MAX 10 NEEK

FPGA Development Kit

User Manual

terasIC
www.terasic.com

ALTERA

CONTENTS

Chapter 1

MAX 10 NEEK Development Kit

The MAX 10 NEEK from Terasic is a full featured embedded evaluation kit based upon the MAX10 family of Altera FPGAs. It offers a comprehensive design environment with everything embedded developers need to create a processing based system. The MAX 10 NEEK delivers an integrated platform that includes hardware, design tools, intellectual property and reference designs for developing a wide range of audio, video and many other exciting applications.

The fully integrated kit allows developers to rapidly customize their processor and IP to suit their specific needs, rather than constraining their software around the fixed feature set of the processor. The all-in-one embedded solution, the MAX 10 NEEK, combines a 5-point LCD touch panel and digital image module that provides developers an ideal platform for multimedia applications, making the best use of the parallel nature of FPGAs.

1.1 Package Contents

Figure 1-1 shows a photograph of the MAX 10 NEEK package.

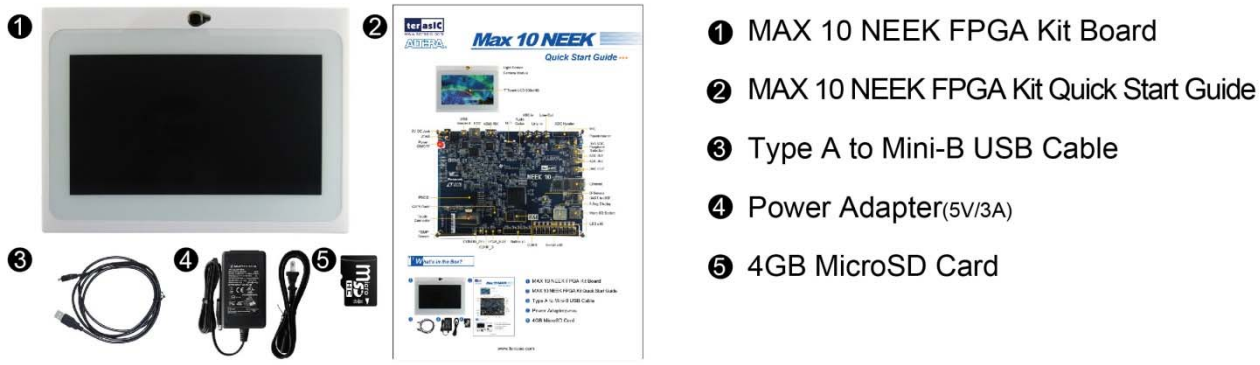

Figure 1-1 The MAX 10 NEEK package contents

The MAX 10 NEEK package includes:

- The MAX 10 NEEK development board
- MAX 10 NEEK Quick Start Guide
- One USB cables (Type A to Mini-B) for USB control and FPGA programming and control
- 5V DC power adapter
- Power Cable

1.2 MAX 10 NEEK System CD

The MAX 10 NEEK System CD contains all the documents and supporting materials associated with MAX 10 NEEK, including the user manual, system builder, reference designs, and device datasheets. Users can download this system CD from the link: http://cd-max10-neek.terasic.com.

1.3 Getting Help

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

- Altera Corporation
- 101 Innovation Drive San Jose, California, 95134 USA

Email: university@altera.com

- Terasic Technologies
- 9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan

Email: support@terasic.com

Tel.: +886-3-575-0880

Website: max10-neek.terasic.com

Chapter 2

Introduction of the MAX 10 NEEK Board

2.1 Layout and Components

Figure 2-1 shows a photograph of the board. It depicts the layout of the board and indicates the location of the connectors and key components.

Figure 2-1 MAX 10 NEEK development board (top view)

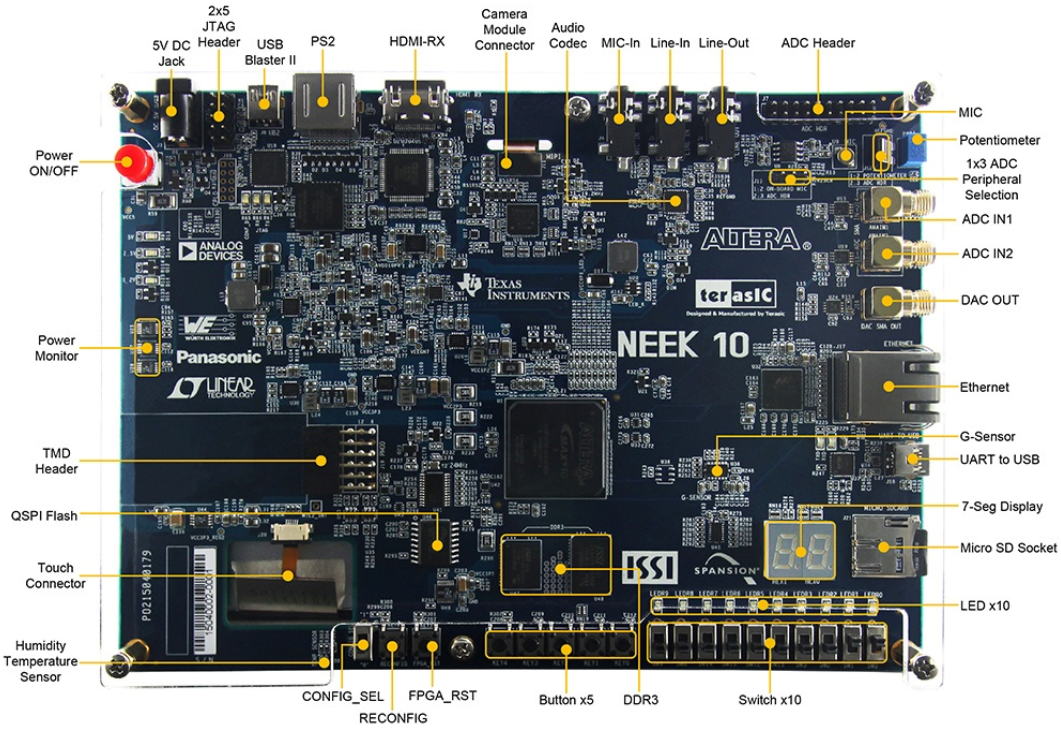

Figure 2-2 MAX 10 NEEK development board (bottom view)

The MAX 10 NEEK board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the board:

- Altera MAX® 10 10M50DAF484C6G device
- USB-Blaster II onboard for programming; JTAG Mode
- 256MB DDR3 SDRAM (64Mx16 and 128Mx8)
- 64MB QSPI Flash
- Micro SD card socket
- Five push-buttons
- Ten slide switches
- Ten red user LEDs
- Two 7-segment displays
- Three 50MHz clock sources from the clock generator
- 24-bit CD-quality audio CODEC with line-in, line-out jacks
- On-board microphone
- HDMI RX, incorporates HDM v1.4a features, including 3D video supporting
- 800x480 7.0 inch Color LCD with 5-point Capacitive-touch
- Gigabit Ethernet PHY with RJ45 connector
- UART to USB, USB Mini-B connector
- 8M pixel MIPI CS2 color camera input

- One ambient light sensor
- One humidity and temperature sensor
- One accelerometer
- One external 16 bit digital-to-analog converter (DAC) device with SMA output
- Potentiometer input to ADC
- Two MAX 10 FPGA ADC SMA inputs
- One 2x10 ADC header with 16 analog inputs connected to MAX10 ADCs

2.2 Block Diagram of the MAX 10 NEEK Board

Figure 2-3 is the block diagram of the board. All the connections are established through the MAX 10 FPGA device to provide maximum flexibility for users. Users can configure the FPGA to implement any system design.

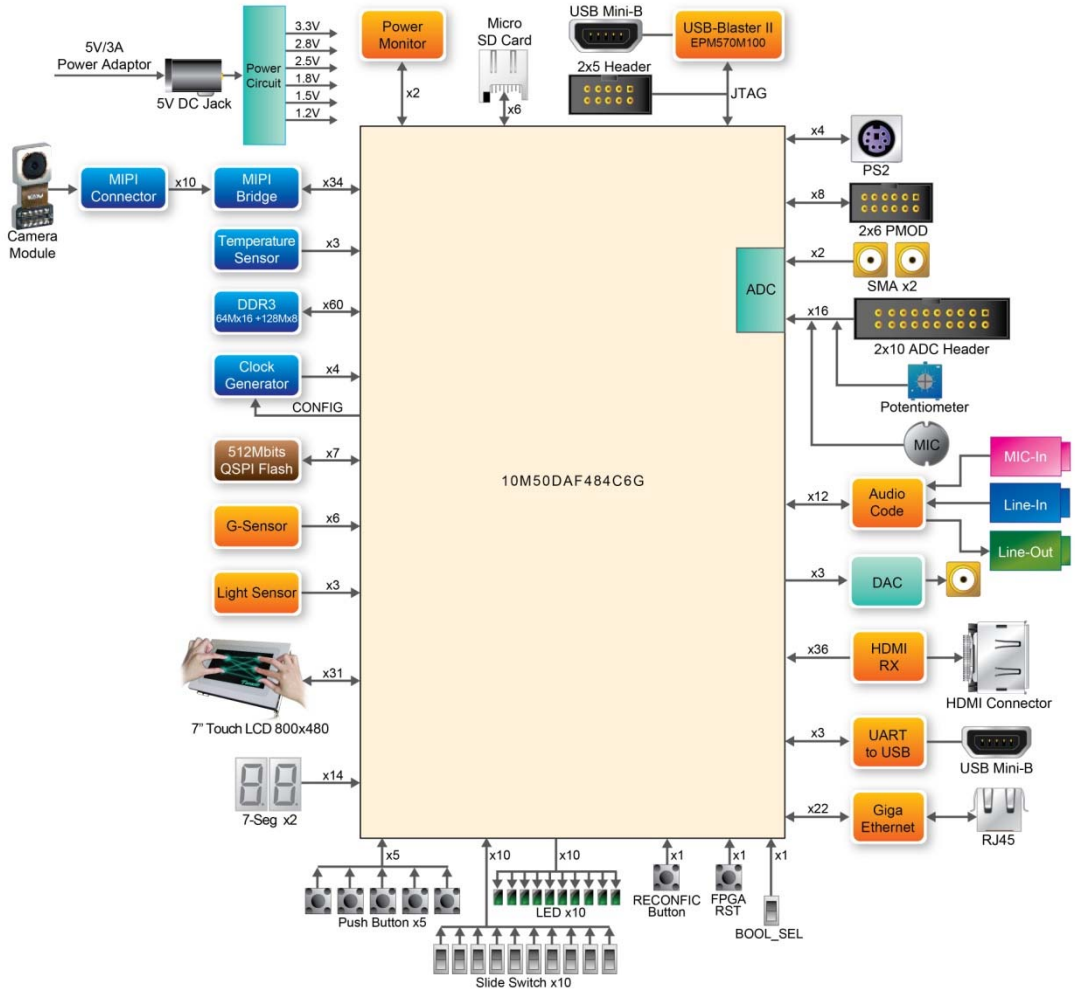

Figure 2-3 Block diagram of MAX 10 NEEK

FPGA Device

- MAX 10 10M50DAF484C6G Device
- Integrated dual ADCs, each ADC supports 1 dedicated analog input and 8 dual function pins
- 50K programmable logic elements
- 1,638 Kbits embedded memory
- 5,888 Kbits user flash memory
- 4 PLLs

Configuration and Debug

- On-board USB-Blaster II (mini USB type B connector)
- Optional JTAG direct via 10-pin header
- One slide switch for dual boot image selection

Memory Device

- 256MB DDR3 SDRAM (64Mx16 and 128Mx8)
- 512Mb QSPI Flash
- Micro SD card socket

Communication and Expansion Header

- Gigabit Ethernet PHY with RJ45 connector
- UART to USB, USB Mini-B connector
- PS/2 mouse/keyboard connector
- 2x6 TMD (Terasic Mini Digital) Expansion Header

Display

- 800x480 7.0 inch Color LCD with 5-point Capacitive-touch

Audio

- 24-bit CD-quality audio CODEC with line-in, line-out jacks

Video Input

- HDMI RX, incorporates HDM v1.4a features, including 3D video supporting
- 8M pixel MIPI CS2 color camera input

Analog

- Two MAX 10 FPGA ADC SMA inputs
- Potentiometer input to ADC
- On-Board MIC input to ADC
- 2x10 ADC header with 16 analog inputs connected to MAX10 ADCs
- One DAC SMA output

Switches, Buttons, and Indicators

- Five push-buttons
- Ten slide switches
- Ten red user LEDs
- Two 7-segment displays

Sensors

- Ambient light sensor
- Humidity and temperature sensor
- Accelerometer
- Power monitor

Power

- 5V/3A DC input

Chapter 3

Using the MAX 10 NEEK Board

This chapter provides an instruction to use the board and describes the peripherals.

3.1 Configuration of MAX 10 FPGA on MAX 10 NEEK

There are two types of configuration method supported by MAX 10 NEEK:

1. JTAG configuration: configuration using JTAG ports.

JTAG configuration scheme allows you to directly configure the device core through JTAG pins - TDI, TDO, TMS, and TCK pins. The Quartus II software automatically generates .sof that are used for JTAG configuration with a download cable in the Quartus II software programmer..

2. Internal configuration: configuration using internal flash.

Before internal configuration, you need to program the configuration data into the configuration flash memory (CFM) which provides non-volatile storage for the bit stream. The information is retained within CFM even if the MAX 10 NEEK board is turned off. When the board is powered on, the configuration data in the CFM is automatically loaded into the MAX 10 FPGA.

■ JTAG Chain on MAX 10 NEEK Board

The FPGA device can be configured through JTAG interface on MAX 10 NEEK board, but the JTAG chain must form a closed loop, which allows Quartus II programmer to the detect FPGA device. **Figure 3-1** illustrates the JTAG chain on MAX 10 NEEK board

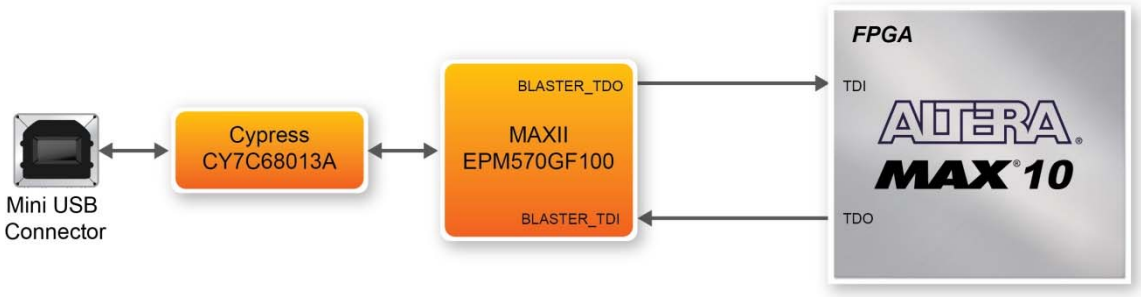

Figure 3-1 Path of the JTAG chain

■ Configure the FPGA in JTAG Mode

The following shows how the FPGA is programmed in JTAG mode step by step.

1. Open the Quartus II programmer and click "Auto Detect", as circled in **Figure 3-2**

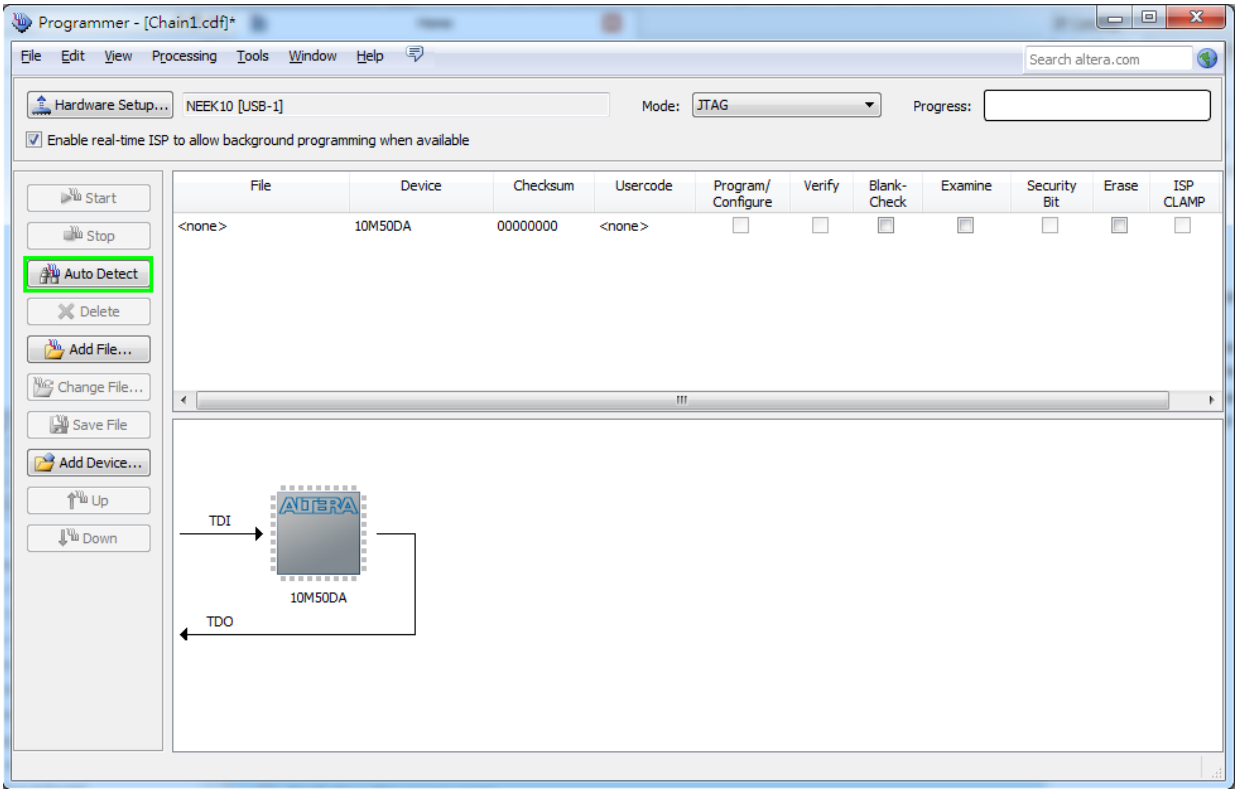

Figure 3-2 Detect FPGA device in JTAG mode

2. Select detected device associated with the board, as circled in **Figure 3-3**.

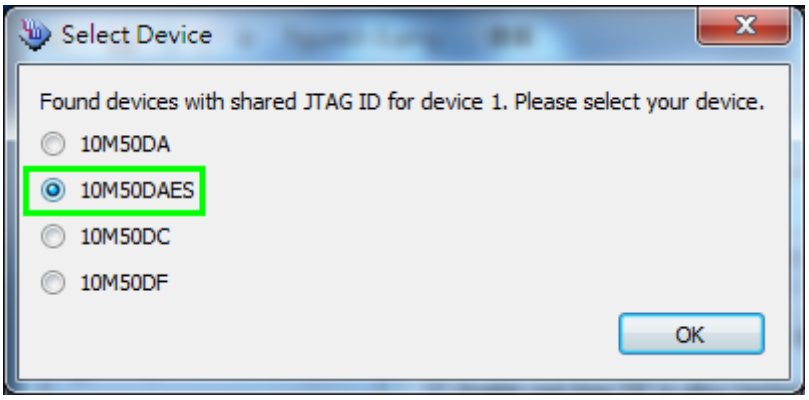

Figure 3-3 Select 10M50DAES device

3. FPGA is detected, as shown in **Figure 3-4**.

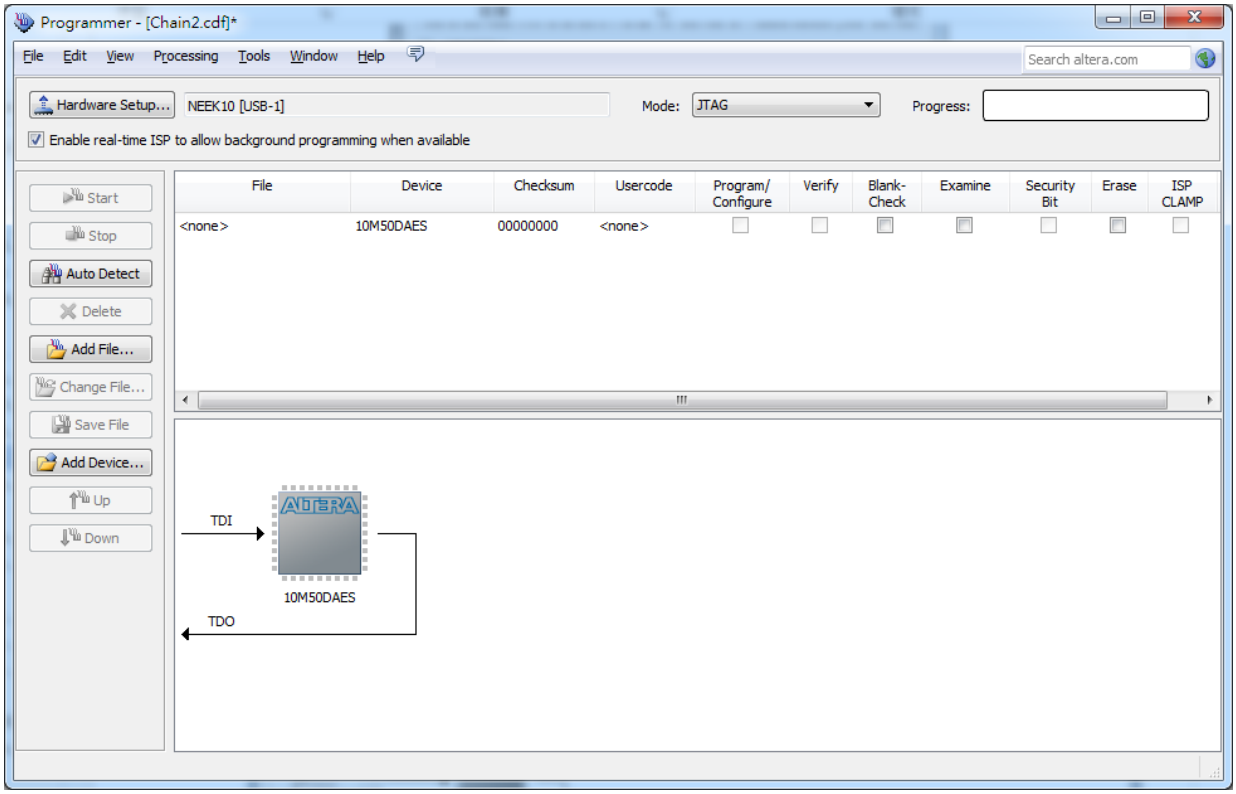

Figure 3-4 FPGA detected in Quartus programmer

4. Right click on the FPGA device and open the .sof file to be programmed, as highlighted in **Figure 3-5**

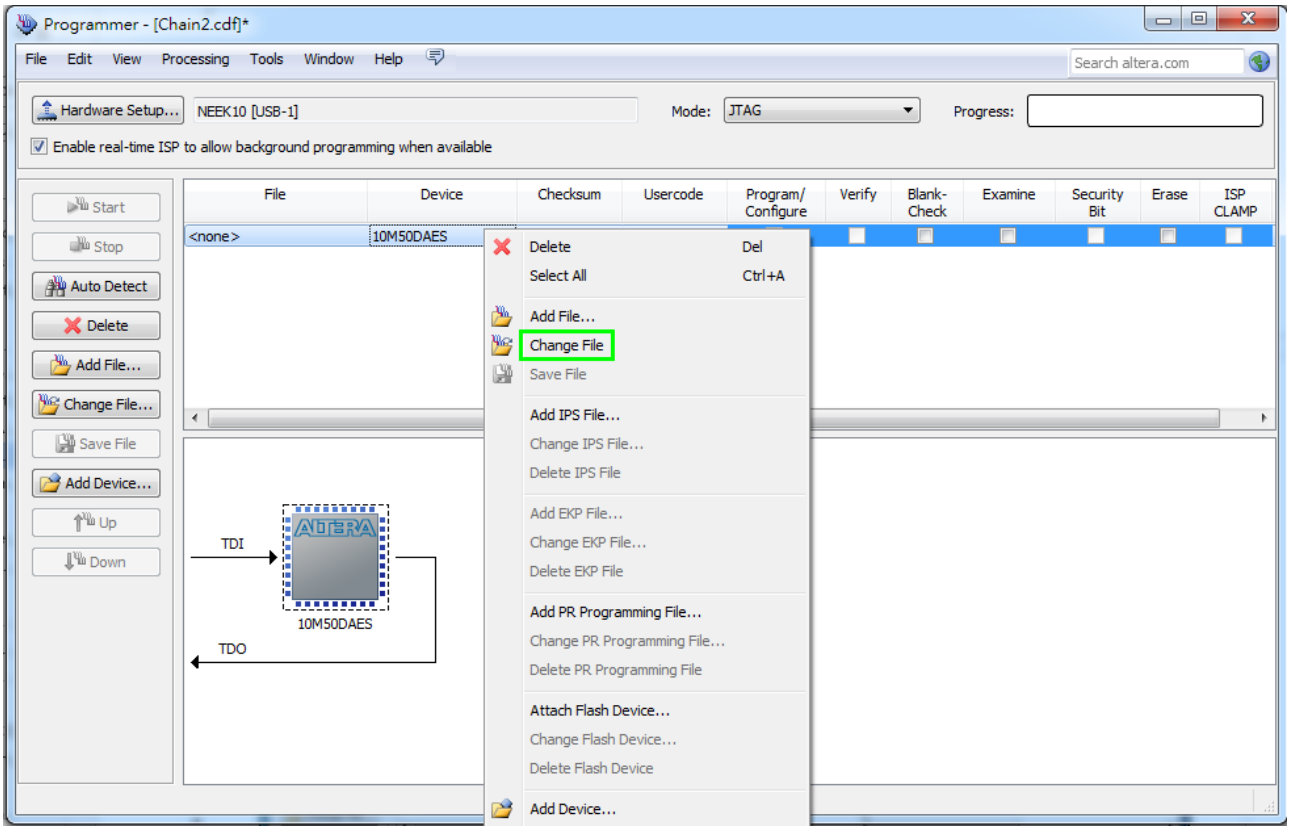

Figure 3-5 Open the .sof file to be programmed into the FPGA device

5. Select the .sof file to be programmed, as shown in **Figure 3-6**.

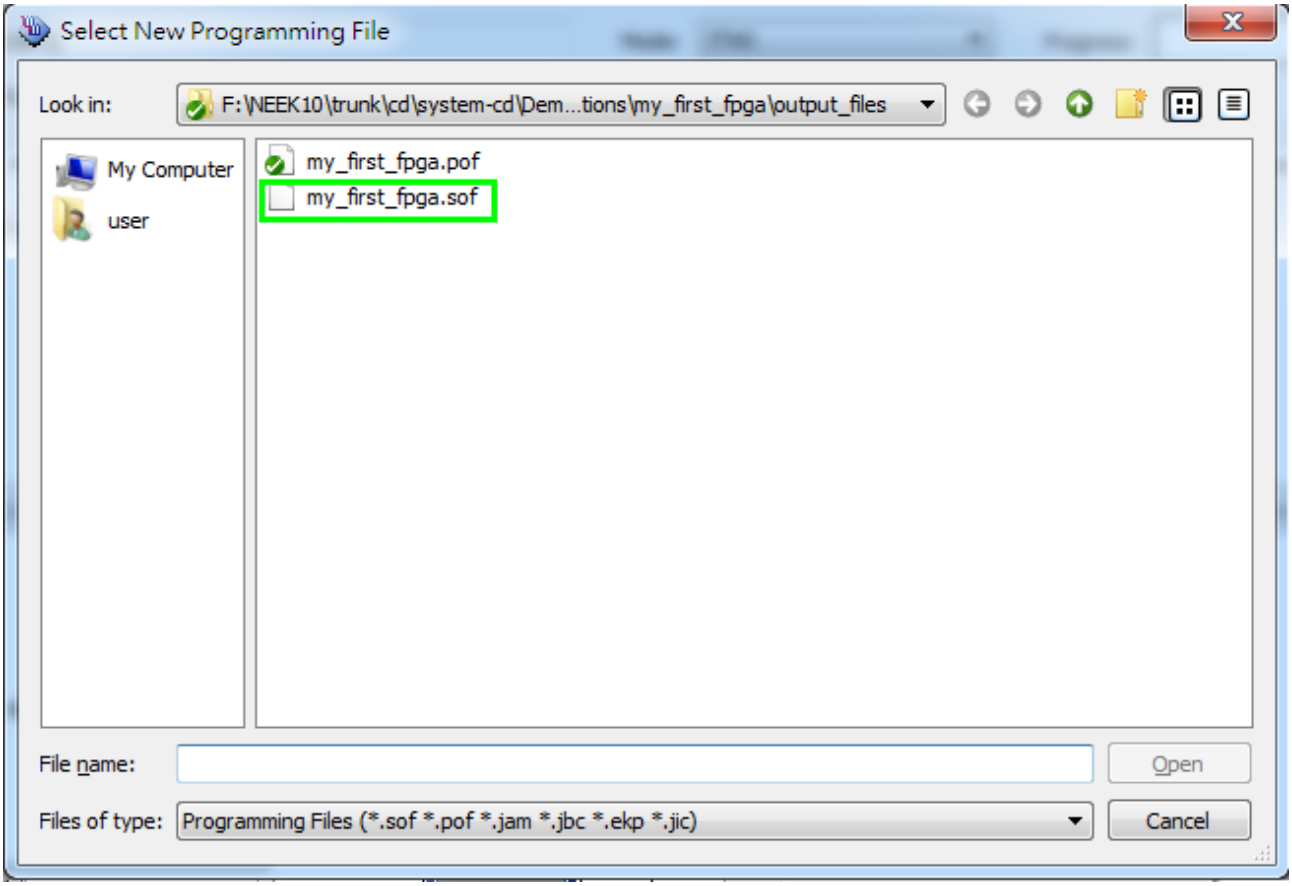

Figure 3-6 Select the .sof file to be programmed into the FPGA device

6. Click "Program/Configure" check box and then click "Start" button to download the .sof file into the FPGA device, as shown in **Figure 3-7**.

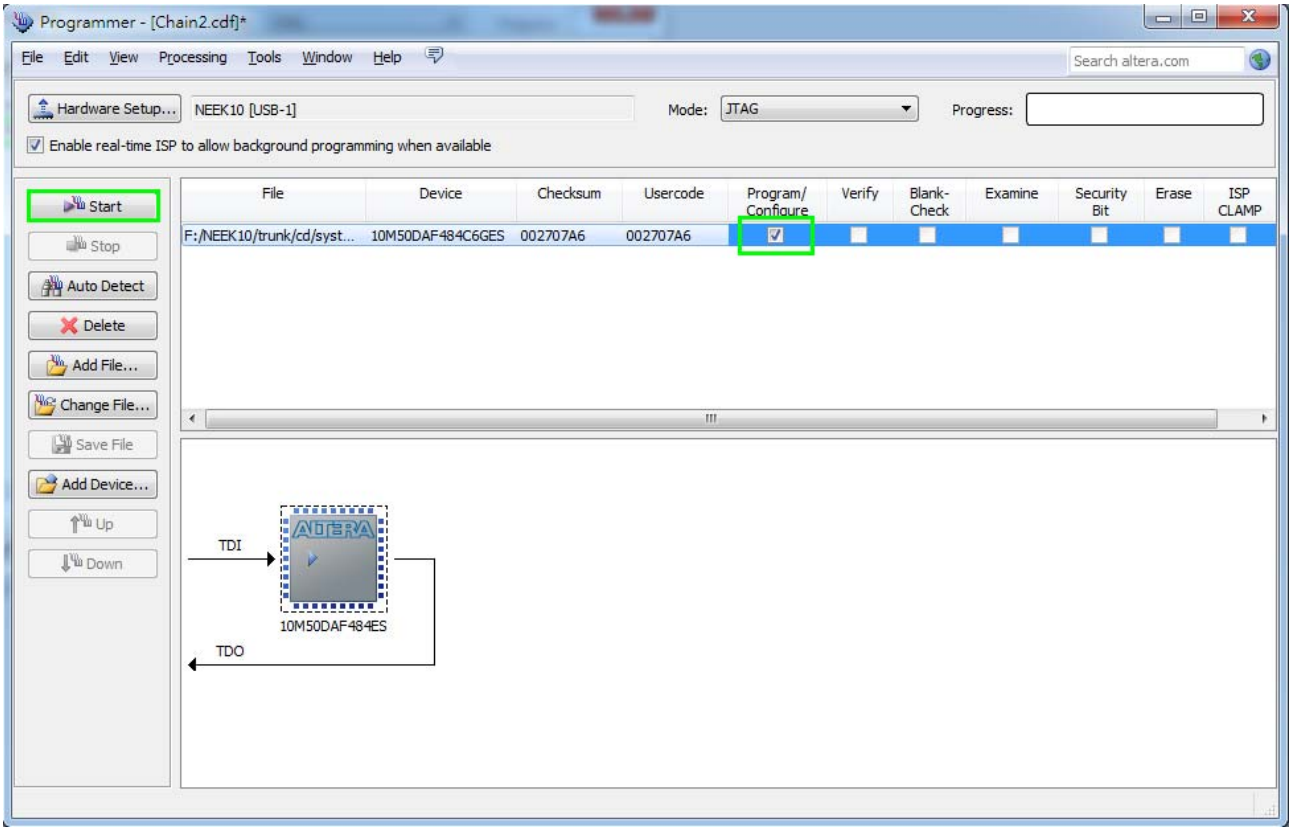

Figure 3-7 Program .sof file into the FPGA device

■ Internal Configuration

- The configuration data to be written to CFM will be part of the programmer object file (.pof). This configuration data is automatically loaded from the CFM into the MAX 10 devices when the board is powered up.
- Please refer to Chapter 8: Programming the Configuration Flash Memory (CFM) for the basic programming instruction on the configuration flash memory (CFM).

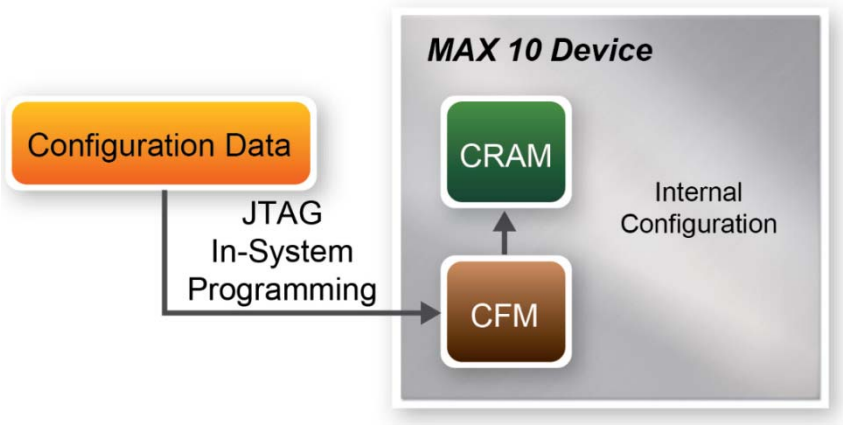

Figure 3-8 High-Level Overview of Internal Configuration for MAX 10 Devices

3.2 Board Status Elements

In addition to the 10 LEDs that FPGA device can control, there are 4 indicators which can indicate the board status (See **Figure 3-9)**, please refer the details in **Table 3-1**

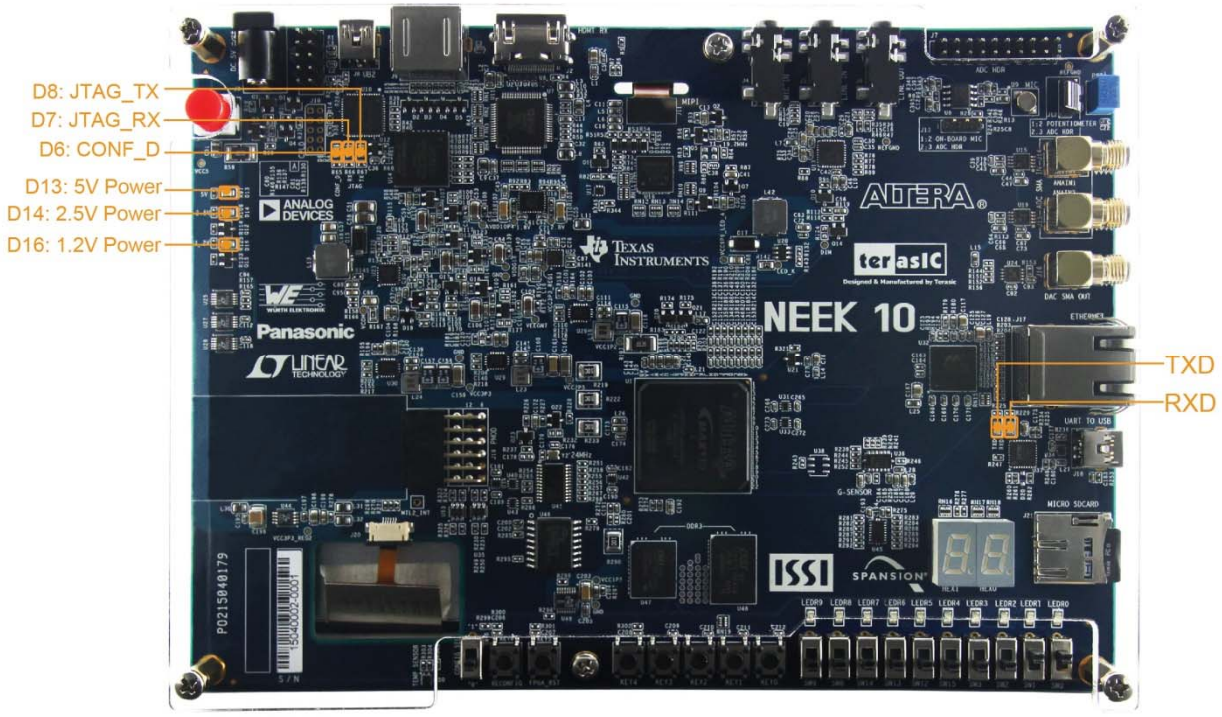

Figure 3-9 LED Indicators on MAX 10 NEEK

Table 3-1 LED Indicators

Board Reference	*LED Name*	*Description*
D13	5V Power	Illuminate when 5V power is active.
D14	2.5V Power	Illuminate when 2.5V power is active.
D16	1.2V Power	Illuminate when 1.2V power is active.
D6	CONF_DONE	Illuminate when configuration data is loaded into MAX 10 device without error.
D7	JTAG_RX	Illuminate during data is uploaded from MAX 10 device to PC through UB2.
D8	JTAG_TX	Illuminate during configuration data is loaded into MAX 10 device from UB2.
TXD	TXD	Illuminate during transmitting data via USB.
RXD	RXD	Illuminate during receiving data via USB.

3.3 Clock Circuitry

Figure 3-10 shows the default frequency of all external clocks to the MAX 10 FPGA. A clock generator is used to distribute clock signals with low jitter. The three 50MHz clock signals connected to the FPGA are used as clock sources for user logic. One 25MHz clock signal is connected to the clock input of Gigabit Ethernet Transceiver. One 24MHz clock signal is connected to the clock inputs of USB microcontroller of USB Blaster II. One 28.63636MHz clock signal is connected to the clock input of HDMI Receiver chip. The other 50MHz clock signal is connected to MAX CPLD of USB Blaster II. One 10MHz clock signal is connected to the PLL1 and PLL3 of FPGA, the outputs of these two PLLs can drive ADC clock. The associated pin assignment for clock inputs to FPGA I/O pins is listed in **Table 3-2.**

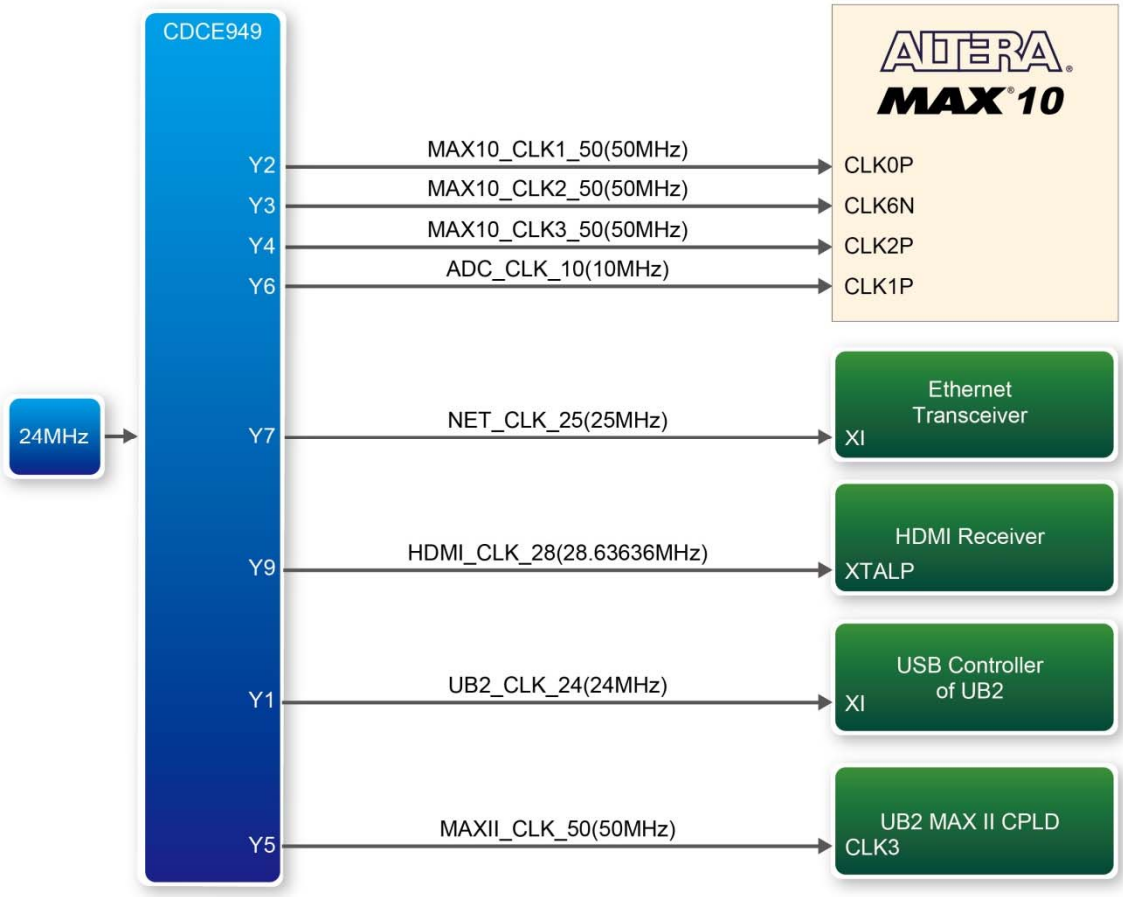

Figure 3-10 Block diagram of the clock distribution on MAX 10 NEEK

Table 3-2 Pin Assignment of Clock Inputs

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MAX10_CLK1_50	PIN_N5	50 MHz clock input	2.5V
MAX10_CLK2_50	PIN_V9	50 MHz clock input	3.3V
MAX10_CLK3_50	PIN_N14	50 MHz clock input	1.5V
ADC_CLK_10	PIN_M9	10 MHz clock input	3.3V

3.4 Peripherals Connected to the FPGA

This section describes the interfaces connected to the FPGA. User can control or monitor different interfaces with user logic from the FPGA.

3.4.1 User Push-buttons, Switches, LEDs

The board has five push-buttons connected to the FPGA, as shown in **Figure 3-11.** MAX 10 devices support Schmitt trigger input on all I/O pins. A Schmitt trigger feature introduces hysteresis to the input signal for improved noise immunity, especially for signal with slow edge rate and act as switch debounce in **Figure 3-12** for the push-buttons connected.

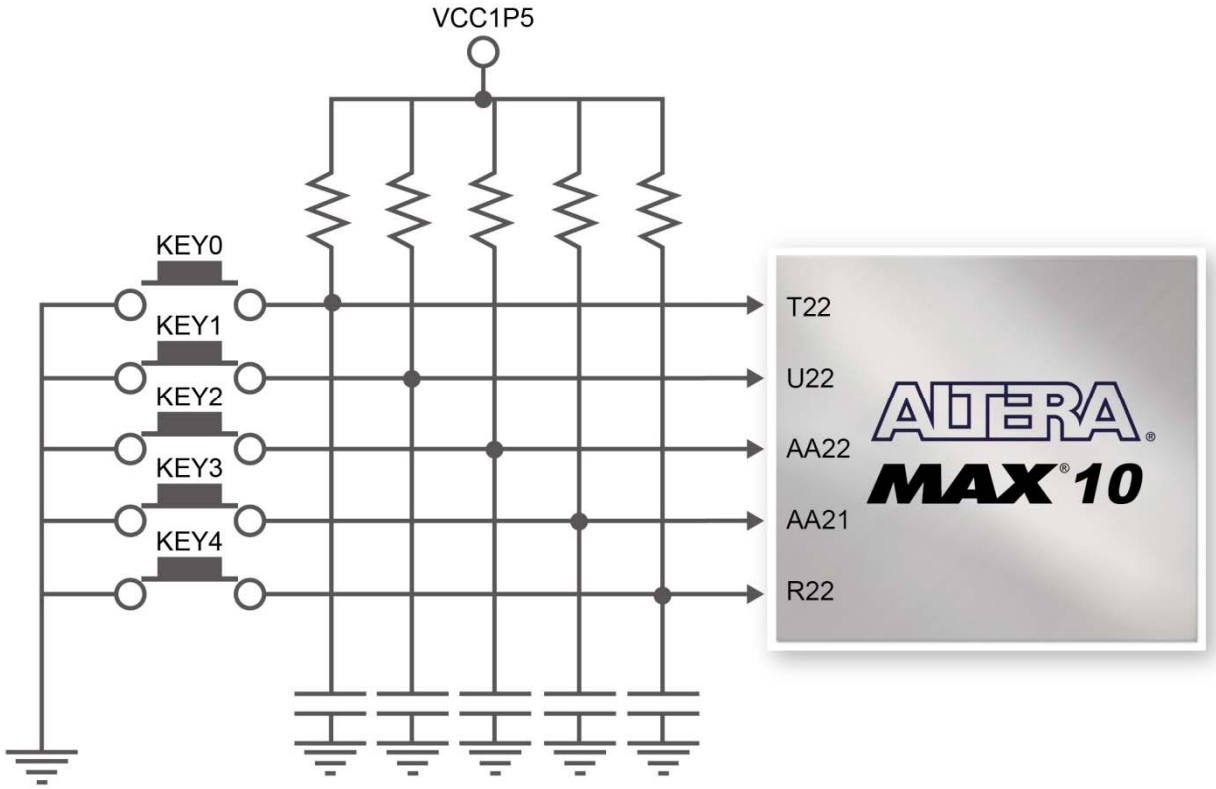

Figure 3-11 Connections between the push-buttons and the MAX 10 FPGA

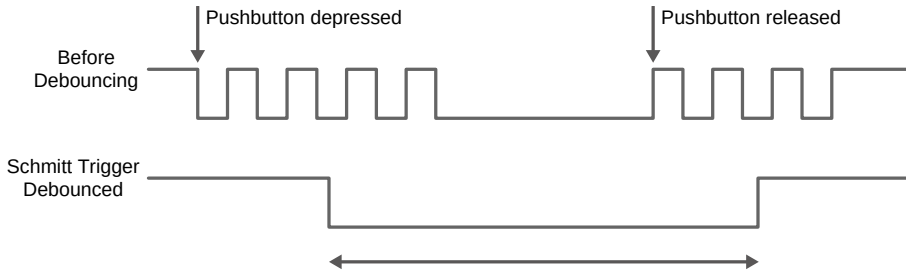

Figure 3-12 Switch debouncing

There are two ten switches connected to the FPGA, as shown in **Figure 3-13**. These switches are used as level-sensitive data inputs to a circuit. Each switch is connected directly and individually to the FPGA. When the switch is set to the DOWN position (towards the edge of the board), it generates a low logic level to the FPGA. When the switch is set to the UP position, a high logic level is generated to the FPGA.

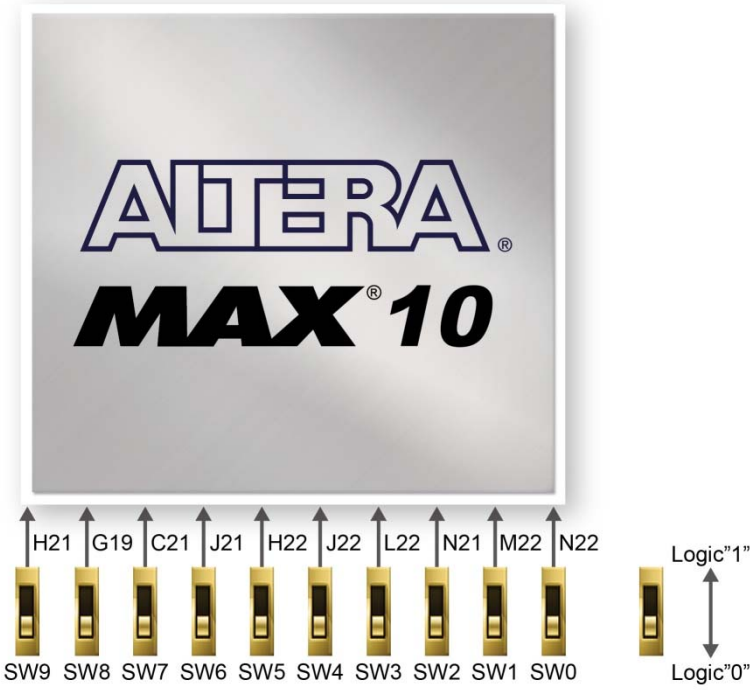

Figure 3-13 Connections between the slide switches and the MAX 10 FPGA

There are also ten user-controllable LEDs connected to the FPGA. Each LED is driven directly and individually by the MAX 10 FPGA; driving its associated pin to a high logic level or low level to turn the LED on or off, respectively. **Figure 3-14** shows the connections between LEDs and MAX 10 FPGA. **Table 3-3**, **Figure 3-14** and **Table 3-9** list the pin assignment of user push-buttons, switches, and LEDs.

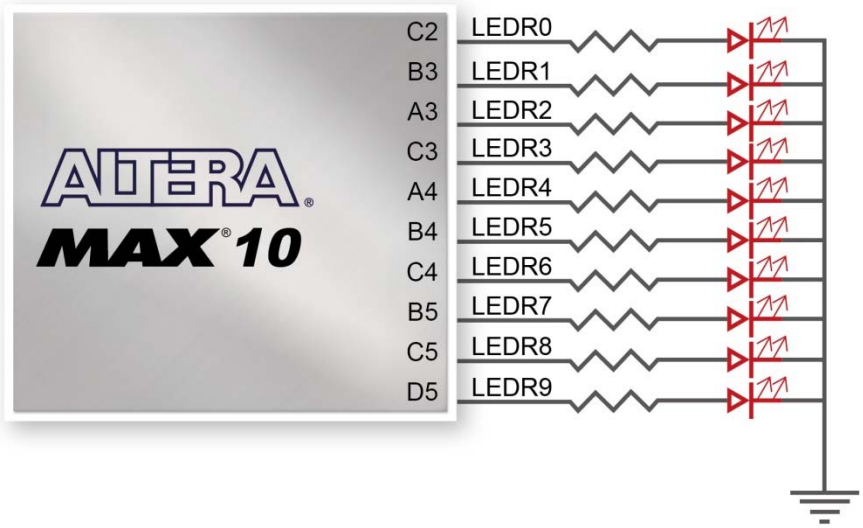

Figure 3-14 Connections between the LEDs and the MAX 10 FPGA

Table 3-3 Pin Assignment of Push-buttons

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
KEY[0]	PIN_T22	Push-button[0]	1.5V
KEY[1]	PIN_U22	Push-button[1]	1.5V
KEY[2]	PIN_AA22	Push-button[2]	1.5V
KEY[3]	PIN_AA21	Push-button[3]	1.5V
KEY[4]	PIN_R22	Push-button[4]	1.5V

Table 3-4 Pin Assignment of Slide Switches

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SW[0]	PIN_N22	Slide Switch[0]	1.5V
SW[1]	PIN_M22	Slide Switch[1]	1.5V
SW[2]	PIN_N21	Slide Switch[2]	1.5V
SW[3]	PIN_L22	Slide Switch[3]	1.5V
SW[4]	PIN_J22	Slide Switch[4]	1.5V
SW[5]	PIN_H22	Slide Switch[5]	1.5V
SW[6]	PIN_J21	Slide Switch[6]	1.5V
SW[7]	PIN_C21	Slide Switch[7]	1.5V
SW[8]	PIN_G19	Slide Switch[8]	1.5V
SW[9]	PIN_H21	Slide Switch[9]	1.5V

Table 3-5 Pin Assignment of LEDs

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
LEDR[0]	PIN_C2	LEDR [0]	3.3V
LEDR[1]	PIN_B3	LEDR [1]	3.3V
LEDR[2]	PIN_A3	LEDR [2]	3.3V

LEDR[3]	PIN_C3	LEDR [3]	3.3V
LEDR[4]	PIN_A4	LEDR [4]	3.3V
LEDR[5]	PIN_B4	LEDR [5]	3.3V
LEDR[6]	PIN_C4	LEDR [6]	3.3V
LEDR[7]	PIN_B5	LEDR [7]	3.3V
LEDR[8]	PIN_C5	LEDR [8]	3.3V
LEDR[9]	PIN_D5	LEDR [9]	3.3V

3.4.2 7-segment Displays

The MAX 10 NEEK has two 7-segment displays. These displays are paired to display numbers in various sizes. **Figure 3-15** shows the connection of seven segments (common anode) to pins on MAX 10 FPGA. The segment can be turned on or off by applying a low logic level or high logic level from the FPGA, respectively.

Each segment in a display is indexed from 0 to 6, with corresponding positions given in **Figure 3-15**. **Table 3-4** shows the pin assignment of FPGA to the 7-segment displays.

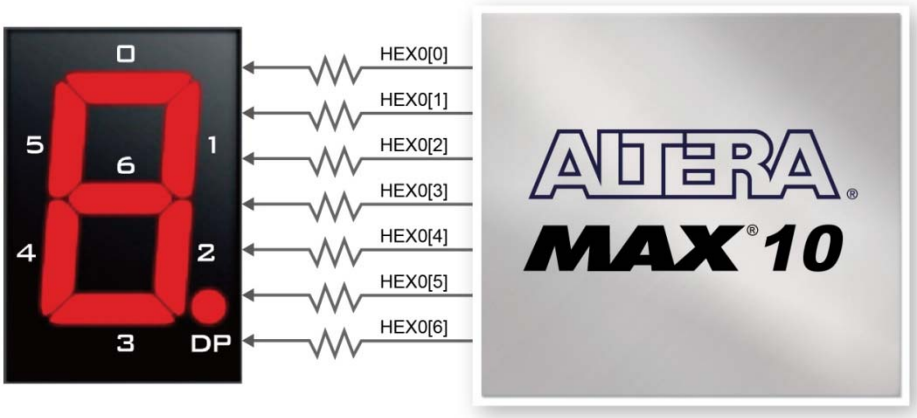

Figure 3-15 Connections between the 7-segment display HEX0 and the MAX 10 FPGA

Table 3-4 Pin Assignment of 7-segment Displays

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HEX0[0]	PIN_D6	Seven Segment Digit 0[0]	3.3V
HEX0[1]	PIN_A5	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_C6	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_A6	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_F7	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_D7	Seven Segment Digit 0[5]	3.3V

HEX0[6]	PIN_B7	Seven Segment Digit 0[6]	3.3V
HEX1[0]	PIN_C7	Seven Segment Digit 1[0]	3.3V
HEX1[1]	PIN_C8	Seven Segment Digit 1[1]	3.3V
HEX1[2]	PIN_D8	Seven Segment Digit 1[2]	3.3V
HEX1[3]	PIN_D10	Seven Segment Digit 1[3]	3.3V
HEX1[4]	PIN_E10	Seven Segment Digit 1[4]	3.3V
HEX1[5]	PIN_H11	Seven Segment Digit 1[5]	3.3V
HEX1[6]	PIN_E6	Seven Segment Digit 1[6]	3.3V

3.4.3 Power Monitor

The MAX 10 NEEK has implemented three power monitor chips to monitor the FPGA core power and VCCIO power voltage and current. **Figure 3-16** shows the connection between the power monitor chip and the MAX 10 FPGA. Through the I2C serial interface, the power monitor can be configured to measure remote voltage and remote current. Programmable calibration value, conversion times, and averaging, combined with an internal multiplier, enable direct readouts of current in amperes and power in watts. **Table 3-5** shows the pin assignment of power monitor I2C bus.

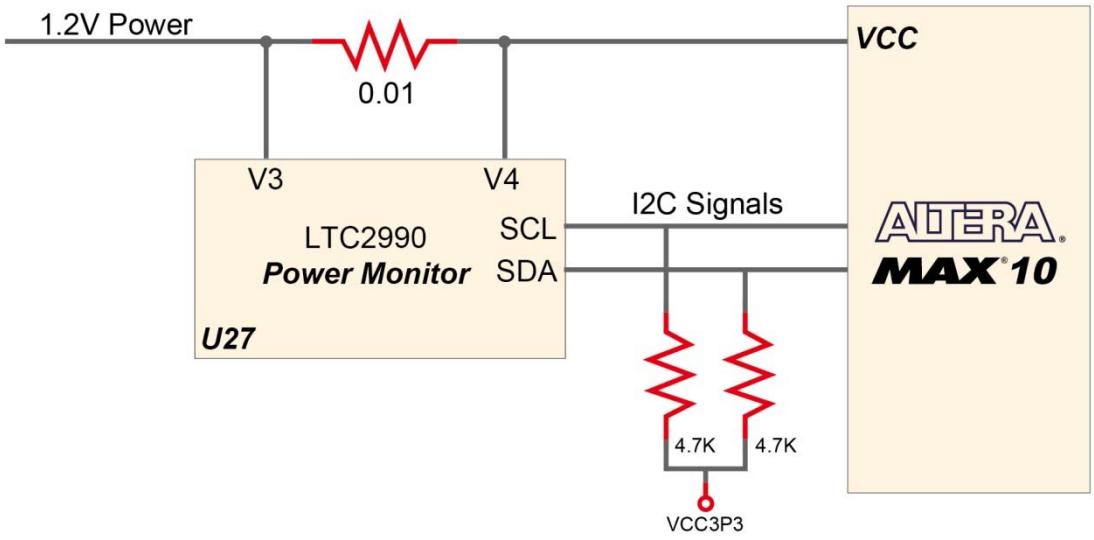

Figure 3-16 Connections between the power monitor chip and the MAX 10 FPGA

Table 3-5 Pin Assignment of Power Monitor I2C bus

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
PM_I2C_SCL	PIN_E8	Power Monitor SCL	3.3V
PM_I2C_SDA	PIN_E9	Power Monitor SDA	3.3V

3.4.4 2x6 TMD Expansion Header

The board has one 2x6 TMD (Terasic Mini Digital) expansion header. The TMD header has 8 digital GPIO user pins connected to the MAX 10 FPGA, two 3.3V power pins and two ground pins. There are two Transient Voltage Suppressor diode arrays used to implement ESD protection for 8 GPIO user pins..

Figure 3-17 shows the connection between the TMD header and MAX 10 FPGA. **Table 3-9** shows the pin assignment of 2x6 TMD header.

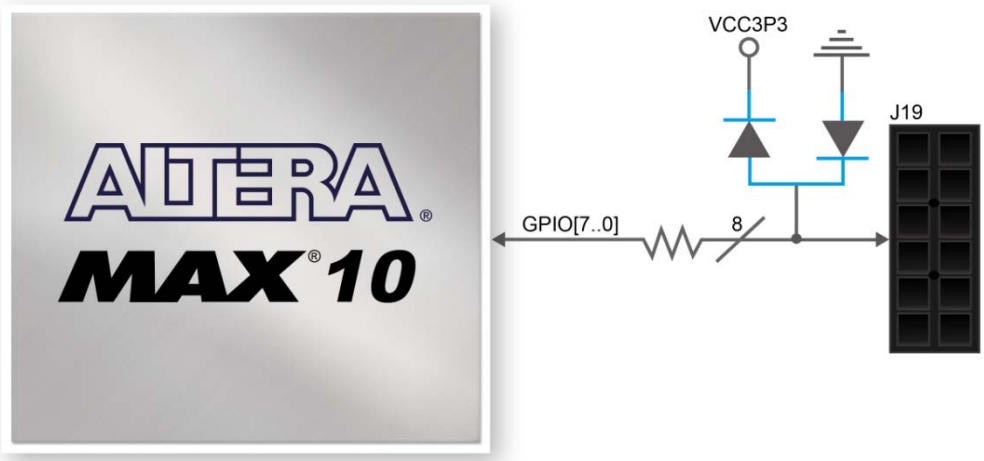

Figure 3-17 Connections between the 2x6 TMD header and MAX 10 FPGA

Table 3-6 Pin Assignment of 2x6 TMD Header

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
GPIO[0]	PIN_Y17	GPIO Connection [0]	3.3V
GPIO[1]	PIN_AA17	GPIO Connection [1]	3.3V
GPIO[2]	PIN_V16	GPIO Connection [2]	3.3V
GPIO[3]	PIN_W15	GPIO Connection [3]	3.3V
GPIO[4]	PIN_AB16	GPIO Connection [4]	3.3V
GPIO[5]	PIN_AA16	GPIO Connection [5]	3.3V
GPIO[6]	PIN_Y16	GPIO Connection [6]	3.3V
GPIO[7]	PIN_W16	GPIO Connection [7]	3.3V

3.4.5 24-bit Audio CODEC

The MAX 10 NEEK offers high-quality 24-bit audio via the Texas Instruments TLV320AIC3254 audio CODEC (Encoder/Decoder). This chip on MAX 10 NEEK supports, line-in, line-out and microphone-in ports with adjustable sample rate from 8kHz to 192kHz. The connection of the audio

circuitry to the FPGA is shown in **Figure 3-18**, and the associated pin assignment to the FPGA is listed in **Table 3-7**. More information about the TLV320AIC3254 CODEC is available in its datasheet, which can be found on the manufacturer's website, or in the directory \MAX10_NEEK_datasheets\Audio CODEC of DECA System CD.

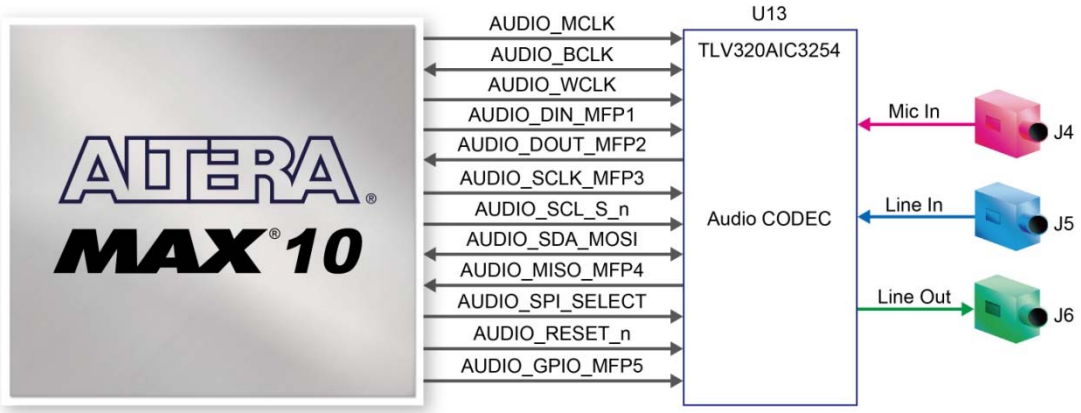

Figure 3-18 Connections between the FPGA and audio CODEC

Table 3-7 Pin Assignment of Audio CODEC

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
AUDIO_MCLK	PIN_J11	Master output Clock	2.5V
AUDIO_BCLK	PIN_J12	Audio serial data bus (primary) bit clock	2.5V
AUDIO_WCLK	PIN_H12	Audio serial data bus (primary) word clock	2.5V
AUDIO_DIN_MFP1	PIN_J13	Audio serial data bus data output/digital microphone output	2.5V
AUDIO_DOUT_MFP2	PIN_H13	Audio serial data bus data input/general purpose input	2.5V
AUDIO_SCLK_MFP3	PIN_H14	SPI serial Clock/headphone-detect output	2.5V
AUDIO_SCL_SS_n	PIN_F15	I2C Clock/SPI interface mode chip-select signal	2.5V
AUDIO_SDA_MOSI	PIN_F16	I2C Data/SPI interface mode serial data output	2.5V
AUDIO_MISO_MFP4	PIN_E13	Serial data input/General purpose input	2.5V
AUDIO_SPI_SELECT	PIN_E14	Control mode select pin	2.5V
AUDIO_RESET_n	PIN_D13	Reset signal	2.5V
AUDIO_GPIO_MFP5	PIN_D14	General Purpose digital IO/CLKOUT input	2.5V

3.4.6 Two Analog Input SMA Connectors

The MAX 10 NEEK board implements two analog input SMA connectors. The analog inputs are amplified and translated by Texas Instruments INA159 gain of 0.2 level translation difference amplifier, then the amplifier's outputs are fed to dedicated single-ended analog input pins for MAX 10 build-in ADC1 and ADC2 respectively. With the amplifiers, the analog input of two SMAs

support from -6.25V to +6.25V range. **Figure 3-19** shows the connection of SMA connectors to the FPGA.

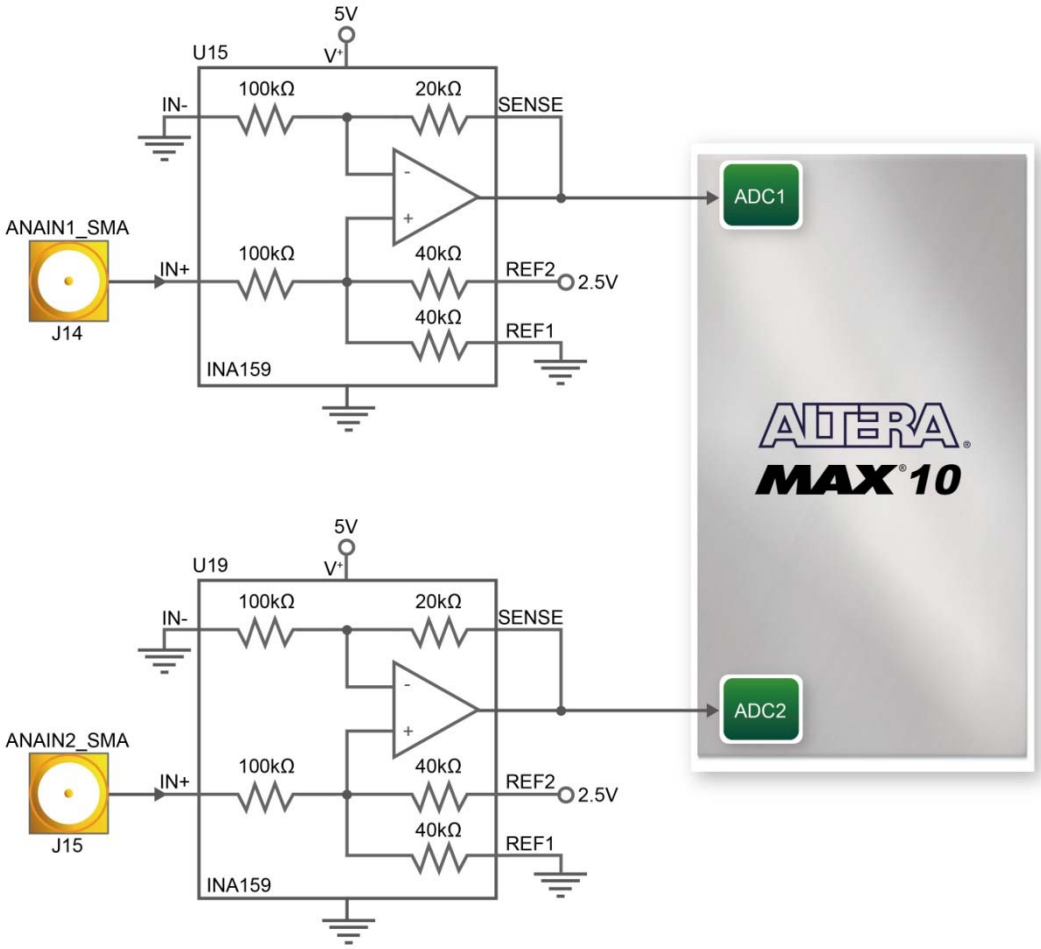

Figure 3-19 Connection of SMA connectors to the FPGA

3.4.7 DDR3 Memory

The board supports 256MB of DDR3 SDRAM comprising of one 16 bit (64Mx16) DDR3 device and one 8 bit (128Mx8) device. The DDR3 devices shipped with this board are running at 300MHz with the soft IP of MAX 10 external memory interface solution. **Figure 3-20** shows the connections between the DDR3 and MAX 10 FPGA. **Table 3-8** shows the DDR3 interface pin assignments.

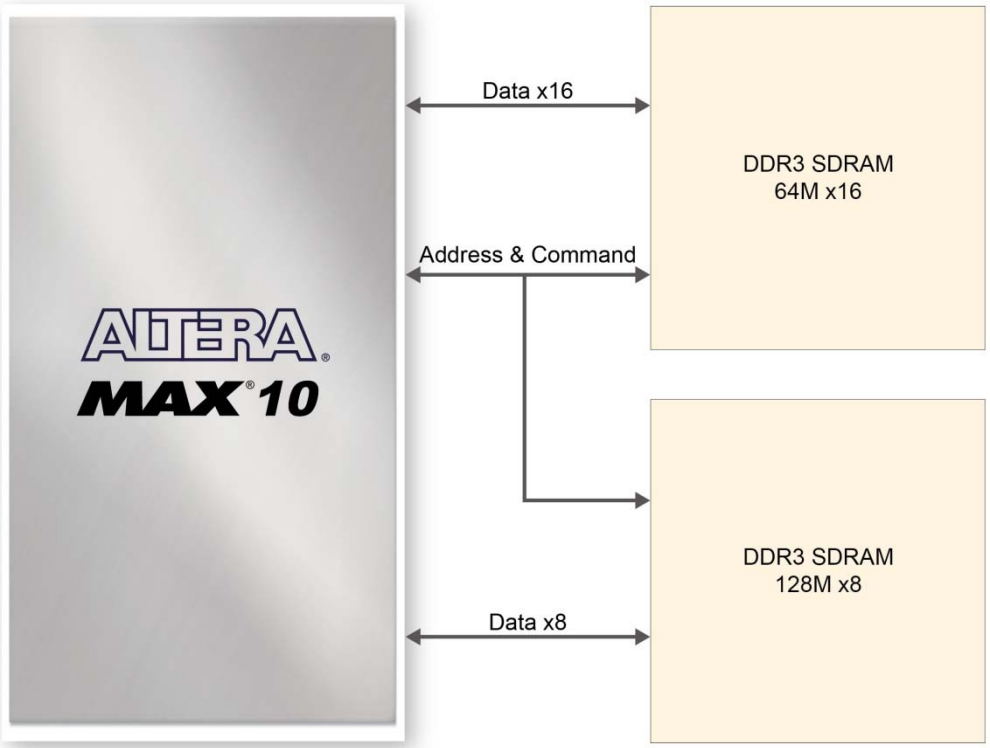

Figure 3-20 Connections between the DDR3 and FPGA

Table 3-8 Pin Assignment of FPGA DDR3 Memory

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
DDR3_A[0]	PIN_U20	DDR3 Address[0]	SSTL-15 Class I
DDR3_A[1]	PIN_F19	DDR3 Address[1]	SSTL-15 Class I
DDR3_A[2]	PIN_V20	DDR3 Address[2]	SSTL-15 Class I
DDR3_A[3]	PIN_G20	DDR3 Address[3]	SSTL-15 Class I
DDR3_A[4]	PIN_F20	DDR3 Address[4]	SSTL-15 Class I
DDR3_A[5]	PIN_E20	DDR3 Address[5]	SSTL-15 Class I
DDR3_A[6]	PIN_E21	DDR3 Address[6]	SSTL-15 Class I
DDR3_A[7]	PIN_Y20	DDR3 Address[7]	SSTL-15 Class I
DDR3_A[8]	PIN_C22	DDR3 Address[8]	SSTL-15 Class I
DDR3_A[9]	PIN_D22	DDR3 Address[9]	SSTL-15 Class I
DDR3_A[10]	PIN_J14	DDR3 Address[10]	SSTL-15 Class I
DDR3_A[11]	PIN_E22	DDR3 Address[11]	SSTL-15 Class I
DDR3_A[12]	PIN_G22	DDR3 Address[12]	SSTL-15 Class I
DDR3_A[13]	PIN_D19	DDR3 Address[13]	SSTL-15 Class I
DDR3_A[14]	PIN_C20	DDR3 Address[14]	SSTL-15 Class I
DDR3_BA[0]	PIN_W22	DDR3 Bank Address[0]	SSTL-15 Class I
DDR3_BA[1]	PIN_Y21	DDR3 Bank Address[1]	SSTL-15 Class I
DDR3_BA[2]	PIN_Y22	DDR3 Bank Address[2]	SSTL-15 Class I

DDR3_CAS_n	PIN_U19	DDR3 Column Address Strobe	SSTL-15 Class I
DDR3_CKE	PIN_V18	Clock Enable pin for DDR3	SSTL-15 Class I
DDR3_CLK_n	PIN_E18	Clock n for DDR3	DIFFERENTIAL 1.5-V SSTL Class I
DDR3_CLK_p	PIN_D18	Clock p for DDR3	Differential 1.5-V SSTL Class I
DDR3_CS_n	PIN_W20	DDR3 Chip Select	SSTL-15 Class I
DDR3_DM[0]	PIN_J15	DDR3 Data Mask[0]	SSTL-15 Class I
DDR3_DM[1]	PIN_N19	DDR3 Data Mask[1]	SSTL-15 Class I
DDR3_DQ[0]	PIN_J18	DDR3 Data[0]	SSTL-15 Class I
DDR3_DQ[1]	PIN_H19	DDR3 Data[1]	SSTL-15 Class I
DDR3_DQ[2]	PIN_K20	DDR3 Data[2]	SSTL-15 Class I
DDR3_DQ[3]	PIN_H18	DDR3 Data[3]	SSTL-15 Class I
DDR3_DQ[4]	PIN_K18	DDR3 Data[4]	SSTL-15 Class I
DDR3_DQ[5]	PIN_H20	DDR3 Data[5]	SSTL-15 Class I
DDR3_DQ[6]	PIN_K19	DDR3 Data[6]	SSTL-15 Class I
DDR3_DQ[7]	PIN_J20	DDR3 Data[7]	SSTL-15 Class I
DDR3_DQ[8]	PIN_L18	DDR3 Data[8]	SSTL-15 Class I
DDR3_DQ[9]	PIN_M18	DDR3 Data[9]	SSTL-15 Class I
DDR3_DQ[10]	PIN_M14	DDR3 Data[10]	SSTL-15 Class I
DDR3_DQ[11]	PIN_N20	DDR3 Data[11]	SSTL-15 Class I
DDR3_DQ[12]	PIN_L20	DDR3 Data[12]	SSTL-15 Class I
DDR3_DQ[13]	PIN_M20	DDR3 Data[13]	SSTL-15 Class I
DDR3_DQ[14]	PIN_M15	DDR3 Data[14]	SSTL-15 Class I
DDR3_DQ[15]	PIN_L19	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[16]	PIN_T19	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[17]	PIN_R20	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[18]	PIN_R15	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[19]	PIN_P15	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[20]	PIN_P19	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[21]	PIN_P14	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[22]	PIN_R14	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQ[23]	PIN_P20	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQS_n[0]	PIN_K15	DDR3 Data Strobe n[0]	Differential 1.5-V SSTL Class I
DDR3_DQS_n[1]	PIN_L15	DDR3 Data Strobe n[1]	Differential 1.5-V SSTL Class I
DDR3_DQS_p[0]	PIN_K14	DDR3 Data Strobe p[0]	Differential 1.5-V SSTL Class I
DDR3_DQS_p[1]	PIN_L14	DDR3 Data Strobe p[1]	Differential 1.5-V SSTL Class I
DDR3_ODT	PIN_V22	DDR3 On-die Termination	SSTL-15 Class I
DDR3_RAS_n	PIN_N18	DDR3 Row Address Strobe	SSTL-15 Class I
DDR3_RESET_n	PIN_B22	DDR3 Reset	SSTL-15 Class I
DDR3_WE_n	PIN_W19	DDR3 Write Enable	SSTL-15 Class I

3.4.8 QSPI Flash

The MAX 10 NEEK supports a 512M-bit serial NOR flash device for non-volatile storage, user data and program. This device has a 4-bit data interface and uses 3.3V CMOS signaling standard. Connections between MAX 10 FPGA and Flash are shown in **Figure 3-21**. **Table 3-9** shows the DDR3 interface pin assignments

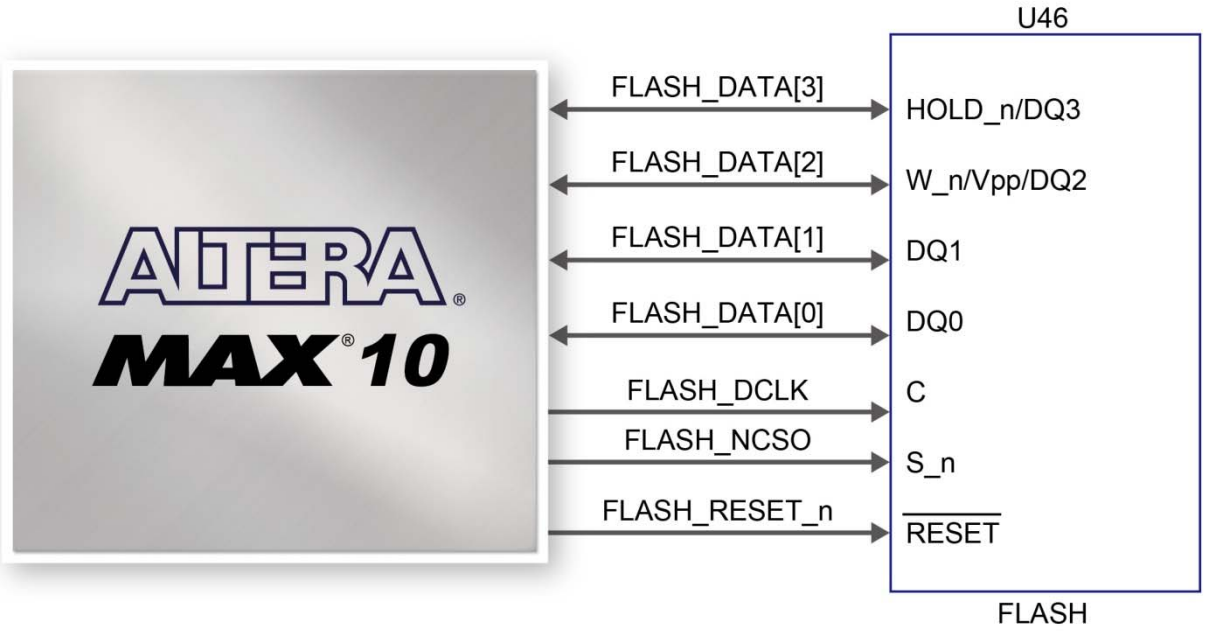

Figure 3-21 Connections between MAX 10 FPGA and QSPI Flash

Table 3-9 Pin Assignment of QSPI Flash

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
FLASH_DATA[0]	**PIN_AB18**	**FLASH Data[0]**	**3.3V**
FLASH_DATA[1]	**PIN_AA19**	**FLASH Data[1]**	**3.3V**
FLASH_DATA[2]	**PIN_AB19**	**FLASH Data[2]**	**3.3V**
FLASH_DATA[3]	**PIN_AA20**	**FLASH Data[3]**	**3.3V**
FLASH_DCLK	**PIN_AB17**	**FLASH Data Clock**	**3.3V**
FLASH_NCSO	**PIN_AB21**	**FLASH Chip Enable**	**3.3V**
FLASH_RESET_n	**PIN_AB20**	**FLASH Chip Reser**	**3.3V**

3.4.9 Ethernet

The board supports Gigabit Ethernet transfer by an external Marvell 88E1111 PHY chip. The

88E1111 chip with integrated 10/100/1000 Mbps Gigabit Ethernet transceiver support GMII/MII/RGMII/TBI MAC interfaces. **Figure 3-22** shows the connections between the MAX 10 FPGA, Ethernet PHY, and RJ-45 connector. The pin assignment associated to Gigabit Ethernet

interface is listed in Table 3-10.

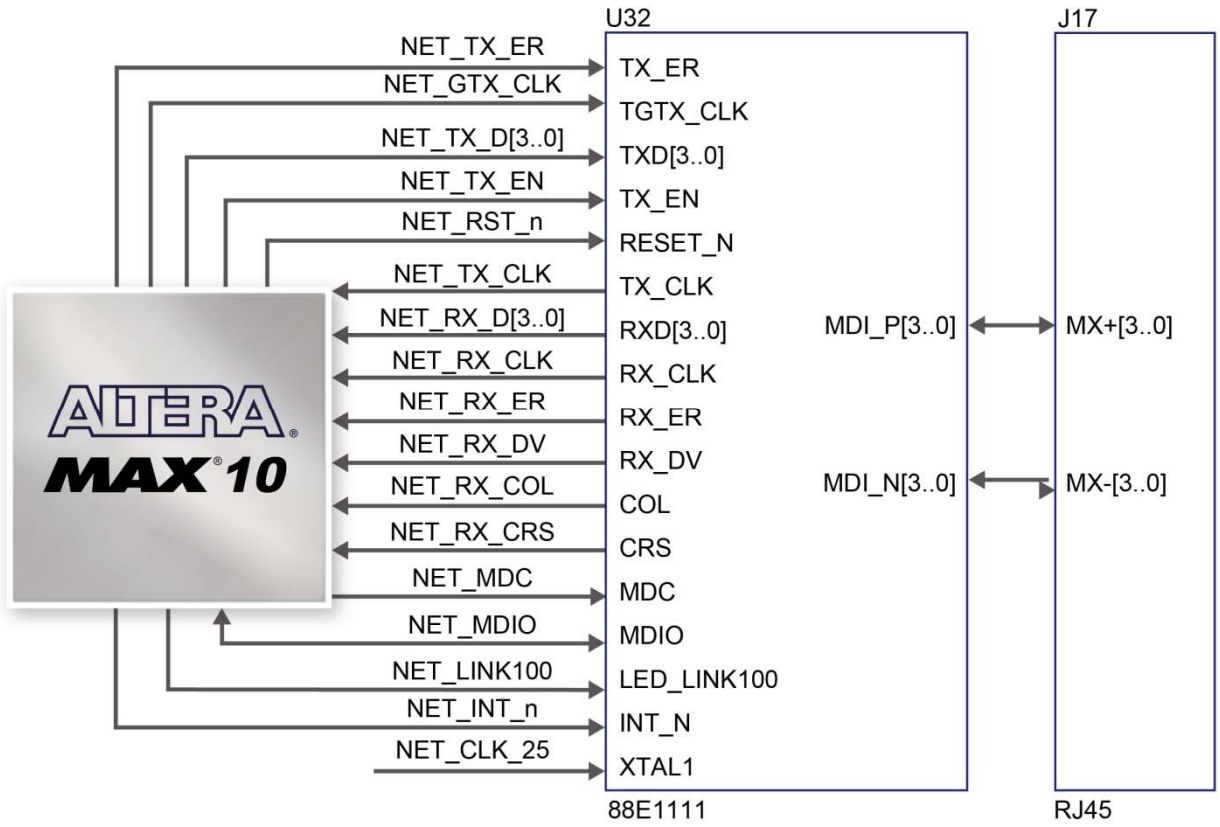

Figure 3-22 Connections between the MAX 10 FPGA and Gigabit Ethernet

Table 3-10 Pin Assignment of Ethernet PHY

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
NET_TX_EN	PIN_C10	GMII and MII transmit enable	2.5V
NET_TX_ER	PIN_C12	GMII and MII transmit error	2.5V
NET_TX_CLK	PIN_E11	MII transmit clock	3.3V
NET_TX_D[0]	PIN_A12	MII transmit data[0]	2.5V
NET_TX_D[1]	PIN_B12	MII transmit data[1]	2.5V
NET_TX_D[2]	PIN_A13	MII transmit data[2]	2.5V
NET_TX_D[3]	PIN_A14	MII transmit data[3]	2.5V
NET_RX_DV	PIN_A8	GMII and MII receive data valid	2.5V
NET_RX_ER	PIN_B8	GMII and MII receive data valid	2.5V
NET_RX_D[0]	PIN_A10	GMII and MII receive data[0]	2.5V
NET_RX_D[1]	PIN_B10	GMII and MII receive data[1]	2.5V
NET_RX_D[2]	PIN_A11	GMII and MII receive data[2]	2.5V
NET_RX_D[3]	PIN_B11	GMII and MII receive data[3]	2.5V
NET_RX_CLK	PIN_J10	GMII and MII receive clock	3.3V
NET_RST_n	PIN_C14	Hardware Reset Signal	2.5V
NET_MDIO	PIN_E12	Management Data	2.5V

NET_MDC	PIN_D12	Management Data Clock Reference	2.5V
NET_RX_COL	PIN_C9	GMII and MII collision	2.5V
NET_RX_CRS	PIN_A9	GMII and MII carrier sense	2.5V
NET_GTX_CLK	PIN_C11	GMII Transmit Clock	2.5
NET_LINK100	PIN_A7	Parallel LED output of 100BASE-TX link	2.5V
NET_INT_n	PIN_C13	Interrupt open drain output	2.5V

3.4.10 HDMI RX

The development board provides High Performance HDMI Receiver via the Analog Devices ADV7611 which incorporates HDMI v1.4a features, including 3D video support, and 165 MHz supports all video formats up to 1080p and UXGA. The ADV7611 is controlled via a serial I2C bus interface, which is connected to pins on the MAX 10 FPGA. A schematic diagram of the HDMI RX circuitry is shown in **Figure 3-23**. Detailed information on using the ADV7611 HDMI RX is available on the manufacturer's website, or under the Datasheets\HDMI folder on the Kit System CD.

Table 3-11 lists the HDMI Interface pin assignments and signal names relative to the MAX 10 device.

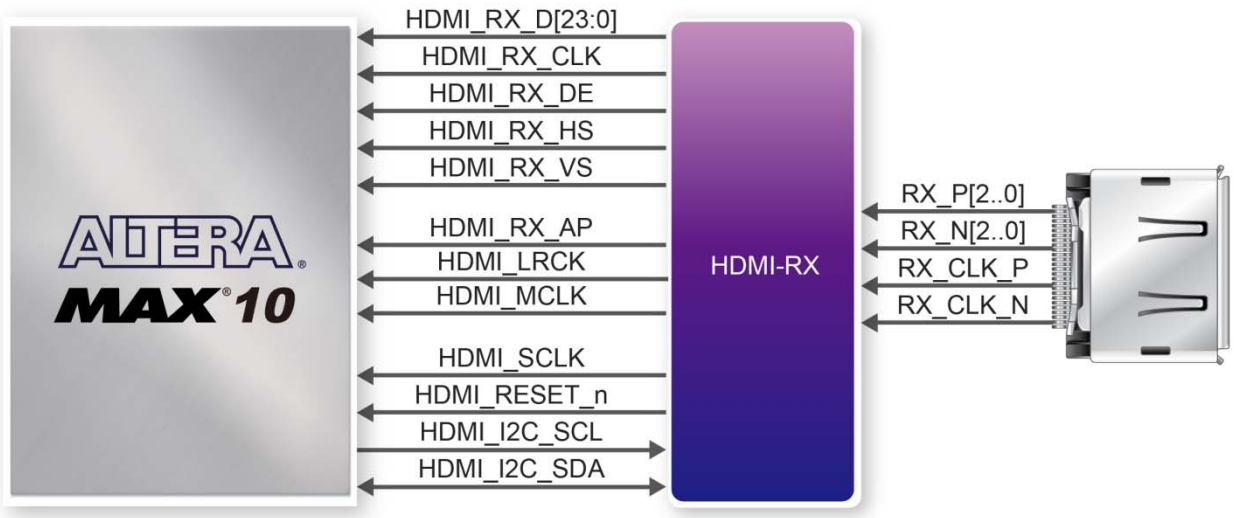

Figure 3-23 Connection between the MAX 10 FPGA and HDMI Receiver

Table 3-11 Pin Assignment of HDMI RX

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HDMI_RX_D0	PIN_AA9	Video Pixel Output Port	3.3V
HDMI_RX_D1	PIN_AB9	Video Pixel Output Port	3.3V
HDMI_RX_D2	PIN_Y10	Video Pixel Output Port	3.3V
HDMI_RX_D3	PIN_AA10	Video Pixel Output Port	3.3V
HDMI_RX_D4	PIN_AB10	Video Pixel Output Port	3.3V

HDMI_RX_D5	PIN_Y11	Video Pixel Output Port	3.3V
HDMI_RX_D6	PIN_AA11	Video Pixel Output Port	3.3V
HDMI_RX_D7	PIN_AB11	Video Pixel Output Port	3.3V
HDMI_RX_D8	PIN_Y14	Video Pixel Output Port	3.3V
HDMI_RX_D9	PIN_AB15	Video Pixel Output Port	3.3V
HDMI_RX_D10	PIN_AA15	Video Pixel Output Port	3.3V
HDMI_RX_D11	PIN_W14	Video Pixel Output Port	3.3V
HDMI_RX_D12	PIN_V14	Video Pixel Output Port	3.3V
HDMI_RX_D13	PIN_V15	Video Pixel Output Port	3.3V
HDMI_RX_D14	PIN_U15	Video Pixel Output Port	3.3V
HDMI_RX_D15	PIN_AB14	Video Pixel Output Port	3.3V
HDMI_RX_D16	PIN_AA14	Video Pixel Output Port	3.3V
HDMI_RX_D17	PIN_AB13	Video Pixel Output Port	3.3V
HDMI_RX_D18	PIN_Y13	Video Pixel Output Port	3.3V
HDMI_RX_D19	PIN_AB12	Video Pixel Output Port	3.3V
HDMI_RX_D20	PIN_AA12	Video Pixel Output Port	3.3V
HDMI_RX_D21	PIN_W13	Video Pixel Output Port	3.3V
HDMI_RX_D22	PIN_W12	Video Pixel Output Port	3.3V
HDMI_RX_D23	PIN_V13	Video Pixel Output Port	3.3V
HDMI_RX_CLK	PIN_P11	Line-Locked Output Clock	3.3V
HDMI_RX_DE	PIN_W10	Data Enable Signal for Digital Video.	3.3V
HDMI_RX_HS	PIN_V12	Horizontal Synchronization	3.3V
HDMI_RX_VS	PIN_W11	Vertical Synchronization	3.3V
HDMI_RX_INT1	PIN_P12	Interrupt Signal	3.3V
HDMI_I2C_SCL	PIN_R13	I2C Clock	3.3V
HDMI_I2C_SDA	PIN_P13	I2C Data	3.3V
HDMI_MCLK	PIN_R12	Audio Master Output Clock	3.3V
HDMI_LRCLK	PIN_V11	Audio Left/Right Clock	3.3V
HDMI_SCLK	PIN_W8	Audio Serial Output Clock	3.3V
HDMI_AP	PIN_W9	Audio Output Pin	3.3V
HDMI_RX_RESET_n	PIN_AA13	System Reset Input	3.3V

3.4.11 8M Pixel MIPI CS2 Camera

The MAX 10 NEEK provides a high performance, 8 megapixel RAW image sensor that delivers 3264x2488 at 30 fps. Through Toshiba TC358748XBG MIPI CSI-2 to Parallel bridge device, converts MIPI data from on-board camera module to MAX 10 FPGA over a parallel port interface. **Figure 3-24** shows the connection of MIPI camera module, MIPI CSI-2 to parallel bridge device and MAX 10 FPGA. The pin assignment associated to this MIPI CSI-2 to parallel interface is shown in **Table 3-12.**

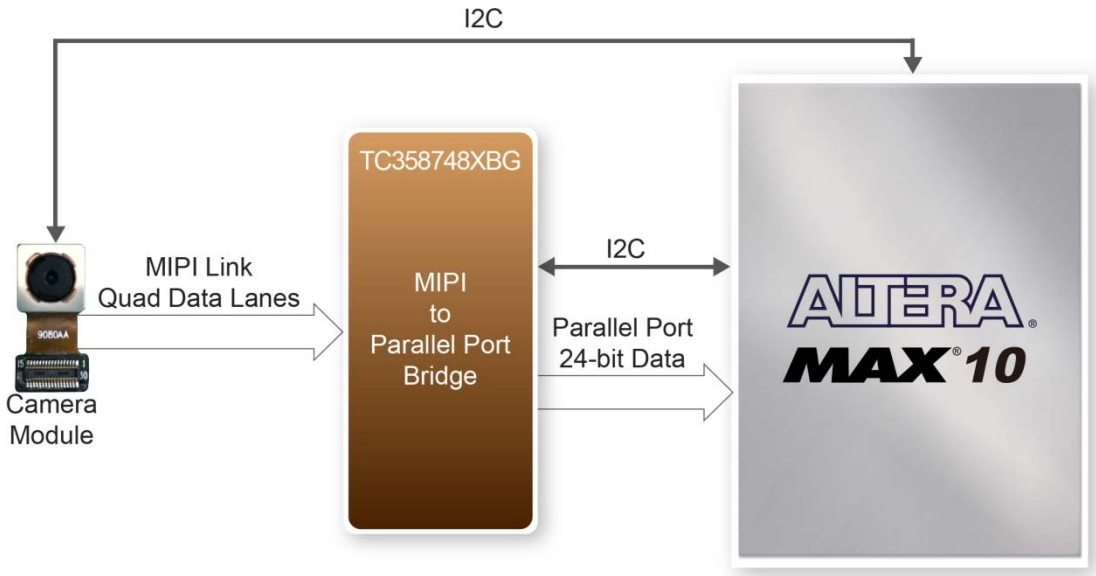

Figure 3-24 Connections between the MAX 10 FPGA, MIPI CSI-2 to parallel bridge device and MIPI camera module

Table 3-12 Pin Assignment of MIPI CSI-2 to parallel interface

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MIPI_PIXEL_D[0]	PIN_U6	Parallel Port Data	3.3V
MIPI_PIXEL_D[1]	PIN_W5	Parallel Port Data	3.3V
MIPI_PIXEL_D[2]	PIN_V7	Parallel Port Data	3.3V
MIPI_PIXEL_D[3]	PIN_V5	Parallel Port Data	3.3V
MIPI_PIXEL_D[4]	PIN_W4	Parallel Port Data	3.3V
MIPI_PIXEL_D[5]	PIN_V4	Parallel Port Data	3.3V
MIPI_PIXEL_D[6]	PIN_W3	Parallel Port Data	3.3V
MIPI_PIXEL_D[7]	PIN_W6	Parallel Port Data	3.3V
MIPI_PIXEL_D[8]	PIN_W7	Parallel Port Data	3.3V
MIPI_PIXEL_D[9]	PIN_Y3	Parallel Port Data	3.3V
MIPI_PIXEL_D[10]	PIN_Y4	Parallel Port Data	3.3V
MIPI_PIXEL_D[11]	PIN_Y5	Parallel Port Data	3.3V
MIPI_PIXEL_D[12]	PIN_AB3	Parallel Port Data	3.3V
MIPI_PIXEL_D[13]	PIN_AB4	Parallel Port Data	3.3V
MIPI_PIXEL_D[14]	PIN_AA5	Parallel Port Data	3.3V
MIPI_PIXEL_D[15]	PIN_AB5	Parallel Port Data	3.3V
MIPI_PIXEL_D[16]	PIN_AB7	Parallel Port Data	3.3V
MIPI_PIXEL_D[17]	PIN_Y8	Parallel Port Data	3.3V
MIPI_PIXEL_D[18]	PIN_AA8	Parallel Port Data	3.3V
MIPI_PIXEL_D[19]	PIN_AA7	Parallel Port Data	3.3V
MIPI_PIXEL_D[20]	PIN_Y7	Parallel Port Data	3.3V

MIPI_PIXEL_D[21]	PIN_AB6	Parallel Port Data	3.3V
MIPI_PIXEL_D[22]	PIN_AA6	Parallel Port Data	3.3V
MIPI_PIXEL_D[23]	PIN_Y6	Parallel Port Data	3.3V
MIPI_RESET_n	PIN_AA3	Master Reset signal for MIPI camera and bridge device	3.3V
MIPI_PIXEL_CLK	PIN_V10	Parallel Port Clock signal	3.3V
MIPI_PIXEL_HS	PIN_AA2	Parallel Port Horizontal Synchronization signal	3.3V
MIPI_PIXEL_VS	PIN_AA1	Parallel Port Vertical Synchronization signal	3.3V
MIPI_CS_n	PIN_U7	Chip Select	3.3V
MIPI_REFCLK	PIN_W17	Reference Clock Input of bridge device	3.3V
MIPI_I2C_SCL	PIN_Y1	I2C Clock for bridge device	3.3V
MIPI_I2C_SDA	PIN_M2	I2C Data for bridge device	3.3V
CAMERA_PWDN_n	PIN_R11	Power Down signal of MIPI camera	3.3V
CAMERA_I2C_SCL	PIN_A20	I2C Clock for MIPI camera	2.5V
CAMERA_I2C_SDA	PIN_B19	I2C Data for MIPI camera	2.5V

3.4.12 7.0 Inch Color LCD with 5-point Capacitive-touch

The MAX 10 NEEK provides a 7.0 inch color LCD module which is an all-purpose 5-point capacitive touch-screen for FPGA applications. **Figure 3-25** shows the connection of 7.0 inch color LCD and MAX 10 FPGA. The pin assignment associated to this 7.0 inch color LCD interface is shown in **Table 3-12**.

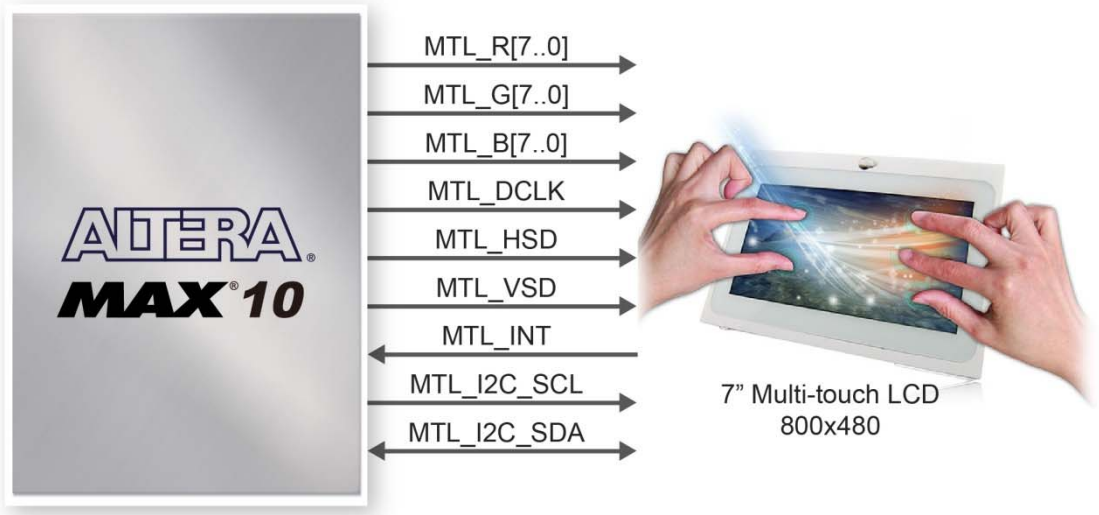

Figure 3-25 Connections between 7.0 inch color LCD and MAX 10 FPGA

Table 3-13 Pin Assignment of 7.0 inch LCD interface

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MTL2_R[0]	PIN_U5	Red Data (LSB)	3.3V
MTL2_R[1]	PIN_U4	Red Data	3.3V

MTL2_R[2]	PIN_U3	Red Data	3.3V
MTL2_R[3]	PIN_W2	Red Data	3.3V
MTL2_R[4]	PIN_U2	Red Data	3.3V
MTL2_R[5]	PIN_V1	Red Data	3.3V
MTL2_R[6]	PIN_T2	Red Data	3.3V
MTL2_R[7]	PIN_T1	Red Data (MSB)	3.3V
MTL2_G[0]	PIN_T5	Green Data (LSB)	3.3V
MTL2_G[1]	PIN_T6	Green Data	3.3V
MTL2_G[2]	PIN_R1	Green Data	3.3V
MTL2_G[3]	PIN_R2	Green Data	3.3V
MTL2_G[4]	PIN_R4	Green Data	3.3V
MTL2_G[5]	PIN_P1	Green Data	3.3V
MTL2_G[6]	PIN_R5	Green Data	3.3V
MTL2_G[7]	PIN_R7	Green Data (MSB)	3.3V
MTL2_B[0]	PIN_P4	Blue Data (LSB)	3.3V
MTL2_B[1]	PIN_P5	Blue Data	3.3V
MTL2_B[2]	PIN_N3	Blue Data	3.3V
MTL2_B[3]	PIN_P8	Blue Data	3.3V
MTL2_B[4]	PIN_N4	Blue Data	3.3V
MTL2_B[5]	PIN_N8	Blue Data	3.3V
MTL2_B[6]	PIN_N9	Blue Data	3.3V
MTL2_B[7]	PIN_M8	Blue Data (MSB)	3.3V
MTL2_DCLK	PIN_W1	Sample Clock	3.3V
MTL2_HSD	PIN_N1	Horizontal Sync Input	3.3V
MTL2_VSD	PIN_N2	Vertical Sync Input	3.3V
MTL2_I2C_SCL	PIN_P9	I2C Serial Clock for Touch Screen	3.3V
MTL2_I2C_SDA	PIN_P10	I2C Serial Data for Touch Screen	3.3V
MTL2_INT	PIN_R10	Interrupt Signal for Touch Screen	3.3V

3.4.13 2x10 ADC Header

The board has a 2x10 ADC header with sixteen analog inputs connected to FPGA ADC1 and ADC2 respectively. The 1x3 header J12 is used to select pin 18 of 2x10 header J7 or potentiometer input to be connected to the channel 8 of FPGA ADC2. Short pin 1 and pin 2 of J12 to select potentiometer, short pin 3 and pin 4 to select pin 18 of 2x10 header J7. The 1x3 header J13 is used to select pin 16 of 2x10 header J7 or on-board microphone to be connected to the channel 7 of FPGA ADC2. Short pin 1 and pin 2 of J13 to select on-board microphone, short pin 3 and pin 4 to select pin 16 of 2x10 header J7. **Figure 3-26** shows the connection of 2x10 ADC header and MAX 10 FPGA.

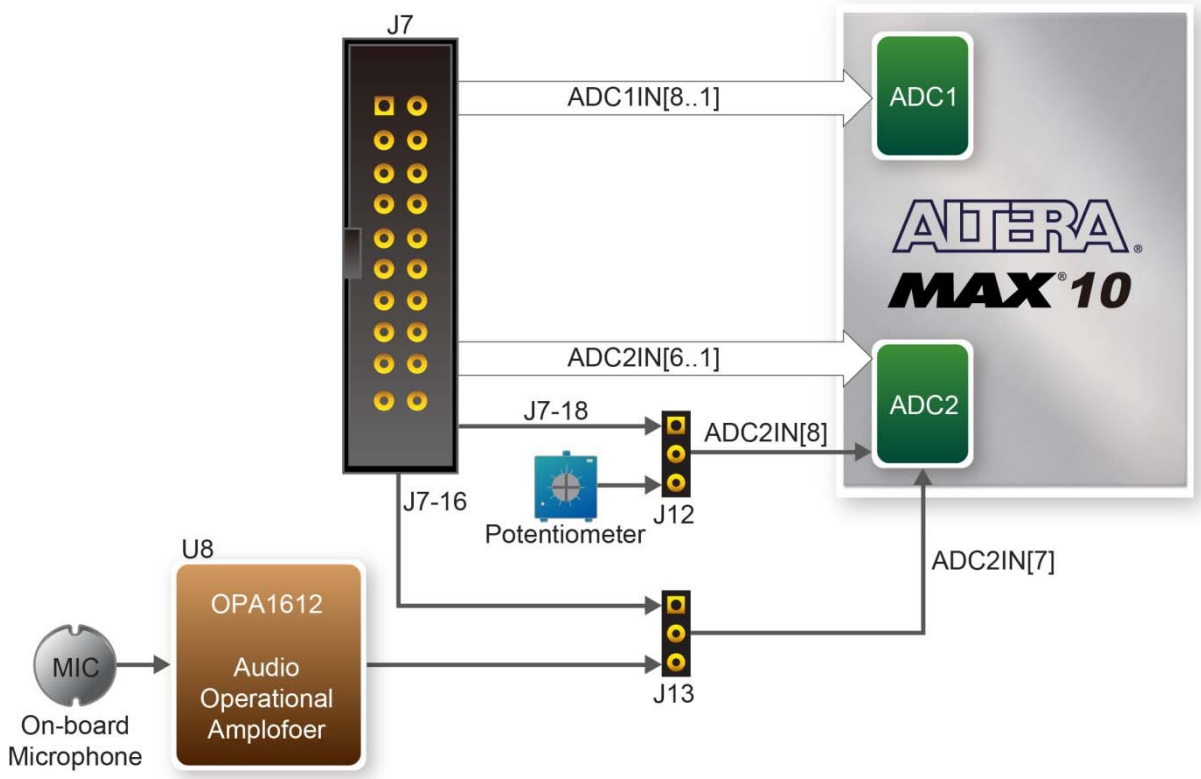

Figure 3-26 Connections between 2x10 ADC header, potentiometer, on-board microphone and MAX 10 FPGA

3.4.14 Potentiometer

Short pin 1 and pin 2 of J12 will select potentiometer for providing adjustable voltage to the channel 8 of FPGA ADC2.

3.4.15 On-board Microphone

The board provides an on-board microphone for audio applications. Short pin 1 and pin 2 of J13 to select on-board microphone to be connected to the channel 7 of FPGA ADC2. The output audio signal from on-board microphone will be pre-amplified by audio operational amplifier OPA1612, then fed into the FPGA ADC. **Figure 3-26** shows the connection of on-board microphone and MAX 10 FPGA.

3.4.16 PS/2 Serial Port

The MAX 10 NEEK comes with a standard PS/2 interface and a connector for a PS/2 keyboard or mouse. **Figure 3-27** shows the connection of PS/2 circuit to the FPGA. Users can use the PS/2 keyboard and mouse on the MAX 10 NEEK simultaneously by a PS/2 Y-Cable, as shown in **Figure 3-28**. Instructions on how to use PS/2 mouse and/or keyboard can be found on various educational

websites. The pin assignment associated to this interface is shown in **Figure 3-28**.

Note: If users connect only one PS/2 equipment, the PS/2 signals connected to the FPGA I/O should be “PS2_CLK” and “PS2_DAT”.

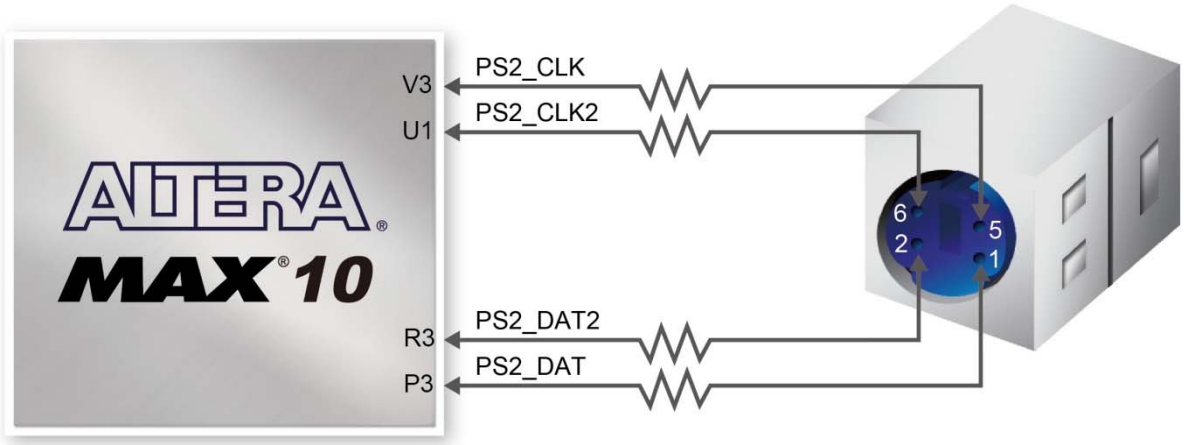

Figure 3-27 Connections between the MAX 10 FPGA and PS/2

Figure 3-28 Y-Cable for using keyboard and mouse simultaneously

Table 3-14 Pin Assignment of PS/2

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
PS2_CLK	PIN_V3	PS/2 Clock	3.3V
PS2_DAT	PIN_P3	PS/2 Data	3.3V
PS2_CLK2	PIN_U1	PS/2 Clock (reserved for second PS/2 device)	3.3V
PS2_DAT2	PIN_R3	PS/2 Data (reserved for second PS/2 device)	3.3V

3.4.17 Digital-to-Analog Converter (DAC)

The board provides a Texas Instruments DAC8551 16-bit digital-to-analog converter (DAC). It is a small, low power, voltage output DAC. The DAC8551 used a versatile 3-wire serial interface that operates at clock rates to 30MHz and is compatible with standard SPI, QSPI, Microwire and DSP

interfaces. The analog voltage output of DAC8551 is connected to a SMA connector. **Figure 3-27** shows the connection between DAC and MAX 10 FPGA. The pin assignment associated to this DAC is shown in **Table 3-15.**

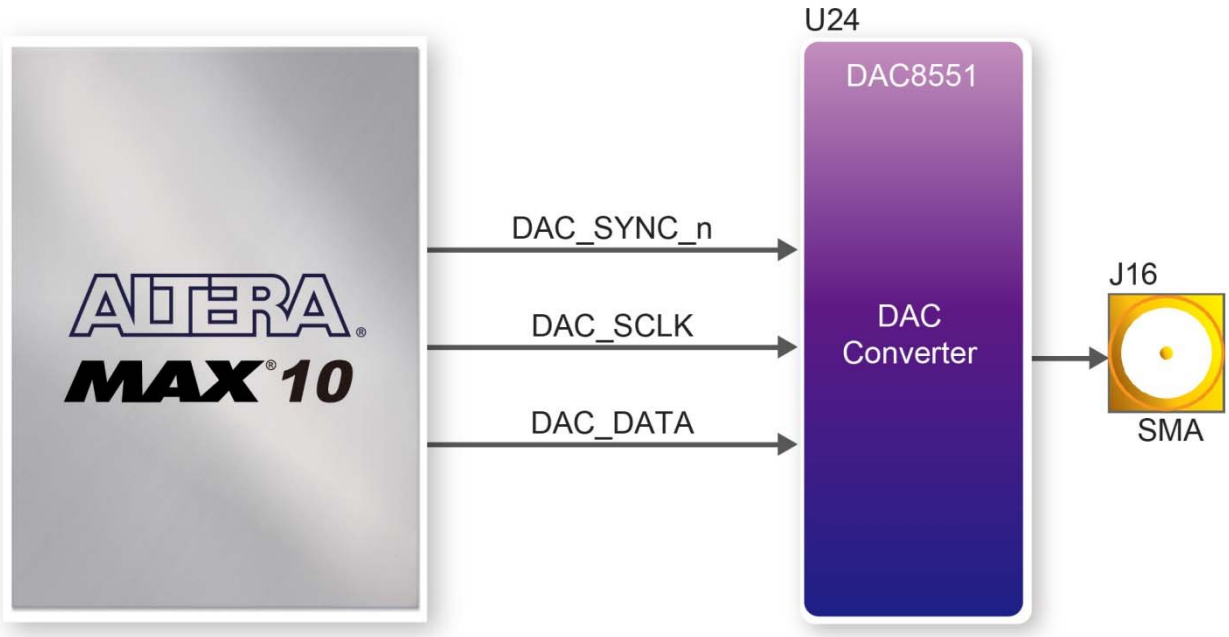

Figure 3-29 Connection between DAC and MAX 10 FPGA

Table 3-15 Pin Assignment of DAC

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
DAC_SYNC_n	PIN_B2	Frame Sync Signal for Input Data	3.3V
DAC_SCLK	PIN_B1	Serial Clock Input	3.3V
DAC_DATA	PIN_A2	Serial Data Input	3.3V

3.4.18 UART to USB

The board has one UART interface connected for communication with the MAX 10 FPGA. This interface doesn't support HW flow control signals. The physical interface is implemented by UART-USB onboard bridge from a FT232R chip to the host with an USB Mini-B connector. More information about the chip is available on the manufacturer's website, or in the directory \Datasheets\UART TO USB of MAX 10 NEEK system CD. **Figure 3-30** shows the connections between the MAX 10 FPGA, FT232R chip, and the USB Mini-B connector. **Figure 3-34** lists the pin assignment of UART interface connected to the MAX 10 FPGA.

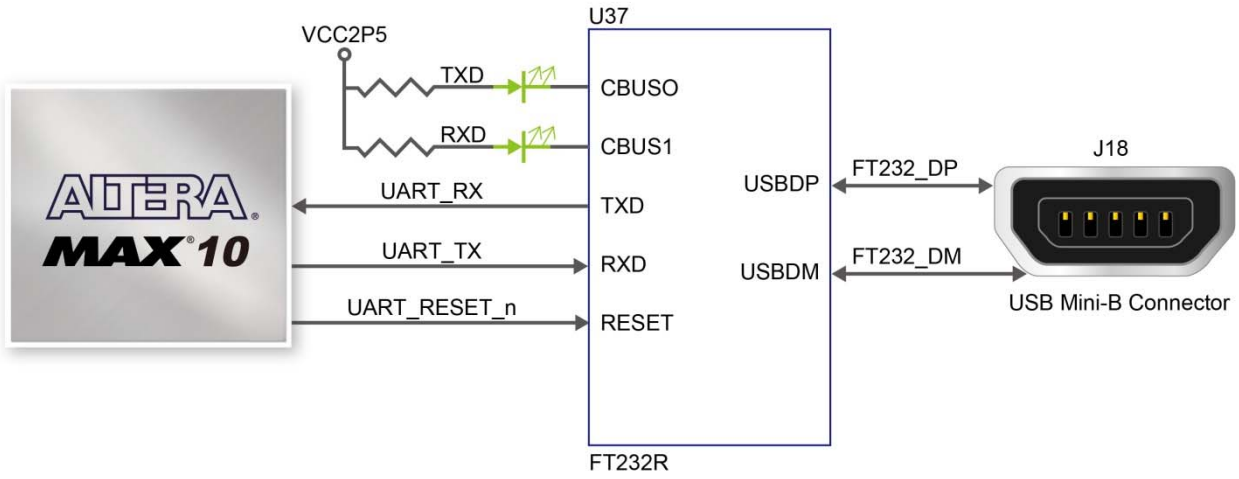

Figure 3-30 Connections between the HPS and FT232R Chip

Table 3-16 Pin Assignment of UART Interface

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
UART_RX	PIN_E16	FPGA UART Receiver	2.5V
UART_TX	PIN_E15	FPGA UART Transmitter	2.5V
UART_RESET_n	PIN_D15	Reset Signal for UART Device	2.5V

3.4.19 Ambient Light Sensor

The MAX 10 NEEK has a Light-to-Digital Ambient Light Sensor (APDS-9301) that converts light intensity to digital signal output capable of I2C interface. with I2C digital interface and programmable-event interrupt output. The digital output of APDS-9301 will be input to the MAX 10 FPGA where illuminance I lux is derived using an empirical formula to approximate the human-eye response. **Figure 3-35** shows the connection of APDS-9301 to the MAX 10 FPGA. **Table 3-17** lists the Ambient Light Sensor pin assignments.

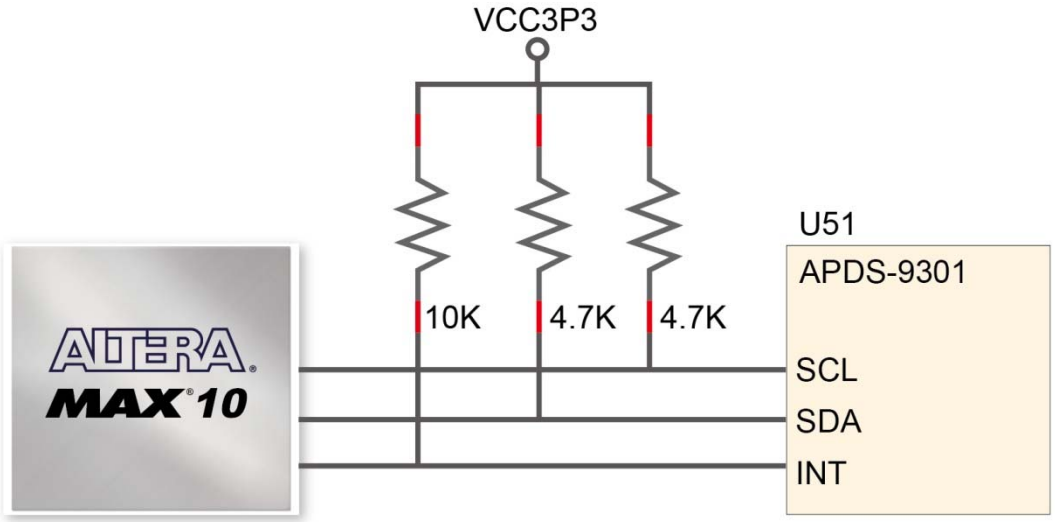

Figure 3-31 hows the connections between the MAX 10 FPGA and Ambient Light Sensor

Table 3-17 Pin Assignment of Ambient Light Sensor

Signal Name	FPGA Pin No.	Description	I/O Standard
LSENSOR_SCL	PIN_M1	I2C Serial Clock	3.3V
LSENSOR_SDA	PIN_T3	I2C Serial Data	3.3V
LSENSOR_INT	PIN_M2	Interrupt signal from Sensor	3.3V

3.4.20 Humidity and Temperature Sensor

The MAX 10 NEEK has a humidity and temperature sensor (HDC1000) that provides excellent measurement accuracy at very low power. The HDC1000 sensor is placed at board edge and away from the heat source of MAX 10 NEEK so that user can make ambient temperature measurement without heat interference from the heat source of MAX 10 NEEK. **Figure 3-32** shows the connection of humidity and temperature sensor to MAX 10 FPGA. **Table 3-18** lists the humidity and temperature sensor pin assignments.

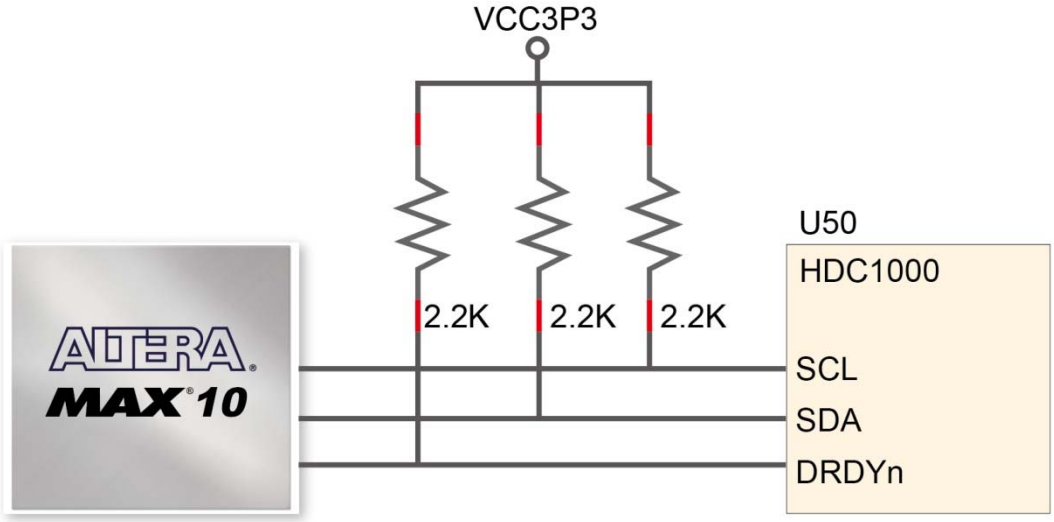

Figure 3-32 shows the connections between the MAX 10 FPGA and humidity and temperature sensor.

Table 3-18 Pin Assignment of Humidity and Temperature Sensor

Signal Name	FPGA Pin No.	Description	I/O Standard
RH_TEMP_I2C_SCL	PIN_Y18	I2C Clock for HDC1000 Sensor	3.3V
RH_TEMP_I2C_SDA	PIN_W18	I2C Data for HDC1000 Sensor	3.3V
RH_TEMP_DRDY_n	PIN_Y19	Data ready input from HDC1000 Sensor	3.3V

3.4.21 Accelerometer Sensor

The board comes with a digital accelerometer sensor module (ADXL345), commonly known as G-sensor. This G-sensor is a small, thin, ultralow power assumption 3-axis accelerometer with high-resolution measurement. Digitalized output is formatted as 16-bit in two's complement and can be accessed through I2C interface. The I2C address of accelerometer is 0xA6/0xA7. More information about this chip can be found in its datasheet, which is available on manufacturer's website or in the directory \Datasheet folder of MAX 10 NEEK system CD. **Figure 3-33** shows the connections between the MAX 10 FPGA and accelerometer. **Table 3-19** lists the pin assignment of accelerometer to the MAX 10 FPGA.

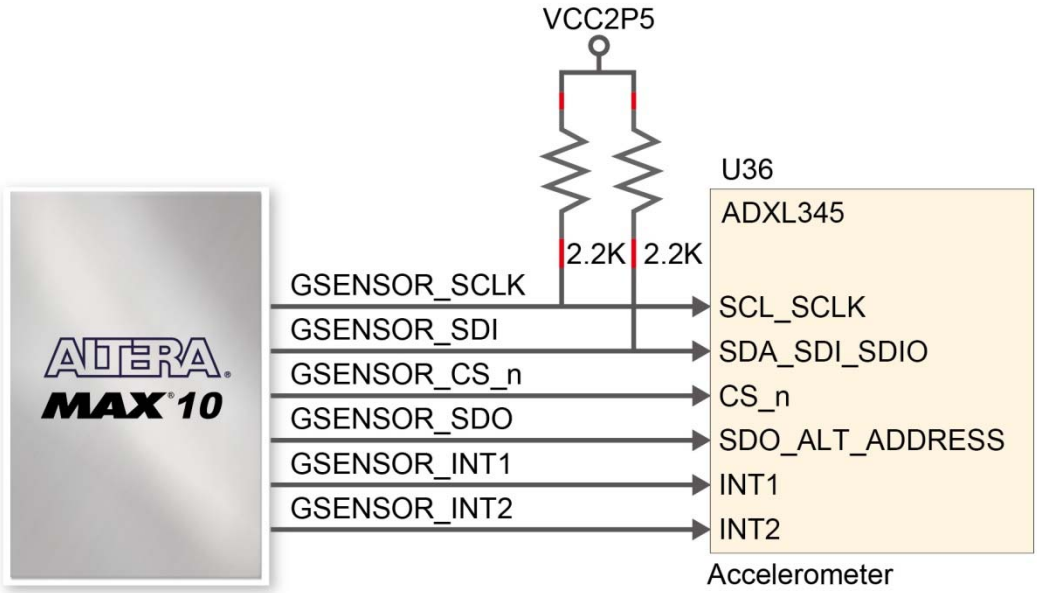

Figure 3-33 shows the connections between the MAX 10 FPGA and accelerometer sensor.

Table 3-19 Pin Assignment of Accelerometer Sensor

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
GSENSOR_SDI	PIN_C15	I2C serial data/SPI serial data input/3-wire interface serial dta output	2.5V
GSENSOR_SDO	PIN_B16	SPI serial data output/I2C less significant bit of the device address	2.5V
GSENSOR_CS_n	PIN_C16	SPI enable, I2C/SPI mode selection: 1: SPI idle mode/I2C communication enabled, 0: SPI communication mode/I2C disabled	2.5V
GSENSOR_SCLK	PIN_A15	I2C serial clock/SPI serial port clock	2.5V
GSENSOR_INT1	PIN_B15	Interrupt pin 1	2.5V
GSENSOR_INT2	PIN_D17	Interrupt pin 2	2.5V

3.4.22 Micro SD Card Socket

The board supports Micro SD card interface with x4 data lines. It serves not only an external storage for the HPS, but also an alternative boot option for MAX 10 NEEK board. **Figure 3-34** shows signals connected between the HPS and Micro SD card socket.

Table 3-20 lists the pin assignment of Micro SD card socket to the MAX 10 FPGA.

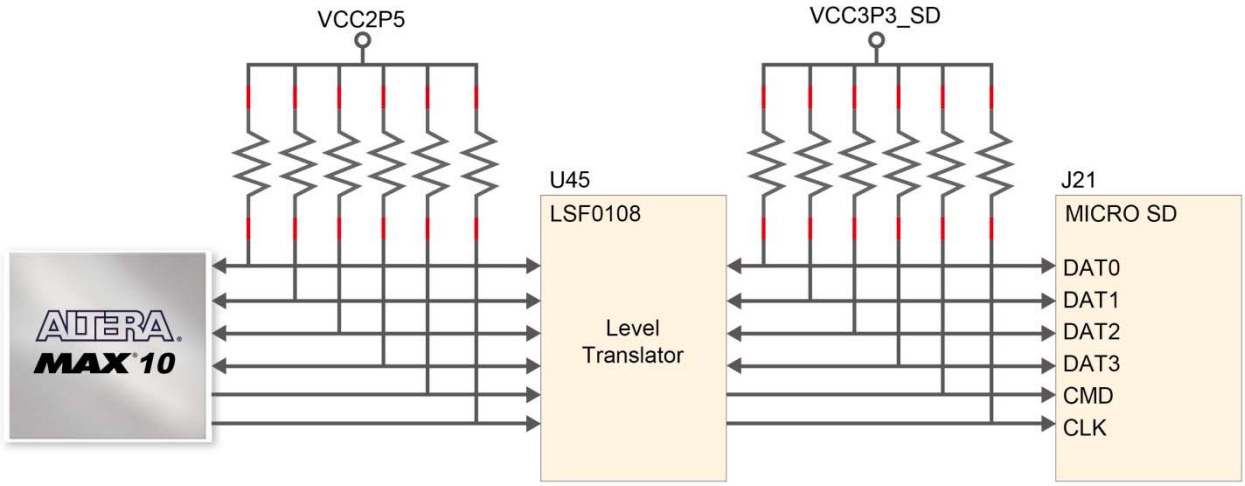

Figure 3-34 Connections between the MAX 10 FPGA and SD card socket

Table 3-20 Pin Assignment of Micro SD Card Socket

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SD_CLK	PIN_A16	SD Clock	2.5V
SD_CMD	PIN_C18	SD Command Line	2.5V
SD_DATA[0]	PIN_A17	SD Data[0]	2.5V
SD_DATA[1]	PIN_A18	SD Data[1]	2.5V
SD_DATA[2]	PIN_B17	SD Data[2]	2.5V
SD_DATA[3]	PIN_C17	SD Data[3]	2.5V

3.4.23 Power Distribution System

The MAX 10 NEEK is powered by Linear Technology's power solution which provides high-efficiency power management for FPGAs and SoCs. **Figure 3-35** shows the power tree of MAX 10 NEEK.

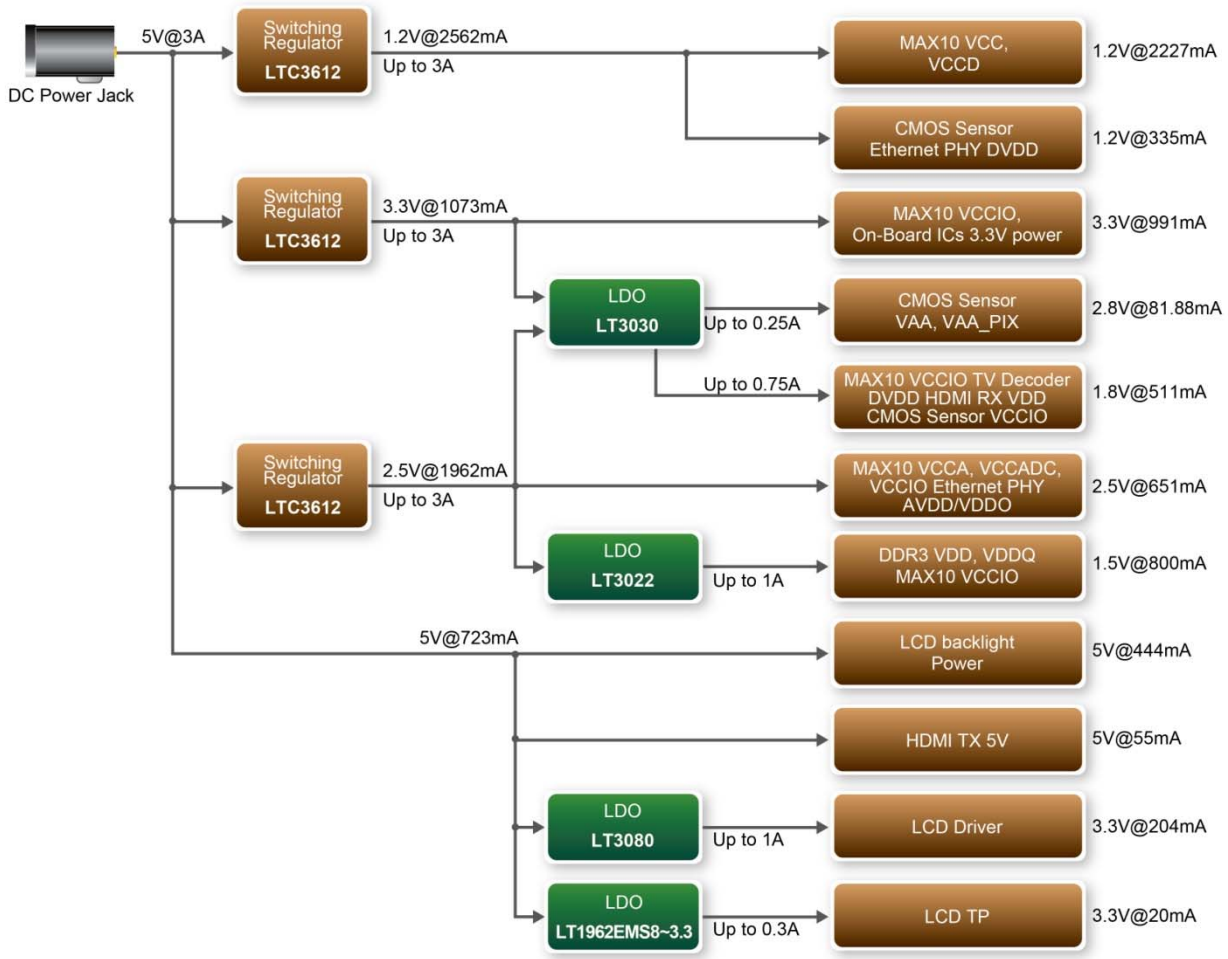

Figure 3-35 Power tree of MAX 10 NEEK

Chapter 4

NEEK10 System Builder

This chapter introduces the NEEK 10 System Builder to help users get started in creating their own projects in literally minutes. It also describes the design flow and includes an example for users to get familiar with the tool.

4.1 Introduction

The NEEK10 System Builder is a Windows-based utility. It is created to help users build a top project for NEEK10 within minutes. The generated Quartus II project files include:

- Quartus II project file (.qpf)
- Quartus II setting file (.qsf)
- Top-level design file (.v)
- Synopsis design constraints file (.sdc)
- Pin assignment document (.htm)

The above files generated by the NEEK10 System Builder can also prevent situations that are prone to compilation error when users manually edit the top-level design file or place pin assignment. The common mistakes that users encounter are:

- Board is damaged due to incorrect bank voltage setting or pin assignment.
- Board is malfunctioned because of wrong device chosen, declaration of pin location, or the direction is incorrect/forgotten.
- Performance degradation due to improper pin assignment.

4.2 General Design Flow

The design flow of building a Quartus II project for NEEK10 using the NEEK10 System Builder is illustrated in **Figure 4-1**. It gives users an overview about the steps, starting from launching the System Builder to configuring the FPGA. The left-hand side of the chart can be done within minutes. After users enter the design requirements, the NEEK10 System Builder will generate Quartus II project files, Quartus II setting file, top-level design file, Synopsis design constraint file, and the pin assignment document.

The top-level design file contains a top-level Verilog HDL 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 the I/O standard for each user-defined I/O pin. These files can be modified according to the project requirements. After the compilation is successful, users can download the .sof file to the developmenet board via JTAG interface using the Qaurtus II programmer.

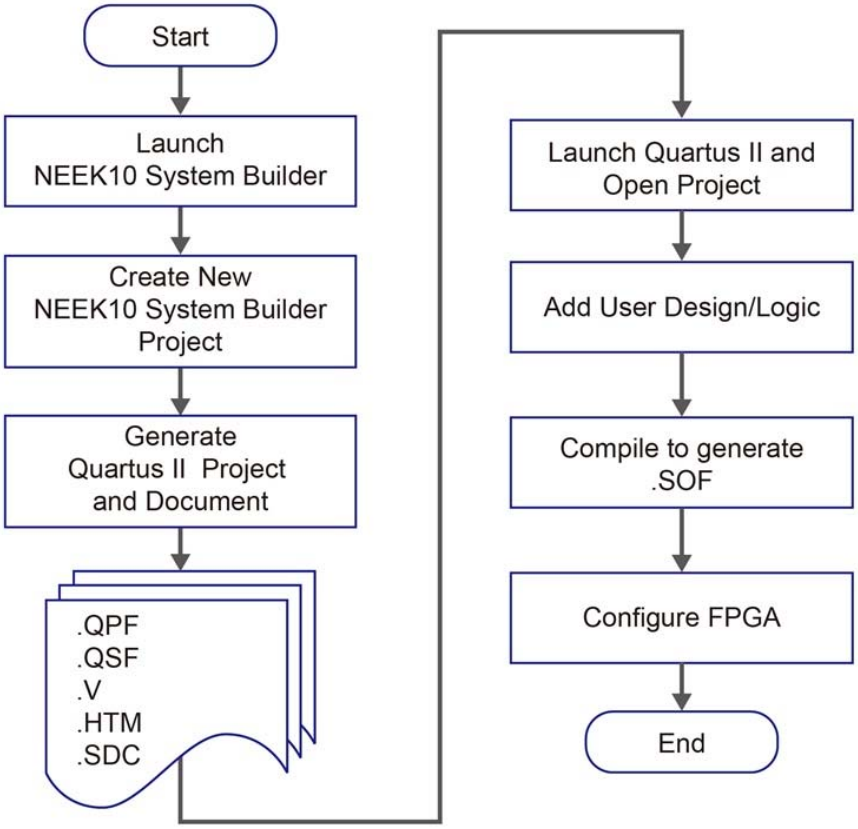

Figure 4-1 Design flow of building a project from the beginning to the end

4.3 Using NEEK10 System Builder

This section provides the procedures in details on how to use the NEEK10 System Builder.

■ Install and Launch the NEEK10 System Builder

The NEEK10 System Builder is located in the directory "Tools\SystemBuilder" of the NEEK10 System CD. Users can copy the entire folder to a host PC without installing the utility. After the execution of the NEEK10 SystemBuilder.exe on the host PC, a window will pop up, as shown in **Figure 4-2.**

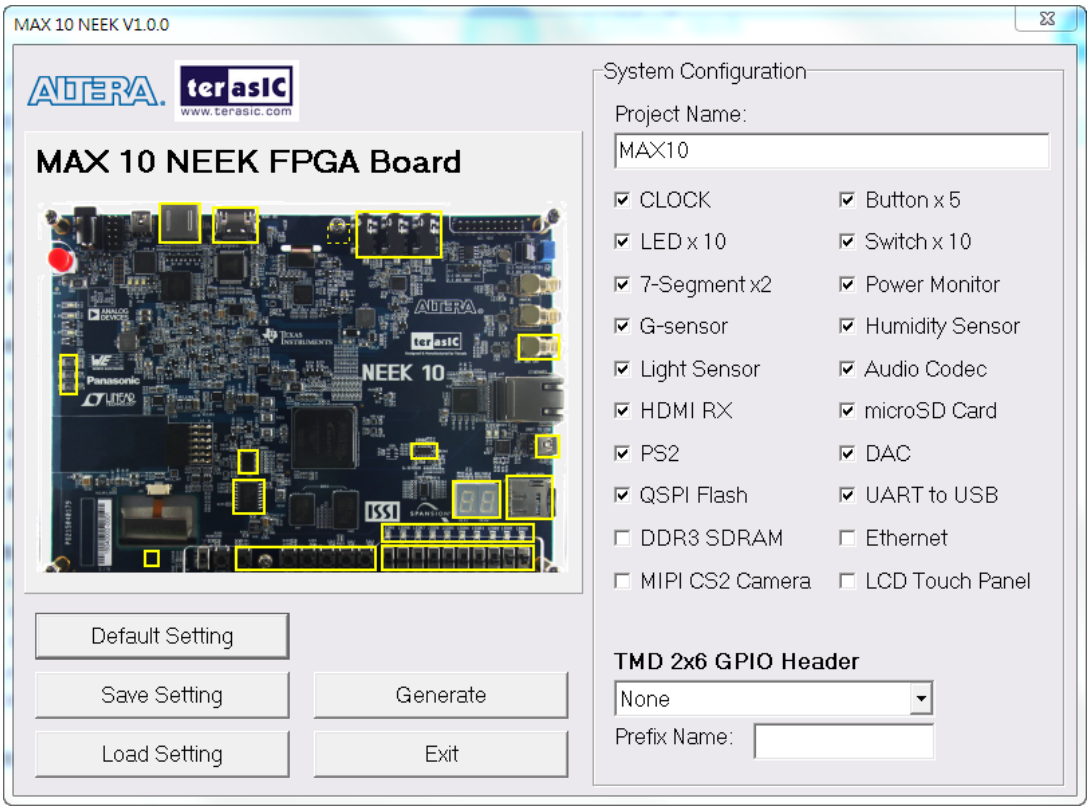

Figure 4-2 The GUI of NEEK10 System Builder

■ Enter Project Name

The project name entered in the circled area, as shown in **Figure 4-3**, will be assigned automatically as the name of the top-level design entity.

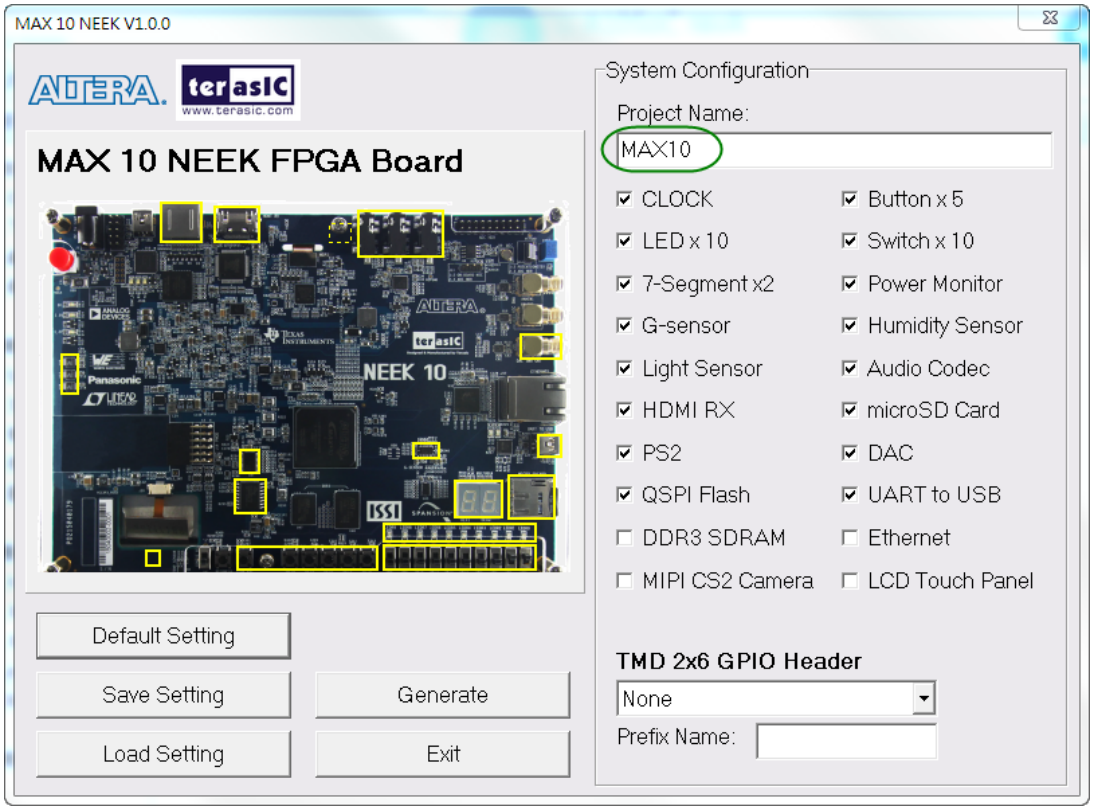

Figure 4-3 Enter the project name

■ System Configuration

Users are given the flexibility in the System Configuration to include one or more onboard peripherals in the project, as shown in **Figure 4-4**. If a component is enabled, the NEEK10 System Builder will automatically generate its associated pin assignment, including the pin name, pin location, pin direction, and I/O standard.

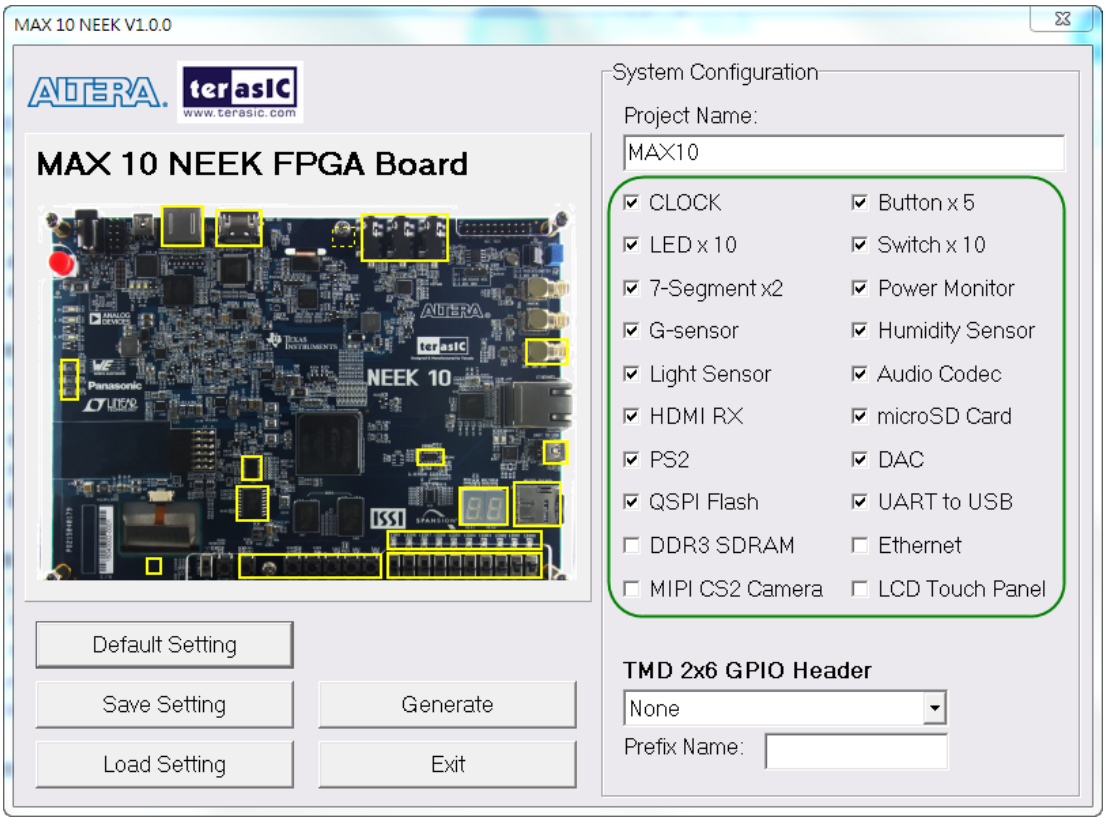

Figure 4-4 List of onboard peripherals in System Configuration

■ Project Settings

The NEEK10 System Builder also provides the option to load a setting or save the current board configuration in .cfg file, as shown in **Figure 4-5.**

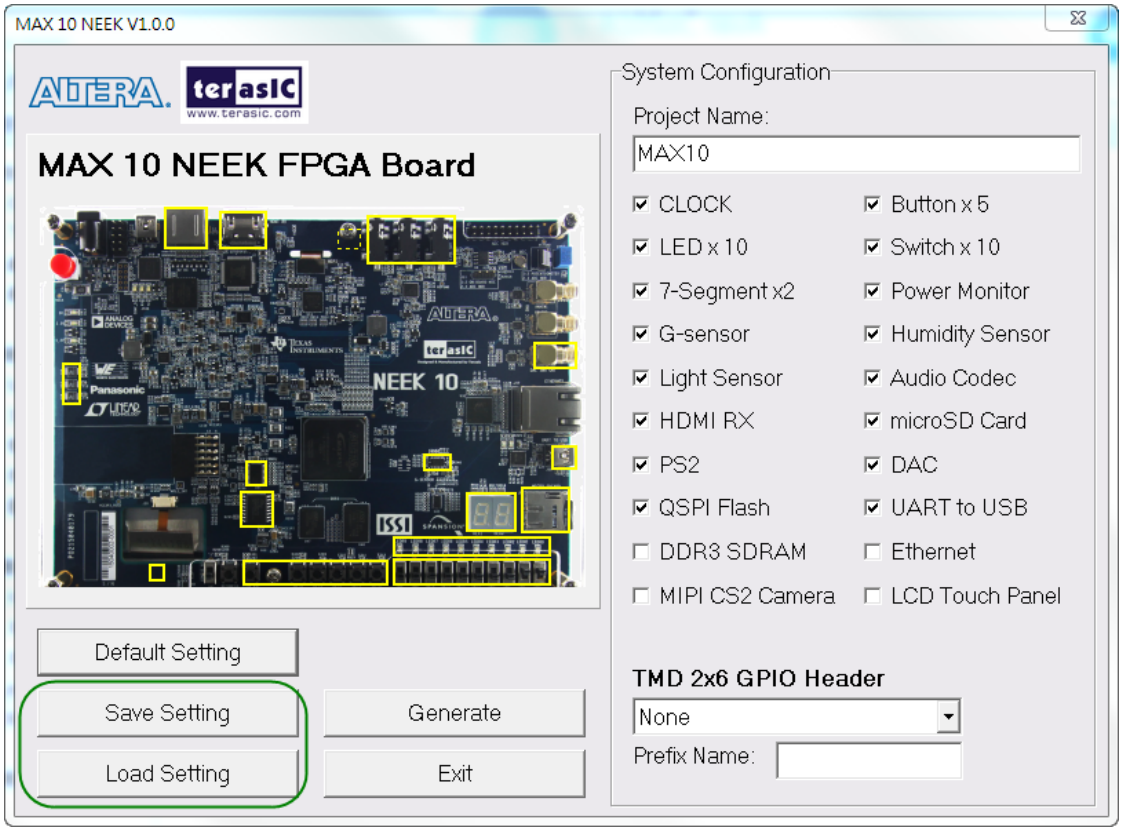

Figure 4-5 Manage project settings

■ Project Generation

When users press the *Generate* button, as shown in **Figure 4-6,** the NEEK10 System Builder will generate the corresponding Quartus II files and documents, as listed in Table 4-1 Files generated by the NEEK10 System Builder:

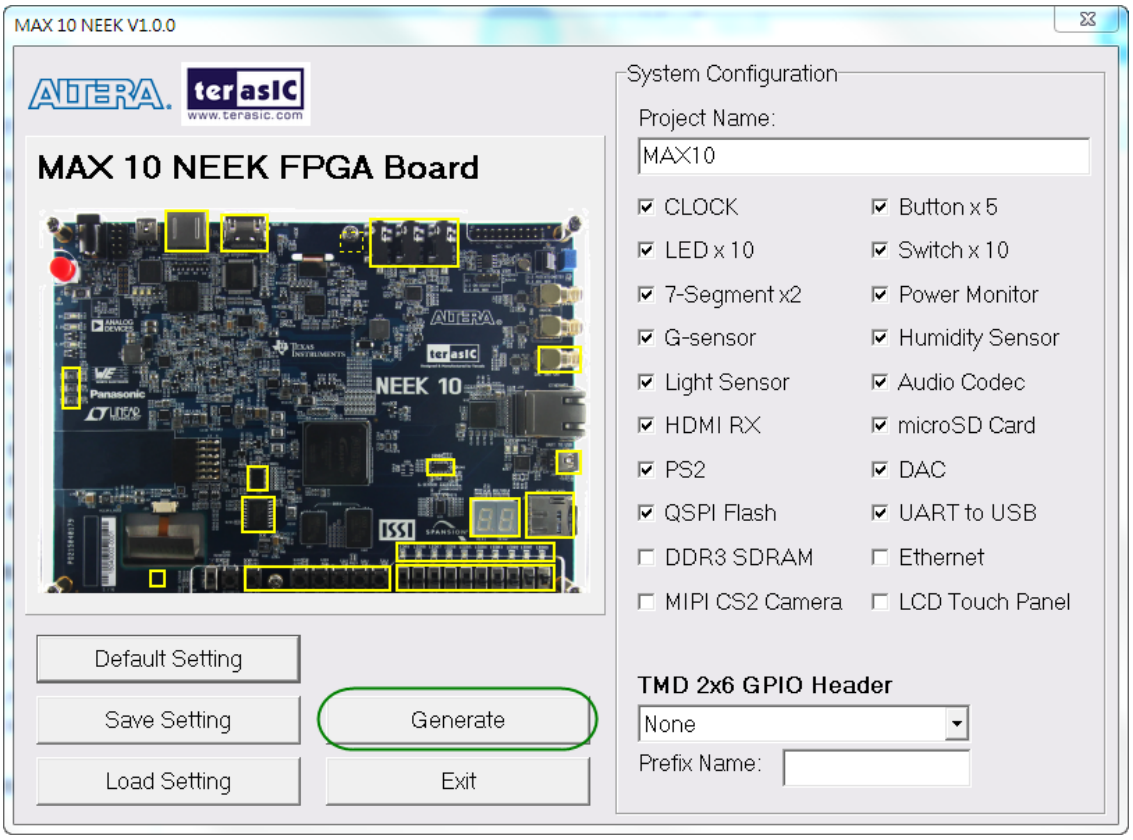

Figure 4-6 Generate Quartus project

Table 4-1 Files generated by the NEEK10 System Builder

No.	*Filename*	*Description*
1	<Project name>.v	Top-level Verilog HDL file for Quartus II
2	<Project name>.qpf	Quartus II project file
3	<Project name>.qsf	Quartus II setting file
4	<Project name>.sdc	Synopsis design constraints file for Quartus II
5	<Project name>.htm	Pin assignment document

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

Chapter 5

RTL Example Codes

This chapter provides examples of advanced designs implemented by RTL on the DECA board. These reference designs cover the features of peripherals connected to the FPGA, such as PS/2 mouse, Power monitor, ADC/DAC application, HDMI input and display. All the associated files can be found in the directory \Demonstrations of System CD. Note: The output files generated after compilation in Quartus II e.g. .sof and .pof files, are saved in the folder "output_files" under the directory of demo project.

5.1 PS/2 Mouse Demonstration

We offer this simple PS/2 controller coded in Verilog HDL to demonstrate bidirectional communication between PS/2 controller and the device, the PS/2 mouse. You can treat it as a how-to basis and develop your own controller that could accomplish more sophisticated instructions, like setting the sampling rate or resolution, which need to transfer two data bytes.

More information about the PS/2 protocol can be found on various websites.

■ Introduction

PS/2 protocol uses two wires for bi-directional communication. One is the clock line and the other one is the data line. The PS/2 controller always has total control over the transmission line, but it is the PS/2 device which generates the clock signal during data transmission.

■ Data Transmission from Device to the Controller

After the PS/2 mouse receives an enabling signal at stream mode, it will start sending out displacement data, which consists of 33 bits. The frame data is cut into three sections and each of them contains a start bit (always zero), eight data bits (with LSB first), one parity check bit (odd check), and one stop bit (always one).

The PS/2 controller samples the data line at the falling edge of the PS/2 clock signal. This is implemented by a shift register, which consists of 33 bits, but be cautious with the clock domain crossing problem.

■ Data Transmission from the Controller to Device

When the PS/2 controller wants to transmit data to device, it first pulls the clock line low for more than one clock cycle to inhibit the current transmission process or to indicate the start of a new transmission process, which is usually called as inhibit state. It then pulls low the data line before releasing the clock line. This is called the request state. The rising edge on the clock line formed by the release action can also be used to indicate the sample time point as for a start bit. The device will detect this succession and generates a clock sequence in less than 10ms time. The transmit data consists of 12bits, one start bit (as explained before), eight data bits, one parity check bit (odd check), one stop bit (always one), and one acknowledge bit (always zero).

After sending out the parity check bit, the controller should release the data line, and the device will detect any state change on the data line in the next clock cycle. If there's no change on the data line for one clock cycle, the device will pull low the data line again as an acknowledgement which means that the data is correctly received.

After the power-on cycle of the PS/2 mouse, it enters into stream mode automatically and disable data transmit unless an enabling instruction is received. **Figure 5-1** shows the waveform while communication happening on two lines.

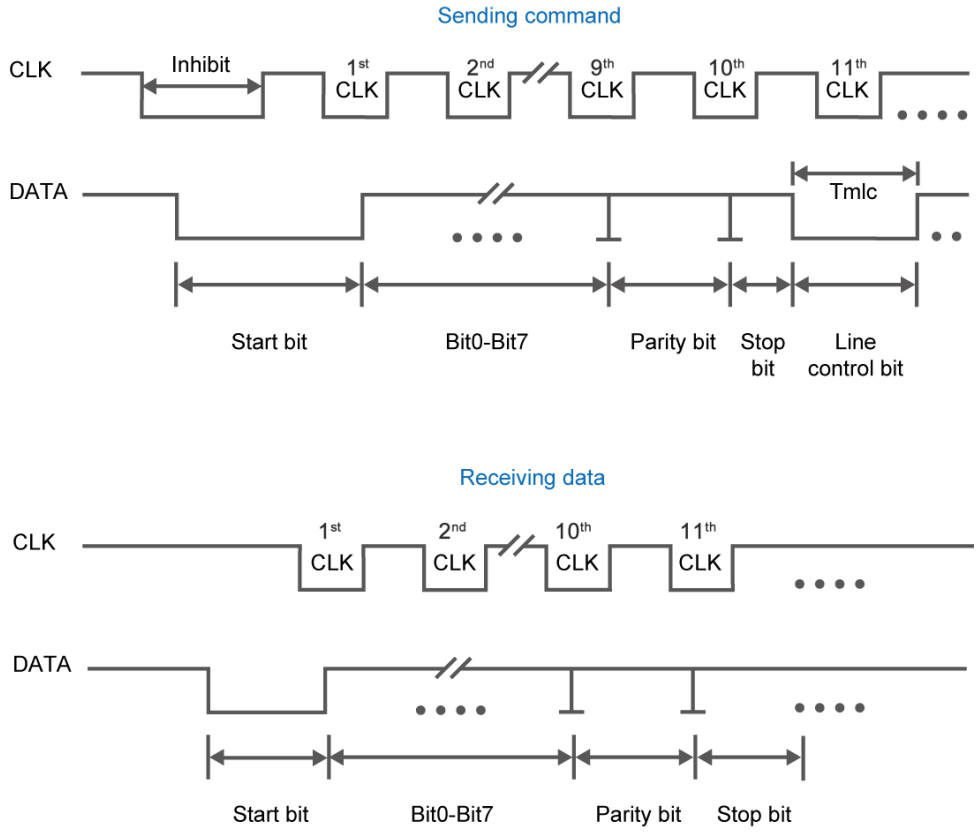

Figure 5-1 Waveform of Clock and Data Signals during Data Transmission

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: ps2_mouse
- Bitstream used: ps2_mouse.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\ps2_mouse\demo_batch
- Batch file: ps2_mouse.bat
- FPGA configuration file: ps2_mouse.sof

■ Demonstration Setup

- Please make sure Quartus II and USB-Blaster II driver are installed on the host PC.
- Connect the USB cable from the USB-Blaster II port (J8) on the NEEK board to the host PC and power on the NEEK board.
- Execute the demo batch file "ps2_mouse.bat" under the folder Demonstrations\ps2_mouse \demo_batch\.
- Plug in the PS/2 mouse.
- Press KEY0 to enable data transfer.
- Press KEY1 to clear the display data cache.
- The 7-segment display shows X displacement when SW0 is low, and Y displacement when SW0 is high. The display should change when the PS/2 mouse moves.
- The LEDR[2:0] will blink according to **Table 5-1** when the left-button, right-button, and/or middle-button is pressed.

Table 5-1 Description of 7-segment Display and LED Indicators

Indicator Name	*Description*
LEDR0	**Left button press indicator**
LEDR1	**Right button press indicator**
LEDR2	**Middle button press indicator**
HEX0	**Low byte of X/Y displacement**
HEX1	**High byte of X/Y displacement**

5.2 Power Monitor

There are three built-in power monitor (LTC2990) on the NEEK10 board to observe total of six voltage and current buses. Each LTC2990 can monitor two sets of voltage and current rail . The six sets correspond to the core voltage/current and I/O voltage/current of MAX 10 device and the power rails of 5V/3.3V/1.2V/1.5V of the system. This demonstration uses these six buses to calculate the power consumption and display the result on the two 7-segments. The voltage measured from the sense resistor will be digitized through the power monitor and feed into the MAX 10 device via I2C protocol. The current is calculated based on the value of the sense resistor and the voltage measured. The power consumption of each bus can be calculated accordingly and switched via SW[2:0] onboard.

■ Function Block Diagram

Figure 5-2 is the function block diagram of this demonstration. The power bus monitored are 5VCORE/2.5VCCIO/1.5VCCIO/1.2VCC/5VCC/3.3VCCIO. The sensing resistors connected between the source and the system are 0.01Ω/0.01Ω/0.01Ω/0.01Ω/0.01Ω/0.01Ω, respectively . When the system is powered on, there will be current floating through each bus and causing voltage drop across the sensing resistors. The power monitor (LTC2990) will convert the voltage drop from analog to digital and the result can be retrieved via I2C protocol. The module POWER_I2C can read out the result of each power monitor and calculate the power consumption accordingly. The power bus can be selected by switching SW[2:0] and the value will be displayed on the two 7-segments on the NEEK10 board. The unit is in mW

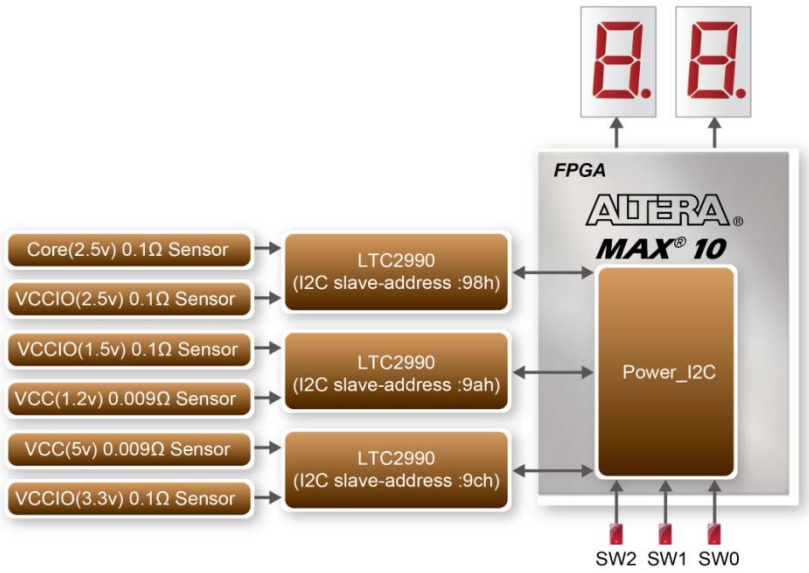

Figure 5-2 Block diagram of Power Monitor demonstration

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: power_monitor
- Bitstream used: power_monitor.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\ power_monitor\demo_batch
- Batch file: test.bat
- FPGA configuration file: power_monitor.sof

■ Demonstration Setup

- Please make sure Quartus II is installed on the host PC.
- Connect the NEEK10 board (J8) to the host PC with USB cable and install the USB-Blaster II driver.
- Plug in the 5V adapter to the NEEK10 Board and power it up.
- Execute the demo batch file " test.bat" from the directory \power_monitor\demo_batch.
- The two 7-segments on the NEE10 board will display mW of the bus chosen in decimal, as **Figure 5-3**.

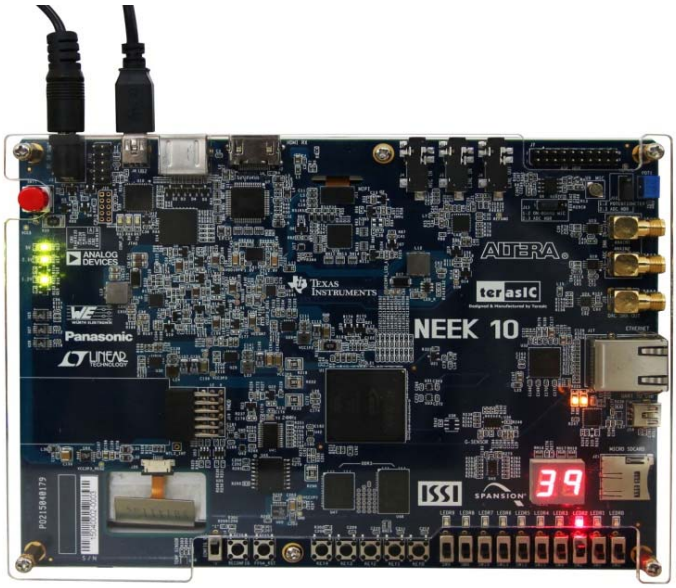

Figure 5-3 Display power (mW) on the 7-segments and LED

- The SW[2:0] can choose which bus to be displayed on the two 7-segments according to the Table 5-2.

Table 5-2 Switch setting for power

SUB POWER	SW[2:0]
1.2VCC	• 000
1.5VCCIO	001
2.5VCCIO	010
2.5VCORE	011
3.3VCCIO	100
5.0VCC	101

5.3 ADC Potentiometer

Nowadays voltage and current monitors play a significant role in high-reliability system. Most of applications can be implemented by an Analog-to-Digital Converter (ADC). MAX10 NEEK provides a potentiometer demonstration using the ADC in MAX10. This ADC solution consists of hard IP blocks in MAX 10 device and soft logic through Altera Modular ADC IP core. Figure 5-4 shows the block diagram of ADC hard IP block in MAX10 device.

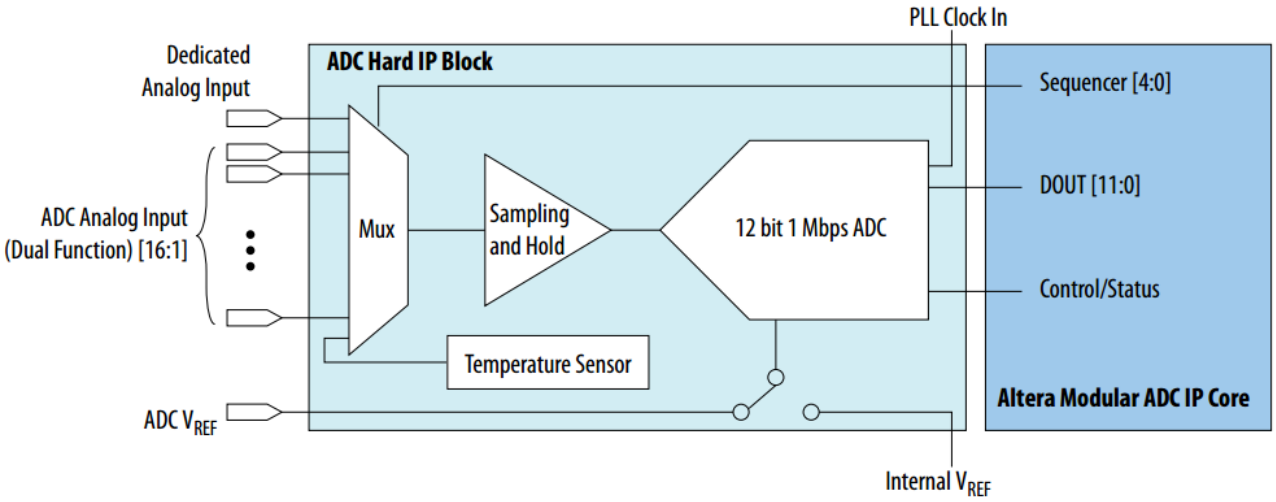

Figure 5-4 Block diagram of ADC hard IP block

This demo uses 2nd ADC of MAX10 on channel 8. The ADC settings are shown in Figure 5-5 .

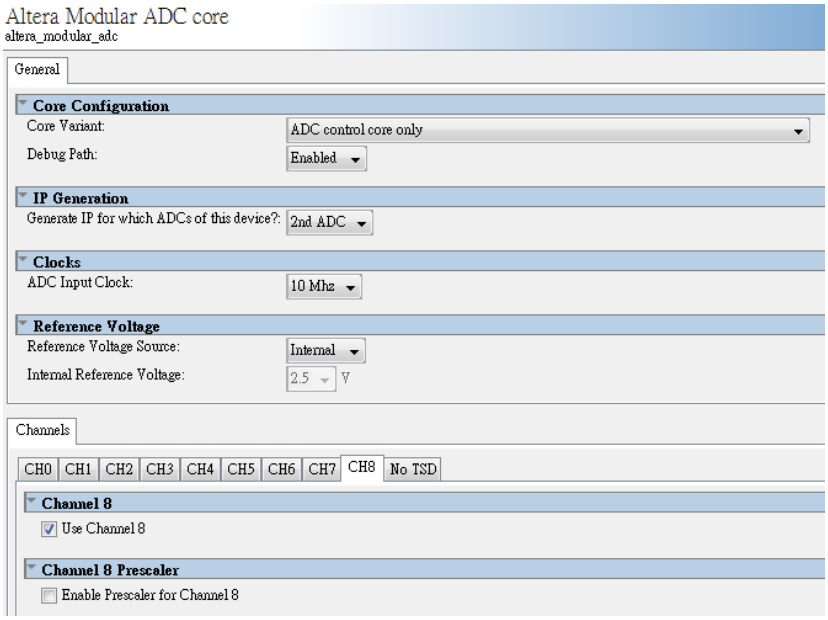

Figure 5-5 Settings of ADC hard IP

The MAX10 NEEK has a Variable Resistor (VR) onboard, which acts as a potentiometer in this demonstration. **Figure 5-6** shows the block diagram of Power Monitor demonstration. The ADC reference clock running at 10MHz is generated by PLL. It feeds into the ADC Hard IP in MAX10 device. The analog voltage input comes from the VR controls the voltage level. The control logic within the ADC Hard IP reads the digitized voltage data. It then converts the data and displays the level value on two 7-segments. Since none of the dot points of two 7-segments is connected to the MAX 10, so HEX1 and HEX0 shows the decimal point and the first digit after the decimal point respectively.

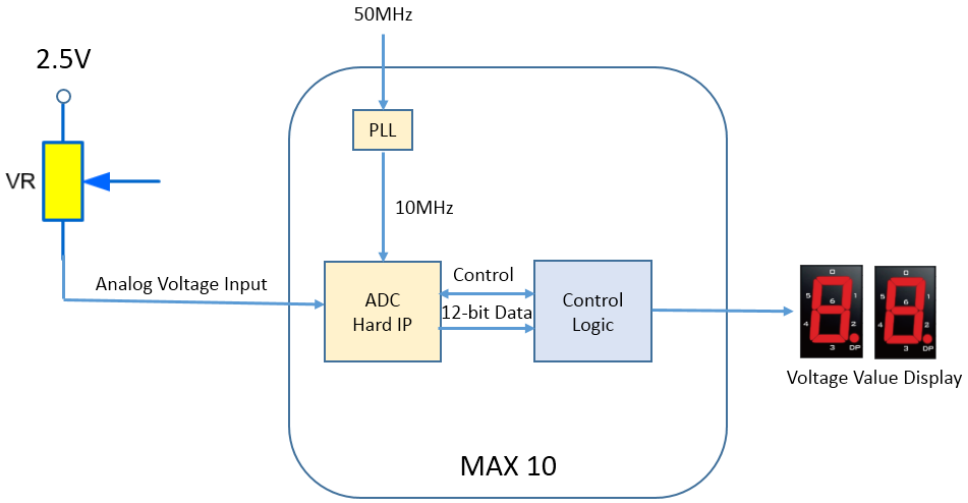

Figure 5-6 Block diagram of ADC Potentiometer

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations\adc_potentionmeter
- Bitstream used: adc_potentionmeter.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\adc_potentionmeter \demo_batch
- Batch file: test.bat
- FPGA configuration file: adc_potentionmeter.sof

■ Demonstration Setup

- Please make sure Quartus II and USB-Blaster II driver are installed on the host PC.
- Connect the USB cable from the USB-Blaster II port (J8) on the NEEK board to the host PC.
- Power on the NEEK board.
- Execute the demo batch file "test.bat" under the folder Demonstrations\ adc_potentionmeter \demo_batch.
- Rotate the VR resistor POT1 with a screwdriver. HEX1 and HEX0 will display the voltage value.

5.4 DAC Demonstration

This demonstration uses the 16-bit Digital to analog converter(DAC) built-in the MAX 10 device to generate square wave in 8 difference frequencies. The signal coming out of the SMA connector on NEEK10 board is transmitted to the oscilloscope. The oscilloscope will display the square wave in different frequencies by switching SW[2:0] on the NEEK10 board.

■ Function Block Diagram

Figure 5-7 is the function block diagram of this demonstration. The source data in parallel is converted to serial data by the DAC16 module. The DAC chip (DAC8551) then converts the serial data from digital to analog. The analog signal coming out of the DAC SMA connector is connected to the oscilloscope and shown in square wave. Users can switch the SW0~2 to change the frequency

of the square wave.

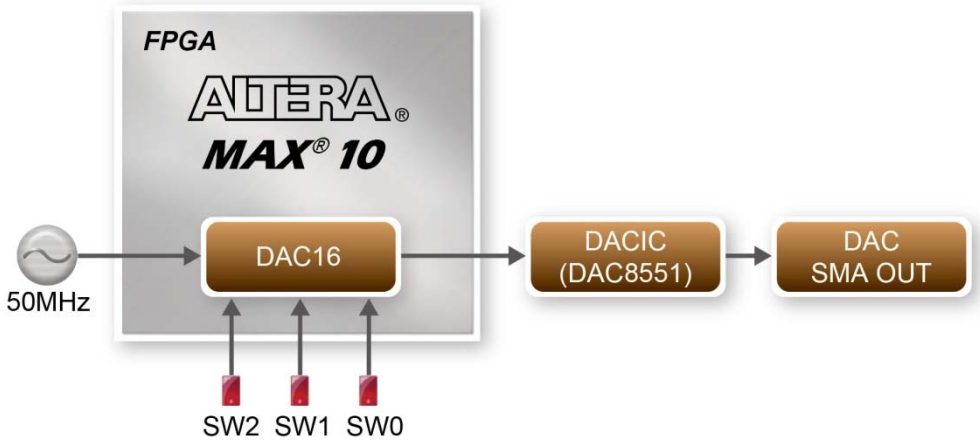

Figure 5-7 Block diagram of the DAC demo

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations\ dac_sma
- Bitstream used: dac_sma.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\ dac_sma\demo_batch
- Batch file: test.bat
- FPGA configuration file: dac_sma.sof

■ Demonstration Setup

- Please make sure Quartus II is installed on the host PC.
- Connect the NEEK10 board (J8) to the host PC with USB cable and install the USB-Blaster II driver.
- Plug in the 5V adapter to the NEEK10 Board and power it up.
- Execute the demo batch file " test.bat" from the directory \ dac_sma\demo_batch.
- Connect the probe of the oscilliscope to the DAC SMA OUT of the NEEK 10 board and adjust

the display until the square wave is visible, as **Figure 5-8** and **Figure 5-9**.

- Switch SW[2:0] from 000 to 111 and the frequency of the square will be changing. The square-wave frequency is twice higher. When SW[2:0]=000, the square wave frequency is at about 2.6KHz; When SW[2:0]=111, the frequency is about 112KHz.

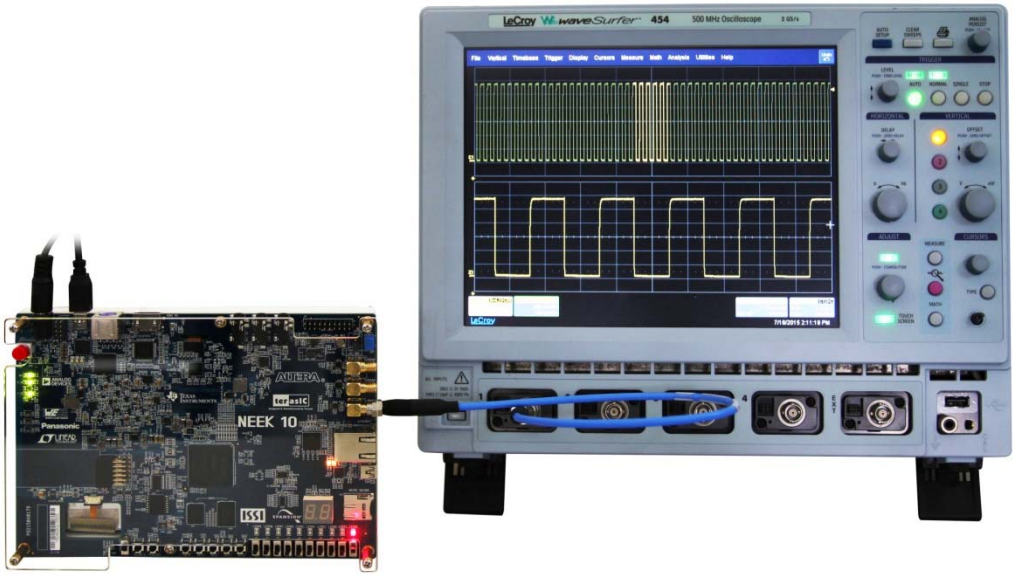

Figure 5-8 Use the oscilloscope to observe the square wave

Figure 5-9 Probe DAC SMA OUT from the oscilloscope

5.5 ADC/MIC/LCD Demonstration

There is a high-sensitive microphone on the NEEK 10 board to receive the surrounding sound. After the sound is collected and amplified approximately 392 times, as shown in **Figure 5-10**. It is feed into the ADC of MAX 10 device. The digitized waveform will be displayed on the LCD. Meanwhile the signal will be sent to the Line-out via audio codec and DAC SMA OUT connector. Both of which can be connected to an external speaker. The data will also be processed based on the volume to be displayed on the 10 LEDs onboard.

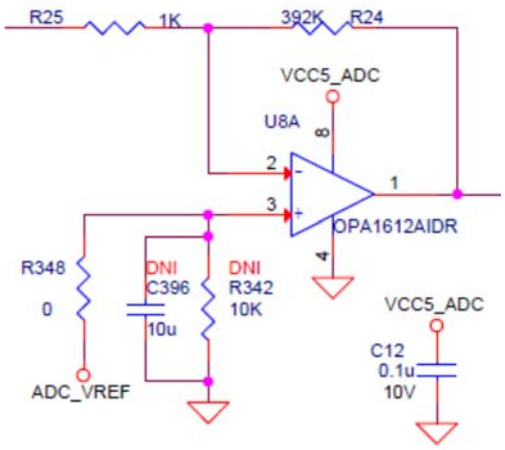

Figure 5-10 Onboard microphone amplifier with Gain (R24/R25) = 392

■ Function Block Diagram

Figure 5-11 is the function block diagram of this demonstration. The built-in MIC is amplified approximately 392 times via two operational amplifiers. The signal is then feed into the ADC of MAX 10 device for conversion. This demonstration uses the timing from the audio codec ((TLV320AIC3254) via I2S protocol to sync the entire system. The module SPI_CTL sets the registers for the audio codec. The module SOUND_SUM syncs the digitized signal coming out of the ADC of the MAX 10 device with the system and converts the data format i.e. adjust unsign 12-bit to sign 16-bit etc. The module SOUND2LCD converts the digitized sound signal into the format of LCD timing to the LCD panel to be displayed in graphical sound wave. The module DAC16 converts the digitized signal in parallel to 16-bit serial format for the DAC chip (DAC8551) to the Line-out via audio codec (TLV320AIC3254). The module LED_METER displays the volume of the sound on the 10 LEDs onboard.

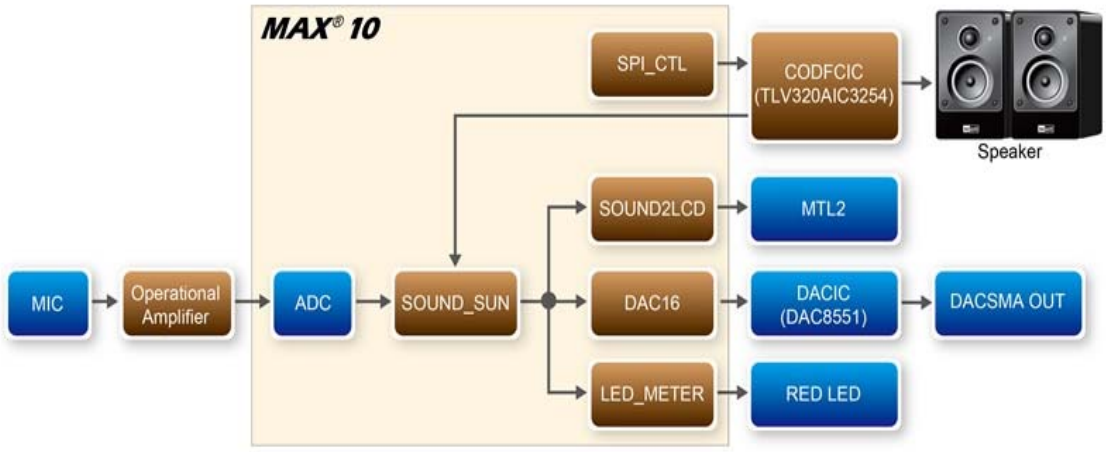

Figure 5-11 Block diagram of the ADC/MIC/LCD demonstration

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations\ adc_mic_lcd
- Bitstream used: adc_mic_lcd.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\adc_mic_lcd\demo_batch
- Batch file: test.bat
- FPGA configuration file: adc_mic_lcd.sof

■ Demonstration Setup

- Please make sure Quartus II is installed on the host PC.
- Connect the NEEK10 board (J8) to the host PC with USB cable and install the USB-Blaster II driver.
- Plug in the 5V adapter to the NEEK10 Board and power it up.
- Execute the demo batch file " test.bat" from the directory \adc_mic_lcd\demo_batch.
- The sound wave from the MIC can be observed on the LCD and probed through DAC SMA OUT from the oscilloscope, or can also connect external speaker to Line-out to hear the sound, as **Figure 5-12**. The volume of the sound from the MIC is displayed digitally on LEDR0~9, as

Figure 5-13.

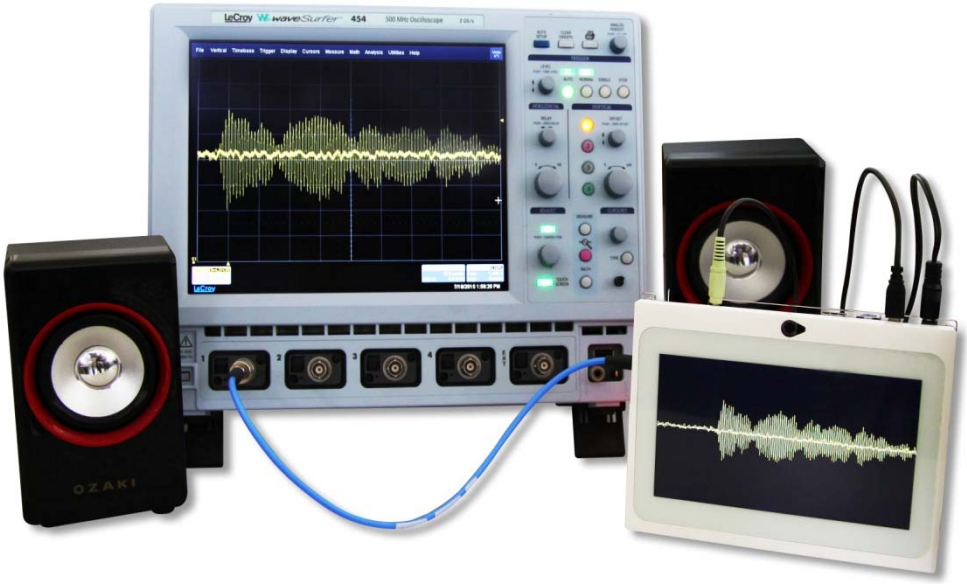

Figure 5-12 The waveform of onboard MIC is displayed on both LCD and oscilloscope. Its sound is played out from the speaker

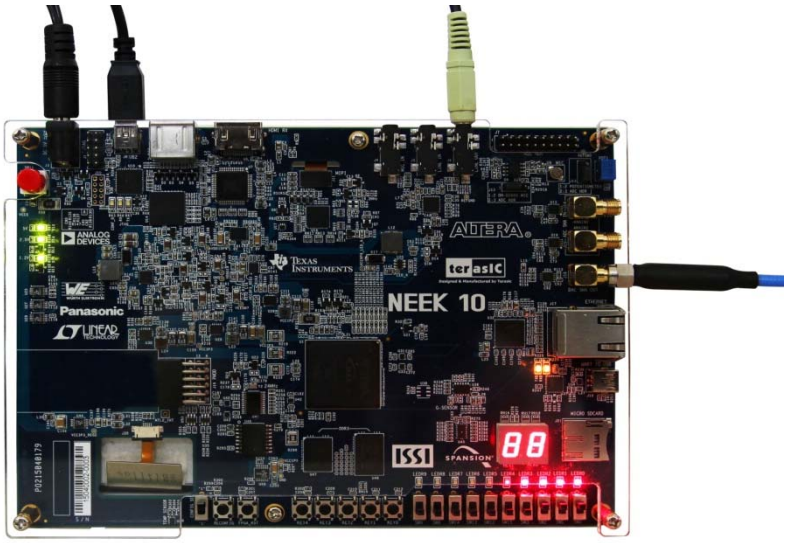

Figure 5-13 LEDR0~9 displays the volume level of onboard MIC

5.6 HDMI RX Demonstration

We now provide more details on the HDMI Receiver(ADV7611) on the MAX 10 NEEK. We will show you how to use MAX 10 NEEK as a HDMI Display in this demo. When the HDMI Receiver receives and decrypts the data, the data can be divided into both audio and video parts where the audio part is played after being encrypted by Audio Codec(TLV320AIC3254_0) and the video will be displayed on the multi-touch LCD panel.

■ Function Block Diagram

Figure 5-14 shows the system block diagram of this reference design where 7"LCD refers to Terasic multi-touch panel. HDMI RX(ADV7611) is a chip to decrypt HDMI video and audio data. Before decrypting HDMI data, a proper setting is necessary where HDMIConfig(I2C) module is used to configure the ADV7611 chip. This demo uses the Video and Image Processing (VIP) IP provided by Altera which requires specific video data format (Avalon-ST-Video image data format). Video In module in the above block diagram is to convert the received video data to Avalon-ST-Video format. Because MAX 10 NEEK Display has a resolution of 800x480, therefore no matter what the input video resolution is, the output has to be 800x480. The Scalar module is to convert resolution of any kind to 800x480.

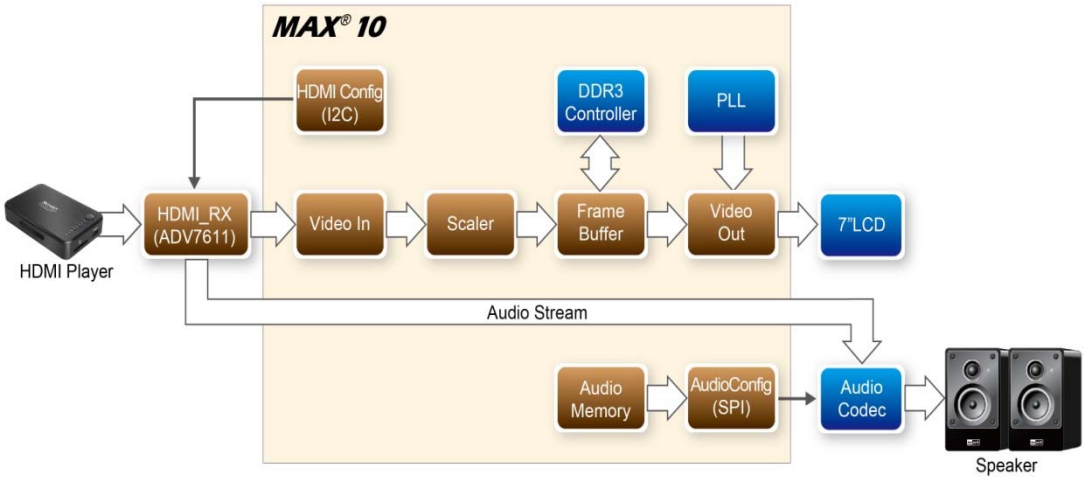

Figure 5-14 Block Diagram of the HDMI RX demonstration

Video data input would require a data buffer and we use DDR3 controller to communicate with the external DDR3 memory onboard. The final video outputs to the multi-touch panel. The dot clock of this LCD panel is 33MHz and therefore it would require a PLL which can generate 33MHz. When there is no video data, the screen will display a color of blue, representing no video data input.

The audio signal decrypted by HDMI_RX(ADV7611) will be sent to Audio Codec. In order for the Audio Codec to play the audio, proper settings would have to be made. In this demo, the setting has been pre-written in the Audio memory. The AudioConfig module in the block diagram will read out the data stored in the Audio Memory and use SPI protocol to configure the Audio Codec chip. If you wish to change the Audio memory setting, kindly refer to the following section "Audio Memory Setting".

■ Audio Memory Setting

When you need to modify internal setting of audio memory, you can open up the Loop.hex file under Audio_SPI directory using Quartus. When opening up the file, you will be requested to input the word size, type in 16 and click OK and you will see Audio memory Table in **Figure 5-15**.From the Audio memory Table, it can be seen the data is 0100 at 00+2 position. This means it will configure the Audio Codec Address 0x01 register (the first 2bit) to 0x00 (the latter 2bit). For more details on register settings, please kindly refer to TLV320AIC3254 Application Reference Guide included in the System CD.

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
00	0000	0101	0100	0400	0B81	0C82	0D00	0E80	.....
08	1C00	3C02	1E84	3FD5	4000	4100	4200	447D	.....
10	4508	46FF	0001	02C9	090C	0108	0201	0A40	.....
18	0E08	0F08	1203	1303	0000	0000	0000	0000	.....

Figure 5-15 Audio Memory Table

■ HDMI Receiver(ADV7611)

After you set up ADV7611, don't forget to set up EDID as well. If you plan to use EDID RAM inside ADV7611, you would have to load relevant parameters. More info on the internal EDID RAM can be referred to HDMIConfig(I2C) module.

■ Audio Codec

The audio format after being decrypted by HDMI RX is I2S format (default) as shown in **Figure 5-16**. Audio Codec configuration setting can be divided to either page0 or page1. During the configuration process, you would have to pay attention if the current setting is in page0 or page1. Paying attention to such details can help you set up Audio Codec more efficiently.

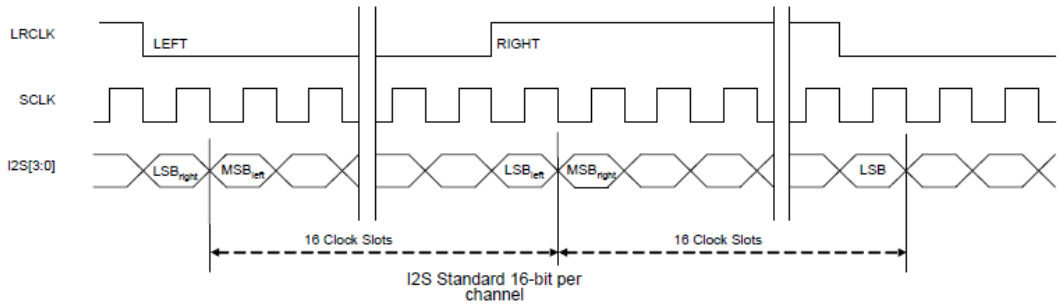

Figure 5-16 I2S Standard Audio – 16 bit per channel

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations\ hdmi_rx_lcd
- Bitstream used: hdmi_rx_lcd.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\ hdmi_rx_lcd \demo_batch
- Batch file: test.bat
- FPGA configuration file: hdmi_rx_lcd.sof

■ Demonstration Setup

- Make sure both Quartus II and USB-Blaster II driver are installed on your PC.
- Use a HDMI DVD player and use HDMI cable to connect the MAX 10 NEEK to the player.
- Power on the MAX 10 NEEK board.
- Use File Manager to locate the "hdmi_rx_lcd\demo_batch" folder. Launch the configuration and program download process by double clicking "test.bat" batch file. This will configure the FPGA, download the demo application to the board and start its execution. After it's done, the screen should look like the one shown in **Figure 5-17**.

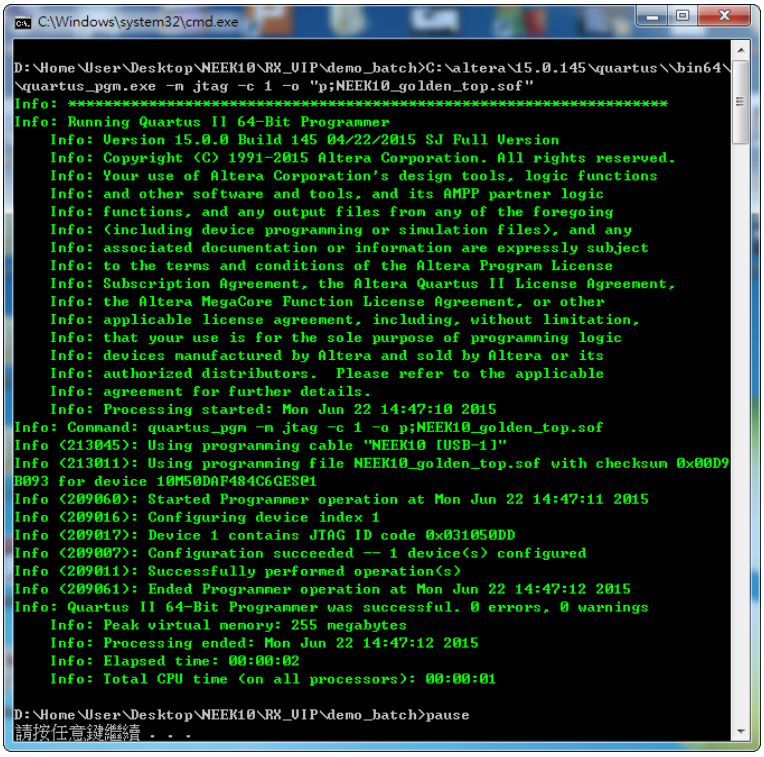

Figure 5-17 Launching The hdmi_rx_lcd Demo

If there is no video input, the screen will display a color of blue. Once there is video input, the screen will show the video. One thing to note is that we do not provide the HDCP KEY and this would cause issues to some specific high resolution video which not being able to play correctly. Therefore in that scenario, we would suggest you contact HDCP and apply for authorization. For more details, you can check at http://www.digital-cp.com/. **Figure 5-18** shows the setup of the demo.

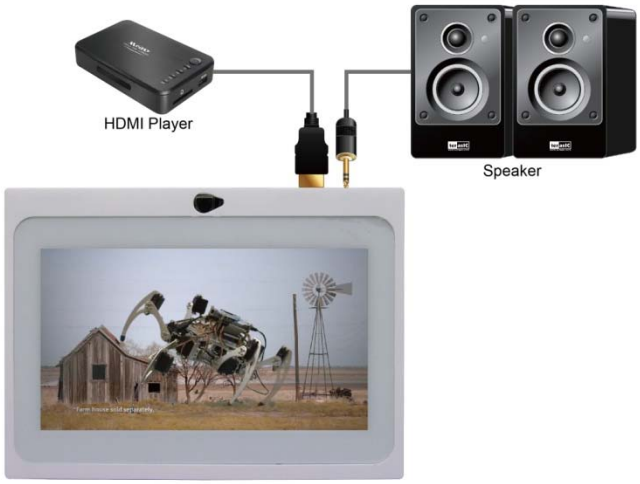

Figure 5-18 The Demo Setup

Chapter 6

NIOS Based Example Codes

There are several NIOS based examples for users to get started and try them on the MAX 10 NEEK board. All the NIOS based examples can be found in the system CD under the folder named **Demonstrations**. Users are free to use or modify these example for personal use or education purpose. Note: The output files generated after compilation in Quartus II e.g. .sof and .pof files, are saved in the folder "output_files" under the directory of demo project. The workspace of Nios II Eclipse project is located in the folder "software" under the directory of demo project.

6.1 Power Monitor

The power monitor demo shows how to measure the power consumed through the onboard power monitor chip LTC2990. There are three LTC2990 to monitor the following power rails:

- 3.3V VCCIO
- 2.5V Core
- 2.5V VCCIO
- 1.5V VCCIO
- 1.2V VCC

The power monitor chip LTC2990 communicates with the FPGA via I2C protocol. The I2C OpenCore IP is used in this demonstration for the MAX 10 device to communicate with the three LTC2990, which have different I2C slave address 98h, 9Ah, and 9Ch.

■ Block Diagram

Figure 6-1 shows the system block diagram of this demonstration. The NIOS program is stored in the onchip memory and the Nios II processor is running at 50 MHz. The I2C library is located in the files named I2C_core.cpp and I2C_core.h. The I2C OpenCore IP is located in the folder “ip/i2c_opencores” under the project directory.

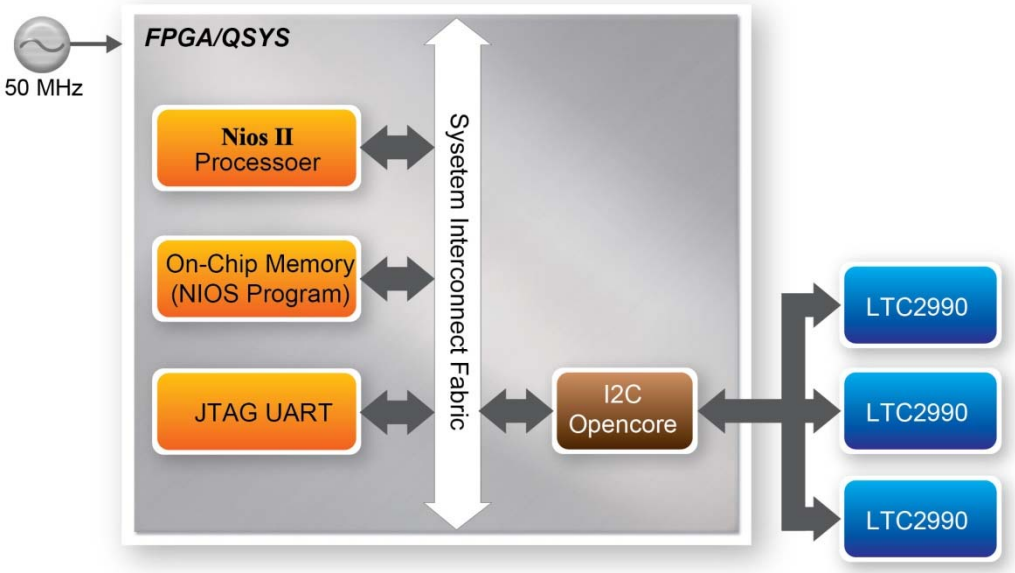

Figure 6-1 Block diagram of Power Monitor demonstration

Figure 6-2 illustrates the idea of how to measure the current for each power rail. A current sense resistor $\mathbf{R_{SENSE}}$ is added to the path of each power rail. LTC2990 measures the voltage difference (V1-V2) and calculate the current based on the formula below.

Current = (V1-V2) / $\mathbf{R_{SENSE}}$

The power consumption can be calculated by the following formula:

Power Consumption = $\mathbf{V_{CC}}$ x Current = $\mathbf{V_{CC}}$ x (V1-V2) / $\mathbf{R_{SENSE}}$

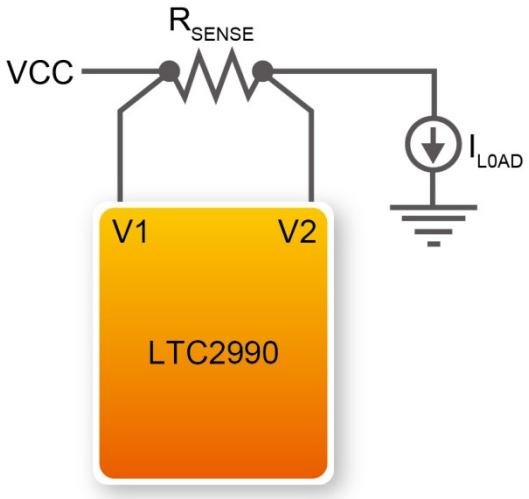

Figure 6-2 Schematic of current sense

Figure 6-3 shows the register content of LTC2990. The voltage difference (V1-V2) measured are written into two registers, MSB (register 06h) and LSB (register 07h). The most significant bit (7th bit) of MSB register is the data_valid bit, which indicates whether the current register content has been accessed since the result was last written to the register. This bit will be "set" i.e. 1 when the register content is renewed and "cleared" i.e. 0 when the register content is accessed. Bit 6 of the MSB register is the sign bit, Bit 5 through 0 represent the result of bits D[13:8] in two's complement conversion. The LSB register holds the conversion bits D[7:0]. The following equations are used to convert the register values to get the differential voltage:

$$V_{DIFFERENTIAL} = D[14:0] \bullet 19.42\mu V, \text{ if Sign} = 0$$
$$V_{DIFFERENTIAL} = (\overline{D[14:0]} + 1) \bullet -19.42\mu V, \text{ if Sign} = 1$$

REGISTER ADDRESS*†	REGISTER NAME	READ/WRITE	DESCRIPTION
00h	STATUS	R	Indicates BUSY State, Conversion Status
01h	CONTROL	R/W	Controls Mode, Single/Repeat, Celsius/Kelvin
02h	TRIGGER**	R/W	Triggers an Conversion
03h	N/A		Unused Address
04h	T_{INT} (MSB)	R	Internal Temperature MSB
05h	T_{INT} (LSB)	R	Internal Temperature LSB
06h	V1 (MSB)	R	V1, V1 – V2 or T_{R1} MSB
07h	V1 (LSB)	R	V1, V1 – V2 or T_{R1} LSB
08h	V2 (MSB)	R	V2, V1 – V2 or T_{R1} MSB
09h	V2 (LSB)	R	V2, V1 – V2 or T_{R1} LSB
0Ah	V3 (MSB)	R	V3, V3 – V4 or T_{R2} MSB
0Bh	V3 (LSB)	R	V3, V3 – V4 or T_{R2} LSB
0Ch	V4 (MSB)	R	V4, V3 – V4 or T_{R2} MSB
0Dh	V4 (LSB)	R	V4, V3 – V4 or T_{R2} LSB
0Eh	V_{CC} (MSB)	R	V_{CC} MSB
0Fh	V_{CC} (LSB)	R	V_{CC} LSB

Figure 6-3 Register of LTC2990

Figure 6-4 shows the control register of LTC2990. The control register must be configured properly to measure the voltage difference (V1-V2). Bits b[2:0] should be set to '110' for measuring voltage difference (V1-V2) and (V3-V4). Bits [4:3] should be set to 00 for all measurements.

BIT	NAME			OPERATION
b7	Temperature Format			Temperature Reported In; Celsius = 0 (Default), Kelvin = 1
b6	Repeat/Single			Repeated Acquisition = 0 (Default), Single Acquisition = 1
b5	Reserved			Reserved
b[4:3]	Mode [4:3]			Mode Description
	0	0		Internal Temperature Only (Default)
	0	1		T_{R1}, V1 or V1 – V2 Only per Mode [2:0]
	1	0		T_{R2}, V3 or V3 – V4 Only per Mode [2:0]
	1	1		All Measurements per Mode [2:0]
b[2:0]	Mode [2:0]			Mode Description
	0	0	0	V1, V2, T_{R2} (Default)
	0	0	1	V1 – V2, T_{R2}
	0	1	0	V1 – V2, V3, V4
	0	1	1	T_{R1}, V3, V4
	1	0	0	T_{R1}, V3 – V4
	1	0	1	T_{R1}, T_{R2}
	1	1	0	V1 – V2, V3 – V4
	1	1	1	V1, V2, V3, V4

Figure 6-4 Control register of LTC2990.

Figure 6-5 shows the status register of LTC2990. Bit b2 should be checked before reading the voltage difference registers to make sure the measurement is finished and register values is the latest for reading.

BIT	NAME	OPERATION
b7	0	Always Zero
b6	V_{CC} Ready	1 = V_{CC} Register Contains New Data, 0 = V_{CC} Register Read
b5	V4 Ready	1 = V4 Register Contains New Data, 0 = V4 Register Read
b4	V3, T_{R2}, V3 – V4 Ready	1 = V3 Register Contains New Data, 0 = V3 Register Data Old
b3	V2 Ready	1 = V2 Register Contains New Data, 0 = V2 Register Data Old
b2	V1, T_{R1}, V1 – V2 Ready	1 = V1 Register Contains New Data, 0 = V1 Register Data Old
b1	T_{INT} Ready	1 = T_{INT} Register Contains New Data, 0 = T_{INT} Register Data Old
b0	Busy*	1= Conversion In Process, 0 = Acquisition Cycle Complete

Figure 6-5 tatus register of LTC2990.

■ Design Tools

- Quartus II v15.0 64-bit
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus project directory: Demonstrations\power_monitor_nios
- Nios II Eclipse project workspace: Demonstrations\power_monitor_nios\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: Demonstrations\power_monitor_nios\demo_batch
- Batch file: test.bat
- FPGA configuration file: power_monitor_nios.sof
- Nios batch file: test.sh
- NIOS program: nios_app.elf

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Please make sure Quartus II and USB-Blaster II driver are installed on the host PC.
- Connect the USB cable from the USB-Blaster II port (J8) on the NEEK board to the host PC.
- Power on the NEEK board.
- Execute the demo batch file " test.bat " under the folder Demonstrations\power_monitor _nios\demo_batch.
- Nios II terminal will display the measured power consumption measured, as shown in **Figure 6-6**.

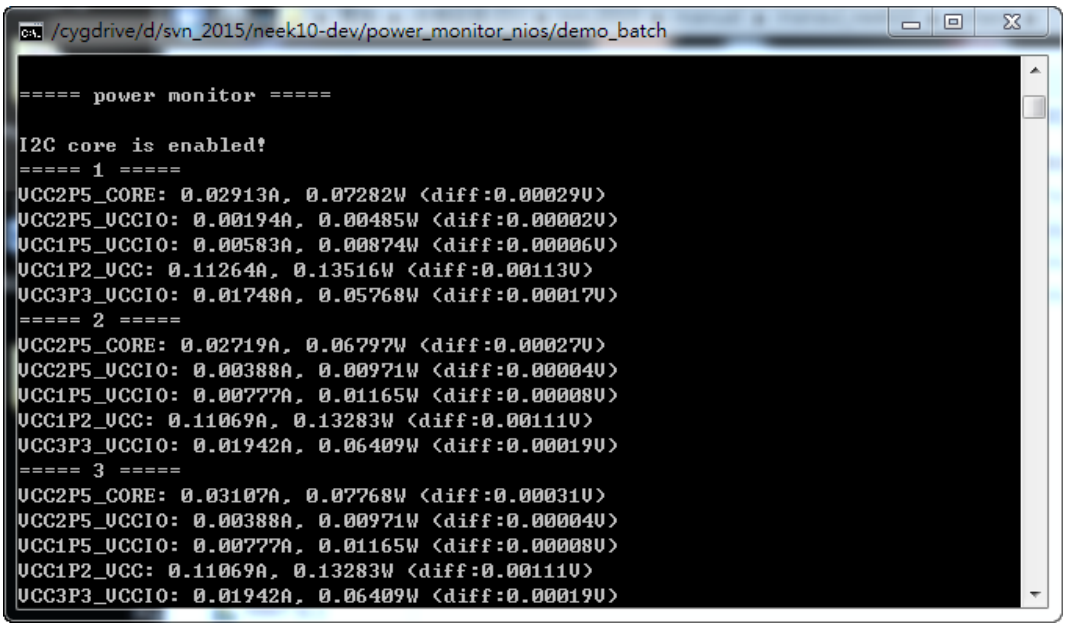

Figure 6-6 Screenshot of power_monitor_nios demo

6.2 UART to USB control LED

Many applications need communication with computer through common ports, the traditional connector is RS232 which needs to connect to a RS232 cable. However, many personal computers nowadays don't have the RS232 connector which makes it very inconvenient to develop projects. The MAX 10 NEEK board is designed to support UART communication through USB cable. The UART to USB circuit is responsible for converting the data format. Developers can use a USB cable rather than a RS232 cable to enable the communication between the FPGA and the host computer. In this demonstration we will show you how to control the LEDRs by sending a command on the computer putty terminal. The command is sent and received through a USB cable to the FPGA. Note that in FPGA, the information was received and sent through a UART IP.

Figure 6-7 shows the hardware block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The PLL generates a 100MHz clock for Nios II processor and the controller IP. The LEDRs are controlled by the PIO IP. The UART controller sends and receives command data and the command is sent through Putty terminal on the computer.

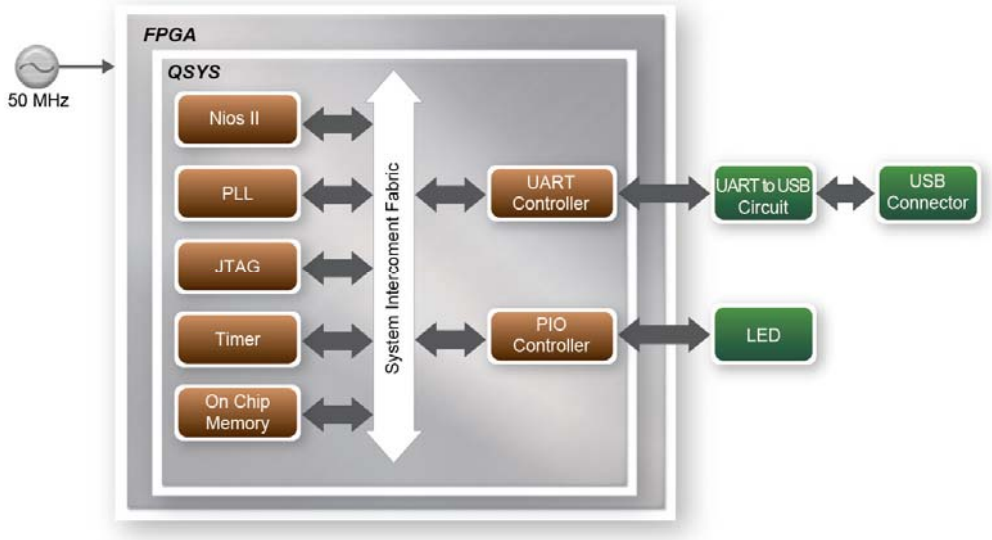

Figure 6-7 Block Diagram of UART Control LED Demo

■ Design Tools

- Quartus II 15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus Project directory: uart_usb
- Nios II Eclipse project workspace: uart_usb\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: \uart_usb\demo_batch

The demo batch file includes following files:

- Batch Files: uart_usb.bat, uart_usb.sh
- FPGA Configure File : uart_usb.sof
- Nios II Program: uart_usb.elf

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Connect a USB cable between your computer and the USB connector(J18) on MAX 10 NEEK board.
- Power on the board, if you find an unrecognized USB Serial Port in Device Manager as shown in **Figure 6-8**.you should install the UART to USB driver before you run the demonstration.

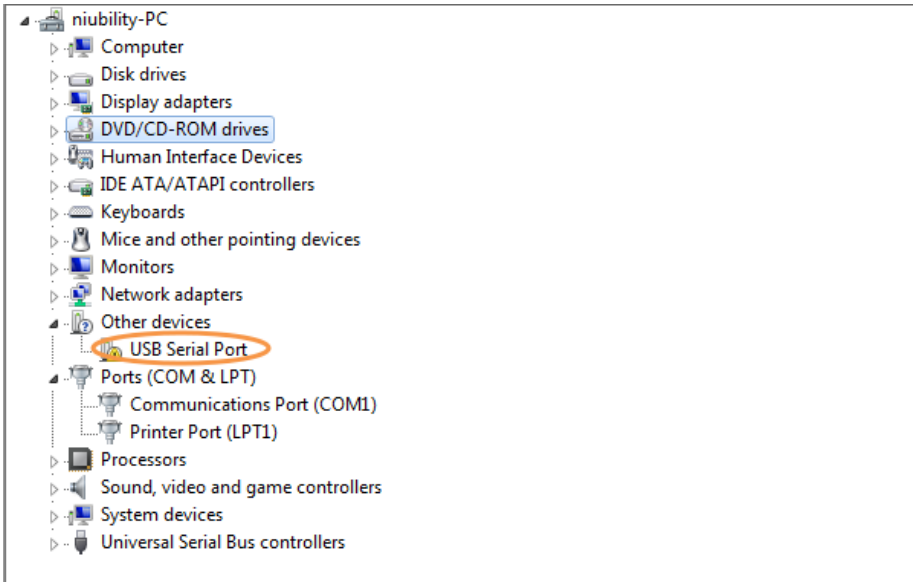

Figure 6-8 Unrecognized USB Serial Port on PC

- To install UART_TO_USB driver on your computer please select the USB Serial Port to update the driver software. The driver file can be downloaded from the following website: http://www.ftdichip.com/Drivers/VCP.htm.
- Open the Device Manager to ensure which common port is assigned to the UART to USB port as shown in **Figure 6-9**. The common number 9 (COM9) is assigned on this computer.

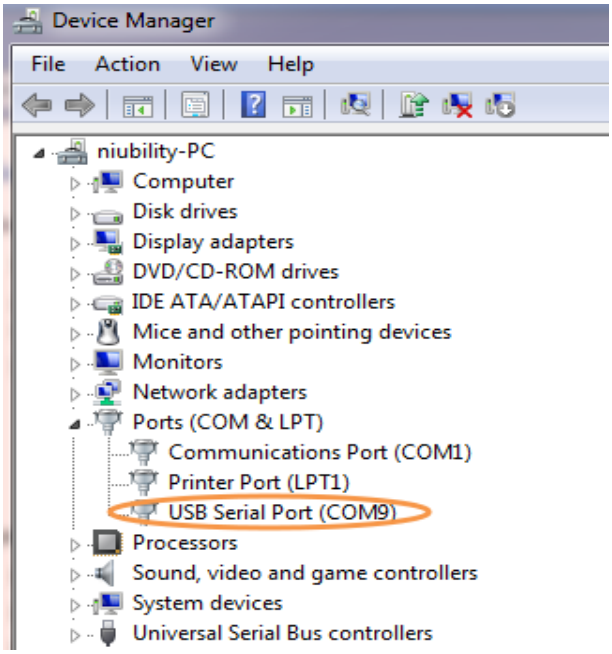

Figure 6-9 Check the Assigned Com Port Number On PC

- Open the putty software, type in the parameter as shown in **Figure 6-10** and click open button to open the terminal.(Here is a link for you to download the putty terminal: Download Putty)

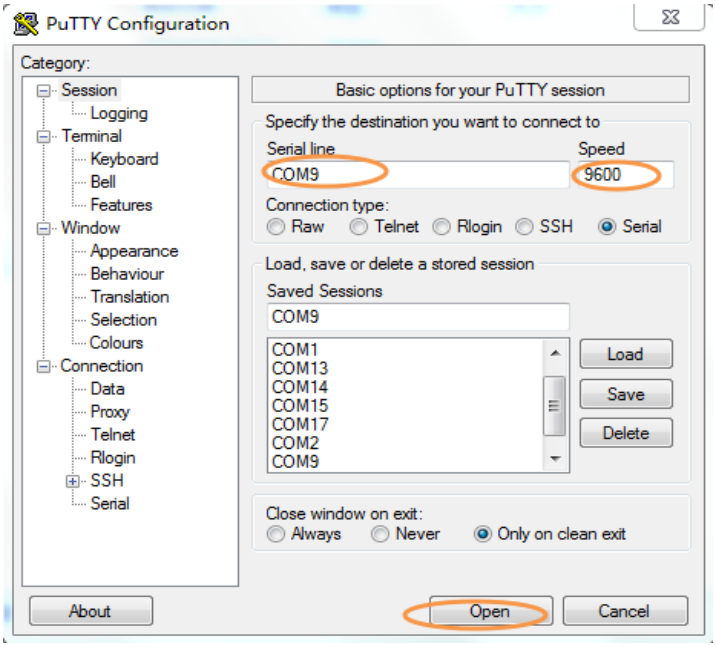

Figure 6-10 Putty Terminal Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Connect an USB cable to the MAX 10 NEEK board (J8) and install USB Blaster driver if necessary.
- Execute the demo batch file " uart_usb.bat" under the batch file folder uart_usb\demo_batch.
- The result of Nios II-terminal and putty terminal is shown in **Figure 6-11**.

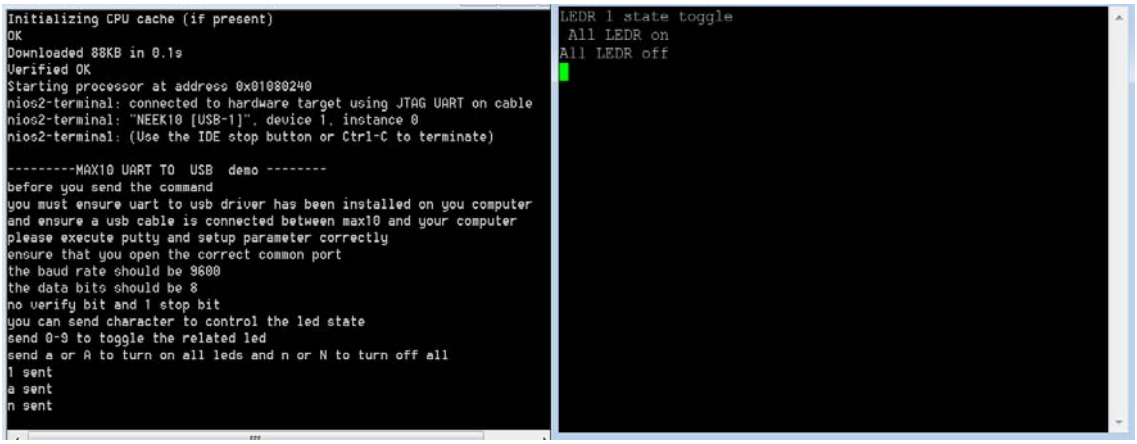

Figure 6-11 Running Result of Uart_USB Demo

- In the putty terminal, type any character to change the LED state. Type a digital number to toggle the LEDR[9..0] state and type a/A or n/N to turn on/off all LEDR.

6.3 SD Card Audio Demonstration

Many commercial media/audio players use a large external storage device, such as an SD Card or

CF card, to store music or video files. Such players may also include high-quality DAC devices such that good audio quality can be produced. The MAX 10 NEEK board provides the hardware and software needed for Micro SD Card access and professional audio performance so that it is possible to design advanced multimedia products using the MAX 10 NEEK board.

Figure 6-12 shows the diagram of this demonstration. In this demonstration we show how to implement an SD Card Music Player on the MAX10 NEEK board, in which the music files are stored in an SD Card and the board can play the music files via its high-quality and low power audio DAC circuits. We use the Nios II processor to read the music data stored in the SD Card and use theTLV320AIC3254 audio CODEC to play the music.

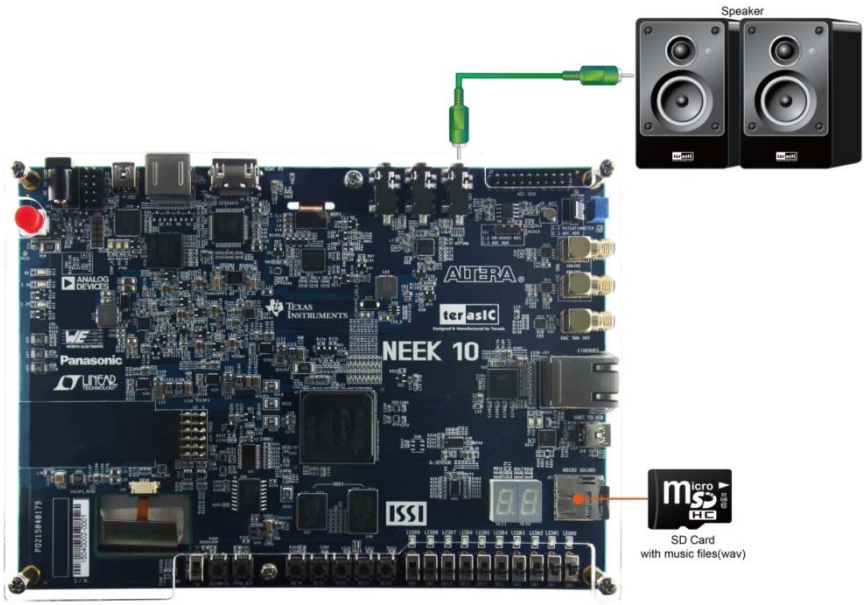

Figure 6-12 System of SD Card Audio Demo

Figure 6-13 shows the hardware block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The PLL generates a 100MHz clock for Nios II processor and the other controllers. The audio chip is controlled by the Audio Controller which is a user-defined SOPC component. The internal PLL in the CODEC chip can generate the clock rate according to the sample rate of the music. A mater clock should be supplied for the CODEC. The mater clock rate in this demonstration is 19.2M provided by the PLL block in FPGA. The audio controller requires the audio chip working in master mode, so the serial bit (BCLK) and the word clock (WCLK) are provided by the audio chip. An OpenCore I2C controller is connected to the CODEC chip for the Nios II CPU to communicate with the CODEC chip. Four PIO pins are connected to the micro SD Card socket. SD 4-Bit Mode is used to access the micro SD Card and is implemented by software. All the other SOPC components in the block diagram are SOPC Builder built-in components. The PIO pins are also connected to the keys, LEDs and switches.

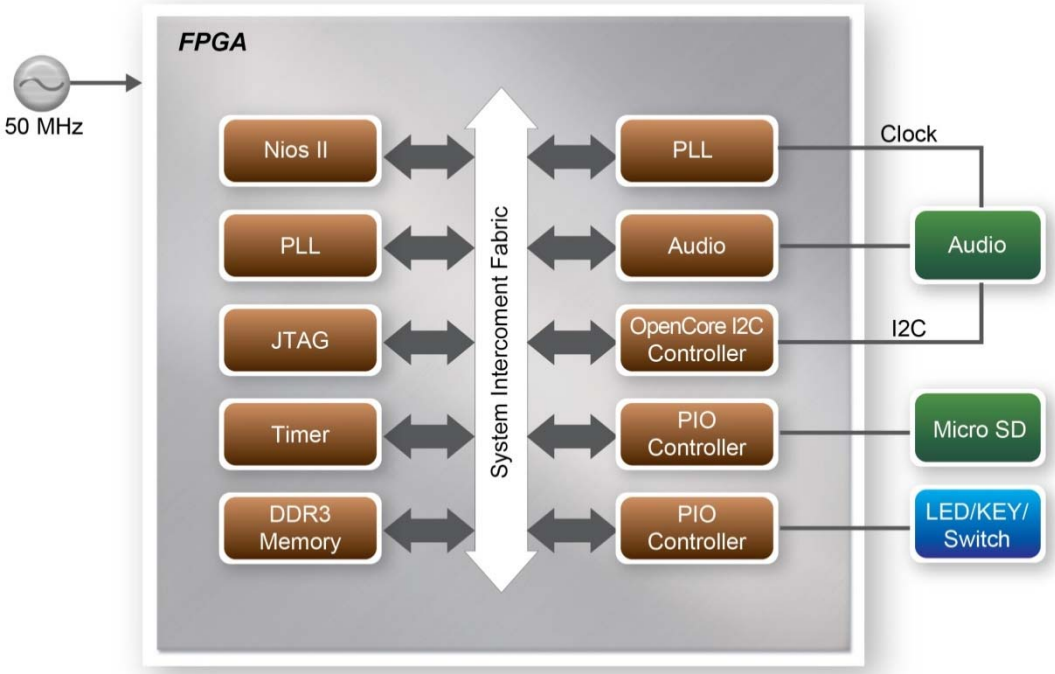

Figure 6-13 Block diagram of SD Card Audio Demo

Figure 6-14 shows the software stack of this demonstration. SD 4-Bit Mode block implements the SD 4-Bit mode protocol for reading raw data from the SD Card. The FAT block implements FAT16/FAT32 file system for reading wave files that are stored in the SD Card. In this block, only read function is implemented. The WAVE Lib block implements WAVE file decoding function for extracting audio data from wave files. The I2C block implements I2C protocol for configuring audio chip. The Audio block implements audio FIFO checking function and audio signal sending/receiving function. The key and switch block acts as a control interface of the music player system.

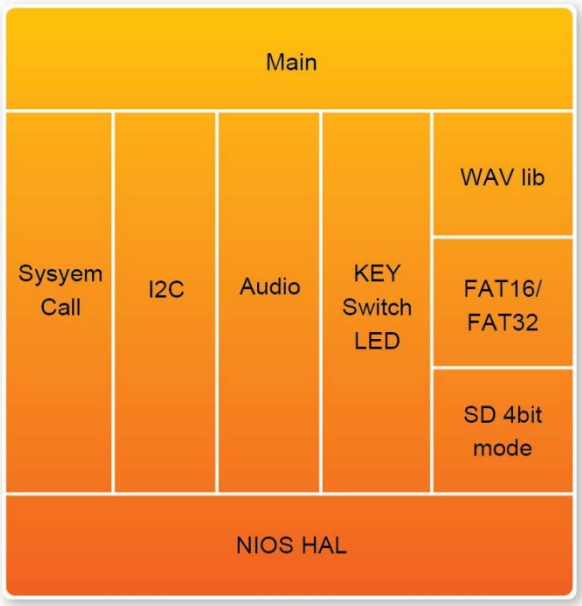

Figure 6-14 Software Stack Of SD Card Audio Demo

The audio chip should be configured before sending audio signal to the audio chip. The main program uses I2C protocol to configure the audio chip working in master mode; the audio output interface working in I2S 16-bits per channel and with sampling rate according to the wave file contents. In audio playing loop, the main program reads 512-byte audio data from the SD Card, and then writes the data to DAC FIFO in the Audio Controller. Before writing the data to the FIFO, the program will verify if the FIFO is full.

While the demonstration is running, users can get the status information through Nios II terminal. You can adjust the volume by pressing key1 or key2. And also you can choice the song by pressing key0 or key3.

■ Design Tools

- Quartus II 15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus Project directory: sdcard_audio
- Nios II Eclipse project workspace: sdcard_audio\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:

- sdcard_audio\demo_batch

The demo batch file includes following files:

- Batch Files: test.bat, test.sh
- FPGA Configure File : sdcard_audio.sof
- Nios II Program: sdcard_audio.elf

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Format your Micro SD Card into FAT16/FAT32 format
- Place the wave files to the root directory of the Micro SD Card. The provided wave files must have a sample rate of the following options: 96K, 48K, 44.1K or 8K. In addition, the wave files must be stereo and 16 bits per channel.
- Connect a headset or speaker to the MAX 10 NEEK board so you can hear the music played from the micro SD Card in later.
- Insert the micro SD card into the micro SD socket on MAX 10 NEEK board.
- Make sure Quartus II and Nios II are installed on your PC.
- Power on the MAX 10 NEEK board.
- Connect an USB cable to the MAX 10 NEEK board and install USB Blaster driver if necessary.
- Execute the demo batch file "test.bat" under the batch file folder sdcard_audio \demo_batch.
- Press KEY3 on the MAX10 NEEK board to play the last music file stored in the micro SD Card and press KEY0 to play the next song.
- Press KEY2 and KEY1 to increase and decrease the output music volume respectively as shown in **Figure 6-15.**

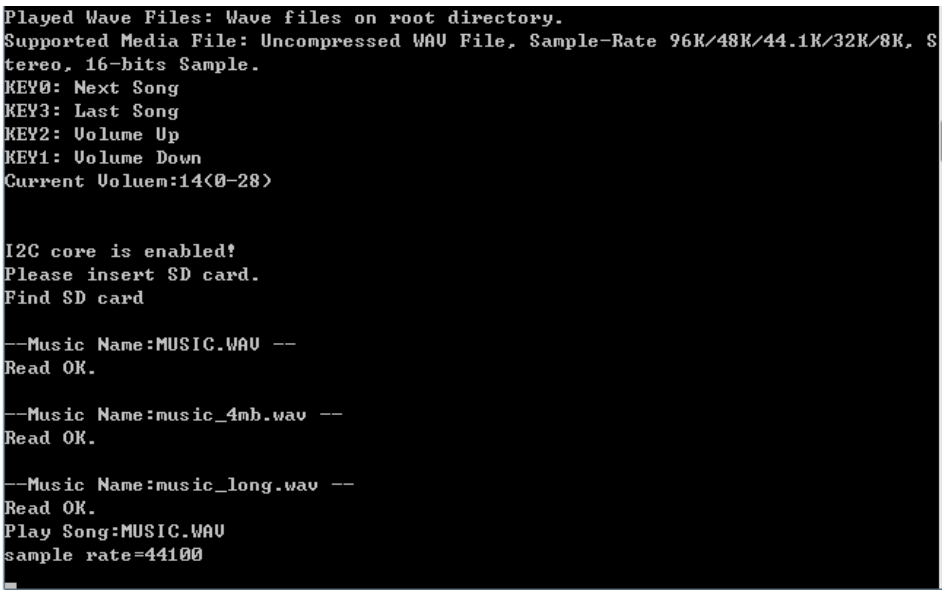

Figure 6-15 Running SD Card Audio Demo

6.4 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 6-16 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 128 MB DDR3-300 controller. The DDR3 IP generates one 300 MHz clock as SDRAM's data clock and one half-rate system clock 150 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 100MHz clock, and the Nios II program is running in the on-chip memory.

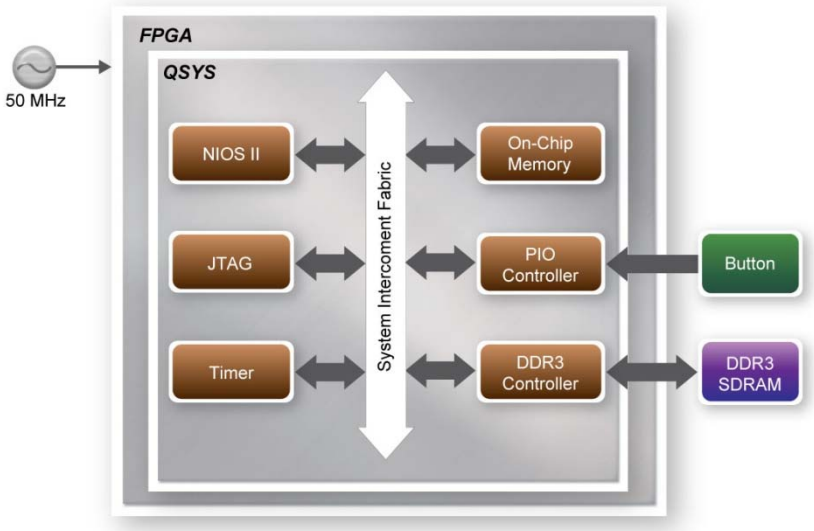

Figure 6-16 Block Diagram of the DDR3 Basic Demo

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the whole 128 MB 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, you need to perform 4 major steps:

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

■ Design Tools

- Quartus II 15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus Project directory: ddr3_nios
- Nios II Eclipse Project workspace: *ddr3_nios/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: *ddr3_nios\demo_batch*

The demo batch folder includes following files:

- Batch Files: ddr3_nios.bat, ddr3_nios.sh
- FPGA Configure File : ddr3_nios.*sof*
- Nios II Program: ddr3_nios.*elf*

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Make sure Quartus II and Nios II are installed on your PC.
- Power on the MAX 10 NEEK board.
- Use an USB cable to connect PC and the MAX 10 NEEK board (J8) and install USB Blaster driver if necessary.
- Execute the demo batch file " ddr3_nios.bat" for USB-Blaster II under the batch file folder, ddr3_nios\demo_batch
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in Nios2-terminal.
- Press KEY4~KEY0 of the MAX 10 NEEK board to start SDRAM verify process. Press KEY0 for continued test.
- The program will display progress and result information, as shown in **Figure 6-17**.

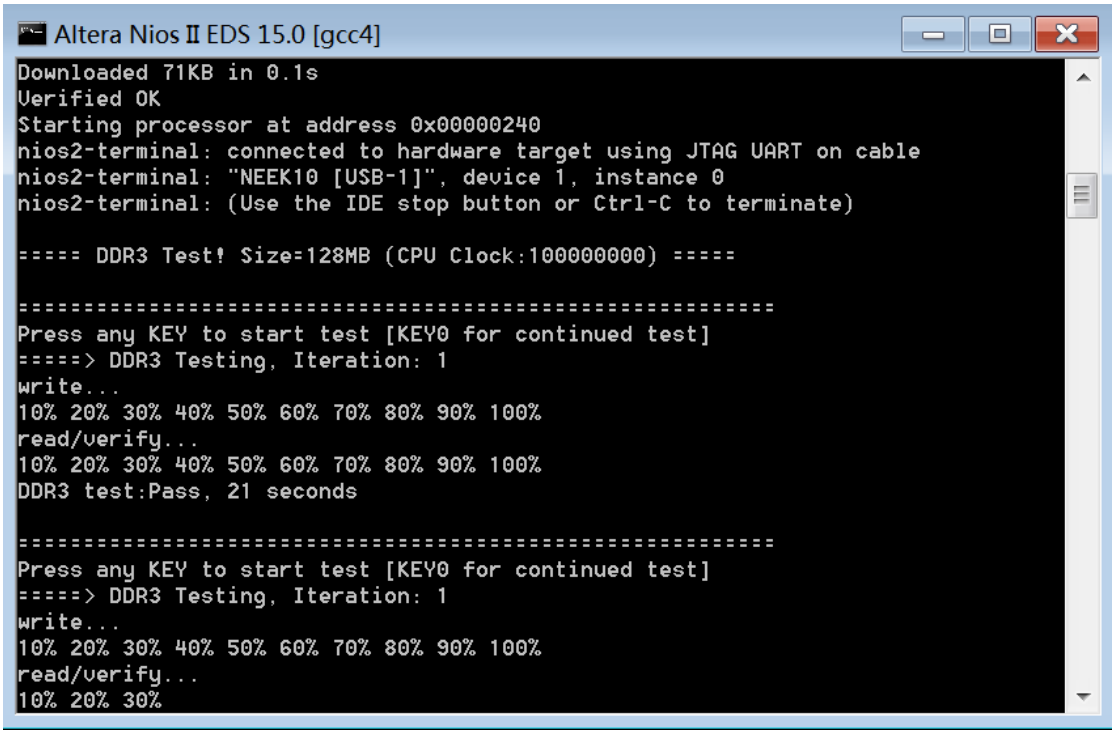

Figure 6-17 Display Progress and Result Information for the DDR3 Demo

6.5 Ethernet Socket server

This design example demonstrates a socket server using the sockets interface of the NicheStack™ TCP/IP Stack Nios II Edition with MicroC/OS-II to serve socket connection to the MAX 10 NEEK board. The server can continuously listen for commands on a TCP/IP port and operate the MAX 10 NEEK LEDs according to the commands from the telnet client.

As Part of the Nios II EDS, NicheStack™ TCP/IP Network Stack is a complete networking software suite designed to provide an optimal solution for network related applications accompanying Nios II.

Also to understand how this demo works, we assume that you already have a basic knowledge of TCP/IP protocols. As indicated in the block diagram in **Figure 6-18**, the Nios II processor is used to communicate with the Client via 88E1111(RGMII/MII interface)Ethernet Device.

Figure 6-18 Block Diagram

We will now cover the Qsys system in this demo which contains Nios II processor, DDR3 memory, JTAG UART, timer, Triple-Speed Ethernet, Scatter-Gather DMA controller and other peripherals etc. In the Core Configuration Tab of the Altera Triple-Speed Ethernet Controller, users need to set the MAC interface as RGMII as shown in **Figure 6-19**.

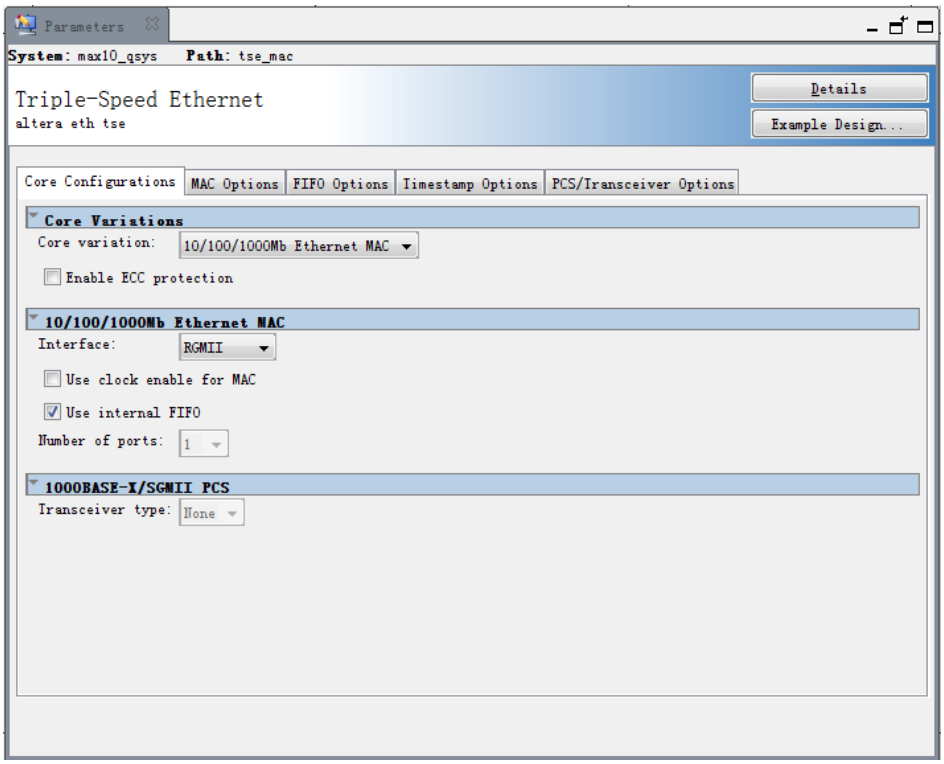

Figure 6-19 Select RGMII Interface under MAC Configuration

In the MAC Options tab (See **Figure 6-20**), users should set up proper values for the PHY chip 88E1111. The MDIO Module should be included, as it is used to generate a 2.5MHz MDC clock for the PHY chip from the controller's source clock(here a 100MHz clock source is expected) to divide the MAC control register interface clock to produce the MDC clock output on the MDIO interface. The MAC control register interface clock frequency is 100MHz and the desired MDC clock frequency is 2.5MHz, so a host clock divisor of 40 should be used.

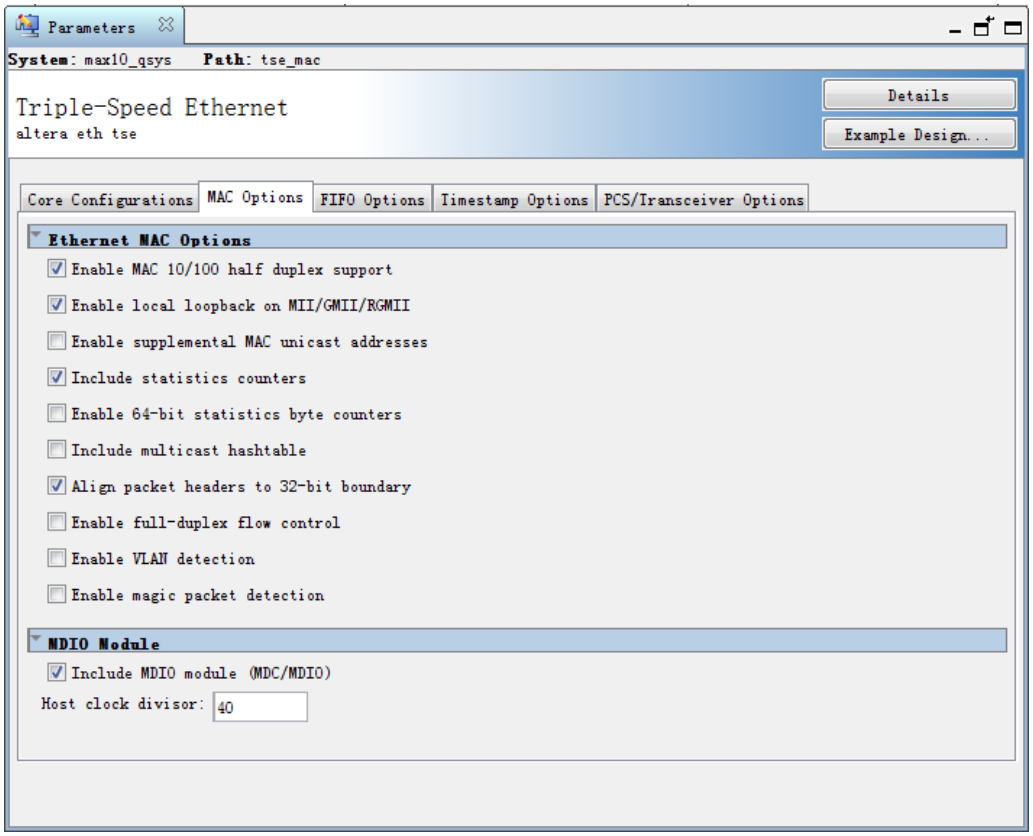

Figure 6-20 MAC Options Configuration

Once the Triple-Speed Ethernet IP configuration has been set and necessary hardware connections have been made as shown in **Figure 6-21**, click on generate.

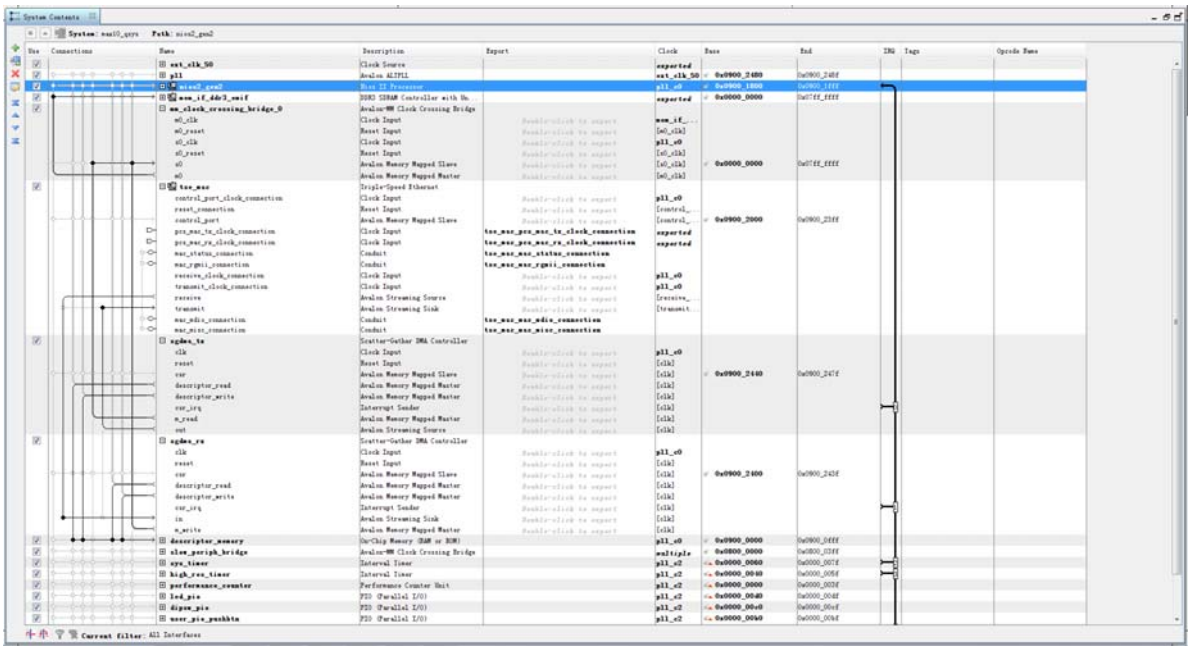

Figure 6-21 Qsys Builder

Figure 6-22 shows the connections for programmable 10/100/1000Mbps Ethernet operation via RGMII.

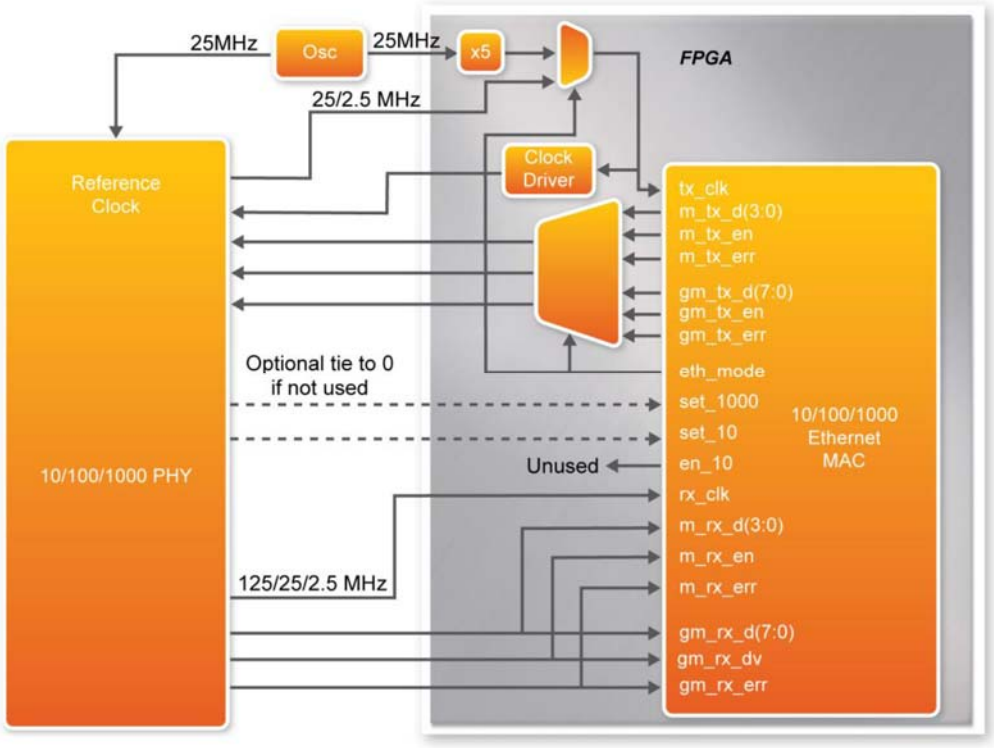

Figure 6-22 PHY connected to the MAC via RGMII

After the Qsys hardware project has been built, develop the Qsys software project, whose basic architecture is shown in **Figure 6-23**. The top block contains the Nios II processor and the necessary hardware to be implemented into the MAX 10 NEEK board. The software device drivers contain the necessary device drivers needed for the Ethernet and other hardware components to work. The HAL API block provides the interface for the software device drivers, while the Micro C/OS-II provides communication services to the NicheStack™ and the Socket Server. The NicheStack™ TCP/IP Stack software block provides networking services to the application block where it contains the tasks for Socket Server and also LED management.

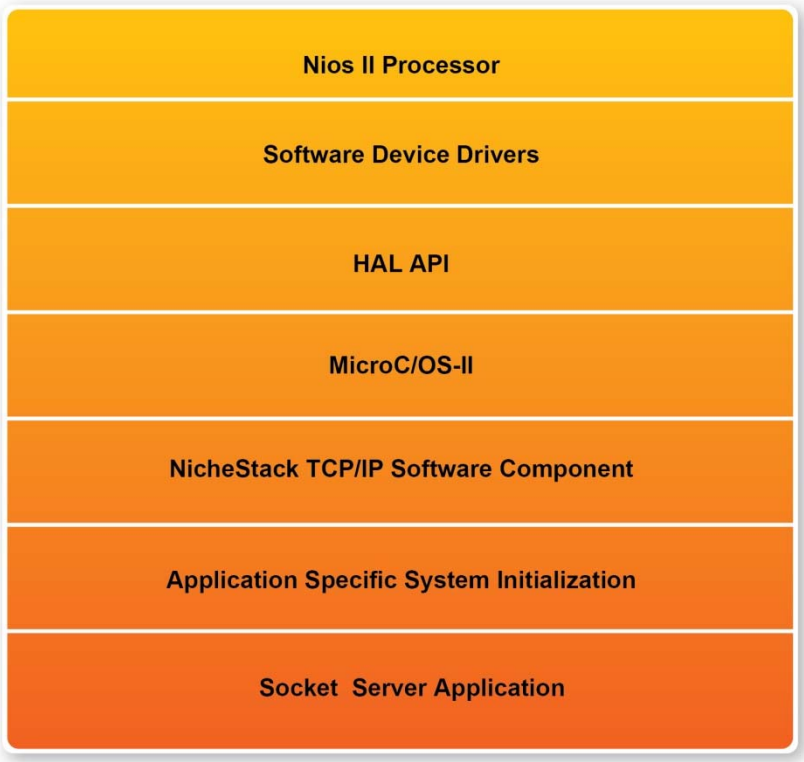

Figure 6-23 Nios II Software Routine Architecture

Finally, the detailed descriptions for Software flow chart of the Socket Server program are listed below:

Firstly, the Socket Server program initiates the MAC and net device then calls the get_mac_addr() function to set the MAC addresses for the PHY. Secondly, it initiates the auto-negotiation process to check the link between the PHY and gateway device. If the link exists, the PHY and gateway devices will broadcast their transmission parameters, speed, and duplex mode. After the auto-negotiation process has been finished, the link will be established. Next, the Socket Server program will prepare the transmitting and receiving path for the link. If the path is created successfully, it will call the get_ip_addr() function to set up the IP address for the network interface. After the IP address is successfully distributed, the NicheStack™ TCP/IP Stack will start to run for Socket Server application.

Note: your gateway should support DHCP because it uses DHCP protocol to request a valid IP from the Gateway, or else you would need to reconfigure the system library to use static IP assignment.

■ Design Tools

- Quartus II v15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Project directory: socket_server
- Nios II Eclipse Project workspace: socket_server/software

■ Demonstration Batch File

- Demo batch file folder: Demonstrations/socket_server /demo_batch/
- Batch file: socket_server.bat
- FPGA configure file: socket_server.sof
- Application file folder: socket_server /demo_batch/
- Application file: open_telnet.bat

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Please make sure both Quartus II and USB-Blaster II driver are installed on the host PC.
- Connect the USB cable from the USB-Blaster II port (J8) on the MAX 10 NEEK board to the host PC.
- Power on the MAX 10 NEEK board.
- Execute the demo batch file "socket_server.bat" under the folder Demonstrations/socket_server /demo_batch , then the IP address and port number are assigned as shown below in **Figure 6-24**.

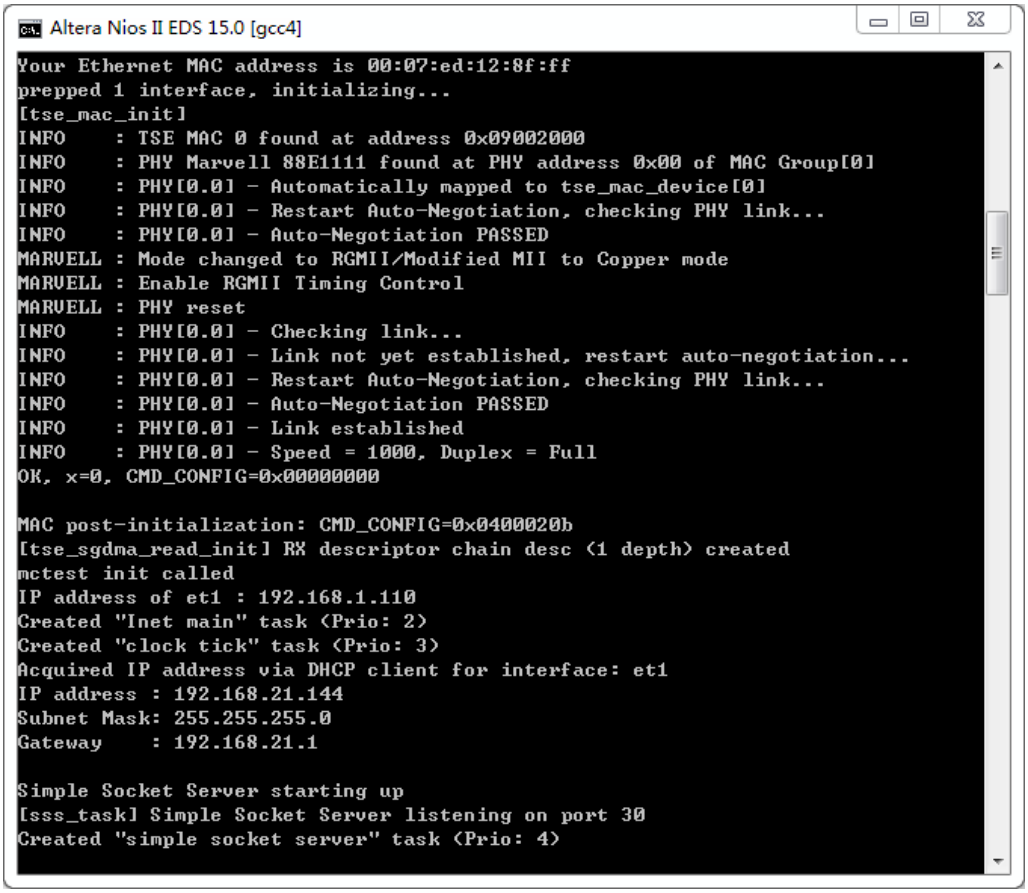

Figure 6-24 Simple Socket Server

- To establish connection, start the telnet client session by executing open_telnet.bat file and include the IP address assigned by the DHCP server-provided IP along with the port number as shown below in **Figure 6-25**.

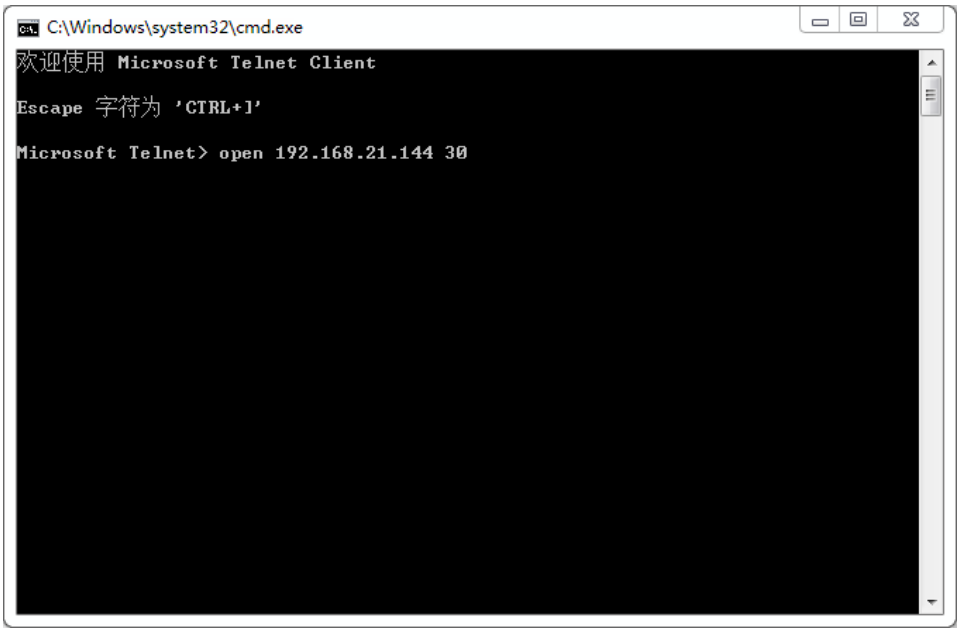

Figure 6-25 Telnet Client

- From the Simple Socket Server Menu, enter the commands in the telnet session. Entering a number from zero through seven, followed by a return, causes the corresponding LEDRs (D0-D7) to toggle on or off on the MAX 10 NEEK board as shown below in **Figure 6-26**.

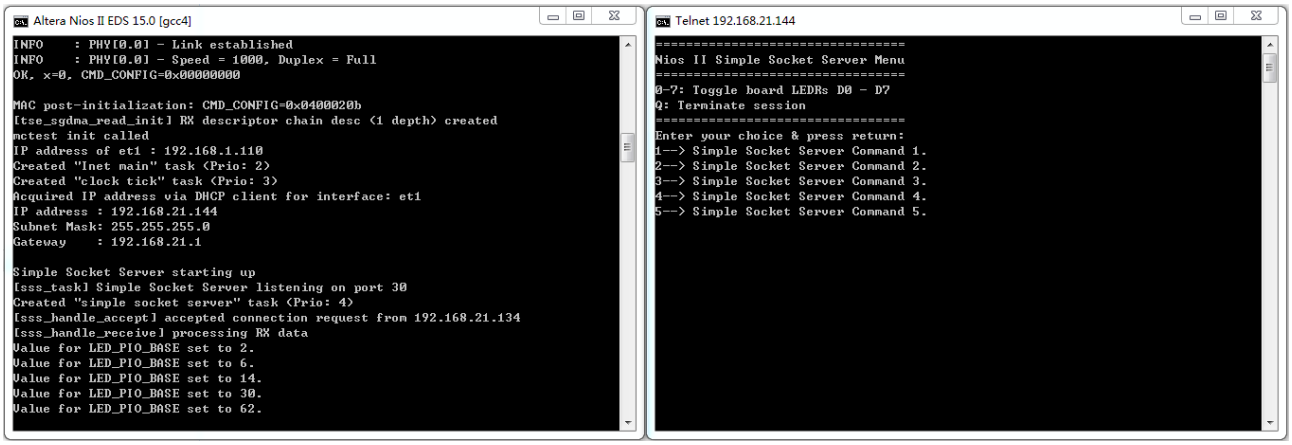

Figure 6-26 Display Progress and Result Information for the Socket Server Demonstration

6.6 LCD Painter

This demonstration shows how to implement a painter demo on the Multi-touch LCD module based on Altera Qsys tool and the Video and Image Processing (VIP) suite. It demonstrates how to use multi-touch gestures and resolution. The GUI of this demonstration is controlled by the program in Nios II.

■ Operation Description

Figure 6-27 shows the Graphical User Interface (GUI) of Painter demo. The GUI is divided into 4 separate areas: Painting Area, Gesture Indicator, Clear Button, and Color Palette. Users can select a color from the color palette and start painting in the paint area. If a gesture is detected, the associated gesture symbol will be shown in the gesture area. To clear the painting area, click the "Clear" button.

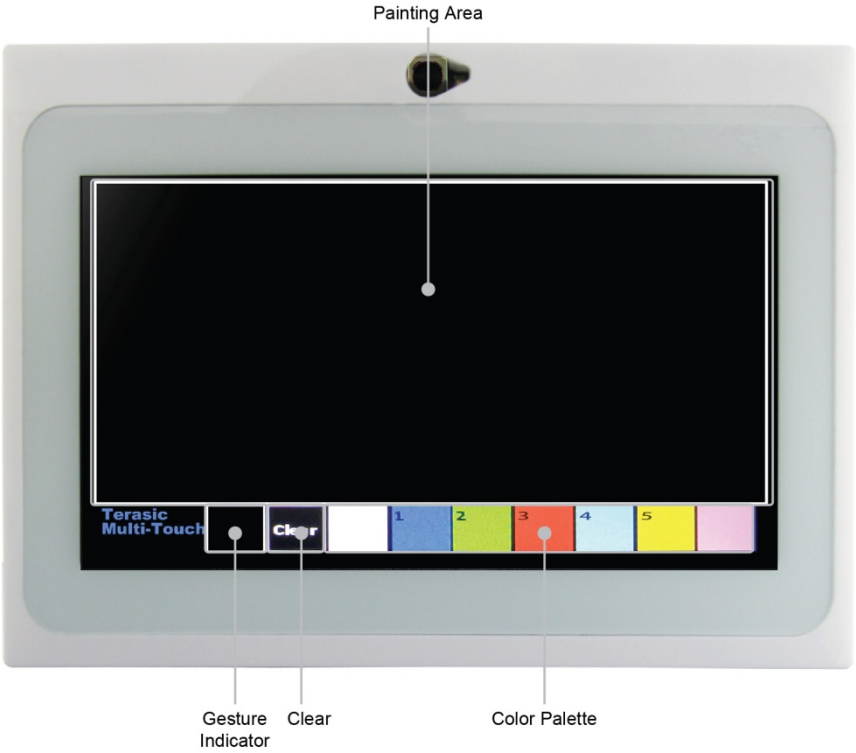

Figure 6-27 GUI of Painter Demo

Figure 6-28 shows the single-finger painting of canvas area.

Figure 6-28 Single-Finger Painting

Figure 6-29 shows the zoom-in gesture.

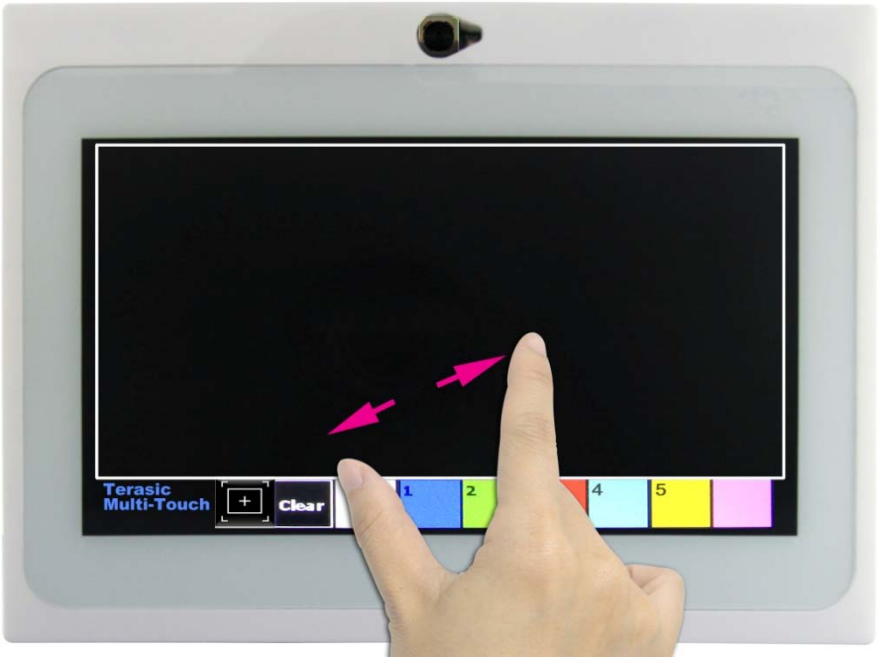

Figure 6-29 Zoom-In Gesture

Figure 6-30 shows the 5-Point painting of canvas area.

Figure 6-30 5-Point Painting

■ System Description

For LCD display processing, the reference design is developed based on Altera's Video and Image Processing (VIP) suite. The Frame Reader VIP is used for reading data to be displayed from the associated video memory, and the VIP Video Out is used to display the video data. The data is drawn by the Nios II processor according to user input.

For multi-touch processing, whenever there is any touch activity occurring, a I2C Controller IP is used to retrieve serial data from the I2C interface, the associated touch information including multi-touch gestures and 5-point touch coordinates can be calculated through the data in NIOS II.

Figure 6-31 shows the system generic block diagram of painter demonstration.

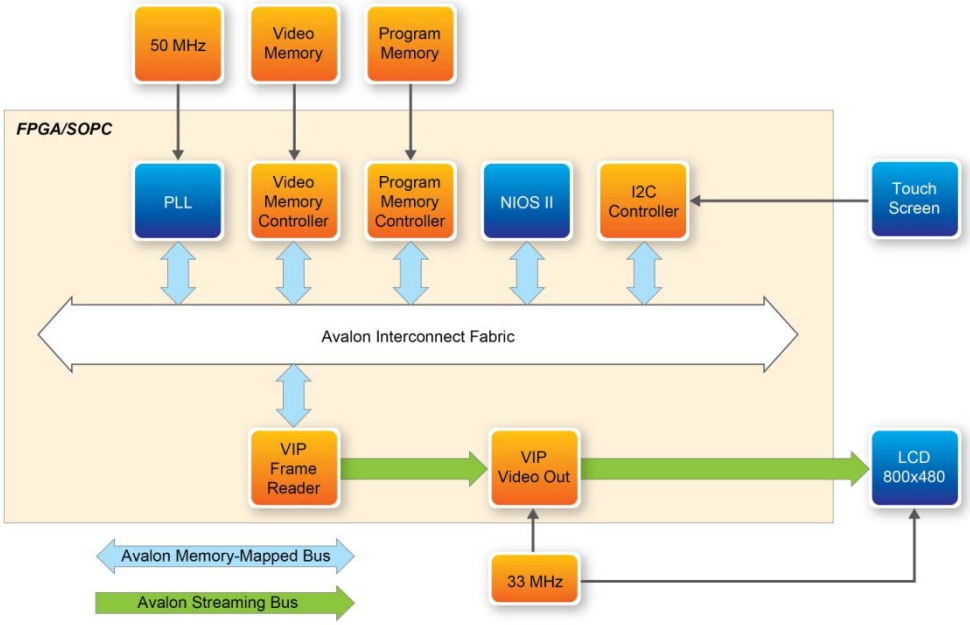

Figure 6-31 System Block Diagram of the Painter Demonstration

■ Design Tools

- Quartus II v15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Project directory: lcd_painter
- Nios II Eclipse Project workspace: lcd_painter/software

■ Demonstration Batch File

- Demo Batch File Folder: \lcd_painter\demo_batch

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Connect a USB cable between your computer and the USB connector(J8) on MAX 10 NEEK board.
- Power on the MAX 10 NEEK Board.
- Please make sure Quartus II has been installed on the host PC.
- Execute the demo batch file "lcd_painter.bat " under the folder lcd_painter\demo_batch.
- The painter GUI will show up on the LCD panel.

6.7 Digital Accelerometer Demonstration

This demonstration shows a bubble level implementation based on a digital accelerometer. We use I^2C protocol to control the ADXL345 digital accelerometer, and the APDS-9301 Miniature Ambient Light Photo Sensor. The LCD displays the interface of our demo. When tilting the MAX 10 NEEK, the ADXL345 measures the static acceleration of gravity. In our Nios II software, we compute the change of angle in the x-axis and y-axis, and show the angle data on the LCD display. The value of light sensor will change as the brightness changes around the light sensor.

Figure 6-32 shows the hardware system block diagram of this demonstration. The system is clocked by an external 50MHz Oscillator. Through the internal PLL module, the generated 100MHz clock is used for Nios II processor and other components, and there is also 40MHz for low-speed peripherals.

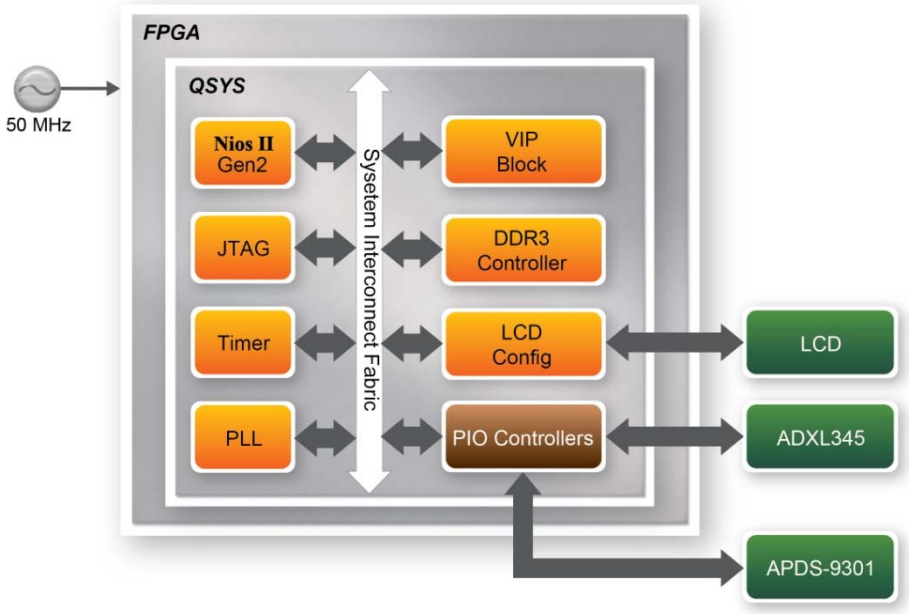

Figure 6-32 Block diagram of the digital accelerometer demonstration

■ Demonstration Source Code

- Project directory: gsensor_lightsensor_lcd
- Bit stream used: gsensor_lightsensor_lcd.sof
- Nios II Eclipse project workspace: gsensor_lightsensor_lcd \software

■ Demonstration Batch File

- Demo Batch File Folder: gsensor_lightsensor_lcd\demo_batch

The demo batch file includes the following files:

- Batch File: gsensor_lightsensor_lcd.bat, gsensor_lightsensor_lcd.sh
- FPGA Configure File: gsensor_lightsensor_lcd.sof
- Nios II Program: gsensor_lightsensor_lcd.elf

■ Demonstration Setup

Please follow the procedures below to set up the demonstration:

- Load the bit stream into the FPGA on the MAX 10 NEEK.
- Run the Nios II Software under the workspace gsensor_lightsensor_lcd\software (Note*).
- After the Nios II program is downloaded and executed successfully, a prompt message will be

displayed in Nios2-terminal: “its ADXL345’s ID = e5”.

- Tilt the MAX 10 NEEK to all directions, and you will find that the angle of the g-sensor and value of light sensor will change. **Figure 6-33** shows the demonstration result.

Figure 6-33 Digital Accelerometer demonstration

Note:

Execute gsensor_lightsensor_lcd \demo_batch\gsensor_lightsensor_lcd.bat to download .sof and .elf files.

6.8 Humidity/Temperature Sensor

This demonstration illustrates steps to evaluate the performance of humidity and temperature sensor HDC1000. The HDC1000 is a fully integrated humidity and temperature sensor, providing excellent measurement accuracy and long term stability. Humidity and temperature results can be read out through the I2C compatible interface. The LCD displays the interface of our game. In our Nios II software, we show temperature and humidity data on the LCD display. The value of the sensor will change as the environment changes.

Figure 6-34 shows the block diagram of this demonstration. In this demonstration, a Nios II

processor is used to achieve i2c operation and display results on the Nios II console and LCD.

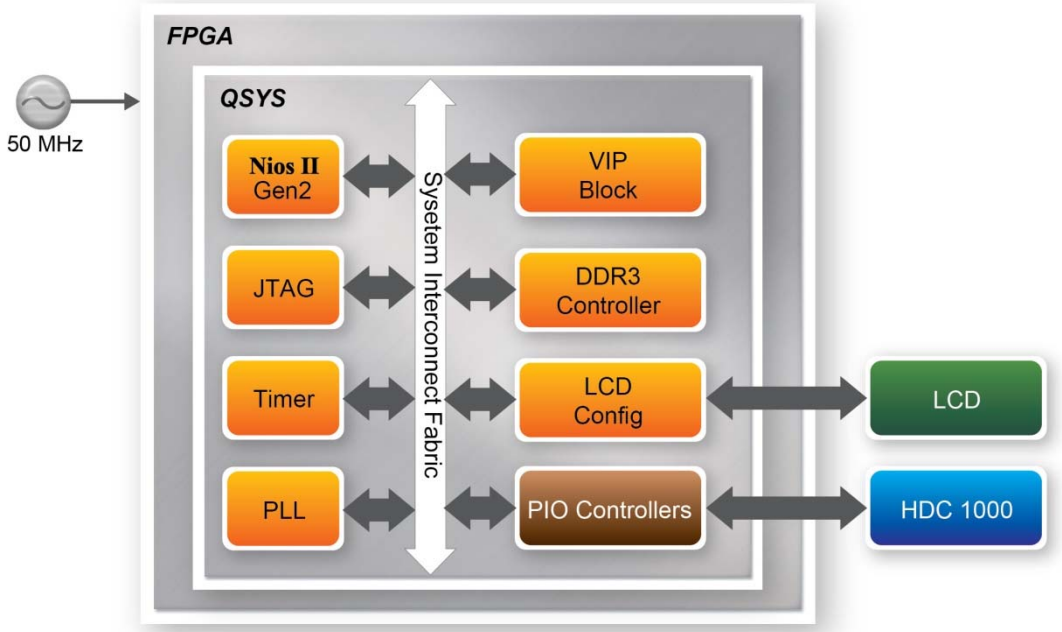

Figure 6-34 Block diagram of Humidity and Temperature Sensor

This demonstration shows basic function of HDC1000 and temperature and humidity reading in different acquisition modes. HDC1000 can perform measurements of both humidity and temperature, or either humidity only or temperature only. The measurement resolution can be set to 8, 11, or 14 bits for humidity; 11 or 14 bits for temperature. Different resolution setting results in a different conversion time. When triggering the measurements, operation should wait for the measurements to complete based on the conversion time. Alternatively, wait for the assertion of DRDYn. In this demonstration, a delay in I2C function is adopted to simplify the process.

■ Design Tools

- Quartus II v15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus project directory: humidity_temperature_lcd
- Nios II Eclipse project workspace: humidity_temperature_lcd \software

■ Demonstration File Locations

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

■ Demonstration Setup and Instructions

Please follow the procedures below to set up the demonstration:

- Make sure Quartus II and USB-Blaster II driver are installed on your PC.
- Connect the USB cable to the USB Blaster II connector (J8) on the MAX 10 NEEK board and host PC.
- Power on the MAX 10 NEEK board.
- Execute the demo batch file "humidity_temperature_lcd.bat" under the batch file folder, humidity_temperature_lcd \demo_batch.
- NIOS terminal and LCD will display the humidity and temperature values. **Figure 6-35** shows the demonstration result.

Figure 6-35 Humidity and Temperature Sensor Demo

6.9 LCD CAMERA Demonstration

This demonstration shows how to implement a camera demo on the Multi-touch LCD module in Altera Qsys tool. Altera VIP (Video Image Processing) suite is used to display image on the LCD panel and a Nios II processor is used to configure the I2C devices. There is a Camera IP from Terasic in Qsys, which translates the Bayer pattern from camera to the RGB video steam format, and feeds it to Altera VIP. The other IP developed by Terasic for auto-focus is used to find the optimized focus settings of user-defined image area.

■ System Block Diagram

Figure 6-36 shows the system block diagram of camera demonstration on the LCD panel.

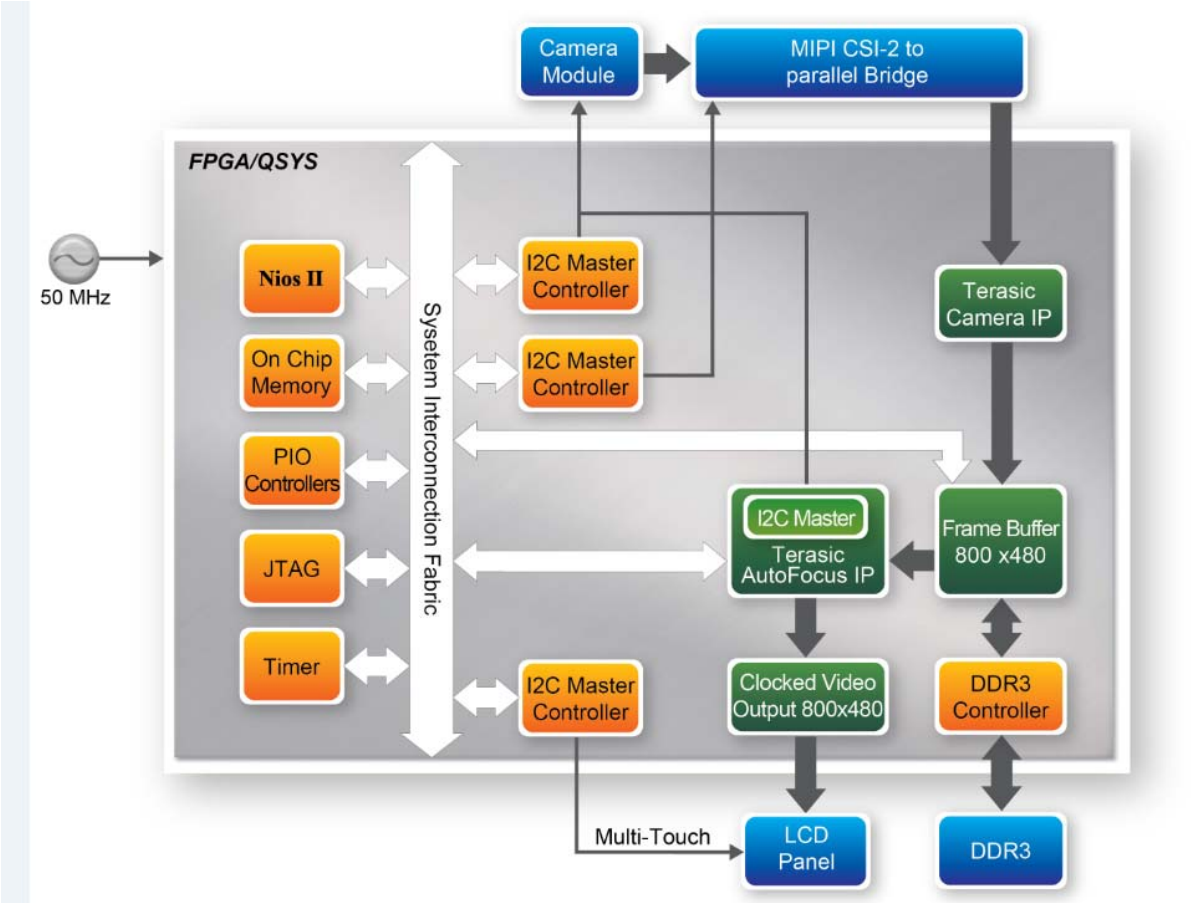

Figure 6-36 System block diagram of camera demonstration

The camera module used includes an image sensor form OminiVision (OV8865) and a VCM (Voice Coil Motor) driver IC (VCM149C). OV8865 is a low-power high-performance 8-megapixel image sensor. The VCM driver IC is used to control the focus of the camera. The camera is configured to output a 4 lanes MIPI CSI-2 protocol in this demonstration.

The settings by default are:

- Resolution : 800x480 (LCD resolution)
- Frame Rate: 60 fps
- Pixel Data: RAW10
- Bin Mode: 1, 2, 4 (achieved ZOOM-IN/ ZOOM-OUT function)

Users can change the settings base on their requirements.

A MIPI CSI-2 bridge chip TC358748XBG is used to decode the video data via MIPI interface from the camera module to the FPGA in parallel. The camera module is configured as RAW10 in this demonstration, so the data width of 24-bit parallel bus between MIPI CSI-2 bridge and FPGA is only 10-bit used.

For image process in FPGA, the reference design is developed based on Altera's Video and Image Processing (VIP) suite. The Terasic Camera IP translates the parallel Bayer pattern data into RGB data to meet the specification of Altera VIP video streaming. The Frame Buffer from VIP is used for buffering image data in DDR3 and matching the frame rate from Terasic camera IP to the Clock Video Output of VIP. It displays the final 800x480 RGB frame image on the LCD panel. The auto-focus IP by Terasic can be used to get a better image quality by finding the optimized focus setting.

The Nios II program running on on-chip memory controls three I2C controllers to configure the image sensor, motor driver, MIPI CSI-2 Bridge IC, and touch device. First I2C controller is used to configure the camera module, including OV8865 image sensor and VCM149C. The second I2C controller is used to configure the MIPI Bridge IC TC358748XBG. The third I2C controller is used to retrieve the touch information from the touch device.

For better image quality with user-defined area focused, a simple auto-focus algorithm is used and implemented as a Qsys IP. The range of motor position, which controls the focus, is defined as 0 ~ 1023. The algorithm searches for the best focus position in two stages. It takes images when the focus position is set to 0, 6, 12 .. etc. in the first stage and calculates the contrast of these images to narrow down the search by finding the most clear image of all at focus position X. It then starts to take images again when the focus position is set to X-3, X-2, X-1, X, X+1, X+2, and X+3 in the second stage to come up with the best focus position after two iterations. The Nios II processor is used to configure and trigger the IP. Users can develop and implement a more efficient algorithm based on the auto-focus IP provided in this demonstration.

Note: The focus driver IC (VCM149C) in the camera module is configured by the Terasic auto-focus IP through its own I2C master controller. The image sensor in the camera module is also configured through the same I2C bus by its own I2C master controller. The arbitration is implemented in the Nios II program to prevent the I2C bus occupied by the two I2C master controllers simultaneously. Users must make sure there is only one I2C master used at the same one

time.

For ZOOM function, the ZOOM-IN and ZOOM-OUT functions are implemented by configuring the bin mode of the image sensor in this demonstration.

For multi-touch processing, the Nios II polling the associated touch information to process the touch event when touch activity occurs.

■ Function Description

Figure 6-37 shows the result of this demonstration.

Figure 6-37 Camera demo running on MAX 10 NEEK board

A Five-fingers touch function is implemented in this demonstration to stop LCD from refreshing the camera image. This is achieved by stopping the input of Frame Buffer VIP and the image retrieved by the Clock Video Output VIP from the Frame Buffer VIP is always the same. The camera image can start refreshing again by a single-finger touch to enable the input of Frame Buffer VIP.

The zoom-in and zoom-out gestures with two fingers are implemented to control the display active readout window on the image sensor.

Auto-focus function is triggered by single-finger touch.

Figure 6-38 shows the Five-fingers touch to stop the camera video.

Figure 6-38 Five-fingers touch to stop the camera video

Figure 6-39 shows the zoom-in or zoom-out gesture with two fingers.

Figure 6-39 Two-fingers gesture to control zoom-in or zoom-out

Figure 6-40 shows the autofocus function with single finger touch.

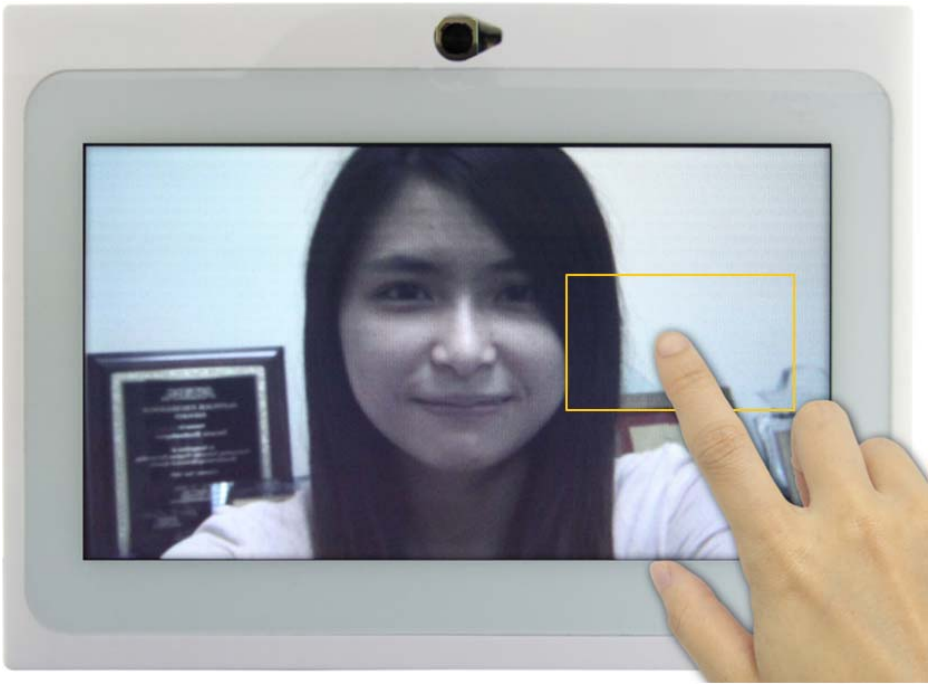

Figure 6-40 Single-finger touch to trigger the auto-focus function

■ Design Tools

- Quartus II v15.0
- Nios II Eclipse 15.0

■ Demonstration Source Code

- Quartus project directory: lcd_camera
- Nios II Eclipse project workspace: lcd_camera\software

■ Demonstration Batch File

- Demo batch file folder: demonstrations\lcd_camera\demo_batch

■ Demonstration Setup

Please follow the procedures below to setup the demonstration:

- Connect a USB cable between the host PC and the USB connector (J8) on the MAX10 NEEK board.
- Power on the MAX10 NEEK board.
- Please make sure Quartus II has been installed on the host PC.
- Execute the demo batch file " lcd_camera.bat" under the batch file folder lcd_camera\demo_batch.
- The LCD panel will start showing the video captured from the camera.
- Use two-point zoom-in or zoom-out gesture to perform zoom fucntion.
- User single touch to specify the focus area.
- User five-point touch to stop the camera video streaming and any touch to resume the video streaming.

Chapter 7

Application Selector

The application selector utility is the default code that powers on the FPGA and offers a graphical interface on the LCD, allowing users to select and run different demonstrations that reside on one micro SD card.

7.1 Ready to Run SD Card Demos

You can find several ready-to-run demos in your micro SD card root directory as well as in the System CD under **Factory_Recovery\SD_content** folder. Figure 7-1 shows the photograph of the application selector main interface.

Figure 7-1 Application Selector Main Interface

Also, you can easily convert your own applications to be loadable by the application selector. For more information see "**Creating Your Own Loadable Applications**" in section 7.4. If you have lost the contained files in the SD card, you could find them on the MAX 10 NEEK System CD under the **Factory_Recovery** folder.

7.2 Application Selector Details

This section describes some details about the design of the application selector utility.

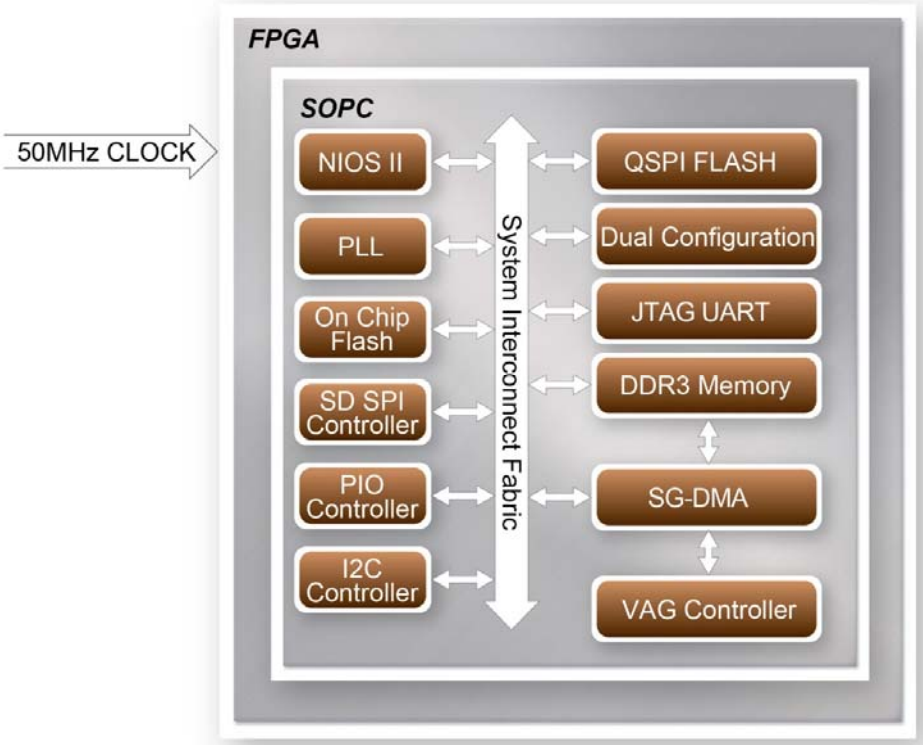

Figure 7-2 Block Diagram Of Application Selector

Figure 7-2 shows the hardware block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The PLL generates a 100MHz clock for Nios II processor and the other controllers. The DDR3 works as the system memory. The Nios II processor writes the display content to the DDR3. The SG-DMA reads the data and translates it to the VGA controller through the Avalon Streaming interface. The I2C controller and the PIO Controller are implemented to get the touch action data of the MTL2 touch screen.

The Nios II processor accesses the SD card through an SD card SPI controller. The Application Selector uses an SD card for storing applications hardware and software binary files. The SD card must be formatted with the FAT16/32 file system. Long file names are supported.

The Dual Configuration IP provides an Avalon-MM interface for NIOS II processor to access the remote system upgrade circuitry in the MAX10 FPGA device. It's the critical part of this demonstration. The Altera Dual Configuration IP core offers the following capabilities through

Avalon-MM interface:

- Asserts RU_nCONFIG to trigger reconfiguration.
- Asserts RU_nRSTIMER to reset watchdog timer if the watchdog timer is enabled.
- Writes configuration setting to the input register of the remote system upgrade circuitry.
- Reads information from the remote system upgrade circuitry.

The onchip flash is divided into three parts for this demonstration. CMF0 is used to store image0 which is the hardware of the application selector code, and CMF1 is used to store image1 which is the hardware of the application being selected. The UFM zone is not used in this project and user can use it as a non-volatile flash if necessary. The software code for the image0 is stored in the QSPI flash memory offset 0x3c00000 address. In addition, the software code for the image1 is stored in the QSPI flash memory offset 0x00. The Nios II processor loads the software from the QSPI flash after powering-up, so the reset vector of the CPU is set to the QSPI flash offset 0x3c00000 address.

7.3 Running the Application Selector

- Connect power to the MAX 10 NEEK
- Insert the micro SD card with applications into the micro SD Card socket of MAX 10 NEEK
- Make sure the CONFIG_SEL switch is set to 0 and Switch on the power (SW18) (1*)
- Scroll to select the demonstration to load using the side-bar
- Tap on the Load button to load and run a demonstration (2*)

Note:

(1)If the board is already powered, the application selector will boot from QPSI flash, and a splash screen will appear while the application selector searches for applications on the SD Card.
(2)The application will begin loading, and a window will be displayed showing the progress. Loading will take about 1 minute according to size of the binary files.

7.4 Creating Your Own Loadable Applications

It is available to convert your own design into an application which is loadable by the Application Selector utility. All you need is a program object image (a .POF file) which contains the hardware and a software image which runs on that hardware (a Nios II .ELF file).

The critical steps are:

- The hardware design must contain a dual configuration IP in the Qsys design. (**1***)
- IF the sof contains a Nios II processor. The hardware design must contain an Altera QSPI Flash controller as shown in **Figure 7-3**.

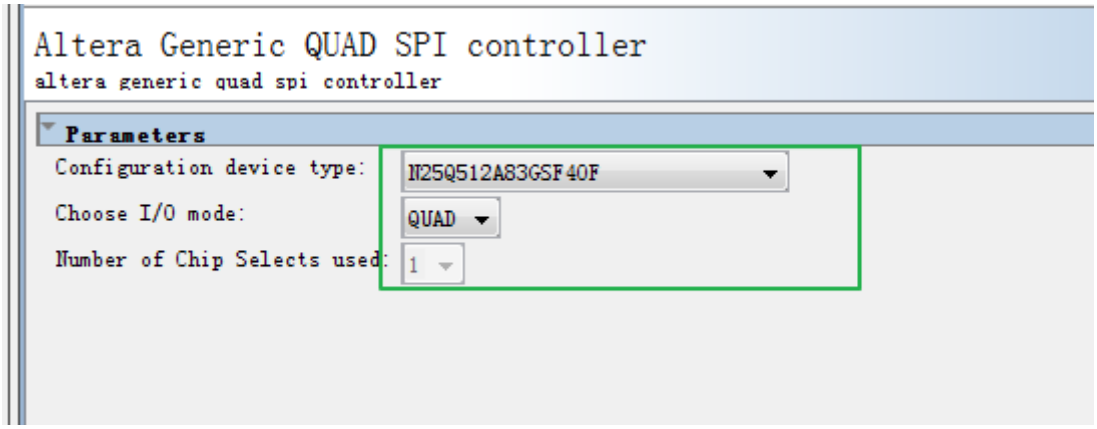

Figure 7-3 Altera QSPI Flash Controller Setting In Qsys

- The avl_mem interface of the Altera QSPI Flash controller must be connected to the data master and instruction master interface of the Nios II CPU. Set the CPU reset vector and exception vector to the QSPI flash zone with offset 0 as shown in **Figure 7-4**.

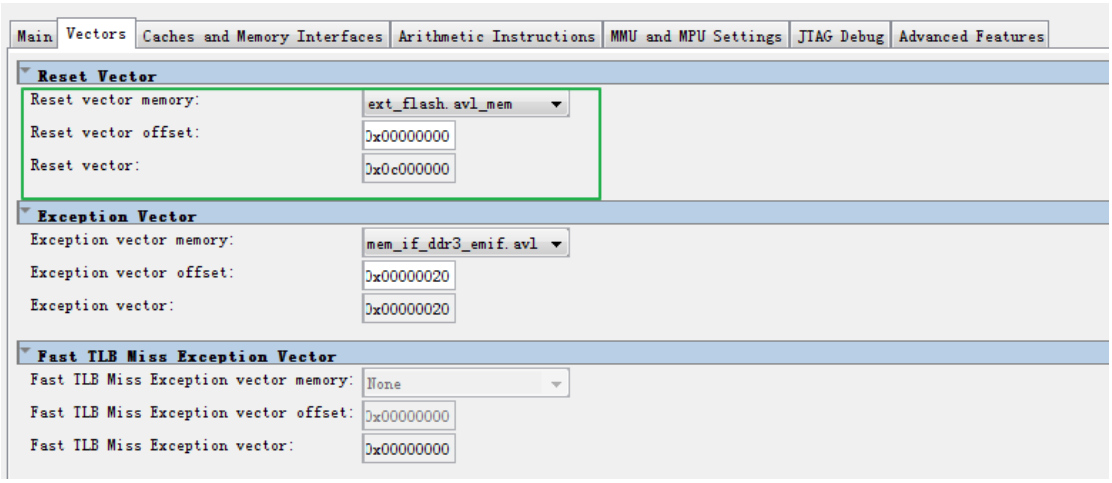

Figure 7-4 NiosII CPU Vectors Setting

- The configuration mode in Quartus device and options setting window should be set to dual compressed image mode as shown in **Figure 7-5**.

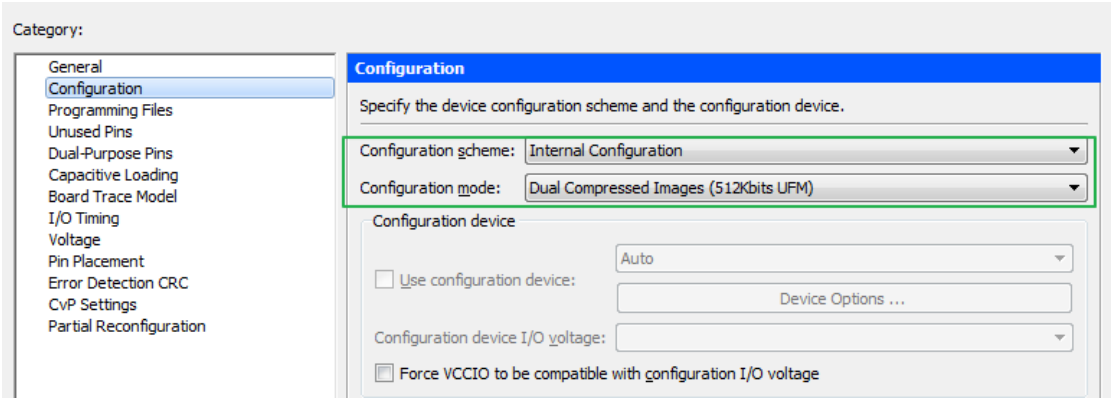

Figure 7-5 Setting Configuration Mode

- Create your software project in Nios II Eclipse and generate the .elf file by building the

software.

- On your host PC, launch a Nios II Command Shell from Start ->Progrms ->Altera ->Nios II<version#>EDS ->Nios II Command Shell
- From the command shell navigate to where your ELF file is located and create your software flash file using the following command.
- elf2flash --base=qspi_flash_base_addrss --end=qspi_flash_end_address --reset= qspi_flash_ base_addrss --input="<your software name>.elf" --output="<you software name>.flash" --boot="$SOPC_KIT_NIOS2/ components/altera_nios2/boot_loader_cfi.srec"
- Convert the software flash file to binary file using the following command.
- Nios2-elf-objcopy –I srec –O binary "<your software name>.flash" "<your software name>.bin".(**2***)
- In the Input files to convert table, choose the application_selector.sof for the Page_0 sof data. Then add a sof item Page_**1** sof data and choose the sof file of your design. Enable the Create config data RPD checkbox as shown in **Figure 7-6**.Click the generate to convert the file.(**3***)

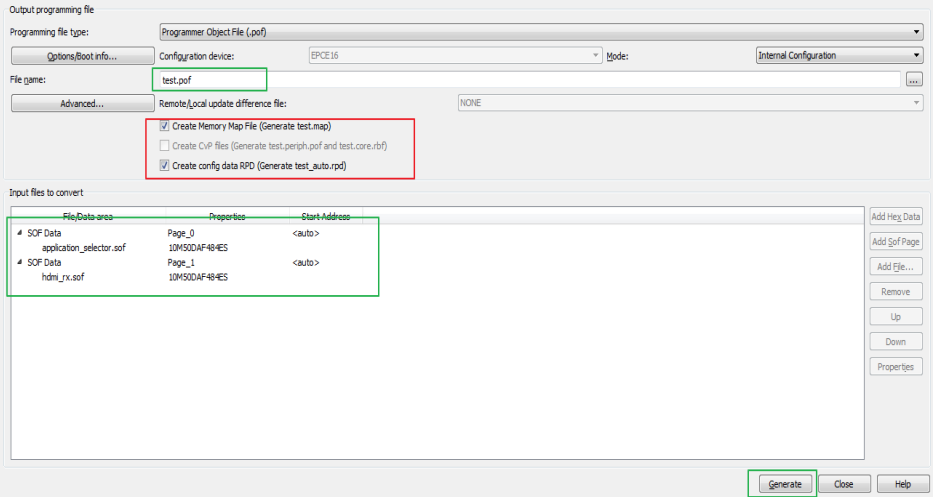

Figure 7-6 Setting Convert Programming File

- In the project directory, you will find the generated file test.pof and test_auto.rpd. The rpd file is the raw program data file which is been written into the onchip flash. Copy the application_hw.exe in tool directory in application project path to your own project directory. Double click the executable file then a new file(test_hw.rpd) will be generated.
- Using an SD card reader, copy the target files <application _name>_sw.bin and <application _name>_hw.rpd into the root directory of the SD card.**(4*)**
- Place the SD card in the MAX 10 NEEK board, and switch on the power. The Application Selector will start up, and you will now see your application appear as one of the selections

Note:

(1) You can't use an initialized memory in the design for dual compressed image configuration mode, or you will meet an error in the Quartus Fitter process.

(2) The users can convert the .elf to .bin with the batch file in elftobin_batch directory. Copy the elf file to the directory and rename the file to test.elf. Make sure the address for the QSPI flash is right and click test.bat to convert the file automatically.

(3) The sof file of your design must be add to Page_1 sof data, please double check it.

(4) If the sof file don't contain a Nios II processor, it is not necessary to generate the "<your software name>.bin" and copy it to your SD card.

7.5 Restoring the Factory Image

This section describes some details about the operation of restoring the Application Selector factory image.

■ Combining factory recovery binary files

In the factory settings, you need to program Application Selector software code to the QSPI flash and the hardware binary to the onchip flash. You should covert the hardware and software into separated .pof for programming onchip Flash and the QSPI flash.

- Copy the elf and the boot_loader_cfi.srec file into a common directory relying on your choice. This directory is where you will convert the elf into a hex file.
- On your host PC, launch a Nios II Command Shell from Start ->Programs -> Altera -> Nios II <version #> EDS -> Nios II Command Shell
- From the command shell navigate to where your elf file is located and create a .flash file using the following commands listed below
- elf2flash --base=0x07c00000 --end=0x08000000 --reset=0x07c00000 --input=application_seletcor.elf --output=application_selector.flash --boot=boot_loader_cfi.srec
- Convert the .flash into the .hex file
- nios2-elf-objcopy –O ihex application_selector.flash application_selector.hex
- In Quartus convert programming file tool window, set the options as **Figure 7-7.**

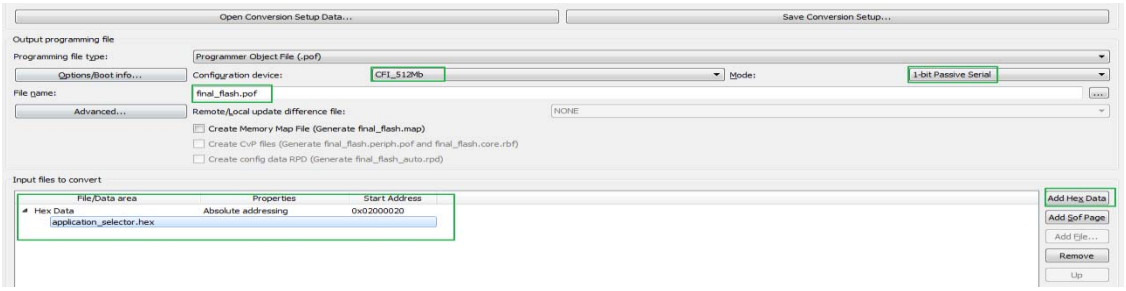

Figure 7-7 Generate The Final_flash Pof

- Click the **Add hex Data** button and choose the application_selector.hex file generated above then press generate button to generate the object file final_flash.pof.
- To convert the sof file into pof, just follow the steps for dual boot programming and make sure the sof for application selector is added to sof Page_0 data. Assuming the name of the pof is app_sel.pof.

Note:

(1) The original command for convert the elf into flash is:

- ***elf2flash --input=elf_filename --output=flash_file_name --base=flash_base_address --end =flash_end_address --reset=flash_base_address --boot="$SOPC_KIT_NIOS2/components/ altera_nios2/boot_loader_cfi.srec"***

(2) You can also convert the elf file into the hex data by executing test.bat in the software_batch directory in the project directory.

■ Restoring the original binary file

- To restore the original contents of the Application Selector, perform the following steps:
- Copy Selector project into a local directory of your choice. The Selector project is placed in Demonstrations\application_selector
- Power on the MAX 10 NEEK board, with the USB cable connected to the UBII(J8) port
- Download the max10_qpfl.sof to the board
- Press Auto Detect button and a QSPI flash will be detected as shown in **Figure 7-8.**

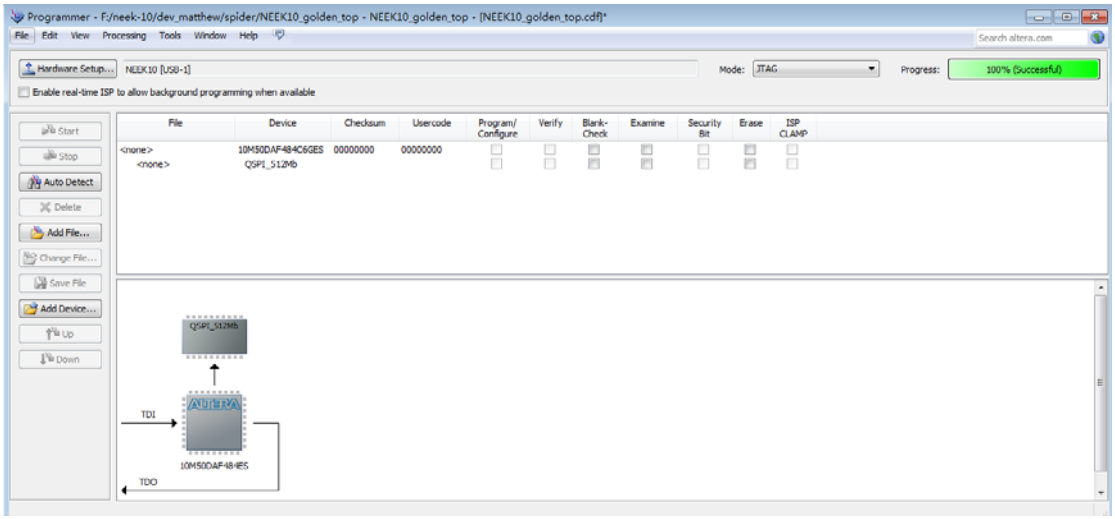

Figure 7-8 Auto Detect The QSPI Flash 1

- Right click on the QSPI_512Mb item and change the file to the final_flash.pof generated above. Check the Program/Configure button and start programming. The programming will take a long time and the software is downloaded to the QSPI flash if successful.
- Add the app_sel.pof and program it into the onchip Flash.
- Make sure the CONFIG_SEL is set to 0 and re-power up the board.

Note:

You can also use 'demo_batch' to restore the original binary file by executing the test.bat under the application_selector\demo_batch folder.

Chapter 8

Programming the Configuration Flash Memory

This tutorial provides comprehensive information that will help you understand how to configure MAX 10 NEEK Board using internal configuration mode. The MAX10 device on MAX 10 NEEK Board supports dual image boot. This tutorial explains the details of the dual image boot. The following sections provide a quick overview of the design flow.

8.1 Internal Configuration

The internal configuration scheme for all MAX 10 devices except for 10M02 device consists of the following mode:

- Dual Compressed Images—configuration image is stored as image0 and image1 in the configuration flash memory(CFM).
- Single Compressed Image.
- Single Compressed Image with Memory Initialization.
- Single Uncompressed Image.
- Single Uncompressed Image with Memory Initialization.

In dual compressed images mode, you can use the BOOT_SEL pin to select the configuration image.

The High-Level Overview of Internal Configuration for MAX 10 Devices as shown in **Figure 8-1.**

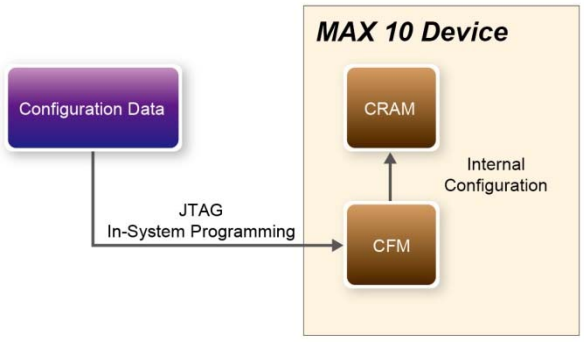

Figure 8-1 High-Level Overview of Internal Configuration for MAX 10 Devices

Before internal configuration, we need to program the configuration data into the configuration flash memory (CFM). The CFM will be part of the programmer object file (.pof) programmed into the internal flash through the JTAG In-System Programming (ISP).
During internal configuration, MAX 10 devices load the configuration RAM (CRAM) with configuration data from the CFM. Both of the application configuration images, image 0 and image 1, are stored in the CFM. The MAX 10 device loads either one of the application configuration image from the CFM. If an error occurs, the device will automatically load the other application configuration image. Remote System Upgrade Flow for MAX 10 Devices is shown in **Figure 8-**.

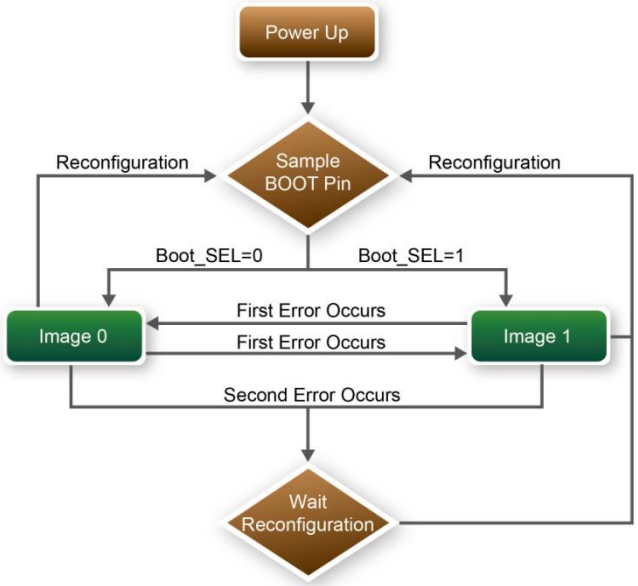

Figure 8-2 Remote System Upgrade Flow for MAX 10 Devices

The operation of the remote system upgrade feature detecting errors is as follows:
1. After powering-up, the device samples the BOOT_SEL pin to determine which application configuration image to boot. The BOOT_SEL pin setting can be overwritten by the input register of the remote system upgrade circuitry for the subsequent reconfiguration.
2. If an error occurs, the remote system upgrade feature reverts by loading the other application configuration image. The following lists the errors that will cause the remote system upgrade feature to load another application configuration image:

- Internal CRC error
- User watchdog timer time-out

3. Once the revert configuration completes and the device is in the user mode, you can use the remote system upgrade circuitry to query the cause of error and which application image failed.
4. If a second error occurs, the device waits for a reconfiguration source. If the auto-reconfig is enabled, the device will reconfigure without waiting for any reconfiguration source.
5. Reconfiguration is triggered by the following actions:

- Driving the nSTATUS low externally
- Asserting internal or external nCONFIG low
- Asserting RU_nCONFIG low

8.2 Using Dual Compressed Images

The internal configuration scheme for all MAX 10 devices except for 10M02 device consists of the following mode:

- Dual Compressed Images—configuration image is stored as image 0 and image 1 in the configuration flash memory(CFM).
- Single Compressed Image.

This section will just introduce how to use MAX10 device Dual Compressed Images feature. If you don't need this feature, skip this section

Two main steps are necessary for a project support dual configuration mode.

- Add dual configuration IP.
- Modify Configuration Mode in device setting.

A **Dual Configuration IP** should be added in an original project, so that the .pof file can be programmed into CFM through it.

- Open Quartus project and choose **Tools** > **Qsys** to open **Qsys** system wizard. Create a new system and add the dual configuration IP. Connect the logic as **Figure 8-3.**

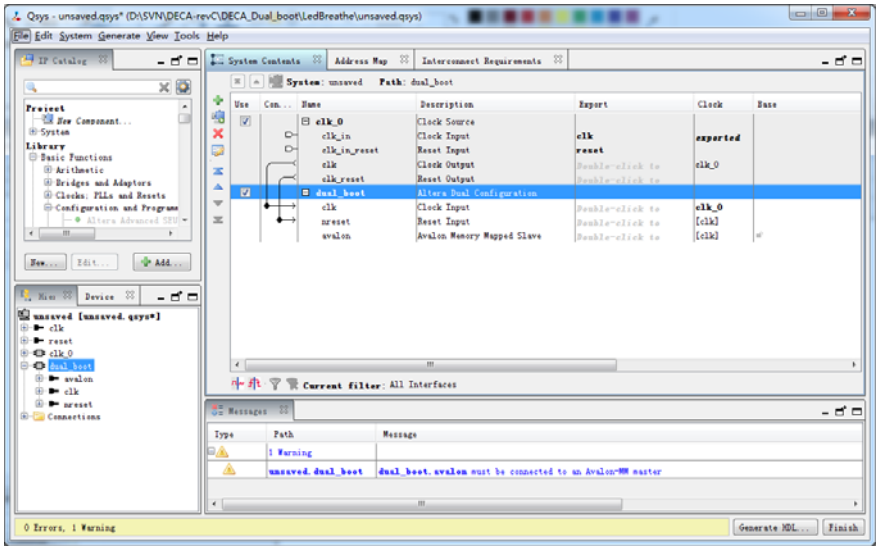

Figure 8-3 Rename and Connect Dual Boot IP

- Save the Qsys as dual_boot.qsys and generate the HDL files. Add the dual_boot.qip into the quartus setting file and add the qsys instance in the top design file as shown in **Figure 8-4**.

```
//=======================================================
//  Structural coding
//=======================================================


dual_boot u0 (
    .clk_clk       (MAX10_CLK1_50),       //   clk.clk
    .reset_reset_n (1'b1)  // reset.reset_n
);
```

Figure 8-4 Add the Qsys instance in Top

- The configuration mode is set in Device and Pin Options window. Choose **Assignments** > **Device** to open Device windows. Click the **Device and Pin Options** and choose Dual Compressed Images in configuration table as shown in **Figure 8-5**. Then compile the project to generate the sof file.

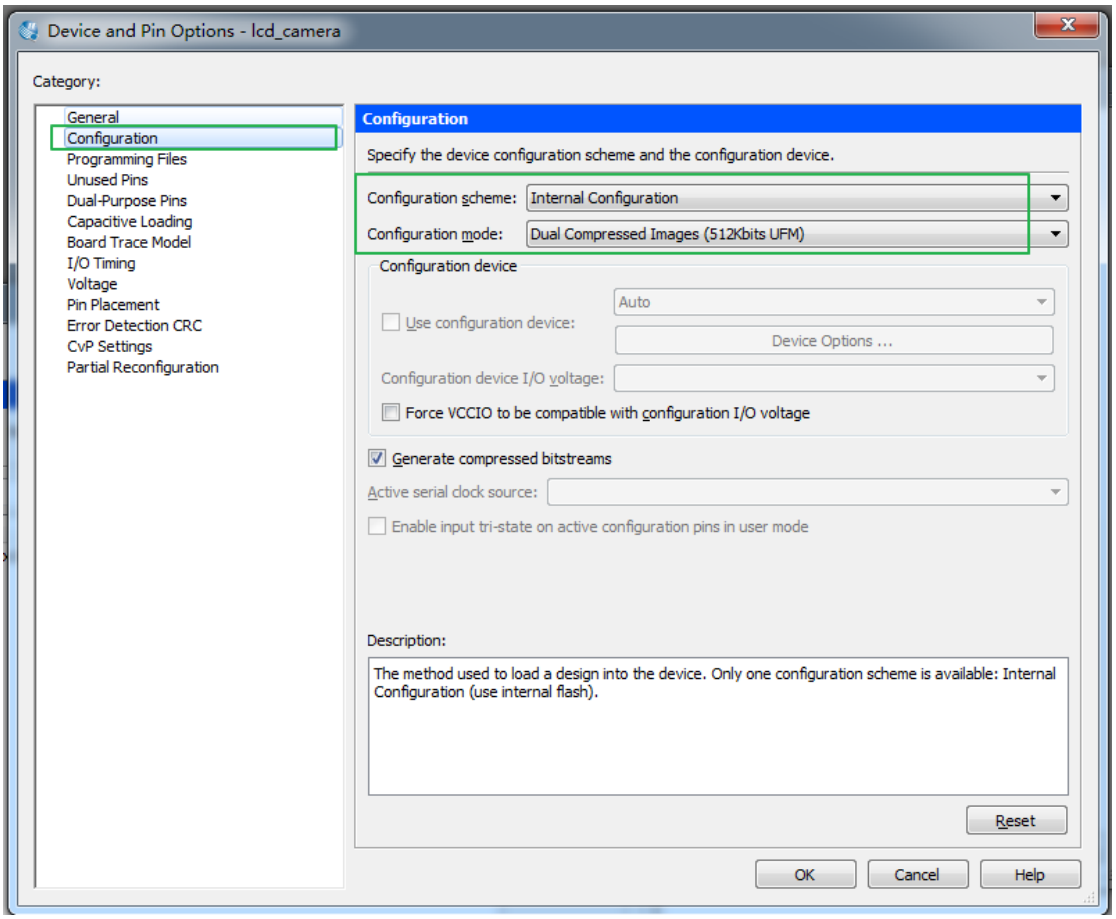

Figure 8-5 Set Dual Configuration Modes

These procedures should be implemented in both projects for dual boot.

- The next step is to convert the two sof files into a pof file for programming the MAX10 FPGA. Open the convert programming Files tool in Quartus and set as **Figure 8-6.**

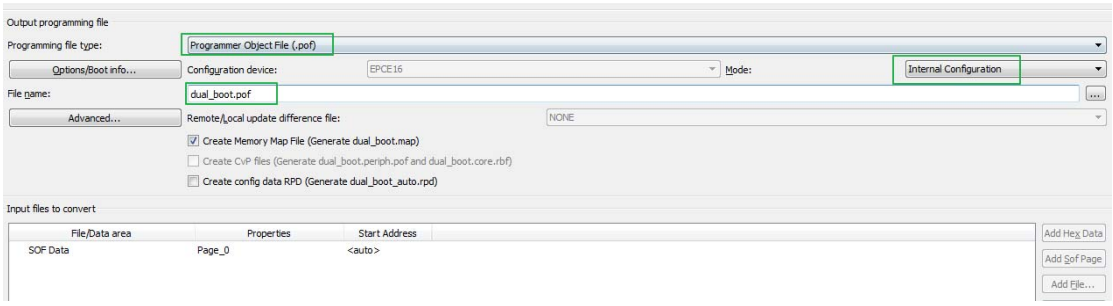

Figure 8-6 Convert File Window Setting

- In the Input files to convert table, click **Add File** button to choose the sof file for image0. Then press the **Add Sof Page** button and add a sof for image1 as shown in **Figure 8-7.** Click generate button to generate the object file—dual_boot.pof.

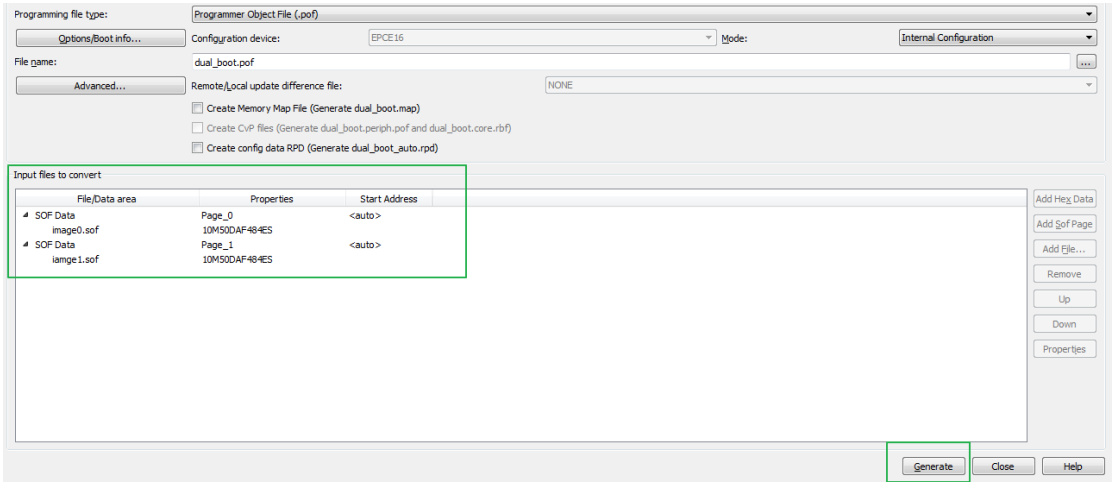

Figure 8-7 Add Sof Files

- The final step is to download the pof into MAX10 FPGA. Open the programmer tool and add the dual_boot.pof as shown in **Figure 8-8.** Click Start button to program the device when the hardware is set OK.

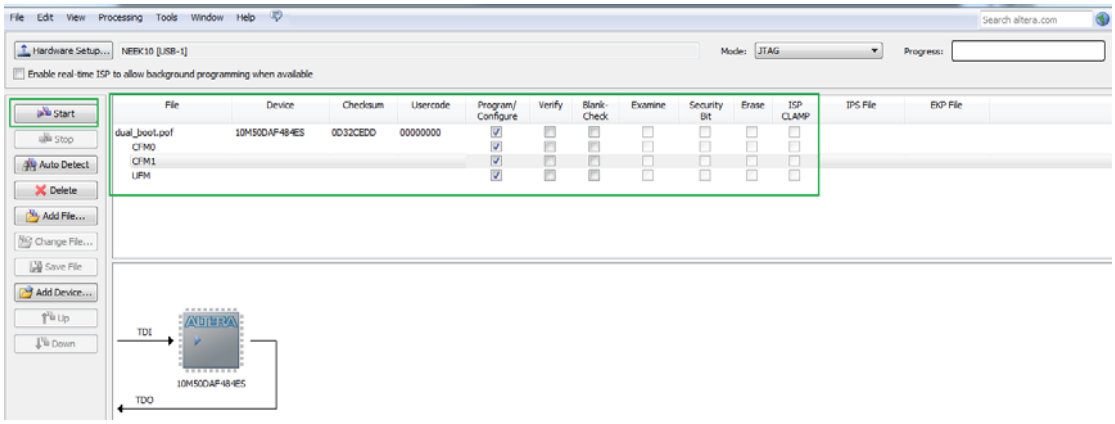

Figure 8-8 Download the pof

- Now, you can set the BOOT_SEL by SW16, you will find if you set BOOT_SEL=0, the image0 will be loaded and if you set BOOT_SEL=1, he image1 will be loaded.

8.3 Nios II Load In Single Boot Image

After the internal configuration, if a Nios II processor is contained in the image and the reset vector of the processor is set to onchip memory data section, the CPU will load the code from the UFM in the onchip flash. The CPU loading procedure occurs either in single image configuration mode or dual compressed image configuration mode. You must make sure only one image contains the processor in dual compressed image configuration mode.

Users need to write the hardware and software binary into the nonvolatile flash memory for the MAX10 FPGA auto boot after power-cycling. The onchip flash memory in the FPGA provides the possibility to boot the software for the Nios II processor.

The demonstration my_first_niosII is designed for Nios II processor loading software after the FPGA configuration complete. The section describes the detailed steps of the design.

- An onchip flash controller ip should be added into the Qsys for storing software code. The CPU data master and instruction master interface are connected to the onchip flash data bus as shown in **Figure 8-9.**

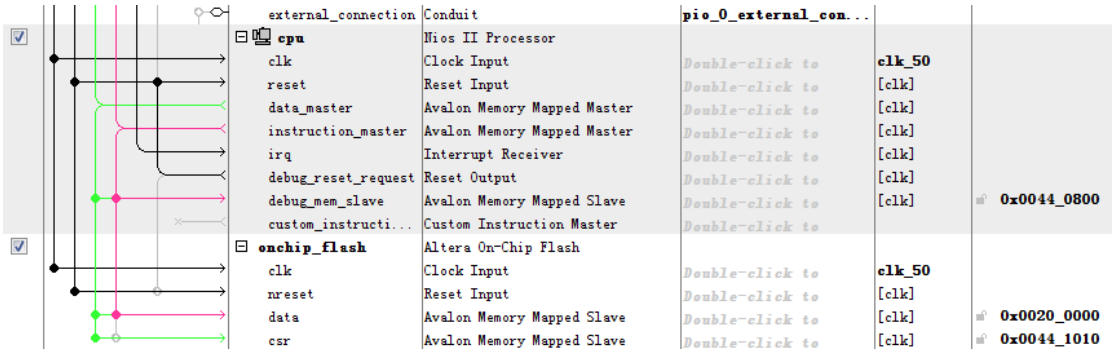

Figure 8-9 Qsys Design

- And the reset vector of the CPU should be onchip flash data section. The configuration mode in the onchip flash parameter setting and the Quartus device setting window should all be set as Single Uncompressed Image.
- In the **BSP Editor (Nios II SBT for Eclipse) utility** of the Eclipse, all the check box in the hal.linker table should be checked as shown in **Figure 8-10**.

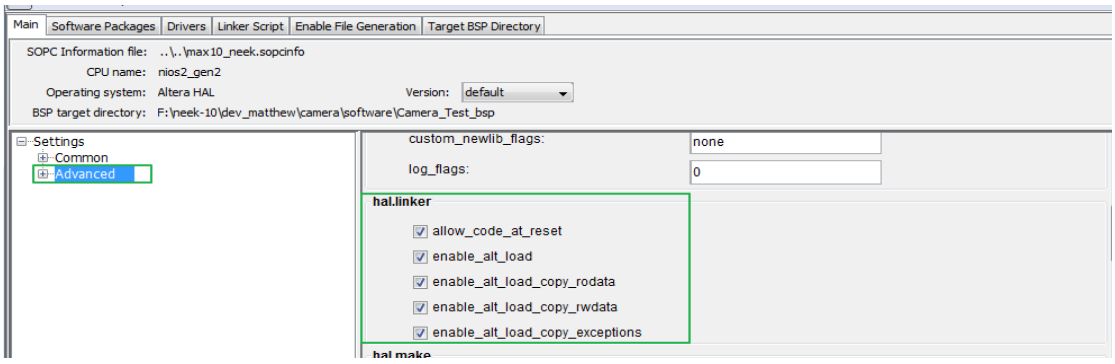

Figure 8-10 BSP Advance Setting

- Make sure you have set your software's program memory (.text section) in onchip Flash memory through BSP Editor (Nios II SBT for Eclipse) utility.
- Compile the target into initial flash format by choosing the mem_init_generate option in the Make Target window in Eclipse as shown in **Figure 8-11.**

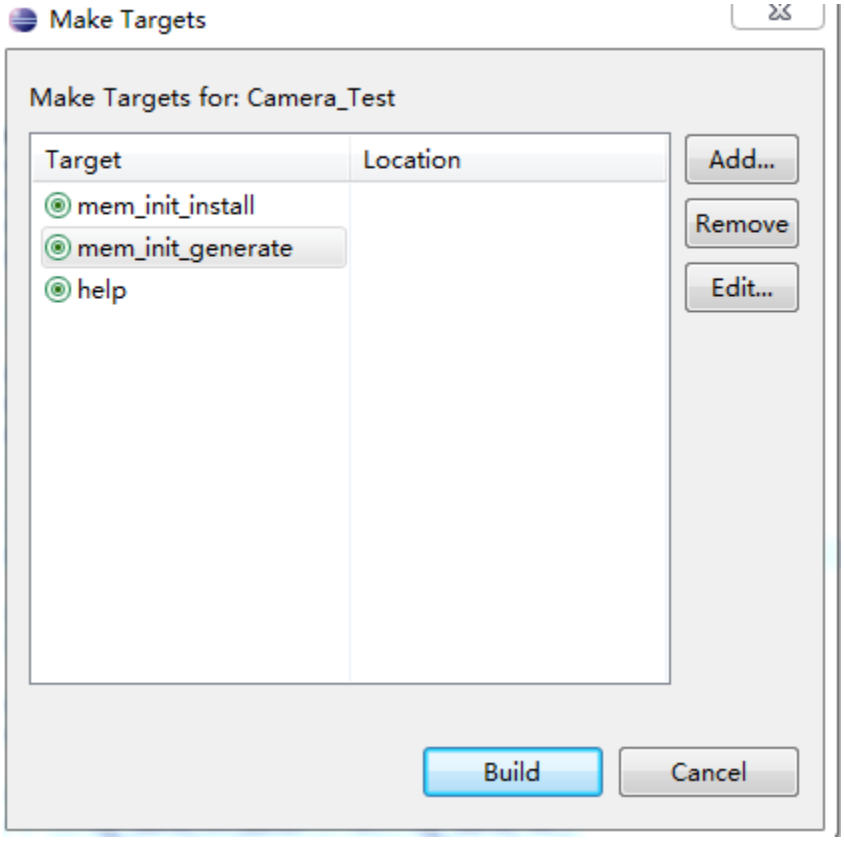

Figure 8-11 Make Target Setting

- After clicking build button, a onchip_flash hex file will be generated in the path software\software_project\mem_init.
- Open the convert programming file window in Quartus II and set the window as shown in **Figure 8-12.**

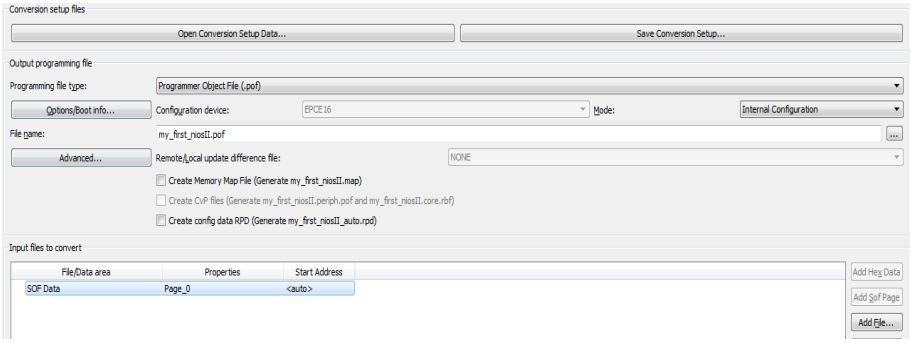

Figure 8-12 Convert File Window Setting

- Click the Options/Boot info… button. Choose the UFM source as the load memory file and click browse button to open the onchip flash hex file as shown in **Figure 8-13**. Press OK button

to close the window.

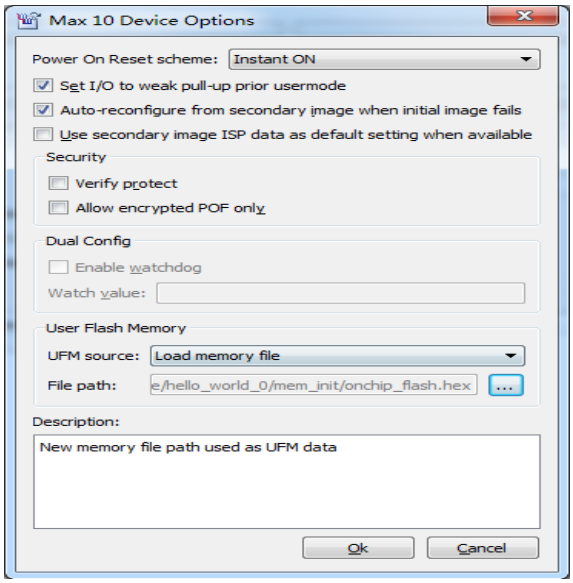

Figure 8-13 Add Hex Data Window

- In the Input files to convert table, choose the sof and generate the pof.
- Open the programmer tool and add the pof generated above to download into the onchip flash. Power cycle the board, the Nios II software will be running after the image has been loaded.

The Nios II processor can also load software from the QSPI flash on MAX 10 NEEK board. Users can refer to Application Selector demonstration for how to design the Nios II processor load software from the QSPI flash.

Chapter 9

Appendix

Revision History

Version	*Change Log*
V1.0	**Initial Version**

Copyright Statement