

TU0509
Tutorial
Implementing PCIe Control Plane Design in IGLOO2
FPGA - Libero SoC v11.8 SP1



Power Matters.™

Microsemi Corporate Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

© 2017 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.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Contents

1	Revision History	1
1.1	Revision 10.0	1
1.2	Revision 9.0	1
1.3	Revision 8.0	1
1.4	Revision 7.0	1
1.5	Revision 6.0	1
1.6	Revision 5.0	1
1.7	Revision 4.0	1
1.8	Revision 3.0	1
1.9	Revision 2.0	1
1.10	Revision 1.0	1
2	Implementing PCIe Control Plane Design in IGLOO2 FPGA	2
2.1	Design Requirements	2
2.1.1	Project Files	3
2.1.2	Components Used	3
2.2	Design Overview	4
2.3	Step 1: Creating a Libero SoC Project	5
2.3.1	Launching Libero SoC	5
2.3.2	Instantiating SERDESIF Component in PCIe_Demo SmartDesign	12
2.3.3	Instantiating Debounce Logic in PCIe_Demo SmartDesign	15
2.3.4	Instantiating Bus Interfaces in PCIe_Demo SmartDesign	16
2.3.5	Instantiating CoreGPIO in PCIe_Demo SmartDesign	21
2.3.6	Instantiating CoreAHBLSRAM in PCIe_Demo SmartDesign	23
2.3.7	Instantiating CCC in PCIe_Demo SmartDesign	24
2.3.8	Connecting Components in PCIe_Demo SmartDesign	25
2.4	Step 2: Developing the Simulation Stimulus	31
2.5	Step 3: Simulating the Design	33
2.5.1	Generating Testbench	35
2.6	Step 4: Generating the Program File	37
2.7	Step 5: Programming the IGLOO2 Board Using FlashPro	40
2.8	Step 6: Connecting the Evaluation Kit to the Host PC	41
2.9	Step 7: Running the Design	41
2.9.1	Running the Design on Windows	42
2.9.2	Running the Design on Linux	49
2.10	Conclusion	57
3	Appendix: IGLOO2 Evaluation Kit Board	58
4	Appendix: IGLOO2 Evaluation Kit Board Setup for Laptop	59

Figures

Figure 1	Design Files Top-Level Structure	3
Figure 2	PCIe Control Plane Block Diagram	4
Figure 3	Libero SoC Project Manager	5
Figure 4	Libero SoC New Project Dialog Box	6
Figure 5	New Project—Device Selection	7
Figure 6	New Project—Device Settings	8
Figure 7	New Project—Device Settings	8
Figure 8	New Project Information	9
Figure 9	System Builder Dialog Box	9
Figure 10	System Builder—Device Features Tab	10
Figure 11	System Builder—Clocks Tab	11
Figure 12	IGLOO2 FPGA System Builder Generated System	12
Figure 13	IP Catalog	12
Figure 14	High Speed Serial Interface Configurator Window	13
Figure 15	PCIe Configuration for Protocol 1	14
Figure 16	Master Interface Configuration Dialog Box	15
Figure 17	DEBOUNCE Component in Design Hierarchy Window	15
Figure 18	DEBOUNCE Component in the PCIe_Demo SmartDesign Canvas	16
Figure 19	IP Catalog	16
Figure 20	CoreAHBLite, CoreAHBtoAPB3, and CoreAPB3 Bus Interfaces in PCIe_Demo SmartDesign Canvas	17
Figure 21	Configuring CoreAHBLite_0	18
Figure 22	Configuring CoreAPB3_0	19
Figure 23	CoreAHBLite and CoreAPB3 Bus Interfaces in PCIe_Demo SmartDesign Canvas After Configuration	20
Figure 24	IP Catalog	21
Figure 25	Configuring CoreGPIO_0	22
Figure 26	IP Catalog	23
Figure 27	Configuring COREAHBLSRAM_0	23
Figure 28	Catalog	24
Figure 29	Configure CCC	24
Figure 30	PCIe_Demo in SmartDesign	25
Figure 31	Quick Connect Window	26
Figure 32	Edit Slices	27
Figure 33	Edit Slices	28
Figure 34	Edit Slices	28
Figure 35	Edit Slices	29
Figure 36	PCIe_Demo Design	30
Figure 37	Generate Component	30
Figure 38	Log Window	31
Figure 39	SmartDesign Generated SERDESIF_0_user.bfm File	31
Figure 40	Modified SERDES User BFM	33
Figure 41	Wave.do file	34
Figure 42	Project Settings—Do File Simulation Runtime Settings	34
Figure 43	Project Settings—Waveform	35
Figure 44	HDL Testbench	35
Figure 45	Create New HDL Testbench File	36
Figure 46	SERDES BFM Simulation	36
Figure 47	Simulation Result with GPIO_OUT Signals	37
Figure 48	Manage Constraints	37
Figure 49	Associate the Constraints File	37
Figure 50	I/O Constraints	37
Figure 51	I/O Editor	39
Figure 52	Generate Programming Data	39

Figure 53	Run PROGRAM Action	40
Figure 54	IGLOO2 Evaluation Kit Setup for Host PC	41
Figure 55	Device Manager	42
Figure 56	Update Driver Software	43
Figure 57	Browse for Driver Software	43
Figure 58	Browse for Driver Software Continued	44
Figure 59	Windows Security	44
Figure 60	Successful Driver Installation	44
Figure 61	GUI Installation	45
Figure 62	Successful GUI Installation	45
Figure 63	Device Manager—PCIe Device Detection	46
Figure 64	PCIe Demo GUI	46
Figure 65	Device Info	47
Figure 66	Demo Controls	47
Figure 67	Demo Controls—Continued	48
Figure 68	Configuration Space	48
Figure 69	PCIe BAR1 Memory Access	49
Figure 70	PCIe Device Detection	49
Figure 71	Edit board.h file	50
Figure 72	PCIe Device Driver Installation	51
Figure 73	Linux PCIe Application Utility	51
Figure 74	Linux Command—LED Control	52
Figure 75	Linux Command—SRAM Read/Write	53
Figure 76	Linux Command—DIP Switch]	54
Figure 77	Linux Command—PCIe Configuration Space Display	55
Figure 78	Linux Command—PCIe Link Speed and Width	56
Figure 79	Linux Command—PCIe Link Speed and Width	56
Figure 80	Linux Command—PCIe Interrupt Control	57
Figure 81	IGLOO2 Evaluation Kit Board	58
Figure 82	Lining Up the IGLOO2 Evaluation Kit Board	59
Figure 83	Inserting the IGLOO2 Evaluation Kit PCIe Connector	60
Figure 84	IGLOO2 Evaluation Kit Connected to the Laptop	61

Tables

Table 1	Design Requirements	2
Table 2	SDIF0_PINS	26
Table 3	INIT_PINS	26
Table 4	HPMS_READY Connections	26
Table 5	GL0 Clock Connections	27
Table 6	Port to Pin Mapping	38
Table 7	IGLOO2 FPGA Evaluation Kit Jumper Settings	40

1 Revision History

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

1.1 Revision 10.0

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

1.2 Revision 9.0

Updated the document for Libero v11.8 software release.

1.3 Revision 8.0

Updated the document for Libero v11.7 software release (SAR 77598).

1.4 Revision 7.0

Updated the document for Libero v11.5 software release (SAR 63980).

1.5 Revision 6.0

Updated the design files link under "Project Files" section.

1.6 Revision 5.0

Updated the document for Libero v11.4 software release (SAR 59562).

1.7 Revision 4.0

Updated the document for Libero v11.3 software release (SAR 55917).

1.8 Revision 3.0

Added the section Step 7: Running the Design.

1.9 Revision 2.0

Updated the document for Libero v11.2 software release (SAR 53311).

1.10 Revision 1.0

Initial release.

2 Implementing PCIe Control Plane Design in IGLOO2 FPGA

This tutorial demonstrates the embedded PCIe feature of IGLOO2 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 from host PC. It can run on both Windows and RedHat Linux Operating Systems (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.

The following tasks are explained in this tutorial:

- Create a Libero® SoC project
- Develop the Simulation Stimulus
- Simulate the design
- Generate the programming file
- Run the PCIe application

2.1 Design Requirements

The following table lists the hardware and software requirements of IGLOO2 PCIe control plane tutorial.

Table 1 • Design Requirements

Design Requirements	Description
Hardware	
IGLOO2 Evaluation Kit: – 12 V adapter – FlashPro4 programmer – USB A to Mini-B cable	Rev D 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	
Libero SoC	v11.8 SP1
FlashPro programming software	v11.8 SP1
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.

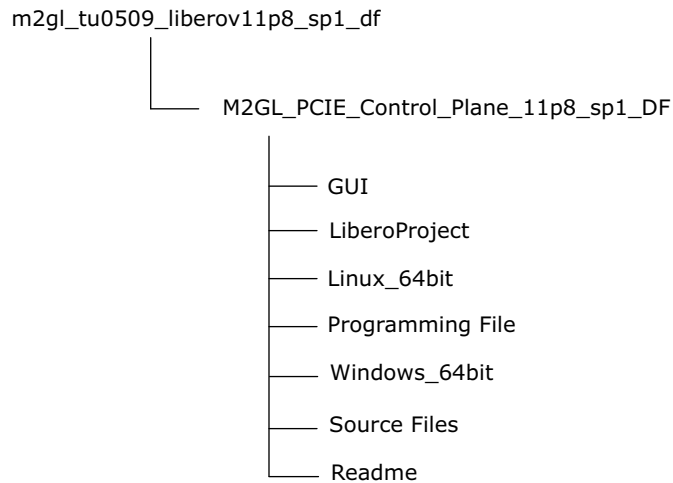
2.1.1 Project Files

The design files are available for download at:

http://soc.microsemi.com/download/rsc/?f=m2gl_tu0509_liberov11p8_sp1_df

The following figure shows the top-level structure of the design files. For further details, see the `Readme.txt` file.

Figure 1 • Design Files Top-Level Structure



2.1.2 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
- CoreAHBSRAM
- Bus interfaces CoreAHBLite, CoreAPB3, and CoreAHBTOAPB3

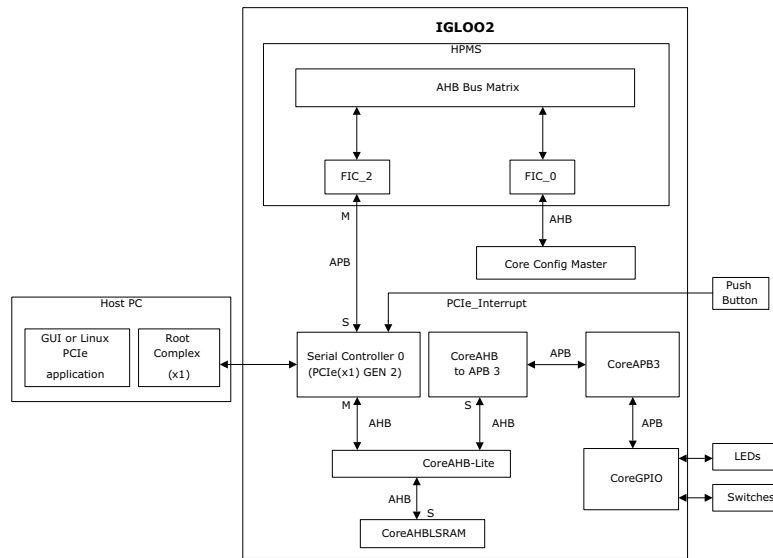
2.2 Design Overview

The 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, 1.1, and 1.0. For more information on SERDESIF, see the [UG0447: SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide](#).

The design helps accessing 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.

The following figure shows a detailed block diagram of the design implementation.

Figure 2 • 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 the CoreAHBLite, CoreAHBTOAPB, and CoreAPB3 bus interfaces.

SERDES_IF_0 is initialized by CoreConfig master. The SERDES_IF_0 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.

2.3 Step 1: Creating a Libero SoC Project

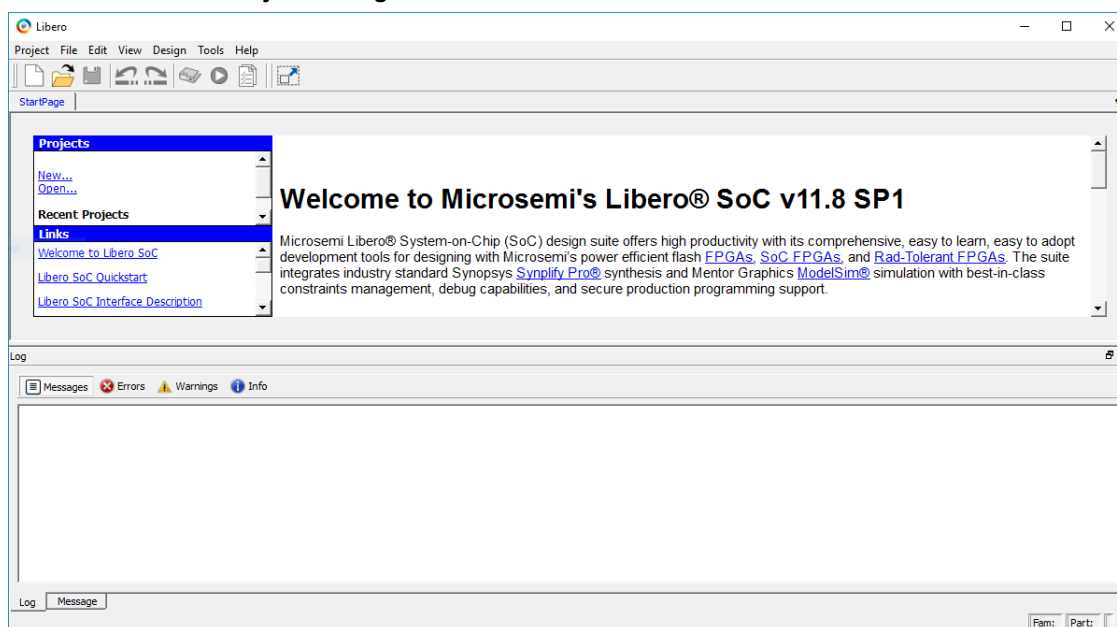
The following steps describe how to create an IGLOO2 PCIe control plane design using the Libero SoC tool.

2.3.1 Launching Libero SoC

The following steps describe how to Launching the Libero SoC:

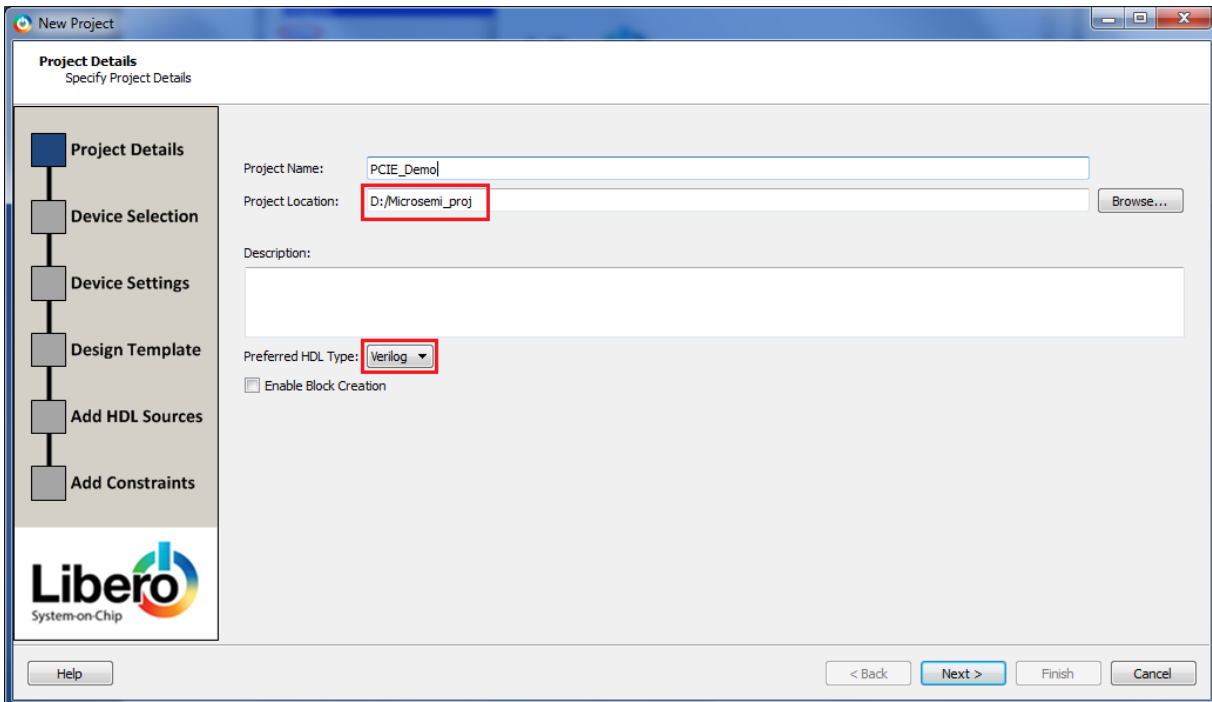
1. Go to **Start > Programs > Microsemi > Libero SoC v11.8**, or double-click the shortcut on desktop to open the Libero SoC v11.8 Project Manager.
2. Create a new project using one of the following options:
 - Select **New** on the **Start Page** tab, as shown in the following figure.
 - Click **Project > New Project** from the Libero SoC menu.

Figure 3 • Libero SoC Project Manager



3. Enter the following **New Project** information as shown in the following figure and click **Next**.
 - **Project Name:** PCIE_Demo
 - **Project Location:** Select an appropriate location (for example, *D:/microsemi_proj*)
 - **Preferred HDL Type:** Verilog or VHDL

Figure 4 • Libero SoC New Project Dialog Box



New Project
Specify Project Details

Project Details

Project Name: PCIE_Demo

Project Location: D:/Microsemi_proj

Description:

Preferred HDL Type: Verilog

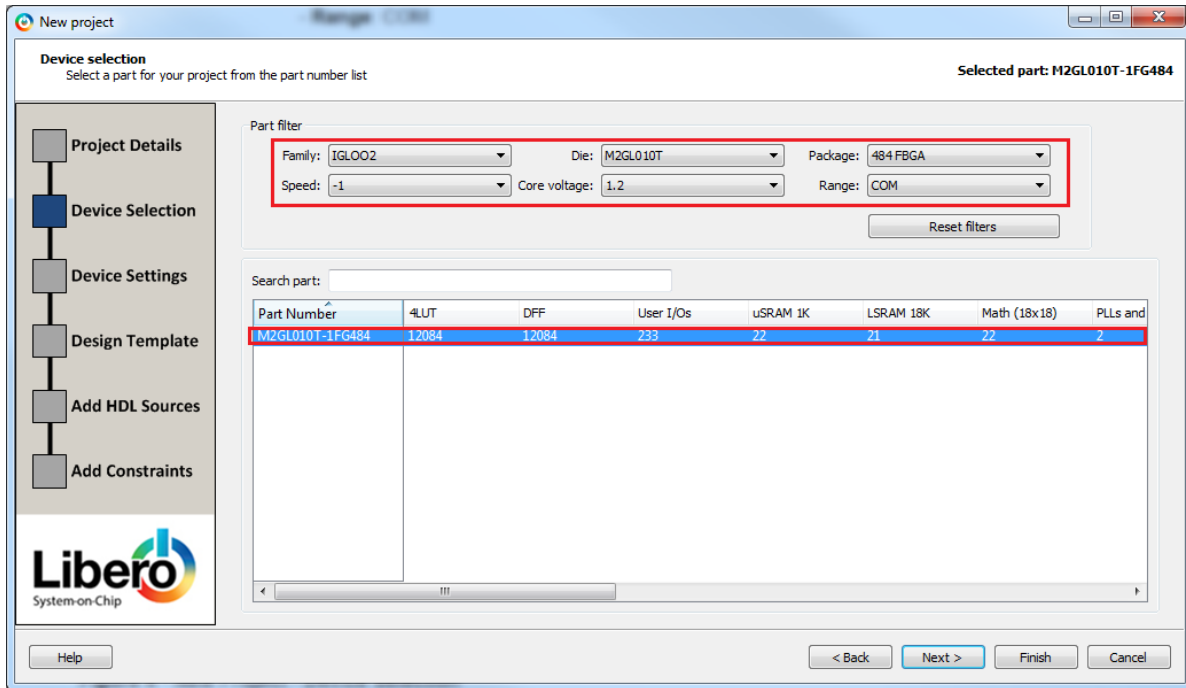
Enable Block Creation

Libero System-on-Chip

Help < Back Next > Finish Cancel

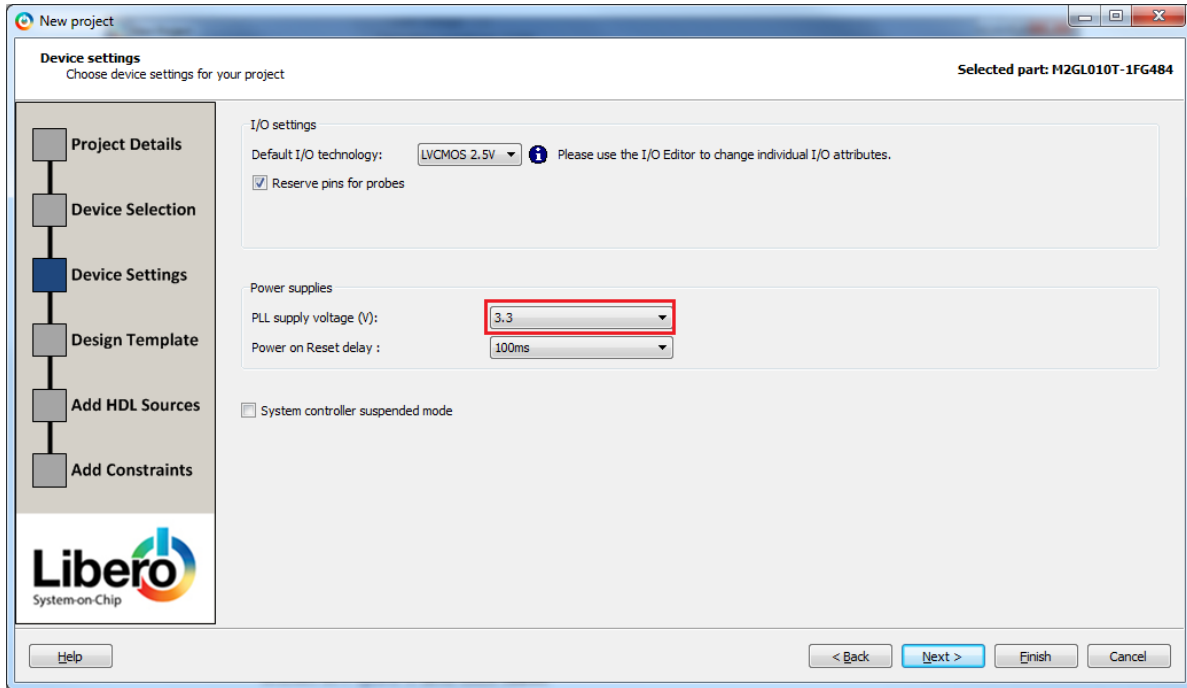
4. Select the following values using the drop-down list for **Device Selection** as shown in the following figure and click **Next**.
 - **Family:** IGLOO2
 - **Die:** M2GL010T
 - **Package:** 484 FBGA
 - **Speed:** -1
 - **Core Voltage:** 1.2
 - **Range:** COM

Figure 5 • New Project—Device Selection



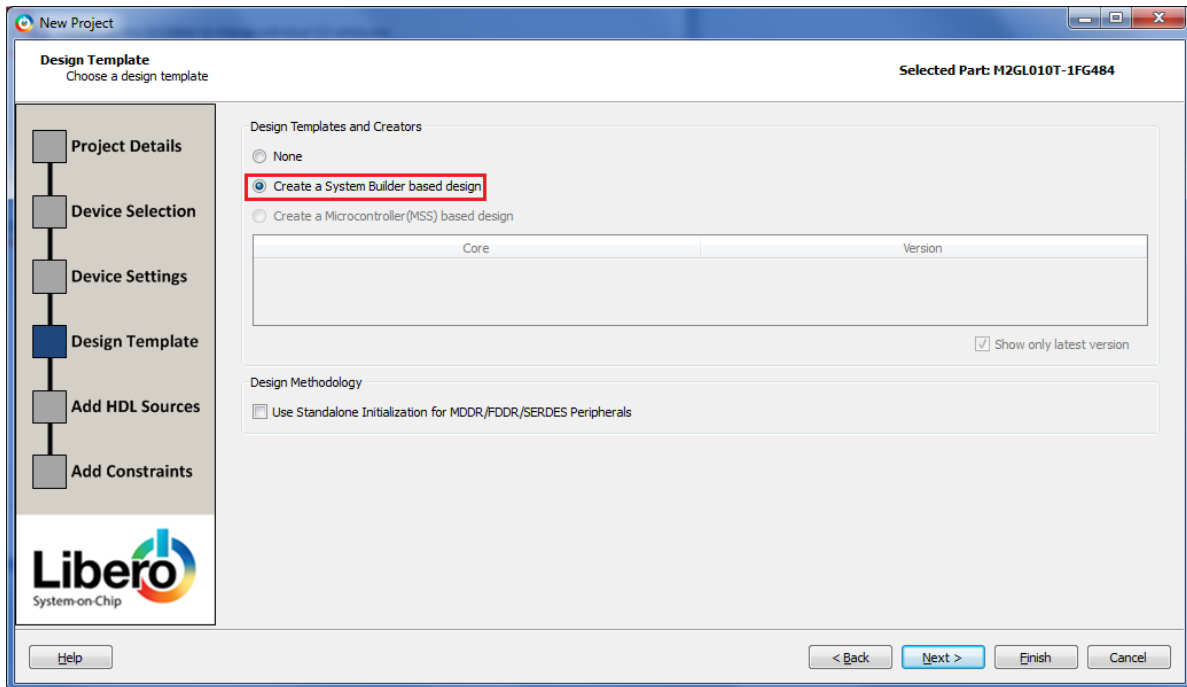
5. Select the **PLL supply voltage (V)** as 3.3 from the drop-down list, as shown in the following figure and click **Next**.

Figure 6 • New Project—Device Settings



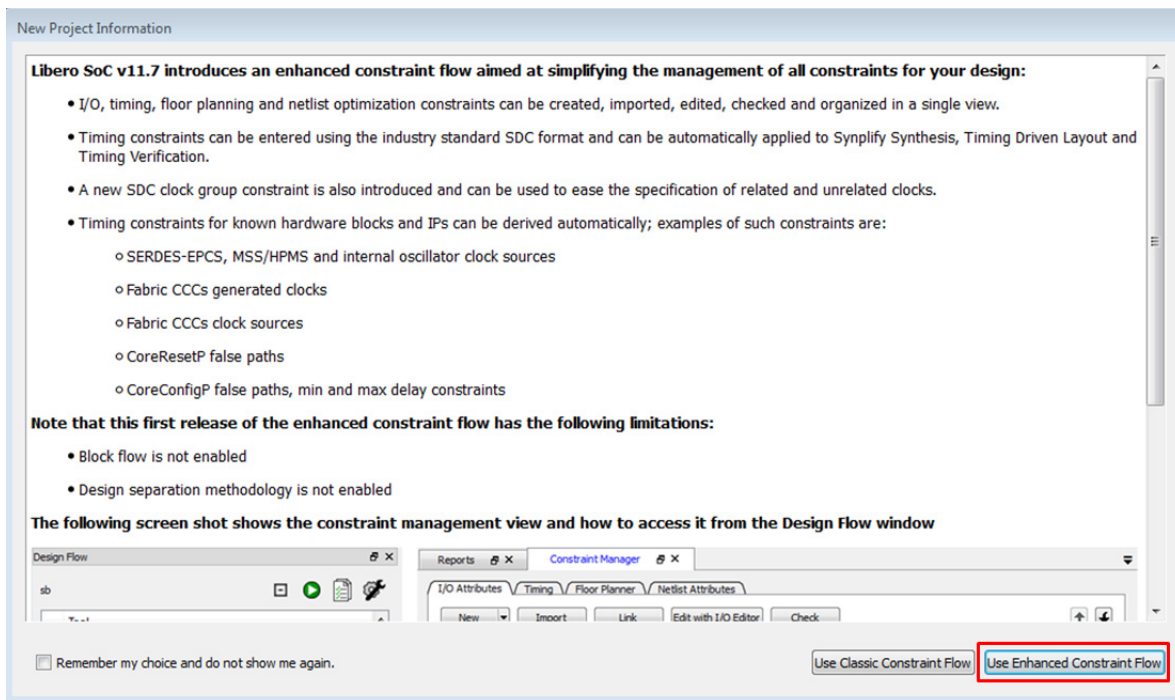
6. Select **Create a System Builder based design** under **Design Templates and Creators**, as shown in the following figure.

Figure 7 • New Project—Device Settings



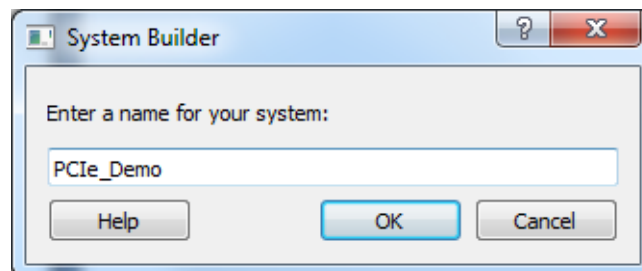
- Click **Finish**. The **New project Information** window is displayed. Select **Use Enhanced Constraint Flow**, as shown in the following figure.

Figure 8 • New Project Information



- Enter **PCIe_Demo** as the name of the system in the **System Builder** dialog box, as shown in the following figure.

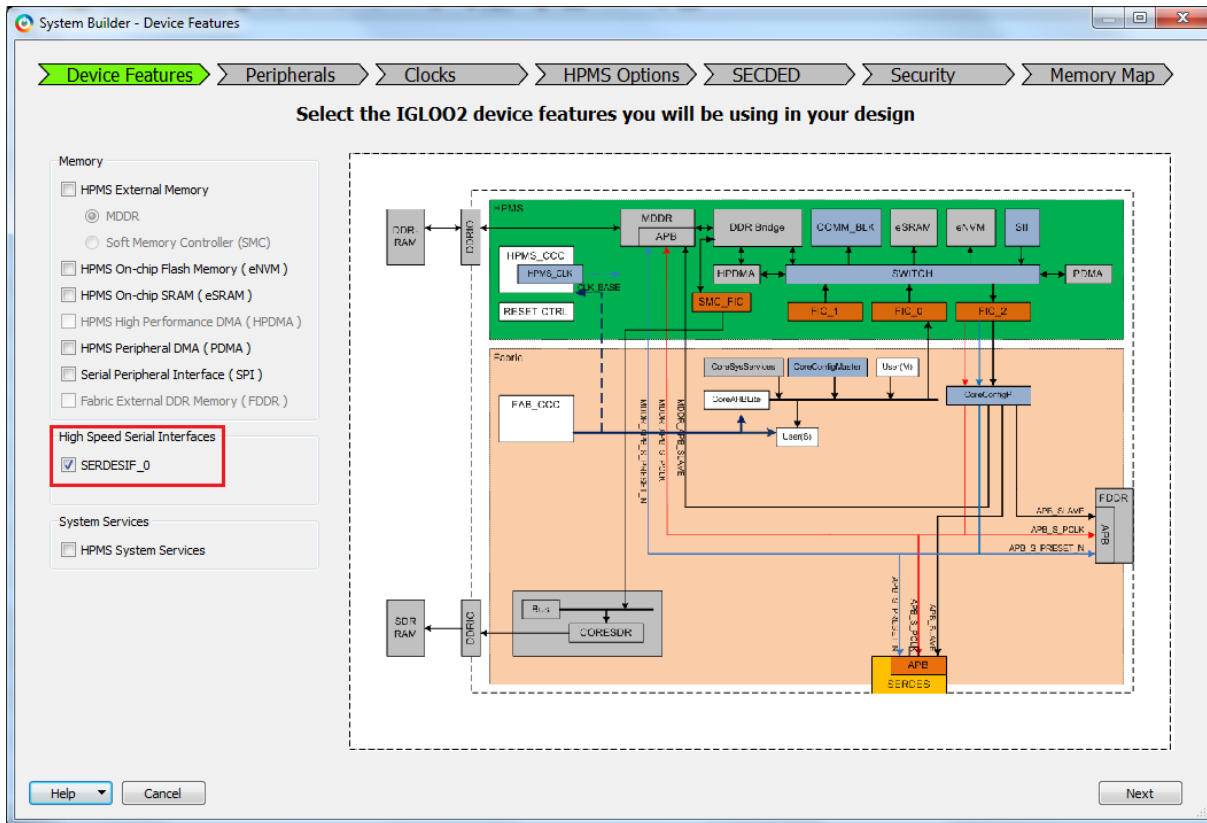
Figure 9 • System Builder Dialog Box



- Click **OK**. The **System Builder - Device Features** window is displayed.

10. In the **System Builder – Device Features** tab, select the **SERDESIF_0** check box under **High Speed Serial Interfaces**, as shown in the following figure.

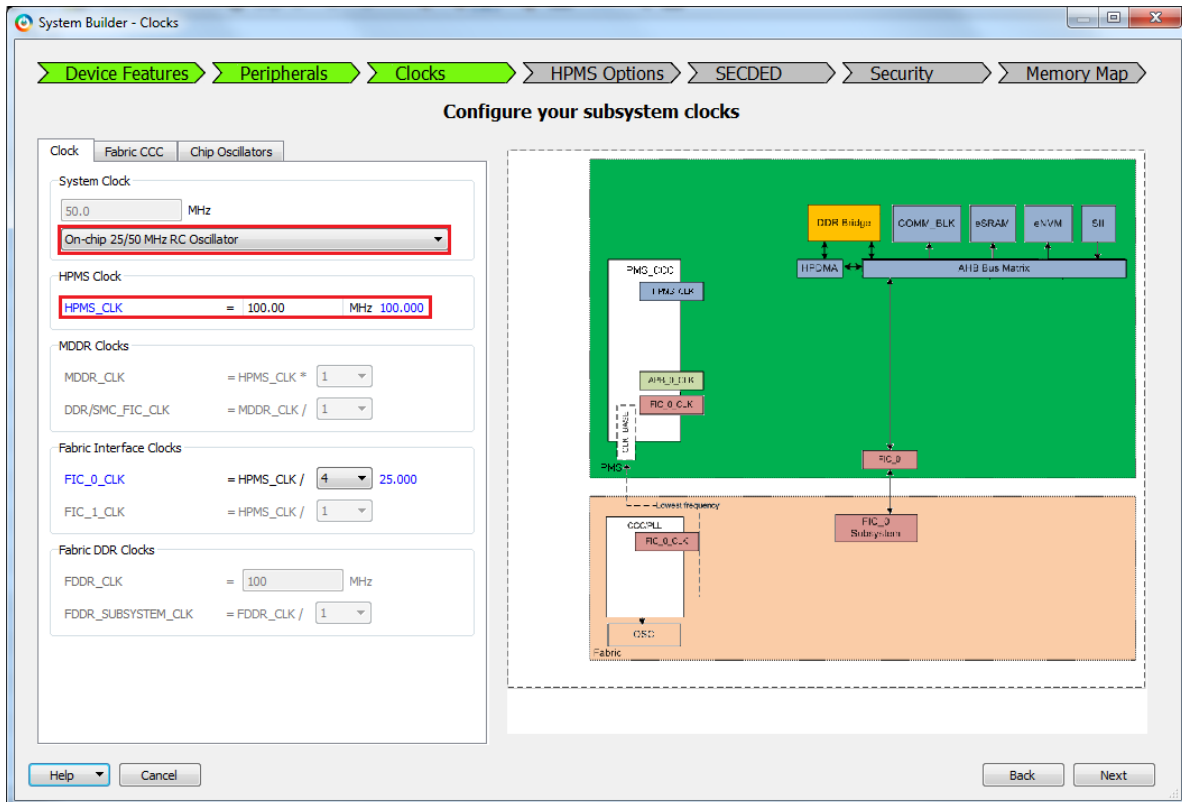
Figure 10 • System Builder—Device Features Tab



11. Click **Next**. The **System Builder – Peripherals** tab is displayed. Do not change the default selections.

- Click **Next**. The **System Builder – Clocks** tab is displayed, as shown in the following figure. Select **System Clock** source as **On-chip 25/50 MHz RC Oscillator** and **HPMS_CLK** as **100 MHz**.

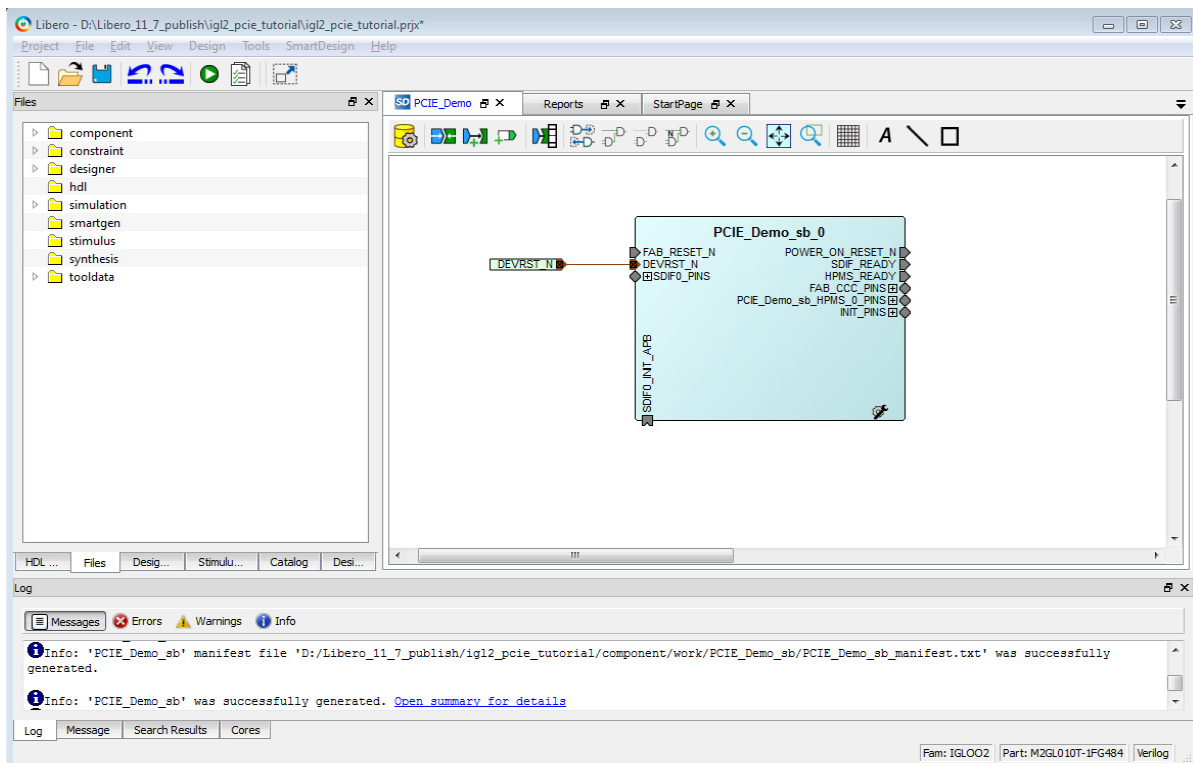
Figure 11 • System Builder—Clocks Tab



- Click **Next**. The **System Builder – HPMS Options** tab is displayed. Do not change the default selections.
- Click **Next**. The **System Builder – SECEDED** tab is displayed. Do not change the default selections.
- Click **Next**. The **System Builder – Security** tab is displayed. Do not change the default selections.
- Click **Next**. The **System Builder – Memory Map** tab is displayed. Do not change the default selections.
- Click **Finish**.

18. 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 the following figure.

Figure 12 • 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.

2.3.2 Instantiating SERDESIF Component in PCIe_Demo 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** SmartDesign, expand the **Peripherals** category in the Libero SoC **Catalog**.

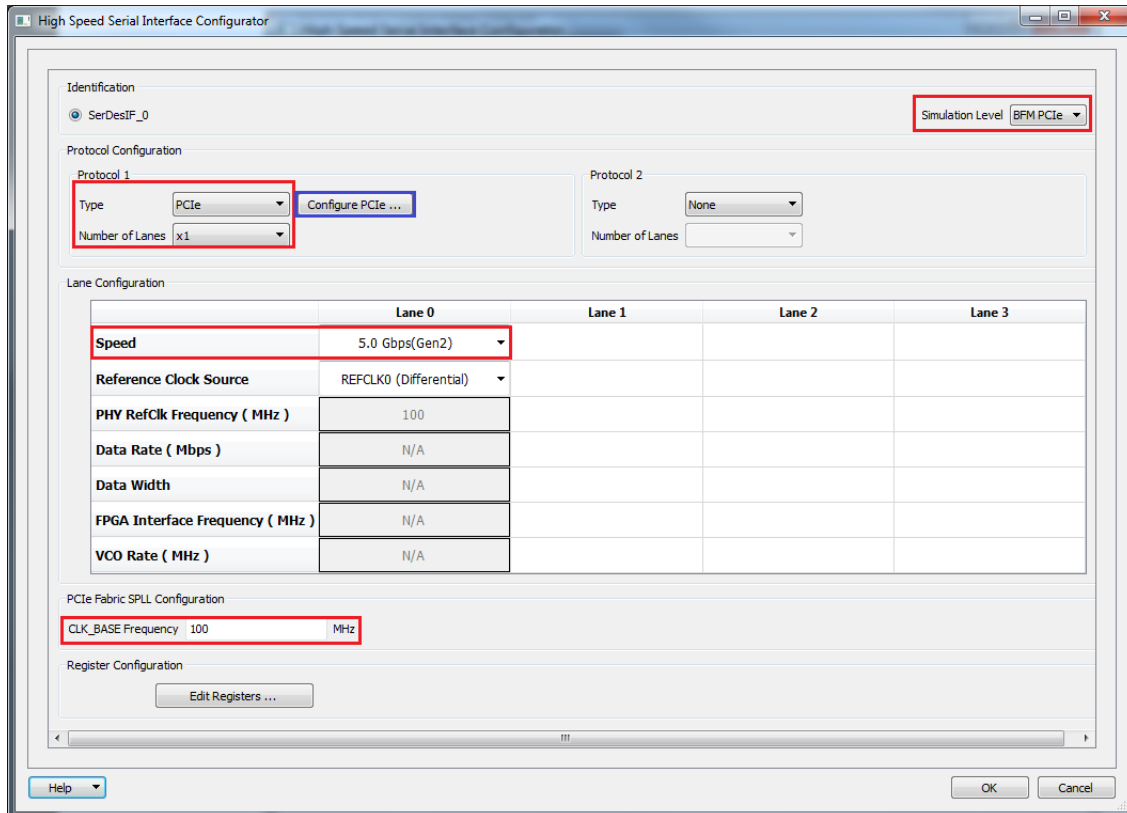
Figure 13 • IP Catalog

CoreTSE_AHB	2.1.105
CoreTimer	2.0.103
CoreUART	5.5.101
CoreUARTapb	5.5.101
CoreWatchdog	1.1.101
High Speed Serial Interface	1.2.210
Processors	
SC/Tamper	
Solutions-MotorControl	
Tamper	

1. Drag the **High Speed Serial Interface** to the **PCIe_Demo SmartDesign** canvas. If the component appears shadowed in the **Vault**, right-click the name and select **Download**.

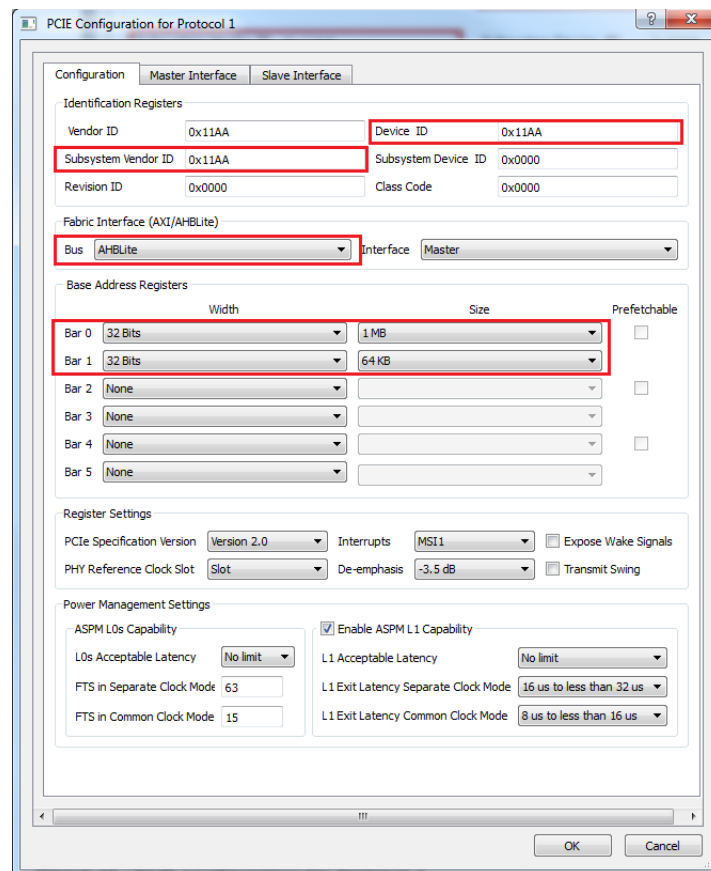
2. Double-click the **SERDES_IF_0** component in the SmartDesign canvas to open the **SERDES** configurator. Configure the SERDES with the following settings, as shown in the following figure:
 - Identification
 - Simulation Level: BFM PCIe
 - Protocol Configuration
 - Protocol1: Type: PCIe
 - Protocol1: Number of Lanes: x1
 - **Lane Configuration**
 - Speed: Lane0: 5.0 Gbps (Gen2)
 - PCIe Fabric SPLL Configuration
 - CLK_BASE Frequency (MHz): 100

Figure 14 • High Speed Serial Interface Configurator Window



3. Click **Configure PCIe** to configure the following settings as shown in the following figure.
 - 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 15 • PCIe Configuration for Protocol 1

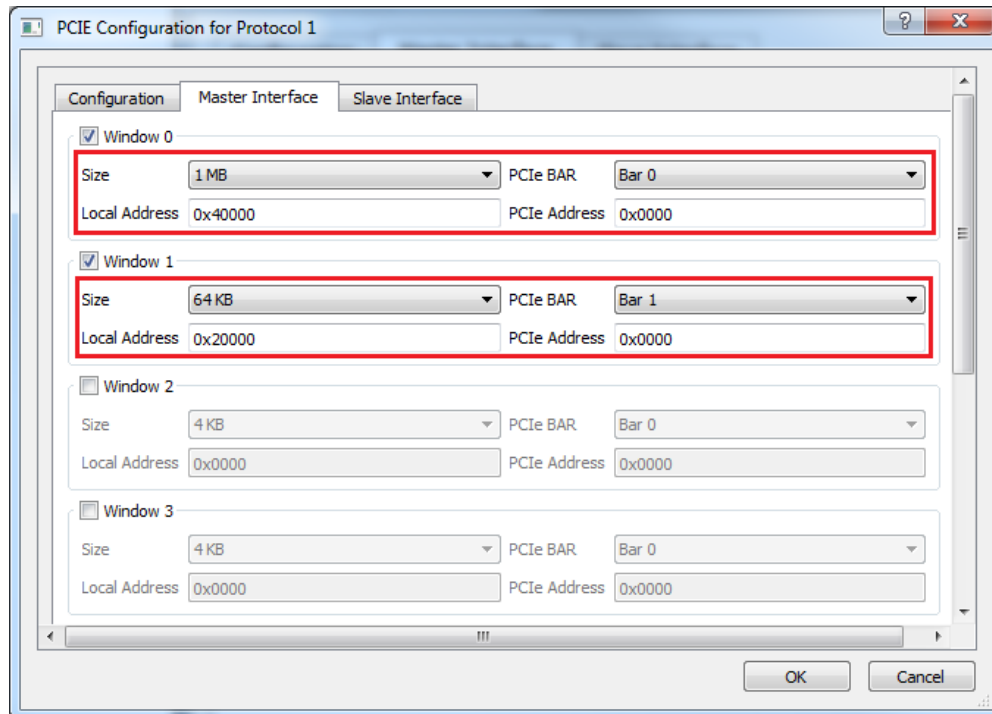


4. Click **Master Interface** tab to configure the PCIe master windows. The PCIe AXI master windows are used to translate the PCIe address domain to the local device address domain. In this tutorial, the PCIe AXI master windows are used to translate the address of BAR0 and BAR1 to CoreGPIO address and CoreAHBLSRAM address.
 - Select Window 0 and configure the following settings:
 - **Size:** Select as 1 MB 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 64 KB 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 about PCIe address translation, see the “Address Translation on the AXI Master Interface” section of the [UG0447: SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide](#).

The following figure shows the **Master Interface Configuration** dialog box.

Figure 16 • Master Interface Configuration Dialog Box



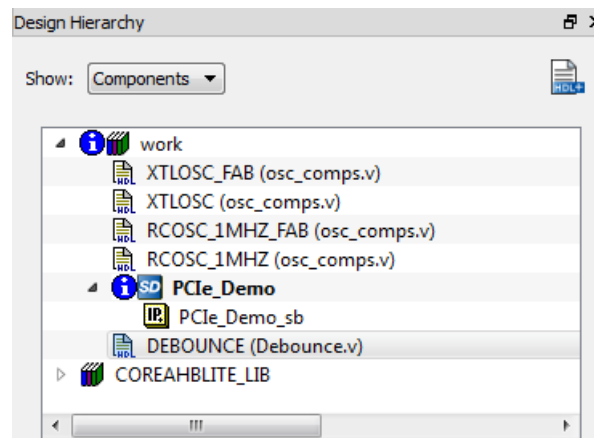
5. Click **OK** to close the PCIe Configuration for protocol 1 dialog box.
6. Click **OK** to save and close the **High Speed Serial Interface Configurator** window.

2.3.3 Instantiating Debounce Logic in PCIe_Demo 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. Click **File > Import > HDL Source files** to add the Debounce logic to the PCIe demo design.
2. Browse to the *M2GL_PCIE_Control_Plane_11p8_sp1_DF*\Source Files file location for *Debounce.v* or *Debounce.vhd* file in the design files folder. The following figure shows the **DEBOUNCE** component in the **Design Hierarchy** window.

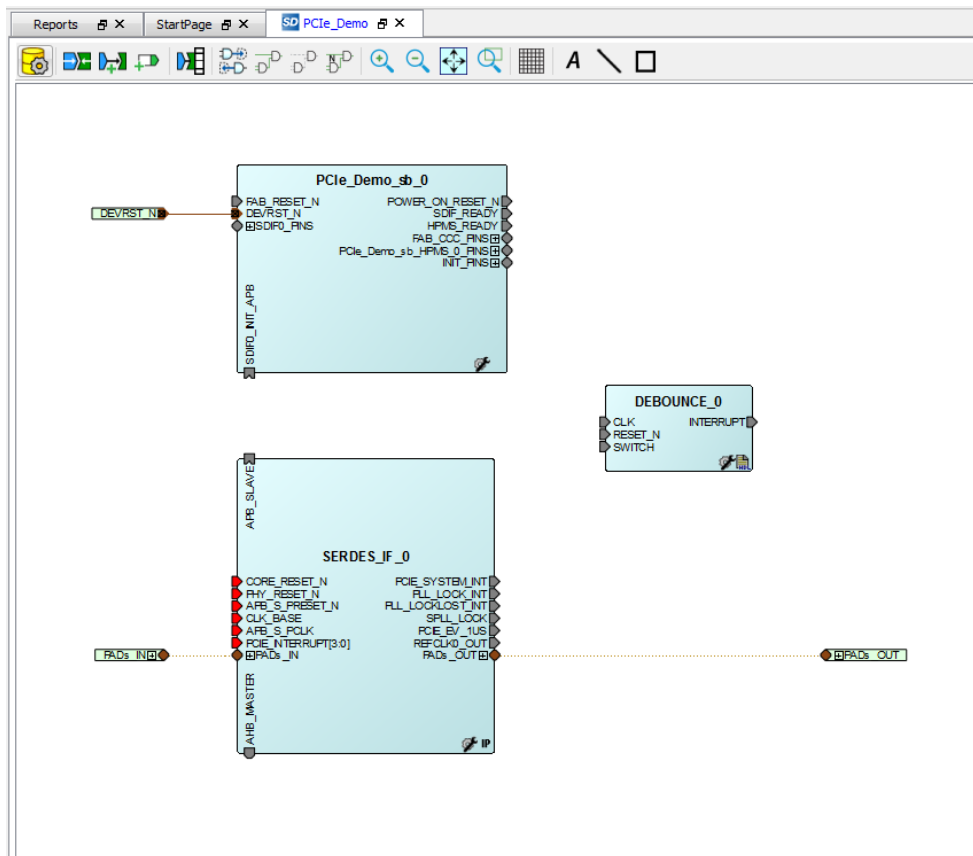
Figure 17 • DEBOUNCE Component in Design Hierarchy Window



3. Drag the **DEBOUNCE** component from the **Design Hierarchy** to the **PCIe_Demo SmartDesign** canvas.

The following figure shows Debounce in PCIe_Demo.

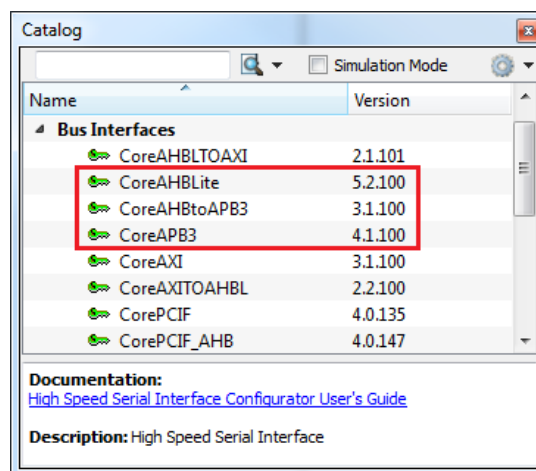
Figure 18 • DEBOUNCE Component in the PCIe_Demo SmartDesign Canvas



2.3.4 Instantiating Bus Interfaces in PCIe_Demo SmartDesign

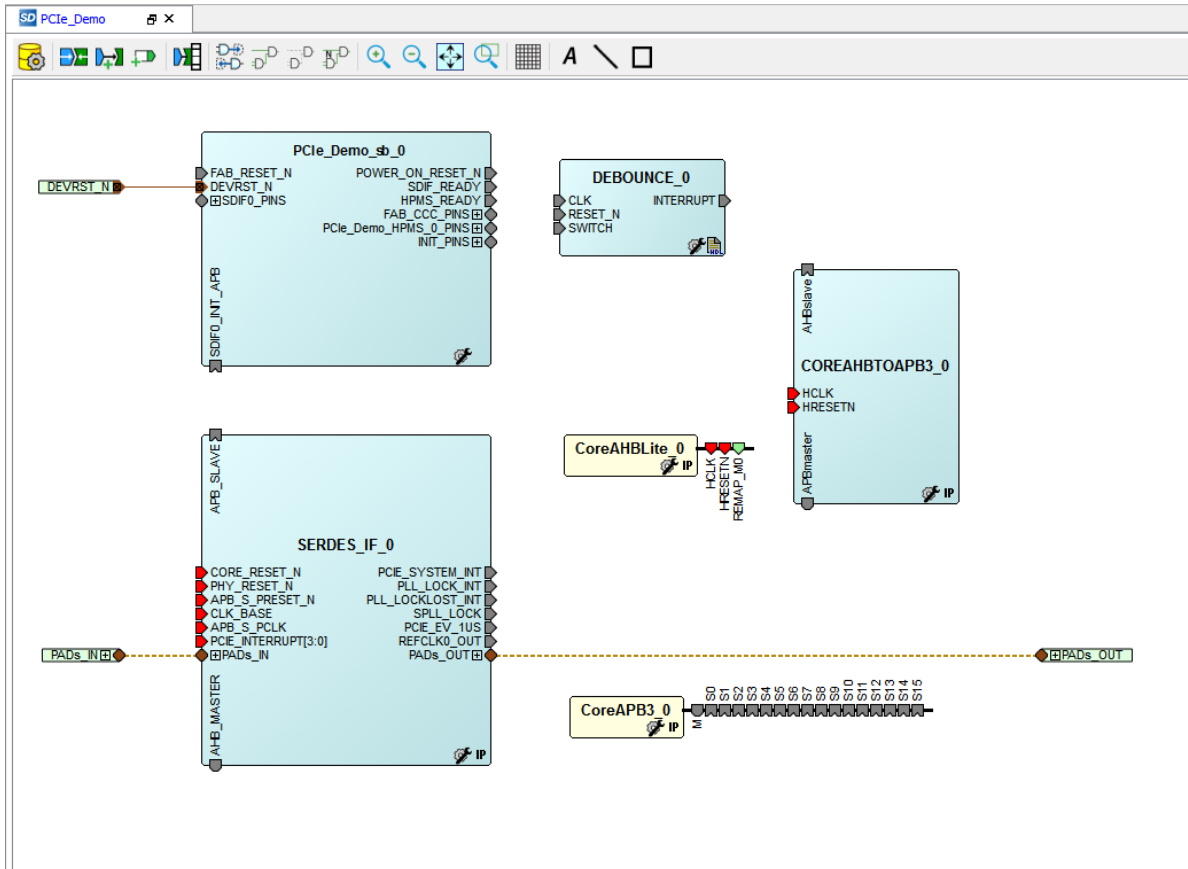
To instantiate the CoreAHBLite, CoreAPB3, and CoreAHBtoAPB3 in the PCIe_Demo SmartDesign, expand the **Bus Interfaces** category in the **Catalog** window. The following figure shows the **Catalog** window.

Figure 19 • IP Catalog



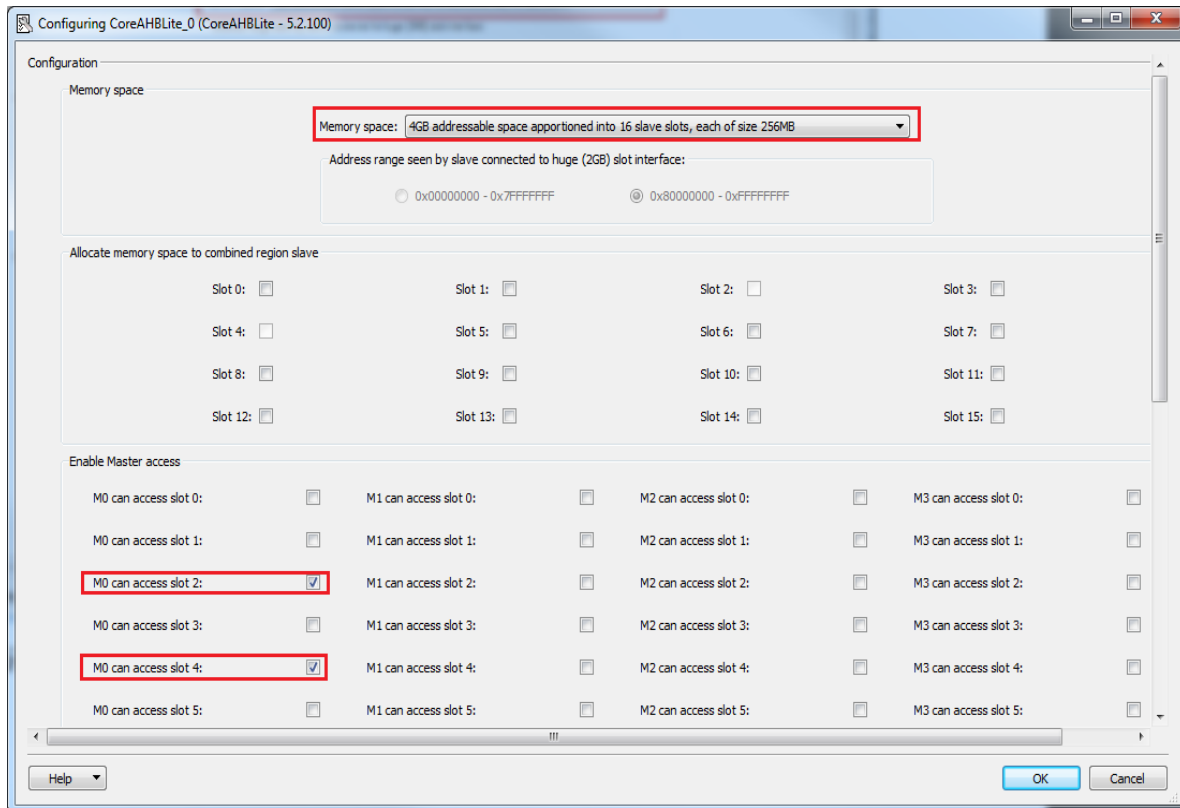
1. Drag CoreAHBLite, CoreAHBtoAPB3, and CoreAPB3 bus interfaces into the PCIe_Demo SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download. The following figure shows the Libero top-level design with bus interfaces.

Figure 20 • CoreAHBLite, CoreAHBtoAPB3, and CoreAPB3 Bus Interfaces in PCIe_Demo SmartDesign Canvas



2. Double-click **CoreAHBLite_0** to configure it. The following figure shows the **Configuring CoreAHBLite_0** window.

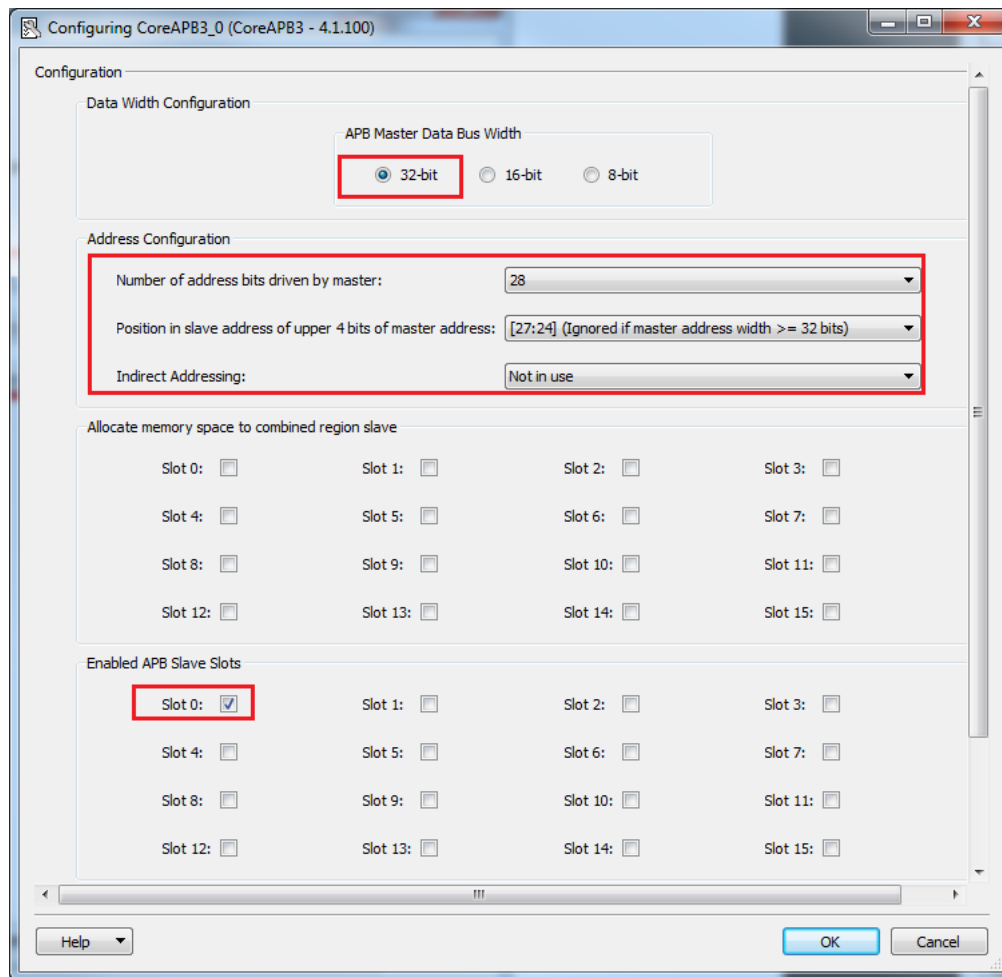
Figure 21 • Configuring CoreAHBLite_0



3. Configuring CoreAHBLite_0
4. 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.
5. Click **OK** to save and close the **Configuring CoreAHBLite_0** window.

6. Double-click **CoreAPB3** to configure it. The following figure shows the **Configuring CoreAPB3_0** window.

Figure 22 • Configuring CoreAPB3_0

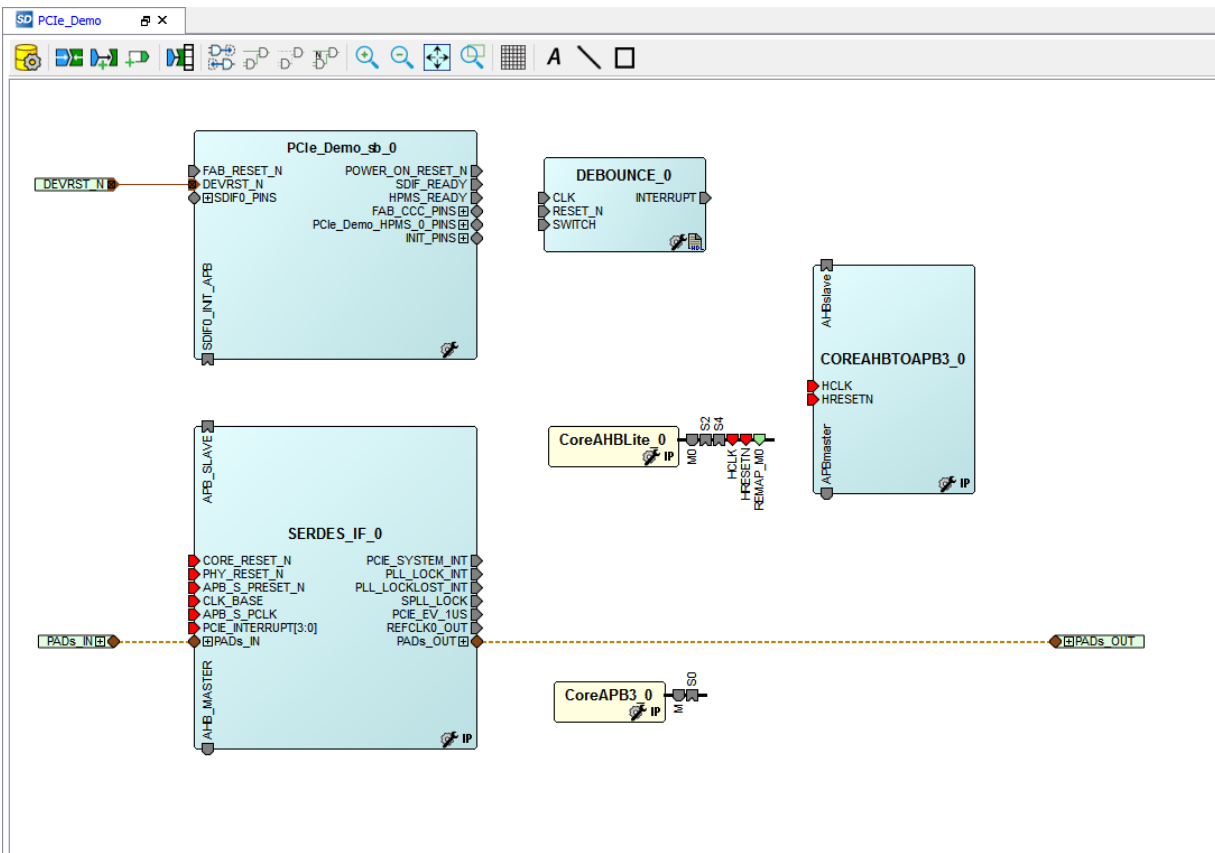


7. Configure **CoreAPB_0** with the below settings:
- Under **Data Width Configuration**, select **APB Master Data Bus Width** as **32-bit**.
 - Under **Address Configuration**, select **Number of address bits driven by master** as **28** and **Position in slave address of upper 4 bits of master address** as **[27:24]**(Ignored if master address width \geq 32 bits) using the drop-down list.
 - Select **Enabled APB Slave Slots** as **Slot 0**.

Click **OK** to save and close the **Configuring CoreAPB3_0** window.

The following figure shows the PCIe_Demo in SmartDesign after configuring CoreAHLite and CoreAPB3 bus interfaces.

Figure 23 • CoreAHLite and CoreAPB3 Bus Interfaces in PCIe_Demo SmartDesign Canvas After Configuration

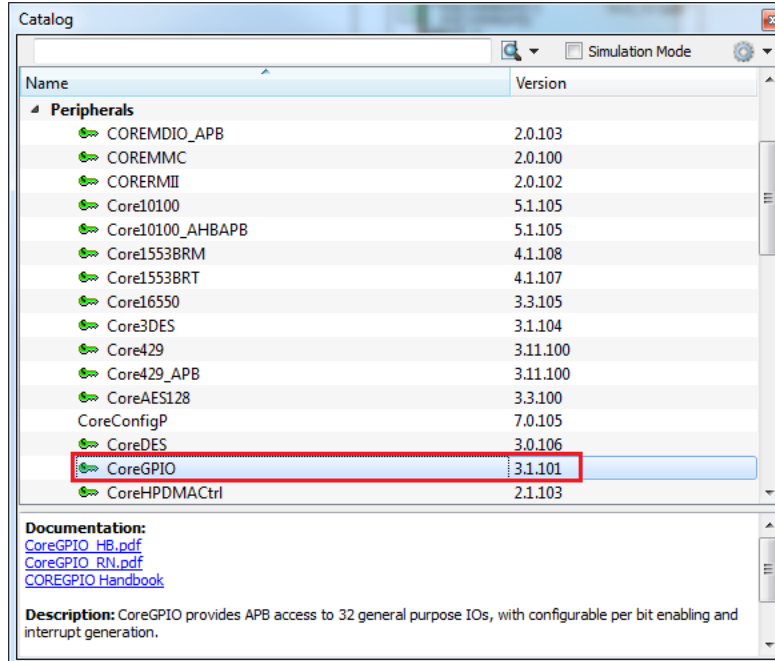


2.3.5 Instantiating CoreGPIO in PCIe_Demo SmartDesign

The following steps describe how to instantiate CoreGPIO in the PCIe_Demo SmartDesign:

1. Expand the **Peripherals** category in the **Catalog** as shown in the following figure.

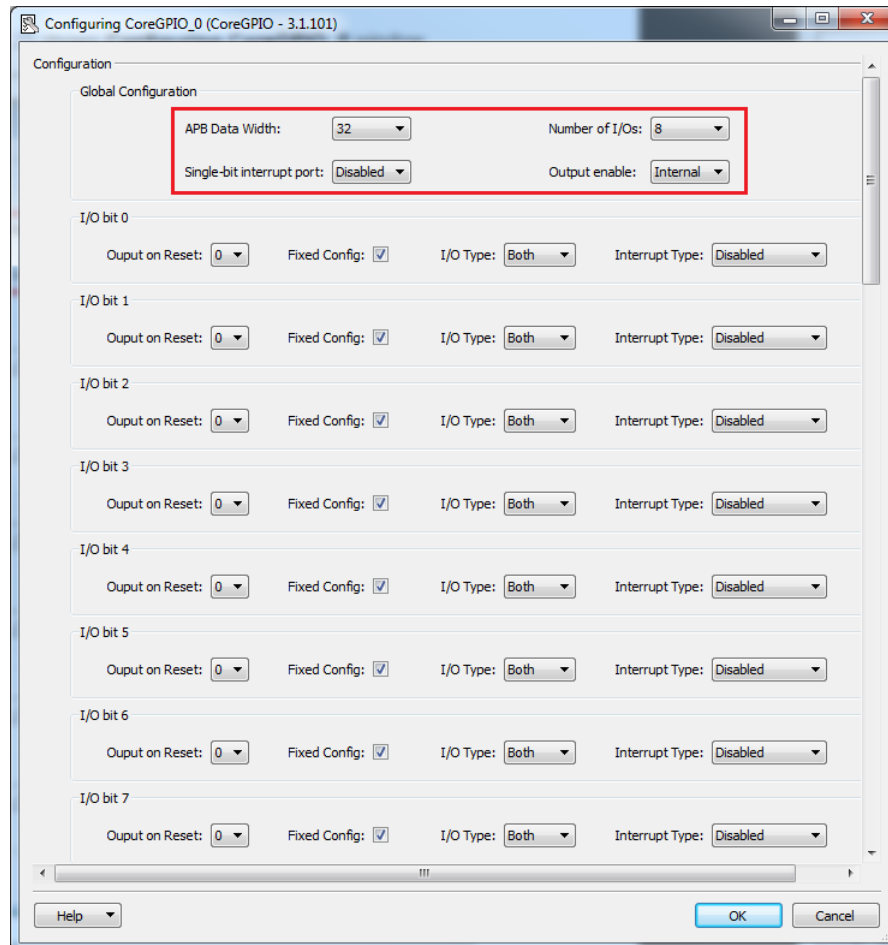
Figure 24 • IP Catalog



2. Drag **CoreGPIO** to the PCIe_Demo SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.

3. Double-click **CoreGPIO** to configure it. The following figure shows **Configuring CoreGPIO_0** window.

Figure 25 • Configuring CoreGPIO_0



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

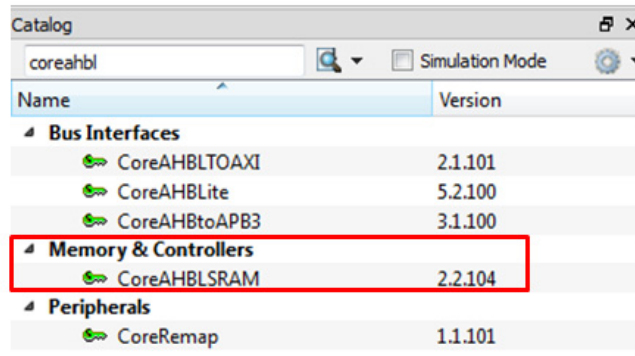
CoreGPIO is configured with 8 outputs connected to LEDs and with four inputs connected to DIP switches.

2.3.6 Instantiating CoreAHBLSRAM in PCIe_Demo SmartDesign

The following steps describe how to instantiate CoreAHBLSRAM in the PCIe_Demo SmartDesign:

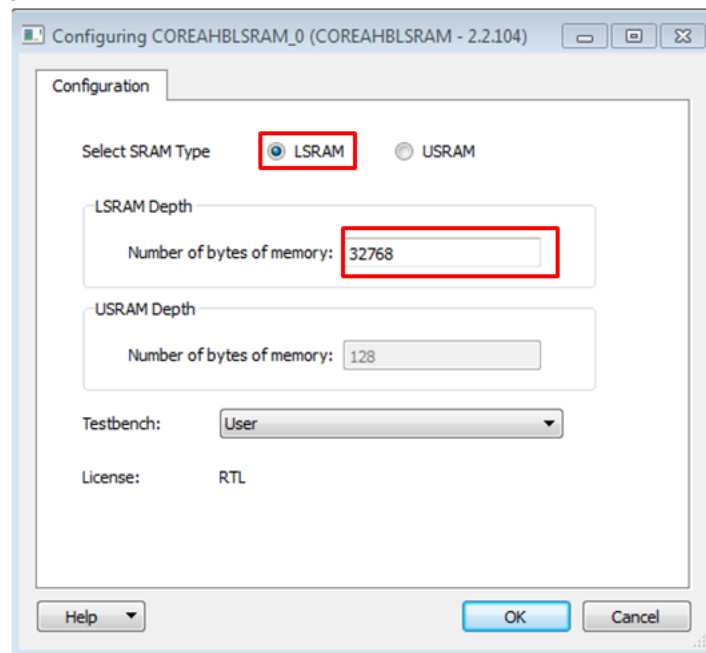
1. Expand the **Memory & Controllers** category in the **Catalog** as shown in the following figure.

Figure 26 • IP Catalog



2. Drag **CoreAHBLSRAM** to the **PCIe_Demo** SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.
3. Double-click COREAHBLSRAM_0 to configure it. The following figure shows the **Configuring COREAHBLSRAM_0** window.

Figure 27 • Configuring COREAHBLSRAM_0



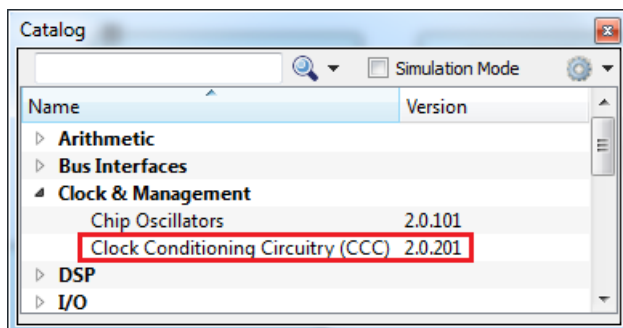
4. Under **Configuration**, select **AHB Data Width** and **AHB Address Width** as **32**.
5. Under **Select SRAM Type**, click **LSRAM**.
6. Under **LSRAM Depth**, enter the **Number of bytes of memory** as **32768**.
7. Click **OK** to save and close the **Configuring COREAHBLSRAM_0** window.

2.3.7 Instantiating CCC in PCIe_Demo SmartDesign

CCC supplies the clock for components instantiated in the fabric. The following steps describe how to instantiate CCC in the PCIe_Demo SmartDesign:

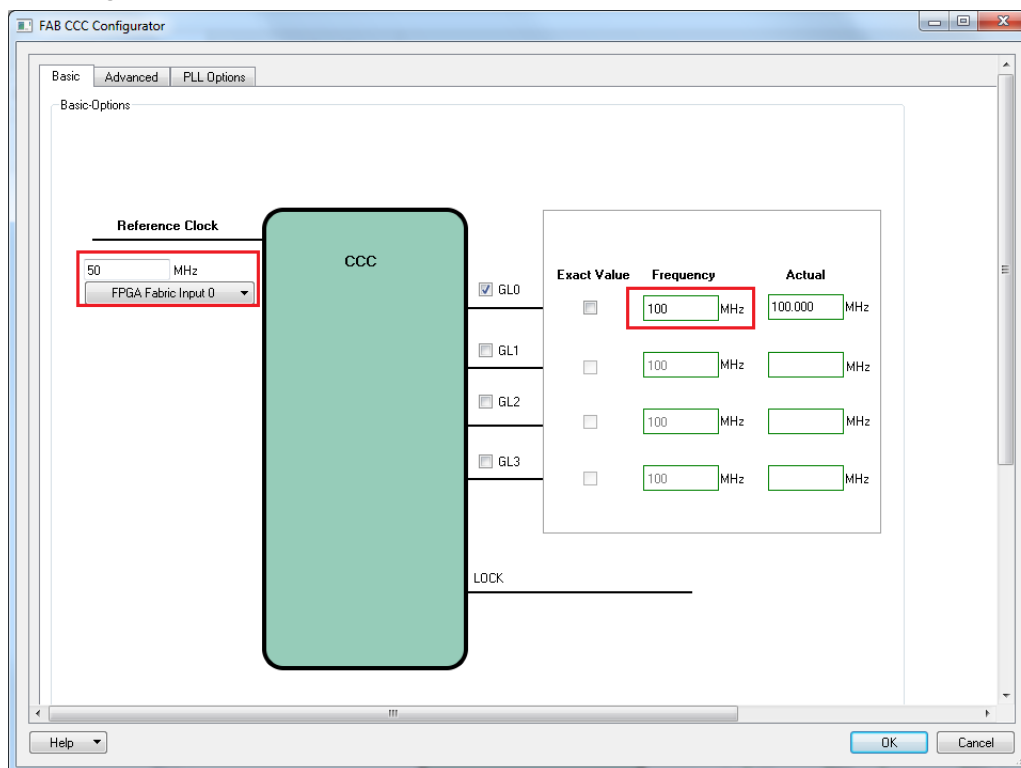
1. Expand the **Clock & Management** category in the Libero SoC **Catalog**. The following figure shows the Libero Catalog.

Figure 28 • Catalog



2. Drag clock conditioning circuit (CCC) to the PCIe_Demo SmartDesign canvas. If the component appears shadowed in the **Vault**, right-click the name and select download.
3. Double-click **CCC** to configure it. The following figure shows the **FAB CCC Configurator** window.

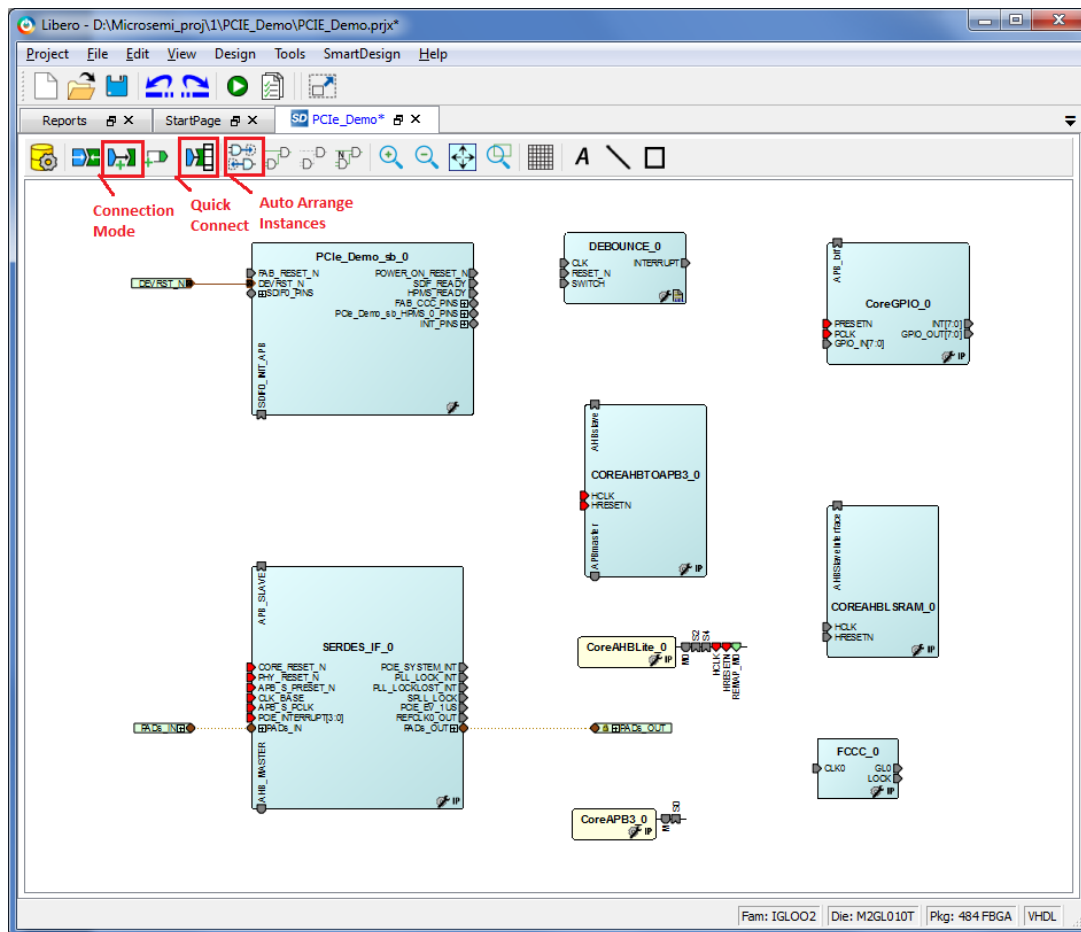
Figure 29 • Configure CCC



4. Select **Reference Clock** as **50 MHz** and **FPGA Fabric Input 0** from the drop-down list.
5. Select **GL0 Frequency** as **100 MHz**.
6. Click **OK** to save and close the **FAB CCC Configurator** window.

The following figure shows PCIe_Demo in SmartDesign after configuring all components.

Figure 30 • PCIe_Demo in SmartDesign



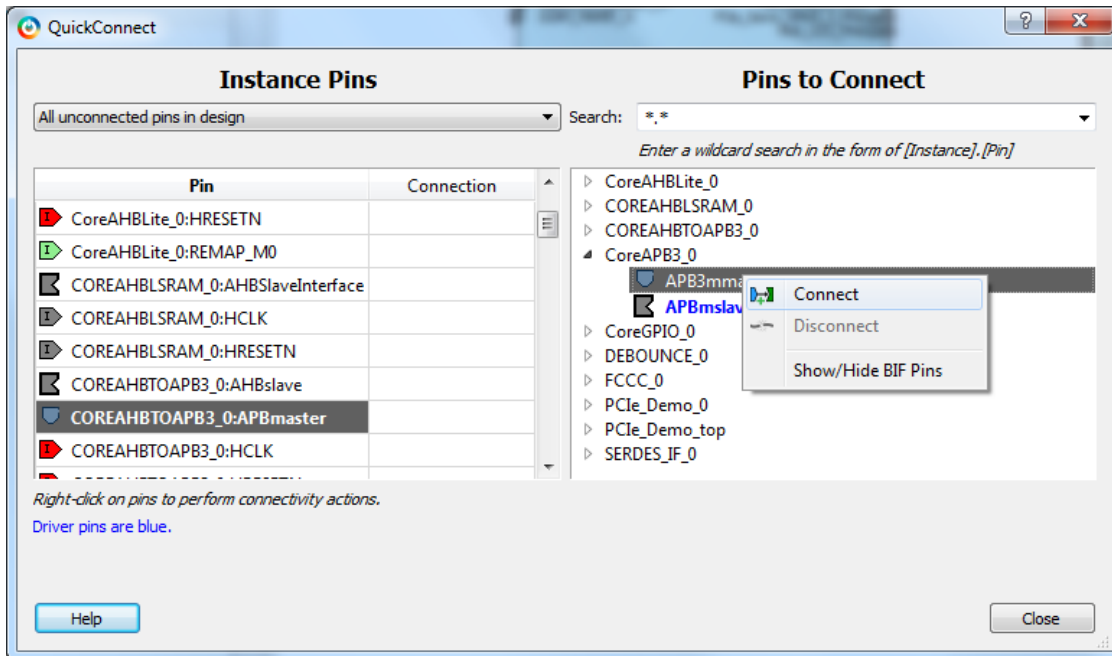
2.3.8 Connecting Components in PCIe_Demo SmartDesign

There are three methods for connecting components in PCIe_Demo 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 30, 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 30, page 25. 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 the following figure.

Figure 31 • Quick Connect Window



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

1. Expand **SDIF0_PINS** of PCIe_Demo_sb_0 and make connections as shown in the following table.

Table 2 • SDIF0_PINS

From PCIe_Demo_sb_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_sb_0 and make connections as shown in the following table.

Table 3 • INIT_PINS

From PCIe_Demo_sb_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_sb_0 to all resets as shown in the following table.

Table 4 • HPMS_READY Connections

From PCIe_Demo_sb_0	To
HPMS_READY	HRESETN of CoreAHBLite_0, COREAHBTOAPB3_0, and COREAHBLSRAM_0 PRESETN of CoreGPIO_0

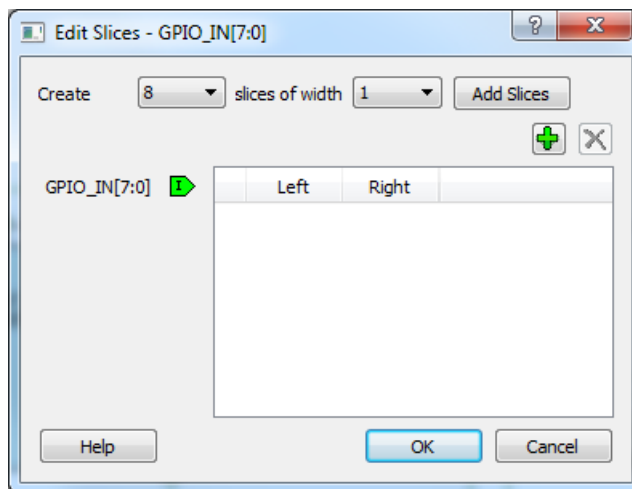
6. Connect **GL0** of **FCCC_0** to all clocks, as shown in the following table.

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

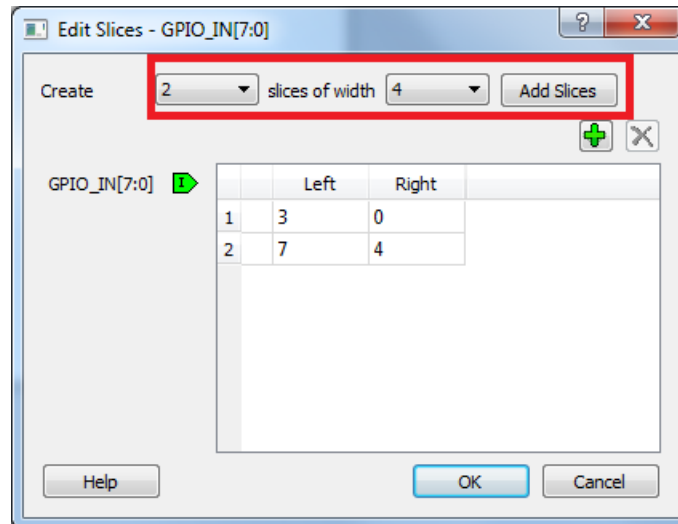
7. Expand **FAB_CCC_PINS** of **PCle_Demo_sb_0**:
 - Right-click the **FAB_CCC_GL0** and select **Mark Unused**.
 - Right-click the **FAB_CCC_GL3** and select **Mark Unused**.
 - Right-click the **FAB_CCC_LOCK** and select **Mark Unused**.
8. Connect **POWER_ON_RESET_N** of **PCle_Demo_sb_0** to **RESET_N** of **DEBOUNCE_0**.
9. Right-click the **SDIF_READY** of **PCle_Demo_sb_0** and select **Mark Unused**.
10. Right-click the **FAB_RESET_N** of **PCle_Demo_sb_0** and select **Tie high**.
11. Expand **PCI_Demo_HPMS_0_PINS**.
 - Right-click the **COMM_BLK_INT** of **PCle_Demo_sb_0** and select **Mark Unused**.
 - Right-click the **HPMS_INT_M2F[15:0]** of **PCle_Demo_sb_0** and select **Mark Unused**.
12. Connect **SDIF0_INIT_APB** of **PCle_Demo_sb_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**. The following figure displays the **Edit Slice** window.

Figure 32 • Edit Slices



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

Figure 33 • Edit Slices



25. Click **OK**.

26. Expand **GPIO_IN [7:0]**, right-click the **GPIO_IN [7:4]** and select **Tie low**.

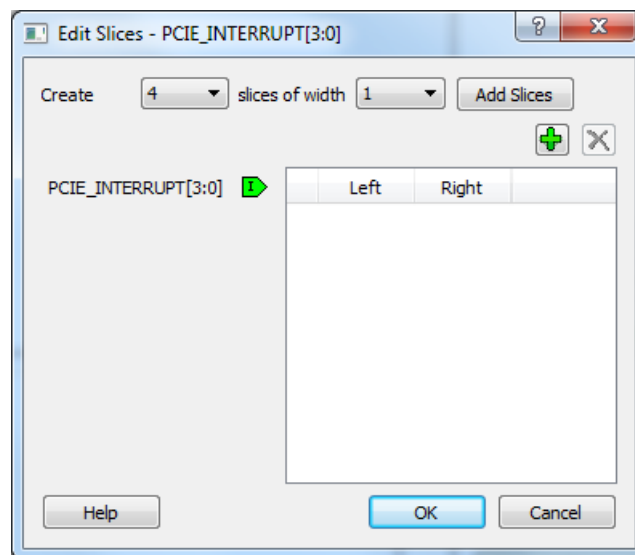
27. Right-click the **GPIO_IN[3:0]** and select **Promote to top level**.

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

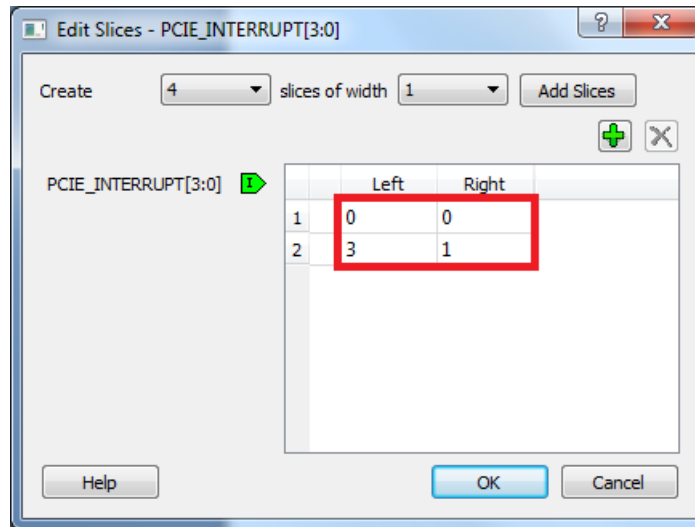
29. The PCIe supports four interrupts. This design uses only one interrupt out of four by connecting the unused interrupts to logic 0. To connect the unused interrupt pins to logic 0, split the interrupt pins to two groups. To do that, right-click the **PCIE_INTERRUPT[3:0]** of **SERDES_IF_0** and select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

Figure 34 • Edit Slices



30. Click + 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 the following figure.

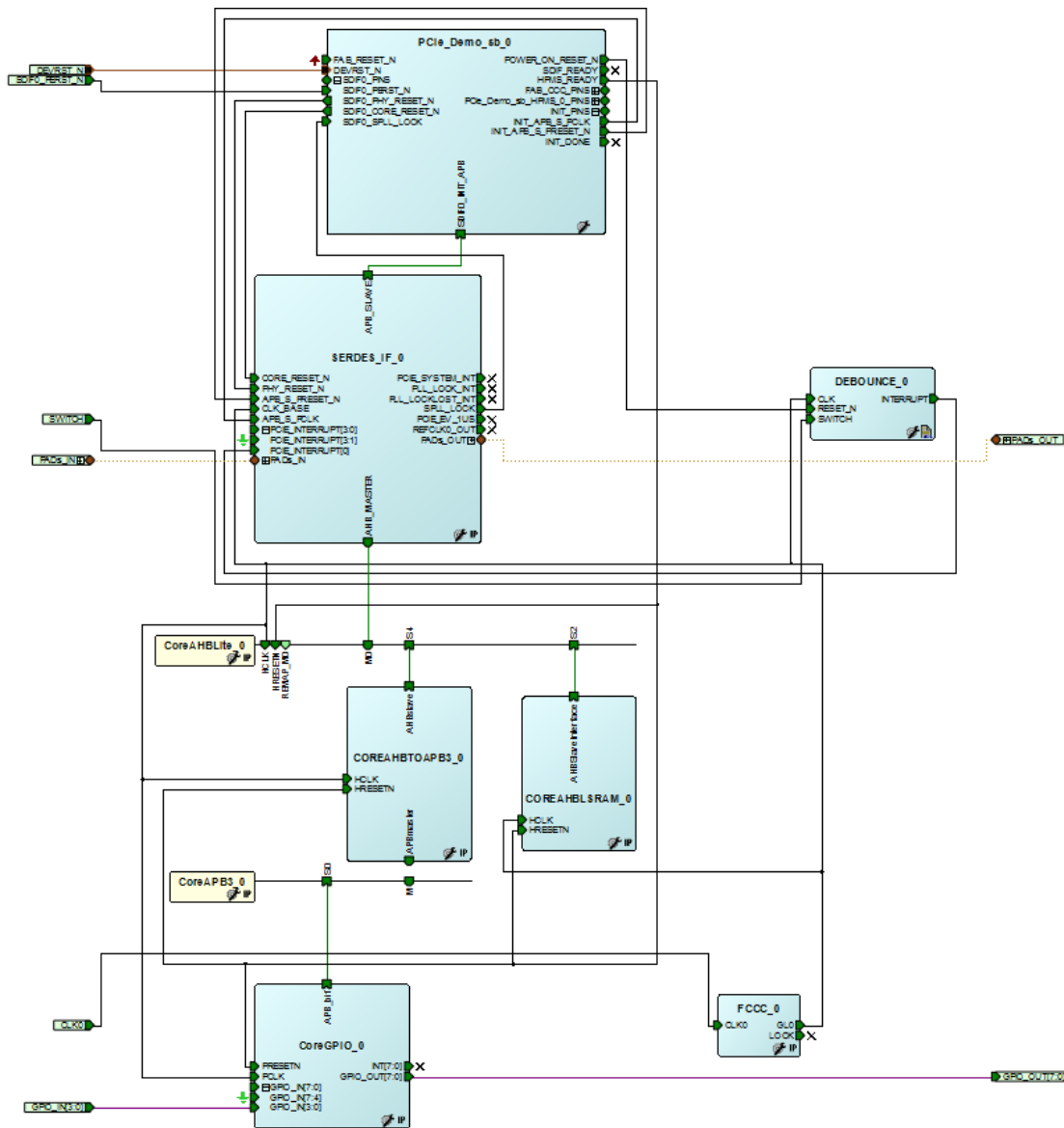
Figure 35 • Edit Slices



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**.

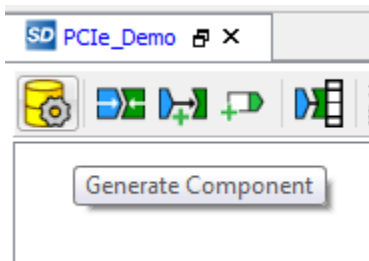
33. Click **Auto arrange instances** to arrange the instances and click **File > Save**. The PCIe_Demo is displayed as shown in the following figure.

Figure 36 • PCIe_Demo Design



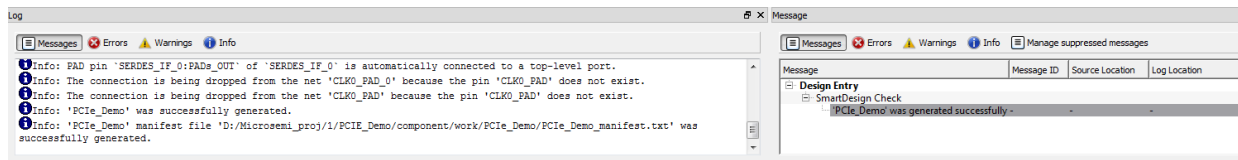
34. Open the **PCIe_Demo** tab and click **Generate Component**, as shown in the following figure.

Figure 37 • Generate Component



35. The message PCIe_Demo was generated is displayed in the Libero SoC Log window, if the design is generated without any errors. The Log window is displayed as shown in the following figure on successful component generation.

Figure 38 • Log Window

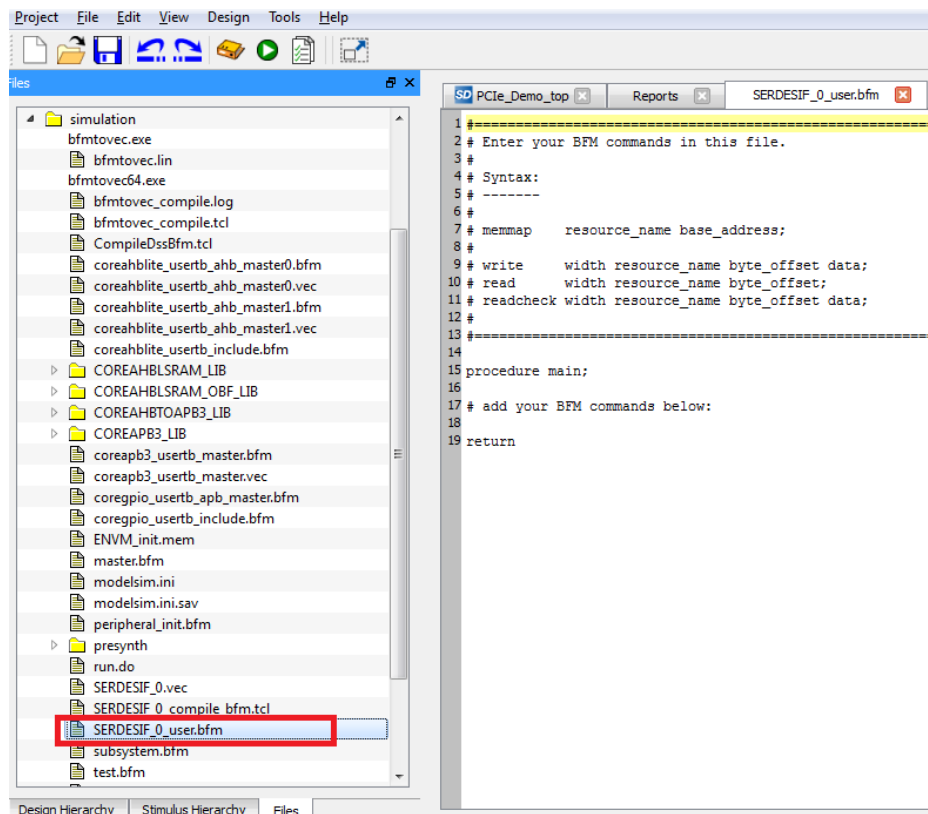


2.4 Step 2: Developing the Simulation Stimulus

During the design process, SERDESIF is 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 as the IGLOO2 PCIe interface is a fully hardened and verified interface. This section describes how to modify the BFM script (`user.bfm`) file that is 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 the following figure.

Figure 39 • 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;

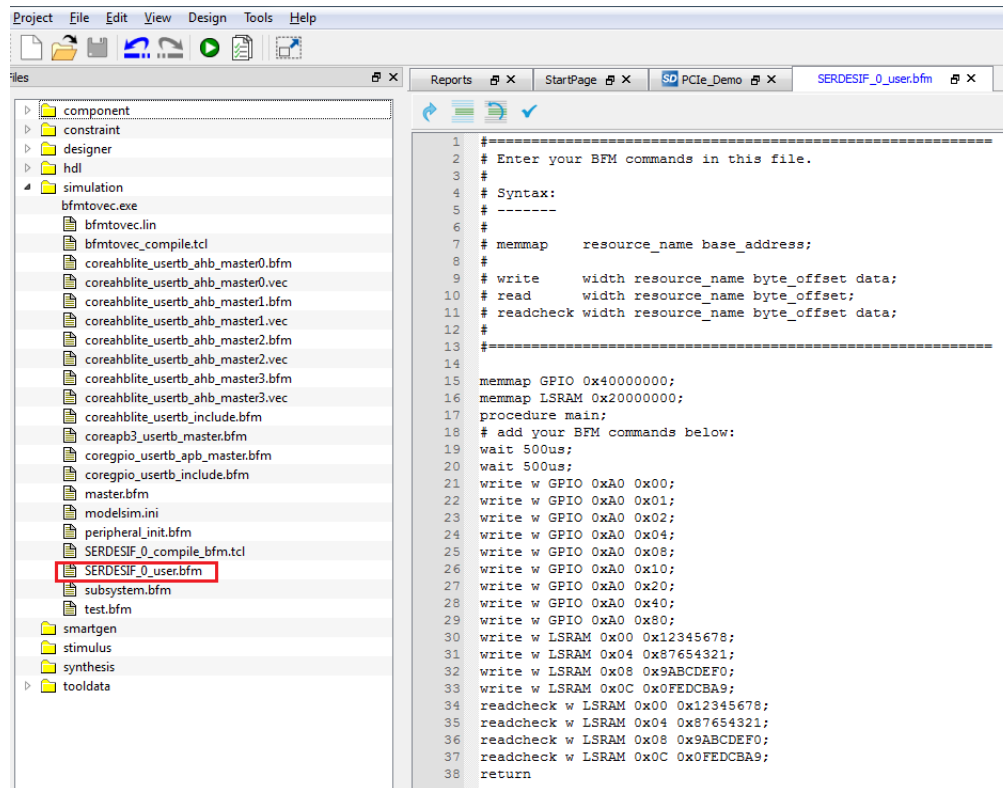
write w LSRAM 0x00 0x12345678;
write w LSRAM 0x04 0x87654321;
write w LSRAM 0x08 0x9ABCDEF0;
write w LSRAM 0x0C 0x0FEDCBA9;
readcheck w LSRAM 0x00 0x12345678;
readcheck w LSRAM 0x04 0x87654321;
readcheck w LSRAM 0x08 0x9ABCDEF0;
readcheck w LSRAM 0x0C 0x0FEDCBA9;
return
```

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

The modified BFM file appears similar to the file shown in the following figure.

Figure 40 • Modified SERDES User BFM



Note: If the target device is IGLOO2 090, BFM commands must be added to the SmartDesign generated BFM script (`SERDESIF_0_PCIE_0_user.bfm`).

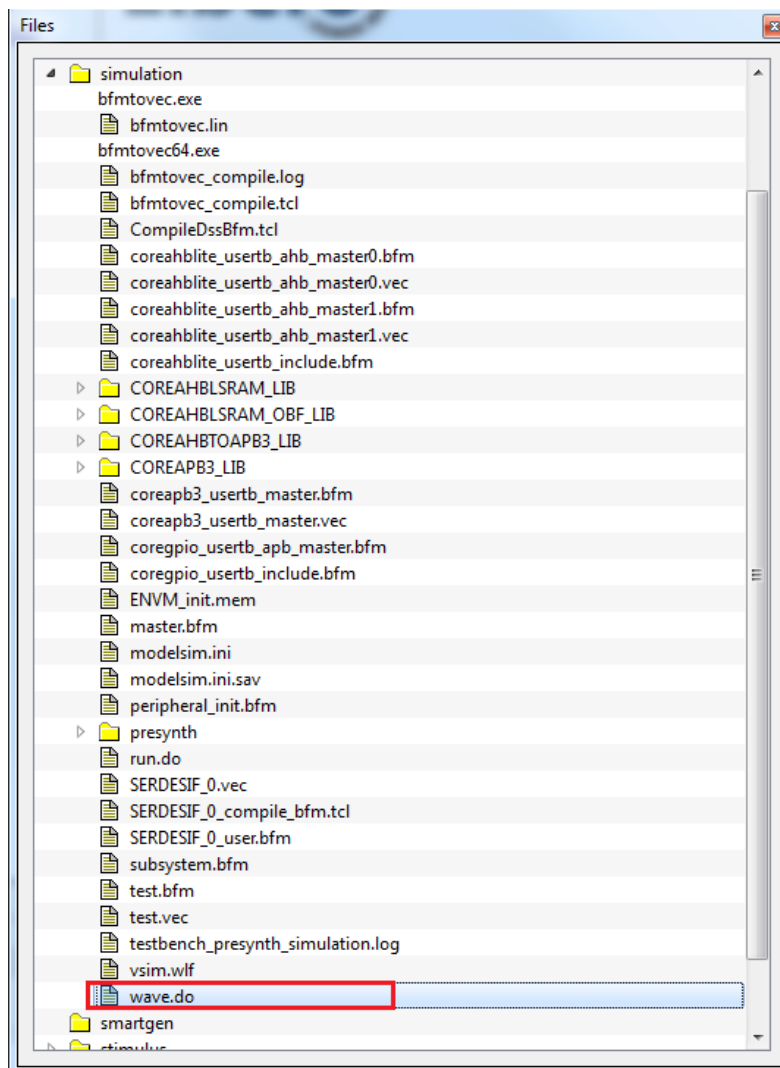
2.5 Step 3: Simulating the Design

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. The following steps describe 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**.

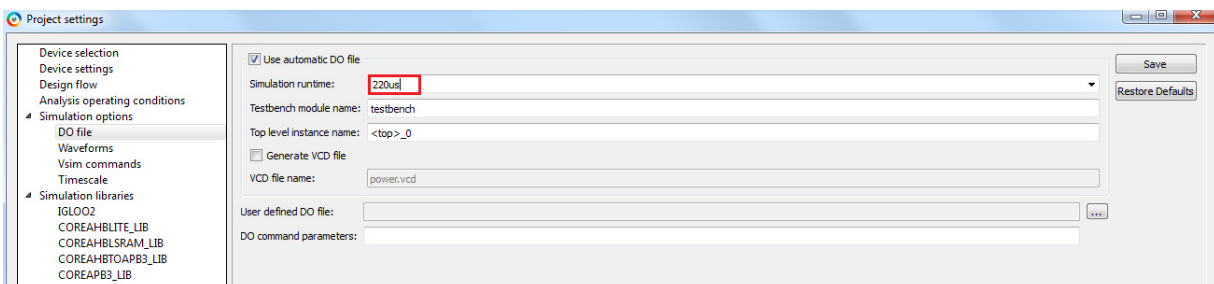
- Browse to the `wave.do` file location in the design files folder: `M2GL_PCIE_Control_Plane_11p8_sp1_DF\Source Files`. The following figure shows the `wave.do` file under simulation folder in the Files window.

Figure 41 • Wave.do file



- Open the Libero SoC project settings (**Project > Project Settings**).
- Select **Do File** under **Simulation Options** in the **Project Settings** window. Change the **Simulation runtime** to **220 μ s**, as shown in the following figure.

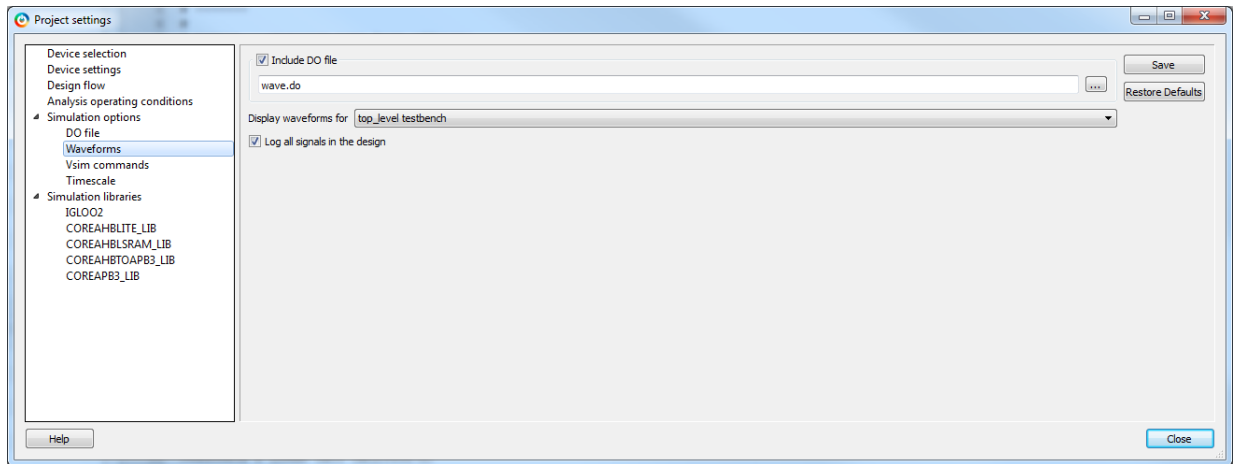
Figure 42 • Project Settings—Do File Simulation Runtime Settings



- Click **Save**.

6. Select **Waveforms** under **Simulation Options** as shown in the following figure.

Figure 43 • Project Settings—Waveform



Select the **Include Do** check box and select the file.
 Select the **Log all signals in the design** check box.

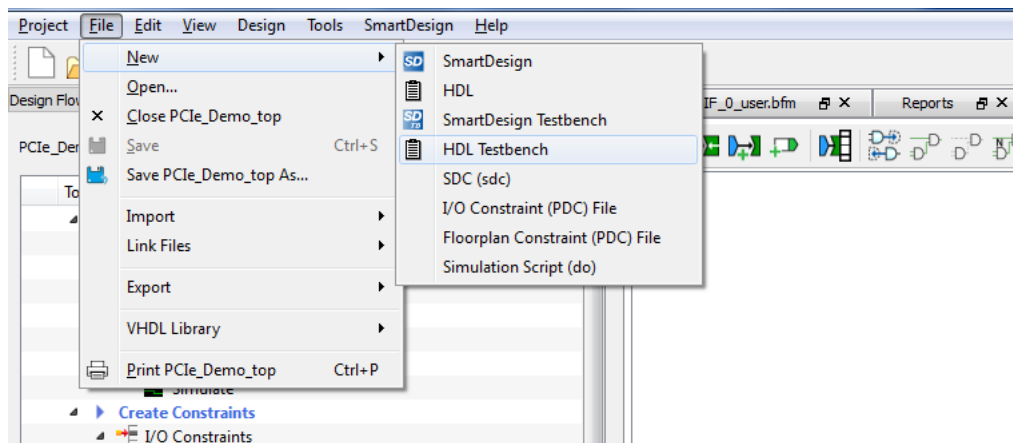
7. Click **Close** to close the **Project Settings** dialog box.
8. Click **Save** when prompted to save the changes.

2.5.1 Generating Testbench

The following steps describe how to generate Testbench.

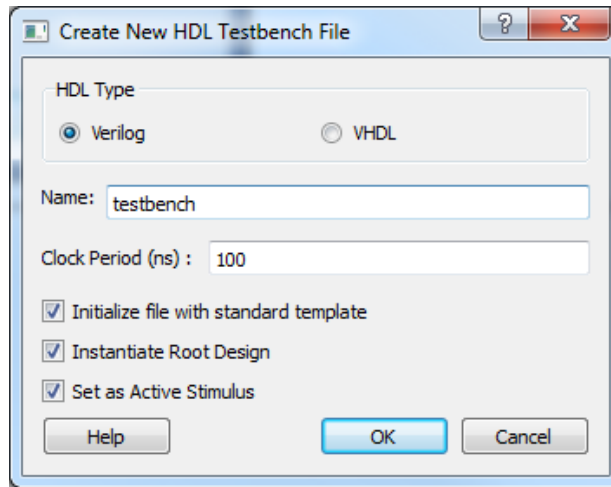
1. Go to **File > New > HDL Test bench**.

Figure 44 • HDL Testbench



2. Select **HDL Type** as **Verilog** or **VHDL** and enter **testbench** as the **Name**.

Figure 45 • Create New HDL Testbench File



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 **220 μ s**. The ModelSim transcript window displays the BFM commands and the BFM simulation completed with no errors, as shown in the following figure.

Figure 46 • SERDES BFM Simulation

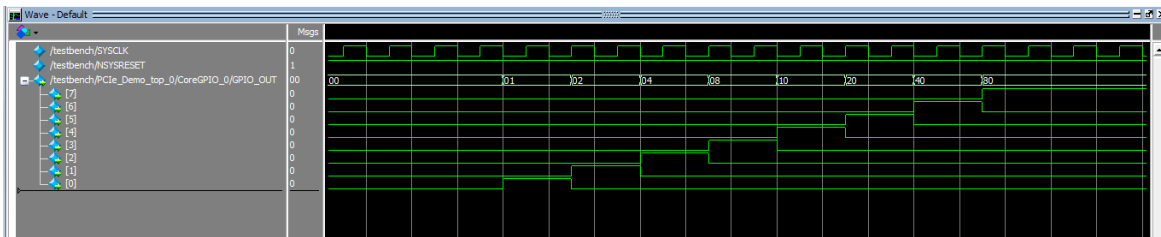
```

Library x Memory List x sim x
Transcript
# BFM: Data Write 400000a0 00000008
# BFM:26:write w 400000a0 00000010 at 211500 ns
# BFM: Data Write 400000a0 00000010
# BFM:27:write w 400000a0 00000020 at 211700 ns
# BFM: Data Write 400000a0 00000020
# BFM:28:write w 400000a0 00000040 at 211900 ns
# BFM: Data Write 400000a0 00000040
# BFM:29:write w 400000a0 00000080 at 212100 ns
# BFM: Data Write 400000a0 00000080
# BFM:30:write w 20000000 12345678 at 212300 ns
# BFM: Data Write 20000000 12345678
# BFM:31:write w 20000004 87654321 at 212600 ns
# BFM: Data Write 20000004 87654321
# BFM:32:write w 20000008 9abcdef0 at 212900 ns
# BFM: Data Write 20000008 9abcdef0
# BFM:33:write w 2000000c 0fedcba9 at 213200 ns
# BFM: Data Write 2000000c 0fedcba9
# BFM:34:readcheck w 20000000 12345678 at 213500 ns
# BFM: Data Read 20000000 12345678 MASK:fffffff at 213700.010000ns
# SFM: Data Read 20000004 87654321 at 213850.010000ns
# BFM:35:readcheck w 20000004 87654321 at 213950 ns
# BFM: Data Read 20000004 87654321 MASK:fffffff at 214150.010000ns
# SFM: Data Read 20000008 9abcdef0 at 214300.010000ns
# BFM:36:readcheck w 20000008 9abcdef0 at 214400 ns
# BFM: Data Read 20000008 9abcdef0 MASK:fffffff at 214600.010000ns
# SFM: Data Read 2000000c 0fedcba9 at 214750.010000ns
# BFM:37:readcheck w 2000000c 0fedcba9 at 214850 ns
# BFM: Data Read 2000000c 0fedcba9 MASK:fffffff at 215050.010000ns
# SFM: Data Read 20000010 0xxxxxxx at 215200.010000ns
# BFM:38:return
#####
#
# SERDESIF_0 BFM Simulation Complete - 20 Instructions - NO ERRORS
#
#####
#
VSIM 2>

```

The following figure shows the waveform window with GPIO_OUT signals.

Figure 47 • Simulation Result with GPIO_OUT Signals

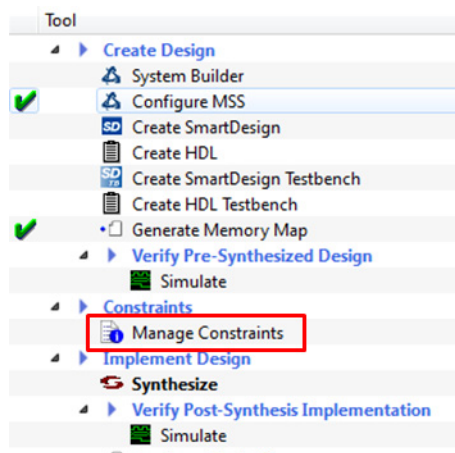


2.6 Step 4: Generating the Program File

The following steps describe how to generate the program file:

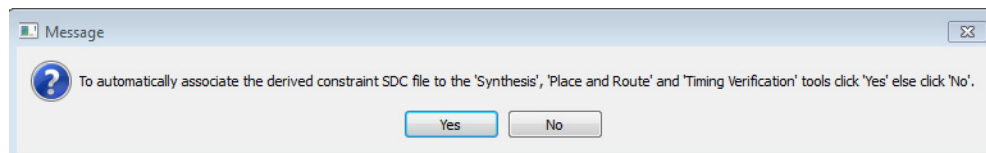
4. Double-click **Manage Constraints** as shown in the following figure, and click **Derive Constraints** option under **Timing** tab to generate the SDC file for root module.

Figure 48 • Manage Constraints



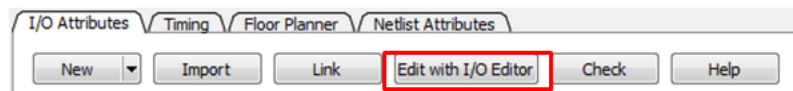
5. Click **Yes** to associate SDC file to synthesis, place and route, and timing verification stages as shown in the following figure.

Figure 49 • Associate the Constraints File



6. Click **Edit with IO Editor** in **Constraint Manager** after completing synthesis. The **IO Editor** window is displayed as shown in.

Figure 50 • I/O Constraints



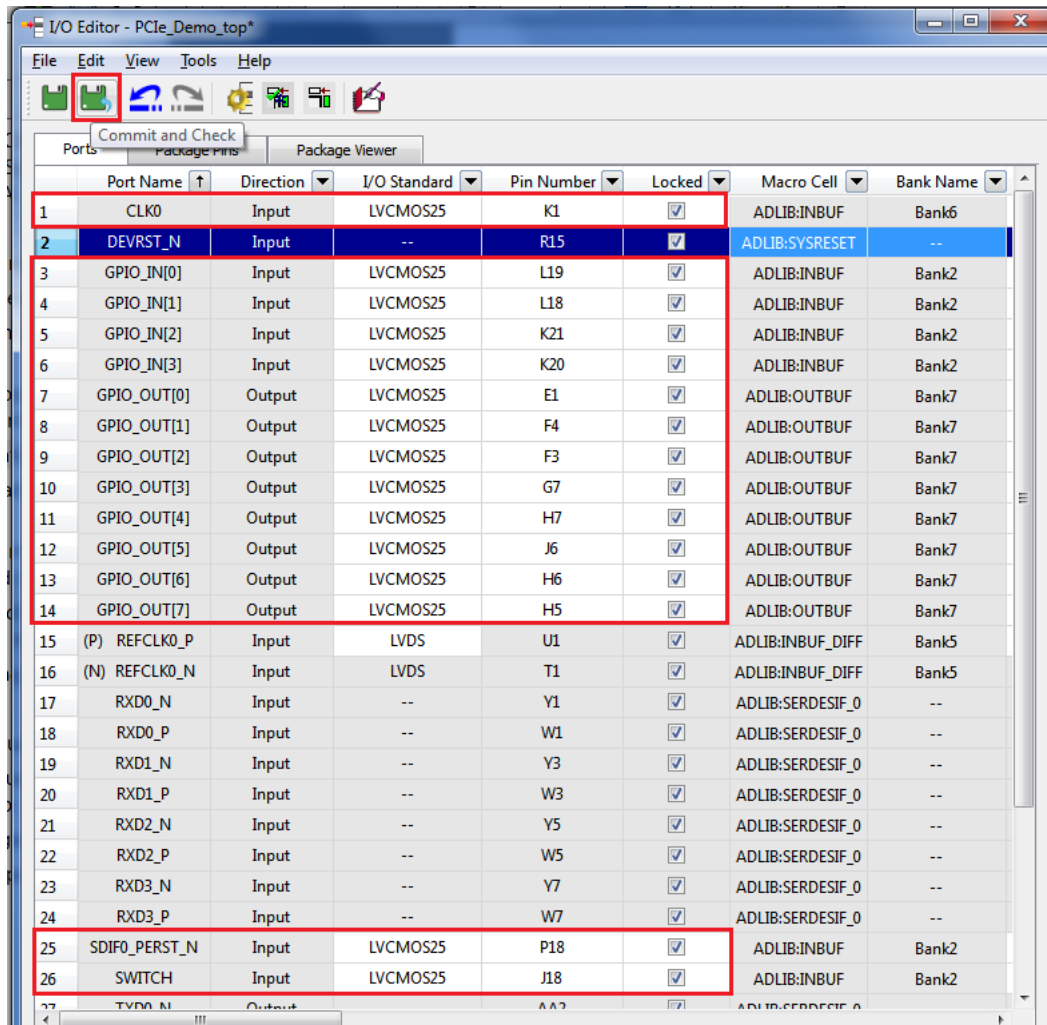
7. In the **I/O Editor** window, make the pin assignments as shown in the following table.

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
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 assigning the pins, the **I/O Editor** is displayed as shown in the following figure.

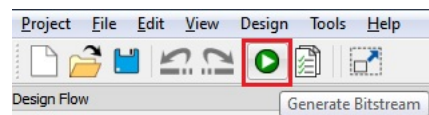
Figure 51 • I/O Editor



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
8. After updating the I/O editor, click **Commit and Check**.
 9. Close the I/O editor.
 10. Click **Verify Timing** to complete place and route, and verify timing.
 11. Click **Generate Bitstream** as shown in the following figure to generate the programming file.

Figure 52 • Generate Programming Data



2.7 Step 5: Programming the IGLOO2 Board Using FlashPro

The following steps describe how to program the IGLOO2 board using Flashpro:

1. Connect the FlashPro4 programmer to the **J5** connector of the IGLOO2 Evaluation Kit board.
2. Connect the jumpers on the IGLOO2 FPGA Evaluation Kit board as shown in the following table.

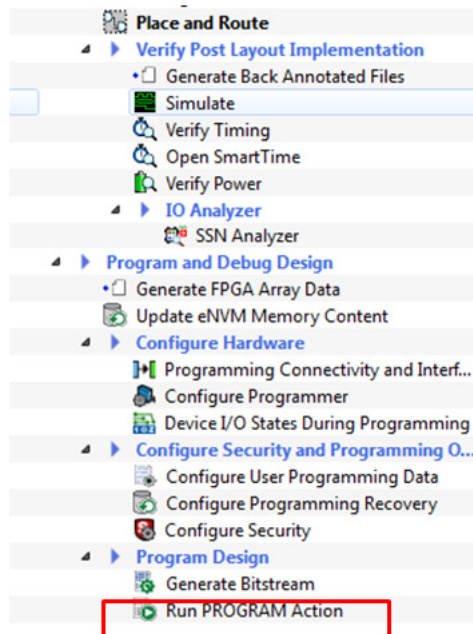
CAUTION: Ensure to switch off **SW7** on the board While connecting the jumpers.

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. Power **ON** the power supply switch, **SW7**. For further details, see [Appendix: IGLOO2 Evaluation Kit Board](#), page 58.
5. To program the IGLOO2 device, double-click **Run PROGRAM Action** in the **Design Flow** window as shown in the following figure.

Figure 53 • Run PROGRAM Action



2.8 Step 6: Connecting the Evaluation Kit to the Host PC

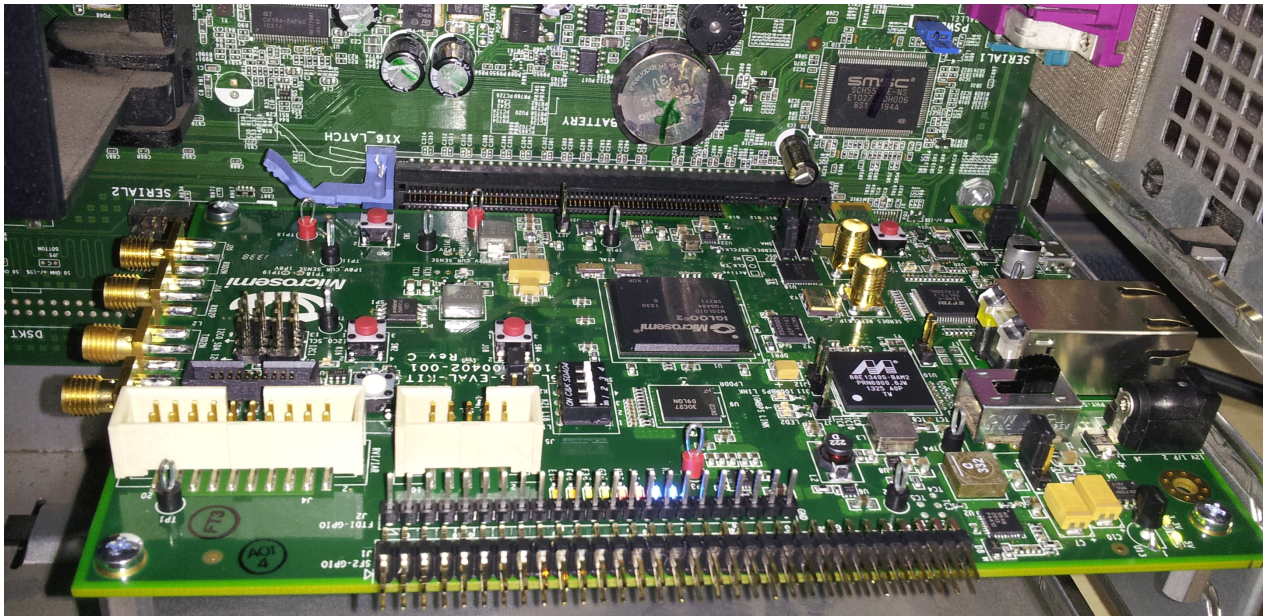
The following steps describe how to connect the IGLOO2 Evaluation Kit to the host PC:

1. After successful programming, power off the IGLOO2 Evaluation Kit board and shut down the host PC.
2. Use the following 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 use a laptop, the express card adapters typically support only Gen1 and the design works on Gen1 mode.

Note: Host PC or laptop must 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 must be powered off during the PCIe card insertion.

The following figure 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, see [Appendix: IGLOO2 Evaluation Kit Board Setup for Laptop](#), page 59.

Figure 54 • IGLOO2 Evaluation Kit Setup for Host PC



2.9 Step 7: Running the Design

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

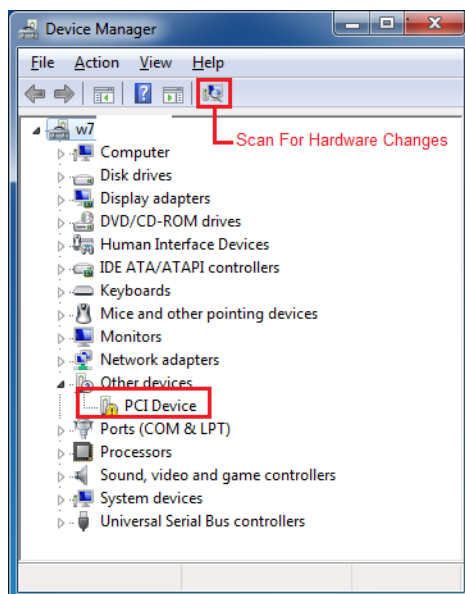
- To run the design on Windows OS GUI, Microsemi PCIe drivers are provided. See "Running the Design on Windows" section.
- To run the design on Linux OS, native RedHat Linux drivers and command line scripts are provided. See "Running the Design on Linux" section.

2.9.1 Running the Design on Windows

The following steps describe how to run the demo design on Windows:

1. Switch **ON** the power supply switch, **SW7**.
2. Power on the host PC and open the host PC **Device Manager** for PCIe device, as shown in the following figure. If the PCIe device is not detected, power cycle the IGLOO2 Evaluation Kit board.
3. Right-click **PCI Device** > **scan for hardware changes** in Device Manager.

Figure 55 • Device Manager



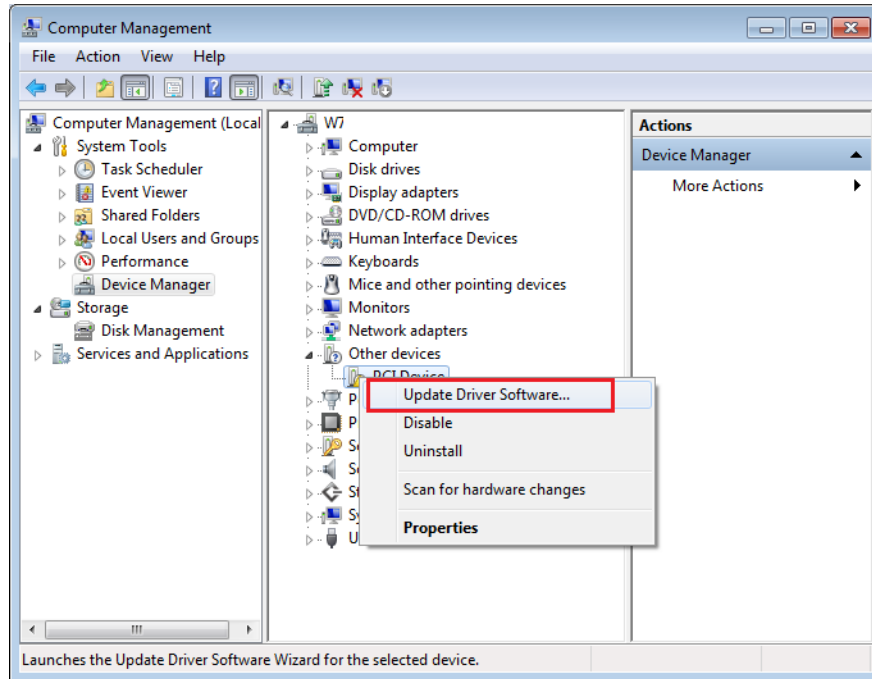
Note: If the device is still not detected, check whether or not the basic input/output system (BIOS) version in host PC is the latest, and if PCIe is enabled in the host PC BIOS.

2.9.1.1 Drivers Installation

Perform the following steps to install the PCIe drivers on the host PC:

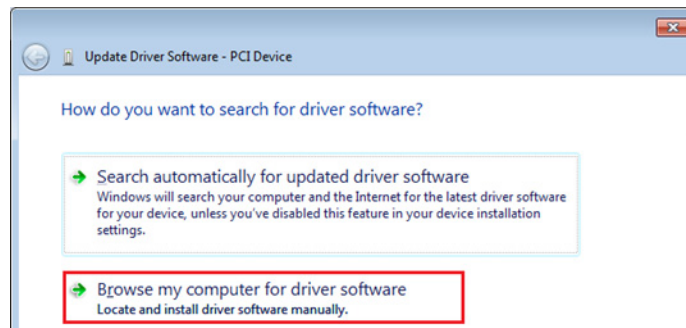
1. Right-click **PCI Device** in Device Manager and select **Update Driver Software...**, as shown in the following figure. To install the drivers, administrative rights are required.

Figure 56 • Update Driver Software



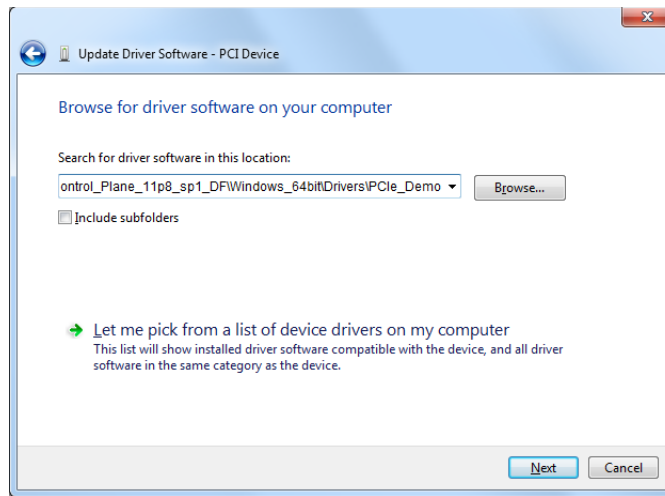
2. In the **Update Driver Software - PCI Device** window, select the **Browse my computer for driver software** option as shown in the following figure.

Figure 57 • Browse for Driver Software



- Browse the drivers folder:
M2GL_PCIE_Control_Plane_11p8_sp1_DF\Windows_64bit\Drivers\PCIe_Demo and click **Next**, as shown in the following figure.

Figure 58 • Browse for Driver Software Continued



- The **Windows Security** dialog box is displayed. Click **Install** as shown in the following figure. After successful driver installation, a message appears. See Figure 60, page 44.

Figure 59 • Windows Security

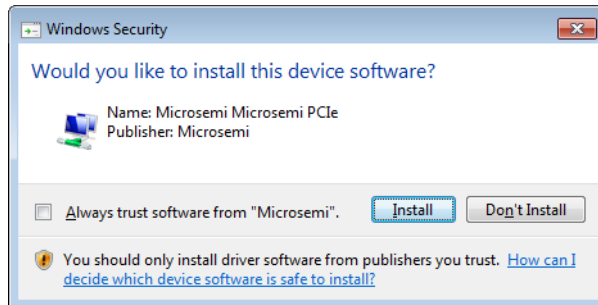
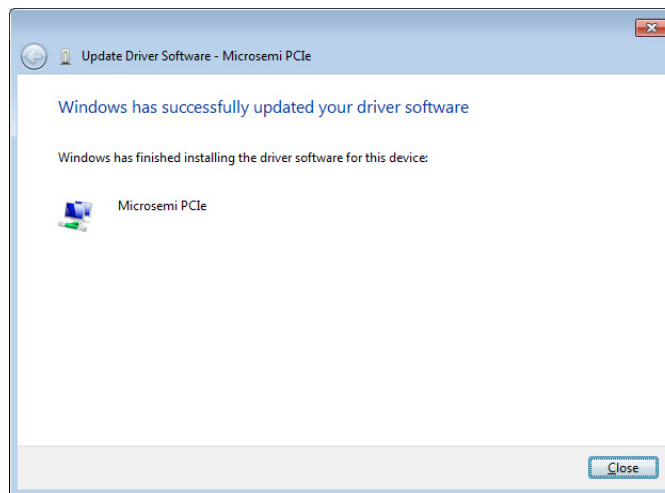


Figure 60 • Successful Driver Installation



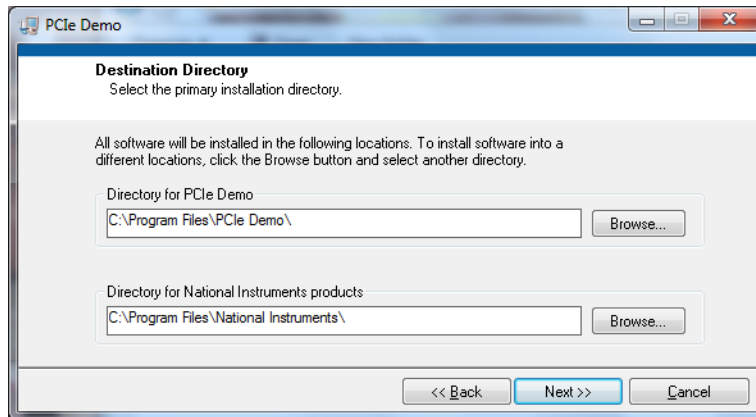
2.9.1.2 PCIe Demo GUI Installation

IGLOO2 PCIe demo 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 the user selection.

To install the GUI:

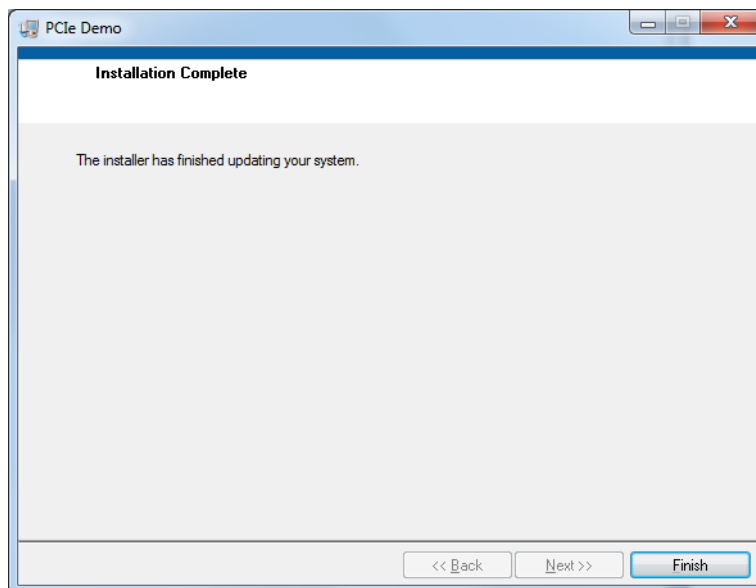
1. Extract the PCIe_Demo_GUI_Installer.rar and locate the files at **M2GL_PCIE_Control_Plane_11p8_sp1_DF\GUI**
2. Double-click `setup.exe` in the provided GUI installation (`PCIe_Control_Plane_Demo_GUI_Installer\setup.exe`). Apply default options, as shown in the following figure.
3. To start the installation, click **Next**.

Figure 61 • GUI Installation



4. Click **Finish** to complete the installation.

Figure 62 • Successful GUI Installation



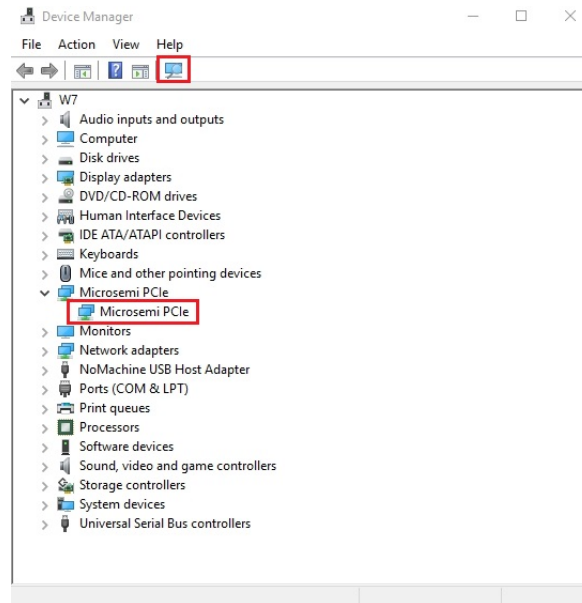
5. Restart the host PC.

2.9.1.3 Running the PCIe GUI

The following steps describe how to run the PCIe GUI:

1. Check the host PC **Device Manager** for the drivers. If the device is not detected, power cycle the IGLOO2 Evaluation Kit board.
2. Click **scan for hardware changes** in Device Manager window. Ensure that the board is switched on.

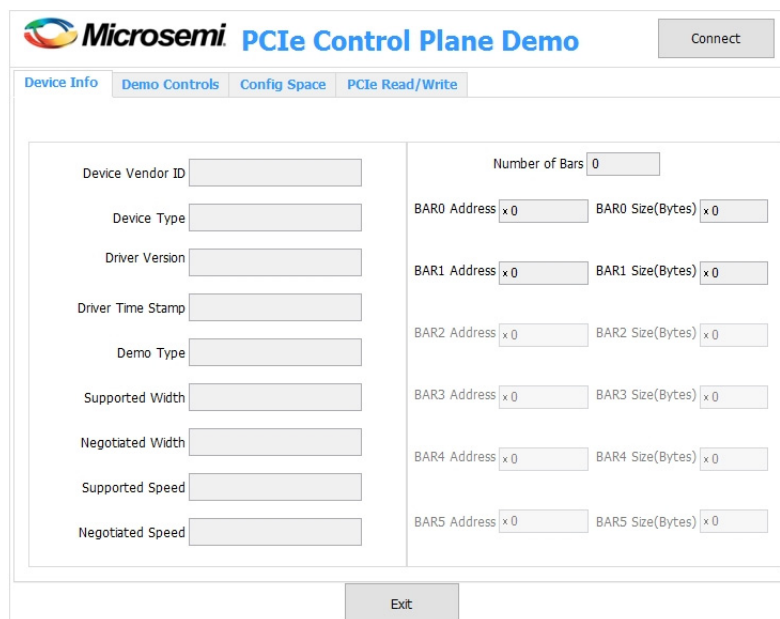
Figure 63 • Device Manager—PCIe Device Detection



Note: If a warning symbol is displayed on the **Microsemi PCIe** in the **Device Manager**, uninstall them and start from step 1 of **Drivers Installation**, page 43.

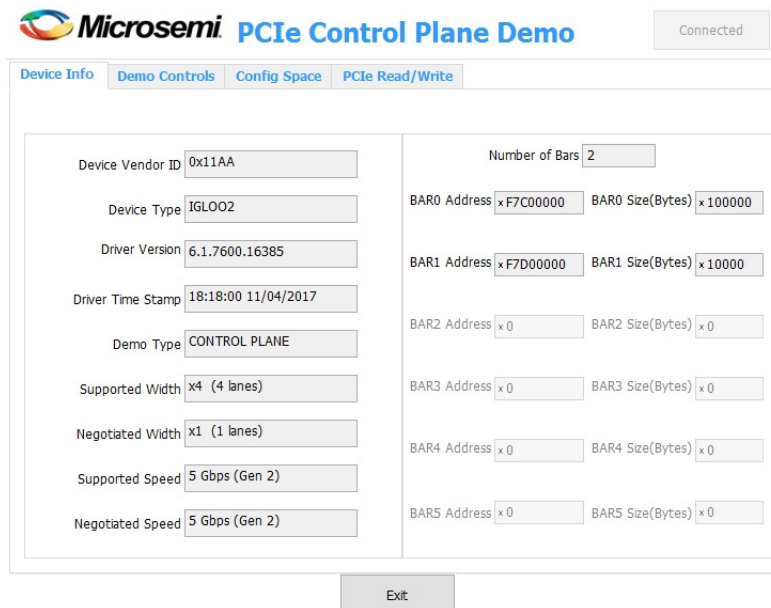
3. Invoke the GUI from **ALL Programs > PCIe Control Plane Demo**. The GUI is displayed, as shown in the following figure.

Figure 64 • PCIe Demo GUI



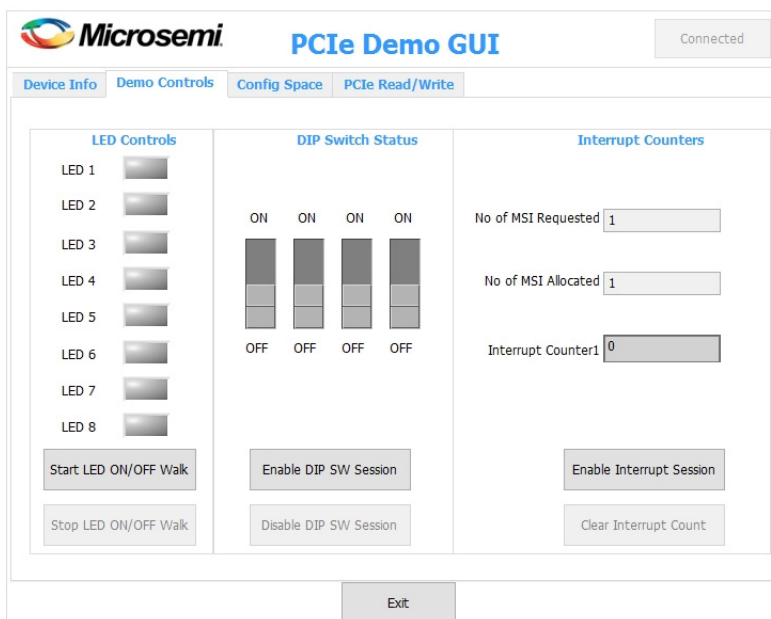
- Click **Connect**. The application detects and displays the information related to the connected kit such as Device Vendor ID, Device Type, Driver Version, Driver Time Stamp, Demo Type, Supported Width, Negotiated Width, Supported Speed, Negotiated Speed, Number of Bars, and BAR Address as shown in the following figure.

Figure 65 • Device Info



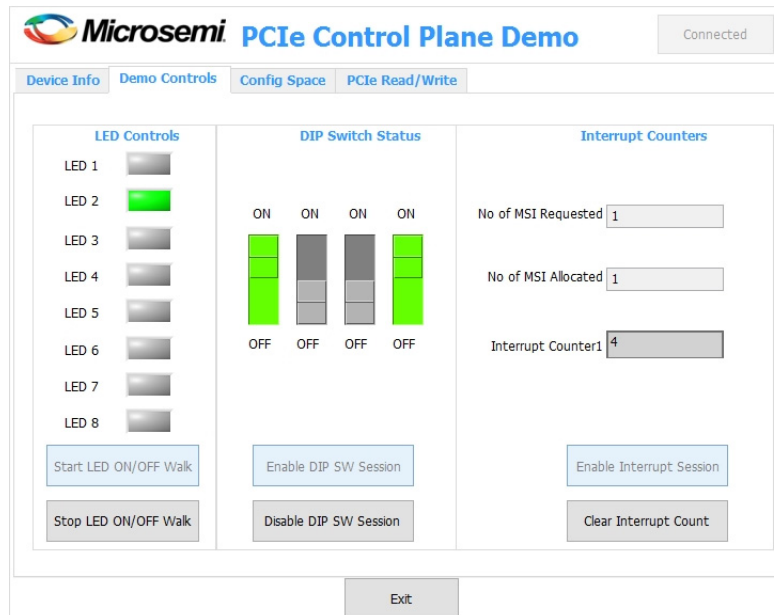
- Click the **Demo Controls** tab to display the **LED Controls**, **DIP Switch Status**, and **Interrupt Counters** as shown in the following figure.

Figure 66 • Demo Controls



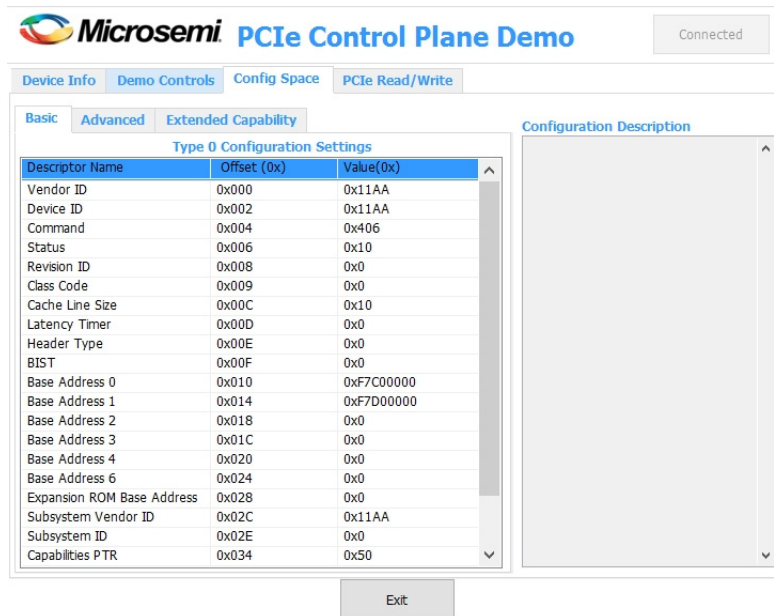
6. Click **Start LED ON/OFF Walk**, **Enable DIP SW Session**, and **Enable Interrupt Session** to view controlling LEDs, getting the DIP switch status, and monitoring the interrupts simultaneously as shown in the following figure.
7. Change the DIP switch positions on the IGLOO2 Evaluation Kit board (**SW5**) and observe the similar position of switches in **GUI SWITCH MODULE**.
8. Press **SW4** on the IGLOO2 Evaluation Kit board and observe the interrupt count on the **Interrupt Counter** field in GUI.

Figure 67 • Demo Controls—Continued



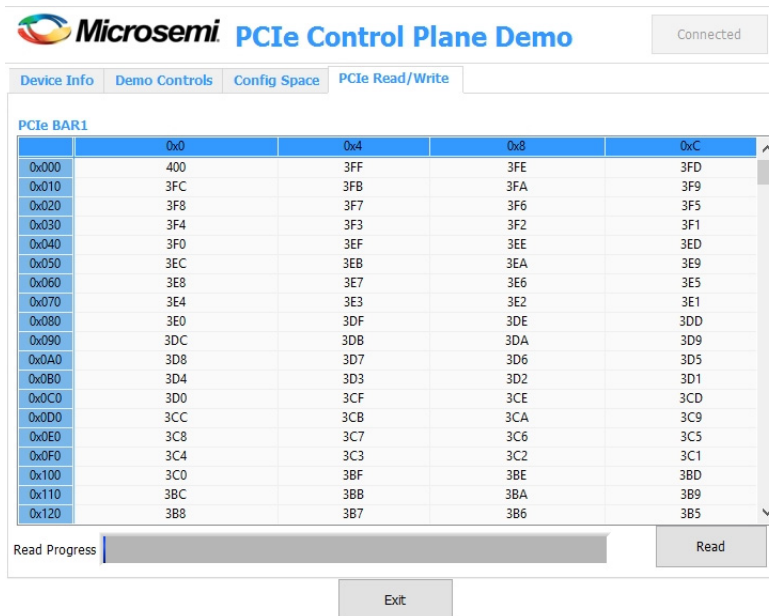
9. Click **Config Space** to view details about the PCIe configuration space. The following figure shows the PCIe configuration space.

Figure 68 • Configuration Space



- Click the **PCIe Read/Write** tab to perform read and writes to LSRAM memory through **BAR1** space. Click **Read** to read the 4 KB memory mapped to BAR1 space as shown in the following figure.

Figure 69 • PCIe BAR1 Memory Access



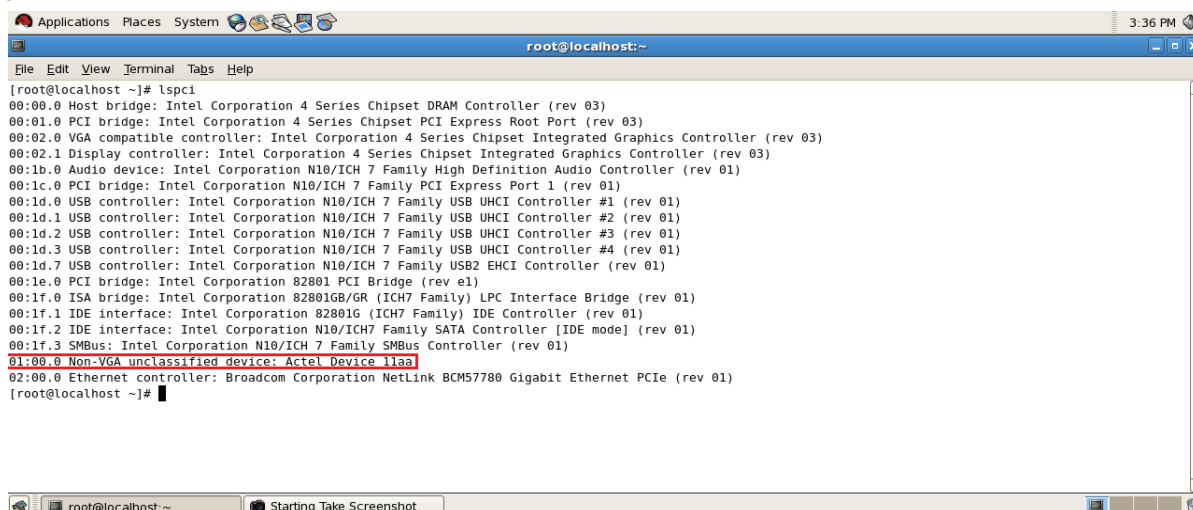
- Click **Exit** to quit the demo.

2.9.2 Running the Design on Linux

The following steps describe how to run the design on Linux:

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

Figure 70 • PCIe Device Detection

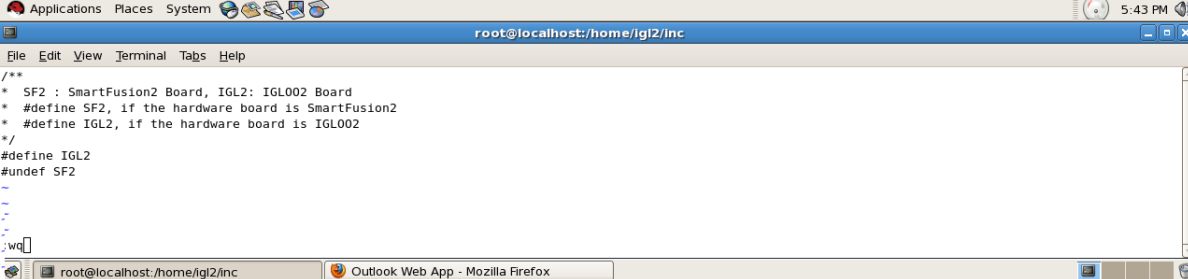


2.9.2.1 Installation

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 `# mkdir /home/igl2` command.
2. Copy the `M2GL_PCIE_Control_Plane_11p8_sp1_DF\Linux_64bit\Drivers\PCle_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_11p8_sp1_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 `PCle_Driver/ inc/` folders.
4. Execute `ls` command to display the contents of `/home/igl2` directory.
ls
5. Change to `inc/` directory by using the `#cd /home/igl2/inc` command.
6. Edit the `board.h` file for IGLOO2 Evaluation Kit:
#vi board.h
#define IGL2
#undef SF2

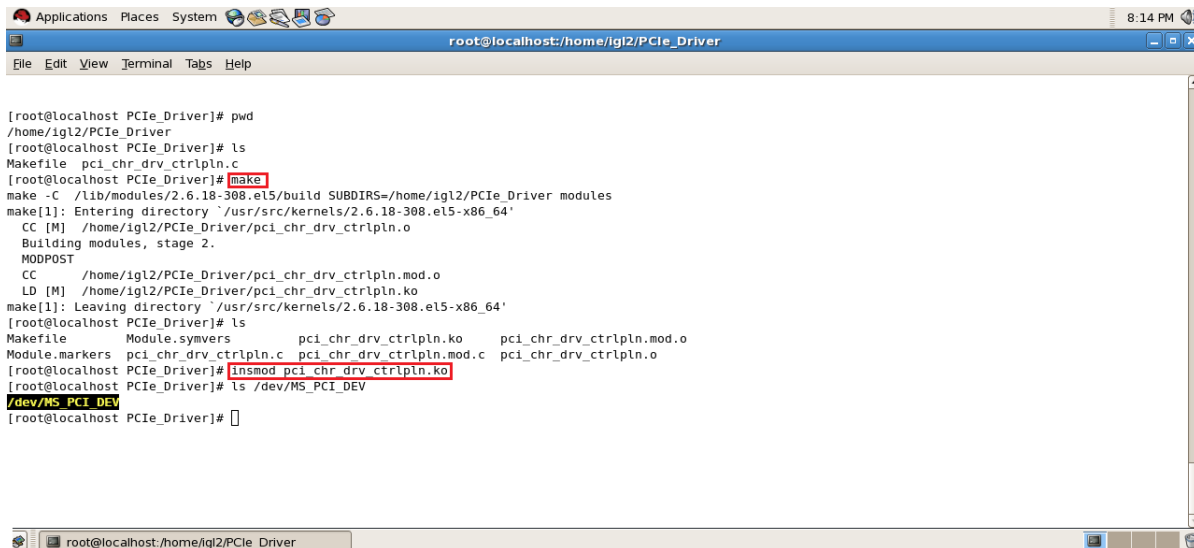
Figure 71 • Edit board.h file



```
root@localhost:/home/igl2/inc
File Edit View Terminal Tabs Help
/**
 * SF2 : SmartFusion2 Board, IGL2: IGL002 Board
 * #define SF2, if the hardware board is SmartFusion2
 * #define IGL2, if the hardware board is IGL002
 */
#define IGL2
#undef SF2
~
~
~
:wq
```

7. To save the selected file, execute the `:wq` command.
8. Change the `PCle_Driver/` directory using `cd` command:
#cd /home/igl2/PCle_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 72 • PCIe Device Driver Installation


```

[root@localhost PCIe_Driver]# pwd
/home/igl2/PCIE_Driver
[root@localhost PCIe_Driver]# ls
Makefile pci_chr_drv_ctrln.c
[root@localhost PCIe_Driver]# make
make -C /lib/modules/2.6.18-308.el5/build SUBDIRS=/home/igl2/PCIE_Driver modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-308.el5-x86_64'
  CC [M] /home/igl2/PCIE_Driver/pci_chr_drv_ctrln.o
  Building modules, stage 2.
  MODPOST
  CC      /home/igl2/PCIE_Driver/pci_chr_drv_ctrln.mod.o
  LD [M] /home/igl2/PCIE_Driver/pci_chr_drv_ctrln.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_ctrln.ko pci_chr_drv_ctrln.mod.o
Module.markers pci_chr_drv_ctrln.c pci_chr_drv_ctrln.mod.c pci_chr_drv_ctrln.o
[root@localhost PCIe_Driver]# insmod pci_chr_drv_ctrln.ko
[root@localhost PCIe_Driver]# ls /dev/MS_PCI_DEV
/dev/MS_PCI_DEV
[root@localhost PCIe_Driver]#

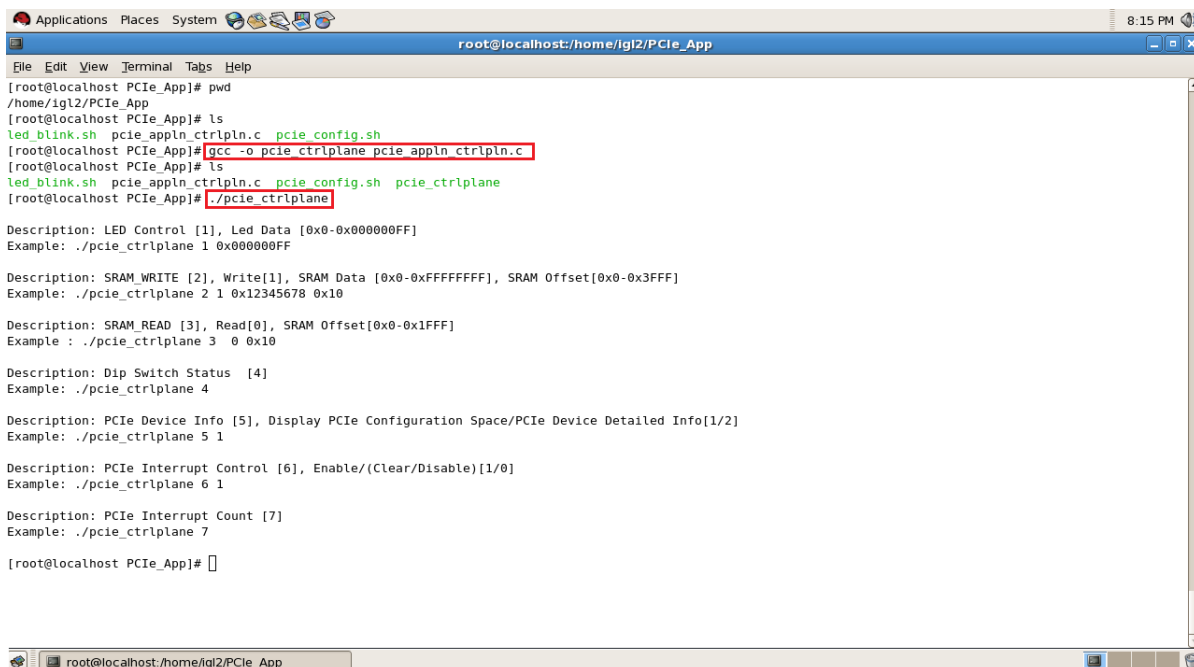
```

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

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

2.9.2.2 Linux PCIe Application Compilation and PCIe Control Plane Utility Creation

1. Change to the `/home/igl2/` directory using the `#cd /home/igl2` command.
2. Copy the `M2GL_PCIE_Control_Plane_11p8_sp1_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 `#cd /home/igl2/PCie_App` command.
4. Compile the Linux user space application `pcie_appln_ctrln.c` by using `gcc` command.
`#gcc -o pcie_ctrlplane pcie_appln_ctrln.c`

Figure 73 • Linux PCIe Application Utility


```

[root@localhost PCIe_App]# pwd
/home/igl2/PCie_App
[root@localhost PCIe_App]# ls
led_blink.sh pcie_appln_ctrln.c pcie_config.sh
[root@localhost PCIe_App]# gcc -o pcie_ctrlplane pcie_appln_ctrln.c
led_blink.sh pcie_appln_ctrln.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[0x0-0x1FFF]
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]#

```

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`

Help menu displays as shown in Figure 73, page 51.

2.9.2.3 Execution of Linux PCIe Control Plane Features

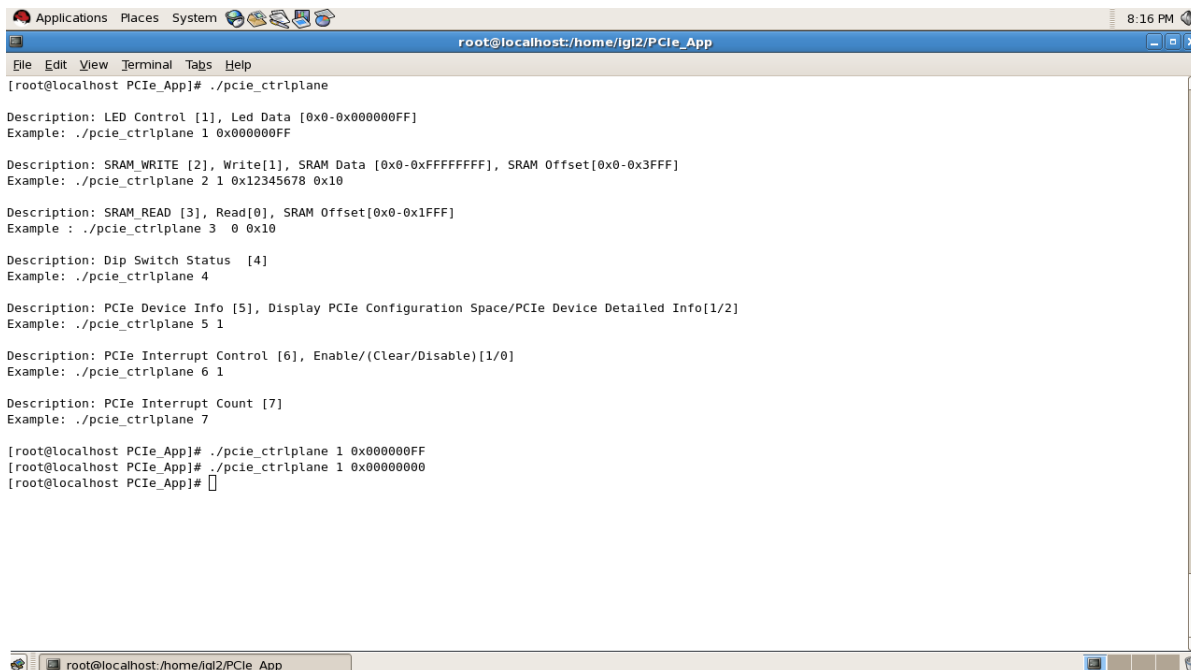
2.9.2.3.1 LED Control

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 74 • Linux Command—LED Control



```

Applications Places System 8:16 PM
root@localhost:/home/ig12/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[0x0-0x1FFF]
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]#

```

`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.

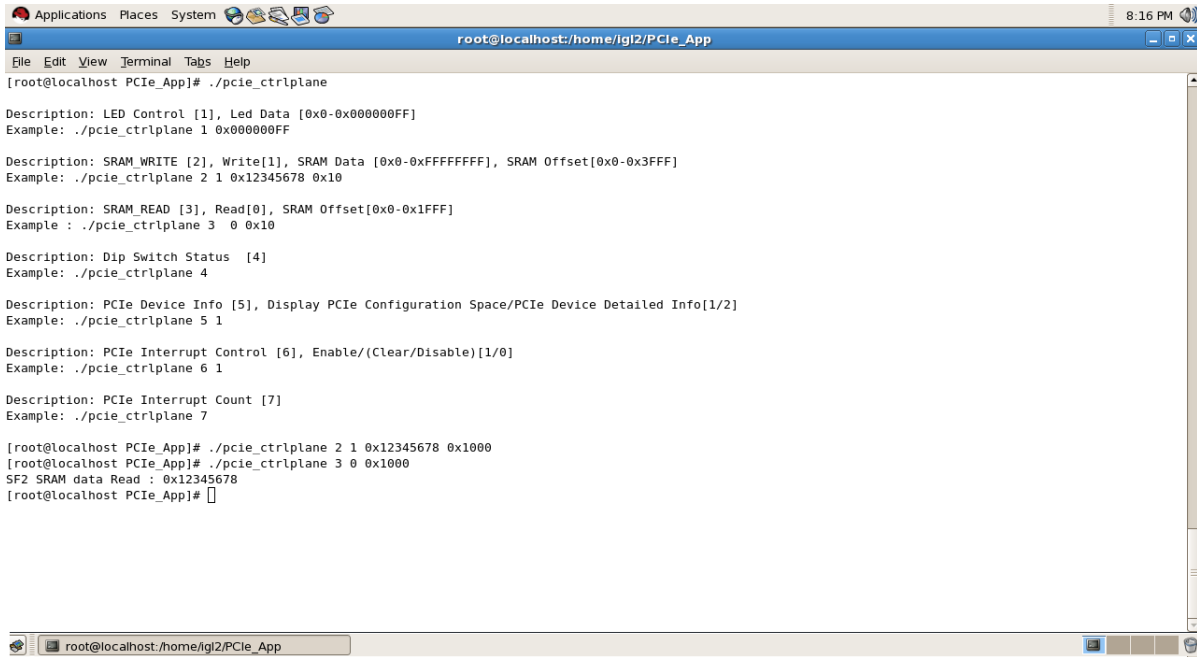
```
#sh led_blink.sh
```

2.9.2.3.2 SRAM Read/Write

32 KB SRAM is accessible for IGLOO2 Evaluation Kit board.

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

Figure 75 • Linux Command—SRAM Read/Write



```

Applications Places System root@localhost:/home/ig2/PCie_App 8:16 PM
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[0x0-0x1FFF]
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]#

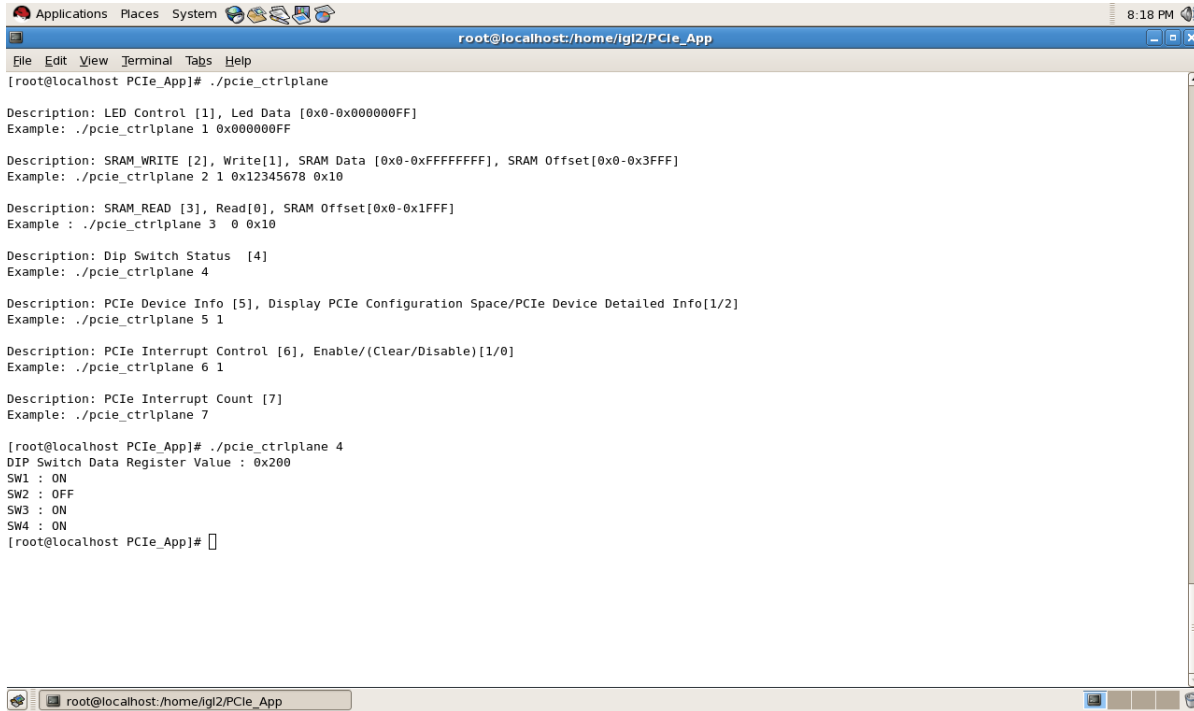
```

2.9.2.3.3 DIP Switch Status

Dip switch on IGLOO2 Evaluation Kit board consists 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 76 • Linux Command—DIP Switch]



```

root@localhost:/home/ig2/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[0x0-0x1FFF]
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 : 0x200
SW1 : ON
SW2 : OFF
SW3 : ON
SW4 : ON
[root@localhost PCie_App]#

```

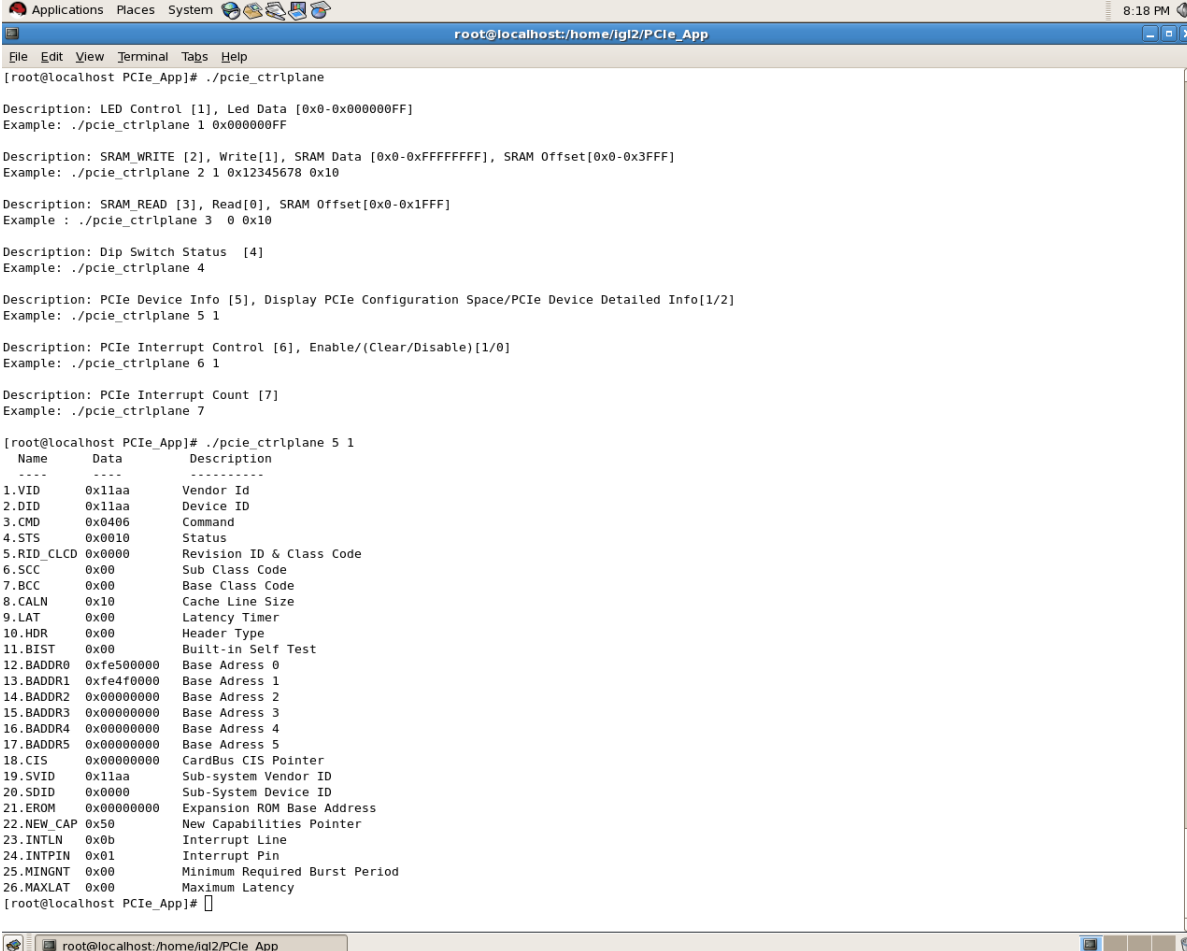
2.9.2.3.4 PCIe Configuration Space Display

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

Root Privileges are required to execute this command.

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

Figure 77 • Linux Command—PCIe Configuration Space Display



```

Applications Places System 8:18 PM
root@localhost:/home/ig2/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[0x0-0x1FFF]
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]#

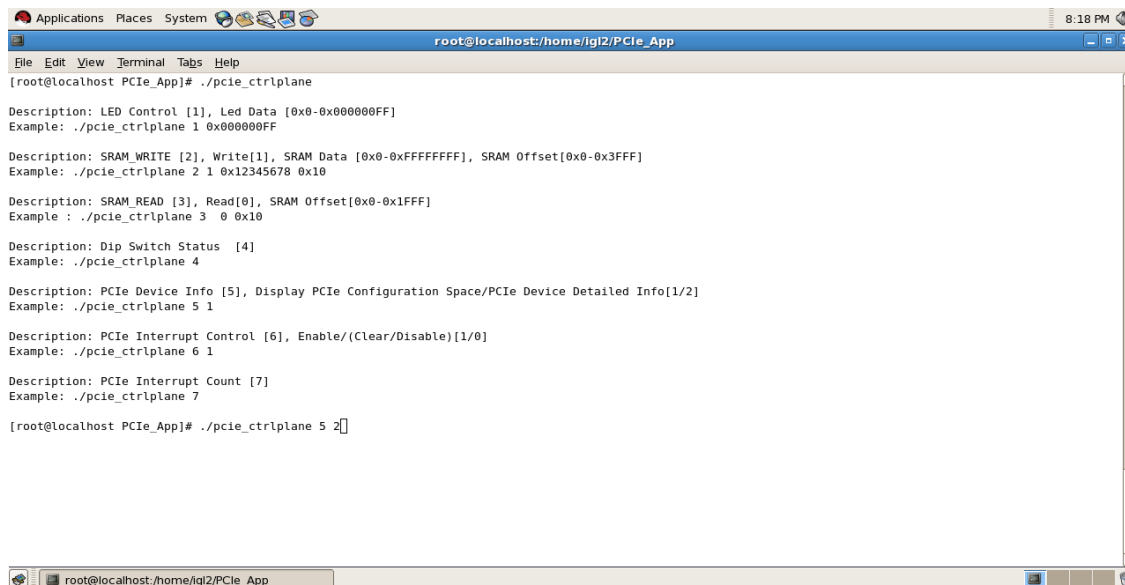
```

2.9.2.3.5 PCIe Link Speed and Width

Root Privileges are required to execute this command.

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

Figure 78 • Linux Command—PCIe Link Speed and Width



```

root@localhost:~/home/ig2/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[0x0-0x1FFF]
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

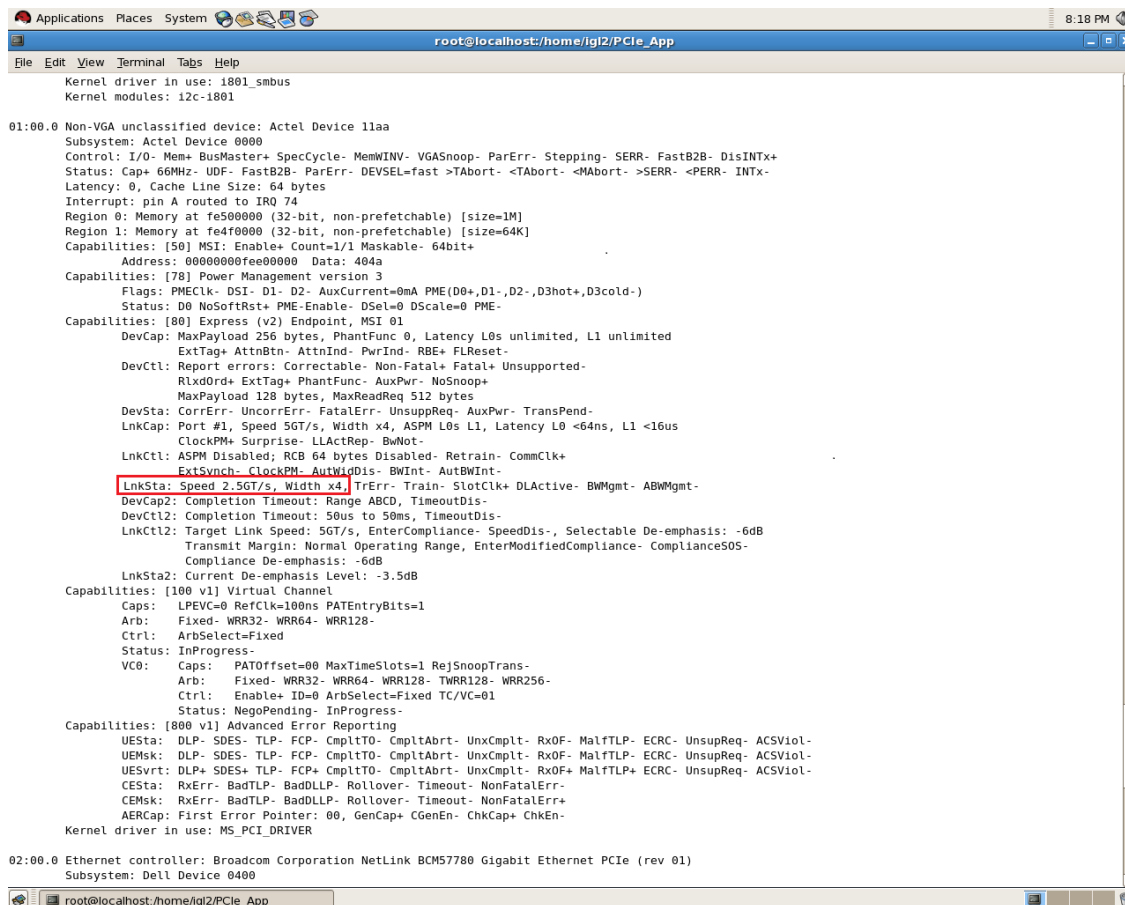
Description: PCIe Interrupt Control [6], Enable/(Clear/Disable)[1/0]
Example: ./pcie_ctrlplane 6 1

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

[root@localhost PCIE_App]# ./pcie_ctrlplane 5 2

```

Figure 79 • Linux Command—PCIe Link Speed and Width



```

root@localhost:~/home/ig2/PCIE_App
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 >Abort- <Abort- <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+ AttnBtm- AttnInd- PwrInd- RBE+ FLReset-
DevCtl: Report errors: Correctable- Non-Fatal+ Fatal+ Unsupported-
Rlx0Ord+ 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- Compliance50S-
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- CmplT0- CmplTAbt- UnxCmpl- RxOF- MalFTLP- ECRC- UnsupReq- ACSViol-
UEMsk: DLP- SDES- TLP- FCP- CmplT0- CmplTAbt- UnxCmpl- RxOF- MalFTLP- ECRC- UnsupReq- ACSViol-
UESvrt: DLP+ SDES+ TLP- FCP+ CmplT0- CmplTAbt- UnxCmpl- RxOF+ MalFTLP+ ECRC- UnsupReq- ACSViol-
CESSta: 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

```

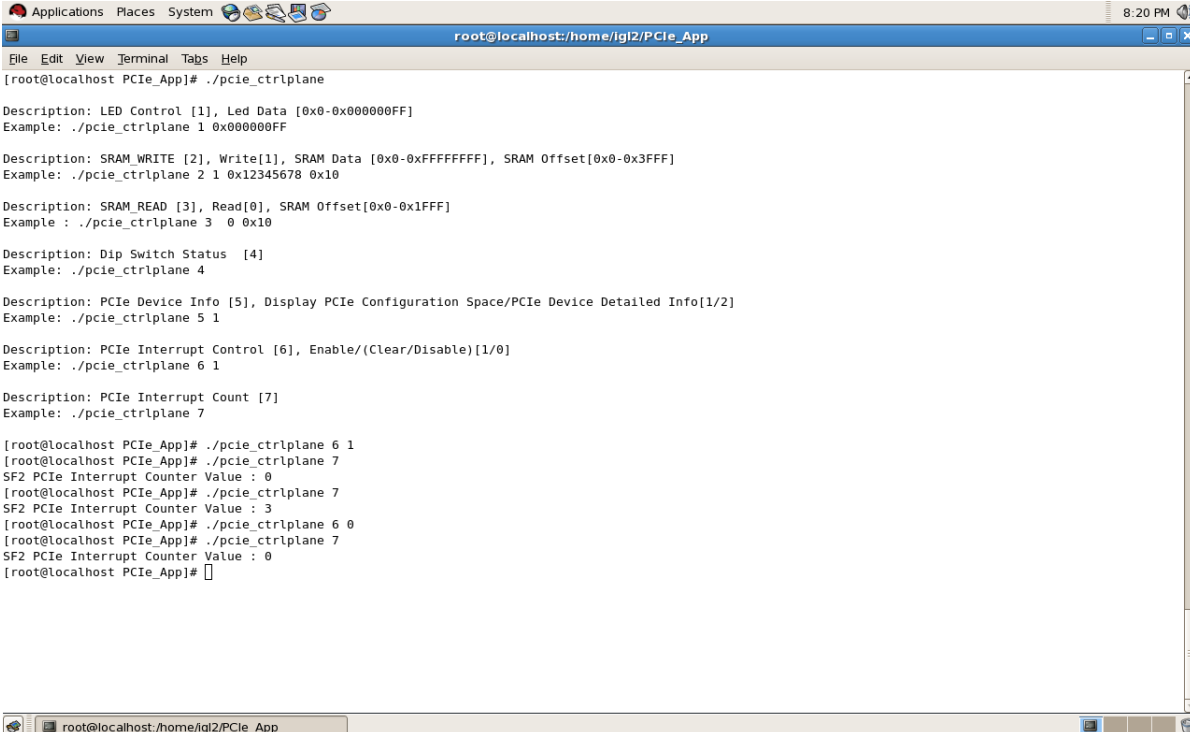
2.9.2.3.6 PCIe Interrupt Control (Enable/Disable) and Interrupt Counter

IGLOO2 Evaluation Kit board enables or disables 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 80 • Linux Command—PCIe Interrupt Control



```
root@localhost:~/home/ig12/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[0x0-0x1FFF]
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 : 3
[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]#
```

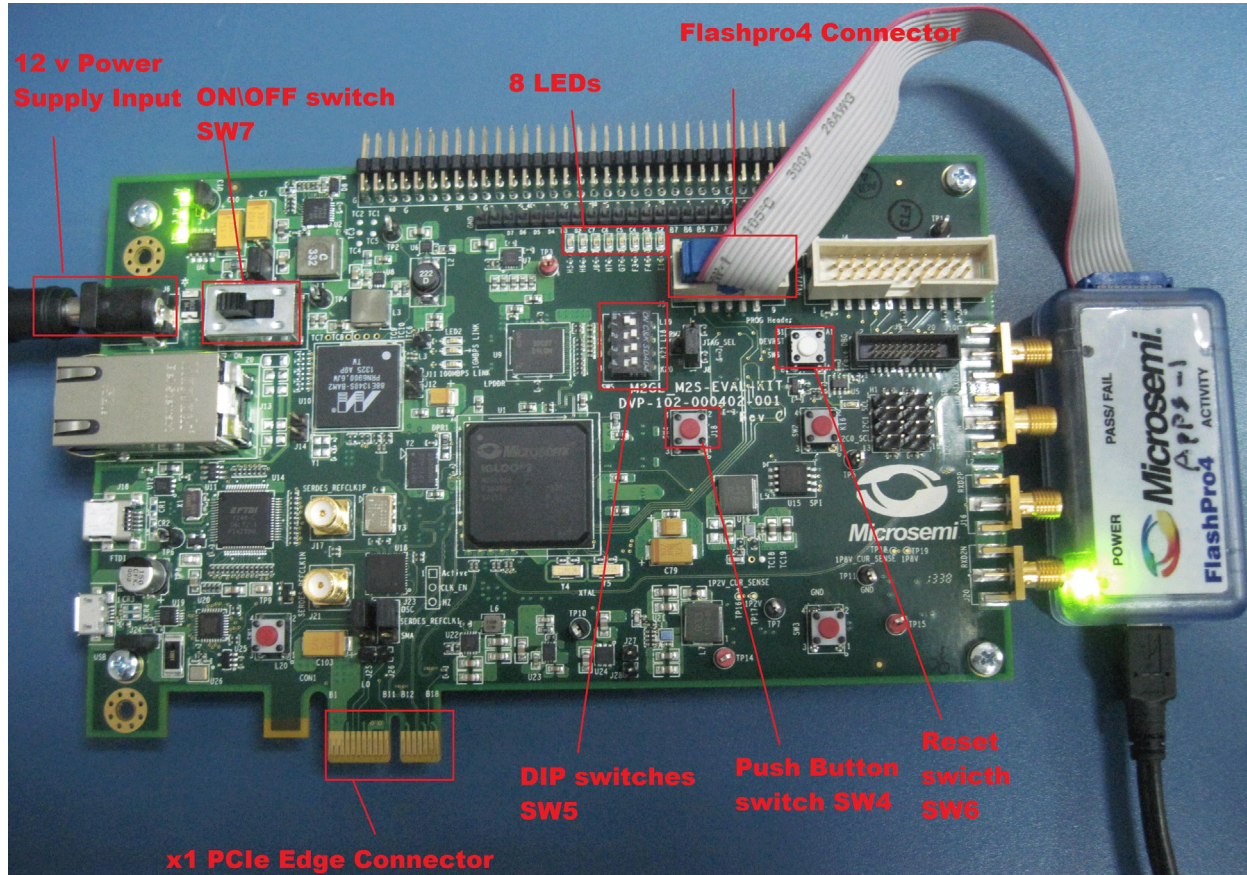
2.10 Conclusion

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

3 Appendix: IGLOO2 Evaluation Kit Board

The following figure shows the IGLOO2 Evaluation Kit board.

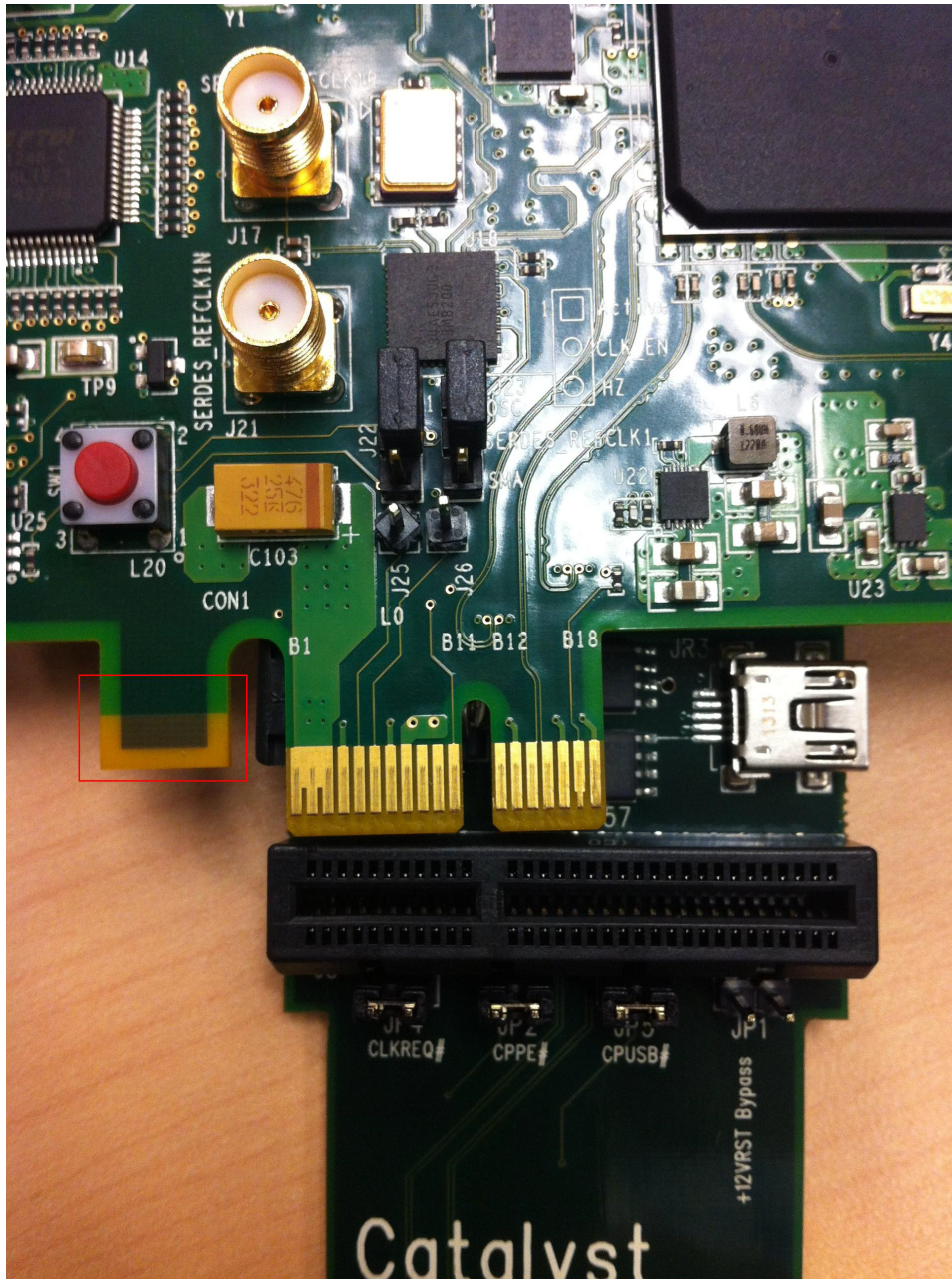
Figure 81 • IGLOO2 Evaluation Kit Board



4 Appendix: IGLOO2 Evaluation Kit Board Setup for Laptop

The following figure shows how to line up the IGLOO2 Evaluation Kit PCIe connector with the adapter card slot.

Figure 82 • Lining Up the IGLOO2 Evaluation Kit Board



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

The following figure shows IGLOO2 Evaluation Kit PCIe connector inserted into the PCIe adapter card slot.

Figure 83 • Inserting the IGLOO2 Evaluation Kit PCIe Connector



The following figure shows the PCIe adapter card and the IGLOO2 Evaluation Kit connected to the laptop.

Figure 84 • IGLOO2 Evaluation Kit Connected to the Laptop

