Chapter #8: Finite State Machine Design

8.3 Alternative State Machine Representations

8.4 Mealy and Moore Machines Design
8.3 Alternative State Machine Representations

*Why State Diagrams Are Not Enough*

Not flexible enough for describing very complex finite state machines

Not suitable for gradual refinement of finite state machine

Do not obviously describe an *algorithm*: that is, well specified sequence of actions based on input data

algorithm = sequencing + data manipulation

separation of control and data

*Gradual shift towards program-like representations:*

- Algorithmic State Machine (ASM) Notation
- Hardware Description Languages (e.g., VHDL)
Alternative State Machine Representations

Algorithmic State Machine (ASM) Notation

Three Primitive Elements:

- State Box
- Decision Box
- Output Box

State Machine in one state block per state time

Single Entry Point

Unambiguous Exit Path for each combination of inputs

Outputs asserted high (.H) or low (.L); Immediate (I) or delayed til next clock
Alternative State Machine Representations

**ASM Notation**

**Condition Boxes:**
Ordering has no effect on final outcome

**Equivalent ASM charts:**

\[ A \text{ exits to } B \text{ on } (I_0 \land I_1) \text{ else exit to } C \]
Alternative State Machine Representations

Example: Odd Parity Checker

Input X, Output Z

Nothing in output list implies Z not asserted

Z asserted in State Odd

Trace paths to derive state transition tables
### Symbolic State Table:

<table>
<thead>
<tr>
<th>Input</th>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Even</td>
<td>Even</td>
<td>—</td>
</tr>
<tr>
<td>T</td>
<td>Even</td>
<td>Odd</td>
<td>—</td>
</tr>
<tr>
<td>F</td>
<td>Odd</td>
<td>Odd</td>
<td>A</td>
</tr>
<tr>
<td>T</td>
<td>Odd</td>
<td>Even</td>
<td>A</td>
</tr>
</tbody>
</table>

### Encoded State Table:

<table>
<thead>
<tr>
<th>Input</th>
<th>Present State</th>
<th>Next State</th>
<th>Output</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>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Alternative State Machine Representations

ASM Chart for Vending Machine

- **0¢**
  - **00**
  - **D**
  - **N**
  - **F**
  - **T**
  - **F**

- **5¢**
  - **01**
  - **D**
  - **N**
  - **F**
  - **T**
  - **F**

- **10¢**
  - **10**
  - **D**
  - **N**
  - **F**
  - **T**

- **15¢**
  - **11**
  - **H.Open**
  - **Reset**
  - **F**
  - **T**

© R.H. Katz  Transparency No. 15-7
8.4 Moore and Mealy Machine Design Procedure

Definitions

Moore Machine
Outputs are function solely of the current state
Outputs change synchronously with state changes

Mealy Machine
Outputs depend on state AND inputs
Input change causes an immediate output change
Asynchronous outputs
Moore and Mealy Machines

State Diagram Equivalents for Vending Machine FSM

Moore Machine

- Outputs are associated with State

Mealy Machine

- Outputs are associated with Transitions
Moore and Mealy Machines

Ex. FSM
asserts the single output whenever its input string has at least two 1’s in sequence.

Same I/O behavior

Different # of states

Equivalent ASM Charts

Mealy Machine typically has fewer states than Moore Machine for same output sequence.
Moore and Mealy Machines

Timing Behavior of Moore Machines

Ex. 1 -- Reverse engineer the Moore Machine

Two Techniques for Reverse Engineering:

- Ad Hoc: Try input combinations to derive transition table
- Formal: Derive transition by analyzing the circuit
Moore and Mealy Machines

Ad Hoc Reverse Engineering

Behavior in response to input sequence 1 0 1 0 1 0:

<table>
<thead>
<tr>
<th>Signal</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clk</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B / Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\Reset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Partially Derived State Transition Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
<th>A+</th>
<th>B+</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Signal tracing is acceptable for small FSM, but becomes untraceable for more complex FSM.
Moore and Mealy Machines

Formal Reverse Engineering

Derive transition table from next state and output combinational functions presented to the flipflops!

\[
\begin{align*}
Ja &= X \\
Jb &= X \\
Ka &= X \cdot \overline{B} \\
Kb &= X \oplus A
\end{align*}
\]

\[Z = B\]

FF excitation equations for J-K flipflop:
\[Q^+ = JQ' + K'Q\]
\[A^+ = Ja \cdot \overline{A} + Ka \cdot A = X \cdot \overline{A} + (X + B) \cdot A\]
\[B^+ = Jb \cdot B + Kb \cdot B = X \cdot B + (X \cdot A + X \cdot A) \cdot B\]

Next State K-Maps:

State 00, Input 0 \rightarrow State 00
State 01, Input 1 \rightarrow State 11
Moore and Mealy Machines

Complete ASM Chart for the Mystery Moore Machine

Note: All Outputs Associated With State Boxes
No Separate Output Boxes — Intrinsic in Moore Machines
Moore and Mealy Machines

Ex. 2 -- Reverse Engineering a Mealy Machine
One D flip-flop and one master/slave J-K flip flop

Input X, Output Z, State A, B
State register consists of D FF and J-K FF
Moore and Mealy Machine

Ad Hoc Method

Signal Trace of Input Sequence 101011:

<table>
<thead>
<tr>
<th>X</th>
<th>Clk</th>
<th>A</th>
<th>B</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>X=1</td>
<td>AB=00</td>
<td>Z=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X=0</td>
<td>AB=00</td>
<td>Z=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X=1</td>
<td>AB=01</td>
<td>Z=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X=0</td>
<td>AB=11</td>
<td>Z=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X=1</td>
<td>AB=10</td>
<td>Z=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X=1</td>
<td>AB=01</td>
<td>Z=0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Partial completion state transition table based on the signal trace

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
<th>A+</th>
<th>B+</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
**Moore and Mealy Machines**

**Formal Method**

\[ A^+ = B \cdot (A + X) = A \cdot B + B \cdot X \]

\[ B^+ = J_b \cdot \overline{B} + K_b \cdot B = (\overline{A} \oplus X) \cdot \overline{B} + X \cdot B \]

\[ = A \cdot B \cdot X + A \cdot B \cdot X + B \cdot X \]

\[ Z = A \cdot X + B \cdot X \]

---

**Missing Transitions and Outputs:**

- State 01, Input 0 -> State 01, Output 1
- State 10, Input 0 -> State 00, Output 0
- State 11, Input 1 -> State 11, Output 1
Moore and Mealy Machines

ASM Chart for Mystery Mealy Machine

S0 = 00, S1 = 01, S2 = 10, S3 = 11

NOTE: Some Outputs in Output Boxes as well as State Boxes
This is intrinsic in Mealy Machine implementation
Moore and Mealy Machines

Synchronous Mealy Machine

Breaks the direct connection between inputs and outputs by introducing storage elements. Prevents glitches as seen in Mealy Machine example.

Combinational Logic for Outputs and Next State

State Register

Clock

latched state AND outputs

avoids glitchy outputs!
HW #15 -- Section 8.3 & 8.4