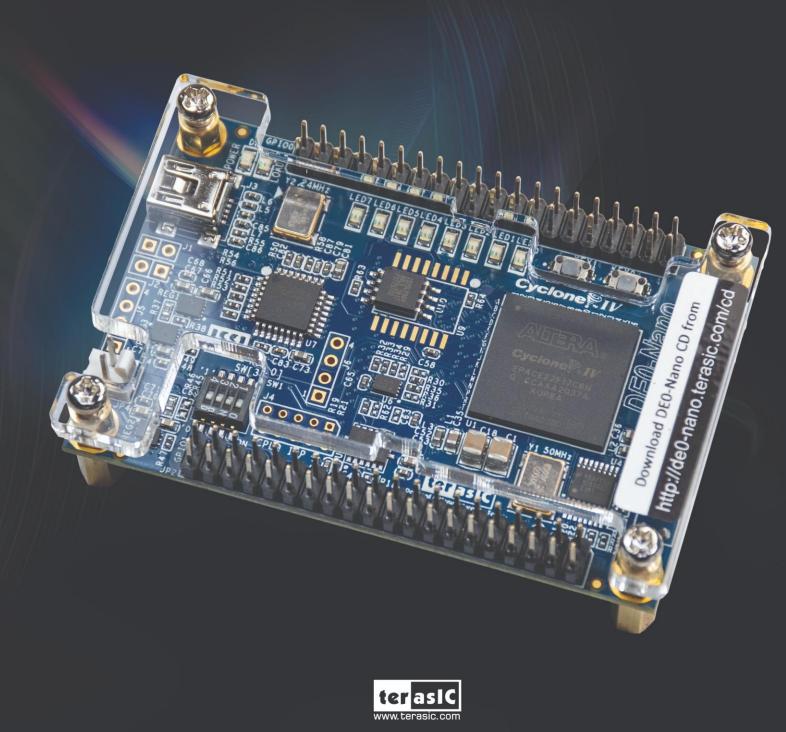
DEO-Nano User Manual World Leading FPGA Based Products and Design Services







| CHAPTER 1 | INTRODUCTION | |
|---------------------|----------------------------------|----|
| 1.1 Features | | 5 |
| 1.2 About the KIT. | | 7 |
| 1.3 Getting Help | | 7 |
| CHAPTER 2 | DE0-NANO BOARD ARCHITECTURE | |
| 2.1 Layout and Cor | nponents | 8 |
| 2.2 Block Diagram | of the DE0-Nano Board | 9 |
| 2.3 Power-up the D | E0-Nano Board | |
| CHAPTER 3 | USING THE DE0-NANO BOARD | 11 |
| 3.1 Configuring the | e Cyclone IV FPGA | |
| 3.2 General User Ir | nput/Output | |
| 3.3 SDRAM Memo | ory | 15 |
| 3.4 I2C Serial EEP | ROM | 16 |
| 3.5 Expansion Hea | ders | 17 |
| 3.6 A/D Converter | and 2x13 Header | 20 |
| 3.7 Digital Acceler | ometer | |
| 3.8 Clock Circuitry | , | |
| 3.9 Power Supply | | |
| CHAPTER 4 | DE0-NANO CONTROL PANEL | |
| 4.1 Control Panel S | etup | |
| 4.2 Controlling the | LEDs | |
| 4.3 Switches and P | ushbuttons | |
| 4.4 Memory Contro | oller | |
| 4.5 Digital Acceler | ometer | |
| 4.6 ADC | | |
| 4.7 Overall Structu | re of the DE0-Nano Control Panel | |
| CHAPTER 5 | DE0-NANO SYSTEM BUILDER | |
| 5.1 Introduction | | |

2

| 5.2 General Desig | n Flow | |
|---------------------|--------------------------------------|-----------------|
| 5.3 Using DE0-Na | no System Builder | |
| CHAPTER 6 | TUTORIAL: CREATING AN FPGA PROJECT | |
| 6.1 Design Flow | | 40 |
| 6.2 Before You Be | gin | 41 |
| 6.3 What You Will | Learn | 45 |
| 6.4 Assign The De | vice | 45 |
| 6.5 Creating an FF | GA design | 49 |
| 6.6 Assign the Pin | S | 71 |
| 6.7 Create a Defau | It TimeQuest SDC File | |
| 6.8 Compile Your | Design | 74 |
| 6.9 Program the F | PGA Device | 76 |
| 6.10 Verify The H | ardware | |
| CHAPTER 7 | TUTORIAL: CREATING A NIOS II PROJECT | |
| 7.1 Required Featu | ıres | |
| 7.2 Creation of Ha | rdware Design | |
| 7.3 Download the | Hardware Design | |
| 7.4 Create a hello_ | _world Example Project | |
| 7.5 Build and Run | the Program | |
| 7.6 Edit and Re-R | un the Program | |
| 7.7 Why the LED | Blinks | |
| 7.8 Debugging the | Application | |
| 7.9 Configure Syst | tem Library | |
| CHAPTER 8 | DE0-NANO DEMONSTRATIONS | |
| 8.1 System Requir | ements | |
| 8.2 Breathing LED | Ds | |
| 8.3 ADC Reading. | | |
| 8.4 SOPC Demo | | |
| 8.5 G-Sensor | | |
| 8.6 SDRAM Test | by Nios II | |
| CHAPTER 9 | APPENDIX | |
| 9.1 Programming | the Serial Configuration Device | |
| 9.2 EPCS Program | nming via nios-2-flash-programmer | |
| terasic | 3 DE0-Nano User Manual | www.terasic.com |



| 9.3 Revision History | 154 |
|-------------------------|-----|
| 9.4 Copyright Statement | 154 |



Chapter 1



The DE0-Nano board introduces a compact-sized FPGA development platform suited for to a wide range of portable design projects, such as robots and mobile projects.

The DE0-Nano is ideal for use with embedded soft processors—it features a powerful Altera Cyclone IV FPGA (with 22,320 logic elements), 32 MB of SDRAM, 2 Kb EEPROM, and a 64 Mb serial configuration memory device. For connecting to real-world sensors the DE0-Nano includes a National Semiconductor 8-channel 12-bit A/D converter, and it also features an Analog Devices 13-bit, 3-axis accelerometer device.

The DE0-Nano board includes a built-in USB Blaster for FPGA programming, and the board can be powered either from this USB port or by an external power source. The board includes expansion headers that can be used to attach various Terasic daughter cards or other devices, such as motors and actuators. Inputs and outputs include 2 pushbuttons, 8 user LEDs and a set of 4 dip-switches.

1.1 Features

Figure 1-1 shows a photograph of the DEO-Nano Board.

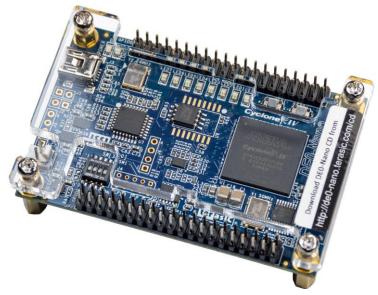


Figure 1-1 The DE0-Nano Board



The key features of the board are listed below:

- Featured device
 - Altera Cyclone® IV EP4CE22F17C6N FPGA
 - o 153 maximum FPGA I/O pins
- Configuration status and set-up elements
 - On-board USB-Blaster circuit for programming
 - Spansion EPCS64
- Expansion header
 - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
 - o 32MB SDRAM
 - o 2Kb I2C EEPROM
- General user input/output
 - o 8 green LEDs
 - o 2 debounced pushbuttons
 - o 4-position DIP switch
- G-Sensor
 - ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
- A/D Converter
 - NS ADC128S022, 8-Channel, 12-bit A/D Converter
 - o 50 Ksps to 200 Ksps
- Clock system
 - On-board 50MHz clock oscillator
- Power Supply
 - USB Type mini-AB port (5V)
 - DC 5V pin for each GPIO header (2 DC 5V pins)
 - 2-pin external power header (3.6-5.7V)



1.2 About the KIT

The kit comes with the following contents:

- DE0-Nano board
- Quick Start Guide
- USB Cable
- CD (Download from Terasic Web)

The system CD contains technical documents for the DE0-Nano board, which includes component datasheets, demonstrations, schematic, and user manual. User can download this System CD from the web: <u>http://de0-nano.terasic.com/cd</u>.

Figure 1-2 shows the photograph of the DE0-Nano kit contents.



Figure 1-2 DE0-Nano kit package contents

1.3 Getting Help

Here is information of how to get help if you encounter any problem:

- Terasic Technologies
- Tel: +886-3-575-0880
- Email: support@terasic.com
- Altera Corporation
- Email: university@altera.com



Chapter 2

DE0-Nano Board Architecture

This chapter describes the architecture of the DE0-Nano board including block diagram and components.

2.1 Layout and Components

The picture of the DEO-Nano board is shown in **Figure 2-1** and **Figure 2-2**. It depicts the layout of the board and indicates the locations of the connectors and key components.

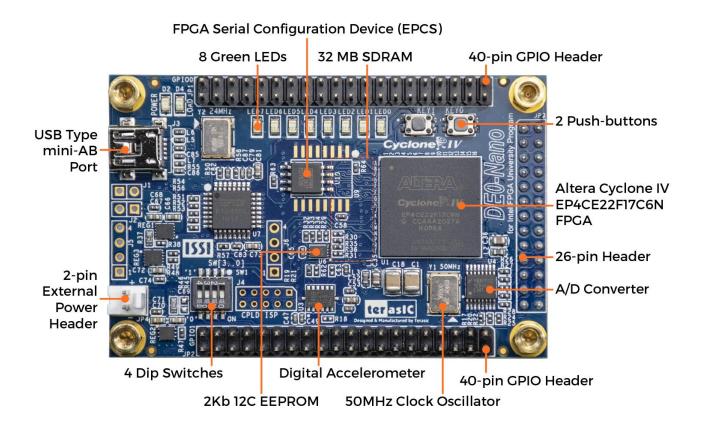


Figure 2-1 The DE0-Nano Board PCB and component diagram (top view)



32 MB SDRAM

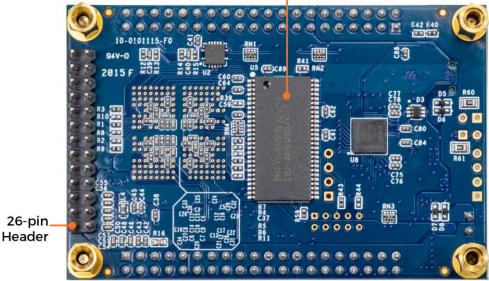


Figure 2-2 The DE0-Nano Board PCB and component diagram (bottom view)

2.2 Block Diagram of the DE0-Nano Board

Figure 2-3 shows the block diagram of the DE0-Nano board. To provide maximum flexibility for the user, all connections are made through the Cyclone IV FPGA device. Thus, the user can configure the FPGA to implement any system design.

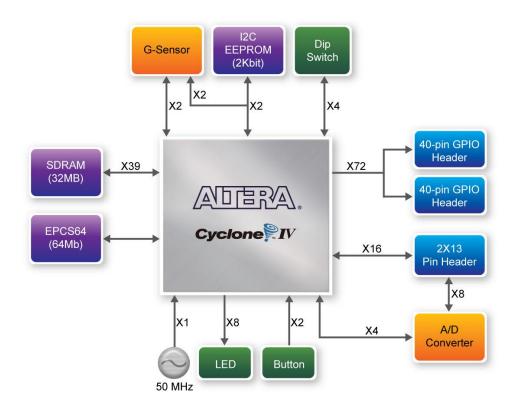


Figure 2-3 Block diagram of DE0-Nano Board

2.3 Power-up the DE0-Nano Board

The DE0-Nano board comes with a preloaded configuration bit stream to demonstrate some features of the board. This allows users to see quickly if the board is working properly. To power-up the board two options are available which are described below:

1. Connect a USB Mini-B cable between a USB (Type A) host port and the board. For communication between the host and the DEO-Nano board, it is necessary to install the Altera USB Blaster driver software.

2. Alternatively, users can power-up the DE0-Nano board by supplying 5V to the two DC +5 (VCC5) pins of the GPIO headers or supplying (3.6-5.7V) to the 2-pin header.

At this point you should observe flashing LEDs on the board.





Chapter 3

Using the DE0-Nano Board

This chapter gives instructions for using the DEO-Nano board and describes in detail its components and connectors, along with the required pin assignments.

3.1 Configuring the Cyclone IV FPGA

The DE0-Nano board contains a Cyclone IV E FPGA which can be programmed using JTAG programming. This allows users to configure the FPGA with a specified design using Quartus II software. The programmed design will remain functional on the FPGA as long as the board is powered on, or until the device is reprogrammed. The configuration information will be lost when the power is turned off.

To download a configuration bit stream file using JTAG Programming into the Cyclone IV FPGA, perform the following steps:

1. Connect a USB Mini-B cable between a host computer and the DE0-Nano.

2. The FPGA can now be programmed through the Quartus II Programmer by selecting a configuration bit stream file with the .sof filename extension.

Configuring the Spansion EPCS64 device

The DE0-Nano board contains a Spansion EPCS64 serial configuration device. This device provides non-volatile storage of the configuration bit-stream, so that the information is retained even when the power supply to the DE0-Nano board is turned off. When the board's power is turned on, the configuration data in the EPCS64 device is automatically loaded into the Cyclone IV E FPGA.

The Cyclone IV E device supports in-system programming of a serial configuration device using the JTAG interface via the serial flash loader design. The serial flash loader is a bridge design for the Cyclone IV E device that uses its JTAG interface to access the EPCS .jic file and then uses the AS interface to program the EPCS device. **Figure 3-1** illustrates the programming method when adopting a serial flash loader solution. Chapter 9 of this document describes how to load a circuit to the serial configuration device.

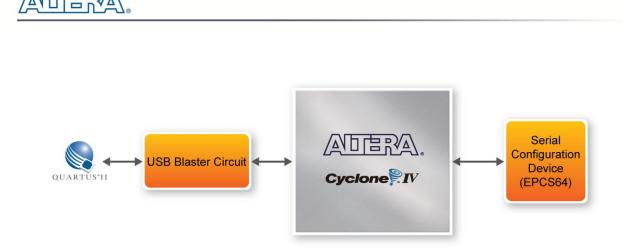
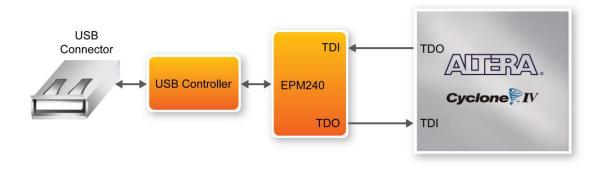


Figure 3-1 Programming a serial configuration device with serial flash loader solution

■ JTAG Chain on DE0-Nano Board

The JTAG Chain on the DEO-Nano board is connected to a host computer using an on-board USB-blaster. The USB-blaster consists of a USB Mini-B connector, a FTDI USB 2.0 Controller, and an Altera MAX II CPLD.

Figure 3-2 illustrates the JTAG configuration setup.





3.2 General User Input/Output

Pushbuttons

The DE0-Nano board contains two pushbuttons shown in **Figure 3-3**. Each pushbutton is debounced using a Schmitt Trigger circuit, as indicated in **Figure 3-4**. The two outputs called KEY0, and KEY1 of the Schmitt Trigger devices are connected directly to the Cyclone IV E FPGA. Each pushbutton provides a high logic level when it is not pressed, and provides a low logic level when pressed. Since the pushbuttons are debounced, they are appropriate for using as clock or reset inputs.



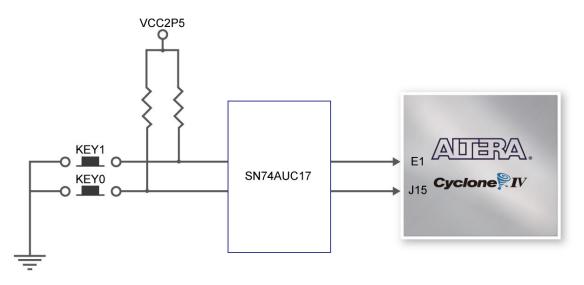


Figure 3-3 Connections between the push-buttons and Cyclone IV FPGA

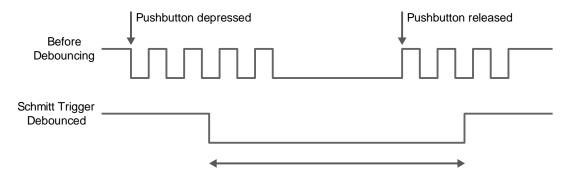


Figure 3-4 Pushbuttons debouncing

■ LEDs

There are 8 green user-controllable LEDs on the DE0-Nano board. The eight LEDs, which are presented in **Figure 3-4**, allow users to display status and debugging information. Each LED is driven directly by the Cyclone IV E FPGA. Each LED is driven directly by a pin on the Cyclone IV E FPGA; driving its associated pin to a high logic level turns the LED on, and driving the pin low turns it off.



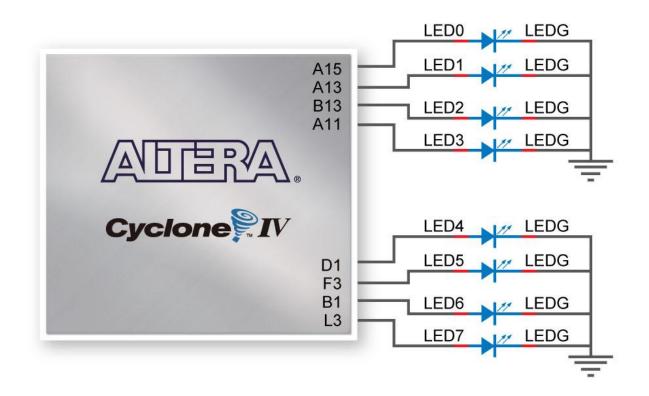


Figure 3-5 Connections between the LEDs and Cyclone IV FPGA

■ **DIP** Switch

The DE0-Nano board contains a 4 dip switches. A DIP switch provides, to the FPGA, a high logic level when it is in the DOWN position, and a low logic level when in the UPPER position.

| Table 3-1 Pin Assignments for Push-buttons | | | | |
|--|--------------|----------------|--------------|--|
| Signal Name | FPGA Pin No. | Description | I/O Standard | |
| KEY[0] | PIN_J15 | Push-button[0] | 3.3V | |
| KEY[1] | PIN_E1 | Push-button[1] | 3.3V | |

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|-------------|--------------|--------------|--------------|
| LED[0] | PIN_A15 | LED Green[0] | 3.3V |
| LED[1] | PIN_A13 | LED Green[1] | 3.3V |
| LED[2] | PIN_B13 | LED Green[2] | 3.3V |
| LED[3] | PIN_A11 | LED Green[3] | 3.3V |
| LED[4] | PIN_D1 | LED Green[4] | 3.3V |
| LED[5] | PIN_F3 | LED Green[5] | 3.3V |
| LED[6] | PIN_B1 | LED Green[6] | 3.3V |
| LED[7] | PIN_L3 | LED Green[7] | 3.3V |

Table 3-2 Pin Assignments for LEDs



| Signal Name | FPGA Pin No. | Description | I/O Standard |
|---------------|--------------|---------------|--------------|
| DIP Switch[0] | PIN_M1 | DIP Switch[0] | 3.3V |
| DIP Switch[1] | PIN_T8 | DIP Switch[1] | 3.3V |
| DIP Switch[2] | PIN_B9 | DIP Switch[2] | 3.3V |
| DIP Switch[3] | PIN_M15 | DIP Switch[3] | 3.3V |

3.3 SDRAM Memory

The board features a Synchronous Dynamic Random Access Memory (SDRAM) device providing 32MB with a 16-bit data lines connected to the FPGA. The chip uses 3.3V LVCMOS signaling standard. All signals are registered on the positive edge of the clock signal, DRAM_CLK. Connections between the FPGA and SDRAM chips are shown in **Figure 3-6**.



Figure 3-6 Connections between FPGA and SDRAM

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|---------------|--------------|-------------------|--------------|
| DRAM_ADDR[0] | PIN_P2 | SDRAM Address[0] | 3.3V |
| DRAM_ADDR[1] | PIN_N5 | SDRAM Address[1] | 3.3V |
| DRAM_ADDR[2] | PIN_N6 | SDRAM Address[2] | 3.3V |
| DRAM_ADDR[3] | PIN_M8 | SDRAM Address[3] | 3.3V |
| DRAM_ADDR[4] | PIN_P8 | SDRAM Address[4] | 3.3V |
| DRAM_ADDR[5] | PIN_T7 | SDRAM Address[5] | 3.3V |
| DRAM_ADDR[6] | PIN_N8 | SDRAM Address[6] | 3.3V |
| DRAM_ADDR[7] | PIN_T6 | SDRAM Address[7] | 3.3V |
| DRAM_ADDR[8] | PIN_R1 | SDRAM Address[8] | 3.3V |
| DRAM_ADDR[9] | PIN_P1 | SDRAM Address[9] | 3.3V |
| DRAM_ADDR[10] | PIN_N2 | SDRAM Address[10] | 3.3V |
| DRAM_ADDR[11] | PIN_N1 | SDRAM Address[11] | 3.3V |



| DRAM_ADDR[12] | PIN_L4 | SDRAM Address[12] | 3.3V |
|---------------|--------|-----------------------------|------|
| DRAM_DQ[0] | PIN_G2 | SDRAM Data[0] | 3.3V |
| DRAM_DQ[1] | PIN_G1 | SDRAM Data[1] | 3.3V |
| DRAM_DQ[2] | PIN_L8 | SDRAM Data[2] | 3.3V |
| DRAM_DQ[3] | PIN_K5 | SDRAM Data[3] | 3.3V |
| DRAM_DQ[4] | PIN_K2 | SDRAM Data[4] | 3.3V |
| DRAM_DQ[5] | PIN_J2 | SDRAM Data[5] | 3.3V |
| DRAM_DQ[6] | PIN_J1 | SDRAM Data[6] | 3.3V |
| DRAM_DQ[7] | PIN_R7 | SDRAM Data[7] | 3.3V |
| DRAM_DQ[8] | PIN_T4 | SDRAM Data[8] | 3.3V |
| DRAM_DQ[9] | PIN_T2 | SDRAM Data[9] | 3.3V |
| DRAM_DQ[10] | PIN_T3 | SDRAM Data[10] | 3.3V |
| DRAM_DQ[11] | PIN_R3 | SDRAM Data[11] | 3.3V |
| DRAM_DQ[12] | PIN_R5 | SDRAM Data[12] | 3.3V |
| DRAM_DQ[13] | PIN_P3 | SDRAM Data[13] | 3.3V |
| DRAM_DQ[14] | PIN_N3 | SDRAM Data[14] | 3.3V |
| DRAM_DQ[15] | PIN_K1 | SDRAM Data[15] | 3.3V |
| DRAM_BA[0] | PIN_M7 | SDRAM Bank Address[0] | 3.3V |
| DRAM_BA[1] | PIN_M6 | SDRAM Bank Address[1] | 3.3V |
| DRAM_DQM[0] | PIN_R6 | SDRAM byte Data Mask[0] | 3.3V |
| DRAM_DQM[1] | PIN_T5 | SDRAM byte Data Mask[1] | 3.3V |
| DRAM_RAS_N | PIN_L2 | SDRAM Row Address Strobe | 3.3V |
| DRAM_CAS_N | PIN_L1 | SDRAM Column Address Strobe | 3.3V |
| DRAM_CKE | PIN_L7 | SDRAM Clock Enable | 3.3V |
| DRAM_CLK | PIN_R4 | SDRAM Clock | 3.3V |
| DRAM_WE_N | PIN_C2 | SDRAM Write Enable | 3.3V |
| DRAM_CS_N | PIN_P6 | SDRAM Chip Select | 3.3V |

3.4 I2C Serial EEPROM

The DE0-Nano contains a 2Kbit Electrically Erasable PROM (EEPROM). The EEPROM is configured through a 2-wire I2C serial interface. The device is organized as one block of 256 x 8-bit memory. The I2C write and read address are 0xA0 and 0xA1, respectively. **Figure 3-7** illustrates its connections with the Cyclone IV FPGA.

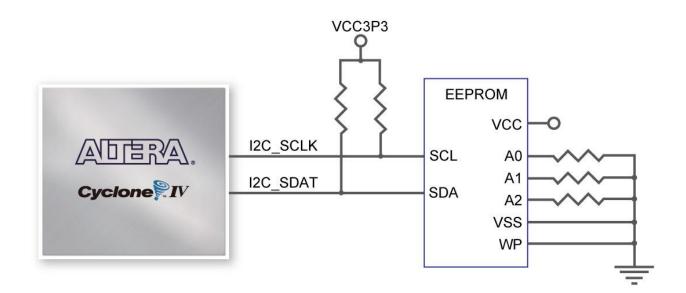


Figure 3-7 Connections between FPGA and EEPROM

| Signal Name | FPGA Pin No. | Description | I/O Standard | |
|-------------|--------------|--------------|--------------|--|
| I2C_SCLK | PIN_F2 | EEPROM clock | 3.3V | |
| I2C_SDAT | PIN_F1 | EEPROM data | 3.3V | |

 Table 3-5 Pin Assignments for I2C Serial EEPROM

3.5 Expansion Headers

The DE0-Nano board provides two 40-pin expansion headers. Each header connects directly to 36 pins of the Cyclone IV E FPGA, and also provides DC +5V (VCC5), DC +3.3V (VCC33), and two GND pins. **Figure 3-8** shows the I/O distribution of the GPIO connectors.



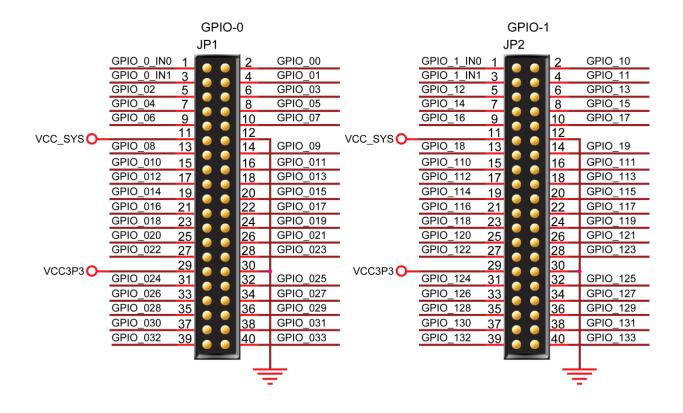


Figure 3-8 Pin arrangement of the GPIO expansion headers

The pictures below indicate the pin 1 location of the expansion headers.

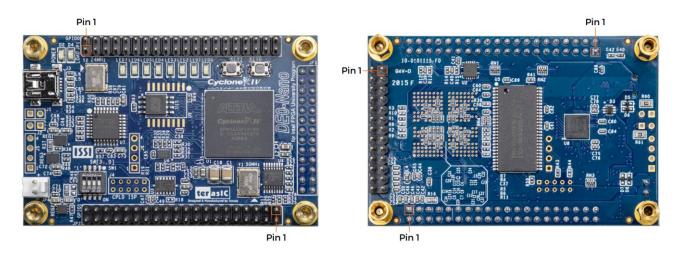


Figure 3-9 Pin1 locations of the GPIO expansion headers

| Signal Name | FPGA Pin No. | Description | I/O Standard | |
|-------------|--------------|----------------------|--------------|--|
| GPIO_0_IN0 | PIN_A8 | GPIO Connection DATA | 3.3V | |
| GPIO_00 | PIN_D3 | GPIO Connection DATA | 3.3V | |
| GPIO_0_IN1 | PIN_B8 | GPIO Connection DATA | 3.3V | |
| GPIO_01 | PIN_C3 | GPIO Connection DATA | 3.3V | |

Table 3-6 GPIO-0 Pin Assignments



| GPIO_02 | PIN_A2 | GPIO Connection DATA | 3.3V |
|----------|---------|-----------------------------|------|
| GPIO_03 | PIN_A3 | GPIO Connection DATA | 3.3V |
| GPIO_04 | PIN_B3 | GPIO Connection DATA | 3.3V |
| GPIO_05 | PIN_B4 | GPIO Connection DATA | 3.3V |
| GPIO_06 | PIN_A4 | GPIO Connection DATA | 3.3V |
| GPIO_07 | PIN_B5 | GPIO Connection DATA | 3.3V |
| GPIO_08 | PIN_A5 | GPIO Connection DATA | 3.3V |
| GPIO_09 | PIN_D5 | GPIO Connection DATA | 3.3V |
| GPIO_010 | PIN_B6 | GPIO Connection DATA | 3.3V |
| GPIO_011 | PIN_A6 | GPIO Connection DATA | 3.3V |
| GPIO_012 | PIN_B7 | GPIO Connection DATA | 3.3V |
| GPIO_013 | PIN_D6 | GPIO Connection DATA | 3.3V |
| GPIO_014 | PIN_A7 | GPIO Connection DATA | 3.3V |
| GPIO_015 | PIN_C6 | GPIO Connection DATA | 3.3V |
| GPIO_016 | PIN_C8 | GPIO Connection DATA | 3.3V |
| GPIO_017 | PIN_E6 | GPIO Connection DATA | 3.3V |
| GPIO_018 | PIN_E7 | GPIO Connection DATA | 3.3V |
| GPIO_019 | PIN_D8 | GPIO Connection DATA | 3.3V |
| GPIO_020 | PIN_E8 | GPIO Connection DATA | 3.3V |
| GPIO_021 | PIN_F8 | GPIO Connection DATA | 3.3V |
| GPIO_022 | PIN_F9 | GPIO Connection DATA | 3.3V |
| GPIO_023 | PIN_E9 | GPIO Connection DATA | 3.3V |
| GPIO_024 | PIN_C9 | GPIO Connection DATA | 3.3V |
| GPIO_025 | PIN_D9 | GPIO Connection DATA | 3.3V |
| GPIO_026 | PIN_E11 | GPIO Connection DATA | 3.3V |
| GPIO_027 | PIN_E10 | GPIO Connection DATA | 3.3V |
| GPIO_028 | PIN_C11 | GPIO Connection DATA | 3.3V |
| GPIO_029 | PIN_B11 | GPIO Connection DATA | 3.3V |
| GPIO_030 | PIN_A12 | GPIO Connection DATA | 3.3V |
| GPIO_031 | PIN_D11 | GPIO Connection DATA | 3.3V |
| GPIO_032 | PIN_D12 | GPIO Connection DATA | 3.3V |
| GPIO_033 | PIN_B12 | GPIO Connection DATA | 3.3V |

Table 3-7 GPIO-1 Pin Assignments

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|-------------|--------------|----------------------|--------------|
| GPIO_1_IN0 | PIN_T9 | GPIO Connection DATA | 3.3V |
| GPIO_10 | PIN_F13 | GPIO Connection DATA | 3.3V |
| GPIO_1_IN1 | PIN_R9 | GPIO Connection DATA | 3.3V |
| GPIO_11 | PIN_T15 | GPIO Connection DATA | 3.3V |
| GPIO_12 | PIN_T14 | GPIO Connection DATA | 3.3V |
| GPIO_13 | PIN_T13 | GPIO Connection DATA | 3.3V |
| GPIO_14 | PIN_R13 | GPIO Connection DATA | 3.3V |
| GPIO_15 | PIN_T12 | GPIO Connection DATA | 3.3V |



| GPIO_16 | PIN_R12 | GPIO Connection DATA | 3.3V |
|----------|---------|-----------------------------|------|
| GPIO_17 | PIN_T11 | GPIO Connection DATA | 3.3V |
| GPIO_18 | PIN_T10 | GPIO Connection DATA | 3.3V |
| GPIO_19 | PIN_R11 | GPIO Connection DATA | 3.3V |
| GPIO_110 | PIN_P11 | GPIO Connection DATA | 3.3V |
| GPIO_111 | PIN_R10 | GPIO Connection DATA | 3.3V |
| GPIO_112 | PIN_N12 | GPIO Connection DATA | 3.3V |
| GPIO_113 | PIN_P9 | GPIO Connection DATA | 3.3V |
| GPIO_114 | PIN_N9 | GPIO Connection DATA | 3.3V |
| GPIO_115 | PIN_N11 | GPIO Connection DATA | 3.3V |
| GPIO_116 | PIN_L16 | GPIO Connection DATA | 3.3V |
| GPIO_117 | PIN_K16 | GPIO Connection DATA | 3.3V |
| GPIO_118 | PIN_R16 | GPIO Connection DATA | 3.3V |
| GPIO_119 | PIN_L15 | GPIO Connection DATA | 3.3V |
| GPIO_120 | PIN_P15 | GPIO Connection DATA | 3.3V |
| GPIO_121 | PIN_P16 | GPIO Connection DATA | 3.3V |
| GPIO_122 | PIN_R14 | GPIO Connection DATA | 3.3V |
| GPIO_123 | PIN_N16 | GPIO Connection DATA | 3.3V |
| GPIO_124 | PIN_N15 | GPIO Connection DATA | 3.3V |
| GPIO_125 | PIN_P14 | GPIO Connection DATA | 3.3V |
| GPIO_126 | PIN_L14 | GPIO Connection DATA | 3.3V |
| GPIO_127 | PIN_N14 | GPIO Connection DATA | 3.3V |
| GPIO_128 | PIN_M10 | GPIO Connection DATA | 3.3V |
| GPIO_129 | PIN_L13 | GPIO Connection DATA | 3.3V |
| GPIO_130 | PIN_J16 | GPIO Connection DATA | 3.3V |
| GPIO_131 | PIN_K15 | GPIO Connection DATA | 3.3V |
| GPIO_132 | PIN_J13 | GPIO Connection DATA | 3.3V |
| GPIO_133 | PIN_J14 | GPIO Connection DATA | 3.3V |

3.6 A/D Converter and 2x13 Header

The DE0-Nano contains an ADC128S022 lower power, eight-channel CMOS 12-bit analog-to-digital converter. This A-to-D provides conversion throughput rates of 50 ksps to 200 ksps. It can be configured to accept up to eight input signals at inputs IN0 through IN7. This eight input signals are connected to the 2x13 header, as shown in Figure 3-10. The remaining I/Os of the 2x13 header are a DC +3.3V (VCC33), a GND and 13 pins, which are connect directly to the Cyclone IV E device.

For more detailed information on the A/D converter chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.



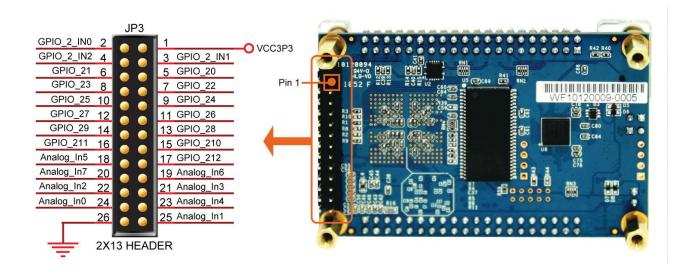


Figure 3-10 Pin distribution of the 2x13 Header

Figure 3-11 shows the connections on the 2x13 header, A/D converter and Cyclone IV device.

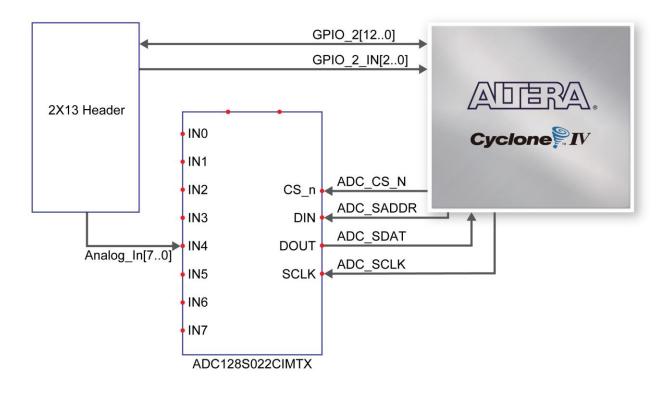


Figure 3-11 Wiring for 2x13 header and A/D converter

The pictures below indicate the pin 1 location of the 2x13 header.



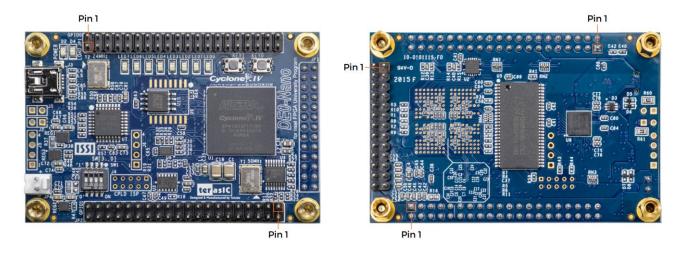


Figure 3-12 Pin1 locations of the 2x13 header

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|--------------|--------------|---------------------------------|--------------|
| GPIO_2[0] | PIN_A14 | GPIO Connection DATA[0] | 3.3V |
| GPIO_2[1] | PIN_B16 | GPIO Connection DATA[1] | 3.3V |
| GPIO_2[2] | PIN_C14 | GPIO Connection DATA[2] | 3.3V |
| GPIO_2[3] | PIN_C16 | GPIO Connection DATA[3] | 3.3V |
| GPIO_2[4] | PIN_C15 | GPIO Connection DATA[4] | 3.3V |
| GPIO_2[5] | PIN_D16 | GPIO Connection DATA[5] | 3.3V |
| GPIO_2[6] | PIN_D15 | GPIO Connection DATA[6] | 3.3V |
| GPIO_2[7] | PIN_D14 | GPIO Connection DATA [7] | 3.3V |
| GPIO_2[8] | PIN_F15 | GPIO Connection DATA[8] | 3.3V |
| GPIO_2[9] | PIN_F16 | GPIO Connection DATA[9] | 3.3V |
| GPIO_2[10] | PIN_F14 | GPIO Connection DATA[10] | 3.3V |
| GPIO_2[11] | PIN_G16 | GPIO Connection DATA[11] | 3.3V |
| GPIO_2[12] | PIN_G15 | GPIO Connection DATA[12] | 3.3V |
| GPIO_2_IN[0] | PIN_E15 | GPIO Input | 3.3V |
| GPIO_2_IN[1] | PIN_E16 | GPIO Input | 3.3V |
| GPIO_2_IN[2] | PIN_M16 | GPIO Input | 3.3V |

| Table 3-8 Pin Assignments for 2x13 Header |
|---|
|---|

 Table 3-9 Pin Assignments for ADC

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|-------------|--------------|---------------------|--------------|
| ADC_CS_N | PIN_A10 | Chip select | 3.3V |
| ADC_SADDR | PIN_B10 | Digital data input | 3.3V |
| ADC_SDAT | PIN_A9 | Digital data output | 3.3V |
| ADC_SCLK | PIN_B14 | Digital clock input | 3.3V |

3.7 Digital Accelerometer

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution measurement. This digital accelerometer can be accessed through a SPI 3-wire digital interface or I2C 2-wire digital interface. Main applications include medical instrumentation, industrial instrumentation, personal electronic aid and hard disk drive protection etc. Some of the key features of this device are listed below. For more detailed information, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.

- Up to 13-bit resolution at +/- 16g
- SPI (3- wire) or I2C (2-wire) digital interface
- Flexible interrupts modes

Figure 3-13 shows the connections between the ADXL345 and the Cyclone IV E device.

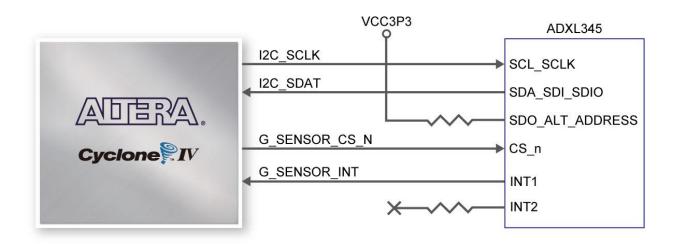


Figure 3-13 Wiring between the ADXL345 and the Cyclone IV E device

| Signal Name | FPGA Pin No. | Description | I/O Standard |
|---------------|--------------|----------------------|--------------|
| I2C_SCLK | PIN_F2 | EEPROM clock | 3.3V |
| I2C_SDAT | PIN_F1 | EEPROM data | 3.3V |
| G_SENSOR_INT | PIN_M2 | G_Sensor Interrupt | 3.3V |
| G_SENSOR_CS_N | PIN_G5 | G_Sensor chip select | 3.3V |

| Table 3-10 Pin Assignments for Digital Accelerometer | Table 3-10 | Pin Assignments | for Digital A | ccelerometer |
|--|-------------------|------------------------|---------------|--------------|
|--|-------------------|------------------------|---------------|--------------|

3.8 Clock Circuitry

The DE0-Nano board includes a 50 MHz oscillator. The oscillator is connected directly to a dedicated clock input pin of the Cyclone IV E FPGA. The 50MHz clock input can be used as a source clock to drive the phase lock loops (PLL) circuit. The clock distribution on the DE0-Nano board is shown in **Figure 3-14**.

terasic DE0-Nano User Manual

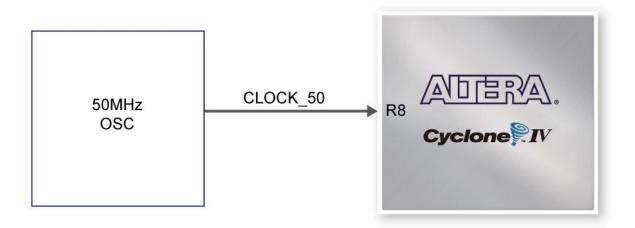
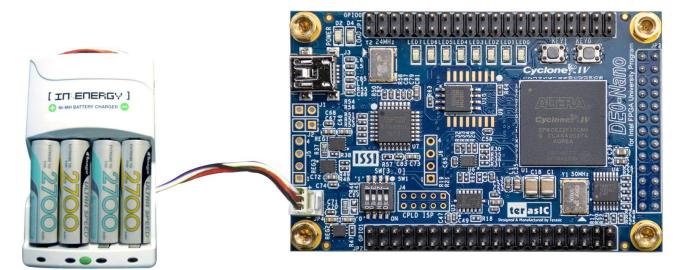


Figure 3-14 Block diagram of the clock distribution

3.9 Power Supply

The DE0-Nano board's power is provided through the USB 5V power, the 5V VCC pins on the two 40-pin headers or the 2-pin power header. The DC voltage is then stepped down to various required voltages. For portable project applications, connect a battery power supply (3.6~5.7V) to the 2-pin external power header shown in **Figure 3-15**.







Power Distribution System

Figure 3-16 shows the power distribution system on the DE0-Nano board.

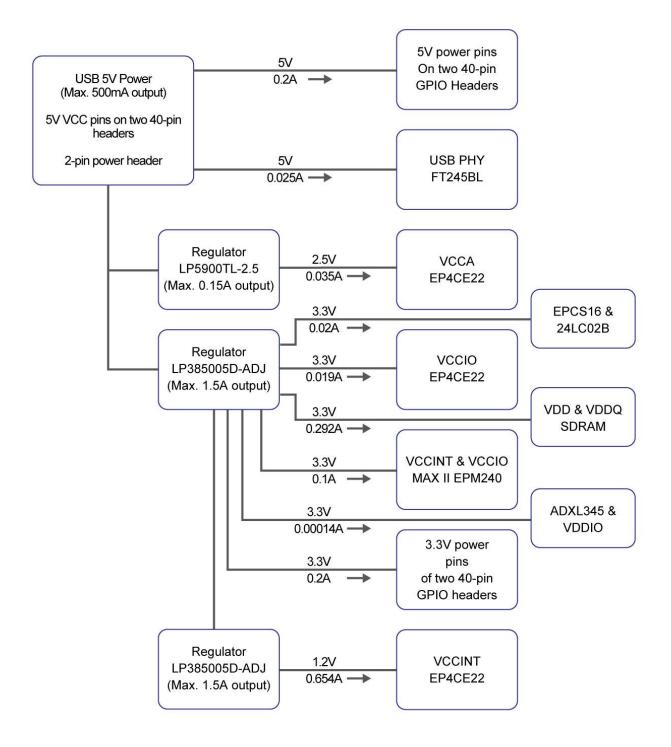


Figure 3-16 DE0-Nano Power Distribution System



Chapter 4

DE0-Nano Control Panel

The DE0-Nano board comes with a Control Panel facility that allows users to access various components on the board from a host computer. The host computer communicates with the board through a USB connection. The facility can be used to verify the functionality of components on the board or be used as a debug tool while developing RTL code.

This chapter first presents some basic functions of the Control Panel, then describes its structure in block diagram form, and finally describes its capabilities.

4.1 Control Panel Setup

The Control Panel Software Utility is located in the directory *"tools/DE0_NANO_ControlPanel"* in the **DE0-Nano System CD**. It's free of installation, just copy the whole folder to your host computer and launch the control panel by executing the "DE0_NANO_ControlPanel.exe".

When Control Panel starts it will attempt to download a configuration file onto the DEO-Nano board. The configuration file contains a design that communicates with the peripheral devices on the board that are attached to the FPGA device. Perform the following steps to ensure that the control panel starts up successfully:

- 1. Make sure Quartus II 10.0 or later version is installed successfully on your PC.
- 2. Connect a USB A to Mini-B cable to a USB (Type A) host port and to the board.

3. Start the executable DE0_NANO_ControlPanel.exe on the host computer. The Control Panel user interface shown in **Figure 4-1** will appear.

5. The DE0_NANO_ControlPanel.sof bit stream is loaded automatically as soon as the DE0_NANO_ControlPanel.exe is launched.

6. In case the connection is disconnected, click on CONNECT where the .sof will be re-loaded onto the board.

Note: the Control Panel will occupy the USB port until you choose to close the program or disconnect it from the board by clicking the Disconnect button. While the Control Panel is connected to the board, you will be unable to use Quartus II to download a configuration file into the FPGA.

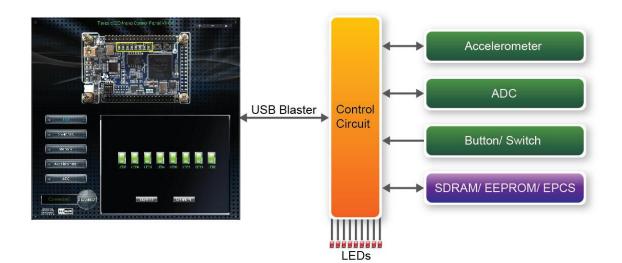


8. The Control Panel is now ready for use; experience it by setting the ON/OFF status for some LEDs and observing the result on the DE0-Nano board.



Figure 4-1 The DE0-Nano Control Panel

The concept of the DEO-Nano Control Panel is illustrated in **Figure 4-2**. The "Control Circuit" that performs the control functions is implemented in the FPGA board. It communicates with the Control Panel window, which is active on the host computer, via the USB Blaster link. The graphical interface is used to issue commands to the control circuit. It handles all requests and performs data transfers between the computer and the DEO-Nano board.





The DE0-Nano Control Panel can be used to light up LEDs, change the buttons/switches status, read/write to SDRAM Memory, read ADC channels, and display the Accelerometer information.

4.2 Controlling the LEDs

A simple function of the Control Panel is to allow setting the values displayed on LEDs. Choosing the **LED** tab displays the window in **Figure 4-3**. Here, you can directly turn the LEDs on or off individually or by clicking "Light All" or "Unlight All".



Figure 4-3 Controlling LEDs

4.3 Switches and Pushbuttons

Choosing the **Switches** tab displays the window in **Figure 4-4**. The function is designed to monitor the status of slide switches and pushbuttons in real time and show the status in a graphical user interface. It can be used to verify the functionality of the slide switches and pushbuttons.





Figure 4-4 Monitoring switches and buttons

The ability to check the status of pushbutton and slider switches is not needed in typical design activities. However, it provides a simple mechanism for verifying if the buttons and switches are functioning correctly. Thus, it can be used for troubleshooting purposes.

4.4 Memory Controller

The Control Panel can be used to write/read data to/from the SDRAM/EEPROM/EPCS on the DE0-Nano board. As an example, we will describe how the SDRAM may be accessed; the same approach is used to access the EEPROM and EPCS. Click on the Memory tab and select "SDRAM" to reach the window in **Figure 4-5**.





Figure 4-5 Accessing the SDRAM

A 16-bit word can be written into the SDRAM by entering the address of the desired location, specifying the data to be written, and pressing the Write button. Contents of the location can be read by pressing the Read button. **Figure 4-5** depicts the result of writing the hexadecimal value 06CA into offset address 200, followed by reading the same location.

The Sequential Write function of the Control Panel is used to write the contents of a file into the SDRAM as follows:

1. Specify the starting address in the Address box.

2. Specify the number of bytes to be written in the Length box. If the entire file is to be loaded, then a checkmark may be placed in the File Length box instead of giving the number of bytes.

3. To initiate the writing process, click on the Write a File to Memory button.

4. When the Control Panel responds with the standard Windows dialog box asking for the source file, specify the desired file in the usual manner.

The Control Panel also supports loading files with a .hex extension. Files with a .hex extension are ASCII text files that specify memory values using ASCII characters to represent hexadecimal values. For example, a file containing the line

0123456789ABCDEF

defines eight 8-bit values: 01, 23, 45, 67, 89, AB, CD, EF. These values will be loaded consecutively into the memory.

The Sequential Read function is used to read the contents of the SDRAM and fill them into a file as follows:

1. Specify the starting address in the Address box.

2. Specify the number of bytes to be copied into the file in the Length box. If the entire contents of the SDRAM are to be copied (which involves all 32 Mbytes), then place a checkmark in the Entire Memory box.

3. Press Load Memory Content to a File button.

4. When the Control Panel responds with the standard Windows dialog box asking for the destination file, specify the desired file in the usual manner.

Users can use the similar way to access the EEPROM and EPCS. Please note that users need to erase the EPCS before writing data to it.

4.5 Digital Accelerometer

The Control Panel can be used to display the status of the Digital Accelerometer where it measures the output of its 3-axis (X, Y, Z). The measurement range and resolution is set to default value $\pm 2g$ (acceleration of gravity) and 10bit twos complement respectively. Figure 4-6 shows the current digital accelerometer status of the DE0-Nano when Accelerometer tab is clicked. The units that are displayed are the raw register values converted to decimal. The value in parentheses is the gravitational acceleration values (mg) calculated from the register values according the formula. Table 4-1 shows the rule.

| Register Value | *Formula | Result (mg) |
|----------------|-----------|-------------|
| 0 | 0/511*2 | 0 |
| 1 | 1/511*2 | 3.9 |
| 2 | 2/511*2 | 6.8 |
| 17 | 17/511*2 | 66.4 |
| 511 | 511/511*2 | 2000 |

 Table 4-1 acceleration values convert rule



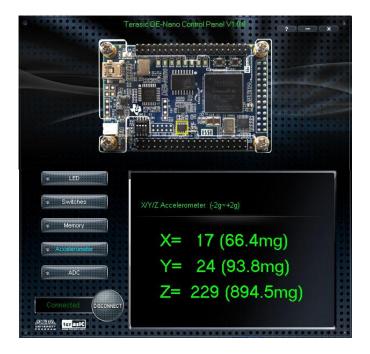


Figure 4-6 Digital Accelerometer status

4.6 ADC

From the Control Panel, users are able to view the eight-channel 12-bit analog-to-digital converter reading. The values shown are the ADC register outputs from all of the eight separate channels. The voltage shown is the voltage reading from the separate pins on the extension header. **Figure 4-7** shows the ADC readings when the ADC tab is chosen.



Figure 4-7 ADC Readings

4.7 Overall Structure of the DE0-Nano Control Panel

The DE0-Nano Control Panel is based on a Nios II SOPC system instantiated in the Cyclone IV E FPGA with software running on the on-chip memory. The software part is implemented in C code; the hardware part is implemented in Verilog HDL code with SOPC builder. The source code is not available on the DE0-Nano System CD.

To run the Control Panel, users should make the configuration according to Section 4.1. **Figure 4-8** depicts the structure of the Control Panel. Each input/output device is controlled by the Nios II Processor instantiated in the FPGA chip. The communication with the PC is done via the USB Blaster link. The Nios II interprets the commands sent from the PC and performs the corresponding actions.

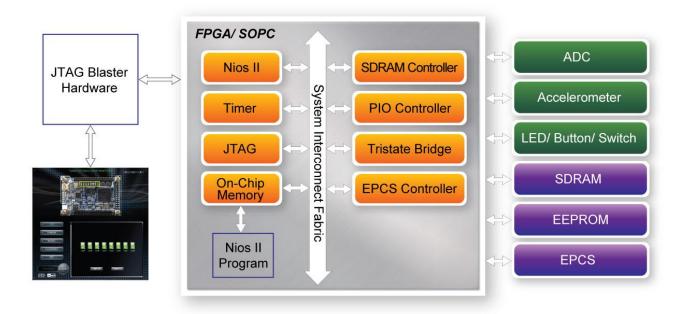


Figure 4-8 The block diagram of the DE0-Nano Control Panel



Chapter 5

DE0-Nano System Builder

This chapter describes how users can create a custom design project on the DE0-Nano board by using DE0-Nano Tool – DE0-Nano System Builder.

5.1 Introduction

The DE0-Nano System Builder is a Windows based software utility, designed to assist users in creating a Quartus II project for the DE0-Nano board within minutes. The generated Quartus II project files include:

- Quartus II Project File (.qpf)
- Quartus II Setting File (.qsf)
- Top-Level Design File (.v)
- Synopsys Design Constraints file (.sdc)
- Pin Assignment Document (.htm)

By providing the above files, DE0-Nano System Builder helps to prevents occurrence of situations that are prone to errors when users manually edit the top-level design file or place pin assignments. The common mistakes that users encounter are the following:

1. Board damaged for wrong pin/bank voltage assignments.

2. Board malfunction caused by wrong device connections or missing pin counts for connected ends.

3. Performance degeneration because of improper pin assignments.

5.2 General Design Flow

This section will introduce the general design flow to build a project for the DEO-Nano board via the DEO-Nano System Builder. The general design flow is illustrated in **Figure 5-1**.

To create a new system using the DEO-Nano System Builder, begin by launching the DEO-Nano System Builder software. The software will then prompt you to specify the name of the project you wish to create, as well as the components on the DEO-Nano board you wish to you. Once your specification is complete, you can generate the system.



The generated system is described using several files. In particular, there is the project file (.qpf), the top-level Verilog wrapper file (.v) that describes the I/O pins you will use in your design, and the Quartus II settings file (.qsf) that specifies which pin on the FPGA each I/O in your design should connect to. A Synopsys Design Constraints (.sdc) file with timing constraints and an HTML file with pin descriptions will be generated as well.

To proceed with your design, open the Quartus II CAD software and open your newly-created project. You will now be able to implement the logic of your design by describing your design in a hardware description language, and connecting it to I/Os in the top-level wrapper file. Once your design is complete, compile the design using Quartus II, and then use the Quartus II Programmer tool to configure the FPGA on the DEO-Nano board, using the JTAG programming mode.

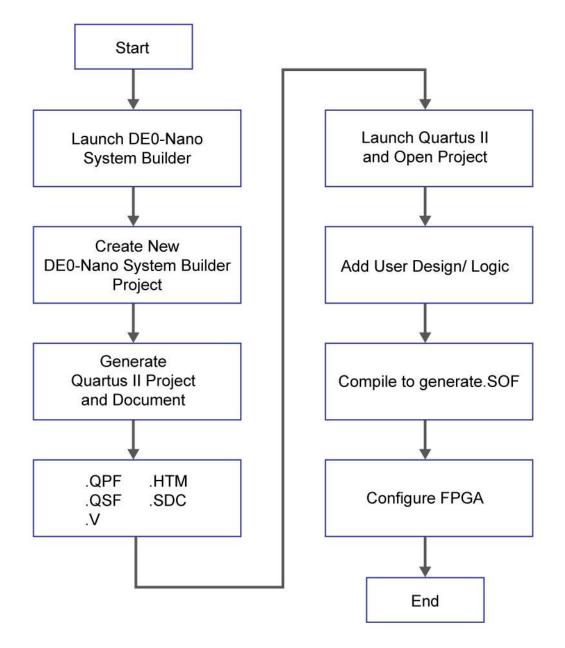


Figure 5-1 The general design flow of building a design

5.3 Using DE0-Nano System Builder

This section provides the detailed procedures on how the to use the DEO-Nano System Builder.

■ Install and launch the DE0-Nano System Builder

The DEO-Nano System Builder is located in the directory: "*Tools\DEO_NANO_SystemBuilder*" on the DEO-Nano System CD. Users can copy the whole folder to a host computer without installing the utility. Launch the DEO-Nano System Builder by executing the DEO_NANO_SystemBuilder.exe on the host computer and the GUI window will appear as shown in Figure 5-2.

| Terasic DEO-Nano System Builder ¥1. | 0.0 | | 🔺 🖬 🔀 |
|-------------------------------------|----------|--|---|
| | | System Configuration Project Name: DE0_NANO | |
| DE0-Nano FPGA | Board | ✓ CLOCK ✓ Button x 2 ✓ SDRAM, 32MB ✓ EEPROM, 2Kb ✓ Accelerometer GPIO-0 Header None Prefix Name: GPIO-1 Header | ✓ LED x 8 ✓ Dip Switch x 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header |
| Load Setting | Generate | None | ~ |
| Save Setting | Exit | Prefix Name: | |

Figure 5-2 The DE0-Nano System Builder window

■ Input Project Name

Input project name as show in Figure 5-3.

Project Name: Type in an appropriate name here, it will automatically be assigned as the name of your top-level design entity.



| Terasic DEO-Nano System Builder ¥1.0.0 | | | |
|--|----------|---|---|
| | | System Configuration Project Name: | |
| DEO-Nano FPGA Board | | CLOCK CLOCK SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header None Prefix Name: GPIO-1 Header | ✓ LED × 8 ✓ Dip Switch × 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header |
| Load Setting | Generate | None | ~ |
| Save Setting | Exit | Prefix Name: | |

Figure 5-3 The DE0-Nano Board Type and Project Name

System Configuration

Under System Configuration users are given the flexibility of enabling their choice of included components on the DEO-Nano as shown in **Figure 5-4**. Each component of the DEO-Nano is listed where users can enable or disable a component according to their design by simply marking a check or removing the check in the field provided. If the component is enabled, the DEO-Nano System Builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and I/O standard.

| Terasic DEO-Nano System Builder ¥1 | 0.0 | | 🛋 🖬 🔀 |
|------------------------------------|----------|---|---|
| | | System Configuration Project Name: DE0_NANO | |
| DEO-Nano FPGA | | CLOCK C Button x 2 SDRAM, 32MB E EPROM, 2Kb Accelerometer GPIO-0 Header None Prefix Name: GPIO-1 Header | ✓ LED x 8 ✓ Dip Switch x 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header |
| Load Setting | Generate | None | ~ |
| Save Setting | Exit | Prefix Name: | |

Figure 5-4 System Configuration Group

GPIO Expansion



Users can connect GPIO expansion card onto GPIO header located on the DEO-Nano board as shown in **Figure 5-5**. Select the appropriate daughter card you wish to include in your design from the drop-down menu. The system builder will automatically generate the associated pin assignments including the pin name, pin location, pin direction, and IO standard.

If a customized daughter board is used, users can select "GPIO Default" followed by changing the pin name and pin direction according to the specification of the customized daughter board.

| Terasic DEO-Nano System Builder ¥1.0.0 | | |
|--|---|---|
| | System Configuration Project Name: DE0_NANO | |
| DEO-Nano FPGA Board | CLOCK Button x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header Prefix Name: GPIO-1 Header | ✓ LED x 8 ✓ Dip Switch x 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header |
| Load Setting Generate | None Prefix Name: | |
| Save Setting Exit | | |

Figure 5-5 GPIO Expansion Group

The "Prefix Name" is an optional feature which denotes the prefix pin name of the daughter card assigned in your design. Users may leave this field empty.

Project Setting Management

The DE0-Nano System Builder also provides functions to restore default setting, loading a setting, and saving users' board configuration file shown in **Figure 5-6**. Users can save the current board configuration information into a .cfg file and load it to the DE0-Nano System Builder.



| Terasic DEO-Nano System Builder ¥1.0.0 | | |
|--|--|---|
| | System Configuration Project Name: DE0_NANO | |
| DEO-Nano FPGA Board | CLOCK Button x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header Prefix Name: GPIO-1 Header | |
| Load Setting Generate | None | ~ |
| Save Setting Exit | Prefix Name: | |

Figure 5-6 Project Settings

Project Generation

When users press the Generate button, the DEO-Nano System Builder will generate the corresponding Quartus II files and documents as listed in the **Table 5-1**:

| Table 5-1 | The files generated by DE0-Nano Sys | tem Builder |
|-----------|-------------------------------------|-------------|
| Table 3-1 | The mes generated by DE0-1 and bys | tem Dunuer |

| No. | Filename | Description |
|-----|---------------------------------|---|
| 1 | <project name="">.v</project> | Top level Verilog HDL file for Quartus II |
| 2 | <project name="">.qpf</project> | Quartus II Project File |
| 3 | <project name="">.qsf</project> | Quartus II Setting File |
| 4 | <project name="">.sdc</project> | Synopsys Design Constraints file for Quartus II |
| 5 | <project name="">.htm</project> | Pin Assignment Document |

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



Chapter 6

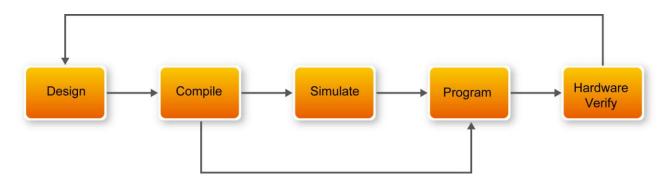
Tutorial: Creating an FPGA Project

This tutorial provides comprehensive information for understanding how to create a FPGA design and run it on the DEO-Nano development and education board. The following sections provide a quick overview of the design flow, explaining what is needed to get started, and describe what is taught in this tutorial.

6.1 Design Flow

Figure 6-1 shows a block diagram of the FPGA design flow.

The first step in the FPGA design flow starts is design entry. The standard design entry methods are using schematics or a hardware description language (HDL), such as Verilog HDL or VHDL. The design entry step is where the designer creates the digital circuit to be implemented inside the FPGA. The flow then proceeds through compilation, simulation, programming, and verification in the FPGA hardware.





This tutorial describes all of the steps except for simulation. Although it is not covered in this document, simulation is very important to learn. There are two types of simulation, Functional and Timing Functional simulation allows you to verify that your hardware is performing the desired functionality. Timing (or post place-and-route) simulation verifies that the design meets timing and functions appropriately in the device. Simulation tutorials can be found on the Altera University Program website at http://university.altera.com.



6.2 Before You Begin

This tutorial assumes the following prerequisites

■ You have a general understanding of FPGAs. This tutorial does not explain the basic concepts of programmable logic.

■ You are somewhat familiar with digital circuit design and electronic design automation (EDA) tools.

You have installed the Altera Quartus II 10.1 software on your computer. If you do not have the Quartus II software, you can download it from the Altera web site at www.altera.com/download.

■ You have a DE0-Nano Development Board on which you will test your project. Using a development board helps you to verify whether your design is really working.

■ You have gone through the quick start guide and/or the getting started user guide for your development kit. These documents ensure that you have:

- Installed the required software.
- Determined that the development board functions properly and is connected to your computer.

Next step is to install the USB-Blaster driver, if not already done. To install the driver, connect a USB cable between the DE0-Nano board and a USB port on a computer that is running the Quartus II software.

The computer will recognize the new hardware connected to its USB port, but it will be unable to proceed if it does not have the required driver already installed. If the USB-Blaster driver is not already installed, the New Hardware Wizard in **Figure 6-2** will appear.





Figure 6-2 Found New Hardware Wizard

The desired driver is not available on the Windows Update Web site, therefore select "No, not this time" and click **Next**. This leads to the window in **Figure 6-3**.



Figure 6-3 The driver is found in a specific location



The driver is available within the Quartus II software. Hence, select "Install from a list or specific location" and click **Next** to get to **Figure 6-4**.

| Found New Hardware Wizard |
|--|
| Please choose your search and installation options. |
| Search for the best driver in these locations. |
| Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed. |
| Search removable media (floppy, CD-ROM) |
| Include this location in the search: |
| C:\altera\10.1\quartus\drivers\usb-blaster Srowse |
| O Don't search. I will choose the driver to install. |
| Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware. |
| |
| < Back Next > Cancel |

Figure 6-4 Specify the location of the driver

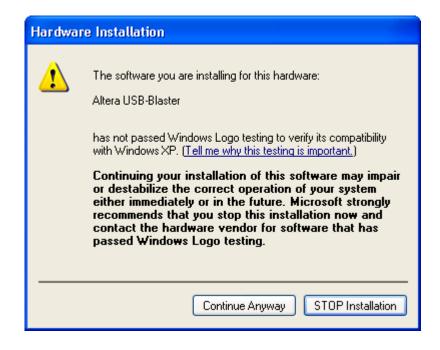
Now, select "Search for the best driver in these locations" and click Browse to get to the pop-up dialog box in **Figure 6-5** Find the desired driver, which is at location

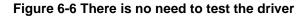
C:\altera\10.1\quartus\drivers\usb-blaster. Click OK and then upon returning to **Figure 6-4** click Next. At this point the installation will commence, but a dialog box in **Figure 6-6** will appear indicating that the driver has not passed the Windows Logo testing. Click Continue Anyway.



| Browse For Folder | ? 🗙 |
|--|-------|
| Select the folder that contains drivers for your hardware. | |
| | |
| 🖃 🧰 10.1 | ~ |
| 🗉 🧰 installer | _ |
| 🖽 🧰 ip | |
| 🗉 🧰 nios2eds | |
| 🖃 🧰 quartus | |
| 🗄 🧰 bin | = |
| 🗄 🧰 bin64 | |
| 🗄 🧰 common | |
| 🗄 🧰 cusp | |
| 🖃 🧰 drivers | |
| i386 | |
| 🗉 🧰 sentinel | |
| 🖃 🧰 usb-blaster | |
| 🛅 x32 | |
| 🛅 x64 | |
| 🗉 🧰 dsp_builder | |
| l 🕀 🖻 eda | |
| To view any subfolders, click a plus sign above. | |
| ОКС | ancel |

Figure 6-5 Browse to find the location







The driver will now be installed as indicated in **Figure 6-7**. Click **Finish** and you can start using the DE0-Nano board.

Figure 6-7 The driver is installed

6.3 What You Will Learn

In this tutorial you will perform the following tasks:

Create a design that causes LEDs on the development board to blink at two distinct rates. This design is easy to create and gives you visual feedback that the design works. Of course, you can use your DE0-Nano board to run other designs as well. For the LED design, you will write Verilog HDL code for a simple 32-bit counter, add a phase-locked loop (PLL) megafunction as the clock source, and add a 2-input multiplexer megafunction. When the design is running on the board, you can press an input switch to multiplex the counter bits that drive the output LEDs.

6.4 Assign The Device

Begin this tutorial by creating a new Quartus II project. A project is a set of files that maintain information about your FPGA design. The Quartus II Settings File (.qsf) and Quartus II Project File (.qpf) files are the primary files in a Quartus II project. To compile a design or make pin assignments, you must first create a project. The steps used to create a project are:



1. In the Quartus II software, select **File > New Project Wizard**. The Introduction page opens, as shown in **Figure 6-8**.

| 🐇 New Proj | ect Vizard | × |
|---------------------------------|--|---|
| Introduc | tion | |
| The New Proje | ct Wizard helps you create a new project and preliminary project settings, including the following: | |
| • | Project name and directory | |
| | Name of the top-level design entity Project files and libraries Target device family and device | |
| : | EDA tool settings | |
| You can chang the various pa | e the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use ges of the Settings dialog box to add functionality to the project. | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| Don't show | rme this introduction again | |
| | | |
| | < Back Next > Einish Cancel Help | |

Figure 6-8 New Project Wizard introduction

2. Click Next.

3. Enter the following information about your project: (Note: File names, project names, and directories in the Quartus II software cannot contain spaces.)

a. What is the working directory for this project? Enter a directory in which you will store your Quartus II project files for this design. For example, **E:\My_design\my_first_fpga**.

b. What is the name of this project? Type **my_first_fpga**.

c. What is the name of the top-level design entity for this project? Type **my_first_fpga**. See **Figure 6-9**.



| 😗 New Project Wizard | X |
|--|---|
| Directory, Name, Top-Level Entity [page 1 of 5] | |
| What is the working directory for this project? | |
| E:\My_design\my_first_fpga | |
| What is the name of this project? | |
| my_first_fpga | |
| What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. | |
| my_first_fpga | |
| Use Existing Project Settings | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| < Back Next > Finish Cancel Help | |

Figure 6-9 Project information

d. Click Next.

e. In the next dialog box, you will assign a specific FPGA device to the design. Select the **EP4CE22F17C6** device, as it is the FPGA on the DE0-Nano, as shown in **Figure 6-10**.



| Family Cycles 7 | | | | | Show in Avail | able devices' list | | |
|-------------------------------------|------------------------|----------------|-----------|------------------|-----------------------|------------------------------------|-----|----|
| Eamily: Cyclone IV E | | | | Package: Any | | | ~ | |
| | | | | ~ | Pin count: | Any | | ~ |
| | | | | | _ | | | |
| Target device | | | | | Speed grade: | Any | | ~ |
| Auto device sel | elected by the Fitter | | | | Show advanced devices | | | |
| - | | | h | | | | | |
| Specific device | selected in 'Available | e devices' | list | | HardCopy | compatible only | | |
| Other: n/a | | | | | | | | |
| Name | Core Voltage | LEs | User I/Os | | - | Embedded multiplier 9-bit elements | PLL | |
| | 1.2V 1.0V | 22320 22320 | 80 80 | 608256 608256 | 13 | | 4 | 20 |
| | 1.2V | 22320 | 154 | 608256 | 13 | | 4 | 20 |
| | 1.2V | 22320 | 154 | 608256 | 13 | - | 4 | 20 |
| | 1.2V | 22320 | 154 | 608256 | 13 | | 4 | 20 |
| | 1.2V | 22320 | 154 | 608256 | 13 | | 4 | 20 |
| EP4CE22F17C8L 1 | 1.0V | 22320 | 154 | 608256 | 13 | 2 | 4 | 20 |
| | 1.01/ | 22220 | 154 | 000050 | 10 | - · | | × |
| C | | | | | | | | > |

Figure 6-10 Specify the Device Example

f. Click Finish.

4. When prompted, select **Yes** to create the my_first_fpga project directory. You just created your Quartus II FPGA project. Your project is now open in Quartus II, as shown in **Figure 6-11**.



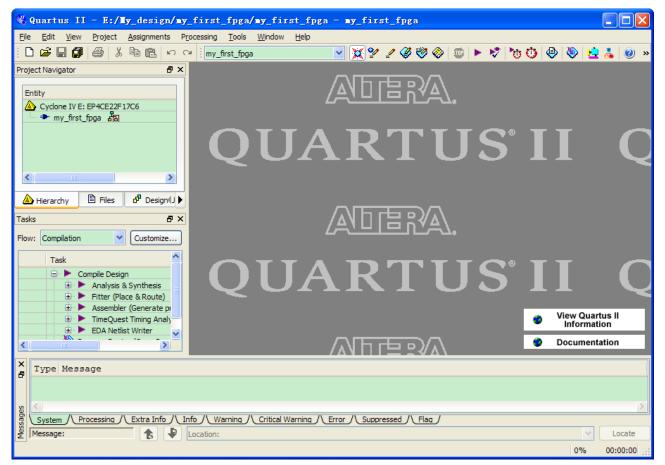


Figure 6-11 my_first_fpga project

6.5 Creating an FPGA design

This section describes how to create an FPGA design. This includes creating the top-level design, adding components (in Verilog HDL and using the megafunctions), adding pins and interconnecting all the components and pins.

First, create a top-level module. In this tutorial, you will use schematic entry, via a Block Design File (.bdf). Alternatively, you could use Verilog HDL or VHDL for the top-level module. The following steps describe how to create the top-level schematic.

1. Select File > New > Block Diagram/Schematic File (see Figure 6-12 to create a new file, Block1.bdf, which you will save as the top-level design.



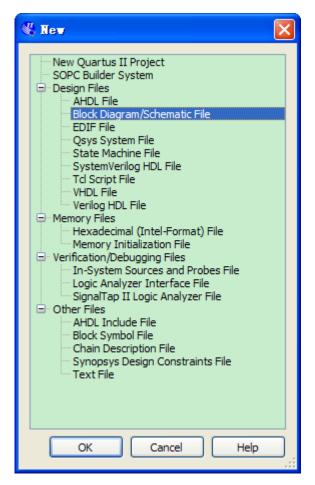


Figure 6-12 New BDF

- 2. Click OK.
- 3. Select **File > Save As** and enter the following information.
 - File name: my_first_fpga
 - Save as type: Block Diagram/Schematic File (*.bdf)
- 4. Click Save. The new design file appears in the Block Editor (see Figure 6-13).



| 🍕 Quartus II - E:/ Iy _design/my | /_first_fpga/my_first_fpga - my_first_fpga | |
|--|--|----------|
| | rocessing <u>T</u> ools <u>W</u> indow <u>H</u> elp | |
| n n 🖨 🖬 🖓 🔂 🖬 🛍 🗠 n | 🗠 : [my_first_fpga 💦 😵 😰 🖉 🤣 🐨 🕨 💌 🦑 😓 🌡 | 🐛 🕜 » |
| Project Navigator 🗗 🗙 | 📸 my_first_fpga.bdf 🛛 | |
| Entity | ░▩◣◕◭◴▰▾▯ヿヿヿヽヽヽੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑੑ | l 🞒 » |
| Cyclone IV E: EP4CE22F17C6 | | |
| → my_first_fpga 品 | | |
| | | |
| | | |
| | | |
| < | | |
| Hierarchy | | |
| Hierarchy 🖹 Files d ⁹ Design(J) | | |
| Tasks 🗗 🗙 | · · · · · · · · · · · · · · · · · · · | ::::: |
| Flow: Compilation V Customize | | |
| | | |
| Task | | |
| Compile Design | | |
| Fitter (Place & Route) | | |
| 🕀 🕨 Assembler (Generate p | | |
| TimeQuest Timing Analy DA Netlist Writer | | |
| EDA Netilst Writer | | 11111 🗹 |
| | | |
| × Type Message | | |
| 5 | | |
| | | |
| System / Processing / Extra Info // Ir | nfo 八 Warning 八 Critical Warning 八 Error 八 Suppressed 八 Flag ノ | <u> </u> |
| | | Locate |
| | 638, 218 0% 0 | 00:00:00 |

Figure 6-13 Bank BDF

• Adding a Verilog HDL to the Schematic

- 1. Add HDL code to the blank block diagram by choosing File > New > Verilog HDL File.
- 2. Select Verilog HDL File in the tree and Click OK.

3. Save the newly created file, by selecting **File** > **Save As** and entering the following information (see Figure 6-14).

- File name: simple_counter.v
- Save as type: Verilog HDL File (*.v, *.vlg, *.verilog)



| Save As | | | | | ? 🛛 |
|---|-----------------------------|--|---------|-------|----------------|
| Save in: My Recent Documents Desktop My Documents My Documents | my_first_fpga | ux.v ux_bb.v | • | | |
| My Network Places | File name: Save as type: | simple_counter.v Verilog HDL Files (*.v *.vlg *.ve Add file to current project | erilog) | • | Save Cancel |

Figure 6-14 Saving the Verilog HDL file

The resulting empty file is ready for you to enter the Verilog HDL code.

4. Type the following Verilog HDL code into the blank simple_counter.v file, as shown in **Figure 6-15**.

//It has a single clock input and a 32-bit output port

module simple_counter (

CLOCK_5,

counter_out

);

input CLOCK_5;

output [31:0] counter_out;

reg [31:0] counter_out;



always @ (posedge CLOCK_5)

// on positive clock edge

begin

counter_out <= counter_out + 1;// increment counter</pre>

end

endmodule

// end of module counter

```
1
      //It has a single clock input and a 32-bit output port
 2
    -module simple counter
 3
                                CLOCK 5,
 4
                                counter_out
 5
                               );
 6
      input
                        CLOCK 5 ;
7
      output
                [31:0] counter out;
8
                [31:0] counter out;
      req
9
10
      always @ (posedge CLOCK 5)
                                                 // on positive clock edge
11
    Ξ
         begin
             counter_out <= counter_out + 1;</pre>
12
                                                 // increment counter
13
         end
      endmodule
                                                 // end of module counter
14
15
```

Figure 6-15 The Verilog File of simple_counter.v

5. Save the file by choosing **File > Save**, pressing **Ctrl + S**, or by clicking the floppy disk icon.

6. Select **File > Create/Update > Create Symbol Files for Current File** to convert the **simple_counter.v** file to a Symbol File (.sym). You will use this Symbol File to add the HDL code to your schematic.

The Quartus II software creates a Symbol File and displays a message (see Figure 6-16).

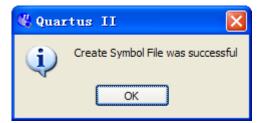


Figure 6-16 Create Symbol File was Successful

- 7. Click OK.
- 8. To add the **simple_counter.v** symbol to the top-level design, click the **my_first_fpga.bdf** tab.



- 9. Right click in the blank area of the BDF file, and select **Insert > Symbol**.
- 10. Double-click the Project directory to expand it.
- 11. Select the newly created simple_counter symbol by clicking its icon.

| 🔁 Symbol | Σ |
|--|--|
| Libraries: | |
| ➡ ➡ Project ➡ ➡ d:/altera/10.1/quartus/libraries/ | Simple_counter CLOCK_5 counter_out[310] |
| Name: | inst |
| simple_counter | |
| Repeat-insert mode | |
| Insert symbol as block | |
| Launch MegaWizard Plug-In | |
| MegaWizard Plug-In Manager | |
| | OK Cancel |

Figure 6-17 Adding the Symbol to the BDF

12. Click OK.

13. Move the cursor to the BDF grid; the symbol image moves with the cursor. Click to place the simple_counter symbol onto the BDF. You can move the block after placing it by simply clicking and dragging it to where you want it and releasing the mouse button to place it. See **Figure 6-18**.

| | | • |
|---------------------------------------|--------------------------|---|
| • • | simple_counter . | • |
| · · · · · · · · · · · · · · · · · · · | CLOCK_5 counter_out[310] | • |
| · · · · | | • |
| · · | | • |
| · · | inst | • |
| · · | 1115L | • |
| · · | | • |
| | | |

Figure 6-18 Placing the simple_counter symbol



14. Press the **Esc key** or click an empty place on the schematic grid to cancel placing further instances of this symbol.

15. Save your project regularly.

Adding a Megafunction to the Schematic

Megafunctions, such as the ones available in the LPM, are pre-designed modules that you can use in FPGA designs. These Altera-provided megafunctions are optimized for speed, area, and device family. You can increase efficiency by using a megafunction instead of writing the function yourself. Altera also provides more complex functions, called MegaCore functions, which you can evaluate for free but require a license file for use in production designs. This tutorial design uses a PLL clock source to drive a simple counter. A PLL uses the on-board oscillator (DE0-Nano Board is 50 MHz) to create a constant clock frequency as the input to the counter. To create the clock source, you will add a pre-built LPM megafunction named ALTPLL.

1. Right click in the blank space in the BDF and select **Insert > Symbol** or click the Add Symbol icon on the toolbar.

2. Click the Megawizard Plug-in Manager button. The MegaWizard[®] Plug-In Manager appears, as shown in **Figure 6-19**.

| 🐇 🛛 🕹 🕷 🐇 🐇 🐇 | g-In Manager [page 1] |
|---|--|
| megafun Which ac O Edit O Copy | gaWizard Plug-In Manager helps you create or modify design files that contain custom variations of action do you want to perform? ate a new custom megafunction variation an existing custom megafunction variation y an existing custom megafunction variation ht (C) 1991-2010 Altera Corporation |
| | Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish |

Figure 6-19 Mega Wizard Plug-In Manager



3. Click Next.

4. In MegaWizard Plug-In Manager [page 2a], specify the following selections (see Figure 6-20):

```
a. Select I/O > ALTPLL.
```

b. Under "Which device family will you be using?" select the **Cyclone IV E** for DE0-Nano development board.

c. Under "Which type of output file do you want to create?" select Verilog HDL.

d. Under "What name do you want for the output file?" type pll at the end of the already created directory name.

e. Click Next.

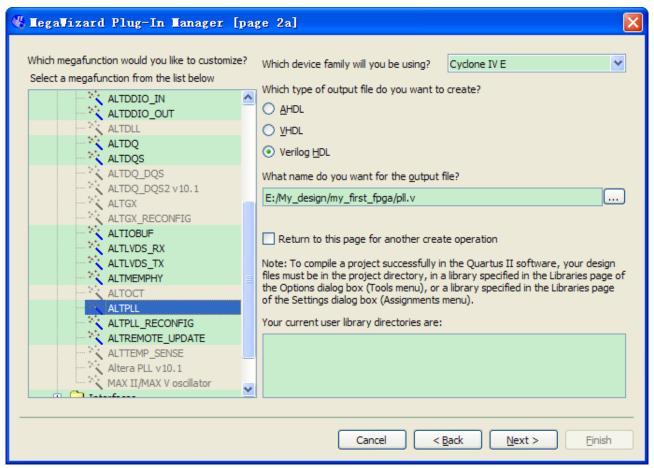


Figure 6-20 MegaWizard Plug-In Manager [page 2a] Selections

5. In the MegaWizard Plug-In Manager [page 3 of 14] window, make the following selections (see Figure 6-21).

- a. Confirm that the currently selected device family option is set to Cyclone IV E.
- b. For device speed grade choose 6 for DE0-Nano.
- c. Set the frequency of the inclock0 input 50 MHz.



d. Click Next.

| NegaVizard Plug-In Manager [page 3] | of 14] ? 🔀 |
|---|---|
| altpll | About Documentation |
| Parameter PLL 3 Output 4 EL Settings Reconfiguration Clocks 4 | DA 5 Summary |
| General/Modes $>$ Inputs/Lock $>$ Bandwidth/SS | Clock switchover |
| General/Modes Inputs/Lock Bandwidth/SS | Currently selected device family: Cydone IV E Cydone IV E Match project/default Able to implement the requested PLL General Which device speed grade will you be using? General Which device speed grade will you be using? Solooo MHz Solooo MHz Mobs PLL Type Which PLL type will you be using? PLT Type Which PLL type will you be using? PLT Type Which PLL type will you be using? PLT Type Which PLL type will you be using? Operation Mode How will the PLL outputs be generated? Use the feedback path inside the PLL Operation Mode How will the PLL outputs be generated? In source-synchronous compensation Mode In zero delay buffer mode Connect the fommic port (bidirectional) With no compensation |
| | Create an 'fbin' input for an external feedback (External Feedback Mode) Which output clock will be compensated for? |
| | Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish |

Figure 6-21 MegaWizard Plug-In Manager [page 3 of 14] Selections

6. Unselect all options on MegaWizard page 4. As you turn them off, pins disappear from the PLL block's graphical preview. See **Figure 6-22** for an example.



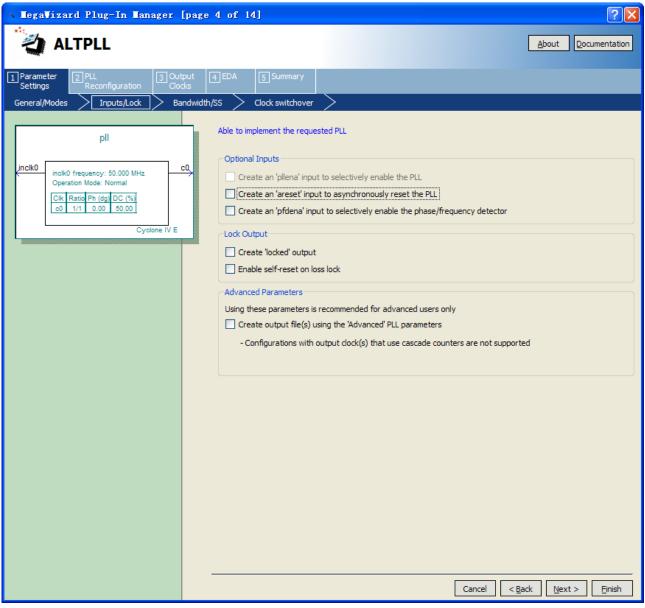


Figure 6-22 MegaWizard Plug-In Manager [page 4 of 14] Selections

- 7. Click **Next** four times to get to page 8.
- 8. Set the Clock division factor to 10, as shown in Figure 6-23.



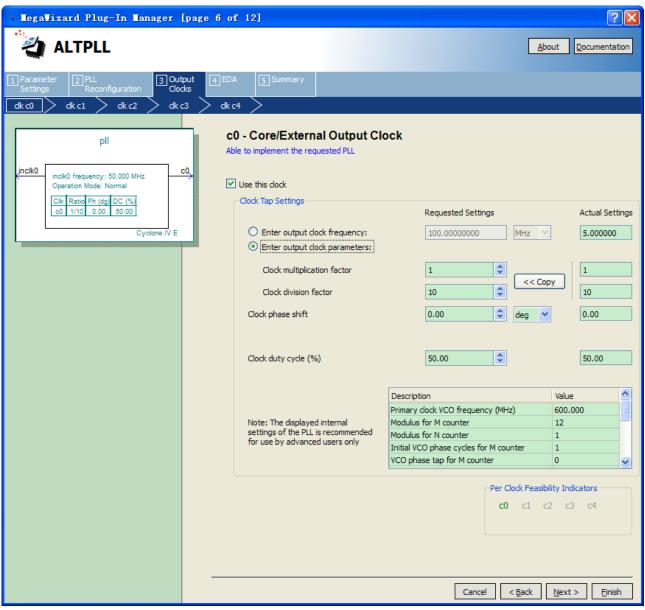


Figure 6-23 MegaWizard Plug-In Manager [page 8 of 14] Selections

9. Click **Next** and then click **Finish**.

10. The wizard displays a summary of the files it creates (see **Figure 6-24**). Select the pll.bsf option and click Finish again.



| 🔨 TegaVizard Plug-In Tanager [pag | ge 12 of 12] | ?X |
|---|---|--|
| altpll | | <u>A</u> bout <u>D</u> ocumentation |
| Parameter PLL Output Settings Reconfiguration Clocks | 4 EDA 5 Summary | |
| pl inclk0 frequency: 50.000 MHz Operation Mode: Normal Cik Ratio Ph (dg DC (%) 00 1/10 0.00 50.00 Cyclone IV E | checkmark indicates an op maintained in subsequent | to generate. A gray checkmark indicates a file that is automatically generated, and a green tional file. Click Finish to generate the selected files. The state of each checkbox is MegaWizard Plug-In Manager sessions. Anager creates the selected files in the following directory: ga\ Description Variation file PinPlanner ports PPF file AHDL Indudé file VHDL component declaration file Quartus II symbol file Instantiation template file Verilog HDL black-box file |
| | | Cancel < <u>Back</u> <u>N</u> ext > <u>Finish</u> |

Figure 6-24 Wizard-Created Files

The Symbol window opens, showing the newly created PLL megafunction, as shown in **Figure 6-25**.



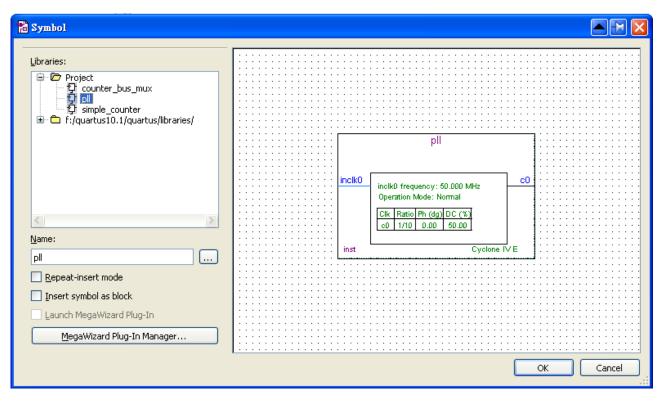


Figure 6-25 PLL Symbol

11. Click **OK** and place the pll symbol onto the BDF to the left of the simple_counter symbol. You can drag and drop the symbols, if you need to rearrange them. See **Figure 6-26**.

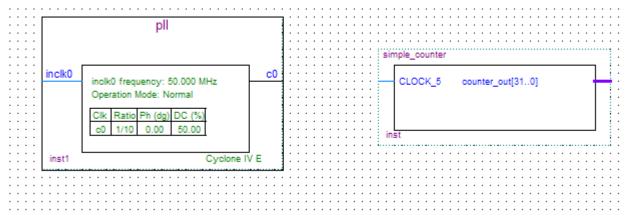


Figure 6-26 Place the PLL Symbol

12. Move the mouse so that the cursor (also called the selection tool) is over the pll symbol's c0 output pin. The orthogonal node tool (cross-hair) icon appears.

13. Click and drag a bus line from the c0 output to the simple_counter clock input. This action ties the pll output to the simple_counter input (see **Figure 6-27**).



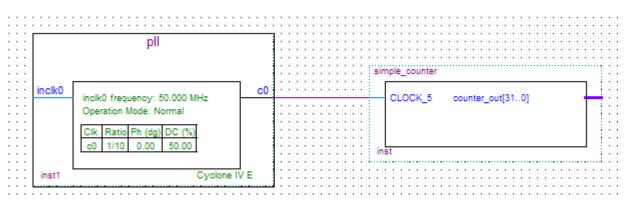


Figure 6-27 Draw a Bus Line connect pll c0 port to simple_counter CLOCK_5 port

■ Adding an Input pin to the Schematic

The following steps describe how to add an input pin to the schematic.

- 1. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 2. Under Libraries, select quartus/libraries > primitives > pin >input. See Figure 6-28
- 3. Click OK

If you need more room to place symbols, you can use the vertical and horizontal scroll bars at the edges of the BDF window to view more drawing space.

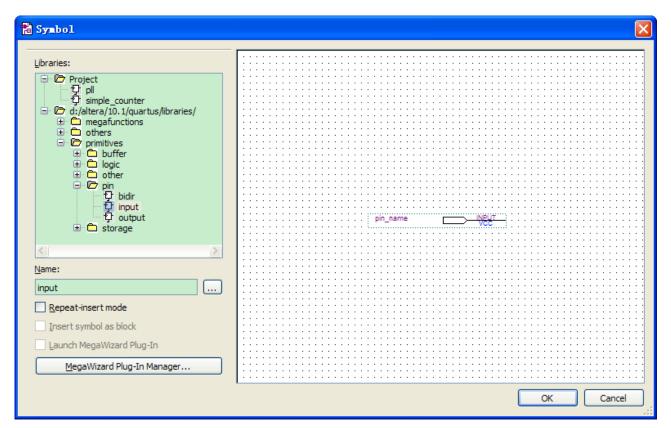


Figure 6-28 Input pin symbol



4. Place the new pin onto the BDF so that it is touching the input to the pll symbol.

5. Use the mouse to click and drag the new input pin to the left; notice that the ports remain connected as shown in **Figure 6-29**.

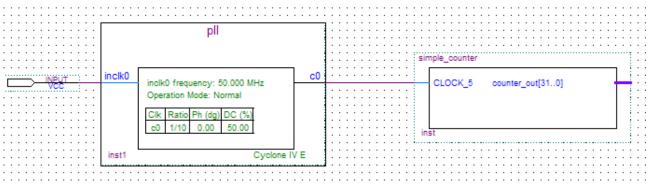


Figure 6-29 Connecting the PLL symbol and Input port

6. Change the pin name by double-clicking pin_name and typing CLOCK_50 (see **Figure 6-30**). This name correlates to the oscillator clock that is connected to the FPGA.

■ Adding an Output bus to the Schematic

The following steps describe how to add an output bus to the schematic.

1. Using the Orthogonal Bus tool, draw a bus line connected on one side to the simple_counter output port, and leave the other end unconnected at about 4 to 8 grid spaces to the right of the simple_counter.



| Pin Properties | × |
|--|---|
| General Format | |
| To create multiple pins, enter a name in AHDL bus notation (For example: "name[30]"), or enter a comma-seperated list of names. | |
| Pin name(s): CLOCK_50 | |
| Default value: VCC | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| OK Cancel Help | |

Figure 6-30 Change the input port name

2. Right-click the new output bus line and select **Properties**.

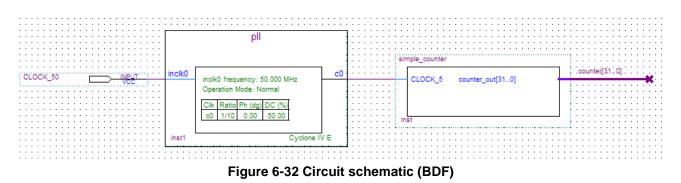
3. Type counter [31..0] as the bus name (see **Figure 6-31**). The notation [X ..Y] is the Quartus II method for specifying the bus width in BDF schematics, where X is the most significant bit (MSB) and Y is the least significant bit (LSB).

4. Click OK. **Figure 6-32** shows the BDF.



| 🖁 Bus Properties 🛛 🔀 | 3 |
|---------------------------------|-----|
| General Font Format | |
| | |
| Name: counter[310] | |
| Hide name in block design file. | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| OK Cancel Help | |
| | .:: |

Figure 6-31 Change the output BUS name



Adding a Multiplexer to the Schematic

This design uses a multiplexer to route the simple_counter output to the LED pins on the DEO-Nano development board. You will use the MegaWizard Plug-In Manager to add the multiplexer, lpm_mux. The design multiplexes two portions of the counter bus to four LEDs on the DEO-Nano board. The following steps describe how to add a multiplexer to the schematic.

65



- 1. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 2. Click Megawizard Plug-in Manager.
- 3. Click Next.
- 4. Select Installed Plug-Ins > Gates > LPM_MUX.

5. Select the **Cyclone IV E** device family, **Verilog HDL** as the output file type, and name the output file **counter_bus_mux.v**, as shown in **Figure 6-33**.

6. Click Next.

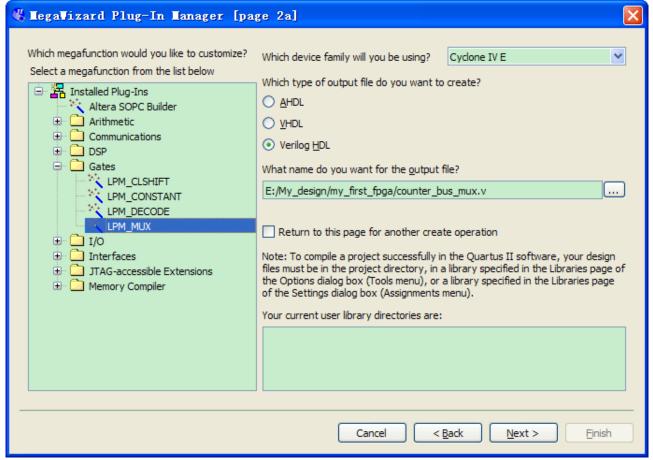


Figure 6-33 Selecting Ipm_mux

7. Under "How many 'data' inputs do you want?" select 2 inputs (default).

8. Under "How wide should the 'data' input and the 'result' output buses be?" select 4, as shown in **Figure 6-34**.



| MegaVizard Plug-In Manager [pag | e 3 of 5] 🤶 🔀 |
|---|---|
| LPM_MUX | <u>About</u> Documentation |
| 1 Parameter 2 EDA 3 Summary Settings | |
| sei result[3.0] data1x[3.0] result[3.0] | Currently selected device family: Cyclone IV E Match project/default How wide should the 'data' input and the 'result' output buses be? O you want to pipeline the multiplexer? No Yes, I want an output latency of 1 ock cycles Create an asynchronous Clear input Create a Clock Enable input |
| Resource Usage 1 lpm_mux | Cancel < <u>B</u> ack <u>N</u> ext > <u>Einish</u> |

Figure 6-34 Ipm_mux settings

- 9. Click Next.
- 10. Click Next.
- 11. Select the **counter_bus_mux.bsf** option.
- 12. Click **Finish**. The Symbol window appears (see **Figure 6-35** for an example).



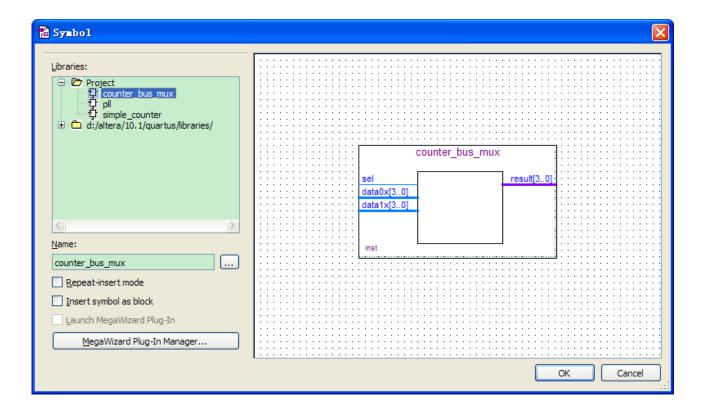
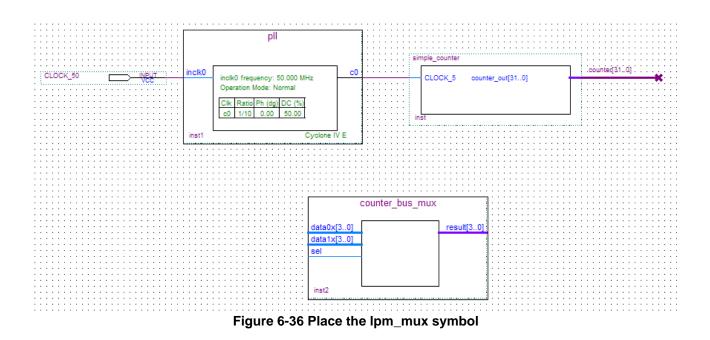


Figure 6-35 lpm_mux Symbol

13. Click OK

14. Place the **counter_bus_mux** symbol below the existing symbols on the BDF, as shown in **Figure 6-36**.





15. Add input buses and output pins to the counter_bus_mux symbol as follows:

a. Using the Orthogonal Bus tool, draw bus lines from the data1x[3..0] and data0x[3..0] input ports to about 8 to 12 grid spaces to the left of counter_bus_mux.

b. Draw a bus line from the result [3..0] output port to about 6 to 8 grid spaces to the right of counter_bus_mux.

c. Right-click the bus line connected to data1x[3..0] and select **Properties**.

d. Name the bus counter[26..23], which selects only those counter output bits to connect to the four bits of the data1x input.

Because the input busses to counter_bus_mux have the same names as the output bus from simple_counter, (counter[x .. y]) the Quartus II software knows to connect these busses.

- e. Click OK.
- f. Right-click the bus line connected to data0x[3..0] and select **Properties**.

g. Name the bus counter [24..21], which selects only those counter output bits to connect to the four bits of the data1x input.

h. Click OK. Figure 6-37 shows the renamed buses.

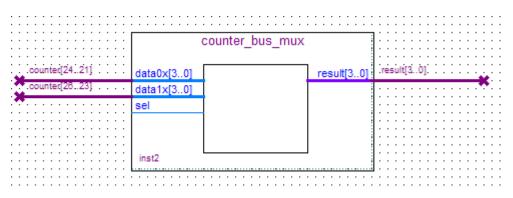


Figure 6-37 Renamed counter_bus_mux Bus Lines

If you have not done so already, you may want to save your project file before continuing.

- 16. Right click in the blank area of the BDF and select **Insert > Symbol**.
- 17. Under Libraries, select quartus/libraries > primitives > pin >output, as shown in Figure 6-38.



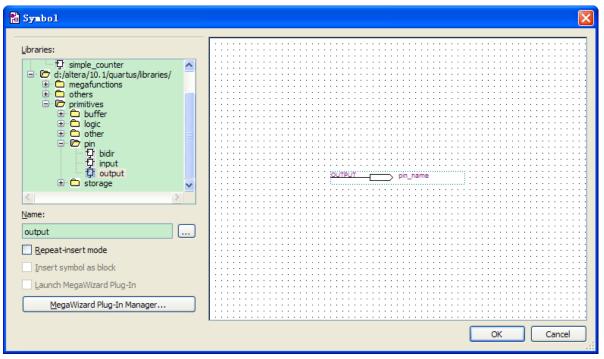


Figure 6-38 Choose output pin

- 18. Click OK.
- 19. Place this output pin so that it connects to the counter_bus_mux's result [3..0] bus output line.
- 20. Rename the output pin as LED [3..0]. (see Figure 6-39).

| · · · · · · · · · · · · · · · · · · · | | counter_bus_mux | | | | | | | | · · · · · | · · · · | · · · · · | · · · · · | · · · · | · · |
|---------------------------------------|------------|-----------------|------------|-----|---|---------|-----|-----|------|--------------|---------|--------------|--------------|---------|-----|
| | data0x[30] | | result[30] | res | | | : : | | | ЛТ | | | LEI | | .01 |
| counter[2623] | data1x[30] | | | 1 | | | | | | | | | | | |
| • | sel |] | | 1 | | | • • | • • | | • • | | | • • | • • • | • • |
| | | | | | • | | • • | • • | | | | | | | |
| | | | | | | | | | | | | | | | |
| | in st2 | | l | | | | • • | | | • • | | | | | |
| | | | |] | : | : : | : : | | | : : | | : : | :: | | |

Figure 6-39 Rename the output pin

- 21. Attach an input pin to the multiplexer select line using an input pin:
- a. Right click in the blank area of the BDF and select **Insert > Symbol**.
- b. Under Libraries, double-click quartus/libraries/ > primitives > pin > input.
- c. Click OK.
- 22. Place this input pin below **counter_bus_mux**.
- 23. Connect the input pin to the **counter_bus_mux** sel pin.
- 24. Rename the input pin as KEY [0] (see Figure 6-40).



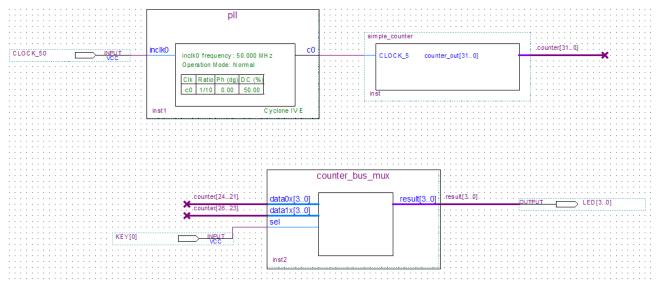


Figure 6-40 Adding the KEY [0] Input Pin

You have finished adding all required components of the circuit to your design. You can add notes or information to the project as text using the Text tool on the toolbar (indicated with the A symbol). For example, you can add the label "OFF = SLOW, ON = FAST" to the KEY [0] input pin and add a project description, such as "DE0-Nano Tutorial Project."

6.6 Assign the Pins

In this section, you will make pin assignments. Before making pin assignments, perform the following steps:

1. Select **Processing > Start > Start Analysis & Elaboration** in preparation for assigning pin locations.

2. Click **OK** in the message window that appears after analysis and elaboration completes.

To make pin assignments to the KEY [0] and CLOCK_50 input pins and to the LED[3..0] output pins, perform the following steps:

1. Select **Assignments > Pin Planner**, which opens the Pin Planner, a spreadsheet-like table of specific pin assignments. The Pin Planner shows the design's six pins. See **Figure 6-41**



| _ | | Planner - E:/Iy | | t_fpga /my _f | irst_fpga - my | _first_fpga | | | |
|---------------|----------|--|------------------------------|----------------------|----------------|-------------|--|-----------------|----------|
| File P | E Gro | | <u>T</u> ools <u>W</u> indow | ₽× | | | log View - Wire Bond ne IV E - EP¢CE33P17C9 | | |
| | Nam | ned: * 💙 | | | | 6888 | | | |
| : | (| Node Name KEY[00] | Direction Input Group | Location | | | | | |
| <u>₿</u> | | DED[30] | Output Group | | | | | | |
| €. ∭ | ÷ | < <new group="">></new> | | | | | | | |
| 8) E | < × | | | > | | | | | |
| 2 | ê | Named: * Node Name | Edit: X V Direction | Location | I/O Bank | VREF Group | I/O Standard | Filter: Pins: a | |
| Ci | | CLOCK_50 | Input | Eocadon | 1/0 00110 | With Group | 2.5 V (default) | Reserved | |
| | | KEY[0] LED[3] | Input Output | | | | 2.5 V (default) 2.5 V (default) | | |
| | | LED[2] | Output | | | | 2.5 V (default) | | |
| E | | LED[1] LED[0] | Output Output | | | | 2.5 V (default) 2.5 V (default) | | |
| 28 | | < <new node="">></new> | | | | | | |] |
| B | All Pins | | | | | | | | |
| × | All | | | | | | | | |
| | | | | | | | | 0% | 00:00:00 |

Figure 6-41 Pin Planner Example

2. In the Location column next to each of the six node names, add the coordinates (pin numbers) as shown in Table 6-1 for the actual values to use with your DEO-Nano board.

| Table 0-1 1 in finformation betting | |
|-------------------------------------|-------------------|
| Pin Name | FPGA Pin Location |
| KEY[0] | J15 |
| LED[3] | A11 |
| LED[2] | B13 |
| LED [1] | A13 |
| LED [0] | A15 |
| CLOCK 50 | R8 |

Table 6-1 Pin Information Setting

Double-click in the Location column for any of the six pins to open a drop-down list and type the location shown in the table. Alternatively, you can select the pin from a drop-down list. For example, if you type F1 and press the Enter key, the Quartus II software fills in the full PIN_F1 location name for you. The software also keeps track of corresponding FPGA data such as the I/O bank and VREF Group. Each bank has a distinct color, which corresponds to the top-view wire bond drawing in the upper right window, as shown in Figure 6-42.



| _ | | Planner - E:/Ey_ | | t_fpga/ ny _fi | rst_fpga - my | _first_fpga | | | - 🗆 🗙 |
|--|----------|--------------------------------------|------------------------------|-----------------------|---------------|-------------|--|-----------------|----------|
| File | Ē | dit <u>V</u> iew P <u>r</u> ocessing | <u>T</u> ools <u>W</u> indow | | | | | | |
| Ę | Gro | ups | | ₽× | | | p View - Wire Bond a M E - EPecE22F17Ce | | |
| ۶ <u>۷</u> | Nam | ned: * 💉 💙 | | | | | | | |
| | | Node Name | Direction | Locati 🛆 | | 8088 | 200000000000000000000000000000000000000 | | |
| | 1 | | | N B13 | | | | | |
| $\left[\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\$ | | | | N_A13 | | 8800 | | | |
| Ð, | | | | N_A15 | | | | | |
| ۳ | L | < <new aroun="">></new> | | ► | | ACOC. | | | |
| | < | | | | | 2 | | | |
| E | × | Named: * 💉 🗙 | Edit: 🗙 🗸 | | | | | Filter: Pins: a | I 🔽 |
| ю | 8 | Node Name | Direction | Location | I/O Bank | VREF Group | I/O Standard | Reserved | |
| Ci l | | CLOCK_50 | Input | PIN_R8 | 3 | B3_N0 | 2.5 V (default) | | |
| _ | | KEY[0] | Input | PIN_J15 | 5 | B5_N0 | 2.5 V (default) | | |
| ::: | | LED[3] | Output | PIN_A11 | 7 | B7_N0 | 2.5 V (default) | | |
| ₩¥ | | LED[2] | Output | PIN_B13 | 7 | B7_N0 | 2.5 V (default) | | |
| E | | LED[1] | Output | PIN_A13 | 7 | B7_N0 | 2.5 V (default) | | |
| | | LED[0] | Output | PIN_A15 | 7 | B7_N0 | 2.5 V (default) | | |
| 21 | | < <new node="">></new> | | L | | | | | |
| 3 | s | | | | | | | | |
| × | All Pins | | | | | | | | |
| | | | | | | | | 0% | 00:00:00 |

Figure 6-42 Completed Pin Planning Example

Now, you are finished creating your Quartus II design!

6.7 Create a Default TimeQuest SDC File

Timing settings are critically important for a successful design. For this tutorial you will create a basic Synopsys Design Constraints File (.sdc) that the Quartus II TimeQuest Timing Analyzer uses during design compilation. For more complex designs, you will need to consider the timing requirements more carefully.

To create an SDC, perform the following steps:

- 1. Open the TimeQuest Timing Analyzer by choosing **Tools > TimeQuest Timing Analyzer**.
- 2. Select **File > New SDC file**. The SDC editor opens.
- 3. Type the following code into the editor:

create_clock -period 20.000 -name CLOCK_50

derive_pll_clocks

derive_clock_uncertainty

4. Save this file as my_first_fpga.sdc (see Figure 6-43)



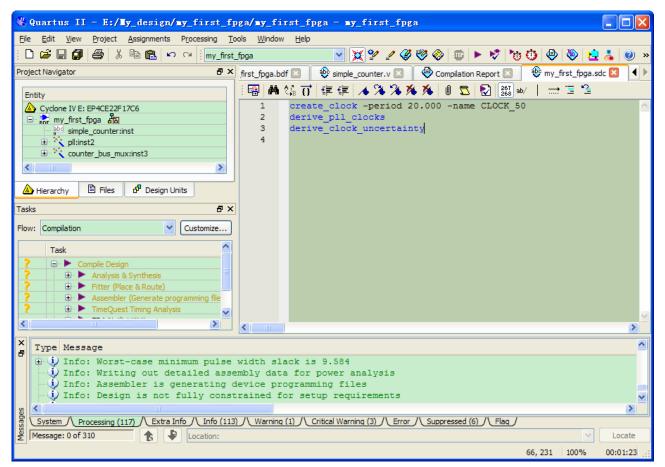


Figure 6-43 Default SDC

Naming the SDC with the same name as the top-level file causes the Quartus II software to use this timing analysis file automatically by default. If you used another name, you would need to add the SDC to the Quartus II assignments file.

6.8 Compile Your Design

After creating your design you must compile it. Compilation converts the design into a bitstream that can be downloaded into the FPGA. The most important output of compilation is an SRAM Object File (.sof), which you use to program the device. Also, the software generates report files that provide information about your circuit as it compiles.

Now that you have created a complete Quartus II project and entered all assignments, you can compile the design.

In the **Processing** menu, select **Start Compilation** or click the **Play** button on the toolbar.

If you are asked to save changes to your BDF, click Yes.

While compiling your design, the Quartus II software provides useful information about the compilation, as shown in **Figure 6-44**.



| 🐇 Quartus II - E:/My_design/my_first_fp | ga/ ny_ first_fpga - ny_ first_fpga | 1 | |
|--|--|--|--|
| File Edit View Project Assignments Processing To | ols <u>W</u> indow <u>H</u> elp | | |
| : 🗅 🖨 🗐 🖨 👗 🖻 🖺 ト റ 🎼 📺 👘 | fpga 💽 💥 🐓 🖉 🦉 | 🥙 🛞 🚥 🕨 💖 🐚 | s 🚯 \ominus 🔖 🚊 🕘 🔹 |
| Project Navigator 🗗 🗙 | first_fpga.bdf 🗵 🛛 🕸 simple_counter.v 🗵 | 😔 Compilation Report 🗵 | 🕸 my_first_fpga.sdc 🖂 🔳 |
| Entity Cyclone IV E: EP4CE22F17C6 Image: Strateging of the | | Flow Status In Quartus II Version If Revision Name m Top-level Entity Name m Family Cy | progress - Fri Jan 14 17:42:11 2011).1 Build 153 11/29/2010 SJ Full Versior y_first_fpga yclone IV E |
| | | < | |
| Type Message Type Message i Info: 5 registers lost all thei i Info: Generating hard block par i Info: Implemented 38 device res i Info: Quartus II Analysis & Syn System // Processing (27) // Extra Info // Info (27) / | tition "hard block:auto generate ources after synthesis - the fin thesis was successful. 0 errors, | d_inst" al resource count : 0 warnings | |
| System (Processing (27) Extra Info / Info (27) / Message: 0 of 107 | | | Locate |
| | | | 66, 231 24% 🧞 00:00:13 |

Figure 6-44 Compilation Message for project

When compilation is complete, the Quartus II software displays a message. Click OK to close the message box.

The Quartus II Messages window displays many messages during compilation. It should not display any critical warnings; it may display a few warnings that indicate that the device timing information is preliminary or that some parameters on the I/O pins used for the LEDs were not set. The software provides the compilation results in the Compilation Report tab as shown in **Figure 6-45**.



| Flow StatusSuccessful - Fri Jan 14 17:42:39 2011Quartus II Version10.1 Build 153 11/29/2010 SJ Full VersionRevision Namemy_first_fpgaTop-level Entity Namemy_first_fpgaFamilyCyclone IV EDeviceEP4CE22F17C6Timing ModelsFinalTotal logic elements31 / 22,320 (< 1 %)Dedicated logic registers27 / 22,320 (< 1 %)Total registers27Total registers27Total virtual pins0Total memory bits0 / 608,256 (0 %)Embedded Multiplier 9-bit elements1 / 4 (25 %) |
|---|

Figure 6-45 Compilation Report Example

6.9 Program the FPGA Device

After compiling and verifying your design you are ready to program the FPGA on the development board. You download the SOF you just created into the FPGA using the USB-Blaster circuitry on the board. Set up your hardware for programming using the following steps:

First, connect the USB cable, which was included in your development kit, between the DEO-Nano and the host computer. Refer to the getting started user guide for detailed instructions on how to connect the cables.

Refer to the getting started user guide for detailed instructions on how to connect the cables.

Program the FPGA using the following steps.

1. Select Tools > Programmer. The Programmer window opens, as shown in Figure 6-46.



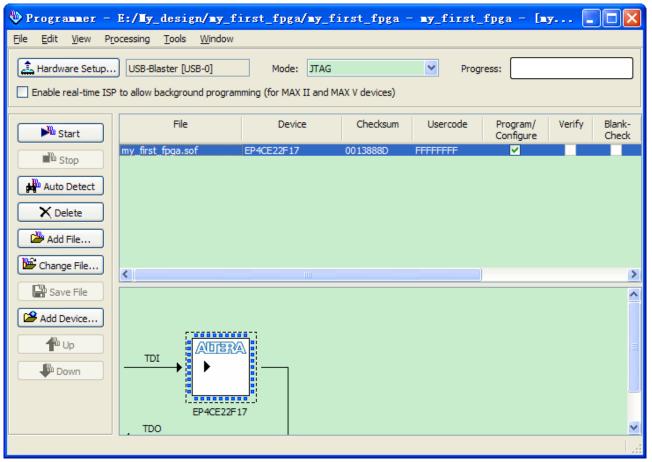


Figure 6-46 Programmer Window

2. Click Hardware Setup.

3. If it is not already turned on, turn on the USB-Blaster [USB-0] option under currently selected hardware, as shown in **Figure 6-47**.



| Ð | Hard | vare Setu | р | | | | | | × |
|---|----------|---|---|---|--------|------------------------------|--------------|----------------|------|
| | Select a | are Settings a programming re setup appli | | setup to us | | ogramming devi er window. | ices. This p | programming | |
| | | ly selected ha | | USB-Blaster No Hardwa USB-Blaster | re | | | | |
| | Hard | lware | | | Server | Port | A | dd Hardware | |
| | USB- | Blaster | | | Local | USB-0 | Re | emove Hardware | |
| | | | | | | | | | |
| | | | | | | | | Close | |

Figure 6-47 Hardware Setting

- 4. Click Close.
- 5. If the file name in the Programmer does not show my_first_fpga.sof, click Add File.
- 6. Select the my_first_fpga.sof file from the project directory (see Figure 6-48).
- 7. Click the **Start** button.



| 🔖 Programmer - I |):/Home/User/Desktop/ | allen/my_first_fpg | a/my_first_fp | ga - my_first_ | fpga - [my_ | | X |
|--------------------|------------------------------|------------------------|----------------|----------------|-----------------------|--------------|-------------|
| File Edit View F | Processing Tools Windov | Ŷ | | | | | |
| 🔔 Hardware Setup | USB-Blaster [USB-0] | Mode: JTAG | | Y Progre | ss: 100% (| (Successful) | |
| Enable real-time I | 5P to allow background progr | amming (for MAX II and | MAX V devices) | | | | |
| Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blar Che |
| 🖬 Stop | my_first_fpga.sof | EP4CE22F17 | 00137CDB | FFFFFFF | V | | |
| Auto Detect | | | | | | | |
| X Delete | | | | | | | |
| Add File | | | | | | | |
| 👺 Change File | < | | | | | | 5 |
| Save File | | | | | | | |
| Add Device | | | | | | | |
| The Up | | | | | | | _ |
| Down | | | | | | | |
| | | | | | | | ~ |
| | | | | | | | 1.8 |

Figure 6-48 Downloading Complete

Congratulations, you have created, compiled, and programmed your first FPGA design! The compiled SRAM Object File (.sof) is loaded onto the FPGA on the development board and the design should be running.

6.10 Verify The Hardware

When you verify the design in hardware, you observe the runtime behavior of the FPGA hardware design and ensure that it is functioning appropriately.

Verify the design by performing the following steps:

1. Observe that the four development board LEDs appear to be advancing slowly in a binary count pattern, which is driven by the simple_counter bits [26..23].

The LEDs are active low, therefore, when counting begins all LEDs are turned on (the 0000 state).

2. Press and hold KEY [0] on the development board and observe that the LEDs advance more quickly. Pressing this KEY causes the design to multiplex using the faster advancing part of the counter (bits [24..21]).

3. If other LEDs emit faintness light, select Assignments > Device. Click Device and Options. See Figure 6-49.



| Device | | | | | | X |
|-----------------------------|-----------------------|--------------|-----------------|-----------------------|------------------------------------|----------|
| Coloct the family a | nd device you want t | a target for | compilation | | | |
| Select the family a | id device you want i | o target for | compliauon. | | | _ |
| Device family — | | | | Show in 'Availab | ble devices' list | |
| Eamily: Cyclone | IVE | | ~ | Package: | Any 😪 | 1 |
| Devices: All | | | | Pin count: | Any 🗸 | 1 |
| Devices: All | | | | | , | |
| Target device — | | | | Sp <u>e</u> ed grade: | Any 🗡 | |
| Auto device | selected by the Fitte | r | | Show advar | nced devices | |
| | e selected in 'Availa | | lint | HardCopy o | ompatible only | |
| ~ - | le selected in Avalia | Die devices | iist | | | |
| Other: n/a | | | | Device and Pin O | ptions | |
| A <u>v</u> ailable devices: | | | | | | |
| - Name | Core Voltage | LEs | User I/Os | Memory Bits | Embedded multiplier 9-bit elements | ^ |
| EP4CE22E22C9L | 1.0V | 22320 | 80 | 608256 | 132 | |
| EP4CE22E22I7 | 1.2V | 22320 | 80 | 608256 | 132 | |
| EP4CE22E22I8L | 1.0V | 22320 | 80 | 608256 | 132 | |
| EP4CE22F17A7 | 1.2V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17C6 | 1.2V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17C7 | 1.2V | 22320 | 154 | 608256 | 132 | -1 |
| EP4CE22F17C8 | 1.2V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17C8L | 1.0V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17C9L | 1.0V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17I7 | 1.2V | 22320 | 154 | 608256 | 132 | |
| EP4CE22F17I8L | 1.0V | 22320 | 154 | 608256 | 132 | |
| EP4CE221114T7 | 1 2V | 22320 | 154 | 608256 | 132 | <u> </u> |
| -Migration compat | | npanion dev | | | | |
| Migration De | vices Har | dCopy: | | | × | |
| 0 migration devic | es selected | Limit DSP & | RAM to HardCopy | device resources | | |
| | | | | | | |
| | | | | | OK Cancel Help | |
| | | | | | | |

Figure 6-49 Device and Options

Select unused pins. Reserve all unused pins: select the As input tri-stated option. See Figure 6-50.



| 🐇 Device and Pin Options - my | _first_fpga 🛛 🗙 |
|-------------------------------|--|
| Category: | |
| | Unused Pins Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor. Reserve all unused pins: As input tri-stated V Specify device-wide options for reserving all unused pins on the device. To reserve other pins individually, use the Assignment Editor. Reserve all unused pins: As input tri-stated V Specify device-wide pins: As input tri-stated V Description: Reserves all unused pins on the target device in one of 5 states: as inputs that are tristated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up. |
| | Input difference Reset OK Cancel |

Figure 6-50 Setting unused pins

Click twice OK.

4. In the Processing menu, choose **Start Compilation**. After the compile, select **Tools** > **Programmer**. Select the **my_first_fpga.sof** file from the project directory. Click **Start**. At this time you could find the other LEDs are off.



Chapter 7

Tutorial: Creating a Nios II Project

This tutorial provides comprehensive information that will help you understand how to create a microprocessor system on your FPGA development board and run software on it. This system will be based on the Altera Nios II processor.

7.1 Required Features

This tutorial requires the Quartus II and Nios II EDS software to be installed. The tutorial was written for version 10.1 of those software packages. If you are using a different version, there may be some difference in the flow. Also, this tutorial requires the DEO-Nano board.

7.2 Creation of Hardware Design

This section describes the flow of how to create a hardware system including a Nios II processor.

1. Launch Quartus II then select **File > New Project Wizard**, start to create a new project. See **Figure 7-1** and **Figure 7-2**.



| 🕊 Qu | artus | II | | | |
|------------|-------------------|----------------|-----------------|---------------------|--------------|
| File | <u>E</u> dit | <u>V</u> iew | <u>P</u> roject | <u>A</u> ssignments | P <u>r</u> o |
| <u>א</u> ב | ≥w | | | Ctrl+N | |
| ൙ O1 | en | | | Ctrl+0 | |
| <u>C</u>] | .ose | | | Ctrl+F4 | |
| 🏠 Ne | ew Proje | ct <u>W</u> iz | ard | | |
| 🛃 01 | oen P <u>r</u> oj | ect | | Ctrl+J | |
| Se | ave Proj | ec <u>t</u> | | | |
| C1 | .os <u>e</u> Pro | ject | | | |
| | ave | | | Ctrl+S | |
| | ave <u>A</u> s | | | | |
| 🗊 Sa | ave All | | | Ctrl+Shift | +S |
| <u>F</u> i | le Prop | erties | | | |
| Cr | reate <u>/</u> | Update | | | • |
| Ep | port <u>.</u> | | | | |
| Co | onvert P | rogram | ming Files | i | |
| P: | age Set <u>u</u> | цр | | | |
| Q Pr | int Pre | <u>v</u> iew | | | |
| 🖨 Pr | int | | | Ctrl+P | |
| Re | ecent F <u>i</u> | les | | | • |
| Re | ecent Pr | ojects | | | × |
| E2 | <u>c</u> it | | | Alt+F4 | |

Figure 7-1 Start to Create a New Project

| Directory, Name, Top-Level Entity [page 1 of 5] Wat is the working directory for this project? D: Home User /Desktop Wat is the name of this groject? Wat is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. Wat is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. Wat is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. Wat is the project Settings | 🐇 New Project Vizard | × |
|---|--|---|
| D: \Home \User \Desktop What is the name of this groject? What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file | Directory, Name, Top-Level Entity [page 1 of 5] | |
| What is the name of this project? What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. | What is the working directory for this project? | |
| What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. | D: \Home \User \Desktop | J |
| What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. | What is the name of this project? | |
| | | J |
| | | 1 |
| Luse Existing Project Settings | |) |
| | Use Existing Project Settings | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | _ |
| < <u>B</u> ack <u>N</u> ext > <u>Einish</u> Cancel <u>H</u> elp | <pre>< <u>Back</u> <u>Next ></u> Einish Cancel <u>Help</u></pre> | |

Figure 7-2 New Project Wizard

2. Select a working directory for this project, type project name and top-level entity name as shown in **Figure 7-3**. Then click **Next**, you will see a window as shown in **Figure 7-4**.



| 🕊 New Project Vizard 🛛 🔀 |
|--|
| Directory, Name, Top-Level Entity [page 1 of 5] |
| What is the working directory for this project? |
| D:/myfirst_niosii |
| What is the name of this project? |
| myfirst_niosii |
| What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file. |
| myfirst_niosii |
| Use Existing Project Settings |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| < Back Next > Einish Cancel Help |

Figure 7-3 Input the working directory, the name of project, top-level design entity

| | sign files you v n always add d | | | oroject. Click Add ect later. | d All to a | dd all design f | files in t | he project dir | ectory to | the proje | ct. | |
|---------------|------------------------------------|-----------|----------------------|----------------------------------|------------|-----------------|------------|----------------|-----------|-----------|-----|-------------------|
| ile name: | | | | | | | | | | | | Add |
| File Name | Type Lib | rary D | esi <i>g</i> n Entry | /Synthesis T | ••1 | NDL Versi | on | | | | | Add Alj Remove |
| pecify the pa | ath names of a | any non-o | lefault librarie | es. Uger Librari | es) | | | | | | | |

Figure 7-4 New Project Wizard: Add Files [page 2 of 5]

3. Click **Next** to skip in **Add Files** window. In the **Family & Device Settings** window, we will choose device family and device settings appropriate for the DEO-Nano board. You should choose settings the same, as shown in **Figure 7-5**. Then click **Next** to get to the window as shown in **Figure 7-6**.



| Image: | Devices: All Pin gount: Any arget device Pin gount: Any Image: Specific device selected by the Fitter Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Other: n/a Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: HardCopy compatible only Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: HardCopy compatible only Image: Advice selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Advice selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Advice selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Specific device selected in 'Available devices' list Image: Advice selected in 'Available devices' list Image: Specific device selected selected in 'Available devices' list Image: Specific devices' list | Devices: All Image: device Pin gount: Any Garget device Speed grade: Any Image: Speed grade: Any O Auto device selected by the Fitter Speed grade: Any Image: Speed grade: Image: Speed grade: Any O Auto device selected in 'Available devices' list Image: Speed grade: Image: Speed grade: Any Image: Speed grade: Any Outer: n/a Image: Speed grade: Image: Speed grade: Image: Speed grade: Any Image: Speed grade: Image: Speed grade: Any Image: Speed grade: Any Image: Speed grade: Image: Speed grade: Any Image: Speed grade: <t< th=""><th>Devices: All Farget device Pin gount: Auto device selected by the Fitter Specific device selected in 'Available devices' list ③ Specific device selected in 'Available devices' list Image: Show advanced devices ④ Qther: n/a HardCopy compatible only Ø alable devices: HardCopy compatible only P4CE22F1761 1. 0V 22320 80 608256 132 P4CE22F1767 1. 2V 22320 154 608256 132 P4CE22F1768 1. 2V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256</th></t<> | Devices: All Farget device Pin gount: Auto device selected by the Fitter Specific device selected in 'Available devices' list ③ Specific device selected in 'Available devices' list Image: Show advanced devices ④ Qther: n/a HardCopy compatible only Ø alable devices: HardCopy compatible only P4CE22F1761 1. 0V 22320 80 608256 132 P4CE22F1767 1. 2V 22320 154 608256 132 P4CE22F1768 1. 2V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256 132 P4CE22F1769 1. 0V 22320 154 608256 |
|---|---|--|--|
| Image: Selected by the Fitter vice selected in 'Available devices' list Image: Selected devices' list Image: Selected d | Devices: All Y 'arget device Pin gount: Any 'arget device Speed grade: Any 'Auto device selected by the Fitter Speed grade: Any 'Speed grade: Any Speed grade: Any 'Other: n/a Show advanced devices 'allable devices: HardCopy compatible only 'Atte222E218L 1.0V 22320 80 608256 132 'Atte222E21705 1.2V 22320 154 608256 132 'Atte222F1705 1.2V 22320 154 608256 132 'Atte222F1705 1.2V 22320 154 608256 132 'Atte22F1705 1.2V 22320 154 608256 132 'Atte22F1705 1.2V 22320 154 608256 132 'Atte22F1705 1.0V 22320 154 608256 132 'Atte22F1705 1.0V 22320 154 608256 132 | Devices: All Image: device Pin gount: Any Image: device Auto device selected by the Fitter Speed grade: Any Image: devices Image: device selected in 'Available devices' list Velop 2221176 | Devices: All Image: Construction of the selected by the Fitter • Auto device selected in 'Available devices' list • Specific device selected in 'Available devices' list • Other: n/a valiable devices: Image: Specific device selected in 'Available devices' list • Show advanced devices Hange Core Yoltage LKs User I/Os Henory Bits HangdCopy compatible only valiable devices: Mane Core Yoltage LKs User I/Os Henory Bits HangdCopy compatible only Pabedded multiplier 9-bit elements P4CE22E218L 1.0V 22320 154 608256 132 132 P4CE22P1705 1.2V 22320 154 608256 132 132 P4CE22P1705 1.2V 22320 154 608256 132 132 P4CE22P1708 1.0V 22320 154 608256 132 132 P4CE22P1708 1.0V 22320 154 608256 132 132 P4CE22P1708 1.0V 22320 154 608256 132 P4CE22P1708 1.0V 22320 154 608256 132 P4CE22P1708 1.0V 22320 154 608256 132 |
| e selected by the Fitter vice selected in 'Available devices' list | arget device Specific device selected by the Fitter • Auto device selected in 'Available devices' list • Other: n/a ailable devices: • Auto device selected in 'Available devices' list • Other: n/a ailable devices: • Manage • Core Voltage LEs Vser I/Os • Benory Bits Enbedded multiplier 9-bit elements * 4CE22E1767 1.2V 22320 154 608256 132 * 4CE22F17C8 1.0V 22320 154 608256 132 * 4CE22F17C9 1.0V 22320 154 608256 132 | Farget device Auto device selected by the Fitter Speed grade: Any • Auto device selected in 'Available devices' list • Other: n/a • Other: Other: Other: Other: Other: Other: Other: Other: | Target device Speed grade: Any • Auto device selected by the Fitter • Specific device selected in 'Available devices' list • Show advanced devices • Other: n/a • Qther: n/a • HardCopy compatible only • Yame • Core Voltage LEs Vser I/Os Embedded multiplier 9-bit elements • P4CE22E18L 1.0V 22320 154 608256 132 • P4CE22F17C8 1.2V 22320 154 608256 132 • P4CE22F17C8 1.2V 22320 154 608256 132 • P4CE22F17C8 1.2V 22320 154 608256 132 • P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V |
| e selected by the Fitter vice selected in 'Available devices' list Core Yoltage LES User I/Os Benory Bits Enbedded multiplier 9-bit elements 1.0V 22320 80 608256 132 1.2V 22320 154 608256 132 | Aget device Auto device selected by the Fitter Specific device selected in 'Available devices' list Qther: n/a ailable devices: Hane Core Voltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements *4CE22E1218L 1. 0V 22320 80 608256 132 *4CE22F17A7 1. 2V 22320 154 608256 132 *4CE22F17C6 1. 2V 22320 154 608256 132 *4CE22F17C8 1. 0V 22320 154 608256 132 *4CE22F17C9 1. 0V 22320 154 608256 132 | Aget device Auto device selected by the Fitter | Auto device selected by the Fitter Specific device selected in 'Available devices' list Other: n/a Auto device selected in 'Available devices' list Other: n/a Auto device selected in 'Available devices' list Other: n/a Auto devices: Auto devices: Auto devices: Auto devices: |
| Core Yoltage LEs User I/Os ■enory Bits Embedded multiplier 9-bit elements 1.0V 22320 80 608256 132 1.2V 22320 154 608256 132 | Specific device selected in 'Available devices' list HardCopy compatible only Other: n/a HardCopy compatible only ailable devices: HardCopy compatible only Y4CE22E2I8L 1.0V 22320 80 608256 132 Y4CE22F1766 1.2V 22320 154 608256 132 Y4CE22F17C6 1.2V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 | Specific device selected in 'Available devices' list HardCopy compatible only Qther: n/a Image: Core Voltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements Image: Core Voltage | • Specific device selected in 'Available devices' list • Qther: n/a valiable devices: Hage • Core Voltage LKs Vser I/Os Henory Bits Embedded multiplier 9-bit elements P4CE22E1767 1. 2V 22320 154 608256 132 P4CE22F17C7 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C9L 1. 0V 22320 154 608256 132 |
| Core Yoltage LEs User I/Os ■enory Bits Embedded multiplier 9-bit elements 1.0V 22320 80 608256 132 1.2V 22320 154 608256 132 | Specific device selected in 'Available devices' list HardCopy compatible only Other: n/a HardCopy compatible only ailable devices: HardCopy compatible only Y4CE22E2I8L 1.0V 22320 80 608256 132 Y4CE22F1766 1.2V 22320 154 608256 132 Y4CE22F17C6 1.2V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 Y4CE22F17C8 1.0V 22320 154 608256 132 | Specific device selected in 'Available devices' list HardCopy compatible only Qther: n/a Image: Core Voltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements Image: Core Voltage | • Specific device selected in 'Available devices' list • Qther: n/a valiable devices: Hage • Core Voltage LKs Vser I/Os Henory Bits Embedded multiplier 9-bit elements P4CE22E1767 1. 2V 22320 154 608256 132 P4CE22F17C7 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C9L 1. 0V 22320 154 608256 132 |
| Core Voltage LEs User I/Os ■emory Bits Embedded multiplier 9-bit elements 1.07 22320 80 608256 132 1.2V 22320 154 608256 132 | Other: n/a allable devices: Hame Core Yoltage LEs User I/Os Hemory Bits Embedded multiplier 9-bit elements P4CE22E2218L 1.0V 22320 80 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Other: n/a Mame Core Voltage LEs User I/Os Hemory Bits Embedded multiplier 9-bit elements P4CE22E101 1.0V 22320 80 608256 132 P4CE22F17A7 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Other: n/a Yallable devices: Hame Core Yoltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements P4CE22E2218L 1.0V 22320 80 608256 132 P4CE22F17A7 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| Core Voltage LEs User I/Os Benory Bits Embedded multiplier 9-bit elements ▲ 1.0V 22320 80 608256 132 1.2V 22320 154 608256 132 | allable devices: Hane Core Voltage LEs User I/Os Bemory Bits Embedded multiplier 9-bit elements P4CE22E22I8L 1.0V 22320 60 606256 132 P4CE22F17A7 1.2V 22320 154 606256 132 P4CE22F17C6 1.2V 22320 154 606256 132 P4CE22F17C8 1.2V 22320 154 606256 132 P4CE22F17C8 1.2V 22320 154 606256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Hane Core Voltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements P4CE22E128L 1.0V 22320 80 608256 132 P4CE22E171A7 1.2V 22320 154 608256 132 P4CE22E17C6 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Mane Core Voltage LEs User I/Os Menory Bits Embedded multiplier 9-bit elements P4CE22E22I3L 1. 0V 22320 80 608256 132 P4CE22F17A7 1. 2V 22320 154 608256 132 P4CE22F17C6 1. 2V 22320 154 608256 132 P4CE22F17C7 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C9L 1. 0V 22320 154 608256 132 |
| Core Voltage LEs User I/Os ■emory Bits Embedded multiplier 9-bit elements ▲ 1.0V 22320 80 608256 132 132 1.2V 22320 154 608256 132 132 | allable devices: Hane Core Voltage LEs User I/Os Bemory Bits Embedded multiplier 9-bit elements P4CE22E22I8L 1.0V 22320 60 606256 132 P4CE22F17A7 1.2V 22320 154 606256 132 P4CE22F17C6 1.2V 22320 154 606256 132 P4CE22F17C8 1.2V 22320 154 606256 132 P4CE22F17C8 1.2V 22320 154 606256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Hane Core Voltage LEs User I/Os Henory Bits Embedded multiplier 9-bit elements P4CE22E128L 1.0V 22320 80 608256 132 P4CE22E171A7 1.2V 22320 154 608256 132 P4CE22E17C6 1.2V 22320 154 608256 132 P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C8 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | Mane Core Voltage LEs User I/Os Menory Bits Embedded multiplier 9-bit elements P4CE22E22I3L 1. 0V 22320 80 608256 132 P4CE22F17A7 1. 2V 22320 154 608256 132 P4CE22F17C6 1. 2V 22320 154 608256 132 P4CE22F17C7 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 2V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C8 1. 0V 22320 154 608256 132 P4CE22F17C9L 1. 0V 22320 154 608256 132 |
| | V4CE22F17C6 1.2V 22320 154 608256 132 V4CE22F17C7 1.2V 22320 154 608256 132 V4CE22F17C8 1.2V 22320 154 608256 132 V4CE22F17C8 1.2V 22320 154 608256 132 V4CE22F17C8L 1.0V 22320 154 608256 132 V4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C6 1.2V 22320 154 608256 132 P4CE22F17C7 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | F4CE22F17C6 1.2V 22320 154 608256 132 F4CE22F17C7 1.2V 22320 154 608256 132 F4CE22F17C8 1.2V 22320 154 608256 132 F4CE22F17C8 1.2V 22320 154 608256 132 F4CE22F17C8L 1.0V 22320 154 608256 132 F4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C7 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C7 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C7 1.2V 22320 154 608256 132 P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608258 132 |
| 1 0V 00000 1E4 0000E6 100 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | | | |
| . 1.0V 22320 154 608256 132 | | | |
| 1.0V 22320 154 608256 132 1.0V 22320 154 608256 132 | | | |
| 1.0V 22320 154 608256 132 1.0V 22320 154 608256 132 1.2V 22320 154 608256 132 | | | |
| 1.0V 22320 154 608256 132 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 . 1.0V 22320 154 608256 132 1.2V 22320 154 608256 132 | | | Companion device |
| . 1.0V 22320 154 608256 132 . 1.0V 22320 154 608256 132 1.2V 22320 154 608256 132 | Companion device | Companion device | Companion device |
| 1.2V 22320 154 608256 132 | | | |
| 1 9V 20200 1E4 6002E6 120 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.27 22320 134 00230 132 | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| 1. 2Y 2232U 134 000230 132 | | | |
| 1.27 22320 134 000230 132 | | | |
| 1.27 22320 134 000230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| א דער אין ארא א | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 11. ZY 122320 134 100230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 11. ZY 122320 134 100230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.27 22320 134 000230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.27 22320 134 000230 132 | | | |
| | | | |
| | | | |
| . 1.0V 22320 154 608256 132 | | ACTOOR1777 1 OV 00000 1E4 0000E6 100 | NGTOOTISTS 1 OV |
| . 1.0V 22320 154 608256 132 | ALV97V17T7 11 7V 199290 11E4 1209262 1129 | | |
| . 1.0V 22320 154 608256 132 | 4UE2ZF1717 1. ZV 22320 154 608256 132 🛛 😽 | (4622F111) 1.29 22320 154 608256 132 1 | (4UBZZFI(1) 1.2V 2232U 154 608256 132 |
| . 1.0V 22320 154 608256 132 | (4CK22K17T7 L1 2V 2232D 154 608256 132 🗸 | 1967/271111 11/29 1223211 1154 1506255 1632 | 19UNZZMICIC LEZY – 1223211 1154 – 16UNZ56 – 1132 |
| . 1.0V 22320 154 608256 132 | | ACRODRITT 1 OV 00000 1E4 0000E2 100 | |
| | | | |
| | | | |
| 1.27 22320 134 000230 132 | | | |
| 1. 27 22320 139 100230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | 4CE22F17C9L 1.0V 22320 154 608256 132 | 4CE22F17C9L 1.0V 22320 154 608256 132 | 4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1 97 199990 164 609966 199 | PACE22F17C9L 1.0V 22320 154 608256 132 | 24CE22F17C9L 1.0V 22320 154 608256 132 | 24CE22F17C9L 1.0V 22320 154 608256 132 |
| 1 97 99220 154 809256 122 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | R4CE22F17C8L 1.0V 22320 154 608256 132 R4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0Y 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0Y 22320 154 608256 132 |
| 1.2V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0Y 22320 154 608256 132 |
| 1 2V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1 2V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1 2V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8 1.2V 22320 154 608256 132 P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.2V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.2V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0Y 22320 154 608256 132 |
| 1 OV 92220 1E4 8000E6 122 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C8L 1.0V 22320 154 608256 132 P4CE22F17C9L 1.0Y 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| 1.27 22320 134 000230 132 | PACE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 | P4CE22F17C9L 1.0V 22320 154 608256 132 |
| | | | |
| | | | |
| | | | |
| | | | |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | | ACRODITIZ 1 OV 00000 1E4 0000E6 100 | AGROOPIGTA 1 OV |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | | ACRODRITE 1 OV 00000 1E4 0000E2 100 | |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | | | |
| . 1.0V 22320 154 608256 132 | 40°K22K17T7 11-2V 122320 154 1608256 1132 👽 | | SALKZZMITETETETETETETETETETETETETETETETETETETE |
| . 1.0V 22320 154 608256 132 | 4UE2ZF1717 1. ZV 22320 154 608256 132 🛛 😽 | 9012211111 1.29 22320 154 606256 132 | 9UEZZFITIT 1.2V 2Z32U 154 608256 132 |
| 1.0V 22320 154 608256 132 1.0V 22320 154 608258 132 | | | International In |
| 1.0V 22320 154 608256 132 1.0V 22320 154 608256 132 | | | Contraction (Lease) (Lease) (Lease) |

Figure 7-5 New Project Wizard: Family & Device Settings [page 3 of 5]

4. Click **Next** and will see a window as shown in **Figure 7-7**. **Figure 7-7** is a summary about the new project. Click **Finish** to complete the New Project Wizard. **Figure 7-8** show the new project.



| New Project Wis | sard | | |
|-------------------|--------------------------------|----------------------------|--|
| EDA Tool Setti | ings [page 4 of 5] | | |
| | ols used with the Quartus II s | | niect |
| | | ontware to develop your pr | ojecu. |
| DA tools: | | | |
| Tool Type | Tool Name | Format (s) | Run Tool Automatically |
| Design Entry/Syn… | <none> 💌</none> | <none></none> | Run this tool automatically to synthesize the current design |
| Simulation | <none></none> | <none></none> | Run gate-level simulation automatically after compilation |
| Timing Analysis | <none></none> | <none></none> | Run this tool automatically after compilation |
| Formal Verificat… | <none></none> | | |
| Board-Level | Timing | <none></none> | |
| | Symbol | <none></none> | |
| | Signal Integrity | <none></none> | |
| | Boundary Scan | <none> 💙</none> | |
| | | | |
| | | | |
| | | | |

Figure 7-6 New Project Wizard: EDA Tool Settings [page 4 of 5]

| 🖑 New Project Vizard | |
|---|---|
| Summary [page 5 of 5] | |
| When you click Finish, the project will be created with the following settings: | |
| Project directory: Project name: Top-level design entity: Number of files added: | D:/myfirst_niosii myfirst_niosii myfirst_niosii 0 |
| Number of thes added: Number of user libraries added: Device assignments: | 0 |
| Family name: | Cyclone IV E |
| Device: | EP4CE22F17C6 |
| EDA tools: | |
| Design entry/synthesis: | <none> (<none>)</none></none> |
| Simulation: | <none> (<none>)</none></none> |
| Timing analysis: | <none> (<none>)</none></none> |
| Operating conditions: | |
| VCCINT voltage: | 1.2V |
| Junction temperature range: | 0-85 °C |
| | |
| | |
| | |
| | |
| | |
| | |
| | < <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel <u>H</u> elp |

Figure 7-7 New Project Wizard: Summary [page 5 of 5]



| 🦞 Quartus II - D:/myfirst_niosii/myfirst_niosii - myfirst_niosii | |
|---|----------------------------------|
| <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>P</u> roject <u>A</u> ssignments Processing <u>T</u> ools <u>W</u> indow <u>H</u> elp | |
| | |
| 🤋 myfirst_niosii 💽 🔀 🌠 🖉 🦉 🥙 💷 🕨 🦻 🕲 😓 👗 🕘 🛡 | |
| Project Navigator | |
| | |
| Cyclone IV E: EP4CE22F17C6 | |
| · → myfirst_niosii 👸 | |
| | |
| QUARTU | |
| | |
| A Hierarchy E Files d ^p Design Units | |
| Status 8 × | |
| | |
| | ew New Quartus II |
| | ew New Quartus II Information |
| | Documentation |
| X Type Message |] |
| | |
| | |
| | |
| | |
| | 2 |
| Wessage: Info Units < | Locate |
| Z Message: Location: | Locate |

Figure 7-8 A New Complete Project

5. Select **Tools** > **SOPC Builder** to open SOPC Builder, the Altera system generation tool, as shown in **Figure 7-9**.

| 5 | <u>T</u> oo | ls <u>Y</u> | <u>"</u> indow | Help | |
|---|-------------|----------------|------------------------|------------------------------------|----|
| | | Run E | DA Sim <u>u</u> | lation Tool | ۱. |
| | | Run E | DA Timi: | ng Analysis Tool | |
| 2 | 'n | Launc | h EDA S | imulation Library <u>C</u> ompiler | |
| | <u>-</u> 8- | Launc | h Desig | n Space E <u>x</u> plorer | |
| | \odot | <u>T</u> imeQ | luest Ti | ming Analyzer | |
| | | <u>A</u> dvis | ors | | • |
| | > | C <u>h</u> ip | Planner | (Floorplan and Chip Editor) | |
| | ۰ | Desig | m Parti | tion Planner | |
| | | Netli | st <u>V</u> iew | ers | ۲. |
| | ا ما | Signs | lTap II | Logic A <u>n</u> alyzer | |
| | m | In-Sy | rstem Mei | mor <u>y</u> Content Editor | |
| | | Logic | : Analyz | er Interface Edito <u>r</u> | |
| | 01 | In <u>-</u> Sy | rstem Sor | urces and Probes Editor | |
| | | Signa | lProbe 1 | Pins | |
| | 1 | Progr | ammer | | |
| | ۲ | <u>J</u> TAG | Chain D | ebugger | |
| | XX | Trans | ceiver ' | Tool <u>k</u> it | |
| | 1 | Exter | nal <u>M</u> em | ory Interface Toolkit | |
| | 20 | Mega <u>M</u> | izard P | lug-In Manager | |
| | | SOPC | \underline{B} uilder | | |
| | | Qsys | (Beta) | | |
| | <u> </u> | Tel S | Scr <u>i</u> pts. | | |
| | | Custo | mi <u>z</u> e | | |
| | | <u>O</u> ptio | ns | | |
| - | | Licen | ise Setu; | p | |

Figure 7-9 SOPC Builder Menu



| 4 Create New System - Alt | era SOPC Builder - unnam | ed.sopc (D:\myfirst | _niosii\unnamed.s | opc) | |
|--|-----------------------------|---------------------|-------------------|------|-----|
| <u>File Edit M</u> odule <u>System View T</u> o | ools <u>H</u> elp | | | | |
| System Contents System Generation | | | | | |
| Component Library | Target | Clock Settings | | | |
| Project | Device Family: Cyclone IV E | Name | Source | MHz | Add |
| Library | | | | | |
| Interface Protocols | Use C Mc Create New | 7 System | Clock | Rase | End |
| Legacy Components Memories and Memory Contro | | | Clock | Base | End |
| Merlin Components Peripherals | System Name: unna | ameo | | | |
| | Target HDL: 💿 Ve | rilog | | | |
| Processor Additions Processors | O VH | IDL | | | |
| | Info: Specify a | new system name. | | | |
| ⊡ University Program ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | | • | | | |
| < | | OK Cancel | | | |
| Sector 2 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ | < | | | | > |
| New Edit Add | Remove Edit | Addre | ess Map | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | E <u>x</u> it Help | Prev Next | Generate | | |

Figure 7-10 Create New SOPC System [0]

6. Rename System Name as shown in **Figure 7-10** and **Figure 7-11**. Click **OK** and your will see a window as shown in **Figure 7-12**.

| 🖳 Create New System 🛛 🔀 | | | | | |
|----------------------------|--|--|--|--|--|
| System Name: DE0_NANO_SOPC | | | | | |
| Target HDL: 💿 Verilog | | | | | |
| | | | | | |
| | | | | | |
| OK Cancel | | | | | |

Figure 7-11 Create New System [1]



| 😃 Altera SOPC Builder | | | | | |
|---|-----------------------------|----------------|----------|-----------------|--------|
| <u>File Edit M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> o | ools <u>H</u> elp | | | | |
| System Contents System Generation | | | | | |
| Component Library | Target | Clock Settings | | | |
| Project | Device Family: Cyclone IV E | Name | Source | MHz | Add |
| Library | | clk_0 | External | 50.0 | Remove |
| ⊞ ••Bridges and Adapters ∎ ••Interface Protocols | | | | | |
| | Use C Module | Description | Clock | Base | End |
| Memories and Memory Contro Merlin Components | | | | | |
| Peripherals | | | | | |
| ⊕PLL ⊕Processor Additions | | | | | |
| | | | | | |
| SLS University Program | | | | | |
| 🗄 Video and Image Processing 🞽 | | | | | |
| | | | | | |
| S | < | | | | > |
| New Edit Add | Remove Edit | Addr | ess Map | Filter: Default | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | E <u>x</u> it Help | Prev Next | Generate | | |

Figure 7-12 Create New System[2]

7. Click the **clk_0** name in the Clock Settings table to rename **clk_0** to **clk_50**. Press **Enter** to complete the update, as shown in **Figure 7-13**.

| 😃 Altera SOPC Builder | | | | | | | | |
|---|-----------------------------------|----------------|----------|------|--------|--|--|--|
| <u>File E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> | <u>T</u> ools <u>H</u> elp | | | | | | | |
| System Contents System Generation | System Contents System Generation | | | | | | | |
| Component Library | Target | Clock Settings | | | | | | |
| Project 🔥 | Device Family: Cyclone IV E 🗸 | Name | Source | MHz | Add | | | |
| New component | | clk_50 | External | 50.0 | | | | |
| | | | | | Remove | | | |
| | | | | | | | | |
| Interface Protocols | | | | | | | | |
| Egacy Components | Use C Module | Description | Clock | Base | End | | | |
| Memories and Memory Contro | | | | | | | | |
| Herlin Components | | | | | | | | |
| | | | | | | | | |
| €…PLL | | | | | | | | |

Figure 7-13 Rename Clock Name

8. In the left hand-side Component Library tree, select **Library > Processors > Nios II Processor** and click the **Add...** button to open the Nios II component wizard, as shown in **Figure 7-14** and **Figure 7-15**.

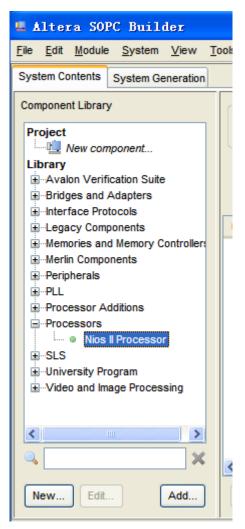


Figure 7-14 Add NIOS II Processor



| Nios II Proces | sor - cpu_0 | | | X |
|--|--------------------------------|--|---|---------------------------------------|
| Nios | II Processor | | | About Documentation |
| Parameter Settings | | | | |
| | es and Memory Interfaces | Advanced Features M | IMU and MPU Settings > JTAG D | Debug Module 🔰 Custom Instructions 🖒 |
| Core Nios II | | | | |
| Select a Nios II core: | | | | |
| | ○Nios II/e | ○Nios II/s | ●Nios II/f | <u>_</u> |
| Nios II Selector Guide Family: Cyclone IV E f _{system:} 50.0 MHz cpuid: 0 | RISC 32-bit | RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide | RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction | ≣ |
| Performance at 50.0 MHz | | Up to 32 DMIPS | Up to 57 DMIPS | — — — — — — — — — — — — — — — — — — — |
| Logic Usage | 600-700 LEs | 1200-1400 LEs | 1400-1800 LEs | _ |
| Hardware Multiply: Embe | dded Multipliers | Y Hardware Divide | | |
| Reset Vector: Mem | ory: | ✓ Offset: 0x0 | | |
| Exception Vector: Memo | ry: | V Offset: 0x20 | | |
| Include MMU Only include the MMU who | en using an operating system t | hat explicitly supports an MMU | | |
| Fast TLB Miss Exception | Vector: Memory: | ~ C | Offset: OxO | |
| Include MPU | | | | |
| 🕂 Warning: Reset vector | and Exception vector cannot b | e set until memory devices are | e connected to the Nios II processor | |
| | | | | Cancel < Back Mext > Finish |

Figure 7-15 Nios II Processor

9. Click **Finish** to return to main window as shown in **Figure 7-16**.



| u Altera SOPC Builder | |
|---|------------|
| <u>File Edit M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> ools Nios II <u>H</u> elp | |
| System Contents System Generation | |
| Component Library Clock Settings | |
| Project Device Family: Cyclone IV E 🗸 Name Source MHz | Add |
| New component clk_50 External 50.0 | Remove |
| Library | |
| Bridges and Adapters | |
| Interface Protocols Use Conn Module Description Clock Base | End |
| Memories and Memory Controllers I | |
| Merlin Components instruction_master Avalon Memory Mapped Master CIk_50 data master Avalon Memory Mapped Master [K] | |
| data_master Avaion Memory Mapped Master [Cik] IRQ 0 | 0v0000fff |
| Processor Additions | 0400000111 |
| | |
| ⊕-SLS | |
| ⊡ University Program | |
| | |
| | |
| | |
| | > |
| New Edit Add Filter: Default | |
| | |
| To Do: cpu_0: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue | |
| To Do: cpu_0: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue | |
| Kurning: cpu_0: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor | |
| | |
| | |
| Exit Help I Prev Next Generate | |

Figure 7-16 Add Nios II CPU completely

10. Select the **cpu_0** component and right-click then select rename, after this, you can update **cpu_0** to **cpu**, as shown in **Figure 7-17** and **Figure 7-18**.



| 🗳 Altera SOPC Builder | | | | | |
|---|-----------------------------|---|-------------------------------|-------------------------------|------------|
| <u>File Edit Module System View Took</u> | s Nios II <u>H</u> elp | | | | |
| System Contents System Generation | | | | | |
| Component Library | Target | Clock Settings | | | |
| Project | Device Family: Cyclone IV E | Name | Source | MHz | Add |
| New component | | clk_50 | External | 50.0 | Remove |
| Library | | | | | |
| Bridges and Adapters | | | | | |
| ⊡…Interface Protocols ⊡…Legacy Components | Use Conn Module | Description | Clock | Base | End |
| Memories and Memory Controllers | ✓ □ cpu_0 | Nico II Droccor | or [clk] | | |
| Merlin Components | instru C | connections • | Mapped Master clk_50 | | |
| ⊕-Peripherals ⊕-PLL | | rint | Mapped Master [clk] | IRQ O | |
| Processor Additions | jtag_(S | how Connected | Mapped Slave [clk] | 0080000x0 [™] | 0x00000fff |
| -Processors | ✓ D | efault | | | |
| Nios Il Processor | A | All Control of the second s | | | |
| i ⊕SLS i ⊕University Program | с | locks | | | |
| | A | valon-MM | | | |
| The view and mage recovering | А | valon-ST | | | |
| | E | dit Ctrl+E | - | | |
| | | lename Ctrl+R | | | |
| | B | lemove | | | > |
| New Edit Add | Remove Edit D | etails | Address Map <u>F</u> ilters | Filter: Default | |
| | s | how Arbitration | | | |
| To Do: cpu_0: No reset vector has been | - | | this issue olve this issue | | |
| To Do: cpu_0: No exception vector has b Warning: cpu 0: Reset vector and Exception | | ock Base Address Ctrl+L | to the Nios II processor | | |
| , | | xpand All | | | |
| | <u>C</u> | ollapse All | _ | | |
| | | et Color | | | |
| | Exit Help | Prev Next | Generate | | |

Figure 7-17 Rename the CPU (1)

| 🖣 Altera SOPC Builder | | | | | |
|--|-------------------------------|---|---------------------------|------------------|------------|
| File Edit Module System View To | ols Nios II <u>H</u> elp | | | | |
| System Contents System Generation | | | | | |
| Component Library | Target | Clock Settings | | | |
| Project | Device Family: Cyclone IV E | Name | Source | MHz | |
| New component | | clk_50 | External | 50.0 | Add |
| | | | | | Remove |
| ⊕ Avalon Verification Suite ⊕ Bridges and Adapters | | | | | |
| | Use Conn Module | Description | 011- | Dava | - |
| | Use Conn Module | Description Nios II Processor | Clock [clk] | Base | End |
| | instruction_mas | | | | |
| ⊕Peripherals ⊕PLL | data_master jtag_debug_mod | Avalon Memory Ma dule Avalon Memory Ma | | URQ 0x0000800 | - |
| Processor Additions | Jtag_debug_mod | dule Avaion Memory Ma | apped Slave [[Cik] | 0x0000800 | 0x00000fff |
| Processors Nios II Processor | | | | | |
| ±SLS | | | | | |
| | | | | | |
| The video and image Processing | | | | | |
| < > | | | | | |
| | | | | | |
| | < | III | | | > |
| New Edit Add | Remove | | dress Map <u>F</u> ilters | Filter: Default | |
| | | | | | |
| To Do: cpu: No reset vector has been s | | | | | |
| To Do: cpu: No exception vector has be A Warning: cpu: Reset vector and Exception | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | E <u>x</u> it Help | Prev Next | Generate | | |

Figure 7-18 Rename the CPU (2)



11. Add a second component by selecting Library > Interface Protocols > Serial > JTAG UART and clicking the Add... button, as shown in Figure 7-19 and Figure 7-20.

| 😃 Altera SOPC Builder | |
|--|----|
| <u>File E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew | Ţo |
| System Contents System Generation | n |
| Component Library | |
| Project | |
| Library Avalon Verification Suite | |
| Bridges and Adapters Interface Protocols | |
| Ethernet | |
| ⊞High Speed ⊞Interlaken | |
| ⊞SDI | |
| Serial | |
| Avalon-ST Serial JTAG UART | |
| ···· SPI (3 Wire Serial | |
| Legacy Components | |
| | |

Figure 7-19 Add the JTAG UART component



| U JTAG UART - : | jtag_uart_0 |
|-----------------------------|---|
| Megecore JTAG altera_ava | UART lon_itag_uart |
| 🔻 Block Diagram | |
| res | ck = clk irq = interrup1 set = reset on = avalon_itag_slave |
| 🔻 Write FIFO (Data fro | om Avalon to JTAG) |
| Buffer depth (bytes): | 64 🗸 |
| IRQ threshold: | 8 |
| Construct using r | egisters instead of memory blocks |
| 🝸 Read FIFO (Data fro | m JTAG to Avalon) |
| Buffer depth (bytes): | 64 🗸 |
| IRQ threshold: | 8 |
| Construct using r | egisters instead of memory blocks |
| Simulated input ch | aracter stream |
| Contents: | |
| Prepare interactive | e windows |
| Options: | INTERACTIVE_ASCII_OUTPUT |
| | |
| | |
| | |
| | Cancel Finish |

Figure 7-20 JTAG UART's add wizard

12. We are going to use the default settings for this component, so click **Finish** to close the wizard and return to the window as shown in **Figure 7-21**.



| L Altera SOPC Builder | | | | | |
|--|---|-----------------------------------|------------------------|-----------------|------------|
| <u>File Edit M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> i | ools Nios II <u>H</u> elp | | | | |
| System Contents System Generation | | | | | |
| Component Library | Target | Clock Settings | | | |
| Project | Device Family: Cyclone IV E 🗸 | Name | Source | MHz | Add |
| New component | | clk_50 | External | 50.0 | Remove |
| Library Avalon Verification Suite | | | | | |
| | | | | | |
| ⊡…Interface Protocols ⊕…ASI = | Use Conn Module | Description | Clock | Base | End |
| thernet | | Nios II Processor | [clk] | | |
| ⊞ High Speed | instruction_mast | | | | |
| | data_master | Avalon Memory Mapp | | IRQ | |
| ⊞-PCI ⊞-SDI | jtag_debug_mod | | | = 0x0000800 | 0x00000fff |
| | itag_uart_0 → avalon_itag_slav | JTAG UART e Avalon Memory Mapp | ed Slave [clk] | | 0x00000007 |
| ···· Avalon-ST JTAG | | o product moniory mapp | | 1= 040000000 | 1020000001 |
| Avalon-ST Serial UTAG UART | | | | | |
| JTAG UART SPI (3 Wire Serial | | | | | |
| ■ UART (RS-232 Se | | | | | |
| 🖃 Legacy Components 💌 | | | | | |
| | | | | | |
| Q X | < | | | | > |
| | | | | | |
| New Edit Add | Remove Edit | Addre: | ss Map <u>F</u> ilters | Filter: Default | |
| | | | | | |
| To Do: cpu: No reset vector has been To Do: cpu: No exception vector has I | | | | | |
| Warning: cpu: Reset vector and Exce | | | | | |
| warning. cpu. Reset vector and Exec | pion vector cannot be set and memory de | | | | |
| | | | | | |
| | | | | | |
| | Exit Help | Prev Next | Generate | | |
| | | | | | |

Figure 7-21 JTAG UART

13. Select the jtag_uart_0 component and rename it to jtag_uart as shown in Figure 7-22.



| 🗳 Altera SOPC Builder | | | | |
|---|--|---|---------------------------------------|------------|
| <u>File Edit M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> o | ools Nios II <u>H</u> elp | | | |
| System Contents System Generation | | | | |
| Component Library | Clock | Settings | | |
| Project | Device Family: Cyclone IV E V | ne Source | MHz | Add |
| New component | cik_ | 50 External | 50.0 | Remove |
| Library ⊕ Avalon Verification Suite | | | | Kelliove |
| | | | | |
| Interface Protocols | Use Conn Module | Description | Clock Base | End |
| | | Nios II Processor | [clk] | Ling |
| ⊞High Speed ⊞Interlaken | instruction_master | Avalon Memory Mapped Master | clk_50 | |
| | data_master jtag_debug_module | Avalon Memory Mapped Master Avalon Memory Mapped Slave | [clk] IRQ 0 [clk] Ox0000800 | |
| ⊡SDI | ✓ □ jtag_uccag_module | JTAG UART | [clk] | 0200000111 |
| Serial | avalon_itag_slave | Avalon Memory Mapped Slave | clk_50 © 0x0000000 | 0x0000007 |
| Avalon-ST Serial | | | | |
| JTAG UART | | | | |
| SPI (3 Wire Serial UART (RS-232 S€ | | | | |
| 🗈 Legacy Components 💌 | | | | |
| | | | | |
| QX | < | | | > |
| New Edit Add | Remove Edit | Address Map Fi | Iters Filter: Default | |
| | | | Piller. Delauk | |
| To Do: cpu: No reset vector has been | specified for this CPU. Please parameterize the | CPU to resolve this issue | | |
| 🖓 To Do: cpu: No exception vector has t | been specified for this CPU. Please parameterize | the CPU to resolve this issue | | |
| A Warning: cpu: Reset vector and Exce | ption vector cannot be set until memory devices | are connected to the Nios II processor | | |
| | | | | |
| | | | | |
| | Exit Help I Pr | ev Next D Generate | 1 | |
| | | | J | |

Figure 7-22 Rename JTAG UART

15. Add the Library > Memories and Memory Controllers > On-Chip > On-Chip Memory (RAM or ROM) component to system, as shown in Figure 7-23 and Figure 7-24.



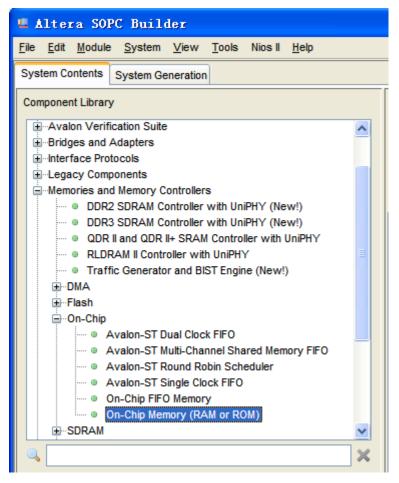


Figure 7-23 Add On-Chip Memory



| 😃 On-Chip Memory (R. | All or ROL) - onchip_m | .emory2_0 🛛 🔀 |
|--|---|-------------------------|
| MogeCore On-Chip Ma altera_avalon_onc | emory (RAM or ROM hip_memory2 |) <u>D</u> ocumentation |
| * Block Diagram | | ^ |
| | clock ➡ clk1 avalon ➡ s1 reset ➡ reset1 | |
| * Memory type | | |
| Туре: | RAM (Writable) | |
| Dual-port access | | |
| Read During Write Mode: | DONT_CARE V | = |
| Block type: | Auto 💙 | = |
| T Size | | |
| Data width: | 32 💌 | |
| Total memory size: | 4096 | bytes |
| Minimize memory block us | age (may impact fmax) | |
| Read latency | | |
| Slave s1 Latency: | 1 🗸 | |
| Slave s2 Latency: | 1 🗸 | |
| Memory initialization | | |
| Initialize memory content | | |
| Enable non default initialis | ration file | <u>∼</u> |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | Cancel Finish |

Figure 7-24 On-Chip Memory Box

16. Modify Total memory size setting to **26000** as shown in **Figure 7-25**. Click **Finish** to return to the window as in **Figure 7-26**.



| 🗳 On-Chip Memory (R | Al or ROL) - onchip_m | emory2_0 | < |
|--|---|-----------------|---|
| Mogeccore On-Chip M altera_avalon_onc | emory (RAM or ROM) |) Documentation | |
| * Block Diagram | | | 4 |
| | clock ➡ clk1 avalon ➡ s1 reset ➡ reset1 | | |
| Memory type | | | |
| Туре: | RAM (Writable) | | |
| Dual-port access | | | |
| Read During Write Mode: | DONT_CARE | | = |
| Block type: | Auto 🗸 | | - |
| ▼ Size | | | |
| Data width: | 32 💌 | | |
| Total memory size: | 26000 | bytes | |
| Minimize memory block us | sage (may impact fmax) | | |
| Read latency | | | |
| Slave s1 Latency: | 1 🗸 | | |
| Slave s2 Latency: | 1 🗸 | | - |
| Memory initialization | | | |
| Initialize memory content | | | |
| Enable non default initiali: | zation file | | ~ |
| | | | ٦ |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Cancel Finish | J |

Figure 7-25 Update Total memory size



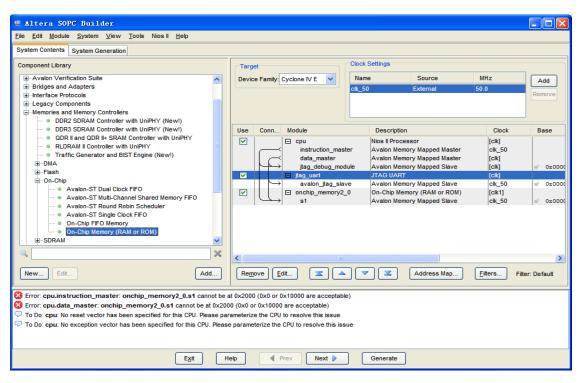


Figure 7-26 Add On-Chip memory

17. Rename onchip_memory2_0 to onchip_memory2 as shown in Figure 7-27.

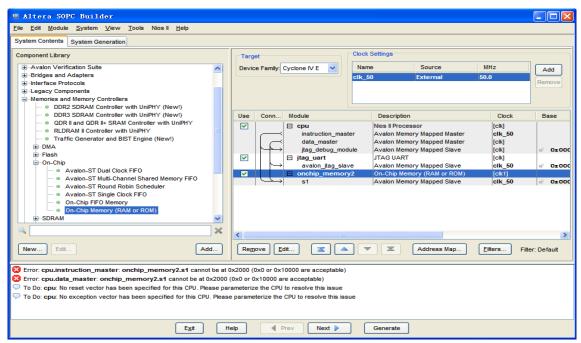


Figure 7-27 Rename On-Chip memory

18. Right click on the **cpu** component table and select **Edit...** from the list. Update the Reset Vector and Exception Vector as shown in **Figure 7-28**. Then, click **Finish** to return to the window as shown **Figure 7-29**.



| L Nios II Proces | sor – cpu | | | | |
|--|--------------------------------|--|---|--------------------------|-----------------------------|
| Mios | II Processor | | | | About Documentation |
| Parameter Settings | | | | | |
| | s and Memory Interfaces | Advanced Features > M | IMU and MPU Settings | JTAG Debug Mod | ule 🔪 Custom Instructions 🔪 |
| Core Nios II | | | | | |
| Select a Nios II core: | | | | | _ |
| | ○Nios II/e | ○Nios II/s | ●Nios II/f | | <u></u> |
| Nios II Selector Guide Family: Cyclone IV E f _{system:} 50.0 MHz cpuid: 0 | RISC 32-bit | RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide | RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Pr | ediction | E |
| Performance at 50.0 MHz | | Up to 32 DMIPS | Up to 57 DMIPS | | |
| Logic Usage | 600-700 LEs | 1200-1400 LEs | 1400-1800 LEs | | ✓ |
| Hardware Multiply: Embe | dded Multipliers | Hardware Divide | | | |
| Reset Vector: Mem Exception Vector: Memo | | ✓ Offset: 0x0 ✓ Offset: 0x20 | | 0x00002000 0x00002020 | |
| Include MMU | | | | | |
| | en using an operating system t | | | | |
| Fast TLB Miss Exception | Vector: Memory: | <u> </u> | Offset: 0x0 | | |
| | | | | | |
| | | | | Cancel | 1 < Back Mext > Finish |

Figure 7-28 Update CPU settings



| # Altera SOPC Builder | | | |
|---|---------------------------------------|---|------------------------------|
| <u>File Edit Module System View T</u> ools Nios II <u>H</u> elp | | | |
| | | | |
| System Contents System Generation | | | |
| Component Library | Target | Clock Settings | |
| Avalon Verification Suite | Device Family: Cyclone IV E | Name Source | MHz Add |
| Bridges and Adapters In-Interface Protocols | | clk_50 External | 50.0 Remove |
| | | | i cinove |
| Memories and Memory Controllers | | | |
| DDR2 SDRAM Controller with UniPHY (New!) | | | |
| DDR3 SDRAM Controller with UniPHY (New!) | Use Conn Module | Description | Clock Base |
| O QDR II and QDR II+ SRAM Controller with UniPHY RLDRAM II Controller with UniPHY | 🔽 🖂 cpu | Nios II Processor | [clk] |
| Traffic Generator and BIST Engine (New!) | instruction_mast | | clk_50 |
| E⊡DMA | data_master itag_debug_mod | Avaion Memory Mapped Master Avaion Memory Mapped Slave | [clk] [clk] 0x00 0 |
| iter Flash | ✓ □ jtag_uart | JTAG UART | [clk] |
| ⊡On-Chip | avalon_itag_slav | Avaion Memory Mapped Slave | clk_50 ■ 0x000 |
| Avalon-ST Dual Clock FIFO Avalon-ST Multi-Channel Shared Memory FIFO | 🔽 📋 🗖 onchip_memory | | [clk1] |
| Avalon-ST Multi-Channel Shared memory Fill O Avalon-ST Round Robin Scheduler | s1 | Avalon Memory Mapped Slave | clk_50 ⊫° 0x000 |
| Avalon-ST Single Clock FIFO | | | |
| ···· On-Chip FIFO Memory | | | |
| On-Chip Memory (RAM or ROM) | | | |
| SDRAM V | | | |
| | < | | > |
| | | | |
| New Edit Add | Remove Edit 🛣 🔺 | Address Map | Filters Filter: Default |
| | | | |
| S Error: cpu.instruction_master: onchip_memory2.s1 cannot be at 0 | x2000 (0x0 or 0x10000 are acceptable) | | |
| Brror: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 (| 0x0 or 0x10000 are acceptable) | | |
| | | | |
| | | | |
| | | | |
| | | | |
| E <u>x</u> it He | elp 🛛 🖣 Prev 🛛 Next 🕨 | Generate | |
| | | | |

Figure 7-29 Updated CPU settings

19. Add the Library > Peripherals > Microcontroller Peripherals >PIO (Parallel I/O) component to the system, as shown in Figure 7-30 and Figure 7-31.



| 🖳 Altera SOPC Builder |
|---|
| <u>F</u> ile <u>E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> ools Nios II <u>H</u> elp |
| System Contents System Generation |
| Component Library |
| Project |
| New component |
| Library |
| |
| ⊕Interface Protocols |
| |
| Memories and Memory Controllers |
| Herlin Components |
| - Peripherals |
| ⊕ Debug and Performance |
| E Display |
| FPGA Peripherals |
| Microcontroller Peripherals Interval Timer |
| PIO (Parallel VO) |
| Multiprocessor Coordination |
| |
| Processor Additions |
| Processors |
| ۹ X |
| New Edit Add |

Figure 7-30 Add PIO



| 🗳 PIO (Parallel | I/O) - pio_O | |
|---|----------------------------|---------------|
| MogeCore PIO (Pa altera_avalor | rallel I/O) _pio [| Documentation |
| * Block Diagram | | ^ |
| clock● reset● avalon● conduit● | reset | |
| Basic Settings | | |
| Width (1-32 bits): | 8 | Ξ. |
| Direction: | O Bidir | |
| | 🔘 Input | |
| | 🔿 InOut | |
| | Output | |
| Output Port Reset Value | | |
| | | |
| Output Register Enable individual bit | setting/clearing | |
| | settingroteuring | |
| Edge capture register | | |
| Synchronously capt | ure | |
| Edge Type: | RISING 😽 | |
| Enable bit-clearing for | or edge capture register | |
| | | ¥ |
| | | |
| | | |
| | | |
| | Can | cel Finish |

Figure 7-31 Add PIO

20. Click **Finish** to use the default settings for this component. This closes the PIO wizard and returns to the window shown in **Figure 7-32**.



| La Altera SOPC Builder | | | |
|---|---------------------------------|---|-------------------------|
| <u>File E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> ools Nios II <u>H</u> elp | | | |
| System Contents System Generation | | | |
| Component Library | Target | Clock Settings | |
| Project | Device Family: Cyclone IV E | Name Source | MHz Add |
| New component | | clk_50 External | 50.0 |
| Library | | | Remove |
| How Bridges and Adapters | | | |
| Interface Protocols | | · · · · · · · · · · · · · · · · · · · | |
| Legacy Components Memories and Memory Controllers | Use Conn Module | Description | Clock Base |
| Memories and Memory Controllers Merlin Components | 🗹 🗆 сри | Nios II Processor | [clk] |
| | instruction_mast | | clk_50 |
| Debug and Performance | data_master jtag_debug_mod | Avaion Memory Mapped Master Avaion Memory Mapped Slave | [clk] [clk] = 0x0000 |
| ⊕ Display | I jtag_uebug_iidu | JTAG UART | [cik] |
| | avalon_itag_slav | | clk_50 = 0x0000 |
| Microcontroller Peripherals | ✓ □ onchip_memory2 | On-Chip Memory (RAM or ROM) | [clk1] |
| ···· Interval Timer | s1 | Avaion Memory Mapped Slave | clk 50 = 0x0000 |
| PIO (Parallel VO) | ✓ □ pio_0 | PIO (Parallel VO) | [clk] |
| Multiprocessor Coordination | | Avalon Memory Mapped Slave | clk_50 📄 0x0000 |
| BPLL | | | |
| Processor Additions Processors | | | |
| | | | |
| Q X | < | | > |
| New Edit Add | Remove | Address Map | Filters Filter: Default |
| | 1 | | |
| Error: cpu.instruction_master: onchip_memory2.s1 cannot be at (| | | |
| Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 | (0x0 or 0x10000 are acceptable) | | |
| | | | |
| | | | |
| | | | |
| | | | |
| E <u>x</u> it H | elp 🖣 Prev Next 🕨 | Generate | |

Figure 7-32 PIO

21. Rename **pio_0** to **pio_led** as shown in **Figure 7-33**.

| 🗳 Altera SOPC Builder | | | | | | | |
|---|---------|------------|----------------------------|---------------------------------------|---|-----------------|-----------------|
| <u>File Edit M</u> odule <u>System View Tools Nios II H</u> elp | | | | | | | |
| System Contents System Generation | | | | | | | |
| | | | | Ole als Cattioner | | | |
| Component Library | Targ | jet | | Clock Settings | | | |
| Project | Devi | ce Family: | Cyclone IV E 🛛 🗸 | Name | Source | MHz | Add |
| New component | | | | clk_50 | External | 50.0 | Remove |
| Library | | | | | | | Itemore |
| | | | | | | | |
| | | | l | · · · · · · · · · · · · · · · · · · · | | | |
| Egacy Components | Use | Conn | Module | Desci | ription | Clock | Base |
| Memories and Memory Controllers | | | 🗆 cpu | | Processor | [clk] | |
| Merlin Components | | | instruction_mast | | Memory Mapped Master | clk_50 | |
| Debug and Performance | | | data_master jtag_debug_mod | | Memory Mapped Master Memory Mapped Slave | [clk] [clk] | - Ox OOC |
| ∎ Display | | | itag uart | JTAG L | | [clk] | |
| | | | avalon_jtag_slav | ve Avalon | Memory Mapped Slave | clk_50 | ■ 0x00 C |
| Microcontroller Peripherals Interval Timer | | | onchip_memory | | p Memory (RAM or ROM) | [clk1] | |
| PIO (Parallel VO) | | | → s1 | | Memory Mapped Slave | clk_50 | =° 0x000 |
| Multiprocessor Coordination | | | pio_led s1 | | rallel I/O) Memory Mapped Slave | [clk] clk_50 | ⊜ 0x00 C |
| | | 1 . | 31 | Perdion | memory mapped stave | CIK_50 | |
| Processor Additions | | | | | | | |
| Processors | | | | | | | |
| QX | < | | | | | | > |
| | | | | | | | |
| New Edit Add | Ren | nove E | _dit 🔳 🔺 | | Address Map | <u>F</u> ilters | Filter: Default |
| |] | | | | | | |
| Error: cpu.instruction_master: onchip_memory2.s1 cannot be at (| | | | | | | |
| Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 | (0x0 or | 0x10000 a | are acceptable) | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | _ | | |
| E <u>x</u> it H | elp | • | Prev Next > | Genera | ite | | |
| | | | | | | | |

Figure 7-33 Rename PIO



22. Select **System > Auto-Assign Base Addresses** as shown in **Figure 7-34**. Then, select **File > Refresh System**. After that you will find that there is no error in the message window as shown in **Figure 7-35**.

| 🗳 Altera SOPC Builder | | | | | | | | | |
|---|----------|--|---|---|--|--|--|--|--|
| <u>F</u> ile <u>E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> ools Nios II <u>H</u> | elp | | | | | | | | |
| System Contents Auto-Assign Base Addresses | | | | | | | | | |
| Component Library Auto-Assign IRQs Insert Avalon-ST Adapters | | Clock Settings | | | | | | | |
| Project Show Transformed System | <u>^</u> | Device Family: Cyclone IV E | Name Source clk 50 External | MHz Add | | | | | |
| Library Avalon Verification Suite Bridges and Adapters | | | cik_50 External | Remove | | | | | |
| Interface Protocols Legacy Components | | Use Conn Module | Description | Clock Base | | | | | |
| -Memories and Memory Controllers -Merlin Components -Peripherals -B-Debug and Performance | = | Cpu instruction_master itag_debug_modi | Avalon Memory Mapped Master | [clk] clk_50 [clk] [clk] ■ 0x000 | | | | | |
| Display | | ∀ | JTAG UART Avalon Memory Mapped Slave | [clk] [clk] [clk_50 ■ 0x000 | | | | | |
| Interval Timer PlO (Parallel I/O) Uniterval Content on Plo Content o | | ✓ s1 ✓ □ pio_led | Avalon Memory Mapped Slave PIO (Parallel VO) | clk_50 | | | | | |
| PLL Processor Additions | | → s1 | Avalon Memory Mapped Slave | clk_50 ≅ 0x00 0 | | | | | |
| Processors | ✓ | | | | | | | | |
| | ~ | < | | > | | | | | |
| New Edit | Add | Remove Edit | Address Map | Filter: Default | | | | | |
| Error: cpu.instruction_master: onchip_memory2.s1 cannot be at 0x2000 (0x0 or 0x10000 are acceptable) Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 (0x0 or 0x10000 are acceptable) | | | | | | | | | |
| Exit Help Prev Next D Generate | | | | | | | | | |

Figure 7-34 Auto-Assign Base Addresses

| 💶 Altera SOPC Builder | | | | | | | | | | |
|---|---|---|---|--|--|--|--|--|--|--|
| $\underline{F} \text{ile} \underline{E} \text{dit} \underline{M} \text{odule} \underline{S} \text{ystem} \underline{V} \text{iew} \underline{T} \text{ools} \text{Nios}$ | II <u>H</u> elp | | | | | | | | | |
| System Contents System Generation | | | | | | | | | | |
| Component Library | Target | Clock Settings | | | | | | | | |
| Project | Device Family: Cyclone IV E 🗸 | Name | Source M | MHz | | | | | | |
| New component Library → Avalon Verification Suite → Bridges and Adapters | | clk_50 E | xternal 50. | | Remove | | | | | |
| ⊕…Interface Protocols ⊕…Legacy Components | Use Conn Module | Description | C | lock Base | | | | | | |
| Memories and Memory Controllers DDR2 SDRAM Controller with Un DDR3 SDRAM Controller with Un QDR II and QDR II-SRAM Control RLDRAM II Controller with UniPH' Traffic Generator and BIST Engir DMA Flash On-Chip SRAM Merin Components | Image: Second state Image: Secon | Nios II Processor Avalon Memory Ma Avalon Memory Ma JTAG UART ve Avalon Memory Ma | apped Master clk apped Master clk apped Master [clk apped Slave [clk [clk apped Slave clk RAM or ROM) [clk apped Slave clk [clk | k] k_50 k] # 0x00 k] k_50 # 0x00 k1] k_50 # 0x00 k] | IRQ 0 00108000 c 0011010 c 00080000 c 00110000 c | | | | | |
| | | | | _ | | | | | | |
| New Edit Add | Remove Edit | Add | dress Map <u>F</u> ilters. | Filter: Default | | | | | | |
| Warning: cpu: Custom Instruction components can be edited through the Component Editor. Warning: cpu: Disabling the assign CPUID control register value manually will no longer auto-assigns unique control register value. This option will always be turned on with default | | | | | | | | | | |
| | Exit Help I Prev | Next 🕨 Ge | enerate | | | | | | | |

Figure 7-35 No errors or warnings



23. Click the Generate button, which will pop up a window, as shown in **Figure 7-36**. Click Save, which bring up the window in **Figure 7-37**. Input the name, **DE0_NANO_SOPC**, and click the save button. The compilation will automatically start. If there are no errors in the generation, the window will show a message of success, as shown in **Figure 7-38**.

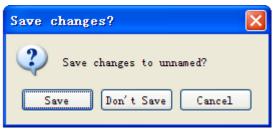


Figure 7-36 Generate SOPC

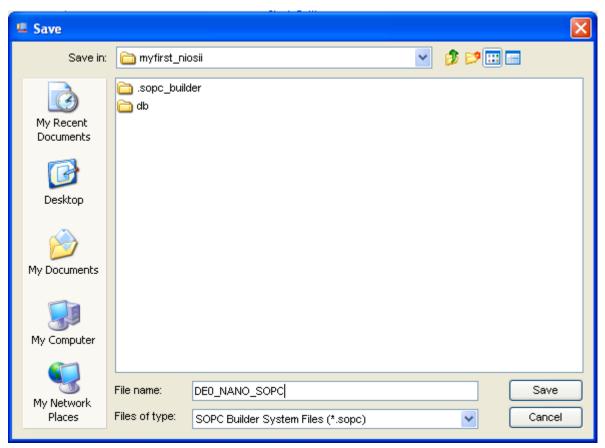


Figure 7-37 Generate SOPC



| Altera SOPC Builder - DE0_NANO_SOPC.sopc (D:\myfirst_niosii\DE0_NANO_SOPC_sopc) | |
|---|----------------|
| File Edit Module System View Tools Nios II Help ¹⁰ | |
| System Contents System Generation | |
| Coptions | |
| System module logic will be created in Verilog. | |
| | |
| Simulation. Create project simulator files. Run Simulator | |
| | |
| | |
| Nios II Software Build Tools for Eclipse | |
| | ~ |
| # 2011.02.24 13:23:09 (*) Generating Quartus symbol for top level: DE0_NANO_SOPC | |
| # 2011.02.24 13:23:09 (*) Generating Symbol D:/myfirst_niosii/DE0_NANO_SOPC.bsf | |
| # 2011.02.24 13:23:09 (*) Creating command-line system-generation script: D:/myfirst_niosii/DE0_NANO_SOPC_generation_script | |
| # 2011.02.24 13:23:09 (*) Running setup for HDL simulator: modelsim | |
| # 2011.02.24 13:23:10 (*) Completed generation for system: DE0_NANO_SOPC. | |
| # 2011.02.24 13:23:10 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED: | |
| SOPC Builder database : D:/myfirst_niosii/DE0_NANO_SOPC.ptf | |
| System HDL Model : D:/myfirst_niosii/DE0_NANO_SOPC.v System Generation Script : D:/myfirst_niosii/DE0_NANO_SOPC_generation_script | |
| # 2011.02.24 13:23:10 (*) SUCCESS: SYSTEM GENERATION COMPLETED. | |
| | > |
| | |
| Warning: cpu: Custom Instruction components can be edited through the Component Editor. Warning: cpu: Disabiling the assign CPUID control register value manually will no longer auto-assigns unique control register value. This option will always be turner Warning: cpu: Disabiling the assign CPUID control register value manually will no longer auto-assigns unique control register value. This option will always be turner | d op with doto |
| | a on with defa |
| | |
| | > |
| Exit Help Prev Next Decemente | |

Figure 7-38 SOPC Builder generation successful

24. Click **Exit** to exit the SOPC Builder and return to the window as shown in **Figure 7-39**.

| 🕊 Quartus II - D:/myfirst_niosii/myfirst_niosii - myfirst_niosii | |
|--|------------------|
| Eile Edit <u>V</u> iew Project <u>A</u> ssignments P <u>r</u> ocessing <u>T</u> ools <u>W</u> indow <u>H</u> elp | |
| | |
| i myfirst_niosii 🔽 🦅 🖉 🏈 🖉 🕨 🕨 🦻 👘 🤨 🗶 👱 👗 🕘 🛡 | |
| Project Navigator 🗗 🗙 | |
| Entity | |
| ▲ Cyclone IV E: EP4CE22F17C6 | |
| myfirst_niosii 🖁 | |
| QUARTU | \mathbf{C}^*] |
| | |
| Hierarchy 🖹 Files d ¹⁹ Design Units | |
| Status PX | |
| | |
| | uartus II |
| | ition |
| | itation |
| X Type Message | |
| 8 Type message | |
| | |
| | |
| 8 | > |
| System / Processing / Extra Info / Info / Warning / Critical Warning / Error / Suppressed / Flag / | |
| Service Message: | Locate |
| 0% | 00:00:00 |

Figure 7-39 Return to Quartus II after exiting SOPC Builder



25. Create a new Verilog HDL file, by selecting **File > New**, **Verilog HDL File** and click **OK**, as shown in **Figure 7-40** and **Figure 7-41**.

| - | Quartus I | I - I | D:/1 | yfirst_ | nios | ;ii/my |
|------|---------------------------|----------------|----------|--------------------|------|-----------|
| Eile | <u>E</u> dit <u>V</u> iev | v <u>P</u> r | oject | <u>A</u> ssignment | s P | rocessing |
| D | <u>N</u> ew | | | Ctrl+N | | ы |
| Ê | Open | | | Ctrl+O | | 00 |
| | Close | | | Ctrl+F4 | | f f |
| 溋 | New Project \ | <u>N</u> izard | | | | |
| Ê | Open P <u>r</u> oject | | | Ctrl+J | | |
| | Save Projec <u>t</u> | | | | | |
| | Clos <u>e</u> Project | | | | | |
| | Save | | | Ctrl+S | | |
| | Save <u>A</u> s | | | | | |
| Ø | Save All | | | Ctrl+Shift | +S | |
| | <u>File</u> Propertie | s | | | | |
| | Create / Upda | ate | | | ۲ | |
| | Export | | | | | |
| | Convert Prog | ra <u>m</u> mi | ng Files | i | | |
| ц, | Page Setup | | | | | |
| à | Print Pre <u>v</u> iew | | | | | |
| 8 | Print | | | Ctrl+P | | |
| | Recent Files | | | | ÷ | |
| | Recent Proje | ts | | | ۲ | |
| | E <u>x</u> it | | | Alt+F4 | | |
| | | | | | | |

Figure 7-40 New Verilog file

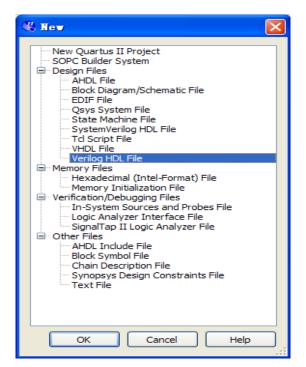


Figure 7-41 New Verilog File



33. Figure 7-42 show a blank Verilog file.

| Summer 11 | C - D:/my | vfirst | _niosi | i/myfi | rst_ | niosii | i — ny | first_ | niosi | ii | | | | | | | | | |
|-------------------|--------------|----------------------|------------|---------|---------------|----------------|----------------|--------------|----------------|---------|------------|--------|--------------|----|---|--------------|-----|-------------|------------|
| File Edit View | | | - | cessing | <u>T</u> ools | <u>W</u> indov | v <u>H</u> elp | | | | | | | | | | | | |
| | 1 1 1 | | | | mitte | | | . | *** 6 | 3. 1105 | | | | _ | | | | | |
| myfirst_niosii | | ~ | X ¥ | · _ | | | | | | 2 🔍 | <u> </u> | 🍝 🖲 | 0 (| => | | | | | |
| Project Navigator | | | | 6 | × | Đ | | Verilog 1 | | _ | × | | | | | | | | |
| Entity | | | | | | | ि ि हि र | } 镡 | <u>ب ‡</u> | 6 🌤 | % % | 6 ×6 | U | 2 | 2 | 267 268 3 | ab/ | | . <u> </u> |
| Cyclone IV E: | | 7C6 | | | | 1 | | | | | | | | | | | | | ~ |
| | E Files | d ⁹ Desig | gn Units | | | | | | | | | | | | | | | | |
| Module | % Progress | s | 🕲 Time | | | | | | | | | | | | | | | | |
| Module | % Progress | s | 🕲 Time | | | - II - m | | | | | | | | | | | | | 2 |
| × Time Maga | | s | 3 Time | | | (m |) | | | | | | | | | | | | > |
| X Type Mess | | | fo /\ Infr | | | | | | <u>r /\ Su</u> | ippress | ed /_ | Flag / | | | | | | Log | > cate |

Figure 7-42 A blank verilog file

34. Type the following Verilog into the blank file, as shown in **Figure 7-43**. The module **DE0_NANO_SOPC** is the system created by SOPC Builder and its Verilog can be found in the **DE0_NANO_SOPC.v** file, as shown in



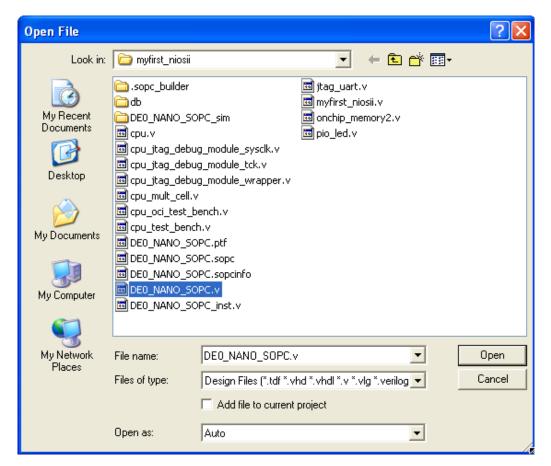


Figure 7-44 and Figure 7-45.

```
module myfirst_niosii
(
    CLOCK_50,
    LED
);
input
               CLOCK_50;
output [7:0]
            LED;
DE0_NANO_SOPC DE0_NANO_SOPC_inst
    (
      .clk_50
                                 (CLOCK_50),
      .out_port_from_the_pio_led (LED),
      .reset_n
                                  (1'b1)
    );
```

```
endmodule
```



| 🖑 Quartus II - D:/myfirst_niosii/my | first_niosii - myfirst_niosii | |
|---|---|----------|
| Eile Edit View Project Assignments Processing | g <u>T</u> ools <u>W</u> indow <u>H</u> elp | |
| · D 📽 🖬 🕼 😂 👗 🛍 🛍 🖗 🛛 | | |
| myfirst_niosii 💽 🔀 💅 🦉 | 🏈 🥙 🐵 🕨 🥙 🐌 😓 😓 🕲 🛡 | |
| Project Navigator 🗗 🗙 | 🕸 myfirst_niosii.v 🛛 | |
| Entity | | |
| Cyclone IV E: EP4CE22F17C6 | 32 // | ~ |
| Cyclone IV E: EP4CE22F17C6 | 33 // Major Functions: myfirst niosii | _ |
| | 34 // | |
| | 35 // | |
| | 36 // | |
| | 37 // Revision History : | |
| | 38 // | |
| | 39 // Ver : Author : Mod. Date : Change | |
| | 40 // V1.0 : Yagun-chang : 02/16/2011 : Initia | L Revis: |
| | 41 // | |
| | 42 Emodule myfirst_niosii | |
| | 43 ⊟ (44 CLOCK 50, | |
| | 45 LED | |
| | 46 -); | |
| | 47 input CLOCK 50; | |
| | 48 output [7:0] LED; | |
| | 49 DEO NANO SOPC DEO NANO SOPC inst | |
| | 50 | |
| | 51 .clk 50 (CLOCK 50), | _ |
| | 52 .out port from the pio led (LED), | |
| | 53 .reset_n (1'b1) | |
| | 54); | |
| | 55 - | |
| | 56 endmodule | |
| | 57 | ~ |
| Hierarchy 🖹 Files 🗗 Design Units | | > |
| | 0% | 00:00:00 |

Figure 7-43 Input verilog Text

| Open File | | | | | ? 🔀 |
|--|------------------------------|--|--|----------------------|----------------|
| Look in: | 🗀 myfirst_niosii | | • | + 🗈 💣 🎟 | . |
| My Recent Documents Desktop My Documents My Computer | 👼 cpu_jtag_debu | g_module_sysclk.v g_module_tck.v g_module_wrapper.v , mench.v h.v PC.ptf PPC.sopc PPC.sopc PPC.sopc | itag_uart myfirst_ni onchip_m pio_led.v | osii.v | |
| My Network Places | File name: Files of type: | DE0_NANO_SOPC.v Design Files (*.tdf *.vi Add file to current | hd ".vhdl ".v ".v | ▼ vlg *.verilog ▼ | Open Cancel |
| | Open as: | Auto | | • | |

Figure 7-44 Open DE0_NANO_SOPC.v

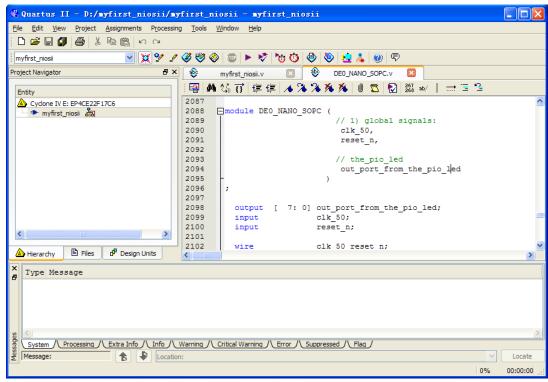


Figure 7-45 DE0_NANO_SOPC module

35. Save the newly created Verilog file as myfirst_niosii.v, as shown in Figure 7-46.

| Save As | | ?× |
|--|--|-------|
| Save jn: | 🖻 myfirst_niosii 💽 🔶 🖻 📸 🎫 | |
| My Recent Documents Desktop My Documents My Computer | sopc_builder db DE0_NANO_SOPC_sim cpu.v cpu_jtag_debug_module_sysclk.v cpu_jtag_debug_module_tck.v cpu_jtag_debug_module_wrapper.v cpu_unut_cell.v cpu_oci_test_bench.v cpu_test_bench.v DE0_NANO_SOPC.v DE0_NANO_SOPC_inst.v jtag_uart.v onchip_memory2.v pio_led.v | |
| T Idees | File <u>name:</u> myfirst_niosii.v <u>S</u> | ave |
| | Save as type: Verilog HDL Files (*.v *.vlg *.verilog) | ancel |
| | Add file to current project | |

Figure 7-46 Save the Verilog file

甘見



36. Compile the project, by selecting **Processing** > **Start Compilation**, as shown in **Figure 7-47**. **Figure 7-48** shows the compilation process.

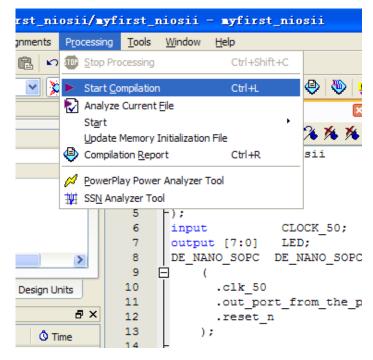


Figure 7-47 Start Compilation

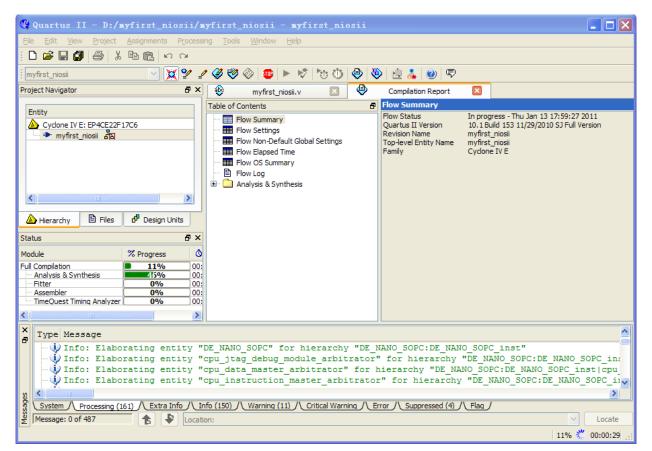
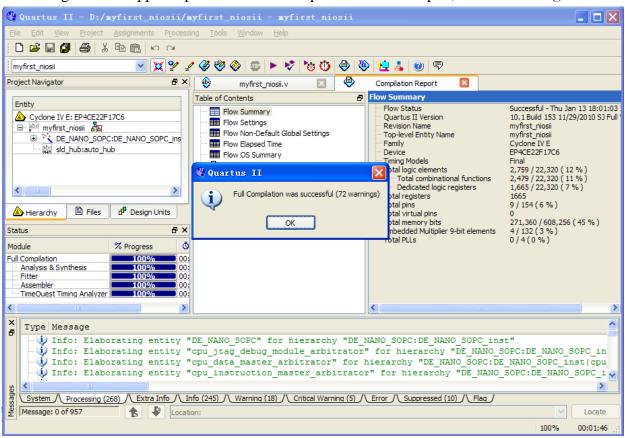


Figure 7-48 Execute Compile





37. A dialog box will appear upon successful completion of the compile, as shown in Figure 7-49.

Figure 7-49 Compile project completely

38. Now, we will assign the inputs and outputs of the circuit to specific pins. Select **Assignments** > **Pin Planner** from the menubar, as shown in **Figure 7-50**. The pin planner is shown in **Figure 7-51**.

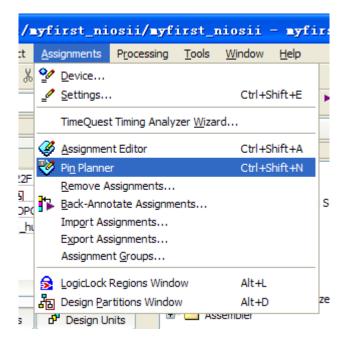


Figure 7-50 Pins menu



| Node Name | Direction | Location | I/O Bank | VREF Group | I/O Standard | Reserved |
|-----------|-----------|----------|----------|------------|-----------------|----------|
| CLOCK_50 | Input | | | | 2.5 V (default) | |
| LED[7] | Output | | | | 2.5 V (default) | |
| LED[6] | Output | | | | 2.5 V (default) | |
| LED[5] | Output | | | | 2.5 V (default) | |
| LED[4] | Output | | | | 2.5 V (default) | |
| LED[3] | Output | | | | 2.5 V (default) | |
| LED[2] | Output | | | | 2.5 V (default) | |
| LED[1] | Output | | | | 2.5 V (default) | |
| LED[0] | Output | | | | 2.5 V (default) | |

Figure 7-51 Blank Pins

39. Input Location values as shown in Figure 7-52.

| Node Name | Direction | Location | I/O Bank | VREF Group | I/O Standard | Reserved |
|---------------------------|-----------|----------|----------|------------|-----------------|----------|
| CLOCK_50 | Input | PIN_R8 | 3 | B3_N0 | 2.5 V (default) | |
| LED[7] | Output | PIN_L3 | 2 | B2_N0 | 2.5 V (default) | |
| 💿 LED[6] | Output | PIN_B1 | 1 | B1_N0 | 2.5 V (default) | |
| LED[5] | Output | PIN_F3 | 1 | B1_N0 | 2.5 V (default) | |
| LED[4] | Output | PIN_D1 | 1 | B1_N0 | 2.5 V (default) | |
| LED[3] | Output | PIN_A11 | 7 | B7_N0 | 2.5 V (default) | |
| LED[2] | Output | PIN_B13 | 7 | B7_N0 | 2.5 V (default) | |
| LED[1] | Output | PIN_A13 | 7 | B7_N0 | 2.5 V (default) | |
| LED[0] | Output | PIN_A15 | 7 | B7_N0 | 2.5 V (default) | |
| < <new node="">></new> | | | | | | |

Figure 7-52 Set Pins

40. Close the pin planner and recompile the project.

7.3 Download the Hardware Design

This section describes how to download the configuration file to the board.

Download the FPGA configuration file (i.e. the SRAM Object File (.sof) that contains the NIOS II based system) to the board by performing the following steps:

1. Connect the board to the host computer via the USB download cable.

- 2. Start the **NIOS II IDE**.
- 3. After the welcome page appears, click **Workbench**.
- 4. Select Tools > Quartus II Programmer.
- 5. Click Auto Detect. The device on your development board should be detected automatically.
- 6. Click the top row to highlight it.

7. Click Change File.

- 8. Browse to the myfirst_niosii project directory.
- 9. Select the programming file (myfirst_niosii.sof).
- 10. Click **OK**.
- 11. Click **Hardware Setup** in the top, left comer of the Quartus II programmer window. The Hardware Setup dialog box appears.

12. Select USB-Blaster from the currently selected hardware drop-down list box, as shown in **Figure 7-53**.

Note: If the appropriate download cable does not appear in the list, you must first install drivers for the cable. Refer to Quartus II Help for information on how to install the driver.

| 🖗 Hardware Setup | | | × |
|--|---|---------------|---------------------------------|
| Hardware Settings JTAG Se Select a programming hardware hardware setup applies only to Currently selected hardware: | setup to use when progra he current programmer w USB-Blaster [USB-0] No Hardware | | . This programming |
| Hardware USB-Blaster | USB-Blaster [USB-0] Server Local | Port USB-0 | Add Hardware Remove Hardware |
| | | | Close |

Figure 7-53 Hardware Setup Window

- 13. Click Close.
- 14. Make sure the **Program/Configure** option for the programming file (see **Figure 7-54** for an example).
- 15. Click Start.



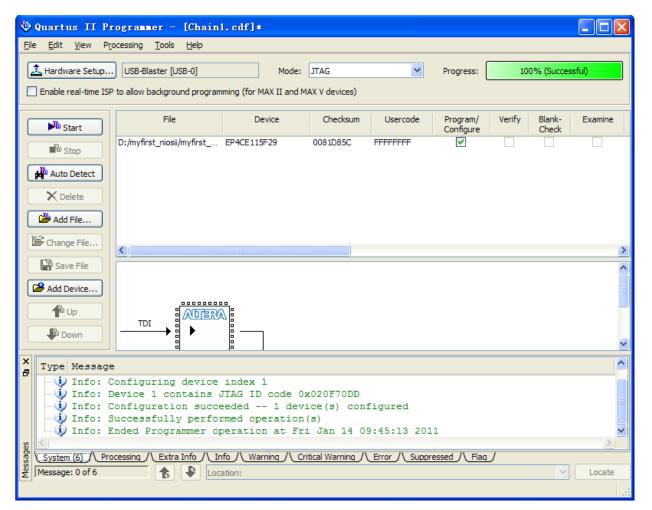


Figure 7-54 Quartus II Programmer

The Progress meter sweeps to 100% after the configuration finished. When configuration is complete, the FPGA is configured with the Nios II system, but it does not yet have a C program in memory to execute.

The Nios II IDE build flow is an easy-to-use graphical user interface (GUI) that automates build and makefile management. The Nios II IDE integrates a text editor, debugger, the Nios II flash programmer, the Quartus II Programmer, and the Nios II C-to-Hardware (C2H) compiler GUI. The included example software application templates make it easy for new software programmers to get started quickly. In this section you will use the Nios II IDE to compile a simple C language example software program to run on the Nios II system on your development board. You will create a new software project, build it, and run it on the target hardware. You will also edit the project, re-build it, and set up a debug session.

7.4 Create a hello_world Example Project

In this section you will create a new NIOS II C/C++ application project based on an installed example. To begin, perform the following steps in the NIOS II IDE:

1. Return to the NIOS II IDE.

Note: you can close the Quartus II Programmer or leave it open in the background if you want to reload the processor system onto your development board quickly.

- 2. Select File > New > NIOS II C/C++ Application to open the New Project Wizard.
- 3. In the New Project wizard, make sure the following things:
- a. Select the Hello World project template.
- b. Give the project a name. (hello_world_0 is default name)

c. Select the target hardware system's PTF file that is located in the previously created hardware project directory, as shown in **Figure 7-55**.



New Project

| Alles Hoject | |
|---|---|
| Nios II C/C++ Application Click Finish to create applicati D:\myfirst_niosii\Software\hello | on with a default system library as |
| Name: hello_world_0 ✓ Specify Location Location: D:\myfirst_niosii\S Select Target Hardware. SOPC Builder System PTF File: CPU: Select Project Template Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Memory Test Small Simple Socket Server Simple Socket Server Simple Socket Server Web Server Web Server (RGMII) Web Server (RGMII) | oftware Browse D:\myfirst_niosii\DE_NANO_SOPC.ptf Prowse cpu Prints 'Hello from Nios II' Details Hello World prints 'Hello from Nios II' to STDOUT. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory. |
| 0 | < Back Next > Finish Cancel |

Figure 7-55 Nios II IDE New Project Wizard

5. Click **Finish**. The NIOS II IDE creates the **hello_world_0** project and returns to the NIOS II C/C++ project perspective, as shown in **Figure 7-56**.



| Nios II C/C++ - hello_world.c - Nios II IDE | | | | | | | | | | | |
|---|---|--|---|--|--|---------------------------------------|------|--|--|--|--|
| File Edit Refactor Mavigate Segrch Project Tools Run Mindow Melp | | | | | | | | | | | |
| 📬 • 🖫 🗁 🗟 🚳 • 🚳 • 💽 • | | | | | | | | | | | |
| Nios II C/C++ Projects 🛛 📃 🗆 | 🖸 hello_world.c 🗙 | | | | | | - 8) | | | | |
| Nios II C/C++ Projects X C C C C C C C C C C C C C C C C C C C | <pre>/* * "Hello World" * * This example p * the Nios II 's * designs. It ru * device in your * The memory foc * using the star * * For a reduced</pre> | prints 'Hello standard', 'fu ins with or wi r system's har tprint of thi hdard reference footprint ver memory footp world" templat | ull_featur ithout the rdware. is hosted ce design. rsion of t rint for a te. | FII' to the STDOU ed', 'fast', and MicroC/OS-II RTO application is ~6 his template, and given applicatio igiven applicatio Path | 'low_cost' S and requ: 9 kbytes by . an explana | example ires a STDOUT y default | | | | | |
| < > | | | | | | | | | | | |
| | E | Writable | Smart Inser | •t 1:1 | | | | | | | |

Figure 7-56 Nios II IDE C++ Project Perspective for hello_world_0

When you create a new project, the NIOS II IDE creates two new projects in the NIOS II C/C++ Projects tab:

■ hello_world_0 is your C/C++ application project. This project contains the source and header files for your application.

■ hello_world_0_syslib is a system library that encapsulates the details of the Nios II system hardware.

Note: When you build the system library for the first time the NIOS II IDE automatically generates files useful for software development, including:

• Installed IP device drivers, including SOPC component device drivers for the NIOS II hardware system

• Newlib C library: a richly featured C library for the NIOS II processor.

• NIOS II software packages which includes NIOS II hardware abstraction layer, Nichestack TCP/IP Network stack, NIOS II host file system, NIOS II read-only zip file system and Micrium's μ C/OS-II realtime operating system (RTOS).

• **system.h:** a header file that encapsulates your hardware system.

• **alt_sys_init.c:** an initialization file that initializes the devices in the system.

• **Hello_world_0.elf:** an executable and linked format file for the application located in hello_world_0 folder under the Debug directory.

7.5 Build and Run the Program

In this section you will build and run the program.

To build the program, right-click the **hello_world_0** project in the Nios II C/C++ Projects tab and select **Build Project**. The **Build Project** dialog box appears and the IDE begins compiling the project. When compilation completes, a message 'Build complete' will appear in the Console tab, as shown in **Figure 7-57**.

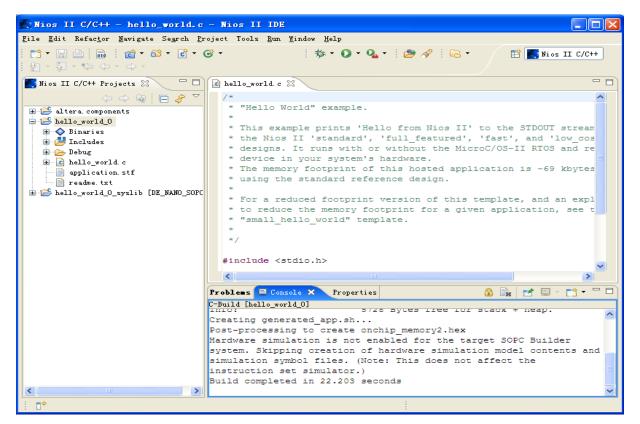


Figure 7-57 Nios II IDE hello_world_0 Build Completed

Note: If there appears in the console tab, an error, "region onchip_memory2 is full(hello_world_0.elf section .text). Region needs to be XXX bytes larger.", please right-click hello_world_0, select System Library Properties menu, then pop a window. In the System Library Properties window, select Small C Library, then click OK to close the window. Rebuild the project.



After a successful compilation, right-click the **hello_world_0** project, select **Run As > NIOS II Hardware**. The IDE will download the program to the target FPGA development board and begin execution. When the target hardware begins executing the program, the message '**Hello from Nios II!**' will appear in the NIOS II IDE Console tab, as shown in Figure 7-58 for an example.

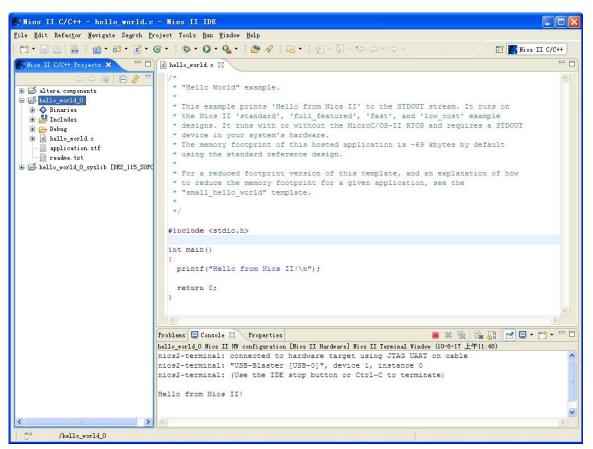


Figure 7-58 Hello_World_0 Program Output

Now you have created, compiled, and run your first software program based on NIOS II. And you can perform additional operations such as configuring the system properties, editing and re-building the application, and debugging the source code.

7.6 Edit and Re-Run the Program

You can modify the **hello_world.c** program file in the IDE, build it, and re-run the program to observe your changes, as it executes on the target board. In this section you will add code that will make the green LEDs, on the DEO-Nano board, blink.

Perform the following steps to modify and re-run the program:

1. In the hello_world.c file, add the text shown in blue in the example below:

#include <stdio.h>



```
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
printf("Hello from Nios II!\n");
int count = 0;
int delay;
while(1)
{
IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
delay = 0;
while(delay < 200000)
{
delay++;
}
count++;
}
return 0;
}
```

2. Save the project.

3. Recompile the project by right-clicking **hello_world_0** in the NIOS II C/C++ Projects tab and choosing **Run** > **Run** As > **Nios II Hardware**.

- Note: You do not need to build the project manually; the Nios II IDE automatically re-builds the program before downloading it to the FPGA.
- 4. Orient your development board so that you can observe LEDs blinking.

7.7 Why the LED Blinks

The Nios II system description header file, **system.h**, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios II hardware system. The **system.h** file is located in the in the **hello_world_0_syslib\Debug\system_description** directory, and is shown in Figure 7-59.

| Nios II C/C++ - system.h | - Nios II IDE | |
|--------------------------|--|-----|
| | | C++ |
| Nios II C/C++ P 🛛 🦳 🗖 | c hello_world.c h system.h | |
| <pre></pre> | <pre>/* * pio_led configuration * */ #define PIO_LED_NAME "/dev/pio_led" #define PIO_LED_TYPE "altera_avalon_pio" #define PIO_LED_TYPE "altera_avalon_pio" #define PIO_LED_BASE 0x00011000 #define PIO_LED_BASE 0x00011000 #define PIO_LED_DOTEST_BENCH_WIRING 0 #define PIO_LED_DRIVEN_SIM_VALUE 0 #define PIO_LED_HAS_TRI 0 #define PIO_LED_HAS_TN 0 #define PIO_LED_HAS_OUT 1 #define PIO_LED_CAPTURE 0 #define PIO_LED_DATA_WIDTH 8 #define PIO_LED_DATA_WIDTH 8 #define PIO_LED_EDGE_TYPE "NONE" #define PIO_LED_EDGE_TYPE "NONE" #define PIO_LED_BIT_CLEARING_EDGE_REGISTER 0 #define PIO_LED_BIT_MODIFYING_OUTPUT_REGISTER 0 #define PIO_LED_BIT_MODIFYING_OUTPUT_REGISTER 0 #define PIO_LED_FREQ 50000000 #define ALT_MODULE_CLASS_pio_led altera_avalon_pio #define ALT_MODULE_CLASS_pio_Seconds</pre> | ^ |
| | | ~ |
| | | |

Figure 7-59 The system.h file

If you look in the **system.h** file for the Nios II project example used in this tutorial, you will notice the **pio_led** function. This function controls the LEDs. The Nios II processor controls the PIO ports (and thereby the LEDs) by reading and writing to the register map. For the PIO, there are four registers: **data**, **direction**, **interruptmask**, **and edgecapture**. To turn the LED on and off, the application writes to the PIO's data register.

The PIO core has an associated software file **altera_avalon_pio_regs.h**. This file defines the core's register map, providing symbolic constants to access the low-level hardware. The **altera_avalon_pio_regs.h** file is located in the directory, **altera\10.1\ip\sopc_builder_ip\altera_avalon_pio**.

When you include the **altera_avalon_pio_regs.h** file, several useful functions that manipulate the PIO core registers are available to your program. In particular, the macro

IOWR_ALTERA_AVALON_PIO_DATA(base, data)



can write to the PIO data register, turning the LED on and off. The PIO is just one of many SOPC peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to Quartus II Version 10.1 Handbook Volume 5: Embedded Peripherals.

When developing your own designs, you can use the software functions and resources that are provided with the Nios II HAL. Refer to the Nios II Software Developer's Handbook for extensive documentation on developing your own Nios II processor-based software applications.

7.8 Debugging the Application

Before you can debug a project in the NIOS II IDE, you need to create a debug configuration that specifies how to run the software. To set up a debug configuration, perform the following steps:

1. In the **hello_world.c** , double-click the front of the line where you would like to set breakpoint, as shown in **Figure 7-60**.

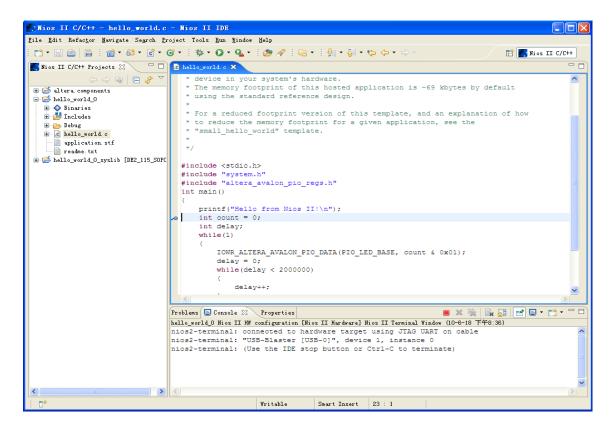


Figure 7-60 Set Breakpoint

- 2. To debug your application, right-click the application, **hello_world_0**, and select **Debug as** > **Nios II Hardware**.
- 3. If the **Confirm Perspective Switch** message box appears, click **Yes**.

After a moment, the main() function appears in the editor. A blue arrow next to the first line of code indicates that execution stopped at that line.

5. Select **Run** > **Resume** to resume execution.

When debugging a project in the Nios II IDE, you can pause, stop or single step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

Note: To return to the Nios II C/C++ project perspective from the debug perspective, click the two arrows >> in the top right corner of the GUI.

7.9 Configure System Library

In this section you will learn how to configure some advanced options in the Nios II IDE. By performing the following steps, you can change all the available settings:

1. In the Nios II IDE, right-click **hello_world_0** and select **System Library Properties**. The **Properties for hello_world_0_syslib** dialog box opens.

2. Click **System Library** in the tree on the left side. The **System Library** page contains settings related to how the program interacts with the underlying hardware. The settings have names that correspond to the targeted NIOS II hardware.

3. In the Linker Script box, observe which memory has been assigned for Program memory(.text), Read-only data memory(.rodata), Read/write data memory(.rwdata), Heap memory, and Stack memory, see Figure 7-61. These settings determine which memory is used to store the compiled executable program. You can also specify which interface you want to use for stdio, stdin, and stderr. You can also add and configure an RTOS for your application and configure build options to support C++, reduced device drivers, etc.

4. Select **onchip_memory2** for all the memory options in the **Linker Script** box, as shown in **Figure 7-61.**



| Properties for he | llo_world_0_syslib | | | |
|--|---|--|---|-------------------|
| type filter text | System Library | | \$~ * | ⇔ - |
| Info Builders C/C++ Build C/C++ Decementation C/C++ File Types C/C++ Include Paths C/C++ Indexer C/C++ Make Project Exposed Project Reforences Refactoring History System Library | Target Hardware SOPC Builder System: D:\myfirst_mic CPU: cpu System Library Contents RTOS: RTOS Options stdout: stdarr: stdarr: stdin: System clock timer: Timestamp timer: Max file descriptors: Program newer exits Support C++ Lightweight device driver API Link with profiling library Uninglemented instruction handler Software Components | sii\DE_UAND_SOPC.ptf none (single-threaded) jiteg_uart jiteg_uart v jiteg_uart v none 32 Clean exit (flush buffers) Reduced device drivers v Smll C library ModelSim only, no hardware support Run time stack checking | none Select O Use auto-generated linker script Frogram memory (text): onchip_memory2 Program memory (text): onchip_memory2 Read-only data memory (rodata): onchip_memory2 Read/write data memory (rodata): onchip_memory2 Keap memory: onchip_memory2 Stack memory: onchip_memory2 Use a separate exception stack Exception stack memory: Muriam superitien stack memory: Data separate provide stack | 2 💙 2 🍸 2 💙 |
| | | | Help Restore Defaults Appl. | y |
| 0 | | | OK Cancel | 1 |

Figure 7-61 Configuring System Library Properties

5. Click **OK** to close the **Properties for hello_world_0_syslib** dialog box and return to the IDE workbench.

Note: If you make changes to the system properties you must rebuild your project. To rebuild, right-click the hello_world_0 project in the Nios II C/C++ Projects tab and select Build Project.



Chapter 8

DEO-Nano Demonstrations

8.1 System Requirements

Make sure Quartus II and NIOS II are installed on your PC.

8.2 Breathing LEDs

This demonstration shows how to use the FPGA to control the luminance of the LEDs by means of pulse-width modulation (PWM) scheme. The LEDs are divided into two groups, while one group dims the other group brightens, vice versa. Users can change the PWM wave's duty ratio and frequency to control the LED luminance and repetition rate.

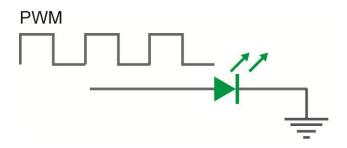
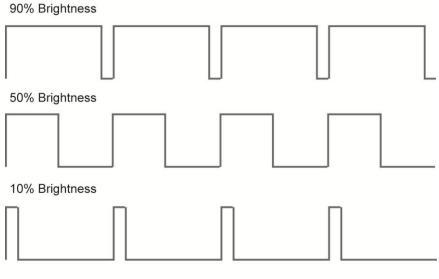


Figure 8-1 Shows a diagram of PWM signals to drive LED.





Pulse Width Modulation

Figure 8-2 Pulse Width Modulation

Figure 8-2 shows the relationship between duty cycle and LED luminance.

Demonstration Source Code

- Project directory: DE0_NANO_Default
- Bit stream used: DE0_NANO.sof

Demonstration Batch File

Demo Batch File Folder: DE0_NANO_Default\demo_batch

The demo batch file includes the following files:

• FPGA Configure File: DE0_NANO.sof

Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Connect USB cable to the DEO-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "DE0_NANO.bat" under the batch file folder, *DE0_NANO_Default \demo_batch.* This will load the demo into the FPGA.



8.3 ADC Reading

This demonstration illustrates steps which can be used to evaluate the performance of the 8-channel 12-bit A/D Converter. The DC 3.3V on the 2x13 header is used to drive the analog signals and by using a trimmer potentiometer, the voltage can be adjusted within the range of 0~3.3V. The 12-bit voltage measurements are indicated on the 8 LEDs. Since there are only 8 LEDs, only bit-4 through bit-11 from the ADC are represented on the LEDs.

Design Concept

This section describes the design concepts for this demo. Figure 8-3 shows the block diagram.

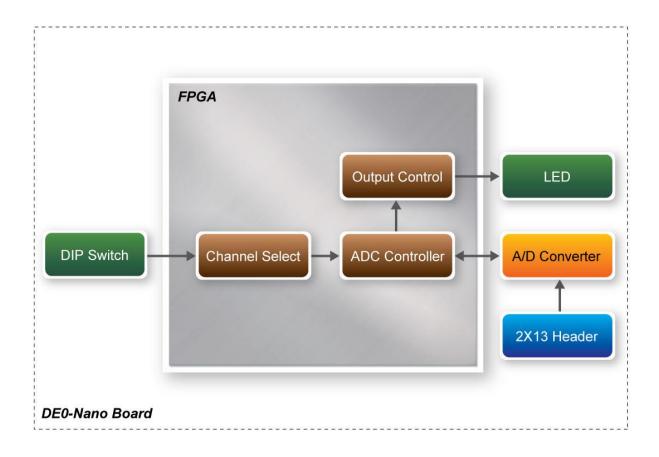


Figure 8-3 ADC Reading Block Diagram

The ADC Controller reads the voltage from the A/D converter through a serial interface and displays its measurement on the LEDs. The on-board dip-switch determines which channel to read from. Table 8-1 lists the DIP Switch settings and its corresponding ADC channel.



| 1able 8-1 | 1 DIP Switch Settings | | | | |
|------------------|-----------------------|-------------|--|--|--|
| DIP Switch (SW1) | Setting | ADC Channel | | | |
| | 0000 | Analog_In0 | | | |
| | 0001 | Analog_In1 | | | |
| | 0010 | Analog_In2 | | | |
| | 0011 | Analog_In3 | | | |
| | 0100 | Analog_In4 | | | |
| | 0101 | Analog_In5 | | | |
| | 0110 | Analog_In6 | | | |
| | 0111 | Analog_In7 | | | |

Table 8-1DIP Switch Settings

Figure 8-4 depicts the pin arrangement of the 2X13 header. Connect the trimmer to the ADC channel which is selected by the DIP Switches (Analog_In0 ~ Analog_In7).



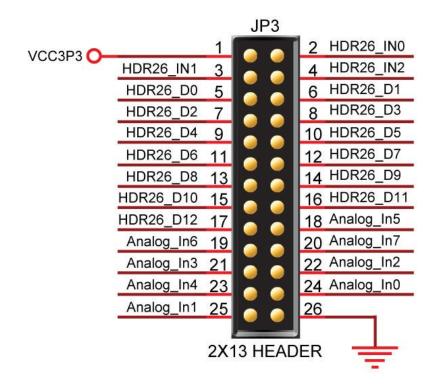


Figure 8-4 2X13 Header

System Requirements

The following items are required for the ADC Reading demonstration

- DE0-Nano board x1
- Trimmer Potentiometer x1
- Wire Strip x3

■ Hardware Setup

• Figure 8-5 shows the hardware setup for the ADC Reading demonstration.



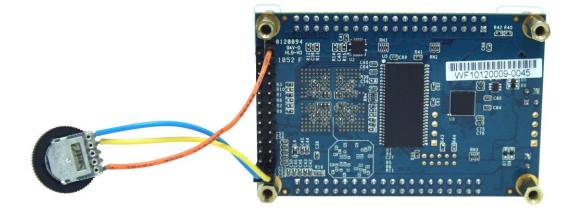


Figure 8-5 ADC Reading hardware setup

Note: the setup shown above is connected ADC channel 1.

Demonstration Source Code

- Project directory: DE0_NANO_ADC
- Bit stream used: DE0_NANO.sof

Demonstration Batch File

Demo Batch File Folder: DE0_NANO_ADC\demo_batch

The demo batch file includes the following files:

- FPGA Configure File: DE0_NANO.sof
- •

Demonstration Setup

- Make sure Quartus II is installed on your PC.
- Connect the trimmer to corresponding ADC channel to read from, as well as the +3.3V and GND signals.
- Adjust the DIP switch according to the ADC channel connected
- Connect USB cable to the DE0-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "DE0_NANO_ADC.bat" under the batch file folder, DE0_NANO_ADC\demo_batch. This will load the demo into the FPGA.
- Adjust the voltage using the trimmer and observe the measurements on the LEDs. Note a fully lit LED bar indicates the voltage is 3.3V and similarly no LED lit indicates 0V.



8.4 SOPC Demo

This demostration illustrates how to use the SOPC Builder to create a system with the following functions:

- Control accelerometer through 3-wire SPI interface
- Control analog to digital conversion through 4-wire SPI interface
- Access EEPROM memory through I2C interface
- Access EPCS memory

System Block Diagram

This section describes the SOPC System Block Diagram of this demo, as shown in Figure 8-6.

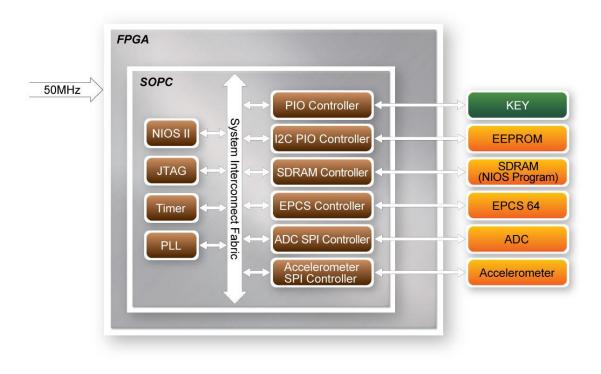


Figure 8-6 SOPC Block Diagram

A 50 MHz Clock is required for the SOPC System. A NIOS II processor is included in the system for flow control. The PLL is used to generate clocks, including 100 MHz, 10 MHz and 2MHz. The NIOS II Processor and SDRAM are running at 100 MHZ. The SDRAM is used to store the NIOS II Program. The ADC SPI Controller is running at 2 MHz. The other peripheral controllers are running at 10 MHz. The ADC SPI Controller and the Accelerometer SPI Controller are custom SOPC component. The source code, for these two controllers, is located in the "ip" folder under this Quartus II project. The other components are standard SOPC Builder components.



■ KEY

The KEY button is driven by PIO Controller with interrupt enabled. It is design to generate an interrupt event when users click KEY0 or KEY1. The interrupt event is used to terminate accelerometer and analog to digital conversion process in this demo.

For default, the interrupt is disabled in the PIO Controller. Users can enable it with the parameter setting as shown in below **Figure 8-7**.

| MogoCore PIO (Pa Attera_avalor | n_pio | 1 |
|-----------------------------------|---|---|
| * Basic Settings | | |
| Width (1-32 bits): | 2 | |
| Direction: | OBidir | |
| | Input | |
| | 🚫 InOut | |
| | 🔘 Output | |
| Output Port Reset Value | E 0×00000000000000000000000000000000000 | |
| Output Register | | |
| Enable individual bit | setting/clearing | |
| Edge capture registe | er | |
| Synchronously cap | | ' |
| Edge Type: | FALLING 🔽 | |
| Enable bit-clearing f | for edge capture register | |
| | | |
| Interrupt | | |
| | | |
| * Interrupt | | |

Figure 8-7 PIO Controller

Accelerometer Control

The accelerometer controller is a custom SOPC component developed by Terasic. The source code is available under the folder \DE0_NANO_SOPC_DEMO\ip\TARASIC_SPI_3WIRE.

In this demo, the accelerometer is controlled through a 3-wire SPI. Before reading any data from the accelerometer, master should set 1 on the SPI bit in the Register $0x31 - DATA_FORMAT$ register, as shown in below **Figure 8-8**, to set the device to 3-wire SPI mode.

| Register (|)x31- | -DATA_FO | RMA | T (Read/W | rite) |
|------------|-------|----------|-----|-----------|-------|
| | | | | | |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|-----------|-----|------------|----|----------|---------|-----|-----|
| SELF_TEST | SPI | INT_INVERT | 0 | FULL_RES | Justify | Rar | ige |

Figure 8-8 DATA_FORMAT Register



The data format is configured as 10 bits, right-justify, $\pm 2g$ mode. The output data rate is configured as 400 HZ. The X/Y/Z value is read using polling mode. Before reading X/Y/Z, the master needs to make sure data is ready by reading the register 0x30-INT_SOURCE, as shown below **Figure 8-9**, and checking the DATA_READY bit. In the demo, multiple-byte read of six bytes X/Y/Z, register from 0x32 to 0x37, is performed to prevent a change in data between reads of sequential register. Note, the output data is twos complement with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z.

| | Register 0x30—INT_SOURCE (Read Only) | | | | | |
|---|--------------------------------------|------------|------------|----------|--|--|
| | D7 | D6 | D5 | D4 | | |
| 9 | DATA_READY | SINGLE_TAP | DOUBLE_TAP | Activity | | |
| 1 | D3 | D2 | D1 | D0 | | |

Ity FREE FALL Watermark

Inactivity

Figure 8-9 Register 0x30

Overrun

The SPI timing scheme follows clock polarity (CPOL)=1 and clock phase (CPHA)=1. (CPOL)=1 means the clock is high in idle. (CPHA)=1 means data is captured on clock's rising edge and data is propagated on a falling edge. The timing diagram of 3-wire SPI is shown below **Figure 8-10**:

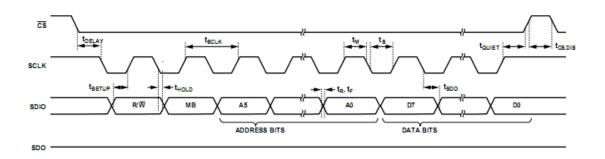


Figure 8-10 3-wire SPI Timing Diagram

ADC Control

The Analog to Digital Conversion is controller through a 4-wire SPI interface with the timing dialog given below **Figure 8-11**. Note, the DIN signal is used to specify the channel (IN0~IN7) for the next data conversion. The DOUT signal is used to read the data conversion result whose channel is specified in previous transaction. The first conversion result after power-up will be on IN0. The output format of conversion result is straight binary.



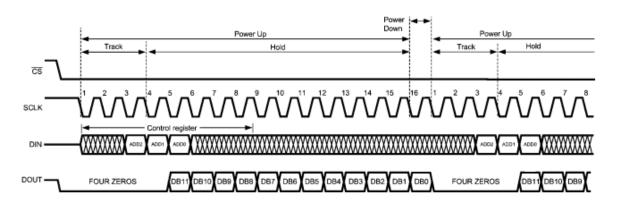


Figure 8-11 4-wire SPI Timing Diagram

EEPROM Control

EEPROM is accessed through the I2C interface. In this demo, I2C signal is toggle by NIOS II through the PIO controller. The I2C clock signal is driver by an OUTPUT PIO Controller and the I2C data signal is driver by a BIDIRECTION PIO Controller. The I2C C code is located in:

DE0_NANO_SOPC_DEMO\software\DE0_NANO\terasic_lib\I2C.c

EPCS Control

EPCS64 is accessed through the EPCS interface. In Quartus 10.0 or later, the EPCS pin assignment is required and should be connected the pins to EPCS Controller as shown below **Figure 8-12**:

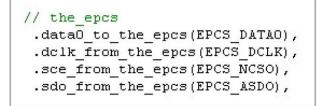


Figure 8-12 EPCS interface connection

For the EPCS access functions, users can refer to:

DE0_NANO_SOPC_DEMO\software\DE0_NANO\terasic_lib\Flash.c



Demonstration Source Code

- Project directory: DE0_NANO_SOPC_DEMO
- Bit stream used: DE0_NANO.sof
- NIOS II elf file: DE0_NANO.elf

Demonstration Batch File

• Demo Batch File Folder: DE0_NANO_SOPC_DEMO\demo_batch

The demo batch file includes the file:

- Batch File: test.bat and test_bashrc
- FPGA Configure File: DE0_NANO.sof
- Nios II Program: DE0_NANO.elf

Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Connect a USB cable to the DEO-Nano board and install USB Blaster driver if necessary.
- Execute the demo batch file "test.bat" under the batch file folder,
 - DE0_NANO_SOPC_DEMO\demo_batch. This will load the demo into the FPGA.
- After executing the batch file, a selection menu appears as follows:

| 🖾 Nios II EDS 10.1 [gcc3] | - 🗆 X |
|--|----------|
| Example designs can be found in | • |
| /cygdrive/c/altera/10.0/nios2eds/examples | |
| | |
| <pre>{You may add a startup script: c:/altera/10.0/nios2eds/user.bashrc></pre> | |
| Using cable "USB-Blaster [USB-0]", device 1, instance 0x00 | |
| Resetting and pausing target processor: OK | |
| Initializing CPU cache (if present) | |
| ок | |
| Downloaded 84KB in 1.4s (60.0KB/s) | |
| Verified OK | |
| Starting processor at address 0x020001C8 | |
| nios2-terminal: connected to hardware target using JTAG UARI on cable nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0 | |
| nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate) | |
| ,, , | |
| DE-Nano Demo | |
| | |
| - Selection function: | |
| – EØJACCELEROMETER – E1JADC | |
| - [2]EEPROM | |
| – [3]EPCS | |
| | |
| Select: | - |

• Input "0" to start the accelerometer demo. The demo starts by displaying the accelerometer's chip ID, and then continues by displaying the X/Y/Z values every 1.0 second. To terminate the demo, press KEY0 or KEY1 on the DE0-Nano board. Upon exiting the demo, the selection menu will be displayed.



| Nios II EDS 10.1 [gcc3] |
|---|
| 2lect:Demo ACCELEROMETER |
| l=E5 h |
| onitor Accerometer Value. Press KEYO or KEY1 to terminal the monitor process. \square |
| =−20 mg, Y=−4 mg, Z=872 mg |
| 32 mg, Y=8 mg, Z=976 mg |
| =−20 mg, Y=8 mg, Z=956 mg |
| 20 mg, Y=4 mg, Z=980 mg |
| =12 mg, Y=12 mg, Z=1004 mg |
| -36 mg, Y=−8 mg, Z=972 mg |
| =-32 mg, Y=8 mg, Z=968 mg |
| 28 mg, Y=8 mg, Z=980 mg |

• Input "1" to start Analog to Digital Conversion demo. The demo repeatedly displays the voltage on eight channels. To terminate the process, press KEY0 or KEY1 on the DE0-Nano board. Upon exiting the demo, the selection menu will be displayed.

| 🛤 Nios II EDS 10.1 [gcc3] | - 🗆 🗙 |
|--|-------|
| | - |
| Select:Demo ADC | |
| Nonitor ADC Value. Press KEYØ or KEY1 to terminal the monitor process. | |
| CH0=0.32 V | |
| CH1=0.29 V | |
| CH2=0.33 V | |
| CH3=0.37 V | |
| CH4=0.39 U | |
| CH5=0.40 U | |
| CH6=0.23 U | |
| CH7=0.32 U | |

• Input "2" to start EEPROM Content Dump demo. The demo displays the values in the first 16 bytes of the EEPROM. The demo automatically exists, and returns to the selection menu.

| 🛤 Nios II EDS 10.1 [gcc3] | _ 🗆 🗙 |
|---------------------------|----------|
| Select:Demo EEPROM | <u> </u> |
| Addr[00] = ffh | |
| Addr[01] = ffh | |
| Addr[02] = ffh | |
| Addr[03] = ffh | |
| Addr[04] = ffh | |
| Addr[05] = ffh | |
| Addr[06] = ffh | |
| Addr[07] = ffh | |
| Addr[08] = ffh | |
| Addr[09] = ffh | |
| Addr[10] = ffh | |
| Addr[11] = ffh | |
| Addr[12] = ffh | |
| Addr[13] = ffh | |
| Addr[14] = ffh | |
| Addr[15] = ffh | - |

• Input "3" to start EPCS demo. The demo displays the memory size of EPCS. The demo automatically exists, and returns to the selection menu.





8.5 G-Sensor

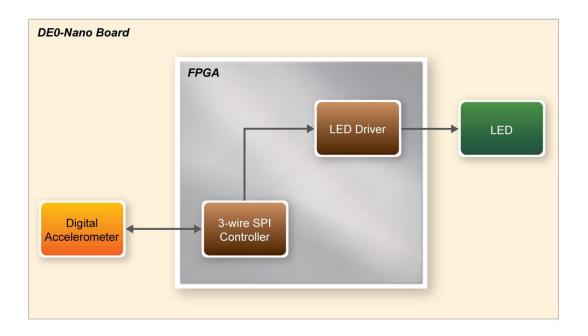
This demonstration illustrates how to use the digital accelerometer on the DEO-Nano board to measure the static acceleration of gravity in tilt-sensing applications. As the board is tilted from left to right and right to left, the digital accelerometer detects the tilting movement and displays it on the LEDs.



Figure 8-13 DE0-Nano on level surface

Design Concept

This section describes the design concepts for this demo. Figure 8-14 shows the block diagram.







In this demo, the accelerometer is controlled through a 3-wire SPI. Before reading any data from the accelerometer, the controller sets 1 on the SPI bit in the Register $0x31 - DATA_FORMAT$ register. The 3-wire SPI Controller block reads the digital accelerometer X-axis value, to determine the tilt of the board. The LEDs are lit up as if they were a bubble, floating to the top of the board.

Demonstration Source Code

- Project directory: DE0_NANO_GSensor
- Bit stream used: DE0_NANO_G_Sensor.sof

Demonstration Batch File

Demo Batch File Folder: DE0_NANO_GSensor\demo_batch

The demo batch file includes the following files:

• FPGA Configure File: DE0_NANO_G_Sensor.sof

Demonstration Setup

- Make sure Quartus II is installed on your PC.
- Connect USB cable to the DEO-Nano board and install the USB Blaster driver if necessary.
- Execute the demo batch file "test.bat" under the batch file folder, *DE0_NANO_GSensor\demo_batch*. This will load the demo into the FPGA.
- Tilt the DE0-Nano board from side to side and observe the result on the LEDs.

8.6 SDRAM Test by Nios II

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

System Block Diagram

Figure 8-15 shows the system block diagram of this demonstration. The system requires a 50 MHz clock provided from the board. The SDRAM controller is configured as a 32MB controller. The working frequency of the SDRAM controller is 100MHz, and the Nios II program is running in the SDRAM.



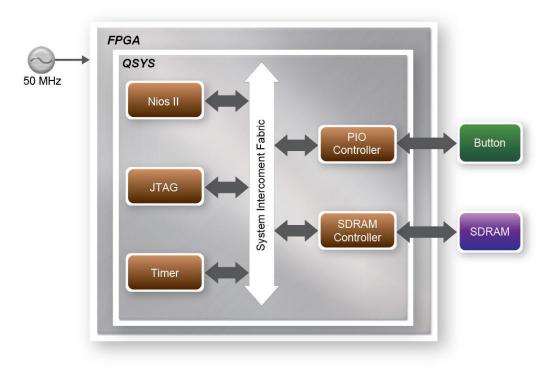


Figure 8-15 Block diagram of the SDRAM Basic Demonstration

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

Design Tools

- Quartus II 13.0 SP1
- Nios II Eclipse 13.0 SP1

Demonstration Source Code

- Quartus Project directory: DE0_NANO_SDRAM_Nios_Test
- Nios II Eclipse: DE0_NANO_SDRAM_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: DE0_NANO_SDRAM_Nios_Test \demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaster : DE0_NANO_SDRAM_Nios_Test.bat, DE0_NANO_SDRAM_Nios_Test.sh
- FPGA Configure File : DE0_NANO_SDRAM_Nios_Test.sof
- Nios II Program: DE0_NANO_SDRAM_Nios_Test.elf

Demonstration Setup

- Make sure Quartus II and Nios II are installed on your PC.
- Connect a USB cable to the DE0-Nano board and install USB Blaster driver if necessary. Execute the demo batch file "DE0_NANO_SDRAM_Nios_Test .*bat*" under the batch file folder, DE0_NANO_SDRAM_Nios_Test *demo_batch*
- *After Nios II program* is downloaded and executed successfully, a prompt message will be *displayed in n*ios2-terminal.
- Press **KEY1~KEY0** of the DE0-Nano board to start SDRAM verify process. Press **KEY0** for continued test.
- The program will display progressing and result information, as shown in Figure 8-16.

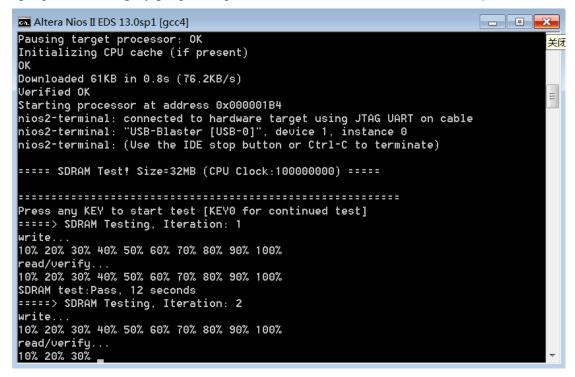


Figure 8-16 Display Progress and Result Information for the SDRAM Demonstration

Chapter 9



9.1 Programming the Serial Configuration Device

This section describes how to program the serial configuration device with Serial Flash Loader (SFL) function via the JTAG interface. User can program serial configuration devices with a JTAG indirect configuration (.jic) file. To generate JIC programming files with the Quartus II software, users need to generate a user-specified SRAM object file (.sof) of the circuit they wish to put in the serial configuration device. Next, users need to convert the SOF to a JIC file. To convert a SOF to a JIC file in Quartus II software, follow these steps:

• Convert SOF to JIC

- 1. Select File > Convert Programming Files...
- 2. In the **Convert Programming Files** dialog box, set the **Programming file type** field to **JTAG Indirect Configuration File** (.jic).
- 3. In the **Configuration device** field, specify the targeted serial configuration device, **EPCS64**.
- 4. In the **File name** field, browse to the target directory and specify an output file name.
- 5. Highlight the **SOF Data** row in the table, as shown in **Figure 9-1**.
- 6. Click Add File.
- 7. Select the SOF that you want to convert to a JIC file.
- 8. Click Open.



- 9. Highlight the Flash Loader and click **Add Device**, as shown in **Figure 9-2**.
- 10. Click **OK**. The Select Devices page displays.

| 🗳 Convert Programming File - | D:/CD/DE | Nano/DE | 0_Nan | o_ v1.0.9 _CDR0 | | | |
|---|--|---------|-------|------------------------|----------------------|--|--|
| <u>F</u> ile <u>W</u> indow | | | | | | | |
| Specify the input files to convert and the type of programming file to generate. You can also import input file information from other files and save the conversion setup information created here for future use. Conversion setup files | | | | | | | |
| Open Con <u>v</u> ersion Setup Data. | Open Conversion Setup Data Save Conversion Setup | | | | | | |
| Output programming file | | | | | | | |
| Programming file type: JTAG Indirect Conf | guration File (.jic | :) | | | * | | |
| Options Configuration device | EPCS64 | ~ | Mode: | Active Serial | × | | |
| File <u>n</u> ame: output_file.jic | | | | | | | |
| Advanced Remote/Local updat | e difference file: | NONE | | | ~ | | |
| Memory Map File | 2 | | | | | | |
| Input files to convert | | | | | | | |
| File/Data area | Pro | perties | | Start Address | Add <u>H</u> ex Data | | |
| Flash Loader SOF Data | Page_0 | | | <auto></auto> | Add Sof Page | | |
| | | | | | Add File | | |
| | | | | | | | |
| | | | | | Remove | | |
| | | | | | Up | | |
| | | | | | Do <u>w</u> n | | |
| | | | | | Properties | | |
| | | | Gene | rate Close | Неір | | |
| | | | | | .: | | |

Figure 9-1 Convert Programming Files Dialog Box



| 🔓 Convert Programming File | - D:/CD/DE_ | Nano/DE | 0_Nano | _ v1.0.9 _CDR | o 🔳 🗖 🔀 |
|--|-----------------------|---------|----------------|----------------------|--------------------------------------|
| <u>File W</u> indow | | | | | |
| Specify the input files to convert and the typ You can also import input file information from future use. Conversion setup files | | | | tup information cre | ated here for |
| Open Conversion Setup Data | a | | <u>S</u> ave | Conversion Setup. | |
| Output programming file | | | | | |
| Programming file type: JTAG Indirect Con | figuration File (.jic |) | | | ~ |
| Options Configuration devi | ce: EPCS64 | * | Mode: | Active Serial | |
| File name: output_file.jic | | | | | |
| Advanced Remote/Local upda | ate difference file: | NONE | | | ~ |
| Memory Map F | ile | | | | |
| Input files to convert | | | | | |
| File/Data area | Pro | perties | | Start Address | Add <u>H</u> ex Data |
| Flash Loader SOF Data | Page_0 EP4CE22F17 | | < | auto> | Add Sof Page Add Device Remove |
| | | | | | Do <u>w</u> n Properties |
| | | | <u>G</u> enera | ate Close | Help .:: |

Figure 9-2 Highlight Flash Loader

- 11. Select the targeted FPGA, Cyclone IV E EP4CE22, as shown in Figure 9-3.
- 12. Click OK. The Convert Programming Files page displays, should look like Figure 9-4.
- 13. Select the .sof file, and Click the **Properties**. Select Compression, click **OK**, as shown in **Figure 9-5**.
- 14. Click Generate.



| 🖗 Select Devices | | |
|---|--|--|
| Device family APEX20K Arria GX Arria II GX Arria II GZ Cyclone Cyclone II Cyclone III LS V Cyclone IV E Cyclone IV GX MAX II MAX V Stratix Stratix II Stratix II Stratix II Stratix II Stratix II Stratix IV | Device name □ EP4CE10 □ EP4CE115 □ EP4CE22 □ EP4CE30 □ EP4CE55 □ EP4CE66 □ EP4CE75 | New Import Export Edit Remove Uncheck All |
| | | |

Figure 9-3 Select Devices Page



| 🖆 Convert Programming File - | D:/CD/DE_Nano/DE | 0_Nano_ | v1.0.9_CDR0. | 🔳 🗖 🚺 |
|---|-------------------------|--|----------------------|----------------------|
| <u>File Window</u> | | | | |
| Specify the input files to convert and the type You can also import input file information from future use. | | | up information creat | ted here for |
| Conversion setup files Open Con <u>v</u> ersion Setup Data | | <u>S</u> ave C | Conversion Setup | |
| Output programming file | | | | |
| Programming file type: JTAG Indirect Config | juration File (.jic) | | | ~ |
| Options Configuration device | EPCS64 | Mode: | Active Serial | ~ |
| File name: output_file.jic | | | | |
| Advanced Remote/Local update | e difference file: NONE | | | * |
| Input files to convert | | | | |
| File/Data area | Properties | | Start Address | Add <u>H</u> ex Data |
| | Properties | | Start Address | |
| SOF Data DE0_NANO.sof | Page_0 EP4CE22F17 | <a< td=""><td>uto></td><td>Add Sof Page</td></a<> | uto> | Add Sof Page |
| | | | | <u>R</u> emove |
| | | | | Up |
| | | | | Down |
| | | | | Properties |
| | | | | |
| | | <u>G</u> enerat | e Close | Help |
| | | | | .: |

Figure 9-4 Convert Programming Files Page



| 🔓 Convert Programming | File - D:/CD/DE_ | Nano/DEO_Nano | _ v1.0.9 _CDRO | 🗖 🗖 🗙 |
|---|--|--|-----------------------|----------------------|
| <u>F</u> ile <u>W</u> indow | | | | |
| Specify the input files to convert an You can also import input file inform future use. Conversion setup files | nd the type of programming nation from other files and sa | file to generate. ave the conversion se | tup information crea | ted here for |
| Open Con <u>v</u> ersion S | Setup Data | <u>S</u> ave | Conversion Setup | |
| Output programming file | | | | |
| Programming file type: JTAG In | direct Configuration File (.jic) |) | | ~ |
| Options Configure | ation device: EPCS64 | Mode: | Active Serial | ~ |
| File <u>n</u> ame: output_ | file.jic | | | |
| Advanced Remote | 🖥 SOF File Prope | rties | × | ~ |
| ✓ Mem | | | _ | |
| Input files to convert File/Data area | OK Can | cel Help | L Address | Add <u>H</u> ex Data |
| Flash Loader EP4CE22 | | peraes | | Add <u>Hex Data</u> |
| DE0_NANO.sof | Page_0 EP4CE22F17 | < | auto> | Add File |
| | | | | |
| | | | | Цр |
| | | | | Down |
| | | | | Properties |
| | | Genera | ate Close | Help |
| | | | | .:: |

Figure 9-5 Compression the sof file

■ Write JIC File into Serial Configuration Device

To program the serial configuration device with the JIC file that you just created, add the file to the Quartus II Programmer window and follow the steps:

- 1. When the SOF-to-JIC file conversion is complete, add the JIC file to the Quartus II Programmer window:
 - i. Select **Tools > Programmer**. The **Chain1.cdf** window displays.
 - ii. Click Add File. From the Select Programming File page, browse to the JIC file.
 - iii. Click **Open**.



2. Program the serial configuration device by checking the corresponding **Program/Configure** box, a Factory default SFL image will be load (See **Figure 9-6**).

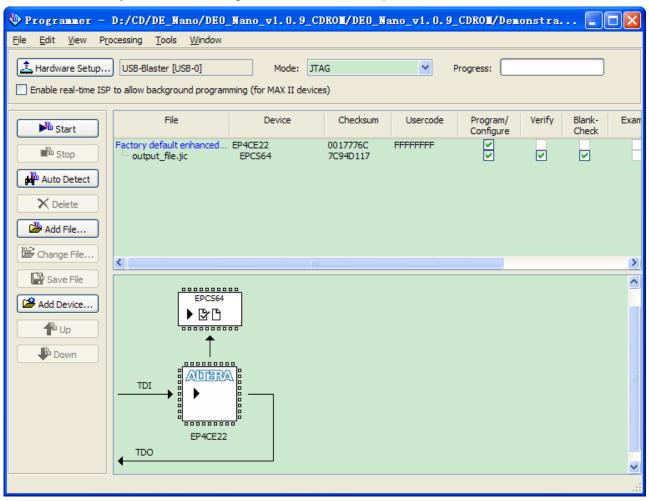


Figure 9-6 Quartus II programmer window with one JIC file

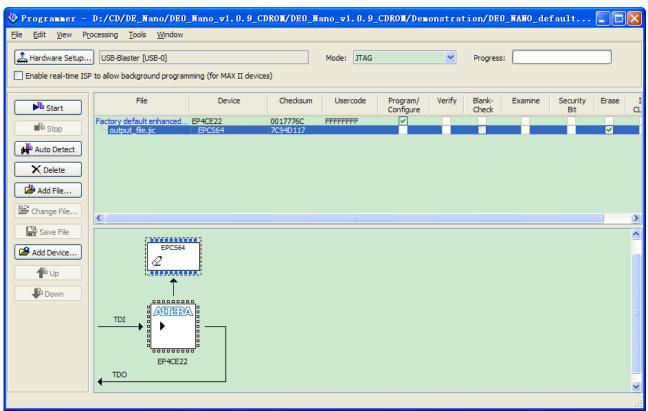
3. Click **Start** to program serial configuration device.

Erase the Serial Configuration Device

To erase the existed file in the serial configuration device, follow the steps listed below:

- 1. Select **Tools > Programmer**. The **Chain1.cdf** window displays.
- 2. Click Add File. From the Select Programming File page, browse to a JIC file.
- 3. Click **Open.**
- 4. Erase the serial configuration device by checking the corresponding Erase box, a Factory





default SFL image will be load (See Figure 9-7).

Figure 9-7 Erasing setting in Quartus II programmer window

5. Click **Start** to erase the serial configuration device.



9.2 EPCS Programming via nios-2-flash-programmer

Before programming the EPCS via nios-2-flash-programmer, users must add an EPCS patch file nios-flash-override.txt into the Nios II EDS folder. The patch file is available in the folder Demonstation\EPCS_Patch of DE0-Nano System CD. Please copy this file to the folder [QuartusInstalledFolder]\nios2eds\bin (e.g. C:\altera\11.1\nios2eds\bin)

If the patch file is not included into the Nios II EDS folder, an error will occur as shown in **Figure 9-8**.

| Using cable "USB-Blaster [USB-0]", device 1, insta | nce ØxØØ |
|---|----------|
| Recetting and nausing target processor: OK | |
| No EPCS layout data - looking for section [EPCS-010 | 92161 |
| Unable to use EPCS device | |
| Leaving target processor paused | |

Figure 9-8 EPCS Message

9.3 Revision History

| Version | Change Log |
|---------|--|
| V1.0 | Initial Version (Preliminary) |
| V1.3 | Add Table 3-1,3-2 and 3-3 |
| V1.4 | Modified Digital Accelerometer Description on page 31 |
| V1.5 | Modified ADC description on page 32 |
| V1.6 | Corrected Digital Accelerometer Schematic on page 23 |
| V1.7 | Modified Altera EPCS16 to be Spansion EPCS64 |
| V1.8 | Add SDRAM test section |
| V1.9 | Modify National logo to TI |
| V2.0 | Remove CD from Kit. CD is downloaded from Terasic Web. |
| V2.1 | Modify Figure 3-10 |
| V2.2 | Modify Figure 8-5 |
| V2.3 | Replace the board picture |

9.4 Copyright Statement

Copyright © Terasic Technologies. All rights reserved.

Always visit the DE0-Nano webpage for new applications.

We will continue providing interesting examples and labs on our DEO-Nano webpage. Please visit <u>www.altera.com</u> or <u>DEO-Nano.terasic.com</u> for more information.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Terasic: <u>P0082</u>