ECE199JL: Introduction to Computer Engineering **Fall 2012**
Homework 8 **Due: at start of lecture on Friday 19 October**

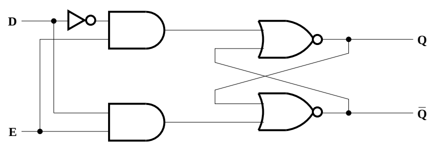
Finite State Machines

Please do problem 3.43 from the textbook.

Here are five additional problems for this week:

1. Understanding Sequential Logic

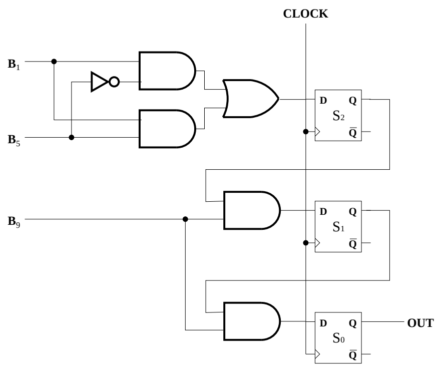
Write a full truth table for the circuit shown to the right. For each input combination, your table should include a row for every stable state.



Can this circuit store a bit? Explain your answer in terms of the truth table.

2. Analyzing an FSM

The state machine shown to the right is driven by three buttons labeled "1," "5," and "9." The inputs shown in the diagram correspond to the buttons being pressed, so, for example, $B_1 = 1$ when the "1" button is pressed, and $B_1 = 0$ when the "1" button is not pressed.

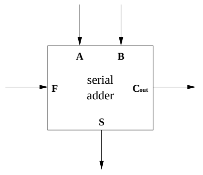


A. Assuming that only one button is pressed in each cycle, what sequence of buttons must be pressed to make this FSM output a 1 (that is, to make $OUT = 1$)?

B. Explain how you can simplify the design (use fewer gates to implement the same FSM).

3. Serial Addition

Design a serial adder to match the block diagram shown to the right. In particular, your adder should accept one bit per cycle on the A and B inputs, starting with the least significant bit and working upward in significance. The F input indicates the start of a new addition, and should be set to 1 when the least significant bit is entered (and reset to 0 for all other bits).



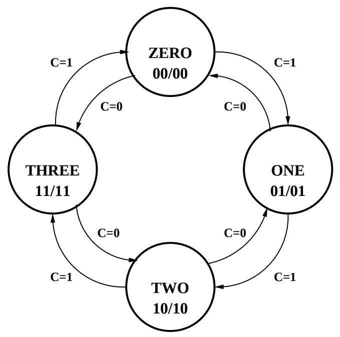
Your adder should produce the sum bit S (each cycle) and carry out bit C_{out} (when the addition is done) *in the next cycle—do NOT route a full adder's output directly to the output of your design!*

Hint: This design requires two flip-flops, a full adder, and one extra gate. If you use an extra gate or two, that's acceptable, but use these components as building blocks to get full credit.

Turn in a clearly drawn implementation diagram (similar to those shown for other problems in this homework).

4. A Bidirectional Binary Counter

Implement the state machine shown to the right, which is a 2-bit binary counter with a control input C that specifies whether the counter counts upward ($C = 1$) or downward ($C = 0$). States are named and labeled with the internal state bits (call them S_1 and S_0) and the outputs (call them Q_1 and Q_0), which in this design are equal to the state bits. In other words, $Q_1 = S_1$ and $Q_0 = S_0$ in all cases.



Turn in a clearly drawn implementation diagram (similar to those shown for other problems in this homework).

5. Software FSMs

Finite state machines also play important roles in software design, including digital control system implementation and event-driven software design (most web services, user interfaces, and a growing number of games are designed in this way) as well as parts of compilers.

In this problem, you must draw a state transition diagram corresponding to an adventure game. In the game, each "room" is a state, and the input values (0, 1, or 2 for our game) correspond to transitions.

Download the program `dungeon.c` from our class' web page and draw a state transition diagram. Use the room number from the code to label the states, and the input value (0, 1, or 2) to label transitions between states. Note that the game ends in some of the rooms, so your diagram should not have transition arcs leaving these states.