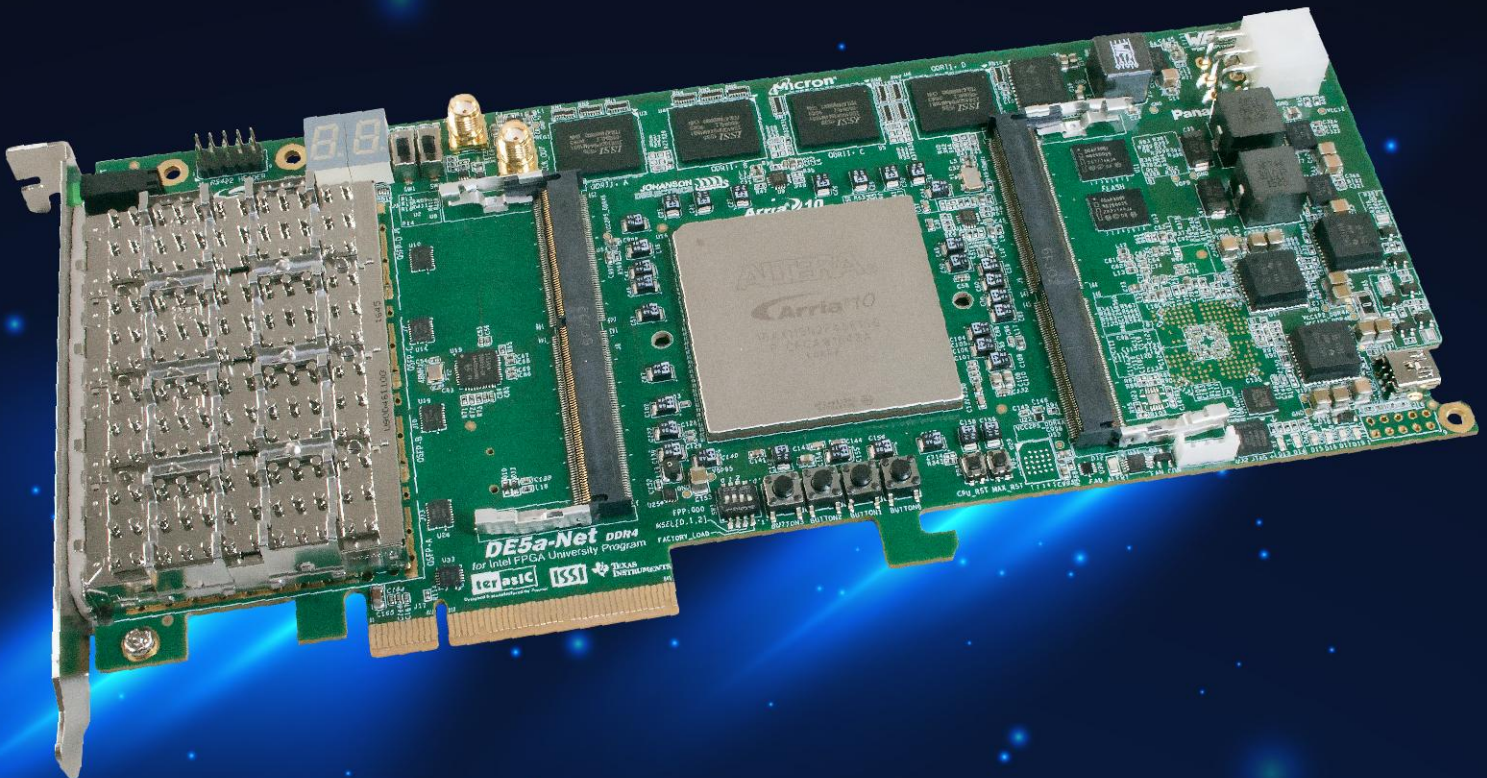


DE5a-Net

DDR4 Edition

FPGA Development Kit User Manual



Copyright © 2003-2017 Terasic Inc. All Rights Reserved.



Contents

| | |
|---|-----------|
| Chapter 1 Overview..... | 5 |
| 1.1 General Description..... | 5 |
| 1.2 Key Features..... | 6 |
| 1.3 Block Diagram..... | 7 |
| 1.4 Operating Temperature - Important..... | 11 |
| | |
| Chapter 2 Board Components | 12 |
| 2.1 Board Overview..... | 12 |
| 2.2 Configuration, Status and Setup..... | 13 |
| 2.3 General User Input/Output | 17 |
| 2.4 Temperature Sensor and Fan Control..... | 21 |
| 2.5 Power Monitor | 23 |
| 2.6 Clock Circuit | 24 |
| 2.7 FLASH Memory..... | 27 |
| 2.8 DDR4 SO-DIMM..... | 30 |
| 2.9 QDRII+ SRAM..... | 42 |
| 2.10 QSPF+ Ports..... | 52 |
| 2.11 PCI Express | 56 |
| 2.12 RS-422 Expansion Header..... | 59 |
| | |
| Chapter 3 System Builder | 61 |
| 3.1 Introduction..... | 61 |
| 3.2 General Design Flow..... | 63 |
| 3.3 Using System Builder | 64 |



Chapter 4 Flash Programming..... 71

| | |
|--|----|
| 4.1 CFI Flash Memory Map | 71 |
| 4.2 FPGA Configure Operation..... | 72 |
| 4.3 Flash Programming with Users Design..... | 73 |
| 4.4 Restore Factory Settings | 75 |

Chapter 5 Peripheral Reference Design 77

| | |
|--|----|
| 5.1 Temperature Monitor: Board Protection | 77 |
| 5.2 Configure Si5340A/B in RTL..... | 80 |
| 5.3 Nios II control for Si5340/Temperature/Power | 89 |
| 5.4 Fan Speed Control | 94 |

Chapter 6 Memory Reference Design 98

| | |
|--------------------------------------|-----|
| 6.1 QDRII+ SRAM Test | 98 |
| 6.2 DDR4 SDRAM Test | 102 |
| 6.3 DDR4 SDRAM Test by Nios II | 104 |

Chapter 7 PCI Express Design for Windows 108

| | |
|---|-----|
| 7.1 PCI Express System Infrastructure..... | 108 |
| 7.2 PC PCI Express Software SDK | 109 |
| 7.3 PCI Express Software Stack | 110 |
| 7.4 PCI Express Library API | 116 |
| 7.5 PCIe Reference Design - Fundamental | 121 |
| 7.6 PCIe Reference Design - DDR4 | 129 |

Chapter 8 PCI Express Reference Design for Linux.... 136



| | |
|---|-----|
| 8.1 PCI Express System Infrastructure..... | 136 |
| 8.2 PC PCI Express Software SDK | 137 |
| 8.3 PCI Express Software Stack | 138 |
| 8.4 PCI Express Library API | 141 |
| 8.5 PCIe Reference Design – Fundamental | 141 |
| 8.6 PCIe Reference Design - DDR4 | 148 |

Chapter 9 Transceiver Verification 156

| | |
|---|-----|
| 9.1 Function of the Transceiver Test Code | 156 |
| 9.2 Loopback Fixture..... | 156 |
| 9.3 Testing..... | 159 |
| 9.4 40G Ethernet Example | 161 |
| 9.5 10GBASE-R Ethernet Example..... | 165 |

Additional Information..... 169

| | |
|--------------------|-----|
| Getting Help | 169 |
|--------------------|-----|

This chapter provides an overview of the DE5a-Net DDR4 Edition Development Board and installation guide.

1.1 General Description

The Terasic DE5a-Net DDR4 Edition Arria 10 GX FPGA Development Kit provides the ideal hardware solution for designs that demand high capacity and bandwidth memory interfacing, ultra-low latency communication, and power efficiency. With a full-height, 3/4-length form-factor package, the DE5a-Net is designed for the most demanding high-end applications, empowered with the top-of-the-line Altera Arria 10 GX, delivering the best system-level integration and flexibility in the industry.

The Arria® 10 GX FPGA features integrated transceivers that transfer at a maximum of 12.5 Gbps, allowing the DE5a-Net to be fully compliant with version 3.0 of the PCI Express standard, as well as allowing an ultra low-latency, straight connections to four external 40G QSFP+ modules. Not relying on an external PHY will accelerate mainstream development of network applications enabling customers to deploy designs for a broad range of high-speed connectivity applications. For designs that demand high capacity and high speed for memory and storage, the DE5a-Net delivers with two independent banks of DDR4 SO-DIMM RAM, four independent banks of QDRII+ SRAM, high-speed parallel flash memory. The feature-set of the DE5a-Net fully supports all high-intensity applications such as low-latency trading, cloud computing, high-performance computing, data acquisition, network processing, and

signal processing.

1.2 Key Features

The following hardware is implemented on the DE5a-Net board:

■ FPGA

- Intel Arria® 10 GX FPGA (10AX115N2F45E1SG)

■ FPGA Configuration

- On-Board USB Blaster II or JTAG header for FPGA programming
- Fast passive parallel (FPPx32) configuration via MAX II CPLD and flash memory

■ General user input/output:

- 8 LEDs
- 4 push-buttons
- 2 slide switches
- 2 seven-segment displays

■ Clock System

- 50MHz Oscillator
- Programmable clock generators Si5340A and Si5340B
- One SMA connector for external clock input
- One SMA connector for clock output

■ Memory

- DDR4 SO-DIMM SDRAM
- QDRII+ SRAM



- FLASH

■ Communication Ports

- Four QSFP+ connectors
- PCI Express (PCIe) x8 edge connector
- One RS422 transceiver with Expansion Header

■ System Monitor and Control

- Temperature sensor
- Fan control
- Power monitor

■ Power

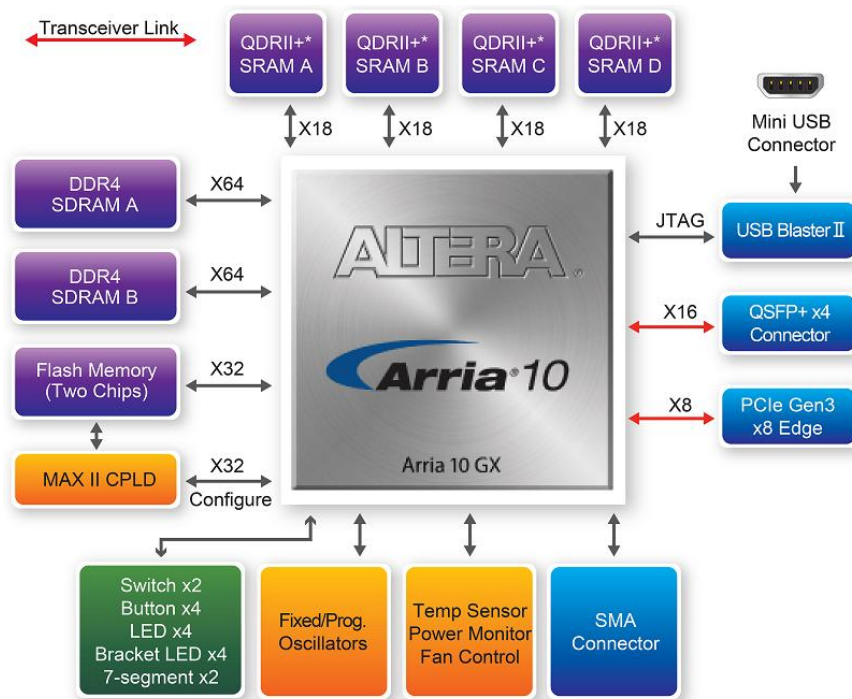
- PCI Express 6-pin power connector, 12V DC Input
- PCI Express edge connector power

■ Mechanical Specification

- PCI Express full-height and 3/4-length

1.3 Block Diagram

Figure 1-1 shows the block diagram of the DE5a-Net board. To provide maximum flexibility for the users, all key components are connected with the Arria 10 GX FPGA device. Thus, users can configure the FPGA to implement any system design.



*Cypress QDRII+ SRAM or functional compatible SRAMS provided by GSI (SigmaQuad-II+) and ISSI (QUADP).

Figure 1-1 Block diagram of the DE5a-Net board

Below is more detailed information regarding the blocks in **Figure 1-1**.

■ Arria 10 GX FPGA

- 10AX115N2F45E1SG
- 1,150K logic elements (LEs)
- 67-Mbits embedded memory
- 48 transceivers (12.5Gbps)
- 3,036 18-bit x 19-bit multipliers
- 1,518 Variable-precision DSP blocks

- 4 PCI Express hard IP blocks
- 768 user I/Os
- 384 LVDS channels
- 32 phase locked loops (PLLs)

■ JTAG Header and FPGA Configuration

- On-board USB Blaster II or JTAG header for use with the Quartus Prime Programmer
- MAXII CPLD 5M2210 System Controller and Fast Passive Parallel (FPP x32) configuration

■ Memory Devices

- 32MB QDRII+ SRAM
- Up to 16GB DDR4 SO-DIMM SDRAM for each DDR4 socket
- 256MB FLASH

■ General User I/O

- 8 user controllable LEDs
- 4 user push buttons
- 2 user slide switches
- 2 seven-segment displays

■ On-Board Clock

- 50MHz oscillator
- Programming PLL providing clock for 40G QSFP+ transceiver
- Programming PLL providing clock for PCIe transceiver
- Programming PLL providing clocks for DDR4 SDRAM and QDRII+ SRAM

■ Four QSFP+ Ports

- Four QSFP+ connector (40 Gbps+)

■ PCI Express x8 Edge Connector

- Support for PCIe x8 Gen1/2/3
- Edge connector for PC motherboard with x8 or x16 PCI Express slot

■ Power Source

- PCI Express 6-pin DC 12V power
- PCI Express edge connector power

■ Temperature Range



- FPGA: 0°C ~100°C

1.4 Operating Temperature - Important

Please read the following instructions carefully to prevent damage to your DE5a-NET board.

The operating temperature range of Arria 10 GX device on DE5a-NET-DDR4 is 0°C ~100°C. When the FPGA temperature stays over 100°C for a long time, the FPGA could be damaged. It is therefore strongly recommended to use this board in an environment with sufficient airflow to dissipate the heat generated. It is also recommended to monitor the FPGA temperature continuously by adding Terasic IP introduced in **chapter 5.1** in the project. When the FPGA temperature is getting close to 100°C, please turn off the board immediately to reduce the FPGA temperature and protect the FPGA device.

Please refer to the directory /Demonstrations/Board_Protection in the DE5a-NET System CD for details about the Verilog based temperature monitor design example.

Board Components

This chapter introduces all the important components on the DE5a-Net.

2.1 Board Overview

Figure 2-1 is the top and bottom view of the DE5a-Net development board. It depicts the layout of the board and indicates the location of the connectors and key components. Users can refer to this figure for relative location of the connectors and key components.

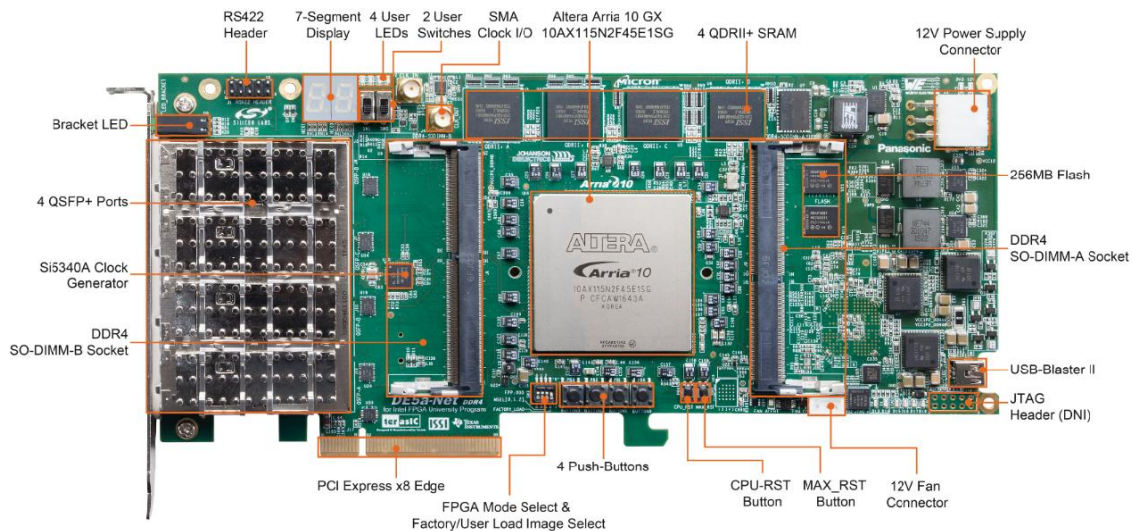


Figure 2-1 FPGA Board (Top)

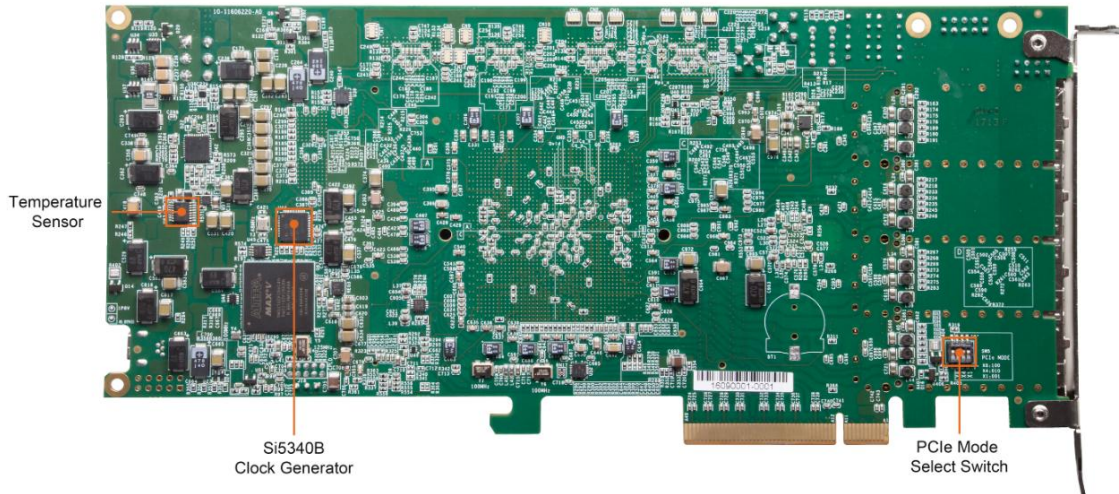


Figure 2-2 FPGA Board (Bottom)

2.2 Configuration, Status and Setup

■ Configure

The FPGA board supports two configuration methods for the Arria 10 FPGA:

- Configure the FPGA using the on-board USB-Blaster II.
- Flash memory configuration of the FPGA using stored images from the flash memory on power-up.

For programming by on-board USB-Blaster II, the following procedures show how to download a configuration bit stream into the Arria 10 GX FPGA:

- Make sure that power is provided to the FPGA board
- Connect your PC to the FPGA board using a mini-USB cable and make sure the USB-Blaster II driver is installed on PC.
- Launch Quartus Prime programmer and make sure the USB-Blaster II is detected.
- In Quartus Prime Programmer, add the configuration bit stream file (.sof), check

the associated “Program/Configure” item, and click “Start” to start FPGA programming.

■ Status LED

The FPGA Board development board includes board-specific status LEDs to indicate board status. Please refer to **Table 2-1** for the description of the LED indicator.

Table 2-1 Status LED

| Board Reference | LED Name | Description |
|-----------------|-------------|--|
| D6 | 12-V Power | Illuminates when 12-V power is active. |
| D5 | 3.3-V Power | Illuminates when 3.3-V power is active. |
| D16 | CONF DONE | Illuminates when the FPGA is successfully configured. Driven by the MAX II CPLD 5M2210 System Controller. |
| D15 | Loading | Illuminates when the MAX II CPLD 5M2210 System Controller is actively configuring the FPGA. Driven by the MAX II CPLD 5M2210 System Controller with the Embedded Blaster CPLD. |
| D17 | Error | Illuminates when the MAX II CPLD 5M2210 System Controller fails to configure the FPGA. Driven by the MAX II CPLD 5M2210 System Controller. |
| D19 | PAGE | Illuminates when FPGA is configured by the factory configuration bit stream. |

■ Setup PCI Express Control DIP switch

The PCI Express Control DIP switch (SW5) is provided to enable or disable different configurations of the PCIe Connector. **Table 2-2** lists the switch controls and description.

Table 2-2 SW5 PCIe Control DIP Switch

| Board Reference | Signal Name | Description | Default |
|-----------------|------------------|---|---------|
| SW5.1 | PCIE_PRSENT2n_x1 | On : Enable x1 presence detect Off: Disable x1 presence detect | Off |
| SW5.2 | PCIE_PRSENT2n_x4 | On : Enable x4 presence detect Off: Disable x4 presence detect | Off |
| SW5.3 | PCIE_PRSENT2n_x8 | On : Enable x8 presence detect Off: Disable x8 presence detect | On |

■ **Setup Configure Mode**

The position 1~3 of DIP switch SW3 are used to specify the configuration mode of the FPGA. As currently only one mode is supported, please set all positions as shown in **Figure 2-3**.

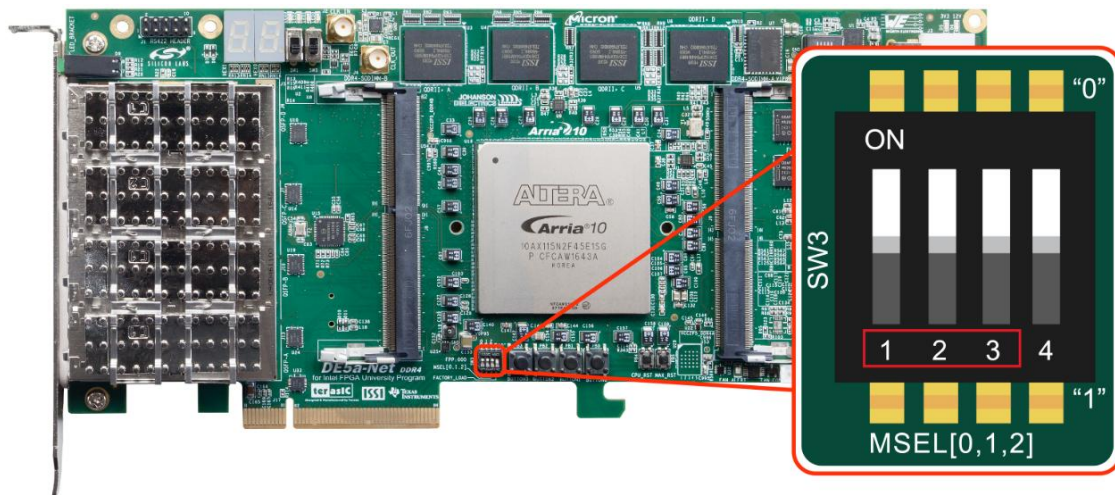


Figure 2-3 Position of DIP switch SW3 for Configure Mode

■ **Select Flash Image for Configuration**

The position 4 of DIP switch SW3 is used to specify the image for configuration of the FPGA. Setting Position 4 of SW3 to “1” (down position) specifies the default factory image to be loaded, as shown in **Figure 2-4**. Setting Position 4 of SW3 to “0” (up position) specifies the DE5a-Net to load a user-defined image, as shown in **Figure 2-5**.

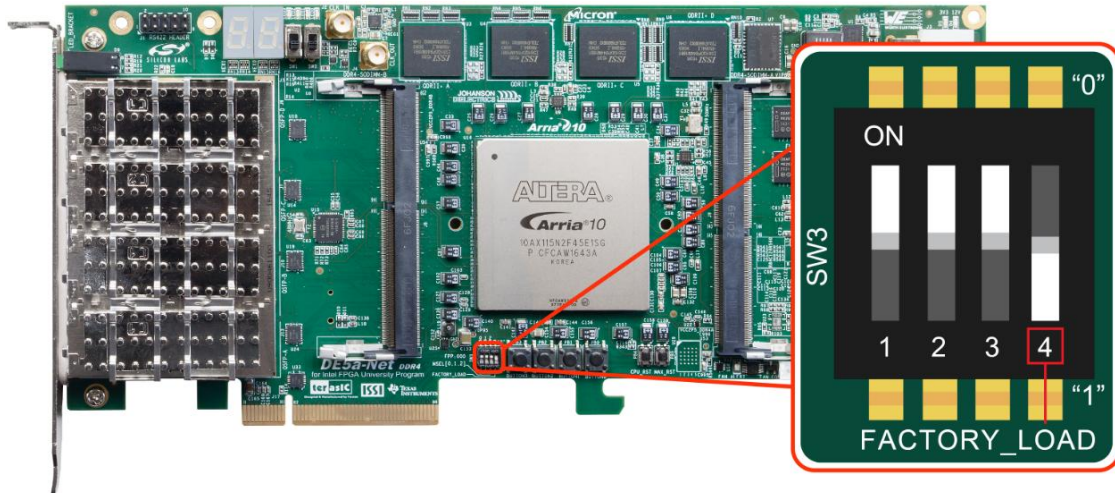


Figure 2-4 Position of DIP switch SW3 for Image Select – Factory Image Load

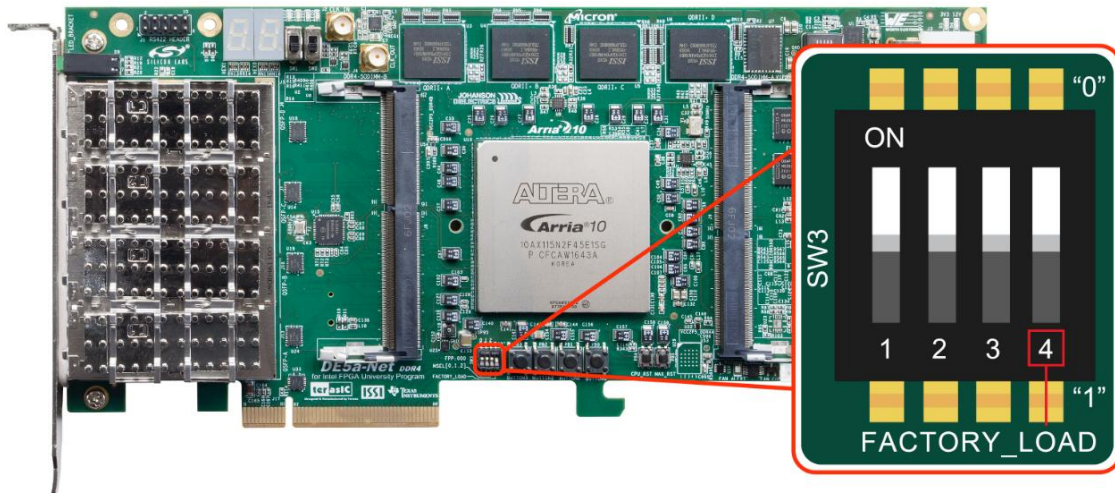


Figure 2-5 Position of DIP switch SW3 for Image Select – User Image Load

2.3 General User Input/Output

This section describes the user I/O interface to the FPGA.

■ User Defined Push-buttons

The FPGA board includes four user defined push-buttons that allow users to interact with the Arria 10 GX device. Each push-button provides a high logic level or a low logic level when it is not pressed or pressed, respectively. **Table 2-3** lists the board references, signal names and their corresponding Arria 10 GX device pin numbers.

Table 2-3 Push-button Pin Assignments, Schematic Signal Names, and Functions

| Board Reference | Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------|-----------------------|---|--------------|------------------------|
| PB0 | BUTTON0 | High Logic Level when the button is not pressed | 1.2-V | PIN_AJ13 |
| PB1 | BUTTON1 | | 1.2-V | PIN_AE13 |
| PB2 | BUTTON2 | | 1.2-V | PIN_AV16 |
| PB3 | BUTTON3 | | 1.2-V | PIN_AR9 |

■ User-Defined Slide Switch

There are two slide switches on the FPGA board to provide additional FPGA input control. When a slide switch is in the DOWN position or the UPPER position, it provides a low logic level or a high logic level to the Arria 10 GX FPGA, respectively, as shown in **Figure 2-6**.

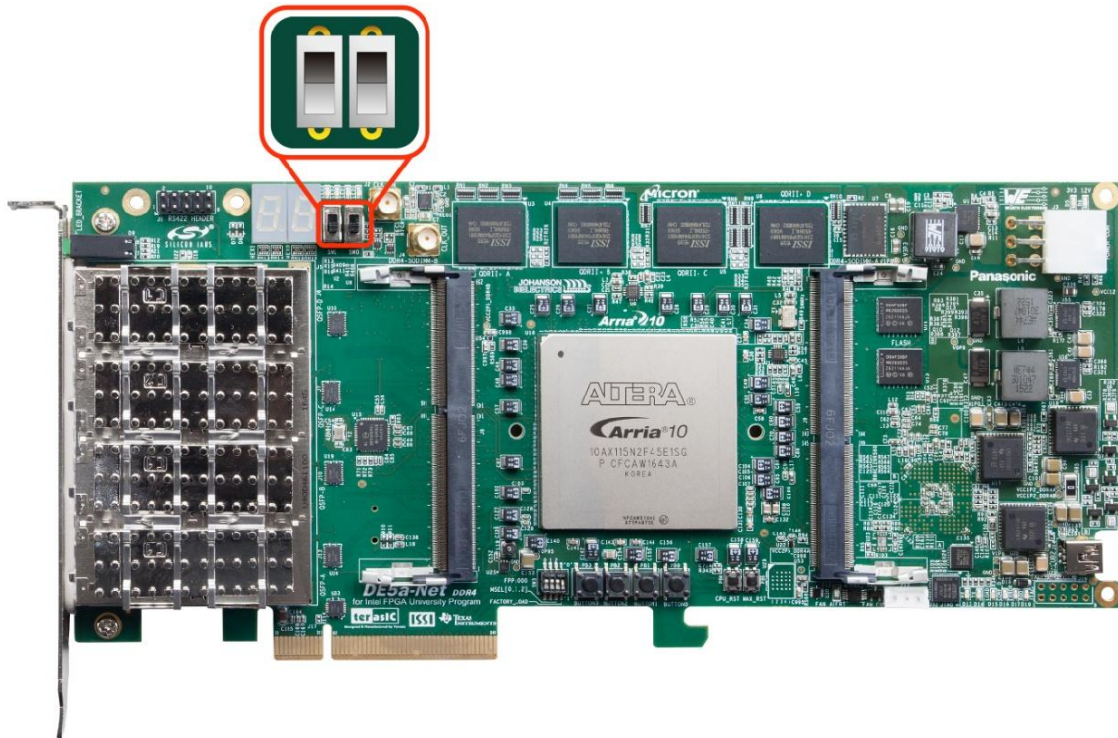


Figure 2-6 2 Slide switches

Table 2-4 lists the signal names and their corresponding Arria 10 GX device pin numbers.

Table 2-4 Slide Switch Pin Assignments, Schematic Signal Names, and Functions

| Board Reference | Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------|-----------------------|---|--------------|------------------------|
| SW0 | SW0 | High logic level when SW in the UPPER position. | 1.2-V | PIN_AY28 |
| SW1 | SW1 | | 1.2-V | PIN_AM27 |

■ User-Defined LEDs

The FPGA board consists of 8 user-controllable LEDs to allow status and debugging signals to be driven to the LEDs from the designs loaded into the Arria 10 GX device. Each LED is driven directly by the Arria 10 GX FPGA. The LED is turned on or off when the associated pins are driven to a low or high logic level, respectively. A list of the pin

names on the FPGA that are connected to the LEDs is given in **Table 2-5**.

Table 2-5 User LEDs Pin Assignments, Schematic Signal Names, and Functions

| Board Reference | Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------|-----------------------|---|--------------|------------------------|
| D4 | LED0 | Driving a logic 0 on the I/O port turns the LED ON. Driving a logic 1 on the I/O port turns the LED OFF. | 1.8-V | PIN_T11 |
| D3 | LED1 | | 1.8-V | PIN_R11 |
| D2 | LED2 | | 1.8-V | PIN_N15 |
| D1 | LED3 | | 1.8-V | PIN_M15 |
| D9-1 | LED_BRACKET0 | | 1.8-V | PIN_AF10 |
| D9-3 | LED_BRACKET1 | | 1.8-V | PIN_AF9 |
| D9-5 | LED_BRACKET2 | | 1.8-V | PIN_Y13 |
| D9-7 | LED_BRACKET3 | | 1.8-V | PIN_W11 |

■ 7-Segment Displays

The FPGA board has two 7-segment displays. As indicated in the schematic in **Figure 2-7**, the seven segments are connected to pins of the Arria 10 GX FPGA. Applying a low or high logic level to a segment will turn it on or turn it off, respectively.

Each segment in a display is identified by an index listed from 0 to 6 with the positions given in **Figure 2-8**. In addition, the decimal point is identified as DP. **Table 2-6** shows the mapping of the FPGA pin assignments to the 7-segment displays.

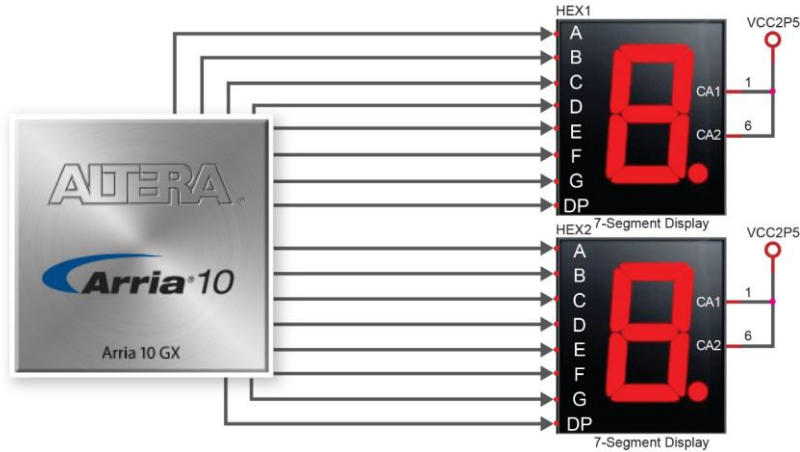


Figure 2-7 Connection between 7-segment displays and Arria 10 GX FPGA



Figure 2-8 Position and index of each segment in a 7-segment display

Table 2-6 User LEDs Pin Assignments, Schematic Signal Names, and Functions

| Board Reference | Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------|-----------------------|--|--------------|------------------------|
| HEX1 | HEX1_D0 | User-Defined 7-Segment Display. Driving logic 0 on the I/O port turns the 7-segment signal ON. Driving logic 1 on the I/O port turns the 7-segment signal OFF. | 1.2-V | PIN_AM32 |
| HEX1 | HEX1_D1 | | 1.2-V | PIN_AN32 |
| HEX1 | HEX1_D2 | | 1.2-V | PIN_AN31 |
| HEX1 | HEX1_D3 | | 1.2-V | PIN_AP31 |
| HEX1 | HEX1_D4 | | 1.2-V | PIN_BA35 |
| HEX1 | HEX1_D5 | | 1.2-V | PIN_BD34 |
| HEX1 | HEX1_D6 | | 1.2-V | PIN_AR31 |
| HEX1 | HEX1_DP | | 1.2-V | PIN_BC28 |
| HEX0 | HEX0_D0 | | 1.2-V | PIN_AW8 |
| HEX0 | HEX0_D1 | | 1.2-V | PIN_AY8 |

| | | | | |
|------|---------|--|-------|----------|
| HEX0 | HEX0_D2 | | 1.2-V | PIN_AY9 |
| HEX0 | HEX0_D3 | | 1.2-V | PIN_BA9 |
| HEX0 | HEX0_D4 | | 1.2-V | PIN_BB9 |
| HEX0 | HEX0_D5 | | 1.2-V | PIN_BD10 |
| HEX0 | HEX0_D6 | | 1.8-V | PIN_V10 |
| HEX0 | HEX0_DP | | 1.8-V | PIN_AG9 |

2.4 Temperature Sensor and Fan Control

The FPGA board is equipped with a temperature sensor, MAX1619, which provides temperature sensing and over-temperature alert. These functions are accomplished by connecting the temperature sensor to the internal temperature sensing diode of the Arria 10 GX device. The temperature status and alarm threshold registers of the temperature sensor can be programmed by a two-wire SMBus, which is connected to the Arria 10 GX FPGA. In addition, the 7-bit POR slave address for this sensor is set to '0011000b'. **Figure 2-9** shows the connection between the temperature sensor and the Arria 10 GX FPGA.

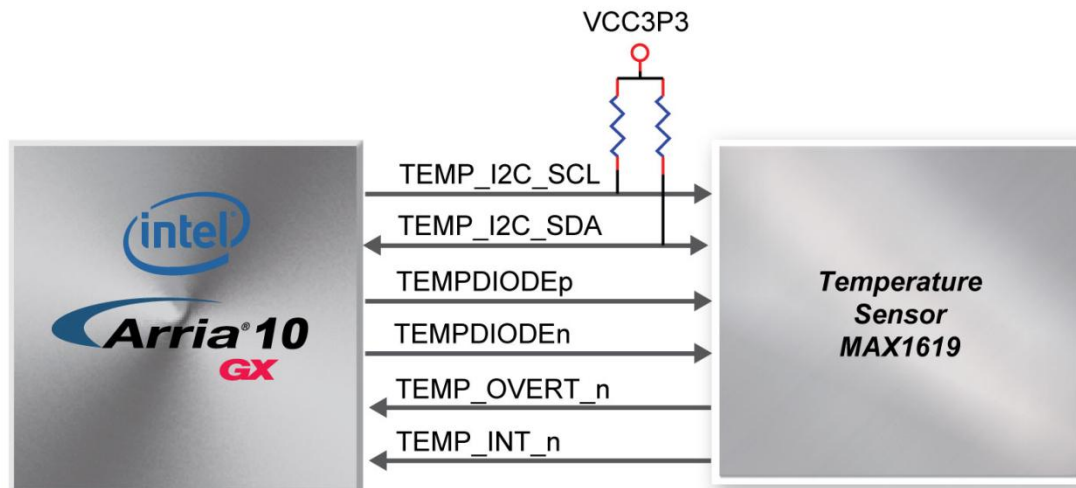


Figure 2-9 Connections between the temperature sensor and FPGA

An optional 3-pin +12V fan located on J15 of the FPGA board is intended to reduce the temperature of the FPGA. The board is equipped with a Fan-Speed regulator and monitor, MAX6650, through an I2C interface, Users regulate and monitor the speed of fan depending on the measured system temperature. **Figure2-10** shows the connection between the Fan-Speed Regulator and Monitor and the Arria 10 GX FPGA.

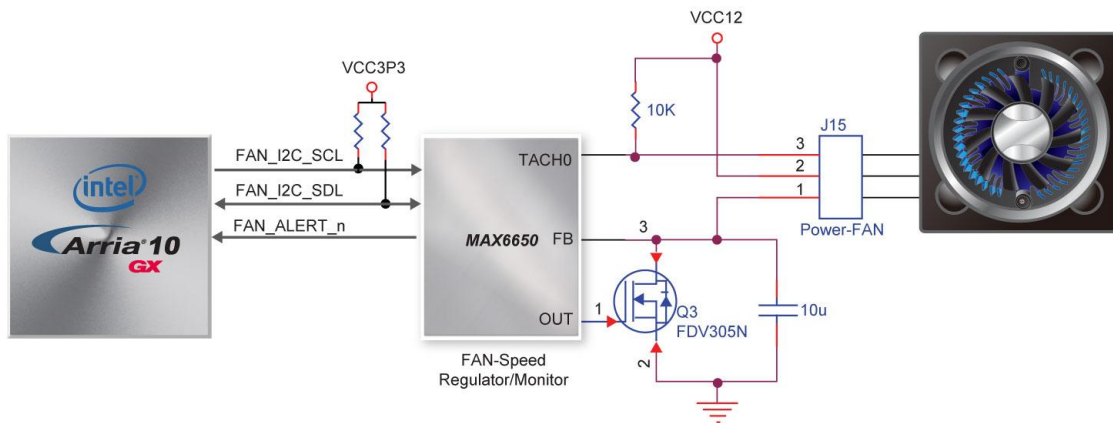


Figure 2-10 Connections between the Fan-Speed Regulator/ Monitor and the Arria 10 GX FPGA

The pin assignments for the associated interface are listed in **Table 2-7**.

Table 2-7 Temperature Sensor and Fan Speed Control Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|---|--------------|------------------------|
| TEMPDIODEp | Positive pin of temperature diode in Arria 10 | - | PIN_N21 |
| TEMPDIODEn | Negative pin of temperature diode in Arria 10 | - | PIN_P21 |
| TEMP_I2C_SCL | SMBus clock | 1.2-V | PIN_AW11 |

| | | | |
|--------------|-------------------------|-------|----------|
| TEMP_I2C_SDA | SMBus data | 1.2-V | PIN_AY12 |
| TEMP_OVERT_n | SMBus alert (interrupt) | 1.2-V | PIN_AT14 |
| TEMP_INT_n | SMBus alert (interrupt) | 1.2-V | PIN_AU12 |
| FAN_I2C_SCL | 2-Wire Serial Clock | 1.2-V | PIN_AJ33 |
| FAN_I2C_SDA | 2-Wire Serial-Data | 1.2-V | PIN_AL32 |
| FAN_ALERT_n | Active-low ALERT input | 1.2-V | PIN_AL31 |

2.5 Power Monitor

The DE5a-Net has implemented a power monitor chip to monitor the board input power voltage and current. **Figure 2-11** shows the connection between the power monitor chip and the Arria 10 GX FPGA. The power monitor chip monitors both shunt voltage drops and board input power voltage allows user to monitor the total board power consumption. Programmable calibration value, conversion times, and averaging, combined with an internal multiplier, enable direct readouts of current in amperes and power in watts. Table 2-8 shows the pin assignment of power monitor I2C bus.

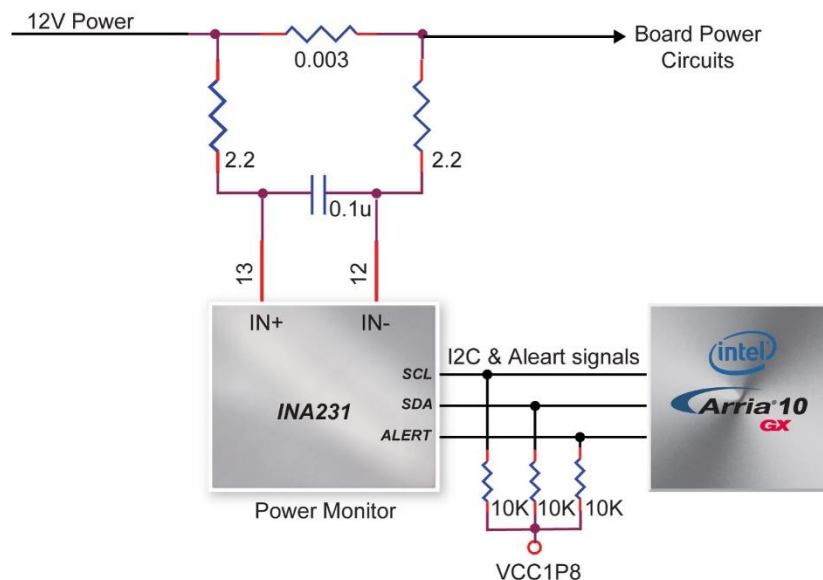


Figure 2-11 Connections between the Power Monitor and FPGA

Table 2-8 Pin Assignment of Power Monitor I2C bus

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|---------------------|--------------|------------------------|
| POWER_MONITOR_I2C_SCL | Power Monitor SCL | 1.8V | PIN_AT26 |
| POWER_MONITOR_I2C_SDA | Power Monitor SDA | 1.8V | PIN_AP25 |
| POWER_MONITOR_ALERT | Power Monitor ALERT | 1.8V | PIN_BD23 |

2.6 Clock Circuit

The development board includes one 50 MHz and two programmable clock generators. **Figure 2-12** shows the default frequencies of on-board all external clocks going to the Arria 10 GX FPGA.

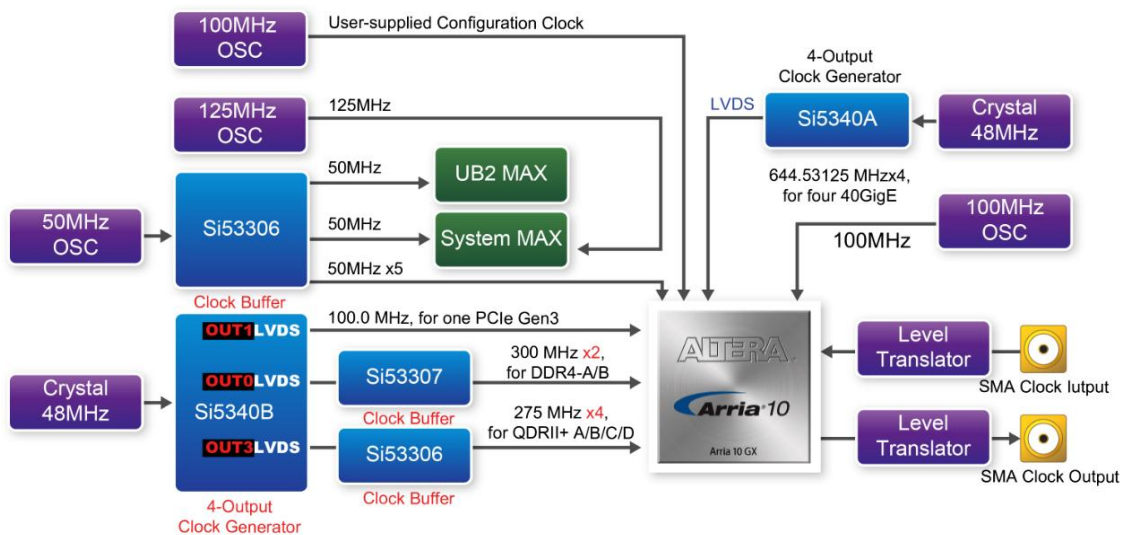


Figure 2-12 Clock circuit of the FPGA Board

A clock buffer is used to duplicate the 50 MHz oscillator, so there are five 50MHz clocks fed into different five FPGA banks. The two programming clock generators are low-jitter oscillators which are used to provide special and high-quality clock signals for high-speed transceivers and high bandwidth memory. Through I2C serial interface, the clock generator controllers in the Arria 10 GX FPGA can be used to program the Si5340A and Si5340B to generate 40G Ethernet QSFP+, PCIe and high bandwidth memory reference

clocks respectively. Two SMA connectors provide external clock input and clock output respectively.

Table 2-9 lists the clock source, signal names, default frequency and their corresponding Arria 10 GX device pin numbers.

Table 2-9 Clock Source, Signal Name, Default Frequency, Pin Assignments and Functions

| Source | Schematic Signal Name | Default Frequency | I/O Standard | Arria 10 GX Pin Number | Application |
|--------|-----------------------|-------------------|--------------|------------------------|---------------------------------|
| Y1 | CLK_50_B2J | 50.0 MHz | 1.8V | PIN_W36 | |
| | CLK_50_B2L | | 1.8V | PIN_H32 | |
| | CLK_50_B3D | | 1.8V | PIN_AN7 | |
| | CLK_50_B3F | | 1.8V | PIN_G12 | |
| | CLK_50_B3H | | 1.8V | PIN_D21 | |
| Y7 | CLK_100_B3D | 100.0MHz | 1.8V | PIN_AH11 | |
| J2 | SMA_CLKIN | User Defined | 1.8V | PIN_AC32 | External Clock Input |
| J4 | SMA_CLKOUT | User Defined | 1.8V | PIN_AA36 | Clock Output |
| U15 | QSFPA_REFCLK_p | 644.53125 MHz | LVDS | PIN_AH5 | 40G QSFP+ A port |
| | QSFPB_REFCLK_p | 644.53125 MHz | LVDS | PIN_AD5 | 40G QSFP+ B port |
| | QSFPD_REFCLK_p | 644.53125 MHz | LVDS | PIN_Y5 | 40G QSFP+ C port |
| | QSFPD_REFCLK_p | 644.53125 MHz | LVDS | PIN_T5 | 40G QSFP+ D port |
| U44 | DDR4A_REFCLK_p | 300 MHz | LVDS | PIN_AV33 | DDR4 reference clock for A port |
| | DDR4B_REFCLK_p | 300 MHz | LVDS | PIN_AP14 | DDR4 reference clock for B port |
| | QDRIIA_REFCLK_p | 275 MHz | LVDS | PIN_L9 | QDRII+ reference |

| | | | | | |
|--|------------------|---------|------|----------|-----------------------------------|
| | | | | | clock for A port |
| | QDRIIB_REFCLK_p | 275 MHz | LVDS | PIN_N18 | QDRII+ reference clock for B port |
| | QDRIIC_REFCLK_p | 275 MHz | LVDS | PIN_G24 | QDRII+ reference clock for C port |
| | QDRIID_REFCLK_p | 275 MHz | LVDS | PIN_M34 | QDRII+ reference clock for D port |
| | OB_PCIE_REFCLK_p | 100 MHz | LVDS | PIN_AK40 | PCIe reference clock |

Table 2-10 lists the programmable oscillator control pins, signal names, I/O standard and their corresponding Arria 10 GX device pin numbers.

Table 2-10 Programmable oscillator control pin, Signal Name, I/O standard, Pin Assignments and Descriptions

| Programmable Oscillator | Schematic Signal Name | I/O Standard | Arria 10 GX Pin Number | Description |
|-------------------------|-----------------------|--------------|------------------------|---------------------------------|
| Si5340A (U15) | Si5340A_I2C_SCL | 1.8-V | PIN_AF11 | I2C bus, connected with Si5340A |
| | Si5340A_I2C_SDA | 1.8-V | PIN_AE11 | |
| Si5340A (U15) | Si5340A_RST_n | 1.8-V | PIN_AN6 | Si5340A reset signal |
| | Si5340A_INTR | 1.8-V | PIN_AM6 | Si5340A interrupt signal |
| | Si5340A_OE_n | 1.8-V | PIN_AJ10 | Si5340A output enable signal |
| Si5340B (U44) | Si5340B_I2C_SCL | 1.8-V | PIN_G37 | I2C bus, connected with Si5340B |
| | Si5340B_I2C_SDA | 1.8-V | PIN_H31 | |
| | Si5340B_RST_n | 1.8-V | PIN_G38 | Si5340B reset signal |
| | Si5340B_INTR | 1.8-V | PIN_G32 | Si5340B interrupt signal |
| | Si5340B_OE_n | 1.8-V | PIN_BD24 | Si5340B output enable signal |

2.7 FLASH Memory

The development board has two 1Gb CFI-compatible synchronous flash devices for non-volatile storage of FPGA configuration data, user application data, and user code space.

Each interface has a 16-bit data bus and the two devices combined allow for FPP x32 configuration. This device is part of the shared flash and MAX (FM) bus, which connects to the flash memory and MAX V CPLD (5M2210) System Controller. **Figure 2-13** shows the connections between the Flash, MAX and Arria 10 GX FPGA.

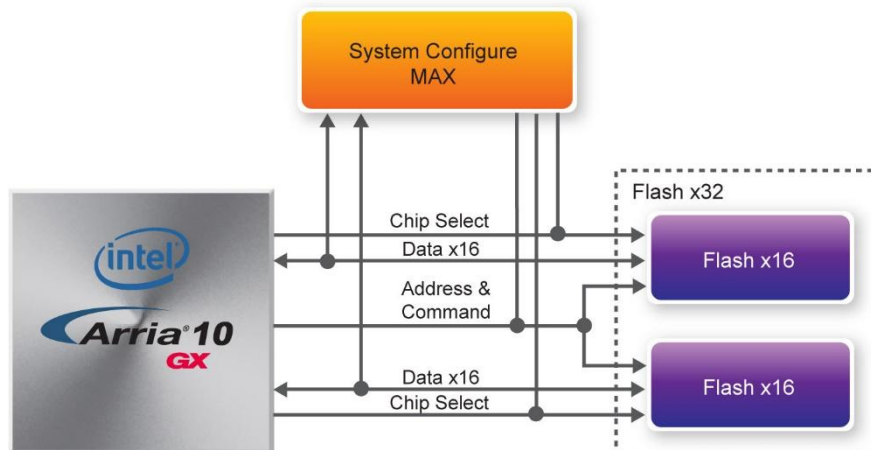


Figure 2-13 Connection between the Flash, Max and Arria 10 GX FPGA

Table 2-11 lists the flash pin assignments, signal names, and functions.

Table 2-11 Flash Memory Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------|--------------|------------------------|
| FLASH_A1 | Address bus | 1.8-V | PIN_H26 |
| FLASH_A2 | Address bus | 1.8-V | PIN_J18 |
| FLASH_A3 | Address bus | 1.8-V | PIN_N17 |
| FLASH_A4 | Address bus | 1.8-V | PIN_P15 |
| FLASH_A5 | Address bus | 1.8-V | PIN_B18 |
| FLASH_A6 | Address bus | 1.8-V | PIN_E18 |
| FLASH_A7 | Address bus | 1.8-V | PIN_D18 |
| FLASH_A8 | Address bus | 1.8-V | PIN_J10 |
| FLASH_A9 | Address bus | 1.8-V | PIN_B17 |
| FLASH_A10 | Address bus | 1.8-V | PIN_J11 |
| FLASH_A11 | Address bus | 1.8-V | PIN_H8 |
| FLASH_A12 | Address bus | 1.8-V | PIN_A17 |
| FLASH_A13 | Address bus | 1.8-V | PIN_G8 |
| FLASH_A14 | Address bus | 1.8-V | PIN_G9 |
| FLASH_A15 | Address bus | 1.8-V | PIN_A16 |
| FLASH_A16 | Address bus | 1.8-V | PIN_K11 |
| FLASH_A17 | Address bus | 1.8-V | PIN_B15 |
| FLASH_A18 | Address bus | 1.8-V | PIN_G7 |
| FLASH_A19 | Address bus | 1.8-V | PIN_F6 |
| FLASH_A20 | Address bus | 1.8-V | PIN_A15 |
| FLASH_A21 | Address bus | 1.8-V | PIN_A14 |
| FLASH_A22 | Address bus | 1.8-V | PIN_H6 |
| FLASH_A23 | Address bus | 1.8-V | PIN_T12 |
| FLASH_A24 | Address bus | 1.8-V | PIN_U12 |
| FLASH_A25 | Address bus | 1.8-V | PIN_F7 |
| FLASH_A26 | Address bus | 1.8-V | PIN_B14 |
| FLASH_D0 | Data bus | 1.8-V | PIN_B35 |
| FLASH_D1 | Data bus | 1.8-V | PIN_A35 |
| FLASH_D2 | Data bus | 1.8-V | PIN_C35 |
| FLASH_D3 | Data bus | 1.8-V | PIN_C33 |
| FLASH_D4 | Data bus | 1.8-V | PIN_C32 |
| FLASH_D5 | Data bus | 1.8-V | PIN_A32 |

| | | | |
|---------------|------------------------------|-------|----------|
| FLASH_D6 | Data bus | 1.8-V | PIN_C26 |
| FLASH_D7 | Data bus | 1.8-V | PIN_B24 |
| FLASH_D8 | Data bus | 1.8-V | PIN_C36 |
| FLASH_D9 | Data bus | 1.8-V | PIN_B34 |
| FLASH_D10 | Data bus | 1.8-V | PIN_A34 |
| FLASH_D11 | Data bus | 1.8-V | PIN_B33 |
| FLASH_D12 | Data bus | 1.8-V | PIN_B32 |
| FLASH_D13 | Data bus | 1.8-V | PIN_A31 |
| FLASH_D14 | Data bus | 1.8-V | PIN_E24 |
| FLASH_D15 | Data bus | 1.8-V | PIN_C25 |
| FLASH_D16 | Data bus | 1.8-V | PIN_K33 |
| FLASH_D17 | Data bus | 1.8-V | PIN_J39 |
| FLASH_D18 | Data bus | 1.8-V | PIN_AA32 |
| FLASH_D19 | Data bus | 1.8-V | PIN_J35 |
| FLASH_D20 | Data bus | 1.8-V | PIN_H36 |
| FLASH_D21 | Data bus | 1.8-V | PIN_AB32 |
| FLASH_D22 | Data bus | 1.8-V | PIN_J34 |
| FLASH_D23 | Data bus | 1.8-V | PIN_AA31 |
| FLASH_D24 | Data bus | 1.8-V | PIN_J36 |
| FLASH_D25 | Data bus | 1.8-V | PIN_J38 |
| FLASH_D26 | Data bus | 1.8-V | PIN_K34 |
| FLASH_D27 | Data bus | 1.8-V | PIN_H38 |
| FLASH_D28 | Data bus | 1.8-V | PIN_H37 |
| FLASH_D29 | Data bus | 1.8-V | PIN_Y31 |
| FLASH_D30 | Data bus | 1.8-V | PIN_H35 |
| FLASH_D31 | Data bus | 1.8-V | PIN_J33 |
| FLASH_CLK | Clock | 1.8-V | PIN_T9 |
| FLASH_RESET_n | Reset | 1.8-V | PIN_C17 |
| FLASH_CE_n[0] | Chip enable of offlash-0 | 1.8-V | PIN_H10 |
| FLASH_CE_n[1] | Chip enable of of flash-1 | 1.8-V | PIN_N16 |
| FLASH_OE_n | Output enable | 1.8-V | PIN_C16 |
| FLASH_WE_n | Write enable | 1.8-V | PIN_U10 |

| | | | |
|--------------------|------------------|-------|---------|
| FLASH_ADV_n | Address valid | 1.8-V | PIN_H7 |
| FLASH_RDY_BSY_n[0] | Ready of flash-0 | 1.8-V | PIN_J8 |
| FLASH_RDY_BSY_n[1] | Ready of flash-1 | 1.8-V | PIN_L36 |

2.8 DDR4 SO-DIMM

The development board supports two independent banks of DDR4 SDRAM SO-DIMM. Each DDR4 SODIMM socket is wired to support a maximum capacity of 16GB with a 64-bit data bus. Using differential DQS signaling for the DDR4 SDRAM interfaces, it is capable of running at up to 1200MHz memory clock for a maximum theoretical bandwidth up to 153.6Gbps. **Figure 2-14** shows the connections between the DDR4 SDRAM SO-DIMMs and Arria 10 GX FPGA.

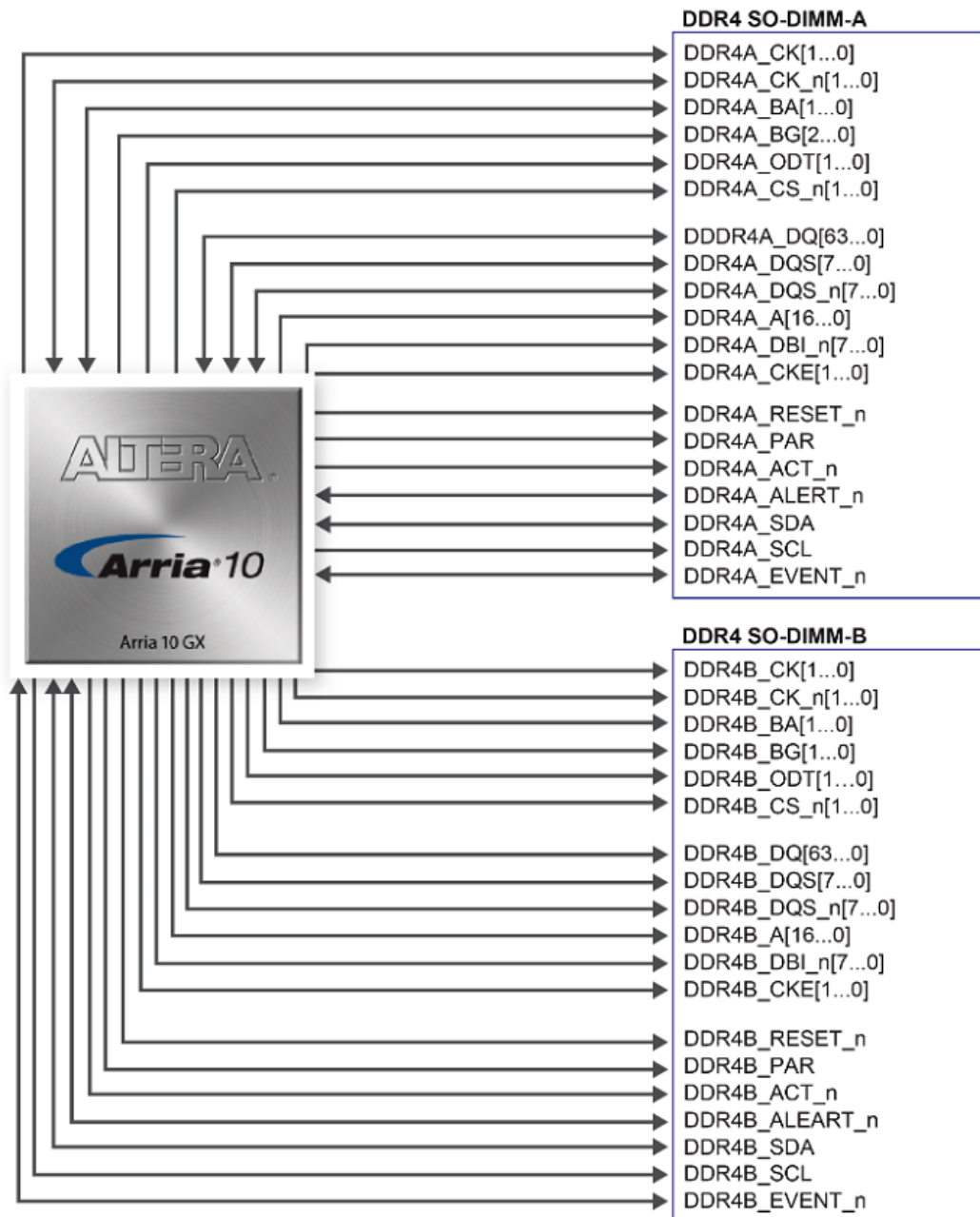


Figure 2-14 Connection between the DDR4 and Arria 10 GX FPGA

The pin assignments for DDR4 SDRAM SO-DIMM Bank-A and Bank-B are listed in [Table 2-12](#) and [Table 2-13](#), in respectively.

Table 2-12 DDR4-A Bank Pin Assignments, Schematic Signal Names, and

Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------|--------------|------------------------|
| DDR4A_DQ0 | Data [0] | 1.2-V POD | PIN_AH35 |
| DDR4A_DQ1 | Data [1] | 1.2-V POD | PIN_AH32 |
| DDR4A_DQ2 | Data [2] | 1.2-V POD | PIN_AH36 |
| DDR4A_DQ3 | Data [3] | 1.2-V POD | PIN_AG35 |
| DDR4A_DQ4 | Data [4] | 1.2-V POD | PIN_AJ34 |
| DDR4A_DQ5 | Data [5] | 1.2-V POD | PIN_AH33 |
| DDR4A_DQ6 | Data [6] | 1.2-V POD | PIN_AJ36 |
| DDR4A_DQ7 | Data [7] | 1.2-V POD | PIN_AJ35 |
| DDR4A_DQ8 | Data [8] | 1.2-V POD | PIN_AK36 |
| DDR4A_DQ9 | Data [9] | 1.2-V POD | PIN_AN35 |
| DDR4A_DQ10 | Data [10] | 1.2-V POD | PIN_AM35 |
| DDR4A_DQ11 | Data [11] | 1.2-V POD | PIN_AM37 |
| DDR4A_DQ12 | Data [12] | 1.2-V POD | PIN_AP34 |
| DDR4A_DQ13 | Data [13] | 1.2-V POD | PIN_AT34 |
| DDR4A_DQ14 | Data [14] | 1.2-V POD | PIN_AL36 |
| DDR4A_DQ15 | Data [15] | 1.2-V POD | PIN_AL35 |
| DDR4A_DQ16 | Data [16] | 1.2-V POD | PIN_AT39 |
| DDR4A_DQ17 | Data [17] | 1.2-V POD | PIN_AV35 |
| DDR4A_DQ18 | Data [18] | 1.2-V POD | PIN_AV37 |
| DDR4A_DQ19 | Data [19] | 1.2-V POD | PIN_AT36 |
| DDR4A_DQ20 | Data [20] | 1.2-V POD | PIN_AU39 |
| DDR4A_DQ21 | Data [21] | 1.2-V POD | PIN_AU35 |
| DDR4A_DQ22 | Data [22] | 1.2-V POD | PIN_AU37 |
| DDR4A_DQ23 | Data [23] | 1.2-V POD | PIN_AV36 |
| DDR4A_DQ24 | Data [24] | 1.2-V POD | PIN_AR36 |
| DDR4A_DQ25 | Data [25] | 1.2-V POD | PIN_AN36 |
| DDR4A_DQ26 | Data [26] | 1.2-V POD | PIN_AM39 |
| DDR4A_DQ27 | Data [27] | 1.2-V POD | PIN_AR39 |
| DDR4A_DQ28 | Data [28] | 1.2-V POD | PIN_AN38 |
| DDR4A_DQ29 | Data [29] | 1.2-V POD | PIN_AN37 |

| | | | |
|------------|-----------|-----------|----------|
| DDR4A_DQ30 | Data [30] | 1.2-V POD | PIN_AM38 |
| DDR4A_DQ31 | Data [31] | 1.2-V POD | PIN_AR37 |
| DDR4A_DQ32 | Data [32] | 1.2-V POD | PIN_AM30 |
| DDR4A_DQ33 | Data [33] | 1.2-V POD | PIN_AN28 |
| DDR4A_DQ34 | Data [34] | 1.2-V POD | PIN_AN27 |
| DDR4A_DQ35 | Data [35] | 1.2-V POD | PIN_AM28 |
| DDR4A_DQ36 | Data [36] | 1.2-V POD | PIN_AN30 |
| DDR4A_DQ37 | Data [37] | 1.2-V POD | PIN_AM29 |
| DDR4A_DQ38 | Data [38] | 1.2-V POD | PIN_AP28 |
| DDR4A_DQ39 | Data [39] | 1.2-V POD | PIN_AL30 |
| DDR4A_DQ40 | Data [40] | 1.2-V POD | PIN_BB32 |
| DDR4A_DQ41 | Data [41] | 1.2-V POD | PIN_BC31 |
| DDR4A_DQ42 | Data [42] | 1.2-V POD | PIN_AW29 |
| DDR4A_DQ43 | Data [43] | 1.2-V POD | PIN_AW28 |
| DDR4A_DQ44 | Data [44] | 1.2-V POD | PIN_BA32 |
| DDR4A_DQ45 | Data [45] | 1.2-V POD | PIN_AY31 |
| DDR4A_DQ46 | Data [46] | 1.2-V POD | PIN_BB30 |
| DDR4A_DQ47 | Data [47] | 1.2-V POD | PIN_AY29 |
| DDR4A_DQ48 | Data [48] | 1.2-V POD | PIN_BB28 |
| DDR4A_DQ49 | Data [49] | 1.2-V POD | PIN_BD30 |
| DDR4A_DQ50 | Data [50] | 1.2-V POD | PIN_BB27 |
| DDR4A_DQ51 | Data [51] | 1.2-V POD | PIN_BD28 |
| DDR4A_DQ52 | Data [52] | 1.2-V POD | PIN_BC30 |
| DDR4A_DQ53 | Data [53] | 1.2-V POD | PIN_BD29 |
| DDR4A_DQ54 | Data [54] | 1.2-V POD | PIN_BC27 |
| DDR4A_DQ55 | Data [55] | 1.2-V POD | PIN_BB29 |
| DDR4A_DQ56 | Data [56] | 1.2-V POD | PIN_AT27 |
| DDR4A_DQ57 | Data [57] | 1.2-V POD | PIN_AV28 |
| DDR4A_DQ58 | Data [58] | 1.2-V POD | PIN_AR28 |
| DDR4A_DQ59 | Data [59] | 1.2-V POD | PIN_AT30 |
| DDR4A_DQ60 | Data [60] | 1.2-V POD | PIN_AU29 |
| DDR4A_DQ61 | Data [61] | 1.2-V POD | PIN_AU27 |
| DDR4A_DQ62 | Data [62] | 1.2-V POD | PIN_AT29 |
| DDR4A_DQ63 | Data [63] | 1.2-V POD | PIN_AR27 |

| | | | |
|--------------|---------------------------|---------------------------|----------|
| DDR4A_DQS0 | Data Strobe p[0] | DIFFERENTIAL 1.2-V POD | PIN_AK37 |
| DDR4A_DQS_n0 | Data Strobe n[0] | DIFFERENTIAL 1.2-V POD | PIN_AL37 |
| DDR4A_DQS1 | Data Strobe p[1] | DIFFERENTIAL 1.2-V POD | PIN_AL34 |
| DDR4A_DQS_n1 | Data Strobe n[1] | DIFFERENTIAL 1.2-V POD | PIN_AM34 |
| DDR4A_DQS2 | Data Strobe p[2] | DIFFERENTIAL 1.2-V POD | PIN_AU38 |
| DDR4A_DQS_n2 | Data Strobe n[2] | DIFFERENTIAL 1.2-V POD | PIN_AV38 |
| DDR4A_DQS3 | Data Strobe p[3] | DIFFERENTIAL 1.2-V POD | PIN_AP38 |
| DDR4A_DQS_n3 | Data Strobe n[3] | DIFFERENTIAL 1.2-V POD | PIN_AR38 |
| DDR4A_DQS4 | Data Strobe p[4] | DIFFERENTIAL 1.2-V POD | PIN_AP29 |
| DDR4A_DQS_n4 | Data Strobe n[4] | DIFFERENTIAL 1.2-V POD | PIN_AR29 |
| DDR4A_DQS5 | Data Strobe p[5] | DIFFERENTIAL 1.2-V POD | PIN_BA29 |
| DDR4A_DQS_n5 | Data Strobe n[5] | DIFFERENTIAL 1.2-V POD | PIN_BA30 |
| DDR4A_DQS6 | Data Strobe p[6] | DIFFERENTIAL 1.2-V POD | PIN_BA26 |
| DDR4A_DQS_n6 | Data Strobe n[6] | DIFFERENTIAL 1.2-V POD | PIN_BA27 |
| DDR4A_DQS7 | Data Strobe p[7] | DIFFERENTIAL 1.2-V POD | PIN_AV30 |
| DDR4A_DQS_n7 | Data Strobe n[7] | DIFFERENTIAL 1.2-V POD | PIN_AW30 |
| DDR4A_DBI_n0 | Data Bus Inversion [0] | 1.2-V POD | PIN_AK34 |
| DDR4A_DBI_n1 | Data Bus Inversion | 1.2-V POD | PIN_AP35 |

| | | | |
|--------------|---------------------------|-----------|----------|
| | [1] | | |
| DDR4A_DBI_n2 | Data Bus Inversion [2] | 1.2-V POD | PIN_AT37 |
| DDR4A_DBI_n3 | Data Bus Inversion [3] | 1.2-V POD | PIN_AP36 |
| DDR4A_DBI_n4 | Data Bus Inversion [4] | 1.2-V POD | PIN_AP30 |
| DDR4A_DBI_n5 | Data Bus Inversion [5] | 1.2-V POD | PIN_BA31 |
| DDR4A_DBI_n6 | Data Bus Inversion [6] | 1.2-V POD | PIN_BD31 |
| DDR4A_DBI_n7 | Data Bus Inversion [7] | 1.2-V POD | PIN_AU30 |
| DDR4A_A0 | Address [0] | SSTL-12 | PIN_AW34 |
| DDR4A_A1 | Address [1] | SSTL-12 | PIN_AY34 |
| DDR4A_A2 | Address [2] | SSTL-12 | PIN_AV31 |
| DDR4A_A3 | Address [3] | SSTL-12 | PIN_AW31 |
| DDR4A_A4 | Address [4] | SSTL-12 | PIN_BA37 |
| DDR4A_A5 | Address [5] | SSTL-12 | PIN_BB37 |
| DDR4A_A6 | Address [6] | SSTL-12 | PIN_AY36 |
| DDR4A_A7 | Address [7] | SSTL-12 | PIN_AY37 |
| DDR4A_A8 | Address [8] | SSTL-12 | PIN_AY32 |
| DDR4A_A9 | Address [9] | SSTL-12 | PIN_AY33 |
| DDR4A_A10 | Address [10] | SSTL-12 | PIN_AW35 |
| DDR4A_A11 | Address [11] | SSTL-12 | PIN_AW36 |
| DDR4A_A12 | Address [12] | SSTL-12 | PIN_AU34 |
| DDR4A_A13 | Address [13] | SSTL-12 | PIN_AT31 |
| DDR4A_A14 | Address [14]/ WE_n | SSTL-12 | PIN_AT32 |
| DDR4A_A15 | Address [15]/ CAS_n | SSTL-12 | PIN_AU32 |
| DDR4A_A16 | Address [16]/ RAS_n | SSTL-12 | PIN_AV32 |
| DDR4A_BA0 | Bank Select [0] | SSTL-12 | PIN_AR32 |

| | | | |
|---------------|----------------------------------|-------------------------|----------|
| DDR4A_BA1 | Bank Select [1] | SSTL-12 | PIN_AP33 |
| DDR4A_BG0 | Bank Group Select [0] | SSTL-12 | PIN_AR33 |
| DDR4A_BG1 | Bank Group Select [1] | SSTL-12 | PIN_BC35 |
| DDR4A_CK0 | Clock p0 | DIFFERENTIAL 1.2-V SSTL | PIN_BA34 |
| DDR4A_CK_n0 | Clock n0 | DIFFERENTIAL 1.2-V SSTL | PIN_BB35 |
| DDR4A_CK1 | Clock p1 | DIFFERENTIAL 1.2-V SSTL | PIN_AM33 |
| DDR4A_CK_n1 | Clock n1 | DIFFERENTIAL 1.2-V SSTL | PIN_AN33 |
| DDR4A_CKE0 | Clock Enable pin 0 | SSTL-12 | PIN_BD33 |
| DDR4A_CKE1 | Clock Enable pin 1 | SSTL-12 | PIN_AK31 |
| DDR4A_ODT0 | On Die Termination[0] | SSTL-12 | PIN_BC32 |
| DDR4A_ODT1 | On Die Termination[1] | SSTL-12 | PIN_AK32 |
| DDR4A_CS_n0 | Chip Select [0] | SSTL-12 | PIN_BB33 |
| DDR4A_CS_n1 | Chip Select [1] | SSTL-12 | PIN_AK33 |
| DDR4A_PAR | Command and Address Parity Input | SSTL-12 | PIN_BA36 |
| DDR4A_ALERT_n | Register ALERT_n output | SSTL-12 | PIN_AG34 |
| DDR4A_ACT_n | Activation Command Input | SSTL-12 | PIN_BB34 |
| DDR4A_RESET_n | Chip Reset | 1.2 V | PIN_BD35 |
| DDR4A_EVENT_n | Chip Temperature Event | 1.2 V | PIN_AR34 |
| DDR4A_SDA | Chip I2C Serial DATA | 1.2 V | PIN_BC33 |
| DDR4A_SCL | Chip I2C Serial Clock | 1.2 V | PIN_AT35 |

| | | | |
|----------------|----------------------------------|------|----------|
| DDR4A_REFCLK_p | DDR4 A port Reference Clock p | LVDS | PIN_AV33 |
| DDR4A_REFCLK_n | DDR4 A port Reference Clock n | LVDS | PIN_AW33 |

Table 2-13 DDR4-B Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10GX Pin Number |
|-----------------------|-------------|--------------|-----------------------|
| DDR4B_DQ0 | Data [0] | 1.2-V POD | PIN_AR19 |
| DDR4B_DQ1 | Data [1] | 1.2-V POD | PIN_AU19 |
| DDR4B_DQ2 | Data [2] | 1.2-V POD | PIN_AM17 |
| DDR4B_DQ3 | Data [3] | 1.2-V POD | PIN_AM18 |
| DDR4B_DQ4 | Data [4] | 1.2-V POD | PIN_AM20 |
| DDR4B_DQ5 | Data [5] | 1.2-V POD | PIN_AR18 |
| DDR4B_DQ6 | Data [6] | 1.2-V POD | PIN_AM19 |
| DDR4B_DQ7 | Data [7] | 1.2-V POD | PIN_AP19 |
| DDR4B_DQ8 | Data [8] | 1.2-V POD | PIN_BA17 |
| DDR4B_DQ9 | Data [9] | 1.2-V POD | PIN_AY17 |
| DDR4B_DQ10 | Data [10] | 1.2-V POD | PIN_AY16 |
| DDR4B_DQ11 | Data [11] | 1.2-V POD | PIN_AU15 |
| DDR4B_DQ12 | Data [12] | 1.2-V POD | PIN_AW18 |
| DDR4B_DQ13 | Data [13] | 1.2-V POD | PIN_AV17 |
| DDR4B_DQ14 | Data [14] | 1.2-V POD | PIN_AV15 |
| DDR4B_DQ15 | Data [15] | 1.2-V POD | PIN_AW16 |
| DDR4B_DQ16 | Data [16] | 1.2-V POD | PIN_BB17 |
| DDR4B_DQ17 | Data [17] | 1.2-V POD | PIN_BD16 |
| DDR4B_DQ18 | Data [18] | 1.2-V POD | PIN_BC16 |
| DDR4B_DQ19 | Data [19] | 1.2-V POD | PIN_BB18 |
| DDR4B_DQ20 | Data [20] | 1.2-V POD | PIN_BD18 |
| DDR4B_DQ21 | Data [21] | 1.2-V POD | PIN_BC18 |
| DDR4B_DQ22 | Data [22] | 1.2-V POD | PIN_BC15 |
| DDR4B_DQ23 | Data [23] | 1.2-V POD | PIN_BD15 |

| | | | |
|------------|-----------|-----------|----------|
| DDR4B_DQ24 | Data [24] | 1.2-V POD | PIN_AP8 |
| DDR4B_DQ25 | Data [25] | 1.2-V POD | PIN_AT12 |
| DDR4B_DQ26 | Data [26] | 1.2-V POD | PIN_AR8 |
| DDR4B_DQ27 | Data [27] | 1.2-V POD | PIN_AR12 |
| DDR4B_DQ28 | Data [28] | 1.2-V POD | PIN_AP11 |
| DDR4B_DQ29 | Data [29] | 1.2-V POD | PIN_AN11 |
| DDR4B_DQ30 | Data [30] | 1.2-V POD | PIN_AR11 |
| DDR4B_DQ31 | Data [31] | 1.2-V POD | PIN_AR7 |
| DDR4B_DQ32 | Data [32] | 1.2-V POD | PIN_AU8 |
| DDR4B_DQ33 | Data [33] | 1.2-V POD | PIN_AU9 |
| DDR4B_DQ34 | Data [34] | 1.2-V POD | PIN_AP10 |
| DDR4B_DQ35 | Data [35] | 1.2-V POD | PIN_AT9 |
| DDR4B_DQ36 | Data [36] | 1.2-V POD | PIN_AV8 |
| DDR4B_DQ37 | Data [37] | 1.2-V POD | PIN_AU7 |
| DDR4B_DQ38 | Data [38] | 1.2-V POD | PIN_AT7 |
| DDR4B_DQ39 | Data [39] | 1.2-V POD | PIN_AT6 |
| DDR4B_DQ40 | Data [40] | 1.2-V POD | PIN_AK13 |
| DDR4B_DQ41 | Data [41] | 1.2-V POD | PIN_AN12 |
| DDR4B_DQ42 | Data [42] | 1.2-V POD | PIN_AF12 |
| DDR4B_DQ43 | Data [43] | 1.2-V POD | PIN_AE12 |
| DDR4B_DQ44 | Data [44] | 1.2-V POD | PIN_AK12 |
| DDR4B_DQ45 | Data [45] | 1.2-V POD | PIN_AM12 |
| DDR4B_DQ46 | Data [46] | 1.2-V POD | PIN_AH13 |
| DDR4B_DQ47 | Data [47] | 1.2-V POD | PIN_AG13 |
| DDR4B_DQ48 | Data [48] | 1.2-V POD | PIN_AP15 |
| DDR4B_DQ49 | Data [49] | 1.2-V POD | PIN_AN16 |
| DDR4B_DQ50 | Data [50] | 1.2-V POD | PIN_AR17 |
| DDR4B_DQ51 | Data [51] | 1.2-V POD | PIN_AN17 |
| DDR4B_DQ52 | Data [52] | 1.2-V POD | PIN_AM15 |
| DDR4B_DQ53 | Data [53] | 1.2-V POD | PIN_AT17 |
| DDR4B_DQ54 | Data [54] | 1.2-V POD | PIN_AN15 |
| DDR4B_DQ55 | Data [55] | 1.2-V POD | PIN_AR16 |
| DDR4B_DQ56 | Data [56] | 1.2-V POD | PIN_AC12 |
| DDR4B_DQ57 | Data [57] | 1.2-V POD | PIN_AG14 |

| | | | |
|--------------|------------------|---------------------------|----------|
| DDR4B_DQ58 | Data [58] | 1.2-V POD | PIN_AD14 |
| DDR4B_DQ59 | Data [59] | 1.2-V POD | PIN_AB14 |
| DDR4B_DQ60 | Data [60] | 1.2-V POD | PIN_AE14 |
| DDR4B_DQ61 | Data [61] | 1.2-V POD | PIN_AC11 |
| DDR4B_DQ62 | Data [62] | 1.2-V POD | PIN_AB13 |
| DDR4B_DQ63 | Data [63] | 1.2-V POD | PIN_AD13 |
| DDR4B_DQS0 | Data Strobe p[0] | DIFFERENTIAL 1.2-V POD | PIN_AV18 |
| DDR4B_DQS_n0 | Data Strobe n[0] | DIFFERENTIAL 1.2-V POD | PIN_AU18 |
| DDR4B_DQS1 | Data Strobe p[1] | DIFFERENTIAL 1.2-V POD | PIN_BA15 |
| DDR4B_DQS_n1 | Data Strobe n[1] | DIFFERENTIAL 1.2-V POD | PIN_BA16 |
| DDR4B_DQS2 | Data Strobe p[2] | DIFFERENTIAL 1.2-V POD | PIN_BD13 |
| DDR4B_DQS_n2 | Data Strobe n[2] | DIFFERENTIAL 1.2-V POD | PIN_BD14 |
| DDR4B_DQS3 | Data Strobe p[3] | DIFFERENTIAL 1.2-V POD | PIN_AT10 |
| DDR4B_DQS_n3 | Data Strobe n[3] | DIFFERENTIAL 1.2-V POD | PIN_AU10 |
| DDR4B_DQS4 | Data Strobe p[4] | DIFFERENTIAL 1.2-V POD | PIN_AV7 |
| DDR4B_DQS_n4 | Data Strobe n[4] | DIFFERENTIAL 1.2-V POD | PIN_AV6 |
| DDR4B_DQS5 | Data Strobe p[5] | DIFFERENTIAL 1.2-V POD | PIN_AG12 |
| DDR4B_DQS_n5 | Data Strobe n[5] | DIFFERENTIAL 1.2-V POD | PIN_AH12 |
| DDR4B_DQS6 | Data Strobe p[6] | DIFFERENTIAL 1.2-V POD | PIN_AT15 |
| DDR4B_DQS_n6 | Data Strobe n[6] | DIFFERENTIAL 1.2-V POD | PIN_AT16 |
| DDR4B_DQS7 | Data Strobe p[7] | DIFFERENTIAL 1.2-V | PIN_AC13 |

| | | | |
|--------------|---------------------------|---------------------------|----------|
| | | POD | |
| DDR4B_DQS_n7 | Data Strobe n[7] | DIFFERENTIAL 1.2-V POD | PIN_AB12 |
| DDR4B_DBI_n0 | Data Bus Inversion [0] | 1.2-V POD | PIN_AT19 |
| DDR4B_DBI_n1 | Data Bus Inversion [1] | 1.2-V POD | PIN_AY18 |
| DDR4B_DBI_n2 | Data Bus Inversion [2] | 1.2-V POD | PIN_BC17 |
| DDR4B_DBI_n3 | Data Bus Inversion [3] | 1.2-V POD | PIN_AT11 |
| DDR4B_DBI_n4 | Data Bus Inversion [4] | 1.2-V POD | PIN_AN10 |
| DDR4B_DBI_n5 | Data Bus Inversion [5] | 1.2-V POD | PIN_AL12 |
| DDR4B_DBI_n6 | Data Bus Inversion [6] | 1.2-V POD | PIN_AU17 |
| DDR4B_DBI_n7 | Data Bus Inversion [7] | 1.2-V POD | PIN_AF14 |
| DDR4B_A0 | Address [0] | SSTL-12 | PIN_AW15 |
| DDR4B_A1 | Address [1] | SSTL-12 | PIN_AW14 |
| DDR4B_A2 | Address [2] | SSTL-12 | PIN_AW13 |
| DDR4B_A3 | Address [3] | SSTL-12 | PIN_AY13 |
| DDR4B_A4 | Address [4] | SSTL-12 | PIN_AY14 |
| DDR4B_A5 | Address [5] | SSTL-12 | PIN_BA14 |
| DDR4B_A6 | Address [6] | SSTL-12 | PIN_BA12 |
| DDR4B_A7 | Address [7] | SSTL-12 | PIN_BB12 |
| DDR4B_A8 | Address [8] | SSTL-12 | PIN_AU13 |
| DDR4B_A9 | Address [9] | SSTL-12 | PIN_AV13 |
| DDR4B_A10 | Address [10] | SSTL-12 | PIN_AY11 |
| DDR4B_A11 | Address [11] | SSTL-12 | PIN_BA11 |
| DDR4B_A12 | Address [12] | SSTL-12 | PIN_AK14 |
| DDR4B_A13 | Address [13] | SSTL-12 | PIN_AM13 |
| DDR4B_A14 | Address [14]/ | SSTL-12 | PIN_AN13 |

| | | | |
|---------------|--|----------------------------|----------|
| | WE_n | | |
| DDR4B_A15 | Address [15]/ CAS_n | SSTL-12 | PIN_AL14 |
| DDR4B_A16 | Address [16]/ RAS_n | SSTL-12 | PIN_AM14 |
| DDR4B_BA0 | Bank Select [0] | SSTL-12 | PIN_AU14 |
| DDR4B_BA1 | Bank Select [1] | SSTL-12 | PIN_AP13 |
| DDR4B_BG0 | Bank Group Select [0] | SSTL-12 | PIN_AR13 |
| DDR4B_BG1 | Bank Group Select [1] | SSTL-12 | PIN_BB14 |
| DDR4B_CK0 | Clock p0 | DIFFERENTIAL 1.2-V SSTL | PIN_BA10 |
| DDR4B_CK_n0 | Clock n0 | DIFFERENTIAL 1.2-V SSTL | PIN_BB10 |
| DDR4B_CK1 | Clock p1 | DIFFERENTIAL 1.2-V SSTL | PIN_AV10 |
| DDR4B_CK_n1 | Clock n1 | DIFFERENTIAL 1.2-V SSTL | PIN_AV11 |
| DDR4B_CKE0 | Clock Enable pin 0 | SSTL-12 | PIN_BC10 |
| DDR4B_CKE1 | Clock Enable pin 1 | SSTL-12 | PIN_AW9 |
| DDR4B_ODT0 | On Die Termination[0] | SSTL-12 | PIN_BC11 |
| DDR4B_ODT1 | On Die Termination[1] | SSTL-12 | PIN_AW10 |
| DDR4B_CS_n0 | Chip Select [0] | SSTL-12 | PIN_BC13 |
| DDR4B_CS_n1 | Chip Select [1] | SSTL-12 | PIN_AV12 |
| DDR4B_PAR | Command and Address Parity Input | SSTL-12 | PIN_BB8 |
| DDR4B_ALERT_n | Register ALERT_n output | SSTL-12 | PIN_AP18 |
| DDR4B_ACT_n | Activation Command Input | SSTL-12 | PIN_BC12 |
| DDR4B_RESET_n | Chip Reset | 1.2 V | PIN_BB13 |

| | | | |
|----------------|-------------------------------|-------|----------|
| DDR4B_EVENT_n | Chip Temperature Event | 1.2 V | PIN_BD11 |
| DDR4B_SDA | Chip I2C Serial Data | 1.2 V | PIN_AP9 |
| DDR4B_SCL | Chip I2C Serial Clock | 1.2 V | PIN_AP16 |
| DDR4B_REFCLK_p | DDR4 A port Reference Clock p | LVDS | PIN_AP14 |
| DDR4B_REFCLK_n | DDR4 A port Reference Clock n | LVDS | PIN_AR14 |

2.9 QDRII+ SRAM

The development board supports four independent QDRII+ SRAM memory devices for very-high speed and low-latency memory access. Each of QDRII+ has a x18 interface, providing addressing to a device of up to a 8MB (not including parity bits). The QDRII+ has separate read and write data ports with DDR signaling at up to 550 MHz.

Table 2-14, **Table 2-15**, **Table 2-16** and **Table 2-17** lists the QDRII+ SRAM Bank A, B, C and D pin assignments, signal names relative to the Arria 10 GX device, in respectively.

Table 2-14 QDRII+ SRAM A Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|----------------|--------------------|------------------------|
| QDRIIA_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_V12 |
| QDRIIA_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_V13 |
| QDRIIA_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_N10 |
| QDRIIA_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_M10 |
| QDRIIA_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_P11 |

| | | | |
|------------|--------------------|--------------------|---------|
| QDRIIA_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_N11 |
| QDRIIA_A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_M9 |
| QDRIIA_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_M8 |
| QDRIIA_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_N7 |
| QDRIIA_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_N8 |
| QDRIIA_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_P10 |
| QDRIIA_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_P9 |
| QDRIIA_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_N6 |
| QDRIIA_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_M7 |
| QDRIIA_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_L10 |
| QDRIIA_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_L7 |
| QDRIIA_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_K7 |
| QDRIIA_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_K8 |
| QDRIIA_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_J9 |
| QDRIIA_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_L6 |
| QDRIIA_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_K6 |
| QDRIIA_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_J6 |
| QDRIIA_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_E8 |
| QDRIIA_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_E9 |
| QDRIIA_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_D8 |
| QDRIIA_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_E11 |
| QDRIIA_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_D9 |
| QDRIIA_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_C8 |
| QDRIIA_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_D10 |
| QDRIIA_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_C10 |
| QDRIIA_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_D11 |
| QDRIIA_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_C13 |
| QDRIIA_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_C12 |
| QDRIIA_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_B12 |
| QDRIIA_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_A12 |
| QDRIIA_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_D13 |
| QDRIIA_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_A11 |
| QDRIIA_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_A10 |
| QDRIIA_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_E13 |

| | | | |
|---------------|--------------------------|---------------------------------|---------|
| QDRIIA_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_B10 |
| QDRIIA_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_R12 |
| QDRIIA_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_R14 |
| QDRIIA_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_N12 |
| QDRIIA_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_M13 |
| QDRIIA_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_M12 |
| QDRIIA_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_M14 |
| QDRIIA_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_L12 |
| QDRIIA_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_K12 |
| QDRIIA_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_G10 |
| QDRIIA_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_H12 |
| QDRIIA_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_H11 |
| QDRIIA_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_J14 |
| QDRIIA_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_K14 |
| QDRIIA_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_K13 |
| QDRIIA_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_L14 |
| QDRIIA_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_N13 |
| QDRIIA_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_P13 |
| QDRIIA_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_R13 |
| QDRIIA_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_C11 |
| QDRIIA_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_B13 |
| QDRIIA_K_P | Clock P | Differential 1.8-V HSTL Class I | PIN_F12 |
| QDRIIA_K_N | Clock N | Differential 1.8-V HSTL Class I | PIN_E12 |
| QDRIIA_CQ_P | Echo clock P | 1.8-V HSTL Class I | PIN_J13 |
| QDRIIA_CQ_N | Echo clock N | 1.8-V HSTL Class I | PIN_H13 |
| QDRIIA_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_U9 |
| QDRIIA_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_U8 |
| QDRIIA_DOFF_n | DLL enable | 1.8-V HSTL Class I | PIN_R9 |
| QDRIIA_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_T10 |
| QDRIIA_QVLD | Valid Output | 1.8-V HSTL Class I | PIN_T14 |

Table 2-15 QDRII+ SRAM B Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------------|--------------------|------------------------|
| QDRIIB_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_L16 |
| QDRIIB_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_L15 |
| QDRIIB_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_E14 |
| QDRIIB_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_D14 |
| QDRIIB_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_G14 |
| QDRIIB_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_F14 |
| QDRIIB_A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_D15 |
| QDRIIB_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_C15 |
| QDRIIB_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_F15 |
| QDRIIB_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_F16 |
| QDRIIB_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_H15 |
| QDRIIB_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_G15 |
| QDRIIB_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_E16 |
| QDRIIB_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_D16 |
| QDRIIB_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_E17 |
| QDRIIB_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_G17 |
| QDRIIB_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_G18 |
| QDRIIB_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_L17 |
| QDRIIB_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_K17 |
| QDRIIB_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_H17 |
| QDRIIB_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_H18 |
| QDRIIB_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_K18 |
| QDRIIB_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_G19 |
| QDRIIB_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_F19 |
| QDRIIB_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_E19 |
| QDRIIB_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_D19 |
| QDRIIB_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_C18 |
| QDRIIB_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_B19 |
| QDRIIB_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_B20 |

| | | | |
|---------------|----------------------|---------------------------------|---------|
| QDRIIB_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_C20 |
| QDRIIB_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_F20 |
| QDRIIB_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_L20 |
| QDRIIB_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_J20 |
| QDRIIB_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_N20 |
| QDRIIB_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_L19 |
| QDRIIB_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_L21 |
| QDRIIB_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_K19 |
| QDRIIB_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_J19 |
| QDRIIB_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_M20 |
| QDRIIB_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_M19 |
| QDRIIB_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_L22 |
| QDRIIB_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_K23 |
| QDRIIB_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_J23 |
| QDRIIB_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_H23 |
| QDRIIB_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_H21 |
| QDRIIB_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_H22 |
| QDRIIB_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_G23 |
| QDRIIB_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_F21 |
| QDRIIB_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_E23 |
| QDRIIB_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_A22 |
| QDRIIB_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_B22 |
| QDRIIB_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_C22 |
| QDRIIB_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_B23 |
| QDRIIB_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_A21 |
| QDRIIB_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_C21 |
| QDRIIB_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_E22 |
| QDRIIB_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_F22 |
| QDRIIB_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_G22 |
| QDRIIB_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_G20 |
| QDRIIB_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_H20 |
| QDRIIB_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_K21 |
| QDRIIB_K_n | Clock N | Differential 1.8-V HSTL | PIN_J21 |

| | | Class I | |
|---------------|--------------------------|--------------------|---------|
| QDRIIB_CQ_p | Echo clock P | 1.8-V HSTL Class I | PIN_D23 |
| QDRIIB_CQ_n | Echo clock N | 1.8-V HSTL Class I | PIN_C23 |
| QDRIIB_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_J16 |
| QDRIIB_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_K16 |
| QDRIIB_DOFF_n | PLL Turn Off | 1.8-V HSTL Class I | PIN_H16 |
| QDRIIB_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_M17 |
| QDRIIB_QVLD | Valid Output Indicator | 1.8-V HSTL Class I | PIN_K22 |

Table 2-16 QDRII+ SRAM C Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-----------------|--------------------|------------------------|
| QDRIIC_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_D25 |
| QDRIIC_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_D26 |
| QDRIIC_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_A26 |
| QDRIIC_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_A27 |
| QDRIIC_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_A29 |
| QDRIIC_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_A30 |
| QDRIIC_A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_B27 |
| QDRIIC_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_B28 |
| QDRIIC_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_C27 |
| QDRIIC_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_C28 |
| QDRIIC_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_B29 |
| QDRIIC_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_B30 |
| QDRIIC_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_C30 |
| QDRIIC_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_C31 |
| QDRIIC_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_L25 |
| QDRIIC_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_K24 |
| QDRIIC_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_J24 |
| QDRIIC_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_G25 |

| | | | |
|------------|--------------------|--------------------|----------|
| QDRIIC_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_F25 |
| QDRIIC_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_J25 |
| QDRIIC_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_H25 |
| QDRIIC_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_J26 |
| QDRIIC_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_AD33 |
| QDRIIC_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_AC33 |
| QDRIIC_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_AB33 |
| QDRIIC_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_AB34 |
| QDRIIC_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_AA34 |
| QDRIIC_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_Y34 |
| QDRIIC_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_W34 |
| QDRIIC_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_AC35 |
| QDRIIC_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_AA35 |
| QDRIIC_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_AF36 |
| QDRIIC_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_AE36 |
| QDRIIC_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_AD34 |
| QDRIIC_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_AE34 |
| QDRIIC_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_AE33 |
| QDRIIC_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_AE32 |
| QDRIIC_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_AE31 |
| QDRIIC_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_AF32 |
| QDRIIC_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_AF31 |
| QDRIIC_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_T36 |
| QDRIIC_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_R36 |
| QDRIIC_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_P35 |
| QDRIIC_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_N36 |
| QDRIIC_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_N37 |
| QDRIIC_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_M38 |
| QDRIIC_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_M39 |
| QDRIIC_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_N38 |
| QDRIIC_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_P36 |
| QDRIIC_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_Y36 |
| QDRIIC_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_M37 |
| QDRIIC_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_M35 |

| | | | |
|---------------|--------------------------|---------------------------------|----------|
| QDRIIC_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_T34 |
| QDRIIC_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_N35 |
| QDRIIC_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_T35 |
| QDRIIC_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_U35 |
| QDRIIC_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_V35 |
| QDRIIC_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_W35 |
| QDRIIC_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_AB35 |
| QDRIIC_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_AD35 |
| QDRIIC_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_AF34 |
| QDRIIC_K_n | Clock N | Differential 1.8-V HSTL Class I | PIN_AF35 |
| QDRIIC_CQ_p | Echo clock P | 1.8-V HSTL Class I | PIN_AD36 |
| QDRIIC_CQ_n | Echo clock N | 1.8-V HSTL Class I | PIN_AC36 |
| QDRIIC_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_E26 |
| QDRIIC_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_F26 |
| QDRIIC_DOFF_n | PLL Turn Off | 1.8-V HSTL Class I | PIN_D24 |
| QDRIIC_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_B25 |
| QDRIIC_QVLD | Valid Output Indicator | 1.8-V HSTL Class I | PIN_U34 |

Table 2-17 QDRII+ SRAM D Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|----------------|--------------------|------------------------|
| QDRIID_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_Y32 |
| QDRIID_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_W33 |
| QDRIID_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_P34 |
| QDRIID_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_P33 |
| QDRIID_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_L32 |
| QDRIID_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_K32 |
| QDRIID_A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_R34 |
| QDRIID_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_R33 |
| QDRIID_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_T32 |

| | | | |
|------------|--------------------|--------------------|---------|
| QDRIID_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_R32 |
| QDRIID_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_N32 |
| QDRIID_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_M32 |
| QDRIID_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_T31 |
| QDRIID_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_R31 |
| QDRIID_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_K38 |
| QDRIID_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_L37 |
| QDRIID_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_K36 |
| QDRIID_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_N33 |
| QDRIID_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_M33 |
| QDRIID_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_L39 |
| QDRIID_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_K39 |
| QDRIID_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_L35 |
| QDRIID_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_E36 |
| QDRIID_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_F34 |
| QDRIID_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_F39 |
| QDRIID_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_F36 |
| QDRIID_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_D33 |
| QDRIID_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_F31 |
| QDRIID_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_G30 |
| QDRIID_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_H30 |
| QDRIID_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_G29 |
| QDRIID_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_E33 |
| QDRIID_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_G39 |
| QDRIID_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_E37 |
| QDRIID_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_F37 |
| QDRIID_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_E34 |
| QDRIID_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_D36 |
| QDRIID_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_C37 |
| QDRIID_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_D35 |
| QDRIID_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_D34 |
| QDRIID_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_L29 |
| QDRIID_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_N30 |
| QDRIID_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_K29 |

| | | | |
|---------------|--------------------------|---------------------------------|---------|
| QDRIID_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_J29 |
| QDRIID_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_M30 |
| QDRIID_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_J30 |
| QDRIID_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_N31 |
| QDRIID_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_P31 |
| QDRIID_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_H33 |
| QDRIID_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_G34 |
| QDRIID_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_G33 |
| QDRIID_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_L31 |
| QDRIID_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_J31 |
| QDRIID_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_K31 |
| QDRIID_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_L30 |
| QDRIID_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_M29 |
| QDRIID_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_M28 |
| QDRIID_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_N28 |
| QDRIID_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_F30 |
| QDRIID_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_E31 |
| QDRIID_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_F32 |
| QDRIID_K_n | Clock N | Differential 1.8-V HSTL Class I | PIN_E32 |
| QDRIID_CQ_p | Echo clock P | 1.8-V HSTL Class I | PIN_G35 |
| QDRIID_CQ_n | Echo clock N | 1.8-V HSTL Class I | PIN_F35 |
| QDRIID_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_V33 |
| QDRIID_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_V32 |
| QDRIID_DOFF_n | PLL Turn Off | 1.8-V HSTL Class I | PIN_W31 |
| QDRIID_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_Y33 |
| QDRIID_QVLD | ValidOutput Indicator | 1.8-V HSTL Class I | PIN_P28 |

2.10 QSFP+ Ports

The development board has four independent 40G QSFP+ connectors that use one transceiver channel each from the Arria 10 GX FPGA device. These modules take in serial data from the Arria 10 GX FPGA device and transform them to optical signals. The board includes cage assemblies for the QSFP+ connectors. **Figure 2-15** shows the connections between the QSFP+ and Arria 10 GX FPGA.

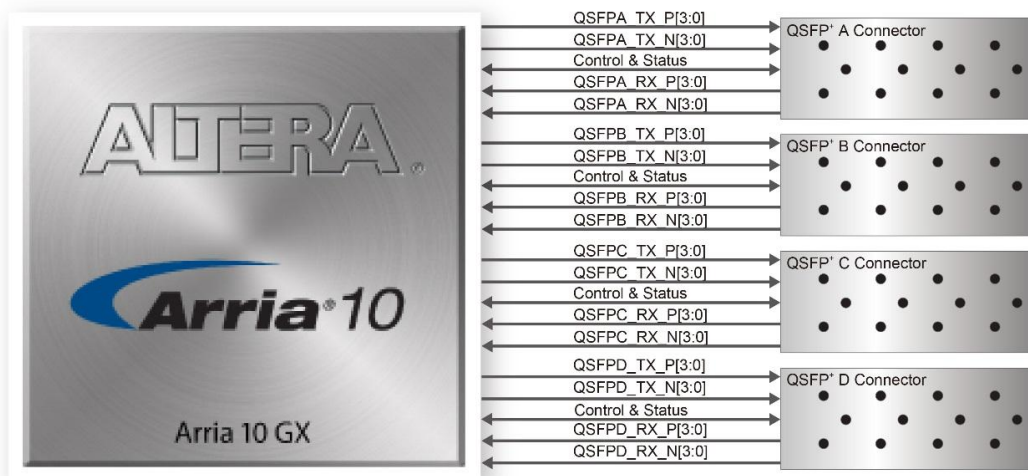


Figure 2-15 Connection between the QSFP+ and Arria GX FPGA

Table 2-18, **Table 2-19**, **Table 2-20** and **Table 2-21** list the QSFP+ A, B, C and D pin assignments and signal names relative to the Arria 10 GX device.

Table 2-18 QSFP+ A Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------------------------|--------------|------------------------|
| QSFPA_TX_P0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_BD5 |
| QSFPA_TX_N0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_BD6 |

| | | | |
|-------------------|-------------------------------|------------|----------|
| QSFPA_RX_P0 | Receiver data of channel 0 | 1.4-V PCML | PIN_BB5 |
| QSFPA_RX_N0 | Receiver data of channel 0 | 1.4-V PCML | PIN_BB6 |
| QSFPA_TX_P1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_BC3 |
| QSFPA_TX_N1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_BC4 |
| QSFPA_RX_P1 | Receiver data of channel 1 | 1.4-V PCML | PIN_AY5 |
| QSFPA_RX_N1 | Receiver data of channel 1 | 1.4-V PCML | PIN_AY6 |
| QSFPA_TX_P2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_BB1 |
| QSFPA_TX_N2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_BB2 |
| QSFPA_RX_P2 | Receiver data of channel 2 | 1.4-V PCML | PIN_BA3 |
| QSFPA_RX_N2 | Receiver data of channel 2 | 1.4-V PCML | PIN_BA4 |
| QSFPA_TX_P3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_AY1 |
| QSFPA_TX_N3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_AY2 |
| QSFPA_RX_P3 | Receiver data of channel 3 | 1.4-V PCML | PIN_AW3 |
| QSFPA_RX_N3 | Receiver data of channel 3 | 1.4-V PCML | PIN_AW4 |
| QSFPA_MOD_SEL_n | Module Select | 1.8V | PIN_AC10 |
| QSFPA_RST_n | Module Reset | 1.8V | PIN_AA9 |
| QSFPA_SCL | 2-wire serial interface clock | 1.8V | PIN_AA10 |
| QSFPA_SDA | 2-wire serial interface data | 1.8V | PIN_Y9 |
| QSFPA_LP_MODE | Low Power Mode | 1.8V | PIN_AB10 |
| QSFPA_INTERRUPT_n | Interrupt | 1.8V | PIN_AB9 |
| QSFPA_MOD_PRS_n | Module Present | 1.8V | PIN_AG10 |

Table 2-19 QSFP+ B Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------------------------|--------------|------------------------|
| QSFPB_TX_P0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_AP1 |
| QSFPB_TX_N0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_AP2 |
| QSFPB_RX_P0 | Receiver data of channel 0 | 1.4-V PCML | PIN_AN3 |
| QSFPB_RX_N0 | Receiver data of channel 0 | 1.4-V PCML | PIN_AN4 |
| QSFPB_TX_P1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_AM1 |
| QSFPB_TX_N1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_AM2 |
| QSFPB_RX_P1 | Receiver data of channel 1 | 1.4-V PCML | PIN_AL3 |
| QSFPB_RX_N1 | Receiver data of channel 1 | 1.4-V PCML | PIN_AL4 |

| | | | |
|-------------------|-------------------------------|------------|----------|
| QSFPB_TX_P2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_AK1 |
| QSFPB_TX_N2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_AK2 |
| QSFPB_RX_P2 | Receiver data of channel 2 | 1.4-V PCML | PIN_AJ3 |
| QSFPB_RX_N2 | Receiver data of channel 2 | 1.4-V PCML | PIN_AJ4 |
| QSFPB_TX_P3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_AH1 |
| QSFPB_TX_N3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_AH2 |
| QSFPB_RX_P3 | Receiver data of channel 3 | 1.4-V PCML | PIN_AG3 |
| QSFPB_RX_N3 | Receiver data of channel 3 | 1.4-V PCML | PIN_AG4 |
| QSFPB_MOD_SEL_n | Module Select | 1.8V | PIN_AP6 |
| QSFPB_RST_n | Module Reset | 1.8V | PIN_AR6 |
| QSFPB_SCL | 2-wire serial interface clock | 1.8V | PIN_AM7 |
| QSFPB_SDA | 2-wire serial interface data | 1.8V | PIN_AM8 |
| QSFPB_LP_MODE | Low Power Mode | 1.8V | PIN_AM10 |
| QSFPB_INTERRUPT_n | Interrupt | 1.8V | PIN_AK9 |
| QSFPB_MOD_PRS_n | Module Present | 1.8V | PIN_AL10 |

Table 2-20 QSFP+ C Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------------------------|--------------|------------------------|
| QSFPC_TX_P0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_AB1 |
| QSFPC_TX_N0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_AB2 |
| QSFPC_RX_P0 | Receiver data of channel 0 | 1.4-V PCML | PIN_AA3 |
| QSFPC_RX_N0 | Receiver data of channel 0 | 1.4-V PCML | PIN_AA4 |
| QSFPC_TX_P1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_Y1 |
| QSFPC_TX_N1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_Y2 |
| QSFPC_RX_P1 | Receiver data of channel 1 | 1.4-V PCML | PIN_W3 |
| QSFPC_RX_N1 | Receiver data of channel 1 | 1.4-V PCML | PIN_W4 |
| QSFPC_TX_P2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_V1 |
| QSFPC_TX_N2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_V2 |
| QSFPC_RX_P2 | Receiver data of channel 2 | 1.4-V PCML | PIN_U3 |
| QSFPC_RX_N2 | Receiver data of channel 2 | 1.4-V PCML | PIN_U4 |

| | | | |
|-------------------|-------------------------------|------------|----------|
| QSFPC_TX_P3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_T1 |
| QSFPC_TX_N3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_T2 |
| QSFPC_RX_P3 | Receiver data of channel 3 | 1.4-V PCML | PIN_R3 |
| QSFPC_RX_N3 | Receiver data of channel 3 | 1.4-V PCML | PIN_R4 |
| QSFPC_MOD_SEL_n | Module Select | 1.8V | PIN_AL9 |
| QSFPC_RST_n | Module Reset | 1.8V | PIN_AJ9 |
| QSFPC_SCL | 2-wire serial interface clock | 1.8V | PIN_AL11 |
| QSFPC_SDA | 2-wire serial interface data | 1.8V | PIN_AK11 |
| QSFPC_LP_MODE | Low Power Mode | 1.8V | PIN_AH10 |
| QSFPC_INTERRUPT_n | Interrupt | 1.8V | PIN_AD11 |
| QSFPC_MOD_PRS_n | Module Present | 1.8V | PIN_AD10 |

Table 2-21 QSFP+ D Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|-------------------------------|--------------|------------------------|
| QSFPD_TX_P0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_K1 |
| QSFPD_TX_N0 | Transmitter data of channel 0 | 1.4-V PCML | PIN_K2 |
| QSFPD_RX_P0 | Receiver data of channel 0 | 1.4-V PCML | PIN_J3 |
| QSFPD_RX_N0 | Receiver data of channel 0 | 1.4-V PCML | PIN_J4 |
| QSFPD_TX_P1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_H1 |
| QSFPD_TX_N1 | Transmitter data of channel 1 | 1.4-V PCML | PIN_H2 |
| QSFPD_RX_P1 | Receiver data of channel 1 | 1.4-V PCML | PIN_G3 |
| QSFPD_RX_N1 | Receiver data of channel 1 | 1.4-V PCML | PIN_G4 |
| QSFPD_TX_P2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_F1 |
| QSFPD_TX_N2 | Transmitter data of channel 2 | 1.4-V PCML | PIN_F2 |
| QSFPD_RX_P2 | Receiver data of channel 2 | 1.4-V PCML | PIN_E3 |
| QSFPD_RX_N2 | Receiver data of channel 2 | 1.4-V PCML | PIN_E4 |
| QSFPD_TX_P3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_D1 |
| QSFPD_TX_N3 | Transmitter data of channel 3 | 1.4-V PCML | PIN_D2 |
| QSFPD_RX_P3 | Receiver data of channel 3 | 1.4-V PCML | PIN_D5 |
| QSFPD_RX_N3 | Receiver data of channel 3 | 1.4-V PCML | PIN_D6 |
| QSFPD_MOD_SEL_n | Module Select | 1.8V | PIN_AA11 |

| | | | |
|-------------------|-------------------------------|------|----------|
| QSFPD_RST_n | Module Reset | 1.8V | PIN_Y11 |
| QSFPD_SCL | 2-wire serial interface clock | 1.8V | PIN_W9 |
| QSFPD_SDA | 2-wire serial interface data | 1.8V | PIN_W10 |
| QSFPD_LP_MODE | Low Power Mode | 1.8V | PIN_AA12 |
| QSFPD_INTERRUPT_n | Interrupt | 1.8V | PIN_W13 |
| QSFPD_MOD_PRS_n | Module Present | 1.8V | PIN_Y12 |

2.11 PCI Express

The FPGA development board is designed to fit entirely into a PC motherboard with x8 or x16 PCI Express slot. Utilizing built-in transceivers on a Arria 10 GX device, it is able to provide a fully integrated PCI Express-compliant solution for multi-lane (x1, x4, and x8) applications. With the PCI Express hard IP block incorporated in the Arria 10 GX device, it will allow users to implement simple and fast protocol, as well as saving logic resources for logic application. **Figure 2-16** presents the pin connection established between the Arria 10 GX and PCI Express.

The PCI Express interface supports complete PCI Express Gen1 at 2.5Gbps/lane, Gen2 at 5.0Gbps/lane, and Gen3 at 8.0Gbps/lane protocol stack solution compliant to PCI Express base specification 3.0 that includes PHY-MAC, Data Link, and transaction layer circuitry embedded in PCI Express hard IP blocks.

Please note that it is a requirement that you connect the PCIe external power connector to 6-pin 12V DC power connector in the FPGA to avoid FPGA damage due to insufficient power. The PCIE_REFCLK_p signal is a differential input that is driven from the PC motherboard on this board through the PCIe edge connector. A DIP switch (SW5) is connected to the PCI Express to allow different configurations to enable a x1, x4, or x8 PCIe.

Table 2-22 summarizes the PCI Express pin assignments of the signal names relative to the Arria 10 GX FPGA.

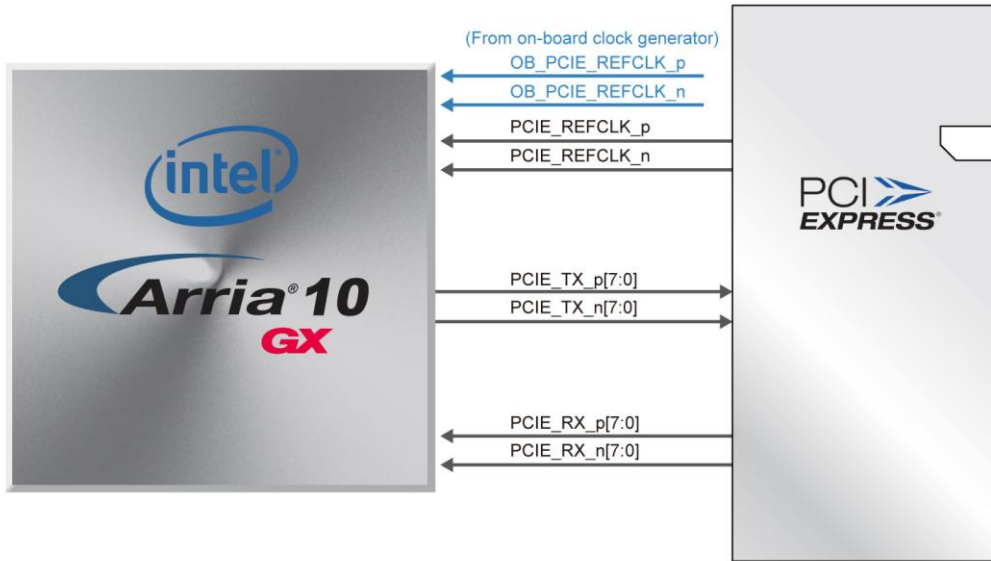


Figure 2-16 PCI Express pin connection

Table 2-22 PCI Express Pin Assignments, Schematic Signal Names, and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|--------------------------|--------------|------------------------|
| PCIE_TX_p0 | Add-in card transmit bus | 1.4-V PCML | PIN_AV44 |
| PCIE_TX_n0 | Add-in card transmit bus | 1.4-V PCML | PIN_AV43 |
| PCIE_TX_p1 | Add-in card transmit bus | 1.4-V PCML | PIN_AT44 |
| PCIE_TX_n1 | Add-in card transmit bus | 1.4-V PCML | PIN_AT43 |
| PCIE_TX_p2 | Add-in card transmit bus | 1.4-V PCML | PIN_AP44 |
| PCIE_TX_n2 | Add-in card transmit bus | 1.4-V PCML | PIN_AP43 |
| PCIE_TX_p3 | Add-in card transmit bus | 1.4-V PCML | PIN_AM44 |
| PCIE_TX_n3 | Add-in card transmit bus | 1.4-V PCML | PIN_AM43 |
| PCIE_TX_p4 | Add-in card transmit bus | 1.4-V PCML | PIN_AK44 |
| PCIE_TX_n4 | Add-in card transmit bus | 1.4-V PCML | PIN_AK43 |
| PCIE_TX_p5 | Add-in card transmit bus | 1.4-V PCML | PIN_AH44 |
| PCIE_TX_n5 | Add-in card transmit bus | 1.4-V PCML | PIN_AH43 |
| PCIE_TX_p6 | Add-in card transmit bus | 1.4-V PCML | PIN_AF44 |
| PCIE_TX_n6 | Add-in card transmit bus | 1.4-V PCML | PIN_AF43 |
| PCIE_TX_p7 | Add-in card transmit bus | 1.4-V PCML | PIN_AD44 |

| | | | |
|------------------|---|------------|----------|
| PCIE_TX_n7 | Add-in card transmit bus | 1.4-V PCML | PIN_AD43 |
| PCIE_RX_p0 | Add-in card receive bus | 1.4-V PCML | PIN_AU42 |
| PCIE_RX_n0 | Add-in card receive bus | 1.4-V PCML | PIN_AU41 |
| PCIE_RX_p1 | Add-in card receive bus | 1.4-V PCML | PIN_AR42 |
| PCIE_RX_n1 | Add-in card receive bus | 1.4-V PCML | PIN_AR41 |
| PCIE_RX_p2 | Add-in card receive bus | 1.4-V PCML | PIN_AN42 |
| PCIE_RX_n2 | Add-in card receive bus | 1.4-V PCML | PIN_AN41 |
| PCIE_RX_p3 | Add-in card receive bus | 1.4-V PCML | PIN_AL42 |
| PCIE_RX_n3 | Add-in card receive bus | 1.4-V PCML | PIN_AL41 |
| PCIE_RX_p4 | Add-in card receive bus | 1.4-V PCML | PIN_AJ42 |
| PCIE_RX_n4 | Add-in card receive bus | 1.4-V PCML | PIN_AJ41 |
| PCIE_RX_p5 | Add-in card receive bus | 1.4-V PCML | PIN_AG42 |
| PCIE_RX_n5 | Add-in card receive bus | 1.4-V PCML | PIN_AG41 |
| PCIE_RX_p6 | Add-in card receive bus | 1.4-V PCML | PIN_AE42 |
| PCIE_RX_n6 | Add-in card receive bus | 1.4-V PCML | PIN_AE41 |
| PCIE_RX_p7 | Add-in card receive bus | 1.4-V PCML | PIN_AC42 |
| PCIE_RX_n7 | Add-in card receive bus | 1.4-VPCML | PIN_AC41 |
| PCIE_REFCLK_p | Motherboard reference clock | HCSL | PIN_AH40 |
| PCIE_REFCLK_n | Motherboard reference clock | HCSL | PIN_AH39 |
| OB_PCIE_REFCLK_p | On-board PCIe reference clock | LVDS | PIN_AK40 |
| OB_PCIE_REFCLK_n | On-board PCIe reference clock | LVDS | PIN_AK39 |
| PCIE_PERST_n | Reset | 1.8-V | PIN_AT25 |
| PCIE_SMBCLK | SMB clock | 1.8-V | PIN_AM25 |
| PCIE_SMBDAT | SMB data | 1.8-V | PIN_AR24 |
| PCIE_WAKE_n | Wake signal | 1.8-V | PIN_AN26 |
| PCIE_PRSENT1n | Hot plug detect | - | - |
| PCIE_PRSENT2n_x1 | Hot plug detect x1 PCIe slot enabled using SW5 dip switch | - | - |
| PCIE_PRSENT2n_x4 | Hot plug detect x4 PCIe slot enabled using SW5 dip switch | - | - |
| PCIE_PRSENT2n_x8 | Hot plug detect x8 PCIe slot enabled using SW5 dip switch | - | - |

2.12 RS-422 Expansion Header

The FPGA board has one RS-422 Expansion Header J1 for expansion function. The pin-out of J1 is shown in **Figure 2-17**. Pin3 to Pin8 of the expansion header are bi-direction 1.8V GPIO (Total 6 GPIOs) connected to the FPGA directly. Note that the nets **RS422_DE** and **RJ45_LED_L** are connected to FPGA dedicated clock input and can be configured as two single-ended clock signals or one differential clock signal. Users can use Terasic defined **RS422-RJ45 board** and **TUB (Timing and UART Board)** for RS422 /UART or external clock inputs applications.

Table 2-23 lists the RS-422 pin assignments, signal names and functions.

RS422 Expansion Header

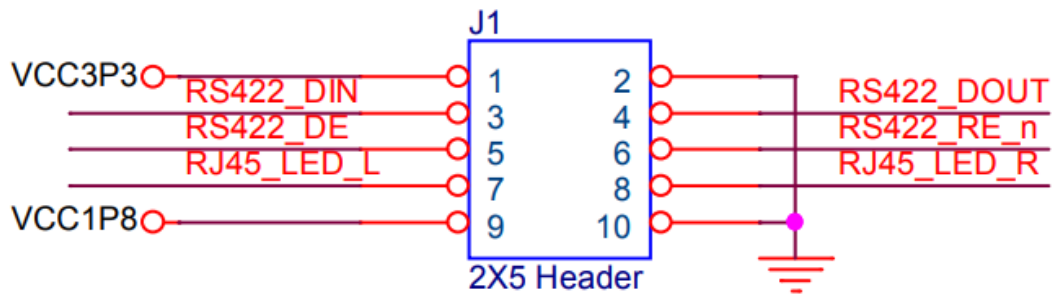


Figure 2-17 Pin-out of RS-422 Expansion Header

Table 2-23 RS-422 Pin Assignments, Schematic Signal Names and Functions

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|-----------------------|------------------------|--------------|------------------------|
| RS422_DE | Bi-direction 1.8V GPIO | 1.8V | PIN_AN8 |
| RS422_DIN | Bi-direction 1.8V GPIO | | PIN_AM9 |
| RS422_DOUT | Bi-direction 1.8V GPIO | | PIN_W14 |
| RS422_RE_n | Bi-direction 1.8V GPIO | | PIN_AD9 |

| | | | |
|------------|------------------------|--|--|
| RJ45_LED_L | Bi-direction 1.8V GPIO | | |
| RJ45_LED_R | Bi-direction 1.8V GPIO | | |

System Builder

This chapter describes how users can create a custom design project for the FPGA board from a software tool named System Builder.

3.1 Introduction

The System Builder is a Windows based software utility. It is designed to help users create a Quartus Prime project for the FPGA board within minutes. The Quartus Prime project files generated include:

- Quartus Prime Project File (.qpf)
- Quartus Prime Setting File (.qsf)
- Top-Level Design File (.v)
- External PLL Controller (.v)
- Synopsis Design Constraints file (.sdc)
- Pin Assignment Document (.htm)

The System Builder not only can generate the files above, but can also provide error-checking rules to handle situation that are prone to errors. The common mistakes that users encounter are the following:

- Board damaged for wrong pin/bank voltage assignment.
- Board malfunction caused by wrong device connections or missing pin counts

for connected ends.

- Performance dropped because of improper pin assignments

3.2 General Design Flow

This section will introduce the general design flow to build a project for the FPGA board via the System Builder. The general design flow is illustrated in the Figure 3-1.

Users should launch System Builder and create a new project according to their design requirements. When users complete the settings, the System Builder will generate two major files which include top-level design file (.v) and the Quartus Prime setting file (.qsf).

The top-level design file contains top-level Verilog wrapper for users to add their own design/logic. The Quartus Prime setting file contains information such as FPGA device type, top-level pin assignment, and I/O standard for each user-defined I/O pin.

Finally, Quartus Prime programmer must be used to download SOF file to the FPGA board using JTAG interface.

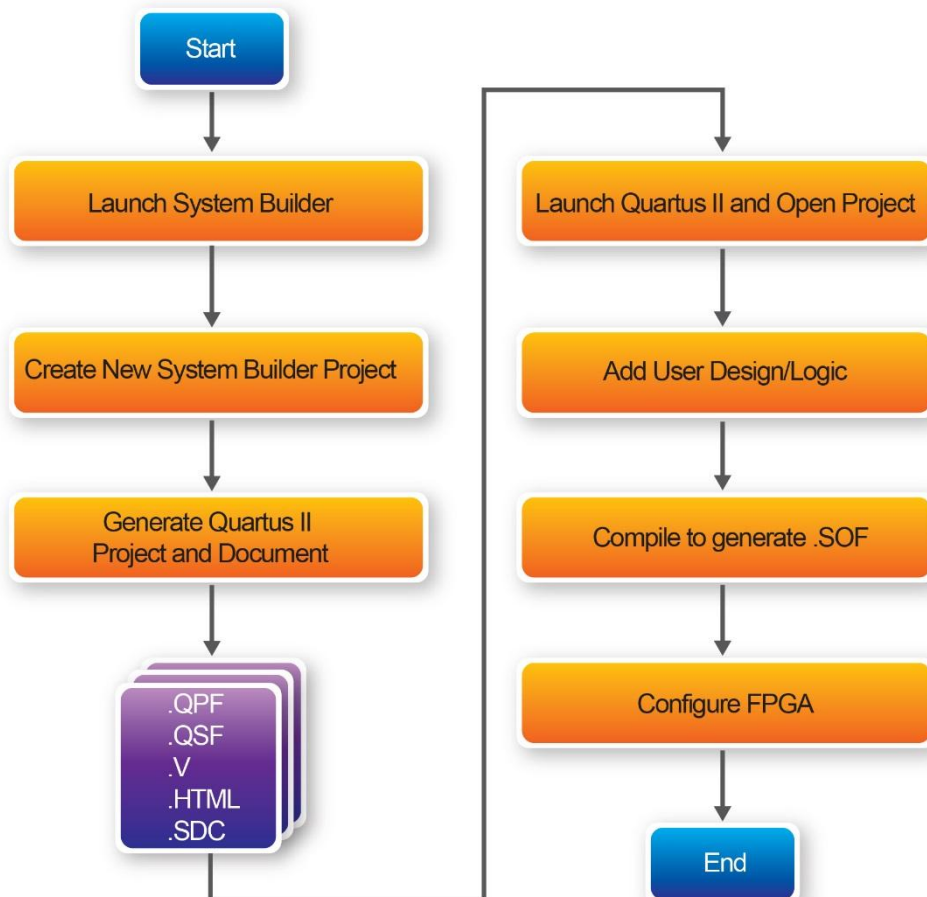


Figure 3-1 The general design flow of building a project

3.3 Using System Builder

This section provides the detailed procedures on how the System Builder is used.

■ Install and Launch the System Builder

The System Builder is located under the directory: "**Tools\SystemBuilder**" in the System CD. Users can copy the entire folder to the host computer without installing the utility. Please execute the SystemBuilder.exe on the host computer, as shown in **Figure 3-2**.

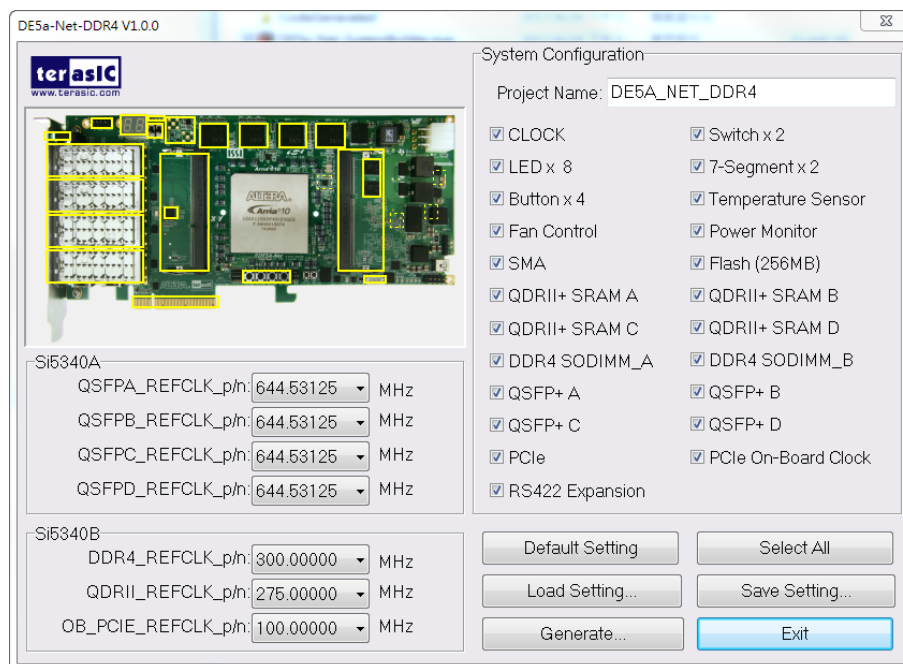


Figure 3-2 The System Builder window

■ Enter Project Name

The project name entered in the circled area as shown in **Figure 3-3**, will be assigned automatically as the name of the top-level design entry.

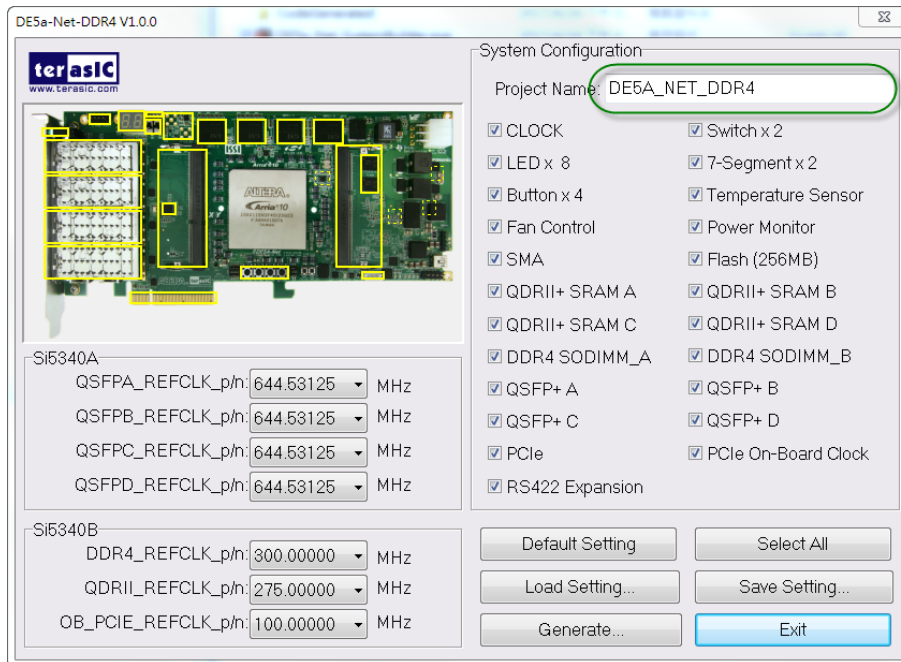


Figure 3-3 The Quartus project name

■ System Configuration

Users are given the flexibility of enabling their choices of components connected to the FPGA under System Configuration, as shown in **Figure 3-4**. Each component of the FPGA board is listed to be enabled or disabled according to users' needs. If a component is enabled, the System Builder will automatically generate the associated pin assignments including its pin name, pin location, pin direction, and I/O standards.

Note: The pin assignments for some components (e.g. DDR4 and QSFP+) require associated controller codes in the Quartus project or it would result in compilation error. Hence please do not select them if they are not needed in the design. To use the DDR4 controller, please refer to the DDR4 SDRAM demonstration in Chapter 6.

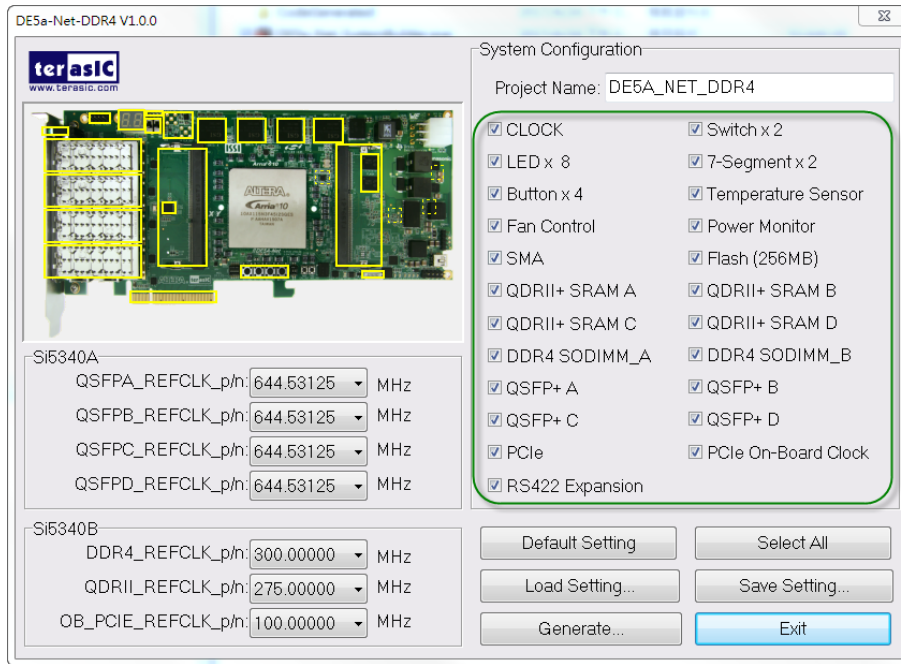


Figure 3-4 System Configuration group

■ Programmable Oscillator

There are two external oscillators on-board that provide reference clocks for the following signals

QSFP*_REFCLK, QSFPB_REFCLK, QSFPC_REFCLK, QSPFD_REFCLK, DDR4_REFCLK, QDRII_REFCLK, and OB_PCIE_REFCLK. To use these clock, users can select the desired frequency on the Programmable Oscillator group, as shown in **Figure 3-5**. QDRII, DDR4, or QSFP+ must be checked before users can start to specify the desired frequency in the programmable oscillators.

As the Quartus project is created, System Builder automatically generates the associated controller according to users' desired frequency in Verilog which facilitates users' implementation as no additional control code is required to configure the programmable oscillator.

Note: If users need to dynamically change the frequency, they would need to modify the generated control code themselves.

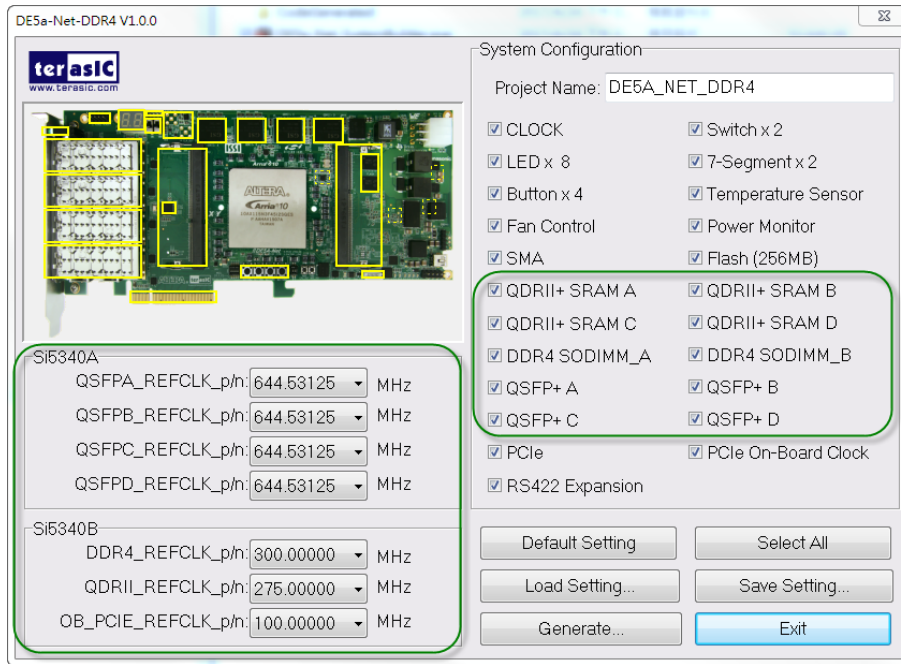


Figure 3-5 External programmable oscillators

■ Project Setting Management

The System Builder also provides functions to restore default setting, load a setting, and save board configuration file, as shown in **Figure 3-6**. Users can save the current board configuration information into a .cfg file and load it into the System Builder.

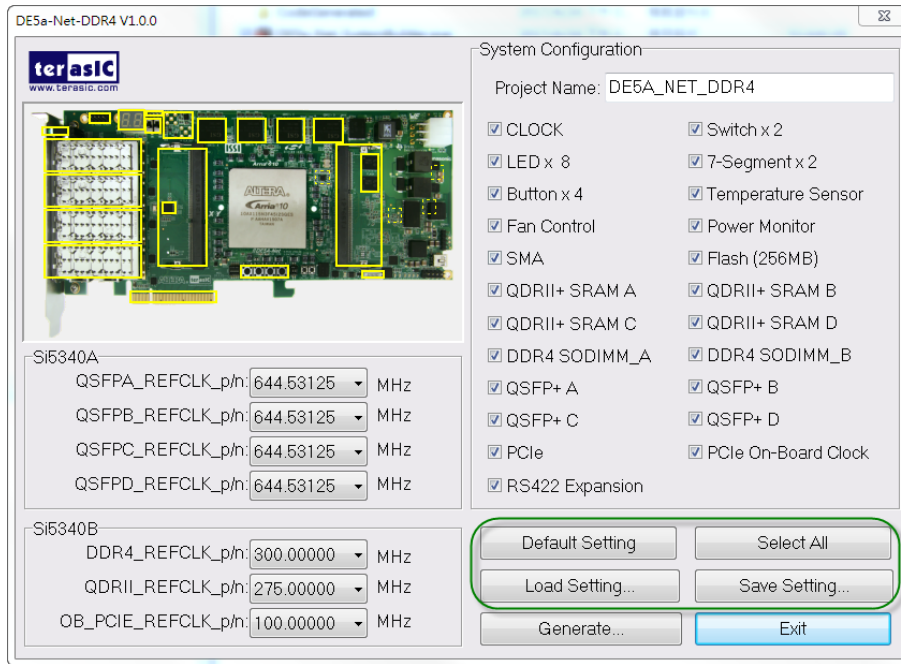


Figure 3-6 Project Settings

■ Project Generation

When users press the Generate button, the System Builder will generate the corresponding Quartus Prime files and documents as listed in the **Table 3-1** in the directory specified by the user.

Table 3-1 Files generated by the System Builder

| No. | Filename | Description |
|-----|-----------------------|---|
| 1 | <Project name>.v | Top Level Verilog File for Quartus Prime |
| 2 | Si5340_controller (*) | Si5340A and Si5340B External Oscillator Controller IP |
| 3 | <Project name>.qpf | Quartus Prime Project File |
| 4 | <Project name>.qsf | Quartus Prime Setting File |
| 5 | <Project name>.sdc | Synopsis Design Constraints File for Quartus Prime |

| | | |
|---|--------------------|-------------------------|
| 6 | <Project name>.htm | Pin Assignment Document |
|---|--------------------|-------------------------|

(*) The Si5340_controller is a folder which contains the Verilog files for the configuration of Si5340A and Si5340B.

Users can add custom logic into the project and compile the project in Quartus Prime to generate the SRAM Object File (.sof).

For Si5340A, its controller will be instantiated in the Quartus Prime top-level file, as listed below:

```

=====
// Configure SI5340A
=====
`define SI5340A_POWER_DOWN 3'h0
`define SI5340A_644M53125 3'h1
`define SI5340A_322M265625 3'h2
`define SI5340A_312M5 3'h3
`define SI5340A_250M 3'h4
`define SI5340A_156M25 3'h5
`define SI5340A_125M 3'h6
`define SI5340A_100M 3'h7

wire si5340a_controller_start;
assign si5340a_controller_start = ~BUTTON[0];

si5340a_controller si5340a_controller(
    .iCLK(CLK_50_B2J),
    .iRST_n(CPU_RESET_n),
    .iStart(si5340a_controller_start),
    .iPLL_OUT0_FREQ_SEL(`SI5340A_644M53125), //QSFP-A
    .iPLL_OUT1_FREQ_SEL(`SI5340A_644M53125), //QSFP-B
    .iPLL_OUT2_FREQ_SEL(`SI5340A_644M53125), //QSFP-C
    .iPLL_OUT3_FREQ_SEL(`SI5340A_644M53125), //QSFP-D
    .oPLL_I2C_ID_READ_ERROR(),
    .I2C_CLK(SI5340A_I2C_SCL),
    .I2C_DATA(SI5340A_I2C_SDA),
    .oPLL_REG_CONFIG_DONE()
);

assign SI5340A_OE_n = 1'b0;
assign SI5340A_RST_n = CPU_RESET_n;

```

For Si5340B, its controller will be instantiated in the Quartus Prime top-level file, as listed below:

```
//=====  
// Configure SI5340B  
//=====  
`define DDR300_QDR275_PCIE100 5'd0  
`define DDR266_QDR275_PCIE100 5'd1  
`define DDR233_QDR275_PCIE100 5'd2  
`define DDR200_QDR275_PCIE100 5'd3  
`define DDR150_QDR275_PCIE100 5'd4  
`define DDR300_QDR250_PCIE100 5'd5  
`define DDR266_QDR250_PCIE100 5'd6  
`define DDR233_QDR250_PCIE100 5'd7  
`define DDR200_QDR250_PCIE100 5'd8  
`define DDR150_QDR250_PCIE100 5'd9  
`define DDR300_QDR225_PCIE100 5'd10  
`define DDR266_QDR225_PCIE100 5'd11  
`define DDR233_QDR225_PCIE100 5'd12  
`define DDR200_QDR225_PCIE100 5'd13  
`define DDR150_QDR225_PCIE100 5'd14  
  
wire si5340b_controller_start;  
wire config_done;  
  
assign si5340b_controller_start = ~BUTTON[0];  
  
si5340b_controller si5340b_controller(  
    .iCLK(CLK_50_B2J),  
    .iRST_n(CPU_RESET_n),  
    .iStart(si5340b_controller_start),  
    .iPLL_OUT_FREQ_SEL(`DDR300_QDR275_PCIE100),  
    .I2C_CLK(SI5340B_I2C_SCL),  
    .I2C_DATA(SI5340B_I2C_SDA),  
    .oPLL_REG_CONFIG_DONE(config_done)  
);  
  
assign SI5340B_OE_n = 1'b0;  
assign SI5340B_RST_n = CPU_RESET_n;
```

If the dynamic configuration for the oscillators required, users need to modify the code according to users' desired behavior.

Flash Programming

As you develop your own project using the Altera tools, you can program the flash memory device so that your own design loads from flash memory into the FPGA on power up. This chapter will describe how to use Altera Quartus Prime Programmer Tool to program the common flash interface (CFI) flash memory device on the FPGA board. The Arria 10 GX FPGA development board ships with the CFI flash device preprogrammed with a default factory FPGA configuration for running the Parallel Flash Loader design example.

4.1 CFI Flash Memory Map

Table 4-1 shows the default memory contents of two interlaced 1Gb (128MB) CFI flash device. Each flash device has a 16-bit data bus and the two combined flash devices allow for a 32-bit flash memory interface. For the factory default code to run correctly and update designs in the user memory, this memory map must not be altered.

Table 4-1 Flash Memory Map (Byte Address)

| Block Description | Size(KB) | Address Range |
|------------------------|----------|-------------------------|
| PFL option bits | 64 | 0x00030000 – 0x0003FFFF |
| Factory hardware | 44,032 | 0x00040000 – 0x02B3FFFF |
| User hardware | 44,032 | 0x02B40000 – 0x0563FFFF |
| Factory software | 8,192 | 0x05640000 – 0x05E3FFFF |
| User software and data | 165,632 | 0x05E40000 – 0x0FFFFFFF |

For user application, user hardware must be stored with start address 0x02B40000, and the user's software is suggested to be stored with start address 0x05E40000. The NIOS II EDS tool nios-2-flash-programmer is used for programming the flash. Before programming, users need to translate their Quartus .sof and NIOS II .elf files into the .flash which is used by the nios-2-flash-programmer. For .sof to .flash translation, NIOS II EDS tool sof2flash can be used. For the .elf to .flash translation, NIOS II EDS tool elf2flash can be used. For convenience, the System CD contains a batch file for file translation and flash programming with users given .sof and .elf file.

4.2 FPGA Configure Operation

Here is the procedure to enable FPGA configuration from Flash:

1. Please make sure the FPGA configuration data has been stored in the CFI flash.
2. Set the FPGA configuration mode to FPPx32 mode by setting SW3 MSEL[0:2] as 000 as shown in **Figure 4-1**.
3. Specify the configuration of the FPGA using the default Factory Configuration or User Configuration by setting SW3 according to **Figure 4-2**.
4. Power on the FPGA board or press MAX_RST button if board is already powered on
5. When configuration is completed, the green Configure Done LED will light. If there is error, the red Configure Error LED will light.



Figure 4-1 SW3 MSEL[0:2]=000



4-2 Configuration Image Selection

4.3 Flash Programming with Users Design

Users can program the flash memory device so that a custom design loads from flash memory into the FPGA on power up. For convenience, the translation and programming batch files are available on the following folder in the System CD.

Demonstrations/Hello/flash_programming_batch

There folder contains five files as shown in **Table 4-2**

Table 4-2 Content of flash_programming_batch folder

| Files Name | Description |
|-------------------|--|
| DE5a_NET_PFL.sof | Parallel Flash Loader Design |
| flash_program.bat | Top batch file to download DE5a_NET_PFL.sof and launch batch flash_program.sh |
| flash_program.sh | Translate .sof and .elf into .flash and programming flash with the generated .flash file |
| DE5a_NET.sof | Hardware design file for Hello Demo |
| HELLO_NIOS.elf | Software design file for Hello Demo |

To apply the batch file to users'.sof and .elf file, users can change the.sof and .elf filename in the flash_program.sh file as shown in **Figure 4-3**.

```
sof2flash --input=DE5a_NET.sof --output=flash_hw.flash --offset=0x02B40000 --pfl --opti
elf2flash --base=0x0 --end=0x0FFFFFFF --reset=0x05E40000 --input=HELLO_NIOS.elf --outpu
```

Figure 4-3 Change to users' .sof and .elf filename

If your design does not contain a NIOS II processor, users can add “#” to comment (disable) the elf2flash and nios-flash-programmer commands (marked with green lines as below) in the flash_program.sh file as shown in **Figure 4-4**.

```
#convert to .flash as factory image
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" sof2flash --input=DE5a_NET.sof --output=flash_hw.flash --o
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" elf2flash --base=0x0 --end=0x0FFFFFFF --reset=0x05E40000 -

#Programming with .flash
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0 flash_hw.flash
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0 flash_sw.flash
```

Figure 4-4 Disable .elf translation and programming

If your design includes a NIOS II processor and the NIOS II program is stored on external memory, users must to perform following items so the NIOS II program can be boot from flash successfully:

1. QSYS should include a Flash controller for the CFI Flash on the development board. Please ensure that the base address of the controller is 0x00, as shown in **Figure 4-5**.
2. In NIOS II processor options, select FLASH as reset vector memory and specify 0x05E40000 as reset vector, as shown in **Figure 4-6**.

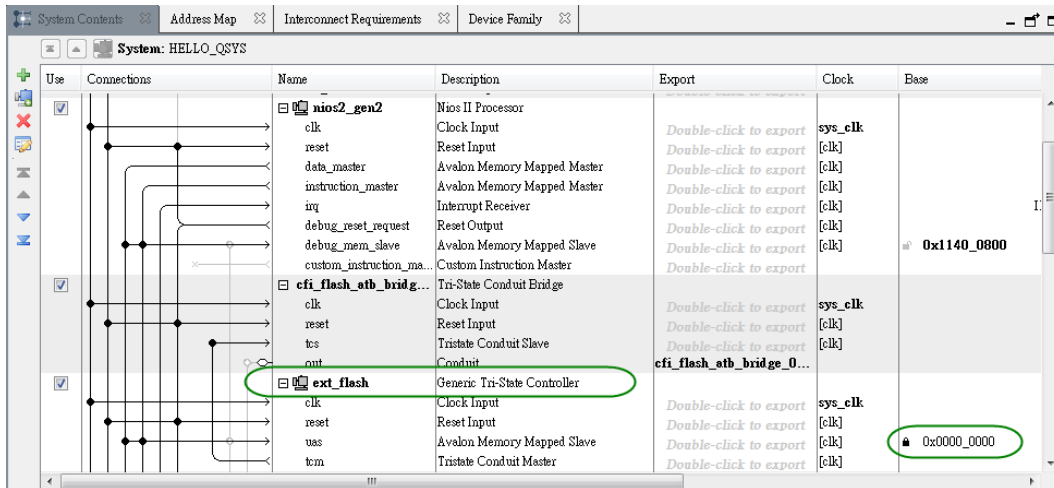


Figure 4-5 Flash Controller Settings in QSYS

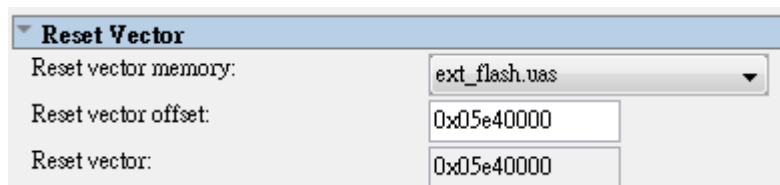


Figure 4-6 Reset Vector Settings for NIOS II Processor

For implementation detail, users can refer the Hello example located in the CD folder: Demonstrations/ Hello

4.4 Restore Factory Settings

This section describes how to restore the original factory contents to the flash memory device on the FPGA development board. Perform the following instructions:

1. Make sure the Nios II EDS and USB-Blaster II driver are installed.
2. Make sure the FPGA board and PC are connected with an UBS Cable.
3. Power on the FPGA board.
4. Copy the "Demonstrations/PFL/flash_programming_batch" folder under the CD

to your PC's local drive.

5. Execute the batch file flash_program.bat to start flash programming.
6. Power off the FPGA Board.
7. Set FPGA configure mode as FPPx32 Mode by setting SW3 MSEL[0:2] to 000.
8. Specify configuration of the FPGA to Factory Hardware by setting the FACTORY_LOAD dip in SW3 to the '1' position.
9. Power on the FPGA Board, and the Configure Done LED should light.

Except for programming the Flash with the default code PFL, the batch file also writes PFL (Parallel Flash Loader) Option Bits data into the address 0x30000. The option bits data specifies 0x2B40000 as start address of your hardware design.

The NIOS II EDS tool nios-2-flash-programmer programs the Flash based on the Parallel Flasher Loader design in the FPGA. The Parallel Flash Loader design is included in the default code PFL and the source code is available in the folder Demonstrations/ PFL in System CD.

Peripheral Reference Design

This chapter introduces DE5a-NET peripheral interface reference designs. It mainly introduces Si5340 chip which is a programmable clock generator. We provide two ways (Pure RTL IP and NIOS/Qsys System) respectively to show how to control Si5340 to output desired frequencies, as well as how to control the fan speed. The source codes and tool of these examples are all available on the System CD.

5.1 Temperature Monitor: Board Protection

This section introduces a Terasic Temperature Monitor IP which can be used to monitor board temperature and raise an alert when the FPGA temperature reaches the specified threshold. **Figure 5-1** shows the block diagram for this demonstration. The User Logic keeps LED blinking to indicate the monitor status is normal. This function can be enabled or disabled through the enable pin. Alert Temperature is set to 80°C by default. The measured FPGA temperature is displayed on the two 7-segment displays. If the FPGA temperature exceeds 80°C, the alert is triggered. It would cause the User Logic to be disabled and the LED will be turned off.

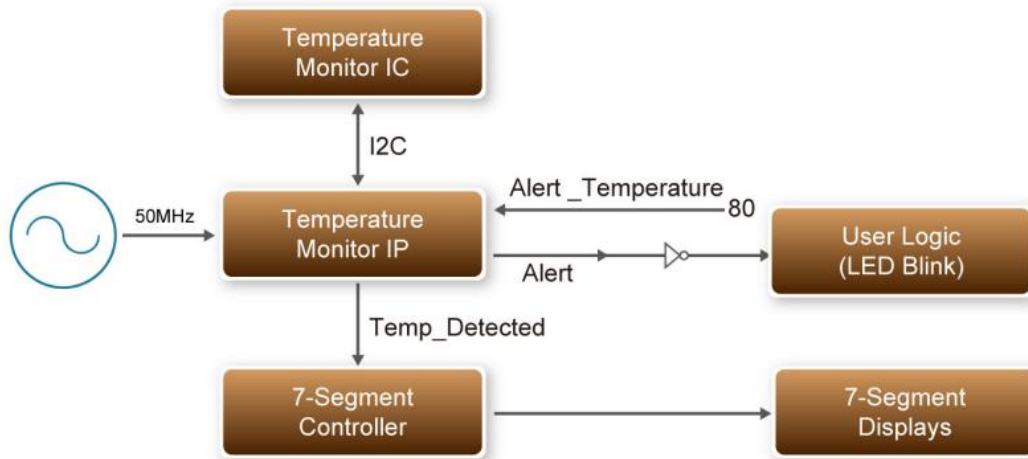


Figure 5-1 Block Diagram of Temperature Monitor

■ Temperature Monitor IP

The temperature Monitor IP is called as Temperature_Monitor. **Table 5-1** shows the interface of the Temperature_Monitor IP. Users need to provide 50 MHz clock signal for this IP and connect the I2C bus to the temperature chip. Users also need to specify the temperature threshold through the Alert_Temperature. If the measured FPGA temperature exceeds the threshold specified by Alert_Temperature, the Alert signal will be pulled high. Users need to turn off the board immediately to avoid any damage to the FPGA.

Table 5-1 Temperature_Monitor Interface

| Port | Direction | Description |
|------------------------|-----------|---|
| iClk50 | input | Provide 50 MHz clock signal to the IP |
| TEMP_I2C_SCL | output | I2C SCL pin for Temperature Sensor |
| TEMP_I2C_SDA | In/out | I2C SDA pin for Temperature Sensor |
| Alert_Temperature[9:0] | input | Set alter temperature in degree C. Typically 80 is recommended. Do NOT exceed 95. |
| Alert | output | High active. When FPGA temperature |

| | | |
|----------------|--------|--|
| | | exceeds the threshold specified by Alert_temperature, the alert pin will be pulled high. Note the Alert signal will never be pulled low once it is pulled except when FPGA is reconfigured. |
| .Temp_Detected | output | Optional Interface. This interface is reserved to provide current measured FPGA temperature. |

■ Demonstration File Locations

- Hardware project directory: Board_Protection
- Bitstream used: Board_Protection.sof
- Demo batch file : Board_Protection\demo_batch\test_ub2.bat

■ Demonstration Setup and Instructions

- Make sure Quartus Prime is installed on your PC.
- Power on the FPGA board.
- Use the USB Cable to connect your PC and the FPGA board and install USB Blaster II driver if necessary.
- Execute the demo batch file “test_ub2.bat” under the batch file folder, Board_Protection\demo_batch.
- After FPGA is configured, the four user LEDs will blink. The measured temperature will be displayed in the two 7-segment displays.
- If the FPGA temperature exceed 80s degree, the LEDs will stop blinking. For test, please modify the Alert_Temperature to a lower value to so the measured temperature value can exceed the temperature specified by .Alert_Temperature the in finally.

5.2 Configure Si5340A/B in RTL

There are two Silicon Labs Si5340 clock generators on DE5a-net FPGA board can provide adjustable frequency reference clock (See **Figure 5-2**) for QSFP, QDRII, DDR4 and PCIe interfaces, etc. Each Si5340 clock generator can output four groups differential frequencies from 100Hz ~ 712.5Mhz though I2C interface configuration. This chapter will show you how to use FPGA RTL IP to configure each Si5340 PLL and generate users desired output frequency to each peripheral. In the following instruction, the two Si5340 chips will be named as Si5340A and Si5340B respectively.

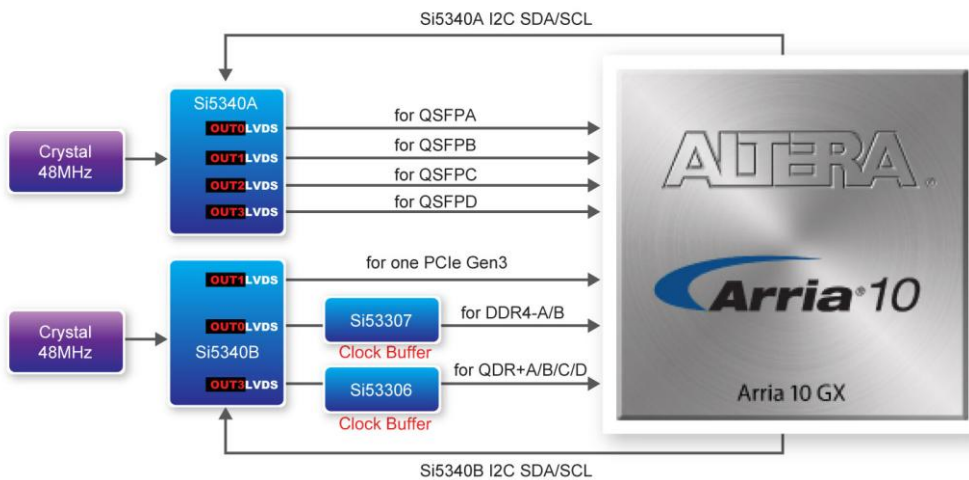


Figure 5-2 Si570 Block diagram

■ Creating Si5340 Control IP

The Si5340 control IP is located in the folder: "\Demonstrations\si5340_control_ip" in the System CD. Also, System Builder tool (locate in System CD) can be used to help users to set Si5340 to output desired frequencies, and generate a Quartus project with control IP. In System Builder window, when checking the boxes of SFP, QDRII , DDR4 and PCIe interfaces, Si5340 corresponding output channels will become available and users can select desired frequencies. For example, when checking QSFP+ A box (See **Figure 5-3**), SI5340A QSFP A_REFCLK_P/N can provide seven frequencies from 100Mhz to 644.5312Mhz for users selecting.

As shown in **Figure 5-4**, if all the receiving Si5340 reference clock interface boxes are checked, then, every frequency channel of the two Si5340 chips is controllable by users.

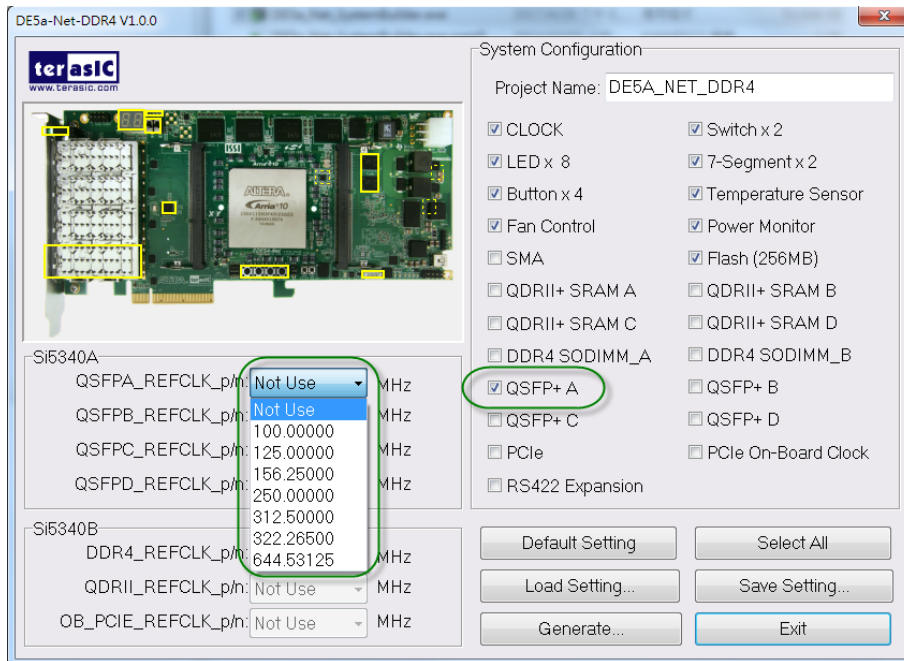


Figure 5-3 Enable Si5340A clock on System Builder

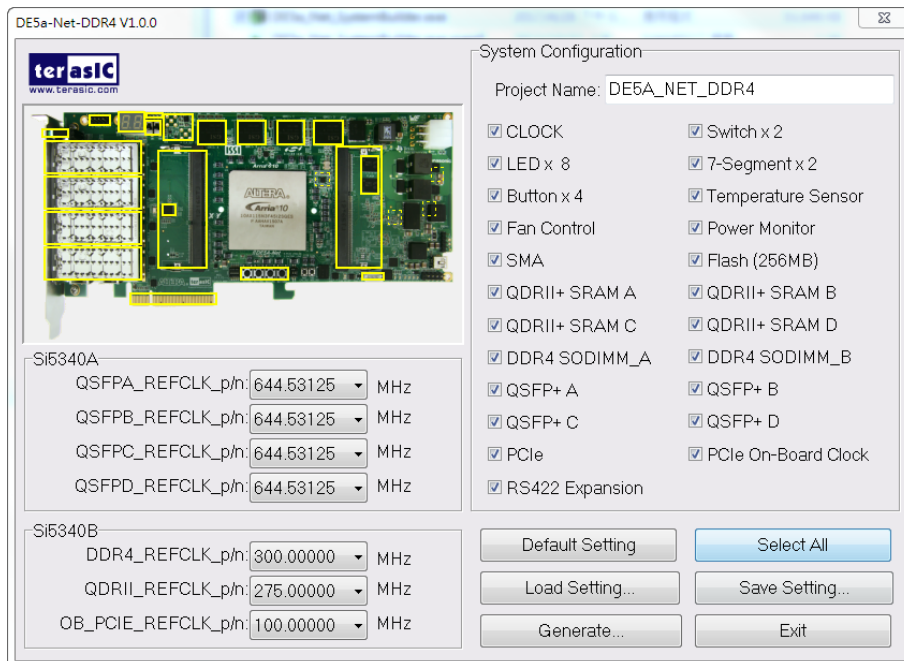


Figure 5-4 Enable Si5340A and Si5340B clock on System Builder

Click "**Generate**" button, then, open the Quartus Project generated by System Builder, the control IPs for Si5340A and Si5340B can be found in the top level file.

```
//=====
// Configure SI5340A
//=====
`define SI5340A_POWER_DOWN 3'h0
`define SI5340A_644M53125 3'h1
`define SI5340A_322M265625 3'h2
`define SI5340A_312M5 3'h3
`define SI5340A_250M 3'h4
`define SI5340A_156M25 3'h5
`define SI5340A_125M 3'h6
`define SI5340A_100M 3'h7

wire si5340a_controller_start;
assign si5340a_controller_start = ~BUTTON[0];

si5340a_controller si5340a_controller(
    .iCLK(CLK_50_B2J),
    .iRST_n(CPU_RESET_n),
    .iStart(si5340a_controller_start),
    .iPLL_OUT0_FREQ_SEL(`SI5340A_100M)//QSFP-A
    .iPLL_OUT1_FREQ_SEL(`SI5340A_125M)//QSFP-B
    .iPLL_OUT2_FREQ_SEL(`SI5340A_156M25)//QSFP-C
    .iPLL_OUT3_FREQ_SEL(`SI5340A_644M53125)//QSFP-D
    .I2C_CLK(SI5340A_I2C_SCL),
    .I2C_DATA(SI5340A_I2C_SDA),
    .oPLL_REG_CONFIG_DONE()
);

assign SI5340A_OE_n = 1'b0;
assign SI5340A_RST_n = CPU_RESET_n;
```

```
//=====
// Configure SI5340B
//=====
`define DDR300_QDR275_PCIE100 5'd0
`define DDR266_QDR275_PCIE100 5'd1
`define DDR233_QDR275_PCIE100 5'd2
`define DDR200_QDR275_PCIE100 5'd3
`define DDR150_QDR275_PCIE100 5'd4
`define DDR300_QDR250_PCIE100 5'd5
`define DDR266_QDR250_PCIE100 5'd6
`define DDR233_QDR250_PCIE100 5'd7
`define DDR200_QDR250_PCIE100 5'd8
`define DDR150_QDR250_PCIE100 5'd9
`define DDR300_QDR225_PCIE100 5'd10
`define DDR266_QDR225_PCIE100 5'd11
`define DDR233_QDR225_PCIE100 5'd12
`define DDR200_QDR225_PCIE100 5'd13
`define DDR150_QDR225_PCIE100 5'd14

wire si5340b_controller_start;

wire config_done;
assign si5340b_controller_start = ~BUTTON[0];

si5340b_controller si5340b_controller(
    .iCLK(CLK_50_B2J),
    .iRST_n(CPU_RESET_n),
    .iStart(si5340b_controller_start),
    .iPLL_OUT_FREQ_SEL(`DDR300_QDR270_PCIE100),
    .I2C_CLK(SI5340B_I2C_SCL),
    .I2C_DATA(SI5340B_I2C_SDA),
    .oPLL_REG_CONFIG_DONE(config_done)
);

assign SI5340B_OE_n = 1'b0;
```

If the output frequency doesn't need to be modified, users can just add their own User Logic and compile it, and then, Si5340 can output desired frequencies. At the same time, System Builder will set Clock constrain according user's preset frequency in a SDC file (as shown in).

```
assign SI5340B_RST_n = CPU_RESET_n;
```

```
*****
# This .sdc file is created by Terasic Tool.
# Users are recommended to modify this file 1
*****

*****
# Create Clock
*****
create_clock -period "100.000000 MHz" [get_p
create_clock -period "100.000000 MHz" [get_p
create_clock -period "50.000000 MHz" [get_por
create_clock -period "50.000000 MHz" [get_por
create_clock -period "50.000000 MHz" [get_por
create_clock -period "50.000000 MHz" [get_por
create_clock -period "50.000000 MHz" [get_por
create_clock -period "300.000000 MHz" [get_p
create_clock -period "300.000000 MHz" [get_p
create_clock -period "275.000000 MHz" [get_p
create_clock -period "275.000000 MHz" [get_p
create_clock -period "275.000000 MHz" [get_p
create_clock -period "275.000000 MHz" [get_p
create_clock -period "644.531250 MHz" [get_p
create_clock -period "644.531250 MHz" [get_p
create_clock -period "644.531250 MHz" [get_p
create_clock -period "644.531250 MHz" [get_p
*****
```

Figure 5-5 SDC file created by System Builder

■ Using Si5340 control IP

Table 5-2 lists the instruction ports of Si5340 Controller IP.

Table 5-2 Si5340 Controller Instruction Ports

| Port | Direction | Description |
|------|-----------|----------------------|
| iCLK | input | System Clock (50MHz) |

| | | |
|----------------------|--------|--|
| iRST_n | input | Synchronous Reset (0: Module Reset, 1: Normal) |
| iStart | input | Start to Configure (positive edge trigger) |
| iPLL_OUTX_FREQ_SEL | input | Setting Si5340 Output Channel Frequency Value |
| oPLL_REG_CONFIG_DONE | output | Si5340 Configuration status (0: Configuration in Progress, 1: Configuration Complete) |
| I2C_DATA | inout | I2C Serial Data to/from Si5340 |
| I2C_CLK | output | I2C Serial Clock to Si5340 |

As shown in **Table 5-3** and **Table 5-4**, both two Si5340 control IPs have preset several output frequency parameters, if users want to change frequency, users can fill in the input port " iPLL_OUTX_FREQ_SEL" with a desired frequency value and recompile the project. For example, in Si5340A control IP, change

```
.iPLL_OUT1_FREQ_SEL(`SI5340A_125M),
```

to

```
.iPLL_OUT1_FREQ_SEL(`SI5340A_156M25),
```

Recompile project, the Si5340A OUT2 channel (for QSFP-C) output frequency will change from 125Mhz to 156.25Mhz.

Table 5-3 Si5340A Controller Reference Clock Frequency Setting

| iPLL_OUTX_FREQ_SEL MODE Setting | Si5340A Channel Clock Frequency(MHz) |
|------------------------------------|--------------------------------------|
| 3'b000 | Power Down |
| 3'b001 | 644.53125 |
| 3'b010 | 322.26 |
| 3'b011 | 312.25 |
| 3'b100 | 250 |
| 3'b101 | 156.25 |

| | |
|--------|-----|
| 3'b110 | 125 |
| 3'b111 | 100 |

Table 5-4 Si5340B Controller Reference Clock Frequency Setting

| iPLL_OUT_FRE Q_SEL MODE Setting | DDR4 Frequency(MHz) | QDRII Frequency(MHz) | PCIE Frequency(MHz) |
|---------------------------------------|------------------------|-------------------------|------------------------|
| 5'b00000 | 300 | 275 | 100 |
| 5'b00001 | 266 | 275 | 100 |
| 5'b00010 | 233 | 275 | 100 |
| 5'b00011 | 200 | 275 | 100 |
| 5'b00100 | 150 | 275 | 100 |
| 5'b00101 | 300 | 250 | 100 |
| 5'b00110 | 266 | 250 | 100 |
| 5'b00111 | 233 | 250 | 100 |
| 5'b01000 | 200 | 250 | 100 |
| 5'b01001 | 150 | 250 | 100 |
| 5'b01010 | 300 | 225 | 100 |
| 5'b01011 | 266 | 225 | 100 |
| 5'b01100 | 233 | 225 | 100 |
| 5'b01101 | 200 | 225 | 100 |
| 5'b01110 | 150 | 225 | 100 |

Users can also dynamically modify the input parameters, and input a positive edge trigger for “iStart”, then, Si5340 output frequency can be modified.

After the manually modifying, please remember to modify the corresponding frequency value in SDC file.

■ Modify Clock Parameter For Your Own Frequency

If the Si5340 control IP build-in frequencies are not users' desired, users can refer to

below steps to modify control IP register parameter settings to modify the IP to output a desired frequency.

1. Firstly, download ClockBuilder Pro Software (See **Figure 5-6**), which is provided by Silicon Labs. This tool can help users to set the Si5340's output frequency of each channel through the GUI interface, and it will automatically calculate the Register parameters required for each frequency. The tool download link:

http://url.terasic.com/clockbuilder_pro_software

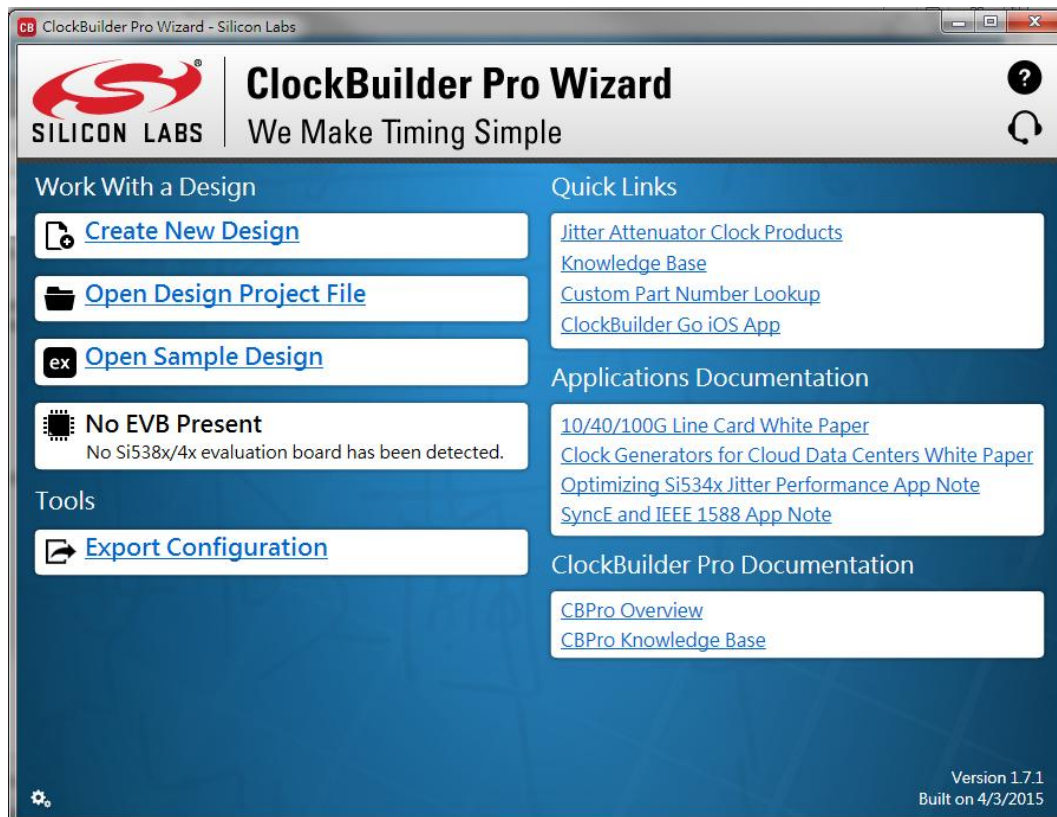


Figure 5-6 ClockBuilder Pro Wizard

2. After the installation, select Si5340, and configure the input frequency and output frequency as shown in **Figure 5-7**.

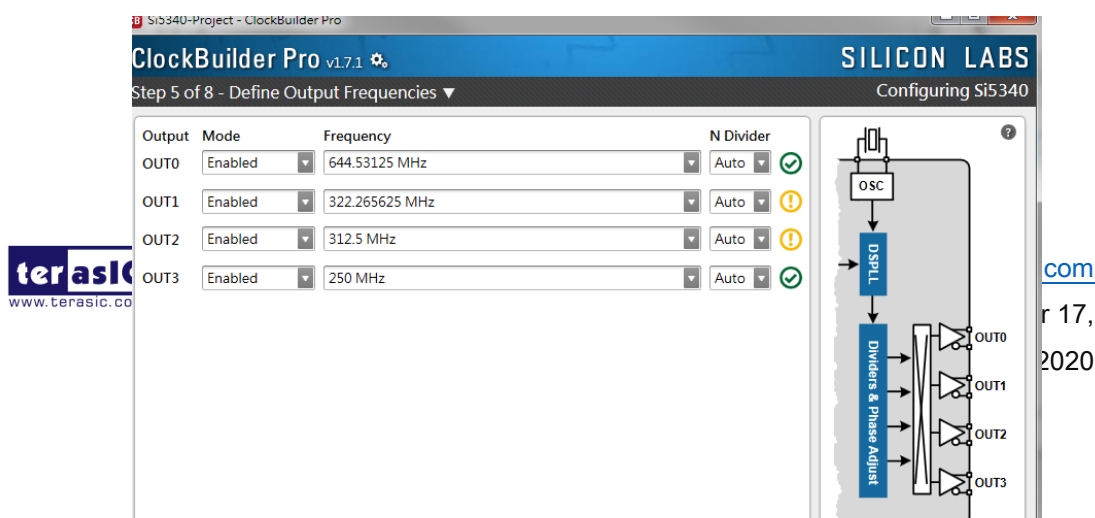


Figure 5-7 Define Output Clock Frequencies on ClockBuilder Pro Wizard

3. After the setting is completed, ClockBuilder Pro Wizard generates a Design Report(text), which contains users setting frequency corresponding register value (See **Figure 5-8**).

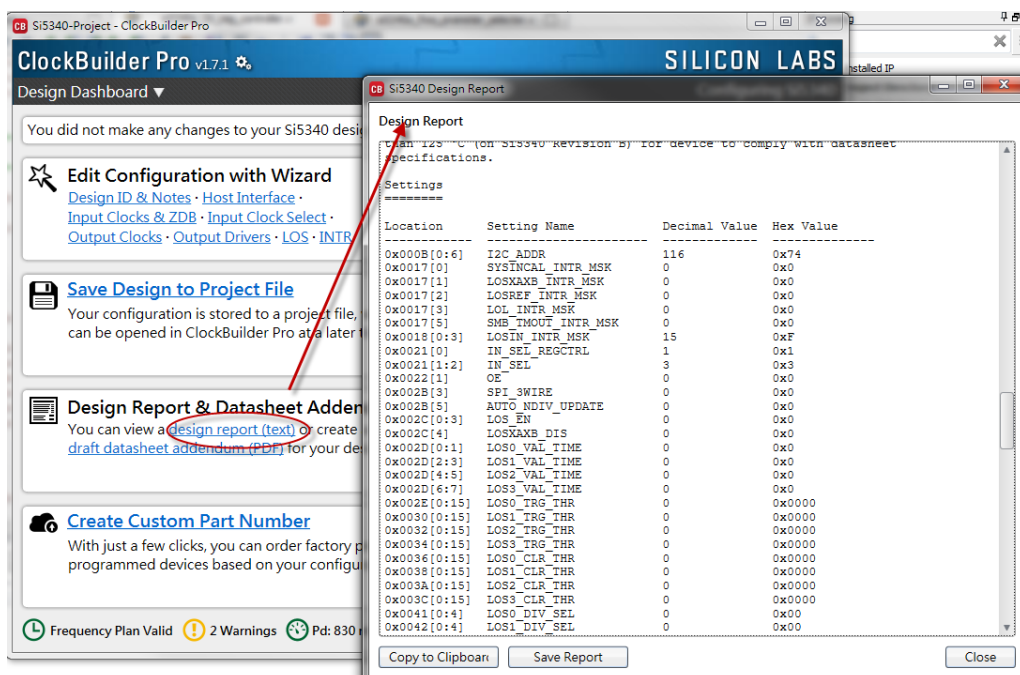


Figure 5-8 Open Design Report on ClockBuilder Pro Wizard

- Open Si5340 control IP sub-module "si5340a_i2c_reg_controller.v" as shown in **Figure 5-9**, refer Design Report parameter to modify sub-module corresponding register value (See **Figure 5-10**).

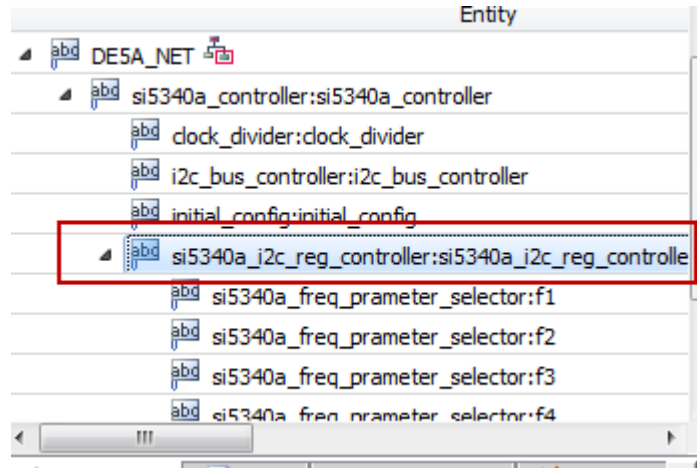


Figure 5-9 Sub-Module file "si5340a_i2c_reg_controller.v"

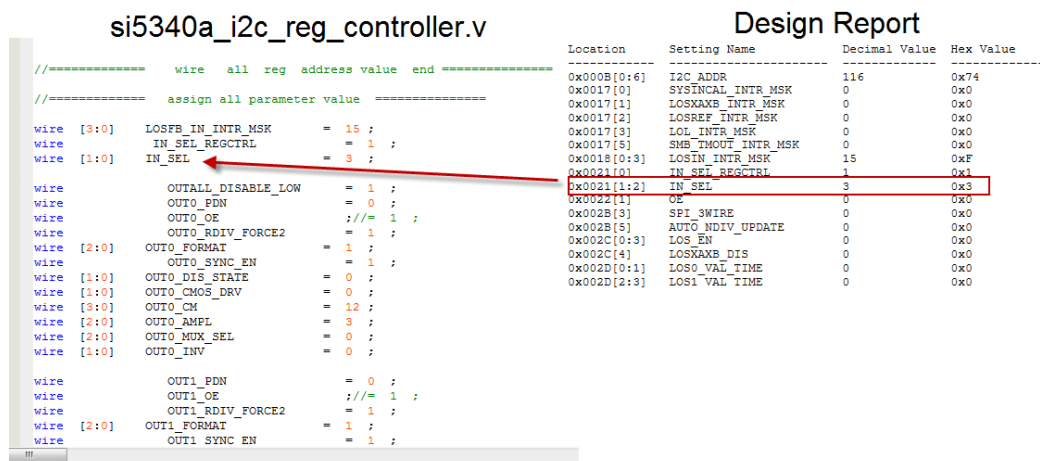


Figure 5-10 Modify Si5340 Control IP Base on Design Report

After modifying and compiling, Si5340 can output new frequencies according to the users' setting.

Note :

- No need to modify all Design Report parameters in si5340a_i2c_reg_controller.v/si5340b_i2c_reg_controller.v, users can ignore

parameters which have nothing to do with the frequency setting

2. After the manually modifying, please remember to modify clock constrain setting in .SDC file

5.3 Nios II control for

Si5340/Temperature/Power

This demonstration shows how to use the Nios II processor to program two programmable oscillators (Si5340A and Si5340B) on the FPGA board, how 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.

■ System Block Diagram

Figure 5-11 shows the system block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The four peripherals (including temperature sensor, Si5340A/B, and INA231) are all controlled by Nios II through the PIO controller, and all of them are programmed through I2C protocol which is implemented in the C code. The I2C pins from chip are connected to Qsys System Interconnect Fabric through PIO controllers. The Nios II program toggles the PIO controller to implement the I2C protocol. The Nios II program is running in the on-chip memory.

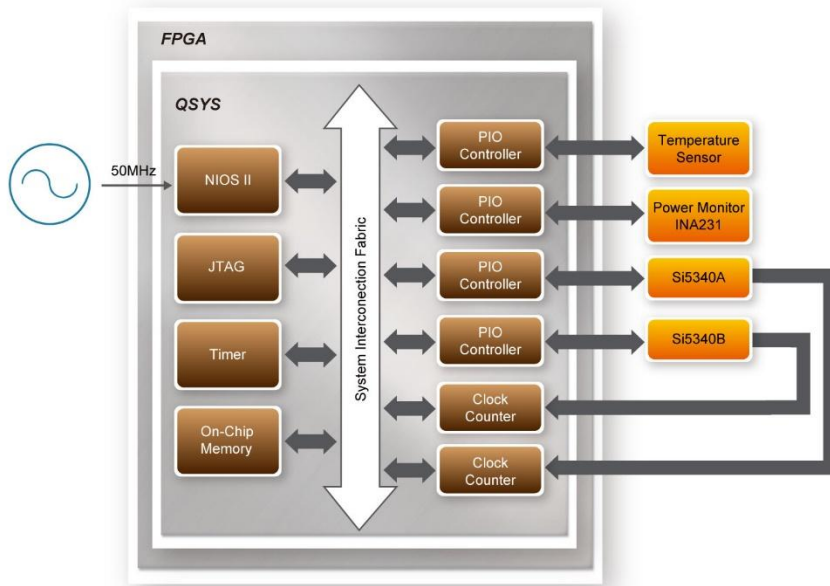


Figure 5-11 Block diagram of the Nios II Basic Demonstration

The program provides a menu in nios-terminal, as shown in **Figure 5-12** to provide an interactive interface. With the menu, users can perform the test for the temperatures sensor, external PLL and power monitor. Note, pressing 'ENTER' should be followed with the choice number.

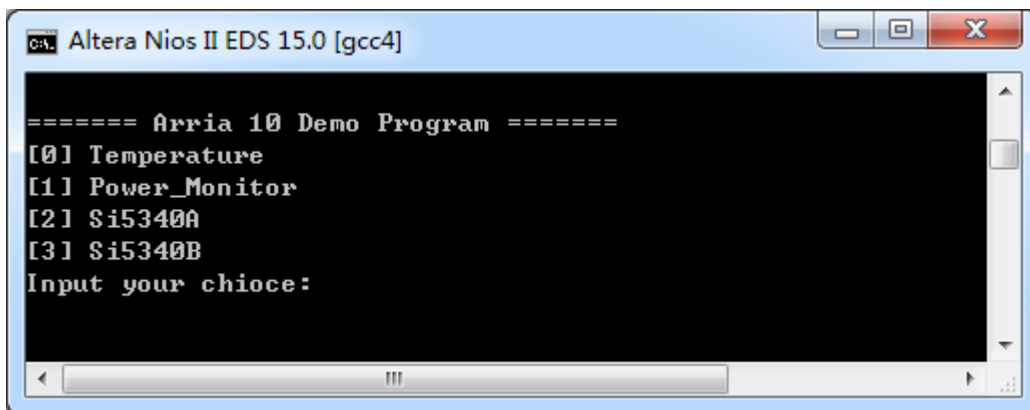


Figure 5-12 Menu of Demo Program

In temperature test, the program will display local temperature and remote temperature.

The remote temperature is the FPGA temperature, and the local temperature is the board temperature where the temperature sensor located.

A power monitor IC (INA231AIYFFT) embedded on the board can monitor Arria10 real-time current and power. This IC can work out current/power value as multiplier and divider are embedded in it. There is a shunt resistor R149 ($R_{SHUNT} = 0.003 \Omega$) for INA231AIYFFT in the circuit, when power on the DE5a-NET board, there will be a voltage drop (named Shut Voltage) on R149. Based on sense resistors, the program of power monitor can calculate the associated voltage, current and power consumption from the IN231 through the I2C interface. Please note the device I2C address is 0x80.

In the external PLL programming test, the program will program the PLL first, and subsequently will use TERASIC QSYS custom CLOCK_COUNTER IP to count the clock count in a specified period to check whether the output frequency is changed as configured. To avoid a Quartus Prime compilation error, dummy transceiver controllers are created to receive the clock from the external PLL. Users can ignore the functionality of the transceiver controller in the demonstration. For Si5340A/B programming, Please note the device I2C address are 0xEE. The program can control the Si5340A to configure the output frequency of QSFPA/B/C/D REFCLK, or control the Si5340B to configure the output frequency of DDR4/PCIE/QDRII REFCLK according to your choice.

■ Demonstration File Locations

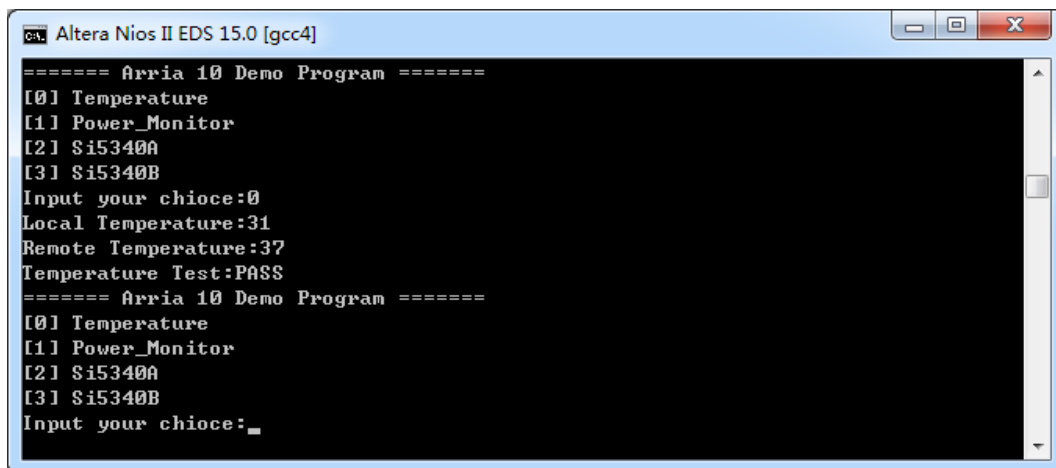
- Hardware project directory: NIOS_BASIC_DEMO
- Bitstream used: NIOS_BASIC_DEMO.sof
- Software project directory: NIOS_BASIC_DEMO \software
- Demo batch file : NIOS_BASIC_DEMO\demo_batch\NIOS_BASIC_DEMO.bat, NIOS_BASIC_DEMO.sh

■ Demonstration Setup and Instructions

- Make sure Quartus Prime and Nios II are installed on your PC.
- Power on the FPGA board.
- Use the USB Cable to connect your PC and the FPGA board and install USB Blaster

II driver if necessary.

- Execute the demo batch file “NIOS_BASIC_DEMO.bat” under the batch file folder, NIOS_BASIC_DEMO\demo_batch.
- After the Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- For temperature test, please input key ‘0’ and press ‘Enter’ in the nios-terminal, , as shown in **Figure 5-13**.
- For power monitor test, please input key ‘1’ and press ‘Enter’ in the nios-terminal, the Nios II console will display the current values of voltage, current and power as shown in **Figure 5-14**.
- For programmable PLL Si5340A test, please input key ‘2’ and press ‘Enter’ in the nios-terminal first, then select the desired output frequency of QSFPA/B/C/C REFCLK, as shown in **Figure 5-15**.
- For programmable PLL Si5340B test, please input key ‘3’ and press ‘Enter’ in the nios-terminal first, then select the desired output frequency of DDR4/PCIE/QDRII REFCLK, as shown in **Figure 5-16**.



```
ca: Altera Nios II EDS 15.0 [gcc4]
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your choice:0
Local Temperature:31
Remote Temperature:37
Temperature Test:PASS
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your choice:_
```

Figure 5-13 Temperature Demo

```

c:\. Altera Nios II EDS 15.0 [gcc4]
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your chioce:1
Configuration ok!
==== Power Monitor Test ====
Shunt_Voltage = 4.807 mV
Bus_Voltage   = 11.494 V
Current       = 1.614 A
Power         = 18.462 W
Power_Monitor Test:PASS
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your chioce:

```

Figure 5-14 power monitor Demo

```

c:\. Altera Nios II EDS 15.0 [gcc4]
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your chioce:2
===== Si5340A Programming =====
[0] 100.000000 MHz
[1] 125.000000 MHz
[2] 156.250000 MHz
[3] 250.000000 MHz
[4] 312.500000 MHz
[5] 322.265625 MHz
[6] 644.531250 MHz
[7] 0.000000 MHz
[Other] exit
please select QSFPA_REFCLK:0
please select QSFPA_REFCLK:1
please select QSFPC_REFCLK:2
please select QSFPA_REFCLK:3
[SI5430] SI5340A_Config_Init success
[SI5430] SI5340A_Configuration success
QSFPA/100.000000MHz ref clock test PASS <clk1=998, clk2=1996, expected clk2=1996
>
QSFPA/125.000000MHz ref clock test PASS <clk1=998, clk2=2495, expected clk2=2495
>
QSFPA/156.250000MHz ref clock test PASS <clk1=998, clk2=3119, expected clk2=3118
>
QSFPA/250.000000MHz ref clock test PASS <clk1=998, clk2=4990, expected clk2=4990
>
Si5340A Test:PASS
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] Si5340A
[3] Si5340B
Input your chioce:

```

Figure 5-15 Si5340A Demo

```

c:\> Altera Nios II EDS 16.1 [gcc4]

===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] $i5340A
[3] $i5340B
Input your choice:3
===== $i5340B Programming =====
[0] DDR4:300.000000 MHz PCIE:100.000000 MHz QDR11:275.000000 MHz
[1] DDR4:266.666992 MHz PCIE:100.000000 MHz QDR11:275.000000 MHz
[2] DDR4:233.332993 MHz PCIE:100.000000 MHz QDR11:275.000000 MHz
[3] DDR4:200.000000 MHz PCIE:100.000000 MHz QDR11:275.000000 MHz
[4] DDR4:150.000000 MHz PCIE:100.000000 MHz QDR11:275.000000 MHz
[5] DDR4:300.000000 MHz PCIE:100.000000 MHz QDR11:250.000000 MHz
[6] DDR4:266.666992 MHz PCIE:100.000000 MHz QDR11:250.000000 MHz
[7] DDR4:233.332993 MHz PCIE:100.000000 MHz QDR11:250.000000 MHz
[8] DDR4:200.000000 MHz PCIE:100.000000 MHz QDR11:250.000000 MHz
[9] DDR4:150.000000 MHz PCIE:100.000000 MHz QDR11:250.000000 MHz
[10] DDR4:300.000000 MHz PCIE:100.000000 MHz QDR11:225.000000 MHz
[11] DDR4:266.666992 MHz PCIE:100.000000 MHz QDR11:225.000000 MHz
[12] DDR4:233.332993 MHz PCIE:100.000000 MHz QDR11:225.000000 MHz
[13] DDR4:200.000000 MHz PCIE:100.000000 MHz QDR11:225.000000 MHz
[14] DDR4:150.000000 MHz PCIE:100.000000 MHz QDR11:225.000000 MHz
[15] DDR4:300.000000 MHz PCIE:100.000000 MHz QDR11:233.332993 MHz
[16] DDR4:266.666992 MHz PCIE:100.000000 MHz QDR11:233.332993 MHz
[17] DDR4:233.332993 MHz PCIE:100.000000 MHz QDR11:233.332993 MHz
[18] DDR4:200.000000 MHz PCIE:100.000000 MHz QDR11:233.332993 MHz
[19] DDR4:150.000000 MHz PCIE:100.000000 MHz QDR11:233.332993 MHz
[Other] exit
please select $i5340B CASE:0
[$i5340] $i5340B_Config_Init success
[$i5340] $i5340B_Configuration success
DDR4/300.000000MHz ref clock test PASS <clk1=998, clk2=5988, expected clk2=5988>
PCIE/100.000000MHz ref clock test PASS <clk1=998, clk2=1996, expected clk2=1996>
QDR11/275.000000MHz ref clock test PASS <clk1=998, clk2=5489, expected clk2=5489>
$i5340B Test:PASS
===== Arria 10 Demo Program =====
[0] Temperature
[1] Power_Monitor
[2] $i5340A
[3] $i5340B
Input your choice:

```

Figure 5-16 Si5340B Demo

5.4 Fan Speed Control

This demo helps users quickly understand how to set the MAX6650 chip from the FPGA to control the fansink. The MAX6650 chip can set or retrieve the RPM of the fansink. It can also monitor if there is any unexpected error and determine which type of error it is. The following section will save lots of time for the development of user application.

■ System Block Diagram

Figure 5-17 shows the system block diagram of this demo. It is necessary to configure the MAX6650 chip prior upon the initialization of fansink control. The MAX6650 chip uses standard I2C protocol for communication. The functions I2C_Config and

I2C_Bus_Controller set and monitor the RPM of the fansink, respectively. A pre-scaler is used as frequency divider for the clock frequency of I2C. Users need to calculate the frequency based on the equations from the datasheet to control the RPM of the fansink. There are three equations in the datasheet and this demo uses one of them. For other equations, please refer to the datasheet MAX6650-MAX6651.pdf in the system CD.

The Switch[0] controls the RPM in this demo. When the Switch[0] is set to 0, the speed is around 2000 RPM. The speed would reach about 5000 RPM if the Switch[0] is set to 1. It would take 10 ~ 30 secs as the buffer time for the conversion. If an error is detected, the LED would lit. Users need to press KEY[1] to reset the LED by turn it off.

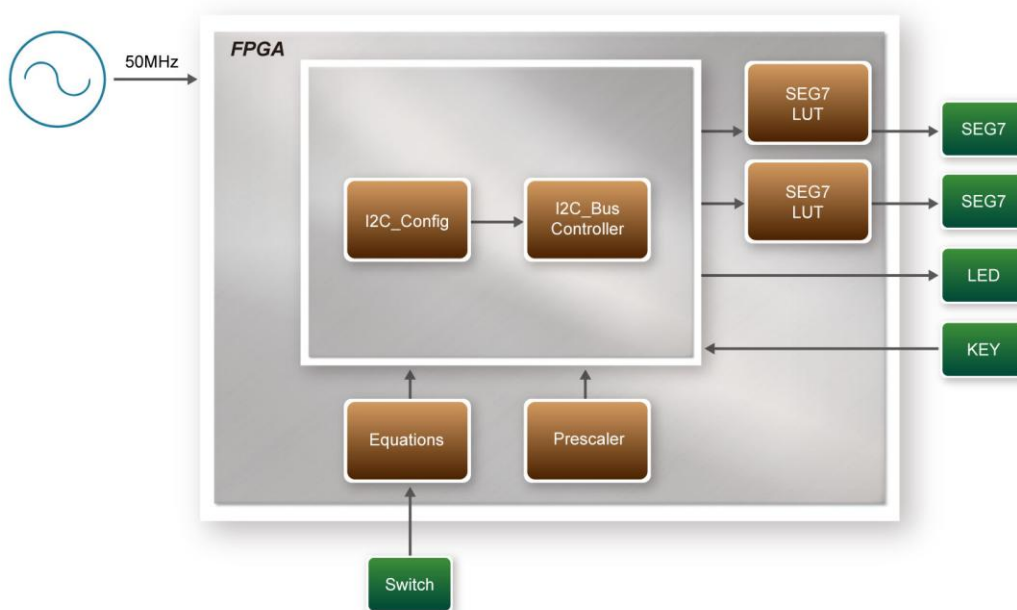


Figure 5-17 Block diagram of the fan speed control demonstration

■ Alarm Status Register Bit Assignments

When the fan is abnormal, the LED will lit. Users can refer to **Table 5-5** and get a better understanding about the malfunction of the fansink accordingly. The status of BIT 4 ~ 7 can be ignored because BIT 4 is for MAX6651 only and BIT 5 ~ 7 are always low.

Table 5-5 Alarm-Enable Register Bit Masks

| BIT | NAME | POR(DEFAULT) STATE | FUNCTION |
|-------------|-------------------------|-----------------------|---|
| 7(MSB) to 5 | --- | 0 | Always 0 |
| 4 | GPIO2 (MAX6651 only) | 0 | GPIO2 Alarm. Set when GPIO2 is low (MAX6651 only) |
| 3(LED[3]) | GPIO1 | 0 | GPIO1 Alarm. Set when GPIO1 is low |
| 2(LED[2]) | TACH | 0 | Tachometer Overflow Alarm |
| 1(LED[1]) | MIN | 0 | Minimum Output Level Alarm |
| 0(LED[0]) | MAX | 0 | Maximum Output Level Alarm |

■ Design Tools

- Quartus Prime 16.1.2 Standard Edition
- Demonstration Source Code
- Project Directory: Demonstration\Fan
- Bit Stream: DE5a_NET.sof
- Demonstration Batch File

Demo Batch File Folder: *\Fan\demo_batch*

The demo batch file includes following files:

- Batch File: test_ub2.bat
- FPGA Configure File: DE5A_NET.sof

■ Demonstration Setup

- Make sure Quartus Prime is installed on the host PC.
- Connect the DE5a-Net and the host PC via USB cable. Install the USB-Blaster II driver if necessary.
- Power on the FPGA Board.
- Execute the demo batch file “test_ub2.bat” under the batch file folder \Fan\demo_batch.
- When SW[0] is set to 0, the RPM would slowly be adjusted to ~2000. When SW[0] is set to 1, the RPM would slowly be adjusted to ~5000.

Memory Reference Design

The FPGA development board includes two kinds of high-speed memories:

- DDR4 SDRAM: two independent banks, update to 1200 MHz
- QDRII+ SRAM: four independent banks, update to 550 MHz

This chapter will show three examples which use the Altera Memory IP to perform memory test functions. The source codes of these examples are all available on the FPGA System CD. These three examples are:

- QDRII+ SRAM Test: Full test of the four banks of QDRII+ SRAM
- DDR4 SDRAM Test: Random test of the two banks of DDR4 SDRAM.
- DDR4 SDRAM Test by Nios II: Full test of one bank of DDR4 SDRAM with Nios II

Note. Quartus 16.1.2 64-bit or later is strongly recommended for compiling these projects.

6.1 QDRII+ SRAM Test

QDR II/QDR II+ SRAM devices enable you to maximize memory bandwidth with separate read and write ports. The memory architecture features separate read and write ports operating twice per clock cycle to deliver a total of four data transfers per cycle. The resulting performance increase is particularly valuable in bandwidth-intensive and low-latency applications.

This demonstration utilizes four QDRII+ SRAMs on the FPGA board. It describes how to use Altera’s “Arria 10 External Memory Interfaces” (Arria 10 EMIF) IP to implement a memory test function.

■ **Function Block Diagram**

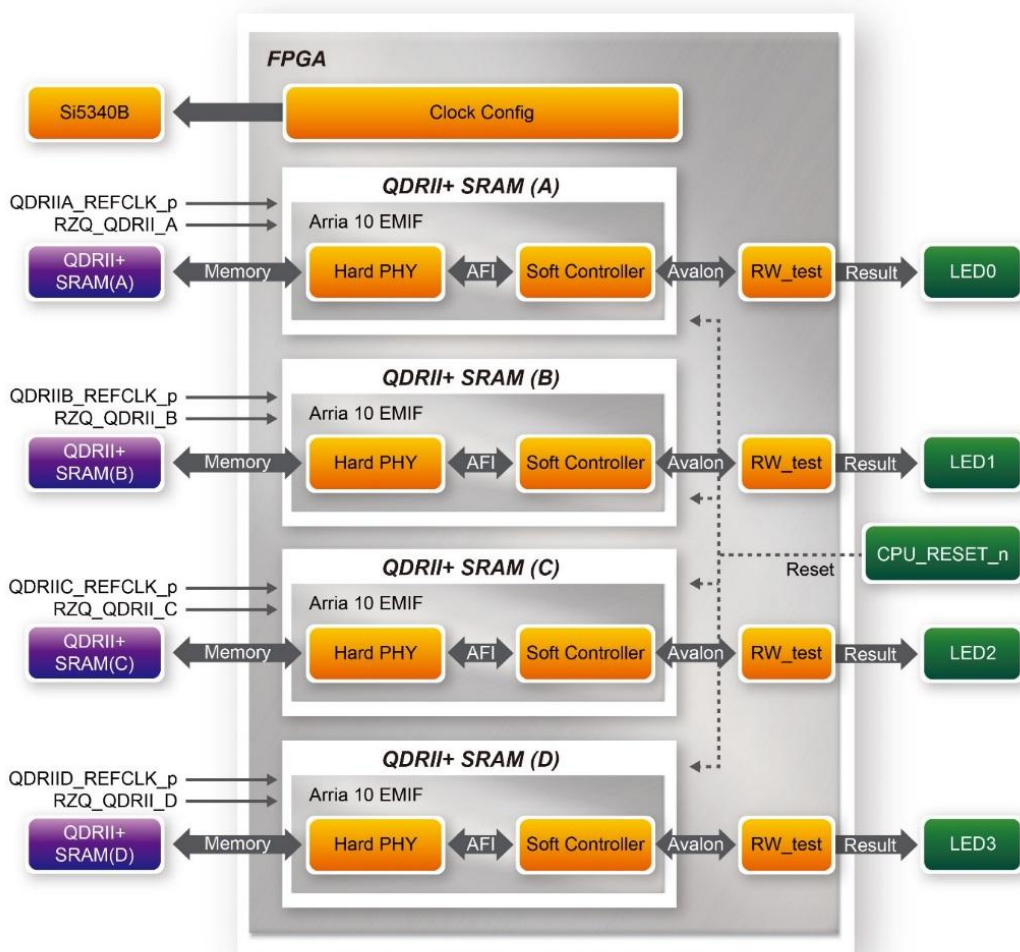


Figure 6-1 Function Block Diagram of the QDRII+ SRAM x4 Demonstration

Figure 6-1 shows the function block diagram of the demonstration. The four QDRII+ SRAM controllers are configured as a 72Mb controller. The QDRII+ SRAM IP generates a

550MHz clock as memory clock and a half-rate system clock, 275MHz, for the controllers.

The QDRIIA/B/C/D_REFCLK is generated from Si5340B which configured 275MHz for QDRII+ 550MHz by Clock Config module. QDRIIA/B/C/D_REFCLK has no default frequency output so that they must be configured first.

In this demonstration, each QDRII+ SRAM has its own PLL, DLL and OCT resources. The Arria 10 EMIF QDRII IP uses a Hard PHY and a soft Controller. The Hard PHY capable of performing key memory interface functionality such as read/write leveling, FIFO buffering to lower latency and improve margin, timing calibration, and on-chip termination.

The Avalon bus read/write test (RW_test) modules read and write the entire memory space of each QDRII+ SRAM through the Avalon interface of each controller. In this project, the RW_test module will first write the entire memory and then compare the read back data with the regenerated data (the same sequence as the write data). Test control signals for four QDRII+ SRAMs will generate from CPU_RESET_n and four LEDs will indicate the test results of four QDRII+ SRAMs.

■ Altera QDRII and QDRII+ SRAM Controller with UniPHY

To use Altera QDRII+ SRAM controller, users need to perform the following steps in order:

1. Create correct pin assignments for QDRII+.
2. Setup correct parameters in QDRII+ SRAM controller dialog.

■ Design Tools

- Quartus Prime 16.1.2 Standard Edition
- Demonstration Source Code
- Project directory: QDRII_x4_Test_550MHz
- Bit stream used: DE5A_NET_DDR4.sof
- Demonstration Batch File

Demo Batch File Folder: QDRII_x4_Test_550MHz\demo_batch

The demo batch files include the followings:

- Batch file for USB-Blaster II: test.bat,
- FPGA configuration file: DE5A_NET_DDR4.sof
- Demonstration Setup
- Make sure Quartus Prime is installed on your PC.
- Connect the USB cable to the FPGA board and host PC. Install the USB-Blaster II driver if necessary.
- Power on the FPGA Board.
- Execute the demo batch file “test.bat” under the batch file folder, QDRII_x4_Test_550MHz\demo_batch.
- Press CPU_RESET_n of the FPGA board to start the verification process. When CPU_RESET_n is held down, all the LEDs will be turned off. All LEDs should turn back on to indicate test passes upon the release of CPU_RESET_n.
- If any LED is not lit up after releasing CPU_RESET_n, it indicates the corresponding QDRII+ SRAM test has failed. **Table 6-1** lists the matchup for the four LEDs.
- Press CPU_RESET_n again to regenerate the test control signals for a repeat test.

Table 6-1 LED Indicators

| NAME | Description |
|------|----------------------------|
| LED0 | QDRII+ SRAM(A) test result |
| LED1 | QDRII+ SRAM(B) test result |
| LED2 | QDRII+ SRAM(C) test result |
| LED3 | QDRII+ SRAM(D) test result |

6.2 DDR4 SDRAM Test

This demonstration performs a memory test function on the two DDR4-SDRAM SO-DIMM on the DE5a-Net. The memory size of each DDR4 SDRAM SO-DIMM used in this test is 4 GB.

■ Function Block Diagram

Figure 6-2 shows the function block diagram of this demonstration. There are two DDR4 SDRAM controllers. The controller uses 300.000 MHz as a reference clock. It generates one 1066MHz clock as memory clock from the FPGA to the memory and the controller itself runs at quarter-rate in the FPGA i.e. 300.000 MHz.

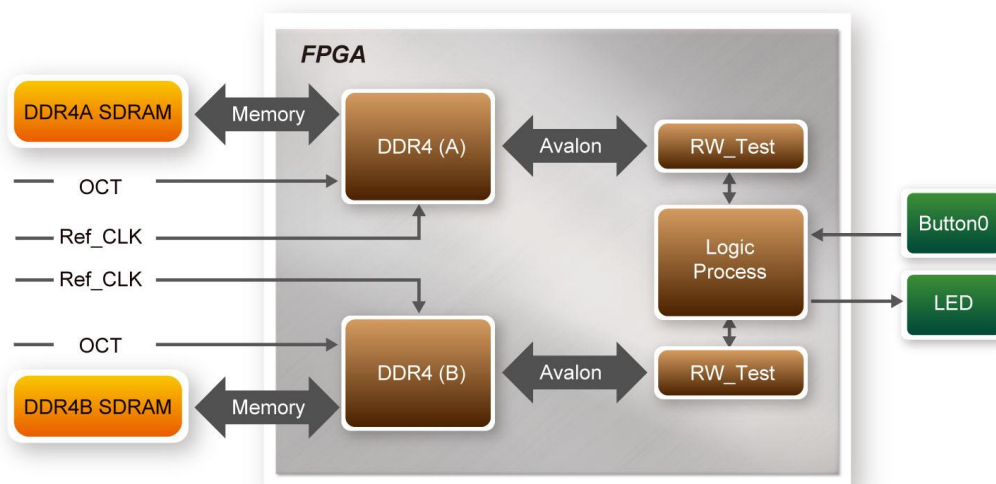


Figure 6-2 Block diagram of DDR4 SDRAM (4G) x2 demonstration

■ Altera DDR4 SDRAM Controller

To use Altera DDR4 controller, please perform the three major steps below:

1. Create correct pin assignments for DDR4.
2. Setup correct parameters in the dialog of DDR4 controller.

■ Design Tools

- Quartus Prime 16.1.2 Standard Edition
- Demonstration Source Code
- Project Directory: Demonstration\RTL_DDR4_4GB_x2
- Bit Stream: DE5A_NET_DDR4.sof
- Demonstration Batch File

Demo Batch File Folder: RTL_DDR4_4GB_x2 \demo_batch

The demo batch file includes following files:

- Batch File: test.bat
- FPGA Configuration File: DE5A_NET_DDR4.sof

■ Demonstration Setup

- Make sure Quartus Prime is installed on the host PC.
- Connect DE5a-Net board to the host PC via USB cable. Install the USB-Blaster II driver if necessary.
- Power on the DE5a-Net board.
- Execute the demo batch file “test.bat” under the batch file folder \RTL_DDR4_4GB_x2\demo_batch.
- Press BUTTON0 on DE5a-Net to start the verification process. When BUTTON0 is pressed, LED1, LED2, and LED3 should start blinking. After approximately 2 seconds, LED1 and LED2 should stop blinking and stay on to indicate the DDR4 (A) and DDR4 (B) have passed the test, respectively. **Table 6-2** lists the LED indicators.

- If LED3 is not blinking, it means the 50MHz clock source is not working.
- If LED1 or LED2 does not start blinking upon releasing BUTTON0, it indicates local_cal_success of the corresponding DDR4 fails.
- If LED1 or LED2 fail to remain on after 2 seconds, the corresponding DDR4 test has failed.
- Press BUTTON0 again to regenerate the test control signals for a repeat test.

Table 6-2 LED Indicators

| NAME | Description |
|------|----------------------|
| LED0 | Reset |
| LED1 | DDR4 (A) test result |
| LED2 | DDR4 (B) test result |
| LED3 | 50MHz clock source |

6.3 DDR4 SDRAM Test by Nios II

Many applications use a high performance RAM, such as a DDR4 SDRAM, to provide temporary storage. In this demonstration hardware and software designs are provided to illustrate how to perform DDR4 memory access in QSYS. We describe how the Altera’s “Arria 10 External Memory Interfaces” IP is used to access the two DDR4-Sodimm on the FPGA board, and how the Nios II processor is used to read and write the SDRAM for hardware verification. The DDR4 SDRAM controller handles the complex aspects of using DDR4 SDRAM by initializing the memory devices, managing SDRAM banks, and keeping the devices refreshed at appropriate intervals.

■ System Block Diagram

Figure 6-3 shows the system block diagram of this demonstration. The QSYS system requires one 50 MHz and two 300MHz clock source. The two 300 MHz clock source is

provided by SI5340B clock generator on the board. Si5340B Config Controller is used to configure the SI5340B to generate the required clock. The 50MHz is used by IO PLL to generate 200MHz for NIOS Processor and On-chip Memory. The two 300MHz clock are used as reference clocks for the DDR4 controllers. There are two DDR4 Controllers are used in the demonstrations. Each controller is responsible for one DDR4-Sodimm. Each DDR4 controller is configured as a 1 GB DDR4-1200Mhz controller. The controllers are designed as 1GB rather 4GB is due to address space limitation of NIOS II processor. Nios II processor is used to perform memory test. The Nios II program is running in the On-Chip Memory. A PIO Controller is used to monitor buttons status which is used to trigger starting memory testing.

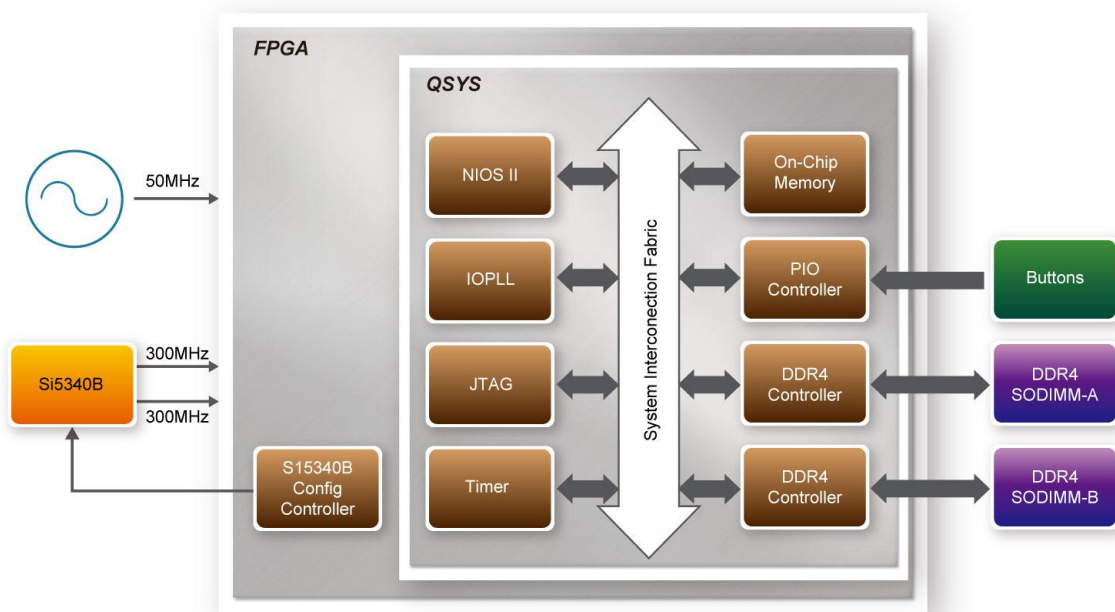


Figure 6-3 Block diagram of the DDR4 Basic Demonstration

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the whole 1 GB of SDRAM. Then, it calls Nios II system function (alt_dache_flush_all) to make sure all data has been written to SDRAM. Finally, it reads data from SDRAM for data verification. The program will show progress in JTAG-Terminal when writing/reading data to/from the SDRAM. When verification process is completed, the result is displayed in the JTAG-Terminal.

■ Design Tools

- Prime 16.1.2 Standard Edition
- Nios II Software Build Tools for Eclipse 16.1

■ Demonstration Source Code

- Quartus Project directory: NIOS_DDR4_X2
- Nios II Eclipse: NIOS_DDR4_X2\software
- Nios II Project Compilation

■ Nios II Project Compilation

Before you attempt to compile the reference design under Nios II Eclipse, make sure the project is cleaned first by clicking 'Clean' from the 'Project' menu of Nios II Eclipse.

■ Demonstration Batch File

Demo Batch File Folder: NIOS_DDR4_X2\demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaseter II: test.bat, test.sh
- FPGA Configure File: DE5A_NET_DDR4.sof
- Nios II Program: TEST_MEM.elf

■ Demonstration Setup

Please follow below procedures to setup the demonstration.

- Make sure Quartus Prime and Nios II are installed on your PC.
- Make sure both DDR4 SODIMMs are installed on the FPGA board.
- Power on the FPGA board.
- Use USB Cable to connect PC and the FPGA board and install USB Blaster II driver if necessary.
- Execute the demo batch file “test.bat” under the folder “NIOS_DDR4_X2\demo_batch”.
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- Press Button3~Button0 of the FPGA board to start SDRAM verify process. Press Button0 for continued test.
- The program will display progressing and result information, as shown in **Figure 6-4**.

```

C:\> Altera Nios II EDS 16.1 [gcc4]
Info <209007>: Configuration succeeded -- 1 device(s) configured
Info <209011>: Successfully performed operation(s)
Info <209061>: Ended Programmer operation at Tue May 16 14:18:00 2017
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 1482 megabytes
Info: Processing ended: Tue May 16 14:18:00 2017
Info: Elapsed time: 00:00:26
Info: Total CPU time (on all processors): 00:00:13
Using cable "DE5 [USB-1]", device 1, instance 0x00
Resetting and pausing target processor: OK
Initializing CPU cache <if present>
OK
Downloaded 77KB in 0.0s
Verified OK
Starting processor at address 0x81020244
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "DE5 [USB-1]", device 1, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

==== DDR4 Test! Size=A:1024MB/B:1024MB====

=====
Press any BUTTON on the board to start test [BUTTON0 for continued test]
====> DDR4 Testing, Iteration: 1
A/B Calibration Duration:0.389 seconds, status=5h
== DDR4-A Testing...
write...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
read/verify...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
DDR4 test:Pass, 94 seconds
== DDR4-B Testing...
write...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
read/verify...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
DDR4 test:Pass, 94 seconds

=====
Press any BUTTON on the board to start test [BUTTON0 for continued test]

```

Figure 6-4 Progress and Result Information for the DDR4 Demonstration

PCI Express 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. Arria 10 Hard IP for PCI Express with Avalon-MM DMA IP is used in this demonstration. For detail about this IP, please refer to Altera document [ug_a10_pcie_avmm_dma.pdf](#).

7.1 PCI Express System Infrastructure

Figure 7-1 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 Arria 10 Hard IP for PCI Express with Avalon-MM DMA. The application software on the PC side is developed by Terasic based on Altera's PCIe kernel mode driver.

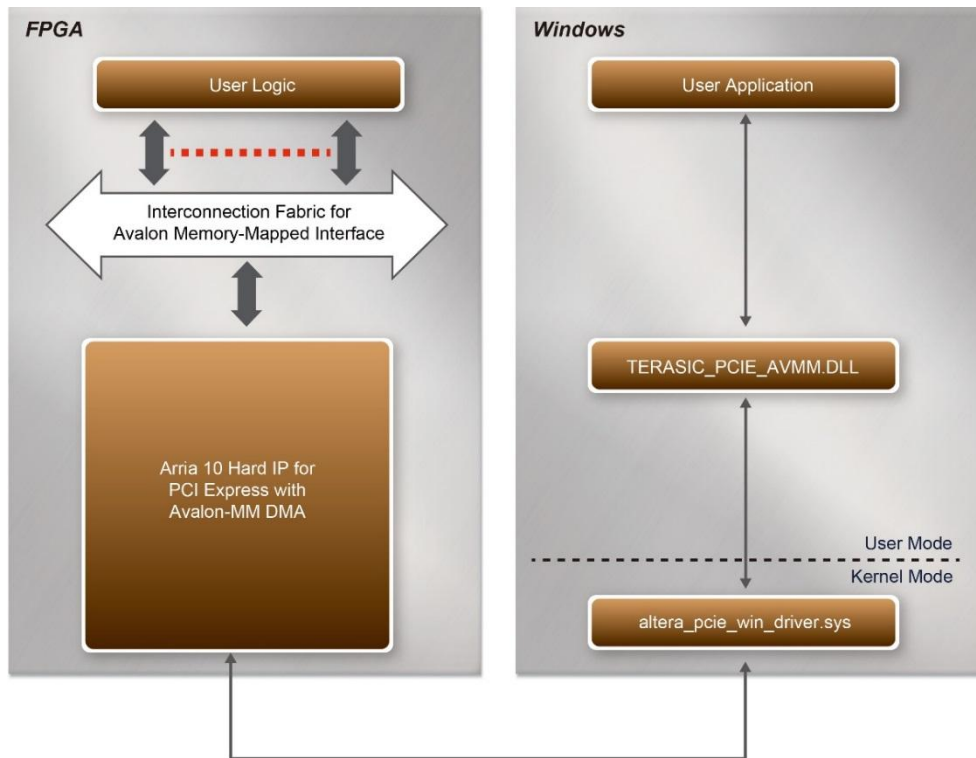


Figure 7-1 Infrastructure of PCI Express System

7.2 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 7 or Window XP. The SDK is located in the "CDROM\Demonstrations\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 0xE003. 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_AVMM.DLL.

This file is a 64-bit DLL. With the DLL is exported to the 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, Altera AVMM DMA is required as the read and write operations are specified under the hardware design on the FPGA.

7.3 PCI Express Software Stack

Figure 7-2 shows the software stack for the PCI Express application software on 64-bit Windows. The PCIe library module TERASIC_PCIE_AVMM.dll provides DMA and direct I/O access for 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.

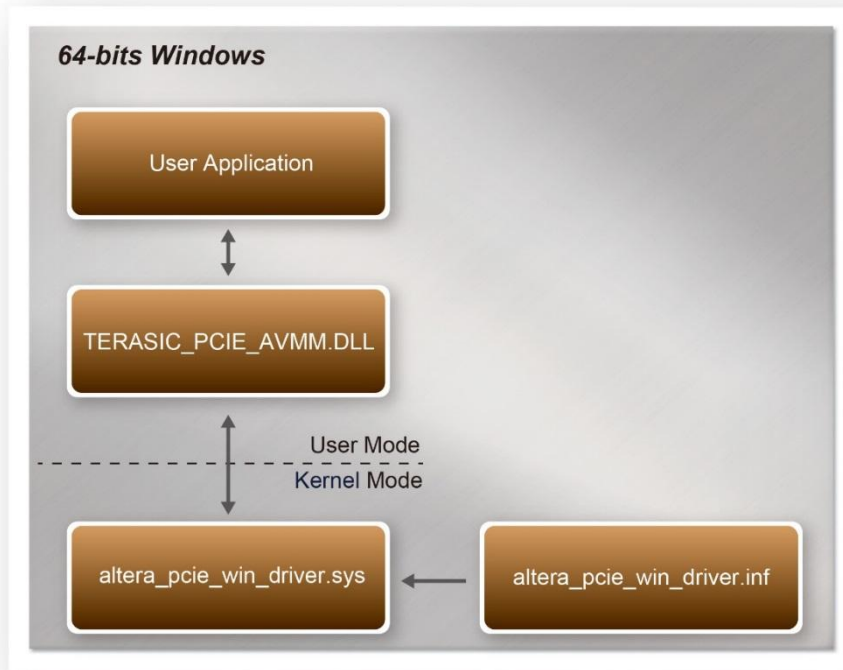


Figure 7-2 PCI Express Software Stack

■ Install PCI Express Driver on Windows

The PCIe driver is located in the folder:

"CDROM\Demonstrations\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 DE5a-Net on the PCIe slot of the host PC
2. Make sure Altera Programmer and USB-Blaster II driver are installed

3. Execute test.bat in "CDROM\Demonstrations\PCIe_Fundamental\demo_batch" to configure the FPGA
4. Restart windows operation system
5. Click Control Panel menu from Windows Start menu. Click 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 7-3**. Move the mouse cursor to the PCI Device item and right click it to select the Update Driver Software... item.

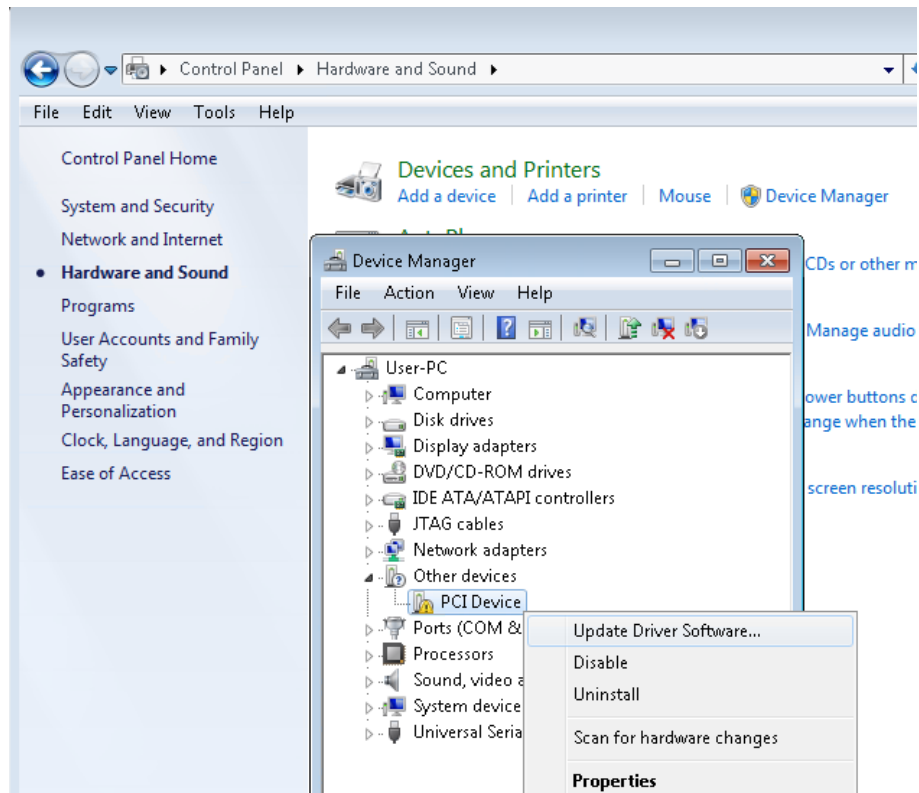


Figure 7-3 Screenshot of launching Update Driver Software... dialog

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

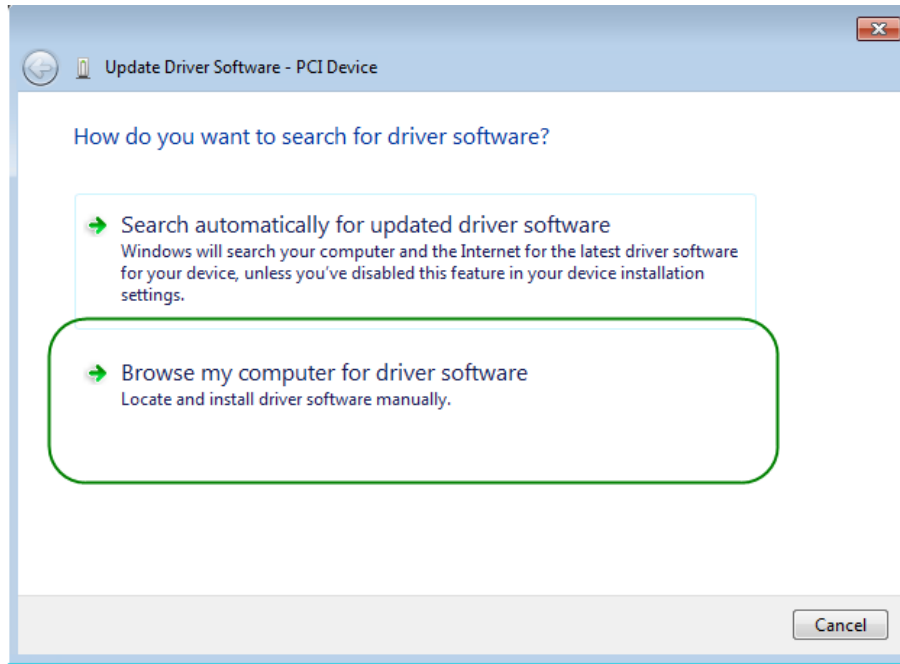


Figure 7-4 Dialog of Browse my computer for driver software

7. In the **Browse for driver software on your computer** dialog, click the **Browse** button to specify the folder where altera_pcie_win_driver.inf is located, as shown in **Figure 7-5**. Click the **Next** button.

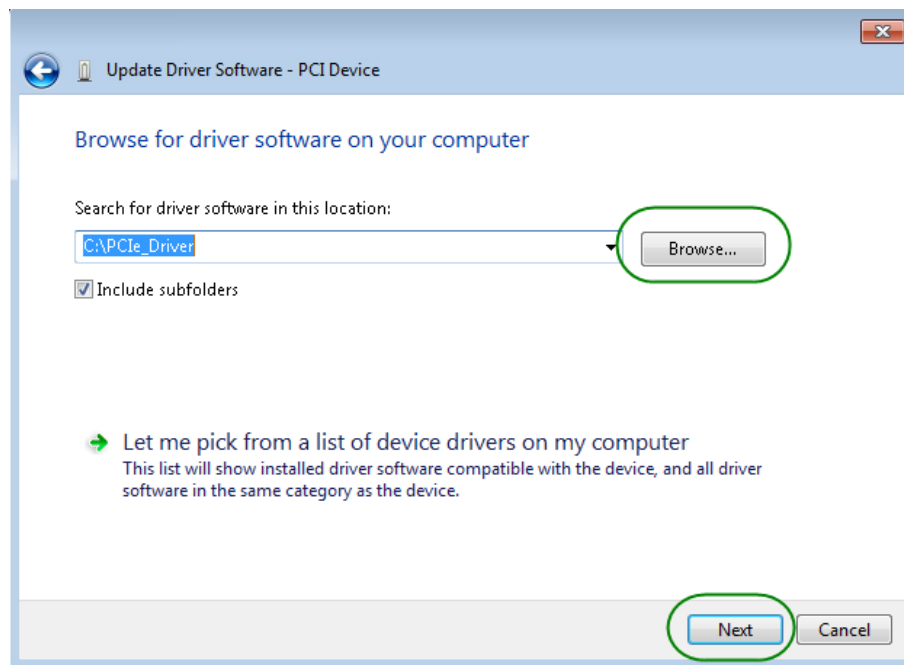


Figure 7-5 Browse for driver software on your computer

8. When the **Windows Security** dialog appears, as shown **Figure 7-6**, click the

Install button.

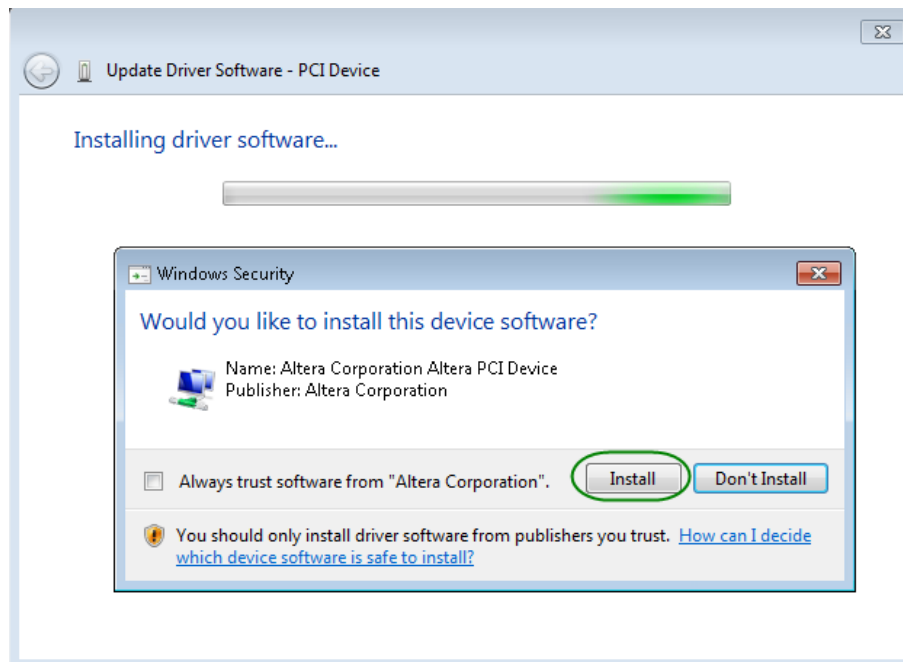


Figure 7-6 Click Install in the dialog of Windows Security

9. When the driver is installed successfully, the successfully dialog will appear, as shown in **Figure 7-7**. Click the **Close** button.

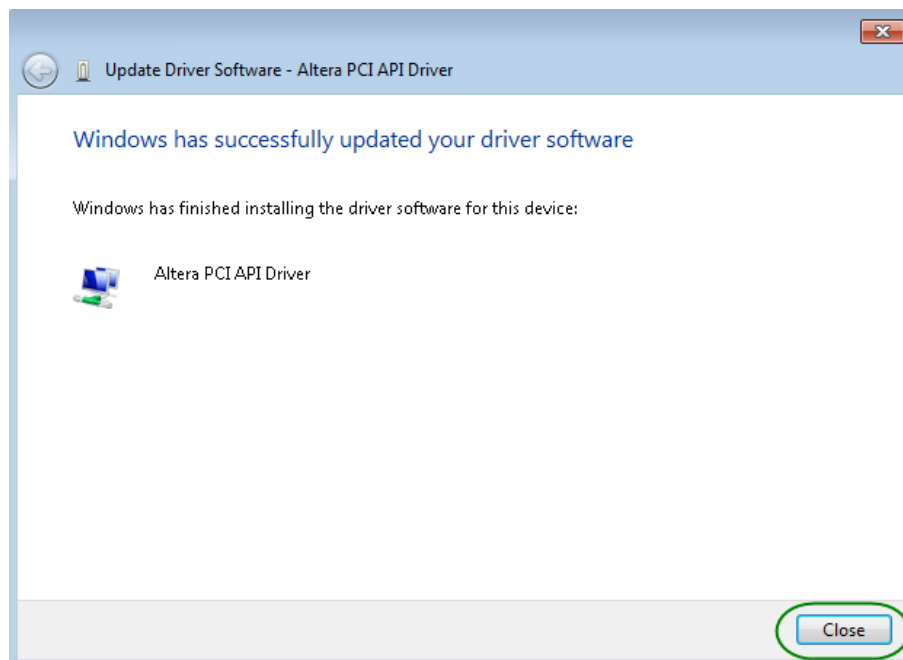


Figure 7-7 Click Close when the installation of Altera PCI API Driver is complete

10. Once the driver is successfully installed, users can see the **Altera PCI API Driver** under the device manager window, as shown in **Figure7-8**.

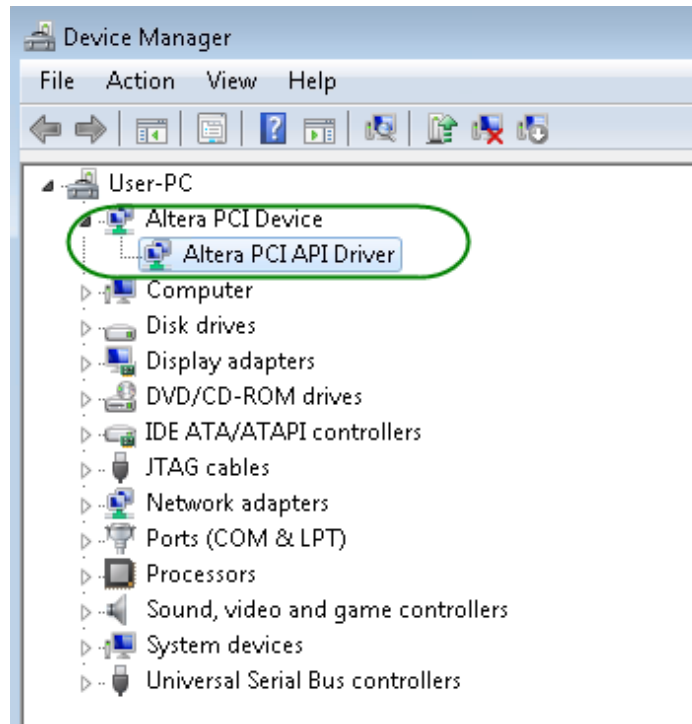


Figure 7-8 Altera PCI API Driver in Device Manager

■ 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_AVMM.h`
- `TERASIC_PCIE_AVMM.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_AVMM.h` in the C/C++ project.
3. Copy `TERASIC_PCIE_AVMM.dll` to the folder where the `project.exe` is located.

4. Dynamically load Terasic_PCIE_AVMM.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_AVMM.dll API. The details of API are described below:

7.4 PCI Express Library API

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

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

■ PCIE_Open

| |
|--|
| <p>Function:</p> <p>Open a specified PCIe card with vendor ID, device ID, and matched card number.</p> |
| <p>Prototype:</p> <pre>PCIE_HANDLE PCIE_Open(uint16_t wVendorID, uint16_t wDeviceID, uint16_t wCardNum);</pre> |
| <p>Parameters:</p> <p>wVendorID: Specify the desired vendor ID. A zero value means to ignore the vendor ID.</p> <p>wDeviceID: Specify the desired device ID. A zero value means to ignore the device ID.</p> <p>wCardNum: Specify the matched card index, a zero based index, based on the matched vendor ID and device ID.</p> |
| <p>Return Value:</p> <p>Return a handle to presents 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.</p> |

Users need to call PCIE_Close to release handle once the handle is no more used.

■ PCIE_Close

Function:

Close a handle associated to the PCIe card.

Prototype:

```
void PCIE_Close(  
    PCIE_HANDLE hFPGA);
```

Parameters:

hFPGA:
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 hFPGA,  
    PCIE_BAR PciBar,  
    PCIE_ADDRESS PciAddress,  
    uint32_t *pdwData);
```

Parameters:

hFPGA:
A PCIe handle return by PCIE_Open function.

PciBar:
Specify the target BAR.

PciAddress:
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 hFPGA,  
    PCIE_BAR PciBar,  
    PCIE_ADDRESS PciAddress,  
    uint32_t dwData);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

PciBar:

Specify the target BAR.

PciAddress:

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 hFPGA,  
    PCIE_BAR PciBar,  
    PCIE_ADDRESS PciAddress,  
    uint8_t *pByte);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

PciBar:

Specify the target BAR.

PciAddress:

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 hFPGA,  
    PCIE_BAR PciBar,  
    PCIE_ADDRESS PciAddress,  
    uint8_t Byte);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

PciBar:

Specify the target BAR.

PciAddress:

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.
Maximal read size is (4GB-1) bytes.

Prototype:

```
bool PCIE_DmaRead(  
    PCIE_HANDLE hFPGA,  
    PCIE_LOCAL_ADDRESS LocalAddress,  
    void *pBuffer,  
    uint32_t dwBufSize  
);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

LocalAddress:

Specify the target memory-mapped address in FPGA.

pBuffer:

A pointer to a memory buffer to retrieved the data from FPGA. The size of buffer should be equal or larger the dwBufSize.

dwBufSize:

Specify the byte number of data retrieved from FPGA.

Return Value:

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

■ PCIE_DmaWrite

Function:

Write data to the memory-mapped memory of FPGA board in DMA.

Prototype:

```
bool PCIE_DmaWrite(  
    PCIE_HANDLE hFPGA,  
    PCIE_LOCAL_ADDRESS LocalAddress,  
    void *pData,  
    uint32_t dwDataSize  
);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

LocalAddress:

Specify the target memory mapped address in FPGA.

pData:

A pointer to a memory buffer to store the data which will be written to FPGA.

dwDataSize:

Specify the byte number of data which will be written to FPGA.

Return Value:

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

■ PCIE_ConfigRead32

Function:

Read PCIe Configuration Table. Read a 32-bit data by given a byte offset.

Prototype:

```
bool PCIE_ConfigRead32 (  
    PCIE_HANDLE hFPGA,  
    uint32_t Offset,  
    uint32_t *pData32  
);
```

Parameters:

hFPGA:

A PCIe handle return by PCIE_Open function.

Offset:

Specify the target byte of offset in PCIe configuration table.

pData32:

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

Return Value:

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

7.5 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 DMA.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM\Demonstrations\PCIe_Fundamental\demo_batch

The folder includes following files:

- FPGA Configuration File: DE5A_NET.sof
- Download Batch file: test.bat
- Windows Application Software folder : windows_app, includes
 - ✧ PCIE_FUNDAMENTAL.exe
 - ✧ TERASIC_PCIE_AVMM.DLL

■ Demonstration Setup

1. Install the FPGA board on your PC as shown in **Figure 7-9**.

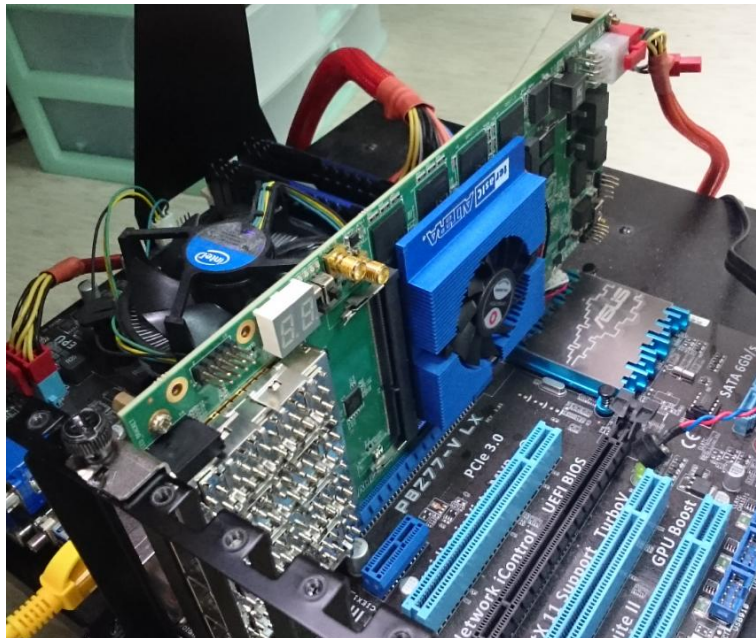


Figure 7-9 FPGA board installation on PC

2. Configure FPGA with DE5A_NET.sof by executing the test.bat.
3. Install 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 the Windows has detected the FPGA Board by checking the Windows Control panel as shown in **Figure 7-10**.

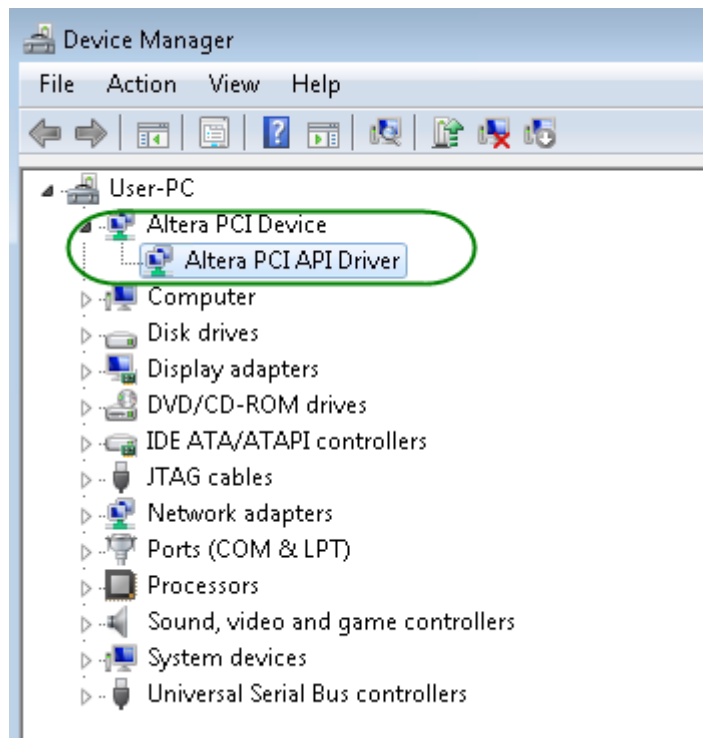


Figure 7-10 Screenshot for PCIe Driver

6. Goto windows_app folder, execute PCIE_FUNDAMENTAL.exe. A menu will appear as shown in **Figure 7-11**.

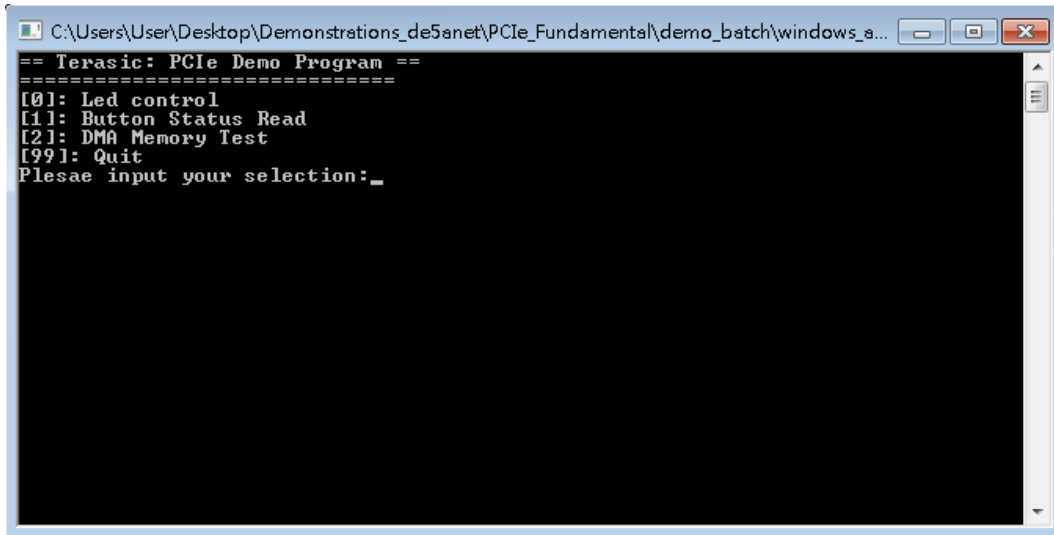


Figure 7-11 Screenshot of Program Menu

7. Type 0 followed by a ENTER key to select Led Control item, then input 15 (hex 0x0f) will make all led on as shown in **Figure 7-12**. If input 0 (hex 0x00), all led will be turn off.

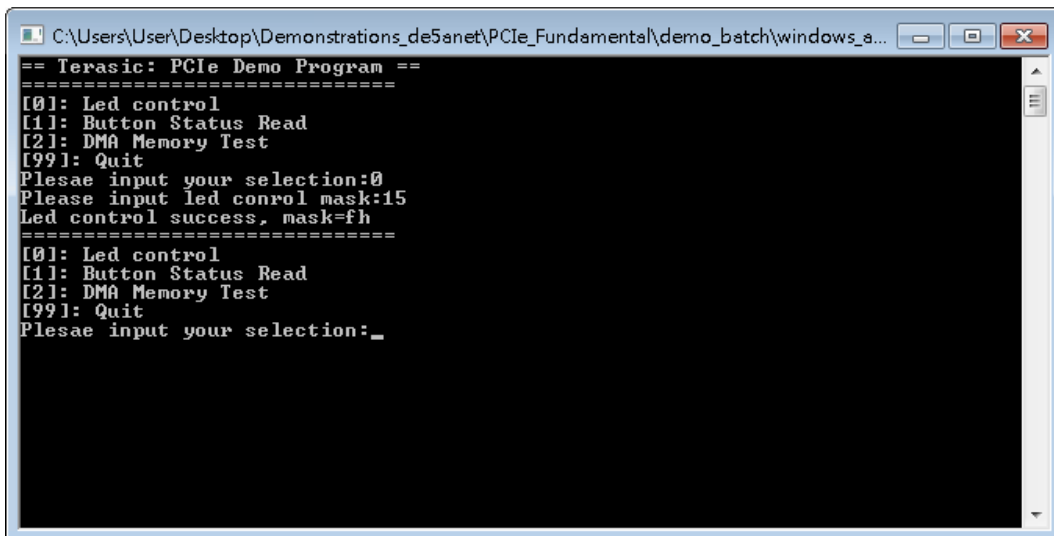


Figure 7-12 Screenshot of LED Control

8. Type 1 followed by an ENTER key to select Button Status Read item. The button status will be report as shown in **Figure 7-13**.

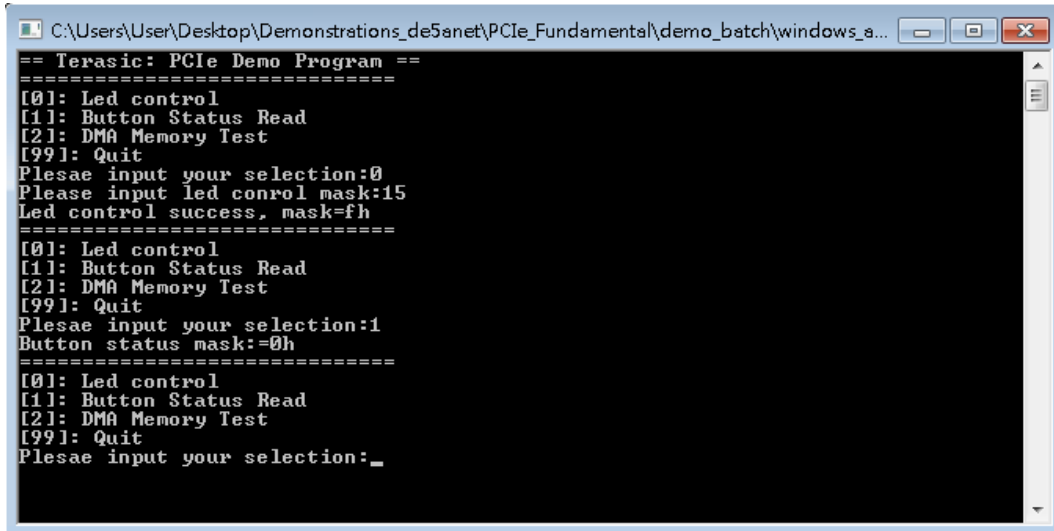


Figure 7-13 Screenshot of Button Status Report

9. Type 2 followed by an ENTER key to select DMA Testing item. The DMA test result will be report as shown in **Figure 7-14**.

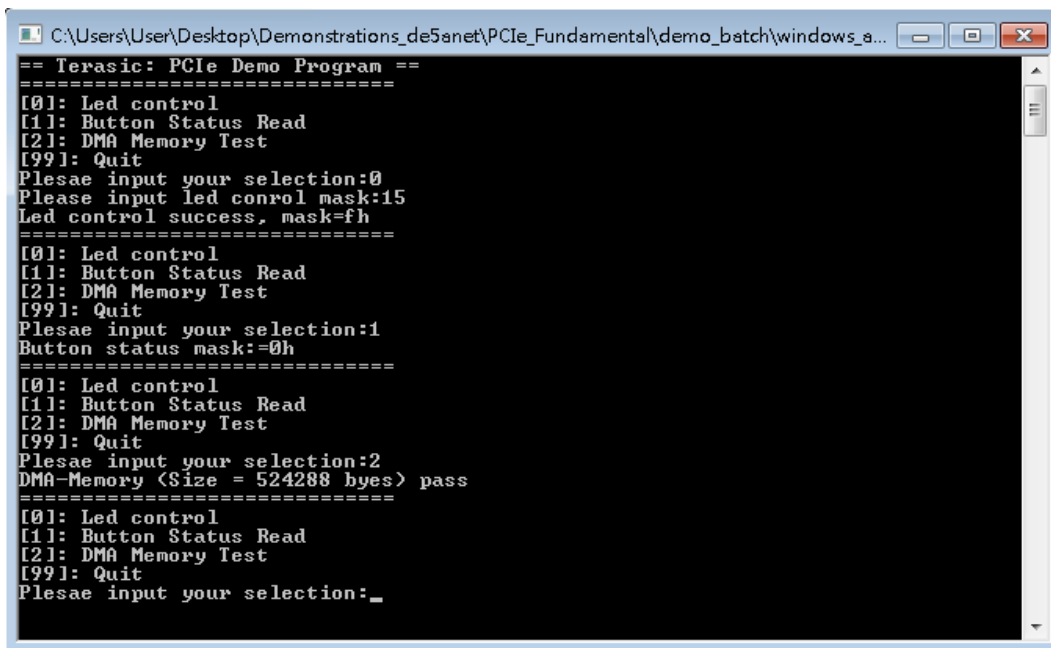


Figure 7-14 Screenshot of DMA Memory Test Result

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

■ Development Tools

- Quartus Prime 16.1.2 Standard Edition
- Visual C++ 2012

■ Demonstration Source Code Location

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

■ FPGA Application Design

Figure 7-15 shows the system block diagram in the FPGA system. In the Qsys, Altera 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.

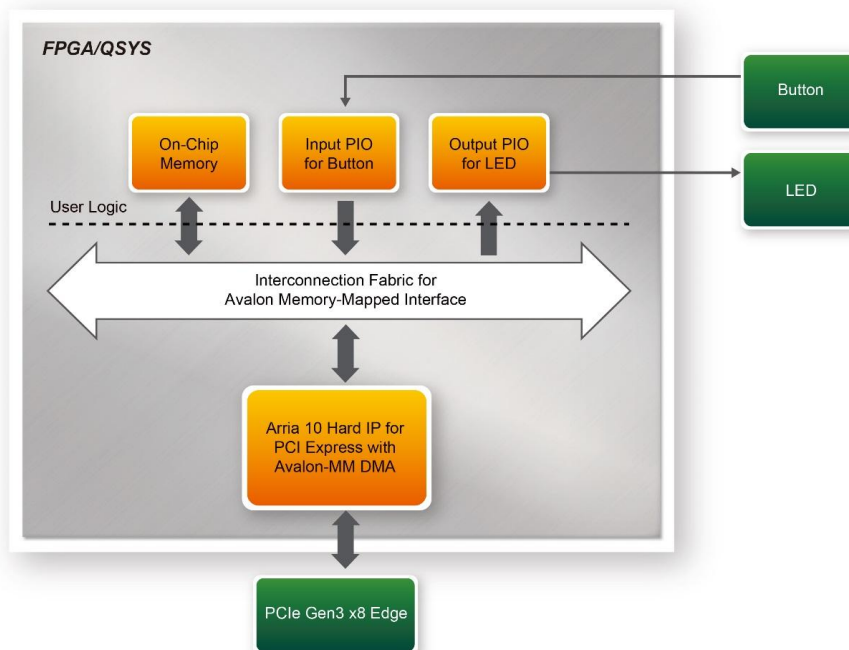


Figure 7-15 Hardware block diagram of the PCIe reference design

■ Windows Based Application Software Design

The application software project is built by Visual C++ 2012. The project includes the following major files:

| Name | Description |
|----------------------|---|
| PCIE_FUNDAMENTAL.cpp | Main program |
| PCIE.c | Implement dynamically load for |
| PCIE.h | TERASIC_PCIE_AVMM.DLL |
| TERASIC_PCIE_AVMM.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.

```

#include "PCIE.h"

#define DEMO_PCIE_USER_BAR          PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR      0x4000010
#define DEMO_PCIE_IO_BUTTON_ADDR  0x4000020
#define DEMO_PCIE_MEM_ADDR         0x00000000

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

The base address of BUTTON and LED controllers are 0x4000010 and 0x4000020 based on PCIE_BAR4, in respectively. The on-chip memory base address is 0x00000000 relative to the DMA controller.

Before accessing the FPGA through PCI Express, the application first calls PCIE_Load to dynamically load the TERASIC_PCIE_AVMM.dll. 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_AVMM.h. If developer change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value define in TERASIC_PCIE_AVMM.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:

```
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:

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


7.6 PCIe Reference Design - DDR4

The application reference design shows how to add DDR4 Memory Controllers for DDR4-A SODIMM and DDR4-B SODIMM into the PCIe Quartus project based on the PCIe_Fundamental Quartus project and perform 4GB data DMA for both SODIMM. 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\PCIe_DDR4\demo_batch

The folder includes following files:

- FPGA Configuration File: DE5A_NET.sof
- Download Batch file: test.bat
- Windows Application Software folder : windows_app, includes
 - ✧ PCIE_DDR4.exe
 - ✧ TERASIC_PCIE_AVMM.dll

■ Demonstration Setup

1. Install both DDR4 2400 4GB SODIMM on the FPGA board.
2. Install the FPGA board on your PC.
3. Configure FPGA with DE5A_NET.sof by executing the test.bat.
4. Install PCIe driver if necessary.
5. Restart Windows
6. Make sure the Windows has detected the FPGA Board by checking the Windows Control panel.
7. Goto windows_app folder, execute PCIE_DDR4.exe. A menu will appear as shown in **Figure 7-16**.

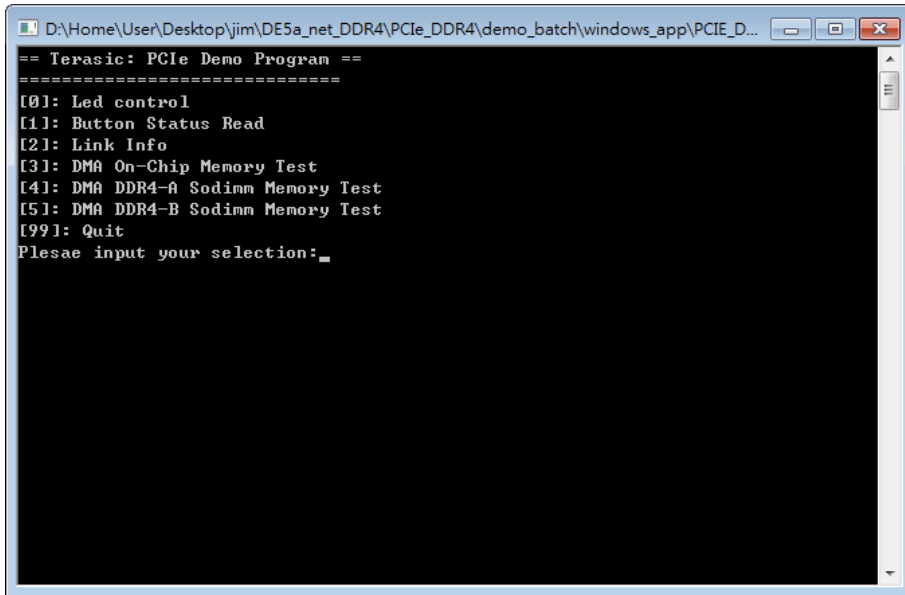


Figure 7-16 Screenshot of Program Menu

- Type 2 followed by a ENTER key to select Link Info item. The PCIe link information will be shown as in **Figure 7-17**. Gen3 link speed and x8 link width are expected.

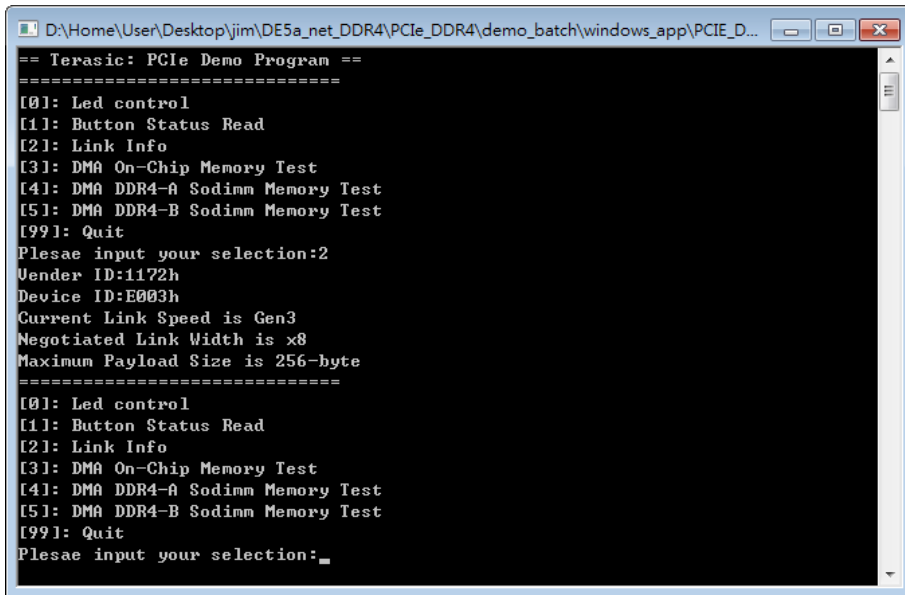


Figure 7-17 Screenshot of Link Info

- Type 3 followed by an ENTER key to select DMA On-Chip Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-18**.

```

D:\Home\User\Desktop\jim\DE5a_net_DDR4\PCIe_DDR4\demo_batch\windows_app\PCIE_D...
Maximum Payload Size is 256-byte
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:3
DMA Memory Test, Address = 0x0, Size = 0x80000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read...
Readback Data Verify...
DMA-Memory Address = 0x0, Size = 0x80000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:

```

Figure 7-18 Screenshot of On-Chip Memory DMA Test Result

10. Type 4 followed by an ENTER key to select DMA DDR4-A SODIMM Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-19**.

```

D:\Home\User\Desktop\jim\DE5a_net_DDR4\PCIe_DDR4\demo_batch\windows_app\PCIE_D...
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:4
DMA Memory Test, Address = 0x200000000, Size = 0x100000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read...
Readback Data Verify...
DMA Read...
Readback Data Verify...
DMA-Memory Address = 0x200000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:

```

Figure 7-19 Screenshot of DDR4-A SOSIMM Memory DAM Test Result

11. Type 5 followed by an ENTER key to select DMA DDR4-B SODIMM Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-20**.

```

D:\Home\User\Desktop\jim\DE5a_net_DDR4\PCIE_DDR4\demo_batch\windows_app\PCIE_D...
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:5
DMA Memory Test, Address = 0x40000000, Size = 0x10000000 Bytes...
Generate Test Pattern...
DMA Write...
DMA Read...
Readback Data Verify...
DMA Read...
Readback Data Uerify...
DMA-Memory Address = 0x40000000, Size = 0x10000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:

```

Figure 7-20 Screenshot of DDR4-B SOSIMM Memory DAM Test Result

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

■ Development Tools

- Quartus Prime 16.1.2 Standard Edition
- Visual C++ 2012

■ Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIE_DDR4
- Visual C++ Project: Demonstrations\PCIE_SW_KIT\Windows\PCIE_DDR4

■ FPGA Application Design

Figure 7-21 shows the system block diagram in the FPGA system. In the Qsys, Altera 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.

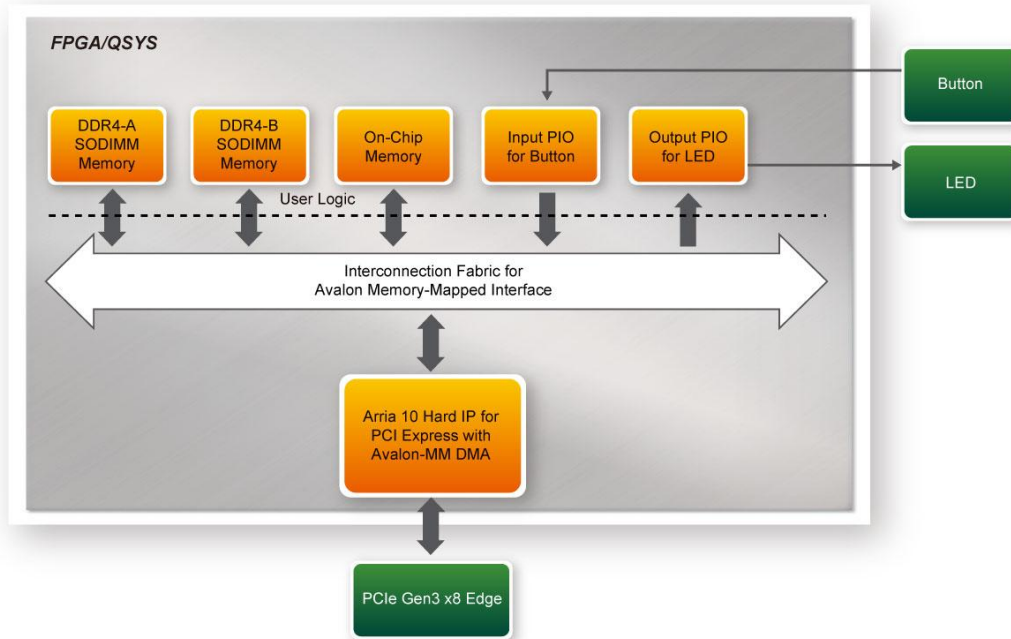


Figure 7-21 Hardware block diagram of the PCIe_DDR4 reference design

■ Windows Based Application Software Design

The application software project is built by Visual C++ 2012. The project includes the following major files:

| Name | Description |
|---------------------|---|
| PCIE_DDR4.cpp | Main program |
| PCIE.c | Implement dynamically load for |
| PCIE.h | TERAISC_PCIE_AVMM.DLL |
| TERASIC_PCIE_AVMM.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      0x4000010
#define DEMO_PCIE_IO_BUTTON_ADDR  0x4000020
#define DEMO_PCIE_ONCHIP_MEM_ADDR  0x00000000
#define DEMO_PCIE_DDR4A_MEM_ADDR   0x200000000
#define DEMO_PCIE_DDR4B_MEM_ADDR   0x400000000

#define ONCHIP_MEM_TEST_SIZE       (512*1024) //512KB
#define DDR4A_MEM_TEST_SIZE        (4u11*1024*1024*1024) //4GB
#define DDR4B_MEM_TEST_SIZE        (4u11*1024*1024*1024) //4GB
```

The base address of BUTTON and LED controllers are 0x4000010 and 0x4000020 based on PCIE_BAR4, in respectively. The on-chip memory base address is 0x00000000 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 PCIE_Load to dynamically load the Terasic_PCIE_AVMM.DLL. 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_AVMM.h. If developer change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value define in Terasic_PCIE_AVMM.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:

```
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:

```
PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
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, 0x90, &Data32)){  
    switch((Data32 >> 16) & 0x0F){  
        case 1:  
            printf("Current Link Speed is Gen1\r\n");  
            break;  
        case 2:  
            printf("Current Link Speed is Gen2\r\n");  
            break;  
        case 3:  
            printf("Current Link Speed is Gen3\r\n");  
            break;  
        default:  
            printf("Current Link Speed is Unknown\r\n");  
            break;  
    }  
    switch((Data32 >> 20) & 0x3F){  
        case 1:  
            printf("Negotiated Link Width is x1\r\n");  
            break;  
        case 2:  
            printf("Negotiated Link Width is x2\r\n");  
            break;  
        case 4:  
            printf("Negotiated Link Width is x4\r\n");  
            break;  
        case 8:  
            printf("Negotiated Link Width is x8\r\n");  
            break;  
        case 16:  
            printf("Negotiated Link Width is x16\r\n");  
            break;  
        default:  
            printf("Negotiated Link Width is Unknown\r\n");  
            break;  
    }  
}else{  
    bPass = false;  
}
```

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. Arria 10 Hard IP for PCI Express with Avalon-MM DMA IP is used in this demonstration. For detail about this IP, please refer to Altera document [ug_a10_pcie_avmm_dma.pdf](#).

8.1 PCI Express System Infrastructure

Figure 8-1 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 Arria 10 Hard IP for PCI Express with Avalon-MM DMA. The application software on the PC side is developed by Terasic based on Altera's PCIe kernel mode driver.

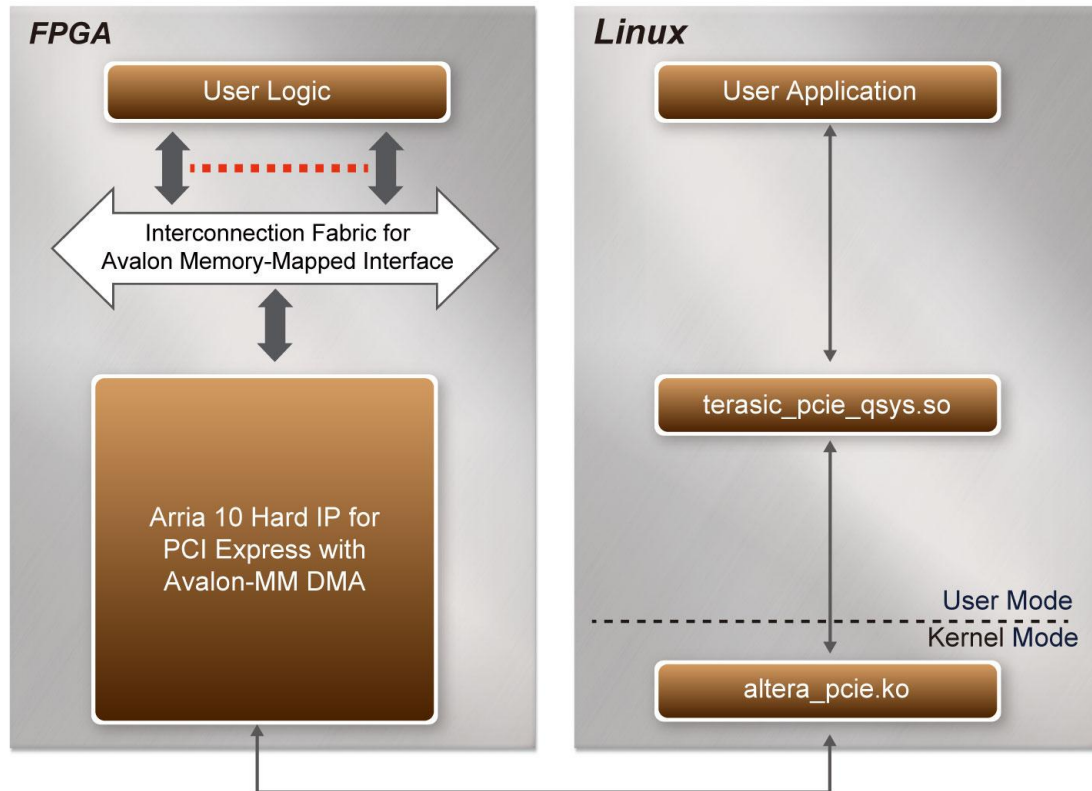


Figure 8-1 Infrastructure of PCI Express System

8.2 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 Linux. CentOS 7.2 is recommended. The SDK is located in the “CDROM/Demonstrations/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 0xE003. 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 project and rebuild the driver. The ID is defined in the file PCIe_SW_KIT/Linux/PCIe_Driver/altera_pcie_cmd.h.

The PCI Express Library is implemented as a single .so file named terasic_pcie_qsys.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, Altera AVMM DMA is required as the read and write operations are specified under the hardware design on the FPGA.

8.3 PCI Express Software Stack

Figure 8-2 shows the software stack for the PCI Express application software on 64-bit Linux. The PCIe library module terasic_pcie_qys.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 altera_pcie.ko kernel driver is provided by Altera.

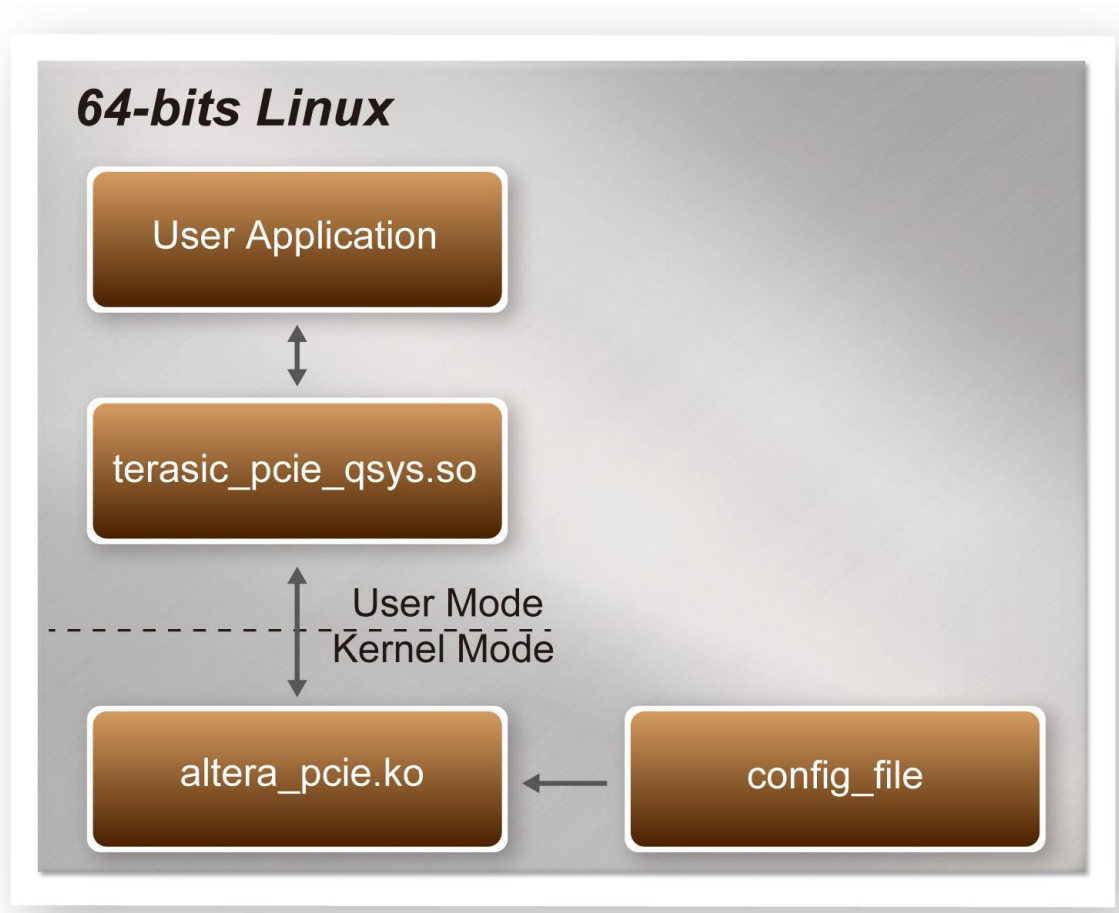


Figure 8-2 PCI Express Software Stack

■ Install PCI Express Driver on Linux

To make sure the PCIe driver can meet your kernel of Linux distribution, the driver altera_pcie.ko should be recompile before use it. The PCIe driver project is locate in the folder:

"CDROM/Demonstrations/PCle_SW_KIT/Linux/PCle_Driver"

The folder includes the following files:

- altera_pcie.c
- altera_pcie.h
- altera_pcie_cmd.h
- Makefile
- load_driver

- unload
- config_file

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

1. Install the DE5a-Net on the PCIe slot of the host PC
2. Make sure Altera Programmer and USB-Blaster II driver are installed
3. Open a terminal and use "cd" command to goto the folder "CDROM/Demonstrations/PCIe_Fundamental/demo_batch".
4. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by tying the following commands in terminal. Replace "/home/centos/intelFPGA/16.1/quartus" to your quartus installation path.


```
export QUARTUS_ROOTDIR=/home/centos/intelFPGA/16.1/quartus
```
5. Execute "sudo -E sh test.sh" command to configure the FPGA
6. Restart Linux operation system. In Linux, open a terminal and use "cd" command to goto the PCIe_Driver folder
7. Type the following commands to compile and install the driver altera_pcie.ko, and make sure driver is loaded successfully and FPGA is detected by the driver as shown in **Figure 8-3**.
 - make
 - sudo sh load_driver
 - dmesg | tail -n 15

```
centos@localhost:PCIe_Driver$ sudo sh load_driver
Matching Device Found
centos@localhost:PCIe_Driver$ dmesg | tail -n 15
[ 35.485745] SELinux: initialized (dev tmpfs, type tmpfs), uses transition SIDs
[ 246.876696] SELinux: initialized (dev tmpfs, type tmpfs), uses transition SIDs
[ 273.531603] Altera PCIe: altera_pcie_init(), May 11 2017 16:29:58
[ 273.531629] Altera PCIe 0000:01:00.0: enabling device (0000 -> 0002)
[ 273.531713] Altera PCIe 0000:01:00.0: pci_enable_device() successful
[ 273.531744] Altera PCIe 0000:01:00.0: irq 134 for MSI/MSI-X
[ 273.531754] Altera PCIe 0000:01:00.0: pci_enable_msi() successful
[ 273.531758] Altera PCIe 0000:01:00.0: BAR[0] 0xe8000000-0xe80001ff flags 0x0014220c, length 512
[ 273.531760] Altera PCIe 0000:01:00.0: BAR[1] 0x00000000-0x00000000 flags 0x00000000, length 0
[ 273.531762] Altera PCIe 0000:01:00.0: BAR[2] 0x00000000-0x00000000 flags 0x00000000, length 0
[ 273.531764] Altera PCIe 0000:01:00.0: BAR[3] 0x00000000-0x00000000 flags 0x00000000, length 0
[ 273.531766] Altera PCIe 0000:01:00.0: BAR[4] 0xe0000000-0xe7ffffff flags 0x0014220c, length 134217728
[ 273.531768] Altera PCIe 0000:01:00.0: BAR[5] 0x00000000-0x00000000 flags 0x00000000, length 0
[ 273.531783] Altera PCIe 0000:01:00.0: BAR[0] mapped to 0xffffc9000307c000, length 512
[ 273.532037] Altera PCIe 0000:01:00.0: BAR[4] mapped to 0xffffc9000c200000, length 134217728
centos@localhost:PCIe_Driver$
```

Figure 8-3 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/PCIe_SW_KIT/Linux/PCIe_Library. It includes the following files:

- TERASIC_PCIE_AVMM.h
- terasic_pcie_qsys.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_AVMM.h in the C/C++ project.
3. Copy terasic_pcie_qsys.so to the folder where the project execution file is located.
4. Dynamically load terasic_pcie_qsys.so in C/C++ program. To load the terasic_pcie_qsys.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_qsys.so API. The details of API are described below:

8.4 PCI Express Library API

The API is the same as Windows Library. Please refer to the section 7.4 PCI Express Library API in this document.

8.5 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 DMA.

■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM/Demonstrations/PCIe_Fundamental/demo_batch

The folder includes following files:

- FPGA Configuration File: DE5A_NET.sof
- Download Batch file: test.sh
- Linux Application Software folder : linux_app, includes
 - ✧ PCIE_FUNDAMENTAL
 - ✧ terasic_pcie_qsys.so

■ Demonstration Setup

1. Install the FPGA board on your PC as shown in **Figure8-4**.

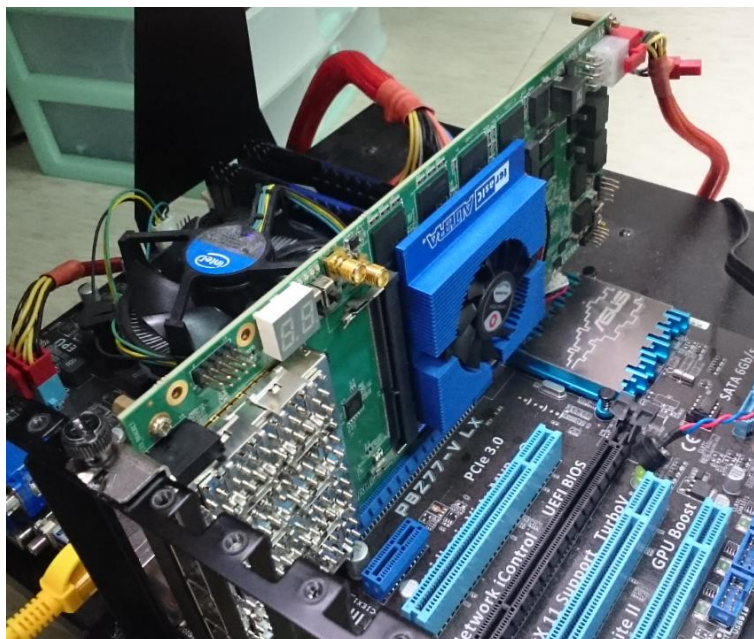


Figure 8-4 FPGA board installation on PC

2. Open a terminal and use "cd" command to goto "CDROM/Demonstrations/PCle_Fundamental/demo_batch".
3. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by typing the following commands in terminal. Replace /home/centos/intelFPGA/16.1/quartus to your quartus installation path.

```
export QUARTUS_ROOTDIR=/home/centos/intelFPGA/16.1/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/PCle_SW_KIT/Linux/PCle_Driver.
7. Type "ls -l /dev/altera_pcie*" to make sure the Linux has detected the FPGA Board. If the FPGA board is detected, developers can find the /dev/altera_pcieX(where X is 0~255) in Linux file system as shown below.

```
centos@localhost:PCie_Driver$ ls -l /dev/altera_pcie*  
crw-rw-rw-. 1 root wheel 248, 0 5月 11 15:42 /dev/altera_pcie0
```

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

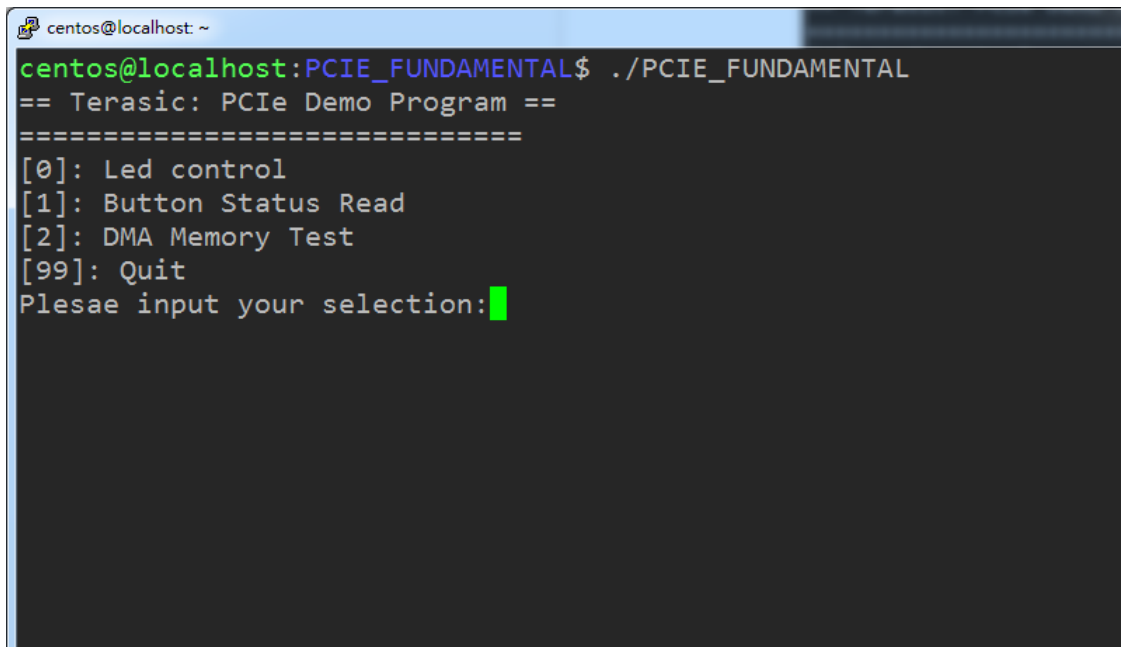


Figure 8-5 Screenshot of Program Menu

- Type 0 followed by a ENTER key to select Led Control item, then input 15 (hex 0x0f) will make all led on as shown in **Figure 8-6**. If input 0 (hex 0x00), all led will be turn off.

```

centos@localhost: ~
centos@localhost:PCIE_FUNDAMENTAL$ ./PCIE_FUNDAMENTAL
== Terasic: PCIe Demo Program ==
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:0
Please input led conrol mask:15
Led control success, mask=fh
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:█

```

Figure 8-6 Screenshot of LED Control

- Type 1 followed by an ENTER key to select Button Status Read item. The button status will be report as shown in **Figure 8-7**.

```

centos@localhost: ~
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:0
Please input led conrol mask:15
Led control success, mask=fh
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:1
Button status mask:=0h
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:█

```


Figure 8-7 Screenshot of Button Status Report

11. Type 2 followed by an ENTER key to select DMA Testing item. The DMA test result will be report as shown in **Figure 8-8**.

```
centos@localhost: ~
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:1
Button status mask:=0h
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:2
DMA-Memory (Size = 524288 bytes) pass
=====
[0]: Led control
[1]: Button Status Read
[2]: DMA Memory Test
[99]: Quit
Plesae input your selection:
```

Figure 8-8 Screenshot of DMA Memory Test Result

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

■ **Development Tools**

- Quartus Prime 16.1.2 Standard Edition
- GNU Compiler Collection, Version 4.8 is recommend

■ **Demonstration Source Code Location**

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

■ **FPGA Application Design**

Figure 8-9 shows the system block diagram in the FPGA system. In the Qsys, Altera 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.

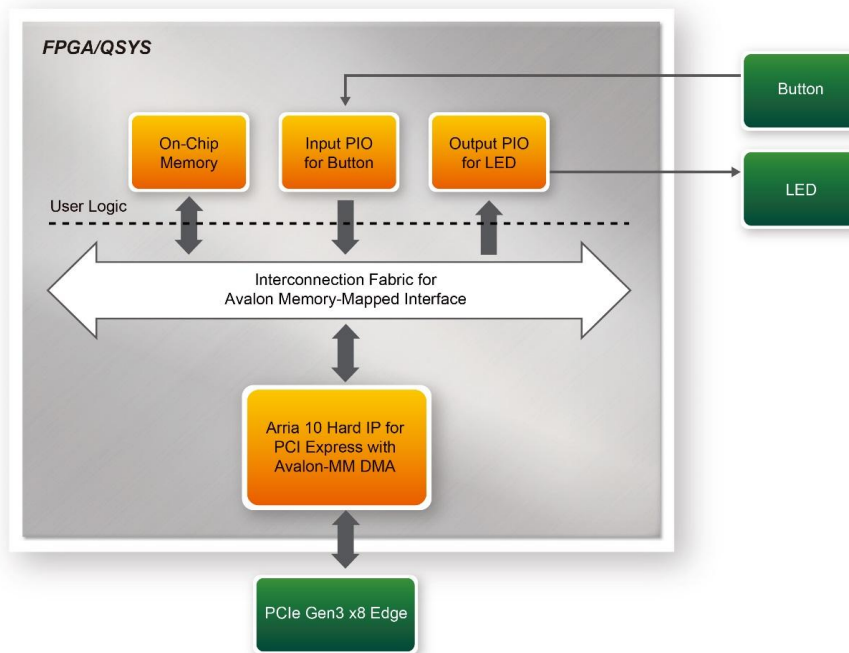


Figure 8-9 Hardware block diagram of the PCIe reference design

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 |
| PCIE.c | Implement dynamically load for terasic_pcie_qsys.so library file |
| PCIE.h | |
| TERASIC_PCIE_AVMM.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.

```
#include "PCIE.h"

#define DEMO_PCIE_USER_BAR      PCIE_BAR4
#define DEMO_PCIE_IO_LED_ADDR   0x4000010
#define DEMO_PCIE_IO_BUTTON_ADDR 0x4000020
#define DEMO_PCIE_MEM_ADDR      0x00000000

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

The base address of BUTTON and LED controllers are 0x4000010 and 0x4000020 based on PCIE_BAR4, in respectively. The on-chip memory base address is 0x00000000 relative to the DMA controller.

Before accessing the FPGA through PCI Express, the application first calls PCIE_Load to dynamically load the terasic_pcie_qsys.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_AVMM.h. If developer change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value define in Terasic_PCIE_AVMM.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:

```
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:

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

8.6 PCIe Reference Design - DDR4

The application reference design shows how to add DDR4 Memory Controllers for DDR4-A SODIMM and DDR4-B SODIMM into the PCIe Quartus project based on the PCIe_Fundamental Quartus project and perform 4GB data DMA for both SODIMM. 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/PCIe_DDR4/demo_batch

The folder includes following files:

- FPGA Configuration File: DE5A_NET.sof
- Download Batch file: test.sh
- Linux Application Software folder : linux_app, includes
 - ✧ PCIE_DDR4
 - ✧ terasic_pcie_qsys.so

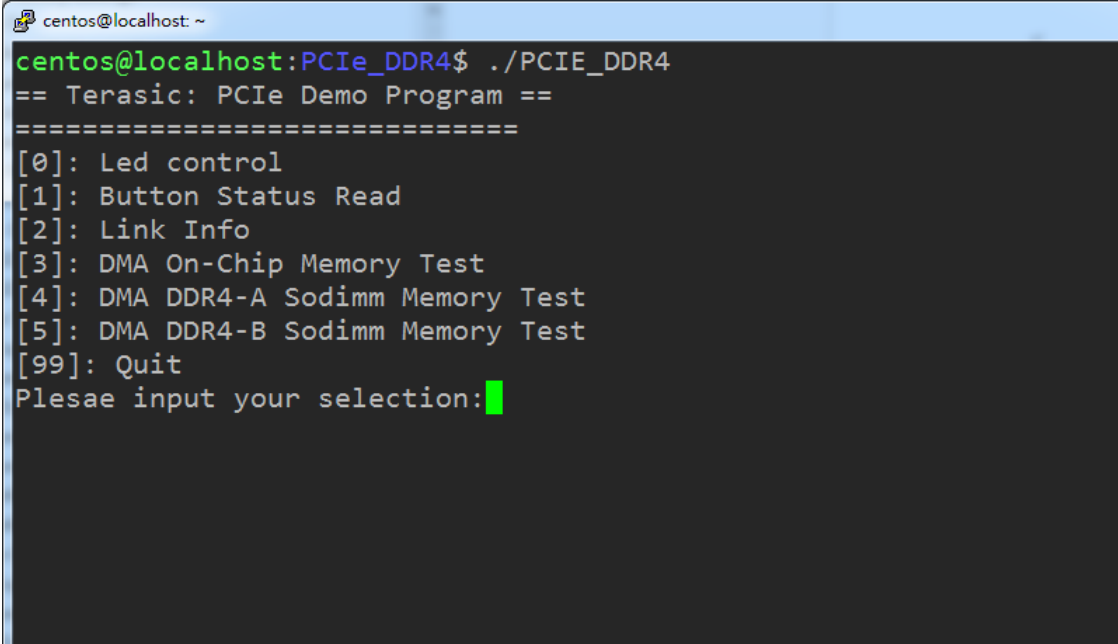
■ Demonstration Setup

1. Install both DDR4 2400 4GB SODIMM on the FPGA board.
2. Install the FPGA board on your PC.
3. Open a terminal and use "cd" command to goto "CDROM/Demonstrations/PCIe_Fundamental/demo_batch".
4. Set QUARTUS_ROOTDIR variable pointing to the Quartus installation path. Set QUARTUS_ROOTDIR variable by tying the following commands in terminal. Replace

/home/centos/intelFPGA/16.1/quartus to your quartus installation path.

```
export QUARTUS_ROOTDIR=/home/centos/intelFPGA/16.1/quartus
```

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



```
centos@localhost: ~  
centos@localhost:PCIE_DDR4$ ./PCIE_DDR4  
== Terasic: PCIe Demo Program ==  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: Link Info  
[3]: DMA On-Chip Memory Test  
[4]: DMA DDR4-A Sodimm Memory Test  
[5]: DMA DDR4-B Sodimm Memory Test  
[99]: Quit  
Plesae input your selection: █
```

Figure 8-10 Screenshot of Program Menu

10. Type 2 followed by an ENTER key to select Link Info item. The PCIe link information will be shown as in **Figure 8-11**. Gen3 link speed and x8 link width are expected.

```
centos@localhost: ~  
[4]: DMA DDR4-A Sodimm Memory Test  
[5]: DMA DDR4-B Sodimm Memory Test  
[99]: Quit  
Plesae input your selection:2  
Vender ID:1172h  
Device ID:E003h  
Current Link Speed is Gen3  
Negotiated Link Width is x8  
Maximum Payload Size is 256-byte  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: Link Info  
[3]: DMA On-Chip Memory Test  
[4]: DMA DDR4-A Sodimm Memory Test  
[5]: DMA DDR4-B Sodimm Memory Test  
[99]: Quit  
Plesae input your selection:█
```

Figure 8-11 Screenshot of Link Info

11. Type 3 followed by an ENTER key to select DMA On-Chip Memory Test item. The DMA write and read test result will be report as shown in **Figure 8-12**.

```
centos@localhost: ~  
[5]: DMA DDR4-B Sodimm Memory Test  
[99]: Quit  
Plesae input your selection:3  
DMA Memory Test, Address = 0x0, Size = 0x80000 Bytes...  
Generate Test Pattern...  
DMA Write...  
DMA Read...  
Readback Data Verify...  
DMA-Memory Address = 0x0, Size = 0x80000 bytes pass  
=====  
[0]: Led control  
[1]: Button Status Read  
[2]: Link Info  
[3]: DMA On-Chip Memory Test  
[4]: DMA DDR4-A Sodimm Memory Test  
[5]: DMA DDR4-B Sodimm Memory Test  
[99]: Quit  
Plesae input your selection:█
```

Figure 8-12 Screenshot of On-Chip Memory DMA Test Result

12. Type 4 followed by an ENTER key to select DMA DDR4-A SODIMM Memory Test item. The DMA write and read test result will be report as shown in **Figure 8-14**.

```
centos@localhost: ~
Plesae input your selection:4
DMA Memory Test, Address = 0x200000000, Size = 0x100000000 Bytes..
Generate Test Pattern...
DMA Write...
DMA Read...
Readback Data Verify...
DMA Read...
Readback Data Verify...
DMA-Memory Address = 0x200000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:█
```

Figure 8-14 Screenshot of DDR4-A SOSIMM Memory DAM Test Result

13. Type 5 followed by an ENTER key to select DMA DDR4-B SODIMM Memory Test item. The DMA write and read test result will be report as shown in **Figure 8-15**.

```
centos@localhost: ~
Plesae input your selection:5
DMA Memory Test, Address = 0x400000000, Size = 0x100000000 Bytes..
Generate Test Pattern...
DMA Write...
DMA Read...
Readback Data Verify...
DMA Read...
Readback Data Verify...
DMA-Memory Address = 0x400000000, Size = 0x100000000 bytes pass
=====
[0]: Led control
[1]: Button Status Read
[2]: Link Info
[3]: DMA On-Chip Memory Test
[4]: DMA DDR4-A Sodimm Memory Test
[5]: DMA DDR4-B Sodimm Memory Test
[99]: Quit
Plesae input your selection:█
```

Figure 8-15 Screenshot of DDR4-B SOSIMM Memory DAM Test Result

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

■ Development Tools

- Quartus Prime 16.1.2 Standard Edition
- GNU Compiler Collection, Version 4.8 is recommended

■ Demonstration Source Code Location

- Quartus Project: Demonstrations/PCIE_DDR4
- C++ Project: Demonstrations/PCie_SW_KIT/Linux/PCie_DDR4

■ FPGA Application Design

Figure 8-16 shows the system block diagram in the FPGA system. In the Qsys, Altera 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.

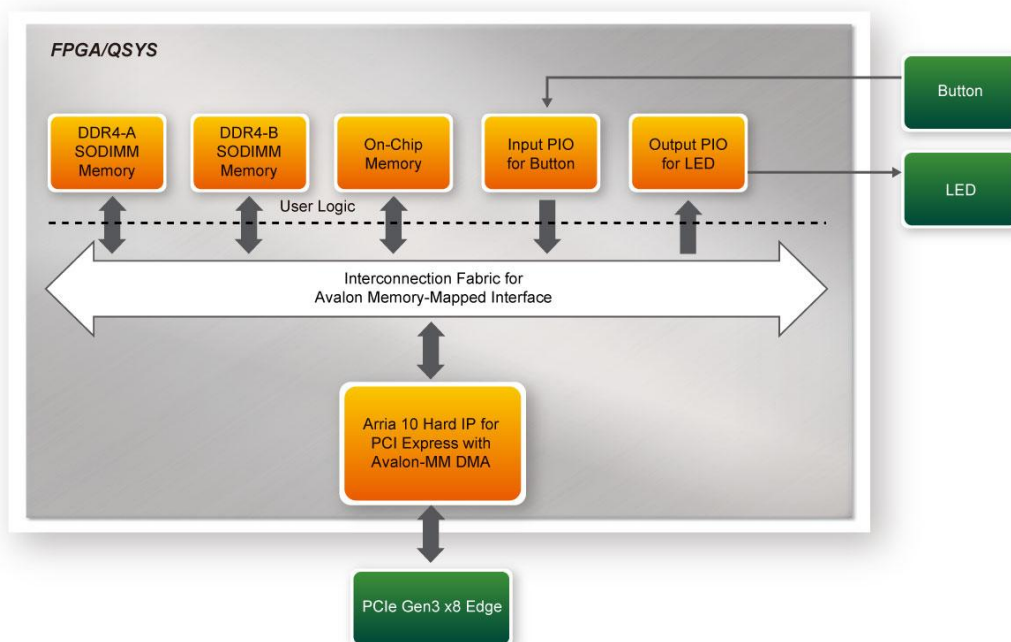


Figure 8-16 Hardware block diagram of the PCIe_DDR4 reference design

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 |
| PCIE.c | Implement dynamically load for terasic_pcie_qsys.so |
| PCIE.h | library file |
| TERASIC_PCIE_AVMM.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   0x4000010
#define DEMO_PCIE_IO_BUTTON_ADDR 0x4000020
#define DEMO_PCIE_ONCHIP_MEM_ADDR 0x00000000
#define DEMO_PCIE_DDR4A_MEM_ADDR 0x200000000
#define DEMO_PCIE_DDR4B_MEM_ADDR 0x400000000

#define ONCHIP_MEM_TEST_SIZE    (512*1024) //512KB
#define DDR4A_MEM_TEST_SIZE     (4u11*1024*1024*1024) //4GB
#define DDR4B_MEM_TEST_SIZE     (4u11*1024*1024*1024) //4GB
```

The base address of BUTTON and LED controllers are 0x4000010 and 0x4000020 based on PCIE_BAR4, in respectively. The on-chip memory base address is 0x00000000 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 PCIE_Load to dynamically load the terasic_pcie_qsys.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_AVMM.h. If developer change the Vendor ID and Device ID and PCI Express IP, they also need to change the ID value define in

TERASIC_PCIE_AVMM.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:

```
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:

```
PCIE_DmaWrite(hPCIE, LocalAddr, pWrite, nTestSize);  
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, 0x90, &Data32)){
    switch((Data32 >> 16) & 0x0F){
        case 1:
            printf("Current Link Speed is Gen1\r\n");
            break;
        case 2:
            printf("Current Link Speed is Gen2\r\n");
            break;
        case 3:
            printf("Current Link Speed is Gen3\r\n");
            break;
        default:
            printf("Current Link Speed is Unknown\r\n");
            break;
    }
    switch((Data32 >> 20) & 0x3F){
        case 1:
            printf("Negotiated Link Width is x1\r\n");
            break;
        case 2:
            printf("Negotiated Link Width is x2\r\n");
            break;
        case 4:
            printf("Negotiated Link Width is x4\r\n");
            break;
        case 8:
            printf("Negotiated Link Width is x8\r\n");
            break;
        case 16:
            printf("Negotiated Link Width is x16\r\n");
            break;
        default:
            printf("Negotiated Link Width is Unknown\r\n");
            break;
    }
} else{
    bPass = false;
}
```

Transceiver Verification

This chapter describes how to verify the FPGA transceivers for the QSFP+ connector. The Low Latency 40G and 10G Ethernet are introduced in this chapter to serve the purpose. Its test code is provided in the DE5a-Net system CD.

9.1 Function of the Transceiver Test Code

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

9.2 Loopback Fixture

To enable an external loopback of transceiver channels, one of the following two fixtures are required:

- QSFP+ Cable, as shown in **Figure 9-1**
- QSFP+ Loopback fixture, as shown in **Figure 9-2**



Figure 9-1 Optical QSFP+ Cable



Figure 9-2 QSFP+ Loopback Fixture

Figure 9-3 shows the FPGA board with two QSFP+ cable installed. **Figure 9-4** shows the FPGA board with four QSFP+ loopback fixtures installed.

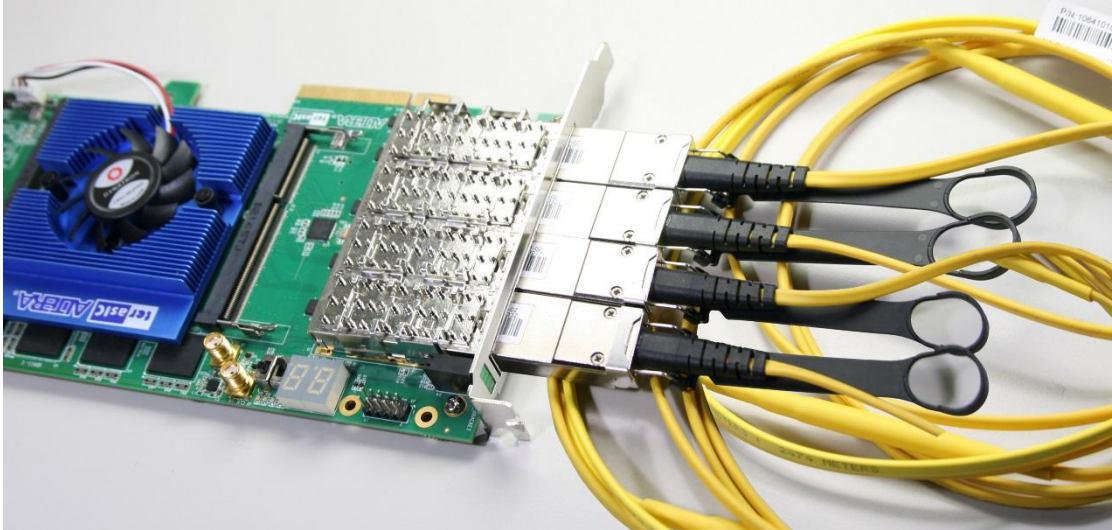


Figure 9-3 Two QSFP+ Cables Installed

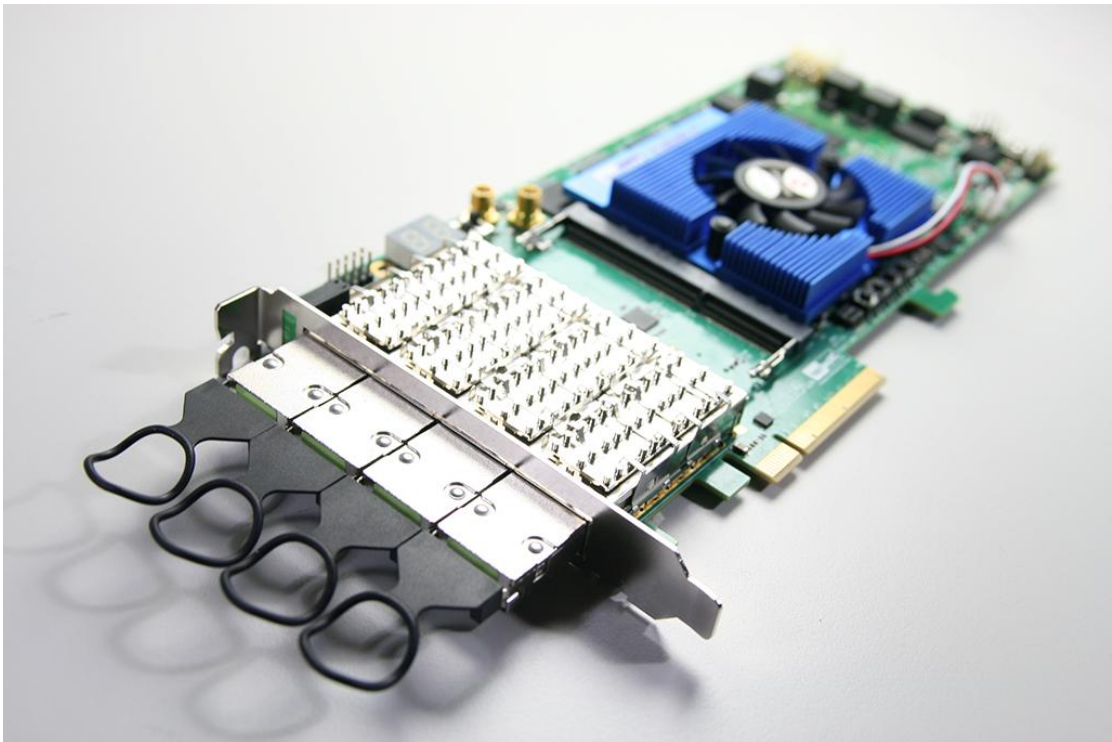


Figure 9-4 Four QSFP+ Loopback Fixtures Installed

9.3 Testing

The transceiver test code is available in the folder System CD\Tool\Transceiver_Test. Here are the procedures to perform transceiver channel test:

1. Copy Transceiver_Test folder to your local disk.
2. Ensure that the FPGA board is NOT powered on.
3. Plug-in the QSPF+ loopback fixtures.
4. Connect your FPGA board to your PC with a mini 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 Nios-Terminal, as shown in **Figure 9-5**.
8. To terminate the test, press one of the BUTTON0~3 buttons on the FPGA board.

The loopback test will terminate as shown in **Figure 9-6**.

```

ca. Altera Nios II EDS 15.0 [gcc4]
Using cable "DE5 [USB-1]", device 1, instance 0x00
Resetting and pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 122KB in 0.1s
Verified OK
Starting processor at address 0x00040240
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "DE5 [USB-1]", device 1, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

Transceiver for QSPF Testing...
Press buttons on the board can terminate the testing.
Apply default settgin...done
==== Time Elapsed: 0 Seconds ====
QSPF_A-0: PASS, XferCnt:546489088
QSPF_A-1: PASS, XferCnt:548224896
QSPF_A-2: PASS, XferCnt:547569664
QSPF_A-3: PASS, XferCnt:518172800
QSPF_B-0: PASS, XferCnt:422683520
QSPF_B-1: PASS, XferCnt:424444928
QSPF_B-2: PASS, XferCnt:424619904
QSPF_B-3: PASS, XferCnt:395225344
QSPF_C-0: PASS, XferCnt:299737984
QSPF_C-1: PASS, XferCnt:301499776
QSPF_C-2: PASS, XferCnt:300852608
QSPF_C-3: PASS, XferCnt:271459072
QSPF_D-0: PASS, XferCnt:176792064
QSPF_D-1: PASS, XferCnt:178555520
QSPF_D-2: PASS, XferCnt:177909120
QSPF_D-3: PASS, XferCnt:148514048
  
```

Figure 9-5 QSPF+ Transceiver Loopback Test in Progress

```
cmd: Altera Nios II EDS 15.0 [gcc4]
QSFP_C-0: PASS, XferCnt:3567951104
QSFP_C-1: PASS, XferCnt:3568034688
QSFP_C-2: PASS, XferCnt:3566551040
QSFP_C-3: PASS, XferCnt:3536299008
QSFP_D-0: PASS, XferCnt:3441572224
QSFP_D-1: PASS, XferCnt:3441656192
QSFP_D-2: PASS, XferCnt:3440174208
QSFP_D-3: PASS, XferCnt:3409920384
==== Time Elapsed: 25 Seconds ====
QSFP_A-0: PASS, XferCnt:4636009984
QSFP_A-1: PASS, XferCnt:4637213056
QSFP_A-2: PASS, XferCnt:4635699968
QSFP_A-3: PASS, XferCnt:4605445760
QSFP_B-0: PASS, XferCnt:4509897088
QSFP_B-1: PASS, XferCnt:4509977856
QSFP_B-2: PASS, XferCnt:4509318016
QSFP_B-3: PASS, XferCnt:4479065088
QSFP_C-0: PASS, XferCnt:4383516288
QSFP_C-1: PASS, XferCnt:4383598336
QSFP_C-2: PASS, XferCnt:4382116480
QSFP_C-3: PASS, XferCnt:4351864192
QSFP_D-0: PASS, XferCnt:4257137152
QSFP_D-1: PASS, XferCnt:4257219840
QSFP_D-2: PASS, XferCnt:4255739264
QSFP_D-3: PASS, XferCnt:4225485568
Transceiver Testing is terminated!
```

Figure 9-6 QSFP Transceiver Loopback Done

9.4 40G Ethernet Example

This 40G Ethernet example is generated according to the document [Low Latency 40G Ethernet Example Design User Guide](#). The Arria 10 LL(Low Latency) 40GbE IP is used in the example design. This example executes external loopback test through one of the QSFP+ ports on the FPGA main board. A QSFP+ loopback fixture is required to perform this demonstration. **Figure 9-7** shows the block diagram of this demonstration.

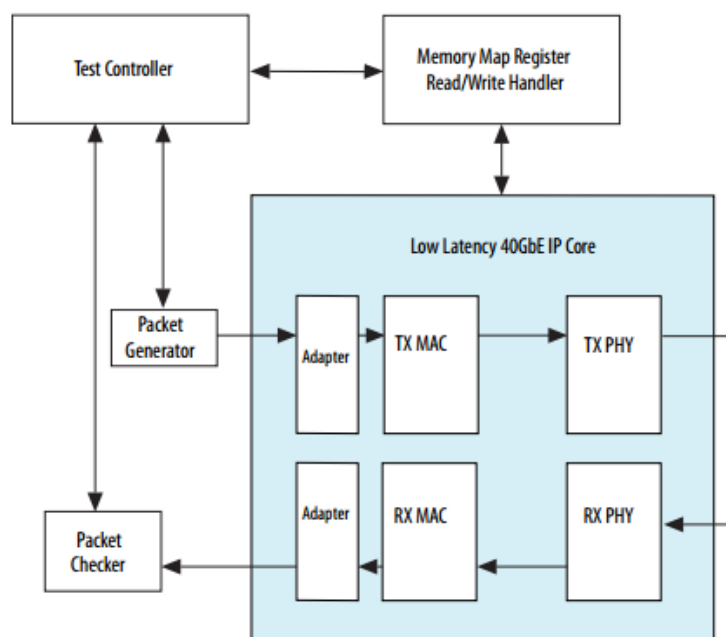


Figure 9-7 Block diagram of 40GbE demo

■ Project Information

The Quartus project information is shown in the table below.

| Item | Description |
|------------------|--|
| Project Location | CDROM/Demonstrations/LL40GbE |
| FPGA Bit Stream | CDROM/Demonstrations/LL40GbE/output_files/DE5A_NET.sof |

| | |
|-----------------|---|
| Test Scrip File | CDROM/Demonstrations/LL40GbE/hwtest/main.tcl main2.tcl is used to enable transceiver serial loopback for internal loopback |
| Quartus Version | Quartus Prime 16.1.2 Standard Edition |

The transceiver PMA settings used in this project are shown in the table below. These settings are defined in the DE5A_NET.qdf file. They are recommended for DE5a-NET board. Developers can adjust these values based on the characteristics of the transmission cable chosen.

| Direction | Item | Value |
|-----------|-----------------------------|-------|
| TX | VOD | 28 |
| | Pre-emphasis first post-tap | -7 |
| | Pre-emphasis pre-tap | -2 |
| RX | DC Gain | 4 |
| | EQ Control | 15 |
| | VGA | 7 |

■ Demonstration Setup

Here is the procedure to setup the demonstration. A QSFP+ loopback fixture is required for this demonstration. If you don't have a QSFP+ loopback fixture, please use main2.tcl instead of main.tcl in the following demonstration procedure. The main2.tcl is used to enable transceiver serial loopback for internal loopback.

1. Insert a QSFP+ loopback fixture into the QSFP-A port on DE5a-NET board, as shown in **Figure 9-8**.
2. Connect the host PC to the FPGA board using a mini-USB cable. Please make sure the USB-Blaster II driver is installed on the host PC.
3. Launch Quartus Prime programmer and make sure the USB-Blaster II is detected correctly.
4. In Quartus Prime Programmer, add the configuration bit stream file (./output_files/DE5A_NET.sof). Check the associated "Program/Configure" item and

click “Start” to start FPGA programming.

5. Launch System Console by selecting the menu item “Tools→System Debugging Tools→System Console” in Quartus.
6. In the System Console window, input the following commands to start the loopback test, as shown in **Figure 9-9**.

```
%cd hwtest  
%source main.tcl  
%run_test
```

7. The loopback test report will be displayed in the Tcl Console, as shown in **Figure 9-10**.



Figure 9-8 Setup QSPF loopback fixture

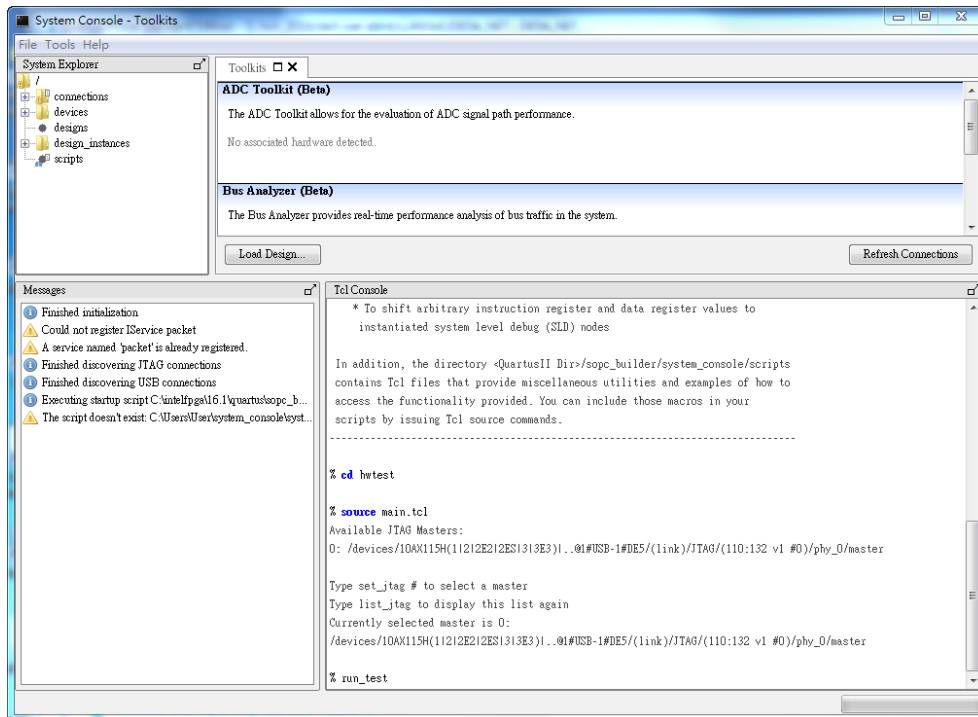


Figure 9-9 Launch the System Console for 40GbE Demo

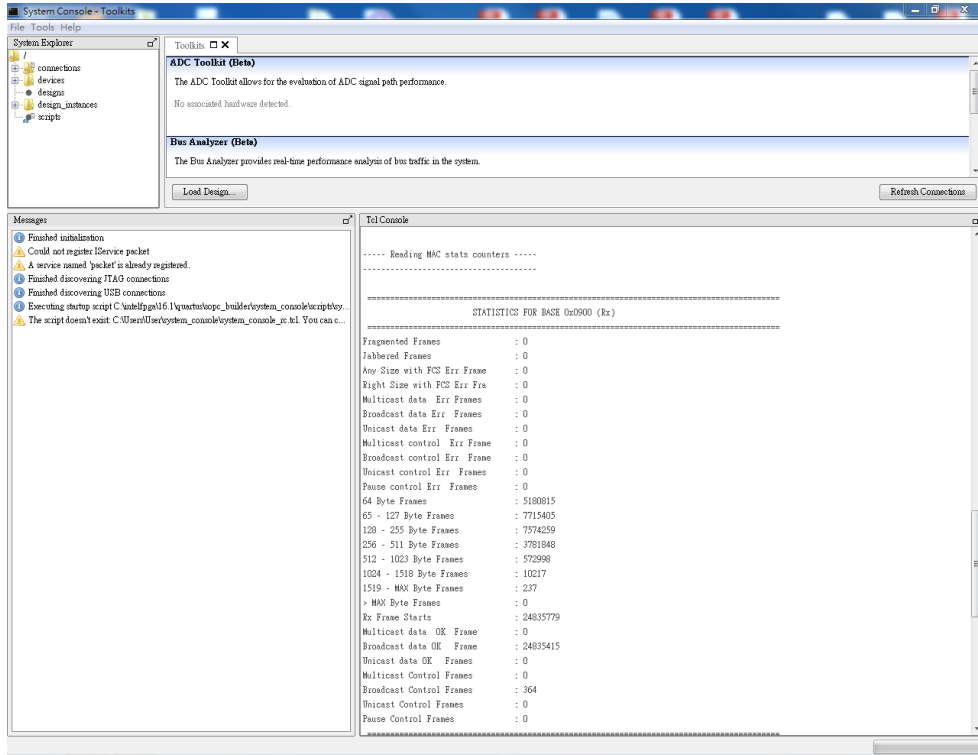


Figure 9-10 40GbE loopback test report

9.5 10GBASE-R Ethernet Example

This 10GBASE-R design example demonstrates an Ethernet solution for Arria 10 devices using the LL 10GbE MAC IP core and the native PHY IP core. This example is generated according to the **10GBASE-R Design Examples** described in the document [Low Latency Ethernet 10G MAC Design Example User Guide](#). The Arria 10 10GBASE-R IP is used in the example design. This example executes external loopback test through one of the QSFP+ ports on the FPGA main board. A QSFP+ loopback fixture is required to perform this demonstration. **Figure 9-11** shows the block diagram of this demonstration.

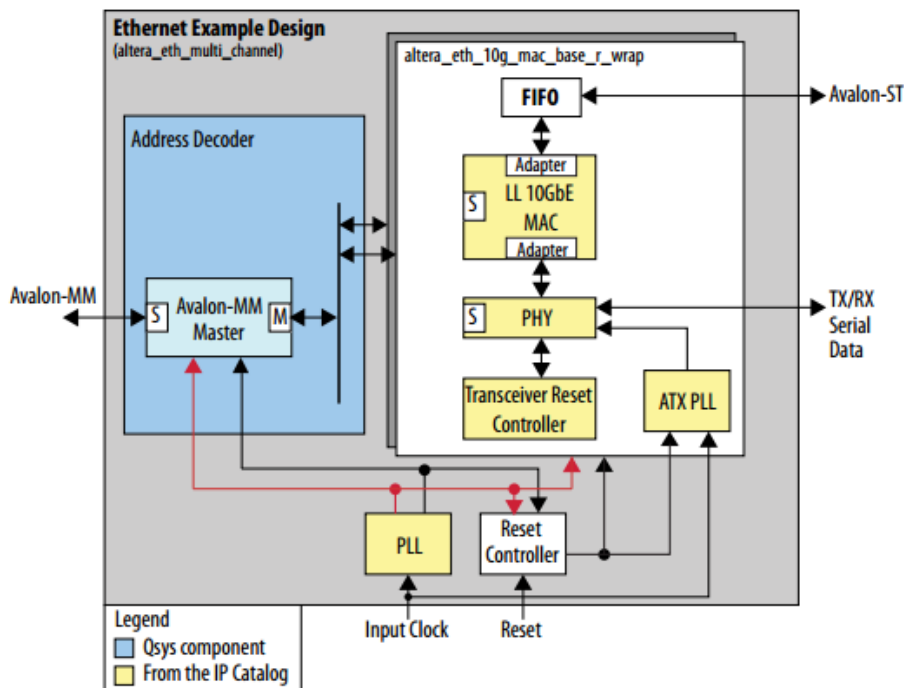


Figure 9-11 Block diagram of 10GBASE-R demo

■ Project Information

The Quartus project information is shown in the table below.

| Item | Description |
|------------------|--|
| Project Location | CDROM/Demonstrations/LL10GbE_10GBASE-R |
| FPGA Bit Stream | CDROM/Demonstrations/ LL10GbE_10GBASE-R /output_files/DE5A_NET.sof |
| Test Scrip File | CDROM/Demonstrations/ LL10GbE_10GBASE-R /hwtesting/system_console |
| Quartus Version | Quartus Prime 16.1.2 Standard Edition |

The transceiver PMA settings used in this project are shown in the table below. These settings are defined in the DE5A_NET.qdf file. They are recommended for DE5a-NET board. Developers can adjust these values based on the characteristics of the transmission cable chosen.

| Direction | Item | Value |
|-----------|-----------------------------|-------|
| TX | VOD | 28 |
| | Pre-emphasis first post-tap | -7 |
| | Pre-emphasis pre-tap | -2 |
| RX | DC Gain | 4 |
| | EQ Control | 15 |
| | VGA | 7 |

■ Demonstration Setup

Here is the procedure to setup the demonstration. A QSFP+ loopback fixture is required for this demonstration.

1. Insert a QSFP+ loopback fixture into the QSFP-A port on DE5a-NET board, as shown in **Figure 9-8** .
2. Connect the host PC to the FPGA board using a mini-USB cable. Please make sure the USB-Blaster II driver is installed on the host PC.
3. Launch Quartus Prime programmer and make sure the USB-Blaster II is detected correctly.
4. In Quartus Prime Programmer, add the configuration bit stream file

(./output_files/DE5A_NET.sof). Check the associated “Program/Configure” item and click “Start” to start FPGA programming.

5. Launch System Console by selecting the menu item “Tools→System Debugging Tools→System Console” in Quartus.
6. In the System Console window, input the following commands to start the loopback test, as shown in **Figure 9-12**. The generator generates and sends about 4 billion packets. Wait 6 minutes for it to complete its tasks.

```
%cd hwtesting/system_console
%source gen_conf.tcl
```

7. In the System Console windows, input the following command to check the good and bad packets received, as shown in **Figure 9-13**.

```
%source monitor_conf.tcl
```

8. In the System Console windows, input the following command to display the values of the statistics counters, as shown in **Figure 9-14**.

```
%source show_stats.tcl
```

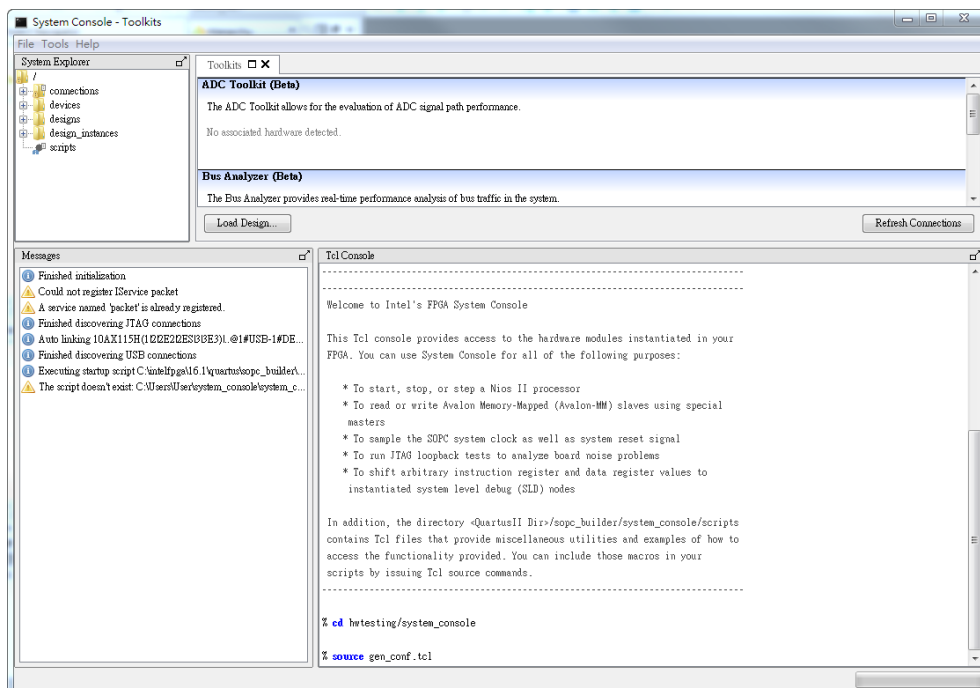


Figure 9-12 Launch the System Console and Start 10GBAE-R Demo

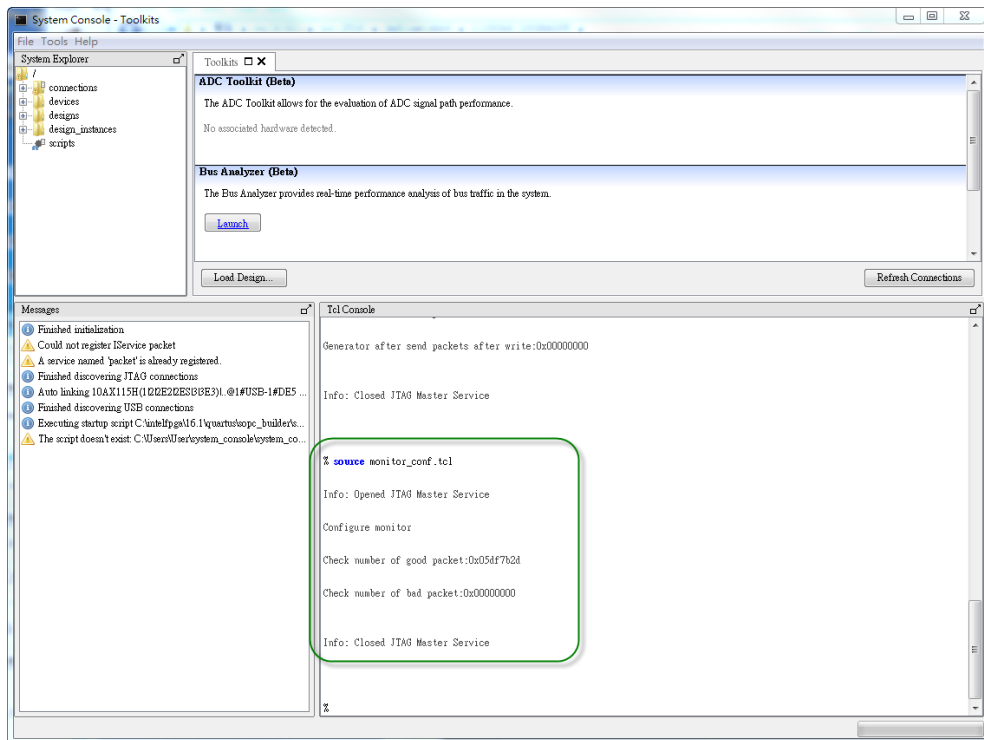


Figure 9-13 Report for monitor_conf.tcl

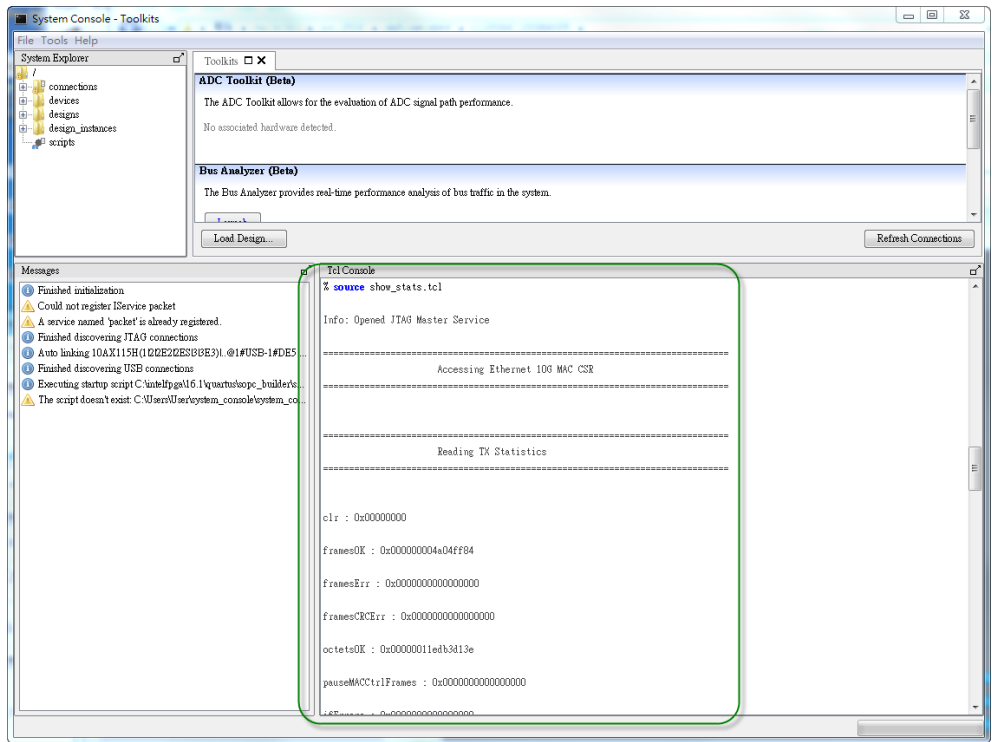


Figure 9-14 Report for show_stats.tcl

Additional Information

Getting Help

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

■ Terasic Technologies

9F., No.176, Sec.2, Gongdao 5th Rd,
East Dist, HsinChu City, Taiwan, 30070

Email: support@terasic.com

Web: www.terasic.com

DE5a-Net DDR4 Edition Web: de5a-net-ddr4.terasic.com

■ Revision History

| Date | Version | Changes |
|---------|-------------------|---|
| 2017.05 | First publication | |
| 2018.03 | V1.1 | Add board protection section |
| 2018.04 | V1.2 | Modify Figures (2-9,2-10,2-11,2-13,2-16,5-2,5-11,5-16,6-3,8-1,8-2) |
| 2018.08 | V1.3 | Modify SW4 to SW3 in section 4.2 and 4.4 |
| 2018.10 | V1.4 | Modify Figure 5-2 |
| 2019.04 | V1.5 | Modify ClockBuider Pro Software download link |
| 2019.10 | V1.6 | Modify section 7.4 |
| 2020.09 | V1.7 | Modify section 2.11 |