State and Finite State Machines

Anne Bracy

CS 3410

Computer Science

Cornell University

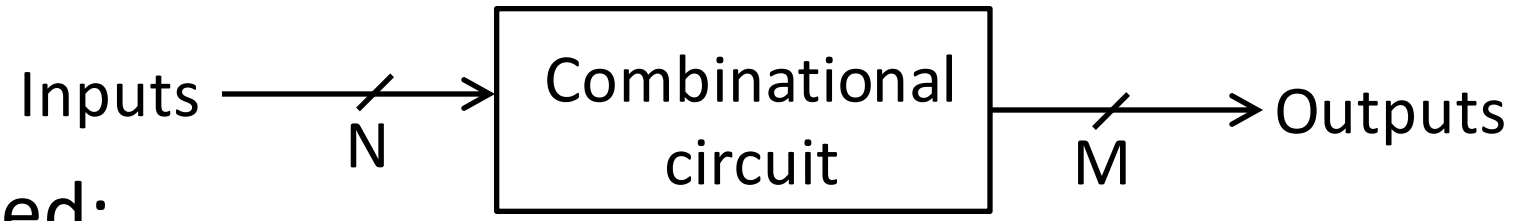
The slides are the product of many rounds of teaching CS 3410 by Professors Weatherspoon, Bala, Bracy, and Sirer.

See P&H Appendix B.7. B.8, B.10, B.11

Stateful Components

Combinational logic

- Output computed directly from inputs
- System has no internal state
- Nothing depends on the past!



Need:

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

Enter: Sequential Logic & Finite State Machines

Goals for Today

State

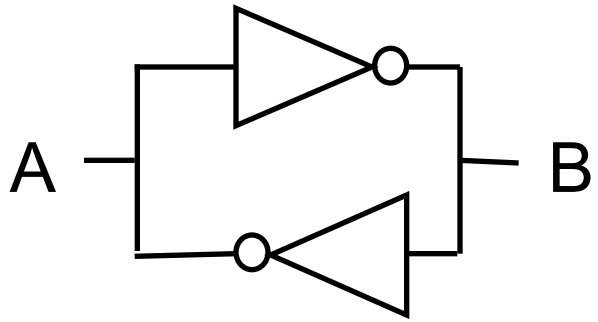
- Storing 1 bit
  - Bistable Circuit
  - Set-Reset Latch
  - D Latch
  - D Flip-Flops
- Storing N bits:
  - Registers
  - *More options in later lectures*

Finite State Machines (FSM)

- Mealy and Moore Machines
- Serial Adder Example

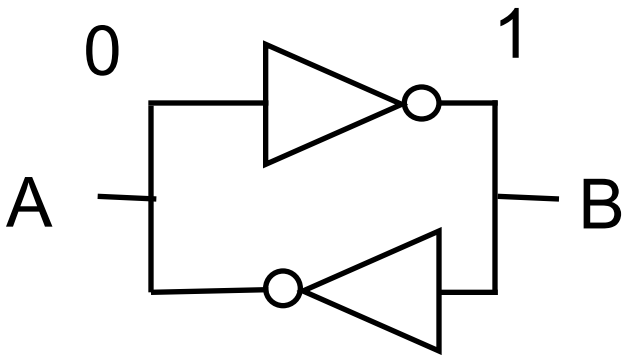
Round 1: Bistable Circuit

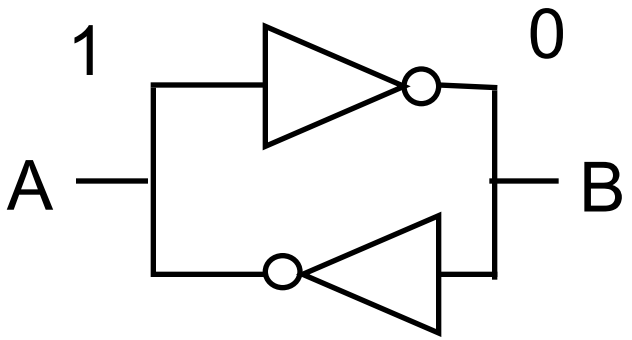
- Stable and unstable equilibria?



A Simple Device

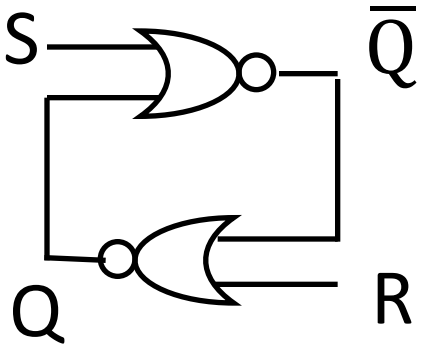
In stable state, $\bar{A} = B$

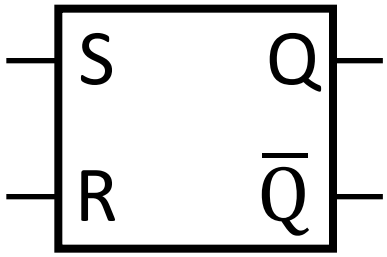




How do we change the state?

Round 2: Set-Reset (SR) Latch



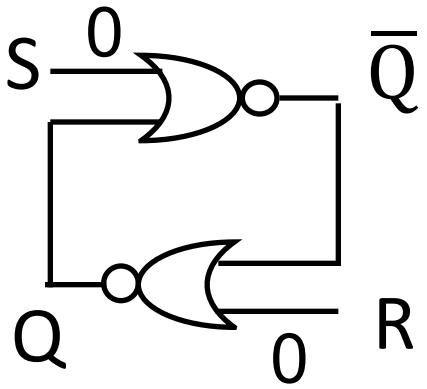


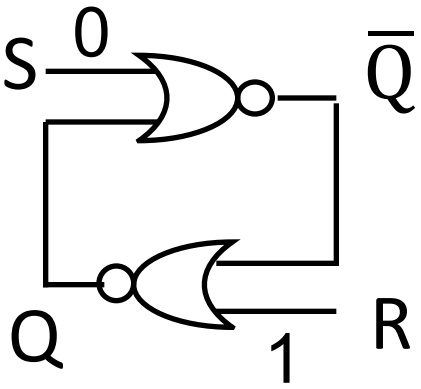
Stores a value Q and its complement

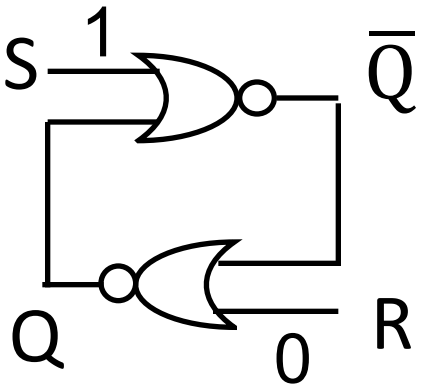
S	R	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		

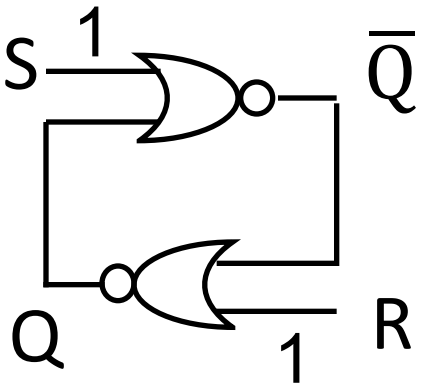
For reference:

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0





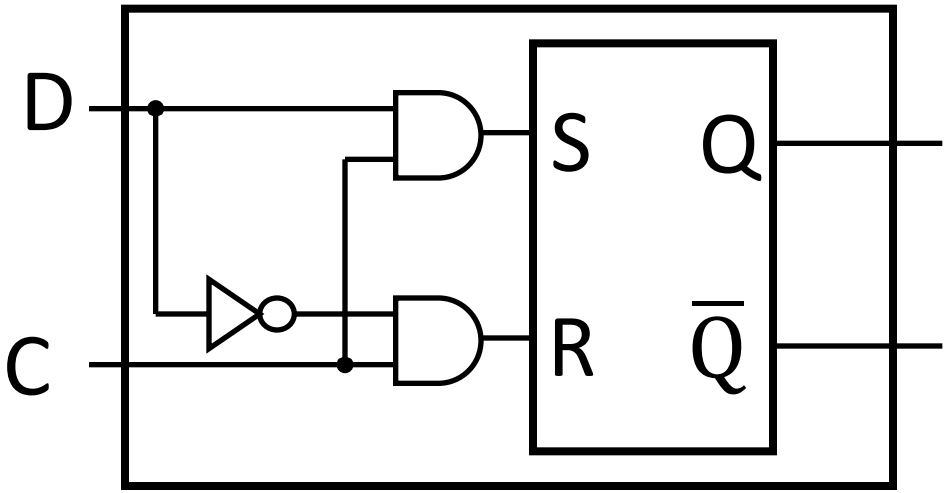




iClicker Question

Frequency should be set to **AA**

Round 3: D Latch (1)

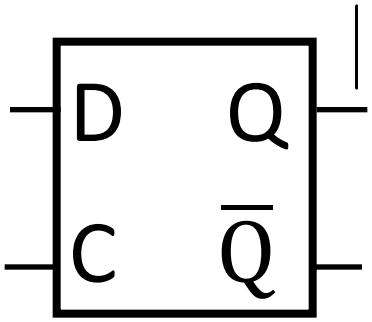


- Inverter prevents SR Latch from entering 1,1 state
- C = enables change

C = 1, D Latch *transparent*: set/reset (according to D)

C = 0, D Latch *opaque*: keep state (ignore D)

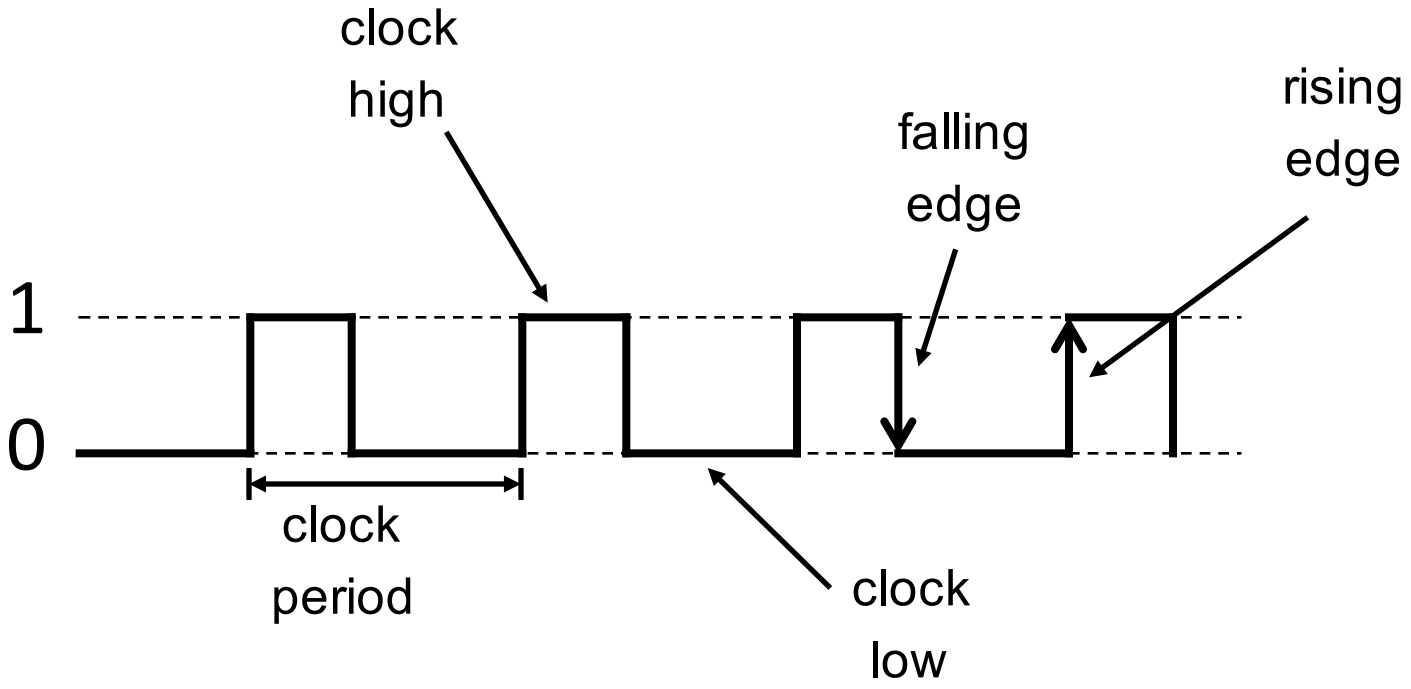
C	D	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		



Aside: Clocks

Clock helps coordinate state changes

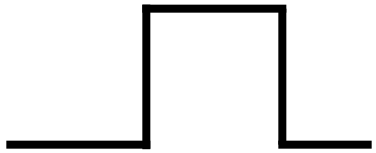
- Fixed period
- Frequency = 1/period



Clock Disciplines

Level sensitive

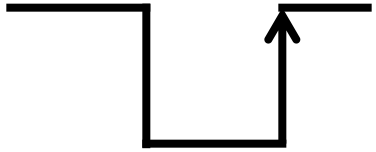
- State changes when clock is high (or low)



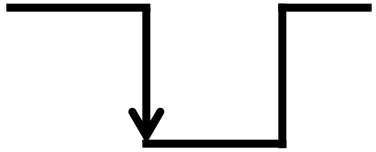
Edge triggered

- State changes at clock edge

positive edge-triggered



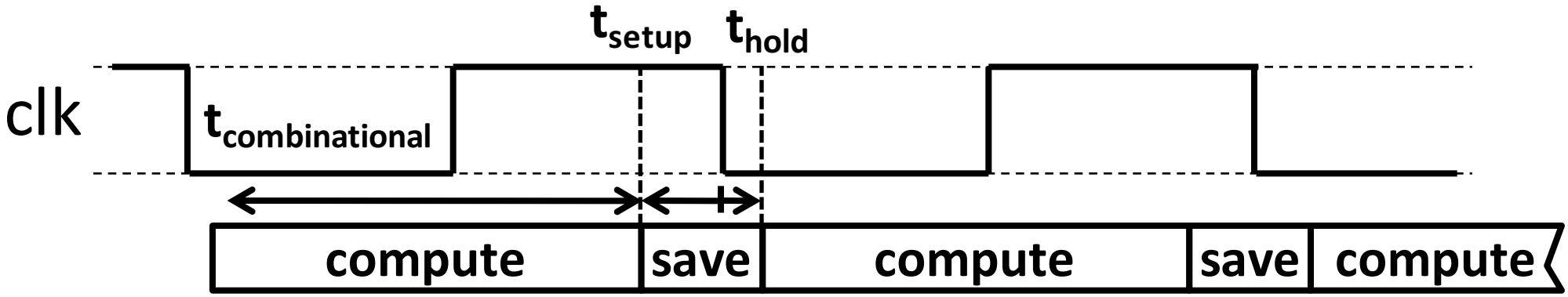
negative edge-triggered



Clock Methodology

Clock Methodology

- Negative edge, synchronous

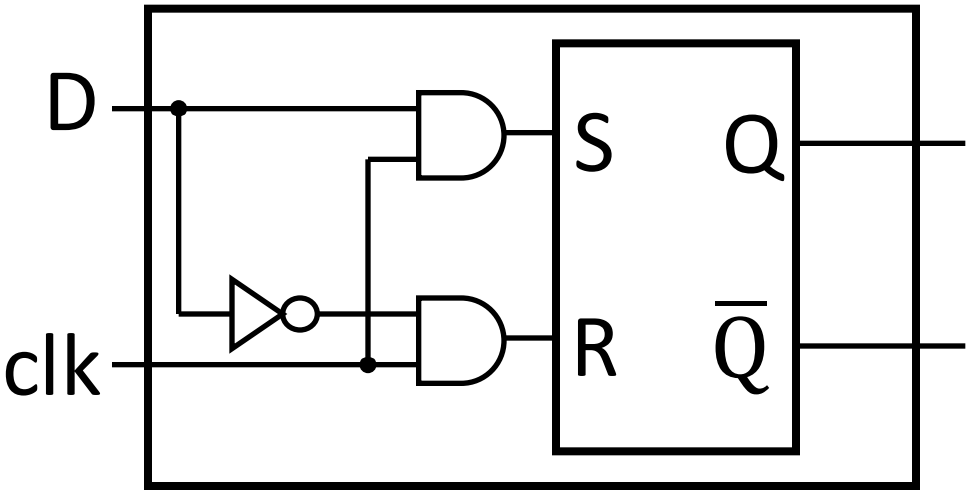


Edge-Triggered → signals must be stable near falling edge

"near" = before and after

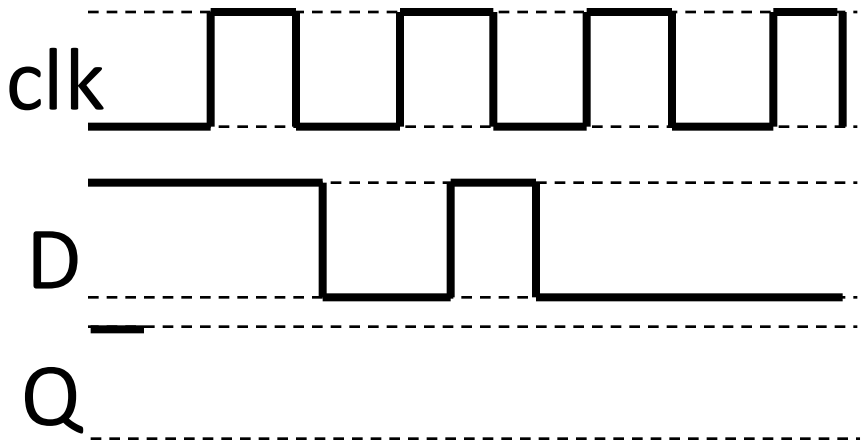
t_{setup} t_{hold}

Round 3: D Latch (2)

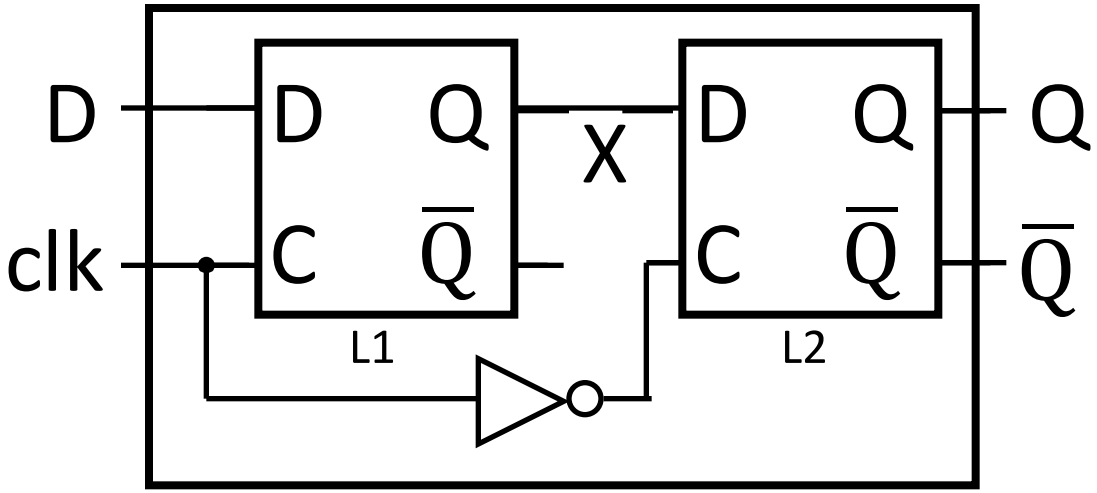


- Level sensitive
- Inverter prevents SR Latch from entering 1,1 state
- clk = enables change

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



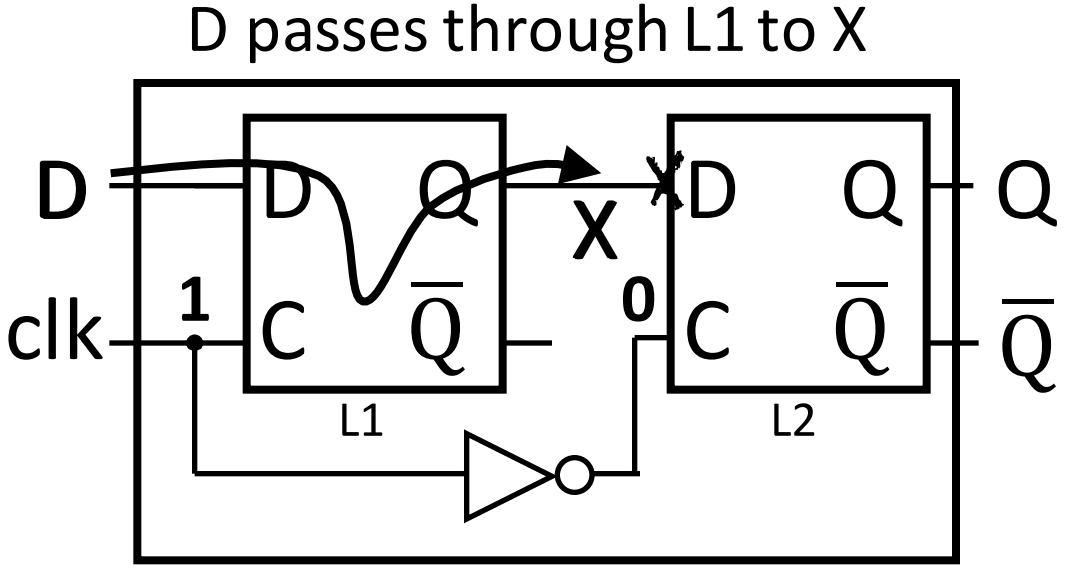
Round 4: D Flip-Flop



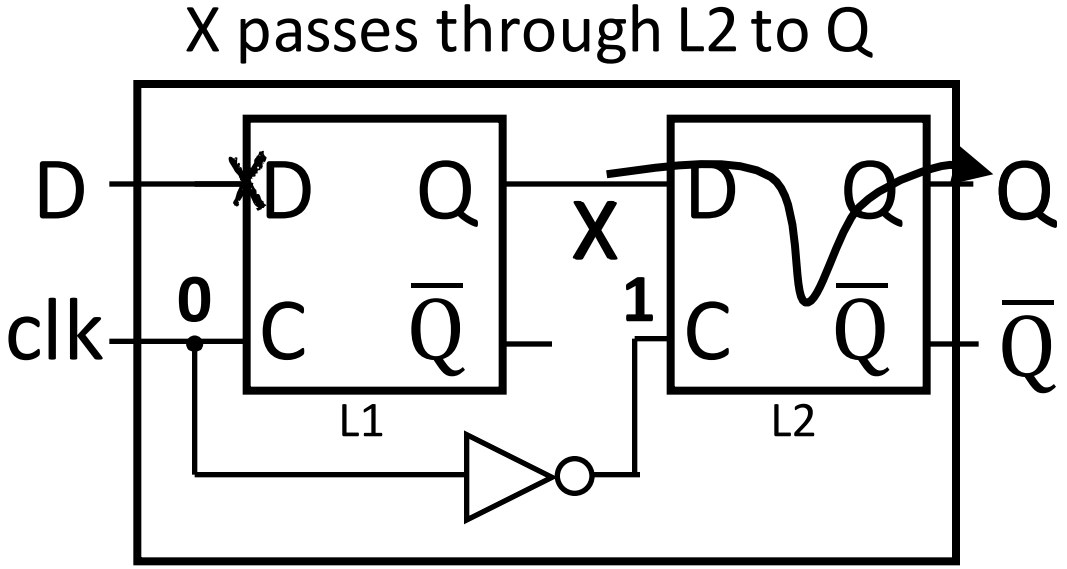
- Edge-Triggered
- Data captured when clock high
- Output changes only on falling edges

Round 4: D Flip-Flop

Clock = 1: L1 *transparent*
L2 *opaque*



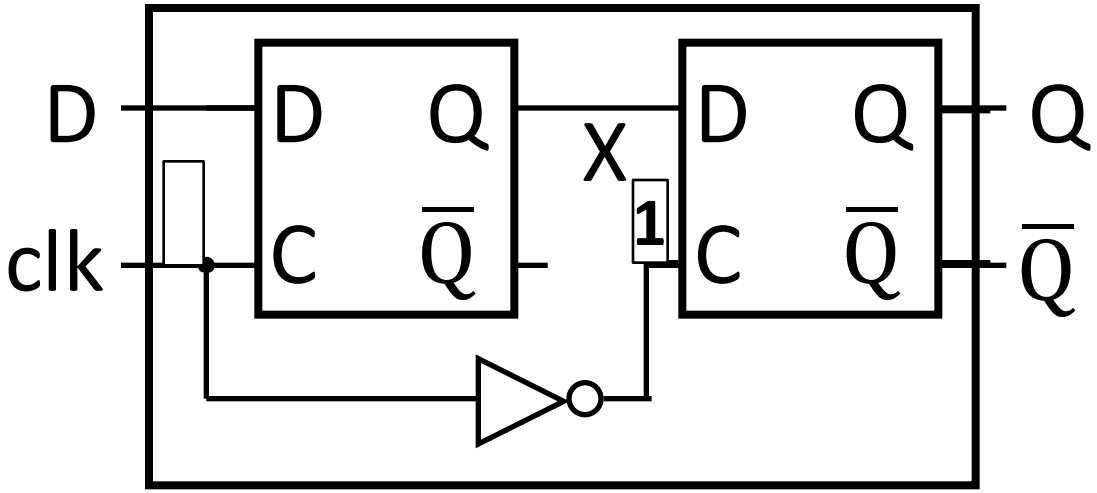
Clock = 0: L1 *opaque*
L2 *transparent*



Thus, on edge of the clock
(when ***CLK* falls from 1→0**)

(*D* passes through to *Q)*

DFF Activity: Fill in the timing graph

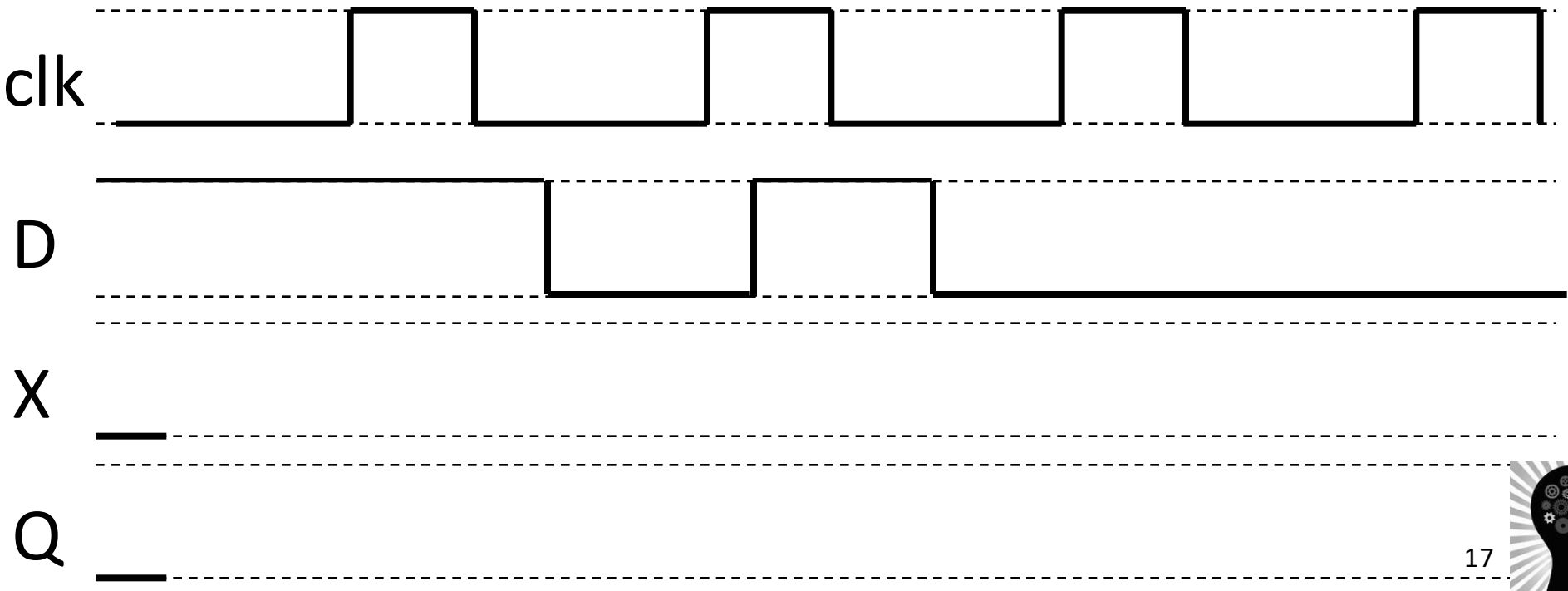


Clock = 1

D passes through L1 to X

Clock = 0

X passes through L2 to Q



Goals for Today

State

- Storing 1 bit
  - Bistable Circuit
  - Set-Reset Latch
  - D Latch
  - D Flip-Flops
- Storing N bits:
  - Registers
  - *More options in later lectures*

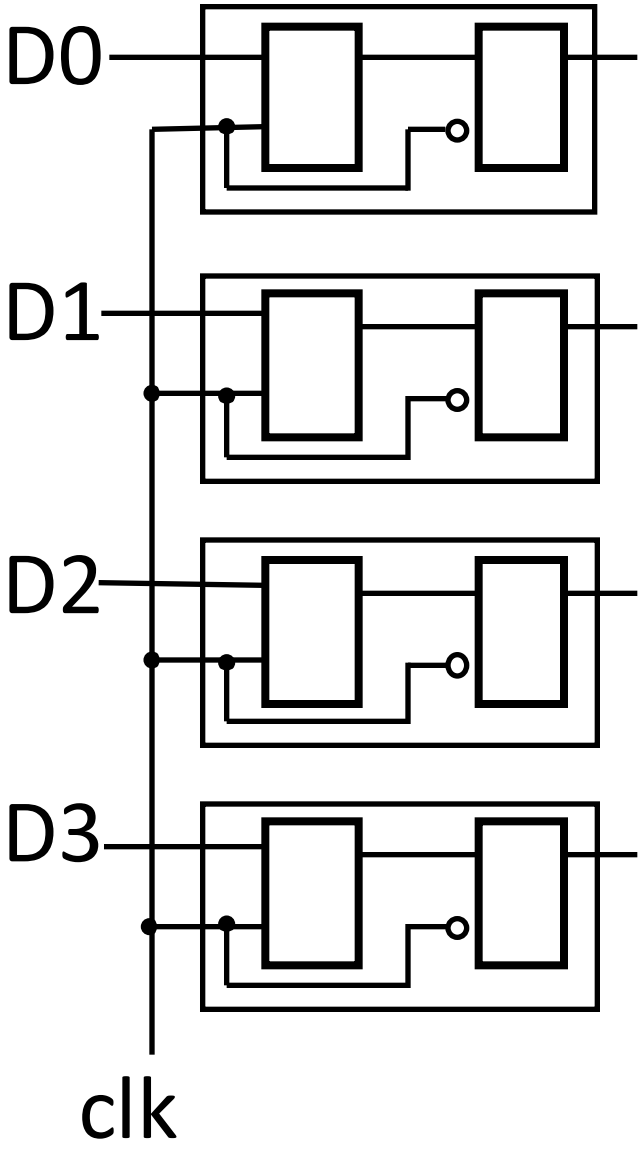
~~A word about clocks~~

A word about terminology

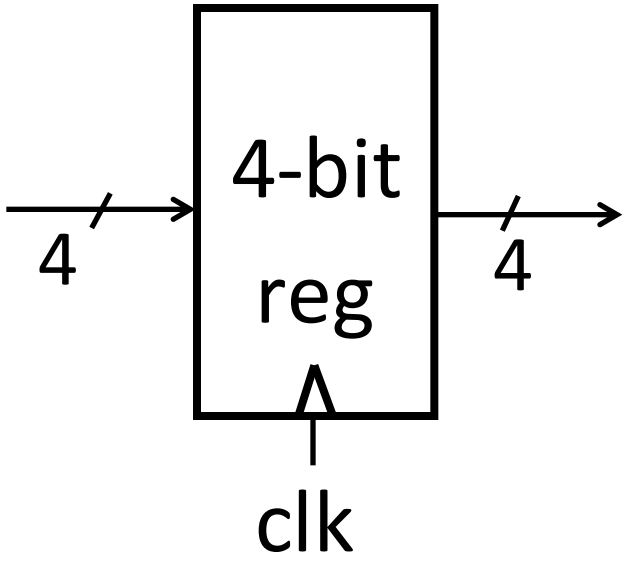
Finite State Machines (FSM)

- Mealy and Moore Machines
- Serial Adder Example

Registers



- D flip-flops in parallel
- shared clock
- Additional (optional) inputs: writeEnable, reset, …



Takeaways

- Set-Reset (SR) Latch can store one bit and we can change the value of the stored bit. But, SR Latch has a forbidden state.
- D Latch can store and change a bit like an SR Latch while avoiding a forbidden state.
- A D Flip-Flip stores one bit. The bit can be changed in a synchronized fashion on the edge of a clock signal.
- An *N*-bit **register** stores *N*-bits. It is be created with *N* D-Flip-Flops in parallel plus a shared clock.

Goals for Today

State

- Storing 1 bit
  - Bistable Circuit
  - Set-Reset Latch
  - D Latch
  - D Flip-Flops
- Storing N bits:
  - Registers
  - *More options in later lectures*

Finite State Machines (FSM)

- Mealy and Moore Machines
- Serial Adder Example

Finite State Machines

An electronic machine which has

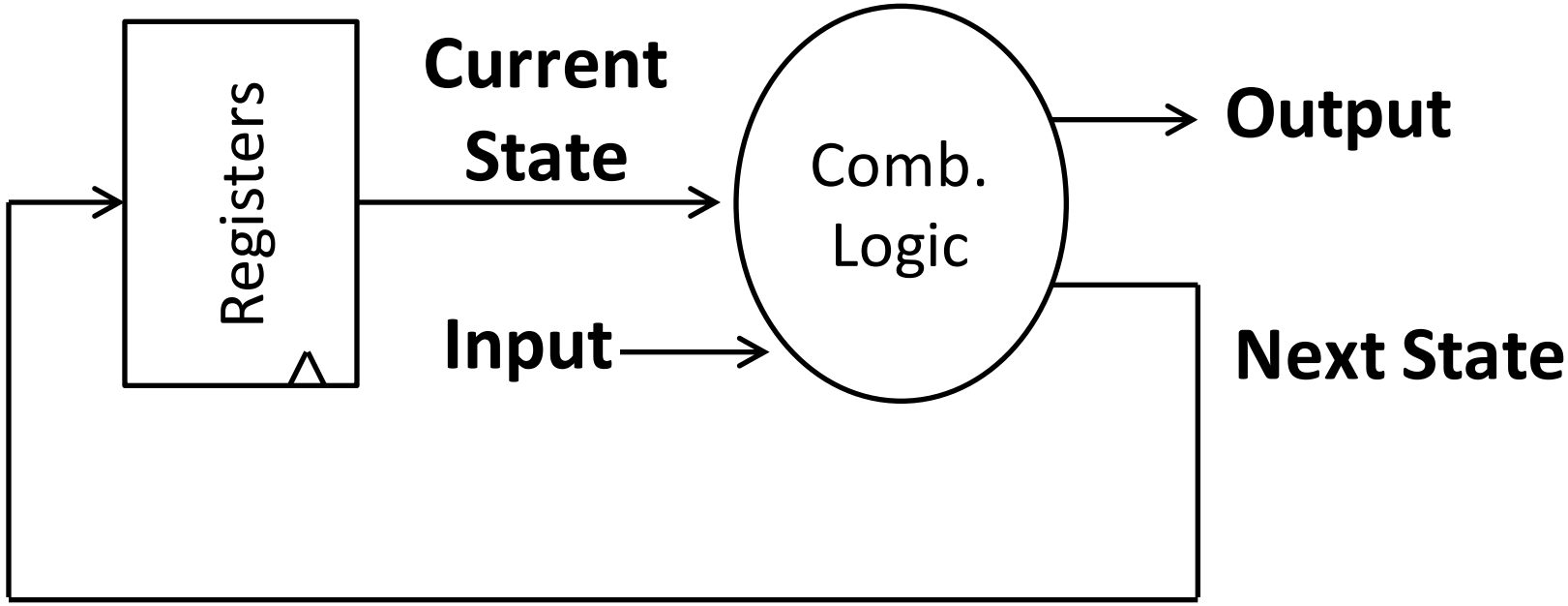
- external inputs
- externally visible outputs
- internal state

Output and next state depend on

- inputs
- current state

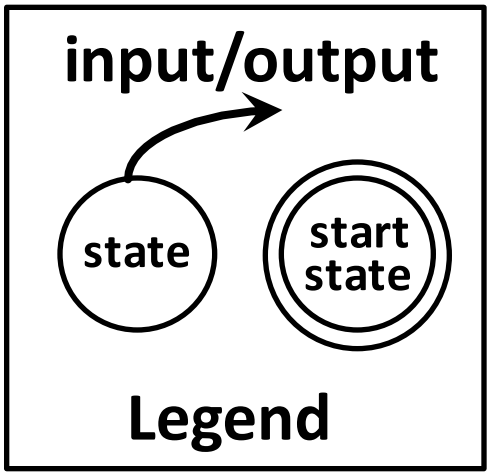
Automata Model

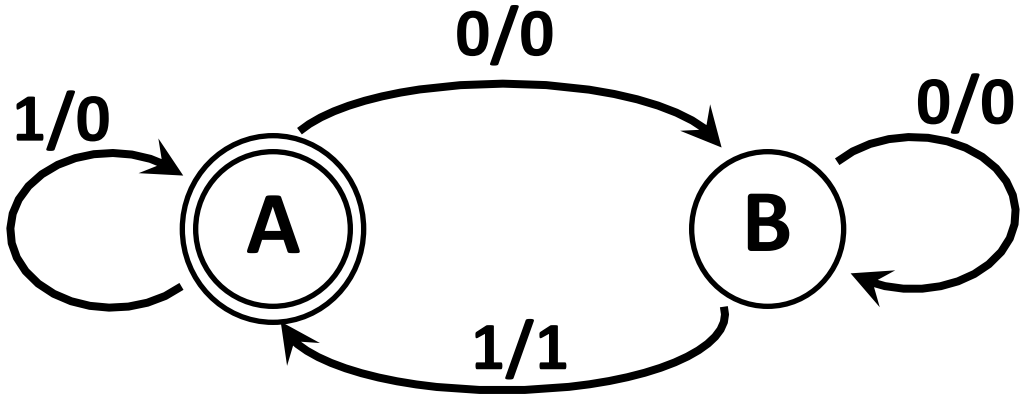
Finite State Machine



- inputs from external world
- outputs to external world
- internal state
- combinational logic

FSM Example





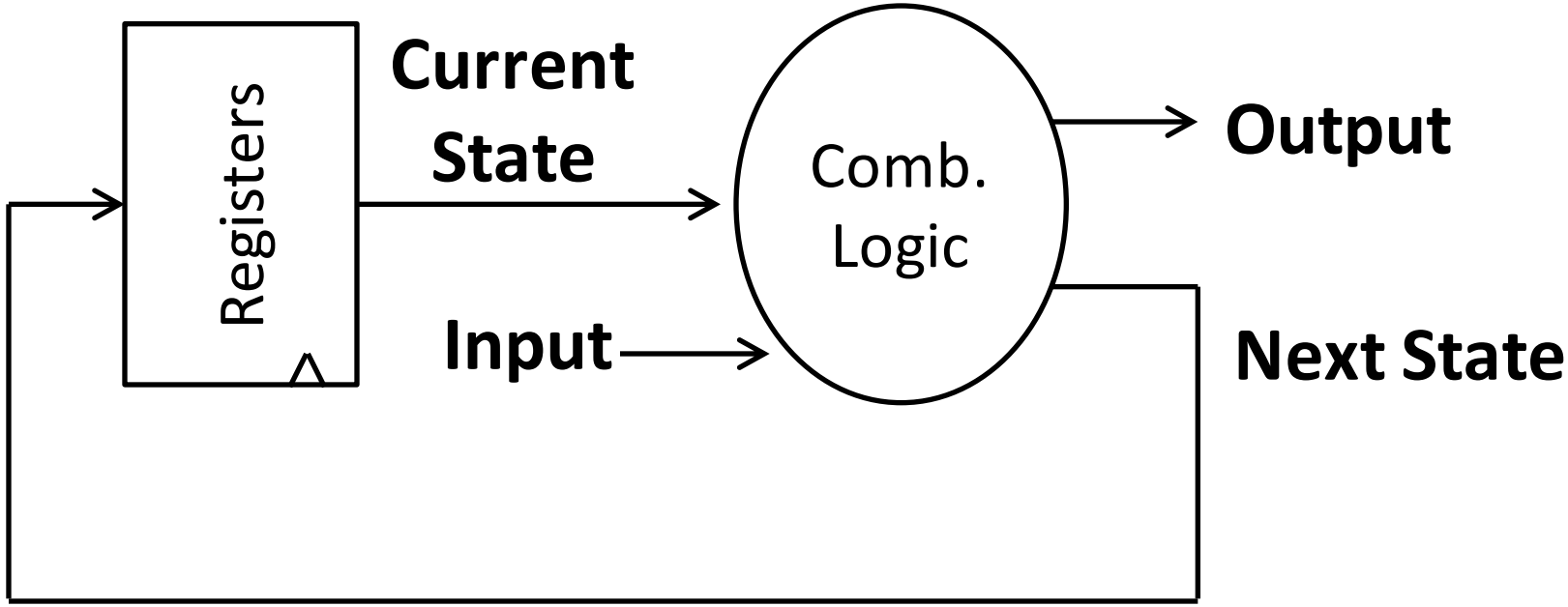
Input: **1** or **0**
Output: **1** or **0**
States: **A** or **B**

What input pattern is the FSM "looking for"?

(Mealy Machine)

Mealy Machine

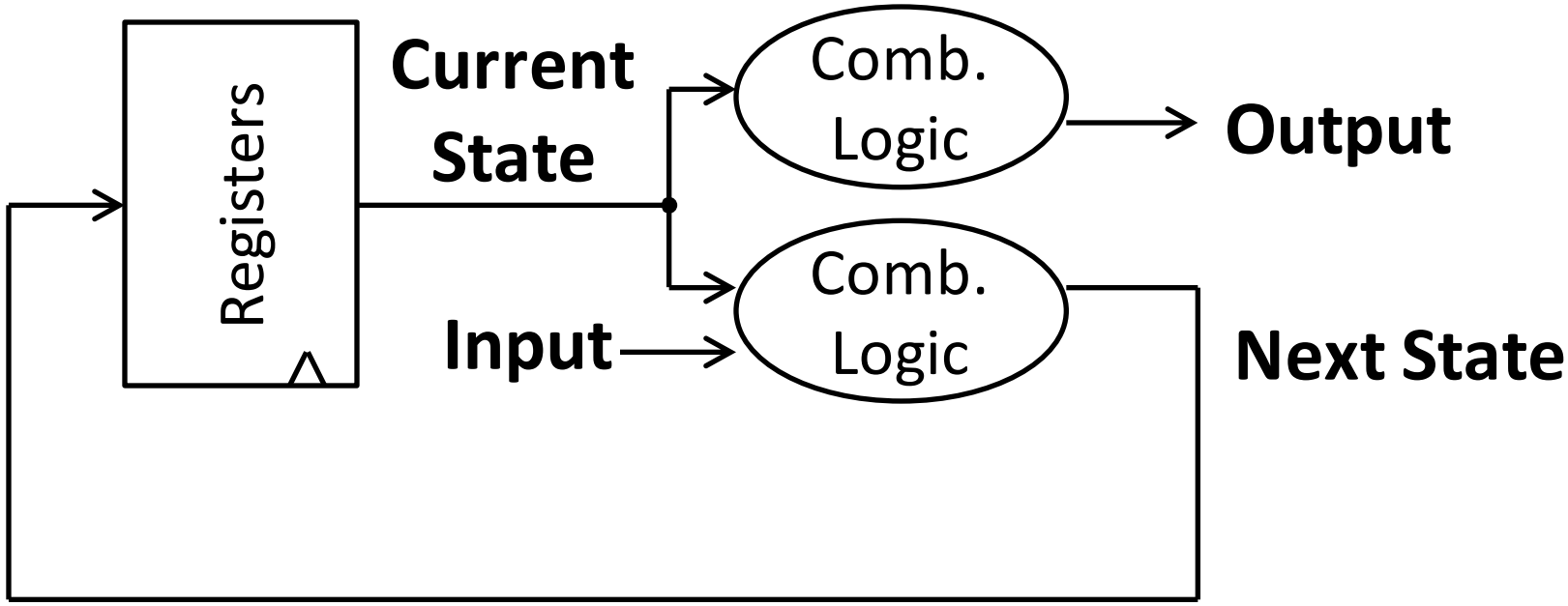
General Case: Mealy Machine



Outputs and next state depend on both current state and input

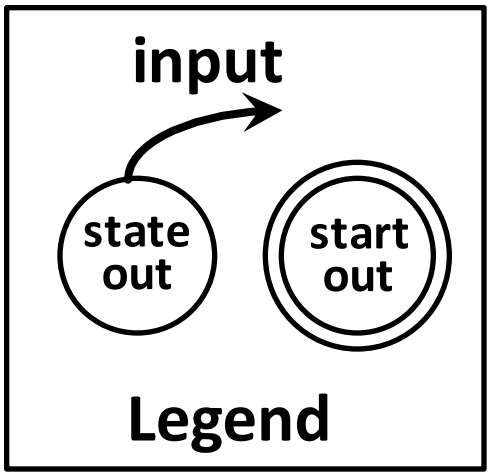
Moore Machine

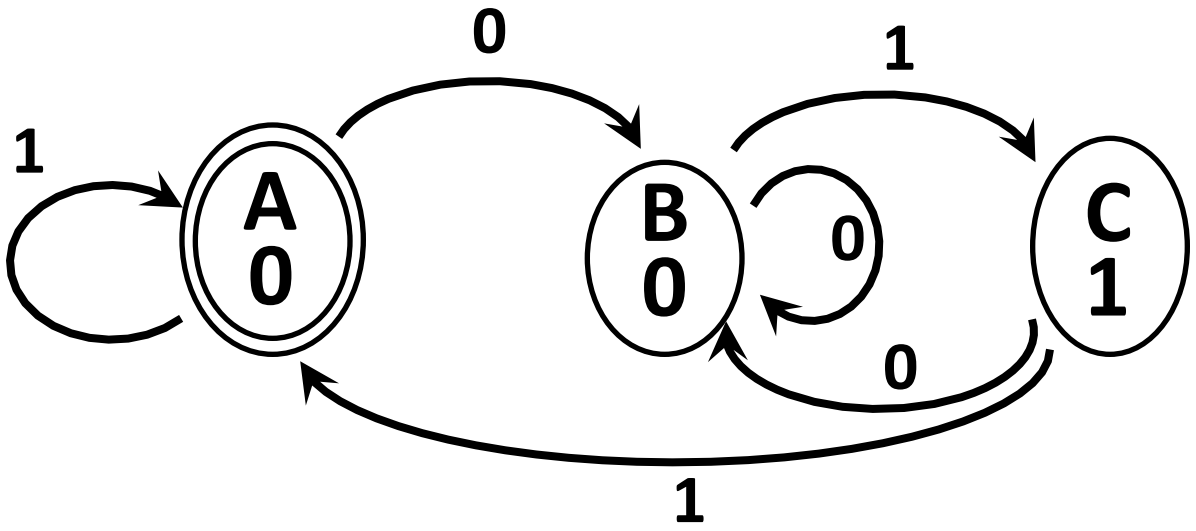
Special Case: Moore Machine



Outputs depend only on current state

Moore Machine FSM Example





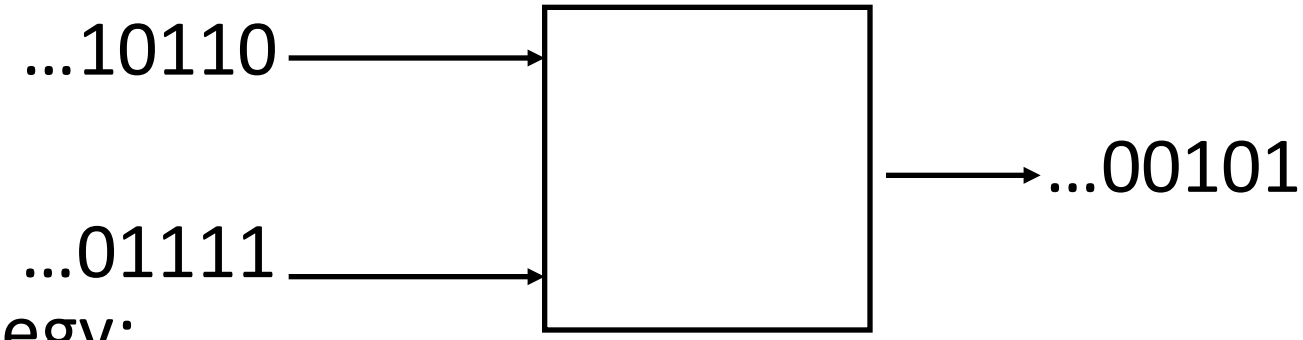
Input: **1** or **0**
Output: **1** or **0**
States: **A**, **B,** or **C**

What input pattern is the FSM "looking for"?

Activity: Build a Logic Circuit for a Serial Adder

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first
- How many states are needed to represent FSM?
- Draw and Fill in FSM diagram



Strategy:

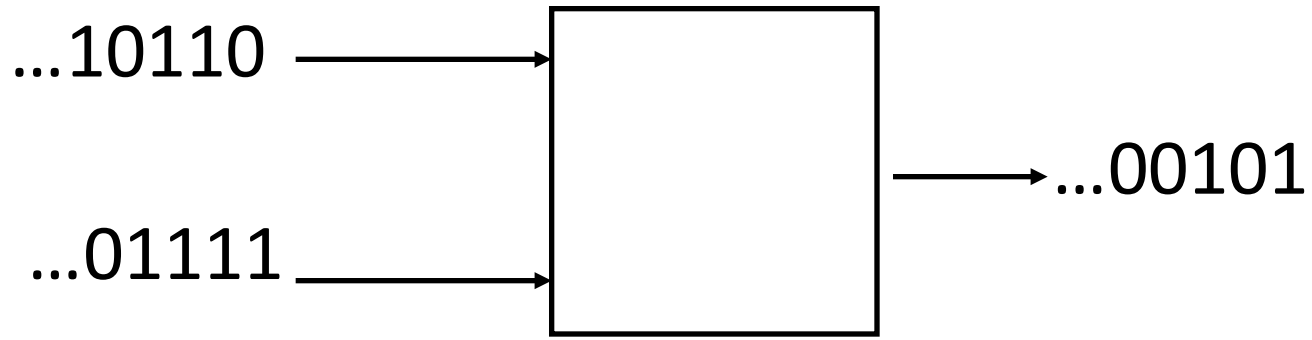
(1) Draw a state diagram (e.g. Mealy Machine)

(2) Write output and next-state tables

(3) Encode states, inputs, and outputs as bits

(4) Determine logic equations for next state and outputs

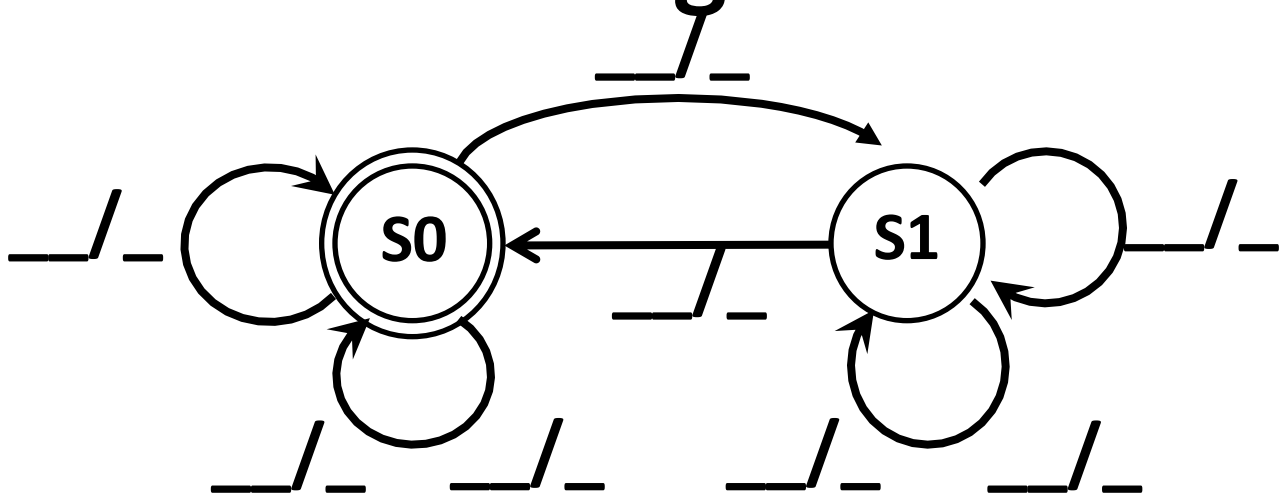
FSM: State Diagram – start here

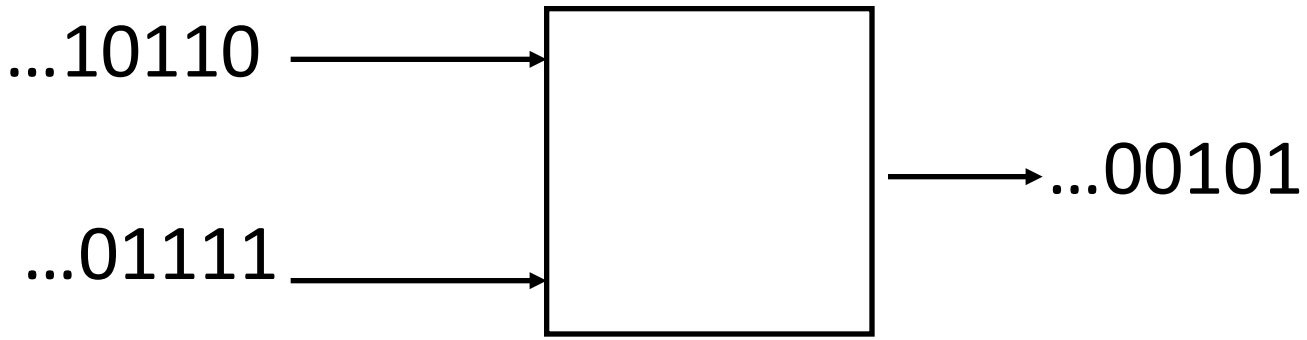


___ states:
Inputs: ??? and ???
Output: ???

- .

FSM: State Diagram

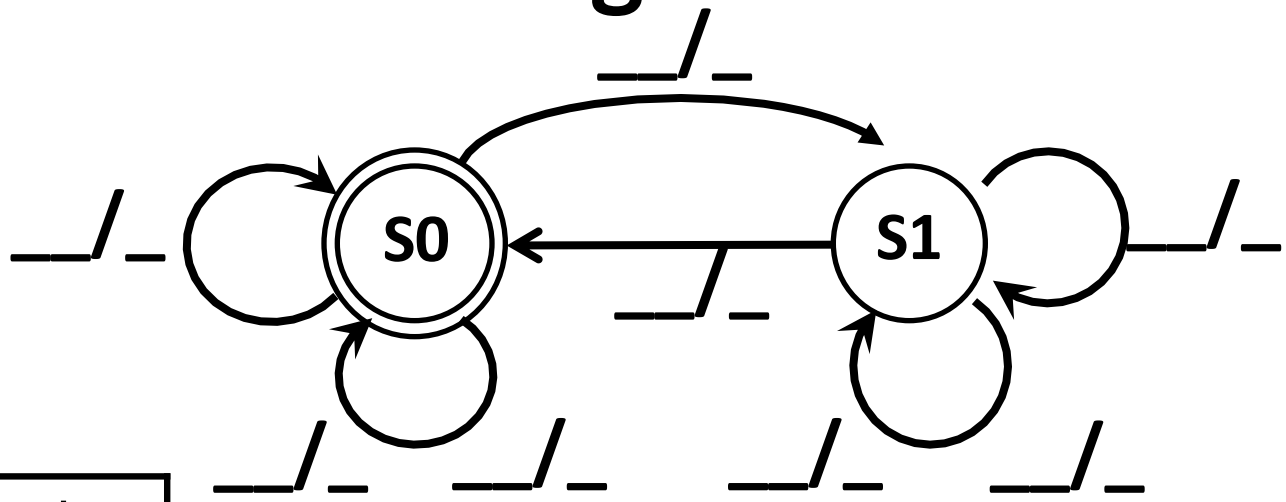




___ states:
Inputs: ??? and ???
Output: ???

- .

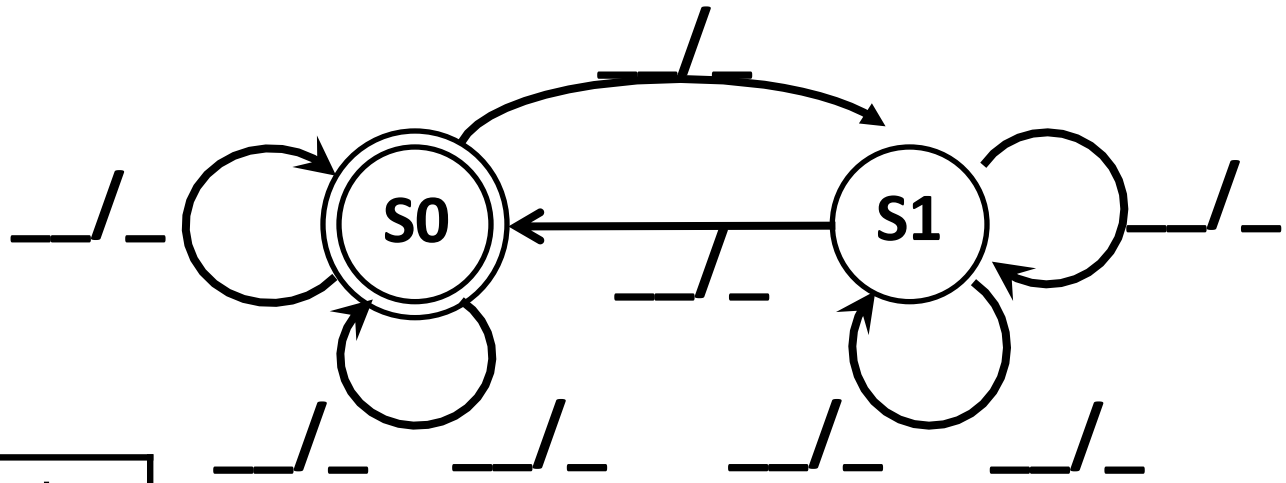
FSM: State Diagram



??	??	Current state	?	Next state

(2) Write down all input and state combinations

FSM: State Diagram – start here



??	??	Current state	?	Next state

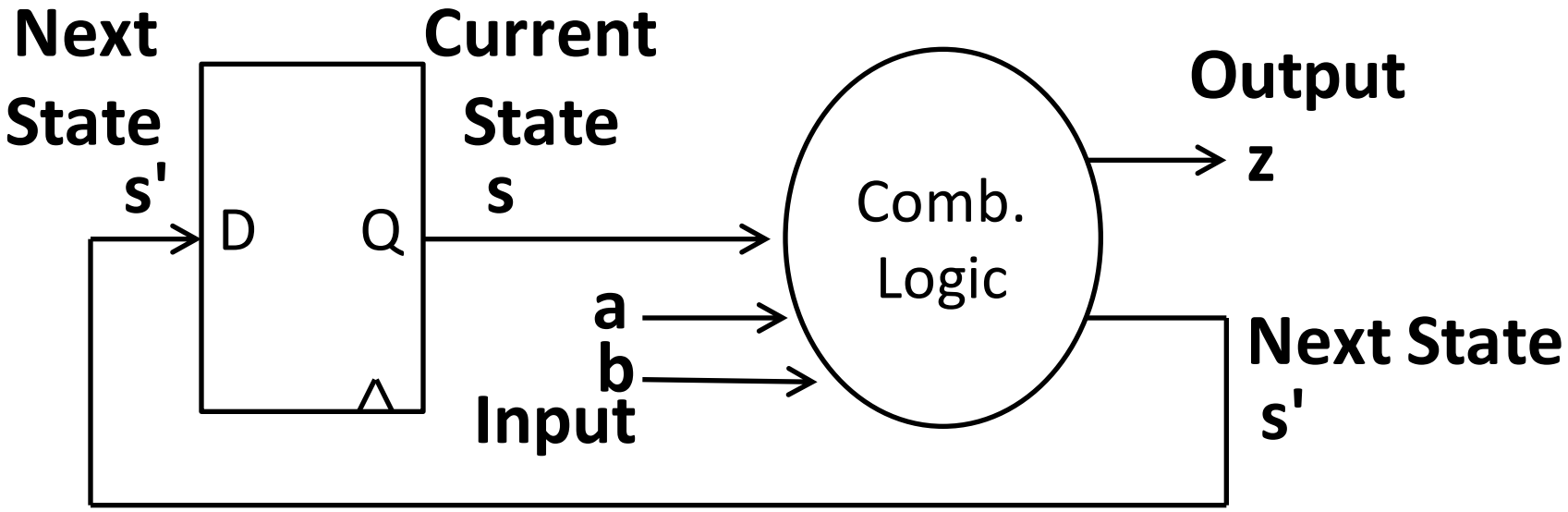
(3) Encode states, inputs, and outputs as bits

FSM: State Diagram

??	??	Current state	?	Next state

(4) Determine logic equations for next state and outputs

Sequential Logic Circuits



$$z = \bar{a}b\bar{s} + a\overline{bs} + \overline{ab}s + abs$$

$$s' = ab\bar{s} + \bar{a}bs + a\bar{b}s + abs$$

Strategy:

(1) Draw a state diagram (e.g. Mealy 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

Summary

We can now store data values

- Stateful circuit elements (D Flip Flops, Registers, …)
- Clock synchronizes state changes
- State Machines or Ad-Hoc Circuits