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

This tutorial explains how the SDRAM chip 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 Design.

The screen captures in the tutorial were obtained using the Quartus^R II version 9.1; if other versions of the software are used, some of the images may be slightly different.

Contents:

The introductory tutorial Introduction to the Altera SOPC Builder Using VHDL Design 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 two SDRAM chip that can store 64Mbytes of data. This memory is organized as 32M x 32 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 VHDL Design tutorial. Figure 1 gives the block diagram of our example system.

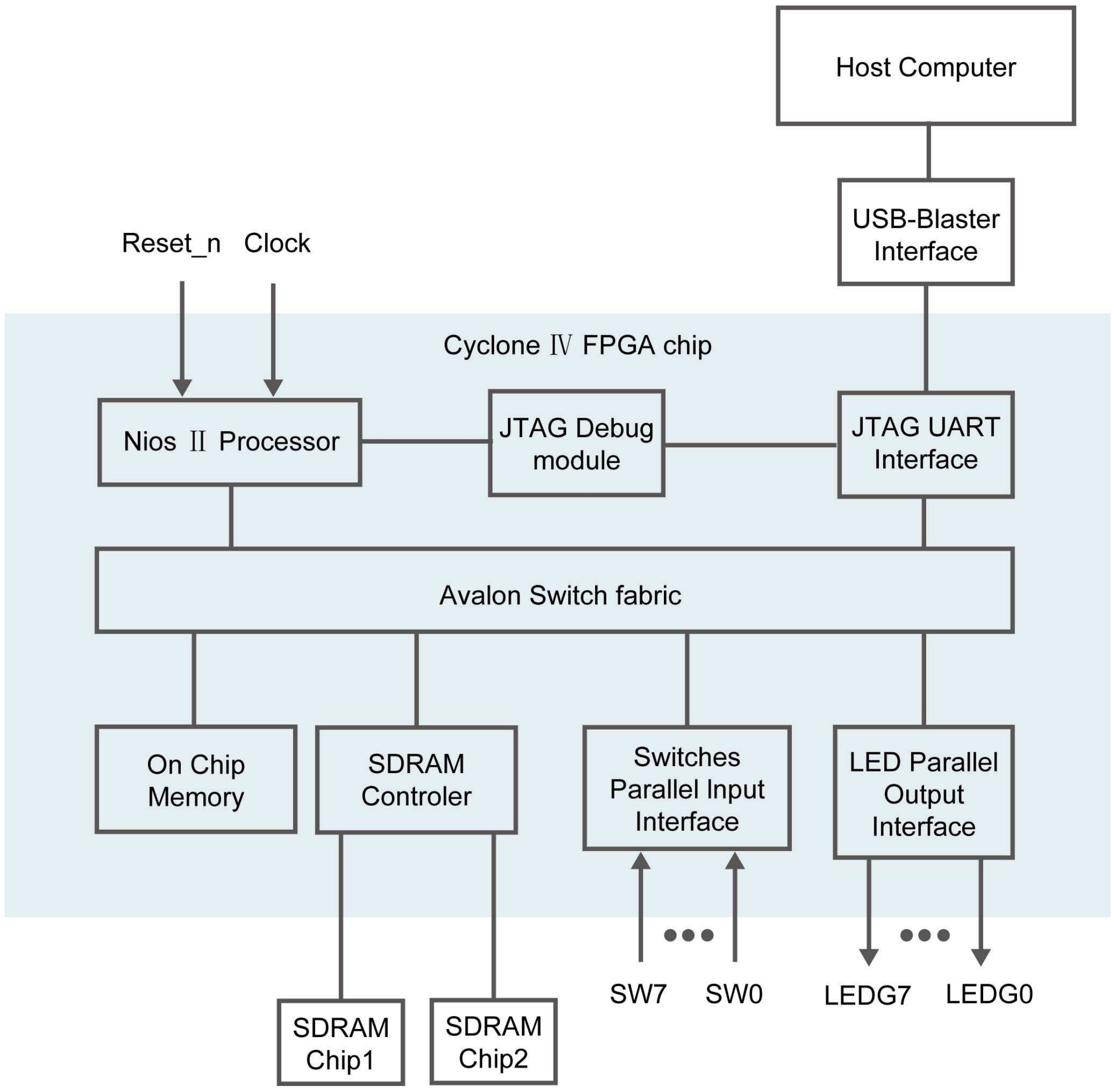


Figure 1.Example Nios II system implemented on the DE2-115 board.

The system realizes a trivial task. Eight toggle switches on the DE2-115 board, SW 7−0, 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 DE2-115 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**
- **Using a phase-locked loop (PLL) to control the clock timing**

2 The SDRAM Interface

The SDRAM chip on the DE2-115 board has the capacity of 128 Mbytes. It is organized as 32M x 32 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.

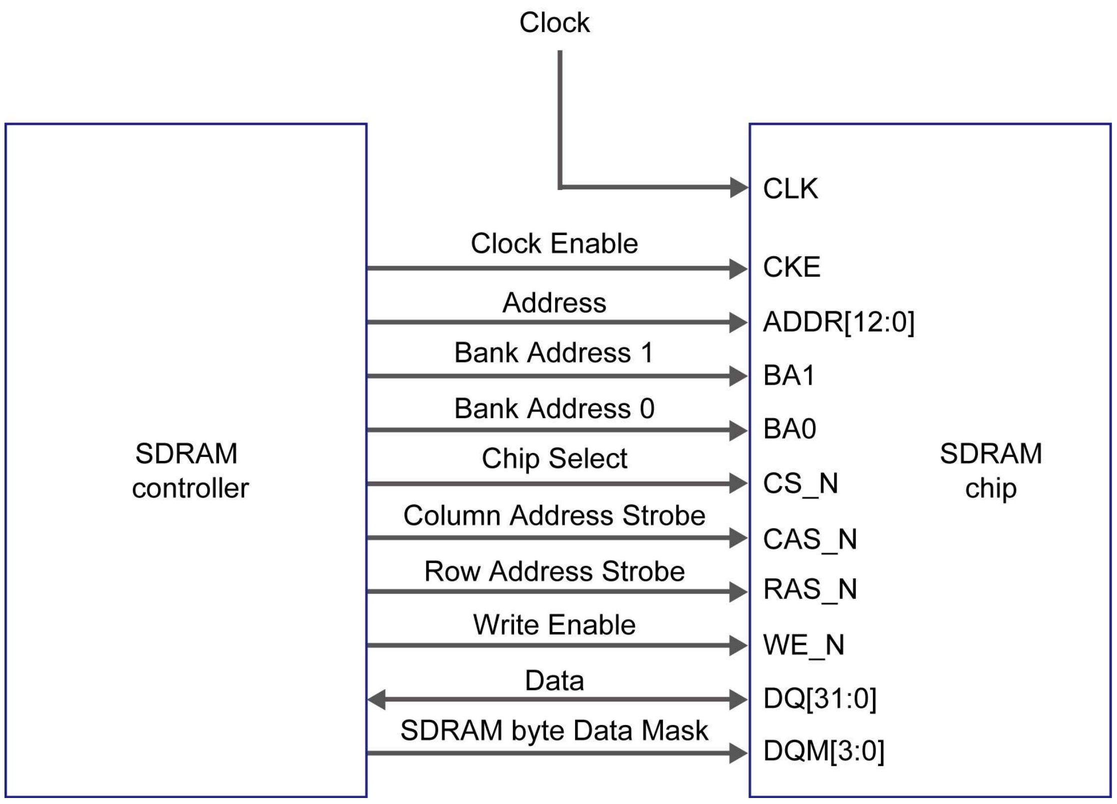
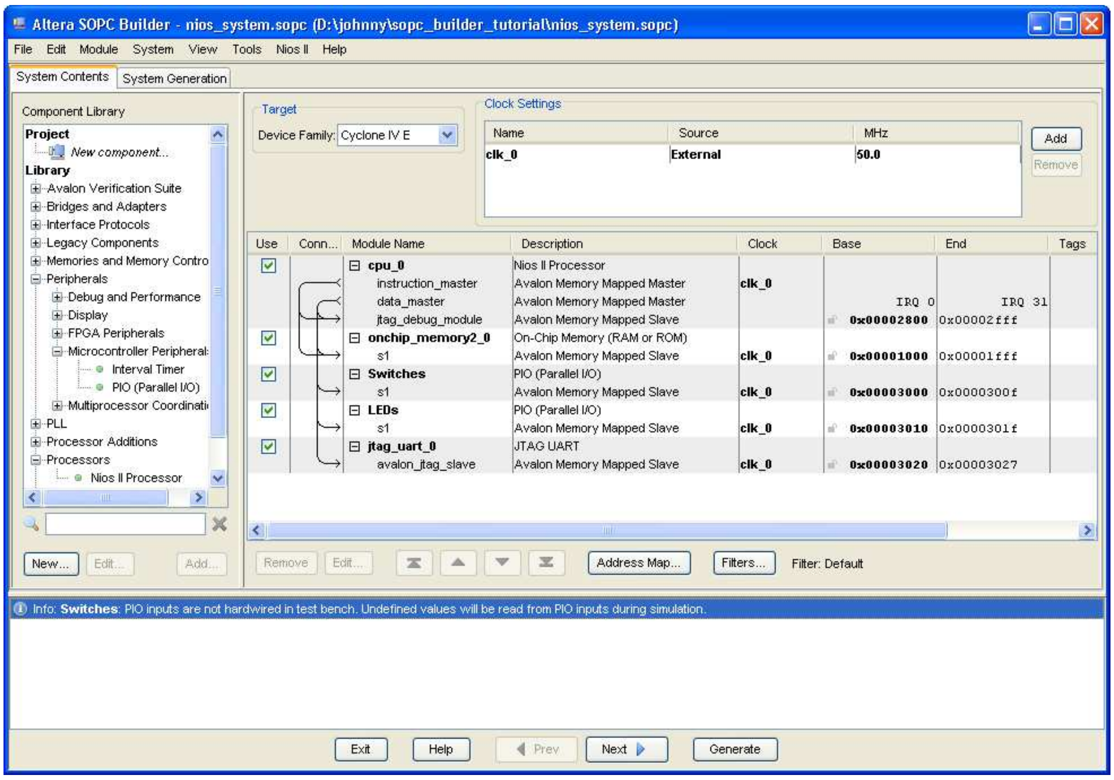


Figure 2.The SDRAM signals.

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 VHDL Design tutorial, which we implemented in a project called lights. We specify the system shown in Figure 3.



Device Family: Cyclone IV E

Name	Source	MHz
clk_0	External	50.0

Use	Conn...	Module Name	Description	Clock	Base	End	Tags
✓		cpu_0	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	clk_0			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00002800	0x00002fff	
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00001000	0x00001fff	
✓		Switches	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00003000	0x0000300f	
✓		LEDs	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00003010	0x0000301f	
✓		jtag_uart_0	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00003020	0x00003027	

Filter: Default

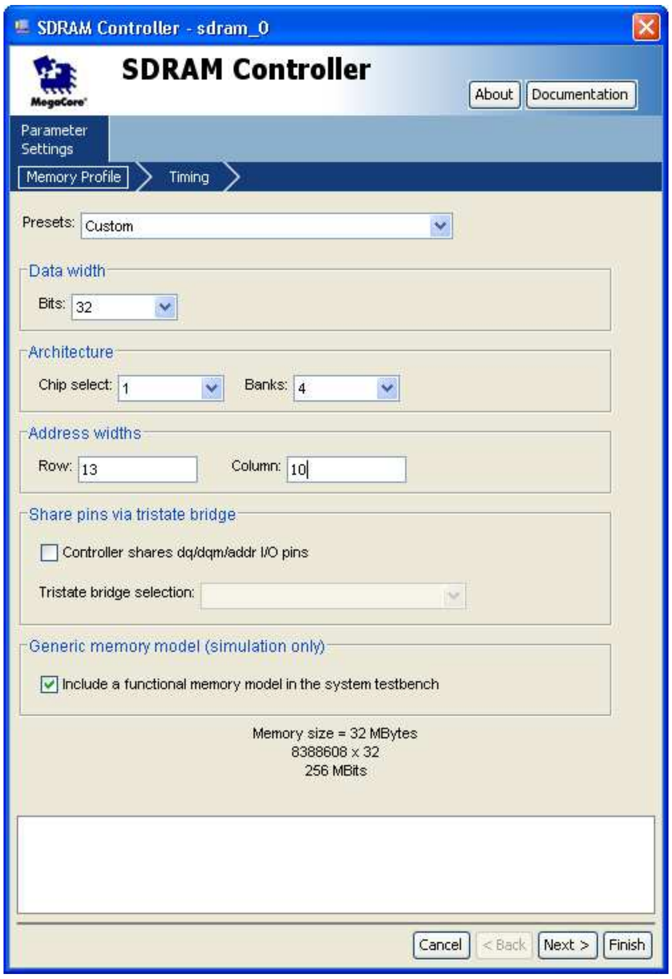
Info: **Switches**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

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. 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 DE2-115 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

SDRAM Controller

Parameter Settings

Memory Profile › Timing

Presets: Custom

Data width

Bits: 32

Architecture

Chip select: 1   Banks: 4

Address widths

Row: 13   Column: 10

Share pins via tristate bridge

☐ Controller shares dq/dqm/addr I/O pins

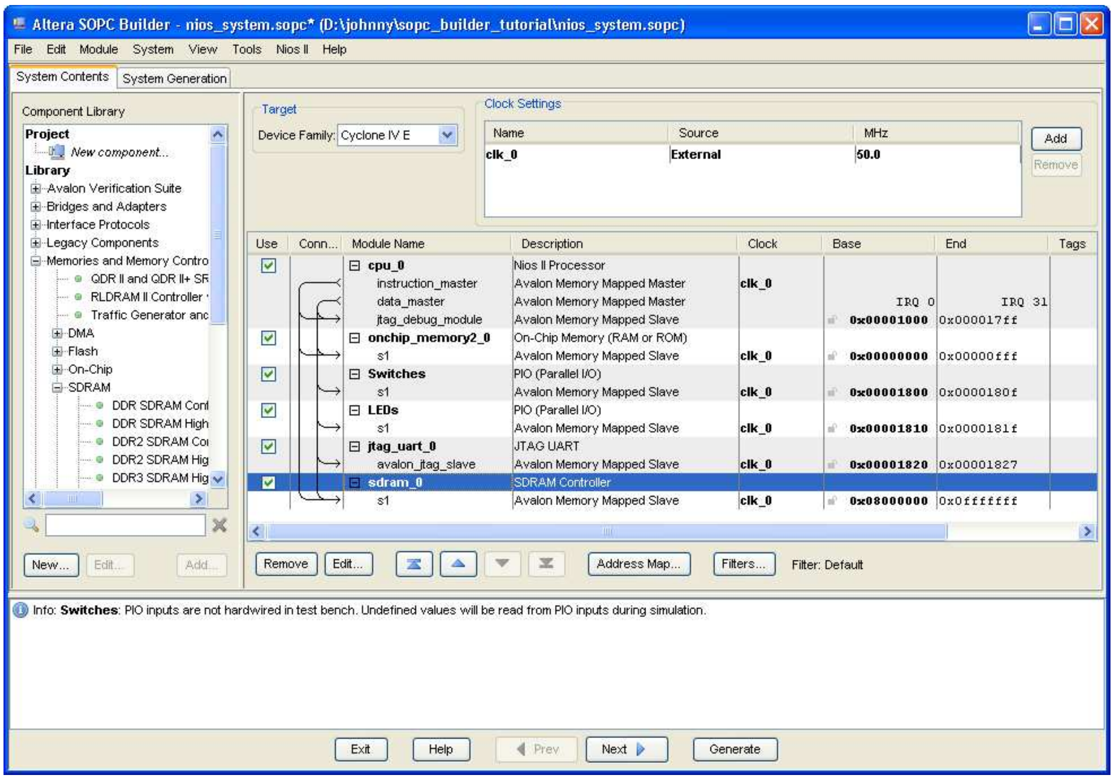
Tristate bridge selection:

Generic memory model (simulation only)

☑ Include a functional memory model in the system testbench

Memory size = 32 MBytes
8388608 x 32
256 MBits

Figure 4.Add the SDRAM Controller.


Altera SOPC Builder - nios_system.sopc* (D:\johnny\sopc_builder_tutorial\nios_system.sopc)
File Edit Module System View Tools Nios II Help
System Contents System Generation
Component Library
Project
New component...
Library
Avalon Verification Suite
Bridges and Adapters
Interface Protocols
Legacy Components
Memories and Memory Contro
QDR II and QDR II+ SR
RLDRAM II Controller
Traffic Generator anc
DMA
Flash
On-Chip
SDRAM
DDR SDRAM Cont
DDR SDRAM High
DDR2 SDRAM Co
DDR2 SDRAM Hig
DDR3 SDRAM Hig
New... Edit... Add...
Remove Edit... Address Map... Filters... Filter: Default
Exit Help Prev Next Generate

Target

Device Family: Cyclone IV E

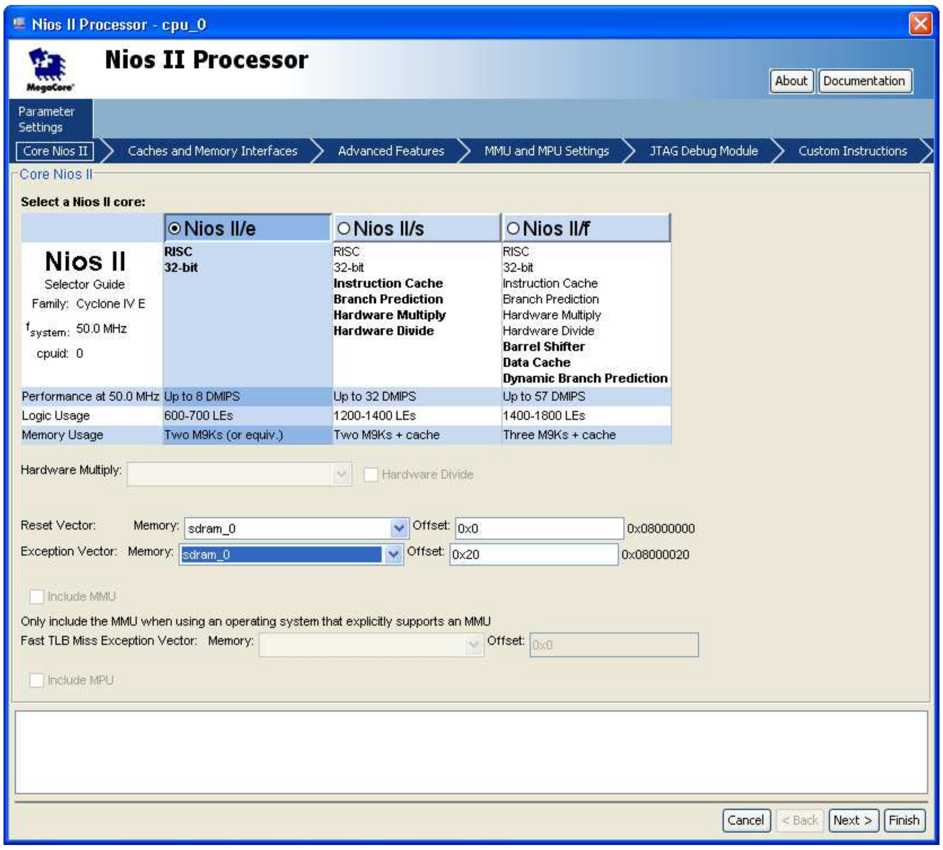
Clock Settings

Name	Source	MHz
clk_0	External	50.0

Use	Conn...	Module Name	Description	Clock	Base	End	Tags
✓		cpu_0	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	clk_0			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00001000	0x000017ff	
✓		onchip_memory2_0	On-Chip Memory (RAM or ROM)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00000000	0x00000fff	
✓		Switches	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00001800	0x0000180f	
✓		LEDs	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00001810	0x0000181f	
✓		jtag_uart_0	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00001820	0x00001827	
✓		sdram_0	SDRAM Controller				
		s1	Avalon Memory Mapped Slave	clk_0	0x08000000	0x0fffffff	

Info: **Switches**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Figure 5. The expanded Nios II system.



Core Nios II

Select a Nios II core:

	[✓ Nios II/e]	Nios II/s	Nios II/f
Nios II Selector Guide Family: Cyclone IV E f_{system}: 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 Dynamic Branch Prediction
Performance at 50.0 MHz	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: ☐ Hardware Divide

Reset Vector:	Memory:	sdram_0	Offset:	0x0	0x08000000
Exception Vector:	Memory:	sdram_0	Offset:	0x20	0x08000020

☐ Include MMU

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

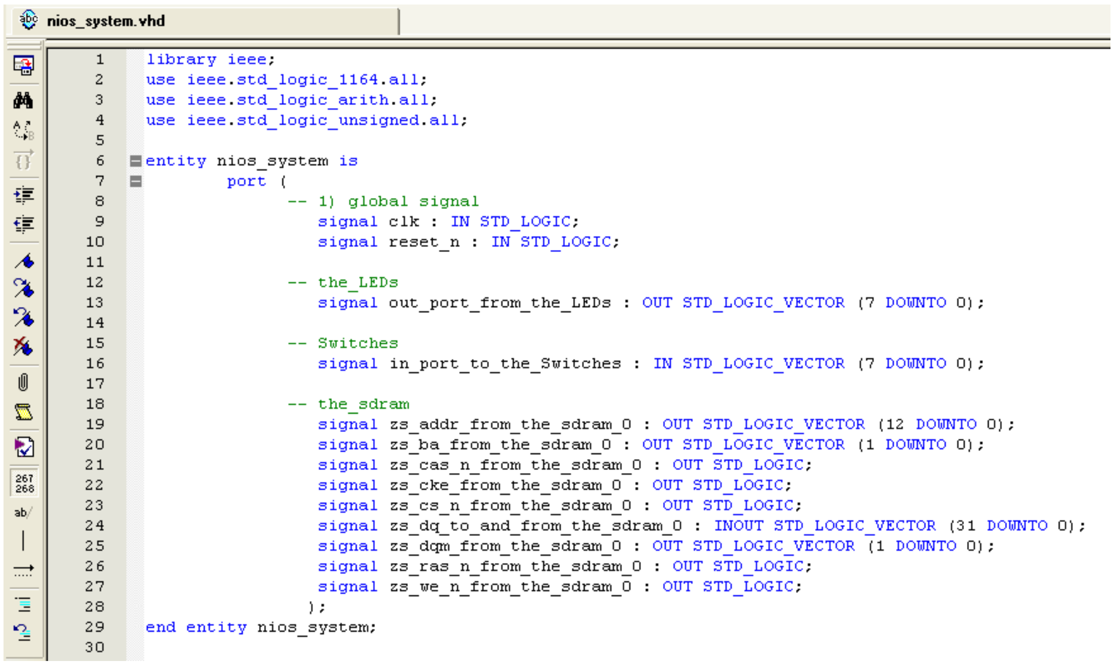
Fast TLB Miss Exception Vector: Memory: Offset: 0x0

☐ Include MPU

Figure 6. Set CPU Reset vector and Exception memory to sdram_0

The augmented VHDL entity generated by the SOPC Builder is in the file nios_system.vhd in the directory of the project. Figure 7 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 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[12:0]. The data lines are referred to as the INOUT vector zs_dq_to_and_from_the_sdram[31:0]. This is a vector of the INOUT type because the data lines are bidirectional.



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity nios_system is
        port (
            -- 1) global signal
               signal clk : IN STD_LOGIC;
               signal reset_n : IN STD_LOGIC;

            -- the_LEDs
               signal out_port_from_the_LEDs : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);

            -- Switches
               signal in_port_to_the_Switches : IN STD_LOGIC_VECTOR (7 DOWNTO 0);

            -- the_sdram
               signal zs_addr_from_the_sdram_0 : OUT STD_LOGIC_VECTOR (12 DOWNTO 0);
               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;
               signal zs_cke_from_the_sdram_0 : OUT STD_LOGIC;
               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);
               signal zs_dqm_from_the_sdram_0 : OUT STD_LOGIC_VECTOR (1 DOWNTO 0);
               signal zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
               signal zs_we_n_from_the_sdram_0 : OUT STD_LOGIC;
              );
end entity nios_system;

```

Figure 7. A part of the generated VHDL entity.

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 VHDL entity, as we have done in the tutorial Introduction to the Altera SOPC Builder Using VHDL Design. 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 8. 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.csv in the directory DE2-115_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 pages.

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 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 5.

```
−− 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_CLK, DRAM_CKE : OUT STD_LOGIC;
DRAM_ADDR : OUT STD_LOGIC_VECTOR(12 DOWNTO 0);
DRAM_BA : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N, DRAM_WE_N : OUT STD_LOGIC;
```

```
DRAM_DQ : INOUT STD_LOGIC_VECTOR(31 DOWNTO 0);
DRAM_DQM : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
END lights;

ARCHITECTURE Structure OF lights IS
COMPONENT nios_system
PORT ( clk : 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 : OUT STD_LOGIC_VECTOR(12 DOWNTO 0);
zs_ba_from_the_sdram : BUFFER STD_LOGIC_VECTOR(1 DOWNTO 0);
zs_cas_n_from_the_sdram : OUT STD_LOGIC;
zs_cke_from_the_sdram : OUT STD_LOGIC;
zs_cs_n_from_the_sdram : OUT STD_LOGIC;
zs_dq_to_and_from_the_sdram : INOUT STD_LOGIC_VECTOR(31 DOWNTO 0);
zs_dqm_from_the_sdram : BUFFER STD_LOGIC_VECTOR(1 DOWNTO 0);
zs_ras_n_from_the_sdram : OUT STD_LOGIC;
zs_we_n_from_the_sdram : OUT STD_LOGIC );
END COMPONENT;
SIGNAL BA : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL DQM : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
BA <= DRAM_BA;
DQM <= DRAM_DQM;
−− Instantiate the Nios II system entity generated by the SOPC Builder.
NiosII: nios_system PORT MAP (CLOCK_50, KEY(0), LEDG, SW,
DRAM_ADDR, BA, DRAM_CAS_N, DRAM_CKE, DRAM_CS_N,
DRAM_DQ, DQM, DRAM_RAS_N, DRAM_WE_N );
DRAM_CLK <= CLOCK_50;
END Structure;
```

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.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 Design, which is shown in Figure 9.

```
.include "nios_macros.s"
.equ Switches, 0x00001800
.equ LEDs, 0x00001810
```

```
.global _start
_start:
movia     r2, Switches
movia     r3, LEDs
loop:     ldbio     r4, 0(r2)
          stbio     r4, 0(r3)
          br        loop
```

Figure 9.Assembly language code to control the lights.