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

*For Quartus II 12.1*

## 1 Introduction

This tutorial explains how the SDRAM chip on Altera's DE2 Development and Education board can be used with a Nios II system implemented by using the Altera Qsys integration tool. The discussion is based on the assumption that the reader has access to a DE2 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 12.1; 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 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 board contains an SDRAM chip that can store 8 Mbytes of data. This memory is organized as 1M x 16 bits x 4 banks. The SDRAM chip requires careful timing control. To provide access to the SDRAM chip, Qsys implements an *SDRAM Controller* circuit. This circuit generates the signals needed to deal with the SDRAM chip.

## 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.

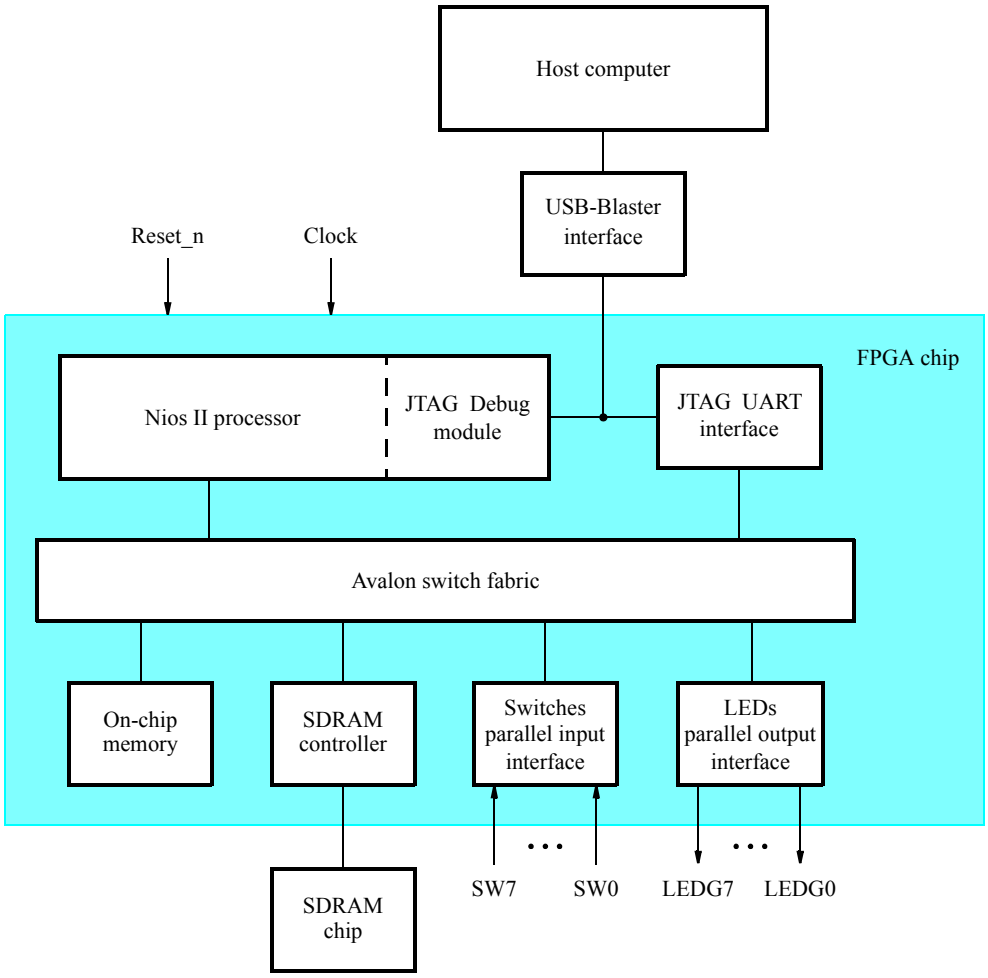


Figure 1. Example Nios II system implemented on the DE2 board.

The system realizes a trivial task. Eight toggle switches on the DE2 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 the SDRAM chip on the DE2 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 board

## 4 The SDRAM Interface

The SDRAM chip on the DE2 board has the capacity of 64 Mbits (8 Mbytes). It is organized as 1M x 16 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 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.

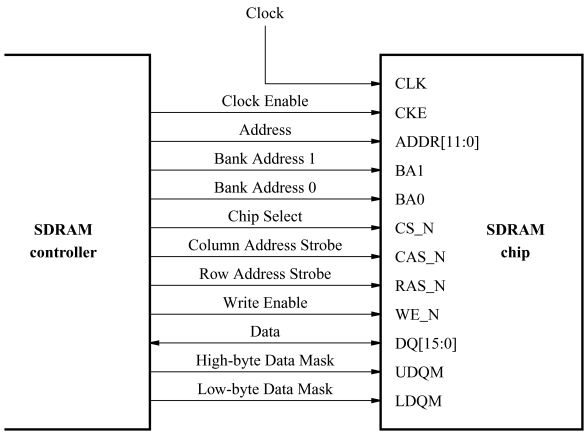


Figure 2. The SDRAM signals.

# 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.

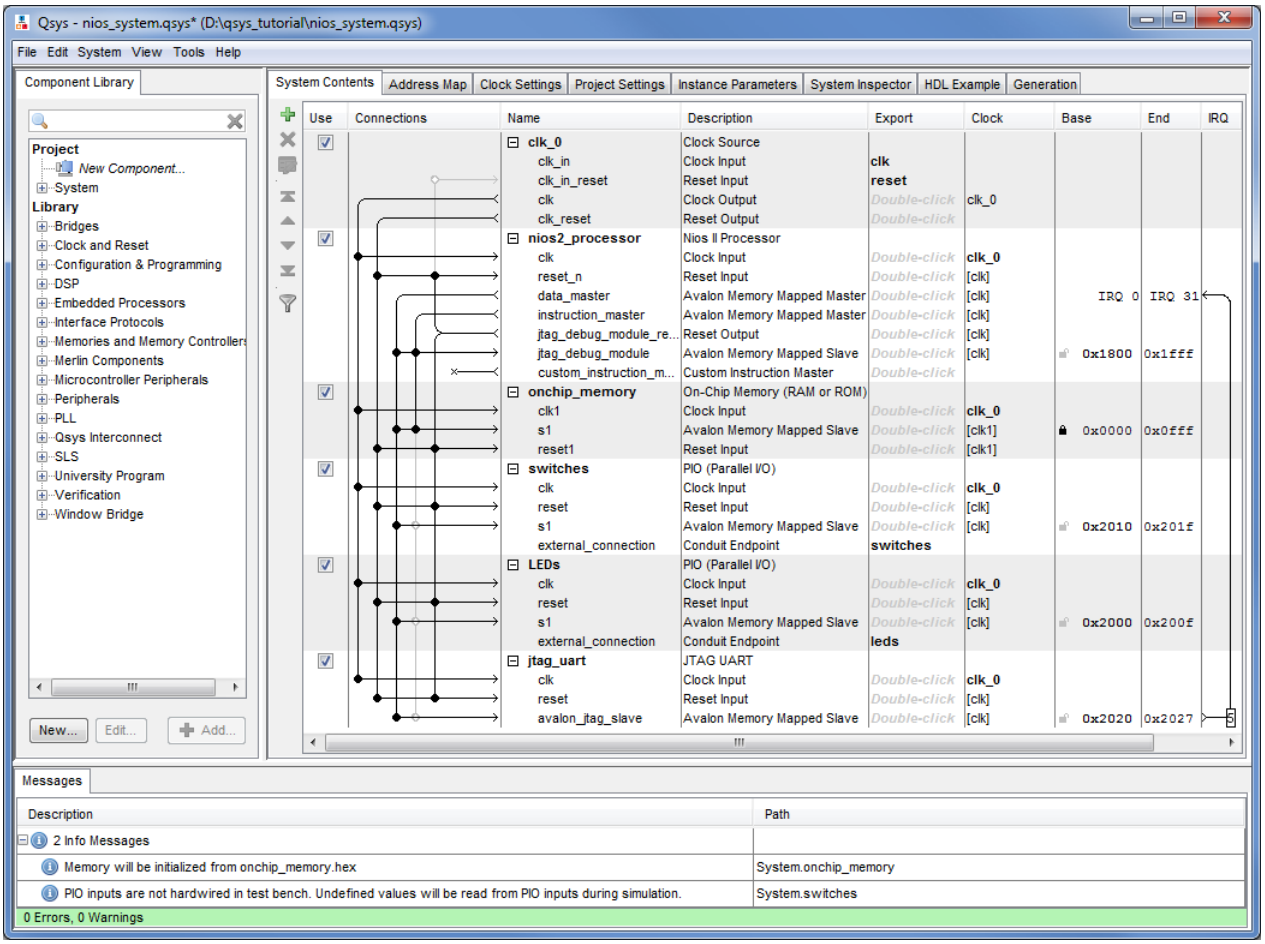


Figure 3. The Nios II system defined in the introductory tutorial.

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. Select *Custom* from the Presets drop-down list. 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* module added to the design. Rename this module to *sdram*. Connect the SDRAM to the rest of the system in the

same manner as the on-chip memory, and export the SDRAM wire port. Select the command System > Assign Base Addresses to produce the assignment shown in Figure 5. Observe that the Qsys tool assigned the base address 0x01000000 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 6. 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.

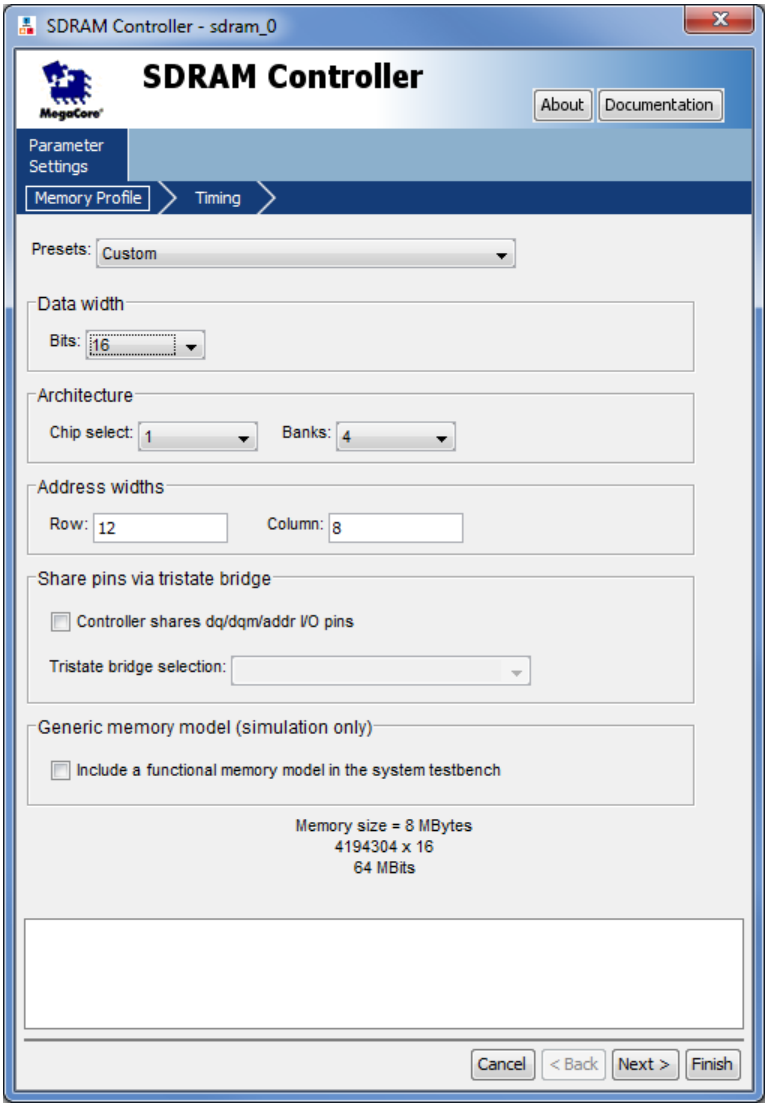


Figure 4. Add the SDRAM Controller.

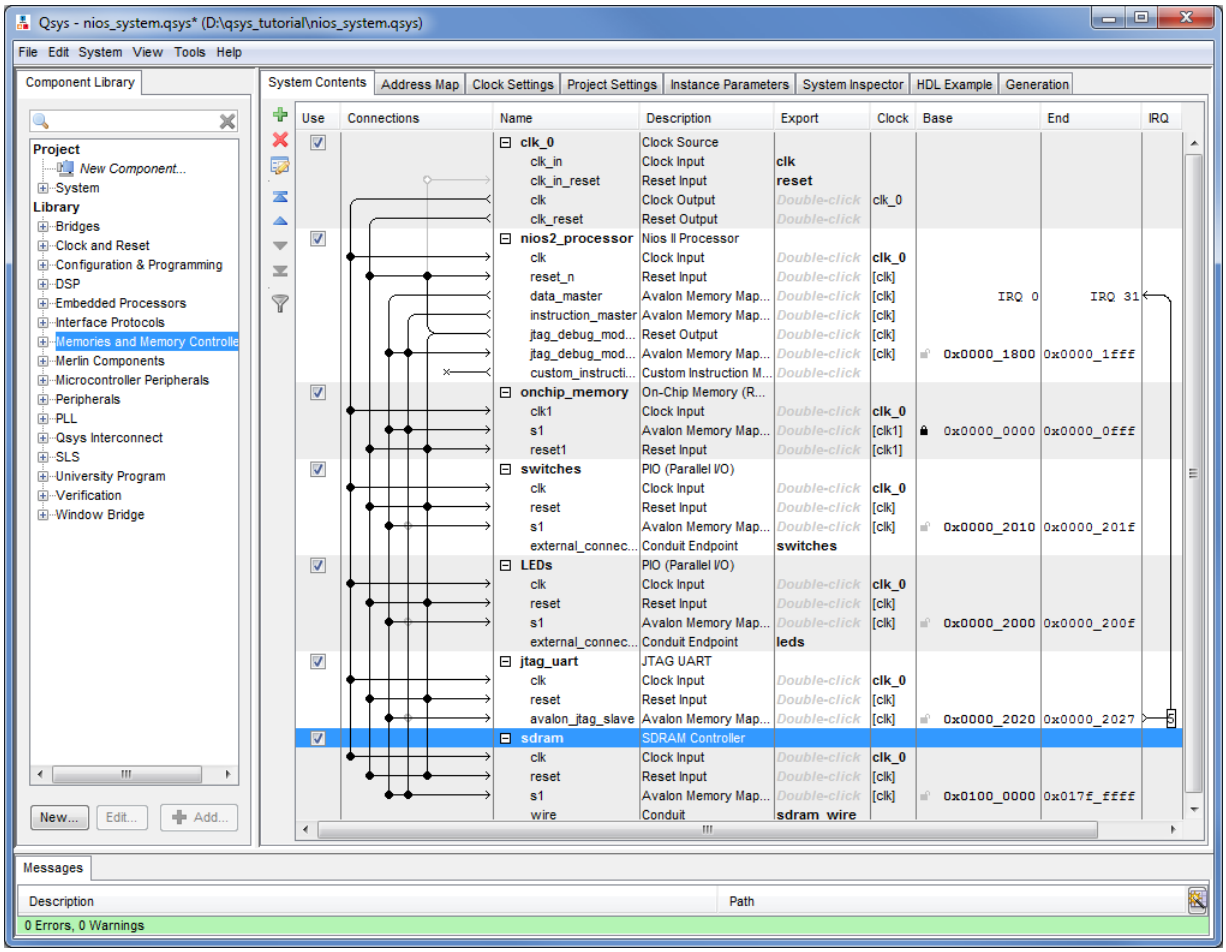


Figure 5. The expanded Nios II system.

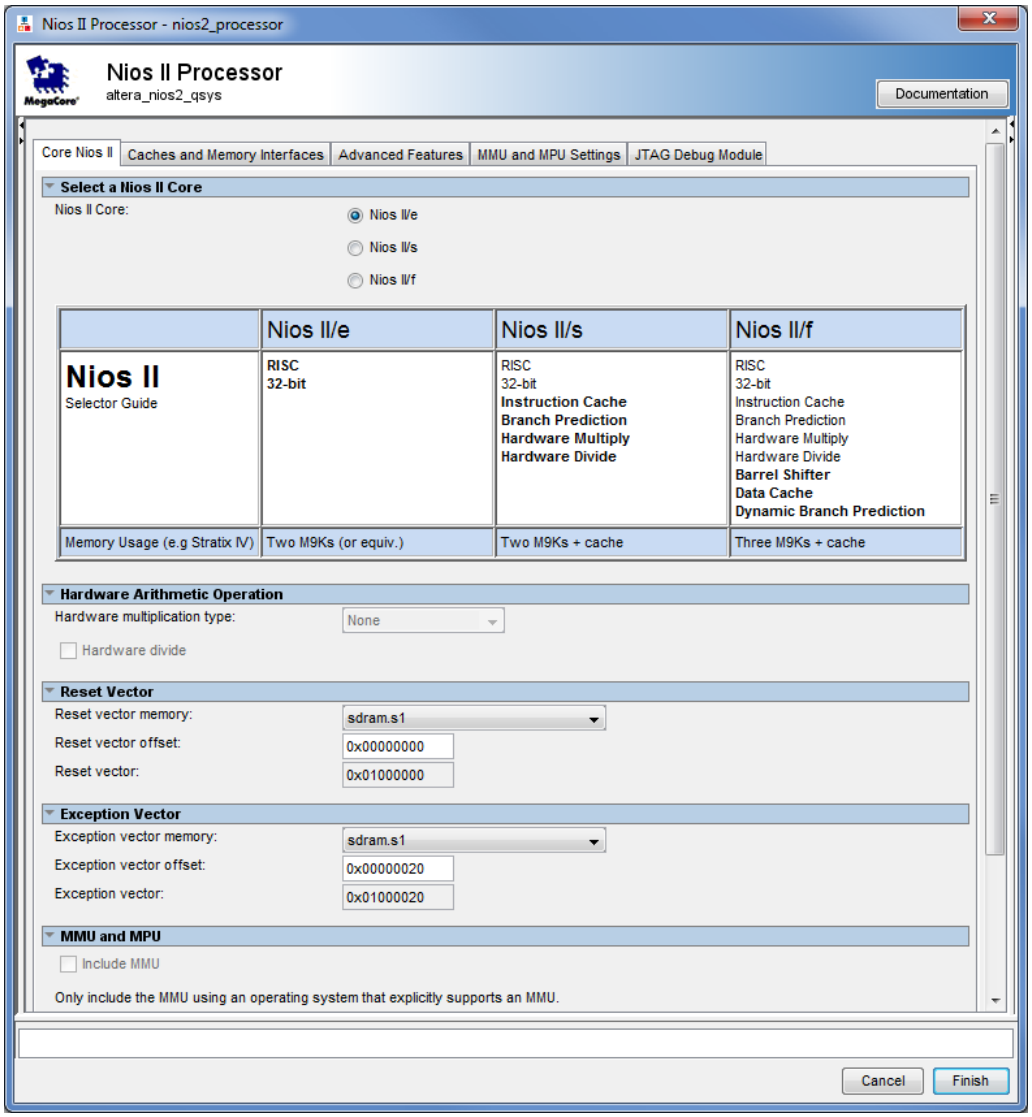


Figure 6. Define the reset vector and the exception vector.

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 7 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[11:0]*. The data lines are referred to as the **inout** vector *sdram_wire_dq[15:0]*. This is a vector of the **inout** type because the data lines are bidirectional.

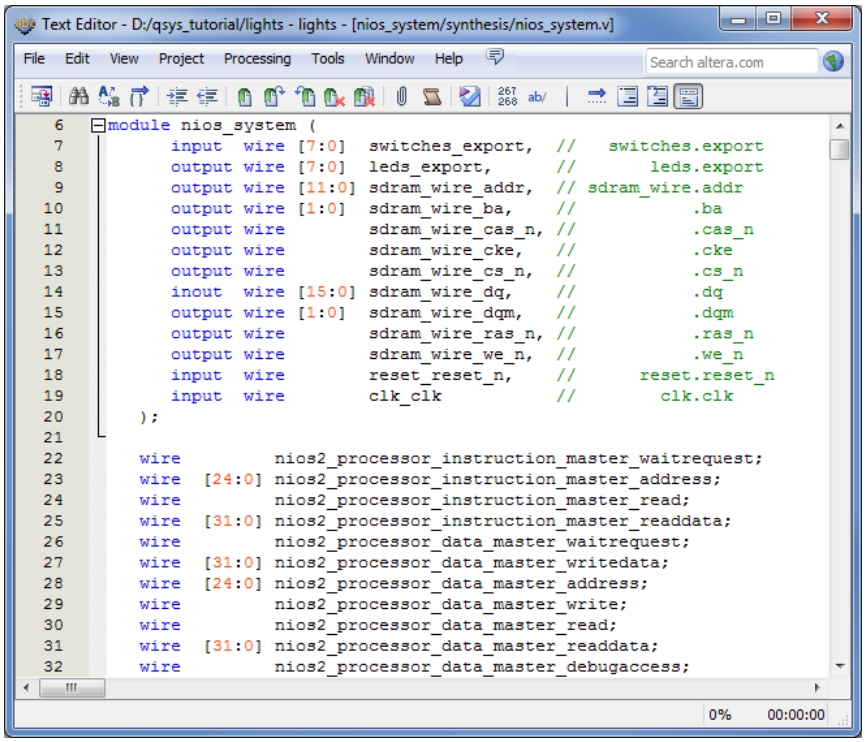


Figure 7. A part of the generated Verilog module.

## 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 8. 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 *DRAM_CLK*, *DRAM_CKE*, *DRAM_ADDR*, *DRAM_BA_1*, *DRAM_BA_0*, *DRAM_CS_N*, *DRAM_CAS_N*, *DRAM_RAS_N*, *DRAM_WE_N*, *DRAM_DQ*, *DRAM_UDQM*, and *DRAM_LDQM*, which correspond to the SDRAM signals indicated in Figure 2. All of these names are those specified in the DE2 User Manual, which allows us to make the pin assignments by importing them from the file called *DE2_pin_assignments.qsf* in the directory *tutorials\design_files*, which is included on the CD-ROM that accompanies the DE2 board and can also be found on Altera's DE2 web page.

Observe that the two *Bank Address* signals are treated by Qsys as a two-bit vector called *sdram_wire_ba[1:0]*, as seen in Figure 7. However, in the *DE2_pin_assignments.qsf* file these signals are given as scalars *DRAM_BA_1* and *DRAM_BA_0*. Therefore, in our Verilog module, we concatenated these signals as *{DRAM_BA_1, DRAM_BA_0}*. Similarly, the vector *sdram_wire_dqm[1:0]* corresponds to *{DRAM_UDQM, DRAM_LDQM}*.

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 **assign** statement in the code. This approach leads to a potential timing problem caused by the clock skew on the DE2 board, which can be fixed as explained in section 7.

```
// Implements the augmented Nios II system for the DE2 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 User Manual.
module  lights (SW, KEY, CLOCK_50, LEDG, DRAM_CLK, DRAM_CKE,
        DRAM_ADDR, DRAM_BA_1, DRAM_BA_0, DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N,
        DRAM_WE_N, DRAM_DQ, DRAM_UDQM, DRAM_LDQM);
        input  [7:0]  SW;
        input  [0:0]  KEY;
        input  CLOCK_50;
        output  [7:0]  LEDG;
        output  [11:0]  DRAM_ADDR;
        output  DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK;
        output  DRAM_CKE, DRAM_CS_N, DRAM_WE_N, DRAM_UDQM, DRAM_LDQM;
        inout  [15:0]  DRAM_DQ;
// Instantiate the Nios II system module generated by Qsys
        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_UDQM, DRAM_LDQM}),
                .sdram_wire_ras_n (DRAM_RAS_N),
                .sdram_wire_we_n (DRAM_WE_N)
        );

        assign  DRAM_CLK = CLOCK_50;
endmodule
```

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.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 board. Use the application program from the tutorial *Introduction to the Altera Qsys System Integration Tool*, which is shown in Figure 9

```
.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
.end
```

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 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.

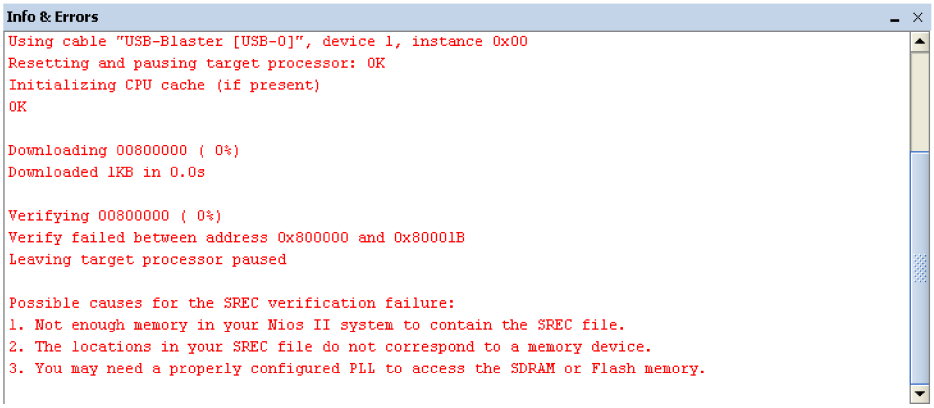


Figure 10. Error message in the Altera Monitor Program that may be due to the SDRAM clock skew problem.

# 7 Using the Clock Signals IP Core

The clock skew depends on physical characteristics of the DE2 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 Qsys tool window of Figure 5 select University Program > Clocks Signals for DE-Series Board Peripherals and click Add. A window depicted in Figure 11 appears. Select *DE2* 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 12. Click on the System Generation tab and regenerate the system.

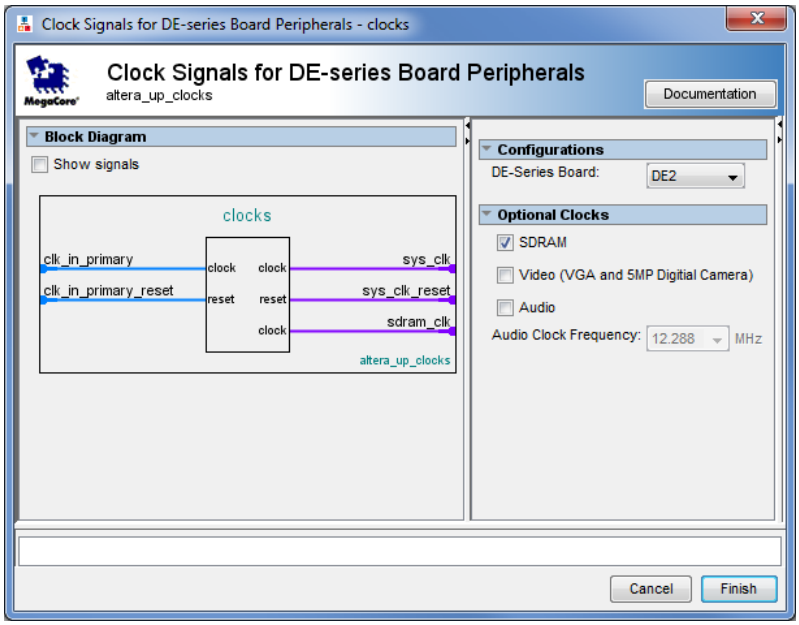


Figure 11. Clock Signals IP Core

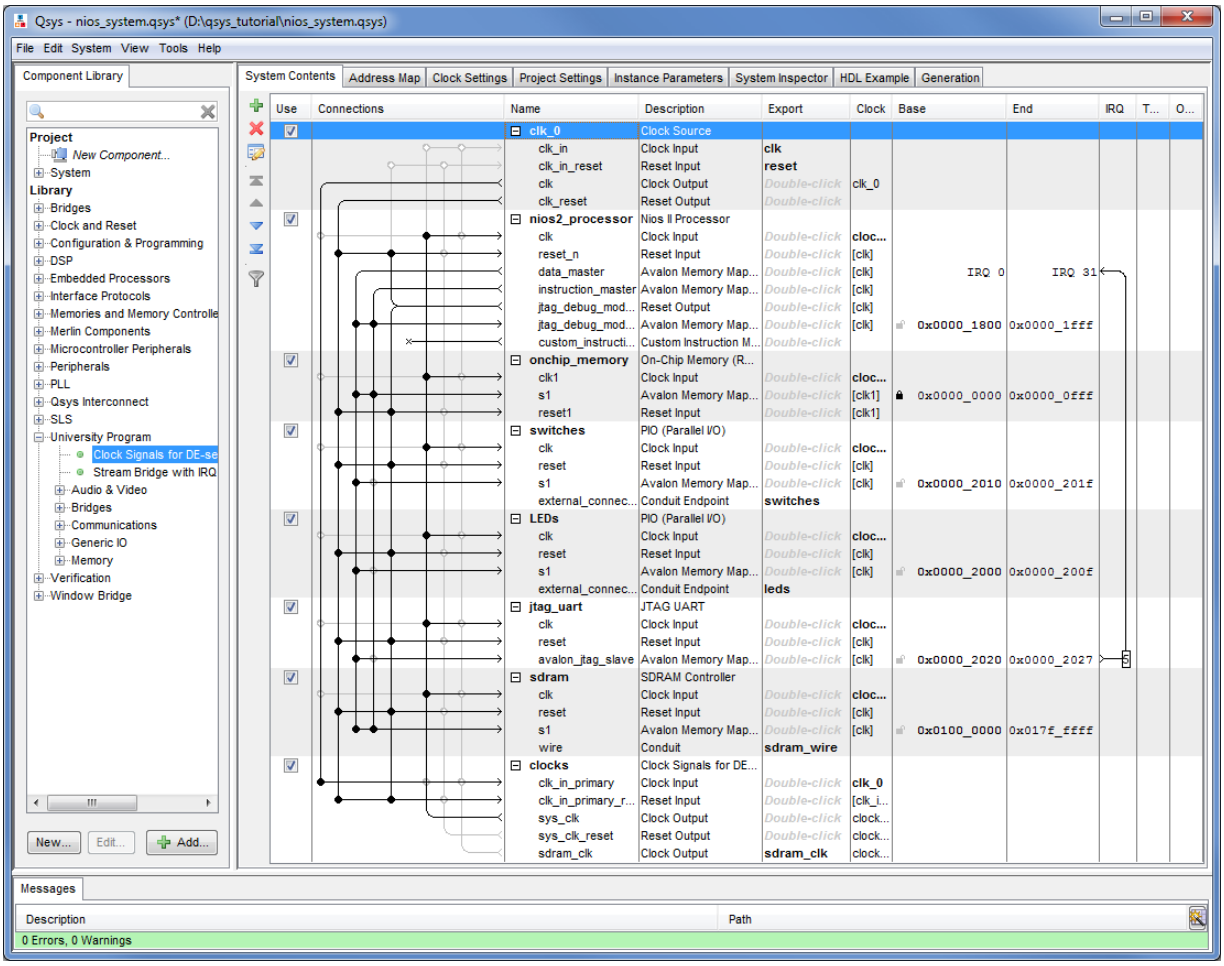


Figure 12. The final Nios II system.

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 13. The SDRAM clock signal *sdram_clk* generated by the Clock Signals core connects to the pin *DRAM_CLK*.

```
// Implements the augmented Nios II system for the DE2 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 User Manual.
module  lights (SW, KEY, CLOCK_50, LEDG, DRAM_CLK, DRAM_CKE,
      DRAM_ADDR, DRAM_BA_1, DRAM_BA_0, DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N,
      DRAM_WE_N, DRAM_DQ, DRAM_UDQM, DRAM_LDQM);
      input  [7:0]  SW;
      input  [0:0]  KEY;
      input  CLOCK_50;
      output  [7:0]  LEDG;
      output  [11:0]  DRAM_ADDR;
      output  DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK;
      output  DRAM_CKE, DRAM_CS_N, DRAM_WE_N, DRAM_UDQM, DRAM_LDQM;
      inout  [15:0]  DRAM_DQ;
// Instantiate the Nios II system module generated by the SOPC Builder
      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_UDQM, DRAM_LDQM}),
            .sdram_wire_ras_n (DRAM_RAS_N),
            .sdram_wire_we_n (DRAM_WE_N)
            .sdram_clk_clk (DRAM_CLK)
      );
endmodule
```

Figure 13. Proper instantiation of the expanded Nios II system.

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

Copyright ©1991-2012 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.