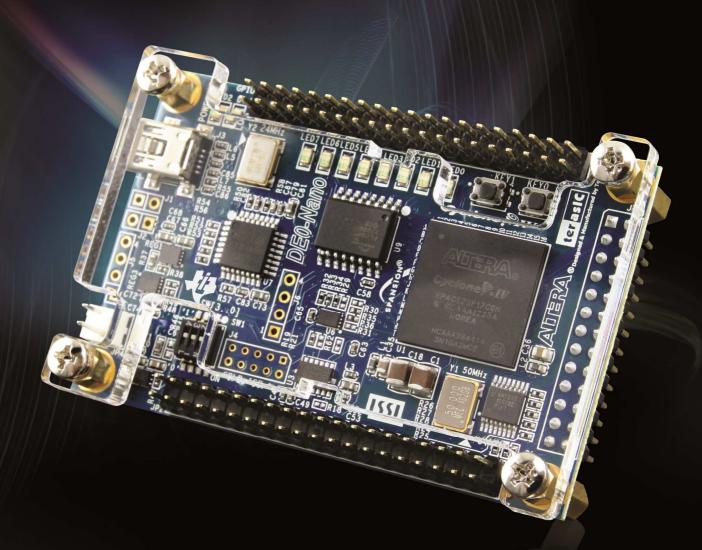
DEO-Nano User Manual World Leading FPGA Based Products and Design Services





Copyright © 2003-2013 Terasic Technologies Inc. All Rights Reserved.





CHAPTER 1	INTRODUCTION	5
1.1 Features		5
1.2 About the KIT	,	7
1.3 Getting Help.		7
CHAPTER 2	DE0-NANO BOARD ARCHITECTURE	8
2.1 Layout and Co	omponents	8
2.2 Block Diagrar	n of the DE0-Nano Board	9
2.3 Power-up the	DE0-Nano Board	
CHAPTER 3	USING THE DE0-NANO BOARD	
3.1 Configuring th	ne Cyclone IV FPGA	
3.2 General User	Input/Output	
3.3 SDRAM Mem	10ry	15
3.4 I2C Serial EE	PROM	16
3.5 Expansion He	aders	17
3.6 A/D Converte	r and 2x13 Header	20
3.7 Digital Accele	rometer	23
3.8 Clock Circuitr	у	23
3.9 Power Supply		24
CHAPTER 4	DE0-NANO CONTROL PANEL	
4.1 Control Panel	Setup	
4.2 Controlling th	e LEDs	
4.3 Switches and	Pushbuttons	
4.4 Memory Cont	roller	29
4.5 Digital Accele	rometer	
4.6 ADC		
4.7 Overall Struct	ure of the DE0-Nano Control Panel	
CHAPTER 5	DE0-NANO SYSTEM BUILDER	
5.1 Introduction		

5.2 General Design I	Flow	
5.3 Using DE0-Nand	System Builder	
CHAPTER 6	TUTORIAL: CREATING AN FPGA PROJECT	
6.1 Design Flow		40
6.2 Before You Begi	n	41
6.3 What You Will L	earn	45
6.4 Assign The Devi	ce	45
6.5 Creating an FPG	A design	49
6.6 Assign the Pins		71
6.7 Create a Default	TimeQuest SDC File	
6.8 Compile Your De	esign	74
6.9 Program the FPC	GA Device	76
6.10 Verify The Hard	lware	
CHAPTER 7	TUTORIAL: CREATING A NIOS II PROJECT	
7.1 Required Feature	25	
7.2 Creation of Hard	ware Design	
7.3 Download the Ha	ardware Design	
7.4 Create a hello_w	orld Example Project	
7.5 Build and Run th	e Program	
7.6 Edit and Re-Run	the Program	
7.7 Why the LED Bl	inks	
7.8 Debugging the A	pplication	
7.9 Configure System	n Library	
CHAPTER 8	DE0-NANO DEMONSTRATIONS	
8.1 System Requirem	nents	
8.2 Breathing LEDs		
8.3 ADC Reading		
8.4 SOPC Demo		
8.5 G-Sensor		
8.6 SDRAM Test by	Nios II	
CHAPTER 9	APPENDIX	
9.1 Programming the	e Serial Configuration Device	
9.2 EPCS Programm	ing via nios-2-flash-programmer	
terasic D	3 E0-Nano User Manual	www.terasic.com



9.3 Revision History	154
9.4 Copyright Statement	154



Chapter 1



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

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

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

1.1 Features

Figure 1-1 shows a photograph of the DE0-Nano Board.







The key features of the board are listed below:

- Featured device
 - o 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
 - o Spansion EPCS64
- Expansion header
 - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
 - o 32MB SDRAM
 - o 2Kb I2C EEPROM
- General user input/output
 - o 8 green LEDs
 - o 2 debounced pushbuttons
 - o 4-position DIP switch
- G-Sensor
 - 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
 - o 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: +886-3-575-0880
- Email: support@terasic.com
- Altera Corporation
- Email: university@altera.com



Chapter 2

DE0-Nano Board Architecture

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

2.1 Layout and Components

The picture of the 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.

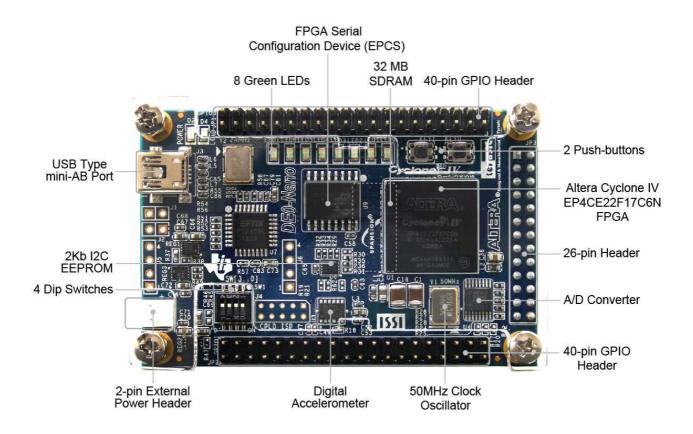


Figure 2-1 The DE0-Nano Board PCB and component diagram (top view)



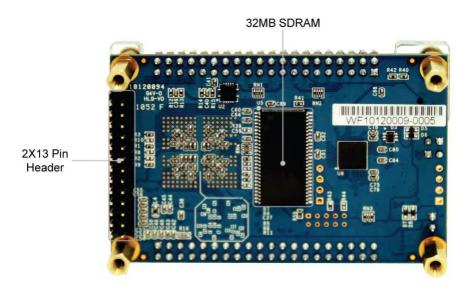


Figure 2-2 The DE0-Nano Board PCB and component diagram (bottom view)

2.2 Block Diagram of the DE0-Nano Board

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

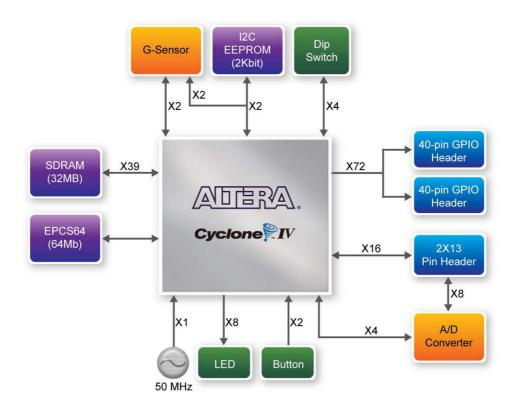


Figure 2-3 Block diagram of DE0-Nano Board

2.3 Power-up the DE0-Nano Board

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

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

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

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





Chapter 3

Using the DE0-Nano Board

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

3.1 Configuring the Cyclone IV FPGA

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

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

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

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

■ Configuring the Spansion EPCS64 device

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

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



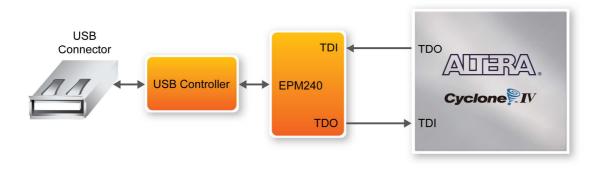


Figure 3-1 Programming a serial configuration device with serial flash loader solution

JTAG Chain on DE0-Nano Board

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

Figure 3-2 illustrates the JTAG configuration setup.





3.2 General User Input/Output

Pushbuttons

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

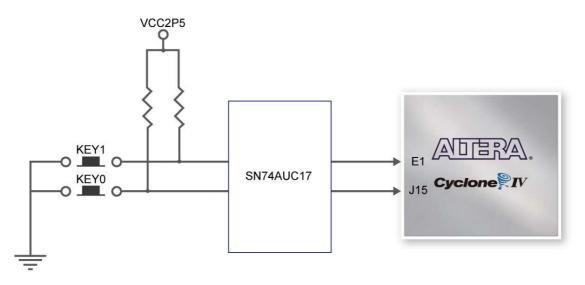


Figure 3-3 Connections between the push-buttons and Cyclone IV FPGA

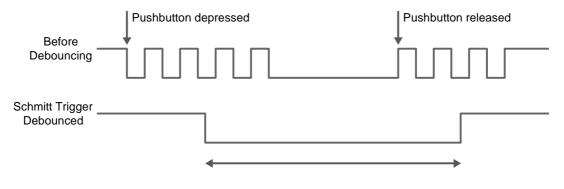


Figure 3-4 Pushbuttons debouncing

■ LEDs

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



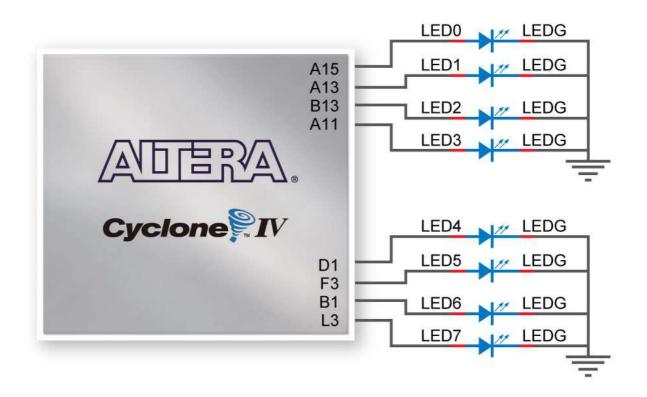


Figure 3-5 Connections between the LEDs and Cyclone IV FPGA

■ **DIP** Switch

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

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

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

Table 3-2 Pin Assignments for LEDs



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

Table 3-3 Pin Assignments for DIP Switches
--

3.3 SDRAM Memory

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



Figure 3-6 Connections between FPGA and SDRAM

Table 3-4 SDRA	M Pin Assignments
----------------	-------------------

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



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

3.4 I2C Serial EEPROM

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

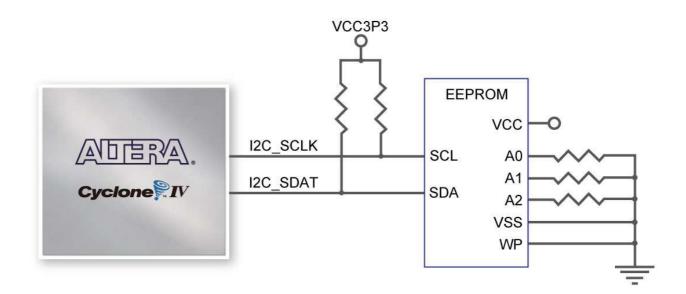


Figure 3-7 Connections between FPGA and EEPROM

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

 Table 3-5 Pin Assignments for I2C Serial EEPROM

3.5 Expansion Headers

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



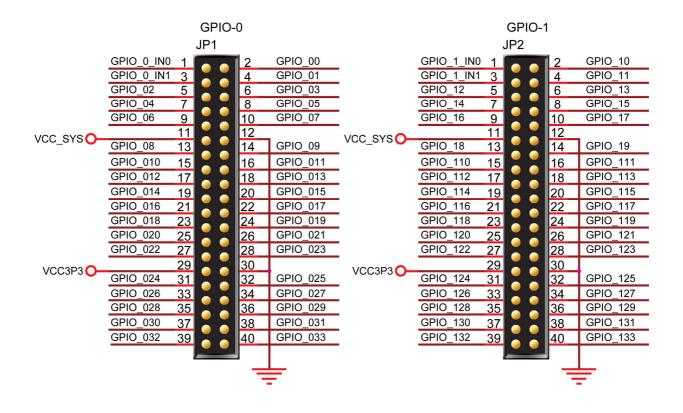


Figure 3-8 Pin arrangement of the GPIO expansion headers

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

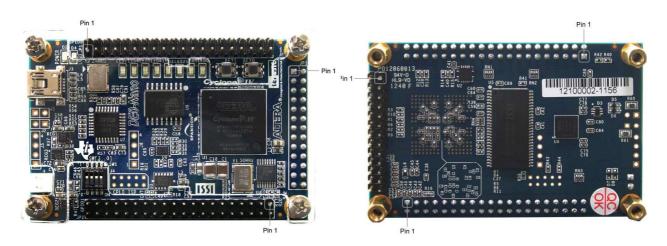


Figure 3-9 Pin1 locations of the GPIO expansion headers

		8	
Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_0_IN0	PIN_A8	GPIO Connection DATA	3.3V
GPIO_00	PIN_D3	GPIO Connection DATA	3.3V
GPIO_0_IN1	PIN_B8	GPIO Connection DATA	3.3V
GPIO_01	PIN_C3	GPIO Connection DATA	3.3V

Table 3-6 GPIO-0 Pin Assignments



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

Table 3-7 GPIO-1 Pin Assignments

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



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

3.6 A/D Converter and 2x13 Header

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

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



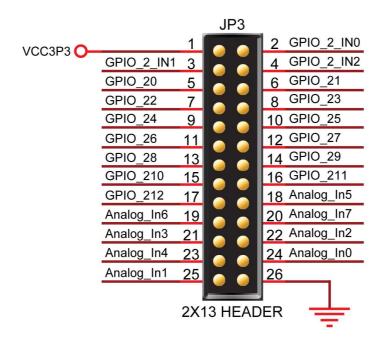
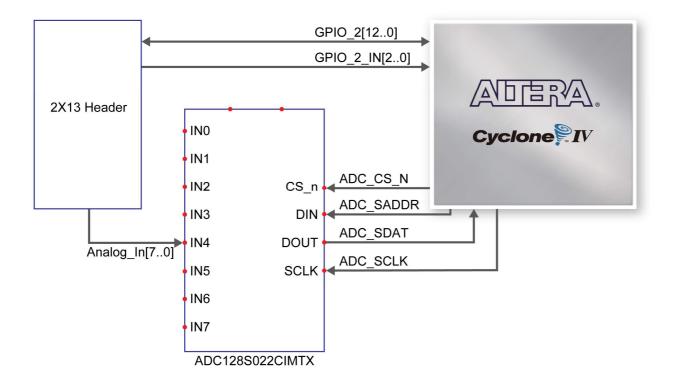
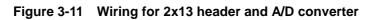


Figure 3-10 Pin distribution of the 2x13 Header

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





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



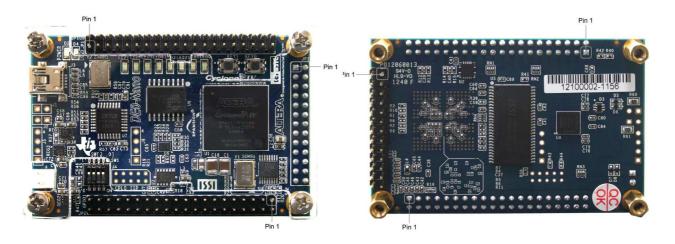


Figure 3-12 Pin1 locations of the 2x13 header

a ,		In Assignments for 2x13 Heade	
Signal Name	FPGA Pin No.	Description	I/O Standard
GPIO_2[0]	PIN_A14	GPIO Connection DATA[0]	3.3V
GPIO_2[1]	PIN_B16	GPIO Connection DATA[1]	3.3V
GPIO_2[2]	PIN_C14	GPIO Connection DATA[2]	3.3V
GPIO_2[3]	PIN_C16	GPIO Connection DATA[3]	3.3V
GPIO_2[4]	PIN_C15	GPIO Connection DATA[4]	3.3V
GPIO_2[5]	PIN_D16	GPIO Connection DATA[5]	3.3V
GPIO_2[6]	PIN_D15	GPIO Connection DATA[6]	3.3V
GPIO_2[7]	PIN_D14	GPIO Connection DATA[7]	3.3V
GPIO_2[8]	PIN_F15	GPIO Connection DATA[8]	3.3V
GPIO_2[9]	PIN_F16	GPIO Connection DATA[9]	3.3V
GPIO_2[10]	PIN_F14	GPIO Connection DATA[10]	3.3V
GPIO_2[11]	PIN_G16	GPIO Connection DATA[11]	3.3V
GPIO_2[12]	PIN_G15	GPIO Connection DATA[12]	3.3V
GPIO_2_IN[0]	PIN_E15	GPIO Input	3.3V
GPIO_2_IN[1]	PIN_E16	GPIO Input	3.3V
GPIO_2_IN[2]	PIN_M16	GPIO Input	3.3V

Table 3-8 Pin Assignments for 2x13 Hea	ıder
--	------

 Table 3-9 Pin Assignments for ADC

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

3.7 Digital Accelerometer

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

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

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

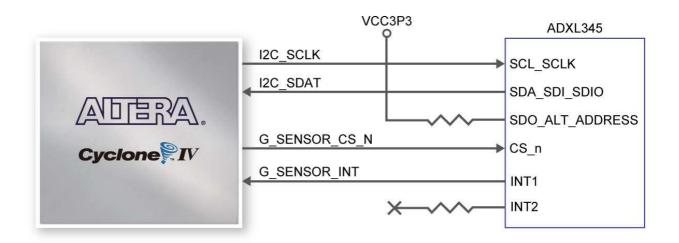


Figure 3-13 Wiring between the ADXL345 and the Cyclone IV E device

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

3.8 Clock Circuitry

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

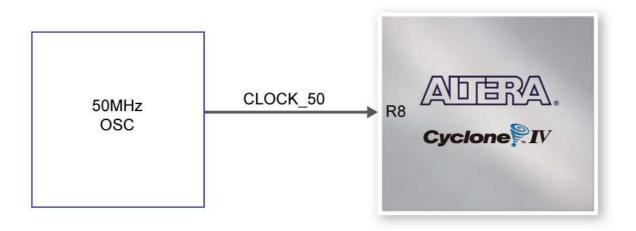


Figure 3-14 Block diagram of the clock distribution

3.9 Power Supply

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

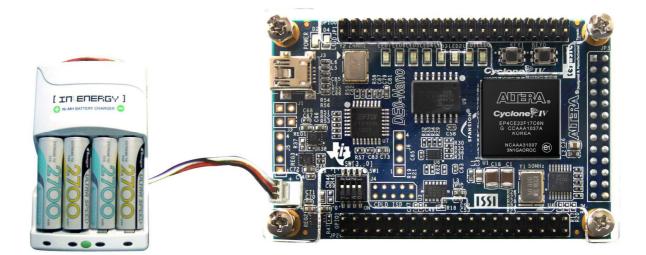


Figure 3-15 Portable Battery Connection



Power Distribution System

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

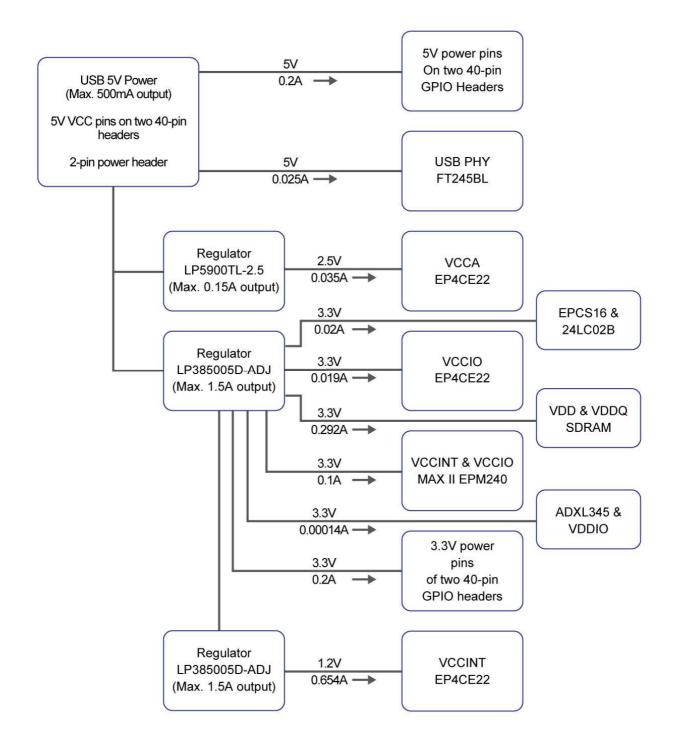


Figure 3-16 DE0-Nano Power Distribution System



Chapter 4

DE0-Nano Control Panel

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

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

4.1 Control Panel Setup

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

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

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

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

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

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

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

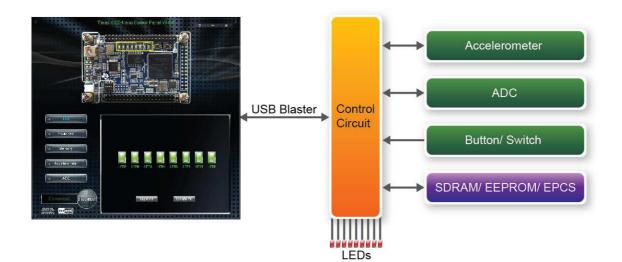


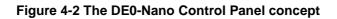
8. The Control Panel is now ready for use; experience it by setting the ON/OFF status for some LEDs and observing the result on the DE0-Nano board.



Figure 4-1 The DE0-Nano Control Panel

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





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

4.2 Controlling the LEDs

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



Figure 4-3 Controlling LEDs

4.3 Switches and Pushbuttons

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





Figure 4-4 Monitoring switches and buttons

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

4.4 Memory Controller

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





Figure 4-5 Accessing the SDRAM

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

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

1. Specify the starting address in the Address box.

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

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

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

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

0123456789ABCDEF

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

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

terasIC Terasic DE0-Nano User Manual

1. Specify the starting address in the Address box.

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

3. Press Load Memory Content to a File button.

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

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

4.5 Digital Accelerometer

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

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

 Table 4-1 acceleration values convert rule



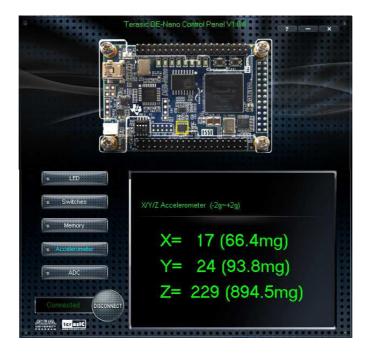


Figure 4-6 Digital Accelerometer status

4.6 ADC

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



Figure 4-7 ADC Readings

4.7 Overall Structure of the DE0-Nano Control Panel

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

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

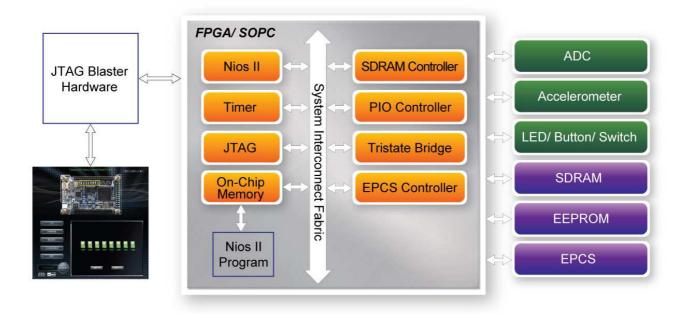


Figure 4-8 The block diagram of the DE0-Nano Control Panel



Chapter 5

DE0-Nano System Builder

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

5.1 Introduction

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

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

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

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

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

3. Performance degeneration because of improper pin assignments.

5.2 General Design Flow

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

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



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

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

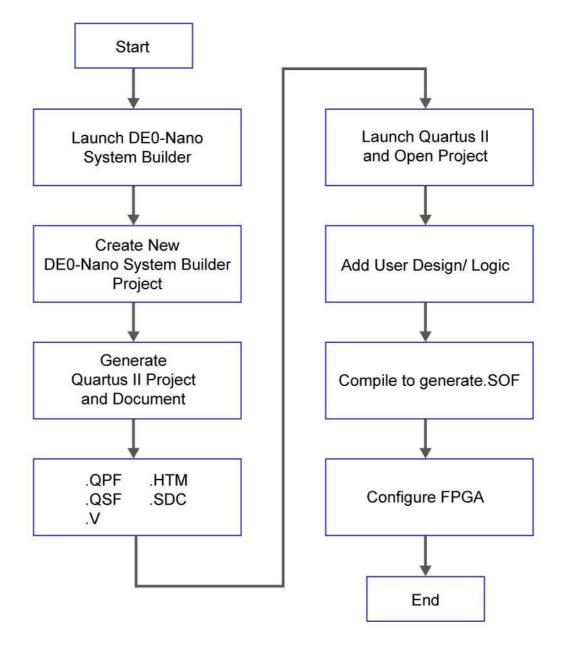


Figure 5-1 The general design flow of building a design

5.3 Using DE0-Nano System Builder

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

■ Install and launch the DE0-Nano System Builder

The 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 DE0-Nano System Builder ¥1.0.0			🔺 🖿 🔀
		System Configuratio Project Name: DE0_NANO	n
DE0-Nano FPGA Board		CLOCK Button x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header	 ☑ LED x 8 ☑ Dip Switch x 4 ☑ ADC ☑ EPCS ☑ 2x13 Pin Header
		None Prefix Name: GPIO-1 Header	
Load Setting Ge	nerate	None	~
Save Setting	Exit	Prefix Name:	

Figure 5-2 The DE0-Nano System Builder window

■ Input Project Name

Input project name as show in Figure 5-3.

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



Terasic DE0-Nano System Builder ¥1.0.0	
NIVERSITY PROGRAM	System Configuration Project Name: DE0_NANO
DEO-Nano FPGA Board	CLOCK CLED x 8 Button x 2 Dip Switch x 4 SDRAM, 32MB ADC EEPROM, 2Kb EPCS Accelerometer 2x13 Pin Header GPIO-0 Header Prefix Name: GPIO-1 Header
Load Setting Generate	None
Save Setting Exit	Prefix Name:

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

System Configuration

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

Terasic DEO-Nano System Builder ¥1.0	.0		- E 🔁 🔀
		System Configuration Project Name: DE0_NANO	
DEO-Nano FPGA B		CLOCK C Button x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header None Prefix Name: GPIO-1 Header	 ✓ LED x 8 ✓ Dip Switch x 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header
Load Setting	Generate	None	~
Save Setting	Exit	Prefix Name:	

Figure 5-4 System Configuration Group

GPIO Expansion



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

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

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



Terasic DE0-Nano System Builder ¥1.0.0		N
NIVERSITY PROGRAM	System Configuration Project Name: DE0_NANO	
DEO-Nano FPGA Board	CLOCK UButton x 2 SDRAM, 32MB EEPROM, 2Kb Accelerometer GPIO-0 Header Prefix Name: GPIO-1 Header	 ✓ LED x 8 ✓ Dip Switch x 4 ✓ ADC ✓ EPCS ✓ 2x13 Pin Header
Load Setting Generate	None	~
Save Setting Exit	Prefix Name:	

Figure 5-6 Project Settings

Project Generation

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

Table 5-1	The files generated by DE0-Nano System Build	er
Table 3-1	The mes generated by DE0-Nano System Dund	CI.

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

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



Chapter 6

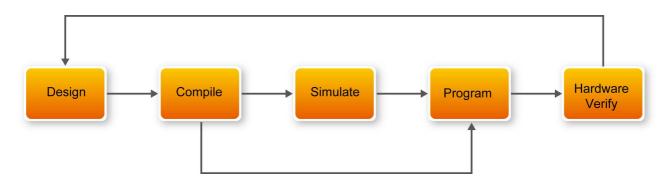
Tutorial: Creating an FPGA Project

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

6.1 Design Flow

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

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





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



6.2 Before You Begin

This tutorial assumes the following prerequisites

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

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

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

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

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

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

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

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





Figure 6-2 Found New Hardware Wizard

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



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



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

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

Figure 6-4 Specify the location of the driver

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

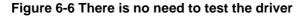
C:\altera\10.1\quartus\drivers\usb-blaster. Click OK and then upon returning to **Figure 6-4** click Next. At this point the installation will commence, but a dialog box in **Figure 6-6** will appear indicating that the driver has not passed the Windows Logo testing. Click Continue Anyway.



Browse For Folder	? 🗙
Select the folder that contains drivers for your hardware.	
🖃 🧰 10.1	^
🗉 🧰 installer	
🗉 🧰 ip	
🗉 🧰 nios2eds	
🗆 🧰 quartus	
🕀 🧰 bin	=
🕀 🧰 bin64	
🗉 🧰 common	
🗉 🧰 cusp	
🖃 🧰 drivers	
i386	
🗉 🧰 sentinel	
🖃 🗁 usb-blaster	
🚞 x32	
🛅 x64	
🗉 🧰 dsp_builder	~
🛛 🕀 🛅 eda	
To view any subfolders, click a plus sign above.	
ок с	ancel

Figure 6-5 Browse to find the location







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

Found New Hardware Wize	ard
	Completing the Found New Hardware Wizard The wizard has finished installing the software for: Altera USB-Blaster
	Click Finish to close the wizard.
	< 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 Proj	ect Vizard	×
Introduc	tion	
The New Proje	ct Wizard helps you create a new project and preliminary project settings, including the following:	
	Project name and directory Name of the top-level design entity Project files and libraries Target device family and device EDA tool settings	
You can chang the various pa	e the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use ges of the Settings dialog box to add functionality to the project.	
Don't show	me this introduction again	
	< <u>B</u> ack <u>N</u> ext > Einish Cancel <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	
	5
< Back Next > Finish Cancel Help	

Figure 6-9 Project information

d. Click Next.

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



			Show in 'Available devices' list					
Eamily: Cyclone IV E					Package: Any			
							~	
Target device				Pin <u>c</u> ount:	Any	~		
				Speed grade: Any			~	
				Show advanced devices				
and Rames and		5 A .		and a second second				
e selected in 'Availab	le devices'	list		HardCopy	compatible only			
1.2V	22320	80	608256			4	20	
1.0V	22320	80	608256	132	2	4	20	
1.2V	22320	154	608256	132	2	4	20	
1.2V	22320	154	608256	13	2	4	20	
1.2V	22320	154	608256	132	2	4	20	
1.2V	22320	154	608256	132	2	4	20	
1.0V	22320	154	608256		T.	4	20	
1 (11)		45.4	C00055	177	·	1	3	
							Collin	
-								
	Selected by the Fitter selected in 'Availab Core Voltage 1.2V 1.2V 1.2V 1.2V 1.2V 1.2V	Core Voltage LEs 1.2V 22320 1.2V 22320	Core Voltage LEs User I/Os 1.2V 22320 80 1.2V 22320 80 1.2V 22320 154 1.2V 22320 154	Core Voltage LEs User I/Os Memory 1.2V 22320 80 608256 1.0V 22320 80 608256 1.2V 22320 154 608256 1.0V 22320 154 608256	Core Voltage LEs User I/Os Memory Bits 1.2V 22320 80 608256 133 1.2V 22320 154 608256 133 1.0V 22320 154 608256 133	Core Voltage LEs User I/Os Memory Bits Embedded multiplier 9-bit elements 1.2V 22320 80 608256 132 1.2V 22320 80 608256 132 1.2V 22320 154 608256 132 1.2V 2232	Image: Selected in 'Available devices' list Memory Bits Embedded multiplier 9-bit elements PLL 1.2V 22320 80 608256 132 4 1.2V 22320 154 608256 132 4	

Figure 6-10 Specify the Device Example

f. Click Finish.

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



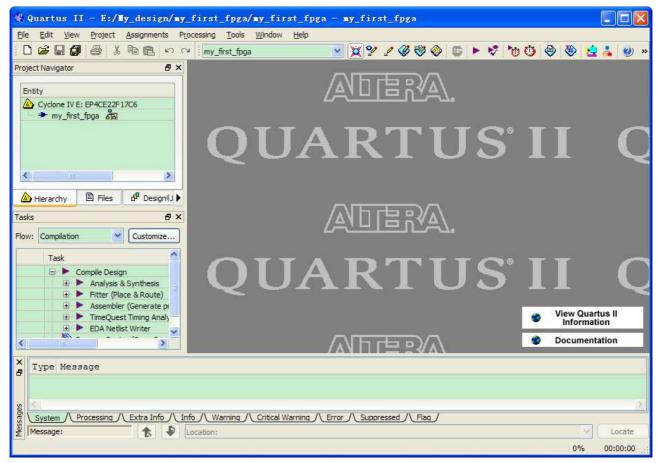


Figure 6-11 my_first_fpga project

6.5 Creating an FPGA design

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

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

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



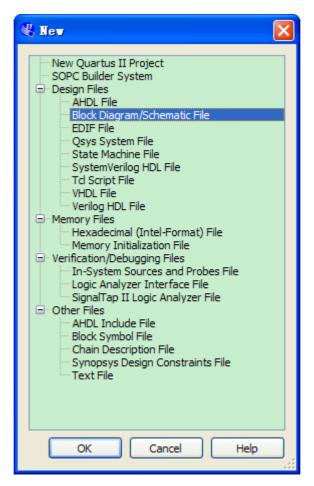


Figure 6-12 New BDF

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



<mark>∛ Quartus II - E:/Iy_design/my</mark> Ele Edit <u>V</u> iew Project Assignments Pr		the local sector is a sector in the sector is a sector in the sector is a sector in the sector is a sector is a	- my_fi	rst_fpga	F1						
	my_first_fpga		🔽 🔀 🕯	1 2 3	3 📎		17	0 🕹	100	2 👗	0 >
Project Navigator 🛛 🗗 🗙	Ð	my_first_fpga.bo	df	×							
Entity	1 🖶 💫 🔍 🥙) A Ð 💑 🕇 🕻		1/1	\square	\circ \land	75				a ,
									· · · · · · · · · · · · · · · · · · ·		· · · · · · · · · · · ·
Hierarchy								· · · · · · · · · · · · · · · · · · ·			· · · · · · · · · · · · · · · · · · ·
Tasks 🗗 🗙			· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · ·				· · · · · · · · · · · · · · · · · · ·	· · · · · · · · ·		
Task											
Comple Design Analysis & Synthesis Fitter (Place & Route) Assembler (Generate pi			· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·				· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·		
EDA Netlist Writer											
X F Type Message	nfo /\ Warning /\	Critical Warning /\	Error /_Sup	pressed /\	Flag /						2
Message:	Location:								×	Lo	ocate
								638, 21	8 0%	00:0	00:00

Figure 6-13 Bank BDF

• Adding a Verilog HDL to the Schematic

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

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

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



Save As					? 🛛
Save in:	🗁 my_first_fpga	-	- 🔁	<u>r </u>	
My Recent	incremental_db				
Documents	counter_bus_m counter_bus_m pll.v				
Desktop	pll_bb.v simple_counter.	.v			
My Documents					
My Computer					
My Network Places	File name:	simple_counter.v		•	Save
1,2000	Save as type:	Verilog HDL Files (*.v *.vlg *.verile	log)	•	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,
                               counter_out
 4
 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
    E
         begin
            counter_out <= counter_out + 1;</pre>
12
                                                // increment counter
13
         end
     L
      endmodule
                                                 // end of module counter
14
15
```

Figure 6-15 The Verilog File of simple_counter.v

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

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

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



Figure 6-16 Create Symbol File was Successful

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



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

🔁 Symbol		
Libraries:	Eimple_counter CLOCK_5 counter_out[310]	
Name:	inst	
<u>R</u> epeat-insert mode		
Insert symbol as block Issuert symbol as block Issuert symbol as block		
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**.

	sin	nple_counter			
· · ·				· · ·	•
		CLOCK_5	counter_out[310]		
	in	st			•
· · ·	· ·				•
· · ·	· · · ·	· · · · · · · ·	· · · · · · · · · · · · · · · · · ·	· · · · · · · · ·	

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**.

🐇 MegaWizard 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> © <u>Greate a new custom megafunction variation</u> © Greate an existing custom megafunction variation © Cogy an existing custom megafunction variation Copyright (C) 1991-2010 Altera Corporation	
Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish	

Figure 6-19 Mega Wizard Plug-In Manager



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

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

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

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

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

e. Click Next.

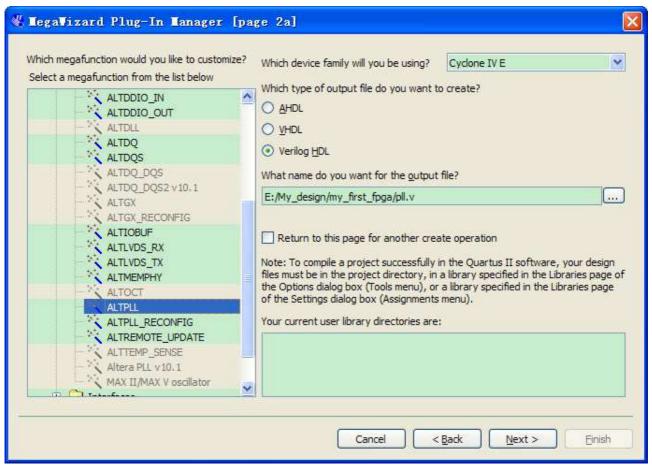


Figure 6-20 MegaWizard Plug-In Manager [page 2a] Selections

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

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



d. Click Next.

• MegaWizard Plug-In Manager [page 3 o	of 14]	? 🛛
altpll		About Documentation
Parameter PLL Output 4ED Settings Clocks 4ED 4ED		
General/Modes > Inputs/Lock > Bandwidth/SS	Clock switchover	
General/Modes Inputs/Lock Bandwidth/SS	Currently selected device family: Able to implement the requested PLL General Which device speed grade will you be using? Use military temperature range devices only What is the frequency of the inclk0 input? Set up PLL in LVDS mode PLL Type Which PLL type will you be using?	MHz V
	Cancel < Ba	ck 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.



▼ MegaVizard Plug-In Manager []	page 4 of 14]
	About Documentation
1 Parameter Settings Reconfiguration Clock	ut 4 EDA 5 Summary
General/Modes Inputs/Lock Ban pll incik0 frequency: 50.000 MHz Operation Mode: Normal Cik Ratio Ph (dg) DC (%) c0 1/1 0.00 50.00 Cyclone IV E	Addith/SS Clock switchover Able to implement the requested PLL Optional Inputs Create an 'pliena' input to selectively enable the PLL [Create an 'areset' input to asynchronously reset the PLL] [Create an 'areset' input to selectively enable the phase/frequency detector Lock Output [Create 'locked' output] [Enable self-reset on loss lock Advanced Parameters Using these parameters is recommended for advanced users only [Create output file(s) using the 'Advanced' PLL parameters - Configurations with output clock(s) that use cascade counters are not supported
	Cancel < Back Next > Finish

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**.



• TegaTizard Plug-In Tanager [pa	age 6 of 12]		? 🛛
			bout Documentation
1 Parameter 2 PLL 3 Output	4 EDA Summary		
Settings Reconfiguration Clocks			
pll	c0 - Core/External Output CI Able to implement the requested PLL	ock	
incik0 frequency: 50.000 MHz C0 Operation Mode: Normal [Clk Ratio Ph (dg) DC (%)	Use this clock		
c0 1/10 0.00 50.00	Clock rap settings	Requested Settings	Actual Settings
Cyclone IV E	Enter output dock frequency: Enter output dock parameters:	100.0000000 MHz	5.000000
	Clock multiplication factor	1	
	Clock division factor	10	opy 10
	Clock phase shift	0.00 🗘 deg	• 0.00
	Clock duty cycle (%)	50.00	50.00
		Description	Value
		Primary clock VCO frequency (MHz)	600.000
	Note: The displayed internal settings of the PLL is recommended	Modulus for M counter Modulus for N counter	12
	for use by advanced users only	Initial VCO phase cycles for M counter	1
		VCO phase tap for M counter	0 🗸
			ibility Indicators
		Cancel < Back	Next > Einish

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.



MegaWizard Plug-In Manager [pag	e 12 of 12]	
		About Documentation
Parameter PLL Settings PLL Clocks	4 EDA 5 Sur	nmary
pll inclk0 inclk0 frequency: 50.000 MHz C0	checkmark indicates maintained in subsec	wish to generate. A gray checkmark indicates a file that is automatically generated, and a green an optional file. Click Finish to generate the selected files. The state of each checkbox is guent MegaWizard Plug-In Manager sessions. g-In Manager creates the selected files in the following directory: rst_fpga\
Operation Mode: Normal	File	Description
Clk Ratio Ph (dg) DC (%)	l ✓ pll.v	Variation file
<u>c0 1/10 0.00 50.00</u>		PinPlanner ports PPF file
Cyclone IV E	pll.inc	AHDL Include file
	pll.cmp	VHDL component declaration file
	⊘ pll.bsf	Quartus II symbol file
	pll_inst.v	Instantiation template file
	✓ pll_bb.v	Verilog HDL black-box file
		Cancel <back next=""> Finish</back>

Figure 6-24 Wizard-Created Files

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



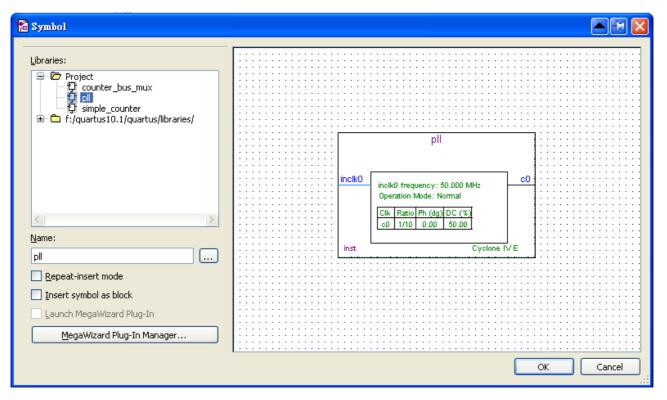


Figure 6-25 PLL Symbol

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

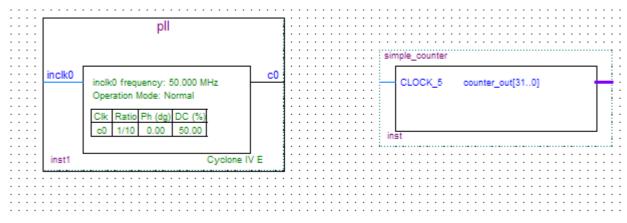


Figure 6-26 Place the PLL Symbol

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

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



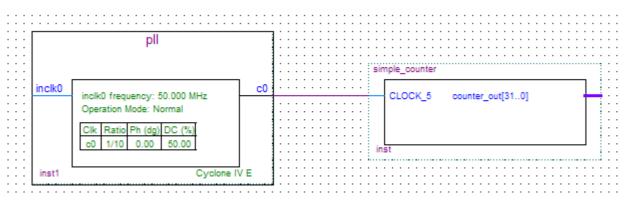


Figure 6-27 Draw a Bus Line connect pll c0 port to simple_counter CLOCK_5 port

■ Adding an Input pin to the Schematic

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

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

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

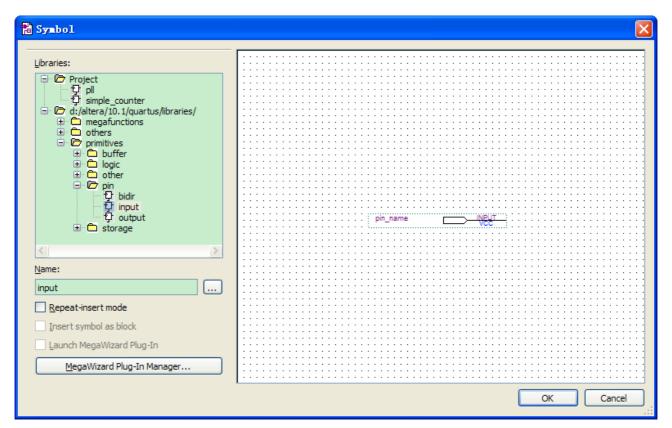


Figure 6-28 Input pin symbol



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

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

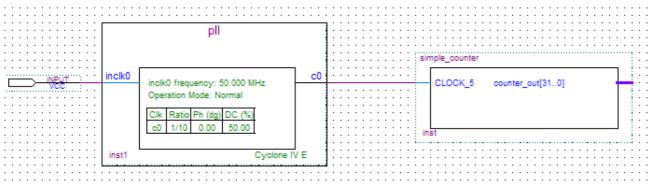


Figure 6-29 Connecting the PLL symbol and Input port

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

■ Adding an Output bus to the Schematic

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

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



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

Figure 6-30 Change the input port name

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

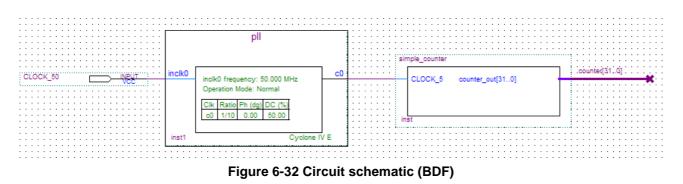
3. Type counter [31..0] as the bus name (see **Figure 6-31**). The notation [X ..Y] is the Quartus II method for specifying the bus width in BDF schematics, where X is the most significant bit (MSB) and Y is the least significant bit (LSB).

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



📸 Bus Pro	operties 🔀	J
General	Font Format	
<u>N</u> ame:	counter[310]	
	name in block design file.	
	lame in block design file.	
	OK Cancel Help	
		2

Figure 6-31 Change the output BUS name



■ Adding a Multiplexer to the Schematic

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



- 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.

Which megafunction would you like to customize? Select a megafunction from the list below Installed Plug-Ins Altera SOPC Builder Aithmetic Communications DSP Gates LPM_CLSHIFT LPM_CONSTANT LPM_MUX Interfaces JTAG-accessible Extensions Memory Compiler	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 E:/My_design/my_first_fpga/counter_bo Return to this page for another creat Note: To compile a project successfully in files must be in the project directory, in the Options dialog box (Tools menu), or of the Settings dialog box (Assignments 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 a library specified in the Libraries page menu).
--	---	---

Figure 6-33 Selecting Ipm_mux

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

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



• MegaVizard Plug-In Manager [page	• 3 of 5] ? 🛛
LPM_MUX	<u>About</u> <u>Documentation</u>
1 Parameter Settings 2 EDA 3 Summary	
counter_bus_mux sel data0x[3.0] data1x[3.0]	Currently selected device family: Cyclone IV E Match project/default How many 'data' inputs do you want? How wide should the 'data' input and the' result' output buses be? Do you want to pipeline the multiplexer? No Yes, I want an output latency of Create an asynchronous Clear input Create a Clock Enable input
Resource Usage 1 lpm_mux	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 6-34 Ipm_mux settings

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



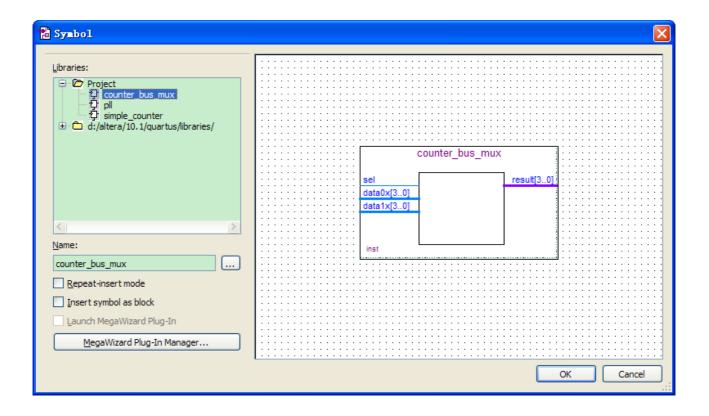


Figure 6-35 lpm_mux Symbol

13. Click OK

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

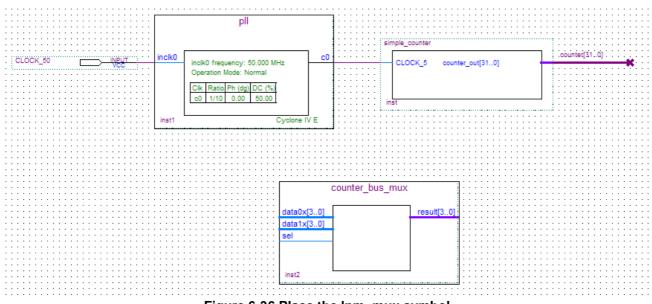


Figure 6-36 Place the lpm_mux symbol

68



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

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

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

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

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

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

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

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

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

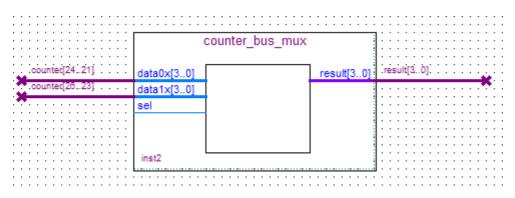


Figure 6-37 Renamed counter_bus_mux Bus Lines

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

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



ibraries:		
	<u>auteur</u> pn_name	
lame:		
output		
Repeat-insert mode Insert symbol as block		
Launch MegaWizard Plug-In		
MegaWizard Plug-In Manager		::::::

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				 	OUT						 [30	
	data1x[30]				 			: :	ļ		···· •					
	sel			· · · ·	 	 	· · ·	 	· · · ·	· · · ·	· · · ·	· · ·	· · ·	· · ·	 	· · · ·
· · · · · · · · · · · · · · · · · · ·	inst2								· · · ·							
					 			· ·	· · ·	::	: :				: :	::

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).



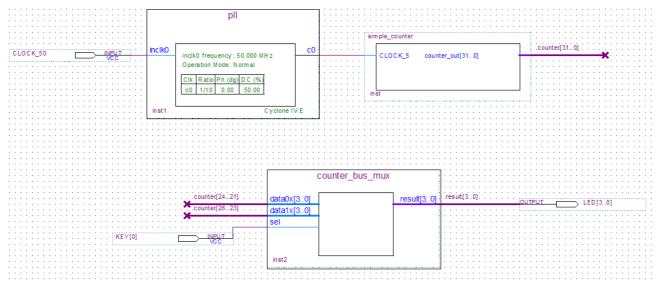


Figure 6-40 Adding the KEY [0] Input Pin

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**



Named: *	~		₽×		C10	Top View - Wite Bond one IV E - EPECE 20F1TCe		
N	lode Name	Direction	Location					
🕀 💕 KI	EY[00]	Input Group				CONCOMPACE.		
🕀 🐨 LE	ED[30]	Output Group						
< <ner< th=""><th>w group>></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></ner<>	w group>>							
<		1	>		265	COOCEDOCOCCOA		
× Named	i: * 🗸 🔀	»Edit: 🗶 🗸					Filter: Pins: a	all
8	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	
De c	LOCK_50	Input				2.5 V (default)		
📂 К	EY[0]	Input				2.5 V (default)		
🔊 Li	ED[3]	Output				2.5 V (default)		
🔊 Li	ED[2]	Output				2.5 V (default)		
🔊 Li	ED[1]	Output				2.5 V (default)		
🕑 Li	ED[0]	Output				2.5 V (default)		
	w node>>							
< <ne< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></ne<>								
All Pins								

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.

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**.



Groups 🗗 🗙 Named: *					Car A00	Top View - Wire Band Ione IV E - EPecE25F1TC0		
	Node Name	Node Name Direction Locati Image: Direction Locati Image: Direction Image: Direction Image: Direction Output PIN_B13 Image: Direction Image: Direction Image: Direction Output PIN_A13 Image: Direction Image: Direction <t< th=""><th></th><th></th><th></th><th></th><th></th></t<>						
	× Named: * 💉	🗱 Edit: 🗶 🗸					Filter: Pins: all	
1	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)		
	KEY[0]	Input	PIN_J15	5	85_N0	2.5 V (default)		
	LED[3]	Output	PIN_A11	7	B7_N0	2.5 V (default)		
1	LED[2]	Output	PIN_B13	7	B7_N0	2.5 V (default)		
	LED[1]	Output	PIN_A13	7	B7_N0	2.5 V (default)		
	LED[0]	Output	PIN_A15	7	B7_N0	2.5 V (default)		
	< <new node="">></new>							

Figure 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)



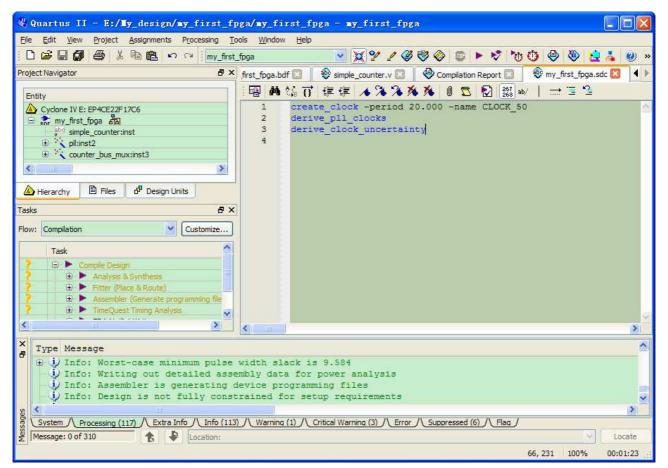


Figure 6-43 Default SDC

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		
Ele Edit View Project Assignments Processing Too			
	pga 🛛 🛛 💥 🐓 🖉 🍕	🥙 💿 🕨 😵 🏷	🛈 🕘 🕲 🖄 🙆 »
Project Navigator & ×	first_fpga.bdf 🔟 🛛 🕸 simple_counter.v 🔯	Compilation Report	🕸 my_first_fpga.sdc 🖾 🚺
Entity Cyclone IV E: EP4CE22F17C6 Task Compilation Customize Task Compile Design Solution Customize Task Customize Task Customize Custo	Table of Contents ₽ Image: Flow Summary Image: Flow Settings Image: Flow Non-Default Global Settings Image: Flow Elapsed Time Image: Flow OS Summary Image: Flow Log Image: Flow Log Image: Flow Log Image: Flow Simple Synthesis Image: Flow Simple Synthesis	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_foga _first_foga done IV E
		<	
Type Message i) Info: 5 registers lost all their i) Info: Generating hard block part i) Info: Implemented 38 device rest i) Info: Quartus II Analysis & Synt System (Processing (27) (Extra Info (Info (27) / Message: 0 of 107	tition "hard block:auto generate burces after synthesis - the fin thesis was successful. 0 errors,	d_inst" al resource count m O warnings	

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 Status Successful - Fri Jan 14 17:42:39 2011 Quartus II Version 10.1 Build 153 11/29/2010 SJ Full Version Revision Name my_first_fpga Top-level Entity Name my_first_fpga Family Cyclone IV E Device EP4CE22F17C6 Timing Models Final Patel Lements 21 (22 200 (< 1.8%)	Flow Summary	
Total logic elements 31/22,320 (<1%)	Flow Status Quartus II Version Revision Name Top-level Entity Name Family Device Timing Models Total logic elements Total combinational functions Dedicated logic registers Total registers Total pins Total virtual pins Total memory bits Embedded Multiplier 9-bit elements	10.1 Build 153 11/29/2010 SJ Full Version my_first_fpga Cyclone IV E EP4CE22F17C6 Final 31 / 22,320 (< 1 %) 31 / 22,320 (< 1 %) 27 / 22,320 (< 1 %) 27 6 / 154 (4 %) 0 0 / 608,256 (0 %) 0 / 132 (0 %)

Figure 6-45 Compilation Report Example

6.9 Program the FPGA Device

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

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

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

Program the FPGA using the following steps.

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



🕸 Programmer -	E:/Hy_design/my_f	irst_fpga/ n y_	first_fpga	- my_first_	fpga - [m	y 📕	
<u>Eile E</u> dit <u>V</u> iew Pr	ocessing <u>T</u> ools <u>W</u> indow						
Hardware Setup	. USB-Blaster [USB-0] P to allow background program	Mode: JTAC		Prog	ress:		
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check
Mt Stop	my first fpga.sof	EP4CE22F17	0013888D	FFFFFFFF	V		
Auto Detect							
X Delete							
Add File							
Change File	<	Auto					>
Save File							^
Add Device							
1 Up	STOR ICT						唐
Down							
	EP4CE22F						
	TDO						×
							1.23

Figure 6-46 Programmer Window

2. Click Hardware Setup.

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



Ð	Hardware Setup			×
	Select a programming har	TAG Settings dware setup to use when pro nly to the current programme		This programming
	Currently selected hardw	No Hardwara		×
	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 - D	:/Home/User/Desktop/	allen/my_first_fp	ga/my_first_fp	ga - my_first_	fpga - [my	. 💶 🗖 🔀
File Edit View Pi	rocessing Tools Window	Ŷ				
🔔 Hardware Setup.	USB-Blaster [USB-0]	Mode: JTAG		Progre	ss: 100% (S	Successful)
Enable real-time IS	P to allow background progr	amming (for MAX II and	MAX V devices)			
📕 Start	File	Device	Checksum	Usercode	Program/ Configure	Verify Blar Che
Stop	my_first_fpga.sof	EP4CE22F17	00137CDB	FFFFFFF		
Auto Detect						
X Delete						
Add File						
👺 Change File	<					>
Save File						<u>^</u>
Add Device						=
The Up	310)A					
- Down		·				
						~

Figure 6-48 Downloading Complete

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

6.10 Verify The Hardware

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

Verify the design by performing the following steps:

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

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

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

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



Eamily: Cyclone i	Device family				Show in 'Available devices' list			
Eamily: Cydone IV E				Package:	Any	~		
					and the second se			
Devices: All			~	Pin <u>c</u> ount:	Any	Y		
Target device				Speed grade:	Any	~		
second in the	W WAY OF SHOP			Show adva	nced devices			
O Auto device s	elected by the Fitter							
Specific device	e selected in 'Availab	le devices	list	HardCopy	compatible only			
O Other: n/a				6				
				Device and Pin C	Options			
vailable devices:								
Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit element	5 ^		
EP4CE22E22C9L	1.0V	22320	80	608256	132			
EP4CE22E22I7	1.2V	22320	80	608256	132			
EP4CE22E22I8L	1.0V	22320	80	608256	132			
EP4CE22F17A7	1.2V	22320	154	608256	132	- 10		
EP4CE22F17C6	1.2V	22320	154	608256	132	40		
EP4CE22F17C7	1.2V	22320	154	608256	132			
EP4CE22F17C8	1.2V	22320	154	608256	132			
EP4CE22F17C8L	1.0V	22320	154	608256	132			
EP4CE22F17C9L	1.0V	22320	154	608256	132			
	1.2V	22320	154	608256	132			
		00000	154	608256	132			
EP4CE22F17I8L	1.0V	22320	154	608256	132	- 0		

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:	
General	Unused Pins
 Configuration Programming Files Unused Pins Dual-Purpose Pins Capacitive Loading Board Trace Model I/O Timing Voltage Pin Placement Error Detection CRC CvPCIe Settings 	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: triangle of the provide other pins on the device. To reserve other pins individually, use the Assignment Editor. Image: triangle other pins Reserve all unused pins: As input tri-stated Image: triangle other pins Description: Reserves all unused pins on the target device in one of 5 states: as inputs that are tristated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.
	Reset
	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 DEO-Nano board.

7.2 Creation of Hardware Design

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

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



🐇 Qu	artus	II			
File	<u>E</u> dit	<u>V</u> iew	<u>P</u> roject	<u>A</u> ssignments	P <u>r</u> o
<u>א</u>	w			Ctrl+N	
൙ Op	en			Ctrl+0	
<u>C</u> 1	ose			Ctrl+F4	
🔀 не	w Proje	ct <u>W</u> iz	ard		
🛃 Op	en P <u>r</u> oj	ect		Ctrl+J	
	we Proj				
C1	.os <u>e</u> Pro	ject			
	ve			Ctrl+S	
	we <u>A</u> s				
🗊 Sa	we All			Ctrl+Shift	+S
Fi	le Prop	erties			
Cr	eate /	Update			•
Ex	port <u>.</u>				
Co	nvert P	rogram	ming Files	i	
Pa	ge Set <u>u</u>	p			
👌 Pr	int Pre	<u>v</u> iew			
🖨 🗄	int			Ctrl+P	
Re	cent F <u>i</u>	les			•
Re	cent Pr	ojects			•
Ex	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>B</u> ack <u>N</u> ext > <u>E</u> inish Cancel <u>H</u> elp	J

Figure 7-2 New Project Wizard

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



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

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

ile name:					Add
File Name	Type Library	Design Entry/Synthesis Tool	HDL Version		Add All Remove
pecify the pa	ath names of any no	n-default libraries. User 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 DEO-Nano board. You should choose settings the same, as shown in **Figure 7-5**. Then click **Next** to get to the window as shown in **Figure 7-6**.



× ×
~
elements 🛆
~
>

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.



fool Type	Tool Name	Format(s)	Run Tool Automatically
)esign Entry/Syn…	<none> 🔽</none>	<none></none>	Run this tool automatically to synthesize the current design
imulation	<none></none>	<none> ·</none>	Run gate-level simulation automatically after compilation
iming Analysis	<none></none>	<none></none>	Run this tool automatically after compilation
ormal Verificat…	<none></none>		
oard-Level	Timing	<none></none>	
	Symbol	<none></none>	
	Signal Integrity	<none></none>	
	Boundary Scan	<none></none>	
	1		
		<u></u>	

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	IS:
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:	0
Device assignments: Family name: Device:	Cyclone IV E EP4CE22F 17C6
EDA tools:	
Design entry/synthesis:	<none> (<none>)</none></none>
Simulation:	<none> (<none>)</none></none>
Timing analysis:	<none> (<none>)</none></none>
Operating conditions:	
VCCINT voltage:	1.2V
Junction temperature range:	0-85 ℃
	< <u>Back</u> <u>Next</u> > <u>Einish</u> Cancel <u>H</u> elp

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



🕊 Quartus II - D:/myfirst_niosii/myfirst_niosii - myfirst_niosii	
<u>File Edit View Project Assignments Processing Tools Window Help</u>	
myfirst_niosii 🛛 😧 💯 🖉 🦑 🥙 🚳 🕨 🖈 🦻 🔞 🗐 😫 🎍 🗐 🛡	
Project Navigator 🛛 🗗 🗙	
Entity / 「自影公	
Cyclone IV E: EP4CE22F17C6	
→ myfirst_niosii da	
OILA DTIC [®]	T
{	
A Hierarchy E Files d ⁹ Design Units	
Status 🖉 🗙	
	- P
Information	
X Type Message	
	121
System Processing Extra Info Info Warning Critical Warning Error Suppressed Flag Message:	
🖞 Message:	te
0% 00:00:	00

Figure 7-8 A New Complete Project

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

5	<u>T</u> oo	ls	<u>W</u> indow <u>H</u> elp	
		Run	EDA Sim <u>u</u> lation Tool	۶.
		Run	. <u>E</u> DA Timing Analysis Tool	
2	'n	Lau	nch EDA Simulation Library <u>C</u> ompiler	
	°9.	Lau	nch Design Space Explorer	
I	\odot	<u>T</u> im	eQuest Timing Analyzer	
		Adv	risors	•
	>	C <u>h</u> i	p Planner (Floorplan and Chip Editor)	
	۰	Des	ign Partition Planner	
		Net	list <u>V</u> iewers	•
	ا للا	Sig	malTap II Logic A <u>m</u> alyzer	
	m	In-	System Memor <u>y</u> Content Editor	
	-	Log	zic Analyzer Interface Edito <u>r</u>	
	01	In_	System Sources and Probes Editor	
		Sig	malProbe Pins	
	\odot	Pro	grammer	
		<u>J</u> TA	G Chain Debugger	
	XX	Tra	msceiver Tool <u>k</u> it	
	1	Ext	ernal <u>M</u> emory Interface Toolkit	
	$\mathbb{N}_{\mathbb{N}}^{n}$	Meg	a <u>₩</u> izard Plug-In Manager	
		SOP	°C <u>B</u> uilder	
		Qsy	rs (Beta)	
	<u> </u>	Tel	. Ser <u>i</u> pts	
		Cus	tomi <u>z</u> e	
		_	ions	
1		Lic	ense Setup	

Figure 7-9 SOPC Builder Menu



🦊 Create New System - Alt File <u>E</u> dit <u>M</u> odule <u>S</u> ystem <u>Vi</u> ew To		ed.sopc (D:\ my f	irst_niosii\unnamed	.sopc)	- 🗆 🗙
System Contents System Generation					
Component Library	Target	Clock Settings			
Project Wew component Library Avalon Verification Suite Bridges and Adapters	Device Family: Cyclone IV E	Name	Source	MHz	Add Remove
Interface Protocols Legacy Components	Use C Mc . Create Ne	v System	Clock	Base	End
Merlin Components Peripherals Processor Additions Processors SLS University Program Video and Imace Processing New Edit Add	Target HDL: VH Imfo: Specify a Remove Edit	IDL a new system name. OK Cancel	Address Map <u>Fitters</u>]	3
	Exit Help		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	
L	
OK Cancel	

Figure 7-11 Create New System [1]



Litera SOPC Builder File Edit Module System View Ic	ools <u>H</u> elp					
System Contents System Generation						
Component Library	Target	Clock Settings				
Project	Device Family: Cyclone IV E 🛛 👻	Name	Source		MHz	Add
Library		clk_0	External		50.0	Remove
⊕Bridges and Adapters ⊕Interface Protocols					1.035-81-5	
Legacy Components Memories and Memory Contro	Use C Module	Description		Clock	Base	End
Peripherals PLL Processor Additions Processors S-SLS University Program Video and Image Processing New Edit. Add	Remove Edit X		Address Map	Filters	Filter: Default	>
	Exit Help	Prev Next	General	te		

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**.

<mark>Altera SOPC Builder</mark> ile <u>E</u> dit <u>M</u> odule <u>S</u> ystem ⊻iew <u>I</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
New component Library Avaion Verification Suite Bridges and Adapters Interface Protocols		clk_50	External	50.0	Remov
Heracci Components Memories and Memory Contro Memories and Memory Contro Merin Components Peripherals PLL	Use C Module	Description	Cloc	k Base	End

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**.

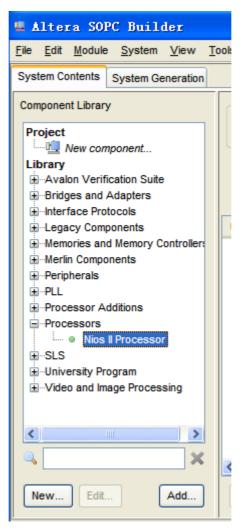


Figure 7-14 Add NIOS II Processor

밀망



😃 Nios II Proces	sor - cpu_0				
Mios	II Processor			A	out Documentation
Parameter Settings					
	es and Memory Interfaces $ ightarrow$	Advanced Features	MMU and MPU Settings > ЛТА	G Debug Module	Custom Instructions
Core Nios II					
Select a Nios II core:					100 C
	ONios II/e	○Nios II/s	●Nios II/f		<u>_</u>
Nios II Selector Guide Family: Cyclone IV E ^f system: 50.0 MHz cpuid: 0	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction		÷.
Performance at 50.0 MHz		Up to 32 DMIPS	Up to 57 DMIPS		
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs		<u>×</u>
Hardware Multiply: Ember Reset Vector: Mem Exception Vector: Memor Include MMU Only include the MMU wh Fast TLB Miss Exception	nory:		J Offset: @x0		
Include MPU		(000)			
Warning: Reset vecto	r and Exception vector cannot b	e set until memory devices a	e connected to the Nios II process	sor	
				Cancel	Back Mext > Finish

Figure 7-15 Nios II Processor

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



Litera SOPC Builder File Edit Module System View To	onle Nine II Heln			🛛
System Contents System Generation	500 M03 1 100			
Component Library	Target	Clock Settings		
Project	Device Family: Cyclone IV E	Name Source	MHz	Add
Library		clk_50 External	50.0	Remove
Avalon Verification Suite Bridges and Adapters				
	Use Conn Module	Description	Clock Base	End
Memories and Memory Controllers Merlin Components Peripherals	C cpu_0	Nios II Processor Avalon Memory Mapped Master Avalon Memory Mapped Master	[clk] clk_50 [clk] IRQ (2
⊕-PLL ⊕-Processor Additions	jtag_debug_mod	ule Avalon Memory Mapped Slave	[clk] # 0x0000800	0x00000fff
 SLS University Program Video and Image Processing 				
	<			>
New Edit Add	Remove	Address Map	Eitters Fitter: Default	
To Do: cpu_0: No reset vector has bee To Do: cpu_0: No exception vector has Marning: cpu_0: Reset vector and Exc	is been specified for this CPU. Please par		ssor	
	Exit Help	Prev Next General	ate	

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>F</u> ile <u>E</u> dit <u>M</u> odule <u>S</u> ystem <u>V</u> iew <u>T</u> o	ools Nios II <u>H</u> elp					
System Contents System Generation						
Component Library	Target	Clock Settings				
Project	Device Family: Cyclone IV E	Name	Source	MHz		Add
Library	n <u> </u>	clk_50	External	50.0		Remove
Legacy Components	Use Conn Module	Description		Clock	Base	End
Memories and Memory Controllers Merlin Components Peripherals Processor Additions Processors Nois I Processor SLS University Program Video and Image Processing	data	 v Connected uit ss on-MM	Mapped Master Mapped Master	[clk] clk_50 [clk] [clk] [clk]	IRQ 0 ⇒ 0x0000800	
× × ×	Edit	. Ctrl+E				>
New Edit Add	Remove Edit Detail		Address Map Eilte	ers Filter:	Default	
To Do: cpu_0: No reset vector has bee To Do: cpu 0: No exception vector has	en specified for this CPU. Plea	v Arbitration Base Address Ctrl+L	this issue olve this issue			
Warning: cpu_0: Reset vector and Exc	eption vector cannot be set u		to the Nios II processor			
	Exit Help	Prev Next	Generate			

Figure 7-17 Rename the CPU (1)

🚇 Altera SOPC Builder						
<u>File Edit Module System View To</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	5	0.0	Remove
Library - Avalon Verification Suite - Bridges and Adapters - Interface Protocols						
E-Legacy Components	Use Conn Module	Descriptio	n	Clock	Base	End
Memories and Memory Controllers Merlin Components	🖌 🖂 сри	Nios II Proc		[Clk]		
Peripherals PLL Processor Additions Processors SSLS University Program Video and Image Processing	instruction_mas data_master jtag_debug_mo	Avalon Mer	nory Mapped Master nory Mapped Master nory Mapped Slave	clk_50 [clk] [clk]		IRQ 0 8800 0x00000fff
×	<					>
New Edit Add	Remove Edit	VI	Address Map	Filters Fi	itter: Default	
To Do: cpu: No reset vector has been To Do: cpu: No exception vector has been Marning: cpu: Reset vector and Exception Vector and Exception		meterize the CPU to re	esolve this issue	Γ		
	Exit Help	Prev Ne:	kt 🕨 Genera	ite		

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> ie	w <u>T</u> o
System Contents System Genera	tion
Component Library	
Project	^
Library	
⊕Bridges and Adapters	
⊡…Interface Protocols ⊡…ASI	=
⊡Interlaken	
⊡Serial	-
 Avalon-ST JTAG 	
····· Avalon-ST Serial JTAG UART	
SPI (3 Wire Serial	
UART (RS-232 Se ⊡-Legacy Components	~
< >>	

Figure 7-19 Add the JTAG UART component



🖳 JTAG UART -	jtag_uart_0
MogoCoro JTAG altera_ava	UART Ion_itag_uart
* Block Diagram	
res	ick = clk irq = interrupt set = reset on = avalon_itag_slave
👕 Write FIFO (Data fro	om Avalon to JTAG)
Buffer depth (bytes): IRQ threshold:	64 V 8 egisters instead of memory blocks
👕 Read FIFO (Data fro	om 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:	
	Cancel

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**.



ystem Contents System Generation						
omponent Library	Target	Clock Settings				
Project 🔼	Device Family: Cyclone IV E	Name	Source		MHz	Add
New component Library Avalon Verification Suite B-Bridges and Adapters		clk_50	External	50	0.0	Remove
interface Protocols interface Protocols	Use Conn Module	Descripti	on	Clock	Base	End
Ethernet	🔽 🗆 сри	Nios II Proc	essor	[clk]		1
⊞⊸High Speed ⊞⊸Interlaken ⊞⊸PCI	instruction_ma data_master jtag_debug_m	Avalon Me	mory Mapped Master mory Mapped Master mory Mapped Slave	clk_50 [clk] [clk]	IRQ (0x00000ff
SOI Serial Avalon-ST JTAG Avalon-ST Serial Avalon-ST Serial SPI (3 Wire Serial UART (RS-232 Sé Legacy Components	Image: Image and the second secon	JTAG UAR lave Avalon Me	T mory Mapped Slave	[cik] cik_50	a ⁰ 0x00000000	0x000000
New Edit Add	Remove Edit	• • 3	Address Map	Filters Fil	ter: Default	2
To Do: cpu: No exception vector has	in specified for this CPU. Please paramet is been specified for this CPU. Please para eption vector cannot be set until memory	ameterize the CPU to r	esolve this issue	n		

Figure 7-21 JTAG UART

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



system Contents System Generation						
Component Library	Target	Clock Settings				
Project	Device Family: Cyclone IV E	Name	Source	N	IHz	Add
Library -Avalon Verification Suite -Bridges and Adapters		clk_50	External	50	.0	Remove
Interface Protocols	Use Conn Module	Description	81	Clock	Base	End
er Ethernet er High Speed er Interlaken er PCI	C cpu instruction_max data_master jtag_debug_mo	Nios II Proces Avalon Memo Avalon Memo	1	[clk] clk_50 [clk] [clk]	0 90800000 x0	0x00000fff
- Serial	✓ jtag_uart	JTAG UART	Named Slave	[clk]	0-00000000	0-0000007
Avalon-ST JTAG Avalon-ST Serial Avalon-ST Serial SPI (3 Wire Serial UART (RS-232 Se Legacy Components	avalon_itag_sk	ave Avalon Memo	ory Mapped Slave	clk_50	a 0x 0000000	
Avalon-ST JTAG Avalon-ST Serial Avalon-ST Serial SPI (3 Wire Serial UART (RS-232 Se Legacy Components New, Edit Add		ave Avalon Memo	Address Map	cik_50	er: Default	0x0000007

Figure 7-22 Rename JTAG UART

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



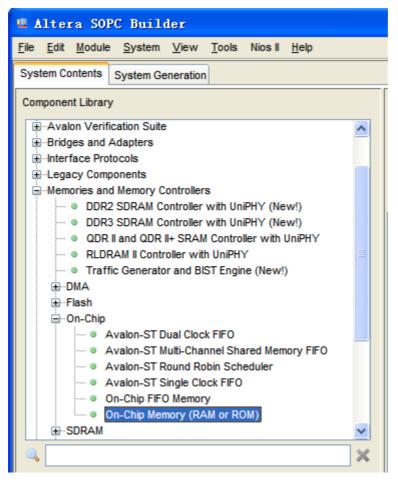


Figure 7-23 Add On-Chip Memory



🗏 On-Chip Memory ()	RAM or ROM) - onchip	_memory2_0	<
Mogecore On-Chip M attera_avalon_or	Memory (RAM or RO	M)	
Block Diagram			^
	clock Clk1 avalon s1 reset reset1	_0	
* 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 (usage (may impact fmax)		
TRead latency		1	
Slave s1 Latency:	1 🗸		
Slave s2 Latency:	1.0		
Memory initialization			
✓ Initialize memory content	+		
Market and the second second			~
Enable non default initia	lization 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**.



😃 On-Chip Hemory ()	RAM or ROM) -	onchip_m	emory2_0		×
Mogecore On-Chip N attera_avalon_or	/lemory (RAN nchip_memory2	l or ROM)	<u>D</u> ocumentation	
* Block Diagram					^
	clock Clk1 avaion Clk1 reset reset1	memory2_0			
* Memory type					
Туре:	RAM (Writable)	*			
Dual-port access					
Read During Write Mode:	DONT_CARE				
Block type:	Auto 👻				
* Size					
Data width:	32 👻		6		
Total memory size:	26000		bytes		
Minimize memory block	usage (may impact fr	nax)			
* Read latency					
Slave s1 Latency:	1 🗸				
Slave s2 Latency:	1 ~				<u></u>
Memory initialization					
Initialize memory conten	t				ine.
Enshle non default initia	lization file				Y
			C	ancel Finist	

Figure 7-25 Update Total memory size



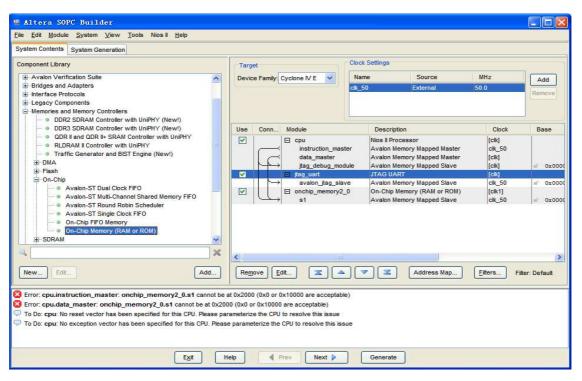


Figure 7-26 Add On-Chip memory

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

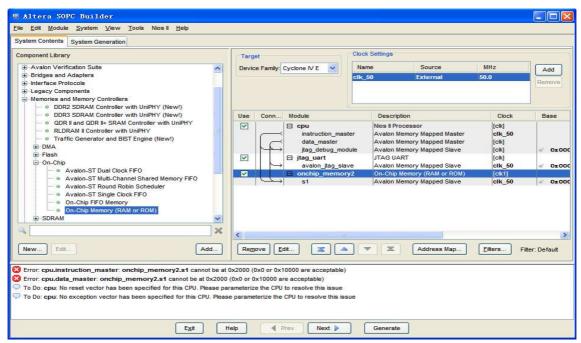


Figure 7-27 Rename On-Chip memory

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



L Nios II Proces	sor – cpu				
Nios	II Processor				About Documentation
Parameter Settings					
	es and Memory Interfaces $>$	Advanced Features > N	1MU and MPU Settings	JTAG Debug Module	e 🔪 Custom Instructions 🔪
Core Nios II					
Select a Nios II core:		1	1		120
	ONios II/e	○Nios II/s	●Nios II/f		<u>^</u>
Nios II Selector Guide Family: Cyclone IV E f _{system:} 50.0 MHz cpuid: 0	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Pr	rediction	ŧ
Performance at 50.0 MH:		Up to 32 DMIPS	Up to 57 DMIPS		_
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs		<u>×</u>
Hardware Multiply: Embe	edded Multipliers	Hardware Divide			
	nory: onchip_memory2	✓ Offset: 0x0		0x00002000	
Exception Vector: Mem	ory. onchip_memory2	Offset: 0x20		0x00002020	
Include MMU					
Only include the MMU wh	nen using an operating system t	hat explicitly supports an MMU	l.		
Fast TLB Miss Exception	Vector: Memory:	~	Offset: Ox0		
Include MPU					
				Cancel	< Back Hext > Finish

Figure 7-28 Update CPU settings



Legacy Components Memories and Memory Controllers ODR2 SDRAM Controller with UniPHY (New!) ODR3 SDRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller with UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I Controller WITH UniPHY ORI I and QDR I+ SRAM I CONTROL I A Avalon Memory Mapped Slave ORI I and QDR I+ SRAM I CONTROL I Avalon Memory Mapped Slave ORI I and QDR I+ SRAM I CONTROL I Avalon Memory Mapped Slave ORI A Avalon Memory Mapped Slave ORI A Aval							
Bridges and Adapters Add Interface Protocols Legacy Components Memories and Memory Controllers DDR3 SDRAM Controller with UniPHY (New!) ● DDR3 SDRAM Controller with UniPHY (New!) Use ● COR II and QDR II-SRAM Controller with UniPHY (New!) Use ● DDR3 SDRAM Controller with UniPHY (New!) Use ● COR II and QDR II-SRAM Controller with UniPHY Instruction_master Avaion Memory Mapped Master clk_50 ● Flash Con-Chip ● Avaion-ST Dual Clock FIFO JTAG UART ● Avaion-ST Round Robin Scheduler Image: Stave ● Avaion-ST Round Robin Scheduler State ● Avaion-ST Round Robin Scheduler State ● On-Chip FIFO Memory On-Chip FIFO Memory (RAM or ROM) ● SDRAM Image: State		3		-			1
Interface Protocols Legacy Components Memories and Memory Controllers • DDR2 SDRAM Controller with UniPHY (New!) • DDR3 SDRAM Controller with UniPHY • RLDRAM II Controller with UniPHY • Avalon-ST Dual Clock FIFO • Avalon-ST Multi-Channel Shared Memory FIFO • Avalon-ST Round Robin Scheduler • Avalon-ST Single Clock FIFO • On-Chip FIF		Device	Family: Cyclone IV E	A DESCRIPTION OF THE OWNER	and the second		Add
Memories and Memory Controllers	Interface Protocols			CIK_00	External	30.0	Remove
ODR2 SORAM Controller with UniPHY (New!) ODR3 SORAM Controller with UniPHY (New!) ODR3 SORAM Controller with UniPHY ORLIPAM II Controller with UniPHY ORLIPAM ORLIPAM II Controller with UniPHY ORLIPAM O							
Image: Signed constraint of the structure of the str		Use	Conn Module		Description	Clock	Base
							4
⊕ -DMA ⊕ -DMA ⊕ -Bash ⊡ On-Chip □ Avalon-ST Dual Clock FIFO □ Avalon-ST Round Robin Scheduler □ Avalon-ST Round Robin Scheduler □ On-Chip Memory (RAM or ROM) □ On-Chip Memory (RAM or ROM) ⊕ SDRAM		1			774.577		
⊕ -Flash □ - On-Chip	⊕ DMA		2 B B B B B B B B B B B B B B B B B B B				Or Or
Avalon-ST Dual Clock FIF0 Avalon-ST Multi-Channel Shared Memory FIF0 Avalon-ST Round Robin Scheduler Avalon-ST Round Robin Scheduler Avalon-ST Single Clock FIF0 On-Chip Memory (RAM or ROM)							
Avalon-ST Multi-Channel Shared Memory FIFO Avalon-ST Round Robin Scheduler Avalon-ST Single Clock FIFO On-Chip FIFO Memory On-Chip FIFO Memory On-Chip FIFO Memory On-Chip Memory (RAM or ROM)					25 dita	1.1	0x0
Avalon-ST Round Robin Scheduler Avalon-ST Single Clock FIFO On-Chip FIFO Memory On-Chip Memory (RAM or ROM) On-Chip Memory (RAM or ROM)			A DARS (0.770)	- C -			
tew Edit Add Remove Edit Address Map Eiters Fitter: Default	On-Chip FIFO Memory On-Chip Memory (RAM or ROM) On-Chip Memory (RAM or ROM)	<					
	New Edit Add	Remov	e Edit 🛣 🔺		Address Map	Filters Fi	tter: Default
Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x2000 (0x0 or 0x10000 are acceptable)							

Figure 7-29 Updated CPU settings

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



🖷 Altera SOPC Builder
<u>File Edit M</u> odule <u>System V</u> iew <u>T</u> ools Nios II <u>H</u> elp
System Contents System Generation
Component Library
Project
New component
Library
Avalon Verification Suite Bridges and Adapters
Druges and Adapters
Memories and Memory Controllers
Herlin Components
- Peripherals
⊡ ··Debug and Performance
E Display
FPGA Peripherals
Microcontroller Peripherals Interval Timer
PIO (Parallel VO)
Multiprocessor Coordination
Processor Additions
Processors
Q X
New Edit Add

Figure 7-30 Add PIO



🛢 PIO (Parallel	I/0) - pio_0	
MogesCore PIO (Pa attera_avalo	arallel I/O) n_pio	Documentation
* Block Diagram		^
avalon	reset	
* Basic Settings	WE WE	
Width (1-32 bits):	8	
Direction:	O Bidir	
	O Input	
	O InOut	
	 Output 	
Output Port Reset Value		_
	L	
Output Register		
Enable individual bi	setting/clearing	
👻 Edge capture regist	er	
Synchronously cap	ture	
Edge Type:	RISING M	
Enable bit-clearing	for edge capture register	
		~
	C	ancel Finish

Figure 7-31 Add PIO

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



tem Contents System Generation	1					
nponent Library	Targ	et	Clock Setting			
oject	Device	e Family: Cyclone IV E 🛛 👻	Name	Source	MHz	Add
- 🔯 New component brary			clk_50	External	50.0	Remove
-Avalon Verification Suite						I the second second
-Bridges and Adapters						
Interface Protocols						-
Legacy Components	Use	Conn Module	Des	cription	Clock	Base
Memories and Memory Controllers		е сри	Nios	l Processor	[clk]	
Merlin Components		instruction_mas		in Memory Mapped Master	clk_50	
Peripherals Debug and Performance		data_master		in Memory Mapped Master	[clk]	
Debug and Performance Display		itag_debug_mo		in Memory Mapped Slave	[clk]	= Ox
FPGA Peripherals		⊟ jtag_uart		UART	[clk]	
Microcontroller Peripherals		avalon_jtag_sla		n Memory Mapped Slave	clk_50	= Ox
Interval Timer		s1		hip Memory (RAM or ROM) In Memory Mapped Slave	[clk1] clk 50	= Ox
PIO (Parallel I/O)		E pio_0		Parallel I/O)	[clk]	in Ox
Multiprocessor Coordination				in Memory Mapped Slave	clk_50	e Ox
PLL			prevenue	in memory inapped clare		1- 04
Processor Additions						
Processors	~					
	X					
		1.00].		
The second se	I Rem	ove Edit 🕱 🔺		Address Map	Filters	Filter: Default
ew Edit Add					(- meren) ·	ator. Doruon

Figure 7-32 PIO

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

omponent Library	Targ	jet	Clock Settings			
Project Mew component Library ⊕ Avalon Verification Suite ⊕ Bridges and Adapters	Devi	ce Family: Cyclone IV E	Name clk_50	Source External	MHz 50.0	Add Remove
Interface Protocols E-Legacy Components	Use	Conn Module	Descri	iption	Clock	Base
Memories and Memory Controllers Metrin Components Peripherals Peripherals ProcA Peripherals Microcontroller Peripherals Microcontroller Peripherals Plutiprocessor Coordination PLL Processor Additions Processors		 ☐ cpu instruction_master jtag_debug_mod ☐ jtag_debug_mod ☐ jtag_uart avalon_jtag_sla ☐ onchip_memor s1 ☐ pio_led > s1 	ter Avalon Avalon Jule Avalon JTAG U. ve Avalon y2 On-Chip Avalon PIO (Par	rocessor Memory Mapped Master Memory Mapped Slave ART Memory Mapped Slave Memory (RAM or ROM) Memory Mapped Slave rallel VO) Memory Mapped Slave	[ck] clk_50 [ck] [ck] clk] clk_50 [ck1] clk_50 [ck] clk_50	11 Ox04 12 Ox04 11 Ox04 11 Ox04
	×					
New Edit Add	Re <u>r</u>	gove		Address Map	<u>Filters</u> Fi	iter: Default
Error: cpu.instruction_master: onchip_memory2.s1 cannot i Error: cpu.data_master: onchip_memory2.s1 cannot be at 0x						

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**.

Component Library Insert Avalon-ST Adapters Project Show Transformed System Bridges and Adapters Device Family: Cyclone IV E B-ridges and Adapters External Show Torins formed System Device Family: Cyclone IV E B-ridges and Adapters External B-ndiges and Adapters External B-memory controllers Use Conn Module Description Clock Base Immersion Memory Controllers Instruction_master B-Debug and Performance Instruction_master B-Debug and Performance Itag_debug_module B-Debug and Performance Itag_uart B-Debug and Performance Itag_uart B-Debug and Performance Itag_uart B-Memory Mapped Slave Ick; 50 B-Microcontroller Peripherals Onchip_memory2 B-Multprocessor Coordination Itag_uart B-Multprocessor Additions S1 Avalon Memory Mapped Slave Itk_50 B-Processors S1	System Contents Auto-Assign Base Addresses Auto-Assign IRQs								
Show Transformed System Add Ibrary Add Bridges and Adapters Interface Protocols Interface Protocols External Breidges and Adapters Interface Protocols Interface Protocols Equation (Ikk) Breidges and Adapters Interface Protocols Interface Protocols Equation (Ikk) Breidges and Adapters Instruction_master Avaion Memory Controllers Instruction_master Avaion Memory Mapped Master Ick, 50 Importantian Instruction_master Avaion Memory Mapped Slave Ick] Importantian Instruction_master Avaion Memory Mapped Slave Ick] Importantian Instruction_master Avaion Memory Mapped Slave Ick] Importantian Interval Timer Important Ick] Interval Timer Important Importantian Important Importantian Important Importantian Important Importantian Important Importantian Importantian Importantian <td< th=""><th>Component Library</th><th></th><th>Target</th><th></th><th colspan="5">Clock Settings</th></td<>	Component Library		Target		Clock Settings				
Library		^	Device Family	Cyclone IV E 🔽	1100000000			Add	
Use Con Module Description Clock Base Memories and Memory Controllers Merrin Components Peripherals Display FPGA Peripherals Merrin Controller Service Polipharals Metrin Controller Service Status Controller Service Status Controller Service Status Controller Service Metrin Controller Service Status Controller Service Status Controller Service Status Controller Service Status Controller Service Metrin Controller Service Status Controller Service Status Controller Service Status Controller Service Stat	Library -Avalon Verification Suite -Bridges and Adapters					EATCHING	30.0	Remove	
Merlin Components Peripherals Per			Use Conn	Module	D	lescription	Clock	Base	
	Herlin Components Peripherals Display FPGA Peripherals Hicrocontroller Peripherals Interval Timer Pionents(VO) Hutiprocessor Coordination PLL Processor Additions	×		✓ instruction_mass data_master itag_debug_mo E jtag_uart avalon_tag_sla E onchip_memoi s1	ster Av Av dule Av JT/ ave Av ry2 On Av	alon Memory Mapped Master raion Memory Mapped Master raion Memory Mapped Slave AG UART raion Memory Mapped Slave -Chip Memory (RAM or ROM) raion Memory Mapped Slave O (Parallel VO)	cik_50 [cik] [cik] [cik] cik_50 [cik1] cik_50 [cik]	a ¹ 0x00	
New Edit Z Address Map Eilters Filter: Defau	New Edit					Address Map	Eitters F	ilter: Default	

Figure 7-34 Auto-Assign Base Addresses

stem Contents System Generation					
omponent Library	Target	Clock Settings			
Project	Device Family: Cyclone IV E	Vame	Source	MHz	Add
New component ibrary Avalon Verification Suite Bridges and Adapters		clk_50	External	50.0	Remove
Interface Protocols Legacy Components	Use Conn Module	Descrip	otion	Clock	Base
Memories and Memory Controllers DR2 SDRAM Controller with Un DDR3 SDRAM Controller with Un QDR II and QDR II+ SRAM Contro RLDRAM II Controller with UniPH' Traftic Generator and BIST Engir DMA Flash On-Chip SDRAM SRAM Merin Components	 ✓ ✓	ter Avalon M g_module Avalon M JTAG UA g_slave Avalon M emory2 On-Chip Avalon M PIO (Pard	demory Mapped Master demory Mapped Master demory Mapped Slave ART Memory Mapped Slave Memory (RAM or ROM) demory Mapped Slave	[cik] clk_50 [cik] [cik] clk_50 [cik1] clk_50 [cik] clk_50 [cik]	IRG © 0x0001080 © 0x0001010 © 0x0000800 © 0x00001000
New Edit Add	Remove Edit		Address Map	Eitters Fit	ter: Default
Warning: cpu: Custom Instruction components c Warning: cpu: Disabling the assign CPUID contro			control register value. This	option will always	s be turned on with d

Figure 7-35 No errors or warnings



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

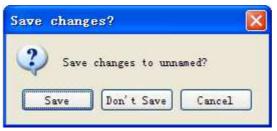


Figure 7-36 Generate SOPC

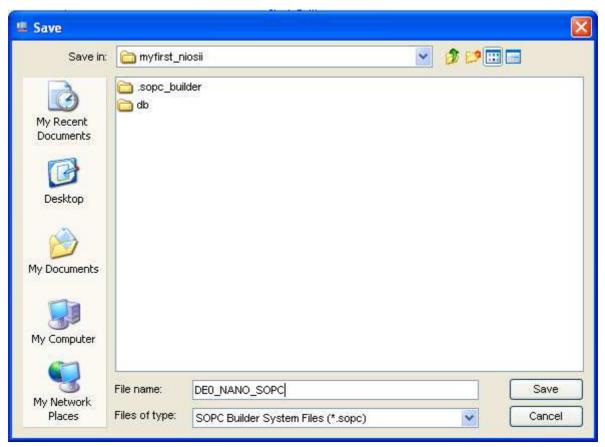


Figure 7-37 Generate SOPC



# Altera SOPC Builder - DEO_NANO_SOPC.sopc (D:\myfirst_niosii\DEO_NANO_SOPC_sopc)	
File Edit Module System View Tools Nios II Help	
System Contents System Generation	
Options	
System module logic will be created in Verilog.	
Simulation. Create project simulator files. Run Simulator	
Sindadon, Geale project sindado nes,	
Nios II Toois	
Nios II Software Build Tools for Eclipse	
Nuos il suitavare buila Tuois for Eclapse	
	~
# 2011.02.24 13:23:09 (*) Generating Quartus symbol for top level: DE0_NANO_SOPC # 2011.02.24 13:23:09 (*) Generating Symbol D:/mvfirst_niosii/DE0_NANO_SOPC.bsf	
# 2011.02.24 13:23:09 (*) Generating Symbol D.mynist_husibubb _vaNo_SOPC.ds1 # 2011.02.24 13:23:09 (*) Creating command-line system-generation script	
# 2011.02.24 13:23:09 (*) Running setup for HDL simulator: modelsim # 2011.02.24 13:23:10 (*) Completed generation for system: DEO_NANO_SOPC.	
# 2011.02.24 13:23:10 (*) Completed generation for system. Deo_NainO_SOPC. # 2011.02.24 13:23:10 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:	
SOPC Builder database : D/myfirst_niosii/DE0_NANO_SOPC.ptf	
Sore Builder database . D./myfirst_niosi/DE0_NANO_SOPC.ptf	
System Fibe worder. During installiosit/DE0_NANO_SOPC_generation_script	(3)
# 2011.02.24 13:23:10 (*) SUCCESS: SYSTEM GENERATION COMPLETED.	
	>
A Warning: cpur Custom Instruction components can be edited through the Component Editor.	
A 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 defa
	>
Exit Help Very Next Generate	

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.

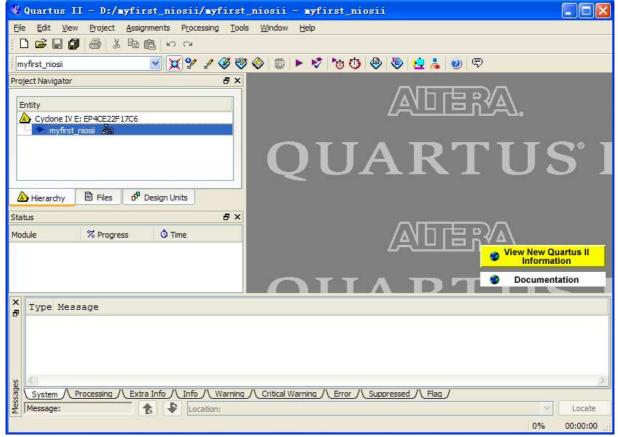


Figure 7-39 Return to Quartus II after exiting SOPC Builder



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**.

-	Quartu	ıs II	- D:/	yfirst_n	iosii/my
Eile	<u>E</u> dit	<u>V</u> iew	Project	<u>A</u> ssignments	Processing
	<u>N</u> ew			Ctrl+N	×
Ē	Open			Ctrl+O	00
	Close			Ctrl+F4	
溋	New Pro	oject <u>W</u> iz	ard		
Ê	Open Pr	oject		Ctrl+J	
	Save Pr	ojec <u>t</u>			
	Clos <u>e</u> Pr	oject			
	Save			Ctrl+S	
	Save <u>A</u> s	·			
ø	Save All			Ctrl+Shift+	s
	<u>F</u> ile Prop	perties			
	Create	Update			•
	Export.				
	Convert	Prograg	<u>m</u> ming File	s	
ц,	Page Se	typ			
à	Print Pre	e <u>v</u> iew			
8	Print			Ctrl+P	
	Recent	Files			•
	Recent	Projects			•
	E <u>x</u> it			Alt+F4	

Figure 7-40 New Verilog file

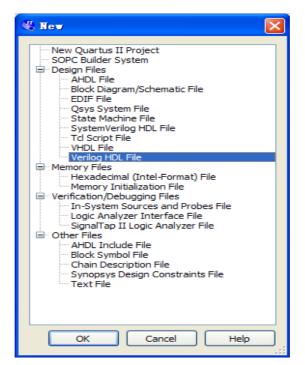


Figure 7-41 New Verilog File



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

Quartus II - D:/myfirst_niosii/myfirst Ele Edit View Project Assignments Processing Too D 2	ls <u>Window H</u> elp
myfirst_niosii 🛛 📝 🖓 🧳 🤴	
Project Navigator 🛛 🕹 🗙	
Entity	· 🗃 🛤 🅼 🕡 律律 🔺 🌤 🌤 🌾 🖉 🖾 💩 💳 🖻
Cyclone IV E: EP4CE22F17C6 Total model Module Cyclone IV E: EP4CE22F17C6 Module Cyclone IV E: EP4CE27 Module Module Cyclone IV E: EP4CE27 Module Cyclone IV E: EP4CE27	1
Type Message	
System / Processing // Extra Info // Info // Warning	/\ Critical Warning /\ Error /\ Suppressed /\ Flag /
System Processing Extra Info Info Warning Message:	Critical Warning / Error / Suppressed / Flag /

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

terasic DE0-Nano User Manual



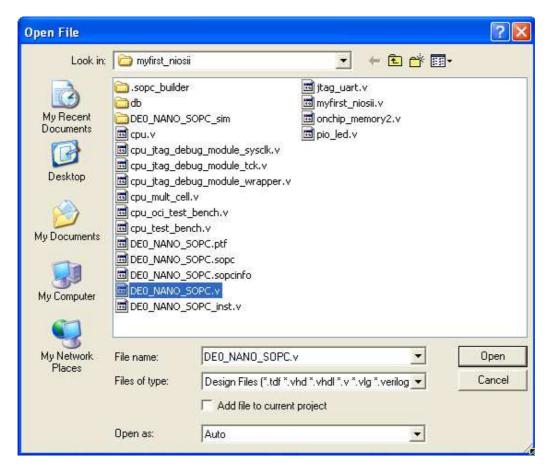


Figure 7-44 and Figure 7-45.

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

endmodule



🖑 Quartus II - D:/myfirst_niosii/myf	first_niosii - myfirst_niosii 📃 🗖 🔀
Eile Edit View Project Assignments Processing	Tools <u>W</u> indow <u>H</u> elp
myfirst_niosii 💽 🔀 💅 🦉	🥙 🕸 💿 🕨 💐 🍓 🕘 😫 🕹 🔍 🛡
Project Navigator 🛛 🗗 🗙	nyfirst_niosii.v 🛛
Entity	圖 桷 編 7 準 ∉ ▲ ≫ ≫ ≫ ▲ ◎ 🗂 🗟 📾 🖦 ⇒ ≡ 🦉
Cyclone IV E: EP4CE22F17C6	<pre>32 // Major Functions: myfirst_niosii 33 // Major Functions: myfirst_niosii 34 // 35 //</pre>
8	56 -enamodule 57 -
Hierarchy 🖹 Files 🗗 Design Units	< >
	0% 00:00:00

Figure 7-43 Input verilog Text

Open File					? 🛛
Look in: My Recent Documents Desktop My Documents My Computer	🔄 cpu_jtag_deb	OPC_sim ug_module_sysclk.v ug_module_tck.v ug_module_wrapper.v .v bench.v ch.v oPC.ptf OPC.sopc OPC.sopcinfo OPC.v	itag_uart	niosii.v nemory2.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 *.	▼ .vlg *.verilog ▼	Open Cancel

Figure 7-44 Open DE0_NANO_SOPC.v

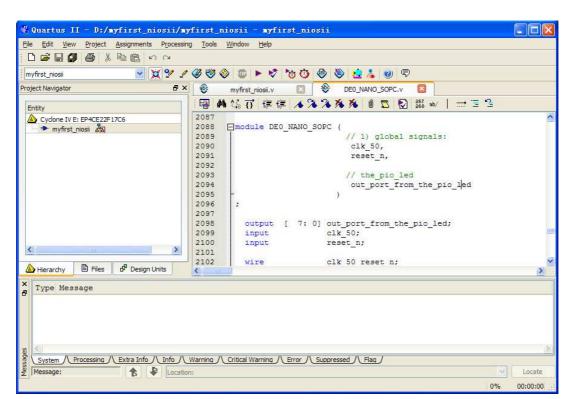


Figure 7-45 DE0_NANO_SOPC module

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

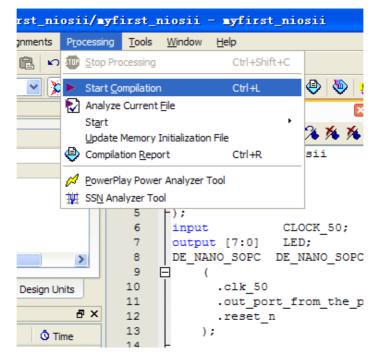
Save As					? 🔀
Save jn: My Recent Documents Desktop My Documents My Computer	👼 cpu_jtag_deb	OPC_sim ug_module_sysclk.v ug_module_tck.v ug_module_wrapper.v .v bench.v ch.v OPC.v OPC_inst.v		•	
	File <u>n</u> ame: Save as <u>t</u> ype:	Verilog HDL Files (*.	•		<u>à</u> ave ancel

Figure 7-46 Save the Verilog file

正じ



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





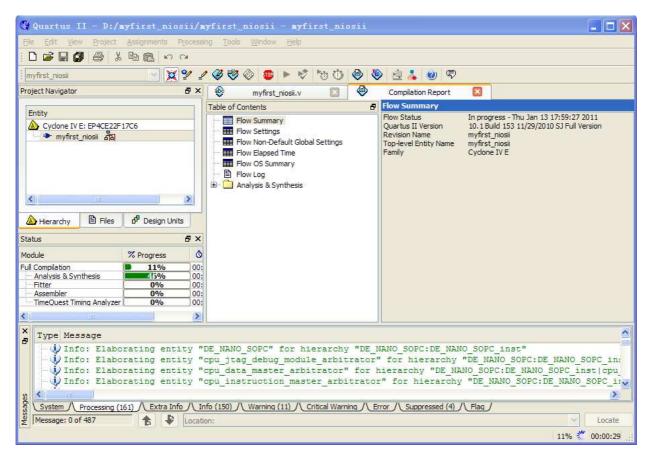
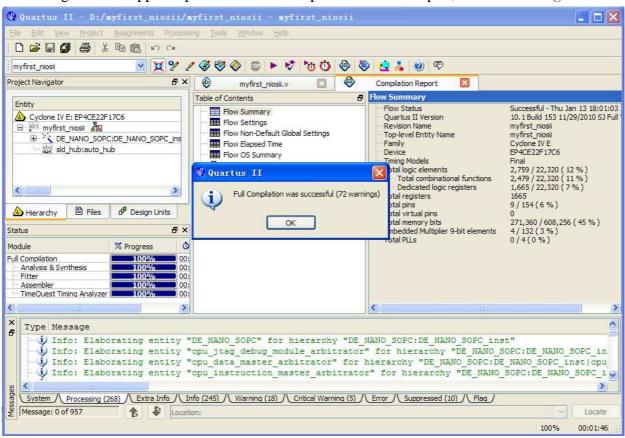


Figure 7-48 Execute Compile





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

Figure 7-49 Compile project completely

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

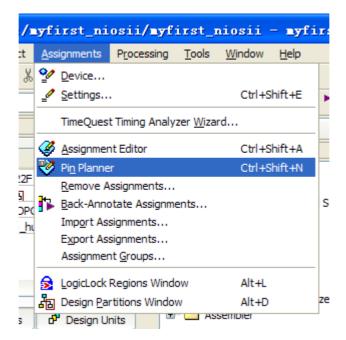


Figure 7-50 Pins menu



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

Figure 7-51 Blank Pins

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

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

Figure 7-52 Set Pins

40. Close the pin planner and recompile the project.

7.3 Download the Hardware Design

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

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

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



7. Click Change File.

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

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

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

Ð	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: USB-Blaster [USB-0] Available hardware items No Hardware						
		No Hardware USB-Blaster [USB-0]					
	Hardware	Server	Port	Add Hardware			
	USB-Blaster	Local	USB-0	Remove Hardware			
				Close			

Figure 7-53 Hardware Setup Window

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



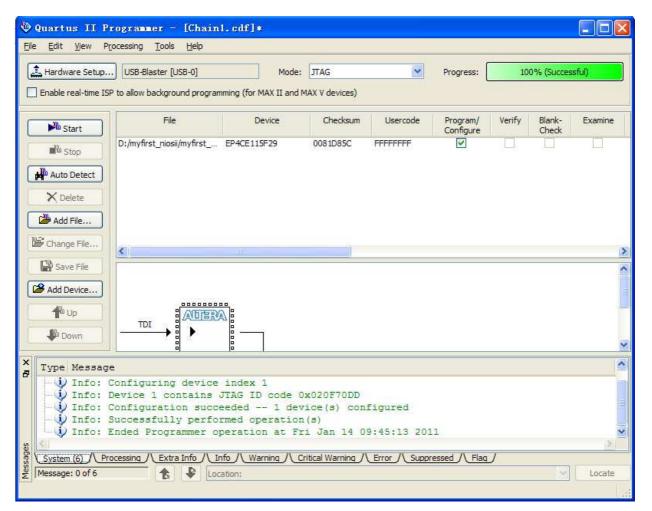


Figure 7-54 Quartus II Programmer

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

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

7.4 Create a hello_world Example Project

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

1. Return to the NIOS II IDE.

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

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

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



New Project

Times replect	
Nios II C/C++ Application Click Finish to create applicati D:\myfirst_niosii\Software\hello	on with a default system library as p_world_0
Name: hello_world_0 Name: hello_world_0 Specify Location Location: D:\myfirst_niosii\S Select Target Hardware. SOPC Builder System PTF File: CPU:	oftware Browse D:\myfirst_niosii\DE_NANO_SOPC.ptf Browse cpu
Select Project Template Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Memory Test Small Simple Socket Server Simple Socket Server Web Server Web Server (RGMII)	Description Prints 'Hello from Nios II' Details Mello 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>F</u> r	oject Tools <u>R</u> un <u>W</u> indow	Help				
i 📬 • 🔜 👜 💼 i 🎯 • 🚳 • 👩 • (3 • 🏘 • Q • Q •	1 🙆 🛷 1 G	s• ∲ • {	• � ⊕ • e	D 🖹 📑 Ni	os II C/C++
💽 Nios II C/C++ Projects 🖾 🖳 🗖	🖸 hello_world c 🗙					- 8
Constraints Const	<pre>* "Hello World" * This example * the Nios II ' * designs. It r * device in you * The memory fo * using the sta * * For a reduced</pre>	prints 'Hell standard', uns with or r system's 1 otprint of 1 ndard refere footprint v memory foot world" temp1 h>	full_featu without th hardware. this hosted ence design version of oprint for late.	red', 'fast' e MicroC/OS- application this templat	STDOUT stream. It runs on , and 'low_cost' example II RTOS and requires a STDOUT is ~69 kbytes by default e, and an explanation of how ication, see the	
	E.					V
						* □
	Problems 🔀 Consol O errors, O warnings, O i	e des concernents en la concernent				
	Description *		Resource	Path	Location	
<u>×</u> ×						
: • *		Writable	Smart Inse	ert 1:1		

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

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

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

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

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

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

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

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

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

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

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

7.5 Build and Run the Program

In this section you will build and run the program.

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

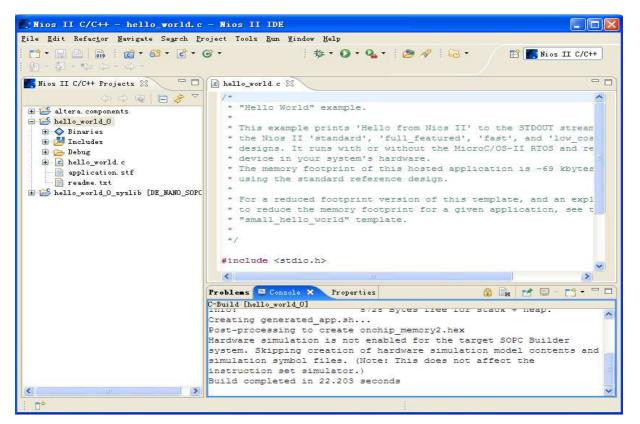


Figure 7-57 Nios II IDE hello_world_0 Build Completed

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



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

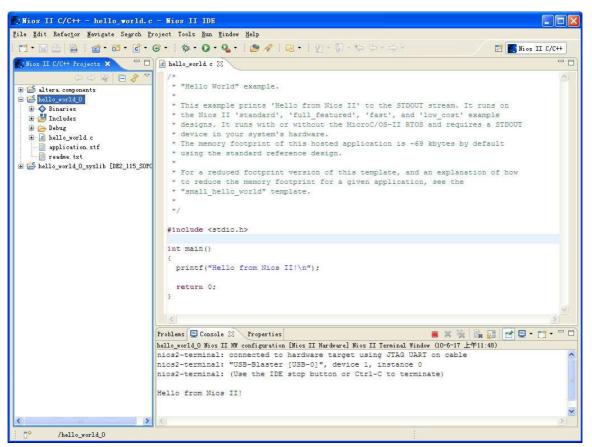


Figure 7-58 Hello_World_0 Program Output

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

7.6 Edit and Re-Run the Program

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

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

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

#include <stdio.h>

terasic DE0-Nano User Manual



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

2. Save the project.

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

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



7.7 Why the LED Blinks

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

Nios II C/C++ - system.h - File Edit Refactor Navigate Search		
		II C/C++
Nios II C/C++ P 🛛 🦳 🗖	🖻 hello_world.c 🛛 🖻 system.h 🕱	- 0
<pre></pre>	<pre>/* * pio_led configuration * * #define FIO_LED_NAME "/dev/pio_led" #define FIO_LED_TYPE "altera_avalon_pio" #define FIO_LED_TYPE "altera_avalon_pio" #define FIO_LED_BASE 0x00011000 #define FIO_LED_DO TEST_BENCH_WIRING 0 #define FIO_LED_DO TEST_BENCH_WIRING 0 #define FIO_LED_DOTEST_BENCH_WIRING 0 #define FIO_LED_DAS_TRI 0 #define FIO_LED_HAS_TRI 0 #define FIO_LED_HAS_TRI 0 #define FIO_LED_HAS_TRI 0 #define FIO_LED_CAPTURE 0 #define FIO_LED_CAPTURE 0 #define FIO_LED_CAPTURE 0 #define FIO_LED_EDET_TYPE "NONE" #define FIO_LED_EDET_TYPE "NONE" #define FIO_LED_BIT_CLEARING_EDEE_REGISTER 0 #define FIO_LED_BIT_MODIFYING_OUTPUT_REGISTER 0 #define FIO_LED_FREQ 5000000 #define ALT_MODULE_CLASS_PIO_led_altera_avalon_PIO</pre>	
	Problems 😑 Console X Properties 🔐 😭 🔛 -	<u>•</u> • • •
readme.txt		^
	Creating generated.x Build completed in 159.609 seconds	~
- - *		

Figure 7-59 The system.h file

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

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

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

IOWR_ALTERA_AVALON_PIO_DATA(base, data)



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

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

7.8 Debugging the Application

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

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

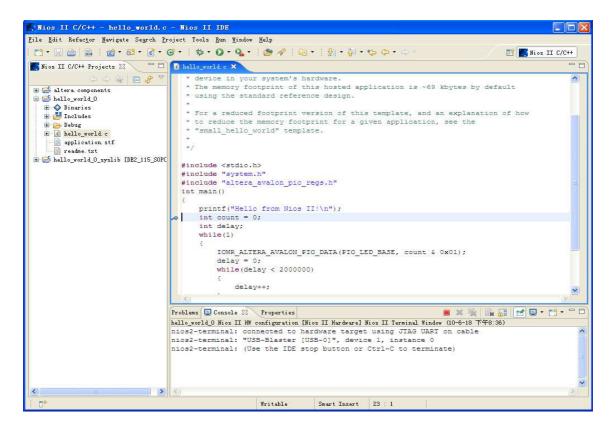


Figure 7-60 Set Breakpoint

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

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

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

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

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

7.9 Configure System Library

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

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

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

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

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



type filter text	System Library				φ×φ.
Info - Builders - C/C++ Build - C/C++ Fuild Types - C/C++ Include Father - C/C++ Include Father - C/C++ Make Froject - C/C++ Aroject References - Refactoring Mistory - System Library	CPU: cpu System Library Contents	sii\DE_MANO_SOFC ptf	~	Linker Script OCustom linker script	Browse
	RTOS: RTOS Options are stdout: stderr: stdin: System clock timer: Timestamp timer: Max file descriptors: Program never exits Support C++ Lightweight device driver API Link with profiling library Unimplemented instruction handler Software Components	jtag_uart jtag_uart jtag_uart none none 22 Cleen exit (flush buffers) Reduced device drivers Small C library ModelSim only, no hardware su		none Duse auto-generated linker script Program memory (text): Read-only data memory (rodata): Read/write data memory (rodata): Heap memory: Stack memory: Use a separate exception stack Exception stack memory: Maximum exception stack size (bytes):	Select onchip_memory2 V onchip_memory2 V onchip_memory2 V onchip_memory2 V onchip_memory2 V
< <u> </u>				Help Restore Defe	ults Apply

Figure 7-61 Configuring System Library Properties

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

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



Chapter 8

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.

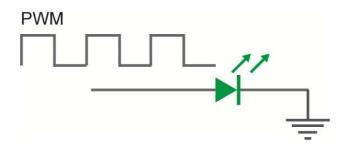
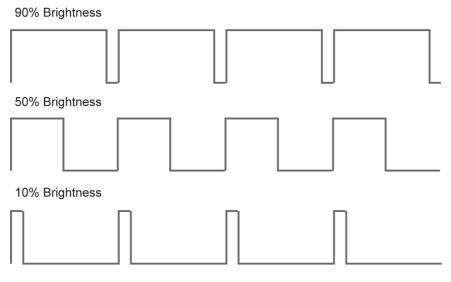


Figure 8-1 Shows a diagram of PWM signals to drive LED.







Pulse Width Modulation

Figure 8-2 Pulse Width Modulation

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

Demonstration Source Code

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

Demonstration Batch File

Demo Batch File Folder: DE0_NANO_Default\demo_batch

The demo batch file includes the following files:

• FPGA Configure File: DE0_NANO.sof

Demonstration Setup

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



8.3 ADC Reading

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

Design Concept

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

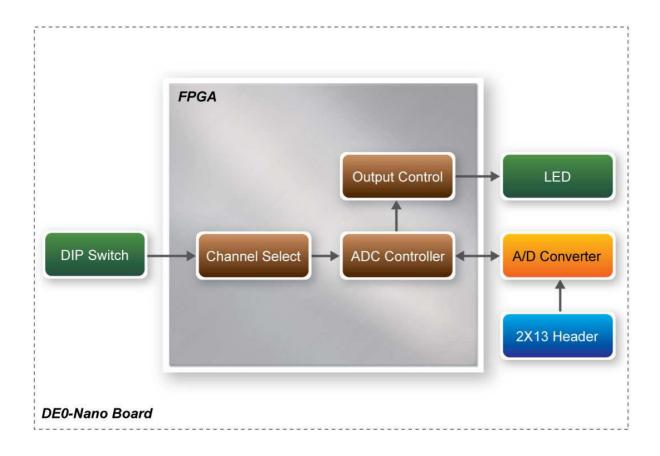


Figure 8-3 ADC Reading Block Diagram

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



Table 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).



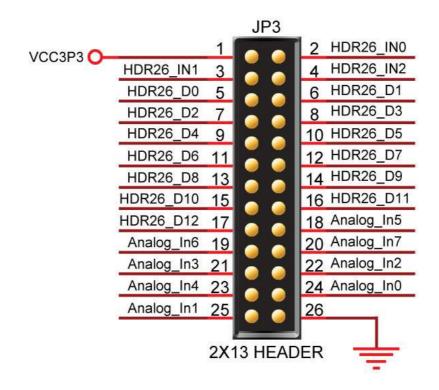


Figure 8-4 2X13 Header

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.



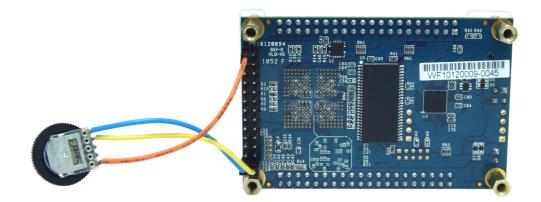


Figure 8-5 ADC Reading hardware setup

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 DEO-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.

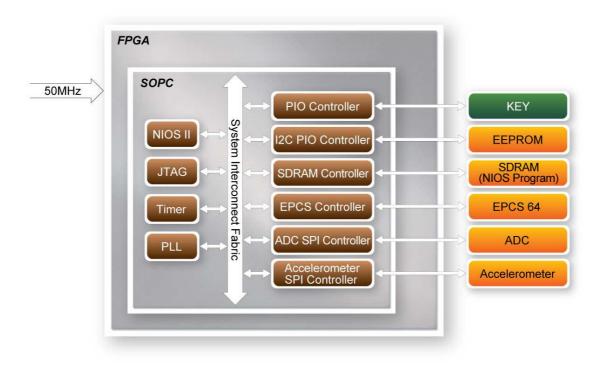


Figure 8-6 SOPC Block Diagram

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**.

logaCore altera_avalo	arallel I/O)	ntation
Basic Settings		
Width (1-32 bits):	2	
Direction:	O Bidir	
	💿 Input	
	🔿 InOut	
	🔿 Output	Í
Output Port Reset Valu	e: 0×00000000000000000000000000000000000	
Output Register		
Enable individual b	it setting/clearing	
Edge capture regis	ter	
Synchronously ca	pture	
Edge Type:	FALLING	
Enable bit-clearing	for edge capture register	
^r Interrupt		
Interrupt ✔ Generate IRQ		

Figure 8-7 PIO Controller

Accelerometer Control

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

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

Register 0x31—DATA_FORMAT (Read/Write)								
D7	D6	D5	D4	D3	D2	D1	DO	

2.2				10 C	· · · · · · · · · · · · · · · · · · ·		
Î	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, $\pm 2g$ mode. The output data rate is configured as 400 HZ. The X/Y/Z value is read using polling mode. Before reading X/Y/Z, the master needs to make sure data is ready by reading the register 0x30-INT_SOURCE, as shown below **Figure 8-9**, and checking the DATA_READY bit. In the demo, multiple-byte read of six bytes X/Y/Z, register from 0x32 to 0x37, is performed to prevent a change in data between reads of sequential register. Note, the output data is twos complement with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z.

Register 0x30—INT_SOURCE (Read Only)						
D7	D6	D5	D4			
DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity			
D3	D2	D1	DO			
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**:

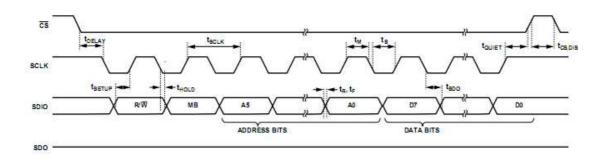


Figure 8-10 3-wire SPI Timing Diagram

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.



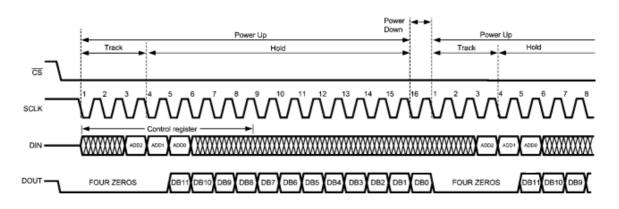


Figure 8-11 4-wire SPI Timing Diagram

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

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

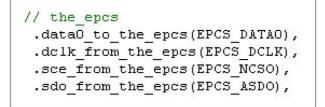


Figure 8-12 EPCS interface connection

For the EPCS access functions, users can refer to:

 $DE0_NANO_SOPC_DEMO\software\DE0_NANO\terasic_lib\Flash.c$



Demonstration Source Code

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

Demonstration Batch File

• Demo Batch File Folder: DE0_NANO_SOPC_DEMO\demo_batch

The demo batch file includes the file:

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

Demonstration Setup

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

🖾 Nios II EDS 10.1 [gcc3]	- 🗆 X
Example designs can be found in /cygdrive/c/altera/10.0/nios2eds/examples	^
(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: - EØIACCELEROMETER - E1IADC - E2IEEPROM - E3IEPCS	
Select:	-

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





Nios II EDS 10.1 [gcc3]	×
elect:Demo ACCELEROMETER	▲
d=E5h	
onitor Accerometer Value. Press KEY0 or KEY1 to terminal the monitor process.	
=-20 мд, Y=-4 мд, Z=872 мд	
=-32 mg, Y=8 mg, Z=976 mg	
=-20 mg, Y=8 mg, Z=956 mg	
=-20 mg, Y=4 mg, Z=980 mg	
=12 mg, Y=12 mg, Z=1004 mg	
=36 mg, Y=−8 mg, Z=972 mg	
=−32 mg, Y=8 mg, Z=968 mg	_
=-28 mg, Y=8 mg, Z=980 mg	•

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

🖎 Nios II EDS 10.1 [gcc3]	- 🗆 ×
Select:Demo ADC	^
Monitor ADC Value. Press KEYO or KEY1 to terminal the monitor process.	
CH1=0.29 V	
CH2=0.33 V	
CH3=0.37 V	
CH4=0.39 U	
CH5=0.40 V	
CH6=0.23 U	
CH7=0.32 V	

• 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	
Addr[00] = ffh	
Addr[01] = ffh	
Addr[02] = ffh	
Addr[03] = ffh	
Addr[04] = ffh	
Addr[05] = ffh	
Addr[06] = ffh	
Addr[07] = ffh	
Addr[08] = ffh	
Addr[09] = ffh	
Addr[10] = ffh	
Addr[11] = ffh	
Addr[12] = ffh	
Addr[13] = ffh	
Addr[14] = ffh	
Addr[15] = ffh	•

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





8.5 G-Sensor

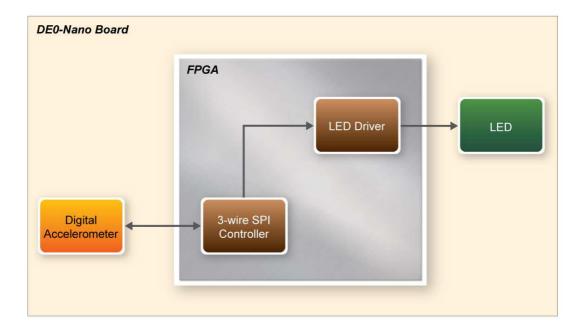
This demonstration illustrates how to use the digital accelerometer on the DEO-Nano board to measure the static acceleration of gravity in tilt-sensing applications. As the board is tilted from left to right and right to left, the digital accelerometer detects the tilting movement and displays it on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

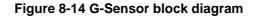


Figure 8-13 DE0-Nano on level surface

Design Concept

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







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

Demonstration Source Code

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

Demonstration Batch File

Demo Batch File Folder: DE0_NANO_GSensor\demo_batch

The demo batch file includes the following files:

• FPGA Configure File: DE0_NANO_G_Sensor.sof

Demonstration Setup

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

8.6 SDRAM Test by Nios II

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

System Block Diagram

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



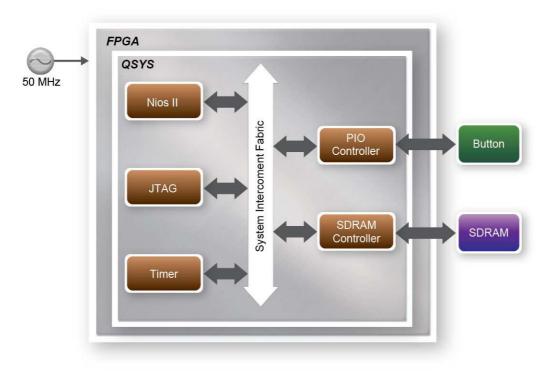


Figure 8-15 Block diagram of the SDRAM Basic Demonstration

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

Design Tools

- Quartus II 13.0 SP1
- Nios II Eclipse 13.0 SP1

Demonstration Source Code

- Quartus Project directory: DE0_NANO_SDRAM_Nios_Test
- Nios II Eclipse: DE0_NANO_SDRAM_Nios_Test \Software

■ Nios II Project Compilation

• Before you attempt to compile the reference design under Nios II Eclipse, make sure the project is cleaned first by clicking 'Clean' from the 'Project' menu of Nios II Eclipse.



Demonstration Batch File

Demo Batch File Folder: DE0_NANO_SDRAM_Nios_Test \demo_batch

The demo batch file includes following files:

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

Demonstration Setup

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

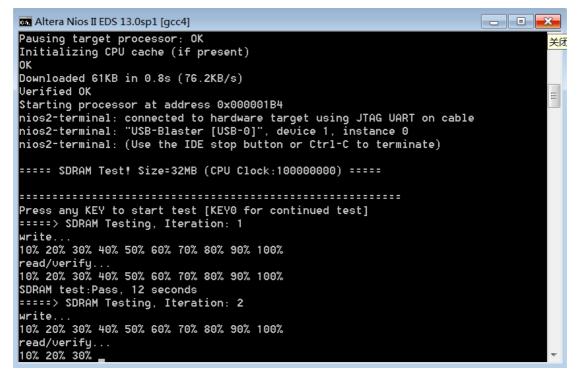
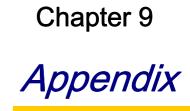


Figure 8-16 Display Progress and Result Information for the SDRAM Demonstration





9.1 Programming the Serial Configuration Device

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

Convert SOF to JIC

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

terasic Terasic DE0-Nano User Manual



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

le <u>W</u> indow pecify the input files to (ronvert and the type o	forogramming	file to cener	rate		
ou can also import input uture use.	file information from of	ther files and sa	ave the con	version se	tup information cre	ated here for
Conversion setup files	version Setup Data		ſ	Sava	Conversion Setup	1
	iversion setup bata		<u></u>	2010	conversion becap	
Output programming file	2					
Programming file type:	JTAG Indirect Configu	ration File (.jic))			~
Options	<u>Configuration device</u> :	EPCS64	~	Mode:	Active Serial	~
File <u>n</u> ame:	output_file.jic					
Advanced	Remote/Local update	difference file:	NONE			×
	Memory Map File					
Input files to convert						
File/Dat	a area	Pro	perties	J	Start Address	Add <u>H</u> ex Data
Flash Loader SOF Data		Page_0		<	auto>	Add Sof Page
ACCORDING TO A						
						Add Eile
						Remove
						Цр
						Down
						Properties
				Genera	ate Close	Help
				Genere	IL LIUSE	1 Meip

Figure 9-1 Convert Programming Files Dialog Box



🔓 Convert Progra	amming File -	D:/CD/DE_Nano/DE	0_Nano_v1.0.9_CDF	w 🔳 🗖 🔀
<u>Fi</u> le <u>W</u> indow				
Specify the input files to o You can also import input future use. Conversion setup files	convert and the type o file information from o	f programming file to gener ther files and save the conv	ate. /ersion setup information cr	eated here for
Open Cor	version Setup Data		Save Conversion Setup	
Output programming file	e			
Programming file type:	JTAG Indirect Configu	iration File (.jic)		~
Options	Configuration device:	EPCS64	Mode: Active Serial	<u>v</u>]
File <u>n</u> ame:	output_file.jic			
Advanced	Remote/Local update	difference file: NONE		*
Input files to convert				
File/Dat	a area	Properties	Start Address	Add <u>H</u> ex Data
Flash Loader		Page_0 P4CE22F17	<auto></auto>	Add Sof Page Add Device Remove
			<u>G</u> enerate Close	Properties 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 Device name APEX20K FP4CE10 Arria GX FP4CE115 Arria II GZ FP4CE10 Cyclone FP4CE15 Cyclone II FP4CE20 Cyclone III FP4CE55 Cyclone IV E FP4CE66 Cyclone IV GX FP4CE75 MAX V FP4CE75 Stratix II Stratix II GX Stratix II GX Stratix II GX Stratix IV Stratix IV	🖗 Select Devices		X
OK Cancel	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 III	□ EP4CE 10 New □ EP4CE 115 Import □ EP4CE 22 Export □ EP4CE 40 Edit □ EP4CE 55 Edit □ EP4CE 75 Uncheck A	

Figure 9-3 Select Devices Page



Convert Progra	amming File -	D:/CD/DE_	Nano/DE	iO_Nano	_ v1.0.9_CDR	🔳 🗖 🔀
<u>File W</u> indow						
Specify the input files to a You can also import input future use. Conversion setup files	convert and the type o file information from o	f programming ther files and sa	file to gene ive the con	erate. Iversion se	tup information crea	ated here for
Open Cor	version Setup Data			<u>S</u> ave	Conversion Setup.	
Output programming file	e j					
Programming file type:	JTAG Indirect Configu	uration File (.jic)	È			×
Options	<u>Configuration device</u> :	EPCS64	*	Mode:	Active Serial	×
File name:	output_file.jic					
Advanced	Remote/Local update	difference file:	NONE			×
Input files to convert						
File/Dat	a area	Prop	perties		Start Address	Add <u>H</u> ex Data
□ Flash Loader EP4CE22 □ SOF Data DE0_NANO.so		Page_0 EP4CE22F17		<	auto>	Add Sof Page Add Device <u>R</u> emove Up Down Properties
				Genera	ate Close	Help

Figure 9-4 Convert Programming Files Page



Convert Progra	amming File - 1	D:/CD/DE_Wa	no/DEO_Wand		0
<u>Eile Window</u>					
Specify the input files to a You can also import input future use. Conversion setup files	convert and the type o file information from ot	f programming file her files and save	to generate. the conversion se	etup information cre	ated here for
Open Cor	version Setup Data		Save	Conversion Setup.	
Output programming file	•				
Programming file type:	JTAG Indirect Configu	ration File (.jic)			~
Options	Configuration device:	EPCS64	Mode:	Active Serial	~
File <u>n</u> ame:	output_file.jic				
Advanced	Remote SOF F	ile Propert	ies		×
Input files to convert	ОК	Cancel	Help		
File/Dat Flash Loader EP4CE22 SOF Data DE0_NANO.so	F	age_0 P4CE22F17		auto>	Add <u>H</u> ex Data Add Sof Page
					Remove
					Down
					Properties
			Gener	ate Close	Help

Figure 9-5 Compression the sof file

■ Write JIC File into Serial Configuration Device

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

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



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

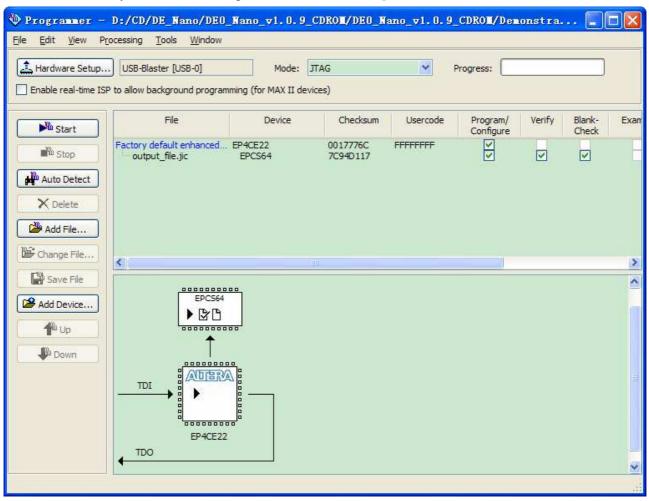


Figure 9-6 Quartus II programmer window with one JIC file

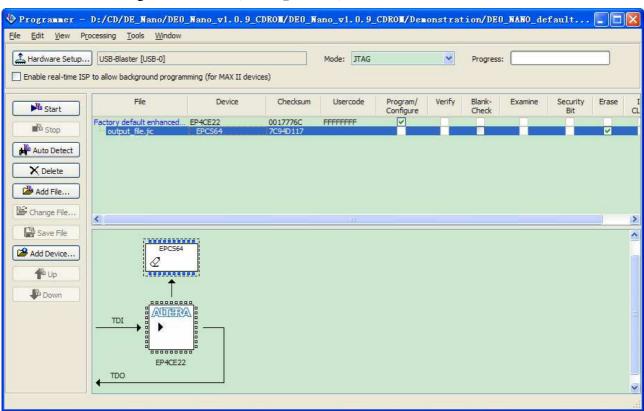
3. Click **Start** to program serial configuration device.

Erase the Serial Configuration Device

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

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





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

Figure 9-7 Erasing setting in Quartus II programmer window

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



9.2 EPCS Programming via nios-2-flash-programmer

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

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

	-							scon: 0		instance	0,000
No	EPCS	layo	ut	data	- 100	king	for	section	(E)	PCS-01021	61
		to us		in the second	10000						
		targ					ed				

Figure 9-8 EPCS Message

9.3 Revision History

Version	Change Log
V1.0	Initial Version (Preliminary)
V1.3	Add Table 3-1,3-2 and 3-3
V1.4	Modified Digital Accelerometer Description on page 31
V1.5	Modified ADC description on page 32
V1.6	Corrected Digital Accelerometer Schematic on page 23
V1.7	Modified Altera EPCS16 to be Spansion EPCS64
V1.8	Add SDRAM test section

9.4 Copyright Statement

Copyright © 2012 Terasic Technologies. All rights reserved.

Always visit the DE0-Nano webpage for new applications.

We will continue 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.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers ("Buyers") who are developing systems that incorporate TI semiconductor products (also referred to herein as "components"). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer's systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER'S USE OF TI REFERENCE DESIGNS.

TI reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques for TI components are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in Buyer's safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed an agreement specifically governing such use.

Only those TI components that TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have **not** been so designated is solely at Buyer's risk, and Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2015, Texas Instruments Incorporated