Mechtron 3TB4: Embedded Systems Design II
Tutorial Lab 4
Hardware Software Co-design

Note: Connect to the VPN prior to starting Quartus Prime, otherwise compilation will be unavailable.

Goals

In this tutorial you will learn how to build a System On a Programmable Chip (SOPC), and to develop software that runs on this system. You will use Quartus Prime to integrate the hardware and the software components of the system. The system will be implemented on the DE1-SoC board. The hardware part of the system consists of:

- Nios II processor
- On-chip memory, which consists of the memory blocks in the FPGA chip; we will specify a 32K byte memory arranged in 32-bit words
- Two parallel I/O interfaces
- JTAG UART interface for communication with the host computer

You could write your own Verilog modules to describe the hardware components, however Altera provides a library of components that can be used to build the desired system. Also, Altera provides a tool named "Platform Designer" ("Qsys" for old versions of Quartus) that allows you to select different components from the library, make required connections between the selected components and generate an HDL module for the system which can be instantiated in a higher level module using Quartus Prime. This higher level module can then be compiled and loaded into an FPGA for subsequent use by the software that is part of the co-design. The software part needs an IDE that can compile the C code into NIOS II instructions and download it into the on-chip memory of the hardware system designed by Platform Designer and loaded into the FPGA as mentioned above. We use the `Nios II Software Build Tools (SBT)` for `Eclipse` for this purpose. A simple C program will be used to make eight LEDs blink every second. You will also learn how to use a debugging tool provided by Altera named "Signal Tap Logic Analyzer".

NOTE: The Platform Designer tutorial material used in this lab is for Quartus 17.0. From Version 17.1, Altera Qsys is replaced by Intel Platform Designer. Therefore, when using Quartus version 17.1,which is installed on the lab computers, whenever the tutorial asks to use Qsys, we will use the tool Platform Designer.

Sections 1 to 3 are worth 5 marks each.

1 Platform Designer System Integration Tool

For an introduction to Platform Designer and to build the hardware described above, we will use the tutorial material, "Introduction to the Qsys System Integration Tool", provided by Altera. You may find this tutorial in the file named `"Introduction to the Qsys Tool.pdf"` on the course web page (it can also be downloaded from Altera's web site).

Please note that:

- As you work through the tutorial, error messages will appear in the Platform Designer's message window. Please ignore them. If you follow the instruction steps properly, these errors will eventually

disappear. If there are still errors at the end of Section 4, before system generation at step 15, you should check your work to correct any mistakes.

- In Step 5 of Section 4, the Altera tutorial sets the on-chip memory size to 4K bytes. For our lab, we will use a size of 128K bytes. Please do not forget to change the on-chip memory size to 128K bytes. By using 128K bytes of on-chip memory for this tutorial and for this lab project, you do not have to use the small C libary to use the printf() function (see `"Tutorial material for NIOS II SBT for Eclipse.pdf"`).

- If you followed the naming convention for the tutorial, at the end of Section 4, a Verilog file named `nios_system.v` will be created and be saved in the folder: `qsys tutorial/nios_system/synthesis`. You should have created a `lights.v` file in Section 5. The file `lights.v` will be the Top-Level Entity file and will use appropriate pins on the DE1-SoC boards. The module `nios_system.v` will be instantiated in the module `lights.v`

- Before compiling the project, please remember to add the the `.qip` file, which is generated by Platform Designer and is saved in the folder `qsys_tutorial/nios_system/Synthesis/`, to the project. Adding the `.qip` file to the project will actually add a group of newly generated files to the project. The Verilog file `nios_system.v` is one file among them. To view the names of the files included in the `.qip` file, click the ">" sign before the `.qip` file name in the Files tab of `Quartus' Project Navigator`.

  When you add files, do not forget to press the "Apply" and then the "Ok" buttons after the "Add" button, otherwise a file may be inserted into the project for the current session only and you will have to re-insert them for another session. Also please remember to import the pin assignment file, `DE1_SoC.qsf`, as done in previous labs.

  To make it easy to edit the NIOS system later, you can also consider adding the .qsys file, which is under the project folder, to the project.

- Compile the project (as described in "`Introduction to the Qsys Tool.pdf`"). Show the pop-up window that notifies a successful compilation of the project to one of the TAs and take a screen shot for submission as part of your pre-lab report. Load the project onto the DE1-SoC board before continuing to the next section.

2 Nios II Software Build Tool (SBT) for Eclipse

After the compiled Quartus/Platform Designer project is downloaded to the DE1-SoC board, the required hardware is configured in the FPGA device. Now you can create and execute application programs to perform desired operations. This can be done by writing programs either in the Nios II assembly language or in other high level programming languages. In our lab, we will use embedded C.

We use the Nios II Software Build Tool for Eclipse as the IDE to develop C applications for the SOPC system that we have implemented on the DE1-SoC FPGA device. A separate (brief) tutorial material for how to use Nios II SBT for Eclipse can be found at the course web page in
`"Tutorial material for NIOSII SBT for Eclipse.pdf"`. **Please work through the tutorial material**.

In the tutorial there is a sample C program for making eight LEDs blink every second. The sample program uses a constant LEDS_BASE, which is defined in `system.h`, as the base address for the eight LEDs. The C application toggles the value at the address of LEDS_BASE to make the LEDs blink. You

must add code to print a welcome message from your group in the Nios console as shown in Figure 1. Show the blinking LEDs to one of the TAs and take a screen shot of the Nios Console output for submission as part of your pre-lab report.

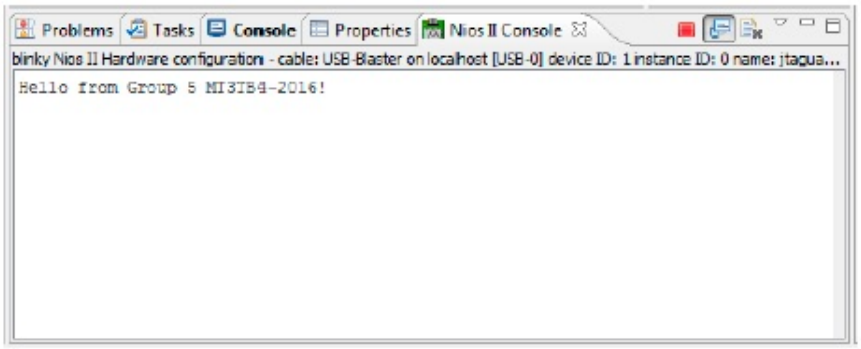

Figure 1: Sample Console Output

If you want to use switches to control the LEDs (as described in `"Introduction to the Qsys Tool.pdf"` tutorial), you need to find the base address constant defined in `system.h` for the switches, and assign the value at this address to the base address of the LEDs. The file `system.h` is a header file generated by the Nios II SBT, which can be found under the project's `_bsp` folder in the Nios II SBT project explorer window. For this tutorial project, if you need to use the base address of the switches, you must declare a volatile variable with type of char * and then assign the switches' base address to it. You can implement this as a separate exercise if you like.

Example:

```
volatile char * SW =(char *) SWITCHES_BASE;
char * LEDs=(char *) LEDS_BASE;
*LEDs=*SW;
```

3 Signal Tap Logic Analyzer

Signal Tap is a way to view what your chip is doing while it is running. It is essentially a logic analyzer (like an oscilloscope but for many digital signals at once, plus it is capable of triggering signal events and showing what happened at those points) built into the DE1-SoC board and the Quartus Prime software. You may find a quickstart for Signal Tap in the tutorial file `SignalTap.pdf`, located on the course web page. Note that this tutorial requires implementation of a different circuit on the DE1-SoC board which can be done as an exercise separate from the current tutorial.

This document will only cover what you need to know to complete this section. You will be analyzing the board while you are running your blink LEDs program from the Eclipse tutorial.

1. Open the Platform Designer tutorial project (the `lights` project) if it is not open.
2. Create a new signal tap file by choosing from the Quartus Prime menu: `Tools | Signal Tap Logic Analyzer`, (in previous versions of Quartus, the name is "SignalTap II Logic Analyzer") or by choosing from the Quartus Prime menu : `File | new` to create a new `Signal Tap Logic Analyzer File`.
3. At the top right of the Signal Tap Logic Analyser window, for the JTAG `Chain Configuration`, set the "Hardware" to DE-SoC. Make sure that the correct device is selected. When setting up the hardware for the JTAG Chain Configuration, ensure that the DE1-SoC board is on before pressing the `"Setup..."` button to set up. Now you should be ready to add nodes to the current instance.

4. In the `Signal Configuration` area, set the clock to CLOCK_50.

5. Double click in the main window to open the `Node Finder`. Set the `Filter to Signal Tap:pre-synthesis`. If the Filter input box is not shown, click the button with two down arrows beside the button List to expand the node finder window. Click the button List, then in the `Matching Nodes` window, select and add the following nodes:

```
nios_system:NIOSII|nios_system_LEDs:leds|address
nios_system:NIOSII|nios_system_LEDs:leds|chipselect
nios_system:NIOSII|nios_system_LEDs:leds|data_out
nios_system:NIOSII|nios_system_LEDs:leds|write_n
nios_system:NIOSII|nios_system_LEDs:leds|writedata
nios_system:NIOSII|nios_system_LEDs:leds|reset_n
```

6. Enable triggering for the signal chipselect, and set the trigger condition to high. Your setup window will look similar to that shown in Figure 2.

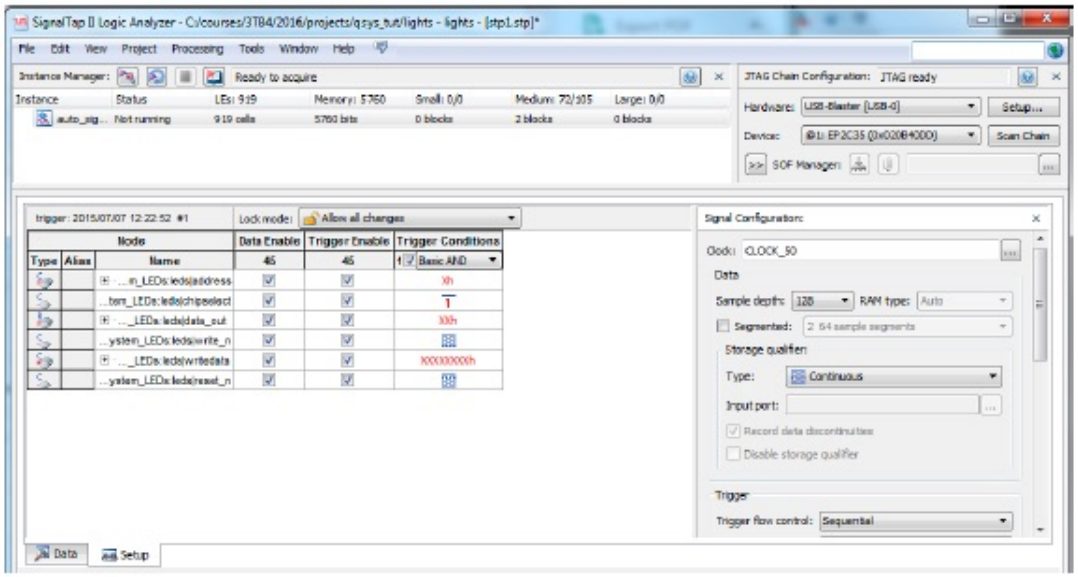

Figure 2: SignalTap Setup

7. Save the Signal Tap file and recompile your Quartus project.

8. Load the new output file onto the chip. You may need to stop the running NIOS SBT program to make this loading process successful, or you may need to re-power on the DE1-SoC board to load the new output file, because the DE1-SoC board may be busy running the NIOS SBT program.

9. Load your NIOS SBT C blinky project onto the DE1-SoC board. You may need to regenerate the bsp library before executing `Run As | Nios II Hardware`. Every time the Quartus project is modified and a new output file is loaded to the DE1-SoC board, the bsp library needs to be regenerated for the application program. To re-generate the bsp library, in the Project Explorer window in Nios II SBT, right click the _bsp folder, then select `Nios II | Generate BSP`.

10. In the Signal Tap window, click the `Autorun Analysis` button or the `Run Analysis` button. Alternatively, from the `Processing` menu, select `Autorun Analysis` or `Run Analysis`. Observe the signal analyzer result.

    Note the changing values of **data_out** and **writedata**. Can you explain this behaviour?

Show the output to one of the TAs and submit your response along with screenshots showing different values similar to Figures 3 and 4.

4 Summary and Overall Understanding

Answer the following questions and submit your responses along with the answers to other questions and screenshots described in the tutorial, as part of your pre-lab report for lab4:

1. Which of the devices in this system were master devices and slave devices?
2. Which of the following components used in this tutorial were software and which were hardware? If you think a part belongs in both, explain why.
   - Quartus
   - Qsys (Platform Designer)
   - Nios SBT for Eclipse
   - Signal Tap Logic Analyzer

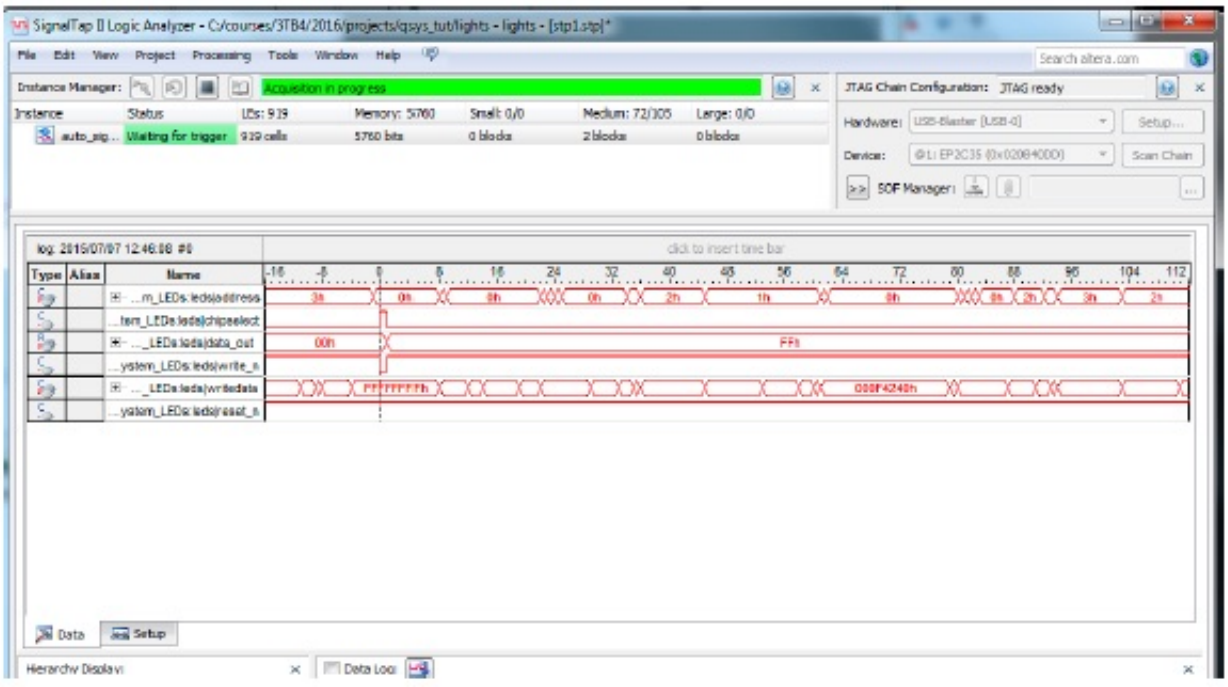

Figure 3: SignalTap Result Example 1

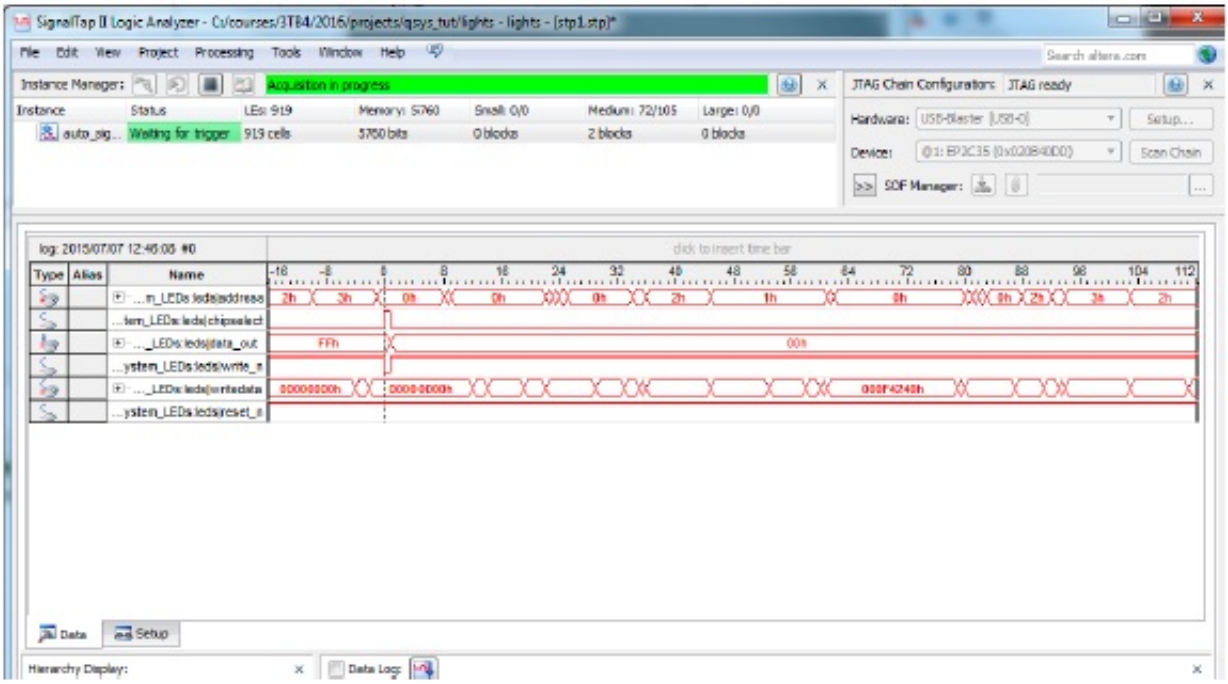

Figure 4: SignalTap Result Example 2