LABORATORY # 1

Xilinx Environment

Objective
(1) To gain experience with the Xilinx ISE schematic editor & simulator.
(2) Practice programming the Spartan3E FPGA evaluation board.

Equipment
PC workstation, Xilinx Webpack ISE, Basys FPGA evaluation board.

<u>Description</u>
The Xilinx software is installed on the computer. Create new project for each problem. You need to draw schematics of each circuit and run a simulation.

NOTE: (constraints file information) The pins on the chart for the anodes are incorrect; the correct pins are AN3-P26, AN2-P32, AN1-P33, and AN0-P34.

<u>Background</u>
(Do not read this section yet, you will come back to this near the end of the lab. Skip to <u>Laboratory Exercise</u>.)
The 7 seven segment displays are 7 LEDs placed in a way to create a number 8 layout. These segments can be turned on and off to create numbers and some letters.
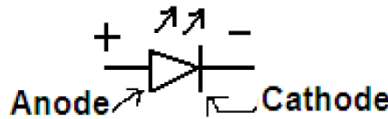
All LEDs have an anode and a cathode. To light up an LED you need to apply high voltage at the anode and low voltage on the cathode. For your purpose, think of the high voltage as a binary 1, and low voltage as a binary 0. For this particular board, the 7 segment display is a common anode display. This means that all the anodes from the 7 LEDs are connected together, and you control one display with one anode, and each individual LED with the 7 cathodes. Look on your board pin sheet, you'll notice there are four anodes, AN0-3, and on the right side there are 7 cathodes, 'A-G'. In this case, the cathodes for all four displays are joined together based on letter. For example, all the 'A' segments in all four displays are tied together. If you apply 1 to all four anodes and 0 to cathode pin 'A', all the 'A' segments will turn on. One last important thing to know is that the seven segments have a NOT gate before every anode.
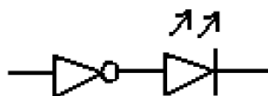
In order to light one of the LEDs, the pins for the anodes must be set to 0 in order for the LED anode to receive a 1. The cathode will also need a 0 to light the LED. If the anode pin is 1 the LEDs will not light up.

<u>Basys Demo bit file</u>
The Basys demo bit file is a very useful code to have loaded in the ROM at all times. You can always power off your board and switch the ROM/JTAG jumper to ROM mode. Then power the board back up. It will load the program from the ROM memory into the chip and run it. You can then use the switches SW0-7 to test the LEDs LD0-7; each switch should turn on one LED when flipped. You can use the buttons BTN0-3 to test the 7 segments above it; each button will cause a "snake" effect on the 7 segment display.

<u>Loading Basys Demo bit file into ROM</u>
Run ***Adept***. Click on windows menu (bottom left corner) Type ***Adept*** in the search, should be the first result.

Plug in your board to the computer, and click *Initialize Chain*. You will see 2 icons show up on the main window.

If you see *Connect: No devices connected,* click and select *OnboardUSB.* You should now see the 2 icons. If you still do not see these icons, wait for drivers to install or consult with your TA.

Click the *Browse…* button next to the PROM icon.

Browse to C:\Digilent_Basys_CD\; you should see a 2 .bit files in there. BasysDemoCCLK.bit is made for loading into ROM; BasysDemoJTAGCLK.bit is made for loading to the board while its in JTAG mode.

Select BasysDemoCCLK.bit.

For Basys2, use basys2_100userdemoCClk.bit.

Click *Program* next to the PROM icon. Wait for the programming to successfully complete.

Make sure the the ROM/JTAG jumper on your board is set to ROM before the next step or your board will not run the code.

Now reset your board and it will load the Demo from the ROM memory into the chip. Test every LED with switches and the 7 segment display with the buttons.

Laboratory Exercise

1. Open Xilinx ISE "32-bit Project Navigator" and click File>New Project.

2. Enter a "Project name" and select a "Project location" for your project.

3. Select "**Schematic**" as your "Top-level source type" and click next.

4. Your next window should look EXACTLY like this EXCEPT for Top-Level Source Type

For Basys:

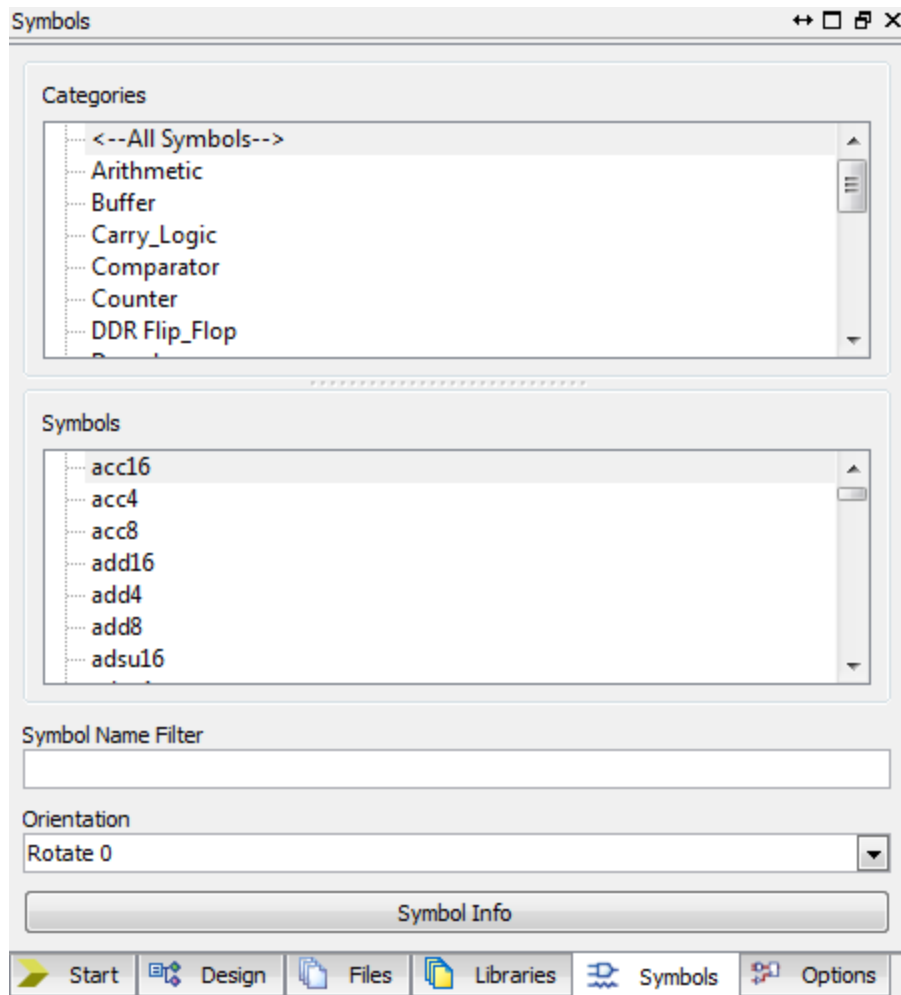| Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	TQ144
Speed	-5
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values

For Basys2:

Device	XC3S100E
Package	CP132
Speed	-5

Click next.

5. Click "New Source," select "Schematic," enter a "File name," click next, click finish, (if you haven"t created the folder yet, click yes), click finish, and click next.

6. You do not need to add any sources, click next. Click finish.

7. You should now see something like this. (Some important items have been boxed in.)
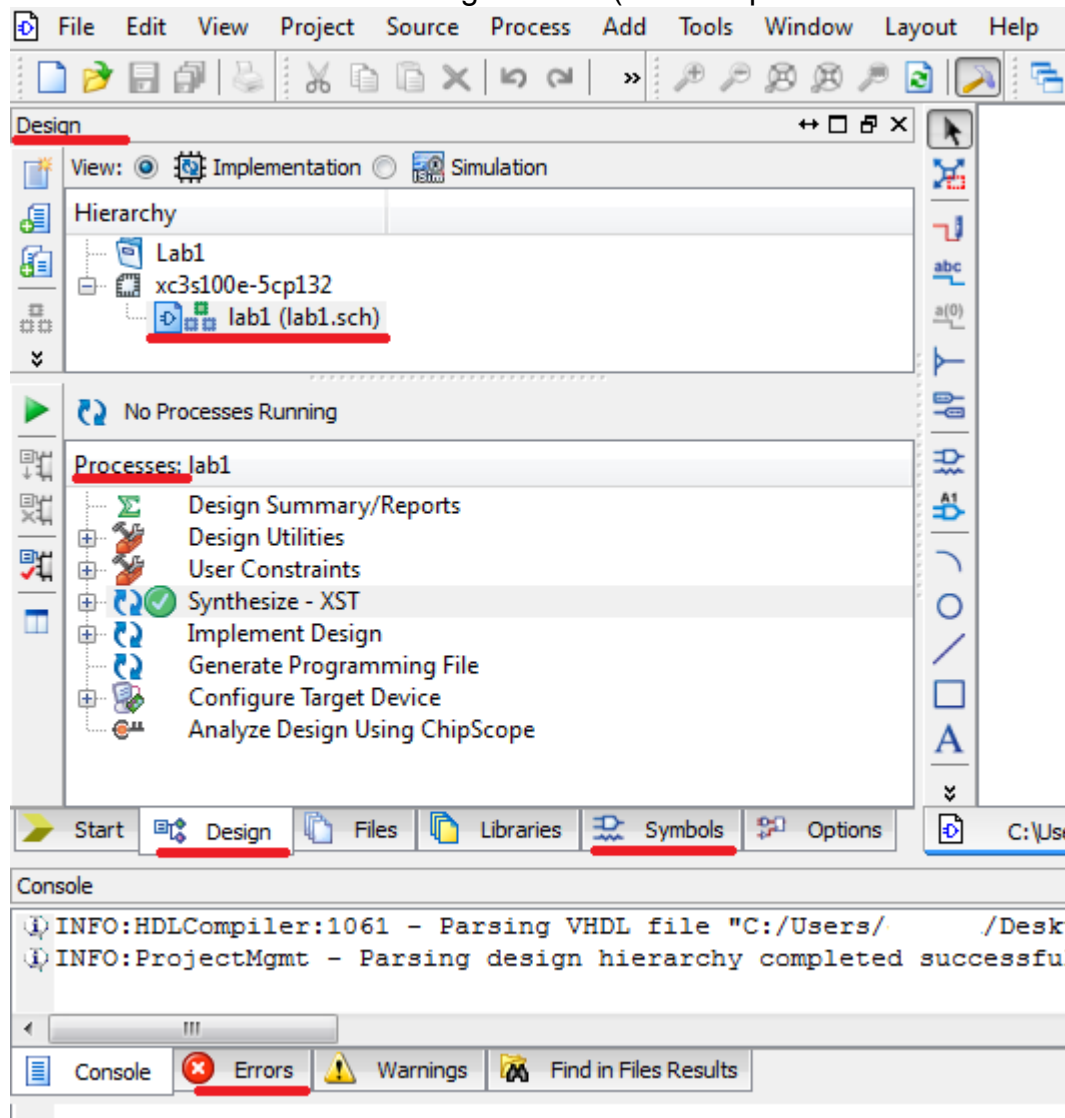
8. Brief description of the above:

a. The Design and Processes windows have many tabs that change as you select different items in the sources.

b. The Design window has two options. You will use these to switch between implementation and simulation modes.

c. The Errors tab in the console window is useful as it shows only the errors that have passed through the console while running a process.

d. The right side window shows files that are open, such as schematics and simulations. You can see your schematic file is open.

9. Click the "Symbols" tab. At this point it is a good idea to make the Symbols window longer, so you can see more of the list of symbols available for use.

Categories

<--All Symbols-->
Arithmetic
Buffer
Carry_Logic
Comparator
Counter
DDR Flip_Flop

Symbols

acc16
acc4
acc8
add16
add4
add8
adsu16

Symbol Name Filter

Orientation
Rotate 0

Symbol Info

Start | Design | Files | Libraries | Symbols | Options

10. Find the 2-Input AND GATE, type in AND2, and drop it on your schematic.
11. Add a 2-Input NAND, OR, and NOR gate, as in step 10.
Tip: You can hit ESC key to deselect a part.
12. Wire the top inputs together and the bottom inputs together to make two inputs only.

1 2 3 4 5 6 7 8

Look for this toolbar [toolbar icons] and select the (second) icon that looks like a pencil drawing a red line.
Tip: When wiring, make sure you a dot appears after placing a wire with another wire to make sure it has connected.
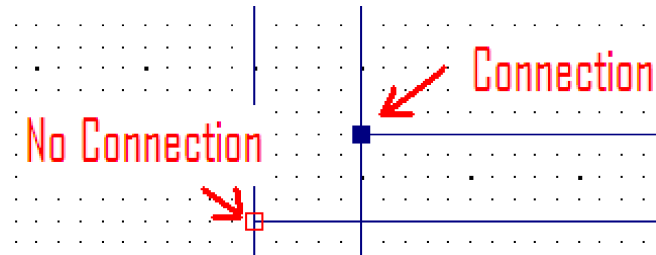
13. From the schematic sidebar, select the "Add I/O Marker" icon. Now click each input and output. A marker will now be created with a name XLXN_XX, where XX is a number.
Tip: You may rename the markers by right clicking them and selecting "Rename Port" after deselecting the "Add I/O Marker" tool.

14. At this point, save your schematic. You now need to synthesize your schematic to simulate it.
NOTE: You will not be able to run a simulation without synthesizing first.
15. In the Design window, make sure your schematic file is selected. In the Processes window, you will see new items.

Before proceeding, rename your markers or you will need to change names in your test benches later, as well.
      -Right click on the marker and click Rename Port, one by one.
      -Always save and re-synthesize after renaming any ports.
      -It will be very easy to have name mismatches between the schematic and the test bench.
      -Always give your input and outputs distinguishable names.
            Example: Top, Bot, A, B, C, D, and so on.
            Avoid: Input_1, Input_2, Output_1, Output_2, Output_3, and so on.
            If you rename sections of code with a lot of Output_1's to Output_2's and you miss one it may be hard to catch.

16. Back in the Design window, right click and select new source.
17. Choose "Verilog Test Bench," type in a name, and click next twice, then click finish. You should see your test bench open.
18. At the top of the Design window, click "Simulation" and select "Behavioral" from the drop down menu.
19. You should now see your test bench in the Design window.
      -You will now need to make some modifications to the test bench.
      -Notice there is a barebones template already created for you.
      -The Architecture describes your test bench, it identifies inputs and outputs, and any signals that will be used within that component. You can think of these signals as the probes you will use to manipulate and observe the schematic you put together earlier.
20. First we'll need to add a clock source to have a time basis for our inputs.
      -Your signals should look something like this.
       --Inputs
       reg Top ;
       reg Bot;
       --Outputs
       wire A   ;
       wire B   ;
       wire C   ;

```
wire D     ;
-- Clock definitions
reg clk;
```

-We have added the bottom line. Keep in mind this is the period of your clock source, which includes a high time and low time of half the period each.

21. We will add two blocks. Each of these will control one of the inputs simultaneously. Since we are creating combinatorial logic, it is easy to create this by using the clock source and making each input twice as lengthy as the previous OR the other way around, it is really up to you and it doesn't matter in most cases.

```verilog
module test_test_sch_tb();

// Inputs
  reg Bot;   reg Top;

// Output
  wire A;   wire B;   wire C;   wire D;

// Clk
reg clk;

// Instantiate the UUT
  test UUT (
               .A(A),
               .B(B),
               .C(C),
               .D(D),
               .Bot(Bot),
               .Top(Top)
  );

  initial begin
   clk = 0 ;
   forever begin
      #20 clk = ~clk;
   end
  end

       initial begin

       #40 ;
       Bot = 1'b1;
       Top = 1'b1;
       #40 ;
       $display("TC1 ");
       if ( {A,B,C,D} != 4'b1010 ) $display  ("Result is wrong %b ", {A,B,C,D});

       Bot = 1'b0;
       Top = 1'b0;
       #40 ;
       $display("TC2 ");
       if ( {A,B,C,D} != 4'b0101 ) $display  ("Result is wrong %b ", {A,B,C,D});

       // Your own test cases

   end
```

22. It is important to understand that ALL blocks in verilog code run concurrently. (The blocks run literally at the same time.)

23. Once you have taken care of the two inputs, save your test bench and make sure your test bench is selected in the Design window.

24. In the options displayed in the Processes window expand the "ISim Simulator" and double click "Simulate Behavioral Model." The simulation window will open up. There are some other items on this window other than your inputs and outputs. These can be ignored or removed.

25. Now that the ISim is running, you can compact the waveforms by holding down the "Ctrl" key and simultaneously scrolling with your mouse.

> -Notice the compaction occurs with respect to the yellow bar. If you'd like to zoom in with respect to the beginning of the waveform, zoom out (compact) and click (or drag) the yellow bar to the beginning of the waveform, then scroll in.
> -Becoming familiar with these shortcuts to manipulating the waveform being displayed will facilitate the analysis of future waveforms.

26. You must always save your test bench after any changes to it.

If you have the ISim window open after any major changes, you will need to close it and re-run the "Simulate Behavioral Model."

Creating a UCF and BIT file for download to the FPGA

The UCF (Implementation Constraints File) is what matches the names of the markers in your schematic to the actual pins on your FPGA hardware.

The board has 7 switches and 4 buttons that can be used as inputs, and it has 7 LEDs and 4 7-segment LED displays that can be used as output.

1. Make sure you are in "Implementation" mode in the Design window.

2. Select your schematic in the Design window.

3. Right click and select "New Source."

4. Select "Implementation Constraints File." Give it a name and click next, then finish.

5. The UCF file was added to your schematic, expand the tree of your schematic and you will see the UCF file. Select it.

6. If you need to open the UCF file, just double click it.

7. Choose 2 inputs on the board (switches or buttons) and 4 LEDs (use only the individual LEDs). Notice each has a letter and a number. These are the pin number names. Refer to the sheet that came with your board for the pin numbers OR if you have a BASYS 2 board look directly at the board.

8. In the UCF file window enter this format:

NET "name" LOC = "pXX";
NOTE: If you copy paste that line from here, you may have to retype your quotation marks.

Where XX is the pin number. Add one of these lines for each input and output. You should have a total of 6 lines with a different name and pin number for each.

9. Save your UCF file.

10. Select your schematic in the Design window. Make sure it is "Set as Top Module", if not, right click it and do so.

11. **Make sure the UCF is under the schematic in the file tree.** If not, remove it and add it again.

12. In the Processes right-click on "Generate Programming File," choose Properties and in the pop-up window select "Startup Options". Make sure that FPGA startup clock is JTAG Clock (not the default CCLK).

13. Now double click "Generate Programming File."

14. You should have all icons displaying green circles with white checks. If you see any red circles you can try selecting "Project>Cleanup Project Files" and repeat step 12.