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

Student Name: Jacob Abel

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

---

**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 applicable is the design process you followed to larger and more general design and large-scale 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**

# Project 1A

ECE3544 CRN:82989

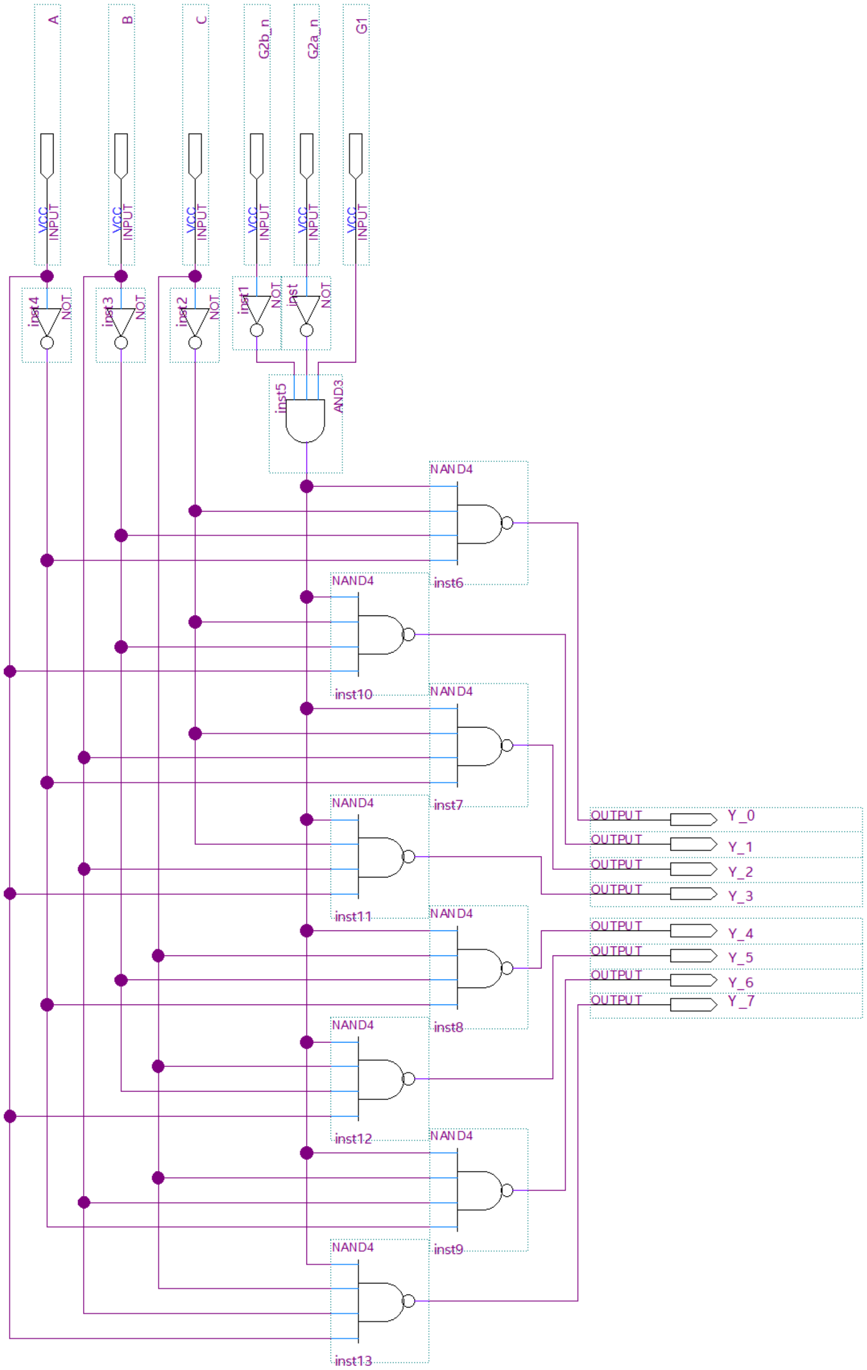
Jacob Abel

September 20, 2018

# Objective

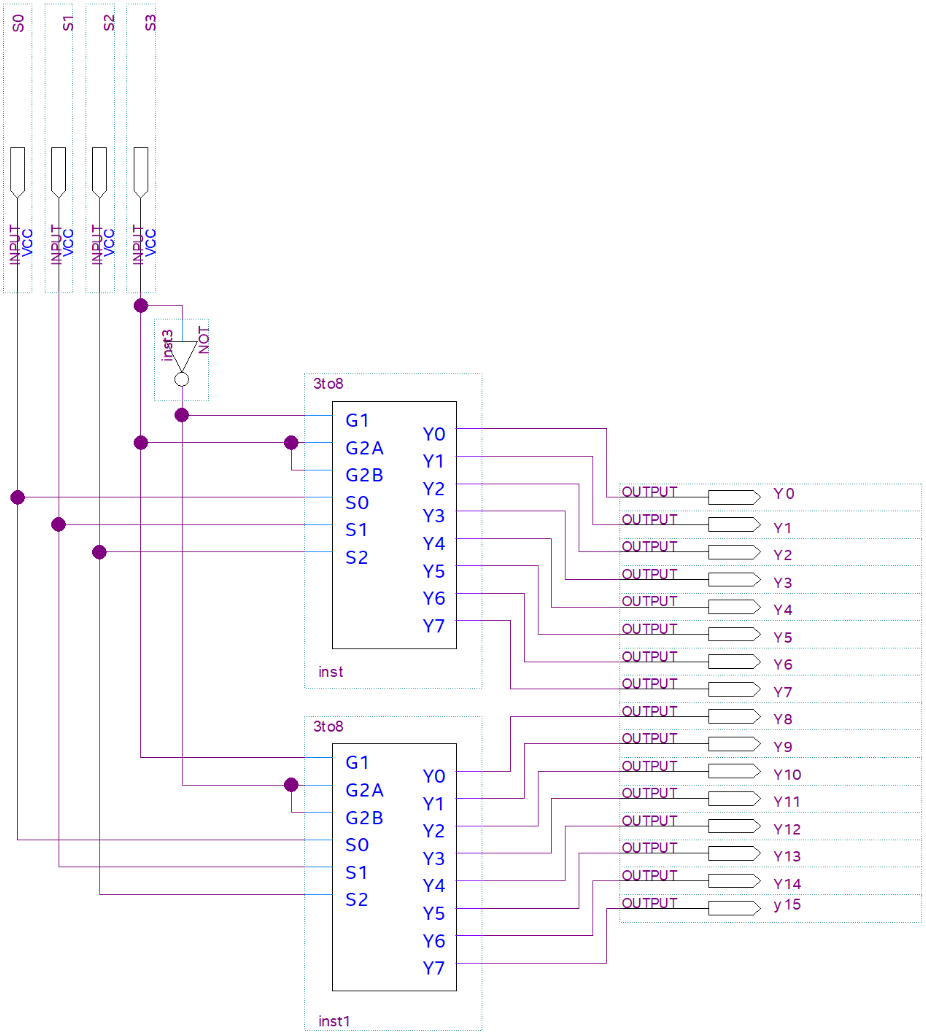
The objective of this project is to demonstrate basic competence with utilising ModelSim tooling and manipulating, designing, and simulating digital circuits written in Verilog.

# 3 to 8 Decoder Gate Diagram



# 4 to 16 Decoder Block Diagram

Note: $S_0$, $S_1$, and $S_2$ map to $A$, $B$, and $C$ respectively.

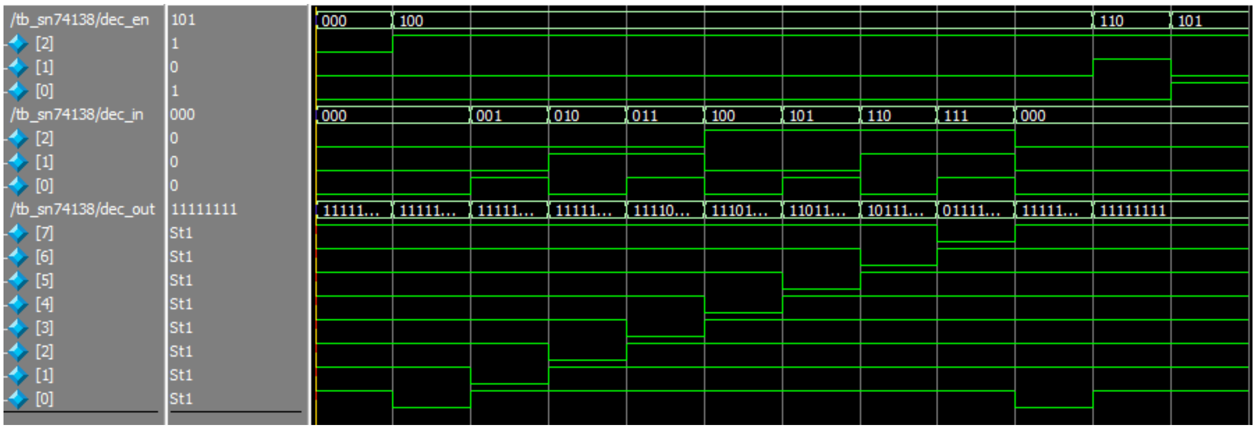


# Test-bench Differences

The 3-to-8 decoder and 4-to-16 decoder test-benches are fairly similar however they differ largely in the implementation of the test coverage. The 3-to-8 decoder test bench has two initial blocks which execute in parallel. This allows the possible variations of each variable to be listed together and execute all the permutations. The 4-to-16 decoder uses a for loop to loop through all possible input values.

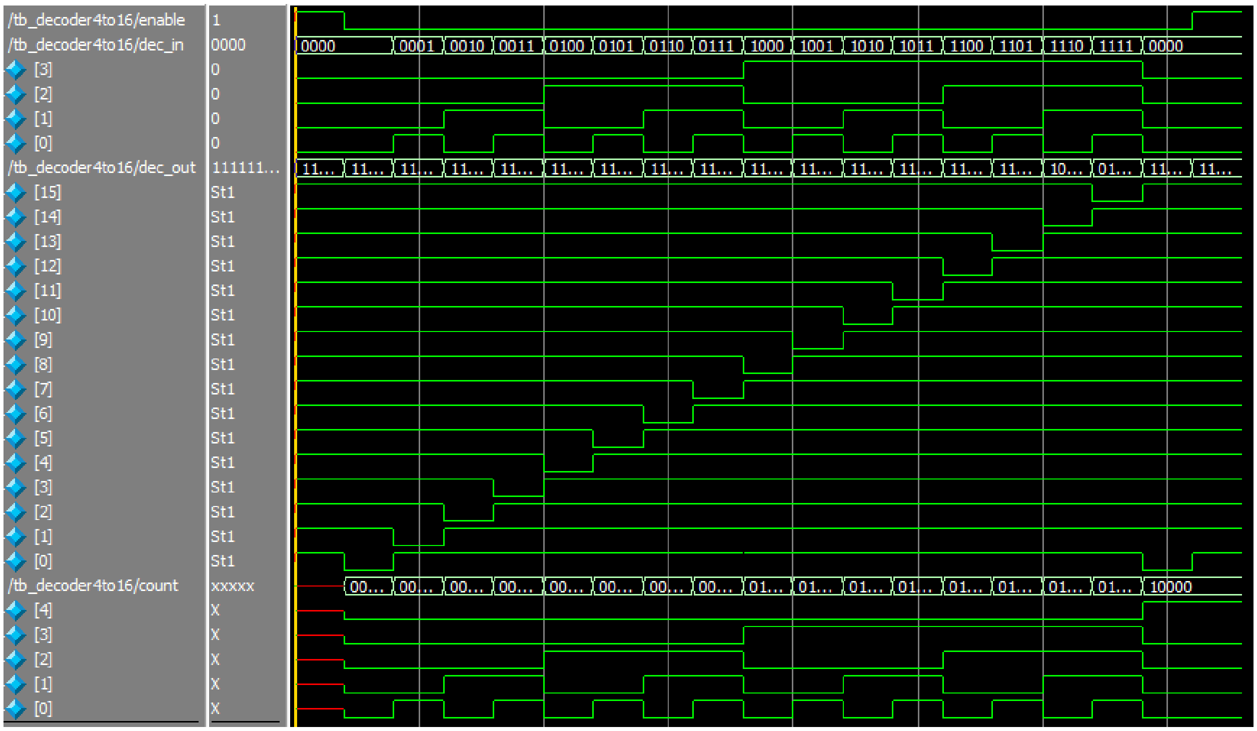
The 3-to-8 test-bench's method is better when multiple variables have to be tested however the 4-to-16 test-bench's method is better when many inputs of one variable have to be tested. The two methods could be combined to make the 3-to-8 test-bench more concise.

# Decoder Simulation



```
/tb_sn74138/dec_en   101
  [2]                1
  [1]                0
  [0]                1
/tb_sn74138/dec_in   000
  [2]                0
  [1]                0
  [0]                0
/tb_sn74138/dec_out  11111111
  [7]                St1
  [6]                St1
  [5]                St1
  [4]                St1
  [3]                St1
  [2]                St1
  [1]                St1
  [0]                St1
```

74138 3-to-8 Decoder Test-Bench Waveform



```
/tb_decoder4to16/enable   1
/tb_decoder4to16/dec_in   0000
  [3]                     0
  [2]                     0
  [1]                     0
  [0]                     0
/tb_decoder4to16/dec_out  111111...
  [15]                    St1
  [14]                    St1
  [13]                    St1
  [12]                    St1
  [11]                    St1
  [10]                    St1
  [9]                     St1
  [8]                     St1
  [7]                     St1
  [6]                     St1
  [5]                     St1
  [4]                     St1
  [3]                     St1
  [2]                     St1
  [1]                     St1
  [0]                     St1
/tb_decoder4to16/count    xxxxx
  [4]                     X
  [3]                     X
  [2]                     X
  [1]                     X
  [0]                     X
```

4-to-16 Decoder Test-Bench Waveform

# Rock Paper Scissors Design

The project requires that a combinatorial circuit be designed that computes the results of a ”Rock, Paper, Scissors” game. The circuit accepts two inputs: `player_a[1:0]` and `player_b[1:0]` and outputs three 1-bit outputs: `player_a_wins`, `player_b_wins`, and `tie_game`. The standard rules of the game apply:

- Rock beats Scissors
- Scissors beat Paper
- Paper beats Rock

The values of the input moves are as follows.

| Move | Rock | Paper | Scissors |
|---|---|---|---|
| Input Value | 11 | 10 | 00 |

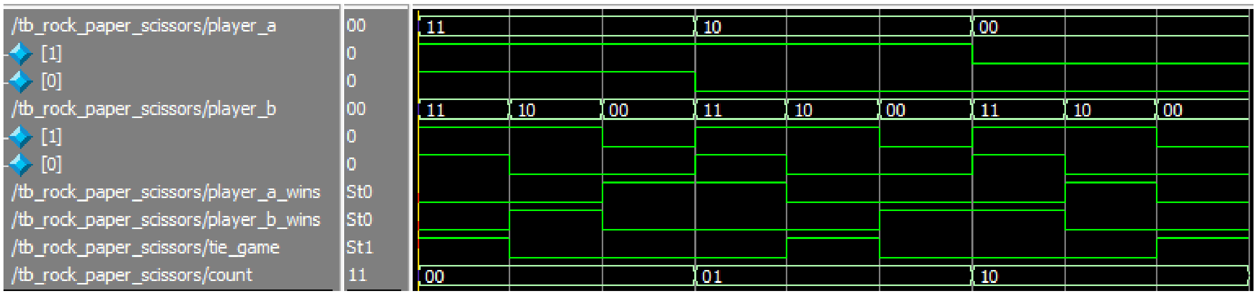
## Design Process Reflection

The design process for this project was rudimentary however due to the simplicity of the project, it had no impact on the overall design. The design of the project was oriented primarily around rapid development and readability rather than optimisation as any optimisations that could be done at this level are trivial for a compiler.

The strategy for design was to design the system from the bottom up rather than top down. Inputs were composed into moves (Rock, Paper, Scissor) and move matchups were evaluated and grouped together by their win state (A, B, tie). The grouped matchups were or-ed together and outputted.

While this approach has some connection to large scale design, it is largely relegated to the mid and lower levels of project design as top down approaches tend to be more valuable at large scales. This methodology does however serve to compose rudimentary elements into logical blocks that can be integrated into large scale methodologies.

# Rock Paper Scissors Simulation


/tb_rock_paper_scissors/player_a
[1]
[0]
/tb_rock_paper_scissors/player_b
[1]
[0]
/tb_rock_paper_scissors/player_a_wins
/tb_rock_paper_scissors/player_b_wins
/tb_rock_paper_scissors/tie_game
/tb_rock_paper_scissors/count
00
0
0
00
0
0
St0
St0
St1
11
11
10
00
11
10
00
11
10
00
11
10
00
00
01
10

Rock Paper Scissors Simulation

The simulation demonstrates that the code matches the intended behaviour. The test bench was designed to cover all possible inputs while minimising repetition. An initial block was used for each variable and a for loop was used for secondary variables(`player_b`) to fill the repetitions for all states of the primary variable (`player_a`).

# Conclusion

This project served as a refresher in Verilog and an introduction to ModelSim. During this project there were a number of issues with ModelSim configurations, mistakes with Verilog syntax, and basic logic mistakes however the project overall was a success. Unfortunately as this project was started far later than intended, many bugs came from being rushed and the project ended up taking more time and effort as a result. Luckily as this is the start of the semester, this project can serve as a reminder to start early from now on.