Simon: Two Player Memory Game

Naomi Felix Monanci

Datapath Implementation, Controller Implementation, Modular Testing
Write-up Question 1: Briefly explain the test cases you wrote for your testbenches. What behaviors do your test cases verify, and why did you focus on those cases?

- For the SimonDatapath testbench, we checked the memory and LED behavior. One test involved writing specific patterns into the memory at different addresses and then reading them back to ensure the memory module stored and retrieved the data correctly. Another test verified the LED output behavior by checking that the module correctly switched between displaying the stored pattern and the player input based on the control signal, ensuring the LEDs reflected the correct game state.

- For the SimonControl testbench, we first checked the reset behavior to ensure the FSM correctly initialized its state and outputs, to ensure that the system started in a safe state Next, we tested the state transitions, including conditions for advancing counters, legal pattern input, and incorrect guesses, to verify that the FSM logic matched the expected game rules/logic. Finally, we tested the control outputs in each state, ensuring that the datapath received the correct instructions and that mode_leds accurately represented the current state of the game.

Write-up Question 2: Did you have to modify your design from Week 1? If so, what did you change?
- We had to modify some changes to our design from Week 1. In our FSM, we renamed some variables for better clarity, like making stars equal legal. In our Datapath Diagram, I relabeled some of the input and output signals for better clarity.

Module Combination and Testing: Connecting the Datapath and Control Units, Running the TA Testbench, Writing Your Own Testbench
Write-up Question 3: Explain your functional tests for the overall Simon module. What use cases do they test, and how do they work?

- For the overall Simon module, we tested the integration of the datapath and control modules to ensure that the game operated as expected. We focused on the Hard level mode version of Simon, where any 4-bit input pattern is considered legal. We manipulate the clock, reset, level, and pattern inputs to simulate different gameplay sequences. In Input mode, the user inputs a sequence of patterns, so we want to verify that the patterns are correctly written into memory and then the LED output reflects the current input. In

Playback mode, the game is supposed to play back the stored sequence. So, we want to check that the LED outputs correctly displays each stored pattern in the correct order and that the playback counter increments appropriately so that we know when to move on to the next state, which is Repeat mode. In Repeat Mode, the user tries to repeat the sequence. So, we test for both correct and incorrect guesses here to make sure the game transitions appropriately to the next state based on the user's guess. In the Done mode, we verify that the user is able to also play back the stored sequence to see what they got wrong.