State Machine Design Procedure

1. Build state/output table (or state diagram) from word description using state names.

2. Minimize number of states (optional).

3. State Assignment: Choose state variables and assign bit combinations to named states.

4. Build transition/output table from state/output table (or state diagram) by substituting state variable combinations instead of state names.

5. Choose flip-flop type (D, J-K, etc.)


7. Derive excitation equations from excitation table.

8. Derive output equations from transition/output table.

9. Draw logic diagram with excitation logic, output logic, and state memory elements.
State Machine Design Example 1: 110 Detector

• Word description (110 input sequence detector):
  – Design a state machine with input A and output Y.
  – Y should be 1 whenever the sequence 1 1 0 has been detected on A on the last 3 consecutive rising clock edges (or ticks).
  – Otherwise, Y = 0
  – Note: this is a Moore machine, that is the output, Y, depends only on inputs at previous clocks rising edges, not on the current input.

• Timing diagram interpretation of word description (only rising clock edges are shown):

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State Machine Design Example 1: 110 Detector

Step 1: Choosing States

- Possible states (What does the state machine need to remember?):
  - Initial: power up, no clocks yet \( Y = 0 \)
  - No1s: first 1 not found \( Y = 0 \)
  - First1: first 1 found \( Y = 0 \)
  - Two1s: at least 2 consecutive 1s found \( Y = 0 \)
  - ALL: found 1 1 0 \( Y = 1 \)

- Are all the states needed?
  - Notice: Initial is equivalent to No1s
  - We can drop the state Initial and replace it with state No1s
State Machine Design Example 1: 110 Detector

Step 1: State/Output Table and Diagram

### State Table

<table>
<thead>
<tr>
<th>S</th>
<th>0</th>
<th>1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>No1s</td>
<td>No1s</td>
<td>First1</td>
<td>0</td>
</tr>
<tr>
<td>First1</td>
<td>No1s</td>
<td>Two1s</td>
<td>0</td>
</tr>
<tr>
<td>Two1s</td>
<td>ALL</td>
<td>Two1s</td>
<td>0</td>
</tr>
<tr>
<td>ALL</td>
<td>No1s</td>
<td>First1</td>
<td>1</td>
</tr>
</tbody>
</table>

### State Diagram

- **Reset**
- **Format:**
  - Arc: input A
  - Node: state/output Y
Step3: State Assignment Considerations

• Why does the choice of state assignment matter?
  – Has a big effect on the complexity of excitation and output equations
    and thus on the amount of combinational logic needed.

• How to find the best state assignment?
  – The only known way is to try all assignments and determine the
    resulting equations.
    • N = 2: \((2^2)! = 4! = 24\) assignments for 2 state bits
    • N = 3: \((2^3)! = 8! = 40,320\) assignments for three state bits.
    • N = 4: \((2^4)! = 16! = 20,922,789,888,000\)
      assignments for 4 state bits!!!

    **THIS IS NOT PRACTICAL APPROACH!**

  \[\therefore\] Use heuristic guidelines for pretty good assignments.

  This is still an active area of research!

• There is no effective way to guarantee a “best” assignment. The
  heuristic methods sometimes perform poorly!
State Assignment Strategies

- **Simplest Assignment:**
  - Straight binary, not best; purely arbitrary assignment.

- **One Hot Assignment:**
  - Redundant encoding, each flip-flop is assigned a state.
  - Uses the same number of bits as there are states (not useful in large designs).
  - Simple to assign; simple next state logic (no state decoding required)
  - Output logic is simple! One OR gate per Moore output!

- **Almost One Hot Assignment:**
  - Almost same as One Hot, but one less state bit.
  - Use all 0’s to represent a state (usually INIT).
  - Must now decode state 0 if it is needed.

- **Decomposed Assignment:**
  - Use the “structure” of the state table to simplify next-state and output logic.
  - An “art” which requires much practice.
Example: State Assignment Strategies

Alternative Assignments

<table>
<thead>
<tr>
<th>Q₀ Q₁ Q₂ Q₃</th>
<th>S</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>INIT</td>
<td>A0</td>
<td>A0</td>
<td>A1</td>
<td>A1</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>A₀</td>
<td>OK0</td>
<td>OK0</td>
<td>A1</td>
<td>A1</td>
<td>0</td>
</tr>
<tr>
<td>0010</td>
<td>A₁</td>
<td>A₀</td>
<td>A₀</td>
<td>OK1</td>
<td>OK1</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>OK0</td>
<td>OK0</td>
<td>OK0</td>
<td>OK1</td>
<td>A₁</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td>OK1</td>
<td>A₀</td>
<td>OK0</td>
<td>OK1</td>
<td>OK1</td>
<td>1</td>
</tr>
</tbody>
</table>

Almost One            One            Decomposed           Simplest
One                  Hot                  Hot

– Example decomposition:

- Initial State = all 0’s for easy RESET
- INIT state is different, so use Q₁ = 1 for non-INIT states; thus D₁=1
- Z = 1 in only 2 states, so use Q₂ =1 for states when Z = 1; thus Z = Q₂
- Use Q₃ = 1 for state transitions caused by A having the value of 1 (all destination states cause by A = 1, i.e. states A₁ and OK1); thus D₃=A

THUS, simpler next state and output logic!
State Assignment Heuristic Guidelines

Starting from the highest priority to the lowest:
• Choose initial coded state that’s easy to produce at reset: (all 0’s or 1’s)
  – This simplifies the initialization circuitry.
• Freely use any of the $2^n$ state codes for best assignment
  (i.e., with s states, don’t just use the first s integers 0,1,…,s-1)
• Define specific bits or fields that have meaning with respect to input or output variables (decomposed codes).
• Consider using more than minimum number of state variables to allow for decomposed codes.
• Minimize number of state variables that change at each transition
• Simplify output logic.
State Machine Design Example 1: 110 Detector

Step 3: State Assignment

- Choose state variable assignments:
  - Initial state all 0s
  - Q2 = last A, so Q2* = A
  - minimize number of transitions

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>S</th>
<th>0</th>
<th>1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No1s</td>
<td>No1s</td>
<td>First1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>First1</td>
<td>No1s</td>
<td>Two1s</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Two1s</td>
<td>ALL</td>
<td>Two1s</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>ALL</td>
<td>No1s</td>
<td>First1</td>
<td>1</td>
</tr>
</tbody>
</table>

A

S*

EECC341 - Shaaban
State Machine Design Example 1: 110 Detector

Step 4: Transition/Output Table

- Step 4: Build transition/output table from state/output table by substituting state variable combinations instead of state names.

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>0</th>
<th>1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>00</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
<td>01</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Q1^{*} Q2^{*} = D1 \quad D2 \quad \text{Step 6} \]

- Step 5: Choose D Flip-Flops, so \( Q^{*} = D \)
- Step 6: Excitation table:
  - Same as Transition/output table with \( Q1^{*} = D1, \; Q2^{*} = D2 \)
State Machine Design Example 1: 110 Detector
Steps 7, 8: Excitation/Output Equations

- **Step 7:** Excitation equations: \( D_1, D_2 = F (A, Q_1, Q_2) \)
  
  \[
  D_1 = Q_1 \cdot Q_2 + Q_2 \cdot A
  \]

- **Step 8:** Output equation: \( Y = G (Q_1, Q_2) \)
  
  \[
  Y = Q_1 \cdot Q_2' \quad \text{(directly read from transition table)}
  \]
State Machine Design Example 1: 110 Detector
Step 9: Logic Diagram

P = Preset
C = Clear
Both active low

RESET_L  reset to initial state (active low)
State Machine Design Example 2: 110/101 Detector

- **Word description (110/101 input sequence detector):**
  - Design a state machine with input A and output Y.
  - \( Y = 1 \) when either sequence 1 1 0 or 1 0 1 has been detected on input A on the last 3 consecutive rising clock edges (or ticks).
  - Otherwise \( Y = 0 \)
  - Note: Correct sequences may overlap and still be accepted.

- **Timing diagram interpretation of word description (only rising clock edges are shown):**

```plaintext
A  0 1 0 1 0 1 1 0 1 0 0 0
CLK
y
```

EECC341 - Shaaban
State Machine Design Example 2: 110/101 Detector

Step 1: Choosing States

- Possible states (What does the state machine need to remember?):
  - Idle: Initial state, no starting 1 yet \( Y = 0 \)
  - Got1: \( A = 1 \) on last tick \( Y = 0 \)
  - Got10: Sequence \( A = 10 \) on last two ticks \( Y = 0 \)
  - Got101: Sequence \( A = 101 \) on last three ticks \( Y = 1 \)
  - Got11: Sequence \( A = 11 \) on last two ticks \( Y = 0 \)
  - Got110: Sequence \( A = 110 \) on last three ticks \( Y = 1 \)
State Machine Design Example 2: 110/101 Detector

Step 1: State/Output Table

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Got1</td>
<td>Got10</td>
<td>Got11</td>
</tr>
<tr>
<td>Got10</td>
<td>IDLE</td>
<td>Got101</td>
</tr>
<tr>
<td>Got101</td>
<td>Got10</td>
<td>Got11</td>
</tr>
<tr>
<td>Got11</td>
<td>Got110</td>
<td>Got11</td>
</tr>
<tr>
<td>Got110</td>
<td>IDLE</td>
<td>Got101</td>
</tr>
</tbody>
</table>

S*
State Machine Design Example 2: 110/101 Detector

Step 1: State Diagram

Format:
Arc: input A
Node: state/output Y
State Machine Design Example 2: 110/101 Detector
Steps 3: State Assignment

- Step 3: Choose state variable assignments:
  - Initial state all 0s
  - Q1 = Y
  - Q3 = last A, so Q3* = A
  - minimum number of transitions

<table>
<thead>
<tr>
<th>Q1 Q2 Q3</th>
<th>S</th>
<th>0</th>
<th>1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>IDLE</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Got1</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Got10</td>
<td>IDLE</td>
<td>Got11</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Got101</td>
<td>Got10</td>
<td>Got11</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Got11</td>
<td>Got110</td>
<td>Got11</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Got110</td>
<td>IDLE</td>
<td>Got101</td>
<td>1</td>
</tr>
</tbody>
</table>

From Step 1:
State Machine Design Example 2: 110/101 Detector

- **Step 4: Transition/output table**
- **Step 5: Choose D Flip-flops**
- **Step 6: Excitation table**
  - Same as Transition table

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000</td>
<td>001</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>000</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>000</td>
<td>111</td>
</tr>
</tbody>
</table>

Unused states?

Q1*Q2*Q3*

= D1 D2 D3
State Machine Design Example 2: 110/101 Detector

Steps 7: Excitation Equations

- **Step 7: Excitation equations**
  - \( D_1, D_2, D_3 = F(A, Q_1, Q_2, Q_3) \)
  - \( D_1 = Q_1' \cdot Q_2 \cdot Q_3 \cdot A' + Q_2 \cdot Q_3' \cdot A \)
  - \( D_2 = Q_2 \cdot A + Q_3 \)
  - \( D_3 = A \) (as planned!)

D1:

\[
\begin{array}{c|c|c|c|c}
Q_1 & Q_2 & Q_3 & A & D_1 \\
00 & 01 & 11 & 10 & d \\
01 & \text{1} & \text{1} & d \\
11 & d & d & d & d \\
10 & d & d & d & d \\
\end{array}
\]

D2:

\[
\begin{array}{c|c|c|c|c}
Q_1 & Q_2 & Q_3 & A & D_2 \\
00 & 01 & 11 & 10 & d \\
01 & \text{1} & \text{1} & d \\
11 & \text{1} & \text{1} & d \\
10 & \text{1} & \text{1} & d \\
\end{array}
\]

D3:

\[
\begin{array}{c|c|c|c|c}
Q_1 & Q_2 & Q_3 & A & D_3 \\
00 & 01 & 11 & 10 & d \\
01 & \text{1} & \text{1} & \text{1} & d \\
11 & \text{1} & \text{1} & \text{1} & d \\
10 & d & d & d & d \\
\end{array}
\]
State Machine Design Example 2: 110/101 Detector

Step 8: Output Equations

• Step 8: Output equation
  – \( Y = Q_1 \) (as planned!)

• Step 9: Logic diagram
  – (3) D-Flip-flops  + (3) 2-input gates  +  (1) 3-input AND gate +
    (1) 4-input AND gate
  – Draw the diagram.

\[
\begin{align*}
D_1 &= Q_1' \cdot Q_2 \cdot Q_3 \cdot A' + Q_2 \cdot Q_3' \cdot A \\
D_2 &= Q_2 \cdot A + Q_3 \\
D_3 &= A
\end{align*}
\]