Word description to state diagram

- Design a vending machine controller that will release (output signal r) an apple as soon as 30¢ have been inserted.
- The machine’s sensors will clock your controller when an event occurs. The machine accepts only dimes (input signal d) and quarters (input signal q) and does not give change.
- When an apple is removed from the open machine, it indicates this by clocking the controller with an input of d.
- The sensors use only a single bit to communicate with the controller.

State diagram to state table

<table>
<thead>
<tr>
<th>Current state</th>
<th>Next state</th>
<th>Output (r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=0</td>
<td>i=1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>E</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

Moore block diagram

Mealy block diagram
Mealy FSMs

```
<table>
<thead>
<tr>
<th>s</th>
<th>s'/q</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D/0</td>
</tr>
<tr>
<td>B</td>
<td>C/1</td>
</tr>
<tr>
<td>C</td>
<td>A/0</td>
</tr>
<tr>
<td>D</td>
<td>C/1</td>
</tr>
</tbody>
</table>
```

**FSM design summary**

- Specify requirements in natural form
- Manually derive state diagram
  - Automatic way to go from English to FSM, however more theory required
  - Can minimize state count, however, more theory also required
  - See me if you want more information on this, or take a compilers course and a graduate-level switching theory course, or take my ECE 303
- Assign values to states to minimize logic complexity
- Optimize implementation of state and output functions

**Review: Clocking conventions**

- Active-high transparent
- Active-low transparent
- Positive (rising) edge
- Negative (falling) edge

**Latch and flip-flop equations**

- RS
  
  \[ Q^+ = S + R \cdot Q \]
- T
  
  \[ Q^+ = T \oplus Q \]

**JK latch**

- Use output feedback to ensure that \( RS \neq 11 \)
  
  \[ Q^+ = Q \oplus R \cdot J \]
Falling edge-triggered D flip-flop

- Use two stages of latches
  - When clock is high
    - First stage samples input w/o. changing second stage
    - Second stage holds value
  - When clock goes low
    - First stage holds value and sets or resets second stage
    - Second stage transmits first stage
  - $Q^+ = D$
  - One of the most commonly used flip-flops

Clock switching

Inputs sampled on falling edge, outputs change after falling edge

Another view of an edge-triggered DFF

Edge triggered timing
RS clocked latch

- Storage element in narrow width clocked systems
- Dangerous
- Fundamental building block of many flip-flop types

D flip-flop

- Minimizes input wiring
- Simple to use
- Common choice for basic memory elements in sequential circuits

Schmitt triggers

![Schmitt trigger diagram]

- Mechanical switches bounce!
- What happens if multiple pulses?
  - Multiple state transitions
  - Need to clean up signal

Debouncing

- Versatile building block
- Building block for D and T flip-flops
- Has two inputs resulting in increased wiring complexity
- Edge-triggered varieties exist

Assigned reading

- Review Sections 5.1–5.7
  - If FSMs don’t make sense now, please ask questions, or see me
  - FSMs are tricky at first – Almost everybody has this moment of epiphany at which they suddenly make sense
- Section 9.1–9.6

Computer geek culture references

- Parsers and lexical analyzers
- Writing problem-specific languages
- Lex and yacc
- Flex and bison