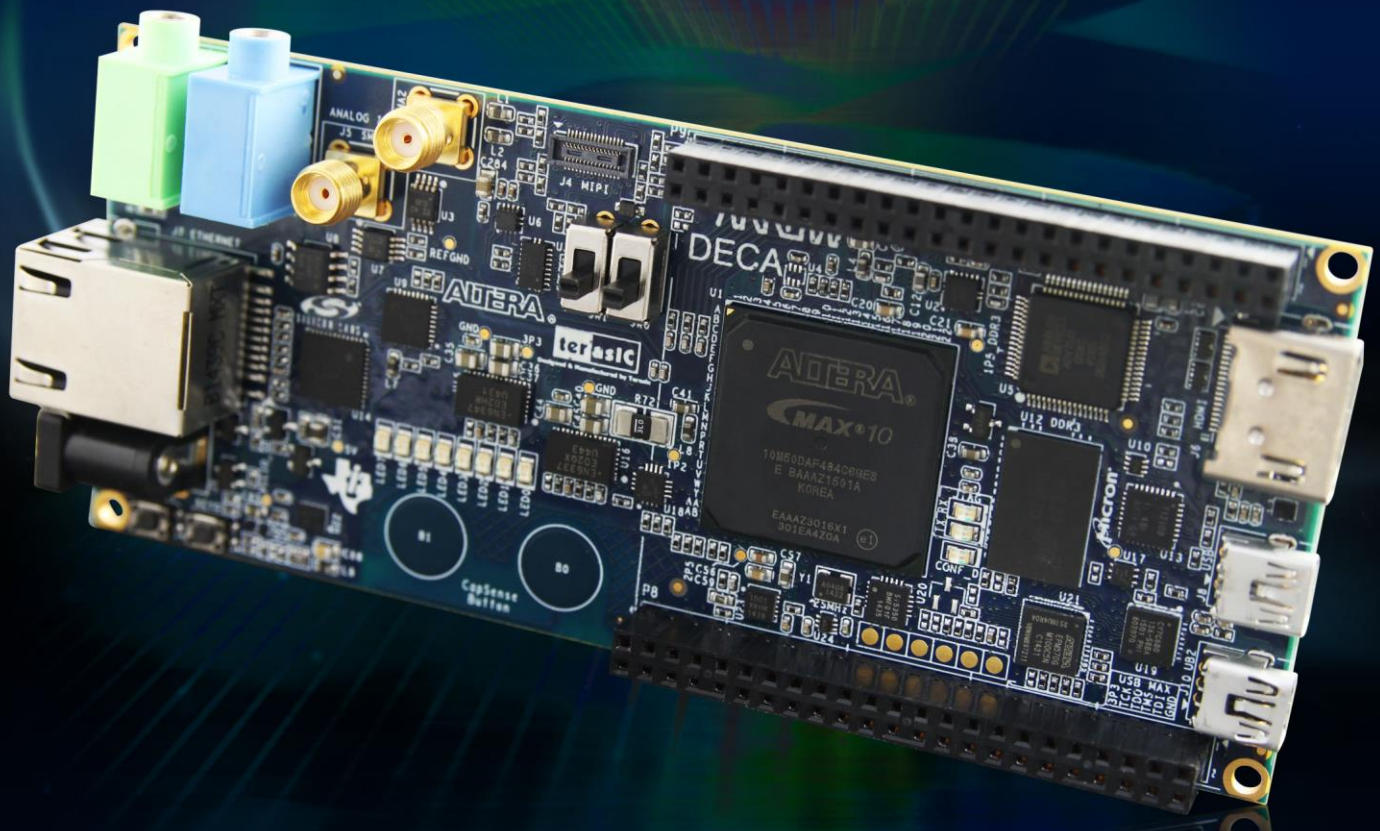
DECA

User Manual



terasic
www.terasic.com

CONTENTS

Chapter 1

DECA Development Kit

The DECA Development Kit presents a robust hardware design platform built around the Altera MAX 10 FPGA, which is the industry's first single chip, non-volatile programmable logic devices (PLDs) to integrate the optimal set of system components. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power FPGA system. Providing internally stored dual images with self-configuration, comprehensive design protection features, integrated ADCs and hardware to implement the Nios II 32-bit microcontroller IP, MAX10 devices are ideal solution for system management, I/O expansion, communication control planes, industrial, automotive and consumer applications. The DECA development board is equipped with high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more that promise many exciting applications.

The DECA Development Kit contains all the tools needed to use the board in conjunction with a computer that runs the Microsoft Windows XP or later.

1.1 Package Contents

Figure 1-1 shows a photograph of the DECA package.

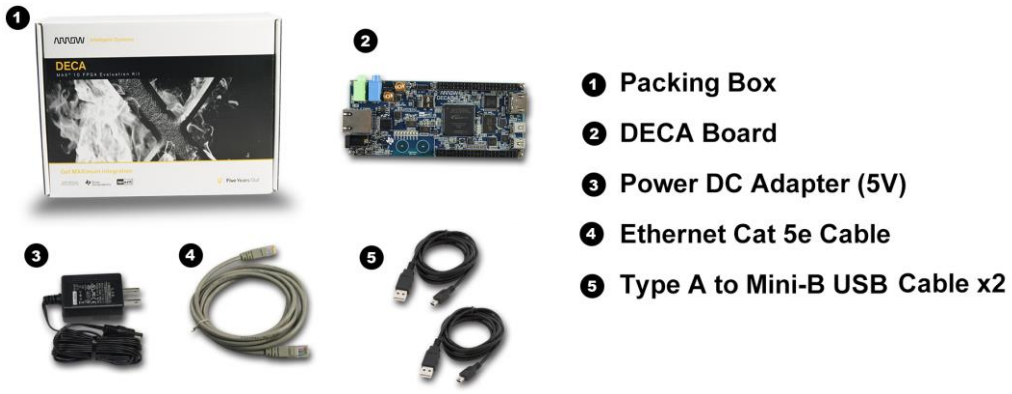


Figure 1-1 The DECA package contents

The DECA package includes:

- The DECA development board
- 5V DC power adapter
- Ethernet Cat 5e Cable
- Two USB cables (Type A to Mini-B) for USB control and FPGA programming and control

1.2 DECA System CD

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

Chapter 2

Introduction of the DECA 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.

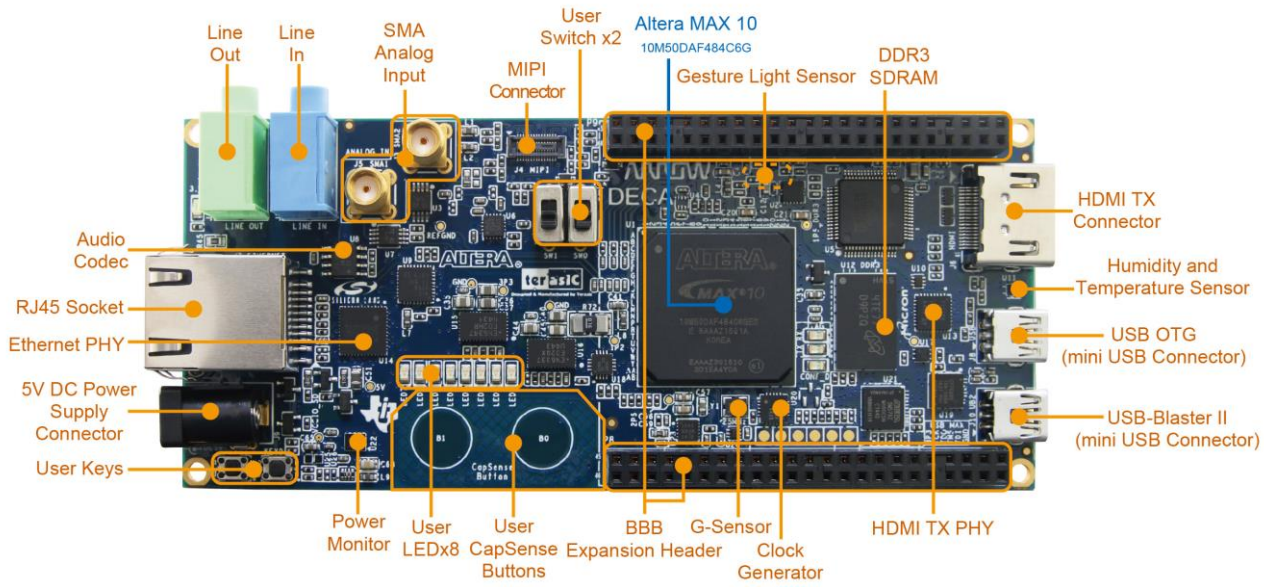


Figure 2-1 DECA development board (top view)

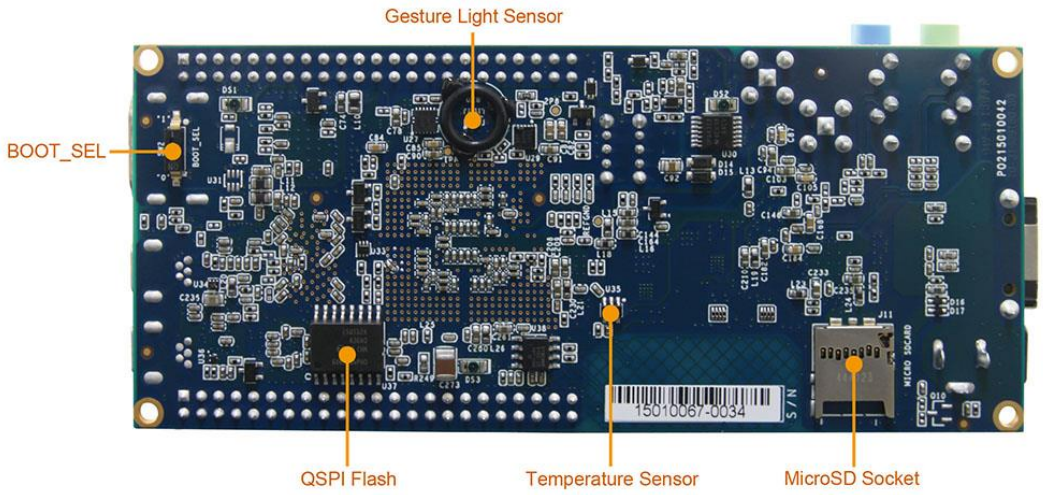


Figure 2-2 DECA development board (bottom view)

The DECA 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
- 512MB DDR3 SDRAM (16-bit data bus)
- 64MB QSPI Flash
- Micro SD card socket
- Two CapSense buttons
- Two push-buttons
- Two slide switches
- Eight blue user LEDs
- Three 50MHz clock sources from the clock generator
- 24-bit CD-quality audio CODEC with line-in, line-out jacks
- HDMI TX, incorporates HDM v1.4 features, including 3D video supporting
- One 10/100 Mbps Ethernet PHY with RJ45 connector
- One USB 2.0 PHY with mini-USB type AB connector
- One MIPI connector interface supports camera module application
- One proximity/ambient light sensor
- One humidity and temperature sensor
- One temperature sensor
- One accelerometer
- Two MAX 10 FPGA ADC SMA inputs

- Two 46-pin BBB expansion headers with 7 analog inputs connected to MAX10 ADC and 69 digital IOs

2.2 Block Diagram of the DECA 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.

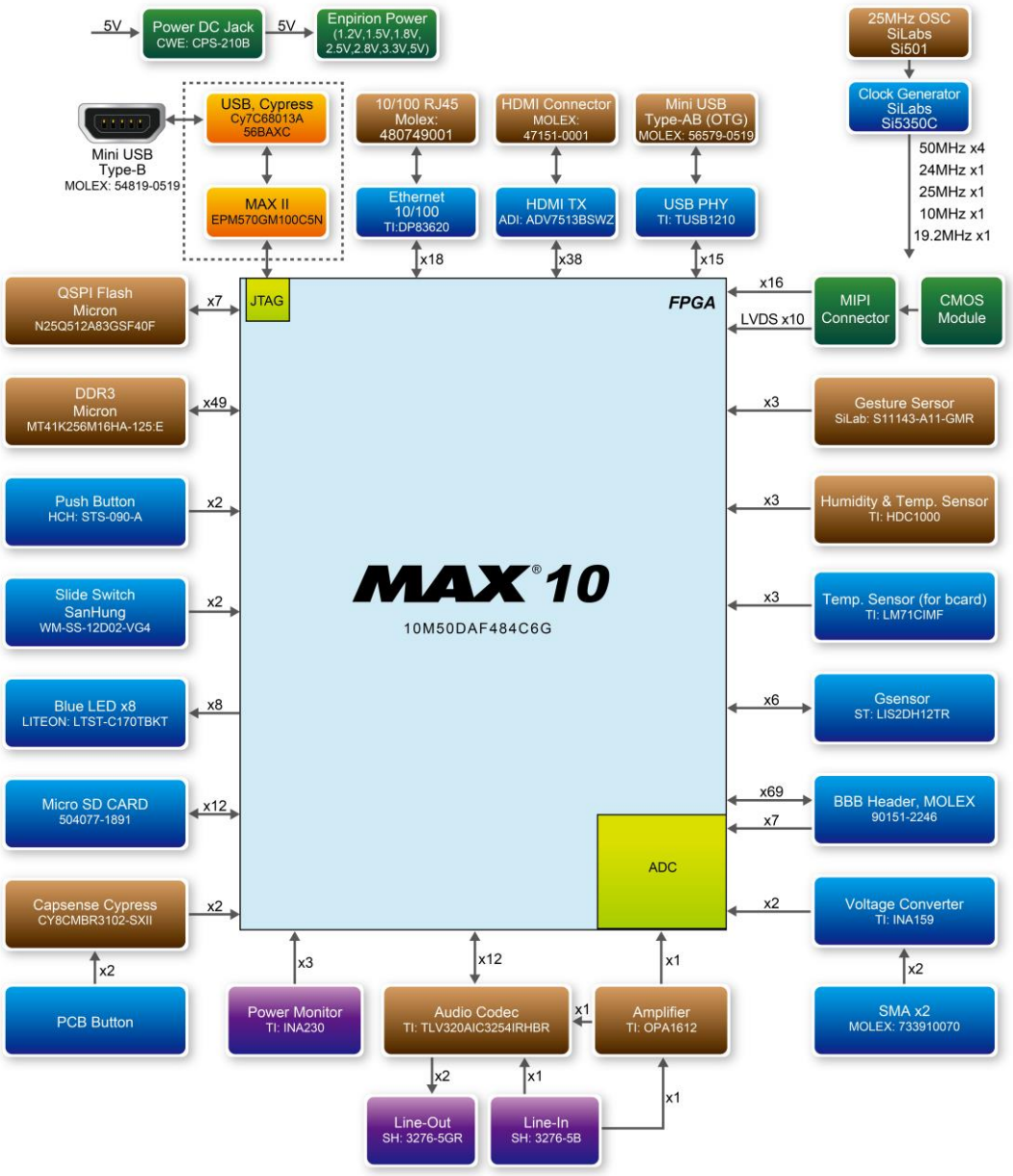


Figure 2-3 Block diagram of DECA

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

- Onboard USB-Blaster II (mini USB type B connector)

Memory Device

- 512MB DDR3 SDRAM (16-bit data bus)
- 64MB QSPI Flash
- Micro SD card socket

Communication

- 10/100 Mbps Ethernet PHY with RJ45 connector
- USB 2.0 PHY with mini-USB type AB connector

Connectors

- Two 46-pin BBB expansion headers
- Two MAX 10 FPGA ADC SMA inputs

Display

- HDMI TX, incorporates HDM v1.4 features, including 3D video supporting

Audio

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

Video Input

- MIPI connector interface supports camera module application

Analog

- Two MAX 10 FPGA ADC SMA inputs
- Seven MAX 10 FPGA ADC inputs from one 46-pin BBB expansion header

Switches, Buttons, and Indicators

- 2 push-buttons
- 2 slide switches
- 8 blue user LEDs

Sensors

- One proximity/ambient lighter sensor
- One humidity and temperature sensor
- One temperature sensor
- One accelerometer

Power

- 5V DC input

Chapter 3

Using the DECA Board

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

3.1 Configuration of MAX 10 FPGA on DECA

There are two types of configuration method supported by DECA:

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 DECA 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 DECA Board

The FPGA device can be configured through JTAG interface on DECA 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 DECA board.

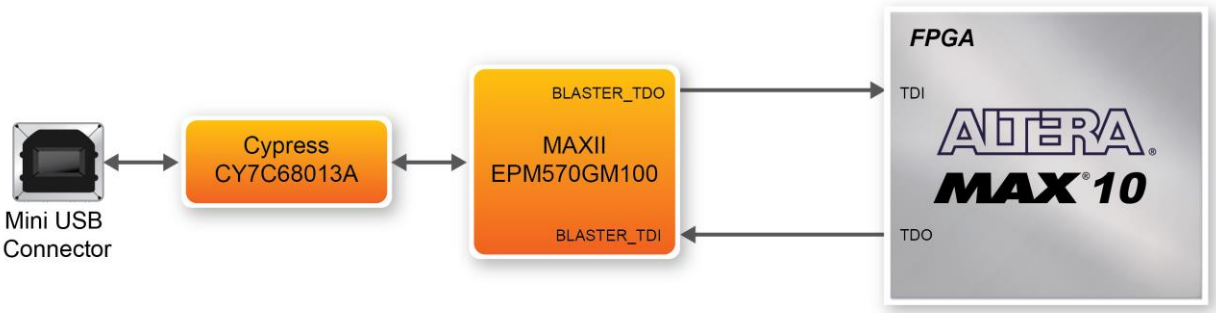


Figure 3-1 Path of the JTAG chain

■ 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**

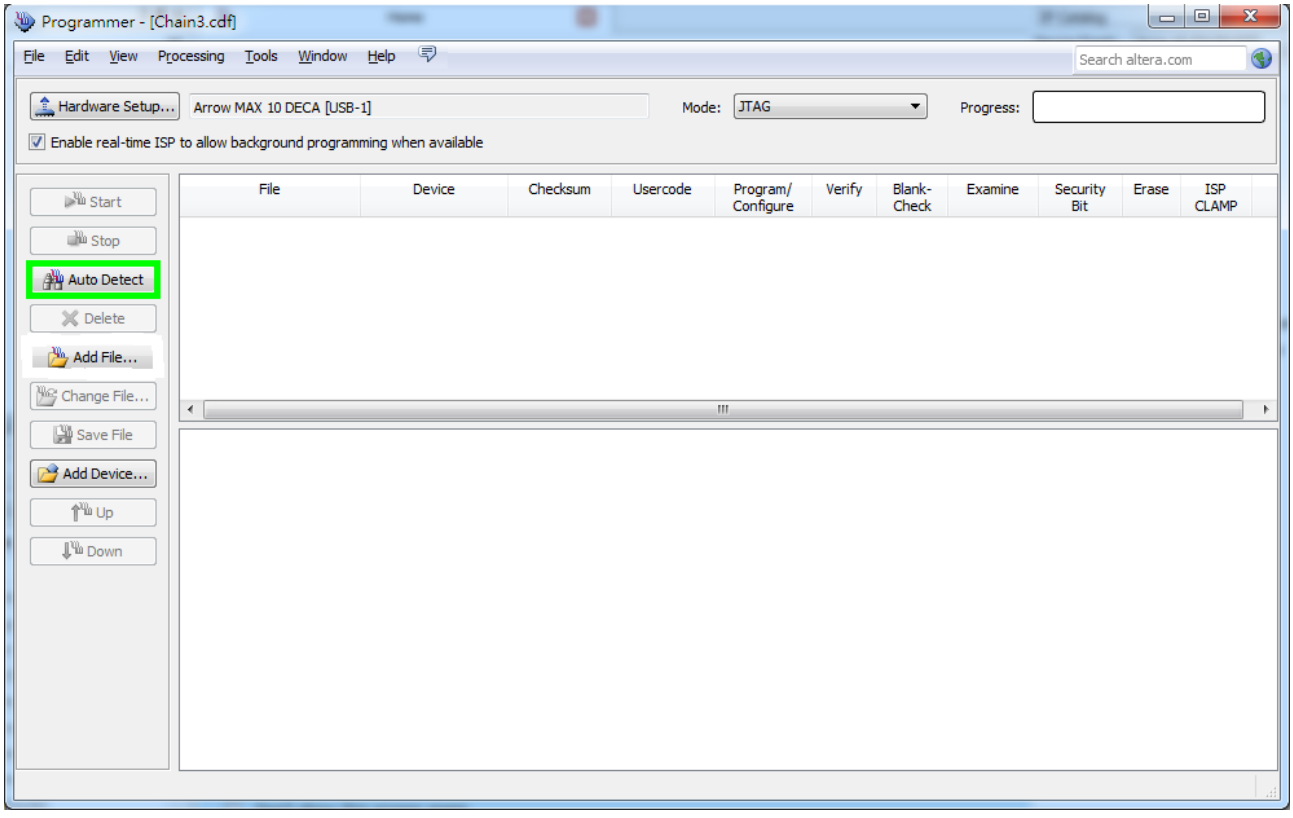


Figure 3-2 Detect FPGA device in JTAG mode

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

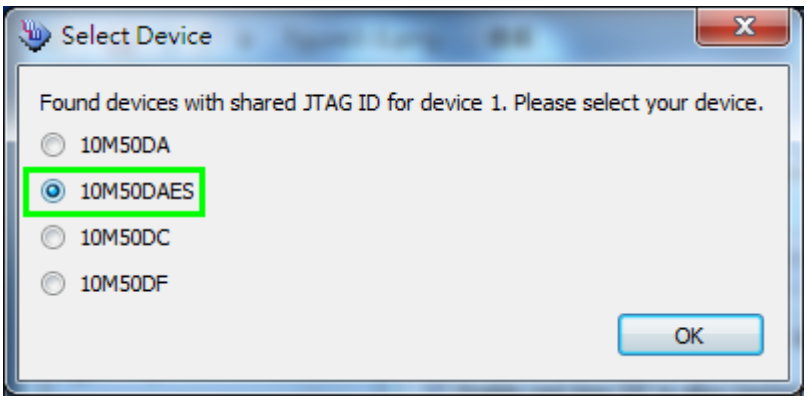


Figure 3-3 Select 10M50DAES device

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

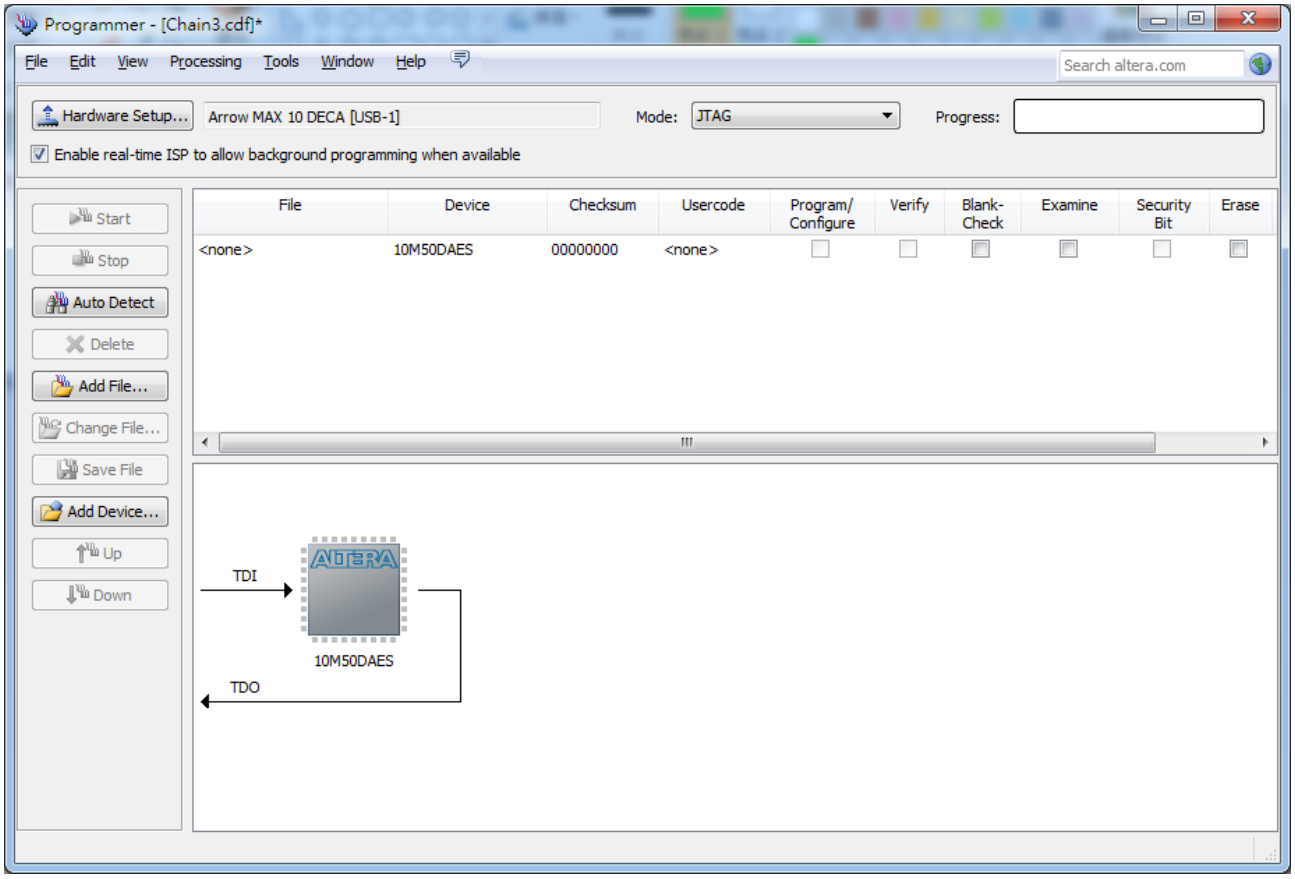


Figure 3-4 FPGA detected in Quartus programmer

4. Right click on the FPGA device and open the .sof file to be programmed, as highlighted in

Figure 3-5.

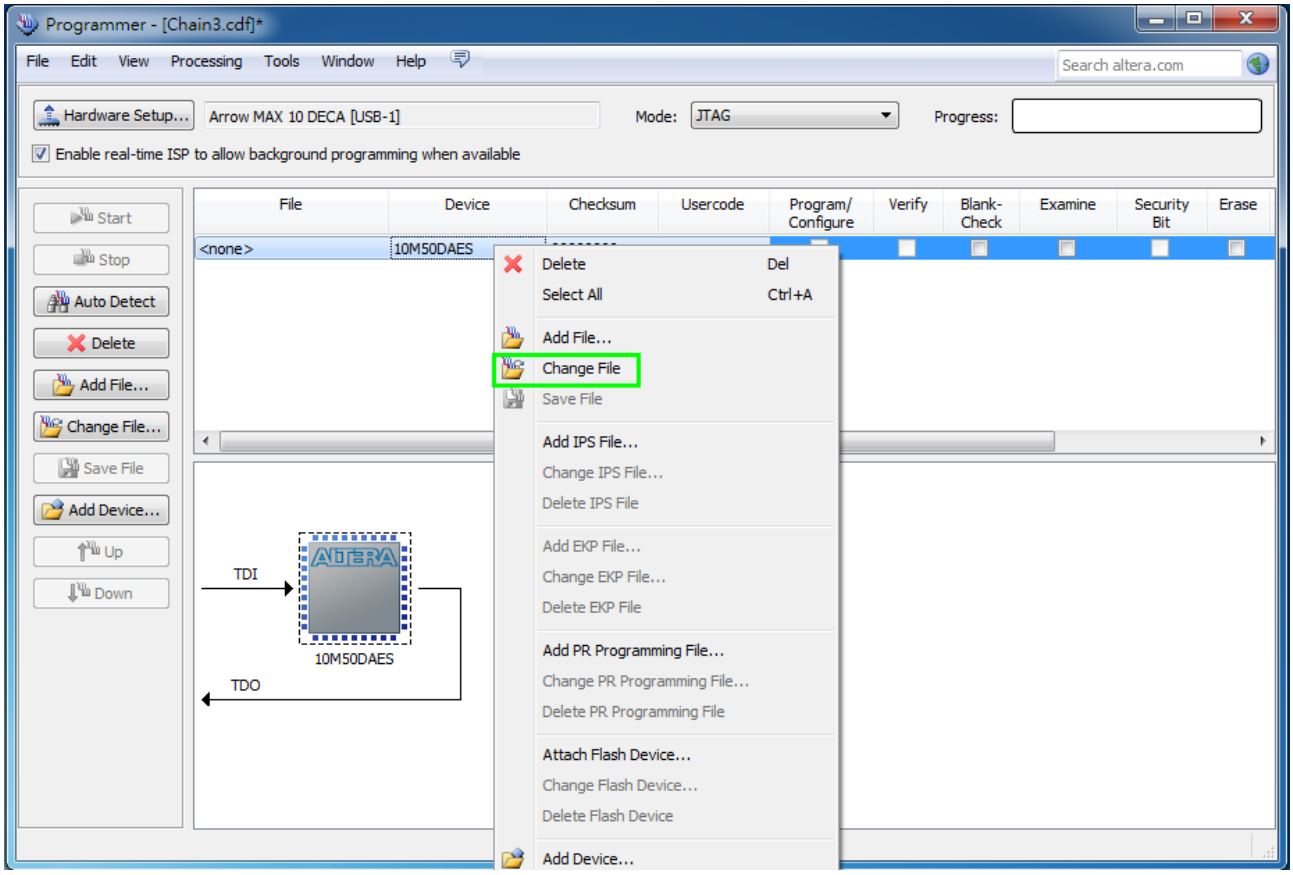


Figure 3-5 Open the .sof file to be programmed into the FPGA device

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

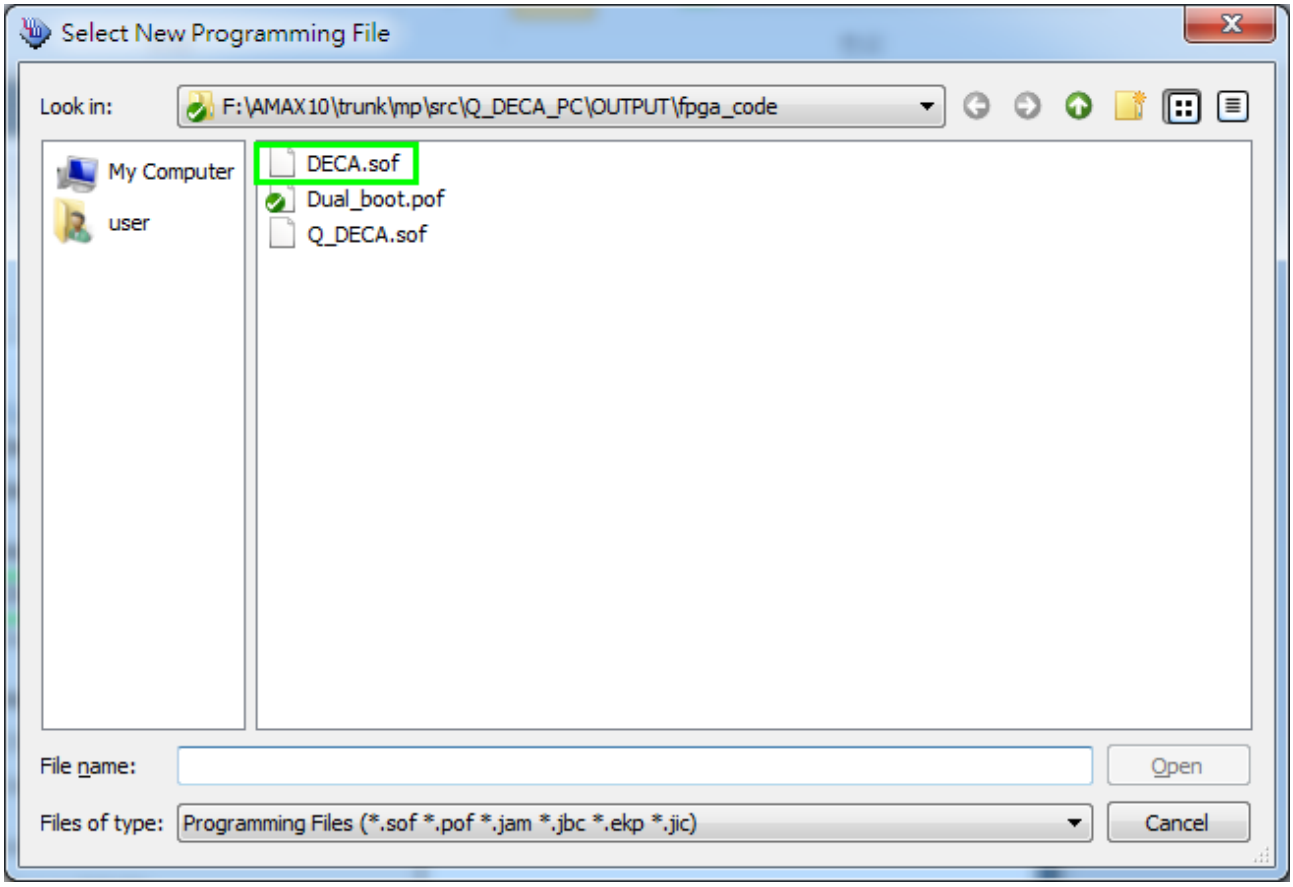


Figure 3-6 Select the .sof file to be programmed into the FPGA device

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**.

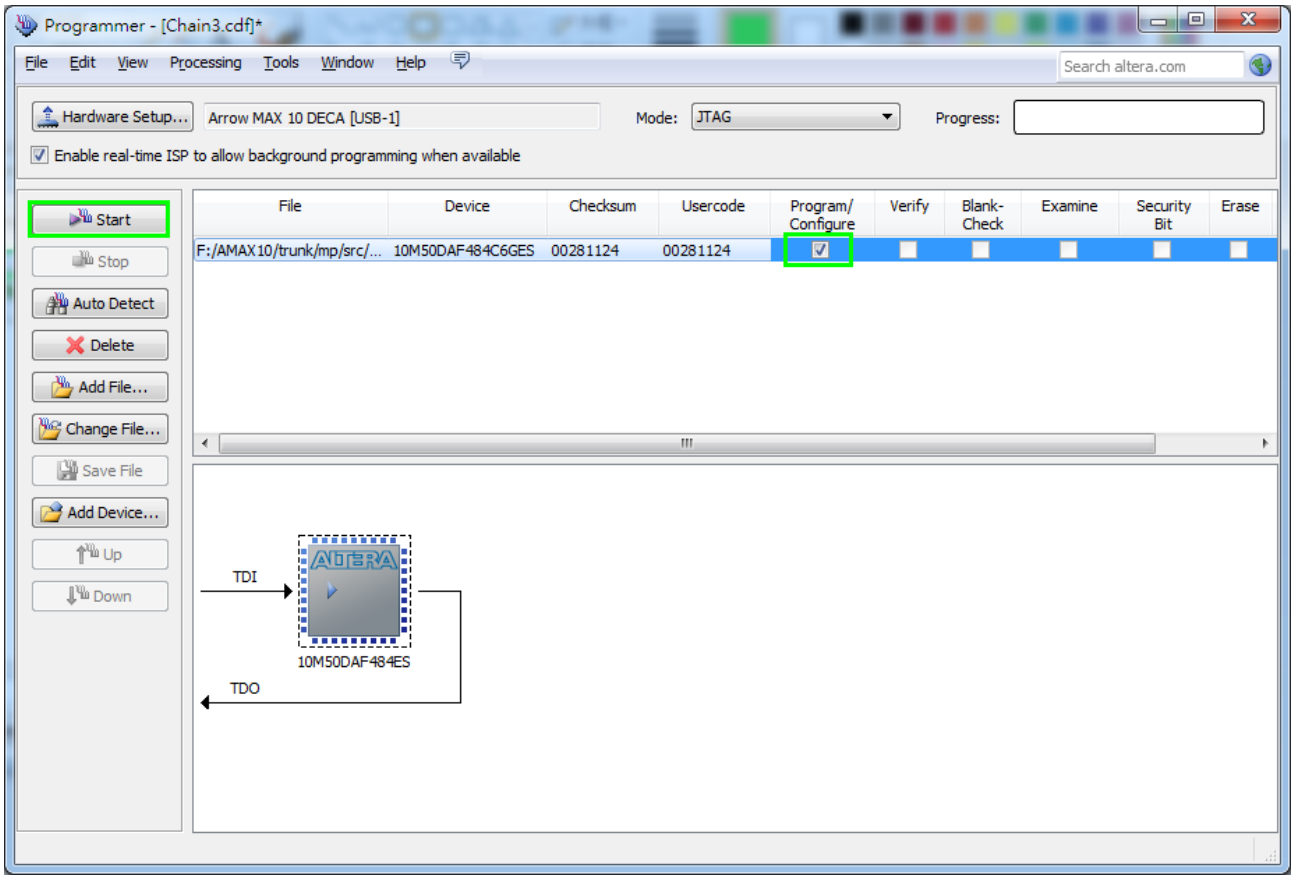


Figure 3-7 Program .sof file into the FPGA device

■ 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).

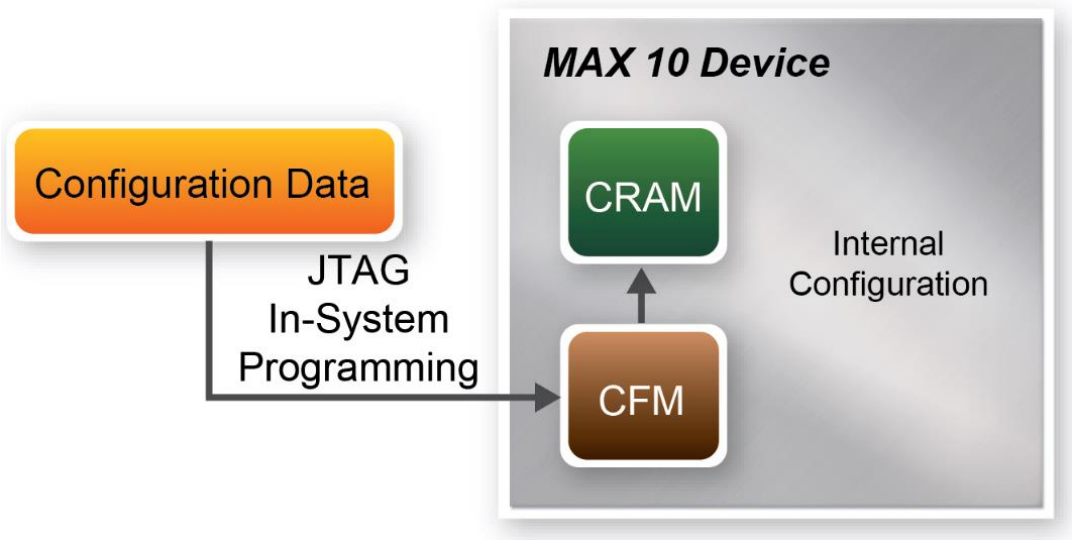


Figure 3-8 High-Level Overview of Internal Configuration for MAX 10 Devices

3.2 Board Status Elements

In addition to the 8 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 DECA

Table 3-1 LED Indicators

Board Reference	*LED Name*	*Description*
D6	3.3V Power	Illuminate when 3.3V power is active.
D8	CONF_DONE	Illuminate when configuration data is loaded into MAX 10 device without error.
D5	JTAG_RX	Illuminate during data is uploaded from MAX 10 device to PC through UB2.

D7	JTAG_TX	Illuminate during configuration data is loaded into MAX 10 device from UB2.

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 19.2MHz clock signal is connected to the reference clock input of USB2.0 PHY transceiver 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.

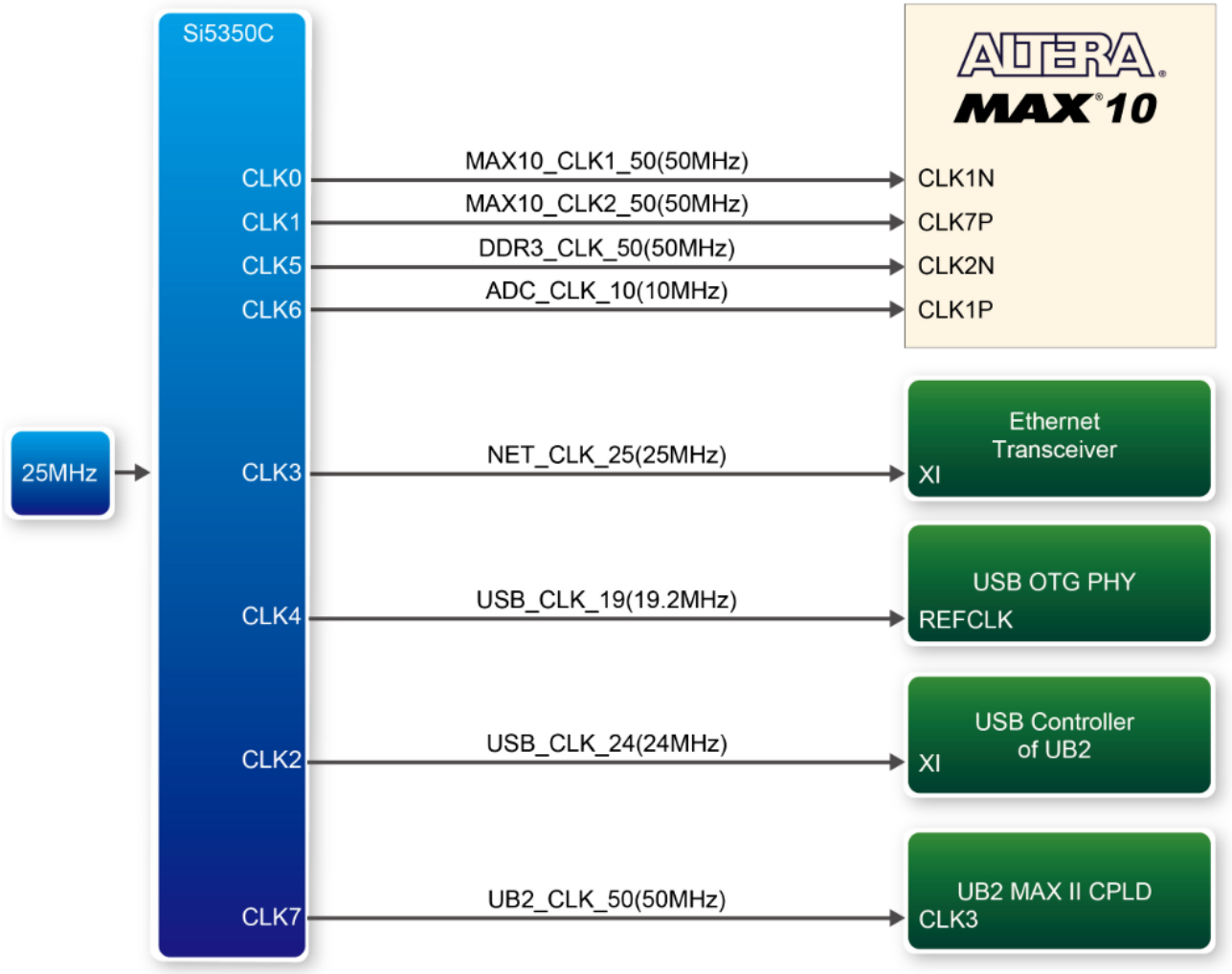


Figure 3-10 Block diagram of the clock distribution on DECA

Table 3-2 Pin Assignment of Clock Inputs

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MAX10_CLK1_50	PIN_M8	50 MHz clock input	2.5V
MAX10_CLK2_50	PIN_P11	50 MHz clock input	3.3V
DDR3_CLK_50	PIN_N15	50 MHz clock input	1.5V
ADC_CLK_10	PIN_M9	10 MHz clock input	2.5V

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, CapSense-buttons, Switches and LEDs

The board has two 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.

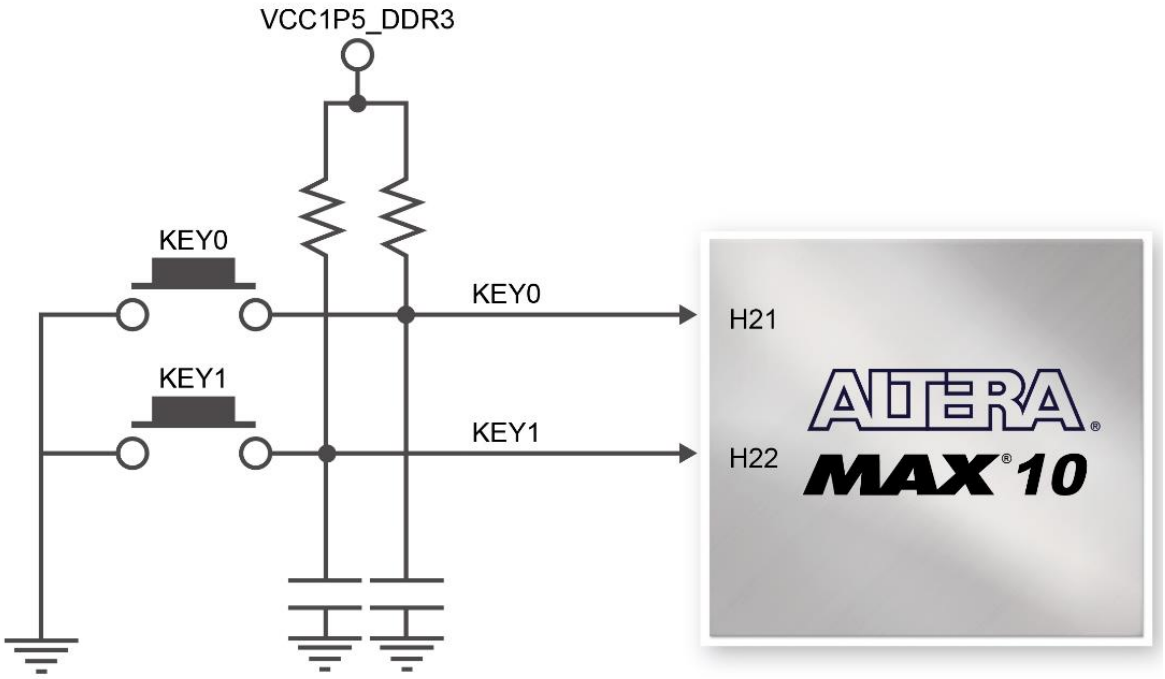


Figure 3-11 Connections between the push-buttons and the MAX 10 FPGA

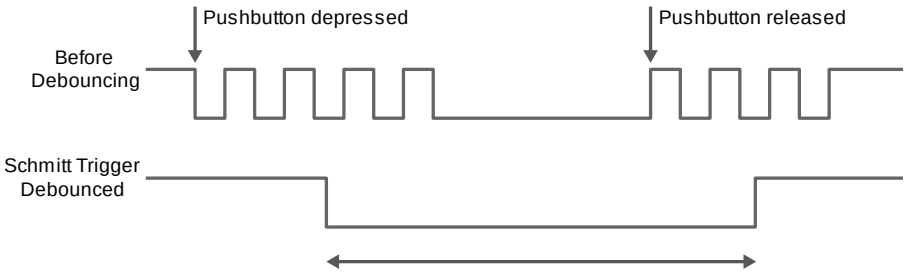


Figure 3-12 Switch debouncing

There are two CapSense buttons connected to the FPGA, as shown in **Figure 3-13**. Through the

CY8CMBR3102 CapSense Express controller, user can implement sleek, reliable capacitive-sensing user interfaces solution to easily replace mechanical buttons. The CapSense controller communicates with FPGA through I2C bus.

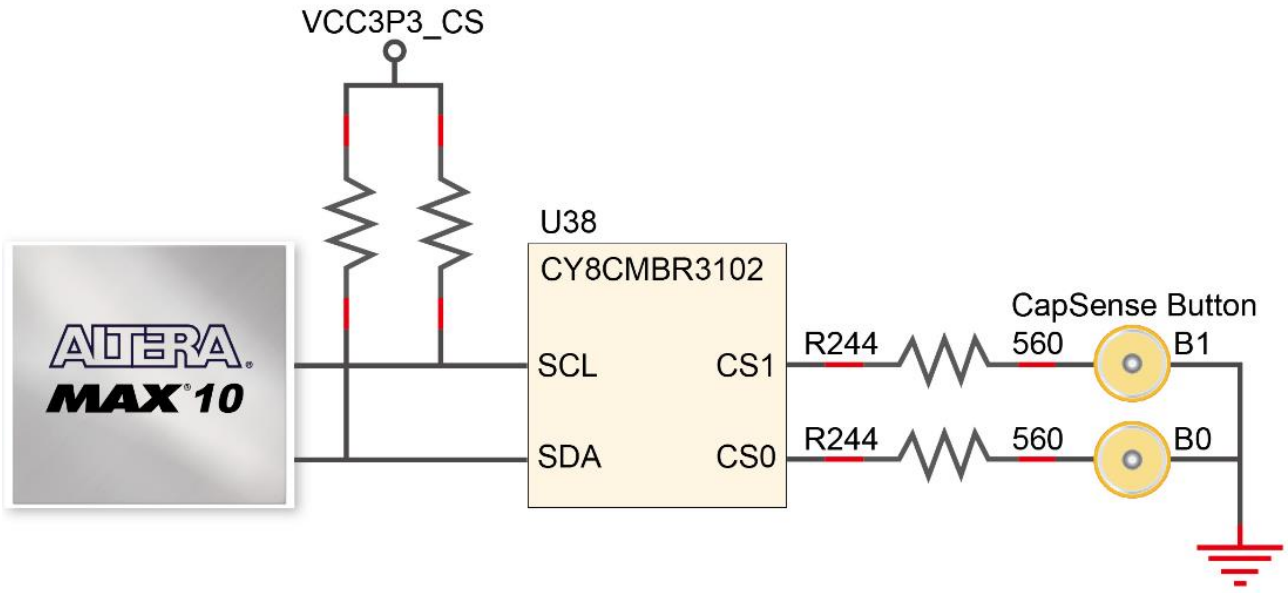


Figure 3-13 Connections between the CapSense-buttons and the MAX 10 FPGA

There are two slide switches connected to the FPGA, as shown in **Figure 3-14**. 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.

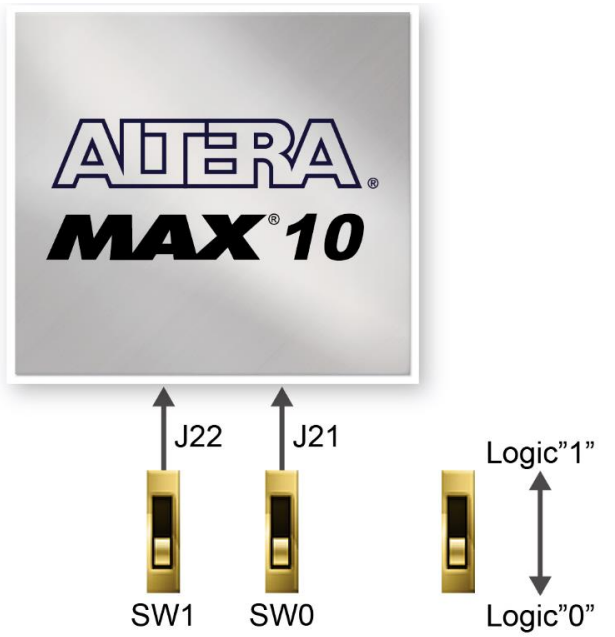


Figure 3-14 Connections between the slide switches and the MAX 10 FPGA

There are also eight 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 off or on, respectively. **Figure 3-15** shows the connections between LEDs and MAX 10 FPGA. **Table 3-3**, **Table 3-4**, **Table 3-5** and **Table 3-6** list the pin assignment of user push-buttons, CapSense-buttons, switches, and LEDs.

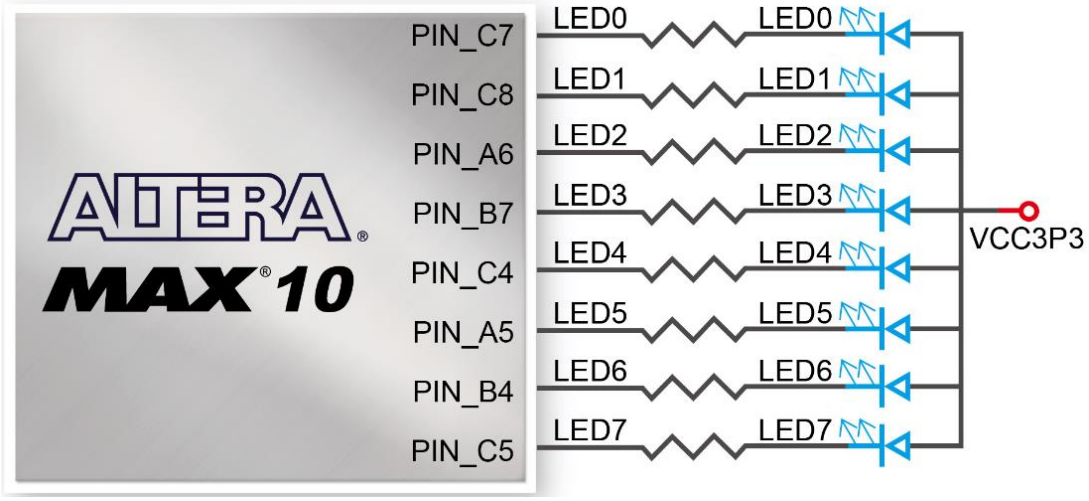


Figure 3-15 Connections between the LEDs and the Cyclone V SoC FPGA

Table 3-3 Pin Assignment of Push-buttons

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY[0]	PIN_H21	Push-button[0]	1.5V
KEY[1]	PIN_H22	Push-button[1]	1.5V

Table 3-4 Pin Assignment of CapSense I2C bus

Signal Name	FPGA Pin No.	Description	I/O Standard
CAP_SENSE_I2C_SCL	PIN_AB2	CapSense SCL	3.3V
CAP_SENSE_I2C_SCA	PIN_AB3	CapSense SDA	3.3V

Table 3-5 Pin Assignment of Slide Switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_J21	Slide Switch[0]	1.5V
SW[1]	PIN_J22	Slide Switch[1]	1.5V

Table 3-6 Pin Assignment of LEDs

Signal Name	FPGA Pin No.	Description	I/O Standard
LED[0]	PIN_C7	LED [0]	1.2V
LED[1]	PIN_C8	LED [1]	1.2V
LED[2]	PIN_A6	LED [2]	1.2V
LED[3]	PIN_B7	LED [3]	1.2V
LED[4]	PIN_C4	LED [4]	1.2V
LED[5]	PIN_A5	LED [5]	1.2V
LED[6]	PIN_B4	LED [6]	1.2V
LED[7]	PIN_C5	LED [7]	1.2V

3.4.2 Power Monitor

The DECA has implemented a power monitor chip to monitor the FPGA core power voltage and current. **Figure 3-16** shows the connection between the power monitor chip and the MAX 10 FPGA. The power monitor chip monitors both shunt voltage drops and FPGA core power voltage. Programmable calibration value, conversion times, and averaging, combined with an internal multiplier, enable direct readouts of current in amperes and power in watts. **Table 3-7** shows the pin assignment of power monitor I2C bus.

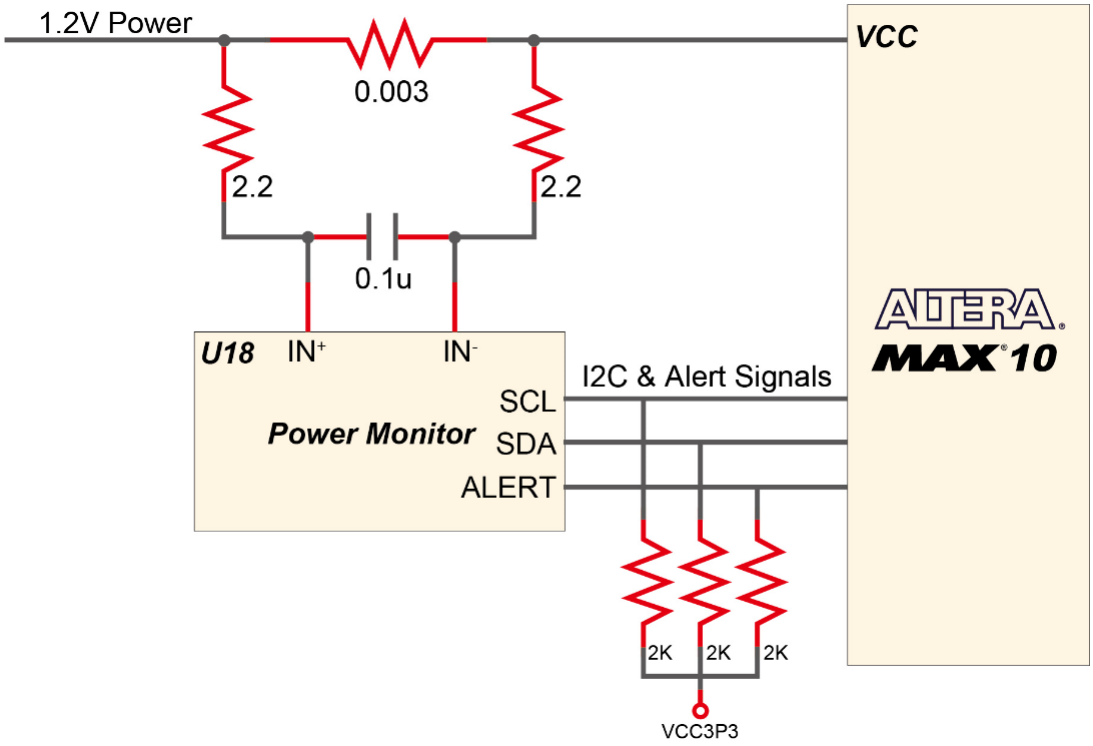


Figure 3-16 Connections between the power monitor chip and the MAX 10 FPGA

Table 3-7 Pin Assignment of Power Monitor I2C bus

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
PMONITOR_I2C_SCL	PIN_Y3	Power Monitor SCL	3.3V
PMONITOR_I2C_SDA	PIN_Y1	Power Monitor SDA	3.3V
PMONITOR_ALERT	PIN_Y4	Power Monitor ALERT	3.3V

3.4.3 Two 2x23 Expansion Headers

The board has two 46-pin expansion headers. The P8 header has 44 digital user pins connected directly to the MAX 10 FPGA, the P9 header has 25 digital and 7 analog user pins connected to the MAX 10 FPGA. The P9 header also comes with DC +5V (VCC5), DC +3.3V (VCC3P3) power pins. The maximum power consumption allowed for a daughter card connected to one or two GPIO ports is shown in Table 3-8.

Table 3-8 Voltage and Max. Current Limit of Expansion Header(s)

Supplied Voltage	*Max. Current Limit*
5V	1A
3.3V	1A

Figure 3-17 shows the connection between the two 2x23 headers and MAX 10 FPGA. Table 3-9 shows the pin assignment of two 2x23 headers.

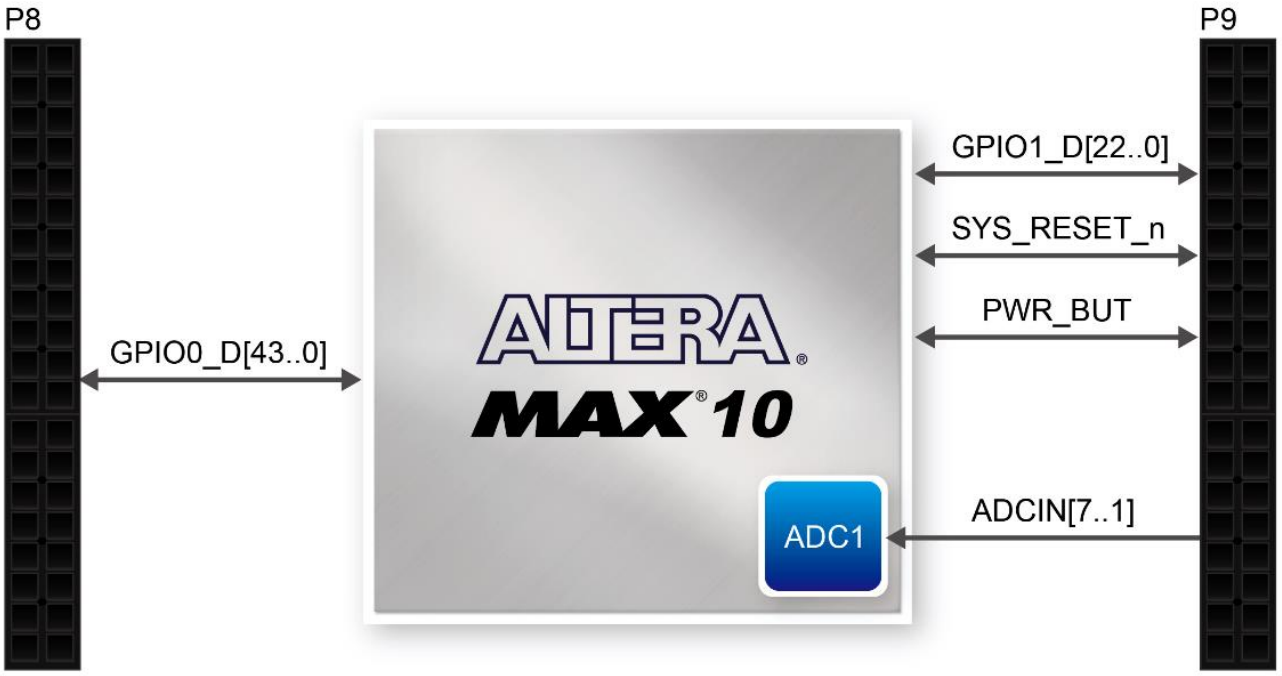


Figure 3-17 Connections between the two 2x23 header and MAX 10 FPGA

Table 3-9 Pin Assignment of Expansion Headers

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
GPIO0_D[0]	PIN_W18	GPIO (P8) Connection 0[0]	3.3V
GPIO0_D[1]	PIN_Y18	GPIO (P8) Connection 0[1]	3.3V
GPIO0_D[2]	PIN_Y19	GPIO (P8) Connection 0[2]	3.3V
GPIO0_D[3]	PIN_AA17	GPIO (P8) Connection 0[3]	3.3V
GPIO0_D[4]	PIN_AA20	GPIO (P8) Connection 0[4]	3.3V
GPIO0_D[5]	PIN_AA19	GPIO (P8) Connection 0[5]	3.3V
GPIO0_D[6]	PIN_AB21	GPIO (P8) Connection 0[6]	3.3V
GPIO0_D[7]	PIN_AB20	GPIO (P8) Connection 0[7]	3.3V
GPIO0_D[8]	PIN_AB19	GPIO (P8) Connection 0[8]	3.3V
GPIO0_D[9]	PIN_Y16	GPIO (P8) Connection 0[9]	3.3V
GPIO0_D[10]	PIN_V16	GPIO (P8) Connection 0[10]	3.3V
GPIO0_D[11]	PIN_AB18	GPIO (P8) Connection 0[11]	3.3V
GPIO0_D[12]	PIN_V15	GPIO (P8) Connection 0[12]	3.3V
GPIO0_D[13]	PIN_W17	GPIO (P8) Connection 0[13]	3.3V
GPIO0_D[14]	PIN_AB17	GPIO (P8) Connection 0[14]	3.3V
GPIO0_D[15]	PIN_AA16	GPIO (P8) Connection 0[15]	3.3V
GPIO0_D[16]	PIN_AB16	GPIO (P8) Connection 0[16]	3.3V
GPIO0_D[17]	PIN_W16	GPIO (P8) Connection 0[17]	3.3V
GPIO0_D[18]	PIN_AB15	GPIO (P8) Connection 0[18]	3.3V

GPIO0_D[19]	PIN_W15	GPIO (P8) Connection 0[19]	3.3V
GPIO0_D[20]	PIN_Y14	GPIO (P8) Connection 0[20]	3.3V
GPIO0_D[21]	PIN_AA15	GPIO (P8) Connection 0[21]	3.3V
GPIO0_D[22]	PIN_AB14	GPIO (P8) Connection 0[22]	3.3V
GPIO0_D[23]	PIN_AA14	GPIO (P8) Connection 0[23]	3.3V
GPIO0_D[24]	PIN_AB13	GPIO (P8) Connection 0[24]	3.3V
GPIO0_D[25]	PIN_AA13	GPIO (P8) Connection 0[25]	3.3V
GPIO0_D[26]	PIN_AB12	GPIO (P8) Connection 0[26]	3.3V
GPIO0_D[27]	PIN_AA12	GPIO (P8) Connection 0[27]	3.3V
GPIO0_D[28]	PIN_AB11	GPIO (P8) Connection 0[28]	3.3V
GPIO0_D[29]	PIN_AA11	GPIO (P8) Connection 0[29]	3.3V
GPIO0_D[30]	PIN_AB10	GPIO (P8) Connection 0[30]	3.3V
GPIO0_D[31]	PIN_Y13	GPIO (P8) Connection 0[31]	3.3V
GPIO0_D[32]	PIN_Y11	GPIO (P8) Connection 0[32]	3.3V
GPIO0_D[33]	PIN_W13	GPIO (P8) Connection 0[33]	3.3V
GPIO0_D[34]	PIN_W12	GPIO (P8) Connection 0[34]	3.3V
GPIO0_D[35]	PIN_W11	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[36]	PIN_V12	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[37]	PIN_V11	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[38]	PIN_V13	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[39]	PIN_V14	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[40]	PIN_Y17	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[41]	PIN_W14	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[42]	PIN_U15	GPIO (P8) Connection 0[35]	3.3V
GPIO0_D[43]	PIN_R13	GPIO (P8) Connection 0[35]	3.3V
GPIO1_D[0]	PIN_Y5	GPIO (P9) Connection 1[0]	3.3V
GPIO1_D[1]	PIN_Y6	GPIO (P9) Connection 1[1]	3.3V
GPIO1_D[2]	PIN_W6	GPIO (P9) Connection 1[2]	3.3V
GPIO1_D[3]	PIN_W7	GPIO (P9) Connection 1[3]	3.3V
GPIO1_D[4]	PIN_W8	GPIO (P9) Connection 1[4]	3.3V
GPIO1_D[5]	PIN_V8	GPIO (P9) Connection 1[5]	3.3V
GPIO1_D[6]	PIN_AB8	GPIO (P9) Connection 1[6]	3.3V
GPIO1_D[7]	PIN_V7	GPIO (P9) Connection 1[7]	3.3V
GPIO1_D[8]	PIN_R11	GPIO (P9) Connection 1[8]	3.3V
GPIO1_D[9]	PIN_AB7	GPIO (P9) Connection 1[9]	3.3V
GPIO1_D[10]	PIN_AB6	GPIO (P9) Connection 1[10]	3.3V
GPIO1_D[11]	PIN_AA7	GPIO (P9) Connection 1[11]	3.3V
GPIO1_D[12]	PIN_AA6	GPIO (P9) Connection 1[12]	3.3V
GPIO1_D[13]	PIN_Y7	GPIO (P9) Connection 1[13]	3.3V
GPIO1_D[14]	PIN_V10	GPIO (P9) Connection 1[14]	3.3V
GPIO1_D[15]	PIN_U7	GPIO (P9) Connection 1[15]	3.3V
GPIO1_D[16]	PIN_W9	GPIO (P9) Connection 1[16]	3.3V
GPIO1_D[17]	PIN_W5	GPIO (P9) Connection 1[17]	3.3V
GPIO1_D[18]	PIN_R9	GPIO (P9) Connection 1[18]	3.3V
GPIO1_D[19]	PIN_W4	GPIO (P9) Connection 1[19]	3.3V

GPIO1_D[20]	PIN_P9	GPIO (P9) Connection 1[20]	3.3V
GPIO1_D[21]	PIN_V17	GPIO (P9) Connection 1[21]	3.3V
GPIO1_D[22]	PIN_W3	GPIO (P9) Connection 1[22]	3.3V
SYS_RESET_n	PIN_AA2	GPIO (P9) Connection SYS_RESET_n	3.3V
PWR_BUT	PIN_U6	GPIO (P9) Connection PWR_BUT	3.3V
ADC1IN1	PIN_F5	GPIO (P9) ADC1 Analog Input [1]	2.5V
ADC1IN2	PIN_F4	GPIO (P9) ADC1 Analog Input [2]	2.5V
ADC1IN3	PIN_J8	GPIO (P9) ADC1 Analog Input [3]	2.5V
ADC1IN4	PIN_J9	GPIO (P9) ADC1 Analog Input [4]	2.5V
ADC1IN5	PIN_J4	GPIO (P9) ADC1 Analog Input [5]	2.5V
ADC1IN6	PIN_H3	GPIO (P9) ADC1 Analog Input [6]	2.5V
ADC1IN7	PIN_K5	GPIO (P9) ADC1 Analog Input [7]	2.5V

3.4.4 24-bit Audio CODEC

The DECA offers high-quality 24-bit audio via the Texas Instruments TLV320AIC3254 audio CODEC (Encoder/Decoder). This chip on DECA supports, line-in, and line-out ports. One of line-in inputs is both connected to the FPGA ADC and the audio CODEC ADC, it allows user to implement audio applications via the MAX 10 build-in ADC. The operational amplifier OPA1612 is used to make impedance match for the two fanouts from one line-in input. 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 3-10**. More information about the TLV320AIC3254 CODEC is available in its datasheet, which can be found on the manufacturer's website, or in the directory \DECA_datasheets\Audio CODEC of DECA System CD.

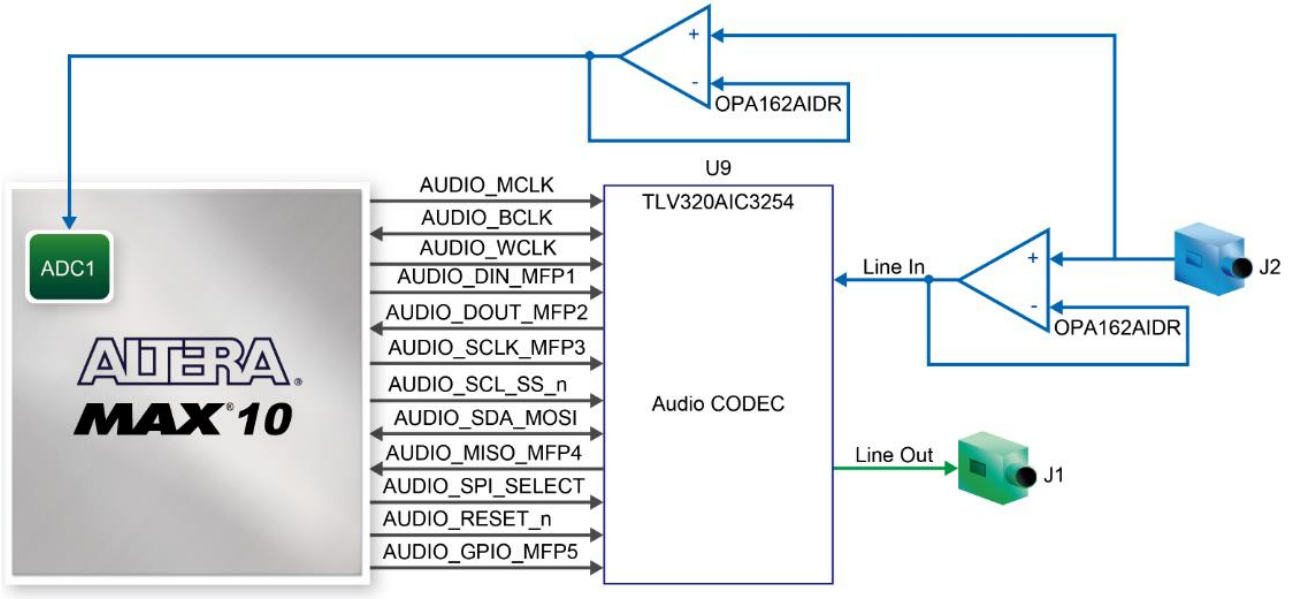


Figure 3-18 Connections between the FPGA and audio CODEC

Table 3-10 Pin Assignment of Audio CODEC

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
AUDIO_MCLK	PIN_P14	Master output Clock	1.5V
AUDIO_BCLK	PIN_R14	Audio serial data bus (primary) bit clock	1.5V
AUDIO_WCLK	PIN_R15	Audio serial data bus (primary) word clock	1.5V
AUDIO_DIN_MFP1	PIN_P15	Audio serial data bus data output/digital microphone output	1.5V
AUDIO_DOUT_MFP2	PIN_P18	Audio serial data bus data input/general purpose input	1.5V
AUDIO_SCLK_MFP3	PIN_P19	SPI serial Clock/headphone-detect output	1.5V
AUDIO_SCL_SS_n	PIN_P20	I2C Clock/SPI interface mode chip-select signal	1.5V
AUDIO_SDA_MOSI	PIN_P21	I2C Data/SPI interface mode serial data output	1.5V
AUDIO_MISO_MFP4	PIN_N21	Serial data input/General purpose input	1.5V
AUDIO_SPI_SELECT	PIN_N22	Control mode select pin	1.5V
AUDIO_RESET_n	PIN_M21	Reset signal	1.5V
AUDIO_GPIO_MFP5	PIN_M22	General Purpose digital IO/CLKOUT input	1.5V

3.4.5 Two Analog Input SMA Connectors

The DECA 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.

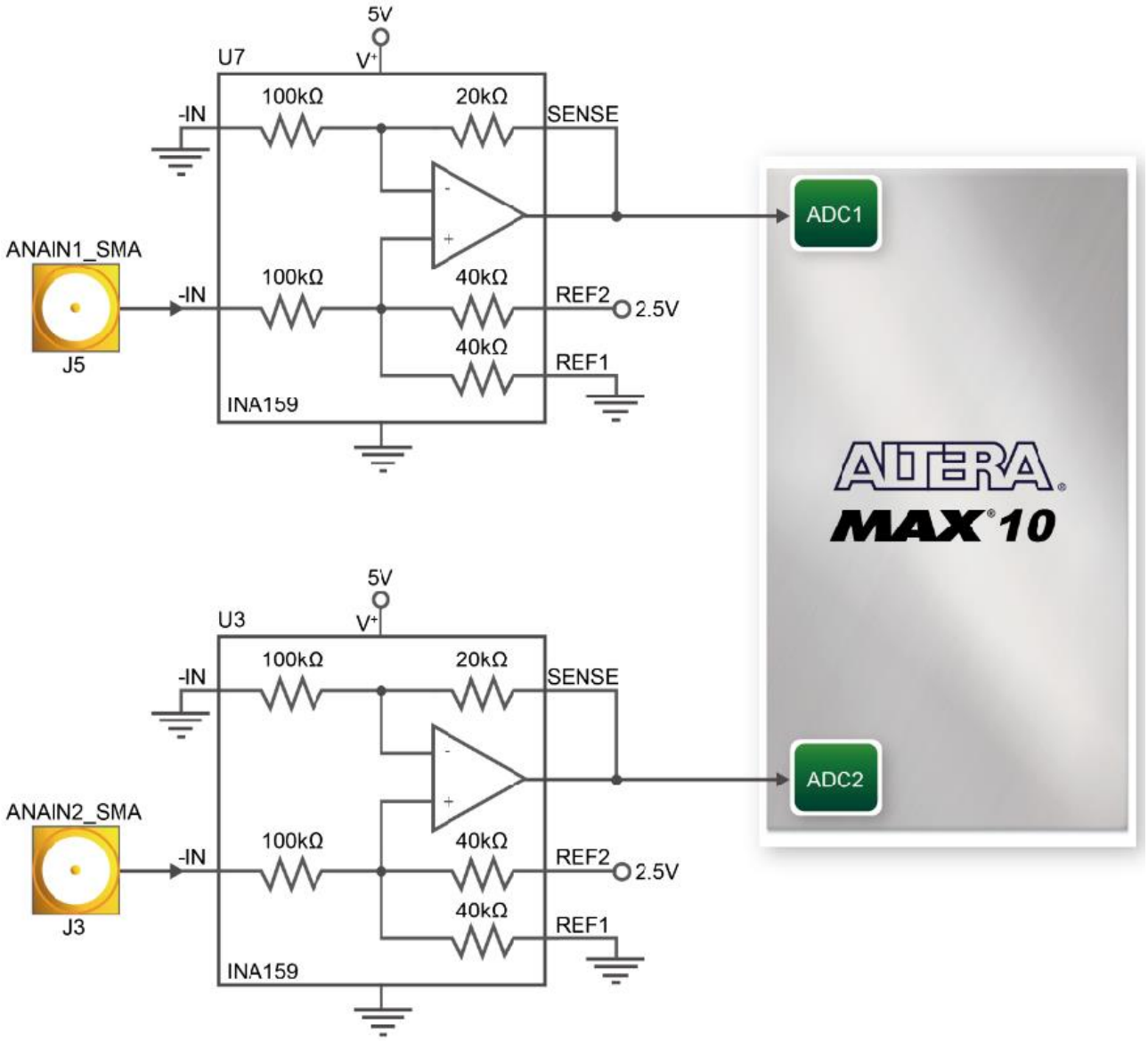


Figure 3-19 Connection of SMA connectors to the FPGA

3.4.6 DDR3 Memory

The board supports 512MB of DDR3 SDRAM comprising of one 16 bit DDR3 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 3-11** shows the DDR3 interface pin assignments.

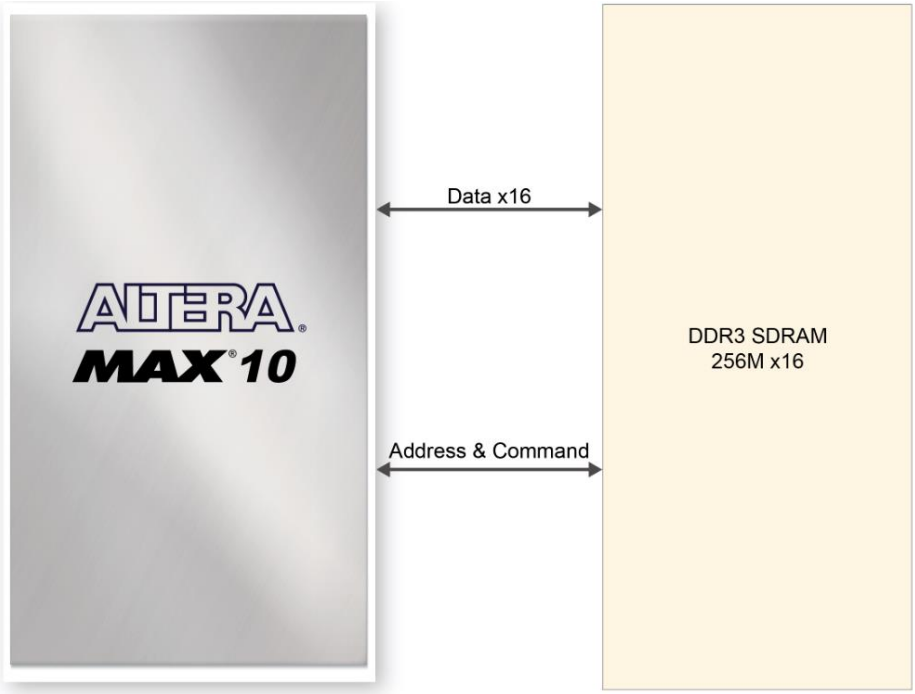


Figure 3-20 Connections between the DDR3 and FPGA

Table 3-11 Pin Assignment of FPGA DDR3 Memory

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
DDR3_A[0]	PIN_E21	DDR3 Address[0]	SSTL-15 Class I
DDR3_A[1]	PIN_V20	DDR3 Address[1]	SSTL-15 Class I
DDR3_A[2]	PIN_V21	DDR3 Address[2]	SSTL-15 Class I
DDR3_A[3]	PIN_C20	DDR3 Address[3]	SSTL-15 Class I
DDR3_A[4]	PIN_Y21	DDR3 Address[4]	SSTL-15 Class I
DDR3_A[5]	PIN_J14	DDR3 Address[5]	SSTL-15 Class I
DDR3_A[6]	PIN_V18	DDR3 Address[6]	SSTL-15 Class I
DDR3_A[7]	PIN_U20	DDR3 Address[7]	SSTL-15 Class I
DDR3_A[8]	PIN_Y20	DDR3 Address[8]	SSTL-15 Class I
DDR3_A[9]	PIN_W22	DDR3 Address[9]	SSTL-15 Class I
DDR3_A[10]	PIN_C22	DDR3 Address[10]	SSTL-15 Class I
DDR3_A[11]	PIN_Y22	DDR3 Address[11]	SSTL-15 Class I
DDR3_A[12]	PIN_N18	DDR3 Address[12]	SSTL-15 Class I
DDR3_A[13]	PIN_V22	DDR3 Address[13]	SSTL-15 Class I
DDR3_A[14]	PIN_W20	DDR3 Address[14]	SSTL-15 Class I
DDR3_BA[0]	PIN_D19	DDR3 Bank Address[0]	SSTL-15 Class I
DDR3_BA[1]	PIN_W19	DDR3 Bank Address[1]	SSTL-15 Class I
DDR3_BA[2]	PIN_F19	DDR3 Bank Address[2]	SSTL-15 Class I
DDR3_CAS_n	PIN_E20	DDR3 Column Address Strobe	SSTL-15 Class I
DDR3_CKE	PIN_B22	Clock Enable pin for DDR3	SSTL-15 Class I
DDR3_CK_n	PIN_E18	Clock n for DDR3	DIFFERENTIAL 1.5-V SSTL Class I

DDR3_CK_p	PIN_D18	Clock p for DDR3	Differential 1.5-V SSTL Class I
DDR3_CS_n	PIN_F22	DDR3 Chip Select	SSTL-15 Class I
DDR3_DM[0]	PIN_N19	DDR3 Data Mask[0]	SSTL-15 Class I
DDR3_DM[1]	PIN_J15	DDR3 Data Mask[1]	SSTL-15 Class I
DDR3_DQ[0]	PIN_L20	DDR3 Data[0]	SSTL-15 Class I
DDR3_DQ[1]	PIN_L19	DDR3 Data[1]	SSTL-15 Class I
DDR3_DQ[2]	PIN_L18	DDR3 Data[2]	SSTL-15 Class I
DDR3_DQ[3]	PIN_M15	DDR3 Data[3]	SSTL-15 Class I
DDR3_DQ[4]	PIN_M18	DDR3 Data[4]	SSTL-15 Class I
DDR3_DQ[5]	PIN_M14	DDR3 Data[5]	SSTL-15 Class I
DDR3_DQ[6]	PIN_M20	DDR3 Data[6]	SSTL-15 Class I
DDR3_DQ[7]	PIN_N20	DDR3 Data[7]	SSTL-15 Class I
DDR3_DQ[8]	PIN_K19	DDR3 Data[8]	SSTL-15 Class I
DDR3_DQ[9]	PIN_K18	DDR3 Data[9]	SSTL-15 Class I
DDR3_DQ[10]	PIN_J18	DDR3 Data[10]	SSTL-15 Class I
DDR3_DQ[11]	PIN_K20	DDR3 Data[11]	SSTL-15 Class I
DDR3_DQ[12]	PIN_H18	DDR3 Data[12]	SSTL-15 Class I
DDR3_DQ[13]	PIN_J20	DDR3 Data[13]	SSTL-15 Class I
DDR3_DQ[14]	PIN_H20	DDR3 Data[14]	SSTL-15 Class I
DDR3_DQ[15]	PIN_H19	DDR3 Data[15]	SSTL-15 Class I
DDR3_DQS_n[0]	PIN_L15	DDR3 Data Strobe n[0]	Differential 1.5-V SSTL Class I
DDR3_DQS_n[1]	PIN_K15	DDR3 Data Strobe n[1]	Differential 1.5-V SSTL Class I
DDR3_DQS_p[0]	PIN_L14	DDR3 Data Strobe p[0]	Differential 1.5-V SSTL Class I
DDR3_DQS_p[1]	PIN_K14	DDR3 Data Strobe p[1]	Differential 1.5-V SSTL Class I
DDR3_ODT	PIN_G22	DDR3 On-die Termination	SSTL-15 Class I
DDR3_RAS_n	PIN_D22	DDR3 Row Address Strobe	SSTL-15 Class I
DDR3_RESET_n	PIN_U19	DDR3 Reset	SSTL-15 Class I
DDR3_WE_n	PIN_E22	DDR3 Write Enable	SSTL-15 Class I

3.4.7 QSPI Flash

The DECA 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 3-12** shows the QSPI Flash pin assignments

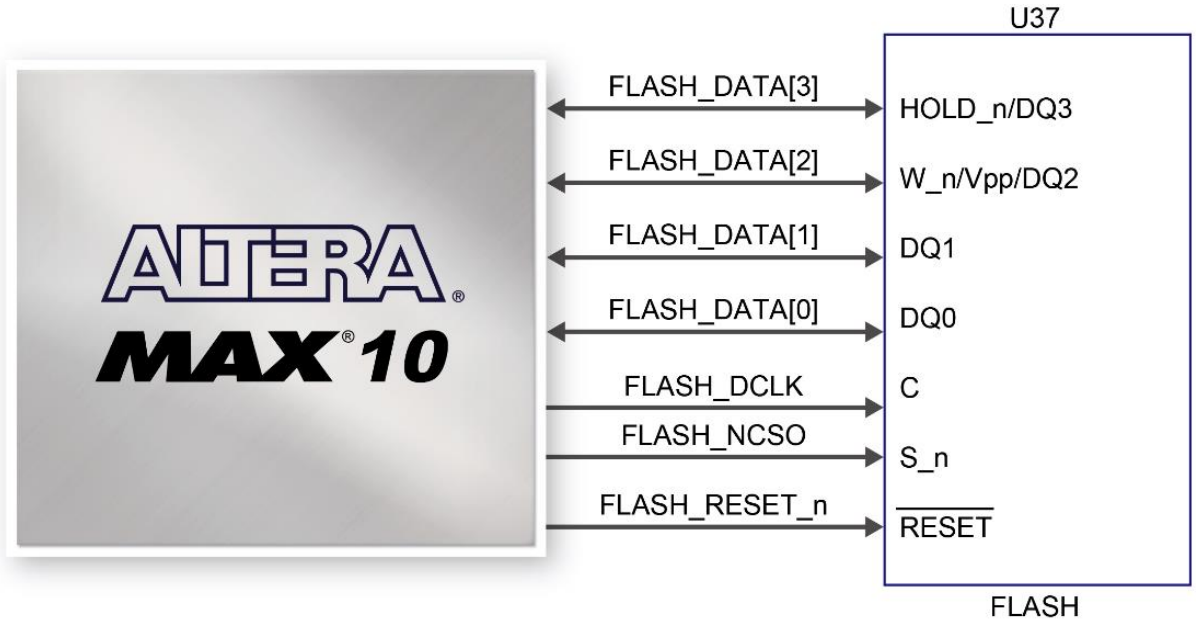


Figure 3-21 Connections between MAX 10 FPGA and QSPI Flash

Table 3-12 Pin Assignment of QSPI Flash

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
FLASH_DATA[0]	PIN_P12	FLASH Data[0]	3.3V
FLASH_DATA[1]	PIN_V4	FLASH Data[1]	3.3V
FLASH_DATA[2]	PIN_V5	FLASH Data[2]	3.3V
FLASH_DATA[3]	PIN_P10	FLASH Data[3]	3.3V
FLASH_DCLK	PIN_R12	FLASH Data Clock	3.3V
FLASH_NCSO	PIN_R10	FLASH Chip Enable	3.3V
FLASH_RESET_n	PIN_W10	FLASH Chip Reset	3.3V

3.4.8 Ethernet

The board supports 10/100 Mbps Ethernet transfer by an external Texas Instruments DP83620 PHY chip. The DP838620 also provides flexibility by supporting both MII and RMII 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 3-13**.

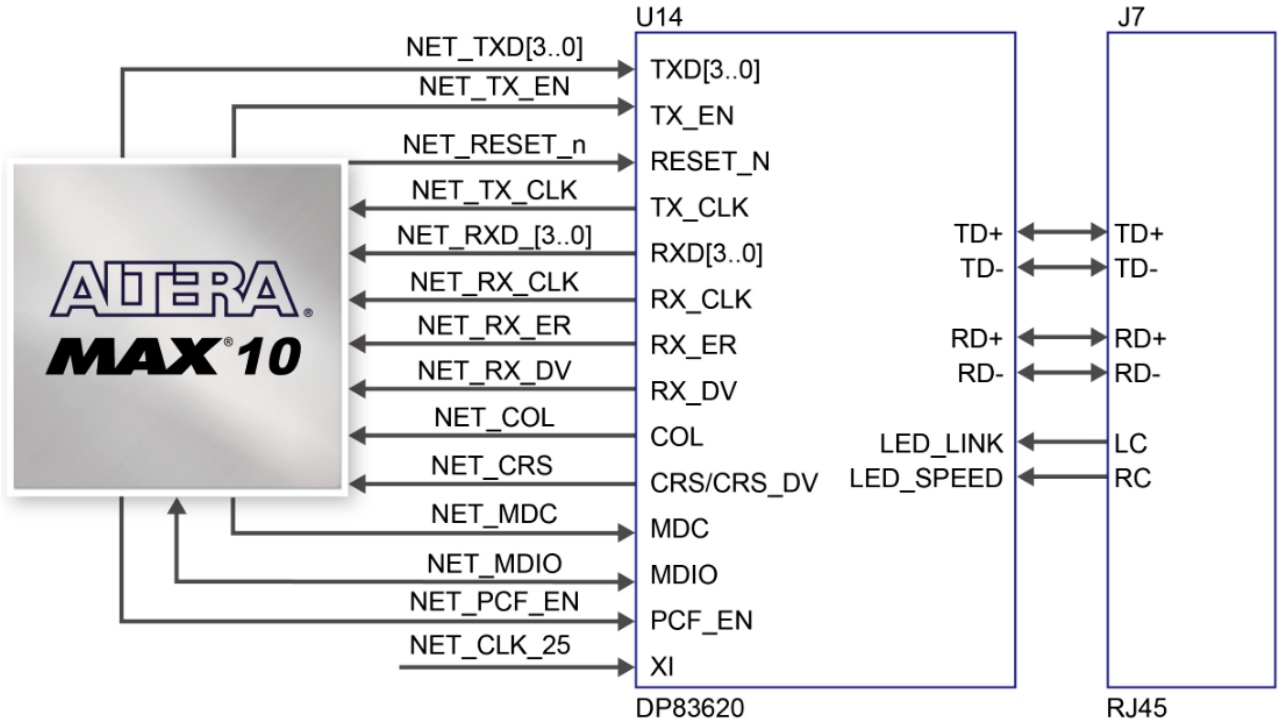


Figure 3-22 Connections between the MAX 10 FPGA and Ethernet PHY

Table 3-13 Pin Assignment of Ethernet PHY

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
NET_TX_EN	PIN_P3	GMII and MII transmit enable	2.5V
NET_TX_CLK	PIN_T5	MII transmit clock	2.5V
NET_TXD[0]	PIN_U2	MII transmit data[0]	2.5V
NET_TXD[1]	PIN_W1	MII transmit data[1]	2.5V
NET_TXD[2]	PIN_N9	MII transmit data[2]	2.5V
NET_TXD[3]]	PIN_W2	MII transmit data[3]	2.5V
NET_RX_DV	PIN_P4	GMII and MII receive data valid	2.5V
NET_RX_ER	PIN_V1	GMII and MII receive data valid	2.5V
NET_RXD[0]	PIN_U5	GMII and MII receive data[0]	2.5V
NET_RXD[1]	PIN_U4	GMII and MII receive data[1]	2.5V
NET_RXD[2]	PIN_R7	GMII and MII receive data[2]	2.5V
NET_RXD[3]	PIN_P8	GMII and MII receive data[3]	2.5V
NET_RX_CLK	PIN_T6	GMII and MII receive clock	2.5V
NET_RESET_n	PIN_R3	Hardware Reset Signal	2.5V
NET_MDIO	PIN_N8	Management Data	2.5V
NET_MDC	PIN_R5	Management Data Clock Reference	2.5V
NET_COL	PIN_R4	Interrupt Open Drain Output	2.5V
NET_CRS	PIN_P5	GMII Transmit Clock	2.5V
NET_PCF_EN	PIN_V9	PHY control frame enable	3.3V

3.4.9 HDMI TX

The development board provides High Performance HDMI Transmitter via the Analog Devices ADV7513 which incorporates HDMI v1.4 features, including 3D video support, and 165 MHz supports all video formats up to 1080p and UXGA. The ADV7513 is controlled via a serial I2C bus interface, which is connected to pins on the MAX 10 FPGA. A schematic diagram of the HDMI TX circuitry is shown in **Figure 3-23**. Detailed information on using the ADV7513 HDMI TX is available on the manufacturer's website, or under the Datasheets\HDMI folder on the Kit System CD.

Table 3-14 lists the HDMI Interface pin assignments and signal names relative to the MAX 10 device.

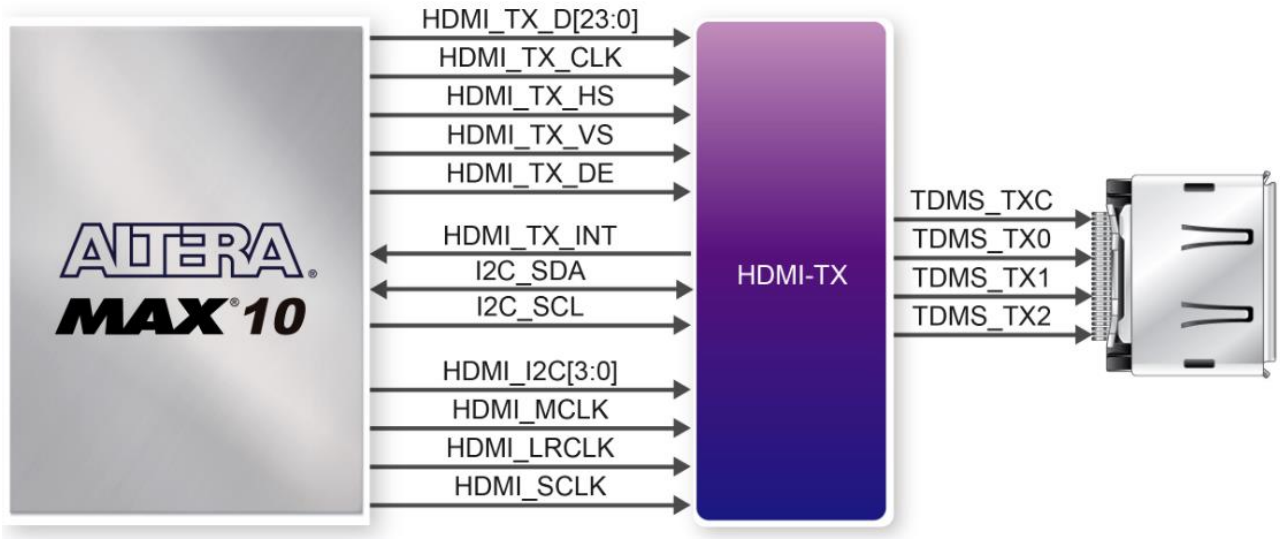


Figure 3-23 Connection between the MAX 10 FPGA and HDMI transmitter

Table 3-14 Pin Assignment of HDMI TX

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
HDMI_TX_D0	PIN_C18	Video Data bus	1.8V
HDMI_TX_D1	PIN_D17	Video Data bus	1.8V
HDMI_TX_D2	PIN_C17	Video Data bus	1.8V
HDMI_TX_D3	PIN_C19	Video Data bus	1.8V
HDMI_TX_D4	PIN_D14	Video Data bus	1.8V
HDMI_TX_D5	PIN_B19	Video Data bus	1.8V
HDMI_TX_D6	PIN_D13	Video Data bus	1.8V
HDMI_TX_D7	PIN_A19	Video Data bus	1.8V
HDMI_TX_D8	PIN_C14	Video Data bus	1.8V
HDMI_TX_D9	PIN_A17	Video Data bus	1.8V
HDMI_TX_D10	PIN_B16	Video Data bus	1.8V
HDMI_TX_D11	PIN_C15	Video Data bus	1.8V

HDMI_TX_D12	PIN_A14	Video Data bus	1.8V
HDMI_TX_D13	PIN_A15	Video Data bus	1.8V
HDMI_TX_D14	PIN_A12	Video Data bus	1.8V
HDMI_TX_D15	PIN_A16	Video Data bus	1.8V
HDMI_TX_D16	PIN_A13	Video Data bus	1.8V
HDMI_TX_D17	PIN_C16	Video Data bus	1.8V
HDMI_TX_D18	PIN_C12	Video Data bus	1.8V
HDMI_TX_D19	PIN_B17	Video Data bus	1.8V
HDMI_TX_D20	PIN_B12	Video Data bus	1.8V
HDMI_TX_D21	PIN_B14	Video Data bus	1.8V
HDMI_TX_D22	PIN_A18	Video Data bus	1.8V
HDMI_TX_D23	PIN_C13	Video Data bus	1.8V
HDMI_TX_CLK	PIN_A20	Video Clock	1.8V
HDMI_TX_DE	PIN_C9	Data Enable Signal for Digital Video.	1.8V
HDMI_TX_HS	PIN_B11	Horizontal Synchronization	1.8V
HDMI_TX_VS	PIN_C11	Vertical Synchronization	1.8V
HDMI_TX_INT	PIN_B10	Interrupt Signal	1.8V
HDMI_I2C_SCL	PIN_C10	I2C Clock	1.8V
HDMI_I2C_SDA	PIN_B15	I2C Data	1.8V
HDMI_MCLK	PIN_A7	Audio Reference Clock Input	1.8V
HDMI_LRCLK	PIN_A10	Left/Right Channel Signal Input	1.8V
HDMI_SCLK	PIN_D12	I2S Audio Clock Input	1.8V
HDMI_I2S0	PIN_A9	I2C Channel 0 Audio Data Input	1.8V
HDMI_I2S1	PIN_A11	I2C Channel 1 Audio Data Input	1.8V
HDMI_I2S2	PIN_A8	I2C Channel 2 Audio Data Input	1.8V
HDMI_I2S3	PIN_B8	I2C Channel 3 Audio Data Input	1.8V

3.4.10 USB 2.0 OTG PHY

The board provides USB interfaces using the Texas Instruments TUSB1210 transceiver chip. The TUSB1210 is designed to interface with a USB controller via a UTMI+ Low Pin Interface (ULPI). It supports all USB2.0 data rates (High-Speed 480Mbps, Full-Speed 12 Mbps and Low-Speed 1.5Mbps), and is compliant to both Host and Peripheral modes. When operating in Host mode, the interface will supply the power to the device through the Mini-USB interface. **Figure 3-24** shows the schematic diagram of the USB circuitry; the pin assignments for the associated interface are listed in **Table 3-15**.

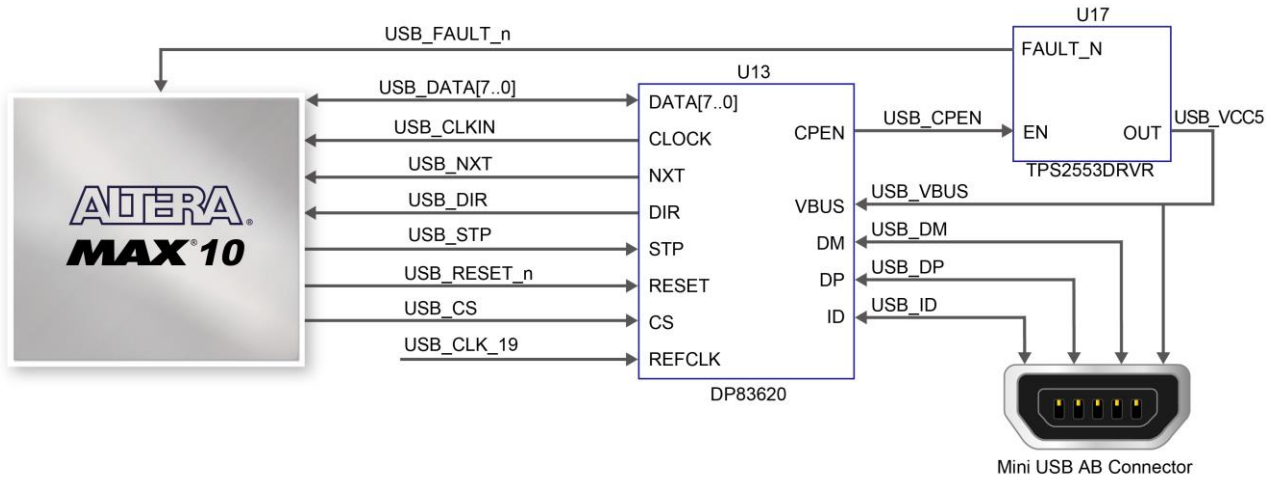


Figure 3-24 Connections between the MAX 10 FPGA and USB OTG PHY

Table 3-15 Pin Assignment of USB OTG PHY

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
USB_DATA[0]	PIN_E12	ULPI DATA input/output signal 0	1.8V
USB_DATA[1]	PIN_E13	ULPI DATA input/output signal 1	1.8V
USB_DATA[2]	PIN_H13	ULPI DATA input/output signal 2	1.8V
USB_DATA[3]	PIN_E14	ULPI DATA input/output signal 3	1.8V
USB_DATA[4]	PIN_H14	ULPI DATA input/output signal 4	1.8V
USB_DATA[5]	PIN_D15	ULPI DATA input/output signal 5	1.8V
USB_DATA[6]	PIN_E15	ULPI DATA input/output signal 6	1.8V
USB_DATA[7]	PIN_F15	ULPI DATA input/output signal 7	1.8V
USB_NXT	PIN_H12	ULPI NXT output signal	1.8V
USB_DIR	PIN_J13	ULPI DIR output signal	1.8V
USB_STP	PIN_J12	ULPI STP input signal	1.8V
USB_CS	PIN_J11	Active-high chip select pin	1.8V
USB_RESET_n	PIN_E16	Reset pin used to reset all digital registers	1.8V
USB_CLKIN	PIN_H11	ULPI 60 MHz output clock	1.2V
USB_FAULT_n	PIN_D8	Fault input from USB power switch	1.2V

3.4.11 MIPI Interface

The DECA provides a Mobile Industry Processor Interface (MIPI) connector, allows to implement the camera module applications. **Figure 3-25** shows the connection of MIPI camera module to the MAX 10 FPGA. With a resistor networks using passive components, the transmitted signals from camera/sensor can be successfully received by MAX 10 FPGA. The pin assignment associated to this MIPI interface is shown in **Table 3-16.**

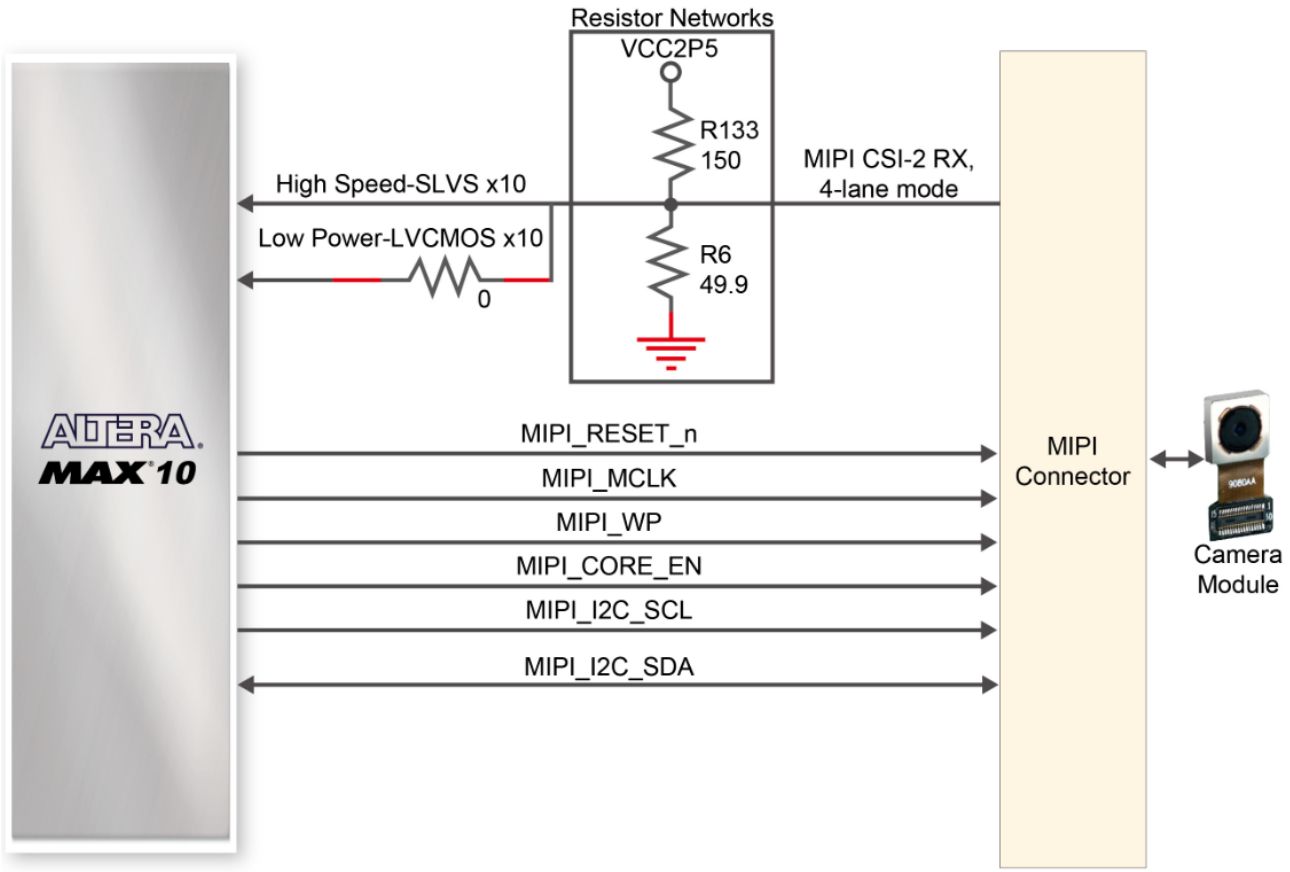


Figure 3-25 Connections between the MAX 10 FPGA and MIPI camera module

Table 3-16 Pin Assignment of MIPI interface

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MIPI_MD_p[0]	PIN_R2	HS Differential MIPI Serial 0 Data Lane(positive)	2.5V
MIPI_MD_n[0]	PIN_R1	HS Differential MIPI Serial 0 Data Lane(negative)	2.5V
MIPI_MD_p[1]	PIN_N1	HS Differential MIPI Serial 1 Data Lane(positive)	2.5V
MIPI_MD_n[1]	PIN_P1	HS Differential MIPI Serial 1 Data Lane(negative)	2.5V
MIPI_MD_p[2]	PIN_T2	HS Differential MIPI Serial 2 Data Lane(negative)	2.5V
MIPI_MD_n[2]	PIN_T1	HS Differential MIPI Serial 2 Data Lane(negative)	2.5V
MIPI_MD_p[3]	PIN_N2	HS Differential MIPI Serial 3 Data Lane(negative)	2.5V
MIPI_MD_n[3]	PIN_N3	HS Differential MIPI Serial 3 Data Lane(negative)	2.5V
MIPI_MC_p	PIN_N5	HS Differential MIPI Serial Clock/Strobe (positive)	2.5V
MIPI_MC_n	PIN_N4	HS Differential MIPI Serial Clock/Strobe (negative)	2.5V
MIPI_LP_MD_p[0]	PIN_A4	LP single-ended MIPI Data Lane 0 signal dp0	1.2V
MIPI_LP_MD_n[0]	PIN_A3	LP single-ended MIPI Data Lane 0 signal dn0	1.2V
MIPI_LP_MD_p[1]	PIN_C3	LP single-ended MIPI Data Lane 1 signal dp1	1.2V
MIPI_LP_MD_n[1]	PIN_C2	LP single-ended MIPI Data Lane 1 signal dn1	1.2V
MIPI_LP_MD_p[2]	PIN_B1	LP single-ended MIPI Data Lane 2 signal dp2	1.2V
MIPI_LP_MD_n[2]	PIN_B2	LP single-ended MIPI Data Lane 2 signal dn2	1.2V

MIPI_LP_MD_p[3]	PIN_B3	LP single-ended MIPI Data Lane 3 signal dp3	1.2V
MIPI_LP_MD_n[3]	PIN_A2	LP single-ended MIPI Data Lane 3 signal dn3	1.2V
MIPI_LP_MC_p	PIN_E11	LP single-ended MIPI Clock signal cp	1.2V
MIPI_LP_MC_n	PIN_E10	LP single-ended MIPI Clock signal cn	1.2V
MIPI_RESET_n	PIN_T3	Master output Reset signal for MIPI camera/sensor	2.5V
MIPI_MCLK	PIN_U3	Master output Clock	2.5V
MIPI_WP	PIN_U1	Master output EEPROM Write Protect	2.5V
MIPI_CORE_EN	PIN_V3	Master output core Enable	2.5V
MIPI_I2C_SCL	PIN_M1	Master output I2C Clock	2.5V
MIPI_I2C_SDA	PIN_M2	I2C Data	2.5V

3.4.12 Proximity/Ambient Light Sensor

The DECA has a low-power, reflectance-based, infrared proximity and ambient light sensor (Si1143) with I2C digital interface and programmable-event interrupt output. The Si1143 is an active optical reflectance proximity detector and ambient light sensor whose operational state is controlled through registers accessible through the I2C interface. The host can command the Si1143 to initiate on-demand proximity detection or ambient light sensing. Figure 3-26 shows the connection of Si1143 to the MAX 10 FPGA. Table 3-17 lists the Proximity/Ambient Light Sensor pin assignments.

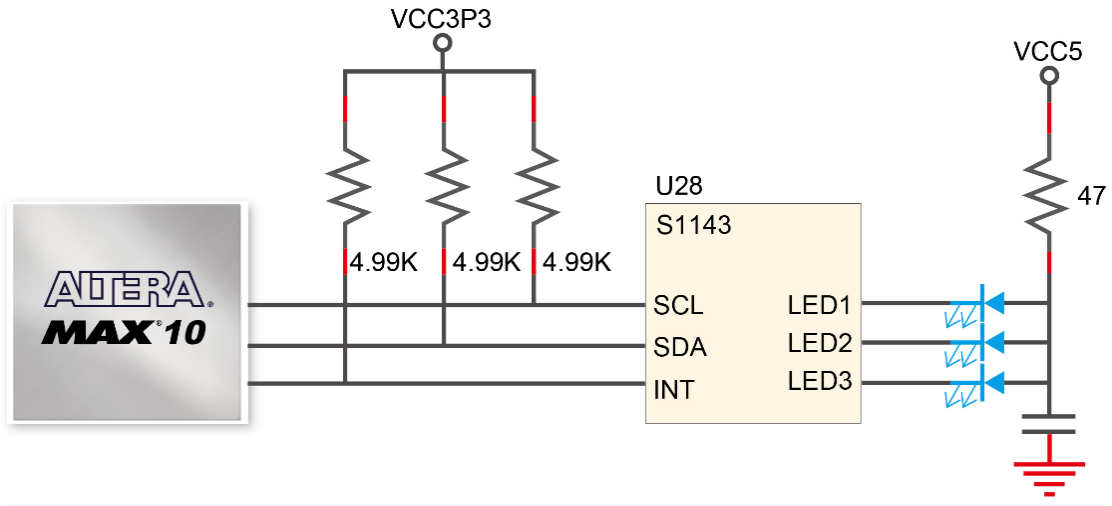


Figure 3-26 shows the connections between the MAX 10 FPGA and Proximity/Ambient Light Sensor

Table 3-17 Pin Assignment of Proximity/Ambient Light Sensor

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
LIGHT_I2C_SCL	PIN_Y8	I2C Clock for Si1143 Sensor	3.3V
LIGHT_I2C_SDA	PIN_AA8	I2C Data for Si1143 Sensor	3.3V

LIGHT_INT	PIN_AA9	Interrupt input from Si1143 Sensor	3.3V

3.4.13 Humidity and Temperature Sensor

The DECA 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 DECA so that user can make ambient temperature measurement without heat interference from the heat source of DECA. Figure 3-27 shows the connection of humidity and temperature sensor to MAX 10 FPGA. Table 3-18 lists the humidity and temperature sensor pin assignments.

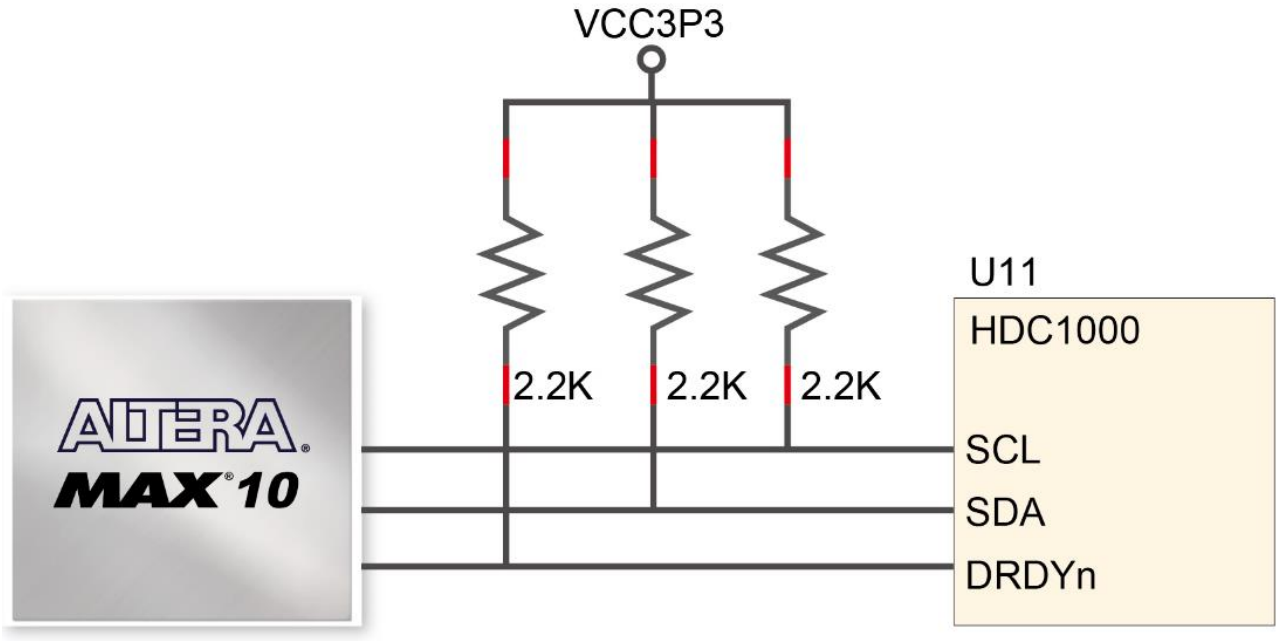


Figure 3-27 shows the connections between the MAX 10 FPGA and humidity and temperature sensor.

Table 3-18 Pin Assignment of Humidity and Temperature Sensor

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
RH_TEMP_I2C_SCL	PIN_Y10	I2C Clock for HDC1000 Sensor	3.3V
RH_TEMP_I2C_SDA	PIN_AA10	I2C Data for HDC1000 Sensor	3.3V
RH_TEMP_DRDY_n	PIN_AB9	Data ready input from HDC1000 Sensor	3.3V

3.4.14 Temperature Sensor

The DECA provides a low-power, high-resolution digital temperature sensor (LM71) with an SPI ans MICROWIRE compatible interface. The host can query the LM71 at any time to read temperature. The LM71 has 13-bit plus sign temperature resolution (0.03125°C per LSB) while operating over a temperature range of −40°C to +150°C. The LM71 temperature sensor is placed at near the heat source of DECA so that user can easily make the board temperature measurement. Figure 3-28 shows the connection of temperature sensor to MAX 10 FPGA. Table 3-19 lists

temperature sensor pin assignments.

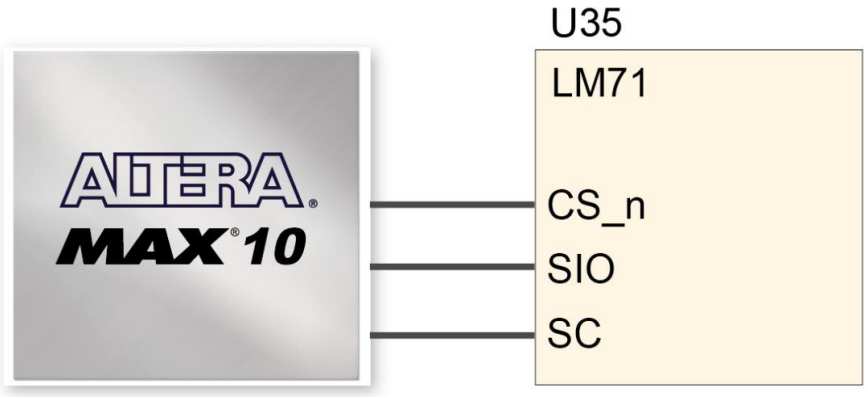


Figure 3-28 shows the connections between the MAX 10 FPGA and temperature sensor.

Table 3-19 Pin Assignment of Temperature Sensor

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
TEMP_SC	PIN_AA1	FPGA output clock to sensor	3.3V
TEMP_SIO	PIN_Y2	Bi-direction data line	3.3V
TEMP_CS_n	PIN_AB4	FPGA output chip secetc	3.3V

3.4.15 Accelerometer Sensor

The board comes with a digital accelerometer sensor (LIS2DH12), commonly known as G-sensor. This G-sensor is a small, thin, ultra-low-power assumption 3-axis accelerometer with digital I2C/SPI serial interface standard output. The LIS2DH12 has user-selectable full scales of +-2g/+-4g/+-8g/+-16g and it is capable of measuring accelerations with output data rates from 1 Hz to 5.3 kHz. **Figure 3-29** shows the connection of accelerometer sensor to MAX 10 FPGA. **Table 3-20** lists accelerometer sensor pin assignments.

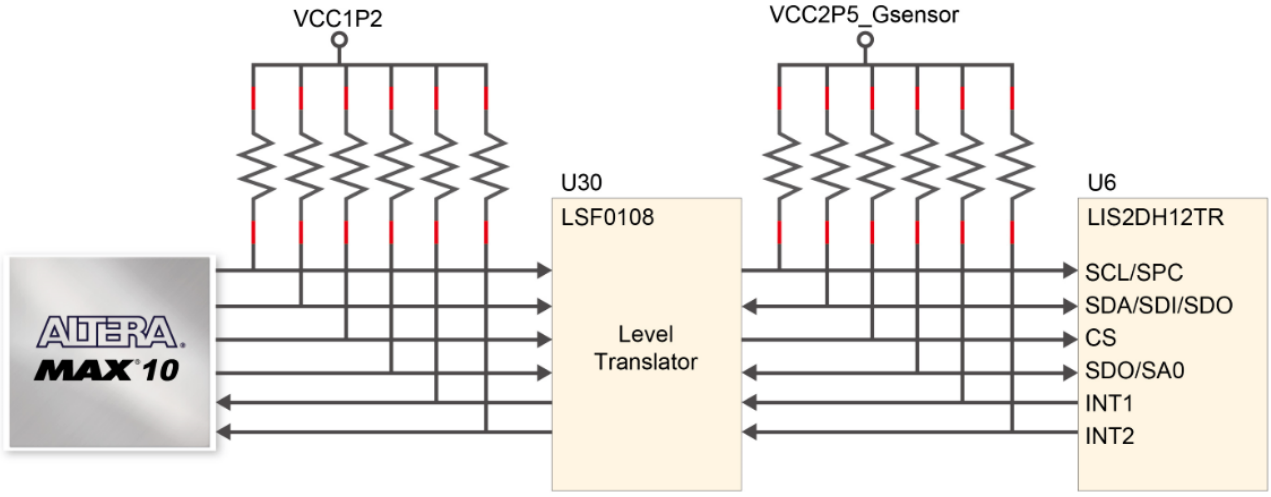


Figure 3-29 shows the connections between the MAX 10 FPGA and accelerometer sensor.

Table 3-20 Pin Assignment of Accelerometer Sensor

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
G_SENSOR_SDI	PIN_C6	I2C serial data/SPI serial data input/3-wire interface serial dta output	1.2V
G_SENSOR_SDO	PIN_D5	SPI serial data output/I2C less significant bit of the device address	1.2V
G_SENSOR_CS_n	PIN_E9	SPI enable, I2C/SPI mode selection: 1: SPI idle mode/I2C communication enabled, 0: SPI communication mode/I2C disabled	1.2V
G_SENSOR_SCLK	PIN_B5	I2C serial clock/SPI serial port clock	1.2V
G_SENSOR_INT1	PIN_E8	Interrupt pin 1	1.2V
G_SENSOR_INT2	PIN_D7	Interrupt pin 2	1.2V

3.4.16 Micro SD Card Socket

The board supports Micro SD card interface with x4 data lines. Through the load switches (U25/U26) and voltage-translation transceiver (U22), user can select 1.8V or 3.3V VCCIO power of Micro SD card to implement external storage application. **Figure 3-30** shows signals connected between the MAX 10 FPGA and Micro SD card socket.

Table 3-21 lists the pin assignment of Micro SD card socket to the MAX 10 FPGA.

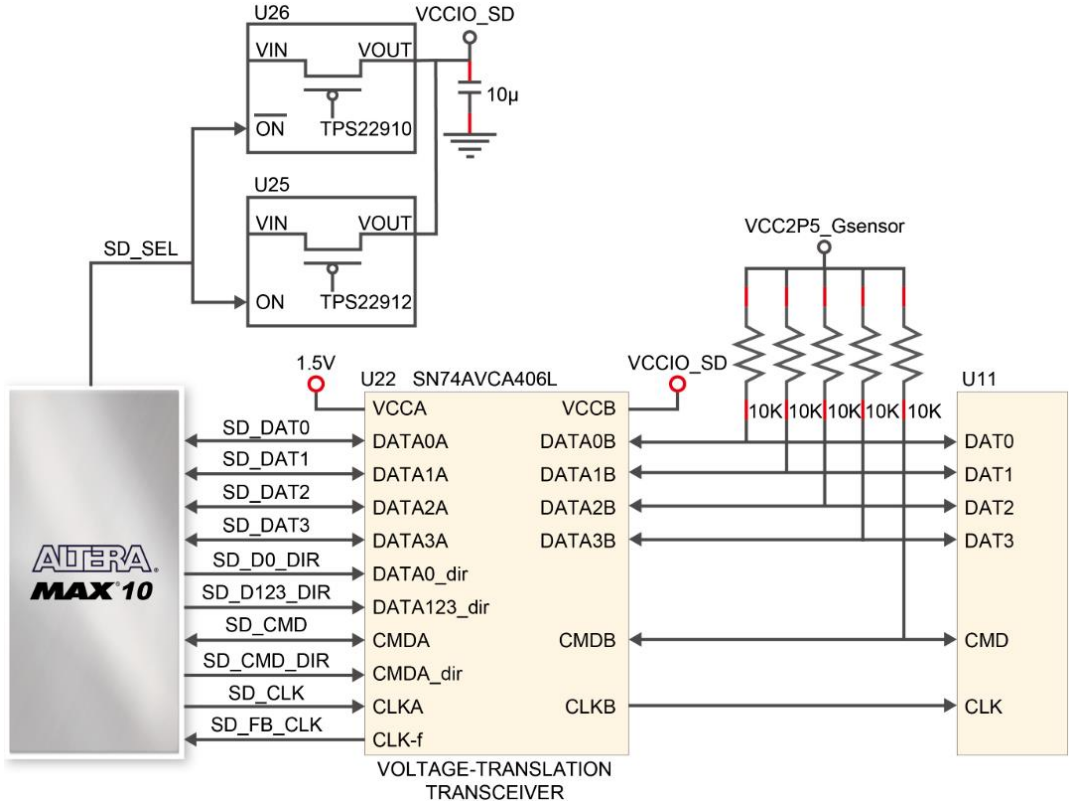


Figure 3-30 Connections between the MAX 10 FPGA and SD card socket

Table 3-21 Pin Assignment of Micro SD Card Socket

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SD_CLK	PIN_T20	FPGA output SD Clock	1.5V
SD_FB_CLK	PIN_R22	Clock feedback to FPGA for resynchronizing data	1.5V
SD_CMD	PIN_T21	Command bit connected to FPGA	1.5V
SD_CMD_DIR	PIN_U22	Direction control for command bit (CMDA/CMDB)	1.5V
SD_D0_DIR	PIN_T22	Direction control for DAT0A/DAT0B	1.5V
SD_D123_DIR	PIN_U21	Direction control for DAT1A/B, DAT2A/B, and DAT3A/B	1.5V
SD_SEL	PIN_P13	FPGA output SD card VCCIO select	3.3V
SD_DAT[0]	PIN_R18	Data bit 0 connected to FPGA	1.5V
SD_DAT[1]	PIN_T18	Data bit 1 connected to FPGA	1.5V
SD_DAT[2]	PIN_T19	Data bit 2 connected to FPGA	1.5V
SD_DAT[3]	PIN_R20	Data bit 3 connected to FPGA	1.5V

3.4.17 DECA Power

The DECA is powered by Altera's Enpirion power solution which provides high-efficiency power

management for FPGAs and SoCs. **Figure 3-31** shows the positions of Altera Enpirion regulators on DECA. **Figure 3-32** is the power tree of DECA.

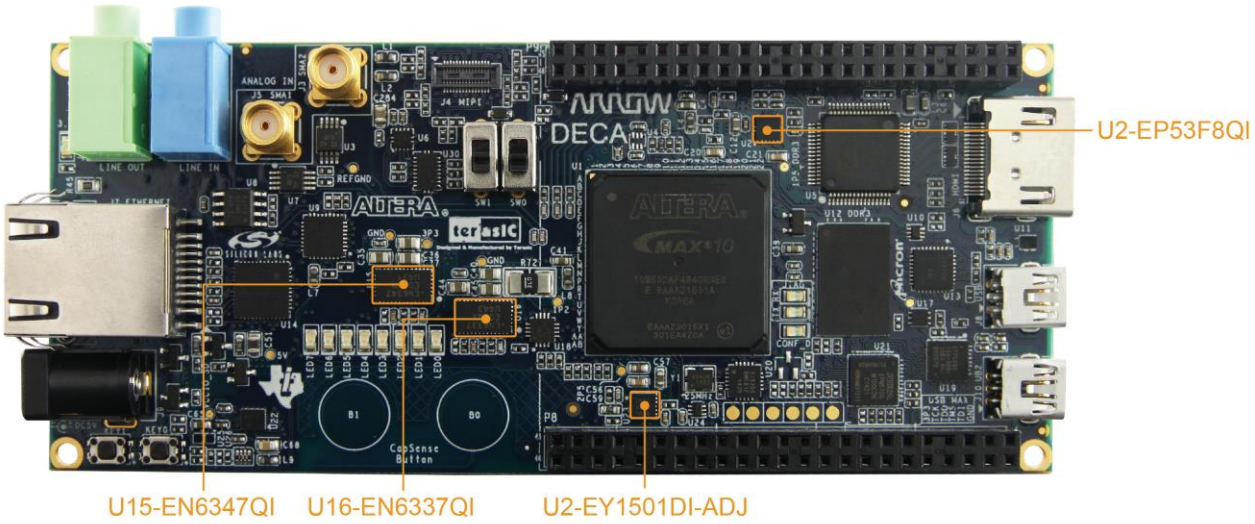


Figure 3-31 Positions of Altera Enpirion regulators

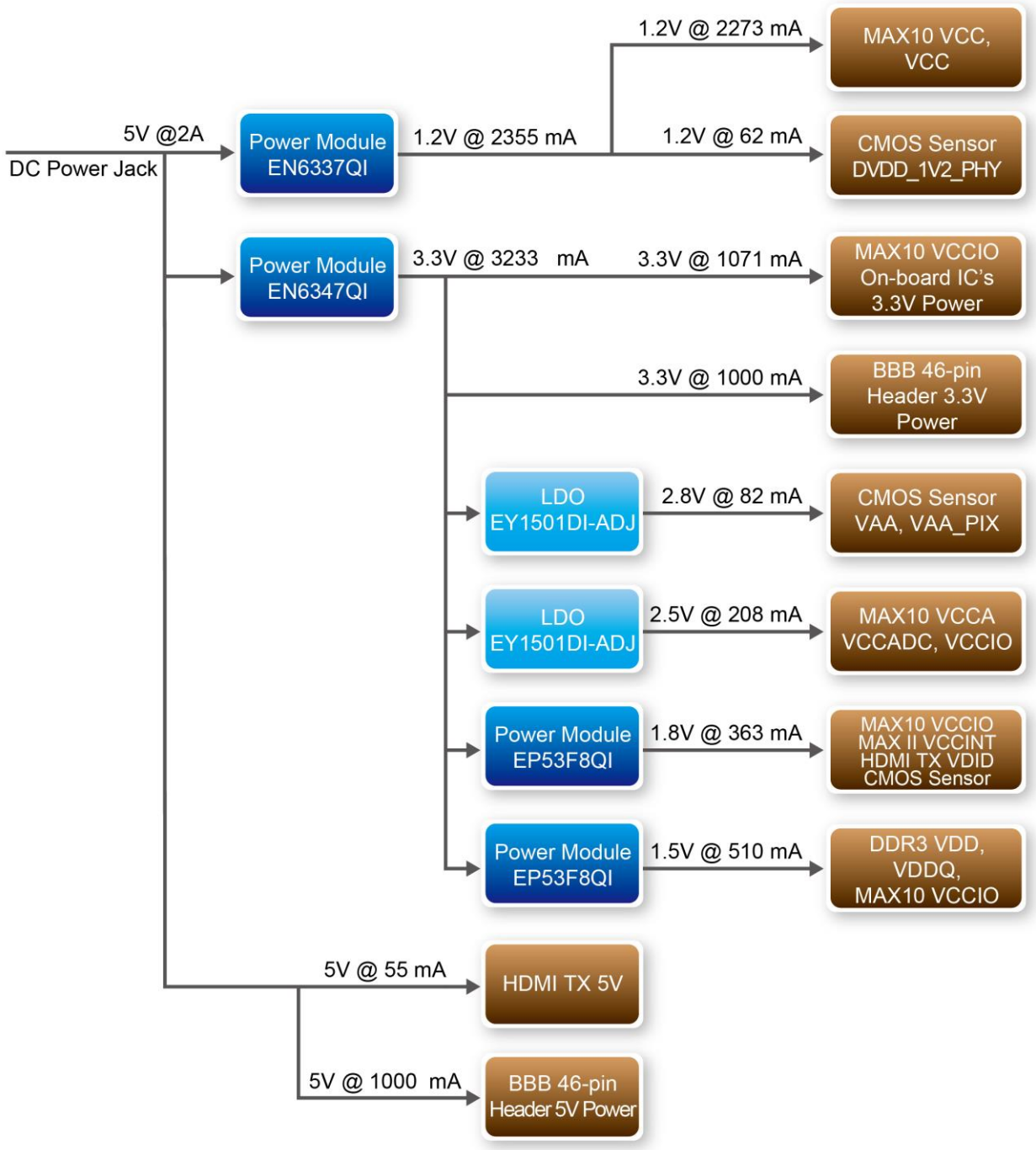


Figure 3-32 Power tree of DECA

Chapter 4

DECA System Builder

Need to create a custome design project but not sure where to start? We have created a DECA System Builder that can help you get started with your own design project in literally minutes.

4.1 Introduction

The DECA System Builder is a Windows-based utility. It is designed to help users create a Quartus II project for DECA 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 DECA System Builder can also prevent occurrence of 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 direction is incorrect or forgotten.
- Performance degradation due to improper pin assignment.

4.2 General Design Flow

We now introduce the design flow of building a Quartus II project for DECA using the DECA System Builder. The chart illustrated in **Figure 4-1.**gives you an overview what are the steps needed from launching the System Builder to Configuring the FPGA. The left-hand side of the chart can be done within minutes. All you need to do is to put in your design requirements and the DECA 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. You are free to modify according to your own project requirement. After all that is required is completed, you can then download the .sof file to the developmenet board via JTAG interface using the Qaurtus II programmer.

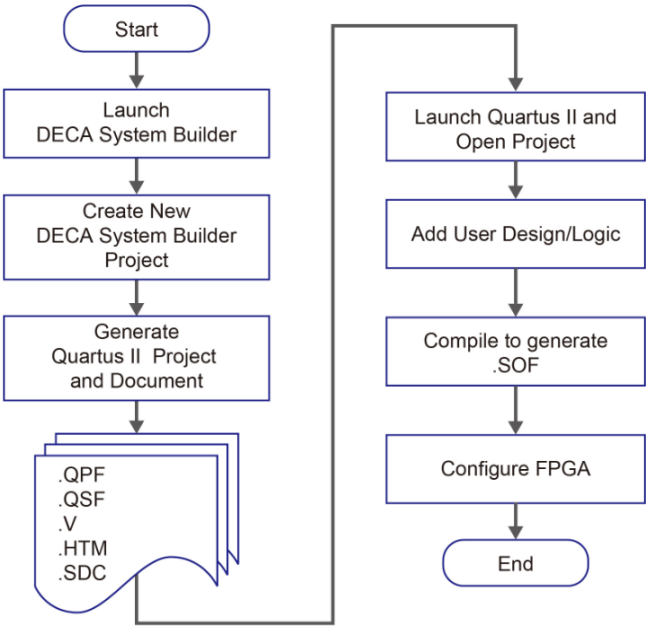


Figure 4-1 Design flow of building a project from the beginning to the end

4.3 Using DECA System Builder

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

■ Install and Launch the DECA System Builder

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

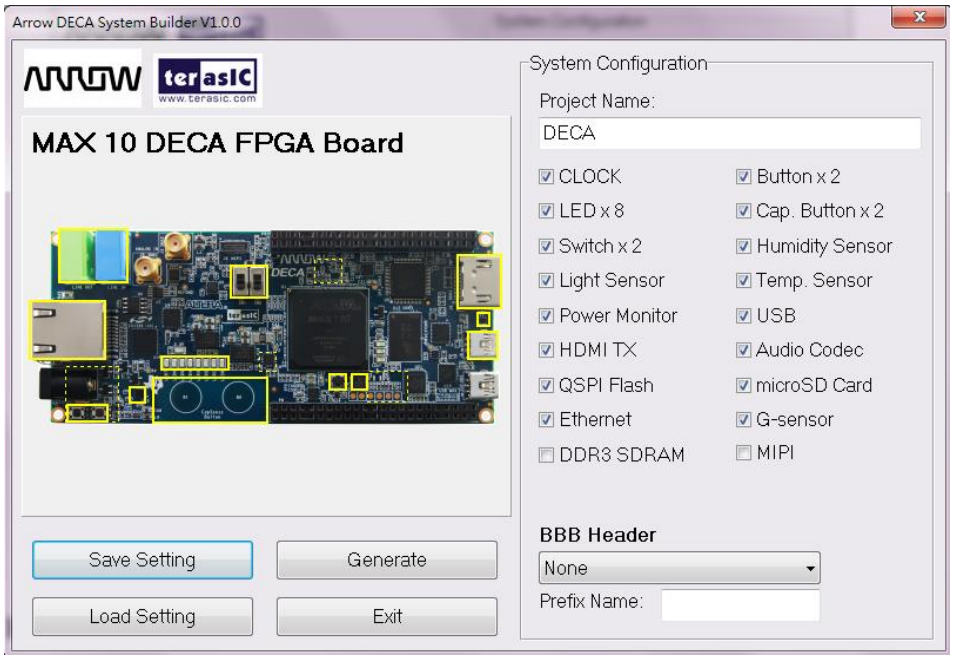


Figure 4-2 The GUI of DECA System Builder

■ Enter Project Name

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

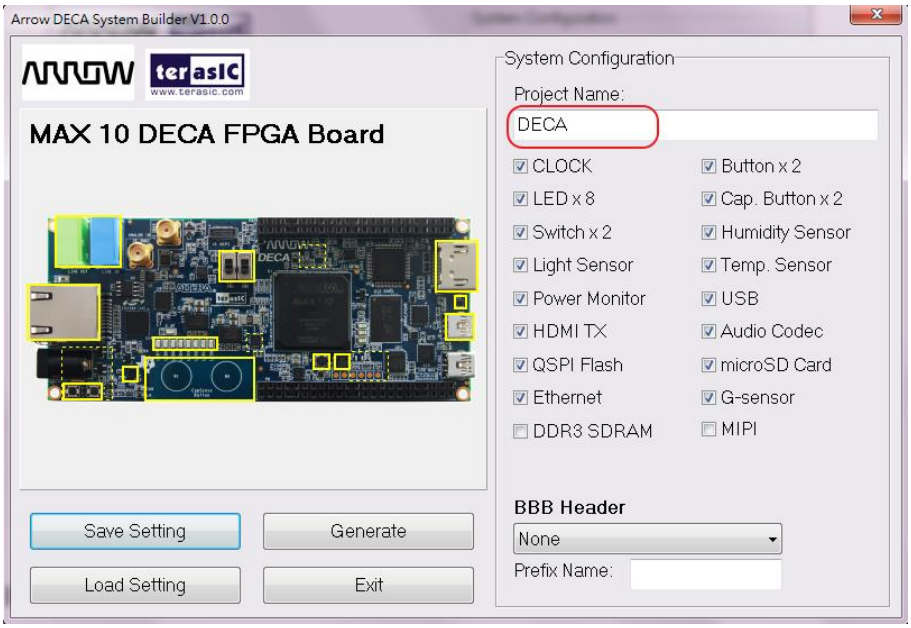


Figure 4-3 Enter the project name

■ System Configuration

Users are given the flexibility in the System Configuration to include their choice of components in the project, as shown in **Figure 4-4**. Each component onboard is listed and users can enable or disable one or more components at will. If a component is enabled, the DECA System Builder will automatically generate its associated pin assignment, including the pin name, pin location, pin direction, and I/O standard.

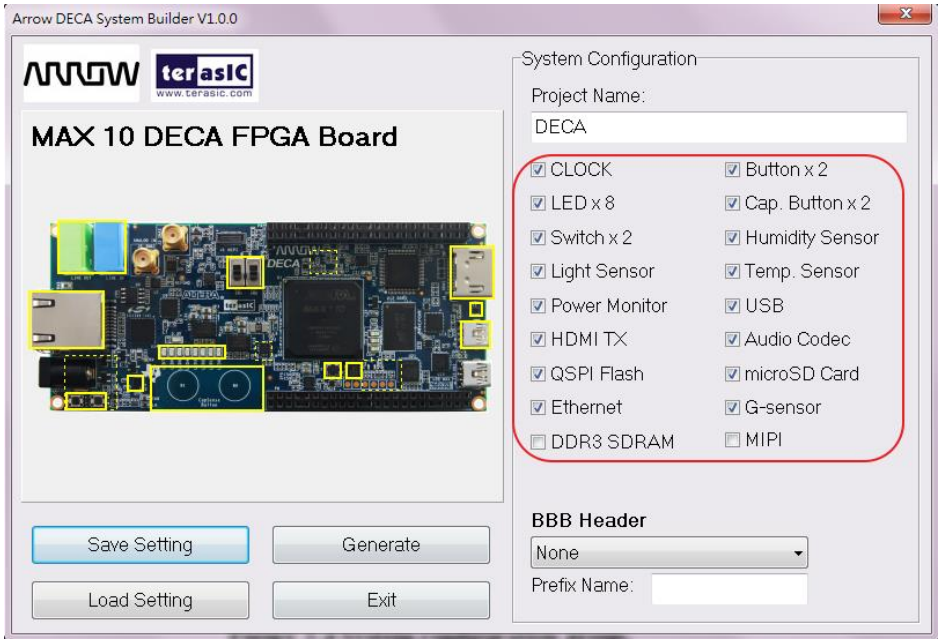


Figure 4-4 System configuration group

■ BBB Header

If user connect BBB daughter card to the BBB header on DECA, the DECA System Builder can selection the desired pinout name as shown in **Figure 4-5**. There are two kind of pinout name. One is the default and another is MODE 7. In default pinout, the pinout name is the same as DECA schematic. If MODE 7, the pinout name is the same as the MODE 7 pinout name defined in BBB product. If a prefer BBB pinout option is selected, the DECA System Builder will automatically generate its associated pin assignment, including the pin name, pin location, pin direction, and I/O standard.

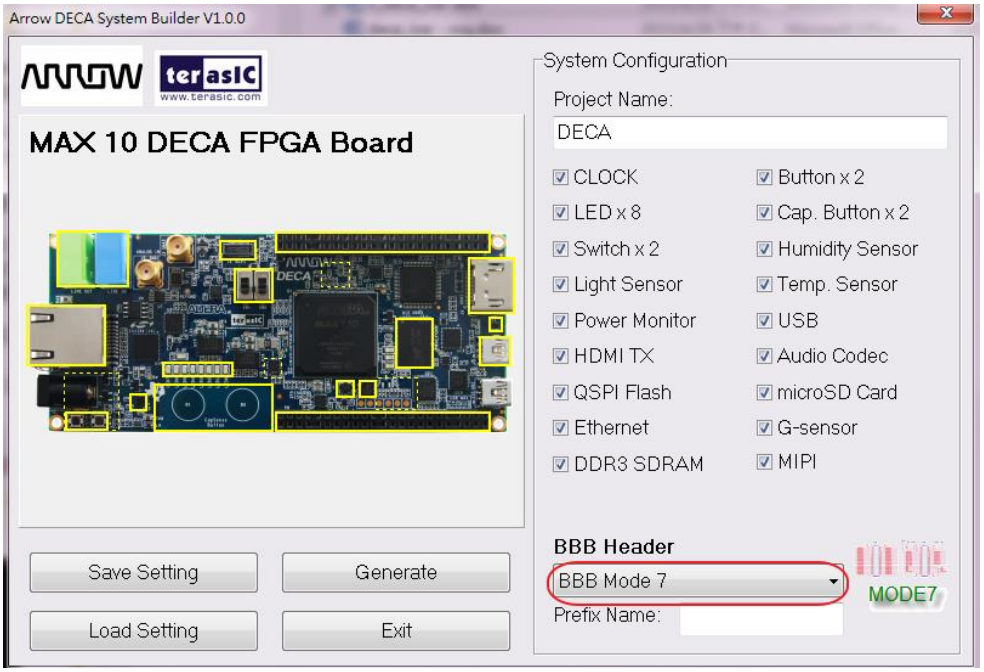


Figure 4-5 GPIO expansion group

The “Prefix Name” is an optional feature that denotes the pin name of the daughter card assigned in your design. Users may leave this field blank.

■ Project Setting Management

The DECA System Builder also provides the option to load a setting or save user’s current board configuration in .cfg file, as shown in **Figure 4-6**.

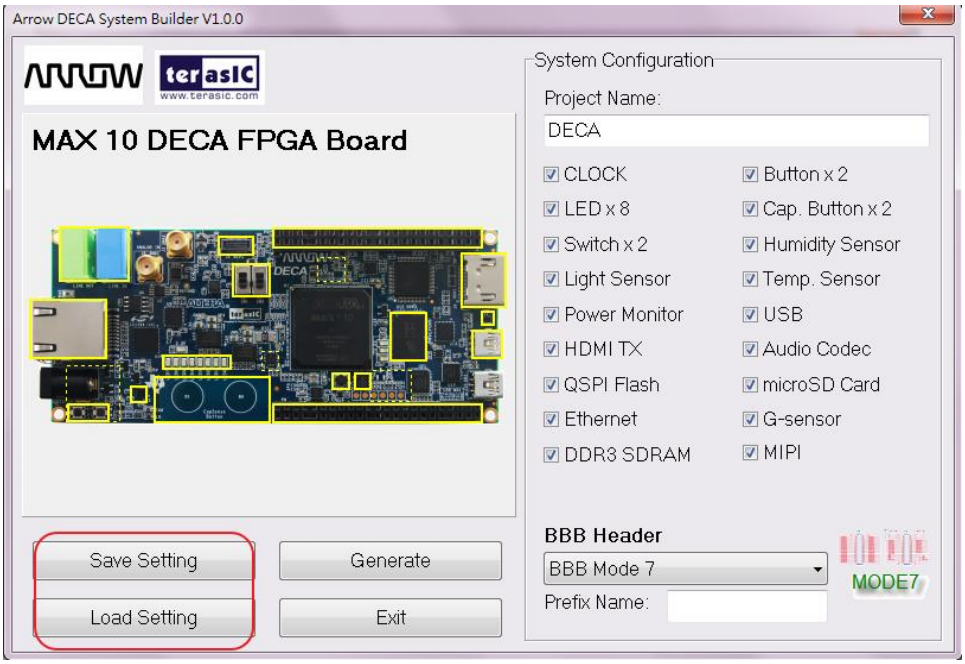


Figure 4-6 Project Settings

■ Project Generation

When users press the *Generate* button as shown in **Figure 4-7,** the DECA System Builder will generate the corresponding Quartus II files and documents, as listed in **Table 4-1**:

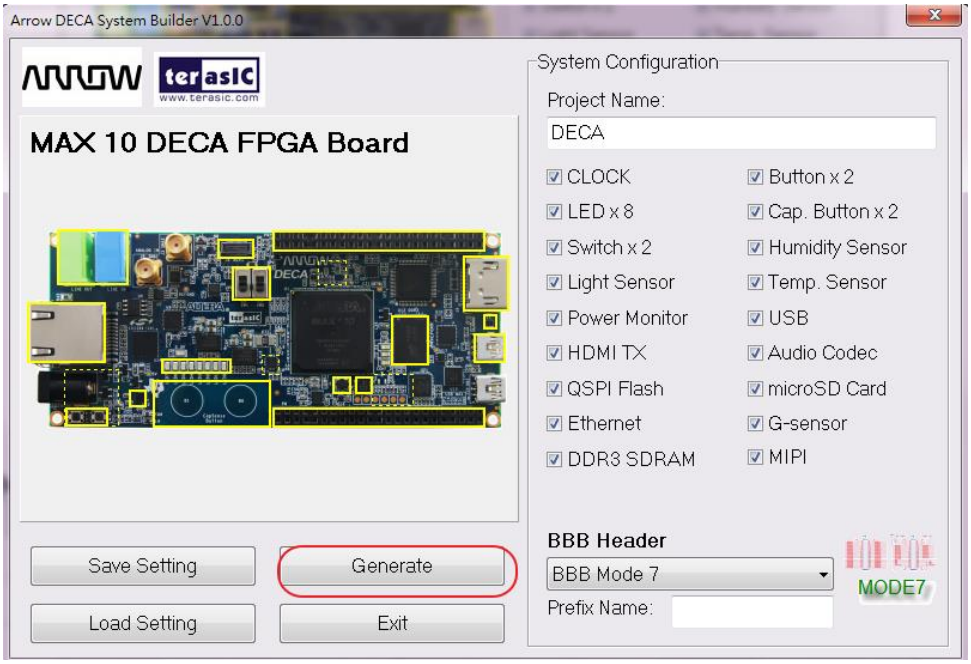


Figure 4-7 Generate Quartus Project

Table 4-1 Files generated by the DECA 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 in Quartus II and compile the project 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 Humidity & Temperature measurement, G-Sensor, and HDMI output. All the associated files can be found in the directory \Demonstrations of DECA System CD

5.1 Breathing LED

The first demo is the breathing LEDs and it's to show you how to use the FPGA to control the luminance of the LEDs by means of PWM (Pulse Width Modulation) scheme shown in **Figure 5-1**. The breathing behavior is similar to the human breath. The LEDs are divided into 2 groups. When one group of LEDs dims the light, the other group of LEDs will light on and vice versa.

Note that users can control the LED luminance by changing the duty cycle of the PWM wave, shown in **Figure 5-2**. A duty cycle is the percentage of one period where the signal is active. One period here is defined to be the time one signal to complete an on and an off cycle. The **Figure 5-2** clearly shows the brightness of the LEDs is proportional to the duty cycle when the LED is high-active.

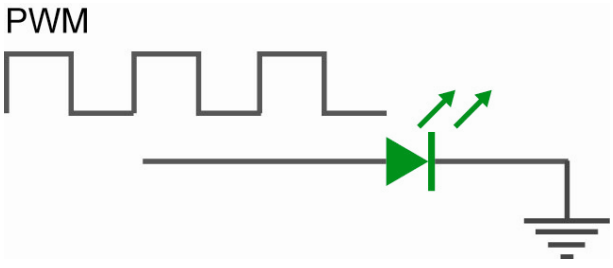


Figure 5-1 This demo uses PWM signal to drive a LED

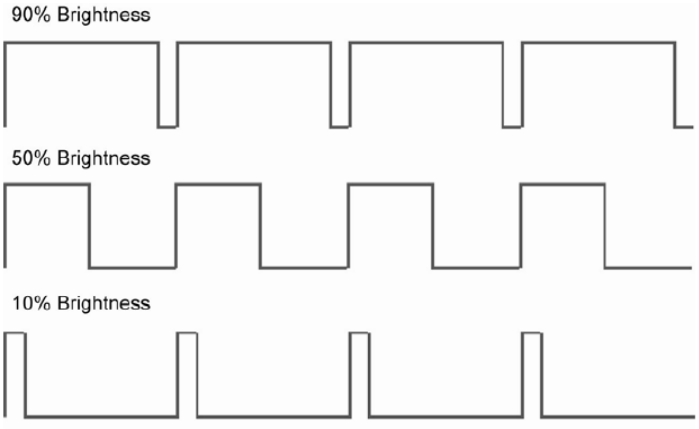


Figure 5-2 the luminance of the LEDs is proportional to the duty cycle

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations\LedBreathe
- Bit stream: DECA.sof

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\LedBreathe
- Batch file: test.bat
- FPGA configure file: DECA.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 (J10) on the DECA board to the host PC.
- Power on the DECA board.
- Execute the demo batch file “test.bat” under the folder Demonstrations\LedBreathe\demo_batch.

FPGA user LEDs should behave as ‘breathing”.

5.2 User IO and CLOCK

This demonstration shows how to control and verify the following user IOs and CLOCKs.

- Two 50MHz CLOCKs.
- Two double-stage switches (SW).
- Two KEYs.
- Eight active-low LEDs.
- Two touch buttons (B0/B1).

8 LEDs which directly connect to MAX10 are used as the final test result indicators in this demonstration. Two CapSense buttons made by PCB pads connect to CapSense IC (CY8CMBR3102), which can process the status value from two CapSense pads and cash it to CapSense IC register, the status value can be read by FPGA I2C bus. In this demonstration CY8CMBR3102 I2C Slave-Address is 0x6e.

■ Function Block Diagram

Figure 5-3 is the function block diagram of this demonstration. As LEDs (LED0~LED7) and KEYs are active-low, the KEY status signals input to MAX10 can be directly output to the LED indicators. When pressing a KEY, the corresponding LED lights up. Two SW signals are reversed by MAX10 and then output to LEDs. Two 50MKZ clocks are divided by 50000000(in MAX10) to acquire 1HZ clocks and output to LEDs, LEDs blink with 1HZ. Two circular pads (B0, B1) on DECA PCB board are Capsense Touch Buttons. CY8CMBR3102 IC will process the capacitance values (which are different according hand touching the pad or not) of two CapSense buttons to digital values and cash them to BUTTON_STAT register (address: 0xaa) bit0 and bit1, these values can be read by FPGA I2C bus. When touching B0, bit0=”0”, otherwise, bit0= “1”. In the same way, when touching B1, bit1=”0”, otherwise, bit1= “1”.

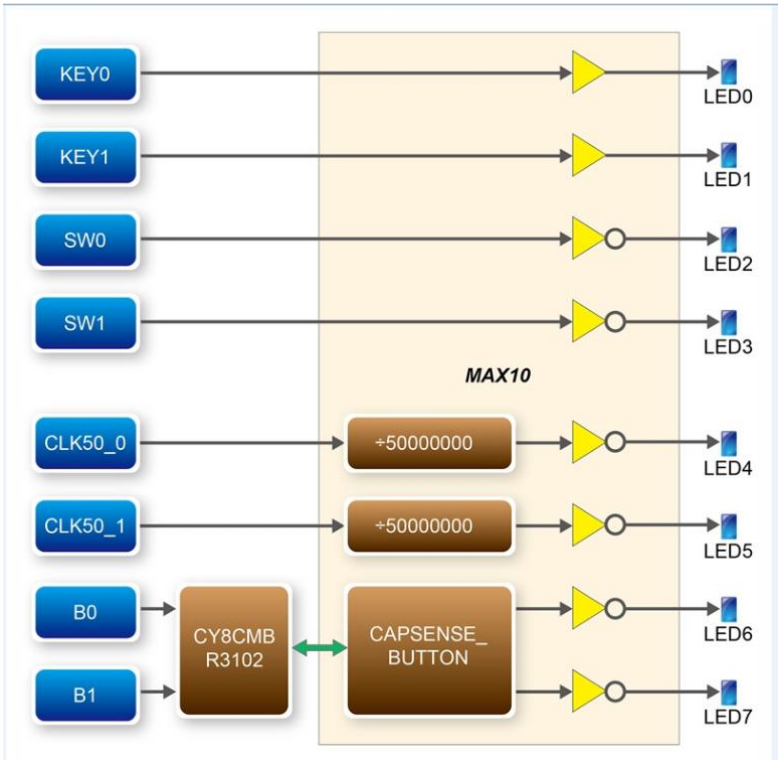


Figure 5-3 Block diagram of the User IO and CLOCK

■ Design Tools

- Quartus II v15.0

■ Demonstration Source Code

- Project directory: User_IO
- Bitstream used: DECA_User_IO.sof

■ Demonstration Batch File

Demo batch file folder: \User_IO\demo_batch

The directory includes the following files:

- Batch file: test.bat

- FPGA configuration file: DECA_User_IO.sof

■ Demonstration Setup

- Quartus II v15.0 must be pre-installed to the host PC.
- Connect the DECA board (J10) to the host PC with a USB cable and install the USB-Blaster II driver if necessary.
- Plug the 5V adapter to DECA Board.
- Execute the demo batch file " test.bat" from the directory \DECA_User_IO\demo_batch.
- Press KEY0, LED0 lights. Release KEY0, LED0 doesn't light.
- Press KEY1, LED1 lights. Release KEY1, LED1 doesn't light.
- Switch SW0 to "1" , LED2 lights. Switch SW0 to "0" , LED2 doesn't light.
- Switch SW1 to "1" , LED3 lights. Switch SW1 to "0" , LED3 doesn't light.
- LED4 and LED5 blink with 1HZ.
- Touch DECA B0 pade with finger, LED6 lights . Remove your finger from B0, LED6 doesn't light.
- Touch DECA B1 pade with finger ,LED7 lights . Remove your finger from B1, LED7 doesn't light.
- Table 5-1 shows the details of each LED status.

Table 5-1 Status of LED Indicators

Name	*Description*
LED0	Press KEY0, LED0 lights. Release KEY0, LED0 doesn't light.
LED1	Press KEY1, LED1 lights. Release KEY1, LED1 doesn't light.
LED2	Switch SW0 to "1" , LED2 lights. Switch SW0 to "0" , LED2 doesn't light.
LED3	Switch SW1 to "1" , LED3 lights. Switch SW1 to "0" , LED3 doesn't light.
LED4	Always blink with 1hz.
LED5	Always blink with 1hz.
LED6	Touch B0, LED6 lights . Don't touch B0, LED6 doesn't light.
LED7	Touch the B1, LED7 light s. Don't touch B1, LED7 doesn't light.

5.3 Humidity and Temperature Measure

A humidity/temperature sensor IC (HDC1000) embedded on DECA board can monitor environment temperature and humidity values and cash the values to the IC register. FPGA (MAX10) can read the values by I2C bus.

■ Function Block Diagram

Figure 5-4 is the function block diagram of this demonstration. RH_TEMP module keeps sending master I2C timing to monitor (HDC1000) to read real-time temperature/humidity value. HDC1000 I2C Slave_address is 0x80 (8-bit), temperature and humidity values are saved respectively in Temperature register (address: 0x00) and Humidity register (address: 0x01). TP module here has no processing function, it's just used to collect monitor signals for SingalTapeII IF. (SignalTap II is Altera Logic Analyzer IP which is used to display real-time measurements in this demonstration).

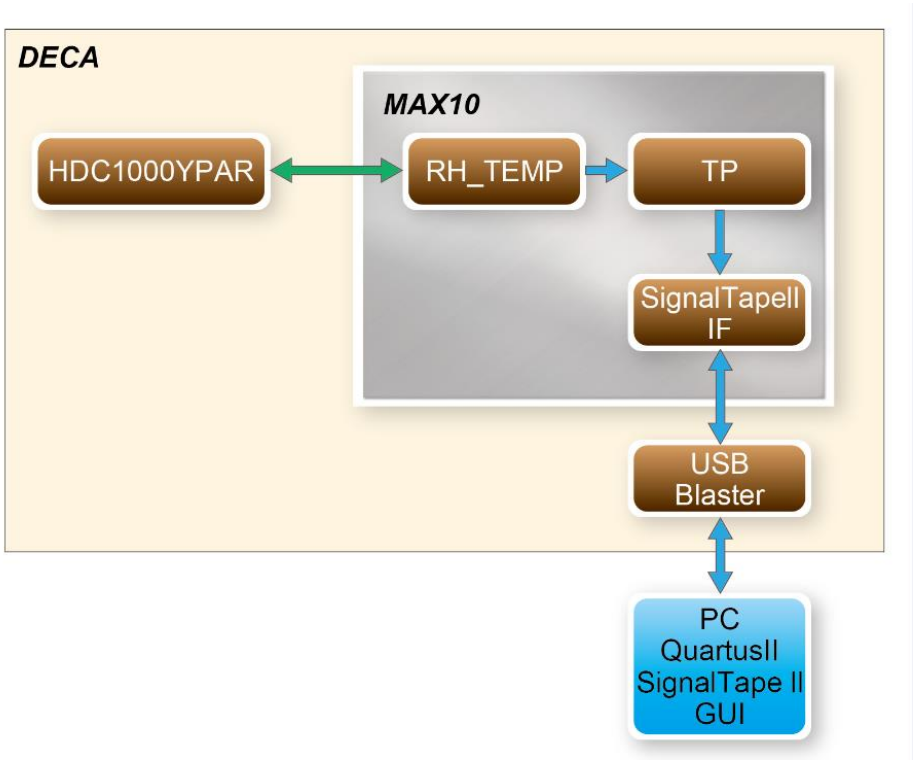


Figure 5-4 Block diagram of the Humidity and Temperature

When reading the IC measurement values, special attention is need on the timing control, otherwise

you can not get the correct values. Control mainly depends on the coordination of “I2C read/write” and “DRDYn”. Steps are as follows (can refer to HDC1000YPAR datasheet, page 10~13):

Step 1.write the temperature/humidity value pointer.

Step2.When polling HDC1000YPAR “DRDYn” is "LOW".

Step3.Read the value by I2C (In this moment, I2C can’t write a pointer as this operation can cause HDC1000 to restart and measure a new value, and then, you will get an incorrect value).

■ Design Tools

- Quartus II v15.0

■ Demonstration Source Code

- Project directory: DECA_Humidity_Temperature
- Bitstream used: DECA_Humidity_Temperature_Measurement.sof

■ Demonstration Batch File

Demo batch file folder: \DECA_Humidity_Temperature\demo

The directory includes the following files:

- Batch file: test.bat
- FPGA configuration file: DECA_Humidity_Temperature_Measurement.sof

■ Demonstration Setup

- Quartus II v15.0 must be pre-installed to the host PC.
- Connect the DECA board (J10) to the host PC with a USB cable and install the USB-Blaster II driver if necessary
- Plug the 5V adapter to DECA Board
- Execute the demo batch file “ test.bat” from the directory \DECA_Humidity_Temperature\demo_batch
- Double click the **HC1000.stp** file in the path \DECA_Humidity_Temperature\demo_batch, QuartusII SignalTap II will be activated as shown in **Figure 5-5**.

- Click Signaltap II "Start" icon to run Signaltap II.
- The signal names "Temperature_S" and "Humidity_S" in the SignalTap II window are the environment temperature and humidity values.

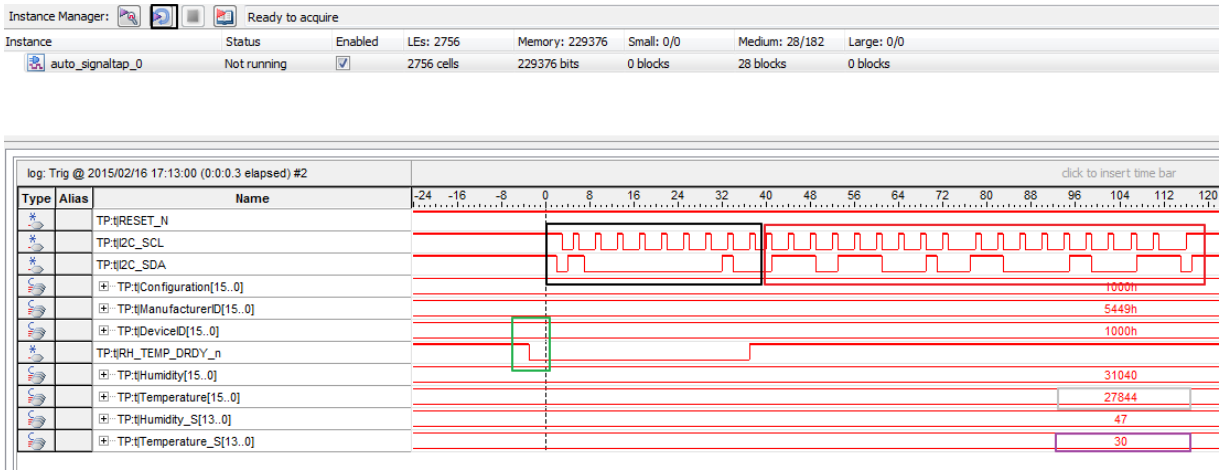


Figure 5-5 Reading the registers of Humidity and Temperature via I2C interface

Figure 5-5 notes:

(1) Green box: State Machine starts to poll, RH_TEMP_RDY_n changes from High level to Low level, prepared for I2C read process.
(2) Black box: I2C read command.
(3) Red box: I2C read temperature data (16 bits).
(4) Grey box: Latch temperature 16 bits data, and get 6cc4h (= 27844d) (it is current Temperature Register value).

Purple box: is the actual temperature value (refer to datasheet, page14: actual temperature value= (Temperature Register / 65536) *165-40 = (27844/65536)*165 -40= 30.1=30).

5.4 Power Monitor

A power monitor IC (INA230AIRGTR) embedded on DECA board can monitor MAX10 real-time current and power. This IC can work out current/power value as multiplier and divider are embedded in it.

There is a shunt resistor R72 (R_{SHUNT} =0.003 Ω) for INA230AIRGTR in the circuit, when MAX10 runs process or MAX10 Core current appears, there will be a voltage drop (named Shut Voltage) on R72. INA230 monitors the Shut Voltage and works out current (I) base on Ohm's Law (V/R).

INA230 can also monitor Bus Voltage (V_B) and use embedded multiplier to work out P (I x V_B). INA230 register value can be read by MAX10 I2C. INA230AIRGTR I2C Slave-address is 0x80.

■ Function Block Diagram

Figure 5-6 is the function block diagram of this demonstration. POWER_MONITOR module is used to read INA230 register values. POWER_MONITOR module keeps sending I2C timing to read INA230 registers which including Bus_Voltage Register (Address: 0x02), Shunt_Voltage Register (Address: 0x01), Current Register (Address: 0x04) and Power Register (Address: 0x03). TP module here has the same function with the TP module in previous project, it is used to collect signals to SignalTap II.

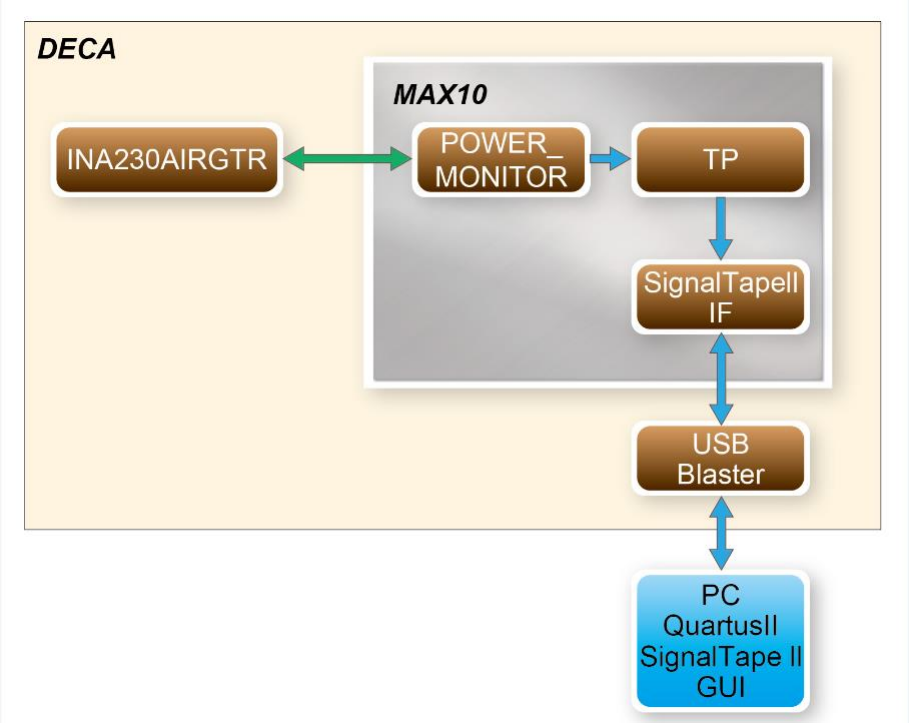


Figure 5-6 Block diagram of the Power Monitor

The precision of the current value converted by INA230 depends on Calibration (CAL) register setting. Refer to following equation for CAL Register setting:

$$CAL = \frac{0.00512}{Current\_LSB \bullet R_{SHUNT}}$$

$$Current\_LSB = \frac{Maximum\ Expected\ Current}{2^{15}}$$

On the DECA board, Rshunt value is 0.003 (R27 resistance). Bus Voltage (V_B) and Short Voltage (V_S) are actual measured values saved in registers. With these two values INA230 can work out current ($I_S = V_S / R_S$) and power ($P = V_B * I_S$). There are two ways to work out Is and P: One is calculating manually according equation; the other is using INA230 CAL setting.

Take this project as an example. For the first way, two registers values are Bus_Voltage =960, Shunt_Voltage=11, and INA230 Bus Voltage 1LSB is 1.25mV, then, Bus Voltage =960 * 1.25mV = 1.2V. In the same way, Shunt Voltage 1LSB is 2.5uV, then, Shunt Voltage = 11 * 2.5uV= 27.5uV. The final results are $I_S = V_S / R_S$ = 9.1mA; $P = V_B * I_S$ = 1.2V *9.1mA = 10.92mW. For the second way, suppose Current register 1LSB = 0.1mA, then, CAL Register value=0.00512 / (Current_LSB * R_{SHUNT}) = 0.00512/(0.1mA *0.003) =17066. When setting CAL Register value to 17066, current register value is read about 92 (Is=92*0.1mA). P (Power) value can also be read in Power register. Power register 1LSB = 2.5mW, and power actual value is P =10.92mW, then, the Power register will show value 4~5 (P=4*2.5mW=10mW). In the system CD there is a demo which can use Altera SignalTap II to observe the Current /Power real-time value.

■ Design Tools

- Quartus II v15.0

■ Demonstration Source Code

- Project directory: DECA_Power_Monitor
- Bitstream used: DECA_Power_Monitor.sof
- SignalTap II :PowerM.stp

■ Demonstration Batch File

- Demo batch file folder: \DECA_Power_Monitor\demo_batch

The directory includes the following files:

- Batch file: test.bat
- FPGA configuration file: DECA_Power_Monitor.sof

■ Demonstration Setup

- Quartus II v15.0 must be pre-installed to the host PC.
- Connect the DECA board (J10) to the host PC with a USB cable and install the USB-Blaster II driver if necessary
- Plug the 5V adapter to DECA Board
- Execute the demo batch file " test.bat" from the directory \DECA_Power_Monitor\demo_batch
- Double click the PowerM.stp in the path\DECA_Power_Monitor\demo_batch
- QuartusII SignalTap II will be activated as shown in **Figure 5-7**.
- Click Signaltap II "Start" to run Signaltap II.
- Calibration value and Current value will display on SignalTap II window.

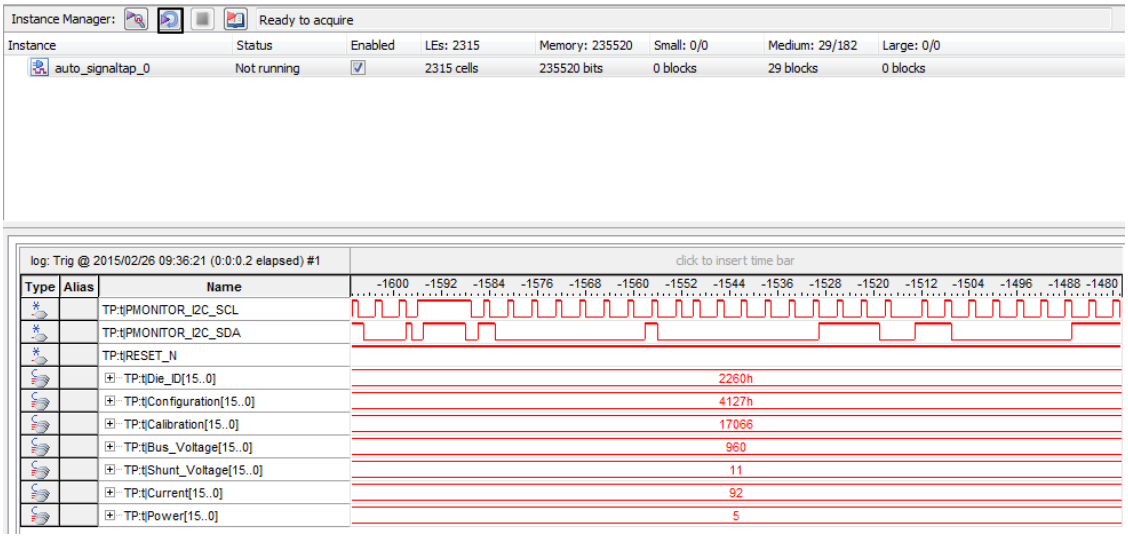


Figure 5-7 Reading the register value of voltage and current via I2C interface

5.5 Proximity/Ambient Light Sensor

There is a Proximity/Ambient Light Sensor IC (Si1143) with three LEDs on the back of D ECA. This demonstration will show how to use this IC to measure the distance between D ECA board and an object. The light from three Infrared Light diodes (DS1~DS3) impinge on an object and reflect to Si1143 photodiode. Photodiode converts the brightness of three i nfrared lights into three electricity values, pass the values to ADC IC to convert into digita

l data and save in register. This IC register can be read by MAX10 I2C bus and the IC sl ave address is 0xb4.

■ Function Block Diagram

Figure 5-8 is the function block diagram of this demonstration. LSEN_CTRL module is used to read three digital values in register. Three digital values are 16-bit, low byte and high byte are saved respectively in two registers. LSEN_CTRL module keep sending I2C master timing to read three registers: PS1_DATA [PS1_DATA1(0x27) :PS1_DATA0(0x26)], PS2_DATA [PS2_DATA1(0x29) : PS2_DATA0(0x28)], PS3_DATA[PS3_DATA1(0x2b) : PS3_DATA0(0x2a)]. In this demonstration, we take the average of the three values as the distance value and send it to LEVEL_CAMP, LEVEL_CAMP will output 8bit level to LED0~LED7.When the object is closest to the light sensor IC, more LEDs(LED0~LED7) light up. On the contrary, when object gets further to DECA board, less LEDS light up.

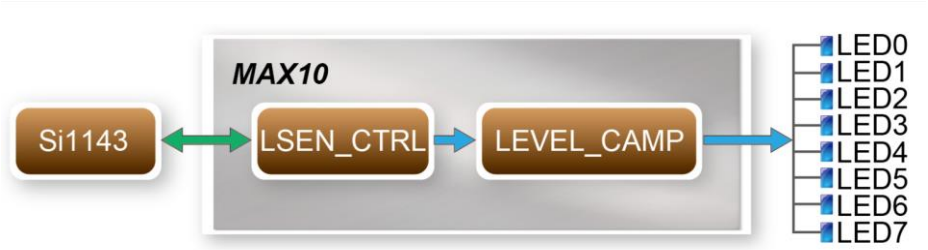


Figure 5-8 Block diagram of the Gesture Light Sensor

■ Design Tools

- Quartus II v15.0

■ Demonstration Source Code

- Project directory: Gesture_Light_Sensor_RTL
- Bitstream used: DECA_Gesture_Light_Sensor.sof

■ Demonstration Batch File

Demo batch file folder: \Gesture_Light_Sensor_RTL\demo_batch

The directory includes the following files:

- Batch file: test.bat
- FPGA configuration file: DECA_Gesture_Light_Sensor.sof

■ Demonstration Setup

- Quartus II v15.0 must be pre-installed to the host PC.
- Connect the DECA board (J10) to the host PC with a USB cable and install the USB-Blaster II driver if necessary
- Plug the 5V adapter to DECA.
- Execute the demo batch file "test.bat" from the directory \Gesture_Light_Sensor_RTL\demo_batch
- Move hand close to DECA back (because Light Sensor is on the back of DECA board), LED0~LED7 light according to the distance LEVEL. When hand gets closer to DECA board, more LEDS light up. When hand is closest to board, all LEDs light up. On the contrary, when hand gets further to DECA board, less LEDS light up, when hand is far enough, only LED0 lights up.

5.6 G-Sensor

A 3-axis G-sensor (LIS2DH12TR) embedded on DECA board can monitor DECA 3D gravitational acceleration. Three axis are x/y/z. Every axis angle value outputs by way of SPI interface. This demonstration is using a SPI_CTL(RTL code) of MAX10 to read x-axis value of LIS2DH12TR to do gradienter. Finally DECA inclination status can be reflected on LED0~LED7.

■ Function Block Diagram

Figure 5-9 is the function block diagram of this demonstration. SPI_CTL is a SPI controller that to read x-axis value of LIS2DH12TR.The value output to led_driver converts inclination angle to inclination value and ouput 8-bit to LED0~LED7.

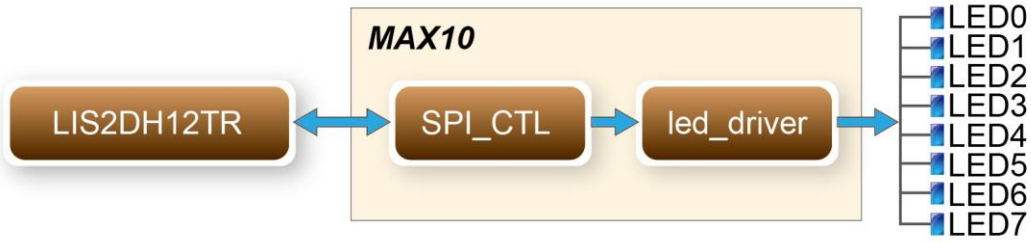


Figure 5-9 Block diagram of the G-sensor

■ Design Tools

- Quartus II v15.0

■ Demonstration Source Code

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

■ Demonstration Batch File

Demo batch file folder: \DECA_Gsensor\demo_batch

The directory includes the following files:

- Batch file: test.bat
- FPGA configuration file: DECA_Gsensor.sof

■ Demonstration Setup and Instructions

- Quartus II v15.0 must be pre-installed to the host PC.
- Connect the DECA board (J10) to the host PC with a USB cable and install the USB-Blaster II driver if necessary
- Plug the 5V adapter to DECA Board.
- Place the DECA board horizontally.
- Execute the demo batch file " test.bat" from the directory:
- \DECA_Gsensor\demo_batch
- LED0~LED7 will flash several times at the very beginning, and finally LED3 and LED4 will light up, others will light off.

- Take up DECA and tilt it left and right, two flash LEDs will moves on LED0~LED7 according the inclination value.

5.7 Line-In ADC

Want to see something even more fun? You can use the built-in ADC (Analog to Digital Converter) within FPGA and the LEDs to check out the audio volume coming from the left-channel of the Line-IN audio jack. The volume is proportional to the number of LEDs being turned on. For example, if all the LEDs are turned on, it indicates the volume is very high, and likewise if all the LEDs are turned off, it indicates there is no input volume or the volume is very low.

Figure 5-10 shows the block diagram of Line-In ADC on the DECA board. The left channel of the audio jack is the audio input to a voltage translation circuit, which later is being added 1.25V to the original signal. The translated signal then goes to the built-in ADC. Finally, the FPGA reads the digitalized 12-bit value from the ADC. The output voltage and input voltage have the following relationship:

Output Voltage (Vou) = Input Voltage (Vin) + 1.25V

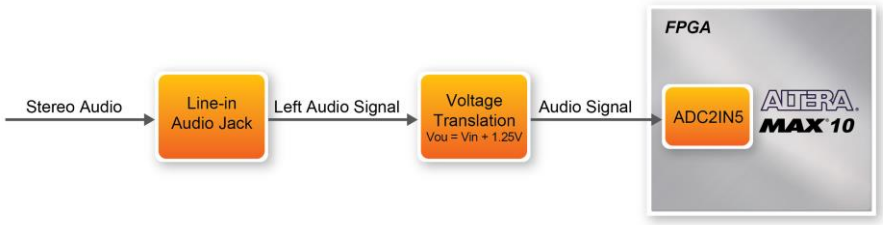


Figure 5-10 Block diagram of Line-In ADC

The built-in ADC2IN5 channel within the FPGA receives the analog input from the voltage translation circuit. Altera Modular ADC IP core is used to control the ADC. Figure 1-4 shows the IP settings required for this demo. Note the Channel 5 in the 2nd ADC needs to be enabled as the Line-IN audio signal is connected to the ADC2in5 pin.

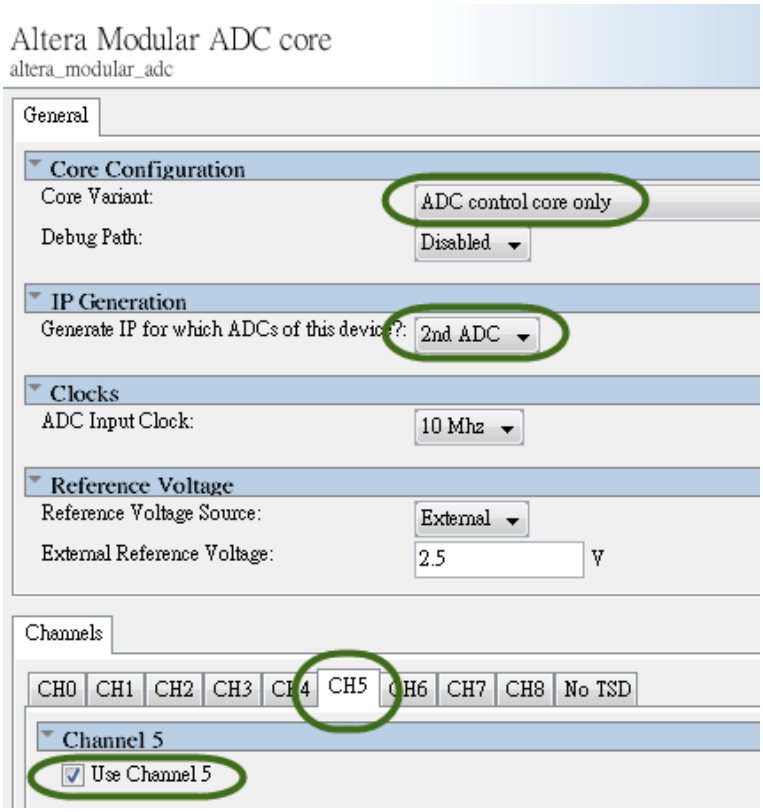


Figure 5-11 Settings of Altera Modular ADC core

The command and response streaming interfaces are used in "ADC control core only" as shown in Figure 5-12. The command interface is used to specify which ADC channel value will be queried and the response interface is used to report the converted digital value associated with the specified ADC channel.

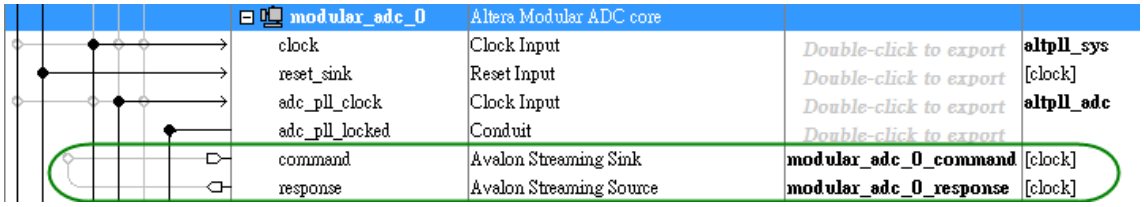


Figure 5-12 Interface of ADC control core only IP

Figure 5-13 shows the timing diagram of command and response interfaces. The responded digital value is stored in an unsigned 12-bit binary format with range from 0 to 4095. It is mapped to the voltage range 0V to 2.5V, as shown in Figure 5-14.

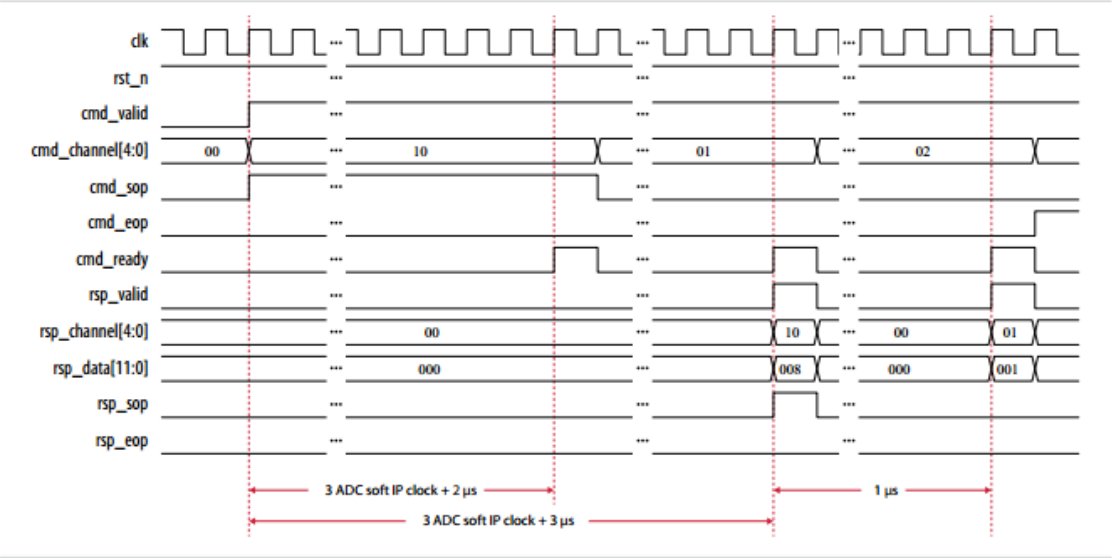


Figure 5-13 ADC timing diagram of command and response interfaces

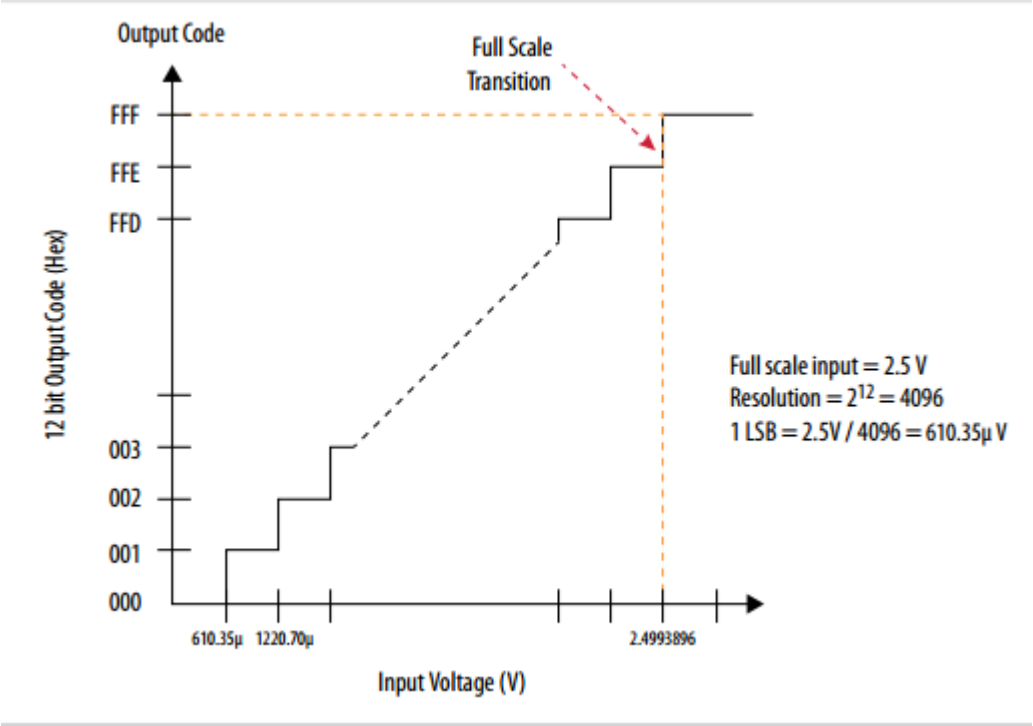


Figure 5-14 Relation between 12-bit output code and input voltage

The 12-bit digital value 0 ~ 4095 represents 0 ~ 2.5V voltage input. It is mapped to the input audio voltage from -1.25V to 1.2V. Hence 0x800 is mapped to 0V audio input, 0xFFF is mapped to 1.25V input audio, and 0x00 is mapped to -1.25V input audio. The absolute audio voltage is first

calculated and then the array LEDs will respond accordingly in this demonstration to indicate the input audio volume.

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: LineIn_ADC
- Bit stream: DECA.sof

■ Demonstration Batch File

- Demo batch file folder: LineIn_ADC\demo_batch
- Batch file: test.bat
- FPGA Configure File: DECA.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 (J10) on the DECA board to the host PC.
- Input audio into the Line-In Jack (J2) on the DECA board.
- Power on the DECA board.
- Execute the demo batch file “test.bat” under the folder LineIn_ADC\demo_batch.
- The LEDs indicate the volume of input audio.

5.8 HDMI TX

Want to know more about HDMI? We now introduce how to program the HDMI transmitter to generate video pattern and audio source. The entire reference is composed into three parts: video design, audio design and I2C design. A set of pre-built video patterns and audio serial data will be sent to the HDMI transmitter to drive the HDMI display with speaker. You can hear beeping sound from the HDMI display speaker when you press KEY[0] on the DECA board.

Figure 5-15 shows the system block diagram of this reference design. I2C Controller and I2C

HDMI Config use I2C interface to configure HDMI transmitter, I2C HDMI Config contains HPD(Hot Plug Detect) interrupt action. When HPD interrupt happens, the HDMI Transmitter will be re-configure. HDMI transmitter needs to be properly configured before occupation. PLL and Video Pattern Generator are designed to send video pattern to HDMI transmitter, the default resolution setting of video pattern is FULL HD (1920*1080p). The resolution of video pattern could be modified by adjusting the parameters of PLL and video pattern generator module. PLL and Audio Generator are designed to transmit audio pattern to HDMI Transmitter, the audio transmitting interface is using I2S in this demo.

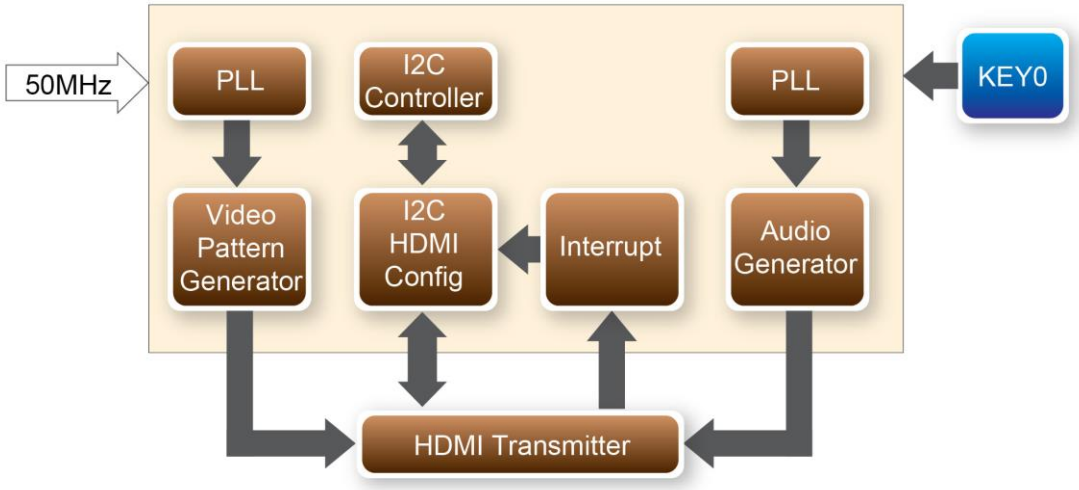


Figure 5-15 Block Diagram of the HDMI TX Demonstration

■ HDMI Transmitter(ADV7513) Register

Before you develop HDMI Transmitter, setting the video format and audio frequency in Address(0x15) Register and the format of Address(0xAF) Register could save you lots of developing time. When we use 48KHz Sampling Rate, the video format is 24 bit RGB 4:4:4. For more details, please refer to "ADV7513_Programming_Guide_R0."

■ Audio Generator

The ADV7513 can accommodate from 2 to 8 channels of I2S audio at up to a 192 KHz sampling rate. The ADV7513 supports standard I2S, left-justified serial audio, and right-justified serial audio. **Figure 5-16** shows the left-justified serial audio with I2S Standard Audio of 16 bit per channel.

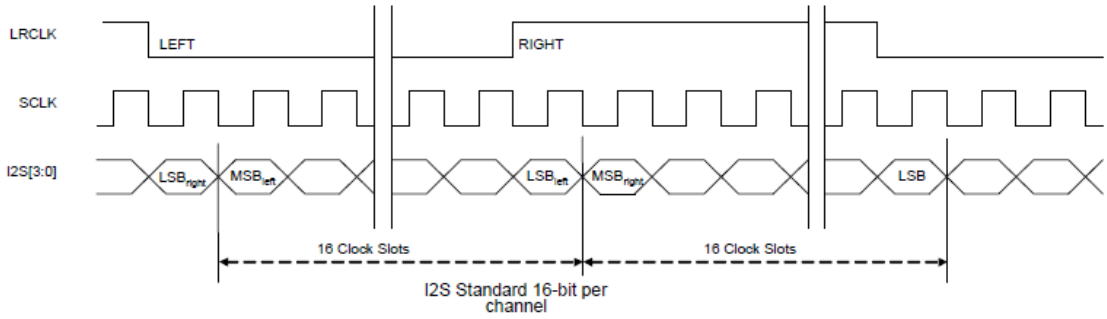


Figure 5-16 I2S Standard Audio – 16 bit per channel

If you want to modify the frequency of audio output, you have to modify the register value at register 0x15 (reference ADV7513_Programming_Guide_R0.pdf). I2S Standard uses MSB to LSB serial way of transmitting. The demo uses sinusoid signal to make sound from a reference of look up table created by calculated sinusoid wave data.

■ Video Pattern Generator

The module “Video Pattern Generator” copes with generating video patterns to be presented on the LCD monitor. The pattern is composed in the way of 24-bit RGB 4:4:4 (RGB888 per color pixel without sub-sampling) color encoding, which corresponding to the parallel encoding format defined in **Table 5-2** of the "ADV7513 Hardware User's Guide," as shown below.

Table 5-2 Display Modes of the HDMI TX Demonstration

Pixel Data [23:0]																	
17	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R[7:0]						**G[7:0]**						**B[7:0]**					

■ Demonstration Source Code

Quartus Project directory: HDMI_TX

- Project directory: HDMI_TX
- Bit stream:HDMI_TX.sof

■ Demonstration Batch File

- Demo batch file folder: HDMI_TX\demo_batch
- Batch file: test.bat
- FPGA Configure File:HDMI_TX.sof

■ Demonstration Setup

- Make sure Quartus II and USB-Blaster II driver are installed on your PC.
- Connect the DECA board to the LCD monitor with the on-board HDMI connector via an HDMI cable.
- Power on the DECA board .
- Use File Manager to locate the "HDMI_TX\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**.
-

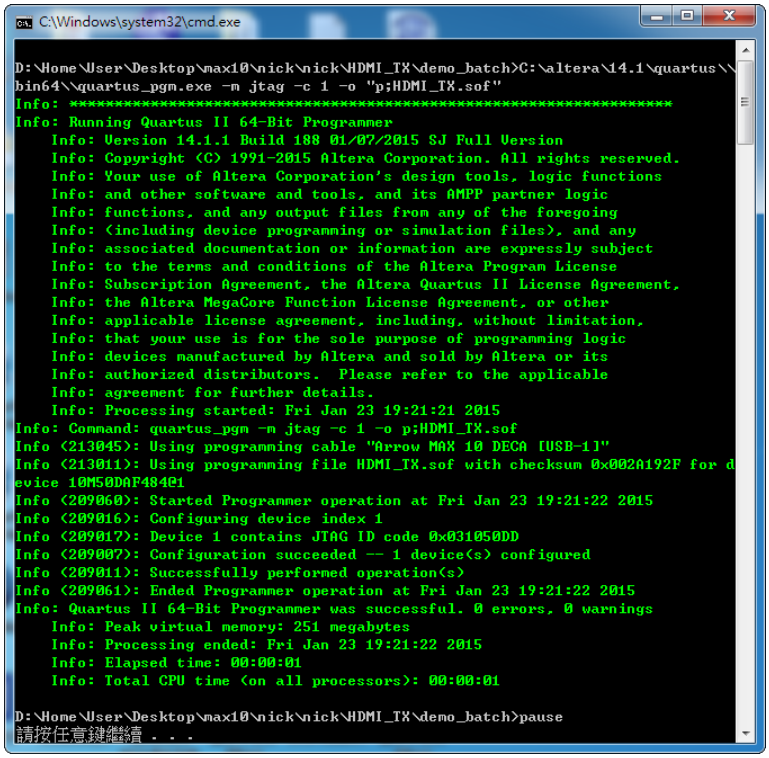


Figure 5-17 Launching the HDMI TX Demonstration using the "demo_batch" Folder

Wait for a few seconds for the LCD monitor to power up itself. And you should see a pre-defined video pattern shown on the monitor, as shown in **Figure 5-18**. you can hear beep sound from the HDMI display speaker when your press KEY0 on the DECA board.

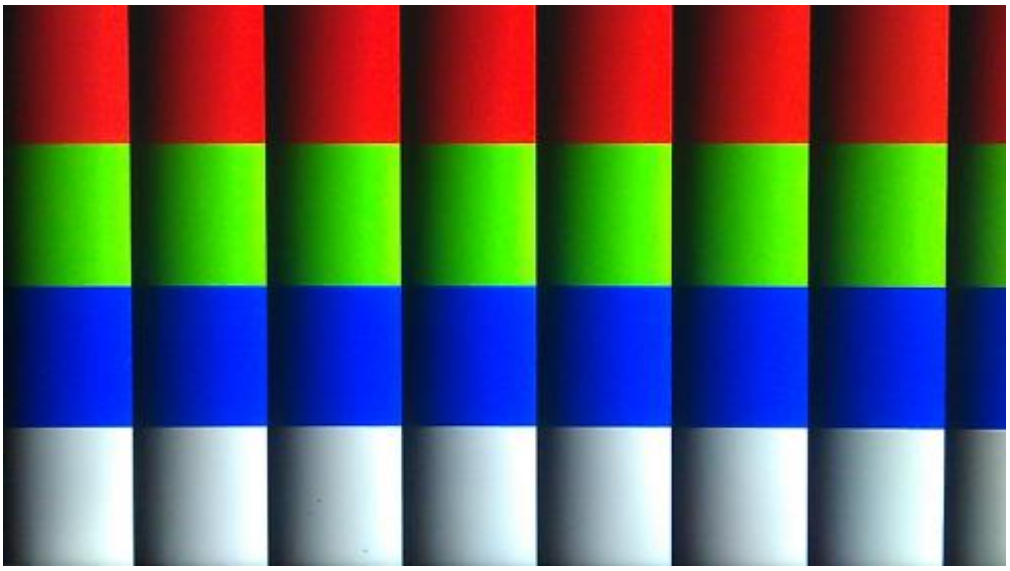


Figure 5-18 The Video pattern used in the HDMI TX Demonstration

Chapter 6

NIOS Based Example Codes

Can't wait to get started with the DECA board? We provide several NIOS based examples for you to try them on the DECA board. All of the NIOS based examples can be found in the System CD under the folder Demonstrations. You are free to use them or modify these examples in your future design!

6.1 CapSense Button

There is a special capacitive touch sensor called CapSense on DECA board. Capacitive touch sensors are user-interface device that use the capacitance of the human body to detect the presence of a finger on, or near, a sensor.

Figure 6-1 shows the system block diagram of this reference design. The capacitive touch sensor chip CY8CMBR3102 is controlled through I2C interface. In this design, an I2C OpenCore controller is used to communicate with the CY8CMBR3102. The NIOS program controlls the IC2 OpenCore based on a specified I2C OpenCore c-code library. The NIOS program use polling method the monitor the touch sensor status.

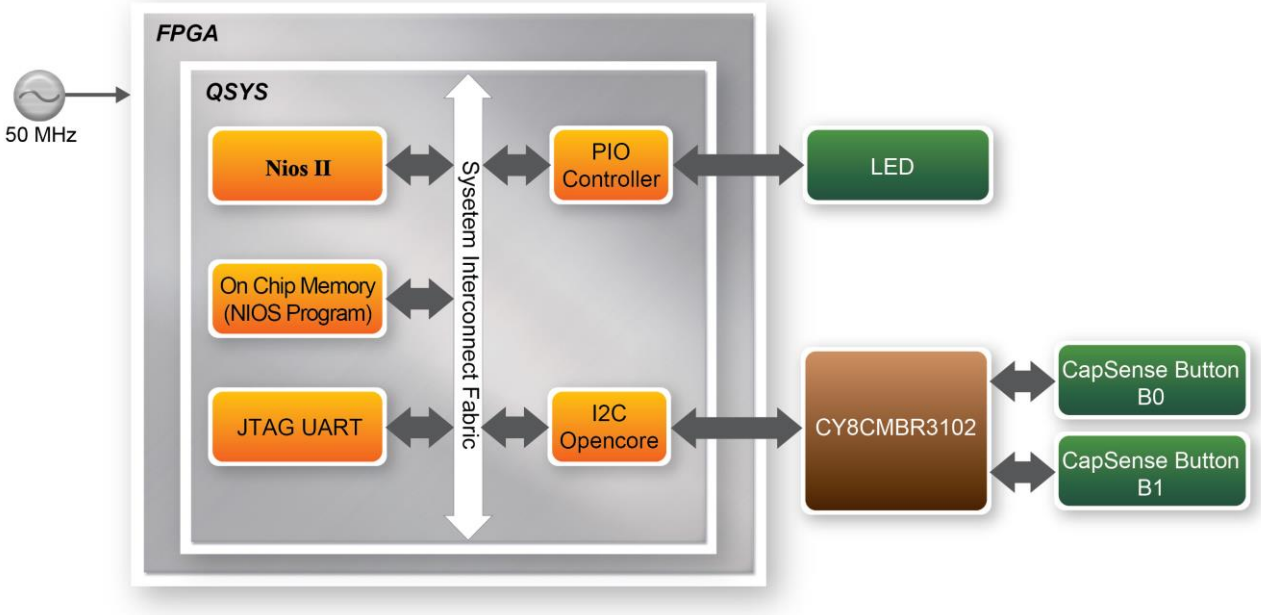


Figure 6-1 Block diagram of CY8CMBR3102 demonstration

The touch status is reported in the state register in CY8CMBR3102. **Figure 6-2** shows the CapSense state register content. The register address is 0xAA. There are 16 bits in the state register, but CY8CMBR3102 has only two sensors so only register bit CS0 (Bit0) and CS1 (Bit1) are used. When CapSense is pressed, the value of corresponding CS bit will be 1. The CS bit will be 0 once CapSense is released.

Bits	15	14	13	12	11	10	9	8
Host Access	R	R	R	R	R	R	R	R
Device Access	RW	RW	RW	RW	RW	RW	RW	RW
Bit Name	CS15	CS14	CS13	CS12	CS11	CS10	CS9	CS8

Bits	7	6	5	4	3	2	1	0
Host Access	R	R	R	R	R	R	R	R
Device Access	RW	RW	RW	RW	RW	RW	RW	RW
Bit Name	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0

0: Sensor is inactive (not touched)

1: Sensor is active (touched)

Figure 6-2 Register table of CapSense state

Figure 6-3 shows the host reading CapSense button state through I2C. Note, the CY8CMBR3102 wakes up from the low-power state upon address match but still sends NACK until it transits into Active state. When the device sends NACK from a transaction, the host is expected to repeat the

transaction until it receives ACK. The section in green shows host sends I2C slave address command repeatedly until CY8CMBR3102 responds with ACK. It indicates host wakes up CY8CMBR3102 from standby mode (low-power state). Host will then send register address command to CY8CMBR3102 until it responds with ACK and return the register data.

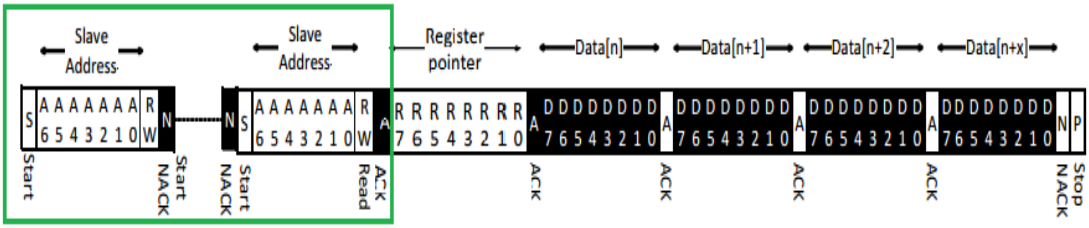


Figure 6-3 Read CapSense button state through I2C

■ Demonstration Source Code

- Project directory: Capsense_NIOS
- Bit stream: Capsense_NIOS.sof
- Nios II Eclipse: Capsense_NIOS\Software

■ Demonstration Batch File

Demo Batch File Folder: Capsense_NIOS\demo_batch

The demo batch includes the following files.

- Demo batch file folder: Capsense_NIOS\demo_batch
- Batch file: test.bat, test.sh
- FPGA configure file: Capsense_NIOS.sof
- NIOS program: Capsense.elf

■ Demonstration Setup

- Please make sure Quartus II and USB-Blaster II driver are installed on the host PC.
- Connect the USB cable to the USB-Blaster II connector (J10) between the DECA board and the host PC.
- Power on the DECA board.
- Execute the demo batch file “test.bat” under the batch file folder Capsense_NIOS \demo_batch.
- Nios terminal will display the “Please touch the CapSense Button on DECA” message.

- Press the CapSense button will light up corresponding LED, as shown in **Figure 6-4** If B0 or B1 is pressed, LED[0] or LED[1] will lit, respectively. If both B0 and B1 are pressed, both LED[0] and LED[1] will lit simultaneously.

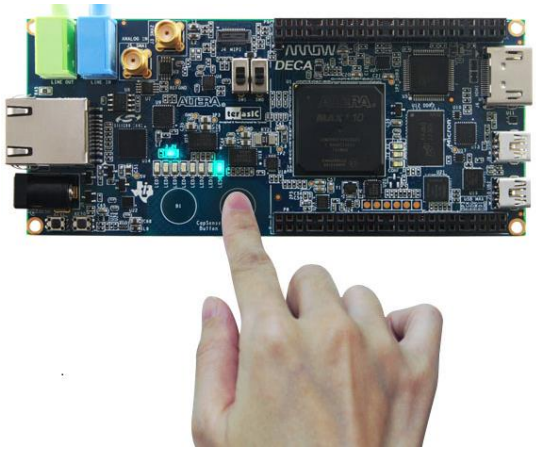


Figure 6-4 Press the CapSense button

6.2 Temperature Sensor

This demonstration illustrates how to use the LM71CIMF sensor with the Nios II Processor to realize the function of board temperature detection. **Figure 6-5** shows the system block diagram of this demonstration. The ambient temperature information, which is collected by the built-in temperature sensor on the DECA board, can be converted into digital data by a 14-bit A/D converter. A **Temp Controller** is used by Nios II software to access the sensor's registers. The C program will configure and read the registers, and then calculates the centigrade degree. The relative values are finally displayed onto the nios2-terminal window, in order to let the user monitor the board real-time temperature.

The sensor has three registers: **Temperature**, **Configuration** and **Manufacturer/Device Identification** registers, respectively. The sensor will output the manufacturer/device ID information in shutdown mode when Configuration Register set to 0xFF, and output 16bits temperature digital data in Continuous conversion mode when Configuration Register set to 0x00.

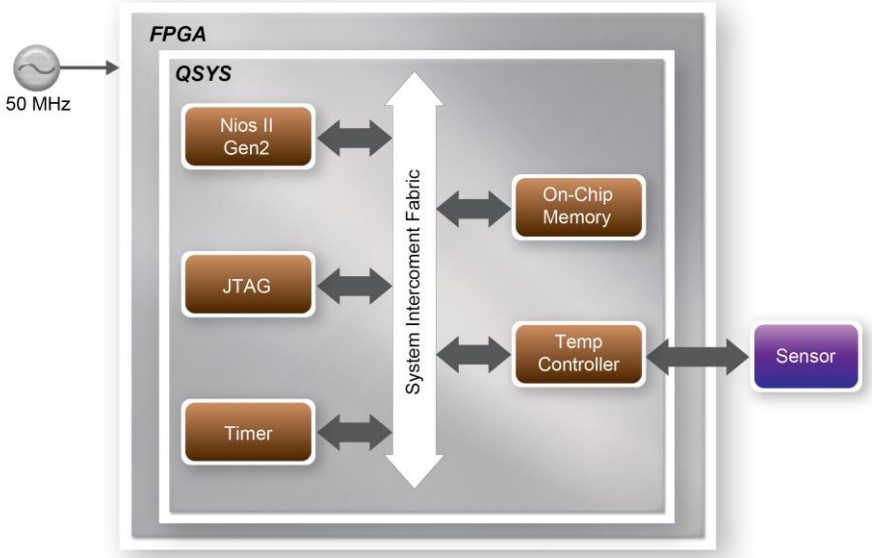


Figure 6-5 Block diagram of the Temperature Demonstration

■ Demonstration Source Code

- Project directory: DECA_TEMP
- Bit stream used: DECA _TEMP.sof
- Nios II Workspace: DECA _TEMP\Software

■ Demonstration Batch File

Demo Batch File Folder: DECA _TEMP\demo_batch

The demo batch file includes the following files:

- Batch File: DECA_TEMP.bat, DECA _TEMP.sh
- FPGA Configure File : DECA _TEMP.sof
- Nios II Program: DECA _TEMP.elf

■ Demonstration Setup, File Locations, and Instructions

- Make sure Quartus II v15.0 and Nios II v15.0 are installed on your PC.
- Power on the DECA board.
- Connect USB Blaster II to the DECA board and install USB Blaster II driver if necessary.

- Execute the demo batch file " DECA _TEMP.bat" under the batch file folder, DECA _TEMP \demo_batch.

After Nios II program is downloaded and executed successfully, the related information will be displayed in nios2-terminal , shown in **Figure 6-6**.

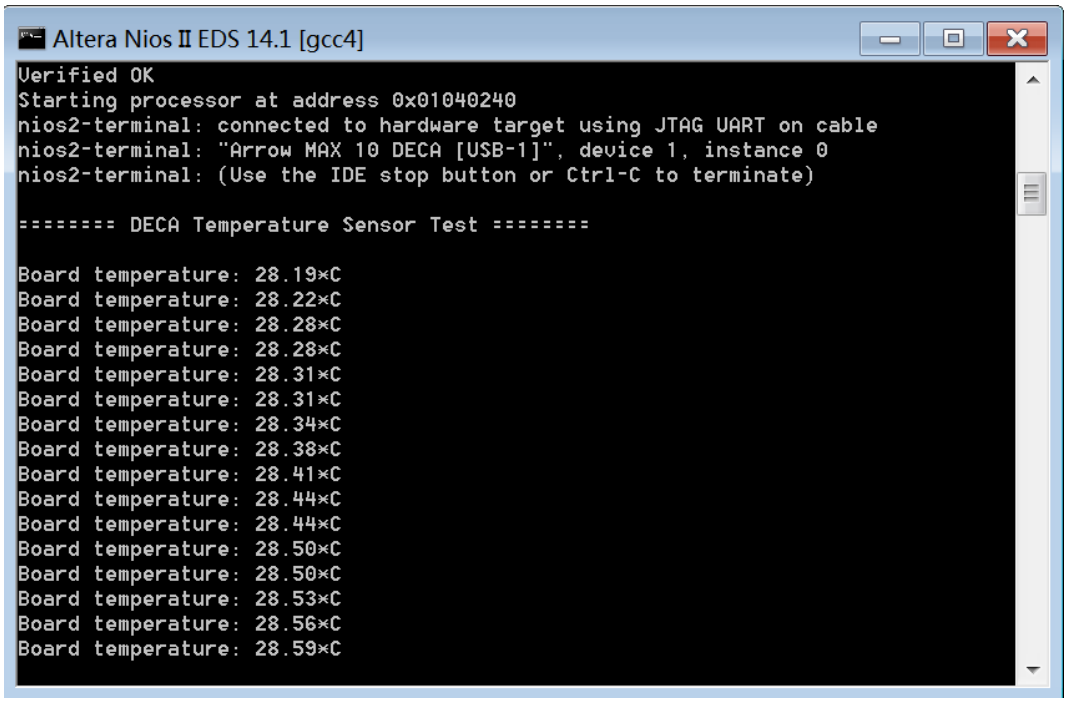


Figure 6-6 Running results of the Temperature demonstration

6.3 Power Monitor

The Power Measurement demonstration illustrates how to measure the DECA power consumption based on the built-in power measure circuit. The power monitor chip INA230 is programmed through I2C protocol which is implemented in the C code. The I2C pins from power monitor chip are connected to Qsys System Interconnect Fabric through PIO controllers. **Figure 6-7** shows the block diagram of this demonstration.

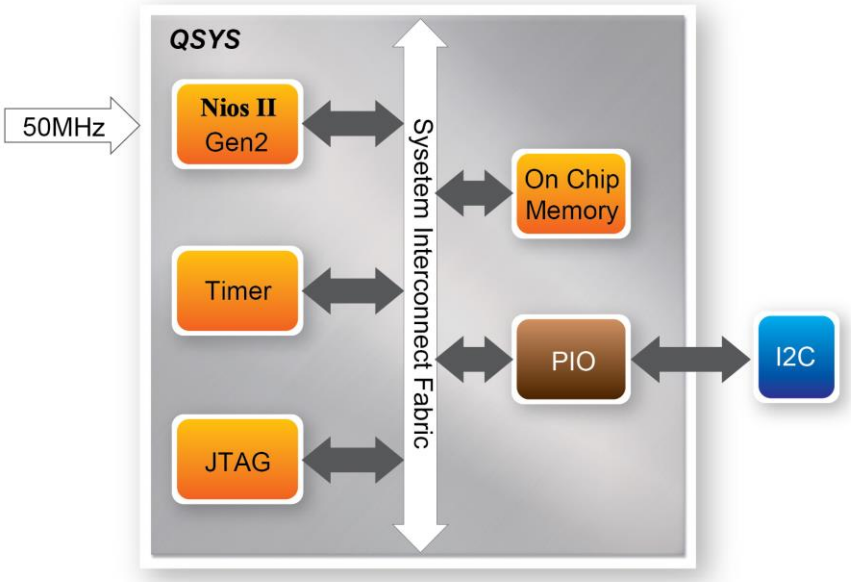


Figure 6-7 Block diagram of Power monitor

This demonstration uses an embedded NiosII gen2 processor to read the power value from the power monitor chip through the I2C interface. Based on sense resistors, the program can calculate the associated voltage, current and power consumption. The relative information is populated on the Nios-Terminal and is updated every two seconds.

■ Demonstration File Locations

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

■ Demonstration Setup and Instructions

- Make sure Quartus II is installed on your PC.
- Connect USB cable to the DECA board and install the USB Blaster II driver if necessary.
- Execute the demo batch file DECA_Power_Monitor_Nios.bat to load the bitstream and software execution file to the FPGA.

- The Nios II console will display the current value of voltage, current and power as shown in Figure 6-8.

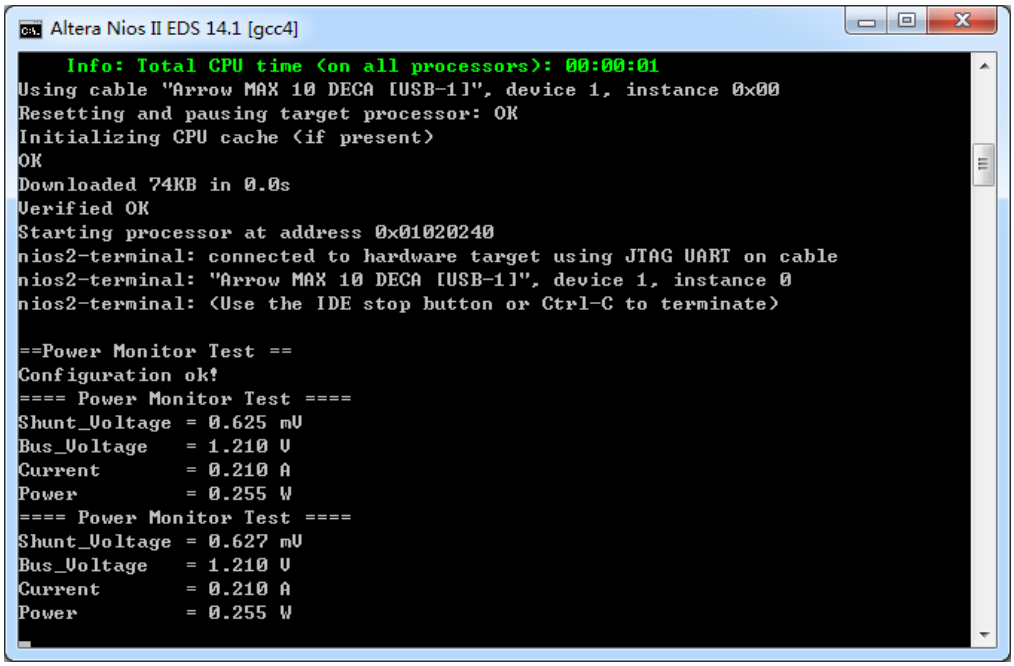


Figure 6-8 Display Progress and Result Information for the Power Monitor Demonstration

6.4 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.

Figure 6-8 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.

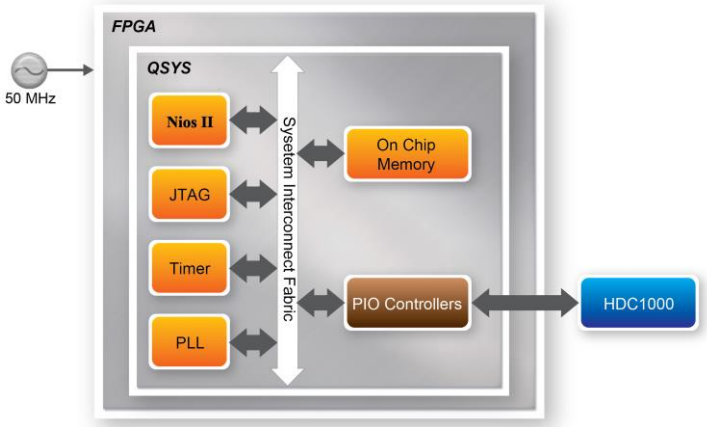


Figure 6-9 Block diagram of Humidity and Temperature Sensor

This demonstration demonstrates basic function of HDC1000 and shows temperature and humidity reading in different acquisition mode. HDC1000 can perform a measurement of both humidity and temperature, or only humidity or temperature. 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 Trigger 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.

■ System Requirements

The following items are required for this demonstration.

- DECA board x1

■ Demonstration File Locations

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

■ Demonstration Setup and Instructions

- Make sure Quartus II and USB-Blaster II driver are installed on your PC.
- Connect the USB cable to the USB Blaster II connector (J10) on the DECA board and host PC.
- Input signals into the both SMA(J3 and J5) on the DECA board.
- Power on the DECA board.
- Execute the demo batch file "test.bat" under the batch file folder, Humidity_Temperature_NIOS \demo_batch.
- NIOS terminal will display the humidity and temperature values as shown in **Figure 6-10**

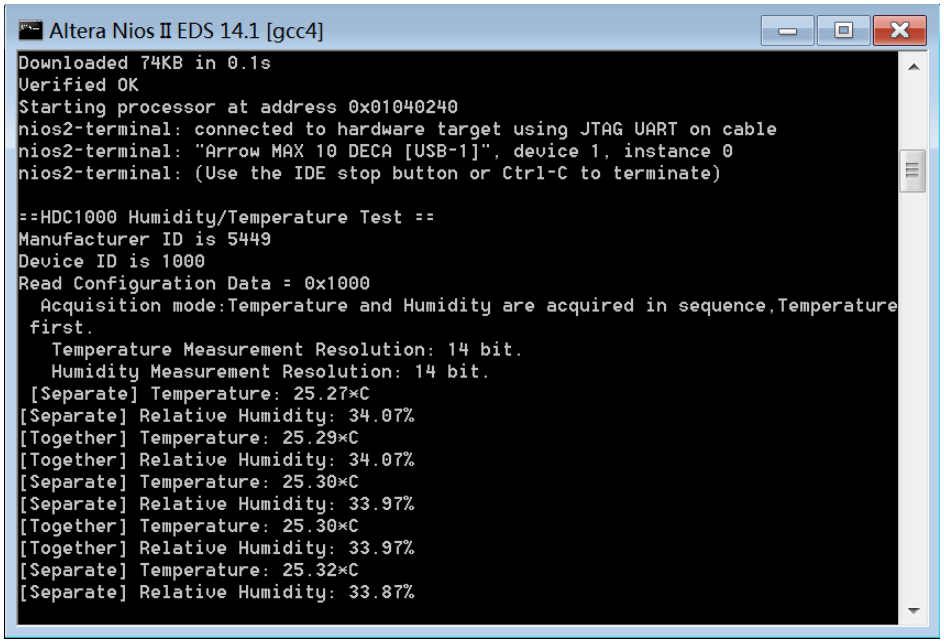


Figure 6-10 Screenshot of Humidity and Temperature Sensor Demo

6.5 G-Sensor

This demonstration illustrates how to use the digital accelerometer on the DECA board to measure the low-power 3-axis LIS2DH12 in tilt-sensing applications. The acceleration measurement is implemented using the I2C/SPI interfaces to measure the accelerations of X/Y/Z. As the board is tilted, the digital accelerometer detects the tilting movement. The accelerations of 3-axis is displayed on the NIOS II console.

The LIS2DH12 has user-selectable full scales of 2g /±4g/8g/16g , and it is capable of measuring accelerations with output data rates from 1 Hz to 5.3 KHz. In this demonstration, The 3-axis accelerometer chip LIS2DH12 is programmed through I2C protocol which is implemented in the C code. I2C interface is selected by setting the CS pin to high on the DECA board. **Figure 6-11** shows the block diagram of this demonstration.

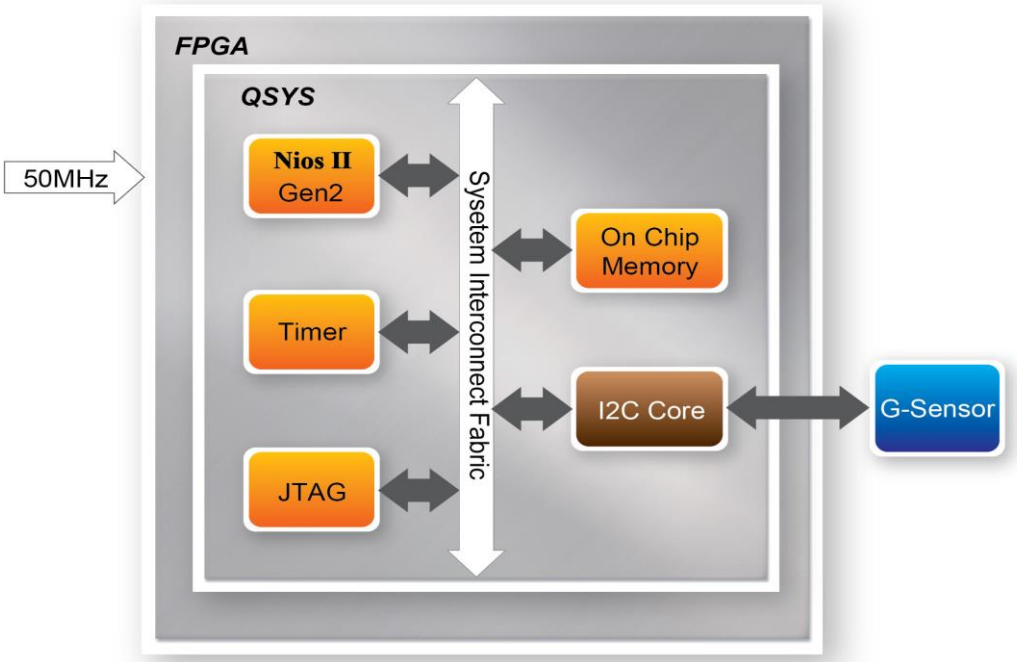


Figure 6-11 Block diagram of G-sensor

■ System Requirements

The following items are required for this demonstration.

- DECA board x1

■ Demonstration File Locations

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

■ Demonstration Setup and Instructions

- Make sure Quartus II is installed on your PC.
- Connect USB cable to the DECA board and install the USB BlasterII driver if necessary.

- Execute the demo batch file DECA_GSensor_Nios.bat to load the bitstream and software execution file to the FPGA.
- Shaking the board, the Nios II console will display the accelerations of 3-axis accelerometer as shown in **Figure 6-12**.

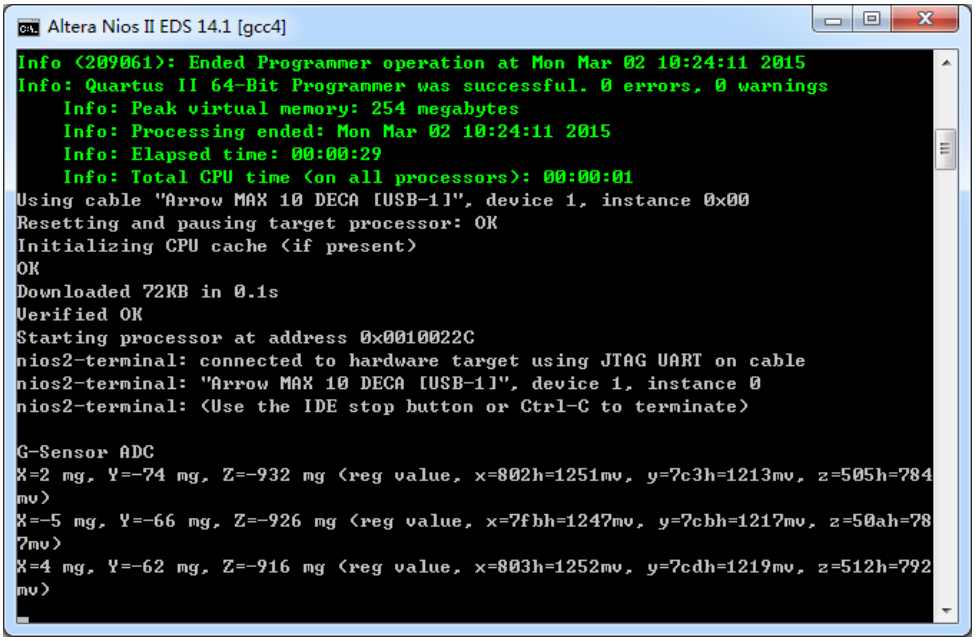


Figure 6-12 Display Progress and Result Information for the G-sensor Demonstration

6.6 SMA ADC

This demonstration shows how to use the built-in ADC to measure the signal voltage from the SMA connector onboard. The measured voltage is displayed in Nios II terminal.

Figure 6-13 shows the block diagram of SMA-ADC on the DECA board. The SMA input signal goes to the amplifier circuit and then to the build-in ADC in MAX 10. The output voltage and input voltage have the following relationship:

$$\text{Output Voltage} = (6.25\text{V} + \text{Input Voltage})/5$$

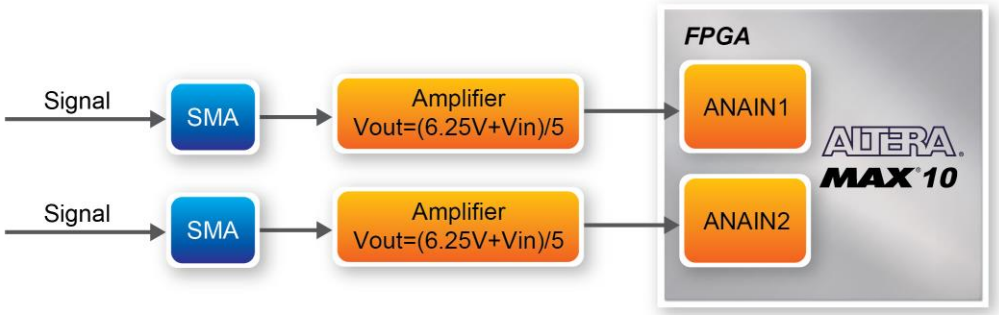


Figure 6-13 Block diagram of SMA ADC

The built-in ANAIN1 and ANAIN2 channels in MAX 10 on the DECA board receive signals coming out of amplifier circuits. **Figure 6-14** shows the settings of Altera Modular ADC core for the first SMA input in this demonstration. "Standard sequencer with Avalon-MM sample storage" is selected as the core to control the ADC. Channel 0 in the first ADC is enabled because the first SMA input is connected to the ANAIN1 pin.

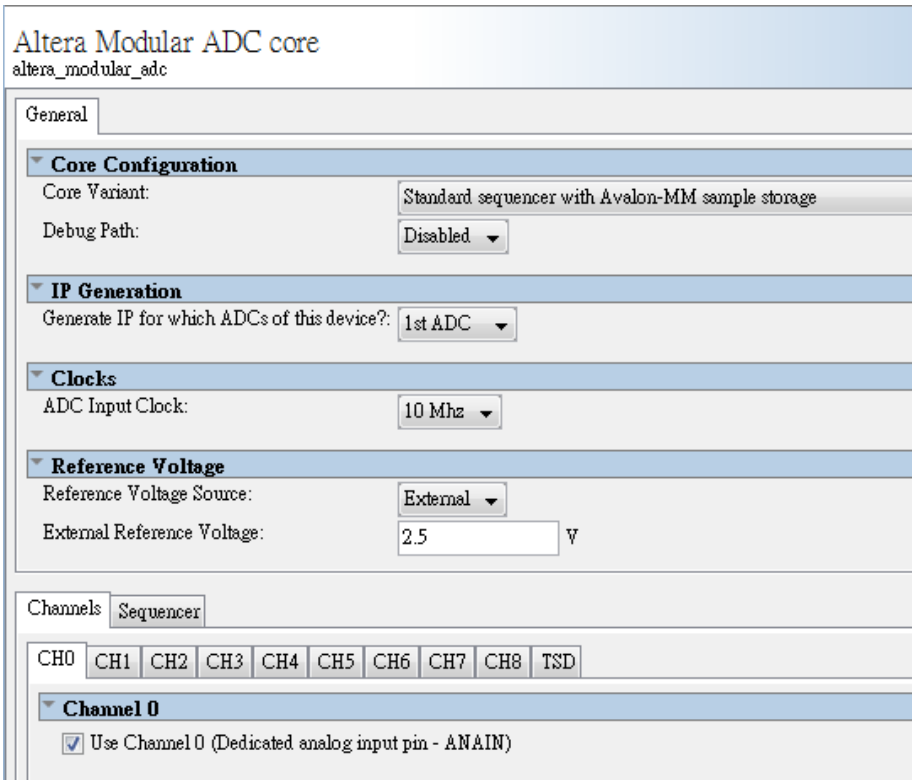


Figure 6-14 Settings of Altera Modular ADC Core

The macro IOWR in Nios II program is used to access the controller's register file. The following statements enable the ADC translation continuously.

```
IOWR(MODULAR_ADC_0_SEQUENCER_CSR_BASE, 0x00, 0x00); // Continued ADC
IOWR(MODULAR_ADC_0_SEQUENCER_CSR_BASE, 0x00, 0x01); // run

IOWR(MODULAR_ADC_1_SEQUENCER_CSR_BASE, 0x00, 0x00); // Continued ADC
IOWR(MODULAR_ADC_1_SEQUENCER_CSR_BASE, 0x00, 0x01); // run
```

The following statements reads the ADC value.

```
value_0 = IORD(MODULAR_ADC_0_SAMPLE_STORE_CSR_BASE, Slot) & 0xFFF; // 12-bit data
value_1 = IORD(MODULAR_ADC_1_SAMPLE_STORE_CSR_BASE, Slot) & 0xFFF; // 12-bit data
```

The following statements calculate the input and output voltage values.

```
Vo_0 = (float)value_0/4095.0*2.5; // 0~4095 mapping to 0~2.5V
Vin_0 = 5.0* Vo_0 - 6.25;

Vo_1 = (float)value_1/4095.0*2.5; // 0~4095 mapping to 0~2.5V
Vin_1 = 5.0* Vo_1 - 6.25;
```

■ Demonstration Source Code

- Project directory: SMA_ADC_NIOS
- Bit stream: DECA.sof

■ Demonstration Batch File

- Demo batch file folder: SMA_ADC_NIOS\demo_batch
- Batch file: test.bat
- FPGA configure file: DECA.sof
- Nios II program: adc_demo.elf

■ 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 (J10) on the DECA board to the host PC.

- Feed signals into the SMA connectors (J3 and J5) on the DECA board.
- Power on the DECA board.
- Execute the demo batch file "test.bat" under the batch file folder SMA_ADC_NIOS\demo_batch.
- Nios II terminal will display the voltage value of signals coming from the SMA connectors, as shown in **Figure 6-15.**

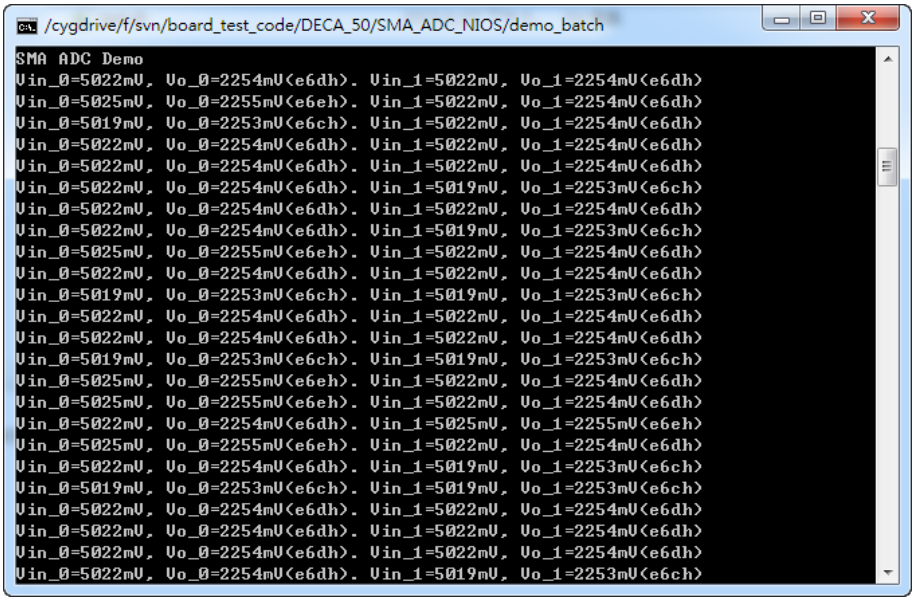


Figure 6-15 Screenshot of SMA-ADC demo

6.7 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 512 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 125 MHz clock, and the Nios II program is running in the on-chip memory.

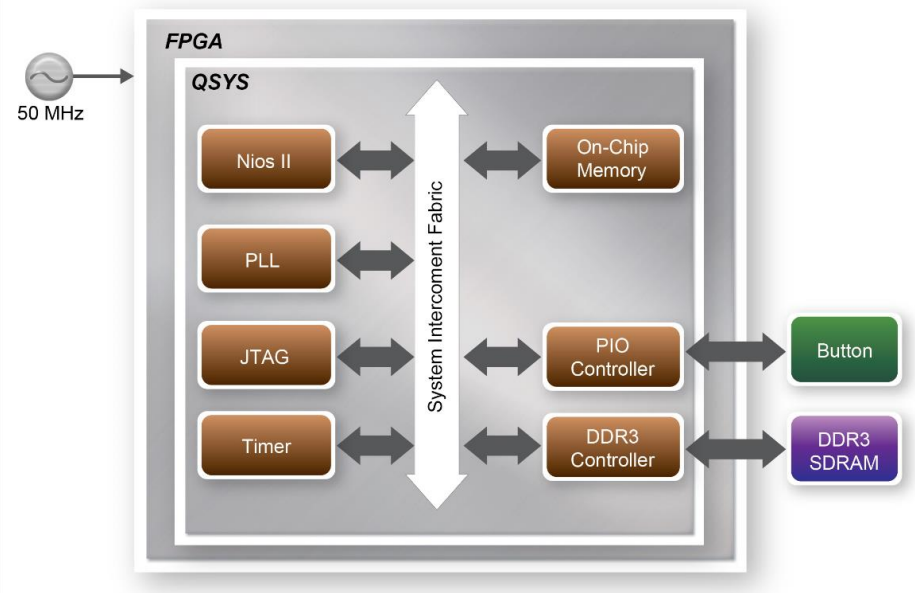


Figure 6-16 Block diagram of the DDR3 Basic Demonstration

The system flow is controlled by a Nios II program. First, the Nios II program writes test patterns into the whole 512 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, users need to perform the four major steps:

1. Create correct pin assignments for DDR3.
2. Setup correct parameters in DDR3 controller dialog.
3. Perform "Analysis and Synthesis" by clicking Quartus menu: Process→Start→Start Analysis & Synthesis.
4. 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: DECA_DDR3_Nios_Test
- Nios II Eclipse: *DECA_DDR3_Nios_Test\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:
DECA_DDR3_Nios_Test\demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaster II : test.bat, test.sh
- FPGA Configure File : DECA.*sof*
- Nios II Program: DECA_DEMO.*elf*

■ Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Power on the DECA board.
- Use USB cable to connect PC and the DECA board (J10) and install USB Blaster driver if necessary.
- Execute the demo batch file "*test.bat*" for USB-Blaster II under the batch file folder, *DECA_DDR3_Nios_Test\demo_batch*
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- The program will display progressing and result information, as shown in **Figure 5-17**.

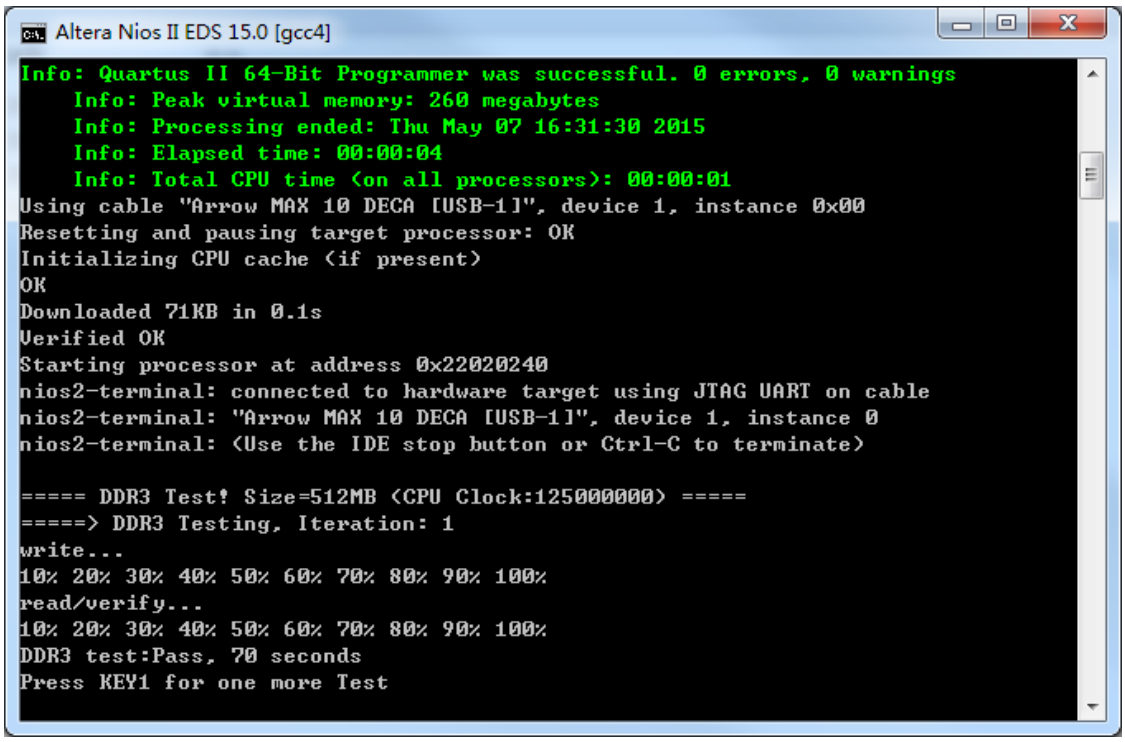


Figure 6-17 Display Progress and Result Information for the DDR3 Demonstration

Chapter 7

Advanced NIOS Based Example Codes

Several advanced NIOS based examples for you to try them on the DECA board. All of the NIOS based examples can be found in the System CD under the folder Demonstrations. You are free to use them or modify these examples in your future design!

7.1 HDMI Video/Audio TX

If you don't want to use RTL to develop HDMI Transmitter, you can try C-code to develop HDMI Transmitter, we introduce a reference design for programming the HDMI transmitter to generate video pattern and audio sound. The entire reference is composed into 2 parts – RTL based hardware controller and C-code main program running on NIOS II processor. A set of pre-built video patterns will be sent out through the HDMI interface and presented on the LCD monitor as the user launches the provided executable binaries. The design incorporates certain activities that user could interact with the on-board HDMI TX transmitter.

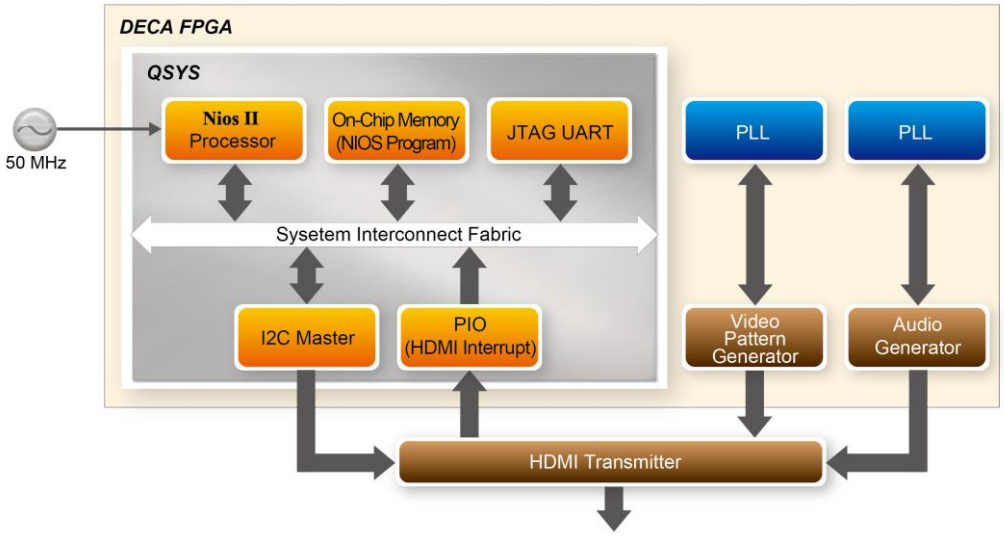


Figure 7-1 Block Diagram of the HDMI TX Demonstration

Figure 7-1 shows the system block diagram of this reference design. Our main program is NIOS Program, and NIOS program will be stored at on-chip memory of FPGA. I2C Master formed by 2 PIO signals of (SCL and SDA) is used to configure HDMI Transmitter, proper configuration is necessary before HDMI Transmitter successfully occupy. PLL and Audio Generator are designed to transmit audio pattern to HDMI-transmitter, audio transmitting interface is I2S in this demo. A PIO (HDMI Interrupt) is used to receive the interrupt signal from HDMI Transmitter. JTAG UART with Avalon interface implements a method to communicate serial character streams between a host PC and Qsys on FPGA, users can dump information out through JTAG UART. The default resolution of video pattern is FULL HD generator (1920*1080p), and it can be modified by adjusting the parameter of PLL and video pattern generator. The module “Video Pattern Generator” copes with generating video patterns to be presented on the LCD monitor. The pattern is composed in the way of 24-bit RGB 4:4:4 (RGB888 per color pixel without sub-sampling) color encoding, which corresponds to the parallel encoding format defined in Table 7-1 of the "ADV7513 Hardware User's Guide," as shown below.

Table 7-1 Build-in Display Modes of the HDMI TX Demonstration

Pixel Data [23:0]																	
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R[7:0]						G[7:0]						B[7:0]					

■ Auto Hot Plug Detection

The demonstration implements an interrupt-driven hot-plug detection mechanism which will automatically power on the transmitter chip when the HDMI cable is plugged into the development board and the LCD monitor is powered.

If the HDMI cable is already plugged into the on-board HDMI connector before powering up the development board, the monitor-sense signal will trigger an interrupt to power up the HDMI transmitter.

When the HDMI transmitter is powered up, a sample video pattern will be displayed on the LCD monitor. If the cable is unplugged, the HDMI transmitter will power off automatically to reduce the power consumption.

If the LCD monitor didn't power up automatically when performing activities described above in this demonstration, users can try to completely switch off the power of the LCD monitor and then switch on again. Alternatively the user can try to unplug and then re-plug the HDMI cable and wait for a reasonable time before the LCD monitor to complete its initialization process and start to sync with the HDMI transmitter.

■ Optimization Level setting

Because our main program is stored at on-chip memory of FPGA, the program will occupy a lot of on-chip memory. So we need to modify the Optimization Level to be Size which allow developers to use memory more efficiently. **Figure 7-2** shows how to set the Optimization level.

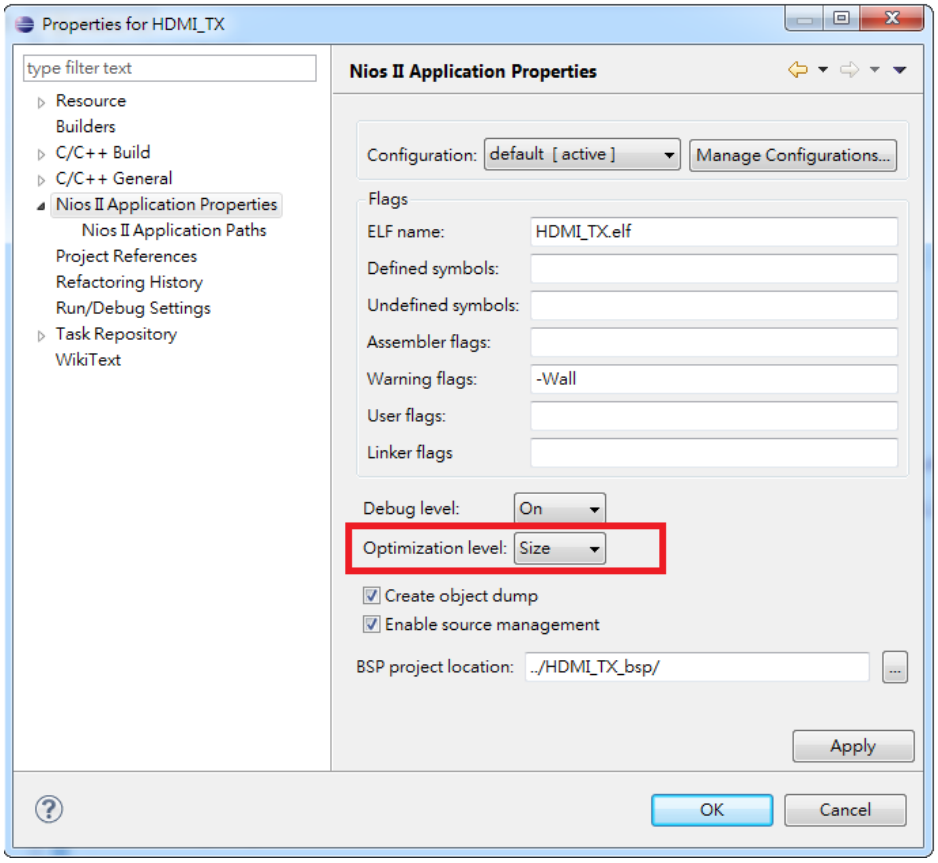


Figure 7-2 Setting Optimization level

■ Demonstration Source Code

- Project directory: HDMI_TX_NIOS
- Nios II Eclipse: HDMI_TX_NIOS\Software

■ Demonstration Batch File

- Demo batch file folder: Demonstrations\ HDMI_TX_NIOS
- Batch file: test.bat
- FPGA configure file: NIOS_HDMI_TX.sof
- Nios II program: HDMI_TX.elf

■ Demonstration Setup

- Make sure Quartus II and USB-Blaster II driver are installed on your PC.

- Connect the DECA board to the LCD monitor with the on-board HDMI connector via an HDMI cable.
- Power on the DECA board.
- Use File Manager to locate the "*HDMI_TX_NIOS\demo_batch*" folder. Launch the configuration and program download process by double clicking "*test.bat*" batch file. This will configure the FPGA, and download the demo application to the board and start its execution. A console terminal will be kept on the screen and the user can interact with the demo application through the console box. After it's done the screen should look like the one shown in **Figure 7-3**. A tiny command line interface is provided to interact with the on-board HDMI transmitter. Note that the commands are all case-sensitive.

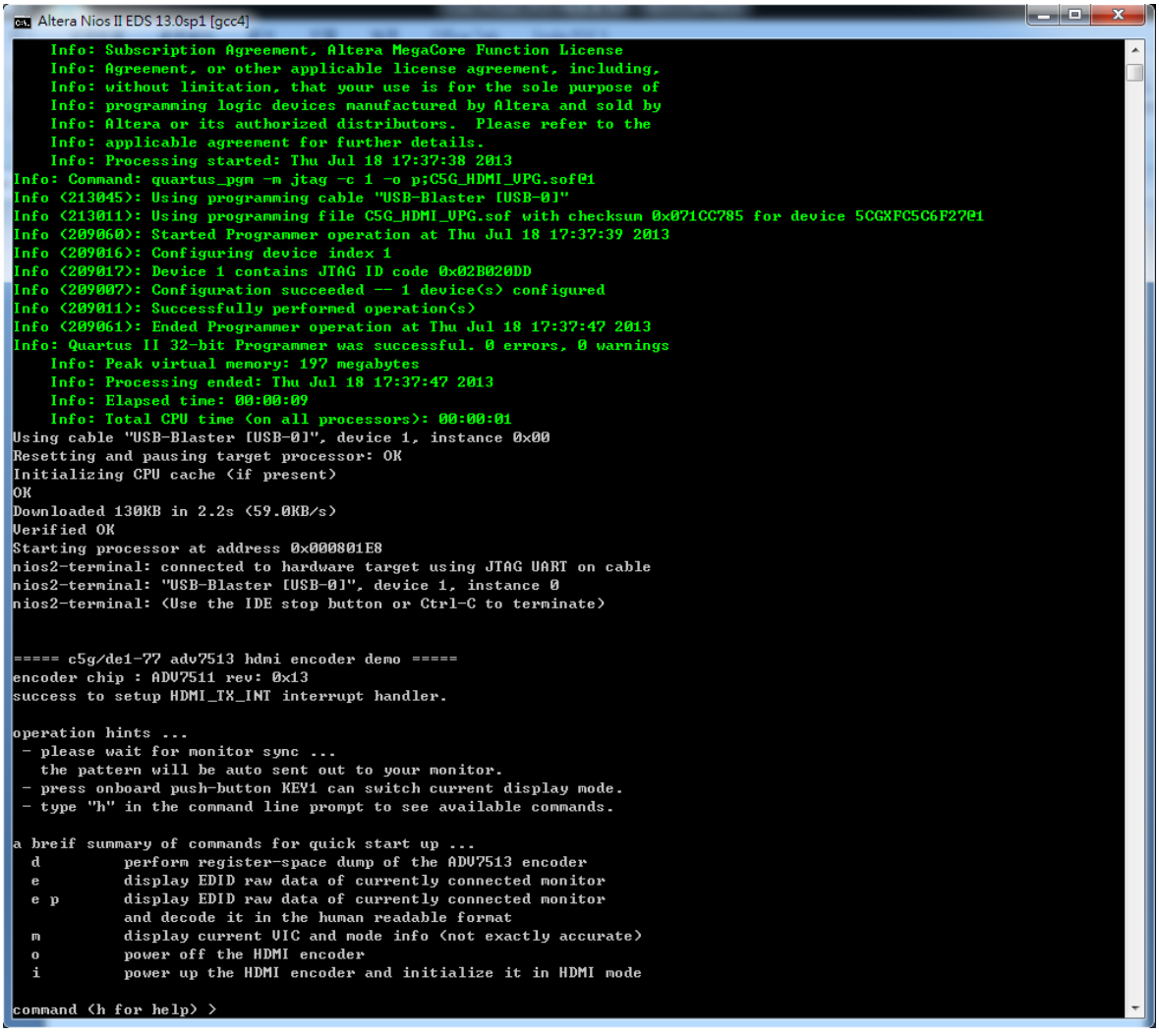


Figure 7-3 Launch the HDMI TX Demonstration using the "demo_batch" Folder

- Wait for a few seconds for the LCD monitor to power up itself. And you should see a pre-defined video pattern shown on the monitor, as shown in **Figure 7-4**.

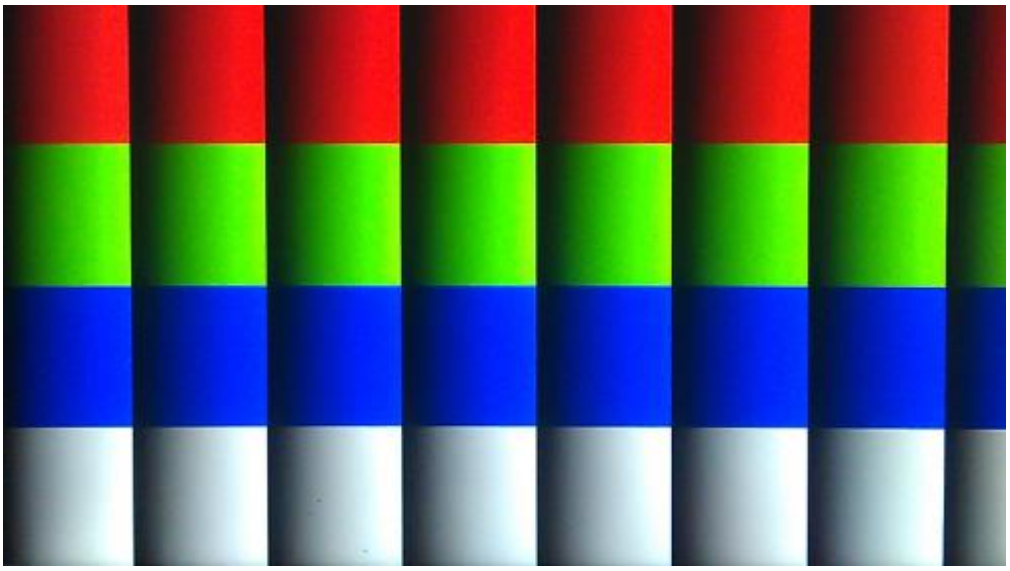


Figure 7-4 The Video pattern used in the HDMI TX Demonstration

The demonstration involves certain activities that user could interact with the on-board HDMI transmitter.

7.2 Gesture Light Sensor

DECA is able to provide advanced human-machine interaction by gesture detection through the Proximity Sensing of 3 irLED (Infrared Light-Emitting Diode) and the computing chip (Si1143).
First we introduces a reference design for programming the irLED. The Si1143 is an active optical reflectance proximity detector and ambient light sensor whose operational state is controlled through registers accessible through the interrupt and I2C interface.

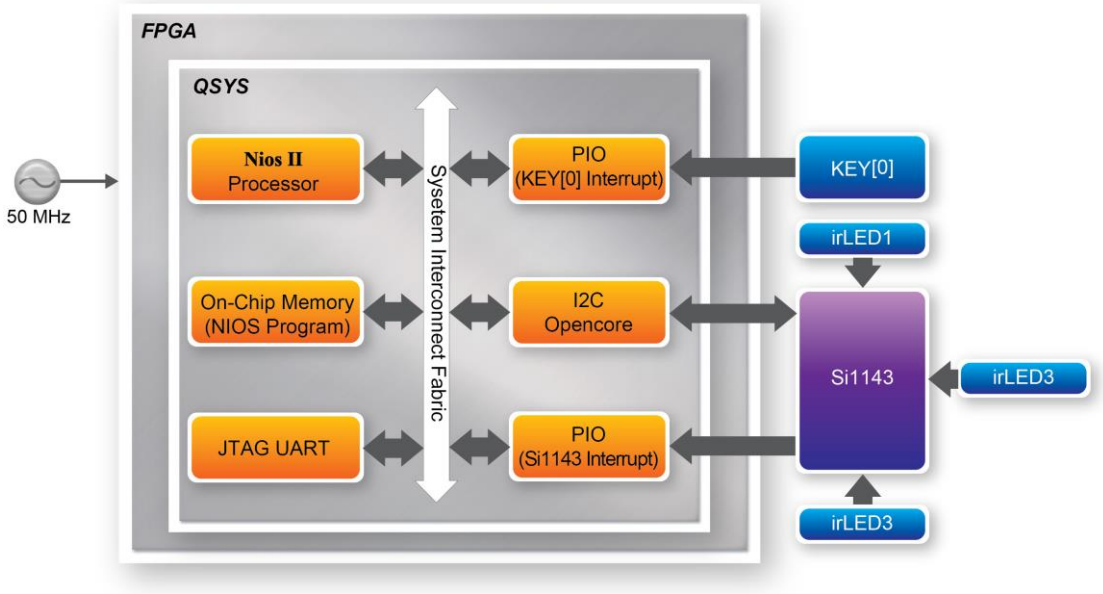


Figure 7-5 Block Diagram of the Si1143 Demonstration

Figure 7-5 shows the system block diagram of this reference design. The main program is NIOS Program running at on-chip memory located in FPGA. The I2C Opencore IP controller is used by NIOS to communicate with the Si1143 chip. There is a relevant C-code library in NIOS program to communicate with "I2C Opencore" controller. When we using I2C Opencore API, we need to make sure to shift one bit left from the original 7-bit I2C Slave to become 8-bit address value (ex: 0x01->0x02). JTAG UART module is used to let user dump information. Si1143 will send out interrupt signal in specific situation, so a PIO is needed to receive the interrupt signal from Si1143. Our demo shows 2 types of operations by press KEY[0] on DECA board to switch operation type.

■ Operation in type 1

Our default demo is Type1, running NIOS will execute PS_FORCE command to renew the data in Proximity Sensing (PS) Register of irLED. Then NIOS will read PS Register and display on NIOS Terminal, the output data indicate the strength of Proximity Sensing of irLED as shown in Figure 7-6.

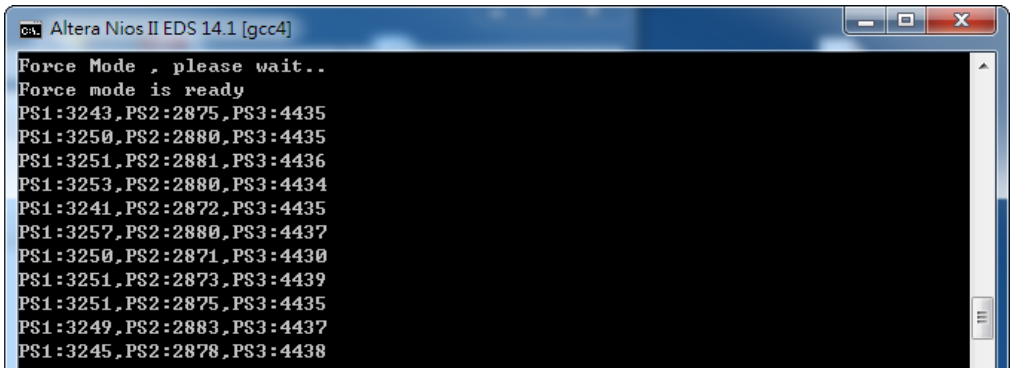


Figure 7-6 PS data on NIOS terminal

■ Operation in type 2

User has to switch from Type 1 to Type 2 by press KEY[0] on the DECA board. In Type 2 demo, Si1143 will read and update PS Register automatically then send out interrupt signal to NIOS when done reading and updating data. We can apply this updating characteristic of timing difference and data changes of Proximity Sensing (PS) to achieve gesture detection. As shown in **Figure 7-7**, indicates the gesture detection data in time domain of different gesture swipe. For more details, please refer to “AN498: Designer's Guide for the Si114x” or “Si114xRev1_3” or “AN580 Method 2: Phase-based Gesture Sensing.”

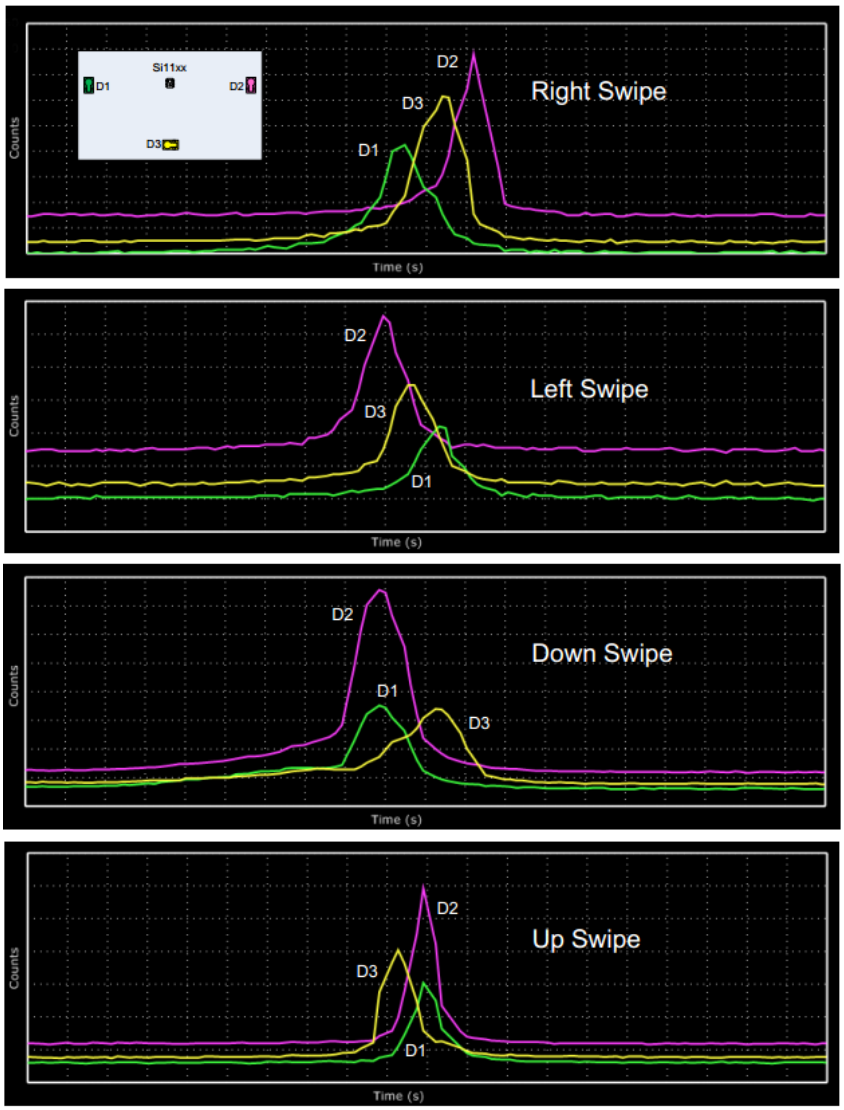


Figure 7-7 Gesture detection data in time domain

■ Demonstration Source Code

- Project directory: Gesture_Light_Sensor_NIOS
- Bit stream: Light_Sensor.sof

■ Demonstration Batch File

- Demo batch file folder: Gesture_Light_Sensor_NIOS\demo_batch
- Batch File: test.bat, test.sh
- FPGA Configure File: Light_Sensor.sof
- NIOS Program: Gesture_Test.elf

■ Demonstration Setup

- Make sure Quartus II and USB-Blaster II driver are installed on your PC.
- Connect the USB cable to the USB Blaster II connector(J10) on the DECA board and host PC.
- Power on the DECA board.
- Execute the demo batch file "test.bat" under the batch file folder, Gesture_Light_Sensor_NIOS \demo_batch.
- In Forced Conversion Mode terminal will display the PS1(proximity sensing) data, PS2 data and PS3 data as shown in **Figure 7-8**.
- After press KEY[0] to switch to Type 2, if the gesture detection is success the result will print on NIOS Terminal for corresponding swipe as shown in **Figure 7-8**, "Left" will be printed out.

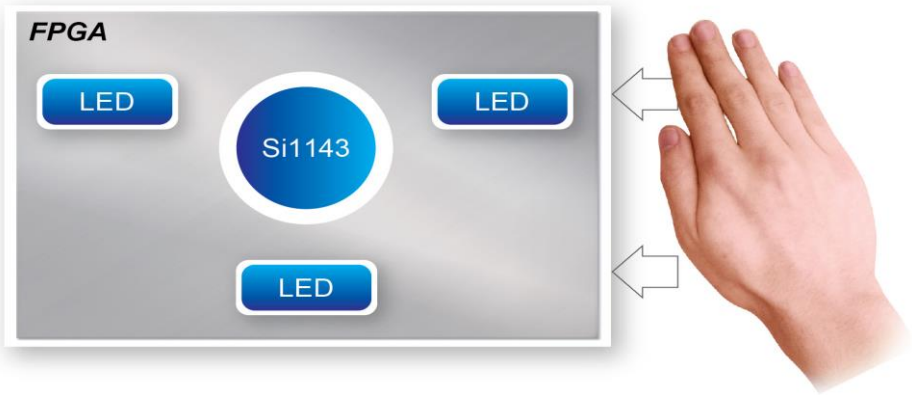


Figure 7-8 "Left Swipe" demonstration

7.3 Ethernet Socket server

This design example shows 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 DECA board. The server can process continues to listen for commands on a TCP/IP port and operates the DECA 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 accompany Nios II.

Using this demo, we assume that you already have a basic knowledge of TCP/IP protocols.

As indicated in the block diagram in **Figure 7-9**, the Nios II processor is used to communicate with the Client via DP83620(MII interface)Ethernet Device.

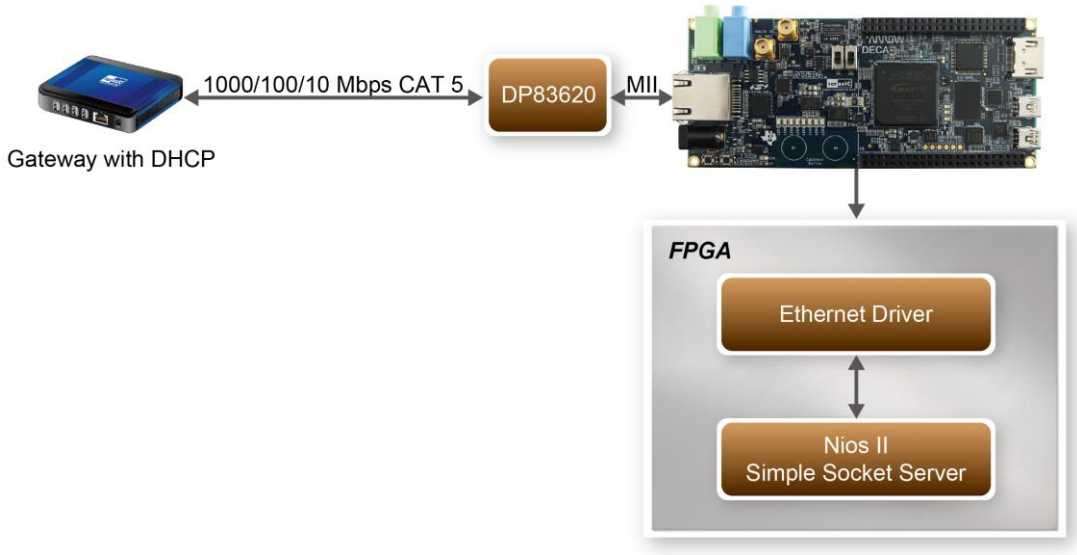


Figure 7-9 Block diagram of the demonstration

The following describes the related Qsys system. The Qsys system used in this demo contains Nios II processor, DDR3 memory, JTAG UART, timer, Triple-Speed Ethernet, Scatter-Gather DMA controller and other peripherals etc. In the configuration page of the Altera Triple-Speed Ethernet Controller, users need to set the MAC interface as MII as shown in **Figure 7-10**.

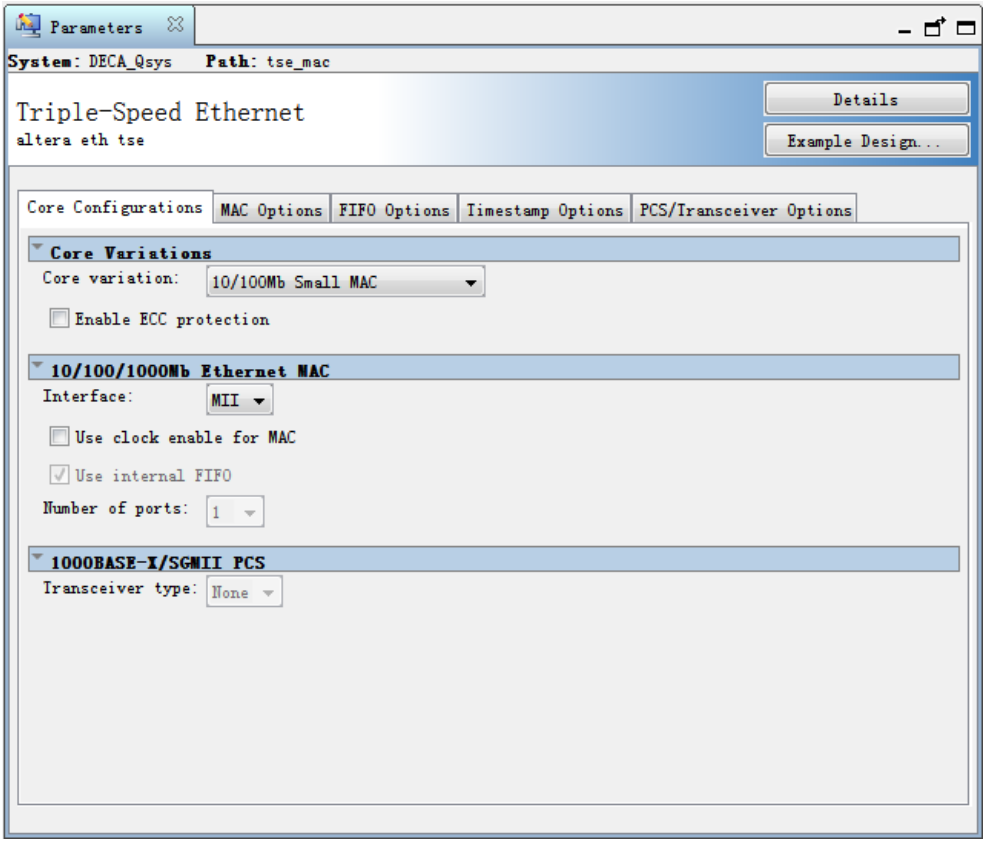


Figure 7-10 MII interface MAC Configuration

In the MAC Options tab (See **Figure 7-11**), users should set up proper values for the PHY chip DP83620. 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.

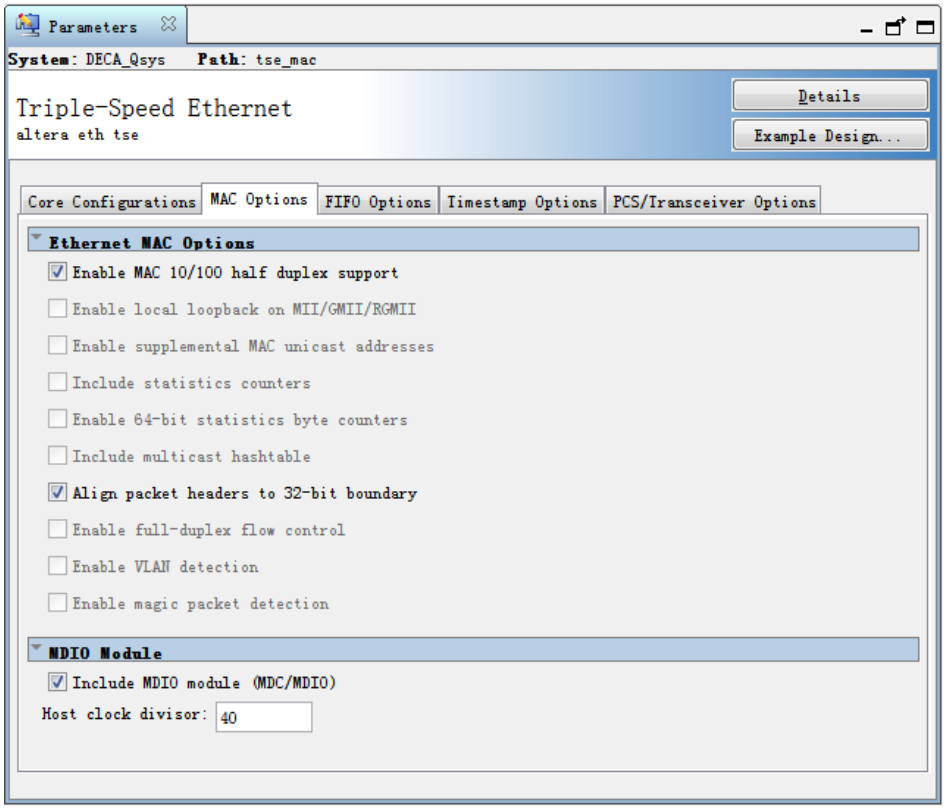


Figure 7-11 MAC Options Configuration

Once the Triple-Speed Ethernet IP configuration has been set and necessary hardware connections have been made as shown in **Figure 7-12**, click on generate.

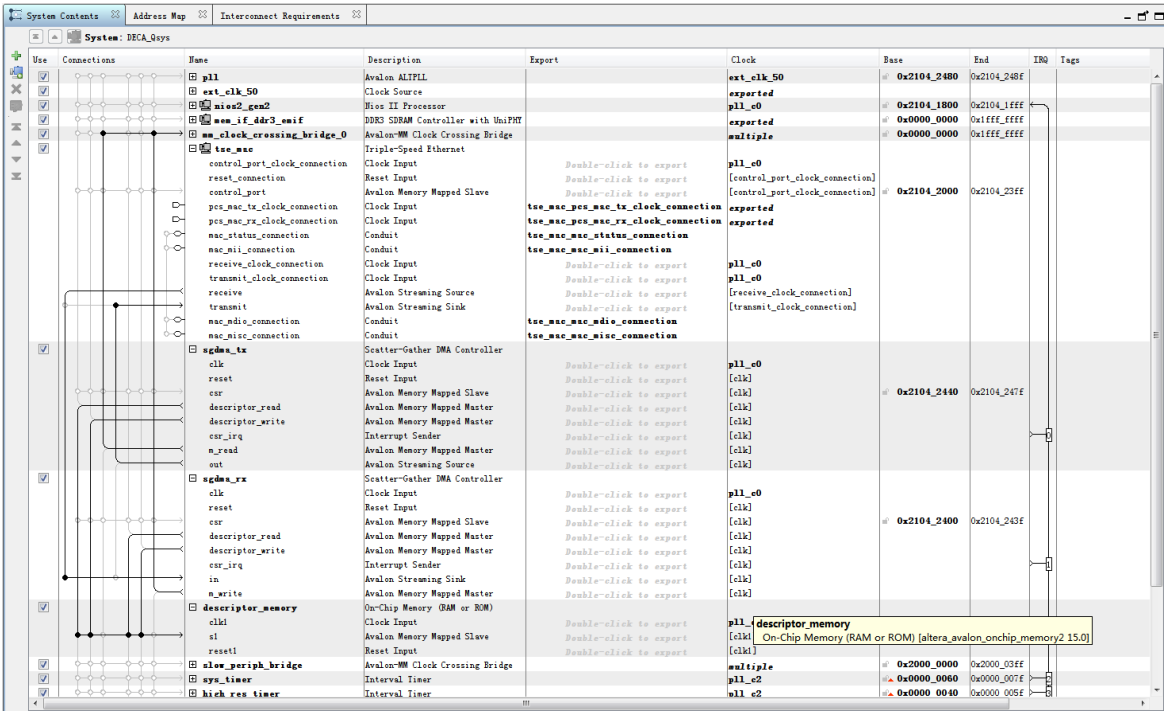


Figure 7-12 Qsys Builder

Figure 7-13 shows the connections for programmable 10/100Mbps Ethernet operation via MII.

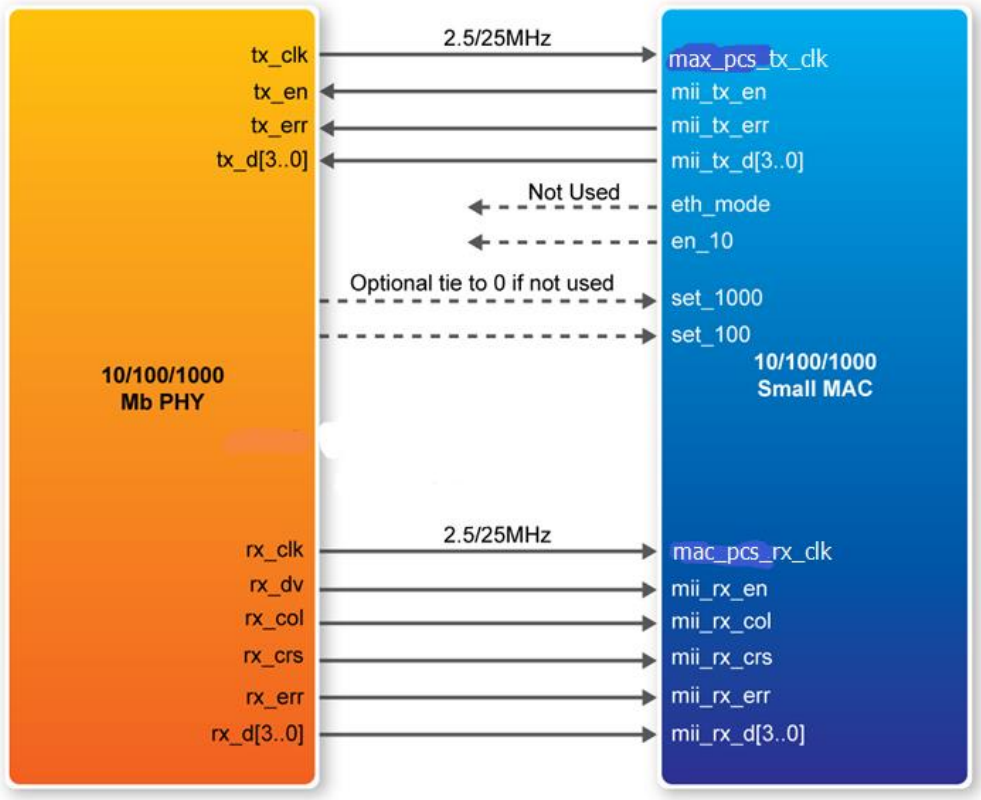


Figure 7-13 PHY connected to the MAC via MII

After the Qsys hardware project has been built, develop the Qsys software project, whose basic architecture is shown in **Figure 7-14**. The top block contains the Nios II processor and the necessary hardware to be implemented into the DECA 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 Socket Server. The NicheStack™ TCP/IP Stack software block provides networking services to the application block where it contains the tasks for Socket Server.

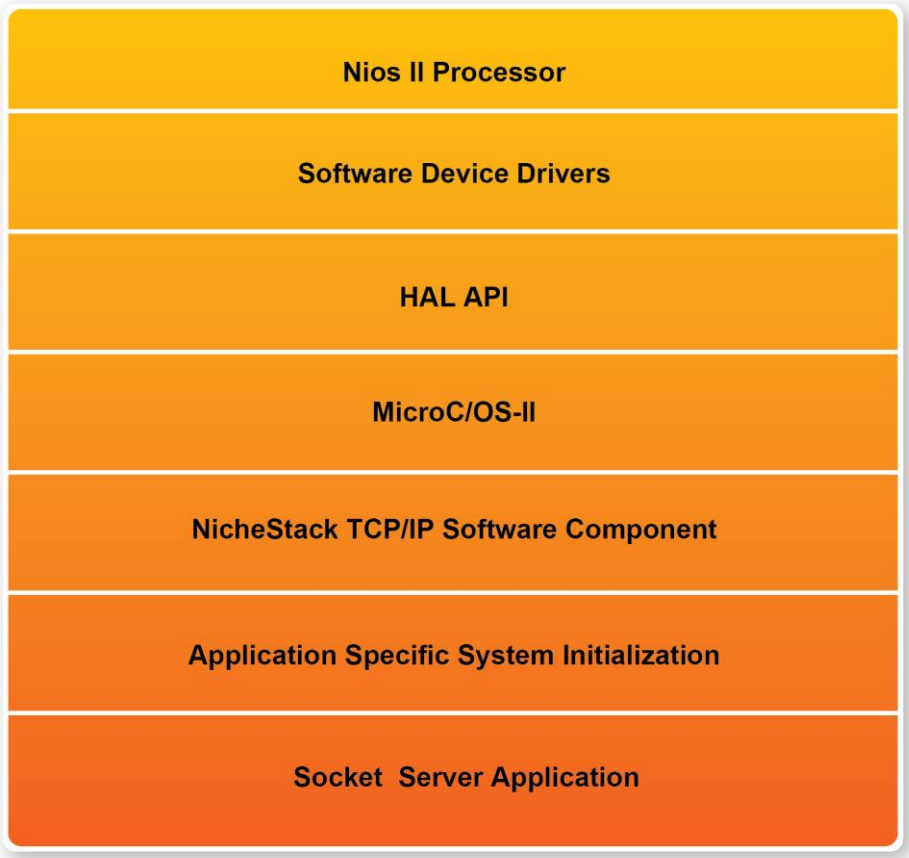


Figure 7-14 Nios II Software Routine Architecture

Finally, the detail descriptions for Software flow chart of the Socket Server program are listed in 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 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 finished, it will establish the link. Thirdly, 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.

Figure 7-15 describes this demo setup and connections on DECA .The Nios II processor is running NicheStack™ on the MicroC/OS-II RTOS.

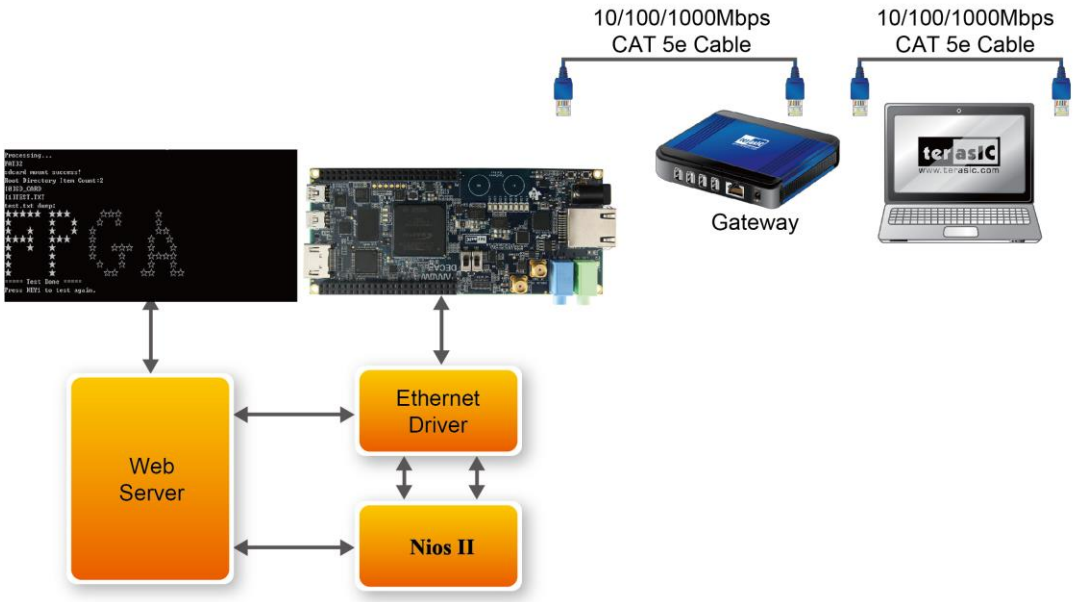


Figure 7-15 System Principle Diagram

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 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations/ DECA_socket_server

■ Demonstration Batch File

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

■ 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 (J10) on the DECA board to the host PC.
- Power on the DECA board.
- Execute the demo batch file "DECA_socket_server.bat" under the folder Demonstrations/ deca_socket_server /demo_batch , then the IP address and port number are assigned as shown below in **Figure 7-16**.

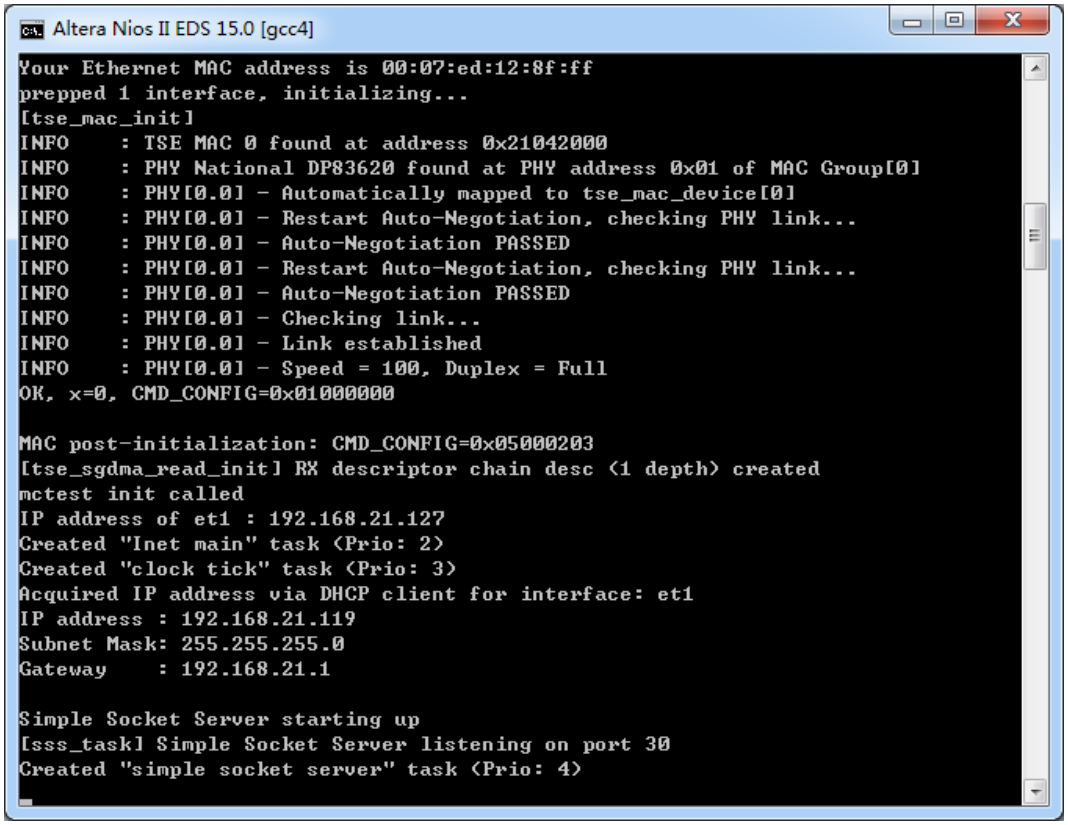


Figure 7-16 Simple Socket Server

- 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 7-17**.

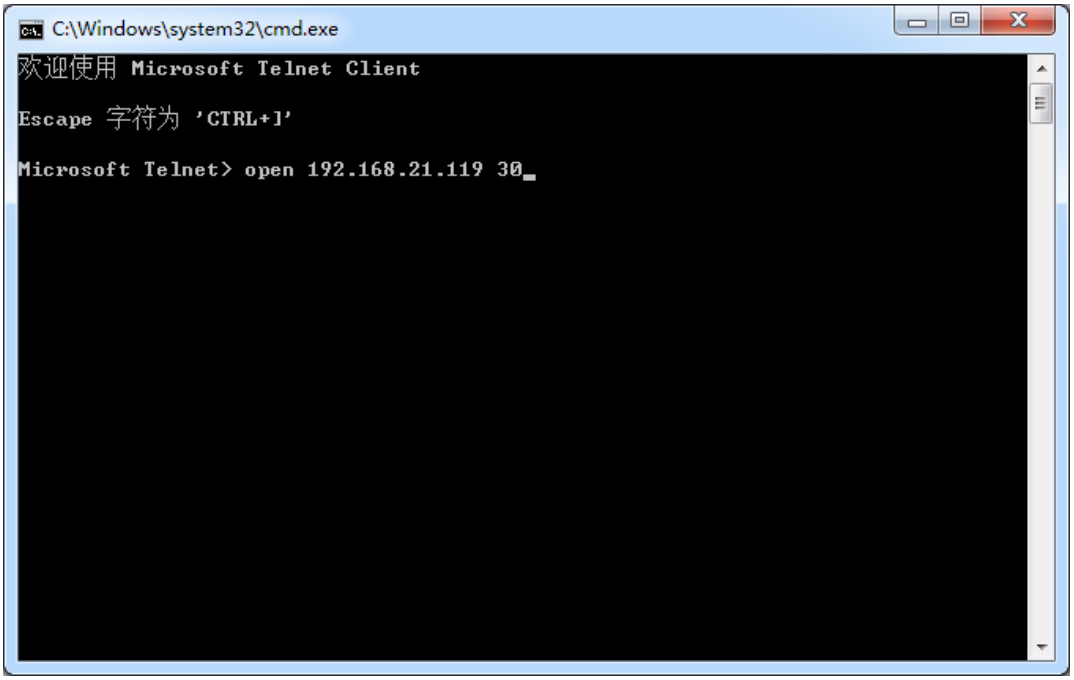


Figure 7-17 Telnet Client

- 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 LEDs (D0-D7) to toggle on or off on the DECA board as shown below in **Figure 7-18**.

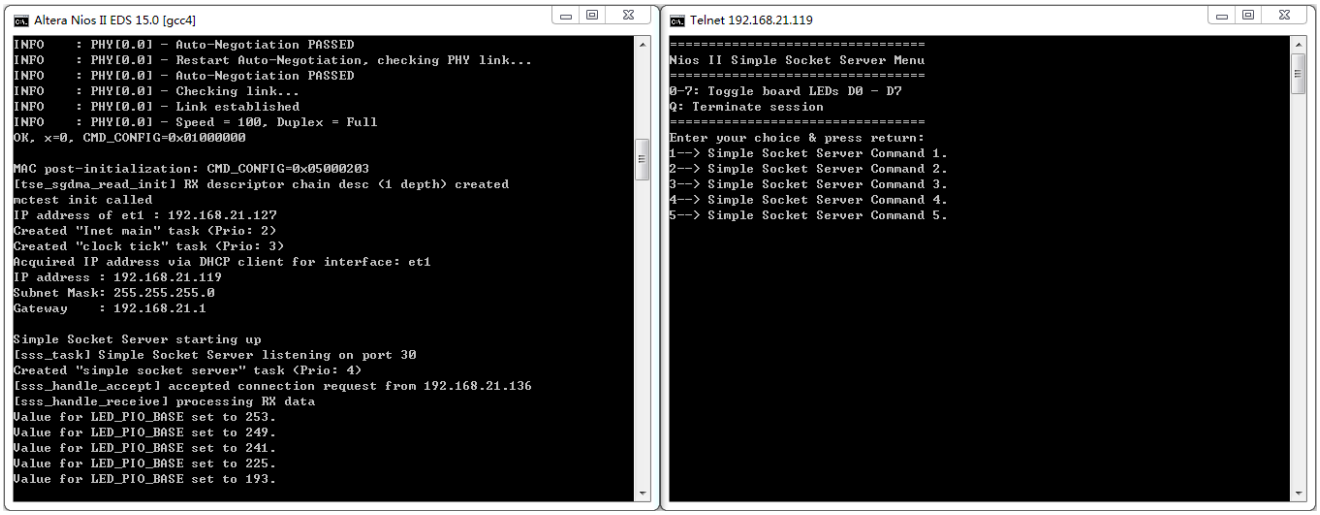


Figure 7-18 Display Progress and Result Information for the Socket Server Demonstration

7.4 Micro SD Card file system read

Many applications use a large external storage device, such as SD Card or CF Card to store data.

The DECA board provides the hardware and software needed for Micro SD Card access. In this demonstration, we will show you how to browse files stored in the root directory of a SD Card and how to read the file contents of a specific file. The Micro SD Card is required to be formatted as FAT File System in advance. Long file name is supported in this demonstration.

Figure 7-19 shows the hardware system block diagram of demonstration. In this demo, Power Mux (SD_SEL) is fixed at 0. When Power Mux is fixed at 0, Hi signal through SN74AVCA406L(Level Shift) will be raised to 3.3V, Low signal is not affected. The direction of CMD and DATA has to be concerned in the transmitting process of SN74AVCA406L in marco defined softare/tersaic_sdcard folder. The system requires a 50MHz clock provided by the board. Four PIO pins are connected to the Micro SD card socket. SD 4-bit Mode is used to access the Micro SD card hardware. The SD 4-bit protocol and FAT File System function are all implemented by Nios II software. The software is stored at on-chip memory.

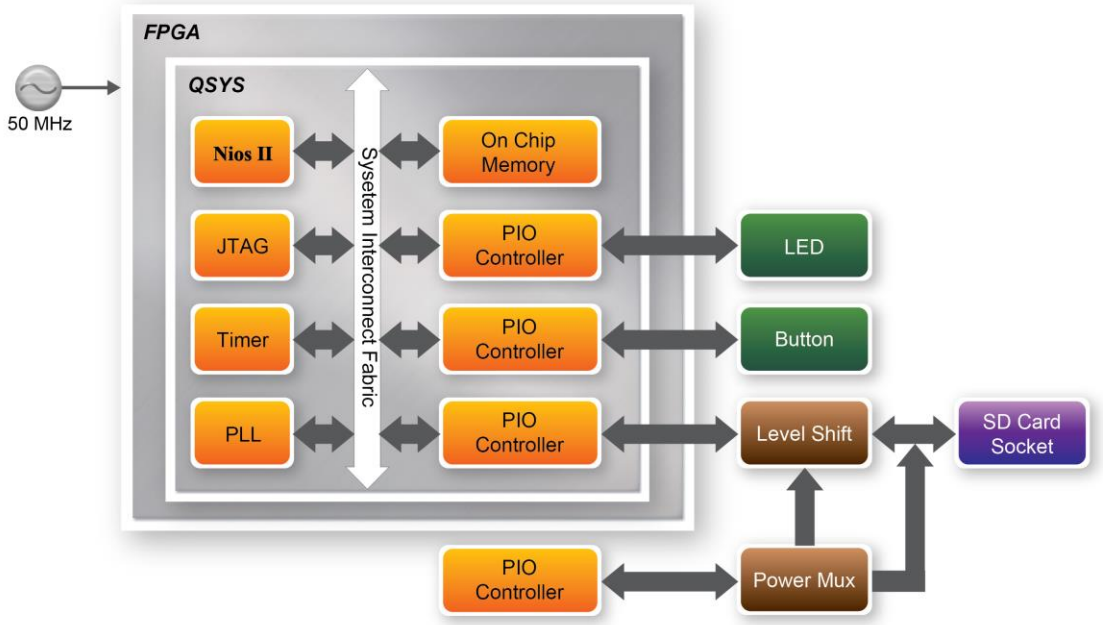


Figure 7-19 Block diagram of the Micro SD demonstration

Figure 7-20 shows the software stack of this demonstration. The Nios PIO block provides basic IO functions to access hardware directly. The functions are provided from Nios II system and the function prototype is defined in the header file <io.h>. The SD Card block implements 4-bit mode protocol for communication with SD Cards. The FAT File System block implements reading function for FAT16 and FAT 32 file systems. Long filename is supported. By calling the public FAT functions, users can browse files under the root directory of the Micro SD Card. Furthermore, users can open a specific file and read the contents from the file. The main block implements main control of this demonstration. When the program is executed, it detects whether an Micro SD Card is inserted. If an Micro SD Card is found, it will check whether the Micro SD Card is formatted as FAT file system. If so, it searches all files in the root directory of the FAT file system and displays

their names in the Nios II terminal. If a text file named “test.txt” is found, it will dump the file contents. If it successfully recognizes the FAT file system, it will turn off the LED. On the other hand, it will turn on the LED[3:0] if it fails to parse the FAT file system or if there is no SD card found in the SD Card socket of the DECA board. If users press KEY[1] on DECA board, the program will perform the above process again.

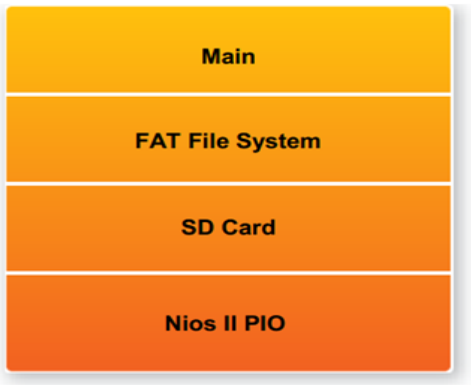


Figure 7-20 Software of micro SD demonstration

■ Demonstration Source Code

- Quartus Project directory: SDCARD
- Nios II Eclipse: SD_DEMO\Software

■ Demonstration Batch File

Demo Batch File Folder:

SDCARD \demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaster :
  DECA_SD_DEMO.bat
  DECA_SD_DEMO.sh

- FPGA Configure File : SDCARD.*sof*
- Nios II Program:SDCARD_DEMO.*elf*

■ Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Power on the DECA board.
- Connect USB Blaster to the DECA board and install USB Blaster driver if necessary.
- Execute the demo batch file "*DECA_SD_DEMO.bat*" for USB-Blaster II under the batch file folder, *SDCARD \demo_batch*
- After Nios II program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.
- Copy *SDCARD \demo_batch\test.txt* files to the root directory of the SD Card.
- Insert the Micro SD Card into the SD Card socket of DECA, as shown in **Figure 7-21.**

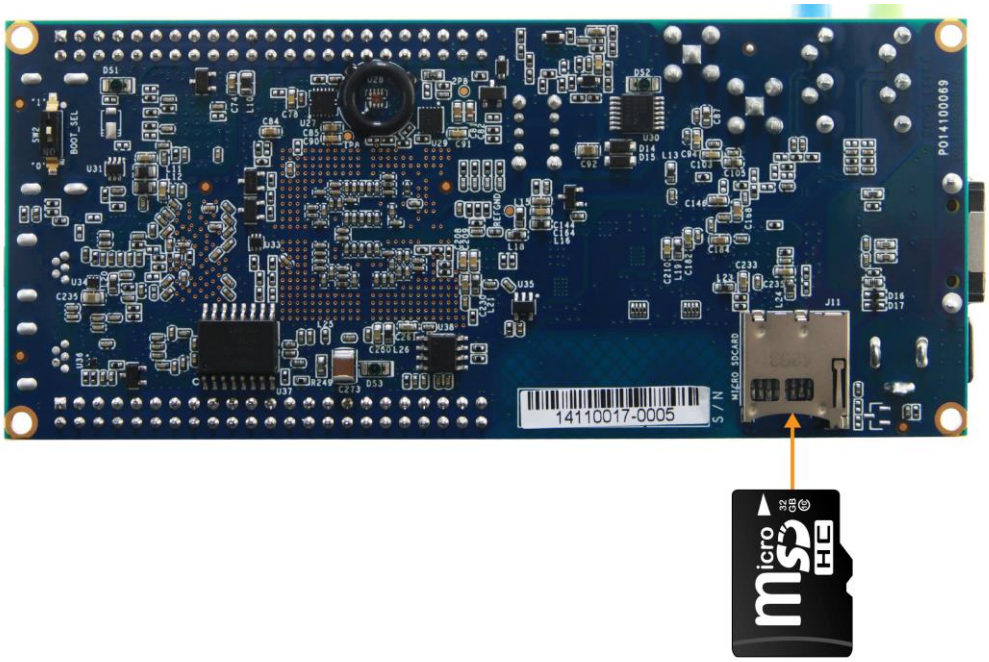


Figure 7-21 Insert the Micro SD card into DECA

- Press KEY[1] on the DECA board to start reading SD Card.
- The program will display SD Card information, as shown in **Figure 7-22.**

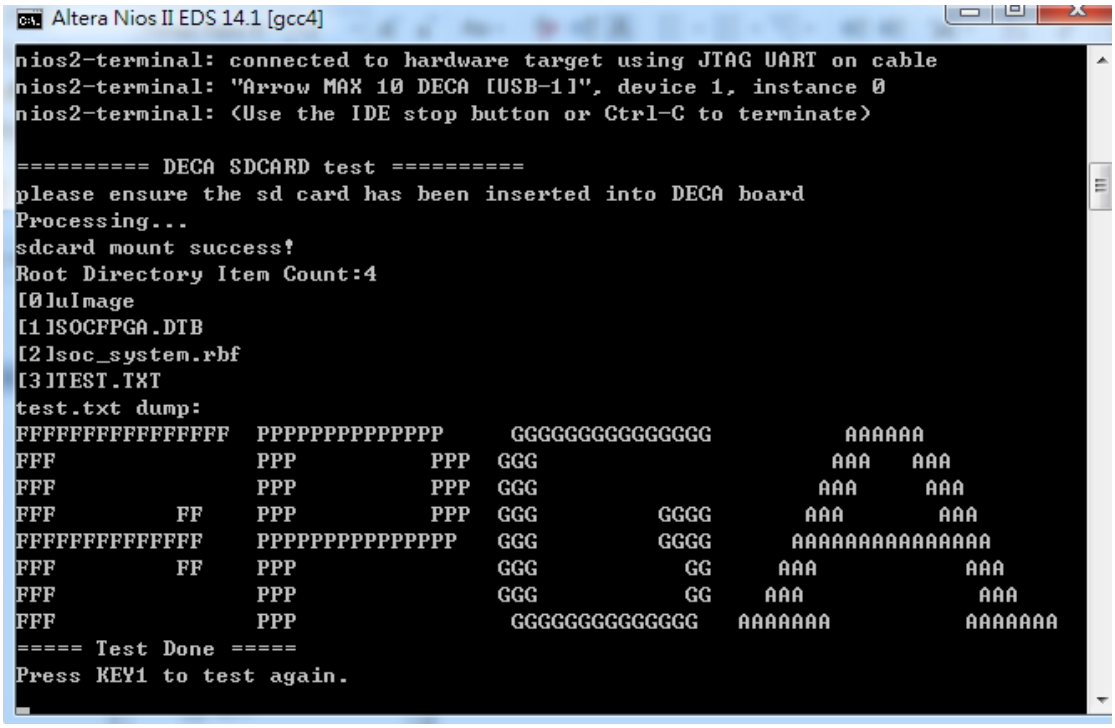


Figure 7-22 Running result of SD_CARD demo on DECA board

7.5 Audio

This demonstration shows how to implement an audio player using the DECA board with the built-in Audio CODEC chip. This demonstration is developed based on Qsys and Eclipse. SW0 is used to configure this audio system which specify playing source to be Line-in or Beep generation. **Figure 7-23** shows the block diagram of the Audio Player design. There are hardware and software parts in the block diagram. The software part stores the NiosII program in the on-chip memory. The software part is built by Eclipse written in the C programming language. The hardware part is built by Qsys under Quartus II. The hardware part includes all the other blocks.

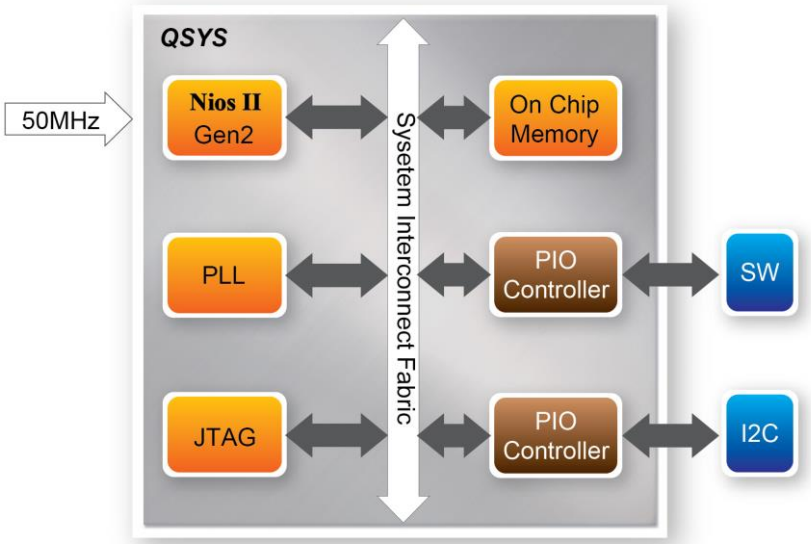


Figure 7-23 Block diagram of Audio

The audio chip is programmed through I2C protocol which is implemented in the C code. The I2C pins from audio chip are connected to Qsys System Interconnect Fabric through PIO controllers. In this example, the audio chip is configured in Master Mode. The audio interface is configured as I2Sand 16-bit mode. 49.152MHz clock generated by the PLL is connected to the MCLK pin of the audio chip.

■ Demonstration File Locations

- Hardware Project directory: DECA_Audio
- Bit stream used: DECA_Audio.sof
- Software Project directory: DECA_Audio \software

■ Demonstration Setup and Instructions

- Connect an audio source to the LINE-IN port of the DECA board.
- Connect a speaker or headset to LINE-OUT port on the DECA board.
- Load the bit stream into FPGA.
- Load the software execution file into FPGA.
- Configure audio with the slide switches as shown in **Table 7-2**.
- Push up SW0 on the DECA board to start audio playing from LINE-IN
- Push down SW0 on the DECA board to start audio playing from Beep Generation

Table 7-2 Slide switches usage for audio source

Slide Switches	*1 – UP Position*	*0 – DOWN Position*
SW0	Audio is from LINE-IN	Audio is from Beep Generation

Figure 7-24 illustrates the setup for this demonstration.

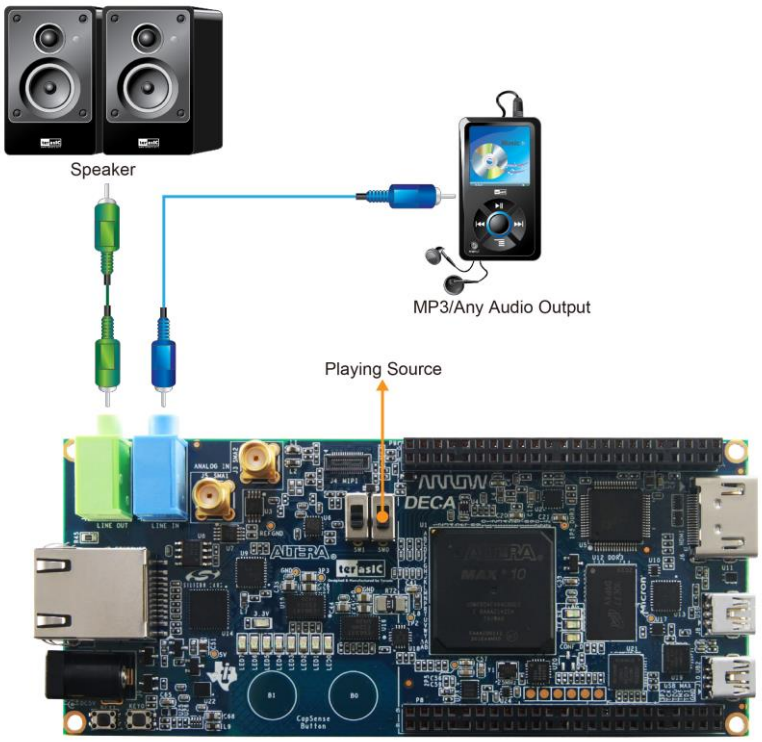


Figure 7-24 The setup for the Audio demonstration

7.6 USB Port Interface

This demonstration demonstrates USB Port Interface application on DECA board using the SLS USB20SR IP with ULPI interface. In the demo the host PC can perform single word or multiple words read and write operation on the DECA board via USB interface.

Figure 7-25 shows the block diagram of the USB Port Interface system. the SLS USB20SR has a ULPI interface, which can connect with the ULPI PHY(TUSB1210), and the PC can communication with the IP via it. The IP's Avalon Bus compliant enables it active as an Avalon Slave in the Qsys.
Following are the USB20SR Device IP core features:

- USB 2.0 USB IF high-speed certified

- Supports both High Speed (480 Mbps) and Full Speed (12 Mbps)
- High speed or Full speed operation selection through Software configuration
- ULPI interface Supports CONTROL,IN and OUT endpoints

Note: for more detailed information about the USB20SR IP, Please visit the SLS website.

In this demo the Nios II embedded CPU initial and configure the USB20SR IP core as a USB device for PC as **Figure 7-25** shows**.** The device speed is Full, and the ULPI interface has 2 endpoints, which are IN (ID 1)and OUT(ID 2) with a max transmission packet size of 64 bytes. You can find the Vendor ID is 0x1772 and the Product ID is 0x0002.

The demo also supports DMA access between USB20SR and the on-chip memory, which can increase the system efficiency and alleviate the Nios II burden on memory access. And the DMA access is bidirectional, which can perform write and read operation on the on-chip memory. The on-chip memory address range for DMA access is 0x100BDCD to 0x1020F57.

User can refer to the documents from the SLS website or IP installation package. The documents include the **Qsys Tutorial** for the SOPC system design, the **User Guide** for IP introduction, **Windows 32 API User Guide** for software application development on the top of USB device driver, and the **Nios II HAL API User Guide for IP** configuration on the DECA board.

The host Application and drivers in the demo are pre-built, and the user can directly execute and load them. The recommend OS environment is 32-bits Win7 or XP.

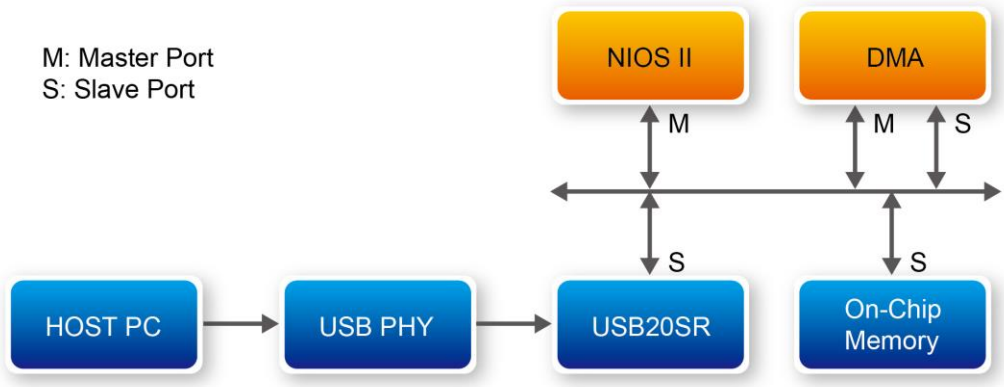


Figure 7-25 Port Interface System Block Diagram

```
          ------>Device Information<------
English product name: "USB2.0"

ConnectionStatus:
Current Config Value:              0x01  -> Device Bus Speed: Full
Device Address:                    0x04
Open Pipes:                           2

          ===>Endpoint Descriptor<===
bLength:                           0x07
bDescriptorType:                   0x05
bEndpointAddress:                  0x81  -> Direction: IN - EndpointID: 1
bmAttributes:                      0x02  -> Bulk Transfer Type
wMaxPacketSize:                  0x0040 = 0x40 bytes
bInterval:                         0x01

          ===>Endpoint Descriptor<===
bLength:                           0x07
bDescriptorType:                   0x05
bEndpointAddress:                  0x02  -> Direction: OUT - EndpointID: 2
bmAttributes:                      0x02  -> Bulk Transfer Type
wMaxPacketSize:                  0x0040 = 0x40 bytes
bInterval:                         0x01

          ===>Device Descriptor<===
bLength:                           0x12
bDescriptorType:                   0x01
bcdUSB:                          0x0110
bDeviceClass:                      0xFF  -> This is a Vendor Specific Device
bDeviceSubClass:                   0x00
bDeviceProtocol:                   0xFF
bMaxPacketSize0:                   0x40 = (64) Bytes
idVendor:                        0x1772idProduct:                        0x0002
bcdDevice:                       0x0010
iManufacturer:                     0x01
     English (United States)  "SLS"
iProduct:                          0x02
     English (United States)  "USB2.0"
iSerialNumber:                     0x03
     English (United States)  "000000000001"
bNumConfigurations:                0x01
```

Figure 7-26 USB Device Information

■ Design Tools

- Quartus II v15.0 64-bit

■ Demonstration Source Code

- Project directory: Demonstrations/deca_usb_ulpi

Note: the USB20SR IP license is Eval version, so the *.sof file is time-limited, keep the usb blaster II cable connected during running.

■ Demonstration Batch File

- Demo batch file folder: Demonstrations/deca_usb_ulpi/demo_batch/
- Batch file: test.bat
- FPGA configure file: DECA_golden_top_time_limited.sof
- Driver folder: /deca_usb_ulpi/demo_batch/app_driver/driver/windows/
- Driver file: slsusb.Inf
- Application file folder: /deca_usb_ulpi/demo_batch/app_driver/utilities/portinterface/
- Application file: PortInterface.exe

■ 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 (J10) on the DECA board to the host PC.
- Power on the DECA board.
- Execute the demo batch file "test.bat" under the folder Demonstrations/deca_usb_ulpi/demo_batch/
- Input "q" to quit the info for USB20SR IP as **Figure 7-27** shows.

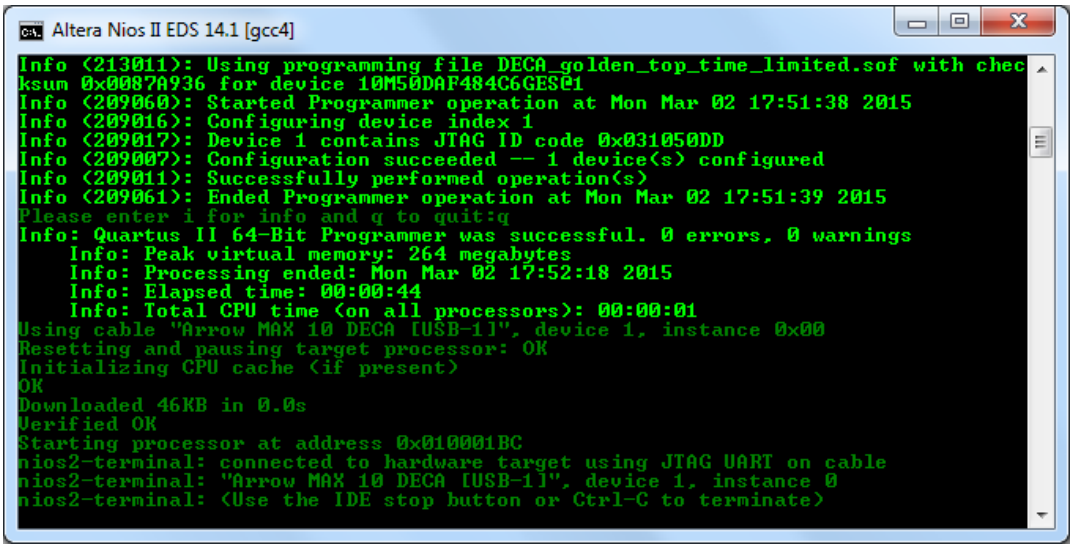


Figure 7-27 Programming the SOF file

- Connect another USB cable from the USB port (J8) on the DECA board to the host PC.
- Open the device manager and install the driver file for the unknown USB device.
- After successfully installed the driver file, the USB2.0 device will be show as **Figure 7-28.**

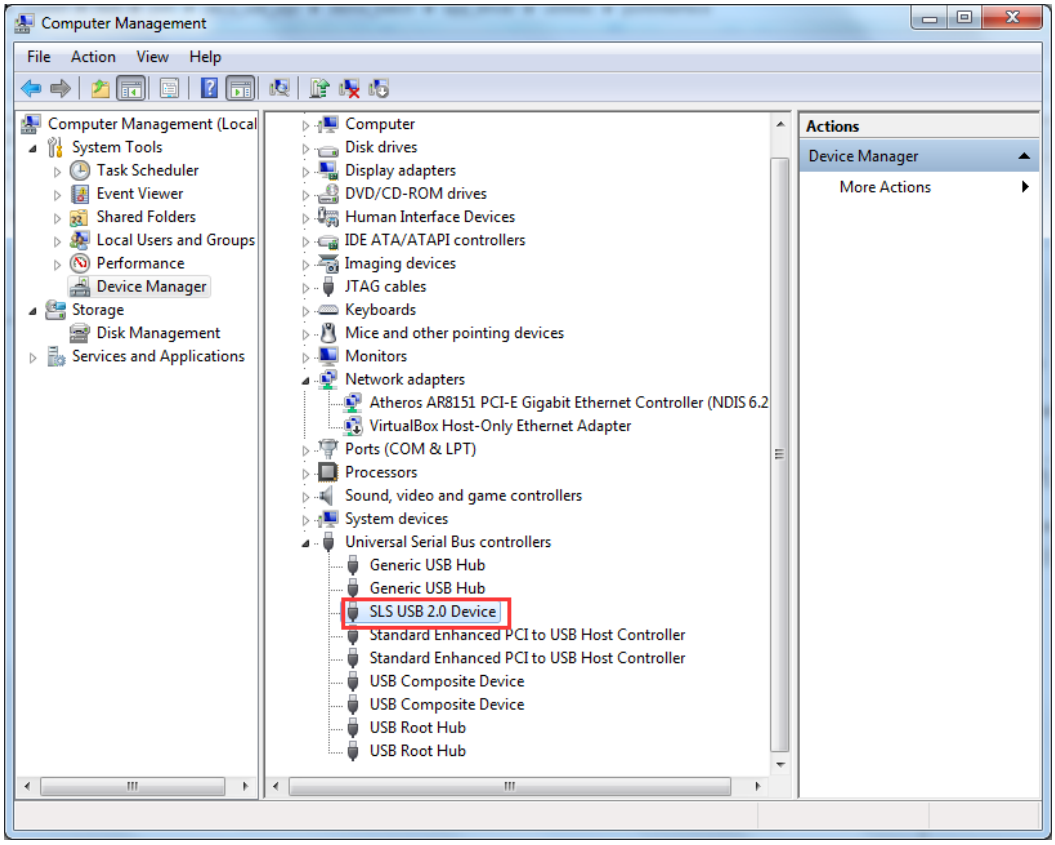


Figure 7-28 SLS USB 2.0 Device

- Execute the application file to make the Word or File Write and Read Verify operation as **Figure 7-29** shows.

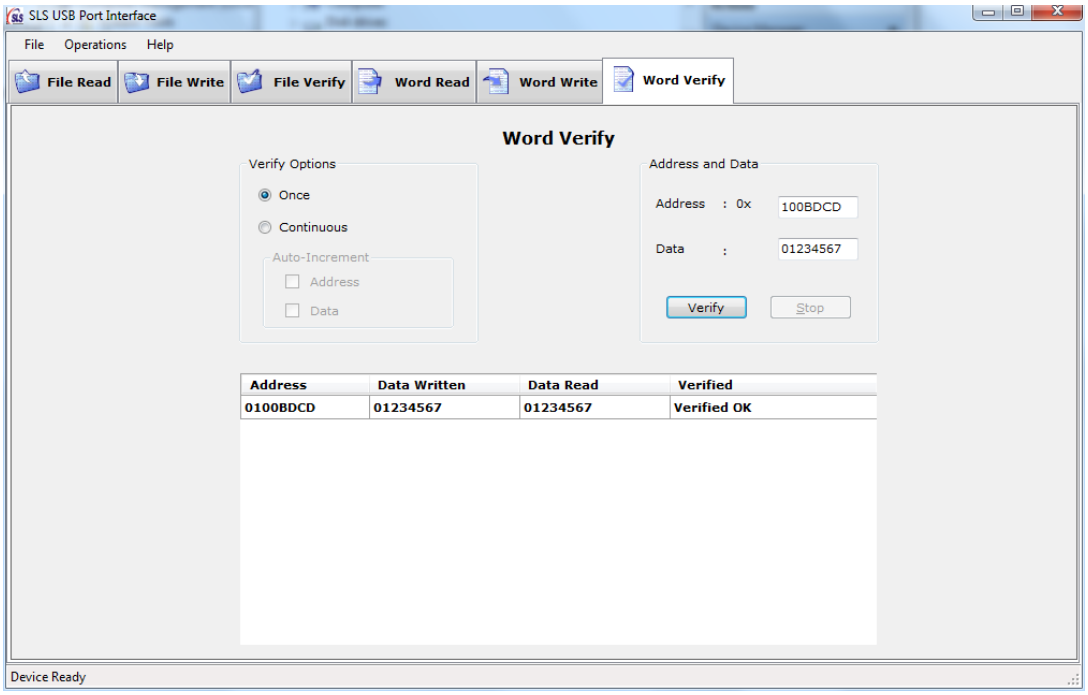


Figure 7-29 Word write and read Verfication

Note: using the on-chip memory address range for DMA access from 0x100BDCD to 0x1020F57 and the file size for File Read and Write Verify should be within the range.

Chapter 8

Programming the Configuration Flash Memory

This tutorial provides comprehensive information that will help you understand how to configure DECA Board using internal configuration mode to support 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 image 0 and image 1 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 5-1**.

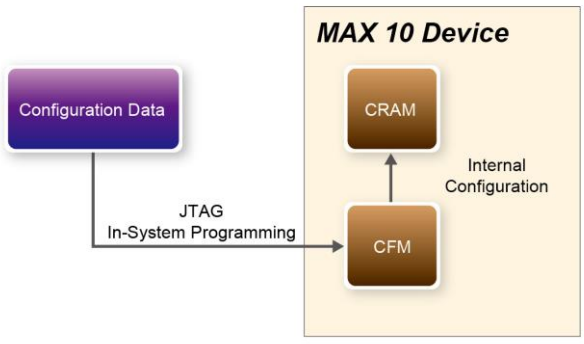


Figure 8-1 High-Level Overview of Internal Configuration for MAX 10 Devices

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 as shown in **Figure 8-2.**

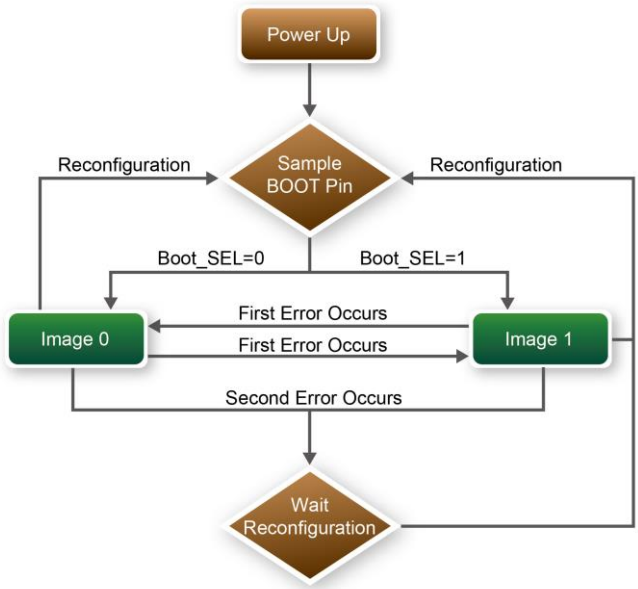


Figure 8-2 Remote System Upgrade Flow for MAX 10 Devices

The operation of the remote system upgrade feature detecting errors is as follows:
1. After power-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 Factory Default Dual Boot Image

DECA provides two factory boot images in MAX10 FPGA configuration flash memory (CFM). As shown in **Figure 8-3**, image0 and iamge1 are located respectively in LED Breathe and G-sensor image files. Before power on the board, users can switch which image to be loaded into MAX10 FPGA by BOOT_SEL (SW2). In addition, these two images' corresponding Quartus II projects can be found in the Demonstrations folder on the DECA System CD. Next section will introduce how to program image file into FPGA CFM.

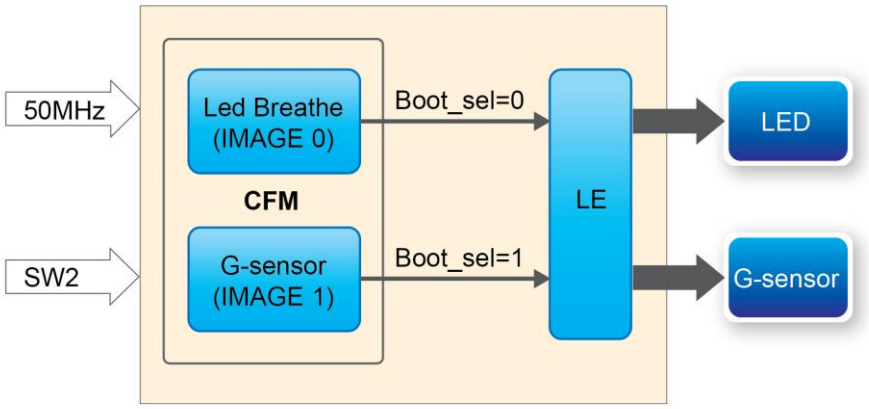


Figure 8-3 Block diagram of Dual Boot

8.3 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
Before using MAX10 Dual Compressed Images feature, users need to set these two image files' Quartus II projects as follows:

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

First of all, add a **Dual Configuration IP** in an original project, so that the .pof file can be programmed into CFM through it. 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.

Here we use a demonstration named **"Led Breathe"** as an example to add **Altera Dual Configuration IP** to the project:

1. Open Quartus project and choose **Tools** > **Qsys** to open new **Qsys** system wizard .See **Figure 8-4**.

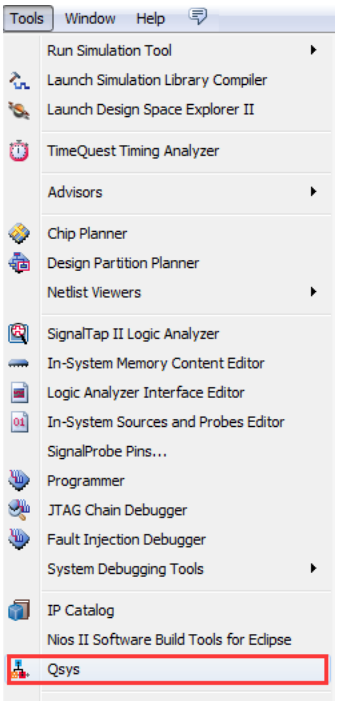


Figure 8-4 Qsys Menu

2. Choose **Library** > **Basic Function**> **Configuration and Programming** > **Altera Dual Configuration** to open wizard of adding Dual boot IP. See **Figure 8-5**.

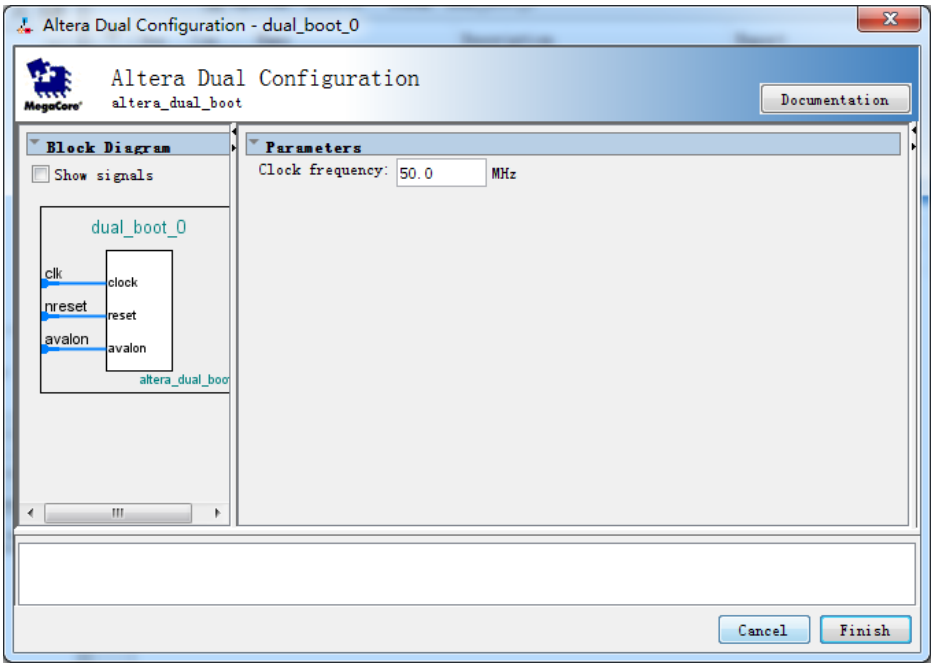


Figure 8-5 Add Dual Boot IP [1]

3. Click **Finish** to close the wizard and return to the window as shown in Figure 8-6.

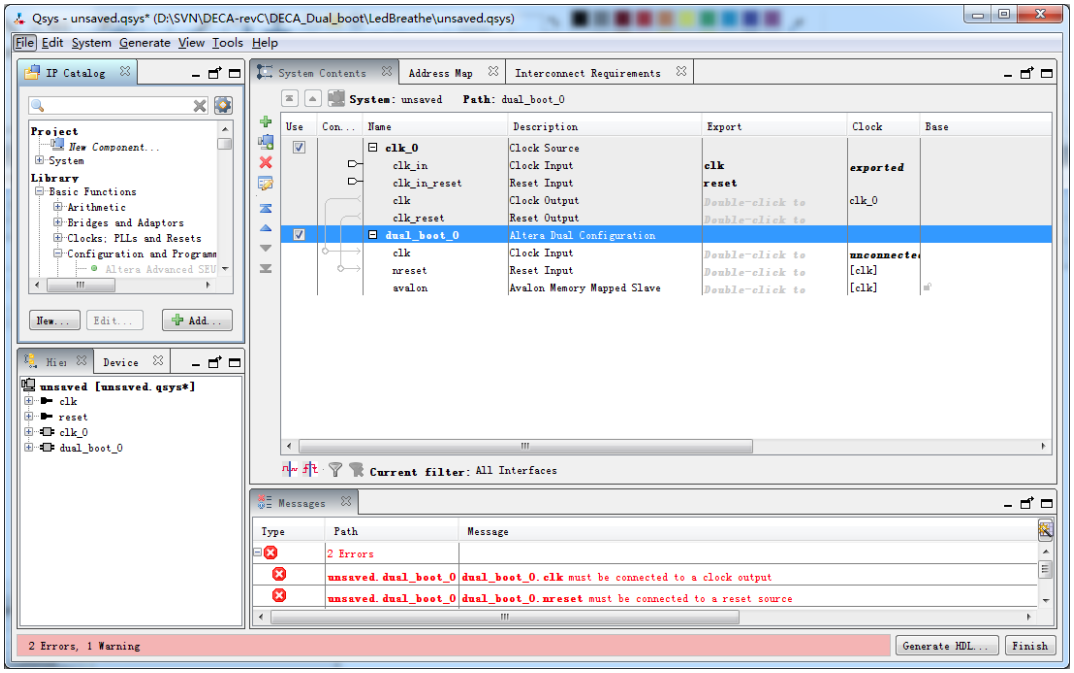


Figure 8-6 Add Dual Boot IP [2]

4. Choose **dual_boot_0** and rename it to **dual_boot**, connect the **clk** and **nreset** to clk_0.clk and clk_0.clk_reset as shown in Figure 8-7.

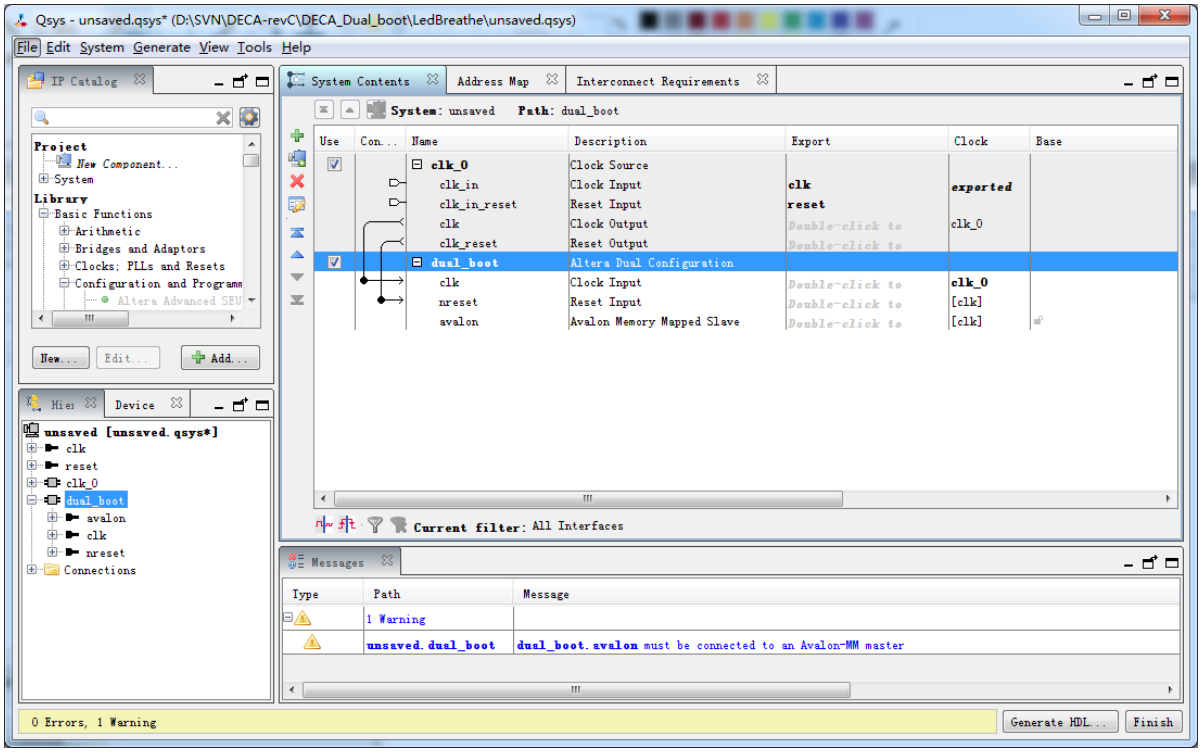


Figure 8-7 Rename and Connect Dual Boot IP

5. Click **Generate tab** and click **Generate** then pop a window as shown in **Figure 8-8**. Click **Save** it as dual_boot.qsys and the generation start. If there is no error in the generation, the window will show successful as shown in **Figure 8-9**.

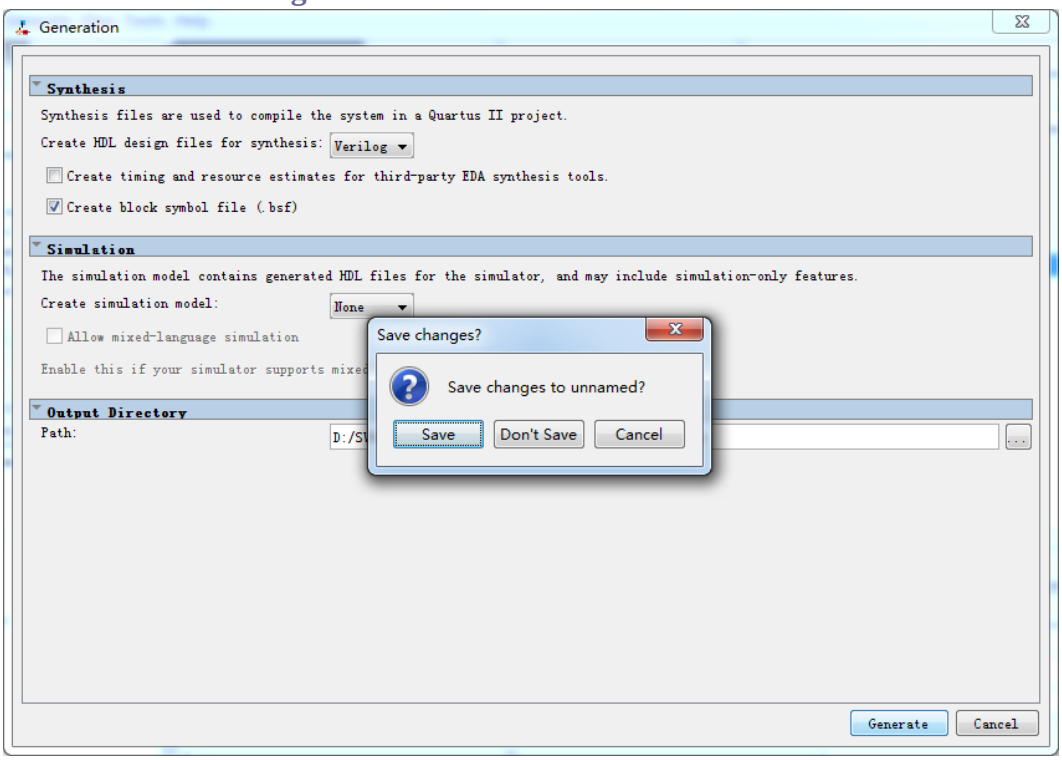


Figure 8-8 Generate and save Qsys

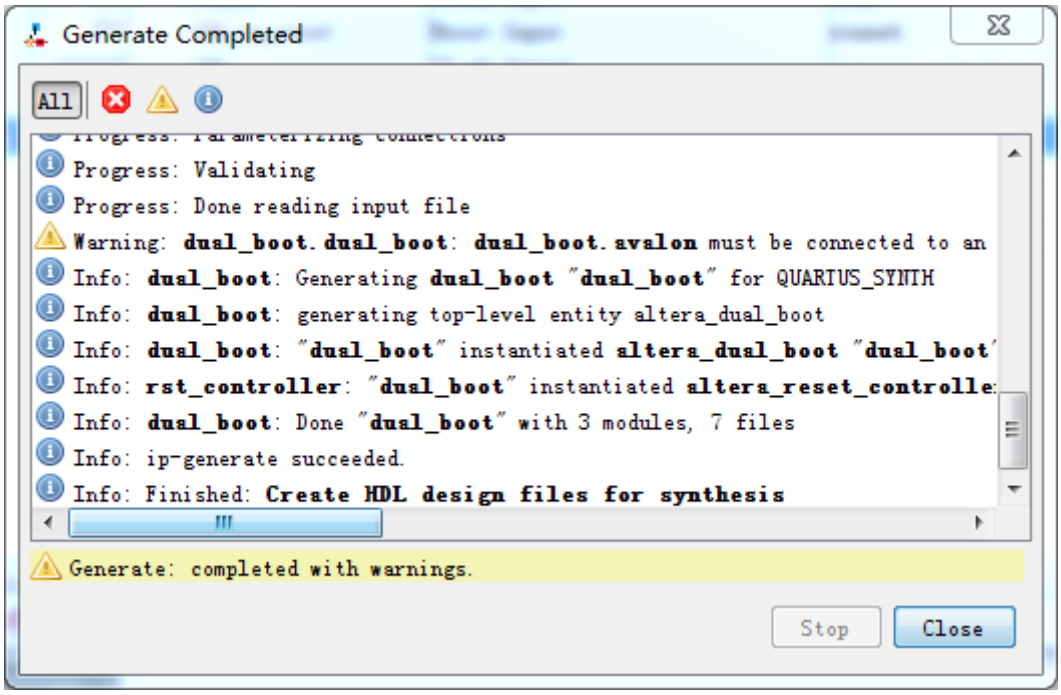


Figure 8-9 Generate Qsys Completely

6. Click **Close** and **Finish** to return to the window and add the dual boot qsys into the top file as shown in Figure 8-10.

```
233                 end
234               else begin
235                     PWM_adj <= ~ counter[25:20];
236                 end
237               display[0] <= ~PWM_width[6];
238               display[1] <= ~PWM_width[6];
239               display[2] <= ~PWM_width[6];
240               display[3] <= ~PWM_width[6];
241               display[4] <= PWM_width[6];
242               display[5] <= PWM_width[6];
243               display[6] <= PWM_width[6];
244               display[7] <= PWM_width[6];
245             end
246     end
247
248
249   assign LED = display;
250
251   dual_boot dual_boot(
252                         .clk_clk        (MAX10_CLK1_50),       //   clk.clk
253                         .reset_reset_n  (reset_n)              // reset.reset_n
254                       );
255
256
257   endmodule
258
```

Figure 8-10 Input verilog Text

7. Add the dual_boot.qip file to the project and save the project.

Secondly, the project needs to be set before the compilation. After adding dual IP successfully,

please set the project mode as Internal Configuration mode, detail steps are as follows:

1. Choose **Assignments** > **Device** to open Device windows shown in **Figure 8-11**.

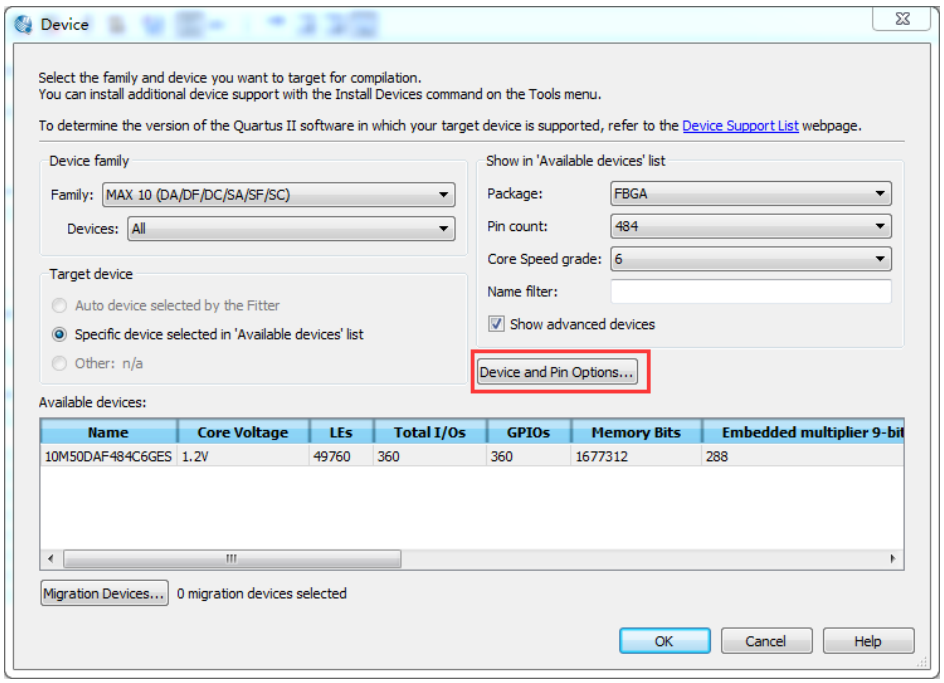


Figure 8-11 Device Window

2. Click **Device and Pin Opinions** to open the Device and Pin Opinions windows, and in the **Configuration** tab, Set the **Configuration Scheme** to **Internal Configuration** and the **Configuration Mode** to **Dual Internal Images**. Check the Option of **Generate compressed bitstreams**. shown in **Figure 8-12**.

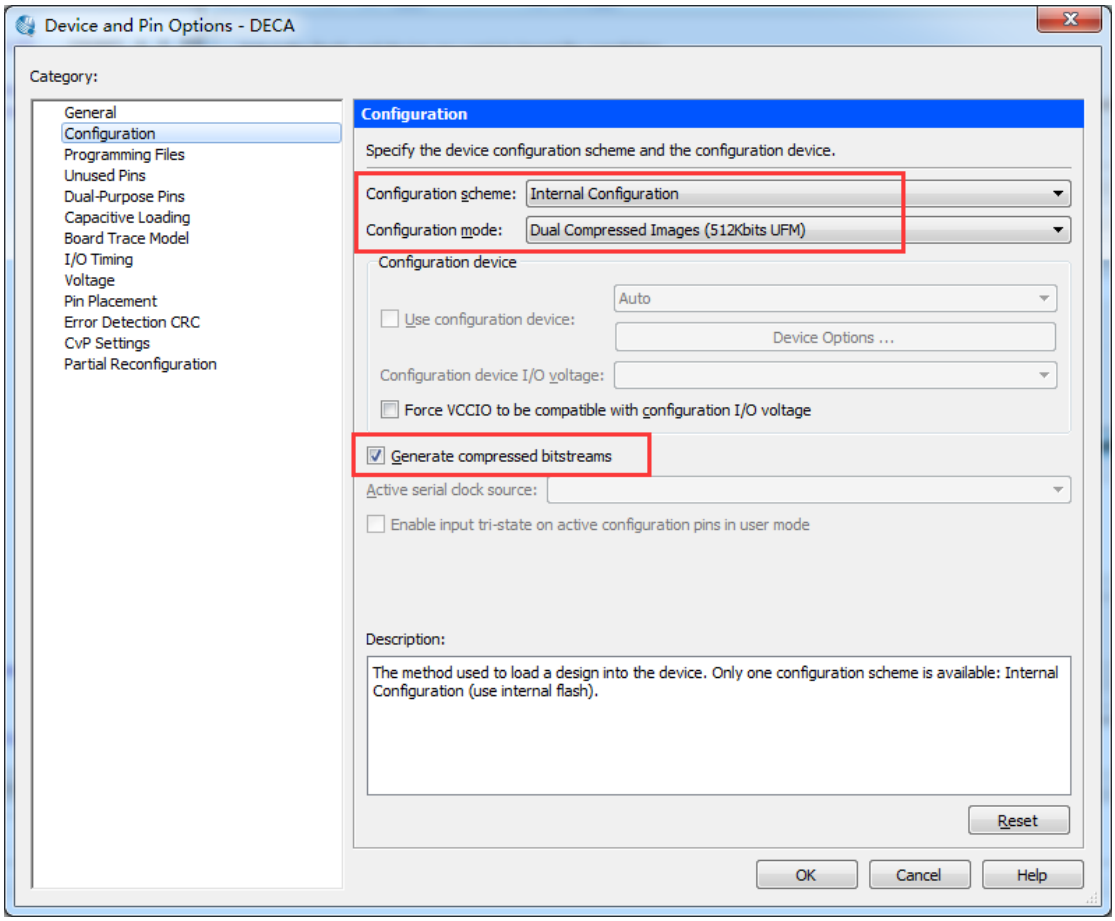


Figure 8-12 Device Window

3. Choose **OK** to return to the Quartus window. In the **Processing** menu, choose **Start Compilation** or click the Play button on the toolbar to compile the project, generate the new .sof file.

4. Use the same flow to add the Dual Configuration IP into DECA_Gsensor demo to generate the new .sof file by internal configuration mode.

Finally, So far, we have successfully obtained two image .sof files for dual boot demo according previous steps. this section describes how to generate .pof from .sof files with the internal

configuration mode and program the .pof into configuration flash memory (CFM) through the JTAG interface.

A. Convert .SOF File to .POF File

1. Choose **Convert Programming Files** from the File menu of Quartus II, as shown in **Figure 8-13**.

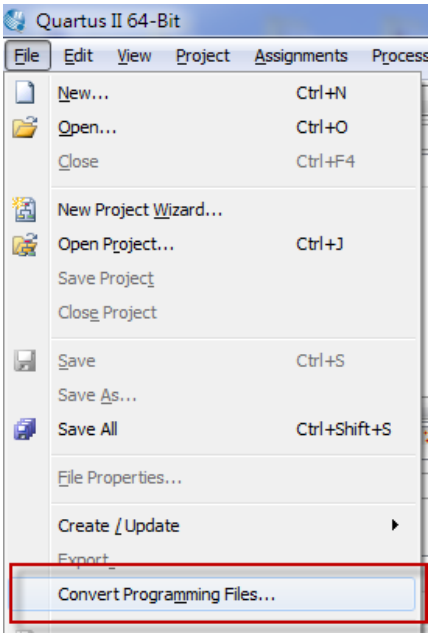


Figure 8-13 File menu of Quartus II

2. Select **Programmer Object File (.pof)** from the **Programming file type** field in the dialog of Convert Programming Files.

3. Choose **Internal Configuration** from the **Mode** filed.

4. Browse to the target directory from the **File name** field and specify the name of output file.

5. Click on the **SOF data** in the section of **Input files to convert**, as shown in **Figure 8-14**.

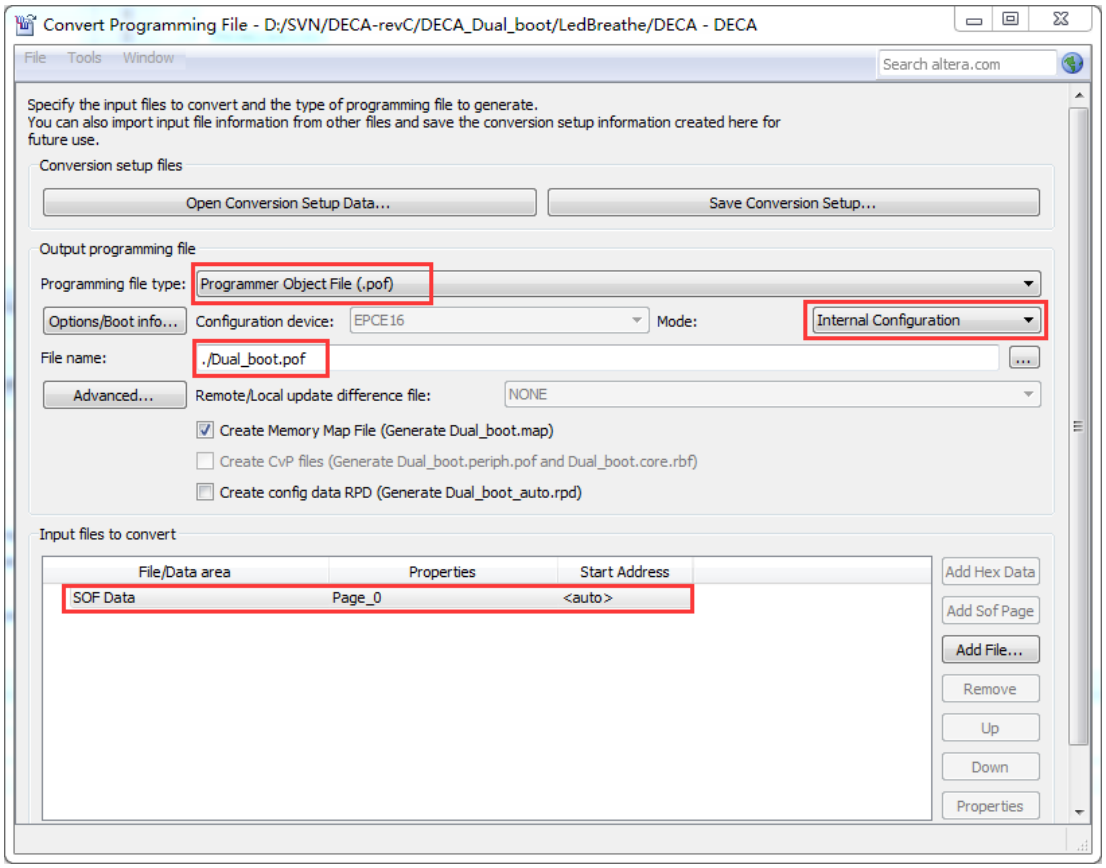


Figure 8-14 Dialog of "Convert Programming Files"

6. Click **Add File**.

7. Select the DECA.sof of LedBreathe demo to be the sof data of Page_0.

8. Click **Add Sof Page** to add Page_1 and click **Add File**, Select the DECA_Gsensor.sof of DECA_Gsensor demo to be the .sof data of Page_1 as shown in **Figure 8-15.**

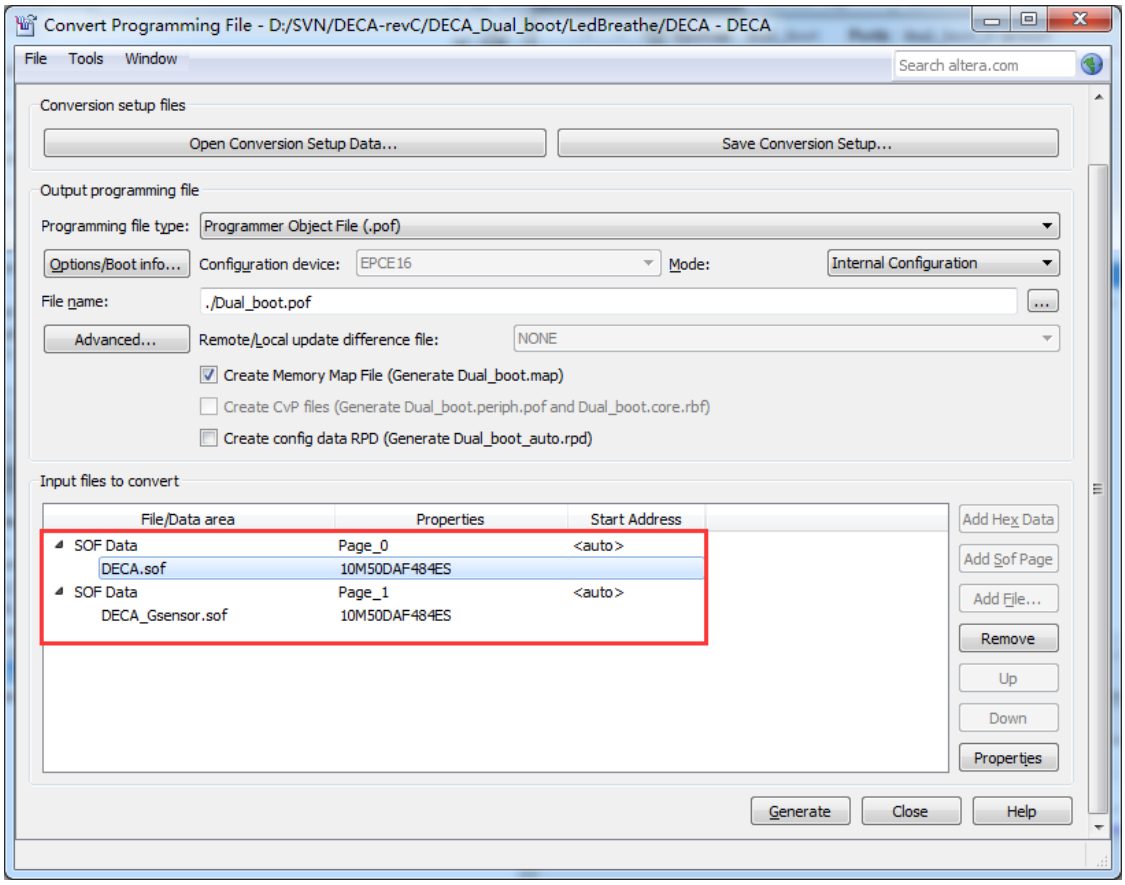


Figure 8-15 Add sof Page and sof Data

9. Click **Generate**.

B. Write POF File into the CFM Device

When the conversion of SOF-to-POF file is complete, please follow the steps below to program the EPCS device with the .pof file created in Quartus II Programmer.

1. Choose **Programmer** from the Tools menu and the **Chain.cdf** window will appear.

2. Click **Hardware Setup** and then select the Arrow MAX 10 DECA as shown in **Figure 8-16.**

3. Click **Add File** and then select the Dual_boot.pof **.**

4. Program the CFM device by clicking the corresponding **Program/Configure** and **Verify** box, as shown in **Figure 8-17**.

5. Click **Start** to program the CFM device.

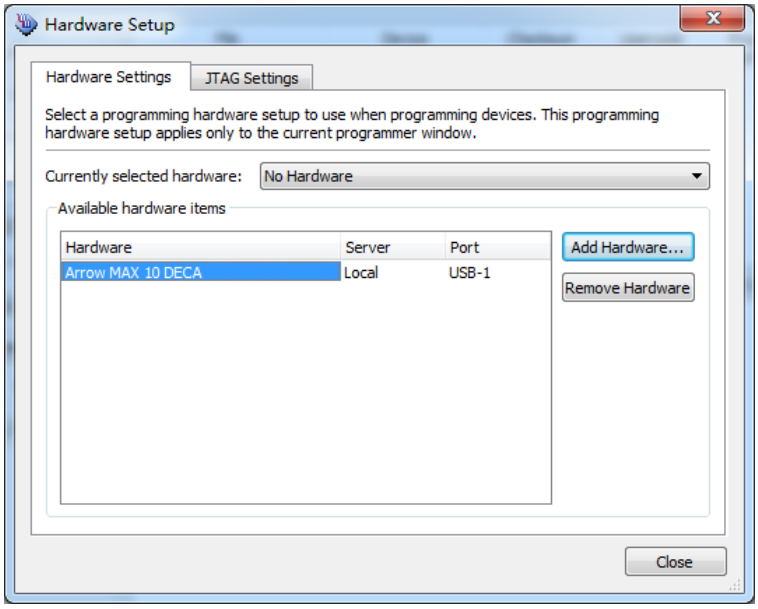


Figure 8-16 Hardware Setup window

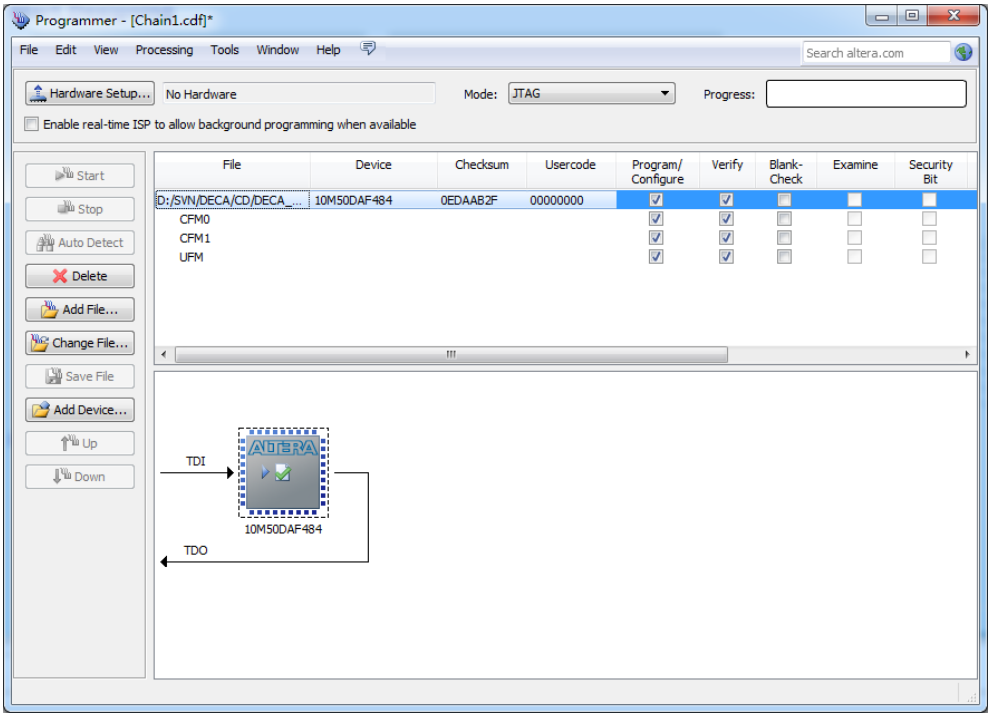


Figure 8-17 Quartus II programmer window with .pof file

Now, you can set the BOOT_SEL by SW2, you will find if you set BOOT_SEL=0, the Led Breathe functions would show. Power down the board, set BOOT_SEL=1, then Power on, you would find the DECA Gsensor functions show.

Chapter 9

Appendix

9.1 Revision History

Version	*Change Log*
V0.1	Initial Version (Preliminary)
V0.2	Add Chapter 1 and Chapter 2
V0.3	Add Chapter 3
V0.4	Modify Chapter 3
V0.5	Add Chapter 4 and Chapter 5
V1.0	Add Chapter 6 and Chapter 7
V1.1	Add Chapter 8
V1.2	Modify Chapter 6
V1.3	Modify Chapter 8
V1.4	Update kit content

9.2 Copyright Statement