Please attach this cover sheet to your completed homework

ECE324 Homework7 (take-home Quiz)
SystemVerilog Finite State Machine Logic and Simulation using Tasks

Name: Alex Blake

Since this is a take-home quiz, please work alone – do not collaborate!

Exercise	Course outcome	Grade
Homework7	2.a, 7.b	/30
Homework7 Extra Credit	2.a, 7.b	/5
	TOTAL:	/30

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.

7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

In this assignment, students are asked to use the coin detector created from the last assignment and finish building a vending machine project in SystemVerilog which will then be tested by a testbench file that is partially written. Students will utilize all of the methods and tools learned previously in the class such as synchronous logic design and flip-flops as well as tasks in the testbench.

Procedure

Before any code can be modified, first download the starter code files ('VendingMachine.sv' and 'VendingMachine_tb.sv'). Add these files to the correct portion of the new Vivado project as well as adding 'coinDetector.sv' to the project as well. The project should produce a vending machine like the one seen in Figure 1.

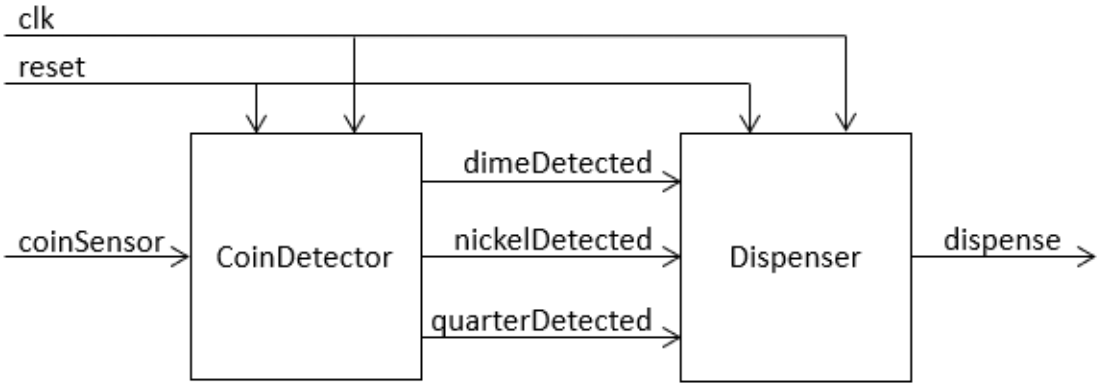

Figure 1: Vending Machine Diagram (Homework 7 Handout, pg. 2)

Now the code is ready to be modified; In the 'VendingMachine.sv' file, add the rest of the states to the FSM towards the bottom of the file. The states are already declared as a big hint as well as one of the states is written already which acts like a model for the rest. This should be very straightforward.

The second step is to finish the testbench file ('VendingMachine_tb.sv'). This file, just like in the previous step, is partially completed. The only requirement for this file to work is to finish the 'RDDDDDDD' task and make the 'RQNQNQNQNQNDD' task. Using the tasks above these as a guide, this step is also very straightforward. Make sure that the last task still starts with a reset.

The tasks require the depositCoin task which will require the expected dispense value, so make sure to do the calculations along with the task's lines to make sure the assert statement will catch errors if there are issues with the FSM.

The final step is to verify the assert statements in the testbench work. This is done by simulating the project as-is and verify no errors are thrown. After this is true, sabotage one of the states in

the FSM that only get caught once. Verify that an error is thrown. This completes the assignment.

Results

Finishing both the design file and testbench file required very little effort and no issues occurred when writing them. The files were simulated and no errors were thrown.

However, I had issues when sabotaging the FSM; if the state was changed that only occurred once towards the end of the simulation, it was not showing up. After some troubleshooting and looking at the waveform, I realized that the issue was that the simulation was ending before the task was finished. This meant that my error was not found. I simply changed the timescale at the beginning of the document, which fixed my problem. The simulation console now reported the errors exactly as expected.

Figure 2 shows the waveform output from the simulation and Figure 3 shows the console output. Both of these are with the correct FSM and should produce zero errors, which is exactly what happened.

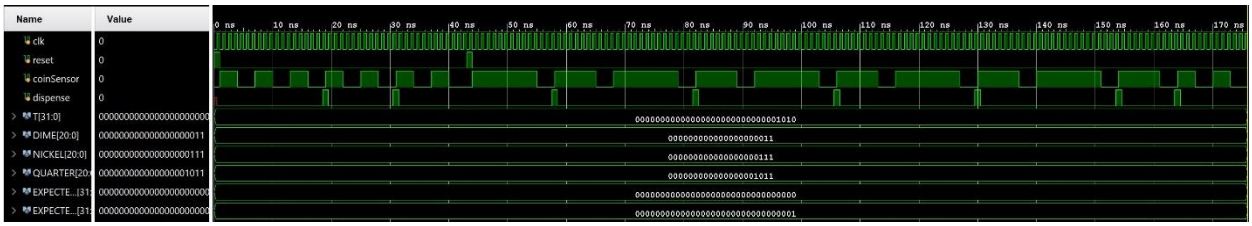

Figure 2: Waveform Output (not Sabotaged)

```
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'VendingMachi
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:06 ; elapsed = 00:00:06
```

Figure 3: Console Output (not Sabotaged)

The FSM was then sabotaged by changing the next state ‘credit20’ so that it goes to ‘credit15’ instead of ‘credit0’ when a nickel is detected. This error should only because one wrong dispense value which should cause only one error. Figure 4 shows the waveform output as well as Figure 5 shows the console output with the expected error.

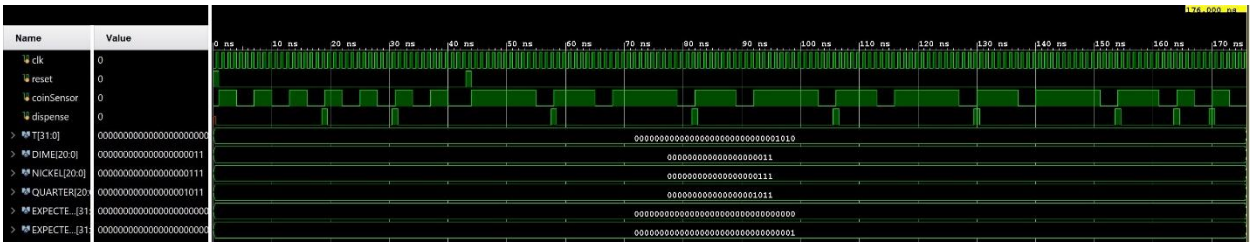

Figure 4: Waveform Output (Sabotaged)

```
# run 1000ns
Error: dispense is 1, but expected 0
Time: 170 ns  Iteration: 0  Process: /VendingMachine_tb/depositCo
INFO: [USF-XSim-96] XSim completed. Design snapshot 'VendingMachi
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:12 ; elapsed = 00:00:11
```

Figure 5: Console Output (Sabotaged)

As seen in Figure 5, the test bench threw an error at 170ns where it did not expect a dispense but it dispensed anyways. This is because the state went to the wrong next state and the state then thought it had the wrong value due to this changed line. This completes the homework.

Conclusion

This assignment was great practice for students to hone their skills with SystemVerilog and to modify and implement different projects together to create a synchronous logic circuit that creates a FSM to control a vending machine. These tools used are going to be the building blocks for any larger design and most likely required for the final project. It was also a great introduction to tasks in a testbench which will also be useful in development of the final project.