3C7 Digital System Design

Lab Session B

Aim:

The purpose of this session is to simulate and explore some combinational logic circuits in Vivado. It will also revise setting up projects in Vivado, simulating and using the waveform viewer.

Learning Outcomes:

On completing this lab session you will be able to:

- Routinely set up new projects to simulate testbenches for modules under test
- Solve compilation errors for Verilog modules
- Write simple Verilog testbenches
- Build circuits from existing Verilog modules by building a hierarchy

Note: (applies to lab machines)
Copy any required files from Blackboard to a new directory on your machine, say *C:/temp/LabB*. Create the project also in the local directory of your machine (if you are using the lab machine, this helps Vivado to utilise the SSD on them for improved speed. In this case, copy the folder to your network location at the end of the lab so that you can access them later.)

2-bit comparator

This block compares two 2-bit numbers to check for equality. It uses two 1-bit comparators to implement this functionality and ANDs the output (refer to Lecture 3). You may assume the 1-bit comparator has been tested and operates correctly.

Instructions:

1. Create a new project containing eq1.v, eq2.v and eq2_tb.v
2. You may find it useful to open a blank MS Word document to paste waveforms and make notes in as you go along.
3. Refer to page 40 in Chu for Verilog operators if you need this to help you understand the code.
4. Prepare to start the simulation (Hint – back to Lab A again if unsure). Is the compile and simulation successful? Fix any errors and make note of any code changes you made. Were the compilation messages helpful? (Get used to this!)
5. Set up the waveform viewer to allow you track the behaviour of the inputs and outputs of the 2-bit comparator.

6. Run the simulation to completion. You will notice that no output is directed to the console.
7. Use the **$monitor** function to monitor the inputs and outputs on screen by adding extra code to the testbench. [You will need to check page 199 of the Chu Book to see how to use **$monitor**]
8. Recompile your code by closing and restarting the simulation. Run the simulation to completion again. Save a waveform for your submission.
9. Is this block working? If not – where is the problem? Make note of any code changes you make to fix the design (if necessary).
10. When you believe the design is working, rerun the simulation and save a waveform that displays the testbench signals appropriately to demonstrate that you have run the full testbench and that the design is working. Include this waveform in your submission. Explain why (or if) you believe the circuit is working.
11. Copy the output that was sent to the Vivado Tcl console screen (due to your **$monitor**) and include this in your submission too.
12. Consider how extensively you have tested the 2-bit comparator with this testbench as provided. What recommendations would you make, if any, to extend the test?

8-bit gate-level greater-than-or-equal-to circuit

You are going to use what you have learnt so far to make a slightly more complex circuit. The greater-than or equals circuit compares two inputs, a and b, and asserts an output when a is greater than b, or if a is equal to b. You must create a 8-bit greater-than circuit from the bottom-up and use only gate-level logical operators. You are given a step-by-step approach to take below.

Instructions:

1. Start a new project for this circuit.
2. Derive the truth-table for a 2-bit greater than circuit and obtain the logic expression in SOP form. Based on this logical expression, write the Verilog code for a 2-bit greater than circuit using only logical operators. (Refer to table of logical operators in Chu book.)
3. Design a test strategy for the 2-bit greater than circuit. What testvectors will you use? Using the previous testbench as a template, write a testbench for this new circuit. Simulate the testbench to verify the operation of your code. Save a waveform that clearly demonstrates the behaviour of your circuit. Does it work? Why? Could you have used more or greater testvectors? Consider how adequately tested you think the module is.
4. Use this new 2-bit greater than circuit along with your 2-bit comparators (and a minimal amount of logic to glue them together) to construct a 8-bit greater than or equals circuit. Explain your approach clearly, preferably with the help of a diagram. (Note: you cannot just design it directly – you MUST use the 2-bit modules as building blocks)

5. You should then draw a block diagram of the blocks you need for your circuit. Then derive the structural Verilog code according to your diagram (i.e. signal names consistent etc.). Include the block diagram in your write up and clearly explain how you have used the other modules to construct the overall circuit.
6. Design a testbench for the 8-bit greater than or equals circuit. What testvectors will you use? Using the previous testbench as a template, write a testbench for this new circuit. Simulate the testbench to verify the operation of your code. Save a waveform that clearly demonstrates the behaviour of your circuit. Does it work? Why? Could you have used more or greater testvectors? Consider how adequately tested you think the module is.
7. In your submission, include the code for the 2-bit greater-than circuit and the code for your 8-bit greater-than or equals circuit along with waveform views demonstrating the correct operation of your circuits.

Submission:

You must submit the following parts, contained in a single PDF, via Blackboard:

- Labelled waveforms demonstrating the design is working where instructed
- Block diagram and brief supporting explanation of how your 8-bit greater than or equals circuit works
- Code for eq2.v, with commenting where any change was made
- Code for 2-bit and 8-bit greater than circuits, plus testbenches
- All code MUST be suitably commented

Name the file LabB_surnameinitial.pdf, e.g. LabB_shankers.pdf.

Deadline:

Group A (Tuesday labs): Tuesday 14th February 2023.
Group B (Friday labs): Friday 17th February 2023.
All submissions are via blackboard.

Plagiarism

Plagiarism is interpreted by the University as the act of presenting the work of others as one's own work, without acknowledgement.

Plagiarism is considered as academically fraudulent, and an offence against University discipline. The University considers plagiarism to be a major offence, and subject to the disciplinary procedures of the University.
Visit http://tcd-ie.libguides.com/plagiarism

All students must complete the TCD Ready Steady Write plagiarism tutorial and sign a declaration when submitting course work, whether in hard or soft copy or via Blackboard, confirming that you understand what plagiarism is and have completed the tutorial. If you read the information on plagiarism, complete the tutorial and still have difficulty understanding what plagiarism is and how to avoid

it, please seek advice from your College tutor, your Course Director, your supervisor, or from Student Learning Development.

Plagiarism will result in loss of ALL marks for this assignment.