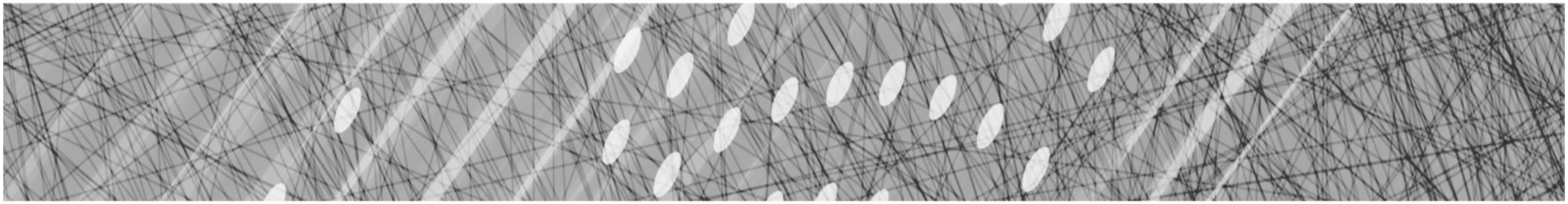


Finite State Machines

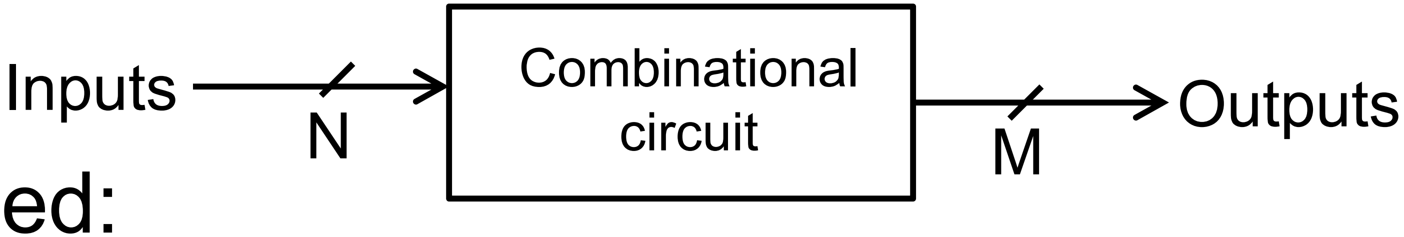
Hakim Weatherspoon
CS 3410
Computer Science
Cornell University

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

Stateful Components

Combinationial 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

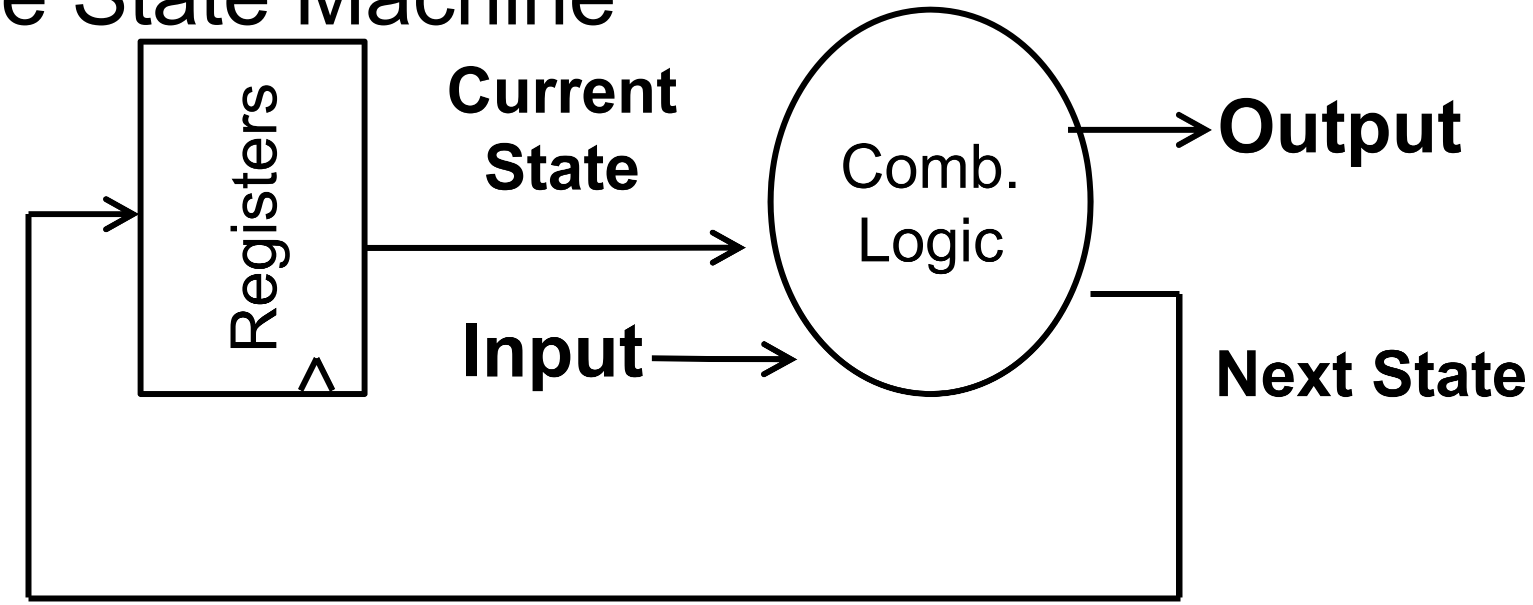
O: Finite set of outputs

δ: State transition function

Next state depends on present input *and* present state

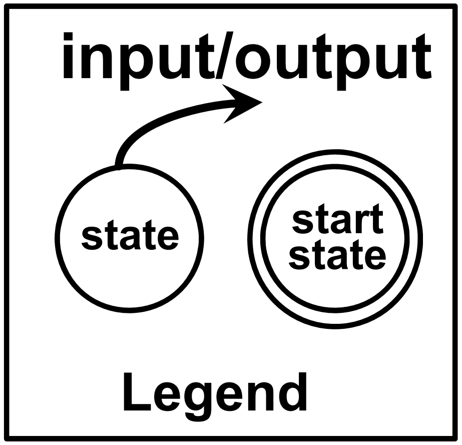
Automata Model

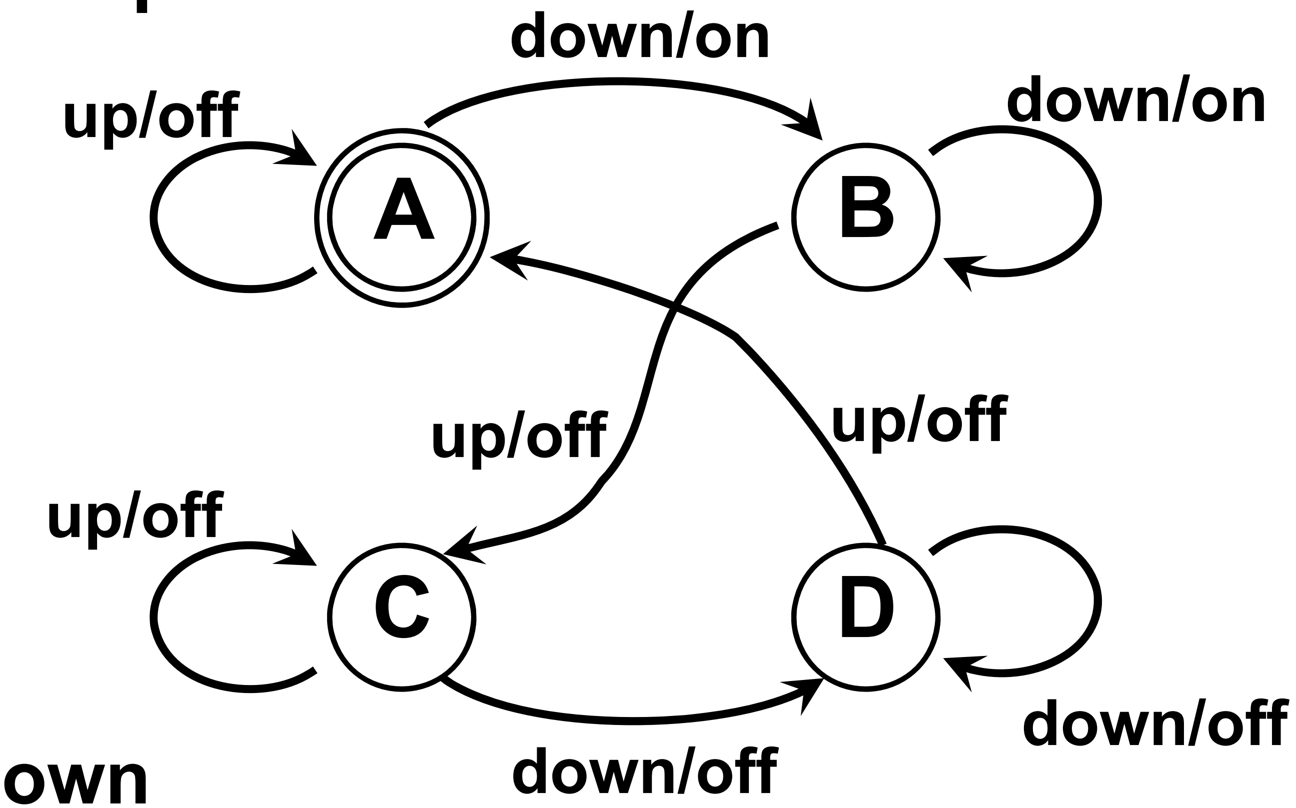
Finite State Machine



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

FSM Example



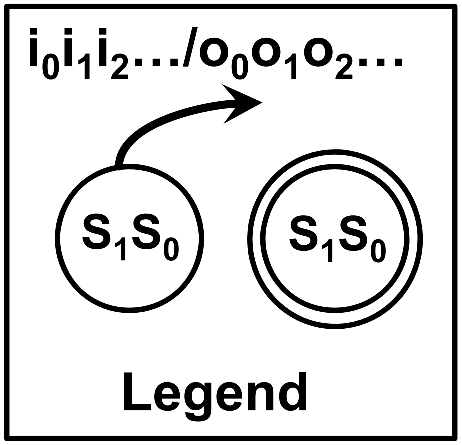


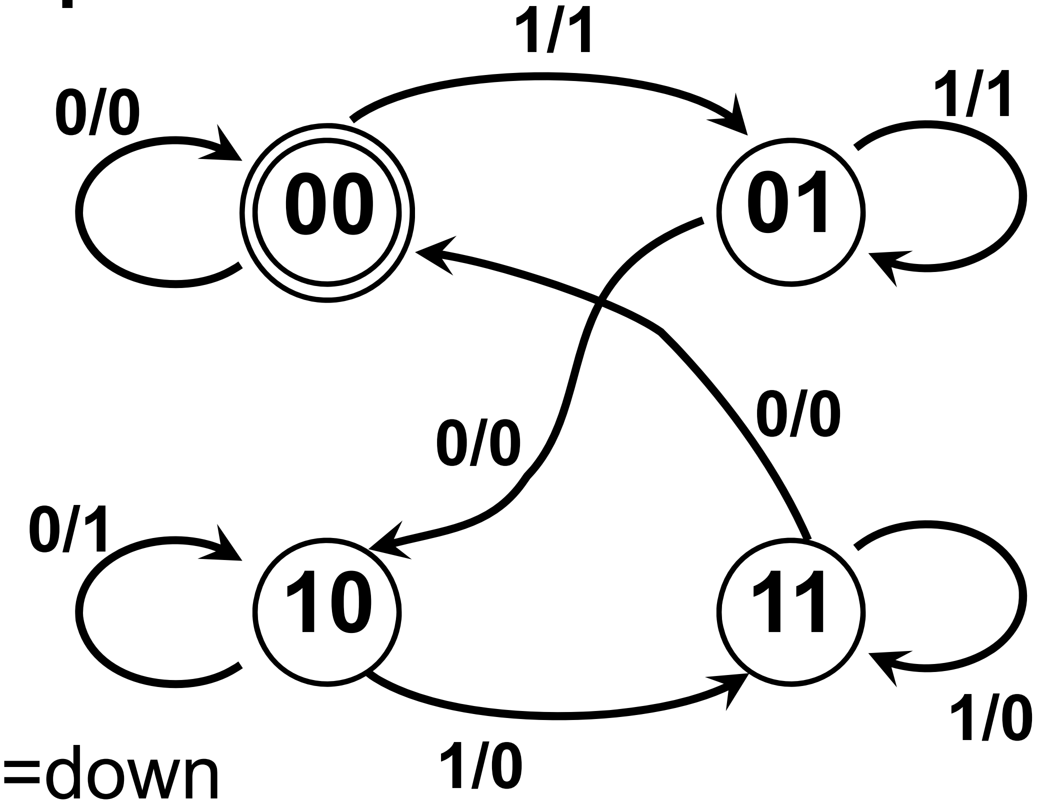
Input: **up** or **down**

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

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

FSM Example





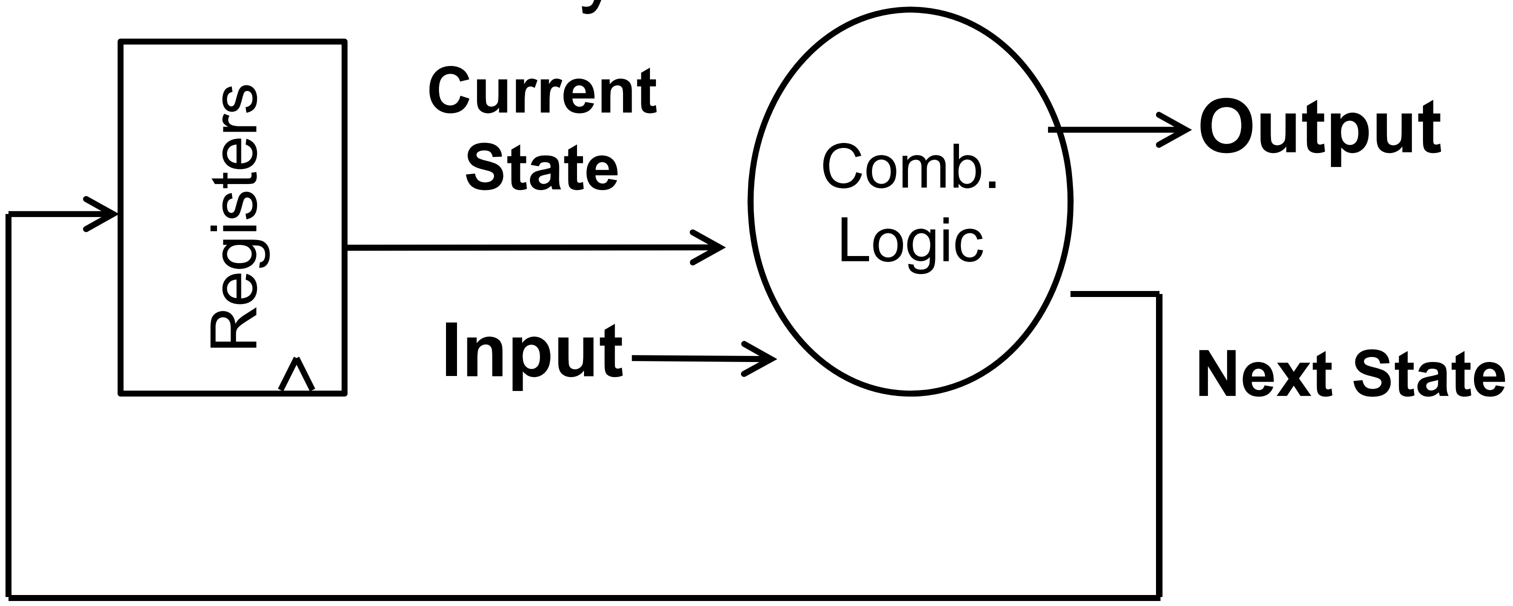
Input: **0**=up or **1**=down

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

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

Mealy Machine

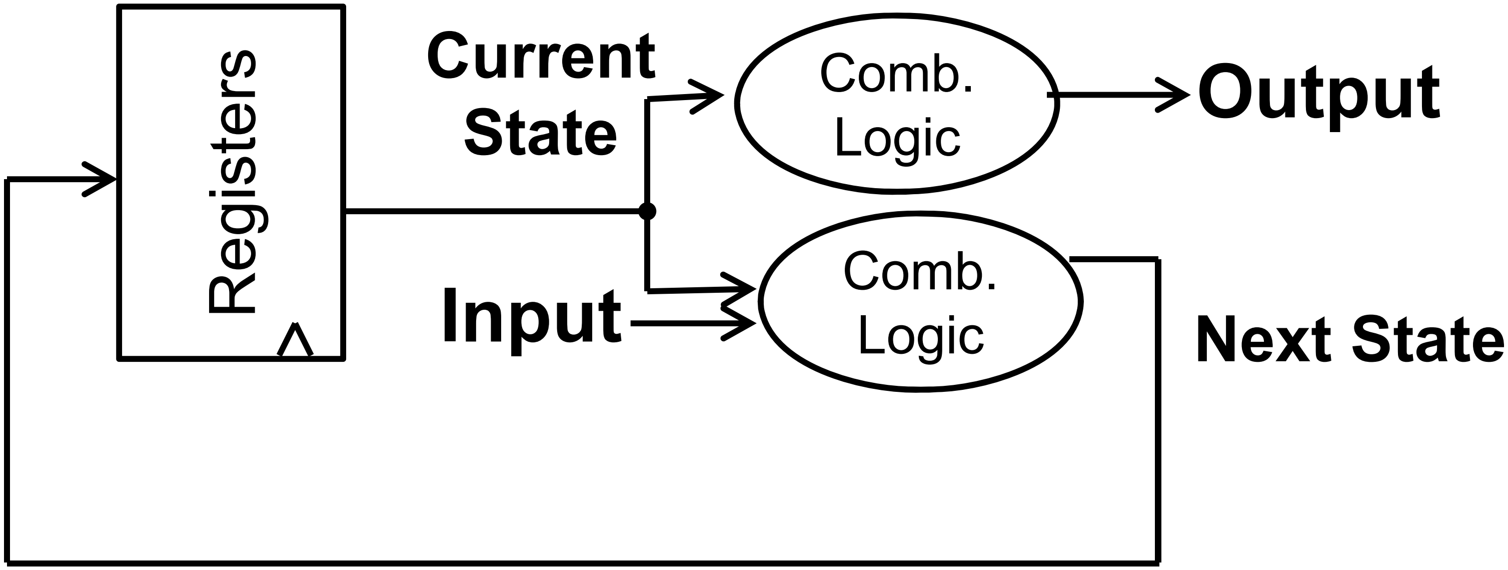
General Case: Mealy Machine



Outputs and next state depend on both current state and input

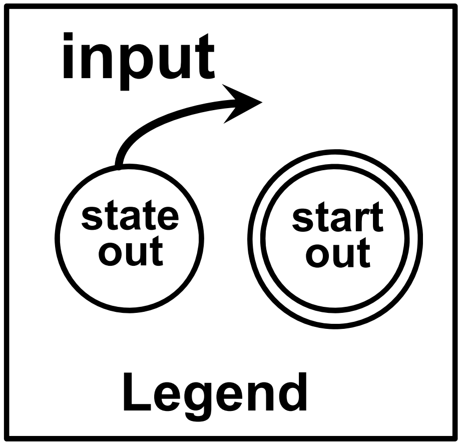
Moore Machine

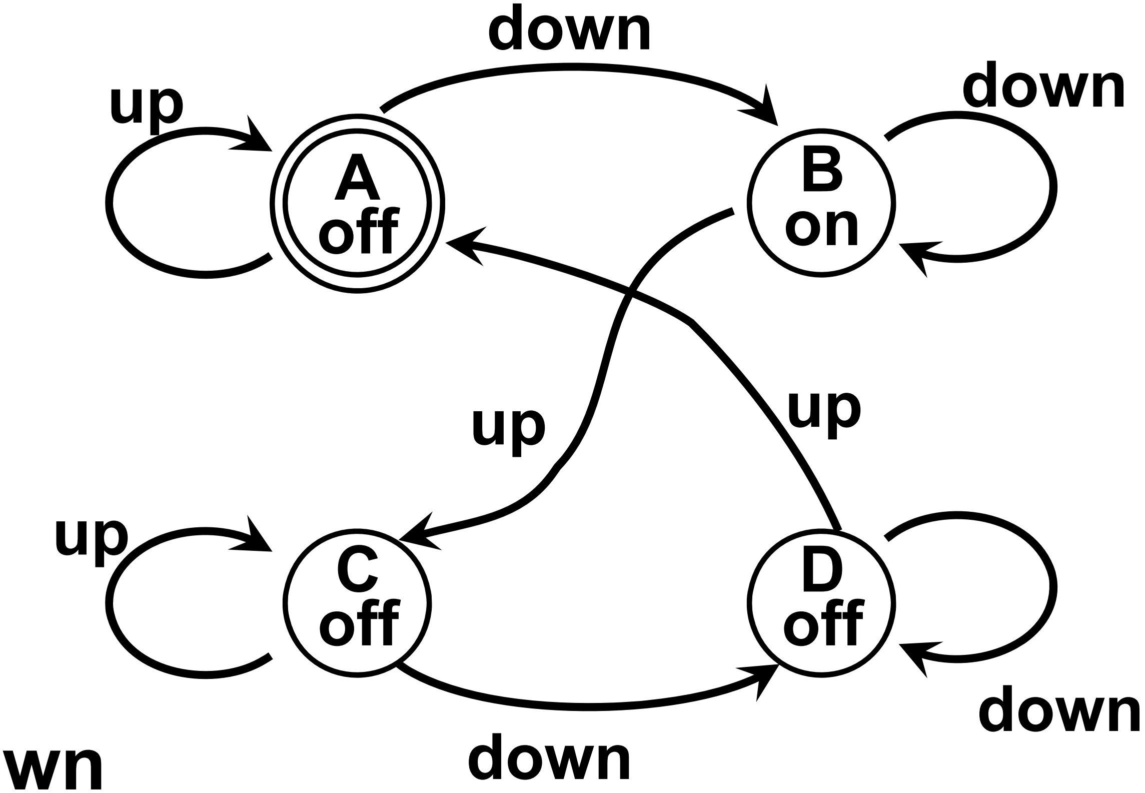
Special Case: Moore Machine



Outputs depend only on current state

Moore Machine FSM Example



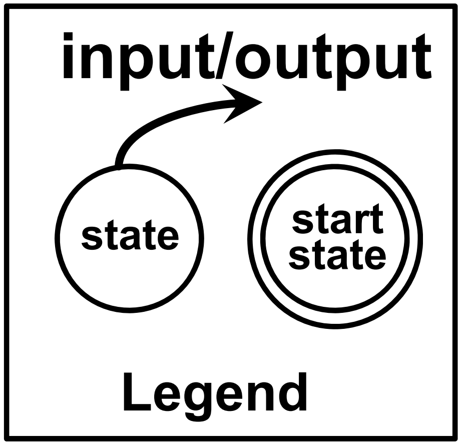


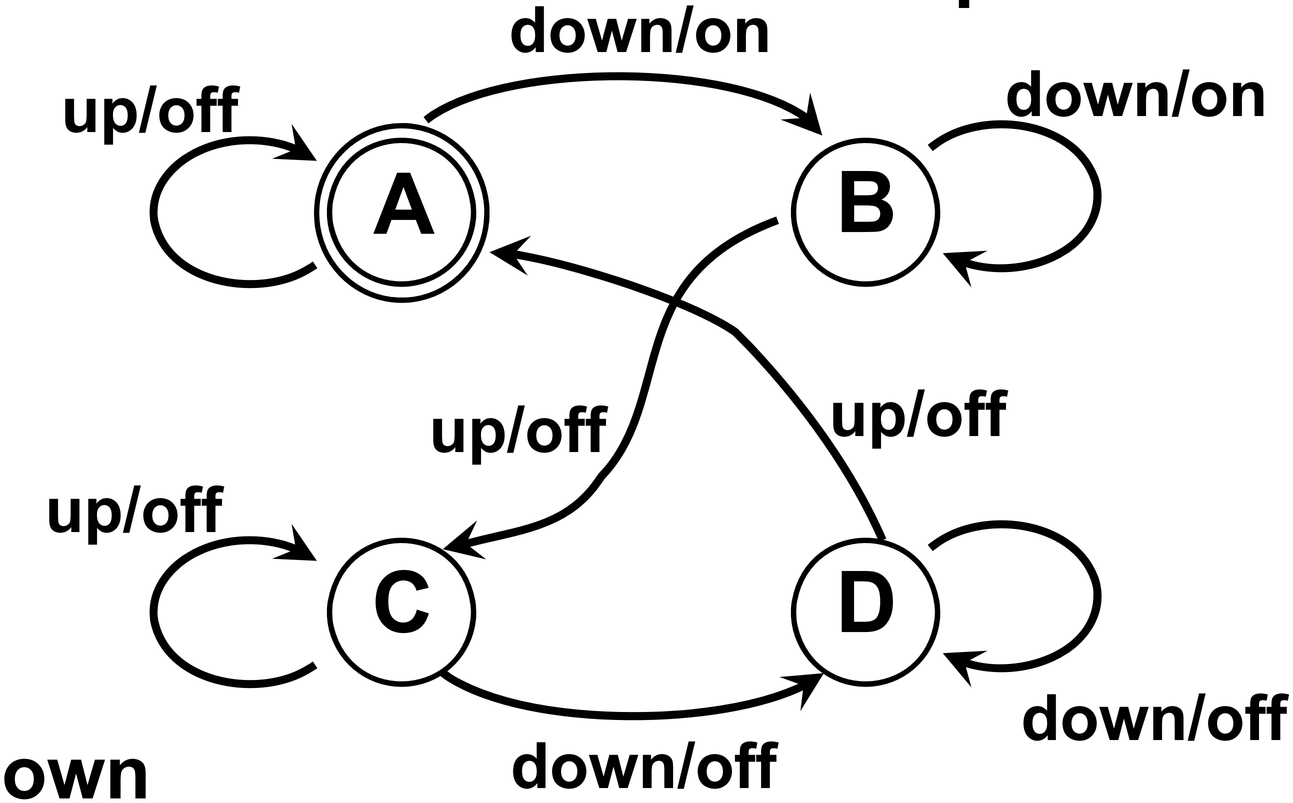
Input: **up** or **down**

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

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

Mealy Machine FSM Example



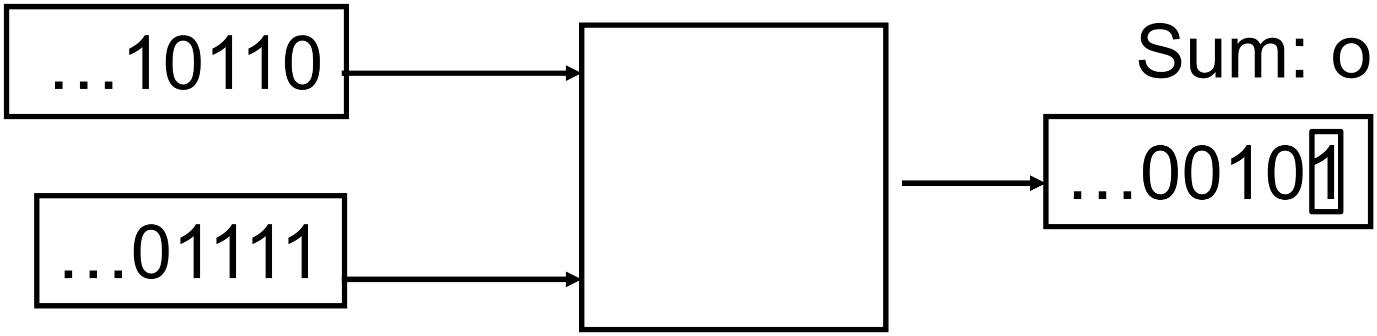


Input: **up** or **down**
Output: **on** or **off**
States: **A**, **B**, **C**, or **D**

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
- 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

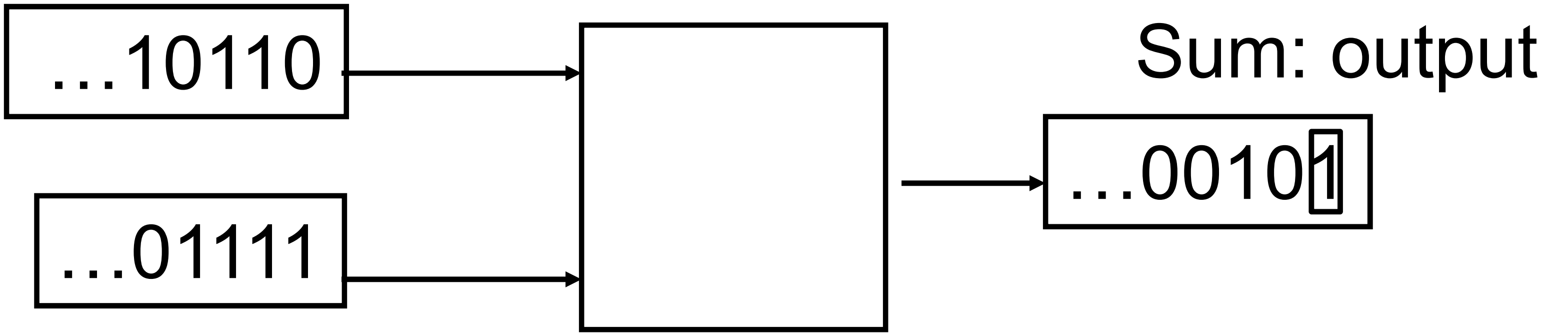
(3) Encode states, inputs, and outputs as bits

(4) Determine logic equations for next state and outputs

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

Add two infinite input bit streams

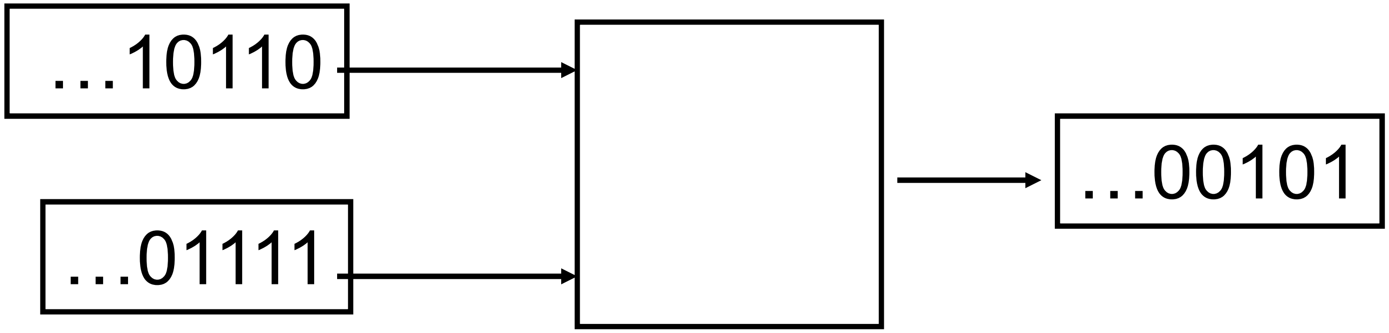
- streams are sent with least-significant-bit (lsb) first



Strategy for Building an FSM

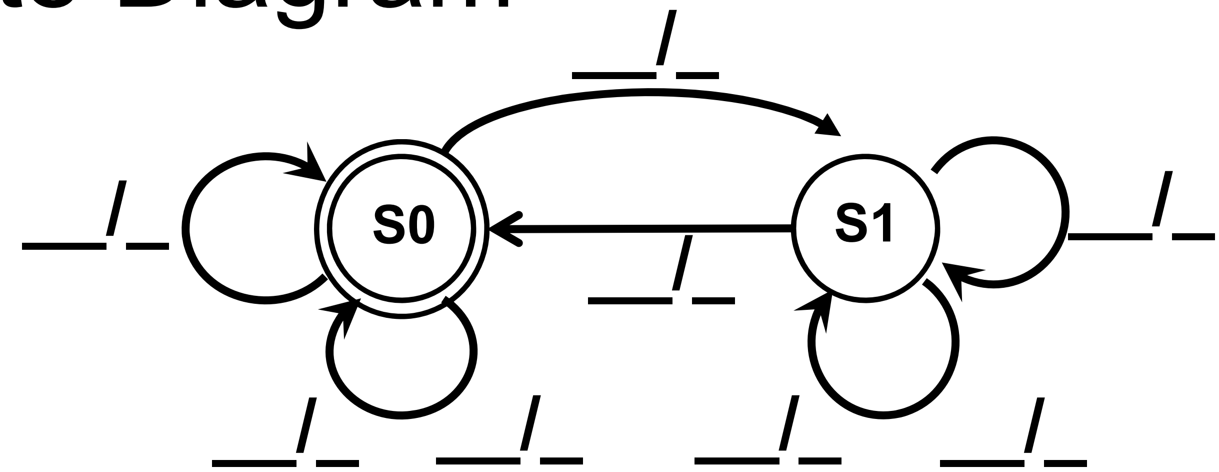
(1) Draw a state diagram (e.g. Mealy Machine)
(2) Write output and next-state tables
(3) Encode states, inputs, and outputs as bits
(4) Determine logic equations for next state and outputs
(5) Draw the Circuit

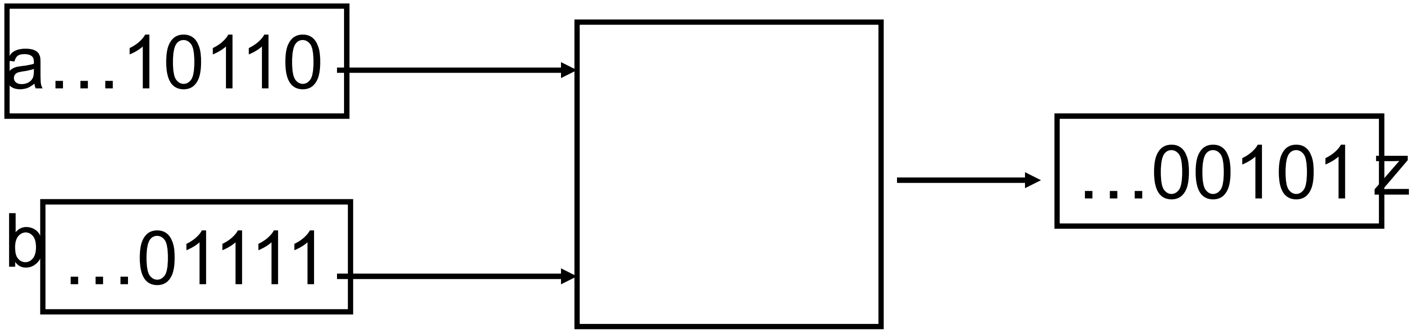
FSM: State Diagram



2 states ___ and ___
Inputs: ___ and ___
Output: ___

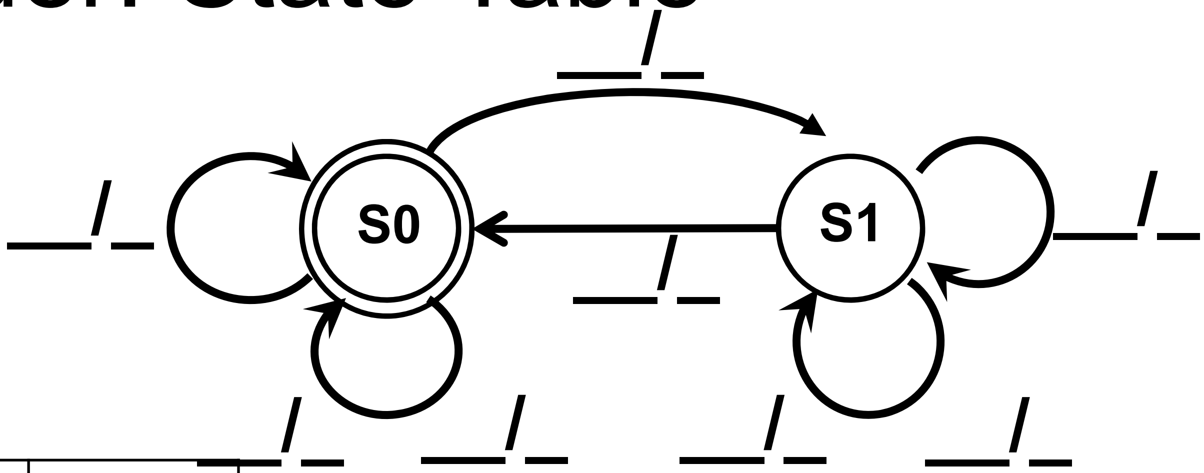
FSM: State Diagram





2 states ___ and ___
Inputs: ___ and ___
Output: ___

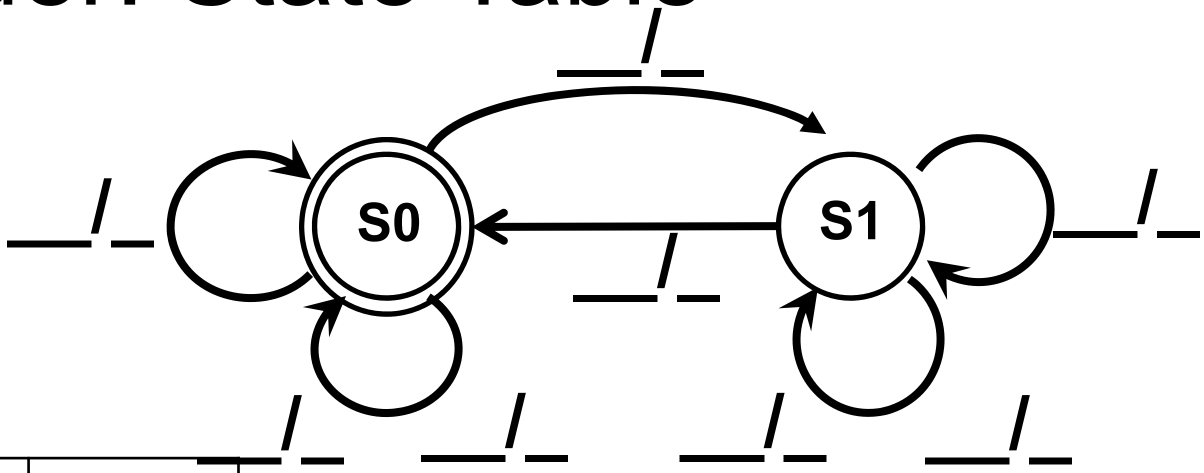
Serial Adder: State Table



a	b	Current state	z	Next state

(2) Write down all input and state combinations

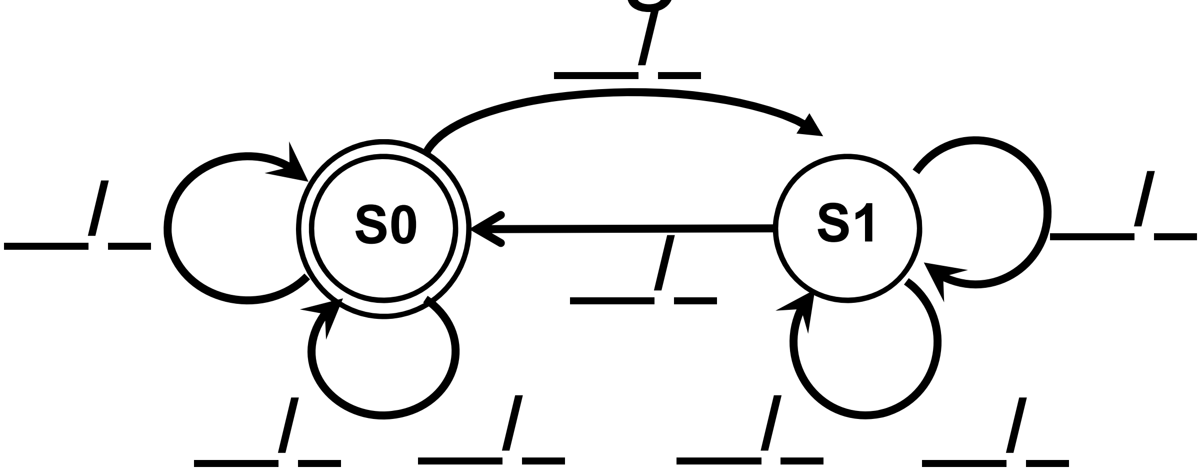
Serial Adder: State Table



a	b	Current state	z	Next state

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

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

(4) Determine logic equations for next state and outputs

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

(4) Determine logic equations for next state and outputs

Example: Digital Door Lock

Digital Door Lock

Inputs:

- keycodes from keypad
- clock

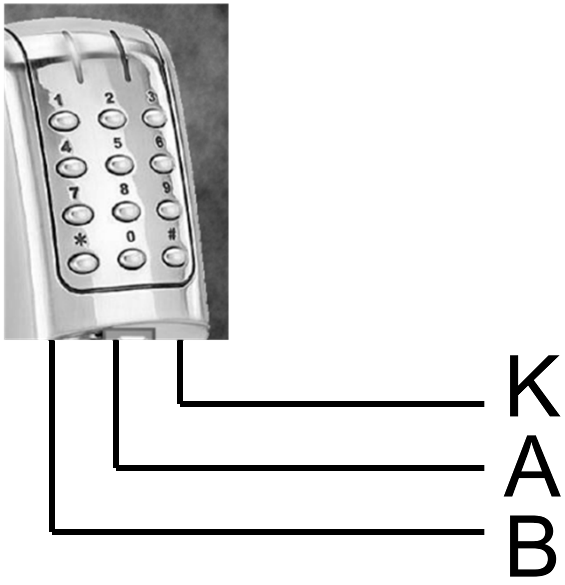
Outputs:

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

Door Lock: Inputs

Assumptions:

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

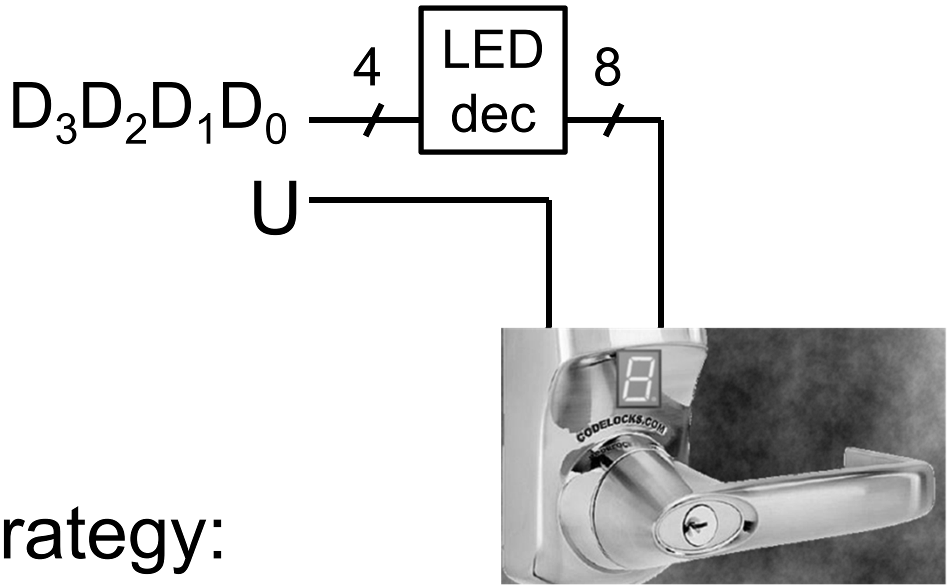


K	A	B	Meaning
0	0	0	Ø (no key)
1	1	0	‘A’ pressed
1	0	1	‘B’ pressed

Door Lock: Outputs

Assumptions:

- High pulse on U unlocks door



Strategy:

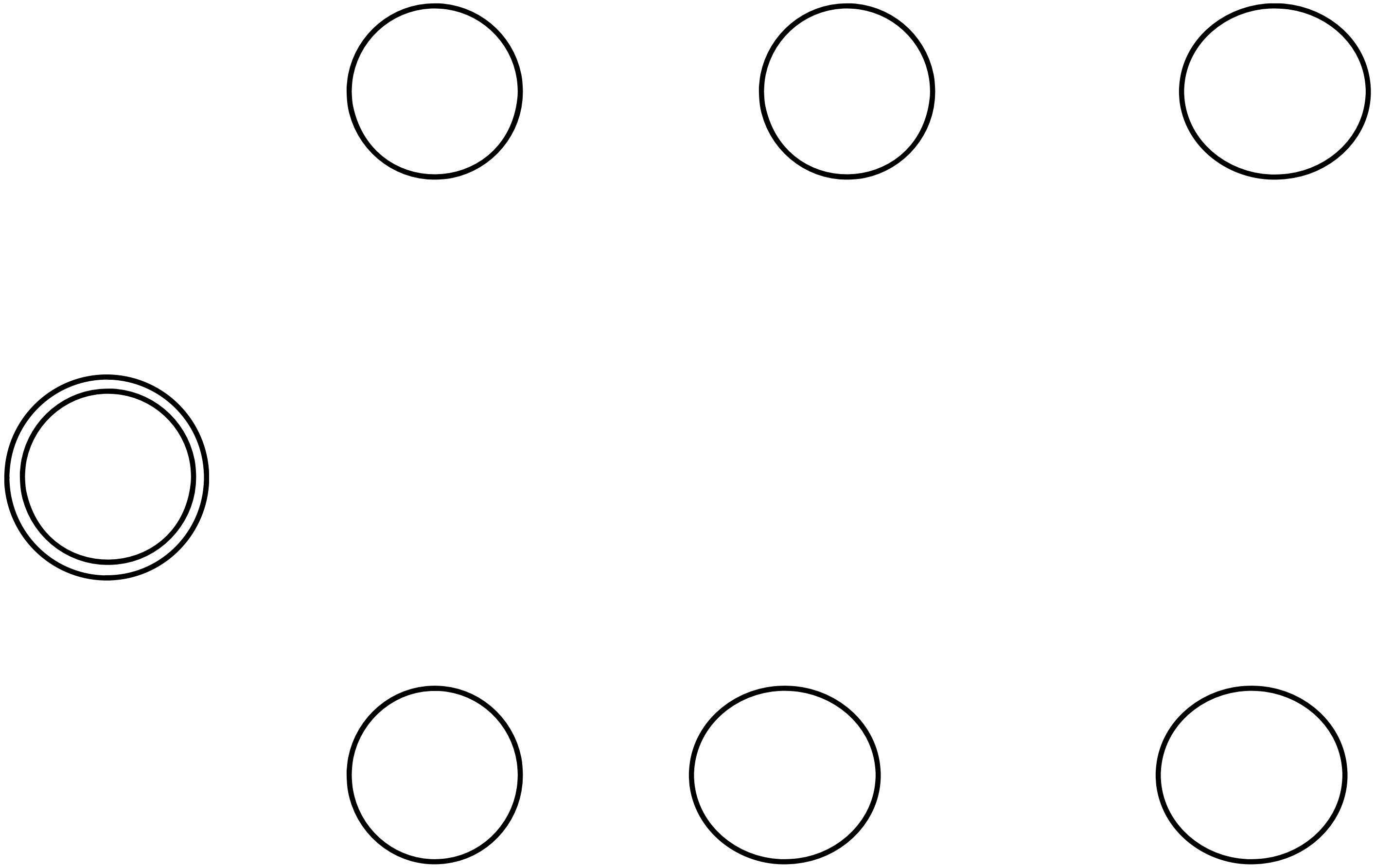
(1) Draw a state diagram (e.g. Moore Machine)

(2) Write output and next-state tables

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

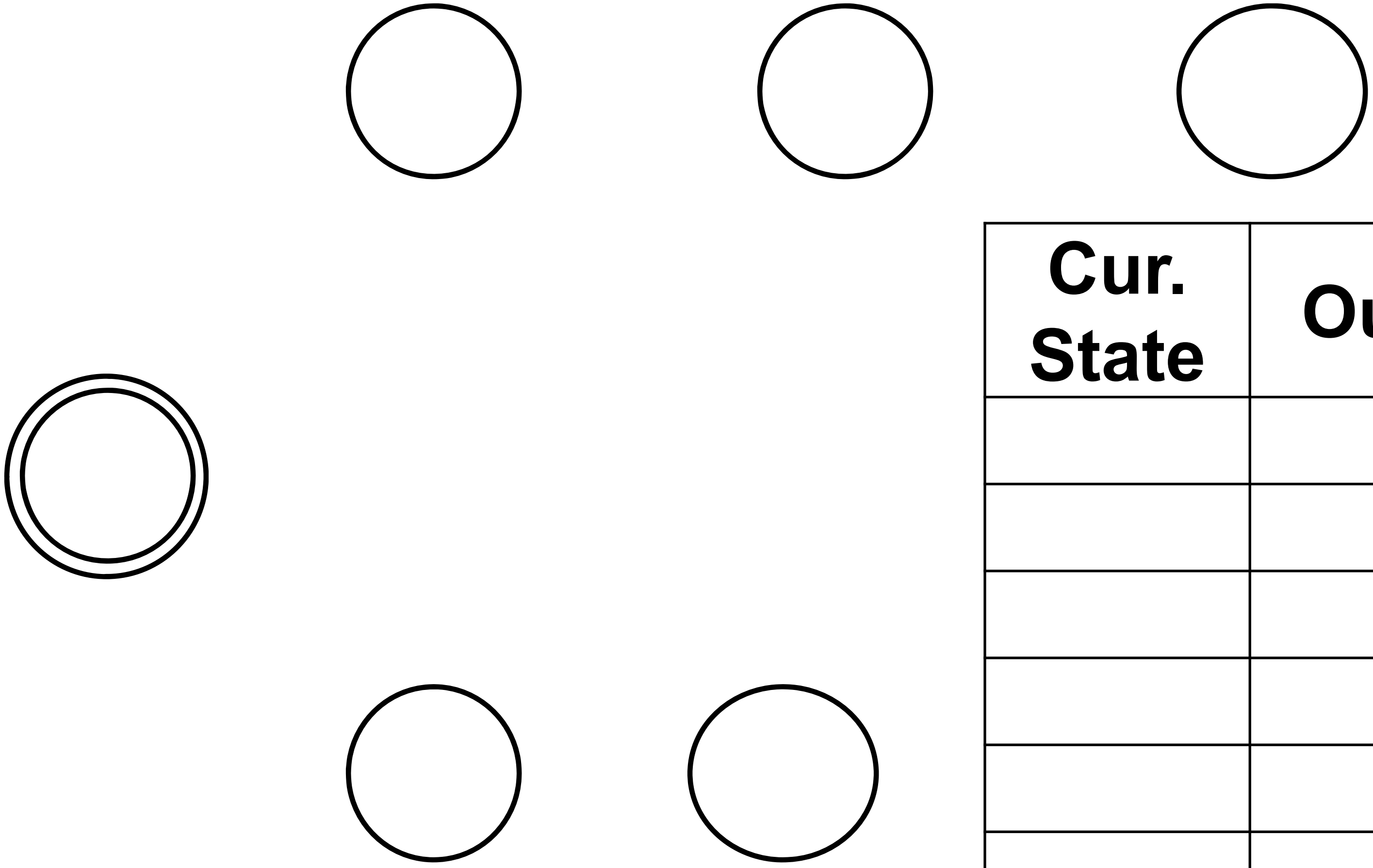
(4) Determine logic equations for next state and outputs

Door Lock: Simplified State Diagram



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

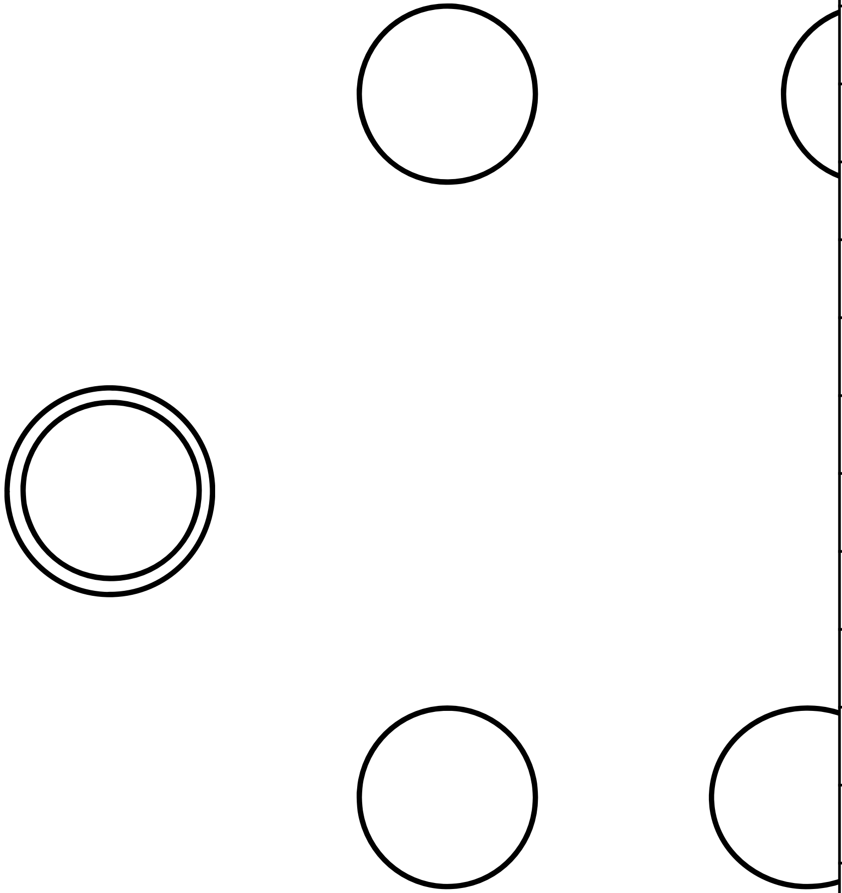
Door Lock: Simplified State Diagram



Cur. State	Output

(2) Write output and next-state tables

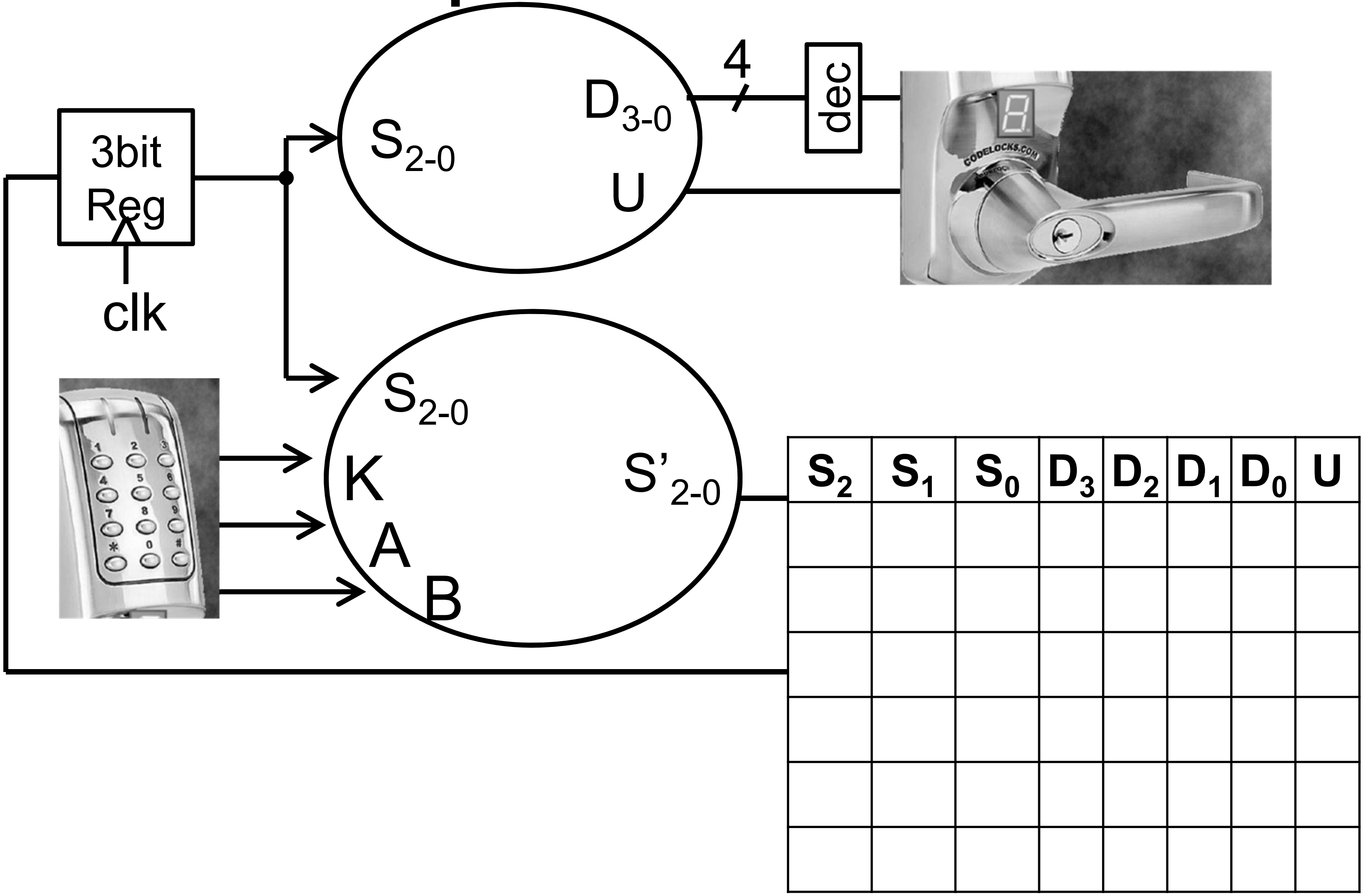
Door Lock: Simplified State Diagram



Cur. State	Input	Next State

(2) Write output and next-s

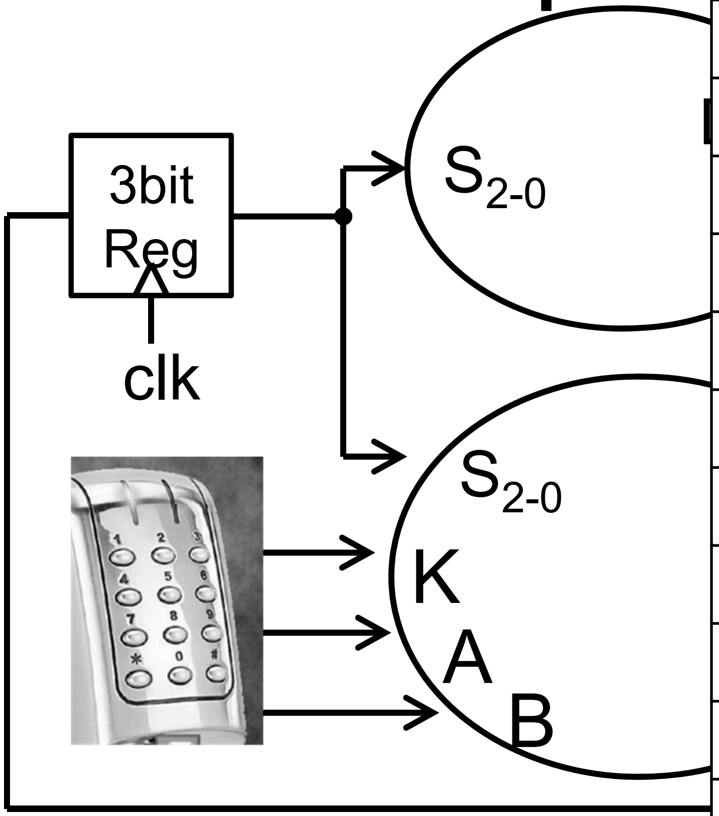
Door Lock: Implementation



S_2	S_1	S_0	D_3	D_2	D_1	D_0	U

(4) Determine logic equations for next state and outputs

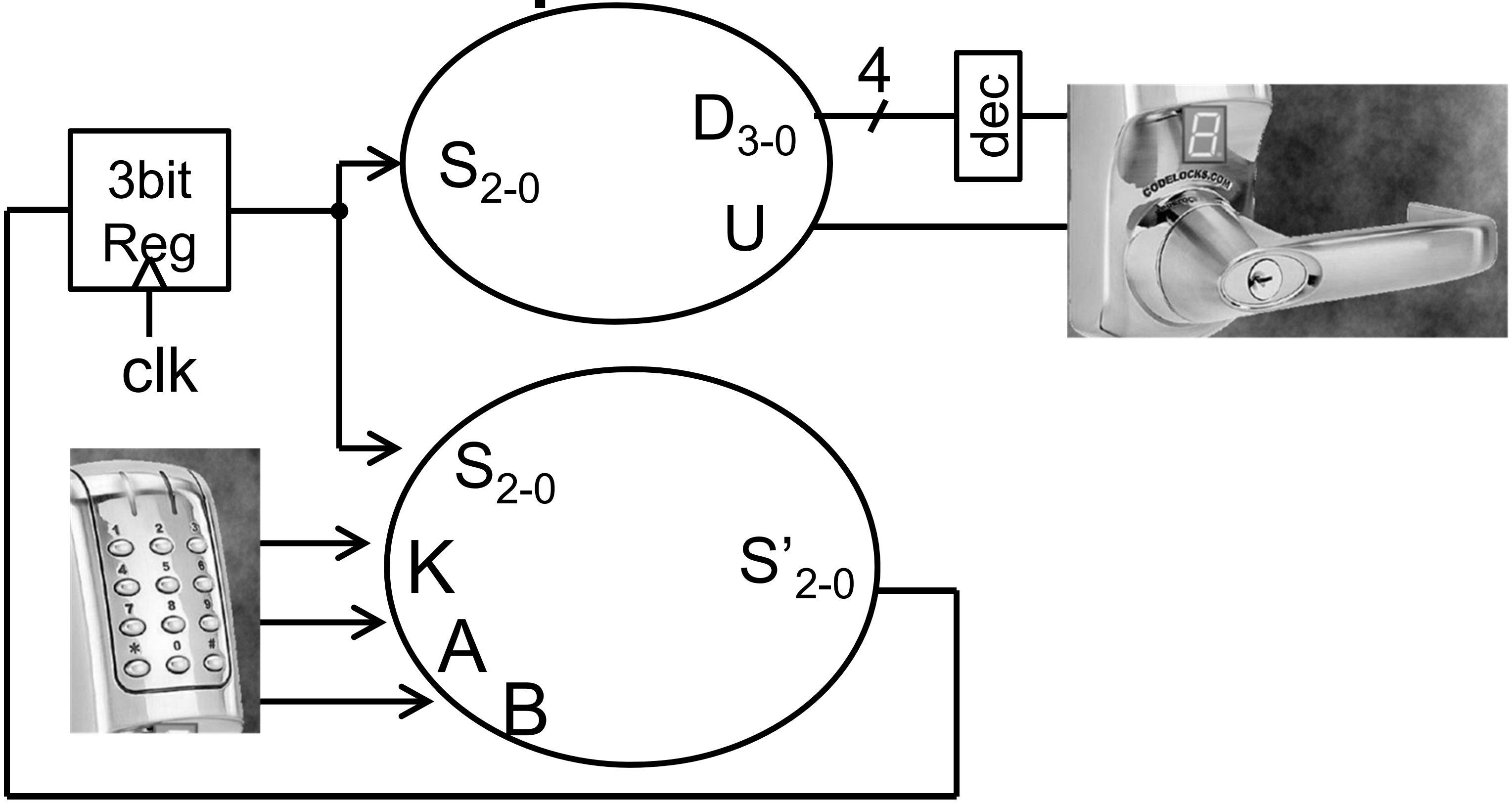
Door Lock: Implementation



S_2	S_1	S_0	K	A	B	S'_2	S'_1	S'_0

(4) Determine logic equations for next state and outputs

Door Lock: Implementation



Strategy:

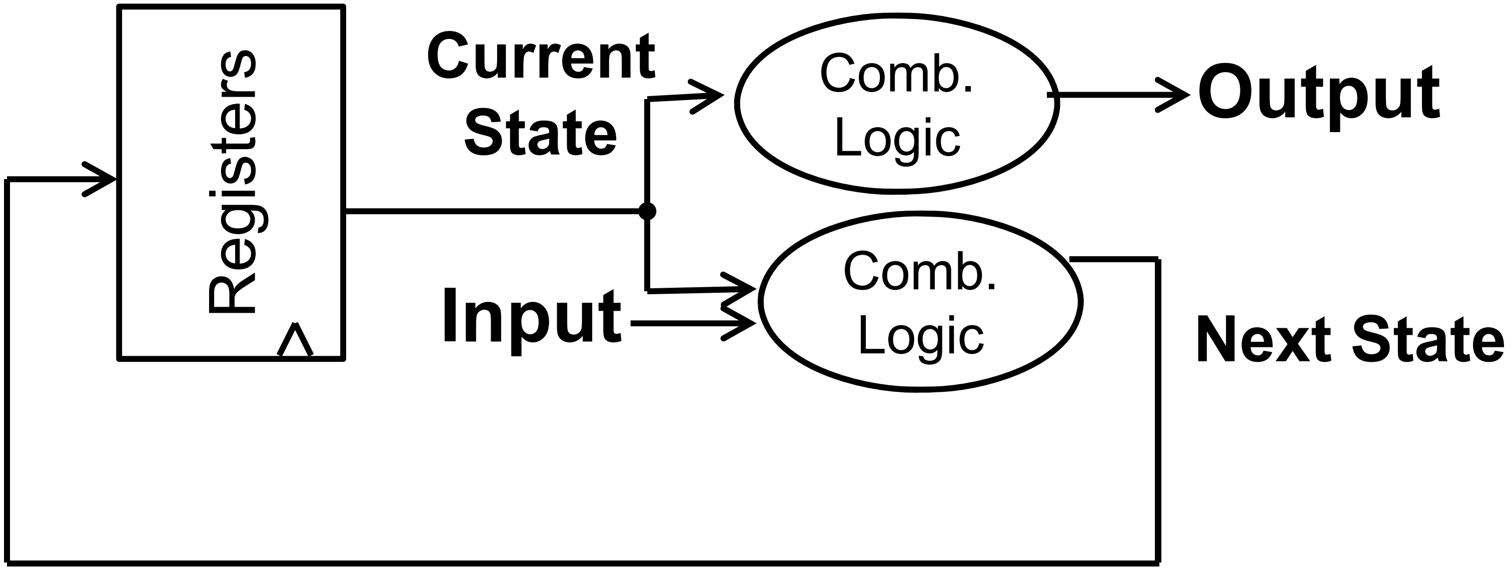
(1) Draw a state diagram (e.g. Moore Machine)

(2) Write output and next-state tables

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

(4) Determine logic equations for next state and outputs

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