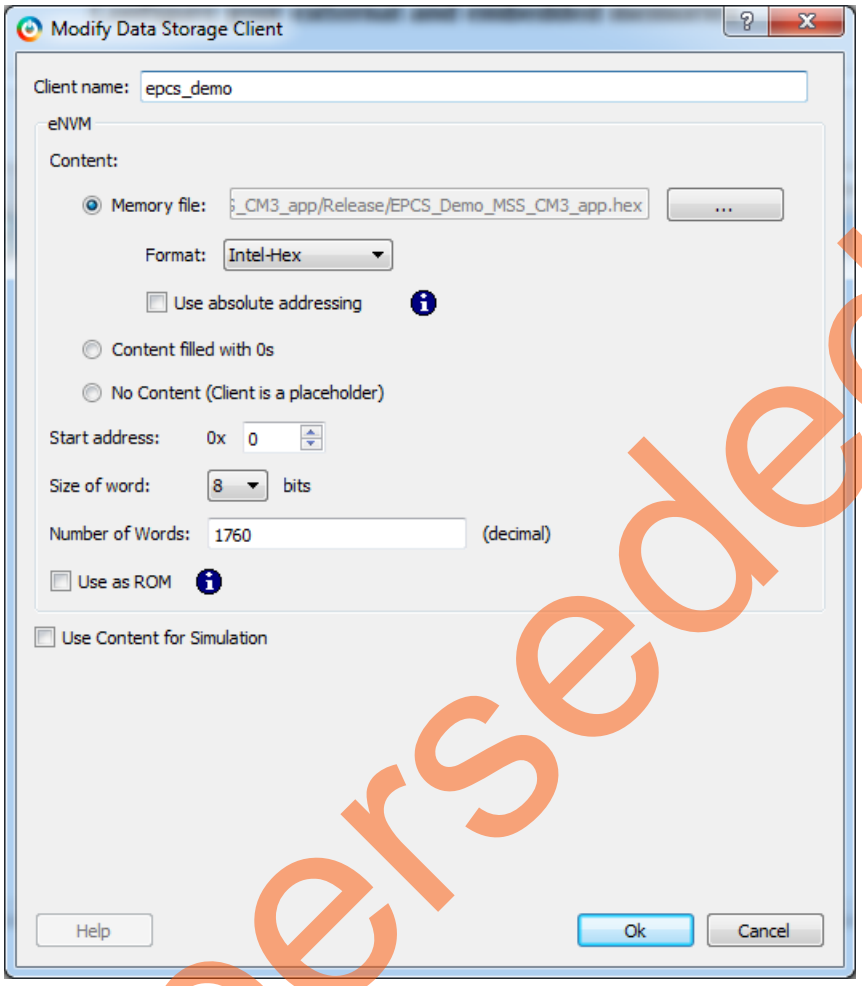


Figure 63 • **Add Data Storage Client**

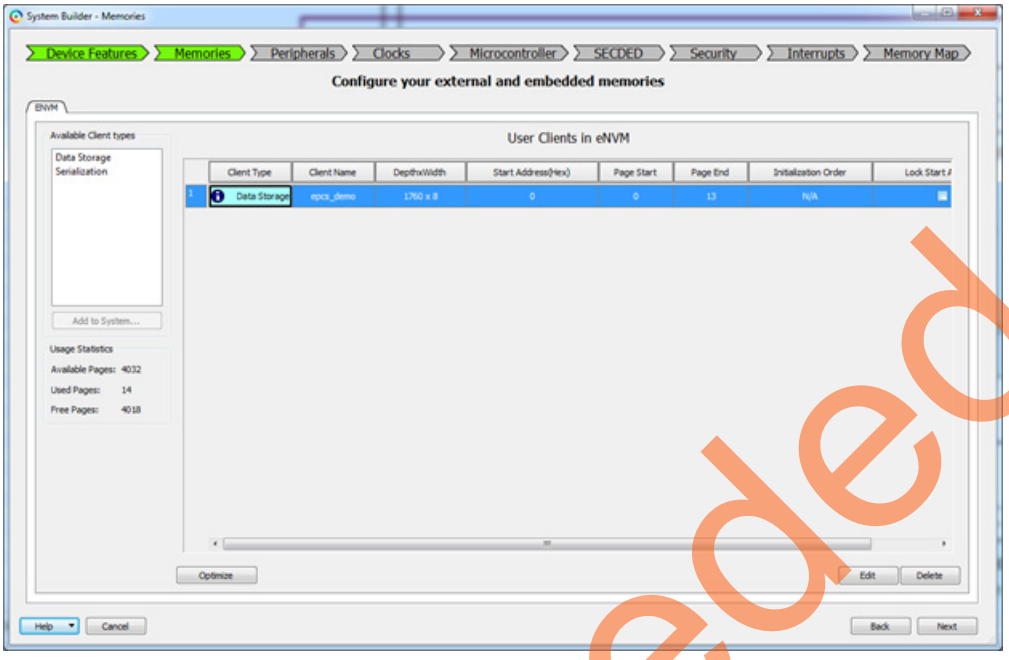


Figure 64 • **Modify Core - eNVM**

19. Save **EPCS_Demo_top** and regenerate the **EPCS_Demo_top** component by clicking **Generate Component** in SmartDesign.

Appendix 1: Using IGLOO2/SmartFusion2 for Customer Design

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 1.

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 1.

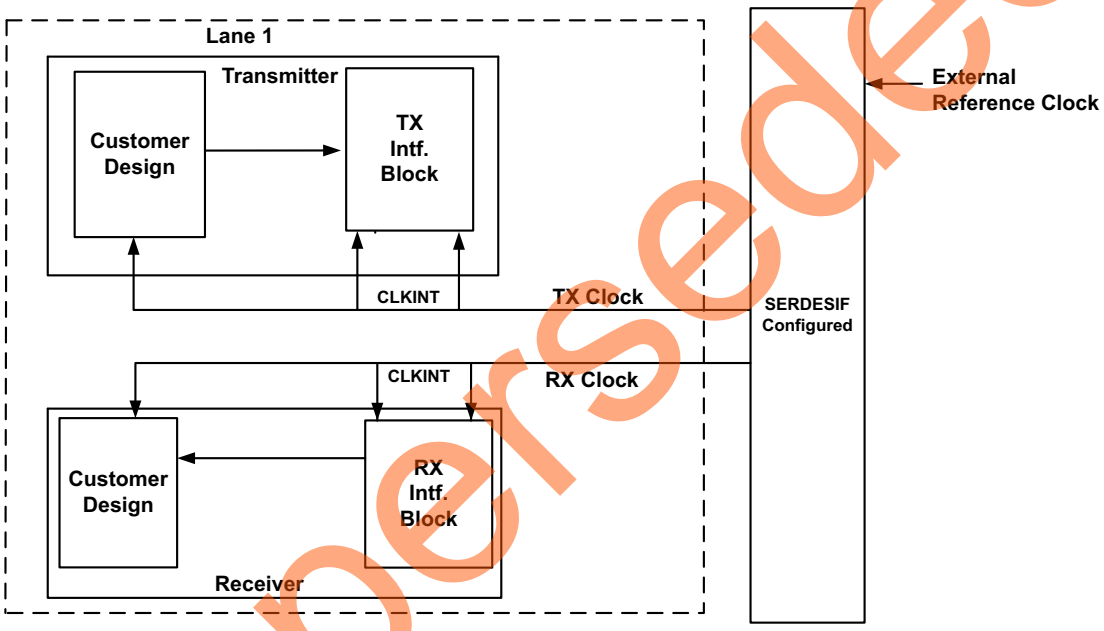


Figure 1 • **Replacing Demo Design with Customer Design**

This demo is targeted to M2GL010T and M2GL025T devices, it is not required to force the assignment of a clock resource. In the larger IGLOO2/SmartFusion2 devices, it is recommended to use 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 IGLOO2 family and optimizes the interface for both setup and hold. These blocks must be used exactly from this demo design into all EPCS designs.

Appendix 2: Simulating the Design

Use the following steps to simulate the design:

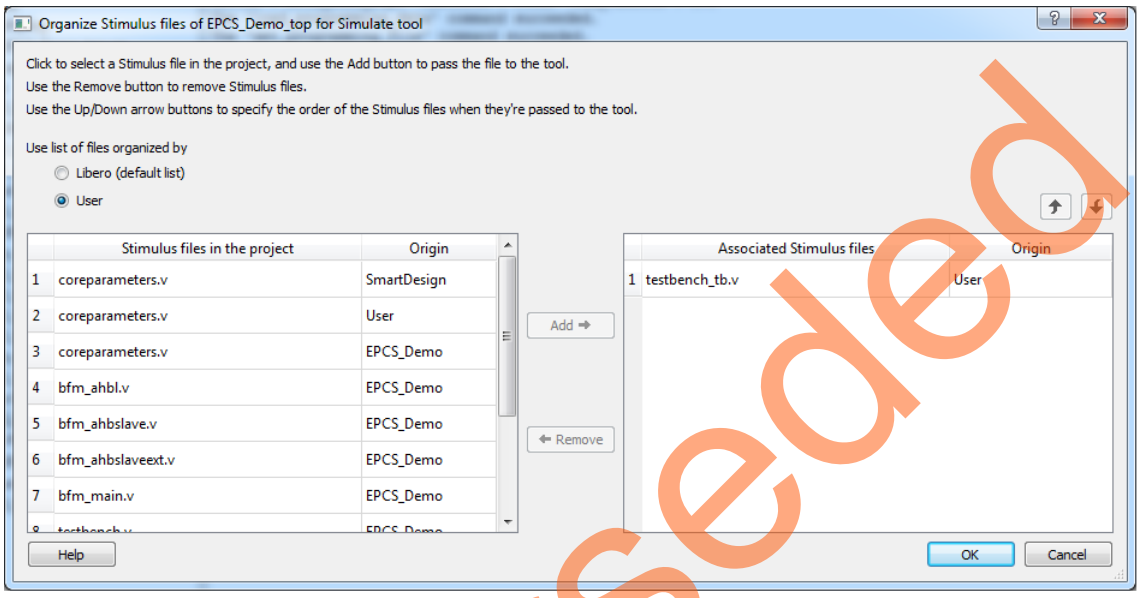


Figure 1 • **Organizing Simulation Testbench in Project**

1. Copy the `testbench_tb.v` file to the **Stimulus** folder of the Libero project. Right-click on **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: `wave.do` and `testbench_tb.v` files are available in the **Test benches** folder of **EPCS_Demo** project.

3. Go to **Project>Project Settings>waveforms** and select the **Include DO file** check box.
4. Change the **Simulation runtime** to 2 us using the **Do File** option under **Project Settings**.
5. Select `vsim command` under **Project Settings** and change the resolution to **1ps**.
6. Click **Save** to save the settings.
7. Right-click on **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. 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 2.

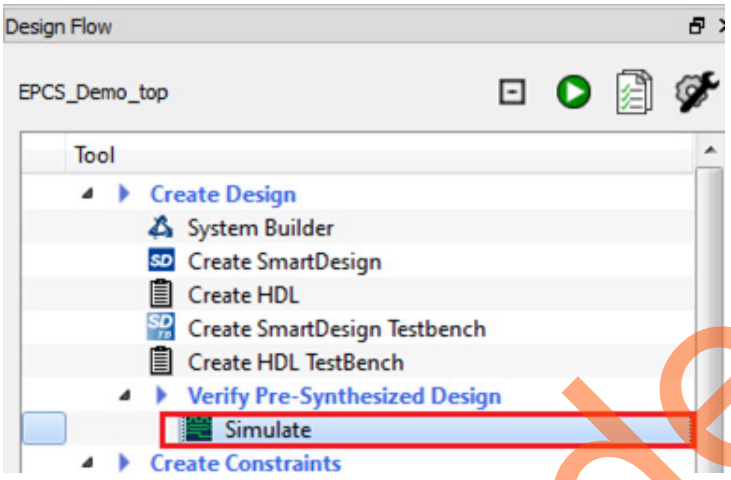


Figure 2 • **Simulating the Design**

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. 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 3.

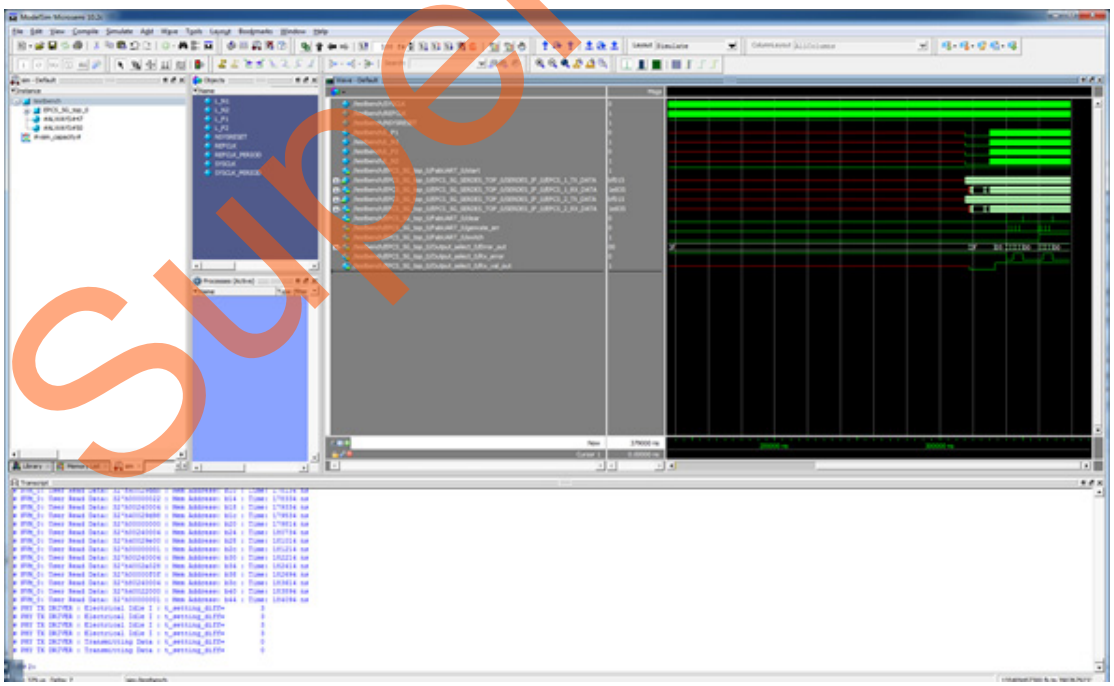


Figure 3 • **Simulation Waveform Window**

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.

Appendix 3: 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. Refer to the **SmartTime Users Guide** from the Libero SoC software Help.

Verify Timing

1. Right-click on **Verify Timing** and select **Open Interactively** to open SmartTime.

Figure 1 • **Verify Timing**

SmartTime window is displayed in Max Delay Analysis View as shown in Figure 2.

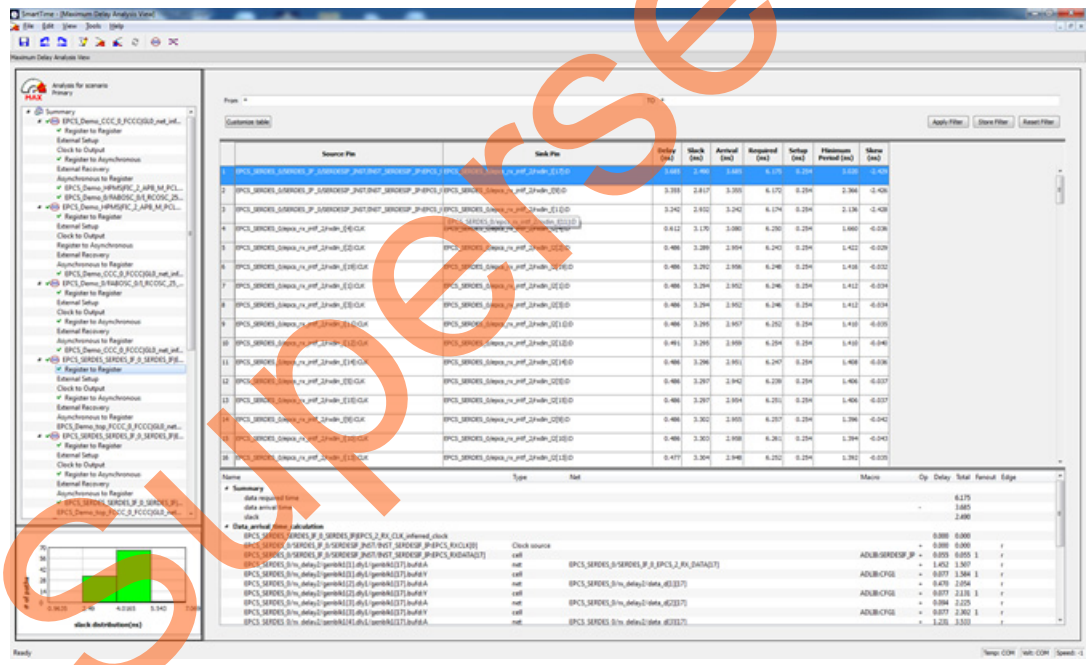


Figure 2 • **SmartTime Session**

Figure 2 shows the SmartTime results. It reports any setup or hold time violations that cause design issues in the actual hardware.

Appendix 4: Status Signals

Table 1 describes the various status signals.

Table 1 • **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 as a result, increments the Error Count display.
Clear Error	Sets error count to zero.

Appendix 5: Jumper Locations

Figure 1 shows the jumper locations in the IGLOO2/SmartFusion2 Evaluation Kit.

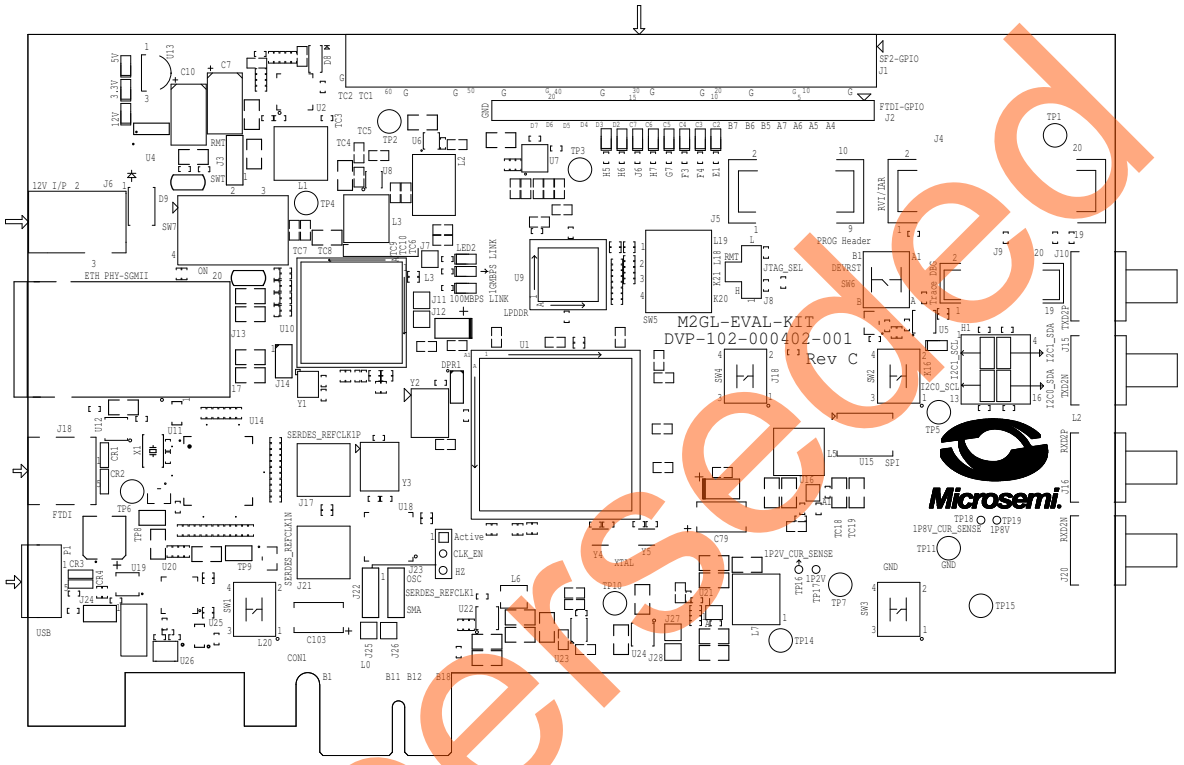


Figure 1 • **IGLOO2/SmartFusion2 Evaluation Kit Silkscreen Top View**

Note: The location of the jumpers in Figure 1 is searchable.

A – List of Changes

The following table lists critical changes that were made in each revision of the chapter in the demo guide.

Date	Changes	Page
Revision 2 (January 2015)	Replaced the updated IGLOO2 design files (SAR 62650).	NA
Revision 1 (October 2014)	Initial release	NA

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

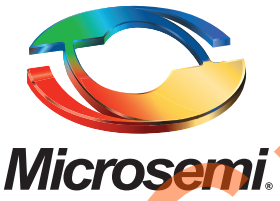
Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.



Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace 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; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate 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

E-mail: sales.support@microsemi.com

50200570-2/01-15