DKAN0010A

Setting Up a Nios II System with Flash Memory on the DE2

04 November 2009

Introduction

This tutorial details how to set up and instantiate a Nios II system on Terasic Technologies, Inc.'s DE2 Altera Development and Education Board. The system includes an interface to the board's 4MB flash memory chip and runs the application program from flash. It also sets up and implements the LCD, seven-segment displays, red and green LEDs, switches, and pushbuttons.

Background

Nios II is a 32-bit RISC CPU designed for implementation as a soft core in Altera FPGAs. Altera's System-on-a-Programmable-Chip (SOPC) Builder allows users to design Nios II systems easily by selecting Nios II processors, setting up the associated memory, and adding any desired standard and/or custom peripherals. Once generated, the system is incorporated into an FPGA design with Altera's Quartus II software and instantiated on the FPGA.

Altera provides the Nios II Integrated Development Environment (IDE) for application software development. This tutorial includes instructions for programming an example C application into flash and running it on the Nios II.

Application

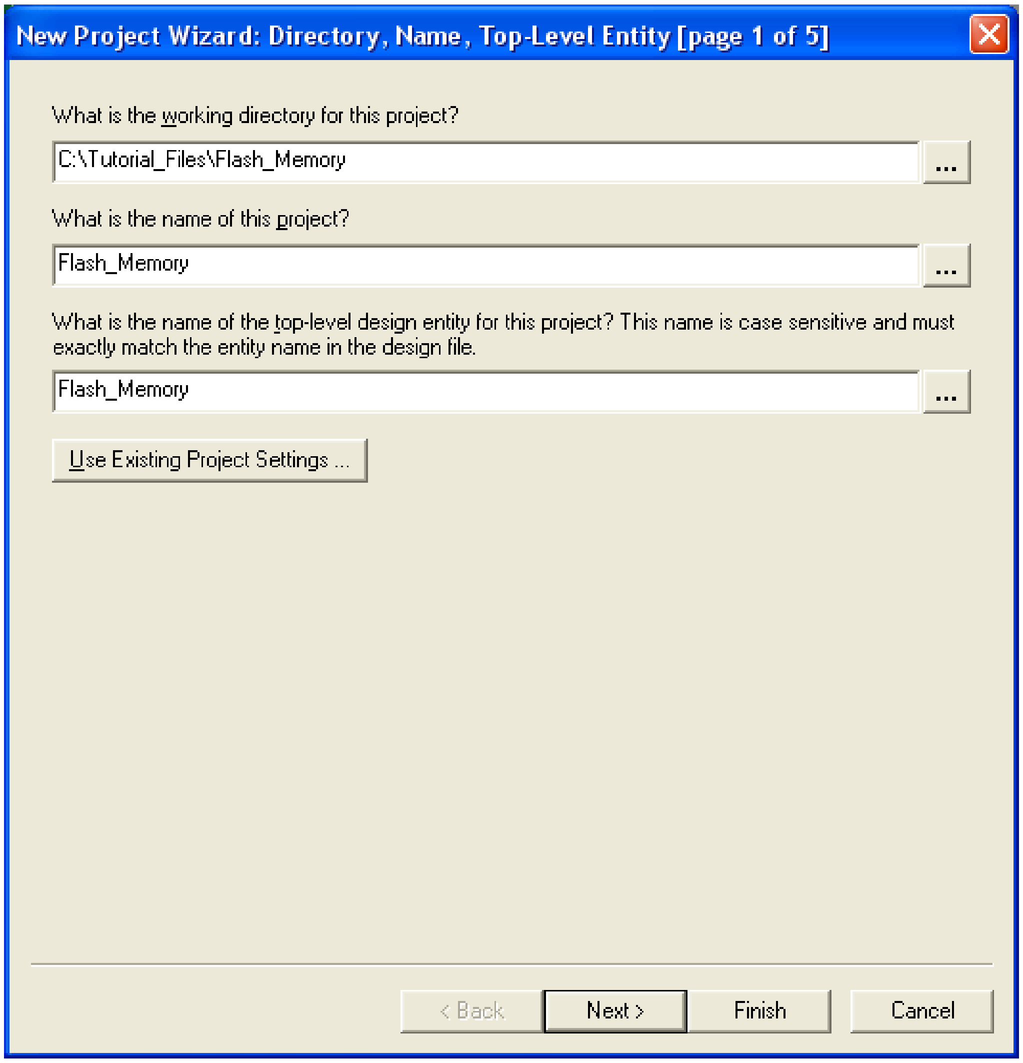
These programs are case sensitive. If a component is named or renamed, it needs to match what is written in this tutorial exactly, or it may not work.

DE2 Setup

This tutorial assumes that the user is familiar with the DE2 board and already has the USB-Blaster device installed. Refer to the *Getting Started with Altera's DE2 Board* tutorial for more information on installing the USB-Blaster driver.

Starting a New Project in Quartus II

Open the Quartus II software and create a new project by selecting **File > New Project Wizard**. Create a folder on the C:\ drive called **Tutorial_Files,** and create another folder inside of that one called **Flash_Memory**. Specify the **Flash_Memory** folder as the working directory for this project. Also, in the second textbox, call the project name **Flash_Memory**. The top level entity automatically fills in. See Figure 1.


New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]
...
Use Existing Project Settings ...
< Back
Next >
Finish
Cancel

What is the working directory for this project?
C:\Tutorial_Files\Flash_Memory

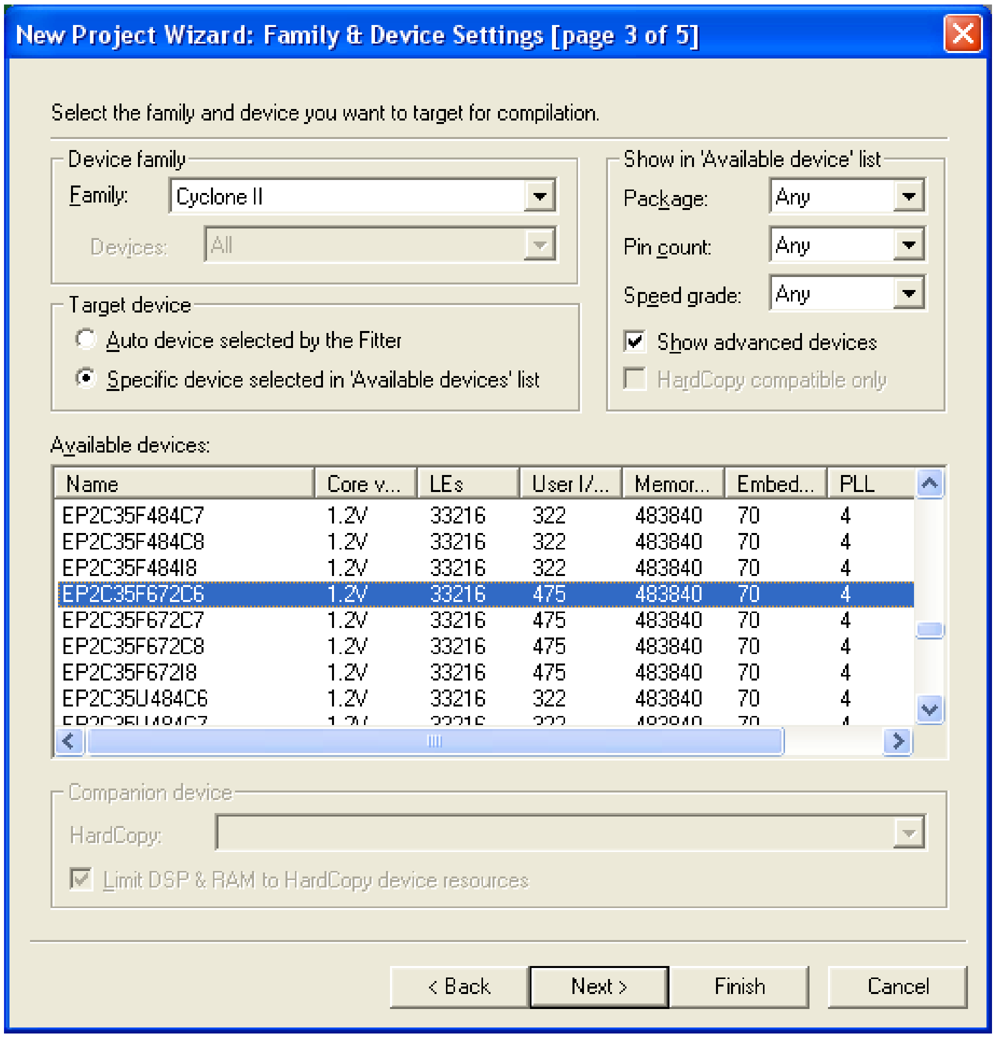
What is the name of this project?
Flash_Memory

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.
Flash_Memory

Figure 1. New Project Wizard

Make sure the new folders were created directly on the C:\ drive and that there are no spaces in the folder names. Folders created in the My Documents folder cause errors. For instance, *C:\Documents and Settings\Firstname_Lastname\My Documents* causes an error, because there are spaces in the directory path.

Click the **Next** button, and click **Next** again on page 2. On the third page, select the **EP2C35F672C6** chip as the target device, which can be seen in Figure 2. This is the FPGA on the DE2 board. Click **Next** on the remaining two screens and click **Finish**.



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

Select the family and device you want to target for compilation.

Device family
Family: Cyclone II
Devices: All

Show in 'Available device' list
Package: Any
Pin count: Any
Speed grade: Any
[✓] Show advanced devices
[ ] HardCopy compatible only

Target device
( ) Auto device selected by the Fitter
(●) Specific device selected in 'Available devices' list

Available devices:

Name	Core v...	LEs	User I/...	Memor...	Embed...	PLL
EP2C35F484C7	1.2V	33216	322	483840	70	4
EP2C35F484C8	1.2V	33216	322	483840	70	4
EP2C35F484I8	1.2V	33216	322	483840	70	4
EP2C35F672C6	1.2V	33216	475	483840	70	4
EP2C35F672C7	1.2V	33216	475	483840	70	4
EP2C35F672C8	1.2V	33216	475	483840	70	4
EP2C35F672I8	1.2V	33216	475	483840	70	4
EP2C35U484C6	1.2V	33216	322	483840	70	4

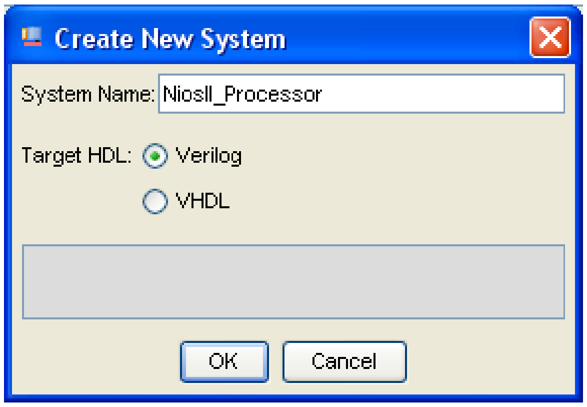
Companion device
HardCopy:
[✓] Limit DSP & RAM to HardCopy device resources

Figure 2. FPGA Chip Selection

Create a new block diagram by selecting **File > New**, and select **Block Diagram/Schematic File**. Save this file by selecting **File > Save As** (not to be confused with Save Project), call it **Flash_Memory**, and click **Save**. Naming it **Flash_Memory** ensures that this block diagram file is the top-most entity of the project, as specified in Figure 1.

Nios II System Design in SOPC Builder

The next step is to build a Nios II processor system. Select **Tools > SOPC Builder** to open the SOPC Builder application. Set the **System Name** to **NiosII_Processor**. Select either Verilog or VHDL as the **Target HDL,** and select **OK**. It is not necessary to know VHDL or Verilog to continue. See Figure 3.



System Name: NiosII_Processor

Target HDL: [✓ Verilog] VHDL

Figure 3. Create New Nios II System

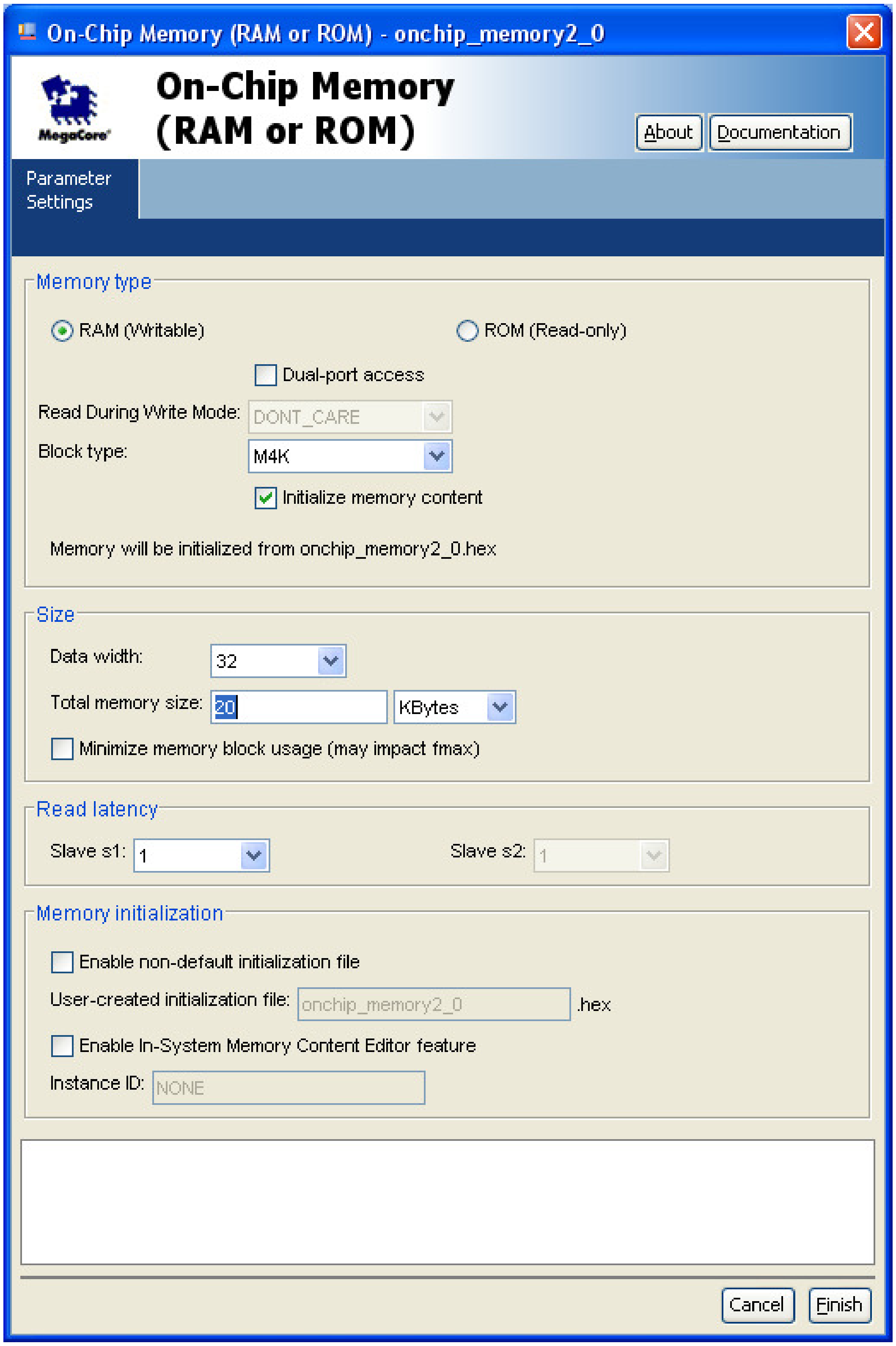
Error and warning messages appear and disappear sporadically in the information box on the bottom of the screen while performing the following steps. Ignore these errors; they disappear upon successful completion of the Nios II system.

In the **Clock Settings** pane, double click **clk_0** and rename it **CLOCK_50**. Press **Enter**.

Add On-Chip Memory

On the upper left side of the SOPC Builder window, under **Component Library,** expand the **Memories and Memory Controllers** column, then expand **On-Chip**, and select **On-Chip Memory (RAM or ROM)**. Click the **Add** button, and a MegaWizard appears. Set **Block Type** to **M4K,** set **Total Memory Size** to **20,** and select **Kbytes**. Do not change any other default settings. Click **Finish**. Refer to Figure 4.

Errors may occur after adding the On-chip memory. Ignore these errors, since they are corrected as more components are added to the system.



On-Chip Memory (RAM or ROM)

Memory type

[✓ RAM (Writable)] ROM (Read-only)

☐ Dual-port access

Read During Write Mode: DONT_CARE

Block type: M4K

✓ Initialize memory content

Memory will be initialized from onchip_memory2_0.hex

Size

Data width: 32

Total memory size: 20 KBytes

☐ Minimize memory block usage (may impact fmax)

Read latency

Slave s1: 1 Slave s2: 1

Memory initialization

☐ Enable non-default initialization file

User-created initialization file: onchip_memory2_0 .hex

☐ Enable In-System Memory Content Editor feature

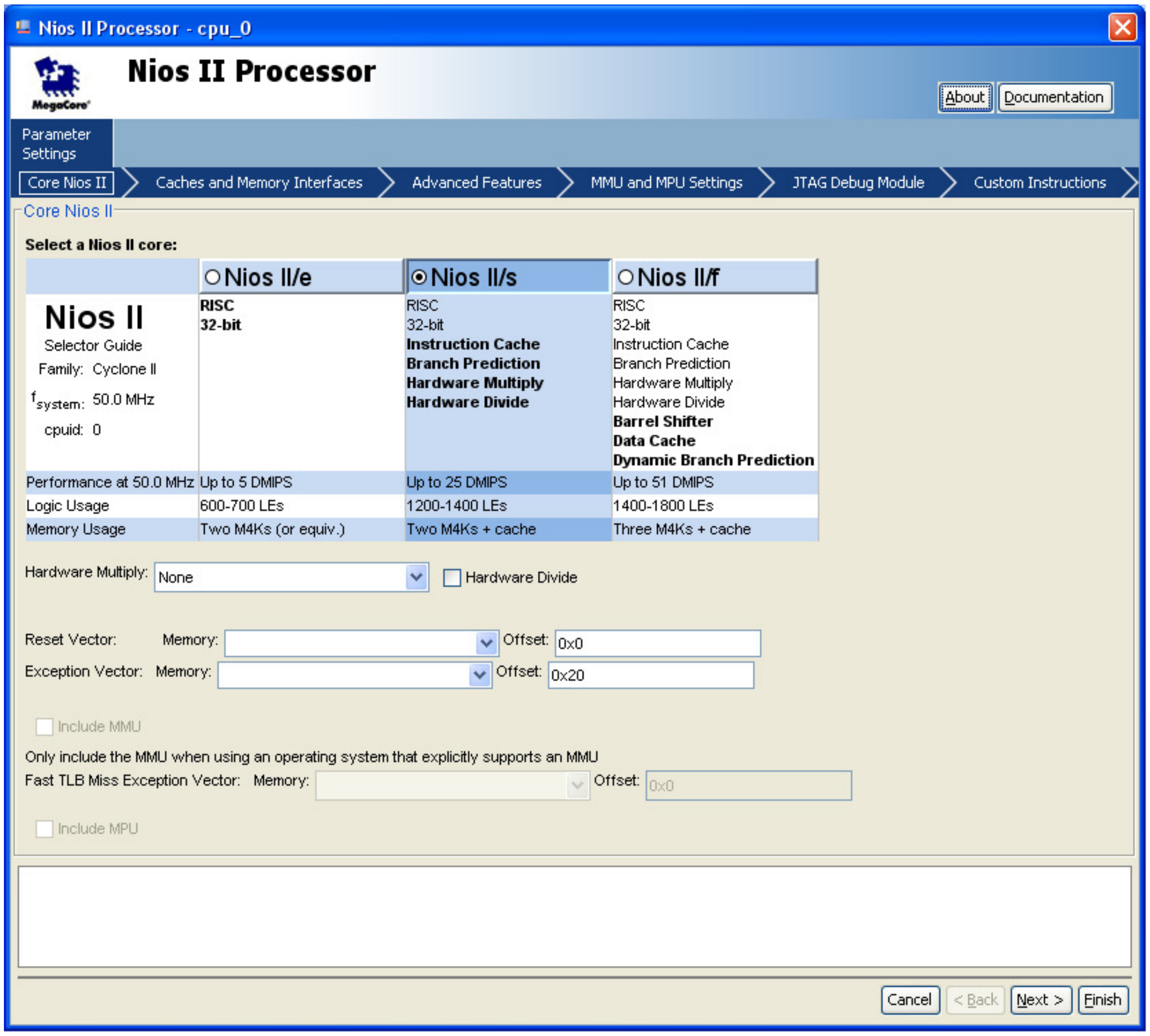
Instance ID: NONE

Figure 4. On-Chip Memory Setup

Add a Nios II Processor

Now, add the Nios II processor to the system. In the upper left corner of the SOPC window, under **Component Library**, select **Nios II Processor**, and click the **Add** button**.**

In the middle of the MegaWizard window, select **Nios II/s** as a Nios II core. Set **Hardware Multiply** to **None**. Do not change any other default value. Click **Finish,** and the Nios II processor is added to the system. See Figure 5.



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: None ☐ Hardware Divide

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

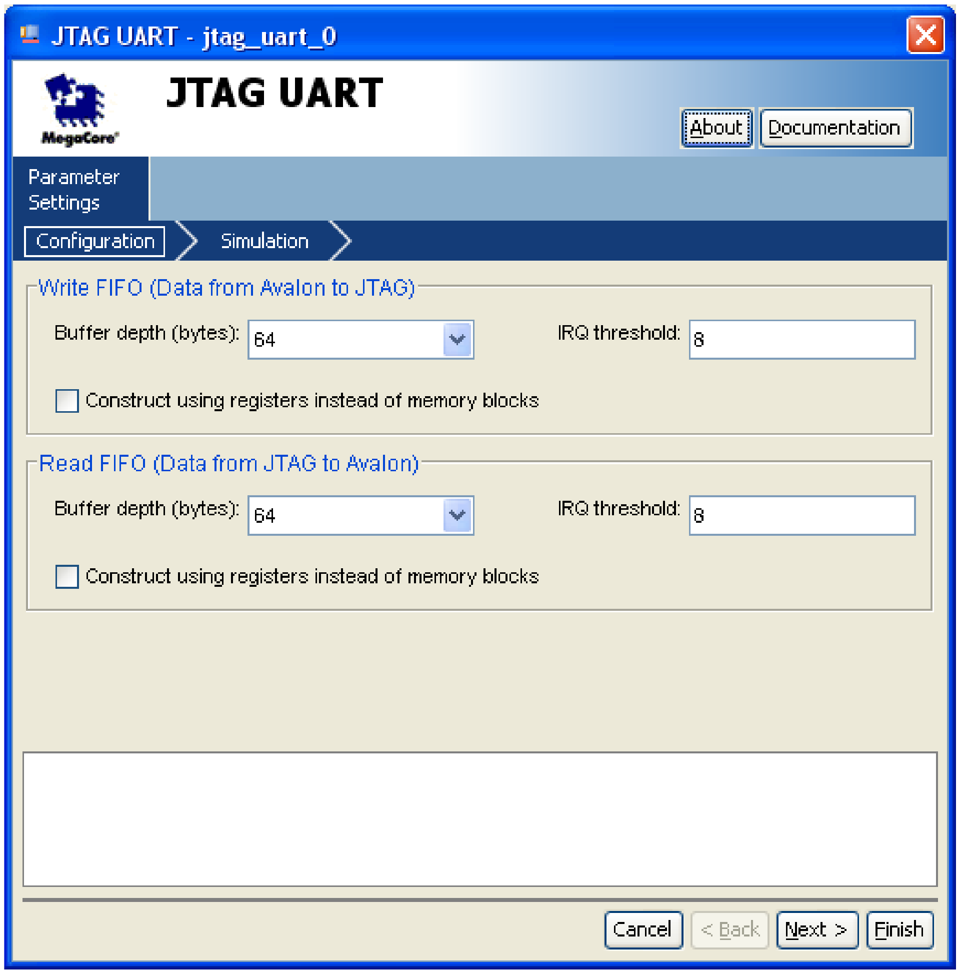
☐ Include MMU
Only include the MMU when using an operating system that explicitly supports an MMU
Fast TLB Miss Exception Vector: Memory: Offset: 0x0

☐ Include MPU

Figure 5. Nios II Processor Setup

Add a JTAG UART

In order to communicate to the Nios II processor through the USB-Blaster, add a JTAG UART to the system. In the upper left corner of the SOPC window, under **Component Library**, expand **Interface Protocols,** expand **Serial,** select **JTAG UART,** and click **Add**. Do not change any of the default settings. Click **Finish**. Refer to Figure 6.



Write FIFO (Data from Avalon to JTAG)
Buffer depth (bytes): 64
IRQ threshold: 8
☐ Construct using registers instead of memory blocks

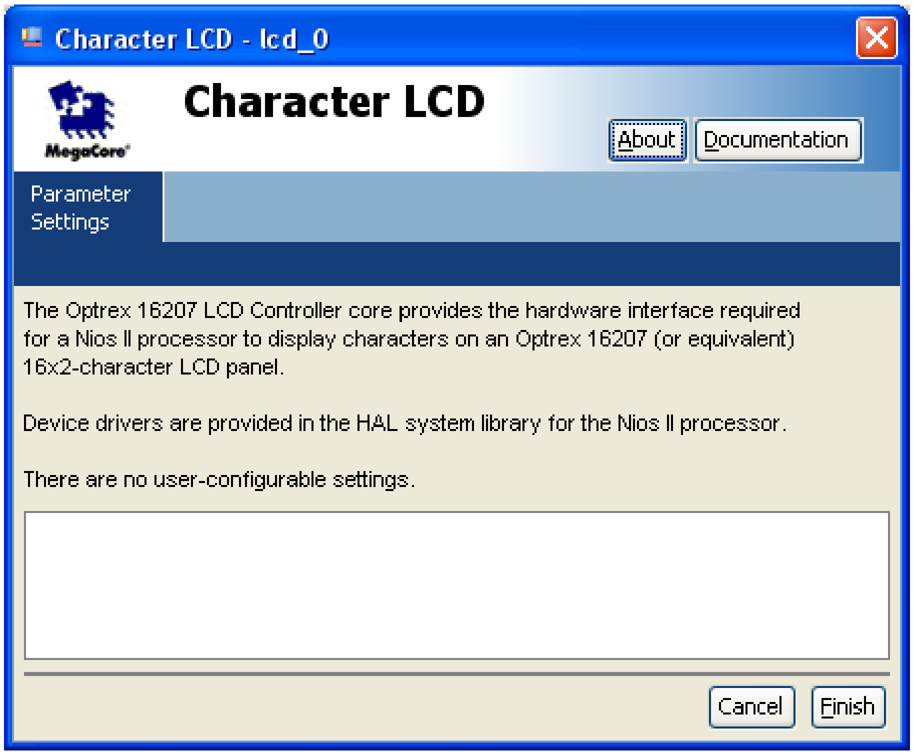
Read FIFO (Data from JTAG to Avalon)
Buffer depth (bytes): 64
IRQ threshold: 8
☐ Construct using registers instead of memory blocks

Figure 6. JTAG UART Setup

Add an LCD

To add the LCD, expand **Peripherals**, then expand **Display**, select **Character LCD**, and click **Add**. There are no settings to modify, so select **Finish**. Refer to Figure 7.

Rename the **lcd_0** that was generated to ensure proper functionality. Select **lcd_0**, right-click on it, select **Rename**, rename it to **lcd**, and press **Enter**.



The Optrex 16207 LCD Controller core provides the hardware interface required for a Nios II processor to display characters on an Optrex 16207 (or equivalent) 16x2-character LCD panel.

Device drivers are provided in the HAL system library for the Nios II processor.

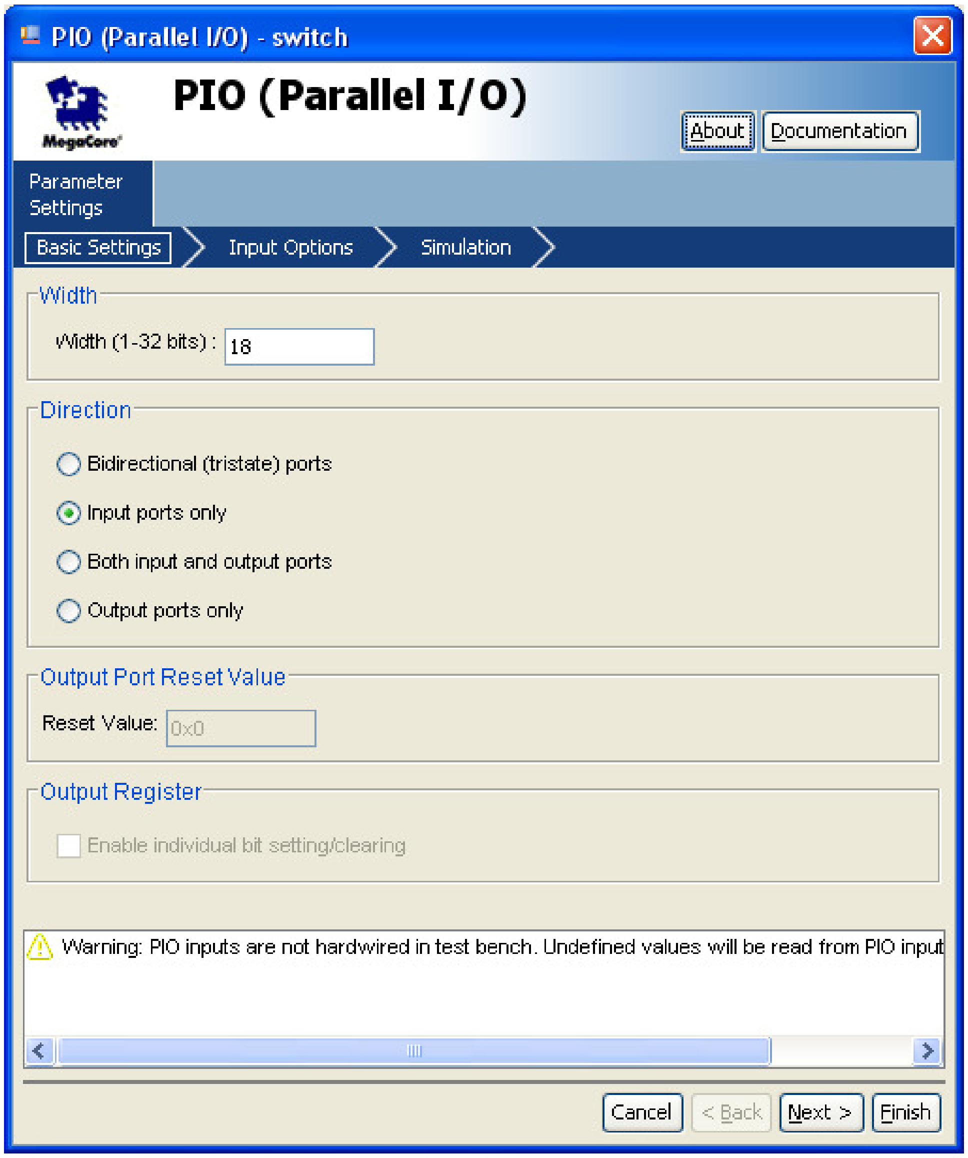
There are no user-configurable settings.

Figure 7. LCD Controller Setup

Add Parallel I/O (PIO)

Add the switches to the system. Expand **Peripherals**, then expand **Microcontroller Peripherals**, select **PIO (Parallel I/O)**, and click **Add**. When the window opens, set the **Width** to **18**, set the **Direction** to **input ports only**, and click **Finish**. Refer to Figure 8.

Rename the **pio_0** that was generated to ensure proper functionality. Select **pio_0**, right-click on it, select **Rename**, rename it to **switch,** and press **Enter**.



Width

Width (1-32 bits) : 18

Direction

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

Output Port Reset Value

Reset Value: 0x0

Output Register

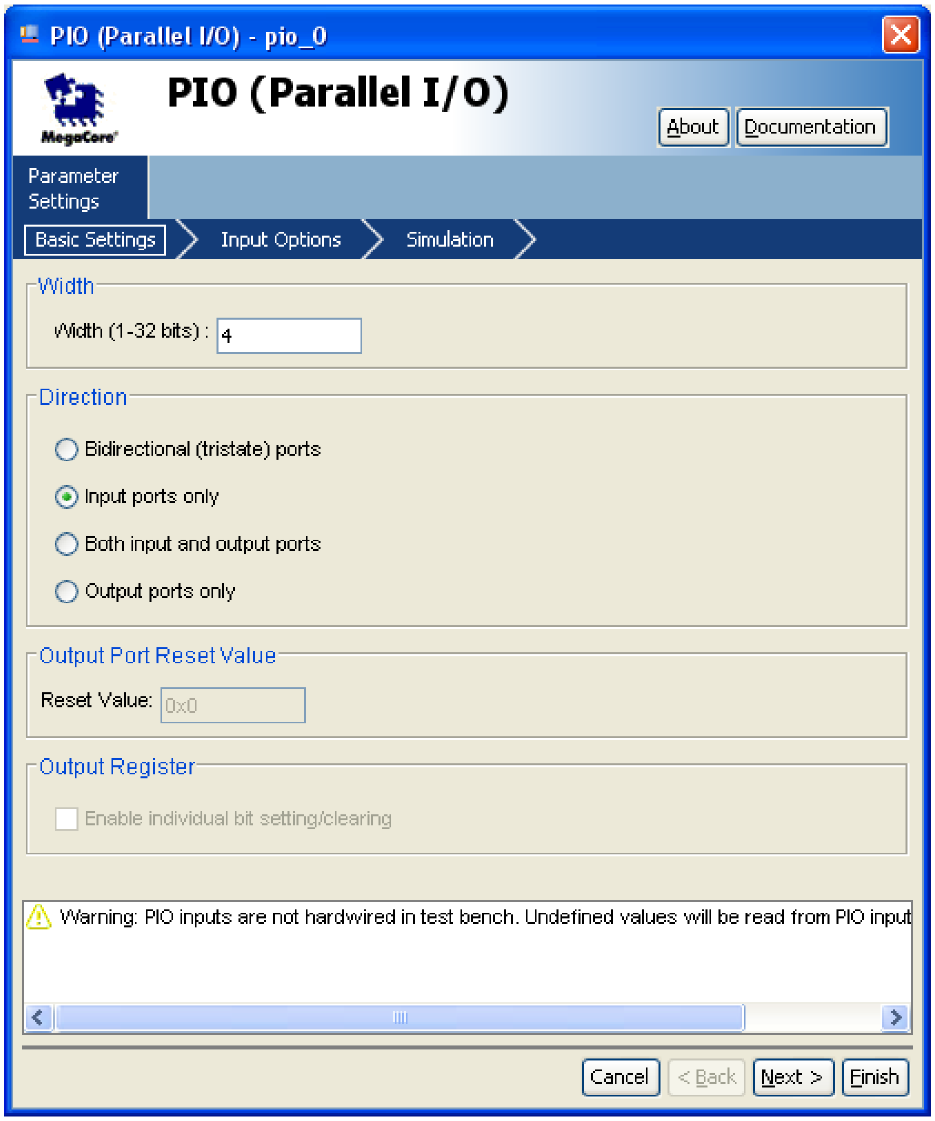
Enable individual bit setting/clearing

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

Figure 8. PIO Switch Setup

Add the pushbuttons next. Similar to the switches, expand **Peripherals,** expand **Microcontroller Peripherals,** select **PIO (Parallel I/O),** and click **Add**. When the window opens, set the **Width** to **4** and set the **Direction** to **input ports only**. Go to the **Input Options** tab, check the **synchronously capture box,** and select **Falling edge**. Also, underneath in the **Interrupt** section, check **Generate IRQ**, select **Edge**, and then click **Finish**. Refer to Figures 9 and 10.

Rename the **pio_0** that was generated to ensure proper functionality. Select **pio_0**, right-click on it, select **Rename**, rename it to **pb**, and press **Enter**.



Width
Width (1-32 bits) : 4

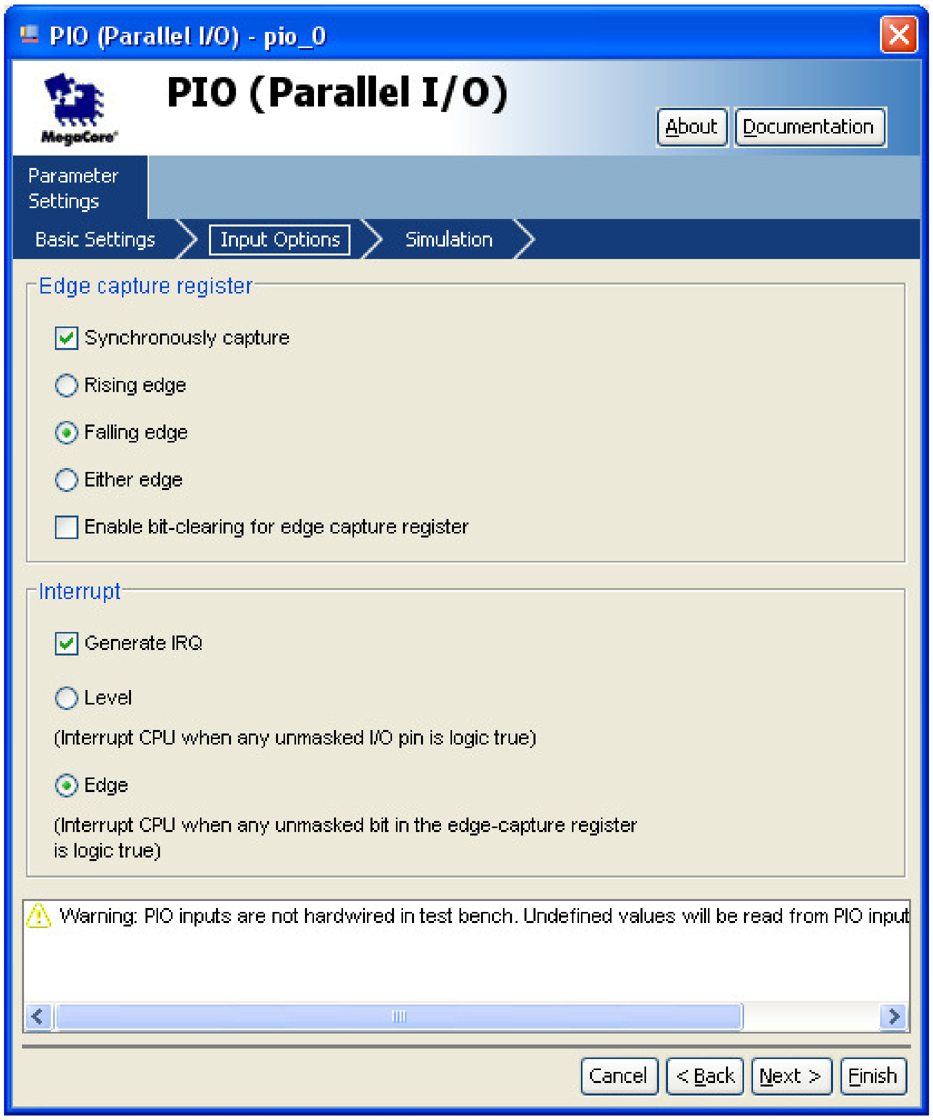
Direction
( ) Bidirectional (tristate) ports
(●) Input ports only
( ) Both input and output ports
( ) Output ports only

Output Port Reset Value
Reset Value: 0x0

Output Register
[ ] Enable individual bit setting/clearing

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

Figure 9. **PIO Pushbutton Setup**



Edge capture register

[✓] Synchronously capture

( ) Rising edge

(●) Falling edge

( ) Either edge

[ ] Enable bit-clearing for edge capture register

Interrupt

[✓] Generate IRQ

( ) Level

(Interrupt CPU when any unmasked I/O pin is logic true)

(●) Edge

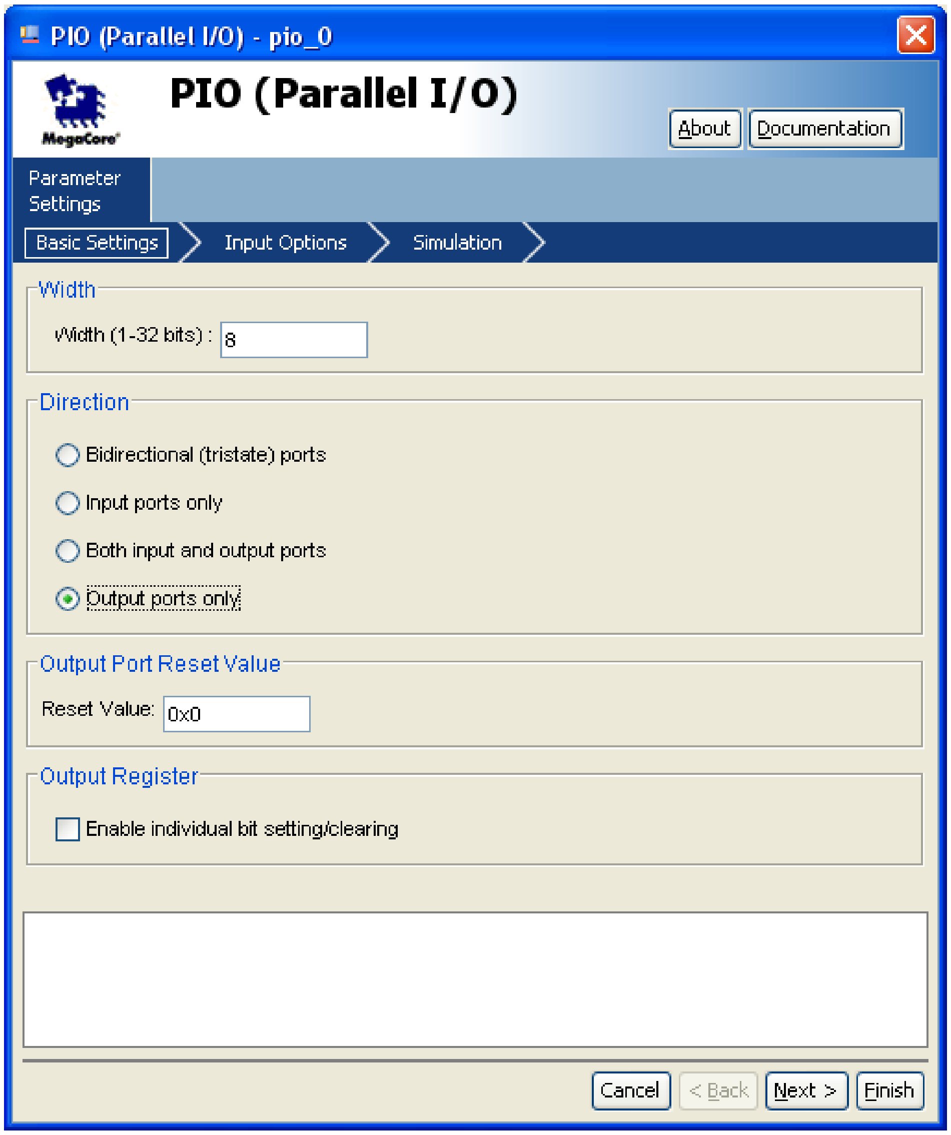
(Interrupt CPU when any unmasked bit in the edge-capture register is logic true)

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

Figure 10. PIO Pushbutton Interrupt Setup

The green LEDs are added next. Expand **Peripherals**, expand **Microcontroller Peripherals**, select **PIO (Parallel I/O)**, and click **Add**. Do not change any of the default settings. The interface is set up for an 8-bit output-only PIO, which is needed to use the green LEDs. Click **Finish.** Refer to Figure 11.

To ensure proper functionality, rename the **pio_0** that was generated. Select **pio_0**, right-click on it, select **Rename,** rename it to **ledg**, and press **Enter**.



Width

Width (1-32 bits) : 8

Direction

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

Output Port Reset Value

Reset Value: 0x0

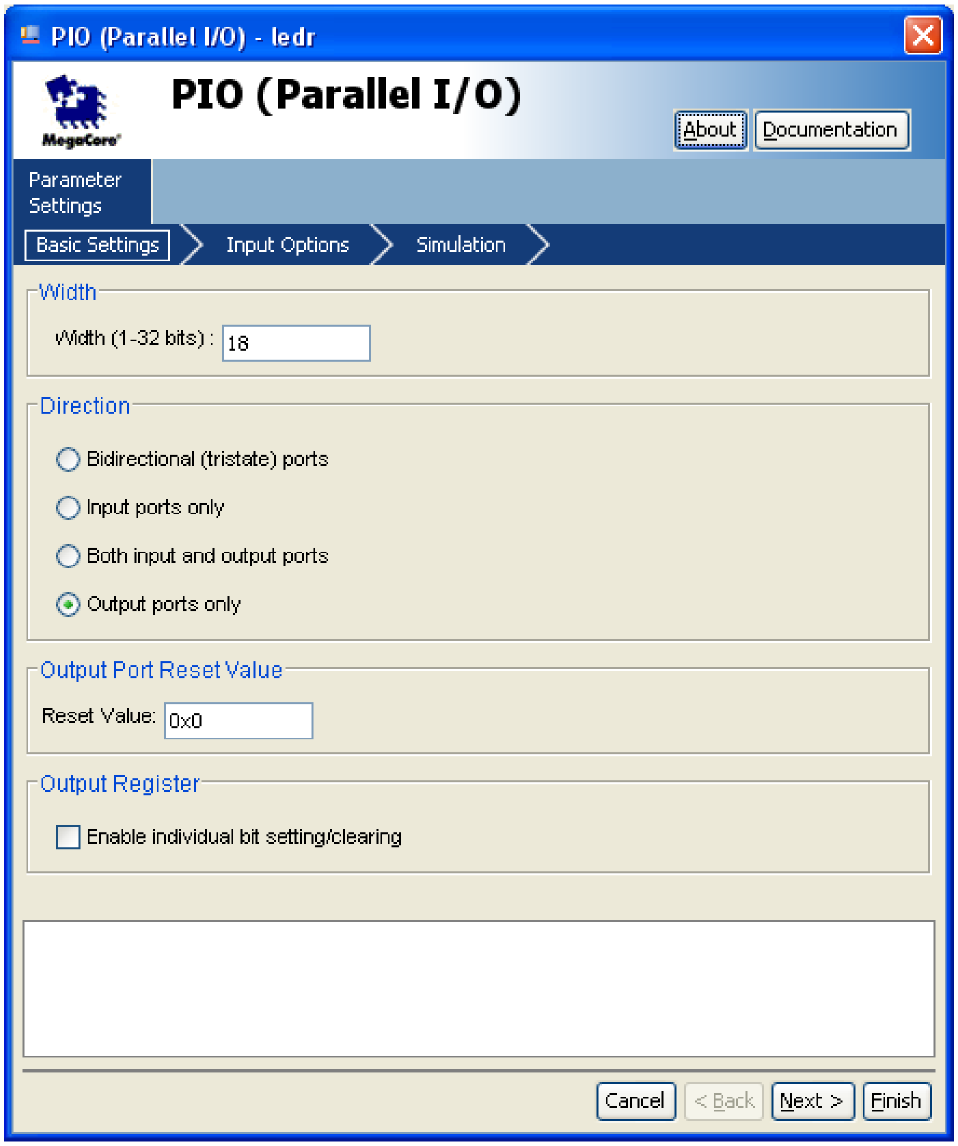
Output Register

- [ ] Enable individual bit setting/clearing

Figure 11. PIO Green LED Setup

Add the red LEDs next. Expand **Peripherals**, expand **Microcontroller Peripherals**, select **PIO (Parallel I/O)**, and click **Add**. Set the **Width** to **18** and click **Finish**. Refer to Figure 12.

To ensure proper functionality, rename the **pio_0** that was generated. Select **pio_0**, right-click on it, select **Rename**, rename it to **ledr**, and press **Enter**.



Width

Width (1-32 bits) : 18

Direction

Bidirectional (tristate) ports

Input ports only

Both input and output ports

[✓ Output ports only]

Output Port Reset Value

Reset Value: 0x0

Output Register

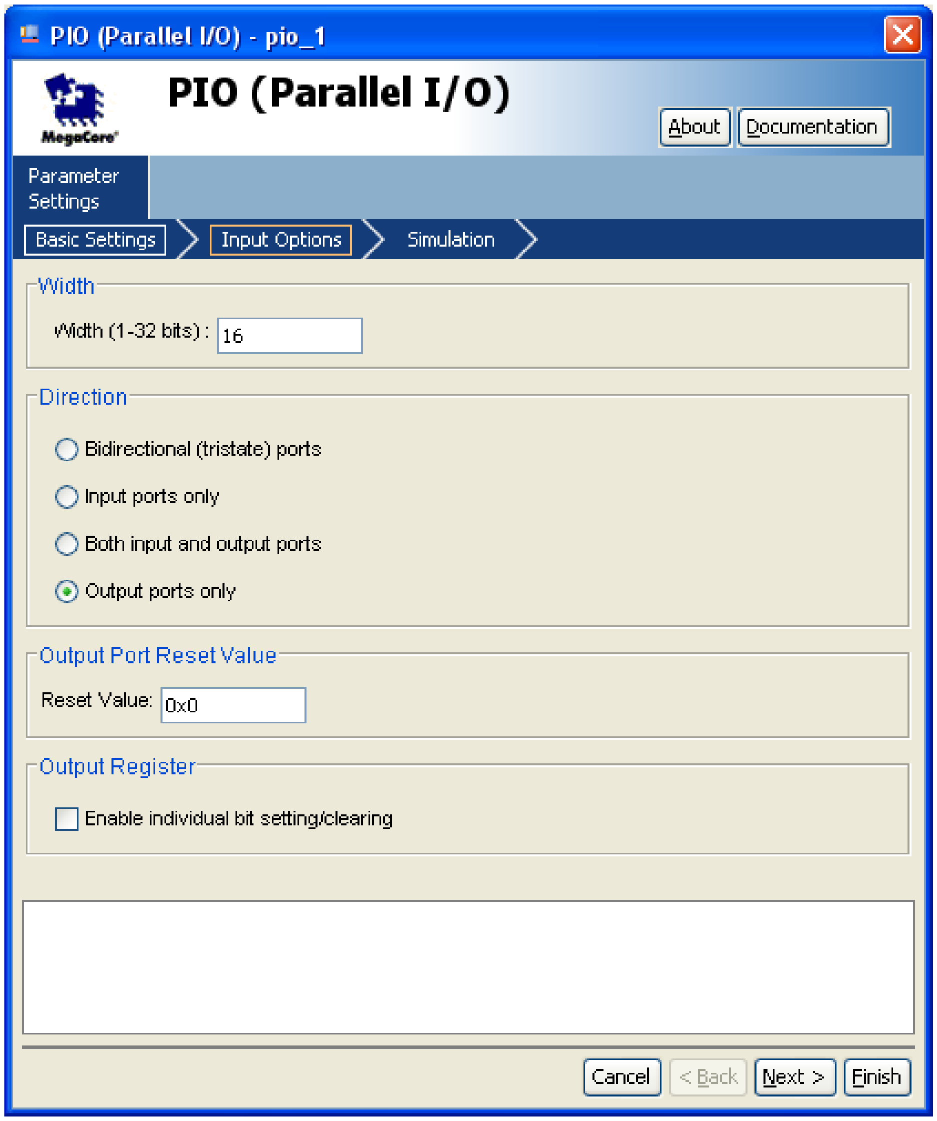
☐ Enable individual bit setting/clearing

Figure 12. PIO Red LED Setup

Add the seven-segment displays to the system. Expand **Peripherals**, expand **Microcontroller Peripherals**, select **PIO (Parallel I/O)**, and click **Add**. Set the **Width** to **16** and click **Finish**.

Repeat the previous step three more times to set up all eight of the seven-segment displays. Refer to Figure 13.

To ensure proper functionality, rename the **pio_0-3** that were generated. Select **pio_0-3**, right-click on them, select **Rename**, and rename them to **seven_seg_01**, **seven_seg_23**, **seven_seg_45**, and **seven_seg_67**.



Width

Width (1-32 bits) : 16

Direction

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

Output Port Reset Value

Reset Value: 0x0

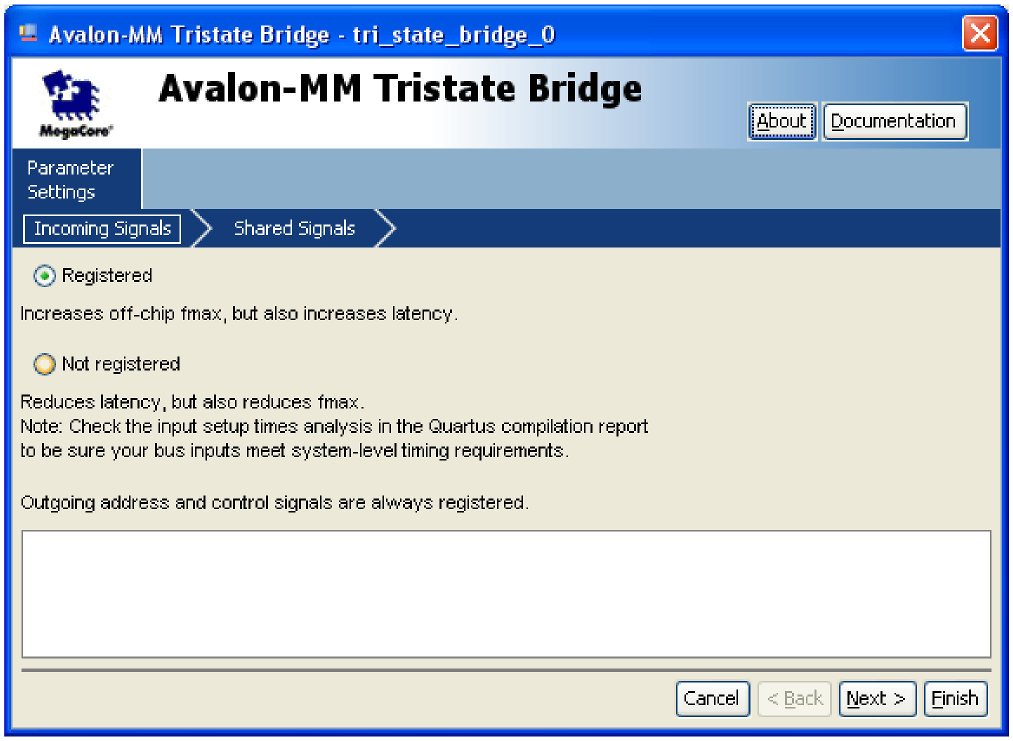
Output Register

☐ Enable individual bit setting/clearing

Figure 13. PIO Seven-Segment Display Setup

Add Flash Memory

To add the external flash memory chip to the system, add the Avalon-MM Tristate Bridge, which allows the Nios II processor to interface with the flash memory. Expand **Bridges and Adaptors**, then expand **Memory Mapped**, select **Avalon-MM Tristate Bridge**, and click **Add**. Leave the default settings and click **Finish**. Refer to Figure 14.



[✓ Registered]

Increases off-chip fmax, but also increases latency.

Not registered

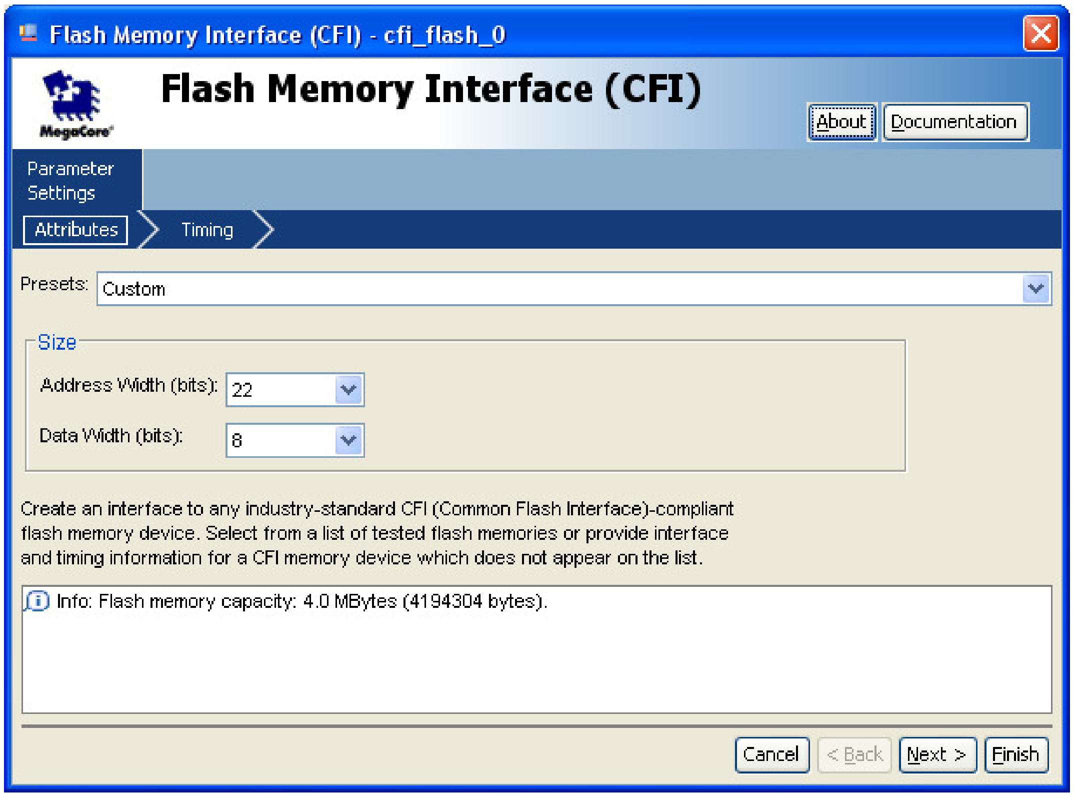
Reduces latency, but also reduces fmax.
Note: Check the input setup times analysis in the Quartus compilation report to be sure your bus inputs meet system-level timing requirements.

Outgoing address and control signals are always registered.

Figure 14. Tristate Bridge Setup

Next, add the flash memory to the system. Expand **Memories and Memory Controllers,** then expand **Flash**, select **Flash Memory Interface (CFI)**, and click **Add**.

Set the **presets** to **custom**, **address width** to **22** bits, and the **Data** to **8** bits. Click on the **Timing** tab and set the **setup** to **40**, the **wait** to **160**, and the **hold** to **40**. For **units** use **ns**. Click **Finish**. See Figures 15 and 16.



Presets: Custom

Size

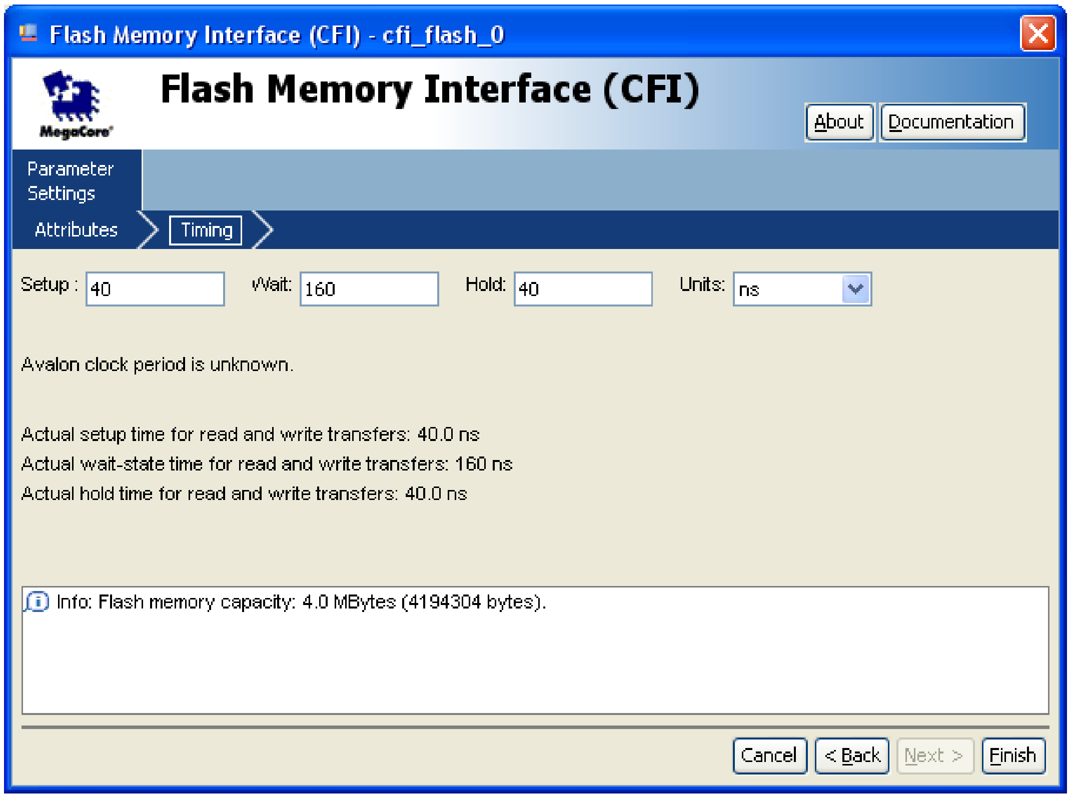
Address Width (bits): 22

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).

Figure 15. Flash Memory Setup



Setup : 40 Wait: 160 Hold: 40 Units: ns

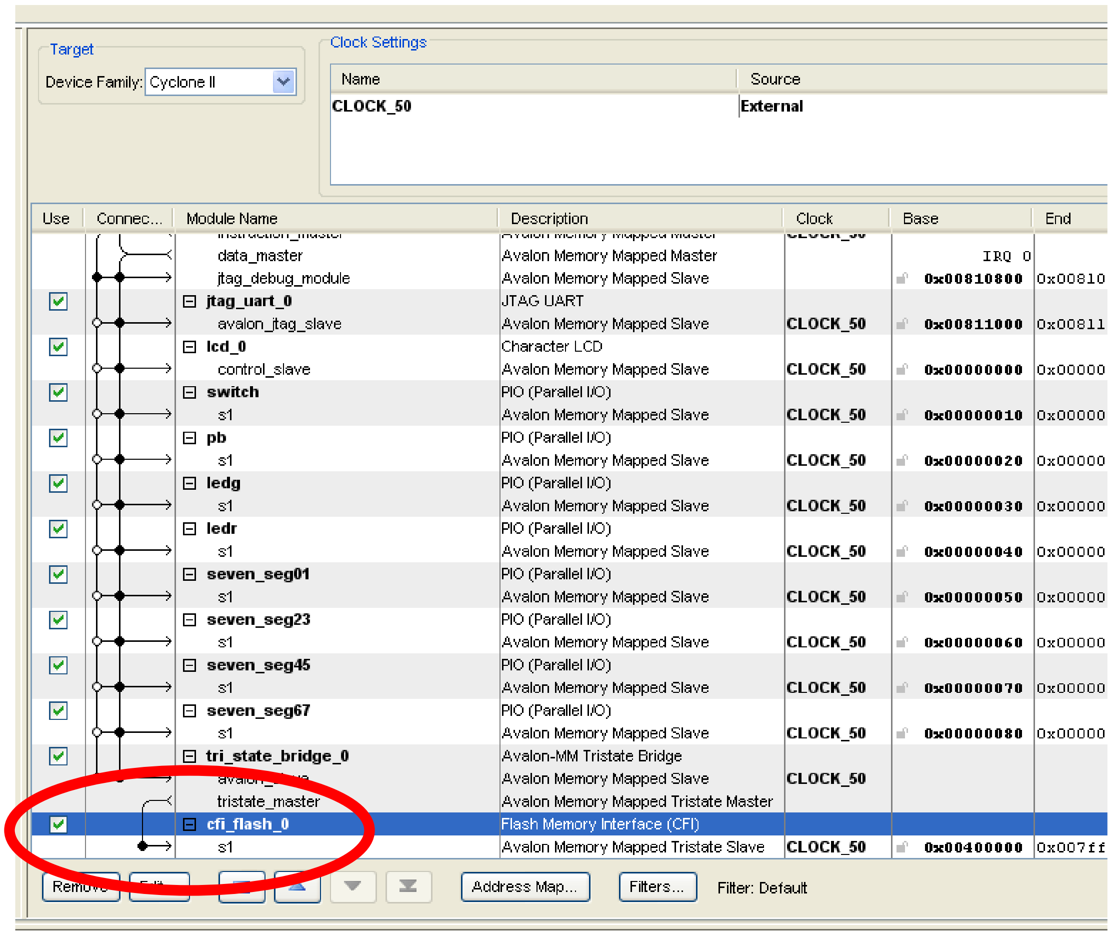
Avalon clock period is unknown.

Actual setup time for read and write transfers: 40.0 ns
Actual wait-state time for read and write transfers: 160 ns
Actual hold time for read and write transfers: 40.0 ns

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

Figure 16. Flash Memory Timing Setup

Connect the flash memory to the Tristate Bridge. Position the mouse over the **Connection** area of the Flash memory. Click the connection circle that connects the **tristate_master** to **s1**. Refer to Figure 17.



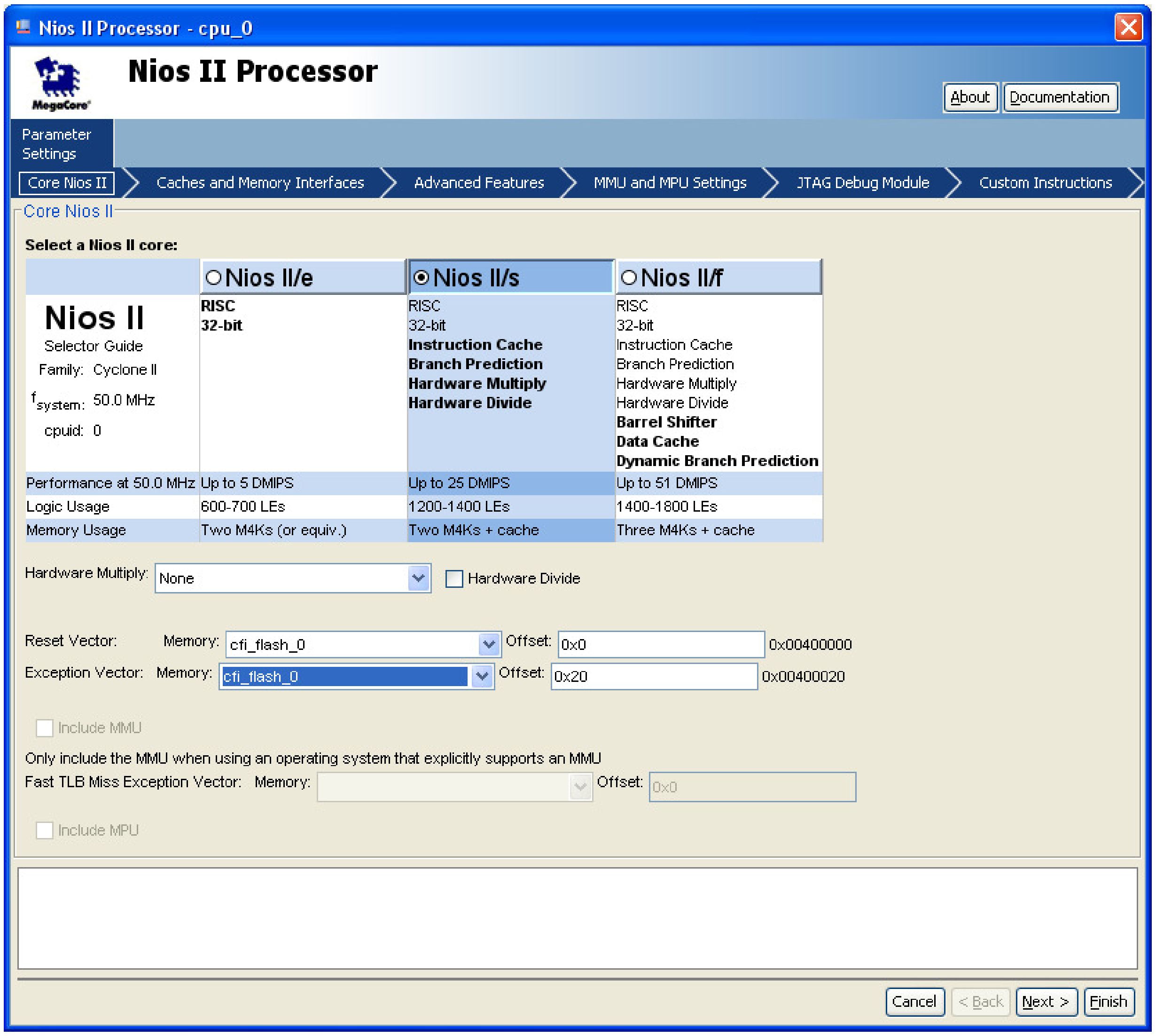
Device Family: Cyclone II

Name	Source
CLOCK_50	External

Use	Connec...	Module Name	Description	Clock	Base	End
		data_master	Avalon Memory Mapped Master		IRQ 0	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00810800	0x00810
✓		jtag_uart_0	JTAG UART			
		avalon_jtag_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00811000	0x00811
✓		lcd_0	Character LCD			
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00000000	0x00000
✓		switch	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000010	0x00000
✓		pb	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000020	0x00000
✓		ledg	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000030	0x00000
✓		ledr	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000040	0x00000
✓		seven_seg01	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000050	0x00000
✓		seven_seg23	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000060	0x00000
✓		seven_seg45	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000070	0x00000
✓		seven_seg67	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00000080	0x00000
✓		tri_state_bridge_0	Avalon-MM Tristate Bridge			
		avalon_slave	Avalon Memory Mapped Slave	CLOCK_50		
		tristate_master	Avalon Memory Mapped Tristate Master			
✓		cfi_flash_0	Flash Memory Interface (CFI)			
		s1	Avalon Memory Mapped Tristate Slave	CLOCK_50	0x00400000	0x007ff

Figure 17. Tristate Bridge - Flash Memory Connection

To run the application from the flash memory instead of the On-chip memory, specify it in the **cpu_0** MegaWizard. Select **cpu_0** in the Module Name list, right-click on it, and select **Edit**. Set the **Reset Vector** and the **Expansion Vector** to **cfi_flash_0,** as shown in Figure 18. Select **Finish**.



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: None ☐ Hardware Divide

Reset Vector: Memory: cfi_flash_0 Offset: 0x0 0x00400000

Exception Vector: Memory: cfi_flash_0 Offset: 0x20 0x00400020

☐ Include MMU

Only include the MMU when using an operating system that explicitly supports an MMU

Fast TLB Miss Exception Vector: Memory: Offset: 0x0

☐ Include MPU

Figure 18. Flash Memory Configuration

Assign Base Addresses

To avoid conflicts between system components, set their base addresses. To auto-assign the base addresses, select **System > Auto-Assign Base Addresses**. Refer to Figure 19.

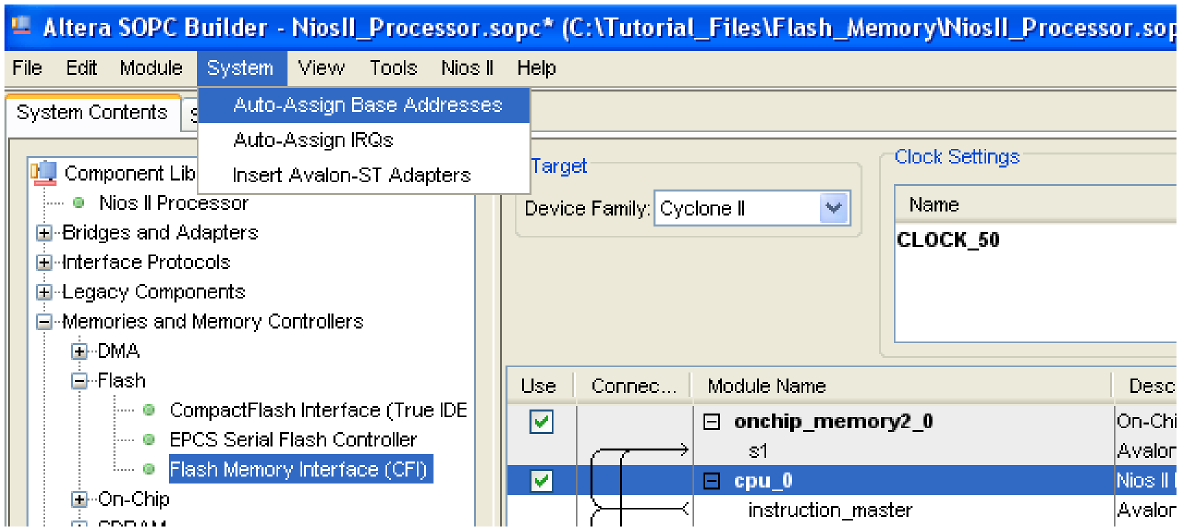
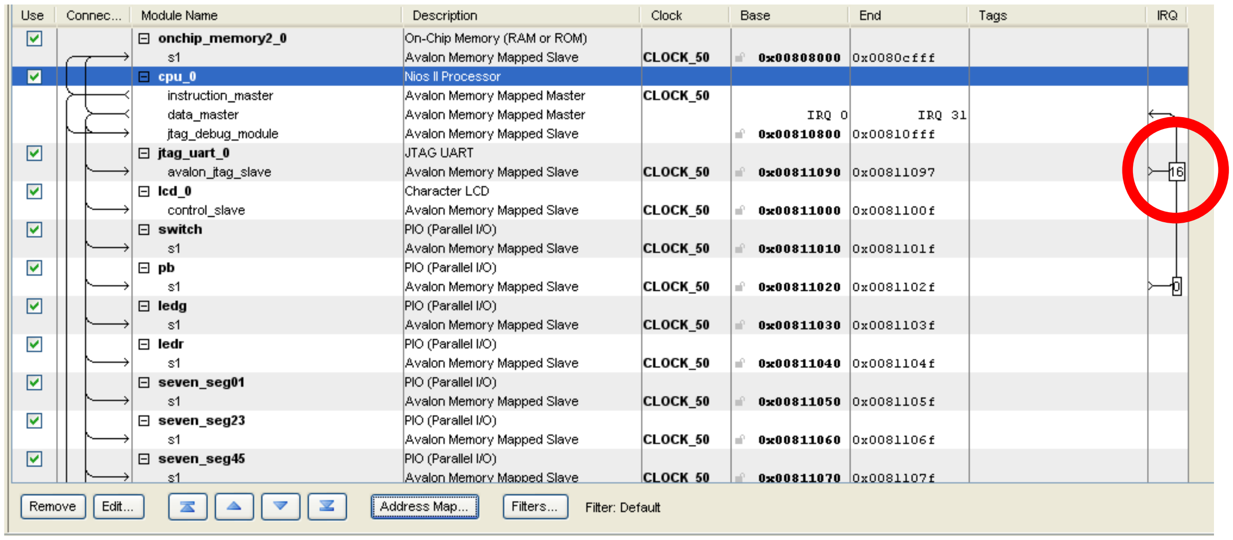


Figure 19. Auto-Assign Base Addresses

Set Interrupt Request Priority

Change the interrupt request (IRQ) priorities for the JTAG UART. Click the **IRQ** value for the **jtag_uart_0** component to select it. Type **16**, and press **Enter** to assign it a new IRQ value. Refer to Figure 20.



Use	Connec...	Module Name	Description	Clock	Base	End	Tags	IRQ
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00808000	0x0080cfff		
✓		cpu_0	Nios II Processor					
		instruction_master	Avalon Memory Mapped Master	CLOCK_50				
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31		
		jtag_debug_module	Avalon Memory Mapped Slave		0x00810800	0x00810fff		
✓		jtag_uart_0	JTAG UART					
		avalon_jtag_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00811090	0x00811097		16
✓		lcd_0	Character LCD					
		control_slave	Avalon Memory Mapped Slave	CLOCK_50	0x00811000	0x0081100f		
✓		switch	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811010	0x0081101f		
✓		pb	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811020	0x0081102f		0
✓		ledg	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811030	0x0081103f		
✓		ledr	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811040	0x0081104f		
✓		seven_seg01	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811050	0x0081105f		
✓		seven_seg23	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811060	0x0081106f		
✓		seven_seg45	PIO (Parallel I/O)					
		s1	Avalon Memory Mapped Slave	CLOCK_50	0x00811070	0x0081107f		

Figure 20. JTAG UART IRQ Setup

Generate System

Finally, generate the system. Click on the **Generate** button on the bottom of the screen. If prompted to save the changes, do so. When the system generation is complete, a message entitled **Info: System generation was successful** appears in the message box. Upon successful system generation, close the SOPC Builder window.

Block Diagram Design in Quartus II

Add Processor

Now that the SOPC system is built, implement it in the block diagram file. Open the Quartus II window. Right-click on the blank block diagram within the **Flash_Memory.bdf** tab. Click **Insert > Symbol**. A **Symbol** window opens. In the **Libraries** pane, expand **Project**, select **NiosII_Processor**, and click **OK**. Refer to Figure 21. Click to place the symbol somewhere in the block diagram window.

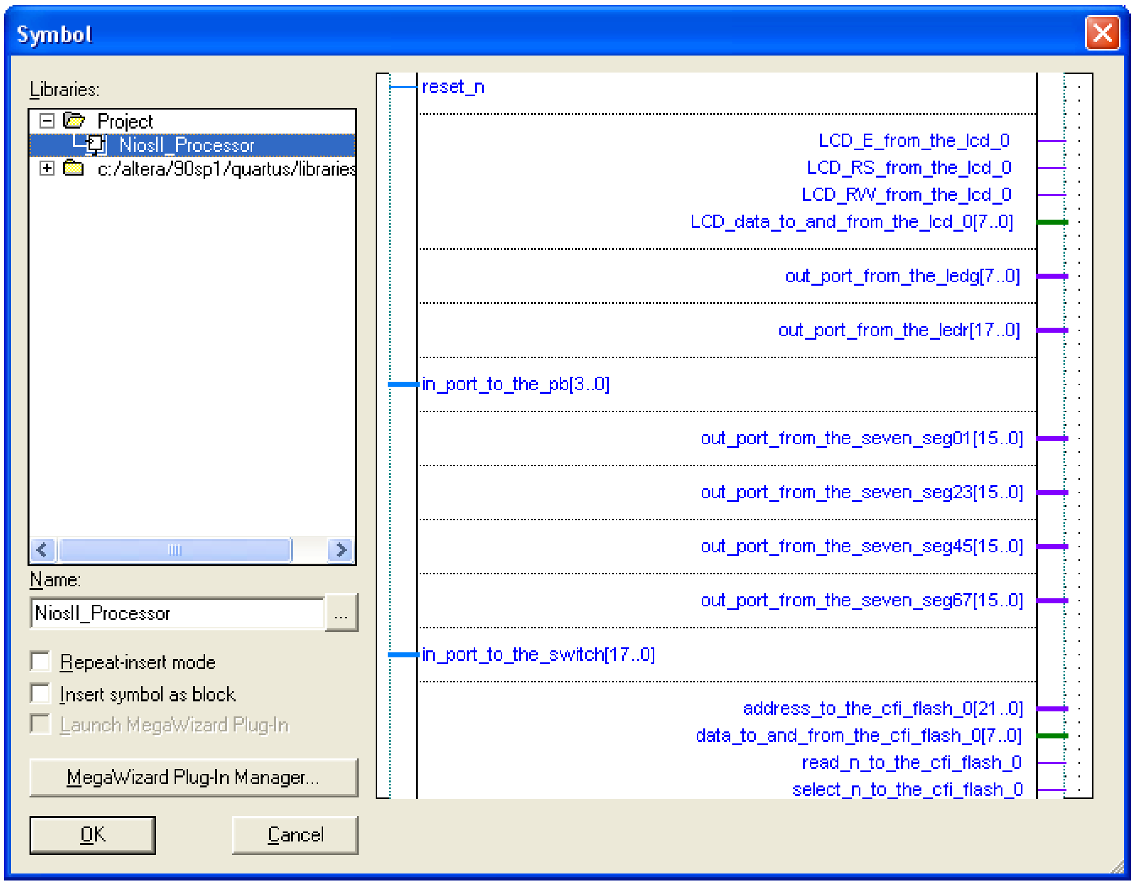

Symbol
Libraries:
Project
NiosII_Processor
c:/altera/90sp1/quartus/libraries
Name:
NiosII_Processor
Repeat-insert mode
Insert symbol as block
Launch MegaWizard Plug-In
MegaWizard Plug-In Manager...
OK
Cancel
reset_n
LCD_E_from_the_lcd_0
LCD_RS_from_the_lcd_0
LCD_RW_from_the_lcd_0
LCD_data_to_and_from_the_lcd_0[7..0]
out_port_from_the_ledg[7..0]
out_port_from_the_ledr[17..0]
in_port_to_the_pb[3..0]
out_port_from_the_seven_seg01[15..0]
out_port_from_the_seven_seg23[15..0]
out_port_from_the_seven_seg45[15..0]
out_port_from_the_seven_seg67[15..0]
in_port_to_the_switch[17..0]
address_to_the_cfi_flash_0[21..0]
data_to_and_from_the_cfi_flash_0[7..0]
read_n_to_the_cfi_flash_0
select_n_to_the_cfi_flash_0

Figure 21. Insert Nios II Processor

Add VCC

Right-click on the block diagram window again, and click **Insert > Symbol**. Expand the **c:/altera/…/libraries** folder, then expand **primitives**, expand **other**, select **vcc**, check the box labeled **Repeat-insert mode**, and click **OK**. Place **vcc** somewhere on the left side of the **NiosII_Processor** block near **reset_n**. Place another **vcc** somewhere on the bottom right side of the block. The initial placements of **vcc** are not important, as they can be moved later. Hit **Esc** on the keyboard after both instances of **vcc** are placed.

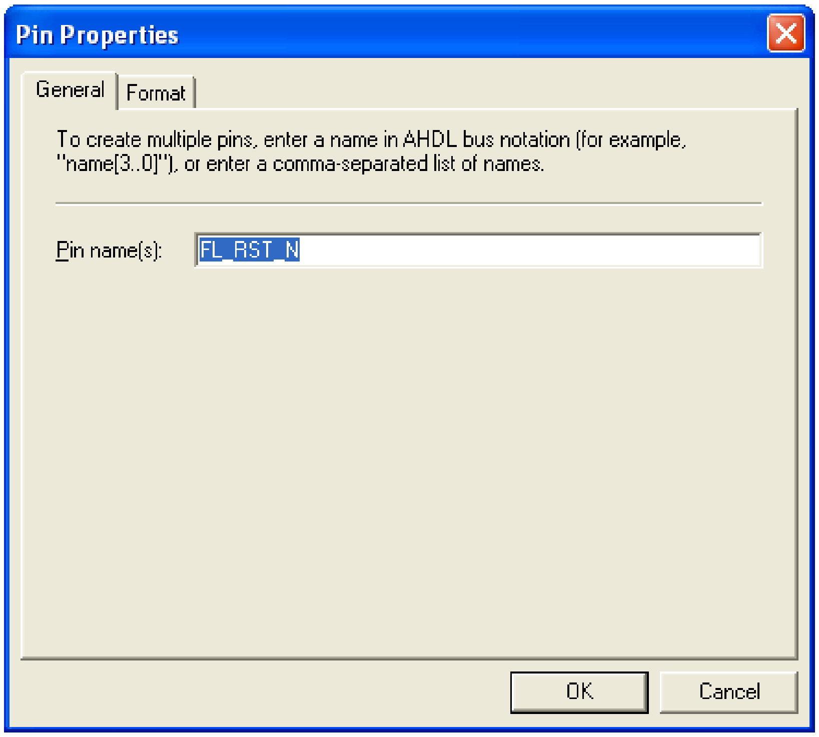
Add Output Pins

Right-click somewhere in the block diagram window again, and click **Insert > Symbol**. Expand the **c:/altera/…/libraries** folder, expand **primitives**, expand **pin**, and select **output**. Check the box called **Repeat-insert mode**, and click **OK**. Place three output pins below the **NiosII_Processor** block. After all three output pins are placed, hit **Esc** on the keyboard.

Move the mouse to the blue line coming from **reset_n** on the block diagram. The mouse cursor should change into a cross-hair shape. When this happens, click and hold to draw a line connecting **reset_n** to the closest **vcc**. Release the mouse when a small box shape appears on **vcc**.

Click and hold **vcc** in order to move it. When **vcc** is moved, the line connected to it follows **vcc**. If the line moves with **vcc**, there is a proper connection. If not, click and drag the mouse to connect the line between **reset_n** and **vcc**. Connect all three of the output pins to the other **vcc** following these same steps.

Double-click one of the output pins to open the **Pin Properties** window. In the **Pin name(s):** section, rename the output pin to **FL_RST_N** and click **OK**. Rename the other pins **LCD_ON** and **LCD_BLON**. Refer to Figure 22.



To create multiple pins, enter a name in AHDL bus notation (for example, "name[3..0]"), or enter a comma-separated list of names.

Pin name(s): FL_RST_N

Figure 22. Rename Output Pins

Right-click the **NiosII_Processor** block, and click **Generate Pins for Symbol Ports**. Input and output pins are automatically generated for the rest of the Nios II system. It is important to verify that the newly generated pins do not cover the **vcc** symbols placed earlier. Move the **vcc** items to a different location if they are covered. Figure 23 shows the completed setup.

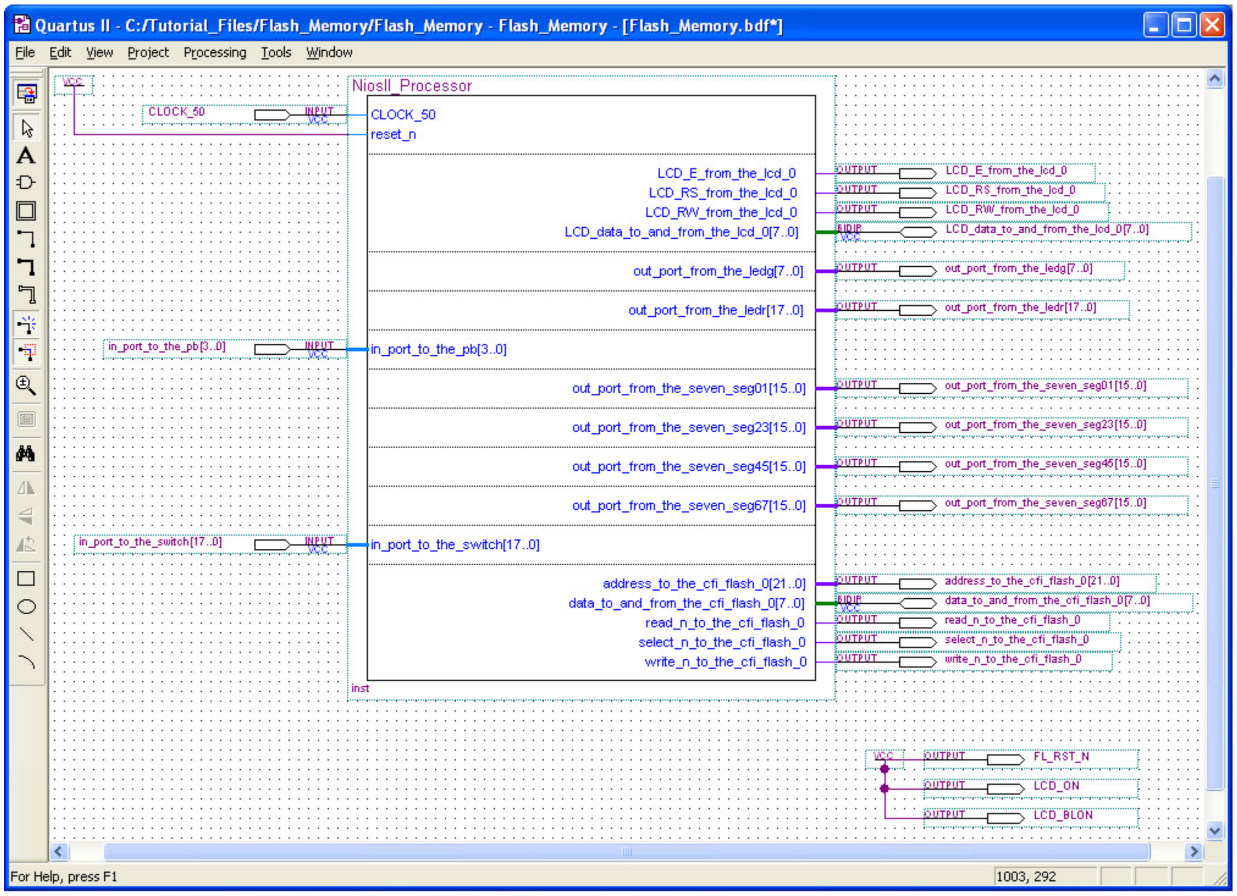

Quartus II - C:/Tutorial_Files/Flash_Memory/Flash_Memory - Flash_Memory - [Flash_Memory.bdf*]
File Edit View Project Processing Tools Window
NiosII_Processor
CLOCK_50
reset_n
LCD_E_from_the_lcd_0
LCD_RS_from_the_lcd_0
LCD_RW_from_the_lcd_0
LCD_data_to_and_from_the_lcd_0[7..0]
out_port_from_the_ledg[7..0]
out_port_from_the_ledr[17..0]
in_port_to_the_pb[3..0]
out_port_from_the_seven_seg01[15..0]
out_port_from_the_seven_seg23[15..0]
out_port_from_the_seven_seg45[15..0]
out_port_from_the_seven_seg67[15..0]
in_port_to_the_switch[17..0]
address_to_the_cfi_flash_0[21..0]
data_to_and_from_the_cfi_flash_0[7..0]
read_n_to_the_cfi_flash_0
select_n_to_the_cfi_flash_0
write_n_to_the_cfi_flash_0
inst
FL_RST_N
LCD_ON
LCD_BLON
For Help, press F1
1003, 292

Figure 23. Complete Quartus II System

Analysis and Elaboration

The next step is Analysis and Elaboration. Under the **Processing** menu, select **Start > Start Analysis & Elaboration**. Save changes if prompted. Upon completion of the process, a message window appears. Ignore any warnings, and click **OK**.

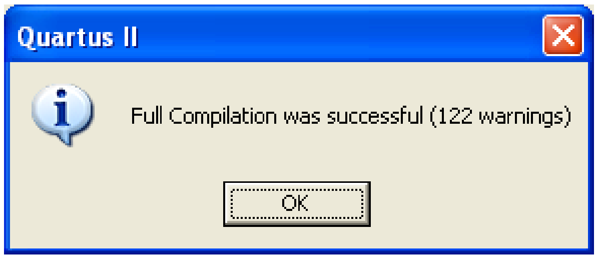
Pin Assignments

Make the pin assignments for the devices. Under the **Assignments** menu, select **Import Assignments…** Click the **…** box to the right of the **File name:** section, and browse for the **Pin_Assignments.csv** file that accompanies this tutorial. Select **Open** and click **OK**.

Verify that the pins are named correctly. Under the **Assignments** menu, select **Pins**. In the **Filter:** menu, choose **Pins: unassigned**. Only 8 pins should appear in the list: pins **[7]** and **[15]** from each of the pin groups **out_port_from_the_seven_seg_***XX* (where *XX* is 01, 23, 45, 67). If there are more than eight pins listed, check the SOPC builder section of this tutorial again to ensure that all of the devices are named exactly as specified in this tutorial. Close the **Pins** window.

Compilation

Compile the design. Under **Processing**, select **Start Compilation.** The success message in Figure 24 appears upon completion. Ignore the warnings that are listed, and click **OK**.

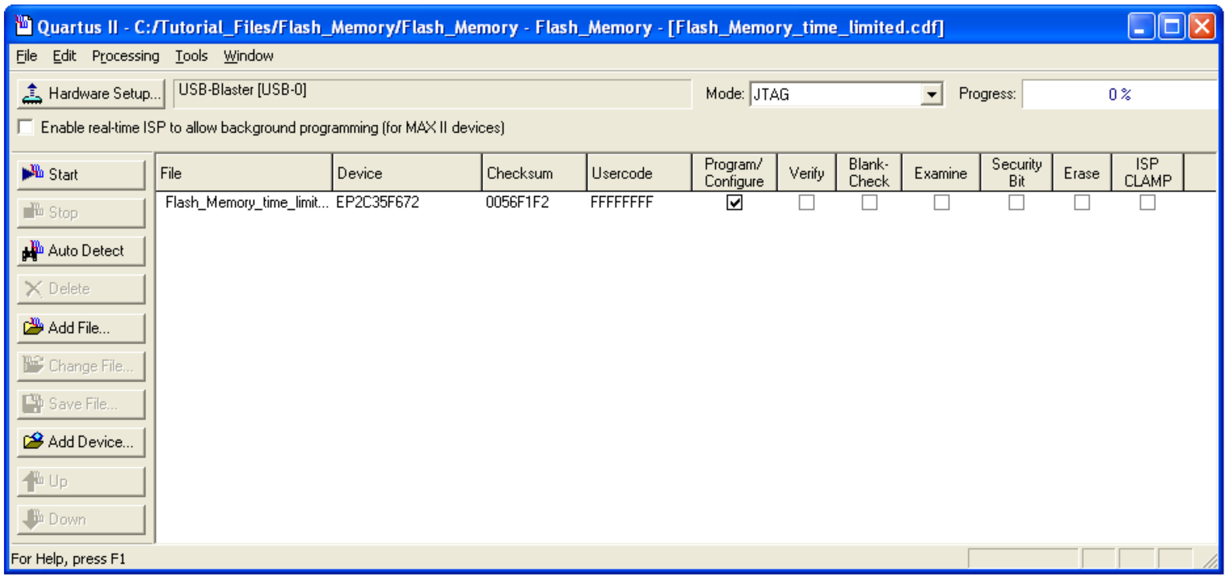


Full Compilation was successful (122 warnings)

Figure 24. Successful Compilation Window

System Programmer

Plug the power supply and USB-Blaster cable into the DE2 board. Hit the red power button to turn on the DE2 board. Under the **Tools** menu, select **Programmer**. If a pop-up window appears, click **OK**. In the **Programmer** window, choose **Hardware Setup…** Verify that **USB-Blaster[USB-0]** is selected, and click **Close**. Verify that the **Mode:** selected is **JTAG**. Set switch SW19 on the DE2 board (next to the LCD) to **Run**. Click **Start** in the programmer window. Refer to Figure 25 for the programmer setup.



Quartus II - C:/Tutorial_Files/Flash_Memory/Flash_Memory - Flash_Memory - [Flash_Memory_time_limited.cdf]

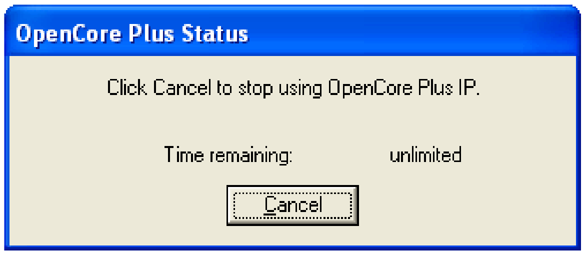
USB-Blaster [USB-0]   Mode: JTAG   Progress: 0 %

Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
Flash_Memory_time_limit...	EP2C35F672	0056F1F2	FFFFFFFF	☑	☐	☐	☐	☐	☐	☐

Figure 25. Programmer Window

The window shown in Figure 26 opens. This window confirms that the FPGA on the DE2 board successfully configured.



Click Cancel to stop using OpenCore Plus IP.

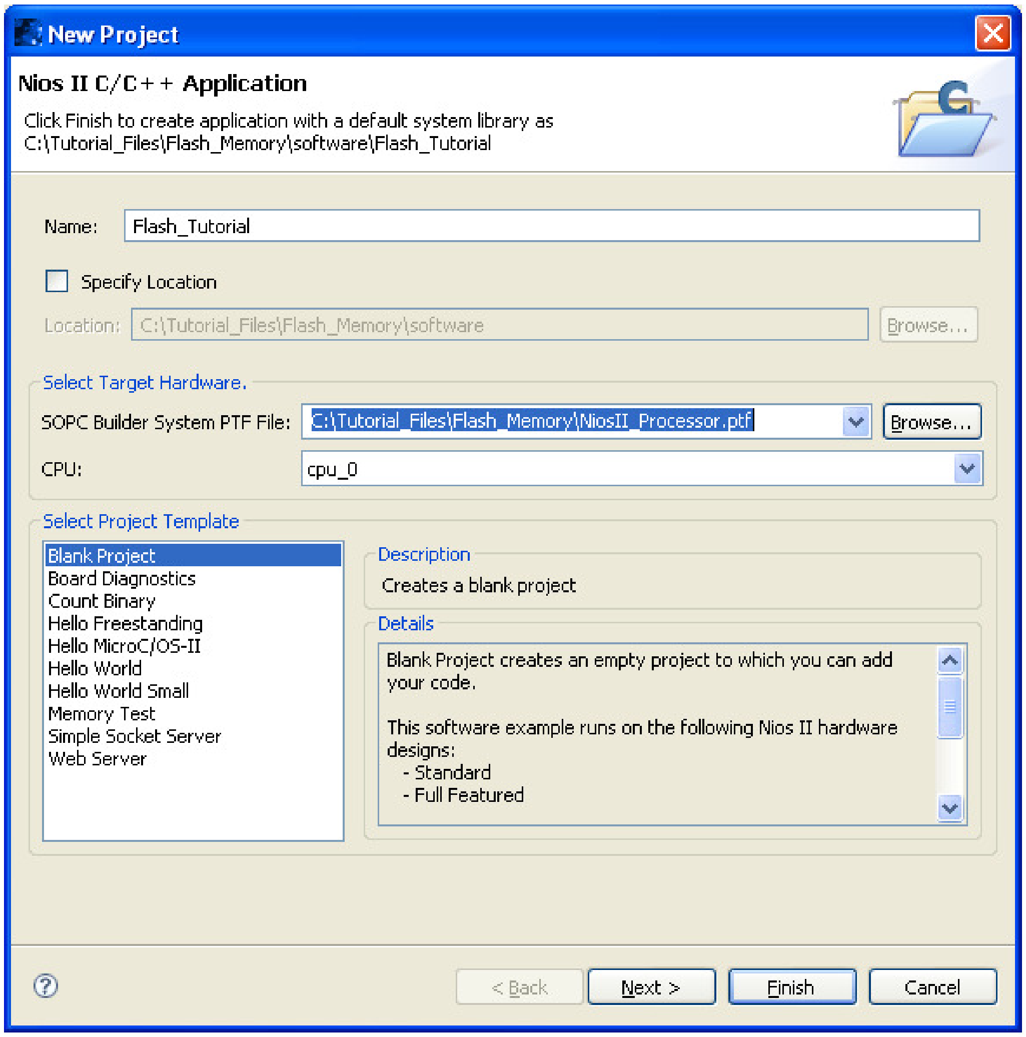
Time remaining: unlimited

Figure 26. Successful Connection Window

Software Design in the Nios II IDE

Creating a New C/C++ Application

Next, implement a software application for the Nios II system. Open the Nios II IDE. Under the **File** menu, select **New > Nios II C/C++ Application** to open the **New Project** window. In the **Name:** section, type **Tutorial_Flash**. In the **SOPC Builder System PTF File,** browse to the location where the **NiosII_Processor** was saved (when created with SOPC Builder), and open it. Under **Select Project Template**, select **Blank Project**. Click **Finish**. Refer to Figure 27.



Nios II C/C++ Application

Click Finish to create application with a default system library as C:\Tutorial_Files\Flash_Memory\software\Flash_Tutorial

Name: Flash_Tutorial

Specify Location

Location: C:\Tutorial_Files\Flash_Memory\software

SOPC Builder System PTF File: C:\Tutorial_Files\Flash_Memory\NiosII_Processor.ptf

CPU: cpu_0

Blank Project
Board Diagnostics
Count Binary
Hello Freestanding
Hello MicroC/OS-II
Hello World
Hello World Small
Memory Test
Simple Socket Server
Web Server

Creates a blank project

Blank Project creates an empty project to which you can add your code.

This software example runs on the following Nios II hardware designs:
- Standard
- Full Featured

Figure 27. New C/C++ Application Window

Importing Source and Header Files

Select the **Tutorial_Flash** folder in the **Nios II C/C++ Projects** pane. In the **File** menu, select **Import…** In the **Import** window, select **File System** as shown in Figure 28, and click **Next**. **Browse** to the folder containing the example files accompanying this tutorial. Check **Flash.c** and **header.h** as shown in Figure 29, and click **Finish**. Flash.c and header.h should now appear in the Tutorial_Flash folder in the Nios II IDE.

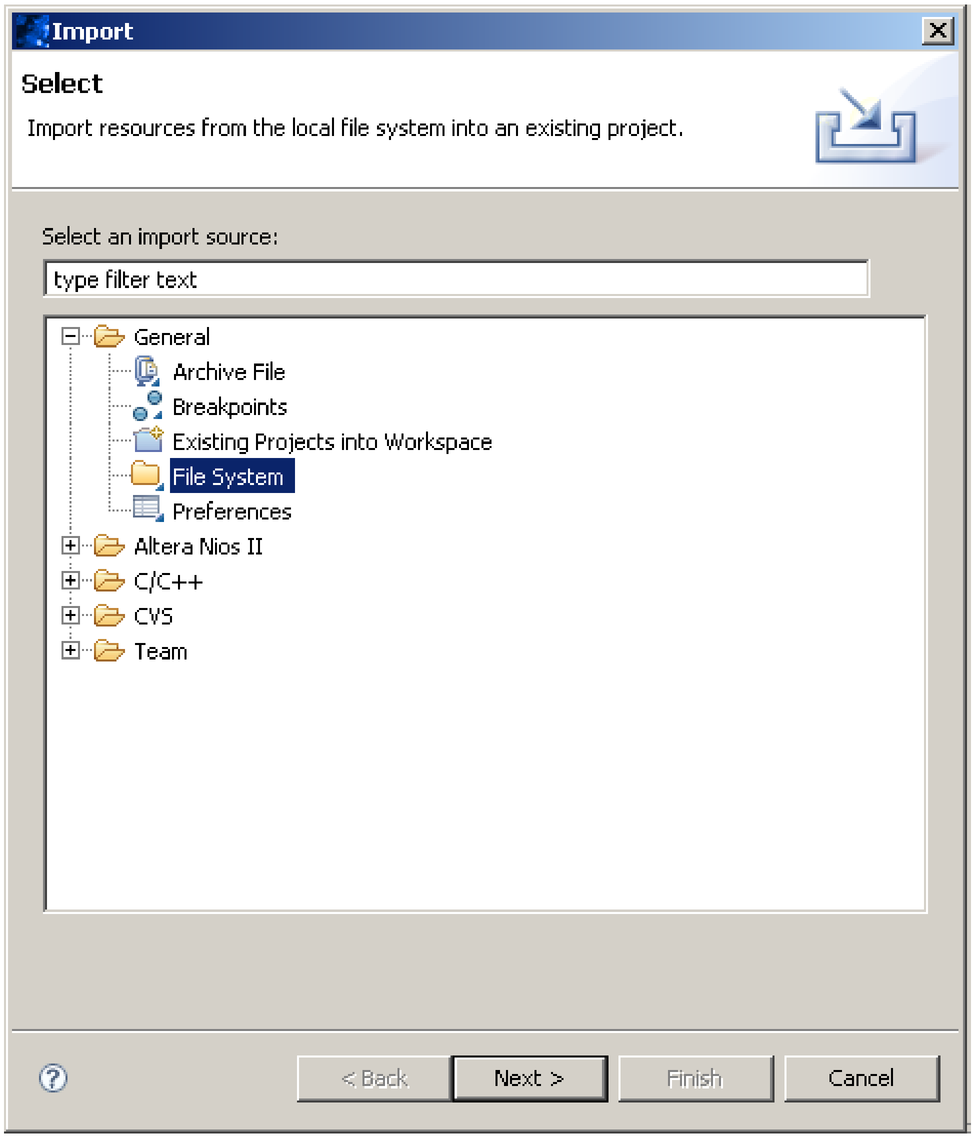
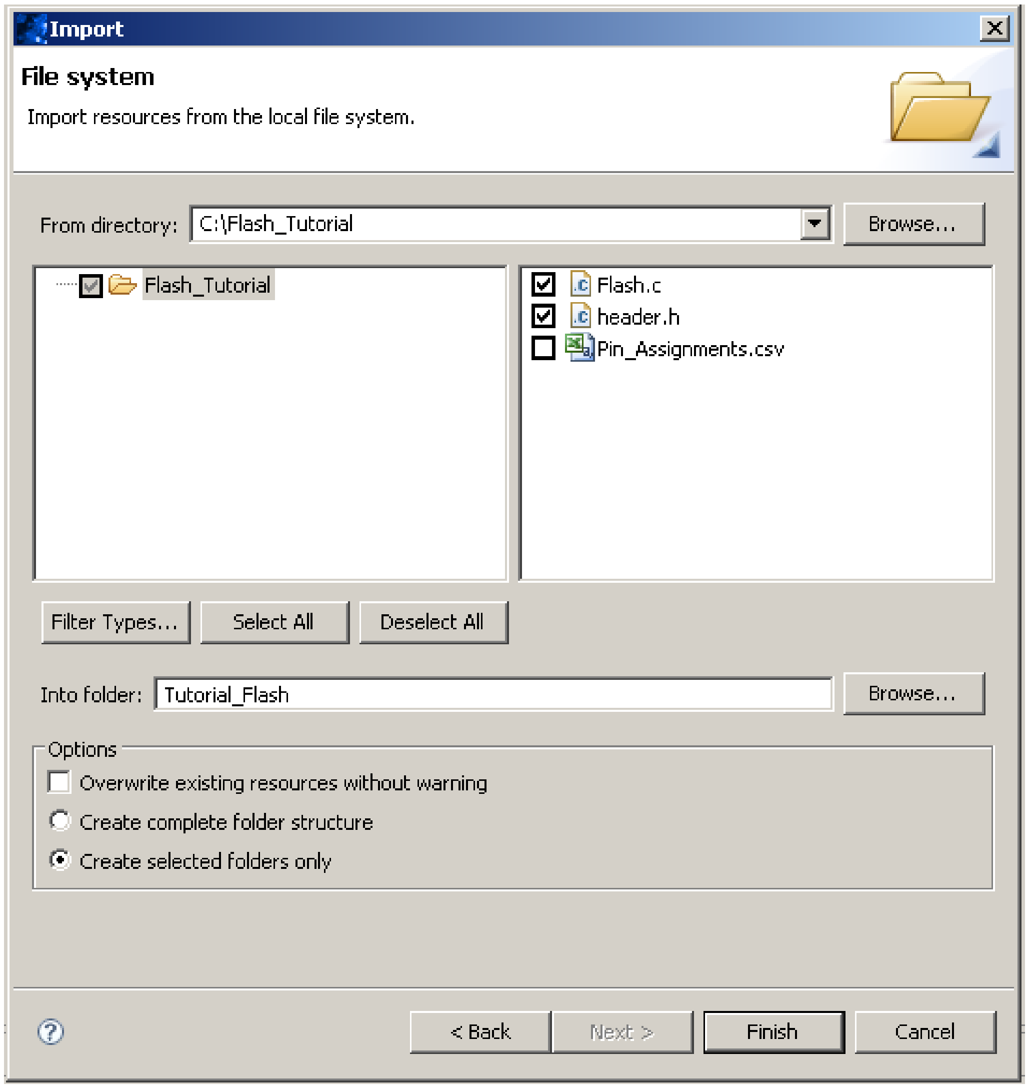


Figure 28. Import File System


Import

File system

Import resources from the local file system.

From directory: C:\Flash_Tutorial

[✓] Flash_Tutorial

[✓] Flash.c
[✓] header.h
[ ] Pin_Assignments.csv

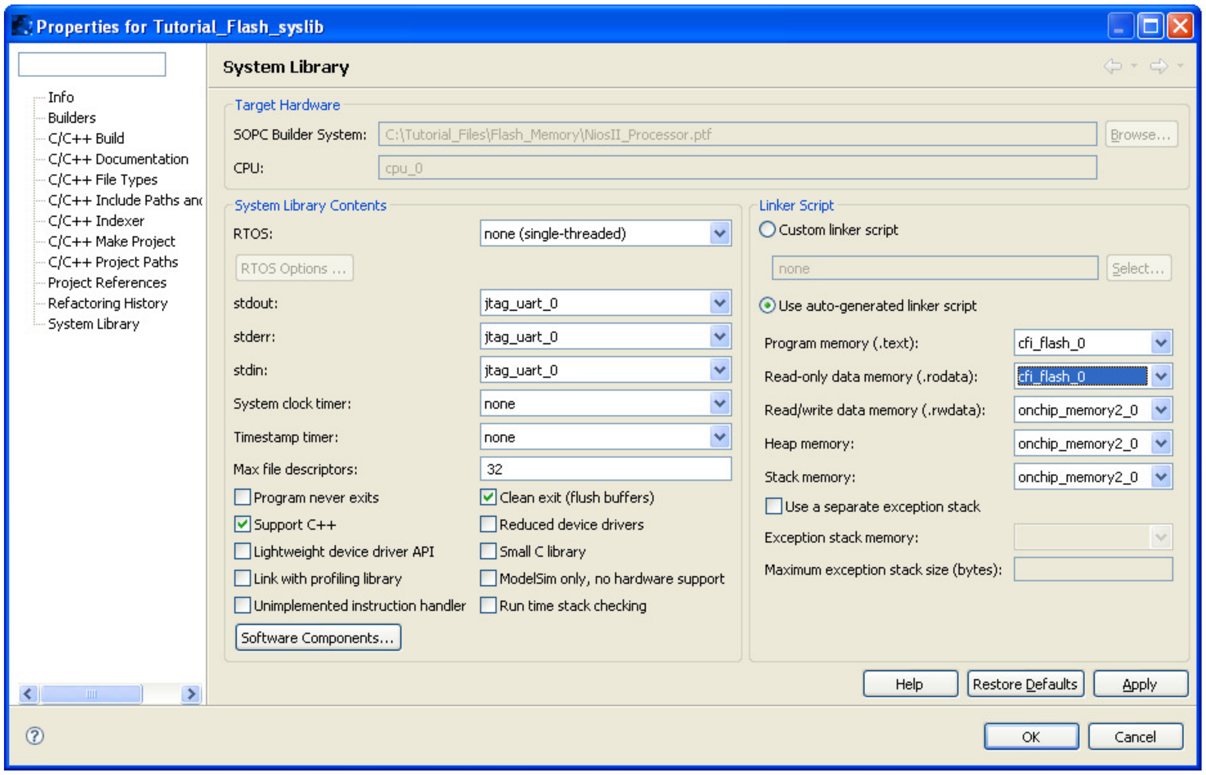
Into folder: Tutorial_Flash

Options
[ ] Overwrite existing resources without warning
○ Create complete folder structure
● Create selected folders only

Figure 29. Import Files

System Library Configuration

Before the program can be run, it is necessary to adjust the system library properties. Right-click on the **Tutorial_Flash** folder, and select **System Library Properties**. A new window opens. Select **cfi_flash_0** in the **Program memory (.text):** drop-down menu. Also select **cfi_flash_0** for the **Read-only data memory (.rodata):** drop-down menu. Do not change any of the other default settings. Click **OK**. Refer to Figure 30.



System Library

Target Hardware

SOPC Builder System: C:\Tutorial_Files\Flash_Memory\NiosII_Processor.ptf

CPU: cpu_0

System Library Contents

RTOS:	none (single-threaded)
stdout:	jtag_uart_0
stderr:	jtag_uart_0
stdin:	jtag_uart_0
System clock timer:	none
Timestamp timer:	none
Max file descriptors:	32

- [ ] Program never exits
- [✓] Clean exit (flush buffers)
- [✓] Support C++
- [ ] Reduced device drivers
- [ ] Lightweight device driver API
- [ ] Small C library
- [ ] Link with profiling library
- [ ] ModelSim only, no hardware support
- [ ] Unimplemented instruction handler
- [ ] Run time stack checking

Linker Script

Custom linker script [✓ Use auto-generated linker script]

Custom linker script: none

Program memory (.text):	cfi_flash_0
Read-only data memory (.rodata):	cfi_flash_0
Read/write data memory (.rwdata):	onchip_memory2_0
Heap memory:	onchip_memory2_0
Stack memory:	onchip_memory2_0

- [ ] Use a separate exception stack

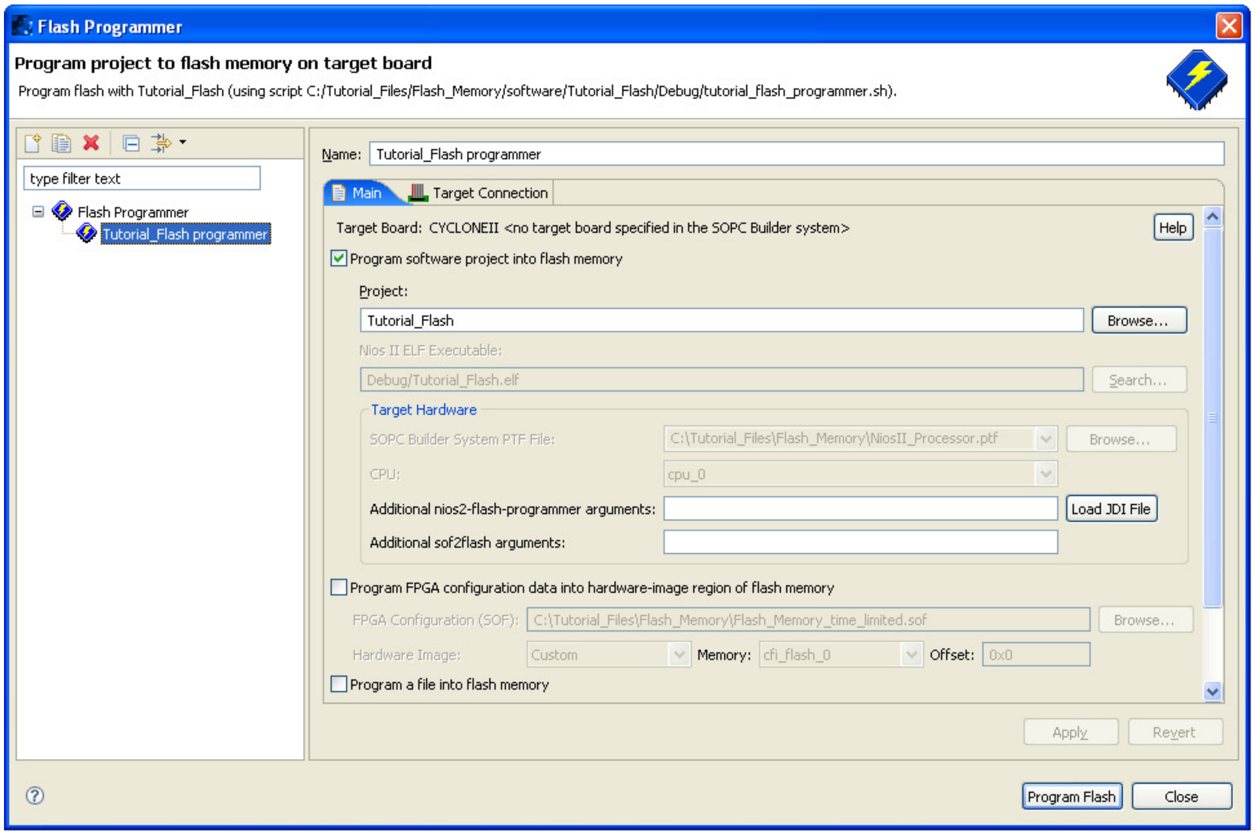
Exception stack memory:

Maximum exception stack size (bytes):

Figure 30. System Library Properties

The Flash Programmer

Next, load the program into flash memory. If any changes are made in the code, save the files before loading them into flash. Store the program in flash memory by selecting **Tools > Flash Programmer…** In the **Flash Programmer** window, select the **Flash Programmer** icon on the upper left side of the window. Click the **New launch configuration** button located above the **Flash Programmer** icon. Now, select the **Program Flash** button in the bottom right corner of the window. If prompted to save changes, click **Yes**. Click **Yes** in the pop-up window that appears. Wait for the Flash Programmer to complete. Refer to Figure 31.


Flash Programmer

Program project to flash memory on target board

Program flash with Tutorial_Flash (using script C:/Tutorial_Files/Flash_Memory/software/Tutorial_Flash/Debug/tutorial_flash_programmer.sh).

type filter text

Flash Programmer
Tutorial_Flash programmer

Name: Tutorial_Flash programmer

Main | Target Connection

Target Board: CYCLONEII <no target board specified in the SOPC Builder system>

[✓] Program software project into flash memory

Project: Tutorial_Flash

Nios II ELF Executable: Debug/Tutorial_Flash.elf

Target Hardware

SOPC Builder System PTF File: C:\Tutorial_Files\Flash_Memory\NiosII_Processor.ptf

CPU: cpu_0

Additional nios2-flash-programmer arguments:

Additional sof2flash arguments:

[ ] Program FPGA configuration data into hardware-image region of flash memory

FPGA Configuration (SOF): C:\Tutorial_Files\Flash_Memory\Flash_Memory_time_limited.sof

Hardware Image: Custom Memory: cfi_flash_0 Offset: 0x0

[ ] Program a file into flash memory

Figure 31. Flash Programmer

Upon successful completion of the Flash Programmer, a message similar to the following displays in the console:

```
Programmed 62KB +2KB in 1.3s (49.2KB/s)
Device contents checksummed OK
Leaving target processor paused
```

Running the Software

Finally, run the program. Right-click on the **Tutorial_Flash** folder, and select **Run As > Nios II Hardware**. Wait for the program to build. Upon successful completion, a message similar to the following text displays in the **Console** pane:

```
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)
```

The DE2 board should be running the program. If the sample code was used, the LCD should display:

You Are AWESOME!
LCD Works ###

The seven-segment displays count in hexadecimal, and the green LEDs count in binary. If a switch is switched to the up position, the red LED above the switch turns on. If the pushbutton KEY0 is pressed, all of the devices count down instead of up. If KEY1 is pressed, the devices resume counting up. Each time KEY2 is pressed, the devices count incrementally faster. Likewise, each time KEY3 is pressed, the devices count incrementally slower.

If any changes are made to the code, save the file, and load the new code into the flash. This is done by selecting **File > Save All**, and then by selecting **Tools > Flash Programmer**. The Flash Programmer is already set up, so simply click the **Program Flash** button at the bottom of the window. Select the **C/C++ Projects** pane, right-click on the **Tutorial_Flash** folder, and select **Run As > Nios II Hardware**. The new code then runs.

Conclusion

This application note explains how to set up a Nios II system that uses the flash memory on Terasic Technologies, Inc.'s DE2 board. The LCD controller, pushbuttons, seven-segment displays, switches, and red and green LEDs on the DE2 board are also used.

Additional Information

Getting Started with Altera's DE2 Board. Altera Corporation. 2008.
DE2 User Manual. Altera Corporation. 2005.
Nios II Processor Reference Handbook. Altera Corporation. 2009.
Nios II Software Developer's Handbook. Altera Corporation. 2009.
Quartus II Handbook, Volume 5: Embedded Peripherals. Altera Corporation. 2009.