SE 2DA4, Pre-Lab5

Tutorial for Platform Designer, NIOS II SBT for Eclipse and Signal Tap Logic Analazer

1 Platform Designer System Integration Tool

For the first part of this tutorial, we will be using the tutorial material provided by Altera, `Introduction to the Qsys System Integration Tool`, which can be found in file "`Introduction_to_Qsys.pdf`" in the "`ref`" subdirectory of the `lab5.zip` once you have uncompressed the file. This tutorial will give you basic familiarity with the tools that you will need to do Lab 5.

The Qsys tutorial material is for Quartus 17.0. From Version 17.1, the Altera Qsys is replaced by the Intel Platform Designer. Therefore, when using Quartus version 17.1, whenever the tutorial asks to use Qsys, we will use the Platform Designer.

Complete Section 1-5.1.1 of the Qsys tutorial.

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 the 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 32K bytes. So please change the on-chip memory size to 32K.

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 folder `qsys_tutorial/nios_system/synthesis`. You should have created a file `lights.v` 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 module `light.v`

Before compiling the project, please remember to add the the `.qip` file, which is generated by Platform Designer and is saved in 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 the Project Navigator in Quartus.

When adding files, do not forget to press the button"Apply" then "Ok" after the button "Add", 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.

To make it easy to edit the NIOS system later, users can also consider adding the `.qsys` file to the project. The `.qsys` file is in the project folder.

Please also remember to import the pin assignment file, `DE1_SoC.qsf`, for assigning pins for the Quartus project. `DE1_SoC.qsf` is in the `data` subdirectory of the `lab5.zip` file.

Compile and 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 we can create and execute application programs to perform desired operations. This can be done by writing programs either in NIOS II assembly language or in other high level programming languages. In our lab, we will use embedded C.

In our lab, we will use the NIOS II Software Build Tool for Eclipse as the IDE to develop C applications for the SOPC system 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 in the `ref` subdirectory of `lab5.zip`, in "`TutMatfNIOSIISBTfEclipse.pdf`". Please work through the tutorial material.

In the tutorial for the NIOS II Software Build Tool for Eclipse, there is a sample C program for making eight LEDs blink every second (Figure 6 of "`TutMatfNIOSIISBTfEclipse.pdf`") . The sample program uses a constant LEDS_BASE, which is defined in file `system.h`, as the base address for eight LEDs. The C application toggles the value at the address of LEDS_BASE to make the LEDs blink. If you need to use switches to control the LEDs, 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 NIOS II SBT, and it 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.

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 capable of triggering signal events and showing what happened at those points) built into the DE1-SoC board and Quartus software. You can find a quickstart for Signal Tap in the tutorial file "`SignalTap.pdf`", located in the `ref` subdirectory of `lab5.zip`.

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 Qsys tutorial project (the "`lights`" project) if it is not open.
2. Create a new Signal Tap file by choosing from the Quartus menu: `Tools | Signal Tap Logic Analyzer`, or by choosing from the Quartus menu : `File | new` to create a new `Signal Tap Logic Analyzer File`.

3. On the right hand side of the `Signal Tap Logic Analyzer` window, in the `Signal Configuration` area, set the clock to `CLOCK_50`.

4. On the top right of the window in `JTAG Chain Configuration` area, set the Hardware to `DE-SoC`.

5. Double click the blank area 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 `List` button, then in the `Matching Nodes` window, select and add the following signals:

   - 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_syste_LEDs:leds|reset_n

6. Enable triggering for the signal `chipselect`, and set the trigger condition to high.

7. Save the Signal Tap file and enable Signal Tap for the project.

8. Recompile your Quartus project and load the new output file onto the chip. You need to stop any running NIOS SBT program before loading a new Quartus project to the FPGA chip. You may need to re-power on the DE1-SoC board to load the new output file if the DE1-SoC board is busy running a 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`". To regenerate the bsp library, in the `Project Explorer` window in NIOS II SBT, right click the `_bsp` folder, then select "`Nios II | Generate BSP`". Each time your hardware configuration (Quartus/Platform Designer project) is changed, you need to regenerate your BSP library in the NIOS II SBT for Eclipse to run your application program.

10. In the Signal Tap Logic Analyzer window, click the "`Autorun Analysis`" button or the "`Run Analysis`" button.

11. On the "`Data`" tab in the Signal Tap Logic Analyzer window, observe the signal analyzer result.

    - What are the values written to the LEDs?
    - Sketch **write_n, data_out,** and **chipselect** signals (copy from the Signal Tap data tab)