## Contents

1 Revision History ................................................................. 1  
   1.1 Revision 5.0 ................................................................. 1  
   1.2 Revision 4.0 ................................................................. 1  
   1.3 Revision 3.0 ................................................................. 1  
   1.4 Revision 2.0 ................................................................. 1  
   1.5 Revision 1.0 ................................................................. 1  

2 SmartFusion2 SoC FPGA PCIe Control Plane Demo For Advanced  
Development Kit ................................................................. 2  
   2.1 Design Requirements .................................................... 2  
   2.2 Demo Design ............................................................... 3  
      2.2.1 Demo Design Features ............................................ 4  
      2.2.2 Demo Design Description ....................................... 4  
   2.3 Setting Up the Demo Design ............................................ 6  
      2.3.1 Board Setup ....................................................... 7  
      2.3.2 Programming the Board ........................................... 7  
      2.3.3 Connecting the Board to the Host PC .......................... 9  
   2.4 Running the Demo Design ............................................... 10  
      2.4.1 Running the Demo Design on Windows .......................... 11  
      2.4.2 Running the Demo Design on Linux ............................. 18  
   2.5 Conclusion ............................................................... 27  

3 Appendix: SmartFusion2 Advanced Development Kit Board ........... 28
## Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>PCIe Control Plane Demo Top-Level Block Diagram</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Demo Design Files Top-Level Structure</td>
<td>3</td>
</tr>
<tr>
<td>Figure 3</td>
<td>PCIe Control Plane Demo Block Diagram</td>
<td>4</td>
</tr>
<tr>
<td>Figure 4</td>
<td>SERDES BFM Simulation</td>
<td>5</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Simulation Result with GPIO_OUT Signals</td>
<td>6</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Device Manager</td>
<td>6</td>
</tr>
<tr>
<td>Figure 7</td>
<td>FlashPro New Project</td>
<td>7</td>
</tr>
<tr>
<td>Figure 8</td>
<td>FlashPro5 Programmer Type</td>
<td>8</td>
</tr>
<tr>
<td>Figure 9</td>
<td>FlashPro Project Configured</td>
<td>9</td>
</tr>
<tr>
<td>Figure 10</td>
<td>SmartFusion2 Advanced Development Kit Setup for Host PC</td>
<td>10</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Device Manager</td>
<td>11</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Update Driver Software</td>
<td>12</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Browse for Driver Software</td>
<td>12</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Browse for Driver Software Continued</td>
<td>13</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Windows Security</td>
<td>13</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Successful Driver Installation</td>
<td>13</td>
</tr>
<tr>
<td>Figure 17</td>
<td>GUI Installation</td>
<td>14</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Successful GUI Installation</td>
<td>14</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Device Manager—PCIe Device Detection</td>
<td>15</td>
</tr>
<tr>
<td>Figure 20</td>
<td>PCIe Demo GUI</td>
<td>15</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Device Info</td>
<td>16</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Demo Controls</td>
<td>16</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Demo Controls—Continued</td>
<td>17</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Configuration Space</td>
<td>17</td>
</tr>
<tr>
<td>Figure 25</td>
<td>PCIe BAR1 Memory Access</td>
<td>18</td>
</tr>
<tr>
<td>Figure 26</td>
<td>PCIe Device Detection</td>
<td>18</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Edit board.h File</td>
<td>20</td>
</tr>
<tr>
<td>Figure 28</td>
<td>PCIe Device Driver Installation</td>
<td>20</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Linux PCIe Application Utility</td>
<td>21</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Linux Command—LED Control</td>
<td>22</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Linux Command—DIP Switch</td>
<td>23</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Linux Command—PCIe Configuration Space Display</td>
<td>24</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Linux Command—PCIe Link Speed and Width</td>
<td>25</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Linux Command—PCIe Link Speed and Width</td>
<td>26</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Linux Command—PCIe Interrupt Control</td>
<td>27</td>
</tr>
<tr>
<td>Figure 36</td>
<td>SmartFusion2 Advanced Development Kit Board</td>
<td>28</td>
</tr>
</tbody>
</table>
### Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Design Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>SmartFusion2 FPGA Advanced Kit Jumper Settings</td>
<td>6</td>
</tr>
</tbody>
</table>
1 Revision History

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

1.1 Revision 5.0
Updated the document for Libero v11.8 SP1 software release.

1.2 Revision 4.0
Updated the document for Libero v11.7 software release.

1.3 Revision 3.0
Updated the document for Libero v11.6 software release.

1.4 Revision 2.0
Updated the document for Libero v11.5 software release.

1.5 Revision 1.0
Revision 1.0 was the first publication of this document.
2 SmartFusion2 SoC FPGA PCIe Control Plane Demo For Advanced Development Kit

The SmartFusion2 SoC FPGA devices integrate a fourth generation flash-based FPGA fabric and an ARM Cortex-M3 processor, along with high performance communication interfaces on a single chip. The SmartFusion2 high speed serial interface (SERDESIF) provides a fully hardened PCIe endpoint (EP) implementation and is compliant with PCIe Base Specification Revision 2.0, 1.1, and 1.0. For more details, refer to the UG0447: SmartFusion2 and IGLOO2 High Speed Serial Interfaces User Guide.

The demo explains the SmartFusion2 embedded PCI Express feature and how this can be used as a low bandwidth control plane interface using the SmartFusion2 Advanced Development Kit board.

The demo provides a simple design to access the SmartFusion2 PCIe EP from a Host PC. A GUI is provided for read and write access to the SmartFusion2 PCIe configuration space and memory space of BAR0 and BAR1. It also provides the Host PC device drivers for the SmartFusion2 PCIe EP. It can run on both Windows and Red Hat Linux operating system (OS).

The following figure shows the top-level block diagram for the PCIe control plane demo. The demo design uses a SmartFusion2 PCIe interface with a maximum link width of x4 to interface with a Host PC PCIe Gen 2 slot. If the Host PC does not support Gen 2 slot, the design automatically changes to Gen 1 slot. The SmartFusion2 microcontroller subsystem (MSS) GPIOs control the LEDs and switches on the SmartFusion2 Advanced Development Kit board using the PCIe interface. The Host PC can also read memory and writes to the SmartFusion2 eSRAM through GUI. The Host PC can also be interrupted by using the push button on the SmartFusion2 Advanced Development Kit board.

![Figure 1 • PCIe Control Plane Demo Top-Level Block Diagram](image)

2.1 Design Requirements

The following table lists the design requirements details.

<table>
<thead>
<tr>
<th>Table 1 • Design Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Requirements</strong></td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
</tr>
<tr>
<td>SmartFusion2 Advanced Development Kit:</td>
</tr>
<tr>
<td>– 12 V adapter</td>
</tr>
<tr>
<td>– FlashPro5</td>
</tr>
<tr>
<td>– USB A to Mini-B cable</td>
</tr>
<tr>
<td>Host PC with an available PCIe 2.0 Gen 1 or Gen 2 compliant slot</td>
</tr>
</tbody>
</table>
Table 1 • Design Requirements (continued)

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libero SoC Design Suite</td>
<td>v11.8 SP1</td>
</tr>
<tr>
<td>FlashPro Programming Software</td>
<td>v11.8 SP1</td>
</tr>
<tr>
<td>Host PC Drivers (provided along with the design files)</td>
<td>–</td>
</tr>
<tr>
<td>GUI executable (provided along with the design files)</td>
<td>–</td>
</tr>
<tr>
<td>SoftConsole</td>
<td>v4.0</td>
</tr>
</tbody>
</table>

2.2 Demo Design

The design files for this demo can be downloaded from the Microsemi website: [http://soc.microsemi.com/download/rsc/?f=m2s_dg0566_libero11p8_sp1_df](http://soc.microsemi.com/download/rsc/?f=m2s_dg0566_libero11p8_sp1_df)

The following figure shows the top-level structure of the design files. For further details, refer to the readme.txt file.

Figure 2 • Demo Design Files Top-Level Structure

```plaintext
<Download folder>

   M2S150_PCIE_Control_Plane_DF
      | GUI
      | Libero_Project
      | Linux_64bit
      | ProgrammingFile
      | Source Files
      | Windows_64bit
      | Readme.txt
```
2.2.1 Demo Design Features
The demo design performs the following tasks:

- Displays PCIe link enable/disable, negotiated link width, and the link speed.
- Controls the status of LEDs on the SmartFusion2 Advanced Development Kit board.
- Displays the position of DIP switches on the SmartFusion2 Advanced Development Kit board.
- Enables read and write to eSRAM.
- Accepts and displays interrupts from the push button on the SmartFusion2 Advanced Development Kit board.
- Displays the SmartFusion2 PCIe Configuration space.

2.2.2 Demo Design Description
The demo design helps to access the SmartFusion2 PCIe EP from the Host PC. The following figure shows a detailed block diagram of the design implementation.

**Figure 3 • PCIe Control Plane Demo Block Diagram**

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

The SERDES_IF_0 is configured for a PCIe 2.0, x4 link width with GEN2 speed for SmartFusion2 Advanced Development Kit board. The PCIe interface to the fabric uses an AMBA High-speed Bus (AHB). The AHB master interface of SERDESIF is enabled and connected to the AHB slave interface of FIC_0 to access the MSS peripherals. The SmartFusion2 PCIe BAR0 and BAR1 are configured in 32-bit memory mapped memory mode.
The advanced eXtensible interface (AXI) master windows of the SERDESIF PCIe provide address translation for accessing one address space from another address space as the PCIe address is different from SmartFusion2 AHB bus matrix address space. The AXI master window 0 is enabled and configured to translate the BAR0 memory address space to the MSS GPIO address space to control the MSS GPIOs. The AXI master window 1 is enabled and configured to translate the BAR1 memory address space to the eSRAM address space to perform read and writes from PCIe.

MSS GPIO block is enabled and configured as below:

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

The PCIe interrupt line is connected to the SW1 push button on the SmartFusion2 Advanced Development Kit. The FPGA clocks are configured to run the FPGA fabric and MSS at 70 MHz.

2.2.2.1 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. Though, the serial communication does not actually go through the High Speed Serial Interface block, this scenario allows validating the fabric interface connections. The SERDESIF_0_user.bfm file under the <LiberoProject>/simulation folder contains the BFM commands to verify the read or write access to MSS GPIOs and eSRAM.

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

- Write to GPIO_OUT[7:0]
- Write to eSRAM
- Read-check from eSRAM

To run the simulation, double-click Simulate under Verify Pre-Synthesized Design in the Design Flow window of the Libero project. ModelSim runs the design for about 270 µs. The ModelSim Transcript window displays the BFM commands and the BFM simulation completed with no errors, as shown in the following figure.

Figure 4 • SERDES BFM Simulation
The following figure shows the Wave window with GPIO_OUT signals.

Figure 5 • Simulation Result with GPIO_OUT Signals

2.3 Setting Up the Demo Design

The following steps describe how to setup the demo for SmartFusion2 Advanced Development Kit board:

1. Connect the Host PC to the J33 Connector using the USB A to mini-B cable. The USB to UART bridge drivers are automatically detected. Verify, if the detection is made in the device manager as shown in the following figure.

Figure 6 • Device Manager

2. Connect the jumpers on the SmartFusion2 Advanced Development Kit board, as listed in the following table.

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

Table 2 • SmartFusion2 FPGA Advanced Kit Jumper Settings

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Pin (From)</th>
<th>Pin (To)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>J116, J353, J354, J54</td>
<td>1</td>
<td>2</td>
<td>These are the default jumper settings of the Advanced Development Kit board. Make sure these jumpers are set accordingly.</td>
</tr>
<tr>
<td>J123</td>
<td>2</td>
<td>3</td>
<td>JTAG programming via FTDI</td>
</tr>
<tr>
<td>J124, J121, J32</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

3. Connect the power supply to the J42 connector on the SmartFusion2 Advanced Development Kit board.
2.3.1 Board Setup

Snapshots of the SmartFusion2 Advanced Development Kit Board with the complete set up are given in the Appendix: SmartFusion2 Advanced Development Kit Board, page 28.

2.3.2 Programming the Board

The following steps describe how to program the board.

1. Download the demo design from: [http://soc.microsemi.com/download/rsc/?f=m2s_dg0566_liberov11p8_sp1_df](http://soc.microsemi.com/download/rsc/?f=m2s_dg0566_liberov11p8_sp1_df)
2. Switch ON the power supply switch, SW7.
3. Launch the FlashPro software.
4. Click New Project.
5. In the New Project window, enter the Project Name as PCIe_Control_Plane.

![FlashPro New Project](image)

6. Click Browse and navigate to the location where you want to save the project.
7. Click Single device as the Programming mode.
8. Click OK to save the project.
9. Click **Configure Device** on the FlashPro GUI.
10. Click **Browse** and navigate to the location where the `PCIe_Demo_top.stp` file is located and select the file. The location for SmartFusion2 Advanced Development Kit board is:
    `<download_folder>\M2S150_PCIe_Control_Plane_DF\programmingFile`.
11. Click **Open**. The required programming file is selected and is ready to be programmed in the device.
12. Click **PROGRAM** to start programming the device. Wait until a message is displayed indicating that the **PROGRAM PASSED**.

### 2.3.3 Connecting the Board to the Host PC

The following steps describe how to connect the board to the Host PC.

1. After successful programming, power **OFF** the SmartFusion2 Advanced Kit board and shut down the Host PC.
   
   This demo is designed to run in any PCIe Gen 2 compliant slot. If the Host PC does not support Gen 2 compliant slot, the demo switches to Gen 1 slot.

2. Connect the **CON1 - PCIe Edge Card Ribbon** cable to Host PC **PCIe Gen 2** slot or **Gen 1** slot as applicable.

   **CAUTION**: Host PC must be powered OFF while inserting the PCIe Edge connector. If it is not, the PCIe device detection and selection of Gen 1 or Gen 2 slot may not occur properly. This is very dependent on the Host PC PCIe configuration. It is recommended that the Host PC is powered OFF before inserting the PCIe card.
The following figure shows the board setup for the Host PC in which SmartFusion2 Advanced Kit board is connected to the Host PC PCIe slot.

*Figure 10* • SmartFusion2 Advanced Development Kit Setup for Host PC

### 2.4 Running the Demo Design

This demo can run on both Windows and RedHat Linux OS.

- To run the demo on Windows OS GUI, Microsemi PCIe are provided. Refer to *Running the Demo Design on Windows*, page 11.
- To run the demo on Linux OS, native RedHat Linux drivers and command line scripts are provided. Refer to *Running the Demo Design on Linux*, page 18.
2.4.1 Running the Demo 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 SmartFusion2 Advanced Development Kit board.
3. Right-click **PCIe Device** > **Scan for hardware changes** in Device Manager.

*Figure 11 • Device Manager*

*Note:* If the device is still not detected, check whether or not the BIOS version in Host PC is the latest, and if PCIe is enabled in the Host PC BIOS.
2.4.1.1 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 12 • Update Driver Software*

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

   *Figure 13 • Browse for Driver Software*
3. Browse the drivers folder: M2S150_PCIE_Control_Plane_DF\Windows_64bit\Drivers\PCIe_Demo and click Next as shown in the following figure.

*Figure 14* • Browse for Driver Software Continued

4. The Windows Security dialog box is displayed. Click Install as shown in the following figure. After successful driver installation, a message appears. See Figure 16, page 13.

*Figure 15* • Windows Security

*Figure 16* • Successful Driver Installation
2.4.1.2 PCIe Demo GUI Installation

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

To install the GUI:

1. Extract the PCIe_Demo(GUI Installer.rar and locate the files at M2S150_PCIE_Control_Plane_DFGUI.
2. Double-click the setup.exe in the provided GUI installation (PCIe_Demo(GUI Installer\setup.exe). Apply default options as shown in the following figure.
3. To start the installation, click Next.

*Figure 17 • GUI Installation*

![GUI Installation]

4. Click Finish to complete the installation.

*Figure 18 • Successful GUI Installation*

![Successful GUI Installation]

5. Restart the Host PC.
2.4.1.3 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 SmartFusion2 Advanced Development Kit board.
2. Click **Scan for hardware changes** in Device Manager. Ensure that the board is switched on.

   *Figure 19 • Device Manager—PCIe Device Detection*

   ![Device Manager—PCIe Device Detection](image)

   **Note:** If a warning symbol is displayed on the **Microsemi PCIe** in the **Device Manager**, uninstall them and start from step1 of **Drivers Installation**, page 12.

3. Invoke the GUI from **ALL Programs > PCIe Control Plane Demo**. The following figure shows the PCIe Demo GUI window.

   *Figure 20 • PCIe Demo GUI*
4. 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 21 • Device Info*

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

*Figure 22 • 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.

*Figure 23 • Demo Controls—Continued*

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

*Figure 24 • Configuration Space*
8. 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 25 • PCIe BAR1 Memory Access*

![PCIe BAR1 Memory Access](image)

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

### 2.4.2 Running the Demo Design on Linux

1. Switch **ON** the power supply switch **SW7** on the SmartFusion2 Advanced Development Kit board.
2. Switch **ON** the Red Hat Linux Host PC.
3. Red Hat Linux Kernel detects the SmartFusion2 PCIe end point as Actel Device.
4. On Linux Command Prompt Use `lspci` command to display the PCIe info.
   ```bash
   # lspci
   ```

*Figure 26 • PCIe Device Detection*

![PCIe Device Detection](image)
2.4.2.1 Drivers Installation

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

1. Create the `sf2` directory under the `home` directory using the following command:
   # mkdir /home/sf2

2. Copy the `M2S150_PCIE_Control_Plane_DF/` design files folder under `/home/sf2` directory, which contains the Linux PCIe device driver files and Linux PCIe application utility files.

3. Copy the Linux PCIe Device Driver file (`PCIe_Driver.zip`) from `M2S150_PCIE_Control_Plane_DF/` design files folder.
   # cp -rf
   /home/sf2/M2S150_PCIE_Control_Plane_DF/Linux_64bit/Drivers/PCIe_Driver.rar
   /home/sf2
   # unrar -e PCIe_Driver.rar
   /home/sf2 directory must contain the `PCIe_Driver/inc` folders.

4. Execute `ls` command to display the contents of `/home/sf2` directory.
   # ls

5. Change to `inc` directory by using the following command:
   # cd /home/sf2/inc

6. Edit the `board.h` file for SmartFusion2 Advanced Development Kit board as shown in Figure 27, page 20.
   # vi board.h
   #define SF2_ADV_KIT
   #undef IGL2
   #undef SF2_DEV_KIT
   #undef SF2_EVAL_KIT

7. Enter `[:wq]` command to save the selected file.

8. Enter the following command to change the directory:
   # cd /home/sf2/PCIe_Driver

9. Enter the `make` command on Linux Command Prompt to compile the Linux PCIe device driver code.
   #make clean [To clean any *.o, *.ko files]
   #make

   The kernel module, `pci_chr_drv_ctrlpln.ko`, is created in the same directory.

10. Enter `insmod` command to insert the Linux PCIe device driver as a module.
    #insmod pci_chr_drv_ctrlpln.ko

**Note:** Root privileges are required to execute this command.
11. After successful Linux PCIe device driver installation, check `/dev/MS_PCI_DEV` got created by using the following Linux command:

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

**Note:** `/dev/MS_PCI_DEV` interface is used to access the SmartFusion2 PCIe end point from Linux user space.
2.4.2.2 Linux PCIe Application Compilation and PCIe Control Plane Utility Creation

1. Change to the `/home/sf2/` directory using the following command:
   
   ```
   # cd /home/sf2
   ```

2. Copy the Linux PCIe application utility file (`PCIe_App.zip`) from `M2S150_Control_Plane_DF/design files` folder.

   ```
   # cp -rf /home/sf2/M2S150_PCIe_Control_Plane_DF/Linux-64bit/Util/PCIe_App.rar /home/sf2
   # unrar -e PCIe_App.rar
   ```

   The `/home/sf2` directory must contain `PCIe_App/` folder along with `led_blink.sh` and `pcie_config.sh` scripts.

3. Execute `ls` command to display the contents of `/home/sf2` directory.

   ```
   # ls
   ```

4. Compile the Linux user space application `pcie_appln_ctrlpln.c` in `/home/sf2/PCIe_App` folder by using `gcc` command.

   ```
   # cd /home/sf2/PCIe_App
   # gcc -o pcie_ctrlplane pcie_appln_ctrlpln.c
   ```

   After successful compilation, Linux PCIe application utility `pcie_ctrlplane` creates in the same directory.

5. On Linux Command Prompt, run the `pcie_ctrlplane` utility as:

   ```
   #./pcie_ctrlplane
   ```

6. Help menu is displayed as shown in the following figure.

   ![Linux PCIe Application Utility](image)

   **Figure 29 • Linux PCIe Application Utility**
2.4.2.3 Execution of Linux PCIe Control Plane Features

2.4.2.3.1 LED Control

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

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

Figure 30 • Linux Command—LED Control

`led_blink.sh` contains the shell script code to perform the LED Walk ON, whereas Ctrl C exits the shell script and LED Walk turns OFF.

```
#sh led_blink.sh
```

Run the `led_blink.sh` shell script using the `sh` command.
2.4.2.3.2 DIP Switch Status

Dip Switch on SmartFusion2 Advanced Development Kit board has four electric switches to hold the device configurations. Linux PCIe utility reads the corresponding switches (ON/OFF) state.

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

Figure 31 • Linux Command—DIP Switch
2.4.2.3.3 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 32 • Linux Command—PCIe Configuration Space Display**
2.4.2.3.4 PCIe Link Speed and Width

Note: Root Privileges are required to execute this command.

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

Figure 33 • Linux Command—PCIe Link Speed and Width
Figure 34 • Linux Command—PCIe Link Speed and Width

```
root@localhost:~# lsusb
Vendor: Actel Corporation (ID 0726:5231) (product: SmartFusion2 SoC FPGA)
Device: Actel Corporation SoC FPGA (ID 0726:5232)
Kernel driver in use: libpci��
```

```
Kernel driver in use: libpci��
Subsystem: Actel Device 8888
  Control: I/O, NoBusMaster, SpecCycle, RmDevLoop, VgaSnoop, ParErr: Stopping, SERR: FastTxB, D1IssNtx
  Status: CapGMPo, UDF: FastSze, ParErr: DEVSEL=fast >TAAbort <TAAbort >SERR <PERR INT
  Latency: 0, Cache Line Size: 64 bytes
  Interrupt pin: A routed to IRQ 74
  Region 0: Memory at f8000000 (32-bit, non-prefetchable) [size=3M]
  Region 1: Memory at f4f80000 (32-bit, non-prefetchable) [size=64K]
  Capability list: [5] MSi: Enable/Disable-Count/1/1(Maxable- 16bit)
  Address: 88000000-90000000 Data: 40a
 Capabilities: [78] Power Management version 3
  Flags: PMe(Clk) DS1- DS2- AuxCurrent=Max PME(D0, D1- D3)- Shot=3cold
  Status: D2 NoSoftSts- PME-Enable: D0=D1= D2=0 D3=0
  Capabilities: [80] Express (v2) Endpoint, MSi 81
    DevCap: MaxPayload 256 bytes, PhantFunc 6, Latency LBs unlimited, L1 unlimited
    ExtTag- AddrtnBlk- AttnBlk- PwrBlk- RBE- PReset
    DevCtl: Report errors, Correctable- Non-Fatal= Fatal= Unsupported-
    RxBrdRdy ExtTag- PhantFunc- AuxPwr- NoSnoop- MaxPayload 128 bytes, MaxReadReq 512 bytes
    DevSts: CorrErr- UncorrErr- FatalErr- UnsugReq- AuxPwr- TransRm
    LinkCap: Port #1, Speed 5GT/s, Width x4, ASPM L0s L1, Latency L0 -44ns, L1 -36us
    ClockMHz- Surprise- LLACap- BwNet
    LinkCtl: ASPM Disabled- NCB 64 bytes Disabled- Retrain- ConnClk- ExtClk- ClockMHz- AutotEst- BMint- AutBMint-
    LinkSts: Speed 2.5GT/s, Width x4, TErr- Train- SlotClk= DLActive- BmMgmt- ApmMgmt-
    LinkDelay: Completion Timeout: 50us to 50ms, TimeoutDis-
    LinkTrg: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-, Selectable De-emphasis: 6dB
    Transmit Margin: Normal Operating Range, EnterModEvtCompliance- ComplianceDis-
    Compliance Dev-emphasis: 6dB
    LinkSts2: Current Dev-emphasis Level: -3.5dB
  Capabilities: [108 v3] Virtual Channel
    Caps: LPEVClk- SrClk=188MHz PATEntryBits=1
    Arb: Fixed- WRH22- WRH64- WRH128-
    Ctrl: ArbSelect=Fixed
    Status: InProgress-
    VC8: Caps: PFC-> Cap0- MaxTimeSlots=1 RejtSnoopTrans-
    Arb: Fixed- WRH22- WRH64- WRH128- WRH256-
    Ctrl: EnBl=0/0 ArbSelect=Fixed TCL=0/1
    Status: NegaPending- InProgress-
  Capabilities: [188 v3] Advanced Error Reporting
    USEtst: DLP- SDES- TLP- FC- CapTtl- CapTtlAb- UvxAst- RxFD- MalFLP- ERC- UnsugReq- ACSViol-
    USEvnt: DLP- SDES- TLP- FC- CapTtl- CapTtlAb- UvxAst- RxFD- MalFLP- ERC- UnsugReq- ACSViol-
    CAP: TxRcv- BxRcv- TxFD- BxFLP- TC- ERC- UnsugReq- ACSViol-
    CESTa: RxRx- RxBLP- BxBLP- RcvTimeout- NonFatalErr-
    CEMs: RxRx- RxBLP- BxBLP- RcvTimeout- NonFatalErr-
    AXRcp: First Error Pointer: 08, 06xCaps: 06xEn- CkhCaps- ChkEn-
  Detector driver in use: M_SPC_Driver
```

```
02:08:0.8 Ethernet controller: Broadcom Corporation BM857780 Gigabit Ethernet PCIe (rev 01)
```

```
root@localhost:~# lsusb
System: Intel Device 0488
```

```
DG0566 Demo Guide Revision 5.0
26
```
2.4.2.3.5 PCIe Interrupt Control (Enable/Disable) and Interrupt Counter

SmartFusion2 Advanced Development 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 SW1 push button.

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

_Figure 35 • Linux Command—PCIe Interrupt Control_

2.5 Conclusion

This demo describes how to access the PCIe EP and displays the device serial number feature of SmartFusion2 by implementing a low bandwidth control plane design with BFM simulation. It provides a GUI for easy control of PCIe EP device through Microsemi PCIe drivers for windows platform. It also provides a Linux PCIe application for easy control of PCIe EP device through Linux PCIe Device Driver.
Appendix: SmartFusion2 Advanced Development Kit Board

The following figure shows the SmartFusion2 Advanced Development Kit board.

*Figure 1* • SmartFusion2 Advanced Development Kit Board.