Using the SDRAM Memory on Altera's DE2 Board with Verilog Design

This tutorial explains how the SDRAM chip on Altera's DE2 Development and Education board can be used with a Nios II system implemented by using the Altera SOPC Builder. The discussion is based on the assumption that the reader has access to a DE2 board and is familiar with the material in the tutorial *Introduction to the Altera SOPC Builder Using Verilog Design*.

The screen captures in the tutorial were obtained using the Quartus^{\mathbb{R}} II version 5.1; if other versions of the software are used, some of the images may be slightly different.

Contents: Example Nios II System The SDRAM Interface Using the SOPC Builder to Generate the Nios II System Integration of the Nios II System into the Quartus II Project Using a Phase-Locked Loop The introductory tutorial *Introduction to the Altera SOPC Builder Using Verilog Design* explains how the memory in the Cyclone II FPGA chip can be used in the context of a simple Nios II system. For practical applications it is necessary to have a much larger memory. The Altera DE2 board contains an SDRAM chip that can store 8 Mbytes of data. This memory is organized as 1M x 16 bits x 4 banks. The SDRAM chip requires careful timing control. To provide access to the SDRAM chip, the SOPC Builder implements an *SDRAM Controller* circuit. This circuit generates the signals needed to deal with the SDRAM chip.

1 Example Nios II System

As an illustrative example, we will add the SDRAM to the Nios II system described in the *Introduction to the Altera SOPC Builder Using Verilog Design* tutorial. Figure 1 gives the block diagram of our example system.

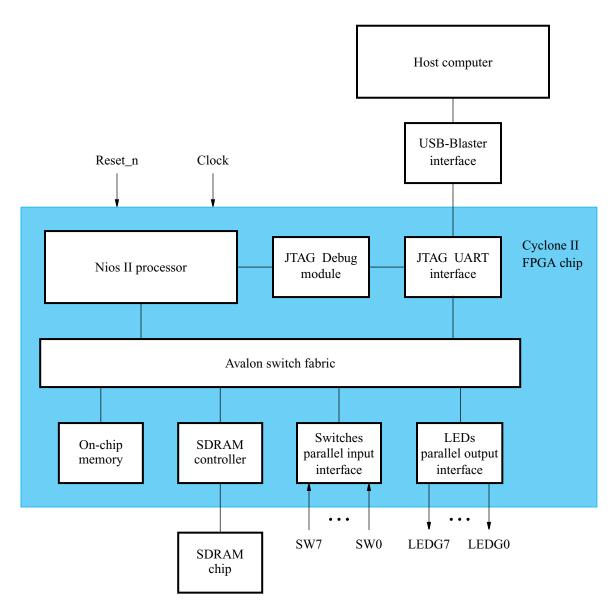


Figure 1. Example Nios II system implemented on the DE2 board.

The system realizes a trivial task. Eight toggle switches on the DE2 board, SW7 - 0, are used to turn on or off the eight green LEDs, LEDG7 - 0. The switches are connected to the Nios II system by means of a parallel I/O

interface configured to act as an input port. The LEDs are driven by the signals from another parallel I/O interface configured to act as an output port. To achieve the desired operation, the eight-bit pattern corresponding to the state of the switches has to be sent to the output port to activate the LEDs. This will be done by having the Nios II processor execute an application program. Continuous operation is required, such that as the switches are toggled the lights change accordingly.

The introductory tutorial showed how we can use the SOPC Builder to design the hardware needed to implement this task, assuming that the application program which reads the state of the toggle switches and sets the green LEDs accordingly is loaded into a memory block in the FPGA chip. In this tutorial, we will explain how the SDRAM chip on the DE2 board can be included in the system in Figure 1, so that our application program can be run from the SDRAM rather than from the on-chip memory.

Doing this tutorial, the reader will learn about:

- Using the SOPC Builder to include an SDRAM interface for a Nios II-based system
- Timing issues with respect to the SDRAM on the DE2 board
- Using a phase-locked loop (PLL) to control the clock timing

2 The SDRAM Interface

The SDRAM chip on the DE2 board has the capacity of 64 Mbits (8 Mbytes). It is organized as $1M \times 16$ bits x 4 banks. The signals needed to communicate with this chip are shown in Figure 2. All of the signals, except the clock, can be provided by the SDRAM Controller that can be generated by using the SOPC Builder. The clock signal is provided separately. It has to meet the clock-skew requirements as explained in section 5. Note that some signals are active low, which is denoted by the suffix N.

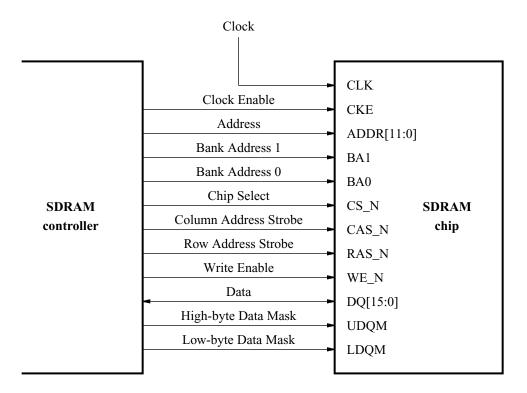


Figure 2. The SDRAM signals.

3 Using the SOPC Builder to Generate the Nios II System

Our starting point will be the Nios II system discussed in the *Introduction to the Altera SOPC Builder Using Verilog Design* tutorial, which we implemented in a project called *lights*. We specified the system shown in Figure 3.

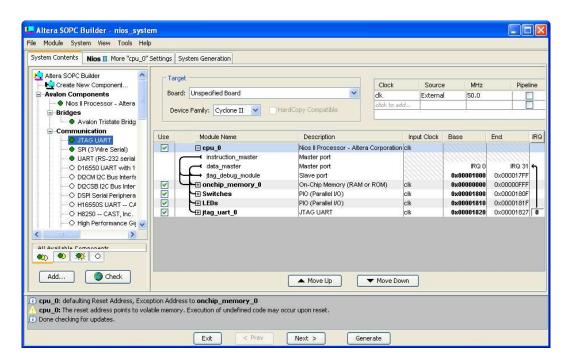


Figure 3. The Nios II system defined in the introductory tutorial.

If you saved the *lights* project, then open this project in the Quartus II software and then open the SOPC Builder. Otherwise, you need to create and implement the project, as explained in the introductory tutorial, to obtain the system shown in the figure.

To add the SDRAM, in the window of Figure 3 select Avalon Components > Memory > SDRAM Controller and click Add. A window depicted in Figure 4 appears. Set the Data Width parameter to 16 bits and leave the default values for the rest. Since we will not simulate the system in this tutorial, do not select the option Include a functional memory model in the system testbench. Click Finish. Now, in the window of Figure 3, there will be an **sdram_0** module added to the design. Since there is only one SDRAM on the DE2 board, change the name of this module to simply **sdram**. Then, the expanded system is defined as indicated in Figure 5. Observe that the SOPC Builder assigned the base address 0x00800000 to the SDRAM. Leave the addresses of all modules as assigned in the figure and regenerate the system.

🗳 SDRAM Controller - sdram_0 🛛 🛛 🔀
Presets: (Custom)
Memory Profile Timing
Data Width Architecture 16 V Bits Chip Selects: 1 V Banks: 4 V
Address Widths Row 12 Column 8
Share Pins via Tristate Bridge
Generic Memory Model (Simulation Only) Include a functional memory model in the system testbench.
Memory size: 8 MBytes 4194304 x 16 64 MBits
Cancel < Prev Next > Finish

Figure 4. Add the SDRAM Controller.

Extra Utilities Interfaces and Periphe Legacy Components Math Coprocessors Memory		et d: Unspecified Board rice Family: Cyclone II	HardCopy Compatible	Clock clk click to a	Souri Extern		
● Cypress CY7C13i ● EPCS Serial Flash	Use	Module Name	Description	Input Clock	Base	End	IR
Flash Memory (Cc		⊡ cpu_0	💓 Nios II Processor - Alter	CONTRACTOR AND			
On-Chip Memory (2	<pre>instruction_master data_master</pre>	Master port Master port		IRQ	0 IRQ	31 4
AMD 29LV800 Fla	1	→ jtag_debug_module	Slave port		0x0000100	T 0.7217.527.12	0.01
- O DDR SDRAM Cont			On-Chip Memory (RAM or R.		0x0000000	이 - 영화가 아이가 다 다	1000
O DDR2 SDRAM Cor		-⊞ Switches	PIO (Parallel I/O)	clk "	0x0000180	C 0.0210.02701	
O IDT71 V016 SRAM		-⊞ LEDs	PIO (Parallel I/O) JTAG UART	clk clk	0x0000181 0x0000182	이 물건가 아파 가지?	22(2)
		fitag_uart_0 Fi sdram	SDRAM Controller	cik	0x0000182	C 0.0210.000.000.000	1000
Add Check Cpu_0 was generated as plain-text HC cpu_0: The reset address points to vo Done checking for updates.		nory, Execution of undefined co		1ove Down	<u> </u>		_

Figure 5. The expanded Nios II system.

The augmented Verilog module generated by the SOPC Builder is in the file *nios_system.v* in the directory of the project. Figure 6 depicts the portion of the code that defines the input and output signals for the module *nios_system*. As in our initial system that we developed in the introductory tutorial, the 8-bit vector that is the input to the parallel port *Switches* is called *in_port_to_the_Switches*. The 8-bit output vector is called *out_port_from_the_LEDs*. The clock and reset signals are called *clk* and *reset_n*, respectively. A new module, called *sdram*, is included. It involves the signals indicated in Figure 2. For example, the address lines are referred to as the **output** vector *zs_addr_from_the_sdram[11:0]*. The data lines are referred to as the **inout** vector *zs_dq_to_and_from_the_sdram[15:0]*. This is a vector of the **inout** type because the data lines are bidirectional.

nios_	_system.v	
3056		
3057 r	nodule nios system (
3058	// 1) global signals:	
3059	clk,	
3060	reset_n,	
3061		
3062	// the_LEDs	
3063	out_port_from_the_LEDs,	
3064		
3065	// the_Switches	
3066	in_port_to_the_Switches,	
3067	and a subscription of the second second	
3068	// the_sdram	
3069	<pre>zs_addr_from_the_sdram,</pre>	
	zs_ba_from_the_sdram,	
3071	<pre>zs_cas_n_from_the_sdram,</pre>	
3072	zs_cke_from_the_sdram,	
3073	zs_cs_n_from_the_sdram,	
3074	zs_dq_to_and_from_the_sdram,	
3075	zs_dqm_from_the_sdram,	
3076	<pre>zs_ras_n_from_the_sdram,</pre>	
3077	zs_we_n_from_the_sdram	
3078)	
3080	5	
3081	output [7: 0] out port from the LEDs;	
3082	output [11: 0] zs addr from the sdram;	
3083	output [1: 0] zs ba from the sdram;	
3084	output zs cas n from the sdram;	
3085	output zs cke from the sdram;	
3086	output zs cs n from the sdram;	
3087	inout [15: 0] zs dq to and from the sdram;	
	output [1: 0] zs dom from the sdram;	
3089	output zs ras n from the sdram;	
	output zs we n from the sdram;	
3091	input clk;	
ALC: N	529658 225	2

Figure 6. A part of the generated Verilog module.

4 Integration of the Nios II System into the Quartus II Project

Now, we have to instantiate the expanded Nios II system in the top-level Verilog module, as we have done in the tutorial *Introduction to the Altera SOPC Builder Using Verilog Design*. The module is named *lights*, because this is the name of the top-level design entity in our Quartus II project.

A first attempt at creating the new module is presented in Figure 7. The input and output ports of the module use the pin names for the 50-MHz clock, *CLOCK_50*, pushbutton switches, *KEY*, toggle switches, *SW*, and green LEDs, *LEDG*, as used in our original design. They also use the pin names *DRAM_CLK*, *DRAM_CKE*, *DRAM_ADDR*, *DRAM_BA_1*, *DRAM_BA_0*, *DRAM_CS_N*, *DRAM_CAS_N*, *DRAM_RAS_N*, *DRAM_WE_N*, *DRAM_DQ*, *DRAM_UDQM*, and *DRAM_LDQM*, which correspond to the SDRAM signals indicated in Figure 2. All of these names are those specified in the DE2 User Manual, which allows us to make the pin assignments by importing them from the file called *DE2_pin_assignments.csv* in the directory *DE2_tutorials\design_files*, which is included on the CD-ROM that accompanies the DE2 board and can also be found on Altera's DE2 web pages.

Observe that the two *Bank Address* signals are treated by the SOPC Builder as a two-bit vector called *zs_ba_from_the_sdram[1:0]*, as seen in Figure 6. However, in the *DE2_pin_assignments.csv* file these signals are given as scalars *DRAM_BA_1* and *DRAM_BA_0*. Therefore, in our Verilog module, we concatenated these signals as {*DRAM_BA_1*, *DRAM_BA_0*}. Similarly, the vector *zs_dqm_from_the_sdram[1:0]* corresponds to {*DRAM_UDQM, DRAM_LDQM*}.

Finally, note that we tried an obvious approach of using the 50-MHz system clock, *CLOCK_50*, as the clock signal, *DRAM_CLK*, for the SDRAM chip. This is specified by the **assign** statement in the code. This approach leads to a potential timing problem caused by the clock skew on the DE2 board, which can be fixed as explained in section 5.

// Implements the augmented Nios II system for the DE2 board.

- // Inputs: SW7–0 are parallel port inputs to the Nios II system.
- // CLOCK_50 is the system clock.
- // KEY0 is the active-low system reset.
- // Outputs: LEDG7-0 are parallel port outputs from the Nios II system.
- // SDRAM ports correspond to the signals in Figure 2; their names are those
- // used in the DE2 User Manual.

module lights (SW, KEY, CLOCK_50, LEDG, DRAM_CLK, DRAM_CKE,

DRAM_ADDR, DRAM_BA_1, DRAM_BA_0, DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N, DRAM_WE_N, DRAM_DQ, DRAM_UDQM, DRAM_LDQM); input [7:0] SW; input [0:0] KEY; input CLOCK_50; output [7:0] LEDG; output [11:0] DRAM_ADDR; output DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK; output DRAM_CKE, DRAM_CS_N, DRAM_WE_N, DRAM_UDQM, DRAM_LDQM; inout [15:0] DRAM_DQ;

// Instantiate the Nios II system module generated by the SOPC Builder

nios_system NiosII (CLOCK_50, KEY[0], LEDG, SW, DRAM_ADDR, {DRAM_BA_1, DRAM_BA_0}, DRAM_CAS_N, DRAM_CKE, DRAM_CKE, DRAM_CS_N, DRAM_DQ, {DRAM_UDQM, DRAM_LDQM}, DRAM_RAS_N, DRAM_WE_N); assign DRAM_CLK = CLOCK_50;

endmodule

Figure 7. A first attempt at instantiating the expanded Nios II system.

As an experiment, you can enter the code in Figure 7 into a file called *lights.v.* Add this file and all the *.v files produced by the SOPC Builder to your Quartus II project. Compile the code and download the design into the Cyclone II FPGA on the DE2 board. Use the application program from the tutorial *Introduction to the Altera SOPC Builder Using Verilog Design*, which is shown in Figure 8.

.include	"nios_m	acros.s"
.equ .equ		s, 0x00001800 x00001810
.global _start:	_start	
loop:	movia movia ldbio stbio br	r2, Switches r3, LEDs r4, 0(r2) r4, 0(r3) loop

Figure 8. Assembly language code to control the lights.

Use the Altera Debug Client, which is described in the tutorial *Altera Debug Client*, to assemble, download, and run this application program. If successful, the lights on the DE2 board will respond to the operation of the toggle switches.

Due to the clock skew problem mentioned above, the Nios II processor may be unable to properly access the SDRAM chip. A possible indication of this may be given by the Altera Debug Client, which may display the message depicted in Figure 9. To solve the problem, it is necessary to modify the design as indicated in the next section.

_ ×

Info & Errors

```
Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Resetting and pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloading 00000000 ( 0%)
Downloaded 2KB in 0.0s
Verifying 00000000 ( 0%)
Verify failed between address 0x0 and 0x66F
Leaving target processor paused
```

Figure 9. Error message in the Altera Debug Client that may be due to the SDRAM clock skew problem.

5 Using a Phase-Locked Loop

The clock skew depends on physical characteristics of the DE2 board. For proper operation of the SDRAM chip, it is necessary that its clock signal, *DRAM_CLK*, leads the Nios II system clock, *CLOCK_50*, by 3 nanoseconds. This can be accomplished by using a *phase-locked loop (PLL)* circuit. There exists a Quartus II Megafunction, called *ALTPLL*, which can be used to generate the desired circuit. The circuit can be created, by using the Quartus II MegaWizard Plug-In Manager, as follows:

1. Select Tools > MegaWizard Plug-In Manager. This leads to the window in Figure 10. Choose the action Create a new custom megafunction variation and click Next.



Figure 10. The MegaWizard.

 In the window in Figure 11, specify that Cyclone II is the device family used and that the circuit should be defined in Verilog HDL. Also, specify that the generated output (Verilog) file should be called *sdram_pll.v*. From the list of megafunctions in the left box select I/O > ALTPLL. Click Next.

MegaWizard Plug-In Manag	er [page 2a] 🛛 🔀
MegaWizard Plug-In Manag Which megafunction would you like to customize' Select a megafunction from the list below Select a megafunction from the list below Altras SOPC Builder Altra Sopc Builder Builde	
ALTGXB	box (Assignments menu).
	Your current user library directories are:
ALTPLL_RECONFIG	
IO/MAX II oscillator	
Parallel Flash Loader	
🚽 🚽 SignalTap II Logic Analyzer 🛛 💌	
	Cancel < Back Next > Finish

Figure 11. Select the megafunction and name the output file.

3. In Figure 12, specify that the frequency of the *inclock0* input is 50 MHz. Leave the other parameters as given by default. Click Next to reach the window in Figure 13.

MegaWizard Plug-In Manager - ALTP	PLL [page 3 of 9]	
	Able to implement the requested PLL	Jump to page for: General/Modes 💌
	General	
	Which device family will you be using?	Cyclone II
sdram_pll	Which device speed grade will you be using?	Any
outuri_pir	What is the frequency of the inclock0 input?	50 MHz 💌
inclk0 inclk0 frequency: 50.000 MHz c0, areset Operation Mode: Normal locked	■ Set up PLL in LVDS mode	Data rate: Not Available 💌 Mbps
Cik Ratio Ph (dg) DC (%) o0 1/1 0.00 50.00 Cyclone II	PLL type Which PLL type will you be using? C Fast PLL C Enhanced PLL C Select the PLL type automatically	
	Operation mode	
	How will the PLL outputs be generated?	
	 Use the feedback path inside the PLL 	
	 In Normal Mode In Source-Synchronous Compensation M 	
	C In Zero Delay Buffer Mode	
	C With no compensation	
	C Create an 'Ibin' input for an external feedback	(External Feedback Mode)
	Which output clock will be compensated for?	c0 💌
	Documentation(Cancel < Back Next > Finish

Figure 12. Define the clock frequency.

MegaWizard Plug-In Manager - /	ALTPLL [page 4 of 9]
	Able to implement the requested PLL Jump to page for: Scan/Lock
sdram_pil inclk0 inclk0 frequency: 50.000 MHz Operation Mode: Normal Clk Ratio Ph (dg) DC (%) e0 1/1 0.00 50.00 Cyclone II	Dynamic configuration Create optional inputs for dynamic reconfiguration Optional inputs Create an 'pllena' input to selectively enable the PLL Create an 'areset' input to asynchronously reset the PLL Create an 'pldena' input to selectively enable the phase/freq. detector Lock output Create 'locked' output Hold 'locked' output low 1048575
	Advanced PLL 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 Documentation Cancel < Back Next > Finish

Figure 13. Remove unnecessary signals.

4. We are interested only in the input signal *inclock0* and the output signal *c0*. Remove the other two signals shown in the block diagram in the figure by de-selecting the optional input **areset** as well as the **locked** output, as indicated in the figure. Click Next on this page as well as on page 5, until you reach page 6 which is shown in Figure 14.

MegaWizard Plug-In Manager	- ALTPLL [page 6 of 9]		×
	c0 - Core/External Output Clock Able to implement the requested PLL I✓ Use this clock	Jump to page for: 🛛 🖬 Clock c0	•
sdram_pll	Clock Tap Settings	Requested settings Actual settin 50 MHz 💌 50.000000	gs
Inclk0 inclk0 frequency: 50.000 MHz 0 Operation Mode: Normal Clk Ratio Ph (dg) D C (%) 00 1/1 -54.00 50.00 S0.00	C Enter output clock parameters: Clock multiplication factor Clock division factor	1	
Cyclone II	Clock phase shift	-3 * ns V -3.00	
	Clock duty cycle (%) More Details >>	50.00 ÷	
		Quick Navigation	
	Documentation	Cancel < Back Next > F	Finish

Figure 14. Specify the phase shift.

- 5. The shifted clock signal is called c0. Specify that the output clock frequency is 50 MHz. Also, specify that a phase shift of -3 ns is required, as indicated in the figure. Click Finish, which advances to page 9.
- 6. In the summary window in Figure 15 click Finish to complete the process.

MegaWizard Plug-In Manager	- ALT	PLL [page 9 of 9	9] Summary	×
Sdram_pil inclk0 Operation Mode: Normal Clk Ratio Ph (dg) DC (%) c0 1/1 -54.00 50.00 Cyclone I	<u>=0</u> ,	create the checked files in the exclude a file by checking or ur respectively. The state of chec for the next MegaWizard Plug-I	n Manager session. ager will create these files in the directory: Description Variation file AHDL Include file VHDL Component declaration file Quartus symbol file Instantiation template file Verilog Black Box' declaration file	
		Documentation	Cancel < Back Next> Finish	1

Figure 15. The summary page.

The desired PLL circuit is now defined as a Verilog module in the file *sdram_pll.v*, which is placed in the project directory. Add this file to the *lights* project. Figure 16 shows the module ports, consisting of signals *inclk0* and *c0*.

3.6 mc	dule sdram pll (
37	inclk0,	~
	c0);	
39		
40	input inclk0;	
41	output cO;	
42		
43	wire [5:0] sub_wire0;	
44	wire [0:0] sub wire4 = 1'h0;	
45	wire [0:0] sub wire1 = sub wire0[0:0];	
46	wire cO = sub wire1;	
47	wire sub wire2 = inclk0;	
48	<pre>wire [1:0] sub wire3 = (sub wire4, sub wire2);</pre>	
49		~

Figure 16. The generated PLL module.

Next, we have to fix the top-level Verilog module, given in Figure 7, to include the PLL circuit. The desired code is shown in Figure 17. The PLL circuit connects the shifted clock output *c0* to the pin *DRAM_CLK*.

// Implements the augmented Nios II system for the DE2 board.

- // Inputs: SW7–0 are parallel port inputs to the Nios II system.
- // CLOCK_50 is the system clock.
- // KEY0 is the active-low system reset.
- // Outputs: LEDG7–0 are parallel port outputs from the Nios II system.
- // SDRAM ports correspond to the signals in Figure 2; their names are those
- // used in the DE2 User Manual.

module lights (SW, KEY, CLOCK_50, LEDG, DRAM_CLK, DRAM_CKE,

DRAM_ADDR, DRAM_BA_1, DRAM_BA_0, DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N, DRAM_WE_N, DRAM_DQ, DRAM_UDQM, DRAM_LDQM); input [7:0] SW; input [0:0] KEY; input CLOCK_50; output [7:0] LEDG; output [11:0] DRAM_ADDR; output DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK; output DRAM_CKE, DRAM_CS_N, DRAM_WE_N, DRAM_UDQM, DRAM_LDQM; inout [15:0] DRAM_DQ;

// Instantiate the Nios II system module generated by the SOPC Builder

nios_system NiosII (CLOCK_50, KEY[0], LEDG, SW, DRAM_ADDR, {DRAM_ADDR, {DRAM_BA_1, DRAM_BA_0}, DRAM_CAS_N, DRAM_CAS_N, DRAM_CKE, DRAM_CKE, DRAM_CS_N, DRAM_DQ, {DRAM_UDQM, DRAM_LDQM}, DRAM_RAS_N, DRAM_WE_N);

// Instantiate the module sdram_pll (inclk0, c0)
 sdram_pll neg_3ns (CLOCK_50, DRAM_CLK);

endmodule

Figure 17. Proper instantiation of the expanded Nios II system.

Compile the code and download the design into the Cyclone II FPGA on the DE2 board. Use the application program in Figure 8 to test the circuit.

Copyright ©2006 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

This document is being provided on an "as-is" basis and as an accommodation and therefore all warranties, representations or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed.