Finite State Machines

- Sequential machine with finite number of combinations of states and inputs
  - FSM
  - Finite: there is a number that is larger

- One description: state table, diagram

- This chapter: algorithmic description of FSM
  - ASM
    - Like flow chart, with timing relationships between states
    - §8.3: binary multiplier example

Goal: Cover through ~ 8.4
ASM Chart

- Three elements
  - State box
  - Decision box (Condition box)
  - Conditional output box

- ASM Block

State Box: • one state box per ASM block
- Single entry path
- For each input combination, one exit
  - *Moore type: Output list describes signals that are asserted when state is entered

Condition Box: • tests input to determine exit path

Output Box: • *signals to be asserted (Mealy)
- Always preceded by a condition box
ASM Chart

- Three elements
  - State box
  - Decision box (Condition box)
  - Conditional output box

- ASM Block

State Box:
- one state box per ASM block
- single entry path
- for each input combination, one exit
  - Moore type: Output list describes signals that are asserted when state is entered

Condition Box:
- tests input to determine exit path

Output Box:
- signals to be asserted (Mealy)
  - always preceded by a condition box
Moore Example: Odd Parity Checker

- Check input stream $X$. Set output $z=1$ if number of 1's is odd, else set $z=0$.

$x: 010011000101$
$z: 01010101010001$

Diagram:

States:
- $A=0, B=1$
- $A/O$
- $B=1$

Transition:
- $X$: $A$ even
- $Z$: $B$ odd
- $F$: $F$
- $T$: $T$
Moore Example: Odd Parity Checker

- Check input stream $X$. Set output $Z = 1$ if number of 1's is odd, else set $Z = 0$.

$X: 010011000101$

$Z: 000100100001$

- Two ASM blocks
- Upper box is "blank" $\iff Z = 0$

<table>
<thead>
<tr>
<th>PS X</th>
<th>NS Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>
Mealy Example: Sequence Identifier (11)

Input X
Output Z

State Encoding

<table>
<thead>
<tr>
<th>PS</th>
<th>X</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example: Mixed Convention

<table>
<thead>
<tr>
<th>PS</th>
<th>X</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\{ "Moore" \\
\{ "Mealy" \\

\[ \text{Output } Z=1 \text{ directly} \]
Example: Vending Machine

- price = 15¢
- accepts nickels and dimes
- no change given
- synchronous reset from final state
- output = OPEN
- variables N, D, R
  - nickel dime reset

<table>
<thead>
<tr>
<th>N</th>
<th>D</th>
<th>outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no coin (hold)</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>add 5¢</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>add 10¢</td>
</tr>
</tbody>
</table>

Diagram:
- Initial state: 0¢
- Transition:
  - N → 5¢
  - D → 10¢
  - N, D → 15¢ (dispense / open)
- Reset: nickel dime reset
Example: Vending Machine

- price = 15¢
- accepts nickels and dimes
- no change given
- synchronous reset from final state
- output = OPEN
- variables N, D, R
  - nickel, dime, reset

Moore:

N D outcome
0 0 no coin (hold)
1 X add 5¢
0 1 add 10¢

reset

0¢

15¢

(Dispense / Open)

N D R
1 XX add 5¢
0 1 X add 10¢
0 0 X no coins / hold
X X O hold at dispense state
X X 1 reset

Output = OPEN
Vending Machine ASM Chart

0¢  00

N  F
D  T
F  F
F  T

5¢  01

N  F
D  T
T  T

10¢  10

N  F
D  T
F  F
F  T

15¢  11
OPEN
RESET

T  F
State Encoding

- States/Encoding?
- Inputs?
- Outputs?
State Encoding

- States/Encoding?
- Inputs? TEST
- Outputs? XYZ

“A E D
00 10 11"

“dense”
≡ minimum # of state
Variables

Implement with D FF’s

<table>
<thead>
<tr>
<th>PS</th>
<th>IN</th>
<th>NS</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, S₀</td>
<td>TEST</td>
<td>S, S₀</td>
<td>X Y Z</td>
</tr>
<tr>
<td>00 0</td>
<td>10</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>00 1</td>
<td>11</td>
<td>1 1</td>
<td>0</td>
</tr>
<tr>
<td>01 0</td>
<td>XX</td>
<td>XX</td>
<td>XXX</td>
</tr>
<tr>
<td>10 0</td>
<td>00</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>11 0</td>
<td>00</td>
<td>0 0</td>
<td>1</td>
</tr>
</tbody>
</table>
State Encoding

- States/Encoding?
- Inputs?
- Outputs?

\[ \begin{align*}
D_0 &= S_0(t+1) = S_1 \cdot \text{TEST} \\
D_1 &= S_1(t+1) = S_1 \\
X &= Y = S_1 \\
Z &= S_0 + S_1 \cdot \overline{\text{TEST}}
\end{align*} \]