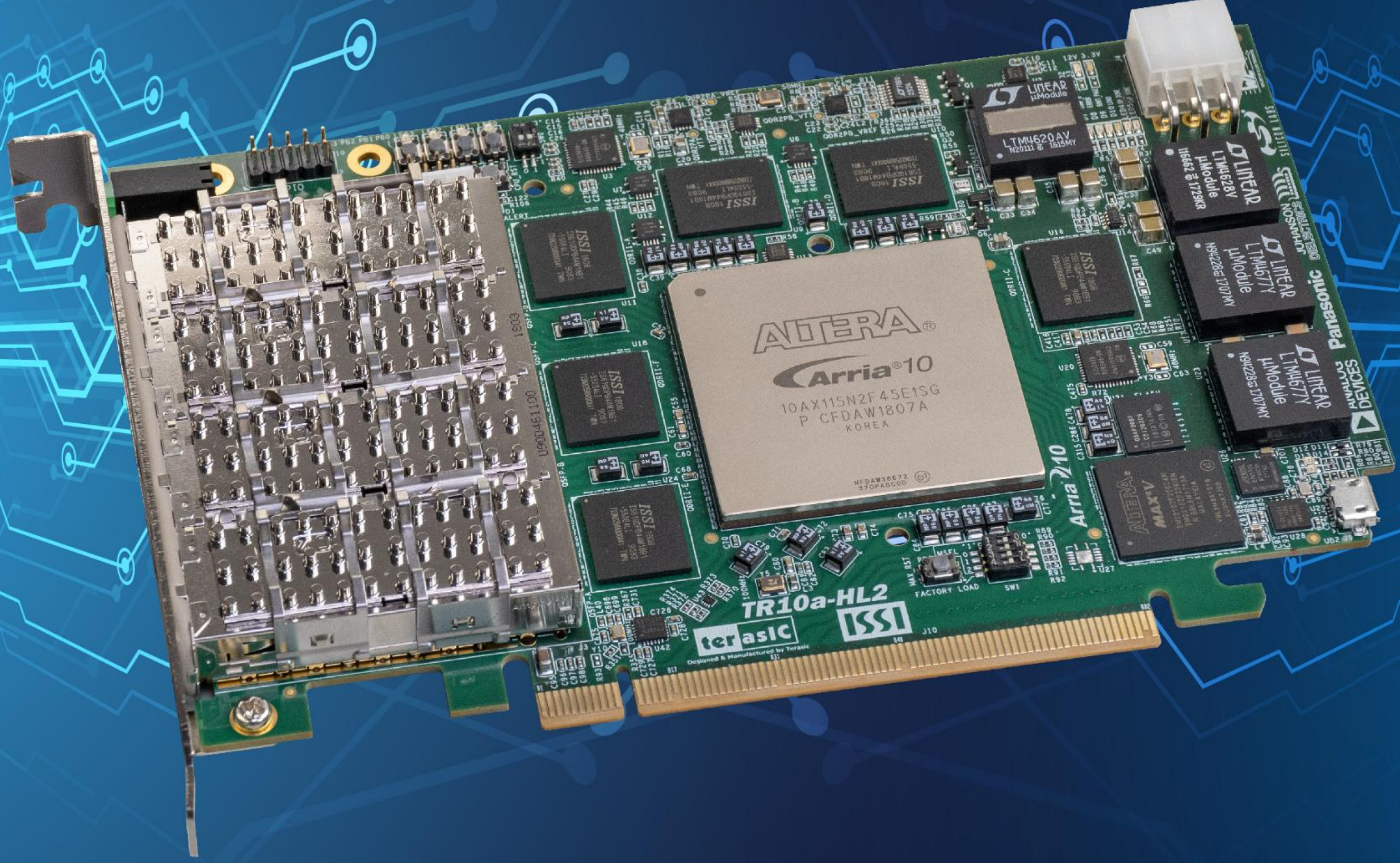
# TR10a-HL2

## FPGA Development Kit

# User Manual



www.terasic.com

Copyright © 2003-2018 Terasic Inc. All Rights Reserved.

# CONTENTS

Chapter 1

# Overview

This chapter provides an overview of the TR10a-HL2 Development Board and installation guide.

## 1.1 General Description

The Terasic TR10a-HL2 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, half length form-factor package, the TR10a-HL2 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 TR10a-HL2 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 TR10a-HL2 delivers with six independent banks of QDRII+ SRAM, high-speed parallel flash memory. The feature-set of the TR10a-HL2 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 TR10a-HL2 board:

- ### FPGA
  - Altera Arria® 10 GX FPGA (10AX115N2F45E1SG)

- ### FPGA Configuration
  - On-Board USB Blaster II or JTAG header for FPGA programming
  - Fast passive parallel (FPPx16) configuration via MAX II CPLD and flash memory

- ### General user input/output:
  - 8 LEDs
  - 4 push-buttons
  - 2 dip switches

- ### Clock System
  - 50MHz Oscillator
  - Programmable clock generators Si5340A and Si5340B

- ### Memory
  - QDRII+ SRAM
  - FLASH

- ### Communication Ports
  - Four QSFP+ connectors
  - Dual PCI Express (PCIe) x8 edge connector
  - One 2x5 GPIO timing 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 1/2-length

# 1.3 Block Diagram

**Figure 1-1** shows the block diagram of the TR10a-HL2 board. To provide maximum flexibility for the users, all key components are connected to the Arria 10 GX FPGA device. Thus, users can configure the FPGA to implement any system design.

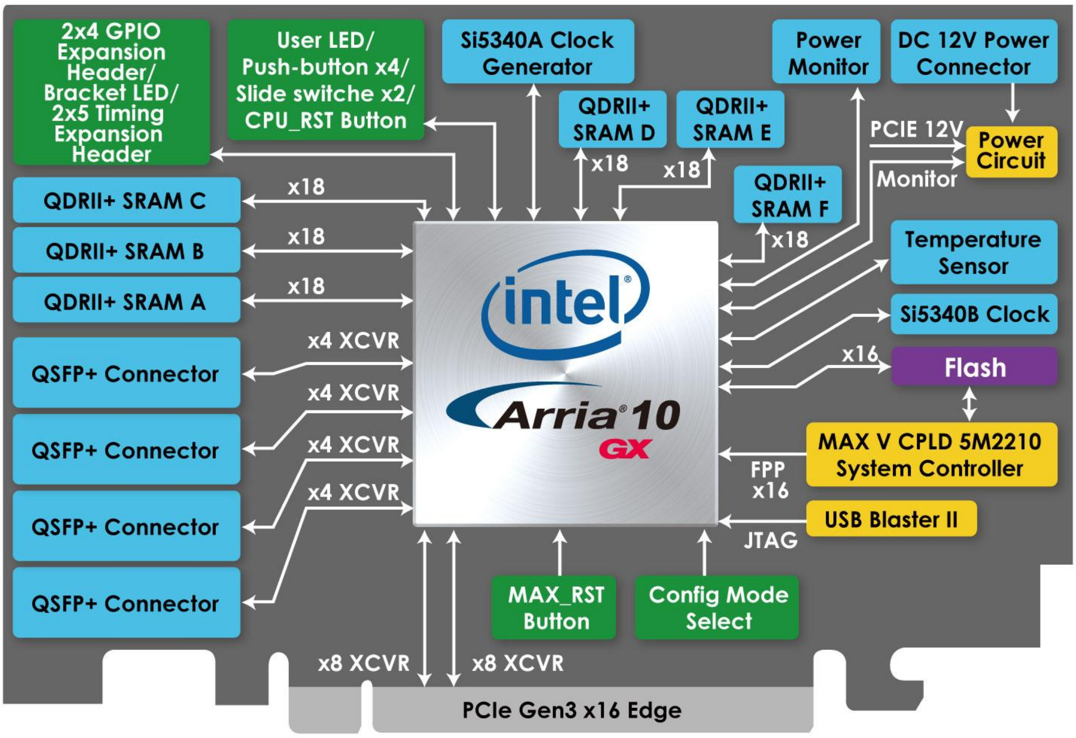



Figure 1-1 Block diagram of the TR10a-HL2 board

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)

## FPGA Configuration

- On-board USB Blaster II for use with the Quartus II Programmer
- MAXII CPLD 5M2210 System Controller and Fast Passive Parallel (FPP x16) configuration

## Memory devices

- 48MB QDRII+ SRAM
- 128MB FLASH

## General user I/O

- 8 user controllable LEDs
- 4 user push buttons
- 2 user dip switches

## 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 QDRII+ SRAM

## Four QSFP+ ports

- Four QSFP+ connector (40 Gbps+)

## Dual PCI Express x8 edge connector

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

## Power Source

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

Chapter 2

# Board Components

This chapter introduces all the important components on the TR10a-HL2.

## 2.1 Board Overview

**Figure 2-1** is the top and bottom view of the TR10a-HL2 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.

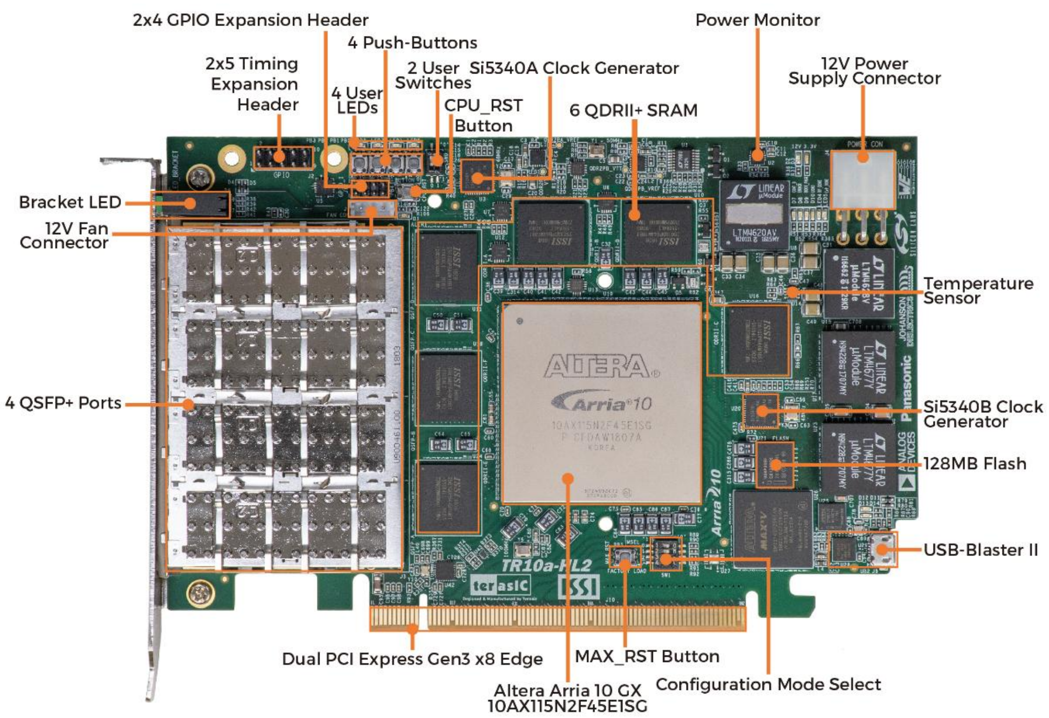



**Figure 2-1 FPGA Board (Top)**

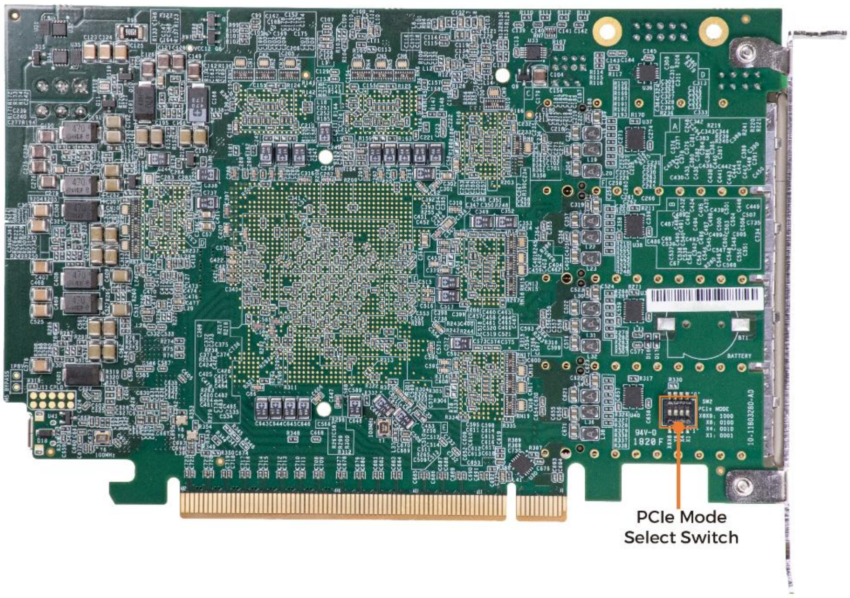



**Figure 2-2 FPGA Board (Bottom)**

# 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 micro-USB cable and make sure the USB-Blaster II driver is installed on PC.
- Launch Quartus II programmer and make sure the USB-Blaster II is detected.
- In Quartus II 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 |
|---|---|---|
| D2 | 12-V Power | Illuminates when 12-V power is active. |
| D3 | 3.3-V Power | Illuminates when 3.3-V power is active. |
| D7 | CONF DONE | Illuminates when the FPGA is successfully configured. Driven by the MAX II CPLD 5M2210 System Controller. |
| D10 | 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. |
| D8 | Error | Illuminates when the MAX II CPLD EPM2210 System Controller fails to configure the FPGA. Driven by the MAX II CPLD EPM2210 System Controller. |
| D9 | PAGE | Illuminates when FPGA is configured by the factory configuration bit stream. |

## Setup PCI Express Control DIP switch

The PCI Express Control DIP switch (SW2) is provided to enable or disable different configurations of the PCIe Connector. **Table 2-2** lists the switch controls and description.

**Table 2-2SW2 PCIe Control DIP Switch**

| Board Reference | Signal Name | Description | Default |
|---|---|---|---|
| SW2.1 | PCIE_PRSNT2n_x1 | On : Enable x1 presence detect<br>Off: Disable x1 presence detect | Off |
| SW2.2 | PCIE_PRSNT2n_x4 | On : Enable x4 presence detect<br>Off: Disable x4 presence detect | Off |
| SW2.3 | PCIE_PRSNT2n_x8 | On : Enable x8 presence detect | On |

| | | | |
|---|---|---|---|
| | | Off: Disable x8 presence detect | |

## Setup Configure Mode

The position 1~3 of DIP switch SW1 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**.

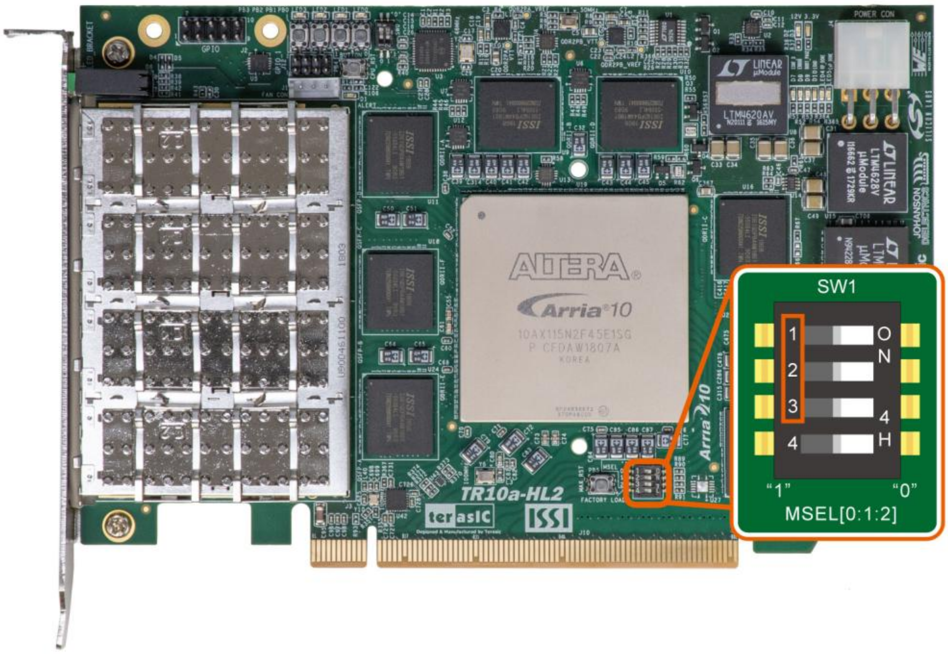



**Figure 2-3 Position of DIP switch SW1 for Configure Mode**

## Select Flash Image for Configuration

The position 4 of DIP switch SW1 is used to specify the image for configuration of the FPGA. Setting Position 4 of SW1 to "1" (down position) specifies the default factory image to be loaded, as shown in **Figure 2-4**. Setting Position 4 of SW1 to "0" (up position) specifies the TR10a-HL2 to load a user-defined image, as shown in **Figure 2-5**.

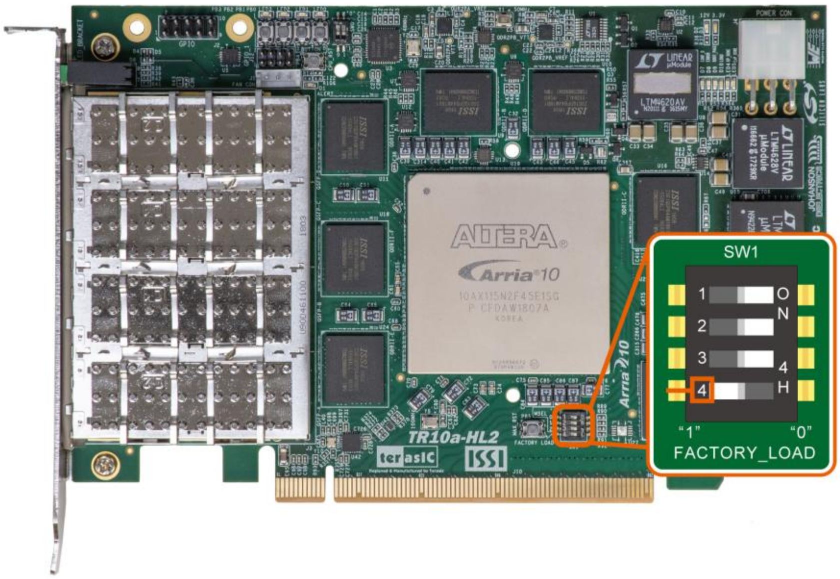



**Figure 2-4 Position of DIP switch SW1 for Image Select – Factory Image Load**

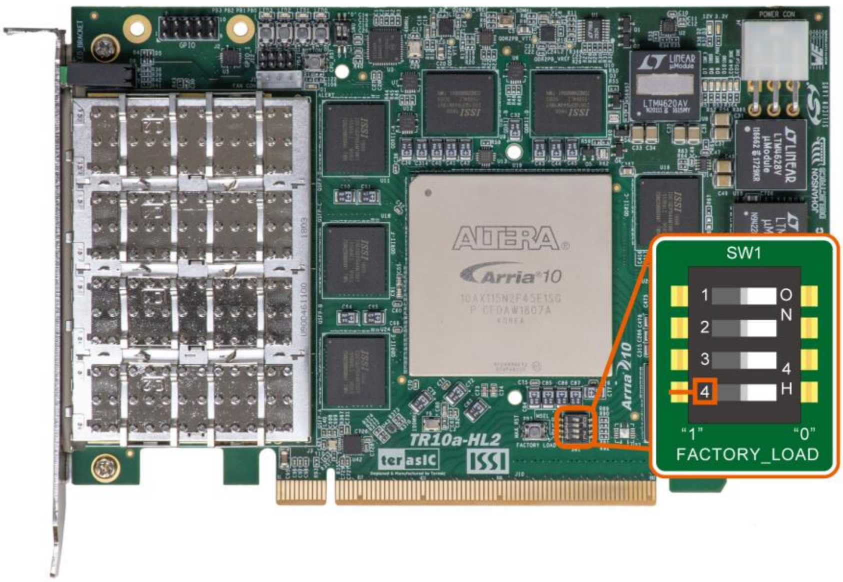



**Figure 2-5 Position of DIP switch SW1 for Image Select – User Image Load**

## 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 | 1.8-V | PIN_AC11 |
| PB1 | BUTTON1 | | 1.8-V | PIN_AC12 |
| PB2 | BUTTON2 | | 1.8-V | PIN_BA10 |
| PB3 | BUTTON3 | | 1.8-V | PIN_AP8 |

## User-Defined Dip Switch

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

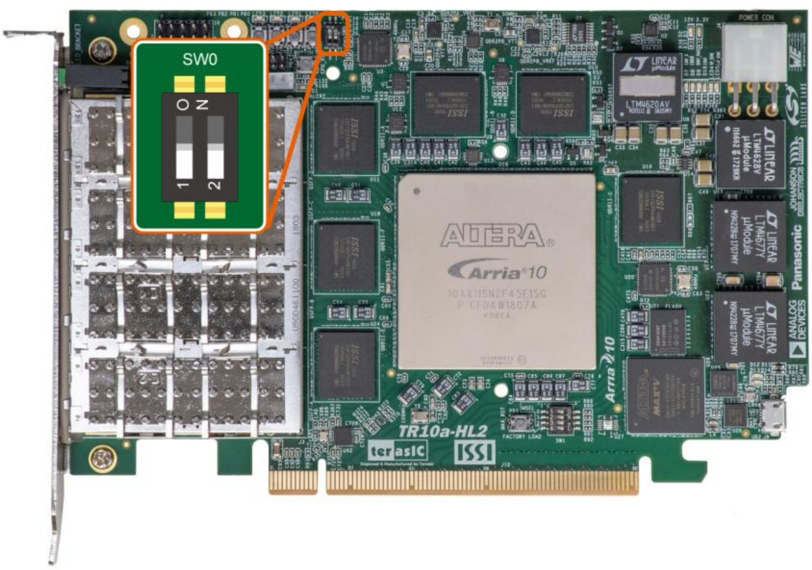



**Figure 2-6 2 Dip switches**

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

**Table 2-4 Dip 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. | 1.8-V | PIN_ BD28 |
| SW1 | SW1 | | 1.8-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 |
|---|---|---|---|---|
| LED0 | 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 |
| LED1 | LED1 | | 1.8-V | PIN_R11 |
| LED2 | LED2 | | 1.8-V | PIN_N15 |
| LED3 | LED3 | | 1.8-V | PIN_M15 |
| D6-1 | LED_BRACKET0 | | 1.8-V | PIN_BB32 |
| D6-3 | LED_BRACKET1 | | 1.8-V | PIN_AW30 |
| D6-5 | LED_BRACKET2 | | 1.8-V | PIN_AV30 |
| D6-7 | LED_BRACKET3 | | 1.8-V | PIN_AM30 |

# 2.4 Temperature Sensor and Fan Control

The FPGA board is equipped with a temperature sensor, TMP441, which provides temperature sensing. 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 holding configuration information 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 '0011100b'.**Figure 2-7** shows the connection between the temperature sensor and the Arria 10 GX FPGA.

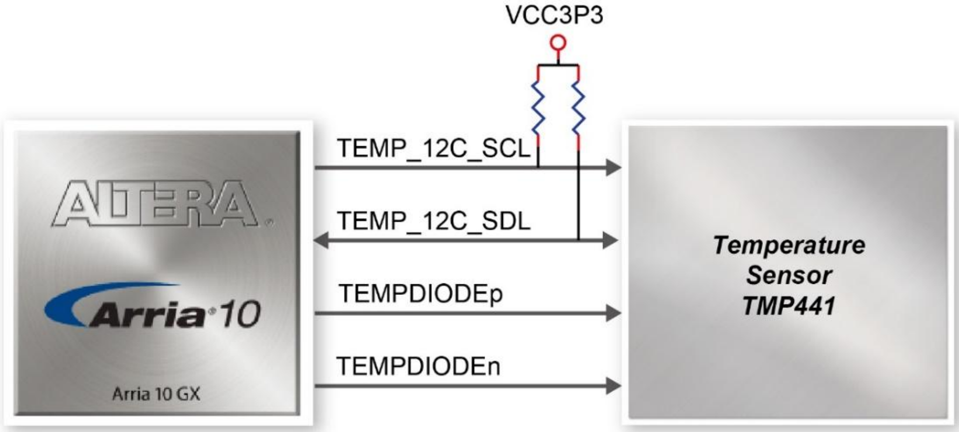



**Figure 2-7 Connections between the temperature sensor and the Arria 10 GX FPGA**

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. **Figure 2-8** shows the connection between the Fan-Speed Regulator and Monitor and the Arria 10 GX FPGA.

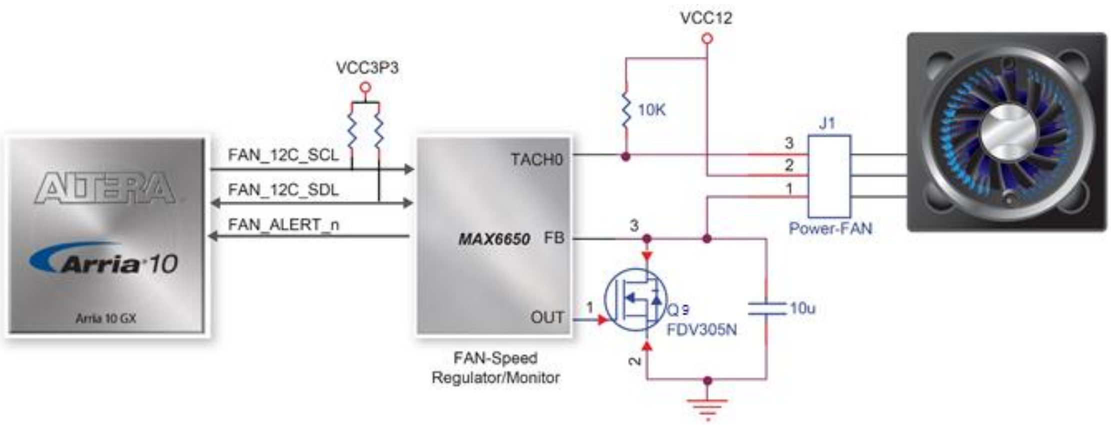



**Figure 2-8 Connections between the Fan-Speed Regulator/ Monitor and the Arria 10 GX FPGA**

The pin assignments for the associated interface are listed in**Table 2-6**.

**Table 2-6 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 | 1.8-V | PIN_AU12 |
| TEMP_I2C_SDA | SMBus data | 1.8-V | PIN_AV12 |
| FAN_I2C_SCL | 2-Wire Serial Clock | 1.8-V | PIN_AJ33 |
| FAN_I2C_SDA | 2-Wire Serial-Data | 1.8-V | PIN_AK33 |
| FAN_ALERT_n | Active-low AL ERT input | 1.8-V | PIN_AL32 |

## 2.5 Power Monitor

The TR10a-HL2 has implemented a power monitor chip to monitor the board input power voltage and current. **Figure 2-9** 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 2-7** shows the pin assignment of power monitor I2C bus.

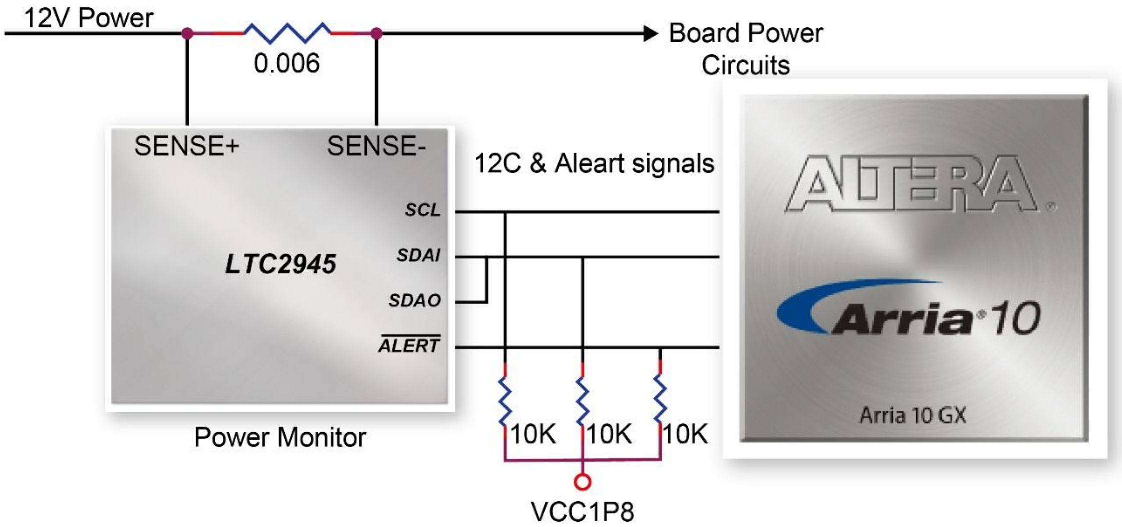


**Figure 2-9 Connections between the Power Monitor chip and the Arria 10 GX FPGA**

**Table 2-7 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_N | Power Monitor ALERT | 1.8V | PIN_BD23 |

## 2.6 Clock Circuit

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

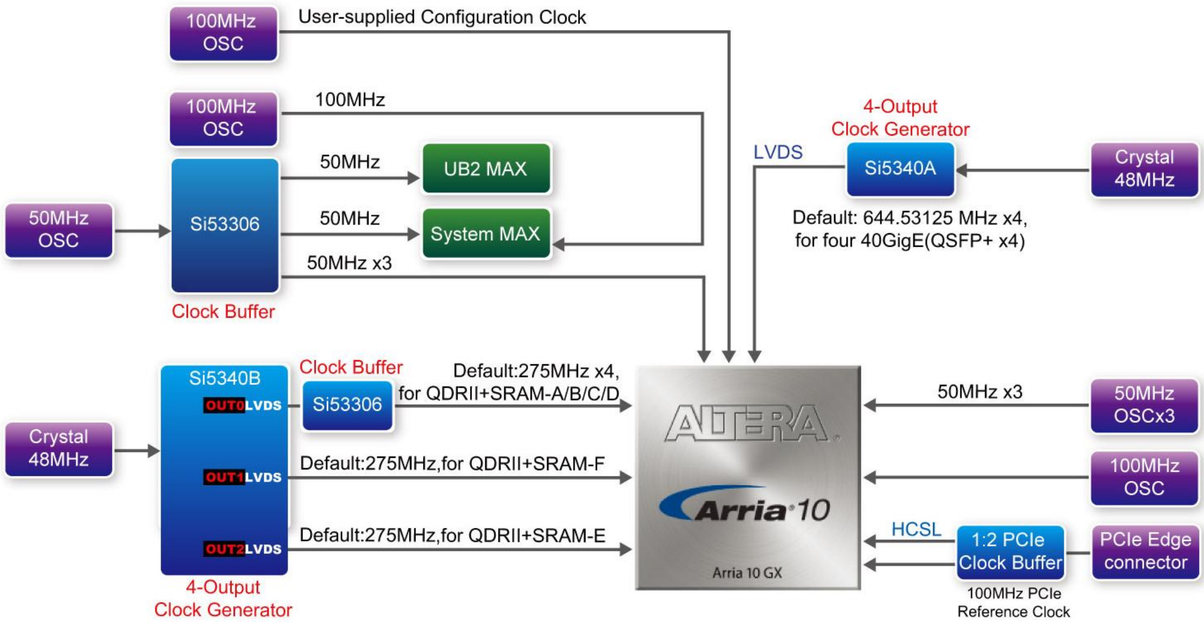



**Figure 2-10 Clock circuit of the FPGA Board**

A clock buffer is used to duplicate the 50 MHz oscillator, so there are six 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+ and high bandwidth memory reference clocks respectively.

**Table 2-8** lists the clock source, signal names, default frequency and their corresponding Arria 10 GX device pin numbers.

**Table 2-8 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 |
|---|---|---|---|---|---|
| Y8 | CLK_50_B2H | 50.0 MHz | 1.8V | PIN_AP34 | |
| Y9 | CLK_50_B2G | | 1.8V | PIN_AW35 | |
| Y10 | CLK_50_B2F | | 1.8V | PIN_AY31 | |
| Y1 | CLK_50_B3D | | 1.8V | PIN_AN7 | |
| | CLK_50_B3F | | 1.8V | PIN_G12 | |
| | CLK_50_B3H | | 1.8V | PIN_D21 | |
| Y5 | CLK_100_B3D | 100.0MHz | 1.8V | PIN_AJ11 | |
| Y7 | OSC_100_CLKUSR | 100.0MHz | 1.8V | PIN_AV26 | User-supplied configuration clock |
| U3 | 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 |
| | QSFPC_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 |
| U20 | 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 |
| | QDRIIE_REFCLK_p | 275 MHz | LVDS | PIN_AP14 | QDRII+ reference clock for E port |
| | QDRIIF_REFCLK_p | 275 MHz | LVDS | PIN_AT7 | QDRII+ reference clock for F port |

**Table 2-9** lists the programmable oscillator control pins, signal names, I/O standard and their corresponding Arria 10 GX device pin numbers.

**Table 2-9 Programmable oscillator control pin, Signal Name, I/O standard, Pin Assignments and Descriptions**

<table>
<tr><th>Programmable Oscillator</th><th>Schematic Signal Name</th><th>I/O Standard</th><th>Arria 10 GX Pin Number</th><th>Description</th></tr>
<tr><td rowspan="5">Si5340A (U3)</td><td>Si5340A_I2C_SCL</td><td>1.8-V</td><td>PIN_AU27</td><td rowspan="2">I2C bus, connected with Si5340A</td></tr>
<tr><td>Si5340A_I2C_SDA</td><td>1.8-V</td><td>PIN_AT27</td></tr>
<tr><td>Si5340A_RST</td><td>1.8-V</td><td>PIN_AW28</td><td>Si5340A reset signal</td></tr>
<tr><td>Si5340A_INTR</td><td>1.8-V</td><td>PIN_AW29</td><td>Si5340A interrupt signal</td></tr>
<tr><td>Si5340A_OE_n</td><td>1.8-V</td><td>PIN_AV28</td><td>Si5340A output enable signal</td></tr>
<tr><td rowspan="5">Si5340B (U20)</td><td>Si5340B_I2C_SCL</td><td>1.8-V</td><td>PIN_G37</td><td rowspan="2">I2C bus, connected with Si5340B</td></tr>
<tr><td>Si5340B_I2C_SDA</td><td>1.8-V</td><td>PIN_H31</td></tr>
<tr><td>Si5340B_RST</td><td>1.8-V</td><td>PIN_G38</td><td>Si5340B reset signal</td></tr>
<tr><td>Si5340B_INTR</td><td>1.8-V</td><td>PIN_G32</td><td>Si5340B interrupt signal</td></tr>
<tr><td>Si5340B_OE_n</td><td>1.8-V</td><td>PIN_AL31</td><td>Si5340B output enable signal</td></tr>
</table>

## 2.7 FLASH Memory

The development board has one 1Gb CFI-compatible synchronous flash device 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 device combined allow for FPP x16 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 2-11** shows the connections between the Flash, MAX and Arria 10 GX FPGA.

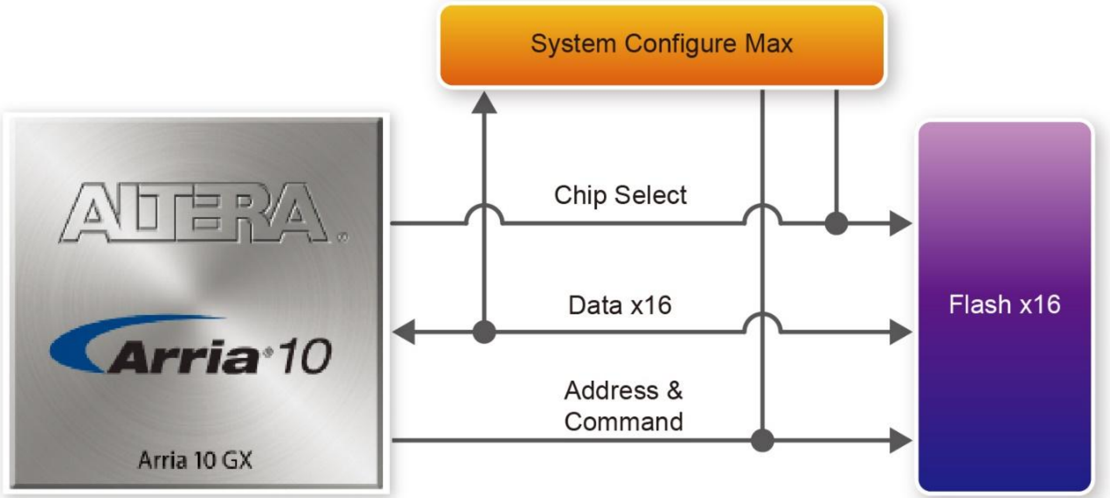



**Figure 2-11 Connection between the Flash, Max and Arria 10 GX FPGA**

**Table 2-10** lists the flash pin assignments, signal names, and functions.

**Table 2-10 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_T12 |
| FLASH_A3 | Address bus | 1.8-V | PIN_U12 |
| FLASH_A4 | Address bus | 1.8-V | PIN_U10 |
| FLASH_A5 | Address bus | 1.8-V | PIN_A15 |
| FLASH_A6 | Address bus | 1.8-V | PIN_H6 |
| FLASH_A7 | Address bus | 1.8-V | PIN_B18 |
| FLASH_A8 | Address bus | 1.8-V | PIN_P15 |
| FLASH_A9 | Address bus | 1.8-V | PIN_H7 |
| FLASH_A10 | Address bus | 1.8-V | PIN_A17 |
| FLASH_A11 | Address bus | 1.8-V | PIN_A16 |
| FLASH_A12 | Address bus | 1.8-V | PNI_C16 |
| FLASH_A13 | Address bus | 1.8-V | PIN_K11 |
| FLASH_A14 | Address bus | 1.8-V | PIN_H8 |
| FLASH_A15 | Address bus | 1.8-V | PIN_F6 |
| FLASH_A16 | Address bus | 1.8-V | PIN_C17 |
| FLASH_A17 | Address bus | 1.8-V | PIN_G8 |

| | | | |
|---|---|---|---|
| FLASH_A18 | Address bus | 1.8-V | PIN_J10 |
| FLASH_A19 | Address bus | 1.8-V | PIN_L36 |
| FLASH_A20 | Address bus | 1.8-V | PIN_G7 |
| FLASH_A21 | Address bus | 1.8-V | PIN_G9 |
| FLASH_A22 | Address bus | 1.8-V | PIN_J18 |
| FLASH_A23 | Address bus | 1.8-V | PIN_A14 |
| FLASH_A24 | Address bus | 1.8-V | PIN_B15 |
| FLASH_A25 | Address bus | 1.8-V | PIN_J11 |
| FLASH_A26 | Address bus | 1.8-V | PIN_H10 |
| FLASH_D0 | Address bus | 1.8-V | PIN_Y31 |
| FLASH_D1 | Data bus | 1.8-V | PIN_E24 |
| FLASH_D2 | Data bus | 1.8-V | PIN_H35 |
| FLASH_D3 | Data bus | 1.8-V | PIN_J39 |
| FLASH_D4 | Data bus | 1.8-V | PIN_H38 |
| FLASH_D5 | Data bus | 1.8-V | PIN_C33 |
| FLASH_D6 | Data bus | 1.8-V | PIN_C32 |
| FLASH_D7 | Data bus | 1.8-V | PIN_C26 |
| FLASH_D8 | Data bus | 1.8-V | PIN_B24 |
| FLASH_D9 | Data bus | 1.8-V | PIN_AA31 |
| FLASH_D10 | Data bus | 1.8-V | PIN_J33 |
| FLASH_D11 | Data bus | 1.8-V | PIN_C36 |
| FLASH_D12 | Data bus | 1.8-V | PIN_C35 |
| FLASH_D13 | Data bus | 1.8-V | PIN_C25 |
| FLASH_D14 | Data bus | 1.8-V | PIN_H37 |
| FLASH_D15 | Data bus | 1.8-V | PIN_J38 |
| FLASH_CLK | Clock | 1.8-V | PIN_T9 |
| FLASH_RESET_n | Reset | 1.8-V | PIN_B14 |
| FLASH_CE_n | Chip enable of flash | 1.8-V | PIN_J8 |
| FLASH_OE_n | Output enable | 1.8-V | PIN_E18 |
| FLASH_WE_n | Write enable | 1.8-V | PIN_B17 |
| FLASH_ADV_n | Address valid | 1.8-V | PIN_F7 |
| FLASH_RDY_BSY_n | Ready of flash | 1.8-V | PIN_D18 |

## 2.8 QDRII+ SRAM

The development board supports six 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 2-11**, **Table 2-12**,**Table 2-13**, **Table 2-14**, **Table 2-15** and **Table 2-16** lists the QDRII+ SRAM Bank A, B, C and D pin assignments, signal names relative to the Arria 10 GX device, in respectively.

**Table 2-11 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_D13 |
| QDRIIA_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_C10 |
| QDRIIA_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_B10 |
| QDRIIA_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_A10 |
| QDRIIA_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_C11 |
| QDRIIA_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_C12 |
| QDRIIA_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_A11 |
| QDRIIA_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_B12 |
| QDRIIA_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_A12 |
| QDRIIA_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_D11 |
| QDRIIA_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_D10 |
| QDRIIA_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_C8 |
| QDRIIA_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_D9 |
| QDRIIA_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_D8 |
| QDRIIA_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_E13 |
| QDRIIA_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_E9 |
| QDRIIA_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_E11 |
| QDRIIA_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_E8 |
| QDRIIA_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_P13 |
| QDRIIA_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_R13 |
| QDRIIA_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_N13 |
| QDRIIA_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_M14 |
| 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_K13 |
| QDRIIA_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_K12 |
| QDRIIA_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_K14 |
| QDRIIA_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_J14 |
| 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_G10 |
| QDRIIA_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_L14 |
| QDRIIA_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_L12 |
| QDRIIA_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_M13 |
| QDRIIA_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_N12 |
| QDRIIA_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_R14 |

| | | | |
|---|---|---|---|
| QDRIIA_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_T14 |
| QDRIIA_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_B13 |
| QDRIIA_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_C13 |
| 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_R12 |

**Table 2-12 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_N20 |
| QDRIIB_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_M19 |
| QDRIIB_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_L19 |
| QDRIIB_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_J19 |
| QDRIIB_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_J20 |
| QDRIIB_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_F19 |
| QDRIIB_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_B19 |
| QDRIIB_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_F20 |
| QDRIIB_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_G20 |
| QDRIIB_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_C20 |
| QDRIIB_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_B20 |
| QDRIIB_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_D19 |
| QDRIIB_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_E19 |
| QDRIIB_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_C18 |
| QDRIIB_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_G19 |
| QDRIIB_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_K19 |
| QDRIIB_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_L20 |
| QDRIIB_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_M20 |
| 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_K22 |
| QDRIIB_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_K23 |
| QDRIIB_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_J23 |
| 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_H23 |
| QDRIIB_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_F22 |
| 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_B23 |
| QDRIIB_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_A22 |
| QDRIIB_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_B22 |

| | | | |
|---|---|---|---|
| QDRIIB_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_C22 |
| QDRIIB_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_C21 |
| QDRIIB_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_E22 |
| QDRIIB_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_A21 |
| QDRIIB_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_F21 |
| QDRIIB_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_G23 |
| QDRIIB_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_H20 |
| QDRIIB_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_L21 |
| QDRIIB_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_K21 |
| QDRIIB_K_n | Clock N | Differential 1.8-V HSTL Class I | PIN_J21 |
| 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_G22 |

**Table 2-13 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_AF36 |
| QDRIIC_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_AF32 |
| QDRIIC_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_AE34 |
| QDRIIC_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_AB35 |
| QDRIIC_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_AA35 |
| QDRIIC_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_AA34 |
| QDRIIC_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_Y34 |
| QDRIIC_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_AB33 |
| QDRIIC_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_AC33 |
| QDRIIC_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_AE31 |
| QDRIIC_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_AE33 |
| 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_AB34 |
| QDRIIC_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_W34 |
| QDRIIC_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_AC35 |
| QDRIIC_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_AD35 |
| QDRIIC_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_AE36 |
| 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_Y36 |
| QDRIIC_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_U34 |
| QDRIIC_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_T34 |
| QDRIIC_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_T35 |
| QDRIIC_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_P35 |
| QDRIIC_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_P36 |
| QDRIIC_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_N35 |

| | | | |
|---|---|---|---|
| QDRIIC_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_N37 |
| QDRIIC_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_N38 |
| QDRIIC_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_M35 |
| 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_N36 |
| QDRIIC_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_M38 |
| QDRIIC_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_M39 |
| QDRIIC_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_R36 |
| QDRIIC_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_T36 |
| QDRIIC_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_U35 |
| QDRIIC_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_V35 |
| QDRIIC_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_AD33 |
| QDRIIC_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_AE32 |
| 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_W35 |

**Table 2-14 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_D36 |
| 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_D34 |
| QDRIID_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_D35 |
| QDRIID_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_E34 |
| QDRIID_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_E33 |
| QDRIID_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_D33 |
| QDRIID_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_F31 |
| QDRIID_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_E31 |
| QDRIID_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_F39 |
| QDRIID_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_E37 |
| QDRIID_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_G39 |
| QDRIID_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_F36 |
| QDRIID_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_E36 |
| QDRIID_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_H30 |
| QDRIID_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_F30 |
| QDRIID_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_G30 |
| QDRIID_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_G29 |
| QDRIID_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_N28 |
| QDRIID_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_N31 |

| | | | |
|---|---|---|---|
| QDRIID_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_M28 |
| QDRIID_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_M30 |
| QDRIID_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_K29 |
| 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_K31 |
| QDRIID_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_G33 |
| QDRIID_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_G34 |
| QDRIID_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_H33 |
| QDRIID_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_J31 |
| 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_L30 |
| QDRIID_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_J29 |
| QDRIID_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_L29 |
| 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_N30 |
| QDRIID_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_P28 |
| QDRIID_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_C37 |
| QDRIID_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_F37 |
| 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_P31 |

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

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|---|---|---|---|
| QDRIIE_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_BB9 |
| QDRIIE_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_BB8 |

| | | | |
|---|---|---|---|
| QDRIIE_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_AW15 |
| QDRIIE_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_AW14 |
| QDRIIE_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_AW13 |
| QDRIIE_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_AY13 |
| QDRIIE_A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_AY14 |
| QDRIIE_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_BA14 |
| QDRIIE_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_BA12 |
| QDRIIE_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_BB12 |
| QDRIIE_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_AU13 |
| QDRIIE_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_AV13 |
| QDRIIE_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_AY11 |
| QDRIIE_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_BA11 |
| QDRIIE_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_AK14 |
| QDRIIE_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_AM13 |
| QDRIIE_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_AN13 |
| QDRIIE_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_AL14 |
| QDRIIE_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_AM14 |
| QDRIIE_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_AT14 |
| QDRIIE_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_AU14 |
| QDRIIE_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_AP13 |
| QDRIIE_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_BD18 |
| QDRIIE_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_BC18 |
| QDRIIE_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_AM20 |
| QDRIIE_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_BC17 |
| QDRIIE_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_AM18 |
| QDRIIE_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_BD15 |
| QDRIIE_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_BB15 |
| QDRIIE_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_AV18 |
| QDRIIE_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_AU18 |
| QDRIIE_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_AT19 |
| QDRIIE_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_AU19 |
| QDRIIE_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_AR19 |
| QDRIIE_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_BD13 |
| QDRIIE_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_BD14 |
| QDRIIE_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_BC15 |
| QDRIIE_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_AP19 |

| | | | |
|---|---|---|---|
| QDRIIE_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_BC16 |
| QDRIIE_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_BD16 |
| QDRIIE_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_AN17 |
| QDRIIE_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_AT17 |
| QDRIIE_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_AU17 |
| QDRIIE_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_BA16 |
| QDRIIE_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_AT16 |
| QDRIIE_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_AT15 |
| QDRIIE_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_AP16 |
| QDRIIE_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_AP15 |
| QDRIIE_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_AN15 |
| QDRIIE_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_AM15 |
| QDRIIE_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_AN16 |
| QDRIIE_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_AR16 |
| QDRIIE_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_AU15 |
| QDRIIE_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_AV15 |
| QDRIIE_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_BA15 |
| QDRIIE_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_AW16 |
| QDRIIE_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_AY16 |
| QDRIIE_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_AY17 |
| QDRIIE_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_AM17 |
| QDRIIE_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_AM19 |
| QDRIIE_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_AP18 |
| QDRIIE_K_n | Clock N | Differential 1.8-V HSTL Class I | PIN_AR18 |
| QDRIIE_CQ_p | Echo clock P | 1.8-V HSTL Class I | PIN_AV16 |
| QDRIIE_CQ_n | Echo clock N | 1.8-V HSTL Class I | PIN_AV17 |
| QDRIIE_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_BD10 |
| QDRIIE_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_BC10 |
| QDRIIE_DOFF_n | PLL Turn Off | 1.8-V HSTL Class I | PIN_BD11 |
| QDRIIE_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_BB13 |
| QDRIIE_QVLD | ValidOutput Indicator | 1.8-V HSTL Class I | PIN_AR17 |

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

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|---|---|---|---|
| QDRIIF_A0 | Address bus[0] | 1.8-V HSTL Class I | PIN_AG14 |
| QDRIIF_A1 | Address bus[1] | 1.8-V HSTL Class I | PIN_AF14 |
| QDRIIF_A2 | Address bus[2] | 1.8-V HSTL Class I | PIN_AJ13 |
| QDRIIF_A3 | Address bus[3] | 1.8-V HSTL Class I | PIN_AK13 |
| QDRIIF_A4 | Address bus[4] | 1.8-V HSTL Class I | PIN_AH13 |
| QDRIIF_A5 | Address bus[5] | 1.8-V HSTL Class I | PIN_AG13 |
| QDRIIF _A6 | Address bus[6] | 1.8-V HSTL Class I | PIN_AG12 |
| QDRIIF_A7 | Address bus[7] | 1.8-V HSTL Class I | PIN_AH12 |
| QDRIIF_A8 | Address bus[8] | 1.8-V HSTL Class I | PIN_AM12 |
| QDRIIF_A9 | Address bus[9] | 1.8-V HSTL Class I | PIN_AN12 |
| QDRIIF_A10 | Address bus[10] | 1.8-V HSTL Class I | PIN_AE12 |
| QDRIIF_A11 | Address bus[11] | 1.8-V HSTL Class I | PIN_AF12 |
| QDRIIF_A12 | Address bus[12] | 1.8-V HSTL Class I | PIN_AK12 |
| QDRIIF_A13 | Address bus[13] | 1.8-V HSTL Class I | PIN_AL12 |
| QDRIIF_A14 | Address bus[14] | 1.8-V HSTL Class I | PIN_AT9 |
| QDRIIF_A15 | Address bus[15] | 1.8-V HSTL Class I | PIN_AV7 |
| QDRIIF_A16 | Address bus[16] | 1.8-V HSTL Class I | PIN_AV6 |
| QDRIIF_A17 | Address bus[17] | 1.8-V HSTL Class I | PIN_AU9 |
| QDRIIF_A18 | Address bus[18] | 1.8-V HSTL Class I | PIN_AV8 |
| QDRIIF_A19 | Address bus[19] | 1.8-V HSTL Class I | PIN_AU8 |
| QDRIIF_A20 | Address bus[20] | 1.8-V HSTL Class I | PIN_AU7 |
| QDRIIF_A21 | Address bus[21] | 1.8-V HSTL Class I | PIN_AP10 |
| QDRIIF_D0 | Write data bus[0] | 1.8-V HSTL Class I | PIN_AJ9 |
| QDRIIF_D1 | Write data bus[1] | 1.8-V HSTL Class I | PIN_AJ10 |
| QDRIIF_D2 | Write data bus[2] | 1.8-V HSTL Class I | PIN_AH10 |
| QDRIIF_D3 | Write data bus[3] | 1.8-V HSTL Class I | PIN_AG10 |
| QDRIIF_D4 | Write data bus[4] | 1.8-V HSTL Class I | PIN_AG9 |
| QDRIIF_D5 | Write data bus[5] | 1.8-V HSTL Class I | PIN_AF10 |
| QDRIIF_D6 | Write data bus[6] | 1.8-V HSTL Class I | PIN_AD10 |
| QDRIIF_D7 | Write data bus[7] | 1.8-V HSTL Class I | PIN_AD11 |
| QDRIIF_D8 | Write data bus[8] | 1.8-V HSTL Class I | PIN_AC10 |
| QDRIIF_D9 | Write data bus[9] | 1.8-V HSTL Class I | PIN_AA9 |
| QDRIIF_D10 | Write data bus[10] | 1.8-V HSTL Class I | PIN_AA10 |
| QDRIIF_D11 | Write data bus[11] | 1.8-V HSTL Class I | PIN_Y9 |

| | | | |
|---|---|---|---|
| QDRIIF_D12 | Write data bus[12] | 1.8-V HSTL Class I | PIN_AE11 |
| QDRIIF_D13 | Write data bus[13] | 1.8-V HSTL Class I | PIN_AF9 |
| QDRIIF_D14 | Write data bus[14] | 1.8-V HSTL Class I | PIN_AF11 |
| QDRIIF_D15 | Write data bus[15] | 1.8-V HSTL Class I | PIN_AK9 |
| QDRIIF_D16 | Write data bus[16] | 1.8-V HSTL Class I | PIN_AK11 |
| QDRIIF_D17 | Write data bus[17] | 1.8-V HSTL Class I | PIN_AL11 |
| QDRIIF_Q0 | Read Data bus[0] | 1.8-V HSTL Class I | PIN_AP6 |
| QDRIIF_Q1 | Read Data bus[1] | 1.8-V HSTL Class I | PIN_AR6 |
| QDRIIF_Q2 | Read Data bus[2] | 1.8-V HSTL Class I | PIN_AM10 |
| QDRIIF_Q3 | Read Data bus[3] | 1.8-V HSTL Class I | PIN_AA12 |
| QDRIIF_Q4 | Read Data bus[4] | 1.8-V HSTL Class I | PIN_AA11 |
| QDRIIF_Q5 | Read Data bus[5] | 1.8-V HSTL Class I | PIN_Y11 |
| QDRIIF_Q6 | Read Data bus[6] | 1.8-V HSTL Class I | PIN_Y12 |
| QDRIIF_Q7 | Read Data bus[7] | 1.8-V HSTL Class I | PIN_W13 |
| QDRIIF_Q8 | Read Data bus[8] | 1.8-V HSTL Class I | PIN_W14 |
| QDRIIF_Q9 | Read Data bus[9] | 1.8-V HSTL Class I | PIN_W11 |
| QDRIIF_Q10 | Read Data bus[10] | 1.8-V HSTL Class I | PIN_V10 |
| QDRIIF_Q11 | Read Data bus[11] | 1.8-V HSTL Class I | PIN_W10 |
| QDRIIF_Q12 | Read Data bus[12] | 1.8-V HSTL Class I | PIN_W9 |
| QDRIIF_Q13 | Read Data bus[13] | 1.8-V HSTL Class I | PIN_Y13 |
| QDRIIF_Q14 | Read Data bus[14] | 1.8-V HSTL Class I | PIN_Y14 |
| QDRIIF_Q15 | Read Data bus[15] | 1.8-V HSTL Class I | PIN_AL10 |
| QDRIIF_Q16 | Read Data bus[16] | 1.8-V HSTL Class I | PIN_AM9 |
| QDRIIF_Q17 | Read Data bus[17] | 1.8-V HSTL Class I | PIN_AN6 |
| QDRIIF_BWS_n0 | Byte Write select[0] | 1.8-V HSTL Class I | PIN_AB10 |
| QDRIIF_BWS_n1 | Byte Write select[1] | 1.8-V HSTL Class I | PIN_AB9 |
| QDRIIF_K_p | Clock P | Differential 1.8-V HSTL Class I | PIN_AE9 |
| QDRIIF_K_n | Clock N | Differential 1.8-V HSTL Class I | PIN_AD9 |
| QDRIIF_CQ_p | Echo clock P | 1.8-V HSTL Class I | PIN_AM8 |
| QDRIIF_CQ_n | Echo clock N | 1.8-V HSTL Class I | PIN_AM7 |
| QDRIIF_RPS_n | Report Select | 1.8-V HSTL Class I | PIN_AB13 |
| QDRIIF_WPS_n | Write Port Select | 1.8-V HSTL Class I | PIN_AB14 |
| QDRIIF_DOFF_n | PLL Turn Off | 1.8-V HSTL Class I | PIN_AB12 |
| QDRIIF_ODT | On-Die Termination Input | 1.8-V HSTL Class I | PIN_AE14 |

| | | | |
|---|---|---|---|
| QDRIIF_QVLD | ValidOutput Indicator | 1.8-V HSTL Class I | PIN_AL9 |

## 2.9 QSPF+ 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 2-12** shows the connections between the QSFP+ and Arria 10 GX FPGA.

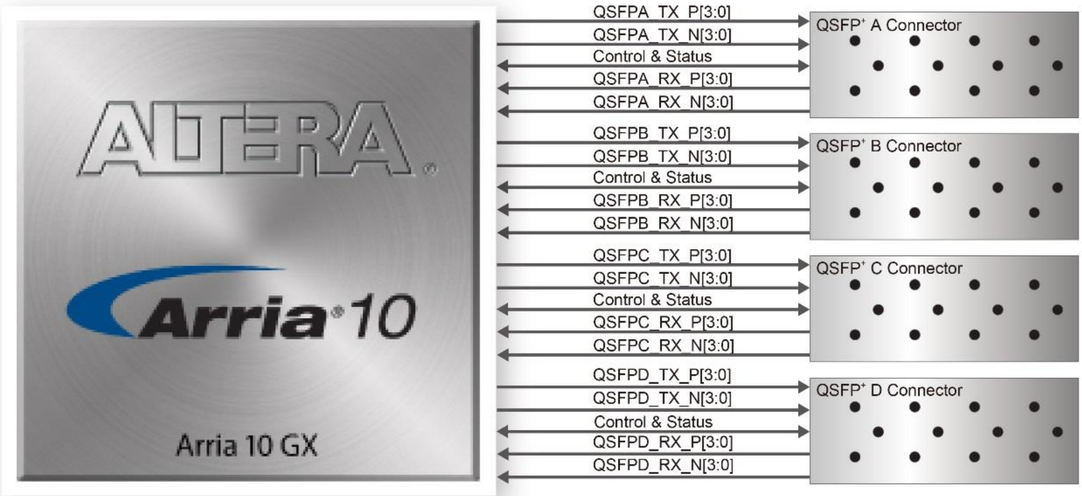



**Figure 2-12 Connection between the QSFP+ and Arria GX FPGA**

**Table 2-17**, **Table 2-18**, **Table 2-19** and **Table 2-20** list the QSFP+ A, B, C and D pin assignments and signal names relative to the Arria 10 GX device.

**Table 2-17 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_AP30 |
| QSFPA_RST_n | Module Reset | 1.8V | PIN_AU30 |
| QSFPA_SCL | 2-wire serial interface clock | 1.8V | PIN_BA32 |
| QSFPA_SDA | 2-wire serial interface data | 1.8V | PIN_BC31 |
| QSFPA_LP_MODE | Low Power Mode | 1.8V | PIN_BD31 |
| QSFPA_INTERRUPT_n | Interrupt | 1.8V | PIN_BA30 |
| QSFPA_MOD_PRS_n | Module Present | 1.8V | PIN_BB30 |

**Table 2-18 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_AY29 |
| QSFPB_RST_n | Module Reset | 1.8V | PIN_BA29 |
| QSFPB_SCL | 2-wire serial interface clock | 1.8V | PIN_BB29 |
| QSFPB_SDA | 2-wire serial interface data | 1.8V | PIN_AY28 |
| QSFPB_LP_MODE | Low Power Mode | 1.8V | PIN_BB28 |
| QSFPB_INTERRUPT_n | Interrupt | 1.8V | PIN_BA27 |
| QSFPB_MOD_PRS_n | Module Present | 1.8V | PIN_BC27 |

**Table 2-19 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_BB27 |
| QSFPC_RST_n | Module Reset | 1.8V | PIN_BA26 |
| QSFPC_SCL | 2-wire serial interface clock | 1.8V | PIN_AU29 |

| | | | |
|---|---|---|---|
| QSFPC_SDA | 2-wire serial interface data | 1.8V | PIN_AU28 |
| QSFPC_LP_MODE | Low Power Mode | 1.8V | PIN_AT30 |
| QSFPC_INTERRUPT_n | Interrupt | 1.8V | PIN_AT29 |
| QSFPC_MOD_PRS_n | Module Present | 1.8V | PIN_AR28 |

**Table 2-20 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_AR27 |
| QSFPD_RST_n | Module Reset | 1.8V | PIN_AP29 |
| QSFPD_SCL | 2-wire serial interface clock | 1.8V | PIN_AP28 |
| QSFPD_SDA | 2-wire serial interface data | 1.8V | PIN_AN30 |
| QSFPD_LP_MODE | Low Power Mode | 1.8V | PIN_AN28 |
| QSFPD_INTERRUPT_n | Interrupt | 1.8V | PIN_AM29 |
| QSFPD_MOD_PRS_n | Module Present | 1.8V | PIN_AM28 |

## 2.10 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 the 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 2-13** presents the pin connection established between the Arria 10 GX and PCI Express.

The Dual 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 (SW2) is connected to the PCI Express to allow different configurations to enable a x1, x4, or x8 PCIe.

**Table 2-21** summarizes the Dual PCI Express pin assignments of the signal names relative to the Arria 10 GX FPGA.

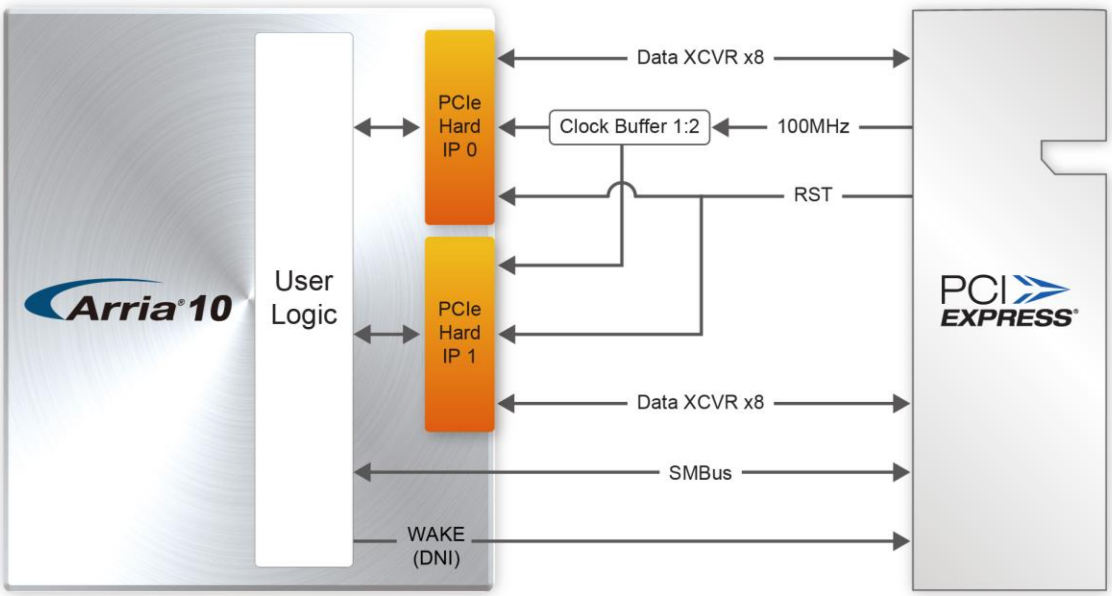



**Figure 2-13 PCI Express pin connection**

**Table 2-21 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 |
| 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_BD24 |
| PCIE_WAKE_n | Wake signal | 1.8-V | PIN_AN26 |
| PCIE_PRSNT1n | Hot plug detect | - | - |
| PCIE_PRSNT2n_x1 | Hot plug detect x1 PCIe slot enabled using SW5 dip switch | - | - |
| PCIE_PRSNT2n_x4 | Hot plug detect x4 PCIe slot enabled using SW5 dip switch | - | - |
| PCIE_PRSNT2n_x8 | Hot plug detect x8 PCIe slot enabled using SW5 dip switch | - | - |
| PCIE2_TX_p0 | Add-in card transmit bus | 1.4-V PCML | PIN_P44 |
| PCIE2_TX_n0 | Add-in card transmit bus | 1.4-V PCML | PIN_P43 |
| PCIE2_TX_p1 | Add-in card transmit bus | 1.4-V PCML | PIN_M44 |
| PCIE2_TX_n1 | Add-in card transmit bus | 1.4-V PCML | PIN_M43 |
| PCIE2_TX_p2 | Add-in card transmit bus | 1.4-V PCML | PIN_K44 |
| PCIE2_TX_n2 | Add-in card transmit bus | 1.4-V PCML | PIN_K43 |
| PCIE2_TX_p3 | Add-in card transmit bus | 1.4-V PCML | PIN_H44 |
| PCIE2_TX_n3 | Add-in card transmit bus | 1.4-V PCML | PIN_H43 |
| PCIE2_TX_p4 | Add-in card transmit bus | 1.4-V PCML | PIN_F44 |
| PCIE2_TX_n4 | Add-in card transmit bus | 1.4-V PCML | PIN_F43 |
| PCIE2_TX_p5 | Add-in card transmit bus | 1.4-V PCML | PIN_D44 |
| PCIE2_TX_n5 | Add-in card transmit bus | 1.4-V PCML | PIN_D43 |
| PCIE2_TX_p6 | Add-in card transmit bus | 1.4-V PCML | PIN_B44 |
| PCIE2_TX_n6 | Add-in card transmit bus | 1.4-V PCML | PIN_B43 |
| PCIE2_TX_p7 | Add-in card transmit bus | 1.4-V PCML | PIN_A42 |
| PCIE2_TX_n7 | Add-in card transmit bus | 1.4-V PCML | PIN_A41 |
| PCIE2_RX_p0 | Add-in card receive bus | 1.4-V PCML | PIN_N42 |
| PCIE2_RX_n0 | Add-in card receive bus | 1.4-V PCML | PIN_N41 |
| PCIE2_RX_p1 | Add-in card receive bus | 1.4-V PCML | PIN_L42 |
| PCIE2_RX_n1 | Add-in card receive bus | 1.4-V PCML | PIN_L41 |
| PCIE2_RX_p2 | Add-in card receive bus | 1.4-V PCML | PIN_J42 |
| PCIE2_RX_n2 | Add-in card receive bus | 1.4-V PCML | PIN_J41 |

| | | | |
|---|---|---|---|
| PCIE2_RX_p3 | Add-in card receive bus | 1.4-V PCML | PIN_G42 |
| PCIE2_RX_n3 | Add-in card receive bus | 1.4-V PCML | PIN_G41 |
| PCIE2_RX_p4 | Add-in card receive bus | 1.4-V PCML | PIN_E42 |
| PCIE2_RX_n4 | Add-in card receive bus | 1.4-V PCML | PIN_E41 |
| PCIE2_RX_p5 | Add-in card receive bus | 1.4-V PCML | PIN_D40 |
| PCIE2_RX_n5 | Add-in card receive bus | 1.4-V PCML | PIN_D39 |
| PCIE2_RX_p6 | Add-in card receive bus | 1.4-V PCML | PIN_C42 |
| PCIE2_RX_n6 | Add-in card receive bus | 1.4-V PCML | PIN_C41 |
| PCIE2_RX_p7 | Add-in card receive bus | 1.4-V PCML | PIN_B40 |
| PCIE2_RX_n7 | Add-in card receive bus | 1.4-VPCML | PIN_B39 |
| PCIE2_REFCLK_p | Motherboard reference clock | HCSL | PIN_Y40 |
| PCIE2_REFCLK_n | Motherboard reference clock | HCSL | PIN_Y39 |
| PCIE2_PERST_n | Reset | 1.8-V | PIN_AR24 |

## 2.11 2x5 Timing Header

The FPGA board has one 2x5 GPIO header J5 for expansion function. The pin-out of J5 is shown in **Figure 2-14**. GPIO_P0 ~ GPIO_P3 are bi-direction 1.8V GPIO. GPIO_CLK0 and GPIO_CLK1 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 and external clock inputs/UART applications.

**Table 2-22** shows the mapping of the FPGA pin assignments to the 2x5 GPIO header..

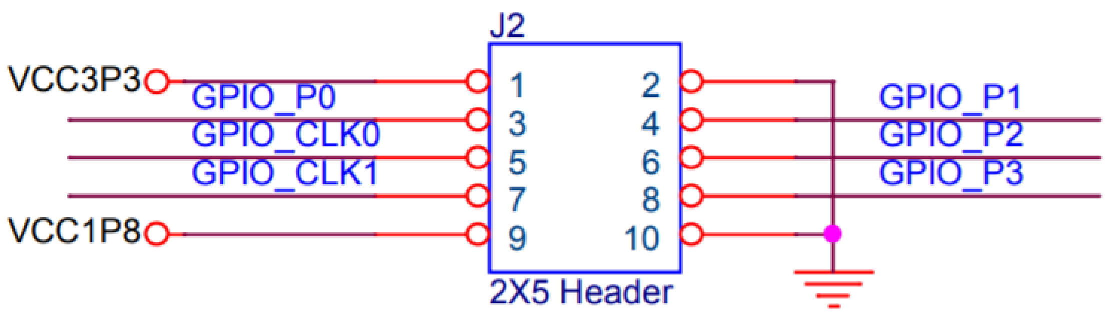



**Figure 2-14 Pin-out of Timing Expansion Header**

**Table 2-22 Timing Expansion Header Pin Assignments, Schematic Signal Names, and Functions**

| Schematic Signal Name | Description | I/O Standard | Stratix 10 GX/SX Pin Number |
|---|---|---|---|
| GPIO_P0 | Bi-direction 1.8V GPIO | 1.8-V | PIN_BC28 |
| GPIO_P1 | Bi-direction 1.8V GPIO | 1.8-V | PIN_BD29 |
| GPIO_P2 | Bi-direction 1.8V GPIO | 1.8-V | PIN_BC30 |
| GPIO_P3 | Bi-direction 1.8V GPIO | 1.8-V | PIN_AR29 |
| GPIO_CLK0 | FPGA dedicated clock input or Bi-direction 1.8V GPIO | 1.8-V | PIN_ AV33 |
| GPIO_CLK1 | FPGA dedicated clock input or Bi-direction 1.8V GPIO | 1.8-V | PIN_ AW33 |

## 2.12 2x4 GPIO Expansion Header

The 2x4, 2.0 mm pitch GPIO expansion header is designed to provide seven user pins connected directly to the FPGA and one GND pin. Figure 2-15 shows the connection between 2x4 GPIO header and Arria 10 GX FPGA. Table 2-23 lists the pin assignment of 2x4 GPIO header.

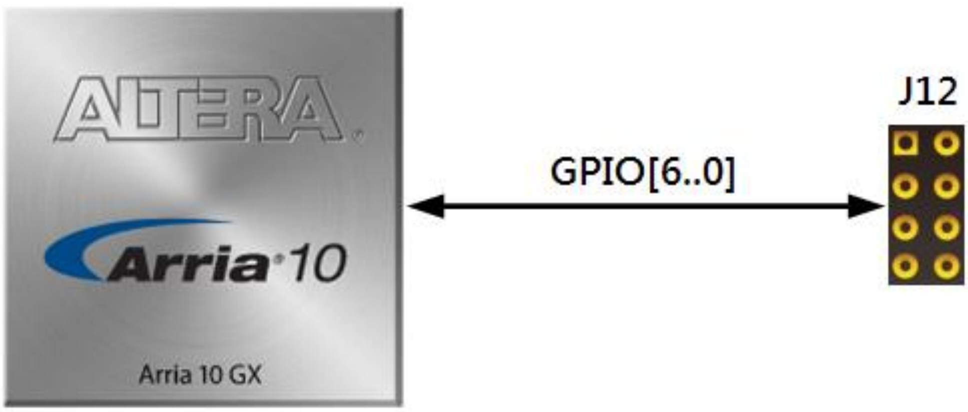



**Figure 2-15 Connection between 2x4 GPIO Header and Arria 10 GX FPGA**

Table 2-23 Pin Assignments of 2x4 GPIO Header

| Schematic Signal Name | Description | I/O Standard | Arria 10 GX Pin Number |
|---|---|---|---|
| GPIO0 | GPIO Connection [0] | 1.8-V | PIN_AT36 |
| GPIO1 | GPIO Connection [1] | | PIN_AT35 |
| GPIO2 | GPIO Connection [2] | | PIN_AU35 |
| GPIO3 | GPIO Connection [3] | | PIN_AU34 |
| GPIO4 | GPIO Connection [4] | | PIN_AV35 |
| GPIO5 | GPIO Connection [5] | | PIN_AU32 |
| GPIO6 | GPIO Connection [6] | | PIN_AV32 |

*Chapter 3*

# 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 II project for the FPGA board within minutes. The Quartus II project files generated include:

- Quartus II Project File (.qpf)
- Quartus II 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 provides the detail procedures on how the System Build
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 II setting file (.qsf).

The top-level design file contains top-level Verilog wrapper for users to add their own design/logic. The Quartus II 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 II programmer must be used to download SOF file to the FPGA board using JTAG interface.

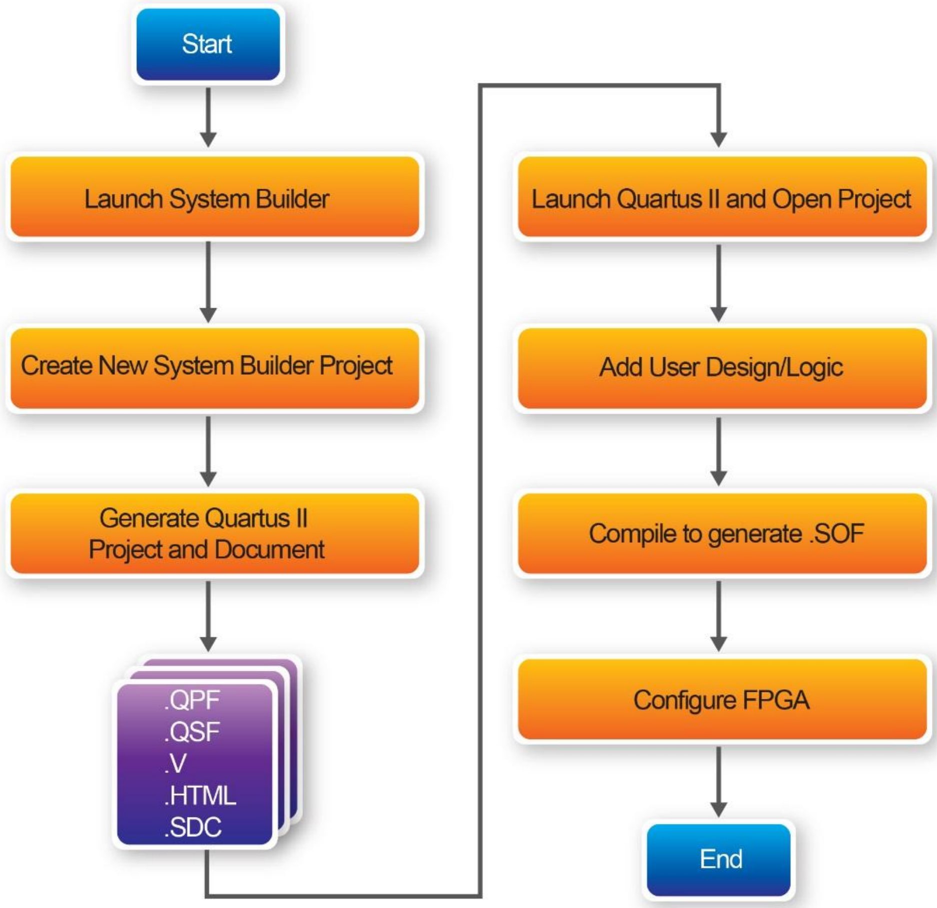



**Figure 3-1Thegeneral design flow of building a project**

## 3.3 Using System Builder

This section provides the detail 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**.

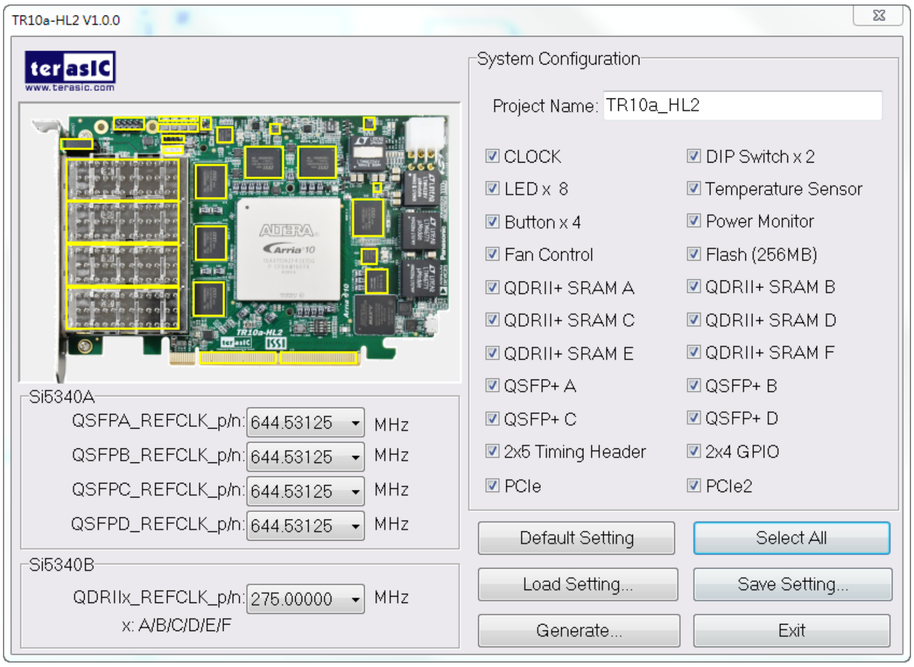



**Figure 3-2 The System Builder window**

### 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.

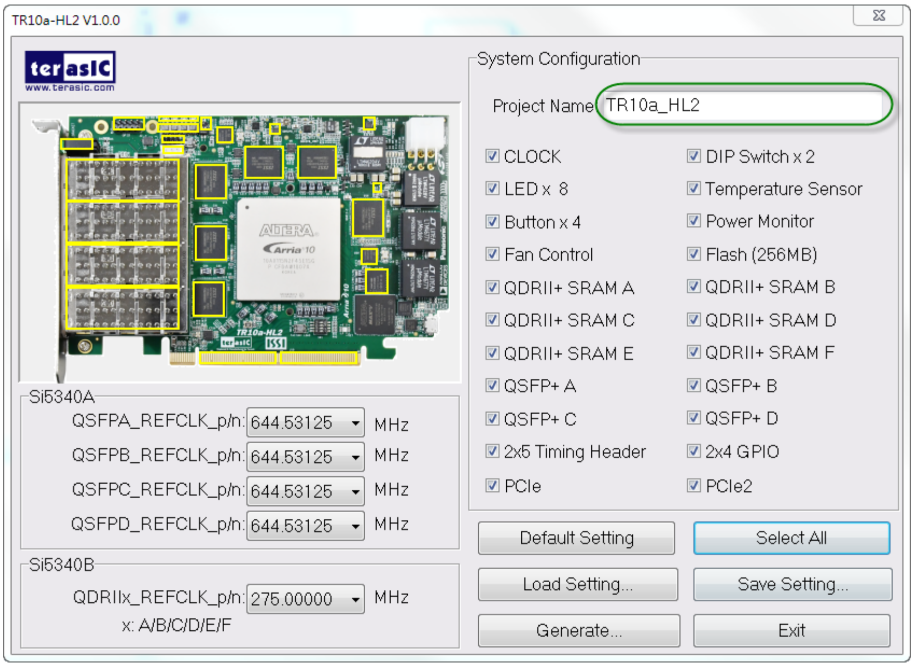



**Figure 3-3 The Quartus project name**

## ■ 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. QDRII+ 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 QDRII+ controller, please refer to the QDRII+ SRAM demonstration in Chapter 6.

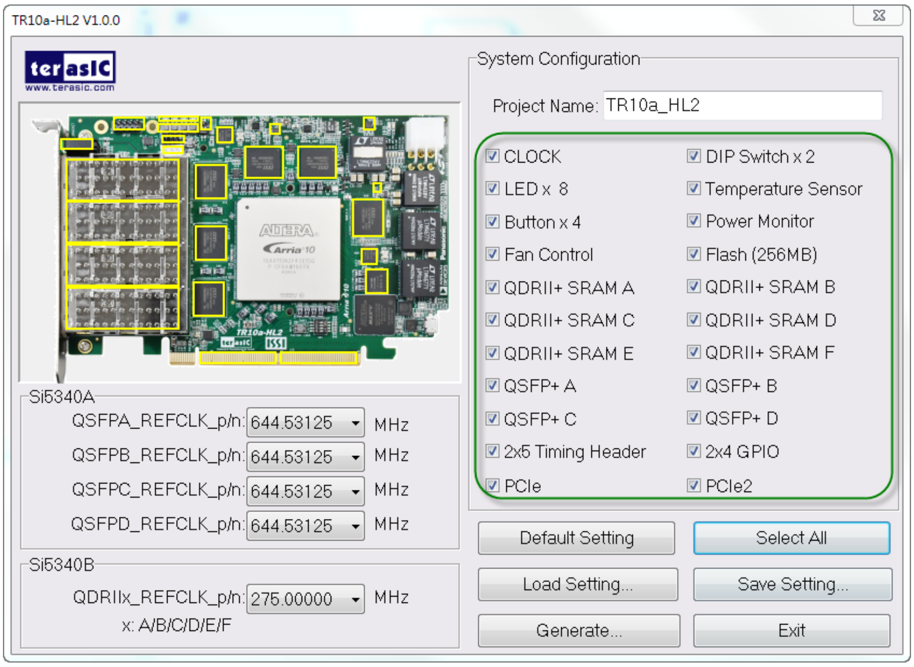



**Figure 3-4 System Configuration group**

## ■ Programmable Oscillator

There are two external oscillators on-board that provide reference clocks for the following signals QSFPA_REFCLK,QSFPB_REFCLK,QSFPC_REFCLK,QSFPD_REFCLK, QDRIIA_REFCLK, QDRIIB_REFCLK, QDRIIC_REFCLK , QDRIID_REFCLK , QDRIIE_REFCLK and QDRIIF_REFCLK. To use these clock, users can select the desired frequency on the Programmable Oscillator group, as shown in **Figure 3-5**. QDRII, 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.

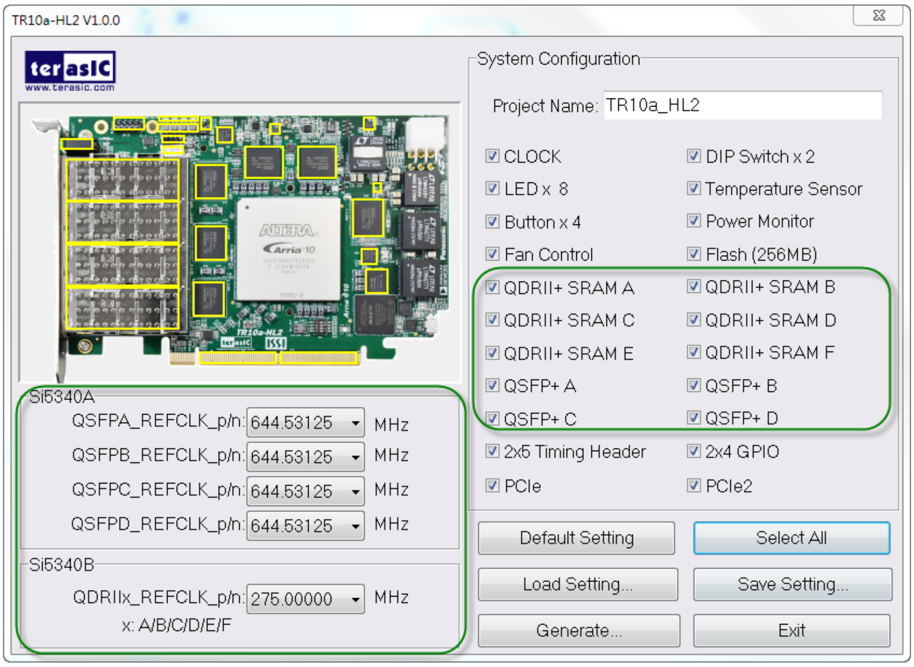



**Figure 3-5 External programmable oscillators**

## ■ 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.

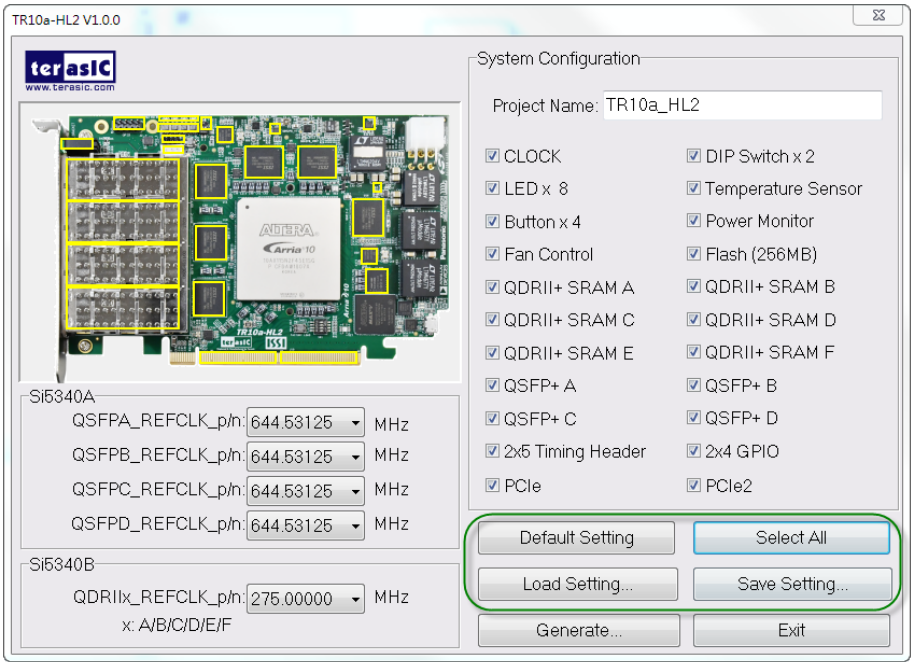



**Figure 3-6 Project Settings**

## Project Generation

When users press the Generate button, the System Builder will generate the corresponding Quartus II 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 II |
| 2 | Si5340_controller (*) | Si5340A and Si5340B External Oscillator Controller IP |
| 3 | <Project name>.qpf | Quartus II Project File |
| 4 | <Project name>.qsf | Quartus II Setting File |
| 5 | <Project name>.sdc | Synopsis Design Constraints File for Quartus II |
| 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 II to generate the SRAM Object File (.sof).

For Si5340A, its controller will be instantiated in the Quartus II 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;
wire si5340a_config_done;

assign si5340a_controller_start = ~BUTTON[0];

si5340a_controller si5340a_controller(
    .iCLK(CLK_50_B2F),
    .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
    .I2C_CLK(SI5340A_I2C_SCL),
    .I2C_DATA(SI5340A_I2C_SDA),
    .oPLL_REG_CONFIG_DONE(si5340a_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 II top-level file, as listed below:

```
//=======================================================
//  Configure SI5340B
//=======================================================
`define REFCLK_QDR275   4'h0
`define REFCLK_QDR250   4'h1
`define REFCLK_QDR225   4'h2

wire si5340b_controller_start;
wire si5340b_config_done;

assign si5340b_controller_start = ~BUTTON[0];

si5340b_controller si5340b_controller(
    .iCLK(CLK_50_B2F),
    .iRST_n(CPU_RESET_n),
    .iStart(si5340b_controller_start),
    .iPLL_OUT_FREQ_SEL(`REFCLK_QDR275),
    .I2C_CLK(SI5340B_I2C_SCL),
    .I2C_DATA(SI5340B_I2C_SDA),
    .oPLL_REG_CONFIG_DONE(si5340b_config_done)
);

assign SI5340B_OE_n = 1'b0;
assign SI5340B_RST_n = CPU_RESET_n;
```

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

Chapter 4

# 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 CFI 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 two FPGA configurations. The two configuration images are called: **factory** image and **user** image, respectively.

## 4.1 FPGA Configure Operation

Below shows the procedure to enable the FPGA configuration from Flash. Users can select one boot image between factory image and user image.

1. Make sure the two default FPGA configurations data has been stored in the CFI flash.
2. Set the FPGA configuration mode to FPPx16 mode by setting SW1 MSEL[2:0] as **000** as shown in **Figure 4-1**.
3. Specify the configuration of the FPGA using the default Factory Configuration Image or User Configuration Image by setting SW1.4 according to **Figure 4-2**. When the switch is in position “1”, the factory image is used when the system boots. When the switch is in position “0”, user image is used when the system boots.
4. Power on the FPGA board or press the MAX_RST button if board is already

powered on,

5. When the configuration is completed, the green Configure Done LED will light. If there is an error, the red Configure Error LED will light.

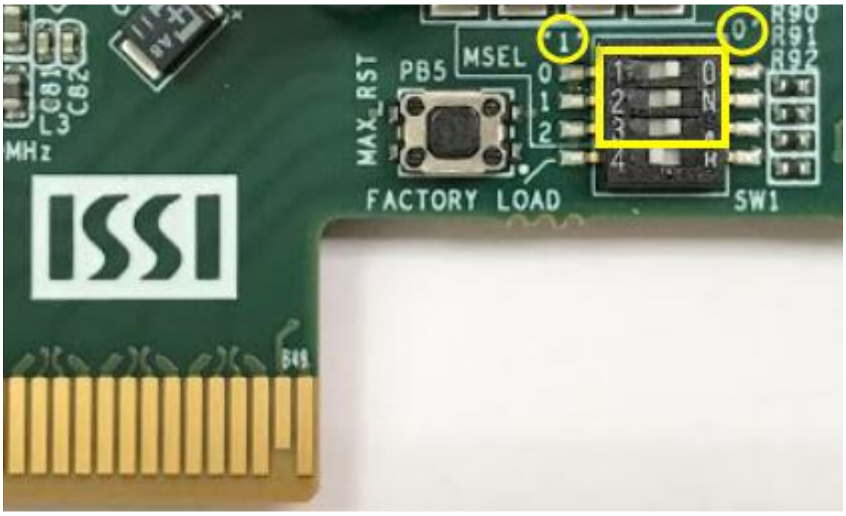



**Figure 4-1 MSEL[2:0]=000**

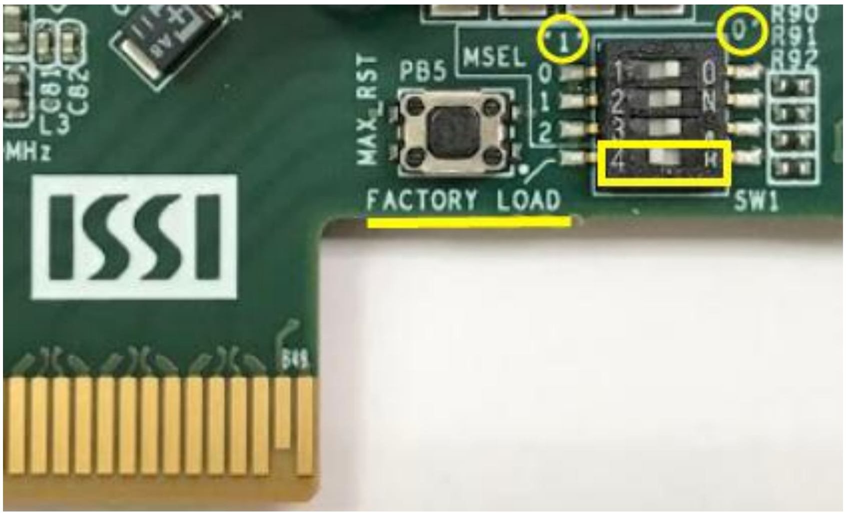



**Figure 4-2 Configuration Image Selection**

## 4.2 CFI Flash Memory Map

The TR10a-HL2 has one 1-Gbit, 16-bit data width, CFI compatible synchronous flash device for non-volatile storage of the FPGA configuration data, user Nios II code, and user data. Both MAX V CPLD and Stratix 10 GX FPGA can access this Flash device.

MAXV CPLD accesses flash for FPP x16 configuration of the FPGA at power-on and board reset events. It uses the PFL-II Mega function. Arria10 10 GX FPGA access to the flash memory's user space is done by Nios II.

**Table 4-1** shows the memory map for the on-board flash. This memory provides non-volatile storage for two FPGA bit-streams and Nios II Program, users data, as well as FPL option bits for PFL II configuration bits and board information. For the factory default code to run correctly and update designs in the user memory, this memory map address must not be altered.

**Table 4-1 Flash Memory Map (Byte Address)**

| Block Description | Size(KB) | Address Range |
|---|---|---|
| Factory Board Information | 128 | 0x00010000 – 0x0002FFFF |
| 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 | 34,560 | 0x05E40000 – 0x07FFFFFF |

The **Factory Board Information** stores the Manufacture Serial Number of the FPGA board. The Serial Number is a 13 digital number with format mmmmmmmm-nnnn. Users can find the number on the serial number sticker on the FPGA board.

The **PFL option bits** contains the image location of the **Factory hardware** and **User hardware**, so the PLF II IP in the MAX V CPLD can know where to find the FPGA configuration data. If developers erase all flash content, please ensure that the PFL option is reprogrammed with the FPGA configuration data.

For user’s application, the **User hardware** must be stored with start address **0x02B40000**, and the user’s software is suggested to be stored with start address **0x05E40000**. Users also can overwrite the Factory hardware and Factory software based on their application. **Factory hardware** must be stored with start address **0x00040000**, and the Factory software should be stored with start address **0x05640000**. We strongly recommend users to use the batch file in the **Flash_Restored** folder to write the hardware and software data into the CFI-Flash.

# 4.3 Flash Example Designs

There are four flash example designs and one programming batch folder in the Demonstration folder under the System CD as shown in **Table 4-2**.

**Table 4-2 Flash Example Design**

| Example Folder | Description |
|---|---|
| Flash_Programming | This is the flash programming design. It is used to write data into FLASH by a Quartus Programmmer. |
| Flash_Factory | A simple example design. Its FPGA configure data and Nios II codes are stored in the Factory Image Area. |
| Flash_User | A simple example design. Its FPGA configure data and Nios II codes are stored in the User Image Area. |
| Flash_Tool | A Nios II program shows how to access flash content. |
| Flash_Restored | A batch file used for to programming Flash_Factory and the Flash_User project into CFI Flash. |

**Figure 4-3** shows the relationship between the three examples – **Flash_Programming**, **Flash_Factory** and **Flash_User**. The **Flash_Programming** example is used to write data into the CFI Flash on the FPGA Board. The **Flash_Factory** and **Flash_User** are simple designs with Nios II processor. These two designed are written into CFI-Flash so they are selected to configure the FPGA when the FPGA is powered on.

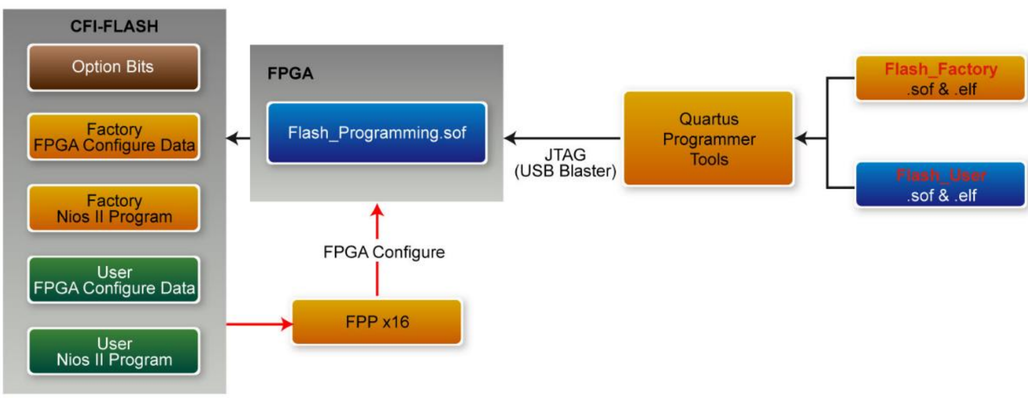



**Figure 4-3 Relationship between three flash examples**

The **Flash_Tool** is designed to show how to access flash via the Nios II processor. The design shows how to erase flash and read flash content.

## 4.4 Flash_Programming Example

The **Flash_Programing** project is designed to program CFI flash by a Quartus Programmer. In the project, Intel Parallel Flash Loader II IP is used to program the CFI-Flash. **Figure 4-4** shows the Generic Setting in the IP. "Flash Programming" operation mode is used, and "CFI Parallel Flash" is selected. **Figure 4-5** shows the Flash Interface Setting. "CFI 1 Gbit" is selected. The DE10-Pro.sof generated by this program is used in the flash programming batch files located in the **Flash_Restored** Folder.

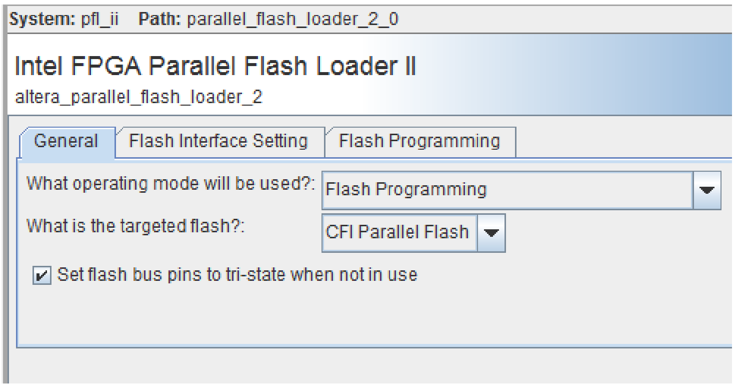



**Figure 4-4 General Setting in PFL II IP**

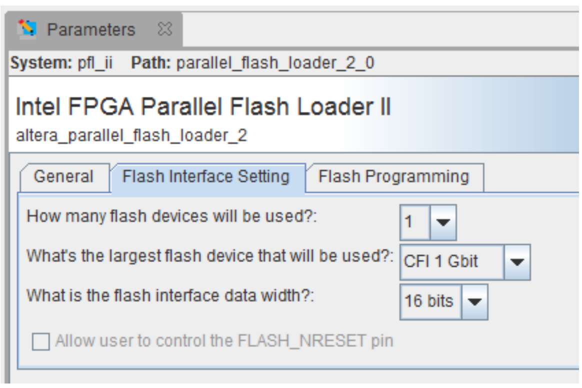



**Figure 4-5 Flash Interface Setting in PFL II IP**

# 4.5 Flash_Factory Example

The **Flash_Factory** is designed to show how to create a Nios II code which is booted from the Factory Software location in the CFI Flash when the board is powered on. This project's FPGA configuration data and Nios II code are stored in the Factory Hard area and Factory Software area of the CFI Flash when the FPGA board is shipped.

To develop this kind of boot code, first, developers need to include the Tri-State Conduit Bridge and the Generic Tri-State Controller in the **Platform Designer** (formerly Qsys) to implement the flash controller function, and connect the Nios II processor's data bus and instruction bus to the flash controller as shown in **Figure 4-6** . Then, specify the Factory Software Location **0x05640000** as Reset Vector in the Nios II Processor component as shown in **Figure 4-7** . Finally, developers need to uncheck the **allow_code_at_reset** and **enable_alt_load** options in the BSP editor under of Nios II IDE tool (Nios II Software Builder Tools for Eclipse) as shown in **Figure 4-8** .

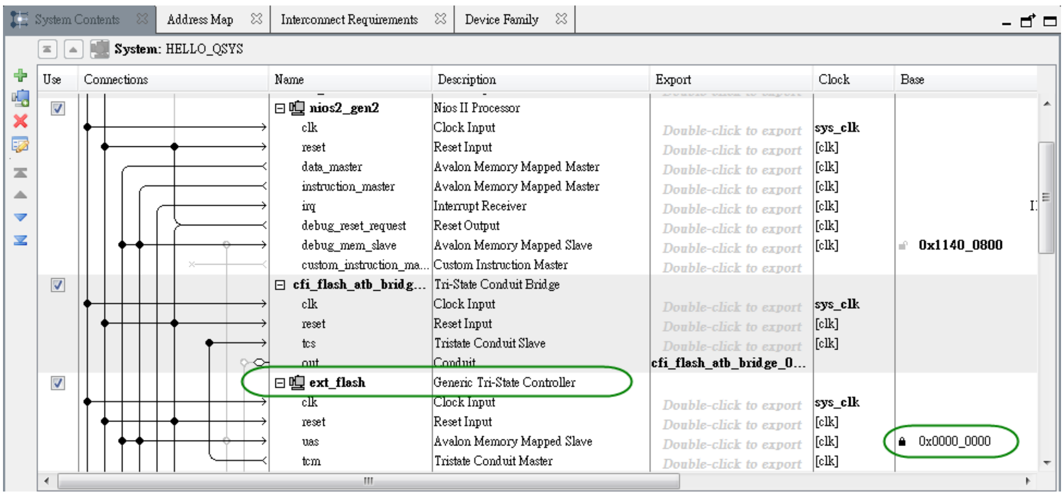



Figure 4-6 Flash Controller Settings in Platform Designer (formerly Qsys)

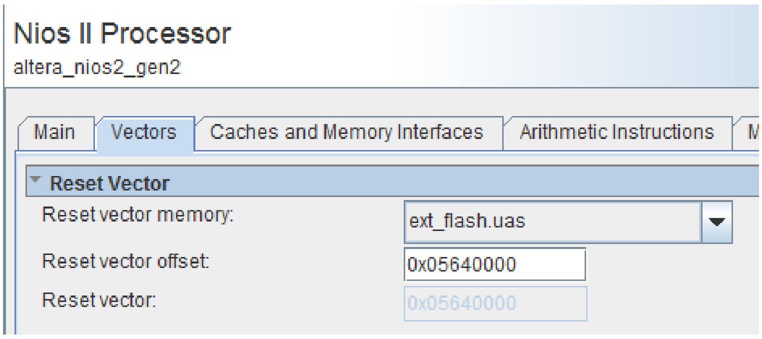



**Figure 4-7 Factory Software Reset Vector Settings for NIOS II Processor**

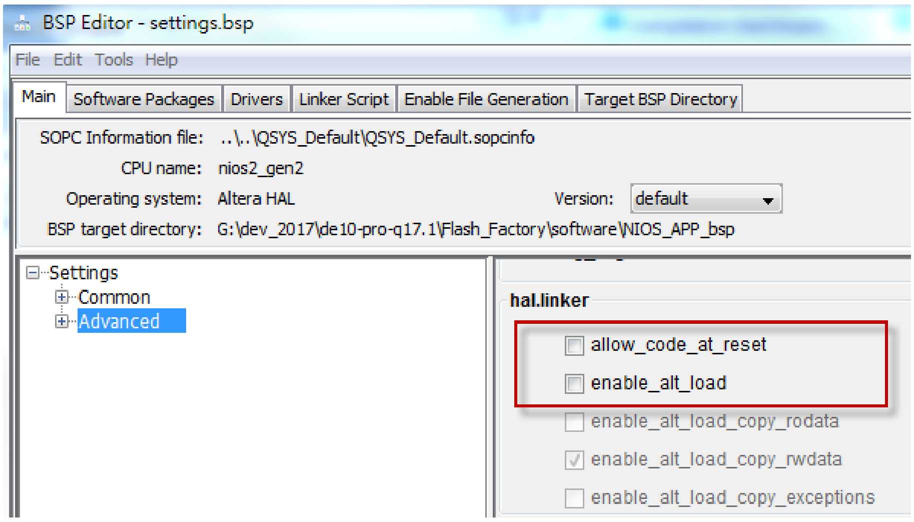



**Figure 4-8 BSP Editor in Nios II IDE**

# 4.6 Flash_User Example

The **Flash_User** project is similar with the above **Flash_Factory** example code. This project's FPGA configuration data and Nios II code are stored in the User Hard area and User Software area when the FPGA board is shipped.

The major difference between the **Flash_User** and **Flash_Factory** is the Reset Vector address in the Nios II processor component and the LED control code in Nios II program. The User Software Location **0x05E40000** is used as Reset Vector as shown in **Figure 4-9** .

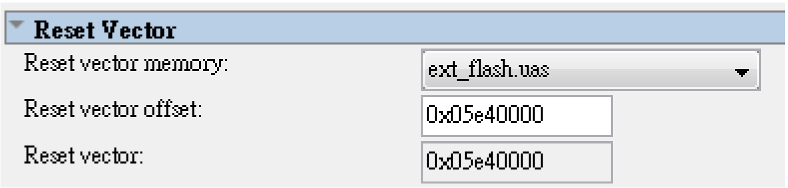



**Figure 4-9 User Software Reset Vector Settings for NIOS II Processor**

# 4.7 Flash_Tool Example

This example show how the Nios II program accesses the FLASH. **Figure 4-10** shows a screenshot of the Flash_Tool menu shown under Nios II terminal.

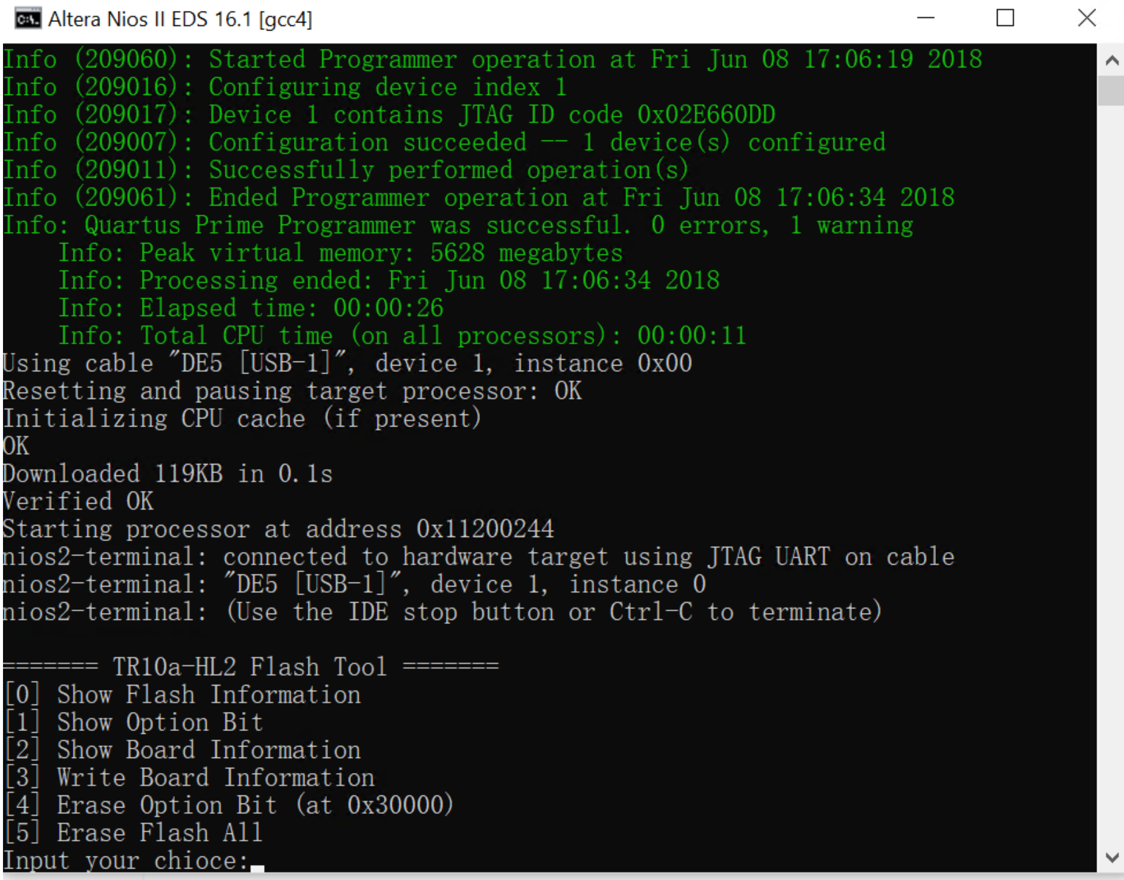



**Figure 4-10 Screenshot of Flash_Tool menu**

The tools provide the following functions:

- Show CFI Flash Size
- Show Option bits used by FPP x16 Configuration
- Read Serial Number from the CFI Flash

- Erase Serial Number to the CFI flash
- Erase option bits used by FPP x16
- Erase whole flash

## 4.8 Programming Batch File

The **Flash_Restored** folder includes batch files to program the Factory image and User image into the CFI flash. **Figure 4-11** shows the contents of the **Flash_Restored** folder. The **factory** subfolder includes the .sof & .elf files generated by the **Flash_Factory** project. The **user** subfolder includes the .sof & .elf files generated by the **Flash_User** project. TR10a_HL2.sof is generated by the **Flash_Programming** project.

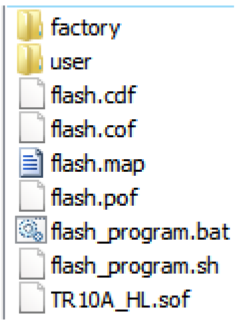



**Figure 4-11 Flash_Restored folder content**

The flash_program.bat is the top batch file for flash programming. The batch file will configure the FPGA with TR10a_HL2.sof (Parallel Flash Loader II IP) and launch flash_program.sh Nios II command batch file to perform the following tasks:

1. Use Nios II utilities **elf2flash** and **nios2-elf-objcopy** to convert **Factory** Nios II code and **User** Nios II code to factory_sw.hex and user_sw.hex, respectively.
2. Use **quartus_cpf** utility according to a given configuration file **flash.cof** to merger all files (factory_sw.hex, user_sw.hex, factory .sof file, user .sof file, and option bit) into a single file **flash.pof**.
3. Use **jtagconfig** utility to adjust jtag speed.
4. Use **quartus_pgm** utility to program flash with flash.pof.

Developers can copy their .sof & .efl files into the factory folder or the user folder, and launch the flash_program.bat to program their code into the CFI-Flash.

## 4.9 Restore Factory Settings

This section describes how to restore the original **Factory** image and **User** image into the flash memory device on the FPGA development board. A programming batch file located in the **Flash_Restored** folder is used to restore the flash content. Performing the following instructions can restore the flash content:

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 USB Cable.
3. Power on the FPGA board.
4. Copy the "Demonstrations/Flash_Restored" folder under the CD to your PC's local drive.
5. Execute the batch file flash_program.bat to start flash programming.

After restoring the flash, perform the following procedures to test the restored boot code.

1. Power off the FPGA Board.
2. Set FPGA configuration mode as AVSTx8 Mode by setting S1 MSEL[2:0] to **000**.
3. Specify configuration of the FPGA to Factory Hardware by setting the FACTORY_LOAD dip in SW1.4 to the '1' position.
4. Power on the FPGA Board, and the Configure Done LED D7 should light up.

The batch file converts the **Factory** and **User** .sof/.elf and PFL option bit into a flash.pof file and use Quartus Programmer to program the CFI-Flash with the generated flash.pof. The **factory** subfolder includes TR10a_HL2.sof and NIOS_APP.elf files generated by **Flash_Factory** project, and the **user** subfolder includes TR10a_HL2.sof and NIOS_APP.elf files generated by **Flash_User** project. The TR10a_HL2.sof under the **Flash_Restored** folder is used to program flash by Quartus Programmer.

Chapter 5

# Peripheral Reference Design

This chapter introduces TR10a-HL2 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 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 four LEDs located at PCie bracket. 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.

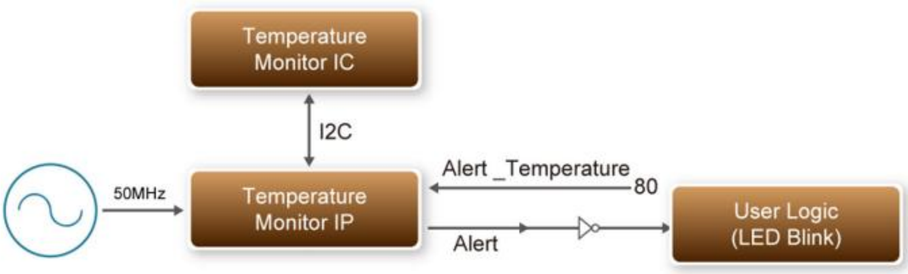



**Figure 5- 1 Block Diagram of Temperature Monitor (!!! Need update by nina)**

## 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.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.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 by the four bracket LEDs as defined in **Table 5-1**.
- 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.

**Table 5-1 Bracket LED indication for Temperature**

| Temperature | Bracket LED<br>1: LED Lighten<br>0: LED Un-lighten |
|---|---|
| >= *100* | 1111 |
| >= *95* | 1110 |
| >= *90* | 1101 |
| >= *85* | 1100 |
| >= *80* | 1011 |
| >= *75* | 1010 |
| >= *70* | 1001 |
| >= *65* | 1000? |
| >= *60* | 0111 |
| >= *55* | 0101 |
| >= *50* | 0101 |
| >= *45* | 0100 |
| >= *40* | 0011 |
| >= *35* | 0010 |
| < *35* | 0001 |

# 5.2 Configure Si5340A/B in RTL

There are two Silicon Labs Si5340 clock generators on TR10a-HL2 FPGA board can provide adjustable frequency reference clock (See **Figure 5-1**) for QSFP and QDRII 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.

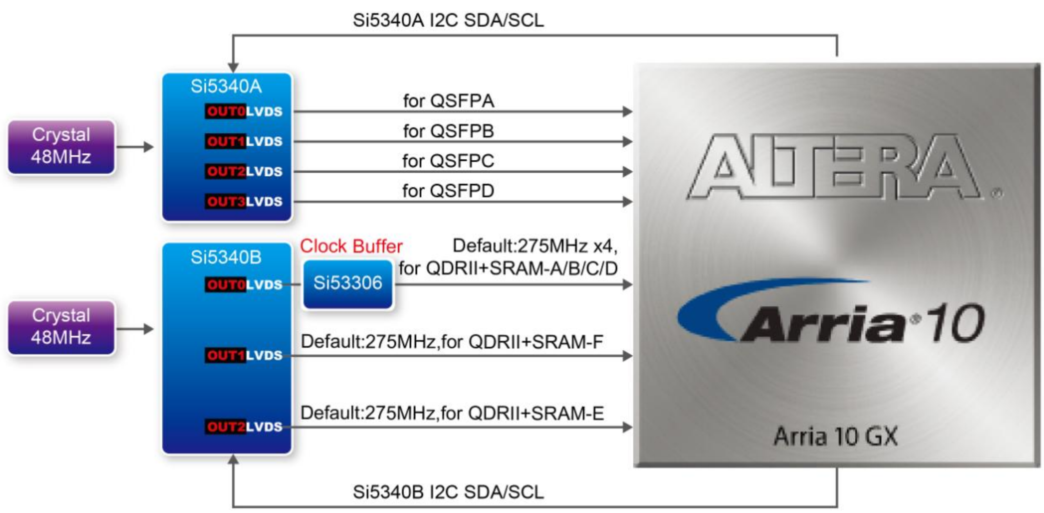



**Figure 5-1 Si5340 Clock Generators**

## Creating Si5340 Control IP

The Si5340 control IP is located in the folder:

\Demonstration\si5340_control_ip”.

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 QSFP and QDRII interfaces, Si5340 corresponding output channels will become available and users can select desired frequencies. For example, when checking QSFP+ A box (See **Figure 5-2**), SI5340A QSFPA_REFCLK_P/N can provide six frequencies from 100Mhz to 644.5312Mhz for users selecting.

As shown in **Figure 5-3**, if all the receiving Si5340 reference clock interface boxes are checked, then, every frequency channel of the two Si5340 chips is controllable by users.

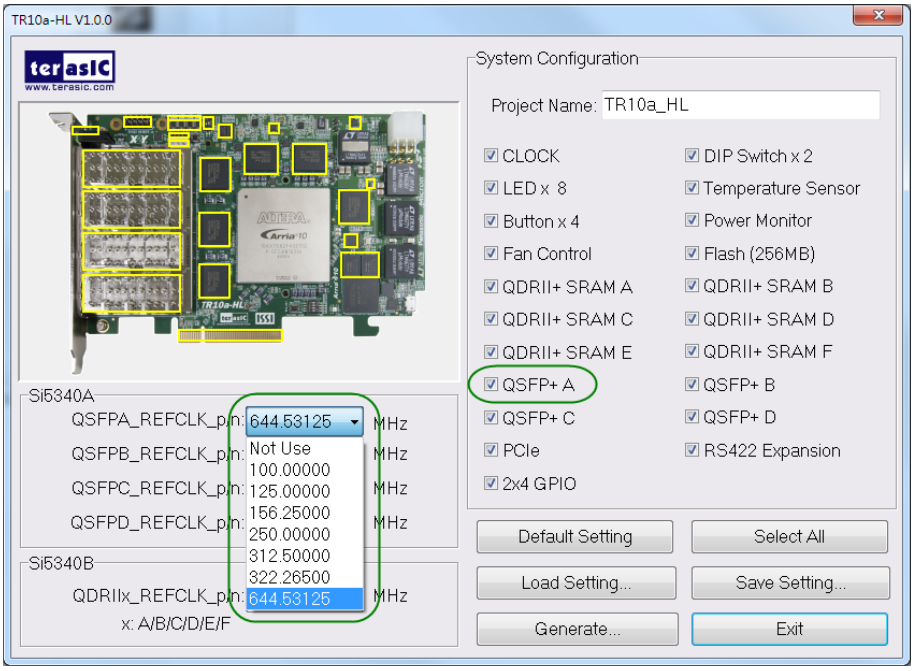



**Figure 5-2 Enable Si5340A clock on System Builder**

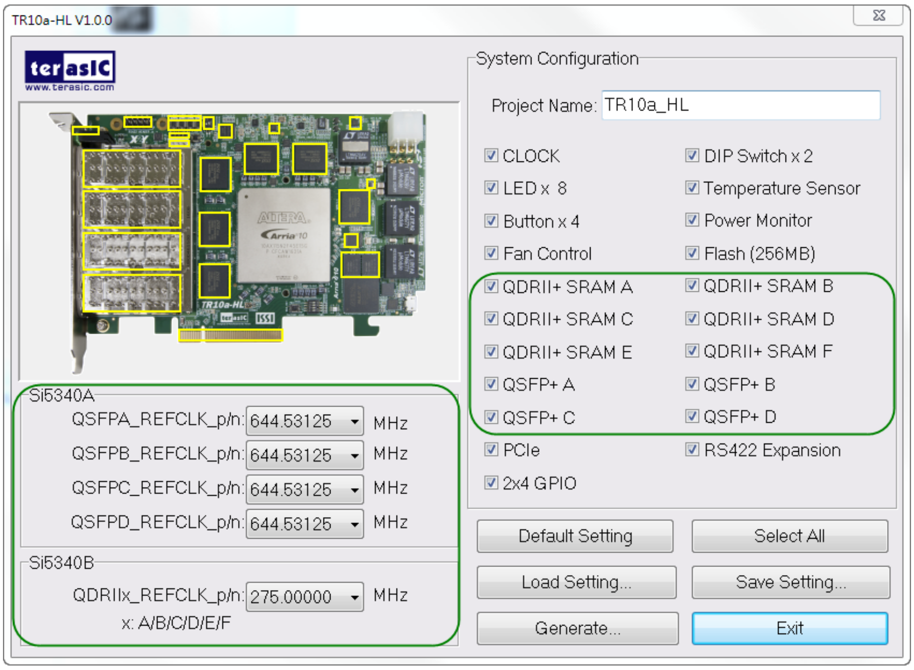



**Figure 5-3 Enable Si5340A and Si5340B clock on System Builder**

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;
wire si5340a_config_done;

assign si5340a_controller_start = ~BUTTON[0];

si5340a_controller si5340a_controller(
      .iCLK(CLK_50_B2F),
      .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
      .I2C_CLK(SI5340A_I2C_SCL),
      .I2C_DATA(SI5340A_I2C_SDA),
      .oPLL_REG_CONFIG_DONE(si5340a_config_done)
);

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

```
//=======================================================
//   Configure SI5340B
//=======================================================
`define REFCLK_QDR275     4'h0
`define REFCLK_QDR250     4'h1
`define REFCLK_QDR225     4'h2
```

```
wire si5340b_controller_start;
wire si5340b_config_done;


assign si5340b_controller_start = ~BUTTON[0];


si5340b_controller si5340b_controller(
	.iCLK(CLK_50_B2F),
	.iRST_n(CPU_RESET_n),
	.iStart(si5340b_controller_start),
	.iPLL_OUT_FREQ_SEL(`REFCLK_QDR275),
	.I2C_CLK(SI5340B_I2C_SCL),
	.I2C_DATA(SI5340B_I2C_SDA),
	.oPLL_REG_CONFIG_DONE(si5340b_config_done)
);


assign SI5340B_OE_n = 1'b0;
assign SI5340B_RST_n = CPU_RESET_n;
```

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 **Figure 5-4**).

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

#**************************************************************
# Create Clock
#**************************************************************
create_clock -period "100.000000 MHz"  [get_ports CLKUSR_100]
create_clock -period "100.000000 MHz"  [get_ports CLK_100_B3D]
create_clock -period "50.000000 MHz"  [get_ports CLK_50_B2J]
create_clock -period "50.000000 MHz"  [get_ports CLK_50_B2L]
create_clock -period "50.000000 MHz"  [get_ports CLK_50_B3D]
create_clock -period "50.000000 MHz"  [get_ports CLK_50_B3F]
create_clock -period "50.000000 MHz"  [get_ports CLK_50_B3H]
create_clock -period "275.000000 MHz"  [get_ports QDRIIA_REFCLK_p]
create_clock -period "275.000000 MHz"  [get_ports QDRIIB_REFCLK_p]
create_clock -period "275.000000 MHz"  [get_ports QDRIIC_REFCLK_p]
create_clock -period "275.000000 MHz"  [get_ports QDRIID_REFCLK_p]
create_clock -period "275.000000 MHz"  [get_ports QDRIIE_REFCLK_p]
create_clock -period "275.000000 MHz"  [get_ports QDRIIF_REFCLK_p]
create_clock -period "644.531250 MHz"  [get_ports QSFPA_REFCLK_p]
create_clock -period "644.531250 MHz"  [get_ports QSFPB_REFCLK_p]
create_clock -period "644.531250 MHz"  [get_ports QSFPC_REFCLK_p]
create_clock -period "644.531250 MHz"  [get_ports QSFPD_REFCLK_p]

#**************************************************************
```

**Figure 5-4 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/fromSi5340 |
| 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 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 Frequency Setting**

| iPLL_OUT_FREQ_SEL MODE Setting | QDRII Frequency(MHz) |
|---|---|
| 4'b0000 | 275 |
| 4'b0001 | 250 |
| 4'b0010 | 225 |

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 5-5**), 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/clockuilder_ro_oftware


**Figure 5-5 ClockBuilder Pro Wizard**

2. After the installation, select Si5340, and configure the input frequency and output frequency as shown in **Figure 5-6**.

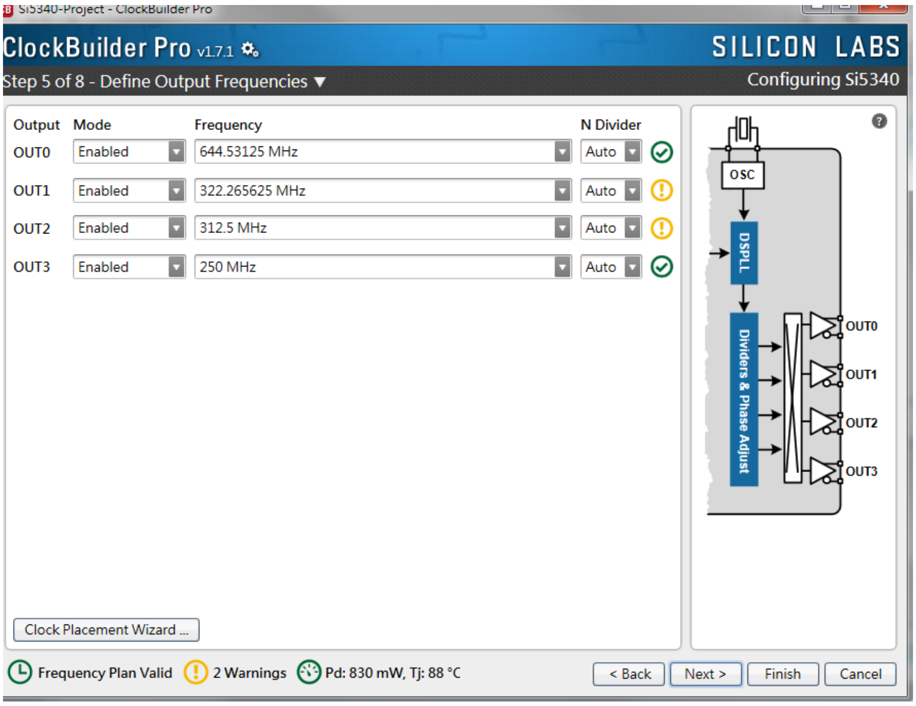



**Figure 5-6 Define Output Clock Frequencies on ClockBuilder Pro Wizard**

3. After the setting is completed, ClockBuider Pro Wizard generates a Design Report(text), which contains users setting frequency corresponding register value (See **Figure 5-7**).

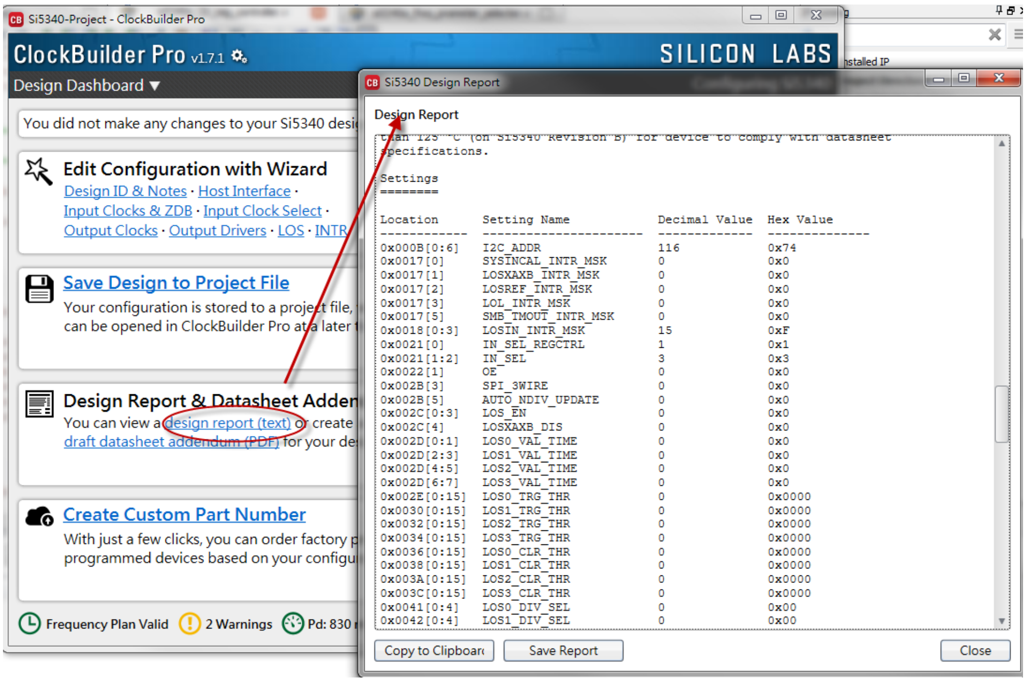



**Figure 5-7 Open Design Report on ClockBuilder Pro Wizard**

4. Open Si5340 control IP sub-module “si5340a_i2c_reg_controller.v “ as shown in**Figure5-8**, refer Design Report parameter to modify sub-module corresponding register value (See **Figure 5-9**).

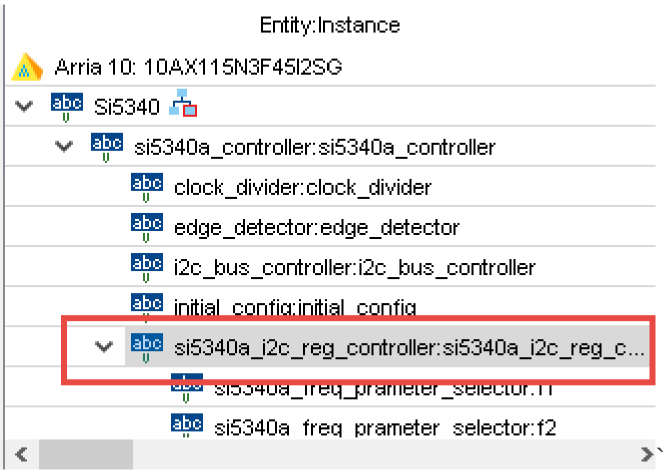



**Figure 5-8 Sub-Module file “si5340a_i2c_reg_controller.v”**

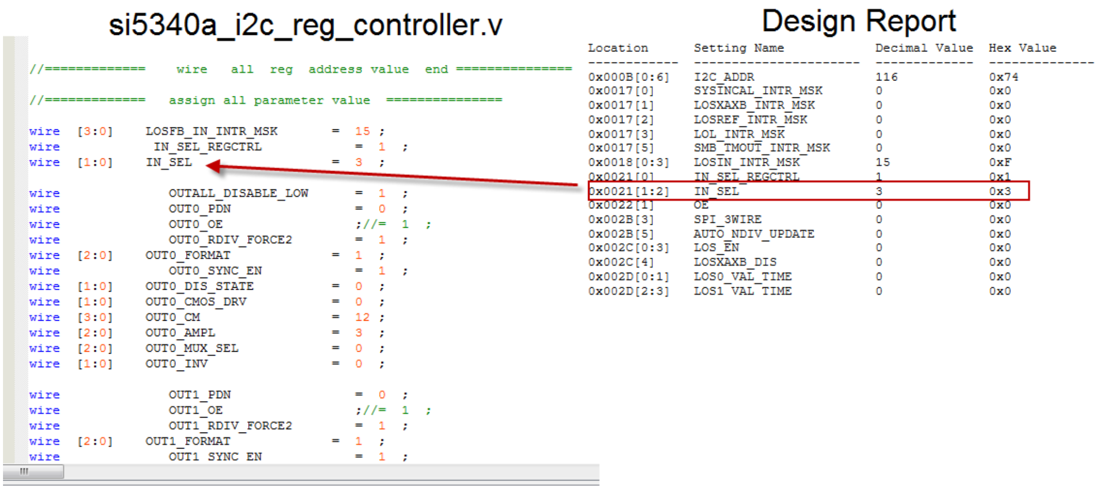



**Figure 5-9 Modify Si5340 Control IP Base on Design Report**

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

Note :

1. 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 5-10** 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 TMP441, Si5340A/B, and power monitor LTC2945) 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.

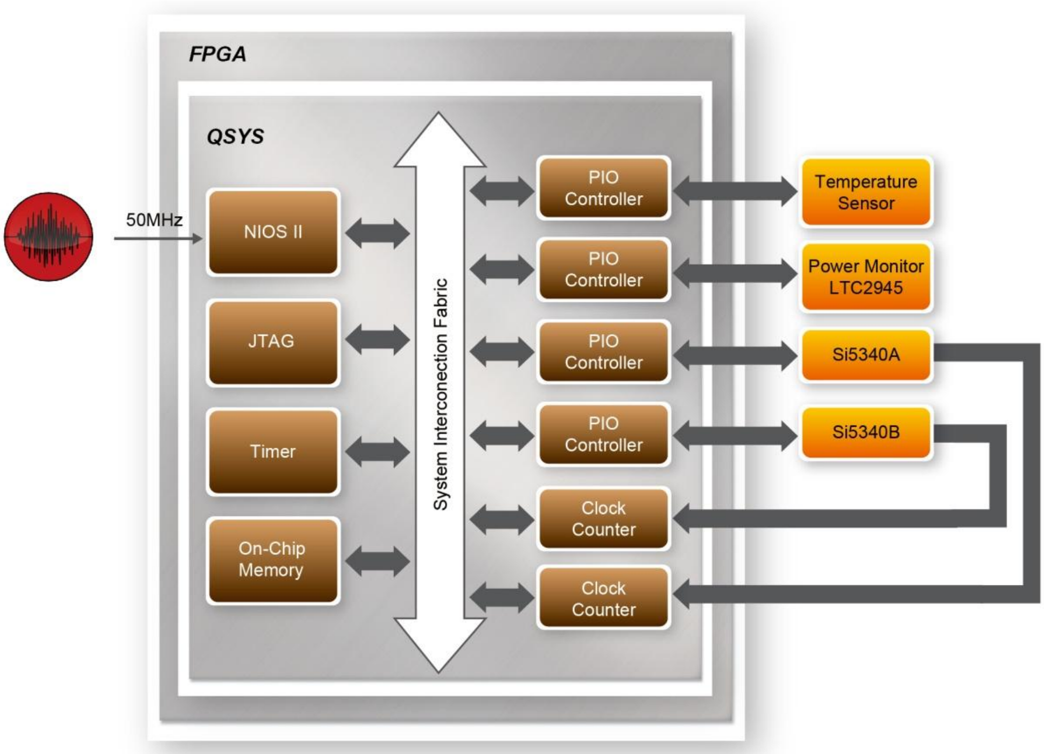



**Figure 5-10 Block diagram of the Nios II Basic Demonstration**

The program provides a menu in nios-terminal, as shown in **Figure 5-11** 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.

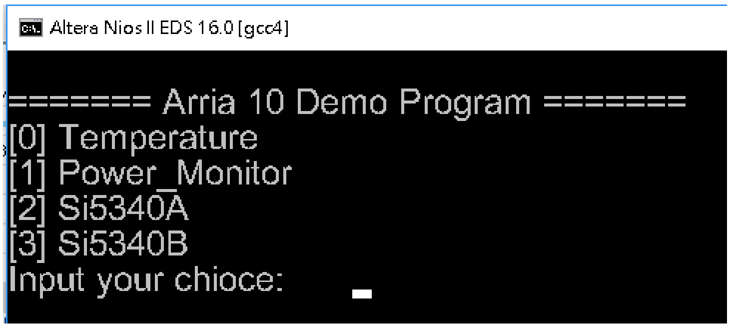



**Figure 5-11 Menu of Demo Program**

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 (LTC2945) 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 sense resistor R96 (0.006 Ω) for LTC2945 in the circuit, when power on the TR10a-HL2 board, there will be a voltage drop (named ΔSENSE Voltage) on R96. Based on sense resistors, the program of power monitor can calculate the associated voltage, current and power consumption from the LTC2945 through the I2C interface. Please note the device I2C address is 0xD4.

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 II 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 QDRIIA/B/C/D/E/F 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 II 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 5-12**.
- 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 5-13**.
- 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/D REFCLK, as shown in **Figure 5-14**.
- For programmable PLL Si5340B test, please input key ‘3’ and press ‘Enter’ in the nios-terminal first, then select the desired output frequency of QDRIIA/B/C/D/E/F REFCLK, as shown in **Figure 5-15**.

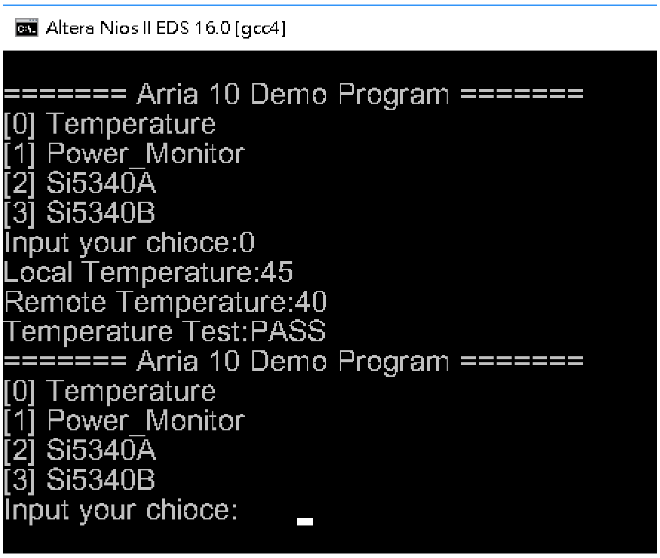



**Figure 5-12 Temperature Demo**

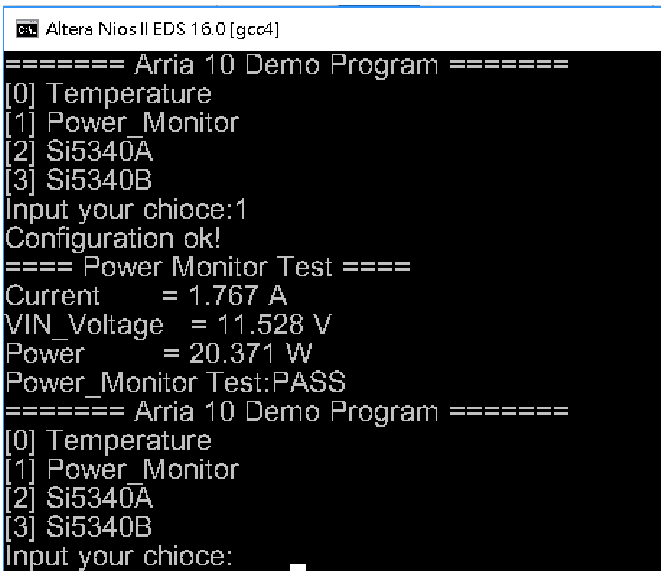



**Figure 5-13 power monitor Demo**

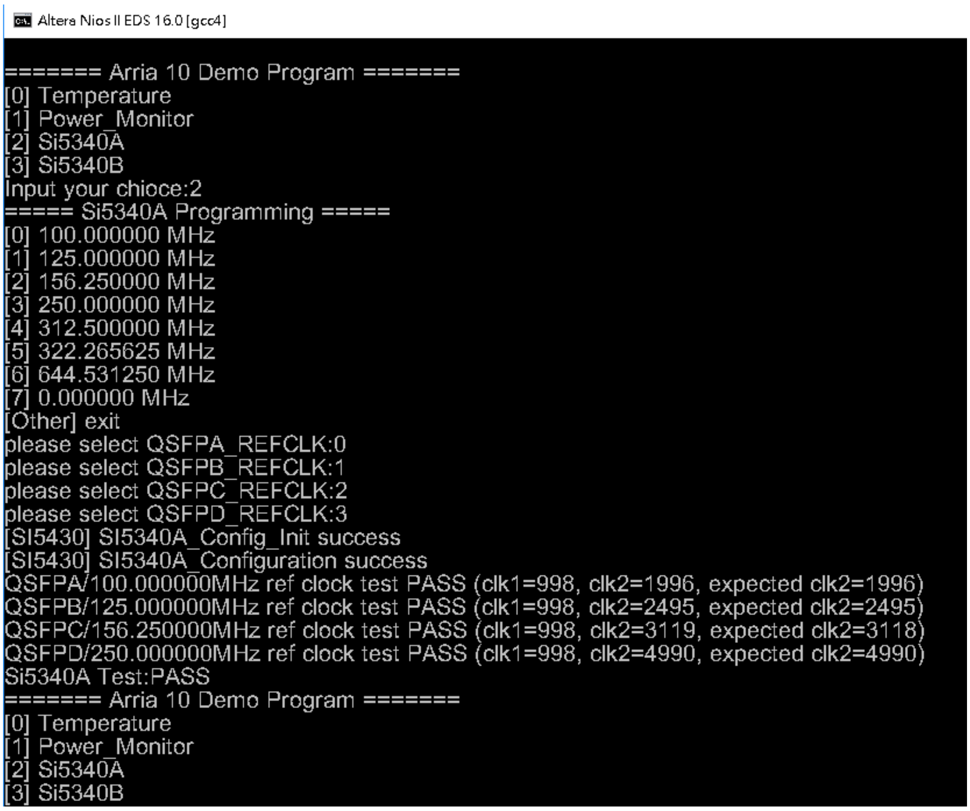



**Figure 5-14 Si5340A Demo**

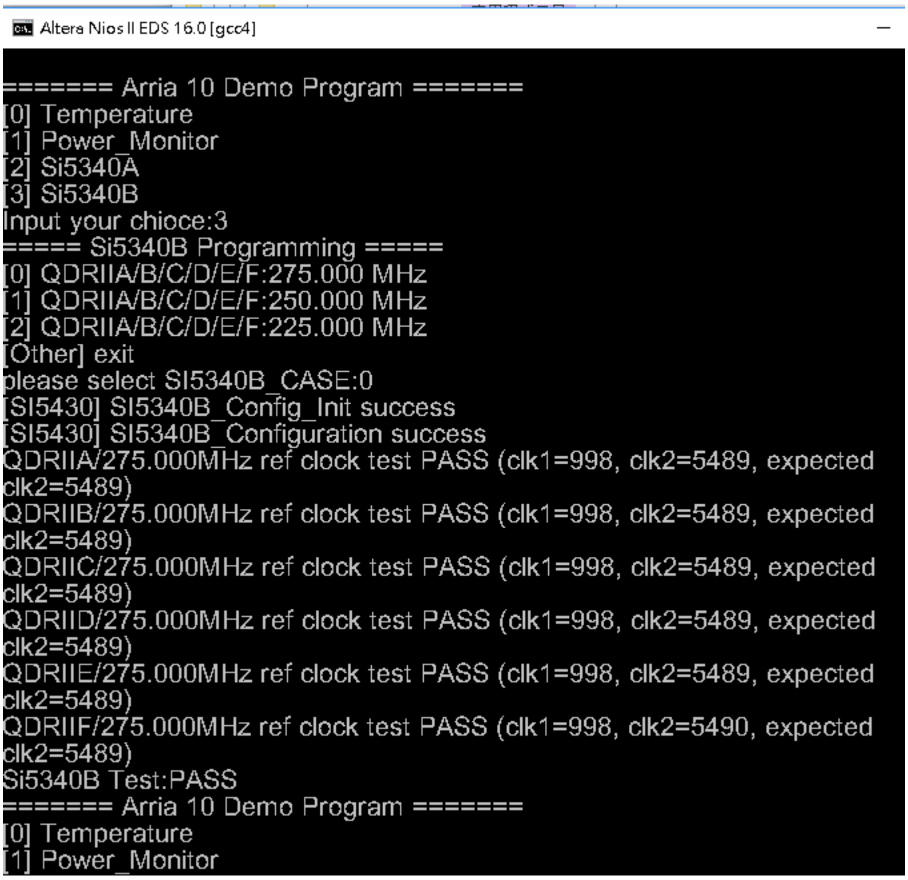



**Figure 5-15 Si5340B Demo**

## 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 5-16** 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.

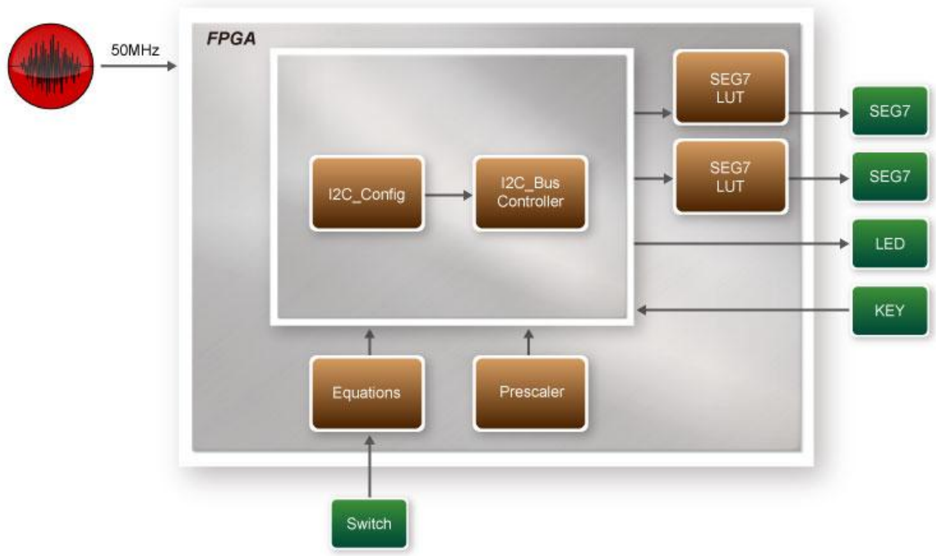



**Figure 5-16 Block diagram of the fan speed control demonstration**

## 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.

| 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 |

**Table 5-5 Alarm-Enable Resgister Bit Masks**

## Design Tools

- 64-bit Quartus II v16.0.2
- Demonstration Source Code
- Project Directory: Demonstration\Fan
- Bit Stream: TR10A_HL_golden_top.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: TR10A_HL_golden_top.sof

## Demonstration Setup

- Make sure Quartus II is installed on the host PC.
- Connect the TR10a-HL2 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 SW0.1is 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.

*Chapter 6*

# Memory Reference Design

This chapter will show two 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 six banks of QDRII+ SRAM
- QDRII+ SRAM Test by Nios II: Full test of six banks of QDRII+ SRAM with Nios II

Note. 64-Bit Quartus16.0.2 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 six 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** shows the function block diagram of the demonstration. The six 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.

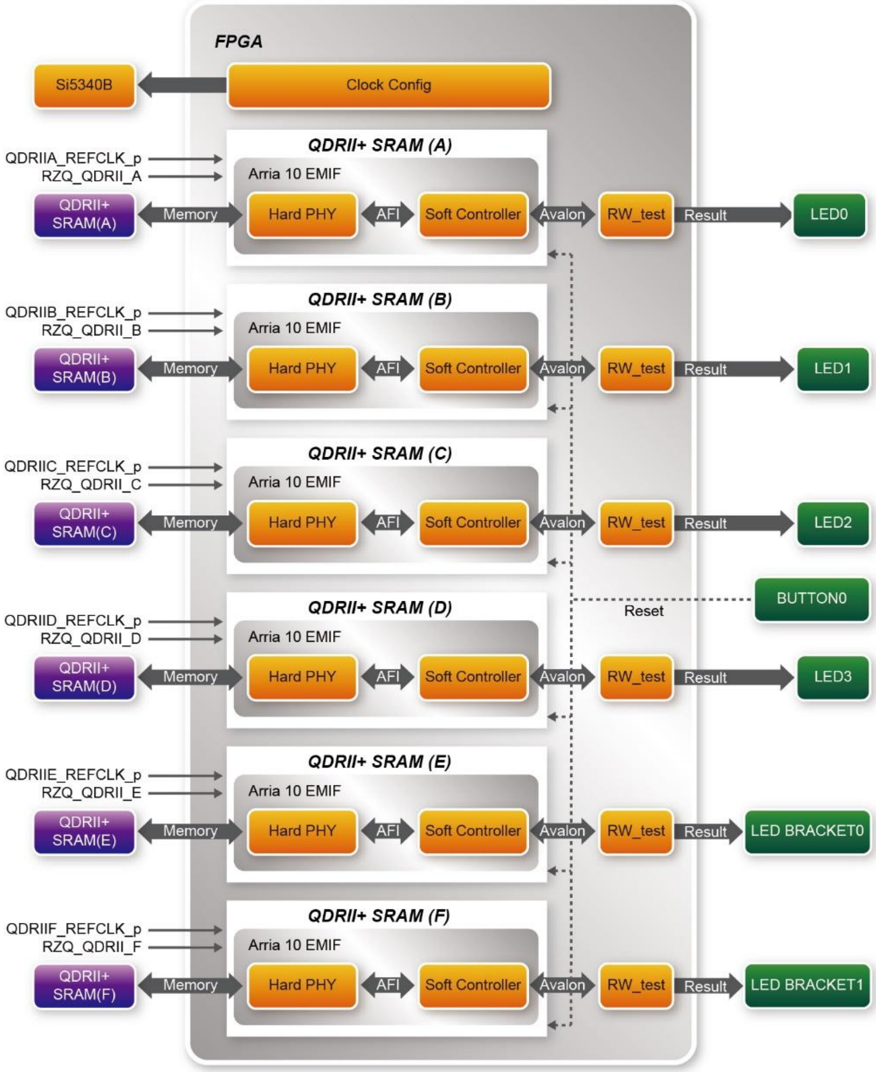



**Figure 6-1 Function Block Diagram of the QDRII+ SRAM x4 Demonstration**

The QDRIIA/B/C/D/E/F_REFCLK is generated from Si5340B which configured 275MHz for QDRII+ 550MHz by Clock Config module. QDRIIA/B/C/D/E/F_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 II 16.0.2
- Demonstration Source Code
- Project directory: QDRII_x6_Test_550MHz
- Bit stream used: TR10A_HL_golden_top.sof
- Demonstration Batch File

Demo Batch File Folder: QDRIIx4_Test\demo_batch

The demo batch files include the followings:

- Batch file for USB-Blaster II: test.bat,
- FPGA configuration file: TR10A_HL_golden_top.sof
- Demonstration Setup
- Make sure Quartus II 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_ub2.bat” under the batch file folder, QDRII_x6_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+ SRAMI test result |
| LED3 | QDRII+ SRAM(D) test result |
| Bracket LED0 | QDRII+ SRAMI test result |
| Bracket LED1 | QDRII+ SRAM(F) test result |

## 6.2 QDRII+ SRAM Test by Nios II

This demonstration hardware and software designs are provided to illustrate how to perform QDRII+ SRAM memory access in QSYS. We describe how the Altera's "Arria 10 External Memory Interfaces" IP is used to access the six QDRII+ SRAM on the FPGA board, and how the Nios II processor is used to read and write the SRAM for hardware verification. The QDRII+ SRAM controller handles the complex aspects of using QDRII+ SRAM by initializing the memory devices, managing SRAM banks, and keeping the devices refreshed at appropriate intervals.

### ■ System Block Diagram

**Figure 6-2** shows the system block diagram of this demonstration. The QSYS system requires one 50 MHz and six 550MHz clock source. The six 550MHz 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 six 550MHz clock are used as reference clocks for the QDRII+ controllers. There are six QDRII+ Controllers are used in the demonstrations. Each controller is responsible for one QDRII+ SRAM. Each QDRII+ controller is configured as a 8 MB QDRII+ controller. 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.

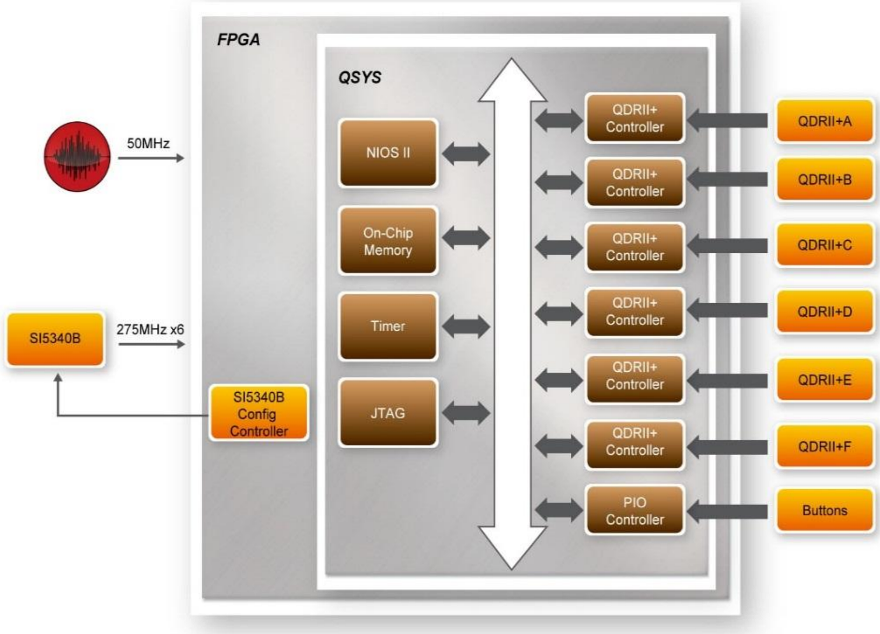



**Figure 6-2 Block diagram of the QDRII+ Demonstration**

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the whole 8 MB of SRAM. Then, it calls Nios II system function, *alt_dcache_flush_all()*, to make sure all data has been written to SRAM. Finally, it reads data from SRAM for data verification. The program will show progress in JTAG-Terminal when writing/reading data to/from the SRAM. When verification process is completed, the result is displayed in the JTAG-Terminal.

## Design Tools

- Quartus II 16.0.2
- Nios II Eclipse 16.0.2

## Demonstration Source Code

- Quartus Project directory: NIOS_QDRII_x6_550
- Nios II Eclipse: NIOS_QDRII_x6_550\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_QDRII_x6_550\demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaseter II: test.bat, test.sh
- FPGA Configure File: NIOS_QDRII_x6_550.sof
- Nios II Program: TEST_QDRII.elf

## Demonstration Setup

Please follow below procedures to setup the demonstartons.

- Make sure Quartus II and Nios II are installed on your PC.
- Make sure both QDRII+ SRAMs 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_QDRII_x6_550\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 SRAM verify process. Press Button0 for continued test.
- The program will display progressing and result information, as shown in **Figure 6-3**.

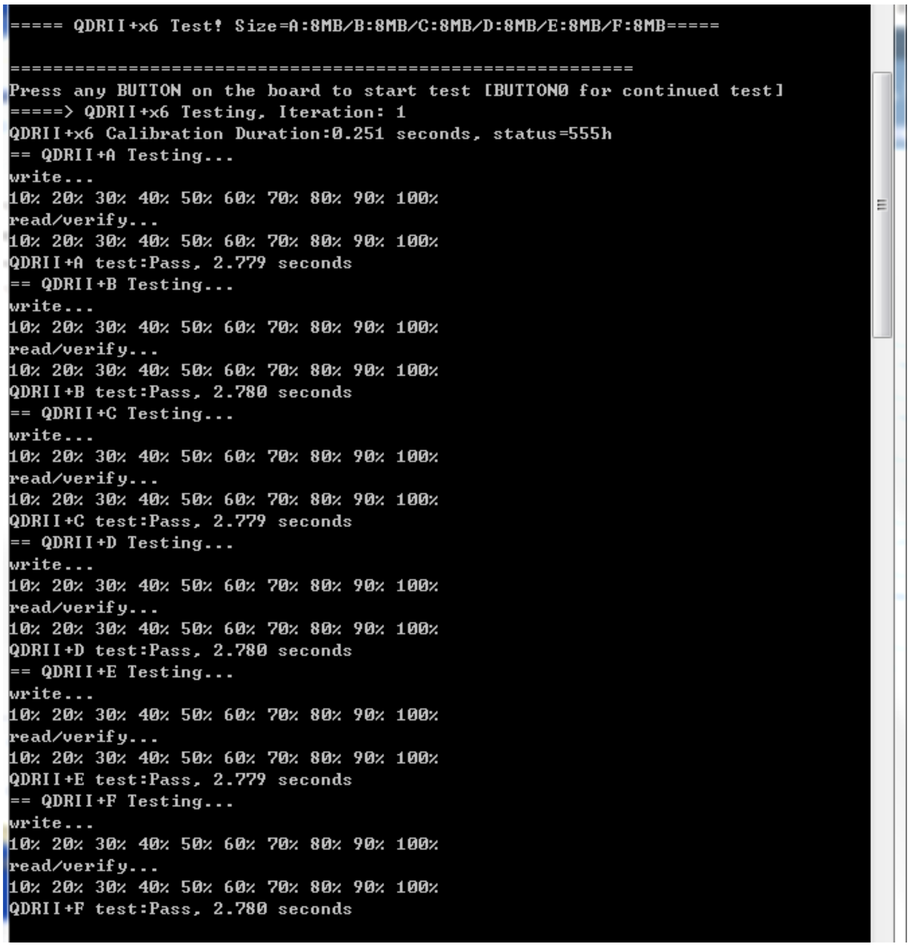



**Figure 6-3 Progress and Result Information for the QDRII+ Demonstration**

*Chapter 7*

# PCI Express Reference Design

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 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.

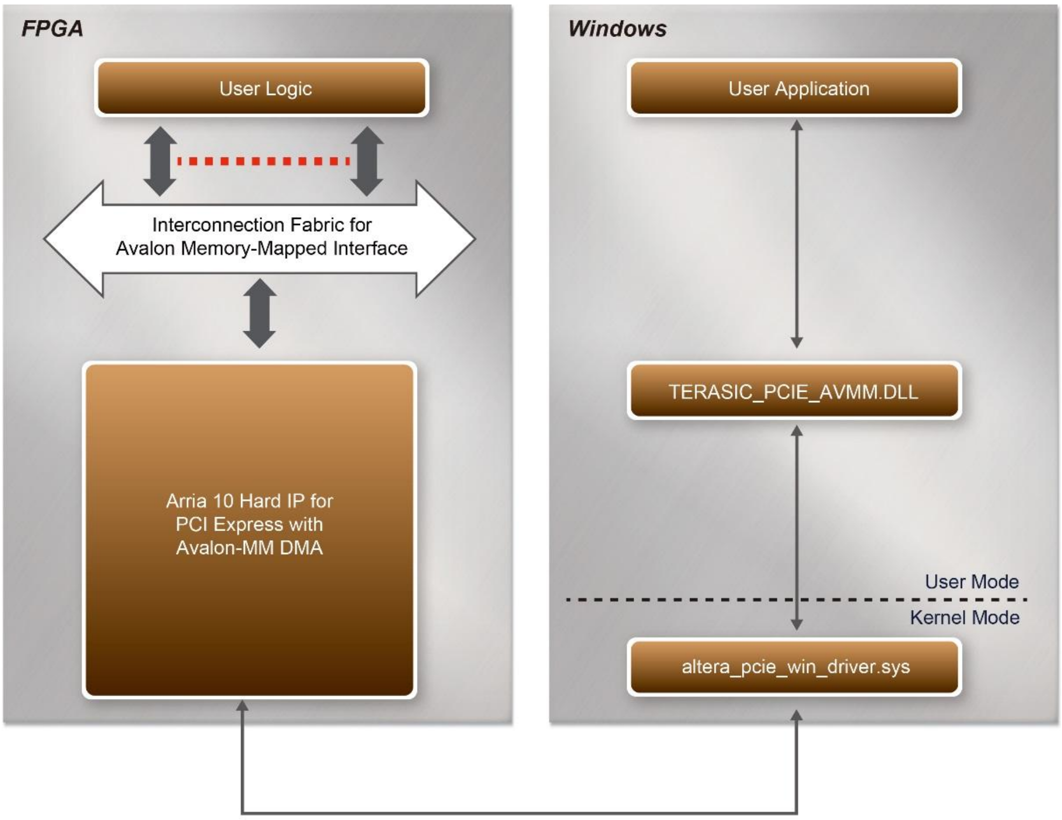



**Figure 7-1 Infrastructure of PCI Express System**

## 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 Windows 7/Window XP 64-bit. The SDK is located in the "CDROM \demonstrations\PCIe_SW_KIT" folder which includes:

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

The kernel mode driver assumes the PCIe vender 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 vender 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 PCI Express driver incorporated in the DLL library is called "TERASIC_PCIE_AVMM.dll". Users can develop their applications based on this DLL. The "altera_pcie_win_driver.sys" kernel driver is provided by Altera.

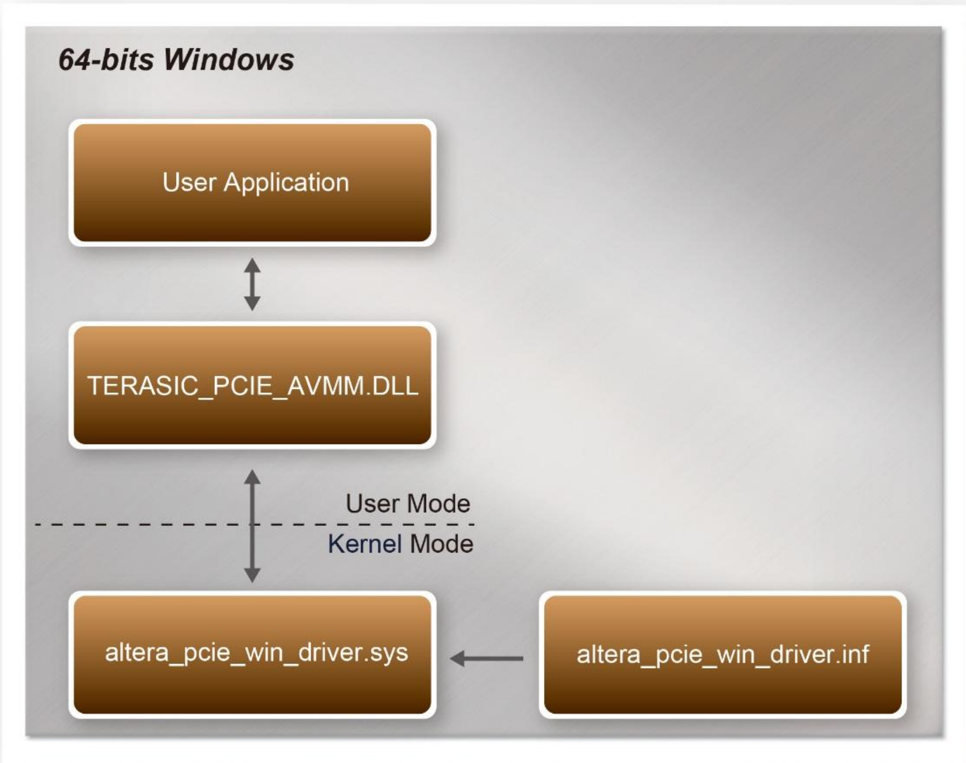



**Figure 7-2 PCI Express Software Stack**

## ■ Install PCI Express Driver on Windows

The PCIe driver is locate in the folder:

CDROM\Demonstrations\PCIe_SW_KIT\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 TR10a-HL2 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.

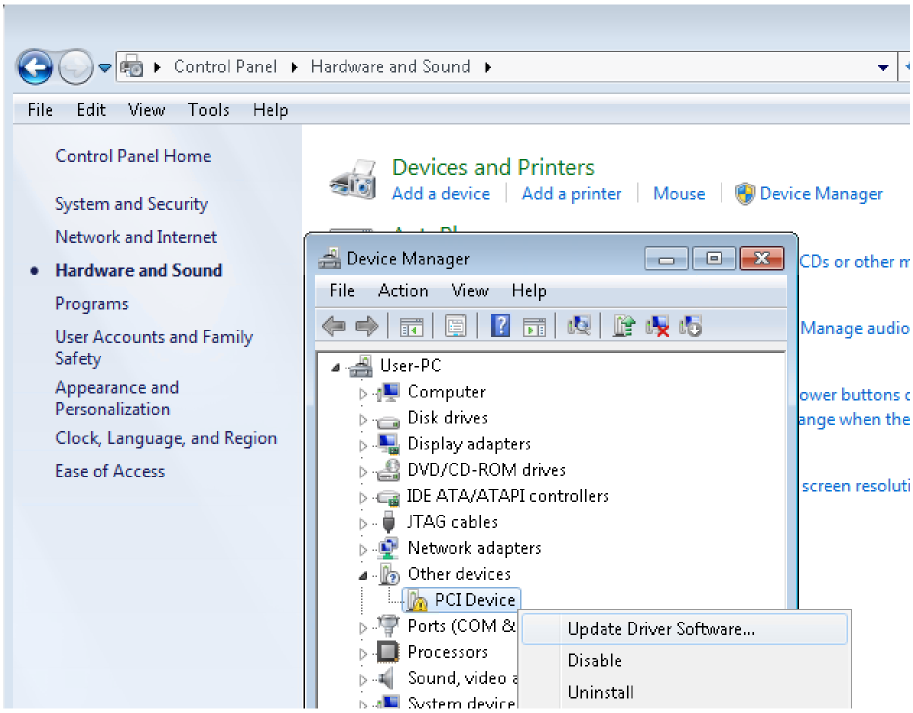



**Figure 7-3 Screenshot of launching Update Driver Software… dialog**

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**

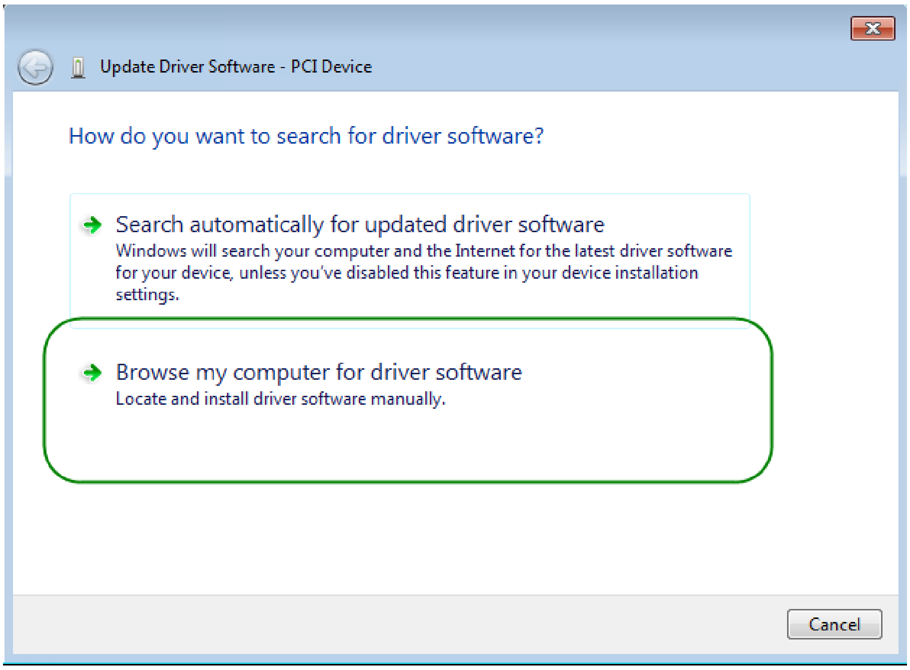



**Figure 7-4 Dialog of Browse my computer for driver software**

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

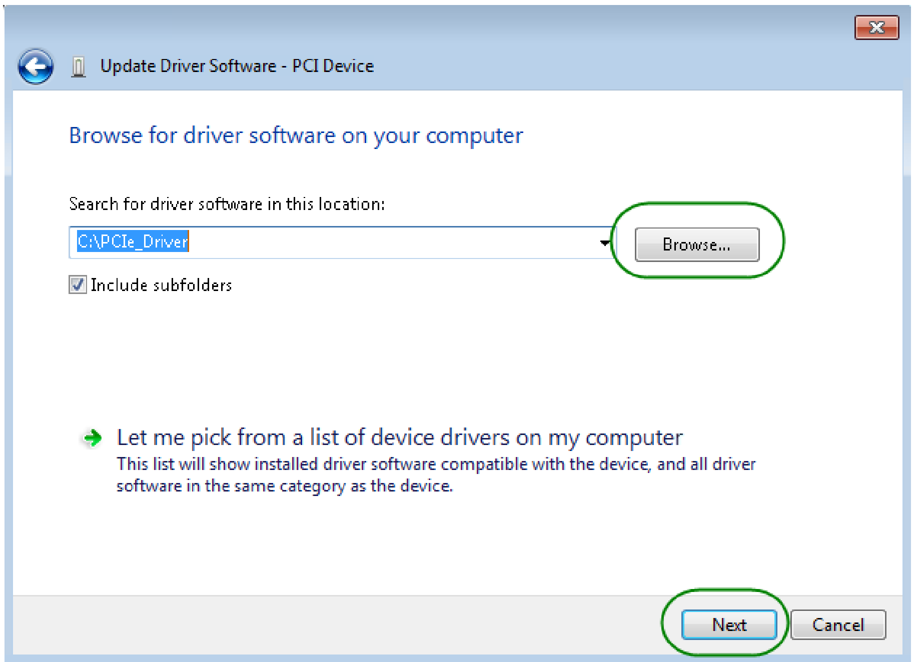



**Figure 7-5 Browse for driver software on your computer**

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

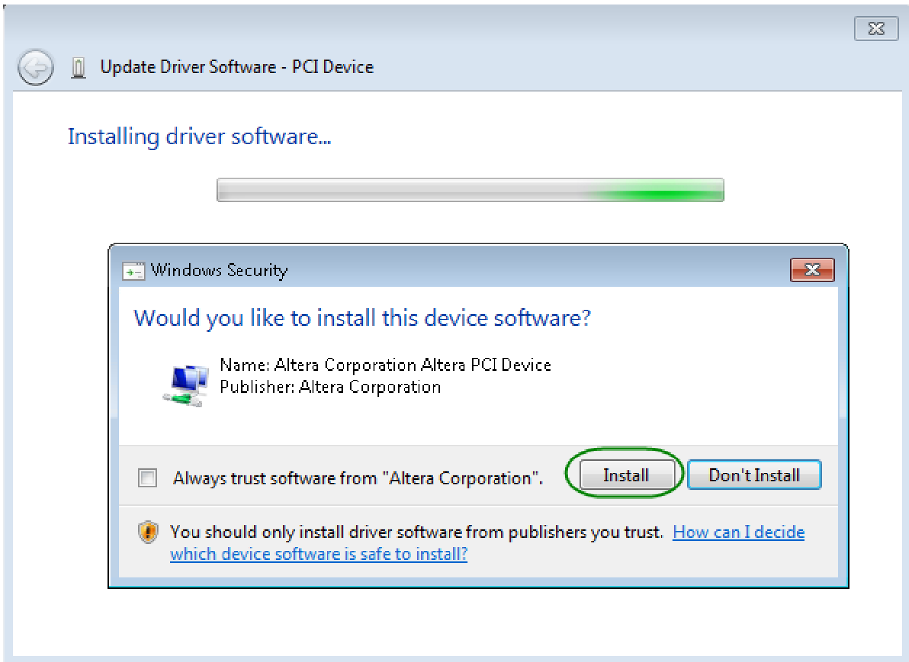



**Figure 7-6 Click Install in the dialog of Windows Security**

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

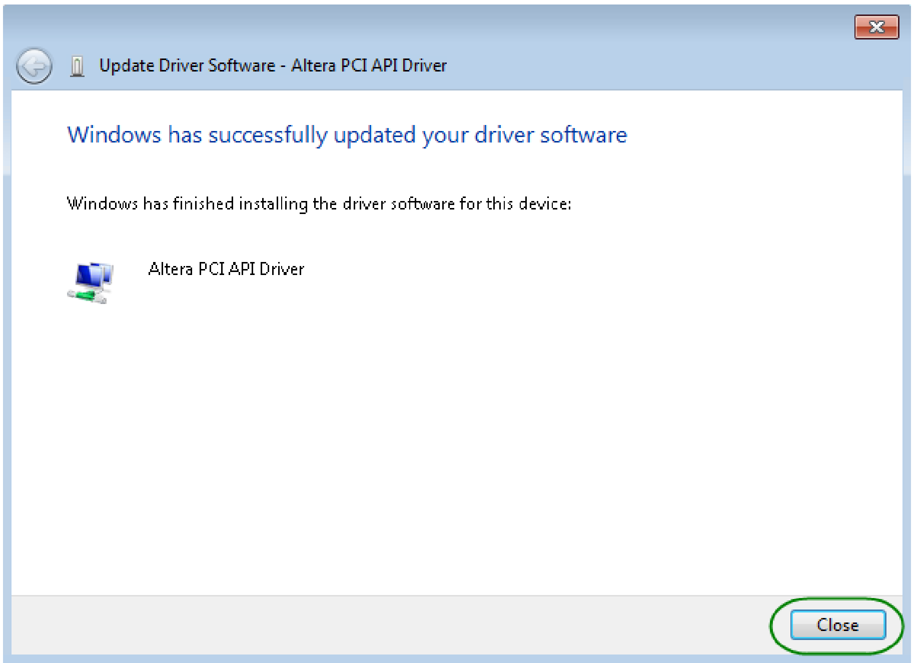



**Figure 7-7 Click Close when the installation of Altera PCI API Driver is complete**

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

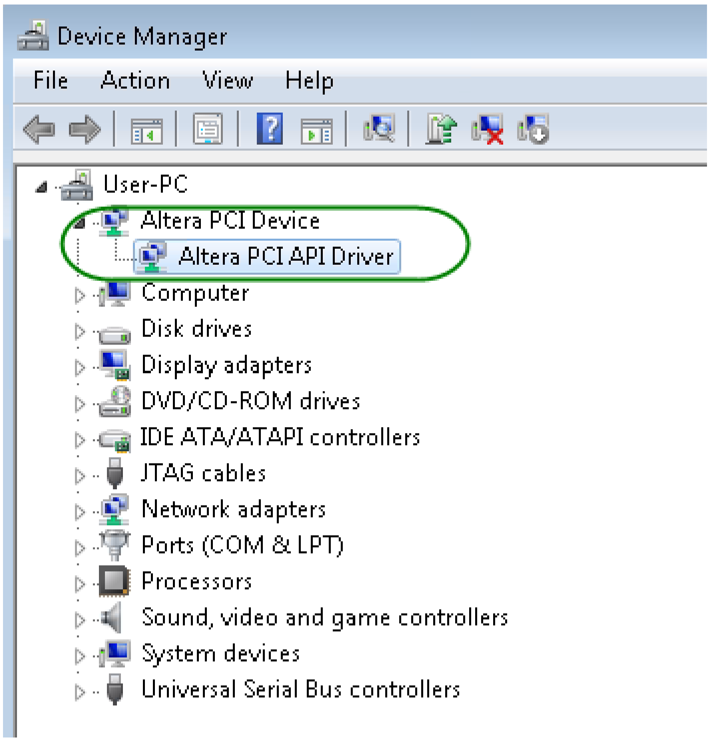



**Figure 7-8 Altera PCI API Driver in Device Manager**

## Create a Software Application

All the files needed to create a PCIe software application are located in the directory CDROM\demonstration\PCIe_SW_KIT\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:

## PCIE_Open

| |
|---|
| **Function:**<br>Open a specified PCIe card with vendor ID, device ID, and matched card index. |
| **Prototype:**<br>PCIE_HANDLE PCIE_Open(<br>WORD wVendorID,<br>WORD wDeviceID,<br>WORD wCardIndex); |
| **Parameters:**<br>wVendorID:<br>Specify the desired vendor ID. A zero value means to ignore the vendor ID.<br>wDeviceID:<br>Specify the desired device ID. A zero value means to ignore the device ID.<br>wCardIndex:<br>Specify the matched card index, a zero based index, based on the matched verder ID and device ID. |
| **Return Value:**<br>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.<br>This handle value is used as a parameter for other functions, e.g. PCIE_Read32.<br>Users need to call PCIE_Close to release handle once the handle is no more used. |

## PCIE_Close

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

## PCIE_Read32

| |
|---|
| **Function:**<br>Read a 32-bit data from the FPGA board. |
| **Prototype:** |

bool PCIE_Read32(
    PCIE_HANDLE hPCIE,
    PCIE_BAR PcieBar,
    PCIE_ADDRESS PcieAddress,
    DWORD * pdwData);

**Parameters:**

hPCIE:

A PCIe handle return by PCIE_Open function.

PcieBar:

Specify the target BAR.

PcieAddress:

Specify the target address in FPGA.

pdwData:

A buffer to retrieve the 32-bit data.

**Return Value:**

Return TRUE if read data is successful; otherwise FALSE is returned.

## PCIE_Write32

**Function:**

Write a 32-bit data to the FPGA Board.

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

**Prototype:**

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

**Parameters:**

hPCIE:

A PCIe handle return by PCIE_Open function.

PcieBar:

Specify the target BAR.

PcieAddress:

Specify the target address in FPGA.

dwData:

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

**Return Value:**

Return TRUE if write data is successful; otherwise FALSE is returned.

## PCIE_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 hPCIE,
    PCIE_LOCAL_ADDRESS LocalAddress,
    void *pBuffer,
    DWORD dwBufSize
    );
```

**Parameters:**

hPCIE:

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 hPCIE,
    PCIE_LOCAL_ADDRESS LocalAddress,
    void *pData,
    DWORD dwDataSize
    );
```

| |
|---|
| **Parameters:**<br>hPCIE:<br>A PCIe handle return by PCIE_Open function.<br>LocalAddress:<br>Specify the target memory mapped address in FPGA.<br>pData:<br>A pointer to a memory buffer to store the data which will be written to FPGA.<br>dwDataSize:<br>Specify the byte number of data which will be written to FPGA. |
| **Return Value:**<br>Return TRUE if write data is successful; otherwise FALSE is returned. |

### ■ PCIE_ConfigRead32

| |
|---|
| **Function:**<br>Read PCIe Configuration Table. Read a 32-bit data by given a byte offset. |
| **Prototype:**<br>bool PCIE_ConfigRead32 (<br>PCIE_HANDLE hPCIE,<br>DWORD Offset,<br>DWORD *pdwData<br>); |
| **Parameters:**<br>hPCIE:<br>A PCIe handle return by PCIE_Open function.<br>Offset:<br>Specify the target byte of offset in PCIe configuration table.<br>pdwData:<br>A 4-bytes buffer to retrieve the 32-bit data. |
| **Return Value:**<br>Return TRUE if read data is successful; otherwise FALSE is returned. |

## 7.4 PCIe 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_funcdamental\demo_batch

The folder includes following files:

- FPGA Configuration File: PCIe_Fundamental.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 PCIE_Fundamental.sof by executing the test.bat.
3. Install PCIe driver if necessary. The driver is located in the folder:

   CDROM\Demonstration\PCIe_SW_KIT\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**.

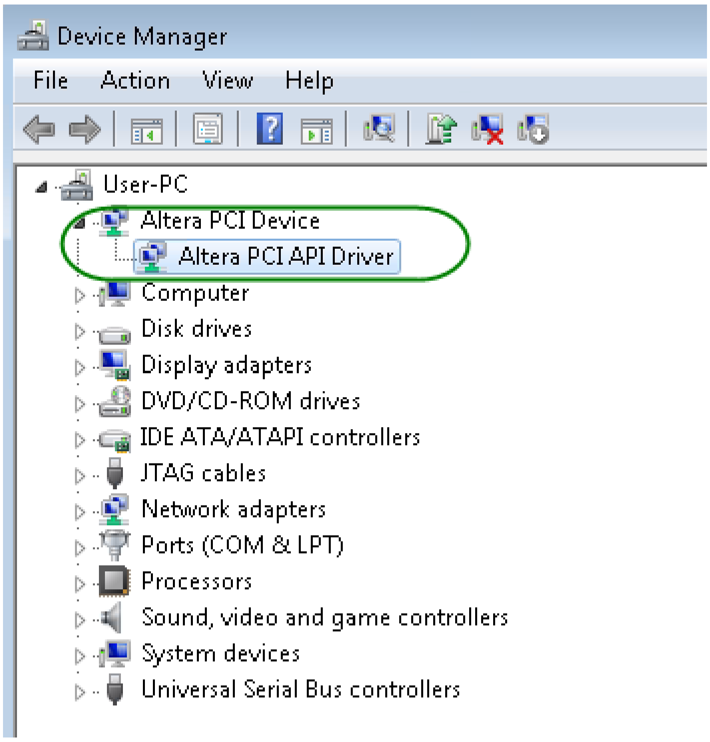



**Figure 7-10 Screenshot for PCIe Driver**

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

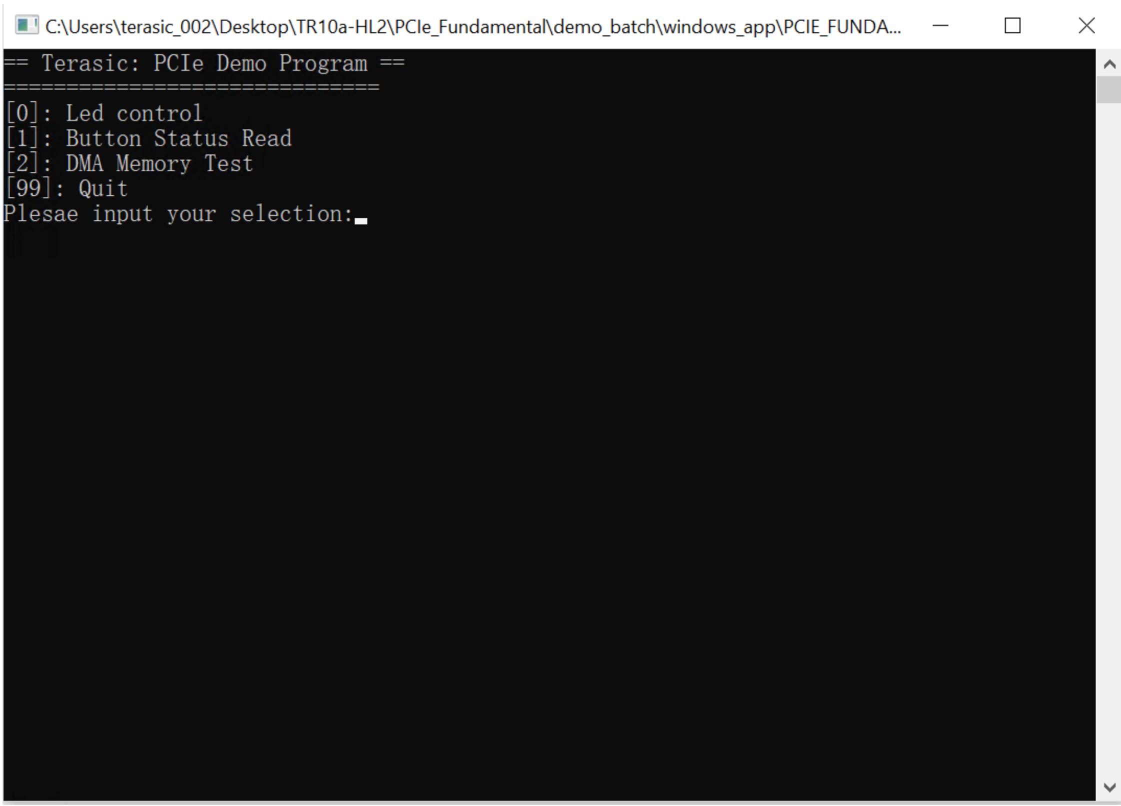



**Figure 7-11 Screenshot of Program Menu**

7. Type 0 followed by a ENTERY 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.

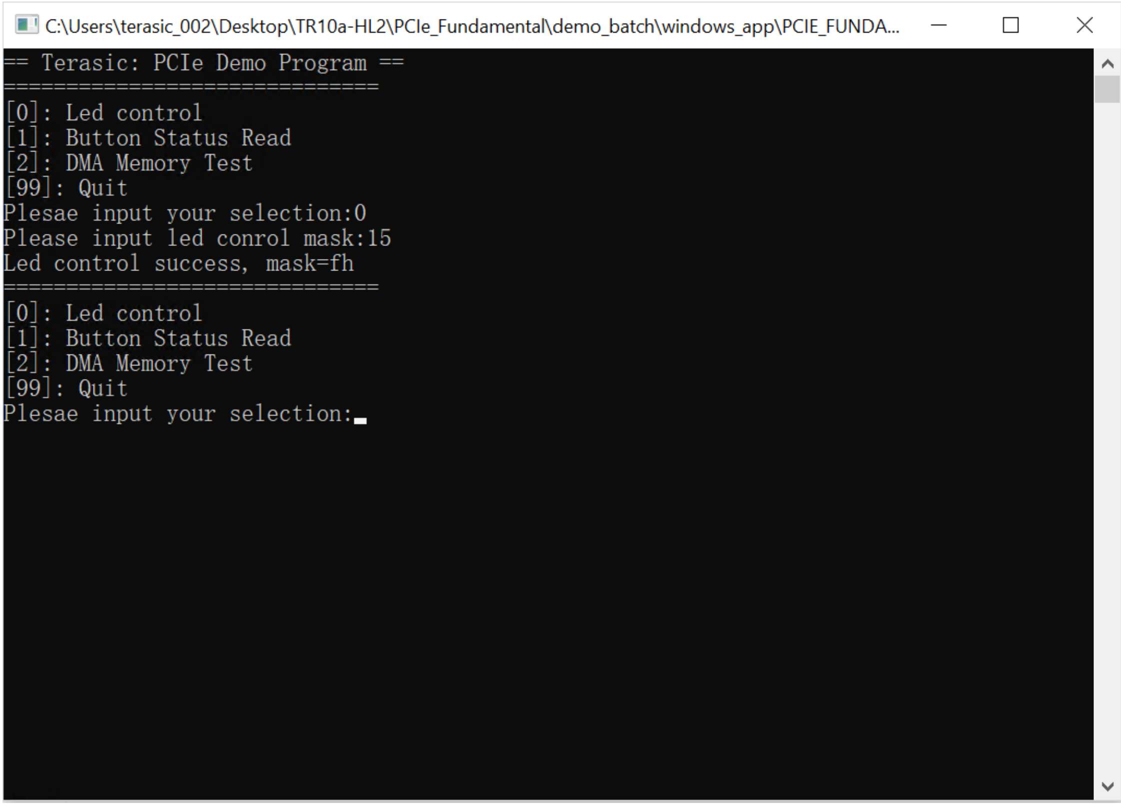



**Figure 7-12 Screenshot of LED Control**

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

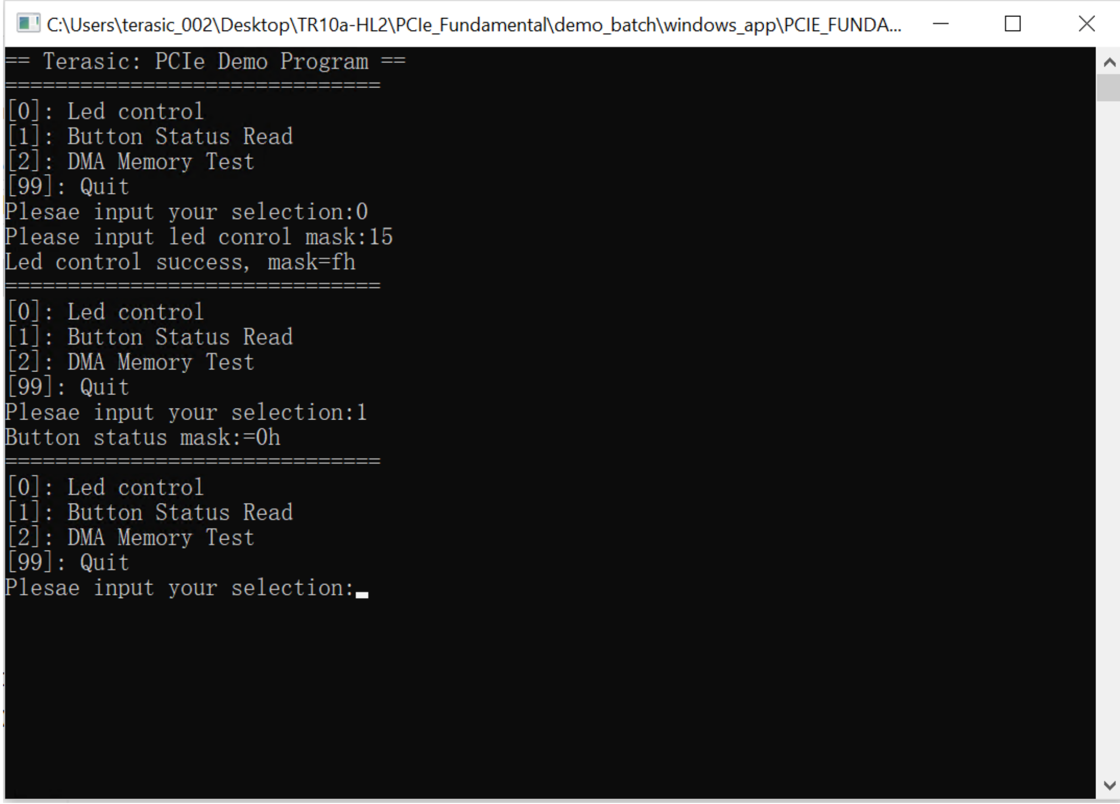



**Figure 7-13 Screenshot of Button Status Report**

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

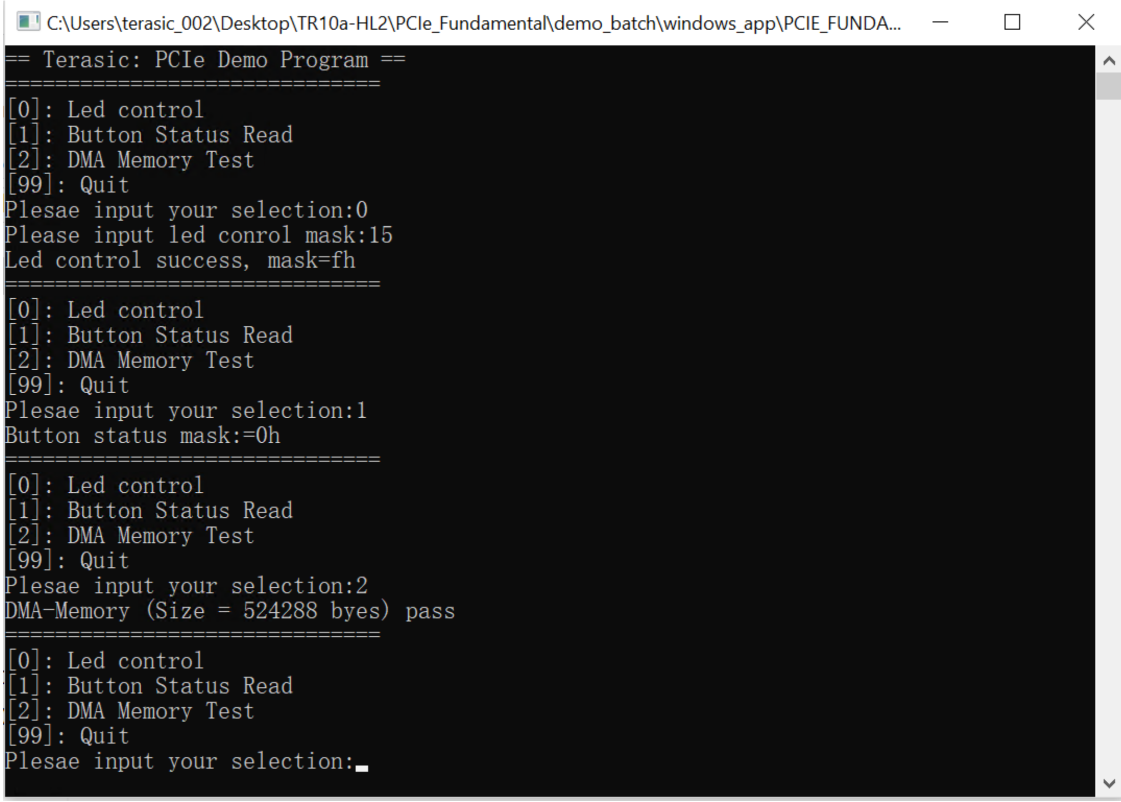



**Figure 7-14 Screenshot of DMA Memory Test Result**

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

## Development Tools

- Quartus II 16.0
- Visual C++ 2012

## Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIe_Fundamental
- Visual C++ Project: Demonstrations\PCIe_SW_KIT\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.

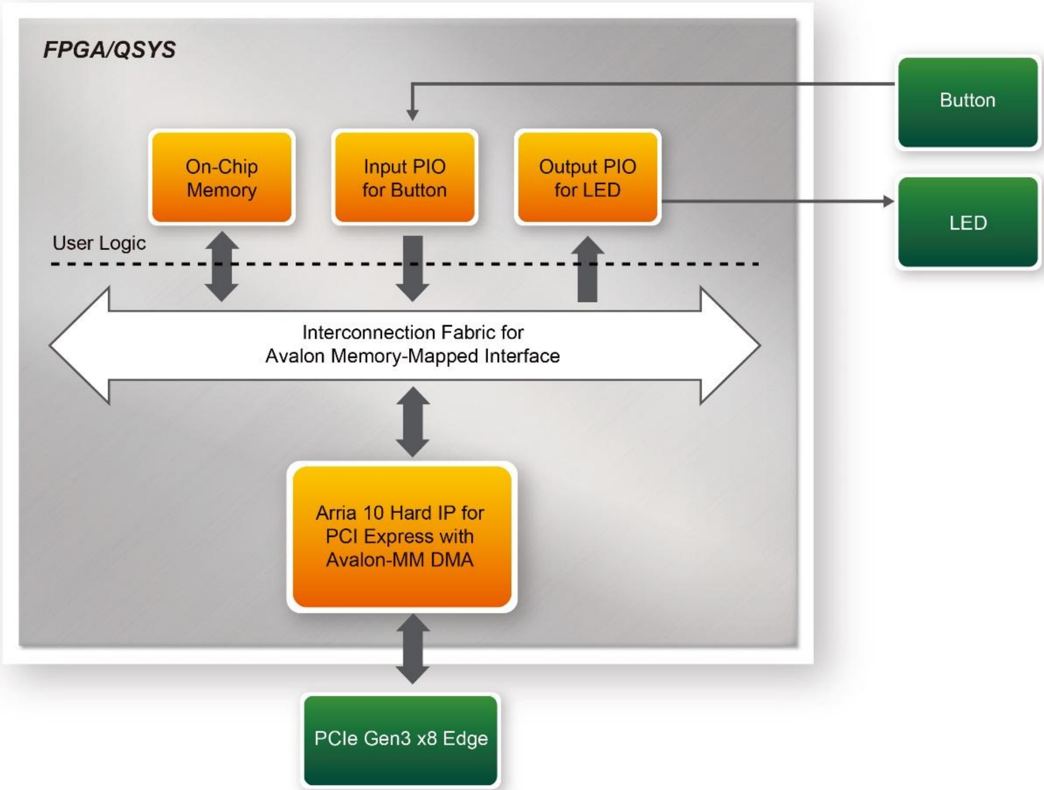



**Figure 7-15 Hardware block diagram of the PCIe reference design**

## 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 TERAISC_PCIE_AVMM.DLL |
| 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_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 Vender 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,(DWORD)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);
```

# 7.5 PCIe Design – QDRII+

The application reference design shows how to add QDRII+ Memory Controllers for six QDRII+ SRAMs into the PCIe Quartus project based on the PCI_Fundamental Quartus project and perform 8MB data DMA for six SRAMs. 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_QDR\demo_batch

The folder includes following files:

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

## ■ Demonstration Setup

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

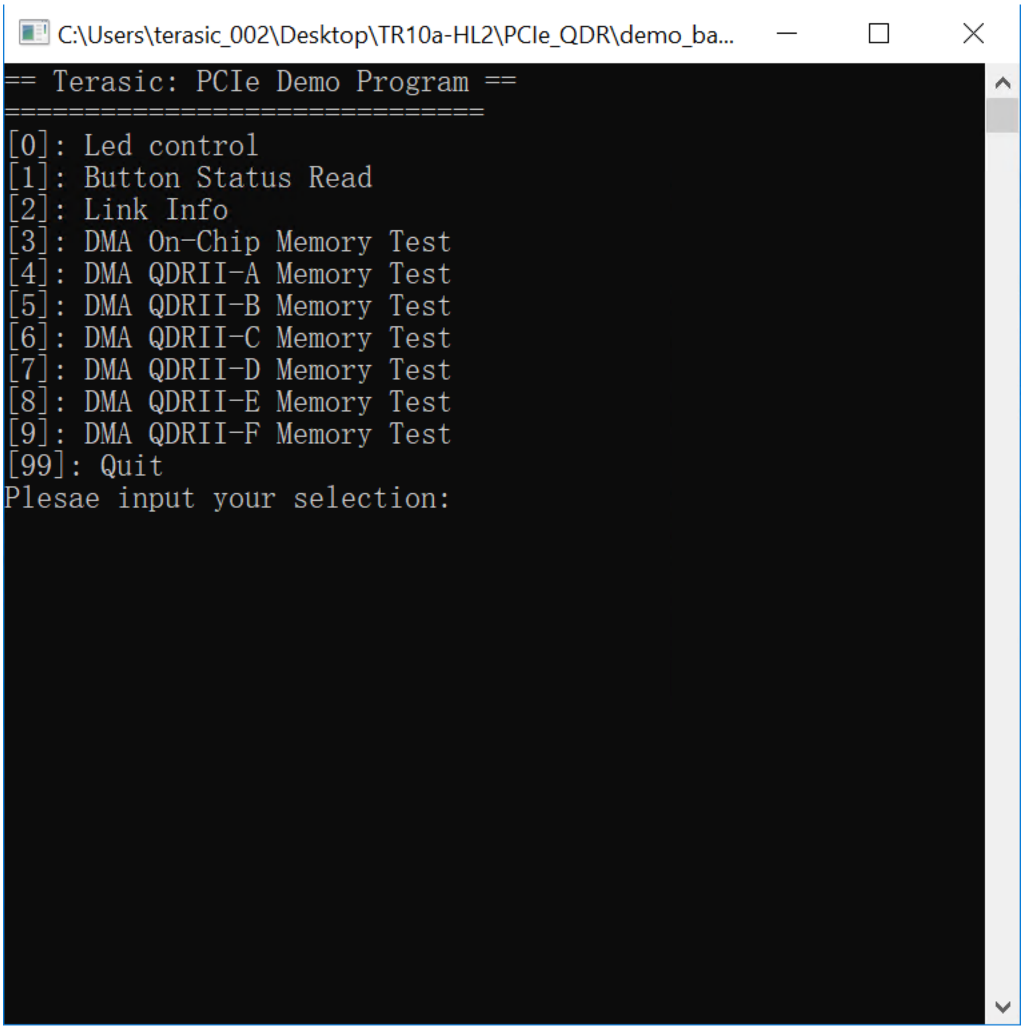



**Figure 7-16 Screenshot of Program Menu**

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

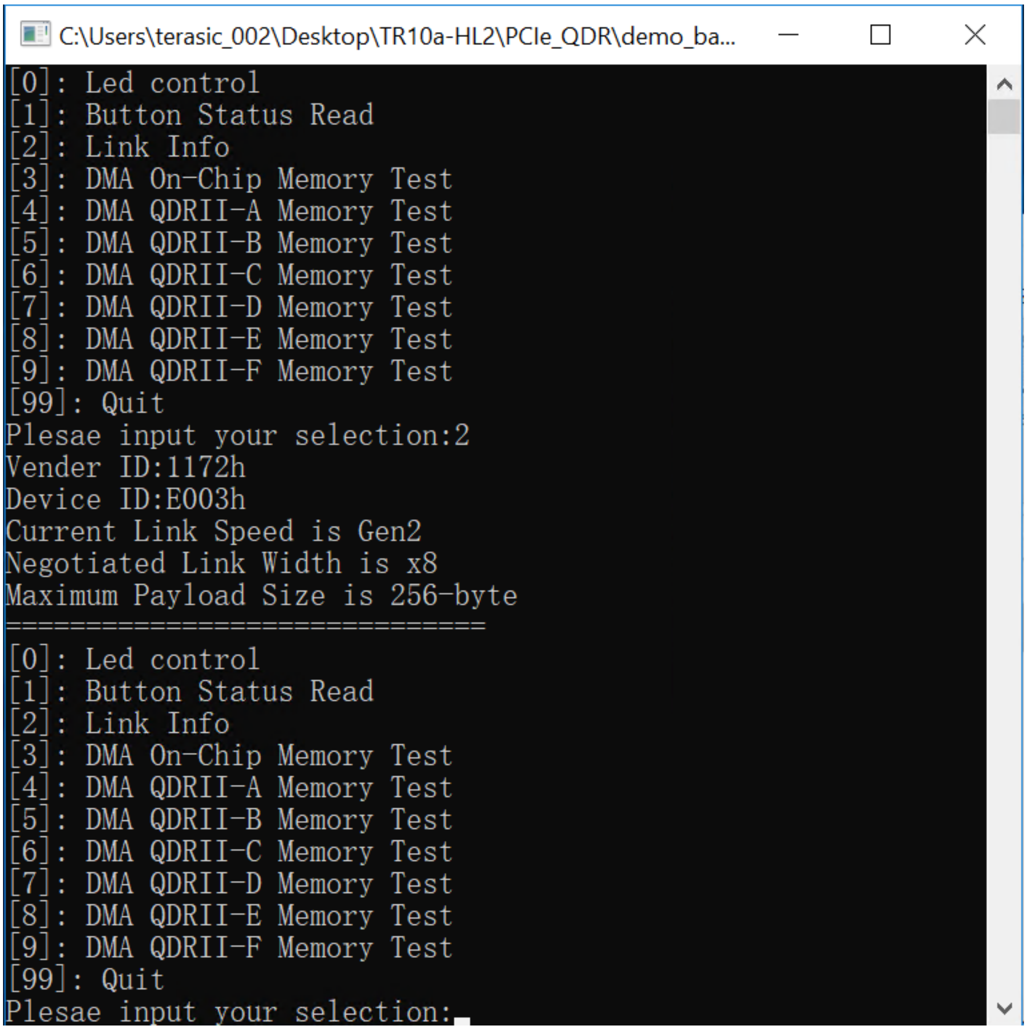



**Figure 7-17 Screenshot of Link Info**

8. Type 3 followed by an ENTERY 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**.

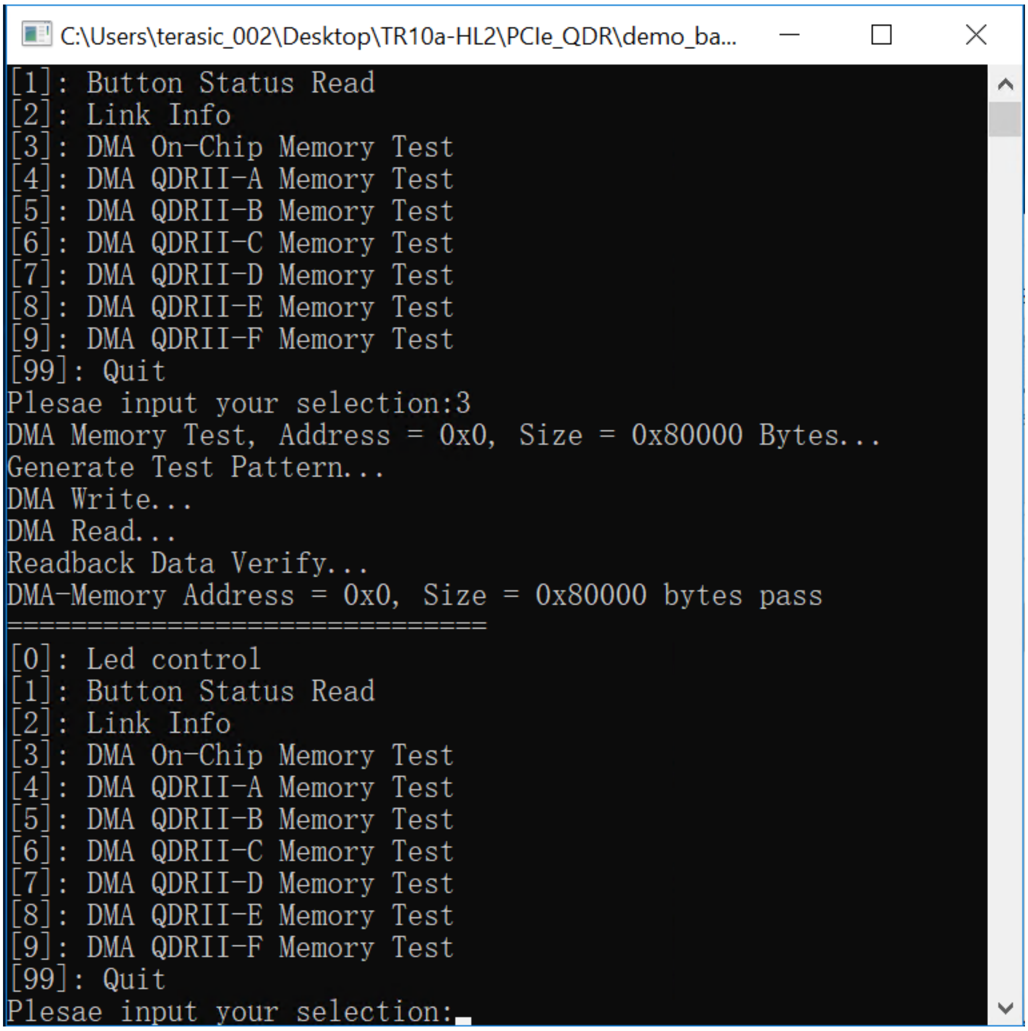



**Figure 7-18 Screenshot of On-Chip Memory DMA Test Result**

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

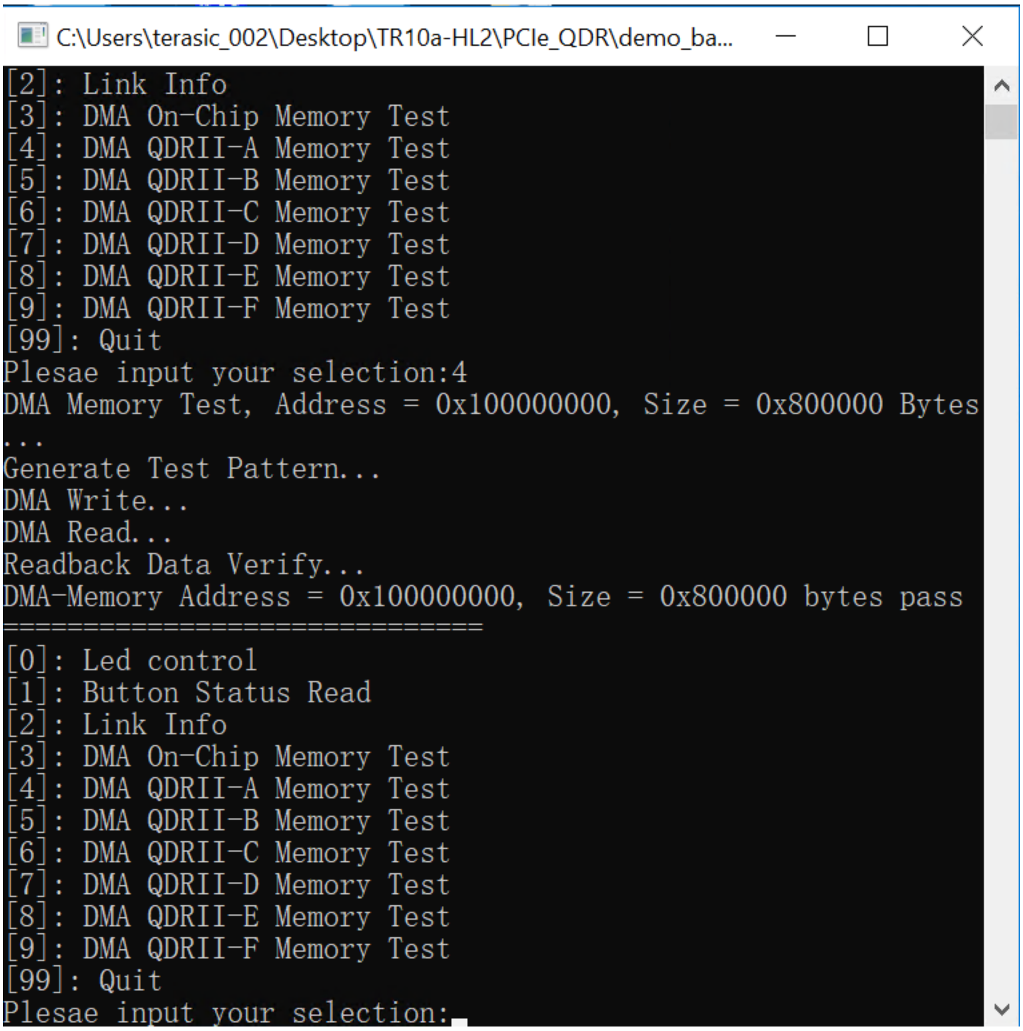



**Figure 7-19 Screenshot of QDRII-A Memory DAM Test Result**

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

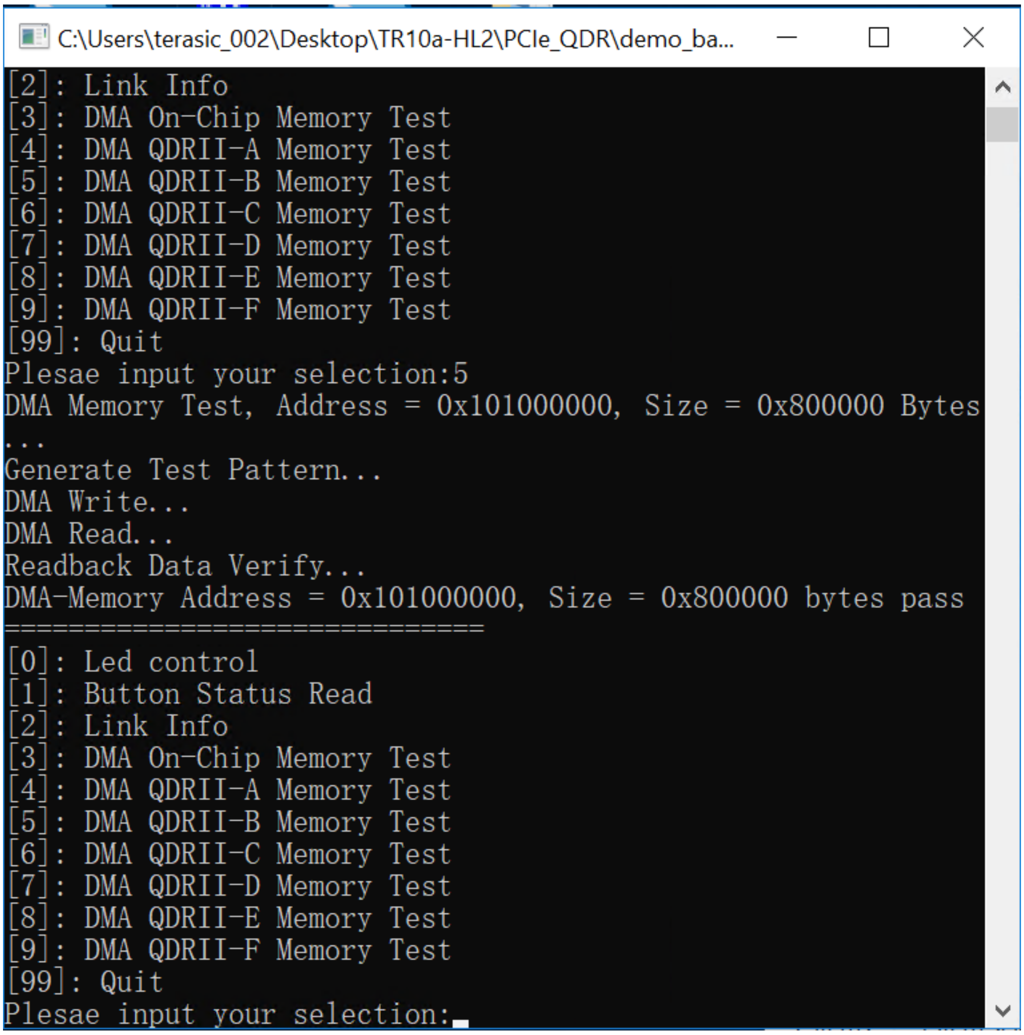



**Figure 7-20 Screenshot of QDRII-B Memory DAM Test Result**

11. Type 6 followed by an ENTERY key to select DMA QDRII-C Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-21**.

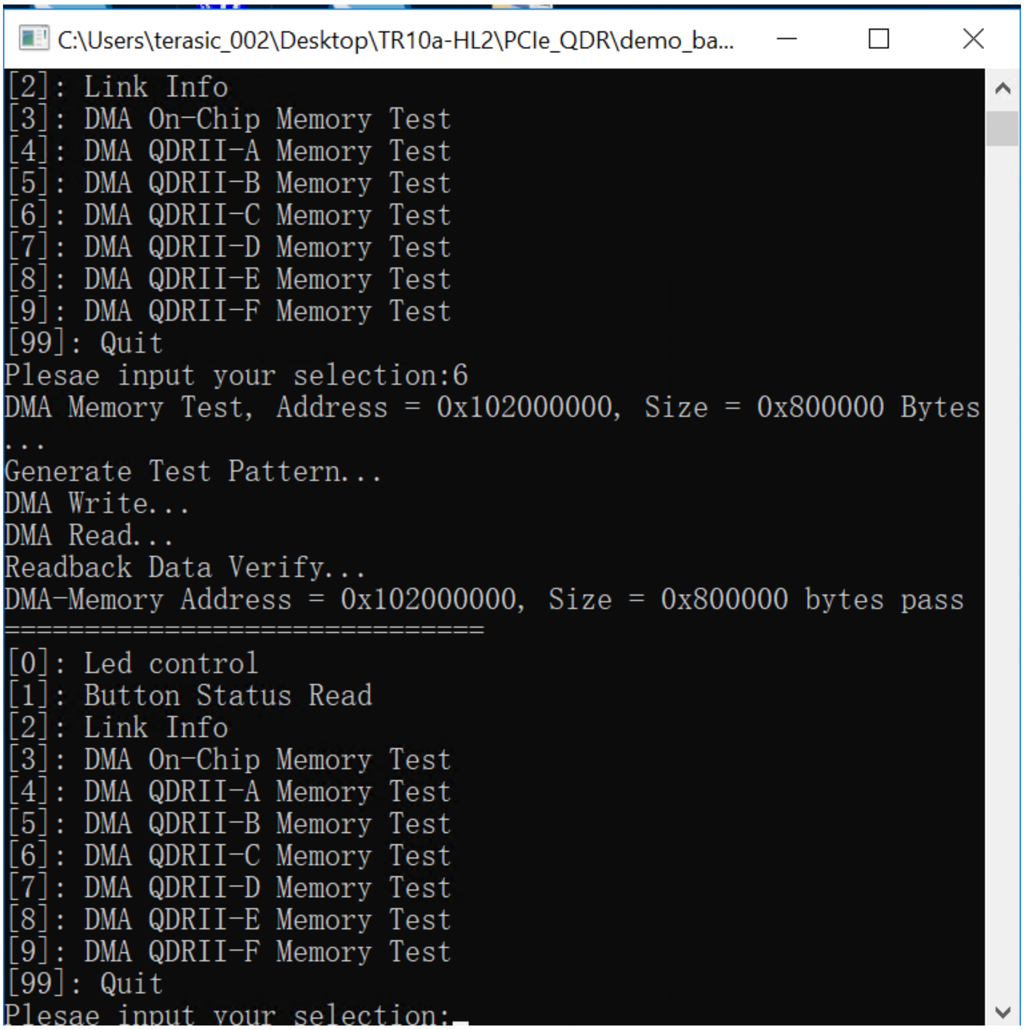



**Figure 7-21 Screenshot of QDRII-C Memory DAM Test Result**

12. Type 7 followed by an ENTERY key to select DMA QDRII-D Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-22**.

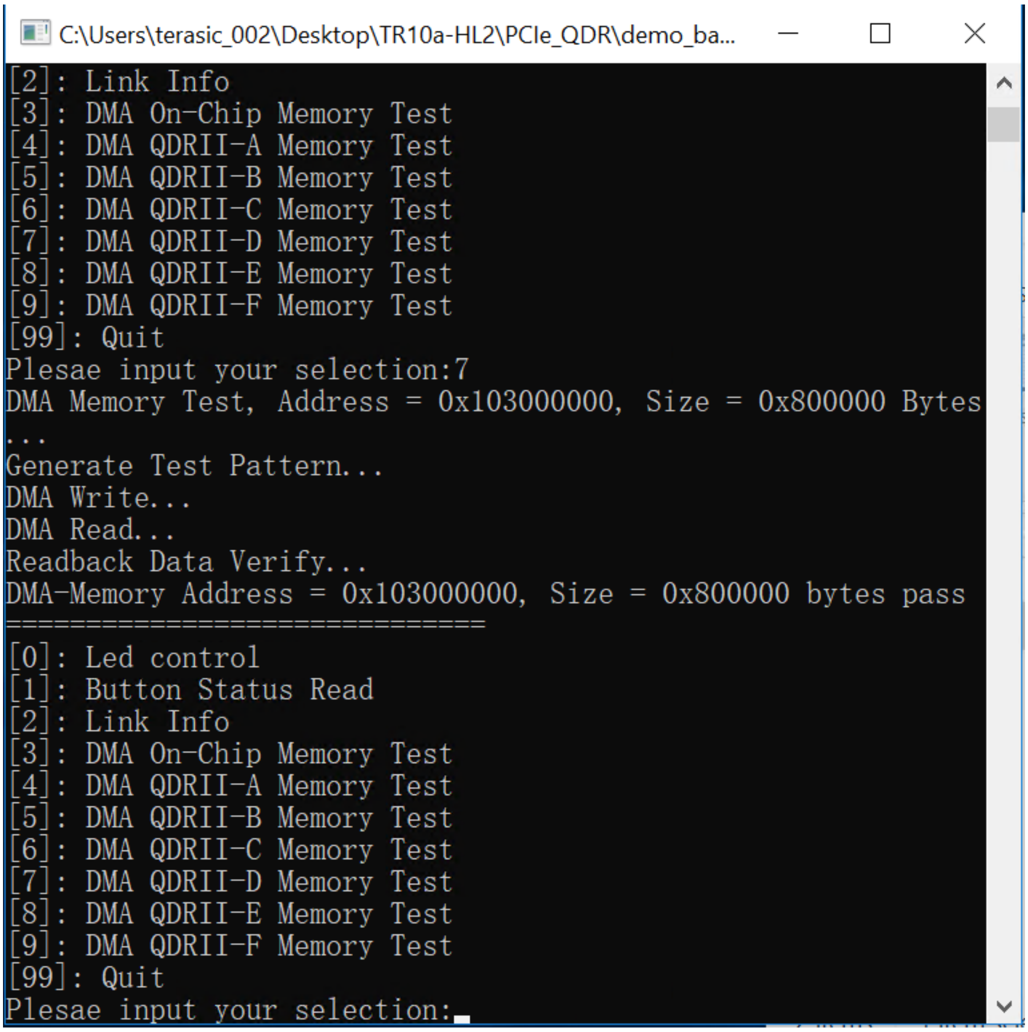



**Figure 7-22 Screenshot of QDRII-D Memory DAM Test Result**

13. Type 8 followed by an ENTERY key to select DMA QDRII-E Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-23**.

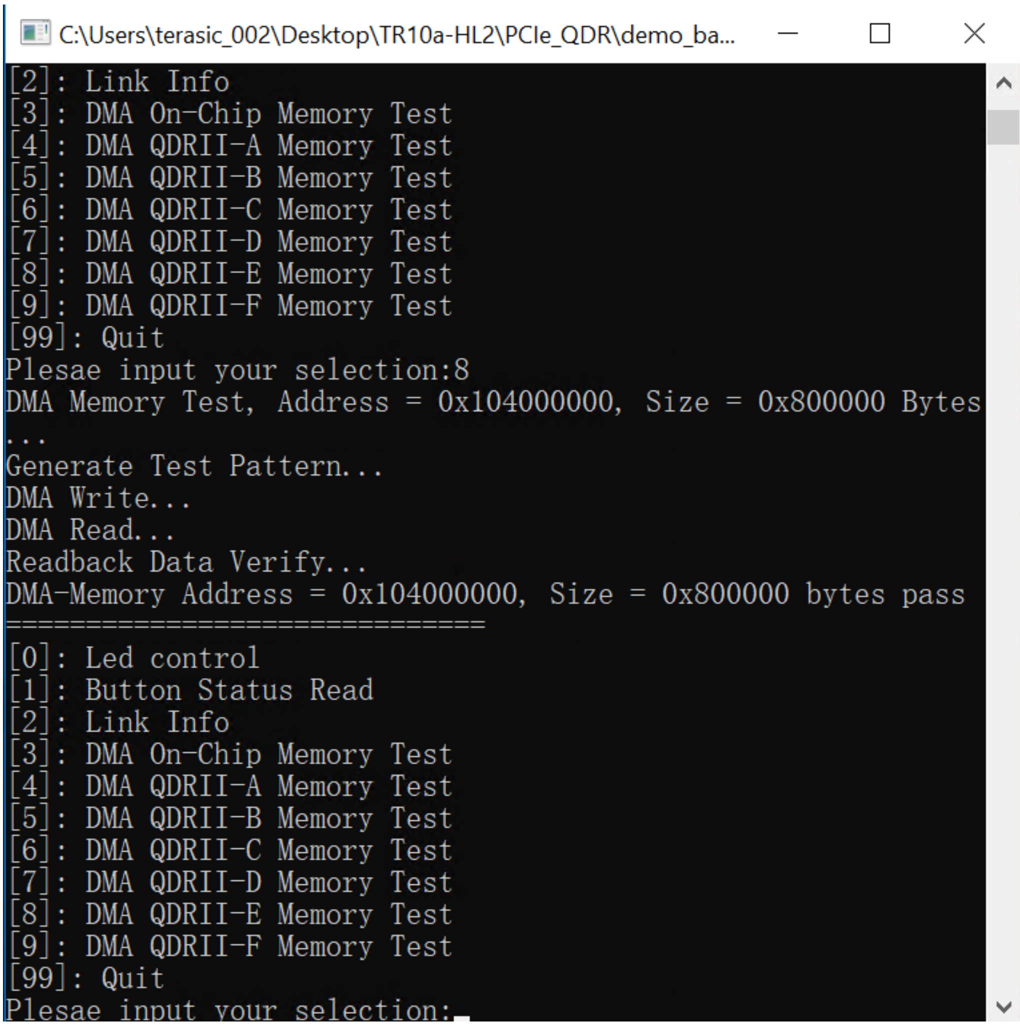



**Figure 7-23 Screenshot of QDRII-E Memory DAM Test Result**

14. Type 9 followed by an ENTERY key to select DMA QDRII-A Memory Test item. The DMA write and read test result will be report as shown in **Figure 7-24**.

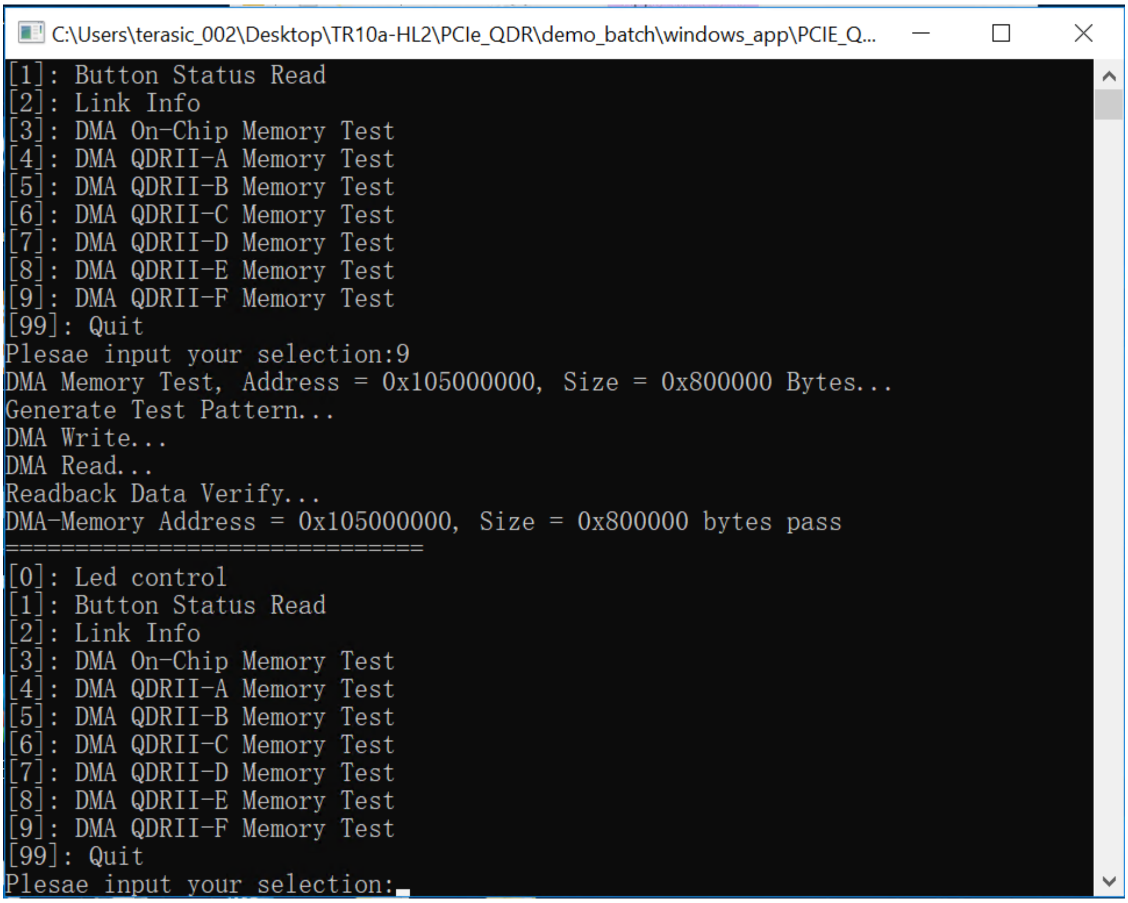



**Figure 7-24 Screenshot of QDRII-F Memory DAM Test Result**

15. Type 99 followed by an ENTERY key to exit this test program.

## Development Tools

- Quartus II 16.0
- Visual C++ 2012

## Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIE_QDR
- Visual C++ Project: Demonstrations\PCIe_SW_KIT\PCIE_QDR

## FPGA Application Design

**Figure 7-25** 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.

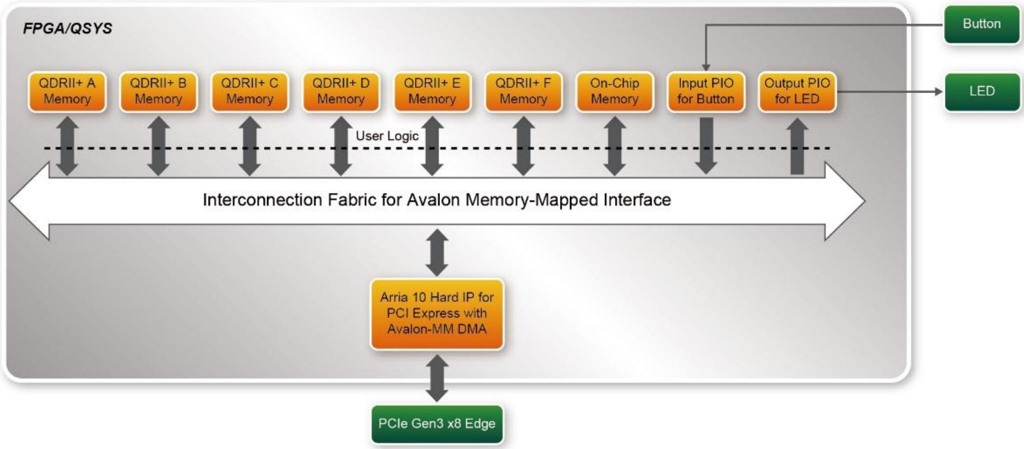



**Figure 7-25 Hardware block diagram of the PCIe QDRII+ reference design**

## 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_QDR.cpp | Main program |
| PCIE.c | Implement dynamically load for TERAISC_PCIE_AVMM.DLL |
| PCIE.h | |
| TERASIC_PCIE_AVMM.h | SDK library file, defines constant and data structure |

The main program PCIE_QDR.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_QDRIIA_MEM_ADDR   0x100000000
#define DEMO_PCIE_QDRIIB_MEM_ADDR   0x101000000
#define DEMO_PCIE_QDRIIC_MEM_ADDR   0x102000000
#define DEMO_PCIE_QDRIID_MEM_ADDR   0x103000000
#define DEMO_PCIE_QDRIIE_MEM_ADDR   0x104000000
#define DEMO_PCIE_QDRIIF_MEM_ADDR   0x105000000

#define ONCHIP_MEM_TEST_SIZE        (512*1024) //512KB
#define QDRIIA_MEM_TEST_SIZE        (8*1024*1024) //8MB
#define QDRIIB_MEM_TEST_SIZE        (8*1024*1024) //8MB
#define QDRIIC_MEM_TEST_SIZE        (8*1024*1024) //8MB
#define QDRIID_MEM_TEST_SIZE        (8*1024*1024) //8MB
#define QDRIIE_MEM_TEST_SIZE        (8*1024*1024) //8MB
#define QDRIIF_MEM_TEST_SIZE        (8*1024*1024) //8MB
```

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 Vender 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,(DWORD)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;
}
```

# 7.6 PCIe Design: PCIe_Fundamental_x2

The application reference design shows how to utilize the dual PCIe Gen3 x8 edge connector on this board. The two PCIe Gen3 x8 Link are directly connected to two Arria 10 PCIe Hard IP individually. There two IP are call **PCIe0** and **PCIe1** in this design. The Host PC can communicated with the two Hard IP independently, so the throughput between Host PC and FPGA is double than the single PCIe Gen3 x8 link. In the design, basic I/O is used to control the BUTTON and LED on the FPGA board. **PCIe0** controls the **User LED** and **PCIe1** controls the **Bracket LED**. High-speed data transfer is performed by DMA.

Note, this demonstration requires the Host PC to support **PCIe Bifurcation**.

## ■ Demonstration Files Location

The demo file is located in the batch folder:

CDROM\demonstrations\PCIe_funcdamental_x2\demo_batch

The folder includes following files:

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

## ■ Demonstration Setup

1. Make sure your Host PC supports PCIe bifurcation and is enabled in BIOS.
2. Install the FPGA board on the bifurcation PCIe slot of your PC.
3. Configure FPGA with PCIE_Fundamental.sof by executing the test.bat.
4. Install PCIe driver if necessary. The driver is located in the folder:

   CDROM\Demonstration\PCIe_SW_KIT\PCIe_Driver
5. Restart Windows
6. Make sure there are two Altera PCI API drivers are enumerated by checking the Windows Control panel as shown in **Figure 7-26**.
7. Goto windows_app folder, execute PCIE_FUNDMENTAL.exe. A menu will appear as shown in **Figure 7-27**.

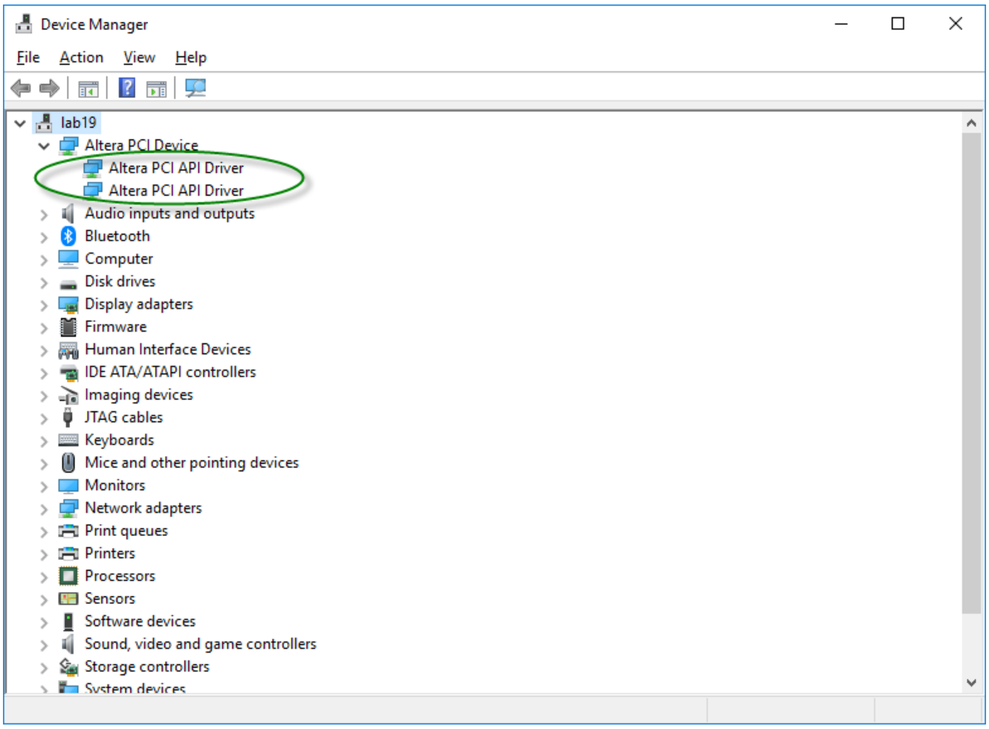



**Figure 7-26 Screenshot of two FPGA PCIe devices are detected**

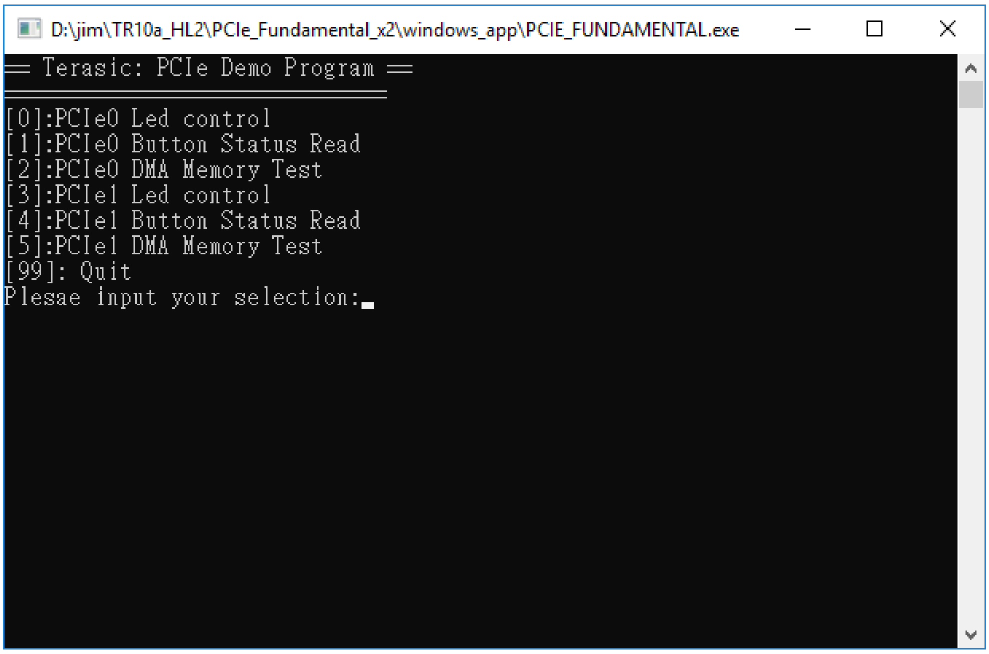



**Figure 7-27 Screenshot of Program Menu**

8. Type 0 followed by a ENTERY key to select **PCIe0** Led Control item, then input 15 (hex 0x0f) will make all **User LED** lighten as shown in **Figure 7-28**. If input 0(hex

0x00), all **User LED** will be turn off.

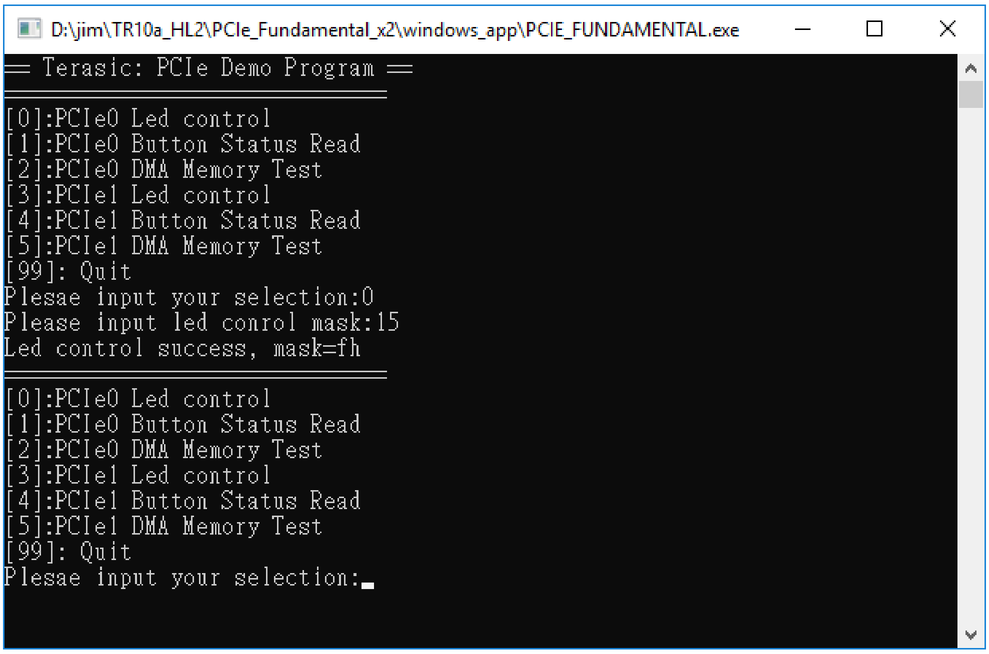



**Figure 7-28 Screenshot of PCIe0 LED Control**

9. Type 1 followed by an ENTERY key to select **PCIe0** Button Status Read item. The button status will be report as shown in **Figure 7-29** .

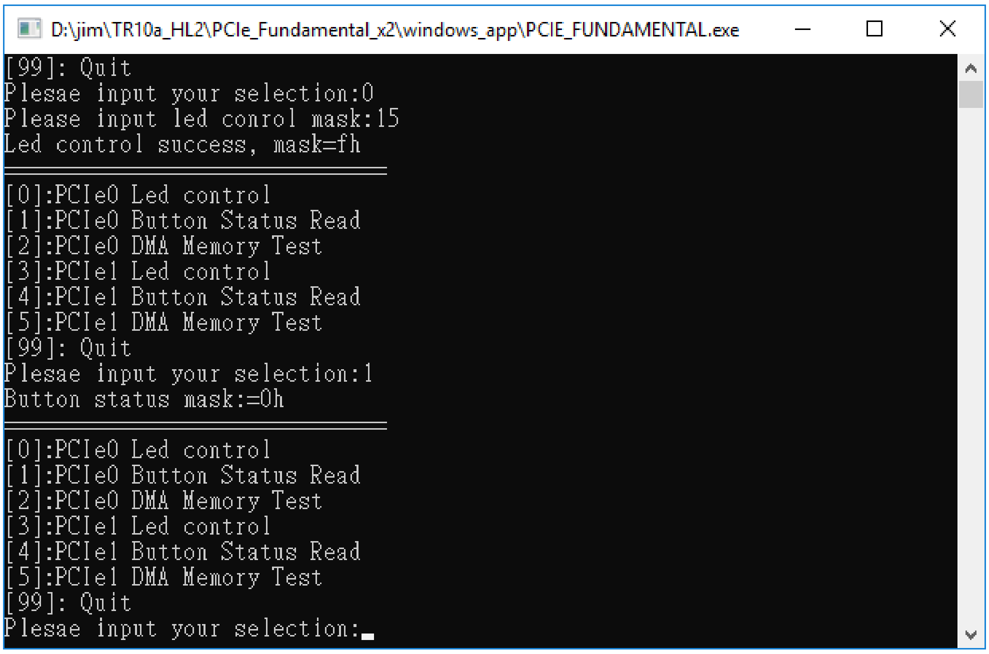



**Figure 7-29 Screenshot of PCIe0 Button Status Report**

10. Type 2 followed by an ENTERY key to select **PCIe0** DMA Testing item. The DMA test result will be report as shown in **Figure 7-30**.

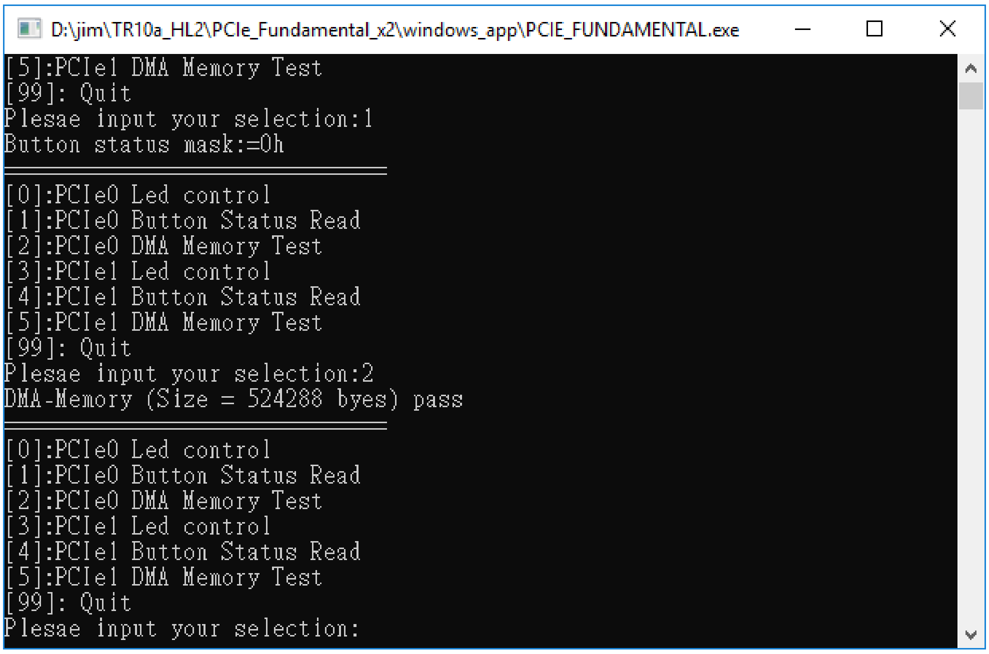



**Figure 7-30 Screenshot of PCIe0 DMA Memory Test Result**

11. Type 3 followed by a ENTERY key to select **PCIe1** Led Control item, then input 15 (hex 0x0f) will make all **Bracket LED** on as shown in **Figure 7-31**. If input 0(hex 0x00), all **Bracket LED** will be turn off.

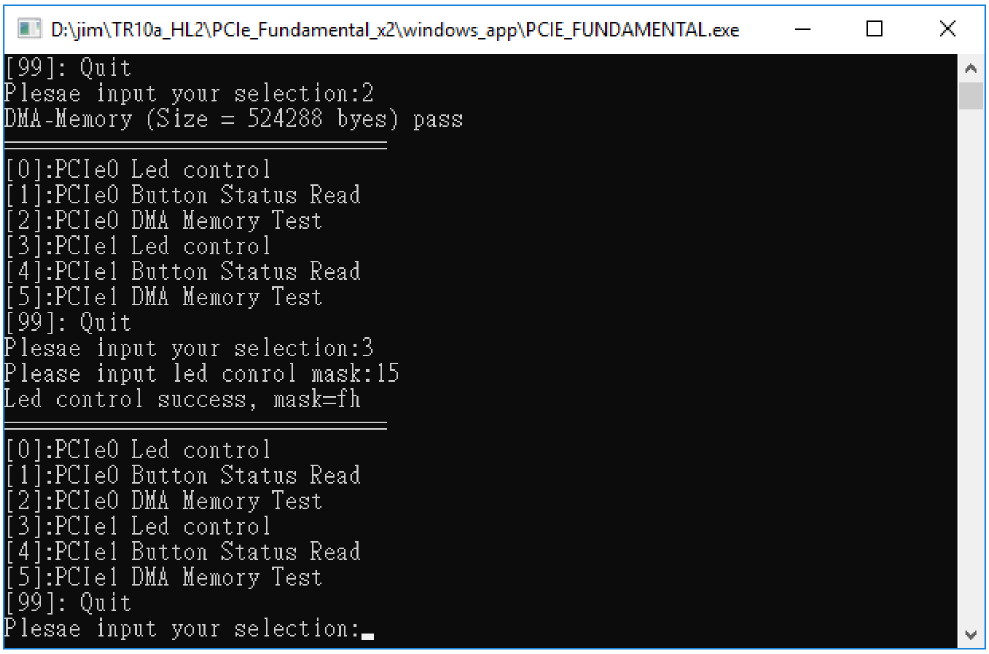



**Figure 7-31 Screenshot of PCIe1 LED Control**

12. Type 4 followed by an ENTERY key to select **PCIe1** Button Status Read item. The button status will be report as shown in **Figure 7-32**.

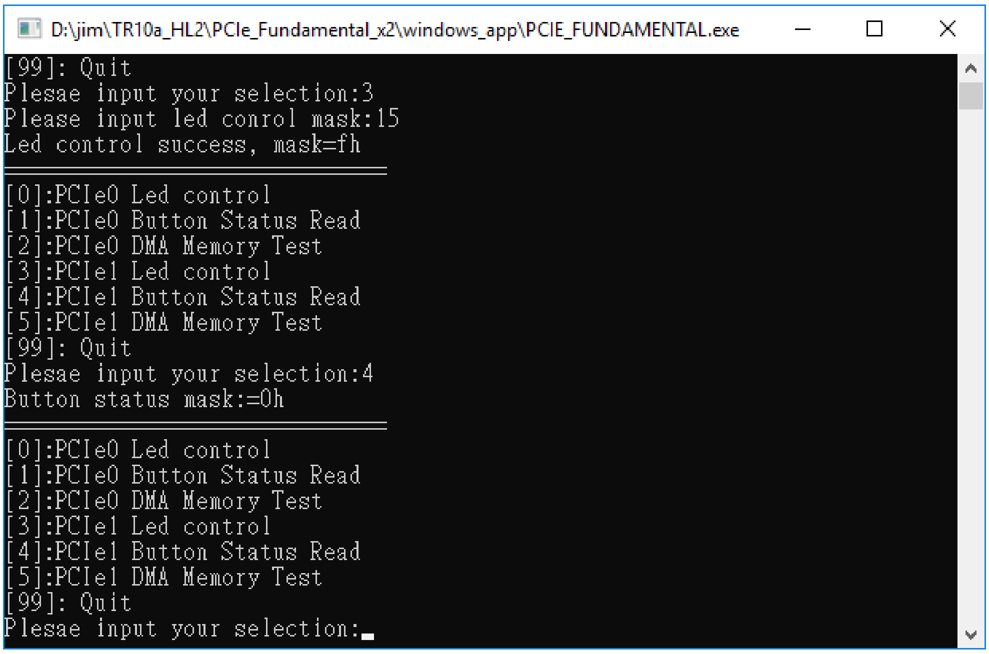



**Figure 7-32 Screenshot of PCIe1 Button Status Report**

13. Type 2 followed by an ENTERY key to select **PCIe1** DMA Testing item. The DMA test result will be report as shown in **Figure 7-33**.

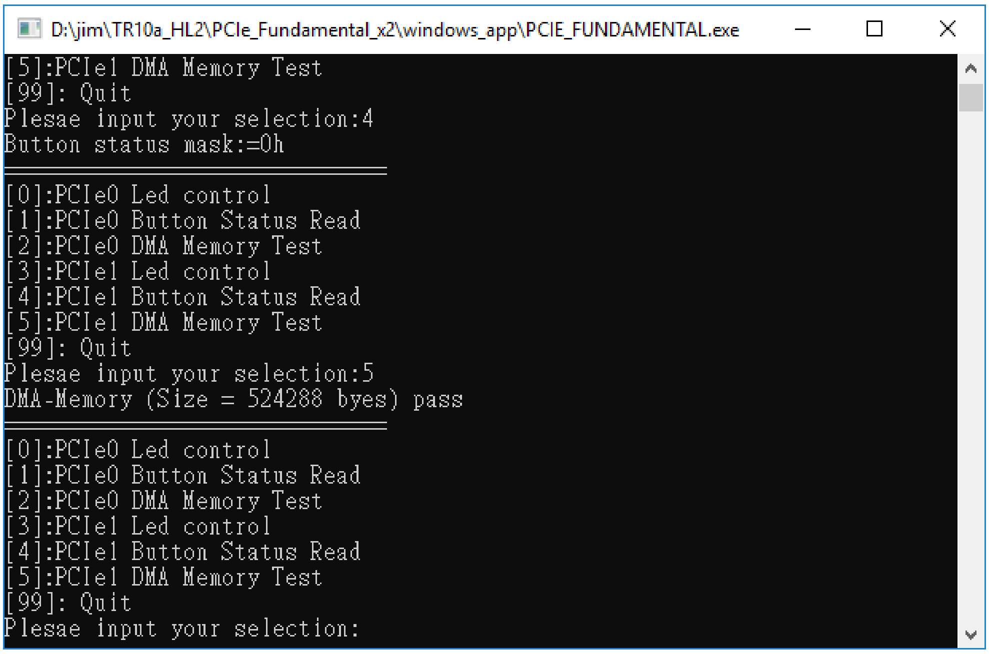



**Figure 7-33 Screenshot of PCIe0 DMA Memory Test Result**

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

## Development Tools

- Quartus II 16.1.2

- Visual C++ 2012

## Demonstration Source Code Location

- Quartus Project: Demonstrations\PCIe_Fundamental_x2
- Visual C++ Project: Demonstrations\PCIe_SW_KIT\PCIE_FUNDAMENTAL_x2

## FPGA Application Design

**Figure 7-34** shows the system block diagram in the FPGA system. In the Qsys, Altera PIO controller is used to control the LED (User LED and Bracket 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.

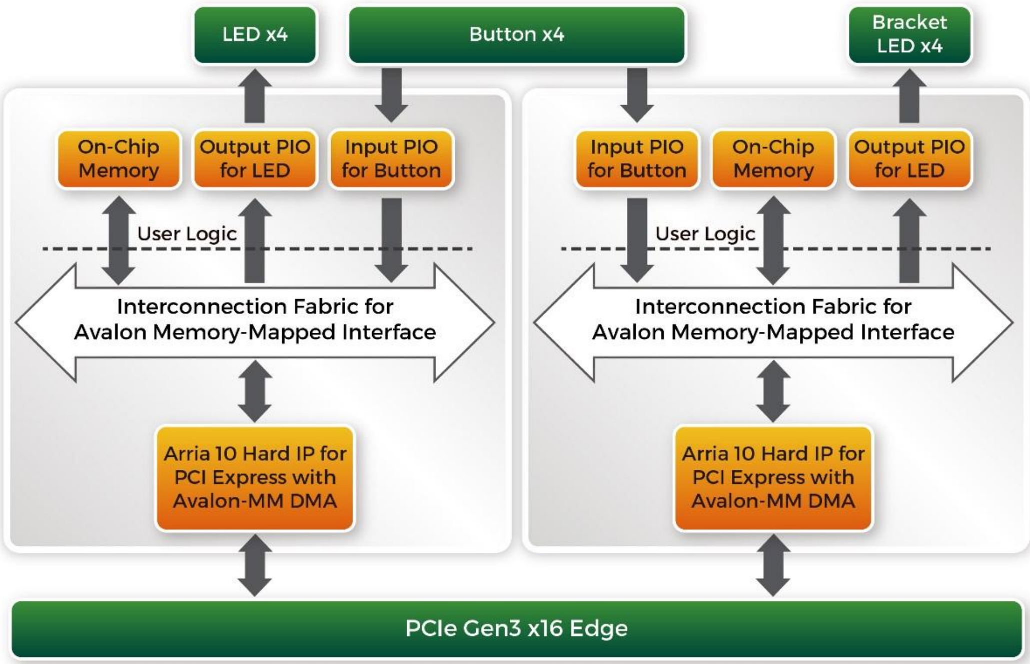



**Figure 7-34 Hardware block diagram of the Dual PCIe reference design**

## 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 TERAISC_PCIE_AVMM.DLL |
| 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_AVMM.DLL. Then, it call PCIE_Open twice to open two PCI Express devices:

hPCIE_0 = PCIE_Open(DEFAULT_PCIE_VID, DEFAULT_PCIE_DID, 0);

hPCIE_1 = PCIE_Open(DEFAULT_PCIE_VID, DEFAULT_PCIE_DID, 1);

The constant DEFAULT_PCIE_VID and DEFAULT_PCIE_DID used in PCIE_Open are defined in TERASIC_PCIE_AVMM.h. If developers change the Vender ID and Device ID and PCI Express IP, they also need to change the ID value define in TERASIC_PCIE_AVMM.h. The third parameter (0 and 1 in this case) is used to specify the PCIe device. 0 means the first device found in the PCIe bus with the given Vender ID and Device ID. 1 means the second device found in the PCIe bus with the given Vender ID and Device ID. 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,(DWORD)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);
```

Chapter 8

# Transceiver Verification

This chapter describes how to verify the FPGA transceivers for the QSFP+ by using the test code provided in the TR10a-HL2 system CD.

## 8.1 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.

## 8.2 Loopback Fixture

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

- QSFP+ Cable, as shown in **Figure 8-1**
- QSFP+ Loopback fixture, as shown in **Figure 8-2**

**Figure 8-1 Optical QSFP+ Cable**

**Figure 8-2 QSFP+ Loopback Fixture**

**Figure 8-3** shows the FPGA board with two QSFP+ cable installed. **Figure 8-4** shows the FPGA board with four QSFP+ loopback fixtures installed.

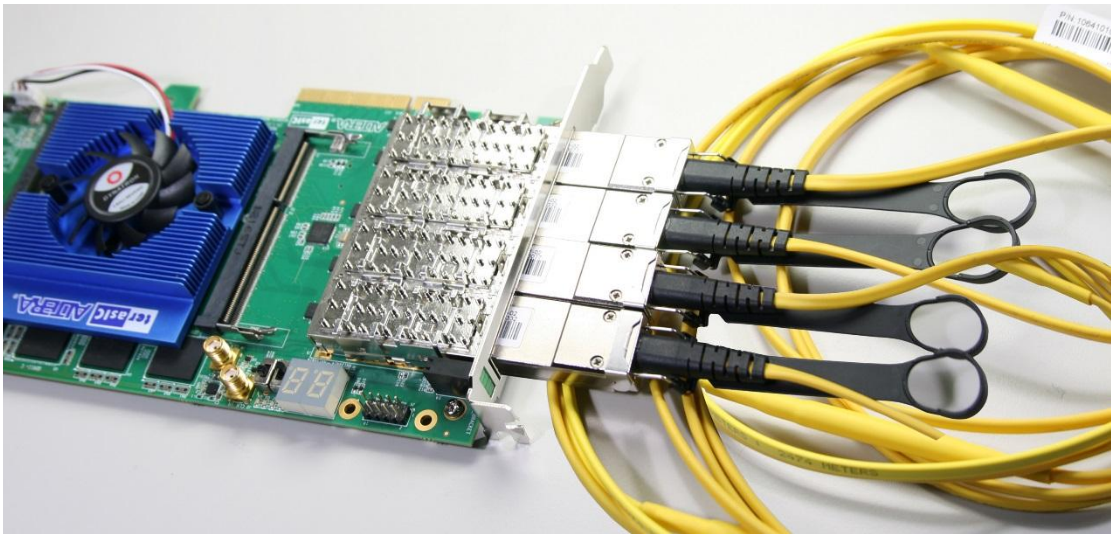


**Figure 8-3 Two QSFP+ Cables Installed**

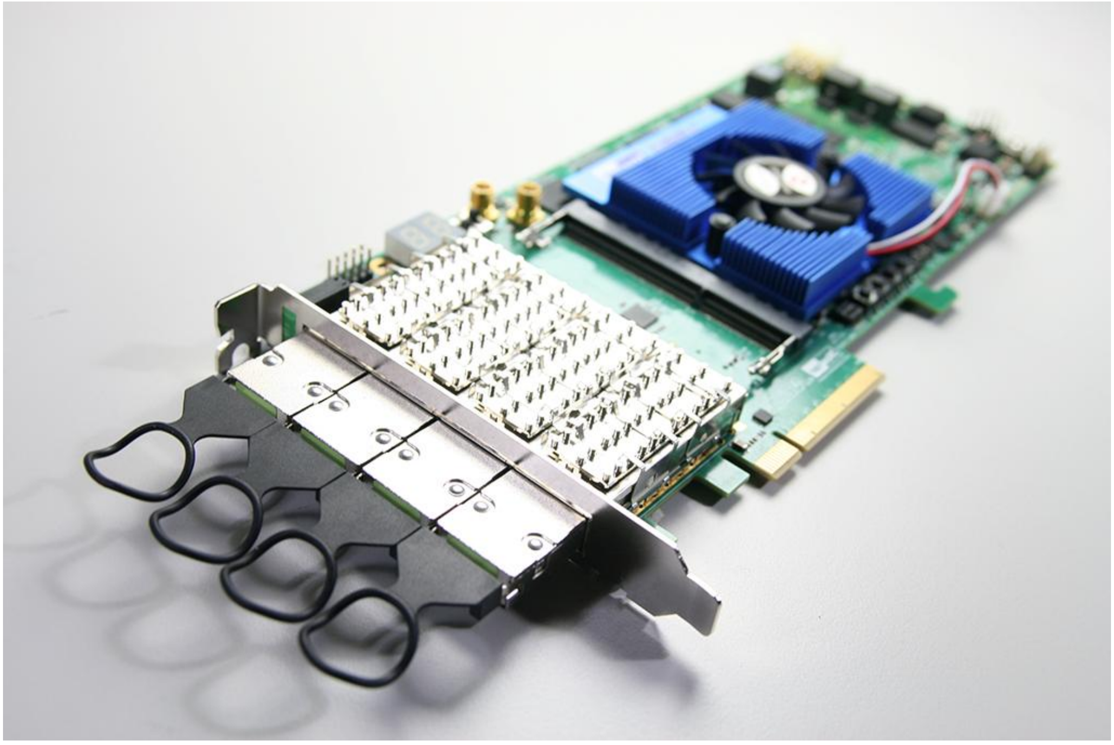


**Figure 8-4 Four QSFP+ Loopback Fixtures Installed**

## 8.3 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 8-5**.
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 8-6**.

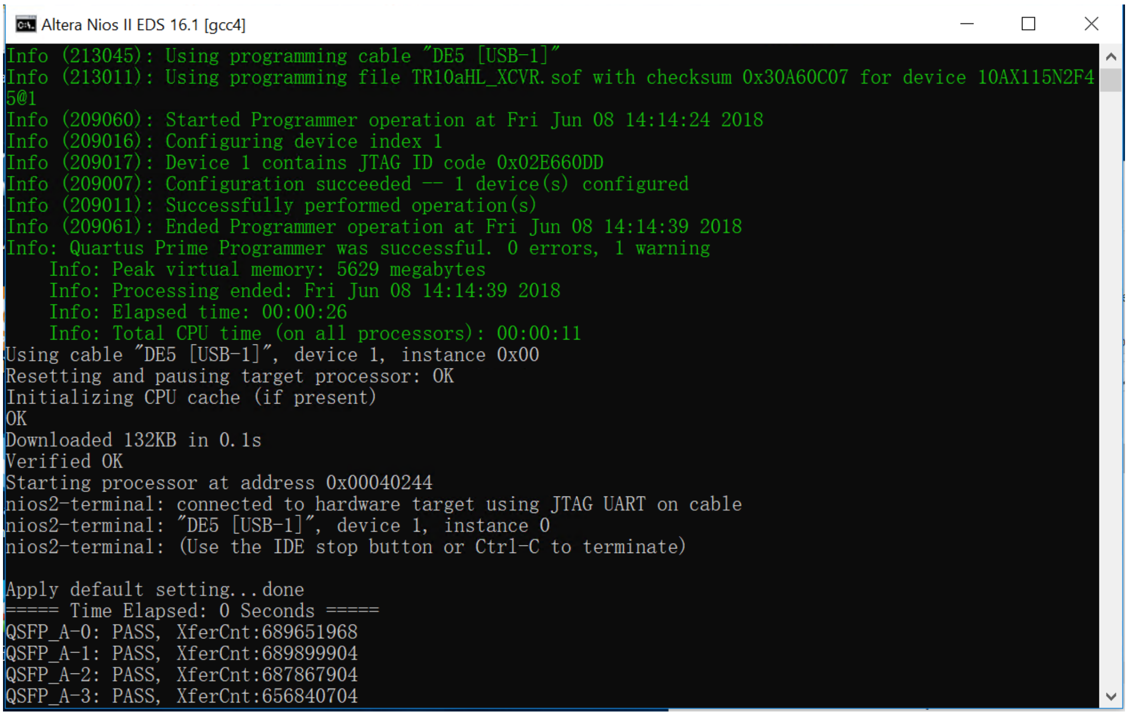



**Figure 8-5 QSFP+ Transceiver Loopback Test in Progress**

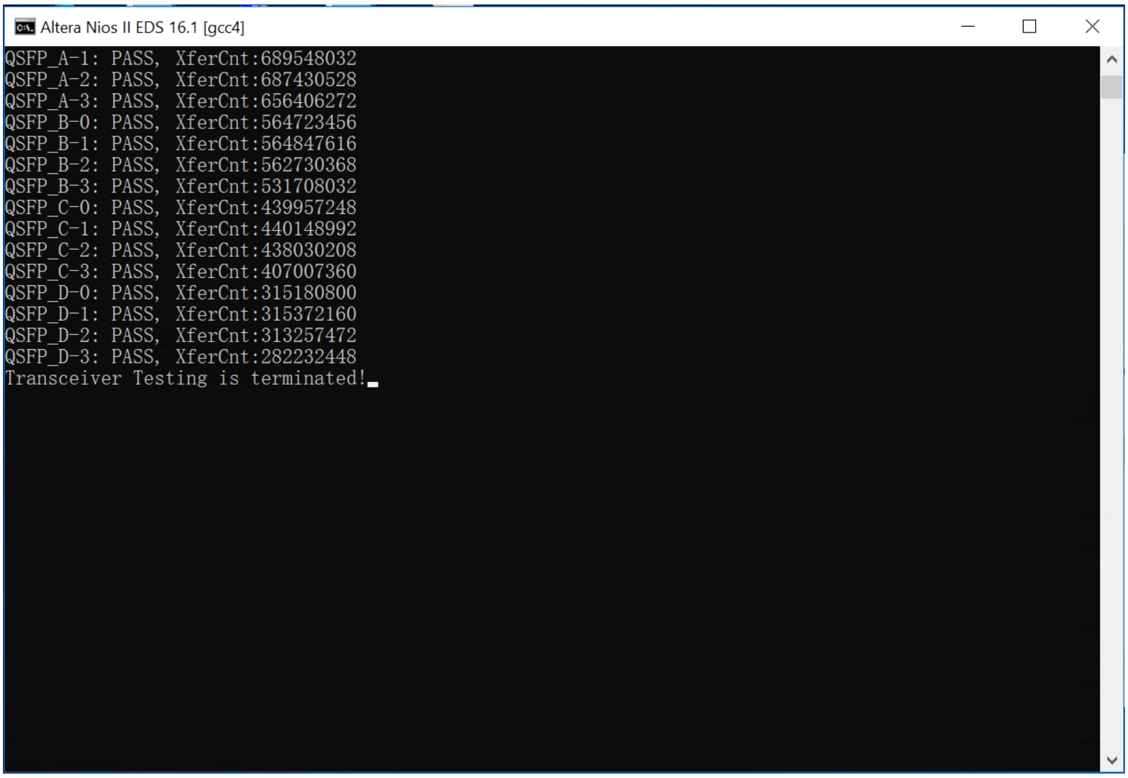



**Figure 8-6 QSFP Transceiver Loopback Done**

# 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
TE10a-HL Web: tr10a-hl2.terasic.com

### ■ Revision History

| Date | Version | Changes |
|---|---|---|
| 2018.05 | First publication | |
| 2019.04 | V1.0 | Modify ClockBuilder Pro Software download link |
| 2019.07 | V1.1 | Modify some section title |

# EPlast1.ru

Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.


### Как с нами связаться

**Телефон:** 8 (812) 309 58 32 (многоканальный)
**Факс:** 8 (812) 320-02-42
**Электронная почта:** org@eplast1.ru
**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.