Lec 3: State and Finite State Machines

Kavita Bala

CS 3410, Fall 2008

Computer Science

Cornell University

Announcements

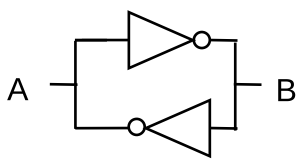
- Class newsgroup created
- Use it for partner finding
- First assignment is to find partners
  - Due this Friday
- Sections are on this week
- HW 1 out tomorrow
  - Work alone

Stateful Components

- Until now is combinatorial logic
  - Output is computed when inputs are present
  - System has no internal state
  - Nothing computed in the present can depend on what happened in the past!

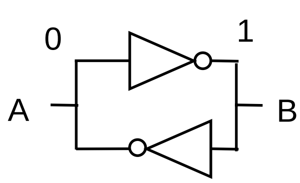
- Need a way to record data
- Need a way to build stateful circuits
- Need a state-holding device

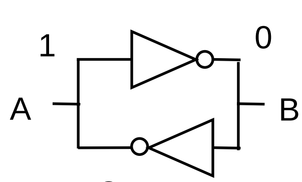
Bistable Devices



A Simple Device

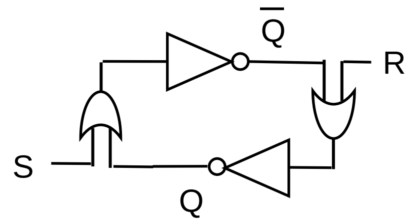
- In stable state, $\bar{A} = B$





- How do we change the state?

SR Latch

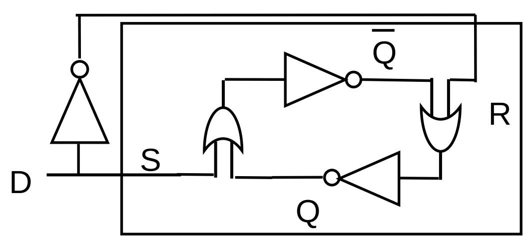


S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	?	?

- Set-Reset (S-R) Latch
- Q: Stored value and its complement
- S=1 and R=1 ?

D Latch

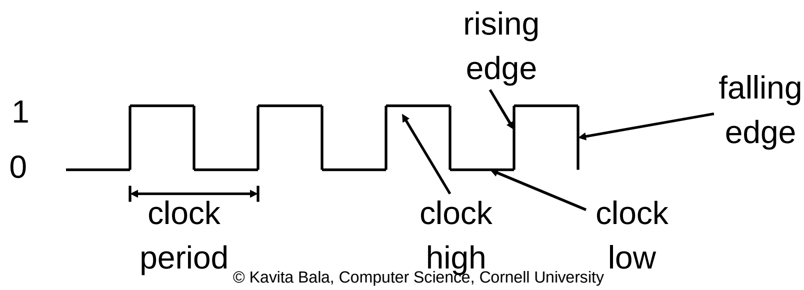
D	$\overline{D}$	Q	$\overline{Q}$
0	1	0	1
1	0	1	0



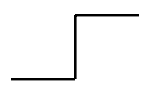
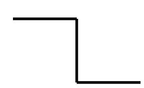
- Data Latch
  - Easier to use than an SR latch
  - No possibility of entering an undefined state
- When D changes, Q changes
  - … immediately (after a delay of 2 Ors and 2 NOTs)
- Need to control when the output changes

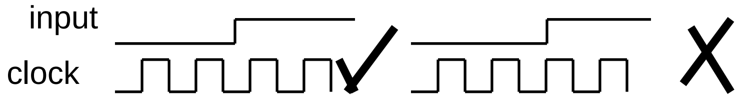
Clocks

- Clocks help with modifying the contents of state-holding elements
- A free running signal
  - Generated by an oscillating crystal
- Clock signal has a fixed cycle time : cycle period
- Clock frequency = 1/cycle time



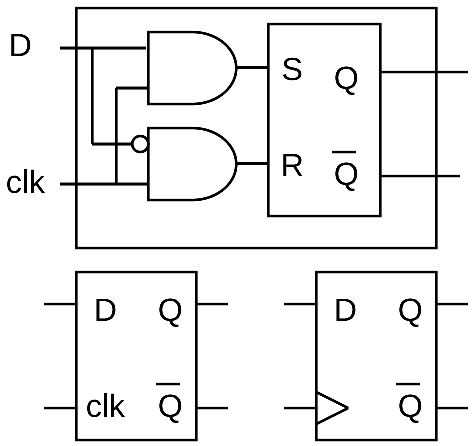
Edge-triggering

- Can design circuits to change on the rising or falling edge
- Trigger on rising edge = positive edge-triggered 
- Trigger on falling edge = negative edge-triggered 
- Inputs must be stable just before the triggering edge



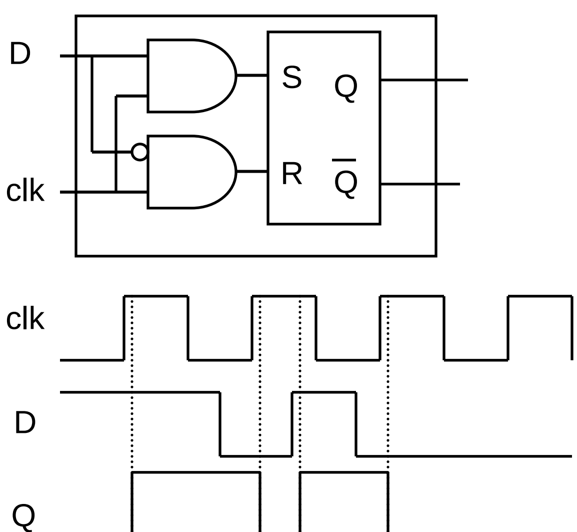
First Attempt

- How does the output behave?

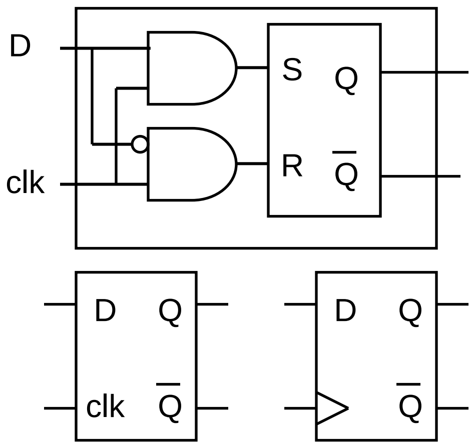


First Attempt

- How does the output behave?

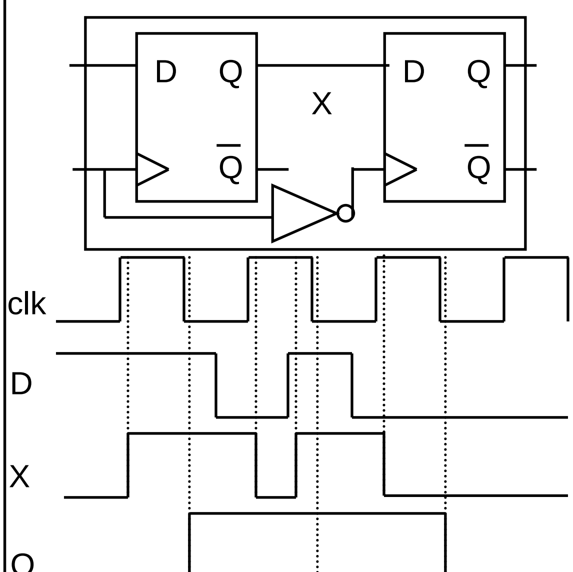


First Attempt



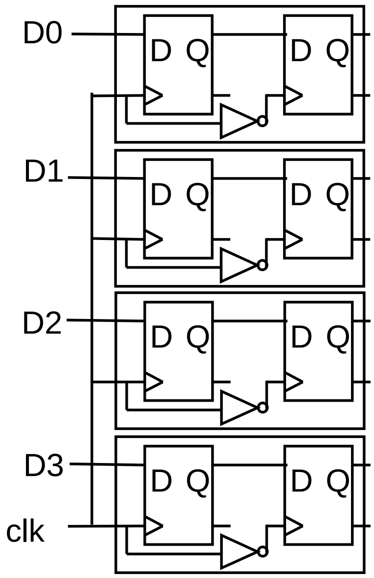
- How does the output behave?
- Changes in D that occur when the clock is low are deferred until clock high
- Changes when clock is high are registered immediately

Master-Slave Flip-Flop

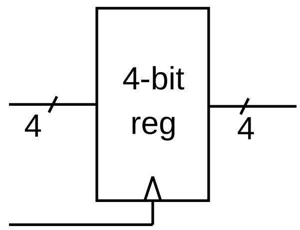


- Outputs change only on falling edges
- Data is captured on rising edges
- 1 cycle delay
  - but works out perfectly – data for the next stage is ready 1 cycle ahead of time

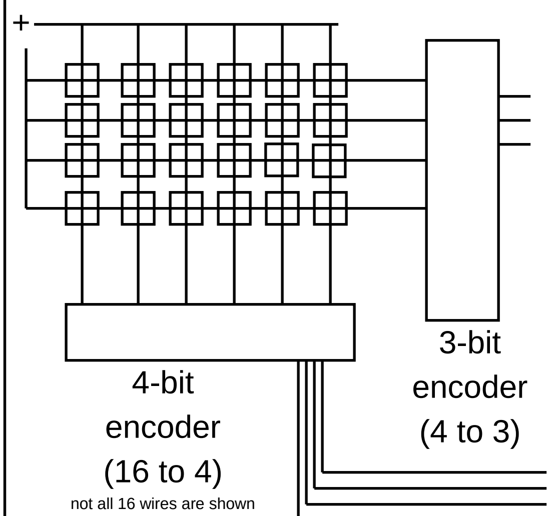
Registers



- A register is simply a set of master-slave flip-flops in parallel with a shared clock

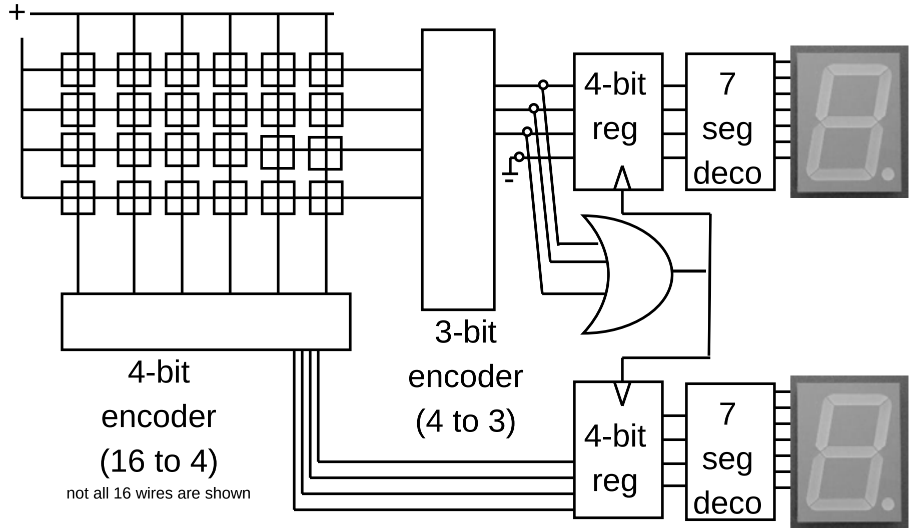


Example: Keyboard



- When a key is pressed
  - Compute a 7-bit key identifier
- Store this keycode
  - The computer may not be ready to read it right away

Keyboard with Last Key Display



Summary

- We can now build interesting devices with sensors
  - Using combinatorial logic
- We can also store data values
  - In state-holding elements
  - Coupled with clocks

Finite State Machines

Finite State Machines

- An electronic machine which has
  - external inputs
  - externally visible outputs
  - internal state
- Output and next state depend on
  - inputs
  - current state

Abstract Model of FSM

Machine is

$$M = (S, I, O, \delta)$$

S: Finite set of states

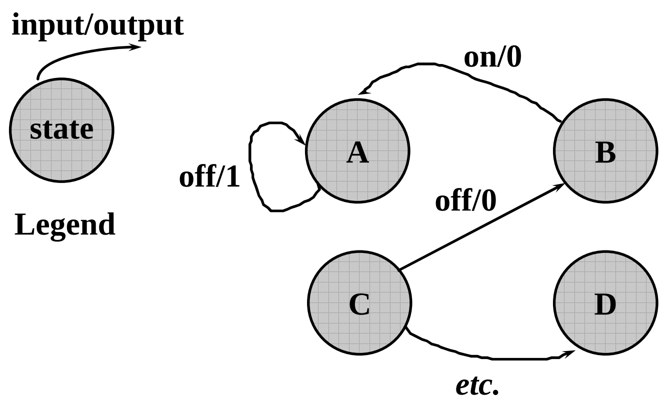
I: Finite set of inputs

O: Finite set of outputs

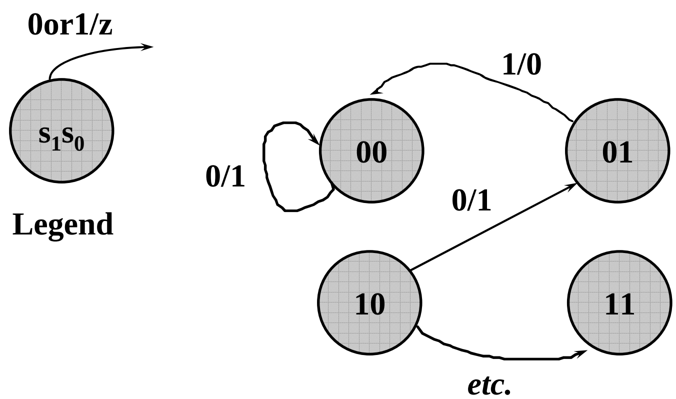
δ: State transition function

- Next state depends on present input and present state

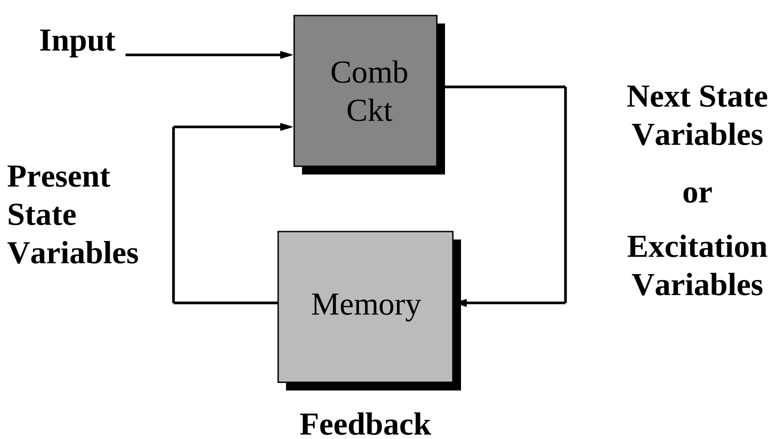
Primitive State Diagram



Machine State Diagram



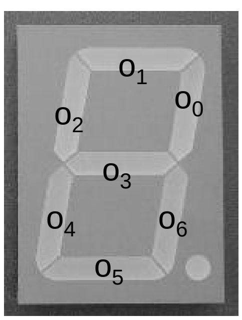
Automata Model



Designing a FSM

- Draw a state diagram
- Write down state transition table
- Assign numbers to states
- Determine logic equations for all flip-flops and outputs

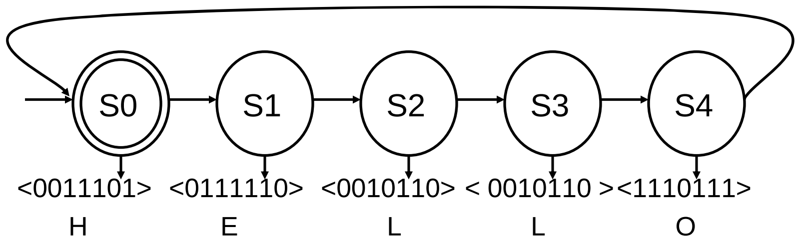
A Simple Example



- Goal: flash hello on LEDs
- Inputs: clock
- Outputs: Just one
  7-segment LED

- Flash “h” then “e” then “l” then “l” then “o”
  - h = <0011101>
  - e = <0111110>
  - l = <0010110>
  - o = <1110111>

HELLObox: State Diagram



- Determine the transitions
  - label all edges (transitions) with the inputs that cause them, unlabeled edges are unconditional transitions
  - show start state

HELLObox: State Table

- Build state table
  - rote encoding of the state diagram

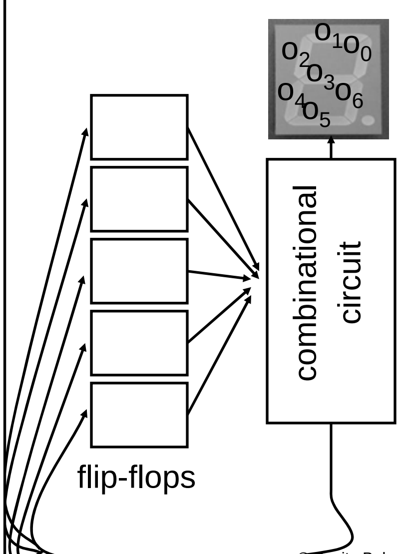
Current State	Next State	Output
S0	S1	0011101
S1	S2	0111110
S2	S3	0010110
S3	S4	0010110
S4	S0	1110111

HELLObox: State Assignment 1

- Assign bit patterns to states
  - Try to make resulting device simple
  - One option is shown
- Determine logic equations for
  - every bit of output
  - next state
  - for every flip-flop and output

Current State	Next State	Output
000	001	0011101
001	010	0111110
010	011	0010110
011	100	0010110
100	000	1110111

HELLObox #1



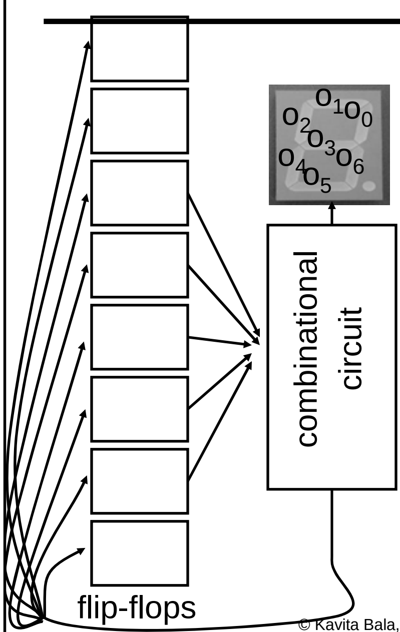
- 10 bits of information (7 outputs + 3 bits of next state) are computed by the combinatorial circuit
- All 10 bits have non-trivial logic equations

HELLObox: State Assignment 2

- Assign bit patterns to states to make the resulting device simple
- Here, we use far more bits than necessary
  - to simplify the combinatorial circuit

Current State	Next State	Output
00111010	01111100	0011101
01111100	00101100	0111110
00101100	00101101	0010110
00101101	11101110	0010110
11101110	00111010	1110111

HELLObox #2



- 15 bits of information (7 outputs + 8 bits of next state) are computed by the combinatorial circuit
- 8 bits have non-trivial logic equations
  - all 7 outputs are simple pass-throughs