Implementing PCIe Control Plane Design

Libero SoC v11.4 Flow Tutorial for IGLOO2 FPGA

Microsemi

Table of Contents

Implementing PCIe Control Plane Design - Libero SoC v11.4 Flow Tutorial for IGLOO2 FPGA

Introduction

This tutorial demonstrates the embedded PCI®express feature of IGLOO®2 field programmable gate array (FPGA) devices and how this can be used as a low bandwidth control plane interface. A sample design is provided to access IGLOO2 PCIe Endpoint (EP) from host PC. It can run on both Windows and RedHat Linux Operating System (OS). A GUI installer, host PC drivers for Windows OS, and a Linux PCIe application for Linux OS are provided for reading and writing to the IGLOO2 PCIe configuration and memory space. This tutorial 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® System-on-Chip (SoC) project
- Develop the Simulation Stimulus
- Simulate the design
- Generate the programming file
- Run the PCIe application

Tutorial Requirements

Table 1 lists the hardware and software requirements of IGLOO2 PCIe Control Plane tutorial.

Table 1 • **Reference Design Requirements and Details**

Reference Design Requirements and Details	Description
Hardware Requirements	
IGLOO2 Evaluation Kit • 12 V adapter • FlashPro4 programmer • USB A to Mini-B cable	Rev C or later
Host PC or Laptop with an available PCIe 2.0 Gen 1 or Gen 2 compliant slot	64-bit Windows 7 OS, 64-bit Red Hat Linux OS (Kernel Version: 2.6.18-308)
Express Card slot and PCIe Express card adapter (for Laptop only)	-
Software Requirements	
Libero® System-on-Chip (SoC)	v11.4
FlashPro programming software	v11.4
Host PC Drivers (provided along with the design files)	-
GUI executable (provided along with the design files)	-

Note: PCIe Express card adapter is not supplied with the IGLOO2 Evaluation Kit.

Associated Project Files

You can download the associated project files for this tutorial from the Microsemi® website:
http://soc.microsemi.com/download/rsc/?f=M2GL_PCIE_Control_Plane_11p4_DF.

Design files include:

1. Libero project
2. Programming files
3. Host PC drivers and GUI executable for Windows OS
4. Host PC drivers and PCIe application for Linux OS
5. Readme file
6. Source files

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

Components Used

This tutorial uses the following components of the IGLOO2 device:

- Fabric clock conditioning circuitry (CCC)
- High speed serial interfaces (SERDES_IF_0)
- CoreGPIO
- CoreAHBLSRAM
- Bus interfaces CoreAHBLite, CoreAPB3, and CoreAHBTOAPB3

Design Overview

IGLOO2 FPGA devices integrate a fourth-generation flash-based FPGA fabric and high performance communication interfaces on a single chip. The IGLOO2 high speed serial interface (SERDESIF) provides a fully hardened PCIe EP implementation and is compliant with PCIe Base Specification Revision 2.0 and 1.1. For more details refer to the *IGLOO2 FPGA High Speed Serial Interfaces User Guide*.

The design helps access the IGLOO2 PCIe EP from the host PC. A GUI and Linux PCIe application are provided for read and write access to the IGLOO2 PCIe configuration and memory space of BAR0 and BAR1. The IGLOO2 PCIe BAR0 and BAR1 are configured in 32-bit mode.

Figure 1 shows a detailed block diagram of the design implementation.

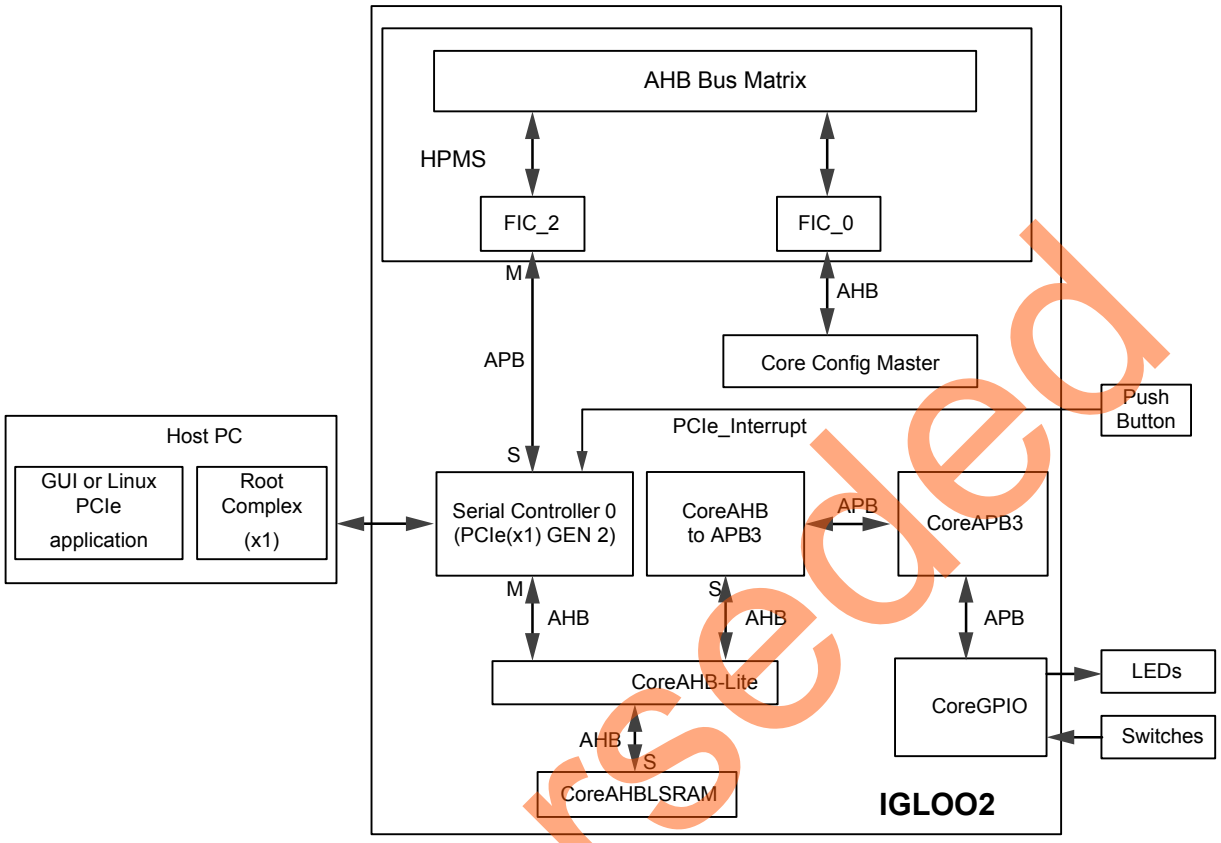


***Figure 1* • PCIe Control Plane Block Diagram**

The PCIe EP device receives commands from the host PC through the GUI or Linux PCIe application and performs corresponding memory writes to the IGLOO2 fabric address space.

The SERDES_IF_0 is configured for a PCIe 2.0, x1 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 slaves CoreAHBLSRAM and CoreGPIO using CoreAHBLite, CoreAHBTOAPB, and CoreAPB3 bus interfaces.

SERDES_IF_0 is initialized by CoreConfig Master. This IP is configured by the System Builder.

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 the IGLOO2 AHB bus matrix address space. The AXI master window 0 is enabled and configured to translate the BAR0 memory address space to the CoreGPIO address space to control the LEDs and DIP switches.

The AXI master window 1 is enabled and configured to translate the BAR1 memory address space to the CoreAHBLSRAM address space to perform read and writes from PCIe.

CoreGPIO is enabled and configured as below:

- GPIO_OUT [7:0] connected LEDs
- GPIO_IN [3:0] connected to DIP switches

The PCIe interrupt line is connected to the **SW4** push button on the IGLOO2 Evaluation Kit. The FPGA clocks are configured to run the FPGA fabric and HPMS at 100 MHz.

Step 1: Creating a Libero SoC Project

This step helps you create an IGLOO2 PCIe control plane design using the Libero tool.

Launching Libero SoC

1. Click **Start > Programs > Microsemi Libero SoC v11.4 > Libero SoC v11.4**, or click the shortcut on your desktop to open the Libero SoC Project Manager.
2. Create a new project by selecting **New** on the **Start Page** tab (highlighted in Figure 2), or by clicking **Project > New Project** from the Libero SoC menu.

***Figure 2* • Libero SoC Project Manager**

3. Enter the information as required for the new project and the device in the **New Project** dialog box as shown in Figure 3.
   - **Project**
     - Name: PCIE_Demo
     - Location: Select an appropriate location (for example, *D:/microsemi_prj*)
     - Preferred HDL type: Select Verilog or VHDL

– **Device** (select the following values using the drop-down list provided):
  - Family: IGLOO2
  - Die: M2GL010T
  - Package: 484 FBGA
  - Speed: -1
  - Core Voltage: 1.2
– Operating conditions: COM

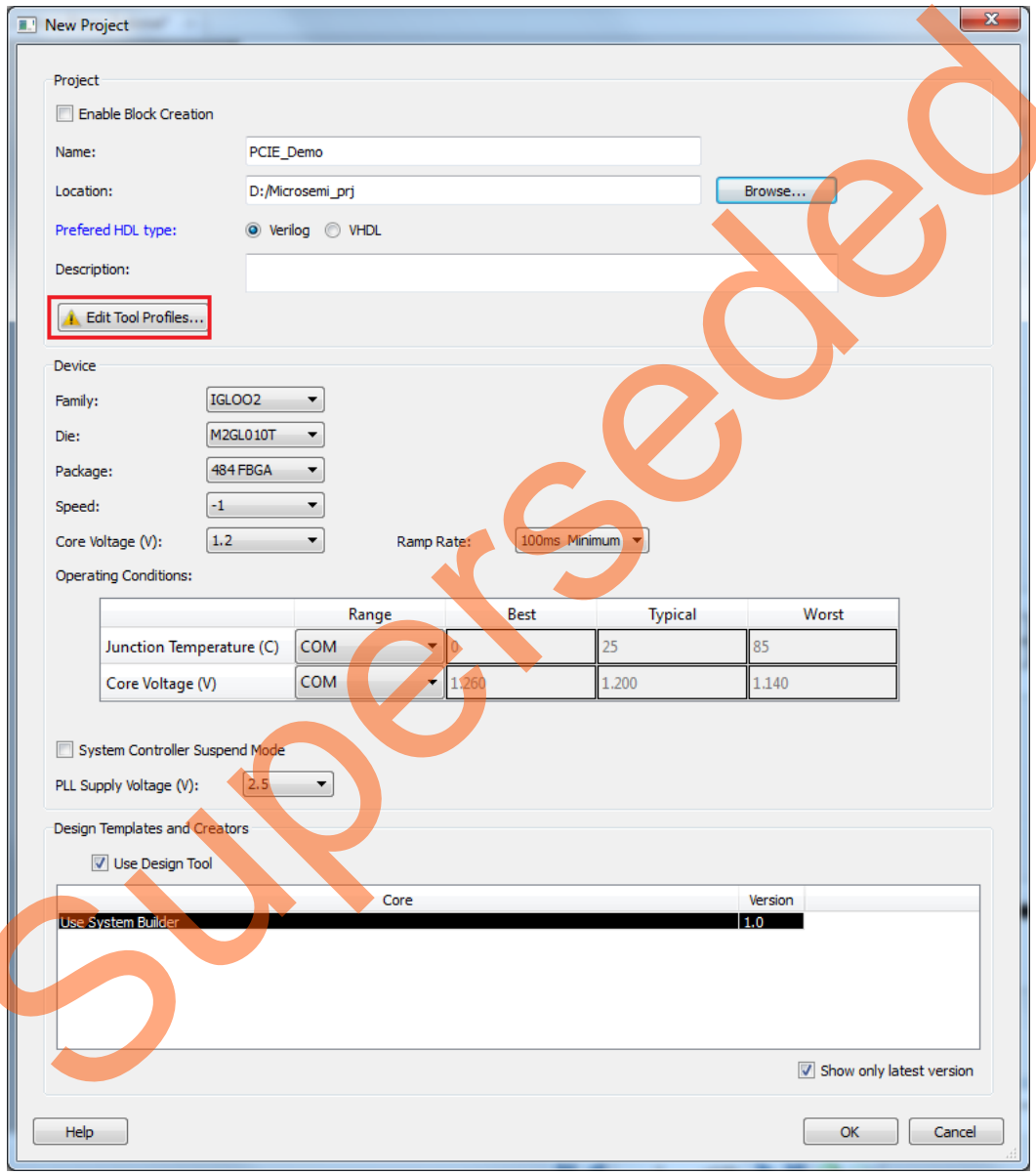


***Figure 3* • Libero SoC New Project Dialog Box**

4. Clicking **Edit Tool Profiles** (highlighted in Figure 3) displays the **Tool Profiles** window as shown in Figure 4. Check the below tool settings:
   – Synthesis: Synplify Pro ME I-2013.09M-SP1
   – Simulation: ModelSim 10.3a

– Programming: FlashPro 11.4

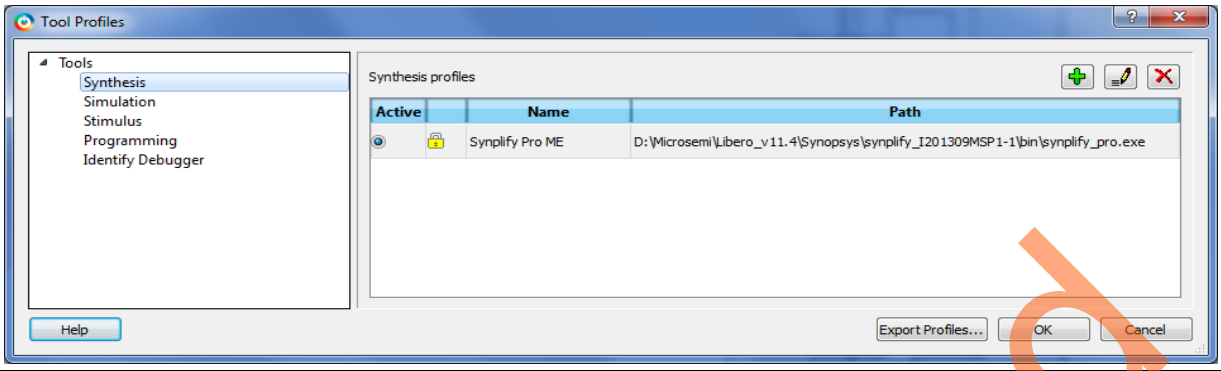


***Figure 4* • Tool Profiles**

5. Click **OK** on the **Tool Profiles** window.
6. Click **OK** on the **New Project** window. This displays the **System Builder** dialog box.
7. **Enter a name for your system** as shown in Figure 5.

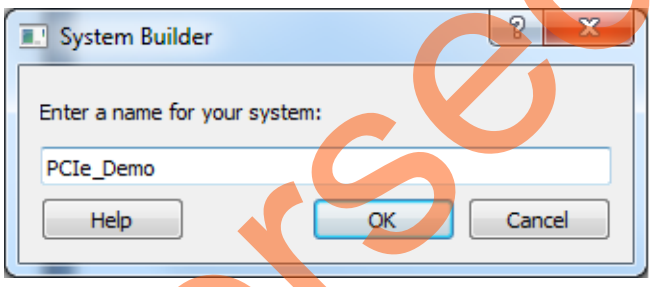


***Figure 5* • Create New System Builder Dialog Box**

8. 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.

9. In the **System Builder – Device Features page**, select **SERDESIF_0** under **High Speed Serial Interfaces** as shown in Figure 6.

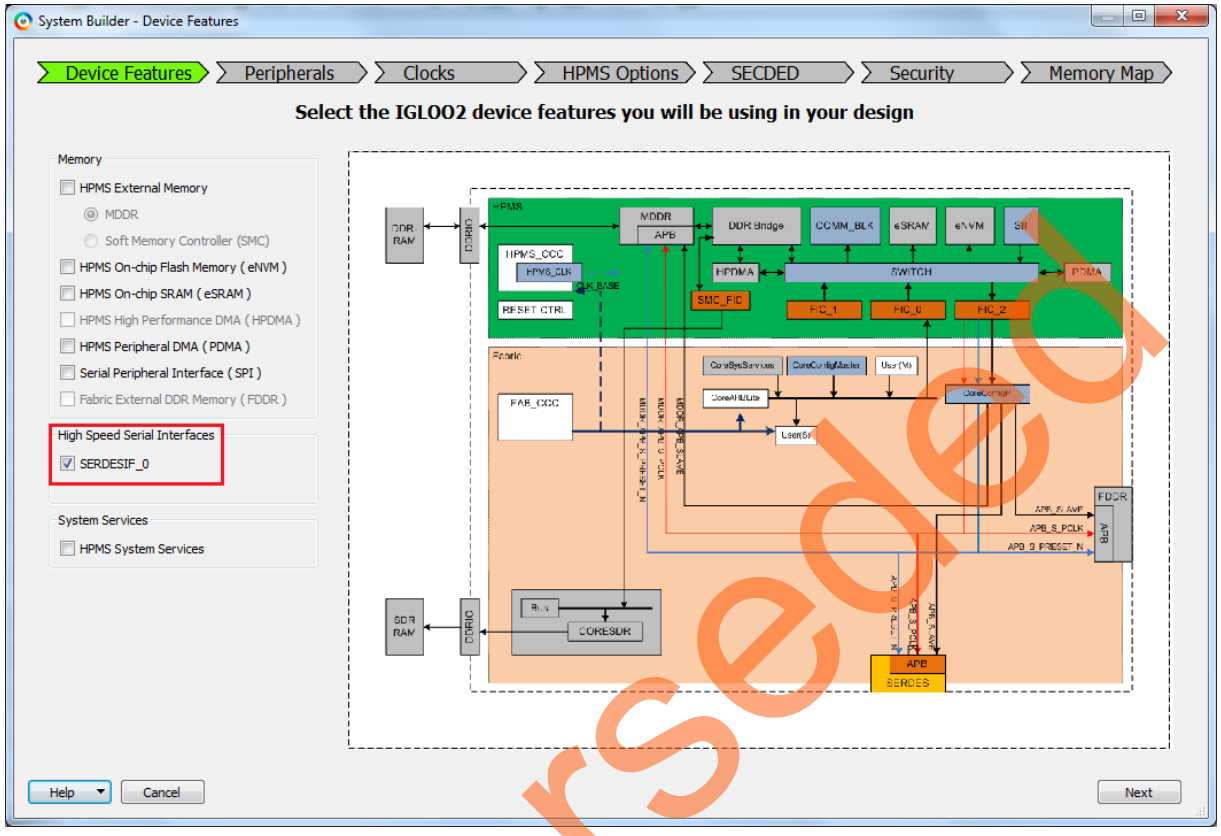


Figure 6 • **System Builder – Device Features Page**

10. Click **Next**. The **System Builder – Peripherals** page is displayed. Leave all the default selections.

11. Click **Next**. The **System Builder – Clock page** is displayed, as shown in Figure 7. Select **System Clock** source as **On-chip 25/50 MHz RC Oscillator** and **HPMS_CLK** as **100 MHz**.

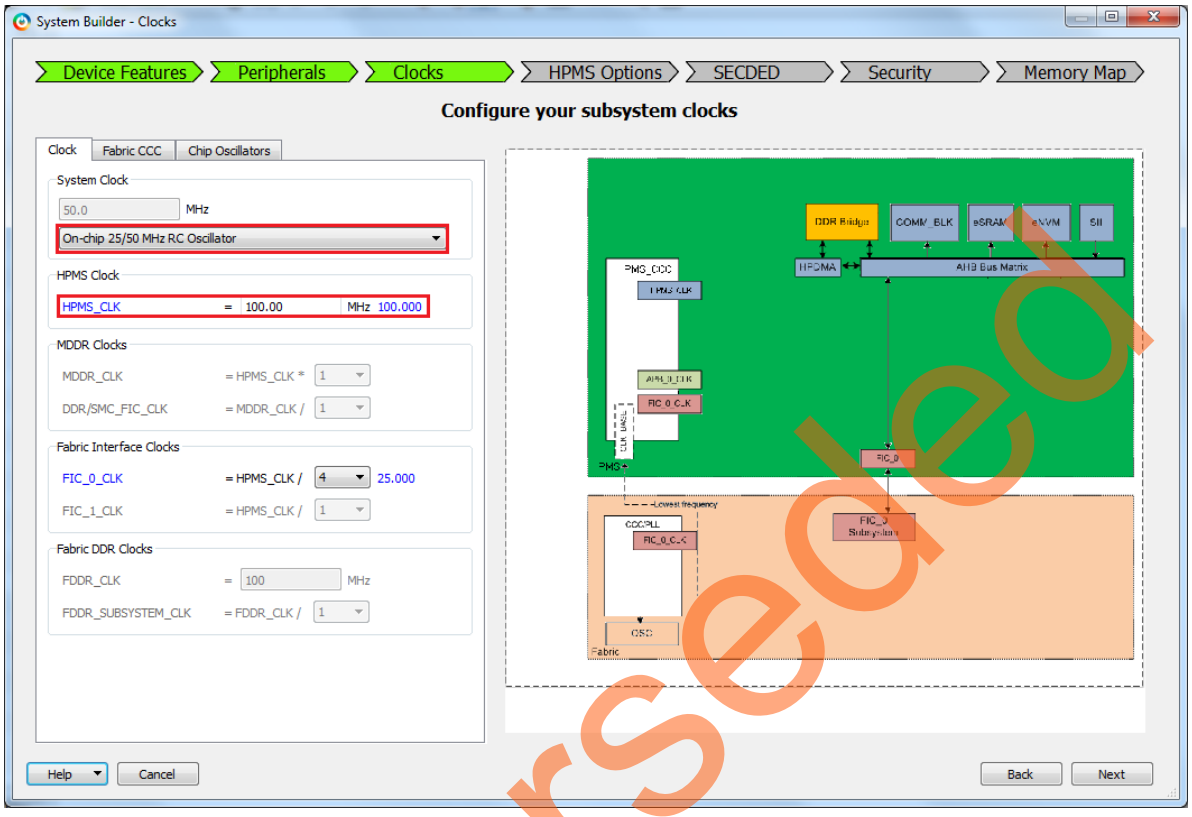


***Figure 7* • System Builder – Clocks Page**

12. Click **Next**. The **System Builder – HPMS Options page** is displayed. Leave all the default selections.
13. Click **Next**. The **System Builder – SECDED page** is displayed. Leave all the default selections.
14. Click **Next**. The **System Builder – Security page** is displayed. Leave all the default selections.
15. Click **Next**. The **System Builder – Memory Map page** is displayed. Leave all the default selections.
16. 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 8.

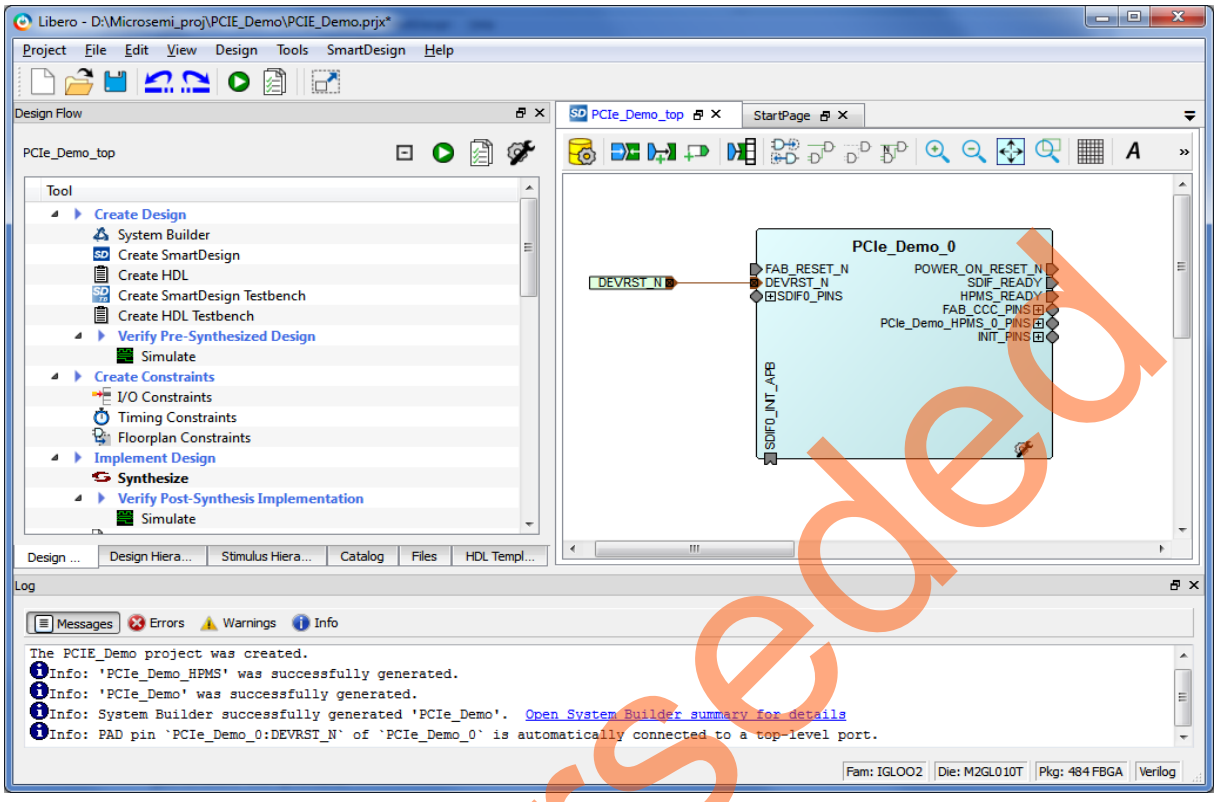


Figure 8 • **IGLOO2 FPGA System Builder Generated System**

The two soft cores (CoreResetP and CoreConfigP) are automatically instantiated and connected by the System Builder.

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

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**.

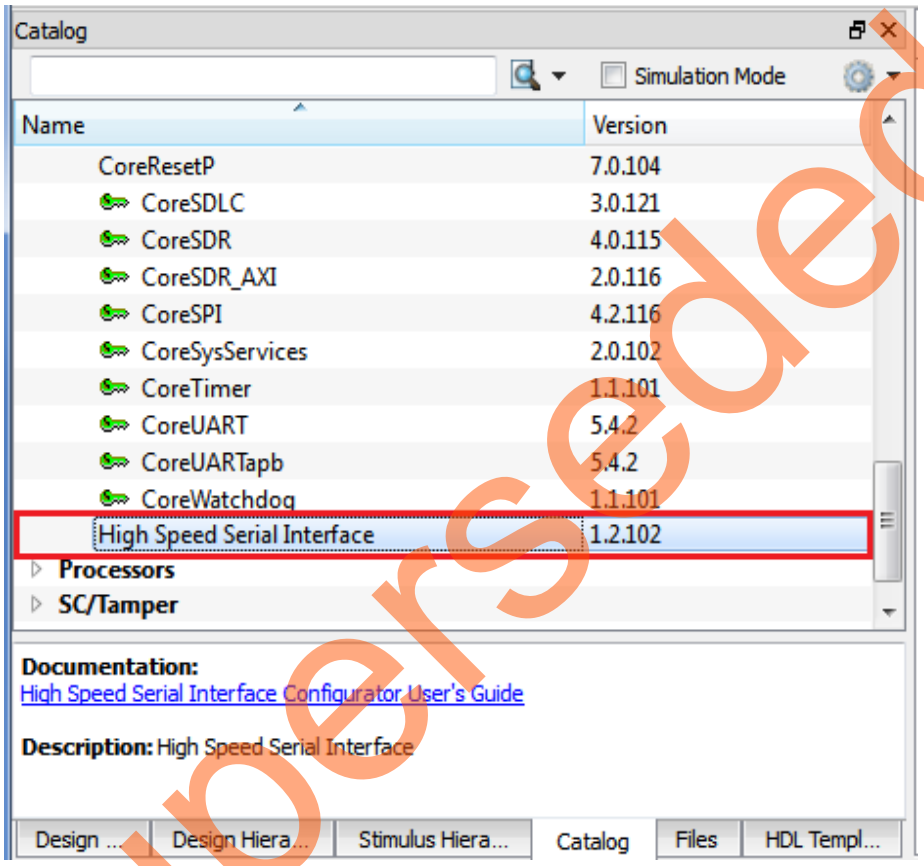


Figure 9 • **IP Catalog**

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 10:
   - **Identification**
     - Simulation Level: BFM PCIe
   - **Protocol Configuration**
     - Protocol1: Type: PCIe
     - Protocol1: Number of Lanes: x1
   - **Lane Configuration**
     - Speed: Lane0: 5.0 Gbps (Gen2)
   - **PCIe/XAUI Fabric SPLL Configuration**
     - CLK_BASE Frequency (MHz): 100

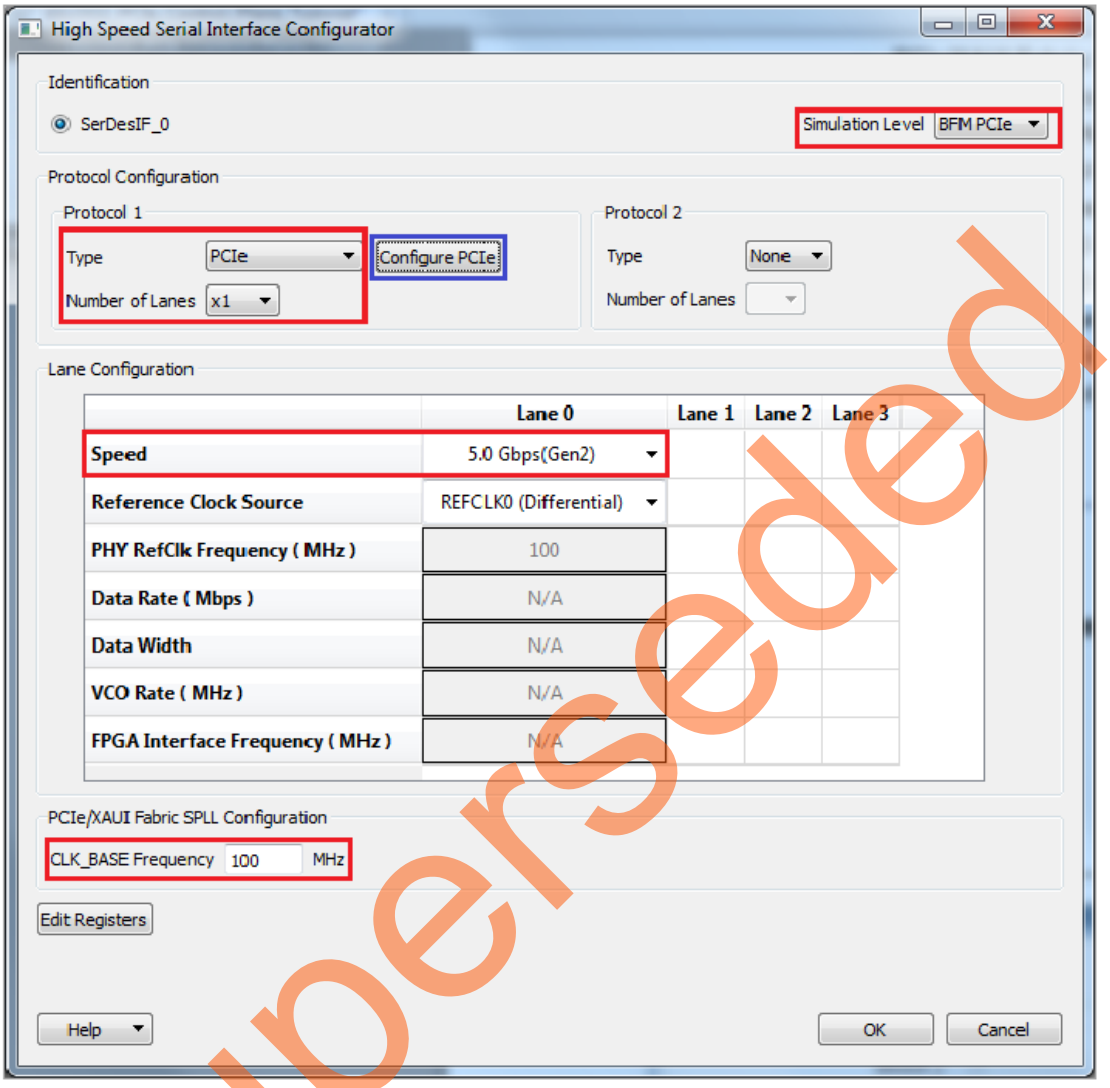


Figure 10 • **High Speed Serial Interfaces Configurator**

3. Click **Configure PCIe** to configure the following settings as shown in Figure 11.
   - Identification Registers
     - Device ID: 0x11AA (Microsemi ID)
     - Subsystem Vendor ID: 0x11AA (Microsemi ID)
   - **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 CoreGPIO address space)
     - Bar 1 Width: 32-bit, Size: 64 KB (to access CoreAHBLSRAM memory)

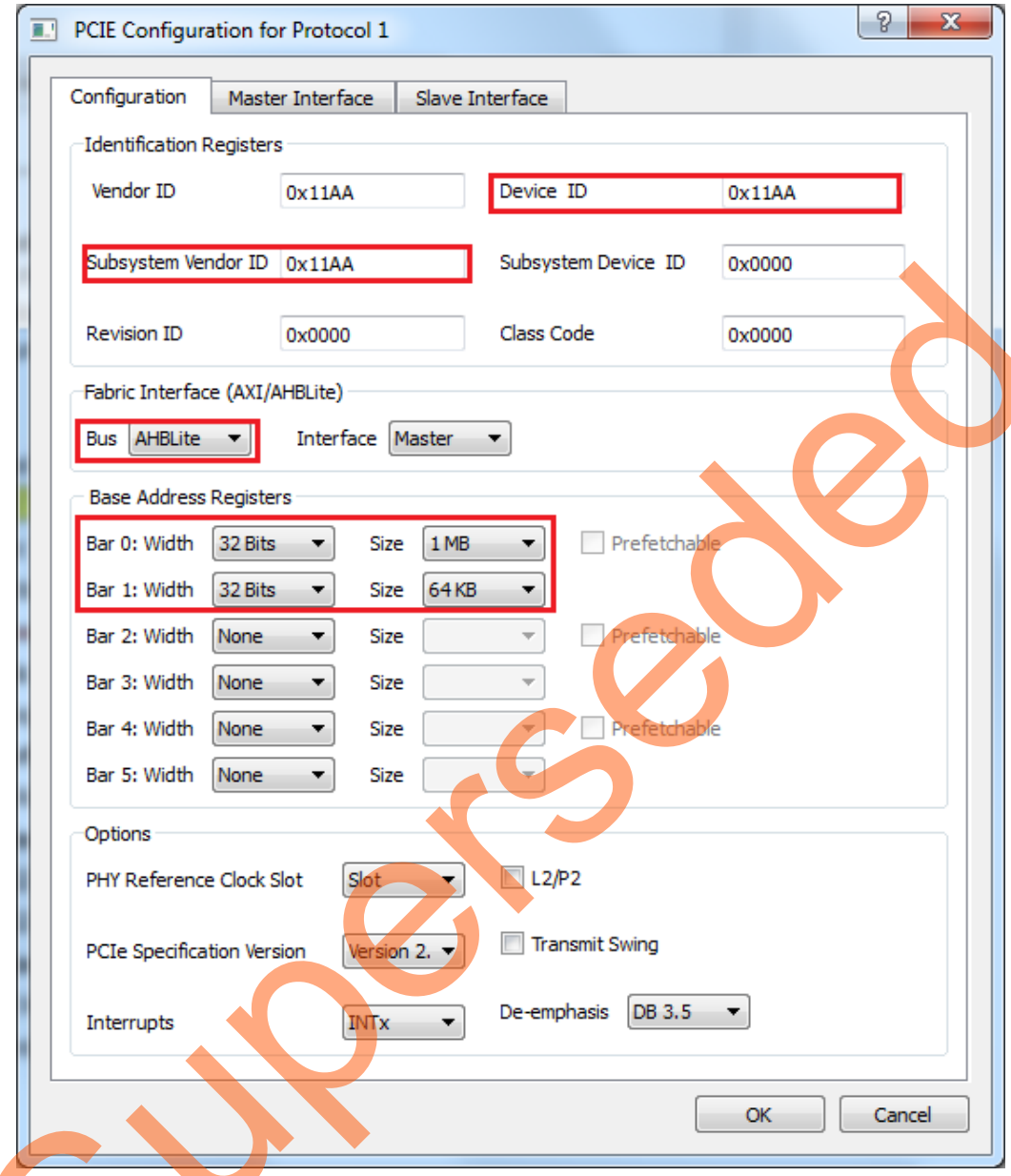


***Figure 11* • PCIE Configuration for Protocol 1**

4. Click **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 tutorial the PCIe AXI master windows are used to translate the address of BAR0 and BAR1 to CoreGPIO address and CoreAHBLSRAM address.
   - Select Window 0 and configure the 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 the 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 *IGLOO2 FPGA High Speed Serial Interfaces User Guide*.

   Figure 12 shows the **Master Interface Configuration** window.

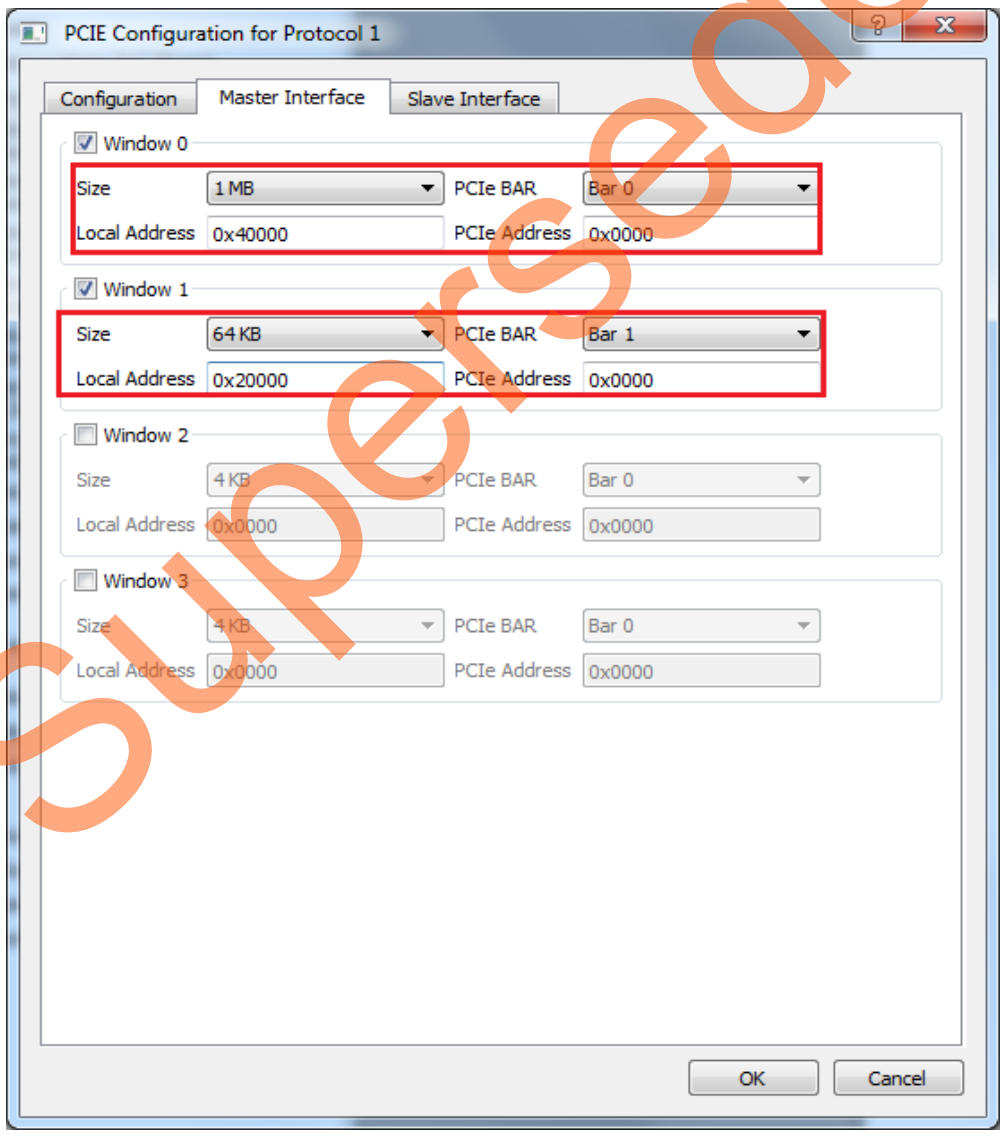


***Figure 12* • Master Interface Configuration Window**

5. Click **OK** to close the PCIE Configuration for protocol 1 window.
6. Click **OK** to save and close the **High Speed Serial Interface Configurator** window.

Instantiating Debounce Logic in PCIe_Demo_top SmartDesign

The tutorial provides a push button (**SW4**) on the IGLOO2 Evaluation 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.

1. To add the Debounce logic to the PCIe demo design, click **File > Import > HDL Source files**.
2. Browse to the **Debounce.v** or **Debounce.vhd** file location in the design files folder: *M2GL_PCIE_Control_Plane_DF\Source Files*. Figure 13 shows the **DEBOUNCE** component in the **Design Hierarchy** window.

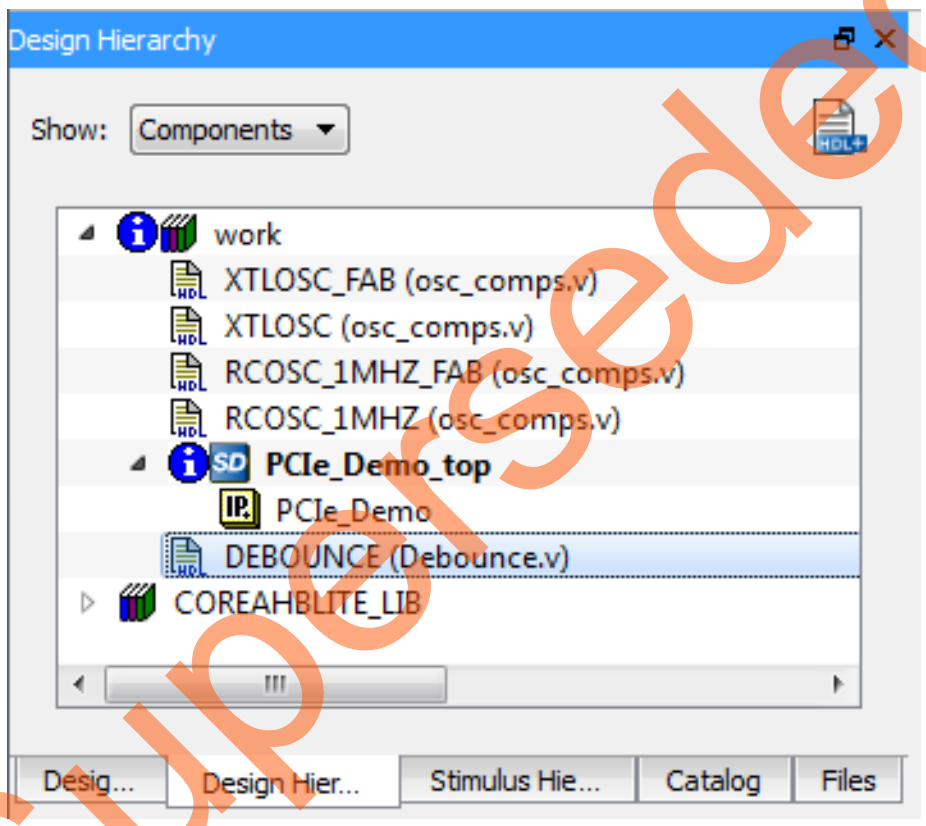


***Figure 13* • DEBOUNCE Component in the Design Hierarchy Window**

3. Drag the **DEBOUNCE** component from the **Design Hierarchy** into the **PCIe_Demo_top SmartDesign** canvas. Figure 14 shows Debounce in **PCIe_Demo_top**.

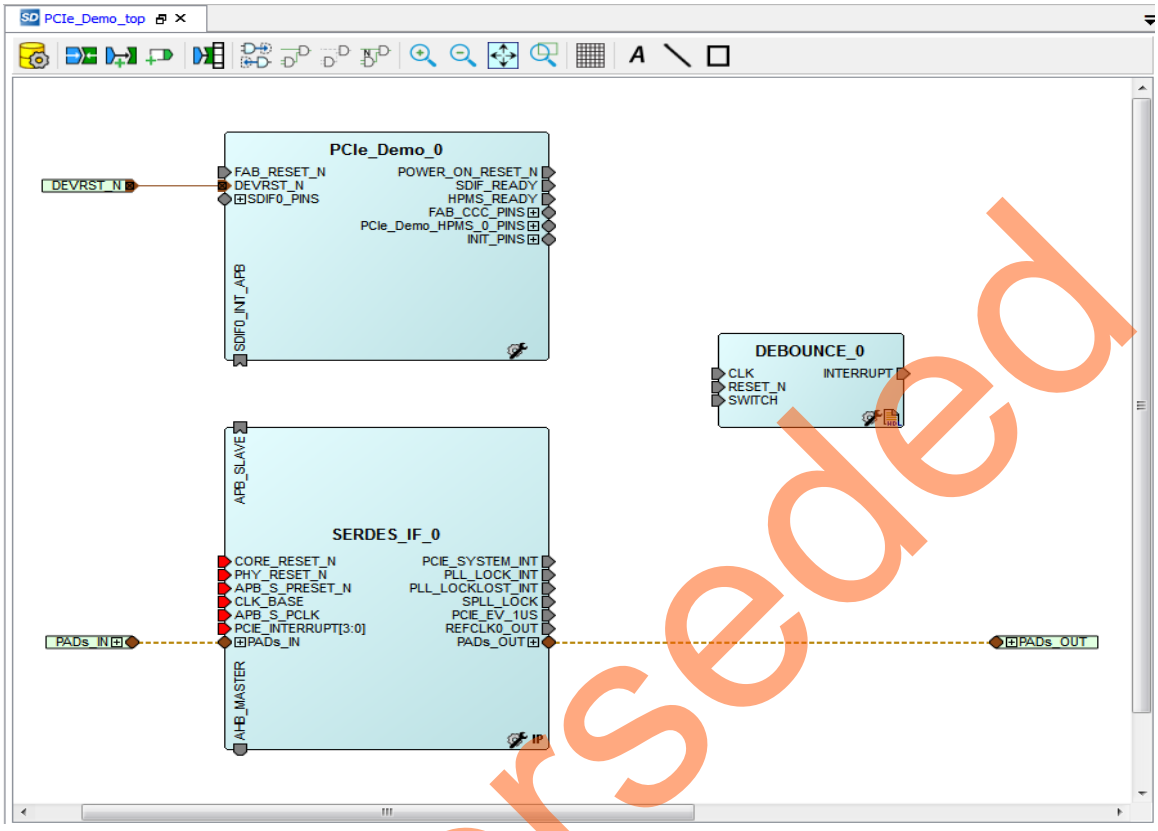


***Figure 14* • DEBOUNCE Component in the PCIe_Demo_top SmartDesign Canvas**

Instantiating the Bus Interfaces in PCIe_Demo_top SmartDesign

To instantiate the CoreAHBLite, CoreAPB3, and CoreAHBtoAPB3 in the PCIe_Demo_top SmartDesign, expand the **Bus Interfaces** category in the Libero SoC **Catalog**. Figure 15 shows the Libero IP **Catalog**.

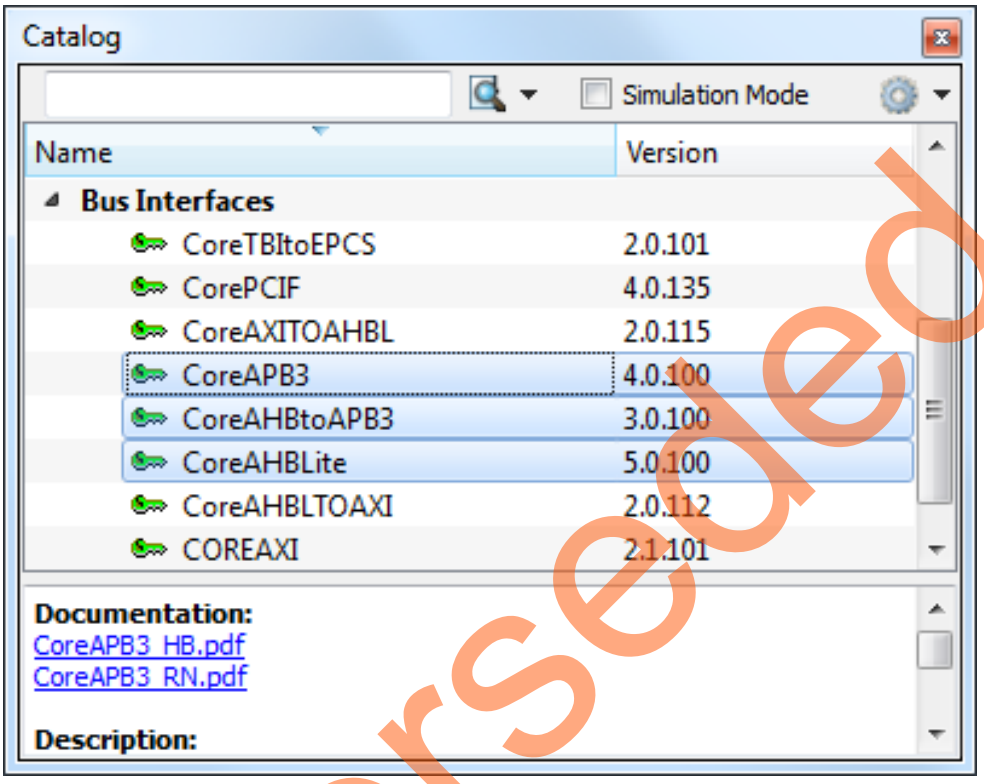


***Figure 15* • IP Catalog**

1. Drag CoreAHBLite, CoreAHBtoAPB3, and CoreAPB3 Bus interfaces into the PCIe_Demo_top SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download. Figure 16 shows the Libero top-level design with bus interfaces.

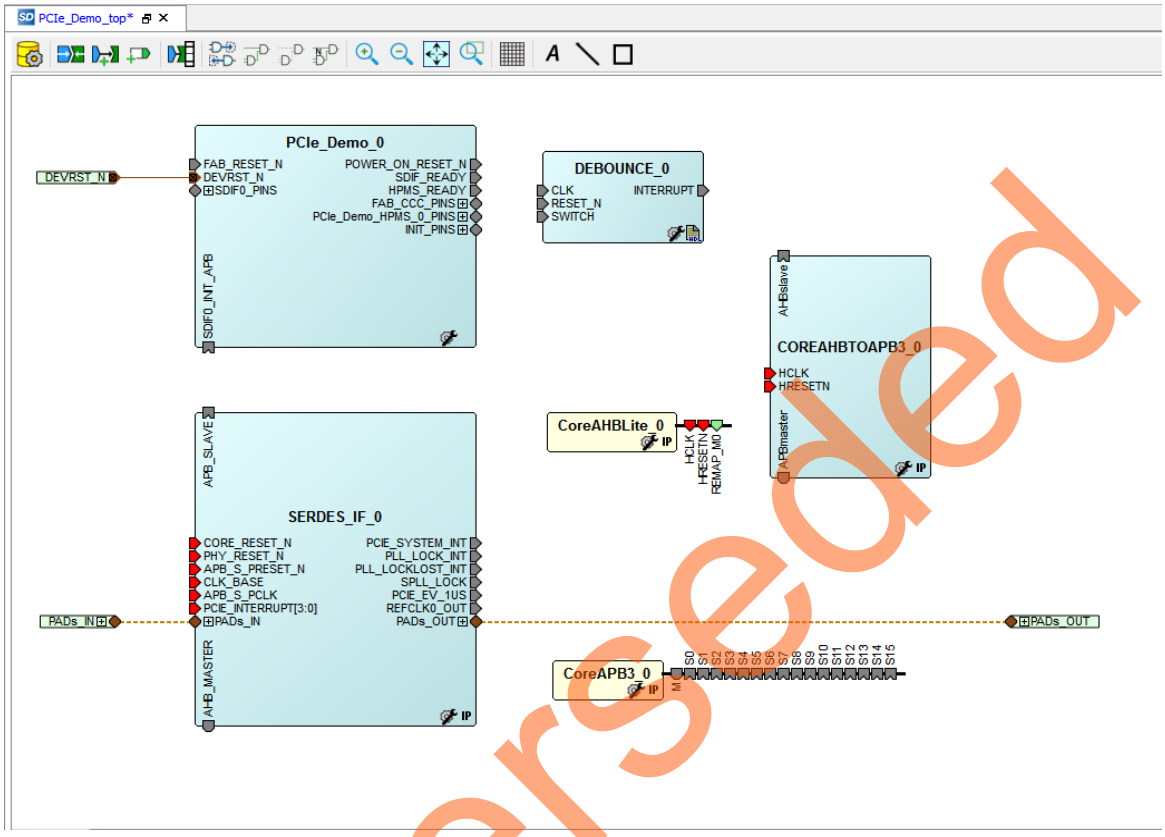


***Figure 16* • CoreAHBLite, CoreAHBtoAPB3, and CoreAPB3 Bus Interfaces in the PCIe_Demo_top SmartDesign Canvas**

2. Double-click **CoreAHBLite_0** to configure it. Figure 17 shows the **Configuring CoreAHBlite_0** window.

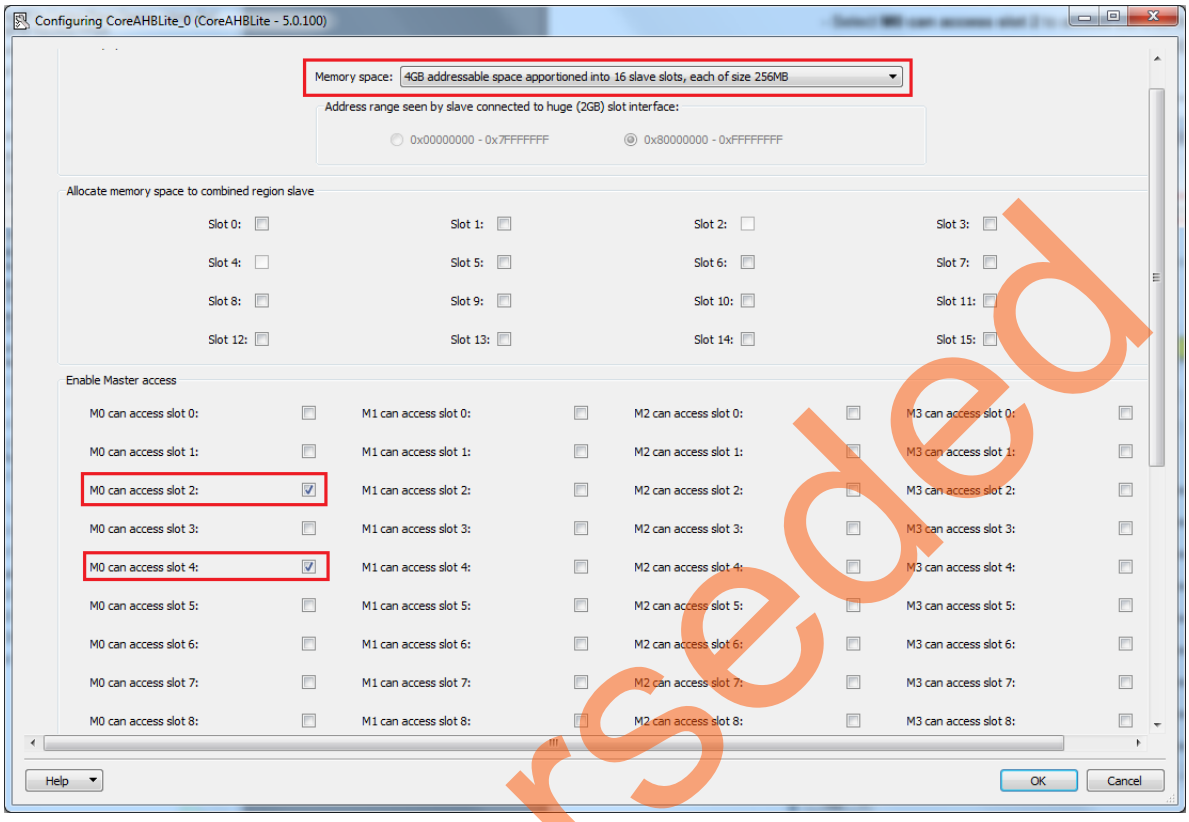


***Figure 17* • Configuring CoreAHBLite_0**

3. Configure **CoreAHBLite_0** with the below settings:
   - **Memory Space**: Select from the drop-down list as '**4 GB addressable space apportioned into 16 slave slots, each of size 256 MB**'.
   - Select **M0 can access slot 2** to access CoreAHBLSRAM from PCIe.
   - Select **M0 can access slot 4** to access CoreGPIO from PCIe.
4. Click **OK** to save and close the **Configuring CoreAHBLite_0** window.

5. Double-click CoreAPB3 to configure it. Figure 18 shows the **Configuring CoreAPB3_0** window.

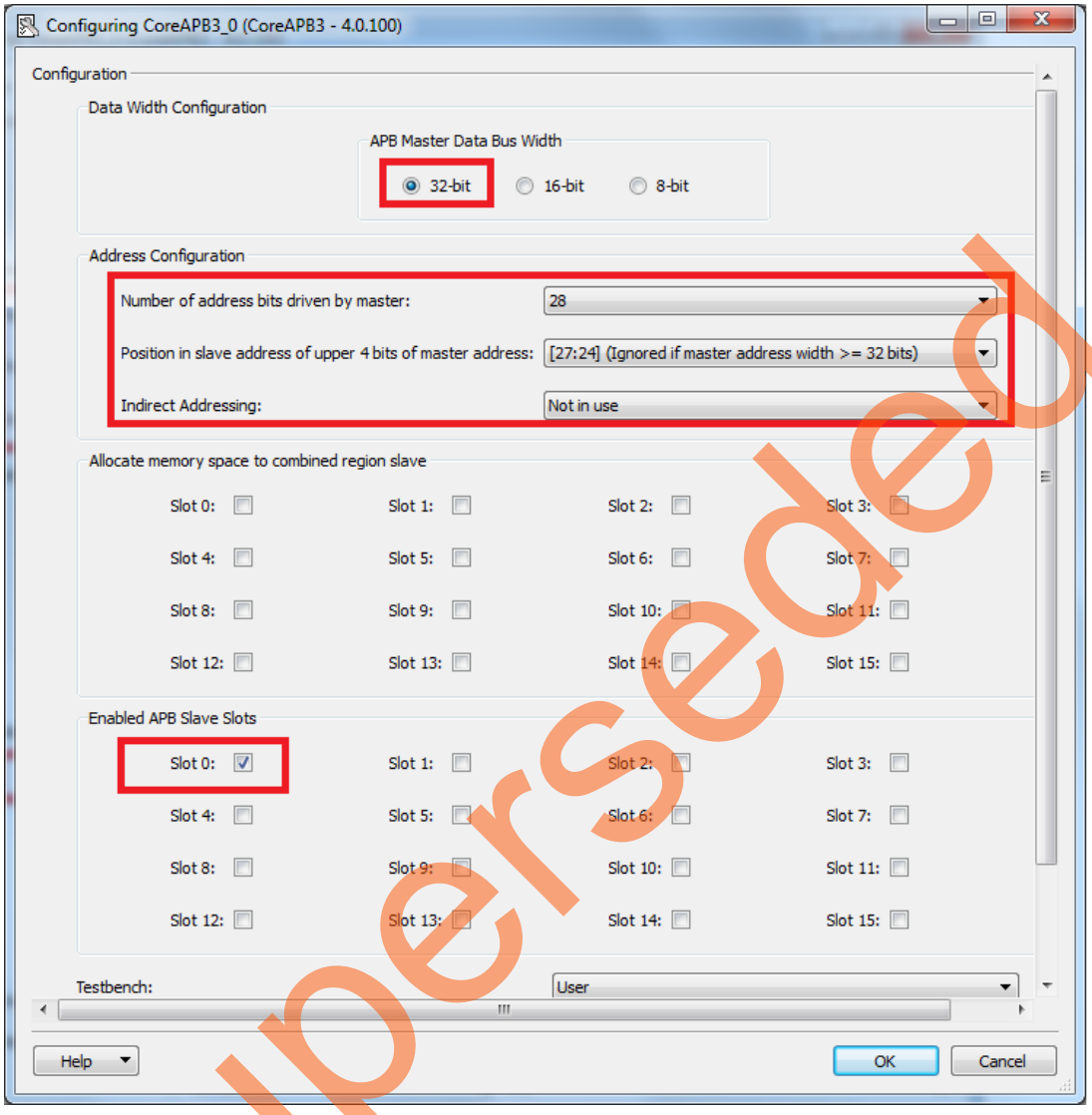


Figure 18 • **Configuring CoreAPB3_0**

6. Configure **CoreAPB_0** with the below settings:
   – Under **Data Width Configuration**, select **APB Master Data Bus Width** as **32-bit**.
   – Under **Address Configuration**, select **Number of address bits driven by master** as '**28**' and **Position in slave address of upper 4 bits of master address** as '**[27:24](Ignored if master address width >=32 bits)**' using the drop-down list.
   – Select **Enabled APB Slave Slots** as **Slot 0**.
7. Click **OK** to save and close the **Configuring CoreAPB3_0** window.

8. Figure 19 shows the PCIe_Demo_top in SmartDesign after configuring CoreAHBLite and CoreAPB3 bus interfaces.

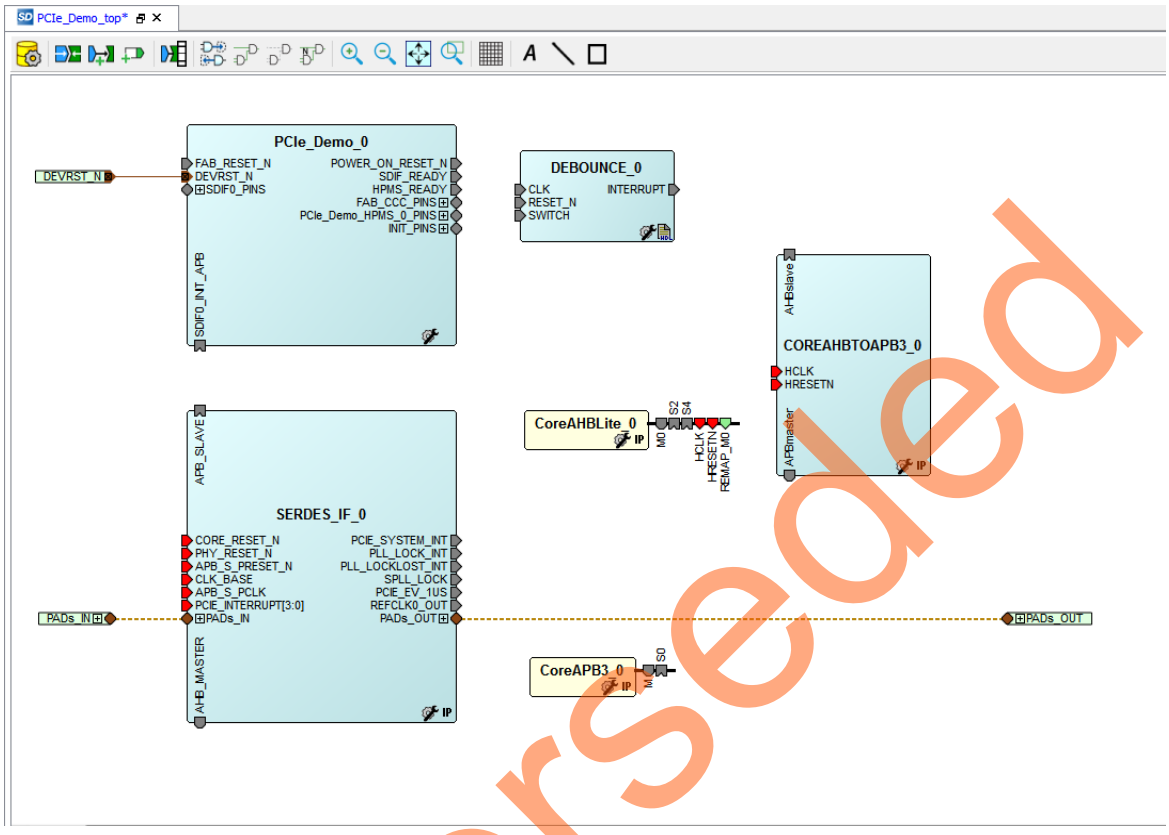


***Figure 19* • CoreAHBLite and CoreAPB3 Bus Interfaces in the PCIe_Demo_top SmartDesign Canvas After Configuration**

Instantiating CoreGPIO in PCIe_Demo_top SmartDesign

To instantiate CoreGPIO in the PCIe_Demo_top SmartDesign, expand the **Peripherals** category in the Libero SoC **Catalog** as displayed in Figure 20.

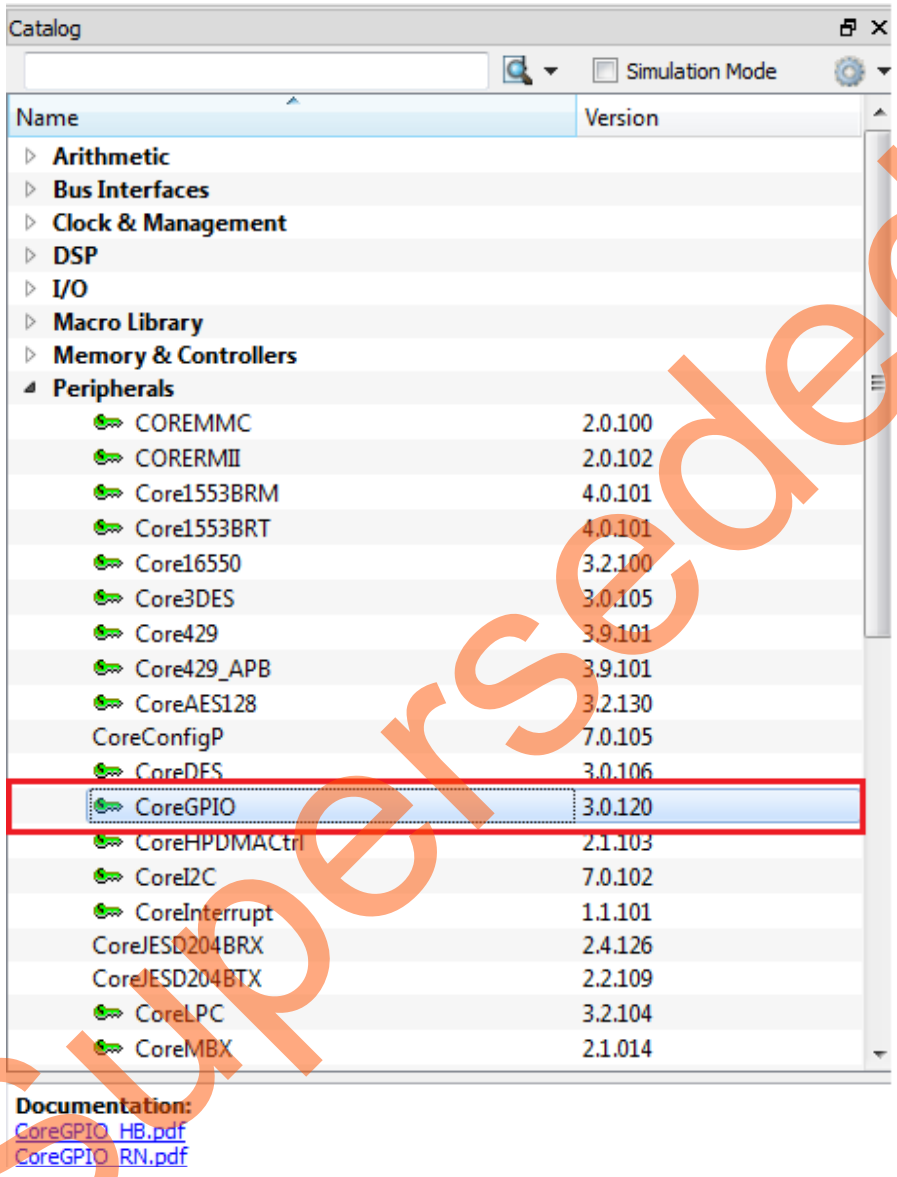


Figure 20 • **IP Catalog**

1. Drag **CoreGPIO** into the PCIe_Demo_top SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.

2. Double-click **CoreGPIO** to configure it. Figure 21 shows **Configuring CoreGPIO_0** window.

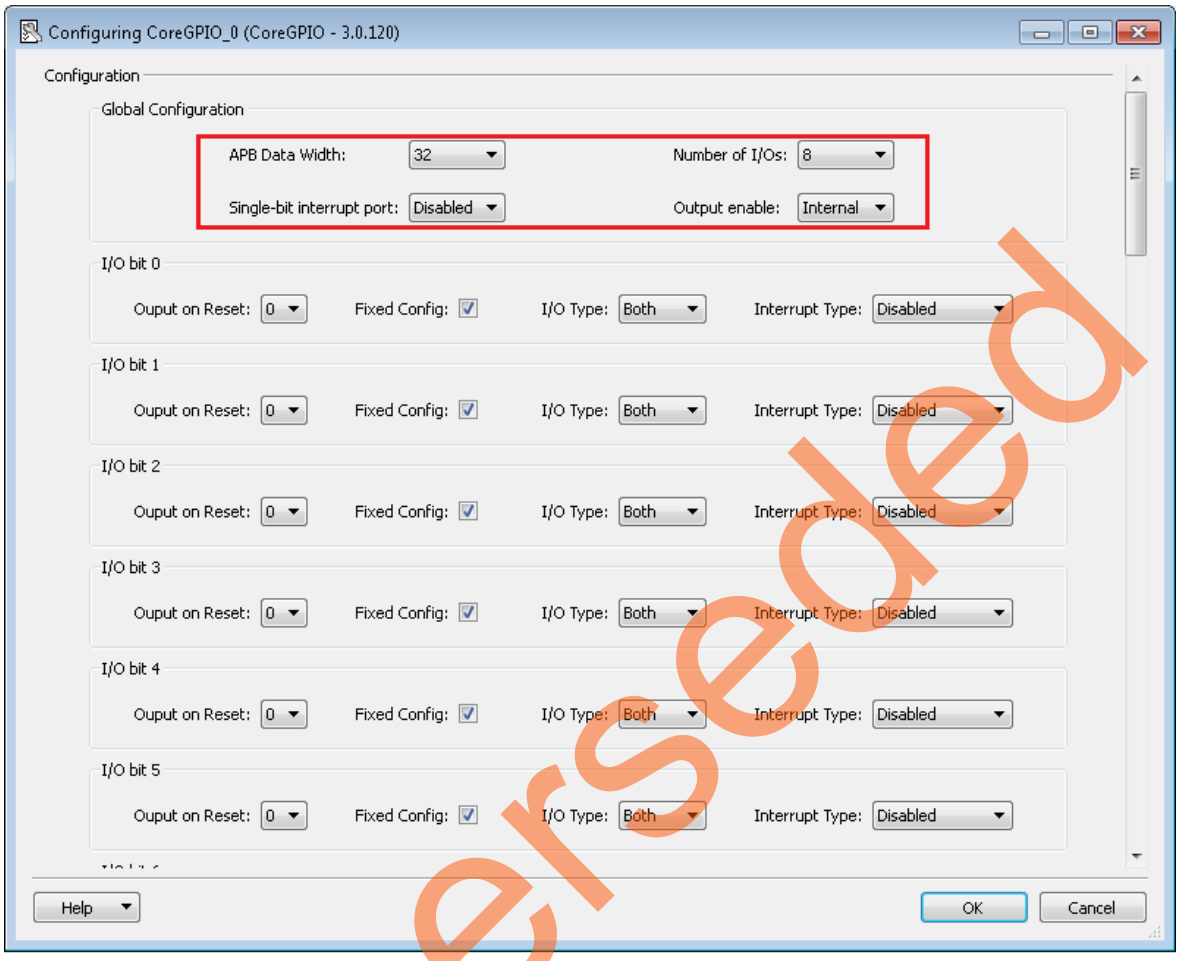


Figure 21 • **Configuring CoreGPIO_0**

3. Under **Global Configuration**, configure the below settings:
   – Select **APB Data Width** as **32**.
   – Select **Number of I/O's** as 8.
   – Select **Output enable** as **Internal**.For all I/O bits from 0 to 7, configure **I/O Type** as **Both**.
4. Click **OK** to save and close the **Configuring CoreGPIO_0** window.

CoreGPIO is configured with 8 outputs connected to LED's and with four inputs connected to DIP switches.

Instantiating CoreAHBLSRAM in PCIe_Demo_top SmartDesign

To instantiate CoreAHBLSRAM in the PCIe_Demo_top SmartDesign, expand the **Memory & Controllers** category in the Libero SoC **Catalog** as displayed in Figure 22

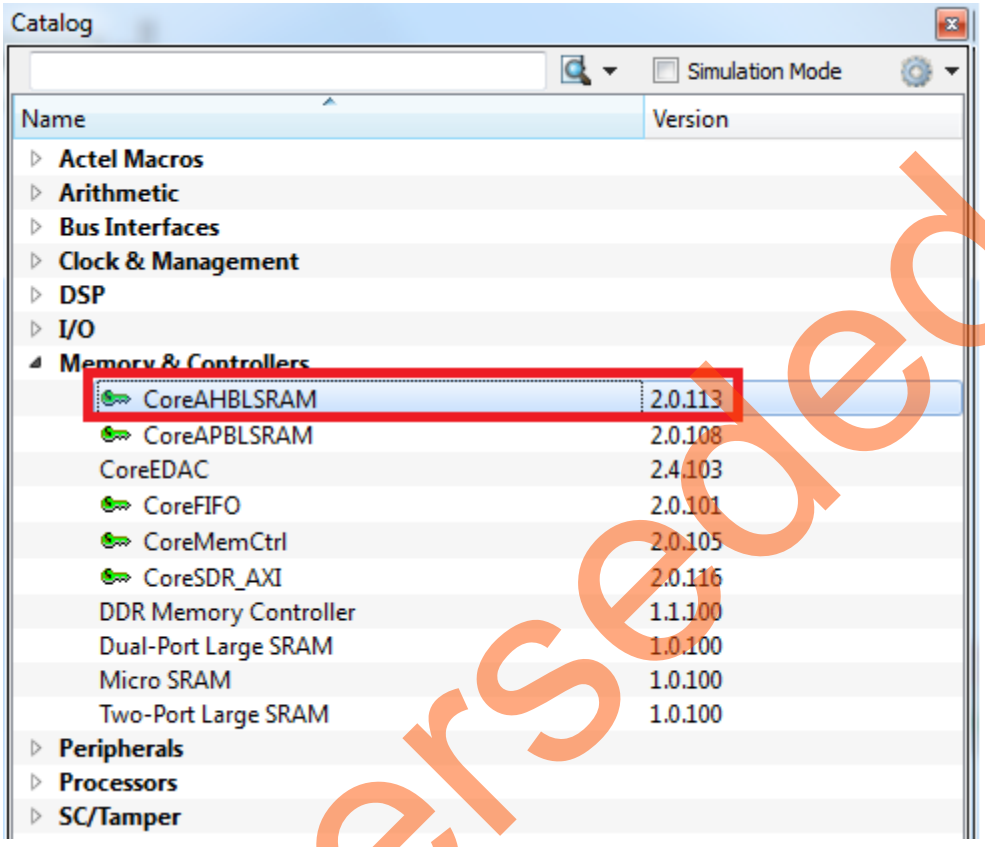


Figure 22 • **IP Catalog**

1. Drag **CoreAHBLSRAM** into the **PCIe_Demo_top** SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.

2. Double-click COREAHBLSRAM_0 to configure it. Figure 23 shows **Configuring COREAHBLSRAM_0** window.

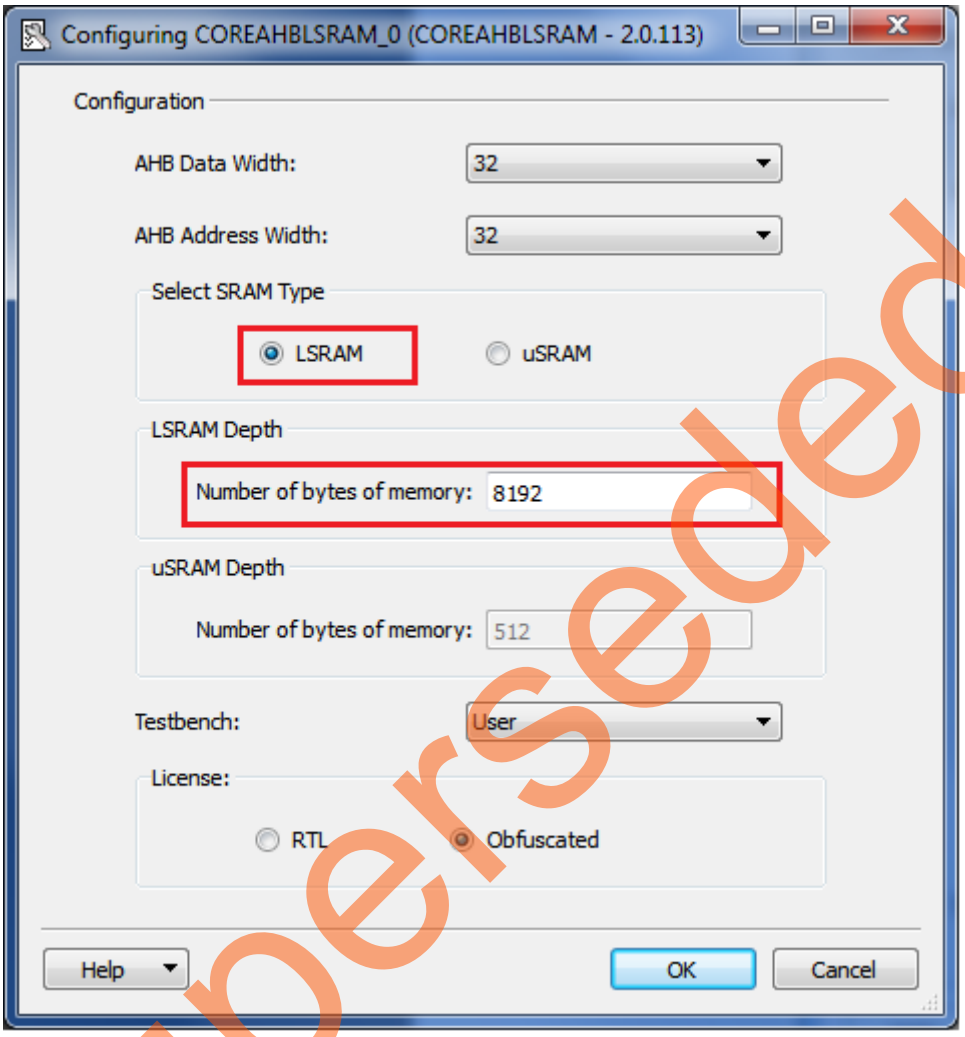


***Figure 23* • Configuring COREAHBLSRAM_0**

– Under **Configuration**, select **AHB Data Width** and **AHB Address Width** as **32**.
– Under **Select SRAM Type,** click **LSRAM**.
– Under **LSRAM Depth**, enter the **Number of bytes of memory** as **8192**.

3. Click **OK** to save and close the **Configuring COREAHBLSRAM_0** window.

Instantiating Clock Conditioning Circuitry (CCC) in PCIe_Demo_top SmartDesign

CCC supplies the clock for components instantiated in the Fabric. To instantiate CCC in the PCIe_Demo_top SmartDesign, expand the **Clock & Management** category in the Libero SoC **Catalog**. Figure 24 shows Libero Catalog.

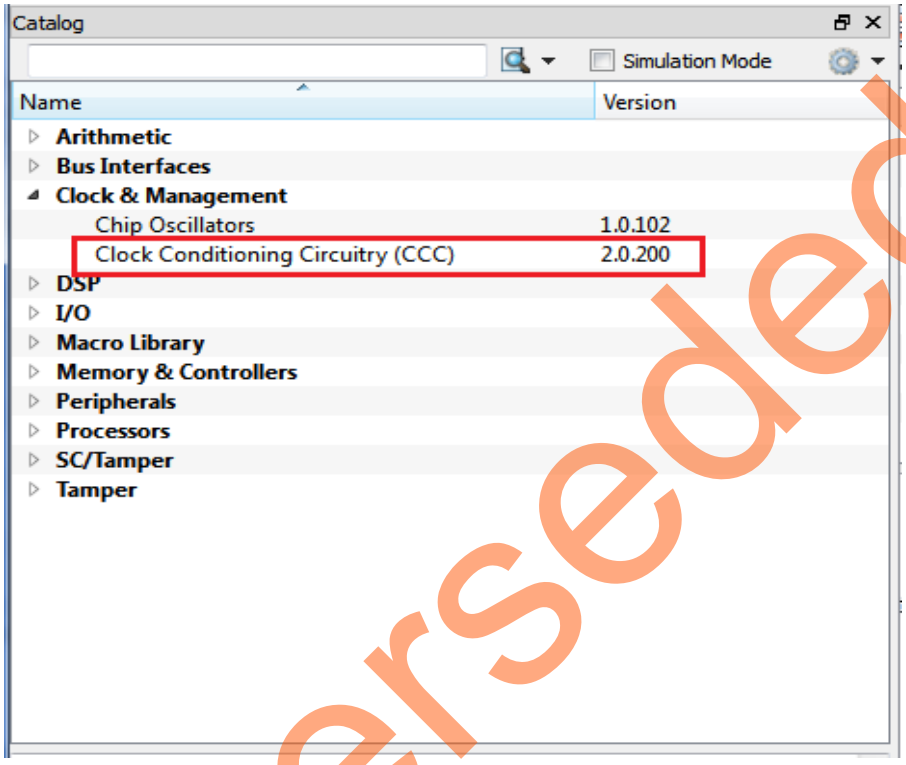


Figure 24 • **IP Catalog**

1. Drag CCC into the PCIe_Demo_top SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.

2. Double-click **CCC** to configure it. Figure 25 shows the **FAB CCC Configurator** window.

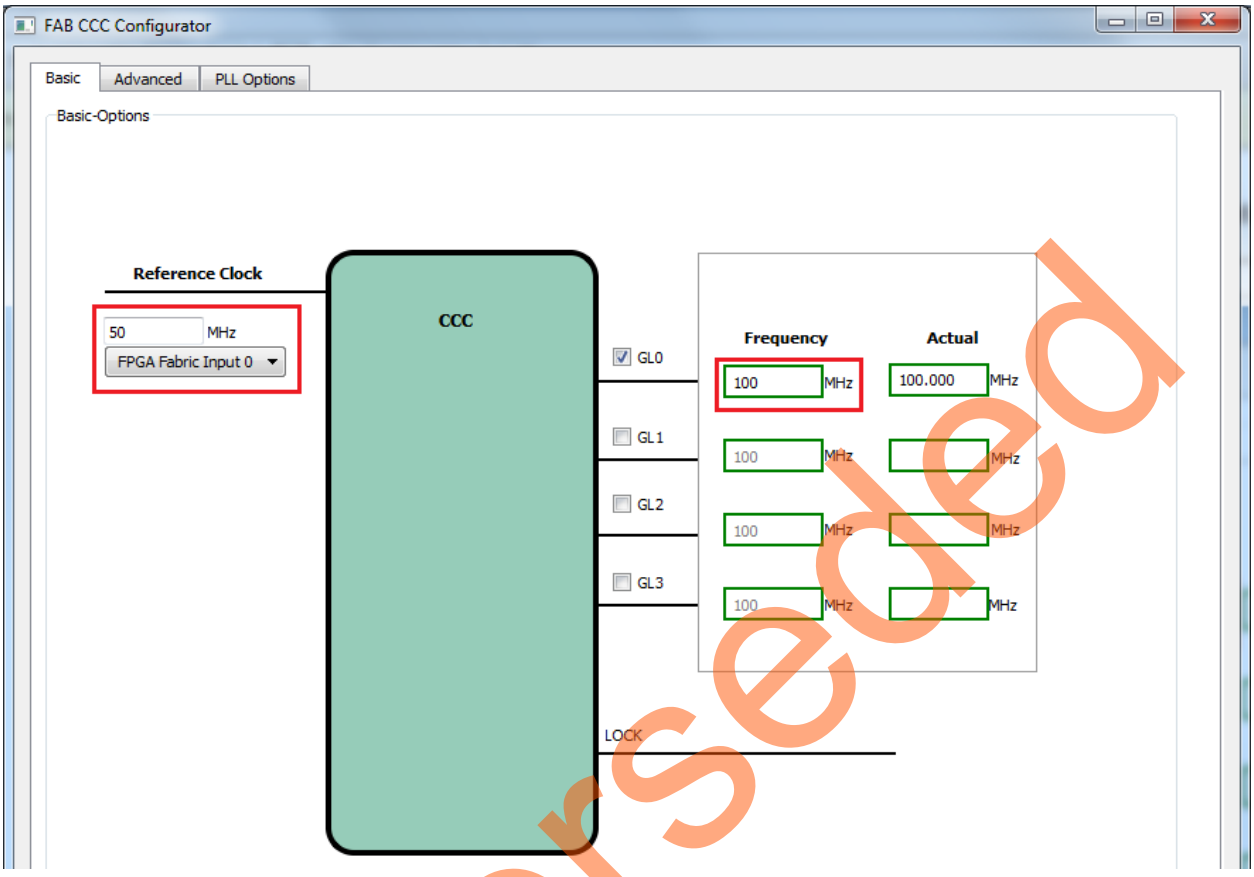


***Figure 25* • Configure CCC**

– Select **Reference Clock** as **50 MHz** and F**PGA Fabric Input 0** from the drop-down list.
– Select **GL0 Frequency** as **100 MHz**.

3. Click **OK** to save and close the **FAB CCC Configurator** window.

4. Figure 26 shows PCIe_Demo_top in SmartDesign after configuring all components.

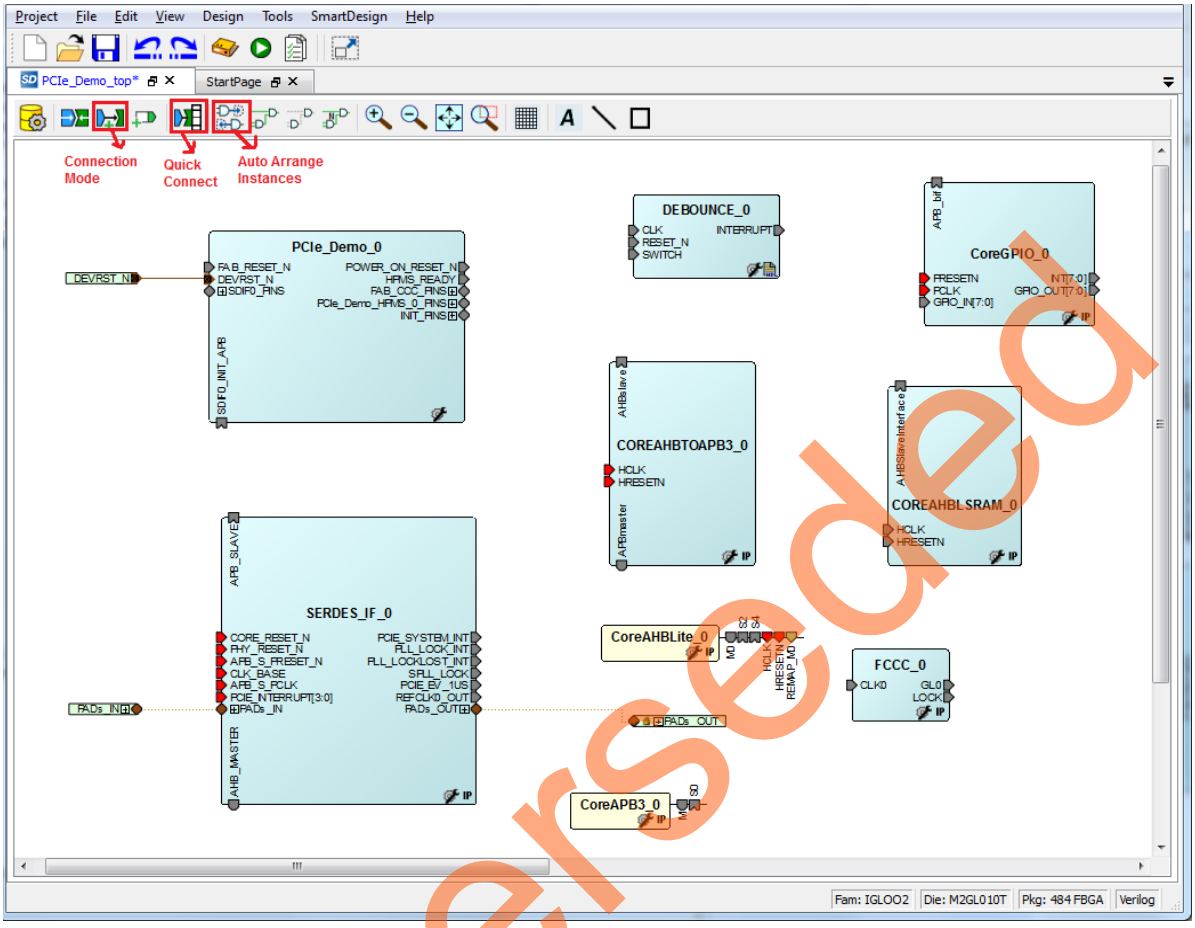


Figure 26 • **PCIe_Demo_top in SmartDesign**

Connecting Components in PCIe_Demo_top SmartDesign

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 26. 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 26. 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 27.

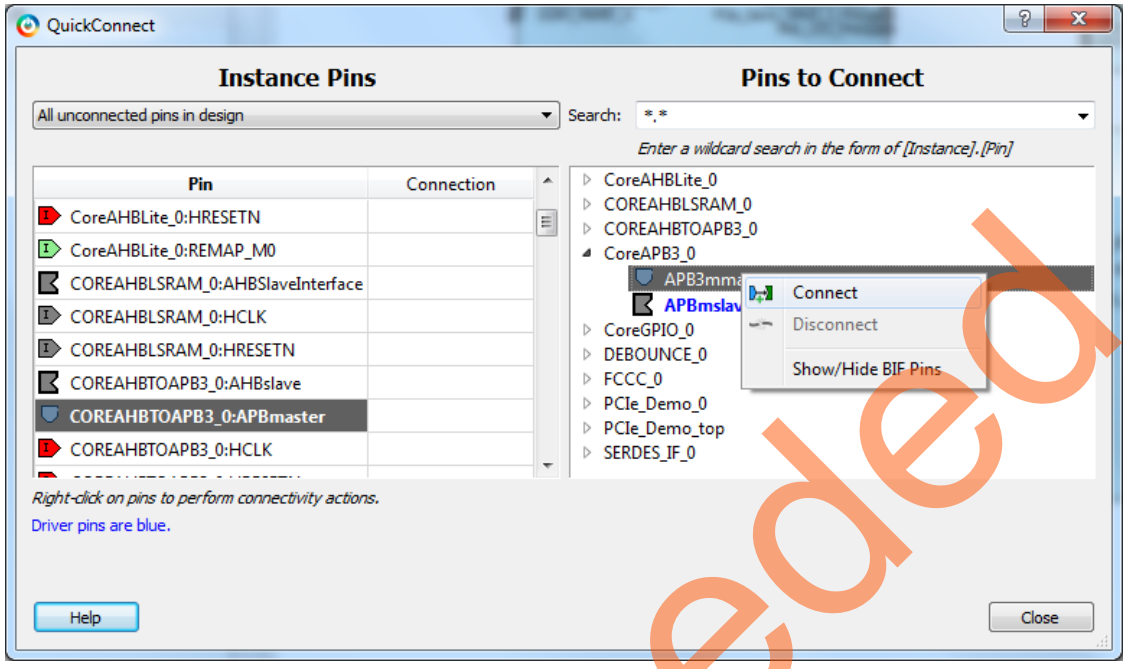


Figure 27 • **Quick Connect Window**

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

1. Expand ***SDIF0_PINS*** of PCIe_Demo_0 and make connections as shown in Table 2.

Table 2 • **SDIF0_PINS**

From PCIe_Demo_0	To SERDES_IF_0
SDIF0_PHY_RESET_N	PHY_RESET_N
SDIF0_CORE_RESET_N	CORE_RESET_N
SDIF0_SPLL_LOCK	SPLL_LOCK

2. Right-click the **SDIF0_PERST_N** and **promote to top level**.
3. Expand **INIT_PINS** of PCIe_Demo_0 and make connections as shown in Table 3.

Table 3 • **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

4. Right-click the **INIT_DONE** and select **mark unused**.
5. Connect **HPMS_READY** of PCIe_Demo_0 to all resets as shown in Table 4.

Table 4 • **HPMS_READY Connections**

From PCIe_Demo_0	To
HPMS_READY	HRESETN of CoreAHBLite_0,COREAHBTOAPB3_0, and COREAHBLSRAM_0
	PRESETN of CoreGPIO_0
	RESET_N of DEBOUNCE_0

6. Connect **GL0** of FCCC_0 to all clocks as shown in Table 5.

Table 5 • **GL0 Clock Connections**

From FCCC_0	To
GL0	HCLK of CoreAHBLite_0,COREAHBTOAPB3_0, and COREAHBLSRAM_0
	PCLK of CoreGPIO_0
	CLK of DEBOUNCE_0
	CLK_BASE of SERDES_IF_0

7. Expand FAB_CCC_PINS of PCIe_Demo_0:
   – Right-click the **FAB_CCC_GL0** and select **Mark Unused**.
   – Right-click the **FAB_CCC_GL3** and select **Mark Unused**.
8. Right-click the **POWER_ON_RESET_N** of PCIe_Demo_0 and select **Mark Unused**.
9. Right-click the **SDIF_READY of PCIe_Demo_0** and select **Mark Unused**.
10. Right-click the **FAB_RESET_N** of **PCIe_Demo_0** and select **Tie high**.
11. Expand **PCI_Demo_HPMS_0_PINS**.
    – Right-click the **COMM_BLK_INT** of PCIe_Demo_0 and select **Mark Unused**.
    – Right-click the **HPMS_INT_M2F[15:0]** of PCIe_Demo_0 and select **Mark Unused**.
12. Connect **SDIF0_INIT_APB** of **PCIe_Demo_0** and **APB_SLAVE** of **SERDES_IF_0**.
13. Connect Master port **M0 of** CoreAHBLite_0 to Master port **AHB_MASTER of SERDES_IF_0**
14. Connect Slave port **S2 of** CoreAHBLite_0 to Slave port **AHBSlaveInterface of COREAHBLSRAM_0**
15. Connect Slave port **S4 of** CoreAHBLite_0 to Slave port **AHBslave of COREAHBTOAPB3_0**
16. Connect Master port **M of CoreAPB3_0** to Master port **APBmaster of COREAHBTOAPB3_0**
17. Connect Slave port **S0 of CoreAPB3_0** to Slave port **APB_bif of CoreGPIO_0**
18. Right-click the **CLK0 of FCCC_0** and select **Promote to top level**.
19. Right-click the **LOCK of FCCC_0** and select **Mark unused**.
20. Right-click the **SWITCH of DEBOUNCE_0** and select **Promote to top level**.
21. Right-click the **INT[7:0] of CoreGPIO_0** and select **Mark unused**.
22. Right-click the **GPIO_OUT[7:0] of CoreGPIO_0** and select **Promote to top level**.
23. This design uses 4 GPIO inputs **GPIO_IN [3:0] of CoreGPIO_0** to connect **DIP switches**. To connect unused **GPIO_IN[7:4]** to logic ‘0’ split the GPIO_IN[7:0] into two groups.

To do that, right-click the **GPIO_IN [7:0]** and select **Edit Slice**. Figure 28 displays the **Edit Slice** window.

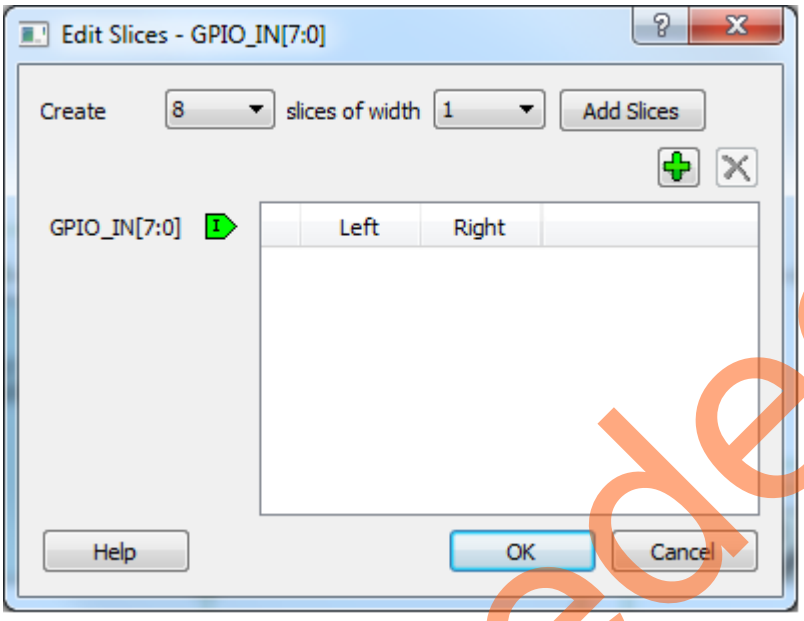


***Figure 28* • Edit Slices**

24. Select 2 slices of width 4, click **Add Slices**, and edit the window as shown in Figure 29.

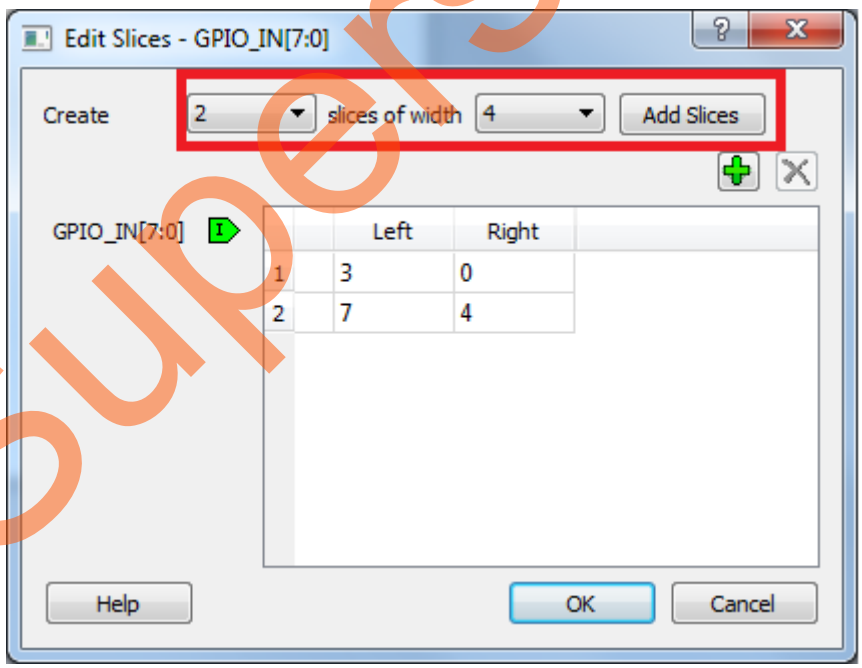


***Figure 29* • Edit Slices**

25. Click **OK**.
26. Expand **GPIO_IN [7:0]**, right-click the **GPIO_IN [7:4]** and select **Tie low**.
27. Right-click the **GPIO_IN[3:0]** and select **Promote to top level**.
28. Select the following 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
29. The PCIe supports four interrupts. This design uses only one interrupt out of four by connecting the unused interrupts to logic '0'. To connect the 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 in Figure 30.

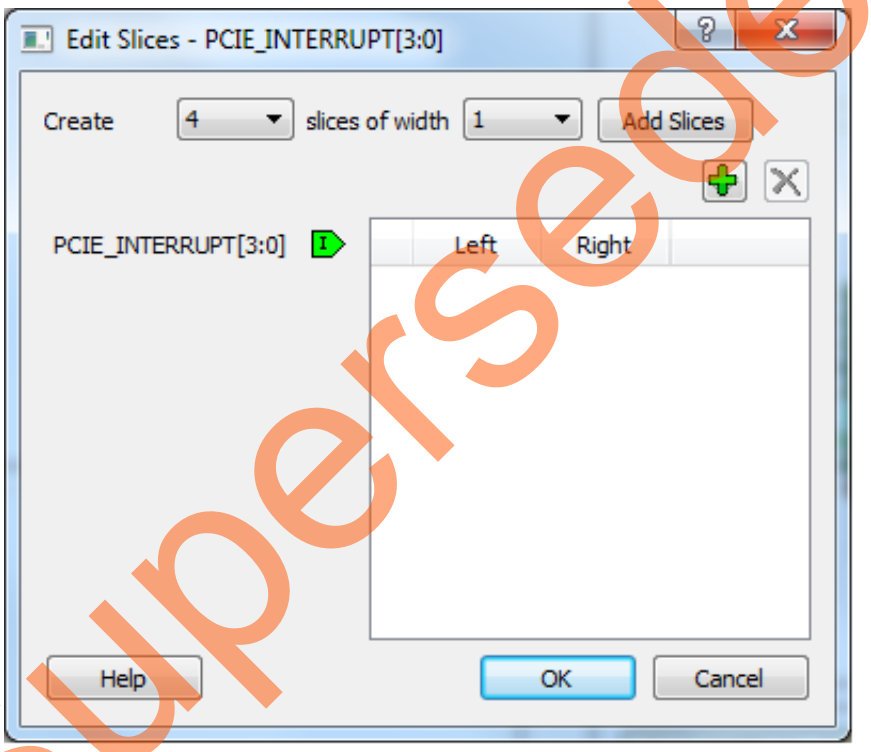


Figure 30 • **Edit Slices**

30. 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 31.

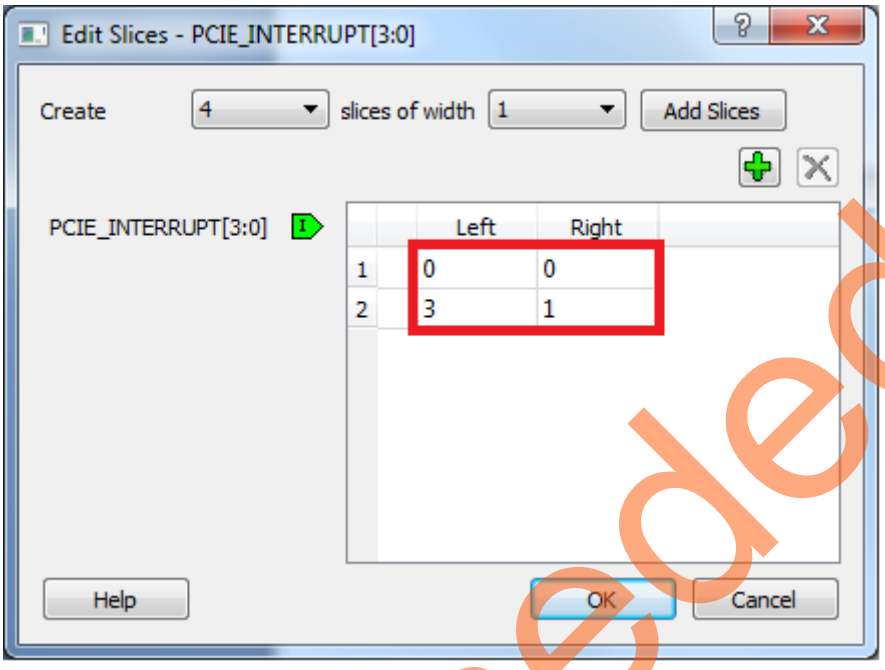


***Figure 31* • Edit Slices**

31. Expand **PCIE_INTERRUPT[3:0]**, right-click the **PCIE_INTERRUPT[3:1]**, and select **Tie low**.
32. Connect **INTERRUPT** of **DEBOUNCE_0** to the **PCIE_INTERRUPT[0]** of **SERDES_IF_0**.

33. Click **Auto arrange instances** to arrange the instances and click **File > Save**. The PCIe_Demo_top is displayed as shown in Figure 32.

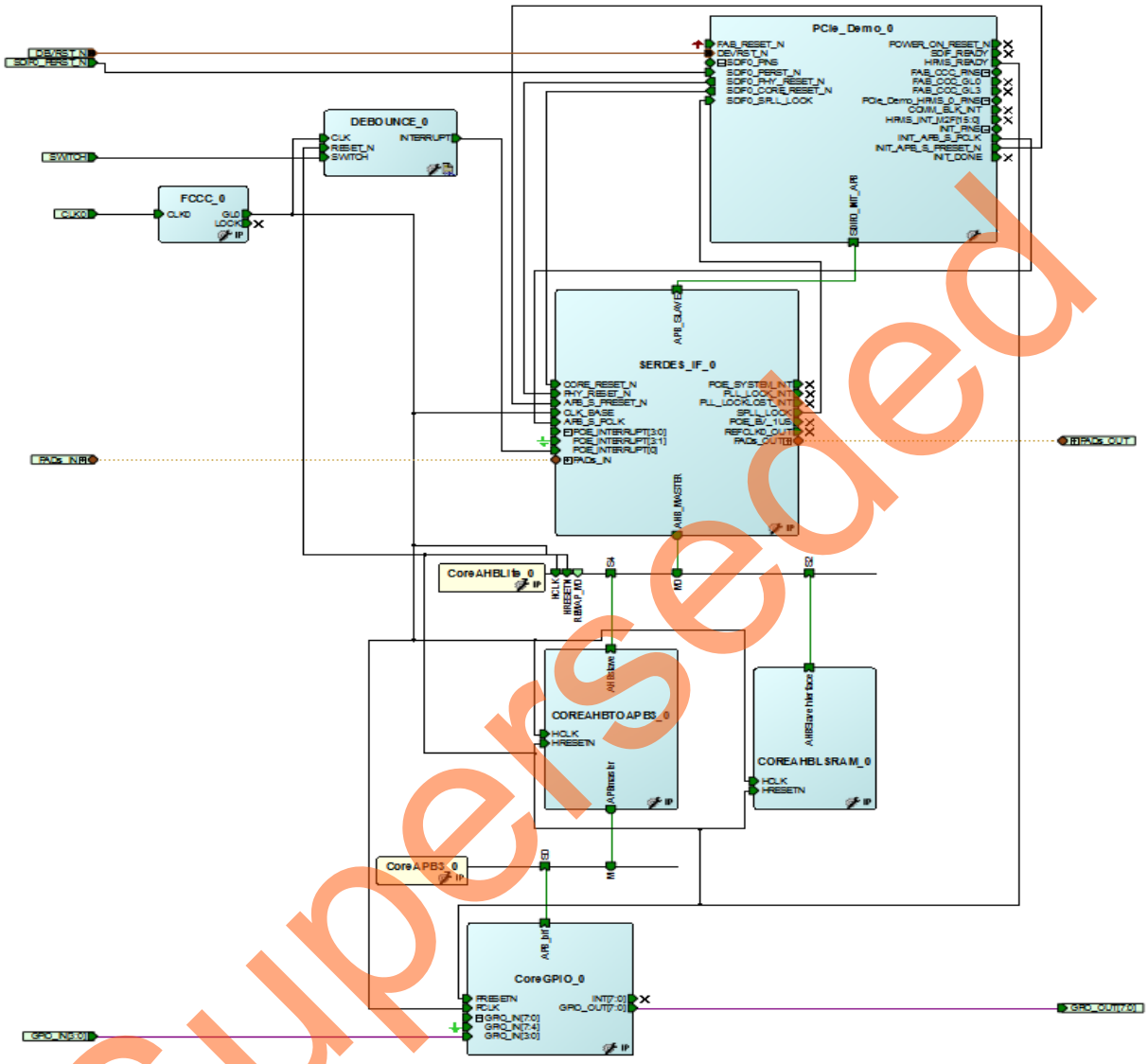


***Figure 32* • PCIe_Demo_top Design**

34. Open the **PCIe_Demo_top** tab and click **Generate Component** as shown in Figure 33.

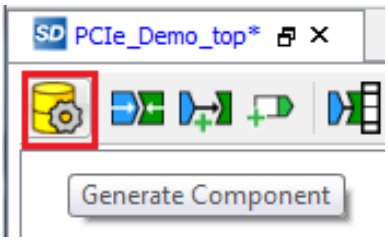


***Figure 33* • Generate Component**

35. The message "'PCIe_Demo_top' was generated" is displayed in the Libero SoC Log window if the design was generated without any errors. The Log window is displayed as shown in Figure 34 on successful component generation.

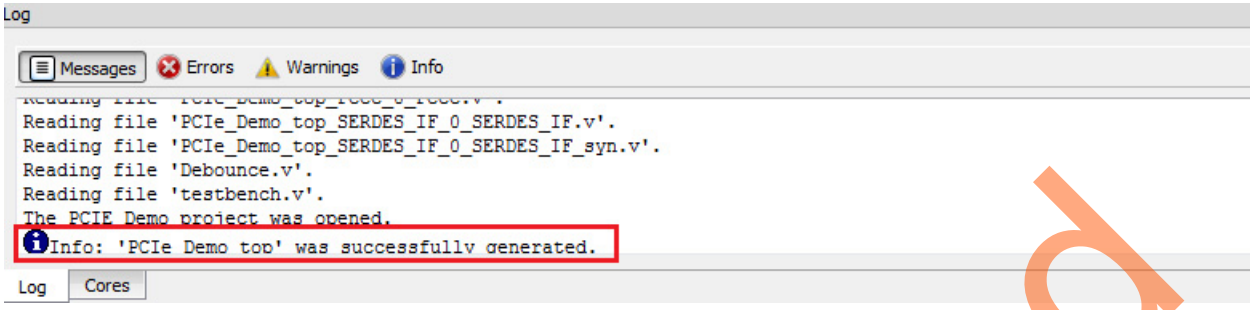


Figure 34 • **Log Window**

36. The Core usage table is displayed in the Libero SoC **Cores** window. Figure 35 shows the **Cores** window that is displayed on successful component generation.

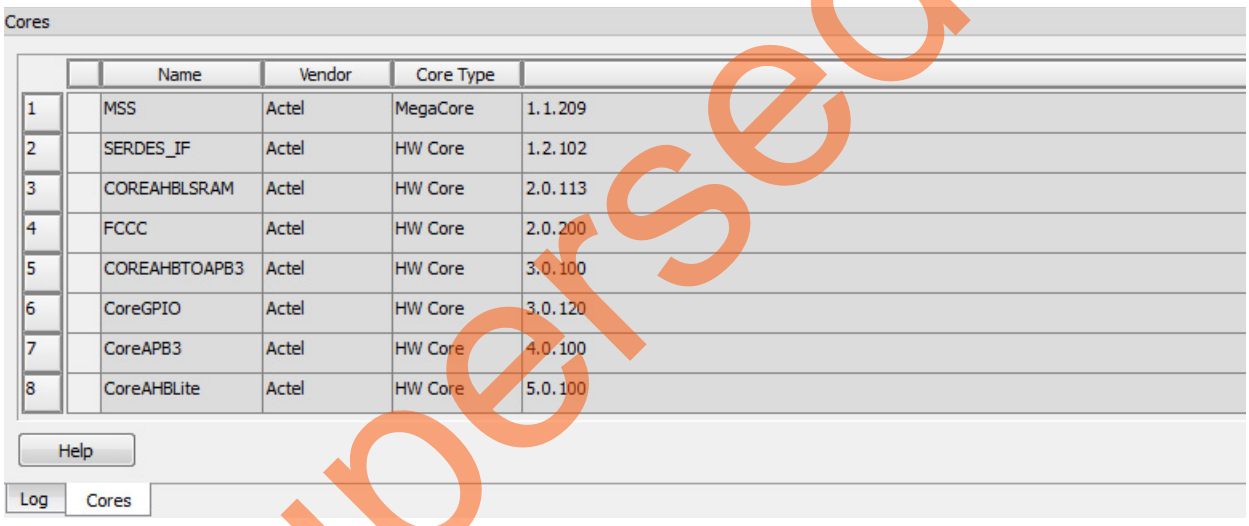


Figure 35 • **Cores Window**

Step 2: 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 AHBLite 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 IGLOO2 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 Fabric CoreAHBLSRAM and CoreGPIO.

1. To open the SERDESIF_0_user.bfm, go to the **Files** tab > **Simulation** folder, and double-click the SERDESIF_0_user.bfm. The SERDESIF_0_user.bfm file is displayed as shown in Figure 36.

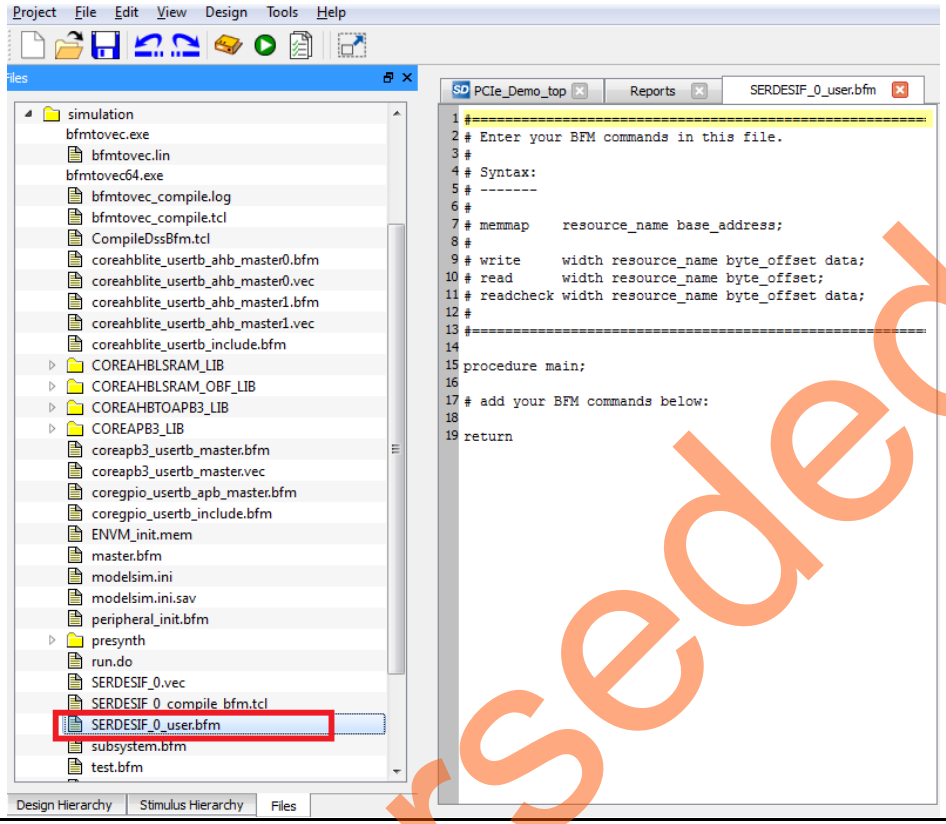


***Figure 36* • SmartDesign Generated SERDESIF_0_user.bfm File**

2. Modify the SERDESIF_0_user.bfm to add the following bfm commands of writing and reading:

```
memmap GPIO 0x40000000;
memmap LSRAM 0x20000000;
procedure main;
# add your BFM commands below:
wait 500us;
wait 500us;
write w GPIO 0xA0 0x00;
write w GPIO 0xA0 0x01;
write w GPIO 0xA0 0x02;
write w GPIO 0xA0 0x04;
write w GPIO 0xA0 0x08;
write w GPIO 0xA0 0x10;
write w GPIO 0xA0 0x20;
write w GPIO 0xA0 0x40;
write w GPIO 0xA0 0x80;

write w LSRAM  0x00  0x12345678;
write w LSRAM  0x04  0x87654321;
write w LSRAM  0x08  0x9ABCDEF0;
write w LSRAM  0x0C  0x0FEDCBA9;
readcheck w LSRAM 0x00 0x12345678;
readcheck w LSRAM 0x04  0x87654321;
readcheck w LSRAM 0x08 0x9ABCDEF0;
readcheck w LSRAM 0x0C  0x0FEDCBA9;
return
```

BFM commands added in the SERDESIF_0_user.bfm do the following:

– Perform write to GPIO_OUT[7:0]
– Perform write to LSRAM
– Perform read-check from LSRAM

3. The modified BFM file appears similar to the file shown in Figure 37.

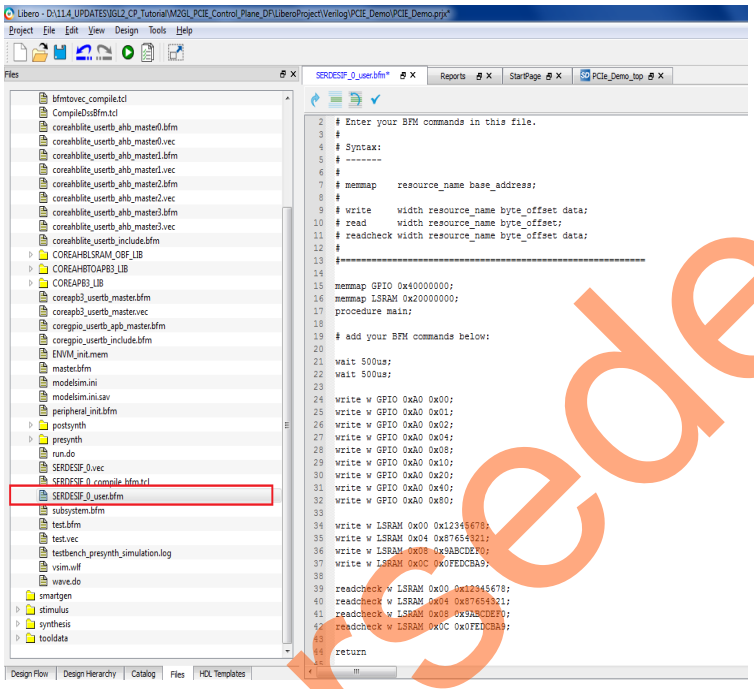


***Figure 37* • Modified SERDES User BFM**

Step 3: Simulating the Design

The design supports the BFM_PCIe 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_0_user.bfm file under the *<Libero project>/simulation* folder contains the BFM commands to verify the read/write access to CoreGPIO and CoreAHBLSRAM. This section describes how to use the SmartDesign testbench and the 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: *M2GL_PCIE_Control_Plane_DF\Source Files*. Figure 38 shows the wave.do file under simulation folder in the Files window.

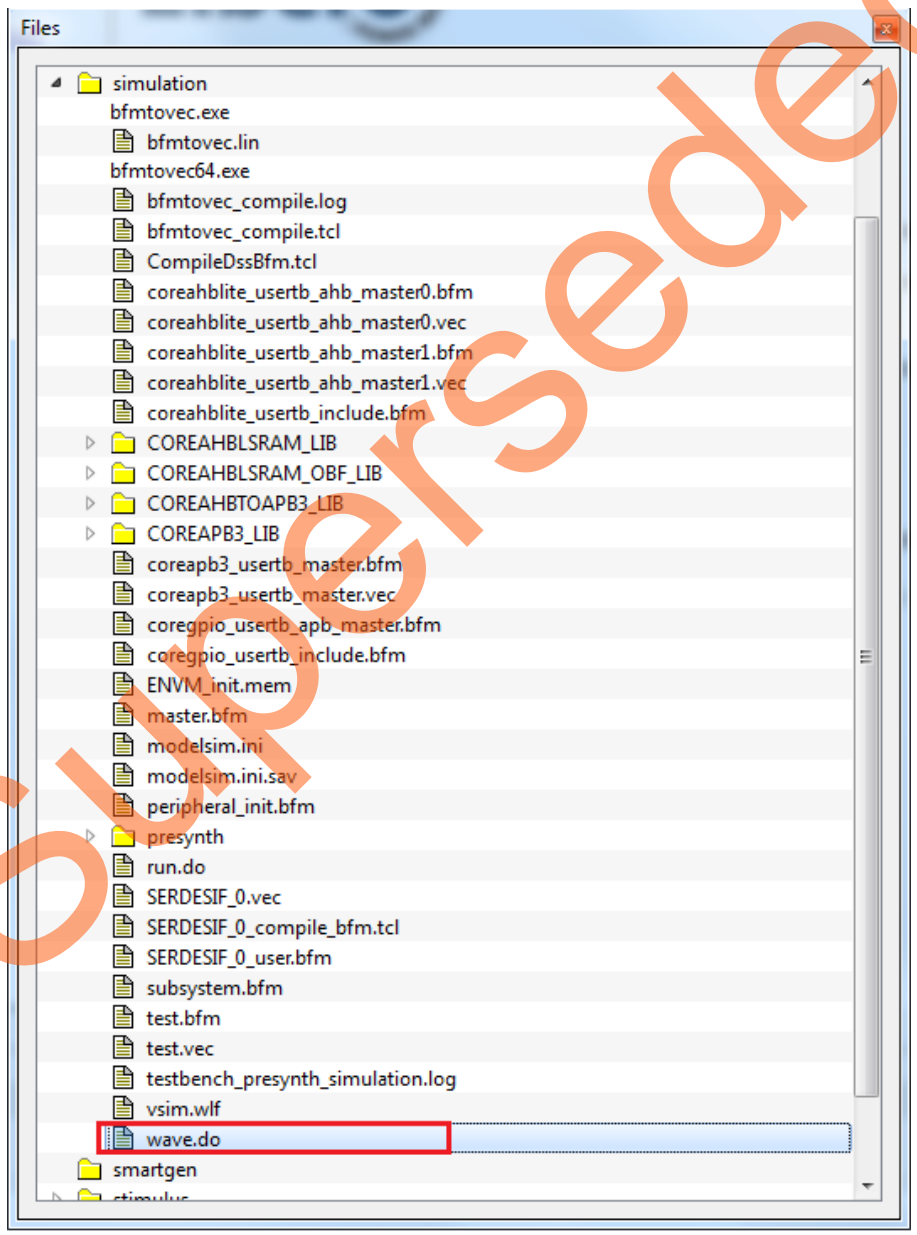


Figure 38 • **Wave.do file**

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 **280us**, as shown in Figure 39.

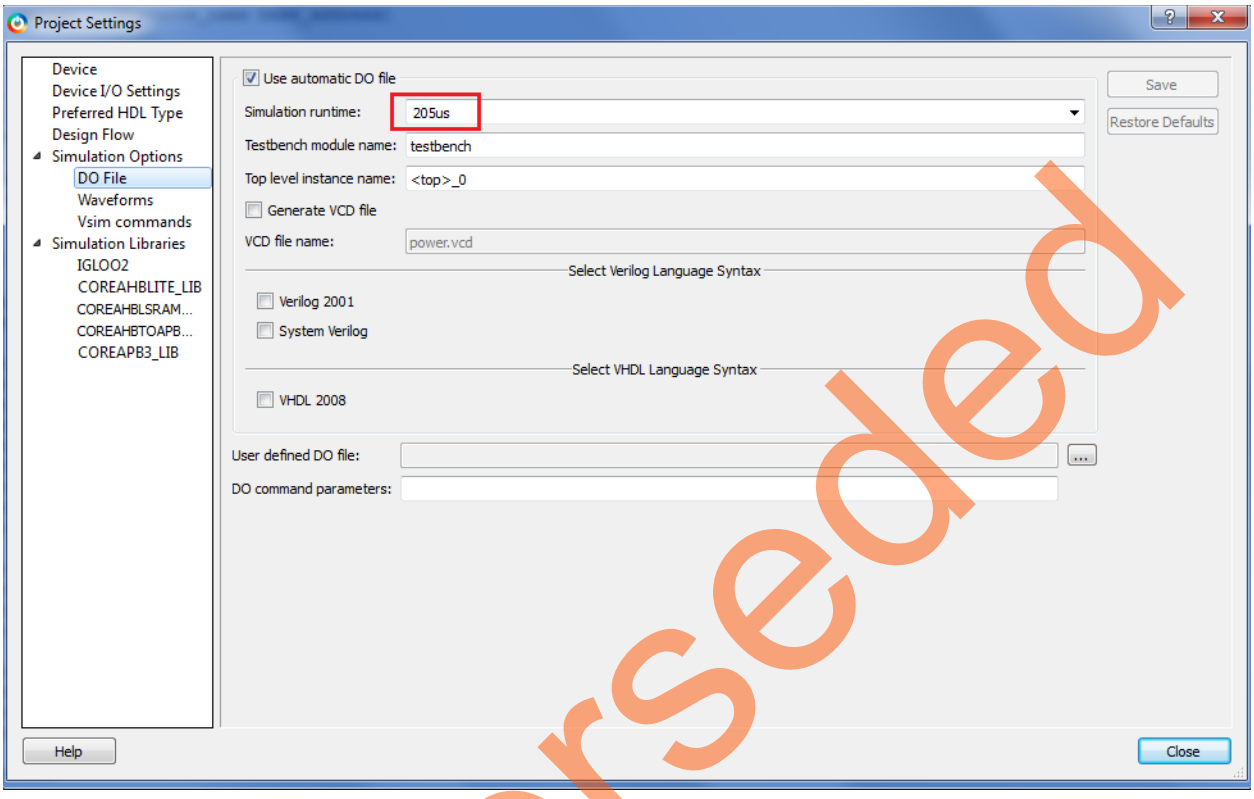


***Figure 39* • Project Setting – Do File Simulation Runtime Setting**

5. Click **Save**.

6. Select **Waveforms** under **Simulation Options** as shown in Figure 40.

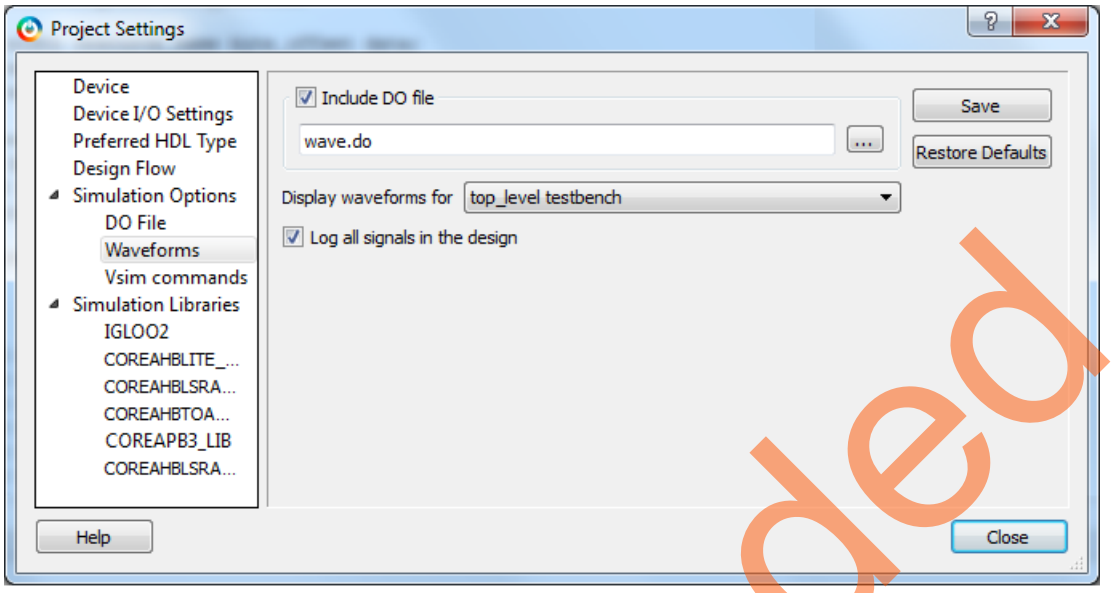


***Figure 40* • Project Setting – Waveform**

– Select the **Include Do** check box and select the file.
– Select the **Log all signals in the design** check box.
– Click **Close** to close the **Project Settings** dialog box.
– Click **Save** when prompted to save the changes.

Generating Testbench

1. Go to **File** > **New** > **HDL Test bench**.

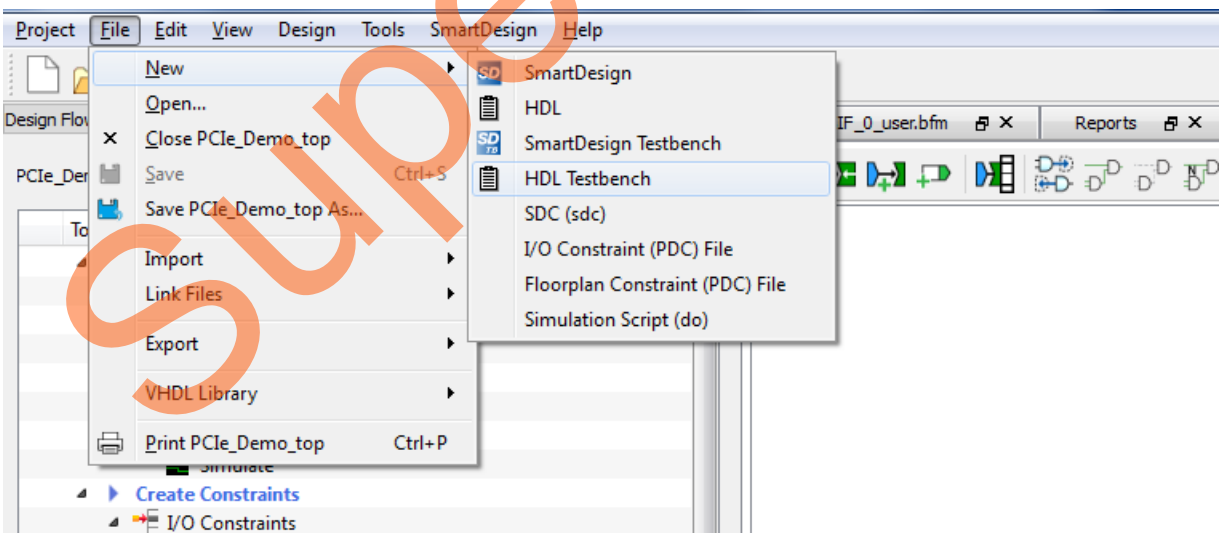


***Figure 41* • HDL Testbench**

2. Select HDL type as Verilog/VHDL and Give testbench as the Name.

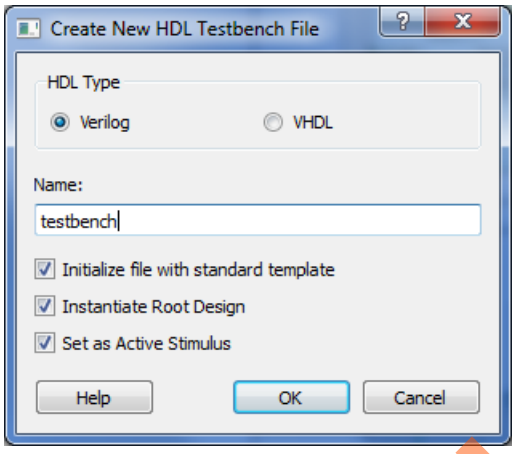


***Figure 42* • Selecting HDL Type as Verilog/VHDL in HDL Testbench**

3. Click **OK**.

To run the simulation, double-click **Simulate** under **Verify Pre-Synthesized Design** in the **Design Flow** window. ModelSim runs the design for about **205us**. The ModelSim transcript window displays the BFM commands and the BFM simulation completed with no errors, as shown in Figure 43.

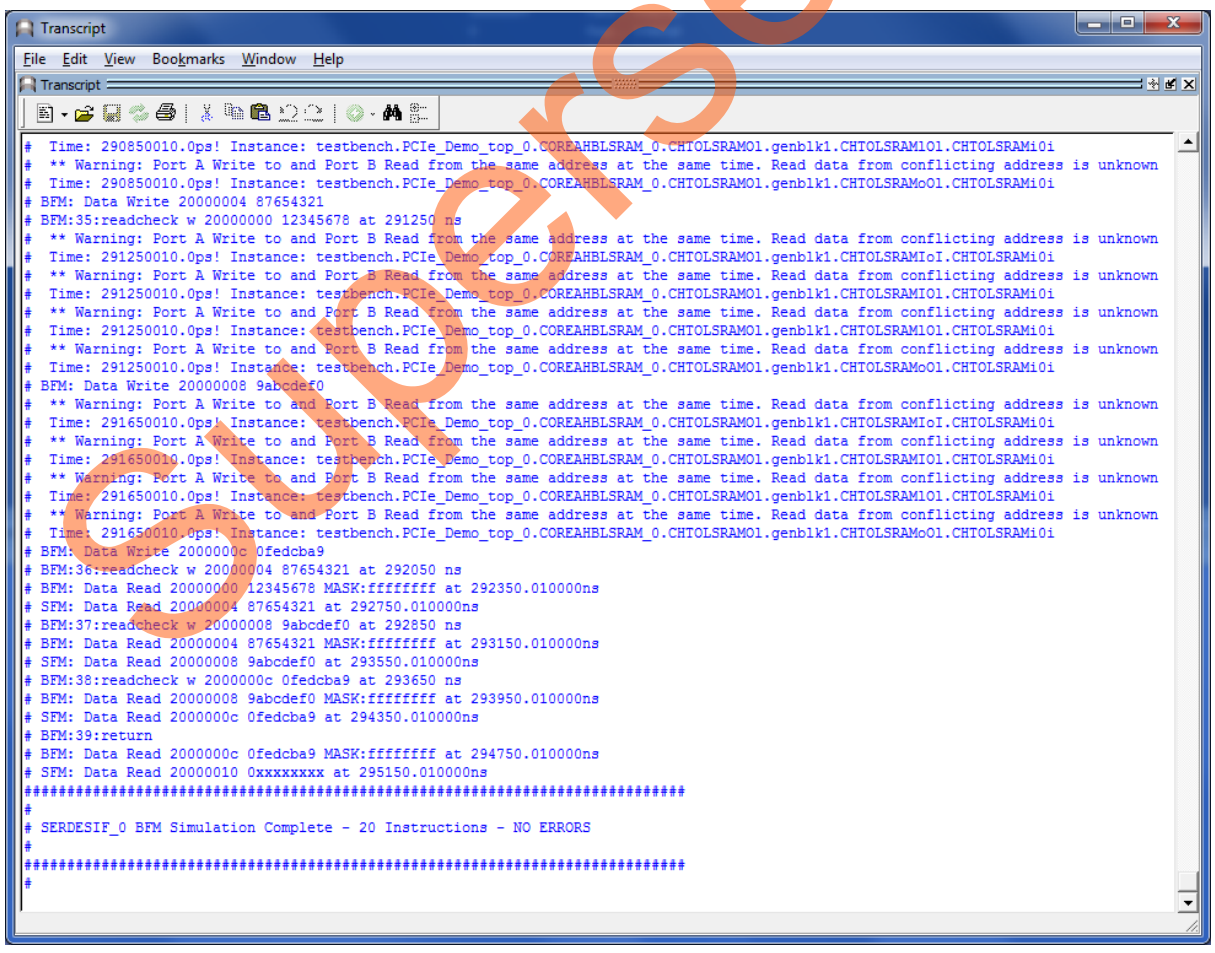


***Figure 43* • SERDES BFM Simulation**

Figure 44 shows the waveform window with GPIO_OUT signals.

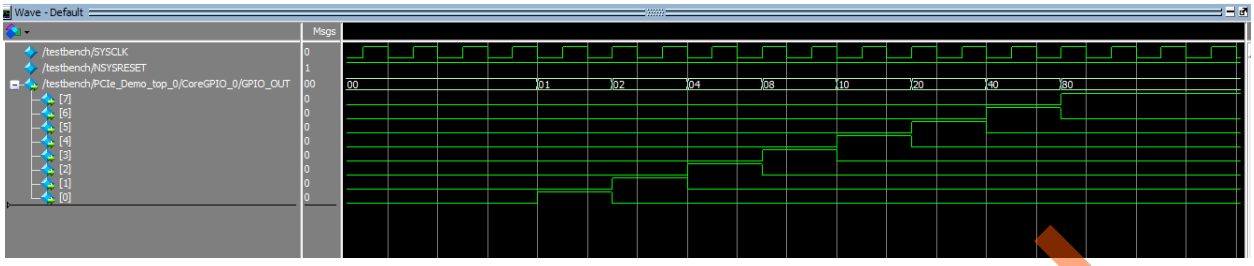


Figure 44 • **Simulation Result with GPIO_OUT Signals**

Step 4: Generating the Program File

1. Double-click **I/O Constraints** in the **Design Flow** window as shown in Figure 45. The **I/O Editor** window is displayed after completing **Synthesize and Compile**.

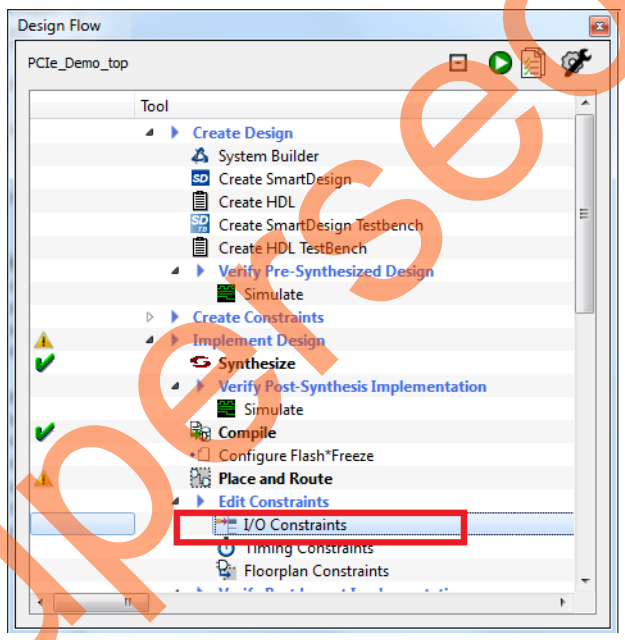


Figure 45 • **I/O Constraints**

2. In the **I/O Editor** window, make the pin assignments as shown in Table 6.

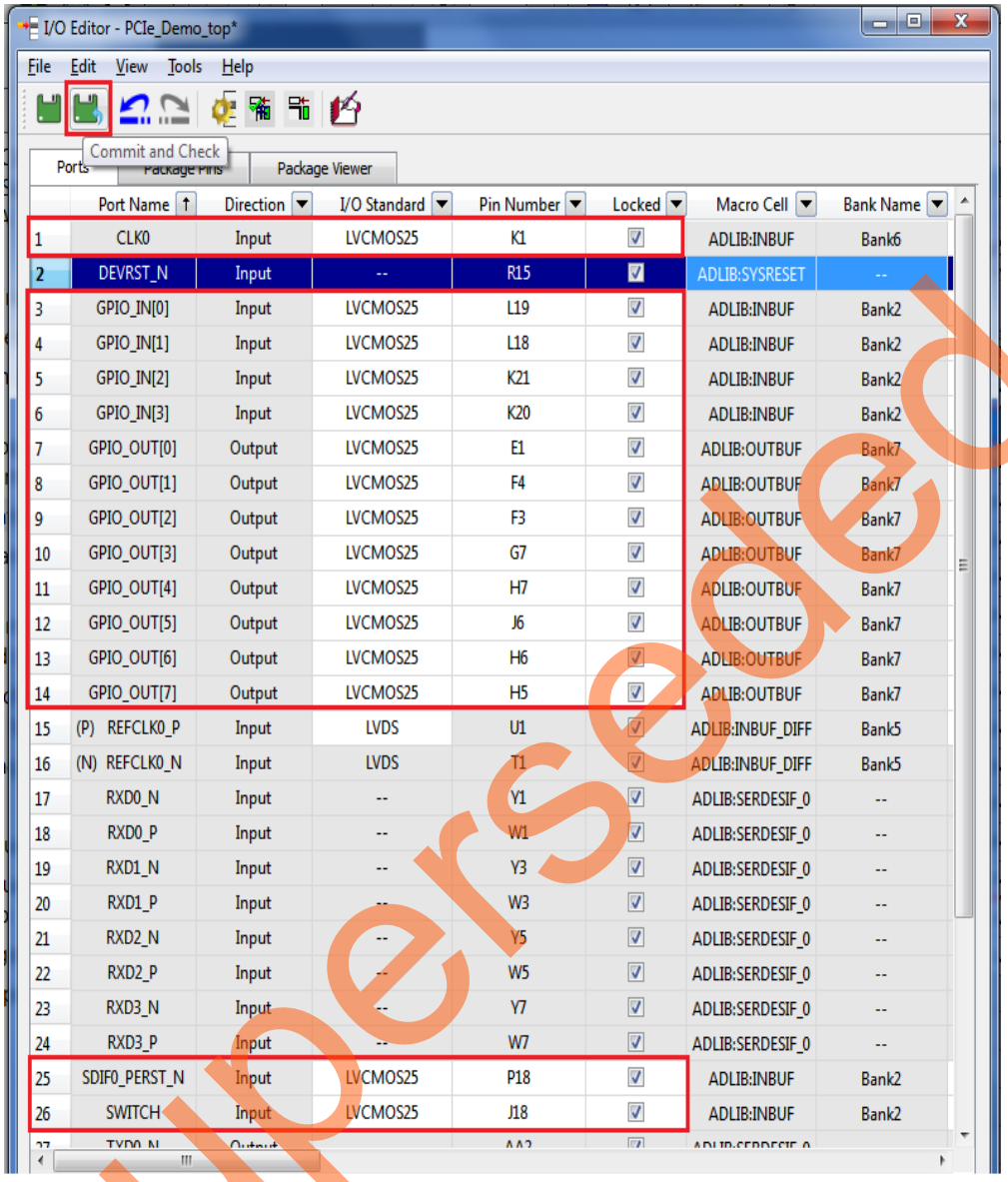
Table 6 • **Port to Pin Mapping**

Port Name	Pin Number
CLK0	K1
GPIO_IN[0]	L19
GPIO_IN[1]	L18
GPIO_IN[2]	K21
GPIO_IN[3]	K20
GPIO_OUT[0]	E1

Table 6 • **Port to Pin Mapping (continued)**

Port Name	Pin Number
GPIO_OUT[1]	F4
GPIO_OUT[2]	F3
GPIO_OUT[3]	G7
GPIO_OUT[4]	H7
GPIO_OUT[5]	J6
GPIO_OUT[6]	H6
GPIO_OUT[7]	H5
SDIF0_PERST_N	P18
SWITCH	J18

After the pins have been assigned, the I/O Editor is displayed as shown in Figure 46.



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name
1	CLK0	Input	LVCMOS25	K1	☑	ADLIB:INBUF	Bank6
2	DEVRST_N	Input	--	R15	☑	ADLIB:SYSRESET	--
3	GPIO_IN[0]	Input	LVCMOS25	L19	☑	ADLIB:INBUF	Bank2
4	GPIO_IN[1]	Input	LVCMOS25	L18	☑	ADLIB:INBUF	Bank2
5	GPIO_IN[2]	Input	LVCMOS25	K21	☑	ADLIB:INBUF	Bank2
6	GPIO_IN[3]	Input	LVCMOS25	K20	☑	ADLIB:INBUF	Bank2
7	GPIO_OUT[0]	Output	LVCMOS25	E1	☑	ADLIB:OUTBUF	Bank7
8	GPIO_OUT[1]	Output	LVCMOS25	F4	☑	ADLIB:OUTBUF	Bank7
9	GPIO_OUT[2]	Output	LVCMOS25	F3	☑	ADLIB:OUTBUF	Bank7
10	GPIO_OUT[3]	Output	LVCMOS25	G7	☑	ADLIB:OUTBUF	Bank7
11	GPIO_OUT[4]	Output	LVCMOS25	H7	☑	ADLIB:OUTBUF	Bank7
12	GPIO_OUT[5]	Output	LVCMOS25	J6	☑	ADLIB:OUTBUF	Bank7
13	GPIO_OUT[6]	Output	LVCMOS25	H6	☑	ADLIB:OUTBUF	Bank7
14	GPIO_OUT[7]	Output	LVCMOS25	H5	☑	ADLIB:OUTBUF	Bank7
15	(P) REFCLK0_P	Input	LVDS	U1	☑	ADLIB:INBUF_DIFF	Bank5
16	(N) REFCLK0_N	Input	LVDS	T1	☑	ADLIB:INBUF_DIFF	Bank5
17	RXD0_N	Input	--	Y1	☑	ADLIB:SERDESIF_0	--
18	RXD0_P	Input	--	W1	☑	ADLIB:SERDESIF_0	--
19	RXD1_N	Input	--	Y3	☑	ADLIB:SERDESIF_0	--
20	RXD1_P	Input	--	W3	☑	ADLIB:SERDESIF_0	--
21	RXD2_N	Input	--	Y5	☑	ADLIB:SERDESIF_0	--
22	RXD2_P	Input	--	W5	☑	ADLIB:SERDESIF_0	--
23	RXD3_N	Input	--	Y7	☑	ADLIB:SERDESIF_0	--
24	RXD3_P	Input	--	W7	☑	ADLIB:SERDESIF_0	--
25	SDIF0_PERST_N	Input	LVCMOS25	P18	☑	ADLIB:INBUF	Bank2
26	SWITCH	Input	LVCMOS25	J18	☑	ADLIB:INBUF	Bank2

Figure 46 • **I/O Editor**

These pin assignments are for connecting the following components on the IGLOO2 Evaluation Kit:

- CLK to 50 MHz Clock Oscillator
- GPIO_OUT [0] to GPIO_OUT [7] for LEDs
- GPIO_IN [0] to GPIO_IN [3] for DIP switches
- SWITCH for SW4
- SDIF0_PERST_N is reset signal from PCIe edge connector

3. After updating I/O editor, click **Commit and Check**.
4. Close the I/O editor.
5. Click **Generate Bitstream** as shown in Figure 47 to complete place and route, verify timing, and generate the programming file.

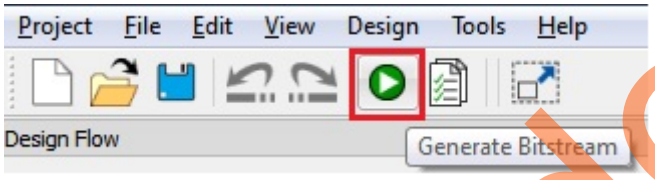


Figure 47 • **Generate Programming Data**

Step 5: Programming the IGLOO2 Board Using FlashPro

1. Connect the FlashPro4 programmer to the J5 connector of the IGLOO2 FPGA Evaluation Kit.
2. Connect the jumpers on the IGLOO2 FPGA Evaluation Kit as shown in Table 7.

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

Table 7 • **IGLOO2 FPGA Evaluation Kit Jumper Settings**

Jumper	Pin (from)	Pin (to)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

3. Connect the power supply to the J6 connector.
4. Switch the power supply switch **SW7** to **ON** position. Refer to the "IGLOO2 Evaluation Kit Board" section for further details.
5. To program the IGLOO2 device, double-click **Run Programming Action** in the **Design Flow** window as shown in Figure 48.

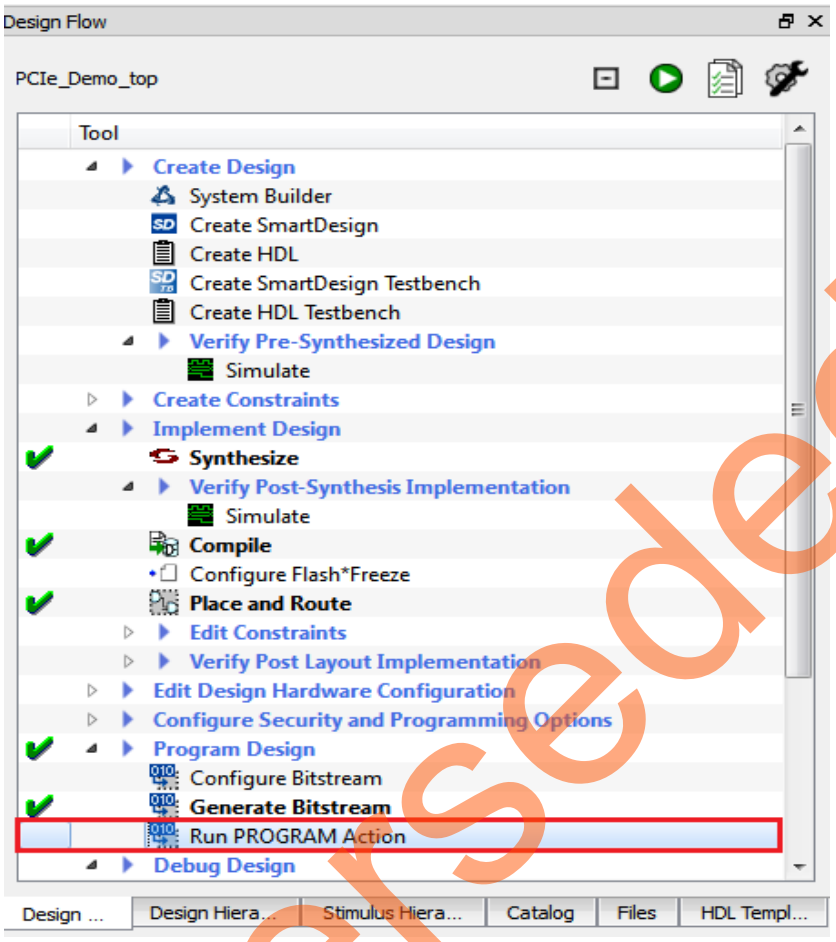


Figure 48 **• Run Programming Action**

Step 6: Connecting the Evaluation Kit to the Host PC

1. After successful programming, power off the IGLOO2 Evaluation Kit and shut down the host PC.
2. Use the below steps to connect the CON1–PCIe Edge Connector either to host PC or laptop,
   a. Connect the CON1–PCIe Edge Connector to host PC's PCIe Gen2 slot or Gen1 slot as applicable. This tutorial is designed to run in any PCIe Gen2 compliant slot. If your host PC does not support the Gen2 compliant slot, the design switches to the Gen1 mode.
   b. Connect the CON1–PCIe Edge Connector to the laptop PCIe slot using the express card adapter. If you are using a laptop, the express card adapters typically support only Gen1 and the design works on Gen1 mode.

Note: Host PC or laptop should be powered OFF while inserting the PCIe Edge Connector. If you do not power off the system, the PCIe device detection and selection of Gen1 or Gen2 do not occur properly. It is recommended that the host PC or laptop should be powered off during the PCIe card insertion.

3. Figure 49 shows the board setup for the host PC in which IGLOO2 Evaluation Kit is connected to the host PC PCIe slot. To connect the IGLOO2 Evaluation Kit to the Laptop using Express card adapter, refer to the "IGLOO2 Evaluation Kit Board Setup for Laptop" section.

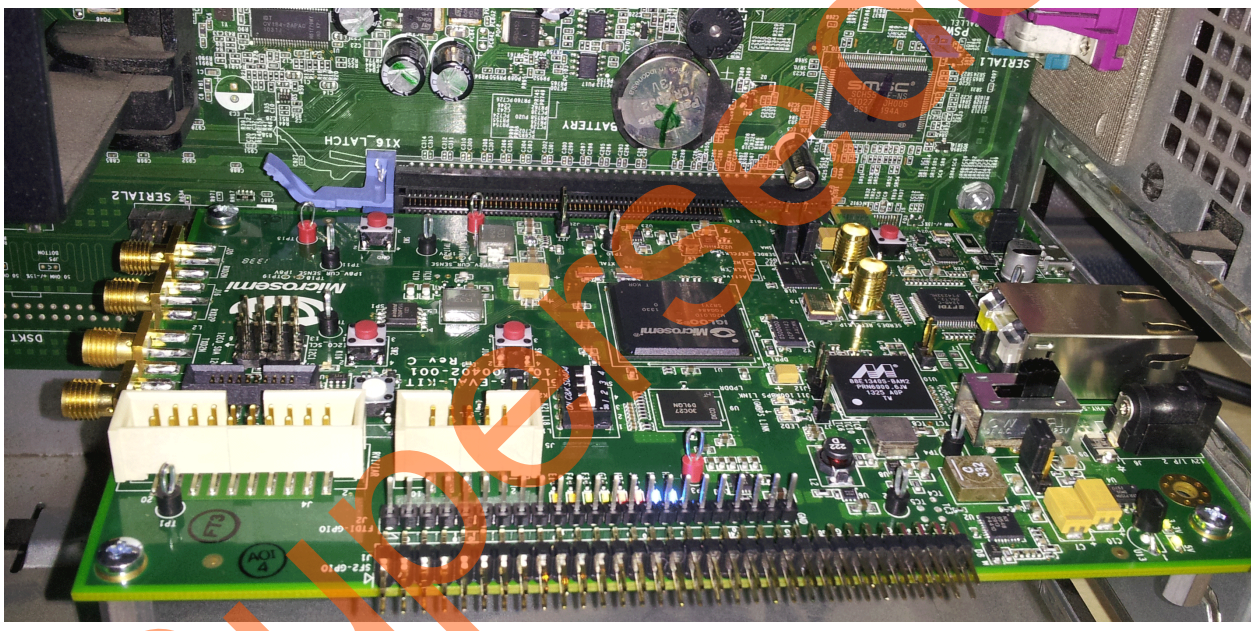


Figure 49 • **IGLOO2 Evaluation Kit Setup for Host PC**

Step 7: Running the Design

This design can be run on both Windows and RedHat Linux OS.

- To run the design on Windows OS GUI, Jungo drivers are provided. Refer to "Running the Design on Windows" section on page 49.
- To run the design on Linux OS, native RedHat Linux drivers and command line scripts are provided. Refer to "Running the Design on Linux" section on page 59.

Running the Design on Windows

1. Switch **ON** the **SW7** power supply switch.
2. Power on the host PC and check the host PC Device Manager for PCIe device. It will be similar to Figure 50. If the PCIe device is not detected, power cycle the IGLOO2 Evaluation Kit board and click "**scan for hardware changes**" in the Device Manager.

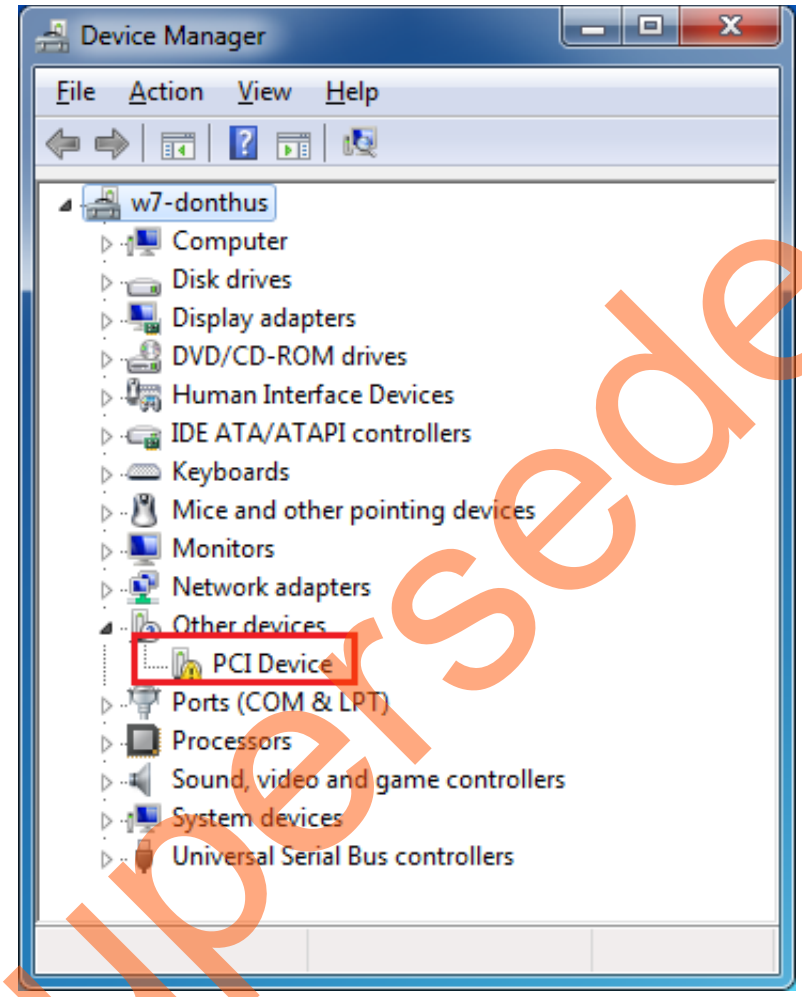


***Figure 50* • Device Manager**

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.

3. If the host PC has any other installed drivers (previous versions of Jungo drivers) for the IGLOO2 PCIe device, uninstall them. To uninstall previous versions of Jungo drivers follow step a and b.

a. To uninstall the previous Jungo drivers, go to device manager and right-click on DEVICE as shown in Figure 51. The DEVICE uninstall window is displayed.

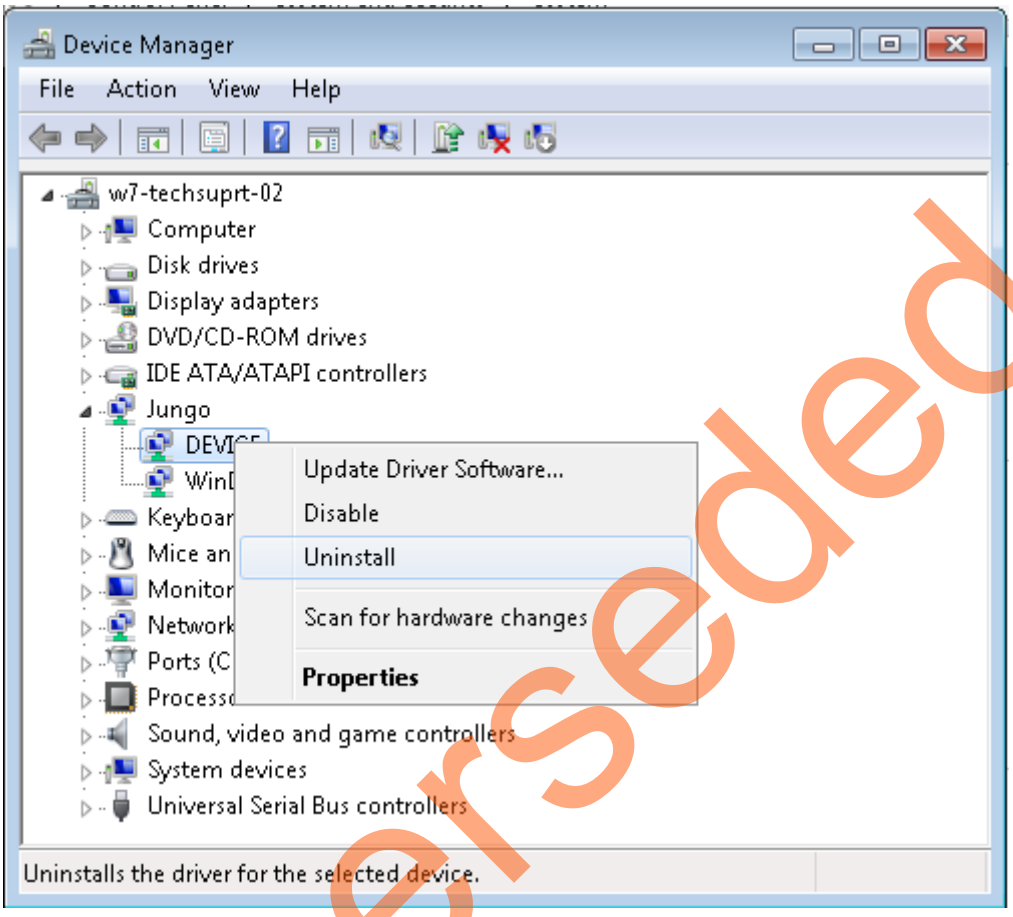


***Figure 51* • Device Uninstall**

b. Select the **Delete the driver software** check box for this device as shown in Figure 52. After uninstalling the previous Jungo drivers, make sure that the PCI Device is detected in the **Device Manager** window as shown in Figure 50.

***Figure 52* • Confirm Device Uninstall**

Installing Jungo Drivers

The PCIe tutorial uses a driver framework provided by Jungo WinDriver Pro. To install the PCIe drivers on the host PC for IGLOO2 Evaluation Kit board, use the following steps:

1. Extract the **PCIe_Demo.rar** to the **C:\ drive**.
   The PCIe_Demo.rar is located in the provided design files:
   M2GL_PCIE_Control_Plane_DF\Windows_64bit\Drivers\PCIe_Demo.rar.

Note: Installing these drivers require host PC Administration rights.

2. Run the batch file *C:\PCIe_Demo\DriverInstall\Jungo_KP_install.bat*.
3. Click **Install** if the window is displayed as shown in Figure 53.

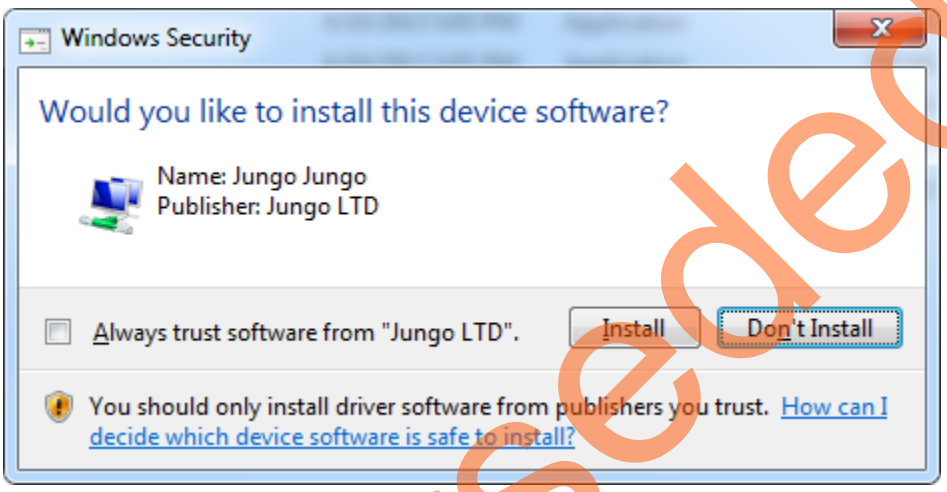


Figure 53 • **Jungo Driver Installation**

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 54.

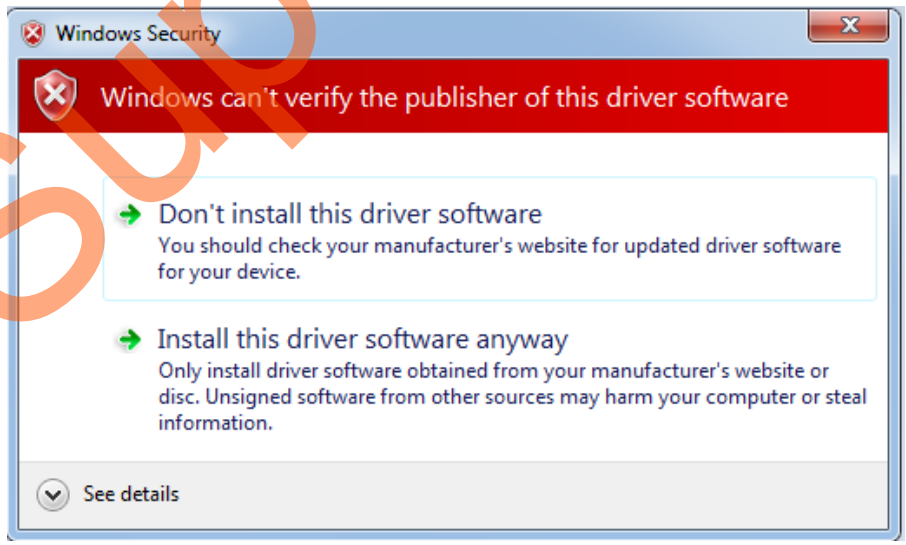


Figure 54 • **Windows Security**

Installing the PCIe GUI

The IGLOO2 PCIe graphic user interface (GUI) is a simple GUI that runs on the host PC to communicate with the IGLOO2 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 your selection. Use the following steps to install the GUI:

1. Extract the PCIe_Demo_GUI_Installer.rar from the provided design files: *M2GL_PCIE_Control_Plane_DF\Windows_64bit\GUI*.
2. Double-click the **setup.exe** in the provided GUI installation (*PCIe_Demo_GUI_Installer\setup.exe*). Apply default options as shown in Figure 55.

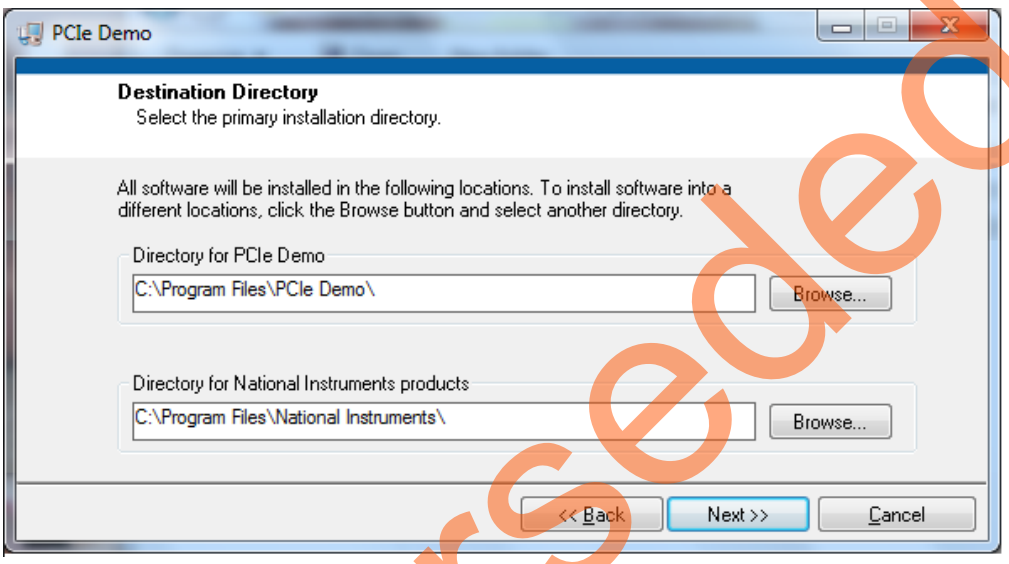


***Figure 55* • GUI Installation**

3. Click **Next** to complete the installation. The Installation Complete window is displayed.

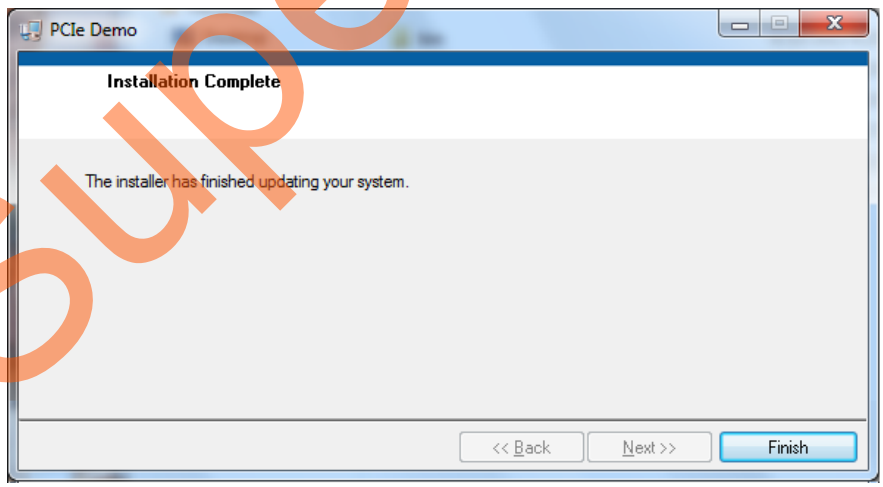


***Figure 56* • Successful GUI Installation**

4. Restart the host PC.

Running the PCIe GUI

1. Check the host PC **Device Manager** for the drivers. Make sure that the board is switched on. If the device is not detected, power cycle the IGLOO2 Evaluation Kit board and click "**scan for hardware changes**" in the **Device Manager**.

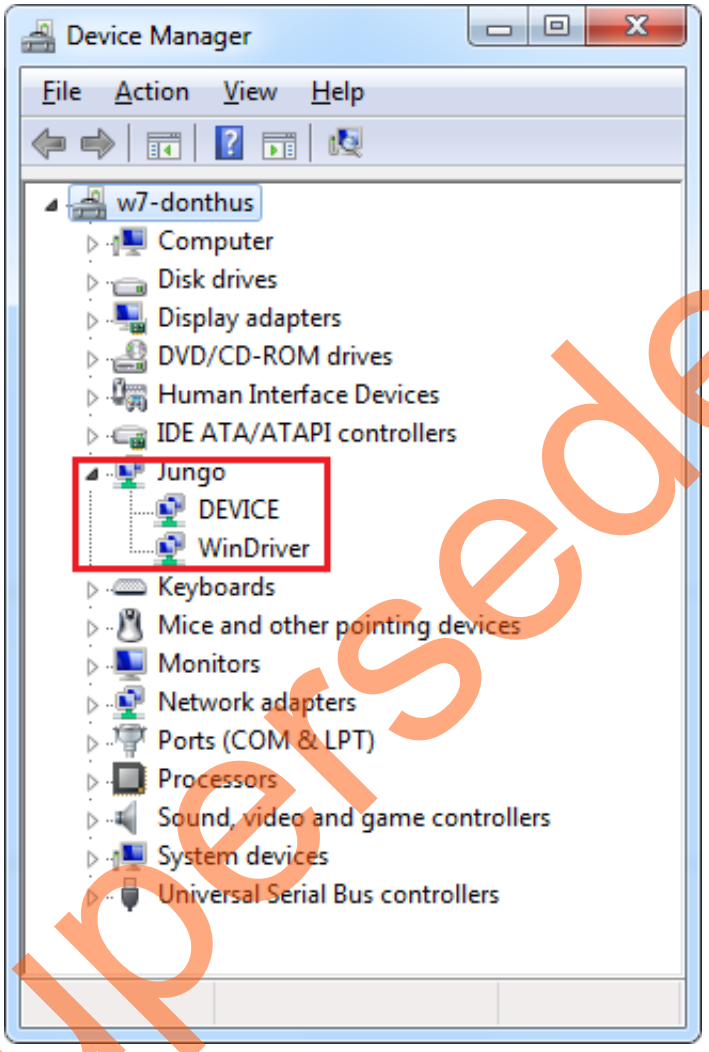


Figure 57 • **Device Manager - PCIe Device Detection**

Note: If a warning symbol is displayed on the **DEVICE** or **WinDriver** icons in the **Device Manager**, uninstall them and start from Step1 of the "Step 7: Running the Design" section.

2. Invoke the GUI from **ALL Programs > PCIe Demo > PCIe Demo GUI**. The GUI is displayed as shown in Figure 58.

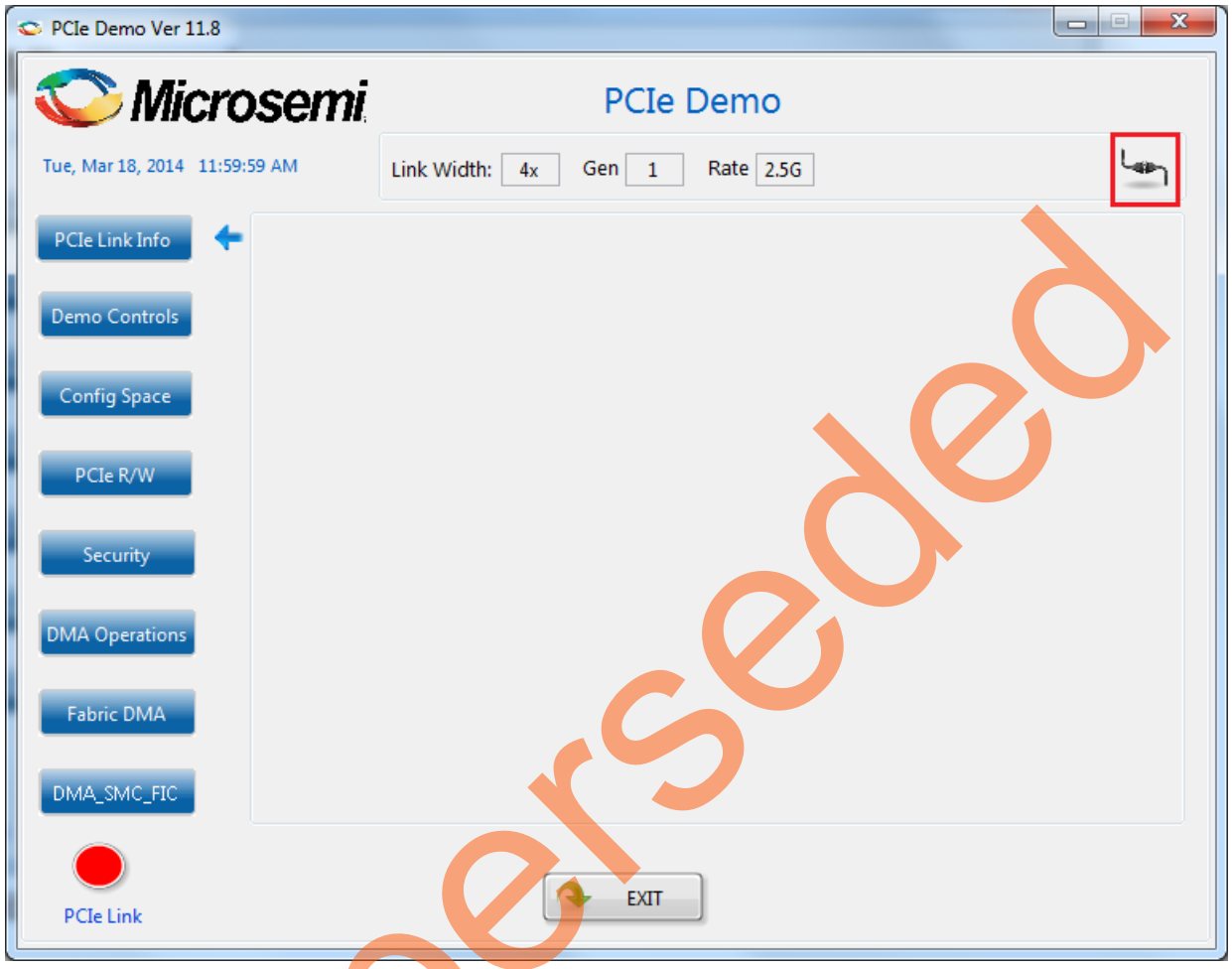


Figure 58 • **PCIe Demo GUI**

3. Click the **Connect** button at the top-right corner of the GUI. The messages are displayed on the GUI as shown in Figure 59.

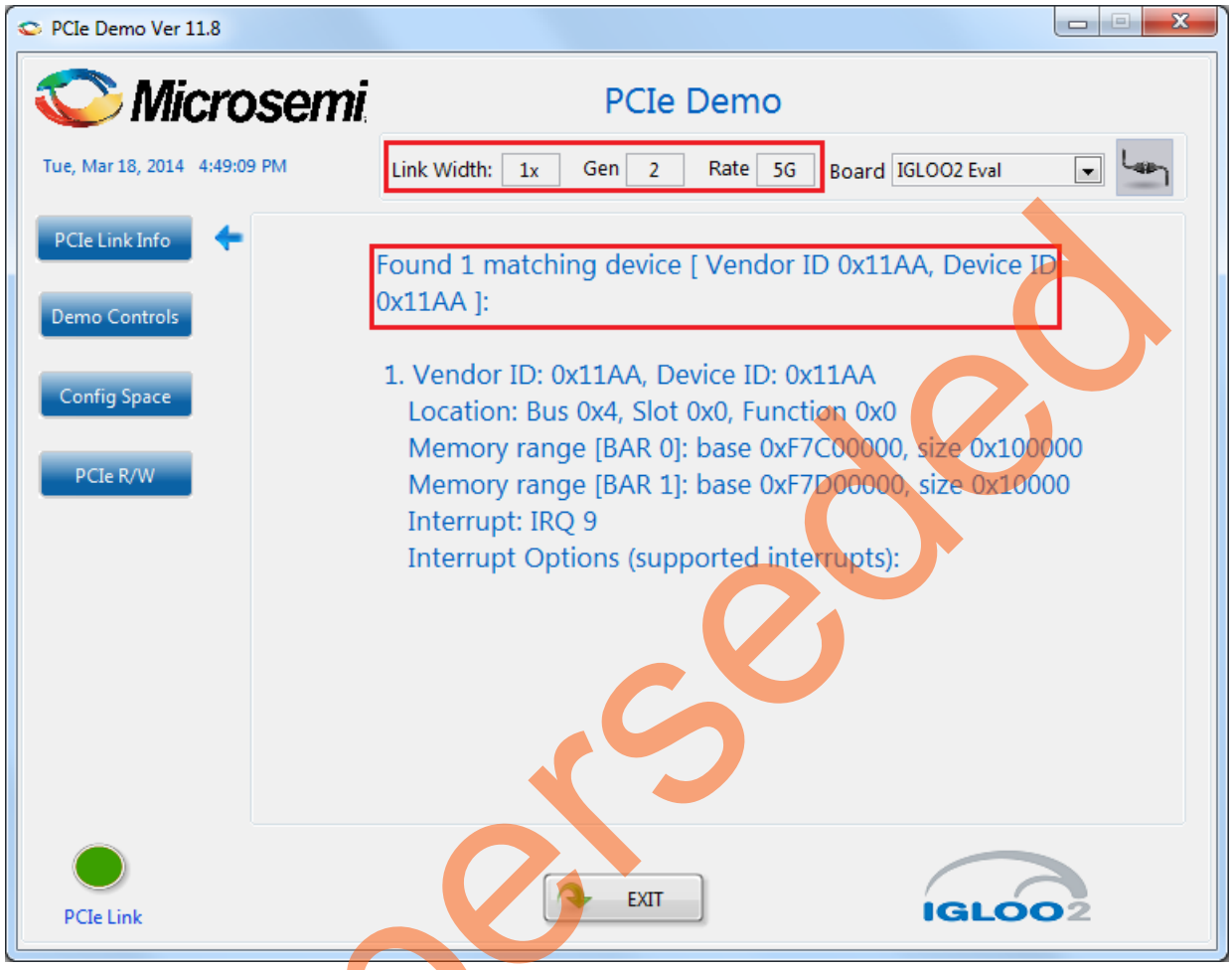


Figure 59 • **Version Information**

Note: If the host PC does not support GEN2 slot, then this design will run at GEN1 speed.

4. Clicking **Demo Controls** in the GUI displays the LED options and DIP switch positions as shown in Figure 60.

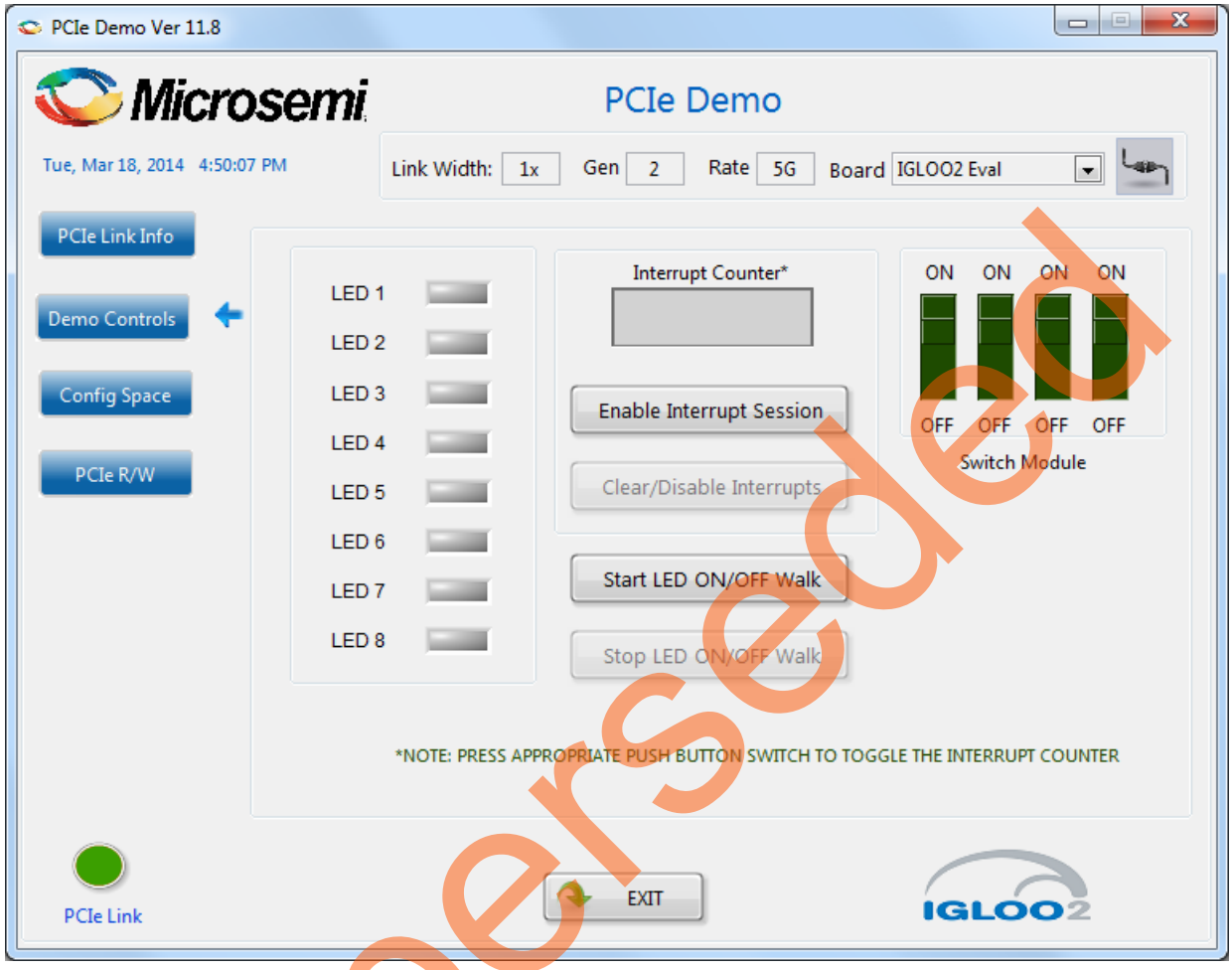


Figure 60 • **Demo Controls**

5. Click LEDs in GUI to ON/OFF the LEDs on the IGLOO2 Evaluation Kit.
6. Click **Start LED ON/OFF Walk** to make the LEDs on IGLOO2 Evaluation Kit blink.
7. Click **Stop LED ON/OFF Walk** to stop the LEDs blinking.
8. Change the DIP switch positions on the IGLOO2 Evaluation Kit (**SW5**) and observe the similar position of switches in the **GUI SWITCH MODULE**.
9. Click **Enable Interrupt Session** to enable the PCIe interrupt.

10. Press the push button **SW4** on the IGLOO2 Evaluation Kit and observe the interrupt count on the **Interrupt Counter** field in the GUI as shown in Figure 61.

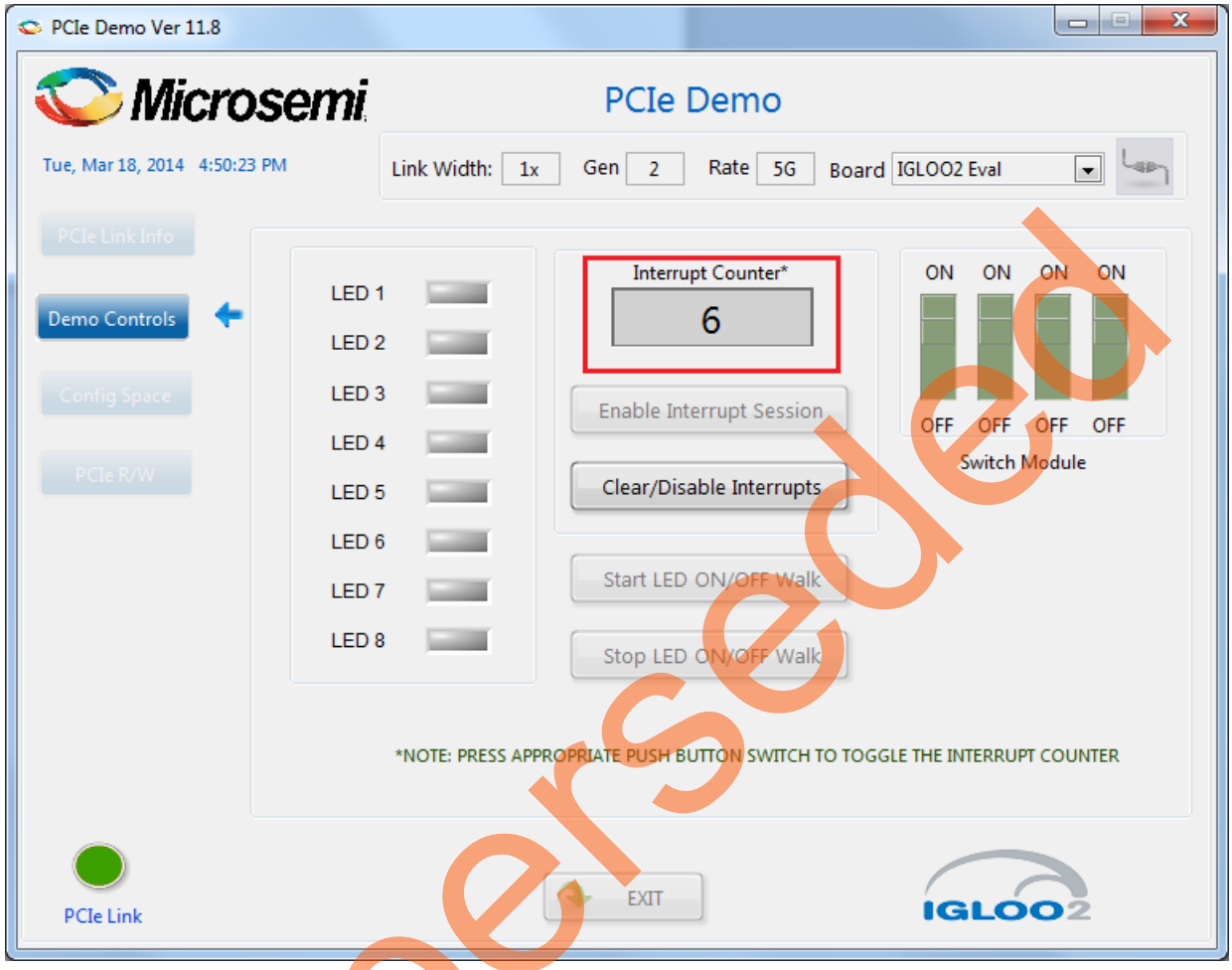


Figure 61 • **Interrupt Counter**

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 62 shows the PCIe configuration space.

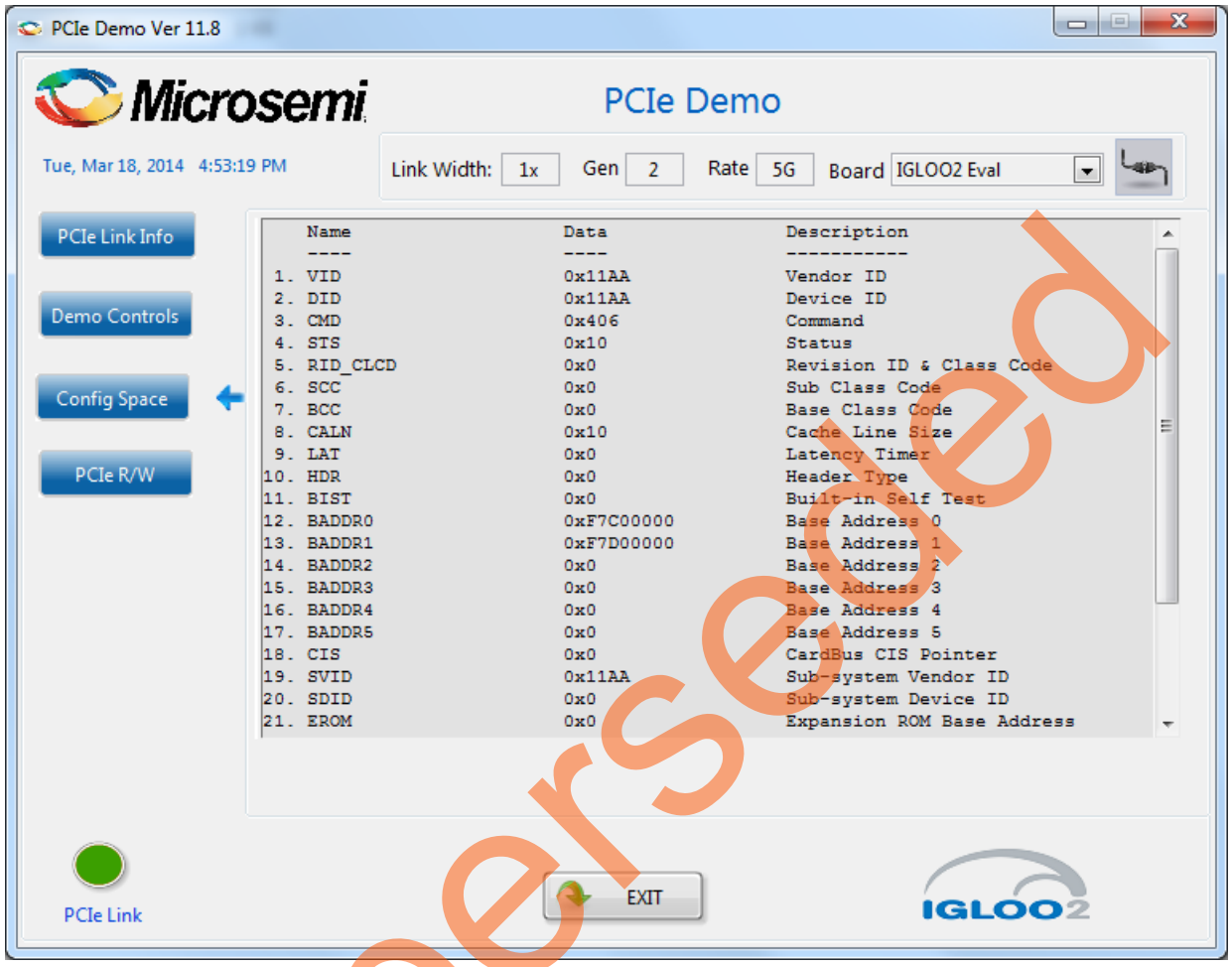


Figure 62 • **Configuration Space**

13. Click **PCIe R/W** to perform read and writes to LSRAM memory through **BAR1** space. shows the PCIe R/W window. Enter the address in the **Address** field between **0x0000** to **0x7FFC**. The **Data** field accepts a 32-bit hexadecimal value.

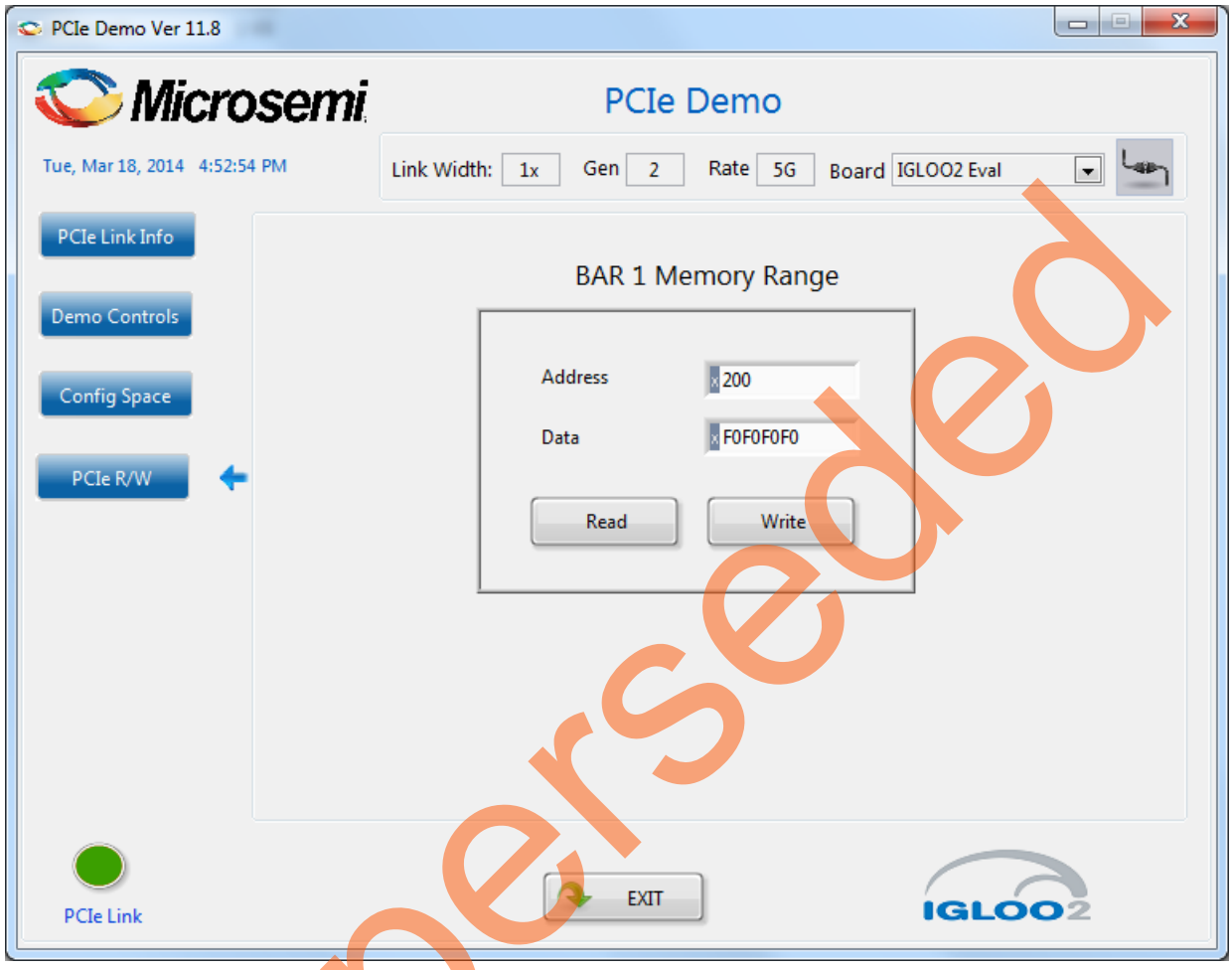


***Figure 63* • Perform Read and Write to LSRAM Using PCIe**

14. Click **Exit**.

Running the Design on Linux

1. Switch **ON** the power supply switch on the IGLOO2 Evaluation Kit board.
2. Switch **ON** the Red Hat Linux host PC.
3. Red Hat Linux Kernel detects the IGLOO2 PCIe end point as Actel Device.
4. On Linux Command Prompt Use `lspci` command to display the PCIe info.

```
# lspci
```

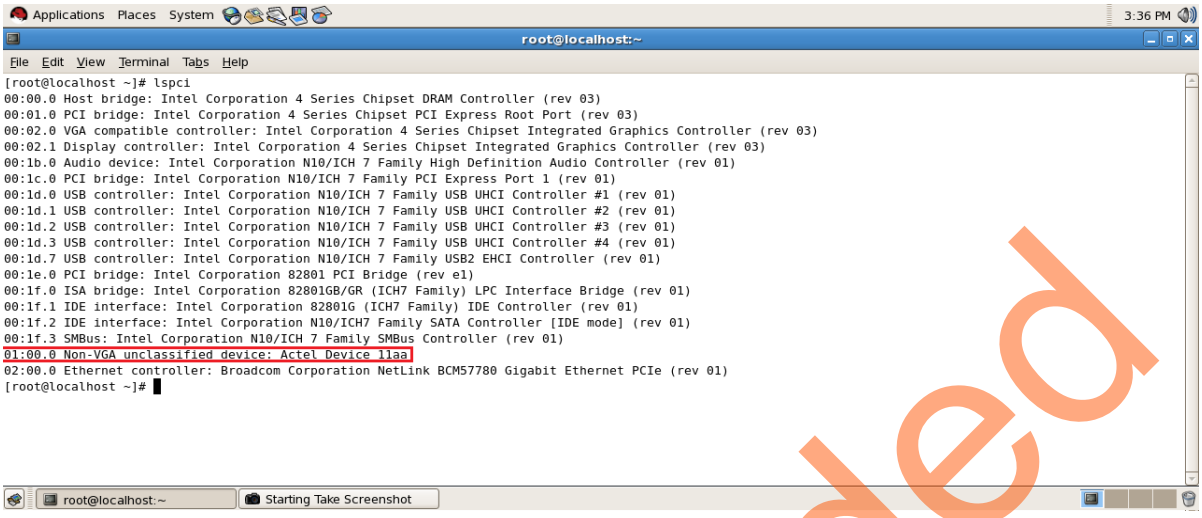


Figure 64 • **PCIe Device Detection**

Installation

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

1. Create the igl2 directory under the home/ directory using the following command:

   `# mkdir /home/igl2`

2. Copy the *M2GL_PCIE_Control_Plane_DF*\Linux_64bit\Drivers\PCIe_Driver folder from the Windows host PC and place it into the /home/igl2 directory of RedHat Linux host PC.
3. Copy the *M2GL_PCIE_Control_Plane_DF*\Linux_64bit\Drivers\inc folder from the Windows host PC and place it into the /home/igl2 directory of RedHat Linux host PC.
   The /home/igl2 directory must contain PCIe_Driver/ inc/ folders.
4. Execute ls command to display the contents of /home/igl2 directory.

   `# ls`

5. Change to inc/ directory by using the following command:

   `#cd /home/igl2/inc`

6. Edit the `board.h` file for IGLOO2 Evaluation Kit.

   `#vi board.h`

   `#define IGL2`

   `#undef SF2`

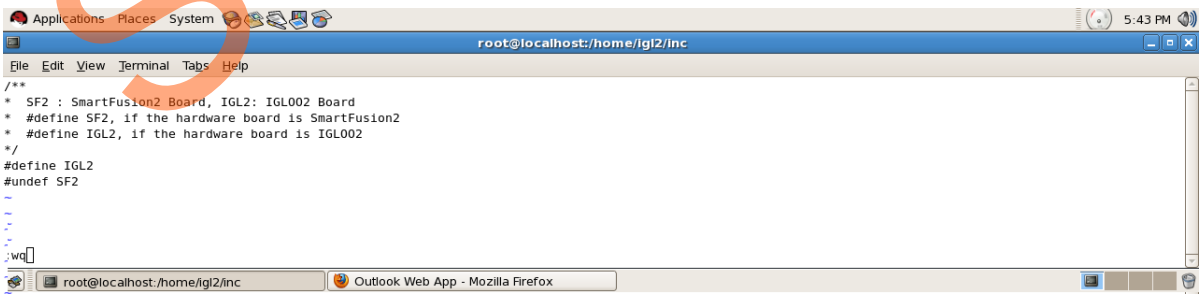


Figure 65 • **Edit board.h file**

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

   `#cd /home/igl2/PCIe_Driver`

9. To compile the Linux PCIe device driver code, execute the make command on Linux Command Prompt.

   #make clean [To clean any *.o, *.ko files]

   #make

10. The kernel module, `pci_chr_drv_ctrlpln.ko`, is created in the same directory.
11. To insert the Linux PCIe device driver as a module, execute `insmod` command on Linux Command Prompt.

    #insmod pci_chr_drv_ctrlpln.ko

Note: Root privileges are required to execute `insmod` command.

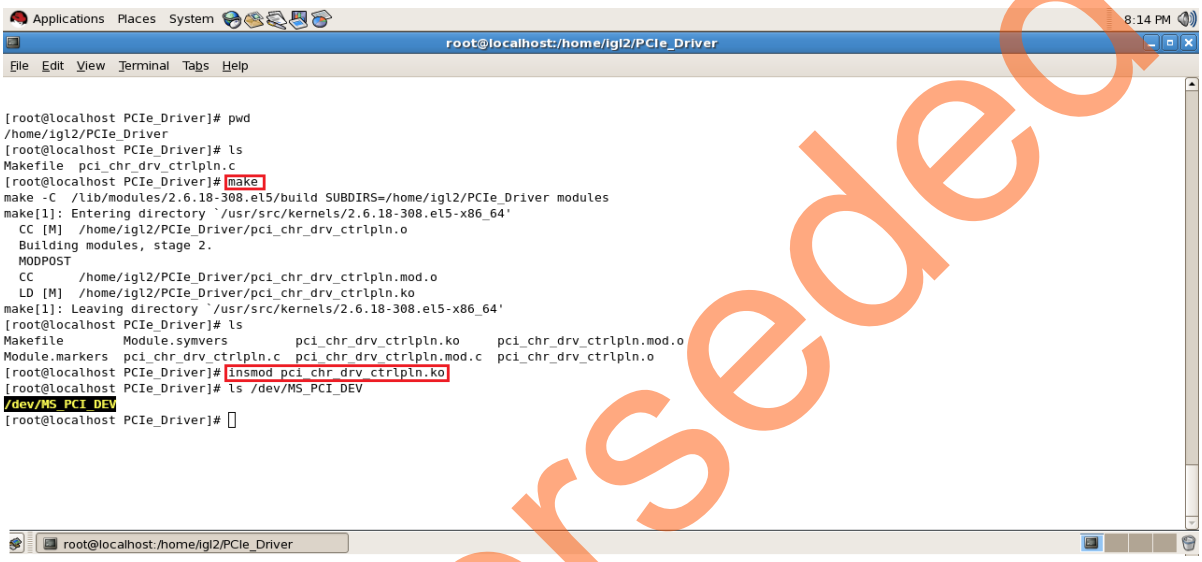


Figure 66 • **PCIe Device Driver Installation**

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

    #ls/dev/MS_PCI_DEV

Note: `/dev/MS_PCI_DEV` interface is used to access the IGLOO2 PCIe end point from Linux user space.

Linux PCIe Application Compilation and PCIe Control Plane Utility Creation

1. Change to the /home/igl2/ directory using the following command:

   # cd /home/igl2

2. Copy the *M2GL_PCIE_Control_Plane_DF*\Linux_64bit\Util\PCIe_App folder from the Windows host PC and place it into the /home/igl2 directory of RedHat Linux host PC.
3. Change to the /home/igl2/PCIe_App directory using the following command:

   #cd /home/igl2/PCIe_App

4. Compile the Linux user space application `pcie_appln_ctrlpln.c` by using **`gcc`** command.

   #gcc -o pcie_ctrlplane pcie_appln_ctrlpln.c

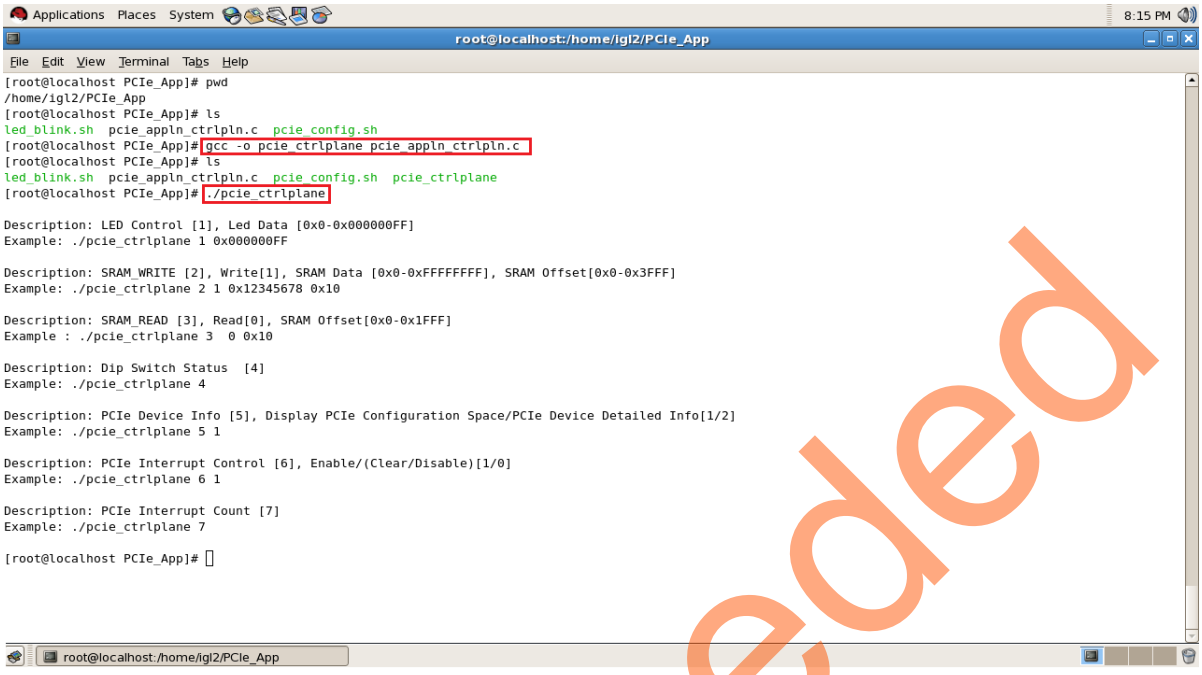


***Figure 67* • Linux PCIe Application Utility**

5. After successful compilation, the Linux PCIe application utility `pcie_ctrlplane` is created 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 67.

Execution of Linux PCIe Control Plane Features

LED Control

LED1 to LED8 is controlled by writing data to IGLOO2 LED control registers.

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

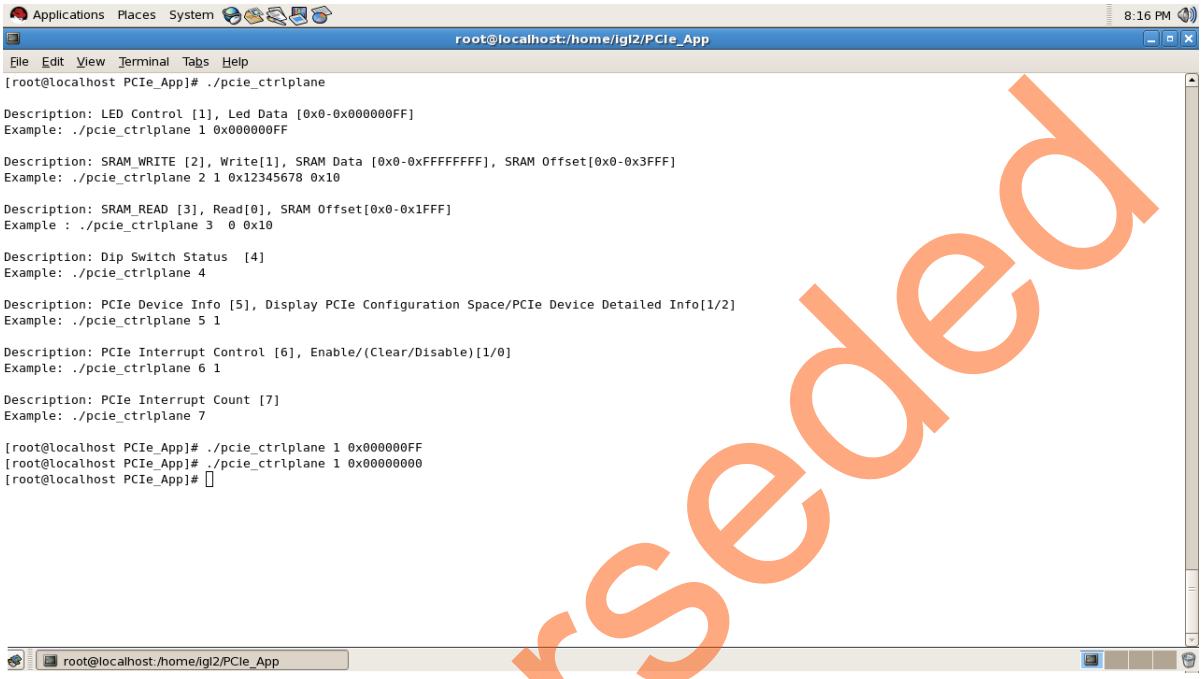


Figure 68 • **Linux Command - LED Control**

`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

32 KB SRAM is accessible for IGLOO2 Evaluation Kit.

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

#./pcie_ctrlplane 3 0 0x1000 [SRAM READ]
```

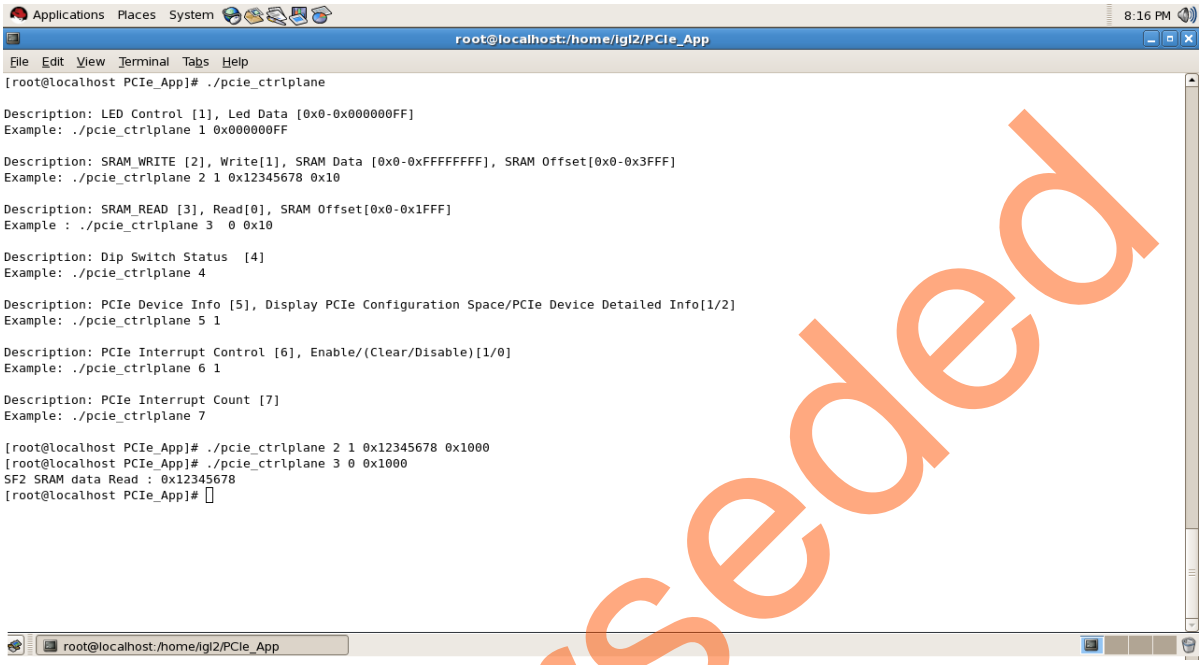


***Figure 69* • Linux Command - SRAM Read/Write**

DIP Switch Status

Dip switch on IGLOO2 Evaluation Kit board 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]
```

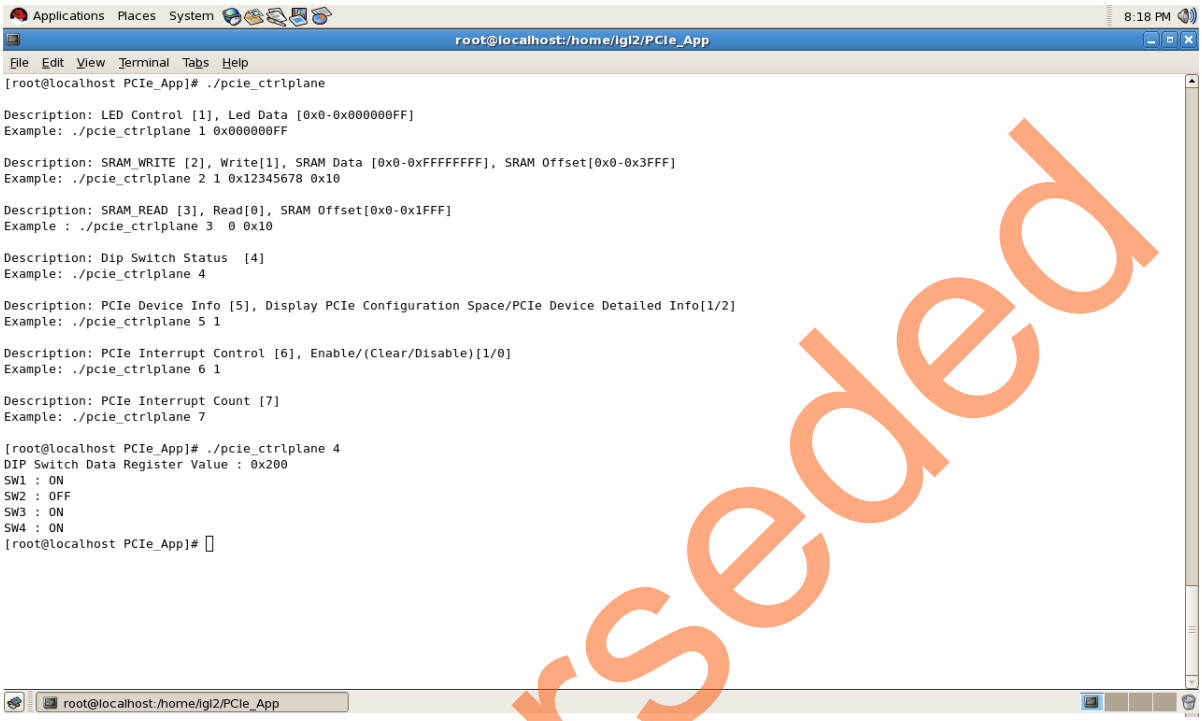


Figure 70 • **Linux Command - DIP Switch**

PCIe Configuration Space Display

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

Note: Root Privileges are required to execute this command.

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

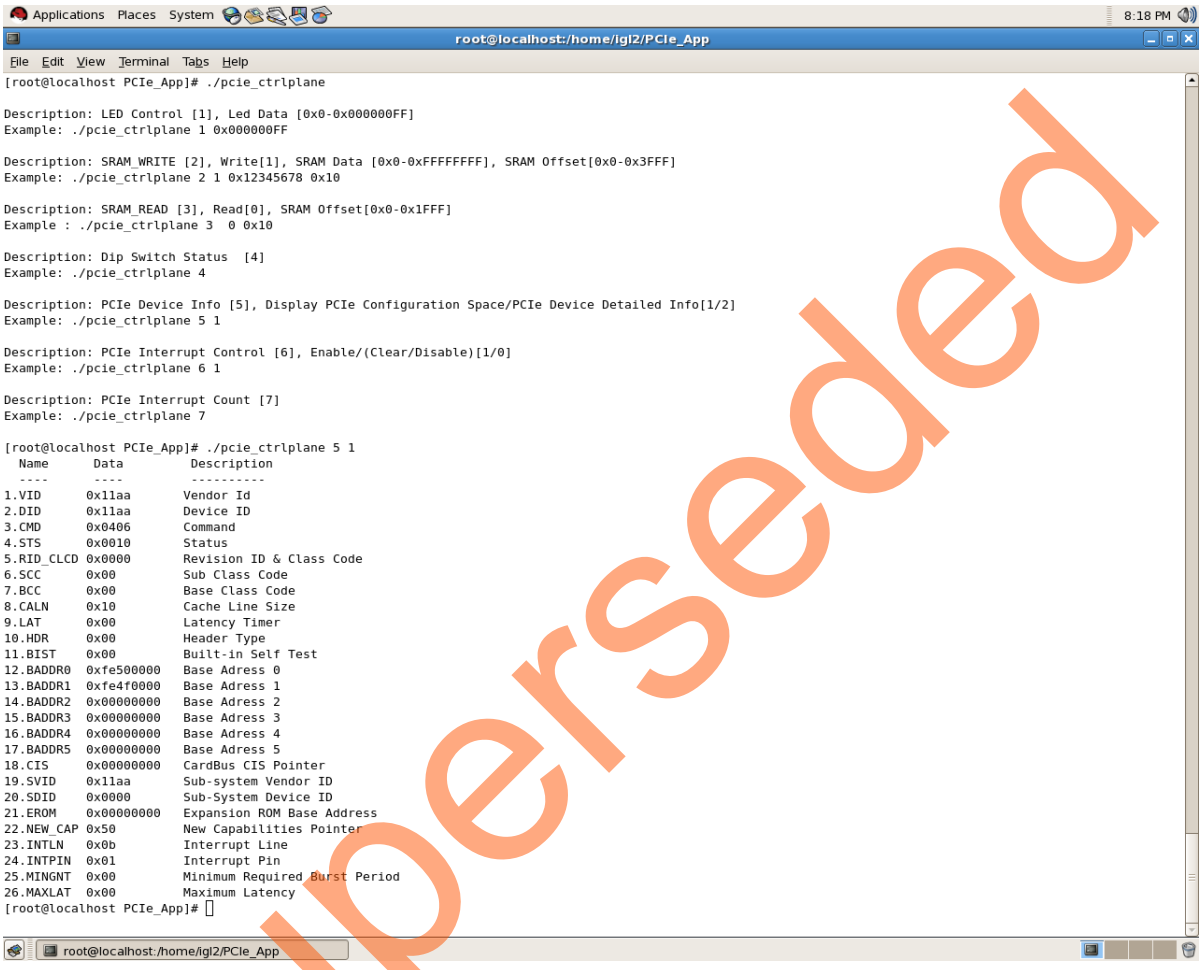


Figure 71 • **Linux Command - PCIe Configuration Space Display**

PCIe Link Speed and Width

Note: Root Privileges are required to execute this command.

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

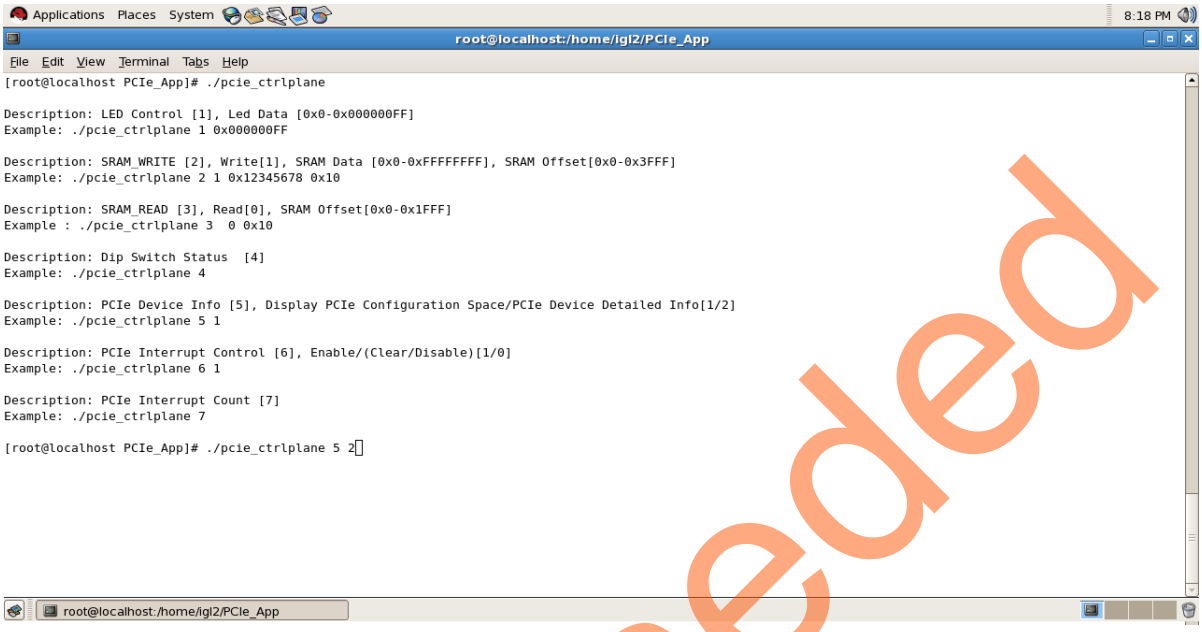


Figure 72 • **Linux Command - PCIe Link Speed and Width**

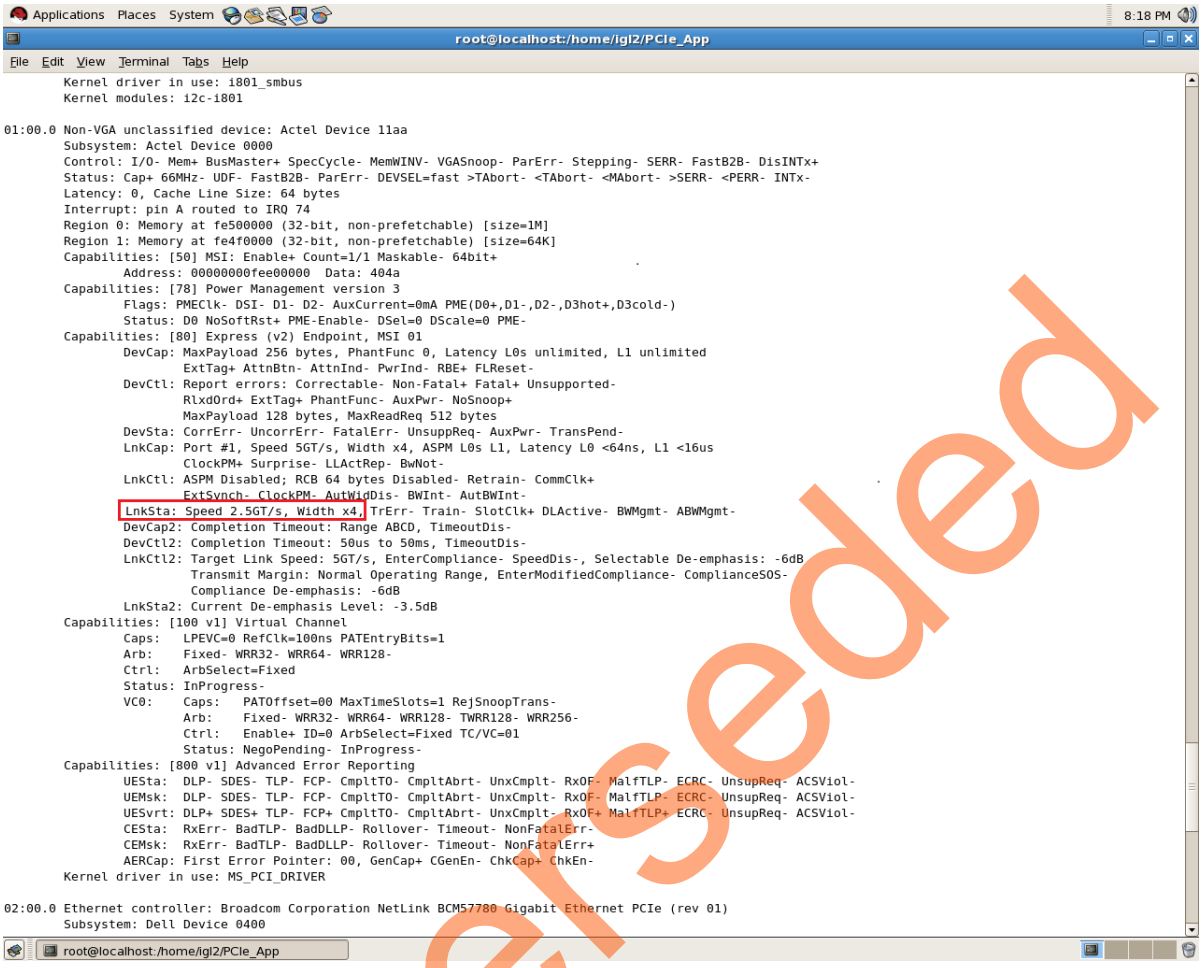


Figure 73 • **Linux Command - PCIe Link Speed and Width**

PCIe Interrupt Control (Enable/Disable) and Interrupt Counter

IGLOO2 Evaluation 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 SW4 push button.

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

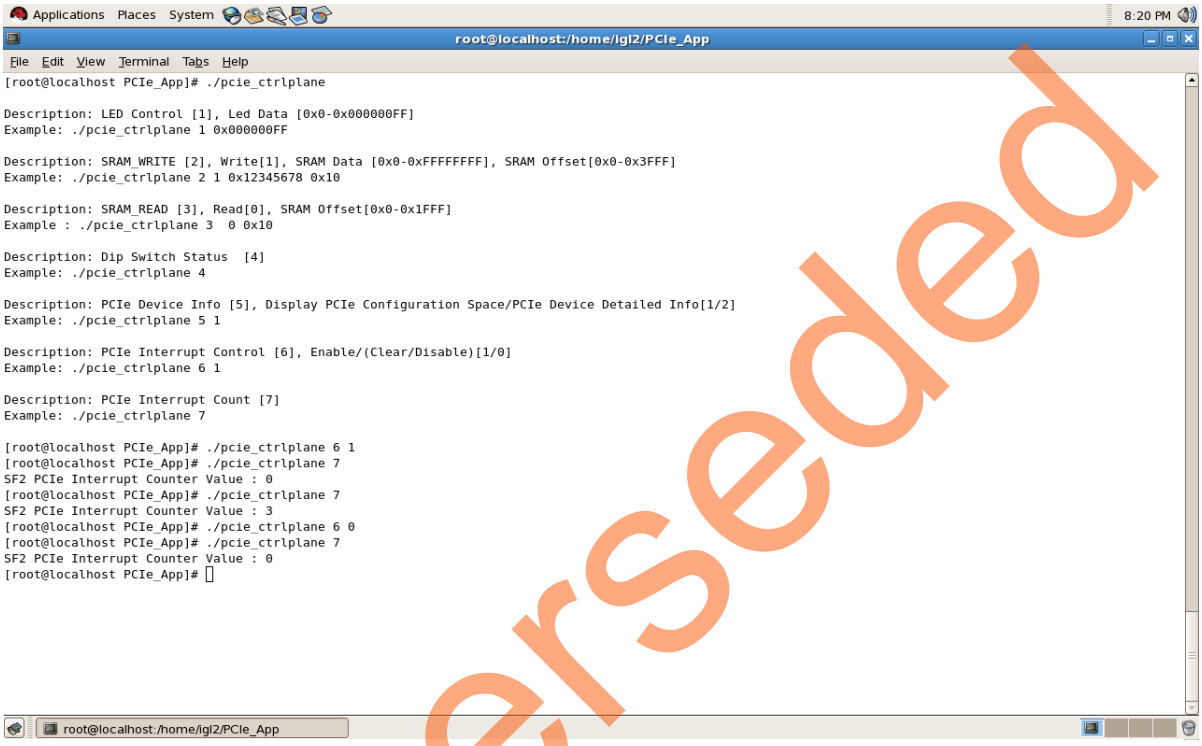


Figure 74 • **Linux Command - PCIe Interrupt Control**

Conclusion

This tutorial describes how to access the PCIe endpoint features of IGLOO2 and how to create a simple design. It describes the steps to verify the design with BFM simulation. This tutorial demonstrates that the host PC can easily communicate with IGLOO2 Evaluation Kit board through the provided GUI and Drivers.This tutorial also provides a Linux PCIe application for accessing the PCIe EP device through the Linux PCIe Device Driver.

A – IGLOO2 Evaluation Kit Board

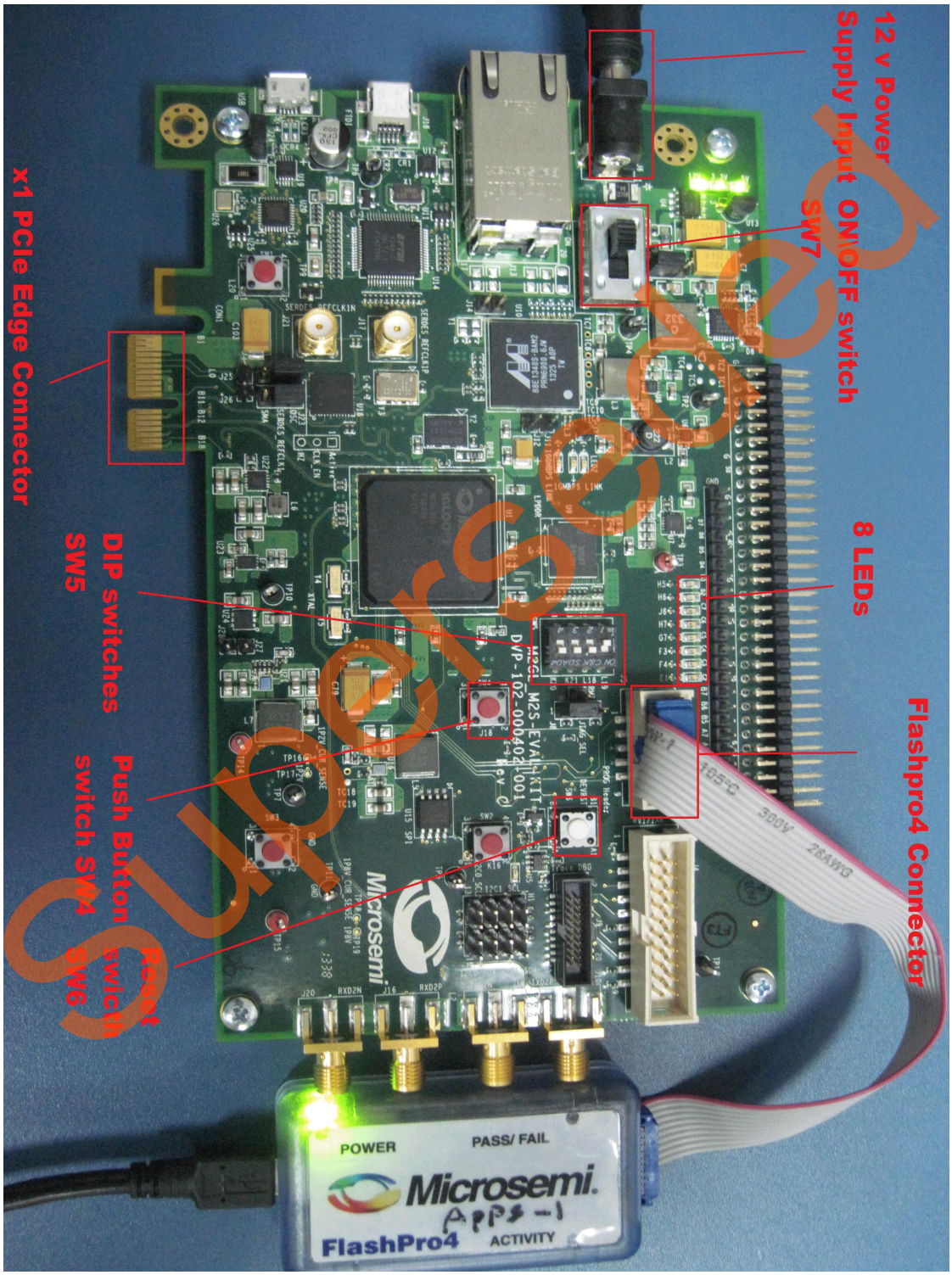


Figure 1 • IGLOO2 Evaluation Kit Board

B – IGLOO2 Evaluation Kit Board Setup for Laptop

Figure 1 shows how to line up the IGLOO2 Evaluation Kit PCIe connector with the adapter card slot.

***Figure 1* • Lining up the IGLOO2 Evaluation Kit Board**

Note: The Notch (highlighted in red) does not go into the adapter card.

Figure 2 shows IGLOO2 Evaluation Kit PCIe connector inserted into the PCIe adapter card slot.

***Figure 2* • Inserting the IGLOO2 Evaluation Kit PCIe Connector**

Figure 3 shows the PCIe adapter card and the IGLOO2 Evaluation Kit connected to the laptop.

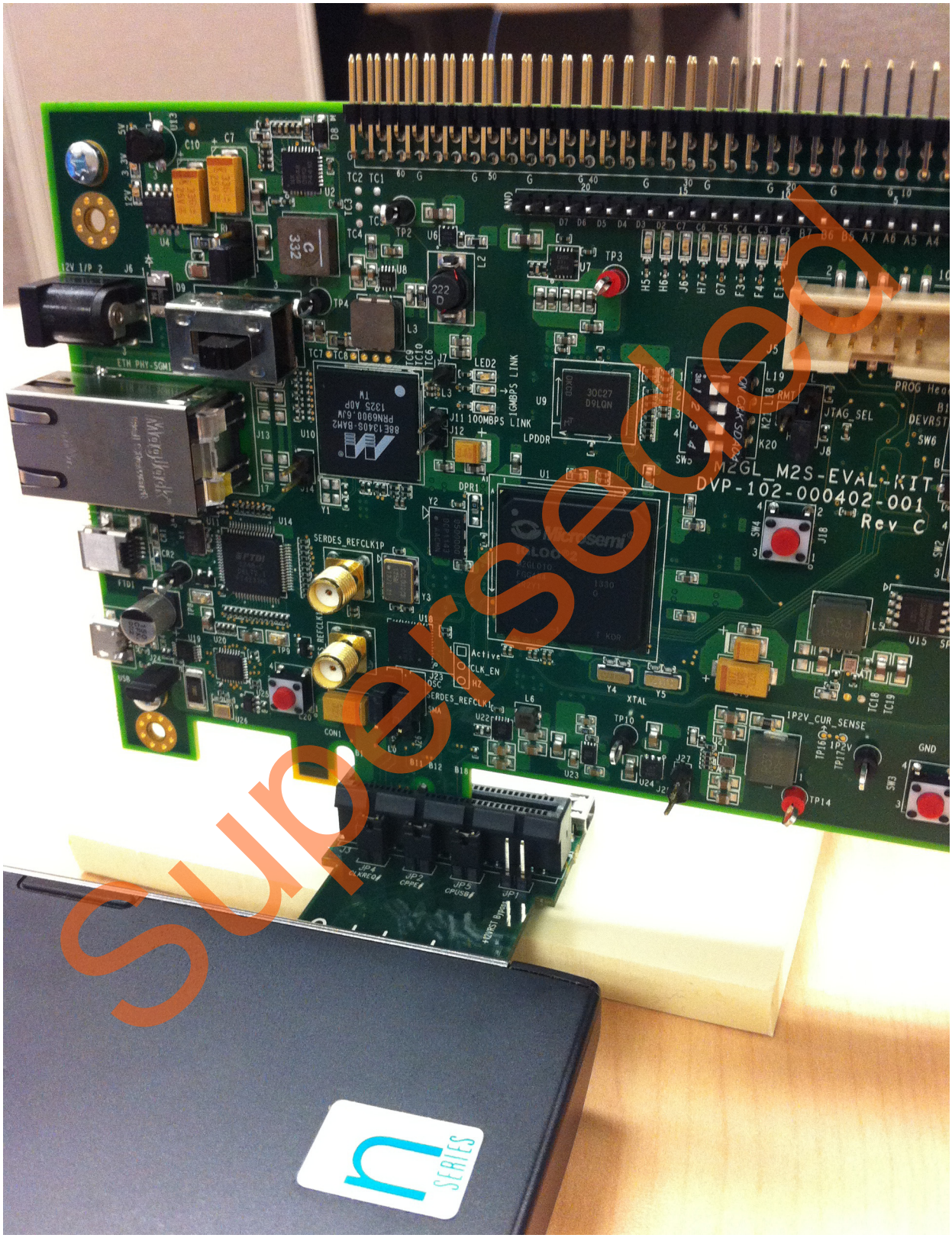


***Figure 3* • IGLOO2 Evaluation Kit Connected to the Laptop**

C – List of Changes

The following table lists critical changes that were made in each revision.

Date	Changes	Page
Revision 5 (August 2014)	Updated the design files link under "Associated Project Files" section.	4
Revision 4 (July 2014)	Updated the document for Libero v11.4 software release (SAR 59562).	NA
Revision 3 (April 2014)	Updated the document for Libero v11.3 software release (SAR 55917).	NA
Revision 2 (February 2014)	Added the section "Step 7: Running the Design".	NA
Revision 1 (January 2014)	Updated the document for Libero v11.2 software release (SAR 53311).	NA
Revision 0 (November 2013)	Initial release.	NA

D – 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 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**.

50200509-5/08.14