ENGR xD52: MP b000

Due September 30th 5PM EST

This lab assignment will develop some of the basic building blocks for your processor and introduce you to Verilog.

You may work in groups of up to 3. One representative from each team must submit all deliverables electronically to comparch13@gmail.com as a single compressed archive. Please format the email's Header as "[CA] [MP0] Name1 Name2 Name3"

The Modules

Construct each of the following modules in Structural Verilog:

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

All logic must be done gate level and structural, using only the basic gates NOT AND NAND OR NOR XOR. You may re-use modules between the four.

 Do **not** use behavioral constructs such as 'assign' or 'case'.

Give all of your basic gates a 50 unit time delay.

For each module, also write a behavioral test bench that shows the module-under-test to be correct.

The Write Up

Create a semi-formal lab write up detailing the modules, their expected behavior, and their tested behavior. Include the waveform outputs of your test benches. If you do not perform exhaustive testing, explain why it was not necessary.

Optionally, include additional information, such as the timing performance or design tradeoffs of your modules.

This should be a single document bundled into the submitted archive.

Hints / Notes

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