

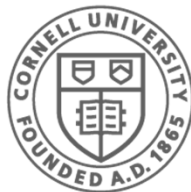
Finite State Machines

Hakim Weatherspoon

CS 3410

Computer Science

Cornell University



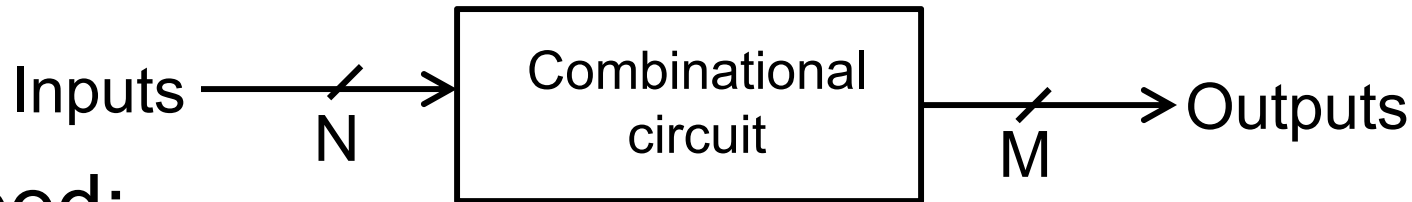
Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[Weatherspoon, Bala, Bracy, McKee, and Sirer]

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

Sequential Logic & Finite State Machines

Goals for Today

- Finite State Machines (FSM)
 - How do we design logic circuits with state?
 - Types of FSMs: Mealy and Moore Machines
 - Examples: Serial Adder and a Digital Door Lock

Next Goal

- How do we design logic circuits with state?

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

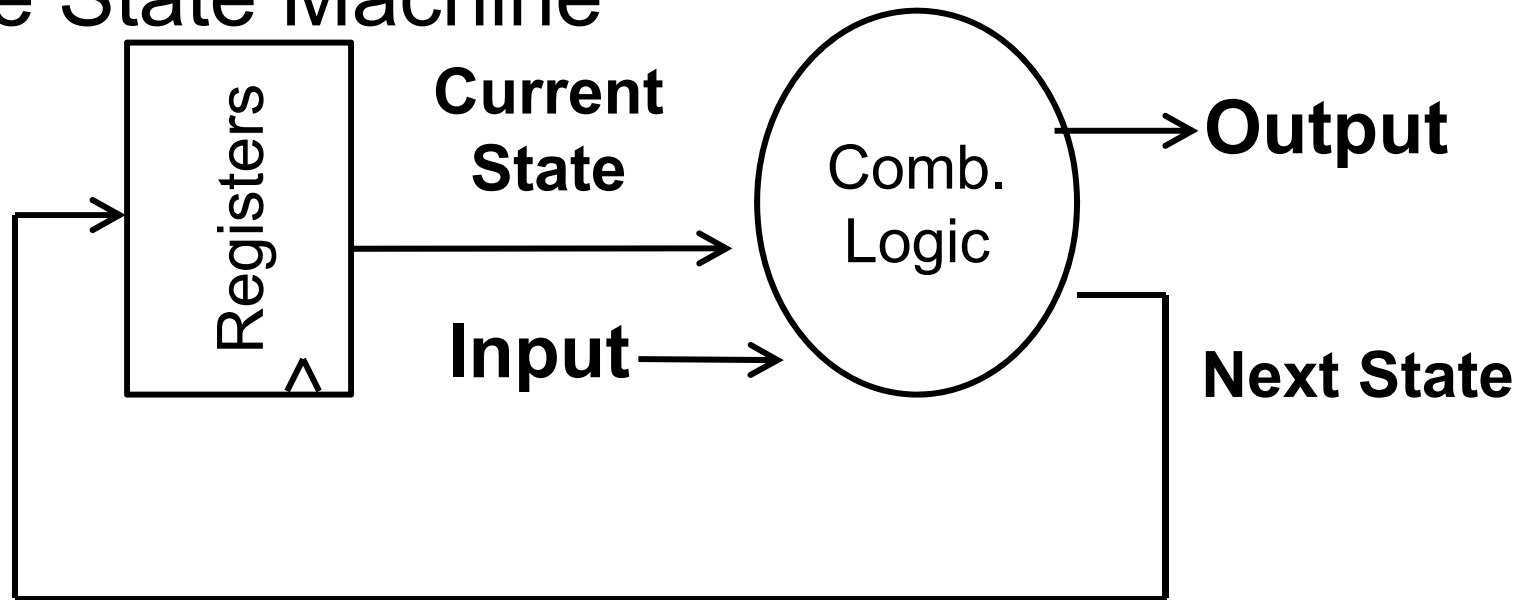
O : Finite set of outputs

δ : State transition function

Next state depends on present input *and* present state

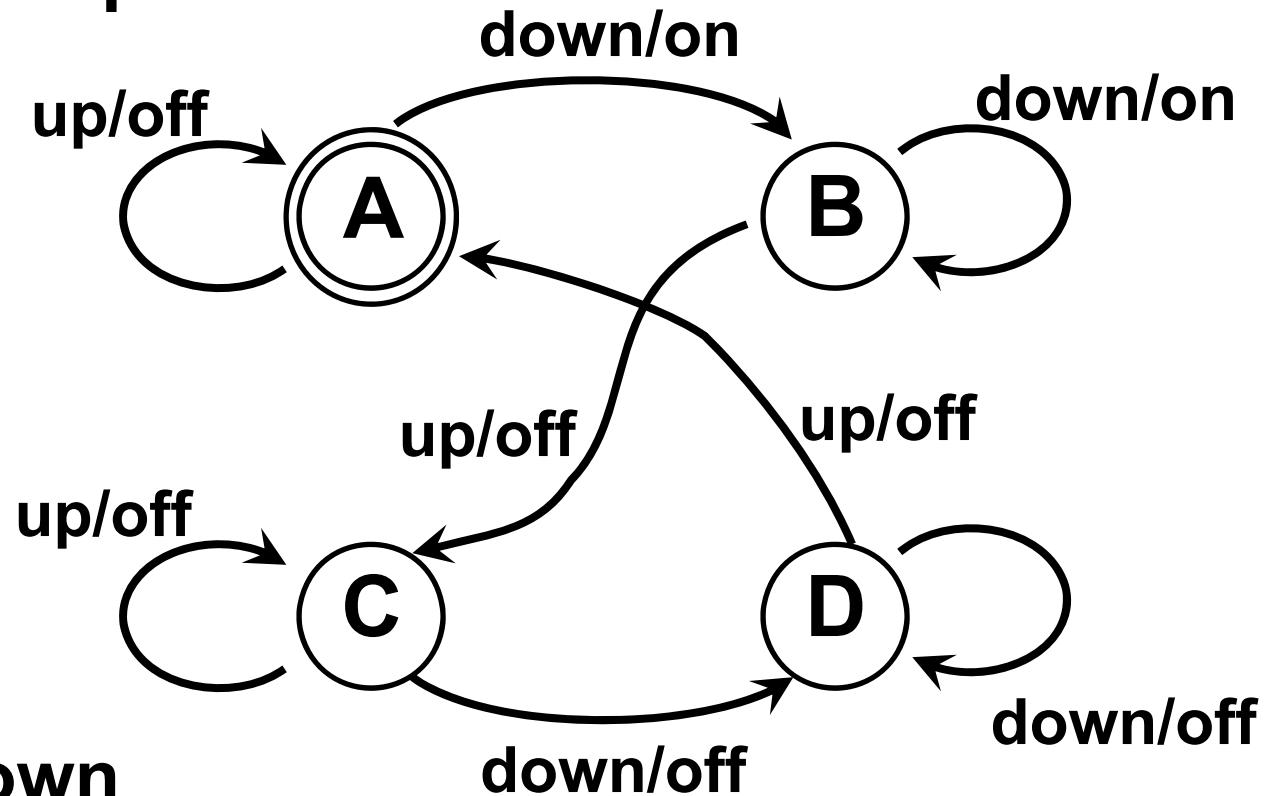
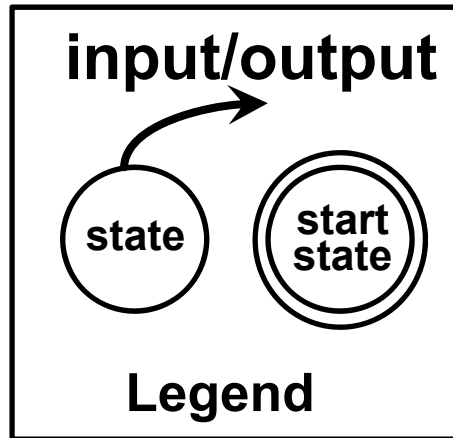
Automata Model

Finite State Machine



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

FSM Example

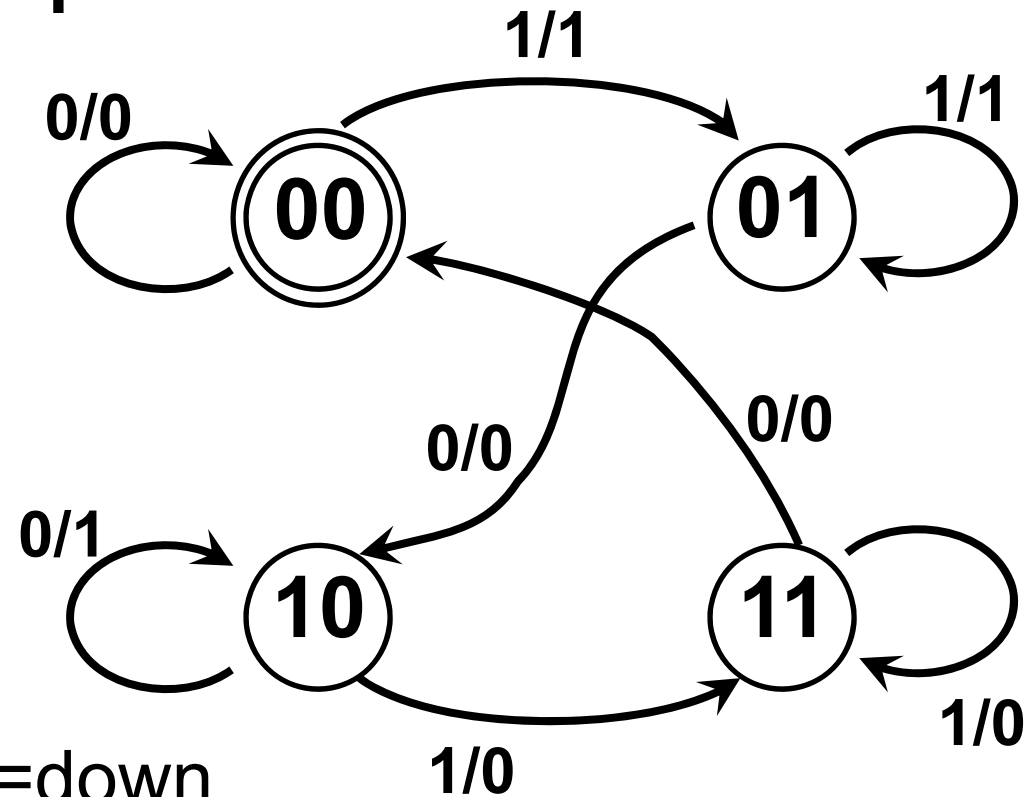
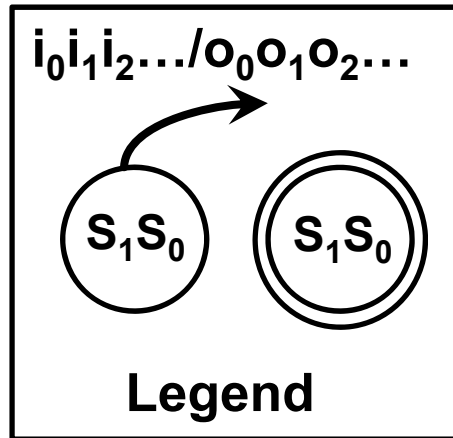


Input: **up** or **down**

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

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

FSM Example



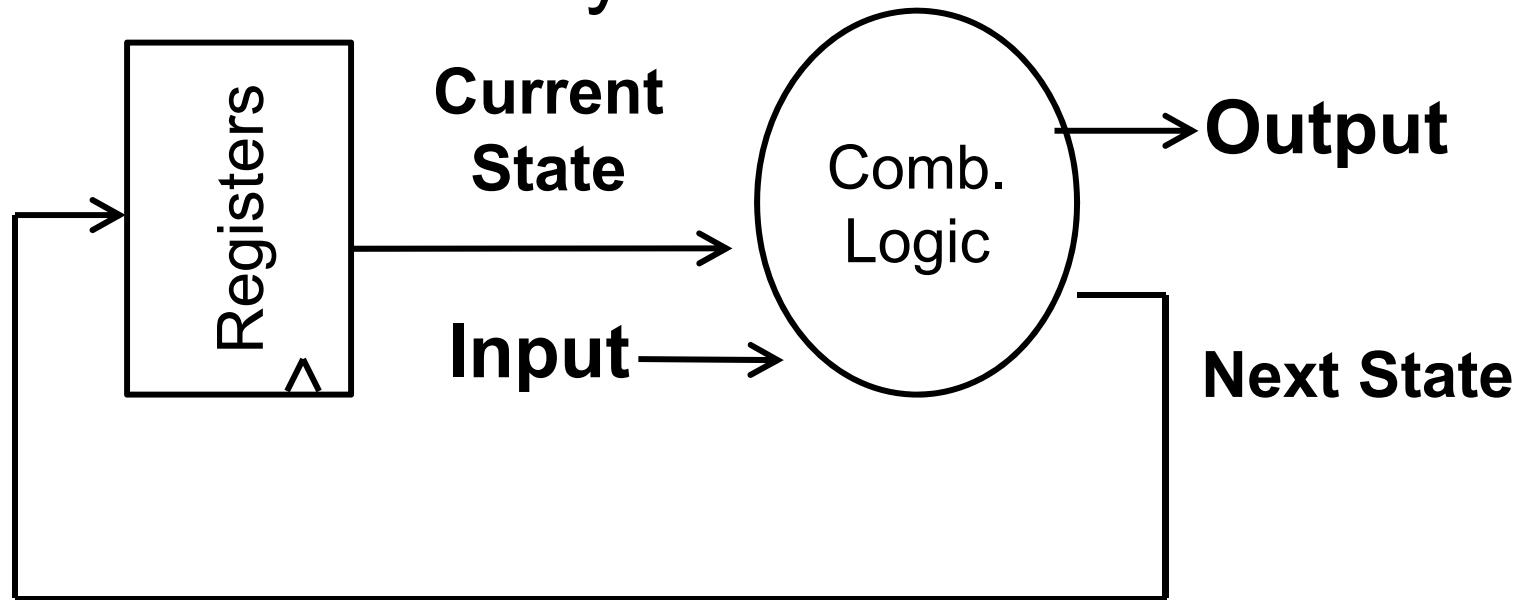
Input: **0**=up or **1**=down

Output: **1**=on or **0**=off

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

Mealy Machine

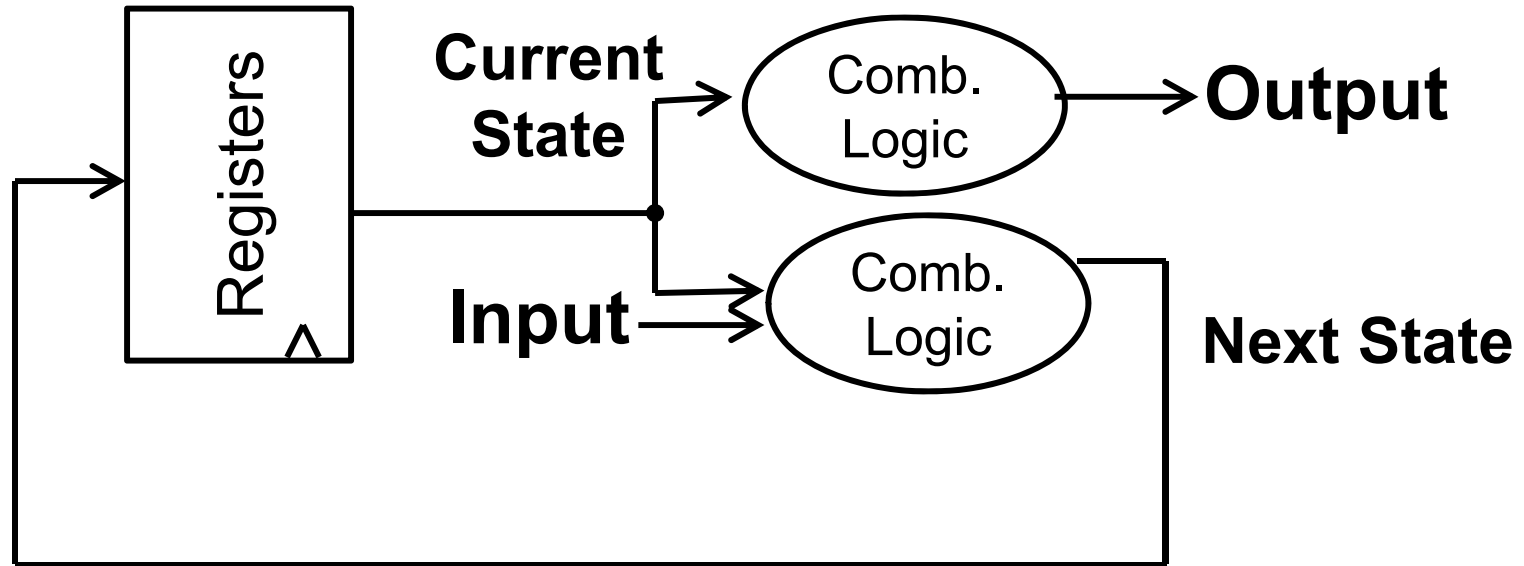
General Case: Mealy Machine



Outputs and next state depend on both current state and input

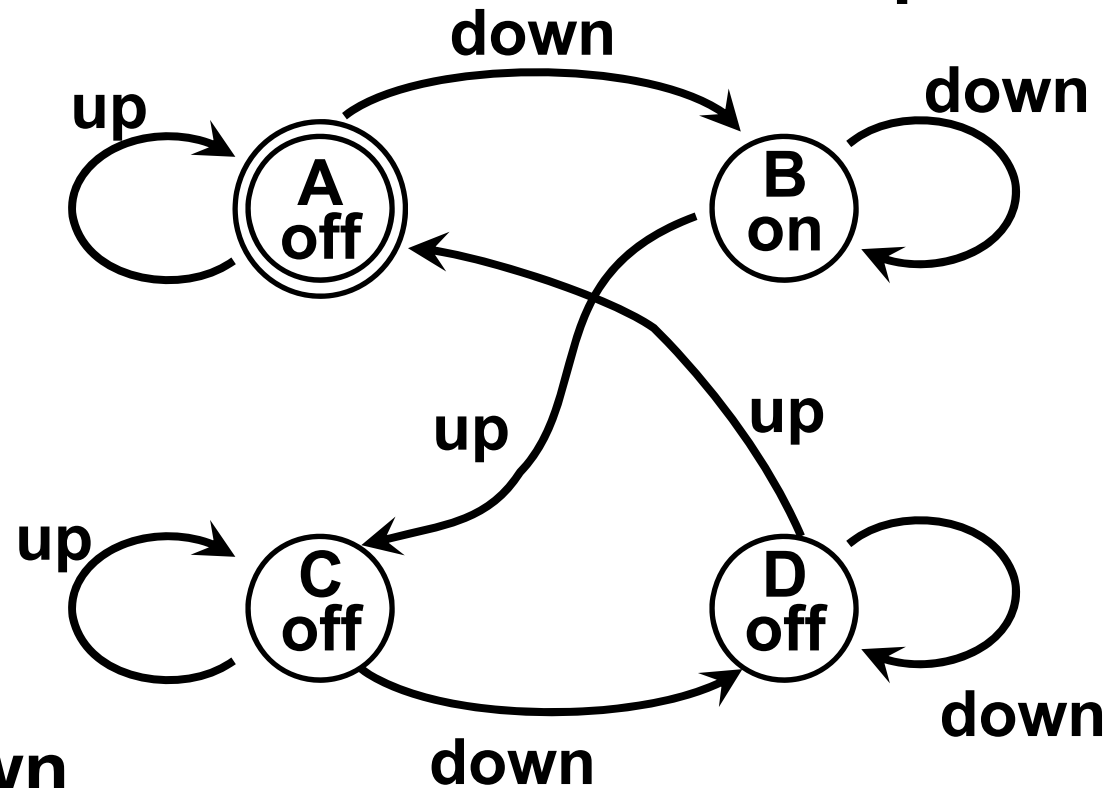
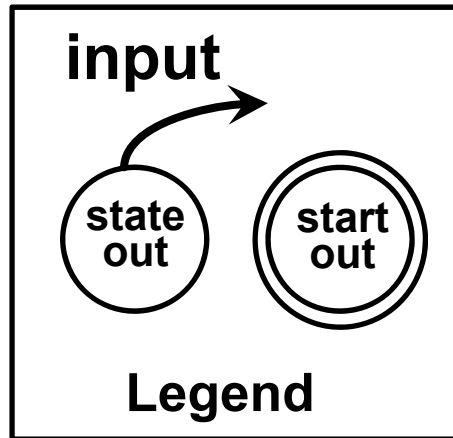
Moore Machine

Special Case: Moore Machine



Outputs depend only on current state

Moore Machine FSM Example

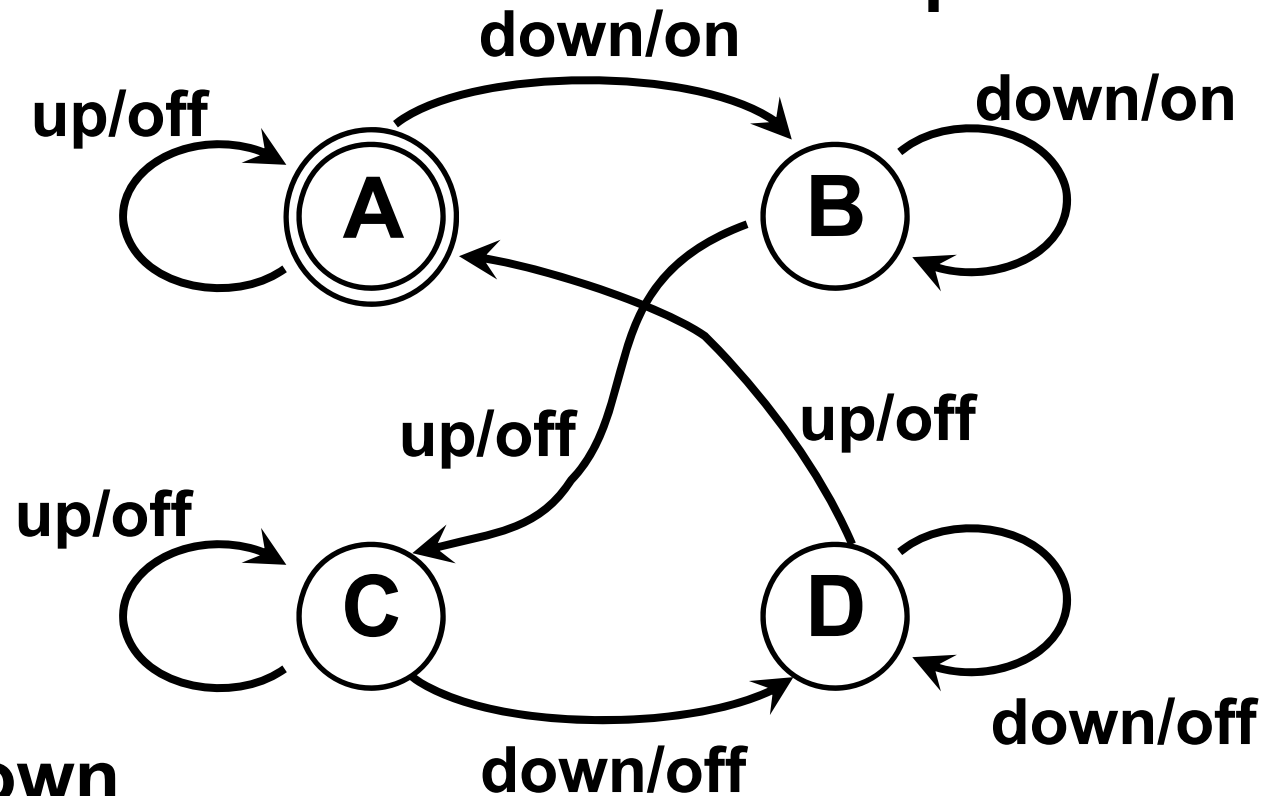
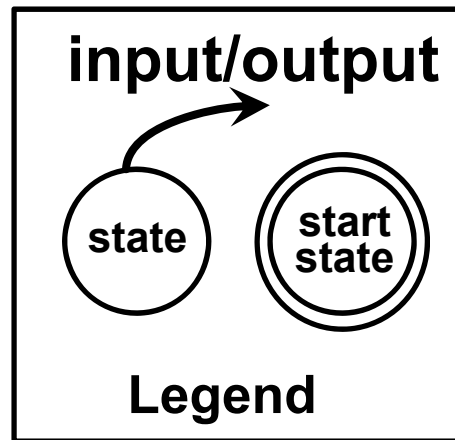


Input: **up** or **down**

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

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

Mealy Machine FSM Example



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

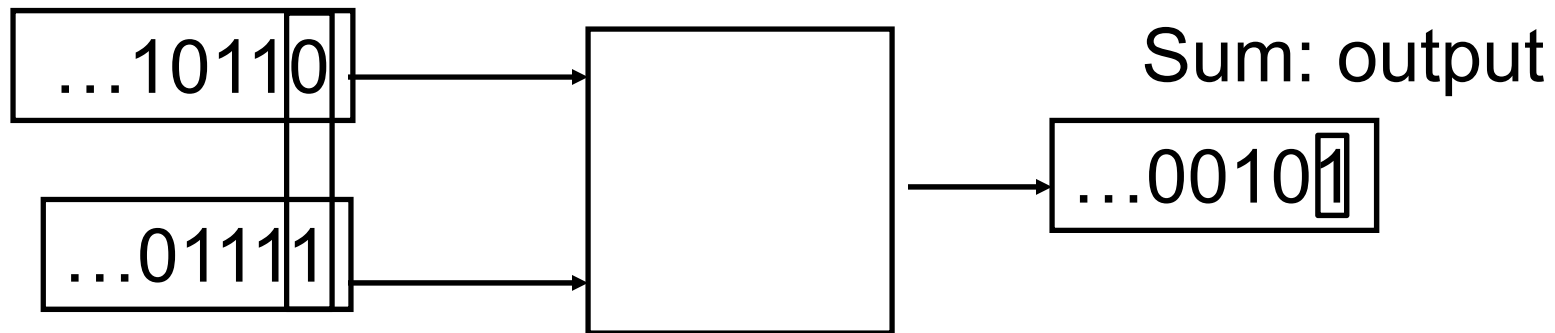
Output: **on** or **off**

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

Activity#2: Create a Logic Circuit for a Serial Adder

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first

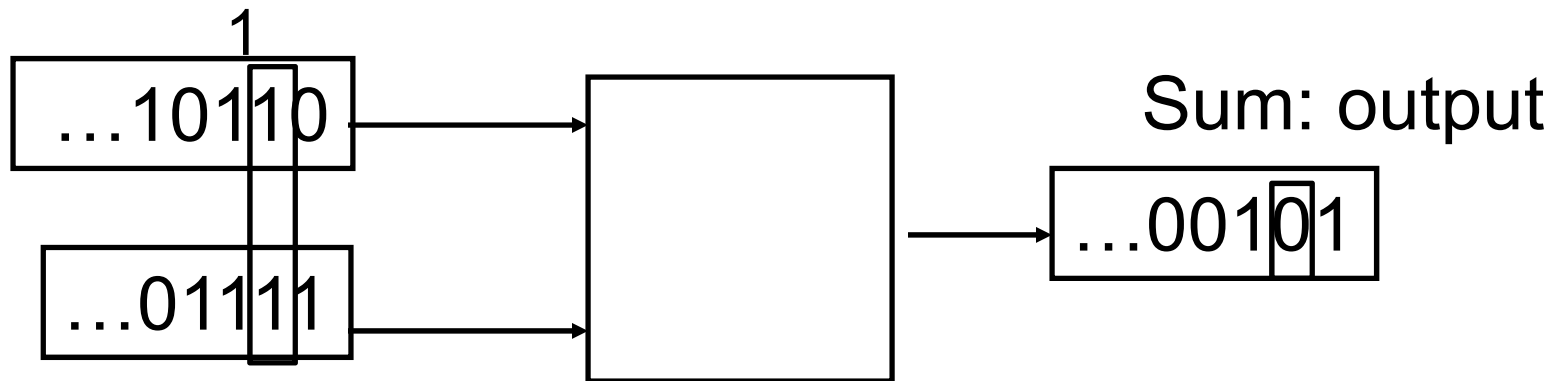


Activity#2: Create a Logic Circuit for a Serial Adder

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first

Carry-out

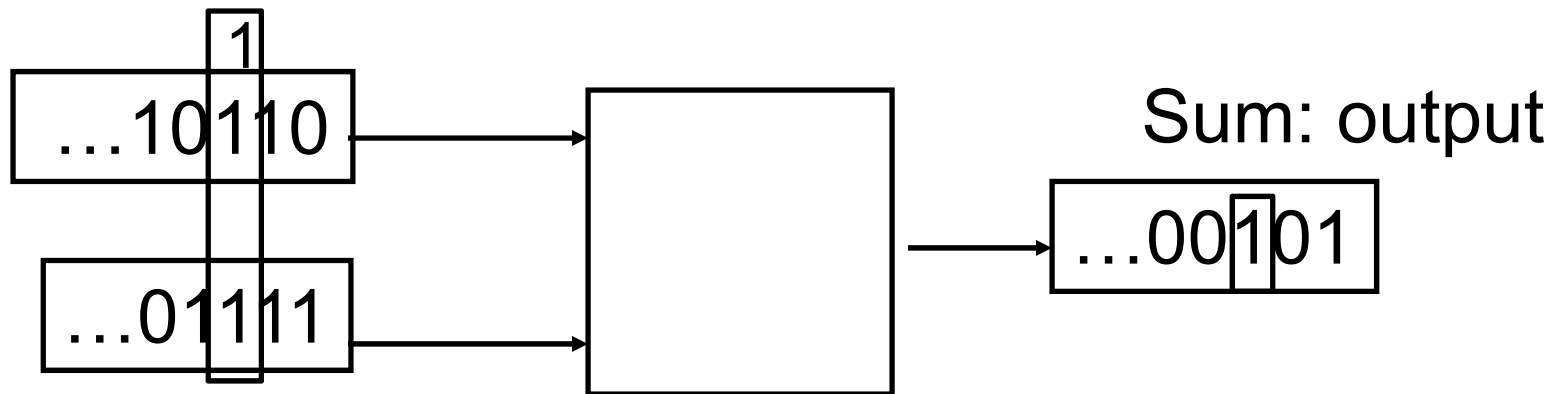


Activity#2: Create a Logic Circuit for a Serial Adder

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first

Carry-in

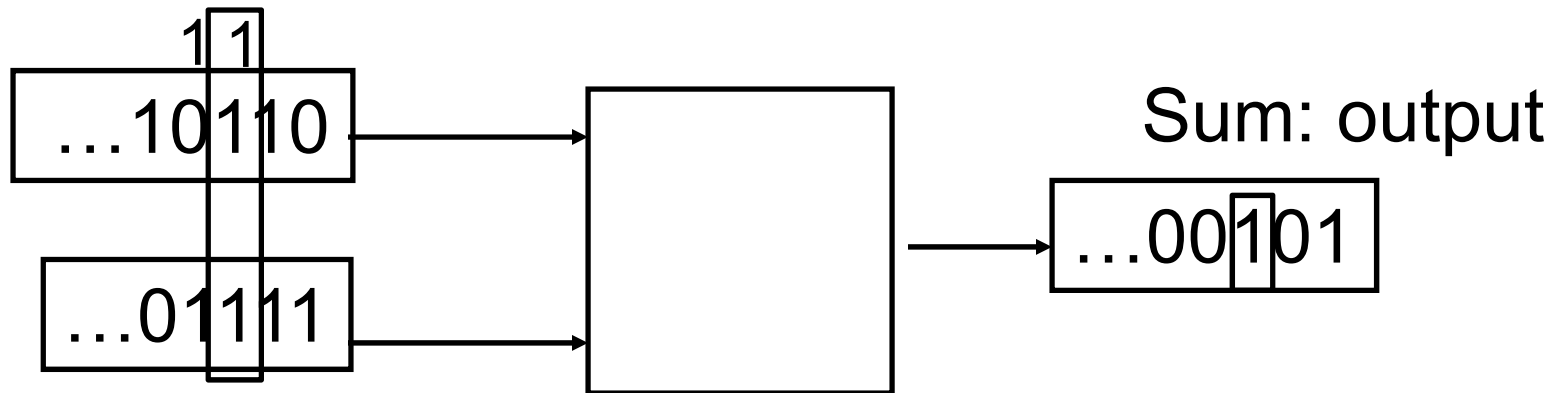


Activity#2: Create a Logic Circuit for a Serial Adder

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first

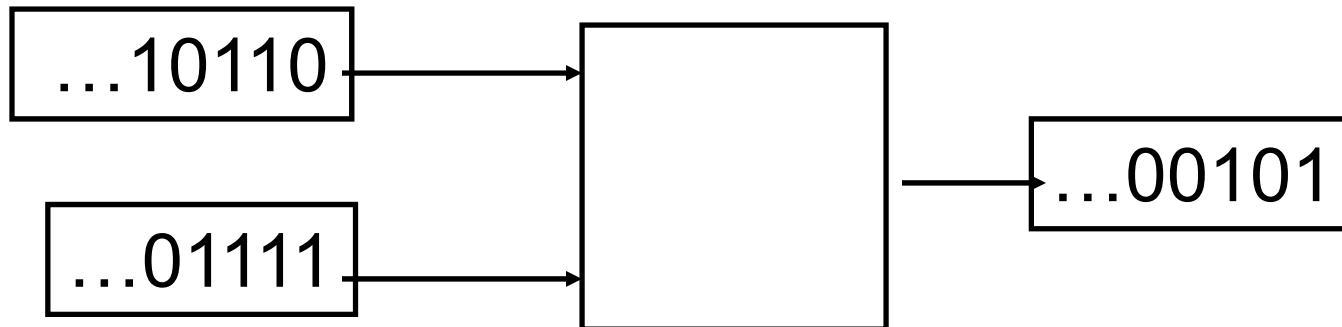
Carry-out



iClicker Question

Add two infinite input bit streams

- streams are sent with least-significant-bit (lsb) first



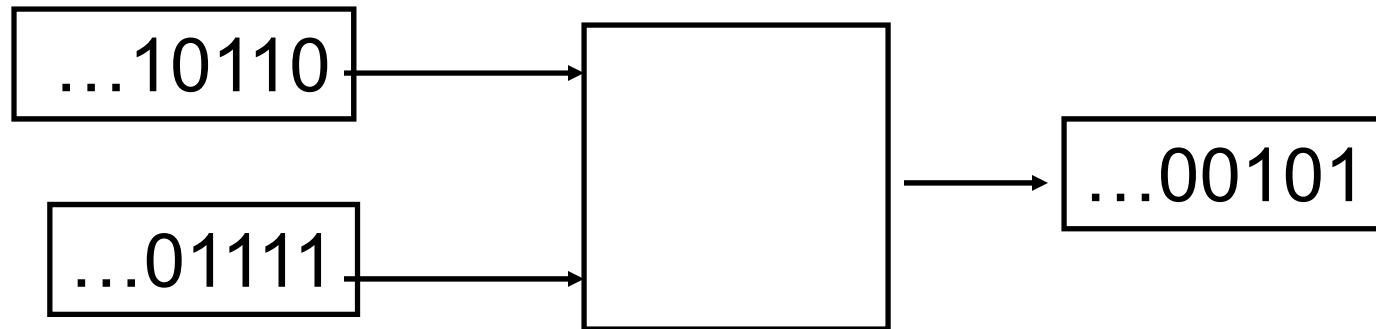
How many states are needed to represent FSM

- a) 0
- b) 1
- c) 2
- d) 3
- e) 4

Strategy for Building an FSM

- (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
- (5) Draw the Circuit

FSM: State Diagram

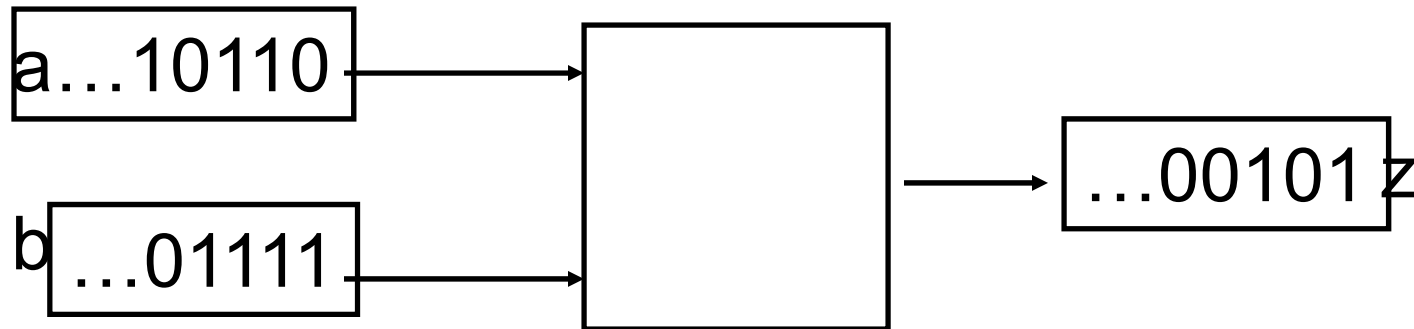


2 states ____ and ____

Inputs: ____ and ____

Output: ____

FSM: State Diagram



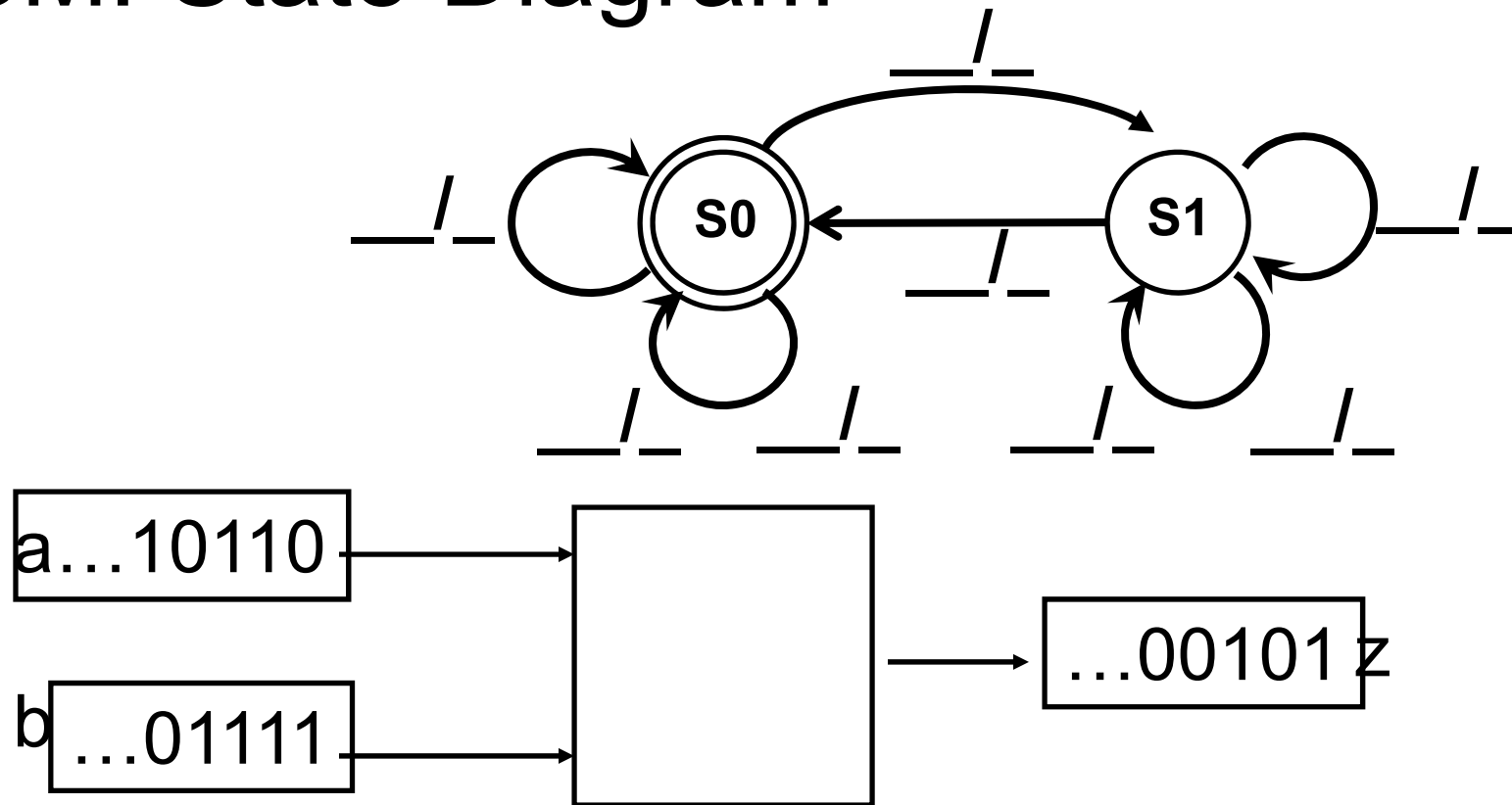
Two states: S0 (no carry in), S1 (carry in)

Inputs: a and b

Output: z

- z is the sum of inputs a, b, and carry-in (one bit at a time)
- A carry-out *is* the next carry-in state.

FSM: State Diagram



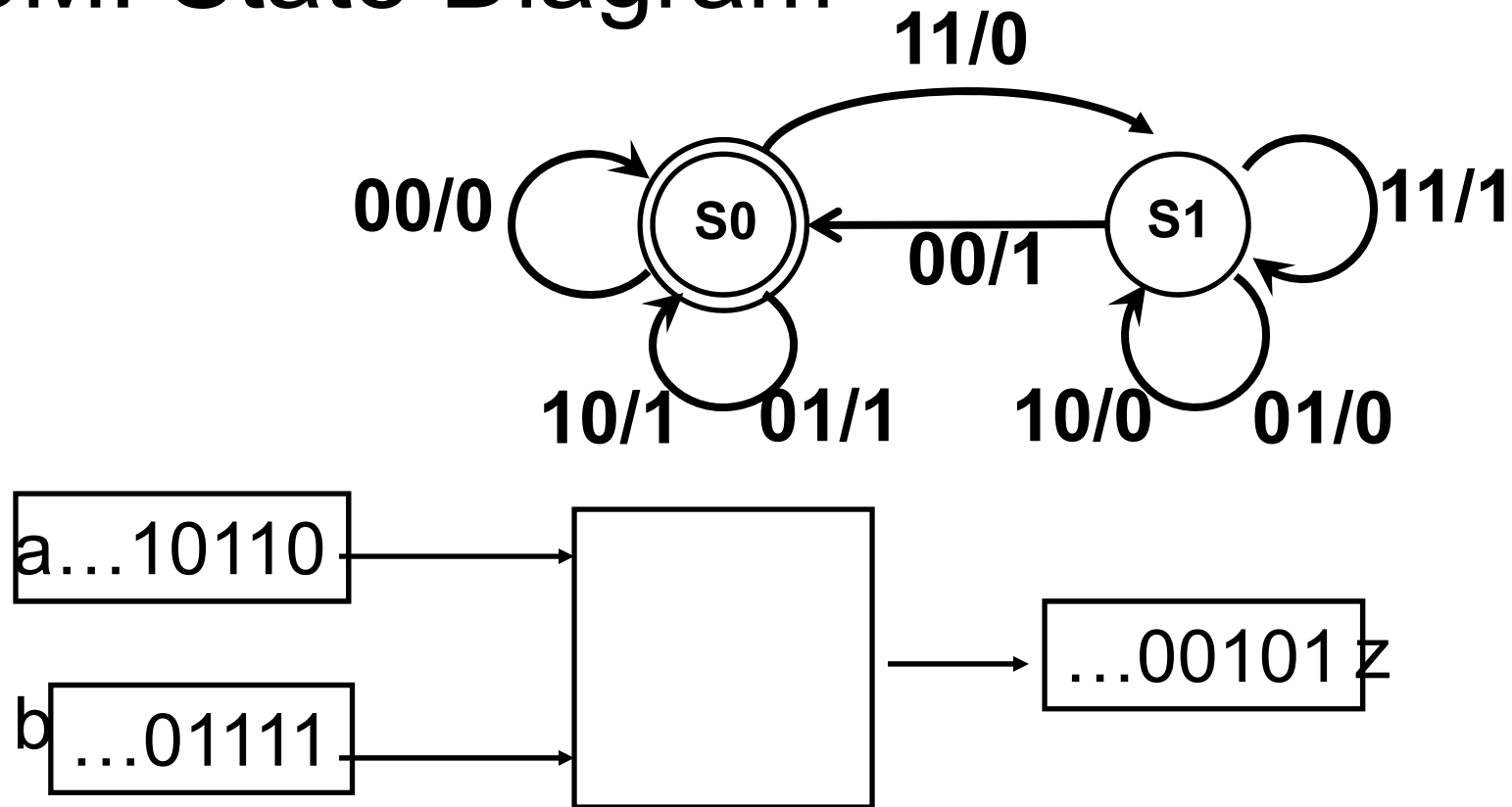
Two states: S0 (no carry in), S1 (carry in)

Inputs: a and b

Output: z

- z is the sum of inputs a, b, and carry-in (one bit at a time)
- A carry-out *is* the next carry-in state.
- Arcs labeled with input bits a and b, and output z

FSM: State Diagram



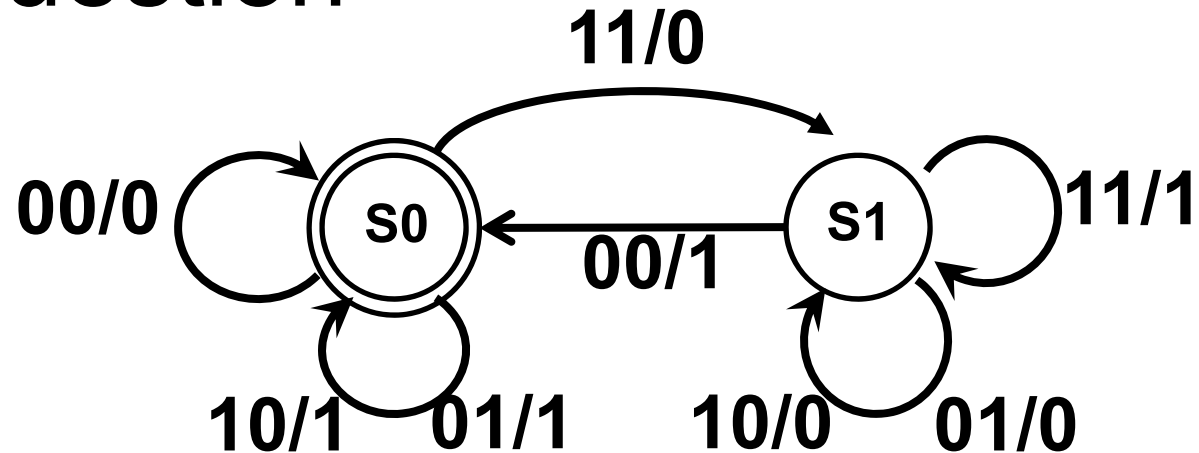
Two states: S0 (no carry in), S1 (carry in)

Inputs: a and b

Output: z

- z is the sum of inputs a, b, and carry-in (one bit at a time)
- A carry-out *is* the next carry-in state.
- Arcs labeled with input bits a and b, and output z

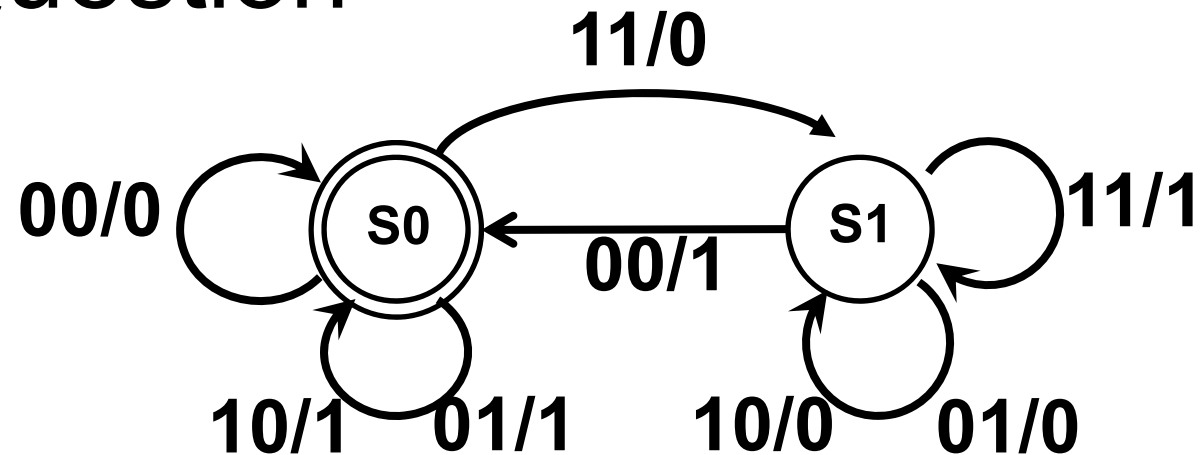
iClicker Question



Is this a Moore or Mealy Machine?

- a) Moore
- b) Mealy
- c) Cannot be determined

iClicker Question



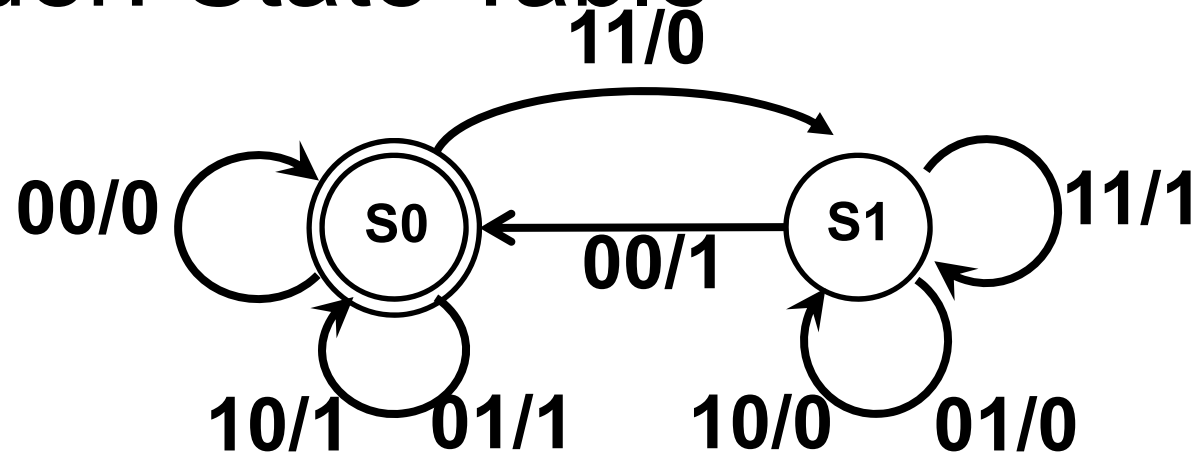
Is this a Moore or Mealy Machine?

a) Moore

b) Mealy

c) Cannot be determined

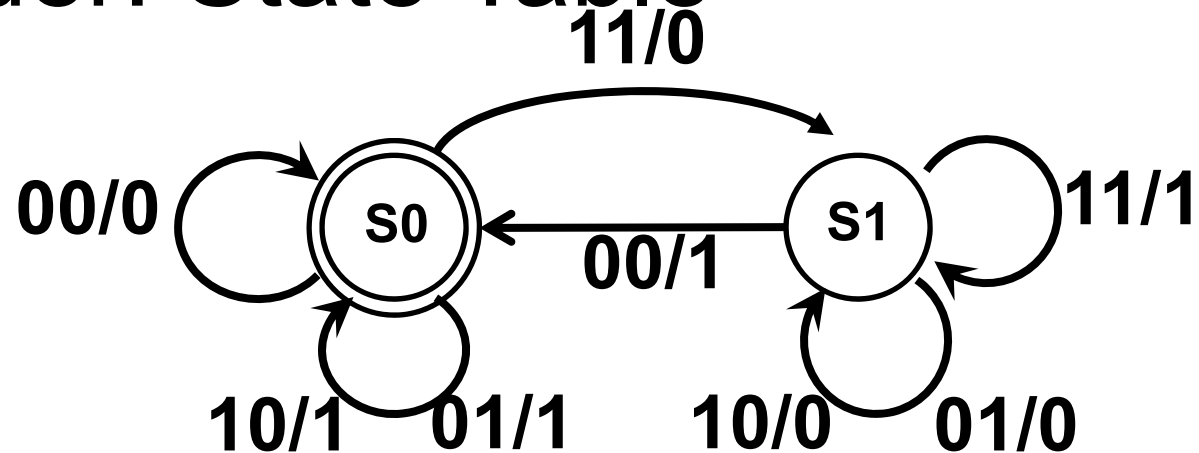
Serial Adder: State Table



a	b	Current state	z	Next state

(2) Write down all input and state combinations

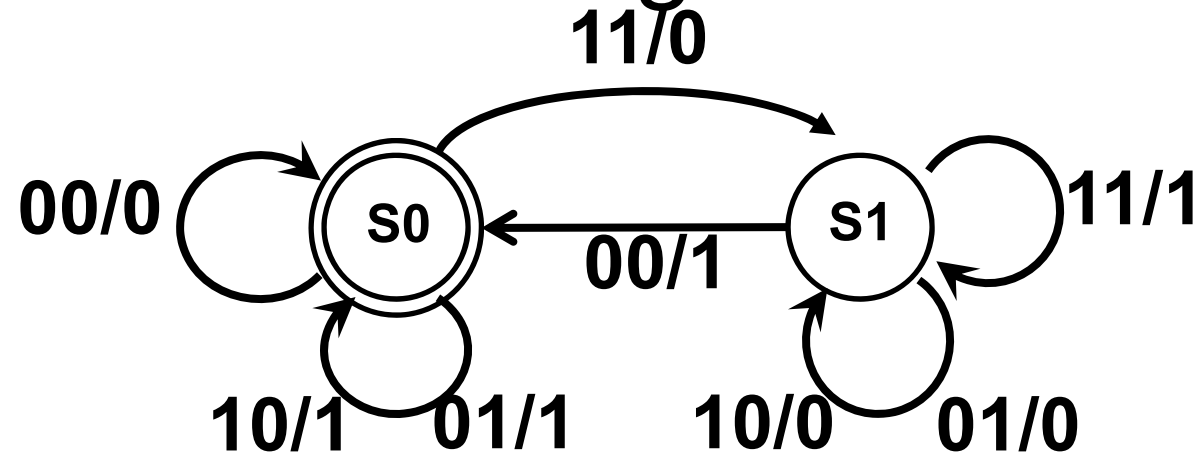
Serial Adder: State Table



a	b	Current state	z	Next state
0	0	S0	0	S0
0	1	S0	1	S0
1	0	S0	1	S0
1	1	S0	0	S1
0	0	S1	1	S0
0	1	S1	0	S1
1	0	S1	0	S1
1	1	S1	1	S1

(2) Write down all input and state combinations

Serial Adder: State Assignment



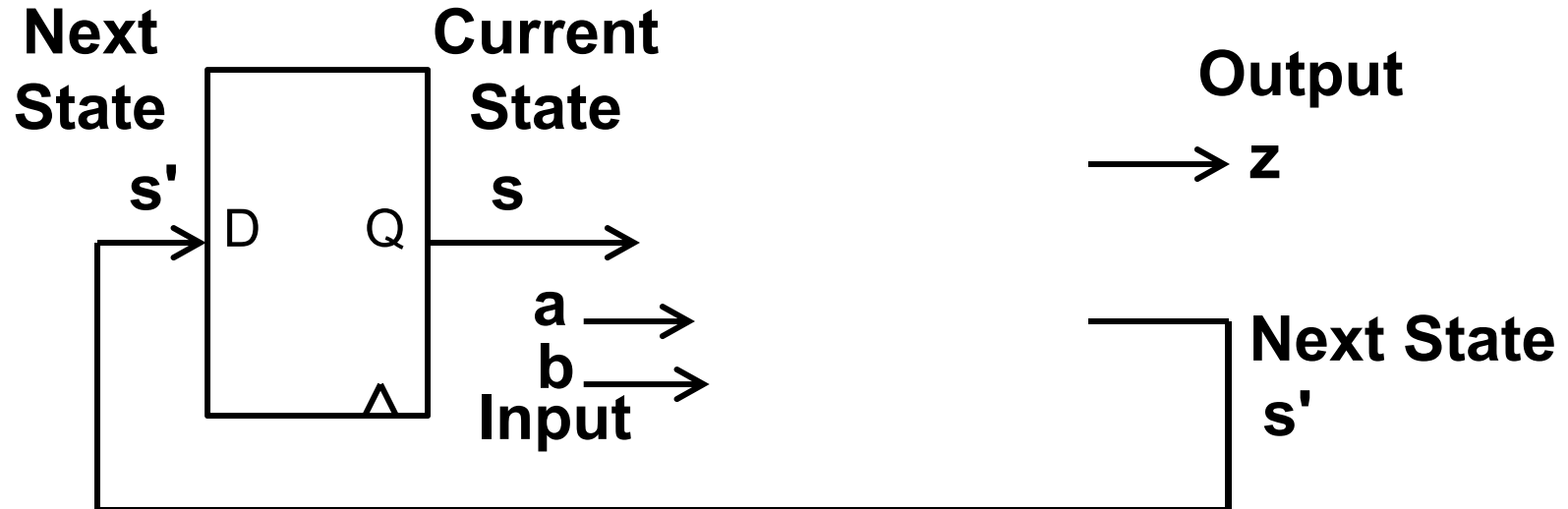
a	b	s	z	s'
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

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

Two states, so 1-bit is sufficient

- A single flip-flop will encode the state

Serial Adder: Circuit



a	b	s	z	s'
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

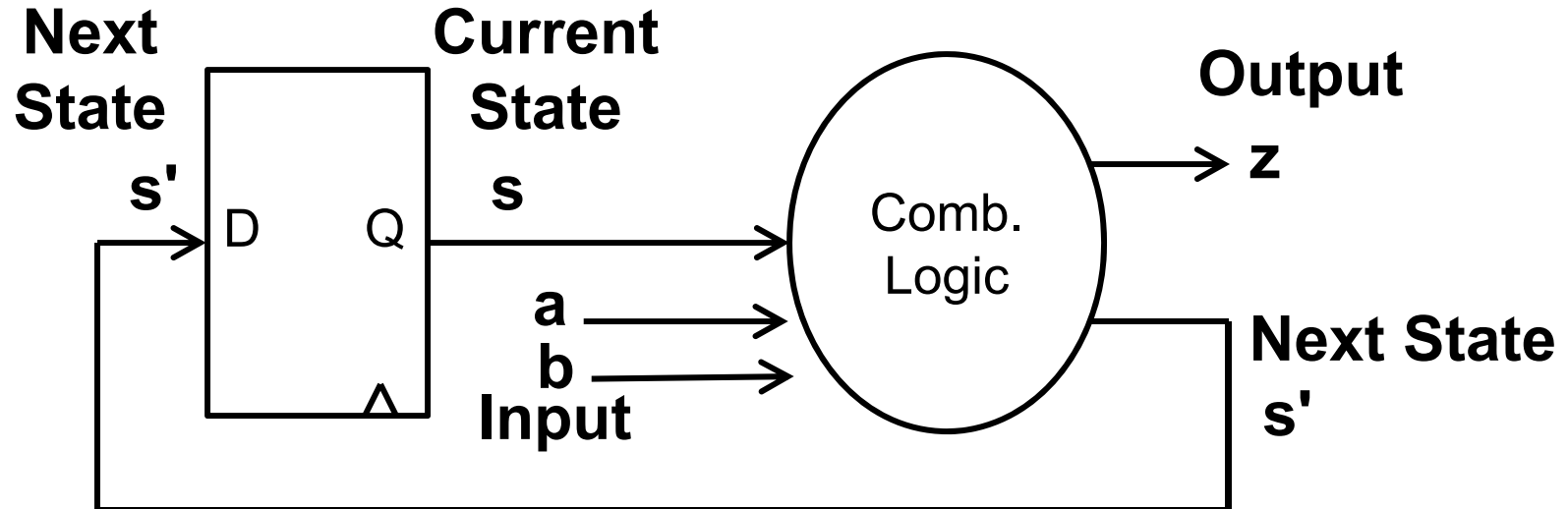
(4) Determine logic equations for next state and outputs

Combinational Logic Equations

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

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

Serial Adder: Circuit



a	b	s	z	s'
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

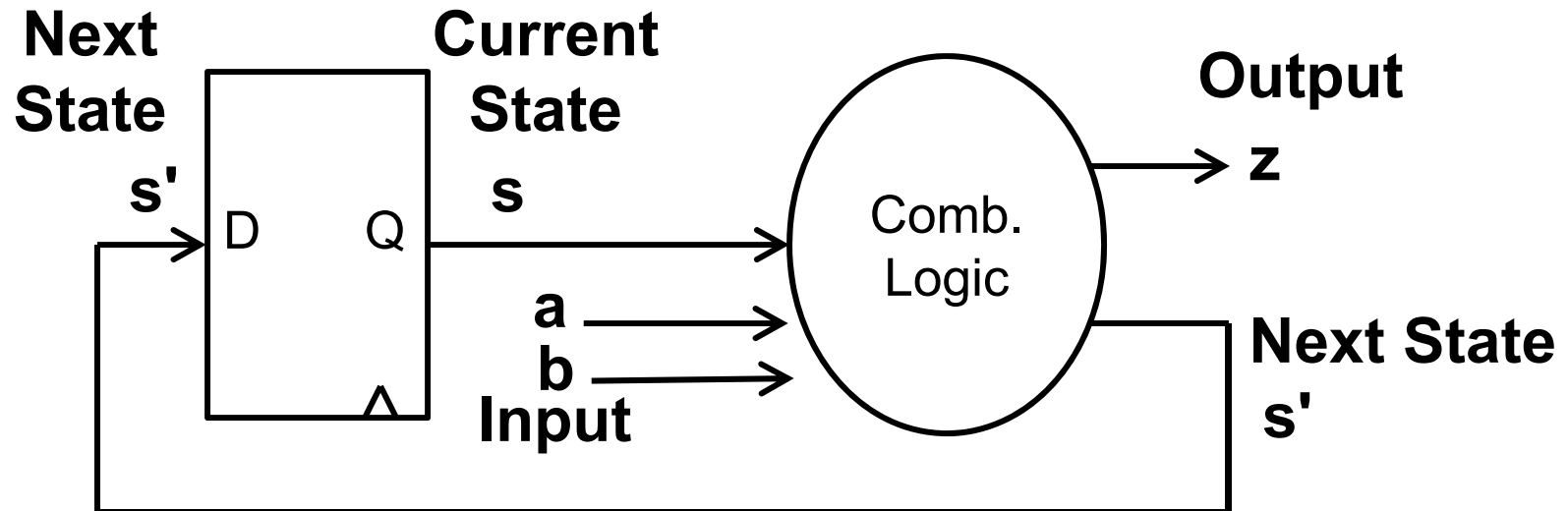
(4) Determine logic equations for next state and outputs

Combinational Logic Equations

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

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

Sequential Logic Circuits



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

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

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

Which statement(s) is true

- (A) In a Moore Machine output depends on both current state and input
- (B) In a Mealy Machine output depends on both current state and input
- (C) In a Mealy Machine output depends on next state and input
- (D) All the above are true
- (E) None are true

Which statement(s) is true

(A) In a Moore Machine output depends on both current state and input

(B) In a Mealy Machine output depends on both current state and input

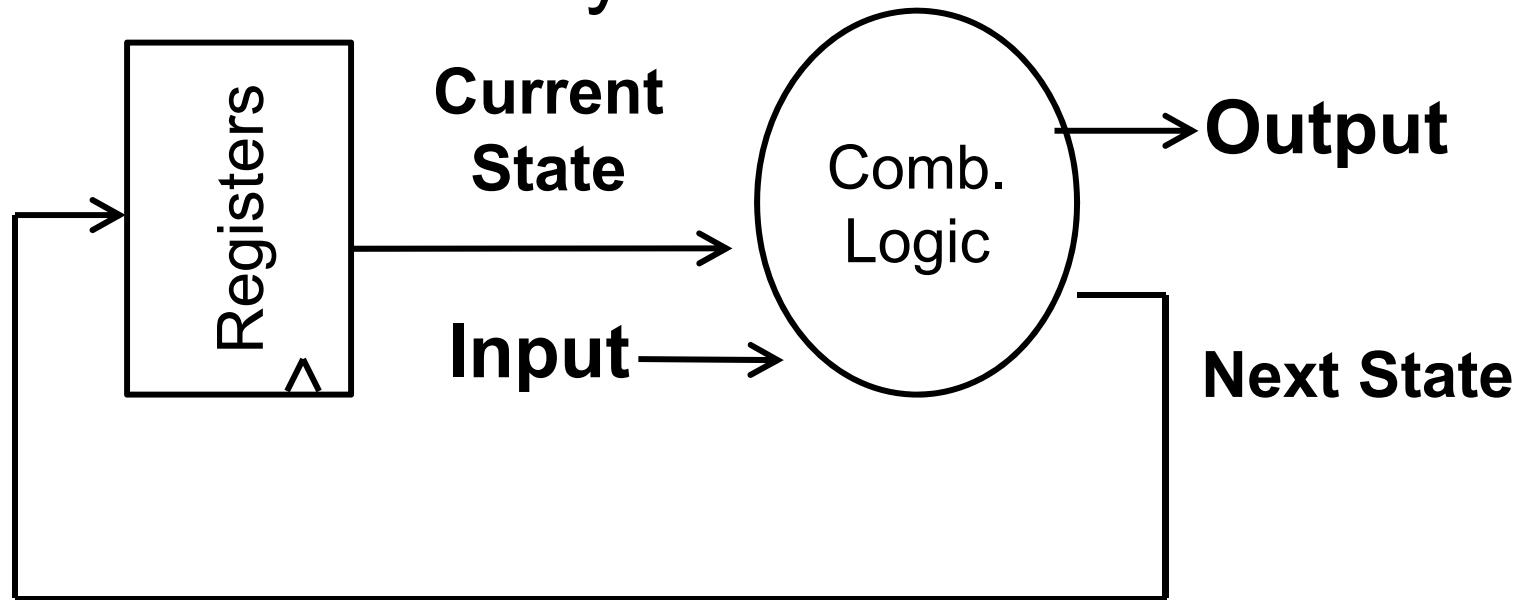
(C) In a Mealy Machine output depends on next state and input

(D) All the above are true

(E) None are true

Mealy Machine

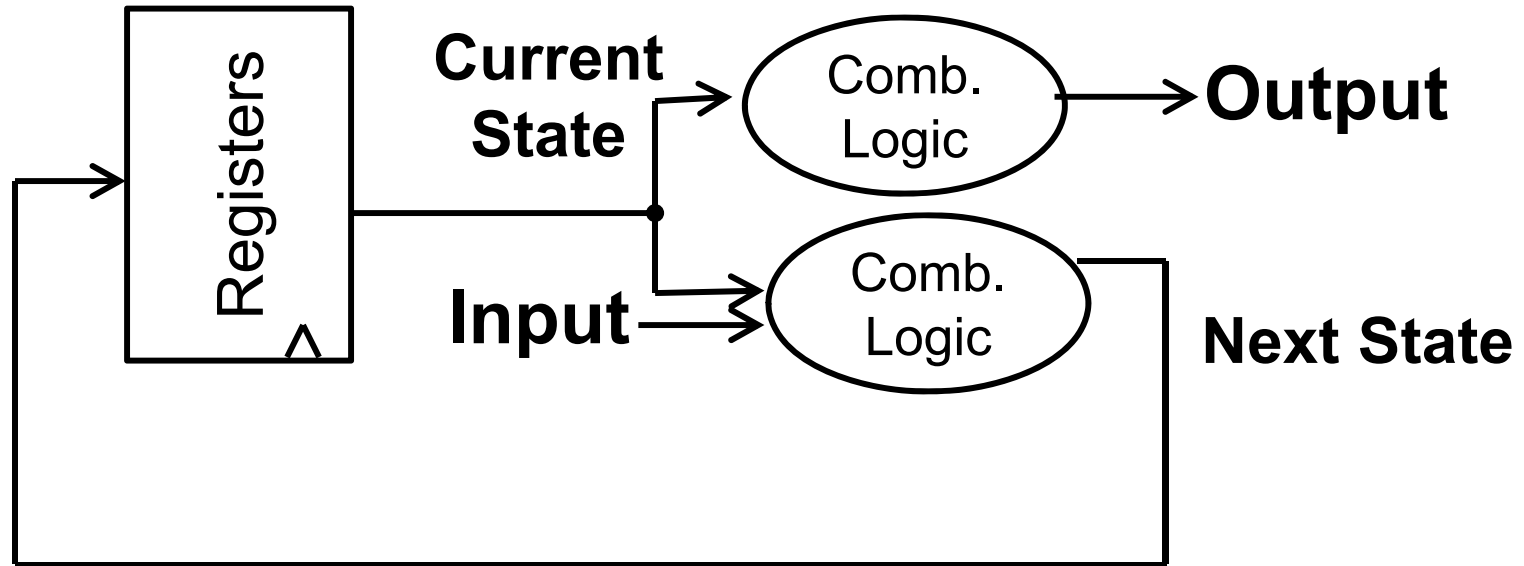
General Case: Mealy Machine



Outputs and next state depend on both current state and input

Moore Machine

Special Case: Moore Machine



Outputs depend only on current state

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



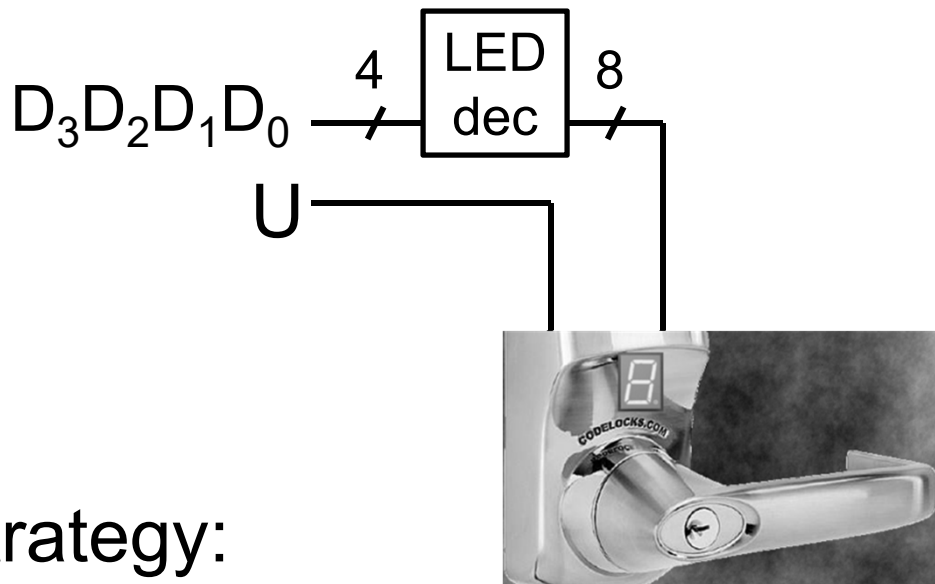
K
A
B

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

Door Lock: Outputs

Assumptions:

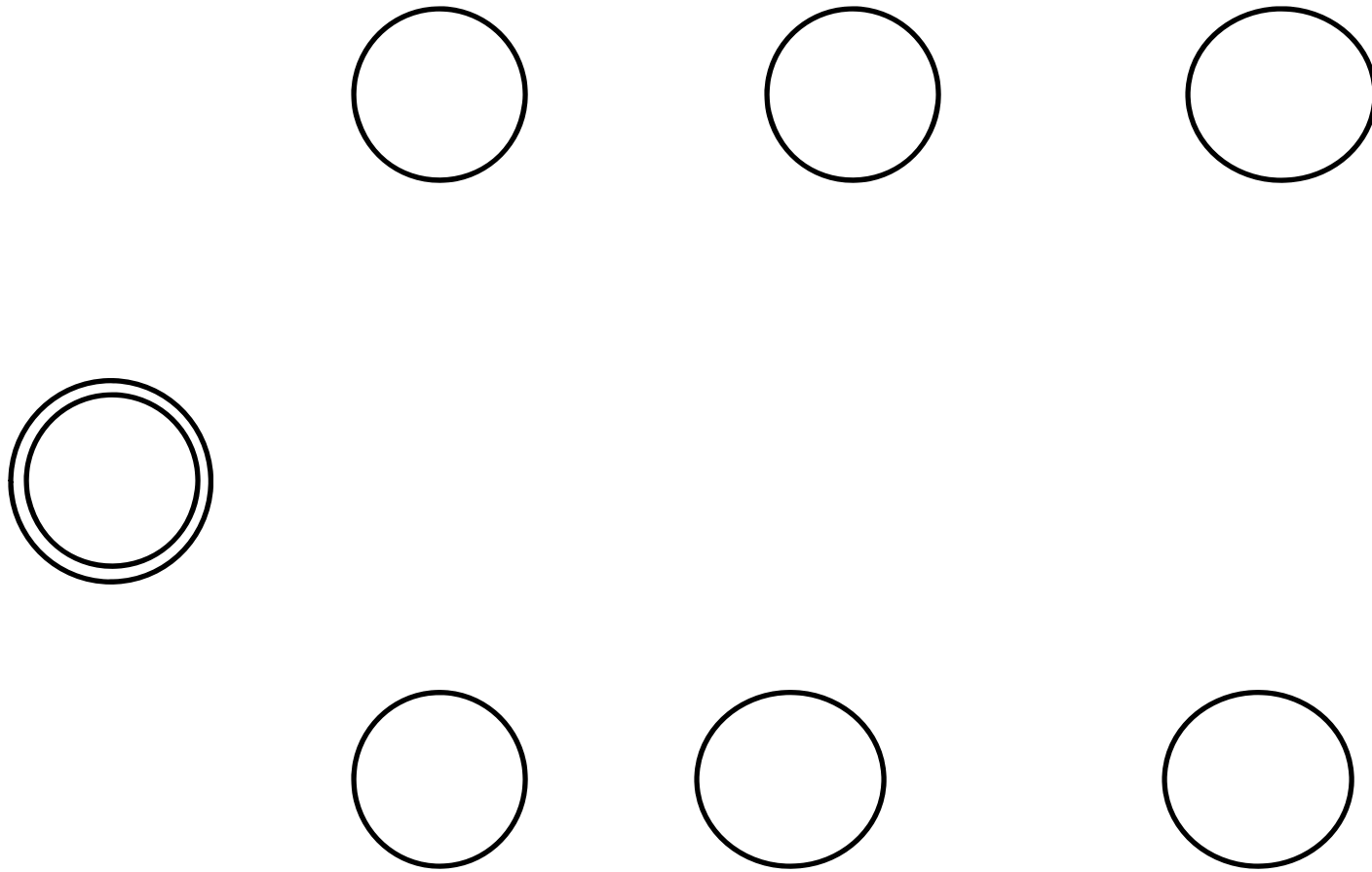
- High pulse on U unlocks door



Strategy:

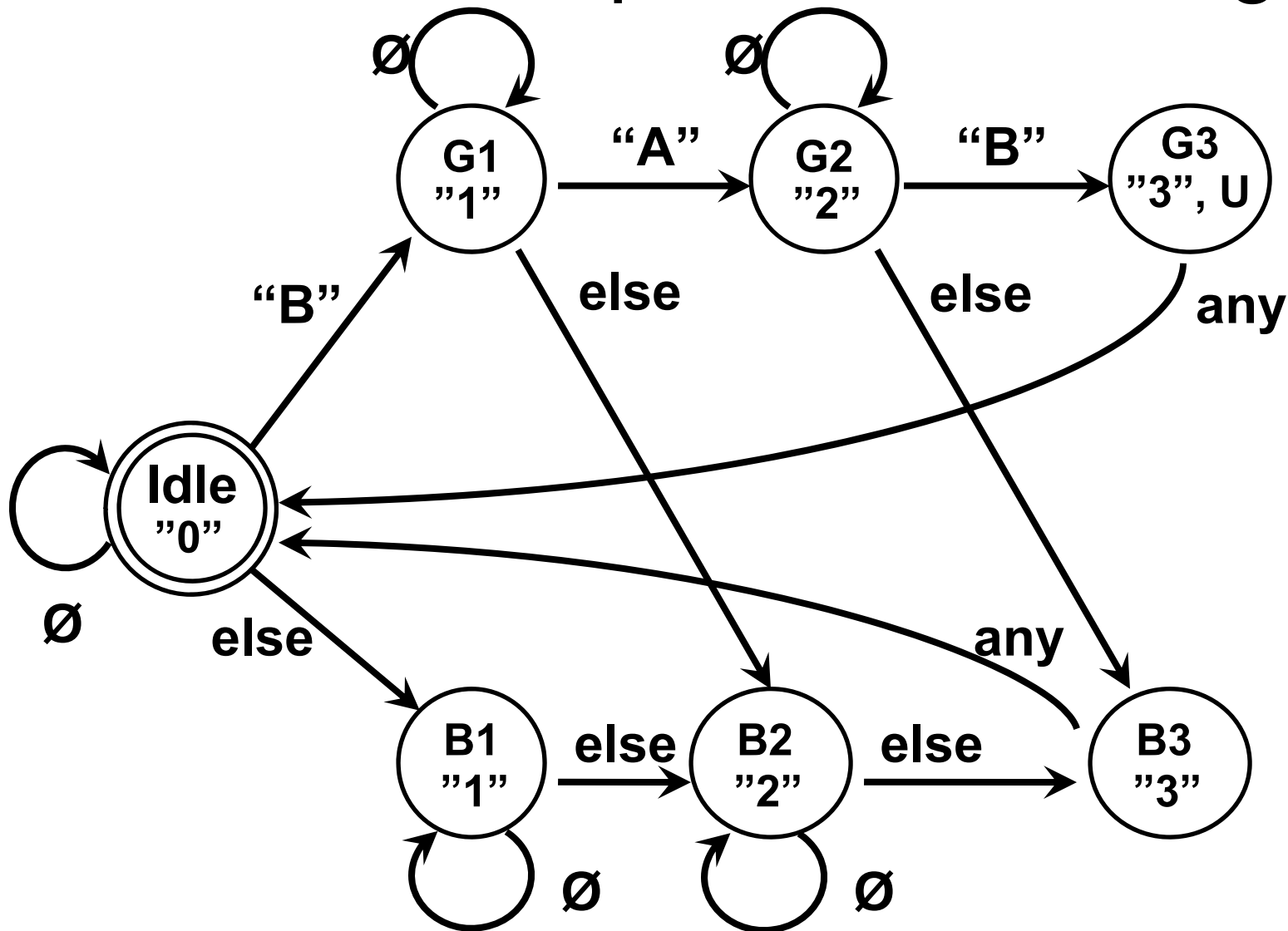
- (1) Draw a state diagram (e.g. Moore Machine)
- (2) Write output and next-state tables
- (3) Encode states, inputs, and outputs as bits
- (4) Determine logic equations for next state and outputs

Door Lock: Simplified State Diagram



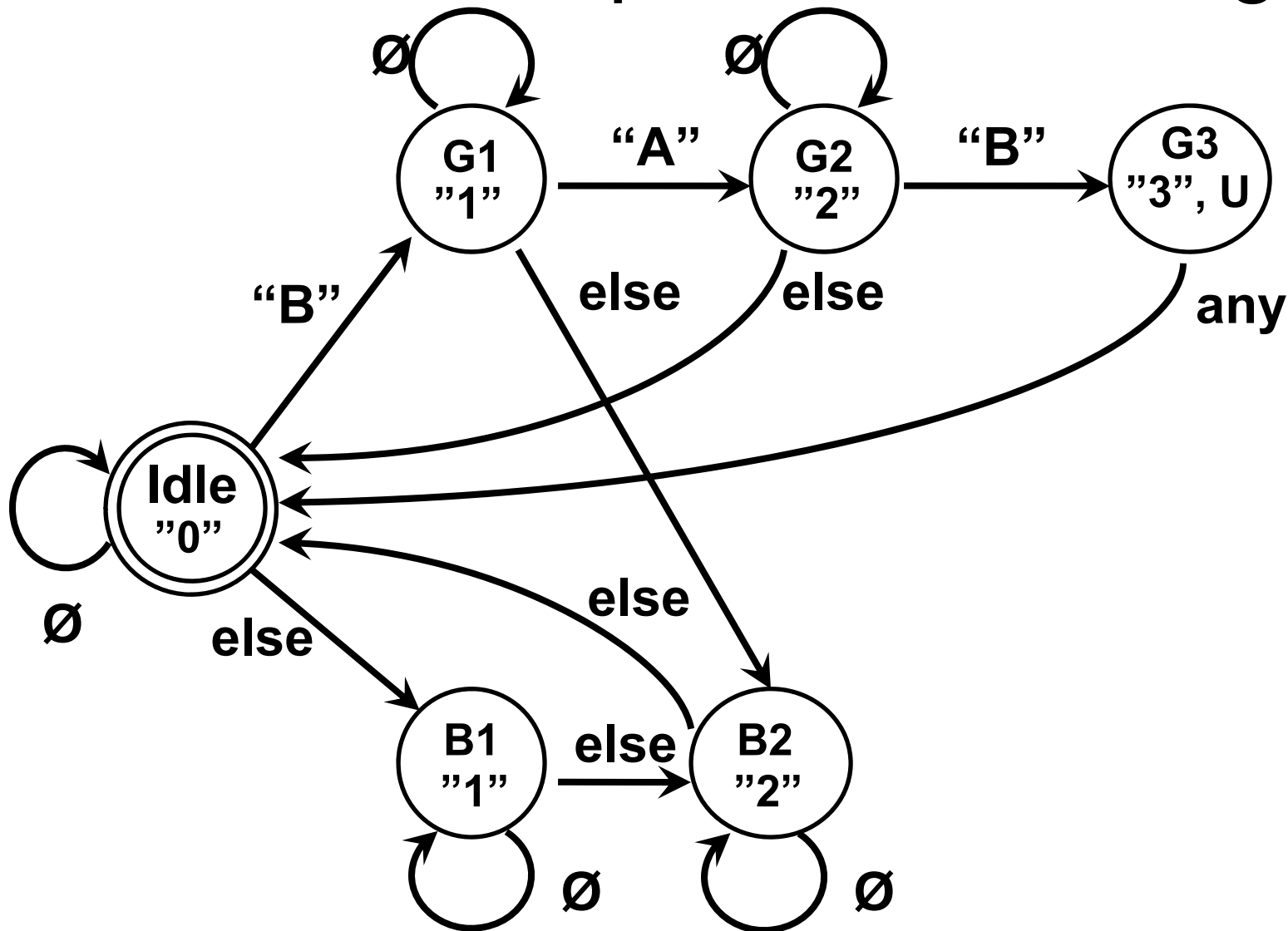
(1) Draw a state diagram (e.g. Moore Machine)

Door Lock: Simplified State Diagram



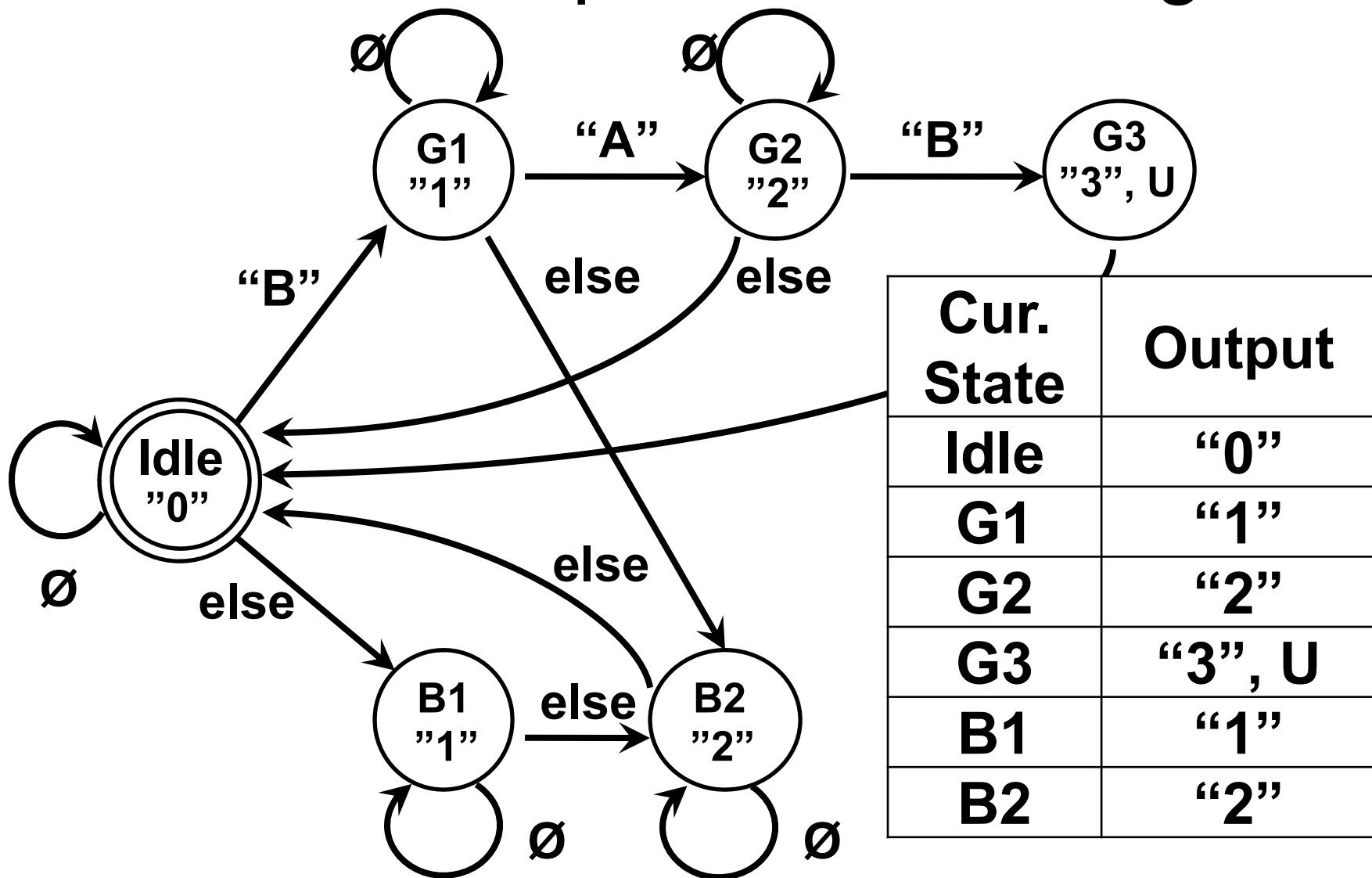
(1) Draw a state diagram (e.g. Moore Machine)

Door Lock: Simplified State Diagram



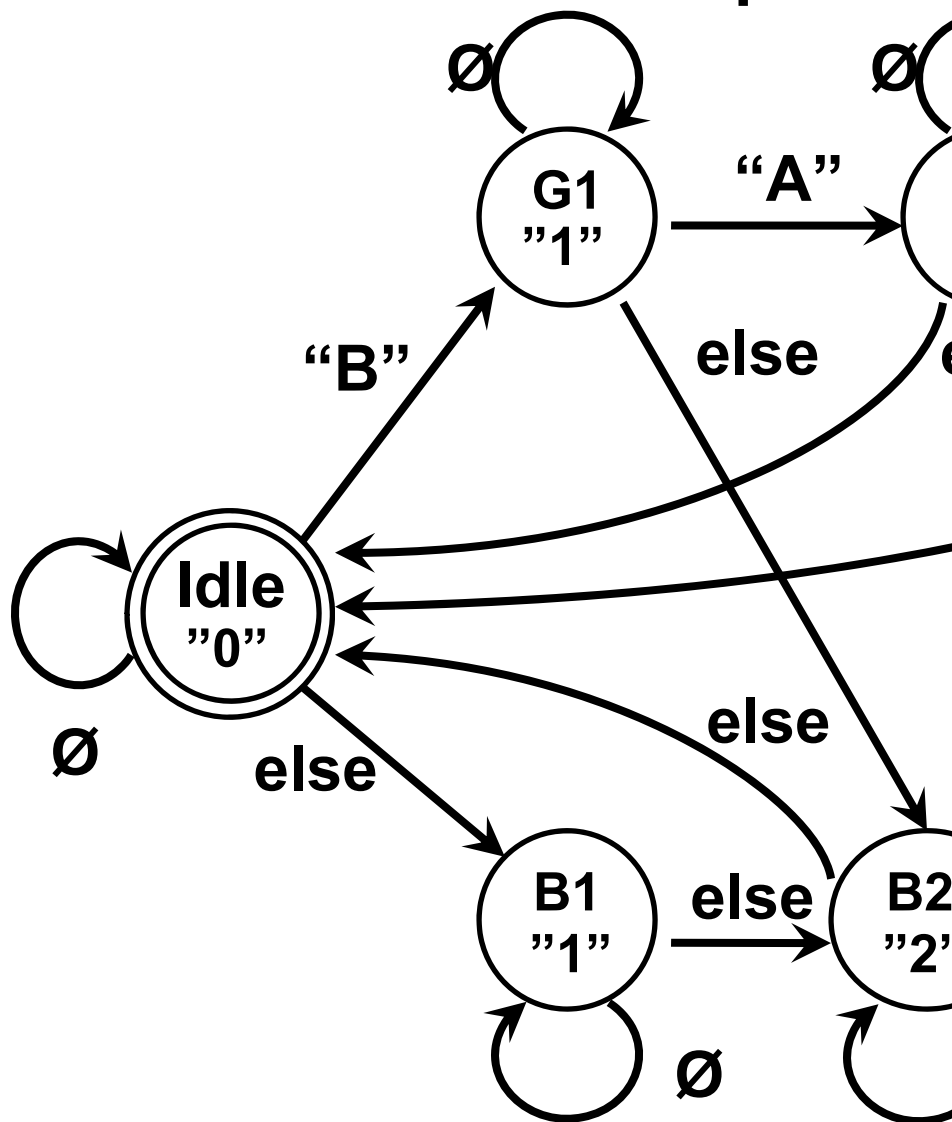
(1) Draw a state diagram (e.g. Moore Machine)

Door Lock: Simplified State Diagram



(2) Write output and next-state tables

Door Lock: Simplified State Diagram



Cur. State	Input	Next State
Idle	Ø	Idle
Idle	"B"	G1
Idle	"A"	B1
G1	Ø	G1
G1	"A"	G2
G1	"B"	B2
G2	Ø	G2
G2	"B"	G3
G2	"A"	Idle
G3	any	Idle
B1	Ø	B1
B1	K	B2
B2	Ø	B2
B2	K	Idle

(2) Write output and next-s

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

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

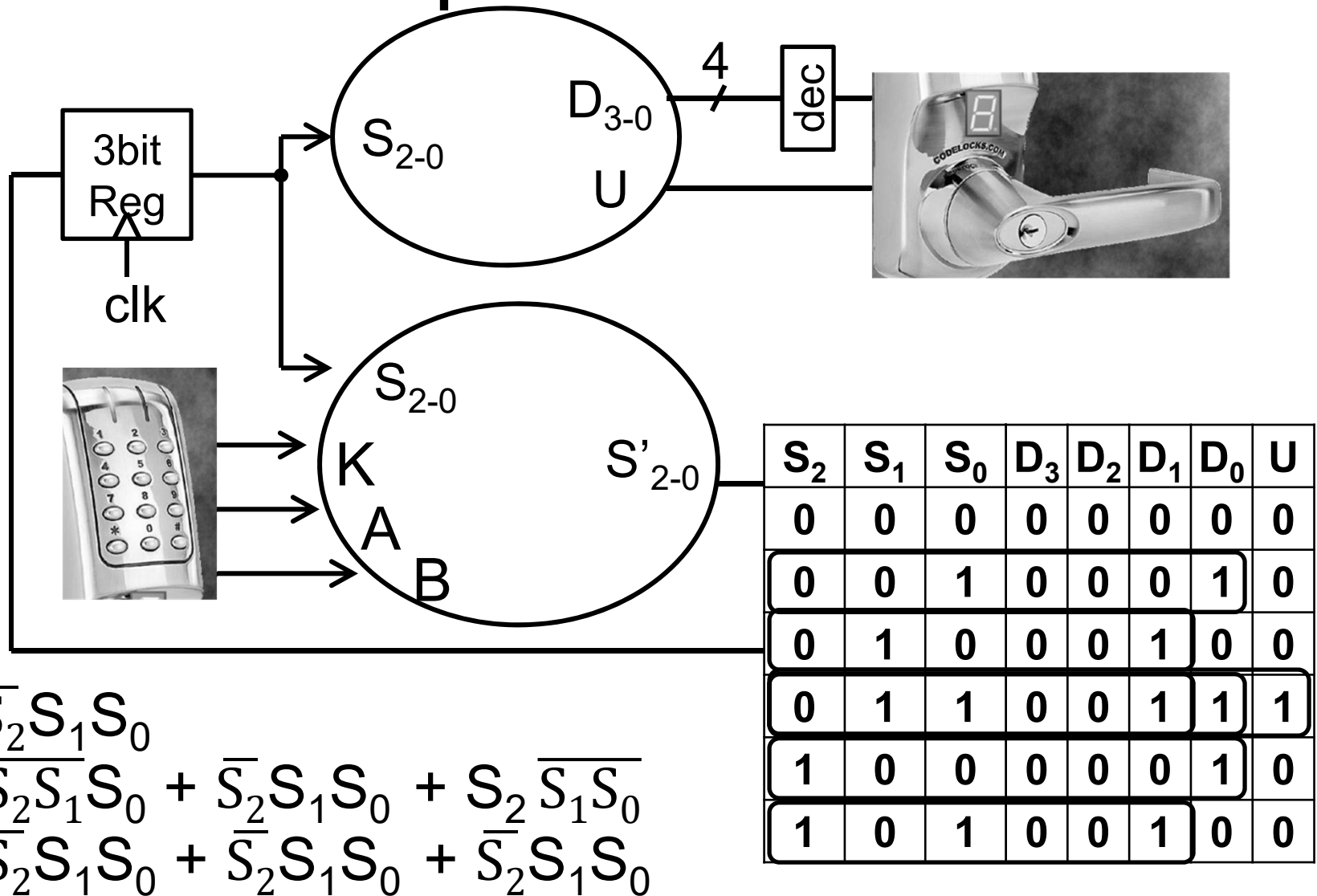
S ₂	S ₁	S ₀	K	A	B	S' ₂	S' ₁	S' ₀
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	1
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	1
0	0	1	1	1	0	0	1	0
0	0	1	1	0	1	1	0	1
0	1	0	0	0	0	0	1	0
0	1	0	1	0	1	0	1	1
0	1	0	1	1	0	0	0	0
0	1	1	x	x	x	0	0	0
1	0	0	0	0	0	1	0	0
1	0	0	1	x	x	1	0	1
1	0	1	0	0	0	1	0	1
1	0	1	1	x	x	0	0	0

(3

s, and outputs as bits

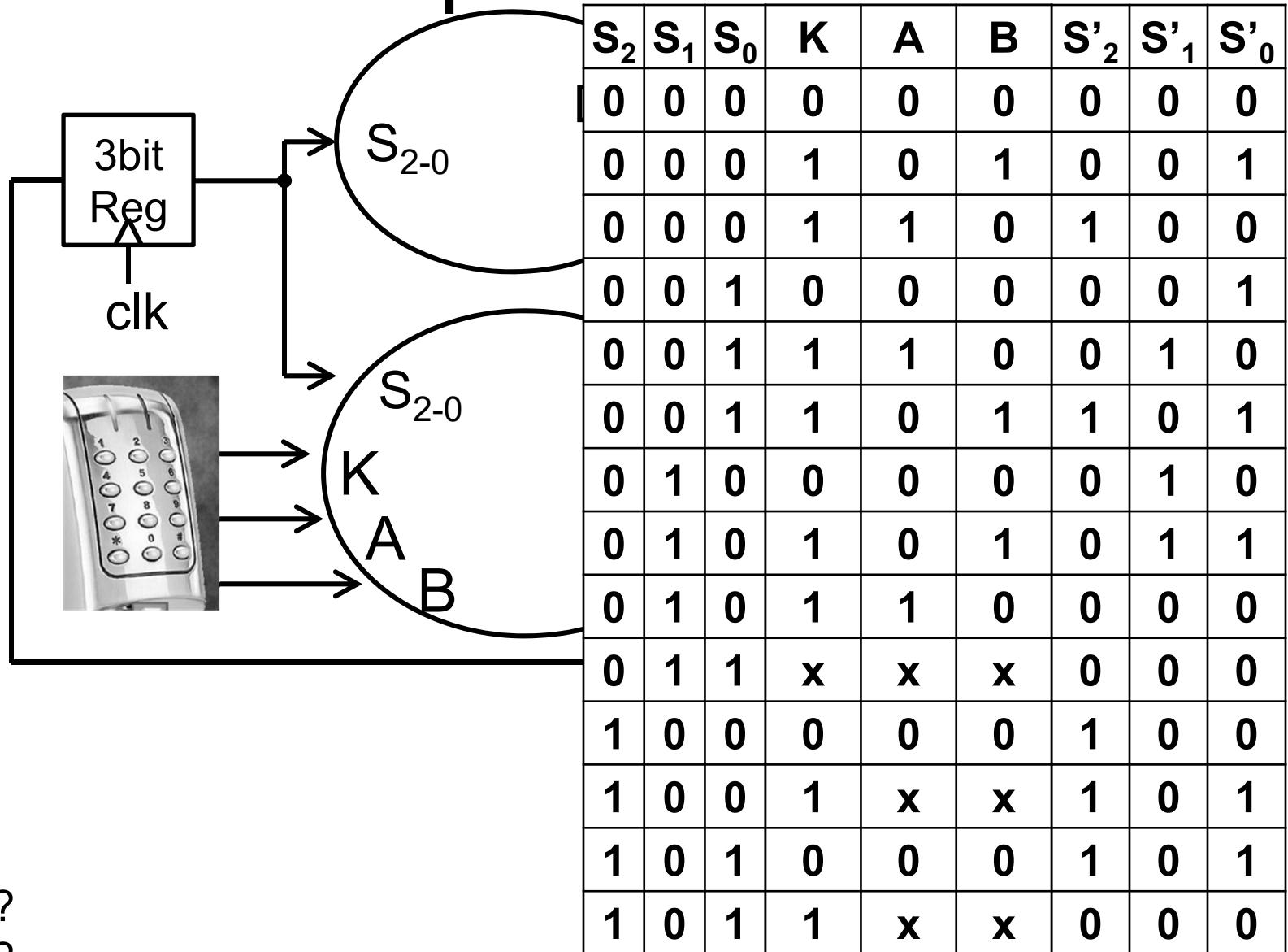
Idle

Door Lock: Implementation



(4) Determine logic equations for next state and outputs

Door Lock: Implementation

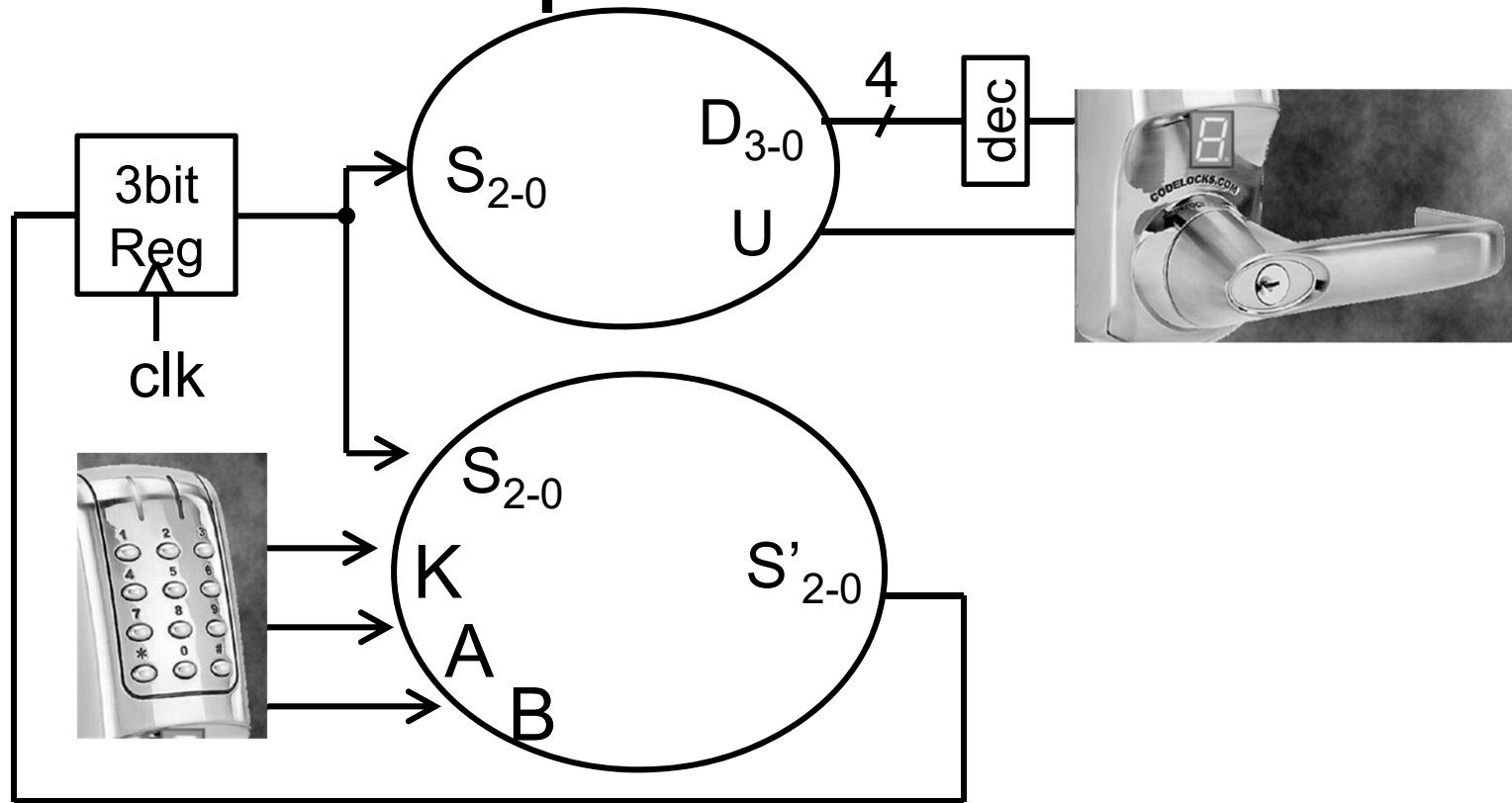


$S'_0 = ?$

$S'_1 = ?$

$$S'_2 = \overline{S_2} \overline{S_1} \overline{S_0} K A \overline{B} + \overline{S_2} \overline{S_1} \overline{S_0} K \overline{A} B + S_2 \overline{S_1} \overline{S_0} K A B + \overline{S_2} S_1 \overline{S_0} K + S_2 \overline{S_1} \overline{S_0} \overline{K A B}$$

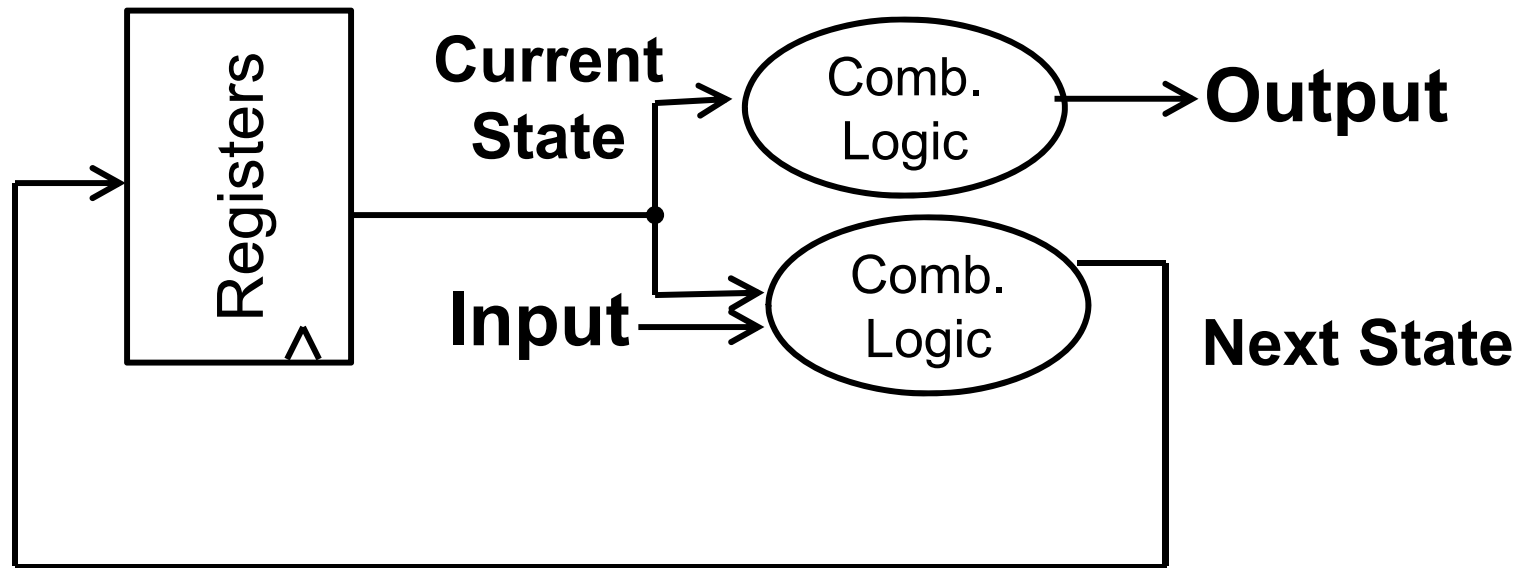
Door Lock: Implementation



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

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, ...)
- State Machines or Ad-Hoc Circuits

