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

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:

The introductory tutorial Introduction to the Altera SOPC Builder Using Verilog 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 an SDRAM chip that can store 32 Mbytes of data. This memory is organized as 32M x16bits 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 Verilog Design tutorial. Figure 1 gives the block diagram of our example system.

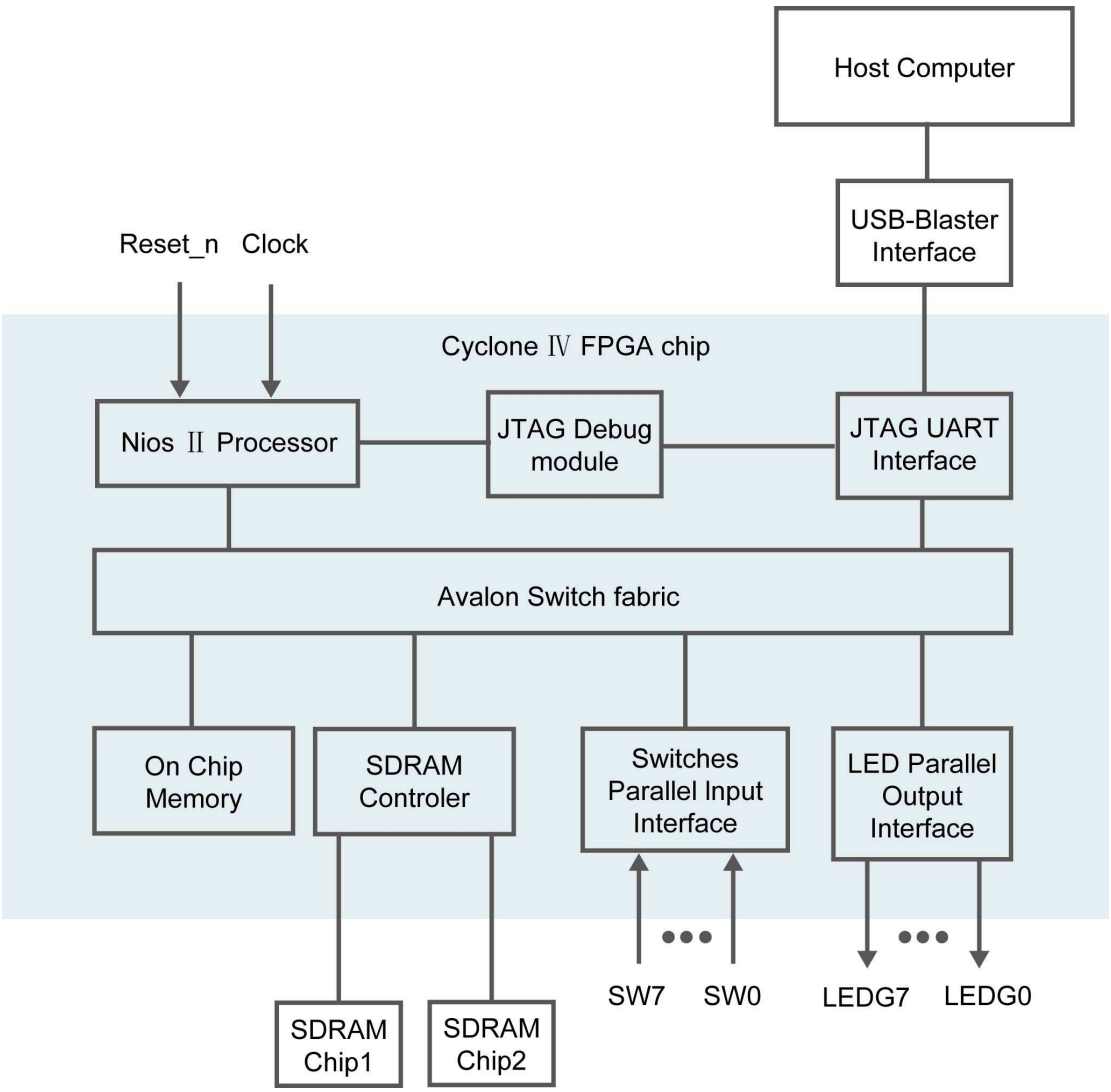

Figure 1 Example Nios II system implemented on the DE2-115 board.

The system realizes a trivial task. Eight toggle switches on the DE2-115 board, 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 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.

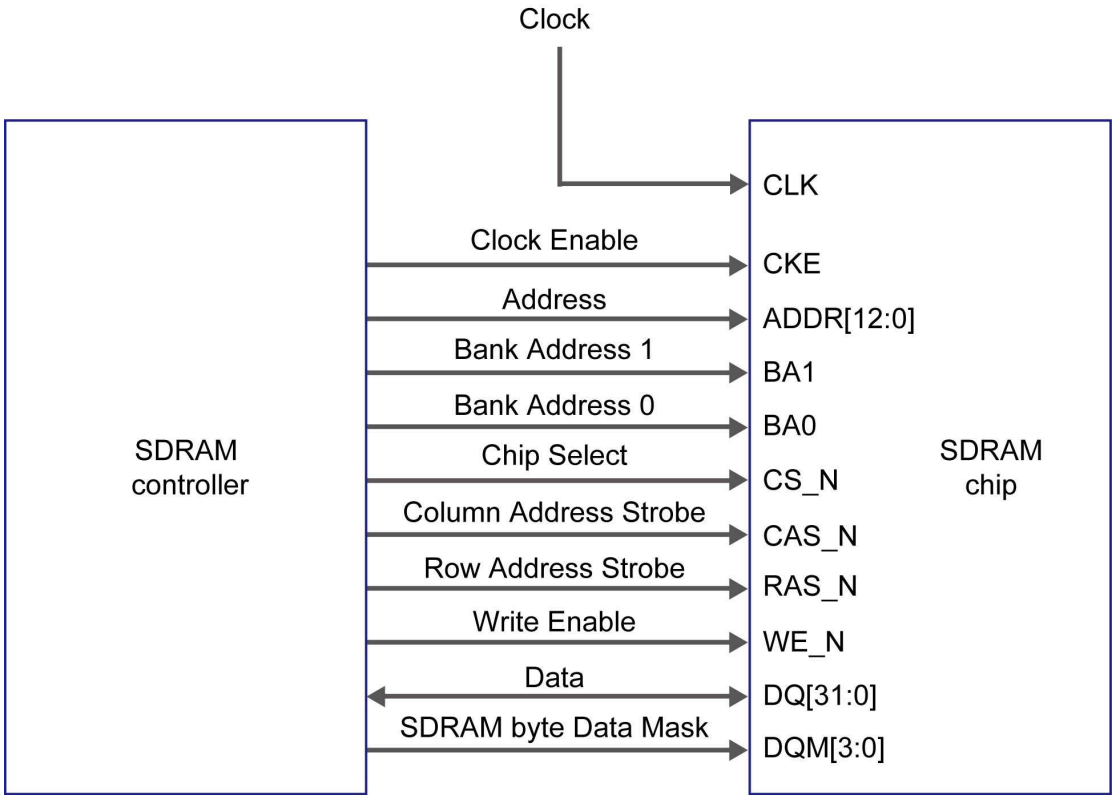

Figure 2 The SDRAM signals.

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

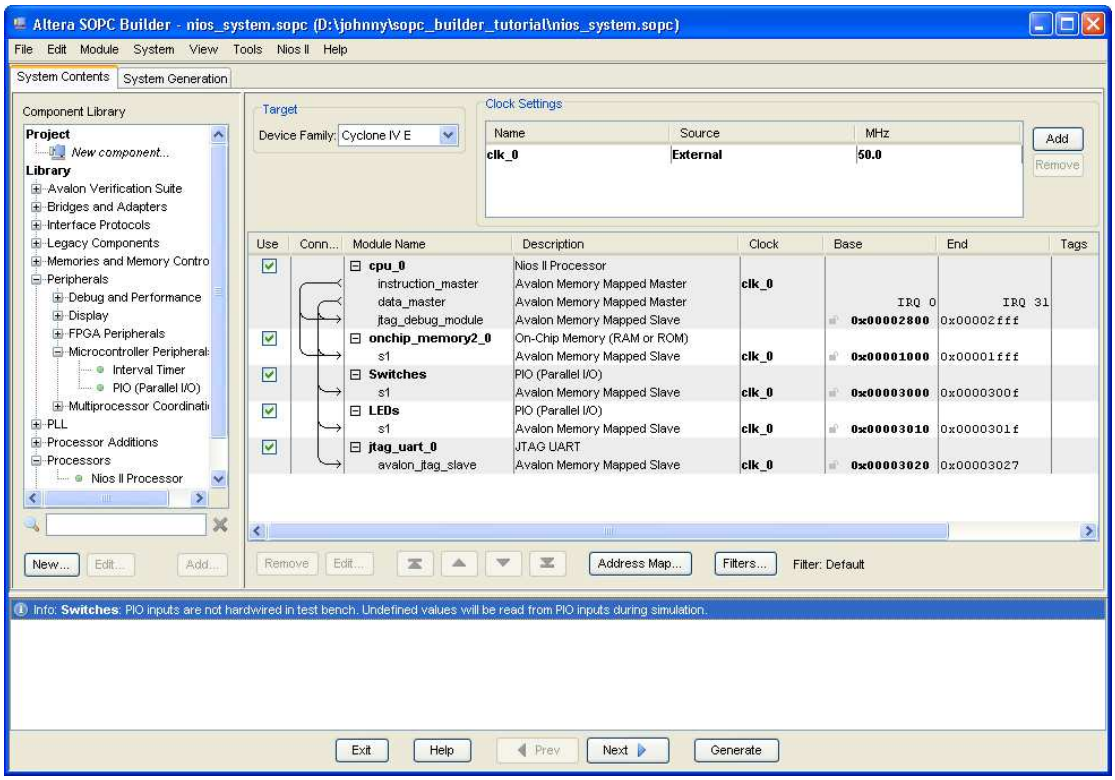

Figure 3.The Nios II system defined in the introductory tutorial.

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. 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.

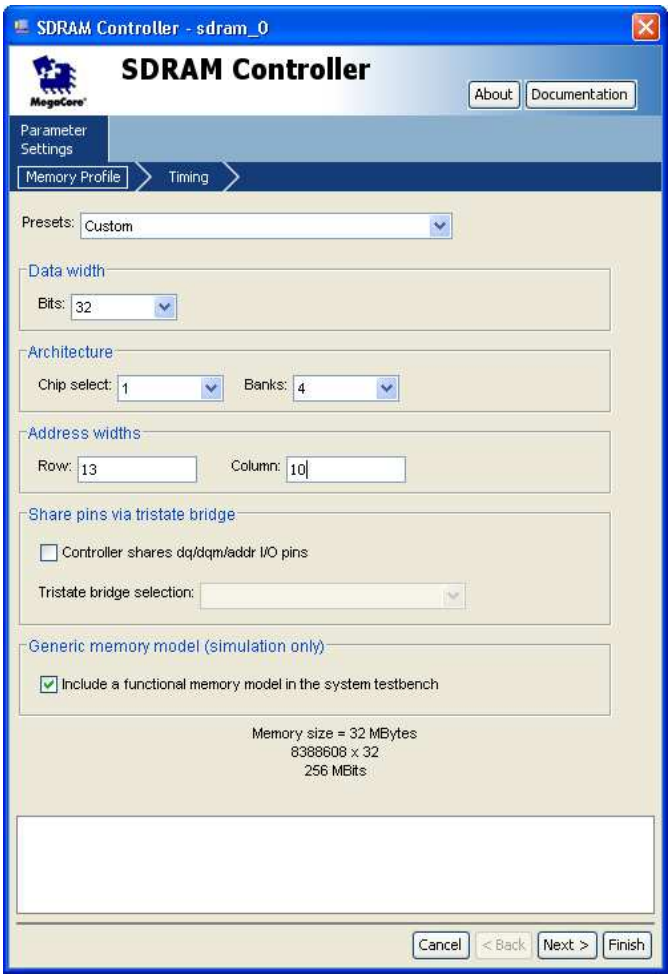

Figure 4.Add the SDRAM Controller.

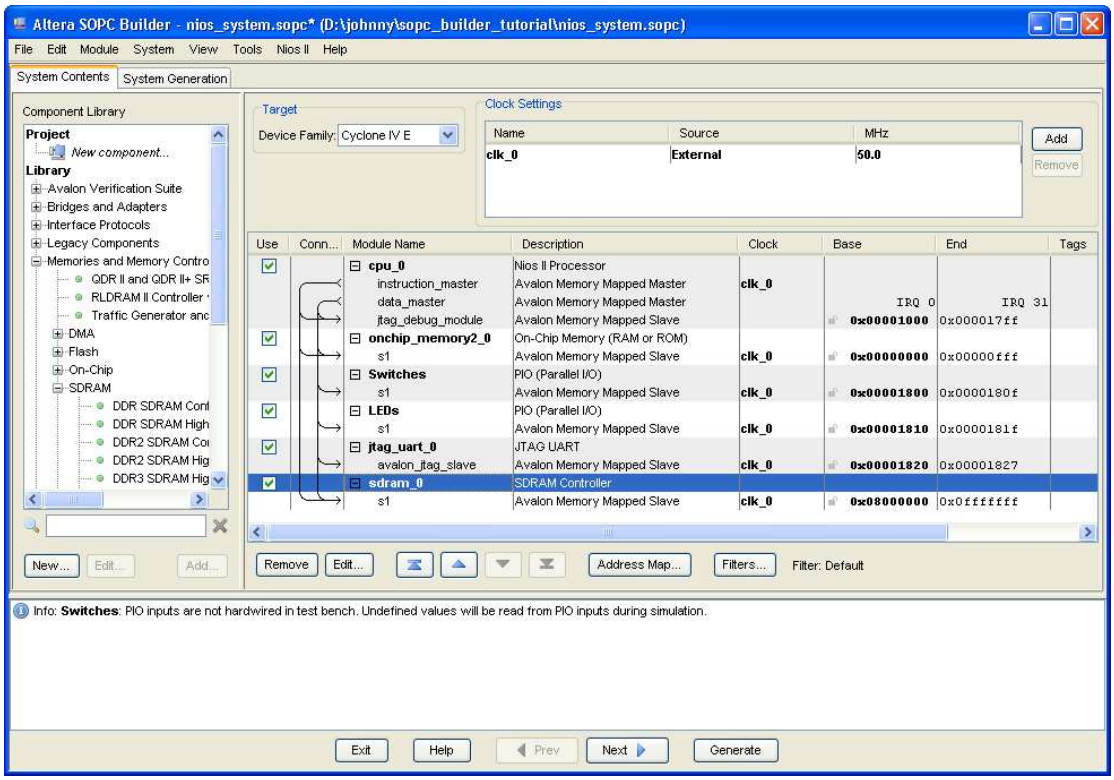

Figure 5.The expanded Nios II system.

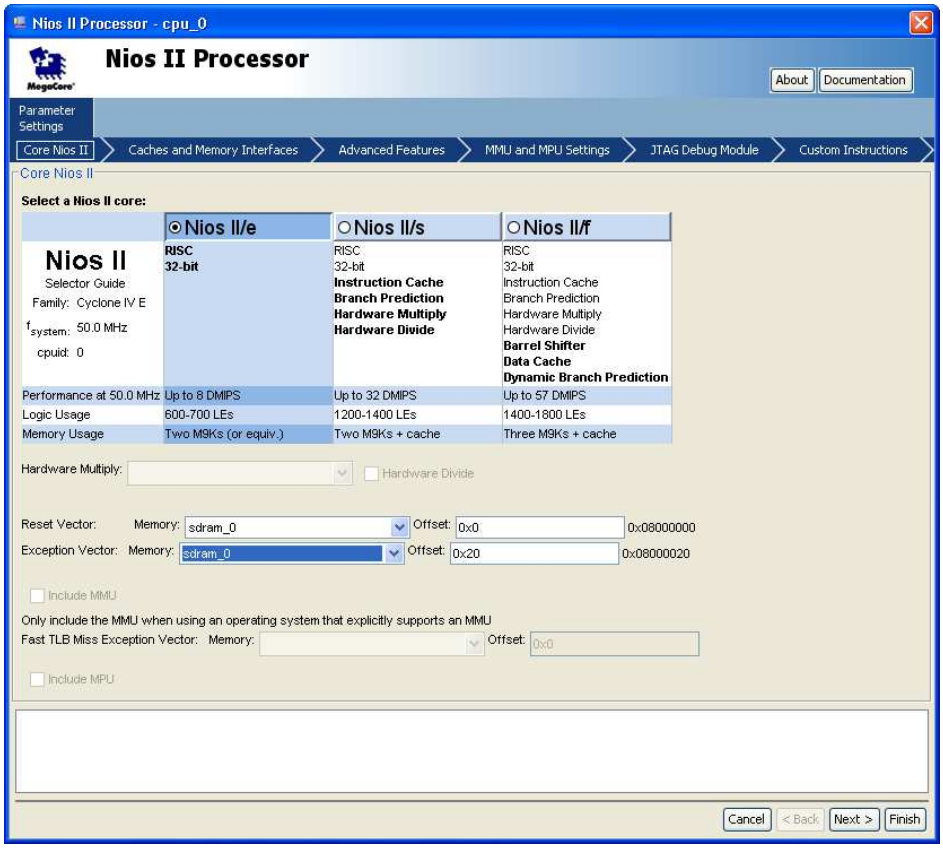

Figure 6. Set CPU Reset vector and Exception memory to sdram_0

The augmented Verilog module generated by the SOPC Builder is in the file

nios_system.v (nios_led.v) in the 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 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 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 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.

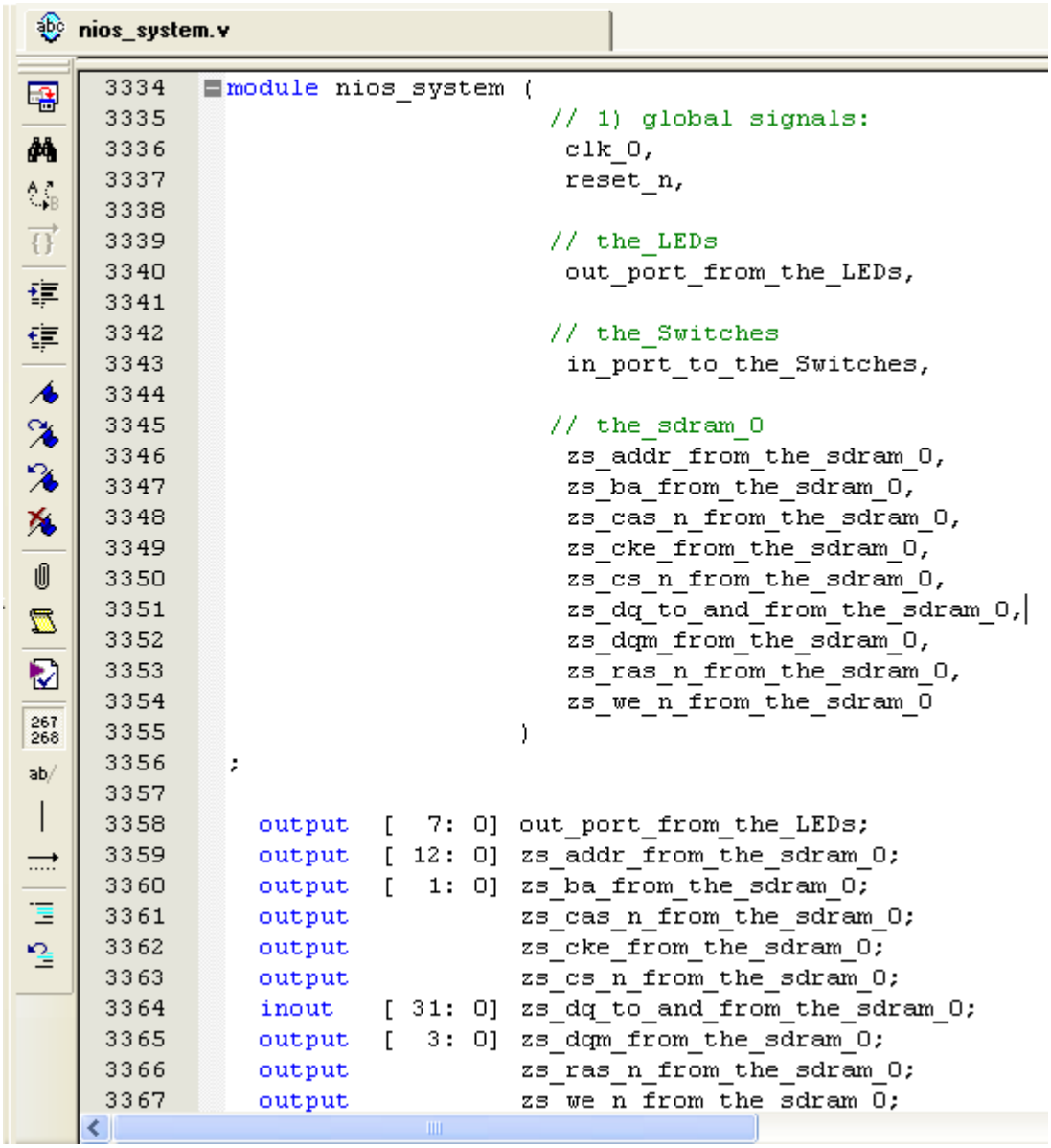

Figure 7.A part of the generated Verilog module.

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 Verilog module, as we have done in the tutorial Introduction to the Altera SOPC Builder Using Verilog Design. 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,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 assign 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.

```
// Implements the augmented Nios II system for the DE2-115 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 DE2-115 User Manual.


module   lights (SW, KEY, CLOCK_50, LEDG, DRAM_CLK, DRAM_CKE,
DRAM_ADDR, DRAM_BA, DRAM_CS_N, DRAM_CAS_N, DRAM_RAS_N,
DRAM_WE_N, DRAM_DQ, DRAM_DQM);
input      [7:0] SW;
input [0:0] KEY;
input      CLOCK_50;
output     [7:0] LEDG;
output     [12:0] DRAM_ADDR;
output     [1:0] DRAM_BA;
output          DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK;
output          DRAM_CKE, DRAM_CS_N, DRAM_WE_N;
output [3:0] DRAM_DQM;
inout      [31:0] DRAM_DQ;

// Instantiate the Nios II system module generated by the SOPC Builder
nios_system NiosII (
CLOCK_50,
KEY [0],
LEDG,
SW,
DRAM_ADDR,
DRAM_BA,
DRAM_CAS_N,
DRAM_CKE,
DRAM_CS_N,
```

```
DRAM_DQ,
DRAM_DQM,
DRAM_RAS_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 all the *.v 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 Verilog 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.