NIOS CPU Based Embedded Computer System on Programmable Chip

EE8205: Embedded Computer Systems

NIOS-II SoPC: PART-II

1 Introduction

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. This part of the Lab introduces you *nios_system* development and design flow for its application software. This lab-part program/configure a Nios II CPU based system on an FPGA and create a software program to run on the Nios II system. The simple Nios II based embedded system that is implemented in Part-I is shown in Figure 1.

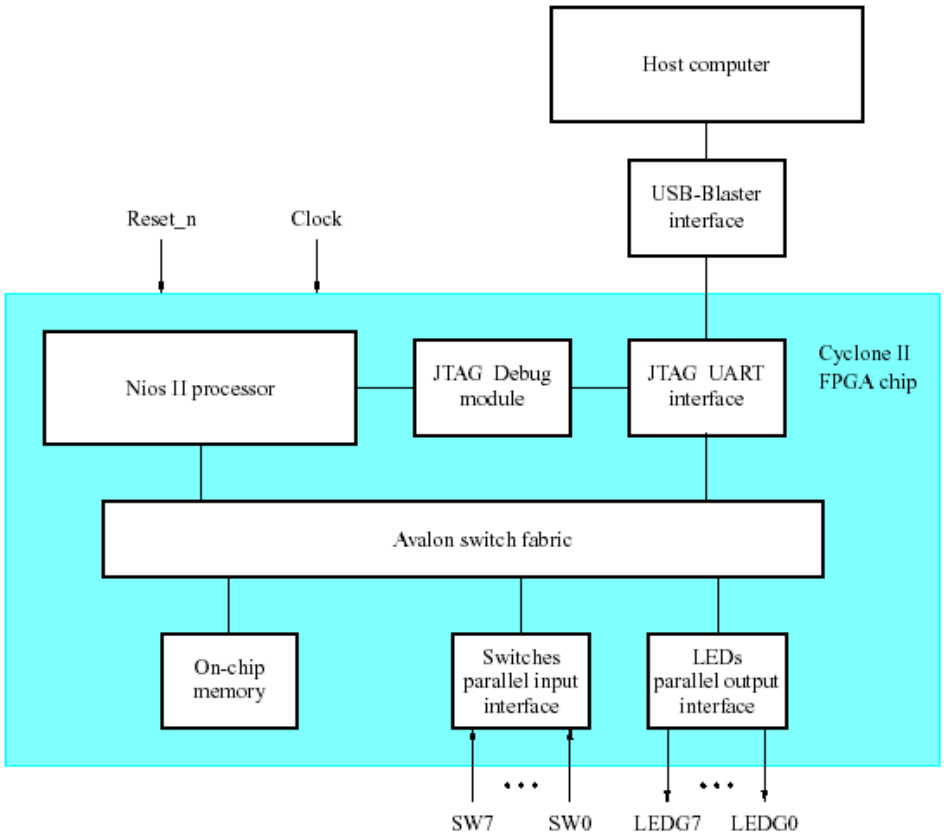

Figure 1: A Simple Nios II based Embedded Computer System.

Our example system of Figure 1 realizes a trivial task. Eight toggle switches on the DE2-115 board, SW[7..0] are used to turn on or off the eight green LEDs, LEDG[7..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 have used the SOPC Builder

to design the hardware depicted in Figure 1. 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.

2. Integration of the Nios II System into a Quartus II Project

To complete the hardware design, we need to perform the following steps:
In this section you will perform the following steps to complete the hardware design:

- Instantiate the SOPC Builder nios-system module in the Quartus II project.
- Assign FPGA pins.
- Compile the Quartus II project

2.1 Instantiation of the Module Generated by the SOPC Builder

The instantiation of the generated module depends on the design entry method chosen for the overall Quartus II project. We have chosen to use BDF (schematic), but the approach is similar for both Verilog and VHDL entry methods. Normally, the Nios II module is likely to be a part of a larger design. However, in the case of our simple example there is no other circuitry needed. All we need to do is instantiate the Nios II system in our top-level BDF file, and connect inputs and outputs of the parallel I/O ports, as well as the clock and reset inputs, to the appropriate pins on the Cyclone IV-E device. You can use the top level BDF file "lights.bdf" provided in lab3 directory of the course web page. At this stage, you make a note of the base addresses for SW and LED PIO of the nios_system you have built with the SOPC builder. We will use these addresses to read from switches and write to LEDs in the C/C++ code for Nios II CPU at the end of this lab.

You will instantiate a system module symbol **nios_system** into the BDF**.** To instantiate the system module in the BDF, perform the following steps:

1. Double click in the empty space between the input and output pins. The Symbol dialog box appears.
2. Under Libraries:, expand Project.
3. Click nios_system. The symbol dialog box displays the nios_system symbol.
4. Click OK. You return to the BDF. The nios_system symbol tracks with your mouse pointer.
5. Connect the inputs on the symbol with the wires on the left-hand side of the BDF.
6. Click the left mouse button to drop the symbol in place. Figure 2 shows the complete BDF using the LED and SW pins.

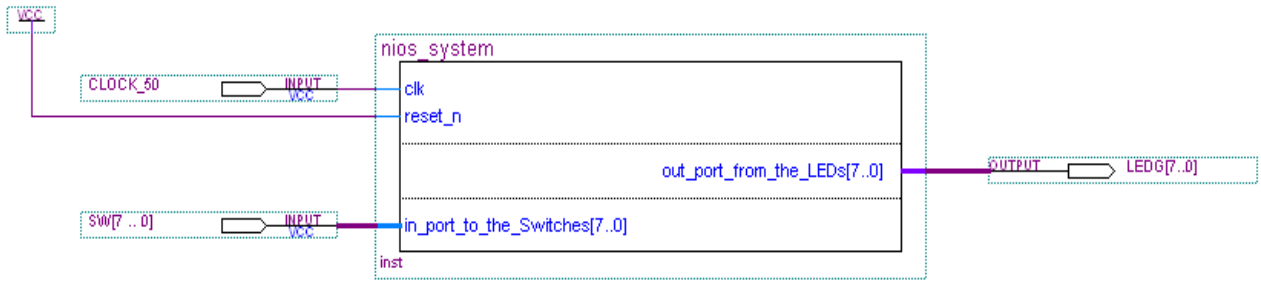

Figure 2: The Example Nios II System with I/O

2.2 PIN Assignments

There are two options for PIN assignments for DE2-115 board based projects. The 1st (and easiest) option is to import the DE2-115 pin assignment file "*DE2_115_pin_assignment.qsf*" by using the Assignments > Import Assignment. The assignment file "*DE2_115_pin_assignment.qsf*" is available at the course web in the /ee8205/project/sopc directory. Note that by importing this file you will only use 17 pin assignments all the other assignments will appear with question marks.

As a 2nd option, you may manually select the pins needed by the FPGA. To do this you will need to select the Start Analysis and Synthesis option. Then choose Pins in the Assignment menu of Quartus II Assignment Editor. You can sort the pin assignments by name by clicking the **To** column in the Assignment Editor as shown in Figure 3. Scroll down until the pin LEDG[0]–LEDG[7] appears in the Assignment Editor. Double click in the location cell for pin LEDG[0] and select the appropriate FPGA pin. The FPGA pins can be copied from the excel file "*DE2_115_pin_assignment.qsf*" in the ../project/sopc/ directory. You can also copy and paste multiple pin assignments. Similarly SW[0]–SW[7] and CLOCK_50 pins can be found for assignment. Finally save the pin assignment by selecting save in the File menu and close the assignment editor.

After proper pin assignments, they should appear in your BDF file that is opened in Quartus II.

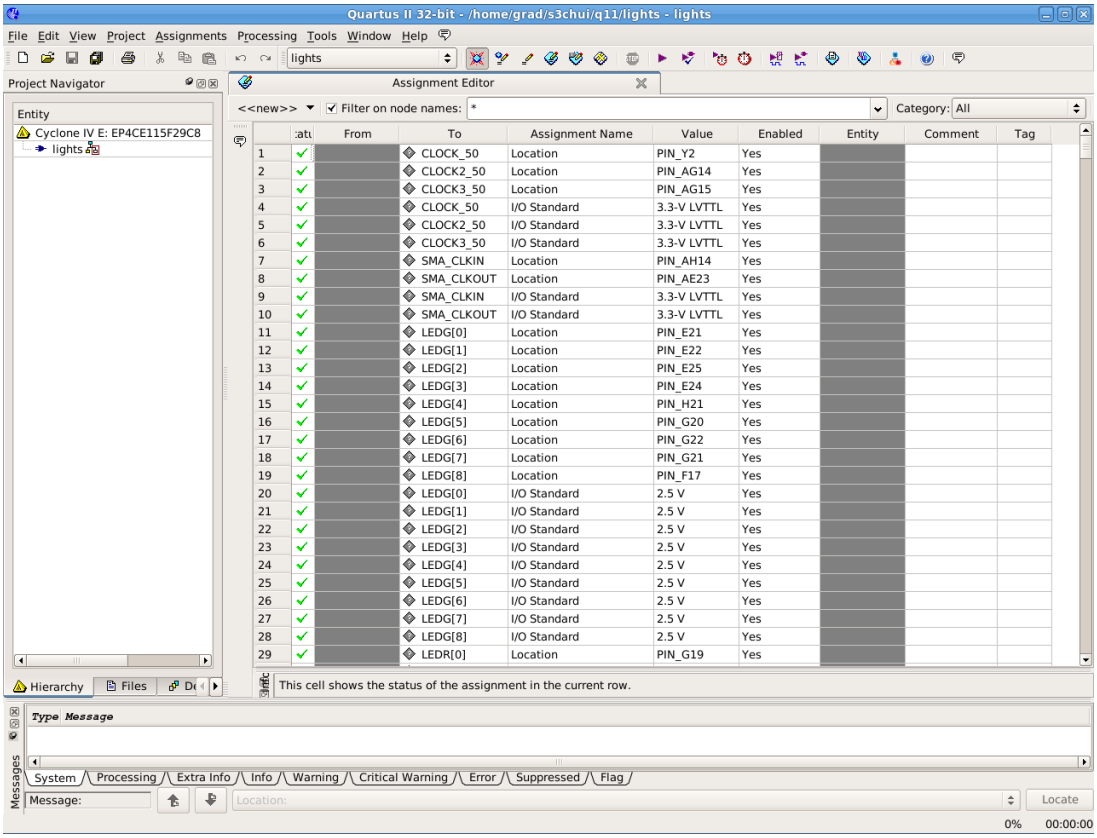

Figure 3: Assigning Pins with the Quartus II Assignment Editor

2.3 Compile the Quartus II Project

At this point you are ready to compile the Quartus II project and verify that the resulting design meets timing requirements. You must compile the hardware design to create an FGPA configuration file that you can download to the board. After the compilation completes, you may like to analyze the timing performance of FPGA design to verify that the design will work in hardware.
Perform the following steps:

1. On the Processing menu, click **Start Compilation**

2. The Quartus II Status utility window displays progress. The compilation process can take several minutes. When compilation completes, a dialog box displays the message "Full compilation was successful."
3. Click **OK**. The Quartus II software displays the Compilation Report window.

You have finished integrating the Nios II system into the Quartus II project. You are ready to download the FPGA configuration file to the target board.

2.4 Download Hardware Design to Target FPGA

Program and configure the Cyclone IV-E FPGA in the JTAG programming mode as following:

1. Connect the DE2-115 board to the host computer by means of a USB cable plugged into the USB-Blaster port. Turn on the power to the DE2-115 board. Ensure that the RUN/PROG switch is in the RUN position.
2. Select Tools > Programmer to reach the window in Figure 4.
3. If not already chosen by default, select JTAG in the Mode box. Also, if the USB-Blaster is not chosen by default, press the Hardware Setup.. button and select the USB-Blaster 3/2 depending on your computer system hardware in the window that pops up.
4. The configuration file *lights.sof* should be listed in the window. If the file is not already listed, then click Add File and select it.
5. Click the box under Program/Configure to select this action.
6. At this point the window settings should appear as indicated in Figure 4. Press Start to configure the FPGA.

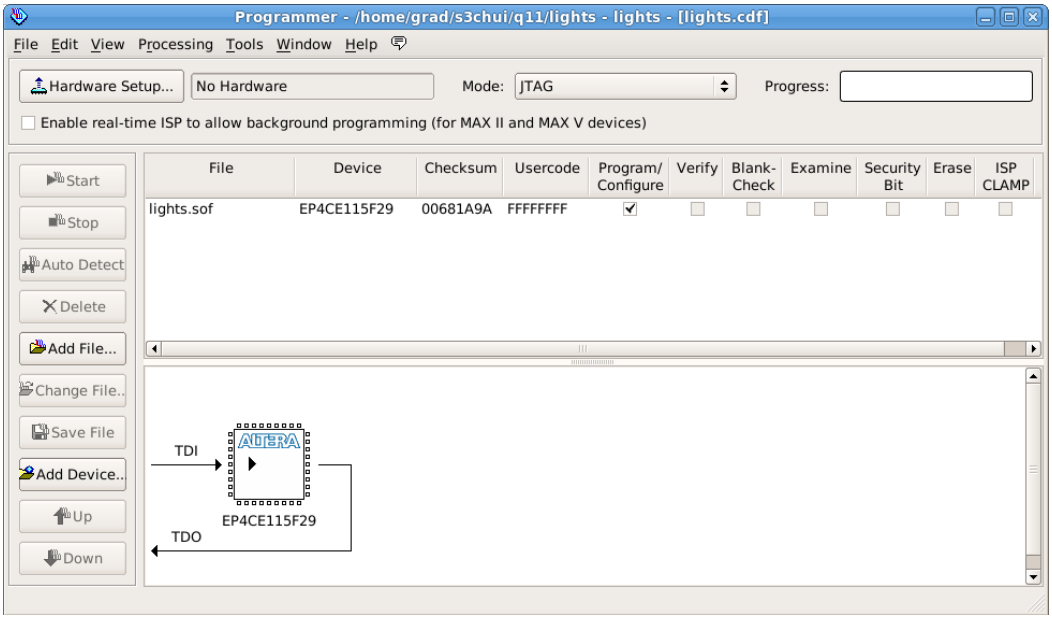

Figure 4. The Programmer window.

3. Development and Execution of the Application Program

Having configured the required hardware in the FPGA device, it is now necessary to create and execute an application program that performs the desired operation of reading from switches and writing to LEDs. This can be done by writing the required program either in the Nios II assembly language or in a high-level language such as C. We will illustrate the C/C++ programming approach. The parallel I/O interfaces (SW and LEDs) generated by the SOPC Builder is accessible by means of registers in the interface.

Depending on how the PIO is configured, there may be as many as four registers. One of these registers is called the Data register. In a PIO configured as an input interface, the data read from the Data register is the data currently present on the PIO input lines. In a PIO configured as an output interface, the data written (by the Nios II processor) into the Data register drives the PIO output lines. If a PIO is configured as a bi-directional interface, then the PIO inputs and outputs use the same physical lines. In this case there is a Data Direction register included, which determines the direction of the input/output transfer. In our unidirectional PIOs, it is only necessary to have the Data register. The addresses assigned by the SOPC Builder in the example are 0x00011000 for the Data register in the PIO called Switches and 0x00011010 for the Data register in the PIO called LED. You should check the base register addresses in your SOPC builder window shown in Figure 5 and must use your SOPC builder window addresses for the *nios_system*. You can also find the full description of the PIO interface by opening the SOPC Builder window in Figure 5 and right-clicking on the module name of a PIO (either Switches or LEDs). Then, in the pop-up box select Data Sheet to open the document *PIO Core with Avalon Interface* that gives a full description of the interface. To use this facility you may need to be connected to the Internet.

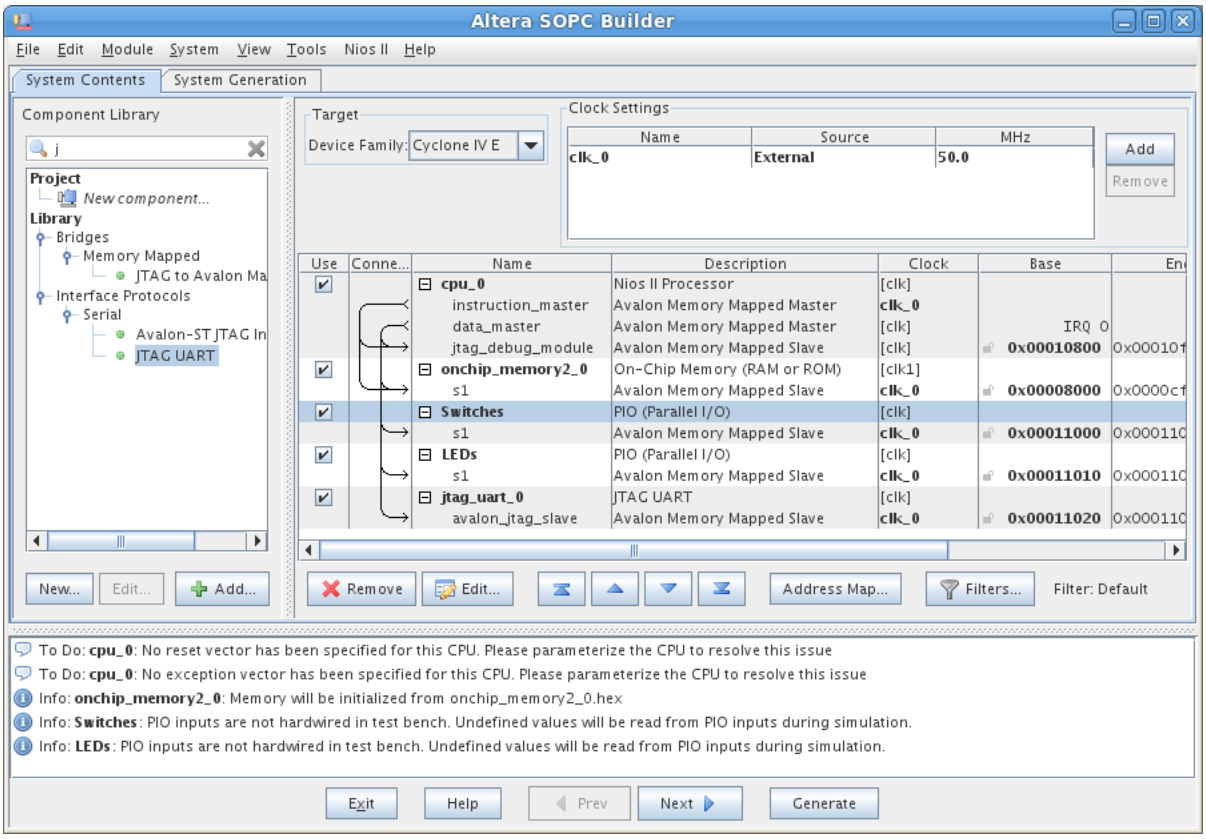

Figure 5. Completed Nios_System.

3.1 Create a New C/C++ Application Project

In this section you will start the Nios II integrated development environment (IDE) and compile a simple C/C++ language program. This section presents only the most basic software development steps to demonstrate software running on the hardware system you created in previous sections. You will create a new Nios II C/C++ Application Project. First of all start the Nios II IDE. Click **Run Nios II Software Build Tools for Eclipse**. Tab in the SOPC builder window already opened by you and shown in Figure 6. When the **Workspace Launcher** dialog box appears, click **OK** to accept the default workspace location or you can specify any other empty directory.

Perform the following steps for the application software development.

1. On the File menu, point to **New**, and then click **Nios II Application and BSP from Template** to create a new project.
2. In the Target hardware information section where it says SOPC Information File name, click on the **…** button. A file window should pop up, choose the file **nios_system.sopcinfo**. Check that **cpu_0** should be selected as the CPU Name. This corresponds to our Nios II CPU.
3. Give the project the name **lights**. Note: This name does not have to match the Quartus II project name.
4. Click **Finish**.

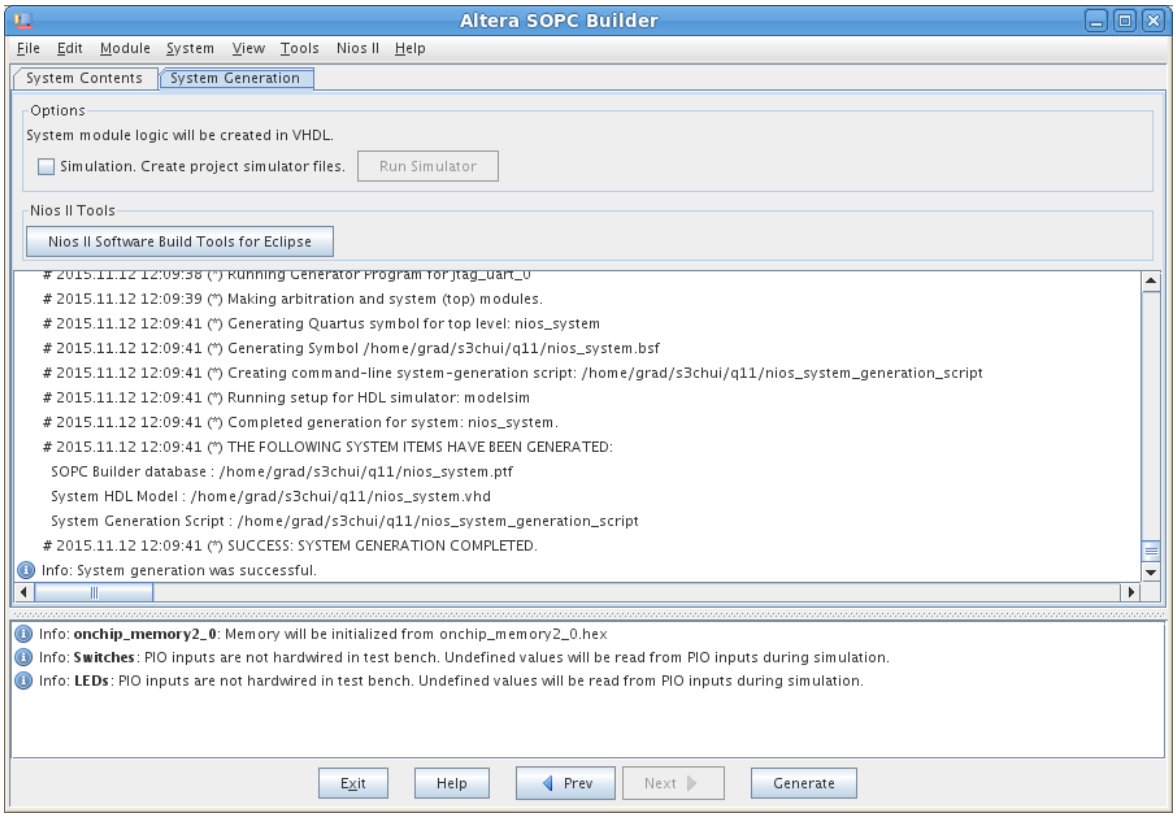

Figure 6. Generated Nios_System.

We get a blank window for a C application code to be typed. The Nios II IDE displays two new projects in the C/C++ Projects view on the left-hand side of the workbench: **lights** and **lights_bsp**. **lights** is your C/C++ application project, and **lights_bsp** is a system library that encapsulates the details of the Nios II system hardware. The left-hand pane of the IDE workbench displays the Project Explorer with projects. Create the source file by right Click on the project selected New > Source File and name it lights.c. Type in your code given in Figure 8 or use copy and paste from the “lights.c” file provided in bonus lab directory.

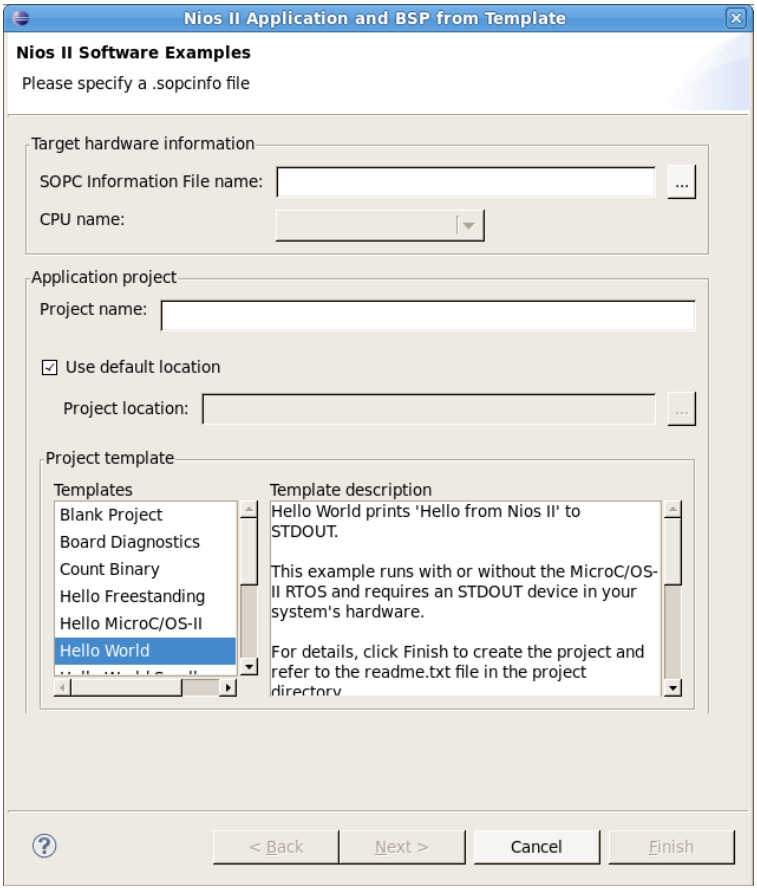

Figure 7. NIOS II IDE New Project Wizard

```
#include <stdio.h>
int main()
{
  unsigned char * Switches;
  unsigned char * LEDs;

  Switches = (unsigned char *)0x0011000;
  LEDs = (unsigned char *)0x00011010;

  printf("Beginning.\n");

  while(1){
    *LEDs = *Switches;
  }

  return 0;
}
```

Figure 8. C language code to control the lights

You can make any edits or type in any **C/C++** program file in the IDE and repeat the compile and run steps described next.

3.2. Compile the Project

In this section you will compile the project to produce an executable software image. For the example lab/tutorial design, you must first adjust the project settings to minimize the memory footprint of the software, because your Nios II hardware system contains only 20 Kbytes of memory.
Perform the following steps:

1. Right-click **lights_bsp** and click **Properties.**
2. The project properties windown will pop up. Select **Nios II BSP Properties.**
3. The **Nios II BSP Properties** page contains all settings related to how the program interacts with the underlying hardware. Therefore, the settings here reflect names you specified when creating the Nios II hardware in section "Define the System in SOPC Builder".
4. Change the following settings, which affect the size of the compiled executable (see Figure 9).
   a. Turn on **Reduce device drivers**.
   b. Turn on **Small C library**.
5. Click **OK** to close the **Properties** dialog box and return to the IDE workbench.
6. Then Build the project

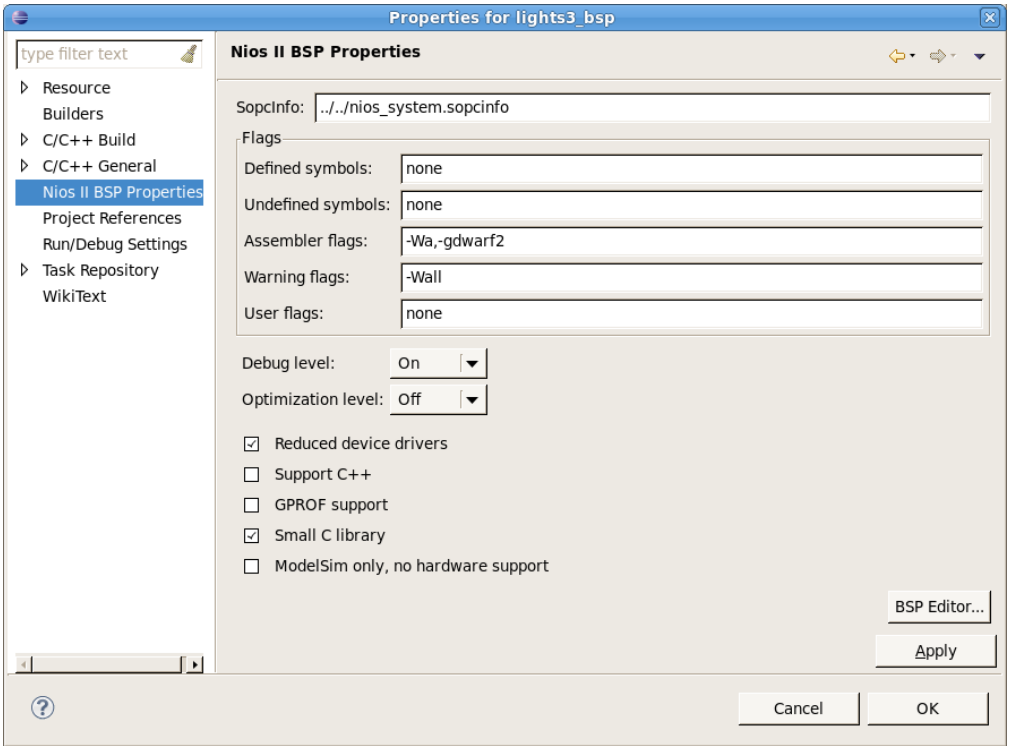

Figure 9. NIOS II IDE New Project Wizard

3.3. Run the Program

To download the software executable to the DE2-115 board, perform the following steps:

1. Right-click the **lights** project, point to **Run As**, and then click **Nios II Hardware**. The IDE downloads the program to the FPGA on the target board and starts execution.
   When the target hardware starts executing the program, the Console view displays the character I/O output “Beginning”. If you connected LEDs to the Nios II system during "Integrate the SOPC Builder System into Quartus II Project", then the LEDs show the switch positions.

2. Click **Terminate** (the red square) on the toolbar at the upper-right hand corner of the Console view to terminate the run session. When you click **Terminate**, the IDE disconnects from the target hardware and leaves the Nios II processor running.

4. What to Hand In

Demonstrate your implementation and successful program running on the NIOS II CPU to the instructor for marking purposes. This part of the lab should be completed by week 5 (along with part I) to receive full marks.

References:

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

Appendix: BDFs for Nios_System