Finite State Machine Design

Design Process
1) Figure out necessary states from problem specification (e.g., a counter from 0 to 10 needs 11 states)
2) Draw state graph
   - Enumerate states
   - Capture transitions between states and inputs
3) Add outputs
   - Moore machine: outputs function of present state only
   - Mealy machine: outputs are function of present state and inputs
4) Figure out number of flip-flops required for implementation (minimum # is \( \log_2 \) (# of states)
5) Assign states (assignment may affect the number of flip-flops)
6) Construct a state table
   - Develop equations for next states
7) Figure out outputs
   - Develop equations for outputs (different for Moore vs. Mealy)

Ex: Count sequence 0, 9, 15, 2, 7 <repeat>
   If input \( X = 1 \), output 2 and start count from there when \( X = 0 \).
   Reset input go to 9 in the sequence

5 values in sequence \( \Rightarrow \) 5 states \( (S_0 - S_4) \)

![State Diagram]

Moore

- no label since we go to \( S_3 \) independent of any input
State Assignments

1) Assign in order of normal operating sequence (easy to think about)
2) Assign at random
3) Assign to minimize logic/circuit implementation
4) One-hot encoding

- Use one flip-flop for each state, so an N-state FSM requires N flip-flops (possibly much larger than \( \log_2 N \))
- Reduces complexity of logic for computing next states and outputs
- Use when flip-flops are relatively cheap (e.g., FPGAs, CPLDs, etc.)
or 5) Gray code (reduces power by reducing number of bit transitions)

**Ex:** Continue previous example; sequential state assignment

<table>
<thead>
<tr>
<th>Present State $Q_2Q_1Q_0$</th>
<th>Next State $X=0$ $X=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q_2^+Q_1^+Q_0^+$</td>
</tr>
<tr>
<td>$S_0$</td>
<td>0 0 0</td>
</tr>
<tr>
<td>$S_1$</td>
<td>0 0 1</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0 1 0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0 1 1</td>
</tr>
<tr>
<td>$S_4$</td>
<td>1 0 0</td>
</tr>
</tbody>
</table>

Equations:

$$Q_0^+ = X + X'Q_2^+Q_0^+$$
$$Q_2^+ = X'Q_1Q_0$$
$$Q_1^+ = X + X'Q_2^+Q_1Q_0^+$$

Now try one-hot state encoding → need 5 flip-flops

<table>
<thead>
<tr>
<th>Present State $Q_4Q_3Q_2Q_1Q_0$</th>
<th>Next State $X=0$ $X=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q_4^+Q_3^+Q_2^+Q_1Q_0$</td>
</tr>
<tr>
<td>$S_0$</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>$S_1$</td>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0 0 1 0 0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>$S_4$</td>
<td>1 0 0 0 0</td>
</tr>
</tbody>
</table>

Equations:

$$Q_0^+ = Q_4^+X'$$
$$Q_1^+ = Q_0^+X'$$
$$Q_2^+ = Q_1^+X'$$
$$Q_3^+ = Q_2^+X' + X$$
$$Q_4^+ = Q_3^+X'$$
Speed of Moore vs. Speed of Mealy

Consider a microprocessor data pipeline -> the controller can be an FSM

64 gate delays!

Maybe 20

add/sub

mult/shift

Controller

Maybe 2-3 gate delays

Older computers ~50 gate delays
Fastest current computers ~15
Special purpose processors ~10

- Easier to design a long path with a Mealy machine
- Purely combinational path from input to output yields possibly faster decisions, but potential for hazards