

Comet-A65 SOM Evaluation Kit



Demonstration Manual

FPGA

Contents

Chapter 1	Overview	4
Chapter 2	Examples For FPGA	5
2.1	Basic Nios V control demo for Temperature/ Power/ Fan	5
2.2	Board Information IP	8
2.3	Si5340B IP	12
2.4	Clock Controller demo for Si5340B	19
2.5	HDMI_out_RTL in Verilog	20
2.6	QTH_lvds_x4_x4	22
2.7	RTL_LPDDR4_AXI4_Test	25
2.8	SLVS_EC_RX_to_HDMI_IMX537 Test	28
2.9	SLVS_EC_RX_to_HDMI_IMX901 Test	31
Chapter 3	PCI Express Reference Design for Windows	34
3.1	PCI Express System Infrastructure	34
3.2	PC PCI Express Software SDK	35
3.3	PCI Express Software Stack	35
3.4	PCI Express Library API	40
3.5	PCIe Reference Design Fundamental	46



3.6	PCIe Reference Design - LPDDR4	51
Chapter 4	<i>PCI Express Reference Design for Linux</i>	62
4.1	PCI Express System Infrastructure	62
4.2	System Requirement.....	63
4.3	PC PCI Express Software SDK.....	63
4.4	PCI Express Software Stack	64
4.5	PCI Express Library API.....	66
4.6	PCIe Reference Design Fundamental.....	67
4.7	PCIe Reference Design - DDR4.....	73
Chapter 5	<i>Transceiver Verification</i>	83
5.1	Transceiver Test Code	83
5.2	Loopback Fixture.....	83
5.3	Testing by Transceiver Test Code	84
Chapter 6	<i>Additional Information</i>	90
6.1	Getting Help	90

Chapter 1

Overview

This Manual will introduce the various application demonstrations on **Comet A65 SoM Evaluation Kit**. These demonstrations cover most of the interfaces on the board. Let users familiarize using these interfaces of the board. Demonstrations according to FPGA fabrics are divided into three categories:

- Pure use of FPGA fabric resources (Chapter 2)
- PCIe reference design resource (Chapter 3&4)
- Transceiver Verification (Chapter 5)

Finally, to complete the following demonstration, user needs to install the following software in the computer:

- [Intel Quartus® Prime Pro Edition Software Version 25.3.1](#) or later.

Note: To run the demo batch file with the Nios V CPU of the demonstration on Windows system, user need to install the Ashling RiscFree IDE together with Quartus software, then you can run the batch file.

Chapter 2

Examples For FPGA

This chapter provides examples of advanced designs implemented by RTL or Qsys on the **Comet A65 SoM Evaluation Kit**. These reference designs cover the features of peripherals connected to the FPGA, such as LPDDR4, temperature monitor, PLL clock setting and Power monitor. All the associated files can be found in the directory **\Demonstrations\FPGA** of Comet A65 SoM Evaluation Kit System CD.

2.1 Basic Nios V control demo for Temperature/ Power/ Fan

This demonstration shows how to use the Nios V processor to measure the power consumption based on the built-in power measure circuit. The demonstration also includes a function of monitoring system temperature with the on-board temperature sensor and monitoring fan rotation speed.

■ System Block Diagram

Figure 2-1 shows the system block diagram of this demonstration. The 12V input power monitor, temperature sensor and fan controller connected to the system MAX10 FPGA and controlled by internal logic circuits. All collected status data or control commands will be sent to the SPI slave block so that the Agilex FPGA can read it through the SPI interface.

In the Agilex FPGA, an SPI master IP (implemented by HDL) will read these external sensor data from the MAX10 FPGA through SPI interface. The Nios system will read this information through PIO controllers.

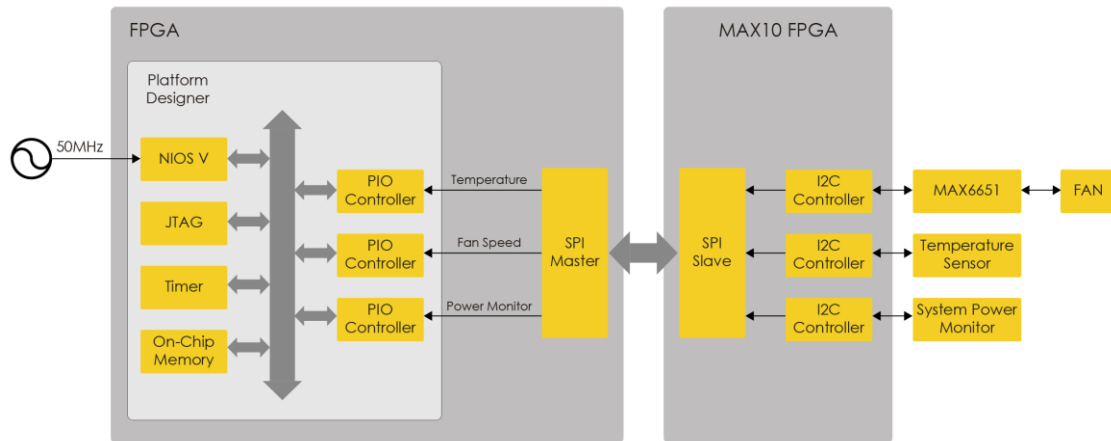


Figure 2-1 Block Diagram of the Nios V Basic Demonstration

The system provides a menu in command line window, as shown in **Figure 2-2** to provide an interactive interface. With the menu, users can perform the test for the board info sensor. Note, pressing 'ENTER' should be followed with the choice number.

```

C:\WINDOWS\system32\cmd... x + v
Transfer rate: 514 KB/sec, 22129 bytes/write.
[Inferior 1 (Remote target) detached]
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Comet A65 [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

===== Agilex Demo Program =====
[0] Display Board Info
Input your chioce:0
==== Temperature ====
    FPGA: 36*C
    Board 1: 32*C
    Board 2: 34*C
    Power: 25*C

==== Fan ====
    Fan RPM: 2580

==== Power (12V) Monitor ====
    Voltage      = 4.975 V
    Current      = 1.041 A
    Power        = 5.179 W
==== Core Power Monitor ====
    Voltage      = 0.792 V
    Current      = 0.750 A
    Power        = 0.594 W
Display Board Info Test:PASS
===== Agilex Demo Program =====
[0] Display Board Info
Input your chioce:|

```

Figure 2-2 Menu of Demo Program

In board info test, the program will display local temperature, remote temperature, 12V input power monitor and fan rotation speed. The remote temperature is the FPGA temperature, and the local temperature is the board temperature where the temperature sensor located. The board also provides circuitry to monitor important voltages, currents and power consumption in real time.

■ **Demonstration File Location**

- Hardware project directory: Board_Info_NiosV
- Bitstream used: golden_top.sof
- Software project directory: Board_Info_NiosV\software
- Demo batch file: Board_Info_NiosV\demo_batch\test.bat

■ **Install Ashling RiscFree IDE**

Before executing this demo with RISC-V core, users need to install Ashling RiscFree IDE for Intel® FPGAs to ensure that this batch file can be executed correctly. The file name should be *RiscFreeSetup-<Quartus Version>-windows.exe*. Users can find it under the [Quartus Pro download page](#) (Individual Files tab).

■ **Demonstration Setup and Instructions**

1. Make sure Quartus Prime is installed on the Host PC.
2. Power on the FPGA board.
3. Use the USB Cable to connect your PC and the FPGA board and install USB Blaster III driver if necessary.
4. Execute the demo batch file “test.bat” under the batch file folder: Board_Info_NiosV\demo_batch.
5. After the Nios V program is downloaded and executed successfully, a prompt message will be displayed in command line window.
6. For temperature, power monitor and fan test, please input key ‘0’ and press ‘Enter’ in the command line window, as shown in **Figure 2-3**.

```
C:\WINDOWS\system32\cmd.exe
Transfer rate: 514 KB/sec, 22129 bytes/write.
[Inferior 1 (Remote target) detached]
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Comet A65 [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

===== Agilex Demo Program =====
[0] Display Board Info
Input your choice:0
===== Temperature =====
      FPGA: 36°C
      Board 1: 32°C
      Board 2: 34°C
      Power: 25°C

===== Fan =====
      Fan RPM: 2580

===== Power (12V) Monitor =====
      Voltage      = 4.975 V
      Current      = 1.041 A
      Power        = 5.179 W
===== Core Power Monitor =====
      Voltage      = 0.792 V
      Current      = 0.750 A
      Power        = 0.594 W
Display Board Info Test:PASS
===== Agilex Demo Program =====
[0] Display Board Info
Input your choice:|
```

Figure 2-3 Board Info Demo

2.2 Board Information IP

This section will introduce an IP which can be placed in the Agilex FPGA and allows users to obtain board status information such as power, temperature status and fan speed on the Comet A65 Evaluation board.

The Comet A65 Evaluation board provides several sensors to monitor the status of the board, such as FPGA temperature, board power monitor, and fan speed status. These interfaces are connected to the system MAX FPGA on the board. The logic in the system MAX FPGA will automatically read the status values of these sensors and store them in the internal register. As shown in [Figure 2-4](#), there is an SPI slave IP in the system MAX FPGA will read the value of the board status from these registers and it can be output to SPI master logic via SPI interface.

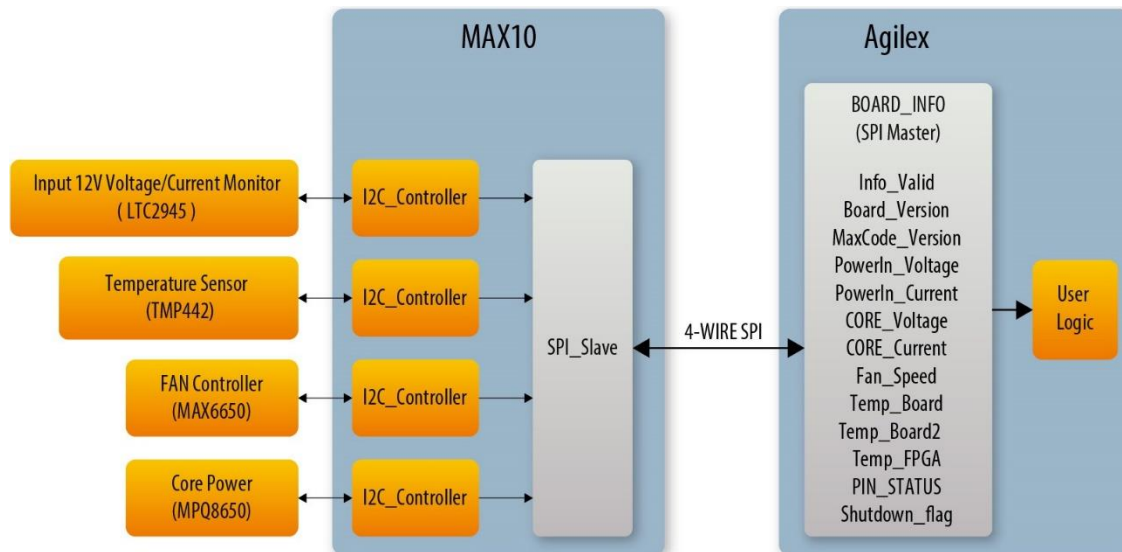


Figure 2-4 Block diagram of the fan speed control demonstration

User can place a board information IP (BOARD_INFO.v ; SPI master) provided by Terasic in the Agilex FPGA, the board status can be obtained via SPI interface from the system MAX FPGA and output to user logic.

The board information IP can be obtained from the following path in the system CD:
Demonstration/FPGA/Board_info_RTL/board_information_ip/BOARD_INFO.v

Figure 2-5 shows the input and output pins of the board information IP. Detailed pin descriptions and functions can be obtained from **Table 2-1**. The user only needs to provide the IP 50Mhz clock and the reset control signal. The IP will automatically communicate with the system MAX FPGA to get the board status value via the SPI interface. When the logic level of the Info_Valid signal is from low to high, it means that the board status has been updated and can be used.

Finally, **Figure 2-6** shows the status of the IP during execution.

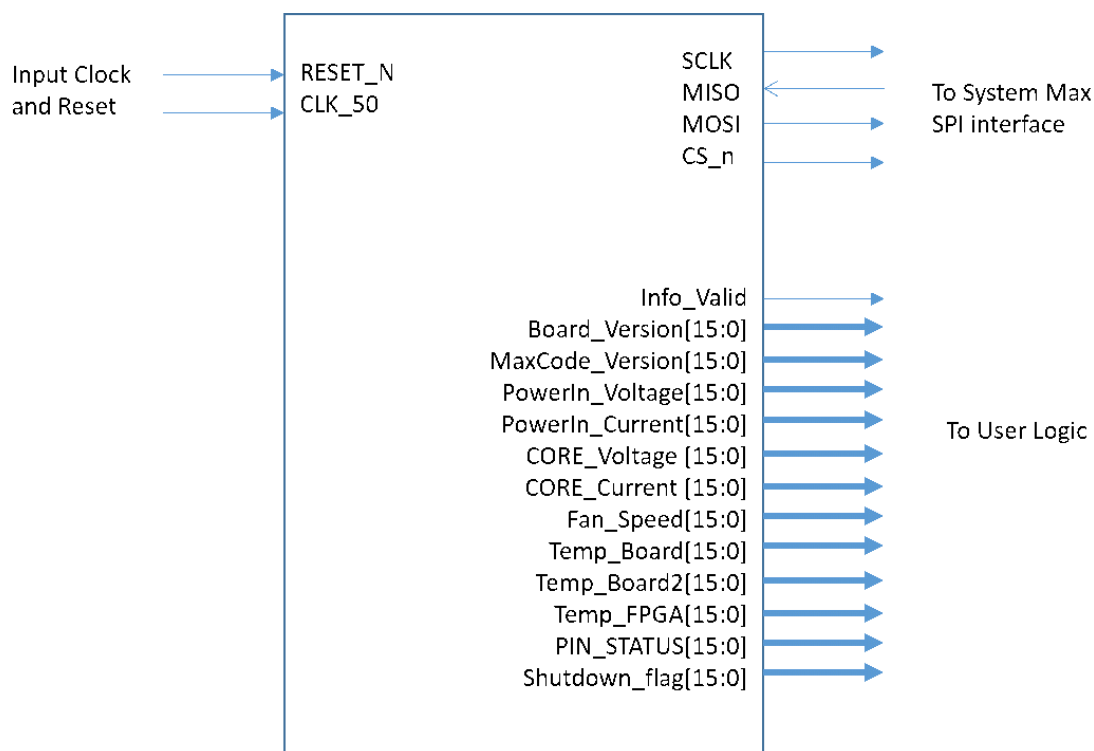


Figure 2-5 Pin out of the board information IP

Table 2-1 Board information IP input and output ports

Port Name	Direction	Width(Bit)	Description
CLK_50	Input	1	Clock input for IP, please input 50Mhz clock.
RESET_N	Input	1	Reset signal for IP, reset all logic.
I2C_SDA	BIR	1	Master I2C data . Please connect this signal to the FPGA_I2C_SDAT pin.
I2C_SCL	Output	1	Master I2C clock, I2C master output to slave. Please connect this signal to the FPGA_I2C_SCLK pin.
Info_Valid	Output	1	Information valid, logic high indicates board status updated ready.
Board_Version	Output	16	This information indicates the version of the DE25-STANDARD board. It will be started at 0x000A.
MaxCode_Versi on	Output	16	This information indicates the version of the System MAX 10 FPGA code. It will be started at 0x0001.

PowerIn_Voltage	Output	16	12V Voltage, the unit of the output value is mV. If the PowerIn_Voltage output value is "12050" that means 12.05V for 12V power
PowerIn_Current	Output	16	12V Current, the unit of the output value is mA. If the PowerIn_Current output value is "1000" that means 1A for 12V power
Fan_Speed	Output	16	First fan speed of the board. The unit of the output value is RPM.
Temp_Board	Output	16	First ambient temperature of the development board. The unit of the output value is Celsius.
Temp_Board2	Output	16	Second ambient temperature of the development board. The unit of the output value is Celsius.
Temp_FPGA	Output	16	Core FPGA temperature of the development board. The unit of the output value is Celsius.
CORE_Voltage	Output	16	CORE Voltage, the unit of the output value is mV. If the CORE_Voltage output value is "790" that means 0.79V for CORE power
CORE_Current	Output	16	CORE Current, the unit of the output value is 10mA. If the CORE_Current output value is "100" that means 1A for CORE power
Shutdown_flag	Output	16	BIT6~15 : Reserved to 0. BIT5: 1 ,when Input 12V power >=80W BIT4: 1 ,when CORE IC Temperature >=95°C BIT3: 1 ,when CORE Current >=45A BIT2: 1 ,when FPGA Temperature >=95°C BIT1: 1 ,when Board2 Temperature >=95°C

			BIT0: 1 ,when Board Temperature >=95°C
PIN_STATUS	Output	16	BIT8~15 : Reserved to 0. BIT7: FAN_ALERT_n , When the fan speed is abnormal, this bit is 0. BIT6: Reserved to 1. BIT5: When shutdown occurs, this bit is 0. BIT4: Reserved to 1. BIT3: Reserved to 0. BIT2: FPGA_CONF_DONE ,FPGA Configure success, this bit is 1. BIT1: Reserved to 1. BIT0: Reserved to 1.

BOARD_INFO_i Board_Version[15..0]	000Bh
BOARD_INFO_i MaxCode_Version[15..0]	0004h
BOARD_INFO_i CORE_Voltage[15..0]	792
BOARD_INFO_i CORE_Current[15..0]	62
BOARD_INFO_i Fan_Speed[15..0]	3180
BOARD_INFO_i Temp_FPGA[15..0]	40
BOARD_INFO_i Temp_Board[15..0]	36
BOARD_INFO_i Temp_Board2[15..0]	38
BOARD_INFO_i PIN_STATUS[15..0]	00F7h
BOARD_INFO_i Shutdown_flag[15..0]	000Bh
BOARD_INFO_i Temp_POWER[15..0]	30
BOARD_INFO_i PowerIn_Voltage[15..0]	4975
BOARD_INFO_i PowerIn_Current[15..0]	1141
BOARD_INFO_i Info_Valid	

Figure 2-6 Waveform of the board status output

2.3 Si5340B IP

There is a SKYWORKS Si5340B clock generator on the Comet A65 Evaluation board can provide adjustable frequency reference clock (See [Figure 2-7](#) for XCVR 1B/4A/4B). The Si5340B clock generator can output differential frequencies though I2C interface configuration. This section will show you how to use FPGA RTL IP to configure each Si5340B PLL and generate users desired output frequency to each peripheral.

The Si5340B control IP is in the System CD folder:

"\\Demonstrations\FPGA\Si5340B_clock_controller\Si5340B_IP"

Developers can use the IP directly in their Quartus top. Developers can refer to the example in Demonstrations/Si5340B_Clock_Controller folder. This example shows how to instantiate the IP in Quartus top project.

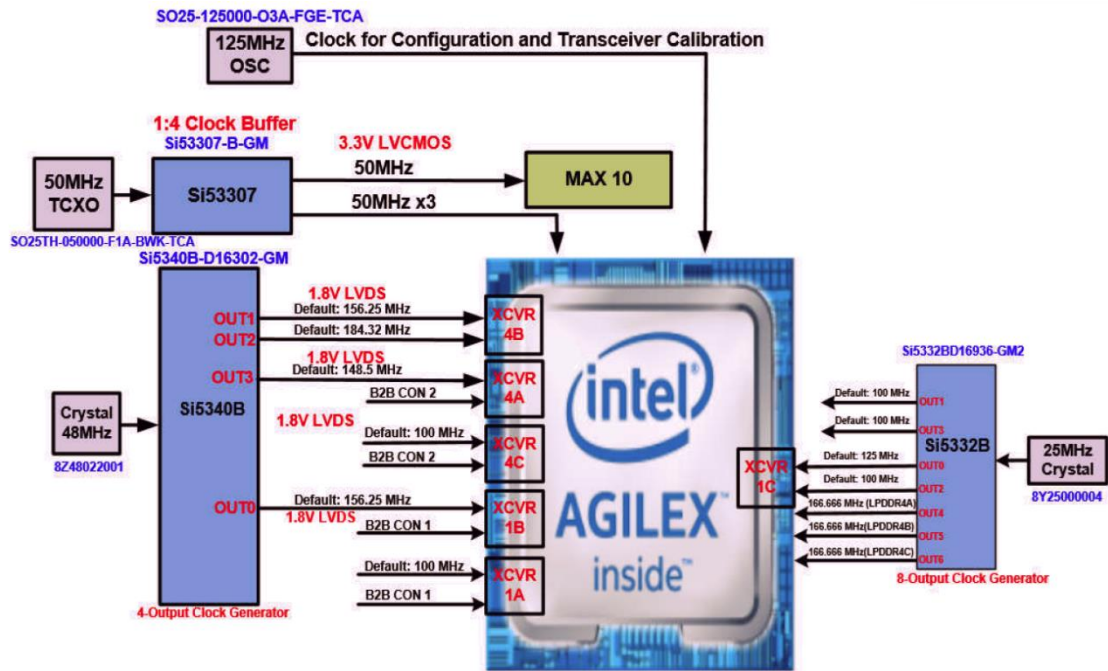


Figure 2-7 Clock tree of the Comet A65 Board

■ Si5340B IP Block Diagram

Figure 2-8 shows the Block Diagram of Si5340B IP. The Si5340B configuration data is stored in a ROM. The ROM contains register data to configure Si5340B via I2C interface. The register data original come from SKYWORKS ClockBuilder Pro Utility. The ROM content is initialized by a Memory Initialization File (.mif) file. The IP is located in the System CD folder:

CD\Demonstration\FPGA\Si5340B_Clock_Controller\Si5340B_IP

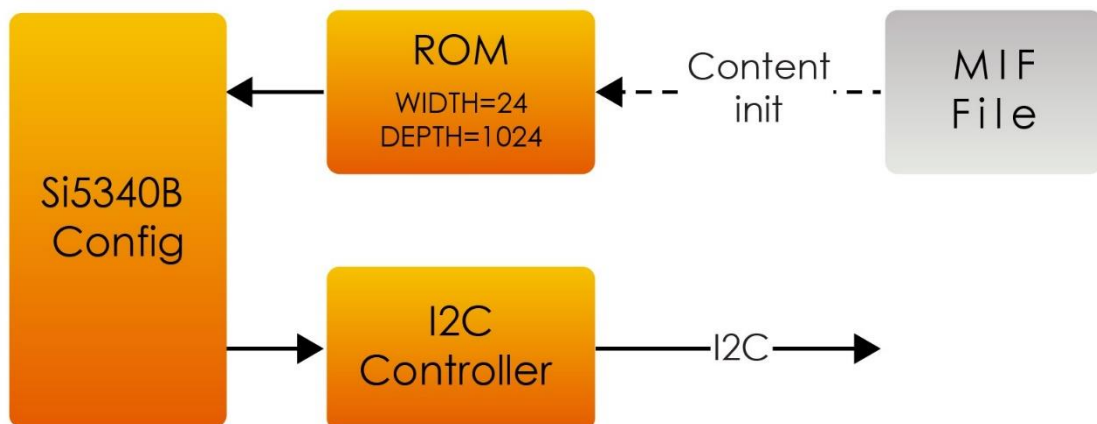


Figure 2-8 Si5340B IP Block Diagram

■ .MIF File Generation

Figure 2-9 shows the flow to generate the .mif Memory Initialization File which is used to initialize ROM in Si5340B IP. First, users have to download SKYWORKS ClockBuilder Pro software utility. Then launch ClockBuilder Pro and open Terasic Si5340B golden project. In ClockBuilder Pro, users modify desired output frequency and export register setting file (.txt). Finally, use Terasic ROM_MIF.exe tool to convert the register setting file (.txt) to a ROM initialization file .mif. The ROM_MIF.exe tool and Si5340B golden project are located in the System CD folder:

CD\Demonstration\FPGA\Si5340B_Clock_Controller\TXT2MIF

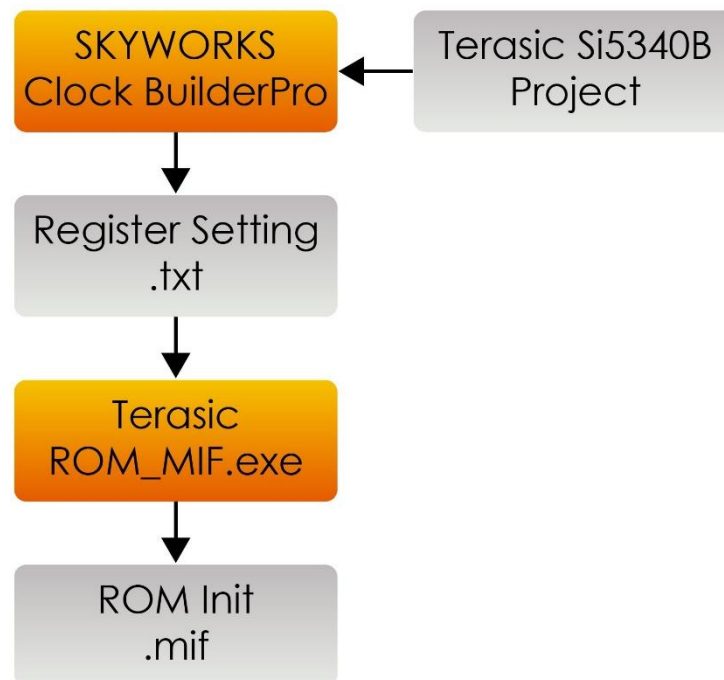


Figure 2-9 MIF Generation Flow

■ Step by Step to Update Register ROM in Si5340B IP

If the Si5340B built-in frequencies are not users' desired, users can refer to the below steps to the modify register parameter settings (stored in ROM) to make Si5340B to output desired frequencies.

1. Firstly, download ClockBuilder Pro Software (See **Figure 2-10**), which is provided by SKYWORKS. This tool can help users to set the Si5340B's output frequency of each channel through the GUI interface, and it will automatically calculate the Register parameters required for each frequency. ClockBuilder Pro 4.11 is recommended. The tool download link:

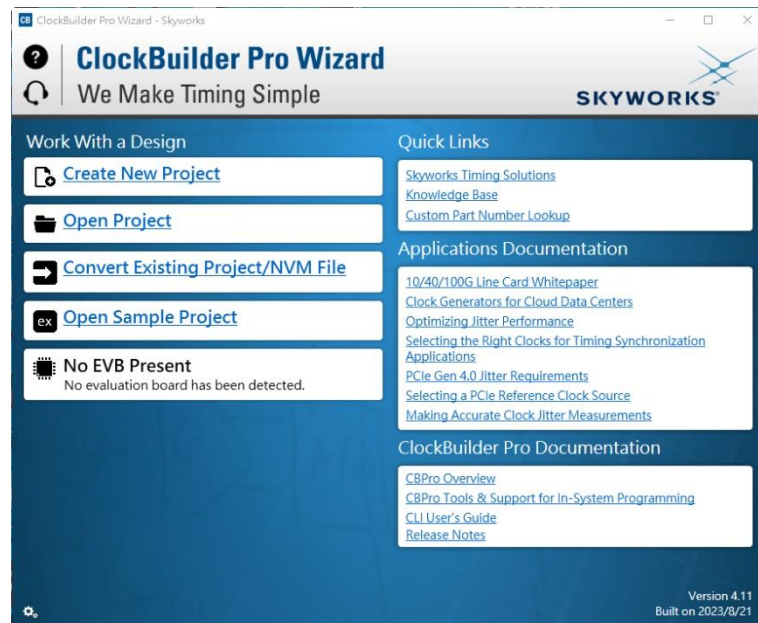


Figure 2-10 ClockBuilder Pro Wizard

2. Launch ClockBuilder Pro, click “Open Project” to open Terasic Si5340B Golden project Terasic_Si5340B.slabtimeproj as shown in **Figure 2-11**. The Si5340B project is located in the system CD folder:

CD\Demonstration\FPGA\Si5340B_Clock_Controller\TXT2MIF



Figure 2-11 Open Project in CBPro

- As shown in **Figure 2-12**, click “Outputs” to open Output Clocks page.

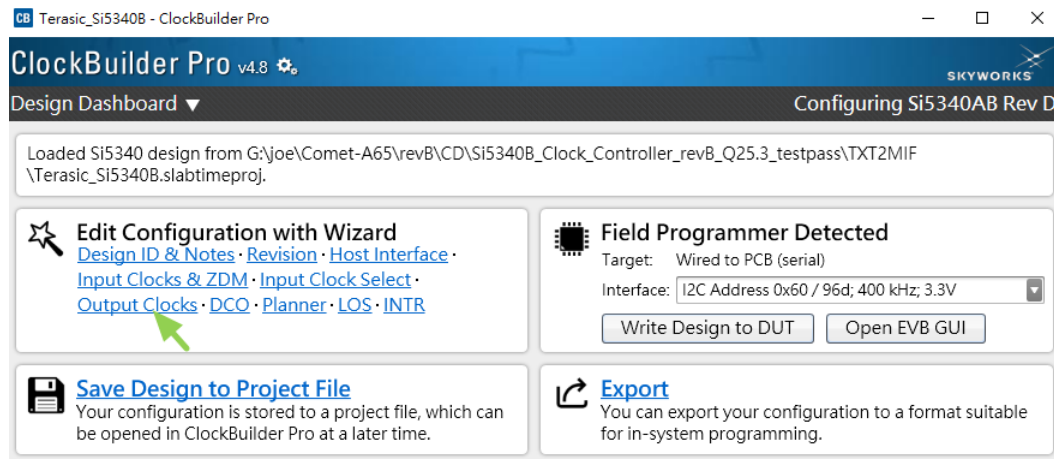


Figure 2-12 Click “Outputs” in CBPro

- In the Output Clocks page, specified desired output frequencies and click “Finish” button as shown in **Figure 2-13**.

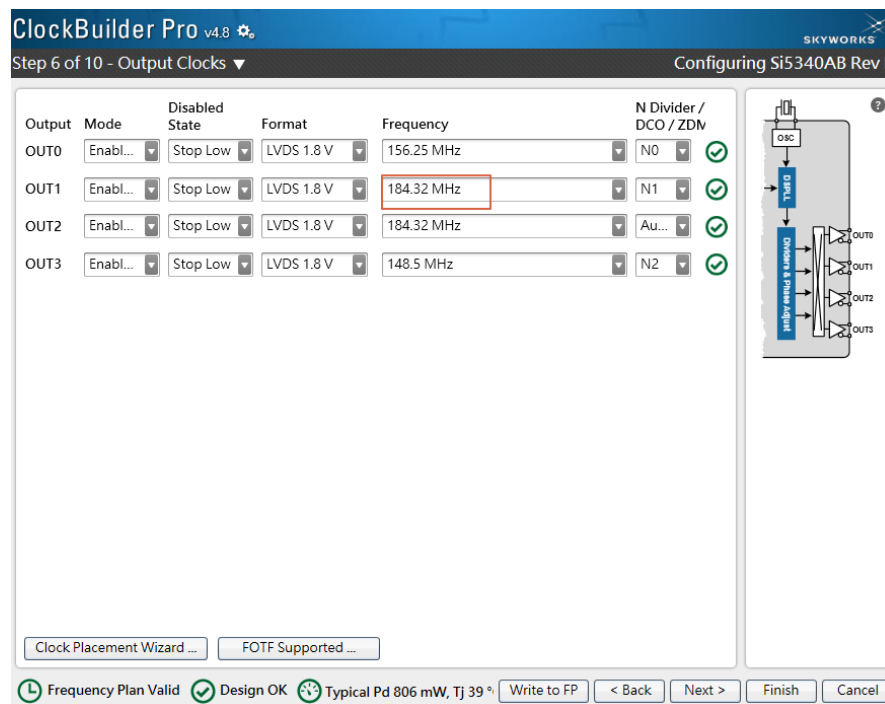


Figure 2-13 Specify Output Clock Frequencies on CBPro

- As shown in **Figure 2-14**, click “Export” in ClockBuilder Pro to open Si5340 Export Dialog.

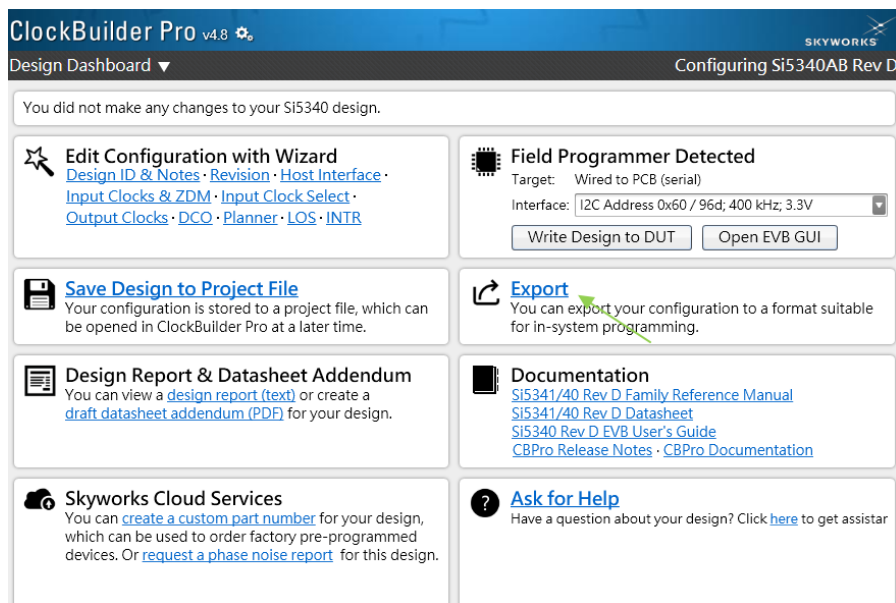


Figure 2-14 Click “Export” in CBPro

6. In the Si5340 Export dialog, select “Register File” tab. Then, select “Comma Separated Values (CSV) File”, and check both items “include summary header” and “include pre- and post-write control register writes” as shown in **Figure 2-15**. Finally, click “Save to File...” button to save file as **Si5340B-Registers.txt**.

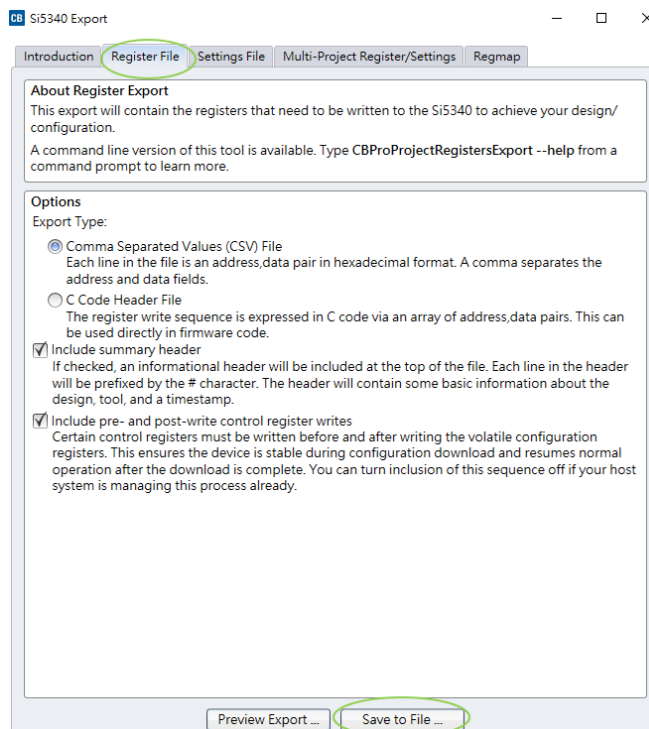


Figure 2-15 Save Register Files

7. Launch Terasic ROM_MIF utility as shown in **Figure 2-16**. Click “Generate .MIF” button and select **Si5340B-Registers.txt** file generated in previous step, then **Si5340B-Registers.mif** is generated.

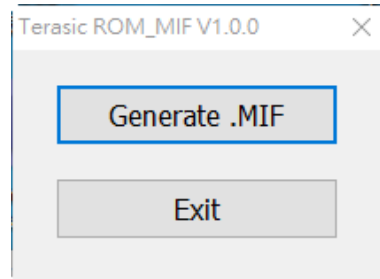


Figure 2-16 Terasic ROM_MIF utility

8. Use Quartus Pro to open ROM IP file (si5340B_rom.ip) in Si5340B IP. Select “Mem Init” tab, and click file name browse button to select the previous generated file **Si5340B-Registers.mif**. Then, click “Generate HDL...” button to regenerate ROM IP with new content. The ROM IP file is located in the System CD folder:

CD\Demonstration\FPGA\Si5340B_Clock_Controller\Si5340B_IP

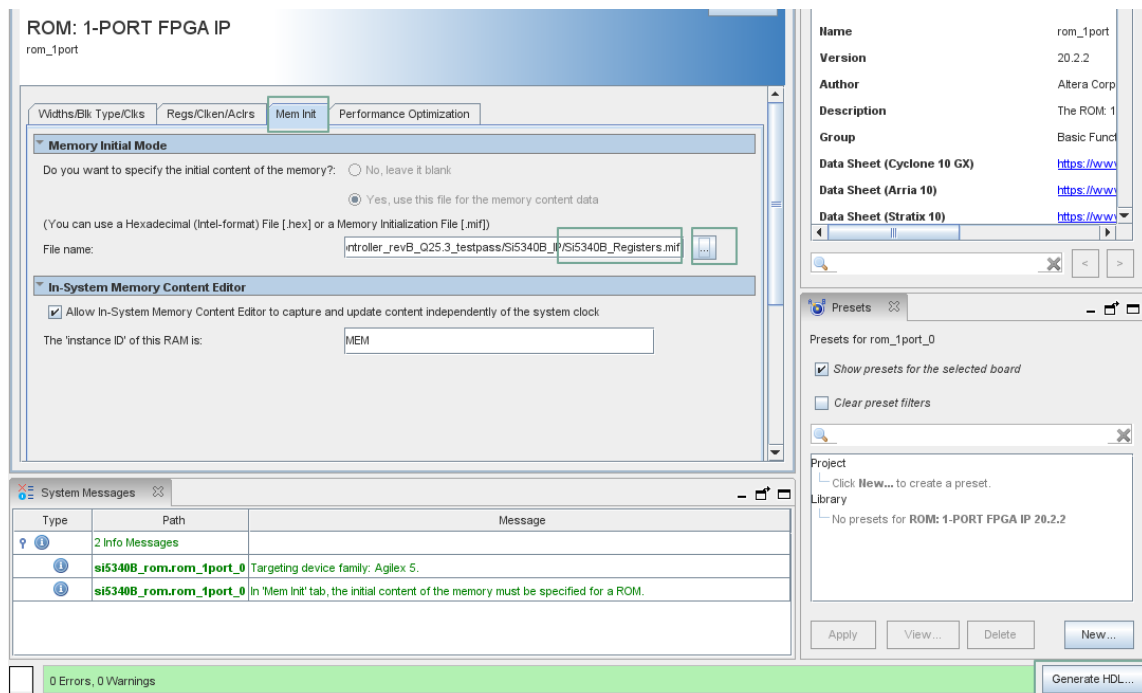


Figure 2-17 .mif file setting in ROM IP

2.4 Clock Controller demo for Si5340B

This demonstration shows how to use Si5340B Control IP (written by Verilog) to configure Si5340B clock generator to generate the desired frequency, and using Quartus SignalTap II tool to verify whether the output frequency of Si5340B is correct.

■ System Block Diagram

Figure 2-18 shows the system block diagram of this demonstration. The Si5340B clock generator is controlled by Si5340B IP through the I2C bus. In this demonstration, Si5340B OUT1 (map to GTSR4B_REFCLK_156M25_p) frequency is changed from 156.25MHz to 184.32MHz when users press the Comet A65 Carrier board's USER BUTTON to active the Si5340B reconfigure process. Press Comet A65 SOM's BUTTON will reset Si5340B, and the Si5340B will output default frequency 156.25MHz.

A clock frequency measurement module is used to measure the frequency of GTSR4B_REFCLK_156M25_p input clocks. The measured frequencies are displayed in SignalTap II tool. Users can use the SignalTap II to observe the measured clock frequency to verify whether the Si5340B output frequency is correct

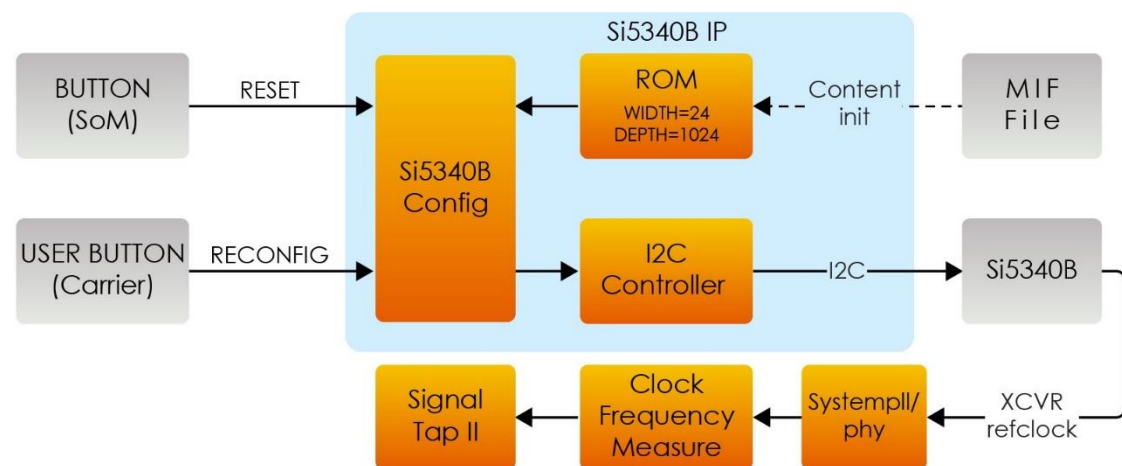


Figure 2-18 Block Diagram of the Si5340 Clock Controller Demonstration

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition

■ Demonstration File Locations

- Hardware project directory: Si5340B_Clock_Controller

- Bitstream used: golden_top.sof
- Demo batch file: Si5340B_Clock_Controller\demo_batch\test.bat

■ Demonstration Setup and Instructions

- Make sure Quartus Pro v25.3 is installed on the host PC.
- Use the Type-C USB Cable to connect your PC and the FPGA board, install USB Blaster III driver if necessary.
- Power on the Comet A65 Evaluation board.
- Execute the demo batch file “test.bat” under the batch file folder: Si5340B_Clock_Controller \demo_batch
- Using Quartus Pro software to open the SignalTap II file “stp1.stp” under the batch file folder: Si5340B_Clock_Controller \demo_batch.
- Switch to the SignalTap II window to observe the clock registers and verify the frequency. GTSR4B_REFCLK_156M25_p shows default output clock frequency 184,320,000 Hz, as shown in **Figure 2-19**.

ftGTSR4B_REFCLK_156M25_Si5340_out1[FREQ[31..0]]	184318265
Si5340B_IP_inst clk_ready	

Figure 2-19 Signal Tap II shows 184.32MHz

- Press Comet A65 Carrier board’s USER BUTTON, to start Si5340 reconfigure process. GTSR4B_REFCLK_156M25_p shows 184,320,000Hz as shown in **Figure 2-19**.
- Press Comet A65 SoM’s BUTTON to reset Si5340. GTSR4B_REFCLK_156M25_p shows 156,250,000Hz (default frequency) as shown in

Pre-Syn	ftGTSR4B_REFCLK_156M25_Si5340_out1[FREQ[31..0]]	156248504
Pre-Syn	Si5340B_IP_inst clk_ready	

Figure 2-20 SignalTap II shows 156.25MHz

2.5 HDMI_out_RTL in Verilog

Comet A65 Evaluation board system CD offers HDMI (DVI 1.0-compliant) test with its test code written in Verilog HDL. That is, the output test for the TFP410 IC includes I2C configuration and the ability to output 1080p@60 with the specified Color Bar.

■ System Block Diagram

Figure 2-21 shows the function block diagram of this demonstration. The VPG (video pattern generator) uses 50MHz in by PLL to generate 148.5MHz to do input. The VPG generates a 1080p@60 video timing color bar, which is sent to the TFP410 and then displayed on the HDMI monitor.

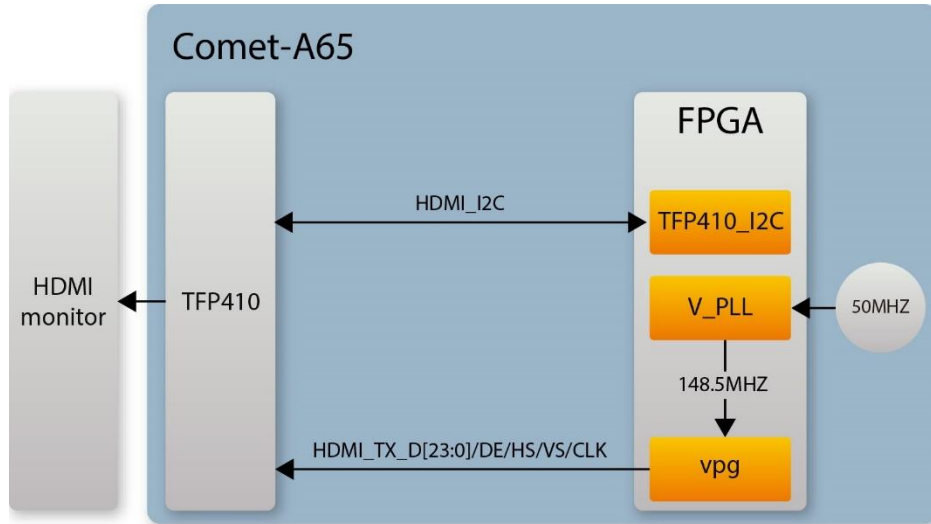


Figure 2-21 Block Diagram of the HDMI TX Demonstration

For the TFP410 to output video, the TFP410_I2C module must be used to configure the device via I2C by setting bit 0 of register 0x08 (CTL_1_MODE) to 1, enabling normal operation.

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: HDMI_out_RTL
- Bitstream used: golden_top.sof

■ Demonstration Batch File

Demo Batch File Folder: HDMI_out_RTL\demo_batch.

The demo batch file includes following files:

- Demo Batch File : test.bat
- FPGA Configure File: golden_top.sof

■ Demonstration Setup and Instructions

- Make sure both Quartus Pro and UBII driver are installed on the host PC.
- Connect the HDMI cable to J13 (HDMI Out) on the Comet A65 Evaluation

board.

- Use the Type-C USB Cable to connect your PC and the FPGA board, install USB Blaster III driver if necessary.
- Power on the Comet A65 Evaluation board.
- Execute the demo batch file “ test.bat” from the directory
- HDMI_out_RTL\demo_batch.
- 1080p@60 color bar should be visible on the monitor, as shown in **Figure 2-22**.

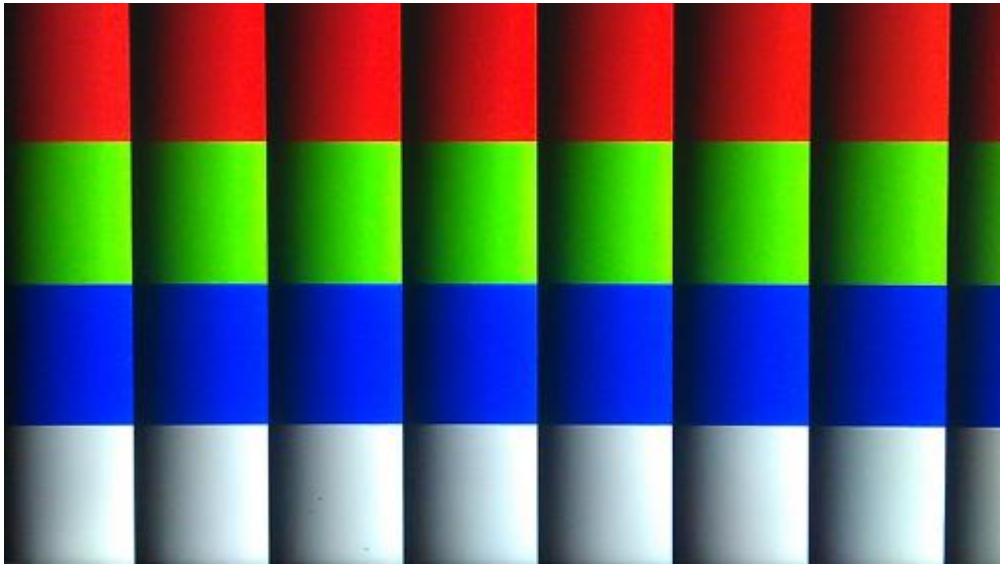


Figure 2-22 The video pattern in the HDMI TX demonstration

2.6 QTH_lvds_x4_x4

This demo demonstrates the LVDS TX-RX loopback transmission using the two groups of 4 lanes each arranged on the carrier board at connector QTH-030 (J3).

■ System Block Diagram

In the project, there are two groups (lvds_top_inst and lvds_bottom_inst) of 4-Lane LVDS loopbacks, and each needs to be compiled separately. **Figure 2-23** Show the block diagram of the 4-Lane LVDS TX-RX loopback.

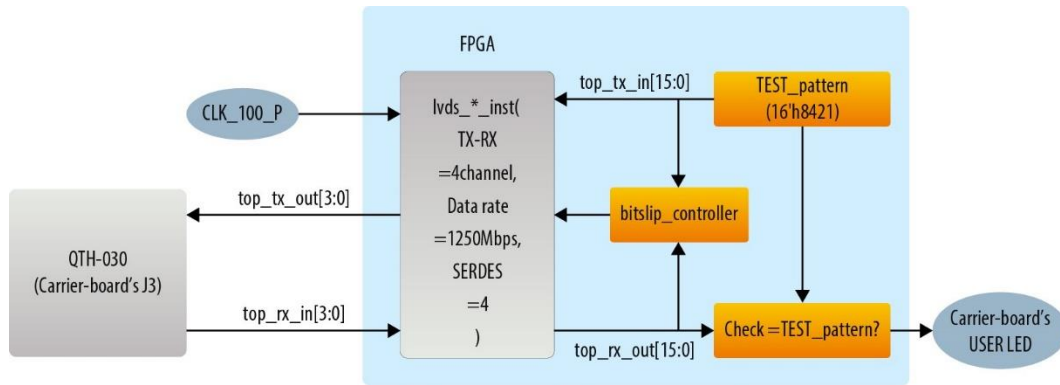


Figure 2-23 Block Diagram of the 4-Lane LVDS TX-RX Loopback

For hardware setup, a loopback board needs to be prepared as shown in **Figure 2-24**. The main requirement is to connect pin5-6, pin7-8, pin11-12, pin13-14, and so on. .

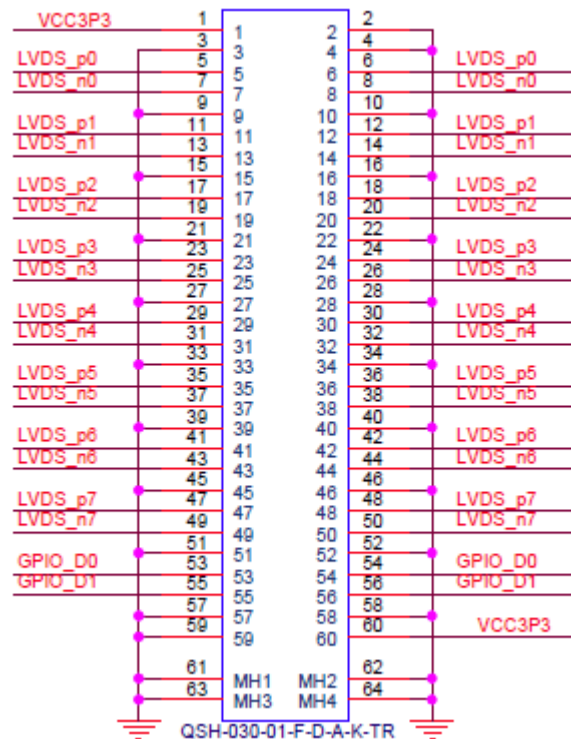


Figure 2-24 loopback board

As shown in the block diagram, a 16-bit test pattern (0x8421) is provided as input to the LVDS instance. After the LVDS is activated, a 16-bit data output will appear at the lvds_inst RX output (rx_out). This output is expected to be 0x8421. When the comparison is successful, the USER_LED on the carrier board will blink. At the same time, both the tx_in and rx_out data are sent to the bitslip_controller. The function of the bitslip_controller is to compare whether tx_in and rx_out have the same value and to detect if there is any bit shift in rx_out. If a bit misalignment is detected, it will generate a “1” pulse to the rx_bitslip_ctrl port of the LVDS instance to perform a bit-shift adjustment on the RX side. This project needs to be compiled separately for two groups.

When you open the project, you can select the compiler option at the top level.

1. `define ENABLE_QTH_LVDS_TOP will compile the **lvds_top_inst** group. The corresponding FPGA pins and LVDS IP byte-pin settings are shown in **Figure 2-25**.

	TX		TX		TX		TX		RX		RX		RX		RX	
Lane	channel	QTH pin	FPGA pin	FPGA bank	LVDS IP byte	LVDS IP pin	channel	QTH pin	FPGA pin	FPGA bank	LVDS IP byte	LVDS IP pin	channel	QTH pin	FPGA pin	FPGA bank
0	TX0	QTH_30_p0	BW89	2AB	2	`0405`	RX0	QTH_30_p1	BE79	2AT	4	`0809`	RX1	QTH_30_p5	BE96	2AT
1	TX1	QTH_30_p4	BF93	2AT	4	`0203`	RX2	QTH_30_p11	BF86	2AT	4	`0405`	RX3	QTH_30_p13	BF75	2AT
2	TX2	QTH_30_p10	BR81	2AB	2	`0607`										
3	TX3	QTH_30_p12	BR89	2AB	2	`0001`										

Figure 2-25 ENABLE_QTH_LVDS_TOP Parameters

2. `define ENABLE_QTH_LVDS_BOTTOM will compile the **lvds_bottom_inst** group. The corresponding FPGA pins and LVDS IP byte-pin settings are shown in **Figure 2-26**.

	TX		TX		TX		TX		RX		RX		RX		RX	
Lane	channel	QTH pin	FPGA pin	FPGA bank	LVDS IP byte	LVDS IP pin	channel	QTH pin	FPGA pin	FPGA bank	LVDS IP byte	LVDS IP pin	channel	QTH pin	FPGA pin	FPGA bank
0	TX0	QTH_30_p2	BM81	2AB	3	`0607`	RX0	QTH_30_p3	BH89	2AB	3	`0203`	RX1	QTH_30_p7	BK89	2AB
1	TX1	QTH_30_p6	BW78	2AB	2	`1011`	RX2	QTH_30_p9	BF92	2AB	3	`0809`	RX3	QTH_30_p15	BH81	2AB
2	TX2	QTH_30_p8	BR78	2AB	2	`0809`										
3	TX3	QTH_30_p14	BM78	2AB	3	`1011`										

Figure 2-26 ENABLE_QTH_LVDS_BOTTOM Parameters

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: QTH_lvds_x4_x4
- Bitstream used: golden_top_top.sof and golden_top_bottom.sof

■ Demonstration Batch File

Demo Batch File Folder: QTH_lvds_x4_x4\demo_batch

The demo batch file includes following files:

- Demo Batch File1 : test_top.bat (golden_top_top.sof)
- Demo Batch File2 : test_bottom.bat (golden_top_bottom.sof)

■ Demonstration Setup

- Make sure both Quartus Pro and UBIll driver are installed on the host PC.
- Insert your loopback board into the J3 connector on the Comet A65 Carrier board.
- Connect the HDMI cable to J13 (HDMI Out) on the Comet A65 Evaluation board.
- Use the Type-C USB Cable to connect your PC and the FPGA board, install USB Blaster III driver if necessary.
- Power on the Comet A65 Evaluation board.

- Execute the demo batch file “test_top.bat” from the directory QTH_lvds_x4_x4\demo_batch.
- Check the USER LED on the carrier board; if it is blinking, it indicates that the top LVDS data is correct.
- Then run “test_bottom.bat” and check the USER LED on the carrier board; if it is blinking, it indicates that the bottom LVDS data is correct.

2.7 RTL_LPDDR4_AXI4_Test

This demonstration performs a memory test function using RTL code on the three LPDDR4 modules of Comet A65 SoM board. The LPDDR4-A module is controlled by FPGA fabric in this demo. The memory size of each LPDDR4 used in this test is 4GB.

■ Function Block Diagram

Figure 2-27 shows the function block diagram of this demonstration. There are three LPDDR4 controllers IP (LPDDR4A, LPDDR4B and LPDDR4C) in this project. All of the controllers use 166.666MHz as a reference clock. The test program will write data into LPDDR4, after writing 4GB capacity, it will read the values from LPDDR4 and check whether there is any error.

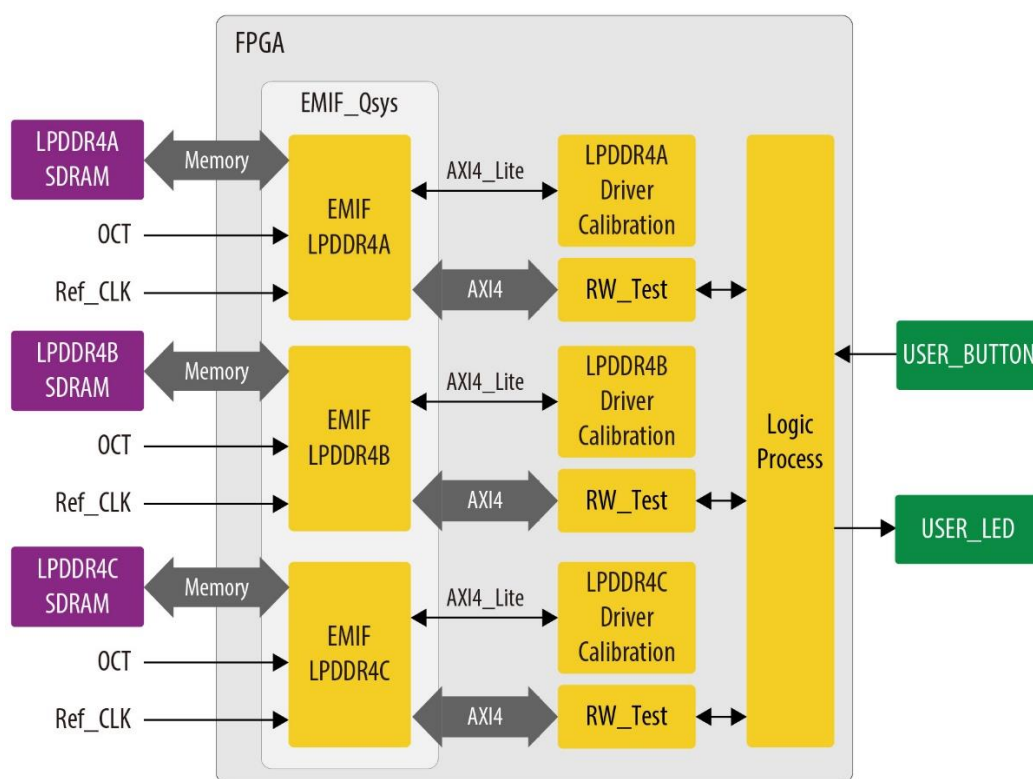


Figure 2-27 Block diagram of the LPDDR4 RTL demonstration

- **EMIF_Qsys:** It includes two LPDDR4 external memory interface (EMIF) IP which will handle all the LPDDR4 protocols and timings. It provides standard AXI4 for high-speed data transmission and AXI4-Lite for register configuration interfaces externally.
- **LPDDR4 Driver Calibration:** LPDDR4 has an extremely high signal rate and therefore must be calibrated at startup to compensate for signal delays and impedance variations on the board. This module acts as an AXI-Lite Master, after the system starts, it automatically gives instructions to the control registers of EMIF_Qsys to trigger and monitor the built-in hardware calibration program. After the calibration is successful, the reset signal will be released to the subsequent test modules.
- **AXI4 Master:** This is the core logic to verify LPDDR4. It acts as an AXI4 Master. After the calibration is completed, it will sequentially execute the test process of write, read and verify. It will generate a large amount of pseudo-random data, write it to LPDDR4 at high speed through the AXI4 bus, then read the data from the same address, and perform real-time comparison internally to ensure the accuracy of the data. After the test is completed, it will output the test_pass and test_complete signals.
- **Reset & Debouncer:** The debouncer module ensures that when users press User_Button, a clean and noise-free pulse can be generated. The hyper_pipe module is responsible for safely synchronizing this reset signal to the high-speed aclk frequency domain, preventing metastability issues and ensuring the reliability of system reset.

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition

■ Demonstration Source Code

- Project Directory: Demonstration\FPGA\LPDDR4_AXI4_RTL
- Bit Stream: golden_top.sof
- Demonstration Batch File : LPDDR4_AXI4_RTL\demo_batch

The demo batch file includes following files:

- ◆ Batch File: test.bat
- ◆ FPGA Configuration File: golden_top.sof

■ Demonstration Setup

1. Make sure Quartus Prime Pro Edition v25.3.1 is installed on the Host PC.
2. Connect the Comet A65 EVK board to Host PC via the Type-C USB cable. Install

the USB-Blaster III driver if necessary.

3. Power on the Comet A65 EVK board.
4. Execute the demo batch file “test.bat” under the batch file folder \LPDDR4_AXI4_RTL\demo_batch.
5. Press **USER_BUTTON** (see **Figure 2-28**) to start LPDDR4 write & loopback verify process. It will take about 1 second to perform the test.
6. The test result will show on **USER_LED**. If USER_LED is light on, it means the three LPDDR4 are tested pass. If there is one LPDDR4 tested fail, USER_LED will blink.
7. User can press **USER_BUTTON** again to regenerate the test control signals for a repeat test.

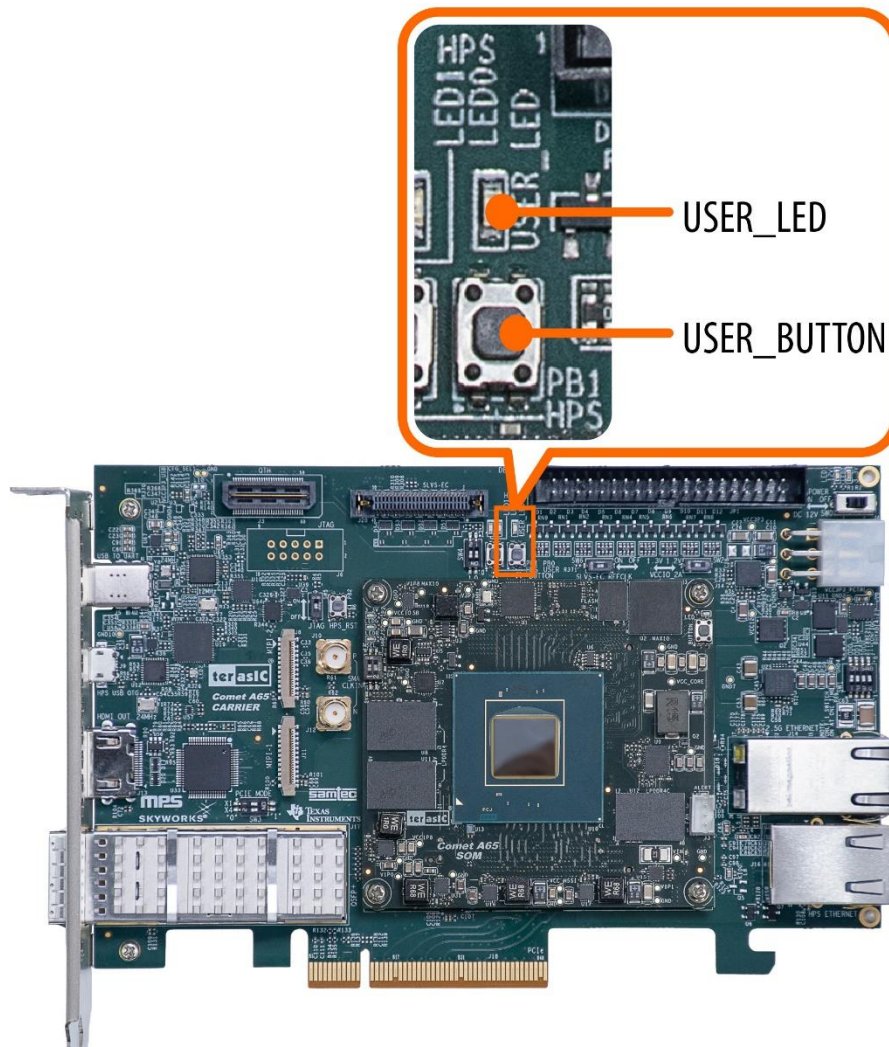


Figure 2-28 Location of the User Button and LED on the Comet A65 EVK board

2.8 SLVS_EC_RX_to_HDMI_IMX537 Test

This design is implemented on Comet A65 EVK. It receives data from CMOS Image Sensor IMX537 and outputs it to HDMI Transmitter TFP410 on the Comet A65 EVK. It offers the following features:

- Interfacing SLVS-EC CMOS Image Sensor IMX537 (RAW12bit 8Lane 2472(H) × 2128(V) 53.7fps format)
- 4GB LPDDR4 memory interface
- Nios V/m soft processor core and peripheral interfaces – SPI, GPIO
- Video Processing (Scaler, Frame Buffer, Picture Combination and Video Timing Generator)
- HDMI Interface – display the video on HDMI based Monitors through HDMI Transmitter TFP410 (1080 60p format)

■ Function Block Diagram

The basic system architecture consists of three main modules: Sensor Interface, Nios V/m processor system and Video/Image Signal Processing (ISP) module. The following **Figure 2-29** show the system architecture of the reference design.

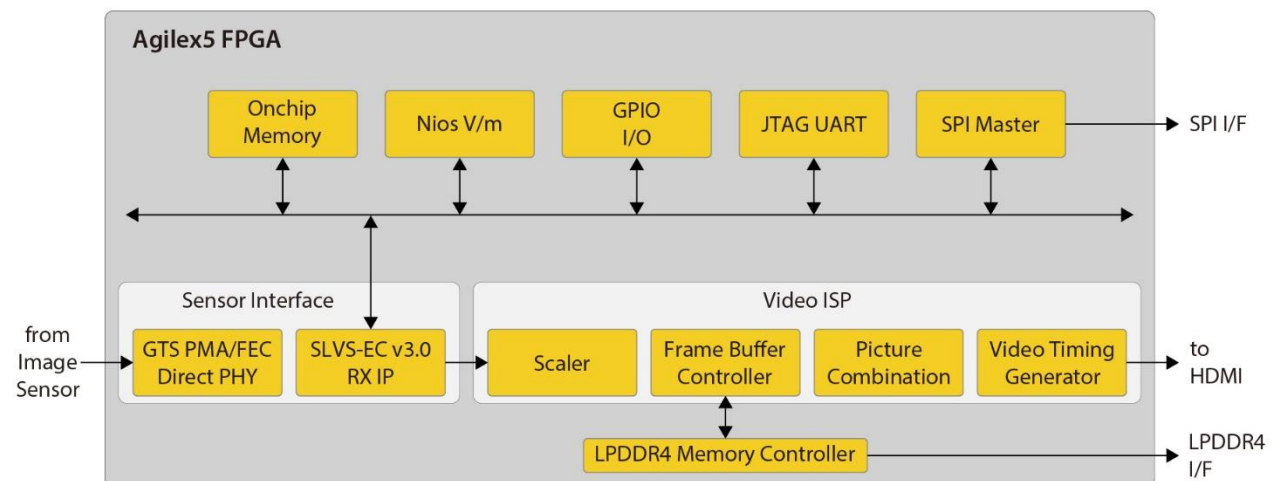


Figure 2-29 System Block Diagram

- **Sensor Interface:** The sensor interface consists of SLVS-EC v3.0 RX IP and GTS PMA/FEC Direct PHY. GTS PMA/FEC Direct PHY is generated by Quartus Prime software. Users can contact Macnica for SLVS-EC v3.0 RX IP license to compile

the design.

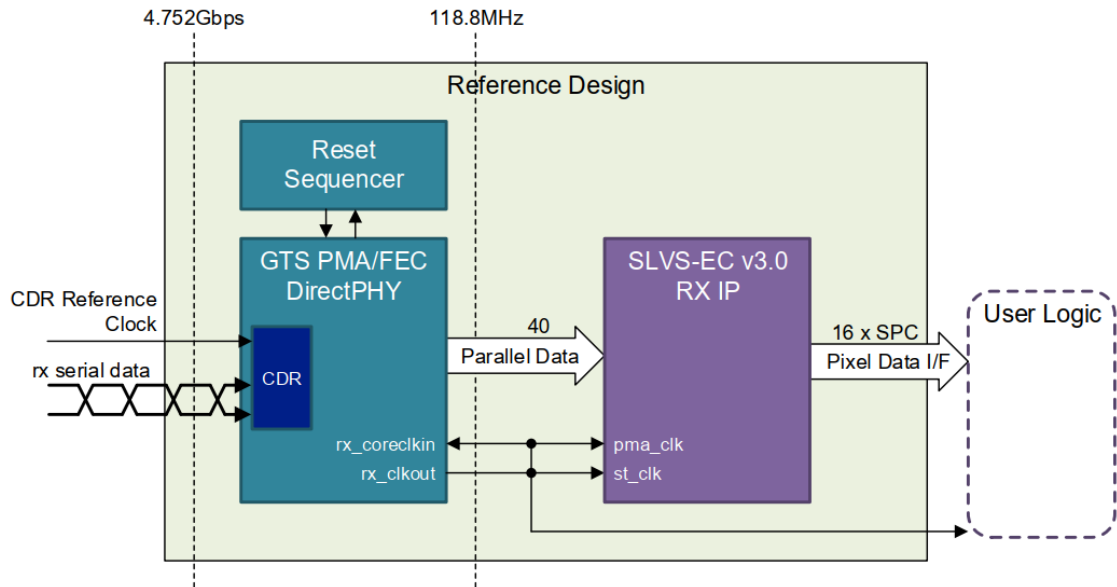


Figure 2-30 Sensor Interface

- **Video/Image Signal Processing:** It is a sub-system component used to process data from image sensor using Video/Image ISP IPs and transmits the processed data to HDMI Transmitter. Scaler scales input data to convert 2472×2128 to 1254×1080 . Frame Buffer Controller controls LPDDR4 Memory Controller. Picture Combination embeds input data of 1254×1080 into an image size of 1920×1080 . Video Timing Generator generates display synchronization signals based on the set display resolution. The synchronization signals are horizontal sync, vertical sync, and data enable. The input video data is also synchronized with these sync signals.
- **Nios V/m Processor System:** SPI Controller is used for IMX537 access. GPIO Controller for IMX537 access. Avalon-MM bridge module for SPI Master and SLVS EC v3.0 RX IP control.

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition
- Ashling RiscFree IDE for Intel FPGA v25.3.1

■ Demonstration Source Code

- Project Directory: Demonstration\FPGA\SLVS_EC_RX_to_HDMI_IMX537
- Bit Stream: slvs_ec_rx_to_hdmi_time_limited.sof
- Nios V Project workspace: SLVS_EC_RX_to_HDMI_IMX537/software

- Nios V bin file: demo1.elf

■ Demonstration Batch File

Demo batch directory: SLVS_EC_RX_to_HDMI_IMX537/demo_batch

The demo batch file includes following files:

- Batch File: test.bat
- FPGA Configuration File: slvs_ec_rx_to_hdmi_time_limited.sof
- Nios V bin file: demo1.elf

■ Demonstration Setup

- Make sure the SW6 on the Comet A65 Carrier board is set to ON, select 148.5 MHz clock for SLVS-EC.
- Connect the USB Blaster III connector (J5) of the board to the host PC USB port through USB Type-C cable. Make sure the Quartus Pro v25.3.1 and USB Blaster III driver are installed.
- Connect HDMI connector (J13) of the board to HDMI monitor through HDMI cable.
- Connect IMX537 camera to the SLVS-EC connector (J20) of the board.
- Connect the DC 12V Power supply to the board and power it on.
- Execute the test.bat to program .sof and .elf to the board.
- Once the configuration is completed, the video will be displayed, as shown in **Figure 2-31**.

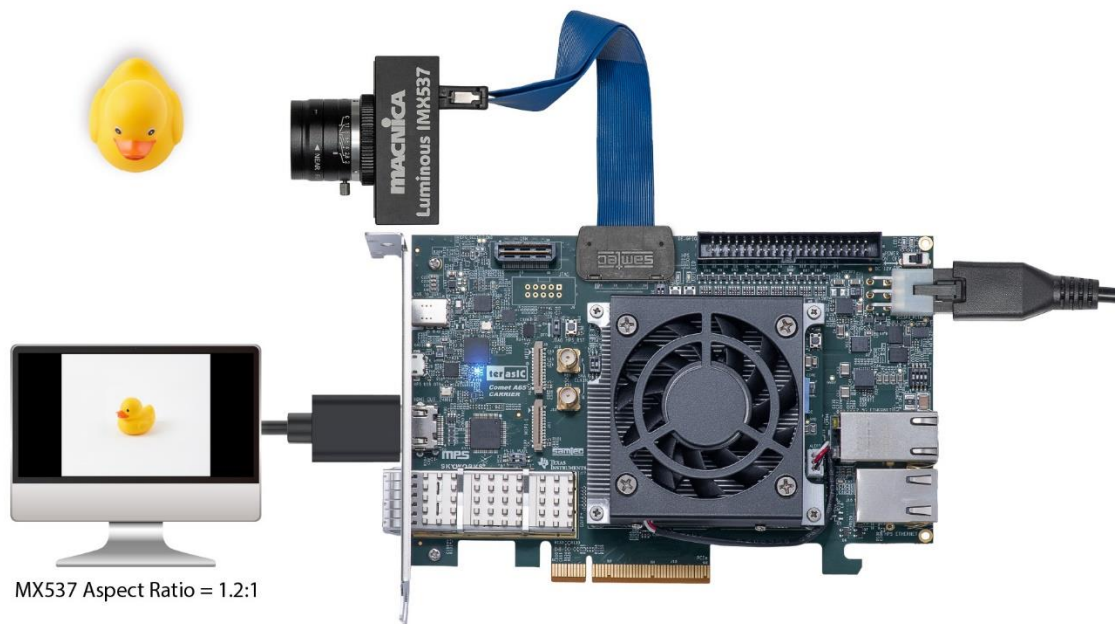


Figure 2-31 SLVS_EC_RX_to_HDMI_IMX537 demo test

2.9 SLVS_EC_RX_to_HDMI_IMX901 Test

This design receives data from CMOS Image Sensor IMX901 and outputs it to HDMI Transmitter TFP410 on the Comet A65 EVK. It offers the following features:

- Interfacing SLVS-EC CMOS Image Sensor IMX901 (RAW12bit 4Lane 8016(H) × 2112(V) 35.0fps format)
- 4GB LPDDR4 memory interface
- Nios V/m soft processor core and peripheral interfaces – SPI, GPIO
- Video Processing (Scaler, Frame Buffer, Picture Combination and Video Timing Generator)
- HDMI display the video on HDMI based Monitors through HDMI Transmitter TFP410 (1080 60p format)

■ Function Block Diagram

The basic system architecture consists of three main modules: Sensor Interface, Nios V/m processor system and Video/Image Signal Processing (ISP) module. The following **Figure 2-32** show the system architecture of the reference design.

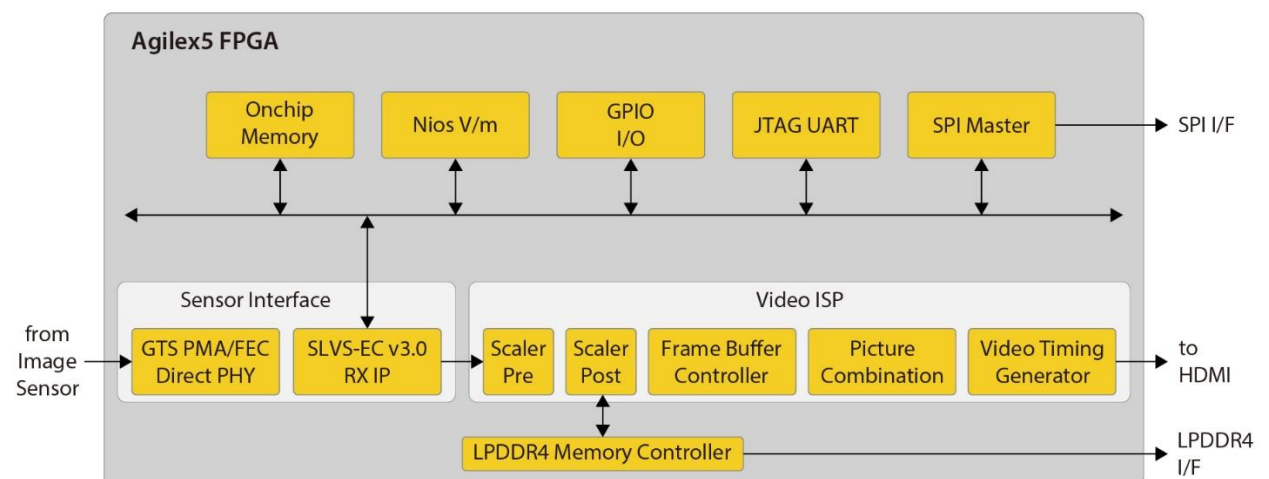


Figure 2-32 System Block Diagram

- **Sensor Interface:** The sensor interface consists of SLVS-EC v3.0 RX IP and GTS PMA/FEC Direct PHY. GTS PMA/FEC Direct PHY is generated by Quartus Prime software. Users can contact Macnica for SLVS-EC v3.0 RX IP license to compile the design.

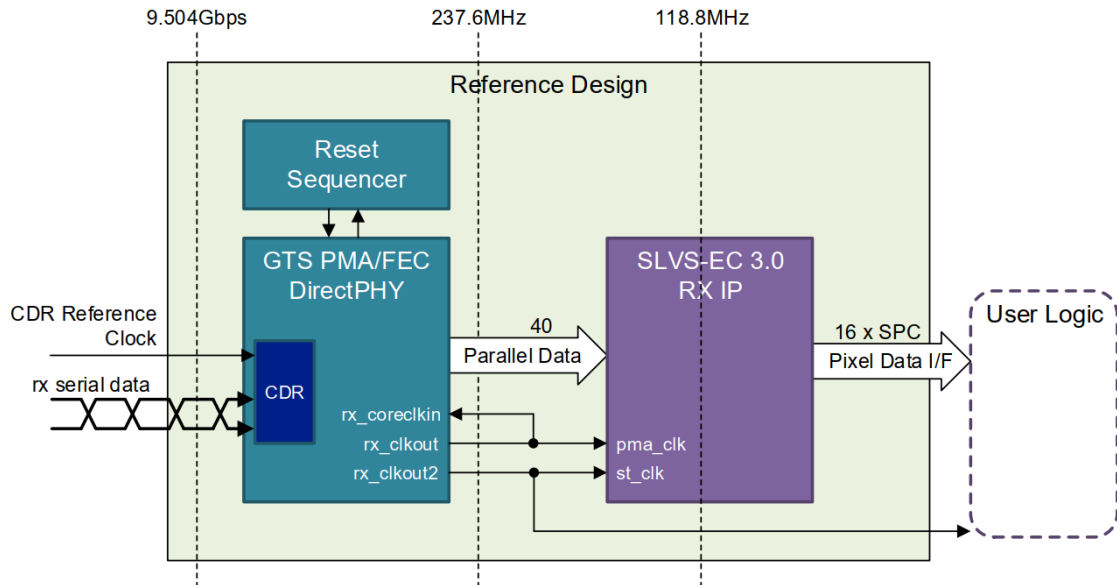


Figure 2-33 Sensor Interface

- **Video/Image Signal Processing:** It is a sub-system component used to process data from image sensor using Video/Image ISP IPs and transmits the processed data to HDMI Transmitter. Scaler Pre scales input data to convert 8016×2112 to 4008×2112 . Scaler Post scales input data to convert 4008×2112 to 1920×506 . Frame Buffer Controller controls LPDDR4 Memory Controller. Picture Combination embeds input data of 1920×506 into an image size of 1920×1080 . Video Timing Generator generates display synchronization signals based on the set display resolution. The synchronization signals are horizontal sync, vertical sync, and data enable. The input video data is also synchronized with these sync signals.
- **Nios V/m Processor System:** SPI Controller is used for IMX901 access. GPIO Controller for IMX901 access. Avalon-MM bridge module for SPI Master and SLVS EC v3.0 RX IP control.

■ Design Tools

- Quartus Prime 25.3.1 Pro Edition
- Ashling RiscFree IDE for Intel FPGA v25.3.1

■ Demonstration Source Code

- Project Directory: Demonstration\FPGA\SLVS_EC_RX_to_HDMI_IMX901
- Bit Stream: slvs_ec_rx_to_hdmi_time_limited.sof
- Nios V Project workspace: SLVS_EC_RX_to_HDMI_IMX901/software
- Nios V bin file: demo1.elf

■ Demonstration Batch File

Demo batch directory: SLVS_EC_RX_to_HDMI_IMX901/demo_batch

The demo batch file includes following files:

- Batch File: test.bat
- FPGA Configuration File: slvs_ec_rx_to_hdmi_time_limited.sof
- Nios V bin file: demo1.elf

■ Demonstration Setup

- Make sure the SW6 on the Comet A65 Carrier board is set to ON, select 148.5 MHz clock for SLVS-EC.
- Connect the USB Blaster III connector (J5) of the board to the host PC USB port through USB Type-C cable. Make sure the Quartus Pro v25.3 and USB Blaster III driver are installed.
- Connect HDMI connector (J13) of the board to HDMI monitor through HDMI cable.
- Connect IMX901 camera to the SLVS-EC connector (J20) of the board.
- Connect the DC 12V Power supply to the board and power it on.
- Execute the test.bat to program .sof and .elf to the board.
- Once the configuration is completed, the video will be displayed, as shown in **Figure 2-34**.

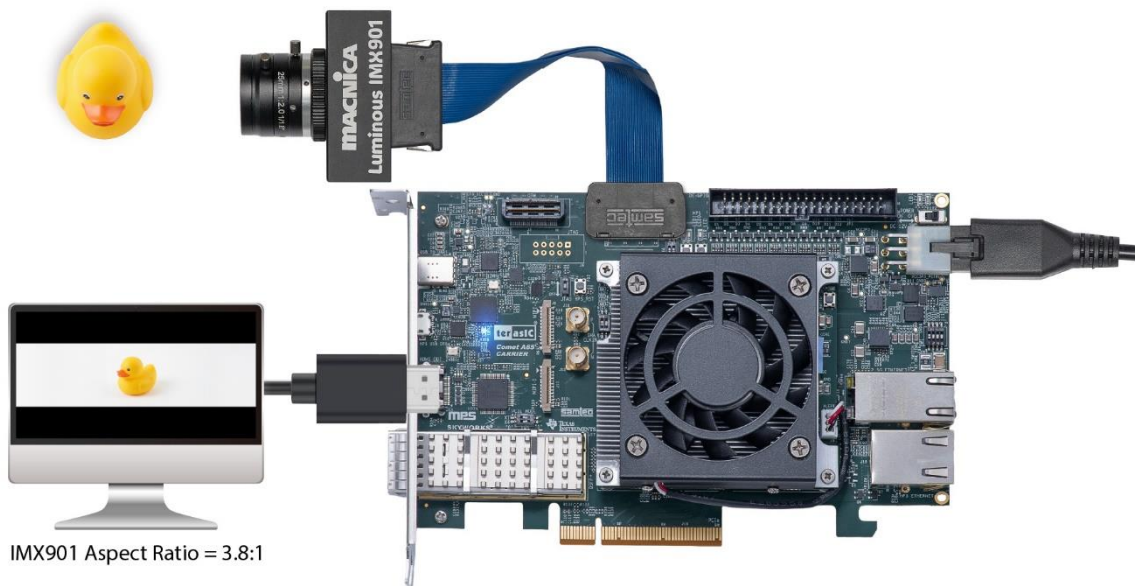


Figure 2-34 SLVS_EC_RX_to_HDMI_IMX901 demo test

Chapter 3

PCI Express Reference Design for Windows

PCI Express is commonly used in consumer, server, and industrial applications, to link motherboard-mounted peripherals. From this demonstration, it will show how the PC Windows and FPGA communicate with each other through the PCI Express interface. GTS AXI Multichannel DMA IP for PCI Express IP is used in this demonstration. For detail about this IP, please refer to Altera document [ug-847470-857392](https://www.altera.com/docs/en/latest/ip/interconnect/gts-axi-multichannel-dma-ip-for-pci-express).

3.1 PCI Express System Infrastructure

Figure 3-1 shows the infrastructure of the PCI Express System in this demonstration. It consists of two primary components: FPGA System and PC System. The FPGA System is developed based on GTS AXI Multichannel DMA IP for PCI Express. The application software on the PC side is developed by Terasic based on Altera's PCIe kernel mode driver.

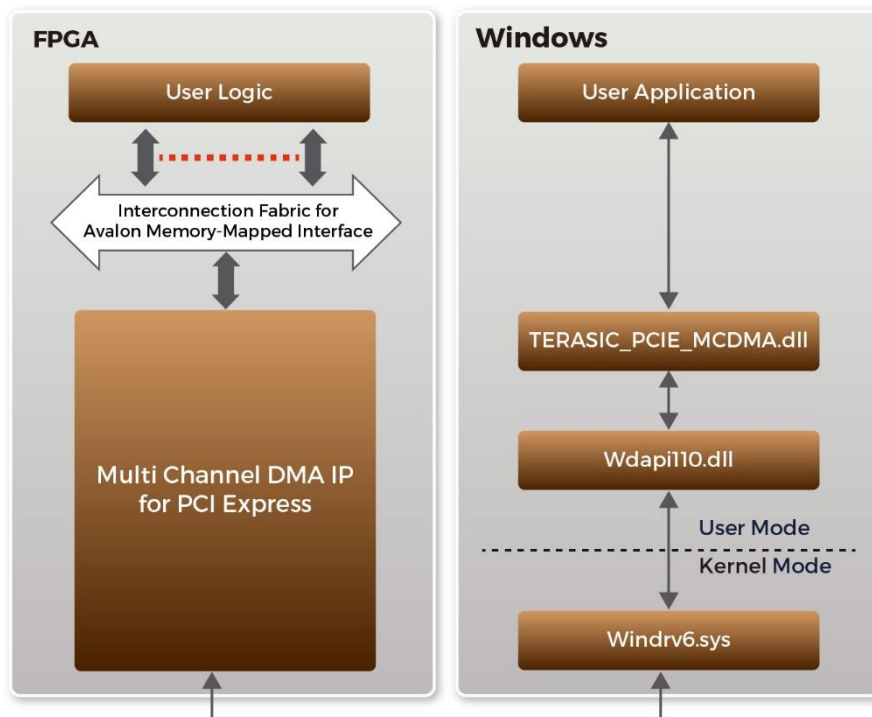


Figure 3-1 Infrastructure of PCI Express System

3.2 PC PCI Express Software SDK

The FPGA System CD contains a PC Windows based SDK to allow users to develop their 64-bit software application on 64-bits Windows 10. The SDK is located in the "CDROM\Demonstrations\FPGA\PCIe_SW_KIT\Windows" folder which includes:

- PCI Express Driver
- PCI Express Library
- PCI Express Examples

The kernel mode driver assumes the PCIe vendor ID (VID) is 0x1172 and the device ID (DID) is 0x09C4. If different VID and DID are used in the design, users need to modify the PCIe vendor ID (VID) and device ID (DID) in the driver INF file accordingly.

The PCI Express Library is implemented as a single DLL named Terasic_Pcie_MCDMA.dll. This file is a 64-bit DLL. With the exported software API, users can easily communicate with the FPGA. The library provides the following functions:

- Basic data read and write
- Data read and write by DMA

For high performance data transmission, MCDMA is required as the read and write operations, which are specified under the hardware design on the FPGA.

3.3 PCI Express Software Stack

Figure 3-2 shows the software stack for the PCI Express application software on 64-bit Windows. The PCIe library module Terasic_Pcie_MCDMA.dll provides DMA and direct I/O access allowing user application program to communicate with FPGA. Users can develop their applications based on this DLL. The altera_pcie_win_driver.sys kernel driver is provided by Altera.

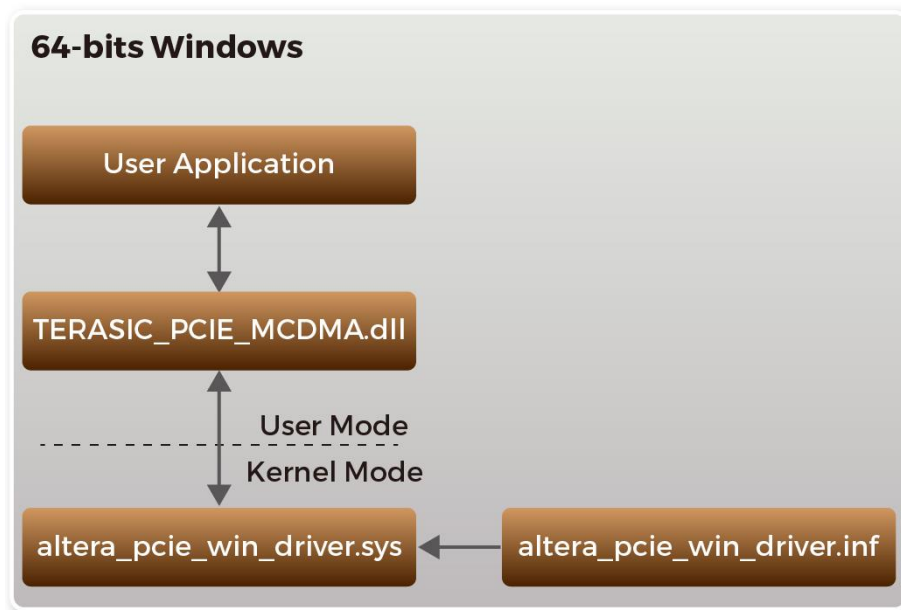


Figure 3-2 PCI Express Software Stack

■ Install PCI Express Driver on Windows

The PCIe driver is located in the folder:

"CDROM\Demonstrations\FPGA\PCIe_SW_KIT\Windows\PCIe_Driver"

The folder includes the following four files:

- Altera_pcie_win_driver.cat
- Altera_pcie_win_driver.inf
- Altera_pcie_win_driver.sys
- WdfCoinstaller01011.dll

To install the PCI Express driver, please execute the steps below:

1. Install the board on the PCIe slot of the host PC.
2. Make sure the Altera Programmer and USB-Blaster III driver are installed.
3. Because the Windows 10 enforces driver signatures by default and the OpenCL drivers for our development kits are not "signed" for Windows 10. So, for Windows10, please refer to the [link](#) to **disable driver signature enforcement and reboot system**.
4. Execute test.bat in "Demonstrations\FPGA\PCIe_Fundamental\demo_batch" to configure the FPGA.
5. Restart windows operation system.
6. Click the Control Panel menu from Windows Start menu. Click the Hardware and Sound item before clicking the Device Manager to launch the Device

Manager dialog. There will be a PCI Device item in the dialog, as shown in **Figure 3-3**. Move the mouse cursor to the PCI Device item and right click it to select the Updated Driver Software... items.

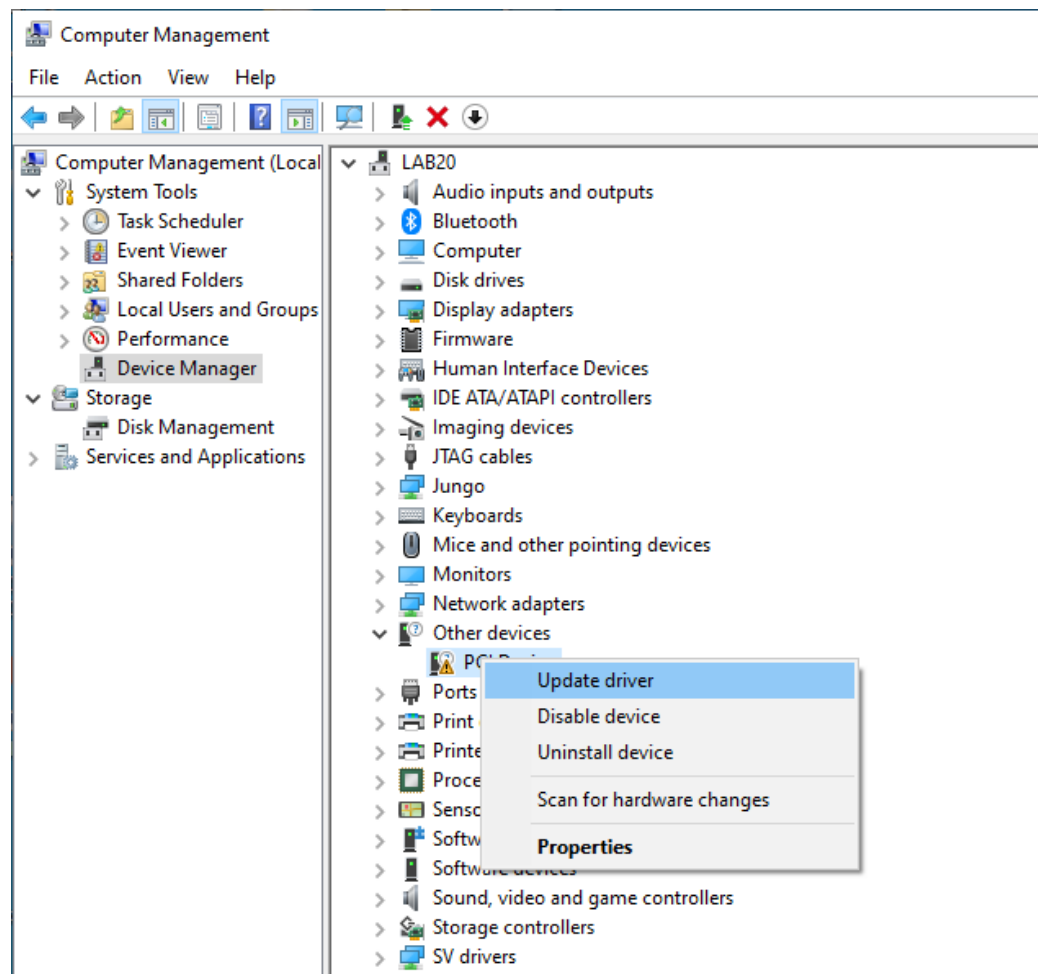


Figure 3-3 Screenshot of launching Update Driver Software... dialog

7. In the **How do you want to search for the driver software** dialog, click **Browse my computer for driver software** item, as shown in **Figure 3-4**

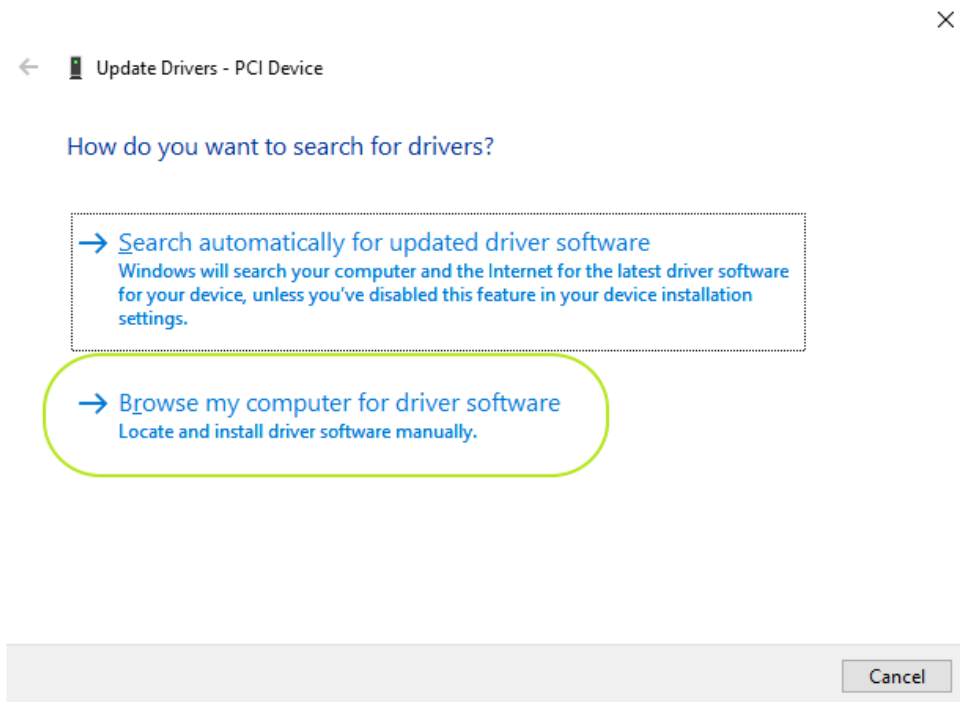


Figure 3-4 Dialog of Browse my computer for the driver software

8. In the **Browse for driver software on your computer** dialog, click the **Browse** button to specify the folder where altera_pcie_din_driver.inf is located, as shown in **Figure 3-5**. Click the **Next** button.

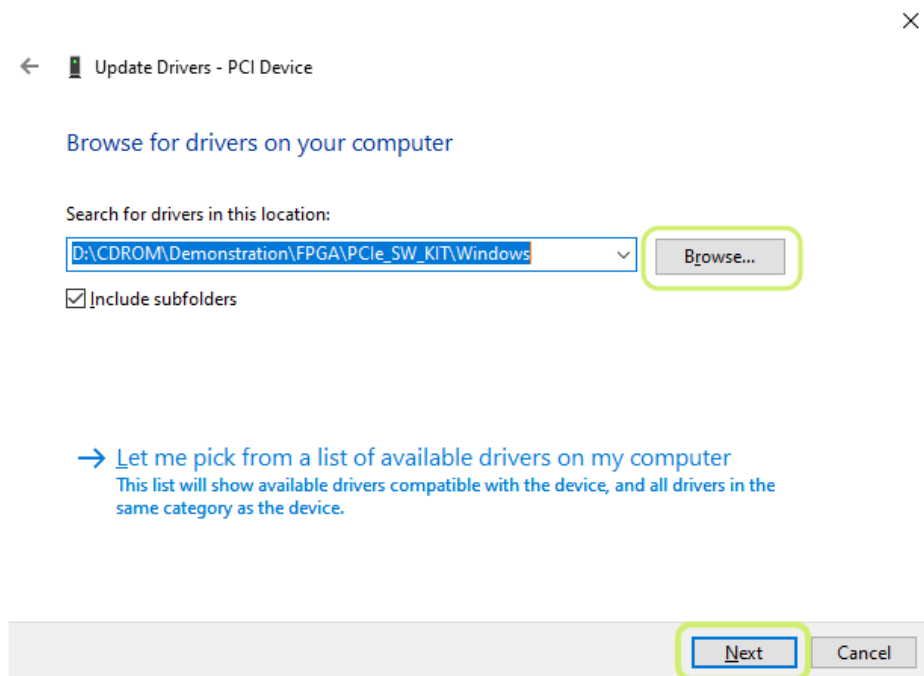


Figure 3-5 Browse for the driver software on your computer

9. When the **Windows Security** dialog appears, as shown **Figure 3-6**, click the **Install** button.

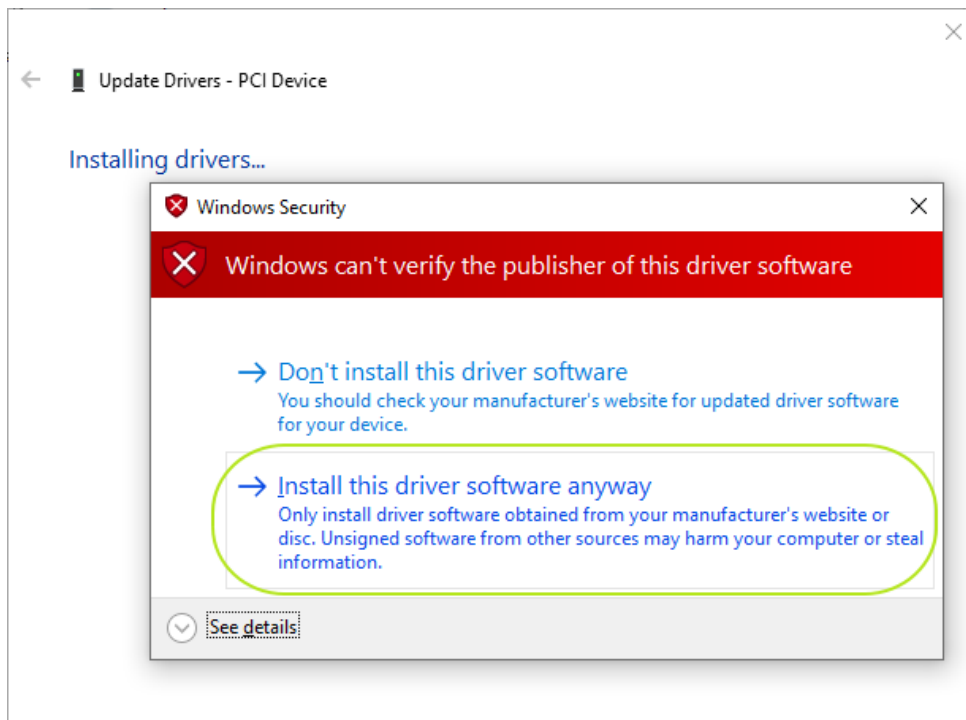


Figure 3-6 Click Install in the dialog of Windows Security

10. When the driver is installed successfully, the successfully dialog will appear, as shown in

11. Figure 3-7. Click the **Close** button.

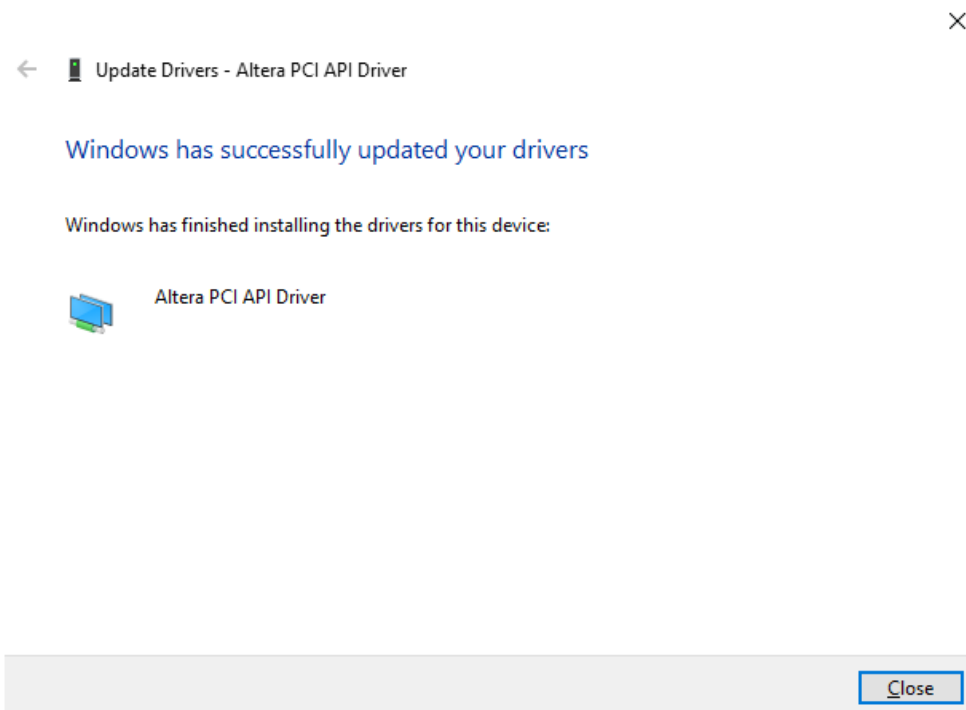


Figure 3-7 Altera PCI API Driver installation is completed

12. Once the driver is successfully installed, users can see the **Altera PCI API**

Driver under the device manager window, as shown in **Figure 3-8**.

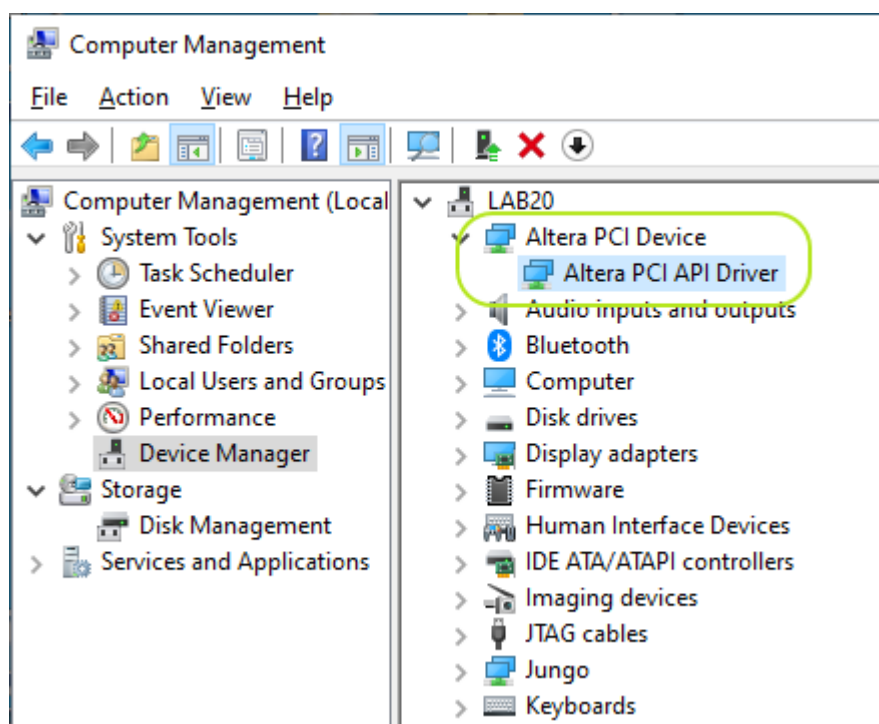


Figure 3-8 Altera PCI API Driver in Device Manager

■ Create a Software Application

All the files needed to create a PCIe software application are located in the directory CDROM\demonstration\PCIe_SW_KIT\Windows\PCIe_Library. It includes the following files:

- TERASIC_PCIE_MCDMA.h
- TERASIC_PCIE_MCDMA.dll (64-bit DLL)

Below lists the procedures to use the SDK files in users' C/C++ project:

1. Create a 64-bit C/C++ project.
2. Include TERASIC_PCIE_MCDMA.h in the C/C++ project.
3. Copy TERASIC_PCIE_MCDMA.dll to the folder where the project.exe is located.
4. Dynamically load TERASIC_PCIE_MCDMA.dll in C/C++ program. To load the DLL, please refer to the PCIe fundamental example below.
5. Call the SDK API to implement the desired application.

Users can easily communicate with the FPGA through the PCIe bus through the TERASIC_PCIE_MCDMA.dll API. The details of API are described below:

3.4 PCI Express Library API

Below shows the exported API in the Terasic_PCIE_MCDMA.dll. The API prototype is defined in the Terasic_PCIE_MCDMA.h.

Note: the Linux library terasic_pcie_mcdma.so also use the same API and header file.

■ PCIE_Open

Function:
Open a specified PCIe card with vendor ID, device ID, and matched card index.
Prototype:
<pre>PCIE_HANDLE PCIE_Open(Uint16_t wVendorID, Uint16_t wDeviceID, Uint16_t wSubVendorID, Uint16_t wSubDeviceID, Uint16_t wCardIndex);</pre>
Parameters:
wVendorID: Specify the desired vendor ID. A zero value means to ignore the vendor ID.
wDeviceID: Specify the desired device ID. A zero value means to ignore the device ID.
wSubVendorID: Specify the desired subsystem vendor ID. A zero value means to ignore the subsystem vendor ID.
wSubDeviceID: Specify the desired subsystem device ID. A zero value means to ignore the subsystem device ID.
wCardIndex: Specify the matched card index, a zero based index, based on the matched vendor ID and device ID.
Return Value:
Return a handle to the specified PCIe card. A positive value is return if the PCIe card is opened successfully. A value zero means failed to connect the target PCIe card. This handle value is used as a parameter for other functions, e.g. PCIE_Read32. Users need to call PCIE_Close to release handle once the handle is no longer used.

■ PCIE_Close

Function:
Close a handle associated to the PCIe card.
Prototype:
void PCIE_Close(PCIE_HANDLE hPCIE);
Parameters:
hPCIE: A PCIe handle return by PCIE_Open function.
Return Value:
None.

■ PCIE_Read32

Function:
Read a 32-bit data from the FPGA board.
Prototype:
bool PCIE_Read32(PCIE_HANDLE hPCIE, PCIE_BAR PcieBar, PCIE_ADDRESS PcieAddress, uint32_t *pdwData);
Parameters:
hPCIE: A PCIe handle return by PCIE_Open function.
PcieBar: Specify the target BAR.
PcieAddress: Specify the target address in FPGA.
pdwData: A buffer to retrieve the 32-bit data.
Return Value:
Return true if read data is successful; otherwise false is returned.

■ PCIE_Write32

Function:
Write a 32-bit data to the FPGA Board.
Prototype:

```
bool PCIE_Write32(
    PCIE_HANDLE hPCIE,
    PCIE_BAR PcieBar,
    PCIE_ADDRESS PcieAddress,
    uint32_t dwData);
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

PcieBar:

Specify the target BAR.

PcieAddress:

Specify the target address in FPGA.

dwData:

Specify a 32-bit data which will be written to FPGA board.

Return Value:

Return **true** if write data is successful; otherwise **false** is returned.

■ PCIE_Read8

Function:

Read an 8-bit data from the FPGA board.

Prototype:

```
bool PCIE_Read8(
    PCIE_HANDLE hPCIE,
    PCIE_BAR PcieBar,
    PCIE_ADDRESS PcieAddress,
    uint8_t *pByte);
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

PcieBar:

Specify the target BAR.

PcieAddress:

Specify the target address in FPGA.

pByte:

A buffer to retrieve the 8-bit data.

Return Value:

Return **true** if read data is successful; otherwise **false** is returned.

■ PCIE_Write8

Function:

Write an 8-bit data to the FPGA Board.

Prototype:

```
bool PCIE_Write8(  
    PCIE_HANDLE hPCIE,  
    PCIE_BAR PcieBar,  
    PCIE_ADDRESS PcieAddress,  
    uint8_t Byte);
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

PcieBar:

Specify the target BAR.

PcieAddress:

Specify the target address in FPGA.

Byte:

Specify an 8-bit data which will be written to FPGA board.

Return Value:

Return **true** if write data is successful; otherwise **false** is returned.

■ PCIE_DmaRead

Function:

Read data from the memory-mapped memory of FPGA board in DMA.

Prototype:

```
bool PCIE_DmaRead(  
    PCIE_HANDLE hPCIE,  
    PCIE_LOCAL_ADDRESS LocalAddress,  
    void *pBuffer,  
    uint64_t dwBufSize64  
);
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

<p>LocalAddress:</p> <p>Specify the target memory-mapped address in FPGA.</p> <p>pBuffer:</p> <p>A pointer to a memory buffer to retrieved the data from FPGA. The size of buffer should be equal or larger the dwBufSize.</p> <p>dwBufSize64:</p> <p>Specify the byte number of data retrieved from FPGA.</p>
<p>Return Value:</p> <p>Return true if read data is successful; otherwise false is returned.</p>

■ PCIE_DmaWrite

<p>Function:</p> <p>Write data to the memory-mapped memory of FPGA board in DMA.</p>
<p>Prototype:</p> <pre>bool PCIE_DmaWrite(PCIE_HANDLE hPCIE, PCIE_LOCAL_ADDRESS LocalAddress, void *pData, uint64_t dwDataSize64);</pre>
<p>Parameters:</p> <p>hPCIE:</p> <p>A PCIe handle return by PCIE_Open function.</p> <p>LocalAddress:</p> <p>Specify the target memory mapped address in FPGA.</p> <p>pData:</p> <p>A pointer to a memory buffer to store the data which will be written to FPGA.</p> <p>dwDataSize64:</p> <p>Specify the byte number of data which will be written to FPGA.</p>
<p>Return Value:</p> <p>Return true if write data is successful; otherwise false is returned.</p>

■ PCIE_ConfigRead32

<p>Function:</p> <p>Read PCIe Configuration Table. Read a 32-bit data by given a byte offset.</p>
<p>Prototype:</p>

```
bool PCIE_ConfigRead32 (
    PCIE_HANDLE hPCIE,
    uint32_t Offset,
    uint32_t *pdwData
);
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

Offset:

Specify the target byte of offset in PCIe configuration table.

pdwData:

A 4-bytes buffer to retrieve the 32-bit data.

Return Value:

Return **true** if read data is successful; otherwise **false** is returned.

3.5 PCIe Reference Design Fundamental

The application reference design shows how to implement fundamental control and data transfer in DMA. In the design, basic I/O is used to control the BUTTON and LED on the FPGA board. High-speed data transfer is performed by the DMA.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM\Demonstrations\FPGA\PCIe_Fundamental\demo_batch

The folder includes following files:

- FPGA Configuration File: golden_top.sof
- Download Batch file: test.bat
- Windows Application Software folder: windows_app, includes
 - ✧ PCIE_FUNDAMENTAL.exe
 - ✧ TERASIC_PCIE_MCDMA.dll

■ Demonstration Setup

1. Install the FPGA board on your PC.
2. Configure FPGA with golden_top.sof by executing the test.bat.
3. Install the PCIe driver if necessary. The driver is located in the folder:

CDROM\Demonstration\PCIe_SW_KIT\Windows\PCIe_Driver.

4. Restart Windows.
5. Make sure that Windows has detected the FPGA Board by checking the Windows Device Manager as shown in **Figure 3-9**.

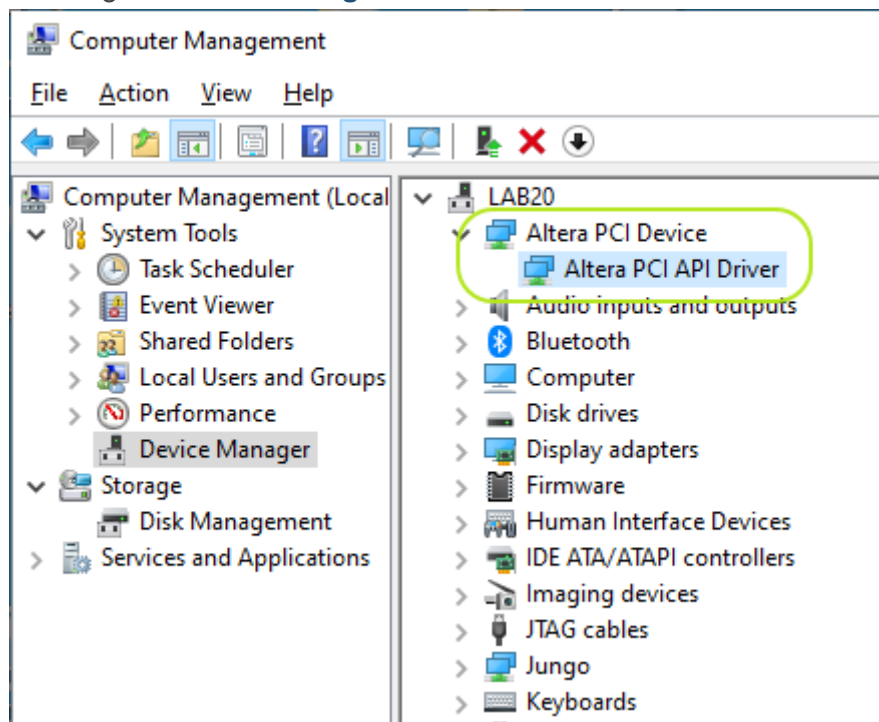


Figure 3-9 Screenshot for PCIe Driver

6. Go to windows_app folder, execute PCIE_FUNDMENTAL.exe. A menu will appear as shown in **Figure 3-10**.

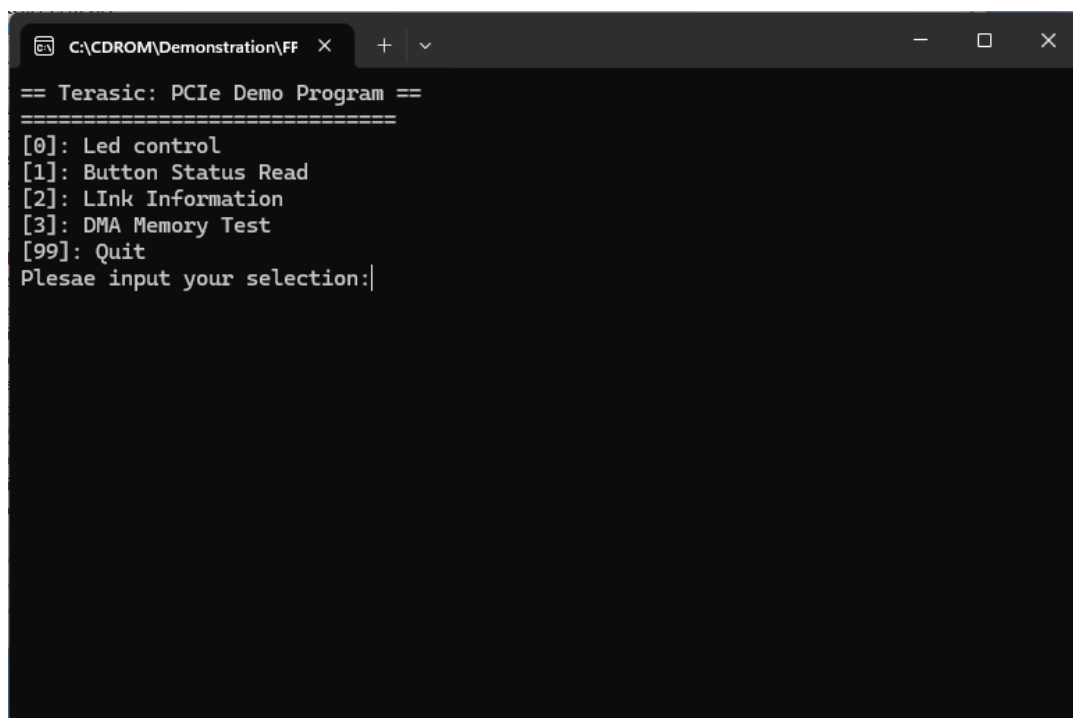
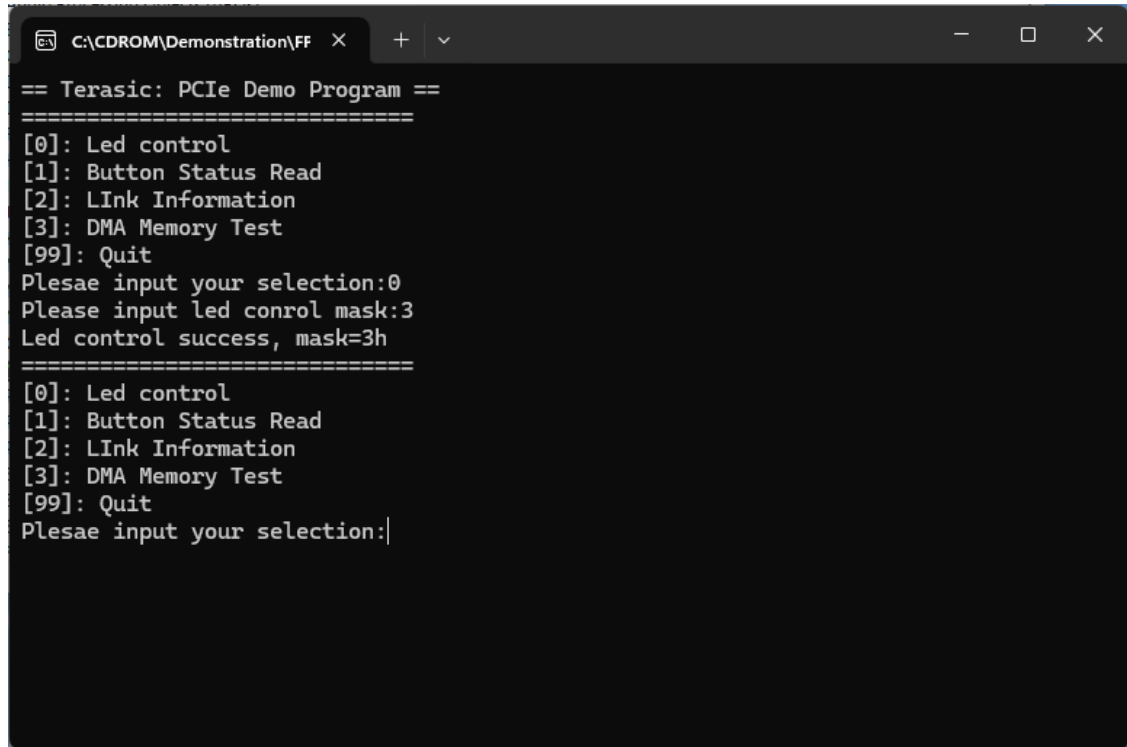


Figure 3-10 Screenshot of Program Menu

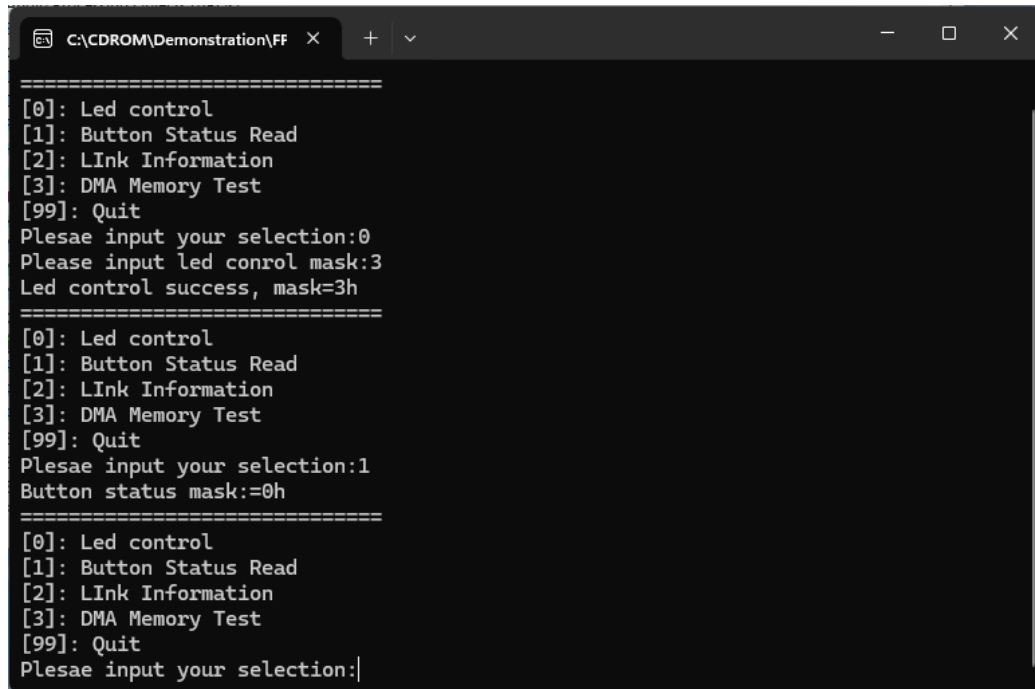
7. Type 0 followed by the ENTER key to select Led Control item, then input 3 (hex 0x03) will make all LEDs on as shown in **Figure 3-11**. If input 0 (hex 0x00), all LEDs will be turned off.



```
C:\CDROM\Demonstration\FF x + v
== Terasic: PCIe Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:0
Please input led control mask:3
Led control success, mask=3h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:|
```

Figure 3-11 Screenshot of LED Control

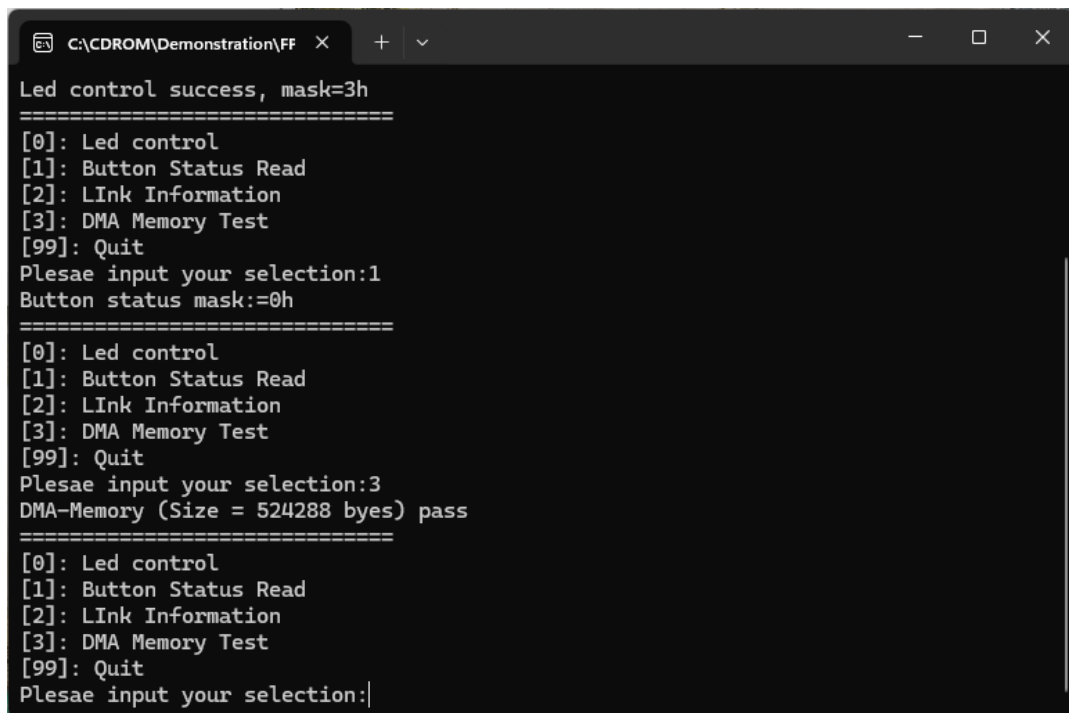
8. Type 1 followed by the ENTER key to select Button Status Read item. The button status will be reported as shown in **Figure 3-12**.



```
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:0  
Please input led conrol mask:3  
Led control success, mask=3h  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:1  
Button status mask:=0h  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:|
```

Figure 3-12 Screenshot of Button Status Report

9. Type 3 followed by an ENTER key to select the DMA Testing item. The DMA test result will be reported as shown in **Figure 3-13**.



```
Led control success, mask=3h  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:1  
Button status mask:=0h  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:3  
DMA-Memory (Size = 524288 bytes) pass  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: LInk Information  
[3]: DMA Memory Test  
[99]: Quit  
Plesae input your selection:|
```

Figure 3-13 Screenshot of DMA Memory Test Result

10. Type 99 followed by the ENTER key to exit this test program

■ Development Tools

- Quartus Prime 25.3.1 Pro Edition
- Visual C++ 2019

■ Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIe_Fundamental
- C++ Project: Demonstrations\PCIe_SW_KIT\Windows\PCIE_FUNDAMENTAL

■ FPGA Application Design

Figure 3-14 shows the system block diagram in the FPGA system. In the **Platform Designer** (formerly Qsys), the PIO controller is used to control the LED and monitor the Button Status, and the On-Chip memory is used for performing DMA testing. The PIO controllers and the On-Chip memory are connected to the PCI Express Hard IP controller through the Memory-Mapped Interface.

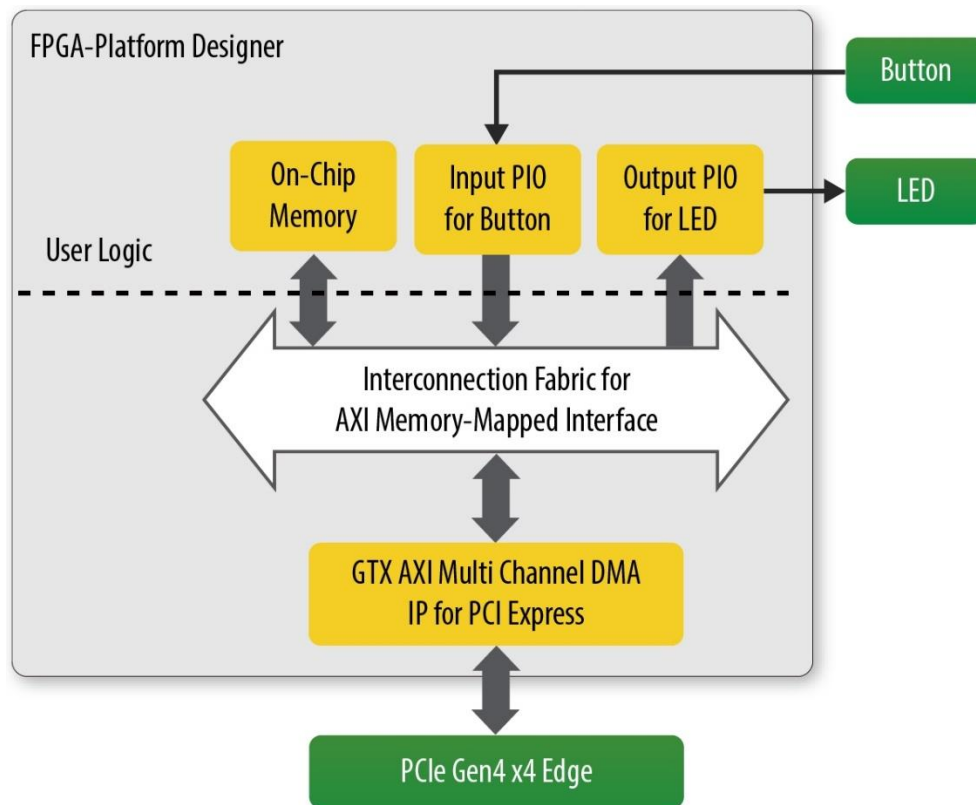


Figure 3-14 Hardware block diagram of the PCIe reference design

■ Windows Based Application Software Design

The application software project is built by Visual C++ 2019. The project includes the following major files:

Name	Description
PCIE_FUNDAMENTAL.cpp	Main program
PCIE.c	Implement dynamically load for TERASIC_PCIE_MCDMA.dll
PCIE.h	
TERASIC_PCIE_MCDMA.h	SDK library file, defines constant and data structure

The main program PCIE_FUNDAMENTAL.cpp includes the header file "PCIE.h" and defines the controller address according to the FPGA design.

```
#define DEMO_PCIE_USER_BAR      PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR   0x800000
#define DEMO_PCIE_IO_BUTTON_ADDR 0x800040
#define DEMO_PCIE_MEM_ADDR      0x100000

#define MEM_SIZE                 (512*1024) //512KB
```

The base address of BUTTON and LED controllers are 0x800040 and 0x800000 based on the PCIE_BAR4, respectively. The on-chip memory base address is 0x100000 relative to the DMA controller.

Before accessing the FPGA through PCI Express, the application first calls the PCIE_Load to dynamically load the TERASIC_PCIE_MCDMA.dll. Then, it calls PCIE_Open to open the PCI Express driver. The constant DEFAULT_PCIE_VID and DEFAULT_PCIE_DID used in the PCIE_Open are defined in TERASIC_PCIE_MCDMA.h. If developers change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value defined in TERASIC_PCIE_MCDMA.h. If the return value of PCIE_Open is zero, it means the driver cannot be accessed successfully. In this case, please make sure:

- The FPGA is configured with the associated bit-stream file and the host is rebooted.
- The PCI express driver is loaded successfully.

The LED control is implemented by calling **PCIE_Write32** API, as shown below:

```
bPass = PCIE_Write32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_LED_ADDR, (uint32_t) Mask);
```

The button status query is implemented by calling the **PCIE_Read32** API, as shown below:

```
bPass = PCIE_Read32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_BUTTON_ADDR, &Status);
```

The memory-mapped memory read and write test is implemented by **PCIE_DmaWrite** and **PCIE_DmaRead** API, as shown below:

```
bPass = PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
bPass = PCIE_DmaRead(hPCIE, LocalAddr, pRead, nTestSize);
```

3.6 PCIe Reference Design - LPDDR4

The application reference design shows how to add the LPDDR4 Memory Controllers for the LPDDR4-A, LPDDR4-B and LPDDR4-C Components into the PCIe Quartus project based on the PCIe_Fundamental Quartus project and perform 4GB data DMA for all components. Also, this demo shows how to call “PCIE_ConfigRead32” API to check PCIe link status.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM\Demonstrations\FPGA\PCIe_DDR4\demo_batch

The folder includes following files:

- FPGA Configuration File: golden_top.sof
- Download Batch file: test.bat
- Windows Application Software folder: windows_app, includes
 - ✧ PCIE_DDR4.exe
 - ✧ TERASIC_PCIE_MCDMA.dll

■ Demonstration Setup

1. Install the FPGA board on your PC.
2. Configure the FPGA with the golden_top.sof by executing the test.bat.
3. Install the PCIe driver if necessary.
4. Restart Windows
5. Make sure that Windows has detected the FPGA Board by checking the Windows Control panel.
6. Go to windows_app folder, execute PCIE_DDR4.exe. A menu will appear as shown in **Figure 3-15**.

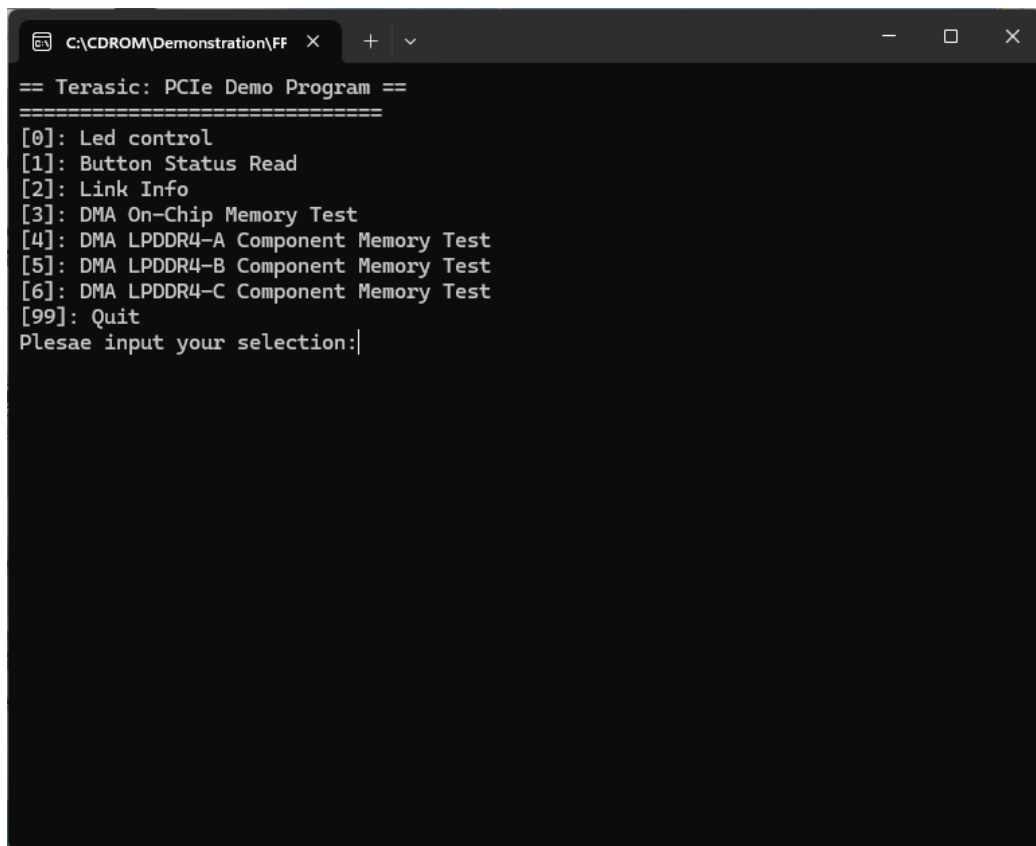
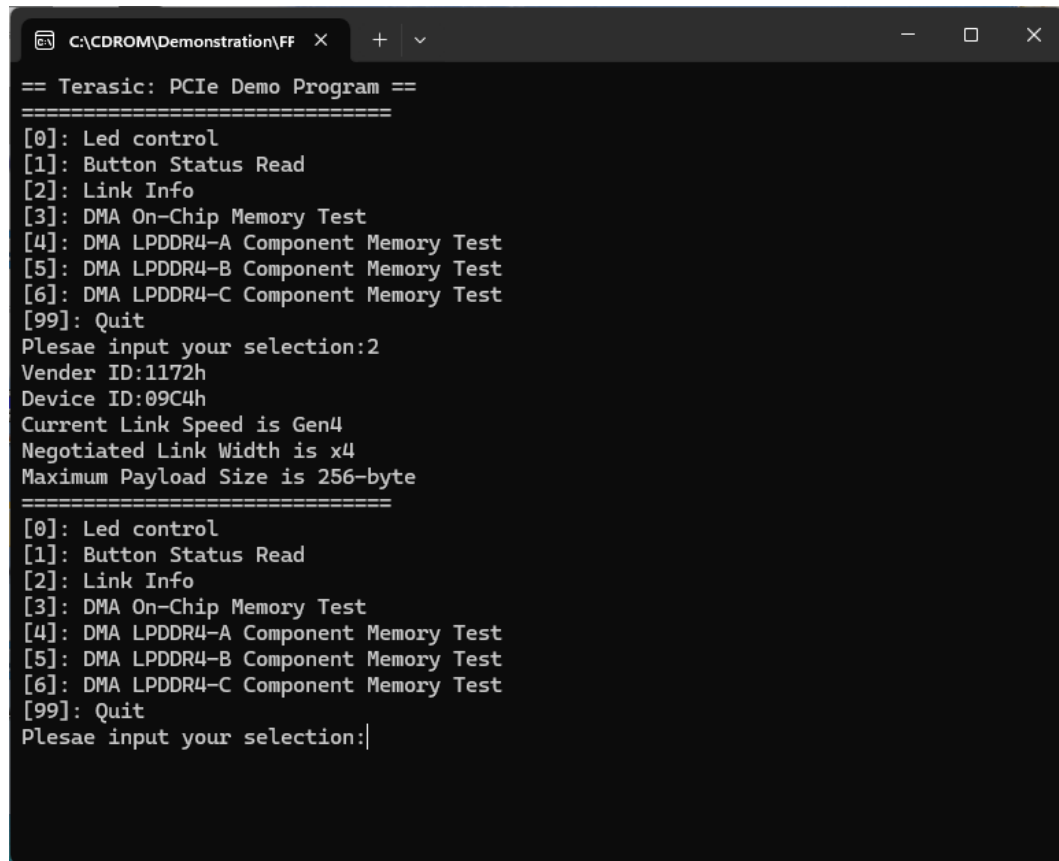


Figure 3-15 Screenshot of Program Menu

7. Type 2 followed by the ENTER key to select the Link Info item. The PCIe link information will be shown as in [Figure 3-16](#). Gen4 link speed and x4 link width are expected.



```
C:\CDROM\Demonstration\FF x + v
== Terasic: PCIe Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:2
Vender ID:1172h
Device ID:09C4h
Current Link Speed is Gen4
Negotiated Link Width is x4
Maximum Payload Size is 256-byte
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:|
```

Figure 3-16 Screenshot of Link Info

8. Type 3 followed by the ENTER key to select DMA On-Chip Memory Test item. The DMA write and read test result will be reported as shown in [Figure 3-17](#).

```
C:\CDROM\Demonstration\FF X + -
Device ID:09C4h
Current Link Speed is Gen4
Negotiated Link Width is x4
Maximum Payload Size is 256-byte
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:3
DMA Memory Test, Address = 0x100000, Size = 0x80000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (100000 - 180000)
Readback Data Verify...
DMA-Memory Address = 0x100000, Size = 0x80000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:|
```

Figure 3-17 Screenshot of On-Chip Memory DMA Test Result

9. Type 4 followed by the ENTER key to select the DMA LPDDR4-A component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 3-18**.

```
C:\CDROM\Demonstration\FF X + - □ X
DMA Write...
DMA Read... (100000 - 180000)
Readback Data Verify...
DMA-Memory Address = 0x100000, Size = 0x80000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:4
DMA Memory Test, Address = 0x100000000, Size = 0x100000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (1000000000 - 1100000000)
Readback Data Verify...
DMA-Memory Address = 0x100000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:|
```

Figure 3-18 Screenshot of the LPDDR4-A Memory DMA Test Result

10. Type 5 followed by the ENTER key to select the DMA LPDDR4-B Component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 3-19**.

```
C:\CDROM\Demonstration\FF X + v
DMA Write...
DMA Read... (1000000000 - 1100000000)
Readback Data Verify...
DMA-Memory Address = 0x100000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:5
DMA Memory Test, Address = 0x1100000000, Size = 0x100000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (1100000000 - 1200000000)
Readback Data Verify...
DMA-Memory Address = 0x1100000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:|
```

Figure 3-19 Screenshot of the LPDDR4-B Component Memory DMA Test Result

11. Type 6 followed by the ENTER key to select the DMA LPDDR4-C Component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 3-20**.

```

C:\CDROM\Demonstration\FF
Generate Test Pattern...
DMA Write...
DMA Read... (1100000000 - 1200000000)
Readback Data Verify...
DMA-Memory Address = 0x1100000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:6
DMA Memory Test, Address = 0x1200000000, Size = 0x100000000 Bytes...
Generate Test Pattern...
DMA Write...
DmaReadWrite_Block failed.nAccXferSize=1274634240, DataSize64=4294967296
DMA Memory:PCIE_DmaWrite failed
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:|

```

Figure 3-20 Screenshot of the LPDDR4-C Component Memory DMA Test Result

12. Type 99 followed by the ENTER key to exit this test program.

■ Development Tools

- Quartus Prime 25.3.1 Pro Edition
- Visual C++ 2019

■ Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIE_DDR4
- Visual C++ Project: Demonstrations\PCle_SW_KIT\Windows\PCle_DDR4

■ FPGA Application Design

Figure 3-21 shows the system block diagram in the FPGA system. In the **Platform Designer** (formerly Qsys), the PIO controller is used to control the LED and monitor the Button Status, and the On-Chip memory is used for performing DMA testing. The PIO controllers and the On-Chip memory are connected to the PCI Express Hard IP controller through the Memory-Mapped Interface.

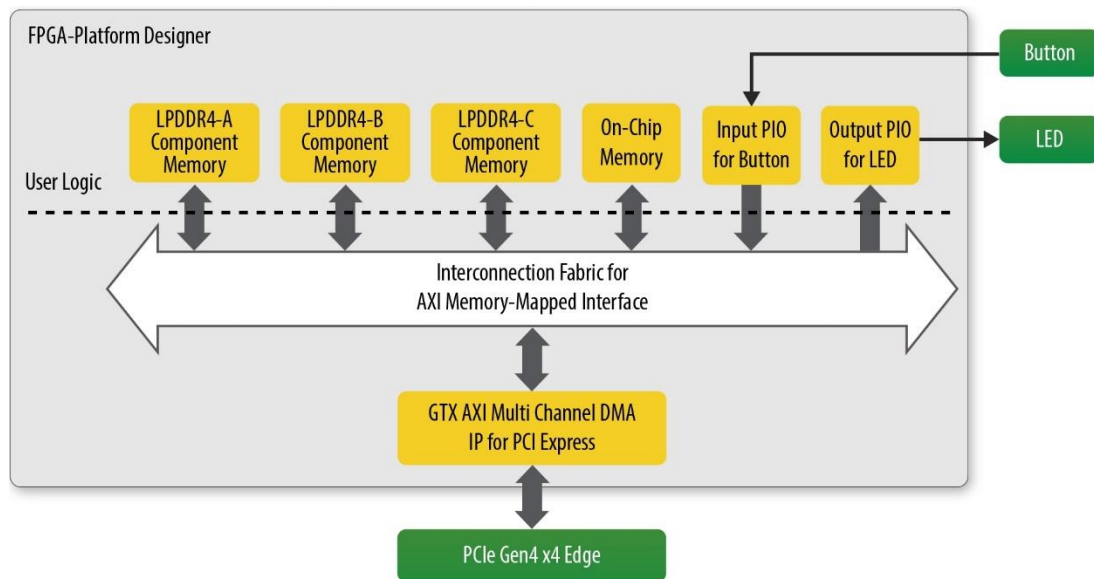


Figure 3-21 Hardware block diagram of the PCIe_DDR4 reference design

■ Windows Based Application Software Design

The application software project is built by Visual C++ 2019. The project includes the following major files:

Name	Description
PCIE_DDR4.cpp	Main program
PCIE.c	Implement dynamically load for TERASIC_PCIE_MCDMA.dll
PCIE.h	
TERASIC_PCIE_MCDMA.h	SDK library file, defines constant and data structure

The main program PCIE_DDR4.cpp includes the header file "PCIE.h" and defines the controller address according to the FPGA design.

```
#define DEMO_PCIE_USER_BAR          PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR       0x800000
#define DEMO_PCIE_IO_BUTTON_ADDR    0x800040
#define DEMO_PCIE_ONCHIP_MEM_ADDR    0x100000
#define DEMO_PCIE_LPDDR4A_MEM_ADDR   0x1000000000
#define DEMO_PCIE_LPDDR4B_MEM_ADDR   0x1100000000
#define DEMO_PCIE_LPDDR4C_MEM_ADDR   0x1200000000

#define ONCHIP_MEM_TEST_SIZE         (512*1024)           //512KB
#define LPDDR4A_MEM_TEST_SIZE         (4ull*1024*1024*1024) //4GB
#define LPDDR4B_MEM_TEST_SIZE         (4ull*1024*1024*1024) //4GB
#define LPDDR4C_MEM_TEST_SIZE         (4ull*1024*1024*1024) //4GB
```

The base address of BUTTON and LED controllers are 0x800040 and 0x800000 based

on PCIE_BAR4, respectively. The on-chip memory base address is 0x100000 relative to the DMA controller. **The above definitions are the same as those in the PCIe Fundamental demo.**

Before accessing the FPGA through PCI Express, the application first calls PCIE_Load to dynamically load the Terasic_PCIE_MCDMA.dll. Then, it calls PCIE_Open to open the PCI Express driver. The constant DEFAULT_PCIE_VID and DEFAULT_PCIE_DID used in the PCIE_Open are defined in Terasic_PCIE_MCDMA.h. If developers change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value defined in Terasic_PCIE_MCDMA.h. If the return value of PCIE_Open is zero, it means the driver cannot be accessed successfully. In this case, please make sure:

- The FPGA is configured with the associated bit-stream file and the host is rebooted.
- The PCI express driver is loaded successfully.

The LED control is implemented by calling PCIE_Write32 API, as shown below:

```
bPass = PCIE_Write32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_LED_ADDR, (uint32_t) Mask);
```

The button status query is implemented by calling the **PCIE_Read32** API, as shown below:

```
bPass = PCIE_Read32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_BUTTON_ADDR, &Status);
```

The memory-mapped memory read and write test is implemented by **PCIE_DmaWrite** and **PCIE_DmaRead** API, as shown below:

```
bPass = PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
bPass = PCIE_DmaRead(hPCIE, LocalAddr, pRead, nTestSize);
```

The PCIe link information is implemented by PCIE_ConfigRead32 API, as shown below:


```

// read config - link status
if (PCIE_ConfigRead32(hPCIE, 0x80, &Data32)) {
    switch ((Data32 >> 16) & 0x0F) {
        case 1:
            printf("Current Link Speed is Gen1\n");
            break;
        case 2:
            printf("Current Link Speed is Gen2\n");
            break;
        case 3:
            printf("Current Link Speed is Gen3\n");
            break;
        case 4:
            printf("Current Link Speed is Gen4\n");
            break;
        default:
            printf("Current Link Speed is Unknown\n");
            break;
    }
    switch ((Data32 >> 20) & 0x3F) {
        case 1:
            printf("Negotiated Link Width is x1\n");
            break;
        case 2:
            printf("Negotiated Link Width is x2\n");
            break;
        case 4:
            printf("Negotiated Link Width is x4\n");
            break;
        case 8:
            printf("Negotiated Link Width is x8\n");
            break;
        case 16:
            printf("Negotiated Link Width is x16\n");
            break;
        default:
            printf("Negotiated Link Width is Unknown\n");
            break;
    }
} else {
    bPass = false;
}

```

Chapter 4

PCI Express Reference Design for Linux

PCI Express is commonly used in consumer, server, and industrial applications, to link motherboard-mounted peripherals. From this demonstration, it will show how the PC Linux and FPGA communicate with each other through the PCI Express interface. GTS AXI Multichannel DMA IP for PCI Express IP is used in this demonstration. For detail about this IP, please refer to Altera document [ug-847470-857392](https://www.altera.com/docs/en/latest/ip/pci-express/gts-axi-multichannel-dma-ip-for-pci-express).

4.1 PCI Express System Infrastructure

Figure 4-1 shows the infrastructure of the PCI Express System in this demonstration. It consists of two primary components: FPGA System and PC System. The FPGA System is developed based on GTS AXI Multichannel DMA IP for PCI Express. The application software on the PC side is developed by Terasic based on Altera's PCIe kernel mode driver.

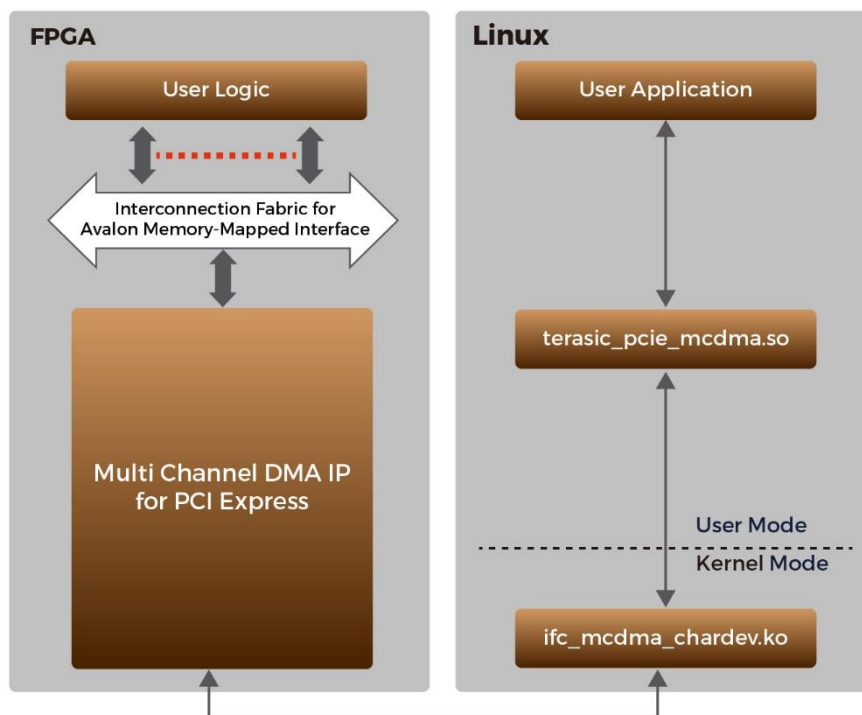


Figure 4-1 Infrastructure of PCI Express System

4.2 System Requirement

- Host PC: A host computer with an available PCIe 4.0 slot.
- Operating System: Ubuntu 24.04 LTS is recommended.
- Linux Kernel: Version 6.6 is required.

Note: Ubuntu 24.04 LTS might default to a newer kernel (e.g., v6.8). If your kernel version is newer than v6.6, you might need to manually install and configure your system to boot with Kernel v6.6.x. Please refer to the IP User Guide Section [6.3.3 Installing the Required Kernel Version \(if Working with Ubuntu 24.04\)](#) for instructions.

You can check your current kernel version using `uname -r`.

4.3 PC PCI Express Software SDK

The FPGA System CD contains a PC Linux based SDK to allow users to develop their 64-bit software application on 64-bits Linux. Ubuntu 24.04 is recommended. The SDK is located in the “CDROM/Demonstrations/FPGA/PCIe_SW_KIT/Linux” folder which includes:

- PCI Express Driver
- PCI Express Library
- PCI Express Examples

The kernel mode driver assumes the PCIe vendor ID (VID) is 0x1172 and the device ID (DID) is 0x09C4. If different VID and DID are used in the design, users need to modify the PCIe vendor ID (VID) and device ID (DID) in the driver config_file file accordingly.

The PCI Express Library is implemented as a single .so file named `terasic_pcie_mcdma.so`. This file is a 64-bit library file. With the library exported software API, users can easily communicate with the FPGA. The library provides the following functions:

- Basic data read and write
- Data read and write by DMA

For high performance data transmission, MCDMA is required as the read and write operations are specified under the hardware design on the FPGA.

■ Set the Boot Parameters

Follow the below step to modify the default hugepages setting in grub files:

1. Edit `/etc/default/grub` file.
2. Append the highlighted parameters in `GRUB_CMDLINE_LINUX` line in `/etc/default/grub` file.

```
GRUB_CMDLINE_LINUX=" rd.lvm.lv=centos/root\  
rd.lvm.lv=centos/swap rhgb default_hugepagesz=1G hugepagesz=1G\  
hugepages=5 panic=1 iommu=pt
```

3. For ubuntu 24.04:

After editing the `/etc/default/grub`, please update `grub.cfg` by "`sudo update-grub`"

4.4 PCI Express Software Stack

Figure 4-2 shows the software stack for the PCI Express application software on 64-bit Linux. The PCIe library module `terasic_pcie_mcdma.so` provides DMA and direct I/O access for user application program to communicate with FPGA. Users can develop their applications based on this `.so` library file. The `ifc_uio.ko` kernel driver is provided by Intel.

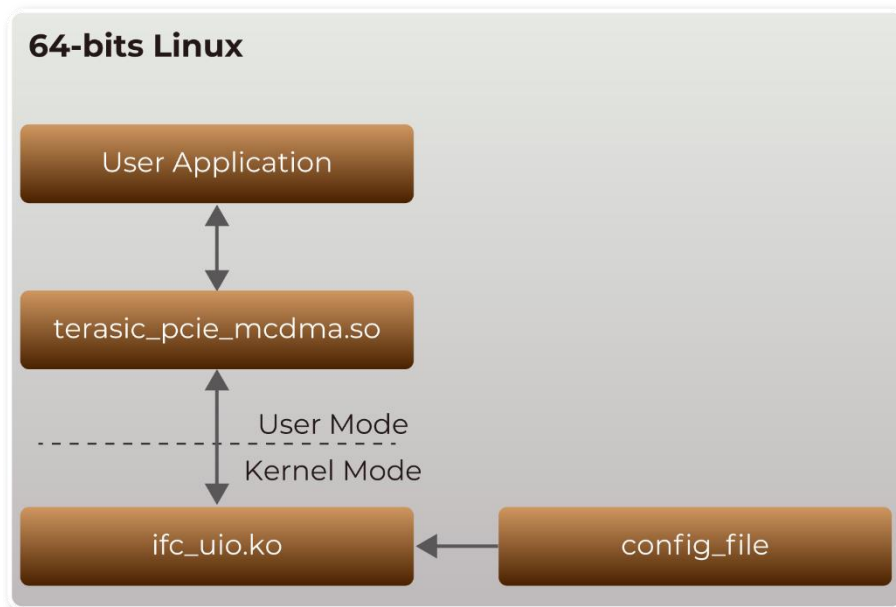


Figure 4-2 PCI Express Software Stack

■ Install PCI Express Driver on Linux

To make sure the PCIe driver can meet your kernel of Linux distribution, the driver `ifc_uio.ko` should be recompiled before it is used. The PCIe driver project is located in the folder:

"CDROM/Demonstrations/FPGA/PCIe_SW_KIT/Linux/PCIe_Driver"

The folder includes the following files:

- `ifc_pci_uio.c`
- `ifc_pci_uio.h`
- `common/include/regs/pio_reg_registers.h`
- `common/include/regs/qdma_regs_2_registers.h`
- `common/include/ifc_mcdma.h`
- `common/include/ifc_mcdma_utils.h`
- `common/include/mcdma_ip_params.h`
- `common/mk/common.mk`
- `common/mk/env.mk`
- `common/src/ifc_mcdma_utils.c`
- `Makefile`
- `load_driver`
- `unload`
- `config_file`

To compile and install the PCI Express driver, please execute the steps below:

1. Install the board the PCIe slot of the host PC
2. Make sure Quartus Programmer and USB-Blaster III driver are installed
3. Open a terminal and use "cd" command to go to the folder
"CDROM/Demonstrations/FPGA/PCIe_Fundamental/demo_batch".
4. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by typing the following commands in terminal. Replace "/home/user/altera_pro/25.3.1/quartus" to your quartus installation path.

`export QUARTUS_ROOTDIR=/home/user/altera_pro/25.3.1/quartus`
5. Execute "sudo -E sh test.sh" command to configure the FPGA
6. Restart the Linux operation system. In Linux, open a terminal and use "cd" command to go to the PCIe_Driver folder
7. Type the following commands to compile and install the driver `ifc_uio.ko`, and make sure driver is loaded successfully and FPGA is detected by the driver as shown in **Figure 4-3**.

- `make`

- sudo sh load_driver
- dmesg | tail -n 10

```
user@localhost:PCie_Driver$ sudo dmesg | tail -n 10
[ 9.545091] Bluetooth: RFCOMM TTY layer initialized
[ 9.545109] Bluetooth: RFCOMM socket layer initialized
[ 9.545114] Bluetooth: RFCOMM ver 1.11
[ 10.769681] rfkill: input handler disabled
[ 11.082787] igc 0000:06:00.0 eno1: NIC Link is Up 1000 Mbps Full Duplex, Flow
[ 98.225894] ifc_uio: loading out-of-tree module taints kernel.
[ 98.225901] ifc_uio: module verification failed: signature and/or required key
[ 98.226261] ifc_uio Intel(R) PCIe end point driver - version 1.0.0.2
[ 98.226264] Copyright (c) 2019-20, Intel Corporation.
[ 98.226306] ifc_uio 0000:01:00.0: enabling device (0000 -> 0002)
user@localhost:PCie_Driver$
```

Figure 4-3 Screenshot of install PCIe driver

■ Create a Software Application

All the files needed to create a PCIe software application are located in the directory CDROM/Demonstrations/FPGA/PCle_SW_KIT/Linux/PCle_Library. It includes the following files:

- Terasic_PCIE_MCDMA.h
- terasic_pcie_mcdma.so (64-bit library)

Below lists the procedures to use the library in users' C/C++ project:

1. Create a 64-bit C/C++ project.
2. Include Terasic_PCIE_MCDMA.h in the C/C++ project.
3. Copy terasic_pcie_mcdma.so to the folder where the project execution file is located.
4. Link terasic_pcie_mcdma.so in C/C++ program. To load the terasic_pcie_mcdma.so, please refer to the PCIe fundamental example below.
5. Call the library API to implement the desired application.

Users can easily communicate with the FPGA through the PCIe bus through the terasic_pcie_mcdma.so API. The details of API are described below sections.

4.5 PCI Express Library API

The API is the same as Windows Library. Please refer to the section **3.4 PCI Express Library API** in this document.

4.6 PCIe Reference Design Fundamental

The application reference design shows how to implement fundamental control and data transfer in the DMA. In the design, basic I/O is used to control the BUTTON and LED on the FPGA board. High-speed data transfer is performed by the DMA.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM/Demonstrations/FPGA/PCIe_Fundamental/demo_batch

The folder includes following files:

- FPGA Configuration File: golden_top.sof
- Download Batch file: test.sh
- Linux Application Software folder : linux_app, includes
 - ✧ PCIE_FUNDAMENTAL
 - ✧ terasic_pcie_mcdma.so

■ Demonstration Setup

1. Install the FPGA board on your PC.
2. Open a terminal and use "cd" command to goto "CDROM/Demonstrations/FPGA/PCIe_Fundamental/demo_batch".
3. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by tying the following commands in terminal. Replace /home/user/altera_pro/25.3.1/quartus to your quartus installation path.

```
export QUARTUS_ROOTDIR=/home/user/altera_pro/25.3.1/quartus
```

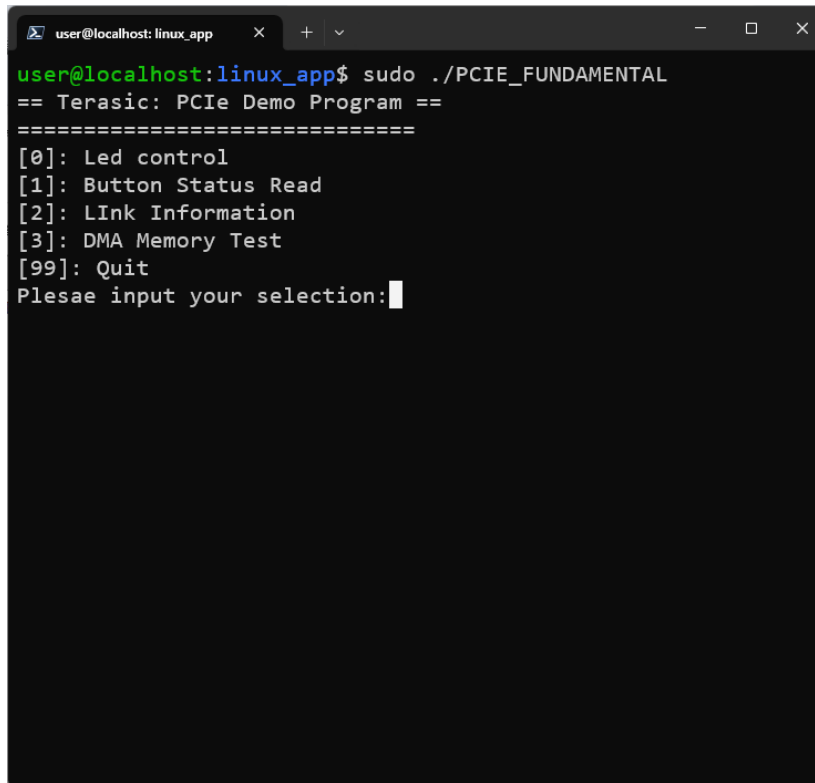
4. Execute "sudo -E sh test.sh" command to configure the FPGA
5. Restart Linux
6. Install PCIe driver. The driver is located in the folder:

CDROM/Demonstration/PCIe_SW_KIT/Linux/PCIe_Driver.
7. Type "lspci -nn | grep 1172:09c4" to make sure the Linux has detected the FPGA board as shown in **Figure 4-4** below.

```
user@localhost:PCIe_Driver$ lspci -nn | grep 1172:09c4
01:00.0 Unassigned class [ff00]: Altera Corporation Device [1172:09c4] (rev 01)
```

Figure 4-4 lspci output

8. Goto linux_app folder, execute "sudo ./PCIE_FUNDAMENTAL". A menu will appear as shown in **Figure 4-5**.

A terminal window titled 'user@localhost: linux_app' with standard window controls. The terminal shows the command 'sudo ./PCIE_FUNDAMENTAL' being executed. The output is a menu titled '== Terasic: PCIe Demo Program ==' with a list of options: '[0]: Led control', '[1]: Button Status Read', '[2]: LInk Information', '[3]: DMA Memory Test', and '[99]: Quit'. The prompt 'Plesae input your selection:' is followed by a cursor.

```
user@localhost:linux_app$ sudo ./PCIE_FUNDAMENTAL
== Terasic: PCIe Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:█
```

Figure 4-5 Screenshot of Program Menu

9. Type 0 followed by the ENTER key to select the Led Control item, then input 3 (hex 0x03) will turn all leds on as shown in [Figure 4-6](#). If input 0 (hex 0x00), all led will be turned off.


```
user@localhost: linux_app
user@localhost:linux_app$ sudo ./PCIE_FUNDAMENTAL
== Terasic: PCie Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:0
Please input led control mask:3
Led control success, mask=3h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-6 Screenshot of LED Control

10. Type 1 followed by the ENTER key to select the Button Status Read item. The button status will be reported as shown in [Figure 4-7](#).

```
user@localhost: linux_app
user@localhost:linux_app$ sudo ./PCIE_FUNDAMENTAL
== Terasic: PCie Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:0
Please input led conrol mask:3
Led control success, mask=3h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:1
Button status mask:=0h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-7 Screenshot of Button Status Report

11. Type 3 followed by the ENTER key to select the DMA Testing item. The DMA test result will be reported as shown in **Figure 4-8**.

```
user@localhost: linux_app
Plesae input your selection:0
Please input led conrol mask:3
Led control success, mask=3h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:1
Button status mask:=0h
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:3
DMA-Memory (Size = 524288 bytes) pass
=====
[0]: Led control
[1]: Button Status Read
[2]: LInk Information
[3]: DMA Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-8 Screenshot of DMA Memory Test Result

12. Type 99 followed by the ENTER key to exit this test program

■ Development Tools

- Quartus Prime 25.3.1 Pro Edition
- GNU Compiler Collection, Version 13.3.0 is recommended

■ Demonstration Source Code Location

- Quartus Project: Demonstrations/PCle_Fundamental
- C++ Project: Demonstrations/PCle_SW_KIT/Linux/PCIE_FUNDAMENTAL

■ FPGA Application Design

Figure 4-9 shows the system block diagram in the FPGA system. In the **Platform Designer** (formerly Qsys), the PIO controller is used to control the LED and monitor the Button Status, and the On-Chip memory is used for performing DMA testing. The PIO controllers and the On-Chip memory are connected to the PCI Express Hard IP controller through the Memory-Mapped Interface.

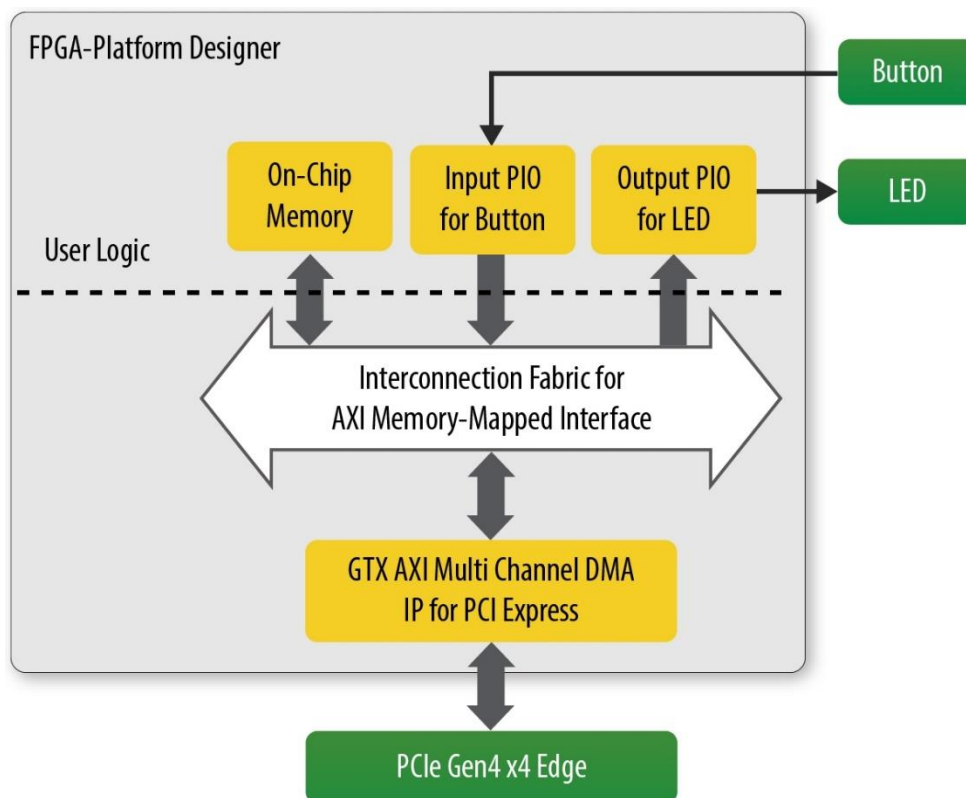


Figure 4-9 Hardware block diagram of the PCIe reference design

■ Linux Based Application Software Design

The application software project is built by GNU Toolchain. The project includes the following major files:

Name	Description
PCIE_FUNDAMENTAL.cpp	Main program
TERASIC_PCIE_MCDMA.h	SDK library file, defines constant and data structure

The main program PCIE_FUNDAMENTAL.cpp defines the controller address according to the FPGA design.

```
#define DEMO_PCIE_USER_BAR      PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR   0x800000
#define DEMO_PCIE_IO_BUTTON_ADDR 0x800040
#define DEMO_PCIE_MEM_ADDR      0x100000

#define MEM_SIZE                 (512*1024) //512KB
```

The base address of BUTTON and LED controllers are 0x800040 and 0x800000 based on PCIE_BAR4, respectively. The on-chip memory base address is 0x100000 relative to the DMA controller.

Before accessing the FPGA through PCI Express, the application first calls PCIE_Load to dynamically load the terasic_pcie_mcdma.so. Then, it call PCIE_Open to open the PCI Express driver. The constant DEFAULT_PCIE_VID and DEFAULT_PCIE_DID used in PCIE_Open are defined in TERASIC_PCIE_MCDMA.h. If developers change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value defined in TERASIC_PCIE_MCDMA.h. If the return value of PCIE_Open is zero, it means the driver cannot be accessed successfully. In this case, please make sure:

- The FPGA is configured with the associated bit-stream file and the host is rebooted.
- The PCI express driver is loaded successfully.

The LED control is implemented by calling PCIE_Write32 API, as shown below:

```
bPass = PCIE_Write32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_LED_ADDR, (uint32_t) Mask);
```

The button status query is implemented by calling the **PCIE_Read32** API, as shown below:

```
bPass = PCIE_Read32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_BUTTON_ADDR, &Status);
```

The memory-mapped memory read and write test is implemented by **PCIE_DmaWrite** and **PCIE_DmaRead** API, as shown below:

```
bPass = PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
bPass = PCIE_DmaRead(hPCIE, LocalAddr, pRead, nTestSize);
```

4.7 PCIe Reference Design - DDR4

The application reference design shows how to add LPDDR4 Memory Controllers for the LPDDR4-A, LPDDR4-B and LPDDR4-C Components into the PCIe Quartus project and perform 4GB data DMA for all components. Also, this demo shows how to call "PCIE_ConfigRead32" API to check PCIe link status.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM\Demonstrations\FPGA\PCIe_DDR4\demo_batch

The folder includes following files:

- FPGA Configuration File: golden_top.sof
- Download Batch file: test.sh
- Linux Application Software folder : linux_app, includes
 - ✧ PCIE_DDR4
 - ✧ terasic_pcie_mcdma.so

■ Demonstration Setup

1. Install the FPGA board on the PCIe Slot of your PC.
2. Open a terminal and use "cd" command to go to "CDROM/Demonstrations/FPGA/PCIe_DDR4/demo_batch".
3. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by typing the following commands in the terminal. Replace /home/user/altera_pro/25.3.1/quartus to your Quartus installation path.

```
export QUARTUS_ROOTDIR=/home/user/altera_pro/25.3.1/quartus
```

4. Execute "sudo -E sh test.sh" command to configure the FPGA
5. Restart Linux
6. Install PCIe driver.
7. Make sure that Linux has detected the FPGA Board.
8. Go to the linux_app folder, execute "sudo ./PCIE_DDR4". A menu will appear as shown in **Figure 4-10**.

```
user@localhost: linux_app
user@localhost:linux_app$ sudo ./PCIE_DDR4
== Terasic: PCIE Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-10 Screenshot of Program Menu

9. Type 2 followed by the ENTER key to select the Link Info item. The PCIe link information will be shown as in [Figure 4-11](#). Gen4 link speed and x4 link width are expected.

```
user@localhost: linux_app
== Terasic: PCIe Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:2
Vender ID:1172h
Device ID:09C4h
Current Link Speed is Gen4
Negotiated Link Width is x4
Maximum Payload Size is 256-byte
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-11 Screenshot of Link Info

10. Type 3 followed by the ENTER key to select DMA On-Chip Memory Test item. The DMA write and read test result will be report as shown in [Figure 4-12](#).

```
user@localhost: linux_app
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:3
DMA Memory Test, Address = 0x100000, Size = 0x80000 Bytes..
.
Generate Test Pattern...
DMA Write...
DMA Read... (100000 - 180000)
Readback Data Verify...
DMA-Memory Address = 0x100000, Size = 0x80000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-12 Screenshot of On-Chip Memory DMA Test Result

11. Type 4 followed by the ENTER key to select the DMA LPDDR4-A component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 4-13**.


```
user@localhost: linux_app
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:4
DMA Memory Test, Address = 0x100000000, Size = 0x10000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (100000000 - 110000000)
Readback Data Verify...
DMA-Memory Address = 0x100000000, Size = 0x10000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:█
```

Figure 4-13 Screenshot of LPDDR4-A component Memory DAM Test Result

12. Type 5 followed by the ENTER key to select the DMA LPDDR4-B Component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 4-14**.

```
user@localhost: linux_app
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:5
DMA Memory Test, Address = 0x110000000, Size = 0x10000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (110000000 - 120000000)
Readback Data Verify...
DMA-Memory Address = 0x110000000, Size = 0x10000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-14 Screenshot of LPDDR4-B Component Memory DAM Test Result

13. Type 6 followed by the ENTER key to select the DMA LPDDR4-C Component Memory Test item. The DMA write and read test result will be reported as shown in **Figure 4-15**.

```
user@localhost: linux_app
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:6
DMA Memory Test, Address = 0x1200000000, Size = 0x100000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read... (1200000000 - 1300000000)
Readback Data Verify...
DMA-Memory Address = 0x1200000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA LPDDR4-A Component Memory Test
[5]: DMA LPDDR4-B Component Memory Test
[6]: DMA LPDDR4-C Component Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 4-15 Screenshot of LPDDR4-C Component Memory DAM Test Result

14. Type 99 followed by the ENTER key to exit this test program.

■ Development Tools

- Quartus Prime 25.3.1 Pro Edition
- GNU Compiler Collection, Version 13.3.0 is recommended

■ Demonstration Source Code Location

- Quartus Project: Demonstrations/PCIE_DDR4
- C++ Project: Demonstrations/PCle_SW_KIT/Linux/PCle_DDR4

■ FPGA Application Design

Figure 4-16 shows the system block diagram in the FPGA system. In the **Platform Designer** (formerly Qsys), the PIO controller is used to control the LED and monitor the Button Status, and the On-Chip memory is used for performing DMA testing. The PIO controllers and the On-Chip memory are connected to the PCI Express Hard IP controller through the Memory-Mapped Interface.

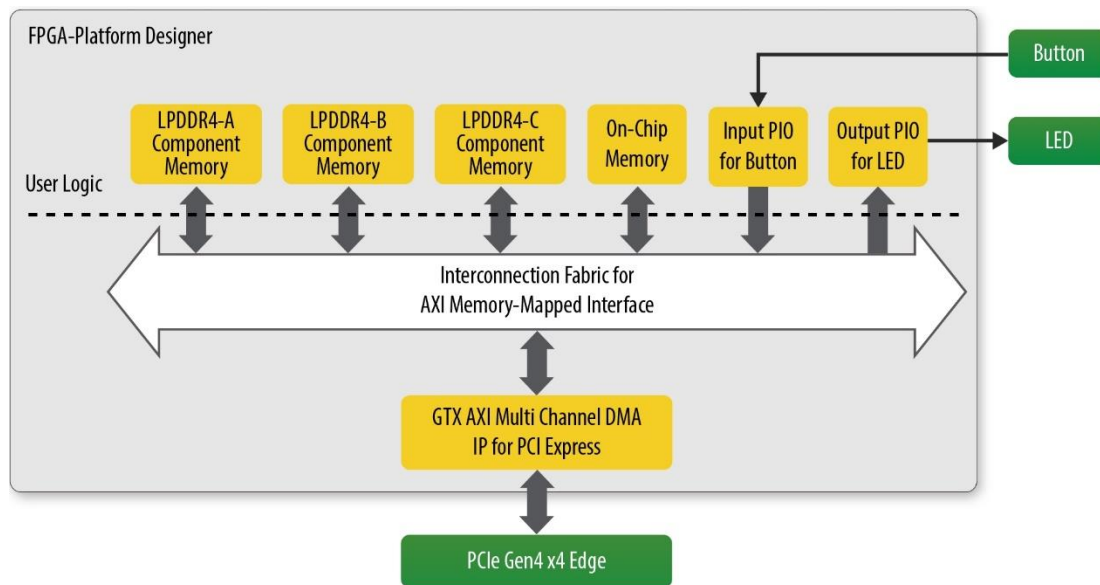


Figure 4-16 Hardware block diagram of the PCIe_DDR4 reference design

Linux Based Application Software Design

The application software project is built by GNU Toolchain. The project includes the following major files:

Name	Description
PCIE_DDR4.cpp	Main program
TERASIC_PCIE_MCDMA.h	SDK library file, defines constant and data structure

The main program PCIE_DDR4.cpp defines the controller address according to the FPGA design.

```

#define DEMO_PCIE_USER_BAR          PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR       0x800000
#define DEMO_PCIE_IO_BUTTON_ADDR    0x800040
#define DEMO_PCIE_ONCHIP_MEM_ADDR   0x100000
#define DEMO_PCIE_DDR4A_MEM_ADDR    0x1000000000
#define DEMO_PCIE_DDR4B_MEM_ADDR    0x1200000000

#define ONCHIP_MEM_TEST_SIZE        (512*1024) //512KB
#define DDR4A_MEM_TEST_SIZE         (8ull*1024*1024*1024) //8GB
#define DDR4B_MEM_TEST_SIZE         (8ull*1024*1024*1024) //8GB

```

The base address of BUTTON and LED controllers are 0x800040 and 0x800000 based on PCIE_BAR4, respectively. The on-chip memory base address is 0x100000 relative to the DMA controller. The above definition is the same as those in PCIe Fundamental

demo.

Before accessing the FPGA through PCI Express, the application first calls the `PCIE_Load` to dynamically load the `terasic_pcie_mcdma.so`. Then, it calls the `PCIE_Open` to open the PCI Express driver. The constant `DEFAULT_PCIE_VID` and `DEFAULT_PCIE_DID` used in the `PCIE_Open` are defined in `TERASIC_PCIE_MCDMA.h`. If developers changes the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value defined in `TERASIC_PCIE_MCDMA.h`. If the return value of the `PCIE_Open` is zero, it means the driver cannot be accessed successfully. In this case, please make sure:

- The FPGA is configured with the associated bit-stream file and the host is rebooted.
- The PCI express driver is loaded successfully.

The LED control is implemented by calling `PCIE_Write32` API, as shown below:

```
bPass = PCIE_Write32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_LED_ADDR, (uint32_t) Mask);
```

The button status query is implemented by calling the **`PCIE_Read32`** API, as shown below:

```
bPass = PCIE_Read32(hPCIE, DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_BUTTON_ADDR, &Status);
```

The memory-mapped memory read and write test is implemented via **`PCIE_DmaWrite`** and the **`PCIE_DmaRead`** API, as shown below:

```
bPass = PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
bPass = PCIE_DmaRead(hPCIE, LocalAddr, pRead, nTestSize);
```

The PCIe link information is implemented by `PCIE_ConfigRead32` API, as shown below:

```

// read config - link status
if (PCIE_ConfigRead32(hPCIE, 0x80, &Data32)) {
    switch ((Data32 >> 16) & 0x0F) {
        case 1:
            printf("Current Link Speed is Gen1\n");
            break;
        case 2:
            printf("Current Link Speed is Gen2\n");
            break;
        case 3:
            printf("Current Link Speed is Gen3\n");
            break;
        case 4:
            printf("Current Link Speed is Gen4\n");
            break;
        default:
            printf("Current Link Speed is Unknown\n");
            break;
    }
    switch ((Data32 >> 20) & 0x3F) {
        case 1:
            printf("Negotiated Link Width is x1\n");
            break;
        case 2:
            printf("Negotiated Link Width is x2\n");
            break;
        case 4:
            printf("Negotiated Link Width is x4\n");
            break;
        case 8:
            printf("Negotiated Link Width is x8\n");
            break;
        case 16:
            printf("Negotiated Link Width is x16\n");
            break;
        default:
            printf("Negotiated Link Width is Unknown\n");
            break;
    }
} else {
    bPass = false;
}

```

Chapter 5

Transceiver Verification

This chapter describes how to quickly verify the FPGA transceivers via the QSFP+ connector.

5.1 Transceiver Test Code

The transceiver test code is used to verify the transceiver channels for the QSFP+ ports through an external loopback method. The transceiver channels are verified with the data rates 10.3125 Gbps with PRBS31 test pattern

5.2 Loopback Fixture

To enable an external loopback of the transceiver channels, QSFP+ loopback fixtures, as shown in **Figure 5-1**, are required. This loopback fixture is manufactured by **Molex**, with part number **74763-0025**.



Figure 5-1 QSFP+ Loopback Cable

Figure 5-2 shows the FPGA board with one QSFP+ loopback fixture installed.

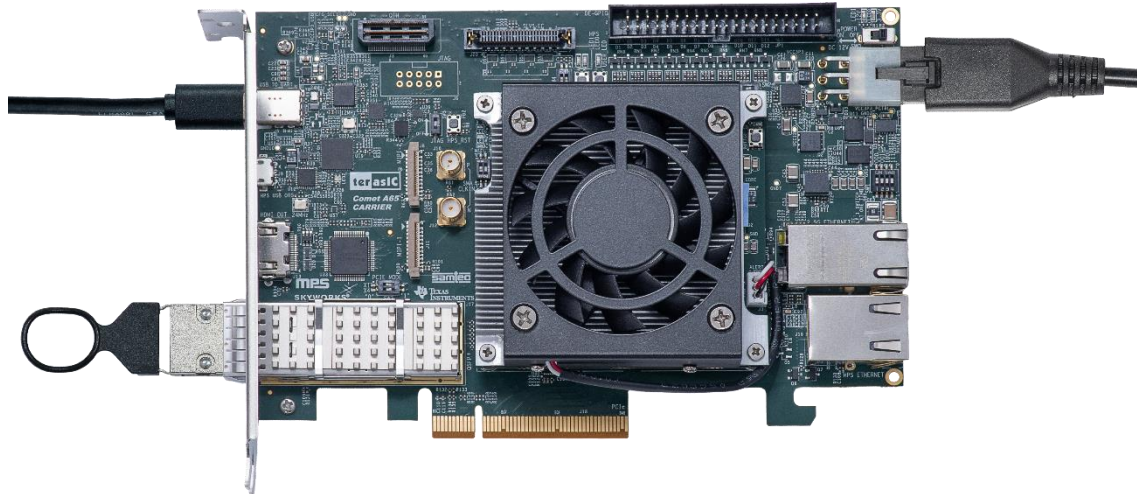


Figure 5-2 QSFP+ Loopback fixture installed on the board

5.3 Testing by Transceiver Test Code

The transceiver test code is available in the folder System CD\Tool\Transceiver_Test.

Figure 5-3, **Figure 5-4** and **Figure 5-5** shows the GTS PMA/FEC Direct PHY settings in the test code. The data rate of each transceiver channel is set to 10312.5 Mbps and the PMA modulation type is NRZ. So the 40Gbps QSFP+ loopback test code is implemented (4 channels in total). Also, the GTS Reference and System PLL setting is shown in **Figure 5-6**.

GTS PMA/FEC Direct PHY IP

intel_directphy_gts

Mode

Direct PHY Operation Mode: Basic Configuration

Common Datapath Options

PMA configuration rules: Basic

Number of PMA lanes: 4

Datapath clocking mode: System PLL

System/HVIO PLL frequency: 322.265625 MHz

PMA mode: Duplex

PMA data rate: 10312.5 Mbps

PMA parallel clock frequency: 322.265625 MHz

PMA width: 32

☐ Provide separate interface for each PMA

☐ Enable refclock to core

TX/RX Common PMA Options

Loopback mode: disabled

Figure 5-3 The Transceiver PHY setting

TX Datapath Options	RX Datapath Options	FEC Options	PCS Options	Avalon Memory-Mapped Interface	Ex
TX PMA					
PRBS generator mode:		DISABLE			
TX PLL Settings					
Output frequency:		5156.250000		MHz	
VCO frequency:		10312.500000		MHz	
<input type="checkbox"/> Enable TX PLL cascade mode					
<input type="checkbox"/> Enable TX PLL fractional mode					
TX PLL integer mode reference clock frequency:		156.250000		MHz	
TX PLL fractional mode reference clock frequency:		156.250000		MHz	
TX PMA Interface					
TX PMA interface FIFO mode:		Elastic			
<input type="checkbox"/> Enable tx_pmaif_fifo_empty port					
<input type="checkbox"/> Enable tx_pmaif_fifo_pempty port					
<input type="checkbox"/> Enable tx_pmaif_fifo_pfull port					
TX Core Interface					
TX Core Interface FIFO					
<input type="checkbox"/> Enable custom cadence generation ports and logic					
<input checked="" type="checkbox"/> Enable tx_cadence_slow_clk_locked port					
TX core interface FIFO Mode:		Phase compensation			
<input checked="" type="checkbox"/> Enable TX double width transfer					
<input type="checkbox"/> Enable tx_fifo_full port					
<input type="checkbox"/> Enable tx_fifo_empty port					
<input type="checkbox"/> Enable tx_fifo_pfull port					
<input type="checkbox"/> Enable tx_fifo_pempty port					
TX Clock Options					
Selected tx_clkout clock source:		Sys/HVIO PLL Clock			
tx_clkout clock div by:		2			
Frequency of tx_clkout:		161.132812		MHz	
<input type="checkbox"/> Enable tx_clkout2 port					
TX User Clock Setting					
TX user clock div by:		100			
TX user clock Frequency:		103.125		MHz	

Figure 5-4 Transceiver TX Data Path Options

TX Datapath Options	RX Datapath Options	FEC Options	PCS Options	Avalon Memory-Mapped Int
▼ RX PMA				
PRBS monitor mode:		DISABLE		
<input type="checkbox"/> Enable rx_cdr_divclk				
▼ RX CDR Settings				
Output frequency:		5156.250000	MHz	
VCO frequency:		10312.500000	MHz	
RX CDR reference clock frequency:		156.250000	MHz	
CDR lock mode:		auto		
<input type="checkbox"/> Enable rx_set_locktohref port				
▼ RX PMA Interface				
RX PMA interface FIFO mode:		Elastic		
<input type="checkbox"/> Enable rx_pmaif_fifo_empty port				
<input type="checkbox"/> Enable rx_pmaif_fifo_pempty port				
<input type="checkbox"/> Enable rx_pmaif_fifo_pfull port				
▼ RX Core Interface				
▼ RX Core Interface FIFO				
RX core interface FIFO mode:		Phase compensation		
<input checked="" type="checkbox"/> Enable RX double width transfer				
<input type="checkbox"/> Enable rx_fifo_full port				
<input type="checkbox"/> Enable rx_fifo_empty port				
<input type="checkbox"/> Enable rx_fifo_pfull port				
<input type="checkbox"/> Enable rx_fifo_pempty port				
<input type="checkbox"/> Enable rx_fifo_rd_en port				
▼ RX Clock Options				
Selected rx_clkout clock source:		sys/HVIO PLL clock		
rx_clkout clock div by:		2		
Frequency of rx_clkout:		161.132812	MHz	
<input type="checkbox"/> Enable rx_clkout2 port				
▼ RX User Clock Setting				
RX user clock div by:		100		
RX user clock Frequency:		103.125	MHz	

Figure 5-5 Transceiver RX Data Path Options

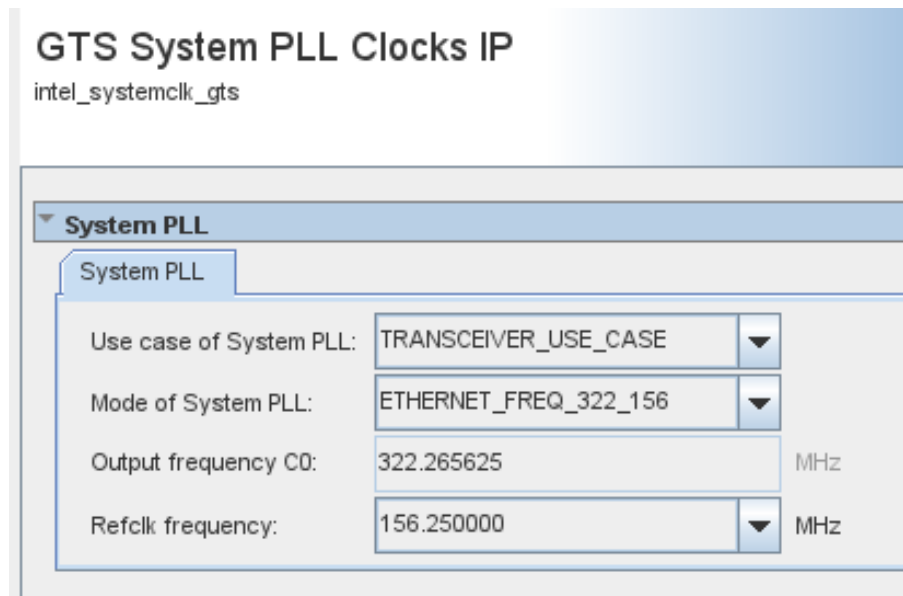


Figure 5-6 The GTS System PLL Clock setting

The FPGA transceiver PMA setting used are shown in the table below.

Direction	Item	Value
TX	pre_tap_2	0
	pre_tap_1	0
	main_tap	45
	post_tap_1	4
RX	Auto Default	

Here are the procedures to perform transceiver channel test:

1. Copy the Transceiver_Test folder to your local disk.
2. Ensure that the FPGA board is NOT powered on.
3. Plug-in the QSFP+ loopback fixtures.
4. Connect your FPGA board to your PC with an USB cable.
5. Power on the FPGA board
6. Execute "test.bat" in the Transceiver_Test folder under your local disk.
7. The batch file will download .sof and .elf files, and start the test immediately.
The test result is shown in the command shell terminal, as shown in **Figure 5-7**. Note, the result show "error count = 0" means the test is pass.
8. To terminate the test, press the BUTTON on the SOM module.

```
C:\Windows\system32\cmd.exe
Transceiver for QSFP+ testing...
Press the button on the SOM board can terminate the testing.
Init...
===== Time Elapsed (d h:m:s): 0 0:0:0 =====
QSFP 4B xcvr 10.3125Gpbs x4, Numbers of bit tested: (0.000E+00) x4
CH-0 error count:0
CH-1 error count:0
CH-2 error count:0
CH-3 error count:0

===== Time Elapsed (d h:m:s): 0 0:0:5 =====
QSFP 4B xcvr 10.3125Gpbs x4, Numbers of bit tested: (5.156E+10) x4
CH-0 error count:0
CH-1 error count:0
CH-2 error count:0
CH-3 error count:0

===== Time Elapsed (d h:m:s): 0 0:0:10 =====
QSFP 4B xcvr 10.3125Gpbs x4, Numbers of bit tested: (1.031E+11) x4
CH-0 error count:0
CH-1 error count:0
CH-2 error count:0
CH-3 error count:0
```

Figure 5-7 QSFP+ Transceiver Loopback Test in Progress

Chapter 6

Additional Information

6.1 Getting Help

Here are the addresses where you can get help if you encounter problems:

■ Terasic Technologies

No.80, Fenggong Rd., Hukou Township, Hsinchu County 303035. Taiwan

Email: support@terasic.com

Web: www.terasic.com

Comet A65 Evaluation Kit Web: <http://comet-a65.terasic.com/>

■ Revision History

Date	Version	Changes
2025.11	First publication	
2025.12	V1.0	Major revision based on comprehensive review and proofreading corrections
2025.12	V2.0	New added Chapter 4&5 for PCIe reference design
2026.04	V2.1	Upgrade Quartus v25.3 to v25.3.1, update few figures for 12V 2x3 power connector