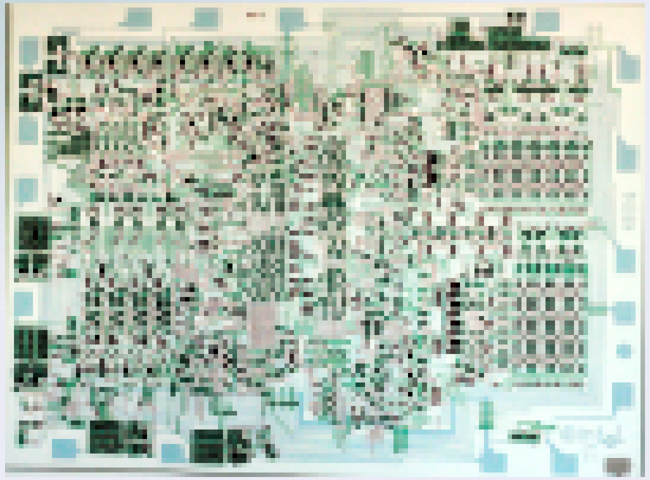


Lecture 3b: Finite State Machines, Sequential Logic and the Clock

CSCI11: **Computer Architecture and Organization**

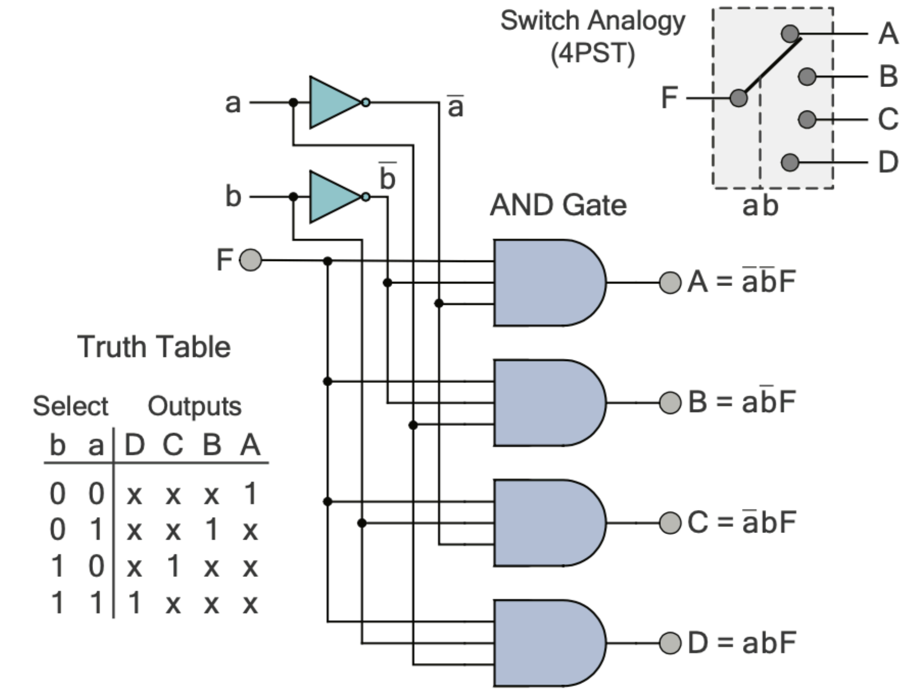
Chapter 3.4-3.6 in Patt and Patel: Introduction to Computing Systems...

- Quick Combinational Circuits Review
- Finite State Machines
- Sequential Logic
- Memory
- The Clock

1 to 4 Demultiplexer

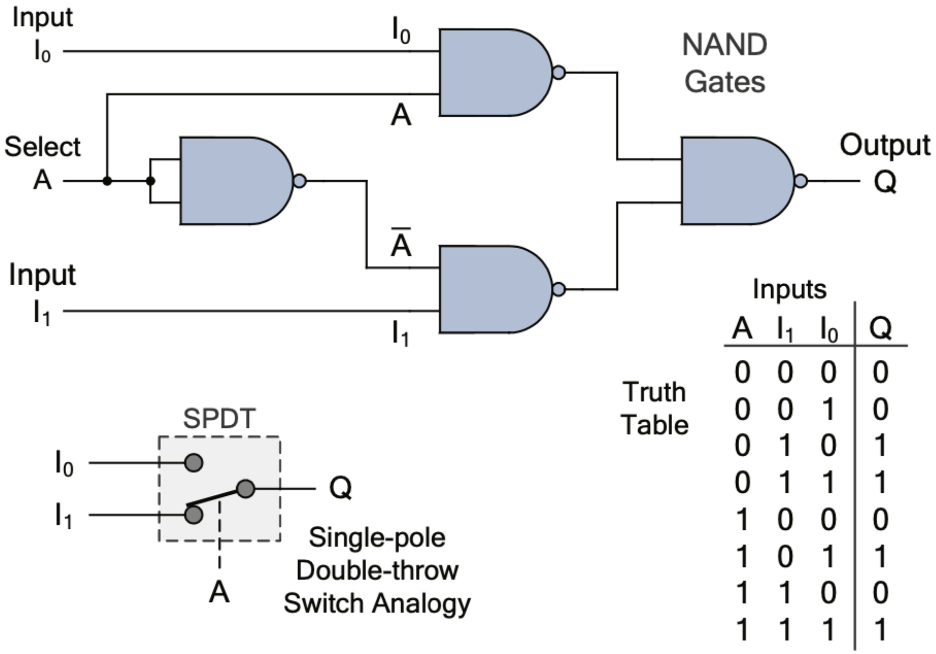
- **data distributor**
- one input to 1 of 4 outputs
- used to light multiple segments of an LED digit display
- not used in the LC3

FIGURE 13. 4-CHANNEL DEMULTIPLEXER CIRCUIT



Select		Outputs			
b	a	D	C	B	A
0	0	x	x	x	1
0	1	x	x	1	x
1	0	x	1	x	x
1	1	1	x	x	x

FIGURE 9. BASIC 2-INPUT MULTIPLEXER USING NAND GATES



A	I_1	I_0	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

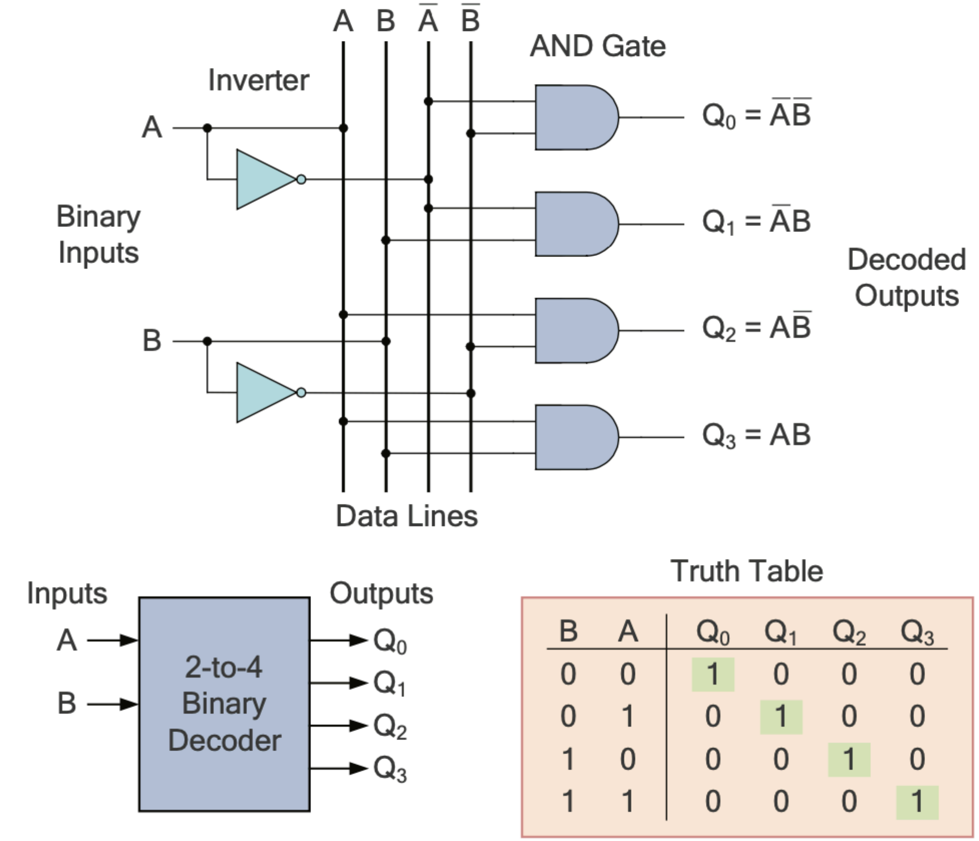
2-input Multiplexer

- **data selector**
- 1 of 2 inputs to one output
- use to select between multiple sets of address/data
- in the LC3, address lines pass through multiple muxes

2-4 Decoder

- **data selector**
- 2^n bits to select one of n lines
- use to select a register in a register file
- in the LC3, opcode's 3 bits select registers R0-R7

FIGURE 3. A 2-TO-4 BINARY DECODER



B	A	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

2 bit Full Adder

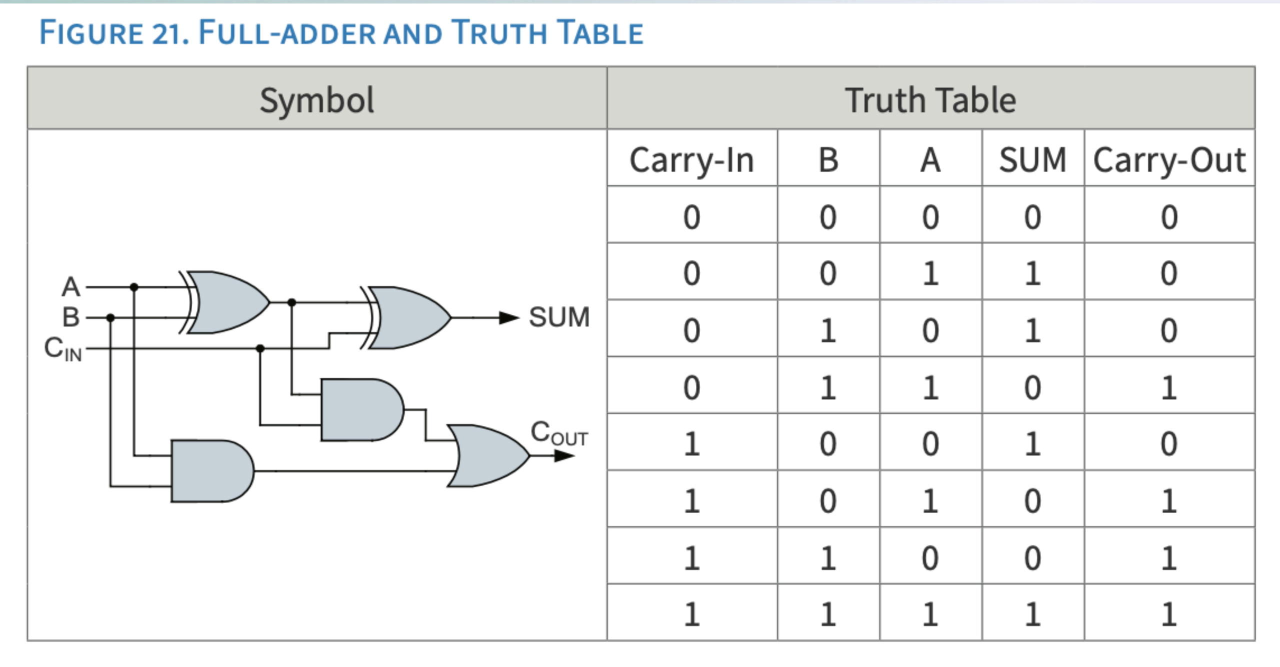
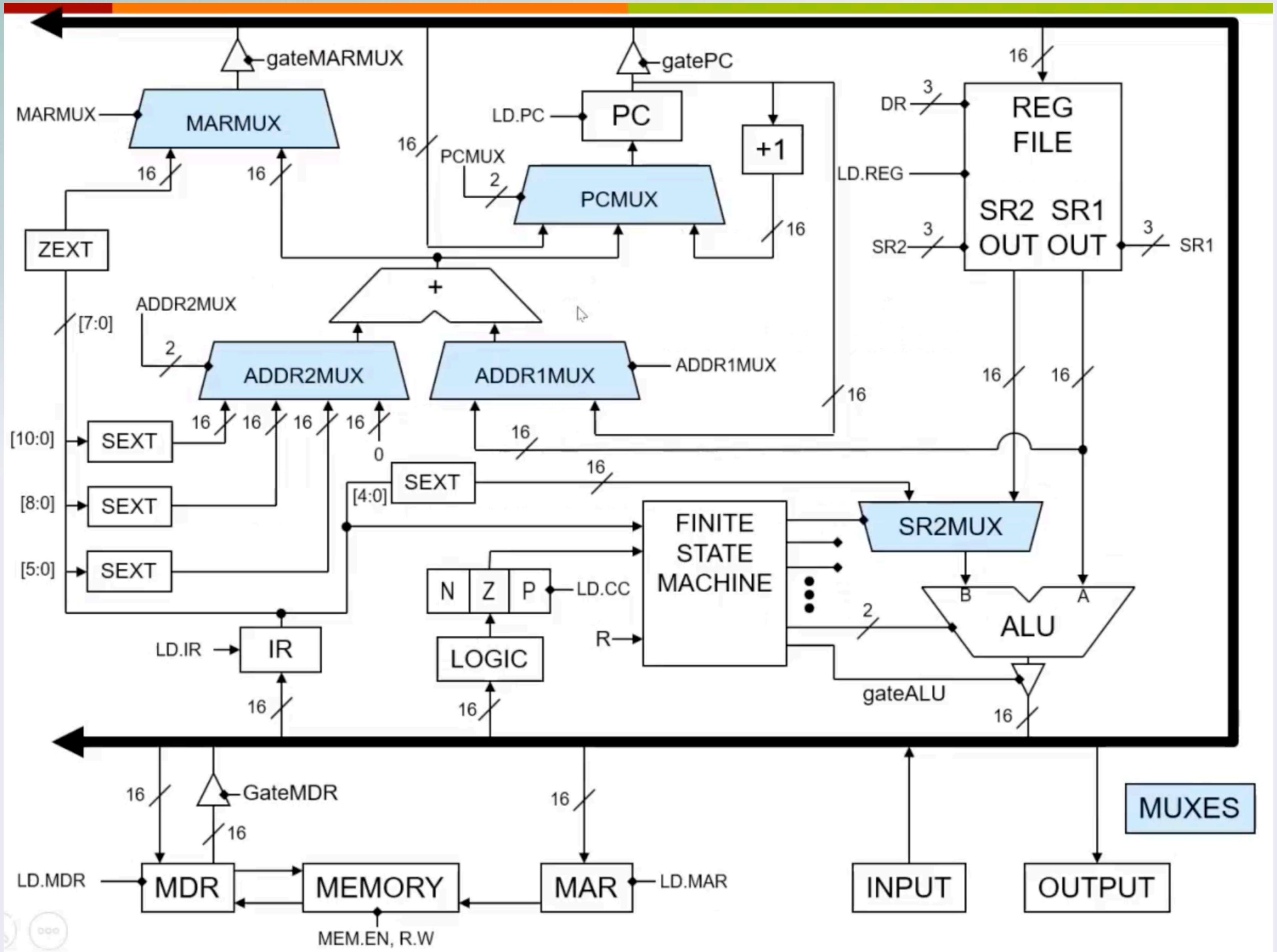

A
B
C_{IN}
SUM
C_{OUT}

FIGURE 21. FULL-ADDER AND TRUTH TABLE

Symbol	Truth Table				
	Carry-In	B	A	SUM	Carry-Out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	0
	0	1	1	0	1
	1	0	0	1	0
	1	0	1	0	1
	1	1	0	0	1
	1	1	1	1	1

- **n bit adder**
- counts up to 2^n bits or divides by 2^bits on Cout
- use as a program counter (**PC**) or a clock divider
- LC3 uses a 16bit full adder as a program counter

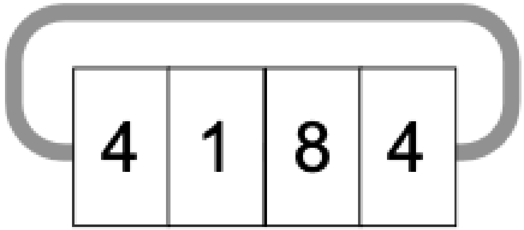

gateMARMUX
MARMUX
MARMUX
16
16
ZEXT
[7:0]
gatePC
LD.PC
PC
+1
16
PCMUX
2
PCMUX
16
+
ADDR2MUX
2
ADDR2MUX
ADDR1MUX
ADDR1MUX
16
16
16
16
16
0
16
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
SEXT
[4:0]
16
16
16
DR
3
REG
FILE
LD.REG
SR2 SR1
OUT OUT
SR2
3
3
SR1
16
16
FINITE
STATE
MACHINE
SR2MUX
B
A
ALU
2
gateALU
16
N
Z
P
LD.CC
R
LOGIC
16
LD.IR
IR
16
16
GateMDR
16
16
LD.MDR
MDR
MEMORY
MEM.EN, R.W
MAR
LD.MAR
INPUT
OUTPUT
MUXES

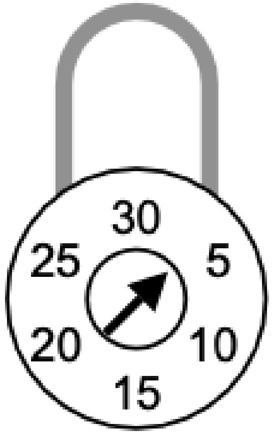
Combinational Logic Circuits

By definition, combinational logic circuits depend on their inputs and have no knowledge of previous events. That is, they are memory-less.

Combinational vs. Sequential

Two types of “combination” locks





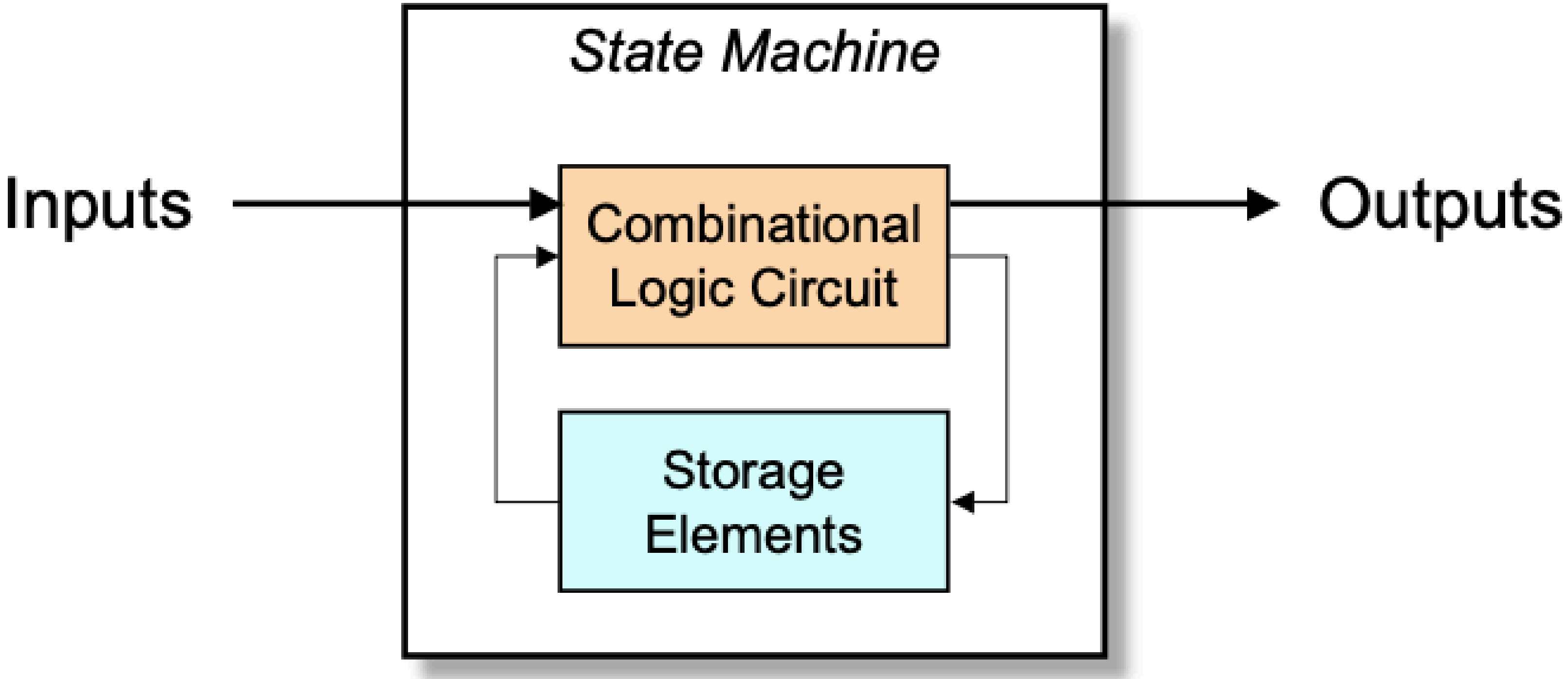
Combinational
Success depends only on the values, not the order in which they are set.

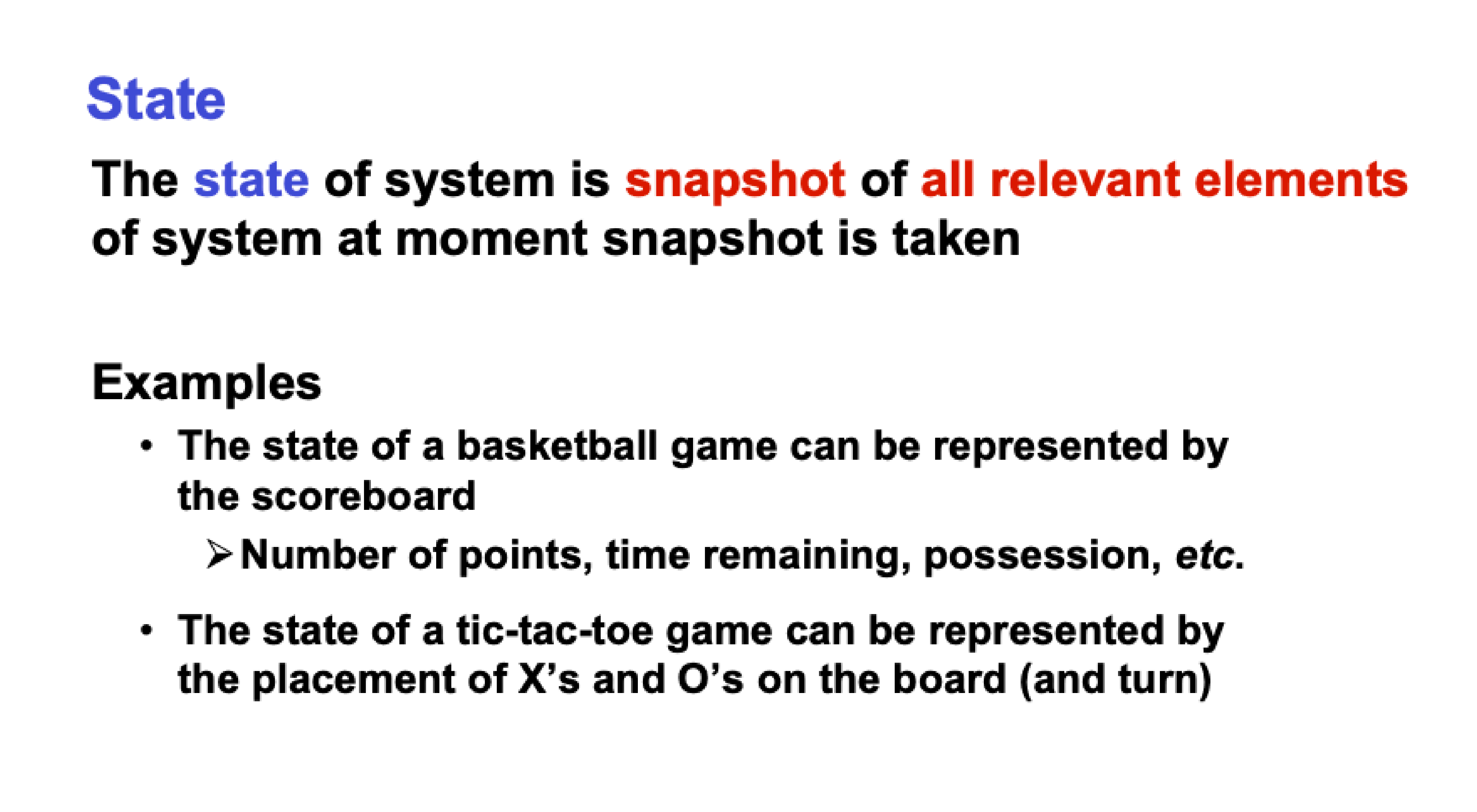
Sequential
Success depends on the sequence of values (e.g, R-13, L-22, R-3).

State Machine

Another type of sequential circuit

- Combines combinational logic with storage
- “Remembers” state, and changes output (and state) based on inputs and current state



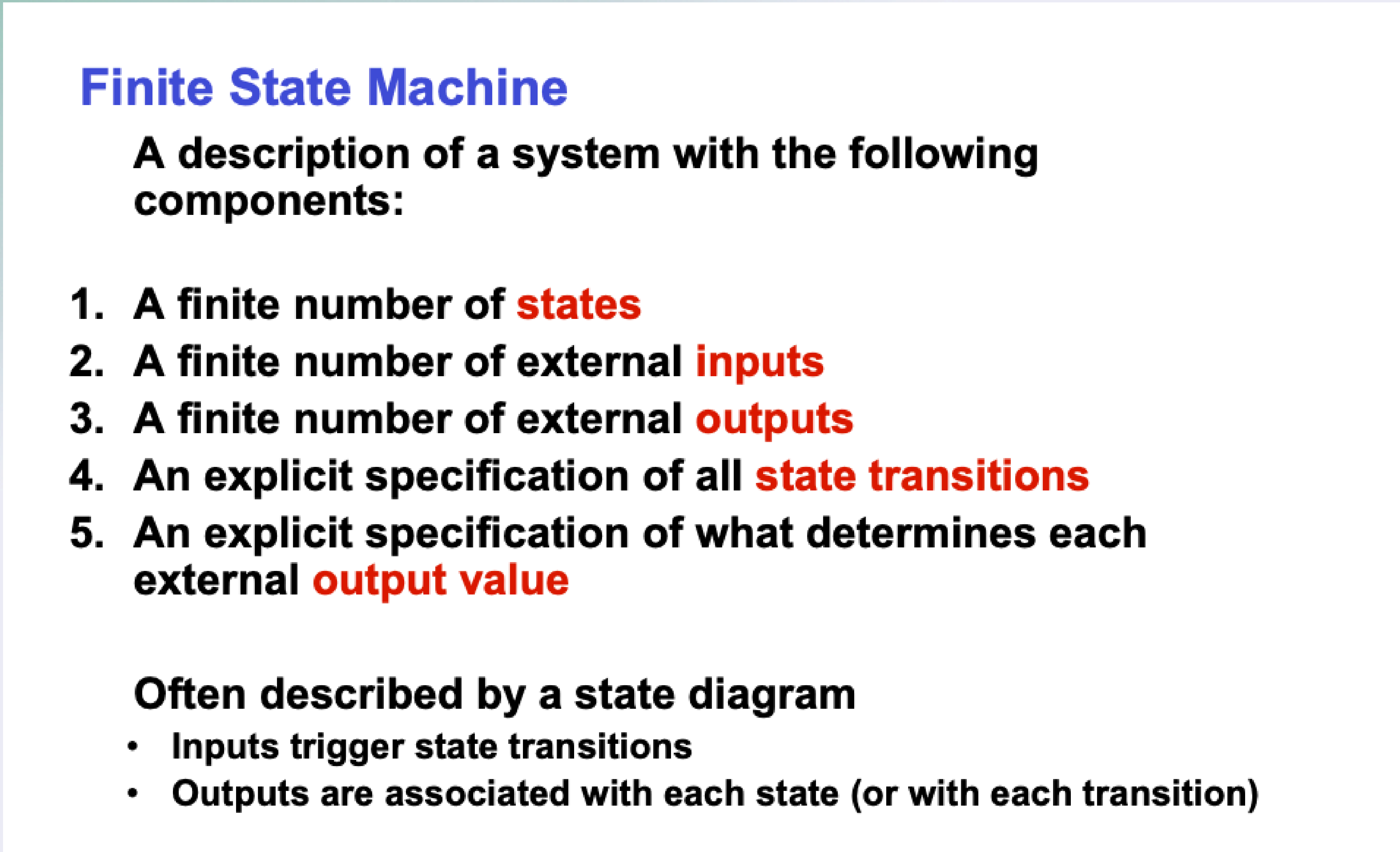


State

The state of system is snapshot of all relevant elements of system at moment snapshot is taken

Examples

- **The state of a basketball game can be represented by the scoreboard**
  - **Number of points, time remaining, possession, *etc*.**
- **The state of a tic-tac-toe game can be represented by the placement of X’s and O’s on the board (and turn)**



Finite State Machine

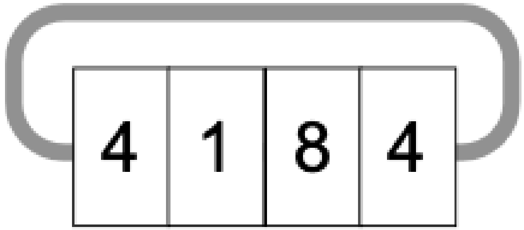
A description of a system with the following components:

1. A finite number of **states**
2. A finite number of external **inputs**
3. A finite number of external **outputs**
4. An explicit specification of all **state transitions**
5. An explicit specification of what determines each external **output value**

Often described by a state diagram

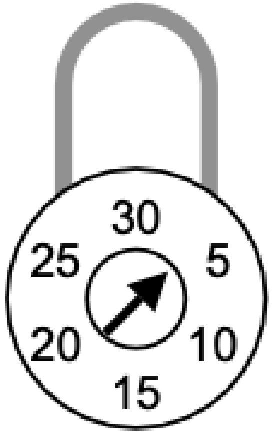
- Inputs trigger state transitions
- Outputs are associated with each state (or with each transition)

Two State Lock

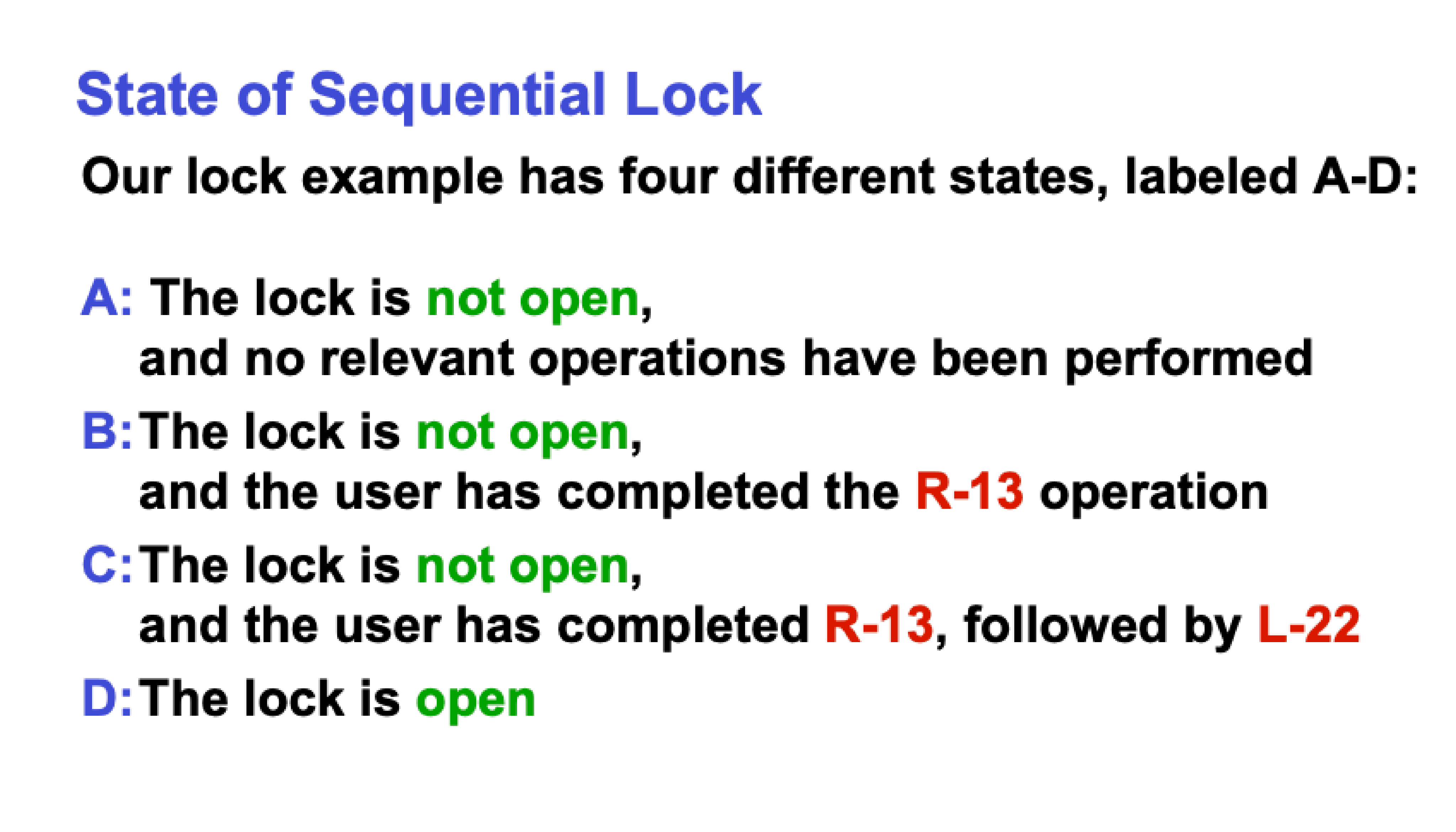


Combinational
Success depends only on the values, not the order in which they are set.

Finite State Machine



Sequential
Success depends on the sequence of values (e.g, R-13, L-22, R-3).



State of Sequential Lock

Our lock example has four different states, labeled A-D:

A: The lock is not open,
and no relevant operations have been performed

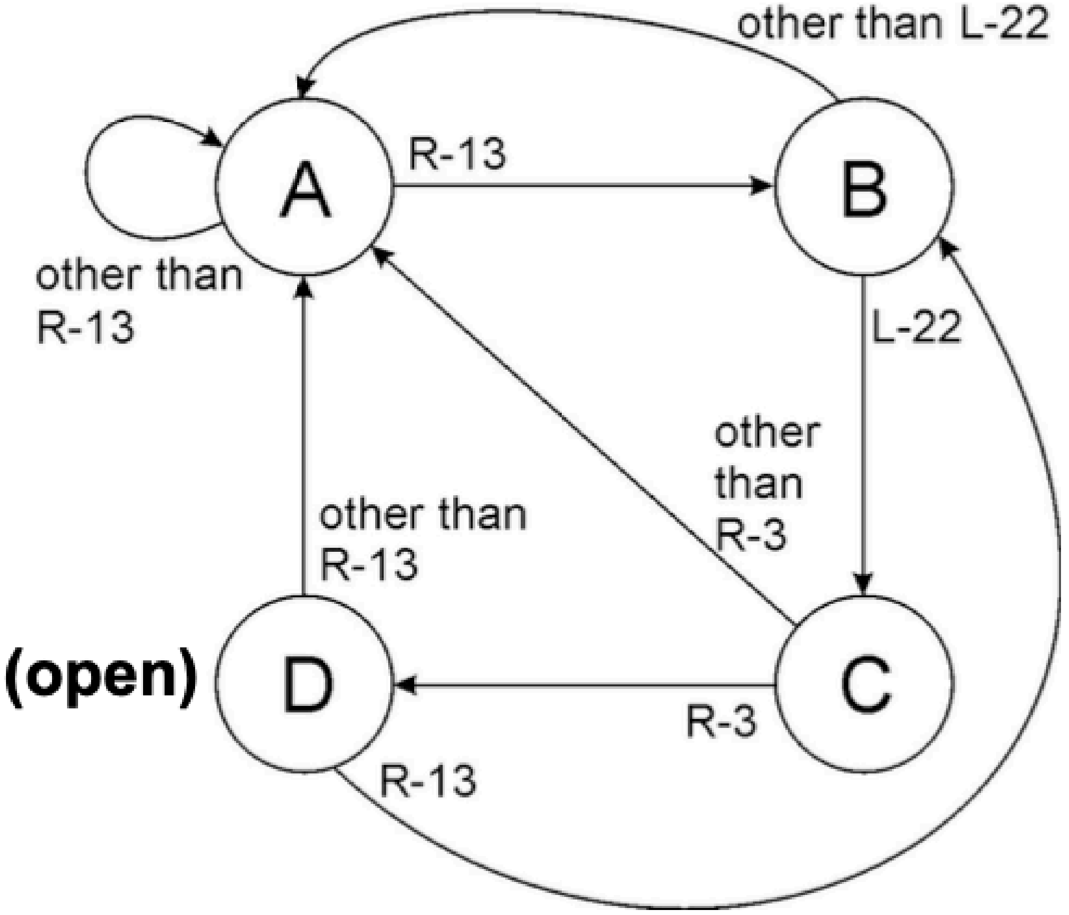
B: The lock is not open,
and the user has completed the R-13 operation

C: The lock is not open,
and the user has completed R-13, followed by L-22

D: The lock is open

Sequential Lock State Diagram

Shows states and actions that cause a transition between states



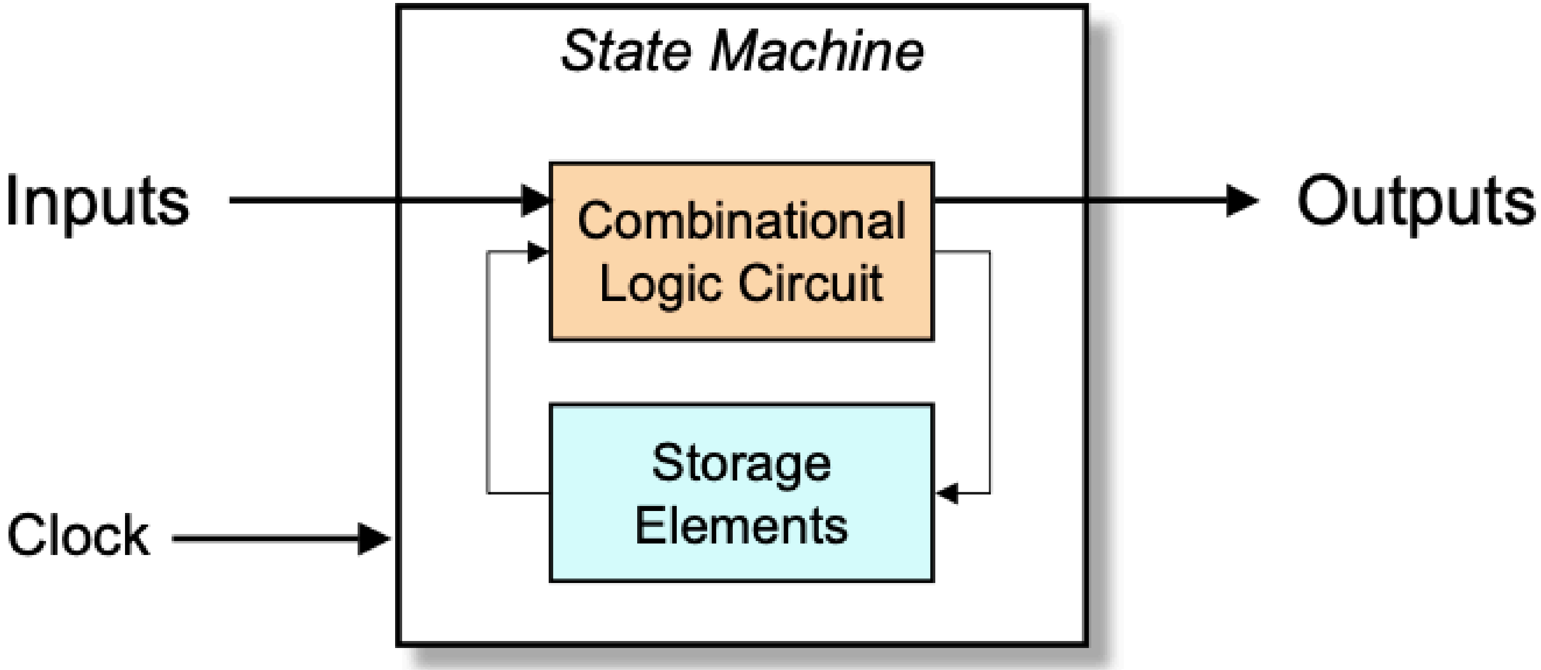
Implementing a Finite State Machine

Combinational logic

- Determine outputs and next state.

Storage elements

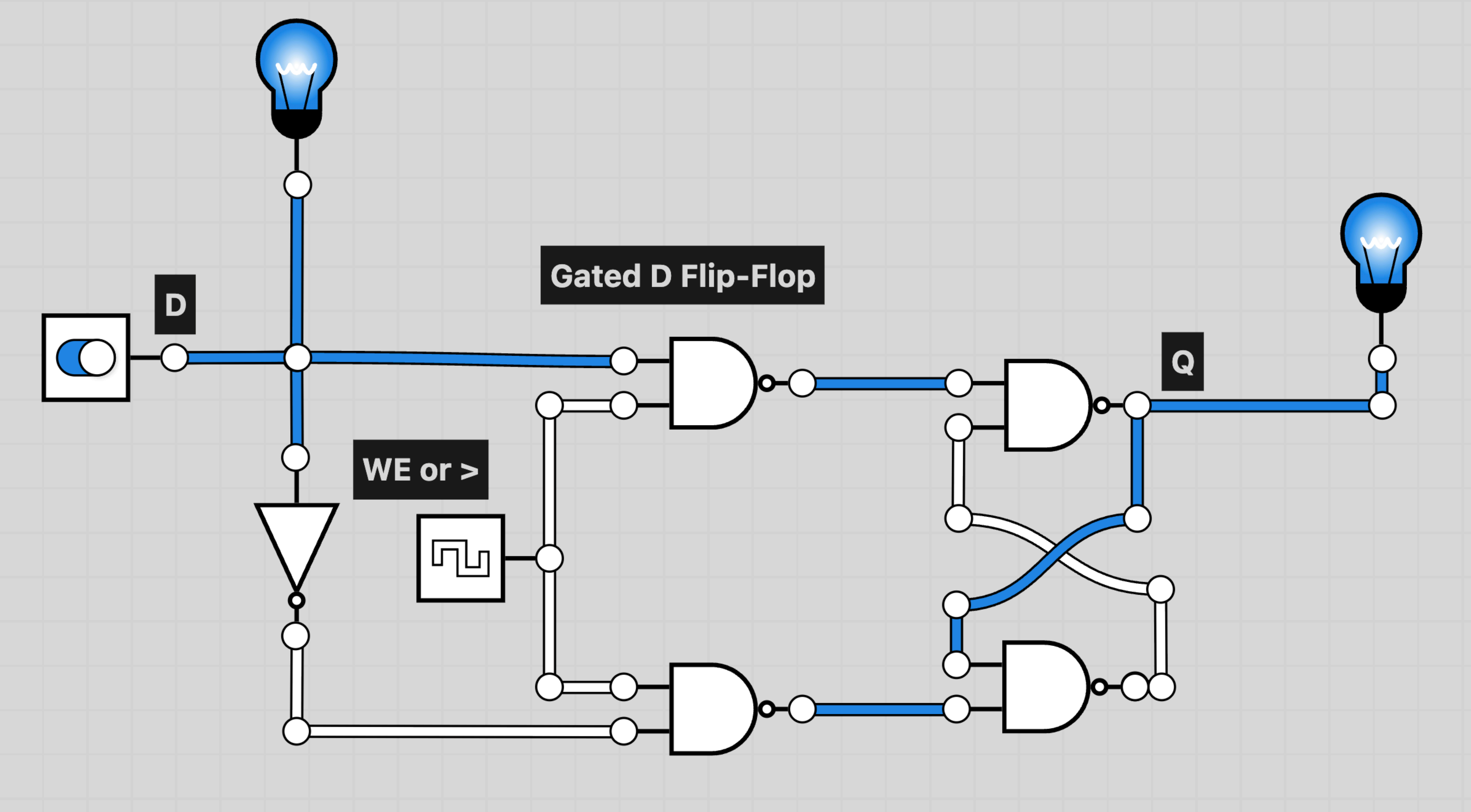
- Maintain state representation.

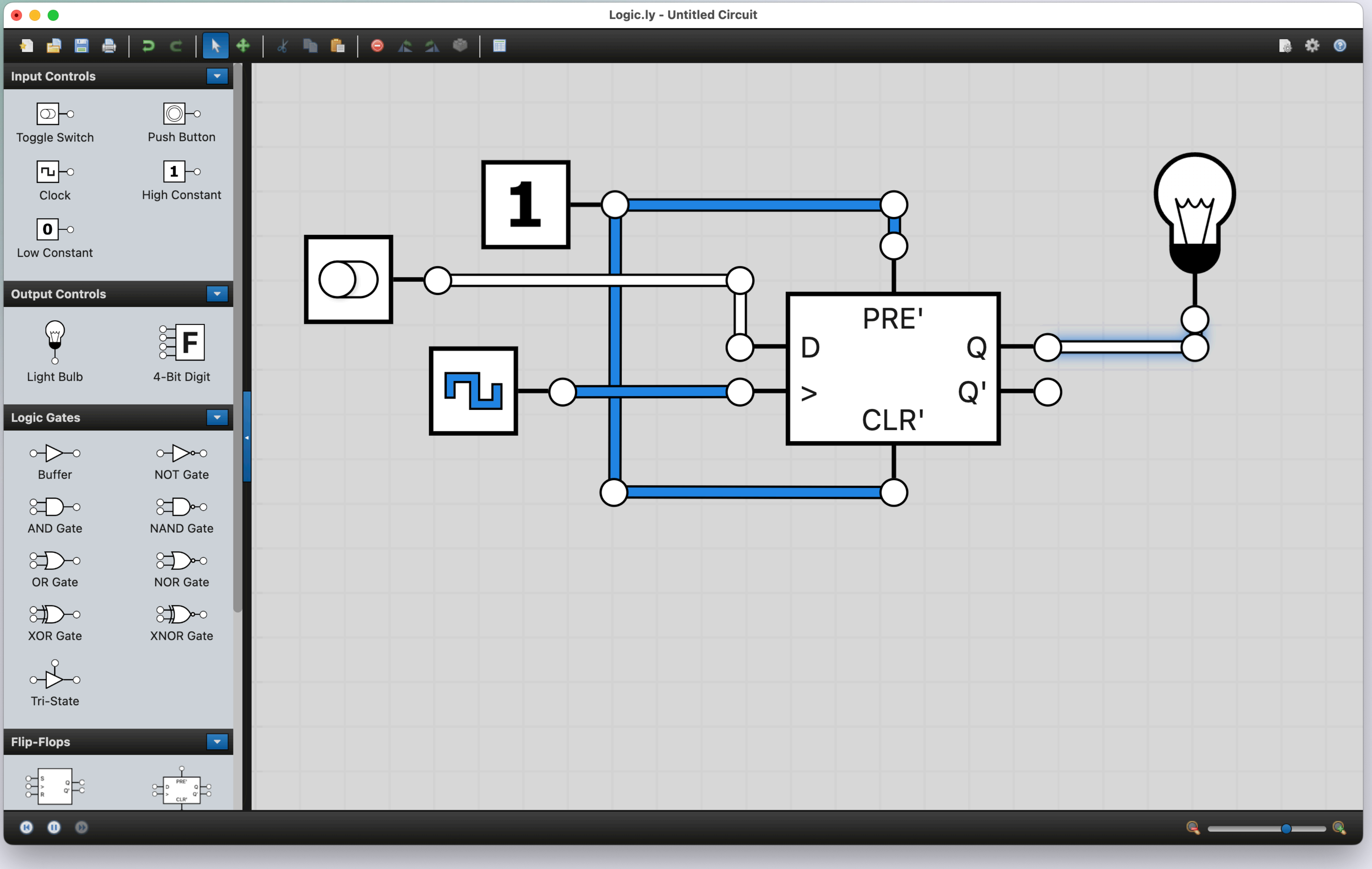


Why is a Clock Needed?

1. To ensure there is a time element to the circuit
2. To ensure things are synchronized

See D Latch as an example


Gated D Flip-Flop
D
WE or >
Q


Logic.ly - Untitled Circuit
Input Controls
Toggle Switch
Push Button
Clock
High Constant
Low Constant
Output Controls
Light Bulb
4-Bit Digit
Logic Gates
Buffer
NOT Gate
AND Gate
NAND Gate
OR Gate
NOR Gate
XOR Gate
XNOR Gate
Tri-State
Flip-Flops
1
PRE'
D
Q
>
Q'
CLR'

Clocks

Clock Characteristics

- Usually generated by an oscillating crystal
- Fixed period
- Frequency = 1/period
- 1Mhz = 1/10^6 seconds or 1 microsecond or 1us
- 1GHz = 1/10^9 seconds or 1 nanosecond or 1ns

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

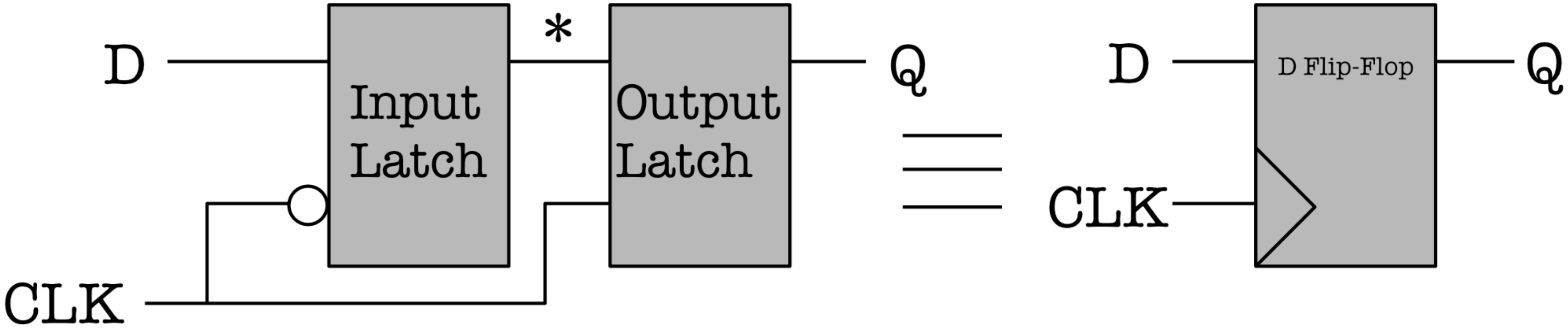
Edge-Triggering

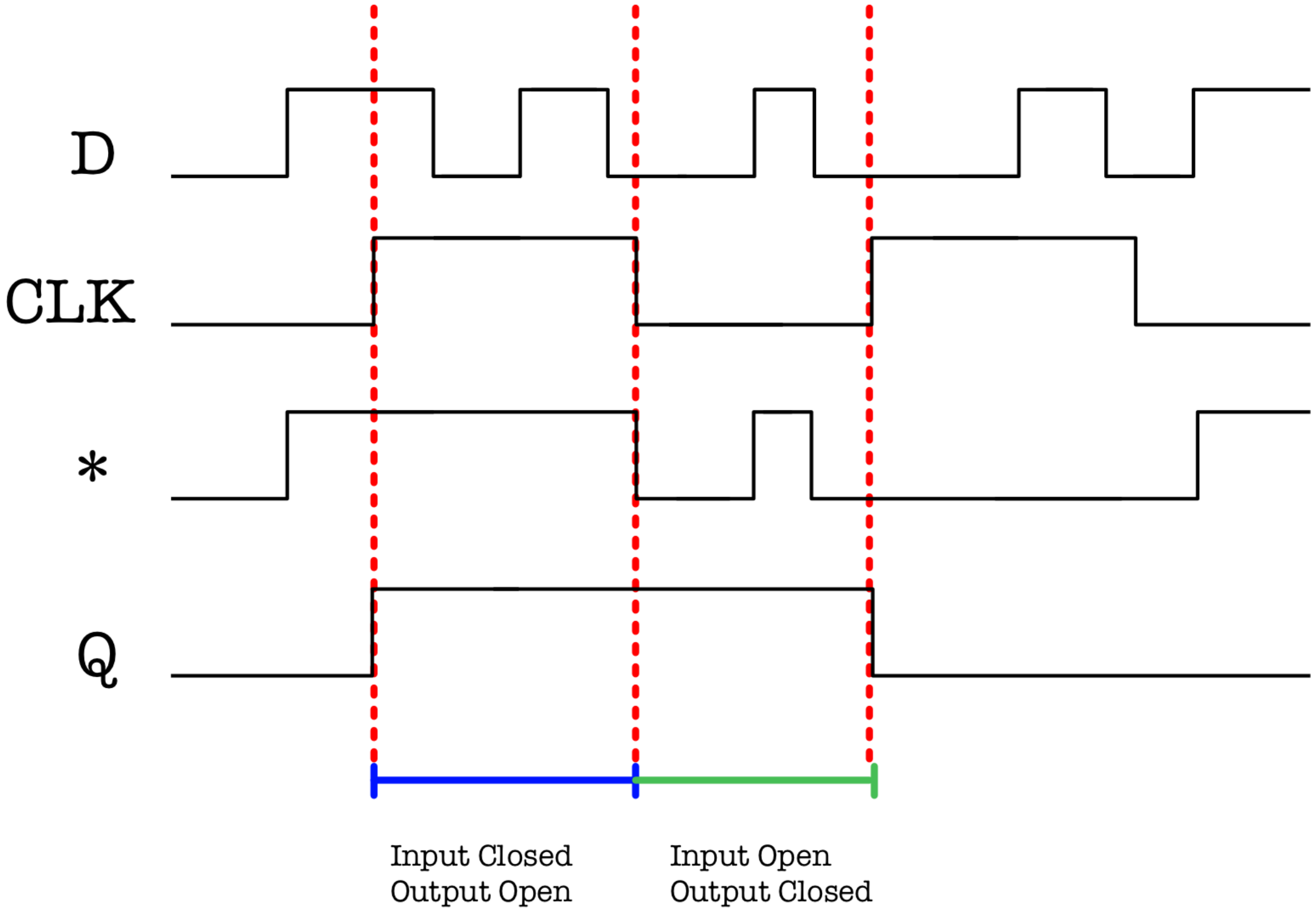
Triggering Mechanisms

- Rising edge: Positive edge-triggered
- Falling edge: Negative edge-triggered

Requirements

- Inputs must be stable just before the triggering edge.

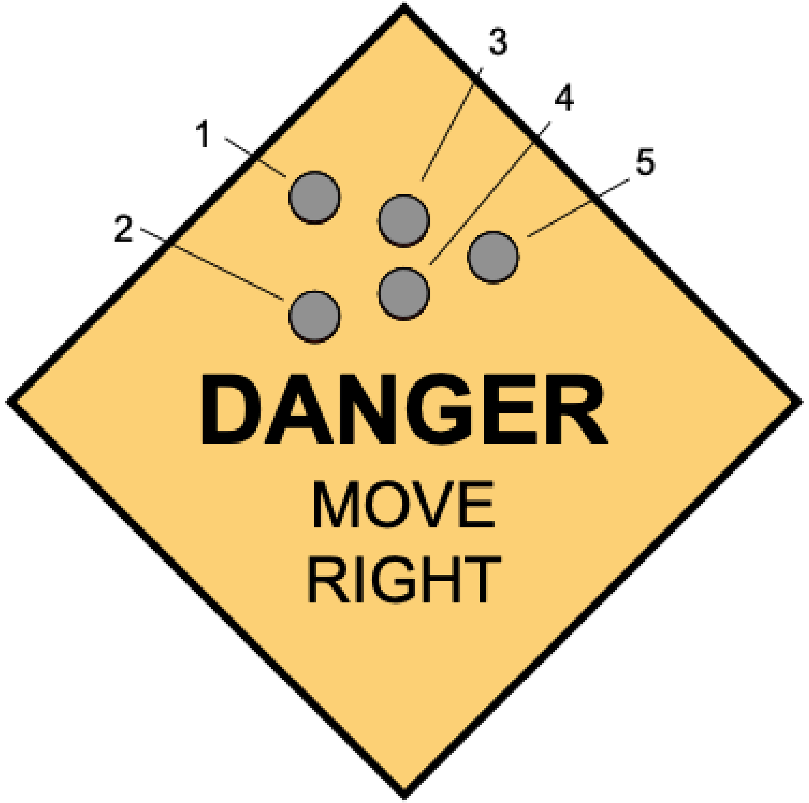

D
Input Latch
*
Output Latch
Q
CLK
D
D Flip-Flop
Q
CLK


D
CLK
*
Q
Input Closed
Output Open
Input Open
Output Closed

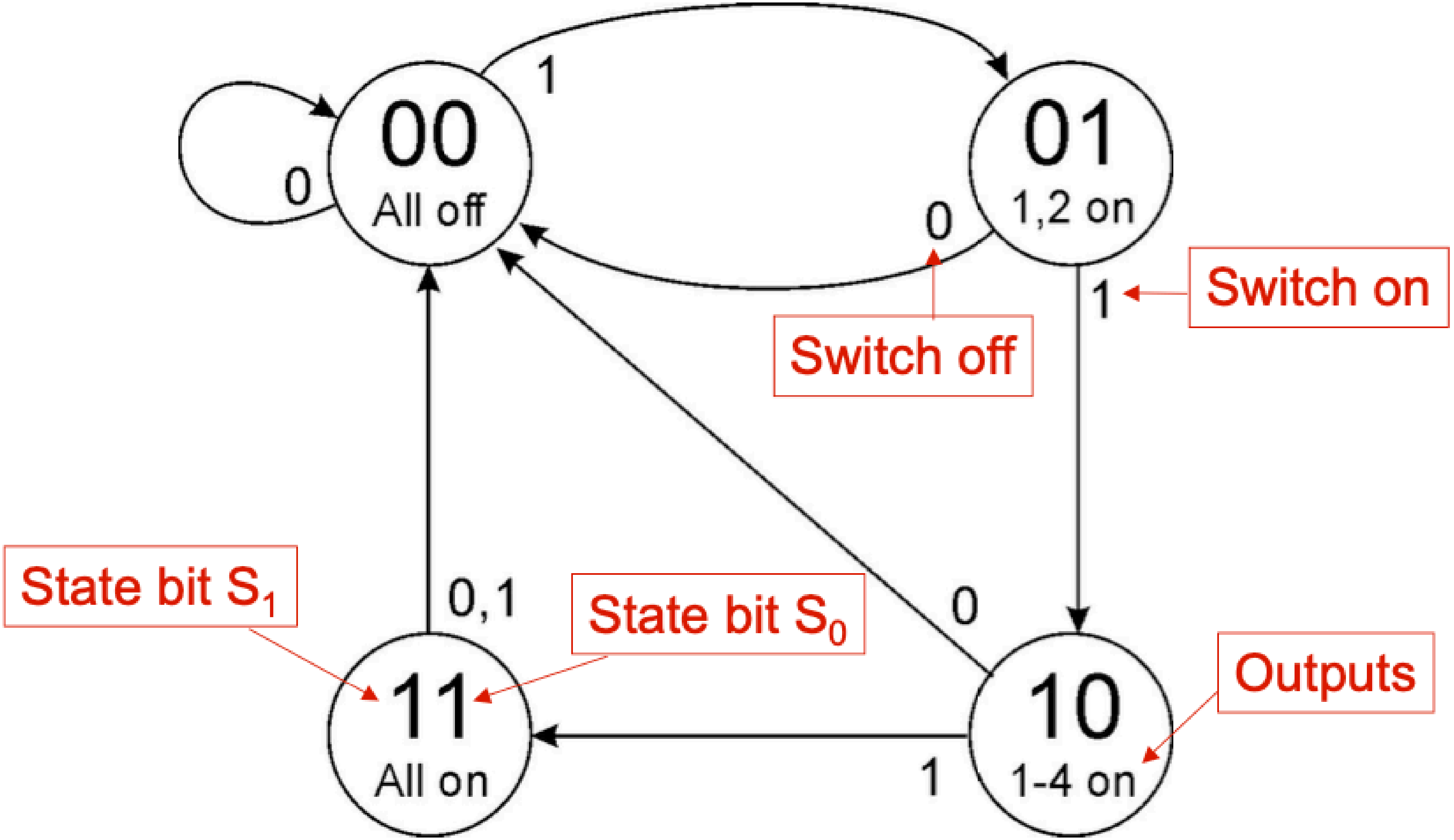
Complete Example

A blinking traffic sign

- No lights on
- 1 & 2 on
- 1, 2, 3, & 4 on
- 1, 2, 3, 4, & 5 on
- (repeat as long as switch is turned on)

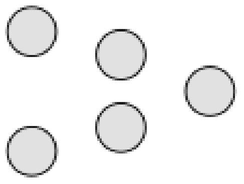


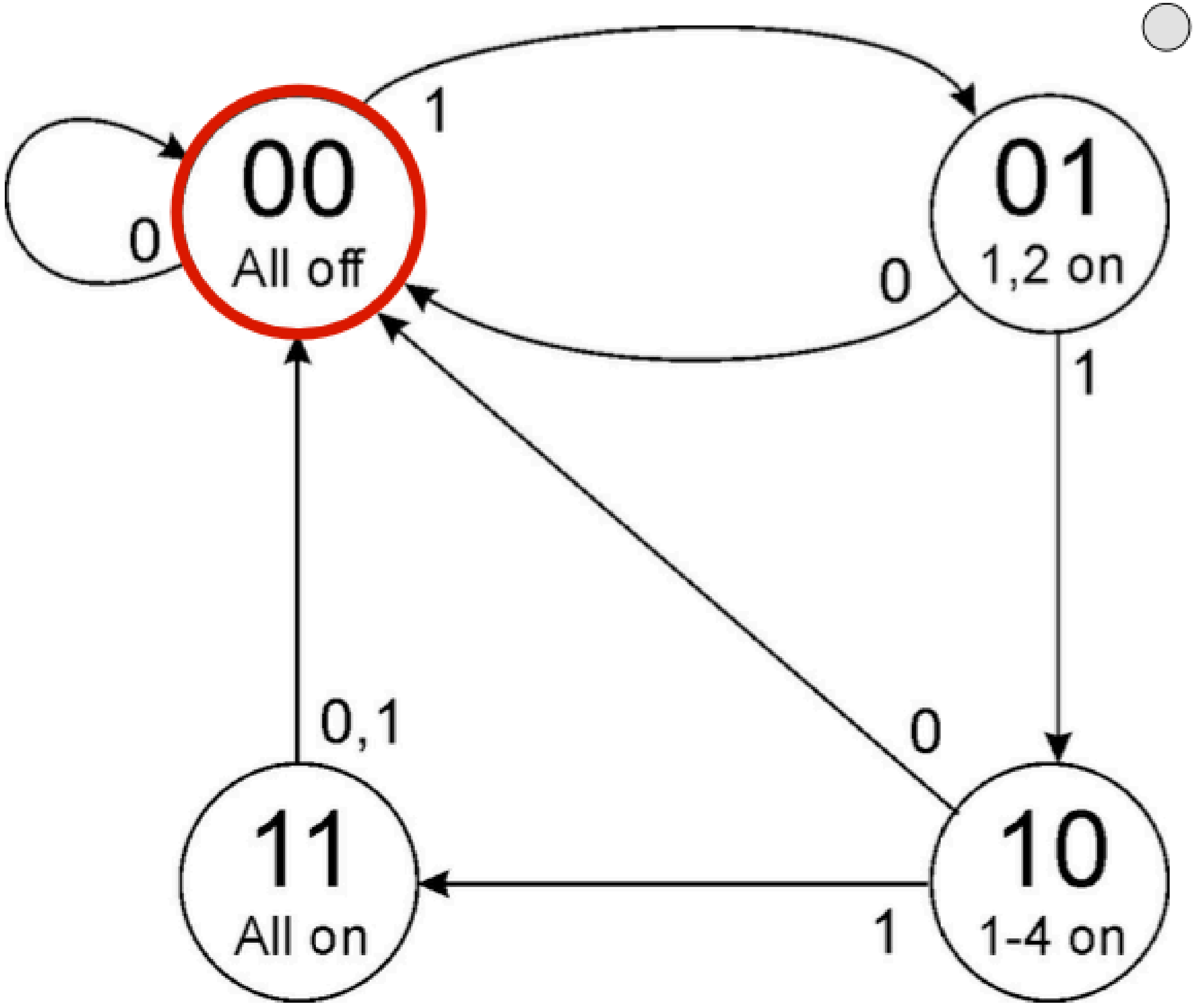
Traffic Sign State Diagram



Transition on each clock cycle.

Traffic Sign State Diagram: State 00

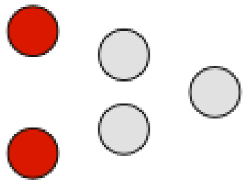


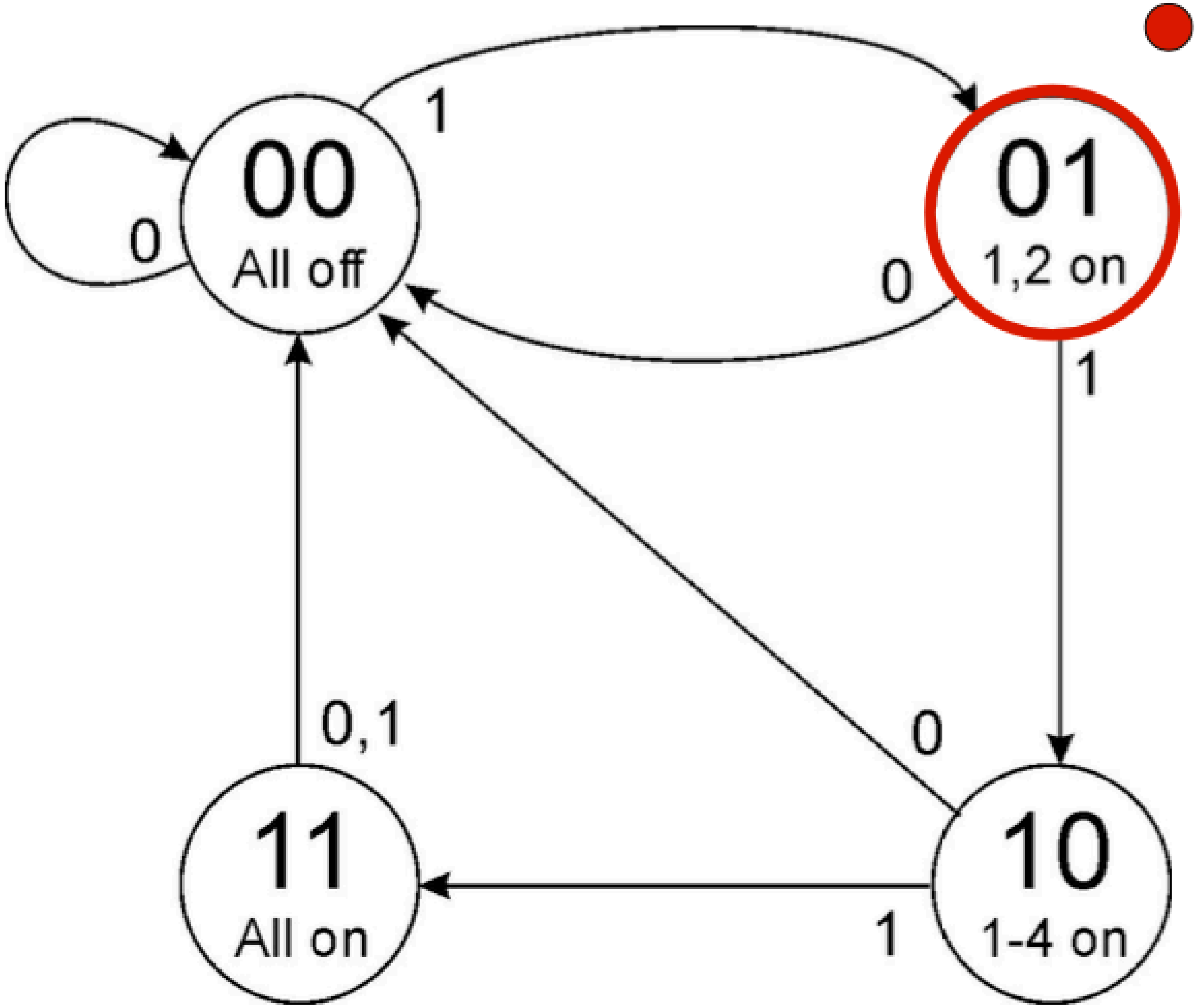


CSE 240

Transition on each clock cycle.

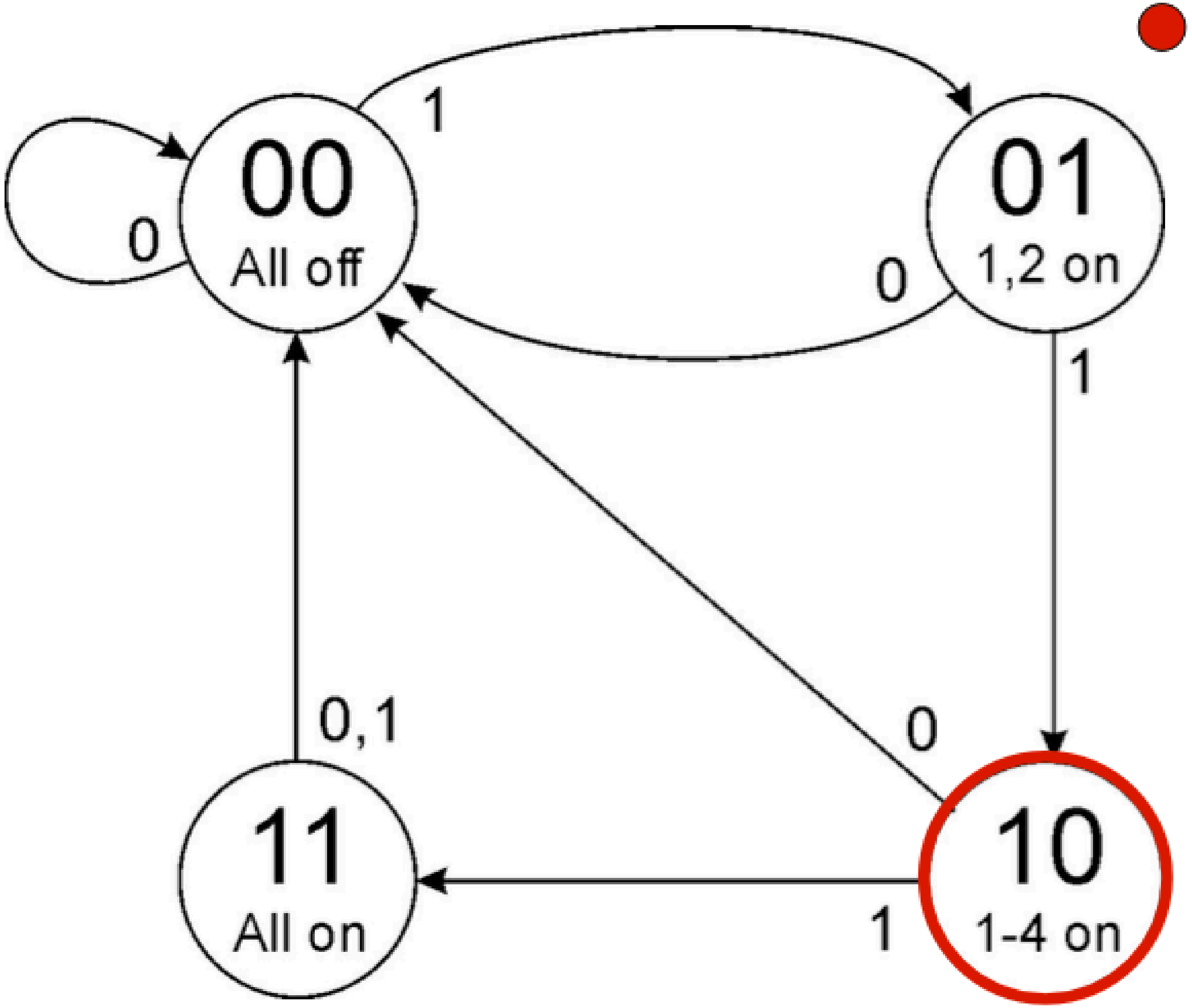
Traffic Sign State Diagram: State 01





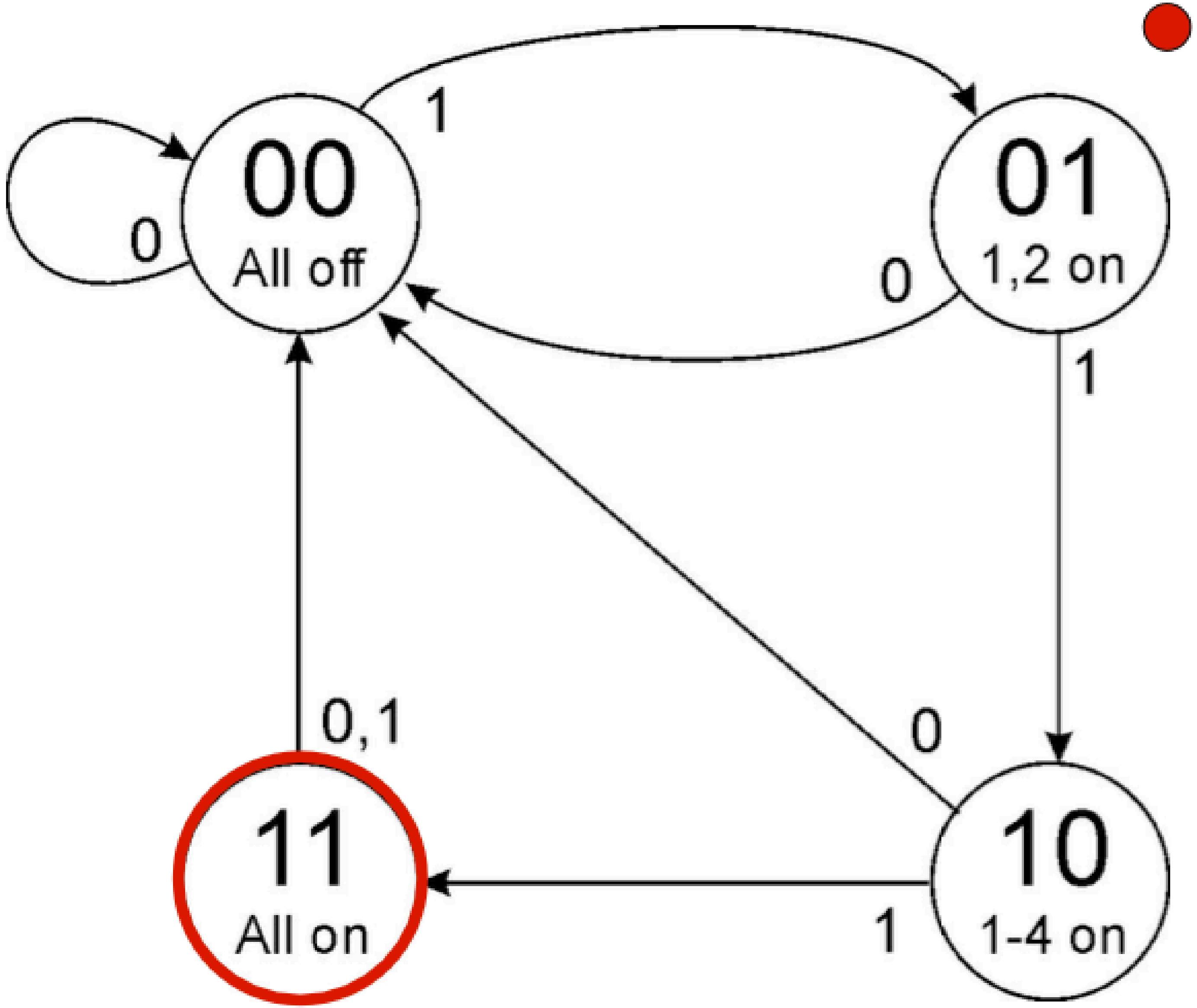
Transition on each clock cycle.

Traffic Sign State Diagram: State 10



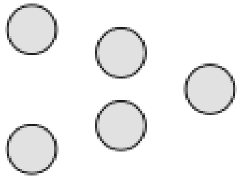
Transition on each clock cycle.

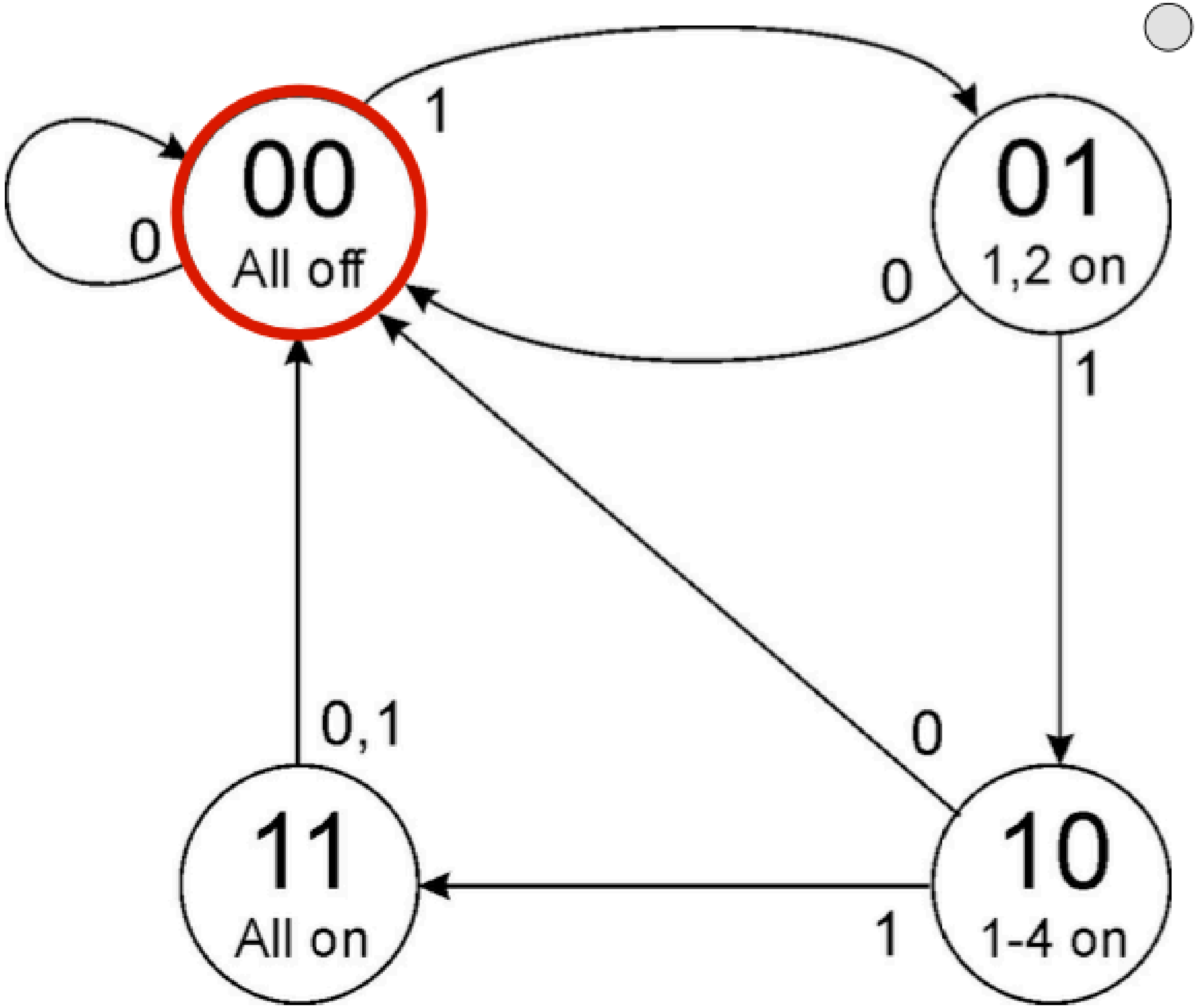
Traffic Sign State Diagram: State 11



Transition on each clock cycle.

Traffic Sign State Diagram: State 00





CSE 240

Transition on each clock cycle.

3-99

Traffic Sign Truth Tables

Outputs
(depend only on state: S_1S_0)

Z: Lights 1 and 2

Y: Lights 3 and 4

X: Light 5

S_1	S_0	Z	Y	X
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

Next State: $S_1'S_0'$
(depend on state and input)

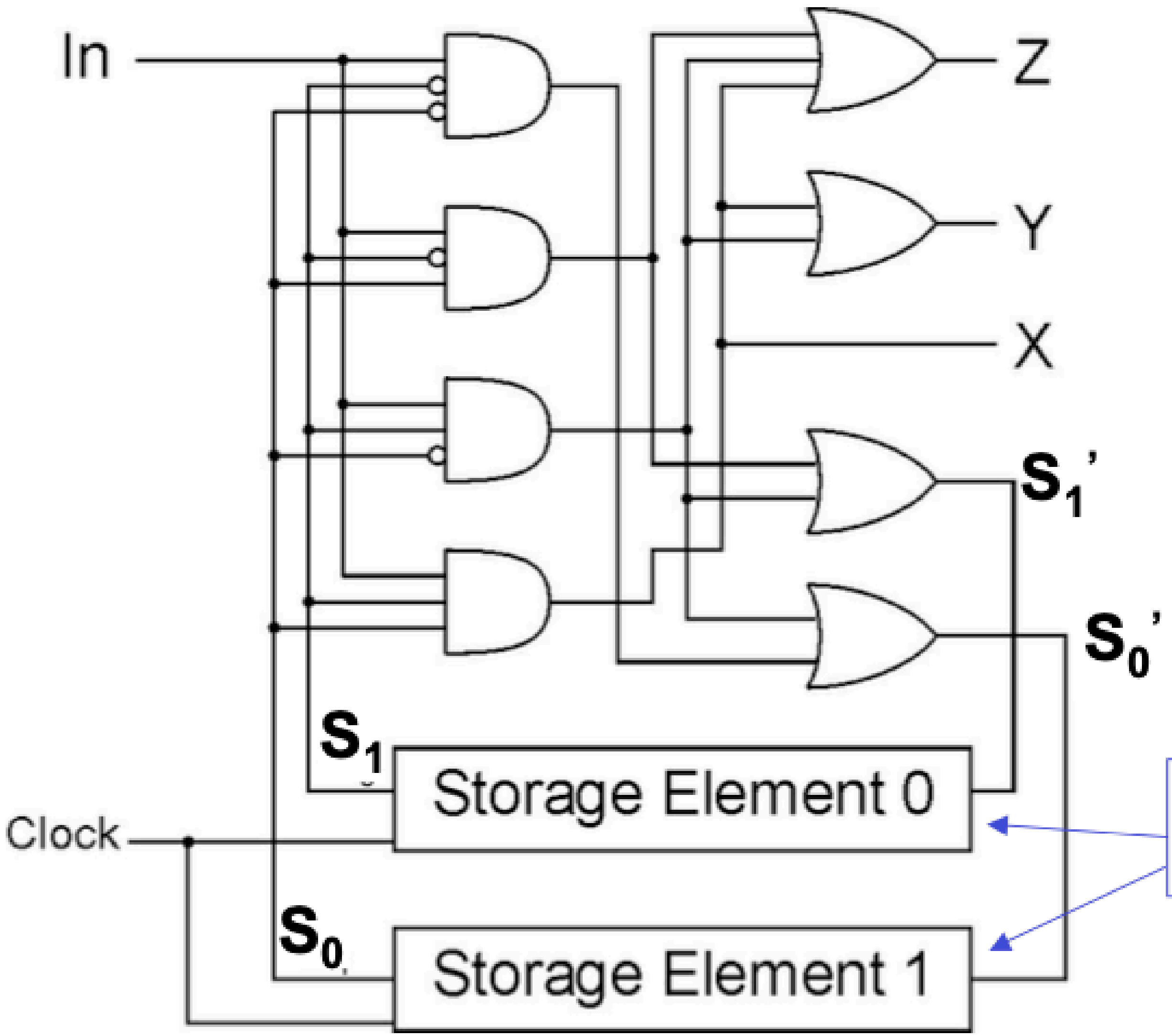
In: Switch

In	S_1	S_0	S_1'	S_0'
0	X	X	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

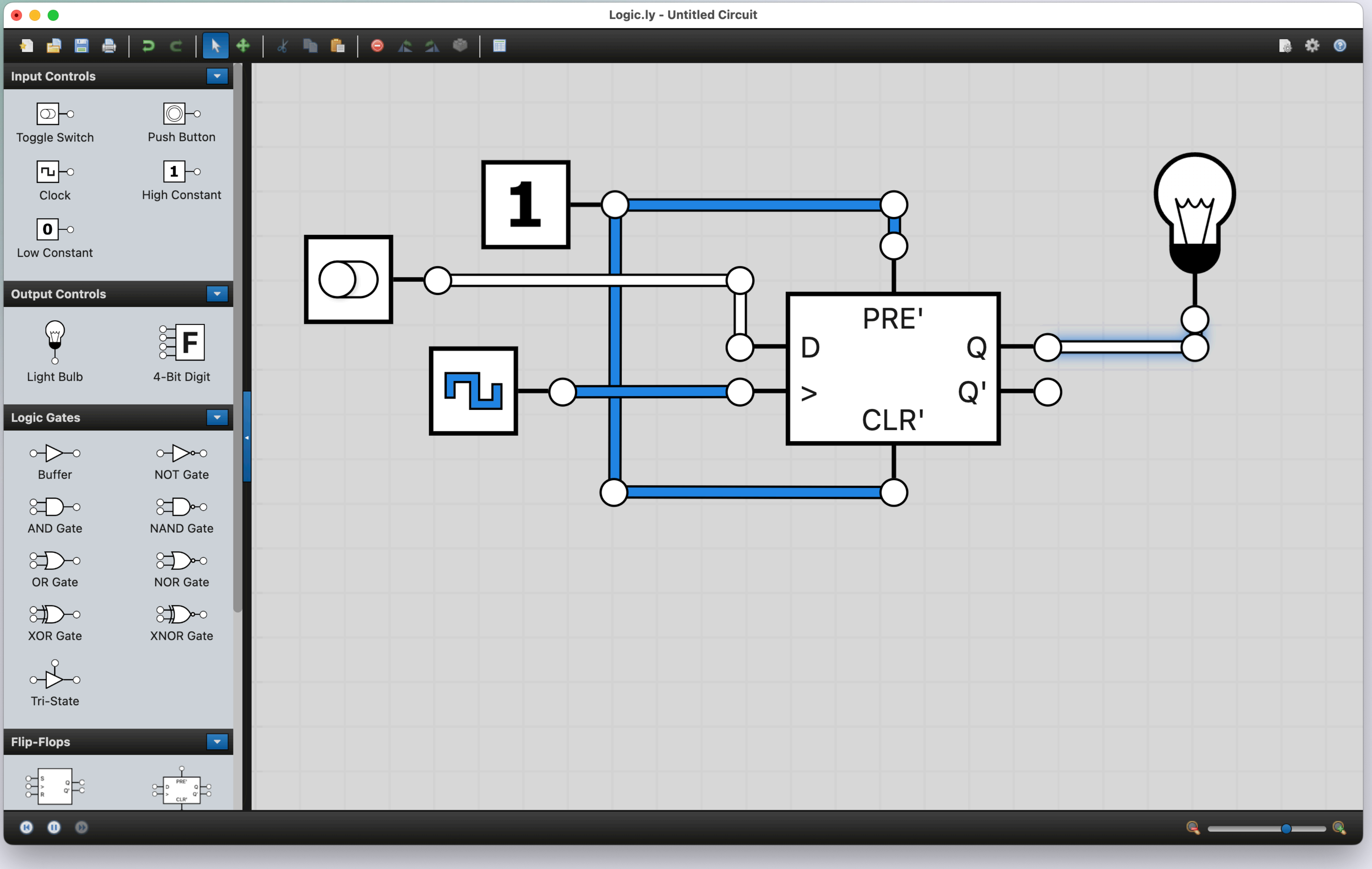
Don't care (1 or 0)

Whenever In=0, next state is 00.

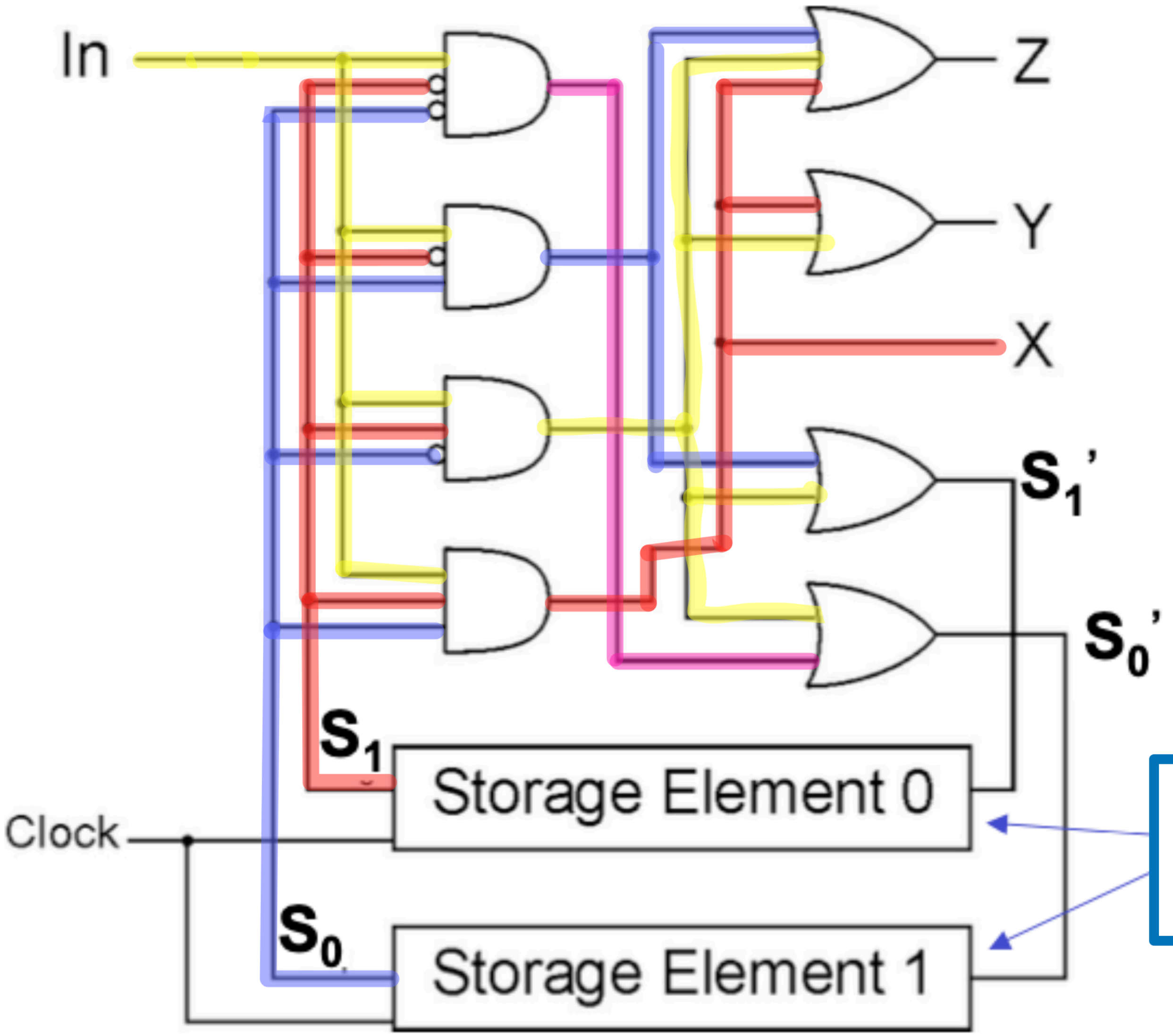
Traffic Sign Logic



Master-slave flip-flop


Logic.ly - Untitled Circuit
Input Controls
Toggle Switch
Push Button
Clock
High Constant
Low Constant
Output Controls
Light Bulb
4-Bit Digit
Logic Gates
Buffer
NOT Gate
AND Gate
NAND Gate
OR Gate
NOR Gate
XOR Gate
XNOR Gate
Tri-State
Flip-Flops
1
PRE'
D
Q
>
Q'
CLR'

Traffic Sign Logic



D Flip-Flop

Credits:

Patt and Patel: "*Introduction to Computing Systems*"

AspenCore: "*Sequential Logic*"

UPenn CSE240 Fall 2006 Fassell: "*Chapter 3 Lecture*"

Sequential Logic and Synchronization CS 50.002