FSM design and implementation

1. Mystery FSM

Consider the finite state Moore machine shown below. The input to this finite state machine (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.
1. Assuming that the FSM starts in the “Start” state, write the sequence of bits appearing as output R if the sequence of input bits is 1000100100100011 (starting from left to right).
2. Describe in words what this FSM does. Specifically, what does the output R indicate about the input sequence?

2. 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:

<table>
<thead>
<tr>
<th>Input Sequence M (starting from left to right)</th>
<th>Output Sequence R (starting from left to right)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000101001011</td>
<td>00010100100</td>
</tr>
</tbody>
</table>

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":

![State Transition Diagram](image-url)
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

<table>
<thead>
<tr>
<th>State Names</th>
<th>States(S_1S_0)</th>
<th>State Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>00</td>
<td>None of pattern elements have been found yet</td>
</tr>
<tr>
<td>&quot;A&quot;</td>
<td>01</td>
<td>First '0' in the pattern '010' has been found</td>
</tr>
<tr>
<td>&quot;B&quot;</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

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.

### 3. 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.

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). Your states should be labeled with state names as well as state bit/output combinations and input bits on transitions.
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.

### 4. Software FSMs

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:

i. Enter room 0 through your new room (transition 0)
ii. Lose the game in one transition (transition 1)
iii. Enter room 1 through your new room (transition 2)
iv. 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.