Chapter 2

Nexys2 FPGA Board

The purpose of this laboratory is to introduce you to the Digilent Nexys2 FPGA board. We will use these tools to synthesize our VHDL circuits onto FPGAs used within the lab.

Learning Objectives

After completing this laboratory, you should:

- Understand the basic architecture of the Nexys2 board and be able to determine the FPGA pin numbers of specific I/O signals on the board
- Design a simple circuit based on VHDL concurrent statements using the buttons, switches, and LEDs
- Run the Xilinx XST tool to synthesize a working circuit that runs on this board.

Exercises

Exercise 1 – Nexsys2 FPGA Board

Each station within the laboratory is equipped with a Digilent Nexys2 FPGA board. You will use these boards for all of the laboratory assignments for this course. It is important that you become familiar with this board and the pins on the FPGA that are attached to the board resources. For this lab, you will create a simple FPGA circuit that uses the switches and LEDs of the board. You will also synthesize your circuit and download it to the Nexys2 FPGA board. To learn more about the Nexsys2 board, download the Nexys2 reference manual for the board we are using in the laboratory. This manual is very important and you will be accessing it regularly throughout the semester.

Download: Nexys2_rm.pdf

Carefully read page 1 and the section titled "User I/O" on pages 4 and 5. Answer the following questions after reading this material:

Question: What is the default logic value of the push buttons when the push button is NOT being pressed (i.e., BTN0, BTN1, BTN2, or BTN3)?

Question: What digital logic level is needed to turn on one of the LEDs (i.e., LD0-LD7)?

Question: What is the purpose of the series resistor used by the buttons and switches?

Figure 8 of the reference manual lists the FPGA pin numbers of the I/O signals you will use in this lab. For example, the pin "B18" is the FPGA in hooked up to the input signal associated with BTN0. You will need to refer to this figure to identify all of the FPGA pins used in this lab.

Question: What FPGA pin is connected to the switch input SW4 signal?

Question: What FPGA pin is connected to the LED LD5 output signal?

You will need to identify the pin numbers of all of the switches, buttons, and LEDs to complete the lab. You may want to write them down now while you have this information in front of you. More details about the Nexys2 board can be found in the Nexys2 schematic. Download this document and review page 9.

Download: Nexys2_sch.pdf

This page provides a detailed circuit diagram of the switches, buttons, LEDs, and seven-segment display used on this board. Answer the following questions after reviewing this page of the schematic.

Question: What is the resistance, in Ohms, of the resistor used between the FPGA and the LEDs?

Question: If the voltage level of the FPGA pin is 3.3V (for a digital '1') and the forward bias voltage of the LED is 1.7V, how much current will flow through the LED when the FPGA pin is high asserted? (This is a simple application of Ohm's law: V=IR)

Exercise 2 – LED Design and Simulation

After completing the previous exercise, you have the skills necessary to create your own simple VHDL designs and download them onto the board. For this exercise, you will create your own VHDL architecture that uses the switches and buttons to drive the LEDs. Once you have created your VHDL entity and architecture, you will thoroughly simulate it.

Begin this exercise by creating a new project in Project Navigator. Consult Figure 1.2 from the previous laboratory assignment to see the required settings for a new project using the Nexys2 FPGA board. You may want to review some of the screencasts in the previous lab to remind you how to create a project and add a VHDL file to the project. Create a single VHDL entity named "**nexys2**" with the following ports (using this name of the entity and the following ports will simplify the use of a testbench in the following exercise):

Port Name	Direction	Width	Purpose
sw	Input	8	Switches input
btn	Input	4	Buttons input
led	Output	8	LED outputs

After creating your entity, create an architecture that meets the following specifications:

1. Create logic that displays the value of the slide switches on the LEDs when no button is pressed. In other words, LED0 should display the value of SW0, LED1 should display the value of SW1, etc.
2. When BTN0 is pressed, display the value of the slide switches *rotated* right one position on the LEDs. Use the concatenate operator (&) to create this rotated value (see page 59 of the text under the heading "concatenation operator" for more details).
3. When BTN1 is pressed, display the inverted value of the slide switches on the LEDs.
4. When BTN2 is pressed, display the value of the slide switches in a "swapped" order on the LEDs. When displaying the swapped value, LED7 should display SW0, LED6 should display SW1, etc.
5. When BTN3 is pressed, swap the upper and lower 4 bits of the switches and display them on the LEDs. In other words, LED7-LED4 should display the values of SW3-SW0 and LED3-LED0 should display the values of SW7-SW4.
6. Note that the buttons have priority: BTN0 has highest priority, followed by BTN1, BTN2, and BTN3.

The following conditional assignment statement could be used for this (see Section 4.2 on page 72 of the text):

```
led <=  <some logic> when btn(0) = '1' else
        <some logic> when btn(1) = '1' else
        <some logic> when btn(2) = '1' else
        <some logic> when btn(3) = '1' else
        <default logic when no buttons pressed>;
```

After creating your logic in VHDL, compile your VHDL file and remove all errors. After the errors have been removed, simulate your architecture with a few simple test cases to remove any obvious errors. Create a simple simulation .tcl file that automates the testing of a number of input conditions.

Upload: Upload your .tcl simulation file.

Once you have created your VHDL, compiled the VHDL, and removed most errors, you are ready to test your architecture with a testbench. Download the following testbench file and add it to your project (don't forget to tag its "Association" as simulation).

Download: nexys2_testbench.vhd

Use the testbench to identify and fix any problems with your architecture. Make sure you simulate your architecture until the "Done" message is printed by the testbench on the console. Keep your simulation handy, as you will need to demonstrate your simulation waveforms to the TA in order to pass off this lab.

Question: What time does the "Simulation Done" message occur when using this testbench?

In the simulation we can see the signals created by the testbench. At the beginning of the simulation the input signals to your circuit are driven with the `std_logic` value of 'U'. U' is one of the nine acceptable values of `std_logic` and indicates that the signal value is "Undefined". The test bench forces the input signals to produce "U"s at the beginning in order to imitate the undefined initial state of signals in hardware.

Open the testbench vhdl file and review how the intput signals to your entity are assigned. Mid-way in the testbench is a sequence of assignment statements that each form a test case to test your circuit. This testbench calls a procedure named "`check_output`" which will compare the inputs with the expected output. If there is a deviation between the inputs and the expected output, an error message is printed.

Modify the testbench to add an additional test case that tests your circuit with the following conditions: both button 4 and button 2 are pressed simultaneously (and the other buttons are not pressed) and the switches are set to "11101000". Rerun your simulation and determine the value of your circuit under this new test case.

Upload: Paste your updated testbench code segment.

Question: What is the value of the LEDs under these new input conditions?

Exercise 3 – LED Synthesis and Download

Now that your VHDL architecture passes all of the simulation tests, you are ready to synthesize and map your architecture onto the FPGA. Before proceeding with this step, you need to create a new .ucf file that contains the mapping between each input and output of your circuit and a FPGA signal name. The name of your UCF file is important - the extension must be ".ucf" (the extension is case sensitive). A UCF file with an extension of ".UCF" will not be accepted by the tools. Refer to the example in the previous lab and the Nexys2 reference manual to create a new .ucf file for this lab. You should have pin mappings for 20 signals: 8 for the switches, 4 for the buttons, and 8 for the LEDs. After creating this file, you should double and triple-check your pin mappings. A common student mistake is to provide the wrong pin mappings in the .ucf file, and it can be very difficult to debug your circuit if the .ucf file has mistakes.

Perform the HDL synthesis on your VHDL architecture (see the Synthesize-XST item in the Design Processes window). You may want to refer back to the screencasts in the previous lab to remind you how to execute the HDL synthesis. After completing the HDL synthesis, review the synthesized design by viewing both the RTL schematic and the technology schematic. Viewing these schematics gives you a visual understanding of the complexity of the circuit that was synthesized.

Access the "Synthesis Report" and answer the following questions:

Question: Review the synthesis report and determine the "Maximum combinational path delay".

Question: Review the synthesis report and determine the number of "slices"

Once the HDL synthesis is complete, proceed with "Implement Design" phase of the design mapping. If all goes well, you will have a green check by the "Implement Design" phase. You may have problems in this phase if you have mistakes in your .ucf file. After successfully completing the "Implement Design" phase, watch the following screencast to evaluate your design and make sure your ports were mapped to the proper FPGA pins.

Screencast: Pinout Check

If your design is properly implemented and the pins are properly mapped, proceed to the "Generate Programming File" and generate the design bitstream. Do not forget to change the "FPGA start-up Clock" option from CCLK to JTAG as discussed in a previous lab (see Figure 1.3 from the previous laboratory summary). Download your bitstream to the board and verify the operation of your circuit on the FPGA board. Once you have successfully downloaded your board to the FPGA, upload your design files.

Upload: Upload the VHDL file for your design

Upload: Upload the .ucf file used for your design

Personal Exploration

Spend some additional time in this lab exploring some of the many new concepts you have learned in the lab. Some suggestions for personal exploration include:

- Review the various "design summary" files that are generated during the Synthesis, Implementation, and Generate Programming file process.
- Right click on any of the Design Processes and select "Process Properties". Review some of the properties that can be changed during design implementation (don't change these at this time—you could change a setting that is necessary for correct synthesis of your design).
- Review some of the messages that were generated during design implementation
- Browse through the board user's guide to learn more about what the board can do

Pass Off

Demonstrate the following to a TA to passoff your lab:

- Show your VHDL testbench simulation waveform.
- Demonstrate to the TA a working circuit on your board.