**ECE324 Homework7 (take-home Quiz)**

**SystemVerilog Finite State Machine Logic and Simulation using Tasks**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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 |

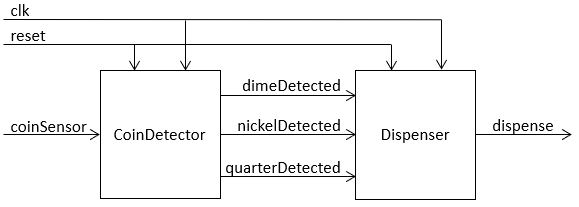
2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.

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

# Problem:

1. Use SystemVerilog to generate a coin-operated vending machine design which dispenses a product under the following conditions:
   1. The machine detects nickels, dimes and quarters.
   2. A product will be dispensed when 25 cents has been accumulated.
   3. If too much money is deposited, the machine will not return coins, but it will credit the buyer toward the next purchase. For example, if 3 dimes are inserted, a product will be dispensed, and 5 cents will be credited toward the next purchase.

The block diagram below is the block diagram of the vending machine you will finish.



CoinDetector may be instantiated without change from your previous homework assignment. Dispenser is a new FSM that you need to design. Its output is “dispense”, which goes high for one clock cycle when a product is to be dispensed. The output “dispense” should not have any glitches on it, so register it (which delays the output by one clock cycle).

1. Finish writing and verifying the correct functionality of a SystemVerilog testbench that includes the following SystemVerilog tasks:
   1. “rst” clears “coinSensor” to 0, and sets “reset” to 1 for one clock cycle, then clears “reset”.
   2. “depositCoin” sets “coinSensor” to ‘1’ for “coinType” clock cycles (where “coinType” is a 21 bit input to the task), then returns “coinSensor” to ‘0’, and waits the appropriate number of clock cycles before verifying, using an assert, the value of “dispense” is what was expected as a result of the deposited coin.
   3. “RDDDDDDD” tests depositing dimes, with all possible amounts of credit, by calling “rst”, and then simulating 7 dime insertions by calling “depositCoin” seven times.
   4. “RQNQNQNQNQNDD” tests depositing quarters and nickels, with all possible amounts of credit, by calling “rst” and then calling “depositCoin” twelve times (using the coin sequence indicated in the task name).
2. Test your testbench by sabotaging your Dispenser FSM: introduce an error that causes one, **and only one**, incorrect state transition, and verify that your testbench catches the erroneous transition and prints a “fail” message. Explain the nature of the sabotage in comments embedded within the sabotaged module SystemVerilog code.

# Deliverables:

Submit to Blackboard the following for grading:

1. All SystemVerilog code that you wrote: FSM, sabotaged FSM, and testbench. All SystemVerilog code must include embedded explanatory comments, and header comments with your name, date, and “ECE 324 Homework 7”
2. Simulation results proving that your module functions correctly: the console message produced by an assert statement in your testbench.
3. A written statement explaining how your simulation results prove that that both your module and testbench function correctly

# Extra Credit (up to 5 points):

Modify your testbench using SystemVerilog’s **$random** system function to generate a random input sequence of nickels, dimes, quarters, and invalid coins. The sequence should be 30 coins long. The testbench’s checker should be able to determine, for any random input sequence, when a product is to be dispensed, and check that the FSM dispenses a product at the expected times.

Submit for grading the modified testbench, simulation results, and a written explanation of the results.