CompArch: HW b010

Due September 25th at 5pm

This homework prepares some of the gate level primitives you will use in the design of your processor. You will reuse these modules in several future designs. Therefore, the structure and test will be in separate modules.

This homework is to be done individually.

The Devices

This homework is based on the following three devices:

1. 2-bit decoder with enable (2+1 inputs, 4 outputs)
2. 4:1 (four input Multiplexor)
3. 1-bit Full Adder

For each of these three devices, you will do the following:

1) Write a test bench to test the functionality of the device.
2) Test your test bench against my version of the device.
3) Write your own version of the device.
4) Test your device against your test bench.

The Test Benches

For each device, first write a test bench that verifies the appropriate behavior of your device. I have already completed this for you for the 2 bit decoder as an example, so you will only have to write the other two.

The test bench should:

1) Instantiate a copy of the device it is testing (Device Under Test = DUT)
2) Show what the truth table should be
3) Show what the truth table is

The Behavioral Devices

I've written versions of each of the three devices in a language subset called "Behavioral Verilog". Use these versions to test your test benches. Connect the test bench to the device definition I've provided and verify that your test bench passes my device.

The Structural Devices

Create the three devices in Structural Verilog, using only the gate primitives we have already gone over:

NOT AND NAND OR NOR XOR

Do not use behavioral constructs such as ‘assign’ or ‘case’.

Give all of your gates a delay of 50 units of time.

The Write Up

This should be a no frills PDF or Markdown file containing pictures of your test bench results and your waveforms that show the gate propagation delays.

Submission

Please submit:

1) Your modified do file
2) Your 3 Verilog Files
3) Your writeup as a PDF or Markdown file

by pushing to your GitHub in the HW2 folder.

Hints / Tricks

Gate delays

In order to model some sort of delay for our gates, simply put these statements at the top of your Verilog source:

```
// define gates with delays
`define AND and #50
`define OR or #50
`define NOT not #50
```

Then, when you go to instantiate an AND, for instance, instead of using just “and”, use `AND. That is, back-tick followed by the define you specified. Think of the back-tick as a macro definition.

That means that the gate, `AND, has a delay of 50 units. Then, in your simulation, you should wait between transitions of the input long enough to allow the signals to propagate to the output of your circuit.

Signal Declaration

You need to declare all your inputs and outputs and all the intermediate signals you use in your designs. Thus, if you have the statement:

```
and myandgate(out, in1, in2)
```

You need to have previously declared out, in1, and in2, to be some sort of physical entity (wire, reg).

Tutorials

There are some Verilog resources listed at http://sites.google.com/site/ca15fall/resources/verilog

The ModelSim tutorial may also be helpful, and is available in the program menu under Help > PDF Documentation.