

State & Finite State Machines

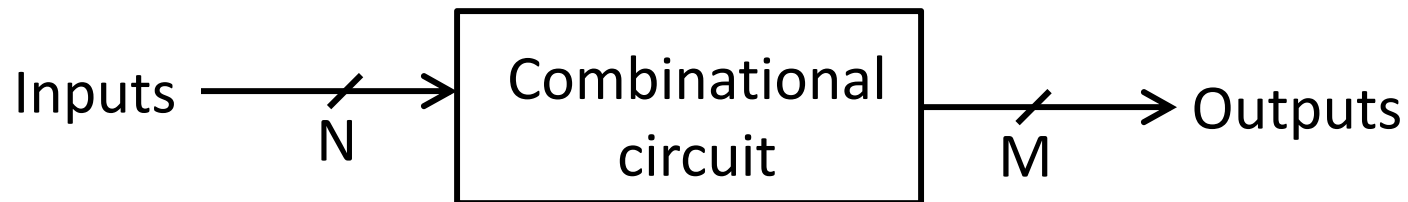
Hakim Weatherspoon
CS 3410, Spring 2012
Computer Science
Cornell University

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

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!



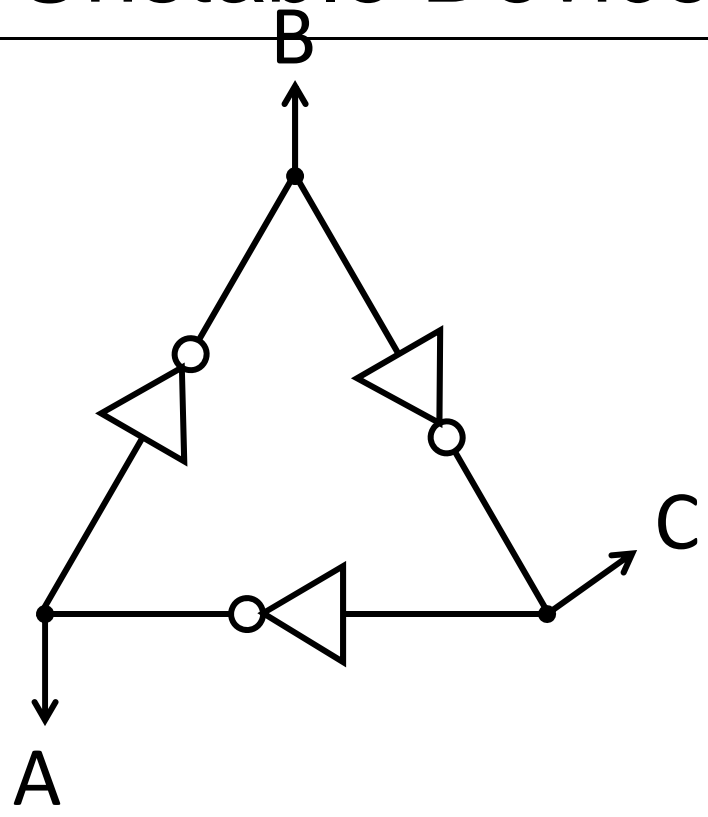
Need a way to record data

Need a way to build stateful circuits

Need a state-holding device

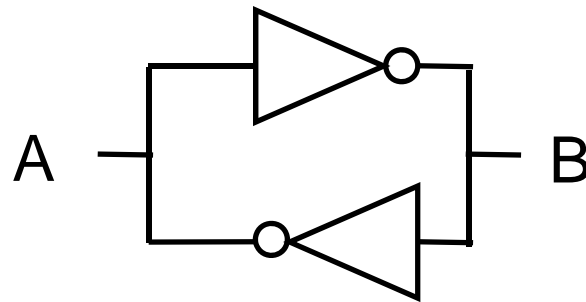
Finite State Machines

Unstable Devices



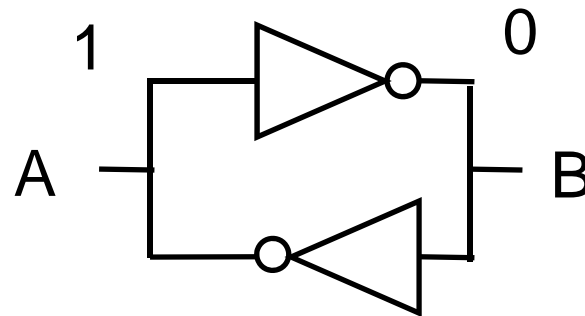
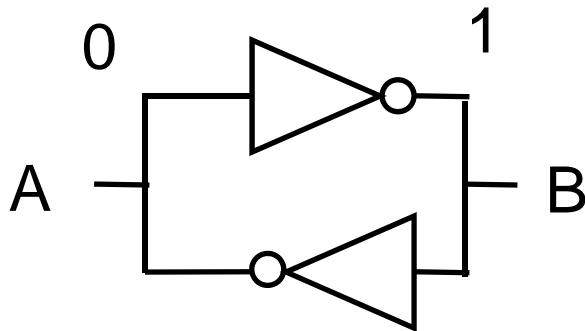
Bistable Devices

- Stable and unstable equilibria?



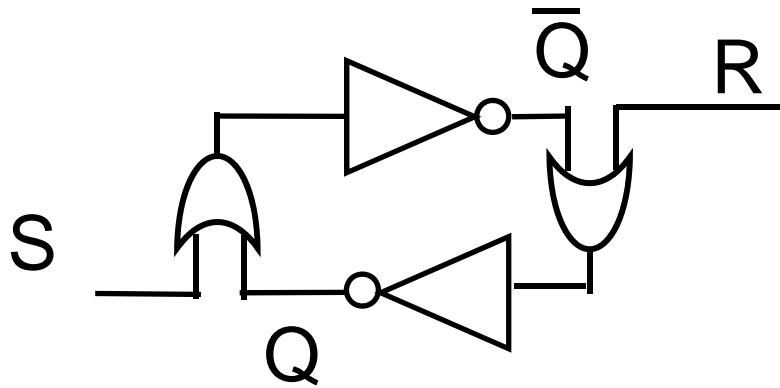
A Simple Device

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



- How do we change the state?

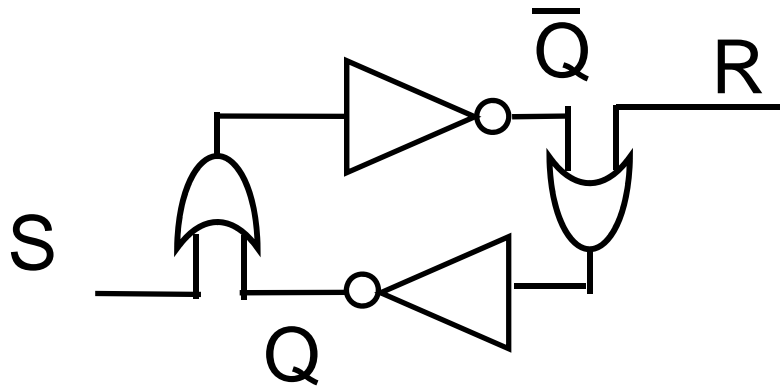
SR Latch



S	R	Q	\bar{Q}
0	0		
0	1		
1	0		
1	1		

- Set-Reset (S-R) Latch
- Stores a value Q and its complement

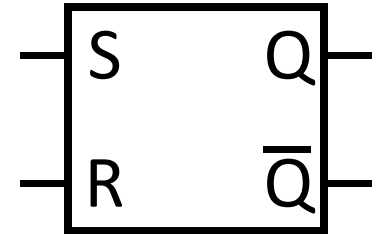
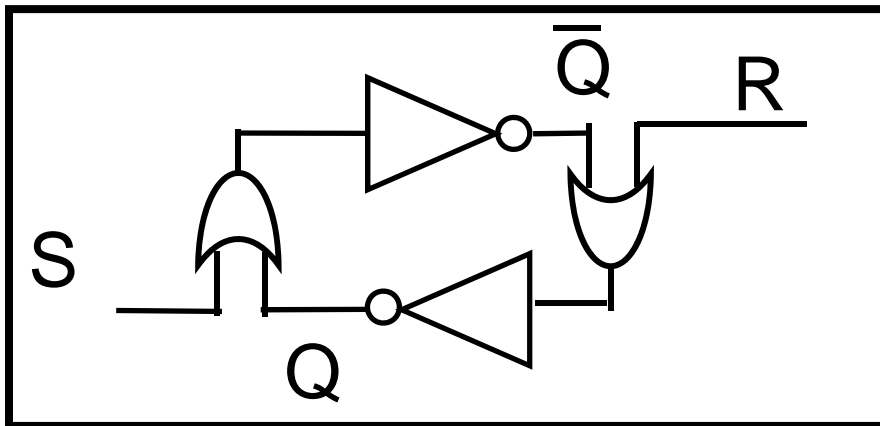
SR Latch



S	R	Q	\bar{Q}
0	0		
0	1		
1	0		
1	1		

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

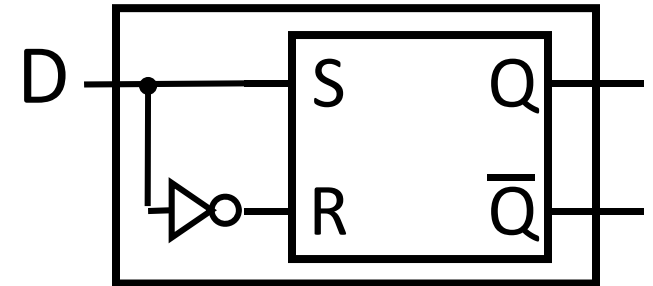
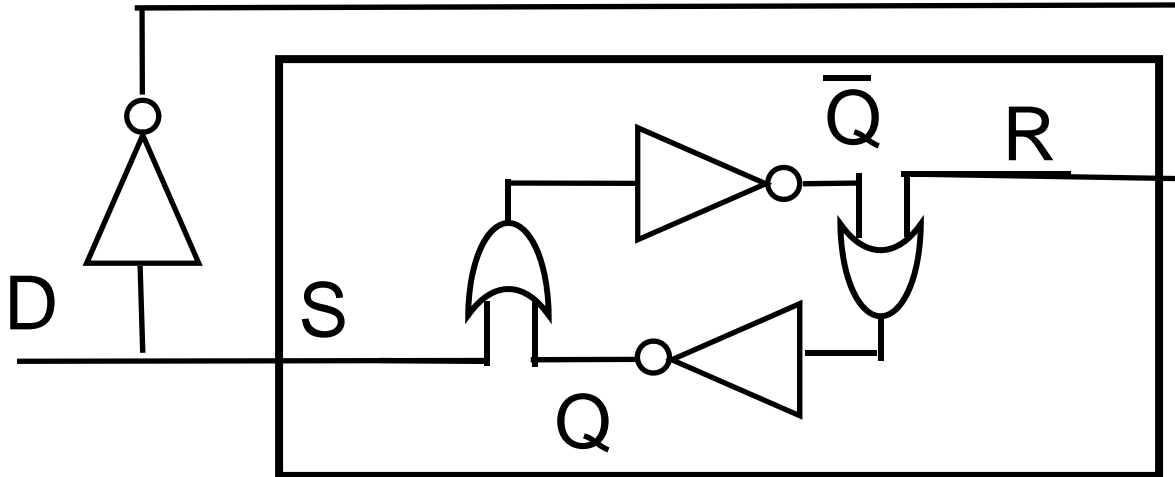
SR Latch



S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	0	1
1	0	1	0
1	1	?	?

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

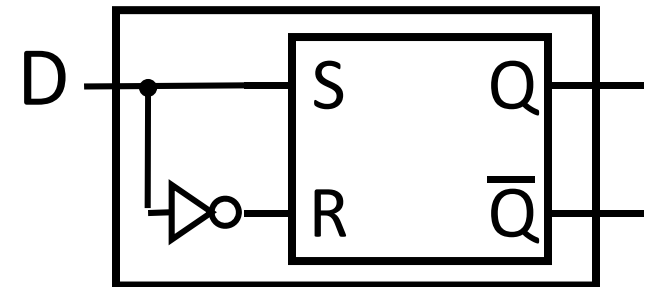
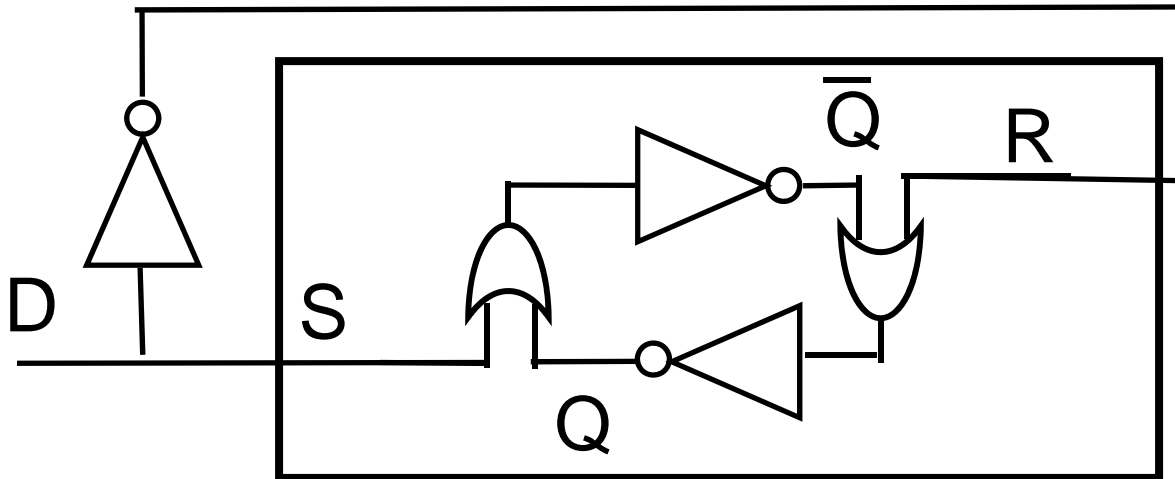
(Unclocked) D Latch



D	Q	\bar{Q}
0		
1		

- Data (D) 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

(Unclocked) D Latch

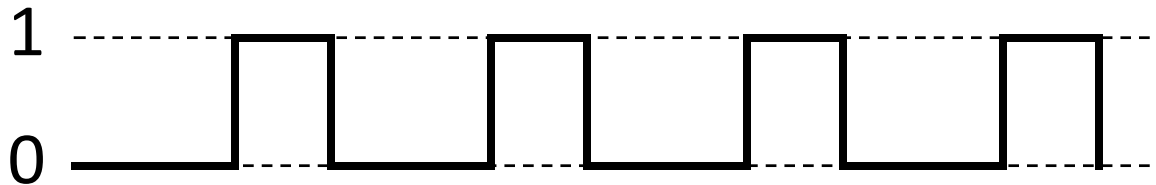


D	Q	\bar{Q}
0	0	1
1	1	0

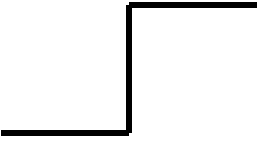
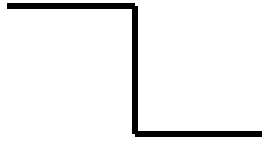
- Data (D) Latch
 - Easier to use than an SR latch
 - No possibility of entering an undefined state
- When D changes, Q changes
 - ... immediately (...after a delay of 2 Ors and 2 NOTs)
- Need to control when the output changes

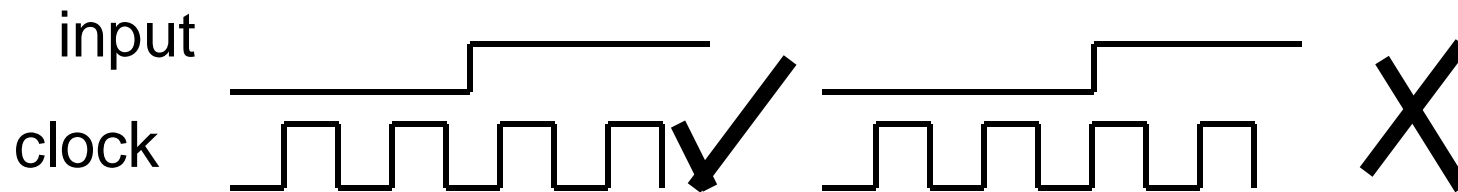
Clocks

- Clock helps coordinate state changes
 - Usually generated by an oscillating crystal
 - Fixed period; frequency = $1/\text{period}$



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



Clock Disciplines

- Level sensitive
 - State changes when clock is high (or low)
- Edge triggered
 - State changes at clock edge

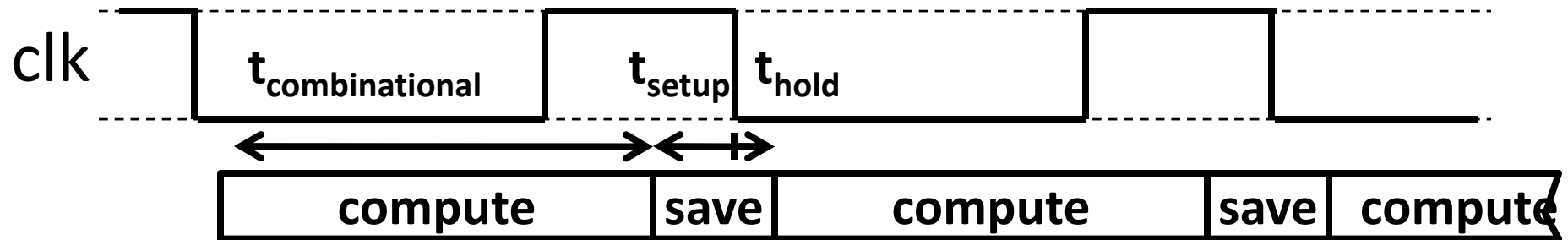
positive edge-triggered 

negative edge-triggered 

Clock Methodology

Clock Methodology

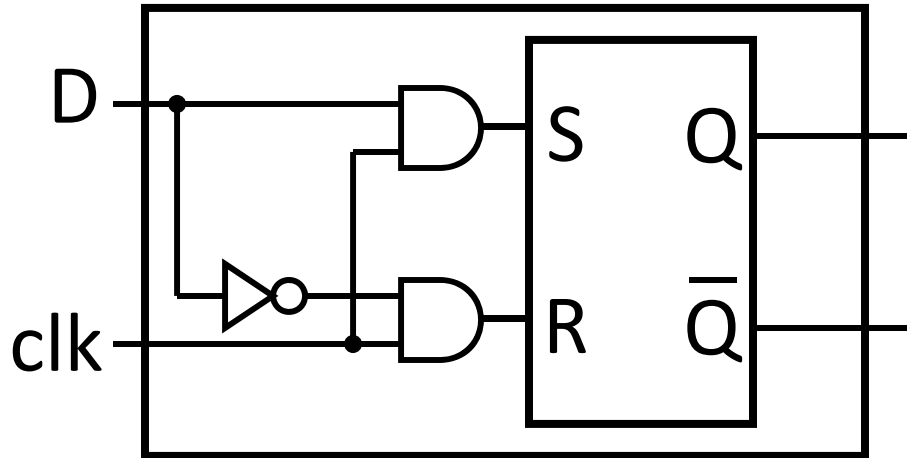
- Negative edge, synchronous



– Signals must be stable near falling clock edge

- Positive edge synchronous
- Asynchronous, multiple clocks, . . .

D Latch with Clock



S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	0	1
1	0	1	0
1	1	forbidden	

clk	D	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	Q	\bar{Q}
1	0	0	1
1	1	1	0

D Latch with Clock

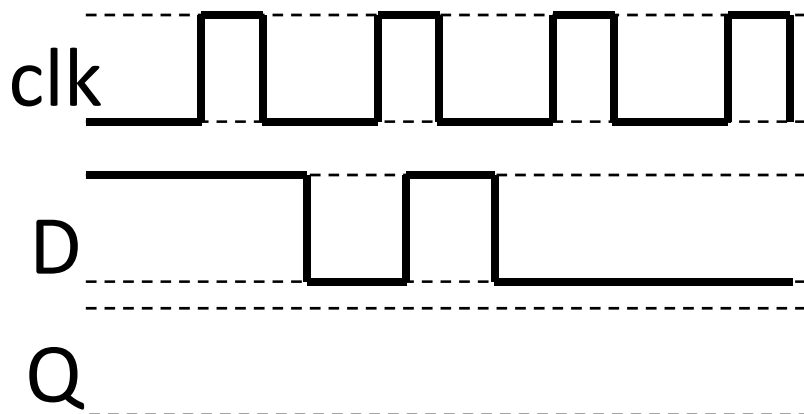
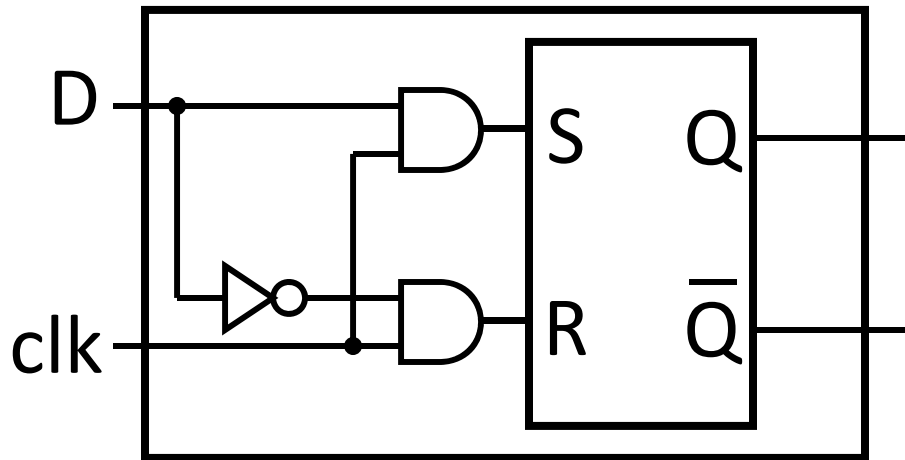
Level Sensitive D Latch

Clock high:

set/reset (according to D)

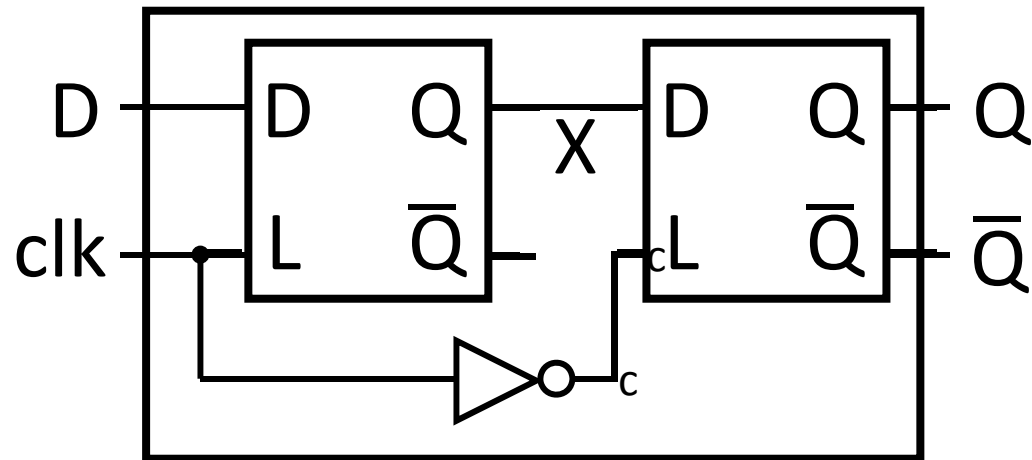
Clock low:

keep state (ignore D)



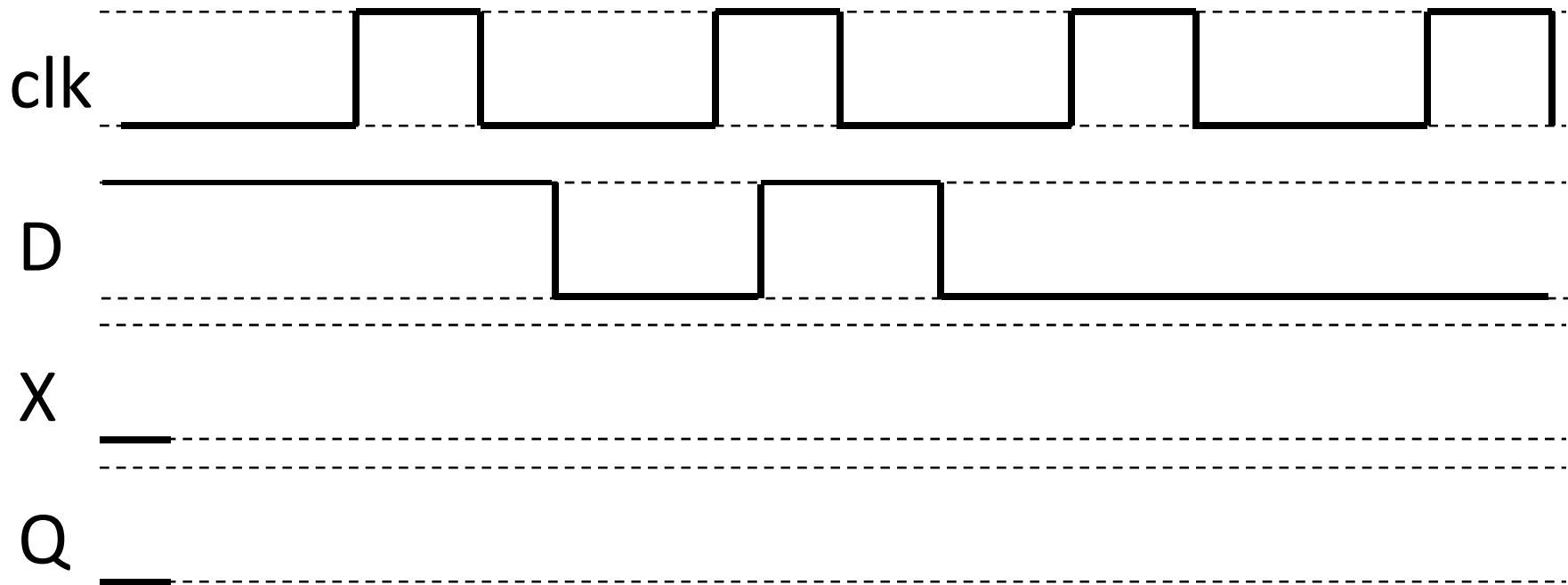
clk	D	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	Q	\bar{Q}
1	0	0	1
1	1	1	0

Edge-Triggered D Flip-Flop

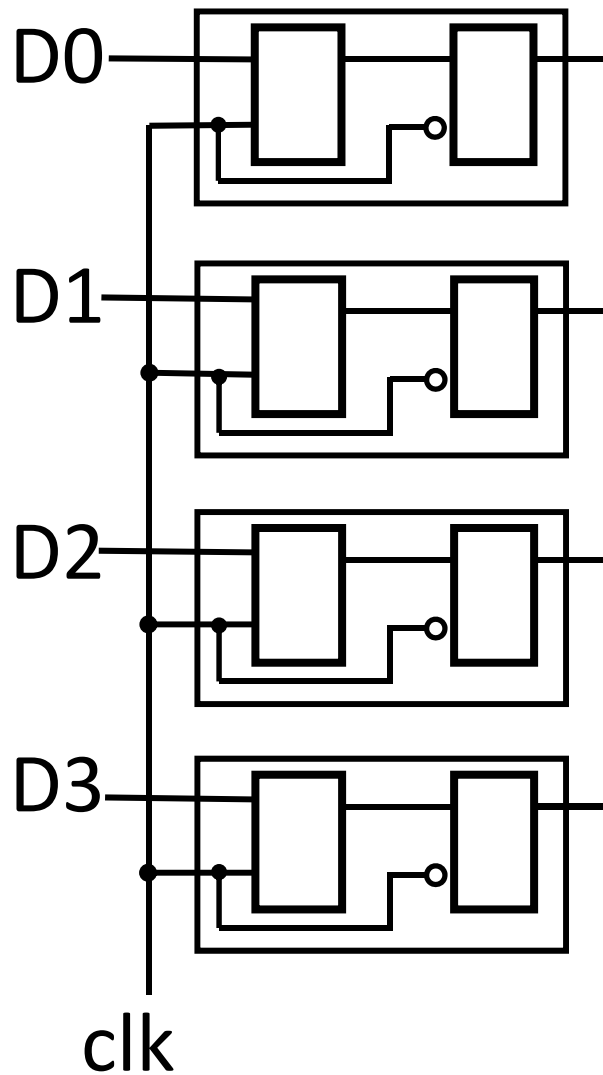


D Flip-Flop

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

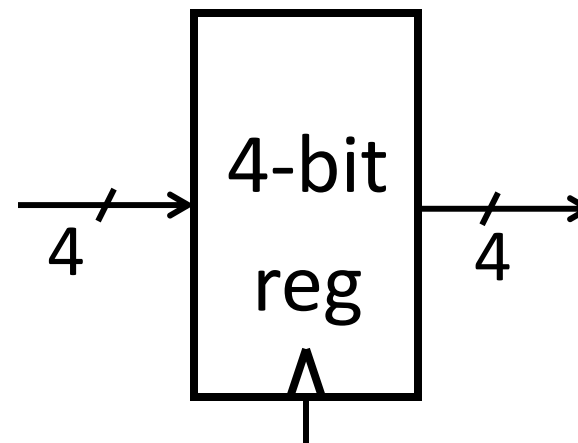


Registers

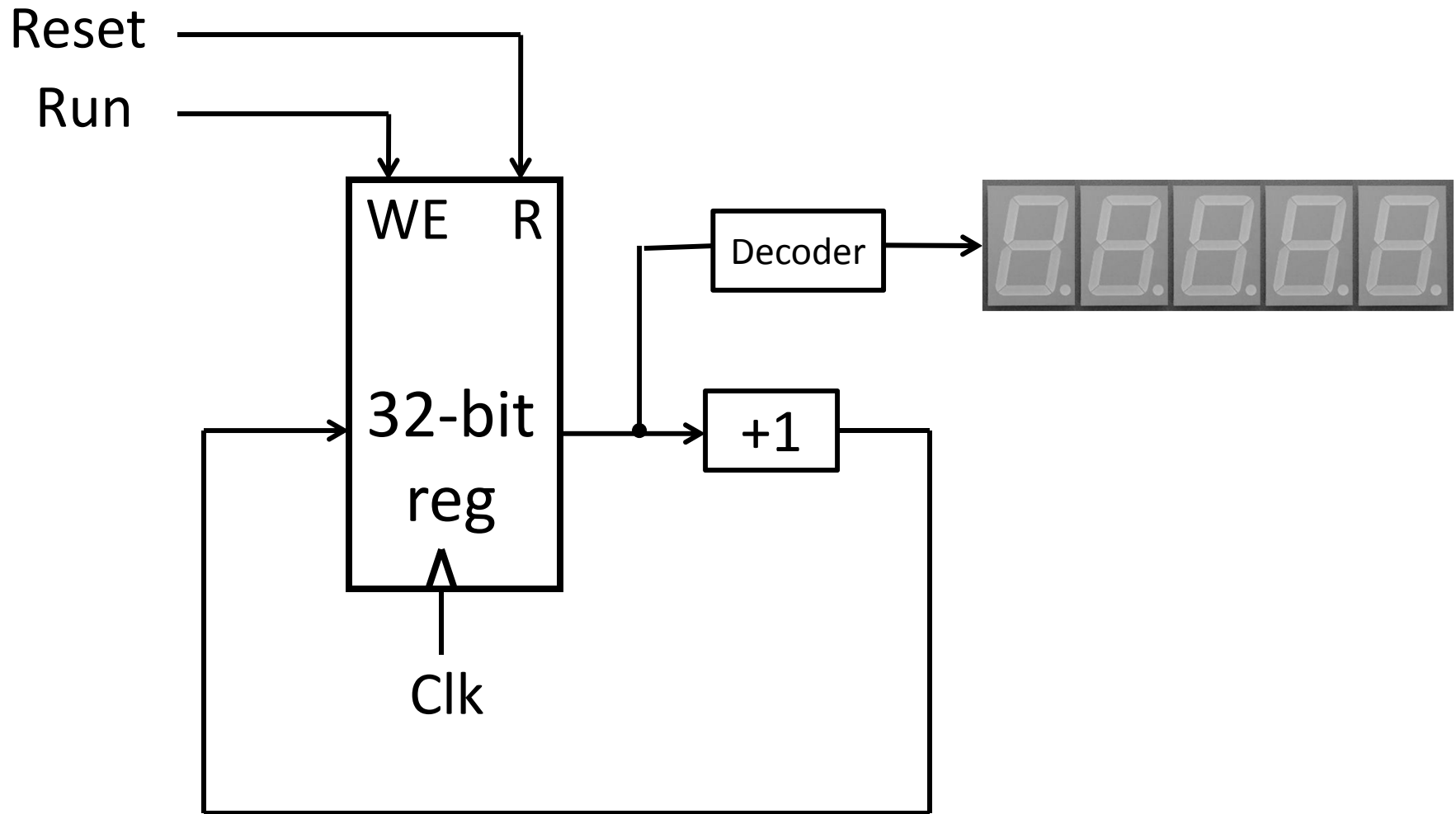


Register

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



An Example: What will this circuit do?



Recap

We can now build interesting devices with sensors

- Using combinatorial logic

We can also store data values

- In state-holding elements
- Coupled with clocks

Administrivia

Make sure partner in same Lab Section ***this week***

Lab2 is out

Due in one week, next Monday, start early

Work alone

But, use your resources

- Lab Section, Piazza.com, Office Hours, Homework Help Session,
- Class notes, book, Sections, CSUGLab

No Homework this week

Administrivia

Check online syllabus/schedule

- <http://www.cs.cornell.edu/Courses/CS3410/2012sp/schedule.html>

Slides and Reading for lectures

Office Hours

Homework and Programming Assignments

Prelims (in evenings):

- Tuesday, February 28th
- Thursday, March 29th
- Thursday, April 26th

Schedule is subject to change

Collaboration, Late, Re-grading Policies

“Black Board” Collaboration Policy

- Can discuss approach together on a “black board”
- Leave and write up solution independently
- Do not copy solutions

Late Policy

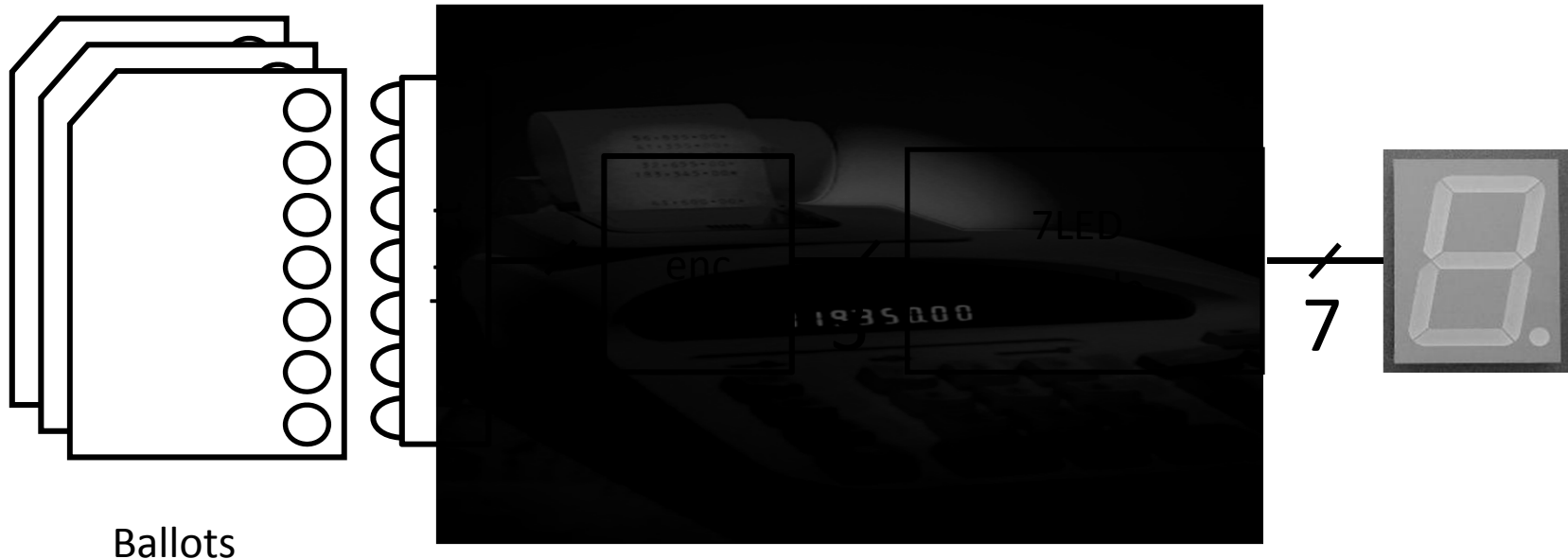
- Each person has a total of **four** “slip days”
- Max of **two** slip days for any individual assignment
- Slip days deducted first for *any* late assignment, cannot selectively apply slip days
- For projects, slip days are deducted from all partners
- 20% deducted per day late after slip days are exhausted

Regrade policy

- Submit written request to lead TA,
and lead TA will pick a different grader
- Submit another written request,
lead TA will regrade directly
- Submit yet another written request for professor to regrade.

Finite State Machines

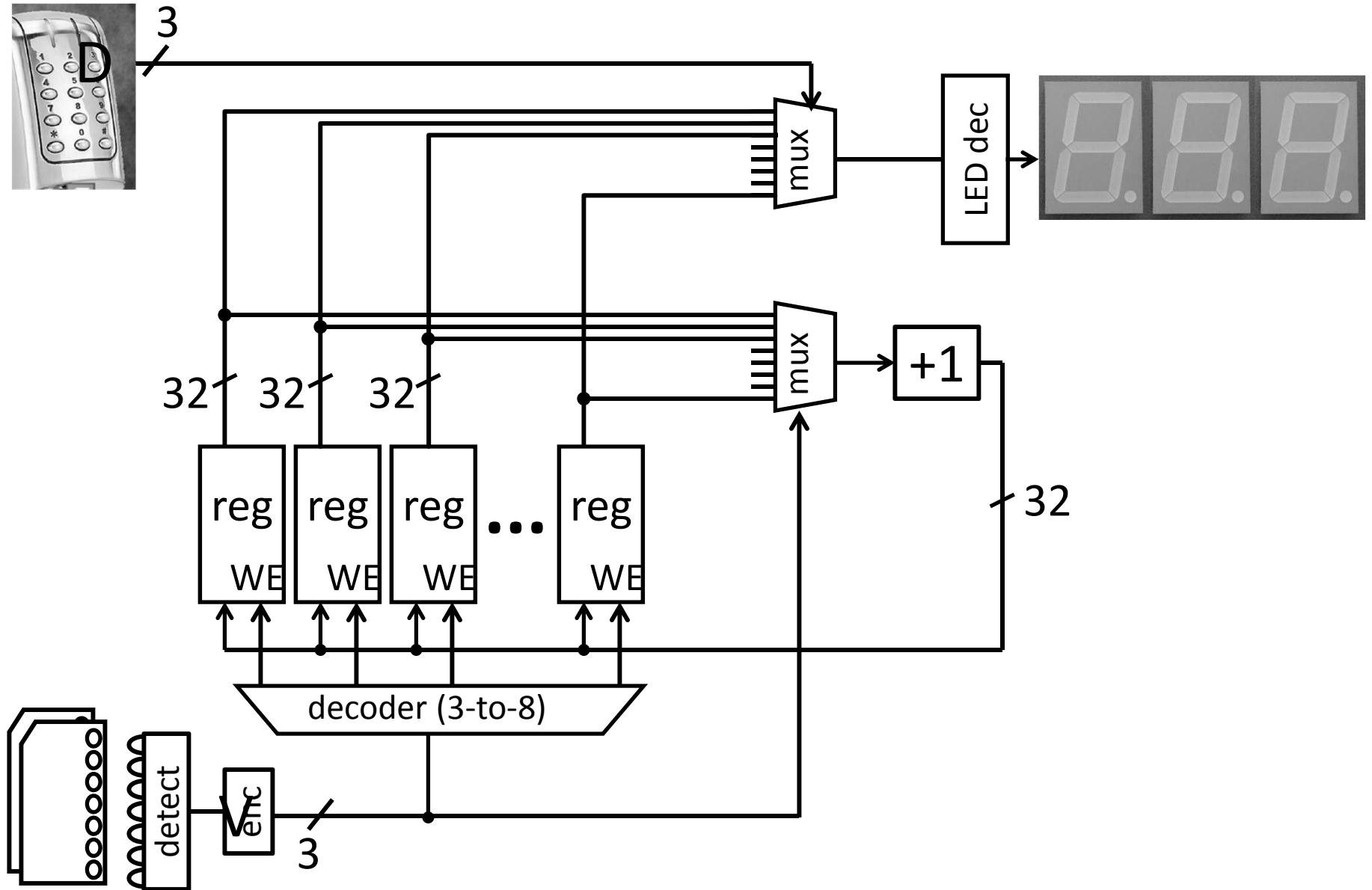
Revisit Voting Machine



Ballots

How do we create
a vote counter
machine

Revisit Voting Machine



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

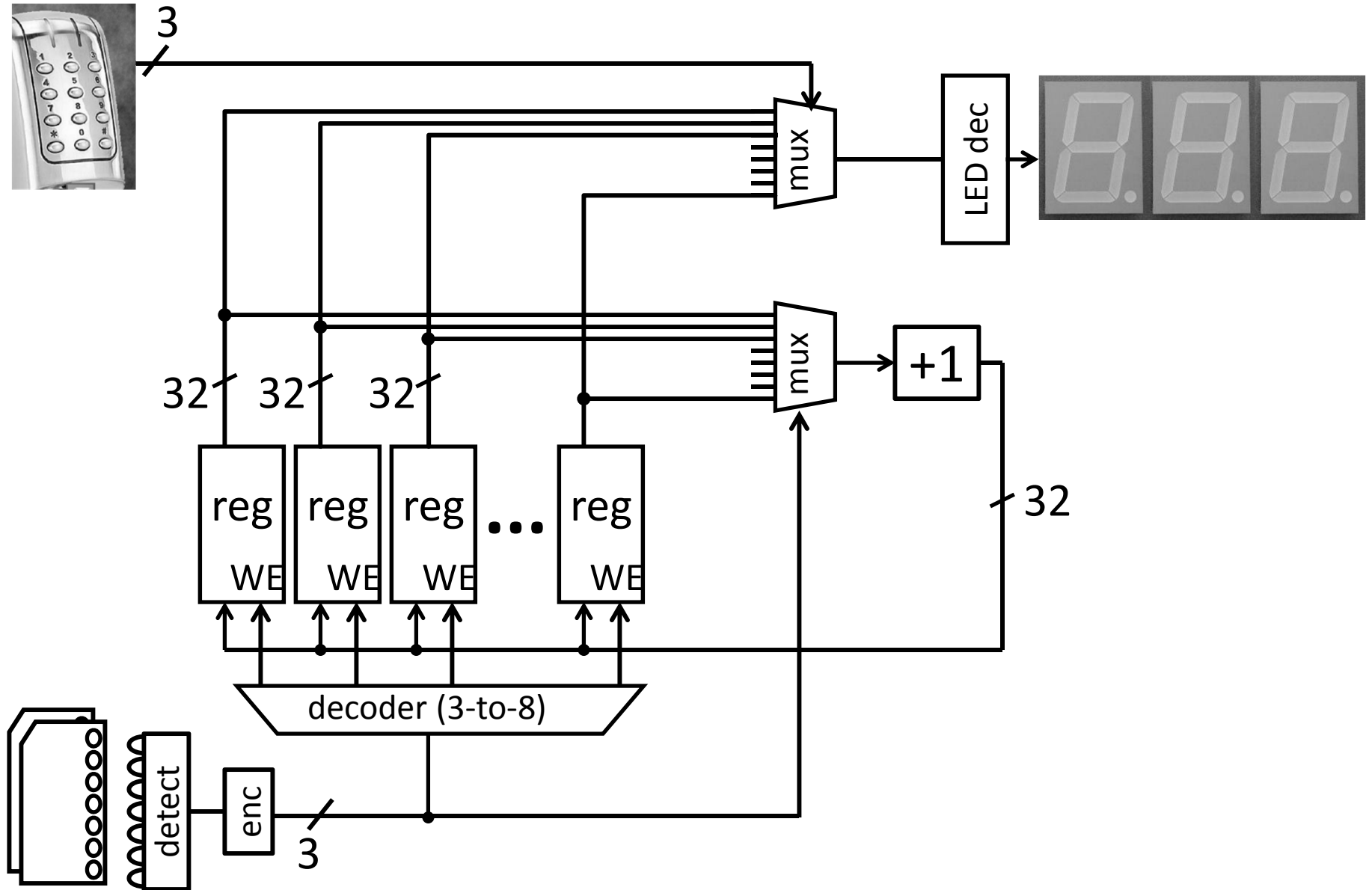
I : Finite set of inputs

O : Finite set of outputs

δ : State transition function

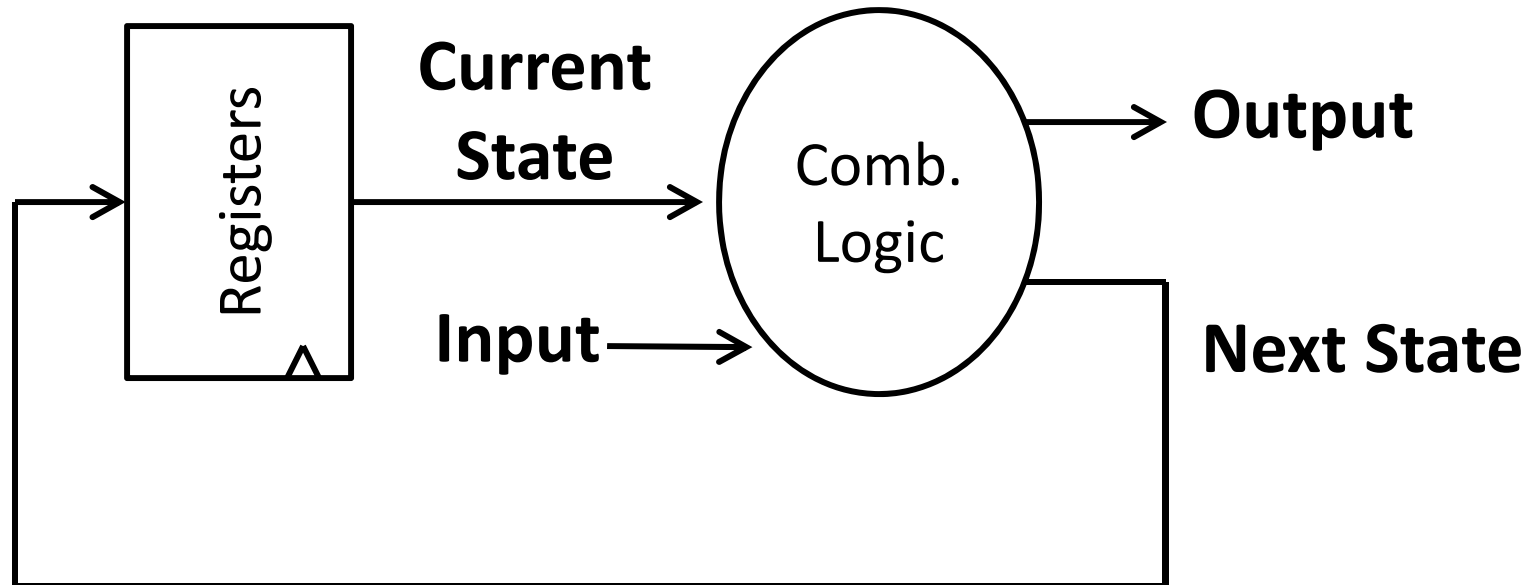
Next state depends on present input *and*
present state

Revisit Voting Machine



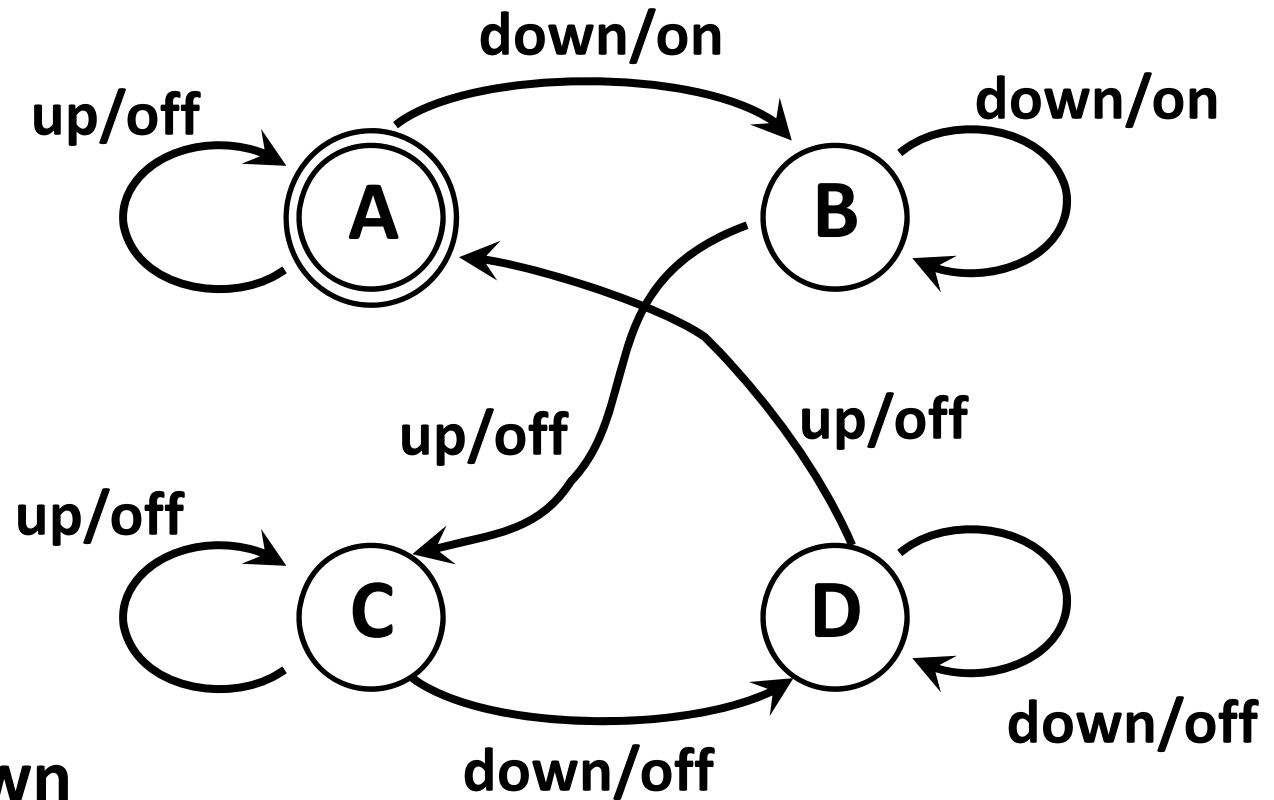
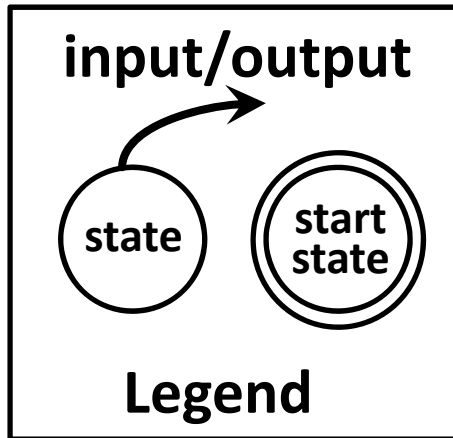
Automata Model

Finite State Machine



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

FSM Example

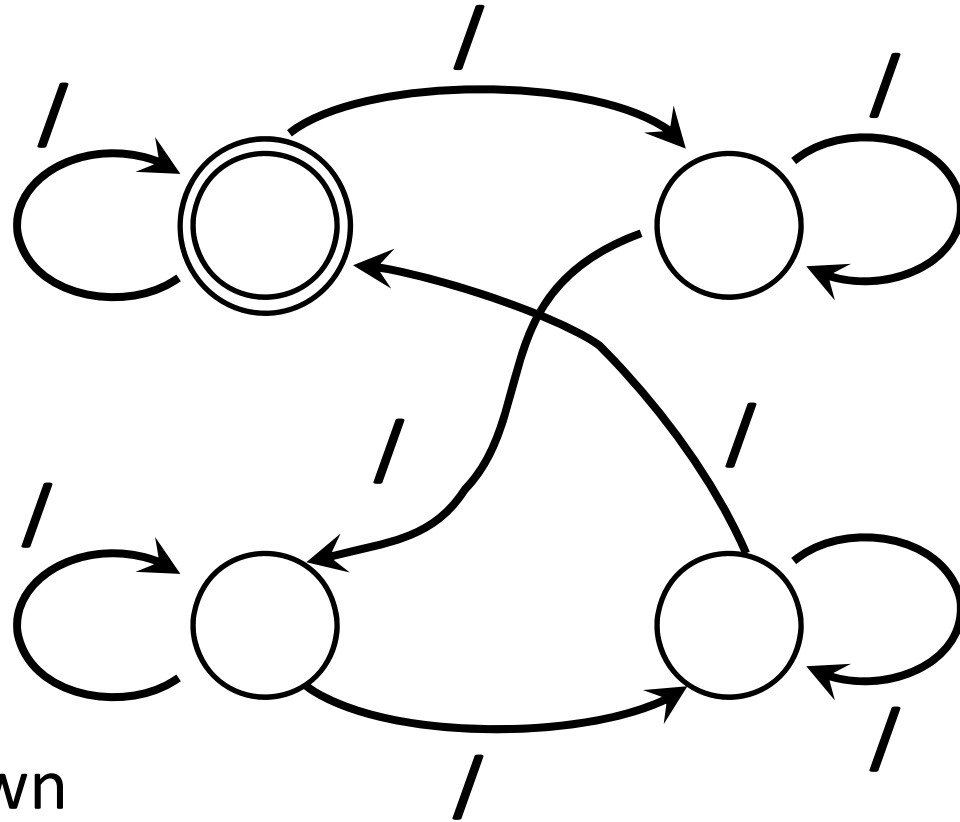
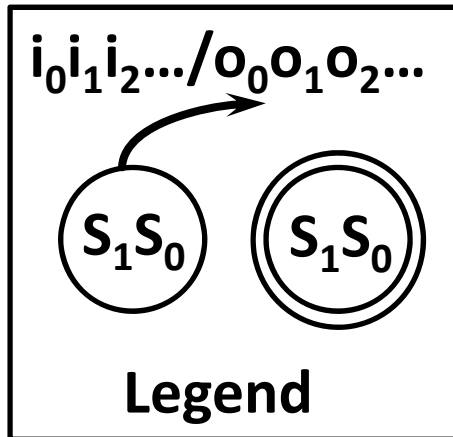


Input: **up** or **down**

Output: **on** or **off**

States: **A**, **B**, **C**, or **D**

FSM Example

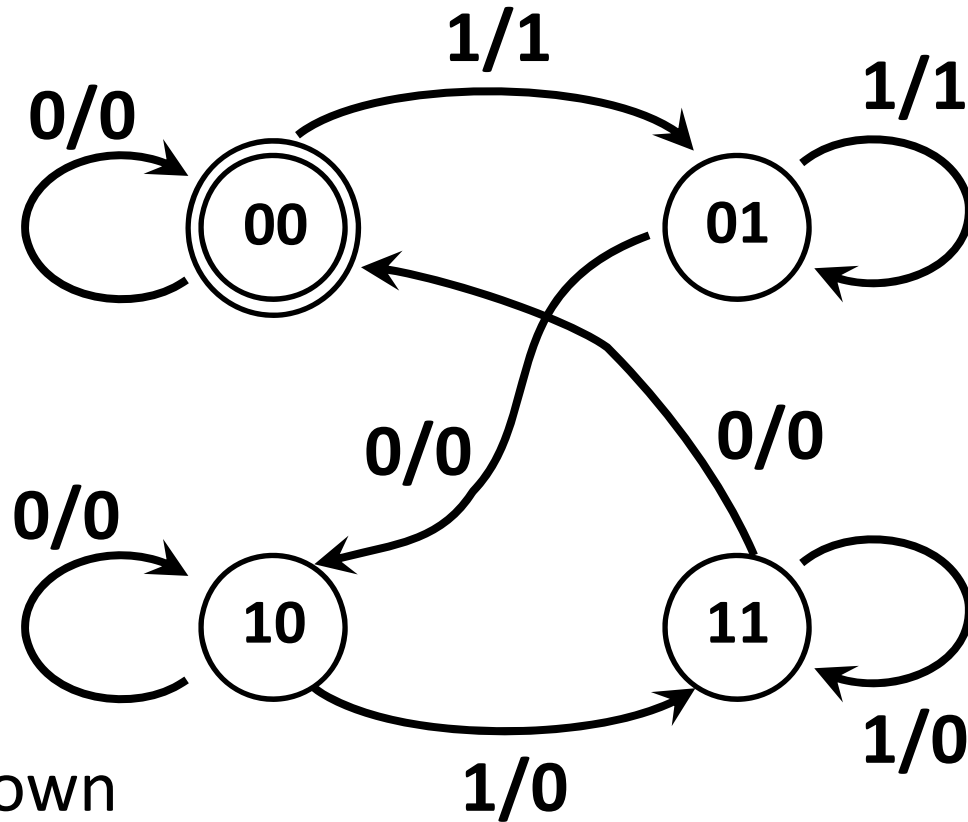
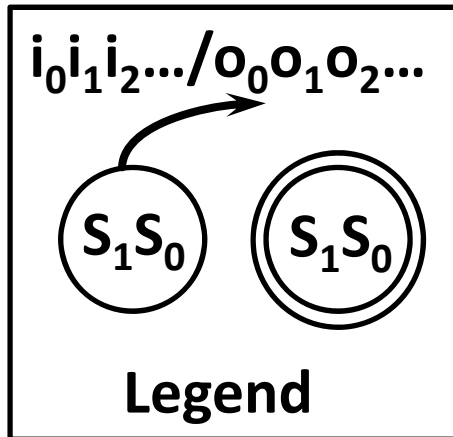


Input: =up or =down

Output: =on or =off

States: =A, =B, =C, or =D

FSM Example



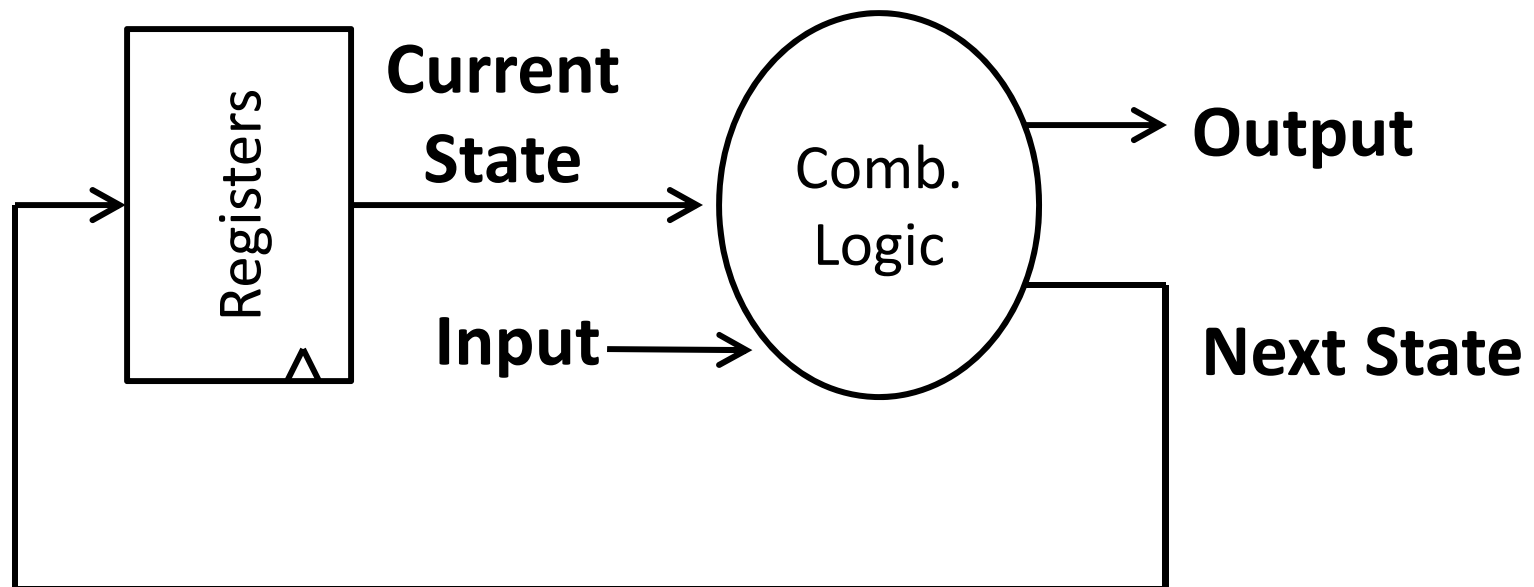
Input: **0**=up or **1**=down

Output: **1**=on or **1**=off

States: **00**=A, **01**=B, **10**=C, or **11**=D

Mealy Machine

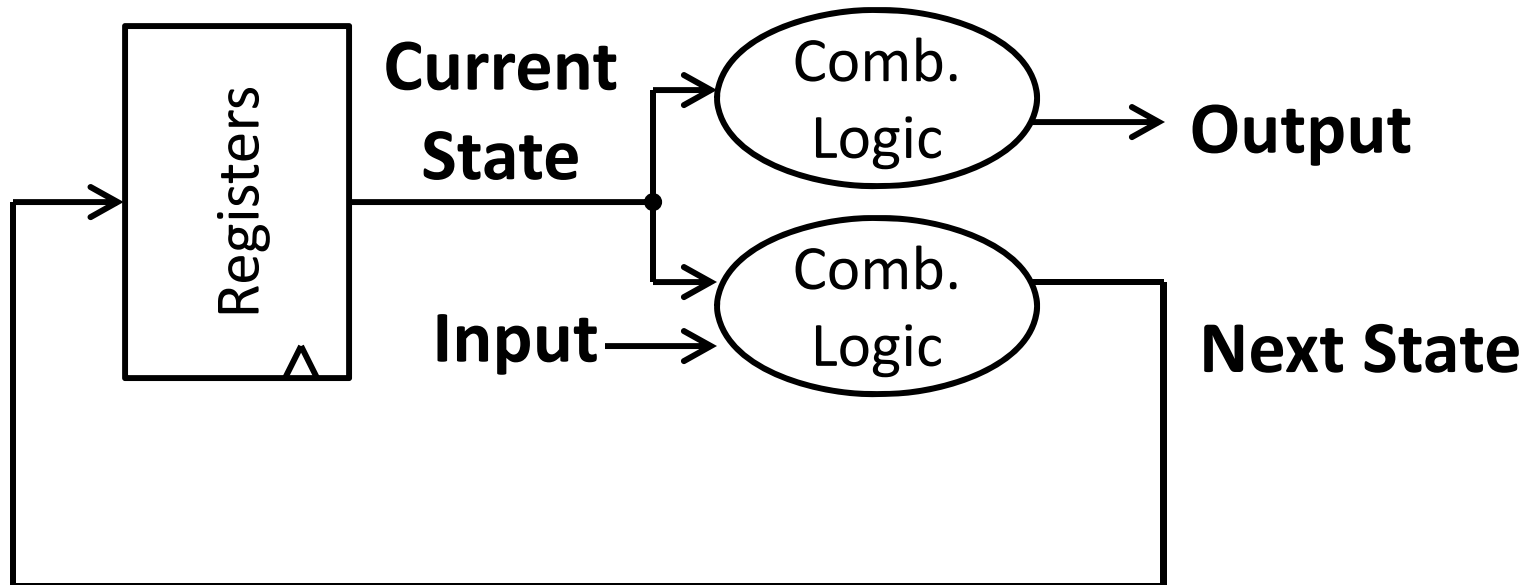
General Case: Mealy Machine



Outputs and next state depend on both current state and input

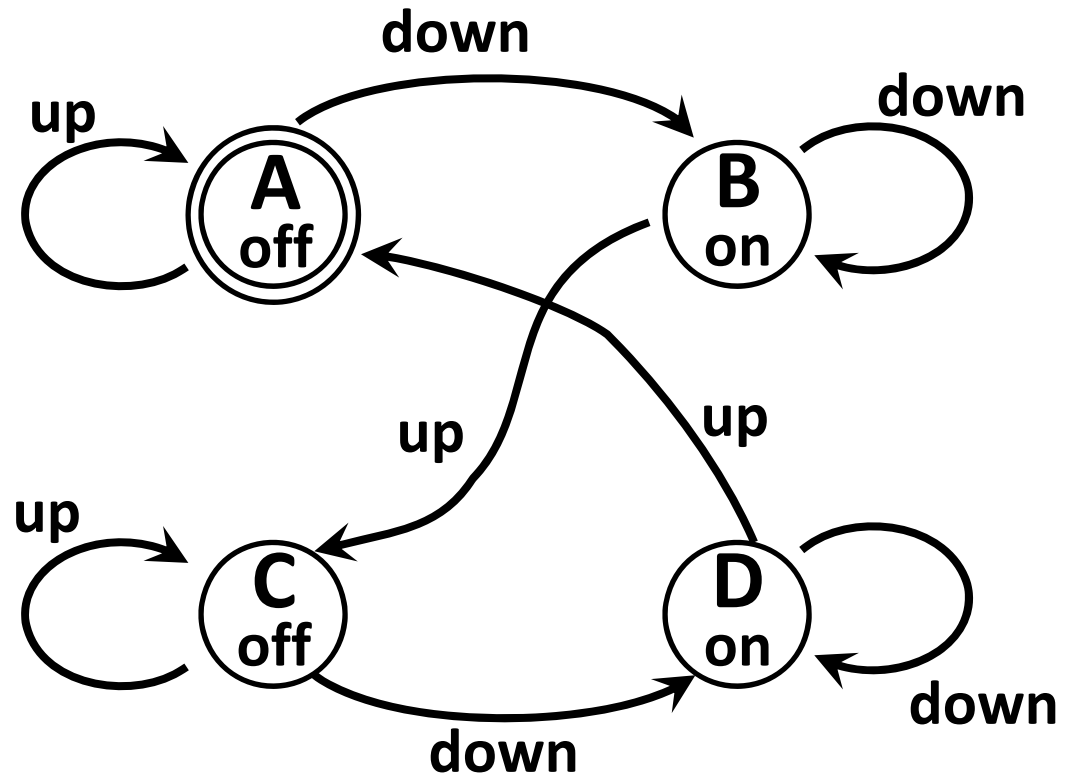
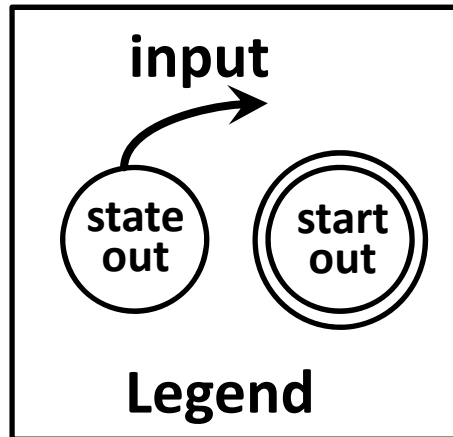
Moore Machine

Special Case: Moore Machine



Outputs depend only on current state

Moore Machine Example



Input: **up** or **down**

Output: **on** or **off**

States: **A, B, C, or D**

Example: Digital Door Lock



Digital Door Lock

Inputs:

- keycodes from keypad
- clock

Outputs:

- “unlock” signal
- display how many keys pressed so far

Door Lock: Inputs

Assumptions:

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



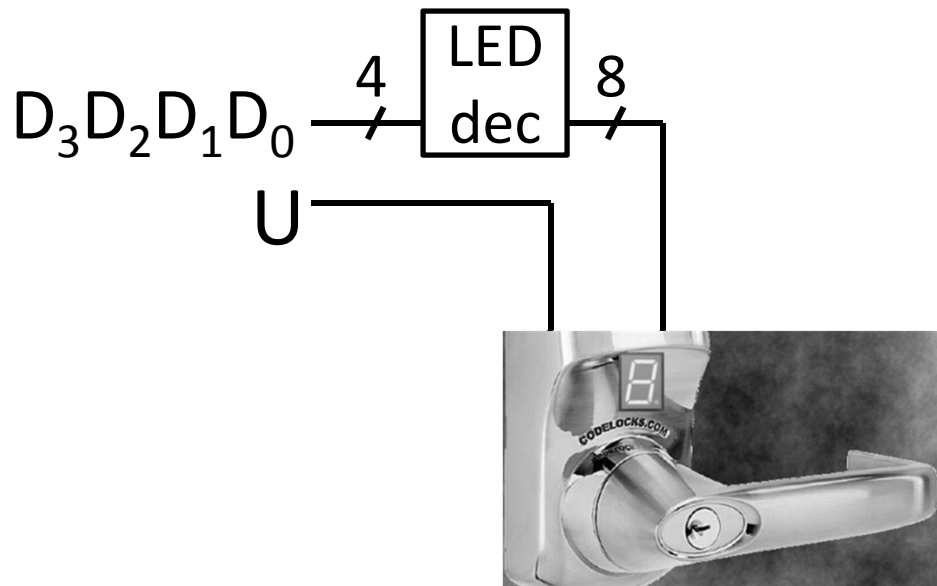
K
A
B

K	A	B	Meaning
0	0	0	∅ (no key)
1	1	0	'A' pressed
1	0	1	'B' pressed

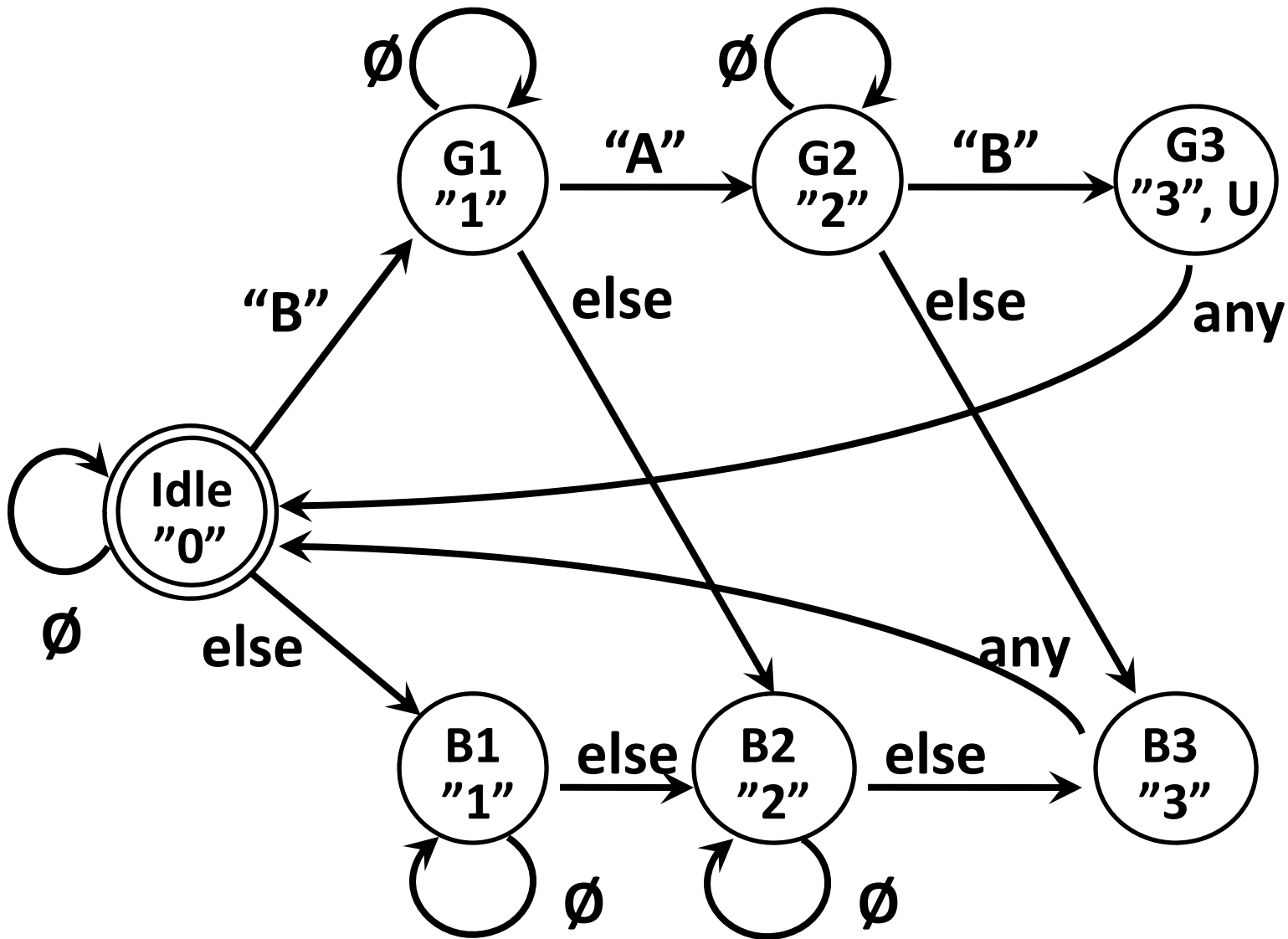
Door Lock: Outputs

Assumptions:

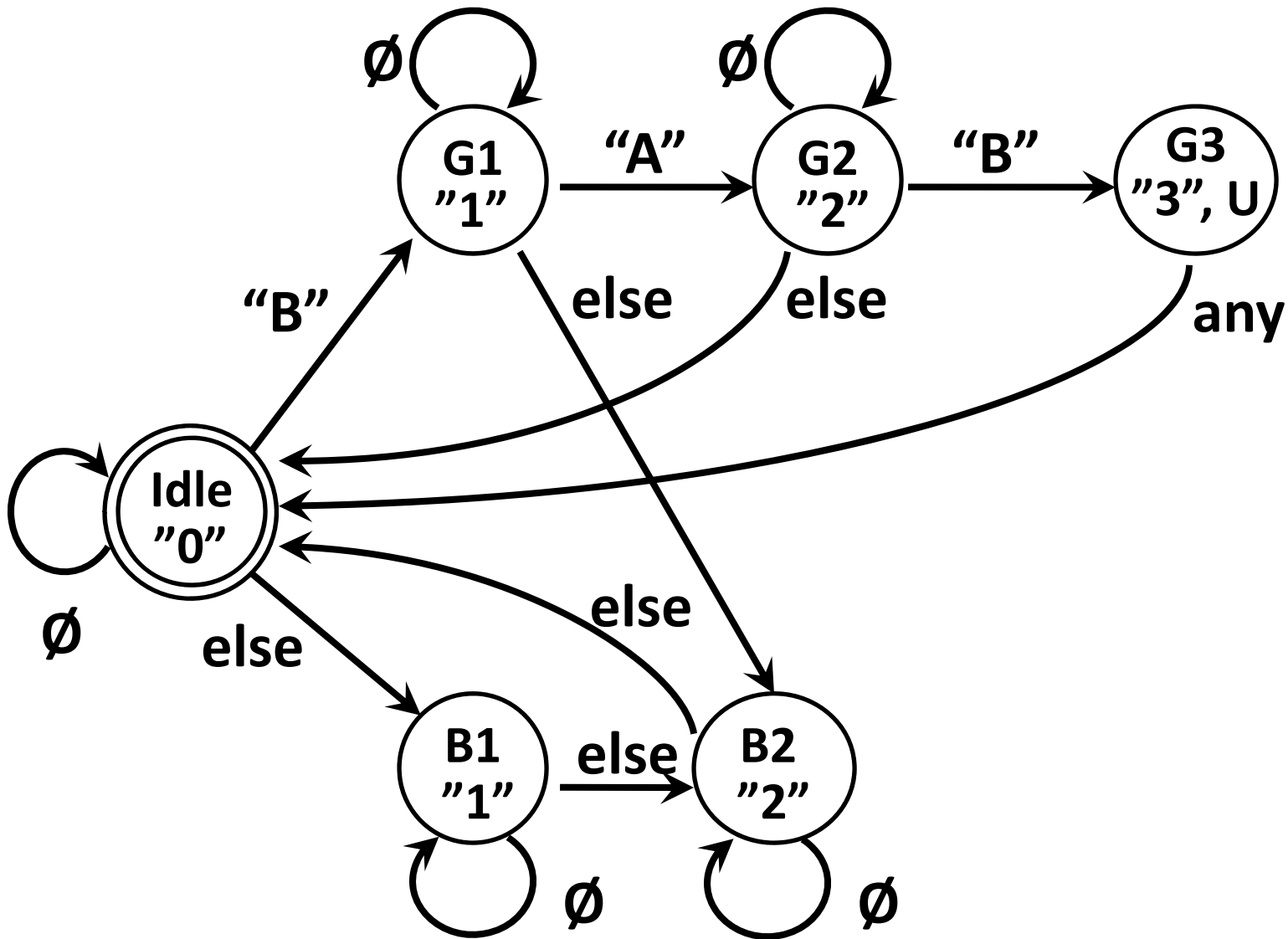
- High pulse on U unlocks door



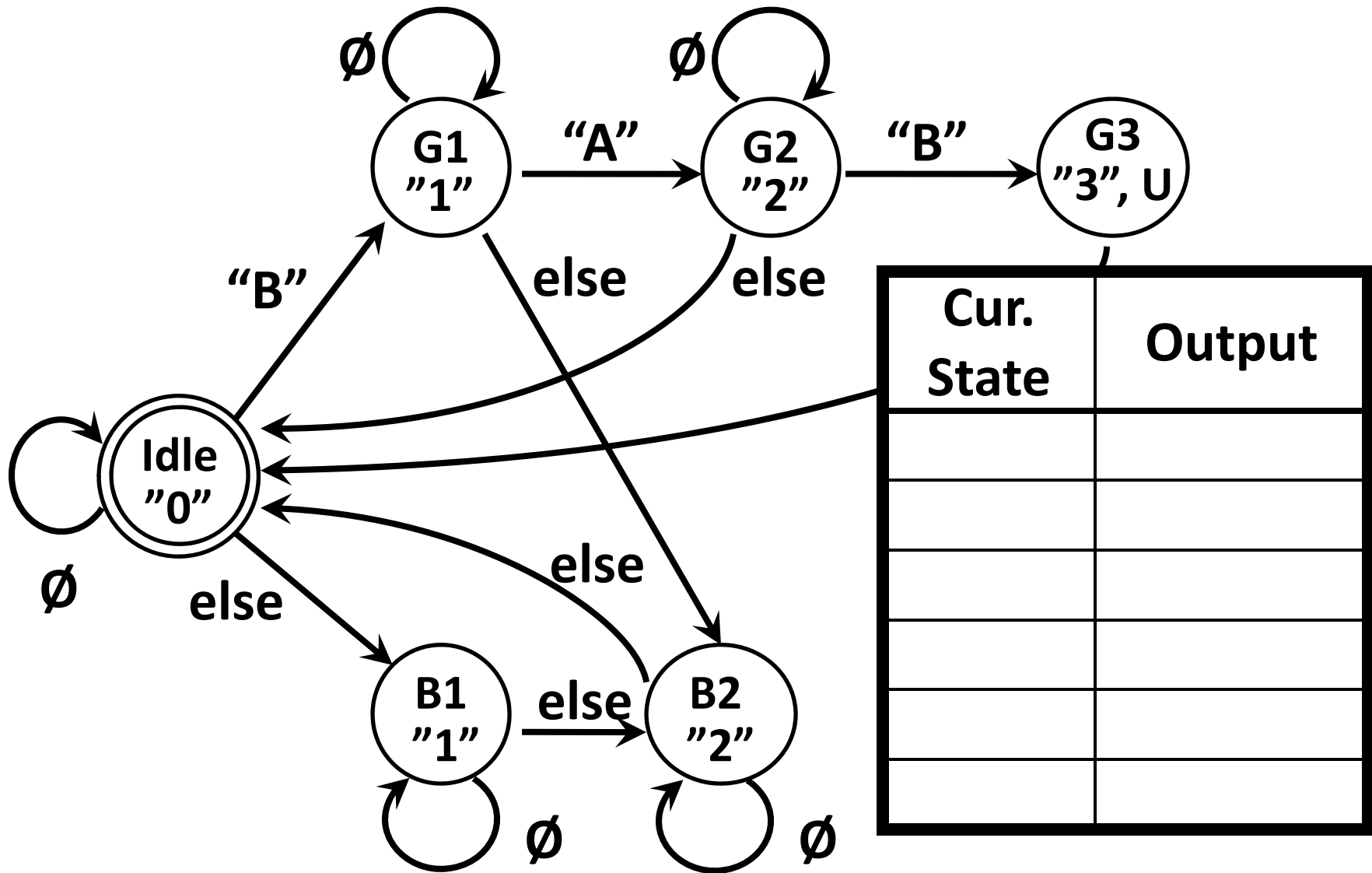
Door Lock: Simplified State Diagram



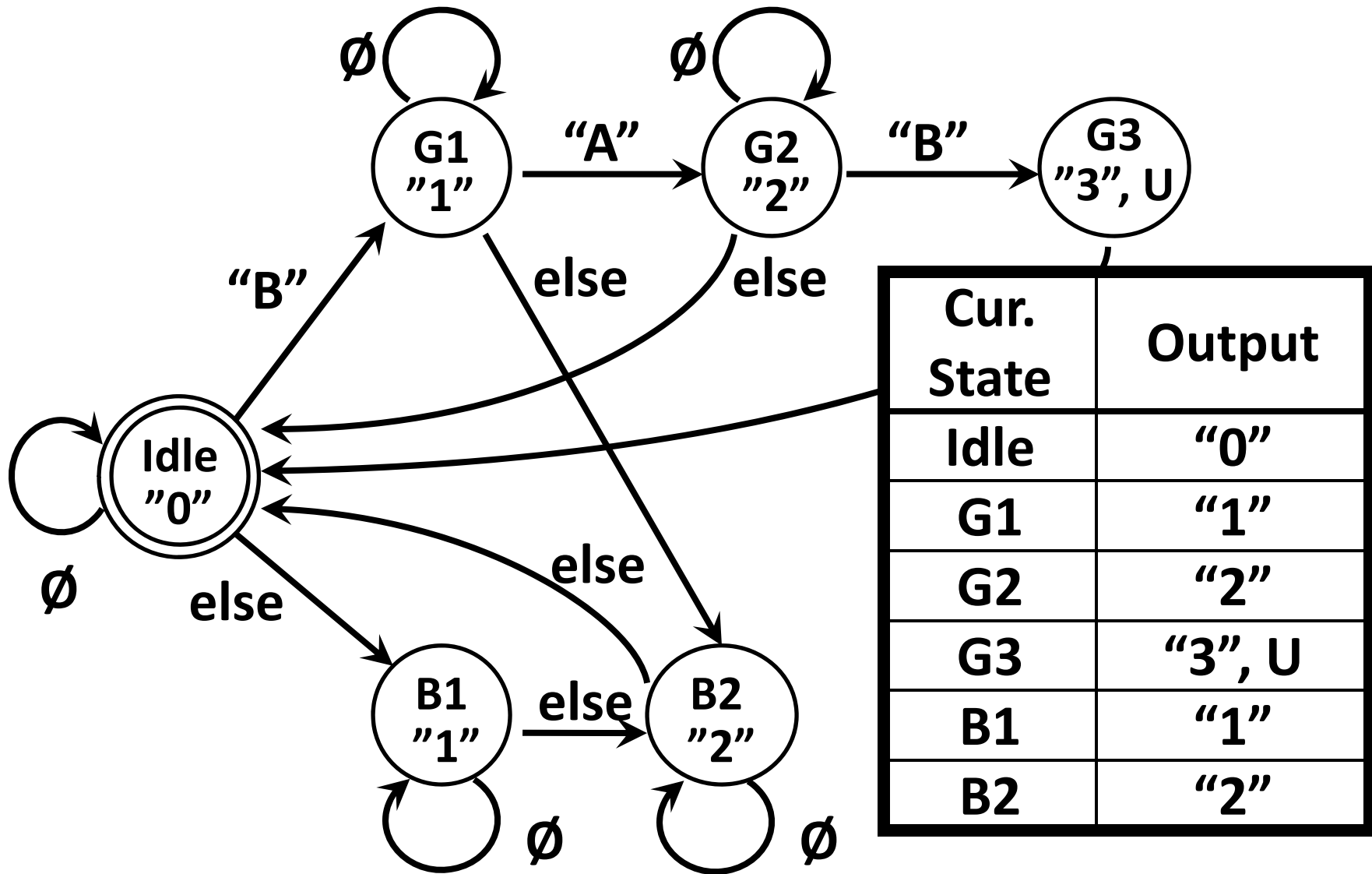
Door Lock: Simplified State Diagram



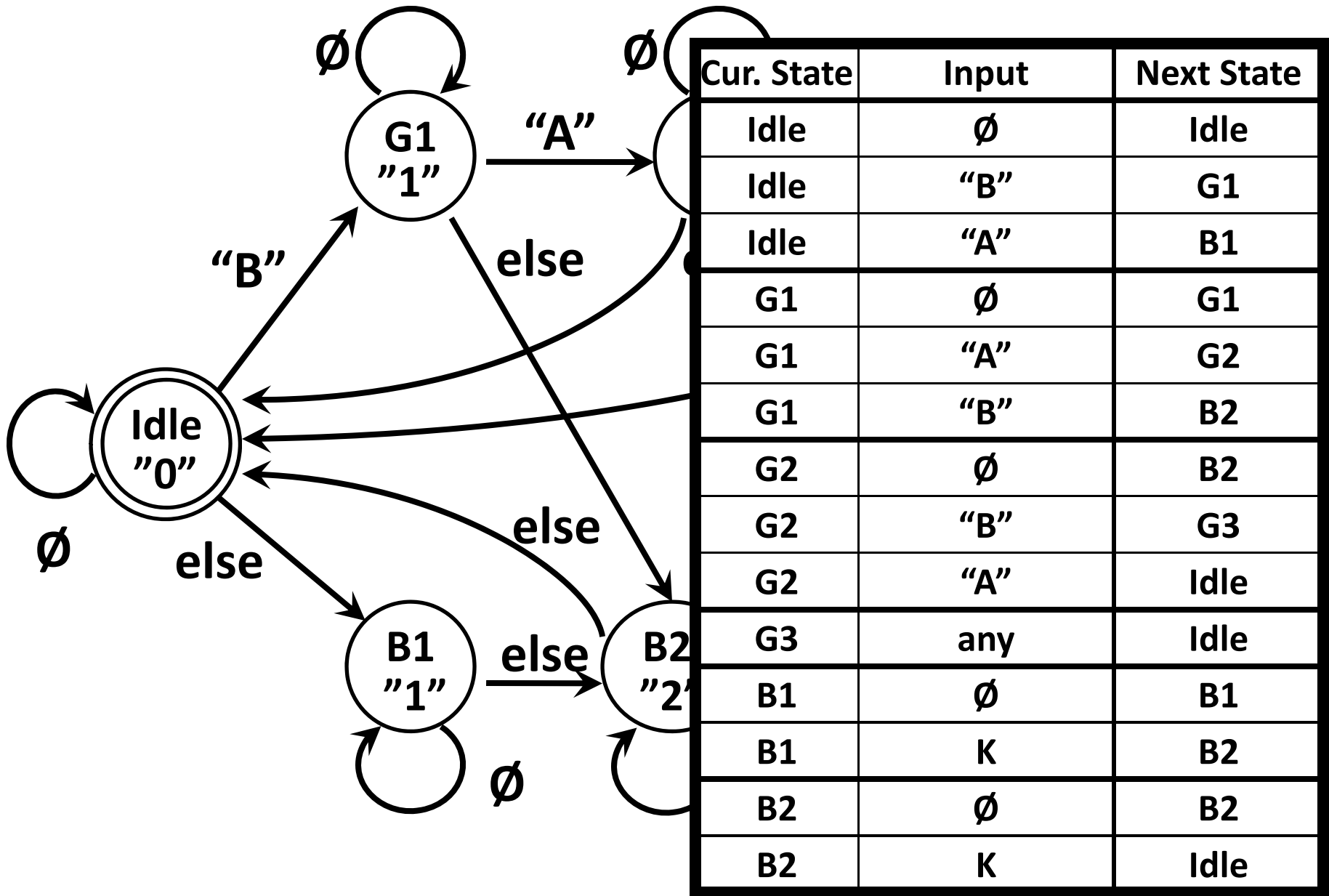
Door Lock: Simplified State Diagram



Door Lock: Simplified State Diagram



Door Lock: Simplified State Diagram



State Table Encoding

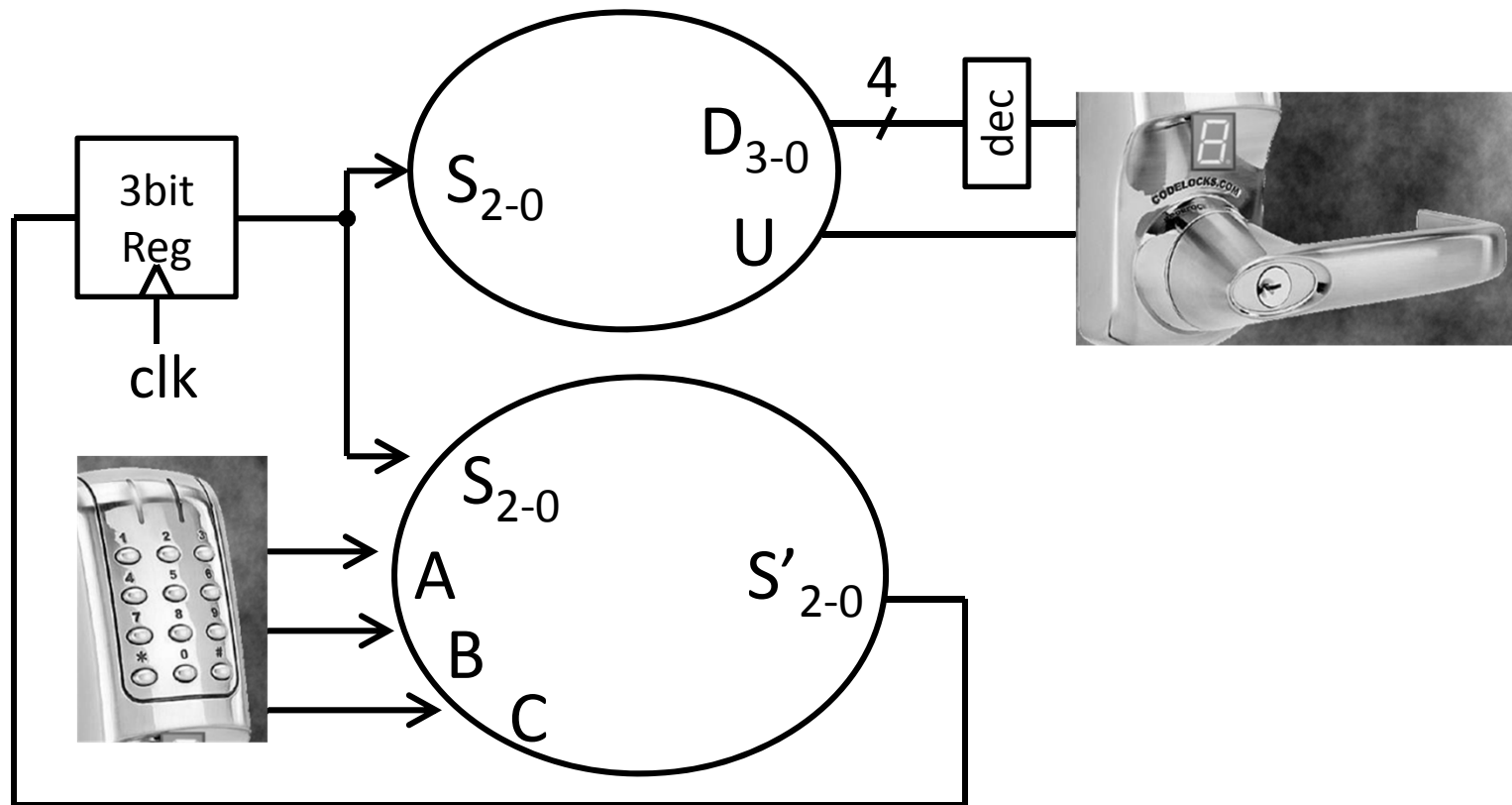
S ₂	S ₁	S ₀	D ₃	D ₂	D ₁	D ₀	U
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	0	0	0	1	0
1	0	1	0	0	1	0	0

D₃ [

State	S ₂	S ₁	S ₀
Idle	0	0	0
G1	0	0	1
G2	0	1	0
G3	0	1	1
B1	1	0	0
B2	1	0	1

S ₂	S ₁	S ₀	K	A	B	S' ₂	S' ₁	S' ₀
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	1
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1
0	0	1	1	1	0	0	1	0
0	0	1	1	0	1	1	0	1
0	1	0	0	0	0	0	1	0
0	1	0	1	0	1	0	1	1
0	1	0	1	1	0	0	0	0
0	1	1	x	x	x	0	0	0
1	0	0	0	0	0	1	0	0
1	0	0	1	x	x	1	0	1
1	0	1	0	0	0	1	0	1
1	0	1	1	x	x	0	0	0

Door Lock: Implementation



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

Summary

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