Homework 9

Homework 9 is due on Monday, April 20, at the start of the lecture. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

All problems in this homework are design questions, so this homework is HARD. Therefore:

- Start early
- Come to office hours
- Elaborate your design clearly: follow the procedures we taught you in class.
- Tell us what each of your states mean in your state diagram
- Draw your circuit clearly (label DFFs, MUXs, etc)

This is not only a hard homework to do, but also a hard homework to grade. Be sure to label all states, transitions, inputs and outputs so we know what you mean. **If we cannot understand your design because it is not clearly labeled, you may not get points at all.**

FSM Design

1. Sequence Recognizer 1

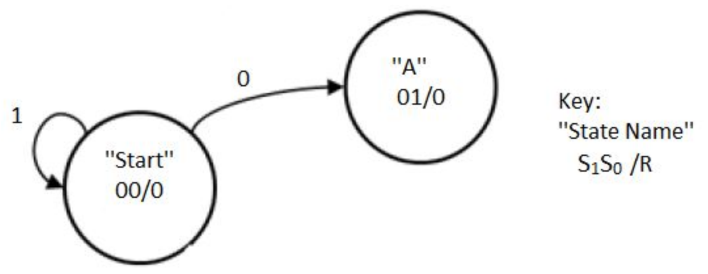
Design a finite state Moore machine that recognizes a particular pattern: "010". The input to this FSM is a sequence of bits in series coming in at input M, and the output is a sequence of bits appearing at output R. When the FSM sees "010" as input, it outputs a "1"; otherwise, it should output a "0". In particular, the output R should be "1" in exactly those cycles in which input M has matched the corresponding sequence in the previous 3 cycles. In particular, the FSM detects overlapping sequences of "010". For example:

```
Input Sequence M (starting from left to right)  - 000101001011
Output Sequence R (starting from left to right) -  000010100100
```

Note that the output sequence is delayed by 1 clock cycle compared to the input sequence because the output R is a function of the flip-flop outputs (i.e. the state variables) in a Moore machine. That's why the output sequence becomes 1 in the cycle after "010" has been completed by the input.

1. 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. Shown below is an (*incomplete*) starting point for your FSM. You may assume that your FSM starts in a particular state, but you must tell us which state. Choose a representation for your states and add it to your state transition diagram. Your states should be labeled with state names as well as state bit /output combinations and input bits on transitions. Develop an FSM with the **minimum** number of states that are necessary.

Incomplete FSM for detecting pattern "010":



Hint : For your convenience, some of the states and their meanings are shown in the table below, fill the meaning of the remaining states and use it to construct your FSM

State Names	States(S_1S_0)	State Meanings
Start	00	None of pattern elements have been found yet
"A"	01	First '0' in the pattern '010' has been found
"B"	--	--
"C"	--	--

2. Fill in K-maps for the next state values S_1^+, and S_0^+ based on your FSM.

3. Calculate minimal **SOP** Boolean logic expressions for the next state values as well as the output R.

4. Implement your design with D flip-flops and gates.

2. Sequence Recognizer 2

Modify the finite state machine from the previous problem such that it only detects non-overlapping patterns of "010". For example, for the following input sequence we would have the following output:

```
Input Sequence M (starting from left to right)  - 000101001011
Output Sequence R (starting from left to right) -  000010000100
```

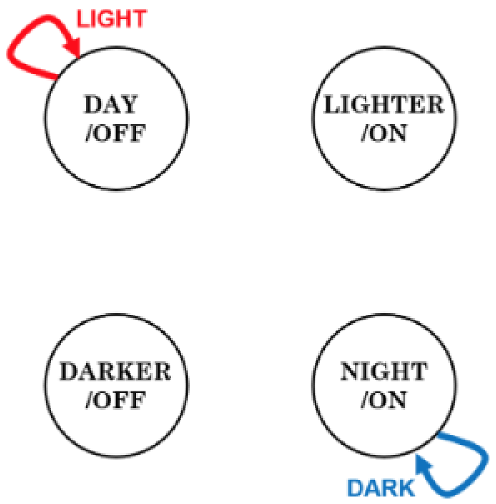
Hint: Notice the difference in the FSM output compared to Problem 2 for the same input sequence.

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). Your states should be labeled with state names as well as state bit/output combinations and input bits on transitions. Do not implement it though, just the abstract model is sufficient for this assignment.

3. Headlight Control

Your aid is needed in developing the next-generation automatic headlight control system for Fjord Motors. You must design an FSM that monitors the current ambient light level, provided to your FSM as a 2-bit unsigned number L, and automatically turns the headlights on/off through an output H (H=1 means the headlights are on). Your design should turn the headlights on whenever the light level L has been 1 or less for two consecutive cycles, and should turn the headlights off whenever the light level L has been 2 or more for two consecutive cycles (remember than L is a 2-bit unsigned number, hence ranges from 0 to 3).

1. Complete the abstract state transition diagram below by adding transitions labeled as DARK and LIGHT from each state. These labels correspond to L < 2 and L 2, respectively.



2. Now choose a representation for the states. The "DAY" state should use S_1S_0 = 00. If possible, choose bit patterns such that all transitions change only one state bit rather than changing both state bits.

3. Write a truth table for the output H in terms of the current state S_1S_0, and a next-state table for the next state bits S_1^+ and S_0^+ in terms of the ambient light level L= L_1L_0 and the current state S_1S_0.

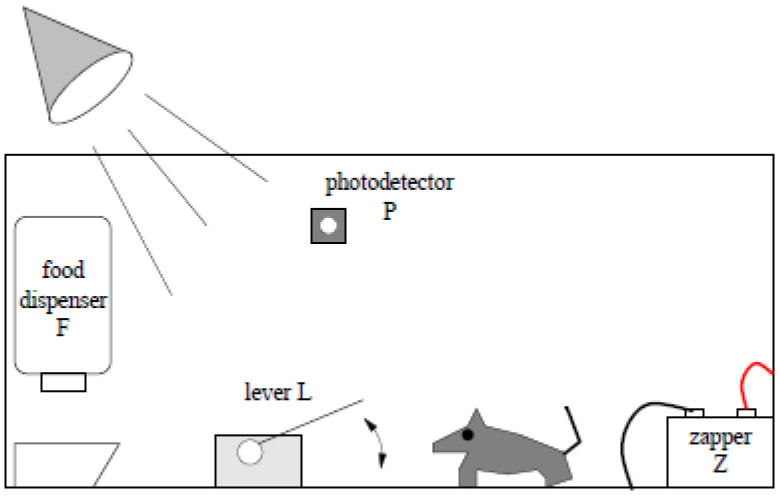
4. Copy the truth table and next-state table from **part 3** into separate K-maps (one for H and one for each next-state variable).

5. Use the K-maps from **part 4** to find expressions for each output and next-state variable with minimal area. (**Minimize each variable independently—do not try to share gates for implementations.**) You must consider both minimal SOP and minimal POS solutions, but you should only circle the better of the two choices (SOP or POS) on your K-maps and write the corresponding expressions when handing in your homework.

6. For the luxury model of Fjord's new vehicle, the driver has a control knob to adjust the automatic headlights' light-level threshold. This knob produces an 8-bit unsigned number T. The light sensor L has also been upgraded to produce an 8-bit unsigned value (instead of a 2-bit value). The values L and T are fed into an 8-bit unsigned comparator that produces a signal X whenever L < T. In other words, X=1 when L < T, and X=0 when L T. Explain how to integrate your FSM design with the comparator for the luxury version of the vehicle. *Hint: You should not need to change your design's structure.*

4. Prof. Zapper's Rats, part II

Pleased with your previous work and the outcome of the experiment, your part-time employer Professor Zapper from Psychology gives you a modest raise and asks you to design the logic for a slightly more complicated experiment to study the memory of rats. The experimental set-up is shown below. In each "cycle," the experimenter turns the light on or off. Your system receives a signal from the photodetector P: when the light is on, P = 1. In the same "cycle" (long cycles), the rat responds by either depressing the lever L (in which case L = 1) or not depressing the lever (L = 0).



Professor Zapper wants the rats to follow a protocol. In the first and second cycles, the rat should match the lever with the light. In other words, the rat should press the lever when the light is on, and not press the lever when the light is off. In the third cycle, the rat should do the opposite: press the lever when the light is off, and don't press the lever when the light is on.

If the rat succeeds in this endeavor, it should receive food (set F = 1 for a cycle). If it fails, it should immediately be zapped for a cycle (set Z = 1 for a cycle). After the penalty/reward cycle, start over.

1. Based on this description, design a finite state machine to implement the desired experimental protocol. In particular, draw a complete state diagram labeled with outputs, internal state bits (you need to choose a representation), and input bit combinations on each transition arc. Hint: you may want to simplify the inputs before your state machine sees them. If you do, explain how.
2. Find simple logical expressions for next-state variables as well as functions mapping your internal state values to the outputs F and Z.
3. Draw a logic schematic circuit diagram for your system.

5. Software FSM

Beginning with the program dungeon.c from Homework 8, add a new room to the game. Your room can be entered only through room 0. Also, it must be possible to do the following:

1. Enter room 0 through your new room (transition 0)
2. Lose the game in one transition (transition 1)
3. Enter room 1 through your new room (transition 2)
4. Enter room 2 through your new room (transition 3)

You cannot win from your room in one transition, however you might enter other rooms through your room to win the game. Draw the expanded state diagram for your new version of the game, then turn in both the diagram and a copy of your code.