ECE 3544: Digital Design I
Project 1 (Part A): Introduction to ModelSim Environment

Objectives
After completing this project, you should be more familiar with the ModelSim tools. You will gain experience with timing models and with the use of test benches for simulating digital circuits in Verilog.

Requirements
You must have ModelSim ALTERA STARTER EDITION 10.3c (or higher) installed on your computer. The instructions in this assignment are consistent with that version of ModelSim. Even if these instructions are consistent with other versions of ModelSim, you should use the version indicated here.

The DE1-SoC Board is not required for this project. This project involves only simulation.

Project Description
Altera ModelSim is one of the most common simulation tools for digital logic design. In this project, you will learn how to use ModelSim to run simulations. First, you will simulate two pre-designed circuits:

- The 74138 3-to-8 decoder, which was written using a structural model using primitive gates, and
- A 4-to-16 decoder that uses two of the 3-to-8 decoders as structural blocks, along with an inverter as an additional primitive gate.

You will use a test bench to simulate each decoder and produce output waveforms.

After you gain proficiency with using ModelSim to simulate existing designs, you will perform the design of a simple digital logic circuit that you can model and simulate using the ModelSim tools.

What is a test bench?
The first thing you should understand is the difference between design files and test bench files.

Imagine you have real hardware that you want to verify. You might use probes to supply inputs to your hardware and to observe its outputs. In this way, you could verify that the hardware is behaving as expected. In the case of simulation, you have the Verilog description of the hardware instead of the hardware itself, but this need not stop you from simulating the verification process.

The module that the designer uses for applying test inputs is called a *test bench*. The test bench normally consist of an instance of your Verilog model (normally called the "device under test") and a procedure containing a sequence of inputs with specified timing. You can use simulation tools such as ModelSim to run the test bench and observe the output. For more complex models, the test bench can both apply inputs and check the outputs. We will use waveforms to see the generated output.

It is important to remember that test bench files will not be synthesized to generate real hardware; they are only used for simulation.

Instructions

Step 1: Make a new project.

Follow the instructions in Step 1 of the document "Creating and Simulating Verilog Source Files in ModelSim" to make a project for the Project 1 Starter Files. (You'll find this and other useful documents in the ECE 2504/ECE 3544 Laboratory Manual.) *Make sure that you follow the recommendations in that document for locating and naming the folder where you place your starter files – namely, create a working folder having a short, space-free name in a folder that is close to the root directory, and not in one having a long pathname that contains spaces*.

Study the example files to make sure that you understand how each model works in Verilog. First, look at sn74138.v. This is a Verilog structural model that describes a 74138 chip (a 3-to-8 decoder) using primitive logic gates. **Based on the structure of the 74138 decoder module, produce a neat gate-level schematic of this module and include it in your report.**

Next, look at decoder4to16.v. This is the Verilog model for a 4-to-16 decoder made from two 3-to-8 decoders and an inverter. Observe how it works. Specifically, take note of the manner in which this module instantiates two of the 74138 chips. **Based on the structure of the 4-to-16 decoder module, produce a neat block diagram of this module and include it in your report.** Instead of drawing another gate-level logic circuit that includes the gate-level schematic you drew previously, use a block symbol to represent the 74138. Here is an example of what a 74138 block might look like:

Now open the test-bench files for each decoder and study the details to understand how they were written. **Compare the two test benches, and briefly describe the differences between the two in your report.**

Step 2: Compile and simulate the starter files.

Follow the instructions in Step 2 of the document "Creating and Simulating Verilog Source Files in ModelSim" to compile the files in the project.

Follow the instructions in Step 3 of the document to simulate the 74138 chip. Remember to perform the simulation on the test bench, and not on the source file.

Repeat the instructions in Step 3 of the document to simulate the behavior of the 4-to-16 decoder.

Include screenshots of your results showing proper operation of the 3-to-8 decoder and 4-to-16 decoder in your final report. If you have the ability to print to PDF, the waveforms can be printed to a PDF file in black-and-white.

Step 3: Create your own models and test bench

For this step, you should create the circuit described below using *only Verilog's built-in gate primitives.* It should be organized in a general manner similar to the model of the 3-to-8 decoder.

The circuit accepts two 2-bit inputs – player_a[1:0] and player_b[1:0] – and provides three outputs – player_a_wins, player_b_wins, and tie_game. The circuit should be designed as a "judge" for the game "Rock, Paper, Scissors." Remember that in this game, rock beats scissors, scissors beats paper, and paper beats rock. If both players make the same move, the game is tied.

Each of a player's possible moves should correspond to a two-bit value, as follows:

Player's Move	Rock	Paper	Scissors
Input Value	11	10	00

The value of the outputs player_a_wins and player_b_wins should equal one if the associated player wins, and should equal 0 otherwise. The output tie_game should equal 1 if the players tie, and should equal 0 otherwise. The outputs are mutually exclusive – only one of the three outputs can equal one for any combination of valid input values, and one of the three will equal one for every valid input combination. For example:

- If player_a = 11 and player_b = 00, then player_a_wins = 1, player_b_wins = 0, and tie_game = 0, since rock crushes scissors.
- If player_a = 11 and player_b = 10, then player_a_wins = 0, player_b_wins = 1, and tie_game = 0, since paper covers rock.
- If player_a = 00 and player_b = 00, then player_a_wins = 0, player_b_wins = 0, and tie_game = 1, since both players played scissors.

Use the starter files in the Step 3 folder, which include module declarations. Name your module and re-name your files according to the directions in the starter files.

Create a new project for this step and add your modules to it. You should also create a test-bench that instantiates your game module and applies all valid combinations of the inputs to it. You should include waveforms from the test-bench in your report that show the proper operation of your design.

Your modules and test bench should include proper header information, contain reasonable comments, and be neatly formatted. The starter files provide examples of header information; make sure that you fill in the header comments.

Project Submission
Your submission should include the following files:

- A project report. Your report should be in Word or PDF format. Include your PID in your report filename, *e.g.*, project1report_thmartin.pdf.

  Your report should contain the following elements:
  - A restatement of the assignment's objectives.
  - A section that addresses the questions posed in Step 1 of the project specification.
  - A section that describes the process by which you derived the design of your circuit and the manner in which you implemented it in Verilog. In particular, you should comment on how applicable you believe your design process and your method for implementing it in Verilog are to performing more general and larger-scale designs and implementations.
  - A section that shows the simulation of the decoders and the results of your game circuit testing. Discuss how you formulated the tests contained in the test bench that you had to generate, and how you verified the correctness of your implementation.
  - A section where you discuss your conclusions and the lessons you learned from the assignment.

- The Verilog files for your game module and test-bench. Name your files according to the convention described in the starter files.

All of the files should be zipped together and submitted on the Canvas assignment page along with your report. Include your PID in the name of your archive file, *e.g.*, project1files_thmartin.zip.