State & Finite State Machines

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

See: P&H Appendix C.7, C.8, C.10, C.11
Prelim 1: 3/18/2010 (evening)
Prelim 2: 4/27/2010 (evening)
Unstable Devices
Stable and unstable equilibria?
So far:

- **Current output** depends only on **current input** (no internal state)

Need a way to record data

- ... a way to build **stateful** circuits
- ... a state-holding device
Set-Reset (SR) Latch

Stores a value $Q$ and its complement $\overline{Q}$

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>$\overline{Q}$</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>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data (D) Latch

<table>
<thead>
<tr>
<th>D</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Level Sensitive D Latch

Clock high:
set/reset (according to D)

Clock low:
keep state (ignore D)
Clock helps coordinate state changes

- Usually generated by an oscillating crystal
- Fixed period; frequency = 1/period
D Flip-Flop

- Edge-Triggered
- Data is captured when clock is high
- Outputs change only on falling edges

Diagram:

- D Input
- Q Output
- Lbar Input
- F Control
- clk Clock Input

Timing:

- clk Clock Signal
- D Data Signal
- F Control Signal
- Q Output Signal
Clock Disciplines

Level sensitive
- State changes when clock is high (or low)

Edge triggered
- State changes at clock edge

positive edge-triggered

negative edge-triggered
Register

- D flip-flops in parallel
- shared clock
- extra clocked inputs: write_enable, reset, ...

4-bit reg
Metastability and Asynchronous Inputs

1-bit reg

Clk
Q: What happens if input is changes near clock edge?
A: Google “Buridan’s Principle” by Leslie Lamport
Clock Methodology

- Negative edge, synchronous
  - Signals must be stable near falling clock edge

- Positive edge synchronous
- Asynchronous, multiple clocks, . . .
Finite State Machine

- inputs from external world
- outputs to external world
- internal state
- combinational logic
Input: **up** or **down**
Output: **on** or **off**
States: A, B, C, or D
Input: 0=up or 1=down
Output: 1=on or 1=off
States: 00=A, 01=B, 10=C, or 11=D
General Case: **Mealy Machine**

Outputs and next state depend on both current state and input
Special Case: **Moore Machine**

Outputs depend only on current state
Moore Machine Example

Legend

Input: up or down
Output: on or off
States: A, B, C, or D
Digital Door Lock

Inputs:
• keycodes from keypad
• clock

Outputs:
• “unlock” signal
• display how many keys pressed so far
Assumptions:

- signals are synchronized to clock
- Password is B-A-B

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Ø (no key)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>‘A’ pressed</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>‘B’ pressed</td>
</tr>
</tbody>
</table>
Assumptions:

- High pulse on U unlocks door
Door Lock: Simplified State Diagram

- **Idle** "0"
  - Transition: "B" to G2
  - Transition: "B" to B1
  - Transition: "B" to B2

- **G1** "1"
  - Transition: "A" to G2
  - Transition: "B" to G3

- **G2** "2"
  - Transition: "A" to G1
  - Transition: "B" to G3

- **G3** "3", U
  - Transition: "B" to G2
  - Transition: any to G3

- **B1** "1"
  - Transition: "B" to G2
  - Transition: "B" to G3

- **B2** "2"
  - Transition: "B" to G2
  - Transition: "B" to G3

- **Ø**
  - Transition: Idle to G1
  - Transition: Idle to G2
  - Transition: Idle to G3
  - Transition: Idle to B1
  - Transition: Idle to B2
### State Table Encoding

<table>
<thead>
<tr>
<th>$S_2$</th>
<th>$S_1$</th>
<th>$S_0$</th>
<th>$D_3$</th>
<th>$D_2$</th>
<th>$D_1$</th>
<th>$D_0$</th>
<th>$U$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>$S_2$</th>
<th>$S_1$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S_2$</th>
<th>$S_1$</th>
<th>$S_0$</th>
<th>$K$</th>
<th>$A$</th>
<th>$B$</th>
<th>$S'_2$</th>
<th>$S'_1$</th>
<th>$S'_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

D3 encodes the state transitions and is used to determine the next state. The state table encodes the current state, output, and next state based on inputs and key presses.
Strategy:
(1) Draw a state diagram (e.g. Moore Machine)
(2) Write output and next-state tables
(3) Encode states, inputs, and outputs as bits
(4) Determine logic equations for next state and outputs
We can now build interesting devices with sensors

• Using combinational logic

We can also store data values

• Stateful circuit elements (D Flip Flops, Registers, ...)
• Clock to synchronize state changes
• But be wary of asynchronous (un-clocked) inputs
• State Machines or Ad-Hoc Circuits