

NIOS CPU Based Embedded Computer System on Programmable Chip

EE8205: Embedded Computer Systems

NIOS-II SoPC: PART-I

1 Lab Objectives

This lab has been constructed to introduce the development of dedicated embedded system based on NIOS-II CPU core and other interface IPs from Altera. DE2-115 Altera board having a Cyclone-IV E FPGA is used to implement the target embedded computer system. This lab introduces you to the system development flow for the Nios II processor. This lab-tutorial is a good starting point to the general concept of building embedded systems in FPGAs. In this lab you will build a Nios II hardware system and create a software program to run on the Nios II system. This lab also presents an introduction to Altera's SOPC Builder software, which is used to implement a system that uses the Nios II processor on an Altera FPGA device.

2 Introduction

In this lab, students will develop and implement a Nios-II based embedded computer system by employing a Cyclone-IV E FPGA based DE2-115 board from Altera. Quartus II CAD tools along with SOPC builder and Nios IDE software development environment is used throughout in this lab.

Nios II CPU is a soft processor, defined in a hardware description language (e.g. Verilog or VHDL), which can be implemented in Altera's FPGA devices by using the Quartus II CAD system. To implement a useful embedded system it is necessary to add other functional units such as memories, input/output interfaces, timers, communications interfaces, etc. To facilitate the implementation of such systems, it is useful to have computer-aided-design (CAD) software for implementing an embedded system-on-a-programmable-chip (SOPC). You are going to build a NIOS CPU based system shown as Quartus block diagram. The NIOS system will read 8 binary switches and show their values on 8 LEDs given on the DE2-115 board.

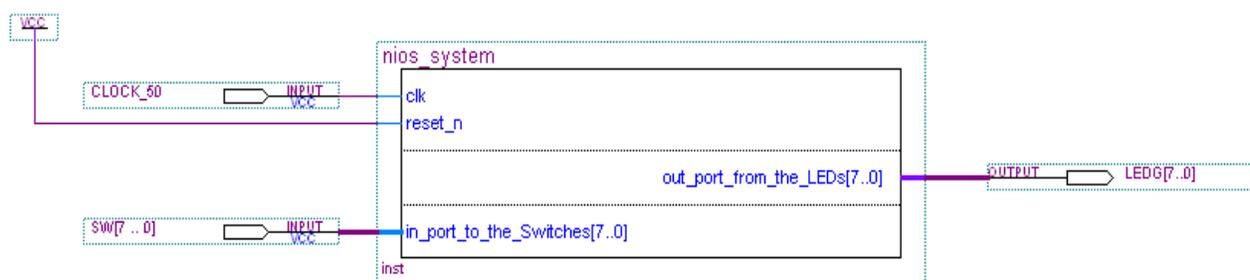


Figure 1: Example Nios II System with I/O

In this lab, the system development flow is illustrated by giving step-by-step instructions for using the SOPC Builder in conjunction with the Quartus II software to implement a simple embedded computer system. The last step in the development process involves configuring the designed circuit in an actual FPGA device, and running an application program. **This lab must be completed with Quartus II version 11.1. Note: You need to open the terminal and type the command “quartus11.1”. Do not use any newer versions.**

3 NIOS II System

In this section the complete design flow for creating a Nios II system and prototyping it on a target board is discussed. Figure 2 shows the Nios II system development flow consists of three types of development: hardware design steps, software design steps, and system design steps, involving both hardware and software. For simpler Nios II systems, one person might perform all the steps. For more complex systems, separate hardware and software designers might be responsible for different steps. System design steps involve both the hardware and software, and might require input from both sides. In the case of separate hardware and software teams, it is important to know exactly what files and information must be passed between teams at the points of intersection in the design flow.

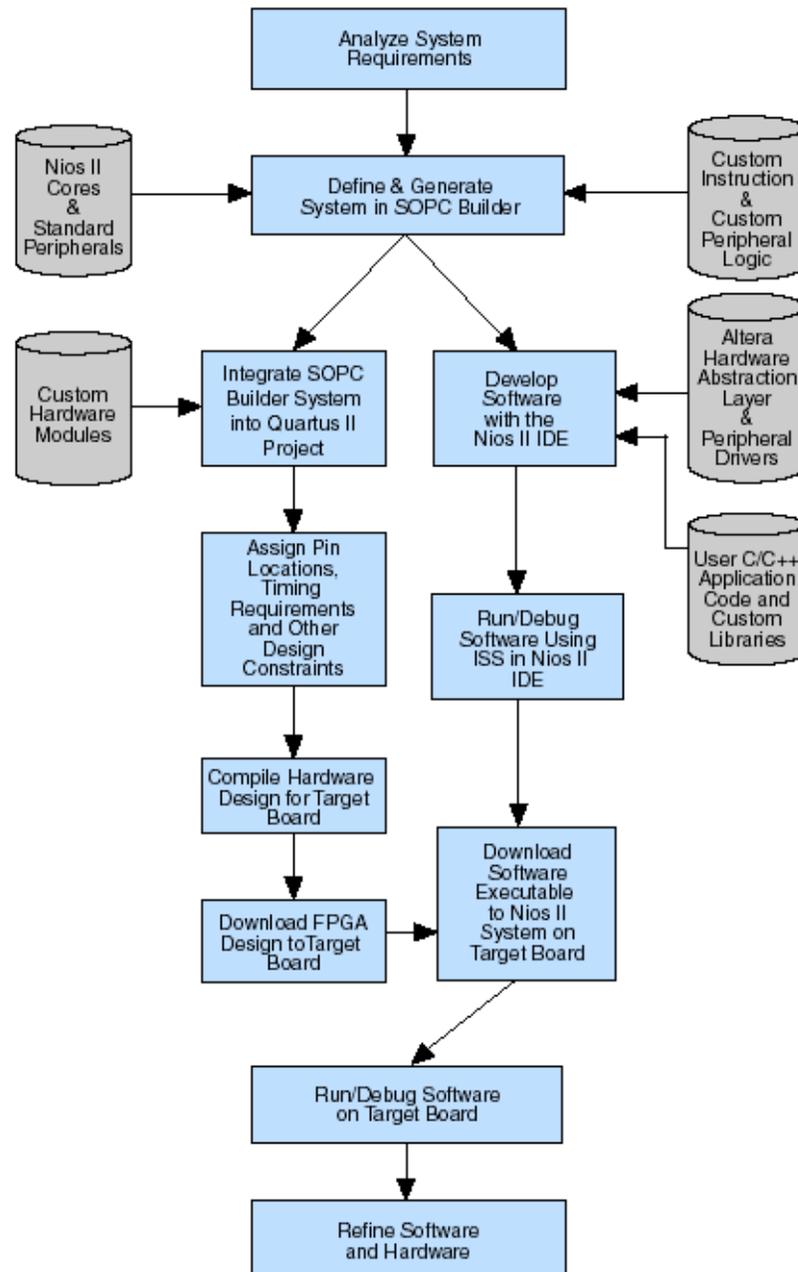


Figure 2: Nios II System Development Flow

A simple Nios II based embedded system that will be implemented in this lab/tutorial on the DE2-115 board is shown in Figure 3. The Nios II processor and the interfaces needed to connect to other ICs on the DE2-115 board are implemented over the Cyclone IV E FPGA. These components are interconnected by means of an interconnection network called the Avalon Switch Fabric. The memory blocks in the Cyclone IV-E device can be used to provide an on-chip memory for the Nios II processor. The SRAM, SDRAM and Flash memory chips on the DE2-115 board are accessed through appropriate interfaces. Parallel and serial input/output interfaces provide typical I/O ports used in embedded computer systems. A special JTAG UART interface is used to connect to the circuitry that provides a Universal Serial Bus (USB) link to the host computer to which the DE2-115 board is connected. This circuitry and the associated software is known as *USB-Blaster*. Another module, JTAG Debug module, is provided to allow the host computer to control the Nios II system. It makes it possible to perform operations such as downloading programs into memory, starting and stopping execution, setting breakpoints, and collecting real-time execution trace data. All parts of the Nios II system implemented on the FPGA are defined by using a hardware description language. One can also use the SOPC Builder tool to implement the desired embedded system simply by choosing the required components and specifying the parameters needed to make each component fit the overall requirements of the system. In this lab/tutorial, we will also illustrate the capability of the SOPC Builder by designing the simple embedded system of Figure 3. The same approach can be used to design large and complex embedded computer systems.

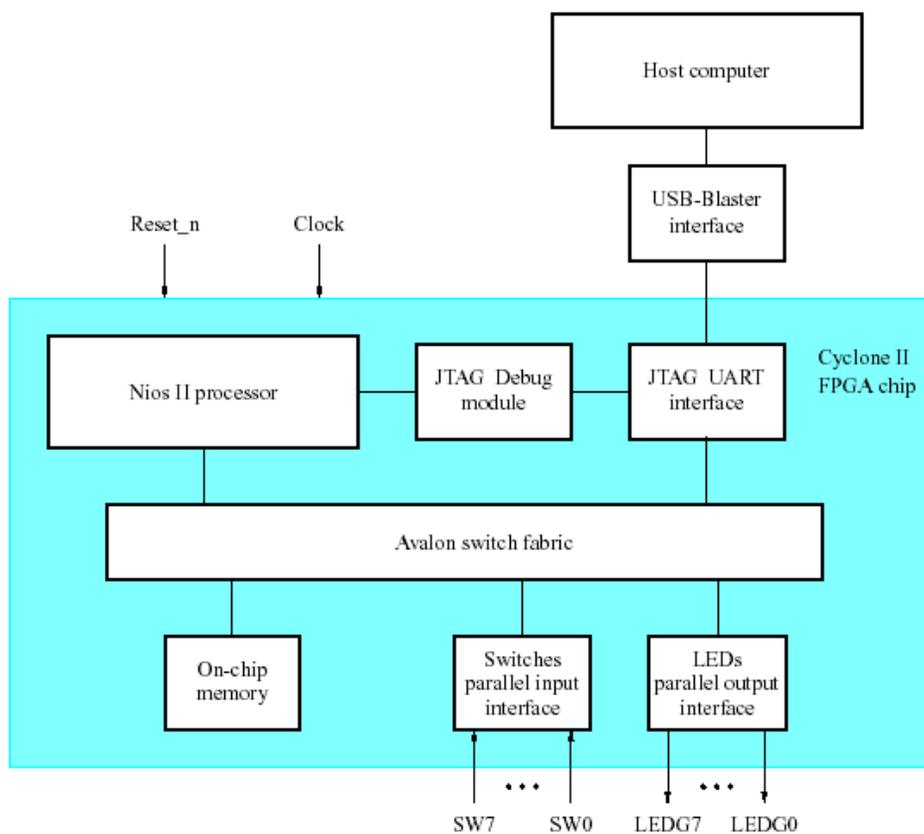


Figure 3: A Simple Nios II based Embedded Computer System.

Our example system of Figure 3 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. The Nios II CPU will perform this by executing a program stored in the on-chip memory. Continuous operation is required, such that as the switches are toggled the lights change accordingly. You will use the SOPC Builder to design the hardware depicted in Figure 3. Next, you will also assign the Cyclone IV E pins to realize the

connections between the parallel interfaces and the switches and LEDs, which act as I/O devices. Then, you will configure the FPGA to implement the designed system. Finally, you will use the IDE software development environment to compile, download and execute a Nios II program that performs the desired task. Doing this lab, the students will learn about:

- Quartus II CAD system
- Using the SOPC Builder to design a Nios II-based embedded computer system
- Integrating the designed Nios II computer system into a Quartus II project
- Implementing the designed system on the DE2-115 board
- Running an application program on the Nios II processor

4 Main Steps for System Configuration by using SOPC Builder

SOPC builder allows the user to easily create a system based on the Nios II processor, by simply selecting the desired functional units and specifying their parameters. To implement the system of Figure 2, you have to instantiate the following functional units:

- Nios II processor, which is referred to as a Central Processing Unit (CPU)
- On-chip memory, which consists of the memory blocks in the Cyclone IV-E chip; we will specify a 24-Kbyte memory arranged in 32-bit words
- Two parallel I/O interfaces
- JTAG UART interface for communication with the host computer

To define the target embedded computer system, start the Quartus II software and perform the following steps: Create a new Quartus II project for your system. As shown in Figure 4, we stored our project in a directory called *sopc_builder_tutorial*, and we assigned the name *lights* to both the project and its top-level design entity. You can choose a different directory or project name. In your project, choose the EP4CE115F29C7 chip as the target device, because this is the FPGA on the DE2-115 board. You may also attach a preliminary block diagram/schematic file “lights.bdf” for your NIOS system while creating the project.

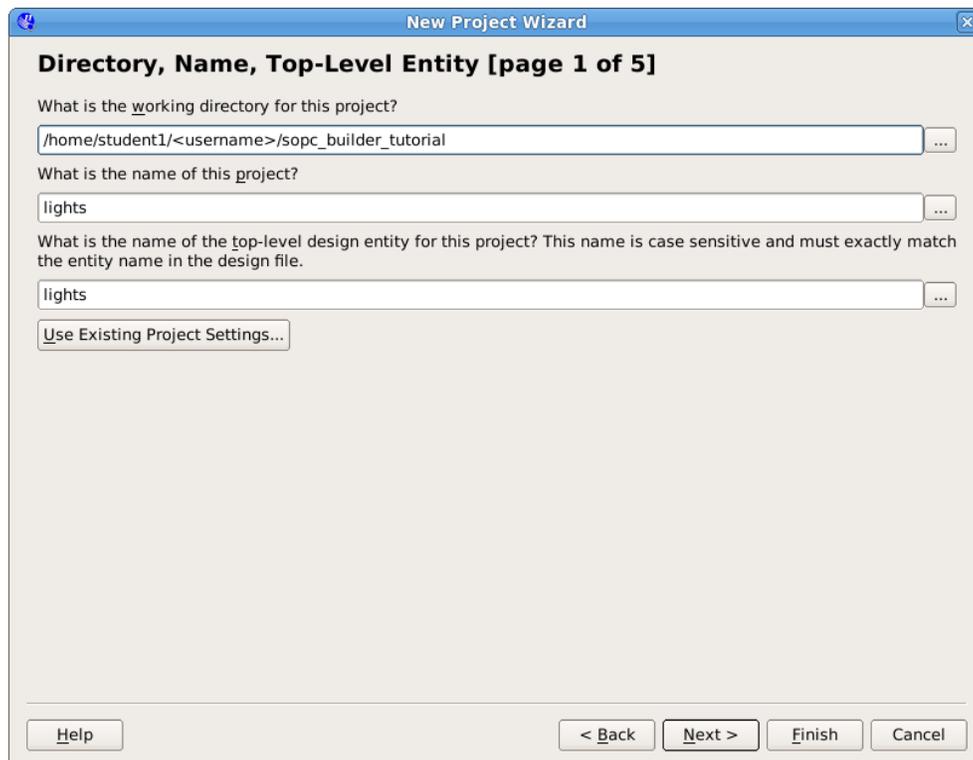


Figure 4. Create a new project.

2. Select Tools > SOPC Builder, which leads to the pop-up box in Figure 5. You may get a notice saying you should migrate to Qsys. You can safely ignore such notice. Enter *nios_system* as the system name; this will be the name of the system that the SOPC Builder will generate. Choose VHDL as the target HDL, in which the system module will be specified. Click OK to reach the window in Figure 6.

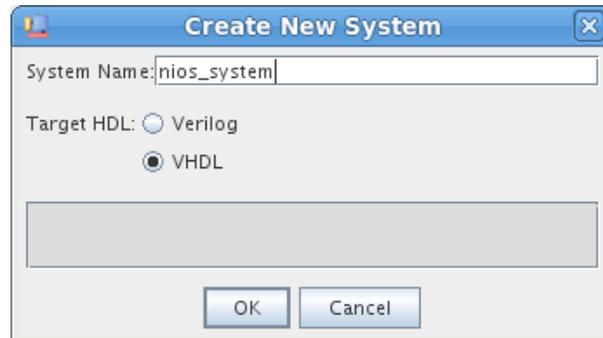


Figure 5. Create a new nios II system.

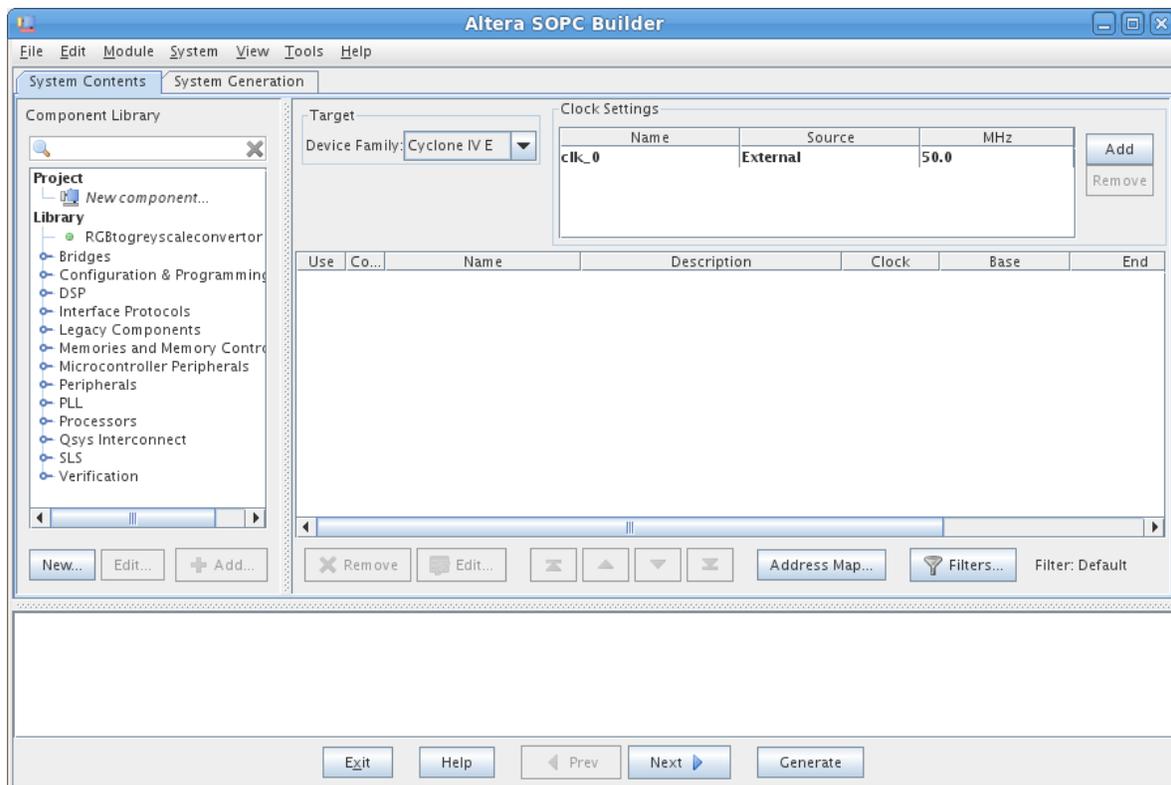


Figure 6. The System Contents tab window.

3. Figure 6 displays the System Contents tab of the SOPC Builder, which is used to add components to the system and configure the selected components to meet the design requirements. The available components are listed on the left side of the window. Before choosing our components, examine the area in the figure labeled Target. Ensure that Cyclone IV E is selected.
4. The Nios II processor runs under the control of a clock. For this lab we will make use of the 50-MHz clock that is provided on the DE2-115 board. It is possible to specify the names and frequency of clock signals in the SOPC Builder. If not already included in this list, specify a clock named *clk* with the source designated as External and the frequency set to 50.0 MHz.
5. In this step, specify the Nios processor as follows:

- On the left side of the window in Figure 6 in Library, select Processors > Nios II Processor and click Add, which leads to the window in Figure 7. Alternatively, you can search the component by name.
- Choose Nios II/e which is the simplest version of the processor. Click Finish to return to the window in Figure 6, which now shows the Nios II processor specified as indicated in Figure 8. There may be some warnings or error messages displayed in the SOPC Builder Messages window (at the bottom of the screen), because some parameters have not yet been specified. Ignore these messages as we will provide the necessary data later. Observe also that a new tab called **Nios II Processor “cpu_0” Settings** appears, which allows further configuration of the processor - we will not use it.

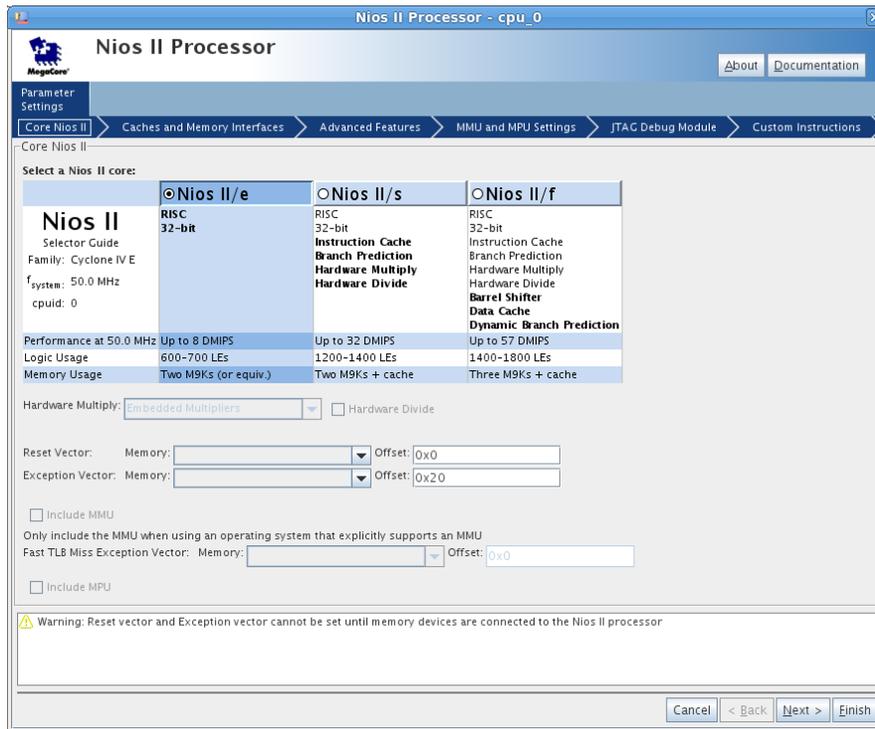


Figure 7. Create a Nios II processor.

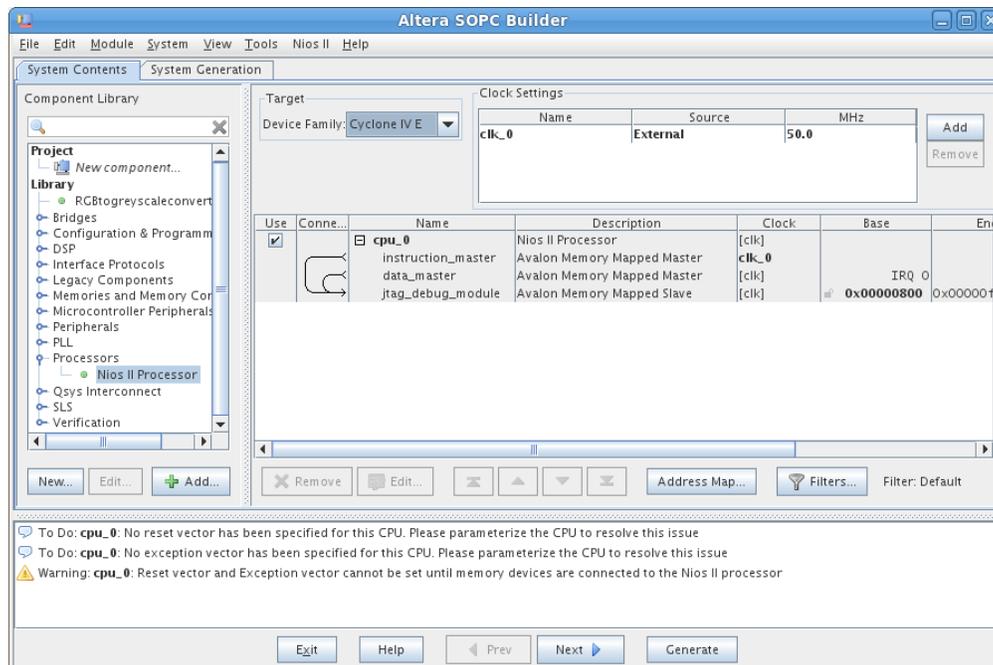


Figure 8. The defined Nios II processor.

6. To specify the on-chip memory perform the following:
- Select in the Library, Memories and Memory Controllers > On-Chip > On-Chip Memory (RAM or ROM) and click Add
 - In the On-Chip Memory Configuration Wizard window, shown in Figure 9, set the memory width to 32 bits and the total memory size to 20480 bytes
 - Do not change the other default settings
 - Click Finish, which returns to the System Contents tab as indicated in Figure 10

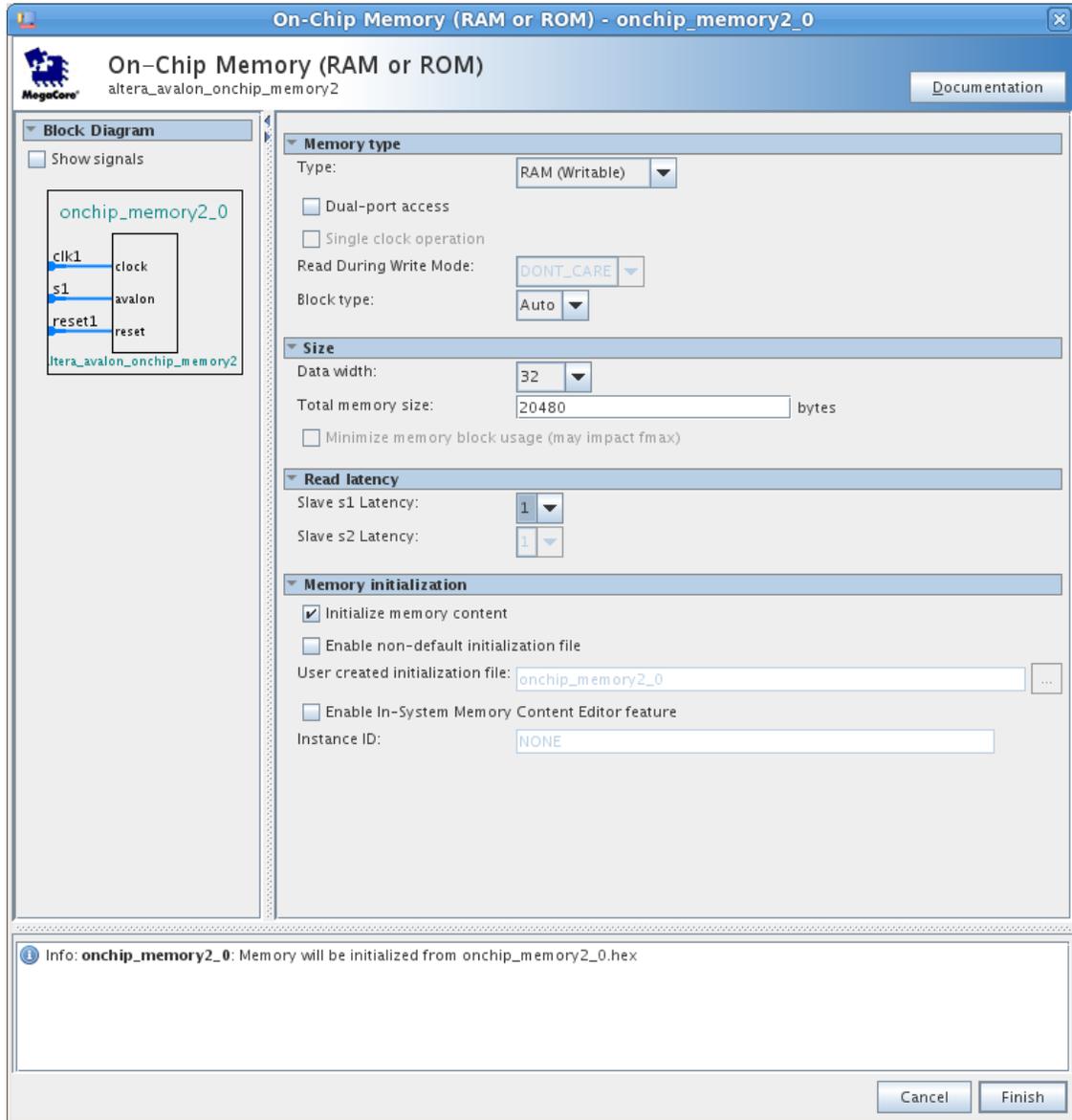


Figure 9. Define the on-chip memory.

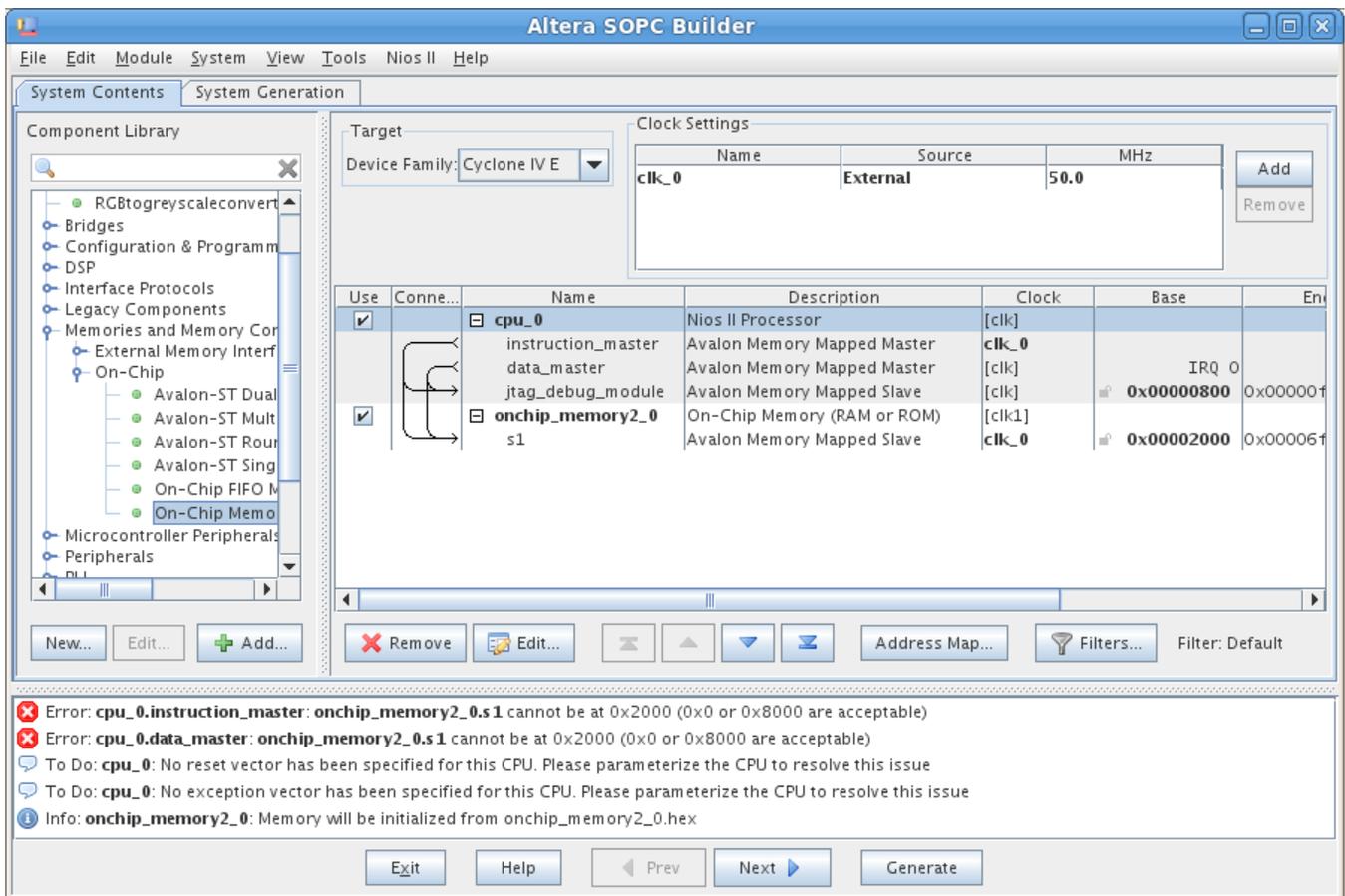


Figure 10. Including the on-chip Memory

7. Specify the input parallel I/O interface as follows:

- Select in the Library, Peripherals > Microcontroller Peripherals > PIO (Parallel I/O) and click Add to reach the PIO Configuration Wizard in Figure 11
- Specify the width of the port to be 8 bits and choose the direction of the port to be Input, as shown in this figure.
- Click Finish to return to the System Contents tab as given in Figure 12

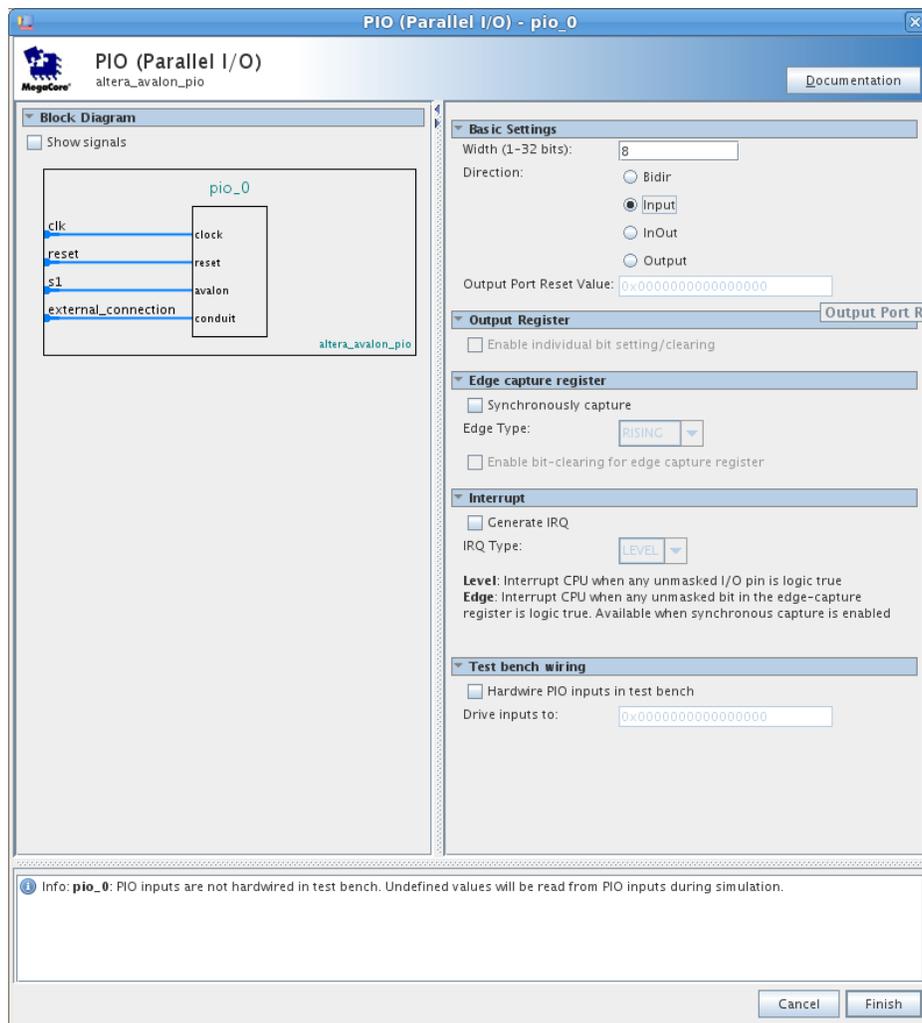


Figure 11. Defining a parallel input interface.

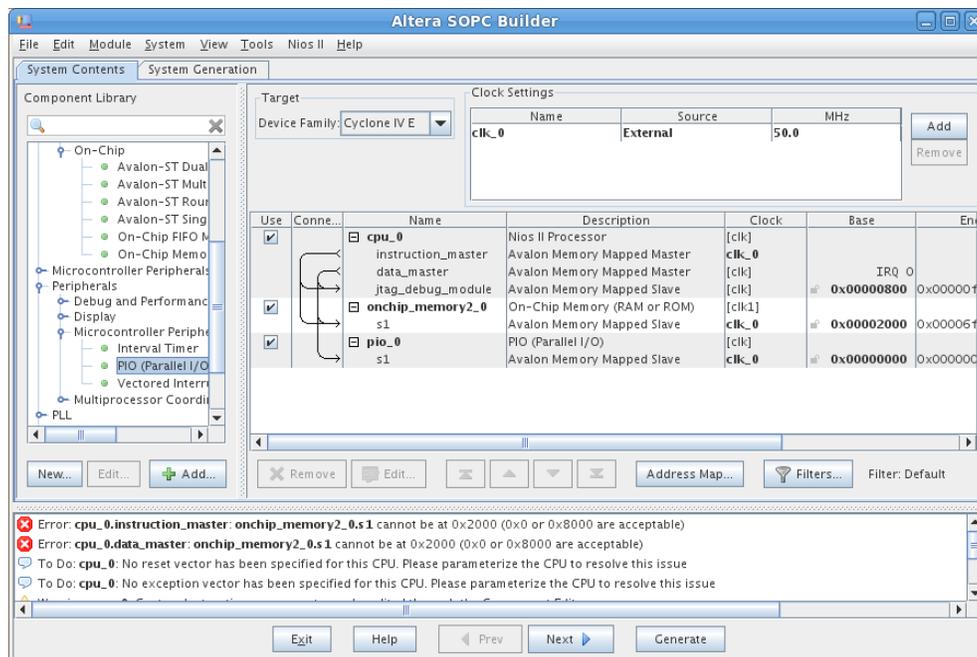


Figure 12. The parallel input interface is included.

8. In the same way, specify the output parallel I/O interface:
 - Select in the Library, Peripherals > Microcontroller Peripherals > PIO (Parallel I/O) and click Add to reach the PIO Configuration Wizard again
 - Specify the width of the port to be 8 bits and choose the direction of the port to be Output
 - Click Finish to return to the System Contents tab.

9. We wish to connect to a host computer and provide a means for communication between the Nios II system and the host computer. This can be accomplished by instantiating the JTAG UART interface as follows:
 - Select in the Library, Interface Protocols > Serial > JTAG UART and click Add to reach the JTAG UART Configuration Wizard in Figure 13
 - Do not change the default settings
 - Click Finish to return to the System Contents tab

10. The complete system is depicted in Figure 14. Note that the SOPC Builder automatically chooses names for the various components. The names are not necessarily descriptive enough to be easily associated with the target design, but they can be changed. In Figure 3, we use the names Switches and LEDs for the parallel input and output interfaces, respectively. These names can be used in the implemented system. Right-click on the pio_0 name and then select Rename. Change the name to Switches. Similarly, change pio_1 to LEDs.

11. The base and end addresses of the various components in the designed system can be assigned by the user, but the SOPC Builder can also assign them automatically. We will choose the latter possibility. So, select the command (using the menus at the top of the SOPC Builder window) System > Auto-Assign Base Addresses, which produces the assignment shown in Figure 15.

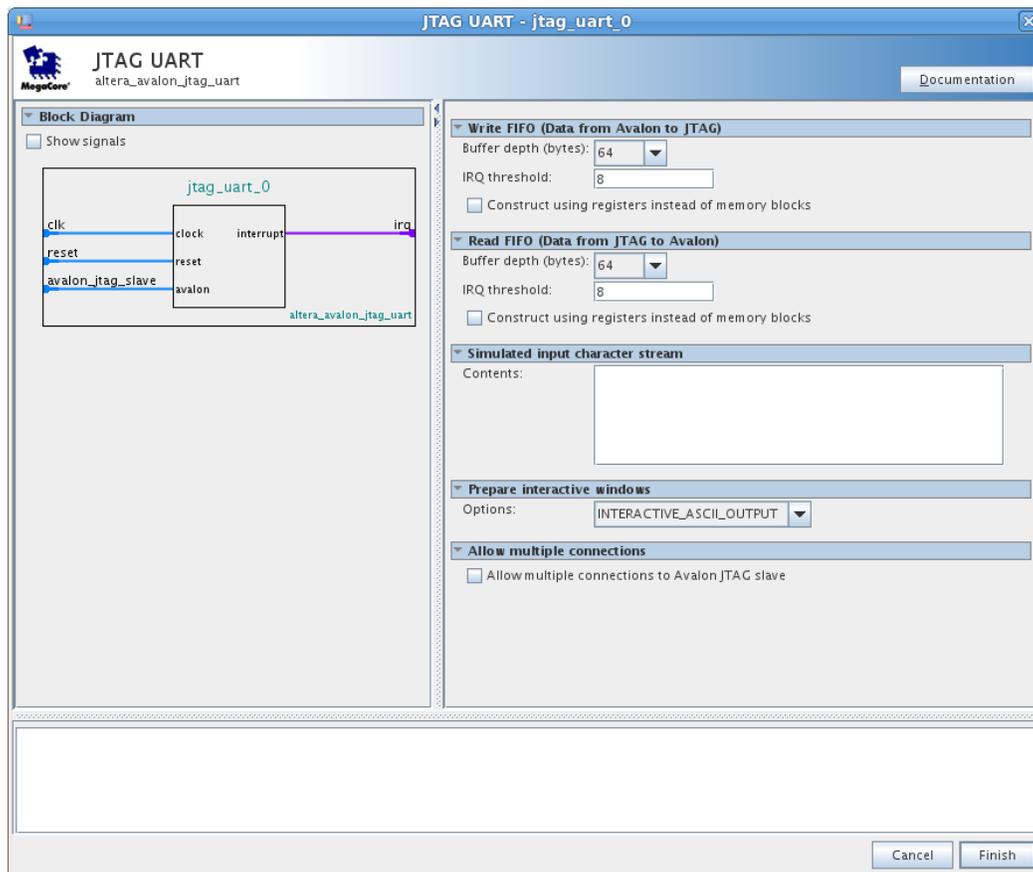


Figure 13. Defining the JTAG UART interface.

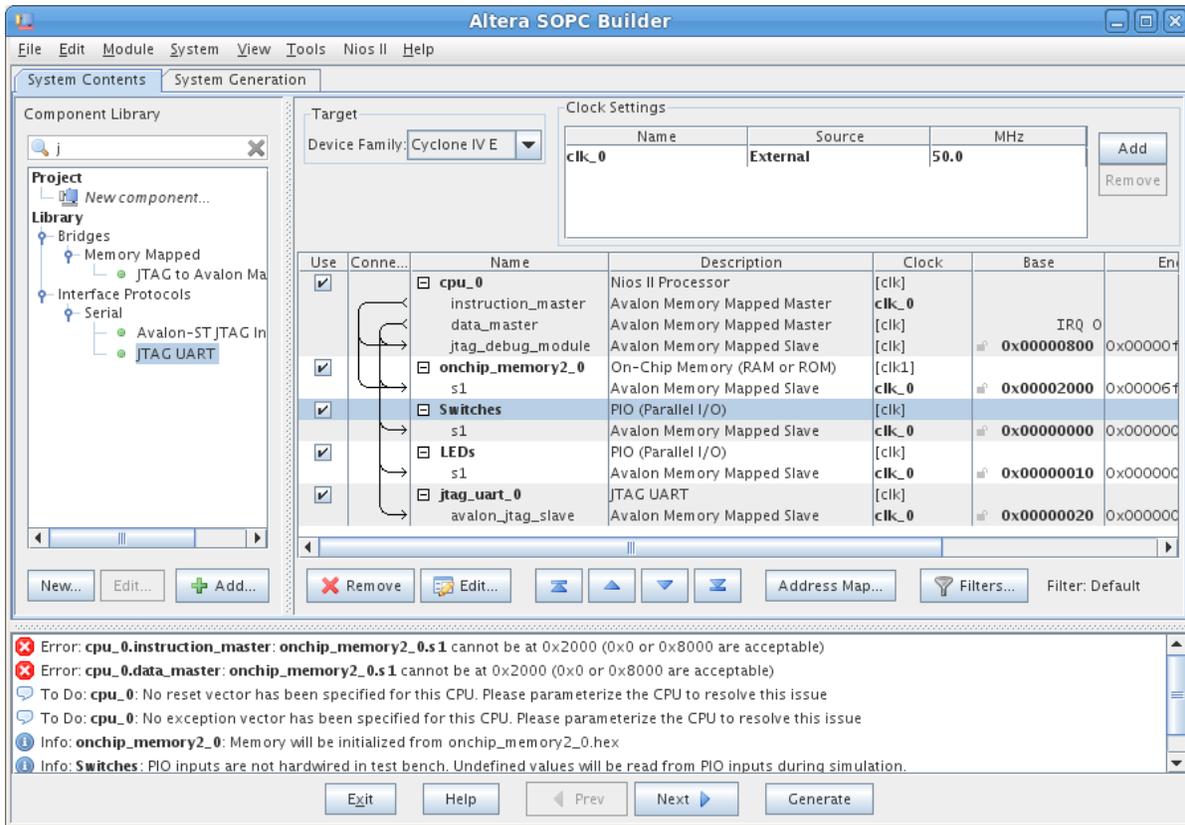


Figure 14. The complete nios- system.

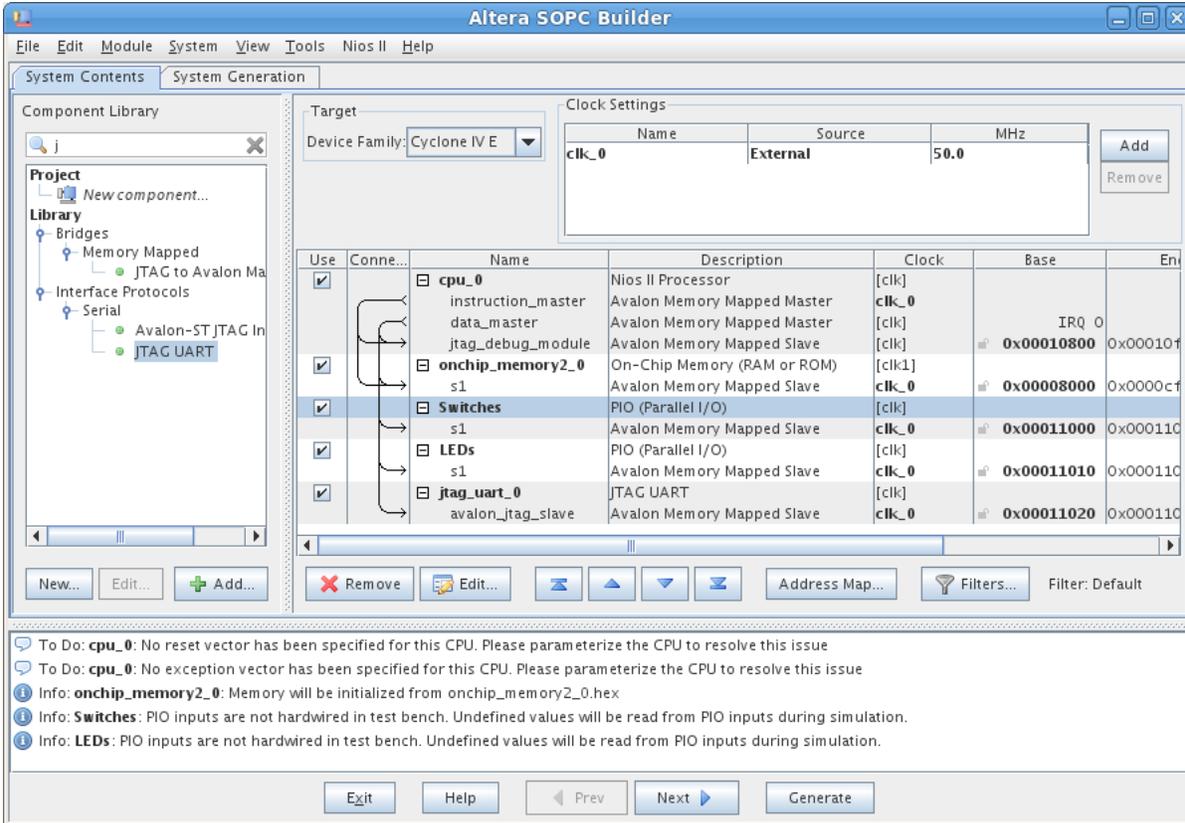


Figure 15. The final system specification.

12. Having specified all the components needed to implement the desired embedded system, it can now be generated. Click on the button Generate. At this time, you may be prompted to save the system if you haven't done so. Save the system as nios_system. Afterwards, this would lead to the window of Figure 16. Leave Simulation – Create simulator project files unchecked, because in this lab we will not deal with the simulation of hardware. The generation process produces the messages displayed in the figure. When the message "SUCCESS: SYSTEM GENERATION COMPLETED" appears, click Exit. This returns to the main Quartus II window.

Changes to the designed system can be easily made at any time by reopening the SOPC Builder tool. Any component in the System Contents tab of the SOPC Builder can be selected and deleted, or a new component can be added and the system regenerated.

What to Hand In

Demonstrate your implementation and successful system generation to your instructor for marking purposes. This tutorial (along with part II) should be completed by week 5 to receive full marks.

References:

1. Introduction to the Altera SOPC Builder Using VHDL Design
2. Nios II Hardware Development Tutorial

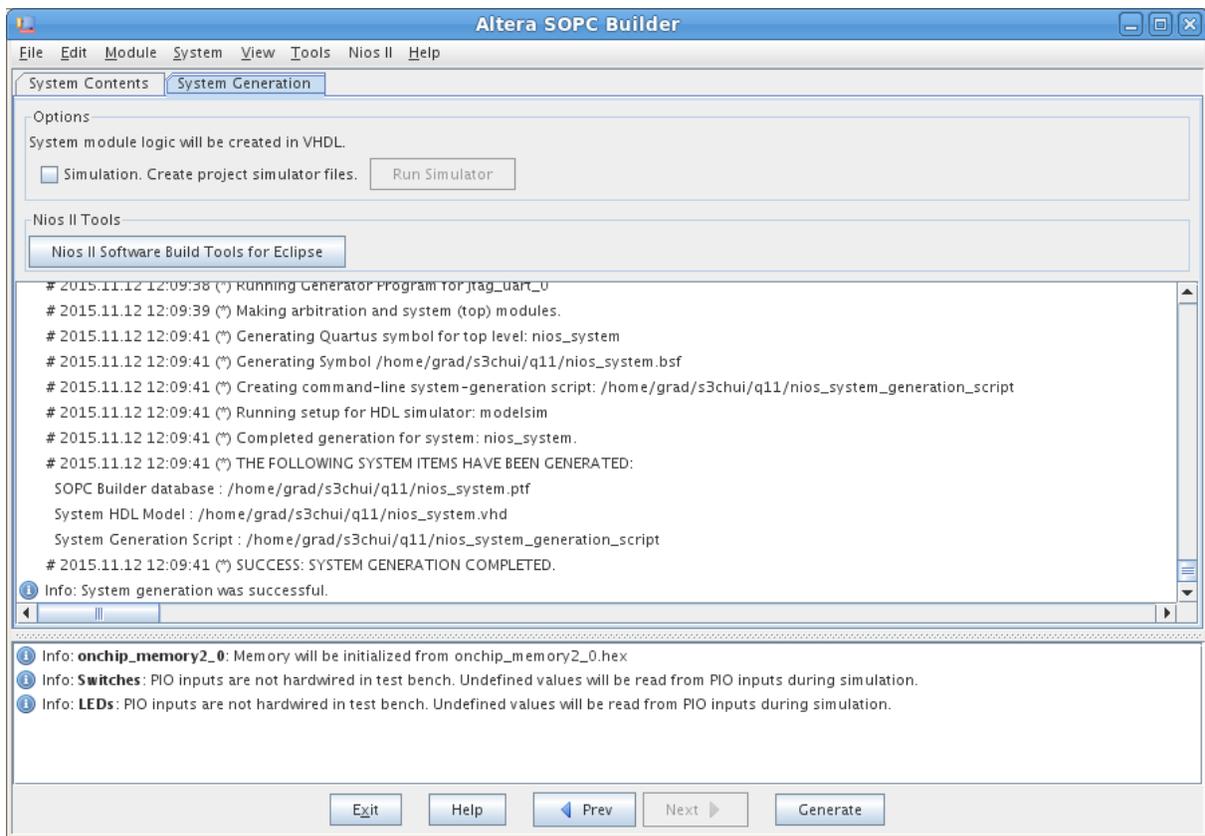
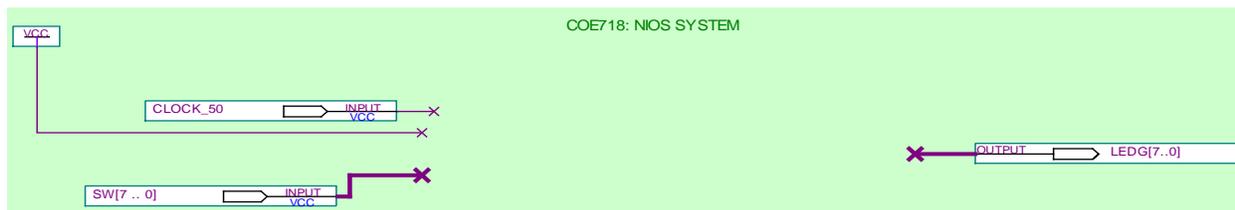
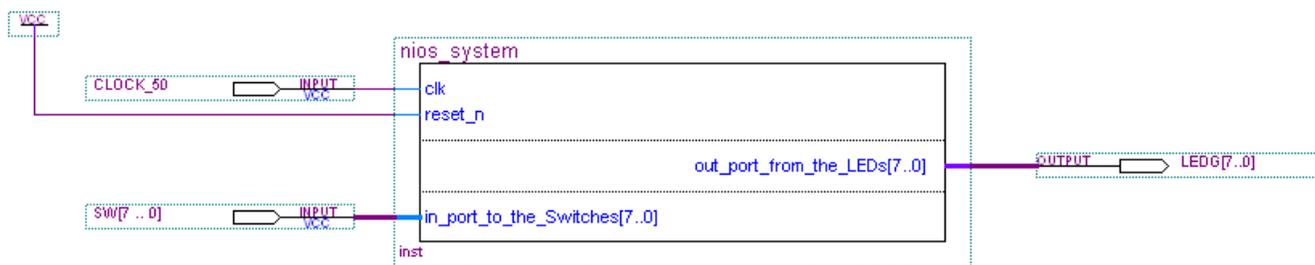


Figure 16. Generation of the Nios system.

Appendix



Empty BDF for Nios_System



Final BDF for Nios_System