ENGR xD52: HW b001

Due September 22nd Midnight EST

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 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

The Test Benches

For each device, first write a test bench that verifies the appropriate behavior of your device. To get you started, I have provided:

1) A completed test bench for the 2 bit decoder (this means you only have 2 to write yourself)
2) Versions of the three devices written in Behavioral Verilog. Your versions will be in Structural.

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

You may want to truncate the truth table for the 4 input multiplexer - it has 2^6 = 64 entries.

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 containing pictures of your test bench results and your waveforms that show the gate propagation delays.

Submission

Submit a zip file with the following to Comparch14@gmail.com:

1) Your modified do file
2) Your 3 Verilog Files
3) Your writeup as a pdf

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 (out, in1, in2)
```

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

Tutorials

These may be helpful:

http://ca.olin.edu/cawiki/attachments/Fall(20)2010(2f)Materials/VerilogTutorial.pdf

http://asic-world.com/verilog/index.html

http://www.asic-world.com/verilog/art_testbench_writing1.html

http://ca.olin.edu/cawiki/attachments/Fall(20)2010(2f)Materials/modelsim_se_tut.pdf

http://ca.olin.edu/cawiki/attachments/Fall(20)2010(2f)Materials/m_qk_guide.pdf

http://ca.olin.edu/cawiki/attachments/Fall(20)2010(2f)Materials/test_logic.v

Do Commands

If you do something repeatedly, add it to your do file. Here is my do file while I was debugging the multiplexer:

```
vlog -reportprogress 300 -work work multiplexer.v decoder.v adder.v
vsim -voptargs="+acc" testDecoder
add wave -position insertpoint  \
sim:/testDecoder/addr0 \
sim:/testDecoder/addr1 \
sim:/testDecoder/enable \
sim:/testDecoder/out0 \
sim:/testDecoder/out1 \
sim:/testDecoder/out2 \
sim:/testDecoder/out3
run -all
wave zoom full
```

add wave … allows you to automatically populate your waveform viewer.

wave zoom full automatically zooms the viewer to see the entire waveform.

Run -all attempts to intelligently decide how long to run for. Note that it is possible to get stuck simulating forever with this option.

Note that it compiles three .v files.