ECE199JL: Introduction to Computer Engineering **Fall 2012**
Homework 9 **Due: at start of lecture on Friday 26 October**

State Machines and Logic Components

There are no problems from the textbook this week!

Here are six problems for this week. The first is drawn from an old ECE190 exam (in case you are wondering about the difficulty of FSM test questions):

1. Analyzing and Designing a Drink Dispenser

After his last attempt to produce a layered drink dispenser for EOH, Professor Lumetta has decided to try something simpler. He has just completed the core of a simple soda machine, but needs your help wiring up the controls.

The goal for the dispenser is to accept **nickels** (5 cents), **dimes** (10 cents), and **quarters** (25 cents) towards the purchase of a can of soda. When the machine has received 40 cents or more, it dispenses a can of soda and returns any money over 40 cents. For example, if someone puts in two quarters, one after another, the machine returns a can of soda and 10 cents. The core of the machine is shown below.

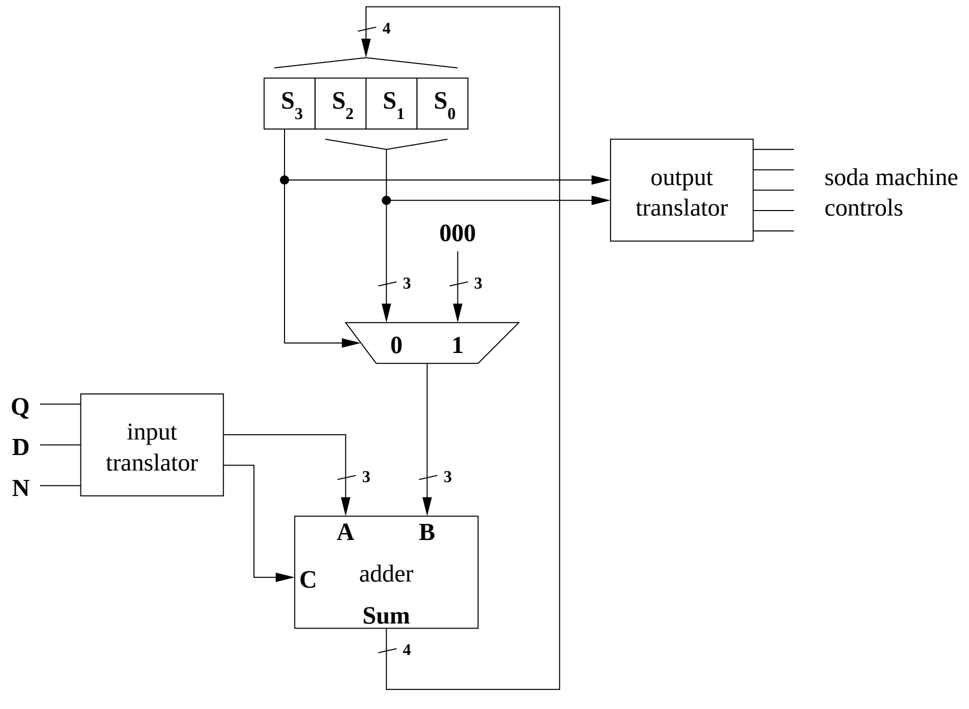
The four-bit register S records the amount of money currently held by the machine as an unsigned number of nickels. For example, if the machine has received 25 cents, $S = 0101$.

The input translator changes the signals for the coin types that have been inserted in a given cycle into a number of nickels to be added into the register S.

The adder is a standard three-bit adder except that the carry output is included as the most significant bit of the sum rather than being separated out. The C input is the carry input for the adder.

The output controller translates the current state S into control signals for dispensing a can of soda and for returning various types of coins.

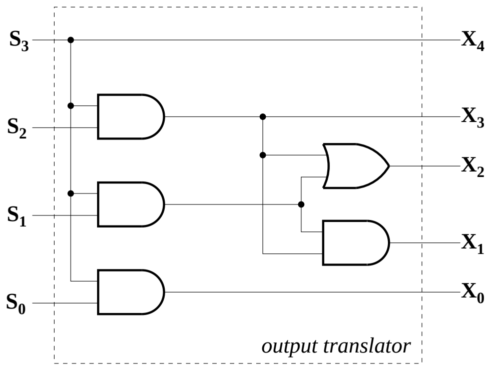
The trapezoid in the middle of the figure represents three 2-to-1 multiplexers (with the select inputs of all three muxes connected to S_3).



A. Explain the purpose of the multiplexer. *Hint: consider what happens if you repeatedly insert a quarter in every cycle.*

B. Prof. Lumetta has run out of components! Design the input controller for him using only wires. The Q input indicates that a quarter was inserted. The D input indicates that a dime was inserted. The N input indicates that a nickel was inserted. **Any combination of coins is possible in a cycle.** Your wires (no gates!) must produce the 3-bit A input as well as the carry-in input C to the adder.

C. One of Prof. Lumetta's graduate students was kind enough to design the output controller for him (shown in the figure to the right). Unfortunately, the student did not produce documentation. The inputs from the FSM state S are labeled, but Prof. Lumetta is not sure what the X outputs are supposed to do. One of them, of course, must release the can of soda. The other four dispense coins, but what kind of coins (nickel, dime, or quarter)?



For each of the fives outputs $X[4:0]$, choose one of the following for the machine to dispense: SODA, NICKEL, DIME, QUARTER. *Hint: you may want to draw a truth table to help you.*

2. Multiplexers (Muxes)

Using ONLY 2-to-1 multiplexers as building blocks (no other gates), design an 8-to-1 multiplexer.

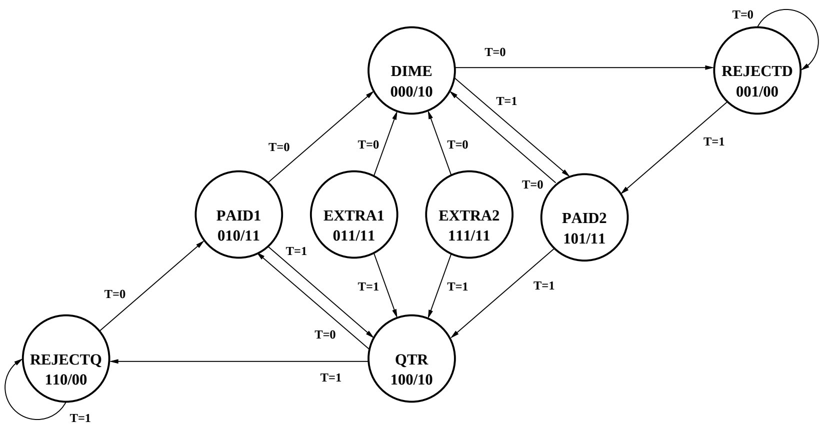
3. A Sequence Recognizer

Design and implement an FSM with one input B and one output R that recognizes the sequence 1x01 on its input. In particular, the output R should be 1 in exactly those cycles in which input B has matched the pattern 1x01 in the previous four cycles. The 'x' in the pattern matches any input value. Develop an abstract FSM design that solves the problem (include specific input bit values for each transition as well as the output bit in each state). Choose a representation for your states and add it to your state transition diagram—you need only turn in one copy, but it should be labeled with state names as well as state bit / output combinations and input bits on transitions. Calculate Boolean logic expressions for the next-state values as well as the output R, then implement your design with flip-flops and gates. You may assume that your FSM starts in a particular state, but you must tell us which state! (NOT REQUIRED: For slightly more challenge, leave one pattern of internal state bits unused without requiring an extra flip-flop.)

Hints: If you think that the 'x' makes this problem easier, your answer is wrong. You may want to write down the known observed input sequence for each state that you create.

4. Preparation for Labs 1 and 2

Over the next three weeks, you will be building combinational logic and a small FSM to control a coin mechanism similar to those used on vending machines. A state diagram for the FSM that you will use appears to the right. The FSM tracks entry of dimes and quarters as well as when thirty-five cents in total have been entered, at which point a light is lit to indicate that the user should receive a candy (for example).



The states are labeled with internal state bits $S_2S_1S_0$ and output bits A and P (written as $S_2S_1S_0/AP$). The output A determines whether a servo accepts or rejects the coin just deposited, and the output P indicates that the machine has received thirty-five cents. Note that the machine starts over at this point, allowing new coins to be entered. The transitions are labeled with values of T, which indicates which type of coin has been inserted (either a dime or a quarter).

A. Start by transcribing the state diagram into a state table with next state and outputs for each input combination, as shown (in part) below.

		next state $S_2^+S_1^+S_0^+$		outputs	
state	$S_2S_1S_0$	$T=0$	$T=1$	A	P
DIME	000	001	101	1	0
REJECTD	001				
and so forth...					

B. For each next state bit and each output, use a K-map to calculate an expression with a minimal number of terms. In particular, you must find S_2^+, S_1^+, S_0^+, A, and P.

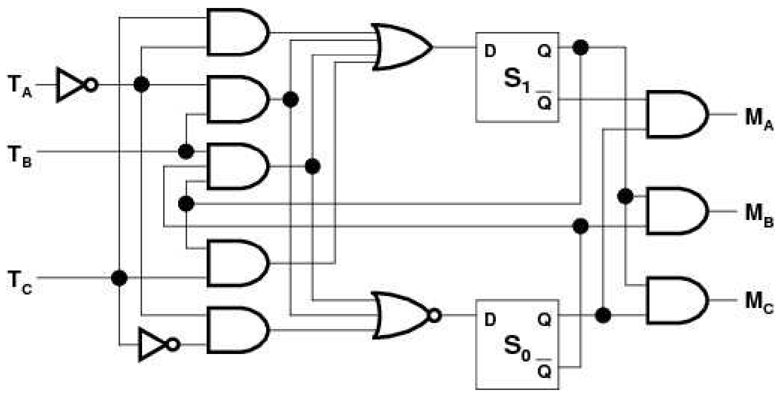
5. Subway Safety

You must check the operation of a safety control unit for a subway before it can be installed. The subway train station serves three separate train lines: A, B, and C. Only one train from one of these three lines can be allowed to enter the station at any time; if more than one is ever allowed to enter, an accident may occur.

The diagram to the right shows the control unit implementation.

The controller has three inputs: T_A, T_B, and T_C. Each of these inputs corresponds to the presence of a train on one of the lines; a 1 indicates that a train is ready to enter or is passing through the station from the given line.

The controller has three outputs: M_A, M_B, and M_C. Each output corresponds to one of the trains, and a 1 grants permission for that train to enter the subway station.



In order to avoid accidents, the controller must only allow one train at a time to enter the station, and must guarantee that any train in the station has left before allowing another train to enter.

A. Explain the meaning of the four possible states S_1S_0. (Look at the outputs.)

B. Draw a state transition diagram for the controller.

C. The controller has an error. Find it and explain it.

D. Fix the error.

6. Software FSMs, Part II

Beginning with the program `dungeon.c` from Homework 8, add a new room to the game. Your room should have at least two exits. Also, it must be possible to (1) reach your new room through normal game play, (2) win from your new room (not necessarily in one transition), and (3) lose from your new room (again, not necessarily in one transition). Draw the expanded state diagram for your new version of the game, then turn in both the diagram and a copy of your code.