CPEN 230L: Introduction to Digital Logic Laboratory
Lab #6: Verilog and ModelSim

Purpose

- Define logic expressions in Verilog using register transfer level (RTL) and structural models.
- Use Quartus II to synthesize the Verilog logic expressions into logic gates that get fitted into a FPGA.
- Use ModelSim with Verilog to simulate and debug the operation of the digital circuits designed.

Background

In previous labs we defined digital logic designs using truth tables, logic expressions, and schematic diagrams. In this lab and those following, we will use the **Verilog** Hardware Description Language (HDL) to define, build (a.k.a. synthesize), and test digital logic. To synthesize the logic expressions written in Verilog into logic gates that get fitted into a FPGA we will use **Quartus II**. To test the correct operation of the circuit we will simulate its behavior using **ModelSim**.

In digital logic terminology, the circuit to be verified is called a **Device Under Test (DUT)** or **Unit Under Test (UUT)**. The environment used to verify the DUT is called a **test bench**. A test bench has two parts:

- **Input vectors or stimulus** to drive the DUT inputs. For example, in the case of a full adder the eight possible input combinations would be the input vectors.
- **Output monitoring** to observe DUT outputs. These observations are normally a **truth table** and/or **timing diagrams**. For example, in the case of a full adder CarryOut and SumOut would be monitored.

Part 1: Combinational Logic Design using Verilog and ModelSim

Pre-lab:

- **Read and thoroughly understand** textbook pages 68-78 “Introduction to Verilog”. Knowing this now will make the rest of the semester go much smoother.
- **Quickly skim** textbook Appendix A “Verilog Reference”. You need to know where to find this information when you need it. Most important are pages 685-703. Next most important are pages 709-712 and 725-731 (skipping Figure A.40 and A.41). The rest is less important to this class.
- Read and understand the Verilog code below.
    - Create files logicMistery_tb.v and logicMistery.v containing the provided code.
    - Compile and execute the code using Icarus Verilog. You should see a truth table.
- What logic expression is being implemented by module logicMistery?
- Make a truth table showing output d for all possible inputs a, b and c.

```
// CPEN 230L Lab 6a, testbench for module logicMistery
// Firstname Lastname, mm/dd/yyyy
// File: logicMistery_tb.v

module logicMistery_tb;   // test bench, so no inputs/outputs
  reg a, b, c;            // inputs to the DUT
  wire d;                 // output from the DUT
  reg [2:0] count;        // 3-bit value to help generate DUT inputs

  logicMistery DUT(d, a, b, c); // instantiate the DUT

  initial begin
    count = 3'b0;                     // initialize count to 000 binary
    $display("time  a b c  d");      // truth table header
    $monitor("%4d %2d %1d %1d %2d", // output formatting
      $time, a, b, c, d);            // signals to be output
    #40 $finish;  // time goes 0 to 40 in steps of 5, then ends
  end

  always begin
    {a, b, c} = count;   // 3-bit count goes to 1-bit a, b, c
    #5 count  = count+1; // increment count every 5 time ticks
  end
endmodule


// CPEN 230L Lab 6a, Combinational Logic
// Firstname Lastname, mm/dd/yyyy
// File: logicMistery.v

module logicMistery(W, X, Y, Z);
  output W;
  input X, Y, Z;

  // RTL model
  assign W = (X || !Y) && Z;
endmodule
```

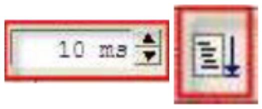
During-Lab do the following to simulate the Verilog design using ModelSim:

A. Create folder \Documents\CPEN230L\Lab6\Lab6a\.
B. Use ModelSim to create a Project. Select File -> New -> Project. This opens the Create Project dialog window.
   a. Set the Project Name to be "logicMistery_tb".
   b. Set the Project Location to be folder Lab6a that you created above.
   c. Leave the Default Library Name set to "work".
   d. Click OK. You should see an "Add items to the Project" window.
C. Create a testbench file.
   a. In the "Add items to the Project" window, click "Create New File".
   b. In the resulting "Create Project File" window, enter File Name "logicMistery_tb" and Add file as type "Verilog". Click OK. The new design file will appear in the project window. Click "Close" to close the "Add items to the Project" window.
   c. Double click the logicMistery_tb.v file. You will see another window that allows you to input your Verilog statements. This is ModelSim's plain text editor.

d. Copy the Verilog code for "module logicMistery_tb" above (not including the "module logicMistery" code) to the input window. This is your testbench code.
e. Select File > Save.

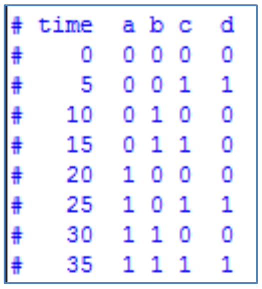
D. Create a new Verilog file that will contain module logicMistery, the circuit to be tested.
   a. Select File -> New -> Source -> Verilog to create a new Verilog file.
   b. Copy the Verilog code for module logicMistery into the Verilog file window.
   c. Select File -> Save and save file "logicMistery.v".
   d. Notice that top-level menu options can change depending on the active window pane. For example, when the logicMistery.v editing pane is active there is a "Source" menu option at the top of ModelSim. Click in the Project pane and see the Source menu option change to a Project menu.
   e. Select Project -> Add to Project -> Existing File. You will see the "Add file to Project" dialog window which allows you to add a file to the current project.
   f. Click Browse and then select logicMistery.v.
   g. Click Open. On the Add file to Project window click OK. You should have 2 files in your Project, logicMistery_tb.v and logicMistery.v. In the Project pane Status column the question marks mean the files haven't been compiled into the project or the source has changed since the last compile.
   h. Compile the files. Select Compile -> Compile All or right click in the Project pane and select Compile -> Compile All. In the transcript window you should see "2 compiles, 0 failed with no errors."

F. Start simulation.
   a. Click the Library tab to see a window of all library units with "work" at the top. Expand "work" by clicking the + in front of it. Right-click on "logicMistery_tb" and select "Simulate". Right-click in the window where you see simulation signals a, b, c, count and d. Select Add to -> Wave -> Signals in Region.
   b. In the Wave pane, drag "/logicMistery_tb/count" to the top so the waveforms are in order from top to bottom: count, a, b, c, d.
   c. In the Run Length entry near the top of the user interface, change simulation time to "10 ms".
   d. Click on the Run button to the right of "10 ms" and see a "Finish Vsim" pop-up asking "Are you sure you want to finish?" Answer "No".



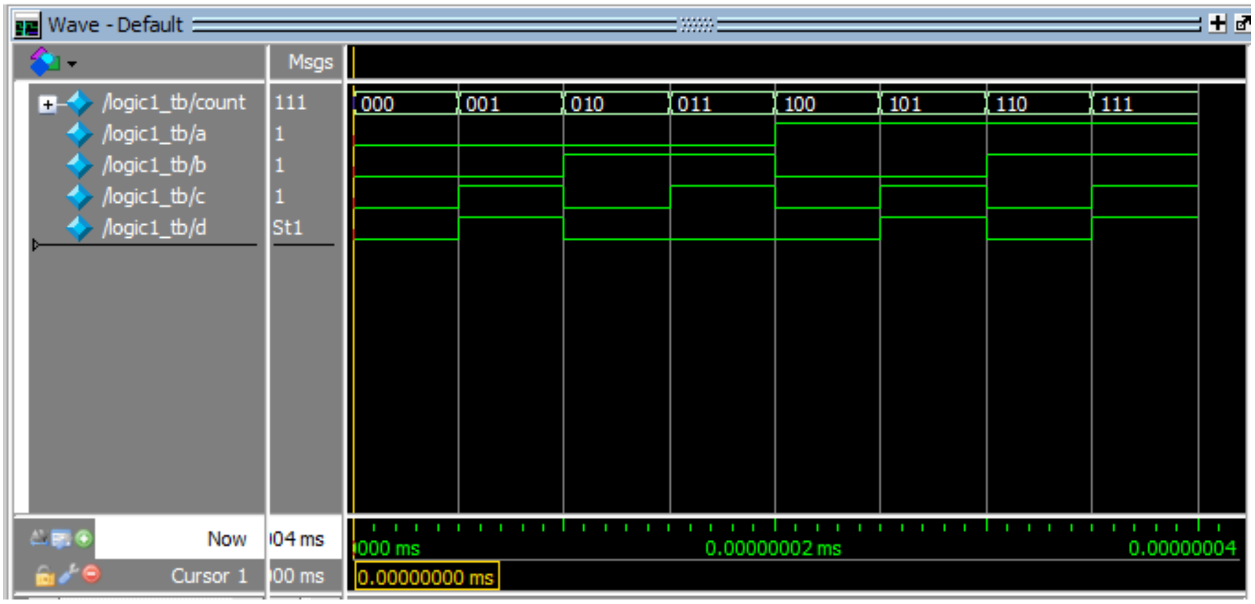
   e. If the waveforms are not visible, display them by clicking on the Wave tab. Right-click in the Wave window to see zoom options and select Zoom Full. You should see similar to the images below. Click any location under the waveforms and you will see a thick yellow line. Click different locations and observe the change in value of variables on the left. Also notice that the Transcript window contains the output of the $display and $monitor statements in the logicMistery_tb.v Verilog code. It shows the values of inputs and outputs as they change state.
   f. Do a screen capture to the entire ModelSim window.

G. Include two images like these in your lab report.
   - Transcript pane showing the resulting truth table



```
# time  a b c  d
#    0  0 0 0  0
#    5  0 0 1  1
#   10  0 1 0  0
#   15  0 1 1  0
#   20  1 0 0  0
#   25  1 0 1  1
#   30  1 1 0  0
#   35  1 1 1  1
```

   - Waveforms showing count, derived inputs a,b,c, and output d



Part 2: Majority Gate Design using Verilog and ModelSim

Pre-lab:

- Write Verilog code to describe a 3-input majority gate using an **RTL** model. Put your code in file majority.v. Use the code in file logicMistery.v code as a starting point.

- **The first lines of any Verilog file you write for this lab should be similar to this:**

```
// CPEN 230L Lab 6 part 2, Majority Gate using RTL coding style
// Firstname Lastname, mm/dd/yyyy
// File: majority.v
```

- Create file majority_tb.v to test module majority. Use the code in file logicMistery_tb.v as a starting point.
  - Debug majority_tb.v and majority.v using Icarus Verilog.
- Your code should include comments similar to the sample code above. A good guideline is to remember that your target reader is a student that hasn't done the lab and is learning from your code. Use comments to describe both **what** you are doing and more importantly **why**.

During-Lab

- Create directory majority_tb and ModelSim Project majority_tb to simulate and test module majority.
- Verify that the simulation waveforms and Transcript window output are as you expected.
- **Show your code and results to your lab instructor.** Implement any change suggestions made and remember them for similar future efforts.
- In your lab report, include Wave and Transcript pane images similar to the examples above. Also include your **well-commented well-formatted source code** as described in the Lab Report Template.

Part 3: Full Adder Design using Verilog and ModelSim

Pre-Lab

- Write Verilog code in file fullAdder.v to describe a full adder using an **RTL** model. In file fullAdder_tb.v create test bench code for module fullAdder.
    - Debug fullAdder_tb.v and fullAdder.v using Icarus Verilog.

During-Lab

- Create directory Lab6c and ModelSim Project fullAdder_tb to simulate and test module fullAdder.
- Verify that the simulation waveforms and Transcript window output are as you expected.
- In your lab report, include Wave and Transcript pane results similar to the examples above. Also in case you made any change to your prelab Verilog code make sure to include your new Verilog code.
- Exit ModelSim and take a backup copy of all your files with you.

<u>Pre-Lab Deliverables</u>

Before the lab starts, please hand in the following documents:

Part 1

- A snapshot of the table generated compiling and executing the files logicMistery.v and logicMistery_tb.v

Part 2

- majority.v and majority_tb.v
- A snapshot of the table generated

Part 3

- fullAdder.v and fullAdder_tb.v
- A snapshot of the table generated

NOTE: The penalty for not handing in the prelab deliverables is 50% of the lab grade