
SmartFusion2 SoC FPGA PCIe Control Plane Demo

User Guide

Superseded



Table of Contents

SmartFusion2 SoC FPGA - PCIe Control Plane Demo	3
Introduction	3
Demo Requirements	4
Hardware and Software Requirements	4
Design Files	4
Demo Design Description	5
Building the Demo	6
Step 1: Creating a Libero SoC Project	6
Instantiating the SERDESIF Component in PCIe_Demo_top SmartDesign	16
Instantiating Debounce Logic in PCIe_Demo_top SmartDesign	22
Connecting Components in PCIe_Demo_top SmartDesign	23
Configuring and Generating Firmware	29
Step 2: Creating an eNVM Client	30
Step 3: Developing the Simulation Stimulus	41
Step 4: Simulating the Design	44
Step 5: Generating the Program File	47
Running the Demo	50
Demo Setup	50
Running the Demo Design	52
Running the Demo Design on Windows	53
Running the Demo Design on Linux	66
Conclusion	76
A List of Changes	77
B Product Support	78
Customer Service	78
Customer Technical Support Center	78
Technical Support	78
Website	78
Contacting the Customer Technical Support Center	78
Email	78
My Cases	79
Outside the U.S.	79
ITAR Technical Support	79

SmartFusion2 SoC FPGA - PCIe Control Plane Demo

Introduction

SmartFusion[®]2 system-on-chip (SoC) field programmable gate array (FPGA) devices integrate a fourth generation flash-based FPGA fabric and an ARM[®] Cortex[™]-M3 processor, along with high performance communication interfaces on a single chip. The SmartFusion2 high speed serial interface (SERDESIF) provides a fully hardened PCIe endpoint (EP) implementation and is compliant with PCIe Base Specification Revision 2.0 and 1.1. For more details, refer to the [SmartFusion2 SoC FPGA High Speed Serial Interfaces User's Guide](#).

The demo explains the SmartFusion2 embedded PCI Express feature and how this can be used as a low bandwidth control plane interface using the SmartFusion2 Development Kit. The demo provides a simple design to access the SmartFusion2 PCIe EP from a Host PC. A GUI is provided for read and write access to the SmartFusion2 PCIe configuration space and memory space of BAR0 and BAR1. The demo also provides Host PC device drivers for the SmartFusion2 PCIe EP. This demo can run on both windows and Red Hat Linux operating system.

Figure 1 shows the top-level block diagram for the PCIe control plane demo. The demo design uses a SmartFusion2 PCIe interface with a maximum link width of x4 to interface with a Host PC PCIe Gen2 slot. The SmartFusion2 microcontroller subsystem (MSS) GPIOs control the LEDs and switches on the SmartFusion2 Development Kit through the PCIe interface. The Host PC can also read memory and writes to the SmartFusion2 eSRAM through the GUI. The Host PC can also be interrupted by using the push button on the SmartFusion2 Development Kit.

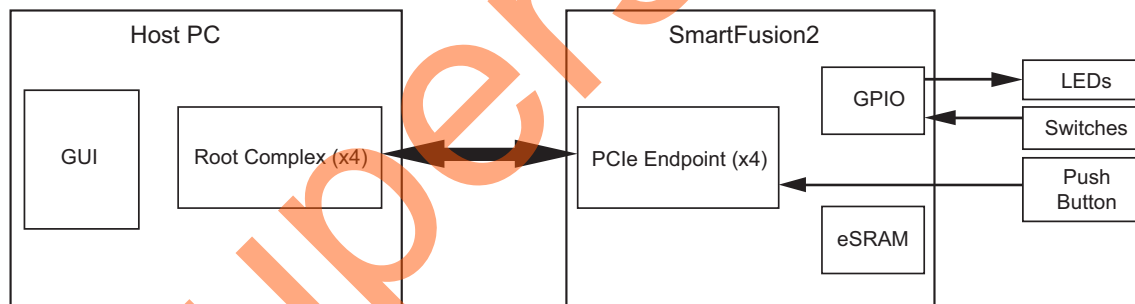


Figure 1 • PCIe Control Plane Demo Top-Level Block Diagram

The demo design performs the tasks listed below:

- Displays the PCIe link enable/disable, negotiated link width, and the link speed.
- Controls the status of LEDs on SmartFusion2 Development Kit according to the command from the GUI.
- Displays the position of DIP Switches on SmartFusion2 Development Kit.
- Enables read and writes to eSRAM.
- Interrupts the Host PC, when the push button is pressed. The GUI displays the count value of the number of interrupts sent from the SmartFusion2 Development Kit.
- Displays the SmartFusion2 PCIe Configuration Space.

Demo Requirements

Hardware and Software Requirements

The hardware and software required to run the demo are listed in Table 1.

Table 1 • Required Hardware and Software to Run the Demo

Hardware	Version
SmartFusion2 Development Kit	Rev C or later ¹
12 V adapter (provided along with the kit)	-
FlashPro4 programmer (provided along with the kit)	-
PCI Edge Card Ribbon Cable (provided along with the kit)	-
Host PC with an available PCIe 2.0 Gen1 or Gen2 compliant slot	Operating system: Windows XP SP2: 64-bit Windows 7: 64-bit or Red Hat Linux Kernel Version: 2.6.18-308
Software	
Libero [®] System-on-Chip (SoC)	11.3
SoftConsole	3.4
Host PC Drivers (provided along with the design files)	-
GUI executable (provided along with the design files)	-
<i>Note: The SmartFusion2 Development Kit has a label to specify the version.</i>	

Design Files

The design files for this demo can be downloaded from the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=M2S_PCIE_Control_DEMO_DF

Design files include:

1. Libero project
2. Linux_64bit
3. ProgrammingFile
4. Windows_64bit
5. Source files
6. Readme file

Refer to the Readme.txt file provided in the design files for the complete directory structure.

Demo Design Description

This demo design implements the SmartFusion2 embedded PCI Express interface as a low bandwidth control plane interface. This design provides Host PC drivers and a Host PC interface over PCIe to control the SmartFusion2 device. Figure 2 on page 5 shows a detailed block diagram of the design implementation. The PCIe EP device receives commands from the Host PC through the GUI and does corresponding memory writes to the SmartFusion2 MSS address space. The MSS address space provides a GPIO block and eSRAM memory block which is accessed through a Fabric Interface Controller (FIC_0).

The SERDES_IF_1 is configured for a PCIe 2.0, x4 link width with GEN2 speed. The PCIe interface to the fabric uses an AMBA High-speed Bus (AHB). The AHB master interface of SERDESIF is enabled and connected to the AHB slave interface of FIC_0 to access the MSS peripherals. The SmartFusion2 PCIe BAR0 and BAR1 are configured in 32-bit memory mapped memory mode.

The AXI master windows of the SERDESIF PCIe provide address translation for accessing one address space from another address space as the PCIe address is different from SmartFusion2 AHB bus matrix address space. The AXI master window 0 is enabled and configured to translate the BAR0 memory address space to the MSS GPIO address space to control the MSS GPIOs. The AXI master window 1 is enabled and configured to translate the BAR1 memory address space to the eSRAM address space to perform read and writes from PCIe.

MSS GPIO block is enabled and configured as below:

- GPIO_0 to GPIO_7 as outputs and connected to LEDs
- GPIO_8 to GPIO_11 as inputs and connected to DIP switches

The PCIe interrupt line is connected to the SW3 push button on the SmartFusion2 Development Kit. The FPGA clocks are configured to run the FPGA fabric and MSS at 100 MHz.

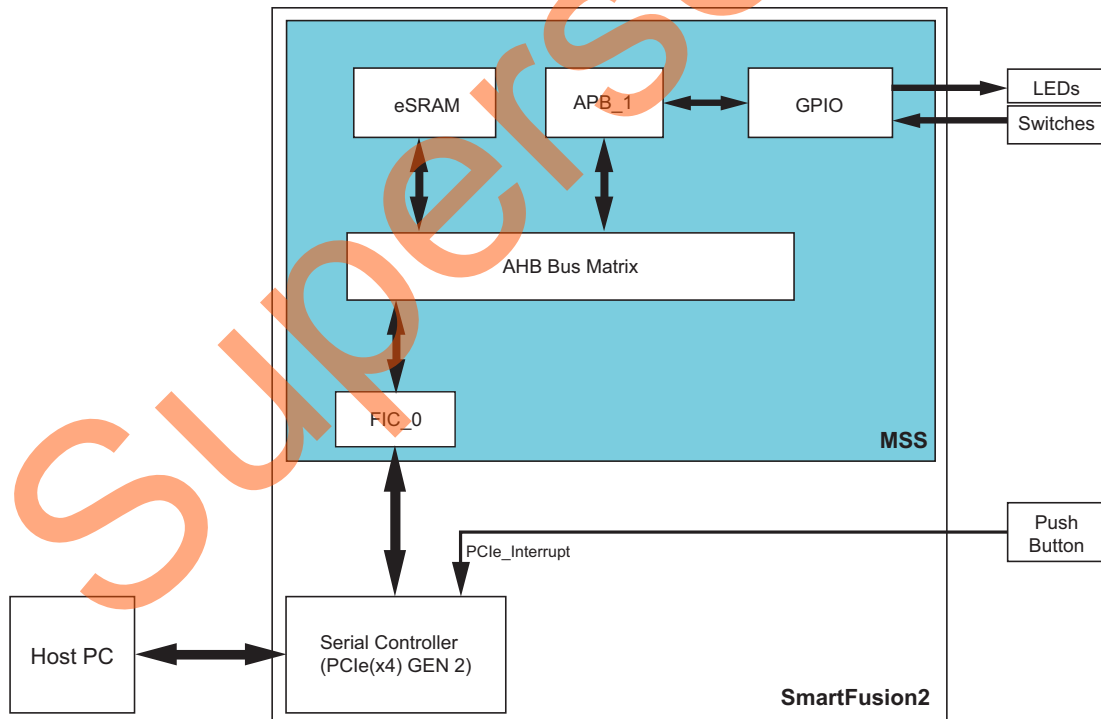


Figure 2 • PCIe Control Plane Demo Block Diagram

Building the Demo

This demo design provides a complete design flow starting from a new project to a working design on the SmartFusion2 Development Kit. This process includes usage of the tools in the Libero SoC design suite to program a SmartFusion2 device.

Building the demo involves the following steps:

1. [Step 1: Creating a Libero SoC Project](#)
2. [Step 2: Creating an eNVM Client](#)
3. [Step 3: Developing the Simulation Stimulus](#)
4. [Step 4: Simulating the Design](#)
5. [Step 5: Generating the Program File](#)

Step 1: Creating a Libero SoC Project

1. Click **Start > Programs > Microsemi Libero SoC v11.3 > Libero SoC v11.3**, or click the shortcut on your desktop. The Libero SoC v11.3 Project Manager is displayed as shown in [Figure 3](#).

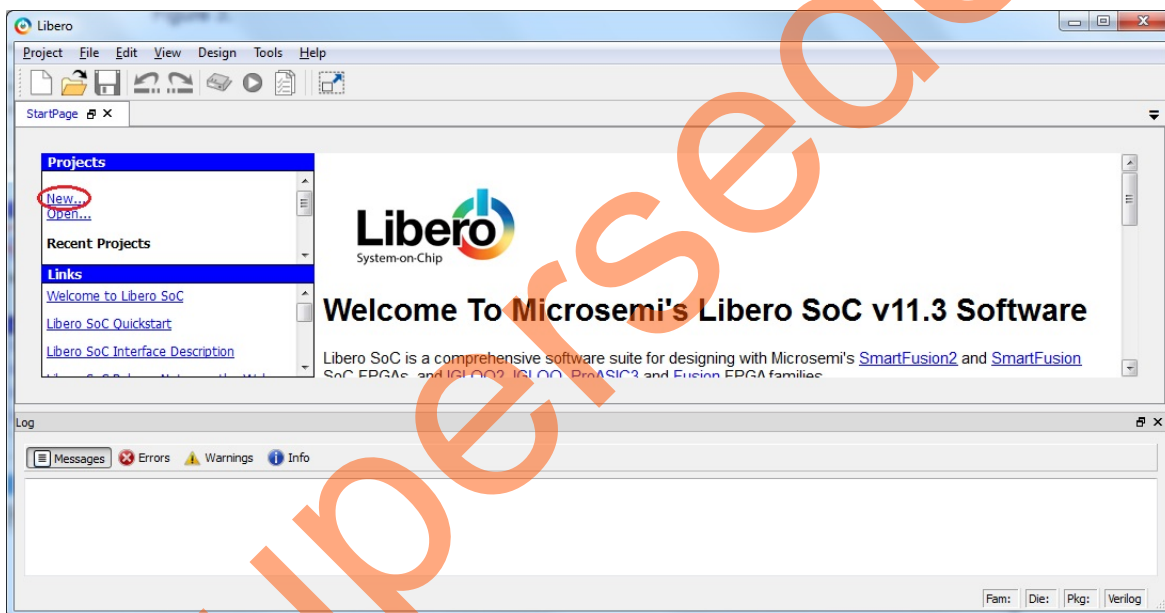


Figure 3 • Libero SoC v 11.3 Project Manager

2. Create a new project by selecting **New** on the Start Page tab (highlighted in [Figure 3](#)), or by clicking **Project > New Project** from the Libero SoC menu. Enter the information as required for the new project and the Device in the **New Project** dialog box as shown in [Figure 4](#) on [page 7](#).
 - Project Name: PCIe_Demo
 - Project Location: Select an appropriate location (for example, D:/Microsemi_prj)
 - Preferred HDL type: Verilog or VHDL
 - Family: SmartFusion2
 - Die: M2S050T
 - Package: 896 FBGA
 - Speed: -1
 - Core Voltage: 1.2
 - Operating conditions: COM

3. Select **Use Design Tool** and **Use System Builder** in the Design Templates and Creators section of the New Project window as shown in [Figure 4](#) on page 7.

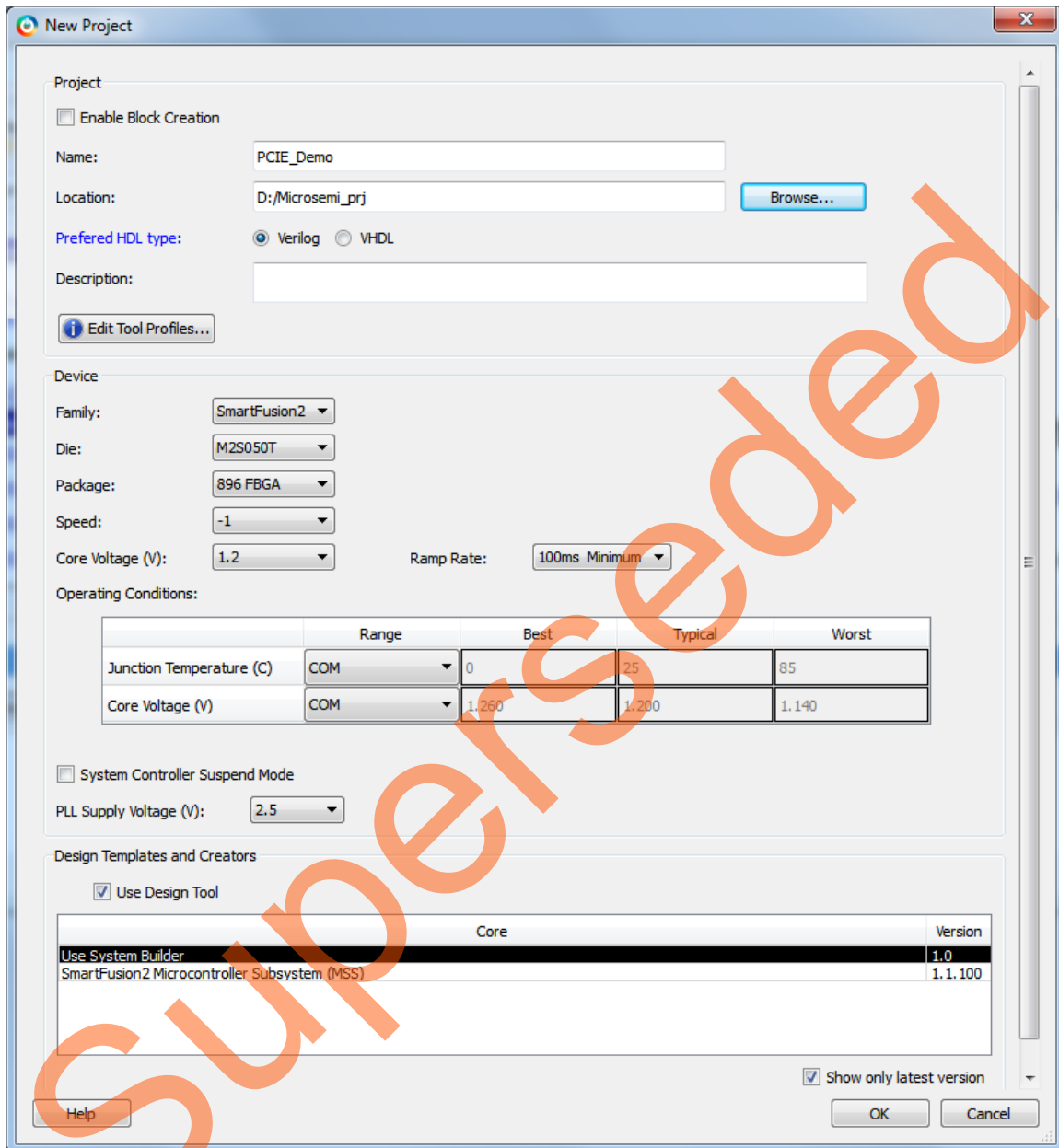


Figure 4 • Libero SoC v 11.3 New Project Dialog Box

4. Clicking **Edit Tool Profiles** displays the Tool Profiles window as shown in [Figure 5](#). Check the below tool settings:
 - Software IDE: SoftConsole
 - Synthesis: Synplify Pro ME I2013.09M SP1-1
 - Simulation: ModelSim 10.2c

- Programming: FlashPro 11.3

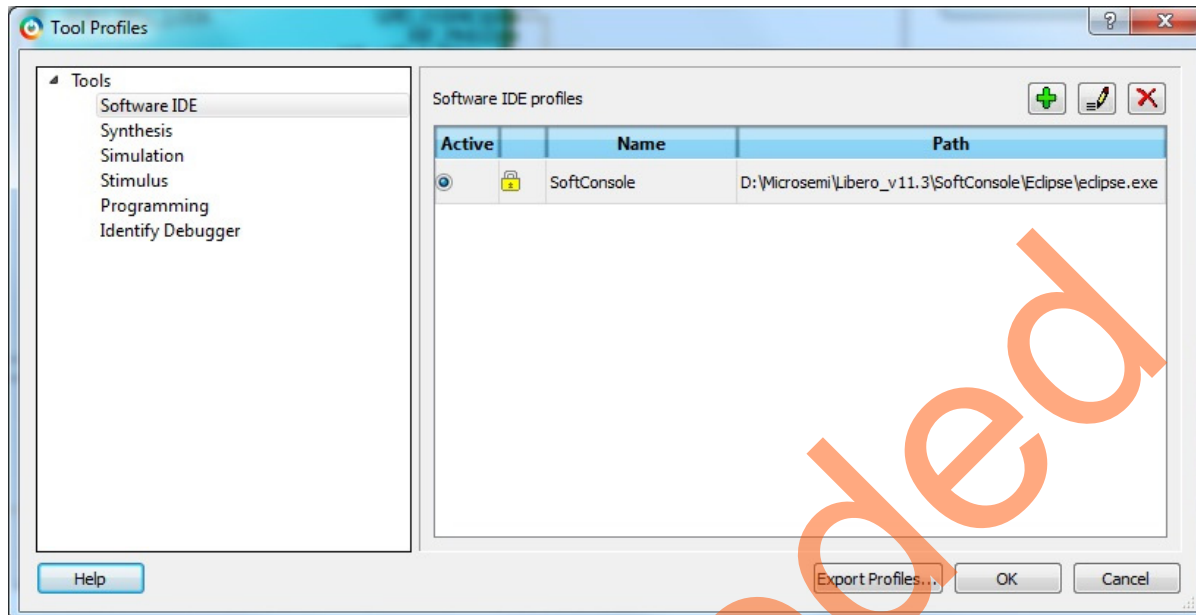


Figure 5 • Tool Profiles

- Click **OK** on the **Tool Profiles** and click **OK** on the **New Project** window.
- Selecting the **Use System Builder** displays the “Enter a name for your system” dialog box, as shown in [Figure 6](#).

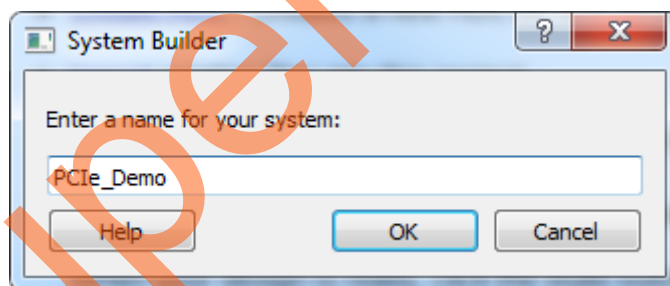


Figure 6 • Create New System Builder Dialog Box

- Enter **PCIe_Demo** as the name of the system and click **OK**.
- The System Builder dialog box is displayed with the Device Features page open by default. Enter the following details in the **System Builder – Device Features** page as shown in [Figure 7](#) on [page 9](#):
 - Memory: Clear all except MSS On-chip Flash Memory (eNVM)
 - High-speed serial interfaces: Check SERDESIF_1
 - Microcontroller Options: Clear All

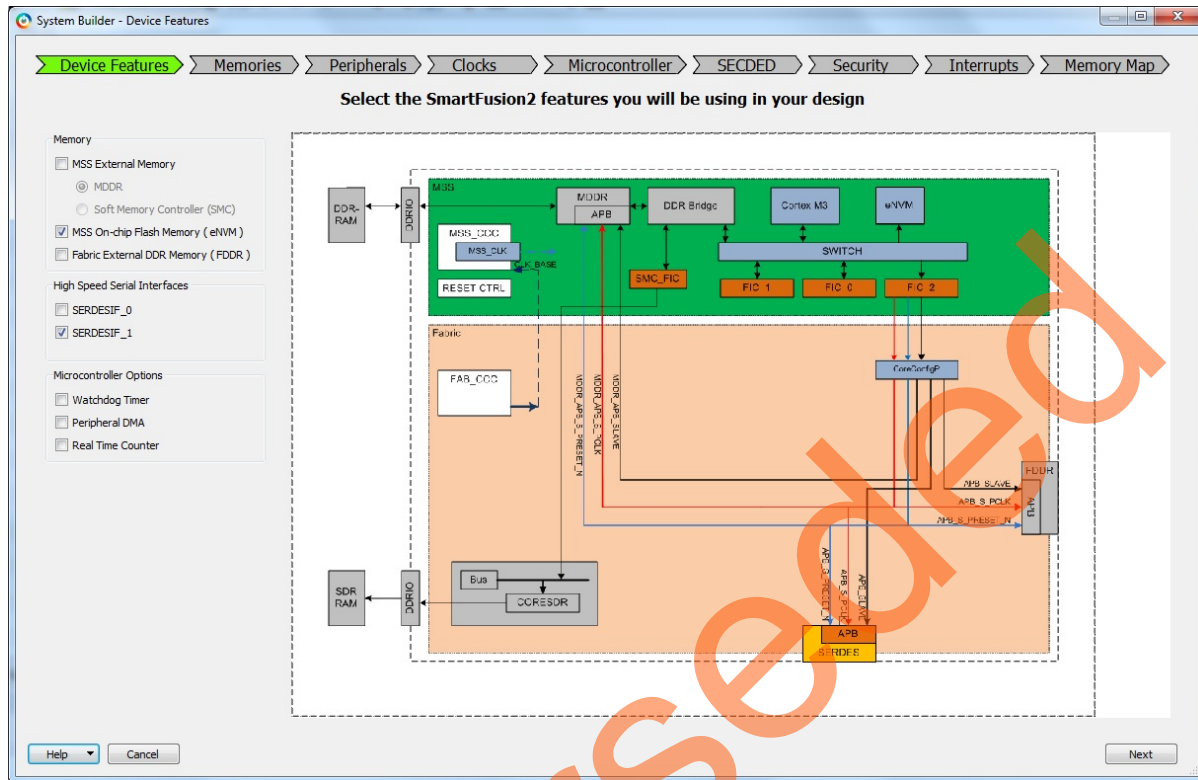


Figure 7 • SmartFusion2 SoC FPGA System Builder Configurator

9. Click **Next**. The **System Builder – Memories** page is displayed.

- Click **Next**. The **System Builder – Peripherals** page is displayed. Drag the **Fabric AMBA Master** to **MSS_FIC_0 – Fabric Master Subsystem** as shown in Figure 8. It enables the MSS_FIC_0 slave interface.

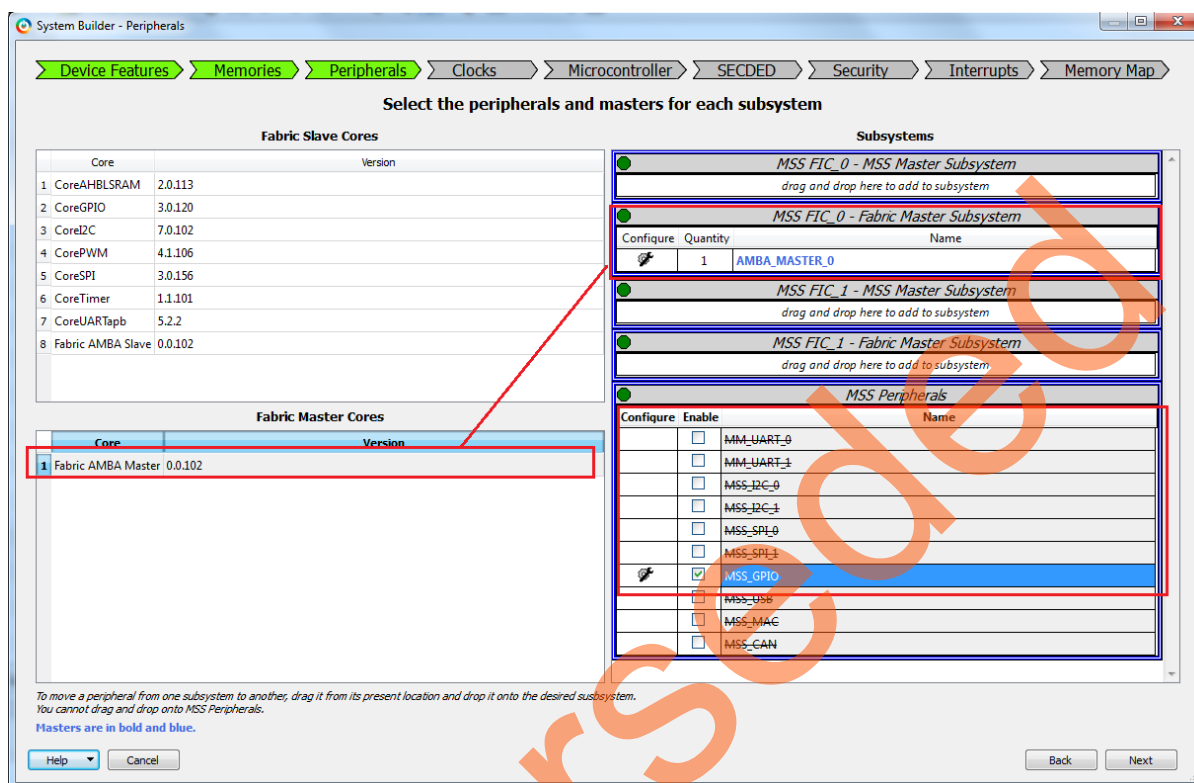


Figure 8 • System Builder Configurator – Peripherals Page

- Disable the MSS Peripherals except MSS_GPIO. The **System Builder – Peripherals** page is displayed as shown in Figure 9 on page 11. Configure **MSS_FIC_0 – Fabric Master Subsystem** for AHB-Lite by clicking on the AMBA_MASTER_0 configurator button highlighted in Figure 9 on page 11. This displays a drop-down list as shown in Figure 10 on page 11.

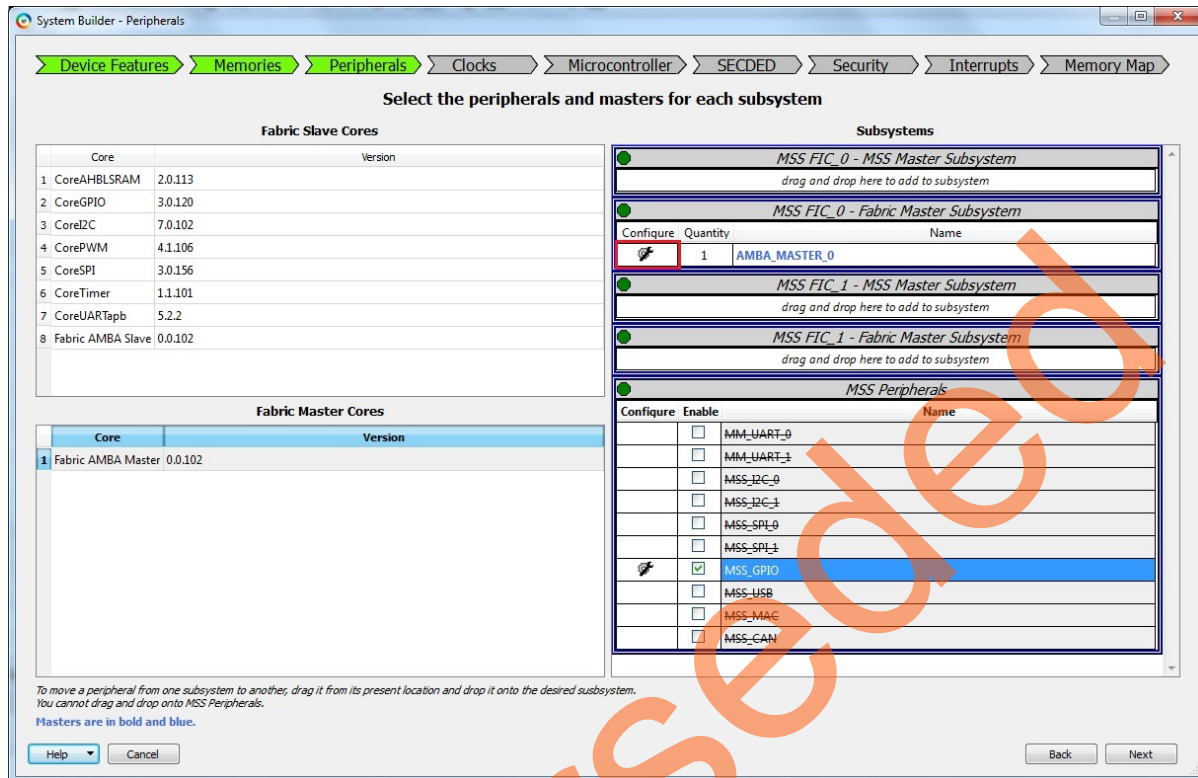


Figure 9 • System Builder Configurator – Peripherals Page

12. Select **AHBLite** from the drop-down list as shown in Figure 10.

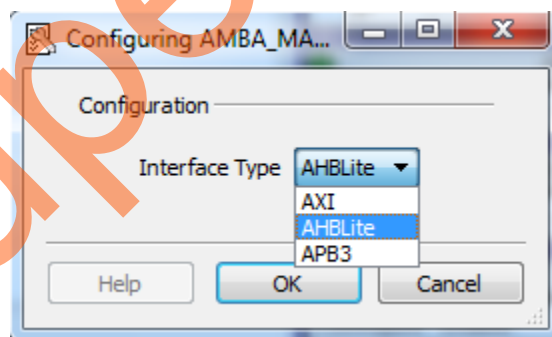


Figure 10 • Configuring AMBA Master

13. Configure MSS_GPIO by clicking **MSS_GPIO Configure** as shown in Figure 11.

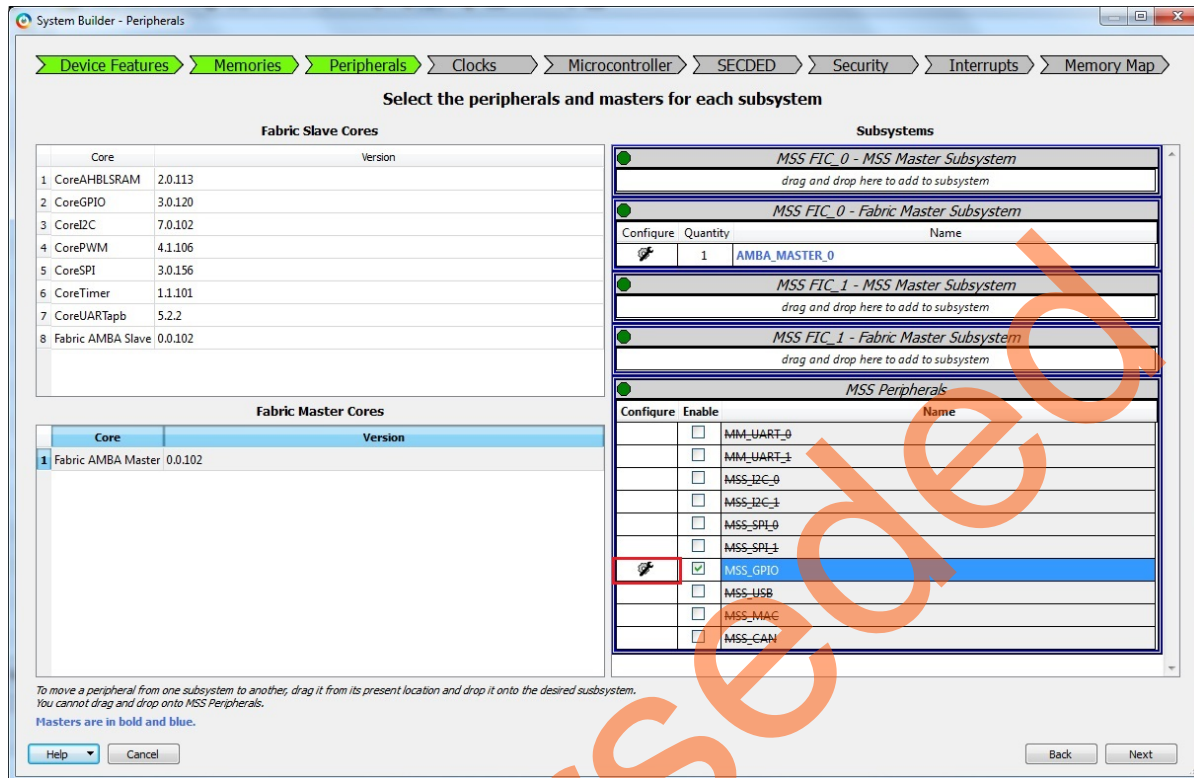


Figure 11 • System Builder – Peripherals Page

14. Double-click **MSS_GPIO** configuration button as shown in Figure 11 and configure:
- GPIO_0 to GPIO_7 as outputs and their connectivity to FABRIC_A to connect with LEDs
 - GPIO_8 to GPIO_11 as inputs and their connectivity to FABRIC_A, to connect with DIP switches

This design requires configuring GPIO_0 to GPIO_7 to drive LED_1 to LED_8 on the SmartFusion2 Development Kit, and GPIO_8 to GPIO_11 to connect DIP1 to DIP4. These signals will be routed through the fabric to the I/O pins.

Figure 12 shows the MSS GPIO Configurator.

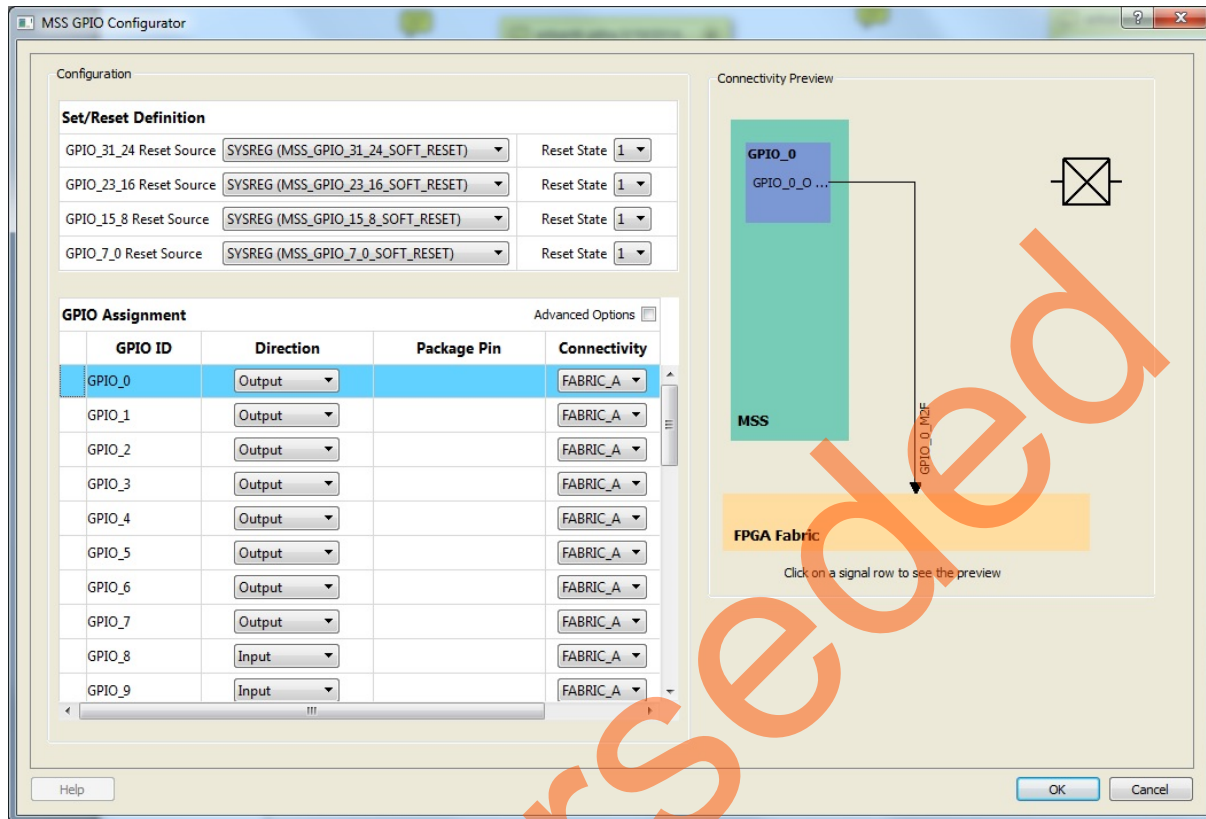


Figure 12 • GPIO Configuration

15. Click **OK** on MSS GPIO Configurator.

SmartFusion2 SoC FPGA - PCIe Control Plane Demo

16. Click **Next**. The **System Builder – Clock** page is displayed, as shown in [Figure 13](#). Change the configuration of **System Clock** from 100 MHz to 50 MHz. The dedicated input pad will be connected to on board 50 MHz oscillator. The M3 CLK is configured to 100 MHz by default.

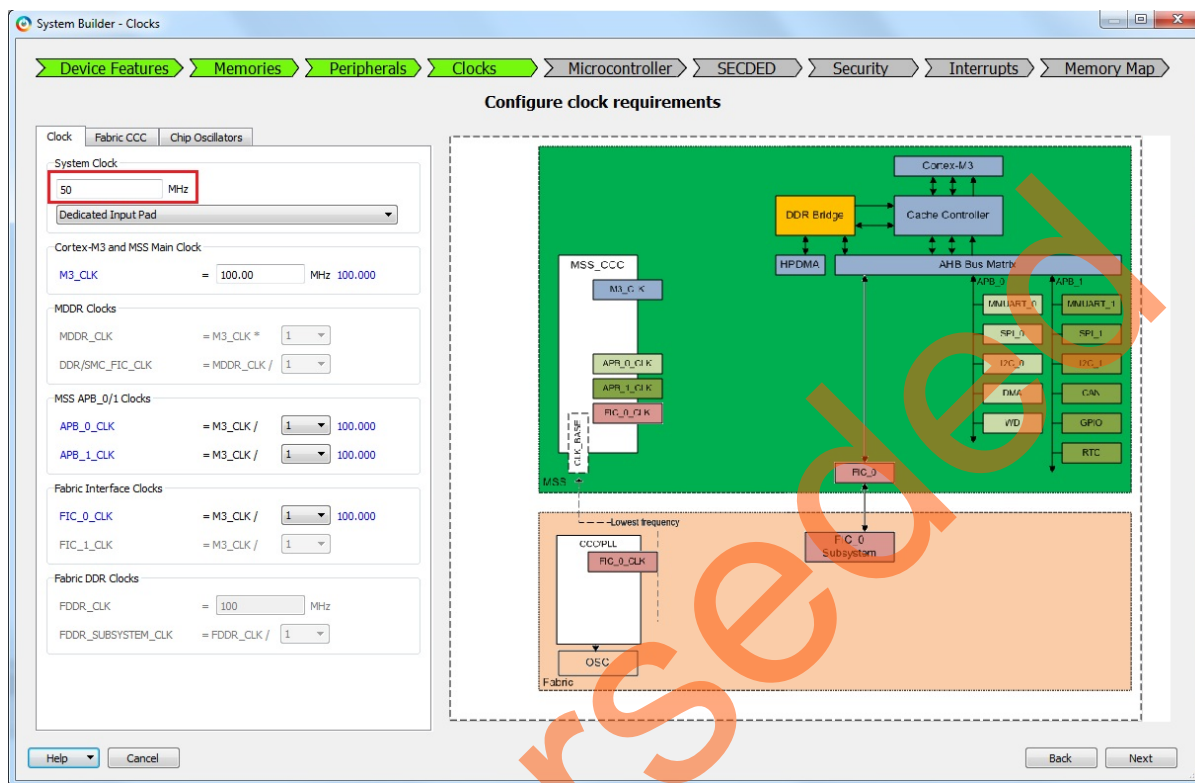


Figure 13 • System Builder Configurator – Clock Page

17. Click **Next**. The **System Builder - Microcontroller** page is displayed. Leave all the default selections.
18. Click **Next**. The **System Builder - SECDDED** page is displayed. Leave all the default selections.
19. Click **Next**. The **System Builder - Security** page is displayed. Leave all the default selections.
20. Click **Next**. The **System Builder - Interrupts** page is displayed. Leave all the default selections.
21. Click **Next**. The **System Builder - Memory Map** page is displayed. Leave all the default selections.
22. Click **Finish**.

The **System Builder** generates the system based on the selected options.

The System Builder block is created and added to Libero SoC project automatically, as shown in Figure 14.

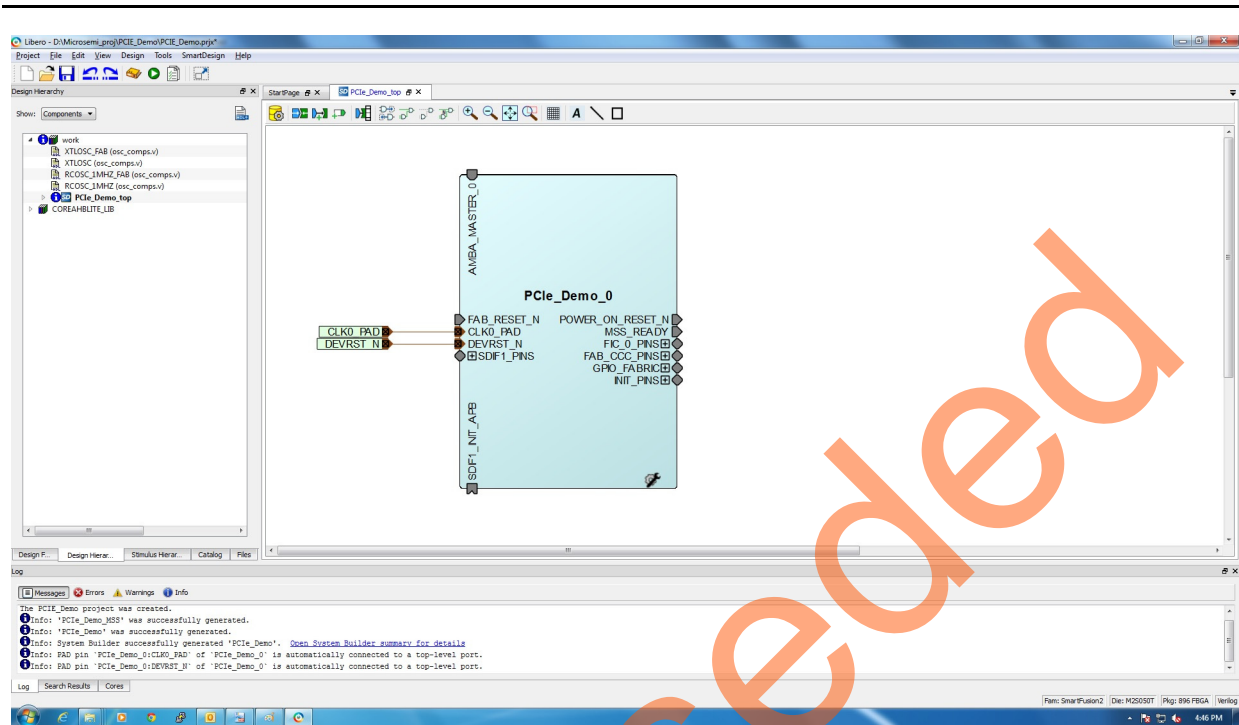


Figure 14 • SmartFusion2 SoC FPGA System Builder Generated System

The two soft cores (CoreResetP and CoreConfigP) will be automatically instantiated and connected by the System Builder. How these blocks are connected can be seen by opening the System Builder component in the SmartDesign canvas.

Note: CoreResetP and CoreConfigP are responsible for the reset and configuration of ASIC peripherals. In this particular demo they are used to reset and configure the SERDESIF module. These modules are included in the System Builder generated component when an ASIC peripheral is selected.

Instantiating the SERDESIF Component in PCIe_Demo_top SmartDesign

The Libero SoC Catalog provides IP cores that can be easily dragged-and-dropped into the SmartDesign Canvas workspace. Many of these IPs are free to use while several require a license agreement. The SERDESIF module that supports the PCIe embedded interface is included in the catalog. To instantiate the SERDESIF component in the **PCIe_Demo_top** SmartDesign, expand the **Peripherals** category in the Libero SoC Catalog.

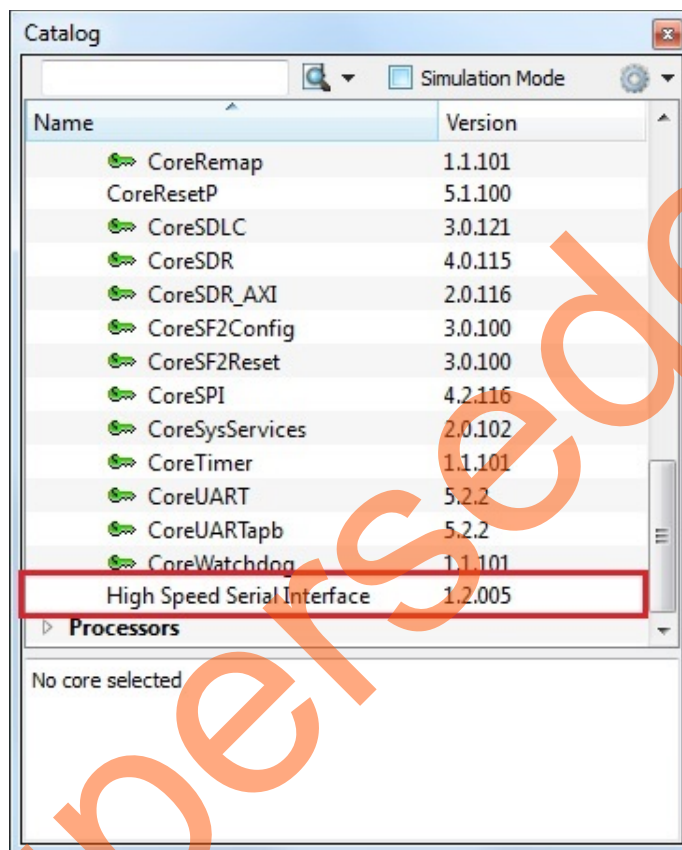
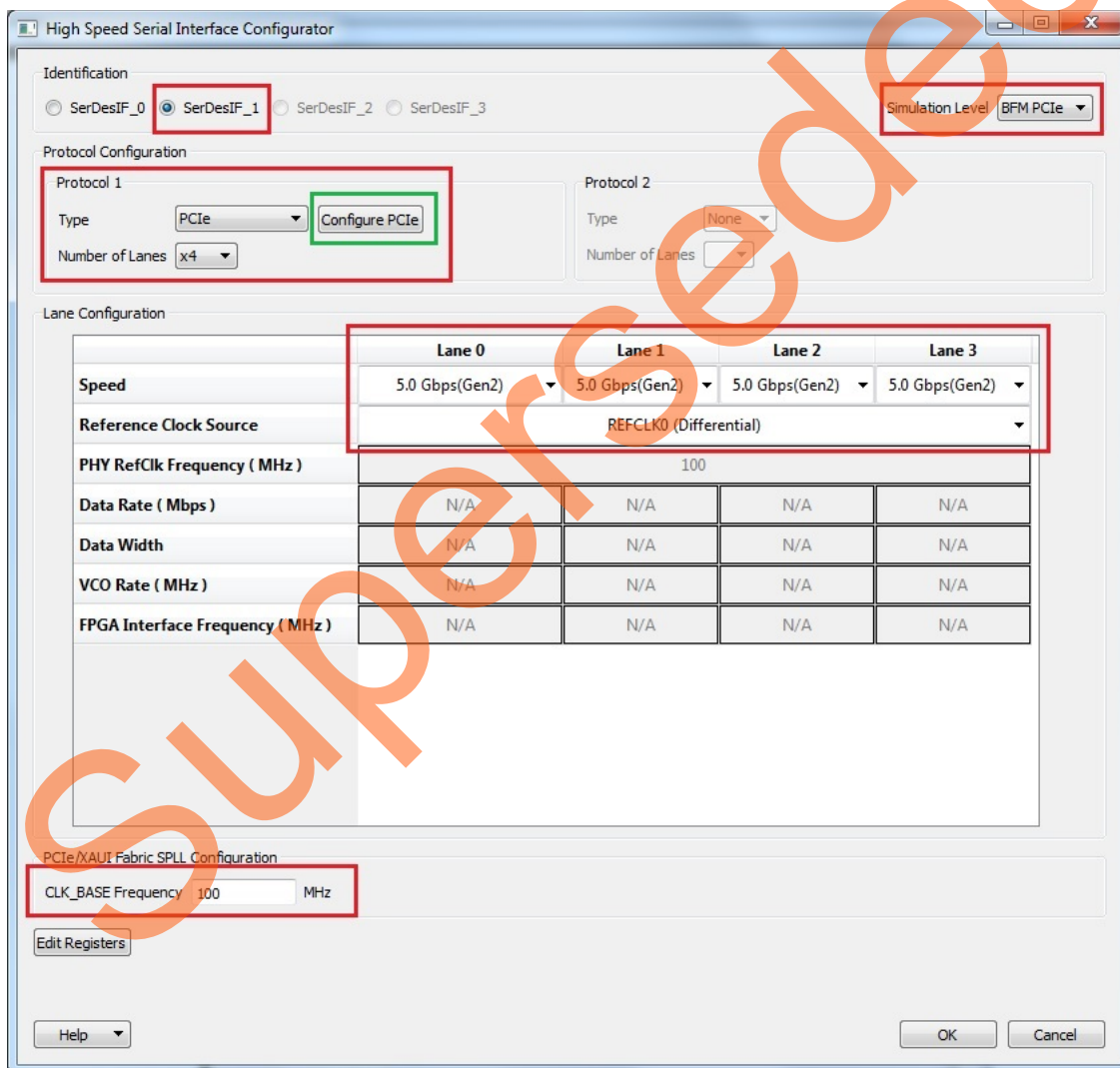


Figure 15 • IP Catalog

1. Drag the **High Speed Serial Interface** onto the **PCIe_Demo_top SmartDesign** canvas. If the component appears shadowed in the Vault, right-click the name and select **Download**.
2. Double-click the **SERDES_IF_0** component in the SmartDesign canvas to open the **SERDES** configurator. Configure the SERDES with the following settings as shown in [Figure 16](#):
 - Select SERDESIF_1
 - Simulation Level: BFM PCIe
 - Protocol1: Number of Lanes: x4
 - Protocol1: Type: PCIe
 - CLK_BASE Frequency (MHz): 100
 - Lane Configuration: Speed: 5.0 Gbps(Gen2)
 - Lane Configuration:
 - Reference Clock Source: REFCLK0 (Differential)



High Speed Serial Interface Configurator

Identification

☐ SerDesIF_0
 ☒ **SerDesIF_1**
☐ SerDesIF_2
 ☐ SerDesIF_3
 Simulation Level: **BFM PCIe**

Protocol Configuration

Protocol 1
 Type: **PCIe** Configure PCIe
 Number of Lanes: **x4**

Protocol 2
 Type: **None**
 Number of Lanes:

Lane Configuration

	Lane 0	Lane 1	Lane 2	Lane 3
Speed	5.0 Gbps(Gen2)	5.0 Gbps(Gen2)	5.0 Gbps(Gen2)	5.0 Gbps(Gen2)
Reference Clock Source	REFCLK0 (Differential)			
PHY RefClk Frequency (MHz)	100			
Data Rate (Mbps)	N/A	N/A	N/A	N/A
Data Width	N/A	N/A	N/A	N/A
VCO Rate (MHz)	N/A	N/A	N/A	N/A
FPGA Interface Frequency (MHz)	N/A	N/A	N/A	N/A

PCIe/XAUI Fabric SPLL Configuration

CLK_BASE Frequency: **100** MHz

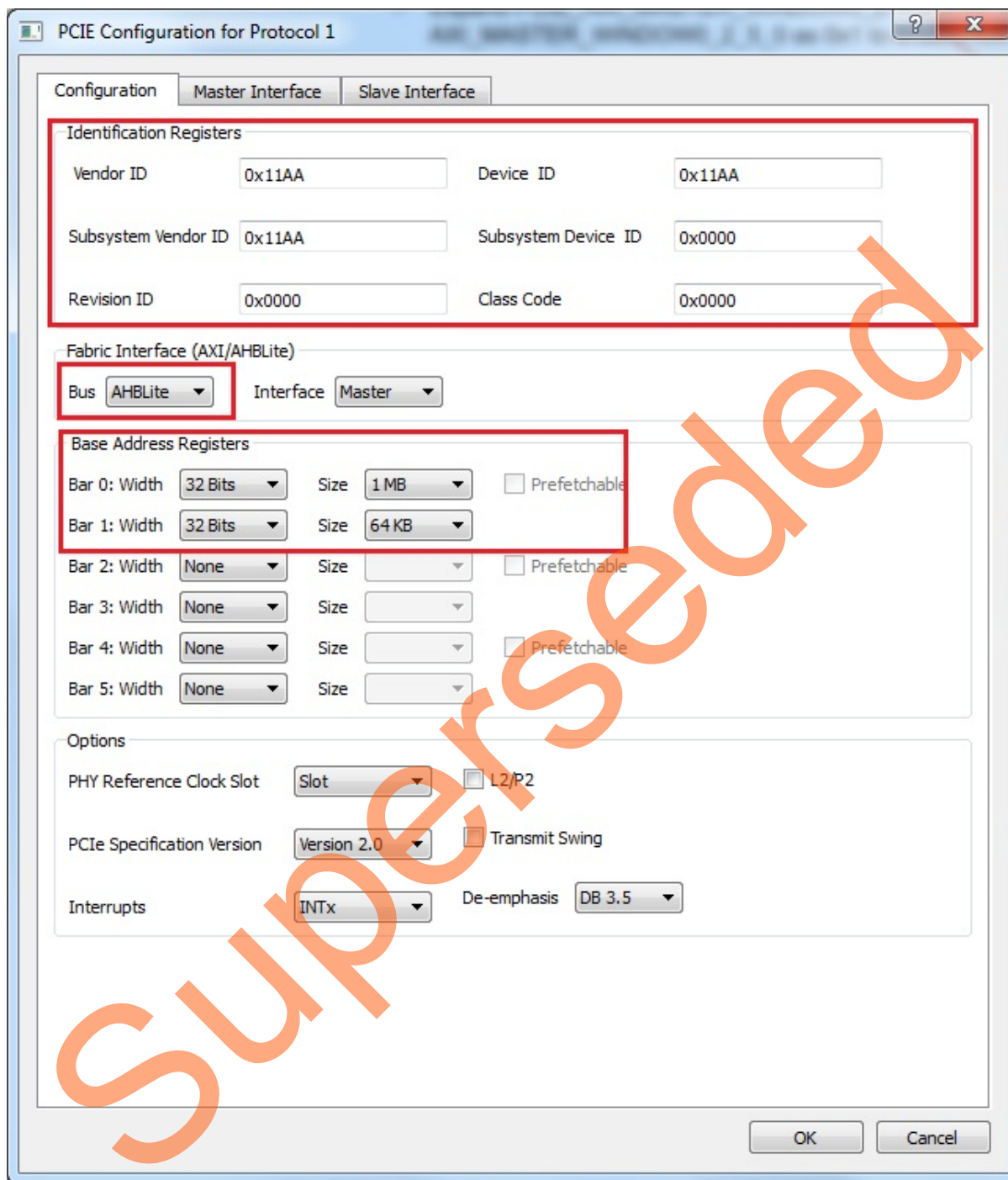
[Edit Registers](#)

Help OK Cancel

Figure 16 • SERDES Configurator

3. Click **Configure PCIe** in Protocol1 as shown in [Figure 16 on page 17](#). Make following settings in the Configuration tab as shown in [Figure 17 on page 19](#).
 - Fabric Interface (AXI/AHBLite)
 - Bus: select as AHBLite from the drop-down list
 - Base Address Registers
 - BAR 0 Width: 32-bit, Size: 1 MB (to access MSS Peripheral address space)
 - BAR 1 Width: 32-bit, Size: 64 KB (to access eSRAM memory)
 - Identification Registers
 - Device ID: 0x11AA (MicroSemi ID)
 - Subsystem Vendor ID: 0x11AA (MicroSemi ID)

Superseded



The image shows a screenshot of the "PCIE Configuration for Protocol 1" dialog box. The dialog has three tabs: "Configuration", "Master Interface", and "Slave Interface". The "Configuration" tab is selected. It contains several sections:

- Identification Registers:** A red box highlights this section, which includes fields for Vendor ID (0x11AA), Device ID (0x11AA), Subsystem Vendor ID (0x11AA), Subsystem Device ID (0x0000), Revision ID (0x0000), and Class Code (0x0000).
- Fabric Interface (AXI/AHBLite):** This section includes a "Bus" dropdown menu set to "AHBLite" and an "Interface" dropdown menu set to "Master". A red box highlights the "Bus" dropdown.
- Base Address Registers:** A red box highlights this section, which includes fields for Bar 0, Bar 1, Bar 2, Bar 3, Bar 4, and Bar 5. Each bar has a "Width" dropdown (set to "32 Bits" for Bar 0 and Bar 1, and "None" for the others) and a "Size" dropdown (set to "1 MB" for Bar 0 and "64 KB" for Bar 1, and empty for the others). There are also checkboxes for "Prefetchable" for each bar.
- Options:** This section includes a "PHY Reference Clock Slot" dropdown (set to "Slot"), a "PCIe Specification Version" dropdown (set to "Version 2.0"), and a "De-emphasis" dropdown (set to "DB 3.5"). There are also checkboxes for "L2/P2" and "Transmit Swing".

The dialog box has "OK" and "Cancel" buttons at the bottom right. A large "Superseded" watermark is visible across the center of the dialog.

Figure 17 • PCIe Configuration for Protocol 1

4. Click the **Master Interface** tab to configure the PCIe master windows. The PCIe AXI master windows are used to translate the PCIe address domain to the local device address domain. In this demo the PCIe AXI master windows are used to translate the address of BAR0 and BAR1 to CoreGPIO address and COREAHBLSRAM address. Make settings as shown in [Figure 18](#).
 - Select Window 0 and configure following settings:
 - Size: Select as 1MB from the drop-down list
 - PCIe BAR: Select as Bar0 from the drop-down list
 - Local Address: Enter values as 0x40000 to translate the BAR0 address space to CoreGPIO address (0x4000_0000)
 - Select Window 1 and configure following settings
 - Size: Select as 64KB from the drop-down list
 - PCIe BAR: Select as Bar1 from the drop-down list
 - Local Address: Enter values as 0x20000 to translate the BAR1 address space to COREAHBLSRAM address (0x2000_0000)

For more information on PCIe address translation, refer to the "Address Translation on the AXI Master Interface" section of the [SmartFusion2 SoC FPGA High Speed Serial Interfaces User Guide](#).

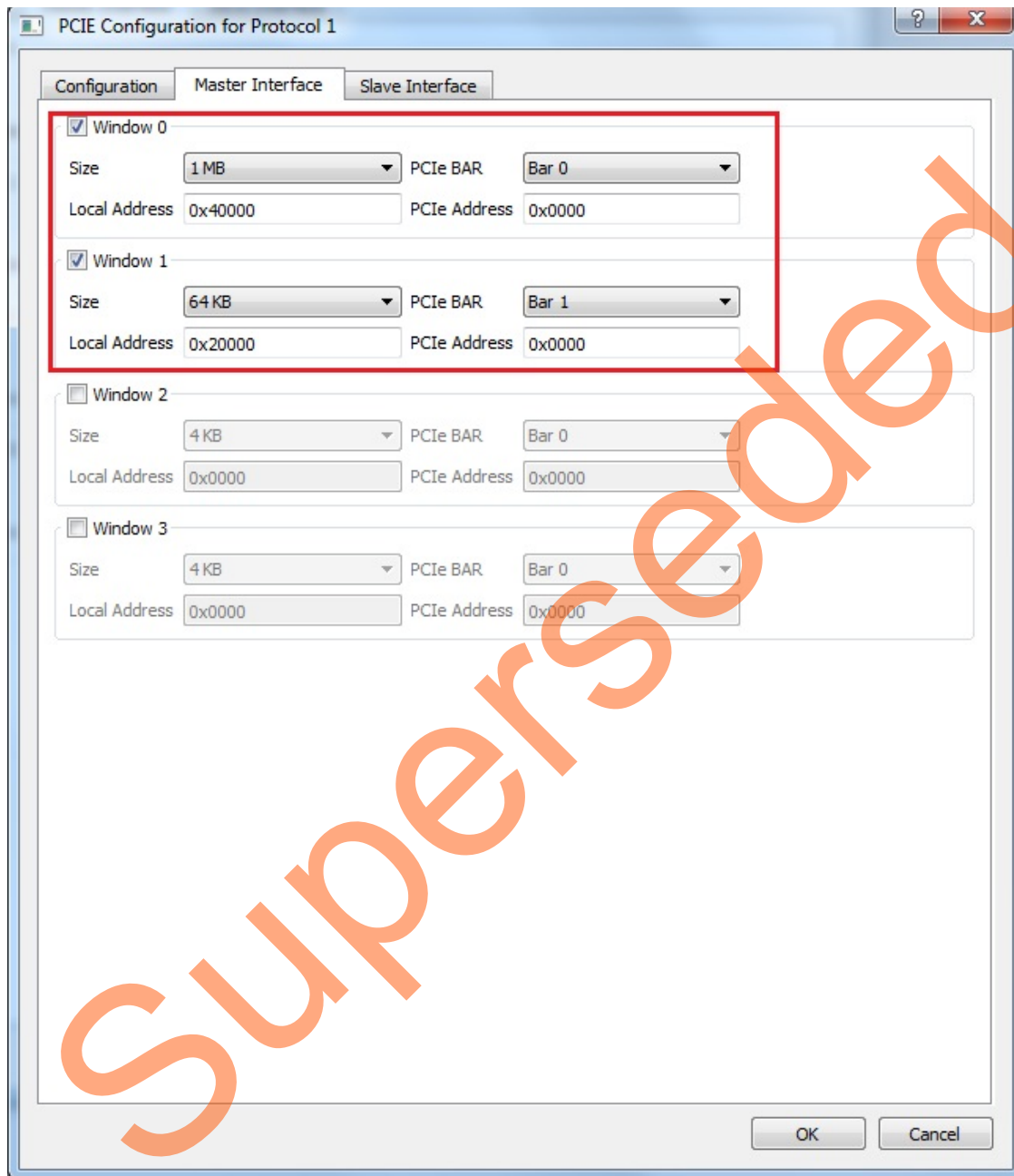


Figure 18 • PCIe Configuration Memory

5. Click **OK** to close PCIe Configuration window.
6. Click **OK** to save and close the High Speed Serial Interface Configurator.

Instantiating Debounce Logic in PCIe_Demo_top SmartDesign

1. The demo provides a push button on the SmartFusion2 Development Kit to send an interrupt to the Host PC. This push button generates switch bounce that causes multiple interrupts to PCIe. Debounce logic is required to avoid the switch bounce.
2. To add the debounce logic to the PCIe demo design, click **File > Import > HDL Source files**.
3. Browse to the Debounce.v or Debounce.vhd file location in the design files folder: *M2S_PCIE_Control_DEMO_DF/Source Files*. [Figure 19](#) shows the DEBOUNCE component in the Design Hierarchy window.

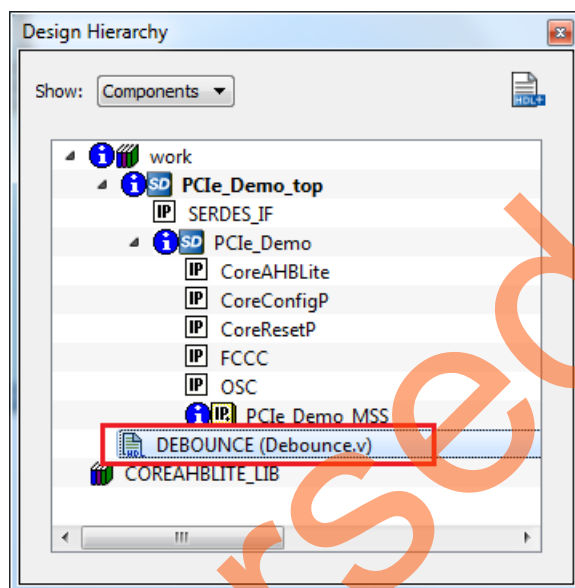


Figure 19 • DEBOUNCE Component in the Design Hierarchy Window

- [illegible]

The PCIe_Demo_top is displayed as shown in [Figure 22 on page 25](#). Connect the pins of all the blocks as described in the ["Connecting Components in PCIe_Demo_top SmartDesign"](#) section.

There are three methods for connecting components in PCIe_Demo_top SmartDesign.

The first method is by using the **Connection Mode** option. To use this method, change the SmartDesign to connection mode by clicking **Connection Mode** on the SmartDesign window, as shown in [Figure 22 on page 25](#). The cursor changes from the normal arrow shape to the connection mode icon shape. To make a connection in this mode, click on the first pin and drag-drop to the second pin that you want to connect.

The second method is by selecting the pins to be connected together and selecting **Connect** from the context menu. To select multiple pins to be connected together, press down the **CTRL** key while selecting the pins. Right-click the input source signal and select **Connect** to connect all the signals together. Similarly, select the input source signal, right-click it, and select **Disconnect** to disconnect the signals already connected.

The third method is by using the Quick Connect option. To use this method, change the SmartDesign to quick connect mode by clicking on Quick Connect mode on the SmartDesign window, as shown in Figure 21. Quick connect window will be opened. Find the Instance Pin you want to connect and click to select it. In Pins to Connect, find the pin you wish to connect, right-click and choose Connect as shown in Figure 21.

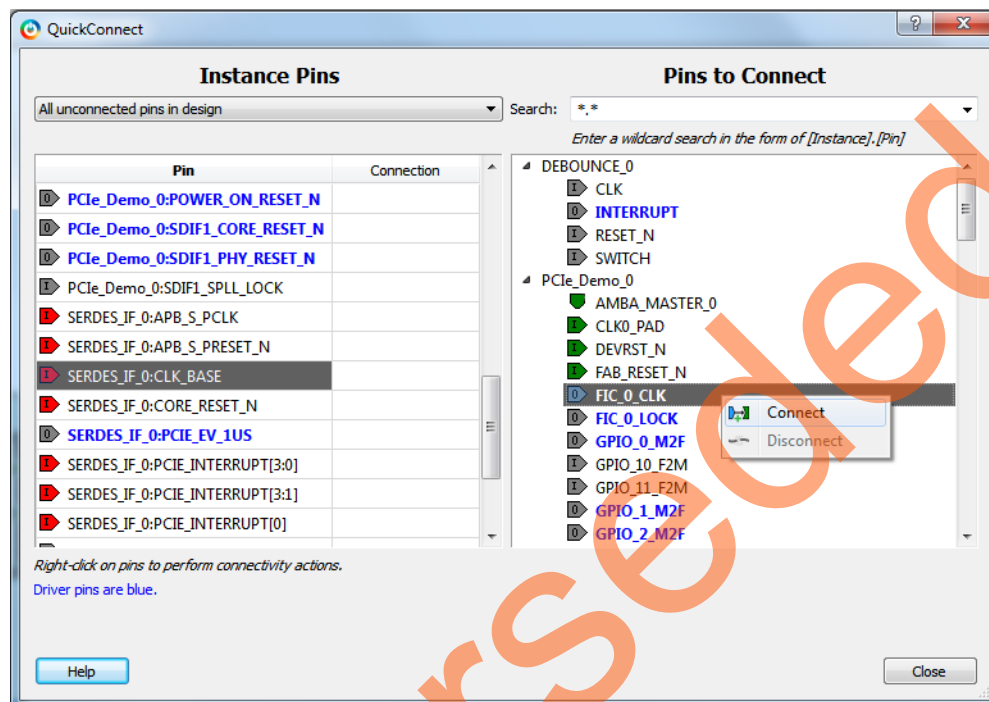


Figure 21 • Quick Connect Window

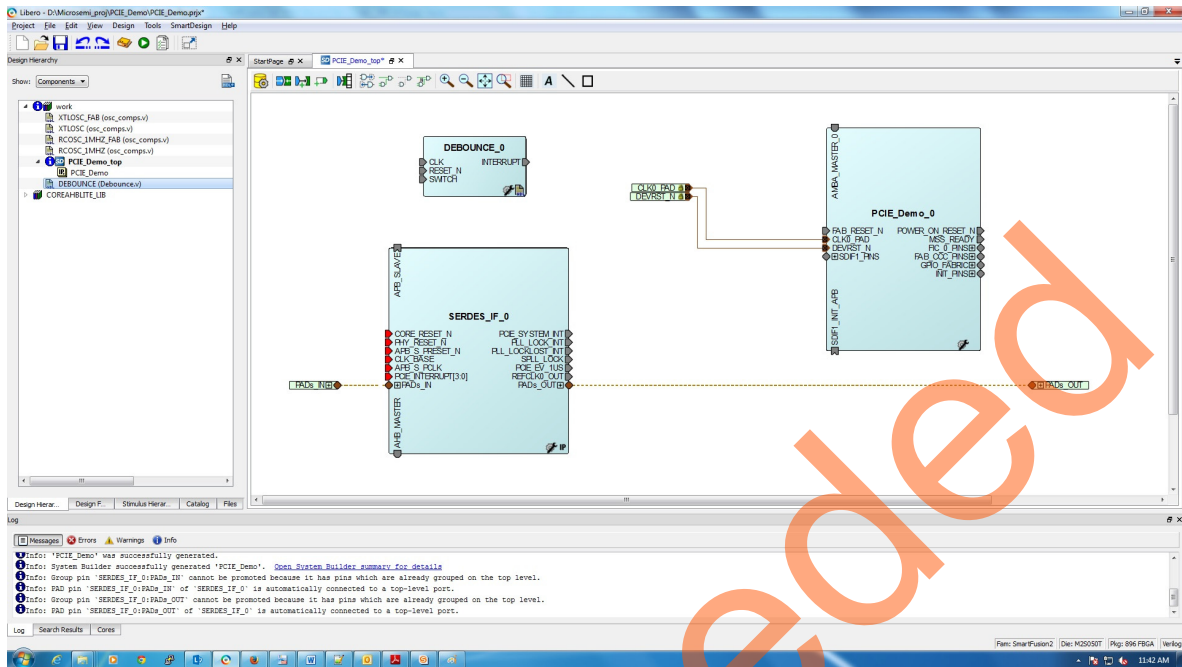


Figure 22 • PCIe Demo Top in SmartDesign

Use one of the three options described above and make the following connections:

1. Expand FIC_0_PINS of PCIe_Demo_0 and make connections as shown in Table 2.
2. Right click **FIC_0_LOCK** and select **mark unused**

Table 2 • FIC_0_PINS

From PCIe_Demo_0	To
FIC_0_CLK	CLK_BASE of SERDES_IF_0
	CLK of DEBOUNCE_0

3. Expand SDIF1_PINS of PCIe_Demo_0 and make connections as shown in Table 3.

Table 3 • SDIF1_PINS

From PCIe_Demo_0	To SERDES_IF_0
SDIF1_PHY_RESET_N	PHY_RESET_N
SDIF1_CORE_RESET_N	CORE_RESET_N
SDIF1_SPILL_LOCK	SPILL_LOCK

4. Right click **SDIF1_PERST_N** and select **Promote to Top Level**.
5. Expand INIT_PINS of PCIe_Demo_0 and make connections as shown in Table 4.

Table 4 • INIT_PINS

From PCIe_Demo_0	To SERDES_IF_0
INIT_APB_S_PCLK	APB_S_PCLK
INIT_APB_S_PRESET_N	APB_S_PRESET_N

6. Right click **INIT_DONE** and select **mark unused**.
7. Connect **MSS_READY** of **PCle_Demo_0** and **RESET_N** of **DEBOUNCE_0**.
8. Right-click the **FAB_RESET_N** of **PCle_Demo_0** and select **Tie High**.
9. Right-click the **GPIO_FABRIC** of **PCle_Demo_0** and select **Promote to Top Level**.
10. Right click **POWER_ON_RESET_N** of **PCle_Demo_0** and select **Mark Unused**.
11. Connect **AMBA_MASTER_0** of **PCle_Demo_0** and **AHB_MASTER** of **SERDES_IF_0**.
12. Expand **FAB_CCC_PINS**, right-click **FAB_CCC_GL3** and select **Mark Unused**.
13. Connect **SDIF1_INIT_APB** of **PCle_Demo_0** and **APB_SLAVE** of **SERDES_IF_0**.
14. Right-click the **SWITCH** of **DEBOUNCE_0** and select **Promote to Top Level**.
15. Select the below ports of **SERDES_IF_0** by pressing down the **CTRL** key, right-click, and select **Mark Unused**.
 - **PCIE_SYSTEM_INT**
 - **PLL_LOCK_INT**
 - **PLL_LOCKLOST_INT**
 - **PCIE_EV_1US**
 - **REFCLK0_OUT**

The PCIe supports four interrupts. This design uses only one interrupt out of four by connecting the unused interrupts to logic '0'. To connect unused interrupt pins to logic '0' split the interrupt pins to two groups. To do that right-click the **PCIE_INTERRUPT[3:0]** of **SERDES_IF_0** and select **Edit Slice**. The Edit Slice window is displayed as shown in Figure 23 on page 26.

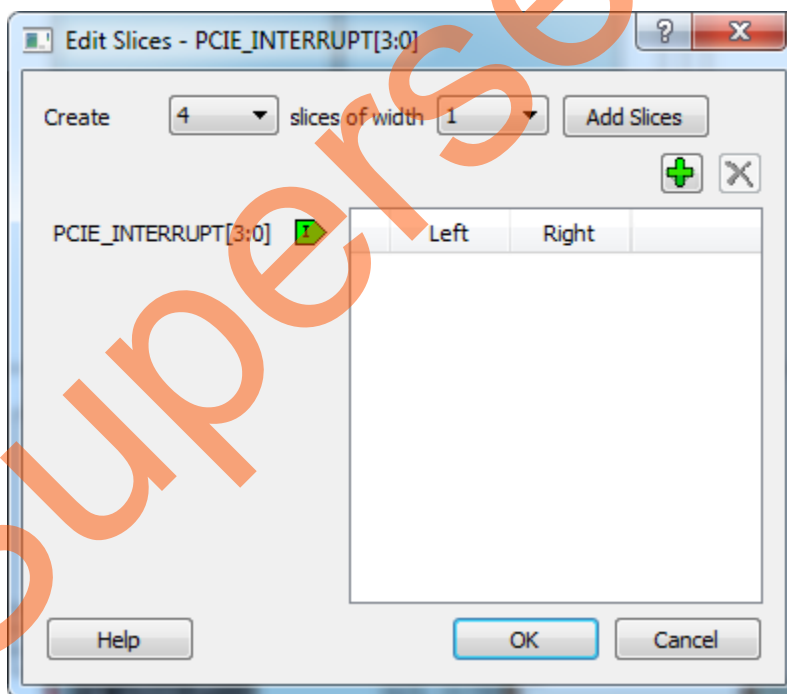


Figure 23 • Edit Slices

16. Click the + sign and create a slice with the Left index 0 and the Right index 0. Click + again to create a second slice with Left index 3 and Right index 1 as shown in [Figure 24](#).

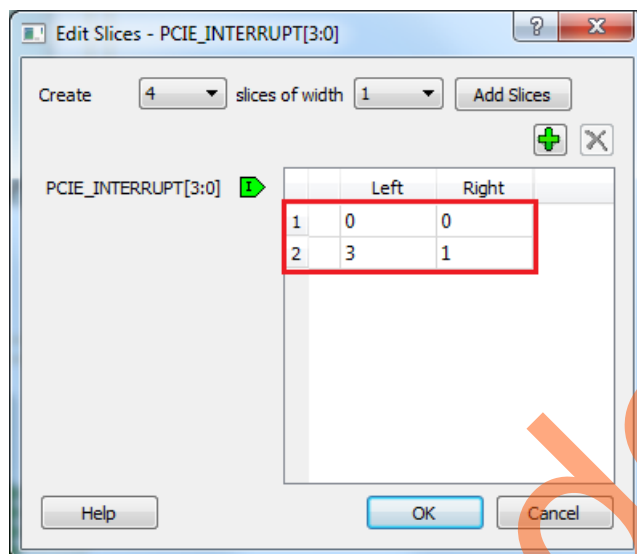


Figure 24 • Edit Slices

17. Expand **PCIE_INTERRUPT[3:0]**, right-click the **PCIE_INTERRUPT[3:1]**, and select **Tie low**.
18. Connect **INTERRUPT** of **DEBOUNCE_0** to the **PCIE_INTERRUPT[0]** of **SERDES_IF_0**.
19. Click **Auto arrange instances** to arrange the instances and click **File > Save**. The PCIe_Demo_top is displayed as shown in [Figure 25](#).

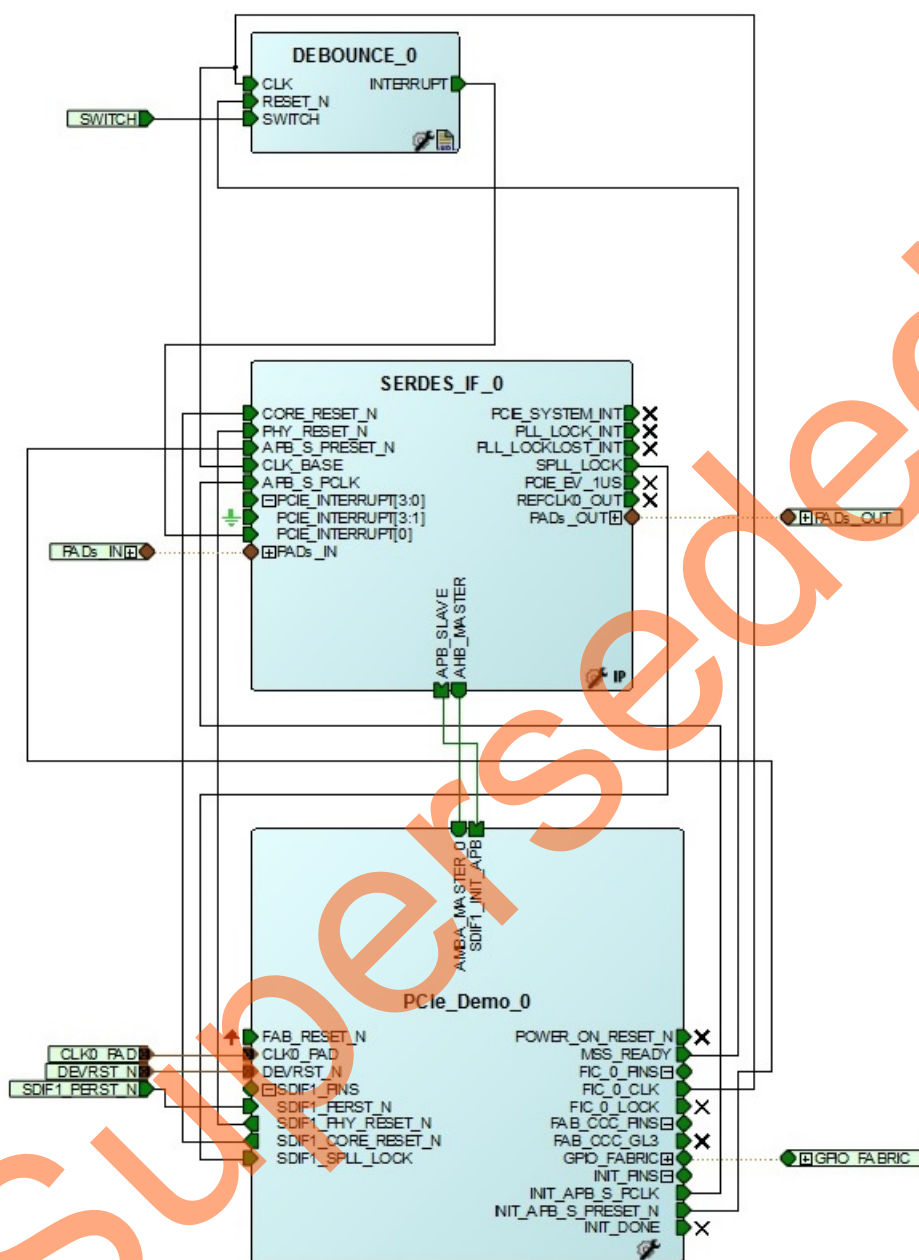
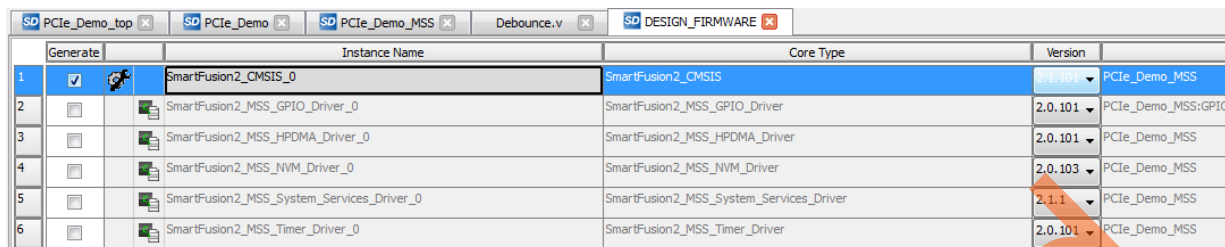


Figure 25 • PCle Demo Top Design

Configuring and Generating Firmware

1. Click **Design > Configure Firmware** and clear all drivers except CMSIS as shown in Figure 26.



	Generate	Instance Name	Core Type	Version	
1	<input checked="" type="checkbox"/>	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.0.101	PCIE_Demo_MSS
2	<input type="checkbox"/>	SmartFusion2_MSS_GPIO_Driver_0	SmartFusion2_MSS_GPIO_Driver	2.0.101	PCIE_Demo_MSS:GPIO
3	<input type="checkbox"/>	SmartFusion2_MSS_HPDM_A_Driver_0	SmartFusion2_MSS_HPDM_A_Driver	2.0.101	PCIE_Demo_MSS
4	<input type="checkbox"/>	SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.0.103	PCIE_Demo_MSS
5	<input type="checkbox"/>	SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.1.1	PCIE_Demo_MSS
6	<input type="checkbox"/>	SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.0.101	PCIE_Demo_MSS

Figure 26 • Configuring Firmware

2. Open the **PCIE_Demo_top** tab and click **Generate Component** as shown in Figure 27.

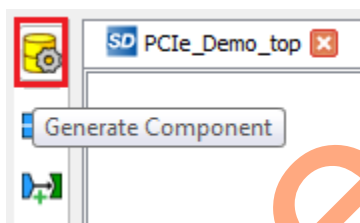


Figure 27 • Generate Component

3. The message “'PCIE_Demo_top' was successfully generated” is displayed in the Libero SoC log window if the design was generated without any errors. Libero also generates the SoftConsole software project. The log window is displayed as shown in Figure 28 on successful component generation.

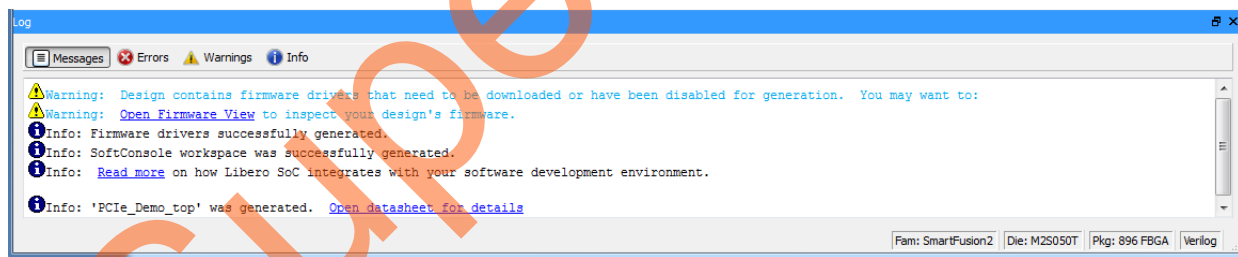


Figure 28 • Log Window

Step 2: Creating an eNVM Client

The HDL and logical design portion of the demo is now complete. The following sections describe the creation of the Cortex-M3 firmware used to initialize the MSS and SERDESIF.

The eNVM client has to be uploaded with the firmware application to initialize the SERDESIF through **CoreConfigP**. The Cortex-M3 processor executes the code in the eNVM after the SmartFusion2 device has been 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, click **Write Application Code** in the Design Flow window as shown in Figure 29.

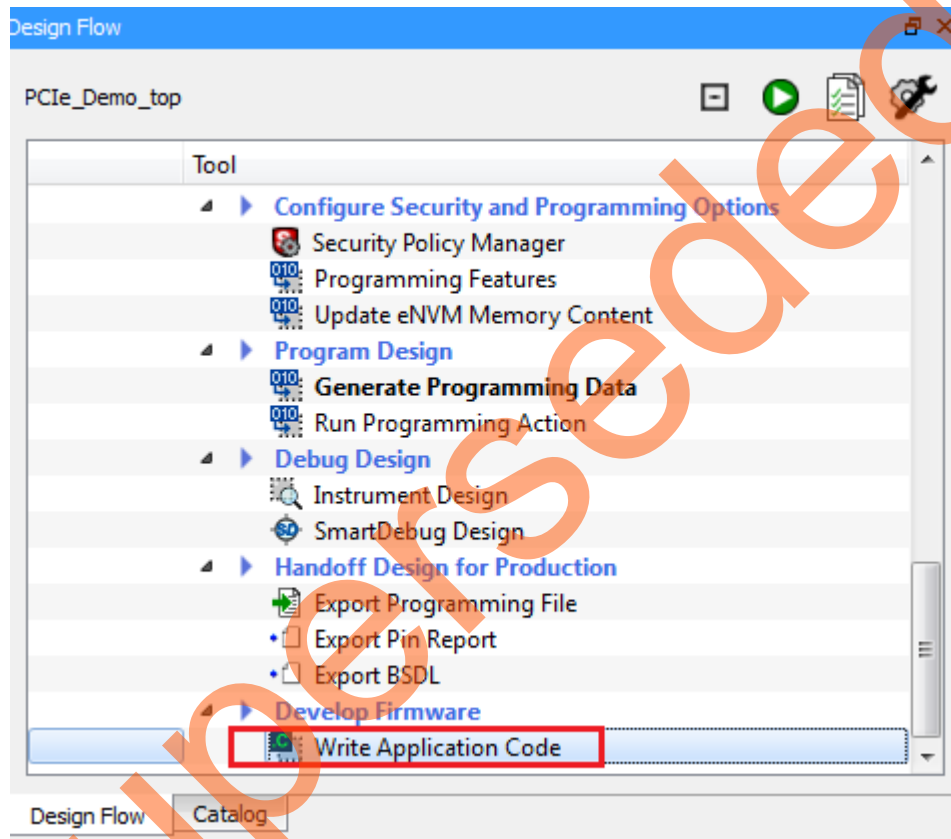


Figure 29 • Invoking a SoftConsole Project

2. The SoftConsole IDE is invoked and the SoftConsole project workspace is displayed as shown in Figure 30.

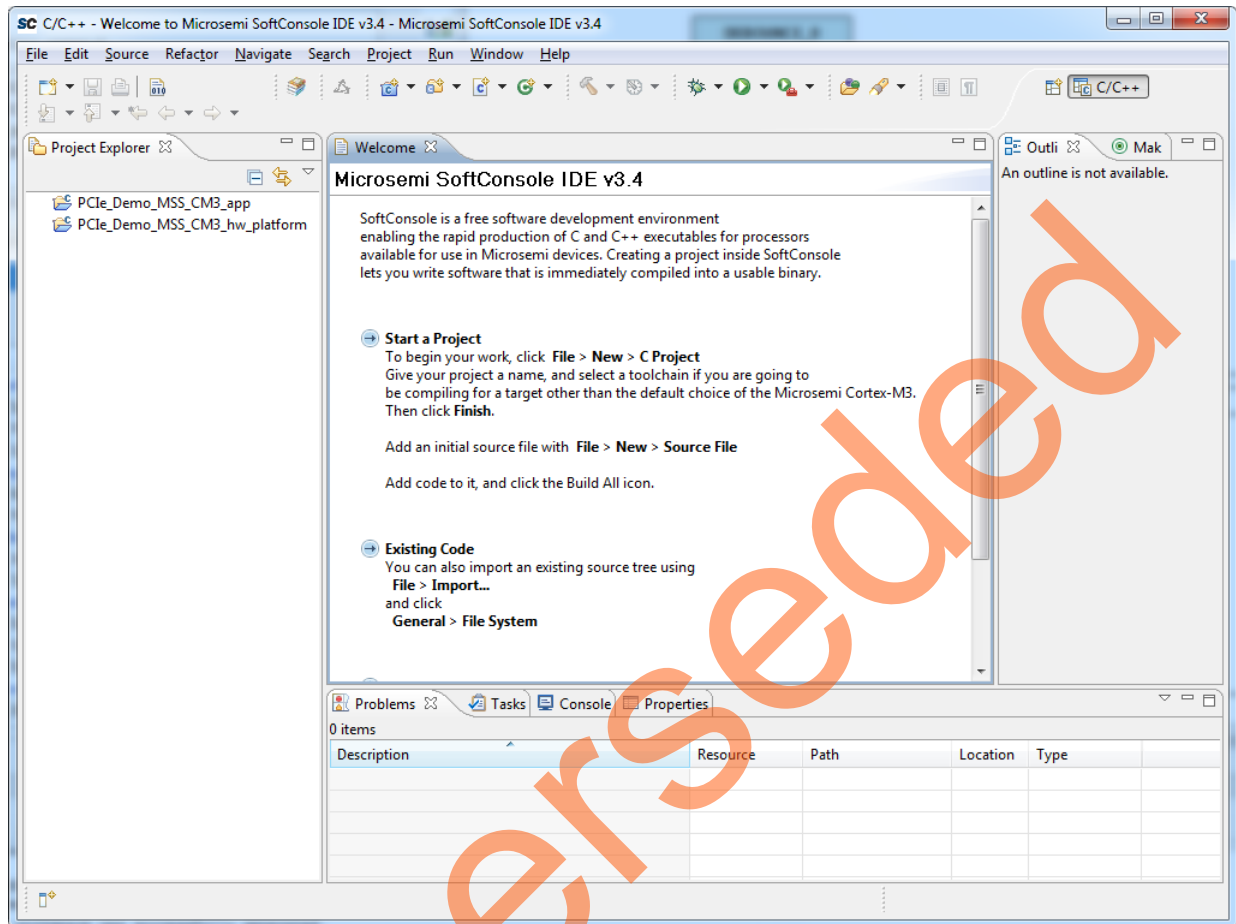


Figure 30 • SoftConsole Workspace

3. Select the projects **PCIe_Demo_MSS_CM3_app** and **PCIe_Demo_MSS_CM3_hw_platform** in the Project Explorer by pressing down the **CTRL** key.

- Right-click and select **Build Configurations > Set Active > Release** as shown in Figure 31.

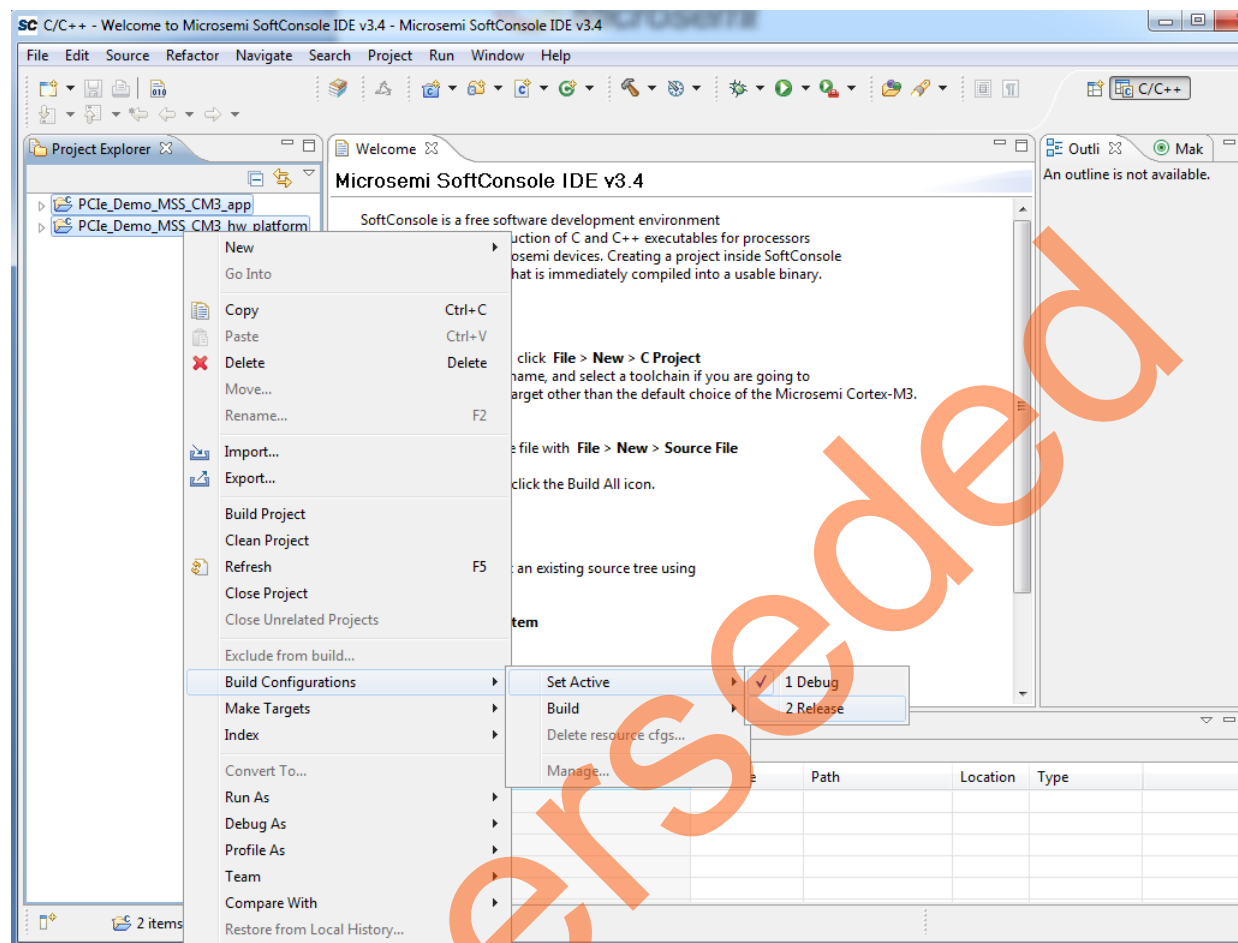


Figure 31 • Release Mode Option

5. Select **PCle_Demo_MSS_CM3_app**. Right-click and select **Properties** as shown in Figure 32.

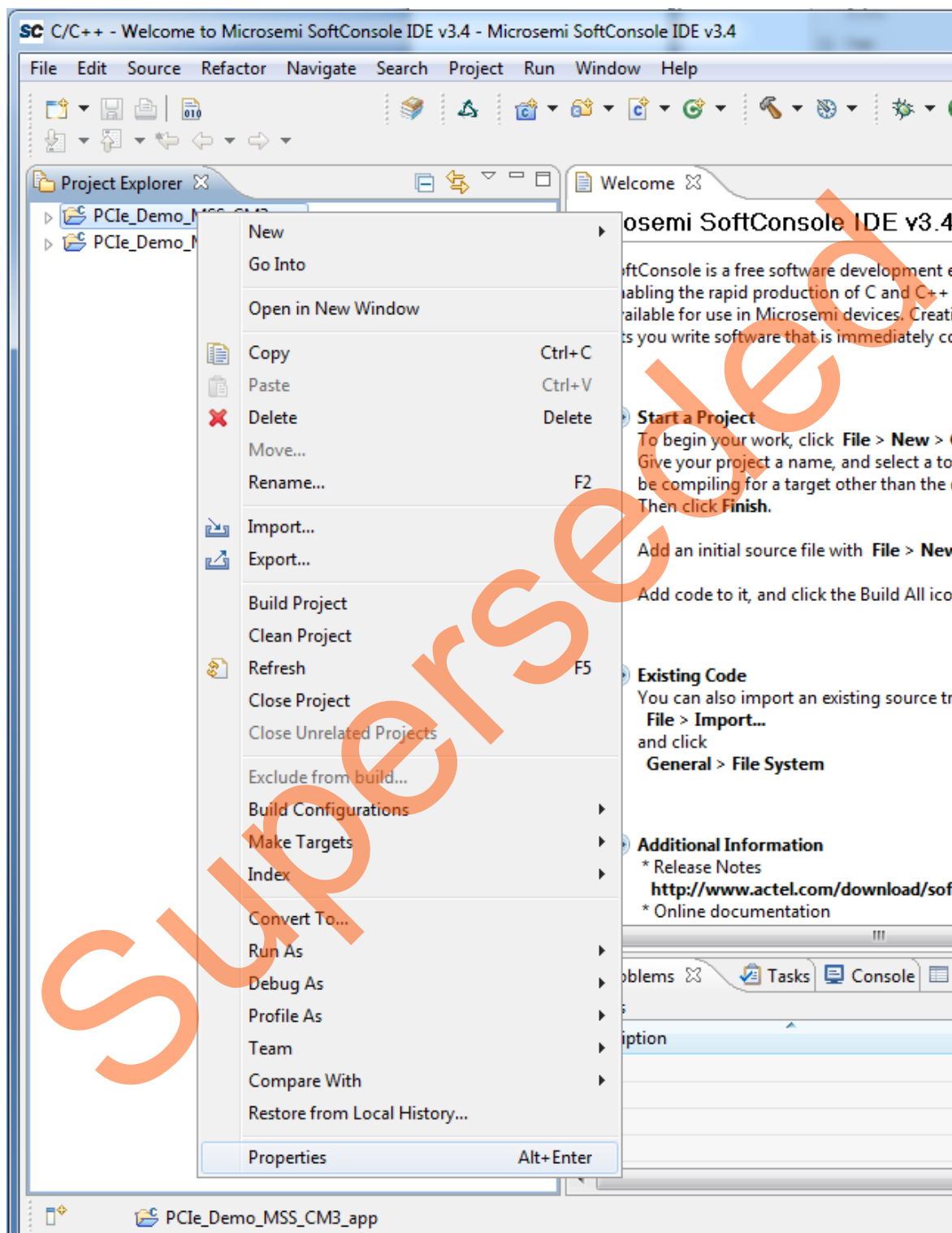


Figure 32 • Properties Option

6. The **Properties for PCIe_Demo_MSS_CM3_app** window is displayed as shown in Figure 33.

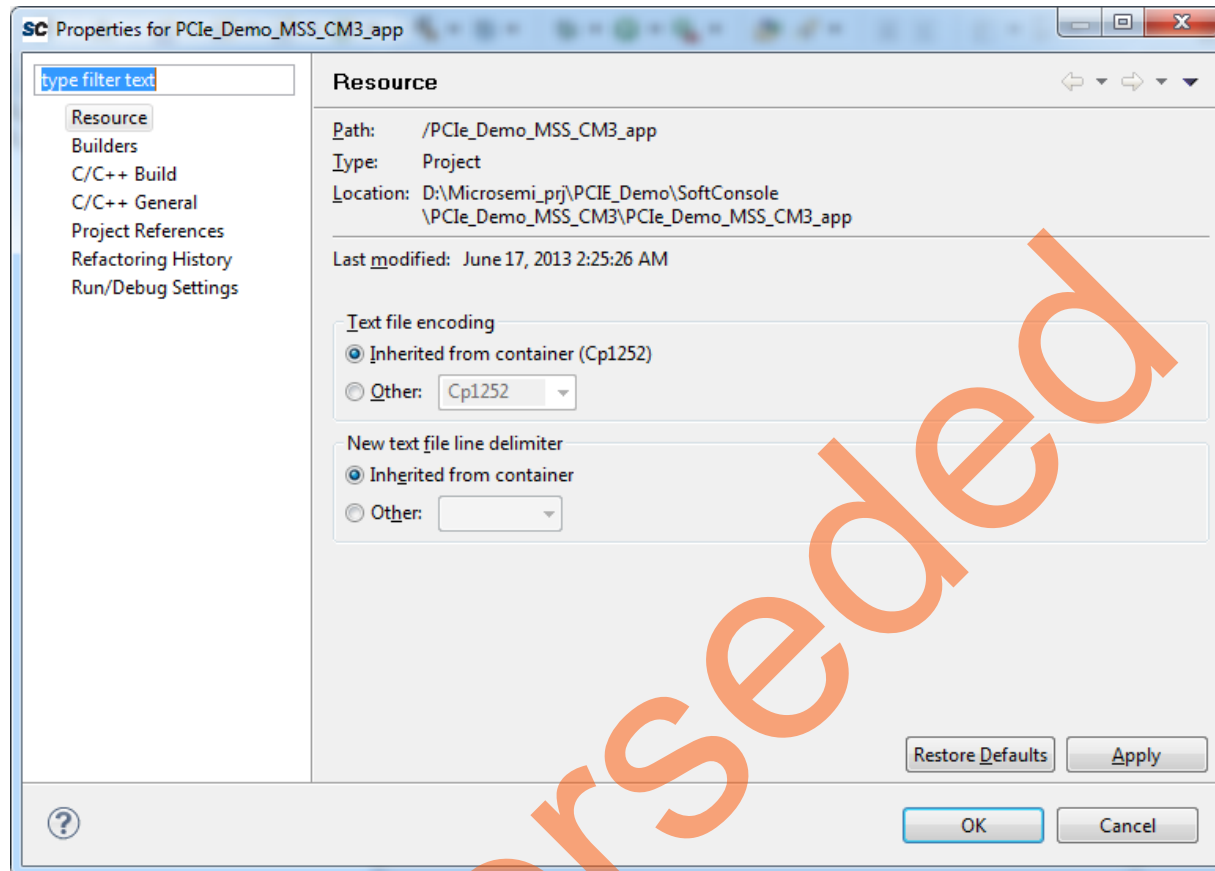


Figure 33 • Properties Window

7. In the **Properties for PCIe_Demo_MSS_CM3_app** window, expand the **C/C++ Build** option 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././PCle_Demo_MSS_CM3_hw_platform/CMSIS/startup_gcc/production-execute-in-place.ld”` as shown in Figure 34.

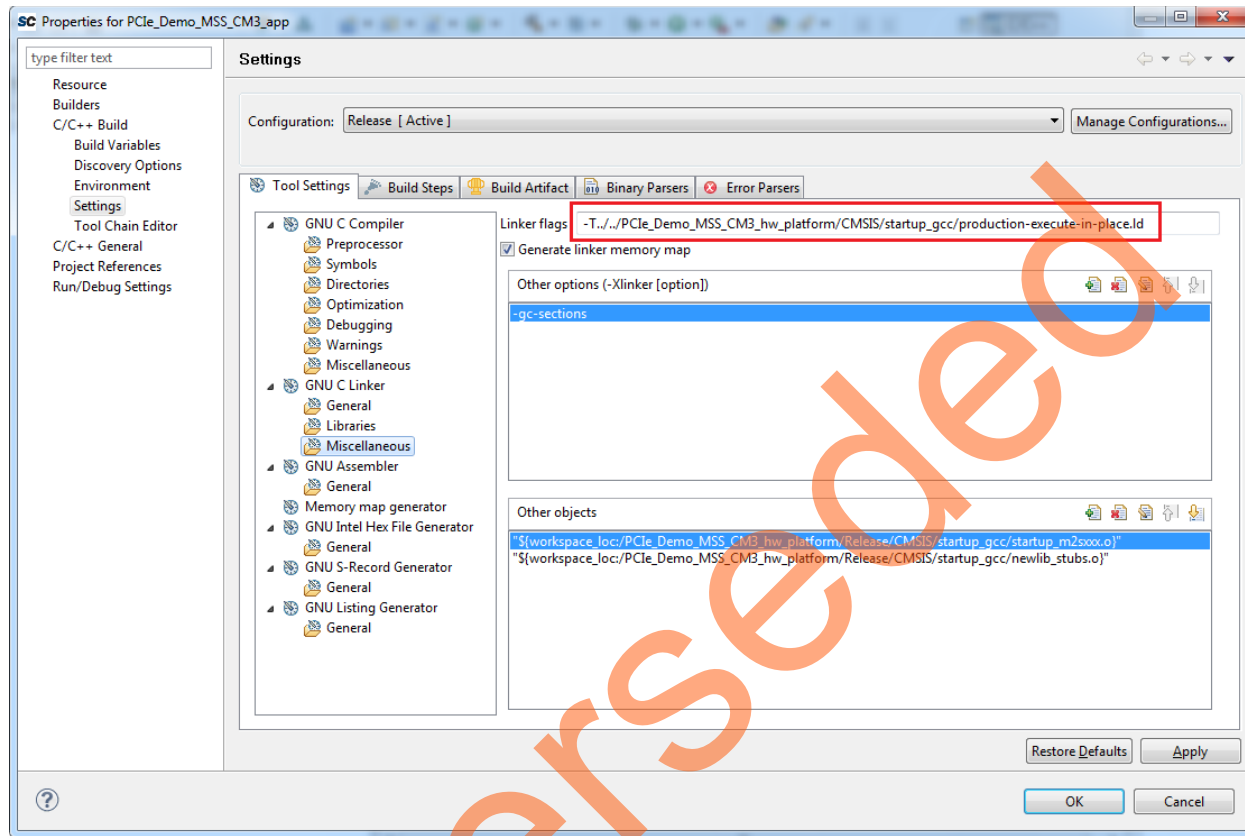


Figure 34 • LD File Option

9. Click **OK** to close the **Properties for PCle_Demo_MSS_CM3_app** window.
10. To clean and build the project, select **Project > Clean** as shown in Figure 35.

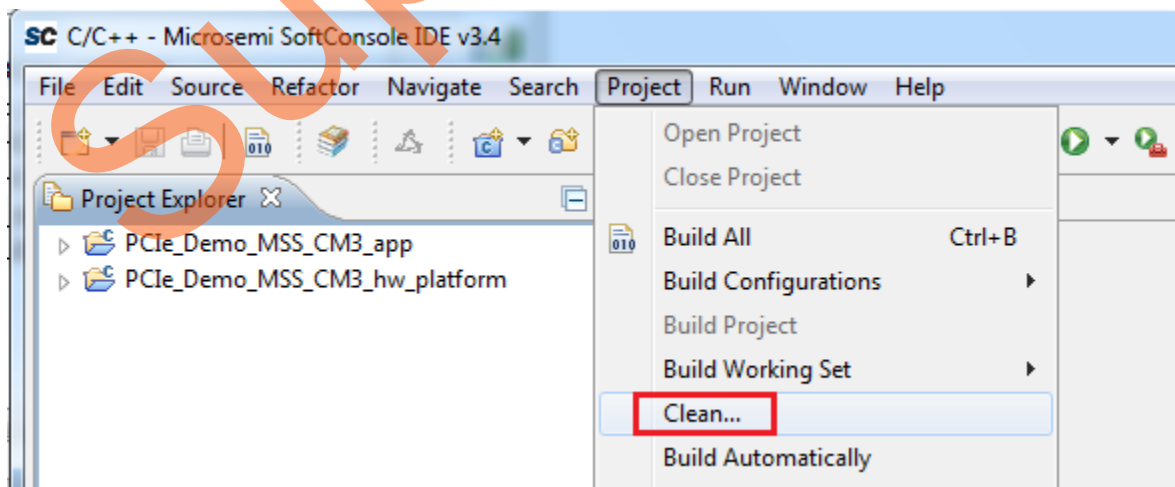


Figure 35 • Building the SoftConsole Project

11. The **Clean** window is displayed. Click **OK** to build the SoftConsole projects as shown in Figure 36.

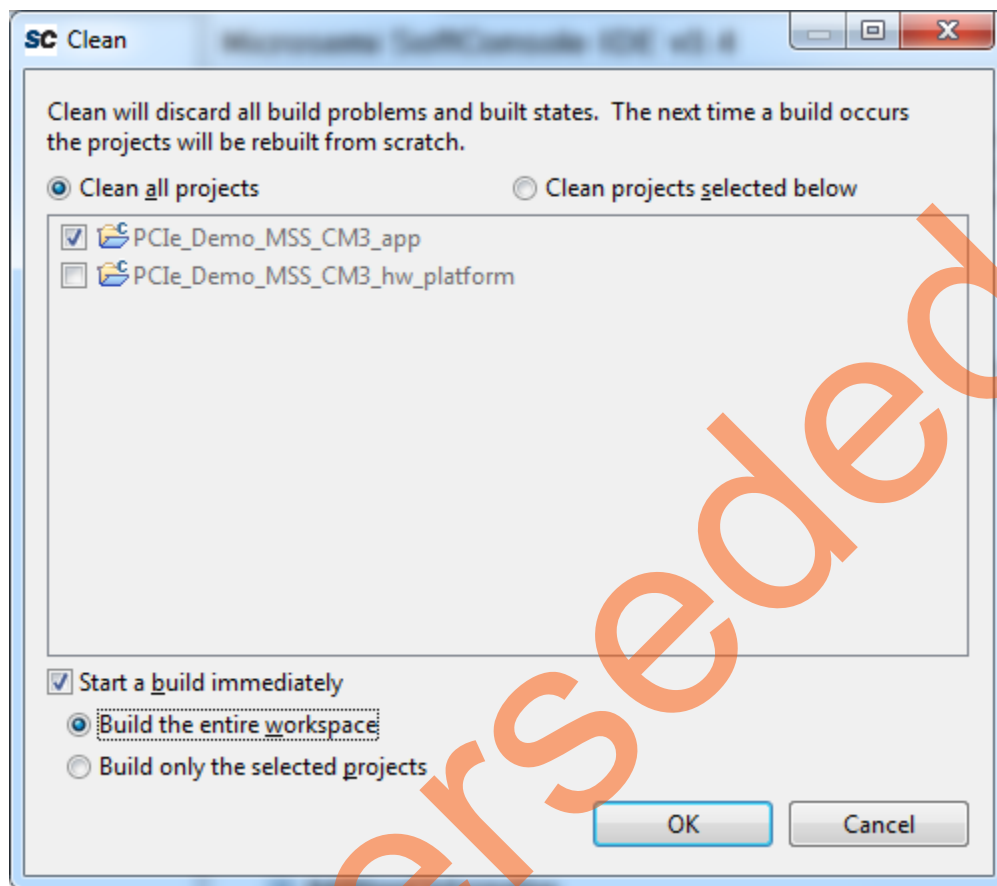


Figure 36 • Clean and Build SoftConsole Projects

12. The SoftConsole creates a hex file in the **Release** folder under the **PCle_Demo_MSS_CM3_app** project as shown in Figure 37.

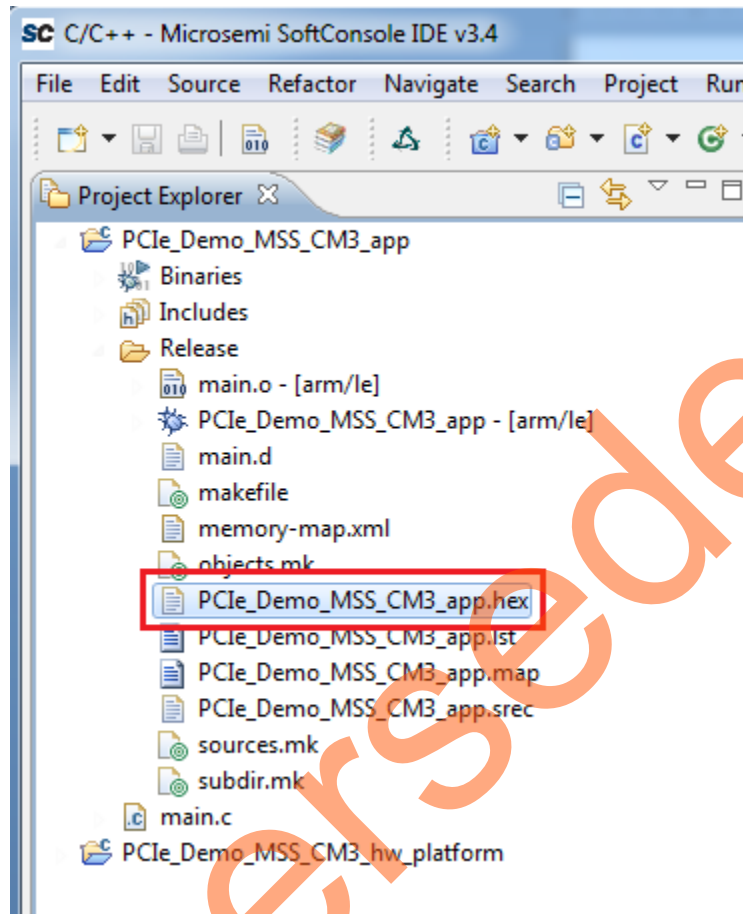


Figure 37 • Generated Hex File

13. Close the SoftConsole project window.
14. Open the Libero project and **PCle_Demo_top** tab. Double-click **PCle_Demo_0** and go to **System Builder - Memories** tab to add the eNVM data storage client.

15. The eNVM configurator window is displayed as shown in Figure 38.

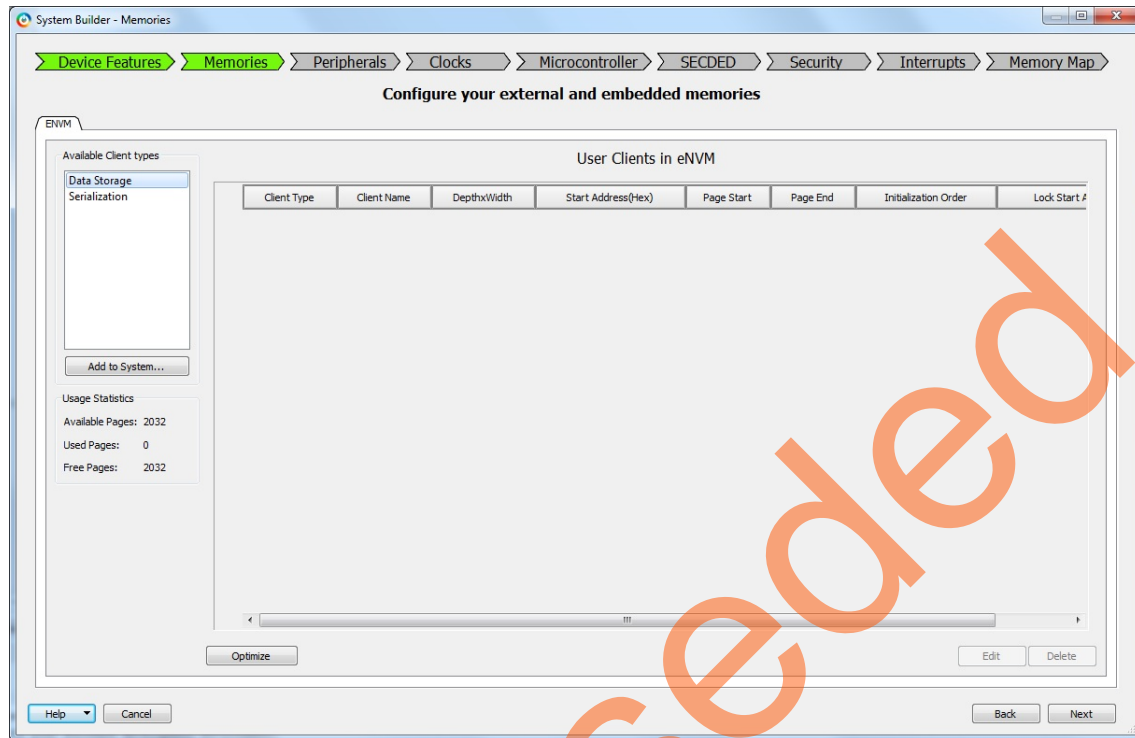


Figure 38 • System Builder - Memory eNVM

16. Select **Data Storage** under the **Available client types** tab and click **Add to System**. The **Add Data Storage Client** window is displayed as shown in [Figure 39](#).

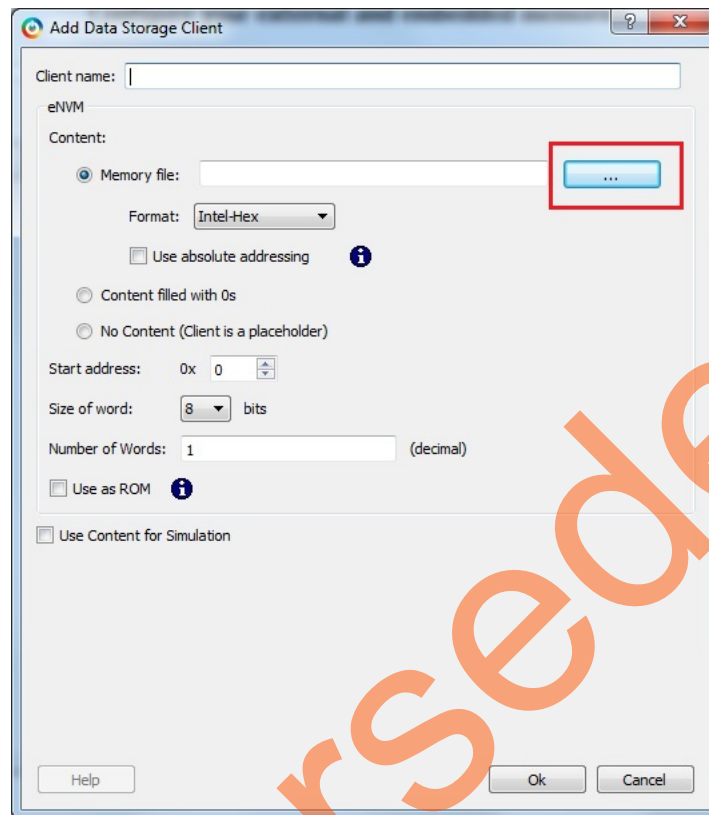


Figure 39 • Add Data Storage Client

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

18. Browse for the .hex file generated (as shown in [Figure 37](#) on [page 37](#)). The generated executable image can be found in the **Release** folder under the SoftConsole project workspace as shown in [Figure 40](#).

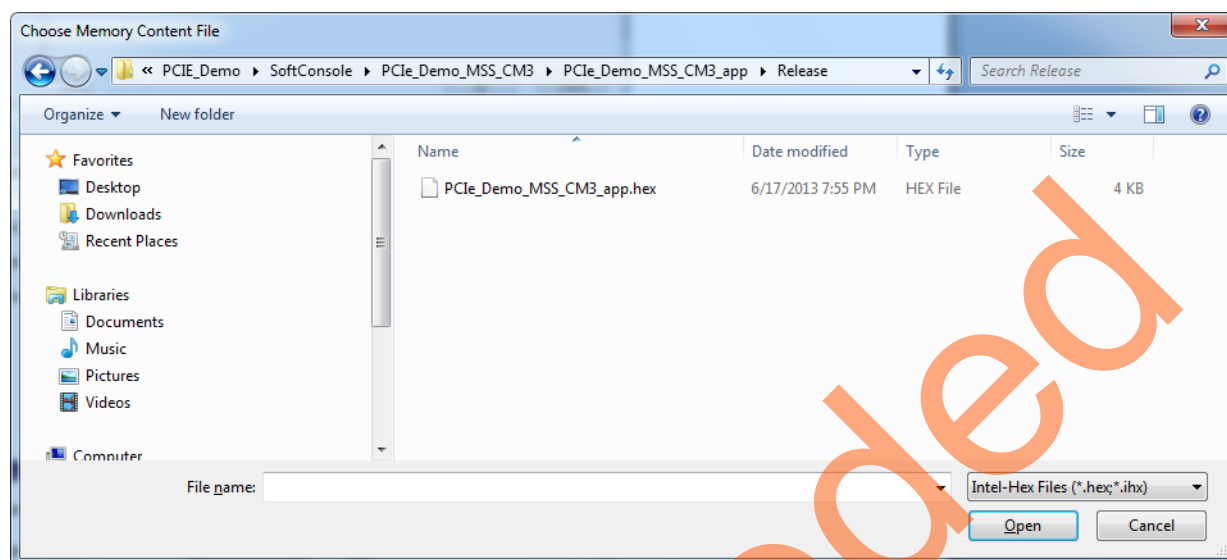


Figure 40 • Browsing for .hex File

19. Click **OK** in the **Add Data Storage Client** window as shown in [Figure 41](#).

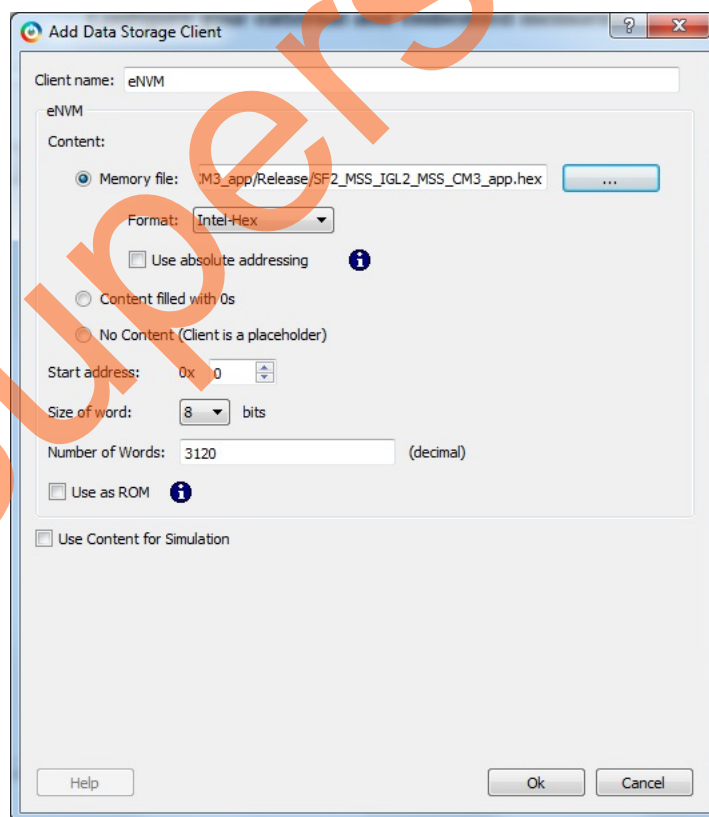


Figure 41 • Add Data Storage Client

20. Click **Next** and keep the rest of the System Builder tabs as default.

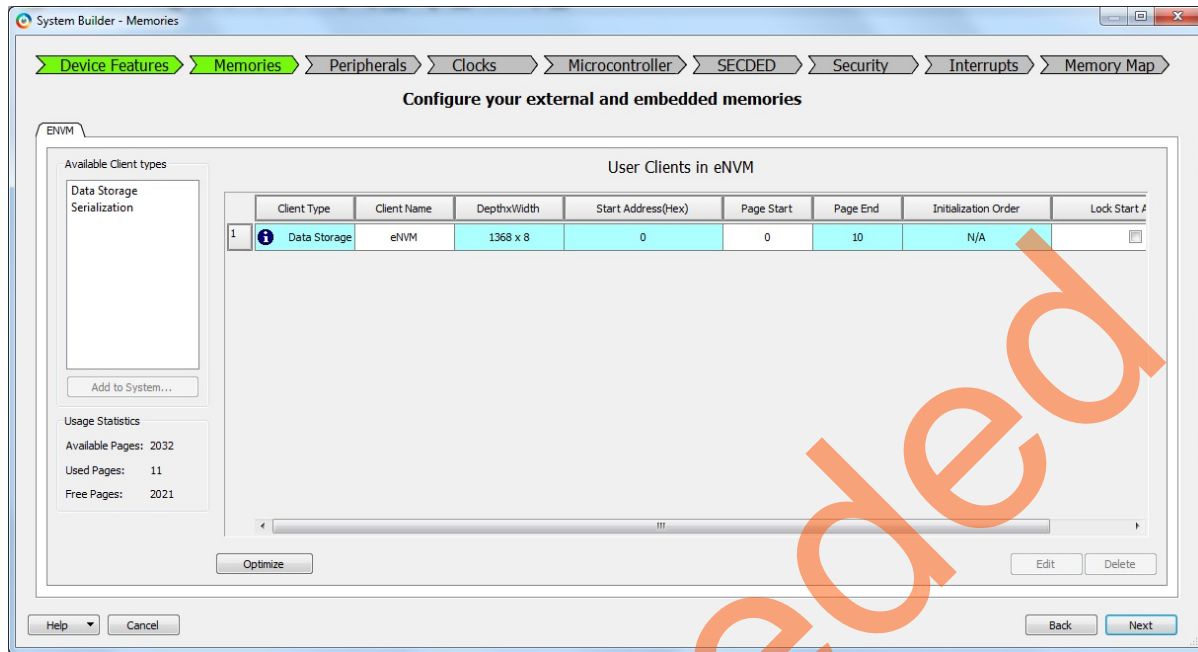


Figure 42 • Modify Core - ENVM

21. Save **PCle_Demo_top** and regenerate the **PCle_Demo_top** component by clicking **Generate Component** in SmartDesign.

Step 3: Developing the Simulation Stimulus

During the design process, SERDESIF was configured for the BFM simulation model. The BFM simulation model replaces the entire PCIe interface with a simple BFM that can send write transactions and read transactions over the AHB-Lite interface. These transactions are driven by a file and allow easy simulation of the FPGA design connected to a PCIe interface. This simulation methodology has the benefit of focusing on the FPGA design since the SmartFusion2 PCIe interface is a fully hardened and verified interface.

This section describes how to modify the BFM script (user.bfm) file that was generated by SmartDesign. The BFM script file simulates PCIe writing/reading to/from the MSS through the FIC_0.

1. Open the SERDESIF_1_user.bfm file. To open the SERDESIF_1_user.bfm, go to the **Files** tab > **Simulation** folder, and double-click the SERDESIF_1_user.bfm. The SERDESIF_1_user.bfm file is displayed, as shown in Figure 43.

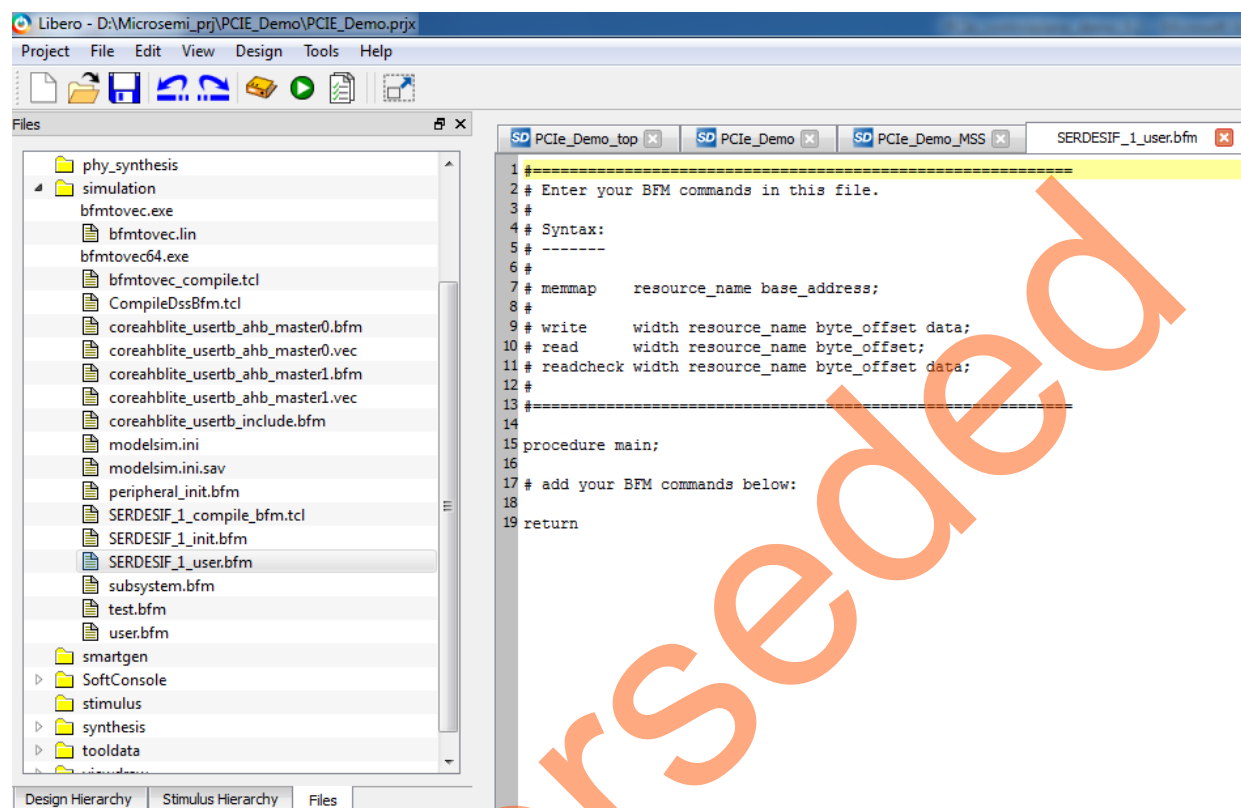


Figure 43 • SmartDesign Generated SERDESIF_1_user.bfm File

2. Modify the SERDESIF_1_user.bfm to add the following bfm commands of writing and reading:

```

memmap GPIO 0x40013000;
memmap eSRAM 0x20000000;
procedure main;

# add your BFM commands below:

write w GPIO 0x00 0x5;
write w GPIO 0x04 0x5;
write w GPIO 0x08 0x5;
write w GPIO 0x0C 0x5;
write w GPIO 0x10 0x5;
write w GPIO 0x14 0x5;
write w GPIO 0x18 0x5;
write w GPIO 0x1C 0x5;

write w GPIO 0x88 0x00;
write w GPIO 0x88 0x01;
write w GPIO 0x88 0x02;
write w GPIO 0x88 0x04;
write w GPIO 0x88 0x08;
write w GPIO 0x88 0x10;
  
```

```

write w GPIO 0x88 0x20;
write w GPIO 0x88 0x40;
write w GPIO 0x88 0x80;

write w eSRAM 0x00 0x12345678;
write w eSRAM 0x04 0x87654321;
write w eSRAM 0x08 0x9ABCDEF0;
write w eSRAM 0x0C 0x0FEDCBA9;

readcheck w eSRAM 0x00 0x12345678;
readcheck w eSRAM 0x04 0x87654321;
readcheck w eSRAM 0x08 0x9ABCDEF0;
readcheck w eSRAM 0x0C 0x0FEDCBA9;

return

```

3. The modified BFM file appears similar to the file shown in Figure 44. BFM commands added in the `SERDESIF_1_user.bfm` do the following:
 - Perform write to MSS GPIO
 - Perform write to eSRAM
 - Perform read-check from eSRAM

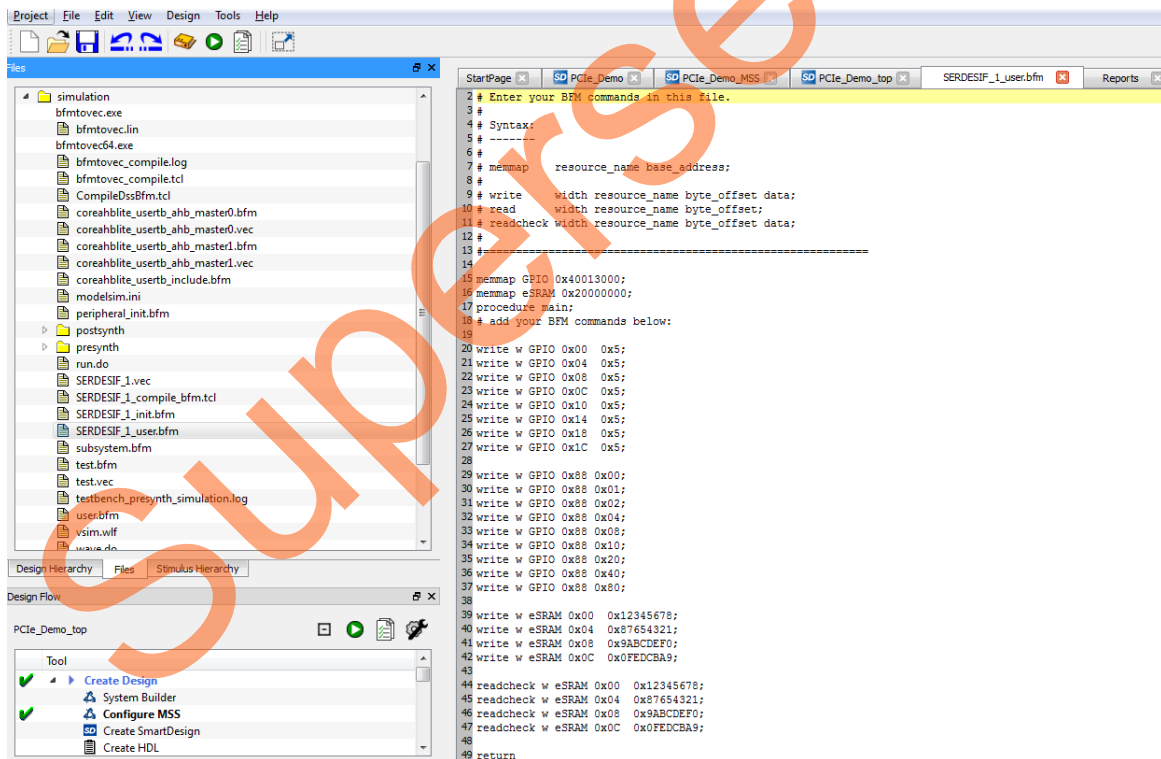


Figure 44 • Modified SERDES User BFM

Step 4: Simulating the Design

The design supports the BFM_PClE simulation level to communicate with the High Speed Serial Interface block through the master AXI bus interface. Although no serial communication actually goes through the High Speed Serial Interface block, this scenario allows validating the fabric interface connections. The SERDESIF_1_user.bfm file under the <Libero project>/simulation folder contains the BFM commands to verify the read/write access to MSS GPIOs and eSRAM.

This section describes how to use the SmartDesign testbench and BFM script file to simulate the design.

1. Add the wave.do file to the PCIe demo design simulation folder by clicking **File > Import > Others**.
2. Browse to the wave.do file location in the design files folder:
M2S_PClE_Control_Demo_DF/Source Files. [Figure 45](#) shows the wave.do file under simulation folder in the Files window.

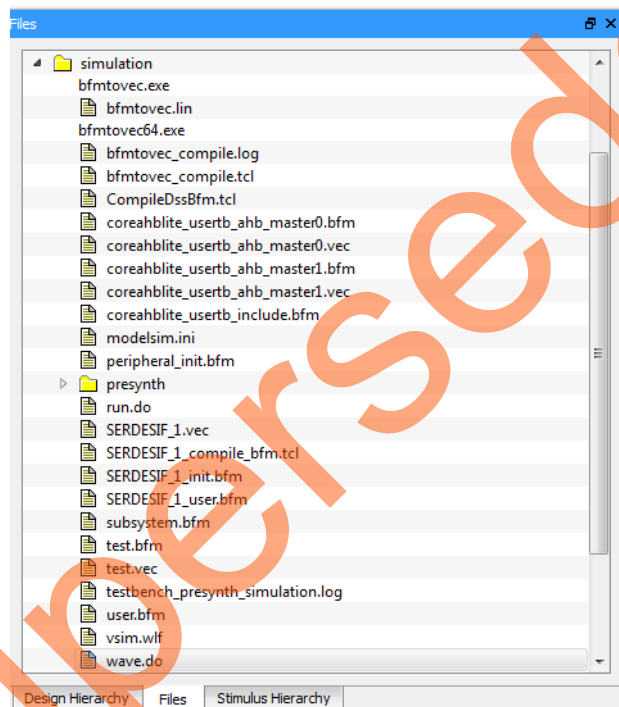


Figure 45 • Wave.do File under Simulation Folder

3. Open the Libero SoC project settings (**Project > Project Settings**).
4. Select **Do File** under **Simulation Options** in the Project Settings window. Change the **Simulation runtime** to 450us, as shown in [Figure 46](#) on page 45.

5. Click **Save**.

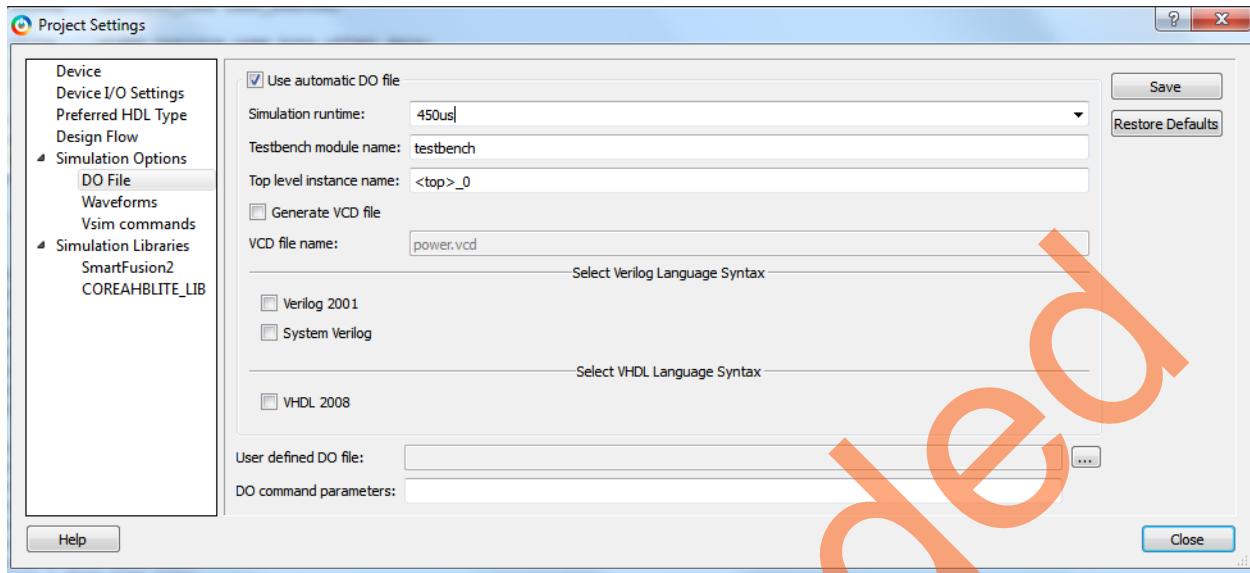


Figure 46 • Project Setting – Do File Simulation Runtime Setting

6. Select **Waveforms** under **Simulation Options** as shown in Figure 47:

- Select Include Do file.
- Select **Log all signals in the design**.
- Click **Close** to close the Project settings dialog box.

Select **Save** when prompted to save the changes.

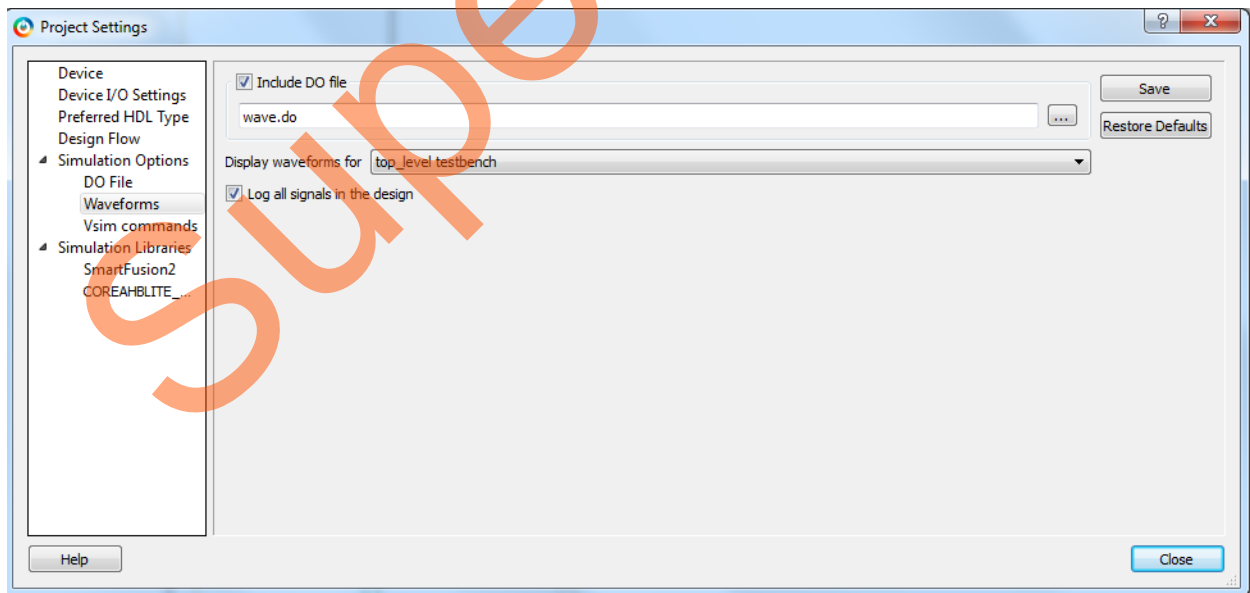


Figure 47 • Project Setting – Waveform

To run the simulation, double-click **Simulate** under **Verify Pre-Synthesized Design** in the **Design Flow** window.

ModelSim runs the design for about 450us. The ModelSim transcript window displays the BFM commands and the BFM simulation completed with no errors, as shown in Figure 48.

```
# BFM:37:write w 40013088 00000040 at 445950 ns
# BFM: Data Write 40013088 00000010
# BFM:38:write w 40013088 00000080 at 446250 ns
# BFM: Data Write 40013088 00000020
# BFM:40:write w 20000000 12345678 at 446550 ns
# BFM: Data Write 40013088 00000040
# BFM:41:write w 20000004 87654321 at 446850 ns
# BFM: Data Write 40013088 00000080
# BFM:42:write w 20000008 9abcdef0 at 447150 ns
# ESRAM0: Word Write 00000000=12345678 at 447250 ns
# BFM: Data Write 20000000 12345678
# BFM:43:write w 2000000c 0fedcba9 at 447350 ns
# ESRAM0: Word Write 00000004=87654321 at 447450 ns
# BFM: Data Write 20000004 87654321
# BFM:45:readcheck w 20000000 00000000 at 447550 ns
# ESRAM0: Word Write 00000008=9abcdef0 at 447650 ns
# BFM: Data Write 20000008 9abcdef0
# BFM:46:readcheck w 20000004 00000000 at 447750 ns
# ESRAM0: Word Write 0000000c=0fedcba9 at 447850 ns
# BFM: Data Write 2000000c 0fedcba9
# BFM:47:readcheck w 20000008 00000000 at 447950 ns
# ESRAM0: Word Read 00000000=12345678 at 448000 ns
# BFM: Data Read 20000000 12345678 MASK:ffffffff at 448100.020000ns
# BFM:48:readcheck w 2000000c 00000000 at 448150 ns
# ESRAM0: Word Read 00000004=87654321 at 448200 ns
# BFM: Data Read 20000004 87654321 MASK:ffffffff at 448300.020000ns
# BFM:50:return
# ESRAM0: Word Read 00000008=9abcdef0 at 448400 ns
# BFM: Data Read 20000008 9abcdef0 MASK:ffffffff at 448500.020000ns
# ESRAM0: Word Read 0000000c=0fedcba9 at 448600 ns
# BFM: Data Read 2000000c 0fedcba9 MASK:ffffffff at 448700.020000ns
#####
# SERDES BFM Simulation Complete - 26 Instructions - NO ERRORS
#####
# Break key hit
```

Figure 48 • SERDES BFM Simulation

7. Figure 49 shows the waveform window with MSS GPIO output signals.

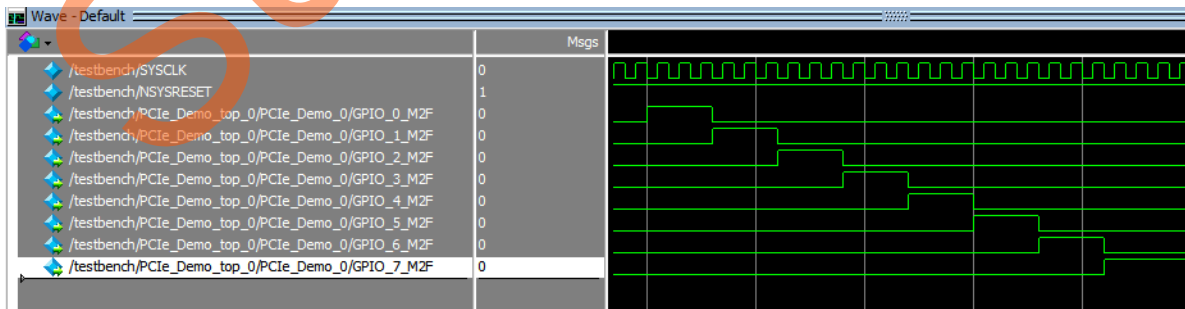


Figure 49 • Simulation Result with MSS GPIO Signals

Step 5: Generating the Program File

1. Double-click **I/O Constraints** in the **Design Flow** window as shown in Figure 50. The **I/O Editor** window is displayed after completing Synthesize and Compile.

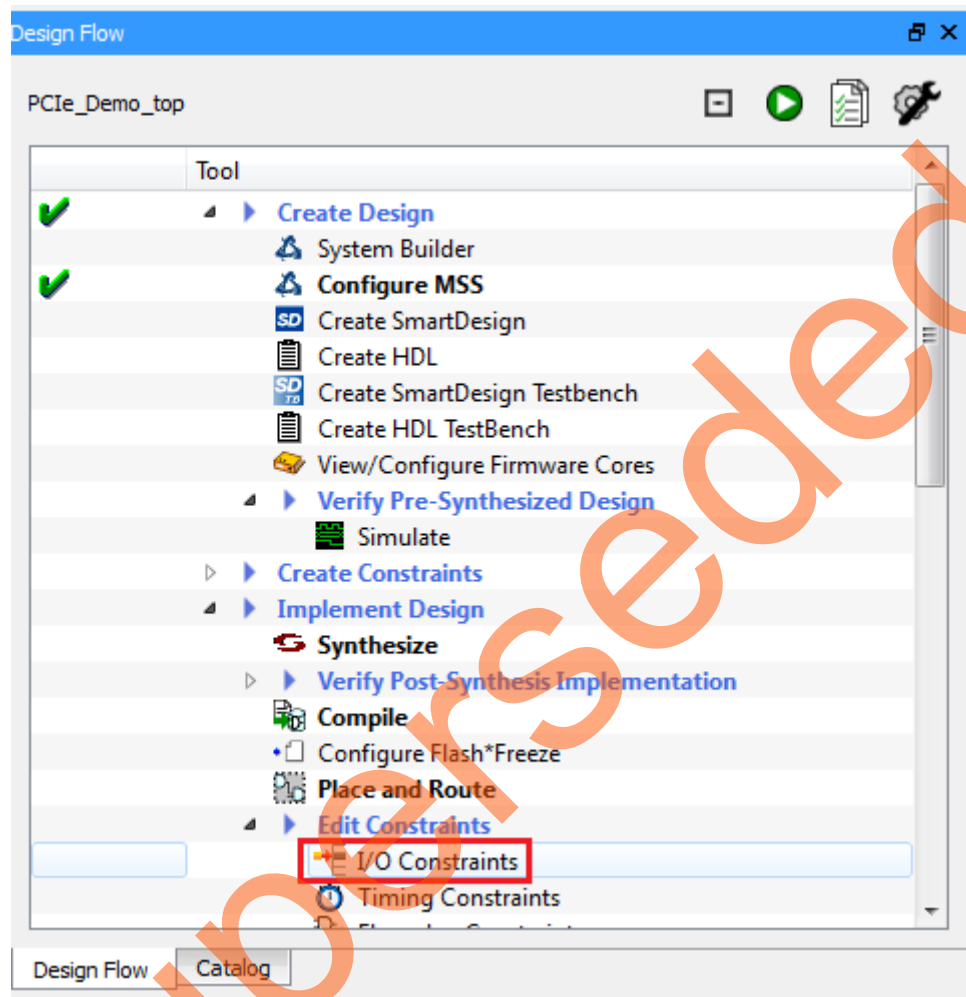


Figure 50 • I/O Constraints

2. The **I/O Editor** is displayed. Make the pin assignments shown in Table 5. After the pins have been assigned, the I/O Editor is displayed as shown in Figure 51 on page 49.

Table 5 • Port to Pin Mapping

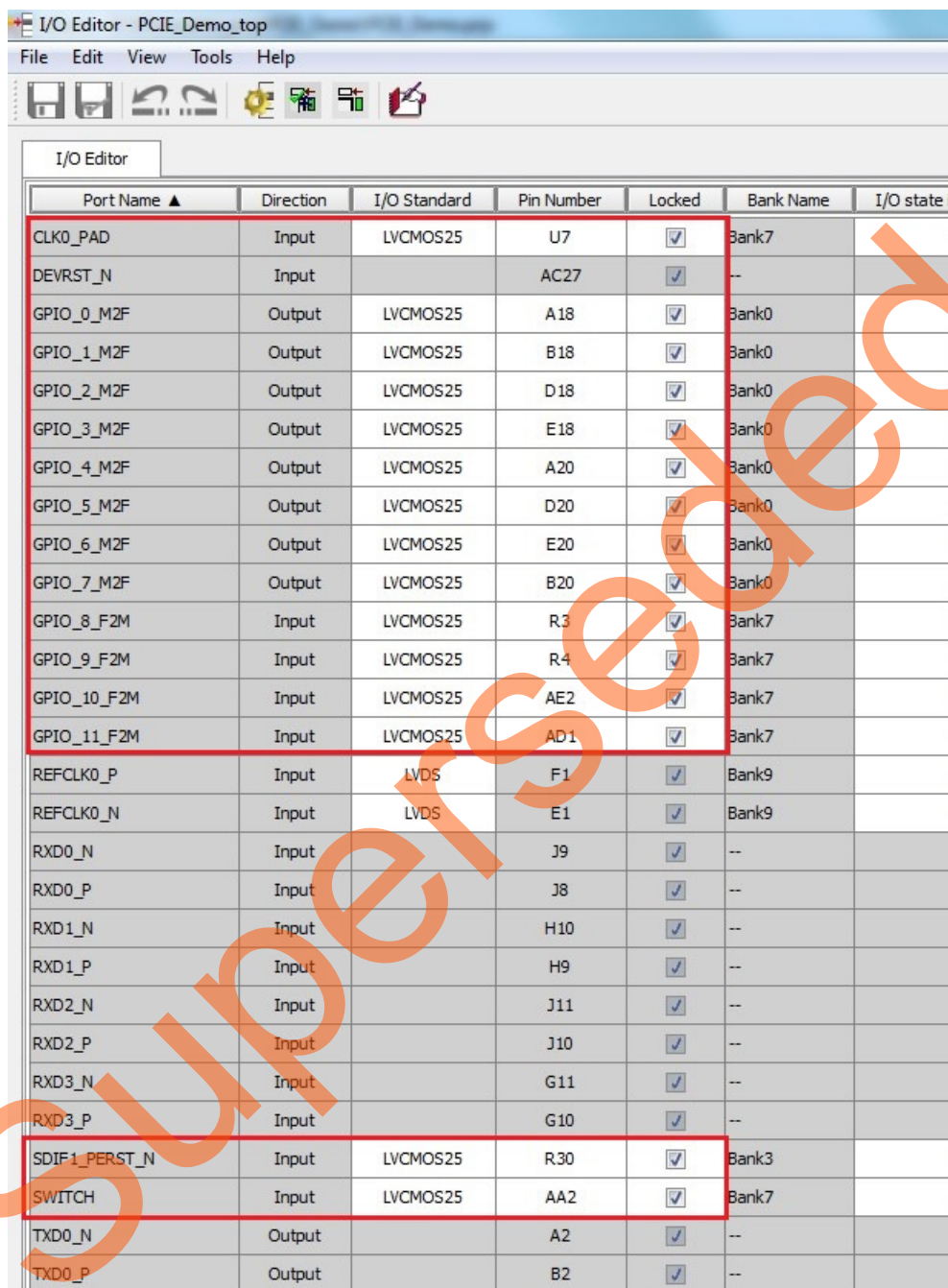
Port Name	Pin Number
CLK0_PAD	U7
GPIO_0_M2F	A18
GPIO_1_M2F	B18
GPIO_2_M2F	D18
GPIO_3_M2F	E18
GPIO_4_M2F	A20
GPIO_5_M2F	D20

Table 5 • Port to Pin Mapping (continued)

Port Name	Pin Number
GPIO_6_M2F	E20
GPIO_7_M2F	B20
GPIO_8_M2F	R3
GPIO_9_M2F	R4
GPIO_10_M2F	AE2
GPIO_11_M2F	AD1
SWITCH	AA2
SDIF1_PERST_N	R30

These pin assignments are for connecting below on the SmartFusion2 Development Kit.

- CLK0_PAD to 50 MHz Clock Oscillator
- GPIO_0 to GPIO_8 for LEDs
- GPIO_8 to GPIO_11 for DIP switches
- SWITCH for SW3
- SDIF1_PERST_N to PERST of PCIe Edge connector



Port Name ▲	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state i
CLK0_PAD	Input	LVC MOS25	U7	<input checked="" type="checkbox"/>	Bank7	
DEVRST_N	Input		AC27	<input checked="" type="checkbox"/>	--	
GPIO_0_M2F	Output	LVC MOS25	A18	<input checked="" type="checkbox"/>	Bank0	
GPIO_1_M2F	Output	LVC MOS25	B18	<input checked="" type="checkbox"/>	Bank0	
GPIO_2_M2F	Output	LVC MOS25	D18	<input checked="" type="checkbox"/>	Bank0	
GPIO_3_M2F	Output	LVC MOS25	E18	<input checked="" type="checkbox"/>	Bank0	
GPIO_4_M2F	Output	LVC MOS25	A20	<input checked="" type="checkbox"/>	Bank0	
GPIO_5_M2F	Output	LVC MOS25	D20	<input checked="" type="checkbox"/>	Bank0	
GPIO_6_M2F	Output	LVC MOS25	E20	<input checked="" type="checkbox"/>	Bank0	
GPIO_7_M2F	Output	LVC MOS25	B20	<input checked="" type="checkbox"/>	Bank0	
GPIO_8_F2M	Input	LVC MOS25	R3	<input checked="" type="checkbox"/>	Bank7	
GPIO_9_F2M	Input	LVC MOS25	R4	<input checked="" type="checkbox"/>	Bank7	
GPIO_10_F2M	Input	LVC MOS25	AE2	<input checked="" type="checkbox"/>	Bank7	
GPIO_11_F2M	Input	LVC MOS25	AD1	<input checked="" type="checkbox"/>	Bank7	
REFCLK0_P	Input	LVDS	F1	<input checked="" type="checkbox"/>	Bank9	
REFCLK0_N	Input	LVDS	E1	<input checked="" type="checkbox"/>	Bank9	
RXD0_N	Input		J9	<input checked="" type="checkbox"/>	--	
RXD0_P	Input		J8	<input checked="" type="checkbox"/>	--	
RXD1_N	Input		H10	<input checked="" type="checkbox"/>	--	
RXD1_P	Input		H9	<input checked="" type="checkbox"/>	--	
RXD2_N	Input		J11	<input checked="" type="checkbox"/>	--	
RXD2_P	Input		J10	<input checked="" type="checkbox"/>	--	
RXD3_N	Input		G11	<input checked="" type="checkbox"/>	--	
RXD3_P	Input		G10	<input checked="" type="checkbox"/>	--	
SDIF1_PERST_N	Input	LVC MOS25	R30	<input checked="" type="checkbox"/>	Bank3	
SWITCH	Input	LVC MOS25	AA2	<input checked="" type="checkbox"/>	Bank7	
TXD0_N	Output		A2	<input checked="" type="checkbox"/>	--	
TXD0_P	Output		B2	<input checked="" type="checkbox"/>	--	

Figure 51 • I/O Editor

- After updating I/O editor, click **Commit and Check**.
- Close the I/O editor.

- Click **Generate Programming Data** as shown in Figure 52 to complete place and route, verify timing, and generate the programming file.



Figure 52 • Generate Programming Data

Running the Demo

Demo Setup

- Connect the FlashPro4 programmer to the J59 connector of SmartFusion2 SoC FPGA Development Kit.
- Connect the jumpers on the SmartFusion2 Development Kit, as shown in Table 6.

Note: While making the jumper connections, the power supply switch **SW7** on the board should be in **OFF** position.

Table 6 • SmartFusion2 SoC FPGA Development Kit Jumper Settings

Jumper	Pin (from)	Pin (to)	Comments
J70, J93, J94, J117, J123, J142, J157, J160, J167, J225, J226, J227	1	2	Default
J2	1	3	Default
J23	2	3	Default

- Connect the power supply to the J18 connector.
- Switch the power supply switch **SW7** to **ON** position.

5. To program the SmartFusion2 device double-click **Run Programming Action** in the **Design Flow** window as shown in Figure 53.

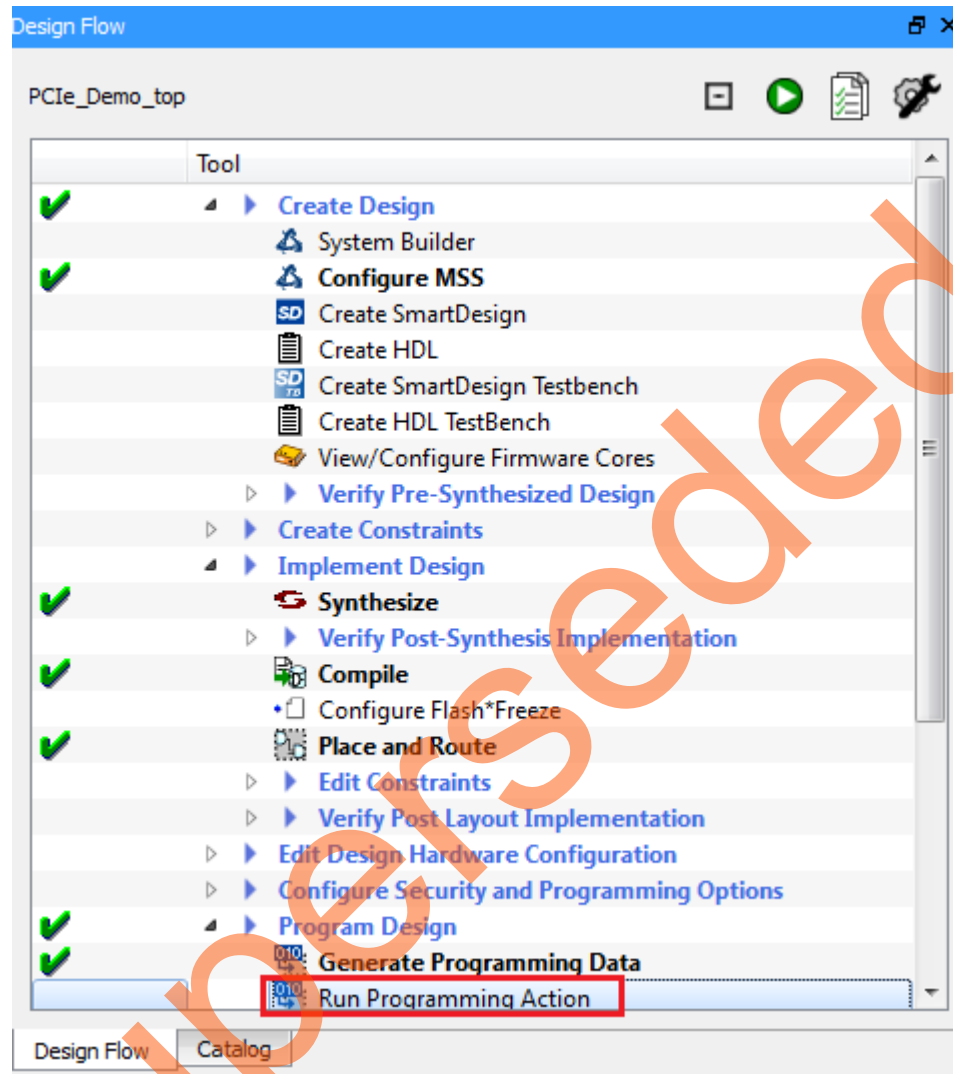


Figure 53 • Run Programming Action

6. After Successful programming, power OFF the SmartFusion2 Development Kit and shut down the Host PC.
This demo is designed to run in any PCIe Gen 2 compliant slot. If the host PC does not support Gen 2 compliant slot, the demo will switch to Gen 1 mode.
7. Connect the **J230 – PCIe Edge Card Ribbon Cable** to **Host PC PCIe Gen 2 slot or Gen 1 slot** as applicable.

Caution: Host PC needs to be powered OFF while inserting the PCIe Edge connector. If it is not, the PCIe device detection and selection of Gen1 or Gen2 mode may not occur properly. This is very dependent on the host PC PCIe configuration. It is recommended that the host PC is powered OFF before inserting the PCIe card.

8. The board setup is shown in [Figure 54](#).

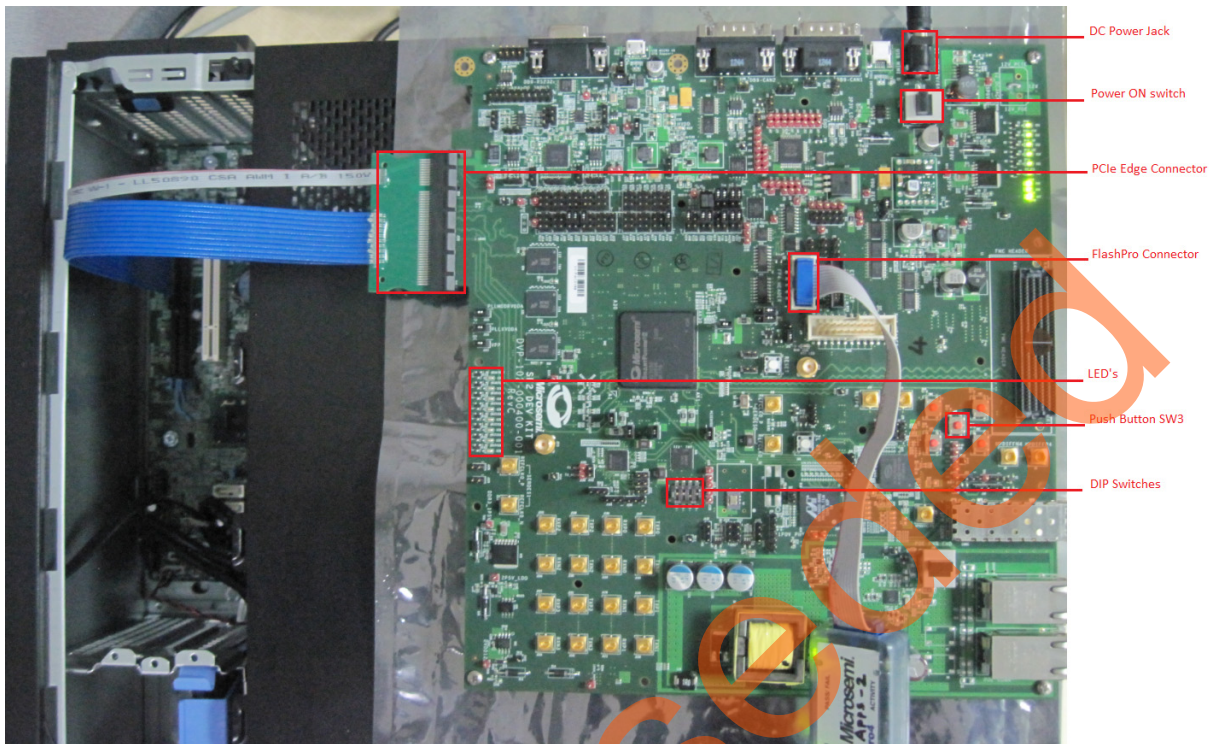


Figure 54 • SmartFusion2 Development Kit Setup

9. Switch the power supply switch **SW7** to **ON** position.

Running the Demo Design

This demo can run on both windows and Red Hat Linux operating system.

To run the demo on Windows operating system GUI, Jungo drivers are provided. Refer to [Running the Demo Design on Windows](#).

To run the demo on Linux operating system native Red Hat Linux drivers and command line scripts are provided. Refer to [Running the Demo Design on Linux](#).

Running the Demo Design on Windows

1. Power on the Host PC and check the Host PC Device Manager for PCIe Device. It will be similar to [Figure 55](#). If the device is not detected, power cycle the SmartFusion2 Development Kit and click “scan for hardware changes” in Device Manager.

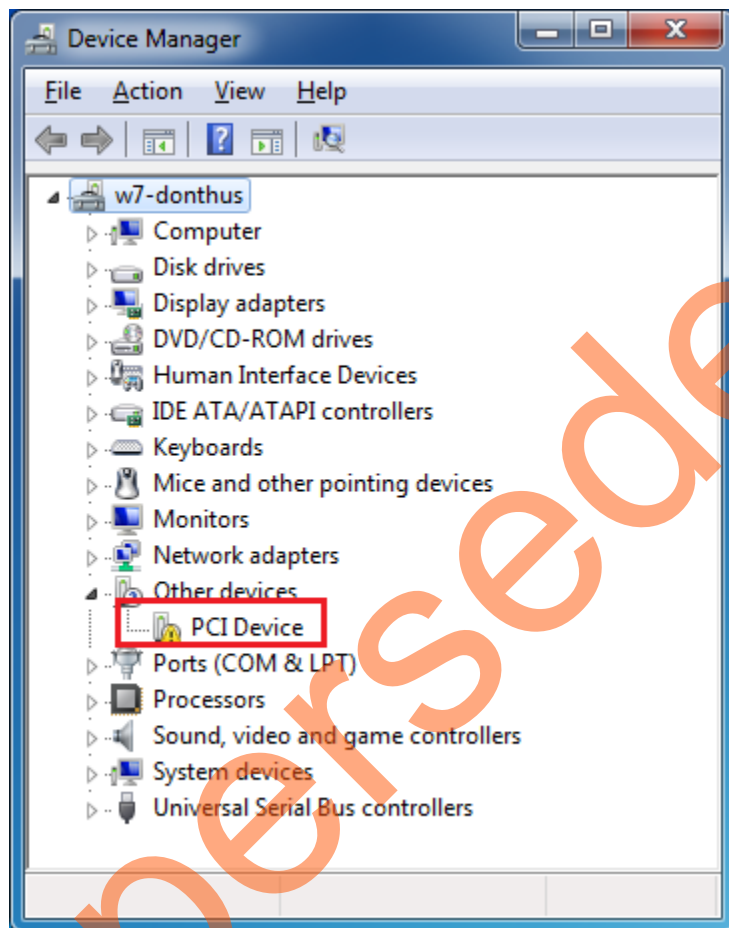


Figure 55 • Device Manager - PCIe Device Detection

2. If the Host PC has any other installed drivers (previous versions of Jungo drivers) for the SmartFusion2PCIe device, uninstall them. To uninstall previous versions of Jungo drivers follow steps 12 and 13.
3. To uninstall previous Jungo drivers go to device manager and right-click on DEVICE as shown in [Figure 56](#) on page 54 device uninstall.

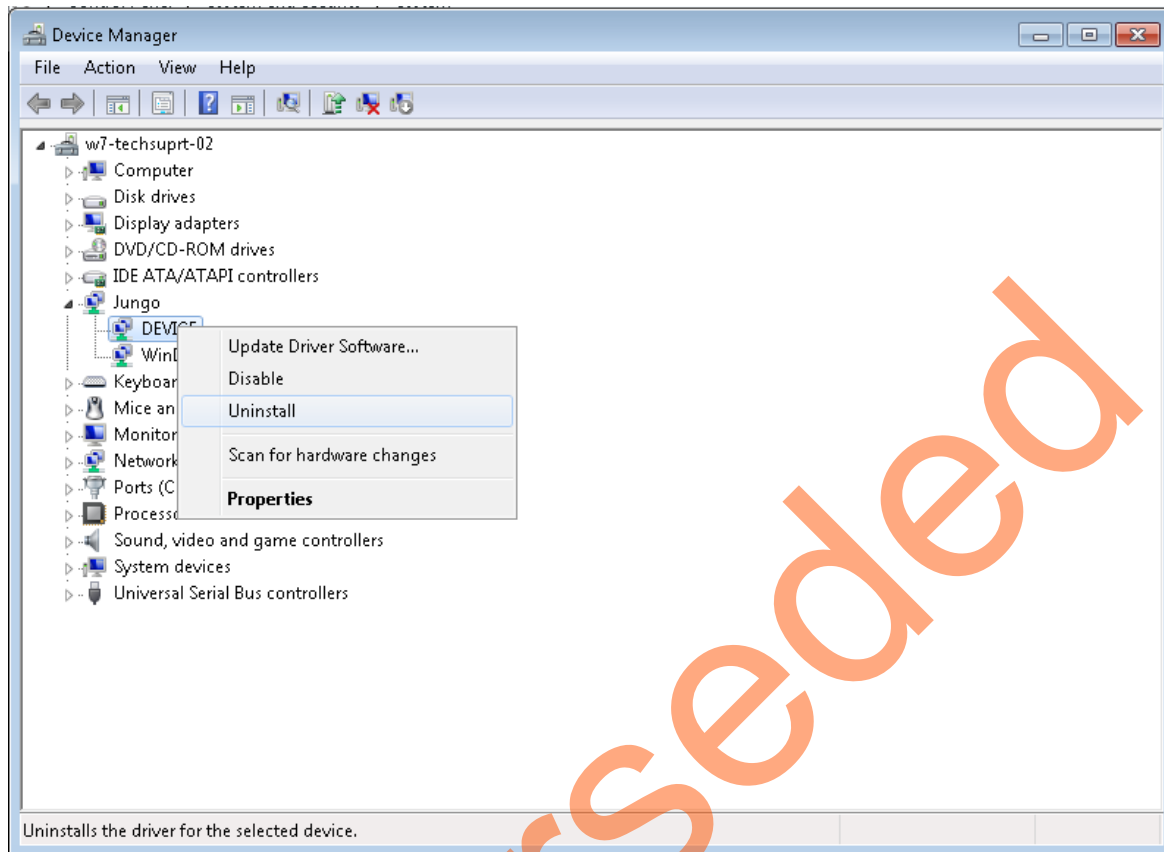


Figure 56 • Device Uninstall

4. The DEVICE uninstall window is displayed as shown in [Figure 57](#) Confirm Device Uninstall Select Delete the driver software for this device. After uninstalling previous Jungo drivers, make sure that the PCI Device is detected in the Device Manager window as shown in [Figure 57](#) Device manager.



Figure 57 • Confirm Device Uninstall

Note: If the device is still not detected, check whether or not the BIOS version in Host PC is latest, and if PCI is enabled in the Host PC BIOS.

Drivers Installation

The PCIe Demo uses a driver framework provided by Jungo WinDriverPro. To install the PCIe drivers on Host PC for SmartFusion2 Development Kit, use the following steps:

1. Extract the **PCle_Demo.rar** to C:\ drive. The PCle_Demo.rar is located in the provided design files:
 - M2S_PCIE_Control_DEMO_DF\Windows_64bit\Drivers\PCle_Demo.rar
2. Run the batch file **C:\PCle_Demo\DriverInstall\Jungo_KP_install.bat**

Note: Installing these drivers require Host PC Administration rights.

3. Click **Install** if the window is displayed as shown in Figure 58.

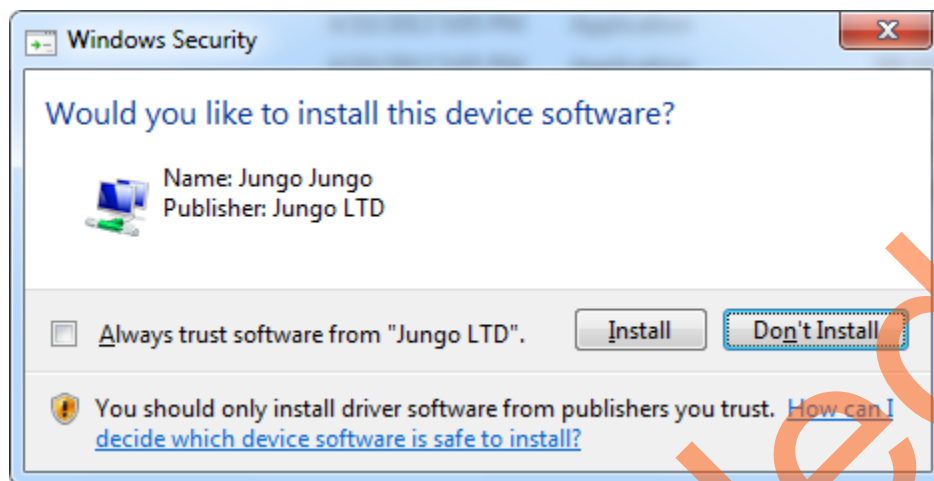


Figure 58 • Jungo Driver Installation

- Note:** If the installation is not in progress, right-click on the command prompt and select Run as administrator. Run the batch file `C:\PCIe_Demo\DriverInstall\Jungo_KP_install.bat` from command prompt.
4. Click **Install this driver software anyway** if the window appears as shown in Figure 59.

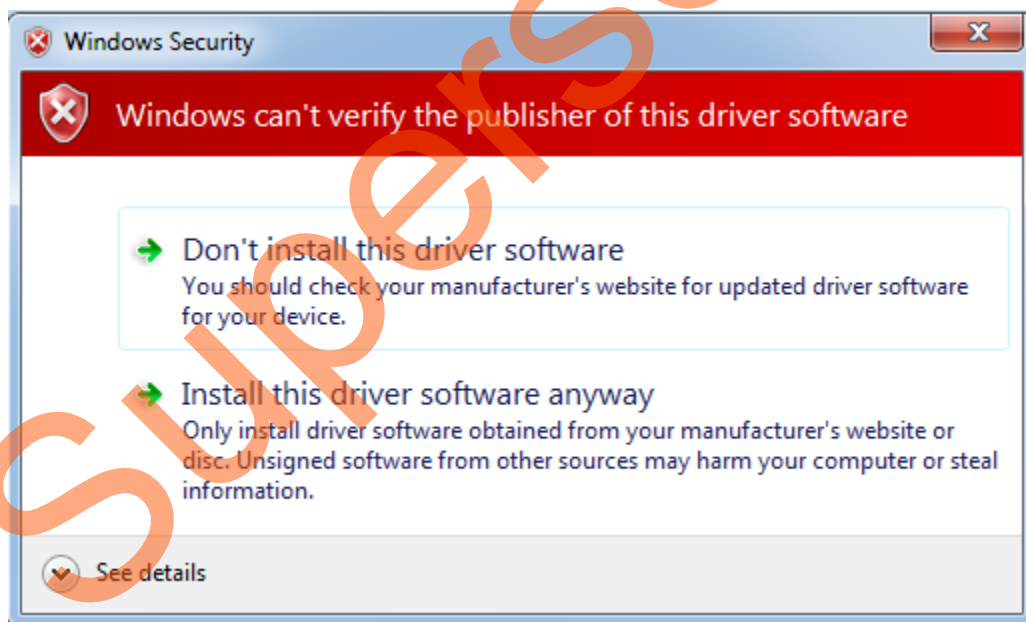


Figure 59 • Windows Security

PCIe Demo GUI

SmartFusion2 PCIe Demo GUI is a simple graphic user interface that runs on the Host PC to communicate with the SmartFusion2 PCIe EP device. The GUI provides the PCIe link status, driver information, and demo controls. The GUI invokes the PCIe driver installed on the Host PC and provides commands to the driver according to the user selection.

Use the following steps to install the GUI:

1. Extract the PCIe_Demo_GUI_Installer.rar from the provided design files:
M2S_PCIE_Control_Demo_DF\Windows_64bit\GUI.
2. Double-click **setup.exe** in the provided GUI installation (PCIe_Demo_GUI_Installer\setup.exe).
Apply default options as shown in [Figure 60](#).

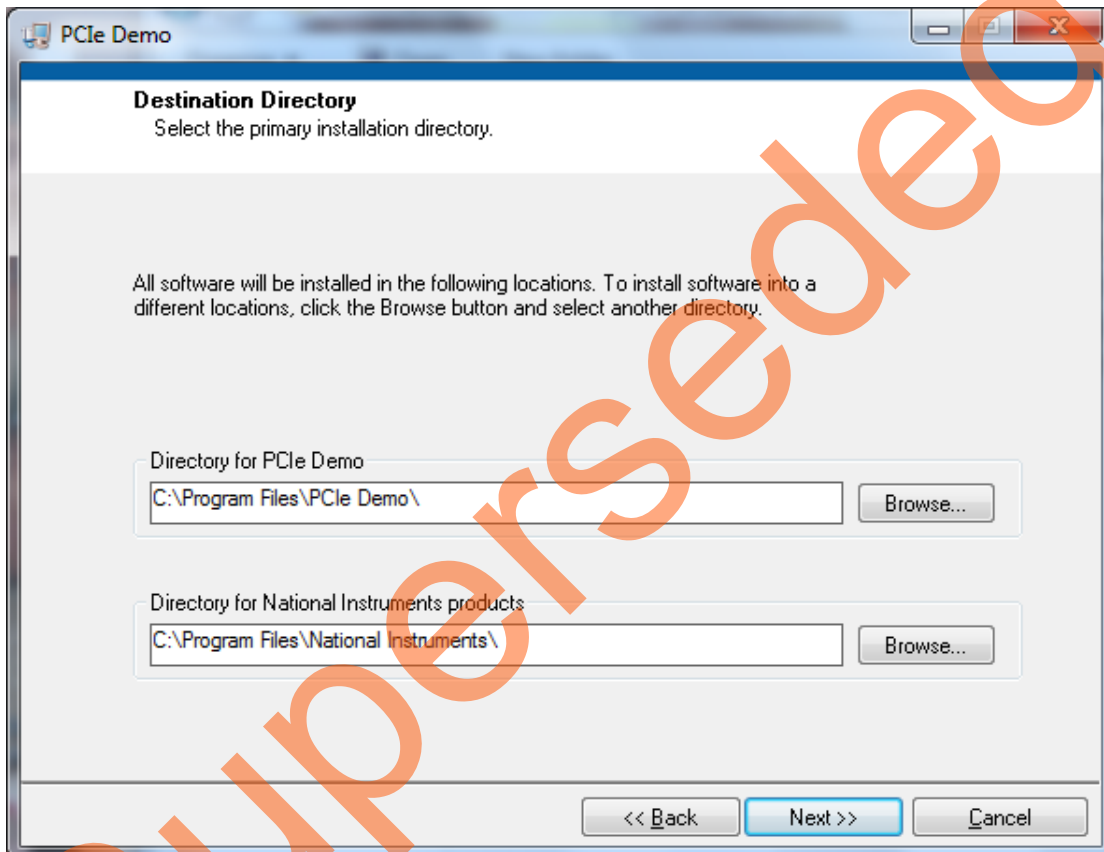


Figure 60 • GUI Installation

3. Click **Next** to complete the installation. After successful installation the following window is displayed:

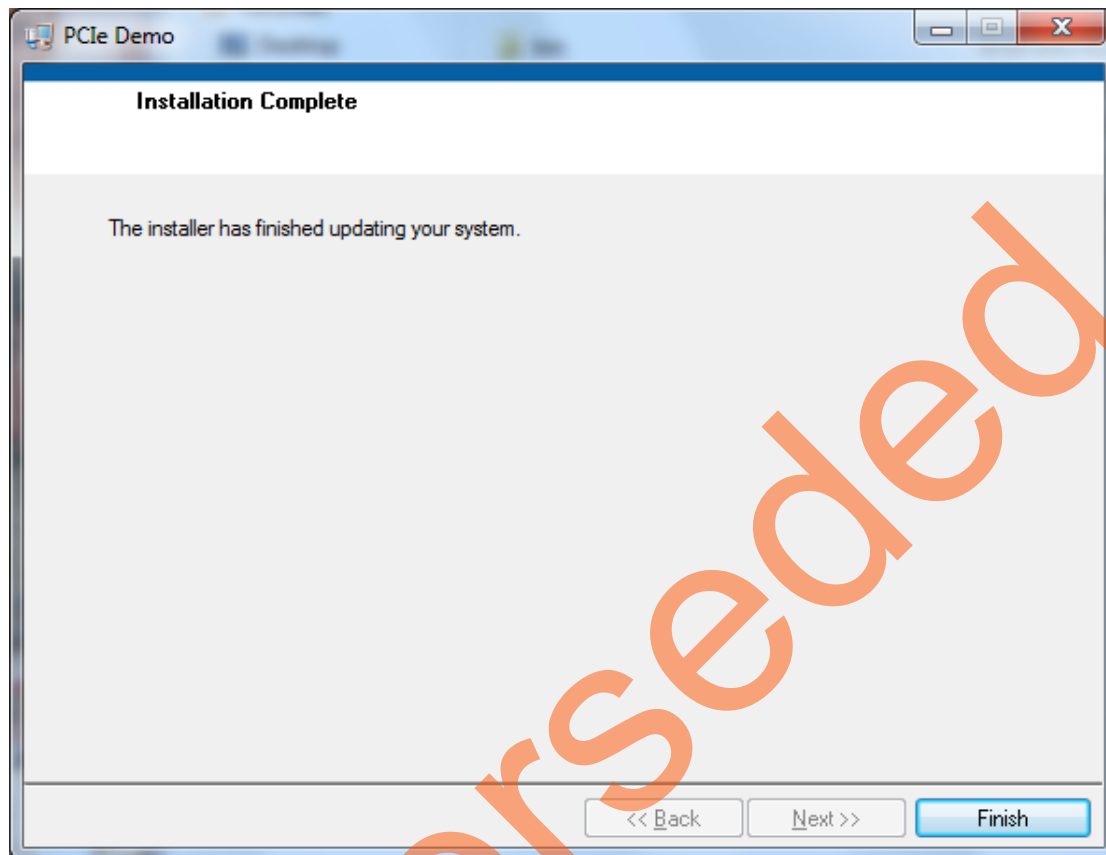


Figure 61 • Successful GUI Installation

4. Restart the host PC.

Running the Design

1. Check the Host PC **Device Manager** for the drivers. If the device is not detected, power cycle the SmartFusion2 Development Kit and click "scan for hardware changes" in Device Manager. Make sure that the board is switched on.

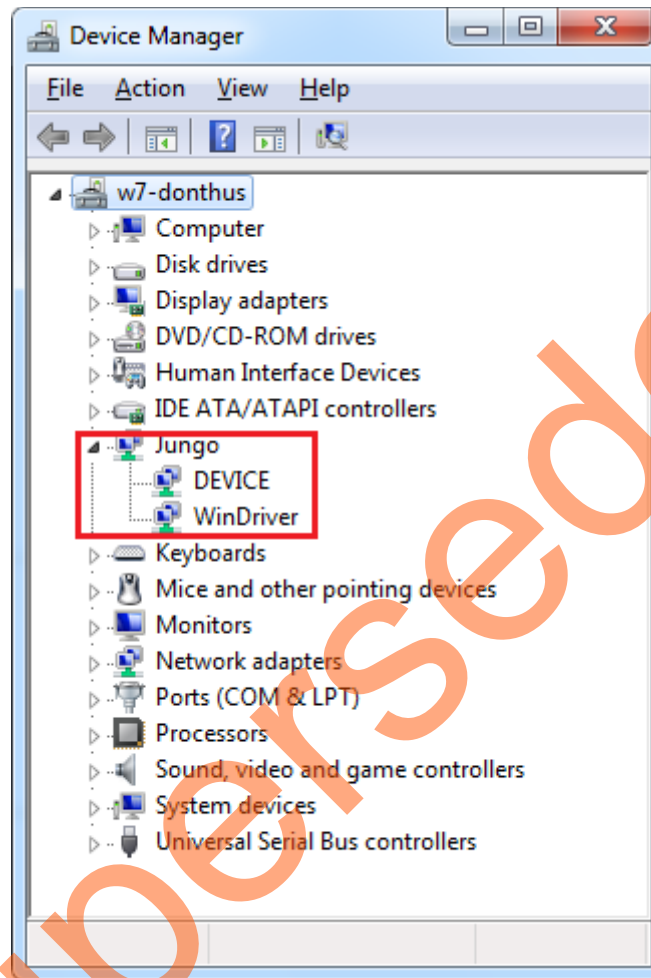


Figure 62 • Device Manager - PCIe Device Detection

Note: If a warning symbol is displayed on the **DEVICE** or **WinDriver** icons in the **Device Manager**, uninstall them and start from step 1 of "Drivers Installation" on page 55.

2. Invoke the GUI from **ALL Programs > PCIeDemo > PCIe Demo**. The GUI is displayed as shown in Figure 63.

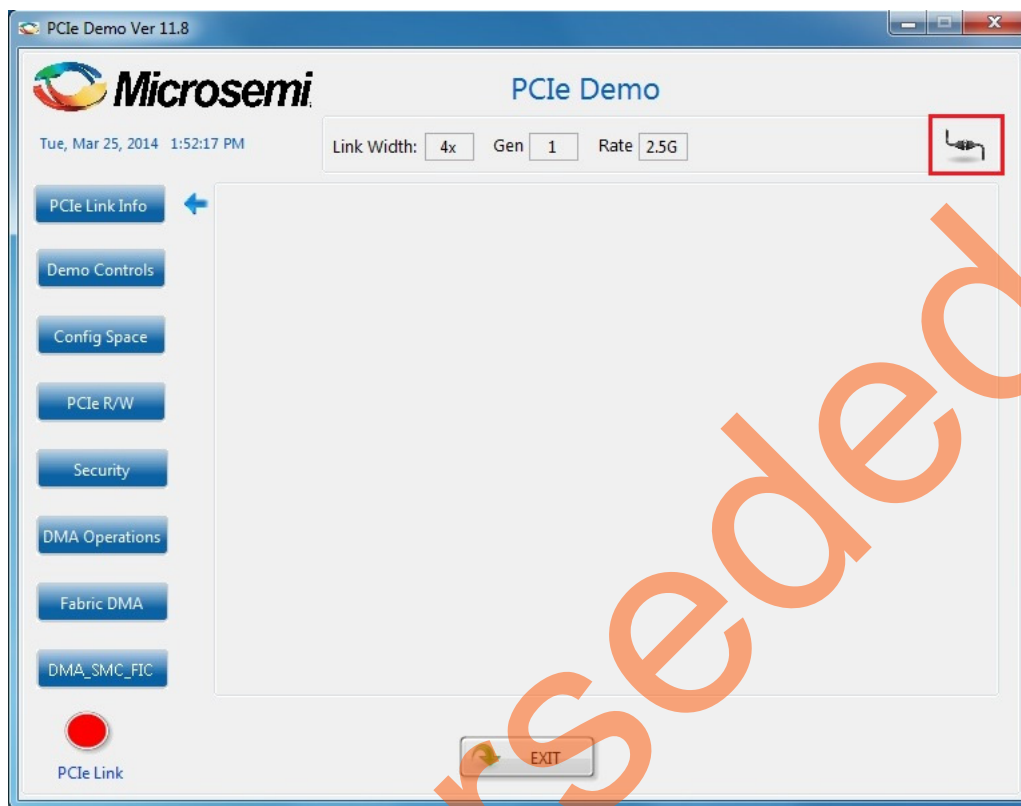


Figure 63 • PCIe Demo GUI

- Click the **Connect** button at the top-right corner of the GUI. The messages will be displayed on the GUI as shown in Figure 64.

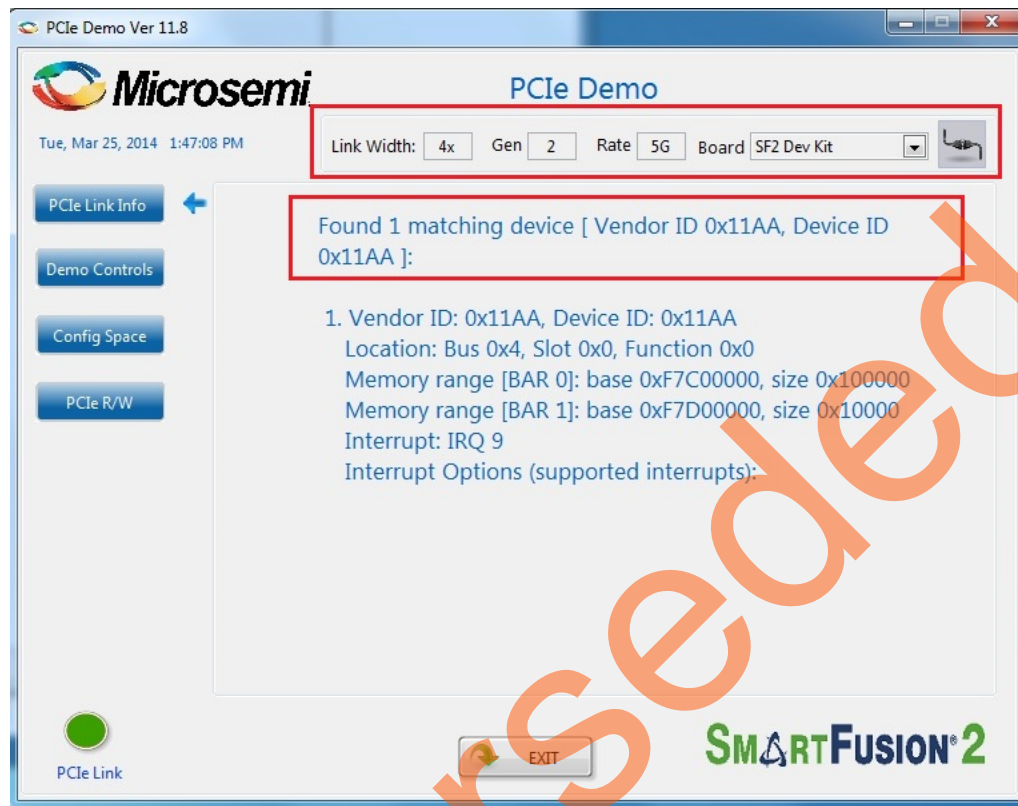


Figure 64 • Version Information

- Clicking **Demo Controls** in the GUI displays the LEDs options and DIP switch positions as shown in Figure 65.

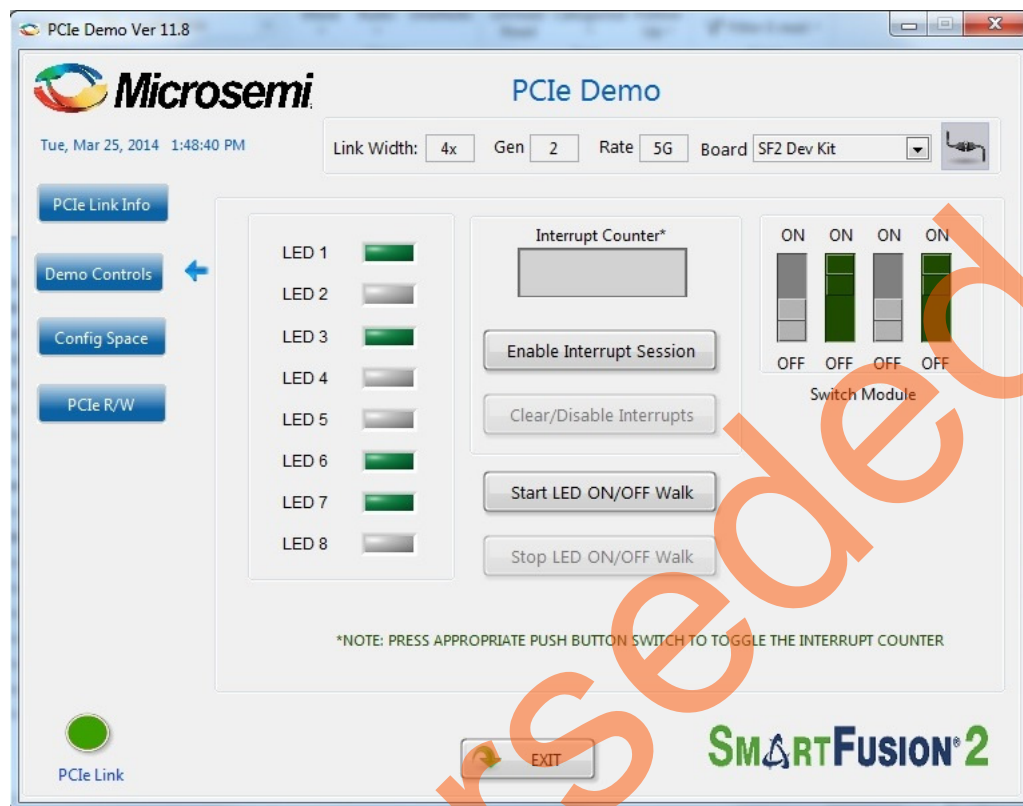


Figure 65 • Demo Controls

- Click LEDs in GUI to ON/OFF the LEDs on SmartFusion2 Development Kit.
- Click **Start LED ON/OFF Walk** to blink the LEDs on SmartFusion2 Development Kit.
- Click **Stop LED ON/OFF Walk** to stop the LEDs blinking.
- Change the DIP switch positions on the SmartFusion2 Development Kit (SW10) and observe the similar position of switches in GUI SWITCH MODULE.
- Click **Enable Interrupt Session** to enable the PCIe interrupt.

10. Press the push button SW3 on the SmartFusion2 Development Kit and observe the interrupt count on the **Interrupt Counter** field in GUI as shown in [Figure 66](#).

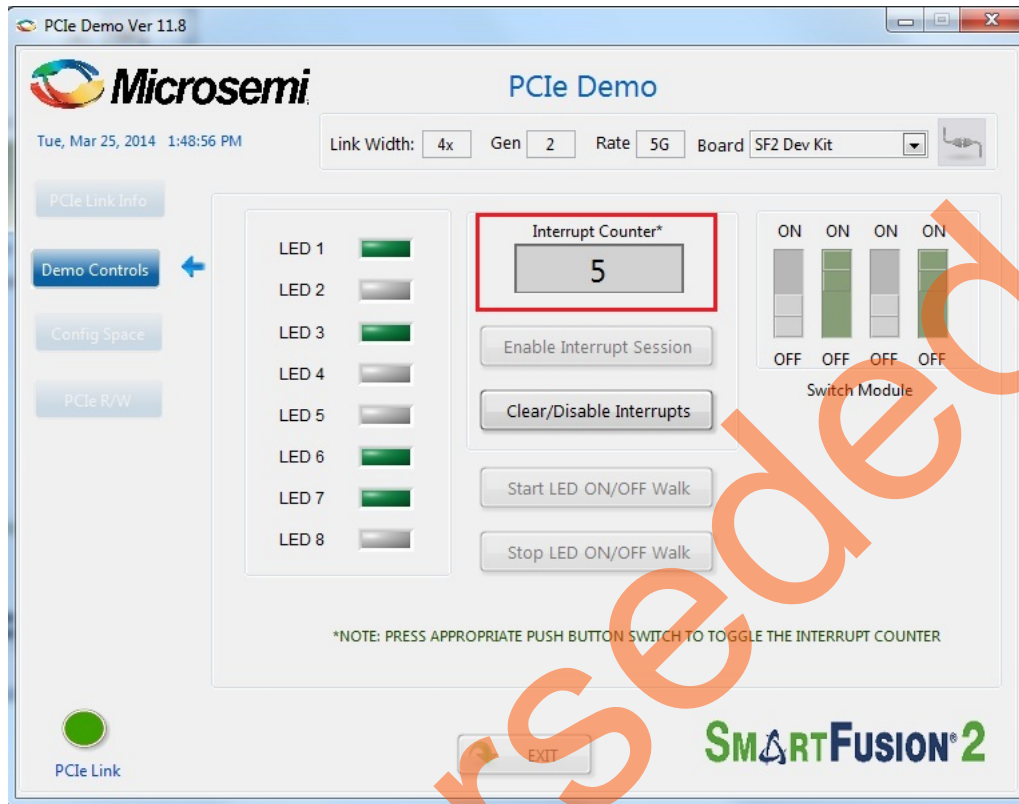


Figure 66 • Interrupt Counter

11. Click **Clear/Disable Interrupts** to clear and disable the PCIe interrupts.

12. Click **Config Space** to read details about the PCIe configuration space. Figure 67 shows the PCIe configuration space.

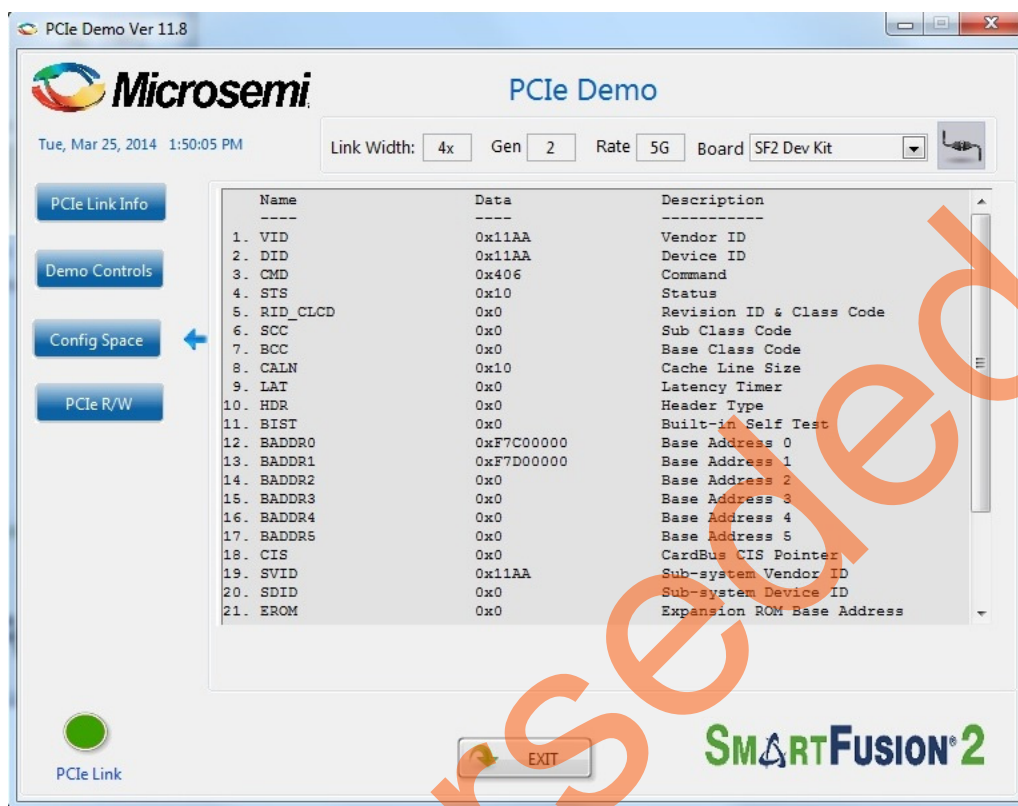


Figure 67 • Configuration Space

13. Click PCIe R/W to perform read and writes to eSRAM memory through BAR1 space. Figure 68 shows the PCIe R/W window. Enter the address in the Address field between 0x0000 to 0xFFFFC. The Data field accepts a 32-bit hexadecimal value.

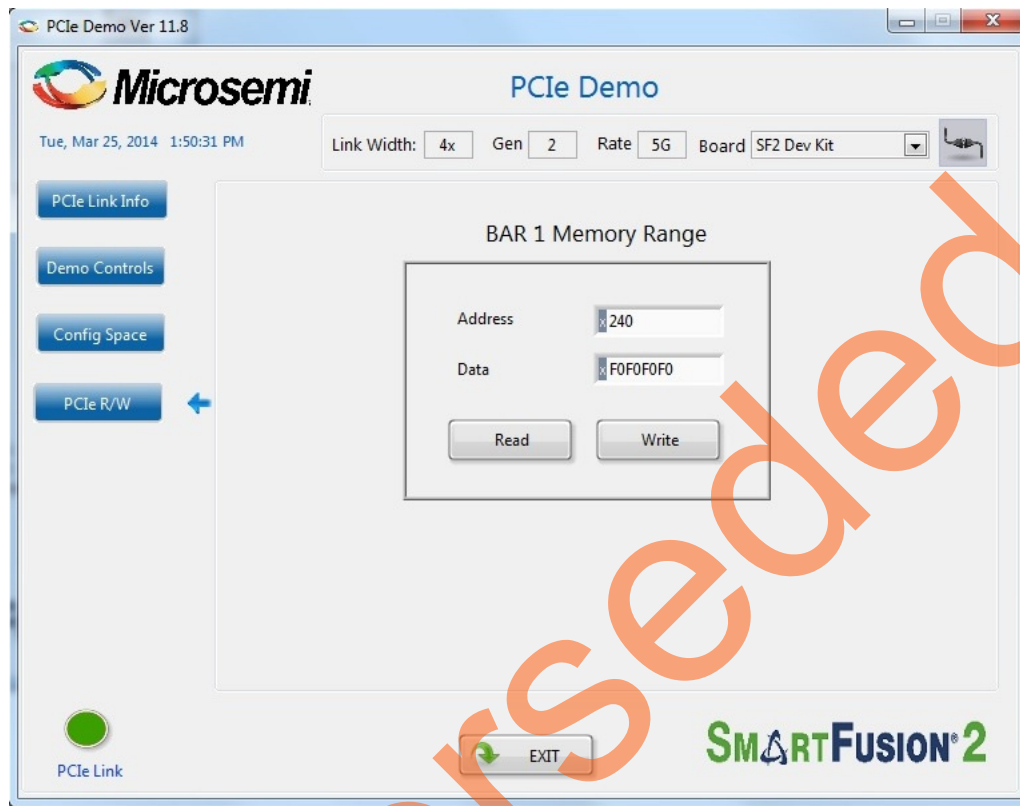


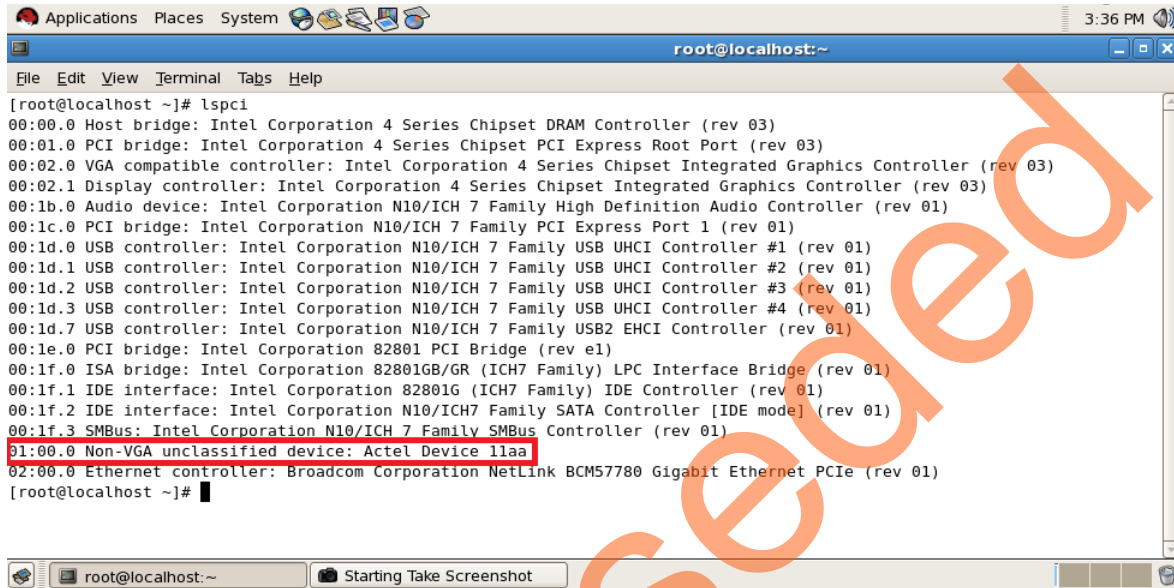
Figure 68 • Perform Read and Write to eSRAM Using PCIe

14. Click **Exit** to quit the demo.

Running the Demo Design on Linux

1. Switch **ON** the Red Hat Linux Host PC.
2. Red Hat Linux Kernel detects the SmartFusion2 PCIe end point as Actel Device.
3. On Linux Command Prompt Use `lspci` command to display the PCIe info.

`lspci`



```
[root@localhost ~]# lspci
00:00.0 Host bridge: Intel Corporation 4 Series Chipset DRAM Controller (rev 03)
00:01.0 PCI bridge: Intel Corporation 4 Series Chipset PCI Express Root Port (rev 03)
00:02.0 VGA compatible controller: Intel Corporation 4 Series Chipset Integrated Graphics Controller (rev 03)
00:02.1 Display controller: Intel Corporation 4 Series Chipset Integrated Graphics Controller (rev 03)
00:1b.0 Audio device: Intel Corporation N10/ICH 7 Family High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation N10/ICH 7 Family PCI Express Port 1 (rev 01)
00:1d.0 USB controller: Intel Corporation N10/ICH 7 Family USB UHCI Controller #1 (rev 01)
00:1d.1 USB controller: Intel Corporation N10/ICH 7 Family USB UHCI Controller #2 (rev 01)
00:1d.2 USB controller: Intel Corporation N10/ICH 7 Family USB UHCI Controller #3 (rev 01)
00:1d.3 USB controller: Intel Corporation N10/ICH 7 Family USB UHCI Controller #4 (rev 01)
00:1d.7 USB controller: Intel Corporation N10/ICH 7 Family USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR (ICH7 Family) LPC Interface Bridge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7 Family) IDE Controller (rev 01)
00:1f.2 IDE interface: Intel Corporation N10/ICH7 Family SATA Controller [IDE mode] (rev 01)
00:1f.3 SMBus: Intel Corporation N10/ICH 7 Family SMBus Controller (rev 01)
01:00.0 Non-VGA unclassified device: Actel Device 11aa
02:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57780 Gigabit Ethernet PCIe (rev 01)
[root@localhost ~]#
```

Figure 69 • PCIe Device Detection

Drivers Installation

Enter the following commands in the Linux command prompt to install the PCIe drivers:

1. Create the **sf2** directory under the **home/** directory using the following command:
`mkdir /home/sf2`
2. Copy the **M2S_PCIE_Control_DEMO_DF\Linux_64bit\Drivers\PCIe_Driver** folder from the Windows host PC and place it into the **/home/sf2** directory of RedHat Linux host PC.
3. Copy the **M2S_PCIE_Control_DEMO_DF\Linux_64bit\Drivers\inc** folder from the Windows host PC and place it into the **/home/sf2** directory of RedHat Linux host PC.

The **/home/sf2** directory must contain **PCIe_Driver/ inc/** folders.

4. Execute `ls` command to display the contents of **/home/sf2** directory.
`ls`
5. Change to **inc/** directory by using the following command:
`cd /home/sf2/inc`

6. Edit the `board.h` file for SmartFusion2 Development Kit as shown in Figure 70.

```
#vi board.h
#define SF2
#undef IGL2
```

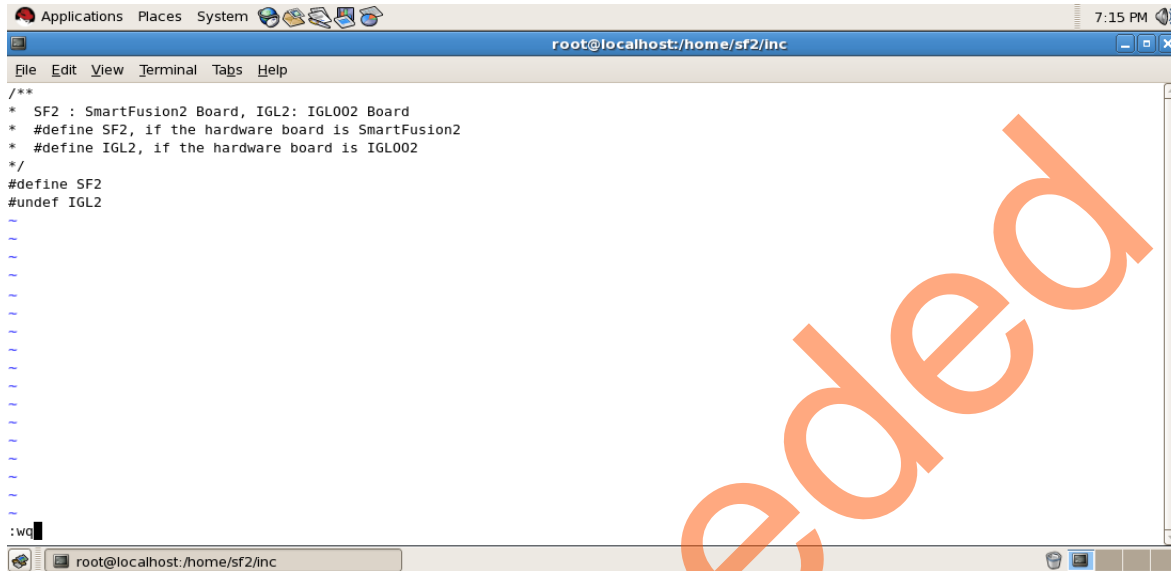


Figure 70 • Edit board.h File

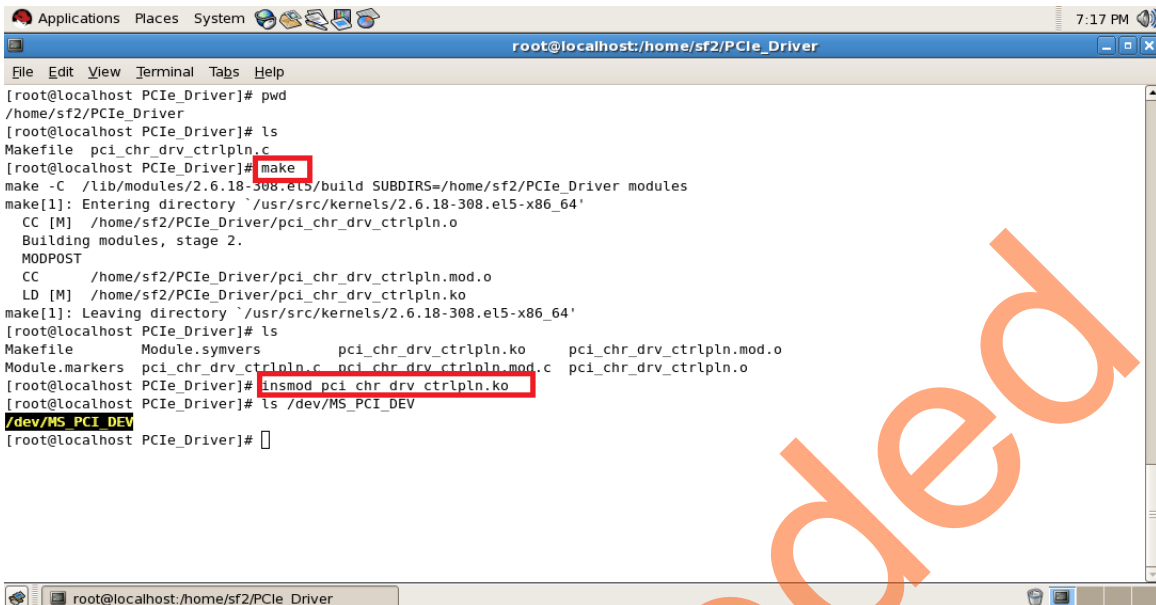
7. To save the selected file, execute the `:wq` command
8. Change to `PCIe_Driver/` directory using the `cd` command:

```
#cd /home/sf2/PCIe_Driver
```
9. To compile the Linux PCIe device driver code, execute `make` command.

```
#make clean [To clean any *.o, *.ko files]
#make
```
10. The kernel module, `pci_chr_drv_ctrlpln.ko` creates in the same directory.
11. To insert the Linux PCIe device driver as a module, execute `insmod` command.

```
#insmod pci_chr_drv_ctrlpln.ko
```

Note: Root privileges are required to execute this command.



```

root@localhost:/home/sf2/PCie_Driver
File Edit View Terminal Tabs Help
[root@localhost PCie_Driver]# pwd
/home/sf2/PCie_Driver
[root@localhost PCie_Driver]# ls
Makefile pci_chr_drv_ctrlpln.c
[root@localhost PCie_Driver]# make
make -C /lib/modules/2.6.18-308.el5/build SUBDIRS=/home/sf2/PCie_Driver modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-308.el5-x86_64'
CC [M] /home/sf2/PCie_Driver/pci_chr_drv_ctrlpln.o
Building modules, stage 2.
MODPOST
CC /home/sf2/PCie_Driver/pci_chr_drv_ctrlpln.mod.o
LD [M] /home/sf2/PCie_Driver/pci_chr_drv_ctrlpln.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.18-308.el5-x86_64'
[root@localhost PCie_Driver]# ls
Makefile Module.symvers pci_chr_drv_ctrlpln.ko pci_chr_drv_ctrlpln.mod.o
Module.markers pci_chr_drv_ctrlpln.c pci_chr_drv_ctrlpln.mod.c pci_chr_drv_ctrlpln.o
[root@localhost PCie_Driver]# insmod pci_chr_drv_ctrlpln.ko
[root@localhost PCie_Driver]# ls /dev/MS_PCI_DEV
/dev/MS_PCI_DEV
[root@localhost PCie_Driver]#

```

Figure 71 • PCIe Device Driver Installation

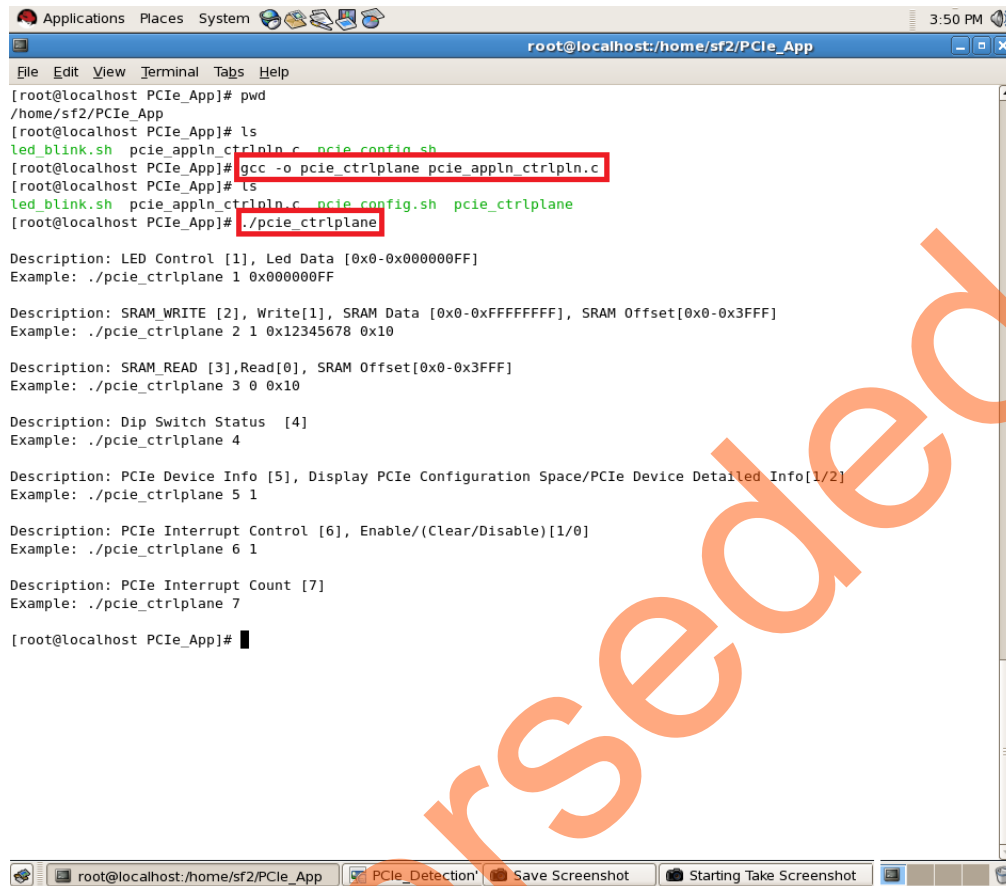
- After successful Linux PCIe device driver installation, check `/dev/MS_PCI_DEV` got created by using the following Linux command:

```
#ls /dev/MS_PCI_DEV
```

Note: `/dev/MS_PCI_DEV` interface is used to access the SmartFusion2 PCIe end point from Linux user space.

Linux PCIe Application Compilation and PCIe Control Plane Utility Creation

- Change to the `/home/sf2/` directory using the following command:
#cd /home/sf2
- Copy the `M2S_PCIE_Control_DEMO_DF\Linux_64bit\Util\PCie_App` folder from the Windows host PC and place it into the `/home/sf2` directory of RedHat Linux host PC.
- Change to the `/home/sf2/PCie_App` directory using the following command:
#cd /home/sf2/PCie_App
- Compile the Linux user space application `pcie_appln_ctrlpln.c` by using `gcc` command.
#gcc -o pcie_ctrlplane pcie_appln_ctrlpln.c



```

root@localhost: /home/sf2/PCie_App
File Edit View Terminal Tabs Help
[root@localhost PCie_App]# pwd
/home/sf2/PCie_App
[root@localhost PCie_App]# ls
led_blink.sh pcie_appln_ctrlpln.c pcie_config.sh
[root@localhost PCie_App]# gcc -o pcie_ctrlplane pcie_appln_ctrlpln.c
[root@localhost PCie_App]# ls
led_blink.sh pcie_appln_ctrlpln.c pcie_config.sh pcie_ctrlplane
[root@localhost PCie_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3], Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[root@localhost PCie_App]#

```

Figure 72 • Linux PCIe Application Utility

5. After successful compilation, Linux PCIe application utility `pcie_ctrlplane` creates in the same directory.
6. On Linux Command Prompt run the `pcie_ctrlplane` utility as:

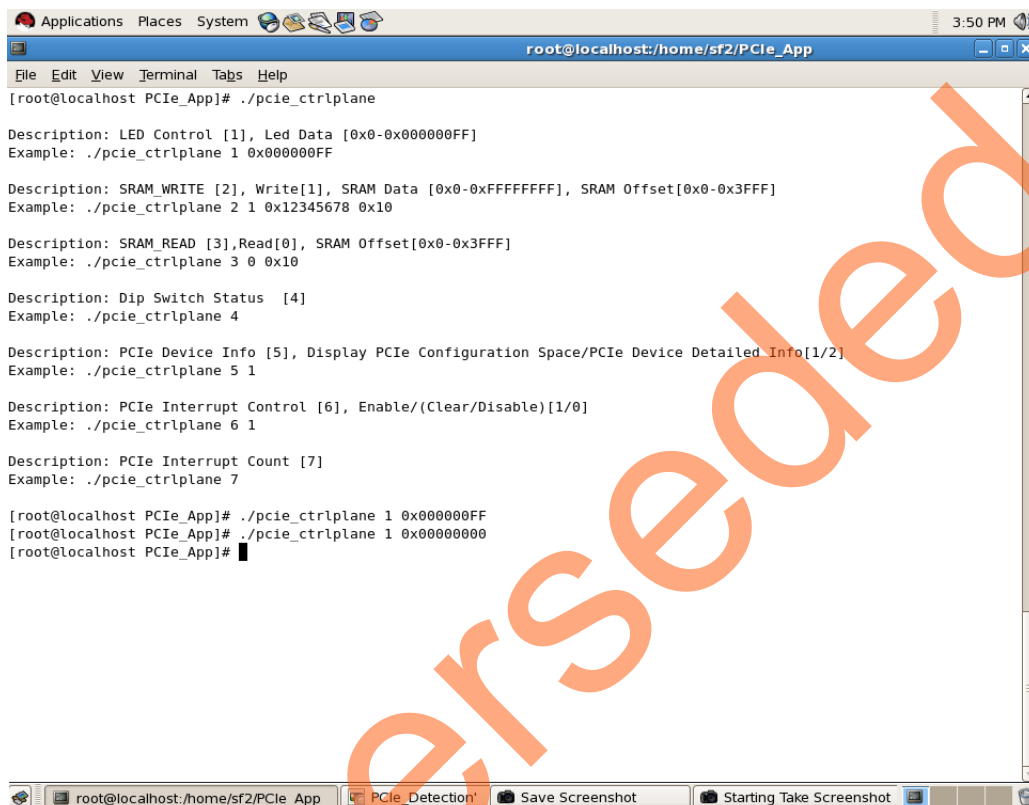
```
#./pcie_ctrlplane
```
7. Help menu displays as shown in Figure 72.

Execution of Linux PCIe Control Plane Features

LED Control

LED1 to LED8 is controlled by writing data to SmartFusion2 LED Control Registers.

```
#!/pcie_ctrlplane 1 0x000000FF [LED ON]
#!/pcie_ctrlplane 1 0x00000000 [LED OFF]
```



```
root@localhost: /home/sf2/PCie_App
File Edit View Terminal Tabs Help
[root@localhost PCie_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3], Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[root@localhost PCie_App]# ./pcie_ctrlplane 1 0x000000FF
[root@localhost PCie_App]# ./pcie_ctrlplane 1 0x00000000
[root@localhost PCie_App]#
```

Figure 73 • Linux Command - LED Control

led_blink.sh, contains the shell script code to perform LED Walk ON where as ctrl c exits the shell script and LED Walk turns OFF.

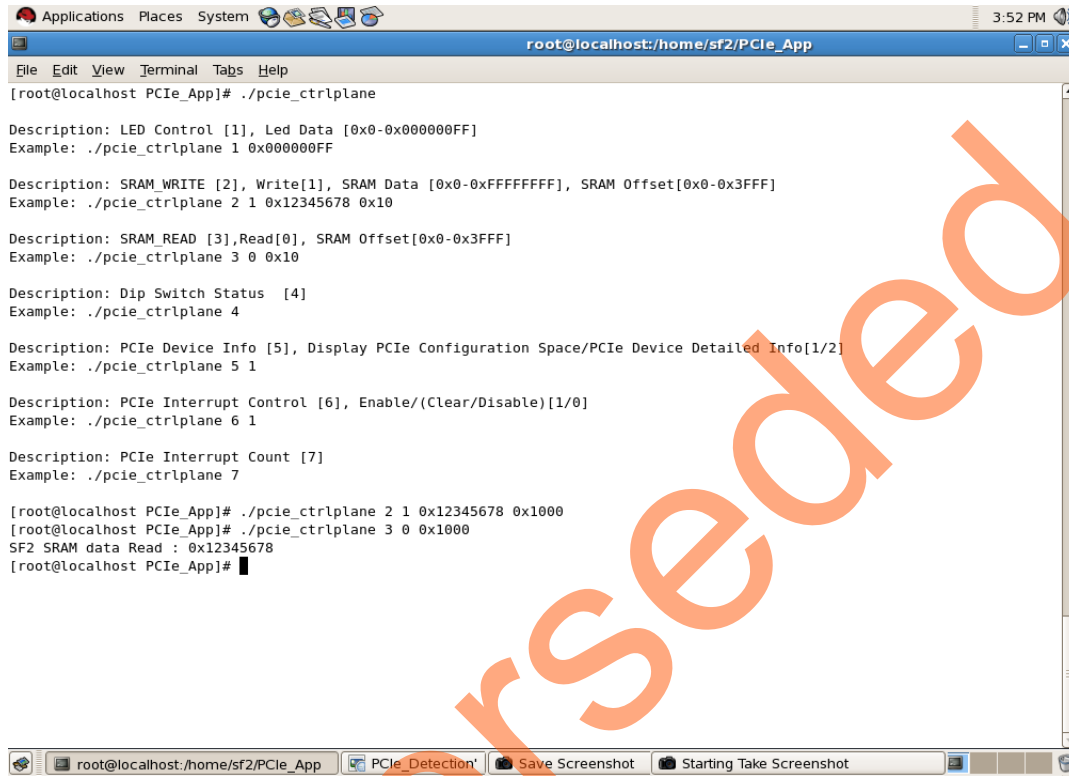
Run the led_blink.sh shell script using sh command.

```
#sh led_blink.sh
```

SRAM Read/Write

64 KB SRAM is accessible for SmartFusion2 Development Kit.

```
#./pcie_ctrlplane 2 1 0xFF00FF00 0x1000 [SRAM WRITE]
#./pcie_ctrlplane 3 0 0x1000 [SRAM READ]
```



```

Applications Places System
root@localhost:/home/sf2/PCie_App
File Edit View Terminal Tabs Help
[root@localhost PCie_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3],Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[root@localhost PCie_App]# ./pcie_ctrlplane 2 1 0x12345678 0x1000
[root@localhost PCie_App]# ./pcie_ctrlplane 3 0 0x1000
SF2 SRAM data Read : 0x12345678
[root@localhost PCie_App]#

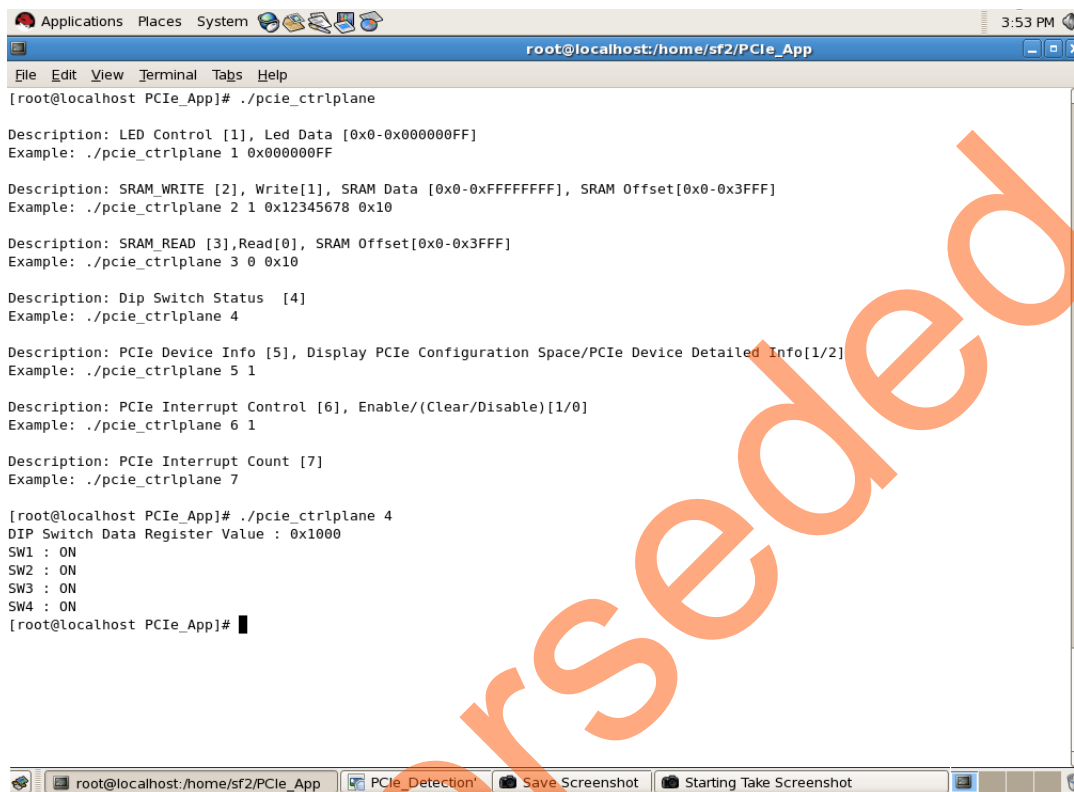
```

Figure 74 • Linux Command - SRAM Read/Write

DIP Switch Status

Dip Switch on SmartFusion2 Development Kit consists of 4 electric switches to hold the device configurations. Linux PCIe utility reads the corresponding switches (ON/OFF) state.

```
#./pcie_ctrlplane 4 [DIP Switch Status]
```



```
root@localhost:~/home/sf2/PCie_App
File Edit View Terminal Tabs Help
root@localhost PCie_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3],Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[root@localhost PCie_App]# ./pcie_ctrlplane 4
DIP Switch Data Register Value : 0x1000
SW1 : ON
SW2 : ON
SW3 : ON
SW4 : ON
[root@localhost PCie_App]#
```

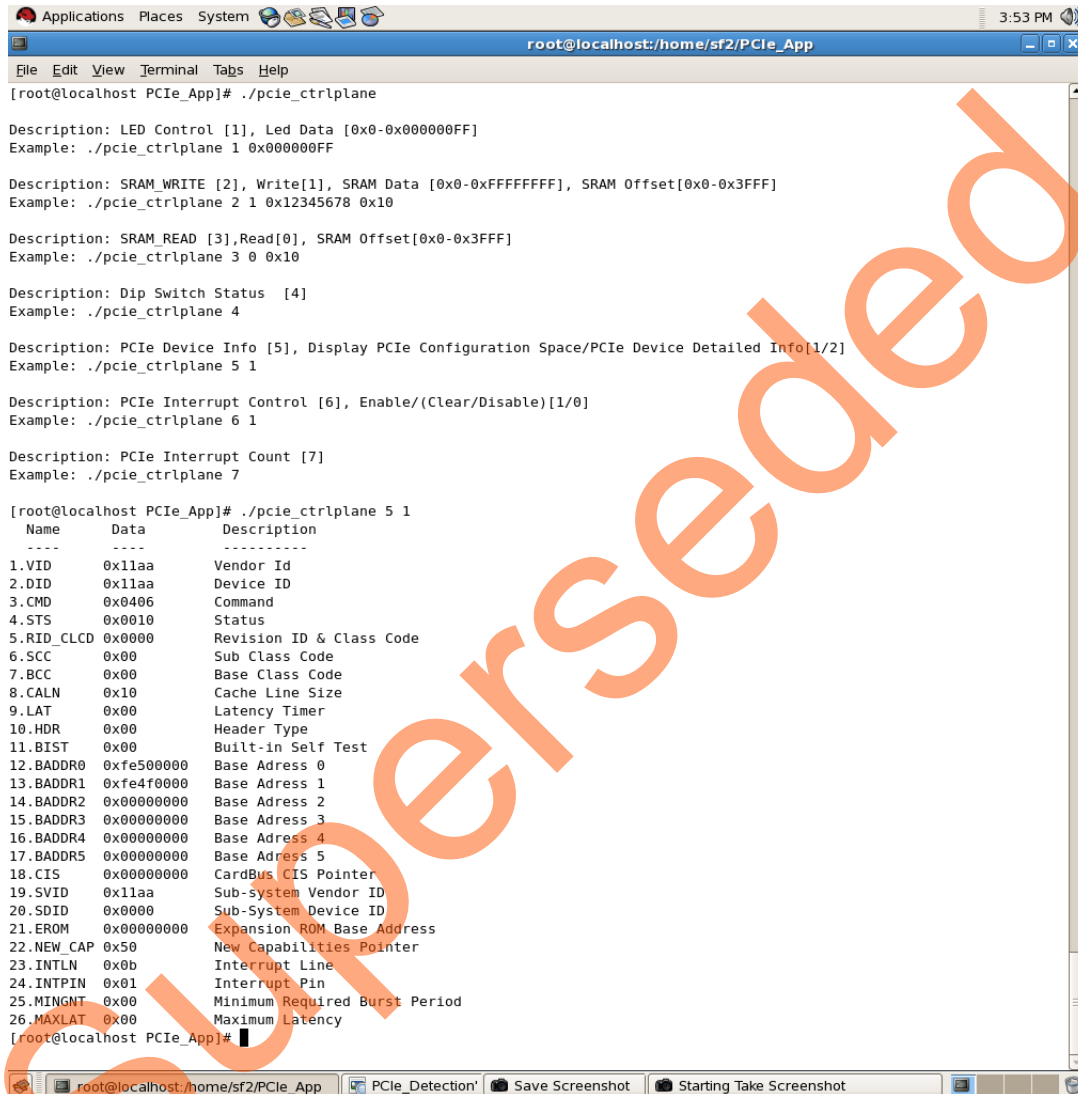
Figure 75 • Linux Command - DIP Switch

PCIe Configuration Space Display

PCIe Configuration Space contains the PCIe device data, such as Vendor ID, Device ID, Base Address 0.

Note: Root Privileges are required to execute this command.

```
#./pcie_ctrlplane 5 1 [Read PCIe Configuration Space]
```



```

root@localhost:/home/sf2/PCIE_App
[ root@localhost PCIE_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3],Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[ root@localhost PCIE_App]# ./pcie_ctrlplane 5 1
Name      Data      Description
-----
1.VID     0x11aa    Vendor Id
2.DID     0x11aa    Device ID
3.CMD     0x0406    Command
4.STS     0x0010    Status
5.RID_CLCD 0x0000    Revision ID & Class Code
6.SCC     0x00      Sub Class Code
7.BCC     0x00      Base Class Code
8.CALN    0x10      Cache Line Size
9.LAT     0x00      Latency Timer
10.HDR     0x00      Header Type
11.BIST    0x00      Built-in Self Test
12.BADDR0 0xfe500000 Base Address 0
13.BADDR1 0xfe4f0000 Base Address 1
14.BADDR2 0x00000000 Base Address 2
15.BADDR3 0x00000000 Base Address 3
16.BADDR4 0x00000000 Base Address 4
17.BADDR5 0x00000000 Base Address 5
18.CIS     0x00000000 CardBus CIS Pointer
19.SVID    0x11aa    Sub-system Vendor ID
20.SDID    0x0000    Sub-System Device ID
21.EROM    0x00000000 Expansion ROM Base Address
22.NEW_CAP 0x50      New Capabilities Pointer
23.INTLN   0x0b      Interrupt Line
24.INTPIN  0x01      Interrupt Pin
25.MINGNT  0x00      Minimum Required Burst Period
26.MAXLAT  0x00      Maximum Latency
[ root@localhost PCIE_App]#

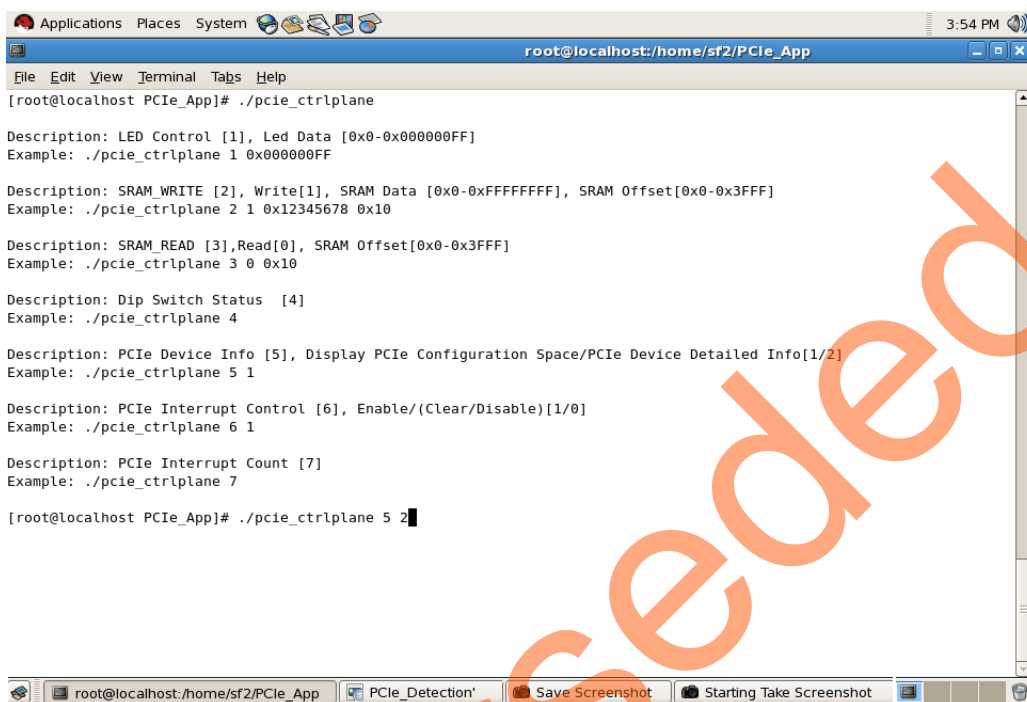
```

Figure 76 • Linux Command - PCIe Configuration Space Display

PCIe Link Speed and Width

Note: Root Privileges are required to execute this command.

```
#./pcie_ctrlplane 5 2 [Read PCIe Link Speed and Link Width]
```



```
root@localhost:~# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3], Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

root@localhost PCIe_App]# ./pcie_ctrlplane 5 2
```

Figure 77 • Linux Command - PCIe Link Speed and Width

```

Applications Places System 3:54 PM
root@localhost:/home/sf2/PCIe_App

File Edit View Terminal Tabs Help

Kernel driver in use: i801_smbus
Kernel modules: i2c-i801

01:00.0 Non-VGA unclassified device: Actel Device 11aa
Subsystem: Actel Device 0000
Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 64 bytes
Interrupt: pin A routed to IRQ 74
Region 0: Memory at fe500000 (32-bit, non-prefetchable) [size=1M]
Region 1: Memory at fe4f0000 (32-bit, non-prefetchable) [size=64K]
Capabilities: [50] MSI: Enable+ Count=1/1 Maskable- 64bit+
Address: 00000000fee00000 Data: 404a
Capabilities: [78] Power Management version 3
Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0+,D1-,D2-,D3hot+,D3cold-)
Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
Capabilities: [80] Express (v2) Endpoint, MSI 01
DevCap: MaxPayload 256 bytes, PhantFunc 0, Latency L0s unlimited, L1 unlimited
ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset-
DevCtl: Report errors: Correctable- Non-Fatal+ Fatal+ Unsupported-
Rlxd0rd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+
MaxPayload 128 bytes, MaxReadReq 512 bytes
DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
LnkCap: Port #1, Speed 5GT/s, Width x4, ASPM L0s L1, Latency L0 <64ns, L1 <16us
ClockPM+ Surprise- LLActRep- BwNot-
LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- Retrain- CommClk+
ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
LnkSta: Speed 2.5GT/s, Width x4, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
DevCap2: Completion Timeout: Range ABCD, TimeoutDis-
DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-
LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-, Selectable De-emphasis: -6dB
Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
Compliance De-emphasis: -6dB
LnkSta2: Current De-emphasis Level: -3.5dB
Capabilities: [100 v1] Virtual Channel
Caps: LPEVC=0 RefClk=100ns PATEntryBits=1
Arb: Fixed- WRR32- WRR64- WRR128-
Ctrl: ArbSelect=Fixed
Status: InProgress-
VC0: Caps: PATOffset=00 MaxTimeSlots=1 RejSnoopTrans-
Arb: Fixed- WRR32- WRR64- WRR128- TWRR128- WRR256-
Ctrl: Enable+ ID=0 ArbSelect=Fixed TC/VC=01
Status: NegoPending- InProgress-
Capabilities: [800 v1] Advanced Error Reporting
UESta: DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
UEMsk: DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- UnxCmplt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-
CESta: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr-
CEMsk: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
AERCap: First Error Pointer: 00, GenCap+ CGenEn- ChkCap+ ChkEn-
Kernel driver in use: MS_PCI_DRIVER

02:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57780 Gigabit Ethernet PCIe (rev 01)
Subsystem: Dell Device 0400

root@localhost:/home/sf2/PCIe_App [PCIE_Detection] Save Screenshot Starting Take Screenshot

```

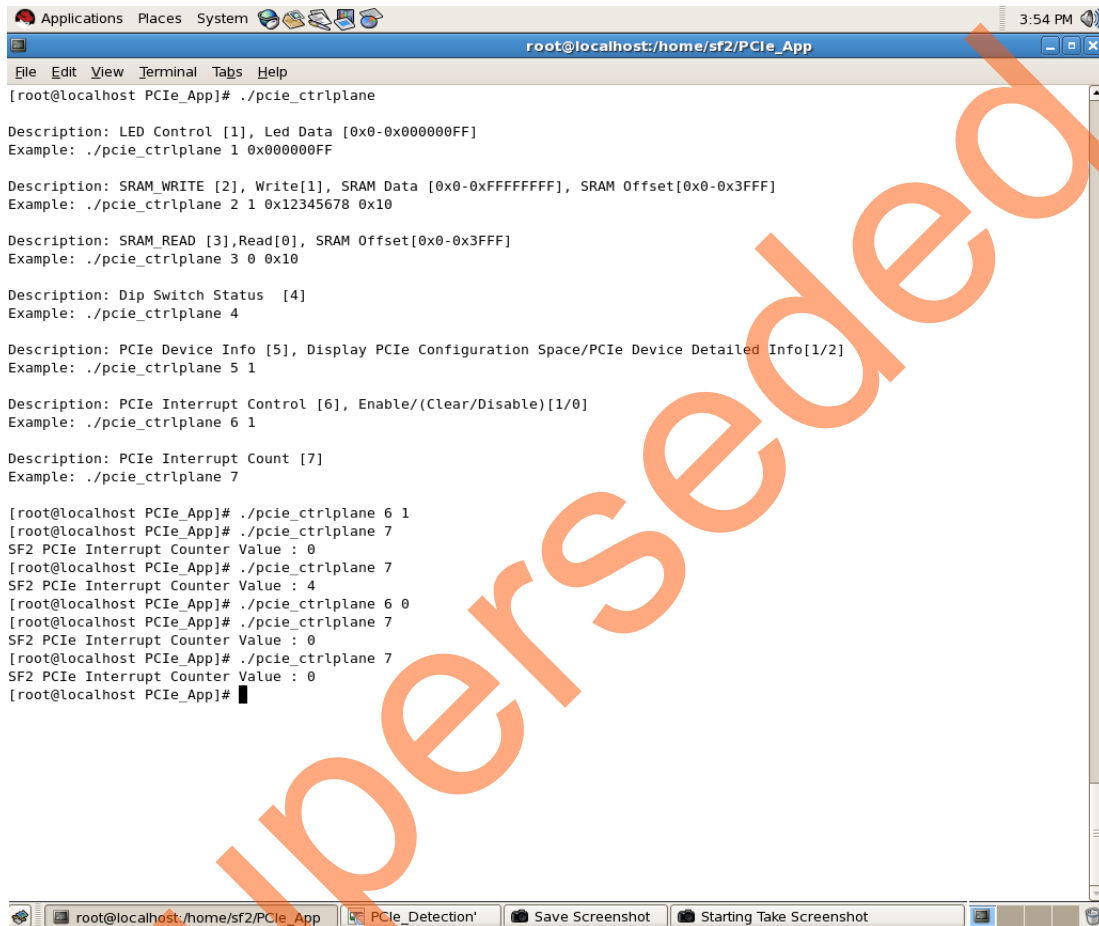
Figure 78 • Linux Command - PCIe Link Speed and Width

PCIe Interrupt Control (Enable/Disable) and Interrupt Counter

SmartFusion2 Development Kit enable/disable the MSI interrupts by writing data to its PCIe configuration space.

Interrupt Counter holds the number of MSI interrupts got triggered by pressing the SW3 Push Button.

```
#. /pcie_ctrlplane 6 0 [Disable Interrupts]
#. /pcie_ctrlplane 6 1 [Enable Interrupts]
#. /pcie_ctrlplane 7 [Interrupt Counter Value]
```



```
root@localhost: /home/sf2/PCie_App
File Edit View Terminal Tabs Help
[root@localhost PCie_App]# ./pcie_ctrlplane

Description: LED Control [1], Led Data [0x0-0x000000FF]
Example: ./pcie_ctrlplane 1 0x000000FF

Description: SRAM_WRITE [2], Write[1], SRAM Data [0x0-0xFFFFFFFF], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 2 1 0x12345678 0x10

Description: SRAM_READ [3], Read[0], SRAM Offset[0x0-0x3FFF]
Example: ./pcie_ctrlplane 3 0 0x10

Description: Dip Switch Status [4]
Example: ./pcie_ctrlplane 4

Description: PCIe Device Info [5], Display PCIe Configuration Space/PCIe Device Detailed Info[1/2]
Example: ./pcie_ctrlplane 5 1

Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

Description: PCIe Interrupt Count [7]
Example: ./pcie_ctrlplane 7

[root@localhost PCie_App]# ./pcie_ctrlplane 6 1
[root@localhost PCie_App]# ./pcie_ctrlplane 7
SF2 PCIe Interrupt Counter Value : 0
[root@localhost PCie_App]# ./pcie_ctrlplane 7
SF2 PCIe Interrupt Counter Value : 4
[root@localhost PCie_App]# ./pcie_ctrlplane 6 0
[root@localhost PCie_App]# ./pcie_ctrlplane 7
SF2 PCIe Interrupt Counter Value : 0
[root@localhost PCie_App]# ./pcie_ctrlplane 7
SF2 PCIe Interrupt Counter Value : 0
[root@localhost PCie_App]#
```

Figure 79 • Linux Command - PCIe Interrupt Control

Conclusion

This demo describes how to access the PCIe endpoint features of SmartFusion2 and how to create a simple design and verify the design using BFM simulation. This demo demonstrates that the Host PC can easily communicate with SmartFusion2 Development Kit through the provided GUI and Drivers. This demo also provides a Linux PCIe application for accessing PCIe EP device through Linux PCIe Device Driver.

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 (April 2014)	Updated the document for Libero v11.3 software release (SAR 56081).	NA
Revision 1 (December 2013)	Updated the document for Libero v11.2 software release (SAR 52109) (SAR 52909) and (SAR 50779).	NA
Revision 0 (June 2013)	Initial Release	NA

Superseded

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.

Superseded

Superseded



Microsemi®

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

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and 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.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.