Lab Introduction/Lab 1

ECE/CS 472/572
Welcome to Lab!

• Lab time: Friday 2pm-3:50pm
  – Most assignments will take longer than this block of time to finish.
  – This time will be very informal.
    • I will cover the assignment at the beginning, and leave the remaining time for you to work.
  – If you need extra help, you can e-mail me or come into my office hours.
    • E-mail: pawlowsr@onid.orst.edu
    • Office Hours: Tuesday/Thursday 10am-11am.
Goals for Lab

• We will focus on using Verilog to model and simulate different topics within computer architecture.
  – Start with smaller blocks.
  – Learn about the implementation of a MIPS processor in Verilog.
  – Figure out how to alter the processor to perform important functions.
Logistics

• 4 Labs + 1 final project
  – The 4 labs = 20% of final grade.
    • Lab 1 = 2%, Lab 2 = 4%, Lab 3 = 6%, Lab 4 = 8%
  – Final project = 30%

• Turn-in will be through TEACH
  – The length of labs varies (could be 1, 2, or 3 weeks), but they will always be due by midnight on a Friday.
  – 1 late lab is allowed. Beyond that you will not get any credit for a late assignment.
  – Work in groups of 2, only 1 person needs to turn the lab in.
  – The expected format for write-ups will be specified at the end of each lab assignment.
Grading

• Labs will have more strict grading than HW.
  – 10 points possible, graded on correctness.
  – The labs are turned in online and you will be given a score with minimal feedback.
    • If you would like more feedback on your assignment send me an e-mail and I can give you more info on the reason for your score.
Today’s Schedule

• Walk-through installation of modelsim.

• Short intro to Verilog.
  – We will cover the necessary areas to help you do this weeks lab.

• This weeks assignment
Getting ModelSim

- If you have Windows, follow the link on the lab webpage to get to the ModelSim student edition download site.
  - You need to fill out a short request form to download.
  - Once you have finished your download and installation, you will need to fill out another short request form to get a license.
  - Put the license file that was e-mailed to you in your top level installation directory.
  - You’re ready!
Getting ModelSim (cont)

• If you don't have Windows, ssh'ing into flip/flop should work fine. Make sure you have X-forwarding turned on.
  – Create an ECE472 directory
  – To invoke ModelSim, type 'vsim'
  – Yeah! You're also ready!!
What is Verilog?

- Verilog is a Hardware Description Language (HDL).
- HDL’s describe the behavior of digital circuits/systems.
- Synthesis tools then take the code and create functional circuits.
- We will be using Verilog to help learn about and simulate different architectures.
Tips for Successful Coding

• Visualize the circuit you are trying to create while you are coding.
  – Write out a truth table or use another method of understanding what the logic inside of the module is doing.
  – It is helpful to draw a block diagram (with labeled connections between blocks).
  – Also draw the logic within modules, showing the individual gates and their connections.

• Verilog code does not run sequentially
  – Operation of the circuit depends strictly on how you connect modules and gates.
Overview using full adder

```verilog
module fulladder(a, b, cin, sum, cout);
  input a, b, cin;
  output sum, cout;
  assign sum = a ^ b ^ cin;
  assign cout = a & b | a & cin | b & cin;
endmodule
```
Notice that internal connections are defined as wires, external connections are defined as inputs or outputs.
'Always'

- Some circuits can be dependant on certain events. For example, a flip-flop is dependent on the positive clock edge.
- When you want to use a behavioral statement to describe your circuit, it should be within an always block.

```vhdl
module flipflop (clk, reset, d_in, d_out);
  input clk, reset;
  input d_in;
  output d_out;
  reg d_out;

  always @(posedge clk)
  begin
    if (reset) d_out <= 0;
    else d_out <= d_in;
  end
endmodule
```

Notice that when a value is being set within an always block (in this case d_out) it needs to be defined as a 'reg' datatype.
Behavioral Verilog

• A variety of other statements are available to use in Verilog including if statements and case statements.
  – Syntax will be different than what most of you are used to.

```verilog
testbench #1
 module mux (in_a, in_b, sel, out)
inpu[t in_a, in_b;
inpu[t sel;
output out;

reg out;

always @(in_a or in_b or sel)
begin
  case(sel)
  1'b0: out=in_a;
  1'b1: out=in_b;
  endcase
end
endmodule
```
The Testbench

For Testbenches, **wire** data types are used for outputs, and **reg** data types are used for inputs.

```verilog
module proj1_testbench;

  wire [3:0] sum;
  wire cout;
  reg [3:0] A, B;
  reg clk;

  //instantiate a ripple adder (DUT = Device Under Test)
  ripple_adder DUT(A, B, sum, cout);

  //Generate a clock that changes every 5 time units (A period of 10)
  always
    #5 clk =~clk;

  //Initialize signals
  initial begin
    clk = 1'b0; //initialize clk value to 0 at t=0
    A = 8'h00; //initialize input A to 00000000
    B = 8'h00; //initialize input B to 00000000
  end

  //add one to the value of A at each positive clock edge
  always @(posedge clk)
    A = A + 1;

endmodule
```

Default time unit is 1ns. To change time units to 10ns, for example, type: `timescale 10ns / 100ps` at the beginning of the file.

Purpose of the testbench is to instantiate the circuit, and feed inputs into it, giving you the chance to simulate and observe the outputs in the circuit.
Simulation

Following step 7 in the lab should lead to a successful and problem-free simulation, but here are two common mishaps:

– Be sure to select your testbench module when you simulate.

– Also be sure to uncheck the “Enable optimization” box.
Questions?

• If you have any questions regarding Verilog syntax or technique, take a look at the manual found on the ‘Assignments’ page of the course website. There are also a number of Verilog tutorials online.

• Also don’t hesitate to e-mail me at: pawlowsr@onid.orst.edu if you have any questions.
This week’s lab

• The purpose of this week’s lab is to get you used to the ModelSim environment, and the very basics of Verilog.

• Follow the steps in the lab manual, you should be able to successfully complete the lab based on the information you’ve been given.