

EE 1315 DIGITAL LOGIC LAB EE Dept, UMD

EXPERIMENT # 1: Logic building blocks

The main objective of this experiment is to let you familiarize with the lab equipment and learn about the operation of the fundamental combinational logic elements, **AND, OR, NAND, and NOR** gates. You will also familiarize with the Field Programmable Gate Array (FPGA) prototyping board (Nexys-2) and the Xilinx software development environment that will be used for future experiments. You will also learn how to use the Xilinx ISE development environment and how to download the bit stream to the Digilent Nexys-2 board.

Please note:

- 1) **It is necessary for this and all subsequent labs that a flash drive be used to save your lab projects.** It is suggested that a folder on your flash drive be created for the sole use of this lab and all projects for this semester be saved to that folder.
- 2) **Steps 9-15 will be conducted every time that a project is downloaded to the FPGA.** It may be useful to keep a copy of those instructions available to reference for a few weeks until you are used to programming the FPGA board.


When developing with Xilinx ISE, some important information about the device being used (the prototype board) must be known. Look at the chip in the center of the board at your lab station (the FPGA chip). The information on the chip denotes the family, device, and package. Please find this information now.

Family: Spartan 3E
Device: XC3S500E
Package: FG320

Step-by-Step Instruction

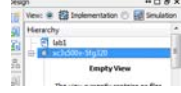
You will now use the Xilinx ISE schematic tool to develop the four logic gates needed for this experiment.




Step 1: Open the Xilinx ISE by clicking on the shortcut on the desktop .

Step 2: Create a new project by choosing **File->New Project**. Give the project a name such as “lab 1” (a more descriptive name could be used). The folder where the project is to be saved can also be chosen (somewhere on you flash drive). Click next.

Step 3: A new project wizard screen will appear. Enter the values for the family, device, and package that are provided above. Leave the other values unchanged. Click next and then click **finish** on the following window.

Step 4: Right click on the device (can be seen highlighted here: ) and choose **new source**. In the window that appears, select the **schematic** type and give it a name. Click **next** and then **finish**.

Step 5: You will then be presented with the screen seen in figure 1. Take a moment to mouse over the tools on the left of the lab1.sch and note their function. Click on the **Add Symbol** button . This will allow you to add a gate to your schematic.

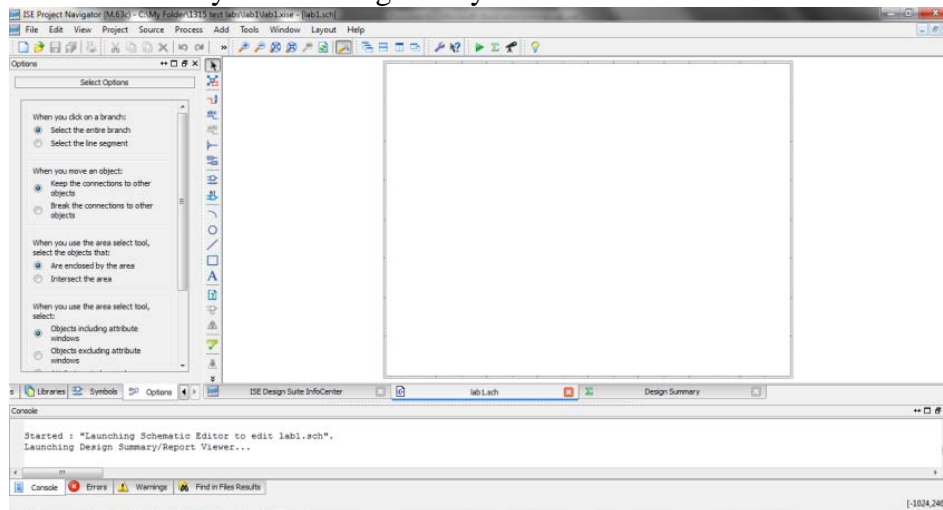



Figure 1: view of empty schematic lab1.sch

Step 6: Notice to the right of the schematic area a symbol dialogue has appeared. Select **Logic** under the category. The lower dialogue now shows the gates available in the logic category. Start with a two input **AND** gate. Find the **and2** and then click it to select (to speed up the selection, the symbol filter text box can be used to limit the choices in the lower window).

Step 7: Drag your cursor over the schematic area. Notice there is now an **AND** gate following your cursor. Click to place it on the schematic. To place additional **AND** gates on the schematic, keep clicking. To stop placing **AND** gates press **esc**. Repeat this process to place one two input **OR**, **NOR**, and **NAND** (**or2**, **nor2**, and **nand2** respectively).

Step 8: It may be beneficial to zoom in closer to the gates that were just placed on the schematic, do this by using the zoom tools at the top of the schematic. Place I/O markers using the I/O tool . Place one marker on the output of each gate and one marker on each of the inputs of the gates. Each I/O marker can be renamed by right-clicking on it. Rename the output of the **OR** gate to z1, the output of the **NOR** gate to z2, the **AND** gate to z3, and the **NAND** gate to z4. Similarly rename one input of each gate to x and the other to y. You will get a warning window when you attempt to name a port with the same name as another. Xilinx will ask if you want to merge the ports. By selecting yes, it is like connecting a wire to the two terminals. Select yes for this lab. Your schematic should look similar to figure 2.

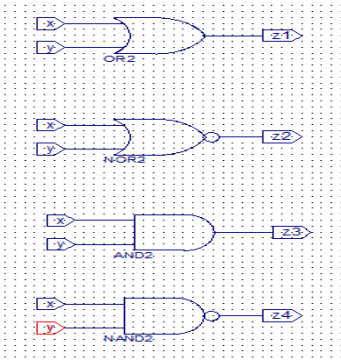


Figure 2: completed schematic

Step 9: You will now generate the binary file and use it to program the FPGA board. Navigate to and expand the User Constraints option in the Design Tab (seen in figure 3). Double click on the **I/O Pin Planning (PlanAhead)-Pre-Synthesis** item. A window indicating a User Constraint File (UCF) must be created will appear. Click **yes**.

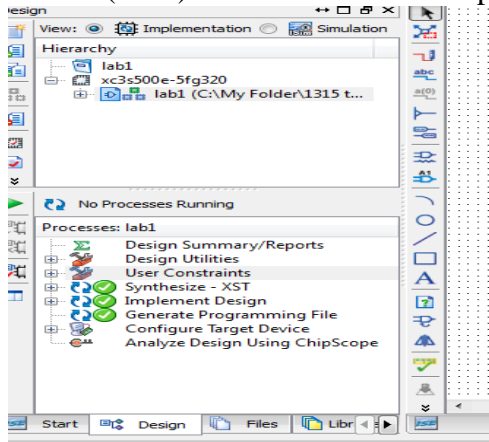


Figure 3: User Constraints

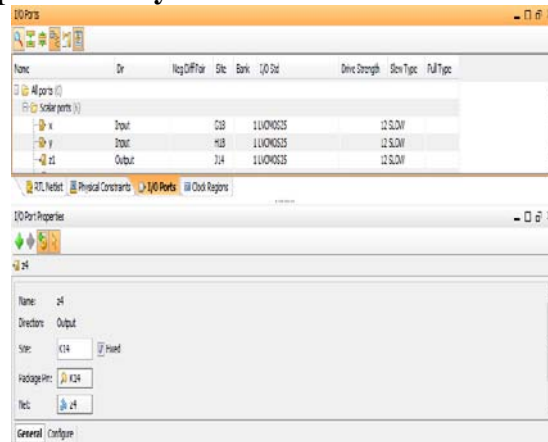


Figure 4: I/O Port Menu

Step 10: A new program (RTL Design) will start (be patient). Navigate to the **scalar ports** menu in the **I/O Ports** tab. This can be seen in figure 4. Select “x” and set the site to G18. Click Apply. Continue by setting “y” to H18, “z1” to J14, “z2” to J15, “z3” to K15, and “z4” to K14. The Ports should be defined as seen in figure 5. If everything looks correct click **File->Save Project**.

Name	Dir	Neg Diff Pair	Site	Bank	I/O Std	Drive Strength	Slew Type	Pull Type
All ports (6)								
Scalar ports (6)								
x	Input		G18	1	LVC MOS25	12	SLOW	
y	Input		H18	1	LVC MOS25	12	SLOW	
z1	Output		J14	1	LVC MOS25	12	SLOW	
z2	Output		J15	1	LVC MOS25	12	SLOW	
z3	Output		K15	1	LVC MOS25	12	SLOW	
z4	Output		K14	1	LVC MOS25	12	SLOW	

Figure 5: Ports with assigned locations

Note: In future labs a table as seen in figure 5a will be provided to aid in assigning Scalar Ports to the proper Site on the board.

Scalar Port	Inputs		Outputs			
	X	Y	Z1	Z2	Z3	Z4
Site	G18	H18	J14	J15	K15	K14
I/O	SW0	SW1	LD0	LD1	LD2	LD3

Figure 5a: Scalar Port assignment table

Step 11: Navigate back to the Xilinx ISE. Under the **User Constraints** item you just expanded, you will find **Synthesize-XST**, double click or right click and select run. Now double click the **Implement Design** found under the **Synthesize-XST**. It is usual for these operations to take some time. If the operations are successful, you will see green check marks appear next to the items. If a red X appears next to either of the items, there was an error, please let your TA know.

Step 12: The clock needs to be set before generating the final bit file. Right click on **Generate Programming File** and select **Process Properties**. Select **Startup Options** and change the **FPGA-Start-Up Clock** to **JTAG Clock** as seen in figure 6. Click **Apply** and **OK**.

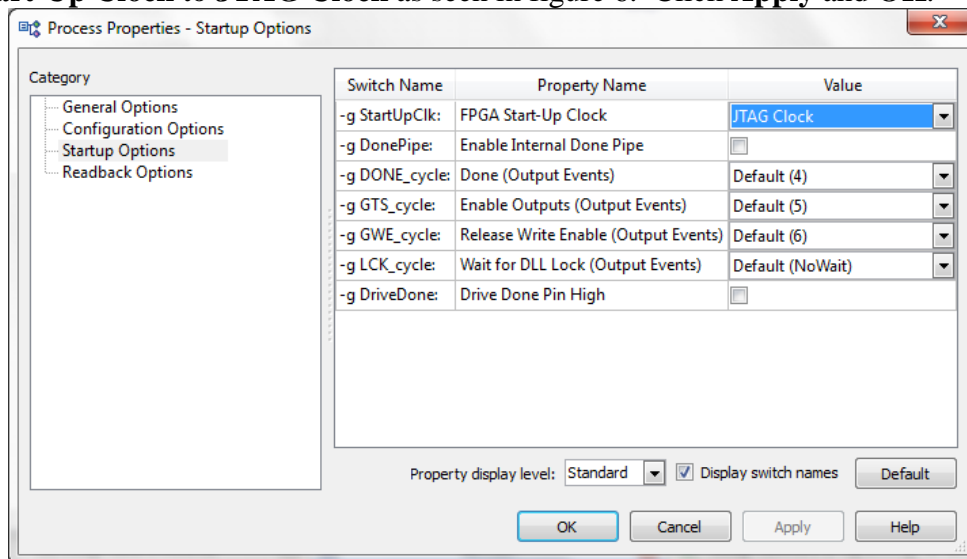


Figure 6: Setting the JTAG Clock

Step 13: Now double click **Generate Programming File**. If a green check mark appears, you have successfully generated a bit file to program the FPGA with. This file will automatically be saved in the directory where your project is saved.

Step 14: Make sure that the Nexsys-2 board is turned ON. Double click on the Adept icon on the desktop. It should show the window seen in figure 7. Note that Adept automatically recognizes the board. If an error is received, check you connections and click **Initialize Chain** to attempt to re-connect to the board.



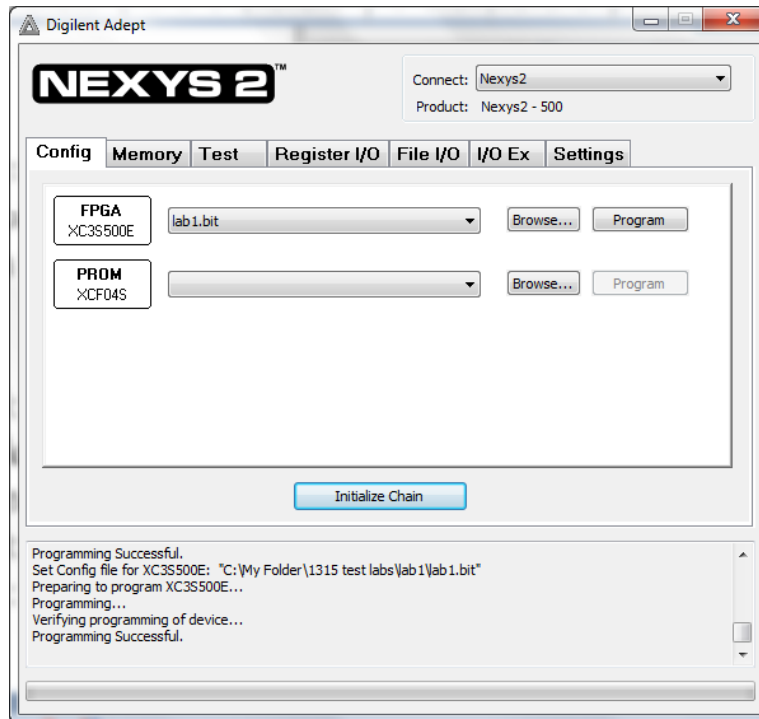


Figure 7: Adept window

Step 15: Click the **Browse...** button on the FPGA row and find the bit file you generated (lab1.bit in this case). Click the **Program** button. A progress bar should grow at the bottom of the window. When the message programming successful appears, the FPGA has been successfully programmed.

EXPERIMENT #1 RESULTS

(Print this page and bring to lab for TA to sign)

The FPGA has now been configured to see a down slide switch as a 0 and an up slide switch as a 1. In addition the outputs are seen by the LEDs above the slide switches with an off state indicating a 0 and an on state indicating a 1. This can be seen in Figure 8. Please show the TA the working circuit and complete the table and questions below.

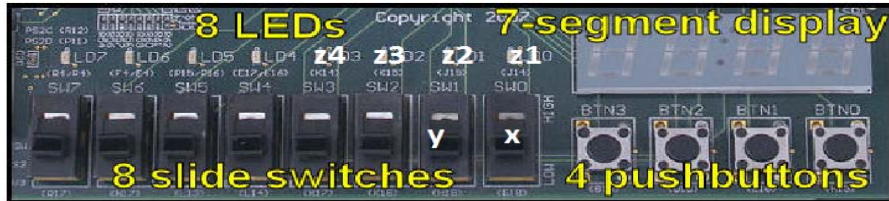


Figure 8: I/O for lab #1

Witnessed by :

Date _____

Recorded Truth Table

x	y	z1 (OR)	z2 (NOR)	z3 (AND)	z4 (NAND)
		IC#:	IC#:	IC#:	IC#:
0	0				
0	1				
1	0				
1	1				

1. Which pairs of gates have inverting relations according to the truth table above?

2. Write an algebraic equation for each type of gate using the inputs x and y (refer to the web pages of the pin-outs, or textbook, or ask TA)

OR:

NOR:

AND:

NAND: