EE-3563
Digital Systems Design
Lab 2

In this lab you will design a simple 4:1 Multiplexer and then write part of the HDL code for it in Verilog. You will also down load this code into the NEXYS4 board and run it. Then we will use a simple "Test Bench" to simulate the design and show that it works in simulation. Create a new project in Vivado just as you did in the 1st lab in the tutorial. It is best to select the source, XDC and TestBench files separately after the basic project is created. (make sure the box is unchecked so you can specific sources later) You can then right click on the Source line in the project window and add source. The file here is the Mux4.v file. Then right click on the constraint line and add the XDC file that goes with your board. Finally click on the simulation line and add test bench and add the test bench file.

You can use the example 2:1 mux code "Mux2" to implement the 4:1 by instantiating 3 copies of the Mux2. The purpose of this lab is to teach you how to instantiate a sub module several times to get the Mux4 functionality. You could just use a mux4 directly, but you would not learn about how to instantiate sub modules. The Mux4.v file already has the Mux2 module pasted in and is mostly set up so all you have to do is to write a bit of code. The main module is called "Mux4" and it is set to use SW4 and SW5 as the select inputs A and B, SW0 – SW3 as I0 – I3 inputs and LED 0 as output. This has been set up in the Mux4.xdc file. Make sure you use the variable names in the xdc file. Show that the output follows which ever input has been selected by A and B. First draw you design and label the in's and out's in a block diagram on your answer sheet in question 4: (or you can paste in the Schematic from Vivado). The test bench is done, work to get it to run and display correctly. List the signals as in the Mux4.v file so it will work with the test bench which is Mux4_tb.v

1. Use Vivado 2016.2 run all the way to loading the bit file on the NEXYS4 board so you can demo the code with the hardware. You have to write the lines to instantiate the Mux2 module 3 times. Be sure an give a unique name to each instantiation of the Mux2.
2. Use the correct configuration file included on blackboard: Mux4.xdc (Rev B or C depending)
3. Then write a test bench file Mux4_tb.v and use it to run the simulation.
4. Answer the questions and turn in on BB the answers and your code pasted in.

Here is the Mux2 code:

```
module Mux2 (
   input wire sel,
    I0,
    I1,
   output reg Y
   );
always  @ (I0 or I1 or sel)
  begin
   if (sel == 0)
   begin
     Y = I0;
   end
   else
   begin
     Y = I1;
   end
  end
endmodule
```

DSD F17
Lab 2 MUX

Name _____Andrew Quick_____ Date_____

1. What problems did you encounter in the lab?

Learning the logic design of the MUX was the hardest part.

2. Would your code be simpler if you just wrote a MUX4 module without using the Mux2 modules?

No, the instances prevent writing redundant code.

3. Can you write a Mux without an always block?
Yes, you can use combinational logic:

assign Y = (sel == 1'b0) ? I0 : I1; // If sel is 0, choose I0, else choose I1

4. How could you improve the test bench?

I could cycle through every possible combination of inputs, I only cycled through the different possible controls signal and two input combinations.

5. Draw your block diagram here: