Using the SDRAM on Altera's DE2-70 Board with Verilog Designs

For Quartus II 13.0

1 Introduction

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 Qsys tool. 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 Qsys System Integration Tool*.

The screen captures in the tutorial were obtained using the Quartus® II version 13.0; 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 Qsys tool 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 Qsys System Integration Tool* 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 SDRAM chips require careful timing control. To provide access to the SDRAM chips, the Qsys tool 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 Qsys System Integration Tool* tutorial. Figure 1 gives the block diagram of our example system.

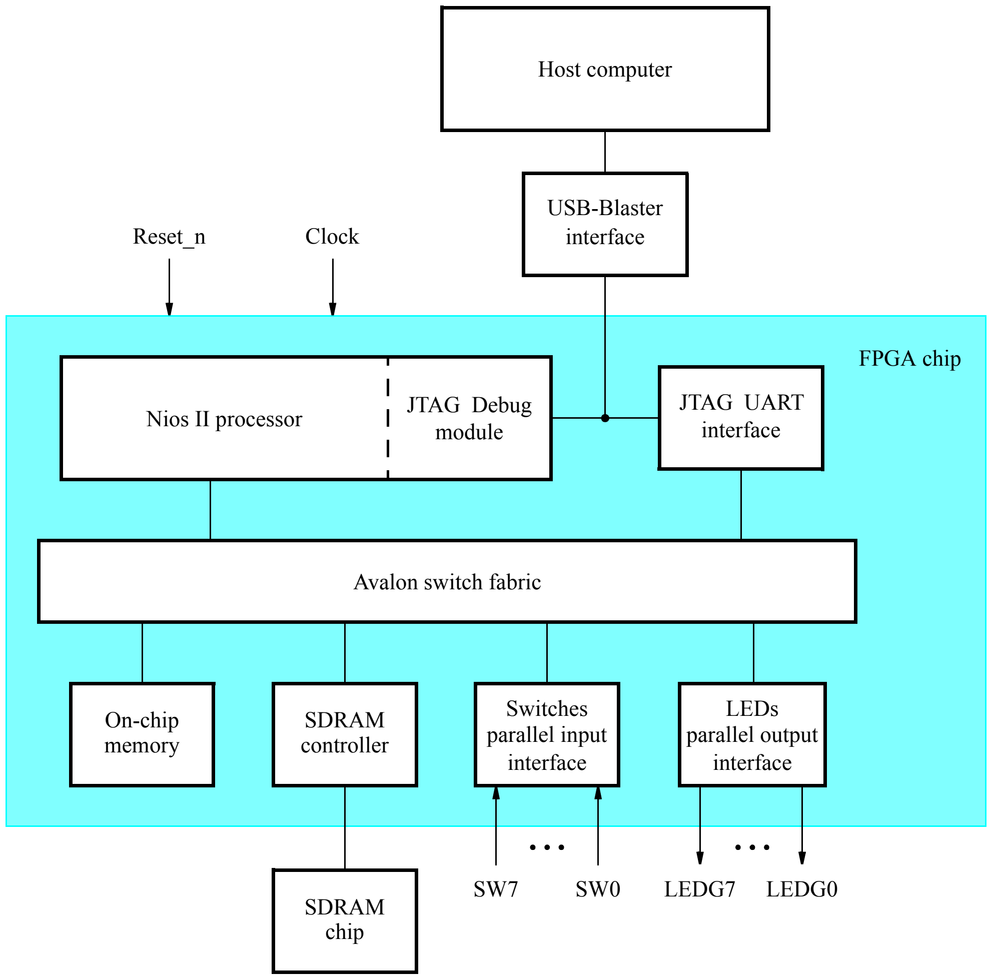

Figure 1. Example Nios II system implemented on the DE2-70 board.

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 Qsys tool 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 DE2-70 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 Qsys tool to include an SDRAM interface for a Nios II-based system
- Timing issues with respect to the SDRAM on the DE2-70 board

4 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 Qsys tool. 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.

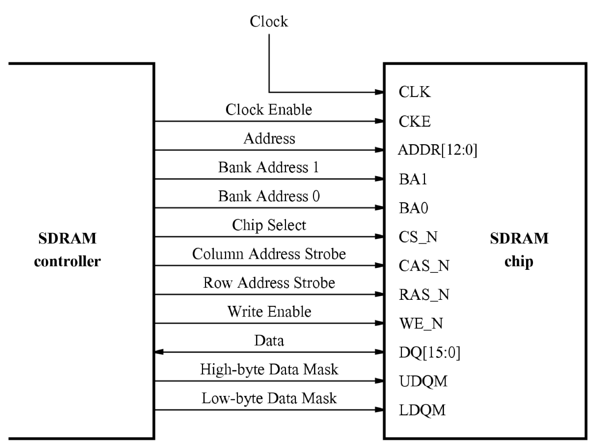

Figure 2. The SDRAM signals.

5 Using the Qsys tool to Generate the Nios II System

Our starting point will be the Nios II system discussed in the *Introduction to the Altera Qsys System Integration Tool* tutorial, which we implemented in a project called *lights*. We specified the system shown in Figure 3.

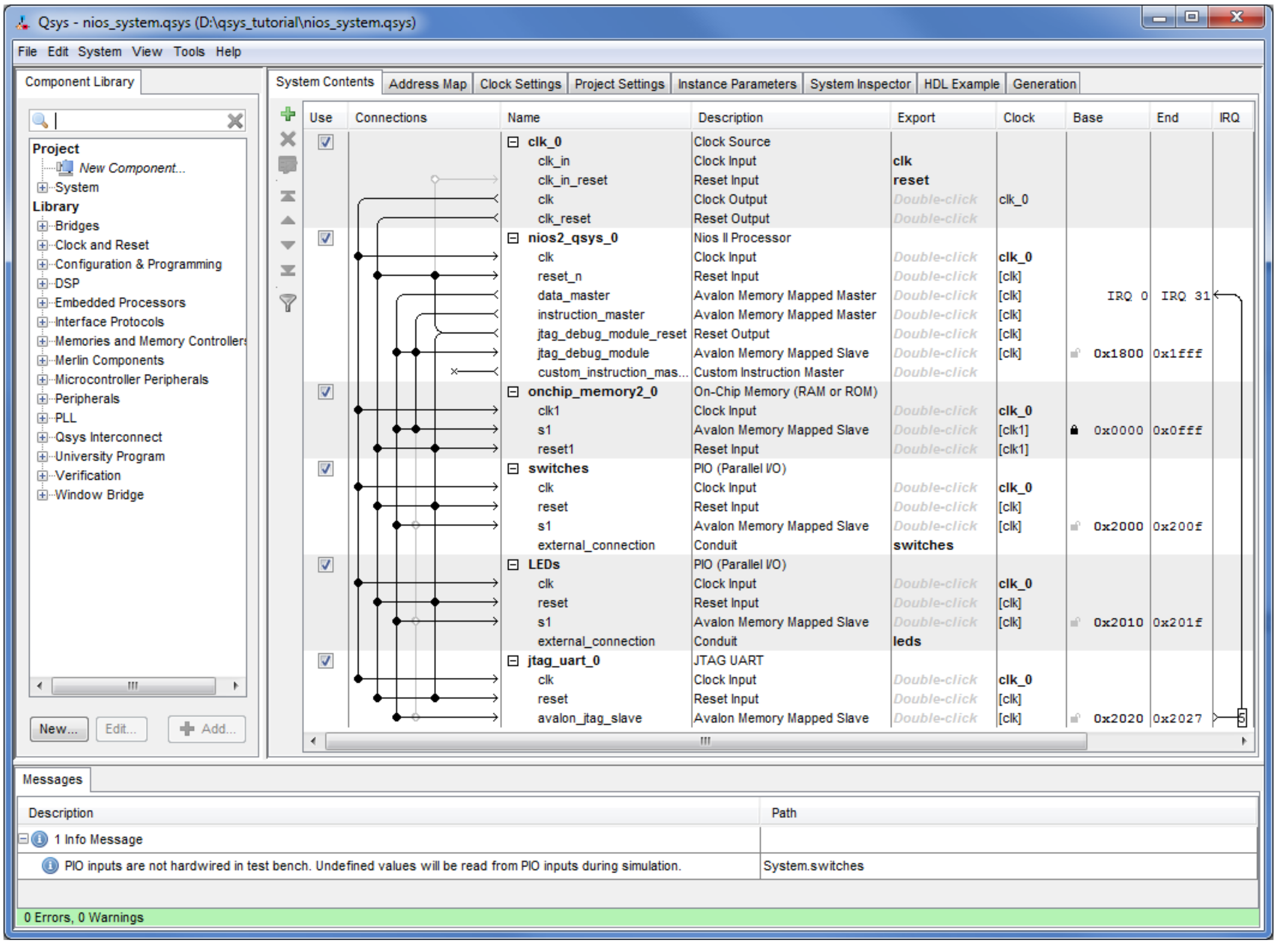

Figure 3. The Nios II system defined in the introductory tutorial.

If you saved the *lights* project, then open this project in the Quartus II software and then open the Qsys tool. 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 > External Memory Interfaces > SDRAM Interfaces > SDRAM Controller and click Add. A window depicted in Figure 4 appears. Set the Data Width parameter to 32 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. 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** module added to the design. Rename this module to *sdram* and connect this module to the system as shown in Figure 6. Then, double click on Double-click (in the Export column of the System Contents tab) for wire of the **sdram** to export the signal.

Select the command System > Assign Base Addresses to produce the assignment shown in Figure 6. Observe that the Qsys tool 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 nios2_processor and then select Edit to reach the window in Figure 7. Select sdram to be the memory device for both reset vector and exception vector, as shown in the figure. Click Finish to return to the System Contents tab and regenerate the system.

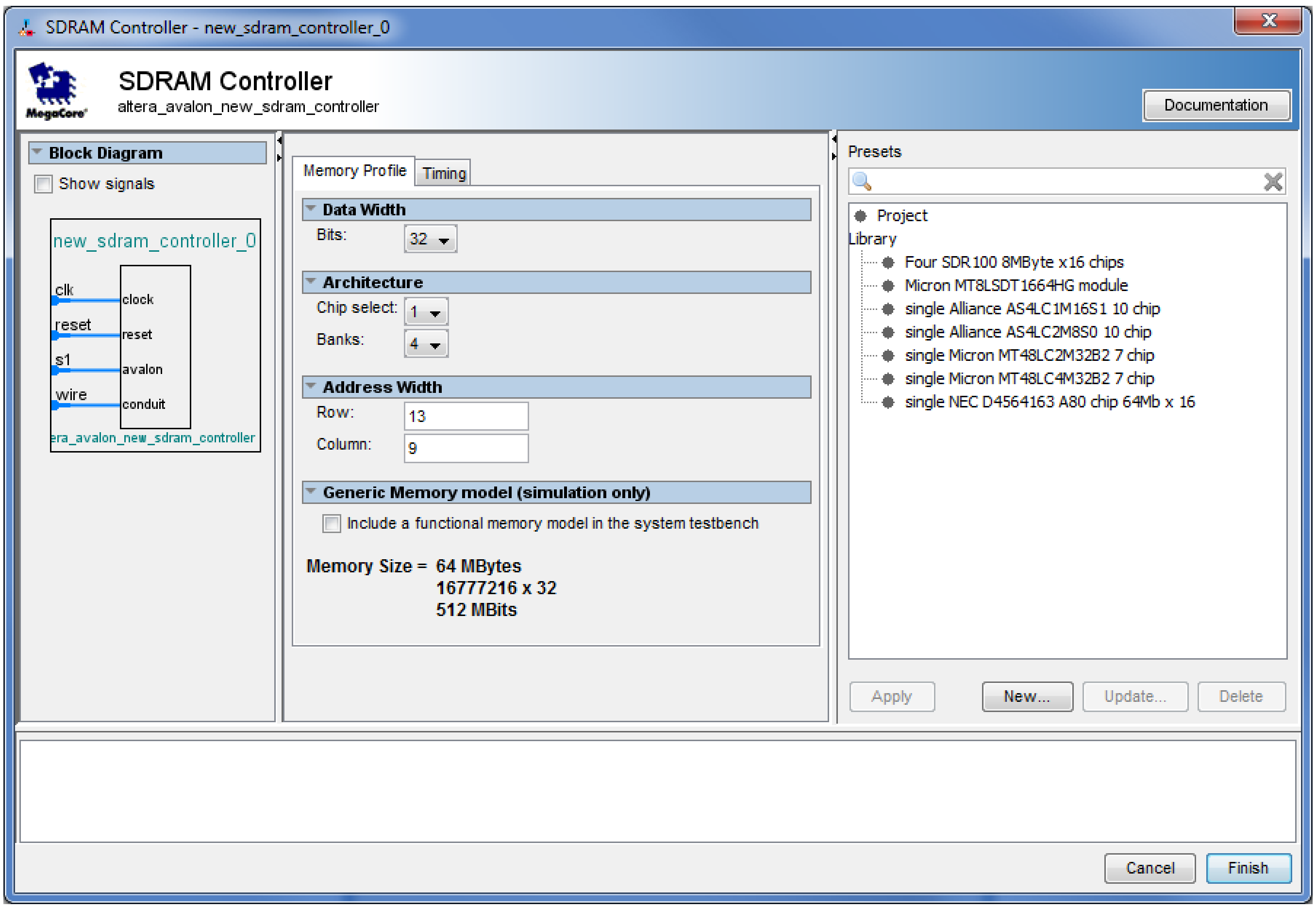

Figure 4. Add the SDRAM Controller.

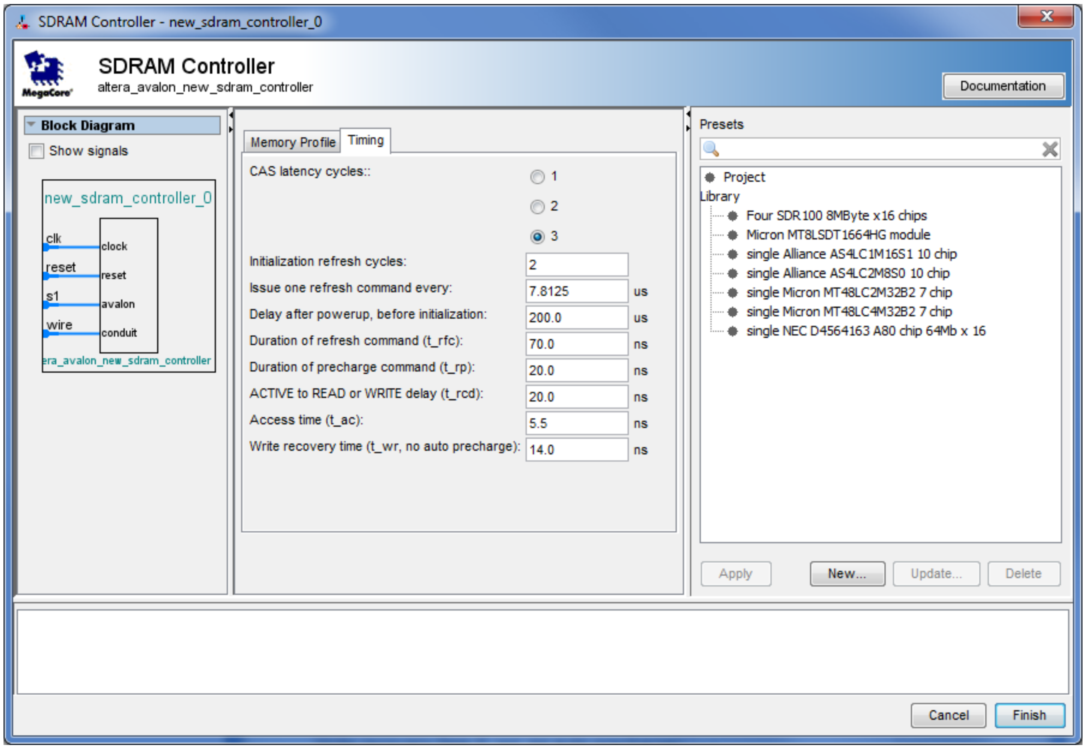

Figure 5. SDRAM Timings

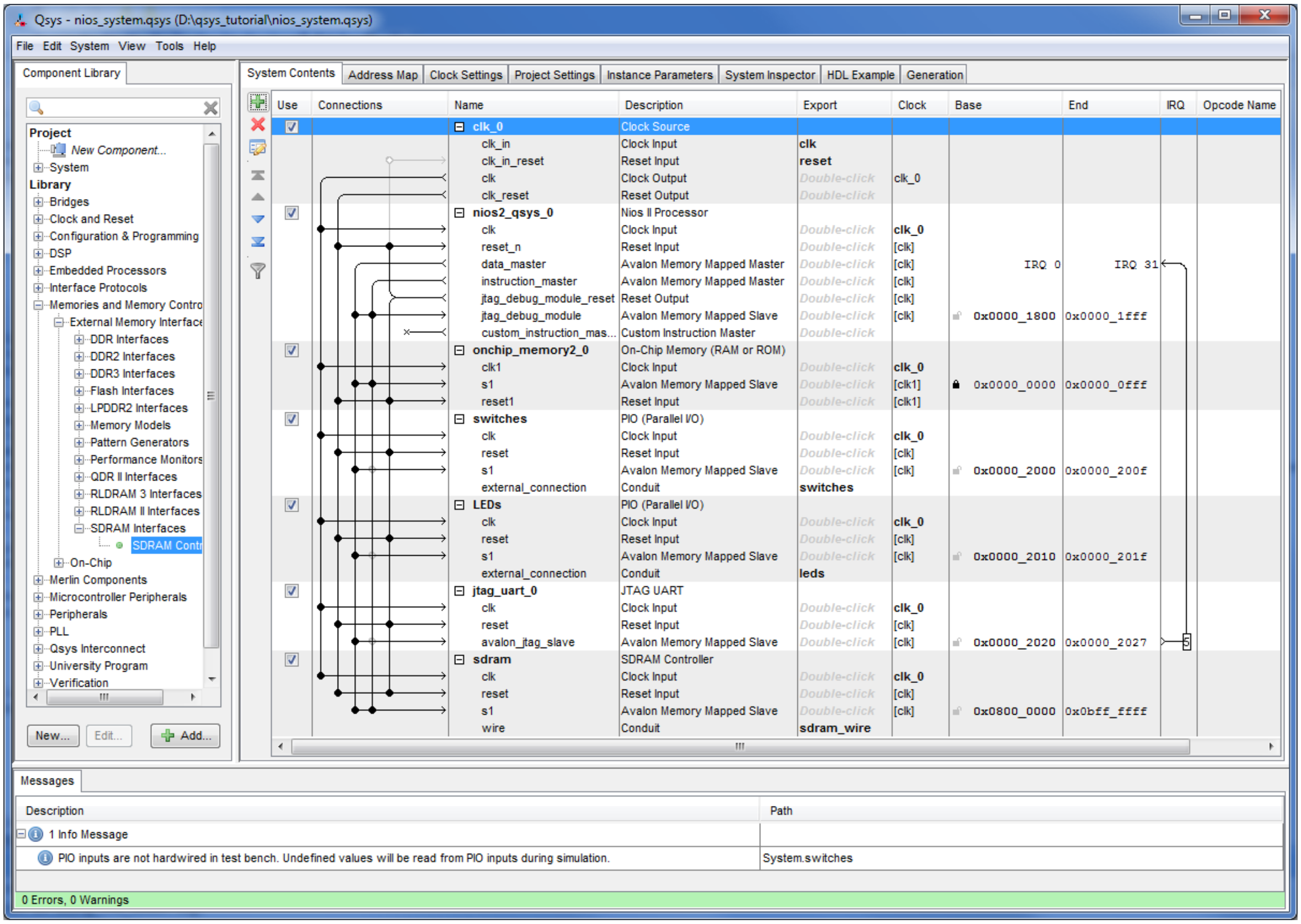

Figure 6. The expanded Nios II system.

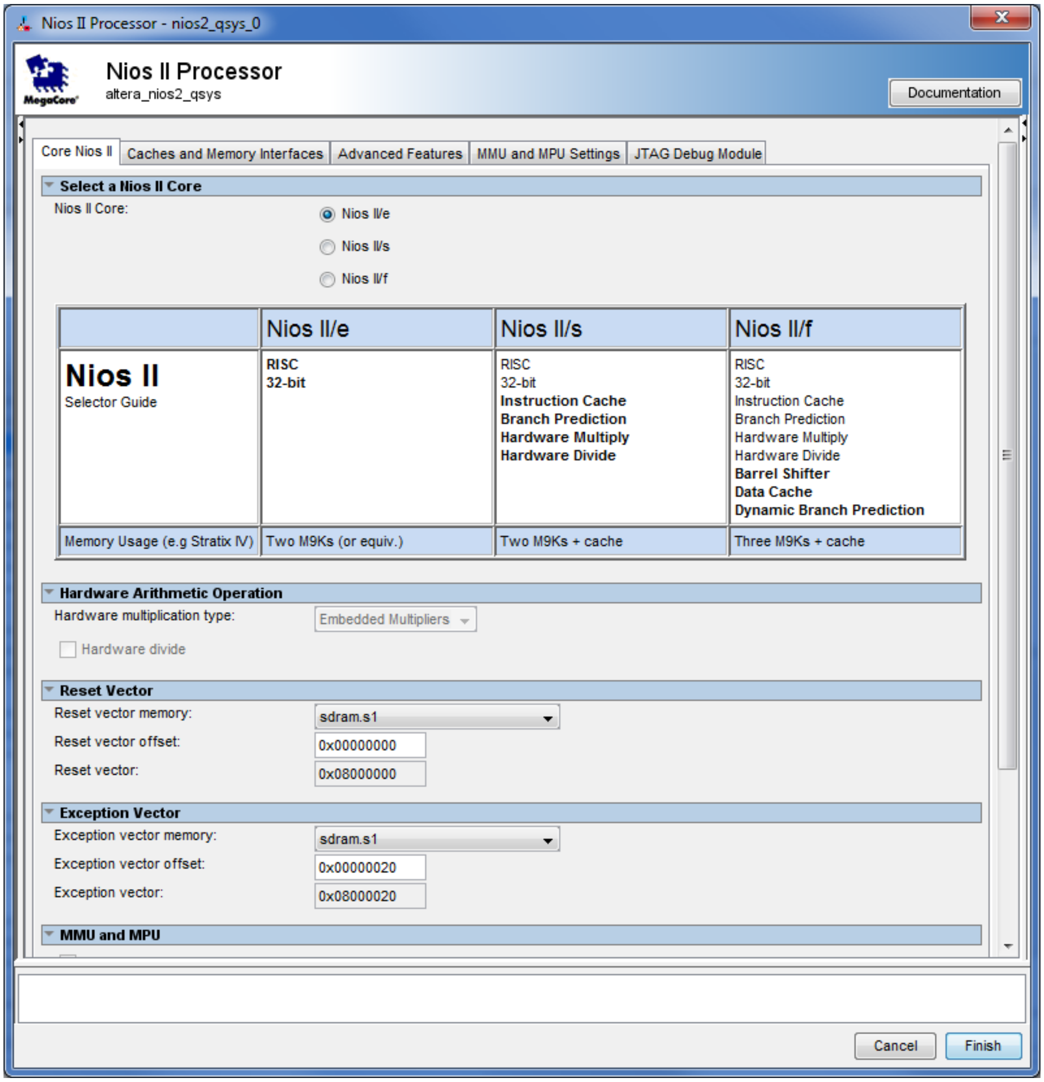

Figure 7. Define the reset vector and the exception vector.

The augmented Verilog module generated by the Qsys tool is in the file *nios_system.v* in the nios_system\synthesis directory of the project. Figure 8 depicts the portion of the code that defines the input and output signals for the module *nios_system*. As in our initial system that we developed in the introductory tutorial, the 8-bit vector that is the input to the parallel port *Switches* is called *switches_export*. The 8-bit output vector is called *leds_export*. The clock and reset signals are called *clk_clk* and *reset_reset_n*, respectively. A new module, called *sdram*, is included. It involves the signals indicated in Figure 2. For example, the address lines are referred to as the **output** vector *sdram_wire_addr[12:0]*. The data lines are referred to as the **inout** vector *sdram_wire_dq[31:0]*. This is a vector of the **inout** type because the data lines are bidirectional.

```
module nios_system (
      input  wire        clk_clk,           //       clk.clk
      input  wire        reset_reset_n,     //     reset.reset_n
      output wire [7:0]  leds_export,       //      leds.export
      input  wire [7:0]  switches_export,   //  switches.export
      output wire [12:0] sdram_wire_addr,   // sdram_wire.addr
      output wire [1:0]  sdram_wire_ba,     //           .ba
      output wire        sdram_wire_cas_n,  //           .cas_n
      output wire        sdram_wire_cke,    //           .cke
      output wire        sdram_wire_cs_n,   //           .cs_n
      inout  wire [31:0] sdram_wire_dq,     //           .dq
      output wire [3:0]  sdram_wire_dqm,    //           .dqm
      output wire        sdram_wire_ras_n,  //           .ras_n
      output wire        sdram_wire_we_n    //           .we_n
  );
```

Figure 8. A part of the generated Verilog module.

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 Verilog module, as we have done in the tutorial *Introduction to the Altera Qsys System Integration Tool*. The module is named *lights*, because this is the name of the top-level design entity in our Quartus II project.

A first attempt at creating the new module is presented in Figure 9. The input and output ports of the module use the pin names for the 50-MHz clock, *CLOCK_50*, pushbutton switches, *KEY*, toggle switches, *SW*, and green LEDs, *LEDG*, as used in our original design. They also use the pin names *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. A similar set of signals are used for the other SDRAM chip. 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.qsf* in the directory *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 page.

Observe that the two *Bank Address* signals are treated by the Qsys tool as a two-bit vector called *sdram_wire_ba[1:0]*, as seen in Figure 8. However, in the *DE2_70_pin_assignments.qsf* file these signals are given as scalars *DRAM0_BA_1* and *DRAM0_BA_0*. Therefore, in our Verilog module, we concatenated these signals as *{DRAM0_BA_1, DRAM0_BA_0}*. Similarly, the vector *dram_wire_dqm[1:0]* corresponds to *{DRAM0_UDQM, DRAM0_LDQM}*.

Finally, note that we tried an obvious approach of using the 50-MHz system clock, *CLOCK_50*, as the clock signal for the SDRAM chips. This is specified by the **assign** statement in the code. This approach leads to a potential timing problem caused by the clock skew on the DE2-70 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 DE2-70 User Manual.
module lights (CLOCK_50, KEY, SW, DRAM_DQ, LEDG, DRAM0_ADDR,
    DRAM0_BA_1, DRAM0_BA_0, DRAM0_CAS_N, DRAM0_RAS_N, DRAM0_CLK,
    DRAM0_CKE, DRAM0_CS_N, DRAM0_WE_N, DRAM0_UDQM, DRAM0_LDQM,
    DRAM1_ADDR, DRAM1_BA_1, DRAM1_BA_0, DRAM1_CAS_N, DRAM1_RAS_N,
    DRAM1_CLK, DRAM1_CKE, DRAM1_CS_N, DRAM1_WE_N, DRAM1_UDQM, DRAM1_LDQM);
    input [7:0] SW;
    input [0:0] KEY;
    input CLOCK_50;
    output [7:0] LEDG;
    output [12:0] DRAM0_ADDR;
    output DRAM0_BA_1, DRAM0_BA_0, DRAM0_CAS_N;
    output DRAM0_RAS_N, DRAM0_CLK, DRAM0_CKE, DRAM0_CS_N;
    output DRAM0_WE_N, DRAM0_UDQM, DRAM0_LDQM;
    output [12: 0] DRAM1_ADDR;
    output DRAM1_BA_1, DRAM1_BA_0, DRAM1_CAS_N, DRAM1_RAS_N, DRAM1_CLK;
    output DRAM1_CKE, DRAM1_CS_N, DRAM1_WE_N, DRAM1_UDQM, DRAM1_LDQM;
    inout [31:0] DRAM_DQ;
    wire [13:0] DRAM_ADDR;
    wire DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N;
    wire DRAM_CLK, DRAM_CKE, DRAM_CS_N, DRAM_WE_N;
```

... continued in Part *b*

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

```
    wire [3:0] DRAM_DQM;
    assign DRAM0_ADDR = DRAM_ADDR;
    assign DRAM0_BA_1 = DRAM_BA_1;
    assign DRAM0_BA_0 = DRAM_BA_0;
    assign DRAM0_CAS_N = DRAM_CAS_N;
    assign DRAM0_CKE = DRAM_CKE;
    assign DRAM0_CLK = DRAM_CLK;
    assign DRAM0_CS_N = DRAM_CS_N;
    assign DRAM0_RAS_N = DRAM_RAS_N;
    assign DRAM0_WE_N = DRAM_WE_N;
    assign DRAM0_UDQM = DRAM_DQM[1];
    assign DRAM0_LDQM = DRAM_DQM[0];
    assign DRAM1_ADDR = DRAM_ADDR;
    assign DRAM1_BA_1 = DRAM_BA_1;
    assign DRAM1_BA_0 = DRAM_BA_0;
    assign DRAM1_CAS_N = DRAM_CAS_N;
    assign DRAM1_CKE = DRAM_CKE;
    assign DRAM1_CLK = DRAM_CLK;
    assign DRAM1_CS_N = DRAM_CS_N;
    assign DRAM1_RAS_N = DRAM_RAS_N;
    assign DRAM1_WE_N = DRAM_WE_N;
    assign DRAM1_UDQM = DRAM_DQM[3];
    assign DRAM1_LDQM = DRAM_DQM[2];
// Instantiate the Nios II system module generated by the Qsys tool
    nios_system NiosII (
        .clk_clk (CLOCK_50),
        .reset_reset_n (KEY[0]),
        .switches_export (SW),
        .leds_export (LEDG),
        .sdram_wire_addr (DRAM_ADDR),
        .sdram_wire_ba ({DRAM_BA_1, DRAM_BA_0}),
        .sdram_wire_cas_n (DRAM_CAS_N),
        .sdram_wire_cke (DRAM_CKE),
        .sdram_wire_cs_n (DRAM_CS_N),
        .sdram_wire_dq (DRAM_DQ),
        .sdram_wire_dqm (DRAM_DQM),
        .sdram_wire_ras_n (DRAM_RAS_N),
        .sdram_wire_we_n (DRAM_WE_N)
    );
    assign DRAM_CLK = CLOCK_50;
endmodule
```

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.v*. Add this file and the *nios_system.qip* file produced by the Qsys tool 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 Qsys System Integration Tool*, which is shown in Figure 10.

```
.include  "nios_macros.s"
.equ      Switches, 0x00002000
.equ      LEDs, 0x00002010
.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 DE2-70 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 11. To solve the problem, it is necessary to modify the design as indicated in the next section.

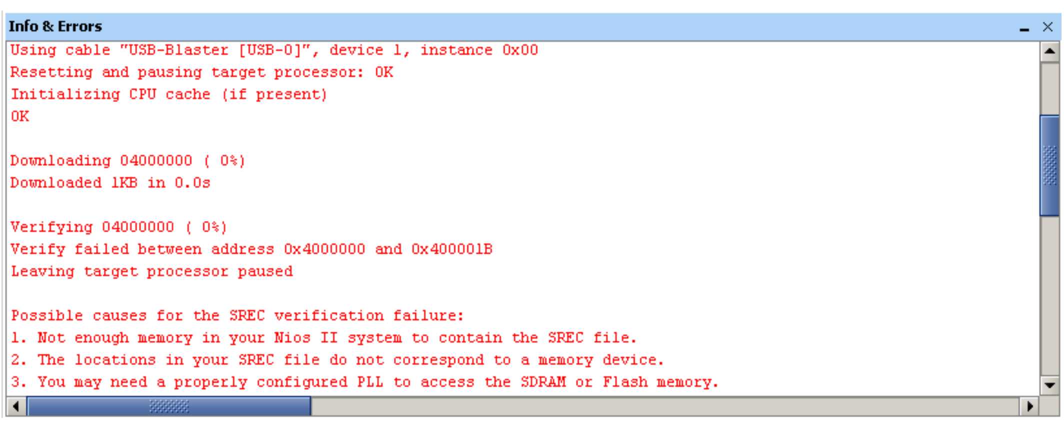

Figure 11. Error message in the Altera Monitor Program that may be due to the SDRAM clock skew problem.

7 Using the Clock Signals IP Core

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, *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 University Program. We will use the latter method in this tutorial.

To add the Clock Signals IP core, in the Qsys tool window of Figure 5 select University Program > Clocks Signals for DE-Series Board Peripherals and click Add. A window depicted in Figure 12 appears. Select *DE2-70* 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 5. Connect the clock and reset ouput of system clock *clk_0* to the clock and reset inputs of the Clock Signal IP core. All other IP cores (including the SDRAM) should be adjusted to use the *sys_clk* output of the Clock Signal core instead of the system clock. Rename the Clock Signal core to *clocks* and export the *sdram_clk* signal under the name *sdram_clk*. The final system is shown in Figure 13. Click on the System Generation tab and regenerate the system.

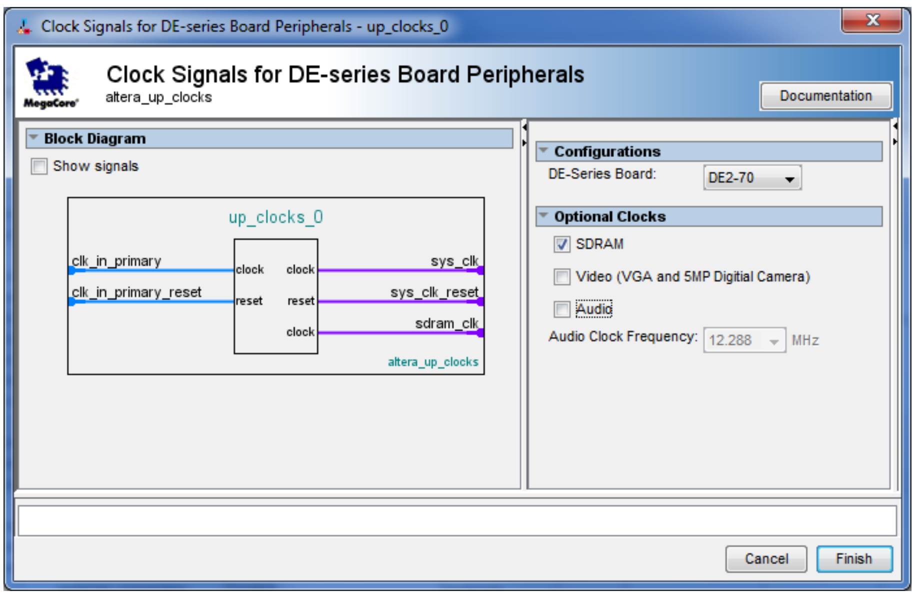

Figure 12. Clock Signals IP Core

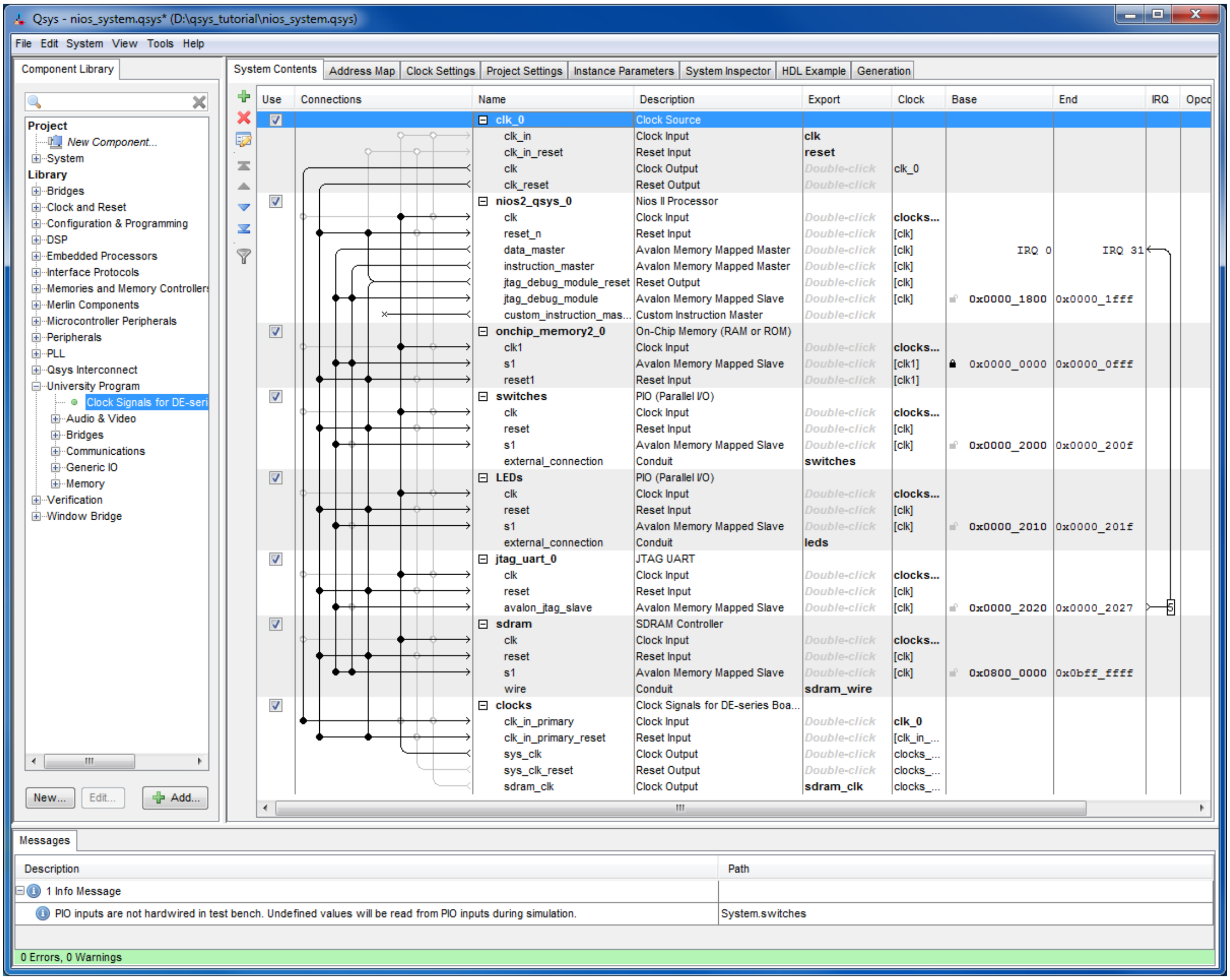

Figure 13. The final Nios II system.

Next, we have to fix the top-level Verilog module, given in Figure 8, to instantiate the Nios II system with the Clock Signals core included. The desired code is shown in Figure 14. The SDRAM clock signal *sdram_clk* generated by the Clock Signals core connects to the wire *DRAM_CLK*.

```
// 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.
module lights (CLOCK_50, KEY, SW, DRAM_DQ, LEDG, DRAM0_ADDR,
    DRAM0_BA_1, DRAM0_BA_0, DRAM0_CAS_N, DRAM0_RAS_N, DRAM0_CLK,
    DRAM0_CKE, DRAM0_CS_N, DRAM0_WE_N, DRAM0_UDQM, DRAM0_LDQM,
    DRAM1_ADDR, DRAM1_BA_1, DRAM1_BA_0, DRAM1_CAS_N, DRAM1_RAS_N,
    DRAM1_CLK, DRAM1_CKE, DRAM1_CS_N, DRAM1_WE_N, DRAM1_UDQM, DRAM1_LDQM);
    input [7:0] SW;
    input [0:0] KEY;
    input CLOCK_50;
    output [7:0] LEDG;
    output [12:0] DRAM0_ADDR;
    output DRAM0_BA_1, DRAM0_BA_0, DRAM0_CAS_N;
    output DRAM0_RAS_N, DRAM0_CLK, DRAM0_CKE, DRAM0_CS_N;
    output DRAM0_WE_N, DRAM0_UDQM, DRAM0_LDQM;
    output [12: 0] DRAM1_ADDR;
    output DRAM1_BA_1, DRAM1_BA_0, DRAM1_CAS_N, DRAM1_RAS_N, DRAM1_CLK;
    output DRAM1_CKE, DRAM1_CS_N, DRAM1_WE_N, DRAM1_UDQM, DRAM1_LDQM;
    inout [31:0] DRAM_DQ;
    wire [13:0] DRAM_ADDR;
    wire DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N;
    wire DRAM_CLK, DRAM_CKE, DRAM_CS_N, DRAM_WE_N;
    wire [3:0] DRAM_DQM;
    assign DRAM0_ADDR = DRAM_ADDR;
    assign DRAM0_BA_1 = DRAM_BA_1;
    assign DRAM0_BA_0 = DRAM_BA_0;
    assign DRAM0_CAS_N = DRAM_CAS_N;
    assign DRAM0_CKE = DRAM_CKE;
    assign DRAM0_CLK = DRAM_CLK;
    assign DRAM0_CS_N = DRAM_CS_N;
    assign DRAM0_RAS_N = DRAM_RAS_N;
    assign DRAM0_WE_N = DRAM_WE_N;
    assign DRAM0_UDQM = DRAM_DQM[1];
    assign DRAM0_LDQM = DRAM_DQM[0];
    assign DRAM1_ADDR = DRAM_ADDR;
    assign DRAM1_BA_1 = DRAM_BA_1;
    assign DRAM1_BA_0 = DRAM_BA_0;
```

... continued in Part *b*

Figure 14. Proper instantiation of the expanded Nios II system. (Part *a*)

```
        assign DRAM1_CAS_N = DRAM_CAS_N;
        assign DRAM1_CKE = DRAM_CKE;
        assign DRAM1_CLK = DRAM_CLK;
        assign DRAM1_CS_N = DRAM_CS_N;
        assign DRAM1_RAS_N = DRAM_RAS_N;
        assign DRAM1_WE_N = DRAM_WE_N;
        assign DRAM1_UDQM = DRAM_DQM[3];
        assign DRAM1_LDQM = DRAM_DQM[2];
// Instantiate the Nios II system module generated by the Qsys tool
        nios_system NiosII (
                .clk_clk (CLOCK_50),
                .reset_reset_n (KEY[0]),
                .switches_export (SW),
                .leds_export (LEDG),
                .sdram_wire_addr (DRAM_ADDR),
                .sdram_wire_ba ({DRAM_BA_1, DRAM_BA_0}),
                .sdram_wire_cas_n (DRAM_CAS_N),
                .sdram_wire_cke (DRAM_CKE),
                .sdram_wire_cs_n (DRAM_CS_N),
                .sdram_wire_dq (DRAM_DQ),
                .sdram_wire_dqm (DRAM_DQM),
                .sdram_wire_ras_n (DRAM_RAS_N),
                .sdram_wire_we_n (DRAM_WE_N),
                .sdram_clk_clk (DRAM_CLK)
        );
endmodule
```

Figure 14. 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.