Lecture 20

- Logistics
  - HW6 due Wednesday
  - Lab 7 this week (Tuesday exception)
  - Midterm 2 Friday (covers material up to simple FSM (today))
  - Review on Thursday
  - Yoky office hour on Friday moved to Thursday 12-1:20pm online

- Last lecture
  - Counter design
  - Finite state machine – started vending machine example

- Today
  - Continue on the vending machine example
  - Moore/Mealy machines

The “WHY” slide

- Finite State Machine (FSM)
  - This is what we have been waiting for in this class. Using combinational and sequential logics, now you can design a lot of clever digital logic circuits for functional products. We will learn different steps you take to go from word problems to logic circuits.

- Moore/Mealy machines
  - There are two different ways to express the FSMs with respect to the output. Both have different advantages so it is good to know them.
FSM design

- Counter-design procedure
  1. State diagram
  2. State-transition table
  3. Next-state logic minimization
  4. Implement the design

- FSM-design procedure
  1. State diagram
  2. state-transition table
  3. State minimization
  4. State encoding
  5. Next-state logic minimization
  6. Implement the design

Example: A vending machine

- 15 cents for a cup of coffee
- Doesn’t take pennies or quarters
- Doesn’t provide any change

- FSM-design procedure
  1. State diagram
  2. state-transition table
  3. State minimization
  4. State encoding
  5. Next-state logic minimization
  6. Implement the design
A vending machine: (conceptual) state diagram

A vending machine: State transition table

<table>
<thead>
<tr>
<th>present state</th>
<th>inputs</th>
<th>next state</th>
<th>output open</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0 0</td>
<td>S0 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>S1 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>S2 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>0 0</td>
<td>S1 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>S3 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>S4 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>0 0</td>
<td>S2 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>S5 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>S6 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>0 0</td>
<td>S3 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>S7 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>S8 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>x x</td>
<td>S4 1</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>x x</td>
<td>S5 1</td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>x x</td>
<td>S6 1</td>
<td></td>
</tr>
<tr>
<td>S7</td>
<td>x x</td>
<td>S7 1</td>
<td></td>
</tr>
<tr>
<td>S8</td>
<td>x x</td>
<td>S8 1</td>
<td></td>
</tr>
</tbody>
</table>
A vending machine: State minimization

symbolic state table

A vending machine: State encoding
A vending machine: Logic minimization

D1 = Q1 + D + Q0 N
D0 = Q0' N + Q0 N' + Q1 N + Q1 D
OPEN = Q1 Q0

A vending machine: Implementation
Generalized FSM model: Moore and Mealy

- Combinational logic computes next state and outputs
  - Next state is a function of current state and inputs
  - Outputs are functions of
    - Current state (Moore machine)
    - Current state and inputs (Mealy machine)

Moore versus Mealy machines

Moore machine
- Outputs are a function of current state
- Outputs change synchronously with state changes

Mealy machine
- Outputs depend on state and on inputs
- Input changes can cause immediate output changes (asynchronous)
Example 10 -> 01: Moore or Mealy?

- Circuits recognize AB=10 followed by AB=01
  - What kinds of machines are they?

CSE370, Lecture 20

Example 01/10 detector: a Moore machine

- Output is a function of state only
  - Specify output in the state bubble

CSE370, Lecture 20
**Example 01/10 detector: a Mealy machine**

- Output is a function of state and inputs
  - Specify outputs on transition arcs

<table>
<thead>
<tr>
<th>reset</th>
<th>input</th>
<th>current state</th>
<th>next state</th>
<th>current output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>—</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>A</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>A</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>B</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>B</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>C</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>C</td>
<td>C</td>
<td>0</td>
</tr>
</tbody>
</table>

**Comparing Moore and Mealy machines**

- **Moore machines**
  - Safer to use because outputs change at clock edge
  - May take additional logic to decode state into outputs

- **Mealy machines**
  - Typically have fewer states
  - React faster to inputs — don't wait for clock
  - Asynchronous outputs can be dangerous

- **We often design synchronous Mealy machines**
  - Design a Mealy machine
  - Then register the outputs
Synchronous (registered) Mealy machine

- Registered state and registered outputs
  - No glitches on outputs
  - No race conditions between communicating machines

Example 0 -> 1: Moore or Mealy?

- Recognize A,B = 0,1
  - Mealy or Moore?