Basic Sequential Design Steps

- Derive circuit state diagram from design specs
- Create state table
- Choose flip-flops
- Create circuit excitation table
- Construct K-maps for:
  - flip-flop inputs
  - primary outputs
- Obtain minimized SOP equations
- Draw logic diagram
- Simulate to verify design & debug as needed
- Perform circuit analysis & logic optimization
Sequential Design Example

Design a 3-bit gray code counter with active low synchronous reset (R)

State Diagram

State Table

<table>
<thead>
<tr>
<th>Inputs R</th>
<th>Current state (X Y Z)</th>
<th>Next state (X Y Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XXX</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>000</td>
<td>001</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>011</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>110</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>110</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
<td>101</td>
</tr>
</tbody>
</table>
3-bit Gray Code Counter

• Choose flip-flops:
  ➢ Let X be a JK
  ➢ Let Y be a D
  ➢ Let Z be a SR

• Create circuit excitation table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Current state (X Y Z)</th>
<th>Next state (X Y Z)</th>
<th>QX</th>
<th>QY</th>
<th>QZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>XXXX</td>
<td>0000</td>
<td>0 1</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0000</td>
<td>0010</td>
<td>0X</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0011</td>
<td>0111</td>
<td>0X</td>
<td>1X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0100</td>
<td>1100</td>
<td>1X</td>
<td>1 0X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0111</td>
<td>0100</td>
<td>0X</td>
<td>1 01</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>0X</td>
<td>0 0X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1010</td>
<td>1000</td>
<td>X0</td>
<td>0 01</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1100</td>
<td>1111</td>
<td>X0</td>
<td>1 10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1111</td>
<td>1011</td>
<td>X0</td>
<td>0 X0</td>
<td></td>
</tr>
</tbody>
</table>
3-bit Gray Code Counter (cont)

- Generate K-Maps & obtain minimized SOPs

\[ K_x = R' + Y'Z' \]
\[ R_x = R' + Y'Z' \]
\[ S_x = R(X + Y)' = (R' + X + Y)' = R' + X' \]
\[ R_z = R' + X'Y + X'Y' \]

Further reductions:
\[ R_z = R' + X \oplus Y \]
\[ S_z = R(X \oplus Y)' = (R' + X \oplus Y)' = R' + X' \]

C. E. Stroud
Sequential Logic Design (11/03)
3-bit Gray Code Counter (cont)

- Logic diagram
- Next would come design verification via logic simulation
  - Debug as necessary to obtain working circuit
  - Update logic diagram, logic equations, etc. to reflect fixes
Sequential Logic Models

• Huffman model consists of two types:
  - Mealy model (aka Mealy machine)
    - Outputs are function of inputs and current state
      ✓ Outputs can change when inputs change or when current state changes
  - Moore model (aka Moore machine)
    - Outputs are function of current state only
      ✓ Outputs can change only when current state changes

- Primary Inputs
- Next State
- Primary Outputs

Comb Logic

Current State

Flip-Flips

Output Logic

Primary Inputs

Next State

Next State Logic

Flip-Flips

Primary Outputs

only for Mealy
Mealy & Moore State Diagrams

- **Mealy model**
  - Outputs associated with state transition
  - Output values shown with inputs

- **Moore model**
  - Outputs associated with states only
  - Output values shown with states
Mealy & Moore State Tables

<table>
<thead>
<tr>
<th>In</th>
<th>X</th>
<th>Y</th>
<th>X⁺</th>
<th>Y⁺</th>
<th>Dₓ</th>
<th>Dᵧ</th>
<th>O_{Mealy}</th>
<th>O_{Moore}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Note: next state (next state logic) is same for both Mealy & Moore – only output is different
# Mealy & Moore Design Examples

\[
\begin{array}{c|cc|c|c|c|}
X & Y & \text{In} & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & \boxed{1} & X & 0 \\
1 & \boxed{1} & 0 & X & 0 \\
\end{array}
\]

\[
D_X = \text{In}'Y + \text{In}X'Y'
\]

\[
\begin{array}{c|cc|c|c|c|}
X & Y & \text{In} & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & \boxed{1} & 0 & X & 0 \\
1 & \boxed{1} & 0 & 0 & \boxed{X} & 1 \\
\end{array}
\]

\[
D_Y = \text{In}X + \text{In}'X'Y'
\]

\[
\begin{array}{c|cc|c|c|c|}
X & Y & \text{In} & 00 & 01 & 11 & 10 \\
\hline
0 & 1 & 0 & \boxed{X} & 1 \\
1 & 1 & 1 & \boxed{X} & 0 \\
\end{array}
\]

\[
O_{\text{Mealy}} = \text{In}'Y' + \text{In}X'
\]

\[
\begin{array}{c|cc|c|c|c|}
X & Y & \text{In} & 00 & 01 & 11 & 10 \\
\hline
0 & \boxed{1} & 0 & \boxed{X} & 0 \\
1 & 1 & 0 & \boxed{X} & 0 \\
\end{array}
\]

\[
O_{\text{Moore}} = X'Y'
\]
Mealy & Moore Design Examples

\[ D_X = \text{In'}Y + \text{In}X'Y' \]
\[ D_Y = \text{In}X + \text{In'}X'Y' \]

\[ O_{\text{Mealy}} = \text{In'}Y' + \text{In}X' \]
\[ O_{\text{Moore}} = X'Y' \]

Note: \( O_{\text{Mealy}} \) is a function of \( \text{In} \) but \( O_{\text{Moore}} \) is not a function of \( \text{In} \)