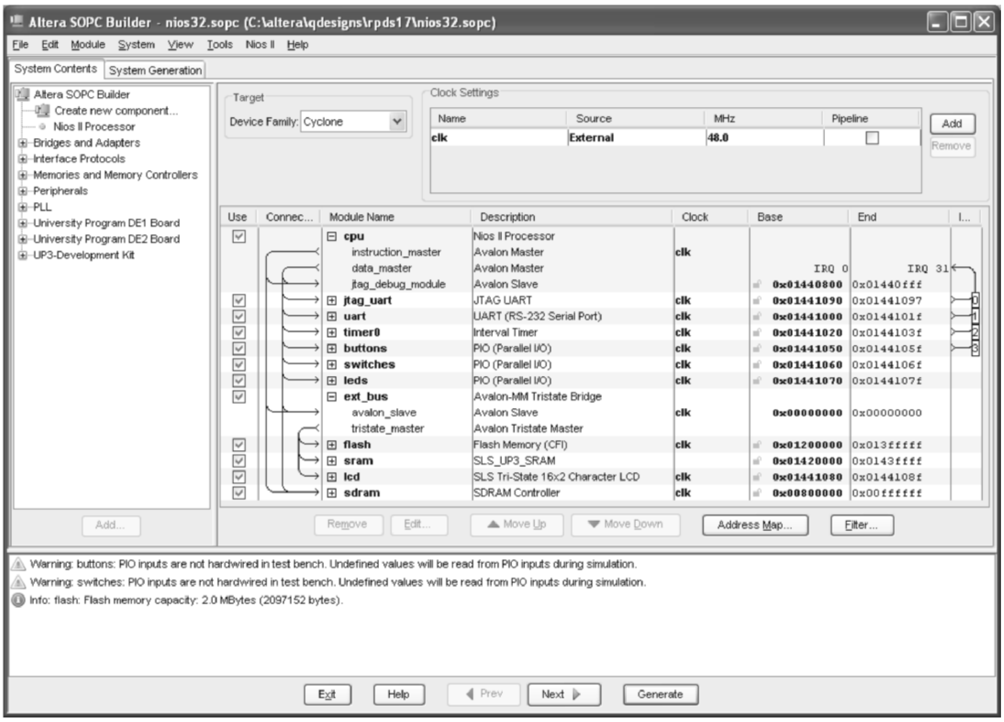
CHAPTER 17

Tutorial IV: Nios II Processor Hardware Design



Target
Device Family: Cyclone

Clock Settings

Name	Source	MHz	Pipeline
clk	External	48.0	☐

Use	Connec...	Module Name	Description	Clock	Base	End	I...
✓		cpu	Nios II Processor				
		instruction_master	Avalon Master	clk			
		data_master	Avalon Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Slave		0x01440800	0x01440fff	
✓		jtag_uart	JTAG UART	clk	0x01441090	0x01441097	0
✓		uart	UART (RS-232 Serial Port)	clk	0x01441000	0x0144101f	1
✓		timer0	Interval Timer	clk	0x01441020	0x0144103f	2
✓		buttons	PIO (Parallel I/O)	clk	0x01441050	0x0144105f	3
✓		switches	PIO (Parallel I/O)	clk	0x01441060	0x0144106f	
✓		leds	PIO (Parallel I/O)	clk	0x01441070	0x0144107f	
✓		ext_bus	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Slave	clk	0x00000000	0x00000000	
		tristate_master	Avalon Tristate Master				
✓		flash	Flash Memory (CFI)	clk	0x01200000	0x013fffff	
✓		sram	SLS_UP3_SRAM		0x01420000	0x0143ffff	
✓		lcd	SLS Tri-State 16x2 Character LCD	clk	0x01441080	0x0144108f	
✓		sdram	SDRAM Controller	clk	0x00800000	0x00ffffff	

```
Warning: buttons: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Warning: switches: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Info: flash: Flash memory capacity: 2.0 MBytes (2097152 bytes).
```

SOPC Builder is a GUI-based hardware design tool used to configure the Nios II processor core options and to design bus and I/O interfaces for the processor.

17 Tutorial IV: Nios II Processor Hardware Design

Designing systems with embeddeded processors requires both hardware and software design elements. A collection of CAD tools developed by Altera enable you to design both the hardware and software for a fully functional, customizable, soft-core processor called Nios II. This tutorial steps you through the hardware implementation of a Nios II processor for the DE1 and DE2 boards, and Tutorial III (in the preceding chapter) introduces the software design tools for the Nios II processor.

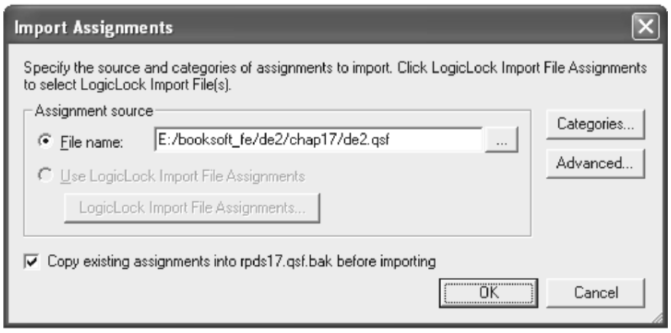
Upon completion of this tutorial, you will be able to:

- Navigate Altera's SOPC Builder (Nios II processor design wizard),
- Generate a custom Nios II processor core,
- Create a PLL that supplies a clock signal for the on-board SDRAM, and
- Specify the top-level pin assignments and project settings necessary for implementing the Nios processor on the DE boards.

THE DVD CONTAINS A VERSION OF CHAPTERS 16 AND 17 FOR THE UP 3 BOARDS.

17.1 Install the DE board files

Run the installation program for Altera's University Program IP Library. This program can be found on the DVD at **\Altera_Software\UP_IP_Library.exe**.



Specify the source and categories of assignments to import. Click LogicLock Import File Assignments to select LogicLock Import File(s).

Assignment source

(● File name: E:/booksoft_fe/de2/chap17/de2.qsf

○ Use LogicLock Import File Assignments

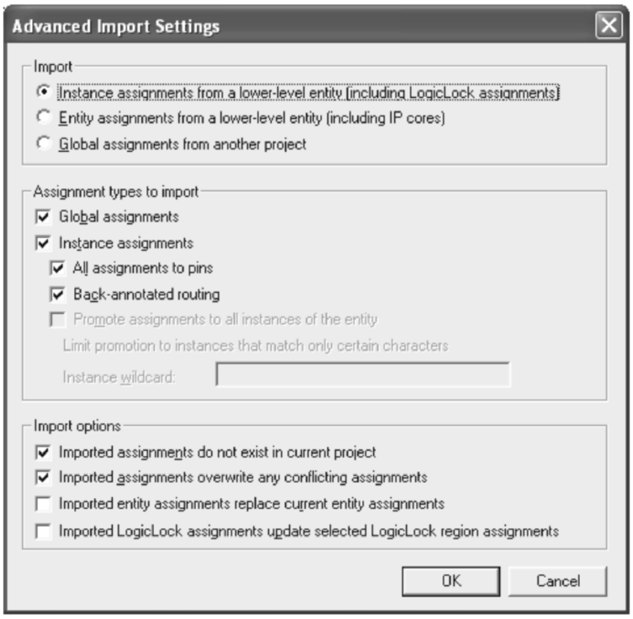
☑ Copy existing assignments into rpds17.qsf.bak before importing

Figure 17.1 Import the default pin and project assignments for the DE board.

17.2 Creating a New Project

Create a new Quartus II project as illustrated in Tutorial I (see Section 1 of Chapter 1). Use the project name **rpds17** and create a top-level Block Diagram/Schematic file named **rpds17.bdf**.

Import the pin assignments and project settings file from the DVD by choosing **Assignments ➩Import Assignments…**. Enter the full path for the **booksoft_fe\de2\chap17\de2.qsf** file located on the DVD that came with this book as shown in Figure 17.1. (If you are using a DE1 board, then you must use the **booksoft_fe\de1\chap17\de1.qsf** file located on the DVD.) Click on the **Advanced** button and verify that the settings match the dialog box in Figure 17.2. If different settings are used, then all of the pin and project assignments may not be made correctly, and downloading your project to the DE board could damage it. When the settings are correct, click **OK** to exit the dialog box. Click **OK** in the **Import Assignments** dialog box to import the settings.


Advanced Import Settings

Import
- [✓] Instance assignments from a lower-level entity (including LogicLock assignments)
- [ ] Entity assignments from a lower-level entity (including IP cores)
- [ ] Global assignments from another project

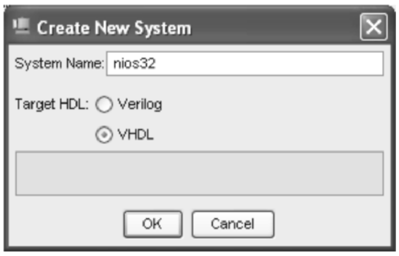
Assignment types to import
- [✓] Global assignments
- [✓] Instance assignments
  - [✓] All assignments to pins
  - [✓] Back-annotated routing
  - [ ] Promote assignments to all instances of the entity
  - Limit promotion to instances that match only certain characters
  - Instance wildcard:

Import options
- [✓] Imported assignments do not exist in current project
- [✓] Imported assignments overwrite any conflicting assignments
- [ ] Imported entity assignments replace current entity assignments
- [ ] Imported LogicLock assignments update selected LogicLock region assignments

Figure 17.2 It is important that the Advanced Import Options be set as shown here.

17.3 Starting SOPC Builder

A Nios II processor is created using the SOPC Builder wizard. Within this wizard, you can specify the settings for the Nios II processor, add peripherals, and select the bus connections, I/O memory mapping, and IRQ assignments for the processor. To start the SOPC Builder, choose **Tools ➩SOPC Builder…**.



System Name: nios32

Target HDL: ○ Verilog ◉ VHDL

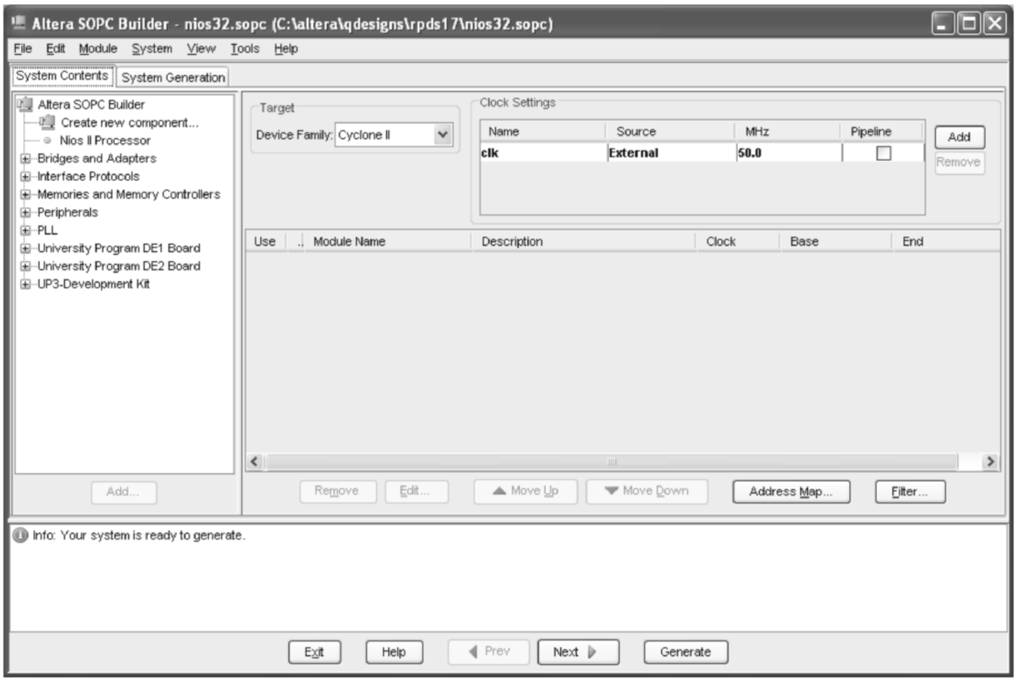
Figure 17.3 Specifying the name of the Nios II processor for your system.

In the **Create New System** dialog box, enter the name **nios32**, and set the **Target HDL** to **VHDL** as shown in Figure 17.3. Click **OK** to open SOPC Builder with a blank project titled nios32.

The system settings in the top part of SOPC Builder window must be set for the board and device that you are using. For the DE boards, the on-board clock circuit generates several clock frequencies, including 24 MHz, 27 MHz, and 50 MHz. For this tutorial, the 50 MHz clock signal will be used; therefore, enter **50.0** in the **clk** field. Select **Cyclone II** as **Device Family**. When these settings have been entered, your SOPC Builder window should look similar to the screen shot in Figure 17.4.

IT IS CRITICAL THAT THE FREQUENCY SELECTED IN THE SOPC BUILDER IS THE ACTUAL CLOCK RATE USED IN YOUR HARDWARE DESIGN. IF A PLL IS USED TO GENERATE A DIFFERENT NIOS II CLOCK SIGNAL, THEN THAT CLOCK FREQUENCY MUST BE ENTERED INTO THE SOPC BUILDER BEFORE THE SYSTEM IS GENERATED. IF YOU MODIFY THE CLOCK FREQUENCY FOR THE NIOS II PROCESSOR LATER, THEN YOU MUST RE-GENERATE THE NIOS II PROCESSOR WITH THE UPDATED FREQUENCY SPECIFIED HERE.

Take a minute to familiarize yourself with the layout of the SOPC Builder window. Along the left-hand side, there is an expandable list of components organized by category that can be added to a Nios II system. Click on the "+" symbol next to the items in this list to expand the list of components for each category. If board support packages have been installed, then those development boards will be listed as an item. Expanding these items will reveal components that are specific to these boards. If you installed the design files as discussed in Section 17.1, then the **University Program DE1 Board** and **University Program DE2 Board** categories will appear.



Altera SOPC Builder - nios32.sopc (C:\altera\qdesigns\rpds17\nios32.sopc)

Device Family: Cyclone II

Name	Source	MHz	Pipeline
clk	External	50.0	☐

Info: Your system is ready to generate.

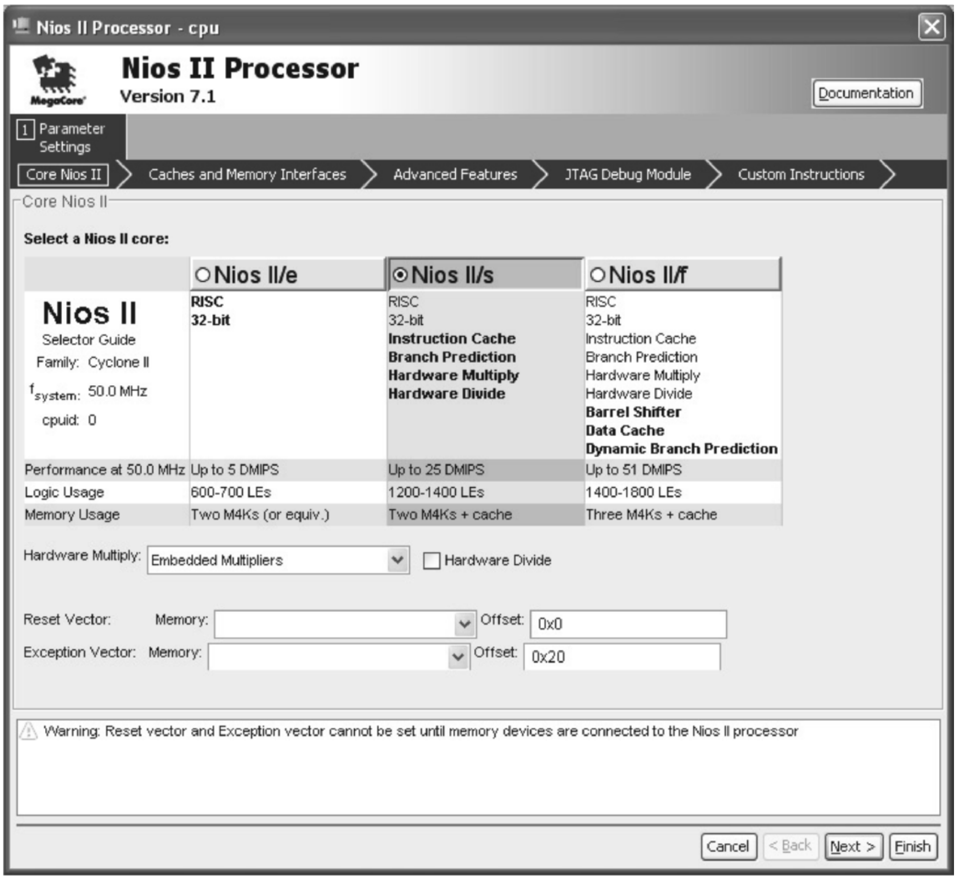
Figure 17.4 Beginning a Nios II design in the SOPC Builder.

17.4 Adding a Nios II Processor

The first component that you will add to your Nios II processor design is the processor core itself. In the list of components on the left-hand side of the SOPC Builder, the **Nios II Processor** component. Click the **Add…** button at the bottom of the component list.

When a component is added to your system, a dialog box will appear that allows you to select options and set specific parameters for this particular implementation. For the Nios II processor, the dialog box shown in Figure 17.5 will appear. This first selection will determine the general parameters of the Nios II processor. Notice that there are three general configurations allowed that vary in size, performance, and functionality. Select the middle configuration, **Nios II/s** as shown in Figure 17.5. In the **Hardware Multiply** field, select **Embedded Multipliers**, and click **Next** to continue.

The next dialog box allows you set the size of the instruction cache in the Nios II processor. Keep the default value (4 KB), and click **Next** twice to advance to the **JTAG Debug Module** dialog box.


Nios II Processor - cpu

Nios II Processor
Version 7.1

Core Nios II

Select a Nios II core:

Nios II Selector Guide Family: Cyclone II f_{system}: 50.0 MHz cpuid: 0	○Nios II/e	⦿Nios II/s	○Nios II/f
	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 5 DMIPS	Up to 25 DMIPS	Up to 51 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M4Ks (or equiv.)	Two M4Ks + cache	Three M4Ks + cache

Hardware Multiply: Embedded Multipliers ☐ Hardware Divide

Reset Vector: Memory: Offset: 0x0
Exception Vector: Memory: Offset: 0x20

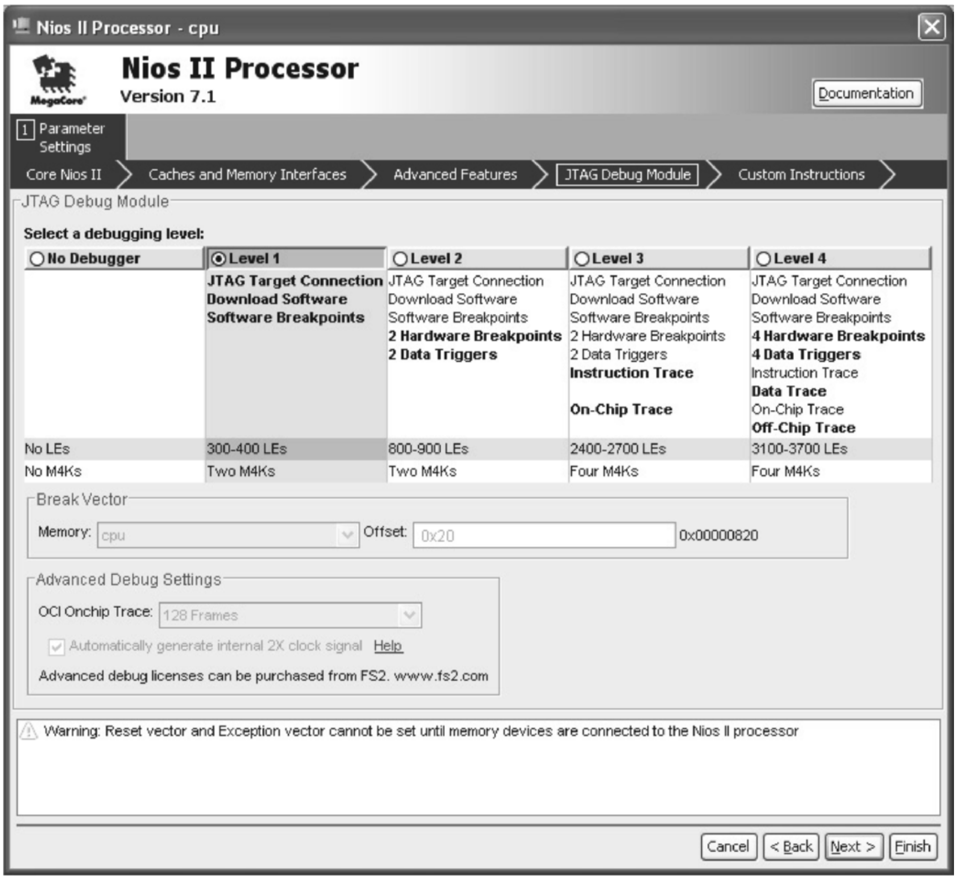
Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

Figure 17.5 Nios II supports three different general configurations. Select Nios II/s for this tutorial.

Nios II processors can be compiled with support for one of four different debugging systems. The differences between them are shown in Figure 17.6, along with the FPGA resources required to implement each type of debugging. There is an order of magnitude difference in the number of logic elements required to implement Level 4 debugging versus Level 1 debugging. This difference is significant when compared to the overall size of the Nios II processor. The Level 4 debugging system is two to three times larger then the Nios II/s processor itself. Since the cost of FPGAs are largely based on their size, the debugging logic will typically be removed before a design enters production to minimize the number of logic elements, and thus the size of the FPGA, required for the production quantities.

The full features of Level 3 and Level 4 debugging are only available when a license from First Silicon Solutions, a third-party company, is purchased. The availability of this license within your company or school along with the complexity of your end system and the size of the FPGA available will be the primary factors in determining which debugging system should be selected for a given system. For this tutorial, select **Level 1** (the default), and click **Next** to continue.

The final option in the Nios II processor configuration is the adding of custom instructions. Nios II processors allow the addition of up to 256 custom instructions into the processor's data path. These can be used to further customize your processor for a specific task. For this tutorial, no custom instructions will be added. Click **Finish** to complete the Nios II configuration for this system.



JTAG Debug Module

Select a debugging level:

No Debugger	[✓ Level 1]	Level 2	Level 3	Level 4
	JTAG Target Connection	JTAG Target Connection	JTAG Target Connection	JTAG Target Connection
	Download Software	Download Software	Download Software	Download Software
	Software Breakpoints	Software Breakpoints	Software Breakpoints	Software Breakpoints
		2 Hardware Breakpoints	2 Hardware Breakpoints	**4 Hardware Breakpoints**
		2 Data Triggers	2 Data Triggers	**4 Data Triggers**
			Instruction Trace	Instruction Trace
				Data Trace
			On-Chip Trace	On-Chip Trace
				Off-Chip Trace
No LEs	300-400 LEs	800-900 LEs	2400-2700 LEs	3100-3700 LEs
No M4Ks	Two M4Ks	Two M4Ks	Four M4Ks	Four M4Ks

Break Vector

Memory: cpu Offset: 0x20 0x00000820

Advanced Debug Settings

OCI Onchip Trace: 128 Frames

[✓] Automatically generate internal 2X clock signal Help

Advanced debug licenses can be purchased from FS2. www.fs2.com

Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

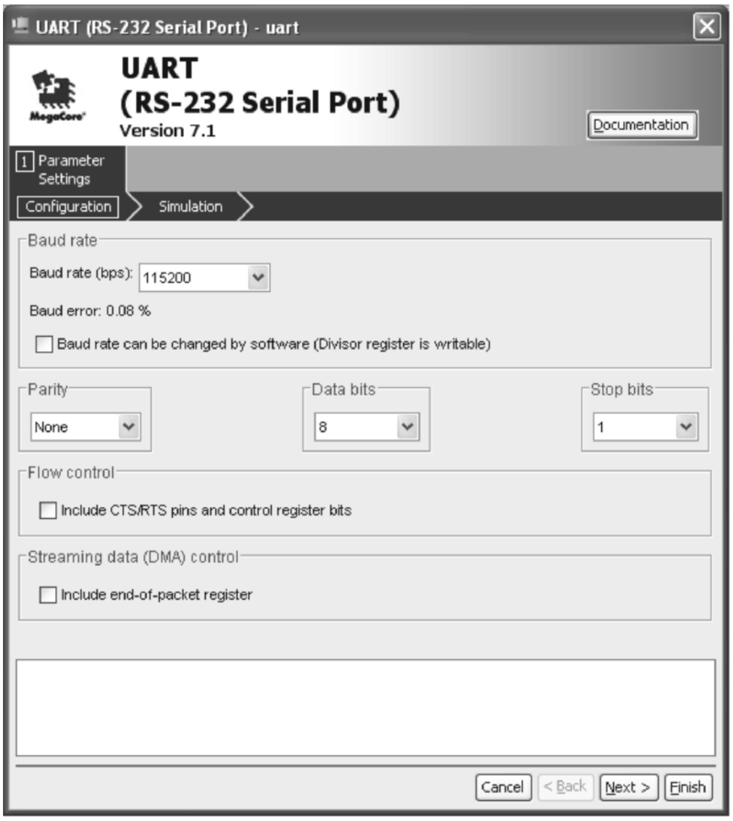
Figure 17.6 Nios II supports four levels of debugging capabilities. Select Level 1 for this tutorial.

When the SOPC Builder window reappears, the Nios II processor will appear as an added component in the main part of the window with the default module name **cpu**. Also, a number of error and warning messages will appear in the console at the bottom of the SOPC Builder window. These messages result from there not being any defined memory in the system yet. When memory is added in the next few sections, the messages will disappear.

17.5 Adding UART Peripherals

Two UART peripherals will be defined for this system: a JTAG UART and an RS-232 serial UART. The USB Blaster JTAG cable that is used to configure the FPGA can also be used as a UART device after the FPGA is configured. (The JTAG cable is also used as the communication channel between the PC and the debugging logic selected for the Nios II processor.) The Nios II software integrated development environment (IDE) uses the JTAG UART as the default device for downloading your software instructions to the Nios II processor and was used for that purpose in the previous tutorial on software design.

Add the JTAG UART device by expanding **Interface Protocols ⇨Serial**. Select **JTAG UART** and click **Add…**. When the JTAG UART Configuration dialog box appears, click **Finish** to accept the default values for all fields and add the component.


UART (RS-232 Serial Port) - uart

UART (RS-232 Serial Port)
Version 7.1

Baud rate
Baud rate (bps): 115200
Baud error: 0.08 %
☐ Baud rate can be changed by software (Divisor register is writable)

Parity: None
Data bits: 8
Stop bits: 1

Flow control
☐ Include CTS/RTS pins and control register bits

Streaming data (DMA) control
☐ Include end-of-packet register

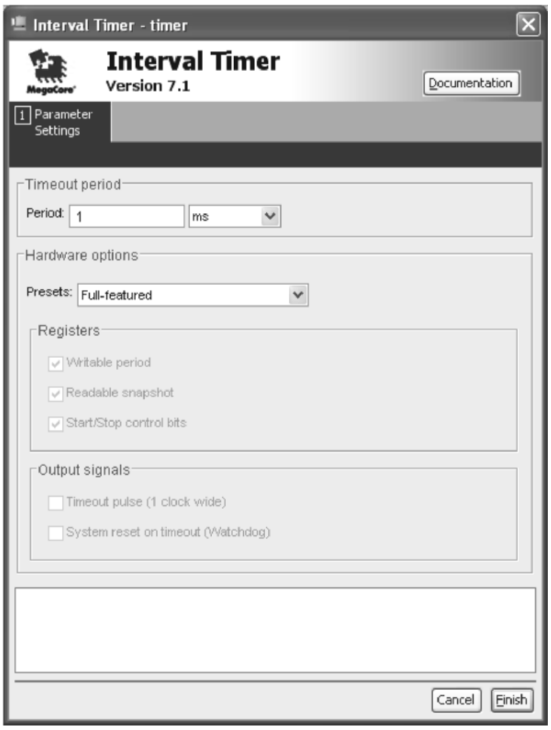
Figure 17.7 These are the settings for the RS-232 UART device to be added to the Nios II system.

Older ByteBlaster II, ByteBlaster MV, and ByteBlaster JTAG cables did not transmit the run-tiem serial data robustly; therefore, a second RS-232 UART module was needed for run-time serial communication. The newer USB Blaster JTAG interface used on the DE boards works quite well as a run-time UART device. Thus, adding a second UART module is not required; however, adding an RS-232 UART module can be useful when debugging systems by providing an additional communication channel.

Add the RS-232 UART peripheral by expanding **Interface Protocols ⇨Serial**. Select **UART (RS-232 serial port)** and click **Add…**. When the UART configuration dialog box appears, set the options as shown in Figure 17.7. Click **Finish** to add the component.

17.6 Adding an Interval Timer Peripheral

Most processor designs require at least one timer. This timer is used to delay the processor, coordinate transactions, timestamp events, generate time slice interrupts for an operating system scheduler, a watchdog timer, and more. The Nios II timer peripheral is flexible; it has several options and three predefined configurations. Add a full-featured interval timer to your Nios II processor by expanding **Peripherals ⇨Microcontroller Peripherals**. Select **Interval Timer** and click **Add…**. When the timer configuration dialog box appears, set the options as shown in Figure 17.8. Click **Finish** to add the component. In the SOPC Builder, rename the timer module to **timer0**. The “0” is appended to the timer name here to provide a consistent naming convention for your timers if additional timers are added at a later time. It is not unusual for a processor to have two or three timers – often of different configurations for specific uses.



Timeout period

Period: 1 ms

Hardware options

Presets: Full-featured

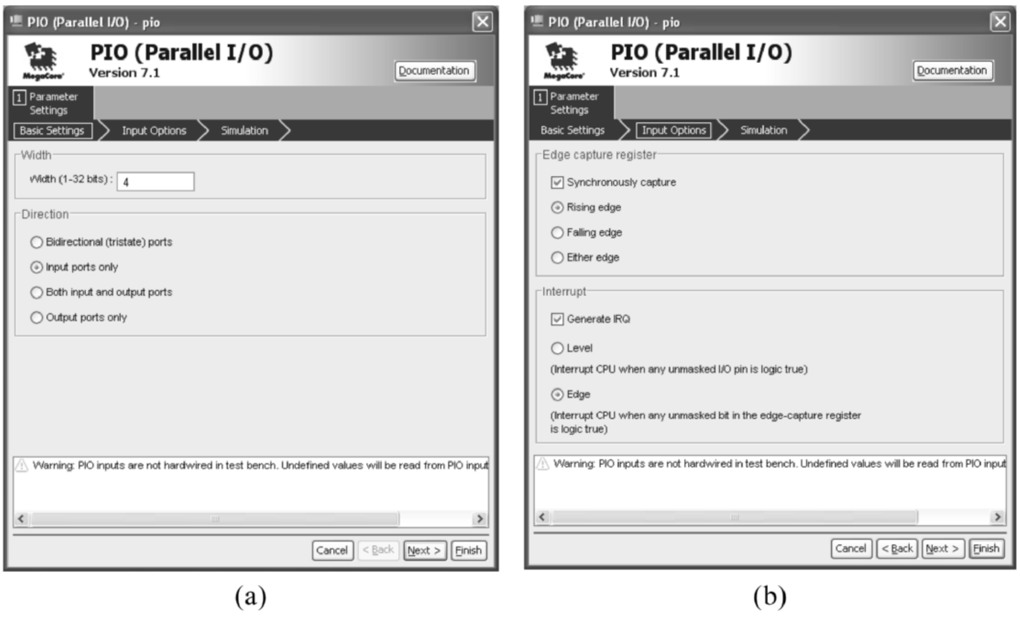
Registers

- [✓] Writable period
- [✓] Readable snapshot
- [✓] Start/Stop control bits

Output signals

- [ ] Timeout pulse (1 clock wide)
- [ ] System reset on timeout (Watchdog)

Figure 17.8 These are the settings for the interval timer device to be added to the Nios II system.



(a)

Width

Width (1-32 bits) : 4

Direction

- Bidirectional (tristate) ports
- [✓ Input ports only]
- Both input and output ports
- Output ports only

Warning: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO input

(b)

Edge capture register

- [✓] Synchronously capture
- [✓ Rising edge]
- Falling edge
- Either edge

Interrupt

- [✓] Generate IRQ
- Level
  (Interrupt CPU when any unmasked I/O pin is logic true)
- [✓ Edge]
  (Interrupt CPU when any unmasked bit in the edge-capture register is logic true)

Warning: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO input

Figure 17.9 These are the settings for the pushbutton PIO device to be added to the Nios II system.

17.7 Adding Parallel I/O Components

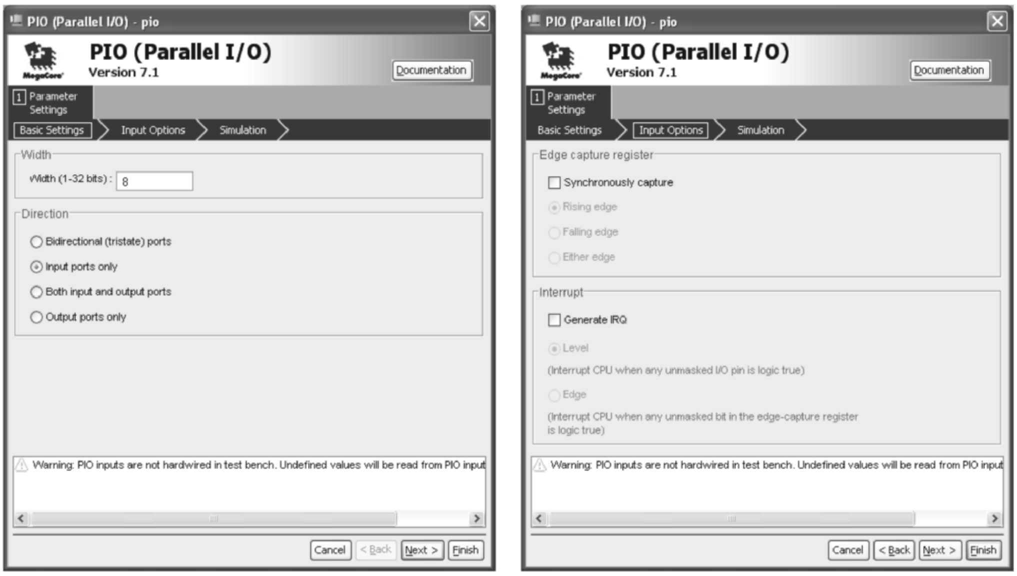
Many processors require a certain amount of general-purpose I/O pins. These pins can be attached directly to pushbuttons, switches, LEDs, and similar I/O devices. They can also be attached to relatively simple or low bandwidth interfaces that don't have a large amount of overhead associated with data transmission. Examples of these types of interfaces include PS/2, I^{2}C, SPI, and parallel data interfaces.

In addition, general-purpose I/O pins can be used to pass low-bandwidth data between a custom VHDL or Verilog block and the Nios II processor. A faster method of transferring data to a VHDL block is to create a custom peripheral that can attach to the Avalon bus. Implementing a VHDL module that is compliant with the Avalon bus specification is more involved and requires more logic elements than using general-purpose I/O pins, but it does provide a faster more efficient interface.

General-purpose I/O pins are added to the Nios II processor with the PIO (Parallel I/O) component. The PIO component has a number of options for customizing general-purpose I/O interfaces. PIO interfaces can be specified as input only, output only, or bidirectional. If bidirectional is selected here, then the direction of each pin must be set in the direction register at run-time via software. Input PIO interfaces can also have various interrupt and edge capture capabilities including the capturing of either or both edges and edge or level-sensitive interrupt triggers.

For this tutorial, you will add three PIO components: one for the pushbuttons, one for the switches, and one for the LEDs. First, add a PIO component for the pushbuttons to your processor design by expanding **Peripherals ⇨ Microcontroller Peripherals**. Select **PIO (Parallel I/O)** and click **Add…**. When the PIO configuration dialog box appears, set the **Width** of the interface to **4** bits (there are four pushbuttons) and set the **Direction** to **Input ports only** as shown in Figure 17.9(a). Click **Next** to continue. On the next configuration page, set the options as shown in Figure 17.9(b). Click **Finish** to add the component. In the SOPC Builder, rename the PIO module to **buttons**.

Using the same procedure as above, add a second PIO component for the dipswitches. The settings for the PIO devices are shown in Figure 17.10. Rename this PIO module to **switches**.



(a)

Width

Width (1-32 bits) : 8

Direction

- Bidirectional (tristate) ports
- [✓ Input ports only]
- Both input and output ports
- Output ports only

Warning: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO input

(b)

Edge capture register

- ☐ Synchronously capture
- [✓ Rising edge]
- Falling edge
- Either edge

Interrupt

- ☐ Generate IRQ
- [✓ Level]
  (Interrupt CPU when any unmasked I/O pin is logic true)
- Edge
  (Interrupt CPU when any unmasked bit in the edge-capture register is logic true)

Warning: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO input

Figure 17.10 These are the settings for the switch PIO device to be added to the Nios II system.

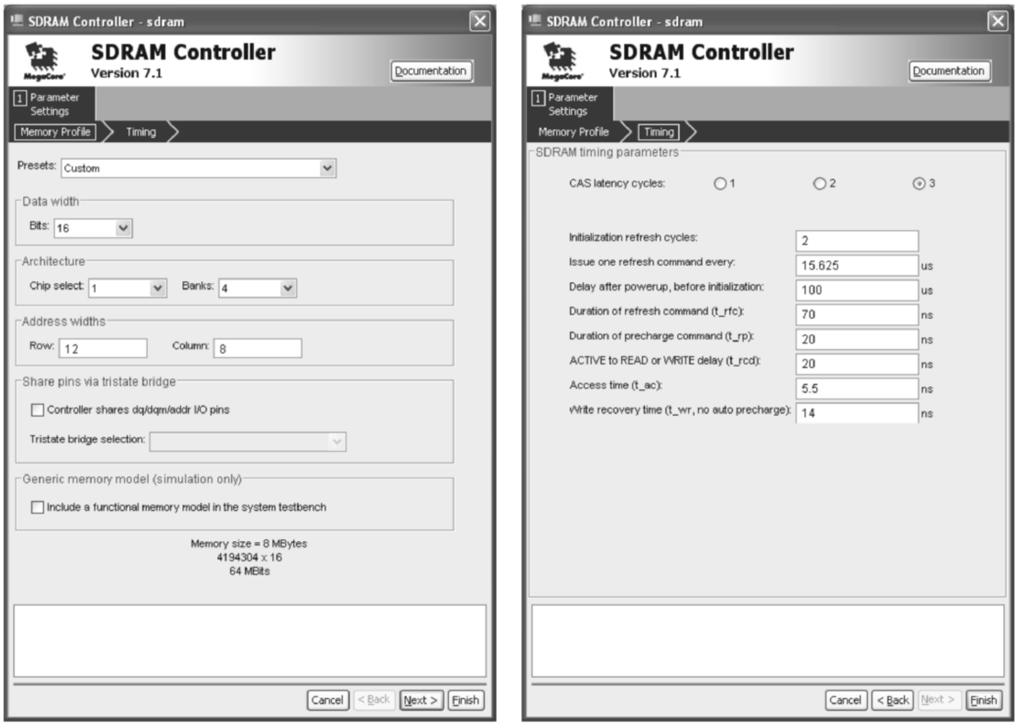
Finally, add a third PIO component for the LEDs. On the first configuration page, set **8** bits for the **Width**, and set the **Direction** to **Output ports only**. When the PIO is an output-only device, the interrupt and edge-capture options are not applicable. Rename this PIO module to **leds**.

17.8 Adding an SRAM Memory Controller

Add the SRAM memory controller to your Nios II processor by expanding **University Program DE2 Board**. (The SRAM components are identical for the DE1 and DE2 boards, so the SRAM component may be added from either library.) Select **SRAM** and click **Add…**. When the SRAM configuration dialog box appears, click **Finish** to add the component. This component does not have any configuration options; therefore, the SRAM component dialog box contains information only. In the SOPC Builder, rename the SRAM module to **sram**.

17.9 Adding an SDRAM Memory Controller

There are three types of memory on the DE board: SDRAM, SRAM, and Flash. Each type of memory requires is own unique memory controller and must be added individually. Add the SDRAM memory controller by expanding **Memories and Memory Controllers ⇨ SDRAM**. Select **SDRAM Controller** and click **Add…**. The SDRAM controller must be configured for the timing requirements of the specific SDRAM brand and model being used. The configuration and timing values requested here are typically available in the datasheet for SDRAM ICs. For the SDRAM modules on the DE board, set the options in the configuration dialog boxes to the values shown in Figure 17.11. Click **Finish** to add the component. In the SOPC Builder, rename the SDRAM controller module to **sdram**.



(a)

Presets: Custom

Data width
Bits: 16

Architecture
Chip select: 1 Banks: 4

Address widths
Row: 12 Column: 8

Share pins via tristate bridge
☐ Controller shares dq/dqm/addr I/O pins
Tristate bridge selection:

Generic memory model (simulation only)
☐ Include a functional memory model in the system testbench

Memory size = 8 MBytes
4194304 x 16
64 MBits

(b)

SDRAM timing parameters

CAS latency cycles: ○ 1 ○ 2 [✓ ⦿ 3]

Initialization refresh cycles:	2	
Issue one refresh command every:	15.625	us
Delay after powerup, before initialization:	100	us
Duration of refresh command (t_rfc):	70	ns
Duration of precharge command (t_rp):	20	ns
ACTIVE to READ or WRITE delay (t_rcd):	20	ns
Access time (t_ac):	5.5	ns
Write recovery time (t_wr, no auto precharge):	14	ns

Figure 17.11 These are the SDRAM controller settings for use with the SDRAM on the DE boards.

17.10 Adding the LCD Module *(DE2 Board Only)*

The DE2 board contains a Liquid Crystal Display (LCD) component. To interface this display to the Nios II processor, add the LCD component to your

Nios II processor by expanding **Peripherals ➪Display**. Select **Character LCD** and click **Add…**. When the LCD component dialog box appears, click **Finish** to add the component. This component does not have any configuration options; therefore, the LCD component dialog box contains information only. The LCD component will be added to the list of peripherals in your Nios II processor.

17.11 Adding an External Bus

Multiple external devices can share the same address and data bus pins and dramatically reduce the number of pins required on the FPGA. The Nios II processor supports this type of bus sharing with its tristate bus components. On many boards the SRAM, SDRAM, Flash, and even an LCD device can share a signal external tristate bus. To accommodate the bidirectional data bus and multiple devices on a single bus, an Avalon Tristate Bridge component must be added. The Avalon tristate bridge creates a peripheral (tristate) bus to which multiple memory controllers and other external components can be attached. It also provides a seamless interface between the peripheral bus and the main system bus. A conceptual drawing of this arrangement is shown in Figure 17.12. For the DE boards, the Flash device is the only peripheral attached to the tristate bus. The SDRAM, SRAM, and LCD devices all attach directly to the main system bus.

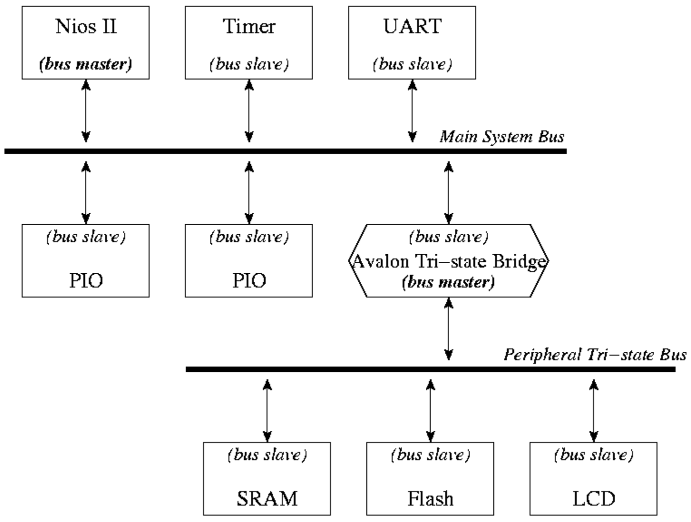
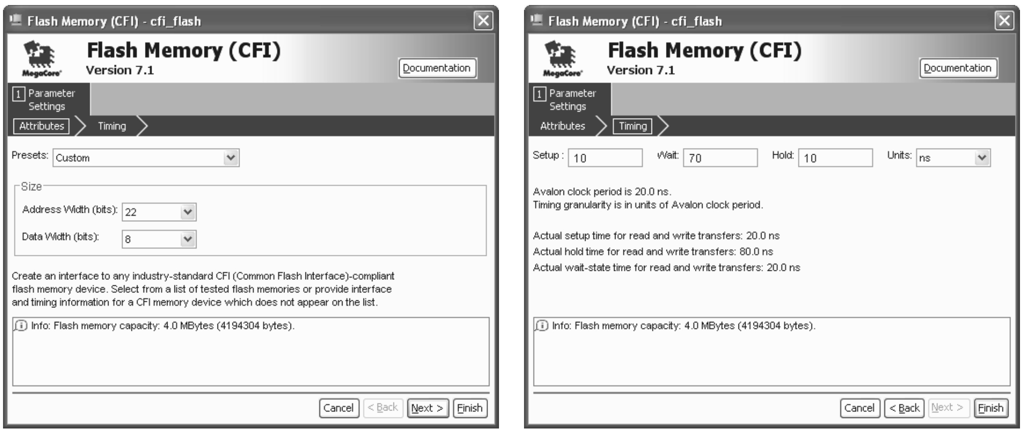


Figure 17.12 This is a conceptual drawing of the bus configuration with the Tristate Bridge connecting the main system bus and the shared peripheral bus.

Add the Avalon Tristate Bridge component by expanding **Bridges and Adapters ➪Memory Mapped**. Select **Avalon-MM Tristate Bridge** and click **Add…**. There is only one option for this component: registered or not registered. Select **Registered** and click **Finish** to add the component. In the SOPC Builder, rename the bridge module to **ext_bus**.

17.12 Adding Components to the External Bus

Once the Avalon tri-state bridge has been added, the peripherals that are going to connect to the external peripheral bus can be added. First, add the Flash memory controller by expanding **Memories and Memory Controllers ⇨Flash**. Select **Flash Memory (CFI)** and click **Add…**. When the Flash memory configuration dialog box appears, set the options as shown in Figure 17.13. In the SOPC Builder, rename the flash module to **flash**.



Presets: Custom

Size

Address Width (bits): 22

Data Width (bits): 8

Create an interface to any industry-standard CFI (Common Flash Interface)-compliant flash memory device. Select from a list of tested flash memories or provide interface and timing information for a CFI memory device which does not appear on the list.

Info: Flash memory capacity: 4.0 MBytes (4194304 bytes).

Setup : 10 Wait: 70 Hold: 10 Units: ns

Avalon clock period is 20.0 ns.
Timing granularity is in units of Avalon clock period.

Actual setup time for read and write transfers: 20.0 ns
Actual hold time for read and write transfers: 80.0 ns
Actual wait-state time for read and write transfers: 20.0 ns

Info: Flash memory capacity: 4.0 MBytes (4194304 bytes).

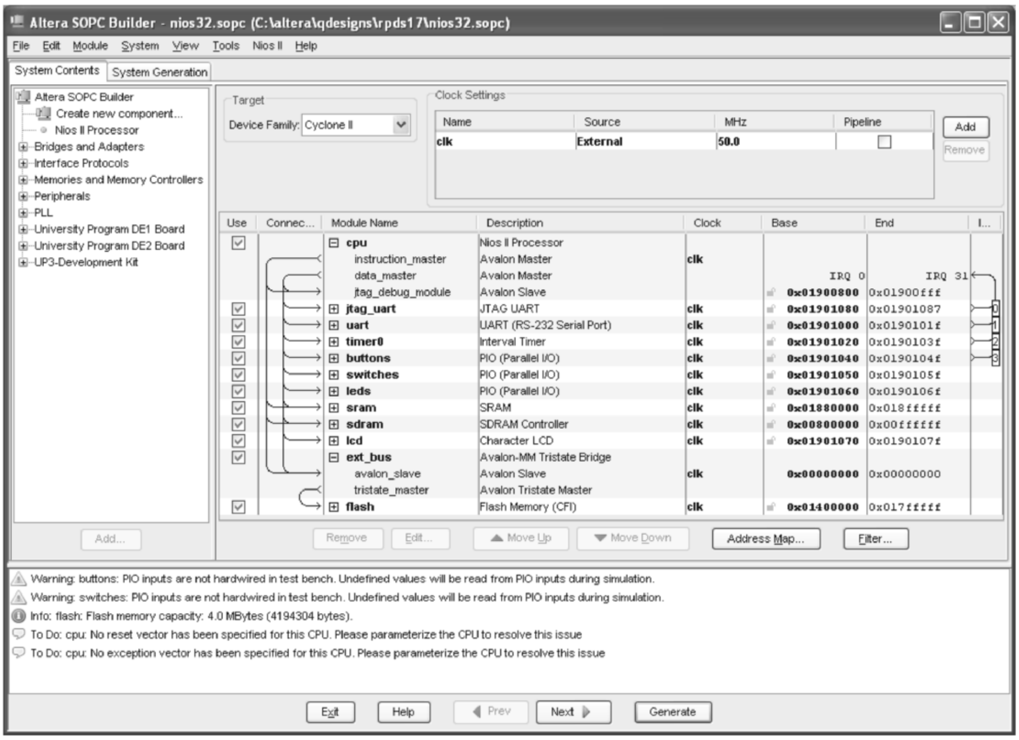
(a) (b)

Figure 17.13 These are the Flash memory settings for use with the Flash on the DE boards.

17.13 Global Processor Settings

All of the necessary peripherals have been added now. The next step is to configure some global settings for your processor.

To view and modify the bus connections in your processor, select **View ⇨Show Connections Column**. (If **Show Connections Column** is already selected, then un-select it and select it again.) This will expand the **cpu** and **ext_bus** modules in the table of peripherals and show the bus connections. The three buses are displayed vertically. From left-to-right, the buses are the main system instruction, main system data, and tri-state data bus. Notice that the UARTs, timer, LCD, and PIO components are only attached to the system data bus since they don't normally interact with instruction memory. SRAM, SDRAM, and the Avalon Tristate Bridge are connected to both the system instruction and system data buses, because the memory devices can store both data and instruction memory. Finally, the Flash memory device is unconnected. It must be manually connected to the appropriate tristate bus. Hover your mouse over the connections column just to the left of the **flash** module. An open circle will appear on the tri-state data bus. Click on the open circle to connect the **flash** module to the external tristate bus (a connection is denoted by a solid, filled-in circle). The final SOPC Builder window should look like the screen shot in Figure 17.14.



Device Family: Cyclone II

Name	Source	MHz	Pipeline
clk	External	50.0	☐

Use	Connec...	Module Name	Description	Clock	Base	End	I...
✓		cpu	Nios II Processor				
		instruction_master	Avalon Master	clk			
		data_master	Avalon Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Slave		0x01900800	0x01900fff	
✓		jtag_uart	JTAG UART	clk	0x01901080	0x01901087	0
✓		uart	UART (RS-232 Serial Port)	clk	0x01901000	0x0190101f	1
✓		timer0	Interval Timer	clk	0x01901020	0x0190103f	2
✓		buttons	PIO (Parallel I/O)	clk	0x01901040	0x0190104f	3
✓		switches	PIO (Parallel I/O)	clk	0x01901050	0x0190105f	
✓		leds	PIO (Parallel I/O)	clk	0x01901060	0x0190106f	
✓		sram	SRAM	clk	0x01880000	0x018fffff	
✓		sdram	SDRAM Controller	clk	0x00800000	0x00ffffff	
✓		lcd	Character LCD	clk	0x01901070	0x0190107f	
✓		ext_bus	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Slave	clk	0x00000000	0x00000000	
		tristate_master	Avalon Tristate Master				
✓		flash	Flash Memory (CFI)	clk	0x01400000	0x017fffff	

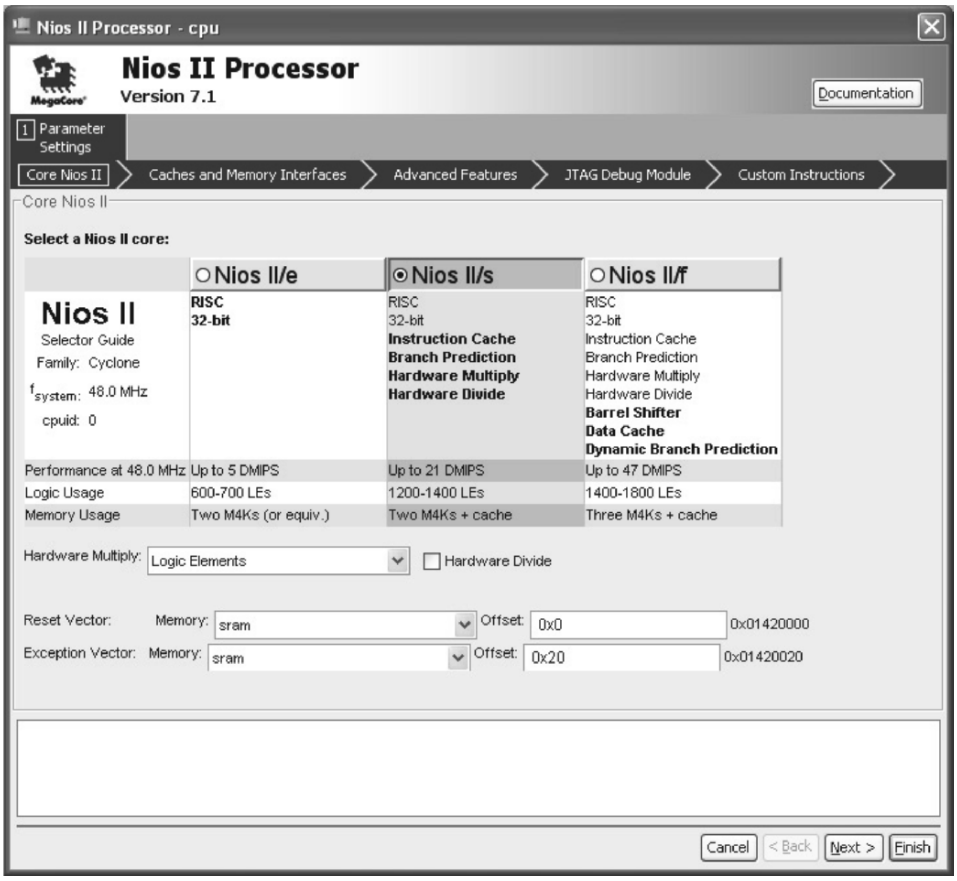
```
Warning: buttons: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Warning: switches: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Info: flash: Flash memory capacity: 4.0 MBytes (4194304 bytes).
To Do: cpu: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
To Do: cpu: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
```

Figure 17.14 This is the completed Nios II design in SOPC Builder.

The Nios II processor uses a memory-mapped I/O scheme for accessing peripherals. Each component added to the system is assigned a unique set of memory addresses. Any device or data registers needed for a particular peripheral can be accessed by reading from or writing to its respective memory address. In general, the specific memory address assignments do not matter as long as the assigned memory address spaces do not overlap. If the Nios II system is going to be a part of a legacy system, there may be some constraints placed on the memory address assignments; however, there is nothing intrinsic within the Nios II system that restricts the settings. For this tutorial, let SOPC Builder make the memory assignments automatically by selecting **System ⇨ Auto-Assign Base Addresses**. Next, select **System ⇨ Auto-Assign IRQs** to have SOPC Builder automatically assign the IRQ values to the devices that support interrupts.

17.14 Finalizing the Nios II Processor

Now that the memory modules have been added, the Nios II processor configuration can be completed. Select the **cpu** module by clicking on it and then click the **Edit…** button. The **Nios II Processor** dialog box will appear allowing you to modify the program memory device and beginning address. For this tutorial, set the **Reset Vector** and **Exception Vector** to **sram** and keep the default offsets as shown in Figure 17.15. Click **Finish** to save the new processor settings.



Core Nios II

Select a Nios II core:

Nios II	Nios II/e	[✓ Nios II/s]	Nios II/f
Selector Guide Family: Cyclone f_{system}: 48.0 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 48.0 MHz	Up to 5 DMIPS	Up to 21 DMIPS	Up to 47 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M4Ks (or equiv.)	Two M4Ks + cache	Three M4Ks + cache

Hardware Multiply: Logic Elements ☐ Hardware Divide

Reset Vector: Memory: sram Offset: 0x0 0x01420000

Exception Vector: Memory: sram Offset: 0x20 0x01420020

Figure 17.15 These are the processor configuration settings for the Nios II processor.

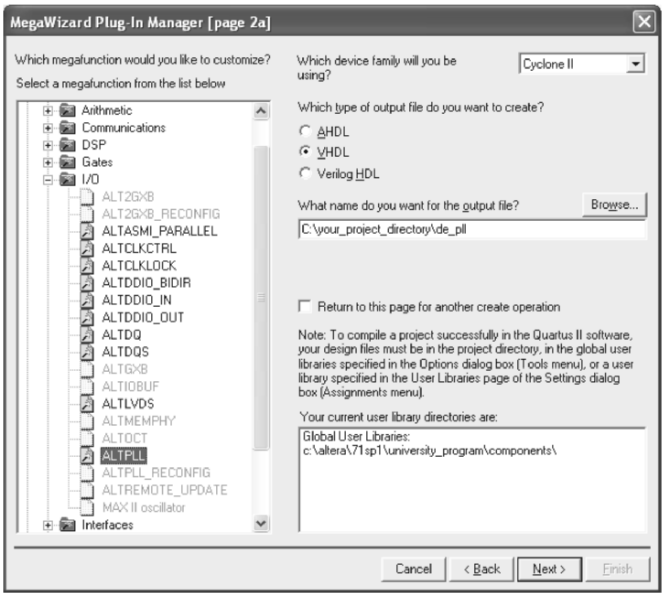
In the SOPC Builder window, click the **Next** button. The **System Generation** dialog box is the final group of settings. For this tutorial, you will not be simulating the processor in ModelSim or other third-party simulation tool; therefore, unselect the **Simulation. Create simulator project files** option. Click the **Generate** button to generate the design files for your Nios II processor. It will take 2-3 minutes to generate your Nios II processor. When it completes, the console should contain a message that states that your processor was generated successfully. If your system does not generate successfully, study the error log display in the console, correct the problem, and re-generate the Nios II processor. When you have successfully generated your Nios II system, click the **Exit** button to close SOPC Builder.

17.15 Add the Processor Symbol to the Top-Level Schematic

When SOPC Builder closes, return to your blank top-level schematic window, **rpds17.bdf**. Double click on a blank area of your empty top-level schematic file to add a component. In the **Libraries** pane of the **Symbol** dialog box, expand the **Project** item and select the **nios32** component. Click **OK** to add the selected component. Click in the middle of schematic file to place your Nios system.

17.16 Create a Phase-Locked Loop Component

SDRAM and the Nios II processor core operate on different clock edges. The Nios processor uses the rising edge and SDRAM the falling edge. The SDRAM would need a clock signal that is phase shifted by 180 degrees. An inverter would do this, but the phase shift also needs to be adjusted a bit to correct for the internal FPGA delays and the distance between the SDRAM and the FPGA on the DE board. To create this SDRAM clock signal, a phase-locked loop (PLL) component can be implemented on the FPGA. To create a PLL, use Quartus II's MegaWizard Plug-in Manager by selecting **Tools ⇨MegaWizard Plug-In Manager…**. Click **Next** on page 1 of the wizard to create a new component. On page 2, select the **Installed Plug-Ins ⇨I/O ⇨ALTPLL** module from the list. Enter the full path of your project directory followed by the filename **up3_pll** into the output filename field. Complete the remaining fields with the information shown in Figure 17.16. Click **Next** to continue.


MegaWizard Plug-In Manager [page 2a]

Which megafunction would you like to customize?
Select a megafunction from the list below

Which device family will you be using? Cyclone II

Which type of output file do you want to create?
AHDL [✓ VHDL] Verilog HDL

What name do you want for the output file?
C:\your_project_directory\de_pll

Return to this page for another create operation

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:
Global User Libraries:
c:\altera\71sp1\university_program\components\

Figure 17.16 These are the initial settings for the ALTPLL module.

On page 3 of the MegaWizard manager, enter **50.00 MHz** as the **frequency of the inclock0 input**. Leave the other options set to their default values. Click **Next** to continue. On page 4 of the MegaWizard manager, *un-select all checkmarks*. Click **Next** twice to advance to page 6.

On page 6 of the MegaWizard manager, enter a **Clock phase shift** of **-54 deg** (-3 ns). Leave the other options set to their default values. Click **Finish** to skip pages 6 and 7 and jump to page 8 of the MegaWizard manager. Click **Finish** again to complete the MegaWizard manager and create the files for the PLL component.

Double click on a blank area of the top-level schematic file. Select the **Project ⇨up3_pll** module and add it to your top-level schematic as shown in the completed schematic in Figure 17.17.

IMPORTANT NOTE: Different or future versions of the Altera software may generate slightly different hardware time delays for the SDRAM clock. If you experience SDRAM errors after running memory tests on your final design or the program downloads to SDRAM do not verify, and after double checking that everything else is correct in your design, the PLL phase shift may need to be adjusted a small amount. Most designs seem to fall within about 30 degrees of -54 degrees. This corresponds to a time delay adjustment of only 1 or 2 ns.

17.17 Complete the Top-Level Schematic

To complete the top-level schematic, add the input, output, and bi-directional pins (and pin names) shown in Figure 17.17. Also, complete the connections between the two top-level components as shown in the figure. Finally, if you added the LCD component for the DE2 board, add a VCC symbol and connect it to the **LCD_ON** and **LCD_BLON** output pins to tie them High. If you have trouble reading the signal names in the figure, the file is available on the DVD.

17.18 Design Compilation

Verify that the pin assignments discussed in Section 17.2 were made correctly by going to **Assignments ⇨Pins**. A long list of pin numbers and names corresponding to the pin names you entered into the top-level schematic should appear. If it does not, then repeat the steps in Section 17.2 to import the pin assignments.

Verify that the global assignments discussed in Section 17.2 were made correctly by going to **Assignments ⇨Device... ⇨Device & Pin Options ⇨ Unused Pins**.

The **Reserve all unused pins** option should be set to **As input tri-stated**. If it is not, then select this option. Click **OK** until all dialog boxes are closed.

Select **Processing ⇨Start Compilation** to begin compiling your project.

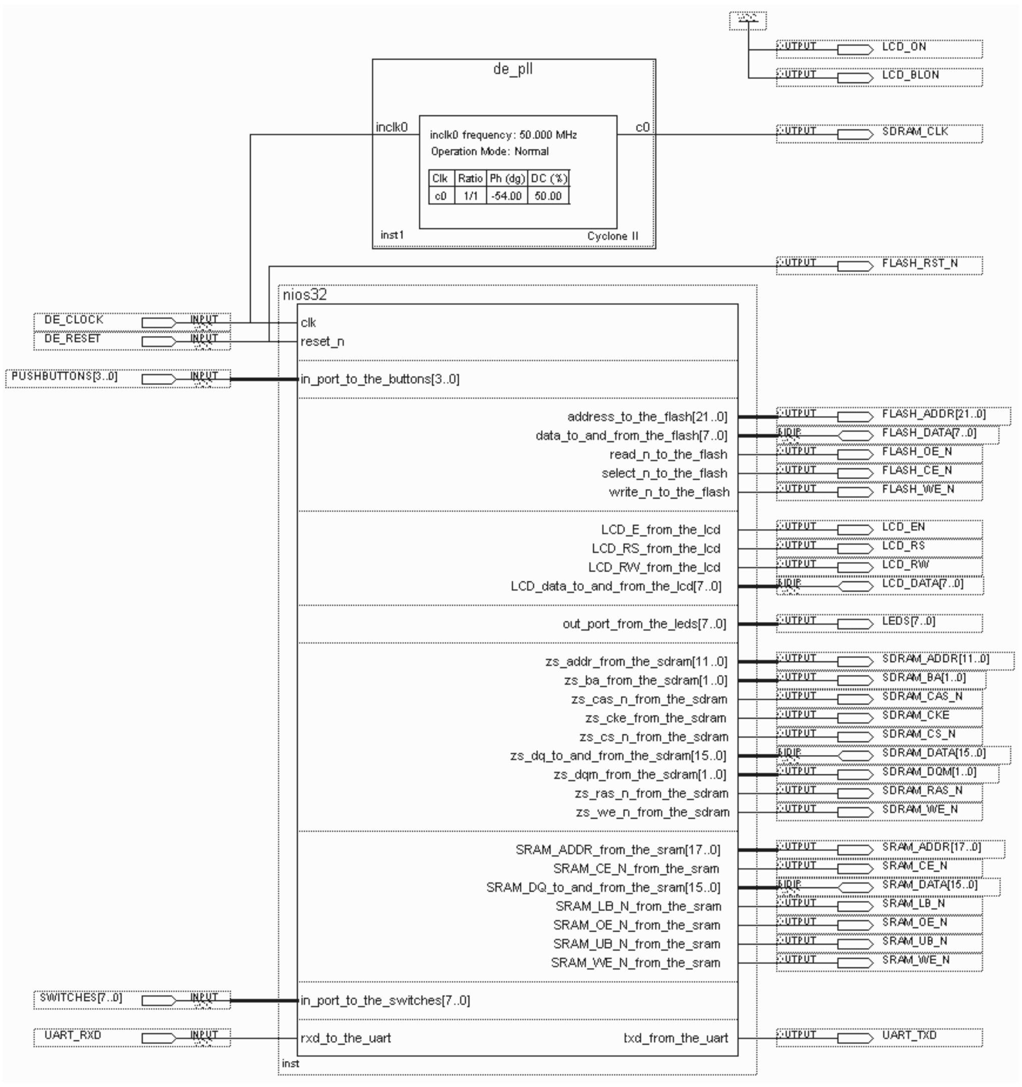


Figure 17.17 The final top-level schematic for the Nios II system on a DE2 board is shown here. The DE1 board schematic is similar except the LCD bus signals will not be present.

17.19 Testing the Nios II Project

To fully test your Nios II project, you will need to write a software program to run on the Nios II processor that tests each component. To complete this task, refer to the previous chapter, which contains *Tutorial III: Nios II Processor Software Design*.

You might want to try your test program from the previous chapter first to verify that memory still works in your new design. After switching to a new workspace for the new project in Nios II IDE, create a blank project with a new system library that is based on your Nios II processor design. You can then import an existing software project into a new design project's software directory using **File ⇨Import**. You will need to clean and rebuild the software project since the system library changes for each new hardware design.

IMPORTANT: In the reference hardware design, SW9 is used as the Nios II processor's reset signal. Before code can be downloaded to the processor, it must be brought out of reset by setting SW9 in the up (or on) position.

ALL SOURCE FILES FOR THIS NIOS II HARDWARE REFERENCE DESIGN

CAN BE FOUND ON THE DVD IN THE \DE*X*\CHAP17 DIRECTORY.

17.20 For additional information

This chapter has provided a brief overview of Nios II hardware development. Additional information can be found at Altera's website (www.altera.com) in the *Nios II Processor Reference Handbook, Embedded Peripherals Handbook* and *Hardware Development Tutorial*. Nios II components for the DE boards and other reference designs can be found at Altera's University Program website. The Nios Community Forum (www.niosforum.com) also contains useful information and downloads for Nios II projects.

17.21 Laboratory Exercises

1. Add two 8-bit PIOs to the Nios II hardware design that connect to the 5 volt I/O pins on the board's header connector. Setup one port for input and one port for output. Connect the PIO port's I/O pins to eight input pins and eight output pins on the header. This is a handy way to interface external devices and sensors like those used in the FPGA robot projects in Chapter 13 to the FPGA board's Nios II processor.

2. Add a PIO port to the Nios II hardware design and use the PIO port's I/O bits to design an I^2C hardware interface to the FPGA board's real-time clock chip. Software will be needed to send I^2C commands, the PIO port just provides a hardware interface to the I^2C SDA and SLC bits (see Section 12.4).

3. Add a parallel port to the Nios II hardware design. Use two 8-bit ports, one for data and one for status and control bits. Connect the PIO port's I/O bits to the parallel port connector on the FPGA board. Software will be needed to monitor and control the handshake lines (see Section 12.1) when connecting to a device like a parallel printer.

4. Add an SPI interface to the Nios II hardware design and use it to interface to an external SPI device connected to one of the FPGA board's expansion connectors.

5. Implement one of the FPGA robotics projects from Chapter 13 using a Nios II processor running C code. See problem 1 for robot interface suggestions.

6. Design an automatic setback HVAC thermostat using the FPGA. Interface a temperature sensor to the FPGA. Some temperature sensors are available with digital outputs that would not require a separate analog-to-digital IC. Display the current time, temperature, heat, fan, and A/C status, and the temperature settings in the LCD. Use the pushbuttons to change the temperature settings and setback times. Use the LEDs to indicate the heat, A/C, and fan control outputs from the thermostat. You can heat the temperature sensor with your finger to cycle the thermostat and cool it with ice or an aerosol spray can of dust off cleaner.

7. Interface a PS/2 keyboard or mouse to the Nios II processor using PIO ports. Write software to demonstrate the new keyboard or mouse interface. Display the output on the LCD or the UART. There are two major options to consider, use the keyboard and mouse cores from Chapter 11 or do everything in software.

8. Use the video sync core and character generation ROM from Chapter 10 to add a video text display to the Nios processor. Add a dual port memory to store a screen full of characters. Write charcters to the dual port memory from the Nios II processor using PIO ports added to the Nios II design. The video system constantly reads the characters out of the dual port memory and then uses the character generation ROM to generate the video display. Write a software driver for the video display and attach a monitor to the FPGA's VGA connector to demonstrate your design.

9. After solving the previous two problems, develop software for a video game that uses the mouse or keyboard for input and displays output on the monitor. If you need graphics for your game, consider replacing the character memory and text display with a larger memory containing only pixels used in a graphics display. Keep in mind that the internal FPGA memory is limited.

10. Add a custom instruction to the Nios II processor designed to speed up a particular application area. See the *Nios II Custom Instruction User Guide*. Demostrate the speedup obtained with the new instruction by running the application with and without the new instruction.

11. Interface the dual port video display memory used in one of the earlier problems directly to the Avalon system bus instead of using PIO ports. See the *Avalon Interface Specification Manual*.

12. Program the FPGA's serial flash device so that your Nios II hardware design loads automatically at power up. See Appendix E for instructions on programming the FPGA's serial flash configuration chip.

13. Program a complete Nios II design into both Flash memories so that the FPGA board loads both the FPGA hardware configuration data and the software from the two Flash memories automatically at power up. See the *Nios II Flash Programmer User Guide* and study the section on how to port the Flash programmer to a new board type. A full version Altera software license is required for Flash programming of Nios II program code.