ECE 3544: Digital Design I
Project 1 (Part B): Simulation in the ModelSim Environment; Continuous Assignment Models

Student Name: Jonathan Lemarroy

Honor Code Pledge: I have neither given nor received unauthorized assistance on this assignment.

Jonathan Lemarroy

Grading: The design project will be graded on a 100 point basis, as shown below:
Manner of Presentation (30 points)

_____ Completed cover sheet included with report (5 points)

_____ Organization: Clear, concise presentation of content; Use of appropriate, well-organized sections (15 points)

_____ Mechanics: Spelling and grammar (10 points)

Technical Merit (70 points)

_____ General discussion: *Did you describe the objectives in your own words? Did you address the questions posed in the project specification? Did you discuss your conclusions and the lessons you learned from the assignment?* (10 points)

_____ Design discussion: *What was your approach to deriving the circuit you had to design? How does the design process that resulted in your continuous assignment model compare to the one you used to obtain a structural model that used primitive gates? How applicable is the design process you followed in this assignment to larger and more general designs?* (10 points)

_____ Testing discussion: *What was your approach to formulating your test benches, and how did you verify the correctness of your design and implementation?* (10 points)

_____ Supporting figures (20 points total)
- *Correct waveforms showing simulation of both decoder modules.* (5 points)
- *Correct waveforms showing correct operation of your design.* (15 points)

_____ Supporting files: *Do the submitted files produce the correct response?* (20 points)

_____ **Project Grade**

Purpose:

The objective of this assignment is to gain experience using dataflow operators to implement a continuous assignment model. And to compare the functionality and code structure of using primitive gates versus the continuous assignment model.

Project Description:

Part 1: Compare the differences of the sn74138 and the 4-to-16 decoder primitive gate implementations versus their continuous assignment implementations. Include screen shots of their testbench waveforms and comment on weather changing their implementation model affected the circuit's behavior.

Part 2: Re-design and simulate a circuit using the continuous assignment model that can determine the winner of a rock, paper, scissors, lizard, or spock game between two players. Each player's input consists of a 3-bit number with some of the possible combinations not being used. The outputs are a bit representing if player1 has won, another bit for if player2 has won, and a last bit if the game is tie. The bit representing a player should be logical true if they won or false otherwise.

Technical Design:

Part 1: No design necessary.

Part 2: The first thing was to determine mappings for the 3-bit input combinations to a chosen representation of rock, paper, scissors, etc. Through some thought process and some trial and error to find combinations could easily be applied using Verilog's dataflow operators, there was no clear evidence that the combinations had any effect on making an efficient design. Thus, the mappings chosen were the same as those found in Project 1A.

Table 1: Input mappings

Player's Move	Rock	Paper	Scissors	Lizard	Spock	*Not Used*
Input Value(s)	001	010	011	100	101	000, 110, 111

The next step was deciding the best operator to easily design this using the continuous assignment model. Because of the explicit statement not to use the exact design from Project 1A, the next easiest way was by comparing the concatenation of the player inputs to expected results found in **Table 2**.

Table 2: Player combinations

player1	player2	p1wins	p2wins	tied
ROCK	ROCK	0	0	1
ROCK	PAPER	0	1	0
ROCK	SCISSORS	1	0	0
ROCK	LIZARD	1	0	0
ROCK	SPOCK	0	1	0
PAPER	ROCK	1	0	0
PAPER	PAPER	0	0	1
PAPER	SCISSORS	0	1	0
PAPER	LIZARD	0	1	0
PAPER	SPOCK	1	0	0
SCISSORS	ROCK	0	1	0
SCISSORS	PAPER	1	0	0
SCISSORS	SCISSORS	0	0	1
SCISSORS	LIZARD	1	0	0
SCISSORS	SPOCK	0	1	0
LIZARD	ROCK	0	1	0
LIZARD	PAPER	1	0	0
LIZARD	SCISSORS	0	1	0
LIZARD	LIZARD	0	0	1
LIZARD	SPOCK	1	0	0
SPOCK	ROCK	1	0	0
SPOCK	PAPER	0	1	0
SPOCK	SCISSORS	1	0	0
SPOCK	LIZARD	0	1	0
SPOCK	SPOCK	0	0	1

But instead of comparing the concatenation of every combination for each output, with just a little analysis one can easily deduce that tie combinations only occur when the players both play the same move. There fore using just the equality (==) operator the ‘tie’ bit can be set. And with a little more analysis it can be easily seen that player2’s winning bit should only be 1 when both the tie and player1’s input bits are set to 0. Therefor player2’s bit can be set to the NOR of the ‘tie’ and player’s bits.

Analysis & Conclusion:

Part 1: Changing the implementation for the sn74138 and 4-to-16 decoder had no effect on their behavior, see **Figure 2** and **Figure 3**.

Part 2: To verify everything above was implemented correctly, the testbench used from Project 1A could be applied against this new implementation because they both maintained the same mappings from **Table 1**. And since the results found in **Figure 1** came out identical to those found in Project 1A this implementation is valid.

The thought process used for designing this implementation is useful/fast for larger scale designs, but as far as gate count and efficiency are concerned, this wouldn't necessarily give us the optimal implementation that could be determined if Karnaugh map's and primitive gates were used.

Appendix:

Figure 1: Game I/O simulation

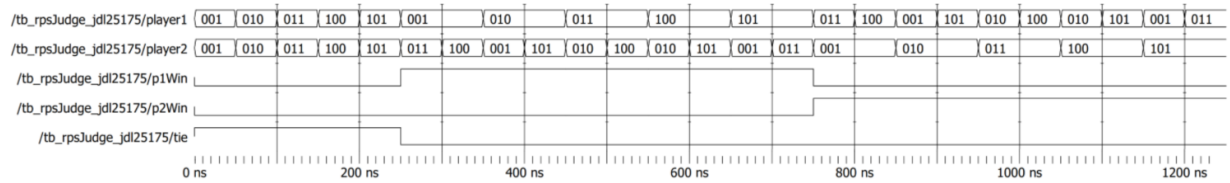

Figure 2: 74138 3 to 8 decoder simulation

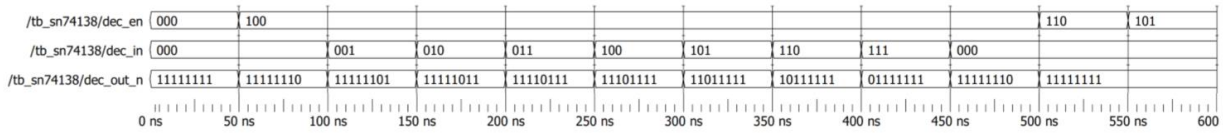

Figure 3: 4 to 16 decoder simulation