CSEE 3827: Fundamentals of Computer Systems, Spring 2011

5. Finite State Machine Design

Prof. Martha Kim (martha@cs.columbia.edu)
Outline (H&H 3.5)

- Finite State Machines
  - Definition
    - Moore
    - Mealy
  - Design procedure
  - Examples
Finite State Machine (FSM)

FSM = State register + combinational logic

Stores the current state and loads the next state @ clock edge

Computation:
- Computes the next state
- Computes the outputs

Diagram:
- CLK
- S' → S
- Next State Logic
- Outputs
Finite State Machines (FSMs)

- Next state is determined by the current state and the inputs
- Two types of finite state machines differ in the output logic:
  - **Moore FSM**: outputs depend only on the current state
  - **Mealy FSM**: outputs depend on the current state and the inputs
Finite State Machine Example

• Traffic light controller

• Traffic sensors: TA, TB (TRUE when there is traffic)

• Lights: LA, LB
FSM Black Box

• Inputs: CLK, Reset, TA, TB

• Outputs: LA, LB
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs

\[ \text{Reset} \quad T_A \]

\[ \begin{align*}
S_0 \\
L_A & : \text{green} \\
L_B & : \text{red}
\end{align*} \]
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs

\[
\begin{align*}
S0 & : L_A: \text{green} \\
& \quad L_B: \text{red} \\
S1 & : L_A: \text{yellow} \\
& \quad L_B: \text{red} \\
S2 & : L_A: \text{red} \\
& \quad L_B: \text{green} \\
S3 & : L_A: \text{red} \\
& \quad L_B: \text{yellow}
\end{align*}
\]
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs
State transitions (from diagram) can be rewritten in a state transition table

\( (S = \text{current state}, \ S' = \text{next state}) \)

<table>
<thead>
<tr>
<th>Current State</th>
<th>Inputs</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGreen</td>
<td>TA: 0</td>
<td>AYellow</td>
</tr>
<tr>
<td>AYellow</td>
<td>TA: 1</td>
<td>AGreen</td>
</tr>
<tr>
<td>BGreen</td>
<td>TB: X</td>
<td>BYellow</td>
</tr>
<tr>
<td>BYellow</td>
<td>TB: X</td>
<td>AGreen</td>
</tr>
</tbody>
</table>

(diagram reprinted for reference)
FSM Encoded State Transition Table

- After selecting a state encoding the symbolic states in the transition table can be annotated with actual state / next state bits.

<table>
<thead>
<tr>
<th>Current State</th>
<th>Encoded Current State</th>
<th>Inputs</th>
<th>Next State</th>
<th>Encoded Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S1</td>
<td>S0</td>
<td>TA</td>
<td>TB</td>
</tr>
<tr>
<td>AGreen</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>AGreen</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>AYellow</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BGreen</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>BGreen</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>BYellow</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- One can then compute the next state logic:

\[
\begin{align*}
S1' &= S1 \text{ XOR } S0 \\
S0' &= \overline{S1}S0\text{TA} + S1\overline{S0}\text{TB}
\end{align*}
\]
FSM Output Table

- FSM output logic is computed in much the same manner as the next state logic.

- Because this is a Moore machine, outputs are a function of the current state only (were it a Mealy output would be a function of current state + inputs).

**output encoding**

<table>
<thead>
<tr>
<th>Output</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>0 0</td>
</tr>
<tr>
<td>Yellow</td>
<td>0 1</td>
</tr>
<tr>
<td>Red</td>
<td>1 0</td>
</tr>
</tbody>
</table>

**output truth table**

<table>
<thead>
<tr>
<th>State</th>
<th>State</th>
<th>LA</th>
<th>LB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1</td>
<td>LA1</td>
<td>LB1</td>
</tr>
<tr>
<td>AGreen</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>AYellow</td>
<td>0 1</td>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>BGreen</td>
<td>1 0</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td>BYellow</td>
<td>1 1</td>
<td>1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

- Compute output bits as function of state bits
  - LA1 = S1; LA0 = S1\text{\textasciicircum}S0
  - LB1 = \text{\textasciicircum}S1; LB0 = S1\text{\textasciicircum}S0
  - red light
FSM Schematic: State Register

state register
FSM Schematic: Next State Logic

inputs

next state logic

state register
What does the Reset signal do to this machine?

S = 00 = A
Green = A gets green light, B gets red light
FSM State Encoding

• Binary encoding: i.e., for four states, 00, 01, 10, 11

• One-hot encoding
  • One state bit per state
  • Only one state bit is HIGH at once
  • I.e., for four states, 0001, 0010, 0100, 1000
  • Requires more flip-flops
  • Often next state and output logic is simpler

• Sometimes a semantically meaningful encoding makes the most sense (e.g., a faucet controller with a volume state and a temperature state)
Moore v. Mealy FSM

Alyssa P. Hacker has a snail that crawls down a paper tape with 1’s and 0’s on it. The snail smiles whenever the last four digits it has crawled over are 1101. Design Moore and Mealy FSMs of the snail’s brain.
Factoring State Machines

- Break complex FSMs into smaller interacting FSMs

- Example: Modify the traffic light controller to have a Parade Mode.
  - The FSM receives two more inputs: P, R
  - When $P = 1$, it enters Parade Mode and the Bravado Blvd. light stays green.
  - When $R = 1$, it leaves Parade Mode
Parade FSM

- Unfactored FSM

- Factored FSM
Unfactored FSM State Transition Diagram
Factored FSM State Transition Diagram

**Lights FSM**
- **S0**
  - \( L_A: \text{green} \)
  - \( L_B: \text{red} \)
- **S1**
  - \( L_A: \text{yellow} \)
  - \( L_B: \text{red} \)
- **S2**
  - \( L_A: \text{red} \)
  - \( L_B: \text{green} \)
- **S3**
  - \( L_A: \text{red} \)
  - \( L_B: \text{yellow} \)

**Mode FSM**
- **S0**
  - \( M: 0 \)
- **S1**
  - \( M: 1 \)
FSM Design Procedure

- Identify the inputs and outputs
- Sketch a state transition diagram
- Write a state transition table
- Select state encodings
- For a Moore machine:
  - Rewrite the state transition table with the selected state encodings
  - Write the output table
- For a Mealy machine:
  - Rewrite the combined state transition and output table with the selected state encodings
- Write Boolean equations for the next state and output logic
- Sketch the circuit schematic
FSM timing characteristics

MEALY

MOORE
Advanced FSM design and implementation

**Unused states:** extra state encodings (e.g., using 3 FFs to represent 6 states leaves 2 unused states) can be treated as “don’t care” values and used to simplify the combinational logic

**State minimization:** two states are equivalent if they transition to the same or equivalent states on the same inputs (while producing the same outputs in the case of a Mealy machine)