ECE 3544: Digital Design I

Project 4 – Design and Synthesis of a Synchronous Finite State Machine

Honor Code Requirements

You must do this assignment individually. The following represent unauthorized aids, as the term is used to define cheating in the Virginia Tech Honor System Constitution:

* Discussing *any aspect or result* of your design with *any person* other than your instructor and your GTA. This includes but is not limited to the implementation of Verilog code, as well as the supporting details for that code – state diagrams, state tables, block diagrams, *etc.*
* Using *any design element* – including Verilog code – from *any printed or electronic source* other than your course textbook and those sources posted on the course Scholar site.

All code submitted is subject to plagiarism checking by MOSS (theory.stanford.edu/~aiken/moss/) to check for plagiarism. Any copying flagged by MOSS will be treated as Honor Code violations and submitted to the Virginia Tech Honor System.

Project Objective

In this project, you will design and implement a finite state machine to model the controller for a vending machine. The core of the state machine will center on two interacting state machines: one to keep track of the amount of money that a user has deposited, and one to sequentially dispense the purchased product and change. The system will contain a number of counters to track the number of coins that the machine contains; these counters will increment as coins are deposited and dispensed. These counters will also need to interact with the state machines at the core of the system.

System Interface

The system has the following inputs:

* Clock: The board has a 50 MHz clock signal. You must use the board’s clock as the clock for the sequential elements of your design. You may not use the switches or pushbuttons as a clock signal.
* Pushbuttons: Key0 is an active-low Reset. Key1 is used as an Enable key. It allows the system to change state based on the present state and the inputs, in the same manner that it did in Project 3.
* Switches:
  + SW[2] (mode): The system is in Normal Mode when mode = 1, and in Maintenance Mode when   
    mode = 0.
  + SW[1:0] (coin): Each combination of coin represents a different coin event: 00 – no coin; 01 – nickel; 10 – dime; 11 – quarter. The system “processes” the coin event at appropriate points in its operation when operator presses the Enable key. Consider using a decoder to derive individual coin signals from the values of SW[1:0].

The system has the following outputs:

* HEX[5:0] – Each pair of hex displays shows the number of coins that the vending machine contains, as represented by one of three coin counters: quarter\_counter, dime\_counter, and nickel\_counter. Use HEX[5:4] to display the value of quarter\_counter. Use HEX[3:2] to display the value of dime\_counter. Use HEX[1:0] to display the value of nickel\_counter. You need not convert the values to decimal; the displays may show any value from 0x00 to 0xFF. (This is a maintenance mode function on its face, but having the hex displays allows us to look at the counter values at all times.)
* LEDS
  + LED[3] should be on if dime\_counter and nickel\_counter both equal 0. LED[3] should be off otherwise.
  + LED[2] should light for one second if the vending machine is dispensing a product. LED[2] should be off at all other times.
  + LED[1] should light for one second if the vending machine is dispensing a dime as change. LED[1] should be off at all other times.
  + LED[0] should light for one second if the vending machine is dispensing a nickel as change. LED[0] should be off at all other times.

Change is dispensed sequentially, so if the vending machine dispenses two coins of the same type in a row, that LED should light twice for one second each. You may de-assert the LED between coins if you like, but doing so is not required. Whether this is visible will depend on the amount of time that the LED is de-asserted between periods when it is asserted.

You may use the remaining switches and LEDS for any debugging purpose that you wish.

Method of Operation

* The vending machine sells a product that costs 60 cents. The vending machine accepts nickels, dimes, and quarters.
* Resetting the system should cause it to enter into a state where a user has not deposited any money, and there are no coins of any kind in the machine.
* In Normal Mode, asserting the Enable when a particular coin event is present on the inputs represents a user depositing money to purchase a product. This event should have two effects: a) it should advance the state of the system to one that reflects the current amount of money deposited, and b) it should increment the appropriate coin counter.
* In Maintenance Mode, asserting the Enable when a particular coin event is present on the inputs represents a technician putting a coin of that type into the machine. Therefore, this event should increment the appropriate coin counter, but it should not advance the state representing the amount of money that a user has deposited.
* In Normal Mode, once the user has deposited enough money to purchase the product, the system needs to dispense the product and change. *Divide the responsibility for keeping track of the money deposited and running through the sequence of dispensing the product and change between separate state machines.* Through a series of states, the system should dispense the product first, followed by a sequence of coins that make change. The system should dispense the largest-valued coin available that correctly makes change for the amount that must still be dispensed, until all change has been given.

For example, if the user deposits a total of 80 cents, then the machine should dispense the product first, followed by a dime if one is available, and a nickel if no dime is available. The dime would be followed by another dime if one is available, and a nickel if the second dime is not available, and so forth. The coin counters must therefore a) provide information to the “dispense” state machine about whether they have coins to give, and b) decrement correctly as the “dispense” state machine dispenses change.

The user will not attempt to deposit coins in the machine while it is dispensing the product and change.

Design Tips

* Design smaller aspects of this system individually. Implement and test one before moving on the next one. *Don’t try to implement the whole system before testing any of it*.
* Decompose the design into communicating finite state machines, e.g., a state machine to keep track of the amount of money that the user has deposited, and another to sequence the dispensing of the product and change.
* Make a block diagram of the system before you write any of the module, as this will help you settle the manner in which the various state machines should interact.
* Use the keypressed module from the previous project to generate a one-clock-cycle enable signal when a pushbutton is pressed and released.

Report

Your report should include waveforms showing correct behavior of your design. Write your testbench to simulate a user depositing different sequences of coins, or a maintenance technician adding coins to the machine. *The validation test sequence is not nearly exhaustive. While you need not produce an exhaustive set of test cases in your testbench, you should not rely on the validation sequence as the sole test of your system’s correctness.*

Discuss the decisions you made about how to implement each aspect of the design. Consider including a hierarchical depiction (such as a block diagram) of the interacting units in your system. Include state diagrams for your design, as well as any information necessary to operate your design that is not included in this project specification. Document any additions you make for debugging purposes, and describe your motivation for each debug operation.

Submission

Submit your report as an electronic document on Scholar. Put your report and source files into a single .zip file. Submit your validation sheet as a hardcopy to your instructor. You must submit all elements of the project by the deadline to avoid late penalties; the assignment is only as timely as the last element to be submitted.

The queue at the CEL can be quite long at this time of the semester, so you should plan on validating early. Being unable to validate on time because the CEL closes before you come to the head of the queue will not be considered a reasonable excuse for not meeting the deadline.

Your project submission on Scholar should include the following items:

1. Project report in PDF
2. Source files for your top-level module and any modules that it requires to function, and for the top-level module’s testbench.

Your top-level module may be tested with our own secret testbench, so it is important that you use the module declaration provided with the archived project. You must also include the source files for every module required for your top-level module. Failure to do so will result in a grade of 0 for that portion of the project.

Grading for your submission will be as described on the cover sheet included with this description.