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





Copyright © 2003-2011 Terasic Technologies Inc. All Rights Reserved.





CHAPTER 1	INTRODUCTION	5
1.1 Features		5
1.2 About the	KIT	7
1.3 Getting He	elp	7
CHAPTER 2	DE0-NANO BOARD ARCHITECTURE	
2.1 Layout and	d Components	
2.2 Block Dia	gram of the DE0-Nano Board	9
2.3 Power-up	the DE0-Nano Board	
CHAPTER 3	USING THE DEO-NANO BOARD	
3.1 Configurir	ng the Cyclone IV FPGA	
3.2 General U	ser Input/Output	
3.3 SDRAM N	Memory	
3.4 I2C Serial	EEPROM	
3.5 Expansion	Headers	
3.6 A/D Conv	erter and 2x13 Header	
3.7 Digital Ac	celerometer	
3.8 Clock Circ	cuitry	
3.9 Power Sup	oply	
CHAPTER 4	DE0-NANO CONTROL PANEL	
4.1 Control Pa	nnel Setup	
4.2 Controllin	g the LEDs	
4.3 Switches a	and Pushbuttons	
4.4 Memory C	Controller	
4.5 Digital Ac	celerometer	
4.6 ADC		
4.7 Overall St	ructure of the DE0-Nano Control Panel	33

## 

CHAPTER 5	DE0-NANO SYSTEM BUILDER	
5.1 Introductio	n	
5.2 General De	esign Flow	
5.3 Using DE0	-Nano System Builder	
CHAPTER 6	TUTORIAL: CREATING AN FPGA PROJECT	
6.1 Design Flo	W	40
6.2 Before You	ı Begin	41
6.3 What You	Will Learn	45
6.4 Assign The	Device	45
6.5 Creating an	n FPGA design	49
6.6 Assign the	Pins	72
6.7 Create a D	efault TimeQuest SDC File	74
6.8 Compile Y	our Design	75
6.9 Program th	e FPGA Device	77
6.10 Verify Th	e Hardware	
CHAPTER 7	TUTORIAL: CREATING A NIOS II PROJECT	
7.1 Required F	eatures	
7.2 Creation of	f Hardware Design	
7.3 Download	the Hardware Design	
7.4 Create a he	llo_world Example Project	
7.5 Build and I	Run the Program	
7.6 Edit and R	e-Run the Program	
7.7 Why the L	ED Blinks	
7.8 Debugging	the Application	
7.9 Configure	System Library	
CHAPTER 8	DE0-NANO DEMONSTRATIONS	
8.1 System Re	quirements	
8.2 Breathing	LEDs	
8.3 ADC Read	ing	
8.4 SOPC Den	10	
8.5 G-Sensor		

## APERA.

CHAPTER 9	APPENDIX	145
9.1 Appendix		145
CHAPTER 10	APPENDIX	153
10.1 Revision H	listory	153
10.2 Copyright	Statement	153



## 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 16 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 DE0-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
  - o On-board USB-Blaster circuit for programming
  - Altera serial configuration device EPCS16
- 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
  - 2Kb I2C EEPROM
- General user input/output
  - o 8 green LEDs
  - o 2 debounced pushbuttons
  - o 4-position DIP switch
- G-Sensor
  - o ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)
- A/D Converter
  - o 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)
  - o DC 5V pin for each GPIO header (2 DC 5V pins)
  - o 2-pin external power header (3.6-5.7V)

## 1.2 About the KIT

The kit comes with the following contents:

- DE0-Nano board
- System CD-ROM.
- USB Cable

The system CD contains technical documents for the DE0-Nano board, which includes component datasheets, demonstrations, schematic, and user manual.

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: +886-3-550-8800
- 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 DE0-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)





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

#### 2.2 Block Diagram of the DEO-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 DEO-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 DE0-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.



# Using the DE0-Nano Board

This chapter gives instructions for using the DE0-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 EPCS16 device**

The DE0-Nano board contains an Altera EPCS16 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 EPCS16 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 DE0-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 3-1**. Each pushbutton is debounced using a Schmitt Trigger circuit, as indicated in **Figure 3-2**. 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 3-1 Connections between the push-buttons and Cyclone IV FPGA



Figure 3-2 Pushbuttons debouncing

#### ■ LEDs

There are 8 green user-controllable LEDs on the DE0-Nano board. The eight LEDs, which are presented in Figure 3 5, 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 3-3 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		

#### Table 3-2 Pin Assignments for LEDs

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



		0	
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 3-4**.



Figure 3-4 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 3-5** illustrates its connections with the Cyclone IV FPGA.



Figure 3-5 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 3-6** shows the I/O distribution of the GPIO connectors.





Figure 3-6 Pin arrangement of the GPIO expansion headers

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



Figure 3-7 Pin1 locations of the GPIO expansion headers

		<u> </u>	
Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_0_IN0	PIN_A8	<b>GPIO Connection DATA</b>	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	<b>GPIO-0</b>	<b>Pin Assignment</b>	İS
-----------	---------------	-----------------------	----



GPIO_02	PIN_A2	<b>GPIO Connection DATA</b>	3.3V
GPIO_03	PIN_A3	<b>GPIO Connection DATA</b>	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

terasic Terasic DE0-Nano User Manual



GPIO_16	PIN_R12	<b>GPIO Connection DATA</b>	3.3V
GPIO_17	PIN_T11	<b>GPIO Connection DATA</b>	3.3V
GPIO_18	PIN_T10	<b>GPIO Connection DATA</b>	3.3V
GPIO_19	PIN_R11	<b>GPIO Connection DATA</b>	3.3V
GPIO_110	PIN_P11	<b>GPIO Connection DATA</b>	3.3V
GPIO_111	PIN_R10	<b>GPIO Connection DATA</b>	3.3V
GPIO_112	PIN_N12	<b>GPIO Connection DATA</b>	3.3V
GPIO_113	PIN_P9	GPIO Connection DATA	3.3V
GPIO_114	PIN_N9	<b>GPIO Connection DATA</b>	3.3V
GPIO_115	PIN_N11	GPIO Connection DATA	3.3V
GPIO_116	PIN_L16	GPIO Connection DATA	3.3V
GPIO_117	PIN_K16	<b>GPIO Connection DATA</b>	3.3V
GPIO_118	PIN_R16	<b>GPIO Connection DATA</b>	3.3V
GPIO_119	PIN_L15	<b>GPIO Connection DATA</b>	3.3V
GPIO_120	PIN_P15	<b>GPIO Connection DATA</b>	3.3V
GPIO_121	PIN_P16	<b>GPIO Connection DATA</b>	3.3V
GPIO_122	PIN_R14	<b>GPIO Connection DATA</b>	3.3V
GPIO_123	PIN_N16	GPIO Connection DATA	3.3V
GPIO_124	PIN_N15	<b>GPIO Connection DATA</b>	3.3V
GPIO_125	PIN_P14	<b>GPIO Connection DATA</b>	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 3-8**. 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 3-8 Pin distribution of the 2x13 Header

Figure 3-9 shows the connections on the 2x13 header, A/D converter and Cyclone IV device.



Figure 3-9 Wiring for 2x13 header and A/D converter

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





Figure 3-10 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	<b>GPIO Connection DATA[2]</b>	3.3V
GPIO_2[3]	PIN_C16	<b>GPIO Connection DATA[3]</b>	3.3V
GPIO_2[4]	PIN_C15	<b>GPIO Connection DATA[4]</b>	3.3V
GPIO_2[5]	PIN_D16	GPIO Connection DATA[5]	3.3V
GPIO_2[6]	PIN_D15	<b>GPIO Connection DATA[6]</b>	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	<b>GPIO Connection DATA[9]</b>	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	<b>GPIO Connection DATA[12]</b>	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
14010	• •		LOSIGNMENTED			LICHUCI

 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 3-11 shows the connections between the ADXL345 and the Cyclone IV E device.



Figure 3-11 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	<b>Pin Assignmen</b>	nts for Digital A	Ccelerometer
1abic 3-10	I III Assignmen	its for Digital F	

#### **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 use as a source clock to drive the phase lock loops (PLL) circuit. The clock distribution on the DE0-Nano board is shown in **Figure 3-12**.





### 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 3-13**.



Figure 3-13 Portable Battery Connection

#### Power Distribution System

Figure 3-14 shows the power distribution system on the DE0-Nano board.



Figure 3-14 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 DE2-115 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.
- 7. 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.

terasic Terasic DE0-Nano User Manual



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 DE0-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 DE0-Nano board.





Figure 4-2 The DE0-Nano Control Panel concept

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.

terasIC Terasic DE0-Nano User Manual

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 read the eight-channel 12-bit analog-to-digital converter. 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 DEO-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 DE0-Nano board via the DE0-Nano System Builder. The general design flow is illustrated in **Figure 5-1**.

To create a new system using the DE0-Nano System Builder, begin by launching the DE0-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 DE0-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 DE0-Nano board, using the JTAG programming mode.



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

## 5.3 Using DEO-Nano System Builder

This section provides the detailed procedures on how the to use the DE0-Nano System Builder.

#### ■ Install and launch the DE0-Nano System Builder

The DE0-Nano System Builder is located in the directory: "*Tools\DE0\_NANO\_SystemBuilder*" on the DE0-Nano System CD. Users can copy the whole folder to a host computer without installing the utility. Launch the DE0-Nano System Builder by executing the DE0\_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	
DEO-Nano FPGA Board	CLOCK       LED x 8         Button x 2       Dip Switch x 4         SDRAM, 32MB       ADC         EEPROM, 2Kb       EPCS         Accelerometer       2x13 Pin Header         GPIO-0 Header         None       Prefix Name:         GPIO-1 Header       Image: Colspan="2">GPIO-1 Header	
Load Setting     Generate       Save Setting     Exit	None	

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	
NIVERSITY PROGRAM	System Configuration Project Name: DE0_NANO
DEO-Nano FPGA Board	Image: CLOCK       Image: LED x 8         Image: Button x 2       Image: Dip Switch x 4         Image: SDRAM, 32MB       Image: ADC         Image: SDRAM, 32MB <t< th=""></t<>
Load Setting Generate	GPIO-1 Header None Prefix Name:
Save Setting Exit	

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 DE0-Nano as shown in **Figure 5-4**. Each component of the DE0-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 DE0-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	Board	CLOCK Button x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header None Prefix Name: GPIO-1 Header	<ul> <li>✓ LED x 8</li> <li>✓ Dip Switch x 4</li> <li>✓ ADC</li> <li>✓ EPCS</li> <li>✓ 2x13 Pin Header</li> </ul>
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 DE0-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 LED x 8 Button x 2 Dip Switch x 4 SDRAM, 32MB ADC EEPROM, 2Kb EPCS Accelerometer C2x13 Pin Header  GPIO-0 Header Prefix Name: GPIO-1 Header
Load Setting Generate	None
Save Setting Exit	Prefix Name:

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	
UNIVERSITY BROGRAM		Project Name:	
		DE0_NANO	
DE0-Nano FPGA	Board	CLOCK	☑ LED x 8
		🗹 Button x 2	Dip Switch x 4
		SDRAM, 32MB	ADC
		🗹 EEPROM, 2Kb	☑ EPCS
		Accelerometer	🗹 2x13 Pin Header
		GPIO-0 Header	
		D5M - 5M Pixel Cam	era 🗸 🏹
		Prefix Name:	
		GPIO-1 Header	
Load Setting	Generate	None	*
		Prefix Name:	
Save Setting	Exit	FISHA NELLIS.	

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 System Builde	r
	The mes generated by DE0-1 and System Dunde	<b>_</b>

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 DE0-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.



Found New Hardw	vare Wizard
	Welcome to the Found New Hardware Wizard Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission). Read our privacy policy Can Windows connect to Windows Update to search for software?
	<ul> <li>Yes, now and every time I connect a device</li> <li>No, not this time</li> <li>Click Next to continue.</li> </ul>
	< Back Next > Cancel

#### 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.
<ul> <li>Search for the best driver in these locations.</li> </ul>
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 VBrowse
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.



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.

Found New Hardware Wize	ard
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: Altera USB-Blaster
	< Back Finish Cancel

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 Project	Vizard	
Introduction	n	
The New Project Wiz Proje Nam Proje Targ EDA You can change the :	Izard helps you create a new project and preliminary project settings, including the following: ject name and directory me of the top-level design entity ject files and libraries get device family and device A tool settings e settings for an existing project and specify additional project-wide settings with the Settings command (Assignments of the Settings dialog box to add functionality to the project.	menu). You can use
Don't show me th	this introduction again)	
	< Back Next > Finish Can	cel <u>H</u> elp

#### 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	×
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	
Rack Next > Finish Cancel He	elp

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**.



IV E       IV E       Padgage:       Any       IV         Padgage:       Any       Pin gount:       Any       Image: Pin gount:       Any       Image: Pin gount:       Image: Pin gount:	Device family					Show in 'Ava	lable devices' list		
Core Voltage         LEs         User I/Os         Memory Bits         Embedded multiplier 9-bit elements         PLL         I           1.2V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	Eamily: Cyclon	e IV E			~	Package:	Any		~
Core Voltage         LEs         User I/Os         Memory Bits         Embedded multiplier 9-bit elements         PLL         I           1.2V         22320         80         608256         132         4         20           1.2V         22320         80         608256         132         4         20           1.2V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	Devices: All								
selected by the Fitter te selected in 'Available devices' list Core Voltage LEs User I/OS Memory Bits Embedded multiplier 9-bit elements PLL 11 1.2V 22320 80 608256 132 4 20 1.0V 22320 80 608256 132 4 20 1.2V 22320 154 608256 132 4 20	Devices. All					Pin <u>c</u> ount:	Any		×
Core Voltage         LEs         User I/Os         Memory Bits         Embedded multiplier 9-bit elements         PLL         II           1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	Target device -					Sp <u>e</u> ed grade	: Any		~
Core Voltage         LEs         User I/Os         Memory Bits         Embedded multiplier 9-bit elements         PLL         II           1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20		selected by the Fitter				Show ad	vanced devices		
Core Voltage         LEs         User I/Os         Memory Bits         Embedded multiplier 9-bit elements         PLL         I           1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	_								
1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	Specific dev	ice selected in 'Availab	e devices'	list		HardCop	y compatible only		
1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	O Other: n/a								
1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20									
1.2V         22320         80         608256         132         4         20           1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20									
1.0V         22320         80         608256         132         4         20           1.2V         22320         154         608256         132         4         20	vailable devices:	1							
1.2V 22320 154 608256 132 4 20	<u>v</u> ailable devices: Name		LEs	User I/Os	Mem	ory Bits	Embedded multiplier 9-bit elements	PLL	al (
	Name	Core Voltage				-	•		_
1.2V 22320 154 608256 132 4 20	Name	Core Voltage	22320	80	608256	13	32	4	20
	Name P4CE22E22I7 P4CE22E22I8L	Core Voltage 1.2V 1.0V	22320 22320	80 80	608256 608256	13	32 32	4	20 20 20
1.2V 22320 154 608256 132 4 20	Name P4CE22E22I7 P4CE22E22I8L P4CE22E17A7	Core Voltage 1.2V 1.0V 1.2V	22320 22320 22320	80 80 154	608256 608256 608256	13 13 13	32 32 32	4 4 4	20 20 20
4 74 77 77 77 77 77 77 77 77 77 77 77 77	Name           EP4CE22E22I7           EP4CE22E22I8L           EP4CE22E17A7           EP4CE22F17A7	Core Voltage 1.2V 1.0V 1.2V 1.2V 1.2V	22320 22320 22320 22320 22320	80 80 154 154	608256 608256 608256 608256	1: 1: 1: 1: 1:	12 12 12 12	4 4 4 4 4	20 20 20 20
1.2V 22320 154 608256 132 4 20	Name           EP4CE22E22I7           EP4CE22E22I8L           EP4CE22F17A7           EP4CE22F17C6           EP4CE22F17C7	Core Voltage 1.2V 1.0V 1.2V 1.2V 1.2V	22320 22320 22320 22320 22320	80 80 154 154	608256 608256 608256 608256	1: 1: 1: 1: 1: 1:	32 32 32 32 32 32	4 4 4 4 4	20 20 20 20
1.0V 22320 154 608256 132 4 20	vailable devices: Name P4CE22E2217 P4CE22E2218 P4CE22F17A7 P4CE22F17C6 P4CE22F17C7 P4CE22F17C8 P4CE22F17C8	Core Voltage           1.2V           1.0V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V	22320 22320 22320 22320 22320 22320 22320	80 80 154 154 154 154 154	608256 608256 608256 608256 608256 608256 608256	1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1:	12 12 12 12 12 12 12 12 12	4 4 4 4 4 4 4 4	20 20 20 20 20 20 20 20 20
	Name P4CE22E22I7 P4CE22E22I8L P4CE22E17A7	Core Voltage 1.2V 1.0V 1.2V	22320 22320 22320	80 80 154	608256 608256 608256	13 13 13	32 32 32	4 4 4	
	Name           P4CE22E22I7           P4CE22E22I8L           P4CE22F17A7           P4CE22F17C6           P4CE22F17C7	Core Voltage           1.2V           1.0V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V	22320 22320 22320 22320 22320 22320	80 80 154 154 154	608256 608256 608256 608256 608256	1: 1: 1: 1: 1: 1:	32 32 32 32 32 32	4 4 4 4 4 4	20 20 20 20 20 20
	Name           P4CE22E22I7           P4CE22E22I8L           P4CE22F17A7           P4CE22F17C6           P4CE22F17C7           P4CE22F17C8	Core Voltage           1.2V           1.0V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V	22320 22320 22320 22320 22320 22320 22320	80 80 154 154 154 154 154	608256 608256 608256 608256 608256 608256	1: 1: 1: 1: 1: 1: 1:	32 32 32 32 32 32 32	4 4 4 4 4 4 4 4	20 20 20 20 20 20 20
	Name           P4CE22E22I7           P4CE22E22I8L           P4CE22F17A7           P4CE22F17C6           P4CE22F17C7           P4CE22F17C8	Core Voltage           1.2V           1.0V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V           1.2V	22320 22320 22320 22320 22320 22320 22320 22320	80 80 154 154 154 154 154 154	608256 608256 608256 608256 608256 608256 608256	1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1:	12 12 12 12 12 12 12 12 12	4 4 4 4 4 4 4 4	20 20 20 20 20 20 20 20 20

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.



🐇 New 🛛 🔀
New Quartus II Project SOPC Builder System Design Files AHDL File Block Diagram/Schematic File EDIF File Qsys System File State Machine File SystemVerilog HDL File Td Script File Verilog HDL File Verilog HDL File Verilog HDL File Hexadecimal (Intel-Format) File Memory Initialization File Memory Initialization File Nemory Initialization File In-System Sources and Probes File Logic Analyzer Interface File SignalTap II Logic Analyzer File Gother Files AHDL Include File Block Symbol File Chain Description File Synopsys Design Constraints File Text File
OK Cancel Help

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**).





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_first_fpga db greybox_tmp incremental_db counter_bus_m counter_bus_m counter_bus_m pll_v pll_bb.v simple_counter.	ux.v ux_bb.v	← 🗈 💣 🖩	<b>? </b> ⊠ ∎-
My Documents				
My Network Places	File name: Save as type:	simple_counter.v Verilog HDL Files (*.v *.vlg *.verilog	• ) •	Save Cancel
		Add file to current project		

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;
      rea
9
10
      always @ (posedge CLOCK 5)
                                                // on positive clock edge
11
    Ξ
         begin
12
             counter_out <= counter_out + 1;</pre>
                                                // 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.

ALERA.

- 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		X
Libraries:	simple_counter CLOCK_6 counter_out[31.0]	
Name:	inst	]
simple_counter		
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**.



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**.



🐇 IegaViza	ard Plug-In Manager [page 1]	×
	The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions. Which action do you want to perform? © <u>Greate a new custom megafunction variation</u> © Edit an existing custom megafunction variation © Cogy an existing custom megafunction variation Copyright (C) 1991-2010 Altera Corporation	
	Cancel     < Back     Next >     Einish	

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.



🖑 MegaVizard Plug-In Manager [paş	ge 2a]	×
Which megafunction would you like to customize? Select a megafunction from the list below ALTDDIO_IN ALTDDIO_OUT ALTDUL ALTDQ ALTDQS ALTDQ_DQS ALTDQ_DQS2 v10.1 ALTGX ALTGX_RECONFIG ALTIOBUF ALTIVDS_TX ALTIVDS_TX ALTIVDS_TX ALTIVDS_TX ALTIVDS_TX ALTPLL_RECONFIG ALTREMOTE_UPDATE ALTREMOTE_UPDATE ALTREMOTE_UPDATE ALTREMOTE_UPDATE ALTREMOTE_UPDATE ALTREMOTE_UPDATE ALTREMOTE_UPDATE	Which device family will you be using? Which type of output file do you want to AHDL YHDL Verilog HDL What name do you want for the output file: [Ny_design/my_first_fpga/pll.v] Return to this page for another creat Note: To compile a project successfully in files must be in the project directory, in a the Options dialog box (Tools menu), or a of the Settings dialog box (Assignments n Your current user library directories are:	ile?  e operation the Quartus II software, your design library specified in the Libraries page of a library specified in the Libraries page
	Cancel < B	ack Next > Finish

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.



LegaVizard Plug-In Manager [page 3	of 14] 🔹 💽 🔀
	About Documentation
Parameter         2 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: Cyclone IV E
	Create an 'fbin' input for an external feedback (External Feedback Mode) Which output dock will be compensated for?
	Cancel     < Back     Next >     Einish

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.



▲ IegaVizard Plug-In Manager [page	a 12 of 12]	28
ALTPLL		<u>About</u> <u>D</u> ocumentation
1 Parameter 2 PLL 3 Output	4 EDA 5 Summary	
Settings Reconfiguration Clocks		
Incik0       Incik0 frequency: 50.000 MHz         Operation Mode:       Normal         Clk       Ratio Ph (dg) DC (%)         c0       1/10         Cyclone IV E	checkmark indicates an opt maintained in subsequent N	o generate. A gray checkmark indicates a file that is automatically generated, and a green lonal file. Click Finish to generate the selected files. The state of each checkbox is degaWizard Plug-In Manager sessions. anager creates the selected files in the following directory: a\ Description Variation file PinPlanner ports PPF file AHDL Include 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.







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	
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 DE0-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 DE0-Nano board. The following steps describe how to add a multiplexer to the schematic.

# ADERA.

- 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.

🐇 MegaWizard Plug-In Manager [page	e 2a]		×
Select a megafunction from the list below	Which device family will you be using? Which type of output file do you want to <u>A</u> HDL <u>V</u> HDL Verilog <u>H</u> DL What name do you want for the <u>output f</u> E:/My_design/my_first_fpga/counter_bu Return to this page for another creat Note: To compile a project successfully in files must be in the project directory, in a the Options dialog box (Tools menu), or a of the Settings dialog box (Assignments of Your current user library directories are:	file? us_mux.v te operation n the Quartus II software, your design a library specified in the Libraries page of a library specified in the Libraries page menu).	
	Cancel	Back Next > Finish	

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**.



LegaVizard Plug-In Manager [pag	e 3 of 5]
LPM_MUX	<u>About</u> <u>D</u> ocumentation
1 Parameter Settings 2 EDA 3 Summary	
counter_bus_mux         sel       result[3.0]         data1x[3.0]       result[3.0]	Currently selected device family: Cyclone IV E   Match project/default   How many 'data' inputs do you want?   Image: Comparison of the transformer of
1 lpm_mux	Cancel < <u>B</u> ack <u>N</u> ext > <u>E</u> inish

Figure 6-34 lpm\_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**.

	pll		· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
				· · · · · · · · · · · · · · · · · · ·
		sin	nple_counter	
inc				.counter[310]
	inclk0 frequency: 50.000 MHz		CLOCK_5 counter_out[310]	<b>X</b> • • •
	Operation Mode: Normal			
	operation mode. Horman			
	Clk Ratio Ph (dg) DC (%)			
	c0 1/10 0.00 50.00			
	00 1110 0.00 00.00	in	st	
	st1 Cyclone	IVE		
	· · · · · · · · · · · · · · · · · · ·	counter hug r		
		counter_bus_r	nux	
	data0	10 21-01	result[30]	
	ualau	Monol	resultoriol	
		x[30]		
	uala	x[00]		
	sel	1		
		1		
		1		
		1		
		L		
	inst2			

Figure 6-36 Place the lpm\_mux symbol

# ADERA.

- 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**.
- 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				· ·	· ·		 • •	· · · ·		• •	• •	• •	· · · ·	•••	
counter[2421]	data0x[30]		result[30]	res	ult[3	i0]		 	• •			• •	• •	LEC	)[30	;;; )]
	data1x[30]							 • •	ł							
	sel							 			: :	· · · ·		· · ·	•••	· · · ·
· · · · · · · · · · · · · · · · · · ·					• •		• •	 • •	· ·			• •	· ·	· · · ·		· · · ·
· · · · · · · · · · · · · · · · · · ·	in st2							 • •	· · ·			• •	· ·	· · ·	· · ·	· · · ·
· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · ·			] · · · · ·	 	· ·	· · ·	 · · · ·	· · · ·	· · · ·	· · · · ·	· · ·	· · · ·	· · · ·		· · · ·

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:/Ey		st_fpga <b>/my</b> _f	irst_fpga - <b>my</b>	_first_fpga				
Eile	E	dit <u>V</u> iew P <u>r</u> ocessing	<u>T</u> ools <u>W</u> indow							
	Gro	ups		5 ×			log View - Wire Bond ne IV E - EPtCE22F17C6			
۶⁄ 	Nam	ned: * 🔽 🗸								
<b>:</b>		Node Name	Direction	Location			X808308X2838			
	÷.	KEY[00]	Input Group							
₿.		LED[30]	Output Group							
Ð,	<									
જી				>		Aco				
	<						<u> </u>			
X	×	Named: * 🛛 🗙	Edit: 🗙 🗸					Filter: Pins: a	il 🕑	
ŝ	8	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved		
CH.		CLOCK_50	Input				2.5 V (default)			
_		KEY[0]	Input				2.5 V (default)			
		LED[3]	Output				2.5 V (default)			
₩Ľ.		LED[2]	Output				2.5 V (default)			
		LED[1]	Output				2.5 V (default)			
E		LED[0]	Output				2.5 V (default)			
淵		< <new node="">&gt;</new>								
3										
<b>–</b>	All Pins									
×	Alle									
								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 DE0-Nano board.

8				
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**.



😻 I	lin	Planner - E:/Ey_	_design/my_firs	t_fpga <b>/my</b> _fi	rst_fpga - my	_first_fpga				
<u>F</u> ile	E	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 - EPtoE222F1706			
	Named: *					•				
•••••				Locati 🔨						
- <u></u>	IED[2] Output PIN B13									
<b>₽</b>	LED[1] Output PIN_A13									
Ð,				A15						
જી	The second secon									
E	×	Named: * 💉 🗙	Edit: 🗙 🗸		Filter: Pins: all			II 🔽		
ю	8	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved		
$\odot$		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)			
<b>28</b>		< <new node="">&gt;</new>		L						
2	w.									
×	All Pins									
Ť								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/my_first_fpga - my_first_fpga							
<u>File Edit View Project Assignments Processing To</u>	ols <u>W</u> indow <u>H</u> elp							
🕴 🗅 🗳 🔛 🎒 🎒 👗 🗈 🛍 🗠 🖙 🏭 my_first_	fpga 💽 🔀 🔮 🖁	🖉 🛞 💷 🕨 🧖 🐚	🝈 🕘 🐌 🖻 👗 🥥 »					
Project Navigator 🗗 🗙	first_fpga.bdf 🗵 🛛 🍄 simple_counter.v 🗵	🔄 Compilation Report 🗵	🕸 my_first_fpga.sdc 🗵 🔳 🕨					
Entity       Cyclone IV E: EP4CE22F17C6		Quartus II Version 10 Revision Name my Top-level Entity Name my	progress - Fri Jan 14 17:42:11 2011 1 Build 153 11/29/2010 SJ Full Versior first_fpga clone IV E					
Company Compan								
		<	· · · · · · · · · · · · · · · · · · ·					
Type Message J Info: 5 registers lost all their fanouts during netlist optimizations. The first 5 are displayed below J Info: Generating hard_block partition "hard_block:auto_generated_inst" J Info: Implemented 38 device resources after synthesis - the final resource count might be different J Info: Quartus II Analysis & Synthesis was successful. 0 errors, 0 warnings								
System (Processing (27) Extra Info (Info (27)) Message: 0 of 107	Warning / Critical Warning / Error / Suppres	ssed / Flag /						
Message: 0 of 107			V 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 Summary	
Quartus II Version       10.1         Revision Name       my_1         Top-level Entity Name       my_1         Family       Cyclo         Device       EP4C         Timing Models       Final         Total logic elements       31/         Dedicated logic registers       27 /         Total registers       27         Total pins       6 / 1         Total wirtual pins       0         Total memory bits       0 / 6         Embedded Multiplier 9-bit elements       0 / 1	essful - Fri Jan 14 17:42:39 2011 Build 153 11/29/2010 SJ Full Version rst_fpga ne IV E E22F17C6 22,320 ( < 1 % ) 22,320 ( < 1 % ) 22,320 ( < 1 % ) 34 ( 4 % ) 08,256 ( 0 % ) 32 ( 0 % ) ( 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 DE0-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**.



🔖 Programmer -	E:/My_design/my_f	irst_fpga <b>/my</b> _f:	irst_fpga ·	- my_first_	fpga - [m	y 📘			
<u>File E</u> dit <u>V</u> iew Pr	ocessing <u>T</u> ools <u>W</u> indow								
Hardware Setup	. USB-Blaster [USB-0]	Mode: JTAG		V Progr	ess:				
Enable real-time ISP to allow background programming (for MAX II and MAX V devices)									
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check		
Stop	my_first_fpga.sof	EP4CE22F17	0013888D	FFFFFFF	<b>V</b>				
Auto Detect									
X Delete									
Add File									
👺 Change File	<			)			>		
Save File							^		
Add Device									
🕐 Up							≡		
Down	→ <b>→</b>								
	EP4CE22F								
	TDO	.,					<u>~</u>		
							1.8		

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**.



🖏 Hardware Setup								
Hardware Settings       JTAG Settings         Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.								
	Currently selected hardware: Available hardware items	USB-Blaster [USB-0] No Hardware USB-Blaster [USB-0]						
	Hardware	Server	Port	Add Hardware				
	USB-Blaster	Local	USB-0	Remove Hardware				
l				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 I	Processing Tools Windov	Ŷ						
🔔 Hardware Setup	USB-Blaster [USB-0]	Mode: JTAG		Y Progre	ss: 100%	(Successful)		
Enable real-time ISP to allow background programming (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	<b>V</b>			
Auto Detect								
X Delete								
Add File								
👺 Change File	<						>	
Save File								
Add Device								
The Up	and a						_	
Down								
							~	
							1.11	

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



Package:	Any Any			
Pin <u>c</u> ount:	Any			
Speed grade:	: Any 💙			
Show adv	vanced devices			
- HaroCopy	y compatible only			
Device and Die	Onting			
	Options			
I/Os Memory Bits	s Embedded multiplier 9-bit elements			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
608256	132			
	Image: Show adv         HardCop           Device and Pir         Device and Pir           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256           608256         608256			

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:	
Category: General Configuration Programming Files Unused Pins Dual-Purpose Pins Capacitive Loading Board Trace Model I/O Timing Voltage Pin Placement Error Detection CRC CVPCIe Settings	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         Image: Specify device-wide options for reserve individual dual-purpose Pins tab. To reserve other pins individually, use the Assignment Editor.       Image: Specify device-wide options for reserve individual dual-purpose Pins tab. To reserve other pins individually, use the Assignment Editor.         Reserve all unused pins:       As input tri-stated         Description:       Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, 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.
	<u>R</u> eset OK Cancel Help

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 DE0-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**.



<b>%</b> (	Juar	tus	II				
Fil	e <u>E</u>	dit	<u>V</u> iew	<u>P</u> roject	<u>A</u> ssignments	P <u>r</u> o	
$\square$	<u>N</u> ew				Ctrl+N		
<b>2</b>	<u>Open</u> .				Ctrl+0		
	<u>C</u> los(	2			Ctrl+F4		
2	New H	Proje	ct <u>W</u> iz	ard			
ŝ	Open	P <u>r</u> oj	ect		Ctrl+J		
	Save Projec <u>t</u>						
	Clos	e Pro	ject				
	Save				Ctrl+S		
	Save	<u>A</u> s					
ø	Save	A11			Ctrl+Shift	+S	
	<u>F</u> ile	Prop	erties				
	Creat	te <u>/</u> '	Update			•	
	Expoi	ct <u>.</u>					
	Conve	ert P	rogram	ming Files	5		
	Page	Setu	р				
à	Print	t Pre	<u>v</u> iew				
6	Print	t			Ctrl+P		
	Recei	at F <u>i</u>	les			٠	
	Recei	at Pr	o <u>i</u> ects			•	
	E <u>x</u> it				Alt+F4		

Figure 7-1 Start to Create a New Project

🕊 New Project Vizard	
Directory, Name, Top-Level Entity [page 1 of 5]	
What is the working directory for this project?	
D: \Home \User \Desktop	
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.	
Use Existing Project Settings	
< <u>Back</u> <u>Next</u> > <u>Einish</u> Cancel <u>E</u>	<u>t</u> elp

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**.



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
< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel <u>H</u> elp

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

ile name:					<u>A</u> dd
Fi⊥e Name	Type   Library	Design Entry/Synthesis Too	1 HDL Version		Add Alj Remove Up Down Properties
pecify the pa	ath names of any no	n-default libraries. Uger Libraries			

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 DE0-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**.



~
~
~
9-bit elements 📥
<u>×</u>

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 Wizard EDA Tool Settings [page 4 of 5] Specify the other EDA tools used with the Quartus II software to develop your project. EDA tools: Tool Name Format(s) Run Tool Automatically Tool Type Design Entry/Syn… <None> <None> Run this tool automatically to synthesize the current design Simulation <None> ✓ <None> Run gate-level simulation automatically after compilation Timing Analysis <None> None> Run this tool automatically after compilation Formal Verificat… <None> Board-Level Timing <None> ~ Symbol <None> ~ Signal Integrity <None> \* Boundary Scan ~ <None> < <u>B</u>ack Next > Einish Cancel Help

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 setting:	52
Project directory: Project name: Top-level design entity: Number of files added:	D:/myfirst_niosii myfirst_niosii myfirst_niosii 0
Number of user libraries added: Device assignments:	0
Family name:	Cydone IV E
Device: EDA tools:	EP4CE22F17C6
Design entry/synthesis:	<none>(<none>)</none></none>
Simulation: Timing analysis:	<none> (<none>) <none> (<none>)</none></none></none></none>
Operating conditions:	
VCCINT voltage: Junction temperature range:	1.2V 0-85 ℃
	< Back Next > Einish Cance Help

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>Assignments</u> Processing <u>T</u> ools <u>W</u> indow <u>H</u> elp	
i myfirst_niosii 💽 🔀 😤 🖉 🥙 💷 🕨 🦻 🐌 😫 🕹 🔘 🛡	
Project Navigator	2° T
A Hierarchy B Files d <sup>®</sup> Design Units Status d <sup>®</sup> ×	<b>7</b> I.
Module % Progress © Time	/ Quartus II mation
	entation
x Ø	
	Σ
Warning       Critical Warning       Error       Suppressed       Flag         Wessage:	
Message:  Image: 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**.



Figure 7-9 SOPC Builder Menu



😃 Create New System - Alt		ed.sopc (D:∖ <b>my</b> fi	rst_niosii\unna	med. sopc)				
File Edit Module System View To	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					Remove			
Avalon Verification Suite								
Bridges and Adapters     Immediate Protocols		9						
	Use C Mc 😃 Create New	ystem.	CI	ock Base	End			
Memories and Memory Contro     Merlin Components	System Name: unna	amed						
	Target HDL: () Ver	rilog						
PLL     Processor Additions								
	O ∨H	DL						
SLS     University Program	Info: Specify a	new system name.						
🕀 Video and Image Processing 🞽			-					
		OK Cancel						
×	<				>			
New Edit Add	Remove Edit		ddress Map <u>F</u> ilte	ers				
	Exit 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 ◯ VHDL	
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> ools <u>H</u> elp							
System Contents System Generation							
Component Library	Target	Clock Settings					
Project	Device Family: Cyclone IV E	Name	Source	MHz	Add		
Wew component		clk_0	External	50.0	Remove		
Avalon Verification Suite							
⊡…bridges and Adapters     ⊡…interface Protocols							
Legacy Components	Use C Module	Description	Clock	Base	End		
Memories and Memory Contro     Merlin Components							
Peripherals							
i mell							
Processor Additions     Processors							
University Program							
Video and Image Processing							
Q. X	<				>		
New Edit Add	Remove Edit	Addre	ess Map <u>F</u> ilters	Filter: Default			
				There bereat			
	Exit 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>	Tools <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			
Library					Remove		
Avalon Verification Suite							
⊕Bridges and Adapters     ⊕Interface Protocols							
Legacy Components	Use C Module	Description	Clock	Base	End		
⊞Memories and Memory Contro							
Peripherals							
±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



u Nios II Proces	sor - cpu_0			X
Nios	II Processor			About Documentation
Parameter Settings				
	es and Memory Interfaces	Advanced Features MI	MU and MPU Settings > Л	TAG 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 <sub>system:</sub> 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 Predictio	n
Performance at 50.0 MHz		Up to 32 DMIPS	Up to 57 DMIPS	<b>_</b>
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs	
Hardware Multiply: Embe	ory:	Hardware Divide		
Exception Vector: Memo	ry:	✓ Offset: 0x20		
Include MMU     Only include the MMU wh	en using an operating system ti	nat explicitly supports an MMU		
Fast TLB Miss Exception			ffset: 0x0	
Include MPU				
🔥 Warning: Reset vector	and Exception vector cannot b	e set until memory devices are	connected to the Nios II proce	ssor
				Cancel ( Back Mext ) Finish

Figure 7-15 Nios II Processor

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



🗳 Altera SOPC Builder				X			
<u>Eile 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 V	ne Source	MHz				
New component		50 External	50.0 Remove				
Library ≟…Avalon Verification Suite							
Bridges and Adapters							
	Use Conn Module	Description Clo	ck Base End				
Memories and Memory Controllers	✓ □ cpu_0	Nios II Processor [clk]					
	instruction_master	Avalon Memory Mapped Master clk_ Avalon Memory Mapped Master [clk]	50 IRQ 0				
	jtag_debug_module	Avalon Memory Mapped Slave [clk]	■ <b>0x00000800</b> 0x00000f	ff			
Processor Additions     Processors							
Nios II Processor							
< >							
Q X	<			>			
New Edit Add	Remove Edit	Address Map Filters	Filter: Default				
To Do: cpu_0: No reset vector has been set vector has been been been been been been been bee	en specified for this CPU. Please parameterize tl	ne CPU to resolve this issue					
	s been specified for this CPU. Please parameter						
Warning: cpu_0: Reset vector and Exc	ception vector cannot be set until memory device	es are connected to the Nios II processor					
	Exit Help I Pr	ev Next Denerate					

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>E</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	Target	Clock Settings						
Project	Device Family: Cyclone IV E	Name	Source	MHz	Add			
New component		clk_50	External	50.0	Remove			
Library					Kennove			
Interface Protocols     Legacy Components	Use Conn Module	Description	Clock	Base	End			
Memories and Memory Controllers	✓ □ cpu_0	Nice II Droces	ender Ppr [clk]	Dusc	Eng			
Merlin Components	instru Co	nnections	Mapped Master clk_50					
⊕Peripherals     ⊕PLL		int	Mapped Master [clk]	IRQ O				
Processor Additions	jtag_( Sh	ow Connected	Mapped Slave [clk]	■ 0x0000800	0x00000fff			
Processors	✓ De	fault						
•••• Nios Il Processor	All	l						
ia.⊷SLS ia.⊷University Program	Cic	ocks						
Wideo and Image Processing	Av	alon-MM						
	Av	alon-ST						
	Ed	it Ctrl+E						
Q X ■	Re	<u>n</u> ame Ctrl+R			>			
	Re	move			<u> </u>			
New Edit Add	Remove Edit De	tails	Address Map <u>Filters</u>	Filter: Default				
To Do: cpu_0: No reset vector has been	Sh	ow Arbitration	this issue					
To Do: cpu_0: No exception vector has I		ck Base Address Ctrl+L	olve this issue					
A Warning: cpu_0: Reset vector and Exce	ption vector cannot be set u		to the Nios II processor					
		pand All						
	<u></u> 0	llapse All	_					
		t Color						
	Exit Help	Prev     Next	Generate					

Figure 7-17 Rename the CPU (1)

😃 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> o	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	Add
New component		clk_50 Ex	xternal 5	50.0	Remove
Library					Remove
⊕Bridges and Adapters					
	Use Conn Module	Description	Clock	Base	End
Memories and Memory Controllers		Nios II Processor	[Clk]	Dusc	LING
Merlin Components	instruction_mas				
Peripherals     Deripherals	data_master	Avalon Memory Mapped		IRQ 0	
Processor Additions	jtag_debug_mod	dule Avalon Memory Mapped	I Slave [clk]	=° 0x0000800	0x00000111
Processors     Nios Il Processor					
University Program					
Wideo and Image Processing					
Q X	<				>
New Edit Add	Remove Edit	Address	Map Filters F	ilter: Default	
To Do: cpu: No reset vector has been	specified for this CPU. Please parameter	ize the CPU to resolve this issue			
To Do: cpu: No exception vector has b			sue		
A Warning: cpu: Reset vector and Excep	tion vector cannot be set until memory d	evices are connected to the Nios	Il processor		
	Exit 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	<u>T</u> o
System Contents System Generation	
Component Library	
Project	
Library	
"Bridges and Adapters     "Interface Protocols	
ASI ≣ Ethernet	
High Speed     High Interlaken	
⊡-Serial	
Avalon-ST JTAG     Avalon-ST Serial	
ITAG UART	
• UART (RS-232 Se	
Legacy Components	

Figure 7-19 Add the JTAG UART component

S JTAG UART -	jtag_uart_0
MogoCore JTAG altera_ava	UART lon_itag_uart
🔻 Block Diagram	
res	ick ■ 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**.



😃 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	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					
Bridges and Adapters					
Interface Protocols	Use Conn Module	Description	Clock	Base	End
€thernet	🔽 🗖 сри	Nios II Processor	[clk]		
High Speed	instruction_mas	ter Avalon Memory Mapp	oed Master clk_50		
⊞Interlaken ⊞PCI	data_master	Avalon Memory Mapp		IRQ	
⊞-PCI ⊕-SDI	jtag_debug_mod			■ 0x0000800	0x00000fff
- Serial	i itag_uart_0	JTAG UART ve Avalon Memory Mapp	ed Slave [clk]	0x0000000	0x00000007
Avalon-ST JTAG	`→  avalon_jtag_sla	ve Avaion memory mapp	Jed Slave  Cik_50	0x0000000	0x0000007
Avalon-ST Serial					
ITAG UART					
SPI (3 Wire Serial					
UART (RS-232 S€ ⊕-Legacy Components					
<					
Q X					
	<	Ш			>
New Edit Add	Remove Edit	Addre	ess Map	Filter: Default	
To Do: cpu: No reset vector has been	specified for this CPU. Please parameter	ize the CPU to resolve this issu	Je		
To Do: cpu: No exception vector has					
	ption vector cannot be set until memory d				
	- · · · · · · · · · · · · · · · · · · ·		-		
	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 E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>1</u>	ools Nios II <u>H</u> elp			
System Contents System Generation				
Component Library	Clo	ck Settings		
Project	Device Family: Cyclone IV E 💙 N	ame Source	MHz	Add
New component	cik	_50 External	50.0	Remove
Library ≟…Avalon Verification Suite				Kennove
Interface Protocols ASI	Use Conn Module	Description	Clock Base	End
Ethernet	🗹 🗆 cpu	Nios II Processor	[clk]	
High Speed     High Speed	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] <b>Ox00000800</b>	
⊡SDI	✓ □ jtag_uart	JTAG UART	[clk]	
Avalon-ST JTAG	avalon_jtag_slave	Avalon Memory Mapped Slave	clk_50 © 0x0000000	0x0000007
Avalon-ST Serial				
JTAG UART     SPI (3 Wire Serial				
◎ UART (RS-232 Sŧ				
⊥egacy Components				
	<	1111		>
New Edit Add	Remove Edit 🗷 🔺	Address Map	Filter: Default	
🗩 To Do: cpu: No reset vector has been	specified for this CPU. Please parameterize th	e CPU to resolve this issue		
·	been specified for this CPU. Please parameteri			
A Warning: cpu: Reset vector and Exce	ption vector cannot be set until memory device	s are connected to the Nios II processor		
	Exit Help	Prev Next Denerate Generate		

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

ADERA.
--------

🗳 On-Chip Memory (R	AM or ROM) - onchip_me	mory2_0	×
MogaCoro On-Chip M attera_avalon_onc	emory (RAM or ROM) hip_memory2	Documentation	
* Block Diagram			^
	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:	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 initiali:	ration file		~
		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**.

	7.
--	----

🗳 On-Chip Memory (R	AM or ROM) - onchip_m	emory2_0	×
MogeCore On-Chip M altera_avalon_onc	emory (RAM or ROM hip_memory2	) <u>D</u> ocumentation	
* Block Diagram			<b>^</b>
	clock  clk1 clock  clk1 clk1 clk1 clk1 clk1 clk1 clk1 clk1		
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	ration file		~
		Cancel Finish	

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.

🖳 Altera SOPC Builder			
<u>F</u> ile <u>E</u> dit <u>M</u> odule <u>System V</u> iew <u>T</u> ools Nios II <u>H</u> elp			
System Contents System Generation			
Component Library	Target	Clock Settings	
Avalon Verification Suite		Name Source	MHz
How Bridges and Adapters	Device Family: Cyclone IV E	cik 50 External	Add 50.0
		CIK_50 EXternal	Remove
ELegacy Components			
Memories and Memory Controllers     DDR2 SDRAM Controller with UniPHY (New!)			
DDR3 SDRAM Controller with UniPHY (New!)	Use Conn Module	Description	Clock Base
QDR II and QDR II+ SRAM Controller with UniPHY	C Cpu	Nios II Processor	[ck]
RLDRAM II Controller with UniPHY     Tartfa Occupation and BIDT Social (New1)	instruction_max	ster Avalon Memory Mapped Master	clk_50
Traffic Generator and BIST Engine (New!)     DMA	data_master	Avalon Memory Mapped Master	[clk]
⊞-Flash	itag_debug_mo	dule Avalon Memory Mapped Slave	[clk] Ox000 [clk]
- On-Chip	avalon itag sla		clk 50 <b>∞ 0x00</b> C
	onchip_memo		[clk1]
Avalon-ST Round Robin Scheduler	<u></u> s1	Avalon Memory Mapped Slave	clk_50 ⊫° 0x000
Avalon-ST Single Clock FIFO			
On-Chip FIFO Memory			
On-Chip Memory (RAM or ROM) ⊕-SDRAM			
۹ ×	<		>
New Edit Add	Remove Edit Z	Address Map	Filters Filter: Default
Error: cpu.instruction master: onchip memory2.s1 cannot be at 0	x2000 (0x0 or 0x10000 are acceptable)	)	
Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000			
To Do: cpu: No reset vector has been specified for this CPU. Please part	ameterize the CPU to resolve this issue		
To Do: cpu: No exception vector has been specified for this CPU. Please	e parameterize the CPU to resolve this is	sue	
E <u>x</u> it H	elp I Prev Next 🕨	Generate	

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					
	es and Memory Interfaces >	Advanced Features > M	MU and MPU Settings	JTAG Debug Modul	e 🔪 Custom Instructions 🔪
Core Nios II					
Select a Nios II core:		(	0		
	○Nios II/e	ONios II/s	●Nios II/f		<u></u>
Nios II Selector Guide Family: Cyclone IV E f <sub>system:</sub> 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	Ē
Performance at 50.0 MHz		Up to 32 DMIPS	Up to 57 DMIPS		_
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs		<u>∽</u>
Hardware Multiply: Embe	dded Multipliers	Hardware Divide			
Reset Vector: Mem Exception Vector: Memo	ory: onchip_memory2	<ul> <li>✓ Offset: 0x0</li> <li>✓ Offset: 0x20</li> </ul>		0x00002000 0x00002020	
Include MMU Only include the MMU wh Fast TLB Miss Exception  Include MPU	en using an operating system ti Vector: Memory:		Offset: 0x0		
				Cancel	< Back Next > Finish

Figure 7-28 Update CPU settings



🖕 Altera SOPC Builder			
<u>File Edit M</u> odule <u>Sy</u> stem <u>V</u> iew <u>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
Bridges and Adapters		clk_50 External	50.0
Hoterface Protocols     Hoterface Components			Remove
Memories and Memory Controllers			
DDR2 SDRAM Controller with UniPHY (New!)			
ODR3 SDRAM Controller with UniPHY (New!)     ODR II and QDR II+ SRAM Controller with UniPHY	Use Conn Module	Description	Clock Base
CODR II and QDR II+ SRAM Controller with UniPHY     RLDRAM II Controller with UniPHY	🔽 🗖 cpu	Nios II Processor	[clk]
<ul> <li>Traffic Generator and BIST Engine (New!)</li> </ul>	instruction_mast	ter Avalon Memory Mapped Master Avalon Memory Mapped Master	clk_50 [clk]
∎∼DMA	jtag_debug_mod	2	[clk] 🗊 Ox000
i ⊕Flash ⊡On-Chip	✓ jtag_uart	JTAG UART	[clk]
Avalon-ST Dual Clock FIFO	avalon_jtag_slav		clk_50 ⊫ 0x000
····      Avalon-ST Multi-Channel Shared Memory FIFO	s1	y2 On-Chip Memory (RAM or ROM) Avalon Memory Mapped Slave	[clk1] clk_50 <b>□ 0</b> x000
Avalon-ST Round Robin Scheduler			
Avalon-ST Single Clock FIFO     On-Chip FIFO Memory			
On-Chip Memory (RAM or ROM)			
Q X			
			<u>&gt;</u>
New Edit Add	Remove	Address Map	Filters Filter: Default
Error: cpu.instruction_master: onchip_memory2.s1 cannot be at 0			
Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 (	(0x0 or 0x10000 are acceptable)		
Exit H	elp I 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.



Figure 7-30 Add PIO



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**.

山辺





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 T</u> ools Nios II <u>H</u> elp								
System Contents System Generation								
		Clock Settings						
Component Library	Target	-						
Project 🔨	Device Family: Cyclone IV E 💙	Name Source	MHz Add					
Library		clk_50 External	50.0 Remove					
Interface Protocols								
	Use Conn Module	Description	Clock Base					
Memories and Memory Controllers    Merlin Components	🗹 🗆 cpu	Nios II Processor	[clk]					
	instruction_mast	ter Avalon Memory Mapped Master Avalon Memory Mapped Master	clk_50 [clk]					
⊕ Debug and Performance	jtag debug mod		[Cik] Ox000					
	✓     □     jtag_uart	JTAG UART	[clk]					
FPGA Peripherals	avalon_jtag_slav		clk_50 <b>₽ 0x00</b> C					
Microcontroller Peripherals     Interval Timer	✓ ☐ onchip_memory		[clk1]					
PIO (Parallel VO)	S1 I pio led	Avalon Memory Mapped Slave PIO (Parallel I/O)	clk_50					
Multiprocessor Coordination	→ s1	Avalon Memory Mapped Slave	clk_50 🖬 Ox 000					
		Available monitory mapped biarte						
Processor Additions								
Processors								
Q	<		>					
New Edit Add	Remove Edit	Address Map	Filters 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 are acceptable)							
Exit Help Prev Next Generate								

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**.

La Altera SOPC Builder								
<u>File Edit M</u> odule <u>System</u> <u>V</u> iew <u>T</u> ools Nios II <u>H</u> elp								
System Contents Auto-Assign Base Addresses								
Auto-Assign IRQs	Target	Clock Settings						
Insert Avalon-ST Adapters		Name Source	MHz					
Project Show Transformed System	Device Family: Cyclone IV E	clk 50 External	MH2 Add					
Library			Remove					
Avalon Verification Suite								
Bridges and Adapters								
Interface Protocols								
Legacy Components    Memories and Memory Controllers	Use Conn Module	Description	Clock Base					
Memories and memory controllers	C cpu	Nios II Processor	[clk]					
- Peripherals	instruction_master	er Avalon Memory Mapped Master Avalon Memory Mapped Master	clk_50 [clk]					
Debug and Performance	itag debug modu		[Cik] [Cik] <b>○ 0±00</b> 0					
⊞Display	Image: stag_debug_mode       Image: stag_uset       Image: stag_uset	JTAG UART						
FPGA Peripherals	avalon itag slav	Avaion Memory Mapped Slave	clk 50 = 0x00C					
Microcontroller Peripherals	Image:	V2 On-Chip Memory (RAM or ROM)	[clk1]					
····   Interval Timer	s1	Avaion Memory Mapped Slave	clk_50 <b>■ 0x00</b> 0					
PIO (Parallel VO)	✓ pio_led	PIO (Parallel I/O)	[clk]					
Multiprocessor Coordination      Horizon	s1	Avalon Memory Mapped Slave	clk_50 <b>≕ 0x00</b> C					
Processors								
	4							
			>					
New Edit Add	Remove	Address Map	Filters Filter: Default					
Error: cpu.instruction master: onchip memory2.s1 cannot be	at 0x2000 (0x0 or 0x10000 are acceptable)							
© Error: cpudata master: onchip.memory2.st cannot be at 0x2000 (0x0 or 0x1000 are acceptable)								
Exit Help A Prev Next D Generate								
		,						

Figure 7-34 Auto-Assign Base Addresses

🐫 Altera SOPC Builder							
<u>File Edit Module System View Tools Nios</u>	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 Library     Avalon Verification Suite     Bridges and Adapters		clk_50 External	50.0	Remove			
	Use Conn Module	Description	Clock B	ase			
Cupper Components  Cupper Components  Cupper Components  Cupper	v     contain     instruction_mass data_master itag_debug_mod       v     istruction_mass data_master itag_uart avalon_itag_sla       v     istruction_mass data_master itag_uart avalon_itag_sla       v     istruction_mass itag_uart avalon_itag_sla       v     istruction_mass itag_uart avalon_itag_sla       v     istruction_mass itag_uart avalon_itag_sla       v     istruction_itag_sla       v     istruction_itag_sla       v     istruction_itag_sla       s1     istruction_itag_sla	Nios II Processor Avalon Memory Mapped Ma Avalon Memory Mapped Ma Jule Avalon Memory Mapped Si JTAG UART ve Avalon Memory Mapped Si	[Cik]           aster         Cik_50           aster         [Cik]           ave         [Cik]           [Cik]         ave           [Cik]         ave           (Cik]         ave	IRQ 0 0x00010800 C 0x00011010 C 0x00008000 C 0x00011000 C			
				<u> </u>			
New Edit Add	Remove Edit	Address Ma	ap <u>Filters</u> Filter: Def	fault			
<ul> <li>Warning: cpu: Custom Instruction components can be edited through the Component Editor.</li> <li>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</li> </ul>							
				>			
	Exit Help I Prev	Next  Generate					

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



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.



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

品記



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**.



Figure 7-40 New Verilog file



Figure 7-41 New Verilog File



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

🍇 Quartus II	I - D:/myf	irst_	_nios	sii/m	yfirst	_nio	sii	- <b>n</b> y	first	_nios	ii								
Eile Edit View	Project As	-		-	ng <u>T</u> ools	s <u>W</u> ir	ndow	<u>H</u> elp											
myfirst niosii					· 🧭 🦁		रक्ते		t ha	হায় ব	es a	n 🔒		<u>en l</u>	Ē				
Project Navigator				=° =°	e x	✓			Verilog 1		<ul> <li>✓</li> <li></li> </ul>				7				
Project Navigator						-	88		-		1 04	- <u>-</u>	6 76			267			
Entity						1		GB L	1 (#	1#   /	<u> </u>	• <b>/</b> • 9	\$ 98	U	2	268	ap/	<del></del>	_ ×
Cyclone IV E:		.6				-	1	1											<u>^</u>
Alierarchy Status Module	E Files d	d <sup>9</sup> Desig	gn Units		E ×														
						<													>
× Type Mess	Jage					<	<u>)</u>												
P Type Mess	age					<										 			
B Type Mess	age	<u>xtra Inf</u>	5 / LI	info_/\				Varning	/\Erro	or /\s	Suppres	ssed /\	Flag /						
5 <u>S</u>		Extra Info			Warning			Varning	/\ Erro	or /\_s	Suppres	ised /\	Flag /				~	Lo	> 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



Open File						? 🗙
Look in:	🗀 myfirst_niosii		•	← €	💣 🎟 -	
My Recent Documents Desktop My Documents My Computer	cpu_v cpu_jtag_debu cpu_jtag_debu cpu_jtag_debu cpu_jtag_debu cpu_mult_cell. cpu_cci_test_ cpu_test_bend cpu_test_bend dcpu_test_bend dcpu_NANO_S0 dcpU_NANO_S0 dcpU_NANO_S0	C db DE0_NANO_SOPC_sim		.v iosii.v emory2.v		
My Network Places	File name: Files of type: Open as:	DE0_NANO_SOPC.v Design Files (*.tdf *.v Add file to current Auto	hd ".vhdl ".v ".v	vlg *.verilog		Open Cancel

Figure 7-44 and Figure 7-45.

```
module myfirst_niosii
(
    CLOCK_50,
    LED
);
              CLOCK_50;
input
output [7:0] LED;
DE0_NANO_SOPC DE0_NANO_SOPC_inst
    (
                                (CLOCK_50),
      .clk 50
      .out_port_from_the_pio_led (LED),
      .reset_n
                                 (1'b1)
    );
```

endmodule



Figure 7-43 Input verilog Text

Open File					? 🗙
Look in:	🗀 myfirst_niosii		•	+ 🗈 💣	<b>Ⅲ</b> ▼
My Recent Documents Desktop My Documents My Computer	🗟 cpu_jtag_debu	_ ug_module_sysclk.v ug_module_tck.v ug_module_wrapper.v v bench.v ch.v DPC.ptf DPC.sopc DPC.sopcinfo DPC.v	itag_uart.	osii.v	
My Network Places	File name: Files of type: Open as:	DE0_NANO_SOPC.v Design Files (*.tdf *.v Add file to current Auto	hd *.vhdl *.v *.v	▼ /lg *.verilog ▼	Open Cancel

Figure 7-44 Open DE0\_NANO\_SOPC.v

正じ

& Quartus II - D:/myfirst_niosii/my	first_niosii – <b>my</b> first_niosii	
<u>File Edit View Project Assignments Processing</u>	g <u>T</u> ools <u>W</u> indow <u>H</u> elp	
: D 🖆 🖬 🗿 🎒 🖁 🛍 🛍 い つ		
myfirst_niosii 💽 🔀 🔮 🖉	I I I I I I I I I I I I I I I I I I I	
Project Navigator 🗗 🗙	🕸 myfirst niosii.v 🛛 🕸 DE0_NANO_SOPC.v 🛛	
Entity	圖  ▲ 编 7)  卓 卓   ▲ 3% 3% 3%   ● ☎   2   题 ∞/   → ℡ ≌	
Cyclone IV E: EP4CE22F17C6	2087	<u>^</u>
myfirst_niosii 🖁 🔠	2088 _module DE0_NANO_SOPC (	
	2089 // 1) global signals:	
	2090 clk_50,	
	2091 reset_n, 2092	
	2093 // the pio led	
	2094 Out port from the pio led	
	2095 - )	
	2096 ;	
	2097	
	2098 output [ 7: 0] out_port_from_the_pio_led;	
	2099 input clk_50;	_
<	2100 input reset_n;	
<	2101	
Hierarchy	2102 wire clk 50 reset n;	>
× Type Message		
B.		
8		>
	Warning / Critical Warning / Error / Suppressed / Flag /	
🖞 Message: 🏠 🖡 Location	и:	Locate
	0%	00:00:00

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						? 🔀
Savejn:	🗀 myfirst_niosii		•	← 🖻	) 💣 🎟 -	
My Recent Documents Desktop My Documents My Computer	🖻 cpu_jtag_debu	_ ig_module_sysclk.v ig_module_tck.v ig_module_wrapper.v / pench.v h.v DPC.v DPC_inst.v				
T Idees	File <u>n</u> ame:	myfirst_niosii.v			•	<u>S</u> ave
	Save as <u>t</u> ype:	Verilog HDL Files (*.v *.vlg *.v	verilog)		•	Cancel
		Add file to current project				

Figure 7-46 Save the Verilog file



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="">&gt;</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.

terasIC Terasic DE0-Nano User Manual

- 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.

Ð	Hardware Setup			$\overline{\mathbf{X}}$			
	Hardware Settings       JTAG Settings         Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.         Currently selected hardware:       USB-Blaster [USB-0]						
	Available hardware items	No Hardware USB-Blaster [USB-0]					
	Hardware	Server	Port	Add Hardware			
	USB-Blaster	Local	USB-0	Remove Hardware			
l				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**.



When blolect							
Nios II C/C++ Application Click Finish to create application with a default system library as D:\myfirst_niosii\Software\hello_world_0							
	ftware 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 Mext > 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							
<u>F</u> ile <u>E</u> dit Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>P</u> r	<u>P</u> roject Tools <u>R</u> un <u>W</u> indow	Help					
📬 • 🖫 👜   🗟   💣 • 😂 • 🖻 •	<b>♂</b> •   <b>\$</b> • <b>Q</b> • <b>Q</b> •	🥭 🖋 🛛 🔂 🗸	·   ∳  • \}	• * + + +		😭 🎆 Ni os	II C/C++
💽 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 'st * designs. It ru: * device in your * The memory foot * using the stand * * For a reduced : *</pre>	rints 'Hello tandard', 'fu ns with or wi system's har tprint of thi dard reference footprint ver memory footpr orld" templat	All_featur thout the rdware. is hosted be design. rsion of t rint for a ce.	II' to the STDOU ed', 'fast', and MicroC/OS-II RTO application is ~6 his template, and given applicatio <b>Path</b>	'low_cost' S and requ 9 kbytes b an explan	example ires a STDOUT y default ation of how	
		Writable	Smart Inser	•t 1:1			

Figure 7-566 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  $\mu$ 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 7-577 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 DE0-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 \leq 2000000)
{
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		
File       Edit       Refactor       Mavigate       Segr         Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector         Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector       Image: Segret in the sector <th></th> <th>/C++</th>		/C++
Nios II C/C++ P 🛛 🦳 🗖	🖻 hello_world.c 🛛 🖻 system.h 🛛	
system. stf	<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_D_TEST_BENCH_WIRING 0 #define PIO_LED_D_TEST_BENCH_WIRING 0 #define PIO_LED_HAS_TRI 0 #define PIO_LED_HAS_TRI 0 #define PIO_LED_HAS_TN 0 #define PIO_LED_CAPTURE 0 #define PIO_LED_CAPTURE 0 #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 \$0000000 #define ALT_MODULE_CLASS_pio_led altera_avalon_pio </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 7-58**.



Figure 7-58 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.

terasIC Terasic DE0-Nano User Manual

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 7-59. 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 7-59**.



Figure 7-59 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**

# **DE0-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.





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 DE0-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  $0\sim3.3V$ . 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	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**.

NogoCoro altera_avalo	arallel I/O)	Documentation
* Basic Settings		
Width (1-32 bits):	2	
Direction:	OBidir	
	💿 Input	
	◯ InOut	
	🔿 Output	[
Output Port Reset Valu	e: 0×00000000000000000000000000000000000	
• Output Register	t setting/clearing	
Edge capture regist	ter	
Synchronously ca	pture	
Edge Type:	FALLING 🔽	
	FALLING	
Enable bit-clearing		
<ul> <li>Enable bit-clearing</li> <li>Interrupt</li> </ul>		

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.

	e)		
D7 D6 D5 D4 D3 D2	)2	D1	

SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range
-----------	-----	------------	---	----------	---------	-------

Figure 8-8 DATA\_FORMAT Register



The data format is configured as 10 bits, right-justify, +/- 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)
negister 0x50—intr_	SOURCE (Read Only)

	negister experiment _seence (neurony)						
	D7	D6	D5	D4			
q	DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity			
	D3	D2	D1	D0			
	Inactivity	FREE_FALL	Watermark	Overrun			

#### Figure 8-9 Register 0x30

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

EPCS16 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**

terasic Terasic DE0-Nano User Manual



- 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 DE0-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	
(You may add a startup script: c:/altera/10.0/nios2eds/user.bashrc) Using cable 'USB-Blaster [USB-0]'', device 1, instance 0x00 Resetting and pausing target processor: OK Initializing CPU cache (if present) OK	
Downloaded 84KB in 1.4s (60.0KB/s) Verified OK Starting processor at address 0x020001C8 nios2-terminal: connected to hardware target using JTAG UART 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: - [0]ACCELEROMETER - [1]ADC - [2]EEPROM - [3]EPCS 	•

• 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]	- 🗆 ×
Select:Demo ACCELEROMETER	
id=E5h	
Monitor Accerometer Value. Press KEYØ or KEY1 to terminal the monitor j	process. 💻
X=-20 mg, Y=-4 mg, Z=872 mg	
X=-32 mg, Y=8 mg, Z=976 mg	
X=-20 mg, Y=8 mg, Z=956 mg	
X=-20 mg, Y=4 mg, Z=980 mg	
X=12 mg, Y=12 mg, Z=1004 mg	
X=36 mg, Y=-8 mg, Z=972 mg	
X=-32 mg, Y=8 mg, Z=968 mg	
X=-28 mg, Y=8 mg, Z=980 mg	<b>_</b>

• 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.



• 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	<b>▲</b>
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 DE0-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 DE0-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.

•





# 9.1 Appendix

#### Programming the Serial Configuration Device

This appendix 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 Ouartus 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, **EPCS16**.
- 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/Demonst	ration/DE_NAN				
<u>F</u> ile <u>T</u> ools <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 Conversion Setup Data	Sav	ve Conversion Setup				
Output programming file						
Programming file type: JTAG Indirect Cor	figuration File (.jic)	•	~			
Options Configuration devi	ce: EPCS16 Mode:	Active Serial				
File name: output_file.jic						
Advanced Remote/Local upda	ate difference file: NONE		~			
Input files to convert						
File/Data area	Properties	Start Address Add Hex Dat				
Flash Loader			5			
SOF Data	Page_0	<auto> Add Sof Pag</auto>	e			
		Add File				
		Remove				
		Up				
		Down				
		Properties				
	Gene	rate Close Help				
			1			

### Figure 9-1 Convert Programming Files Dialog Box



🖿 Convert Progra	mming File -	D:/Home/User/Des	ktop/DE_NANO_C4/	jic_photo/new/DE	J 🗖 🗖 🔀		
<u>F</u> ile <u>T</u> ools <u>W</u> indow							
Specify the input files to co You can also import input f future use. Conversion setup files							
Ope	n Conversion Setup E	Data	Sa	ve Conversion Setup			
Output programming file							
Programming file type:	JTAG Indirect Config	uration File (.jic)			¥		
Options	Configuration device:	EPCS16	✓ Mode:	Active Serial	~		
File <u>n</u> ame:	output_file.jic						
Advanced	Remote/ <u>L</u> ocal update	difference file: NON	NE		~		
[	Memory Map File						
Input files to convert							
File/Data	area	Properties	Start Address		Add He <u>x</u> Data		
Flash Loader		Page_0 EP4CE22F17	<auto></auto>		Add Sof Page Add Devige Remove Up Down Properties		
			(	Generate 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.



Device family         AFEX20K         Arria GX         Arria II GZ         Cyclone II         Cyclone III         Cyclone IIV E         Cyclone IV K         Cyclone IV GX         Stratix GX         Stratix II         Stratix II         Stratix II         Stratix IV             OK	🕸 Select Devices		
	APEX20K Arria GX Arria II GX Arria II GZ Cyclone Cyclone II Cyclone III Cyclone III LS Cyclone IV E Cyclone IV E Cyclone IV GX MAX II MAX V Stratix Stratix GX Stratix II Stratix II Stratix II GX Stratix II GX Stratix III	□       EP4CE 10       N         □       EP4CE 15       Im         ☑       EP4CE 22       Ex         □       EP4CE 30       Ex         □       EP4CE 40       Ex         □       EP4CE 55       EP4CE 6       R         □       EP4CE 75       Und	port port dit emove check All

Figure 9-3 Select Devices Page



🖿 Convert Progra	amming File -	D:/Home/User/D	esktop/DE_	NANO_C4/	jic_photo/new/DF	:0 🔳 🗖 🔀		
<u>F</u> ile <u>T</u> ools <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.							
	en Conversion Setup	Data		Sav	e Conversion Setup	]		
Output programming file	e							
Programming file type:	JTAG Indirect Config	guration File (.jic)				<b>~</b>		
Options	Config <u>u</u> ration device	EPCS16	~	Mode:	Active Serial	~		
File <u>n</u> ame:	output_file.jic							
Advanced	Remote/Local update	e difference file:	IONE			<u> </u>		
	Memory Map File							
_Input files to convert								
File/Dat	a area	Properties	Sta	art Address		Add Hex Data		
EP4CE22		Page_0	<aut< td=""><td>22</td><td>•</td><td>Add <u>S</u>of Page</td></aut<>	22	•	Add <u>S</u> of Page		
DE0_NANO.sc	of	EP4CE22F17				Add Device		
						Remove		
						Up		
						Down		
						Properties		
				6	Constant Constant			
				L	<u>G</u> enerate Close	Help		

Figure 9-4 Convert Programming Files Page



Convert Progra	amming File - D:/Home/User/Desk	top/DE_WANO_C4/jic_photo/new/DE0	🗖 🗖 🗙				
<u>File Tools Window</u>							
Specify the input files to a You can also import input future use. Conversion setup files	convert and the type of programming file to genera file information from other files and save the conve	ite. ersion setup information created here for					
Ope	Open Conversion Setup Data Save Conversion Setup						
Output programming file	ej						
Programming file type:	JTAG Indirect Configuration File (.jic)		~				
Options	Configuration device: EPCS16	Mode: Active Serial	~				
File <u>n</u> ame:	output_file.jic						
Advanced	Remote/Local update difference file: NONE		~				
Input files to convert File/Data		Cancel Address	Add He <u>x</u> Data Add <u>S</u> of Page				
SOF Data	Page_0 of EP4CE22F17	<auto></auto>	Add File Remove				
			Down Properties				
		<u>G</u> enerate 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**).



	D:/CD/DE_Nano/D		/DE_NANO_det	ault/DE_NA	NO – DE_NA	<b>NNO - [</b>	DE_NANO	.cdf]*		
<u>File E</u> dit <u>V</u> iew Pr	ocessing <u>T</u> ools <u>W</u> ind	ow								
Hardware Setup	. USB-Blaster [USB-0]		Ν	lode: JTAG		*	Progress:			
Enable real-time ISF	P to allow background pro	gramming (for MAX I	I and MAX V device	s)			·			
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase
	tory default enhanced		0017F1FA	FFFFFFF						
Stop	output_file.jic	EPCS16	1CF4D117							2
Auto Detect										
X Delete										
Add File										
躇 Change File	<									>
Save File										~
										1
Add Device	7.00000									
1 Up	EPC	516								
	2									
- Down										
		2022								~

Figure 9-7 Erasing setting in Quartus II programmer window

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

# Chapter 10



# **10.1 Revision History**

Version	Change Log	
V1.0	Initial Version (Preliminary)	
V1.3	Add Table 3-1,3-2 and 3-3	
V1.4	Modify Digital Accelerometer Description on page 31	

# **10.2 Copyright Statement**

Copyright © 2011 Terasic Technologies. All rights reserved.

Always visit DE0-Nano webpage for new applications.

We will be continuing providing interesting examples and labs on our DE0-Nano webpage. Please visit <u>www.altera.com</u> or <u>DE0-Nano.terasic.com</u> for more information.