

Using the SDRAM on Altera's DE2-115 Board with VHDL Designs

#### 1 Introduction

This tutorial explains how the SDRAM chips on Altera's DE2-115 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 DE2-115 board and is familiar with the material in the tutorial *Introduction to the Altera SOPC Builder Using VHDL Designs*.

The screen captures in the tutorial were obtained using the Quartus<sup>®</sup> II version 9.1; 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 the Clock Signals IP Core

## 2 Background

The introductory tutorial *Introduction to the Altera SOPC Builder Using VHDL Designs* explains how the memory in the Cyclone IV 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 DE2-115 board contains 2 SDRAM chips that can each store 64 Mbytes of data. Each chip is organized as 8M x 16 bits x 4 banks. The SDRAM chips require careful timing control. To provide access to the SDRAM chips, the SOPC Builder implements an *SDRAM Controller* circuit. This circuit generates the signals needed to deal with the SDRAM chips.

# 3 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 VHDL Designs* tutorial. Figure 1 gives the block diagram of our example system.

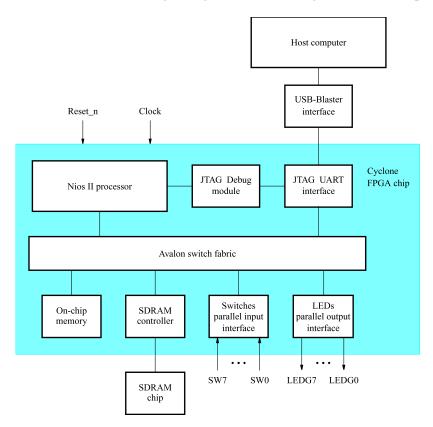


Figure 1. Example Nios II system implemented on the DE2-115 board.

The system realizes a trivial task. Eight toggle switches on the DE2-115 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 SDRAM chips on the DE2-115 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 DE2-115 board

## 4 The SDRAM Interface

The two SDRAM chips on the DE2-115 board each have a capacity of 512 Mbits (64 Mbytes). Each chip is organized as 8M x 16 bits x 4 banks. The signals needed to communicate with a 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 7. Note that some signals are active low, which is denoted by the suffix N.

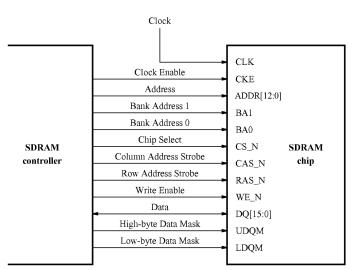


Figure 2. The SDRAM signals.

# 5 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 VHDL Designs* tutorial, which we implemented in a project called *lights*. We specified the system shown in Figure 3.

Lie Edit Module System View Ic		ial\nios_system.sopc)				
System Contents System Generation						
Component Library	Target	Clock Settings				
Nios Il Processor	Device Family: Cyclone II	Name	Source		MHz	Add
Bridges and Adapters     Interface Protocols     B-Ethernet		clk_0	External		50.0	Remove
⊕-PCI ⊟-Serial						
-      Avalon-ST JTAG I	Use Con Module Name	Description	Clock	Base	End	Tags
<ul> <li>Avalon-ST Serial I</li> <li>JTAG UART</li> <li>SPI (3 Wire Serial)</li> <li>UART (RS-232 Se</li> </ul>	cpu_0     instruction_master     tag debug modul	Avalon Memory Mapped Master	cik_0	IRQ 0 ■ 0×00002800		<u>^</u>
-Legacy Components     -Memories and Memory Control     -DMA		2_0 On-Chip Memory (RAM or ROM) Avalon Memory Mapped Slave PIO (Parallel I/O)	clk_0	=° 0x00001000	0x00001fff	
i Flash	✓         ⊡         Switches           ✓         ≲1           ✓         □         LEDs	Avalon Memory Mapped Slave PIO (Parallel I/O)	clk_0	<u>≓</u> 0x00003000	0x0000300f	
	S1 I jtag_uart_0	Avalon Memory Mapped Slave JTAG UART	clk_0	≓ 0x00003010	0x0000301f	
٩	avaion_itag_slave	Avalon Memory Mapped Slave	clk_0	<b>₽ 0x00003020</b>	0x00003027	×
New Edit Add	Remove Edit	Address Map	Eiters Fitter:	Default		
A Warning: Switches: PIO inputs are not	t hardwired in test bench. Undefined value:	s will be read from PIO inputs during simu	ation.			
	Exit	Help Prev Next	Generat	e -		

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 32 bits, the Row Width to 13 bits, the Column Width to 10 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. Select the *Timing* tab to get to the window in Figure 5. Configure the SDRAM timing parameters by setting the refresh command rate to once every 7.8125 microseconds and the delay after powerup to 200 microseconds. Click Finish. Now, in the window of Figure 3, there will be an **sdram** entity added to the design. Select the command System > Auto-Assign Base Addresses to produce the assignment shown in Figure 6. Observe that the SOPC Builder assigned the base address 0x08000000 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\_0 and then select Edit to reach the window in Figure 7. Select sdram\_0 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_0
SDRAM Controller
Parameter Settings
Memory Profile > Timing >
Presets: Custom
Data width
Bits: 32
Architecture
Chip select: 1 V Banks: 4 V
Address widths
Row: 13 Column: 10
Share pins via tristate bridge
Controller shares dq/dqm/addr I/O pins
Tristate bridge selection:
Generic memory model (simulation only)
Include a functional memory model in the system testbench
Memory size = 128 MBytes 33554432 × 32 1024 MBits
Cancel < Back Next > Finish

Figure 4. Add the SDRAM Controller.

SDRAM Controller - sdram_0		
SDRAM Cont	roller	About Documentation
Parameter Settings		
Memory Profile Timing		
SDRAM timing parameters		
CAS latency cycles: 01	02	⊙ 3
Initialization refresh cycles:	2	
Issue one refresh command every:	7.8125	us
Delay after powerup, before initializa	ation: 200	us
Duration of refresh command (t_rfc)	70	ns
Duration of precharge command (t_r		ns
ACTIVE to READ or WRITE delay (t_r	cd): 20	ns
Access time (t_ac):	5.5	ns
Write recovery time (t_wr, no auto p	recharge): 14	ns
	[court	< Back Next > Finish
	Cancel	S DACK INEXC > FINISH

Figure 5. SDRAM Timings

mponent Library	Target	Clo	ck Settings				
Avalon-ST Multi-C		Cyclone IV E 🗸 N	ame	Source	MHz		
Avalon-ST Round     Avalon-ST Single     On-Chip FIFO Mer     On-Chip Memory I	Device Failing	cik		ternal	50.0		Remove
SDRAM     DDR SDRAM Conf	Use Conn	Module Name	Description	Clock	Base	End Te	ags
- ODR SDRAM High		E cou 0	Nios Il Processor				
ODR2 SDRAM Col     ODR2 SDRAM Hig     ODR3 SDRAM Hig		✓ instruction_master ✓ data_master → itag debug module	Avaion Memory Mapped Master Avaion Memory Mapped Master Avaion Memory Mapped Slave	cik_0	IRQ (	) IRQ 31 0x10002fff	
SDRAM Controller SRAM		⇒ onchip_memory2_0 ⇒ s1 ⇒ Switches	On-Chip Memory (RAM or ROM) Avalon Memory Mapped Slave PIO (Parallel I/O)	cik_0	₽ 0×10001000	0x10001fff	
Debug and Performance     Display		→ s1 □ LEDs	Avalon Memory Mapped Slave PIO (Parallel I/O)	clk_0		0x1000300f	
FPGA Peripherals     Microcontroller Peripheral:		→ s1 ⊟ jtag_uart_0	Avalon Memory Mapped Slave JTAG UART	cik_0		0x1000301f	
Interval Timer     PIO (Parallel I/O)		→ avalon_itag_slave	Avalon Memory Mapped Slave	clk_0	= 0x10003020	0x10003027	
Multiprocessor Coordinati     PLL		⊟ sdram_0 → s1	SDRAM Controller Avalon Memory Mapped Slave	cik_0	<b>₽</b> 0x0800000	0x0fffffff	
Processor Additions	<						
Vew Edit Add	Remove	Edit	Address Map	Filters Fil	ter: Default		
Warning: <b>cpu_0</b> : The address range o nfo: <b>Switches</b> : PIO inputs are not ha					oss 28-bit boundaries is	not supported by GCC a	nd will result

Figure 6. The expanded Nios II system.

MogaCore'					About Documentation
Parameter Settings					
	nes and Memory Interfaces	Advanced Features	MMU and MPU Settings	JTAG Debug Mo	dule 🔪 Custom Instruction
Core Nios II		<u> </u>			
Select a Nios II core:					
	Nios II/e	○Nios II/s	ONios Ⅱ/f		
Nios II Selector Guide Family: Cyclone IV E f <sub>system</sub> : 50.0 MHz cpuid: 0	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dvnamic Branch Pi	rediction	
Performance at 50.0 MH	z Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS		
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs		
Memory Usage	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache	•	
Hardware Multiply:	nory: sdram_0	Hardware Divide		0×08000000	
Exception Vector: Mem	ory: sdram_0	Offset: 0x20		0x08000020	
Include MMU		m that explicitly supports an MN	IU Offset: <sub>Ox0</sub>		
Fast TLB Miss Exception					

Figure 7. Define the reset vector and the exception vector.

The augmented VHDL entity generated by the SOPC Builder is in the file *nios\_system.vhd* in the directory of the project. Figure 8 depicts the portion of the code that defines the port signals for the entity *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\_0* and *reset\_n*, respectively. A new entity, called *sdram*, is included. It involves the signals indicated in Figure 2. For example, the address lines are referred to as the OUT vector *zs\_addr\_from\_the\_sdram\_0[12:0].* The data lines are referred to as the INOUT vector zs\_dq\_to\_and\_from\_the\_sdram\_0[31:0]. This is a vector of the INOUT type because the data lines are bidirectional.

```
4009
      entity nios system is
4010
      port (
                      -- 1) global signals:
4011
                         signal clk 0 : IN STD LOGIC;
4012
4013
                         signal reset_n : IN STD_LOGIC;
4014
4015
                      -- the LEDs
                         signal out_port_from_the_LEDs : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
4016
4017
                      -- the Switches
4018
4019
                         signal in_port_to_the_Switches : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
4020
4021
                      -- the sdram O
4022
                         signal zs addr from the sdram 0 : OUT STD LOGIC VECTOR (12 DOWNTO 0);
4023
                         signal zs_ba_from_the_sdram_0 : OUT STD_LOGIC_VECTOR (1 DOWNTO 0);
                         signal zs_cas_n_from_the_sdram_0 : OUT STD_LOGIC;
4024
                         signal zs_cke_from_the_sdram_0 : OUT STD_LOGIC;
402.5
4026
                         signal zs_cs_n_from_the_sdram_0 : OUT STD_LOGIC;
                         signal zs_dq_to_and_from_the_sdram_0 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
4027
                         signal zs_dqm_from_the_sdram_0 : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
4028
                         signal zs ras n from the sdram 0 : OUT STD LOGIC;
4029
                         signal zs_we_n_from_the_sdram_0 : OUT STD_LOGIC
4030
4031
                      );
        end entity nios system;
4032
```

Figure 8. A part of the generated VHDL entity.

# 6 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 VHDL entity, as we have done in the tutorial *Introduction to the Altera SOPC Builder Using VHDL Designs*. The entity 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 entity is presented in Figure 9. The input and output ports of the entity 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*, *DRAM\_CS\_N*, *DRAM\_CAS\_N*, *DRAM\_RAS\_N*, *DRAM\_WE\_N*, *DRAM\_DQ*, and *DRAM\_DQM*, which correspond to the SDRAM signals indicated in Figure 2. All of these names are those specified in the DE2-115 User Manual, which allows us to make the pin assignments by importing them from the file called *DE2\_115\_pin\_assignments.qsf* in the directory *tutorials\design\_files*, which is included on the CD-ROM that accompanies the DE2-115 board and can also be found on Altera's DE2-115 web page.

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 chips. This is specified by the last assignment statement in the code. This approach leads to a potential timing problem caused by the clock skew on the DE2-115 board, which can be fixed as explained in section 7.

- -- 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 DE2-115 User Manual.
- LIBRARY ieee;
- USE ieee.std\_logic\_1164.all;
- USE ieee.std\_logic\_arith.all;
- USE ieee.std\_logic\_unsigned.all;
- ENTITY lights IS
  - PORT (SW: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);
  - KEY : IN STD\_LOGIC\_VECTOR(0 DOWNTO 0);
  - CLOCK\_50 : IN STD\_LOGIC;
  - LEDG : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);
  - DRAM\_DQ : INOUT STD\_LOGIC\_VECTOR (31 DOWNTO 0);
  - DRAM\_ADDR : OUT STD\_LOGIC\_VECTOR (12 DOWNTO 0);
  - DRAM\_BA : OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);
  - DRAM\_CAS\_N, DRAM\_RAS\_N, DRAM\_CLK : OUT STD\_LOGIC;
  - DRAM\_CKE, DRAM\_CS\_N, DRAM\_WE\_N : OUT STD\_LOGIC;
  - DRAM\_DQM : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0));
- END lights;
- ARCHITECTURE Structure OF lights IS
  - COMPONENT nios\_system
    - PORT ( clk\_0 : IN STD\_LOGIC;
      - reset\_n : IN STD\_LOGIC;
        - out\_port\_from\_the\_LEDs : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); in\_port\_to\_the\_Switches : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); zs\_addr\_from\_the\_sdram\_0 : OUT STD\_LOGIC\_VECTOR(12 DOWNTO 0); zs\_ba\_from\_the\_sdram\_0 : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0); zs\_cas\_n\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_cke\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_cs\_n\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_dq\_to\_and\_from\_the\_sdram\_0 : INOUT STD\_LOGIC\_VECTOR(31 DOWNTO 0);
- ... continued in Part  $\boldsymbol{b}$

```
zs_dqm_from_the_sdram_0 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
          zs_we_n_from_the_sdram_0 : OUT STD_LOGIC);
  END COMPONENT;
BEGIN
-- Instantiate the Nios II system entity generated by the SOPC Builder.
  NiosII: nios_system
     PORT MAP (
          clk_0 \Rightarrow CLOCK_50,
          reset n \Rightarrow KEY(0),
          out_port_from_the_LEDs => LEDG,
          in port to the Switches => SW,
          zs_addr_from_the_sdram_0 => DRAM_ADDR,
          zs ba from the sdram 0 \Rightarrow DRAM BA,
          zs cas n from the sdram 0 \Rightarrow DRAM CAS N,
          zs_cke_from_the_sdram_0 => DRAM_CKE,
          zs_cs_n_from_the_sdram_0 => DRAM_CS_N,
          zs_dq_to_and_from_the_sdram_0 => DRAM_DQ,
          zs_dqm_from_the_sdram_0 => DRAM_DQM,
          zs_ras_n_from_the_sdram_0 => DRAM_RAS_N,
          zs_we_n_from_the_sdram_0 => DRAM_WE_N );
     DRAM_CLK <= CLOCK_50;
END Structure;
```

Figure 9. A first attempt at instantiating the expanded Nios II system. (Part *b*).

As an experiment, you can enter the code in Figure 9 into a file called *lights.vhd*. Add this file and all the \*.vhd files produced by the SOPC Builder to your Quartus II project. Compile the code and download the design into the Cyclone IV FPGA on the DE2-115 board. Use the application program from the tutorial *Introduction to the Altera SOPC Builder Using VHDL Designs*, which is shown in Figure 10. Notice in our expanded system, the addresses assigned by the SOPC Builder are 0x10003000 for Switches and 0x10003010 for LEDs, which are different from the original system. These changes are already reflected in the program in Figure 10.

.include	"nios_m	acros.s"
.equ	Switche	s, 0x10003000
.equ	LEDs, 0	x10003010
.global	_start	
_start:		
	movia	r2, Switches
	movia	r3, LEDs
loop:	ldbio	r4, 0(r2)
	stbio	r4, 0(r3)
	br	loop

Figure 10. Assembly language code to control the lights.

Use the Altera Monitor Program, which is described in the tutorial *Altera Monitor Program*, to assemble, download, and run this application program. If successful, the lights on the DE2-115 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 Monitor Program, which may display the message depicted in Figure 11. 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 Contraction of the second se		
Downloading 04000000 ( 0%)		
Downloaded 1KB in 0.0s		
Verifying 04000000 ( 0%)		
Verify failed between address 0x4000000 and 0x400001B		
Leaving target processor paused		
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 11. Error message in the Altera Monitor Program that may be due to the SDRAM clock skew problem.

### 7 Using Clock Signals IP Core

The clock skew depends on physical characteristics of the DE2-115 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 which can be manually created using the

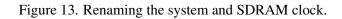
*MegaWizard* plug-in. It can also be created automatically using the Clock Signals IP core provided by the Altera University Program. We will use the latter method in this tutorial.

To add the Clock Signals IP core, in the SOPC Builder window of Figure 3 select University Program > Clock Signals for DE-Series Board Peripherals and click Add. A window depicted in Figure 12 appears. Select *DE2-115* from the DE Board drop-down list and uncheck Video and Audio clocks as these peripherals are not used in this tutorial. Click Finish to return to the window in Figure 3. Now, select the command System > Auto-Assign Base Addresses to re-assign the base address of the Clock Signals IP core. In this tutorial, we will name the system and SDRAM clocks as *sys\_clk* and *sdram\_clk*, respectively. In order to do so, in the Clock Settings window, double-click on the name of the clocks and rename them as shown in Figure 13. All cores, except Clock Signals, should be clocked using the system clock *sys\_clk*. This assignment can done by choosing the correct clock from the drop-down box in the Clock column for each core. The final system is shown in Figure 14. Click on the System Generation tab and regenerate the system.

Clocks Signals	for DE-Series Board Peripherals - clocks_0	
	Ks Signals for DE-Series Board Peripherals	Info
Block Diagram		
	clocks_0         clock - clk_in_primary       sys_clk - clock         avalon - avalon_clocks_slave       sdram_clk - clock	
Configurations		
DE Board:	DE2-115 💌	
Optional Clocks		
SDRAM:		
Video (VGA and 5M	IP Digitial Camera): 📃	
Audio:		
Audio Clock Freque	ncy: 12.288 💉 MHz	
L	Cancel	Finish

Figure 12. Clock Signals IP Core

Name	Source	MHz	Add
:lk_0	External	50.0	
sys_cik sdram_cik	clocks_0.sys_clk	50.0	Remo
sdram_clk	clocks_0.sdram_cik	50.0	



vice Family: C	C S	Name Ik_0 yys_clk dram_clk Description	Source External clocks_0.sys_clk clocks_0.sdram_clk	MHz 50.0 50.0 50.0		Add
e Conn	Module Name	ys_cik xdram_cik	clocks_0.sys_clk	50.0		
	Module Name	ys_cik xdram_cik	clocks_0.sys_clk	50.0		Remove
	Module Name	dram_clk		50.0		
	Module Name					
		Description				
		Description				
	E cou 0		Clock	Base	End	Tags
		Nios II Processor				
	instruction master	Avaion Memory Mapped Master	sys cik			
	data_master	Avaion Memory Mapped Master		IRQ C	D IRQ 31	
	jtag_debug_module	Avaion Memory Mapped Slave		<b>0</b> x10002800	0x10002fff	
	onchip_memory2_0	On-Chip Memory (RAM or ROM)	,			
	s1	Avaion Memory Mapped Slave	sys_clk		0x10001fff	
	Switches	PIO (Parallel I/O)				
$   \rightarrow$	s1	Avalon Memory Mapped Slave	sys_clk	<b>■ 0x10003000</b>	0x1000300f	
	🗆 LEDs	PIO (Parallel I/O)				
	s1	Avalon Memory Mapped Slave	sys_clk	<b>₽ 0x10003010</b>	0x1000301f	
	∃ jtag_uart_0	JTAG UART				
$   \rightarrow$	avalon_jtag_slave	Avaion Memory Mapped Slave	sys_clk	<b>■ 0x10003020</b>	0x10003027	
	⊟ sdram_0	SDRAM Controller				
	s1	Avaion Memory Mapped Slave	sys_clk	<b>₽ 0x0800000</b>	OxOfffffff	
	🗆 clocks_0	-				
$  \rightarrow$	avalon_clocks_slave	Avalon Memory Mapped Slave	cik_0	<b>■ 0x10003028</b>	0x10003029	
			Filtero Filter	. Defeut		
		Address Map	Filler.	. Derault		
		<pre>     onchip_memory2_0     s1     Switches     s1     LEDs     s1     itag_uart_0     avalon_itag_slave     sf1     clocks_0     avalon_clocks_slave </pre>	On-Chip Memory (RAM or ROM) S1 Avaion Memory Mapped Slave Switches PIO (Parallel I/O) S1 Avaion Memory Mapped Slave LEDs PIO (Parallel I/O) s1 Avaion Memory Mapped Slave PIO (Parallel I/O) Avaion Memory Mapped Slave starm_0 starm_0 starm_0 starm_0 sorAM Controller starm_0 starm_0 sorAM Controller starm_0 sorAM Controller starm_0 starm_0 sorAM Controller starm_0 sta	□ onchip_memory2_0       On-Chip Memory (RAM or ROM) Avaion Memory Mapped Slave       sys_clk         □ Switches       PIO (Parallel I/O) s1       Avaion Memory Mapped Slave       sys_clk         □ LEDs       PIO (Parallel I/O) s1       Avaion Memory Mapped Slave       sys_clk         □ LEDs       PIO (Parallel I/O) s1       Avaion Memory Mapped Slave       sys_clk         □ avaion_itag_slave       Avaion Memory Mapped Slave       sys_clk         □ stram_0       SDRAM Controller s1       Avaion Memory Mapped Slave       sys_clk         □ clocks_0 avaion_clocks_slave       Clocks Signal for DE-Series Board Pe Avaion Memory Mapped Slave       clk_0	□ onchip_memory2_0       On-Chip Memory (RAM or ROM) Avaion Memory Mapped Slave       sys_cik       □ 0x10001000         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003000         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003000         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003000         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003010         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003010         St       Avaion Memory Mapped Slave       sys_cik       □ 0x10003020         st       Avaion Memory Mapped Slave       sys_cik       □ 0x10003020         St       SDRAM Controller       sys_cik       □ 0x088000000         St       Clocks Signate for DE-Series Board Pe       □ 0x088000000         Avaion Memory Mapped Slave       sys_cik       □ 0x088000000         Avaion_don_clocks_slave       Avaion Memory Mapped Slave       sys_cik       □ 0x088000000	□ onchip_memory2_0       On-Chip Memory (RAM or ROM) Avaion Memory Mapped Slave       sys_cik       □ 0x10001000       0x10001fff         □ Switches       PIO (Parallel I/O)       st       0x10003000       0x10003000       0x10003000         □ LEDs       PIO (Parallel I/O)       st       Avaion Memory Mapped Slave       sys_cik       □ 0x10003010       0x1000300f         □ LEDs       PIO (Parallel I/O)       st       Avaion Memory Mapped Slave       sys_cik       □ 0x10003010       0x1000301f         □ staran_0       SDRAM Controller       sys_cik       □ 0x0000000       0x10003020       0x10003027         □ staran_0       SDRAM Controller       Avaion Memory Mapped Slave       sys_cik       □ 0x0000000       0x00ffffff         □ clocks_0       Clocks Signals for DE-Series Board Pe       avaion_clocks_slave       Avaion Memory Mapped Slave       sys_cik       □ 0x0000000       0x00ffffff

Figure 14. The final Nios II system.

Next, we have to fix the top-level VHDL entity, given in Figure 9, to instantiate the Nios II system with the Clock Signals core included. The desired code is shown in Figure 15. The SDRAM clock signal *sdram\_clk* generated by the Clock Signals core connects to the pin *DRAM\_CLK*. Note that the *sys\_clk* signal is not connected since it is for internal use only.

- -- 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 DE2-115 User Manual.

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all;

USE ieee.std\_logic\_unsigned.all;

ENTITY lights IS

PORT (SW: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); KEY: IN STD\_LOGIC\_VECTOR(0 DOWNTO 0); CLOCK\_50: IN STD\_LOGIC; LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); DRAM\_DQ: INOUT STD\_LOGIC\_VECTOR (31 DOWNTO 0); DRAM\_ADDR: OUT STD\_LOGIC\_VECTOR (12 DOWNTO 0); DRAM\_BA: OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0); DRAM\_CAS\_N, DRAM\_RAS\_N, DRAM\_CLK: OUT STD\_LOGIC; DRAM\_CKE, DRAM\_CS\_N, DRAM\_WE\_N: OUT STD\_LOGIC; DRAM\_DQM: OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0));

END lights;

ARCHITECTURE Structure OF lights IS

COMPONENT nios\_system

PORT ( clk\_0 : IN STD\_LOGIC; reset\_n : IN STD\_LOGIC; sdram\_clk : OUT STD\_LOGIC; sys\_clk : OUT STD\_LOGIC; out\_port\_from\_the\_LEDs : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); in\_port\_to\_the\_Switches : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); zs\_addr\_from\_the\_sdram\_0 : OUT STD\_LOGIC\_VECTOR(12 DOWNTO 0); zs\_ba\_from\_the\_sdram\_0 : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0); zs\_cas\_n\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_cke\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_cs\_n\_from\_the\_sdram\_0 : OUT STD\_LOGIC; zs\_dq\_to\_and\_from\_the\_sdram\_0 : INOUT STD\_LOGIC\_VECTOR(31 DOWNTO 0);

... continued in Part b

Figure 15. Proper instantiation of the expanded Nios II system. (Part *a*).

```
zs_dqm_from_the_sdram_0 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
          zs_we_n_from_the_sdram_0 : OUT STD_LOGIC);
  END COMPONENT;
BEGIN
-- Instantiate the Nios II system entity generated by the SOPC Builder.
  NiosII: nios_system
     PORT MAP (
          clk_0 \Rightarrow CLOCK_50,
          reset n \Rightarrow KEY(0),
          sdram_clk => DRAM_CLK,
          out port from the LEDs => LEDG,
          in_port_to_the_Switches => SW,
          zs addr from the sdram 0 \Rightarrow DRAM ADDR,
          zs ba from the sdram 0 \Rightarrow DRAM BA,
          zs_cas_n_from_the_sdram_0 => DRAM_CAS_N,
          zs_cke_from_the_sdram_0 => DRAM_CKE,
          zs_cs_n_from_the_sdram_0 => DRAM_CS_N,
          zs_dq_to_and_from_the_sdram_0 => DRAM_DQ,
          zs_dqm_from_the_sdram_0 => DRAM_DQM,
          zs_ras_n_from_the_sdram_0 => DRAM_RAS_N,
          zs_we_n_from_the_sdram_0 => DRAM_WE_N );
END Structure;
```

Figure 15. Proper instantiation of the expanded Nios II system. (Part b).

Compile the code and download the design into the Cyclone IV FPGA on the DE2-115 board. Use the application program in Figure 10 to test the circuit.

Copyright ©2010 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.