Using the SDRAM Memory on Altera's DE2-70 Board with VHDL Design

This tutorial explains how the SDRAM chips on Altera's DE2-70 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 DE2-70 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® II version 9.0; 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 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 DE2-70 board contains 2 SDRAM chips that can each store 32 Mbytes of data. Each chip is organized as 4M x 16 bits x 4 banks. The SDRAMs chip 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. For the purposes of this tutorial, we will deal with only one of the SDRAM chips. The second chip can be used in a similar fashion.

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.

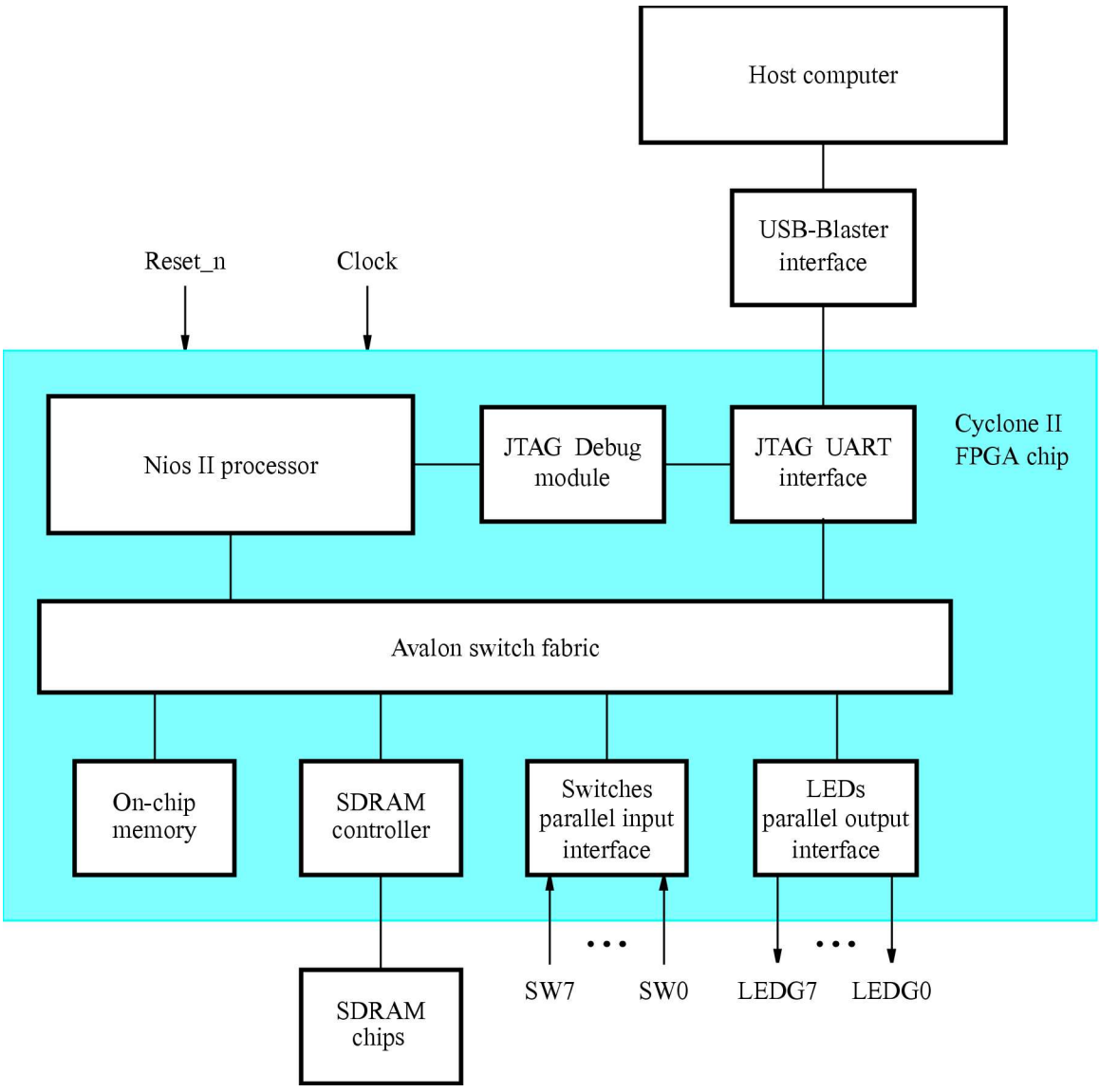

Figure 1. Example Nios II system implemented on the DE2-70 board.

The system realizes a trivial task. Eight toggle switches on the DE2-70 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 an SDRAM chip on the DE2-70 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 DE2-70 board
- Using a phase-locked loop (PLL) to control the clock timing

2 The SDRAM Interface

The two SDRAM chips on the DE2-70 board each have a capacity of 256 Mbits (32 Mbytes). Each chip is organized as 4M 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 5. Note that some signals are active low, which is denoted by the suffix N.

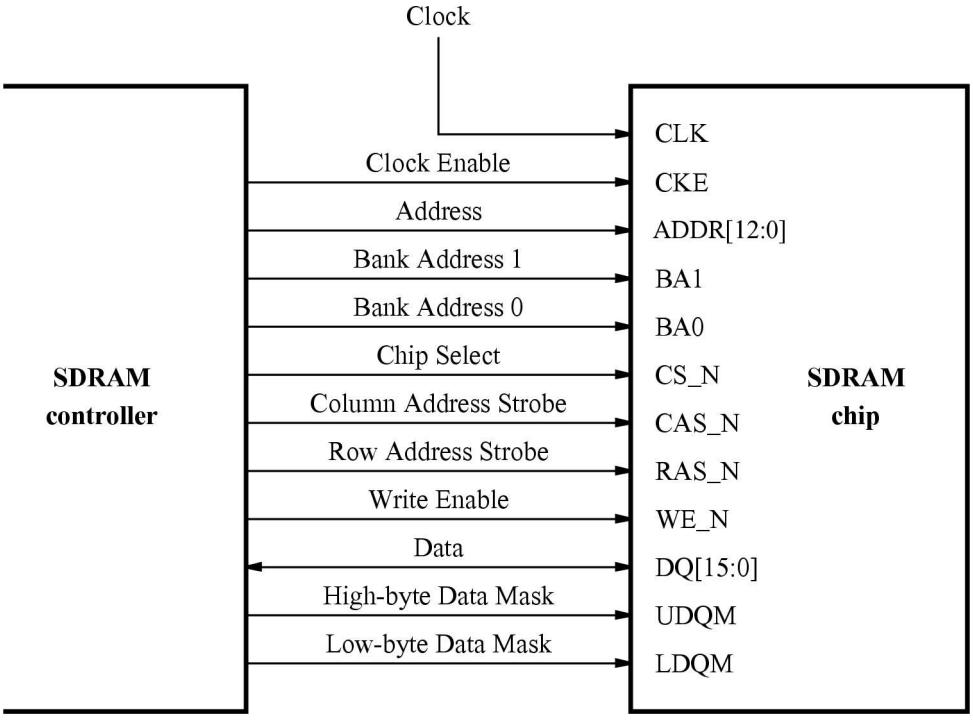

Figure 2. The SDRAM signals.

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 specified the system shown in Figure 3.

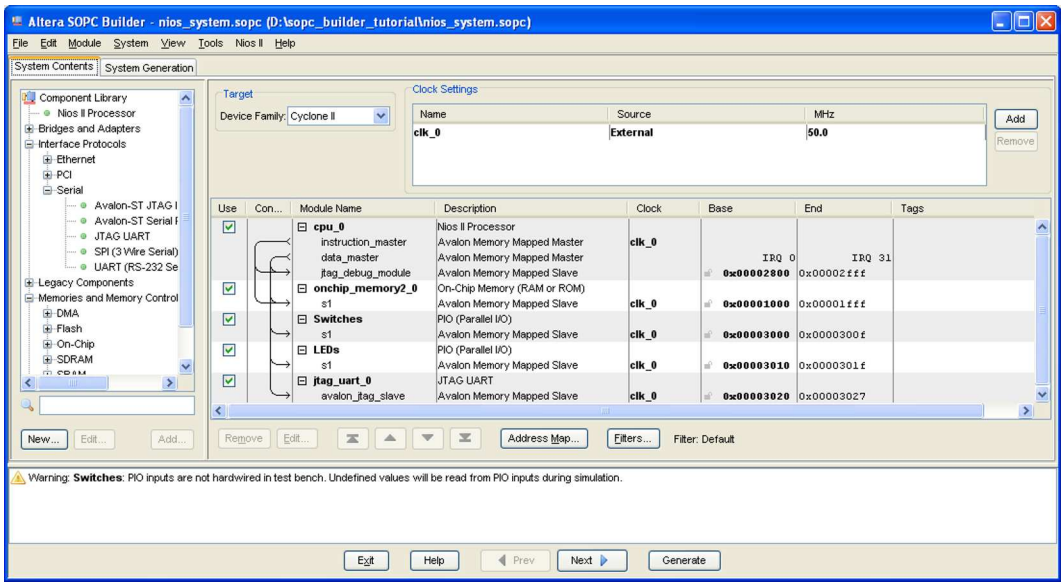

Figure 3. The Nios II system defined in the introductory tutorial.

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, the Row Width to 13 bits, the Column Width to 9 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 0x02000000 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 6. 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.

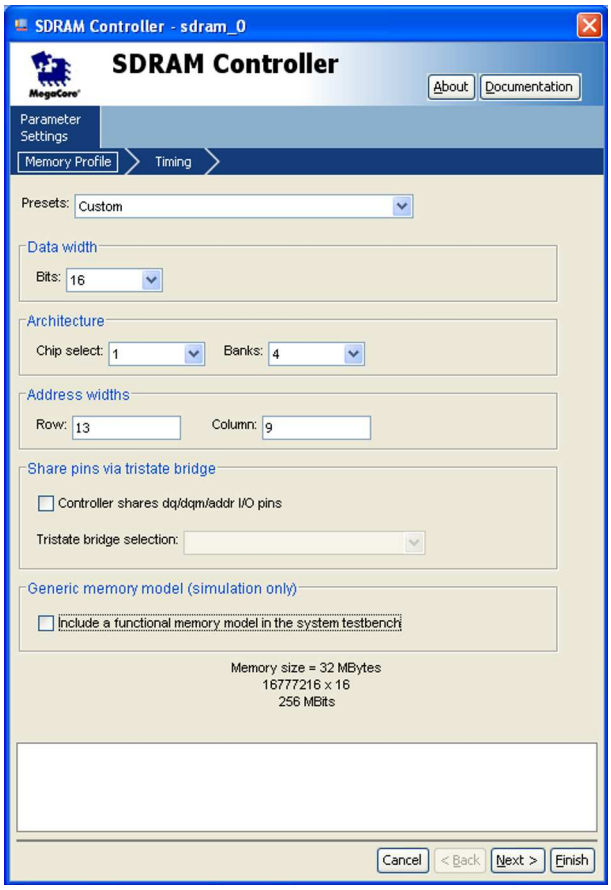

Figure 4. Add the SDRAM Controller.

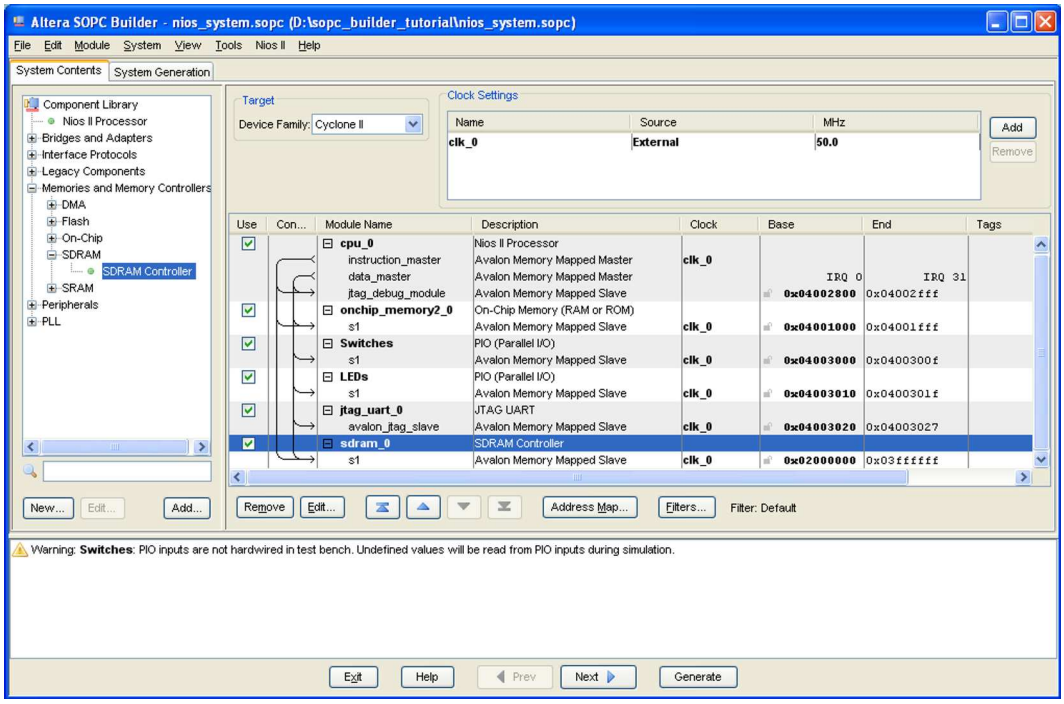

Figure 5. The expanded Nios II system.

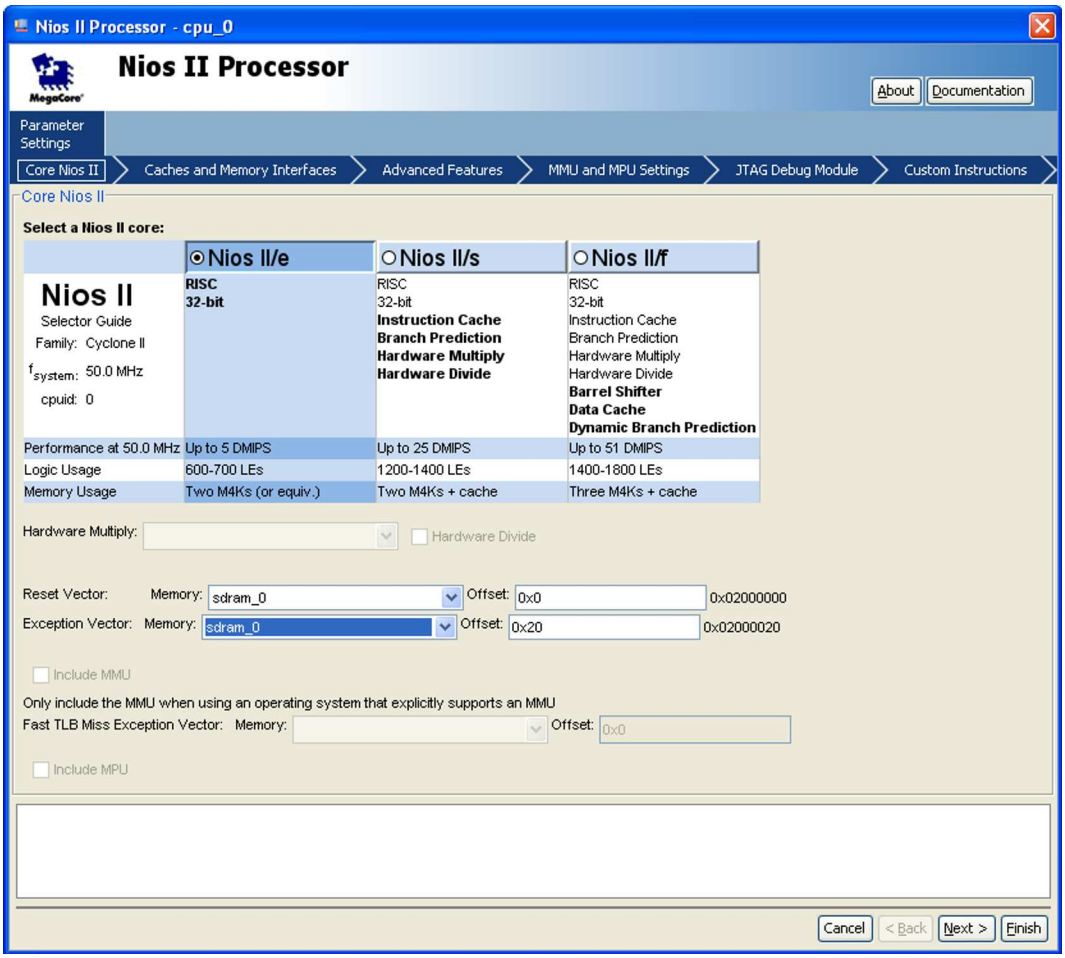

Figure 6. Define the reset vector and the exception vector.

The augmented VHDL entity generated by the SOPC Builder is in the file *nios_system.vhd* in the directory of the project. Figure 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_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[15:0]*. This is a vector of the INOUT type because the data lines are bidirectional.

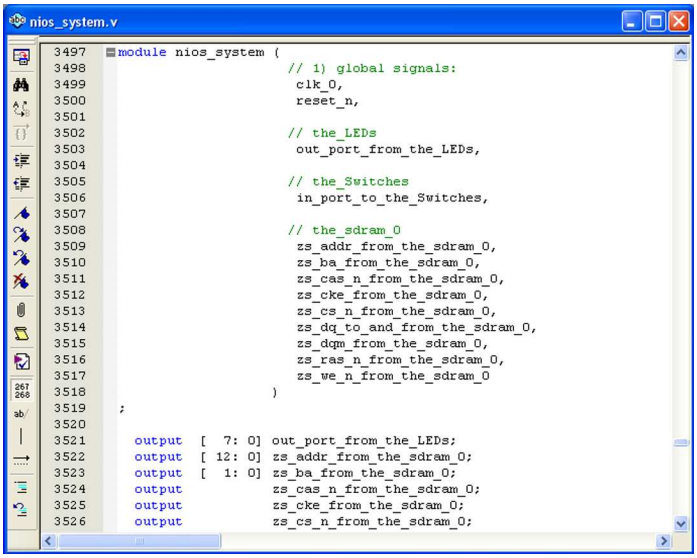

Figure 7. A part of the generated VHDL entity.

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 *DRAM0_CLK*, *DRAM0_CKE*, *DRAM0_ADDR*, *DRAM0_BA_1*, *DRAM0_BA_0*, *DRAM0_CS_N*, *DRAM0_CAS_N*, *DRAM0_RAS_N*, *DRAM0_WE_N*, *DRAM_DQ*, *DRAM0_UDQM*, and *DRAM0_LDQM*, which correspond to the SDRAM signals indicated in Figure 2. All of these names are those specified in the DE2-70 User Manual, which allows us to make the pin assignments by importing them from the file called *DE2_70_pin_assignments.csv* in the directory *DE2_70_tutorials\design_files*, which is included on the CD-ROM that accompanies the DE2-70 board and can also be found on Altera's DE2-70 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_0[1:0]*, as seen in Figure 7. However, in the *DE2_70_pin_assignments.csv* file these signals are given as separate signals *DRAM0_BA_1* and *DRAM0_BA_0*. This is accommodated by our VHDL code. Similarly, the vector *zs_dqm_from_the_sdram_0[1:0]* corresponds to the signals *(DRAM0_UDQM* and *DRAM0_LDQM)*.

Finally, note that we tried an obvious approach of using the 50-MHz system clock, *CLOCK_50*, as the clock signal, *DRAM0_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 DE2-70 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 DE2-70 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)
        DRAM0_CLK, DRAM0_CKE : OUT STD_LOGIC;
        DRAM0_ADDR : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
        DRAM0_BA_1, DRAM0_BA_0 : BUFFER STD_LOGIC;
        DRAM0_CS_N, DRAM0_CAS_N, DRAM0_RAS_N, DRAM0_WE_N : OUT STD_LOGIC;
        DRAM_DQ : INOUT STD_LOGIC_VECTOR(15 DOWNTO 0);
        DRAM0_UDQM, DRAM0_LDQM : BUFFER STD_LOGIC );
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_0 : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
            zs_ba_from_the_sdram_0 : BUFFER 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(15 DOWNTO 0);
            zs_dqm_from_the_sdram_0 : BUFFER STD_LOGIC_VECTOR(1 DOWNTO 0);
            zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
            zs_we_n_from_the_sdram_0 : OUT STD_LOGIC );
    END COMPONENT;
    SIGNAL BA : STD_LOGIC_VECTOR(1 DOWNTO 0);
    SIGNAL DQM : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
    DRAM0_BA_1 <= BA(1);   DRAM0_BA_0 <= BA(0);
    DRAM0_UDQM <= DQM(1);   DRAM0_LDQM <= DQM(0);
-- Instantiate the Nios II system entity generated by the SOPC Builder.
    NiosII: nios_system PORT MAP (CLOCK_50, KEY(0), LEDG, SW,
        DRAM0_ADDR, BA, DRAM0_CAS_N, DRAM0_CKE, DRAM0_CS_N,
        DRAM_DQ, DQM, DRAM0_RAS_N, DRAM0_WE_N );
    DRAM0_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 II FPGA on the DE2-70 board. Use the application program from the tutorial *Introduction to the Altera SOPC Builder Using VHDL Design*, which is shown in Figure 9. Notice in our expanded system, the addresses assigned by the SOPC Builder are 0x04003000 for Switches and 0x04003010 for LEDs, which are different from the original system. These changes are already reflected in the program in Figure 9.

```
.include  "nios_macros.s"

.equ      Switches, 0x04003000
.equ      LEDs, 0x04003010

.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.

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 DE2-70 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 10. To solve the problem, it is necessary to modify the design as indicated in the next section.

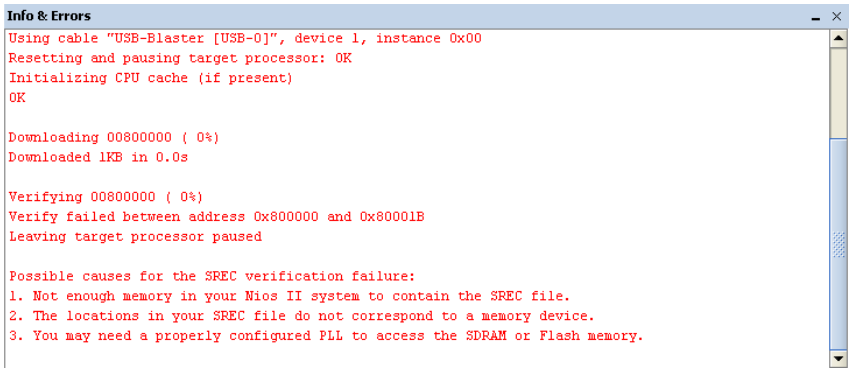

Figure 10. Error message in the Altera Monitor Program that may be due to the SDRAM clock skew problem.

5 Using a Phase-Locked Loop

The clock skew depends on physical characteristics of the DE2-70 board. For proper operation of the SDRAM chip, it is necessary that its clock signal, *DRAM0_CLK*, leads the Nios II system clock, *CLOCK_50*, by 3 nanosec-

onds. 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.

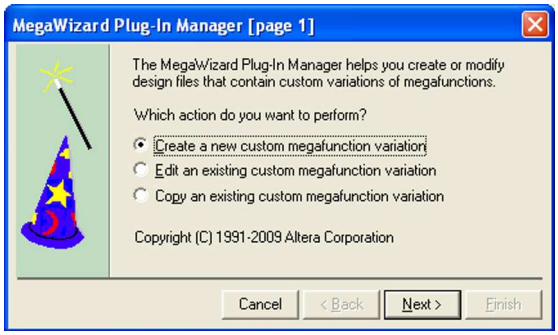

Figure 11. The MegaWizard.

2. In the window in Figure 12, specify that Cyclone II is the device family used and that the circuit should be defined in VHDL. Also, specify that the generated output (VHDL) file should be called *sdram_pll.vhd*. From the list of megafunctions in the left box select I/O > ALTPLL. Click Next.

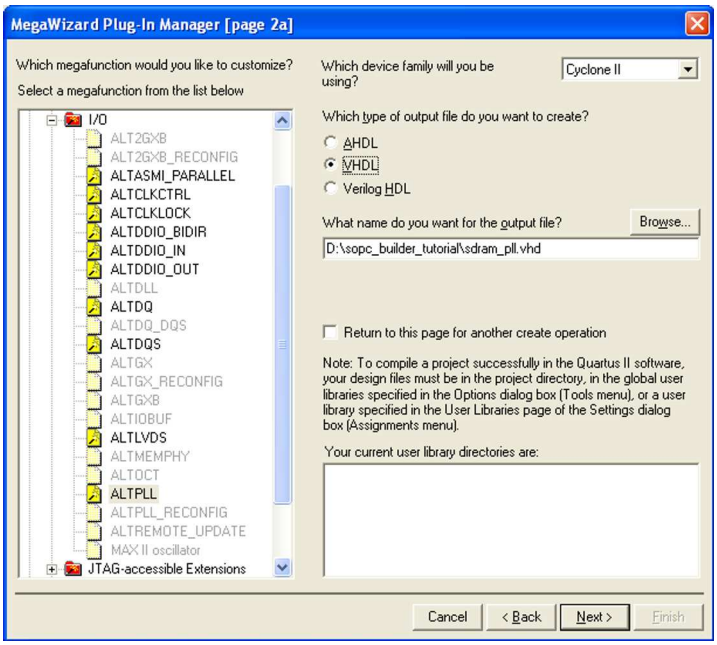

Figure 12. Select the megafunction and name the output file.

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.

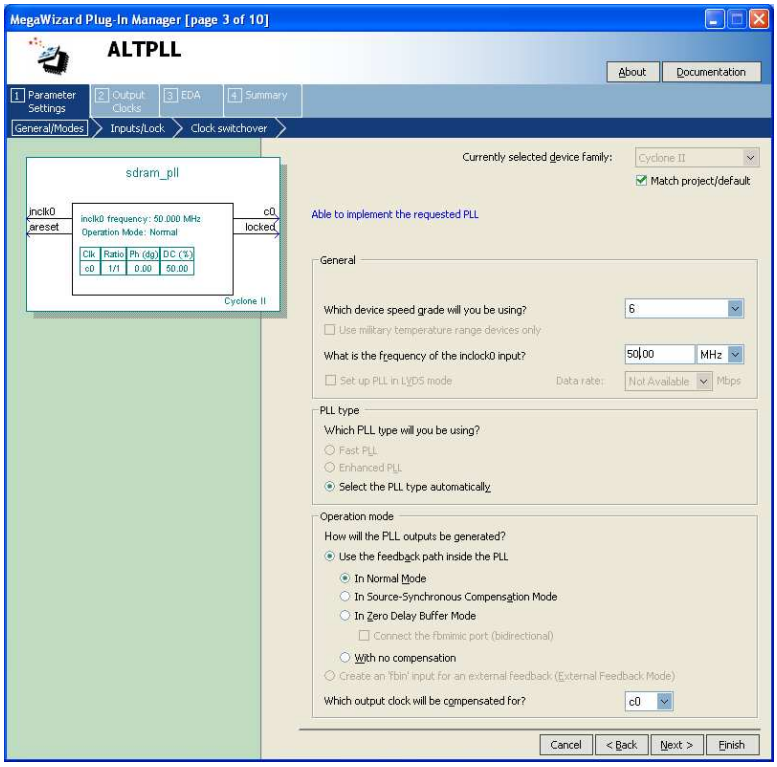

Figure 13. Define the clock frequency.

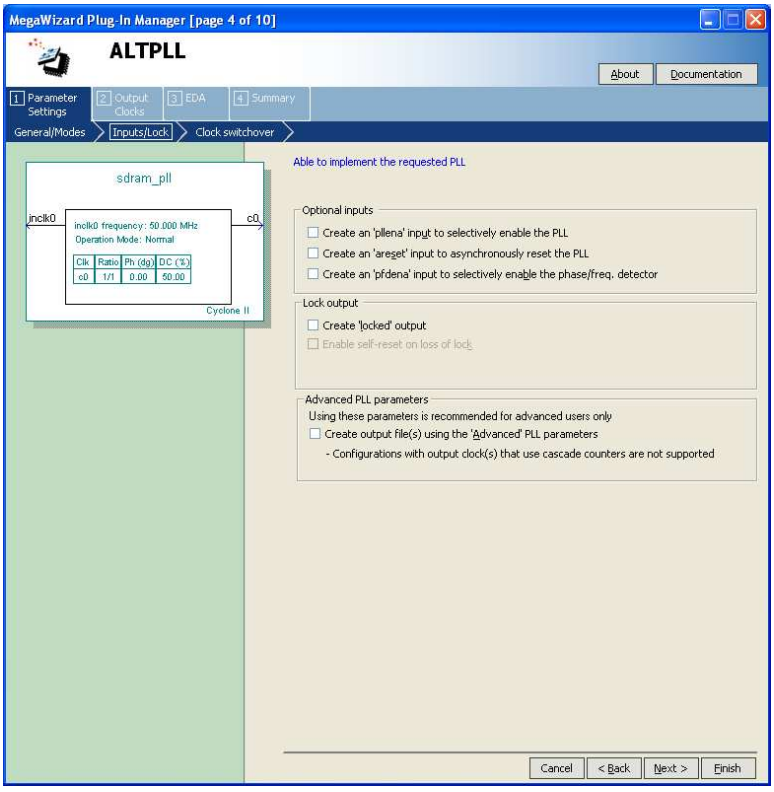

Figure 14. Remove unnecessary signals.

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.

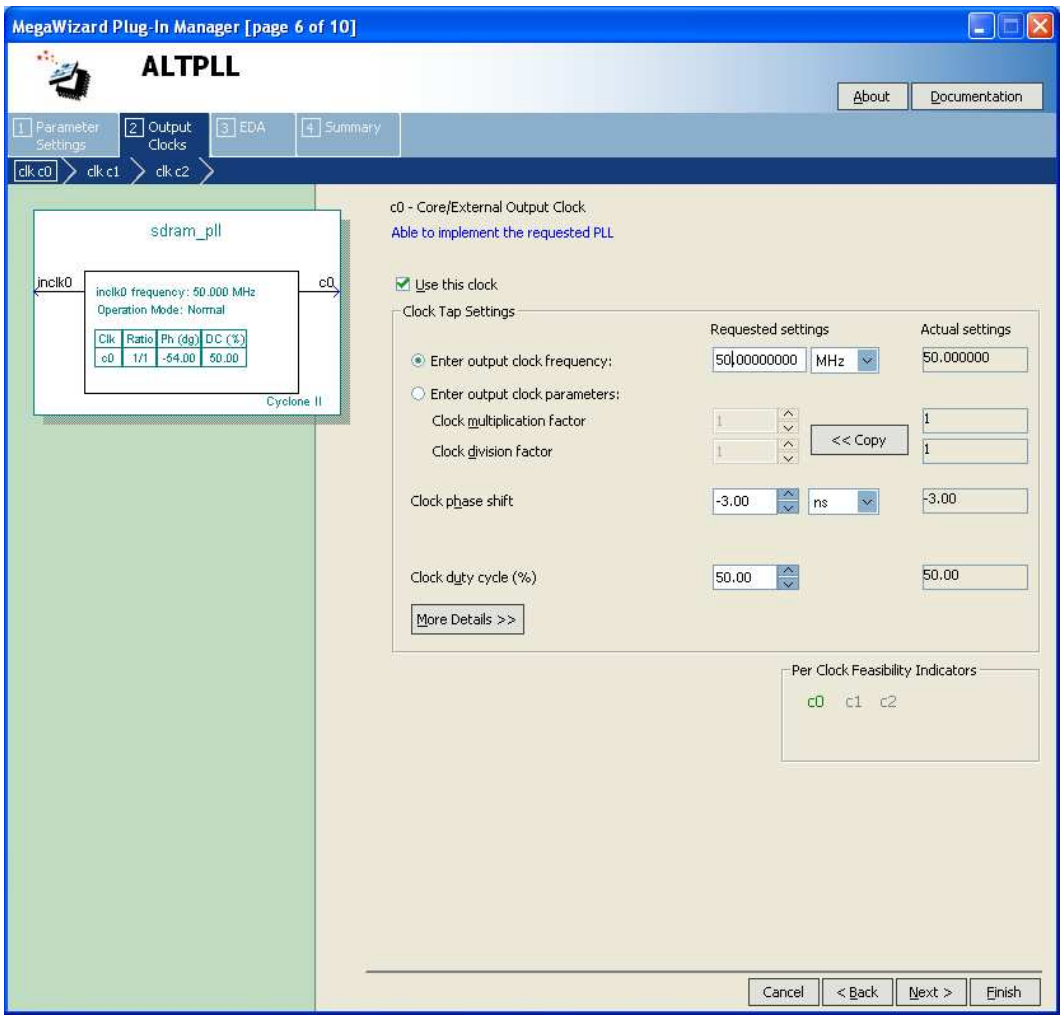

Figure 15. Specify the phase shift.

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 Next to reach the window in figure 16.

6. In order to ensure that the phase shift is exactly -3 ns, we will drive the original (non-shifted) clock through the PLL as well, but without any modifications. It will be called *c1*. Select Use this clock and specify that the output clock frequency is 50 MHz. Leave all the other settings unchanged and click Finish, which advances to page 10.

7. In the summary window in Figure 17 click Finish to complete the process. At this point, a box may pop up asking you to add the newly generated files to the project. In this case, select No, since this is not needed when using VHDL.

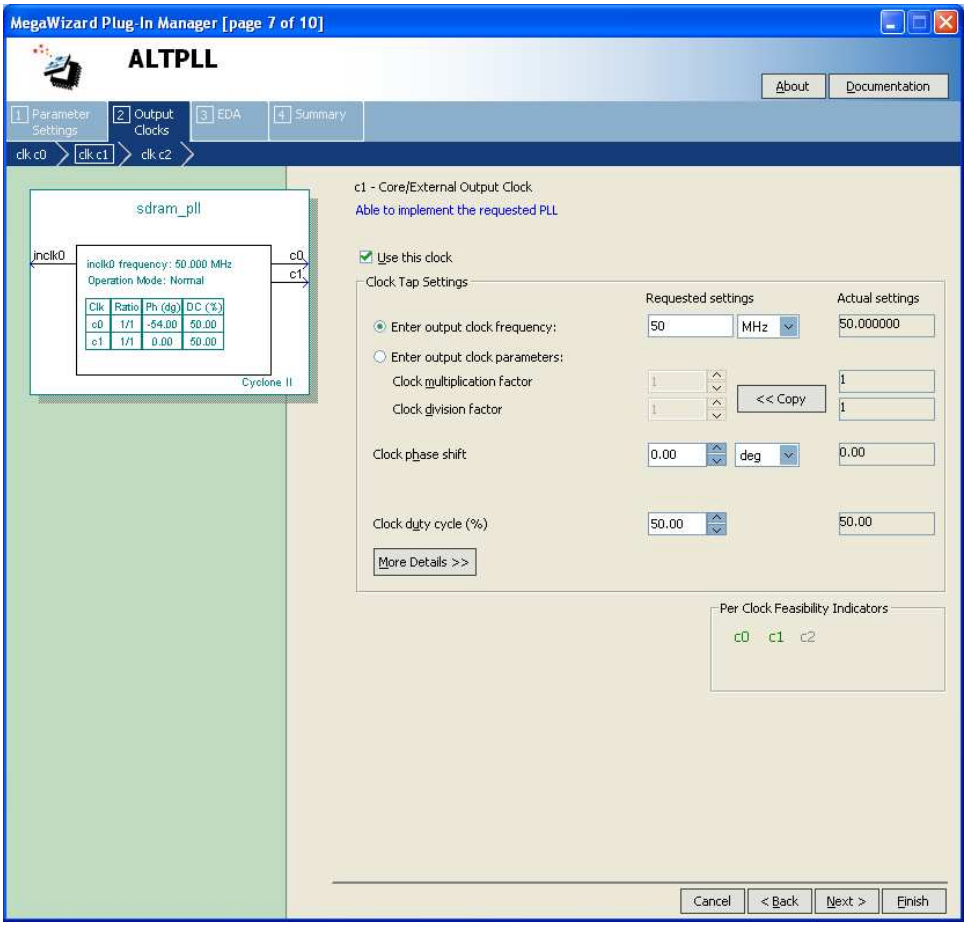

Figure 16. Drive the original clock signal through the PLL.

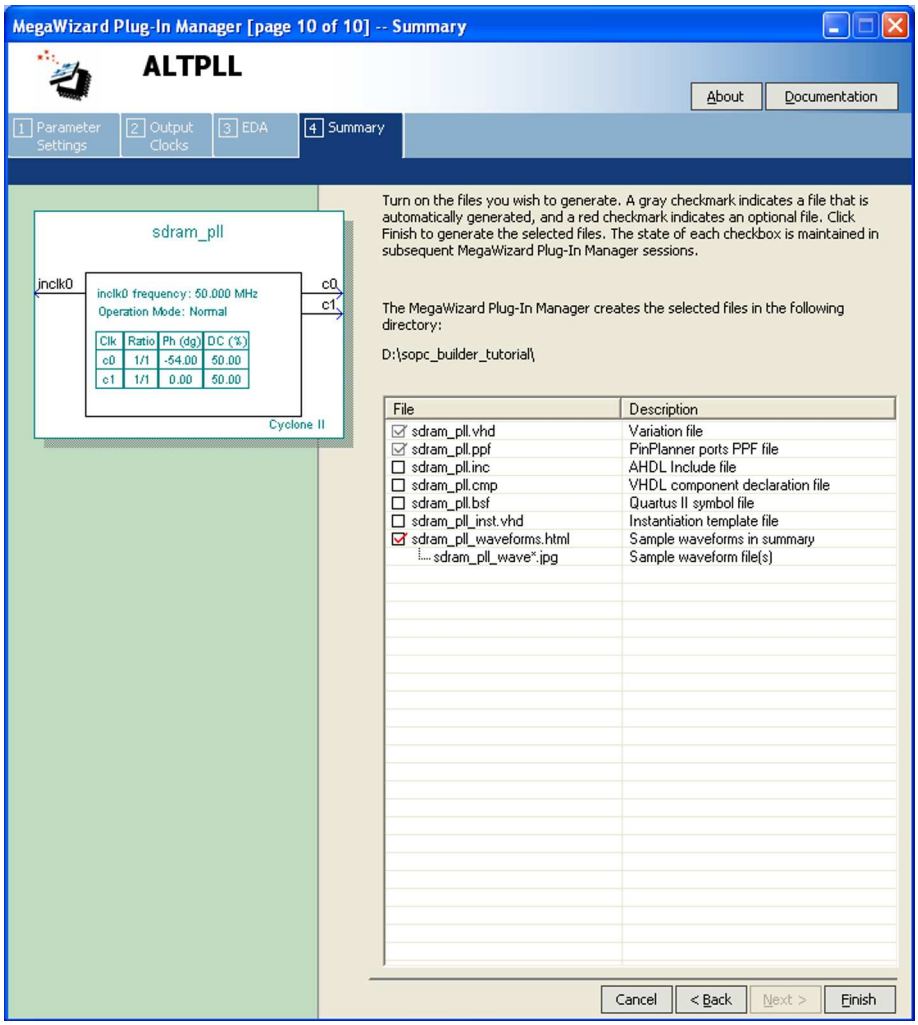

Figure 17. The summary page.

The desired PLL circuit is now defined as a VHDL entity in the file *sdram_pll.vhd*, which is placed in the project directory. Figure 18 shows the entity ports, consisting of signals *inclk0*, *c0*, and *c1*.

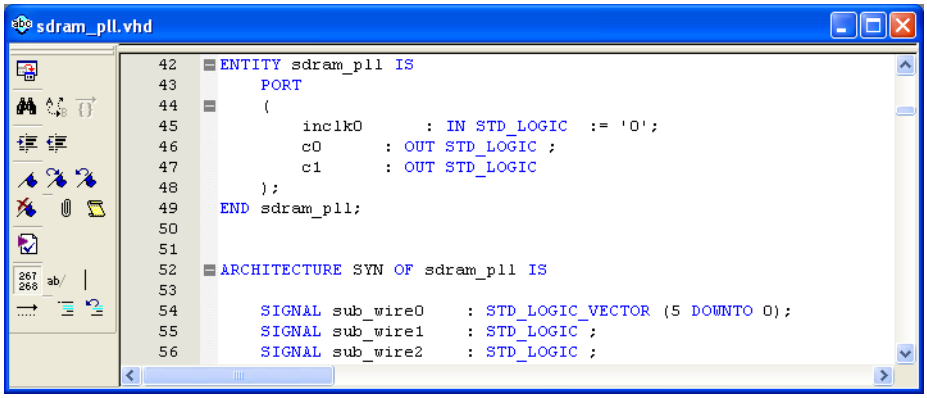

Figure 18. The generated PLL entity.

Next, we have to fix the top-level VHDL entity, given in Figure 8, to include the PLL circuit. The desired code is shown in Figure 19. The PLL circuit connects the shifted clock output *c0* to the pin *DRAM0_CLK*, and the unmodified clock signal *c1* to the clock signal required to drive the Nios II system.

```
-- Implements a simple Nios II system for the DE2-70 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 DE2-70 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)
        DRAM0_CLK, DRAM0_CKE : OUT STD_LOGIC;
        DRAM0_ADDR : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
        DRAM0_BA_1, DRAM0_BA_0 : BUFFER STD_LOGIC;
        DRAM0_CS_N, DRAM0_CAS_N, DRAM0_RAS_N, DRAM0_WE_N : OUT STD_LOGIC;
        DRAM_DQ : INOUT STD_LOGIC_VECTOR(15 DOWNTO 0);
        DRAM0_UDQM, DRAM0_LDQM : BUFFER STD_LOGIC );
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_0 : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
            zs_ba_from_the_sdram_0 : BUFFER 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(15 DOWNTO 0);
            zs_dqm_from_the_sdram_0 : BUFFER STD_LOGIC_VECTOR(1 DOWNTO 0);
            zs_ras_n_from_the_sdram_0 : OUT STD_LOGIC;
            zs_we_n_from_the_sdram_0 : OUT STD_LOGIC );
    END COMPONENT;

...continued in Part b
```

Figure 19. Proper instantiation of the expanded Nios II system (Part a).

```
    COMPONENT sdram_pll
        PORT ( inclk0 : IN STD_LOGIC;
               c0 : OUT STD_LOGIC;
               c1 : OUT STD_LOGIC );
    END COMPONENT;

    SIGNAL BA : STD_LOGIC_VECTOR(1 DOWNTO 0);
    SIGNAL DQM : STD_LOGIC_VECTOR(1 DOWNTO 0);

-- This signal is used to connect the unmodified clock signal c1 from the PLL to the
-- NIOS II system
    SIGNAL pll_c1 : STD_LOGIC;
BEGIN
    DRAM0_BA_1 <= BA(1);
    DRAM0_BA_0 <= BA(0);
    DRAM0_UDQM <= DQM(1);
    DRAM0_LDQM <= DQM(0);

-- Instantiate the Nios II system entity generated by the SOPC Builder.
    NiosII: nios_system PORT MAP (pll_c1, KEY(0), LEDG, SW,
        DRAM0_ADDR, BA, DRAM0_CAS_N, DRAM0_CKE, DRAM0_CS_N,
        DRAM_DQ, DQM, DRAM0_RAS_N, DRAM0_WE_N );

-- Instantiate the entity sdram_pll (inclk0, c0, c1).
    neg_3ns: sdram_pll PORT MAP (CLOCK_50, DRAM0_CLK, pll_c1);

END Structure;
```

Figure 19. Proper instantiation of the expanded Nios II system (Part b).

Compile the code and download the design into the Cyclone II FPGA on the DE2-70 board. Use the application program in Figure 9 to test the circuit.