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® 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.

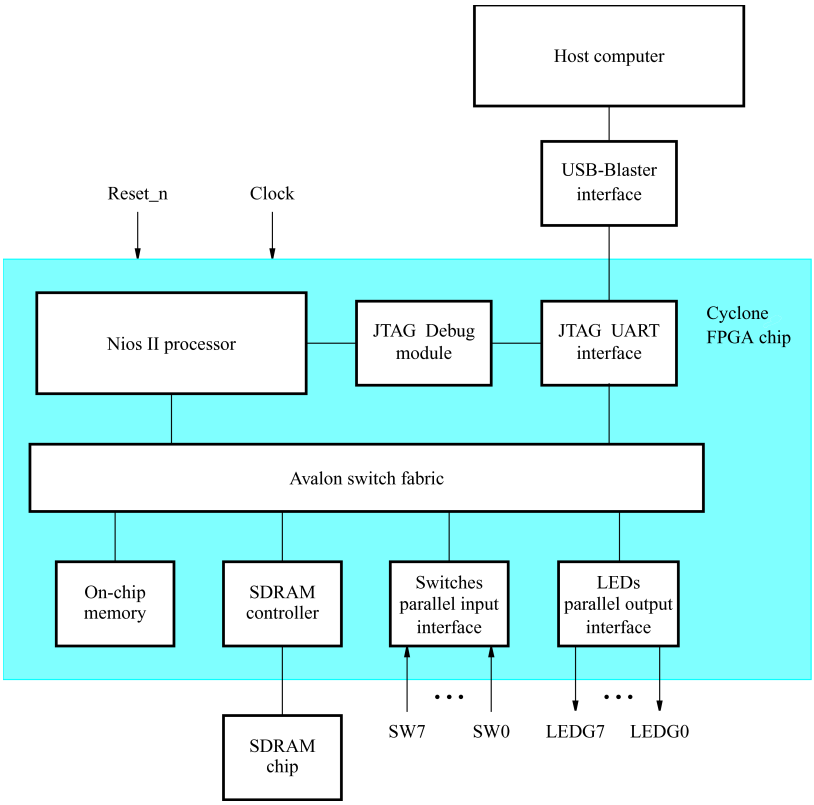

Figure 1. Example Nios II system implemented on the DE2-115 board.

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.

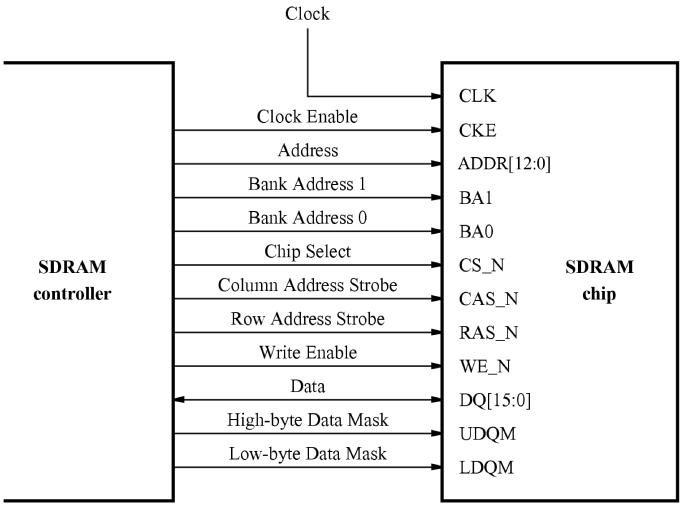

Figure 2. The SDRAM signals.

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.

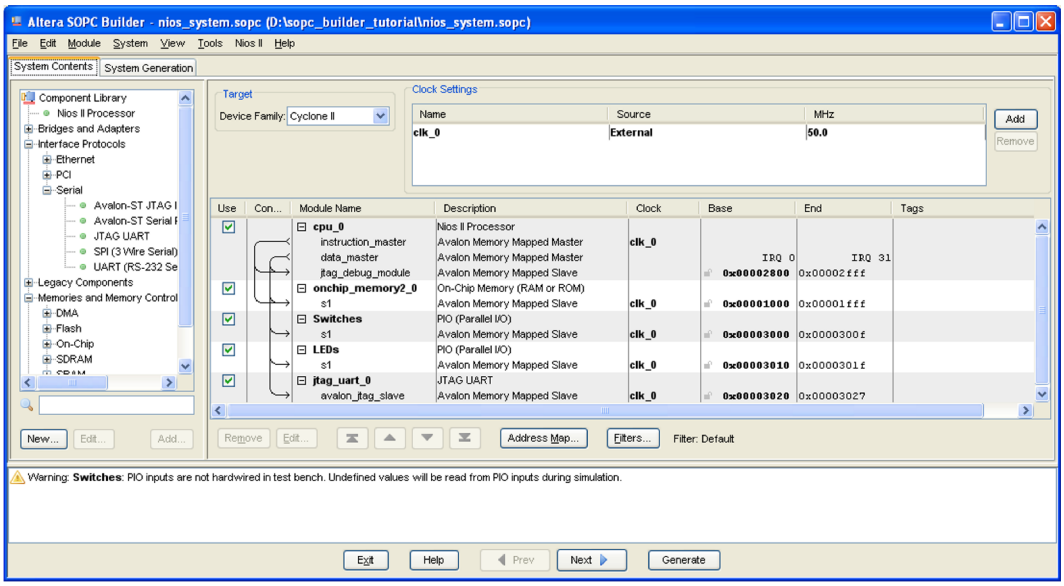

Figure 3. The Nios II system defined in the introductory tutorial.

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.

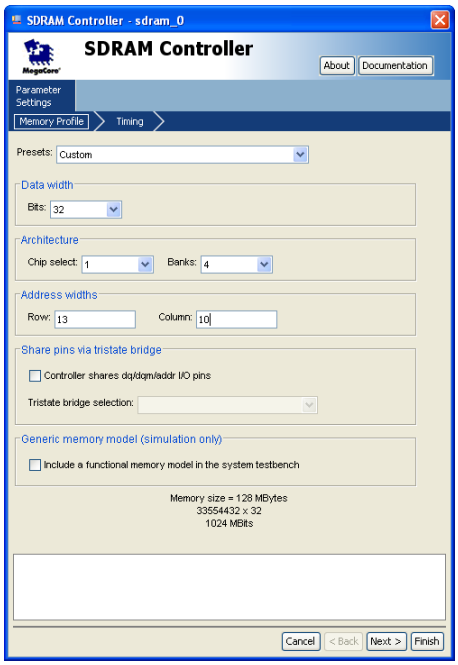

Figure 4. Add the SDRAM Controller.

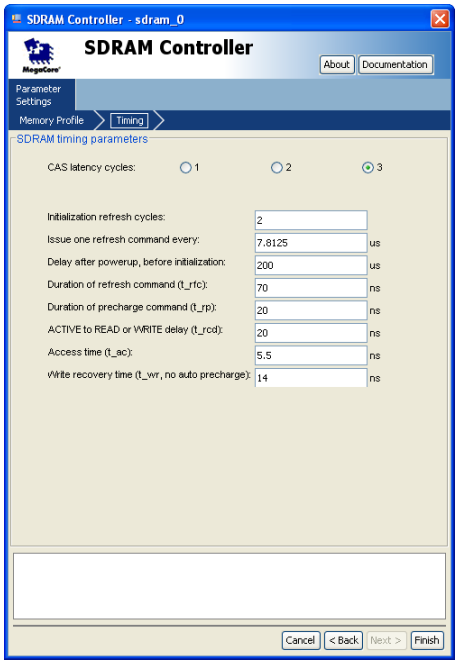

Figure 5. SDRAM Timings

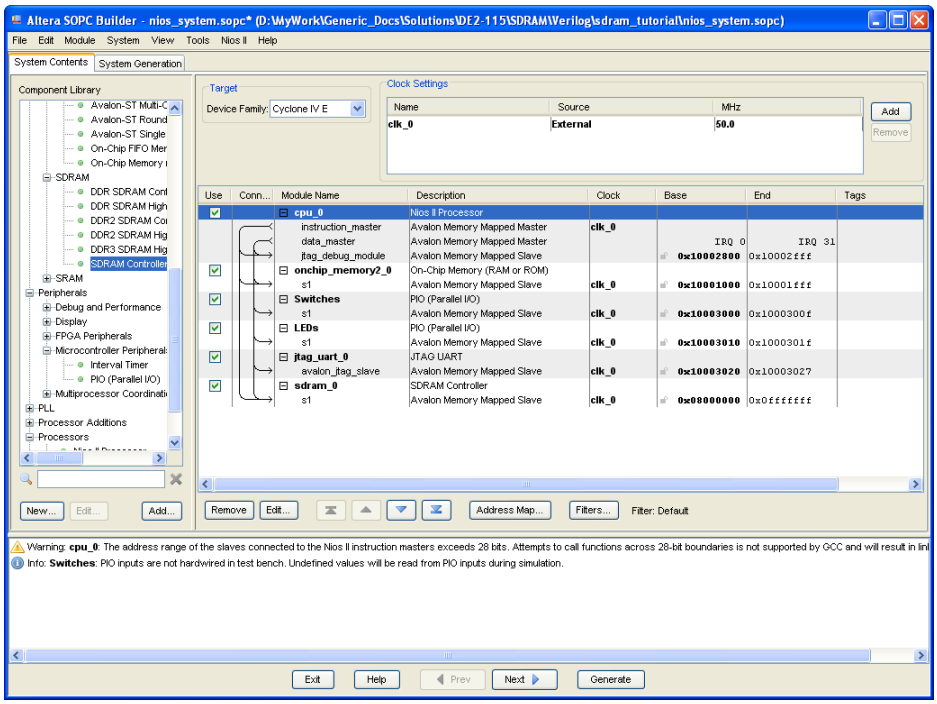

Figure 6. The expanded Nios II system.

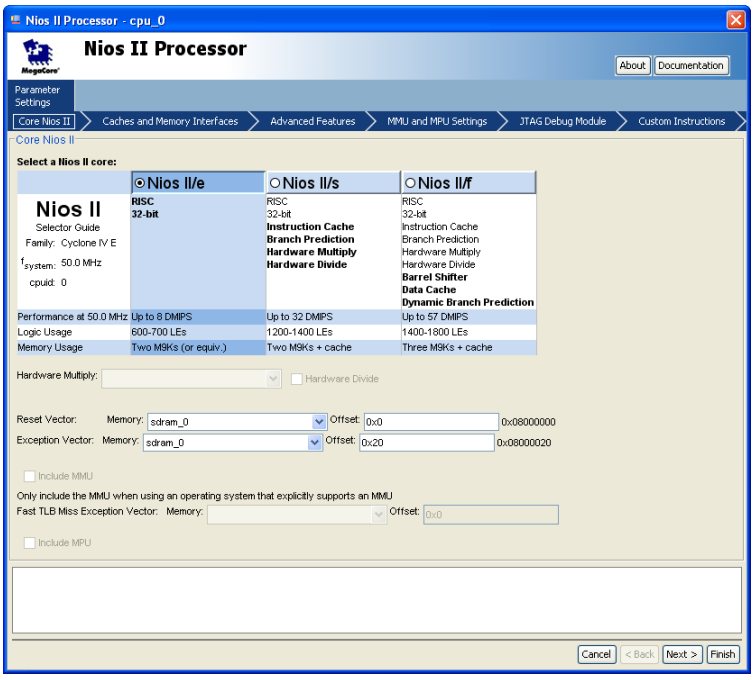

Figure 7. Define the reset vector and the exception vector.

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 (
4011                -- 1) global signals:
4012                   signal clk_0 : IN STD_LOGIC;
4013                   signal reset_n : IN STD_LOGIC;
4014
4015                -- the_LEDs
4016                   signal out_port_from_the_LEDs : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
4017
4018                -- the_Switches
4019                   signal in_port_to_the_Switches : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
4020
4021                -- the_sdram_0
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);
4024                   signal zs_cas_n_from_the_sdram_0 : OUT STD_LOGIC;
4025                   signal zs_cke_from_the_sdram_0 : OUT STD_LOGIC;
4026                   signal zs_cs_n_from_the_sdram_0 : OUT STD_LOGIC;
4027                   signal zs_dq_to_and_from_the_sdram_0 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
4028                   signal zs_dqm_from_the_sdram_0 : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
4029                   signal zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
4030                   signal zs_we_n_from_the_sdram_0 : OUT STD_LOGIC
4031                );
4032  end entity nios_system;
```

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 *b*

Figure 9. A first attempt at instantiating 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 => CLOCK_50,
            reset_n => 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 => 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 );
        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_macros.s"
.equ      Switches, 0x10003000
.equ      LEDs, 0x10003010
.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.

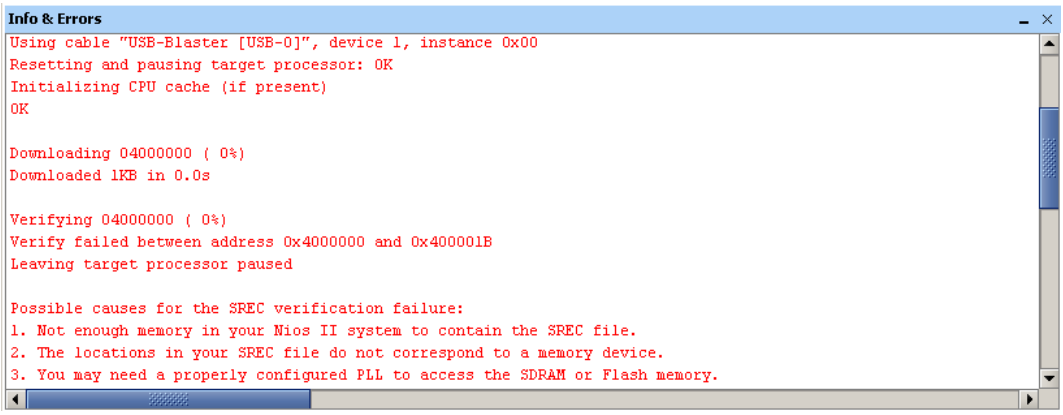

Figure 11. Error message in the Altera Monitor Program that may be due to the SDRAM clock skew problem.

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.

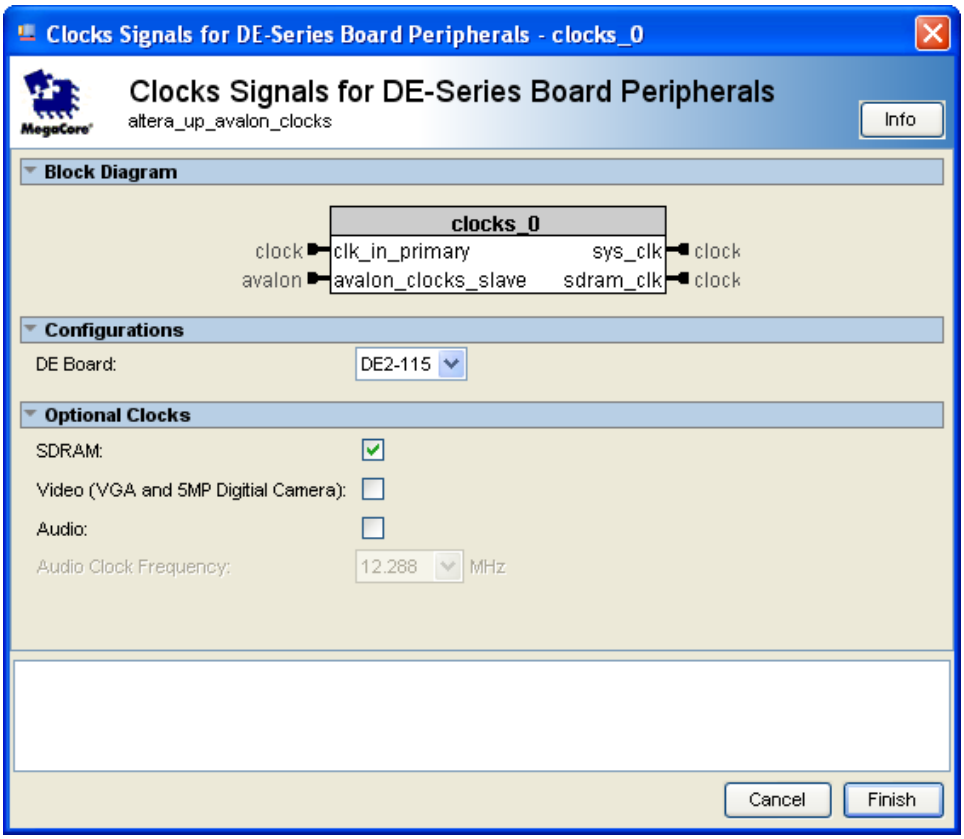

Figure 12. Clock Signals IP Core

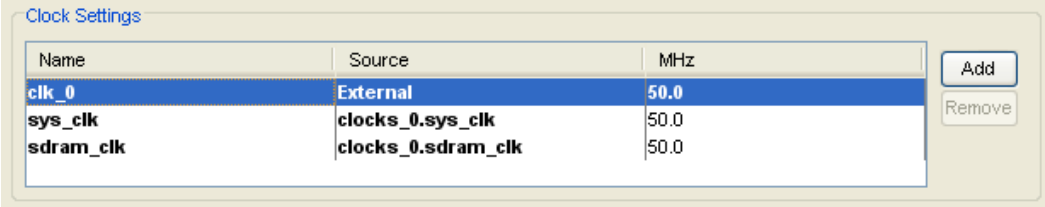

Figure 13. Renaming the system and SDRAM clock.

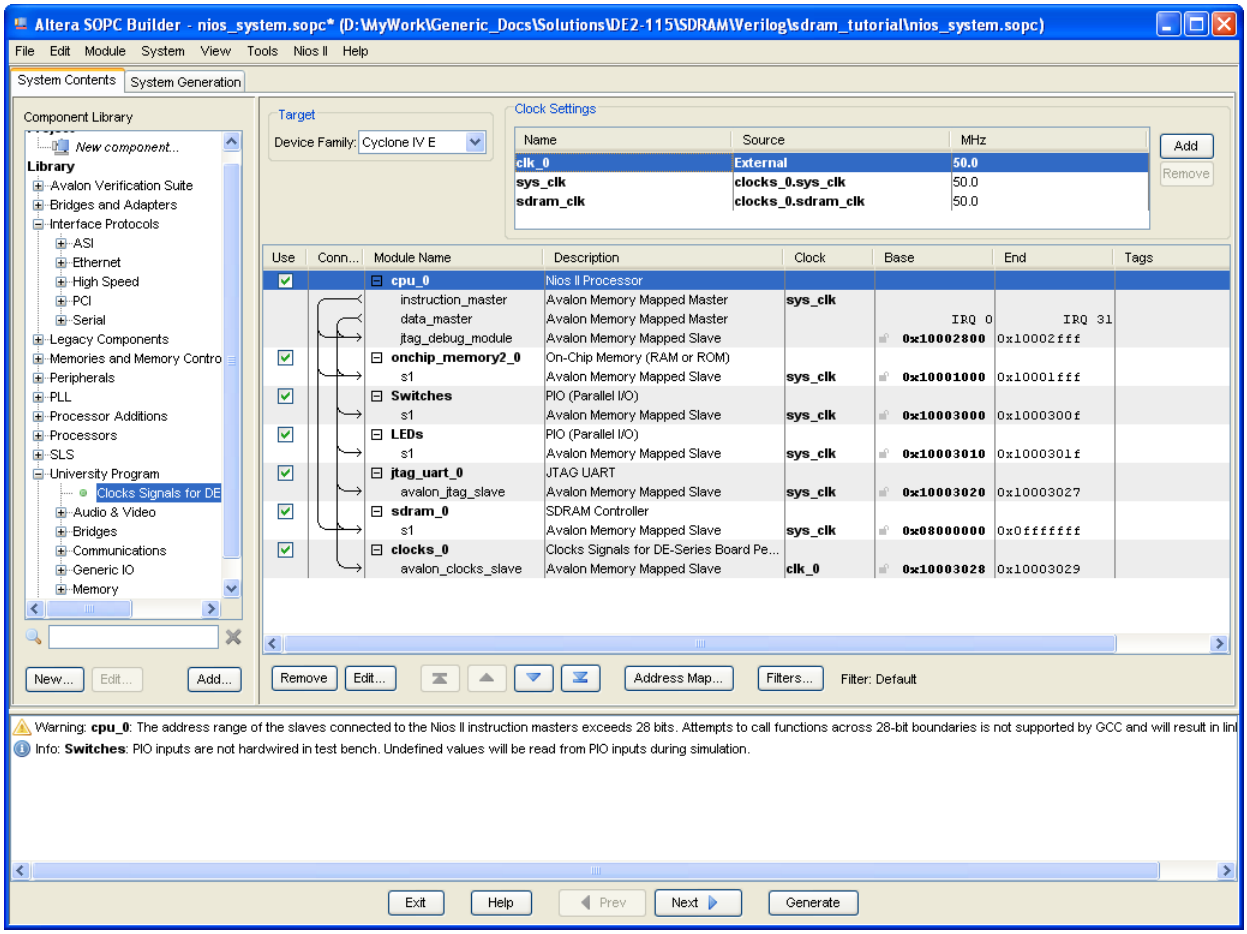

Figure 14. The final Nios II system.

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 => CLOCK_50,
            reset_n => KEY(0),
            sdram_clk => DRAM_CLK,
            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 => 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.