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 7.1; 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.

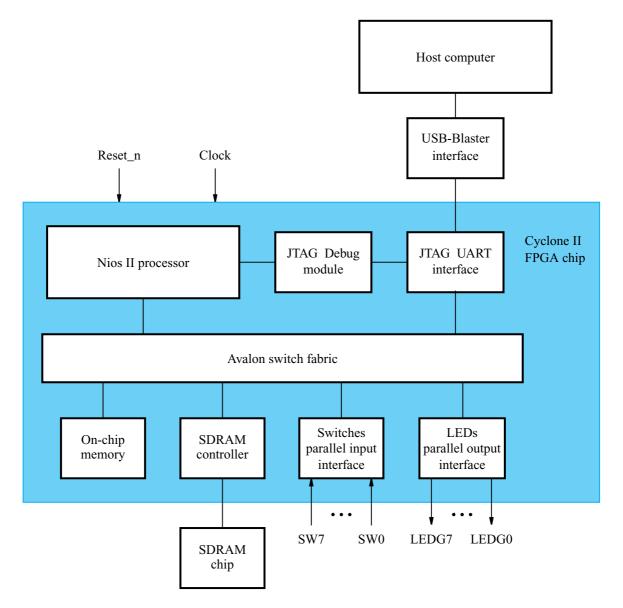


Figure 1. Example Nios II system implemented on the DE2 board.

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.

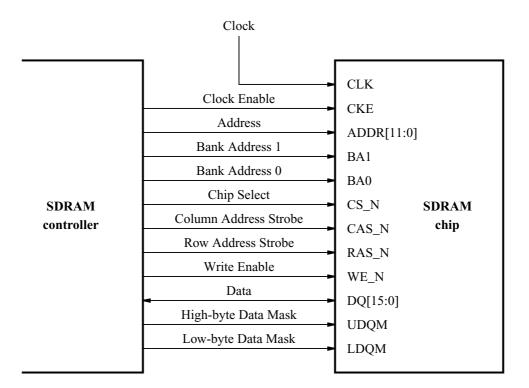


Figure 2. The SDRAM signals.

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.

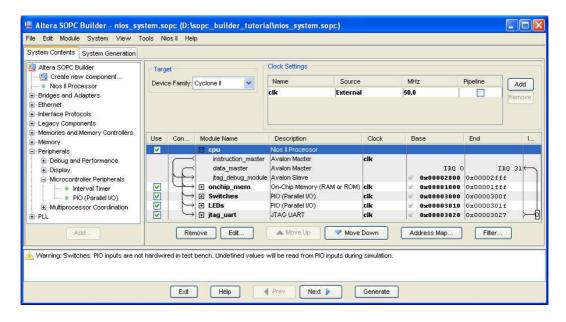
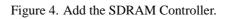


Figure 3. The Nios II system defined in the introductory tutorial.

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 Memories and Memory Controllers > SDRAM > SDRAM Controller and click Add. A window depicted in Figure 4 appears. Select *Custom* from the Presets drop-down list. 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** module added to the design. Select the command System > Auto-Assign Base Addresses to produce the assignment shown in Figure 5. Observe that the SOPC Builder assigned the base address 0x00800000 to the SDRAM. To make use of the SDRAM, we need to configure the reset vector and exception vector of the Nios II processor. Right-click on the cpu and then select Edit to reach the window in Figure 6. Select sdram to be the memory device for both reset vector and exception vector, as shown in the figure. Click Finish to return to the System Contents tab and regenerate the system.

SDRAM Controller - sdram	
SDRAM Controller Version 7.1	Documentation
1 Parameter Settings	
Memory Profile > Timing >	
Presets: Custom	
Data width Bits: 16	
Architecture Chip select: 1 V Banks: 4 V	
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)	
Memory size = 8 MBytes 4194304 × 16 64 MBits	
Cancel	Next > Finish



ystem Contents System Generation									
Altera SOPC Builder	Targ	et		Clock Settings					
	Devic	e Family:	Cyclone II 🛛 👻	Name	Source		MHz	Pipeline	Add
E-Bridges and Adapters	1			clk	Externa	I.	50.0		Remove
Ethernet									Nombye
-Interface Protocols									
Legacy Components				4					-
Memories and Memory Controller	Use	Con	Module Name	Description		Clock	Base	End	SI.
⊞ Flash			🖃 cpu	Nios II Processor					
⊡ On-Chip		\sim	instruction_master	Avalon Master		cik			
SDRAM		\parallel	data_master	Avalon Master			IRQ	0 3	ERQ 31←
SDRAM Controller		\rightarrow	jtag_debug_module				0x0100280	0 0x01002f	ff
		$ \rightarrow $	Image: Book of the second s	On-Chip Memory (R	AM or ROM)	122203	i 0x0100100		
Memory		$ \rightarrow $		PIO (Parallel I/O)		cik	= 0x0100300		2,770.0
Peripherals		\rightarrow	⊞ LEDs	PIO (Parallel I/O)		cik	0x0100 301	and a set of a second second second second	and a second second
🗄 Debug and Performance 🤍	V		⊞ jtag_uart	JTAG UART		cik	0.000.000.000.0	0 0x010030	
Dianlau 🛁			🗄 sdram	SDRAM Controller		cik	0x0080000	0 0x00ffff	11
			nove Edit	Move Up	V Move		(ALL ALL ALL ALL ALL ALL ALL ALL ALL AL		
Add		Ren	nove Edit	A Move op	Move	Down	Address Map	Filter.	
	1					777 7778			
VVarning: Switches: PIO inputs are not	hardwir	ed in test l	bench. Undefined values	will be read from PIO	inputs during	simulation.			

Figure 5. The expanded Nios II system.

Parameter Settings Caches and Memory Interfaces Advanced Features JTAG Debug Module Custom Instructions Core Nilos II Select a Nios II core: Image: Select a Nios II core: Image: Select a Nios II core: Image: Select a Nios II core: Selector Guide RISC RISC RISC RISC Select a Nios II core: Selector Guide RISC Select a Nios II core: Imstruction Cache Instruction Cache Family: Cyclone II Hardware Multiply Hardware Multiply Hardware Multiply 1/system: S0.0 MHz Up to 5 DMIPS Up to 25 DMIPS Up to 51 DMIPS Logic Usage S00-700 LEs 1200-1400 LEs 1400-1800 LEs Memory Usage Hardware Multiply: Imstruction Cache Two M4Ks + cache Three M4Ks + cache Hardware Multiply: Imstruction Cache Dout Cache Dout Cache Barrel Shifter Data Cache Dynamic Branch Prediction Hardware Multiply: Imstruction Les 1400-1800 LEs Memory Usage Hardware Multiply: Imstruction Cache Two M4Ks + cache Three M4Ks + cache Hardware Multiply: Imstruction Cache Dout Cache		s II Processor			Documentatio
Core Nios II Select a Nios II core: Image: Select a Nios II Image: Select a Nios II/S Nios II/S Select a Nios II Select a Nios II/S Risc Select of Guide Sabit Sabit Sabit Select of Guide Sabit Sabit Sabit Select of Guide Branch Prediction Branch Prediction Family: Cyclone II Hardware Multiply Hardware Multiply fsystem: 50.0 MHz Up to 5 DMIPS Up to 25 DMIPS Up to 51 DMIPS cpuid: 0 Up to 5 DMIPS Up to 51 DMIPS Up to 51 DMIPS cogic Usage 600-700 LEs 1200-1400 LEs 1400-1800 LEs Memory Usage Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache Hardware Multiply: Imarch Multiply: Hardware Divide Divide					
Select a Hios II core: Image: Select a Hios II core: Image: Select Coulde RISC RISC Selector Guide Sabit Sabit Sabit Selector Guide Family: Cyclone II Instruction Cache Branch Prediction If system: S0.0 MHz Emarch Prediction Barrel Shifter cpuid: 0 Up to 5 DMIPS Up to 5 DMIPS Up to 51 DMIPS Logic Usage 600-700 LEs 1200-1400 LEs 1400-1800 LEs Hardware Divide Hardware Multiply: Image: Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache Hardware Multiply: Image: Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache		ches and Memory Interfaces 🔪	> Advanced Features	> JTAG Debug Module >	> Custom Instructions >
Image: Nios II/e Nios II/s Nios II/f Nios II Risc 32-bit Risc Selector Guide 32-bit 32-bit 32-bit Family: Cyclone II Tastruction Cache Branch Prediction Hardware Multiply Hardware Multiply Hardware Multiply Hardware Divide Barrel Shifter pate: 0 0 Up to 5 DMIPS Up to 52 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	Core Nios II				
Nios II Selector Guide Family: Cyclone II RISC 32-bit RISC 32-bit RISC 32-bit family: Cyclone II Instruction Cache Branch Prediction Hardware Multiply Instruction Cache Branch Prediction Hardware Divide Instruction Cache Branch Prediction Hardware Divide fsystem: 50.0 MHz cpuid: 0 Up to 5 DMIPS Up to 25 DMIPS Up to 25 DMIPS Logic Usage 600-700 LEs 1200-1400 LEs 1400-1800 LEs Memory Usage Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache	Select a Nios II core				
NIOS II 32-bit 32-bit 32-bit 32-bit Selector Guide Family: Cyclone II Instruction Cache Branch Prediction Hardware Multiply Hardware Multiply Hardware Divide epuid: 0 0 Dynamic Branch Prediction Performance at 50.0 MHz Up to 5 DMIPS Up to 52 DMIPS Up to 51 DMIPS Logic Usage 600-700 LEs 1200-1400 LEs 1400-1800 LEs Hardware Multiply: Image: Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache Hardware Multiply: Image: Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache Reset Vector: Memory: sdram Offset: 0x0 0x00800000		Nios II/e	ONios II/s	○Nios II/f	
Performance at 50.0 MHz Up to 5 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: Image: Comparison of the second se	Selector Guide Family: Cyclone II f _{system:} 50.0 MHz		32-bit Instruction Cache Branch Prediction Hardware Multiply	32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache	rediction
Memory Usage Two M4Ks (or equiv.) Two M4Ks + cache Three M4Ks + cache Hardware Muttiply: Hardware Divide Hardware Divide Memory: sdram Offset: 0x0 0x00800000 Offset: 0x0 0x00800000 Memory: sdram Offset: 0x0 0x00800000 0x00800000 Offset: 0x0 0x00800000 0x00800000 Other context (context (context	Performance at 50.0 M	IHz Up to 5 DMIPS	Up to 25 DMIPS	all and the second second second	
Hardware Multiply:					
Reset Vector: Memory: sdram Offset: 0x0 0x00800000	Memory Usage	Two M4Ks (or equiv.)	I wo M4Ks + cache	Three M4Ks + cache	
Exception Vector: Memory: sdram		emory: sdram			0x00800000
	Exception Vector: Me	mory: sdram	Viffset:	Dx20	0×00800020

Figure 6. Define the reset vector and the exception vector.

The augmented Verilog module generated by the SOPC Builder is in the file *nios_system.v* in the directory of the project. Figure 7 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.

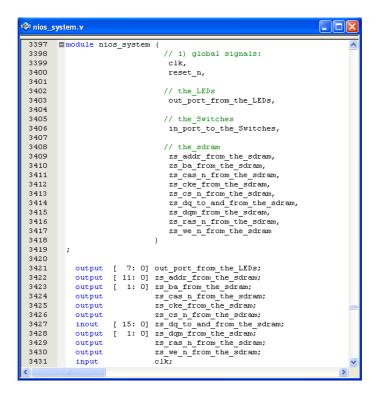


Figure 7. A part of the generated Verilog module.

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 8. 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 7. 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 [7:0] 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

endmodule

Figure 8. A first attempt at instantiating the expanded Nios II system.

As an experiment, you can enter the code in Figure 8 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 9. Notice in our expanded system, the addresses assigned by the SOPC Builder are 0x01003000 for Switches and 0x01003010 for LEDs, which are different from the original system. These changes are already reflected in the program in Figure 9.

.include	"nios_m	acros.s"
.equ .equ		s, 0x01003000 x01003010
.global _start:	_start	
loop:	movia movia ldbio stbio	r2, Switches r3, LEDs r4, 0(r2) r4, 0(r3)
	br	loop

Figure 9. 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 10. 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)	
Х	
Downloading 00800000 (0%)	
Downloaded 1KB in 0.0s	
Verifying 00800000 (0%)	
Verify failed between address 0x800000 and 0x80001B	
Leaving target processor paused	8
	8
Possible causes for the SREC verification failure:	
1. Not enough memory in your Nios II system to contain the SREC file.	
2. The locations in your SREC file do not correspond to a memory device.	
3. You may need a properly configured PLL to access the SDRAM or Flash memory.	

Figure 10. 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 11. Choose the action Create a new custom megafunction variation and click Next.



Figure 11. The MegaWizard.

In the window in Figure 12, 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 Manager [page 2a]	
Which megafunction would you like to customize? Select a megafunction from the list below	Which device family will you be Cyclone II
ALT2GX8 ALTASM_PARALLEL ALTASM_PARALLEL ALTASM_PARALLEL ALTCLKCTRL ALTCLKCTRL ALTDDIO_BIDIR ALTDDIO_BIDIR ALTDDIO_UN ALTDDIO_UN ALTDQS ALTOQS ALTG8UF ALTIOBUF ALTUVDS ALTUVDS	What name do you want for the output file? Browse D:\DE2_sdram_tutorial\sdram_pil.v
ALTOCT ALTOCT ALTOLT ALTOLT ALTPLL, RECONFIG ALTREMOTE_UPDATE MAX II oscillator MAX II oscillator	Cancel < Back Next > Finish

Figure 12. Select the megafunction and name the output file.

3. In Figure 13, 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 14.

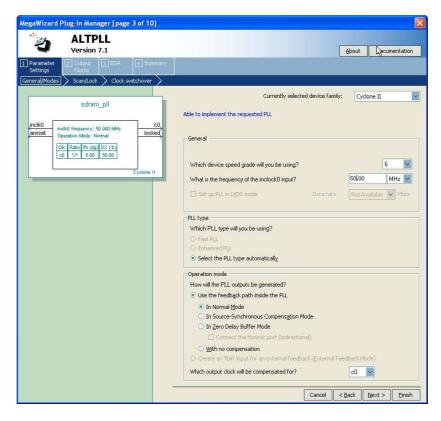


Figure 13. Define the clock frequency.

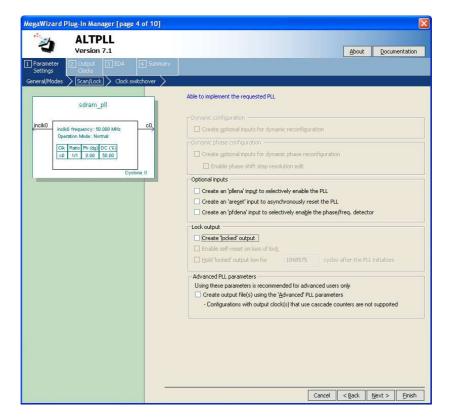


Figure 14. Remove unnecessary signals.

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 15.

MegaWizard Plug-In N	Manager [page 6 of 10]				X
~	TPLL			a <u></u>	
	ion 7.1	20		About	Documentation
Parameter 2 Output Settings Clocks	ut 3 EDA 4 Summary s				
ckc0 ckc1 > ckc	2 >				
sdra	and the second se	c0 - Core/External Output Clock Able to implement the requested PLL			
inclk0 inclk0 frequency		☑ Use this clock			
Operation Mode:	: Normal	Clock Tap Settings	Requested setting	15	Actual settings
Cik Ratio Ph (c0 1/1 -54.	(dg) DC (%) .00 50.00	Enter output clock frequency:		MHz 🔽	50.000000
	Cyclone II	O Enter output clock parameters:			
		Clock multiplication factor		<< Copy	1
		Clock division factor	1	<< copy	1
		Clock phase shift	-3 🖍 r	IS 🔽	-3.00
		Clock duty cycle (%)	50.00		50.00
		More Details >>			
			Per	Clock Feasibility	/ Indicators
			c) c1 c2	
	_				
			Cancel	< <u>B</u> ack	<u>N</u> ext > Einish

Figure 15. 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 10.
- 6. In the summary window in Figure 16 click Finish to complete the process.

ALTPLL Version 7.1		
arameter Clocks EDA Summary		<u>About</u> <u>D</u> ocumentation
sdram_pil ncik0 frequency: 50 000 MHz Operation Mode: Normal [Cik: Ratio [Ph (dg) DC (3)]	automatically generated, and a rec Finish to generate the selected file subsequent MegaWizard Plug-In M	ate. A gray checkmark indicates a file that is I checkmark indicates an optional file. Click s. The state of each checkbox is maintained i anager sessions. creates the selected files in the following
c0 1/1 -54.00 50.00	File	Description
Cyclone II	G stam_pllv G stam_plippi stam_plicmp G stam_plicmp G stam_pli_list g stam_pl_bt g stam_pl_btv G stam_pl_waveforms.hml └_stam_pl_wave*.jpg	Variation file PinPlanner pots PPF file. AHDL Londponent declaration file UHDL component declaration file Quartus II symbol file Instantiation template file Verlog HDL black-box file Sample waveforms in summary Sample waveform file(s)

Figure 16. 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 17 shows the module ports, consisting of signals *inclk0* and *c0*.

🕸 sdram	_pll.v	
39	module sdram pll (~
40	inclkO,	_
41	c0);	
42		
43	input inclk0;	
44	output c0;	
45		
46	wire [5:0] sub_wire0;	
47	wire [0:0] sub wire4 = 1'h0;	
48	<pre>wire [0:0] sub_wire1 = sub_wire0[0:0];</pre>	
49	wire cO = sub wire1;	
50	<pre>wire sub_wire2 = inclk0;</pre>	
51	<pre>wire [1:0] sub wire3 = {sub wire4, sub wire2};</pre>	
52		
53	altpl1 altpl1_component (
54	.inclk (sub_wire3),	*
<		>

Figure 17. The generated PLL module.

Next, we have to fix the top-level Verilog module, given in Figure 8, to include the PLL circuit. The desired code is shown in Figure 18. 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 [7:0] 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_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 18. 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 9 to test the circuit.

Copyright ©2007 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.