Please attach this cover sheet to your completed homework

ECE 324 Homework6: FSM design and simulation

Name: Alex Blake

Exercise	Course outcome	Grade
Homework6	2.a, 7.b	/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 lab, students were asked to use SystemVerilog and previously learned skills to design a Finite State Machine (FSM). This FSM would detect different coins. It a great example of how FSMs can be used and how they can be implemented with an HDL.

Procedure

The first step for the assignment is to obtain the almost completed .sv file for the FSM. Add this to a new Vivado project.

After the project is setup, read the code and analyze how it works. The code should be ready to go except for the state machine which will control the three outputs (dimeDetected, nickelDetected, and quarterDetected) and be controlled by two inputs (sensor and changeCoinTest). Figure 1 shows a block diagram of the entire module and Figure 2 shows the behavior of the FSM to be implemented.

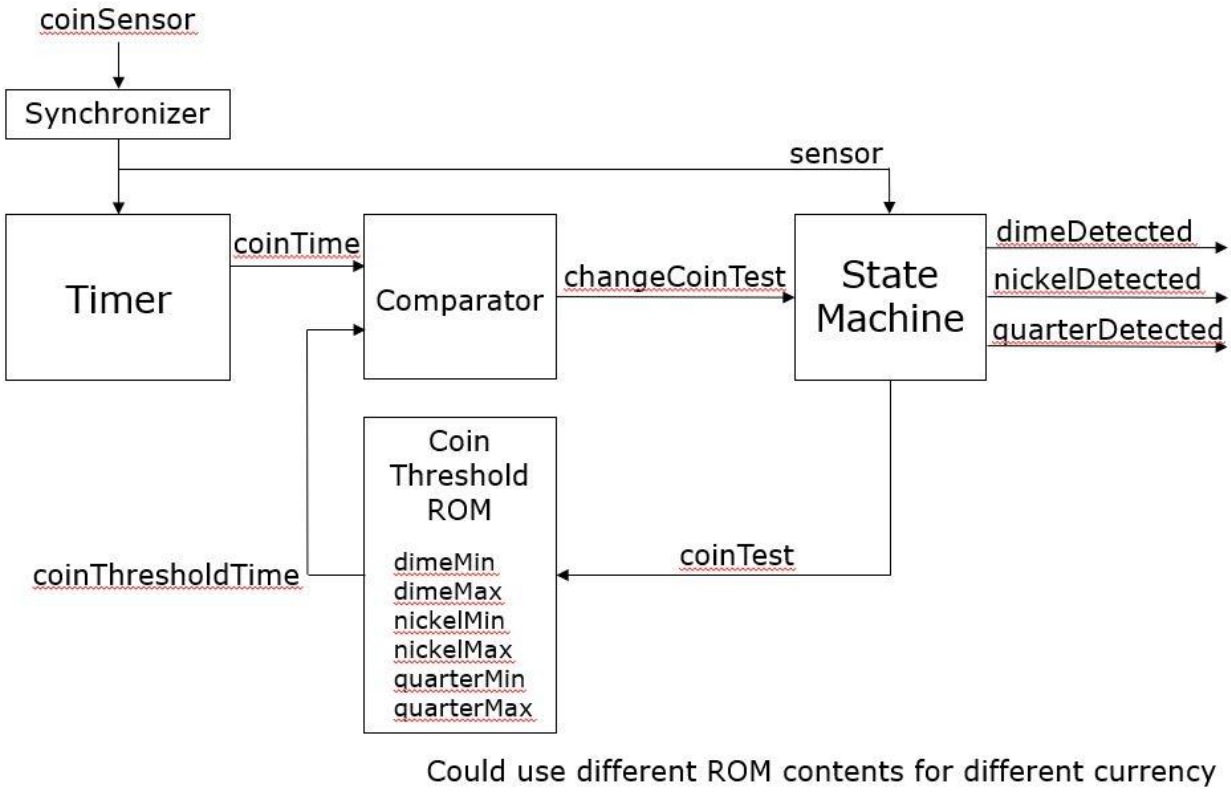

Figure 1: Block Diagram of Coin Detector (Lecture 7 – Slide 24)

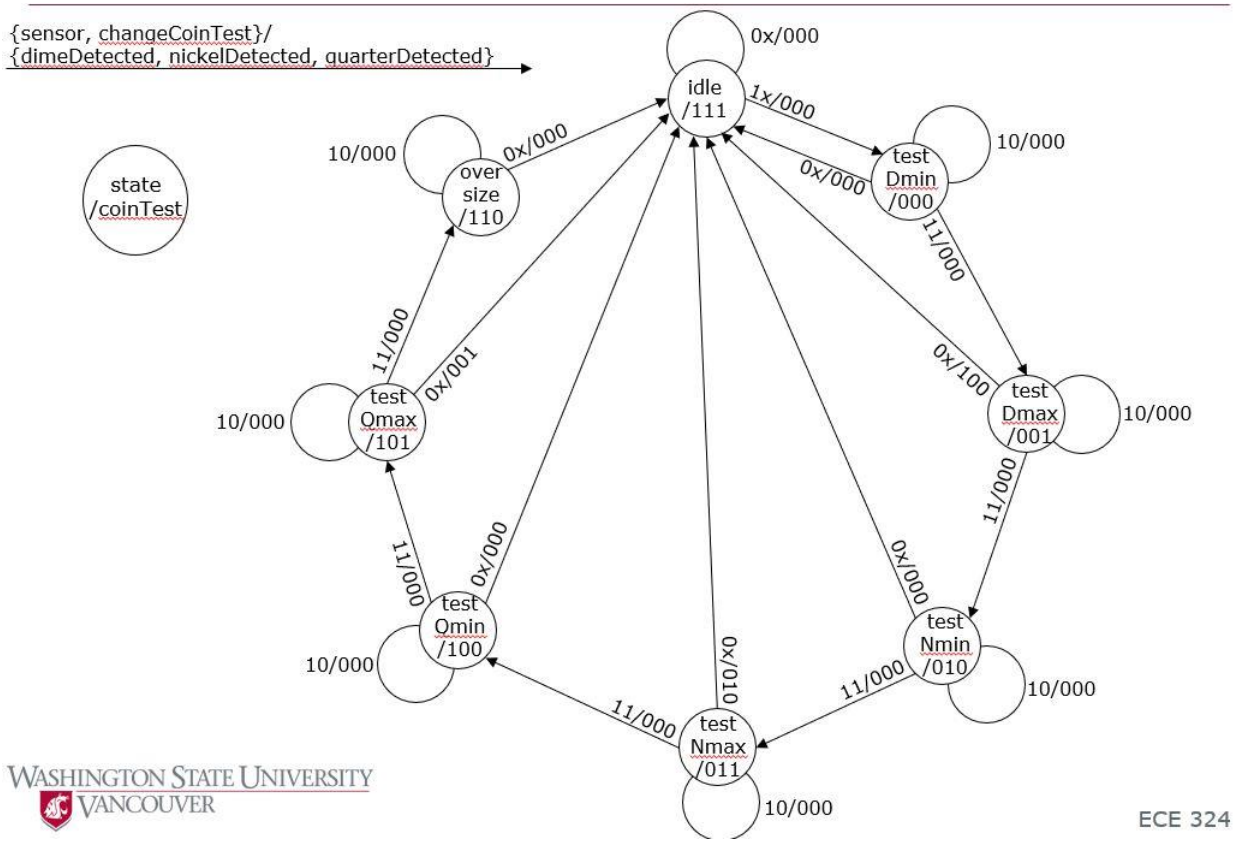

Figure 2: Diagram Describing FSM to be Implemented (Lecture 7 – Slide 25)

To do this, a case statement can be used to determine a state, and set a stateNext variable to update for the next clock cycle. Make sure each case creates the correct outputs and change depending on the correct inputs as Figure 2. This completes this section of the assignment.

The last part is to implement a testbench for the module. The test bench should test every possible state action to ensure everything works correctly. Within this, make sure to add assert statements to make sure the outputs are correct. This concludes the assignment.

Results

The first part of this assignment went well and was not an issue; the FSM was easy to implement using a combination of Figures 1 and 2.

The problems began to rise when trying to run the test bench. The test bench was setup correctly if there was no input latency from when the signal goes high on the input of the module vs when it affects the FSM inside. Since the signal goes into a flip-flop buffer to synchronize and minimize metastability, it takes 2 clock cycles of delay to affect the circuit. This means the circuit will output exactly what is expected 2 cycles late.

The waveform is correct and doesn't require any changes, but since the delay exists, the assert statements were off by 2 clock cycles and caused quite a headache. Once that was determined, I added a 2-cycle delay within the for loop of the code which fixed that issues. Figures 3 through Figures 6.

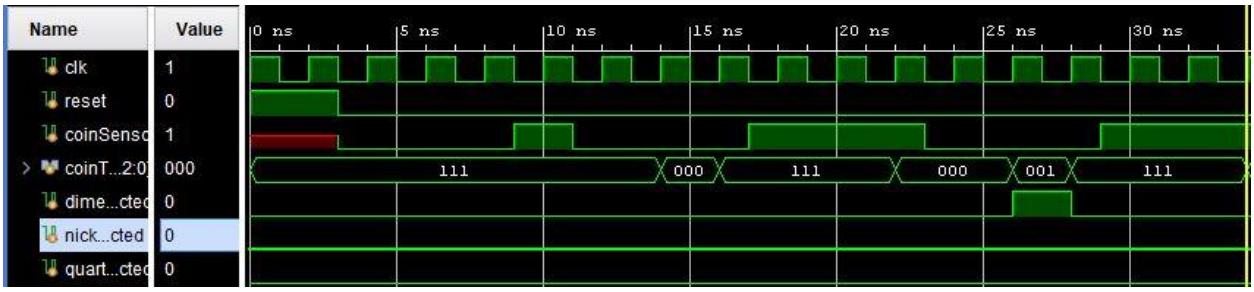

Figure 3: Waveform to Show dimeDetected Output

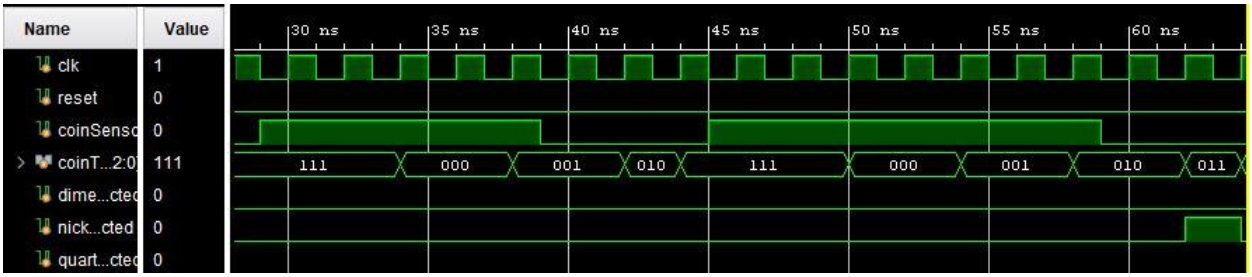

Figure 4: Waveform to Show nickelDetected Output

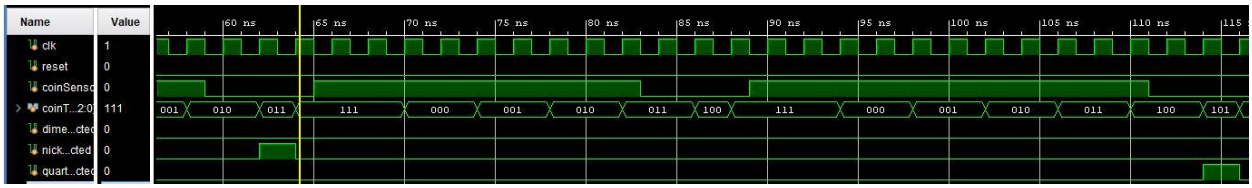

Figure 5: Waveform to Show quarterOutput

In Figure 3, the waveform shows the module being reset, a couple of cycles without an input sensor signal, then one clock cycle of the sensor being high. As mentioned before, the output is 2 cycles behind, so the state does not change until 2 cycles later from idle (111) to testDmin (000). The signal is then turned off, and the state returns to idle. Then the for-loop increments to the next state by keeping the sensor input high for 2*I clock cycles. This process is done for Figure 4 and Figure 5 to test if the other outputs match, which they do.

```
Info: Dime Detected!
Time: 25 ns  Iteration: 0
Info: Nickel Detected!
Time: 57 ns  Iteration: 0
Info: Quarter Detected!
```

Figure 6: TCL Console Output for Output Assertions

Figure 6 shows the output of the console window which shows that the assertions showed that the dime, nickel, and quarter are detected when expected in the testbench simulation. Combining the waveforms from Figures 3-5 and output from Figure 6, it is safe to assume this module works exactly as expected. This completes the assignment.

Conclusion

This assignment was a great way to introduce students to FSM's and how they can be implemented with the skills obtained earlier in the course using Vivado and SystemVerilog. Students can use these skills to solve real-world problems and build upon this simple FSM to