DE5-Net

FPGA Development Kit

User Manual

Stratix V | Flash | PCI EXPRESS | DDR3 | QDRII+ | SFP+ | SERIAL ATA

www.terasic.com

CONTENTS

Chapter 1

Overview

This chapter provides an overview of the DE5-Net Development Board and installation guide.

1.1 General Description

The Terasic DE5-Net Stratix V GX FPGA Development Kit provides the ideal hardware solution for designs that demand high capacity and bandwidth memory interfacing, ultra-low latency communication, and power efficiency. With a full-height, 3/4-length form-factor package, the DE5-Net is designed for the most demanding high-end applications, empowered with the top-of-the-line Altera Stratix V GX, delivering the best system-level integration and flexibility in the industry.

The Stratix® V GX FPGA features integrated transceivers that transfer at a maximum of 12.5 Gbps, allowing the DE5-Net to be fully compliant with version 3.0 of the PCI Express standard, as well as allowing an ultra low-latency, straight connections to four external 10G SFP+ 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 DE5-Net delivers with two independent banks of DDR3 SO-DIMM RAM, four independent banks of QDRII+ SRAM, high-speed parallel flash memory, and four SATA ports. The feature-set of the DE5-Net fully supports all high-intensity applications such as low-latency trading, cloud computing, high-performance computing, data acquisition, network processing, and signal processing.

1.2 Key Features

The following hardware is implemented on the DE5-Net board:

- **FPGA**
  - Altera Stratix® V GX FPGA (5SGXEA7N2F45C2)

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

- **General user input/output:**
  - 10 LEDs
  - 4 push-buttons
  - 4 slide switches
  - 2 seven-segment displays

- **Clock System**
  - 50MHz Oscillator
  - Programmable oscillators Si570, CDCM61001 and CDCM61004
  - One SMA connector for external clock input
  - One SMA connector for clock output

- **Memory**
  - DDR3 SO-DIMM SDRAM
  - QDRII+ SRAM
  - FLASH

- **Communication Ports**
  - Four SFP+ connectors
  - Two Serial ATA host ports
  - Two Serial ATA device ports
  - PCI Express (PCIe) x8 edge connector
  - One RS422 transceiver with RJ45 connector

■ **System Monitor and Control**

- Temperature sensor
- Fan control

■ **Power**

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

■ **Mechanical Specification**

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

1.3 Block Diagram

Figure 1-1 shows the block diagram of the DE5-Net board. To provide maximum flexibility for the users, all key components are connected with the Stratix V GX FPGA device. Thus, users can configure the FPGA to implement any system design.

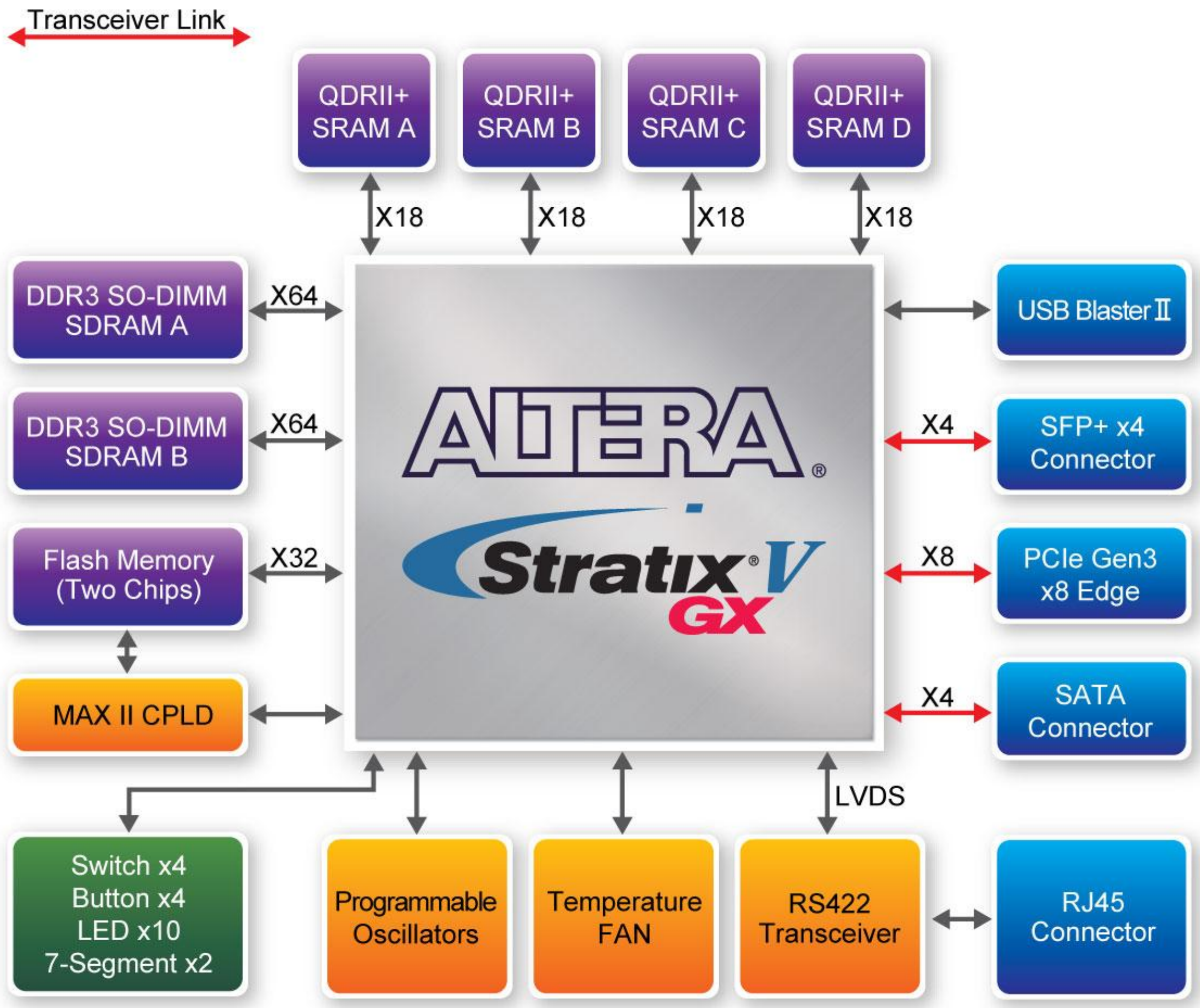


Figure 1-1 Block diagram of the DE5-Net board

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

Stratix V GX FPGA

- 5SGXEA7N2F45C2
- 622,000 logic elements (LEs)
- 50-Mbits embedded memory
- 48 transceivers (12.5Gbps)
- 512 18-bit x 18-bit multipliers
- 256 27-bit x 27-bit DSP blocks

- 2 PCI Express hard IP blocks
- 840 user I/Os
- 210 full-duplex LVDS channels
- 28 phase locked loops (PLLs)

JTAG Header and FPGA Configuration

- On-board USB Blaster II or JTAG header for use with the Quartus II Programmer
- MAXII CPLD EPM2210 System Controller and Fast Passive Parallel (FPP) configuration

Memory devices

- 32MB QDRII+ SRAM
- Up to 8GB DDR3 SO-DIMM SDRAM
- 256MB FLASH

General user I/O

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

On-Board Clock

- 50MHz oscillator
- Programming PLL providing clock for 10G SFP+ transceiver
- Programming PLL providing clock for SATA or 1G SFP+ transceiver

Four Serial ATA ports

- SATA 3.0 standard at 6Gbps signaling rate

Four SFP+ ports

- Four SFP+ connector (10 Gbps+)

PCI Express x8 edge connector

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

Power Source

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

Chapter 2

Board Components

This chapter introduces all the important components on the DE5-Net.

2.1 Board Overview

Figure 2-1 is the top and bottom view of the DE5-Net development board. It depicts the layout of the board and indicates the location of the connectors and key components. Users can refer to this figure for relative location of the connectors and key components.

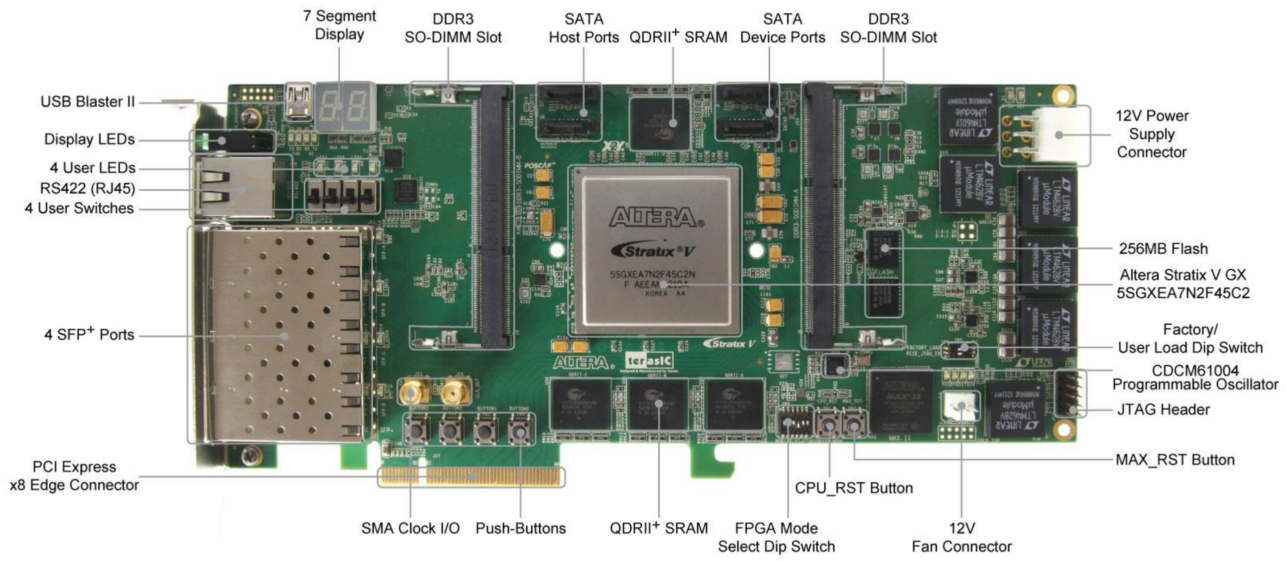


Figure 2-1 FPGA Board (Top)

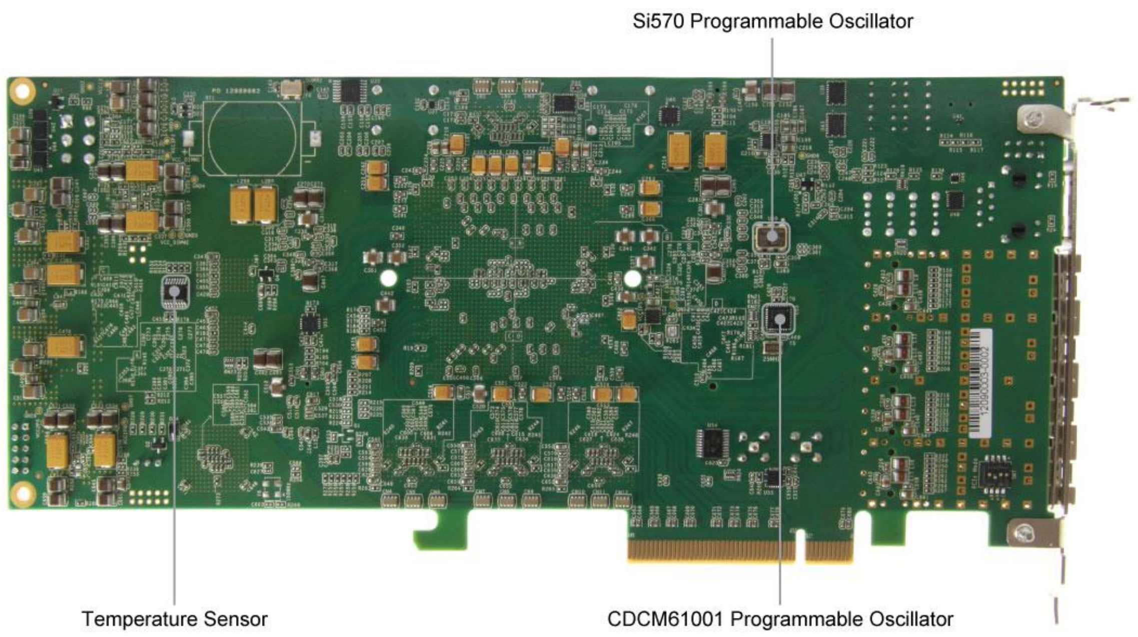

Si570 Programmable Oscillator
Temperature Sensor
CDCM61001 Programmable Oscillator

Figure 2-2 FPGA Board (Bottom)

2.2 Configuration, Status and Setup

■ Configure

The FPGA board supports two configuration methods for the Stratix V 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 Stratix V GX FPGA:

- Make sure that power is provided to the FPGA board
- Connect your PC to the FPGA board using a mini-USB cable and make sure the USB-Blaster II driver is installed on PC.
- Launch Quartus 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.**
D1	**3.3-V Power**	**Illuminates when 3.3-V power is active.**
D15	**CONF DONE**	**Illuminates when the FPGA is successfully configured. Driven by the MAX II CPLD EPM2210 System Controller.**
D16	**Loading**	**Illuminates when the MAX II CPLD EPM2210 System Controller is actively configuring the FPGA. Driven by the MAX II CPLD EPM2210 System Controller with the Embedded Blaster CPLD.**
D17	**Error**	**Illuminates when the MAX II CPLD EPM2210 System Controller fails to configure the FPGA. Driven by the MAX II CPLD EPM2210 System Controller.**
D18	**PAGE**	**Illuminates when FPGA is configured by the factory configuration bit stream.**

■ Setup PCI Express Control DIP switch

The PCI Express Control DIP switch (SW7) is provided to enable or disable different configurations of the PCIe Connector. Table 2-2 lists the switch controls and description.

Table 2-2 SW3 PCIe Control DIP Switch

Board Reference	*Signal Name*	*Description*	*Default*
SW7.1	**PCIE_PRSNT2n_x1**	**On : Enable x1 presence detect Off: Disable x1 presence detect**	**Off**
SW7.2	**PCIE_PRSNT2n_x4**	**On : Enable x4 presence detect Off: Disable x4 presence detect**	**Off**
SW7.3	**PCIE_PRSNT2n_x8**	**On : Enable x8 presence detect Off: Disable x8 presence detect**	**On**

Setup Configure Mode Control DIP switch

The Configure Mode Control DIP switch (SW6) is provided 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**.

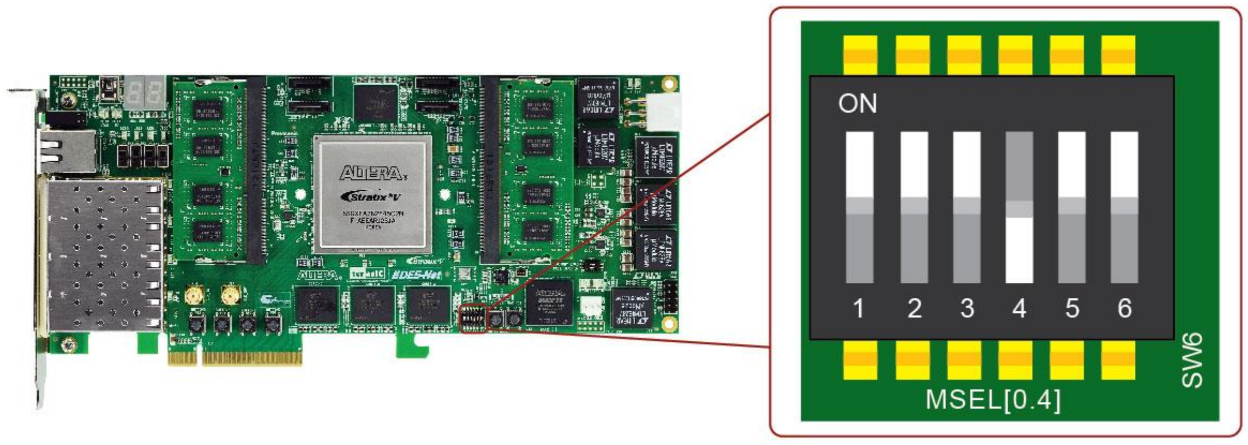


Figure 2-3 6-Position DIP switch for Configure Mode

Select Flash Image for Configuration

The Image Select DIP switch (SW5) is provided to specify the image for configuration of the FPGA. Setting Position 2 of SW5 to high (right) specifies the default factory image to be loaded, as shown in **Figure 2-4**. Setting Position 2 of SW5 to low (left) specifies the DE5-Net to load a user-defined image, as shown in **Figure 2-5**.

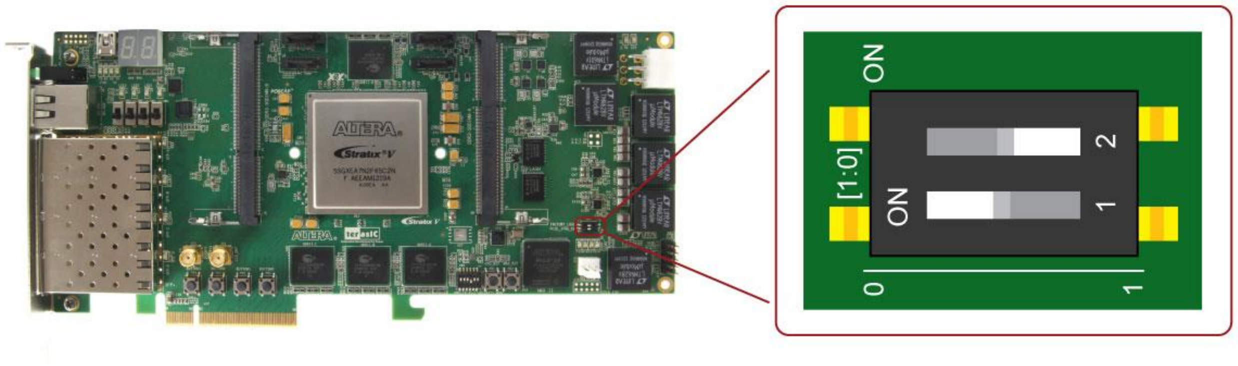


Figure 2-4 2-position DIP switch for Image Select – Factory Image Load

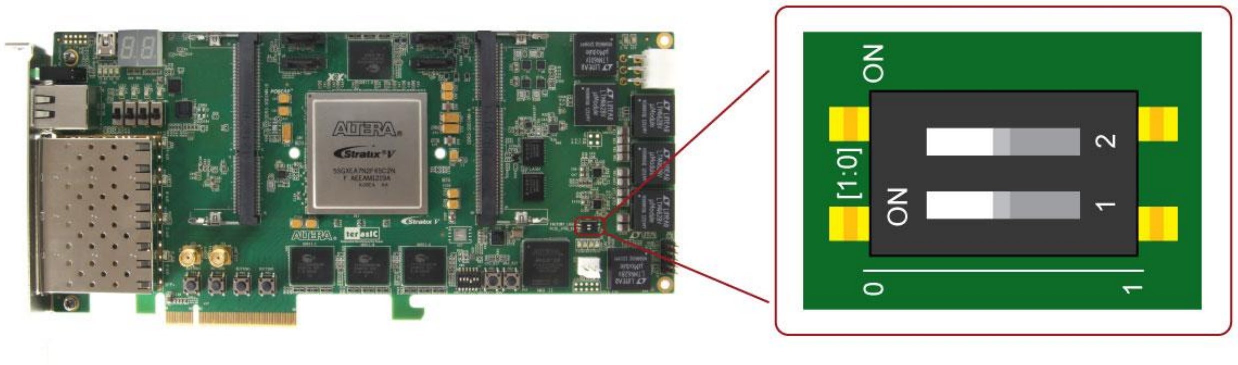


Figure 2-5 2-position DIP switch for Image Select – User Image Load

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 Stratix V 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 Stratix V 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***	***Stratix V GX Pin Number***
PB6	**BUTTON0**	**High Logic Level when the button is**	**2.5-V**	**PIN_AK15**

PB5	BUTTON1	not pressed	2.5-V	PIN_AK14
PB4	BUTTON2		2.5-V	PIN_AL14
PB3	BUTTON3		2.5-V	PIN_AL15

User-Defined Slide Switch

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

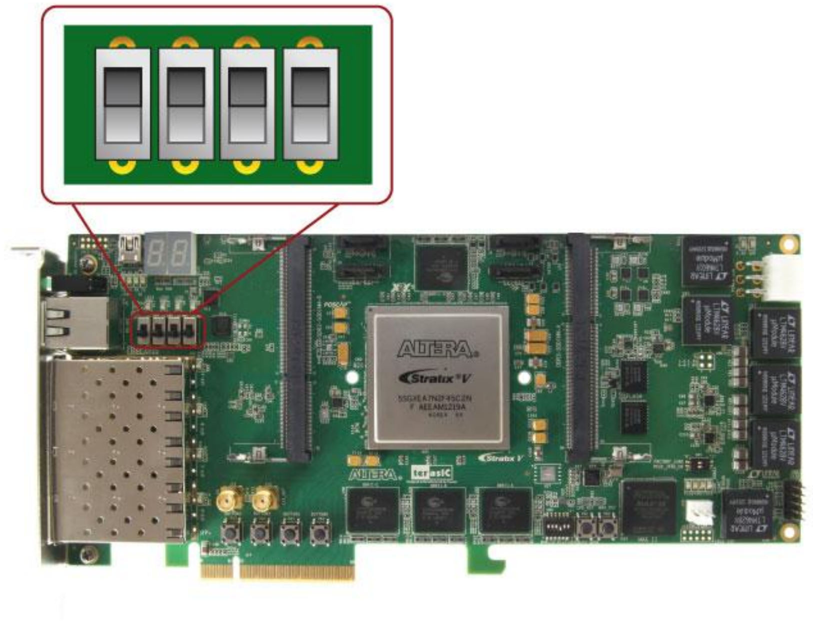


Figure 2-6 4 Slide switches

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

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

Board Reference	*Schematic Signal Name*	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
SW0	SW0	High logic level when SW in the UPPER position.	1.8-V	PIN_B25
SW1	SW1		1.8-V	PIN_A25
SW2	SW2		1.8-V	PIN_B23
SW3	SW3		1.8-V	PIN_A23

■ User-Defined LEDs

The FPGA board consists of 10 user-controllable LEDs to allow status and debugging signals to be driven to the LEDs from the designs loaded into the Stratix V GX device. Each LED is driven directly by the Stratix V 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*	*Stratix V GX Pin Number*
D8	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.	2.5-V	PIN_AW37
D9	LED1		2.5-V	PIN_AV37
D10	LED2		2.5-V	PIN_BB36
D11	LED3		2.5-V	PIN_BB39
D7-1	LED_BRACKET0		2.5-V	PIN_AH15
D7-3	LED_BRACKET1		2.5-V	PIN_AH13
D7-5	LED_BRACKET2		2.5-V	PIN_AJ13
D7-7	LED_BRACKET3		2.5-V	PIN_AJ14
J8-10	LED_RJ45_L		2.5-V	PIN_AG15
J8-12	LED_RJ45_R		2.5-V	PIN_AG16

■ 7-Segment Displays

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

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

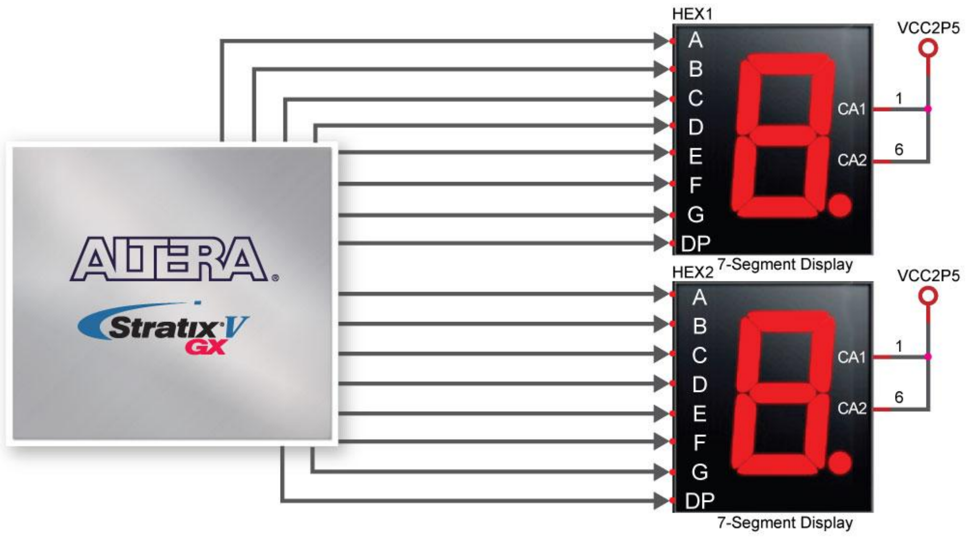


Figure 2-7 Connection between 7-segment displays and Stratix V GX FPGA

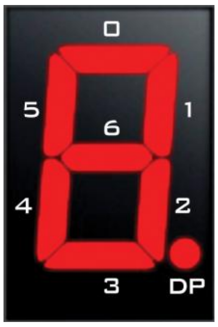


Figure 2-8 Position and index of each segment in a 7-segment display

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

Board Reference	*Schematic Signal Name*	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
HEX1	HEX1_D0	User-Defined 7-Segment Display. Driving logic 0 on the I/O port turns the 7-segment signal ON. Driving logic 1 on the I/O port turns the 7-segment signal OFF.	1.5-V	PIN_H18
HEX1	HEX1_D1		1.5-V	PIN_G16
HEX1	HEX1_D2		1.5-V	PIN_F16
HEX1	HEX1_D3		1.5-V	PIN_A7
HEX1	HEX1_D4		1.5-V	PIN_B7
HEX1	HEX1_D5		1.5-V	PIN_C9
HEX1	HEX1_D6		1.5-V	PIN_D10

HEX1	HEX1_DP		1.5-V	PIN_E9
HEX0	HEX0_D0		1.5-V	PIN_G8
HEX0	HEX0_D1		1.5-V	PIN_H8
HEX0	HEX0_D2		1.5-V	PIN_J9
HEX0	HEX0_D3		1.5-V	PIN_K10
HEX0	HEX0_D4		1.5-V	PIN_K8
HEX0	HEX0_D5		1.5-V	PIN_K9
HEX0	HEX0_D6		1.5-V	PIN_N8
HEX0	HEX0_DP		1.5-V	PIN_P8

2.4 Temperature Sensor and Fan Control

The FPGA board is equipped with a temperature sensor, MAX1619, which provides temperature sensing and over-temperature alert. These functions are accomplished by connecting the temperature sensor to the internal temperature sensing diode of the Stratix V GX device. The temperature status and alarm threshold registers of the temperature sensor can be programmed by a two-wire SMBus, which is connected to the Stratix V GX FPGA. In addition, the 7-bit POR slave address for this sensor is set to ‘0011000b’.

An optional 3-pin +12V fan located on J15 of the FPGA board is intended to reduce the temperature of the FPGA. Users can control the fan to turn on/off depending on the measured system temperature. The FAN is turned on when the FAN_CTRL pin is driven to a high logic level.

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

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
TEMPDIODEp	Positive pin of temperature diode in Stratix V	1.8-V	PIN_P6
TEMPDIODEn	Negative pin of temperature diode in Stratix V	1.8-V	PIN_P7
TEMP_CLK	SMBus clock	2.5-V	PIN_D21
TEMP_DATAT	SMBus data	2.5-V	PIN_D20
TEMP_OVERT_n	SMBus alert (interrupt)	2.5-V	PIN_C22
TEMP_INT_n	SMBus alert (interrupt)	2.5-V	PIN_C21
FAN_CTRL	Fan control	2.5-V	PIN_AR32

2.5 Clock Circuit

The development board includes one 50 MHz and three programmable oscillators. Figure 2-9 shows the default frequencies of on-board all external clocks going to the Stratix V GX FPGA. The figures also show an off-board external clock from PCI Express Host to the FPGA.

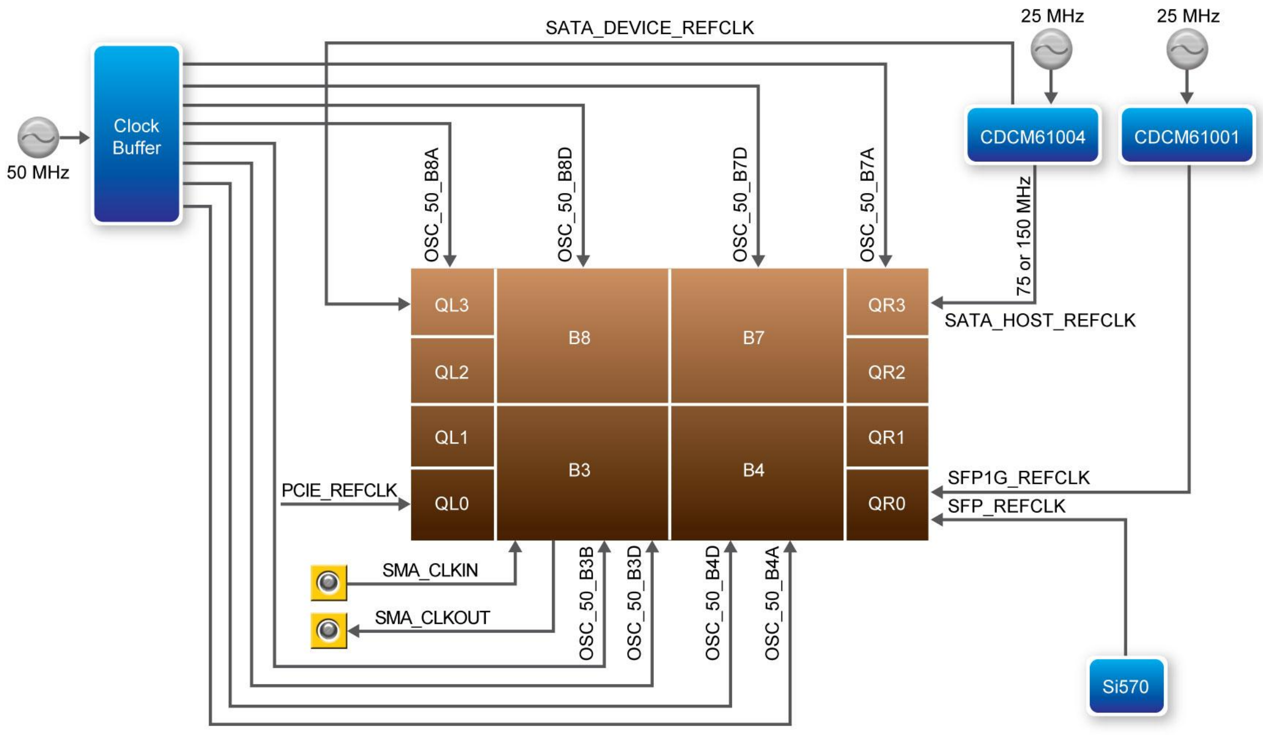


Figure 2-9 Clock circuit of the FPGA Board

A clock buffer is used to duplicate the 50 MHz oscillator, so each bank of FPGA I/O bank 3/4/7/8 has two clock inputs. The three programming oscillators are low-jitter oscillators which are used to provide special and high quality clock signals for high-speed transceivers. Figure 2-10 shows the control circuits of programmable oscillators. The clock generator controller in the MAX II CPLD can be used to program the CDCM61001 and CDCM61004 to generate 1G Ethernet SFP+ and SATA reference clocks respectively. The Si570 programmable clock generator is programmed via an I2C serial interface to generate the 10G Ethernet SFP+ reference clock. Two SMA connectors provide external clock input and clock output respectively.

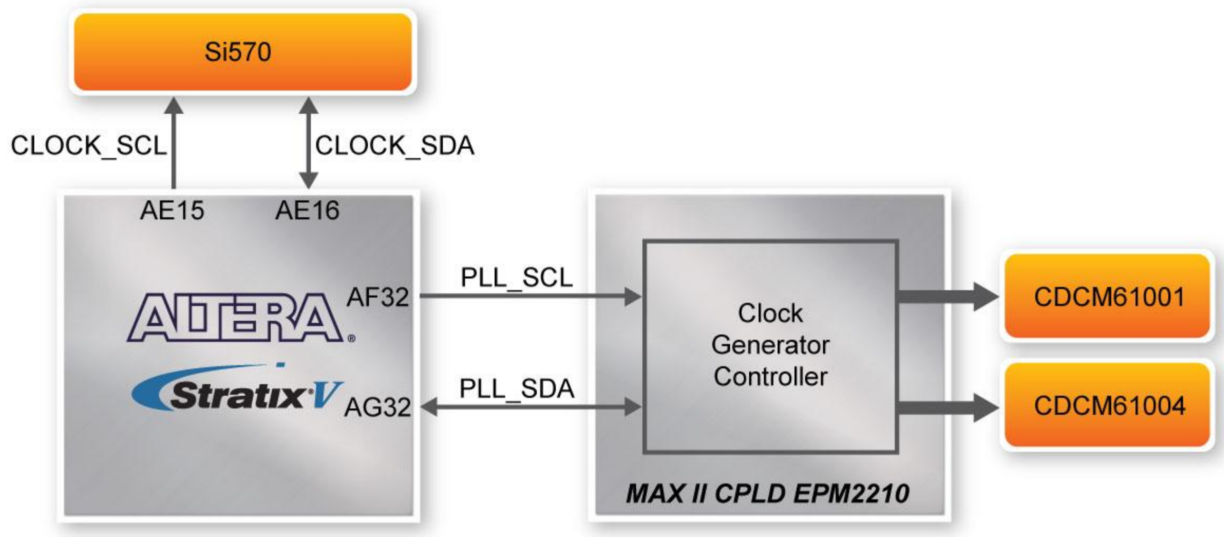


Figure 2-10 Control circuits of Programmable Oscillators

Table 2-8 lists the clock source, signal names, default frequency and their corresponding Stratix V 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*	*Stratix V GX Pin Number*	*Application*
Y4	OSC_50_B3B	50.0 MHz	2.5-V	PIN_AW35	
	OSC_50_B3D		1.8-V	PIN_BC28	
	OSC_50_B4A		1.8-V	PIN_AP10	
	OSC_50_B4D		1.8-V	PIN_AY18	
	OSC_50_B7A		1.5-V	PIN_M8	
	OSC_50_B7D		1.5-V	PIN_J18	
	OSC_50_B8A		1.5-V	PIN_R36	
	OSC_50_B8D		1.8-V	PIN_R25	
J13	SMA_CLKIN	User Defined	2.5V	PIN_BB33	External Clock Input
J14	SMA_CLKOUT	User Defined	2.5V	PIN_AV34	Clock Output
U49	SFP_REFCLK _p	100.0 MHz	LVDS	PIN_AK7	10G SFP+
U53	SFP1G_REFCLK_p	125.0 MHz	LVDS	PIN_AH6	1G SFP+
U28	SATA_HOST_REFCLK_p	125.0 MHz	LVDS	PIN_V6	SATA HOST
U28	SATA_DEVICE_REFCLK_p	125.0 MHz	LVDS	PIN_V39	SATA DEVICE
J17	PCIE_REFCLK_p	From Host	LVDS	PIN_AK38	PCI Express

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

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

Programmable Oscillator	*Schematic Signal Name*	*I/O Standard*	*Stratix V GX Pin Number*	*Description*
Si570 (U49)	CLOCK_SCL	2.5-V	PIN_AE15	I2C bus, direct connected with Si570
	CLOCK_SDA	2.5-V	PIN_AE16	
CDCM61001 (U53)	PLL_SCL	2.5-V	PIN_AF32	I2C bus, connected with MAX II CPLD
	PLL_SDA	2.5-V	PIN_AG32	
CDCM61004 (U28)	PLL_SCL	2.5-V	PIN_AF32	I2C bus, connected with MAX II CPLD
	PLL_SDA	2.5-V	PIN_AG32	

2.6 RS-422 Serial Port

The RS-422 is designed to perform communication between boards, allowing a transmission speed of up to 20 Mbps. Figure 2-11 shows the RS-422 block diagram of the development board. The full-duplex LTC2855 is used to translate the RS-422 signal, and the RJ45 is used as an external connector for the RS-422 signal.

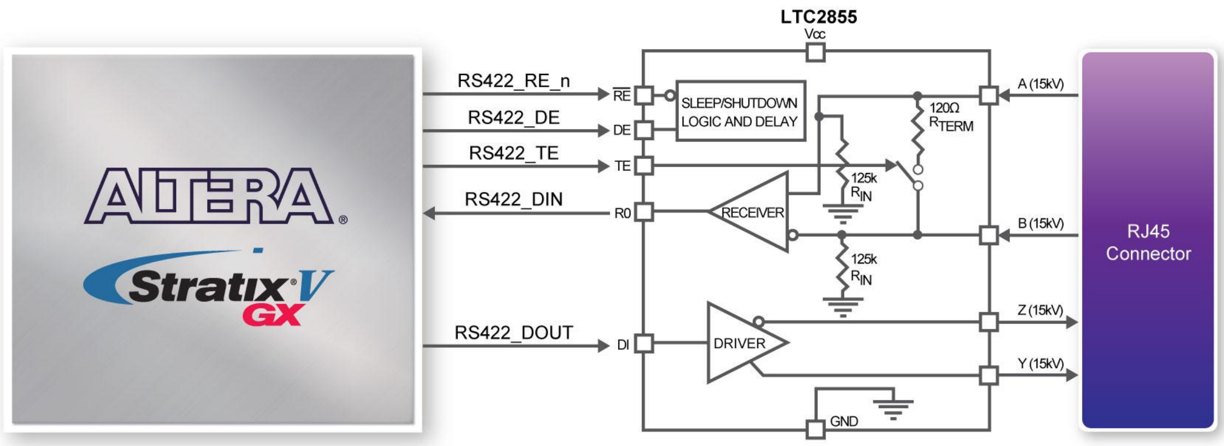


Figure 2-11 Block Diagram of RS-422

Table 2-10 lists the RS-422 pin assignments, signal names and functions.

Table 2-10 RS-422 Pin Assignments, Schematic Signal Names, and Functions

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
RS422_DE	**Driver Enable. A high on DE enables the driver. A low input will force the driver outputs into a high impedance state.**	**2.5-V**	**PIN_AG14**
RS422_DIN	**Receiver Output. The data is send to FPGA.**		**PIN_AE18**
RS422_DOUT	**Driver Input. The data is sent from FPGA.**		**PIN_AE17**
RS422_RE_n	**Receiver Enable. A low enables the receiver. A high input forces the receiver output into a high impedance state.**		**PIN_AF17**
RS422_TE	**Internal Termination Resistance Enable. A high input will connect a termination resistor (120Ω typical) between pins A and B.**		**PIN_AF16**

2.7 FLASH Memory

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

Each interface has a 16-bit data bus and the two devices combined allow for FPP x32 configuration. This device is part of the shared flash and MAX (FM) bus, which connects to the flash memory and MAX II CPLD (EPM2210) System Controller. Figure 2-12 shows the connections between the Flash, MAX and Stratix V GX FPGA.

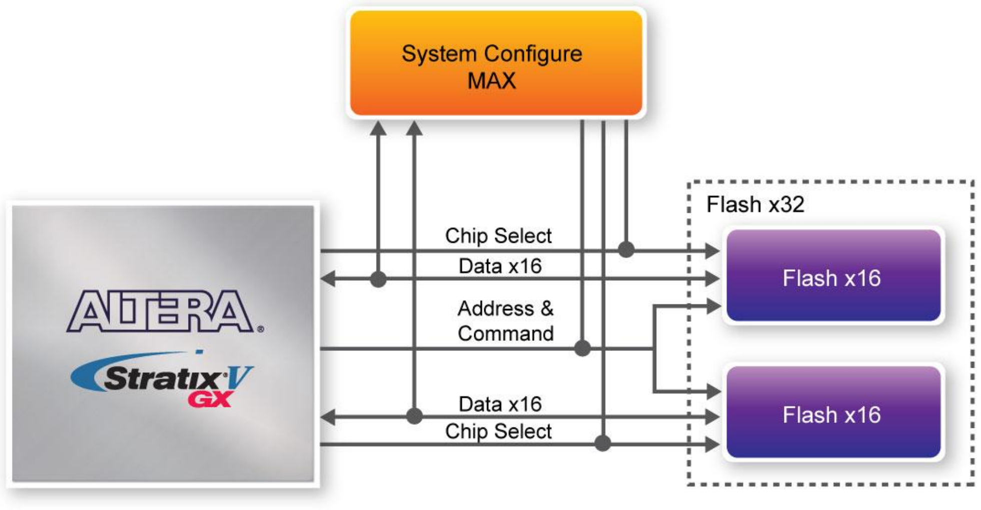


Figure 2-12 Connection between the Flash, Max and Stratix V GX FPGA

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

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
FSM_A0	Address bus	2.5-V	PIN_AU32
FSM_A1	Address bus	2.5-V	PIN_AH30
FSM_A2	Address bus	2.5-V	PIN_AJ30
FSM_A3	Address bus	2.5-V	PIN_AH31
FSM_A4	Address bus	2.5-V	PIN_AK30
FSM_A5	Address bus	2.5-V	PIN_AJ32
FSM_A6	Address bus	2.5-V	PIN_AG33
FSM_A7	Address bus	2.5-V	PIN_AL30
FSM_A8	Address bus	2.5-V	PIN_AK33
FSM_A9	Address bus	2.5-V	PIN_AJ33
FSM_A10	Address bus	2.5-V	PIN_AN30
FSM_A11	Address bus	2.5-V	PIN_AH33
FSM_A12	Address bus	2.5-V	PIN_AK32
FSM_A13	Address bus	2.5-V	PIN_AM32
FSM_A14	Address bus	2.5-V	PIN_AM31
FSM_A15	Address bus	2.5-V	PIN_AL31
FSM_A16	Address bus	2.5-V	PIN_AN33
FSM_A17	Address bus	2.5-V	PIN_AP33
FSM_A18	Address bus	2.5-V	PIN_AT32
FSM_A19	Address bus	2.5-V	PIN_AT29

FSM_A20	Address bus	2.5-V	PIN_AP31
FSM_A21	Address bus	2.5-V	PIN_AR30
FSM_A22	Address bus	2.5-V	PIN_AU30
FSM_A23	Address bus	2.5-V	PIN_AJ31
FSM_A24	Address bus	2.5-V	PIN_AP30
FSM_A25	Address bus	2.5-V	PIN_AN31
FSM_A26	Address bus	2.5-V	PIN_AT30
FSM_D0	Data bus	2.5-V	PIN_AG26
FSM_D1	Data bus	2.5-V	PIN_AD33
FSM_D2	Data bus	2.5-V	PIN_AE34
FSM_D3	Data bus	2.5-V	PIN_AF31
FSM_D4	Data bus	2.5-V	PIN_AG28
FSM_D5	Data bus	2.5-V	PIN_AG30
FSM_D6	Data bus	2.5-V	PIN_AF29
FSM_D7	Data bus	2.5-V	PIN_AE29
FSM_D8	Data bus	2.5-V	PIN_AG25
FSM_D9	Data bus	2.5-V	PIN_AF34
FSM_D10	Data bus	2.5-V	PIN_AE33
FSM_D11	Data bus	2.5-V	PIN_AE31
FSM_D12	Data bus	2.5-V	PIN_AF28
FSM_D13	Data bus	2.5-V	PIN_AE30
FSM_D14	Data bus	2.5-V	PIN_AG29
FSM_D15	Data bus	2.5-V	PIN_AG27
FSM_D16	Data bus	2.5-V	PIN_AP28
FSM_D17	Data bus	2.5-V	PIN_AN28
FSM_D18	Data bus	2.5-V	PIN_AU31
FSM_D19	Data bus	2.5-V	PIN_AW32
FSM_D20	Data bus	2.5-V	PIN_BD32
FSM_D21	Data bus	2.5-V	PIN_AY31
FSM_D22	Data bus	2.5-V	PIN_BA30
FSM_D23	Data bus	2.5-V	PIN_BB30
FSM_D24	Data bus	2.5-V	PIN_AM29
FSM_D25	Data bus	2.5-V	PIN_AR29
FSM_D26	Data bus	2.5-V	PIN_AV31
FSM_D27	Data bus	2.5-V	PIN_AV32
FSM_D28	Data bus	2.5-V	PIN_BC31
FSM_D29	Data bus	2.5-V	PIN_AW30
FSM_D30	Data bus	2.5-V	PIN_BC32
FSM_D31	Data bus	2.5-V	PIN_BD31

FLASH_CLK	Clock	2.5-V	PIN_AL29
FLASH_RESET_n	Reset	2.5-V	PIN_AE28
FLASH_CE_n[0]	Chip enable of of flash-0	2.5-V	PIN_AE27
FLASH_CE_n[1]	Chip enable of of flash-1	2.5-V	PIN_BA31
FLASH_OE_n	Output enable	2.5-V	PIN_AY30
FLASH_WE_n	Write enable	2.5-V	PIN_AR31
FLASH_ADV_n	Address valid	2.5-V	PIN_AK29
FLASH_RDY_BSY_n[0]	Ready of flash-0	2.5-V	PIN_BA29
FLASH_RDY_BSY_n[1]	Ready of flash-1	2.5-V	PIN_BB32

2.8 DDR3 SO-DIMM

The development board supports two independent banks of DDR3 SDRAM SO-DIMM. Each DDR3 SODIMM socket is wired to support a maximum capacity of 8GB with a 64-bit data bus. Using differential DQS signaling for the DDR3 SDRAM interfaces, it is capable of running at up to 800MHz memory clock for a maximum theoretical bandwidth up to 95.4Gbps. **Figure 2-13** shows the connections between the DDR3 SDRAM SO-DIMMs and Stratix V GX FPGA.

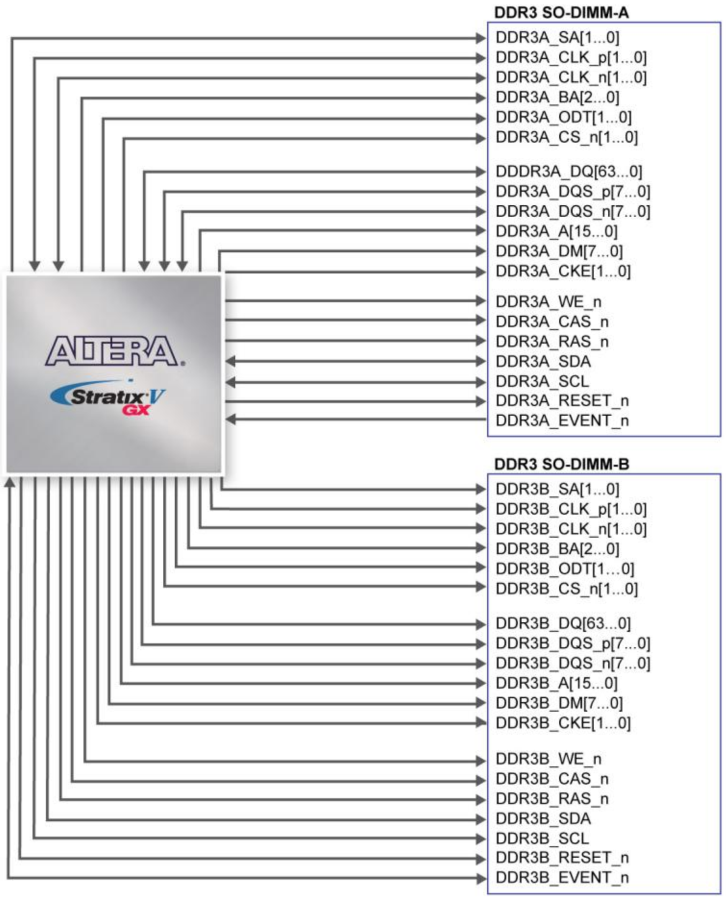


Figure 2-13 Connection between the DDR3 and Stratix V GX FPGA

The pin assignments for DDR3 SDRAM SO-DIMM Bank-A and Bank-B are listed in **Table 2-12** and **Table 2-13**, in respectively.

Table 2-12 DDR3-A Bank Pin Assignments, Schematic Signal Names, and Functions

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
DDR3A_DQ0	Data [0]	SSTL-15 Class I	PIN_A35
DDR3A_DQ1	Data [1]	SSTL-15 Class I	PIN_A34
DDR3A_DQ2	Data [2]	SSTL-15 Class I	PIN_D36
DDR3A_DQ3	Data [3]	SSTL-15 Class I	PIN_C33
DDR3A_DQ4	Data [4]	SSTL-15 Class I	PIN_B32
DDR3A_DQ5	Data [5]	SSTL-15 Class I	PIN_D35
DDR3A_DQ6	Data [6]	SSTL-15 Class I	PIN_D33
DDR3A_DQ7	Data [7]	SSTL-15 Class I	PIN_E33
DDR3A_DQ8	Data [8]	SSTL-15 Class I	PIN_A32
DDR3A_DQ9	Data [9]	SSTL-15 Class I	PIN_A31

DDR3A_DQ10	Data [10]	SSTL-15 Class I	PIN_C30
DDR3A_DQ11	Data [11]	SSTL-15 Class I	PIN_D30
DDR3A_DQ12	Data [12]	SSTL-15 Class I	PIN_B29
DDR3A_DQ13	Data [13]	SSTL-15 Class I	PIN_E30
DDR3A_DQ14	Data [14]	SSTL-15 Class I	PIN_F31
DDR3A_DQ15	Data [15]	SSTL-15 Class I	PIN_G31
DDR3A_DQ16	Data [16]	SSTL-15 Class I	PIN_F35
DDR3A_DQ17	Data [17]	SSTL-15 Class I	PIN_G34
DDR3A_DQ18	Data [18]	SSTL-15 Class I	PIN_J33
DDR3A_DQ19	Data [19]	SSTL-15 Class I	PIN_J34
DDR3A_DQ20	Data [20]	SSTL-15 Class I	PIN_F34
DDR3A_DQ21	Data [21]	SSTL-15 Class I	PIN_E35
DDR3A_DQ22	Data [22]	SSTL-15 Class I	PIN_J31
DDR3A_DQ23	Data [23]	SSTL-15 Class I	PIN_K31
DDR3A_DQ24	Data [24]	SSTL-15 Class I	PIN_P34
DDR3A_DQ25	Data [25]	SSTL-15 Class I	PIN_R33
DDR3A_DQ26	Data [26]	SSTL-15 Class I	PIN_M34
DDR3A_DQ27	Data [27]	SSTL-15 Class I	PIN_L33
DDR3A_DQ28	Data [28]	SSTL-15 Class I	PIN_R34
DDR3A_DQ29	Data [29]	SSTL-15 Class I	PIN_T34
DDR3A_DQ30	Data [30]	SSTL-15 Class I	PIN_W34
DDR3A_DQ31	Data [31]	SSTL-15 Class I	PIN_V35
DDR3A_DQ32	Data [32]	SSTL-15 Class I	PIN_P33
DDR3A_DQ33	Data [33]	SSTL-15 Class I	PIN_P32
DDR3A_DQ34	Data [34]	SSTL-15 Class I	PIN_V33
DDR3A_DQ35	Data [35]	SSTL-15 Class I	PIN_V34
DDR3A_DQ36	Data [36]	SSTL-15 Class I	PIN_N31
DDR3A_DQ37	Data [37]	SSTL-15 Class I	PIN_M31
DDR3A_DQ38	Data [38]	SSTL-15 Class I	PIN_U32
DDR3A_DQ39	Data [39]	SSTL-15 Class I	PIN_U33
DDR3A_DQ40	Data [40]	SSTL-15 Class I	PIN_R31
DDR3A_DQ41	Data [41]	SSTL-15 Class I	PIN_W31
DDR3A_DQ42	Data [42]	SSTL-15 Class I	PIN_U30
DDR3A_DQ43	Data [43]	SSTL-15 Class I	PIN_P31
DDR3A_DQ44	Data [44]	SSTL-15 Class I	PIN_T31
DDR3A_DQ45	Data [45]	SSTL-15 Class I	PIN_Y32
DDR3A_DQ46	Data [46]	SSTL-15 Class I	PIN_T29
DDR3A_DQ47	Data [47]	SSTL-15 Class I	PIN_P30
DDR3A_DQ48	Data [48]	SSTL-15 Class I	PIN_H32
DDR3A_DQ49	Data [49]	SSTL-15 Class I	PIN_H31
DDR3A_DQ50	Data [50]	SSTL-15 Class I	PIN_L30
DDR3A_DQ51	Data [51]	SSTL-15 Class I	PIN_L29
DDR3A_DQ52	Data [52]	SSTL-15 Class I	PIN_F32

DDR3A_DQ53	Data [53]	SSTL-15 Class I	PIN_G32
DDR3A_DQ54	Data [54]	SSTL-15 Class I	PIN_M30
DDR3A_DQ55	Data [55]	SSTL-15 Class I	PIN_N29
DDR3A_DQ56	Data [56]	SSTL-15 Class I	PIN_U29
DDR3A_DQ57	Data [57]	SSTL-15 Class I	PIN_V28
DDR3A_DQ58	Data [58]	SSTL-15 Class I	PIN_Y28
DDR3A_DQ59	Data [59]	SSTL-15 Class I	PIN_W29
DDR3A_DQ60	Data [60]	SSTL-15 Class I	PIN_V30
DDR3A_DQ61	Data [61]	SSTL-15 Class I	PIN_V29
DDR3A_DQ62	Data [62]	SSTL-15 Class I	PIN_W28
DDR3A_DQ63	Data [63]	SSTL-15 Class I	PIN_Y27
DDR3A_DQS0	Data Strobe p[0]	Differential 1.5-V SSTL Class I	PIN_C34
DDR3A_DQS_n0	Data Strobe n[0]	Differential 1.5-V SSTL Class I	PIN_B34
DDR3A_DQS1	Data Strobe p[1]	Differential 1.5-V SSTL Class I	PIN_C31
DDR3A_DQS_n1	Data Strobe n[1]	Differential 1.5-V SSTL Class I	PIN_B31
DDR3A_DQS2	Data Strobe p[2]	Differential 1.5-V SSTL Class I	PIN_H35
DDR3A_DQS_n2	Data Strobe n[2]	Differential 1.5-V SSTL Class I	PIN_G35
DDR3A_DQS3	Data Strobe p[3]	Differential 1.5-V SSTL Class I	PIN_U35
DDR3A_DQS_n3	Data Strobe n[4]	Differential 1.5-V SSTL Class I	PIN_T35
DDR3A_DQS4	Data Strobe p[4]	Differential 1.5-V SSTL Class I	PIN_T33
DDR3A_DQS_n4	Data Strobe n[4]	Differential 1.5-V SSTL Class I	PIN_T32
DDR3A_DQS5	Data Strobe p[5]	Differential 1.5-V SSTL Class I	PIN_T30
DDR3A_DQS_n5	Data Strobe n[5]	Differential 1.5-V SSTL Class I	PIN_R30
DDR3A_DQS6	Data Strobe p[6]	Differential 1.5-V SSTL Class I	PIN_J30
DDR3A_DQS_n6	Data Strobe n[6]	Differential 1.5-V SSTL Class I	PIN_H30
DDR3A_DQS7	Data Strobe p[7]	Differential 1.5-V SSTL Class I	PIN_Y30
DDR3A_DQS_n7	Data Strobe n[7]	Differential 1.5-V SSTL Class I	PIN_Y29
DDR3A_DM0	Data Mask [0]	SSTL-15 Class I	PIN_C36
DDR3A_DM1	Data Mask [1]	SSTL-15 Class I	PIN_E32
DDR3A_DM2	Data Mask [2]	SSTL-15 Class I	PIN_H34
DDR3A_DM3	Data Mask [3]	SSTL-15 Class I	PIN_L32
DDR3A_DM4	Data Mask [4]	SSTL-15 Class I	PIN_N32
DDR3A_DM5	Data Mask [5]	SSTL-15 Class I	PIN_W32
DDR3A_DM6	Data Mask [6]	SSTL-15 Class I	PIN_K30
DDR3A_DM7	Data Mask [7]	SSTL-15 Class I	PIN_T28
DDR3A_A0	Address [0]	SSTL-15 Class I	PIN_M39
DDR3A_A1	Address [1]	SSTL-15 Class I	PIN_L35
DDR3A_A2	Address [2]	SSTL-15 Class I	PIN_N38
DDR3A_A3	Address [3]	SSTL-15 Class I	PIN_L36
DDR3A_A4	Address [4]	SSTL-15 Class I	PIN_H36
DDR3A_A5	Address [5]	SSTL-15 Class I	PIN_K29
DDR3A_A6	Address [6]	SSTL-15 Class I	PIN_D37
DDR3A_A7	Address [7]	SSTL-15 Class I	PIN_K35

DDR3A_A8	Address [8]	SSTL-15 Class I	PIN_K32
DDR3A_A9	Address [9]	SSTL-15 Class I	PIN_K37
DDR3A_A10	Address [10]	SSTL-15 Class I	PIN_M38
DDR3A_A11	Address [11]	SSTL-15 Class I	PIN_C37
DDR3A_A12	Address [12]	SSTL-15 Class I	PIN_K36
DDR3A_A13	Address [13]	SSTL-15 Class I	PIN_M33
DDR3A_A14	Address [14]	SSTL-15 Class I	PIN_K34
DDR3A_A15	Address [15]	SSTL-15 Class I	PIN_B38
DDR3A_RAS_n	Row Address Strobe	SSTL-15 Class I	PIN_P38
DDR3A_CAS_n	Column Address Strobe	SSTL-15 Class I	PIN_M36
DDR3A_BA0	Bank Address [0]	SSTL-15 Class I	PIN_M37
DDR3A_BA1	Bank Address [1]	SSTL-15 Class I	PIN_P39
DDR3A_BA2	Bank Address [2]	SSTL-15 Class I	PIN_J36
DDR3A_CK0	Clock p0	Differential 1.5-V SSTL Class I	PIN_G37
DDR3A_CK_n0	Clock n0	Differential 1.5-V SSTL Class I	PIN_F36
DDR3A_CK1	Clock p1	Differential 1.5-V SSTL Class I	PIN_J37
DDR3A_CK_n1	Clock n1	Differential 1.5-V SSTL Class I	PIN_H37
DDR3A_CKE0	Clock Enable pin 0	SSTL-15 Class I	PIN_E36
DDR3A_CKE1	Clock Enable pin 1	SSTL-15 Class I	PIN_B35
DDR3A_ODT0	On Die Termination[0]	SSTL-15 Class I	PIN_V36
DDR3A_ODT1	On Die Termination[1]	SSTL-15 Class I	PIN_W35
DDR3A_WE_n	Write Enable	SSTL-15 Class I	PIN_N37
DDR3A_CS_n0	Chip Select [0]	SSTL-15 Class I	PIN_P36
DDR3A_CS_n1	Chip Select [1]	SSTL-15 Class I	PIN_R28
DDR3A_RESET_n	Chip Reset	SSTL-15 Class I	PIN_H33
DDR3A_EVENT_n	Chip Temperature Event	SSTL-15 Class I	PIN_K19
DDR3A_SDA	Chip I2C Serial Clock	1.5V	PIN_P15
DDR3A_SCL	Chip I2C Serial Data Bus	1.5V	PIN_C15

Table 2-13 DDR3-B Pin Assignments, Schematic Signal Names, and Functions

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
DDR3B_DQ0	Data [0]	SSTL-15 Class I	PIN_Y17
DDR3B_DQ1	Data [1]	SSTL-15 Class I	PIN_W17
DDR3B_DQ2	Data [2]	SSTL-15 Class I	PIN_V15
DDR3B_DQ3	Data [3]	SSTL-15 Class I	PIN_T15
DDR3B_DQ4	Data [4]	SSTL-15 Class I	PIN_V13
DDR3B_DQ5	Data [5]	SSTL-15 Class I	PIN_V16
DDR3B_DQ6	Data [6]	SSTL-15 Class I	PIN_W14
DDR3B_DQ7	Data [7]	SSTL-15 Class I	PIN_U15

DDR3B_DQ8	Data [8]	SSTL-15 Class I	PIN_T17
DDR3B_DQ9	Data [9]	SSTL-15 Class I	PIN_T16
DDR3B_DQ10	Data [10]	SSTL-15 Class I	PIN_R16
DDR3B_DQ11	Data [11]	SSTL-15 Class I	PIN_P16
DDR3B_DQ12	Data [12]	SSTL-15 Class I	PIN_N16
DDR3B_DQ13	Data [13]	SSTL-15 Class I	PIN_M15
DDR3B_DQ14	Data [14]	SSTL-15 Class I	PIN_M14
DDR3B_DQ15	Data [15]	SSTL-15 Class I	PIN_L14
DDR3B_DQ16	Data [16]	SSTL-15 Class I	PIN_T14
DDR3B_DQ17	Data [17]	SSTL-15 Class I	PIN_U14
DDR3B_DQ18	Data [18]	SSTL-15 Class I	PIN_U11
DDR3B_DQ19	Data [19]	SSTL-15 Class I	PIN_T13
DDR3B_DQ20	Data [20]	SSTL-15 Class I	PIN_U12
DDR3B_DQ21	Data [21]	SSTL-15 Class I	PIN_R13
DDR3B_DQ22	Data [22]	SSTL-15 Class I	PIN_P13
DDR3B_DQ23	Data [23]	SSTL-15 Class I	PIN_N13
DDR3B_DQ24	Data [24]	SSTL-15 Class I	PIN_K12
DDR3B_DQ25	Data [25]	SSTL-15 Class I	PIN_J12
DDR3B_DQ26	Data [26]	SSTL-15 Class I	PIN_J10
DDR3B_DQ27	Data [27]	SSTL-15 Class I	PIN_H12
DDR3B_DQ28	Data [28]	SSTL-15 Class I	PIN_N11
DDR3B_DQ29	Data [29]	SSTL-15 Class I	PIN_M11
DDR3B_DQ30	Data [30]	SSTL-15 Class I	PIN_H10
DDR3B_DQ31	Data [31]	SSTL-15 Class I	PIN_H11
DDR3B_DQ32	Data [32]	SSTL-15 Class I	PIN_T10
DDR3B_DQ33	Data [33]	SSTL-15 Class I	PIN_R10
DDR3B_DQ34	Data [34]	SSTL-15 Class I	PIN_M12
DDR3B_DQ35	Data [35]	SSTL-15 Class I	PIN_L12
DDR3B_DQ36	Data [36]	SSTL-15 Class I	PIN_V10
DDR3B_DQ37	Data [37]	SSTL-15 Class I	PIN_V9
DDR3B_DQ38	Data [38]	SSTL-15 Class I	PIN_R12
DDR3B_DQ39	Data [39]	SSTL-15 Class I	PIN_P12
DDR3B_DQ40	Data [40]	SSTL-15 Class I	PIN_D14
DDR3B_DQ41	Data [41]	SSTL-15 Class I	PIN_C13
DDR3B_DQ42	Data [42]	SSTL-15 Class I	PIN_B14
DDR3B_DQ43	Data [43]	SSTL-15 Class I	PIN_B13
DDR3B_DQ44	Data [44]	SSTL-15 Class I	PIN_E14
DDR3B_DQ45	Data [45]	SSTL-15 Class I	PIN_F14
DDR3B_DQ46	Data [46]	SSTL-15 Class I	PIN_A14
DDR3B_DQ47	Data [47]	SSTL-15 Class I	PIN_A13
DDR3B_DQ48	Data [48]	SSTL-15 Class I	PIN_K13
DDR3B_DQ49	Data [49]	SSTL-15 Class I	PIN_K16
DDR3B_DQ50	Data [50]	SSTL-15 Class I	PIN_H13

DDR3B_DQ51	Data [51]	SSTL-15 Class I	PIN_H14
DDR3B_DQ52	Data [52]	SSTL-15 Class I	PIN_J13
DDR3B_DQ53	Data [53]	SSTL-15 Class I	PIN_J16
DDR3B_DQ54	Data [54]	SSTL-15 Class I	PIN_G13
DDR3B_DQ55	Data [55]	SSTL-15 Class I	PIN_F13
DDR3B_DQ56	Data [56]	SSTL-15 Class I	PIN_D11
DDR3B_DQ57	Data [57]	SSTL-15 Class I	PIN_C10
DDR3B_DQ58	Data [58]	SSTL-15 Class I	PIN_A10
DDR3B_DQ59	Data [59]	SSTL-15 Class I	PIN_B10
DDR3B_DQ60	Data [60]	SSTL-15 Class I	PIN_G11
DDR3B_DQ61	Data [61]	SSTL-15 Class I	PIN_F11
DDR3B_DQ62	Data [62]	SSTL-15 Class I	PIN_E11
DDR3B_DQ63	Data [63]	SSTL-15 Class I	PIN_E12
DDR3B_DQS0	Data Strobe p[0]	Differential 1.5-V SSTL Class I	PIN_Y16
DDR3B_DQS_n0	Data Strobe n[0]	Differential 1.5-V SSTL Class I	PIN_W16
DDR3B_DQS1	Data Strobe p[1]	Differential 1.5-V SSTL Class I	PIN_V17
DDR3B_DQS_n1	Data Strobe n[1]	Differential 1.5-V SSTL Class I	PIN_U17
DDR3B_DQS2	Data Strobe p[2]	Differential 1.5-V SSTL Class I	PIN_P14
DDR3B_DQS_n2	Data Strobe n[2]	Differential 1.5-V SSTL Class I	PIN_N14
DDR3B_DQS3	Data Strobe p[3]	Differential 1.5-V SSTL Class I	PIN_K11
DDR3B_DQS_n3	Data Strobe n[3]	Differential 1.5-V SSTL Class I	PIN_L11
DDR3B_DQS4	Data Strobe p[4]	Differential 1.5-V SSTL Class I	PIN_U9
DDR3B_DQS_n4	Data Strobe n[4]	Differential 1.5-V SSTL Class I	PIN_T9
DDR3B_DQS5	Data Strobe p[5]	Differential 1.5-V SSTL Class I	PIN_E15
DDR3B_DQS_n5	Data Strobe n[5]	Differential 1.5-V SSTL Class I	PIN_D15
DDR3B_DQS6	Data Strobe p[6]	Differential 1.5-V SSTL Class I	PIN_L15
DDR3B_DQS_n6	Data Strobe n[6]	Differential 1.5-V SSTL Class I	PIN_K14
DDR3B_DQS7	Data Strobe p[7]	Differential 1.5-V SSTL Class I	PIN_D12
DDR3B_DQS_n7	Data Strobe n[7]	Differential 1.5-V SSTL Class I	PIN_C12
DDR3B_DM0	Data Mask [0]	SSTL-15 Class I	PIN_R15
DDR3B_DM1	Data Mask [1]	SSTL-15 Class I	PIN_K15
DDR3B_DM2	Data Mask [2]	SSTL-15 Class I	PIN_V12
DDR3B_DM3	Data Mask [3]	SSTL-15 Class I	PIN_G10
DDR3B_DM4	Data Mask [4]	SSTL-15 Class I	PIN_T12
DDR3B_DM5	Data Mask [5]	SSTL-15 Class I	PIN_C16
DDR3B_DM6	Data Mask [6]	SSTL-15 Class I	PIN_H15
DDR3B_DM7	Data Mask [7]	SSTL-15 Class I	PIN_B11
DDR3B_A0	Address [0]	SSTL-15 Class I	PIN_G17
DDR3B_A1	Address [1]	SSTL-15 Class I	PIN_F17
DDR3B_A2	Address [2]	SSTL-15 Class I	PIN_N17
DDR3B_A3	Address [3]	SSTL-15 Class I	PIN_F19
DDR3B_A4	Address [4]	SSTL-15 Class I	PIN_N19
DDR3B_A5	Address [5]	SSTL-15 Class I	PIN_H16

DDR3B_A6	Address [6]	SSTL-15 Class I	PIN_M17
DDR3B_A7	Address [7]	SSTL-15 Class I	PIN_T18
DDR3B_A8	Address [8]	SSTL-15 Class I	PIN_H17
DDR3B_A9	Address [9]	SSTL-15 Class I	PIN_J19
DDR3B_A10	Address [10]	SSTL-15 Class I	PIN_C19
DDR3B_A11	Address [11]	SSTL-15 Class I	PIN_R18
DDR3B_A12	Address [12]	SSTL-15 Class I	PIN_K18
DDR3B_A13	Address [13]	SSTL-15 Class I	PIN_E18
DDR3B_A14	Address [14]	SSTL-15 Class I	PIN_T19
DDR3B_A15	Address [15]	SSTL-15 Class I	PIN_R19
DDR3B_RAS_n	Row Address Strobe	SSTL-15 Class I	PIN_H19
DDR3B_CAS_n	Column Address Strobe	SSTL-15 Class I	PIN_A17
DDR3B_BA0	Bank Address [0]	SSTL-15 Class I	PIN_C18
DDR3B_BA1	Bank Address [1]	SSTL-15 Class I	PIN_G19
DDR3B_BA2	Bank Address [2]	SSTL-15 Class I	PIN_M20
DDR3B_CK0	Clock p0	Differential 1.5-V SSTL Class I	PIN_B16
DDR3B_CK_n0	Clock n0	Differential 1.5-V SSTL Class I	PIN_A16
DDR3B_CK1	Clock p1	Differential 1.5-V SSTL Class I	PIN_E17
DDR3B_CK_n1	Clock n1	Differential 1.5-V SSTL Class I	PIN_D17
DDR3B_CKE0	Clock Enable pin 0	SSTL-15 Class I	PIN_P17
DDR3B_CKE1	Clock Enable pin 1	SSTL-15 Class I	PIN_V18
DDR3B_ODT0	On Die Termination[0]	SSTL-15 Class I	PIN_M18
DDR3B_ODT1	On Die Termination[1]	SSTL-15 Class I	PIN_A19
DDR3B_WE_n	Write Enable	SSTL-15 Class I	PIN_D18
DDR3B_CS_n0	Chip Select [0]	SSTL-15 Class I	PIN_B19
DDR3B_CS_n1	Chip Select [1]	SSTL-15 Class I	PIN_B17
DDR3B_RESET_n	Chip Reset	SSTL-15 Class I	PIN_T20
DDR3B_EVENT_n	Chip Reset	SSTL-15 Class I	PIN_K17
DDR3B_SDA	Chip I2C Serial Clock	1.5V	PIN_P19
DDR3B_SCL	Chip I2C Serial Data Bus	1.5V	PIN_P18

2.9 QDRII+ SRAM

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

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

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
QDRIIA_A0	Address bus[0]	1.8-V HSTL Class I	PIN_AU29
QDRIIA_A1	Address bus[1]	1.8-V HSTL Class I	PIN_BA28
QDRIIA_A2	Address bus[2]	1.8-V HSTL Class I	PIN_AP27
QDRIIA_A3	Address bus[3]	1.8-V HSTL Class I	PIN_AK27
QDRIIA_A4	Address bus[4]	1.8-V HSTL Class I	PIN_AN27
QDRIIA_A5	Address bus[5]	1.8-V HSTL Class I	PIN_AM28
QDRIIA_A6	Address bus[6]	1.8-V HSTL Class I	PIN_AV28
QDRIIA_A7	Address bus[7]	1.8-V HSTL Class I	PIN_AY27
QDRIIA_A8	Address bus[8]	1.8-V HSTL Class I	PIN_BC29
QDRIIA_A9	Address bus[9]	1.8-V HSTL Class I	PIN_AU28
QDRIIA_A10	Address bus[10]	1.8-V HSTL Class I	PIN_AW27
QDRIIA_A11	Address bus[11]	1.8-V HSTL Class I	PIN_AY28
QDRIIA_A12	Address bus[12]	1.8-V HSTL Class I	PIN_BD28
QDRIIA_A13	Address bus[13]	1.8-V HSTL Class I	PIN_AV29
QDRIIA_A14	Address bus[14]	1.8-V HSTL Class I	PIN_AW29
QDRIIA_A15	Address bus[15]	1.8-V HSTL Class I	PIN_BB29
QDRIIA_A16	Address bus[16]	1.8-V HSTL Class I	PIN_BD29
QDRIIA_A17	Address bus[17]	1.8-V HSTL Class I	PIN_AL27
QDRIIA_A18	Address bus[18]	1.8-V HSTL Class I	PIN_AR27
QDRIIA_A19	Address bus[19]	1.8-V HSTL Class I	PIN_AL28
QDRIIA_A20	Address bus[20]	1.8-V HSTL Class I	PIN_AR28
QDRIIA_D0	Write data bus[0]	1.8-V HSTL Class I	PIN_AH28
QDRIIA_D1	Write data bus[1]	1.8-V HSTL Class I	PIN_AH27
QDRIIA_D2	Write data bus[2]	1.8-V HSTL Class I	PIN_AH25
QDRIIA_D3	Write data bus[3]	1.8-V HSTL Class I	PIN_AJ28
QDRIIA_D4	Write data bus[4]	1.8-V HSTL Class I	PIN_AJ27
QDRIIA_D5	Write data bus[5]	1.8-V HSTL Class I	PIN_AJ26
QDRIIA_D6	Write data bus[6]	1.8-V HSTL Class I	PIN_AJ25
QDRIIA_D7	Write data bus[7]	1.8-V HSTL Class I	PIN_AL25
QDRIIA_D8	Write data bus[8]	1.8-V HSTL Class I	PIN_AH24
QDRIIA_D9	Write data bus[9]	1.8-V HSTL Class I	PIN_AN25
QDRIIA_D10	Write data bus[10]	1.8-V HSTL Class I	PIN_AM26
QDRIIA_D11	Write data bus[11]	1.8-V HSTL Class I	PIN_AM25
QDRIIA_D12	Write data bus[12]	1.8-V HSTL Class I	PIN_AL26
QDRIIA_D13	Write data bus[13]	1.8-V HSTL Class I	PIN_AK26
QDRIIA_D14	Write data bus[14]	1.8-V HSTL Class I	PIN_AU27
QDRIIA_D15	Write data bus[15]	1.8-V HSTL Class I	PIN_AU26
QDRIIA_D16	Write data bus[16]	1.8-V HSTL Class I	PIN_AV26
QDRIIA_D17	Write data bus[17]	1.8-V HSTL Class I	PIN_AW26
QDRIIA_Q0	Read Data bus[0]	1.8-V HSTL Class I	PIN_AK23

QDRIIA_Q1	Read Data bus[1]	1.8-V HSTL Class I	PIN_BB26
QDRIIA_Q2	Read Data bus[2]	1.8-V HSTL Class I	PIN_BD26
QDRIIA_Q3	Read Data bus[3]	1.8-V HSTL Class I	PIN_BA24
QDRIIA_Q4	Read Data bus[4]	1.8-V HSTL Class I	PIN_AL23
QDRIIA_Q5	Read Data bus[5]	1.8-V HSTL Class I	PIN_AJ23
QDRIIA_Q6	Read Data bus[6]	1.8-V HSTL Class I	PIN_AL21
QDRIIA_Q7	Read Data bus[7]	1.8-V HSTL Class I	PIN_AK21
QDRIIA_Q8	Read Data bus[8]	1.8-V HSTL Class I	PIN_AJ22
QDRIIA_Q9	Read Data bus[9]	1.8-V HSTL Class I	PIN_AW24
QDRIIA_Q10	Read Data bus[10]	1.8-V HSTL Class I	PIN_BC26
QDRIIA_Q11	Read Data bus[11]	1.8-V HSTL Class I	PIN_AY25
QDRIIA_Q12	Read Data bus[12]	1.8-V HSTL Class I	PIN_AU24
QDRIIA_Q13	Read Data bus[13]	1.8-V HSTL Class I	PIN_AV25
QDRIIA_Q14	Read Data bus[14]	1.8-V HSTL Class I	PIN_AU25
QDRIIA_Q15	Read Data bus[15]	1.8-V HSTL Class I	PIN_AR25
QDRIIA_Q16	Read Data bus[16]	1.8-V HSTL Class I	PIN_AP24
QDRIIA_Q17	Read Data bus[17]	1.8-V HSTL Class I	PIN_AL24
QDRIIA_BWS_n0	Byte Write select[0]	1.8-V HSTL Class I	PIN_AJ24
QDRIIA_BWS_n1	Byte Write select[1]	1.8-V HSTL Class I	PIN_AT27
QDRIIA_K_P	Clock P	Differential 1.8-V HSTL Class I	PIN_AP25
QDRIIA_K_N	Clock N	Differential 1.8-V HSTL Class I	PIN_AR26
QDRIIA_CQ_P	Echo clock P	1.8-V HSTL Class I	PIN_AH22
QDRIIA_CQ_N	Echo clock N	1.8-V HSTL Class I	PIN_BA25
QDRIIA_RPS_n	Report Select	1.8-V HSTL Class I	PIN_AT26
QDRIIA_WPS_n	Write Port Select	1.8-V HSTL Class I	PIN_AK24
QDRIIA_DOFF_n	DLL enable	1.8-V HSTL Class I	PIN_AR23
QDRIIA_ODT	On-Die Termination Input	1.8-V HSTL Class I	PIN_AN23
QDRII_QVLD	Valid Output Indicator	1.8-V HSTL Class I	PIN_AM23

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
QDRIIB_A0	Address bus[0]	1.8-V HSTL Class I	PIN_AR24
QDRIIB_A1	Address bus[1]	1.8-V HSTL Class I	PIN_BB23
QDRIIB_A2	Address bus[2]	1.8-V HSTL Class I	PIN_AK20
QDRIIB_A3	Address bus[3]	1.8-V HSTL Class I	PIN_AJ19
QDRIIB_A4	Address bus[4]	1.8-V HSTL Class I	PIN_AL20
QDRIIB_A5	Address bus[5]	1.8-V HSTL Class I	PIN_AG19
QDRIIB_A6	Address bus[6]	1.8-V HSTL Class I	PIN_AT23

QDRIIB_A7	Address bus[7]	1.8-V HSTL Class I	PIN_AU23
QDRIIB_A8	Address bus[8]	1.8-V HSTL Class I	PIN_AV23
QDRIIB_A9	Address bus[9]	1.8-V HSTL Class I	PIN_AM22
QDRIIB_A10	Address bus[10]	1.8-V HSTL Class I	PIN_AJ20
QDRIIB_A11	Address bus[11]	1.8-V HSTL Class I	PIN_AG20
QDRIIB_A12	Address bus[12]	1.8-V HSTL Class I	PIN_AW23
QDRIIB_A13	Address bus[13]	1.8-V HSTL Class I	PIN_BB24
QDRIIB_A14	Address bus[14]	1.8-V HSTL Class I	PIN_AY24
QDRIIB_A15	Address bus[15]	1.8-V HSTL Class I	PIN_BD23
QDRIIB_A16	Address bus[16]	1.8-V HSTL Class I	PIN_BC23
QDRIIB_A17	Address bus[17]	1.8-V HSTL Class I	PIN_AG21
QDRIIB_A18	Address bus[18]	1.8-V HSTL Class I	PIN_AM20
QDRIIB_A19	Address bus[19]	1.8-V HSTL Class I	PIN_AK18
QDRIIB_A20	Address bus[20]	1.8-V HSTL Class I	PIN_AN22
QDRIIB_D0	Write data bus[0]	1.8-V HSTL Class I	PIN_BB21
QDRIIB_D1	Write data bus[1]	1.8-V HSTL Class I	PIN_BD20
QDRIIB_D2	Write data bus[2]	1.8-V HSTL Class I	PIN_BC20
QDRIIB_D3	Write data bus[3]	1.8-V HSTL Class I	PIN_AR22
QDRIIB_D4	Write data bus[4]	1.8-V HSTL Class I	PIN_BB20
QDRIIB_D5	Write data bus[5]	1.8-V HSTL Class I	PIN_AU22
QDRIIB_D6	Write data bus[6]	1.8-V HSTL Class I	PIN_BA21
QDRIIB_D7	Write data bus[7]	1.8-V HSTL Class I	PIN_AY21
QDRIIB_D8	Write data bus[8]	1.8-V HSTL Class I	PIN_AW21
QDRIIB_D9	Write data bus[9]	1.8-V HSTL Class I	PIN_AT21
QDRIIB_D10	Write data bus[10]	1.8-V HSTL Class I	PIN_AR21
QDRIIB_D11	Write data bus[11]	1.8-V HSTL Class I	PIN_AP21
QDRIIB_D12	Write data bus[12]	1.8-V HSTL Class I	PIN_BD22
QDRIIB_D13	Write data bus[13]	1.8-V HSTL Class I	PIN_BC22
QDRIIB_D14	Write data bus[14]	1.8-V HSTL Class I	PIN_BA22
QDRIIB_D15	Write data bus[15]	1.8-V HSTL Class I	PIN_AV22
QDRIIB_D16	Write data bus[16]	1.8-V HSTL Class I	PIN_AY22
QDRIIB_D17	Write data bus[17]	1.8-V HSTL Class I	PIN_AW22
QDRIIB_Q0	Read Data bus[0]	1.8-V HSTL Class I	PIN_AR19
QDRIIB_Q1	Read Data bus[1]	1.8-V HSTL Class I	PIN_AM19
QDRIIB_Q2	Read Data bus[2]	1.8-V HSTL Class I	PIN_AL19
QDRIIB_Q3	Read Data bus[3]	1.8-V HSTL Class I	PIN_AM17
QDRIIB_Q4	Read Data bus[4]	1.8-V HSTL Class I	PIN_AL18
QDRIIB_Q5	Read Data bus[5]	1.8-V HSTL Class I	PIN_AN19
QDRIIB_Q6	Read Data bus[6]	1.8-V HSTL Class I	PIN_AU18
QDRIIB_Q7	Read Data bus[7]	1.8-V HSTL Class I	PIN_AK17
QDRIIB_Q8	Read Data bus[8]	1.8-V HSTL Class I	PIN_AL17
QDRIIB_Q9	Read Data bus[9]	1.8-V HSTL Class I	PIN_AG17
QDRIIB_Q10	Read Data bus[10]	1.8-V HSTL Class I	PIN_AJ18

QDRIIB_Q11	Read Data bus[11]	1.8-V HSTL Class I	PIN_AJ17
QDRIIB_Q12	Read Data bus[12]	1.8-V HSTL Class I	PIN_AG18
QDRIIB_Q13	Read Data bus[13]	1.8-V HSTL Class I	PIN_AU19
QDRIIB_Q14	Read Data bus[14]	1.8-V HSTL Class I	PIN_AW19
QDRIIB_Q15	Read Data bus[15]	1.8-V HSTL Class I	PIN_AV19
QDRIIB_Q16	Read Data bus[16]	1.8-V HSTL Class I	PIN_AP19
QDRIIB_Q17	Read Data bus[17]	1.8-V HSTL Class I	PIN_AN20
QDRIIB_BWS_n0	Byte Write select[0]	1.8-V HSTL Class I	PIN_AV20
QDRIIB_BWS_n1	Byte Write select[1]	1.8-V HSTL Class I	PIN_AU21
QDRIIB_K_p	Clock P	Differential 1.8-V HSTL Class I	PIN_AR20
QDRIIB_K_n	Clock N	Differential 1.8-V HSTL Class I	PIN_AT20
QDRIIB_CQ_p	Echo clock P	1.8-V HSTL Class I	PIN_AJ15
QDRIIB_CQ_n	Echo clock N	1.8-V HSTL Class I	PIN_AP18
QDRIIB_RPS_n	Report Select	1.8-V HSTL Class I	PIN_AW20
QDRIIB_WPS_n	Write Port Select	1.8-V HSTL Class I	PIN_AU20
QDRIIB_DOFF_n	PLL Turn Off	1.8-V HSTL Class I	PIN_AH19
QDRIIB_ODT	On-Die Termination Input	1.8-V HSTL Class I	PIN_AH18
QDRIIB_QVLD	Valid Output Indicator	1.8-V HSTL Class I	PIN_AJ16

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
QDRIIC_A0	Address bus[0]	1.8-V HSTL Class I	PIN_AV16
QDRIIC_A1	Address bus[1]	1.8-V HSTL Class I	PIN_AW16
QDRIIC_A2	Address bus[2]	1.8-V HSTL Class I	PIN_AP16
QDRIIC_A3	Address bus[3]	1.8-V HSTL Class I	PIN_AW9
QDRIIC_A4	Address bus[4]	1.8-V HSTL Class I	PIN_BD7
QDRIIC_A5	Address bus[5]	1.8-V HSTL Class I	PIN_BC7
QDRIIC_A6	Address bus[6]	1.8-V HSTL Class I	PIN_AR17
QDRIIC_A7	Address bus[7]	1.8-V HSTL Class I	PIN_AR18
QDRIIC_A8	Address bus[8]	1.8-V HSTL Class I	PIN_AT17
QDRIIC_A9	Address bus[9]	1.8-V HSTL Class I	PIN_BB9
QDRIIC_A10	Address bus[10]	1.8-V HSTL Class I	PIN_AH21
QDRIIC_A11	Address bus[11]	1.8-V HSTL Class I	PIN_AG20
QDRIIC_A12	Address bus[12]	1.8-V HSTL Class I	PIN_AU16
QDRIIC_A13	Address bus[13]	1.8-V HSTL Class I	PIN_BB8
QDRIIC_A14	Address bus[14]	1.8-V HSTL Class I	PIN_AT18
QDRIIC_A15	Address bus[15]	1.8-V HSTL Class I	PIN_AW17
QDRIIC_A16	Address bus[16]	1.8-V HSTL Class I	PIN_AV17
QDRIIC_A17	Address bus[17]	1.8-V HSTL Class I	PIN_AU8
QDRIIC_A18	Address bus[18]	1.8-V HSTL Class I	PIN_AT9

QDRIIC_A19	Address bus[19]	1.8-V HSTL Class I	PIN_AV8
QDRIIC_A20	Address bus[20]	1.8-V HSTL Class I	PIN_AN17
QDRIIC_D0	Write data bus[0]	1.8-V HSTL Class I	PIN_AG9
QDRIIC_D1	Write data bus[1]	1.8-V HSTL Class I	PIN_AG10
QDRIIC_D2	Write data bus[2]	1.8-V HSTL Class I	PIN_AG12
QDRIIC_D3	Write data bus[3]	1.8-V HSTL Class I	PIN_AG11
QDRIIC_D4	Write data bus[4]	1.8-V HSTL Class I	PIN_AV10
QDRIIC_D5	Write data bus[5]	1.8-V HSTL Class I	PIN_AH12
QDRIIC_D6	Write data bus[6]	1.8-V HSTL Class I	PIN_AK12
QDRIIC_D7	Write data bus[7]	1.8-V HSTL Class I	PIN_AL12
QDRIIC_D8	Write data bus[8]	1.8-V HSTL Class I	PIN_AJ12
QDRIIC_D9	Write data bus[9]	1.8-V HSTL Class I	PIN_AN12
QDRIIC_D10	Write data bus[10]	1.8-V HSTL Class I	PIN_AM13
QDRIIC_D11	Write data bus[11]	1.8-V HSTL Class I	PIN_AR12
QDRIIC_D12	Write data bus[12]	1.8-V HSTL Class I	PIN_AR13
QDRIIC_D13	Write data bus[13]	1.8-V HSTL Class I	PIN_AU9
QDRIIC_D14	Write data bus[14]	1.8-V HSTL Class I	PIN_AU10
QDRIIC_D15	Write data bus[15]	1.8-V HSTL Class I	PIN_AU11
QDRIIC_D16	Write data bus[16]	1.8-V HSTL Class I	PIN_AV11
QDRIIC_D17	Write data bus[17]	1.8-V HSTL Class I	PIN_AT12
QDRIIC_Q0	Read Data bus[0]	1.8-V HSTL Class I	PIN_BA12
QDRIIC_Q1	Read Data bus[1]	1.8-V HSTL Class I	PIN_AF14
QDRIIC_Q2	Read Data bus[2]	1.8-V HSTL Class I	PIN_AE13
QDRIIC_Q3	Read Data bus[3]	1.8-V HSTL Class I	PIN_AD14
QDRIIC_Q4	Read Data bus[4]	1.8-V HSTL Class I	PIN_AE12
QDRIIC_Q5	Read Data bus[5]	1.8-V HSTL Class I	PIN_AF11
QDRIIC_Q6	Read Data bus[6]	1.8-V HSTL Class I	PIN_AE11
QDRIIC_Q7	Read Data bus[7]	1.8-V HSTL Class I	PIN_AE10
QDRIIC_Q8	Read Data bus[8]	1.8-V HSTL Class I	PIN_AE9
QDRIIC_Q9	Read Data bus[9]	1.8-V HSTL Class I	PIN_BB11
QDRIIC_Q10	Read Data bus[10]	1.8-V HSTL Class I	PIN_AW11
QDRIIC_Q11	Read Data bus[11]	1.8-V HSTL Class I	PIN_AF10
QDRIIC_Q12	Read Data bus[12]	1.8-V HSTL Class I	PIN_AY12
QDRIIC_Q13	Read Data bus[13]	1.8-V HSTL Class I	PIN_AW10
QDRIIC_Q14	Read Data bus[14]	1.8-V HSTL Class I	PIN_AY10
QDRIIC_Q15	Read Data bus[15]	1.8-V HSTL Class I	PIN_BB12
QDRIIC_Q16	Read Data bus[16]	1.8-V HSTL Class I	PIN_BC10
QDRIIC_Q17	Read Data bus[17]	1.8-V HSTL Class I	PIN_BA10
QDRIIC_BWS_n0	Byte Write select[0]	1.8-V HSTL Class I	PIN_AJ11
QDRIIC_BWS_n1	Byte Write select[1]	1.8-V HSTL Class I	PIN_AJ10
QDRIIC_K_p	Clock P	Differential 1.8-V HSTL Class I	PIN_AP12
QDRIIC_K_n	Clock N	Differential 1.8-V HSTL Class I	PIN_AP13

QDRIIC_CQ_p	Echo clock P	1.8-V HSTL Class I	PIN_BC11
QDRIIC_CQ_n	Echo clock N	1.8-V HSTL Class I	PIN_AF13
QDRIIC_RPS_n	Report Select	1.8-V HSTL Class I	PIN_AH10
QDRIIC_WPS_n	Write Port Select	1.8-V HSTL Class I	PIN_AL11
QDRIIC_DOFF_n	PLL Turn Off	1.8-V HSTL Class I	PIN_AE14
QDRIIC_ODT	On-Die Termination Input	1.8-V HSTL Class I	PIN_BD10
QDRIIC_QVLD	Valid Output Indicator	1.8-V HSTL Class I	PIN_BD11

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
QDRIID_A0	Address bus[0]	1.8-V HSTL Class I	PIN_N26
QDRIID_A1	Address bus[1]	1.8-V HSTL Class I	PIN_P28
QDRIID_A2	Address bus[2]	1.8-V HSTL Class I	PIN_N28
QDRIID_A3	Address bus[3]	1.8-V HSTL Class I	PIN_L26
QDRIID_A4	Address bus[4]	1.8-V HSTL Class I	PIN_K27
QDRIID_A5	Address bus[5]	1.8-V HSTL Class I	PIN_L27
QDRIID_A6	Address bus[6]	1.8-V HSTL Class I	PIN_U26
QDRIID_A7	Address bus[7]	1.8-V HSTL Class I	PIN_T26
QDRIID_A8	Address bus[8]	1.8-V HSTL Class I	PIN_T27
QDRIID_A9	Address bus[9]	1.8-V HSTL Class I	PIN_V27
QDRIID_A10	Address bus[10]	1.8-V HSTL Class I	PIN_U27
QDRIID_A11	Address bus[11]	1.8-V HSTL Class I	PIN_R27
QDRIID_A12	Address bus[12]	1.8-V HSTL Class I	PIN_P27
QDRIID_A13	Address bus[13]	1.8-V HSTL Class I	PIN_V25
QDRIID_A14	Address bus[14]	1.8-V HSTL Class I	PIN_V26
QDRIID_A15	Address bus[15]	1.8-V HSTL Class I	PIN_T25
QDRIID_A16	Address bus[16]	1.8-V HSTL Class I	PIN_P26
QDRIID_A17	Address bus[17]	1.8-V HSTL Class I	PIN_M27
QDRIID_A18	Address bus[18]	1.8-V HSTL Class I	PIN_M28
QDRIID_A19	Address bus[19]	1.8-V HSTL Class I	PIN_P29
QDRIID_A20	Address bus[20]	1.8-V HSTL Class I	PIN_D29
QDRIID_D0	Write data bus[0]	1.8-V HSTL Class I	PIN_H25
QDRIID_D1	Write data bus[1]	1.8-V HSTL Class I	PIN_H24
QDRIID_D2	Write data bus[2]	1.8-V HSTL Class I	PIN_H23
QDRIID_D3	Write data bus[3]	1.8-V HSTL Class I	PIN_J25
QDRIID_D4	Write data bus[4]	1.8-V HSTL Class I	PIN_J24
QDRIID_D5	Write data bus[5]	1.8-V HSTL Class I	PIN_K25
QDRIID_D6	Write data bus[6]	1.8-V HSTL Class I	PIN_D26

QDRIID_D7	Write data bus[7]	1.8-V HSTL Class I	PIN_F25
QDRIID_D8	Write data bus[8]	1.8-V HSTL Class I	PIN_G25
QDRIID_D9	Write data bus[9]	1.8-V HSTL Class I	PIN_N23
QDRIID_D10	Write data bus[10]	1.8-V HSTL Class I	PIN_P24
QDRIID_D11	Write data bus[11]	1.8-V HSTL Class I	PIN_P23
QDRIID_D12	Write data bus[12]	1.8-V HSTL Class I	PIN_L24
QDRIID_D13	Write data bus[13]	1.8-V HSTL Class I	PIN_R24
QDRIID_D14	Write data bus[14]	1.8-V HSTL Class I	PIN_U23
QDRIID_D15	Write data bus[15]	1.8-V HSTL Class I	PIN_U24
QDRIID_D16	Write data bus[16]	1.8-V HSTL Class I	PIN_T24
QDRIID_D17	Write data bus[17]	1.8-V HSTL Class I	PIN_T23
QDRIID_Q0	Read Data bus[0]	1.8-V HSTL Class I	PIN_C27
QDRIID_Q1	Read Data bus[1]	1.8-V HSTL Class I	PIN_A26
QDRIID_Q2	Read Data bus[2]	1.8-V HSTL Class I	PIN_B26
QDRIID_Q3	Read Data bus[3]	1.8-V HSTL Class I	PIN_F26
QDRIID_Q4	Read Data bus[4]	1.8-V HSTL Class I	PIN_G26
QDRIID_Q5	Read Data bus[5]	1.8-V HSTL Class I	PIN_C28
QDRIID_Q6	Read Data bus[6]	1.8-V HSTL Class I	PIN_A29
QDRIID_Q7	Read Data bus[7]	1.8-V HSTL Class I	PIN_A28
QDRIID_Q8	Read Data bus[8]	1.8-V HSTL Class I	PIN_B28
QDRIID_Q9	Read Data bus[9]	1.8-V HSTL Class I	PIN_G28
QDRIID_Q10	Read Data bus[10]	1.8-V HSTL Class I	PIN_F28
QDRIID_Q11	Read Data bus[11]	1.8-V HSTL Class I	PIN_D27
QDRIID_Q12	Read Data bus[12]	1.8-V HSTL Class I	PIN_G29
QDRIID_Q13	Read Data bus[13]	1.8-V HSTL Class I	PIN_F29
QDRIID_Q14	Read Data bus[14]	1.8-V HSTL Class I	PIN_H28
QDRIID_Q15	Read Data bus[15]	1.8-V HSTL Class I	PIN_K28
QDRIID_Q16	Read Data bus[16]	1.8-V HSTL Class I	PIN_J28
QDRIID_Q17	Read Data bus[17]	1.8-V HSTL Class I	PIN_H29
QDRIID_BWS_n0	Byte Write select[0]	1.8-V HSTL Class I	PIN_E26
QDRIID_BWS_n1	Byte Write select[1]	1.8-V HSTL Class I	PIN_K26
QDRIID_K_p	Clock P	Differential 1.8-V HSTL Class I	PIN_L23
QDRIID_K_n	Clock N	Differential 1.8-V HSTL Class I	PIN_K24
QDRIID_CQ_p	Echo clock P	1.8-V HSTL Class I	PIN_E29
QDRIID_CQ_n	Echo clock N	1.8-V HSTL Class I	PIN_H27
QDRIID_RPS_n	Report Select	1.8-V HSTL Class I	PIN_F24
QDRIID_WPS_n	Write Port Select	1.8-V HSTL Class I	PIN_M23
QDRIID_DOFF_n	PLL Turn Off	1.8-V HSTL Class I	PIN_E27
QDRIID_ODT	On-Die Termination Input	1.8-V HSTL Class I	PIN_H26

QDRIID_QVLD	**Valid Output Indicator**	**1.8-V HSTL Class I**	**PIN_J27**

2.10 SPF+ Ports

The development board has four independent 10G SFP+ connectors that use one transceiver channel each from the Stratix V GX FPGA device. These modules take in serial data from the Stratix V GX FPGA device and transform them to optical signals. The board includes cage assemblies for the SFP+ connectors. Figure 2-14 shows the connections between the SFP+ and Stratix V GX FPGA.

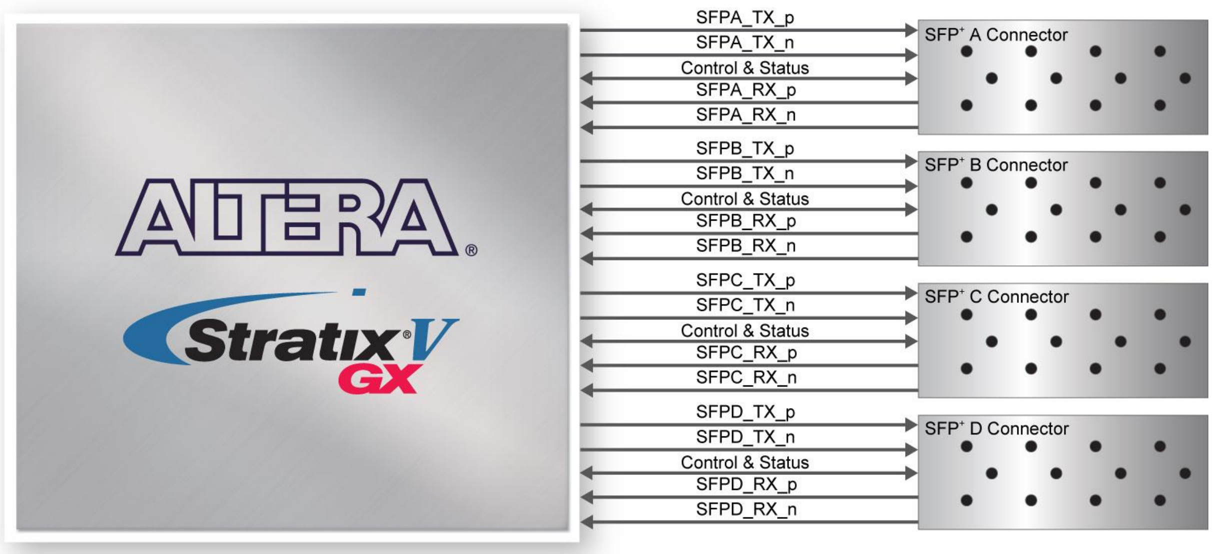


Figure 2-14 Connection between the SFP+ and Stratix V GX FPGA

Table 2-18 and Table 2-19 list the SFP+ A, B, C and D pin assignments and signal names relative to the Stratix V GX device.

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

Schematic Signal Name	***Description***	***I/O Standard***	***Stratix V GX Pin Number***
SFPA_TX_p	**Transmitter data**	**1.4-V PCML**	**PIN_AG4**
SFPA_TX_n	**Transmitter data**	**1.4-V PCML**	**PIN_AG3**
SFPA_RX_p	**Receiver data**	**1.4-V PCML**	**PIN_AK2**

SFPA_RX_n	Receiver data	1.4-V PCML	PIN_AK1
SFPA_LOS	Signal loss indicator	2.5V	PIN_F22
SFPA_MOD0_PRSNT_n	Module present	2.5V	PIN_E21
SFPA_MOD1_SCL	Serial 2-wire clock	2.5V	PIN_B20
SFPA_MOD2_SDA	Serial 2-wire data	2.5V	PIN_A20
SFPA_RATESEL0	Rate select 0	2.5V	PIN_E20
SFPA_RATESEL1	Rate select 1	2.5V	PIN_G22
SFPA_TXDISABLE	Turns off and disables the transmitter output	2.5V	PIN_B22
SFPA_TXFAULT	Transmitter fault	2.5V	PIN_A22

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
SFPB_TX_p	Transmitter data	1.4-V PCML	PIN_AL4
SFPB_TX_n	Transmitter data	1.4-V PCML	PIN_AL3
SFPB_RX_p	Receiver data	1.4-V PCML	PIN_AP2
SFPB_RX_n	Receiver data	1.4-V PCML	PIN_AP1
SFPB_LOS	Signal loss indicator	2.5V	PIN_R22
SFPB_MOD0_PRSNT_n	Module present	2.5V	PIN_K22
SFPB_MOD1_SCL	Serial 2-wire clock	2.5V	PIN_K21
SFPB_MOD2_SDA	Serial 2-wire data	2.5V	PIN_K20
SFPB_RATESEL0	Rate select 0	2.5V	PIN_R21
SFPB_RATESEL1	Rate select 1	2.5V	PIN_T22
SFPB_TXDISABLE	Turns off and disables the transmitter output	2.5V	PIN_H22
SFPB_TXFAULT	Transmitter fault	2.5V	PIN_H20

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
SFPC_TX_p	Transmitter data	1.4-V PCML	PIN_AT6
SFPC_TX_n	Transmitter data	1.4-V PCML	PIN_AT5
SFPC_RX_p	Receiver data	1.4-V PCML	PIN_AW4
SFPC_RX_n	Receiver data	1.4-V PCML	PIN_AW3
SFPC_LOS	Signal loss indicator	2.5V	PIN_L21
SFPC_MOD0_PRSNT_n	Module present	2.5V	PIN_J21
SFPC_MOD1_SCL	Serial 2-wire clock	2.5V	PIN_H21
SFPC_MOD2_SDA	Serial 2-wire data	2.5V	PIN_G20
SFPC_RATESEL0	Rate select 0	2.5V	PIN_J22
SFPC_RATESEL1	Rate select 1	2.5V	PIN_P21
SFPC_TXDISABLE	Turns off and disables the transmitter output	2.5V	PIN_F21
SFPC_TXFAULT	Transmitter fault	2.5V	PIN_F20

Table 2-21 SFP+ C Pin Assignments, Schematic Signal Names, and Functions

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
SFPD_TX_p	Transmitter data	1.4-V PCML	PIN_AY6
SFPD_TX_n	Transmitter data	1.4-V PCML	PIN_AY5
SFPD_RX_p	Receiver data	1.4-V PCML	PIN_BB2
SFPD_RX_n	Receiver data	1.4-V PCML	PIN_BB1
SFPD_LOS	Signal loss indicator	2.5V	PIN_N22
SFPD_MOD0_PRSNT_n	Module present	2.5V	PIN_V20
SFPD_MOD1_SCL	Serial 2-wire clock	2.5V	PIN_U21
SFPD_MOD2_SDA	Serial 2-wire data	2.5V	PIN_V19
SFPD_RATESEL0	Rate select 0	2.5V	PIN_V21
SFPD_RATESEL1	Rate select 1	2.5V	PIN_M22
SFPD_TXDISABLE	Turns off and disables the transmitter output	2.5V	PIN_U20
SFPD_TXFAULT	Transmitter fault	2.5V	PIN_T21

2.11 PCI Express

The FPGA development board is designed to fit entirely into a PC motherboard with x8 or x16 PCI Express slot. Utilizing built-in transceivers on a Stratix V 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 Stratix V GX device, it will allow users to implement simple and fast protocol, as well as saving logic resources for logic application. **Figure 2-15** presents the pin connection established between the Stratix V GX and PCI Express.

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

Please note that it is a requirement that you connect the PCIe external power connector to 6-pin 12V DC power connector in the FPGA to avoid FPGA damage due to insufficient power. The PCIE_REFCLK_p signal is a differential input that is driven from the PC motherboard on this board through the PCIe edge connector. A DIP switch (SW7) is connected to the PCI Express to allow different configurations to enable a x1, x4, or x8 PCIe.

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

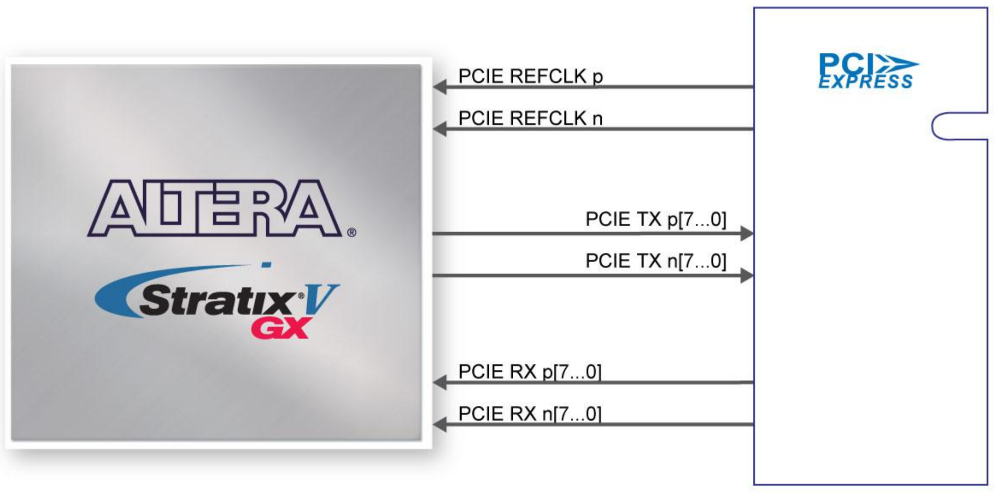


Figure 2-15 PCI Express pin connection

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

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
PCIE_TX_p0	Add-in card transmit bus	1.4-V PCML	PIN_AY39
PCIE_TX_n0	Add-in card transmit bus	1.4-V PCML	PIN_AY40
PCIE_TX_p1	Add-in card transmit bus	1.4-V PCML	PIN_AV39
PCIE_TX_n1	Add-in card transmit bus	1.4-V PCML	PIN_AV40
PCIE_TX_p2	Add-in card transmit bus	1.4-V PCML	PIN_AT39
PCIE_TX_n2	Add-in card transmit bus	1.4-V PCML	PIN_AT40
PCIE_TX_p3	Add-in card transmit bus	1.4-V PCML	PIN_AU41
PCIE_TX_n3	Add-in card transmit bus	1.4-V PCML	PIN_AU42
PCIE_TX_p4	Add-in card transmit bus	1.4-V PCML	PIN_AN41
PCIE_TX_n4	Add-in card transmit bus	1.4-V PCML	PIN_AN42
PCIE_TX_p5	Add-in card transmit bus	1.4-V PCML	PIN_AL41
PCIE_TX_n5	Add-in card transmit bus	1.4-V PCML	PIN_AL42
PCIE_TX_p6	Add-in card transmit bus	1.4-V PCML	PIN_AJ41
PCIE_TX_n6	Add-in card transmit bus	1.4-V PCML	PIN_AJ42
PCIE_TX_p7	Add-in card transmit bus	1.4-V PCML	PIN_AG41
PCIE_TX_n7	Add-in card transmit bus	1.4-V PCML	PIN_AG42
PCIE_RX_p0	Add-in card receive bus	1.4-V PCML	PIN_BB43
PCIE_RX_n0	Add-in card receive bus	1.4-V PCML	PIN_BB44
PCIE_RX_p1	Add-in card receive bus	1.4-V PCML	PIN_BA41

PCIE_RX_n1	Add-in card receive bus	1.4-V PCML	PIN_BA42
PCIE_RX_p2	Add-in card receive bus	1.4-V PCML	PIN_AW41
PCIE_RX_n2	Add-in card receive bus	1.4-V PCML	PIN_AW42
PCIE_RX_p3	Add-in card receive bus	1.4-V PCML	PIN_AY43
PCIE_RX_n3	Add-in card receive bus	1.4-V PCML	PIN_AY44
PCIE_RX_p4	Add-in card receive bus	1.4-V PCML	PIN_AT43
PCIE_RX_n4	Add-in card receive bus	1.4-V PCML	PIN_AT44
PCIE_RX_p5	Add-in card receive bus	1.4-V PCML	PIN_AP43
PCIE_RX_n5	Add-in card receive bus	1.4-V PCML	PIN_AP44
PCIE_RX_p6	Add-in card receive bus	1.4-V PCML	PIN_AM43
PCIE_RX_n6	Add-in card receive bus	1.4-V PCML	PIN_AM44
PCIE_RX_p7	Add-in card receive bus	1.4-V PCML	PIN_AK43
PCIE_RX_n7	Add-in card receive bus	1.4-V PCML	PIN_AK44
PCIE_REFCLK_p	Motherboard reference clock	HCSL	PIN_AK38
PCIE_REFCLK_n	Motherboard reference clock	HCSL	PIN_AK39
PCIE_PERST_n	Reset	2.5-V	PIN_AU33
PCIE_SMBCLK	SMB clock	2.5-V	PIN_BD34
PCIE_SMBDAT	SMB data	2.5-V	PIN_AT33
PCIE_WAKE_n	Wake signal	2.5-V	PIN_BD35
PCIE_PRSNT1n	Hot plug detect	-	-
PCIE_PRSNT2n_x1	Hot plug detect x1 PCIe slot enabled using SW3 dip switch	-	-
PCIE_PRSNT2n_x4	Hot plug detect x4 PCIe slot enabled using SW3 dip switch	-	-
PCIE_PRSNT2n_x8	Hot plug detect x8 PCIe slot enabled using SW3 dip switch	-	-

2.12 SATA

Four Serial ATA (SATA) ports are available on the FPGA development board which are computer bus standard with a primary function of transferring data between the motherboard and mass storage devices (such as hard drives, optical drives, and solid-state disks). Supporting a storage interface is just one of many different applications an FPGA can be used in storage appliances. The Stratix V GX device can bridge different protocols such as bridging simple bus I/Os like PCI Express (PCIe) to SATA or network interfaces such as Gigabit Ethernet (GbE) to SATA. The SATA interface supports SATA 3.0 standard with connection speed of 6 Gbps based on Stratix V GX device with integrated transceivers compliant to SATA electrical standards.

The four Serial ATA (SATA) ports include two available ports for device and two available ports for host capable of implementing SATA solution with a design that consists of both host and target (device side) functions. Figure 2-16 depicts the host and device design examples.

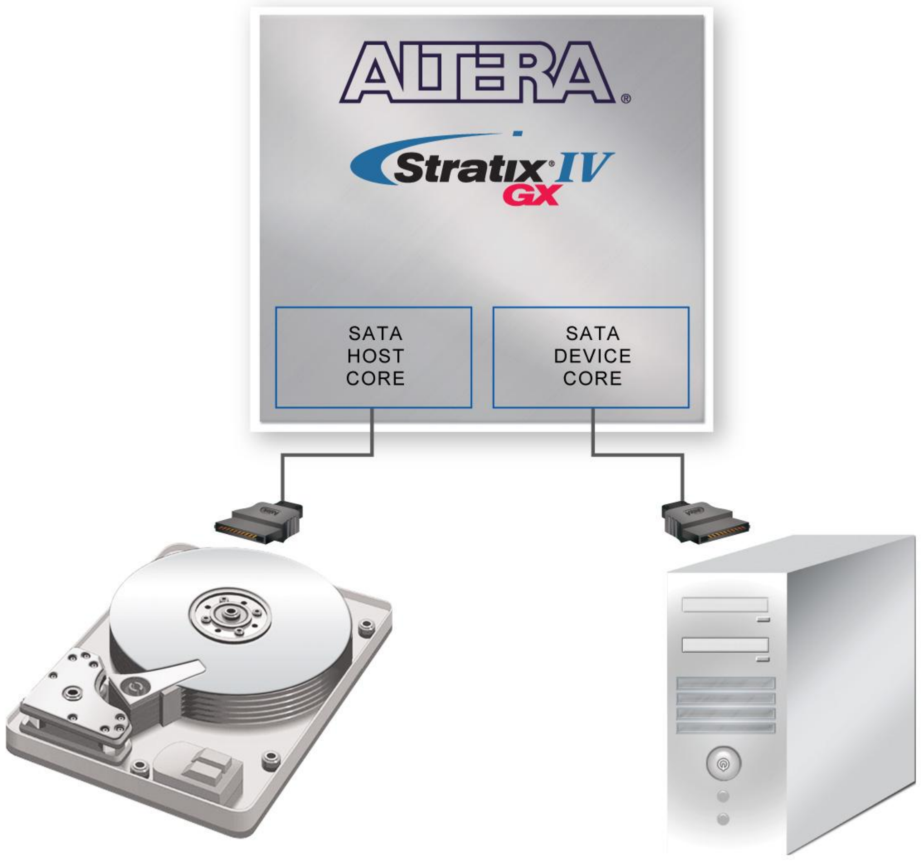


Figure 2-16 PC and storage device connection to the Stratix V GX FPGA

The transmitter and receiver signals of the SATA ports are connected directly to the Stratix V GX transceiver channels to provide SATA IO connectivity to both host and target devices. To verify the functionality of the SATA host/device ports, a connection can be established between the two ports by using a SATA cable as Figure 2-17 depicts the associated signals connected. Figure 2-17 lists the SATA pin assignments, signal names and functions.

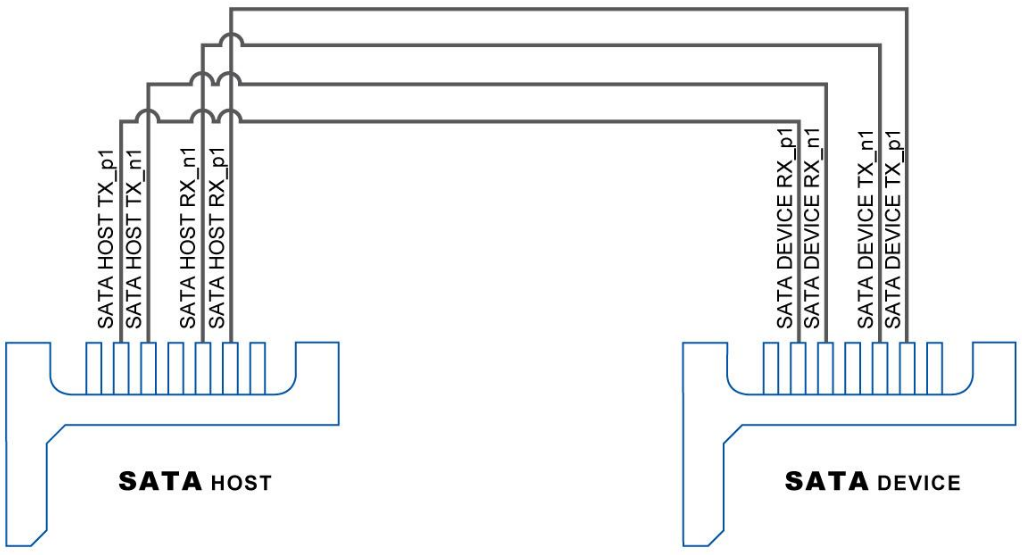


Figure 2-17 Pin connection between SATA connectors

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

Table 2-23 Serial ATA Pin Assignments, Schematic Signal Names, and Functions

Schematic Signal Name	*Description*	*I/O Standard*	*Stratix V GX Pin Number*
Device			
SATA_DEVICE_RX_p0	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_K43
SATA_DEVICE_RX_n0	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_K44
SATA_DEVICE_TX_n0	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_K40
SATA_DEVICE_TX_p0	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_K39
SATA_DEVICE_RX_p1	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_H43
SATA_DEVICE_RX_n1	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_H44
SATA_DEVICE_TX_n1	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_H40
SATA_DEVICE_TX_p1	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_H39
SATA_DEVICE_REFCLK_p	Reference Clock	HCSL	PIN_V39

SATA_DEVICE_REFCLK_n	Reference Clock	HCSL	PIN_V40
Host			
SATA_HOST_TX_p0	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_K6
SATA_HOST_TX_n0	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_K5
SATA_HOST_RX_n0	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_K1
SATA_HOST_RX_p0	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_K2
SATA_HOST_TX_p1	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_H6
SATA_HOST_TX_n1	Differential transmit data output before DC blocking capacitor	1.4-V PCML	PIN_H5
SATA_HOST_RX_n1	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_H1
SATA_HOST_RX_p1	Differential receive data input after DC blocking capacitor	1.4-V PCML	PIN_H2
SATA_HOST_REFCLK_ p	Reference Clock	HCSL	PIN_V6
SATA_HOST_REFCLK_ n	Reference Clock	HCSL	PIN_V5

Chapter 3

System Builder

This chapter describes how users can create a custom design project on the FPGA board by using the Software Tools – System Builder.

3.1 Introduction

The System Builder is a Windows based software utility, designed to assist users to create a Quartus II project for the FPGA board within minutes. The generated Quartus II project files 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 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.

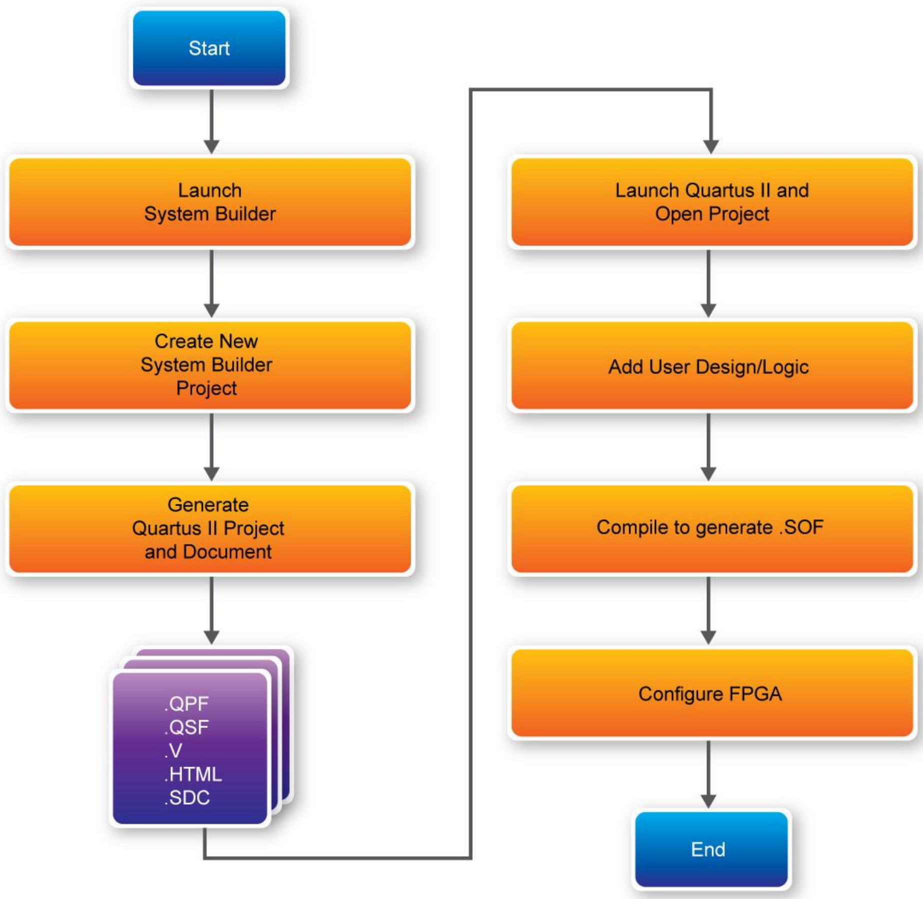

Start
Launch System Builder
Create New System Builder Project
Generate Quartus II Project and Document
.QPF
.QSF
.V
.HTML
.SDC
Launch Quartus II and Open Project
Add User Design/Logic
Compile to generate .SOF
Configure FPGA
End

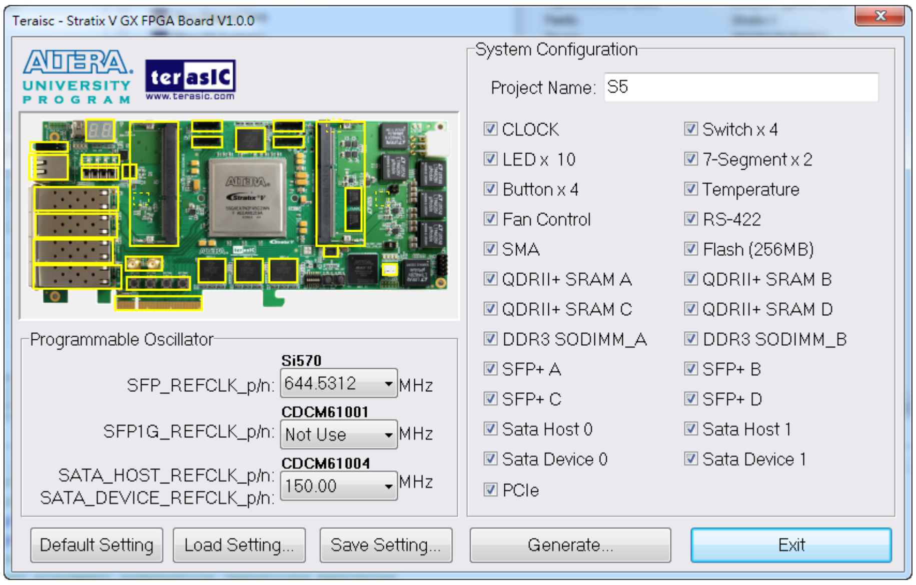
Figure 3-1 The general design flow of building a design

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 in the directory: "**Tools\SystemBuilder**" in the System CD. Users can copy the whole folder to a host computer without installing the utility. Before using the System Builder, execute the **SystemBuilder.exe** on the host computer as appears in **Figure 3-2**.


Terasic - Stratix V GX FPGA Board V1.0.0

System Configuration

Project Name: S5

☑ CLOCK ☑ Switch x 4
☑ LED x 10 ☑ 7-Segment x 2
☑ Button x 4 ☑ Temperature
☑ Fan Control ☑ RS-422
☑ SMA ☑ Flash (256MB)
☑ QDRII+ SRAM A ☑ QDRII+ SRAM B
☑ QDRII+ SRAM C ☑ QDRII+ SRAM D
☑ DDR3 SODIMM_A ☑ DDR3 SODIMM_B
☑ SFP+ A ☑ SFP+ B
☑ SFP+ C ☑ SFP+ D
☑ Sata Host 0 ☑ Sata Host 1
☑ Sata Device 0 ☑ Sata Device 1
☑ PCIe

Programmable Oscillator

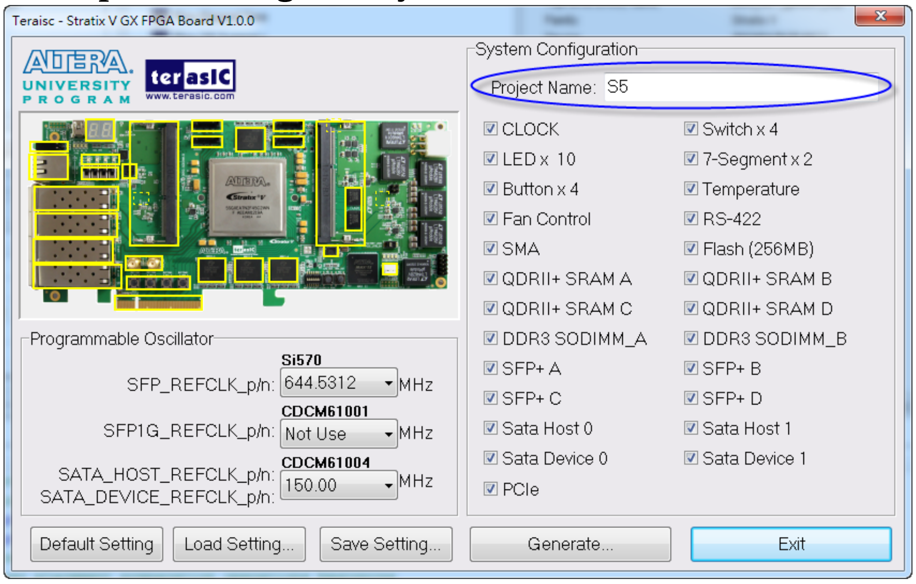
Si570
SFP_REFCLK_p/n: 644.5312 MHz
CDCM61001
SFP1G_REFCLK_p/n: Not Use MHz
CDCM61004
SATA_HOST_REFCLK_p/n:
SATA_DEVICE_REFCLK_p/n: 150.00 MHz

Figure 3-2 The System Builder window

■ Select Board Type and Input Project Name

Select the target board type and input project name as show in **Figure 3-3**.

■ Project Name: Specify the project name as it is automatically assigned to the name of the top-level design entity.


Terasic - Stratix V GX FPGA Board V1.0.0

System Configuration

Project Name: S5

☑ CLOCK ☑ Switch x 4
☑ LED x 10 ☑ 7-Segment x 2
☑ Button x 4 ☑ Temperature
☑ Fan Control ☑ RS-422
☑ SMA ☑ Flash (256MB)
☑ QDRII+ SRAM A ☑ QDRII+ SRAM B
☑ QDRII+ SRAM C ☑ QDRII+ SRAM D
☑ DDR3 SODIMM_A ☑ DDR3 SODIMM_B
☑ SFP+ A ☑ SFP+ B
☑ SFP+ C ☑ SFP+ D
☑ Sata Host 0 ☑ Sata Host 1
☑ Sata Device 0 ☑ Sata Device 1
☑ PCIe

Programmable Oscillator

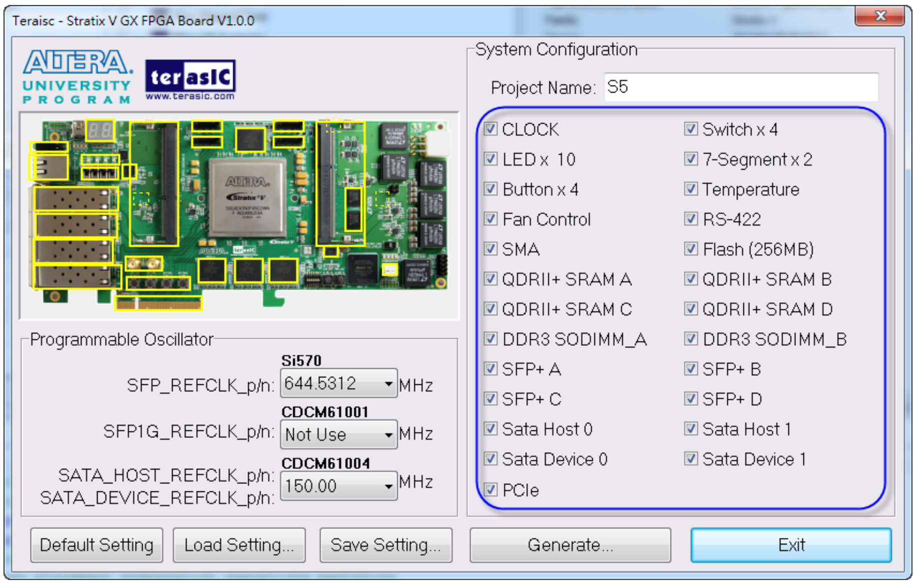
Si570
SFP_REFCLK_p/n: 644.5312 MHz
CDCM61001
SFP1G_REFCLK_p/n: Not Use MHz
CDCM61004
SATA_HOST_REFCLK_p/n:
SATA_DEVICE_REFCLK_p/n: 150.00 MHz

Figure 3-3 The Quartus Project Name

■ System Configuration

Under System Configuration users are given the flexibility of enabling their choice of components on the FPGA as shown in **Figure 3-4**. Each component of the FPGA board is listed where users can enable or disable a component according to their design by simply marking a check or removing the check in the field provided. If the component is enabled, the System Builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and I/O standards.

Note: The pin assignments for some components (e.g. DDR3 and SFP+) require associated controller codes in the Quartus project otherwise Quartus will result in compilation errors. Therefore, do not select them if they are not necessary in your design. To use the DDR3 controller, please refer to the DDR3 SDRAM demonstration in Chapter 6.


Terasic - Stratix V GX FPGA Board V1.0.0

System Configuration

Project Name: S5

☑ CLOCK ☑ Switch x 4
☑ LED x 10 ☑ 7-Segment x 2
☑ Button x 4 ☑ Temperature
☑ Fan Control ☑ RS-422
☑ SMA ☑ Flash (256MB)
☑ QDRII+ SRAM A ☑ QDRII+ SRAM B
☑ QDRII+ SRAM C ☑ QDRII+ SRAM D
☑ DDR3 SODIMM_A ☑ DDR3 SODIMM_B
☑ SFP+ A ☑ SFP+ B
☑ SFP+ C ☑ SFP+ D
☑ Sata Host 0 ☑ Sata Host 1
☑ Sata Device 0 ☑ Sata Device 1
☑ PCIe

Programmable Oscillator

Si570
SFP_REFCLK_p/n: 644.5312 MHz
CDCM61001
SFP1G_REFCLK_p/n: Not Use MHz
CDCM61004
SATA_HOST_REFCLK_p/n:
SATA_DEVICE_REFCLK_p/n: 150.00 MHz

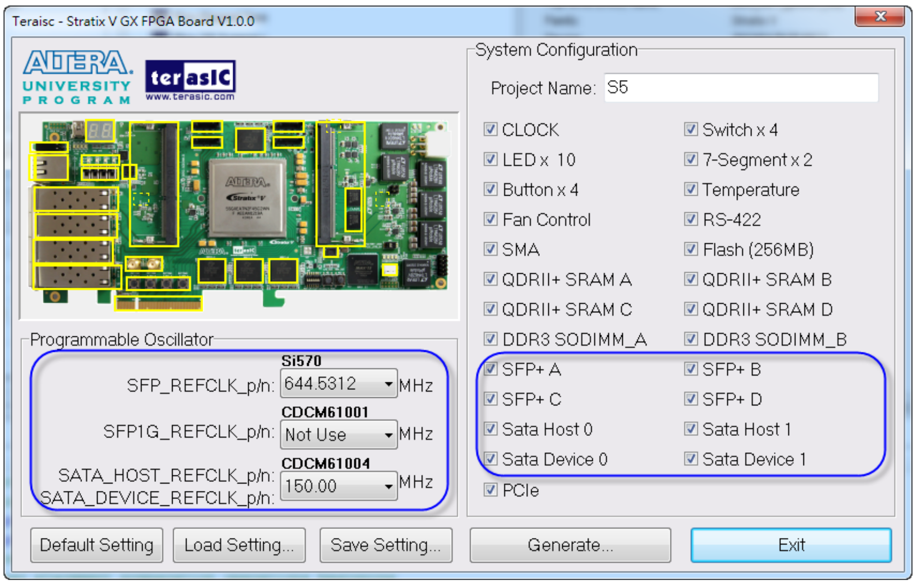
Figure 3-4 System Configuration Group

■ Programmable Oscillator

There are two external oscillators on-board that provide reference clocks for the following signals SFP_REFCLK, SFP1G_REFCLK, SATA_HOST_REFCLK and SATA_DEVICE_REFCLK. To use these oscillators, users can select the desired frequency on the Programmable Oscillator group, as shown in **Figure 3-5**. SPF+ or SATA should 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.


Terasic - Stratix V GX FPGA Board V1.0.0

System Configuration

Project Name: S5

✓ CLOCK	✓ Switch x 4
✓ LED x 10	✓ 7-Segment x 2
✓ Button x 4	✓ Temperature
✓ Fan Control	✓ RS-422
✓ SMA	✓ Flash (256MB)
✓ QDRII+ SRAM A	✓ QDRII+ SRAM B
✓ QDRII+ SRAM C	✓ QDRII+ SRAM D
✓ DDR3 SODIMM_A	✓ DDR3 SODIMM_B
✓ SFP+ A	✓ SFP+ B
✓ SFP+ C	✓ SFP+ D
✓ Sata Host 0	✓ Sata Host 1
✓ Sata Device 0	✓ Sata Device 1
✓ PCIe	

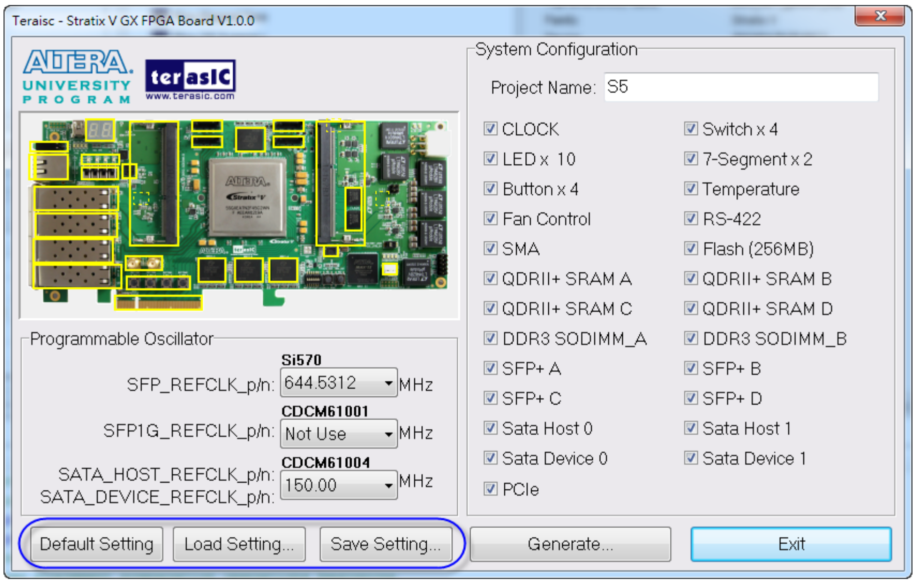
Programmable Oscillator

SFP_REFCLK_p/n:	Si570	644.5312	MHz
SFP1G_REFCLK_p/n:	CDCM61001	Not Use	MHz
SATA_HOST_REFCLK_p/n: SATA_DEVICE_REFCLK_p/n:	CDCM61004	150.00	MHz

Figure 3-5 External Programmable Oscillators

■ Project Setting Management

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


Terasic - Stratix V GX FPGA Board V1.0.0

System Configuration

Project Name: S5

✓ CLOCK	✓ Switch x 4
✓ LED x 10	✓ 7-Segment x 2
✓ Button x 4	✓ Temperature
✓ Fan Control	✓ RS-422
✓ SMA	✓ Flash (256MB)
✓ QDRII+ SRAM A	✓ QDRII+ SRAM B
✓ QDRII+ SRAM C	✓ QDRII+ SRAM D
✓ DDR3 SODIMM_A	✓ DDR3 SODIMM_B
✓ SFP+ A	✓ SFP+ B
✓ SFP+ C	✓ SFP+ D
✓ Sata Host 0	✓ Sata Host 1
✓ Sata Device 0	✓ Sata Device 1
✓ PCIe	

Programmable Oscillator

SFP_REFCLK_p/n:	Si570	644.5312	MHz
SFP1G_REFCLK_p/n:	CDCM61001	Not Use	MHz
SATA_HOST_REFCLK_p/n: SATA_DEVICE_REFCLK_p/n:	CDCM61004	150.00	MHz

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.

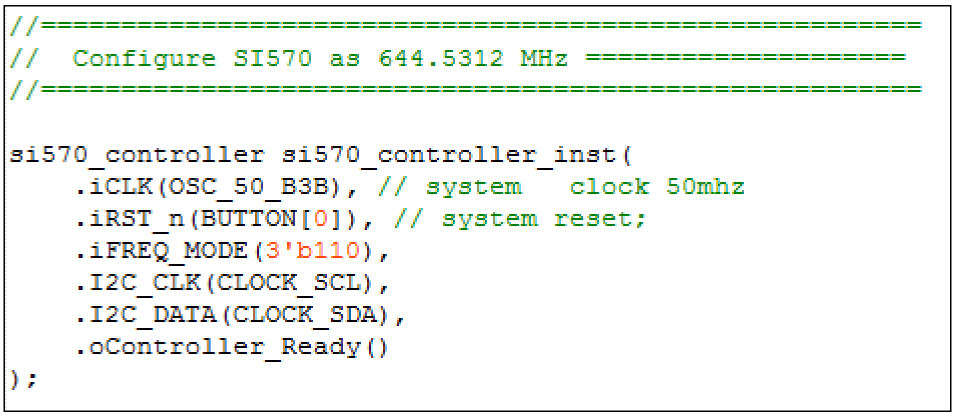
Table 3-1 The files generated by System Builder

No.	***Filename***	***Description***
1	**<Project name>.v**	**Top level Verilog file for Quartus II**
2	**Si570_controller.v(*)**	**Si570 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 Si570 Controller includes seven files: Si570_controller.v, initial_config.v, clock_divider.v, edge_detector.v, i2c_reg_controller.v, i2c_controller.v and i2c_bus_controller.v.

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

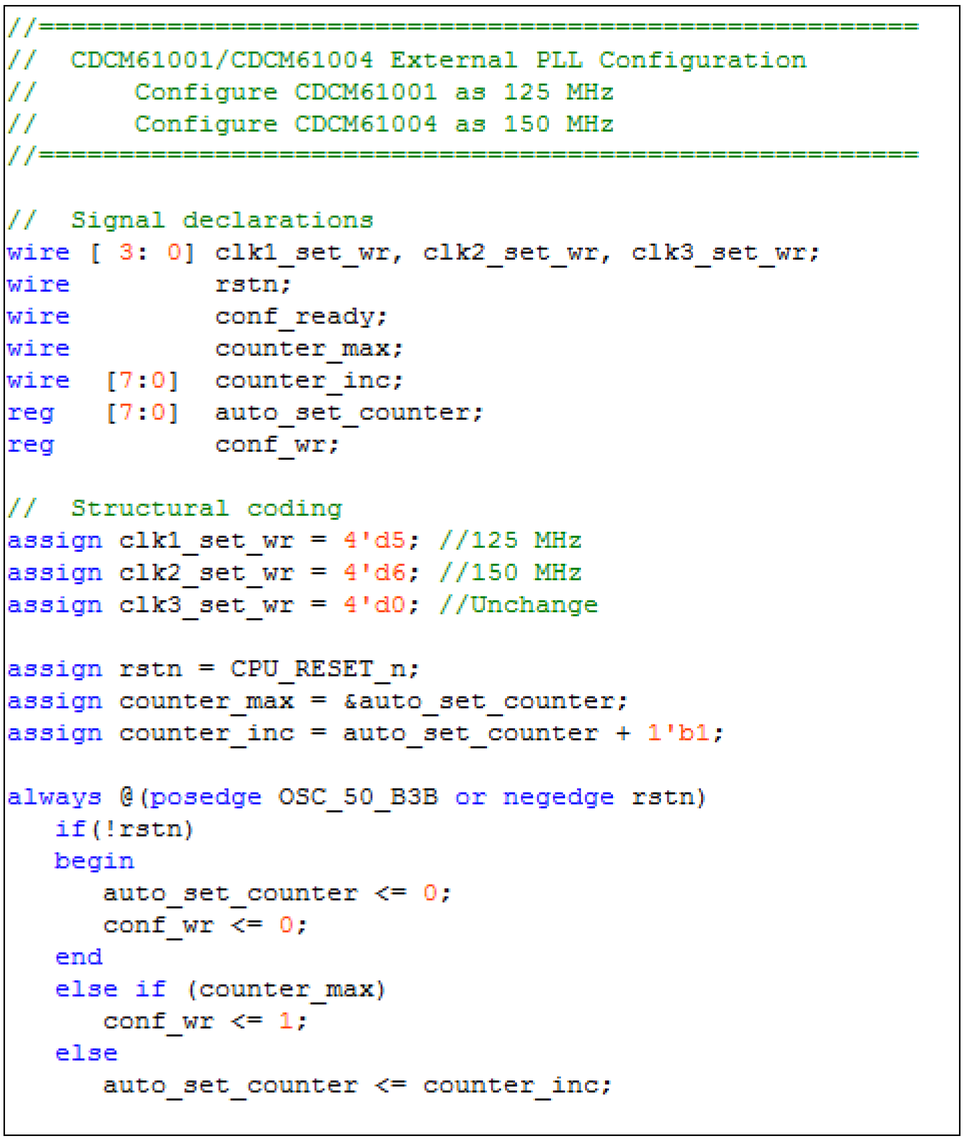
For Si570, the Controller will be instantiated in the Quartus II top-level file as listed below:



```
//=========================================================
//  Configure SI570 as 644.5312 MHz =====================
//=========================================================

si570_controller si570_controller_inst(
    .iCLK(OSC_50_B3B), // system   clock 50mhz
    .iRST_n(BUTTON[0]), // system reset;
    .iFREQ_MODE(3'b110),
    .I2C_CLK(CLOCK_SCL),
    .I2C_DATA(CLOCK_SDA),
    .oController_Ready()
);
```

For CDCM61001 and CDCM61004, the Controller will be instantiated in the Quartus II top-level file as listed below:



```
//=========================================================
//  CDCM61001/CDCM61004 External PLL Configuration
//      Configure CDCM61001 as 125 MHz
//      Configure CDCM61004 as 150 MHz
//=========================================================

//  Signal declarations
wire [ 3: 0] clk1_set_wr, clk2_set_wr, clk3_set_wr;
wire         rstn;
wire         conf_ready;
wire         counter_max;
wire  [7:0]  counter_inc;
reg   [7:0]  auto_set_counter;
reg          conf_wr;

//  Structural coding
assign clk1_set_wr = 4'd5; //125 MHz
assign clk2_set_wr = 4'd6; //150 MHz
assign clk3_set_wr = 4'd0; //Unchange

assign rstn = CPU_RESET_n;
assign counter_max = &auto_set_counter;
assign counter_inc = auto_set_counter + 1'b1;

always @(posedge OSC_50_B3B or negedge rstn)
   if(!rstn)
   begin
      auto_set_counter <= 0;
      conf_wr <= 0;
   end
   else if (counter_max)
      conf_wr <= 1;
   else
      auto_set_counter <= counter_inc;
```

```
ext_pll_ctrl ext_pll_ctrl_Inst(
   .osc_50(OSC_50_B3B), //50MHZ
   .rstn(rstn),

   // device 1 (SFP1G_REFCLK_p)
   .clk1_set_wr(clk1_set_wr),
   .clk1_set_rd(),

   // device 2 (SATA_HOST_REFCLK_p/SATA_DEVICE_REFCLK_p)
   .clk2_set_wr(clk2_set_wr),
   .clk2_set_rd(),

   // device 3 (reserved)
   .clk3_set_wr(clk3_set_wr),
   .clk3_set_rd(),

   // setting trigger
   .conf_wr(conf_wr), // 1T 50MHz
   .conf_rd(), // 1T 50MHz

   // status
   .conf_ready(conf_ready),

   // 2-wire interface
   .max_sclk(PLL_SCL),
   .max_sdat(PLL_SDA)

);
```

If 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 flash memory into the FPGA on power up. This chapter will describe how to use Altera Quartus II Programmer Tool to program the common flash interface (CFI) flash memory device on the FPGA board. The Stratix V GX FPGA development board ships with the CFI flash device preprogrammed with a default factory FPGA configuration for running the Parallel Flash Loader design example.

4.1 CFI Flash Memory Map

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

Table 4-1 Flash Memory Map (Byte Address)

Block Description	*Size(KB)*	*Address Range*
PFL option bits	**64**	**0x00030000 – 0x0003FFFF**
Factory hardware	**33,280**	**0x00040000 – 0x020BFFFF**
User hardware	**33,280**	**0x020C0000 – 0x0413FFFF**
Factory software	**8,192**	**0x04140000 – 0x0493FFFF**
User software and data	**187,136**	**0x04940000 – 0x0FFFFFFF**

For user application, user hardware must be stored with start address **0x020C0000**, and the user's software is suggested to be stored with start address **0x04940000**. The NIOS II EDS tool **nios-2-flash-programmer** is used for programming the flash. Before programming, users need to translate their Quartus .sof and NIOS II .elf files into the .flash which is used by the

nios-2-flash-programmer. For .sof to .flash translation, NIOS II EDS tool **sof2flsh** can be used. For the .elf to .flash translation, NIOS II EDS tool **elf2flash** can be used. For convenience, the System CD contains a batch file for file translation and flash programming with users given .sof and .elf file.

4.2 FPGA Configure Operation

Here is the procedure to enable FPGA configuration from Flash:

1. Please make sure the FPGA configuration data has been stored in the CFI flash.
2. Set the FPGA configuration mode to FPPx32 mode by setting SW6 MSEL[0:4] as 00010
3. Specify the configuration of the FPGA using the default Factory Configuration or User Configuration by setting SW5 according to **Figure 4-1.**
4. Power on the FPGA board or press MAX_RST button if board is already powered on
5. When configuration is completed, the green Configure Done LED will light. If there is error, the red Configure Error LED will light.

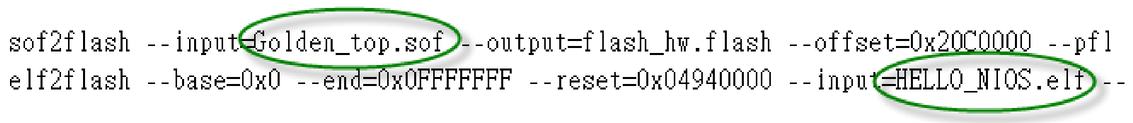
4.3 Flash Programming with Users Design

Users can program the flash memory device so that a custom design loads from flash memory into the FPGA on power up. For convenience, the translation and programming batch files are available on the Demonstrations/Hello/flash_programming_batch folder in the System CD. There folder contains five files as shown in **Table 4-2**

Table 4-2 Flash Memory Map (Byte Address)

Files Name	***Description***
S5_PFL.sof	**Parallel Flash Loader Design**
flash_program_ub2.bat	**Top batch file to download S5_PFL.sof and launch batch flash_program_bashrc_ub2**
flash_program_bashrc_ub2	**Translate .sof and .elf into .flash and programming flash with the generated .flash file**
Golden_top.sof	**Hardware design file for Hello Demo**
HELLO_NIOS.elf	**Software design file for Hello Demo**

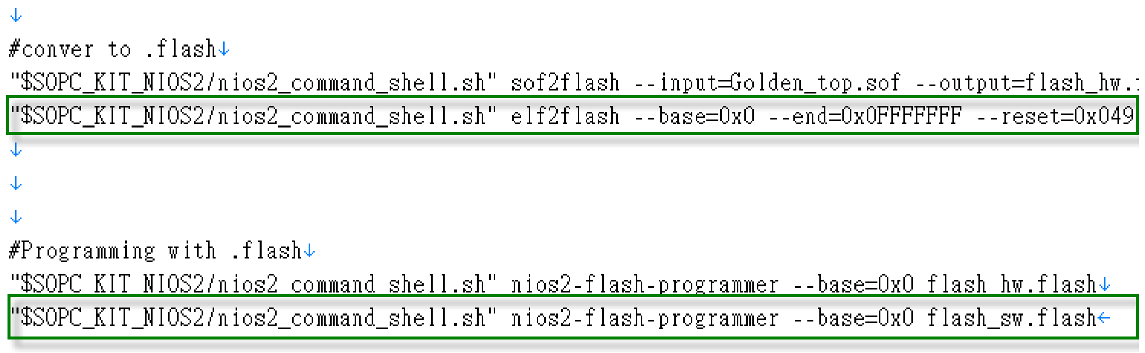
To apply the batch file to users' .sof and .elf file, users can change the .sof and .elf filename in the **flash_program_bashrc_ub2** file as shown in Figure 4-2.



```
sof2flash --input=Golden_top.sof --output=flash_hw.flash --offset=0x20C0000 --pfl
elf2flash --base=0x0 --end=0x0FFFFFFF --reset=0x04940000 --input=HELLO_NIOS.elf --
```

Figure 4-1 Change to usrs' .sof and .elf filename

If your design does not contain a NIOS II processor, users can add "#" to comment (disable) the elf2flash and nios-flash-programmer commands in the **flash_program_bashrc_ub2** file as shown in Figure 4-2.

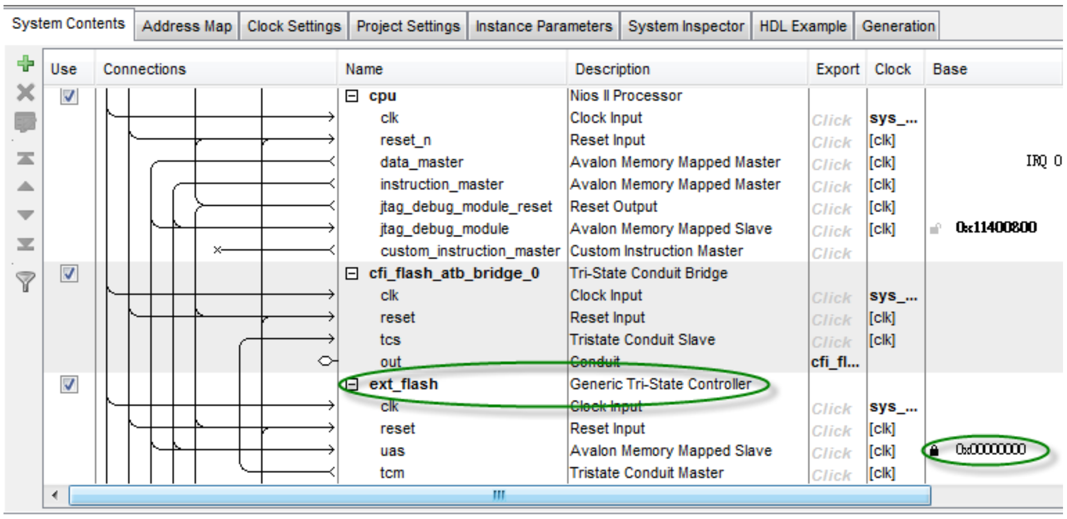


```
↓
#conver to .flash↓
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" sof2flash --input=Golden_top.sof --output=flash_hw.:
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" elf2flash --base=0x0 --end=0x0FFFFFFF --reset=0x049
↓
↓
↓
#Programming with .flash↓
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0 flash_hw.flash↓
"$SOPC_KIT_NIOS2/nios2_command_shell.sh" nios2-flash-programmer --base=0x0 flash_sw.flash←
```

Figure 4-2 Disable .elf translation and programming

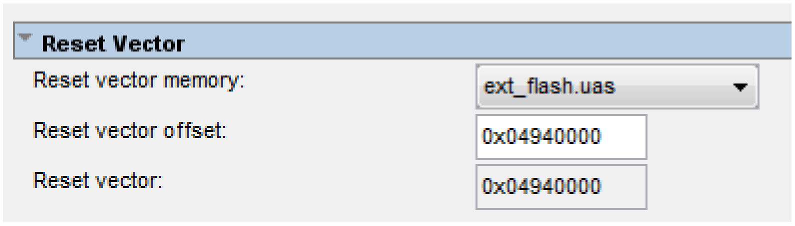
If your design includes a NIOS II processor and the NIOS II program is stored on external memory, users must to perform following items so the NIOS II program can be boot from flash successfully:

1. QSYS should include a Flash controller for the CFI Flash on the development board. Please ensure that the base address of the controller is 0x00, as shown in Figure 4-3.
2. In NIOS II processor options, select FLASH as reset vector memory and specify 0x04940000 as reset vector, as shown in Figure 4-4.



Use	Connections	Name	Description	Export	Clock	Base
✓		cpu	Nios II Processor			
		clk	Clock Input	Click	sys_...	
		reset_n	Reset Input	Click	[clk]	
		data_master	Avalon Memory Mapped Master	Click	[clk]	IRQ 0
		instruction_master	Avalon Memory Mapped Master	Click	[clk]	
		jtag_debug_module_reset	Reset Output	Click	[clk]	
		jtag_debug_module	Avalon Memory Mapped Slave	Click	[clk]	0x11400800
		custom_instruction_master	Custom Instruction Master	Click		
✓		cfi_flash_atb_bridge_0	Tri-State Conduit Bridge			
		clk	Clock Input	Click	sys_...	
		reset	Reset Input	Click	[clk]	
		tcs	Tristate Conduit Slave	Click	[clk]	
		out	Conduit	cfi_fl...		
✓		ext_flash	Generic Tri-State Controller			
		clk	Clock Input	Click	sys_...	
		reset	Reset Input	Click	[clk]	
		uas	Avalon Memory Mapped Slave	Click	[clk]	0x00000000
		tcm	Tristate Conduit Master	Click	[clk]	

Figure 4-3 Flash Controller Settings in QSYS



Reset Vector

Reset vector memory:	ext_flash.uas
Reset vector offset:	0x04940000
Reset vector:	0x04940000

Figure 4-4 Reset Vector Settings for NIOS II Processor

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

Demonstrations/ Hello

4.4 Restore Factory Settings

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

1. Make sure the Nios II EDS and USB-Blaster II driver are installed.
2. Make sure the FPGA board and PC are connected with an UBS Cable.

3. Power on the FPGA board.
4. Copy the "Demonstrations/PFL/flash_programming_batch" folder under the CD to your PC's local drive.
5. Execute the batch file flash_program_ub2.bat to start flash programming.
6. Power off the FPGA Board.
7. Set FPGA configure mode as FPPx32 Mode by setting SW6 MSEL[0:4] to 00010.
8. Specify configuration of the FPGA to Factory Hardware by setting the FACTORY_LOAD dip in SW5 to the '1' position.
9. Power on the FPGA Board, and the Configure Done LED should light.

Except for programming the Flash with the default code PFL, the batch file also writes PFL (Parallel Flash Loader) Option Bits data into the address 0x30000. The option bits data specifies 0x20C0000 as start address of your hardware design.

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

Chapter 5

Programmable Oscillator

This chapter describes how to program the two programmable oscillators Si570 and CDCM61004 on the FPGA board. Also, RTL code based and Nios based reference design are explained in the chapter. The source codes of these examples are all available on the FPGA System CD.

5.1 Overview

This section describes how to program Si570- and CDCM61004. For detail programming information, please refer to their datasheets which are available on the FPGA System CD.

■ Si570

The Si570 utilizes Silicon Laboratories advanced DSPLL® circuitry to provide a low-jitter clock at any frequency. The Si570 are user-programmable to any output frequency from 10 to 945 MHz and select frequencies to 1400 MHz with < 1ppb resolution. The device is programmed via an I2C serial interface. The differential clock output of the Si570 directly connects to dedicated reference clock input of the Stratix V GX transceiver for SFP+ channels. Many applications can be implemented using this function. For example, the 10G Ethernet application can be designed onto this board by feeding a necessary clock frequency of 644.53125MHz or 322.265625MHz from the Si570.

Figure 5-1 shows the block diagram of Si570 device. Users can modify the value of the three registers RFREQ, HS_DIV, and N1 to generate the desired output frequency.

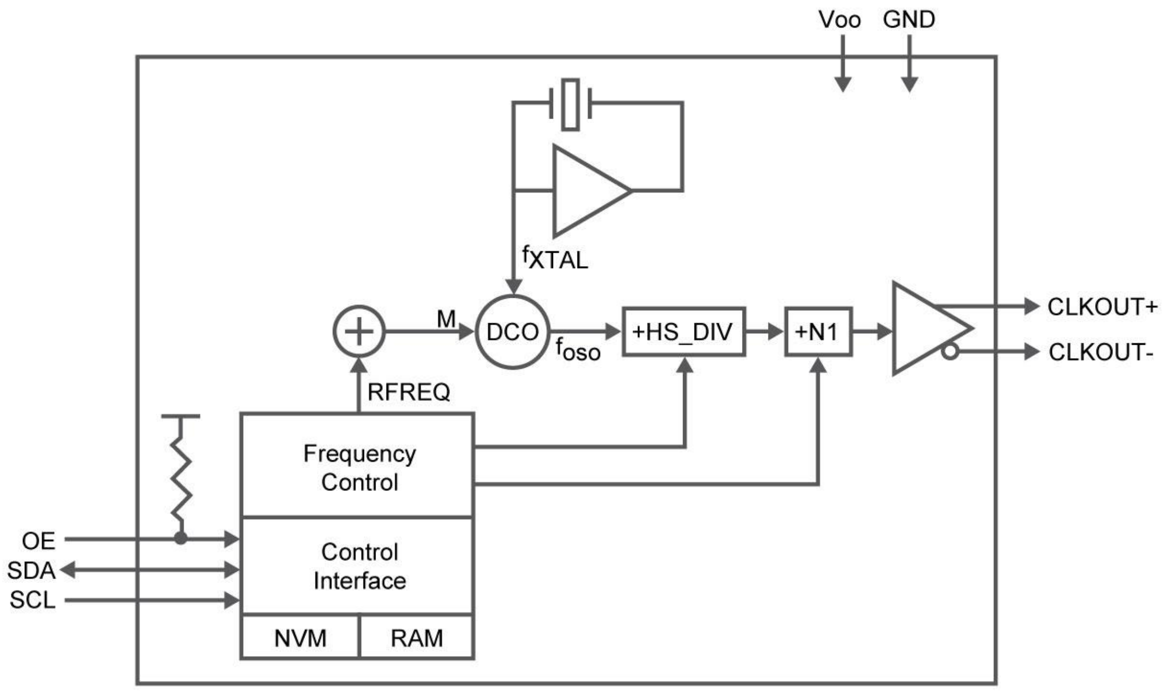


Figure 5-1 Si570 Block diagram

The output frequency is calculated using the following equation:

$$\boxed{f_{out} = \frac{f_{DCO}}{\text{Output Dividers}} = \frac{f_{XTAL} \times RFREQ}{HSDIV \times N1}}$$

When Si570 is powered on, the default output frequency is 100 MHz. Users can program the output frequency through the I2C interface using the following procedure.

6. Freeze the DCO (bit 4 of Register 137).
7. Write the new frequency configuration (RFREQ,HSDIV, and N1) to Register 7 – 12.
8. Unfreeze the DCO and assert the NewFreq bit (bit 6 of Register 135).

The I2C address of Si570 is zero and it supports fast mode operation whose transfer rate is up to 400 kbps. Table 5-1 shows the register table for Si570.

Table 5-1 Si570 Register Table

Register	*Name*	*Bit 7*	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
7	High Speed/N1 Dividers	HS_DIV[2:0]			N1[6:2]				
8	Reference Frequency	N1[1:0]		RFREQ[37:32]					
9	Reference Frequency	RFREQ[31:24]							
10	Reference Frequency	RFREQ[23:16]							
11	Reference Frequency	RFREQ[15:8]							
12	Reference Frequency	RFREQ[7:0]							
135	Reference Frequency	RST_REG	NewFreq	Freeze M	Freeze VCADC				RECALL
137	Reference Frequency				Freeze DCO				

Table 5-2 lists the register settings for some common used frequency.

Table 5-2 Si570 Register Table

Output Frequency (MHz)	*HS_DIV*	*HS_DIV Register Setting*	*NI*	*NI Register Setting*	*REF_CLK Register Setting*
100	9	101	6	0000101	02F40135A9(hex)
125	11	111	4	0000011	0302013B65(hex)
156.25	9	101	4	0000011	0313814290(hex)
250	11	111	2	0000001	0302013B65(hex)
312.5	9	101	2	0000001	0313814290(hex)
322.265625	4	000	4	0000011	02D1E127AF(hex)
644.53125	4	000	2	0000001	02D1E127AF(hex)

■ CDCM61004

The FPGA board includes another programmable PLL CDCM61004. The CDCM61004 supports output frequency range from 43.75 MHz to 683.264 MHz. It provides a parallel interface for selecting a desired output frequency. The Stratix V GX FPGA's IOs connect to the interface directly. The differential clock outputs of the CDCM61004 are designed for SFP+ and SATA applications on FPGA board.

When CDCM61004 is powered on, the default output frequency is 100 MHZ. Users can change the output frequency by the following control pins:

1. PR0 and PR1
2. OD0, OD1, and OD2
3. RSTN
4. CE
5. OS0 and OS1

The following table lists the frequency which CDCM61004 can generate in the FPGA board.

PRESCALLR DIVIDER	FEEDBACK DIVIDER	OUTPUT DEVIDER	OUTPUT FREQUENCY(MHz)	APPLICATION
4	20	8	62.5	GigE
3	24	8	75	SATA
3	24	6	100	PCI Express
4	20	4	125	GigE
3	24	4	150	SATA
3	25	4	156.25	10 GigE
5	15	2	187.5	12 GigE
3	24	3	200	PCI Express
4	20	2	250	GigE
4	20	2	312.5	XGMII
3	25	1	625	10 GigE

The both values of PRESCALER DIVIDER and FEEDBACK DIVIDER can be specified by the PR0 and PR1 control pins according to the following table:

CONTROL INPUTS		PRESCALER DIVIDER	FEEDBACK DIVIDER
PR1	PR0		
0	0	3	24
0	1	5	15
1	0	3	25
1	1	4	20

The value of OUTPUT DIVIDER can be specified by the OD0, OD1 and OD2 control pins according to the following table:

CONTROL INPUTS			OUTPUT DIVIDER
OD2	OD1	OD0	
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	Reserved
1	0	1	6
1	1	0	Reserved
1	1	1	8

After specifying the desired output frequency in the parallel interface, developers must assert the output enable pin CE and control the RSTN pin to generate a rising signal to start the PLL Recalibration process. In the FPGA board, the required output type is LVDS, so always set OS0 and SO1 to 0 and 1, respectively.

5.2 Si570 Example by RTL

In this section we will demonstrate how to use the Terasic Si570 Controller implemented in Verilog to control the Si570 programmable oscillator on the FPGA board. This controller IP can configure the Si570 to output a clock with a specific frequency via I2C interface. For demonstration, the output clock is used to implement a counter where the MSB is used to drive an LED, so the user can get the result from the frequency of the LED blinking. We will also introduce the port declarations and associated parameter settings of this IP. **Figure 5-2** shows the block diagram of this demonstration.

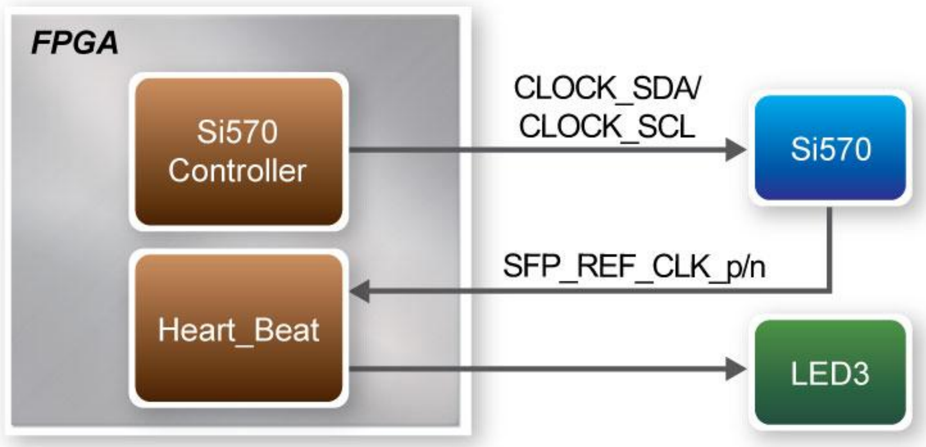


Figure 5-2 Block Diagram of this Demonstration

■ Block Diagrams of Si570 Controller IP

The block diagram of the Si570 controller is shown on **Figure 5-3.** Shown here are four blocks named i2c_reg_controller, i2c_bus_controller, clock_divider and initial_config in Si570 controller IP. Firstly, the i2c_reg_controller will generate an associated Si570 register value for the i2c_bus_controller based on user-desired frequency. Once i2c_bus_controller receives this data, it will transfer these settings to Si570 via serial clock and data bus using I2C protocol. The registers in Si570 will be configured and output the user-desired frequency.

Secondly, the clock_divider block will divide system clock (50 MHz) into 97.6 KHz which is used as I2C interface clock of i2c_bus_controller. Finally, the initial_config block will generate a control signal to drive i2c_reg_controller which allows the Si570 controller to configure Si570 based on default settings.

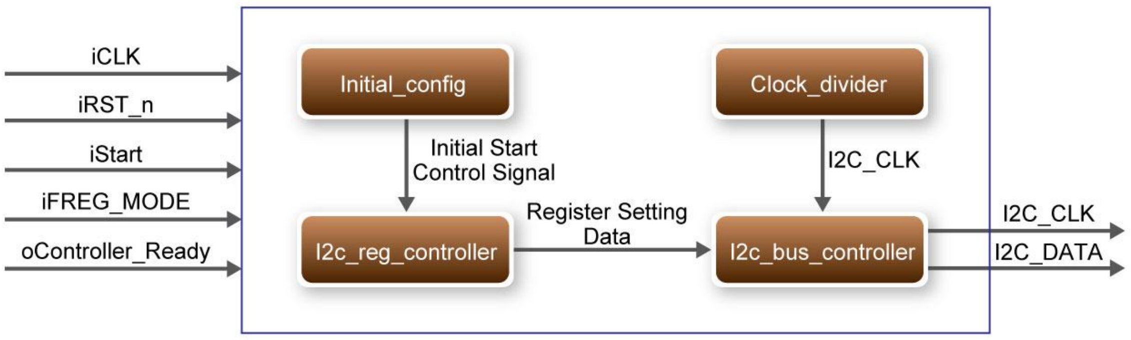


Figure 5-3 Block Diagram of Si570 Controller IP

■ Using Si570 Controller IP

Table 5-3 lists the instruction ports of Si570 Controller IP

Table 5-3 Si570 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）
iFREG_MODE	input	Setting Si570 Output Frequency Value
oController_Ready	output	Si570 Configuration status (0: Configuration in Progress, 1: Configuration Complete)
I2C_DATA	inout	I2C Serial Data to/from Si570
I2C_CLK	output	I2C Serial Clock to Si570

To use the Si570 Controller, the first thing users need to determine is the desired output frequency in advance. The Si570 controller provides six optional clock frequencies. These options can be set through an input port named "iFREG_MODE" in Si570 controller. The specified settings with corresponding frequencies are listed in Table 5-4. For example, setting "iFREG_MODE" as 3'b110 will configure Si570 to output 655.53 MHz clock.

Table 5-4 Si570 Controller Frequency Setting

iFREG MODE Setting	Si570 Clock Frequency(MHz)
3'b000	100
3'b001	125
3'b010	156.25
3'b011	250
3'b100	312.25
3'b101	322.26
3'b110	644.53125
3'b111	100

When the output clock frequency is decided, the next thing users need to do is to enable the controller to configure Si570. Before sending enable signal to Si570 controller, users need to monitor an output port named "oController_Ready". This port indicates if Si570 controller is ready to be configured or not. If it is ready, logic high will be outputted and the user needs to send a high level logic to "iStart" port to enable the Si570 Controller as shown in Figure 5-4. During Si570 configuring, the logic level of "oController_Ready" is low; when it rises to high again that means the user can configure another frequency value.

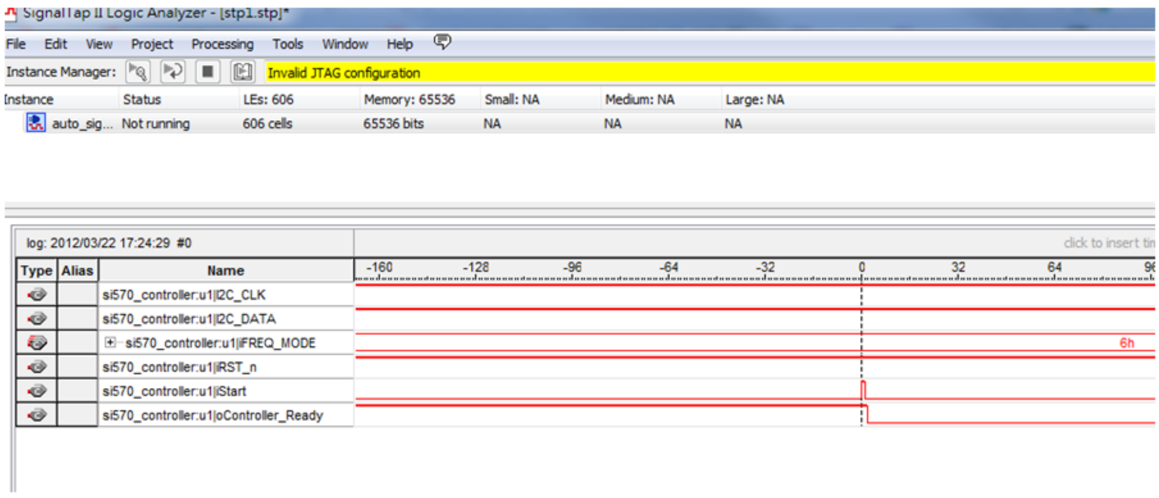

SignalTap II Logic Analyzer - [stp1.stp]*
File Edit View Project Processing Tools Window Help
Instance Manager:
Invalid JTAG configuration
Instance Status LEs: 606 Memory: 65536 Small: NA Medium: NA Large: NA
auto_sig... Not running 606 cells 65536 bits NA NA NA
log: 2012/03/22 17:24:29 #0
click to insert tin
Type Alias Name
-160 -128 -96 -64 -32 0 32 64 96
si570_controller:u1|I2C_CLK
si570_controller:u1|I2C_DATA
si570_controller:u1|iFREQ_MODE
6h
si570_controller:u1|iRST_n
si570_controller:u1|iStart
si570_controller:u1|oController_Ready

Figure 5-4 Timing Waveform of Si570 Controller

■ Modify Clock Parameter For Your Own Frequency

If all the six clock frequencies are not desired, you can perform the following steps to modify Si570 controller.

1. Open i2c_reg_controller.v

2. Locate the Verilog code shown below:

```
always @(*)
    begin
        case(iFREQ_MODE)
            3'h0 :      //100Mhz
                begin
                    new_hs_div = 4'b0101 ;
                    new_n1 = 8'b0000_1010 ;
                    fdco = 28'h004_E200 ;
                end
            3'h1 :      //125Mhz
                begin
                    new_hs_div = 4'b0101 ;
                    new_n1 = 8'b0000_1000 ;
                    fdco = 28'h004_E200 ;
                end
    3'h2 :      //156.25Mhz
      begin
        new_hs_div = 4'b0100 ;
        new_n1 = 8'b0000_1000 ;
        fdco = 28'h004_E200 ;
      end
    3'h3 :      //250Mhz
      begin
        new_hs_div = 4'b0101 ;
        new_n1 = 8'b0000_0100 ;
        fdco = 28'h004_E200 ;
      end
    3'h4 :      //312.5Mhz
      begin
        new_hs_div = 4'b0100 ;
        new_n1 = 8'b0000_0100 ;
        fdco = 28'h004_E200 ;
      end
    3'h5 :      //322.265625Mhz
      begin
        new_hs_div = 4'b0100 ;
        new_n1 = 8'b0000_0100 ;
        fdco = 28'h005_0910 ;
      end
    3'h6 :      //644.53125Mhz
      begin
        new_hs_div = 4'b0100 ;
        new_n1 = 8'b0000_0010 ;
        fdco = 28'h005_0910 ;
      end
    default :      //100Mhz
      begin
        new_hs_div = 4'b0101 ;
        new_n1 = 8'b0000_1010 ;
        fdco = 28'h004_E200 ;
      end
  endcase
```

```
    end
```

Users can get a desired frequency output from si570 by modifying these three parameters : **new_hs_div** ,**new_n1** and **fdco**.

Detailed calculation method is in following equation:

fdco = output frequency * new_hs_div * new_n1 * 64

There are three constraints for the equation:

1. ***4850 < output fequency * new_hs_div * new_n1 < 5600***
2. ***4 <= new_hs_div <= 11***
3. ***1 <= new_n1 < =128***

For example, you want to get a 133.5 mhz clock, then

fdco = 133.5 x 4 x 10 x 64 = 341760d = 0x53700

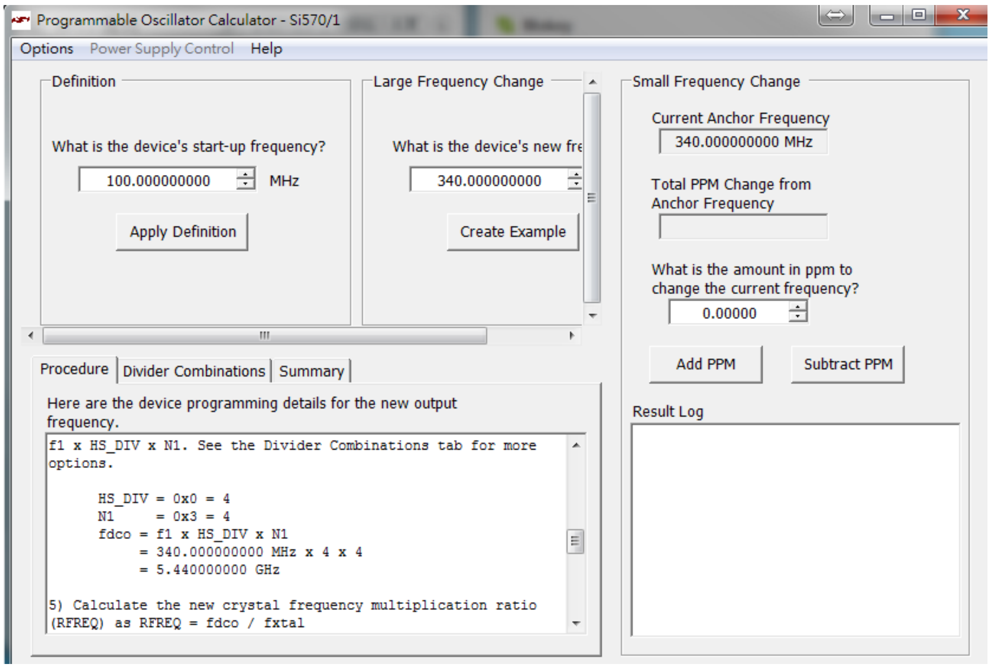
Find a mode in this RTL code section and modify these three parameters,as shown below:

```
new_hs_div = 3'b100 ;
new_n1 = 4'b1010 ;
fdco = 23'h05_3700 ;
```

In addition, Silicon Lab also provide the corresponding calculation tool.

Users can refer to the Programmable Oscillator tool (See **Figure 5-5)** mentioned in below link to calculate the values of new_hs_div and new_n1, then, the fdco value can be calcuted with above ftdo equation.

http://www.silabs.com/products/clocksoscillators/oscillators/Pages/oscillator-software-development-tools.aspx



Here are the device programming details for the new output frequency.

```
f1 x HS_DIV x N1. See the Divider Combinations tab for more
options.

      HS_DIV = 0x0 = 4
      N1     = 0x3 = 4
      fdco = f1 x HS_DIV x N1
           = 340.000000000 MHz x 4 x 4
           = 5.440000000 GHz

5) Calculate the new crystal frequency multiplication ratio
(RFREQ) as RFREQ = fdco / fxtal
```

Figure 5-5 Programmable Oscillator Calculator tool

In addition, if the user doesn't want Si570 controller to configure Si570 as soon as the FPGA configuration finishes, users can change settings in Si570_controller.v, shown below.

```
initial_config initial_config(

.iCLK(iCLK), // system     clock 50mhz
.iRST_n(iRST_n), // system reset
.oINITIAL_START(initial_start),
.iINITIAL_ENABLE(1'b1),
);
```

Changing the setting from ".iINITIAL_ENABLE(1'b1) " to ".iINITIAL_ENABLE(1'b0)" will disable the initialization function of Si570 Controller.

■ Design Tools

- Quartus II 13.1

■ Demonstration Source Code

- Project directory: Si570_Demonstration
- Bit stream used: Si570_Demonstration.sof
- Demonstration Batch File : test_ub2.bat
- Demo Batch File Folder: Si570_Demonstration \demo_batch

The demo batch file folders include the following files:

- Batch File: test_ub2.bat
- FPGA Configuration File: Si570_Demonstration.sof

■ Demonstration Setup

- Make sure Quartus II is installed on your PC.
- Connect the USB Blaster cable to the FGPA 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, Si570_Demonstration\demo_batch
- Press BUTTON1 to configure the Si570.
- Observe LED3 status.

5.3 Si570 and CDCM Programming by Nios II

This demonstration shows how to use the Nios II processor to program both programmable oscillators Si570 and CDCM on the FPGA board. The demonstration also includes a function to monitor system temperature with the on-board temperature sensor.

■ System Block Diagram

Figure 5-5 shows the system block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The three peripheral temperature sensor, Si570, and CDCM61004 are all controlled by Nios II through the PIO controller. The temperature sensor and external PLL Si570 are controlled through I2C interface. The Nios II program toggles the PIO controller to implement the I2C protocol. The CDCM 61004 is programmed through the PIO directly. The Nios

II program is running in the on-chip memory.

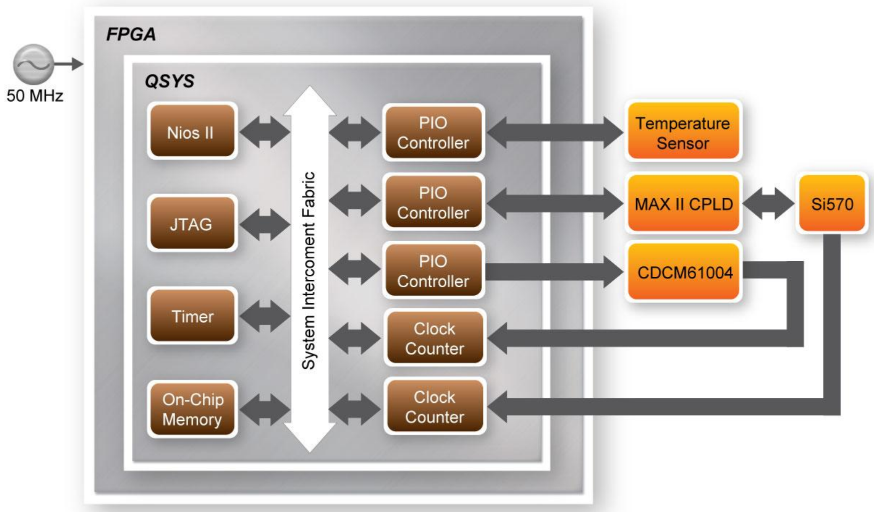
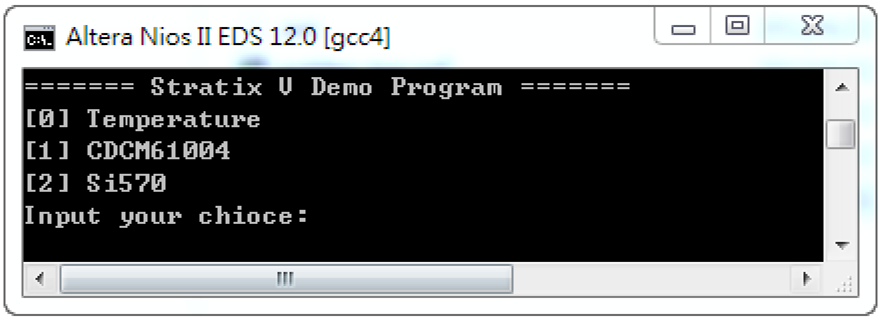


Figure 5-6 Block diagram of the Nios II Basic Demonstration

The program provides a menu in nios-terminal, as shown in **Figure 5-6** to provide an interactive interface. With the menu, users can perform the test for the temperatures sensor and external PLL. Note, pressing 'ENTER' should be followed with the choice number.



```
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:
```

Figure 5-7 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 the board temperature where the temperature sensor located.

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 CDMC61004 programming, users must trigger the CLK_RST_n to notify the chip to perform PLL recalibration. For Si570 programming, please note the device I2C address is 0x00. Also, before configuring the output frequency, users must freeze the DCO (bit 4 of Register 137) first. After configuring the output frequency, users must un-freeze the DCO and assert the NewFreq bit (bit 7 of Register 135).

- Design Tools
- Quartus II 12.1
- Nios II Eclipse 12.1
- Demonstration Source Code
- Quartus II Project directory: Nios_BASIC_DEMO
- Nios II Eclipse: Nios_BASIC_DEMO\Software
- Nios II IDE Project Compilation
- Before you attempt to compile the reference design under Nios II Eclipse, make sure the project is cleaned first by clicking on 'Clean' in the 'Project' menu of Nios II Eclipse.
- Demonstration Batch File

Demo Batch File Folder: *Nios_BASIC_DEMO\demo_batch*

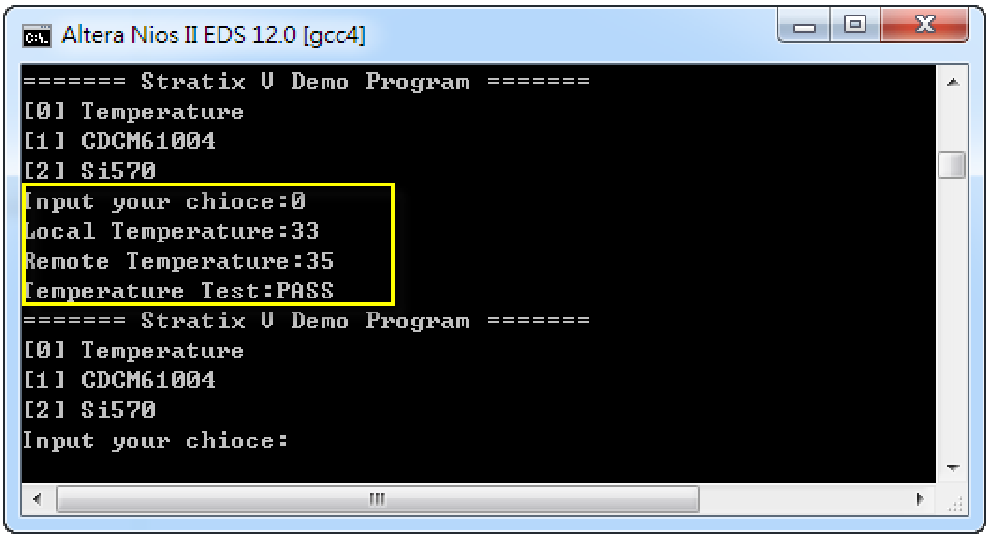
The demo batch file includes following files:

- Batch File for USB-Blaster II: test_ub2.bat, test_bashrc_ub2
- FPGA Configure File: golen_top.sof
- Nios II Program: Nios_DEMO.elf

Demonstration Setup

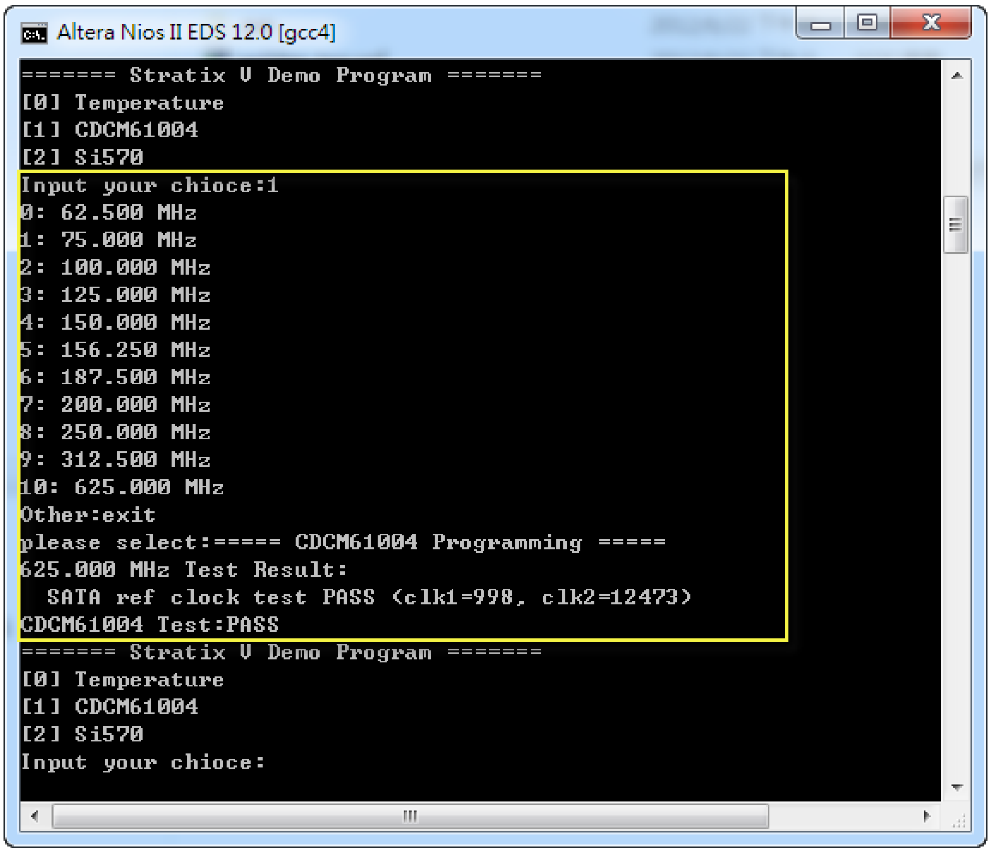
- Make sure Quartus II and Nios II are installed on your PC.
- Power on the FPGA board.
- Use the USB Cable to connect your PC and the FPGA board and install USB Blaster II driver if necessary.
- Execute the demo batch file "test_ub2.bat" under the batch file folder, Nios_BASIC_DEMO\demo_batch
- After 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 5-7**.
- For programming PLL CDCD61004 test, please input key ‘1’ and press ‘Enter’ in the nios-terminal first, then select the desired output frequency , as shown in 158H158H **Figure 5-9**.
- For programming PLL Si570 test, please input key ‘2’ and press ‘Enter’ in the nios-terminal first, then select the desired output frequency , as shown in 159H159H **Figure 5-10**.



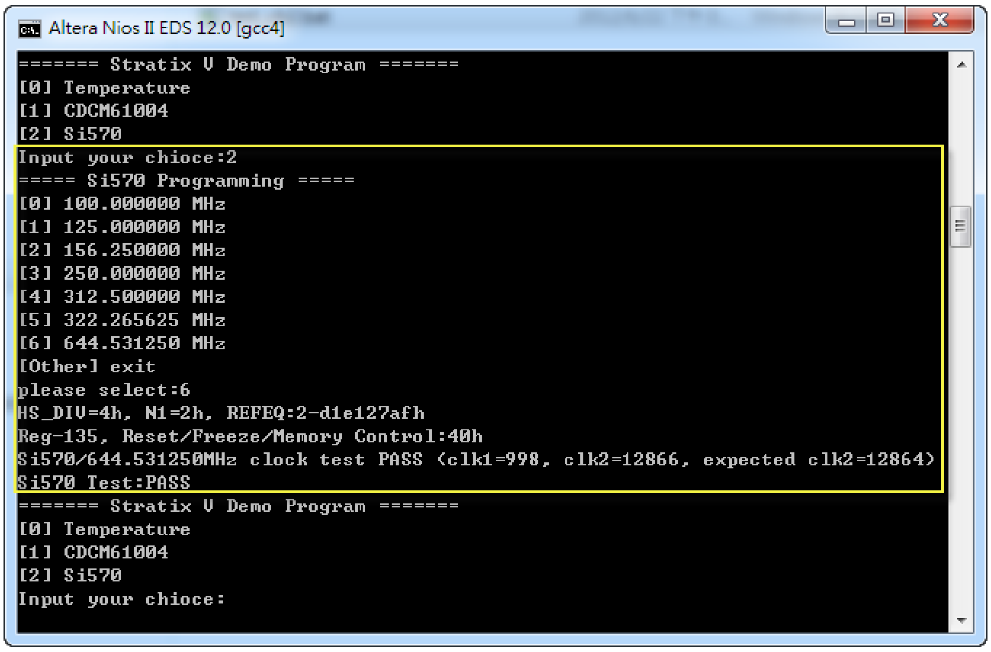
```
Altera Nios II EDS 12.0 [gcc4]
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:0
Local Temperature:33
Remote Temperature:35
Temperature Test:PASS
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:
```

Figure 5-8 Temperature Demo



```
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:1
0: 62.500 MHz
1: 75.000 MHz
2: 100.000 MHz
3: 125.000 MHz
4: 150.000 MHz
5: 156.250 MHz
6: 187.500 MHz
7: 200.000 MHz
8: 250.000 MHz
9: 312.500 MHz
10: 625.000 MHz
Other:exit
please select:===== CDCM61004 Programming =====
625.000 MHz Test Result:
  SATA ref clock test PASS (clk1=998, clk2=12473)
CDCM61004 Test:PASS
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:
```

Figure 5-9 CDCM 61004 Demo



```
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:2
===== Si570 Programming =====
[0] 100.000000 MHz
[1] 125.000000 MHz
[2] 156.250000 MHz
[3] 250.000000 MHz
[4] 312.500000 MHz
[5] 322.265625 MHz
[6] 644.531250 MHz
[Other] exit
please select:6
HS_DIV=4h, N1=2h, REFEQ:2-d1e127afh
Reg-135, Reset/Freeze/Memory Control:40h
Si570/644.531250MHz clock test PASS (clk1=998, clk2=12866, expected clk2=12864)
Si570 Test:PASS
======= Stratix V Demo Program =======
[0] Temperature
[1] CDCM61004
[2] Si570
Input your chioce:
```

Figure 5-10 Si570 Demo

Chapter 6

Memory Reference Design

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

- DDR3 SDRAM: two independent banks, update to 800 MHz
- QDRII+ SRAM: four independent banks, update to 550 MHz

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

- QDRII+ SRAM Test: Full test of the four banks of QDRII+ SRAM
- DD3 SDRAM Test: Random test of the two banks of DDR3 SDRAM.
- DDR3 SDRAM Test by Nios II: Full test of one bank of DDR3 SDRAM with Nios II

Note. 64-Bit Quartus 12.1 or later is strongly recommended for compiling these projects.

6.1 QDRII+ SRAM Test

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

This demonstration utilizes four QDRII+ SRAMs on the FPGA board. It describes how to use Altera's "QDRII and QDRII+ SRAM Controller with UniPHY" IP to implement a memory test function. In the design, the four QDRII controllers share the PLL/DLL/OCT due to limited DLL numbers in the FPGA.

■ Function Block Diagram

Figure 6-1 shows the function block diagram of the demonstration. The four QDRII+ SRAM controllers are configured as a 72Mb controller. The QDRII+ SRAM IP generates a 550MHz clock as memory clock and a half-rate system clock, 275MHz, for the controllers.

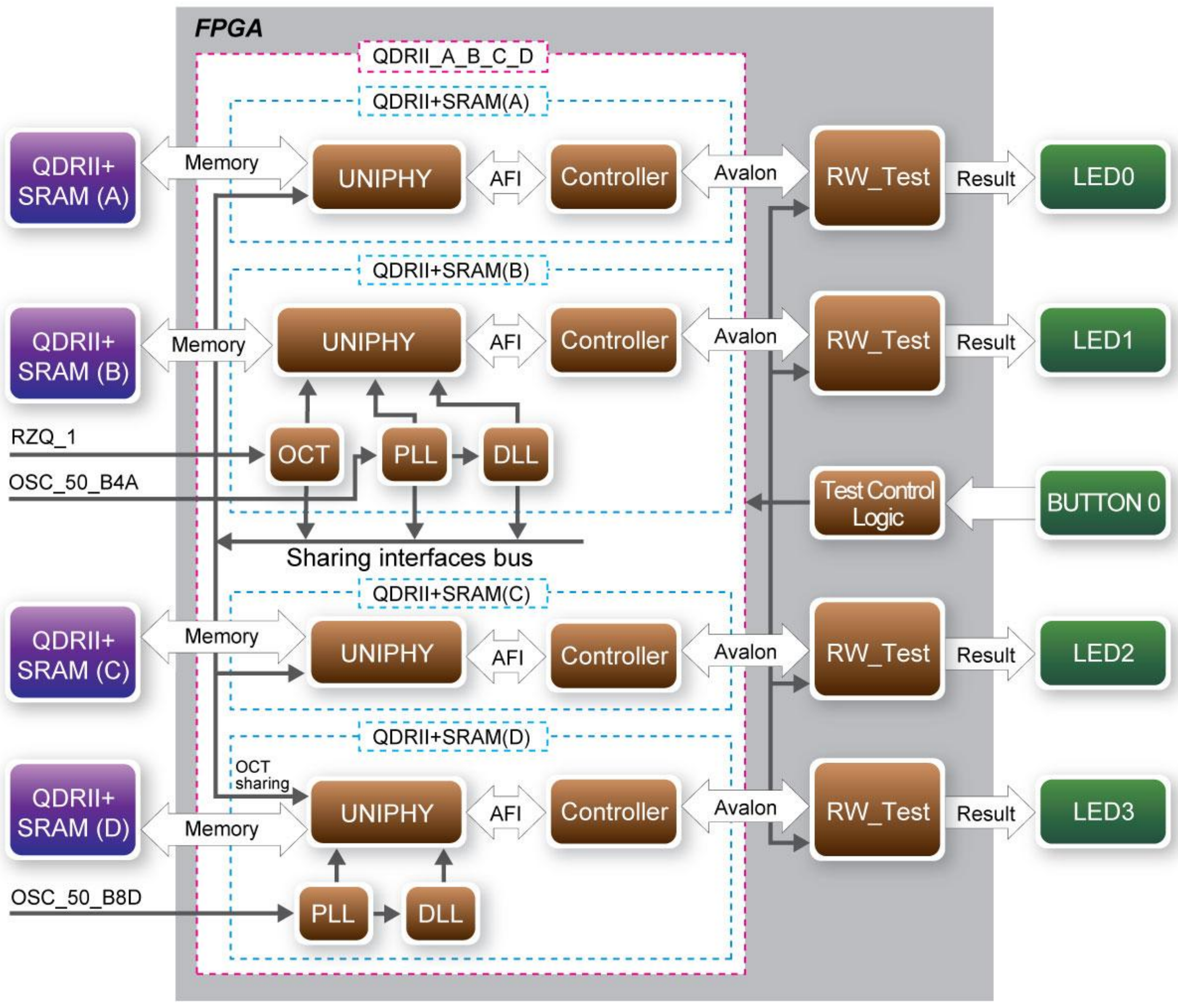


Figure 6-1 Function Block Diagram of the QDRII+ SRAM x4 Demonstration

In this demonstration, four QDRII+ SRAM controllers are sharing the FPGA resources (OCT, PLL, and DLL), and the QDRII+ SRAM (B) is configured as the master to share the resource to the other three slave QDRII+ SRAM (A/C/D).QDRII+ SRAM (A/C) share OCT, PLL, DLL from QDRII+ SRAM (B). QDII+ SRAM (D) shares OCT from QDRII+ SRAM (B) and it has its own PLL and DLL resources. 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 BUTTON0 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.
3. Perform "Analysis and Synthesis" by clicking Quartus menu: Process→Start→Start Analysis & Synthesis.
4. Run the TCL files generated by QDRII+ IP by clicking Quartus menu: Tools→TCL Scripts…

- Design Tools
- Quartus II 12.1
- Demonstration Source Code
- Project directory: QDRIIx4_Test
- Bit stream used: QDRIIx4_Test.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_ub2.bat,
- FPGA configuration file: QDRIIx4_Test.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, QDRIIx4_Test\demo_batch.

- Press BUTTON0 of the FPGA board to start the verification process. When BUTTON0 is held down, all the LEDs will be turned off. All LEDs should turn back on to indicate test passes upon the release of BUTTON0.

- If any LED is not lit up after releasing BUTTON0, it indicates the corresponding QDRII+ SRAM test has failed. 161H161H **Table 6-1** lists the matchup for the four LEDs.
- Press BUTTON0 again to regenerate the test control signals for a repeat test.

Table 6-1 LED Indicators

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

6.2 DDR3 SDRAM Test

This demonstration presents a memory test function on the two sodimm of DDR3-SDRAM on the FPGA board. The memory size of each DDR3 SDRAM sodimm used in this test is 2 GB.

■ Function Block Diagram

Figure 6-2 shows the function block diagram of this demonstration. There are two DDR3 SDRAM controllers. One is the master controller which shares resources with a slave controller. The shared resources include delay-locked loops (DLLs), phase-locked loops (PLLs), and on-chip termination (OCT). The controller uses 50 MHz as a reference clock, generates one 800.0 MHz clock as memory clock, and generates one quarter-rate system clock 200.0 MHz for the controller itself.

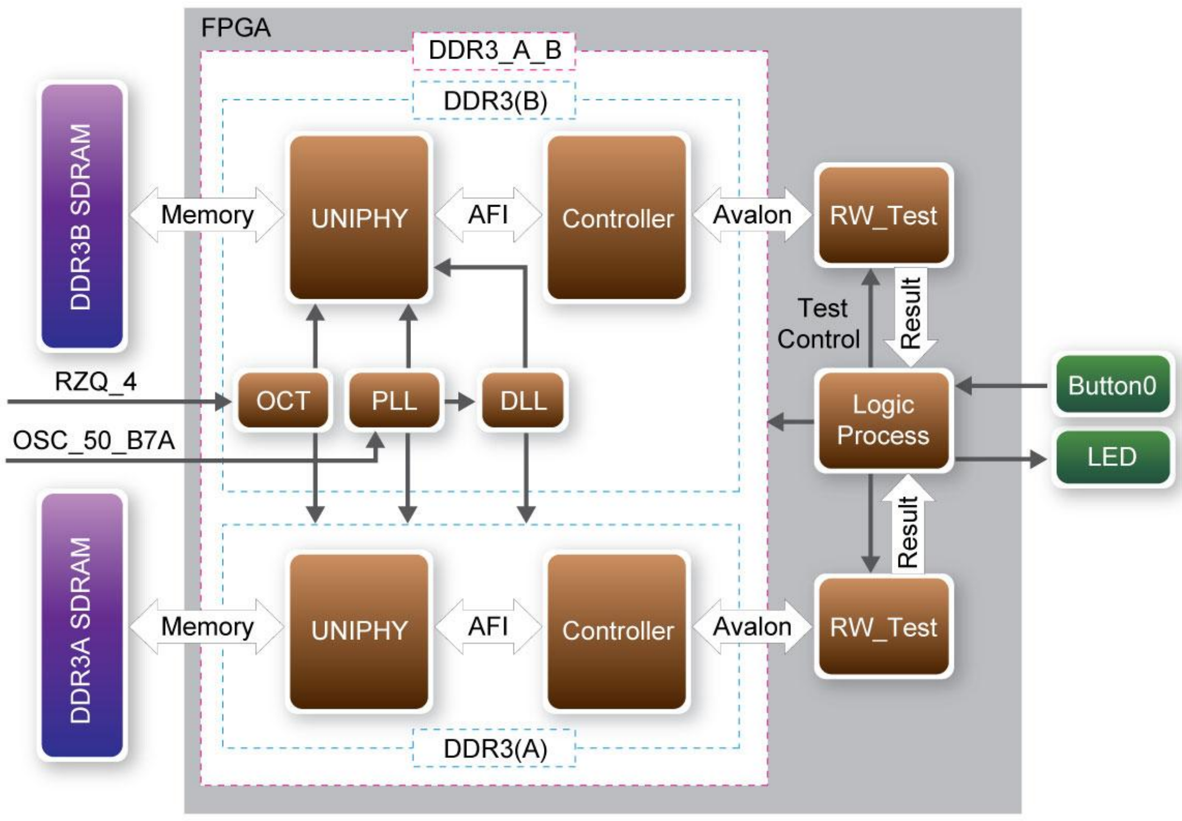


Figure 6-2 Block Diagram of the DDR3 SDRAM (2G) x2 Demonstration

■ Altera DDR3 SDRAM Controller with UniPHY

To use the Altera DDR3 controller, users need to perform three major steps:

1. Create correct pin assignments for the DDR3.
2. Setup correct parameters in the DDR3 controller dialog.
3. Perform "Analysis and Synthesis" by selecting from the Quartus II menu: Process→Start→Start Analysis & Synthesis.
4. Run the TCL files generated by DDR3 IP by selecting from the Quartus II menu: Tools→TCL Scripts…

■ Design Tools

- 64-Bit Quartus 12.1
- Demonstration Source Code
- Project directory: DDR3x2_Test
- Bit stream used: DDR3x2_Test.sof
- Demonstration Batch File

Demo Batch File Folder: *DDR3x2_Test \demo_batch*

The demo batch file includes following files:

- Batch File: test_ub2.bat
- FPGA Configure File: DDR3x2_Test .sof

Demonstration Setup

- Make sure Quartus II is installed on your PC.
- Connect the USB Blaster 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, DDR3x2_Test \demo_batch.
- Press BUTTON0 on the FPGA board to start the verification process. When BUTTON0 is pressed, all the LEDs (LED [3:0]) should turn on. At the instant of releasing BUTTON0, LED1, LED2, LED3 should start blinking. After approximately 5 seconds, LED1 and LED2 should stop blinking and stay on to indicate that the DDR3 (A) and DDR3 (B) have passed the test, respectively. Table 6-2 lists the LED indicators.
- If LED3 is not blinking, it means the 50MHz clock source is not working.
- If LED1 or LED2 do not start blinking after releasing BUTTON0, it indicates local_init_done or local_cal_success of the corresponding DDR3 failed.
- If LED1 or LED2 fail to remain on after 5 seconds, the corresponding DDR3 test has failed.
- Press BUTTON0 again to regenerate the test control signals for a repeat test.

Table 6-2 LED Indicators

NAME	*Description*
LED0	Reset
LED1	DDR3 (A) test result
LED2	DDR3 (B) test result
LED3	Blinks

6.3 DDR3 SDRAM Test by Nios II

Many applications use a high performance RAM, such as a DDR3 SDRAM, to provide temporary storage. In this demonstration hardware and software designs are provided to illustrate how to perform DDR3 memory access in QSYS. We describe how the Altera's "DDR3 SDRAM Controller with UniPHY" IP is used to access a DDR3-SDRAM, and how the Nios II processor is used to read and write the SDRAM for hardware verification. The DDR3 SDRAM controller handles the complex aspects of using DDR3 SDRAM by initializing the memory devices, managing SDRAM banks, and keeping the devices refreshed at appropriate intervals.

■ System Block Diagram

Figure 6-3 shows the system block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The DDR3 controller is configured as a 1 GB DDR3-800Mhz controller. The DDR3 IP generates one 800 MHz clock as SDRAM's data clock and one quarter-rate system clock 800/4=200 MHz for those host controllers, e.g. Nios II processor, accessing the SDRAM. In the QSYS, Nios II and the On-Chip memory are designed running with the 233.333 MHz clock, and the Nios II program is running in the on-chip memory.

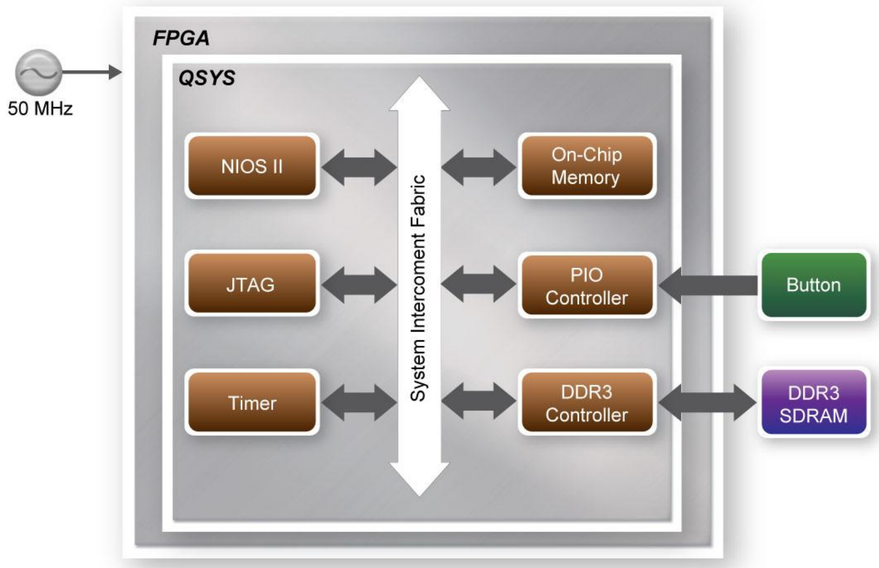


Figure 6-3 Block diagram of the DDR3 Basic Demonstration

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

■ Altera DDR3 SDRAM Controller with UniPHY

To use Altera DDR3 controller, users need to perform the four major steps:

5. Create correct pin assignments for DDR3.
6. Setup correct parameters in DDR3 controller dialog.
7. Perform "Analysis and Synthesis" by clicking Quartus menu: Process→Start→Start Analysis & Synthesis.
8. Run the TCL files generated by DDR3 IP by clicking Quartus menu: Tools→TCL Scripts…

■ Quartus II Project

The Quartus II project is designed to only access DDR3-A or DDR3-B at same time due to the address space limitation of Nios II. Users can change the accessed memory target at Quartus compile time by defining the constant USE_DDR3_A for DDR3-A or constant USE_DDR3_B for DDR3-B bank. After the constant is defined, please perform Analysis and Synthesis and then run the TCL files generated by DDR3 IP before starting Quartus II compilation.

■ Design Tools

- Quartus II 12.1
- Nios II Eclipse 12.1

■ Demonstration Source Code

- Quartus Project directory: Nios_DDR3
- Nios II Eclipse: NIOS_DDR3\Software
- 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

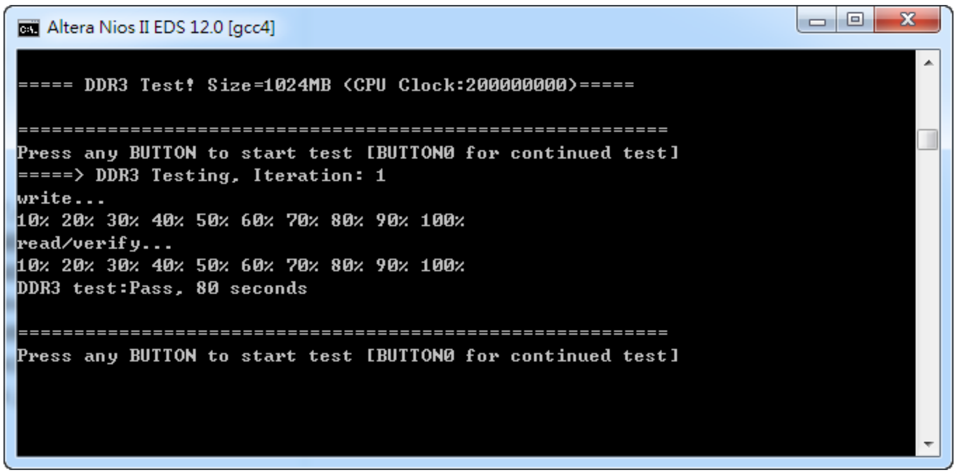
Demo Batch File Folder:

Nios_DDR3\demo_batch\DDR3_A or

Nios_DDR3\demo_batch\DDR3_B

The demo batch file includes following files:

- Batch File for USB-Blaseter II: test_ub2.bat, test_bashrc_ub2
- FPGA Configure File: Golen_top.sof
- Nios II Program: TEST_DDR3.elf
- Demonstration Setup
- Make sure Quartus II and Nios II are installed on your PC.
- 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_ub2.bat" under the batch file folder, NIOS_DDR3\demo_batch\DDR3_A or NIOS_DDR3\demo_batch\DDR3_B
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- Press Button1~Button0 of the FPGA board to start SDRAM verify process. Press Button0 for continued test and press any to terminate the continued test.
- The program will display progressing and result information, as shown in 164H164H **Figure 6-4**.



```
===== DDR3 Test! Size=1024MB (CPU Clock:200000000)=====

==========================================================
Press any BUTTON to start test [BUTTON0 for continued test]
=====> DDR3 Testing, Iteration: 1
write...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
read/verify...
10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
DDR3 test:Pass, 80 seconds

==========================================================
Press any BUTTON to start test [BUTTON0 for continued test]
```

Figure 6-4 Display Progress and Result Information for the DDR3 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.

7.1 PCI Express System Infrastructure

The system consists of two primary components, the FPGA hardware implementation and the PC-based application. The FPGA hardware component is developed based on Altera PCIe IP, and the PC-based application is developed under the Jungle driver. **Figure 7-1** shows the system infrastructure. The Terasic PCIe IP license is located in the FPGA System CD under the directory (/CDROM/License/Terasic_PCIe_TX_RX). This license is required in order to compile the PCIe design projects provided below. In case the license expires, please visit the FPGA website (DE5-Net.terasic.com) to acquire and download a new license.

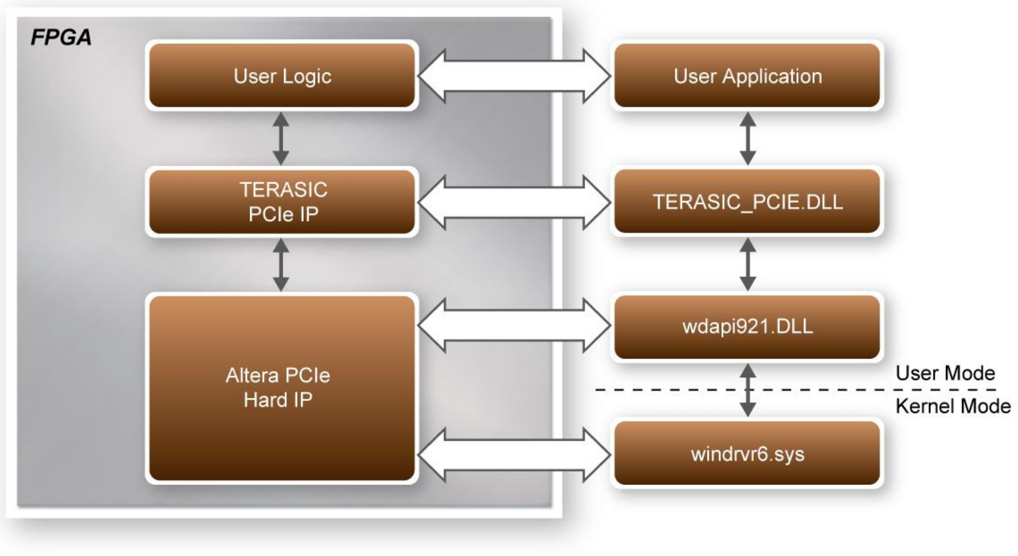


Figure 7-1 PCI Express System Infrastructure

7.2 FPGA PCI Express System Design

The PCI Express edge connector is able to allow interconnection to the PCIe motherboard slots. For basic I/O control, a communication is established through the PCI Express bus where it is able to control the LEDs and monitor the button status of the FPGA board. By implementing an internal RAM and FIFO, the demonstration is capable of direct memory access (DMA) transfers. The FPGA can also generate an interrupt to notify the Host System through the PCI Express bus.

■ PCI Express Basic I/O Transaction

Under read operation, the Terasic PCIe IP issues a read signal followed by the address of the data. Once the address is received, a 32-bit data will be sent along with a read valid signal. Under write operation, the PCIe IP issues a write signal accompany with the address to be written. A 32-bit data is written to the corresponding address with a data enable signal of write operation. All the write commands are issued on the same clock cycle. Table 7–1 lists the associated port names along with the description.

Table 7–1 Single Cycle Transaction Signals of Terasic PCIe IP

Name	*Type*	*Polarity*	*Description*
oCORE_CLK	Output	-	Clock. The reference clock output of PCIe local interface.
oSC_RD_ADDR[11..0]	Output	-	Address bus of read transaction. It is a 32-bit data per address.
iSC_RD_DATA [31..0]	Input	-	Read data bus.

oSC_RD_READ	**Output**	**High**	**Read signal.**
iSC_RD_DVAL	**Input**	**High**	**Read data valid.**
oSC_WR_ADDR[11..0]	**Output**	**-**	**Address bus of write transaction. It is a 32-bit data per address.**
oSC_WR_DATA[31..0]	**Output**	**-**	**Write data bus.**
oSC_WR_WRITE	**Output**	**High**	**Write signal.**

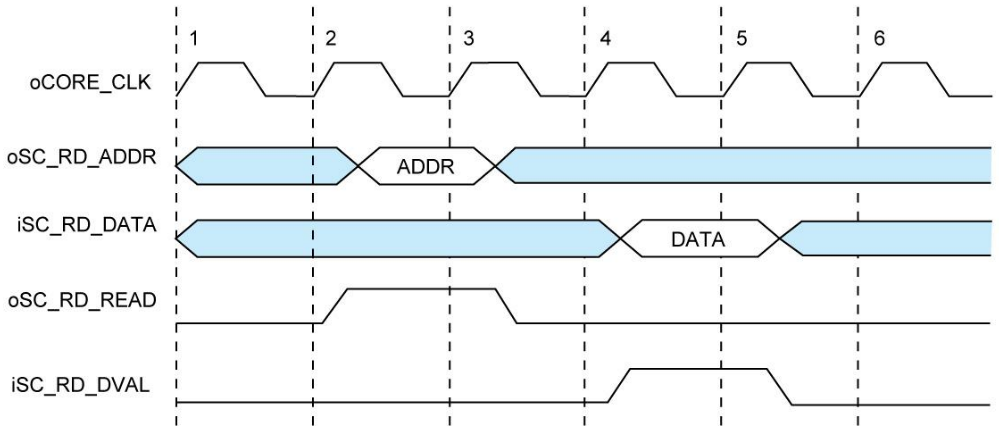


Figure 7-2 Read transaction waveform of the PCIe basic I/O interface

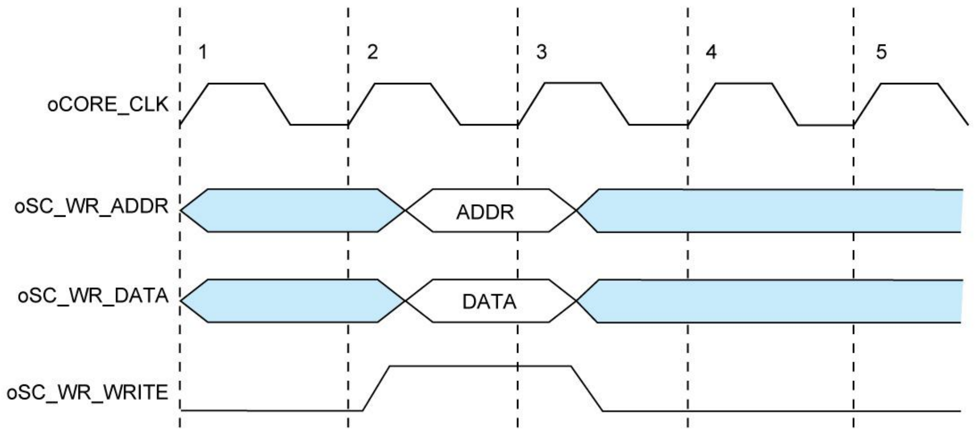


Figure 7-3 Write transaction waveform of the PCIe basic I/O interface

■ PCI Express DMA Transaction

To support greater bandwidth and to improve latency, Terasic PCIe IP provides a high speed DMA channel with two modes of interfaces including memory mapping and FIFO link. The oFIFO_MEM_SEL signal determines the DMA channel used, memory mapping or FIFO link, which is enabled with the assertion of a low and high signal, respectively. The address bus of DMA indicates the FIFO ID which is defined by user from the PC software API.

Most interfaces experience read latency during the event data is read and processed to the output. To mitigate the overall effects of read latency, minimum delay and timing efficiency is required to enhance the performance of the high-speed DMA transfer. As oDMARD_READ signal is asserted, the read data valid signal oDMARD_RDVALID is inserted high to indicate the data on the iDMARD_DATA data bus is valid to be read after two clock cycles.

Table 7–2 DMA Channel Signals of Terasic PCIe IP

Name	*Type*	*Polarity*	*Description*
oCORE_CLK	Output	-	Clock. The reference clock output of PCIe local interface.
oDMARD_ADDR[31..0]	Output	-	When oFIFO_MEM_SEL is set to low, it is address bus of DMA transfer and the value of address bus is cumulative by PCIe IP and it is 128-bit data per address. When oFIFO_MEM_SEL is set to high, oDMARD_ADDR bus is a FIFO ID that is used to indicate that which FIFO buffer is selected by PC API.
iDMARD_DATA [127..0]	Input	-	Read data bus.
oDMARD_READ	Output	High	Read signal.
oDMARD_RDVALID	Input	High	Read data valid.
oDMAWR_ADDR[31..0]	Output	-	When oFIFO_MEM_SEL is set to low, it is address bus of DMA transfer and the value of address bus is cumulative by PCIe IP and it is 128-bit data per address. When oFIFO_MEM_SEL is set to high, oDMARD_ADDR bus is a FIFO ID that is used to indicate that which FIFO buffer is selected by PC API.
oDMAWR_DATA[127..0]	Output	-	Write data bus.
oDMAWR_WRITE	Output	High	Write signal.
oFIFO_MEM_SEL	Output	-	Indicates that DMA channel is memory mapping interface or FIFO-link interface. When this signal is asserted high, DMA channel FIFO-link interface. When the signal is asserted low, it is memory mapping interface.

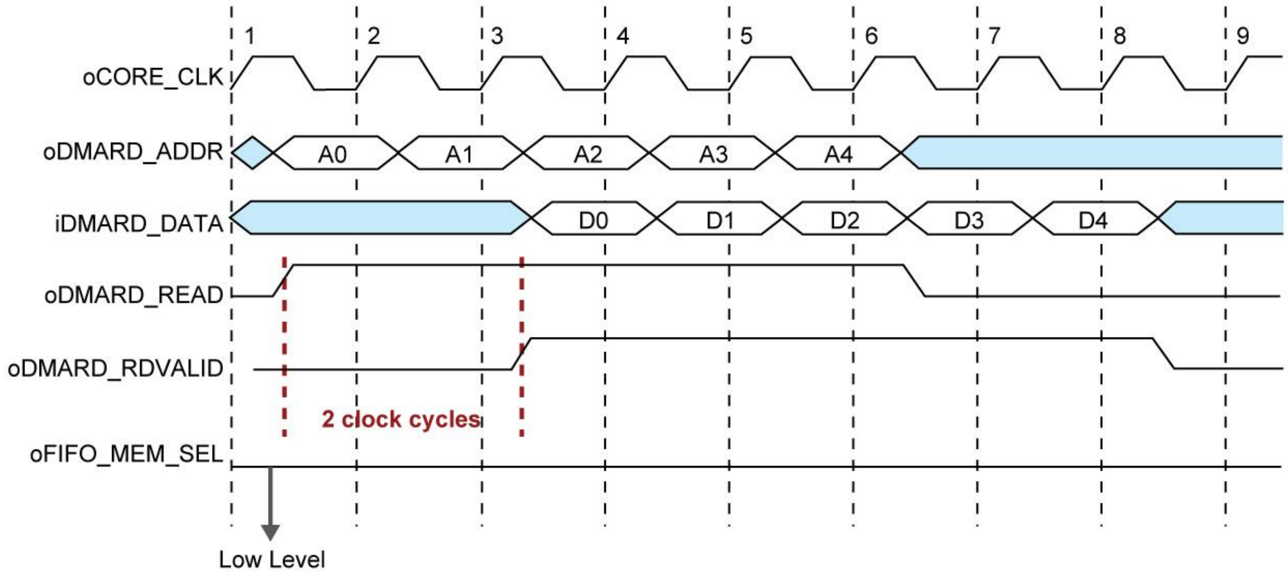


Figure 7-4 Read transaction waveform of the PCIe DMA channel on memory mapping mode

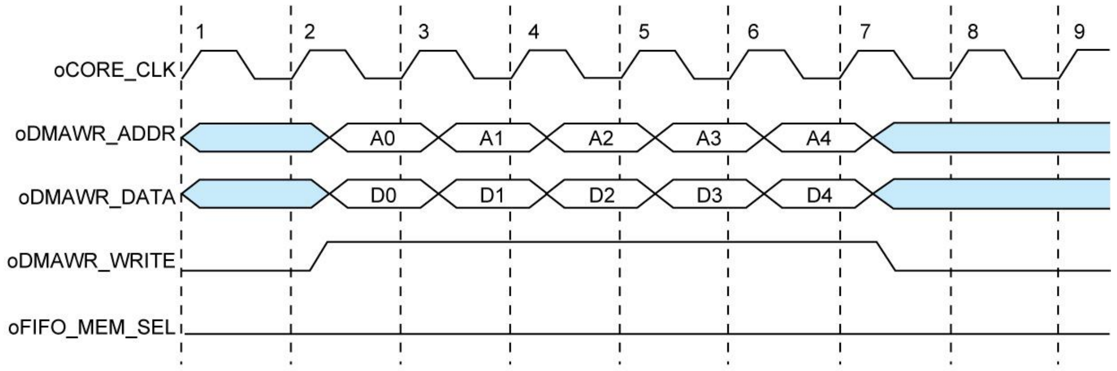


Figure 7-5 Write transaction waveform of the PCIe DMA channel on memory mapping mode

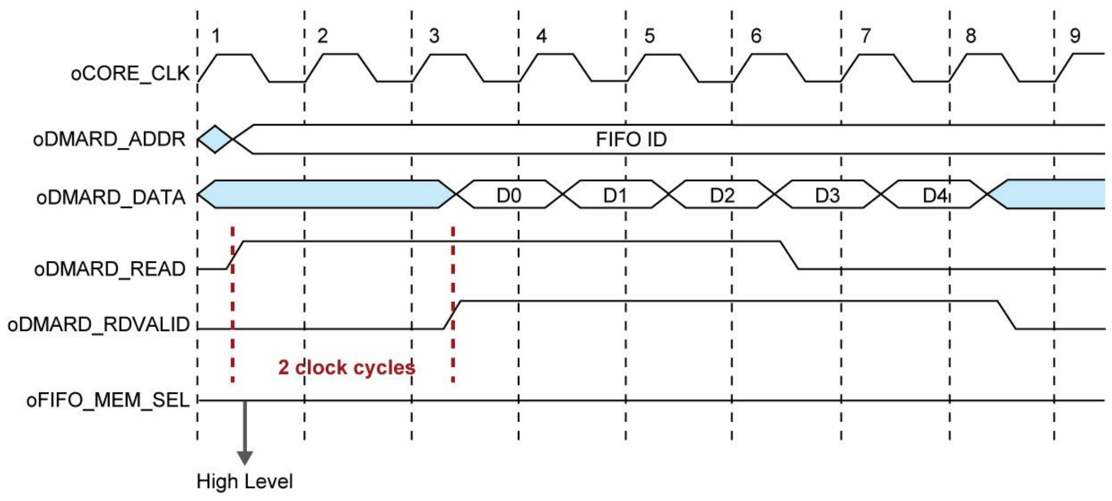


Figure 7-6 Read transaction waveform of the PCIe DMA channel on FIFO-link mode

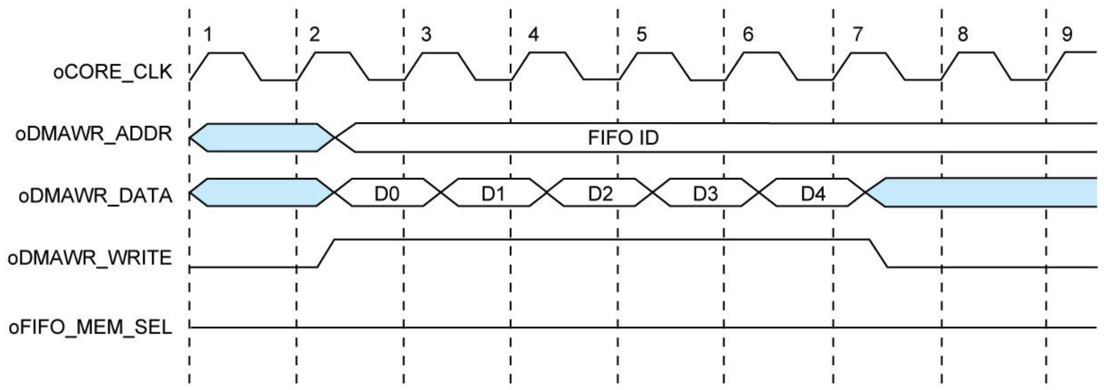


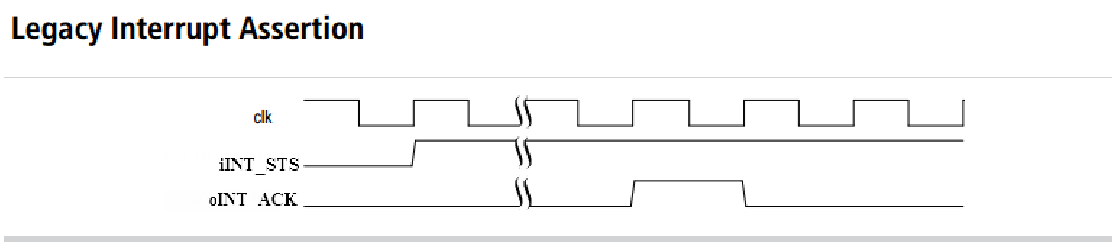
Figure 7-7 Write transaction waveform of the PCIe DMA channel on FIFO-link mode

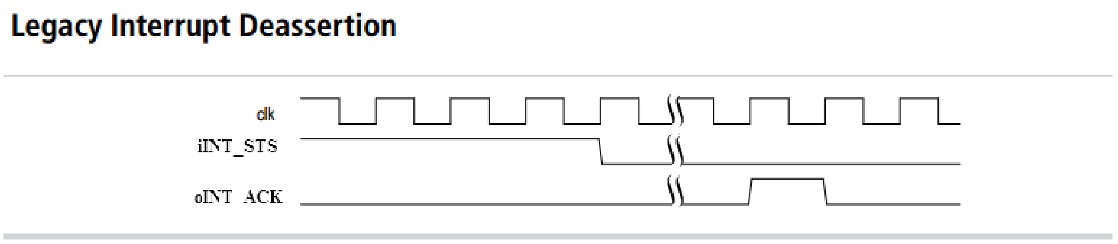
■ PCI Express Interrupt

Altera PCI Express IP supports both interrupt types specified in PCI Express protocol. One of them is Legacy interrupt and the other one is MSI interrupt. This reference design shows how to use Legacy interrupt mechanism to handle interrupt. Table 7-3 lists the associated port names along with the description. The Edge Detector captures the timing when the button is pressed and generates a pulse on iINT_STS. The PCIe Hard IP sends an INTA_Assert message upstream to the Root Complex in response to a low-to-high transition. It also sends an INTA_Deassert message in response to a high-to-low transition. A pulse on oINT_ACK to user logic indicates that an INTA_Assert or INTA_ Deassert message has been sent.

Table 7–3 Interrupt Channel Signals of Terasic PCIe IP

NAME	Type	Description
iINT_STS	output	The user logic uses this signal to generate a legacy INT interrupt.
oINT_ACK	input	A pulse on this output indicates that an INTx_Assert or INTX_ Deassert message has been sent.





7.3 PC PCI Express System Design

The FPGA System CD contains a PC Windows based SDK to allow users to develop their 32-bits software application on Windows 7/Window XP 32/64-bits. 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 requires users to modify the PCIe vender ID (VID) and device ID (DID) in the driver INF file to match the design in the FPGA where Windows searches for the associated driver.

The PCI Express Library is implemented as a single DLL called TERASIC_PCIE.DLL (TERASIC_PCIEx64.DLL for 64-bits Windows). With the DLL exported to the software API, users can easily communicate with the FPGA. The library provides the following functions:

- Device Scanning on PCIe Bus
- Basic Data Read and Write
- Data Read and Write by DMA

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

■ PCI Express Software Stack

Figure 7-8 shows the software stack for the PCI Express application software on 32-bits Windows. The PCI Express driver is incorporated in the DLL library called TERASIC_PCIE.dll. Users can develop their application based on this DLL. In 64-bits Windows, TERASIC_PCIE.dll is replaced by TERASIC_PCIEx64.dll, and wdapi921.dll is replaced by wdapi1100.dll.

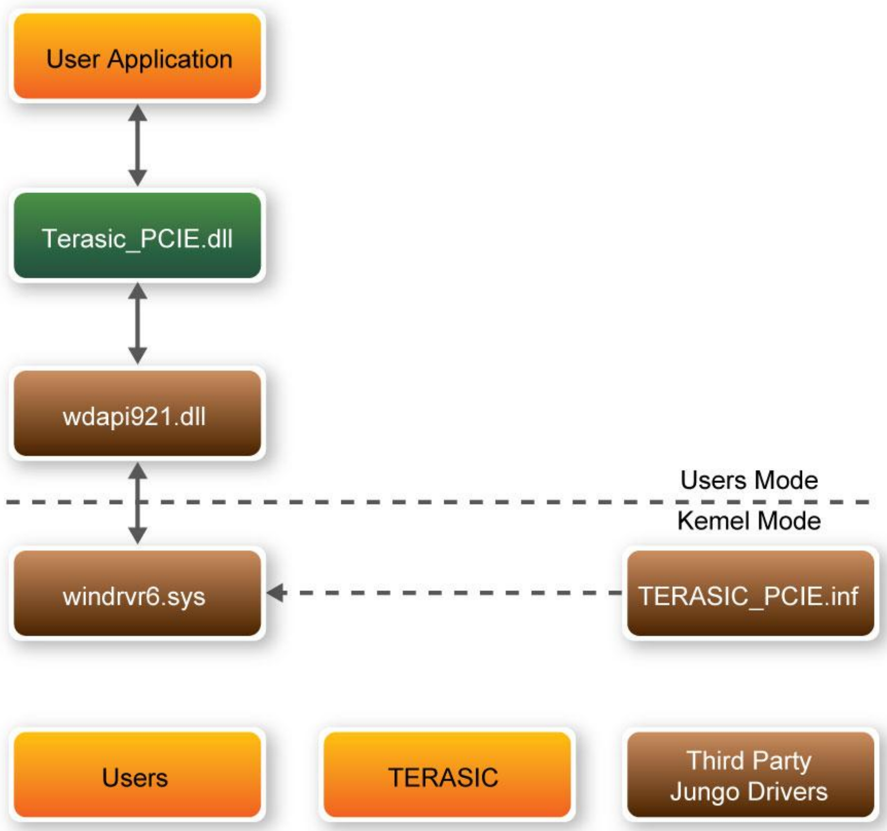


Figure 7-8 PCI Express Software Stack

■ Install PCI Express Driver

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

1. From the FPGA system CD locate the PCIe driver folder in the directory \CDROM\demonstrations\PCIe_SW_KIT\PCIe_DriverInstall.

2. Double click the "PCIe_DriverInstall.exe" executable file to launch the installation program shown in **Figure 7-9**.

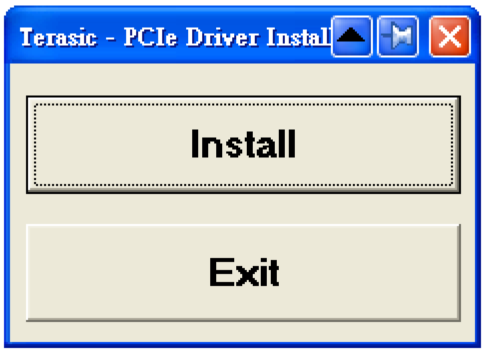
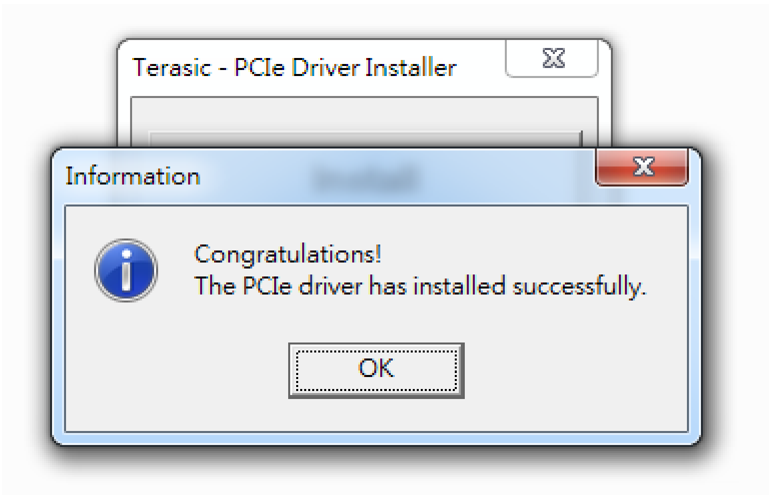


Figure 7-9 PCIe Driver Installation Program

3. Click “Install” to begin installation process.
4. It takes several seconds to install the driver. When installation is complete, the following dialog window will popup shown in Figure 7-10. Click “OK” and then “Exit” to close the installation program.



Congratulations!
The PCIe driver has installed successfully.

Figure 7-10 PCIe driver installed successfully

5. Once the driver is successfully installed, users can view the device under the device manager window shown in Figure 7-11.

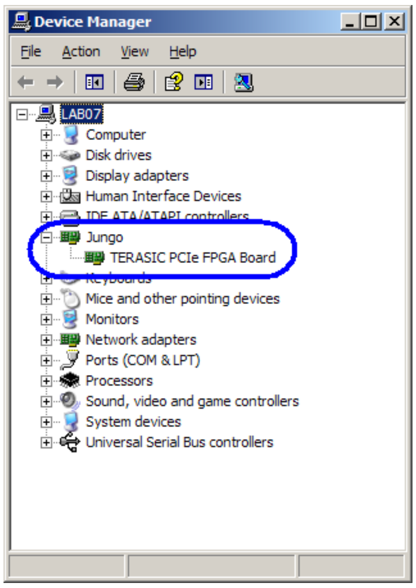


Figure 7-11 Device Manager

■ Create a Software Application

All necessary files to create a PCIe software application are located in the *CDROM\demonstration\PCIe_SW_KIT\PCIe_Library* which includes the following files:

- TERASIC_PCIE.h
- TERASIC_PCIE.DLL (for 32-bits Windows)
- TERASIC_PCIEx64.DLL (for 64-bits Windows)
- wdapi921.dll (for 32-bits Windows)
- wdapi1100.dll (for 64-bits Windows)

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

- Create a C/C++ project.
- Include TERASIC_PCIE.h in the 32-bits C/C++ project.
- Copy TERASIC_PCIE.DLL(TERASIC_PCIEx64.DLL for 64-bits Windows) to the folder where the project.exe is located.
- Dynamically load TERASIC_PCIE.DLL(TERASIC_PCIEx64 for 64-bits Windows) in C/C++ project. To load the DLL, please refer to two examples below.
- Call the SDK API to implement the desired application.
- TERASIC_PCIE.DLL/TERASIC_PCIEx64.DLL Software API

Using the TERASIC_PCIE.DLL/TERASIC_PCIEx64.DLL software API, users can easily communicate with the FPGA through the PCIe bus. The API details are described below :

PCIE_ScanCard

Function: Lists the PCIe cards which matches the given vendor ID and device ID. Set Both ID to zero to lists the entire PCIe card.
Prototype: BOOL PCIE_ScanCard(WORD wVendorID, WORD wDeviceID, DWORD *pdwDeviceNum, PCIE_CONFIG szConfigList[]);
Parameters: wVendorID: Specify the desired vendor ID. A zero value means to ignore the vendor ID. wDeviceID: Specify the desired device ID. A zero value means to ignore the produce ID. pdwDeviceNum: A buffer to retrieve the number of PCIe card which is matched by the desired vendor ID and product ID. szConfigList: A buffer to retrieve the device information of PCIe Card found which is matched by the desired vendor ID and device ID.
Return Value: Return TRUE if PCIe cards are successfully enumeated; otherwise, FALSE is return.

PCIE_Open

Function: Open a specified PCIe card with vendor ID, device ID, and matched card index.
Prototype: PCIE_HANDLE PCIE_Open(

WORD wVendorID,
WORD wDeviceID,
WORD wCardIndex);

Parameters:

wVendorID:

Specify the desired vendor ID. A zero value means to ignore the vendor ID.

wDeviceID:

Specify the desired device ID. A zero value means to ignore the device ID.

wCardIndex:

Specify the matched card index, a zero based index, based on the matched verder ID and device ID.

Return Value:

Return a handle to presents specified PCIe card. A positive value is return if the PCIe card is opened successfully. A value zero means failed to connect the target PCIe card.

This handle value is used as a parameter for other functions, e.g. PCIE_Read32.

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

PCIE_Close

Function:

Close a handle associated to the PCIe card.

Prototype:

void PCIE_Close(
PCIE_HANDLE hPCIE);

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

Return Value:

None.

PCIE_Read32

Function:

Read a 32-bits 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-bits data.

Return Value:

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

PCIE_Write32

Function:

Write a 32-bits data to the FPGA Board.

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-bits 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 function.

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 function.

Prototype:

```
bool PCIE_DmaWrite(
    PCIE_HANDLE hPCIE,
    PCIE_LOCAL_ADDRESS LocalAddress,
    void *pData,
    DWORD dwDataSize
    );
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

LocalAddress:

Specify the target memory mapped address in FPGA.

pData:

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

dwDataSize:

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

Return Value:

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

PCIE_DmaFifoRead

Function:

Read data from the memory fifo of FPGA board in DMA function.

Prototype:

```
bool PCIE_DmaFifoRead(
    PCIE_HANDLE hPCIE,
    PCIE_LOCAL_FIFO_ID LocalFifoId,
    void *pBuffer,
    DWORD dwBufSize
    );
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

LocalFifoId:

Specify the target memory fifo ID 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_DmaFifoWrite

Function: Write data to the memory fifo of FPGA board in DMA function.
Prototype: bool PCIE_DmaFifoWrite(PCIE_HANDLE hPCIE, PCIE_LOCAL_FIFO_ID LocalFifoId, void *pData, DWORD dwDataSize);
Parameters: hPCIE: A PCIe handle return by PCIE_Open function. LocalFifoId: Specify the target memory fifo ID in FPGA. pData: A pointer to a memory buffer to store the data which will be written to FPGA. dwDataSize: Specify the byte number of data which will be written to FPGA.

PCIE_IntEnable

Function: Enable Interrupt function and given a user defined ISR(Interrupt Service Routine) while will be callback when an interrupt happens.
Prototype: bool PCIE_IntEnable (PCIE_HANDLE hPCIE, TERASIC_INT_HANDLER funcIntHandler);
Parameters: hPCIE:

A PCIe handle return by PCIE_Open function.

funcIntHandler:

Specify the interrupt service routine which will be callback when interrupt happens. TERASIC_INT_HANDLER is the prototype definition of Interrupt Service Routing. It is defined as:

```
typedef void (PCIE_API *TERASIC_INT_HANDLER)( PCIE_HANDLE hALTERA, PCIE_INT_RESULT *intResult)
```

The data structure of PCIE_INT_RESULT is defined as:

```
typedef struct
{
        DWORD dwCounter;      // number of interrupts received
        DWORD dwLost;         // number of interrupts not yet dealt with
        BOOL fStopped;        // was interrupt disabled during wait
} PCIE_INT_RESULT;
```

Return Value:

Return TRUE if interrupt is enabled successful; otherwise FALSE is returned.

PCIE_IntDisable

Function:

Disable Interrupt function. When interrupt is disabled, the user defined ISR(Interrupt Service Routine) will not be callback when an interrupt happens.

Prototype:

```
bool PCIE_IntEnable (
    PCIE_HANDLE hPCIE
    );
```

Parameters:

hPCIE:

A PCIe handle return by PCIE_Open function.

Return Value:

Return TRUE if interrupt is disabled successful; otherwise FALSE is returned.

7.4 Fundamental Communication

The application reference design shows how to implement fundamental control, interrupt and data transfer. In the design, basic I/O is used to control the BUTTON and LED on the FPGA board, and BUTTONs also are used to trigger interrupt event . High-speed data transfer is performed by DMA. Both Memory-Mapped and FIFO memory types are demonstrated in the reference design. The demonstration also lists the associated PCIe cards.

■ Demonstration Files Location

The demo file is located in the folder: CDRAOM\demonstrations\PCIE_Fundamental\Demo_batch

The folder includes following files:

- PC Application Software: PCIe_Fundamental_Demo.exe
- FPGA Configuration File: PCIE_Fundamental.sof
- PCIe Library : TERASIC_PCIE.DLL and wapi921.dll (TERASIC_PCIEx64.DLL and wapi1100.dll for 64-bits Widnows)
- Demo Batch File : test_ub2.bat
- Demonstration Setup
- Install the FPGA board on your PC.
- Download the PCIE_Fundamental.sof into the FPGA board using Quartus II Programmer.
- Restart Windows
- Install PCIe driver if necessary. The driver is located in the folder CDROM\demonstration \PCIe_SW_KIT\PCIe_DriverInstall.
- Launch the demo program PCIe_Fundamental_Demo.exe shown in **Figure 7-12**.

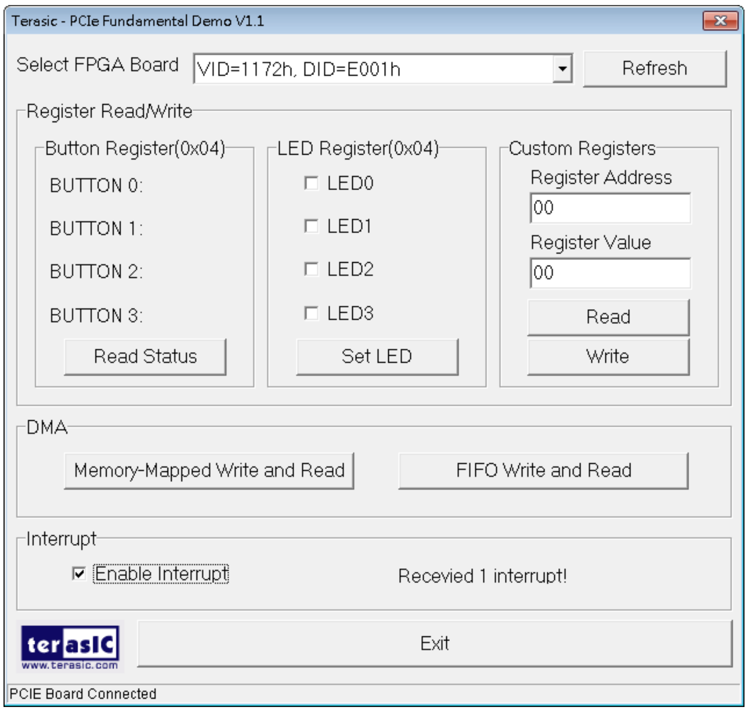


Figure 7-12 Program GUI of PCIe Fundamental Demo

- Make sure the 'Selected FPGA Board' appear as the target board "VID=1172, DID=E001".
- Press Button0-3 on the FPGA board and click the Read Status in this application software.
- Check/Uncheck the LED0-3 in this application software, then click 'Set LED'. The LED in the FPGA board will change.
- Click 'Memory-Mapped Write and Read' to test memory –mapped DMA. A report dialog will appear when the DMA process is completed.
- Click 'FIFO Write and Read' to test FIFO DMA. A report dialog box will appear when the DMA process is completed.
- The 'Custom Registers Group' is used to test custom design register on the FPGA side. Users can use this function to verify custom register design.
- Check "Enable Interrupt" to enable interrupt accordingly. Press Button0-3 on the FPGA board to trigger interrupt event and the count will be displayed on the program UI. Uncheck and Check the "Enable Interrupt" can reset the interrupt counter.

Demonstration Setup

- Quartus II 12.0
- Borland C++ Builder 6.0

Demonstration Source Code Location

- Quartus Project: PCIE_Fundamental
- Borland C++ Project: PCIe_SW_KIT/PCIe_Fundamental

■ FPGA Application Design

Figure 7-13 shows the block diagram of the PCIe Fundamental demostrtion. The PCI Express demonstration uses the basic I/O interface and DMA channel on the Terasic PCIe IP to control I/O (Button/LED) and access two internal memories (RAM/FIFO) through the MUX block. Interrupt function also is included in this demonstration. Buttons on the FPGA board are used to trigger interrupt signals which is detected by Edge Detector IP. When the Edge Detect IP senses buttons are pressed, it will generate a pulse to Terasic PCIe IP.

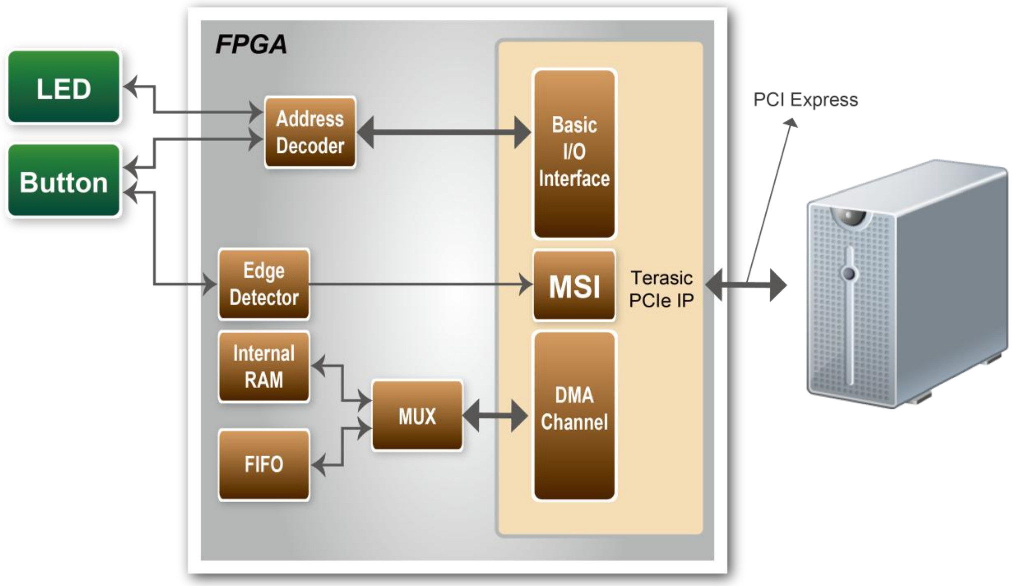
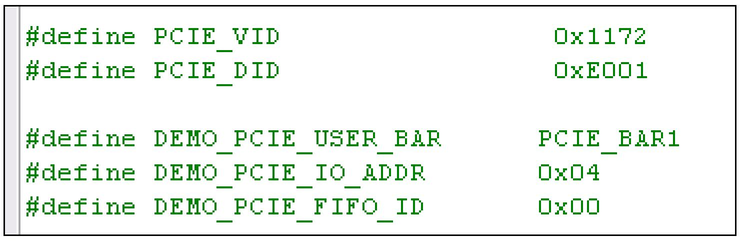


Figure 7-13 Hardware block diagram of the PCIe reference design

■ PC Application Design

The application shows how to call the TERASIC_PCIE.DLL(TERAISC_PCIEx64.DLL under 64-bits Windows) exported API. To enumerate all PCIe cards in system call, the software design defines some constant based on FPGA design shown below:



```
#define PCIE_VID                    0x1172
#define PCIE_DID                    0xE001

#define DEMO_PCIE_USER_BAR        PCIE_BAR1
#define DEMO_PCIE_IO_ADDR         0x04
#define DEMO_PCIE_FIFO_ID         0x00
```

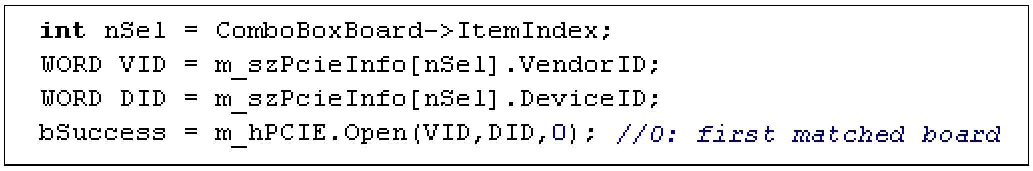
The vender ID is defined as 0x1172 and the device ID is defined as 0xE001. The BUTTON/LED register address is 0x04 based on PCIE_BAR1.

A C++ class **PCIE** is designed to encapsulate the DLL dynamic loading for TERASIC_PCIE.DLL (loading TERAISC_PCIEx64.DLL under 64-bits Windows). A PCIE instance is created with the name **m_hPCIE**. To enumerate all PCIe cards in system, call the function

```
m_hPCIE.ScanCard(wVendorID, wDeviceID, &dwDeviceNum, m_szPcieInfo);
```

where wVendorID and wDeviceID are zeros. The return value dwDeviceNum represents the number of PCIe cards found in the system. The m_szPcieInfo array contains the detail information for each PCIe card.

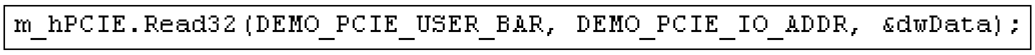
To connect the selected PCIe card, the functions are called:



```
int nSel = ComboBoxBoard->ItemIndex;
WORD VID = m_szPcieInfo[nSel].VendorID;
WORD DID = m_szPcieInfo[nSel].DeviceID;
bSuccess = m_hPCIE.Open(VID,DID,0); //0: first matched board
```

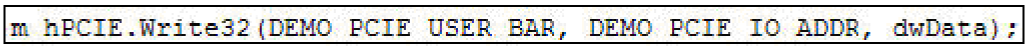
where nSel is selected index in the 'Selected FPGA Board' poll-down menu. Based on the return m_szPcieInfo, we can find the associated PID and DID which can use to specifiy the target PCIe card.

To read the BUTTON status, the function is called:



```
m_hPCIE.Read32(DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_ADDR, &dwData);
```

To set LED status, the function is called:



```
m_hPCIE.Write32(DEMO_PCIE_USER_BAR, DEMO_PCIE_IO_ADDR, dwData);
```

To write and read memory-mapped memory, call the functions:

```
// write
bSuccess = m_hPCIE.DmaWrite(LocalAddr, pWrite, nTestSize);
if (bSuccess){
    // read
    bSuccess = m_hPCIE.DmaRead(LocalAddr, pRead, nTestSize);
}
```

To write and read FIFO memory, call the functions:

```
// write
bSuccess = m_hPCIE.DmaFifoWrite(FifoID, pWrite, nTestSize);
if (bSuccess){
    // read
    bSuccess = m_hPCIE.DmaFifoRead(FifoID, pRead, nTestSize);
}
```

To enable interrupt, call the function: (PCIE_ISR is interrupt service routine).

```
m_hPCIE.IntEnable(PCIE_ISR);
```

The PCIE_ISR is user defined ISR (Interrupt Service Routine). In this demo, the routine is defined as:

```
void PCIE_API PCIE_ISR( PCIE_HANDLE hALTERA, PCIE_INT_RESULT *intResult){
    AnsiString strText;

    strText.printf("Recevied %d interrupt!", intResult->dwCounter);

    FormMain->LabelInterruptReport->Caption = strText;

    // call 'm_hPCIE.IntEnable()' again
    // can clear the interrupt count 'intResult->dwCounter'
}
```

To disable interrupt, call the function:

```
m_hPCIE.IntDisable();
```

7.5 Example 2: Image Process Application

This example shows how to utilize computing power of the FPGA for image processing. The application demonstrates the ‘invert’ image processing by utilizing the FPGA. The PC and FPGA source code of the application layer are all available in the FPGA system CD, allowing users to easily extent the image process function based on this fundamental reference design.

In the demonstration, a memory-mapped memory is designed in the FPGA to work as an image frame buffer. The memory size is 320x240x3 bytes with start address 0x00. The raw image is downloaded to and uploaded from FPGA by DMA. The image process command and status is controlled by a register which can be accessed from the PC by basic IO control. The register address is 0x10 under PCIE BAR1. Writing any value into this register will start the image process. The status of the image process is reported by a read to this register. The PCIe vender ID and device ID is 0x1172 and 0xE001, respectively. The block diagram of FPGA PCIe design is shown in **Figure 7-14**.

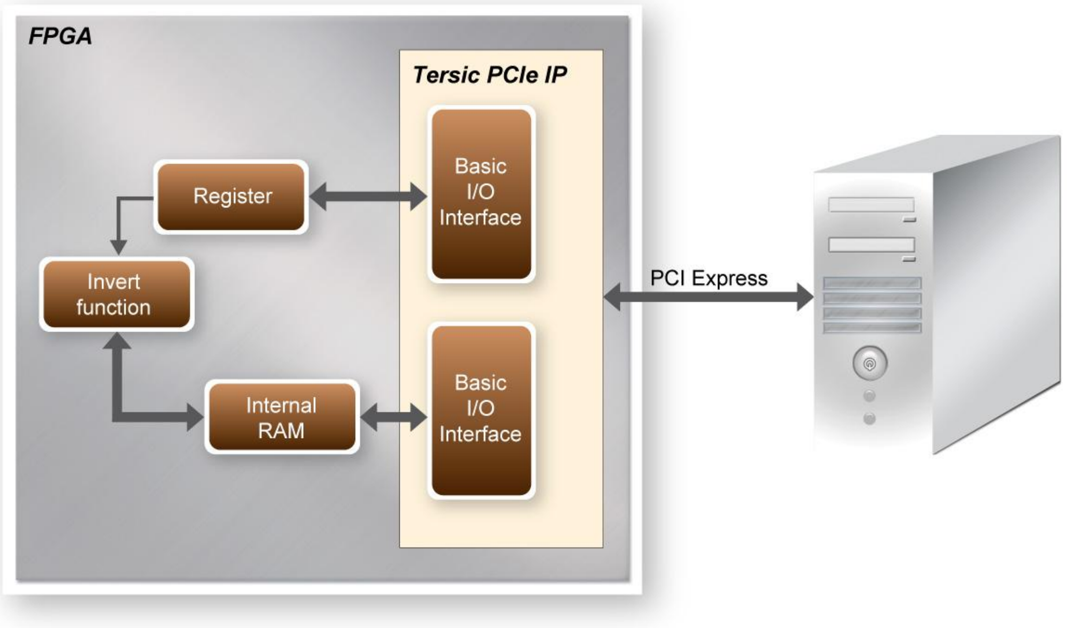


Figure 7-14 Block Diagram of Image Process in FPGA

■ Demonstration Files Location

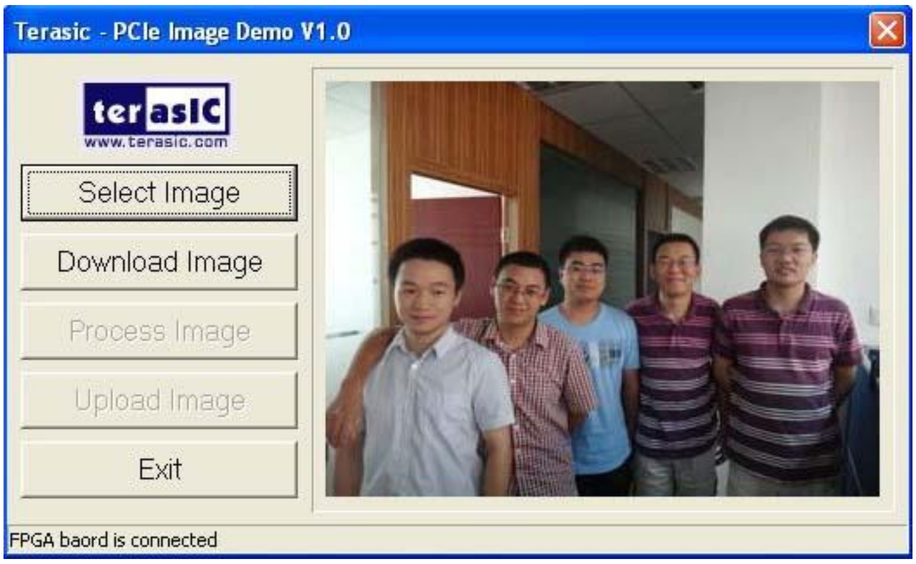
The demo file is located in the folder: *PCIE_ImageProcess\demo_batch*

The folder includes following files:

- PC Application Software: PCIe_Image_Demo.exe
- FPGA Configuration File: PCIe_ImageProcess.sof
- PCIe Library : TERASIC_PCIE.DLL and wapi921.dll (TERASIC_PCIEx64.DLL and wapi1100.dll for 64-bits Widnows)
- Demo Batch File : test_ub2.bat

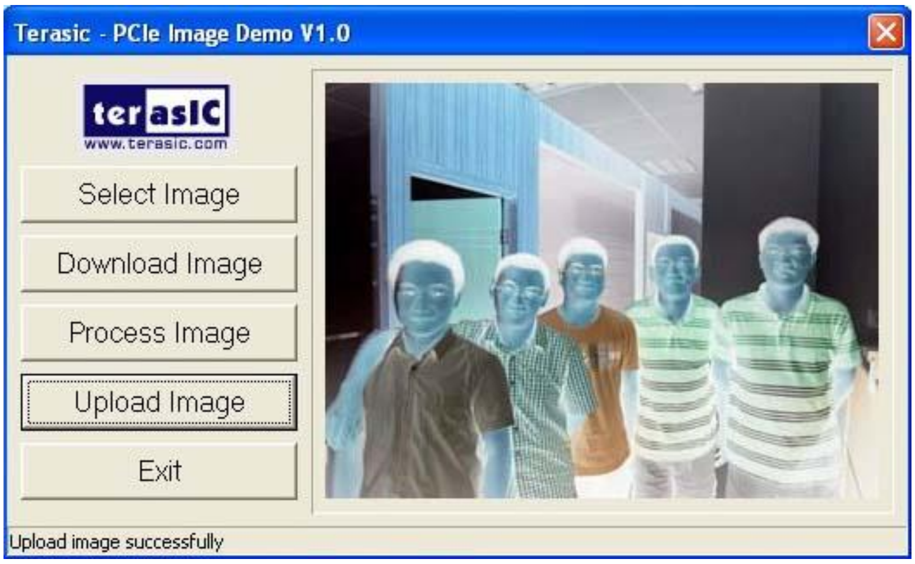
■ Demonstration Setup

- Installed the FPGA board on your PC.
- Locate demo folder: PCIE_ImageProcess\Demo_batch
- Download PCIe_ImageProcess.sof into the FPGA board using Quartus II Programmer.
- Restart Windows.
- Installed PCIe driver if necessary. The driver is located in the folder CDROM\demonstration\PCIe_SW_KIT\PCIe_DriverInstall
- Launch demo program PCIe_Image_Demo.exe
- Click “Select Image” to select a bitmap or jpeg file for image processing.



FPGA baord is connected

- Click “Download Image” to download image raw data into the local memory of FPGA.
- Click “Process Image” to trigger ‘invert’ image process.
- Click “Upload Image” to upload image to PC from local memory of FPGA to be displayed on the window demo application.



Upload image successfully

■ Design Tools

- Quartus II 12.0
- Borland C++ Builder 6.0

■ Demonstration Source Code Location

- Quartus Project: PCIE_ImageProcess
- Borland C++ Project: PCIe_SW_KIT\PCIE_ImageProcess

■ FPGA Application Design

This demonstration uses the DMA channel of PCIe IP to download/upload the image into the internal RAM of FPGA, and controls the user register that switches the function which inverts the image data from the internal RAM.

■ PC Application Design

The software design defines some constant based on FPGA design as shown below:

```
#define PCIE_VID                0x1172
#define PCIE_DID                0xE001

#define IMAGE_WIDTH 320
#define IMAGE_HEIGH 240

#define DEMO_PCIE_USER_BAR      PCIE_BAR1
#define DEMO_IMAGE_REG_ADDR     0x10
#define DEMO_IMAGE_DATA_ADDR    0
```

The vender ID is defined as 0x1172 and the device ID is defined as 0xE001. The image dimension is defined as 320x240. The register address is 0x10 and memory address is 0x00.

A C++ class **PCIE** is designed to encapsulate the DLL dynamic loading for TERASIC_PCIE.DLL (loading TERAISC_PCIEx64.DLL under 64-bits Windows). A PCIE instance is created with the name **m_hPCIE**. To open a connection with FPGA the function is called:

```
m_hPCIE.Open(PCIE_VID,PCIE_DID,0); //0: first matched board
```

To download the raw image from PC to FPGA memory, the function is called:

```
m_hPCIE.DmaWrite(DEMO_IMAGE_DATA_ADDR, pImage, nImageSize);
```

where pImage is a pointer of the image raw data, and the nImageSize specifies the image size. In this reference design, nImageSize = 320x240x3 byte.

To start the image process, the function is called:

```
m_hPCIE.Write32(DEMO_PCIE_USER_BAR, DEMO_IMAGE_REG_ADDR, 1);
```

The image process is started whenever the register is written with any value.

To check whether the image process is finished, the control register is monitored by calling the function:

```
m_hPCIE.Read32(DEMO_PCIE_USER_BAR, DEMO_IMAGE_REG_ADDR, &dwStatus);
```

When the image process is finished, the value of dwStatus becomes zero.

To update the processed image from FPGA memory to PC, the function is called:

```
m_hPCIE.DmaRead(DEMO_IMAGE_DATA_ADDR, pImage, nImageSize);
```

Chapter 8

Transceiver Verification

This chapter describes how to verify the FPGA transceivers using the test code provided in the DE5-Net system CD.

8.1 Test Code

The transceiver test code verifies the transceiver channels through an external loopback method. The following transceiver channels can be verified with different data rates:

- 10.3125 Gbps: SPF-A, SPF-B, SPF-C and SPF-D
- 6.0 Gbps: SATA Host-0, SATA Host-1, SATA Device-0, and SATA Device-1

Gbps: PCIe Channel 0~7

8.2 Loopback Fixture

To enable an external loopback of transceiver channels, specific loopback fixtures are required. Some fixtures may be proprietary to Terasic.

For SFP+ loopback, optical SFP+ loopback fixtures are required. **Figure 8-1** shows the optical SFP+ loopback fixture.

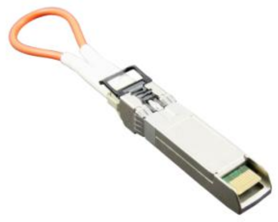


Figure 8-1 Optical SFP+ Loopback Fixture

.

Figure 8-2 **shows the SATA loopback fixture.**

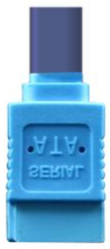


Figure 8-2 SATA Loopback Fixture

Figure 8-3 shows the Terasic PCIe loopback fixture.

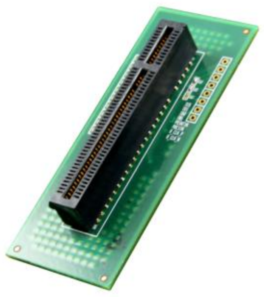


Figure 8-3 PCIe Loopback Fixture

Figure 8-4 shows the FPGA board with all transceiver loopback fixtures installed.

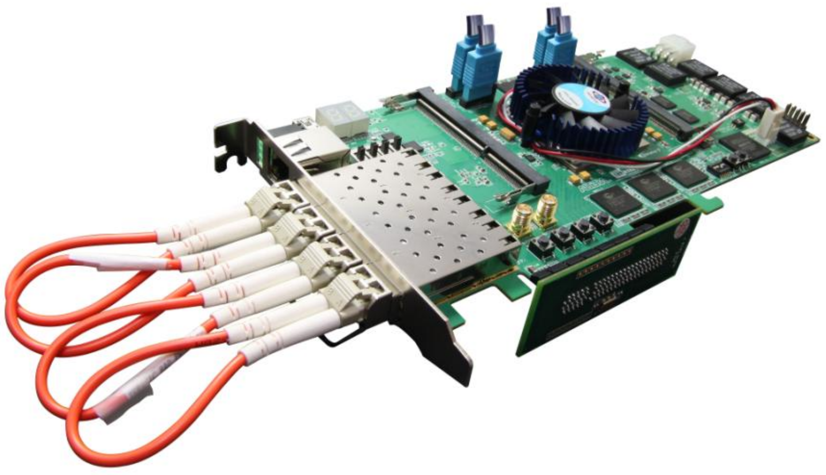
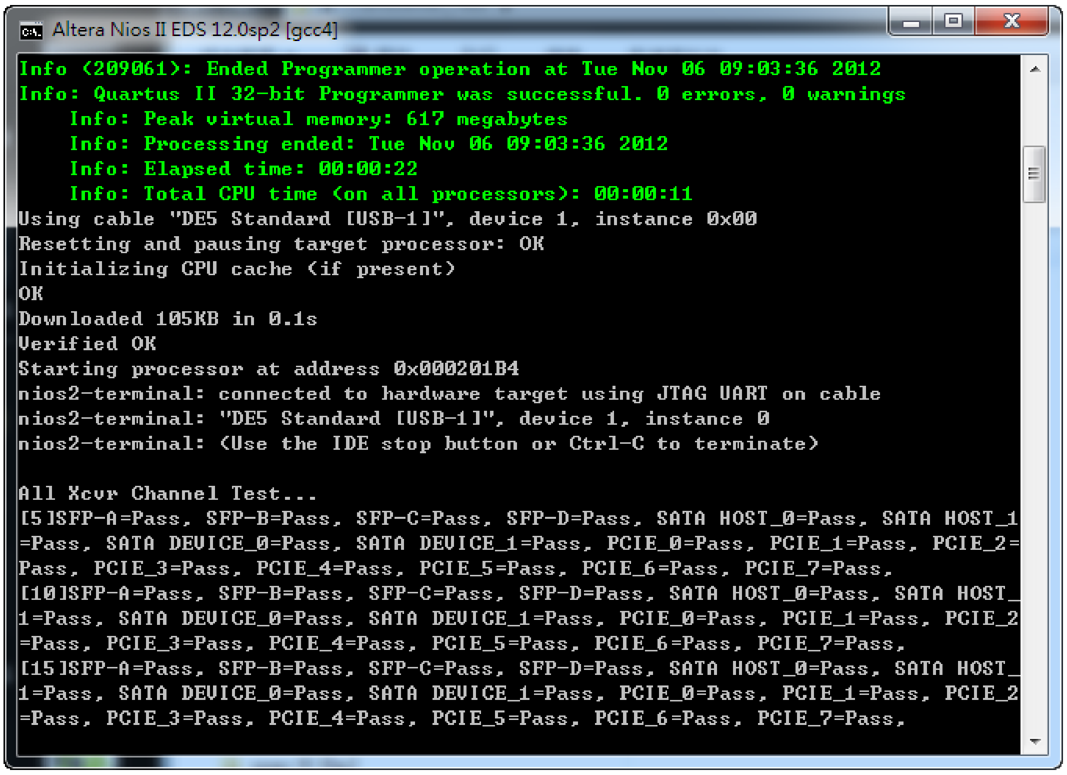


Figure 8-4 Transceiver Loopback Fixtures Installed

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:

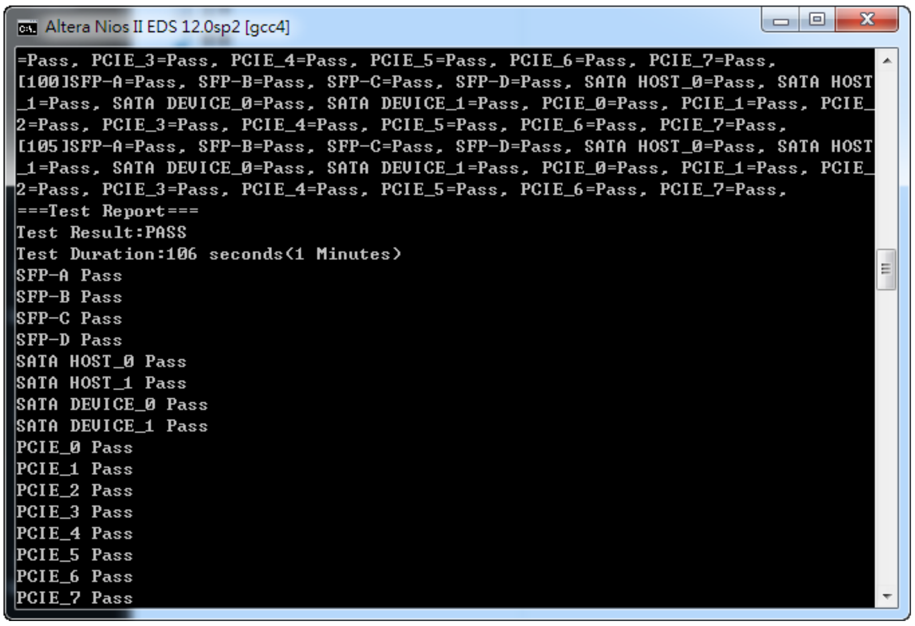
1. Copy **Transceiver_Test** folder to your local disk.
2. Ensure that the FPGA board is NOT powered on.
3. Plug-in the SPF+ loopback fixtures if the SPF+ transceivers will be tested.
4. Plug-in the SATA loopback fixtures if the SATA transceivers will be tested.
5. Plug-in the PCIe loopback fixture if PCIe transceivers will be tested. Also, make sure PCIe Mode SW7 is switched to x8 mode.
6. Connect your FPGA board to your PC with an mini USB cable.
7. Power on the FPGA board
8. Execute 'xcvr_test.bat" in the **Transceiver_Test** folder under your local disk.
9. 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.**
10. To terminate the test, press one of the BUTTON0~3 buttons on the FPGA board. The loopback test will terminate, and the test summary will be shown in the Nios-Terminal, as shown in **Figure 8-6**.



```
Info (209061): Ended Programmer operation at Tue Nov 06 09:03:36 2012
Info: Quartus II 32-bit Programmer was successful. 0 errors, 0 warnings
    Info: Peak virtual memory: 617 megabytes
    Info: Processing ended: Tue Nov 06 09:03:36 2012
    Info: Elapsed time: 00:00:22
    Info: Total CPU time (on all processors): 00:00:11
Using cable "DE5 Standard [USB-1]", device 1, instance 0x00
Resetting and pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 105KB in 0.1s
Verified OK
Starting processor at address 0x000201B4
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "DE5 Standard [USB-1]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

All Xcvr Channel Test...
[5 ]SFP-A=Pass, SFP-B=Pass, SFP-C=Pass, SFP-D=Pass, SATA HOST_0=Pass, SATA HOST_1
=Pass, SATA DEVICE_0=Pass, SATA DEVICE_1=Pass, PCIE_0=Pass, PCIE_1=Pass, PCIE_2=
Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
[10]SFP-A=Pass, SFP-B=Pass, SFP-C=Pass, SFP-D=Pass, SATA HOST_0=Pass, SATA HOST_
1=Pass, SATA DEVICE_0=Pass, SATA DEVICE_1=Pass, PCIE_0=Pass, PCIE_1=Pass, PCIE_2
=Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
[15]SFP-A=Pass, SFP-B=Pass, SFP-C=Pass, SFP-D=Pass, SATA HOST_0=Pass, SATA HOST_
1=Pass, SATA DEVICE_0=Pass, SATA DEVICE_1=Pass, PCIE_0=Pass, PCIE_1=Pass, PCIE_2
=Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
```

Figure 8-5　Transceiver Loopback Test in Progress



```
=Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
[100]SFP-A=Pass, SFP-B=Pass, SFP-C=Pass, SFP-D=Pass, SATA HOST_0=Pass, SATA HOST
_1=Pass, SATA DEVICE_0=Pass, SATA DEVICE_1=Pass, PCIE_0=Pass, PCIE_1=Pass, PCIE_
2=Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
[105]SFP-A=Pass, SFP-B=Pass, SFP-C=Pass, SFP-D=Pass, SATA HOST_0=Pass, SATA HOST
_1=Pass, SATA DEVICE_0=Pass, SATA DEVICE_1=Pass, PCIE_0=Pass, PCIE_1=Pass, PCIE_
2=Pass, PCIE_3=Pass, PCIE_4=Pass, PCIE_5=Pass, PCIE_6=Pass, PCIE_7=Pass,
===Test Report===
Test Result:PASS
Test Duration:106 seconds(1 Minutes)
SFP-A Pass
SFP-B Pass
SFP-C Pass
SFP-D Pass
SATA HOST_0 Pass
SATA HOST_1 Pass
SATA DEVICE_0 Pass
SATA DEVICE_1 Pass
PCIE_0 Pass
PCIE_1 Pass
PCIE_2 Pass
PCIE_3 Pass
PCIE_4 Pass
PCIE_5 Pass
PCIE_6 Pass
PCIE_7 Pass
```

Figure 8-6　Transceiver Loopback Test Result Summary

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, 30070. Taiwan, 30070

  Email: support@terasic.com

  Web: www.terasic.com

  DE5-Net Web: DE5-Net.terasic.com

Revision History

Date	Version	Changes
2012.6	First publication	
2012.11	V1.01	Update si570 parameter
2014.10	V1.02	Update section 5.2 for modifying si570 function
2015.12	V1.03	Add interrupt function for PCIe examples