Sequential Circuits

Prof. Usagi

Recap: What's 16777216 special about?

0 10010111 0000 0000 0000 0000 0000 000

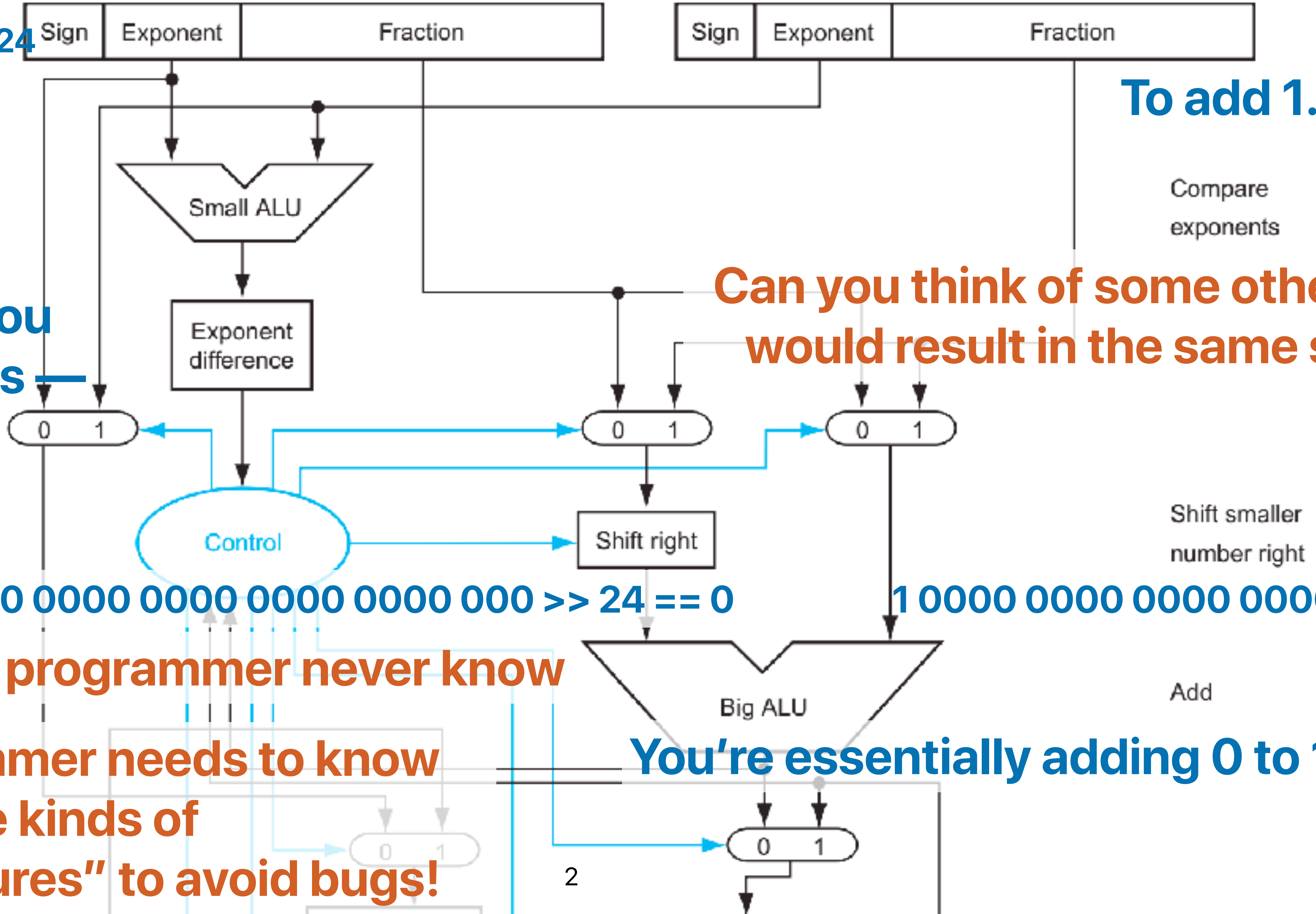
0 0111 1111 0000 0000 0000 0000 0000 000

16777216 = 1.0 * 2^{24}

To add 1.0 = 1.0 *2^{0}

to this number, you have to shift 24 bits —

Can you think of some other numbers would result in the same situation?



1 0000 0000 0000 0000 0000 000 >> 24 == 0

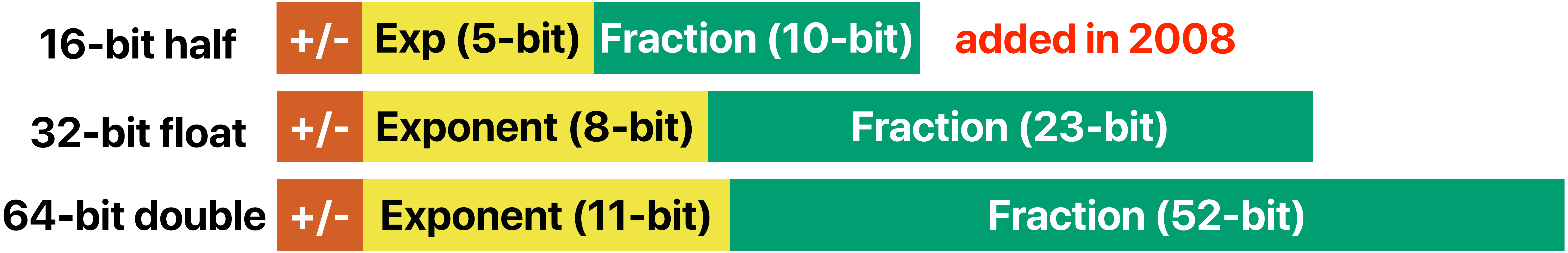
1 0000 0000 0000 0000 0000 000

— even worse — programmer never know

You're essentially adding 0 to 16777216

A good programmer needs to know these kinds of "hardware features" to avoid bugs!

Recap: Other floating point formats



16-bit half +/- Exp (5-bit) Fraction (10-bit) added in 2008

32-bit float +/- Exponent (8-bit) Fraction (23-bit)

64-bit double +/- Exponent (11-bit) Fraction (52-bit)

- Not all applications require “high precision”
  - Deep neural networks are surprisingly error tolerable
- Again — Trade-off between area-efficiency/cost/performance/accuracy

Recap: Combinational v.s. sequential logic

- Combinational logic
  - The output is a pure function of its current inputs
  - The output doesn't change regardless how many times the logic is triggered — Idempotent
- Sequential logic
  - The output depends on current inputs, previous inputs, their history

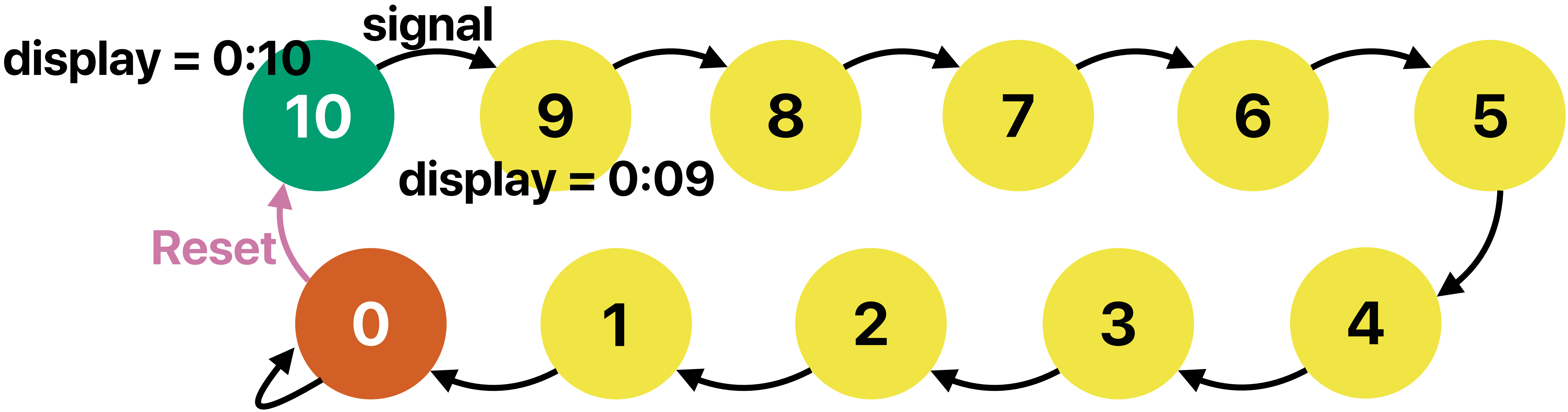
Sequential circuit has memory!

Recap: Theory behind each

- A **Combinational logic** is the implementation of a **Boolean Algebra** function with only Boolean Variables as their inputs
- A **Sequential logic** is the implementation of a **Finite-State Machine**

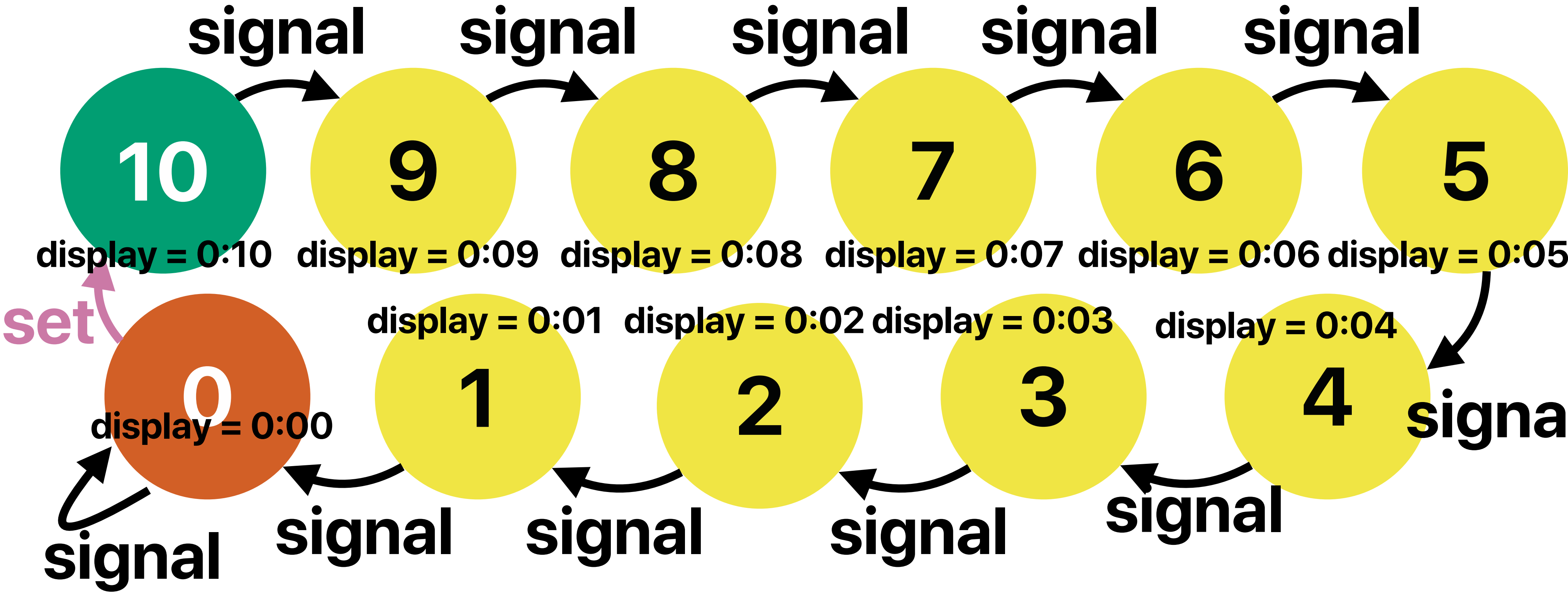
Recap: Count-down Timer

- What do we need to implement this timer?
  - Set an initial value/“state” of the timer
  - “Signal” the design every second
  - The design changes its “state” every time we received the signal until we reaches “0” — the final state



Recap: Finite State Machines

- FSM consists of
  - Set of states
  - Set of inputs, set of outputs
  - Initial state
  - Set of transitions
    - Only one can be true at a time
- FSM representations:
  - State diagram
  - State table



Current State	Next State Signal	
	0	1
10	10	9
9	9	8
8	8	7
7	7	6
6	6	5
5	5	4
4	4	3
3	3	2
2	2	1
1	1	0
0	0	0

Recap: Life on Mars

- Mars rover has a binary input x. When it receives the input sequence x(t-2, t) = 001 from its life detection sensors, it means that the it has detected life on Mars and the output y(t) = 1, otherwise y(t) = 0 (no life on Mars).
- This pattern recognizer should have

A. One state because it has one output

B. One state because it has one input

C. Two states because the input can be 0 or 1

[✓ D. More than two states because ....]

E. None of the above

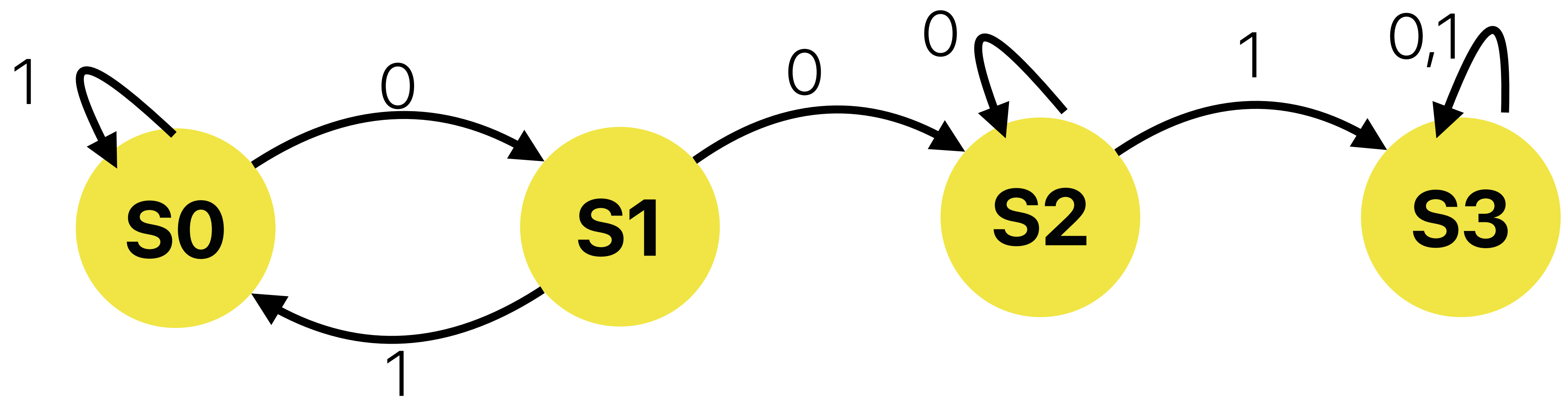
Outline

- Finite State Machines
- The Basic Form of Memory
- Clock

Finite-State Machines (cont.)

Poll close in 1:30

What is the longest string this FSM can recognize without visiting any state more than once?



A. 1

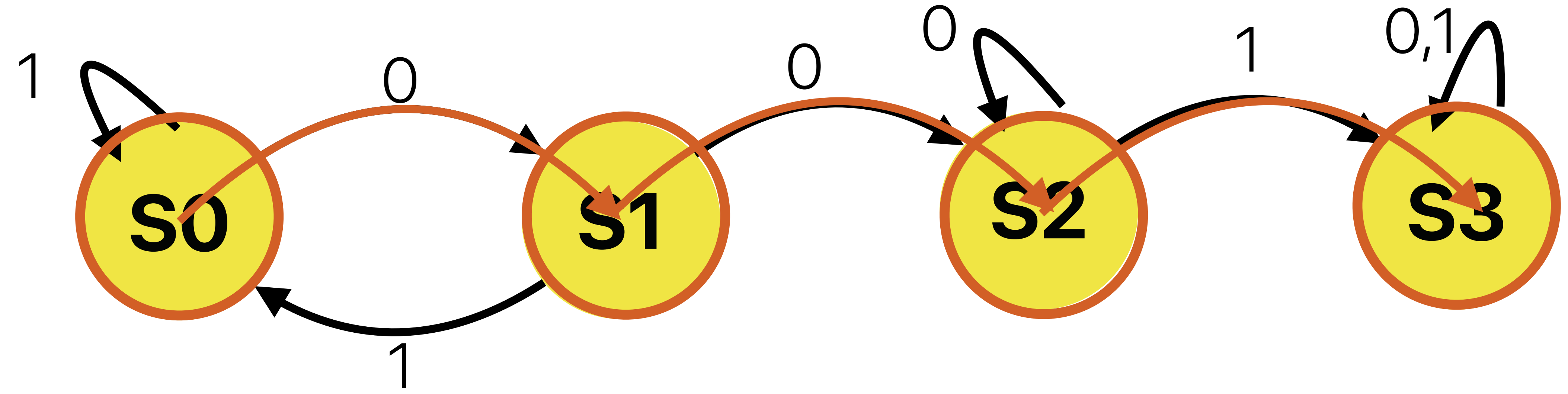
B. 2

C. 3

D. 4

E. None of the above

What is the longest string this FSM can recognize without visiting any state more than once?



A. 1

B. 2

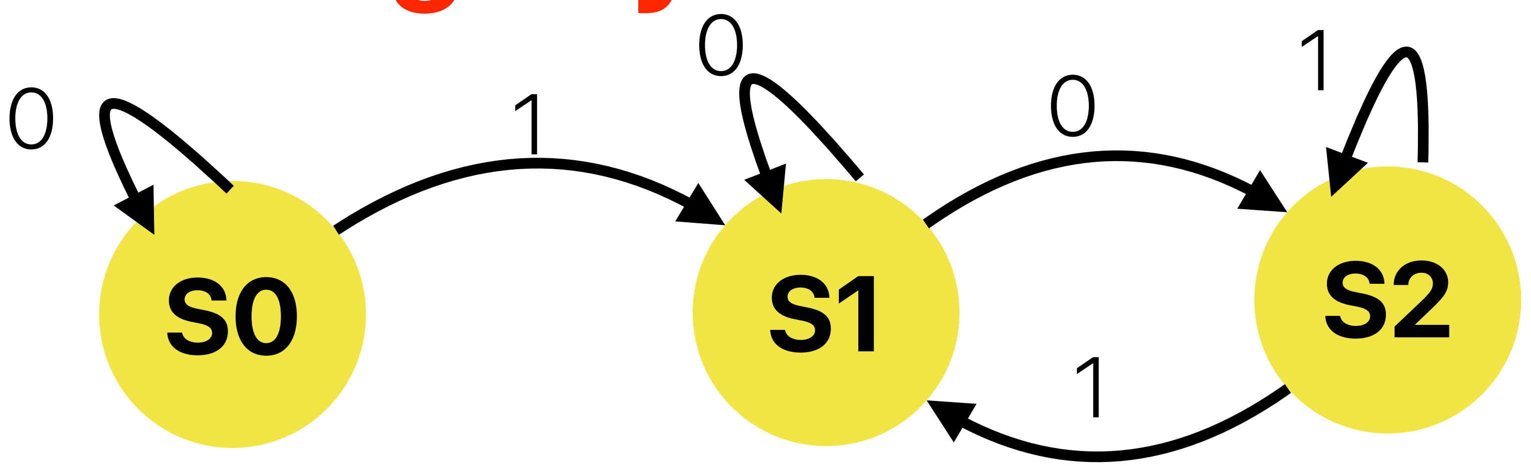
[✓ C. 3]

D. 4

E. None of the above

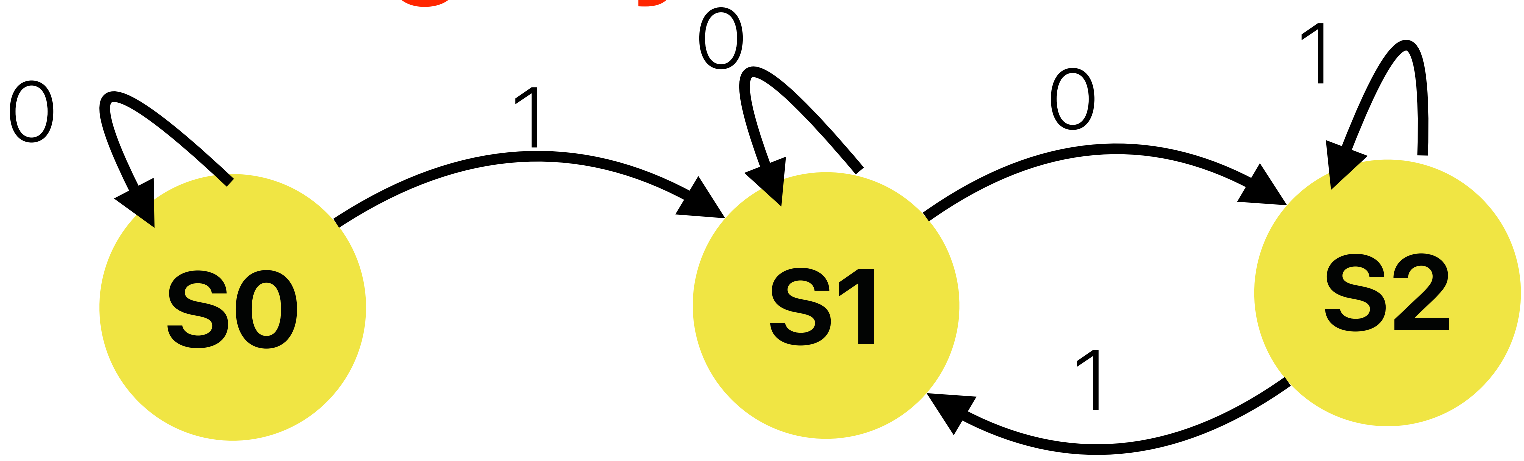
Poll close in 1:30

What is the longest string this FSM can recognize without visiting any state more than once?



A. 1
B. 2
C. 3
D. 4
E. None of the above

What is the longest string this FSM can recognize without visiting any state more than once?



A. 1

[✓ B. 2]

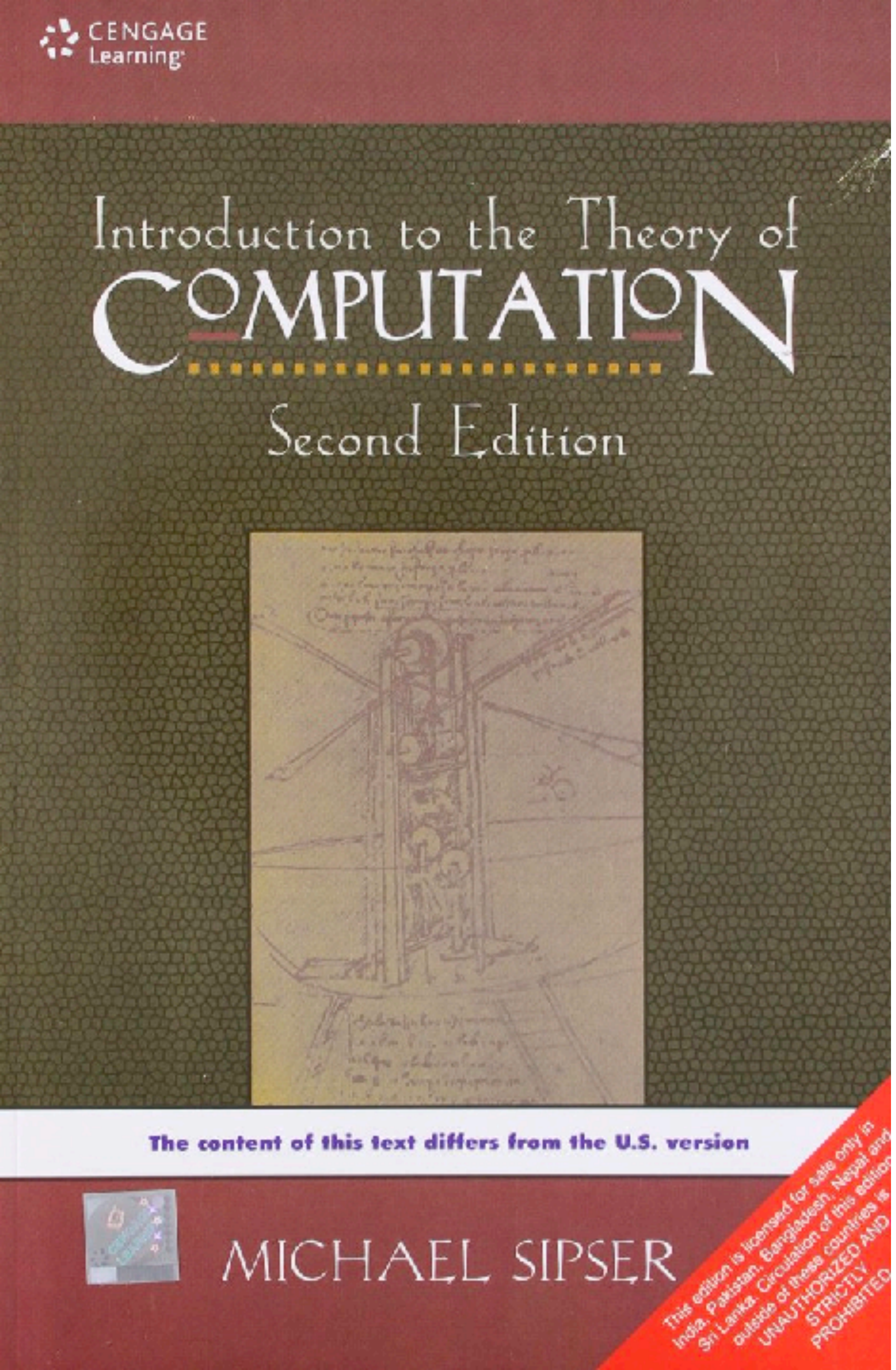
C. 3

D. 4

E. None of the above

Generalization

- Given a string **s** and a FSM **M** that accepts/recognize **s**:
  - If |**s**| > |**Q**|, then when **M** processes **s**, it must visit one (or more) state(s) more than once
    - For a second, let's just consider one state (**q_x**) being twice visited
  - Let **y** be the substring of **s** that is read between the first and second times the twice-visited state (**q_x**) is visited
  - It must be the case that **y** could appear repeatedly in **s**:
    - s = [first part of s]y[last part of s] is accepted by the FSM, then also possible in this FSM:
    - [first part of s]yy[last part of s]
    - [first part of s]yyyyyyyyyyy[last part of s]
    - [first part of s][last part of s]


CENGAGE Learning
Introduction to the Theory of
COMPUTATION
Second Edition
The content of this text differs from the U.S. version
MICHAEL SIPSER
This edition is licensed for sale only in India, Pakistan, Bangladesh, Nepal and Sri Lanka. Circulation of this edition outside of these countries is UNAUTHORIZED AND STRICTLY PROHIBITED.

If you want to learn more … CS 150

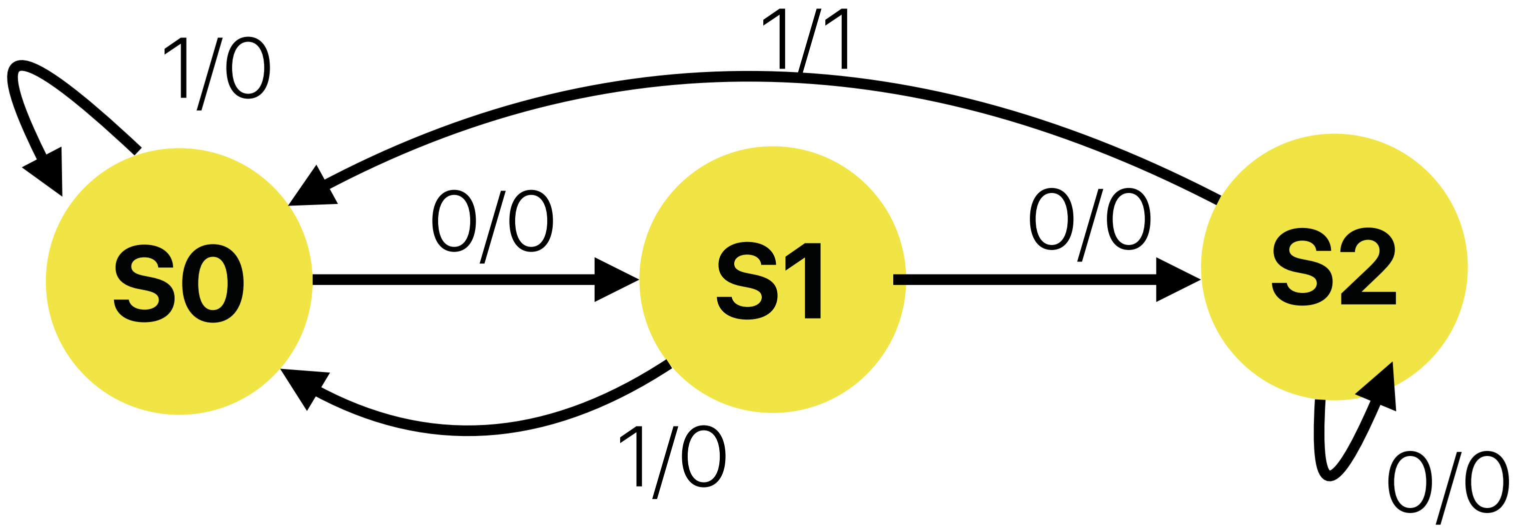
Poll close in 1:30

FSM for Life on Mars

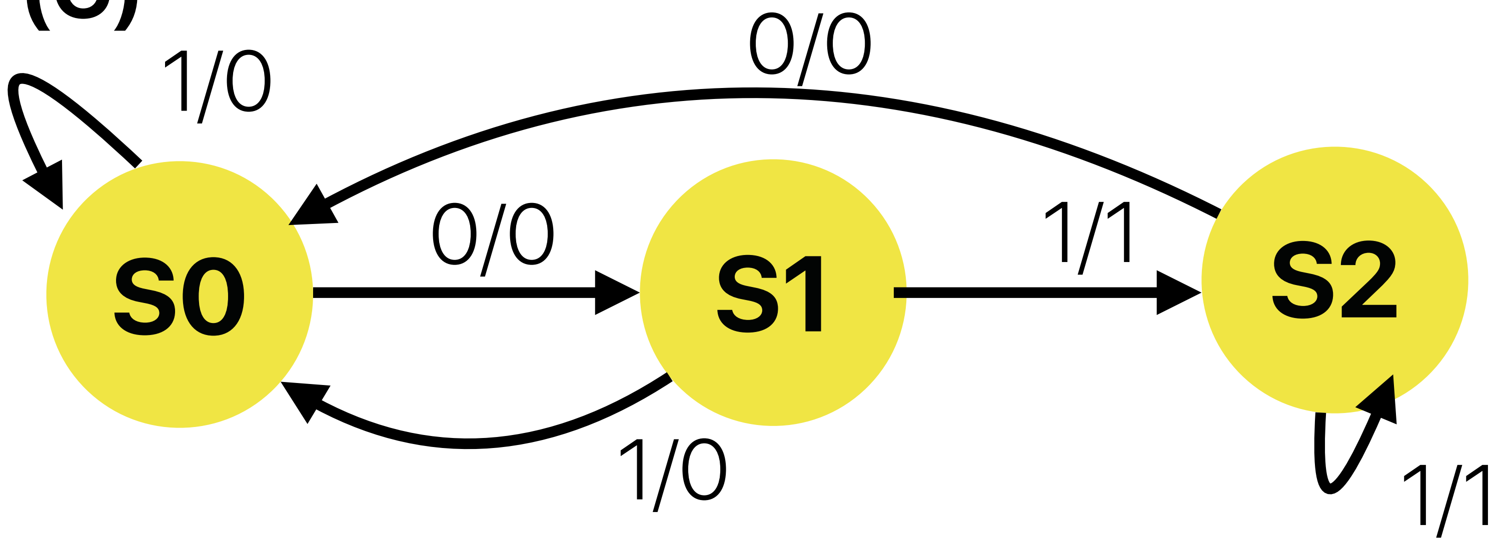
1/0 == Input 1/Output 0

- Which of the following diagrams is a correct FSM for the 001 pattern recognizer on the Mars rover? (If sees “001”, output “1”)

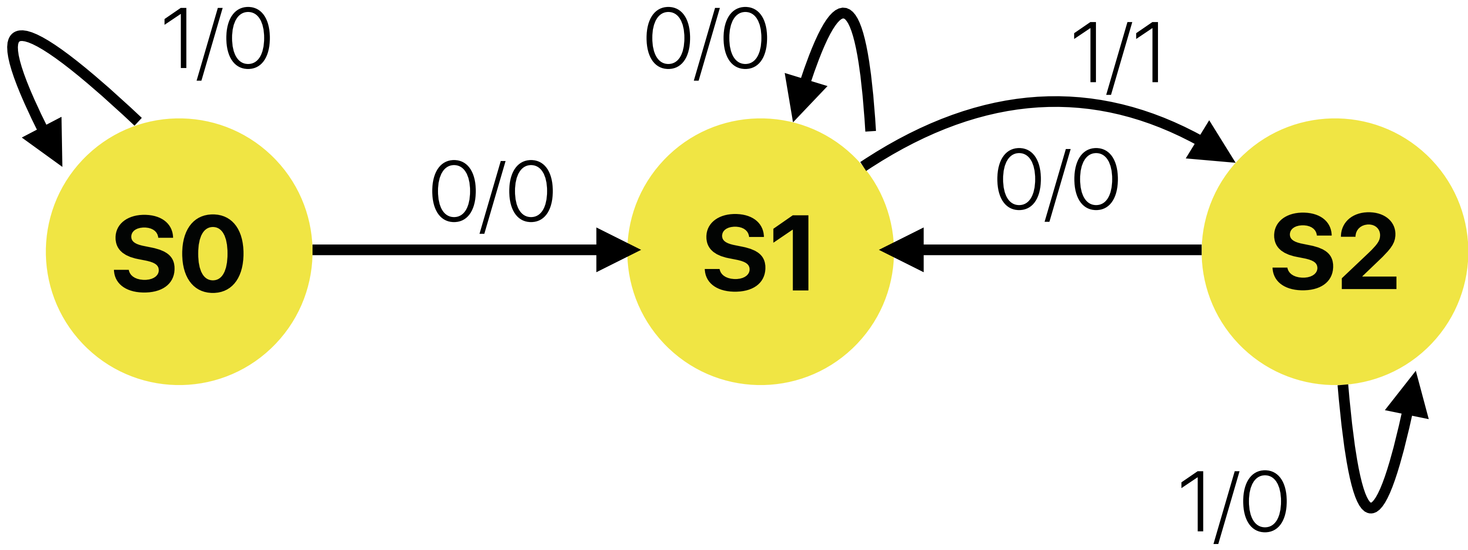
(A)



(C)



(B)



(D) All are correct

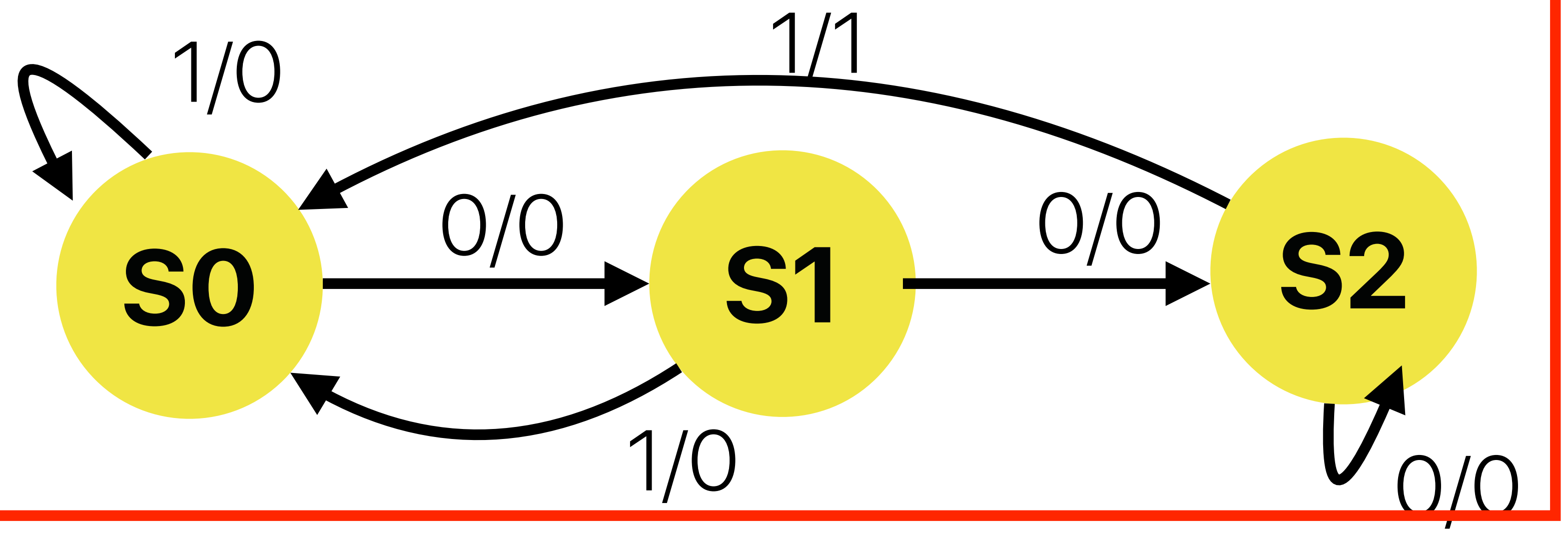
(E) None is correct

FSM for Life on Mars

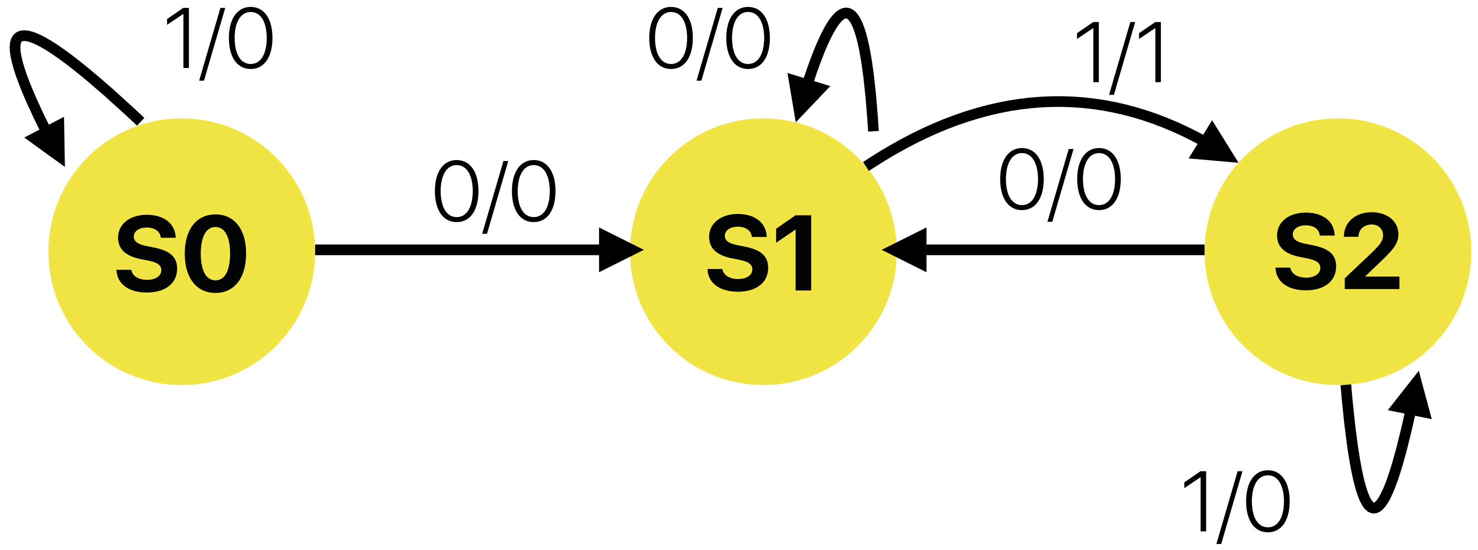
1/0 == Input 1/Output 0

- Which of the following diagrams is a correct FSM for the 001 pattern recognizer on the Mars rover? (If sees "001", output "1")

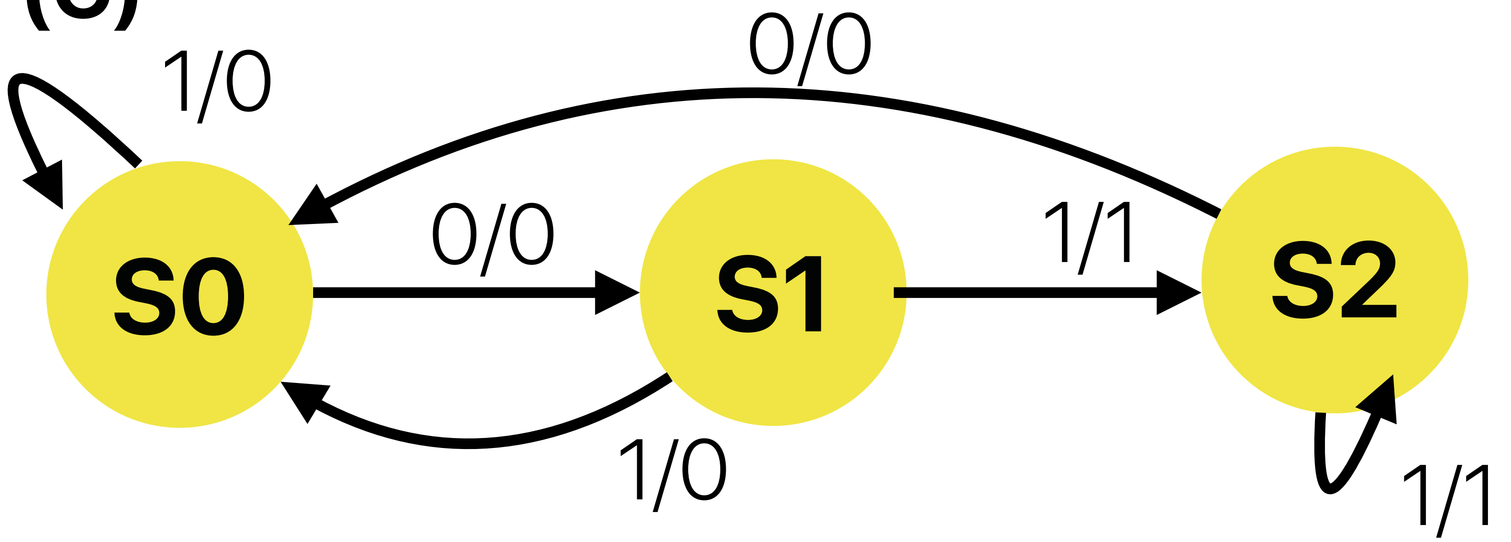
(A)



(B)



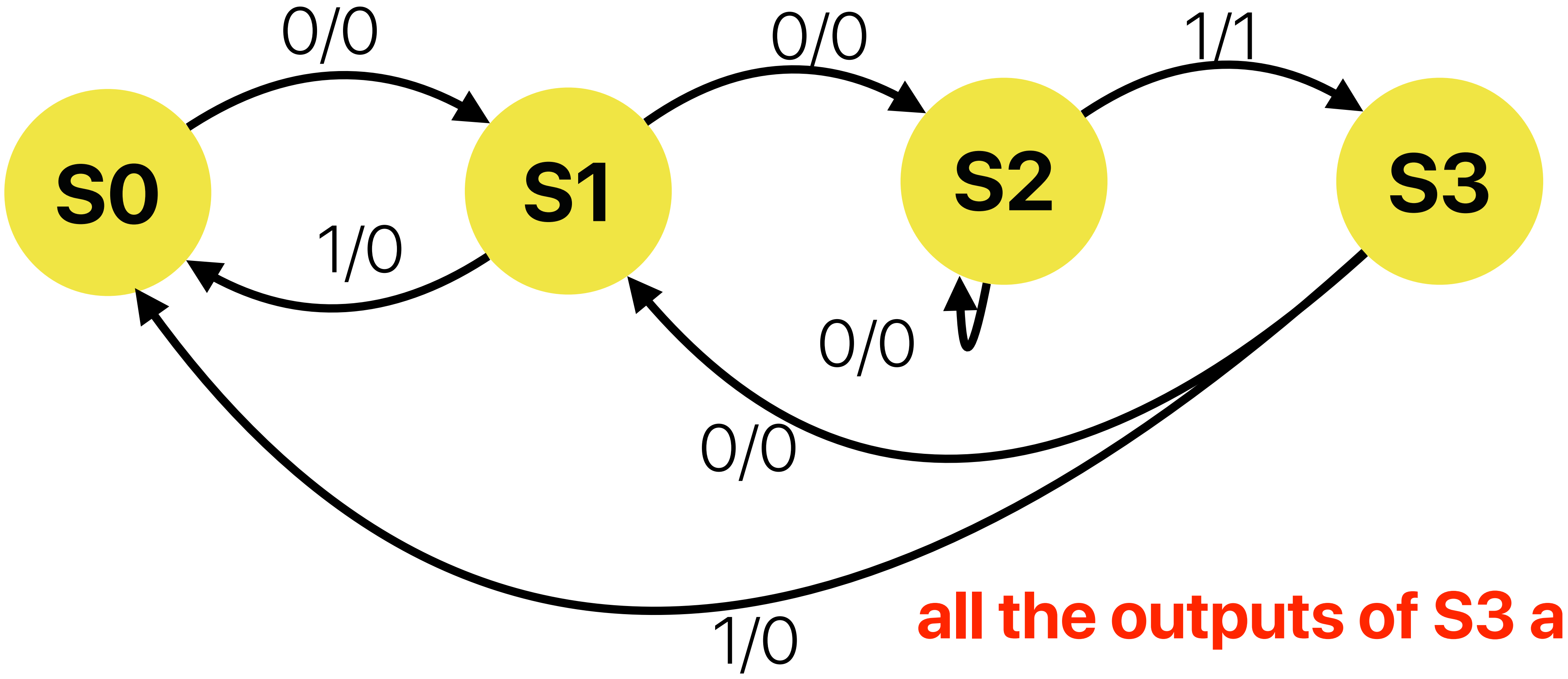
(C)



(D) All are correct

(E) None is correct

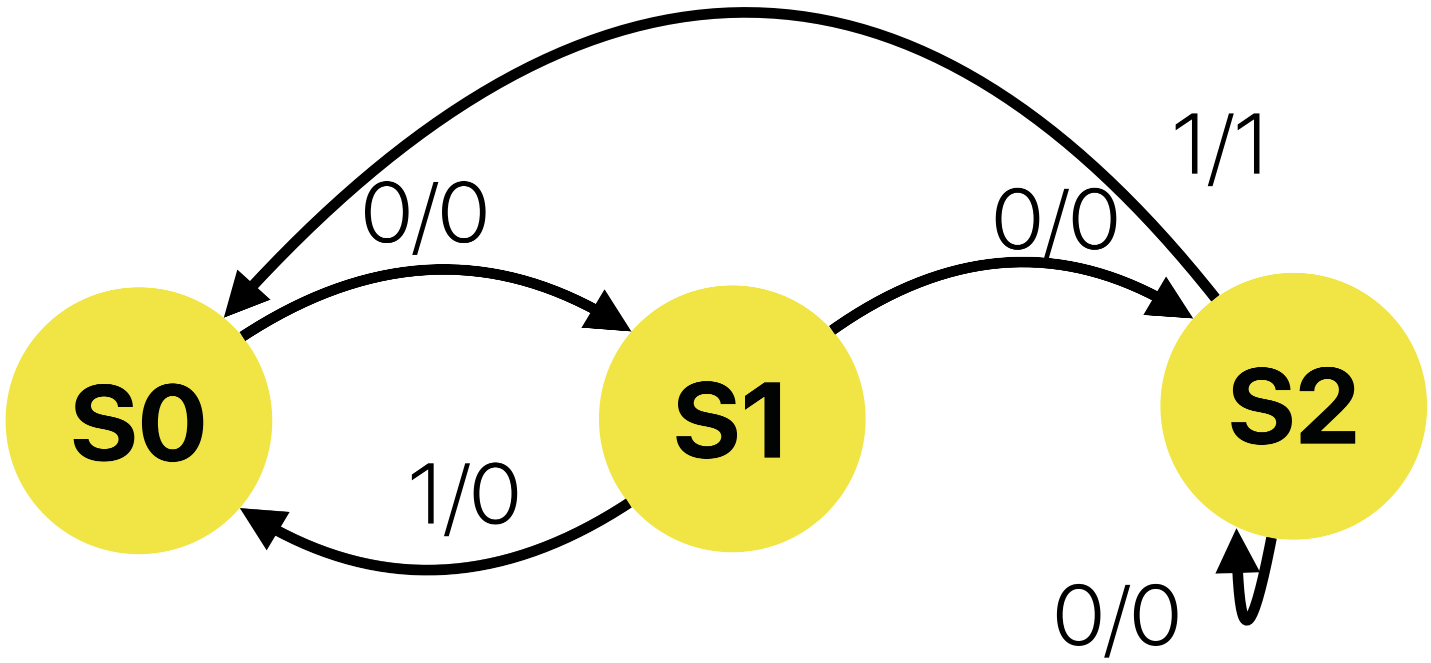
FSM for Life on Mars



all the outputs of S3 are equal to S0!

Merge S3 into S0

FSM for Life on Mars



Merge S3 into S0

State Transition Table of Life on Mars

Current State	Next State Input 0	 1
S0 — something else	S1	S0
S1 — 0	S2	S0
S2 — 00	S2	S3
S3 — 001	S1	S0

Poll close in 1:30

FSM 101

- Mars rover has a binary input x. When it receives the input sequence $x(t-2, t)$ = **101** from its life detection sensors, it means that the it has detected life on Mars and the output $y(t) = 1$, otherwise $y(t) = 0$ (no life on Mars).
- How many states in the FSM of the pattern recognizer should have

A. 1

B. 2

C. 3

D. 4

E. None of the above

State Transition Table of Life on Mars

Current State	Next State	
	Input	
	0	1
S0 — something else	S0	S1
S1 — 1	S2	S1
S2 — 10	S0	S3
S3 — 101	S2	S1

FSM 101

- Mars rover has a binary input x. When it receives the input sequence x(t-2, t) = **101** from its life detection sensors, it means that the it has detected life on Mars and the output y(t) = 1, otherwise y(t) = 0 (no life on Mars).
- How many states in the FSM of the pattern recognizer should have
  A. 1
  B. 2
  [✓ C. 3]
  D. 4
  E. None of the above

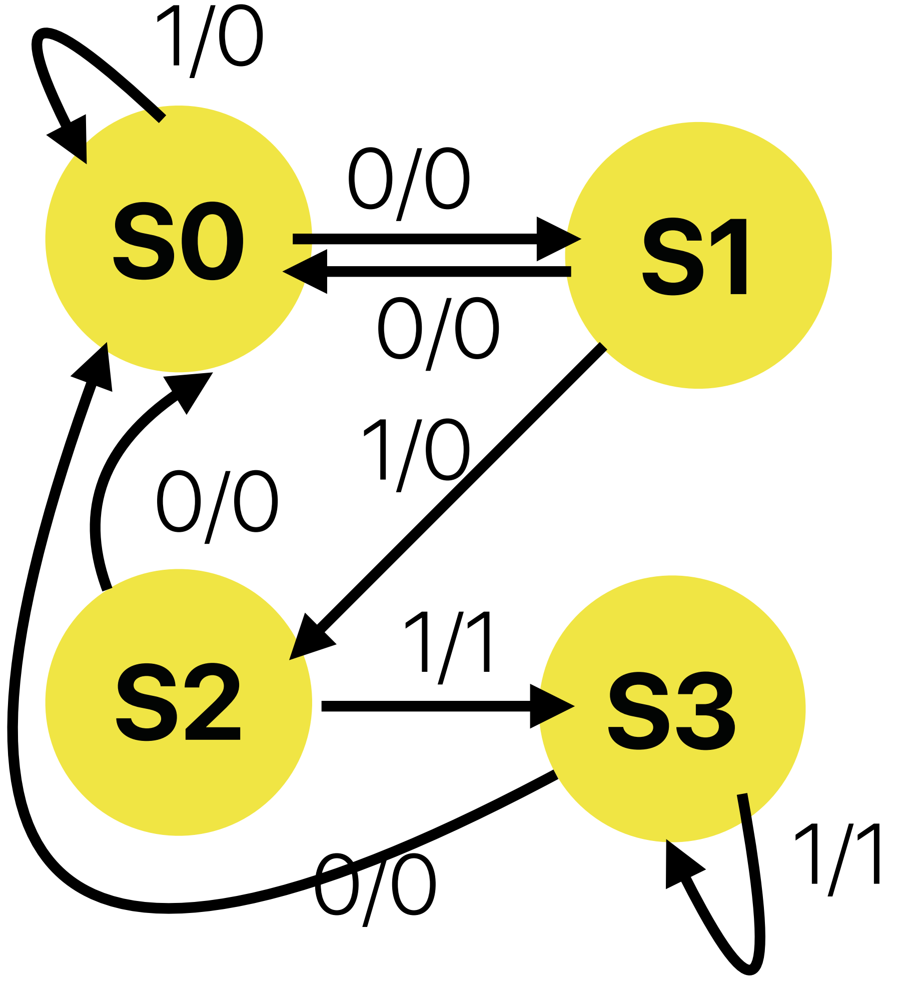
Poll close in 1:30

FSM 101

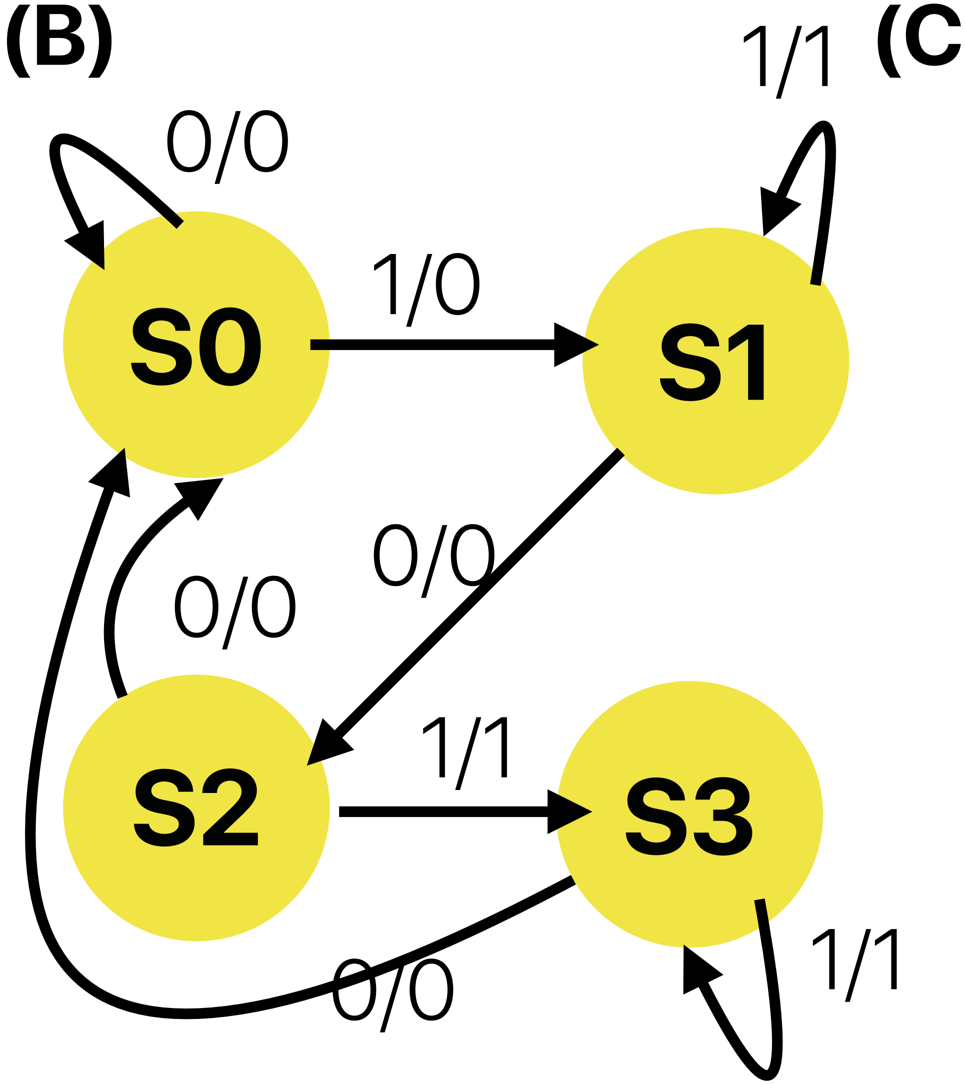
1/0 == Input 1/Output 0

- Which of the following diagrams is a correct FSM for the “101" pattern recognizer? (If sees “101”, output “1”)

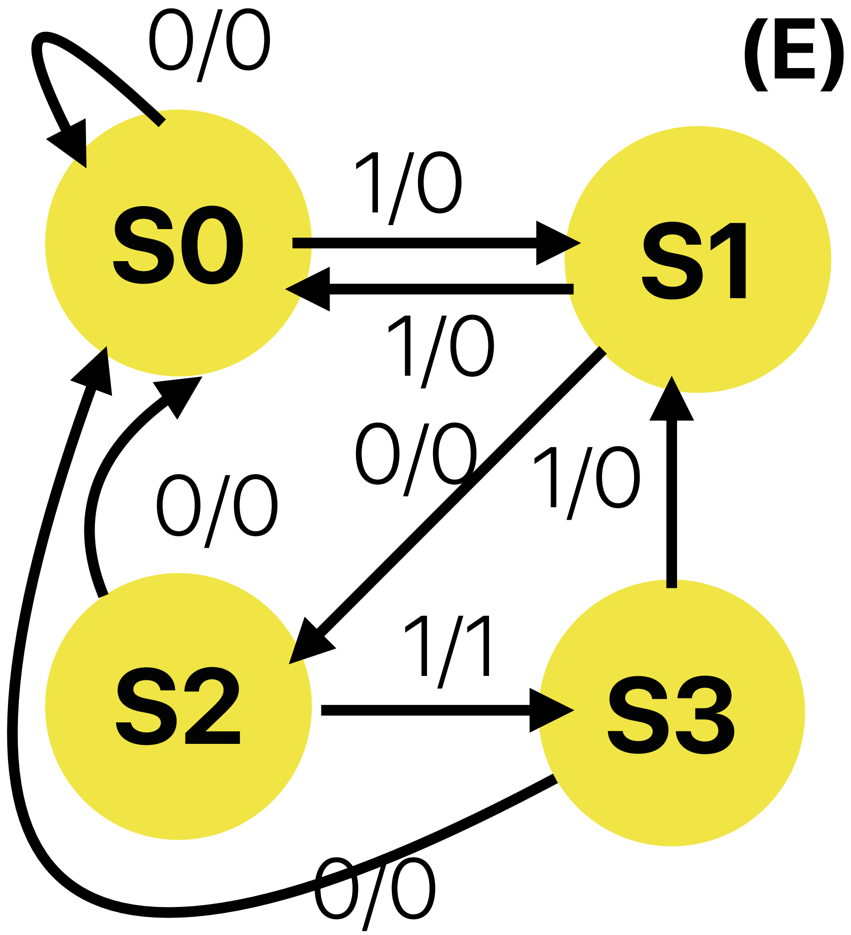
(A)



(B)



(C)



(D) All are correct

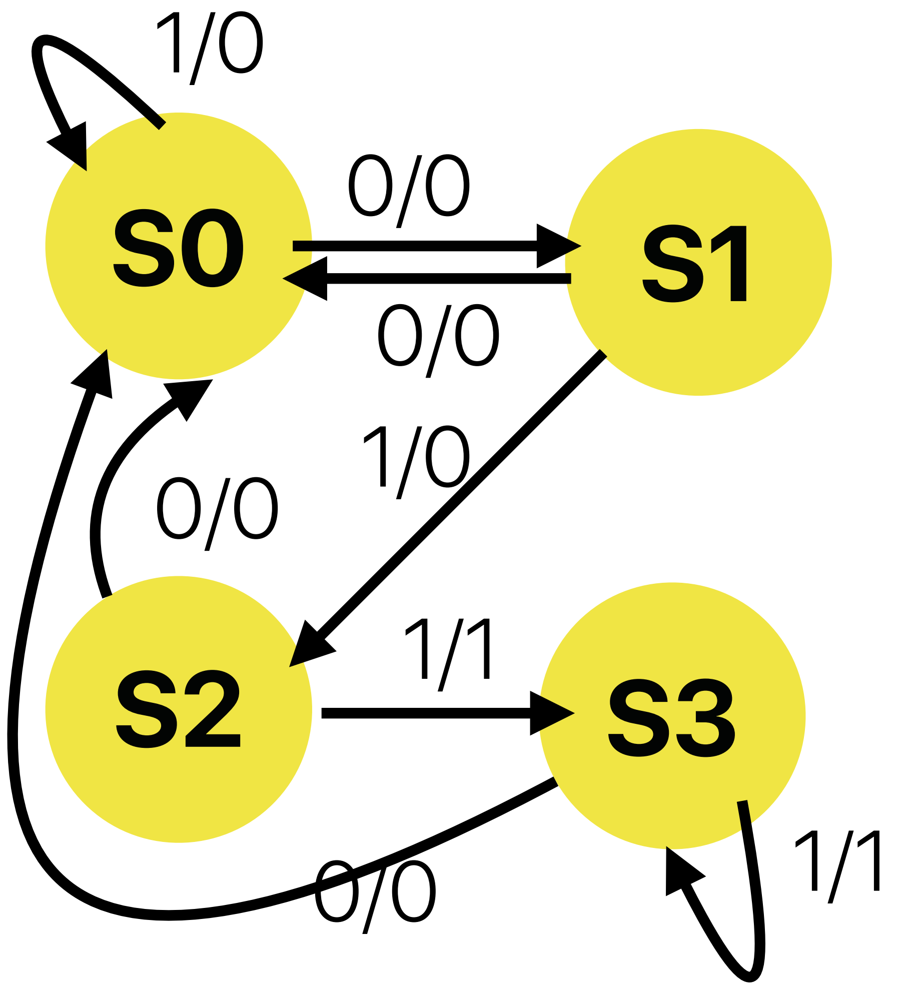
(E) None is correct

FSM 101

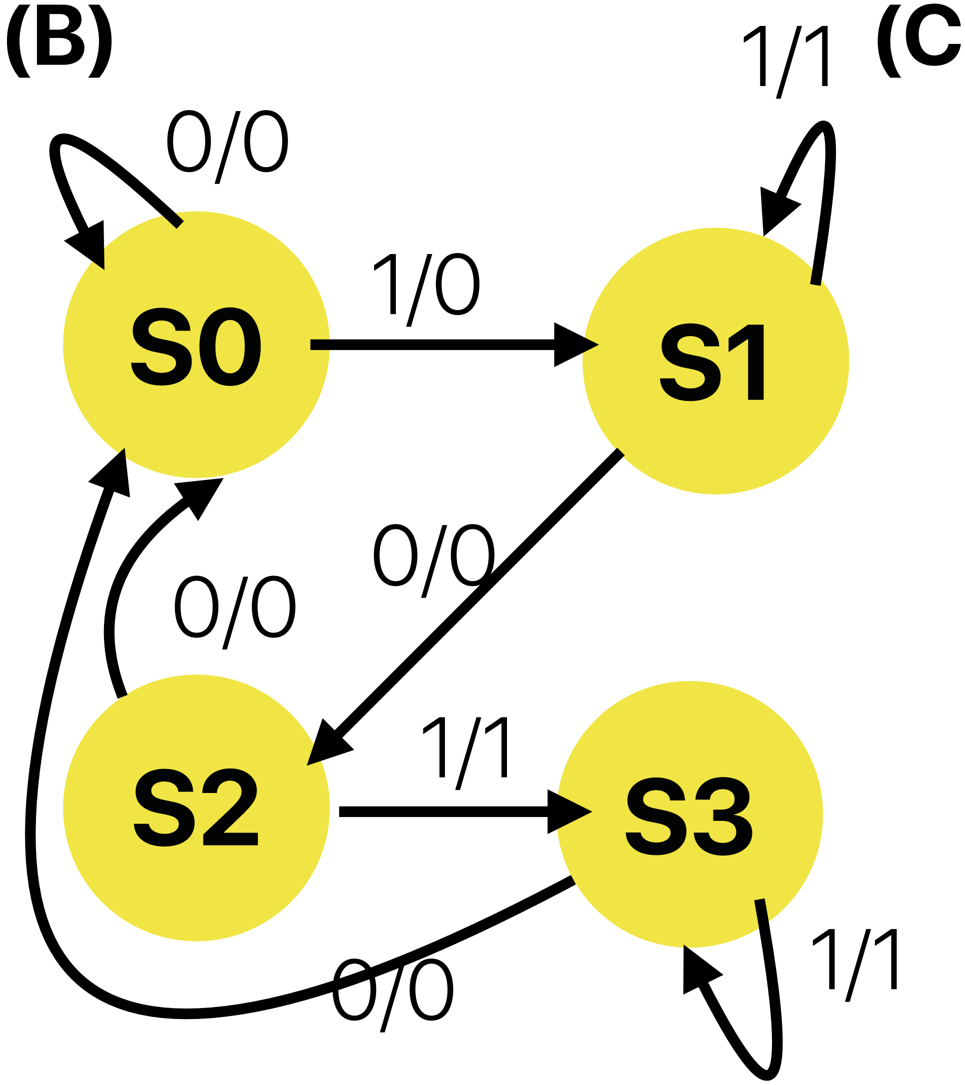
1/0 == Input 1/Output 0

- Which of the following diagrams is a correct FSM for the “101" pattern recognizer? (If sees “101”, output “1”)

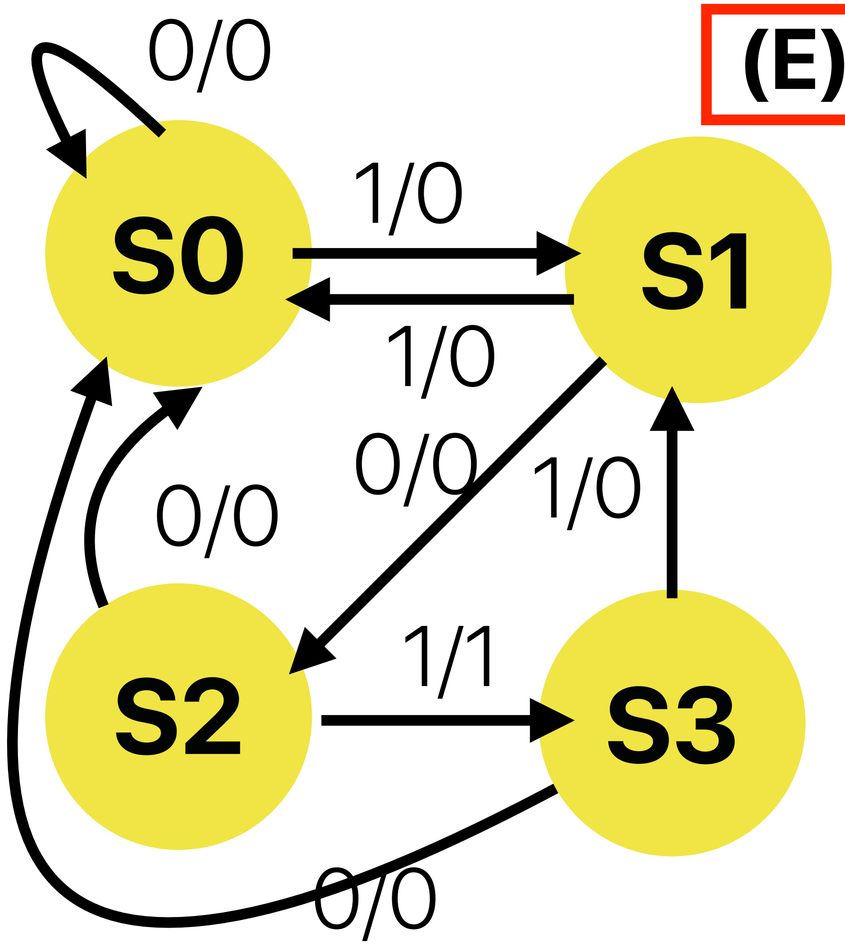
(A)



(B)



(C)



(D) All are correct

(E) None is correct

How make FSM true?

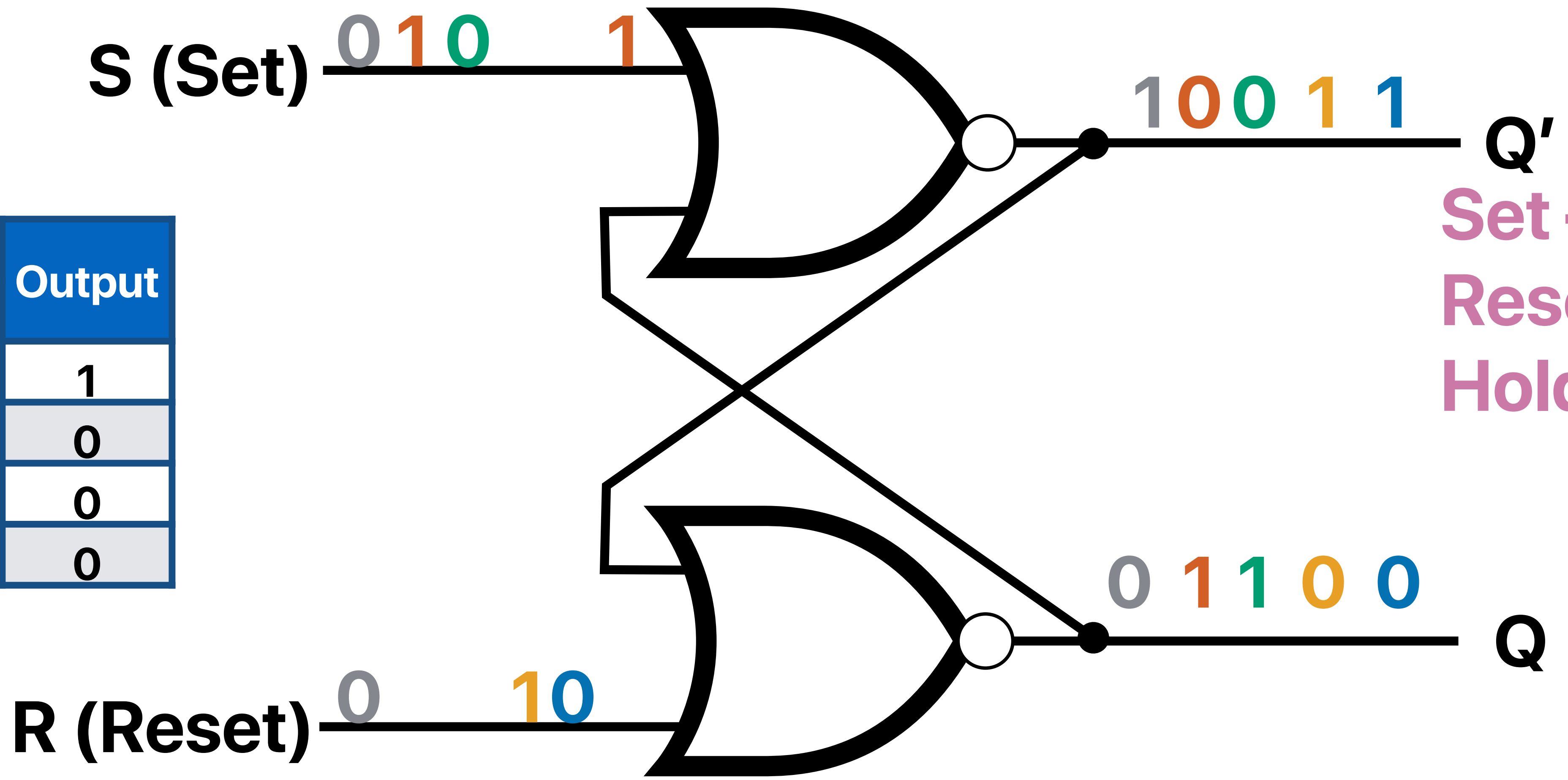
What do we need to physically implement the timer?

- A set of logic to display the remaining time **— we know how to do this already**
- A logic to keep track of the "current state" **— memory**
- A set of logic that uses the "current state" and "a new input" to transit to a new state and generate the output **— we also know how to build this**

The basic form of memory

SR-Latch: the very basic "memory"

Input		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0



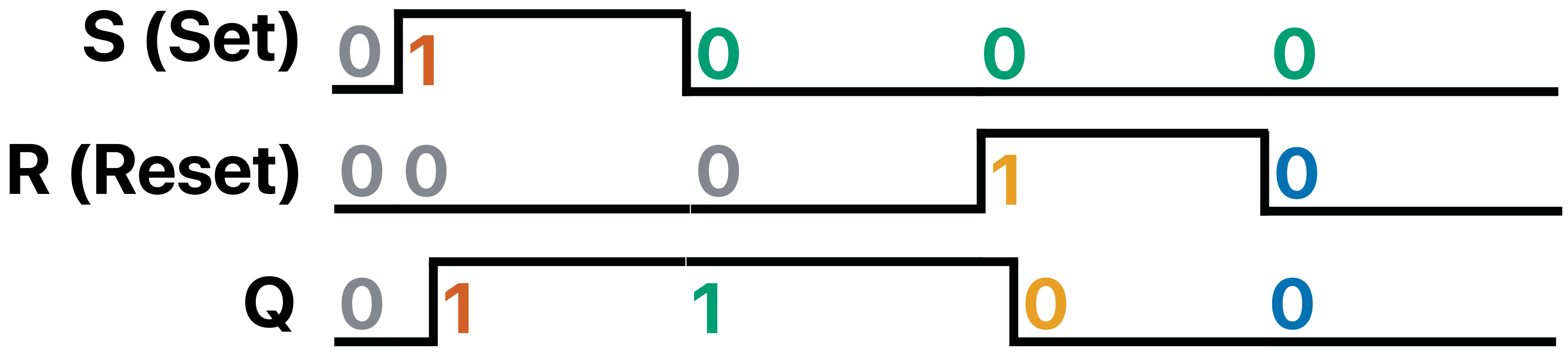
Set — Make the "stored bit 1"

Reset — Make the "stored bit 0"

Hold — both set/reset are 0

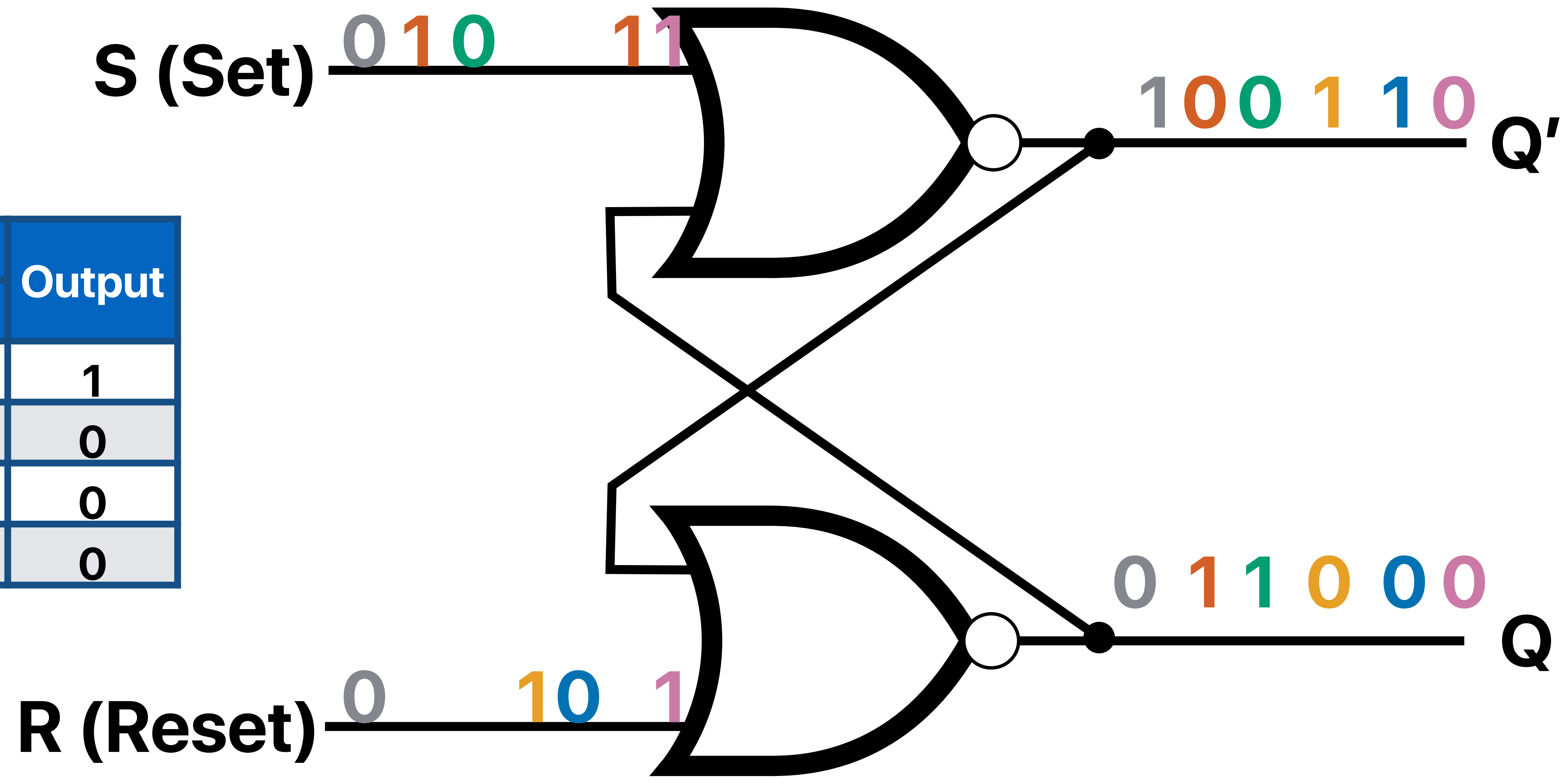
The circuit has memory!

S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	
1	1	1	



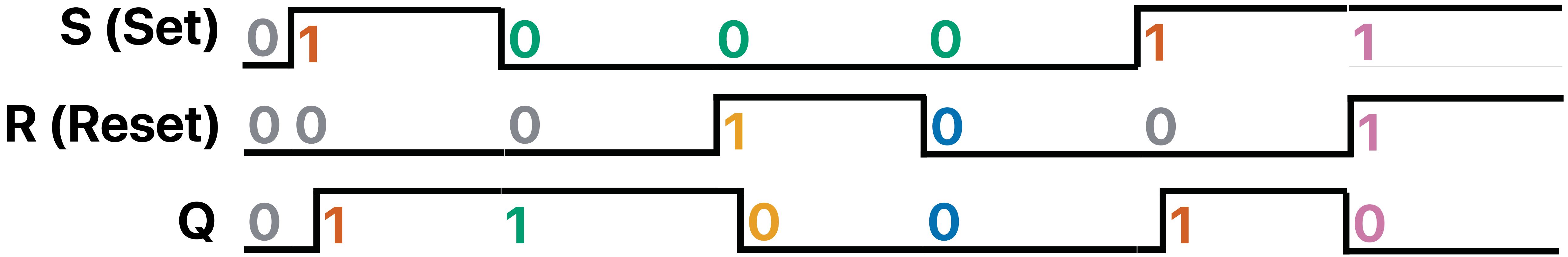
What if S/R are both 1s?

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

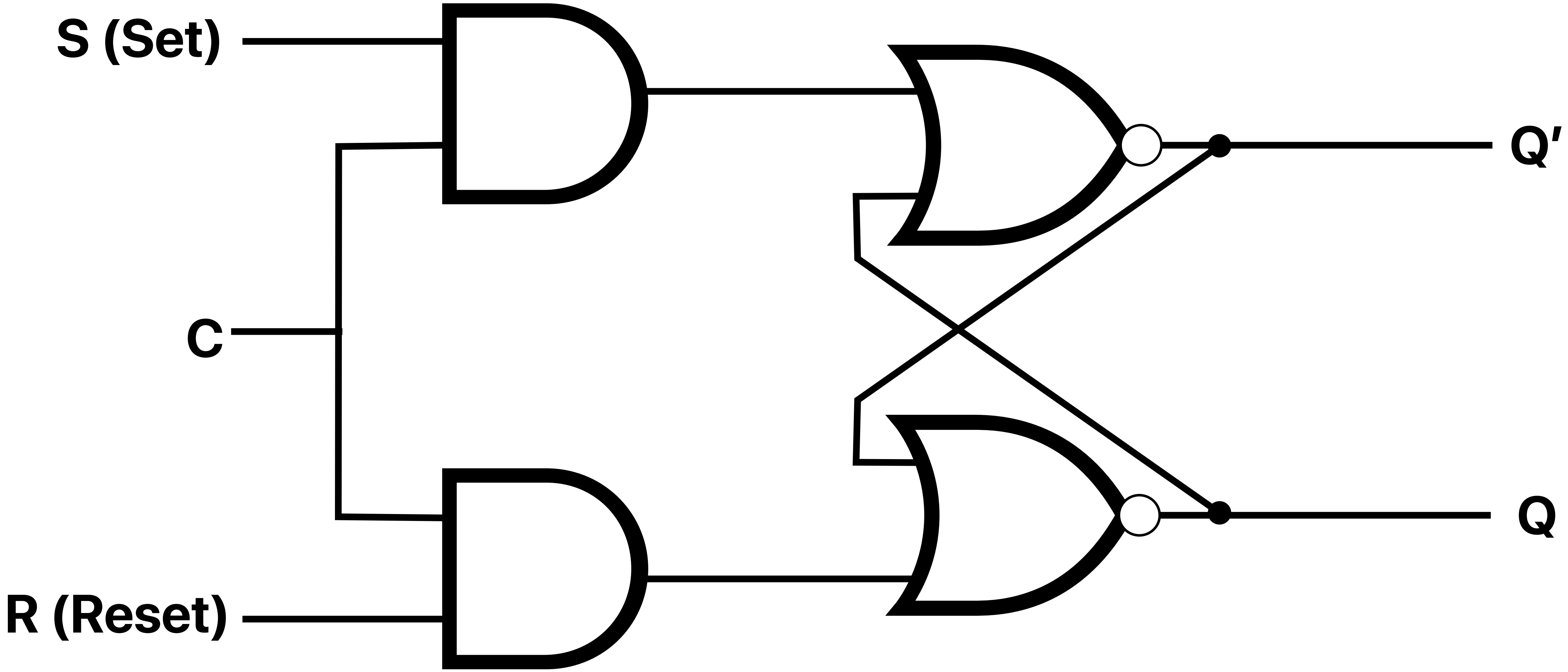


S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Doesn’t function if both are 1s!



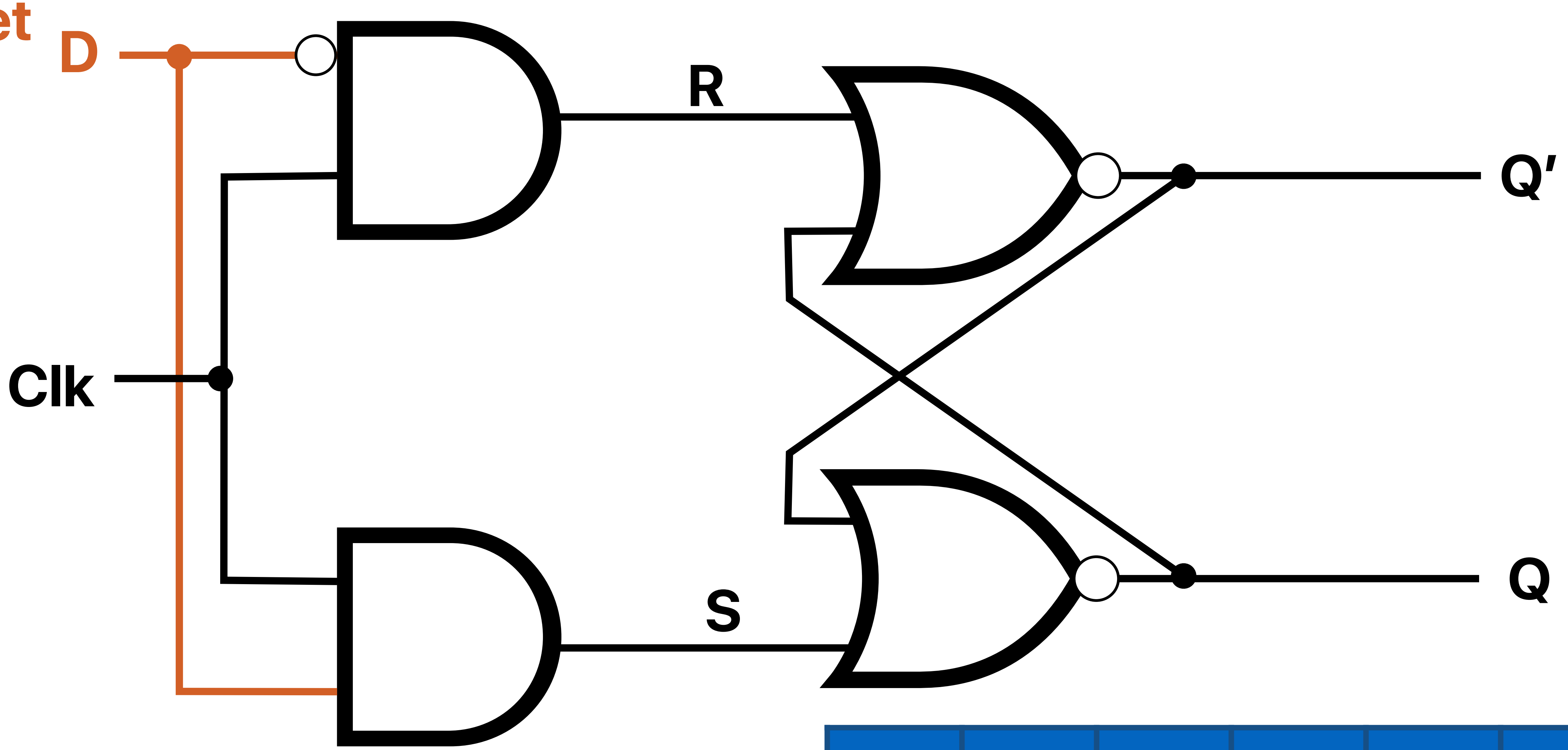
Add an input! — Level-Sensitive SR Latch



- Change C to 1 only after S and R are **stable**
- C is a signal aware of the **timing** of gates — **clock**

D-Latch

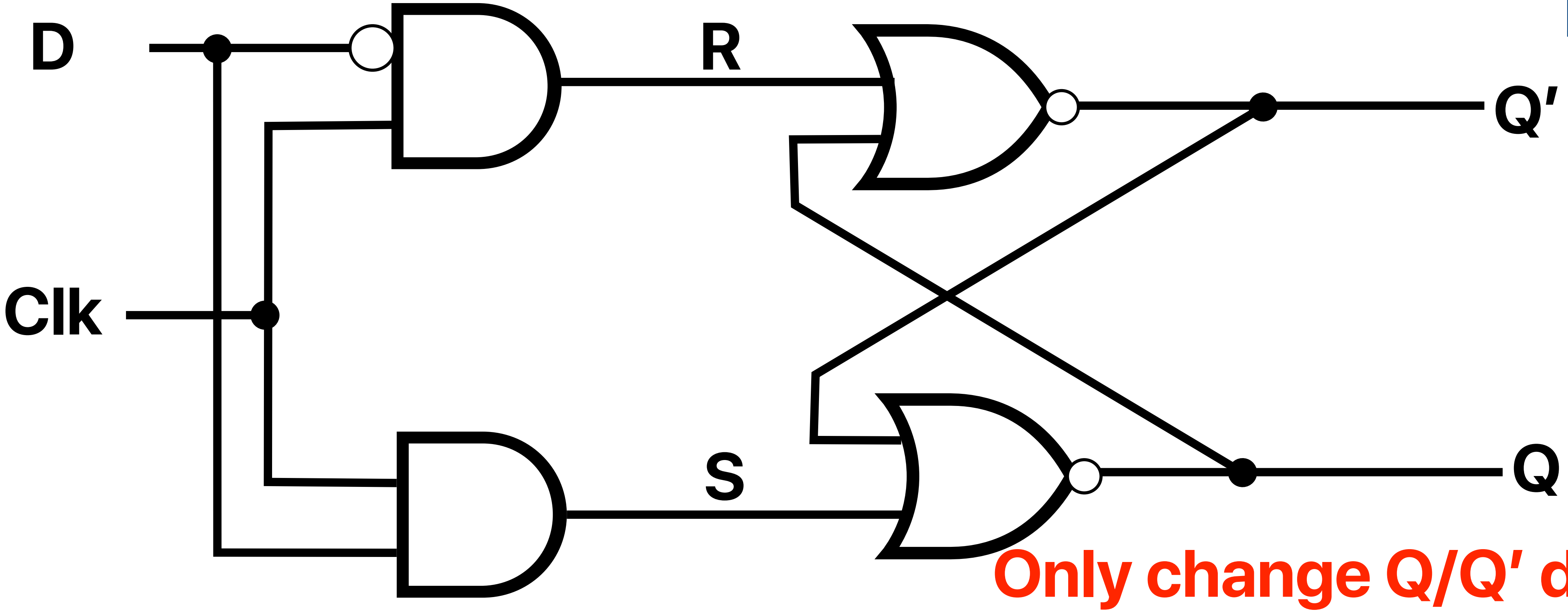
We will never get 1, 1 in this way



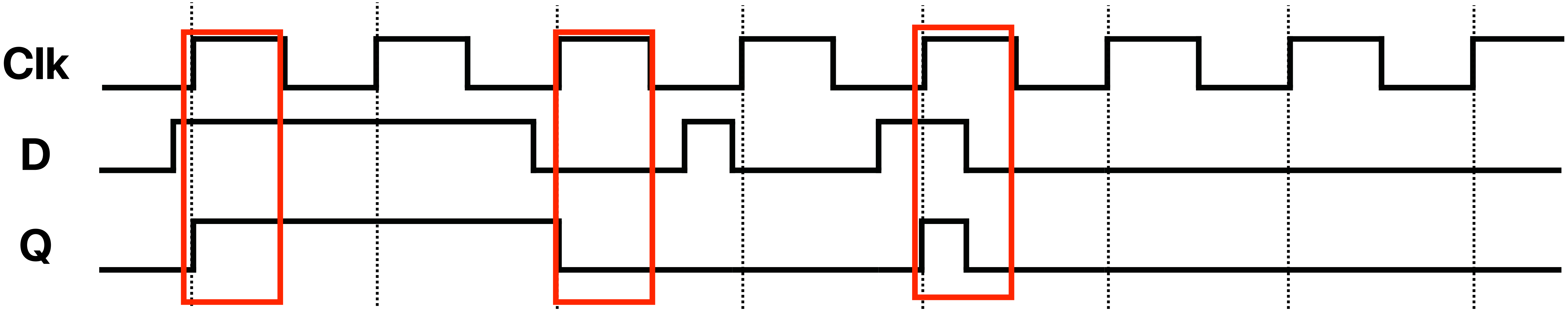
CLK	D	D′	S	R	Q	Q′
0	X	X′	0	0	Qprev	Qprev′
0	0	1	0	1	0	1
1	1	0	1	0	1	0

D-Latch

CLK	D	D′	S	R	Q	Q′
0	X	X′	0	0	Qprev	Qprev′
0	0	1	0	1	0	1
1	1	0	1	0	1	0

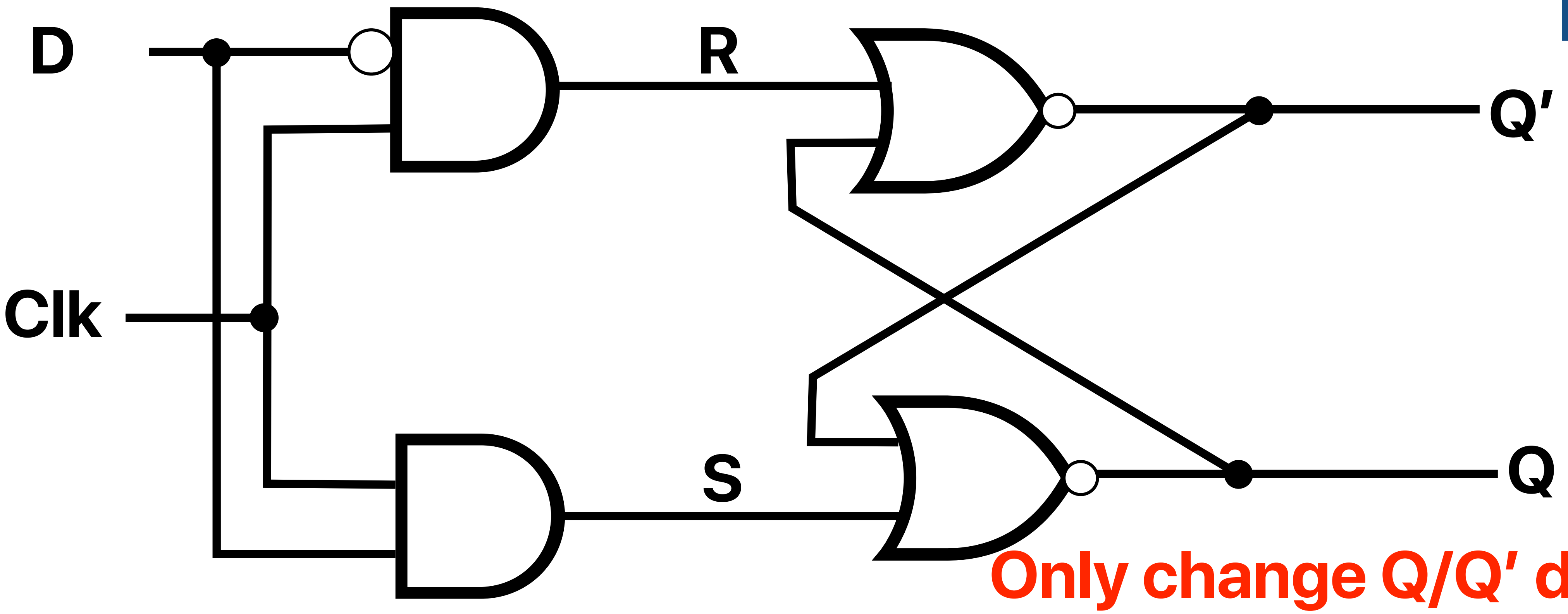


Only change Q/Q′ during positive clock edges

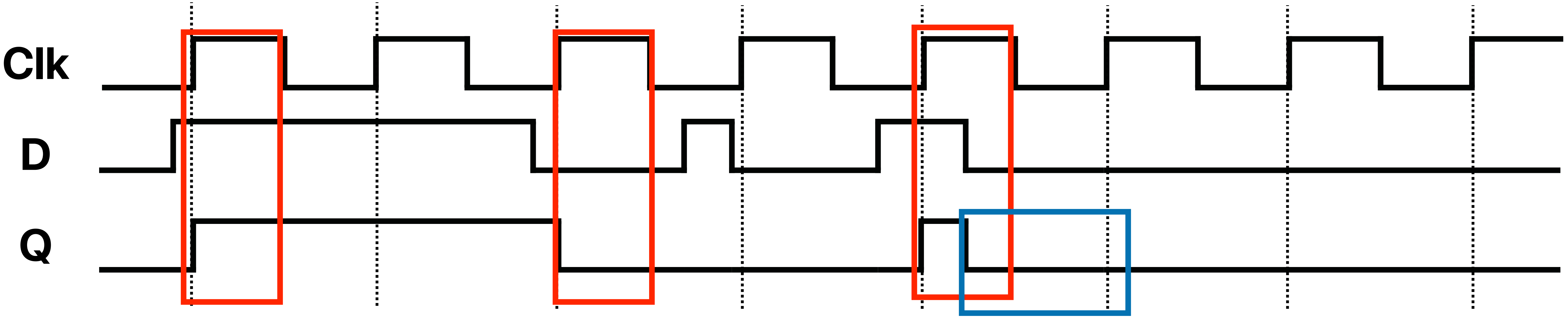


D-Latch

CLK	D	D′	S	R	Q	Q′
0	X	X′	0	0	Qprev	Qprev′
0	0	1	0	1	0	1
1	1	0	1	0	1	0

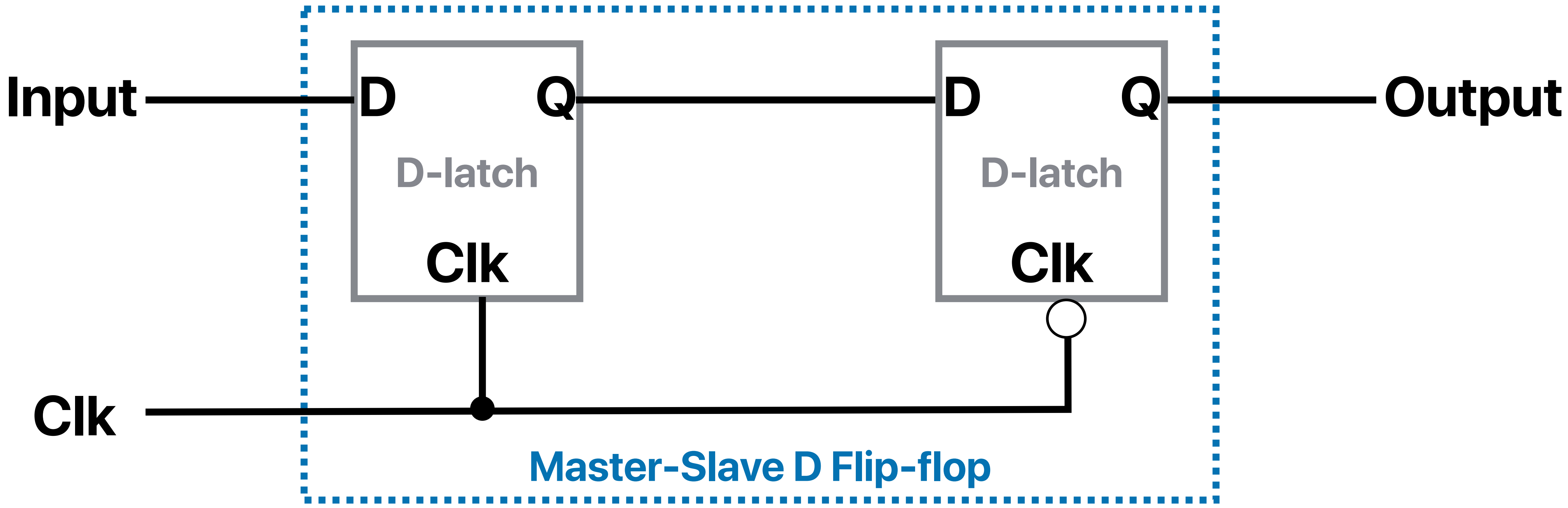


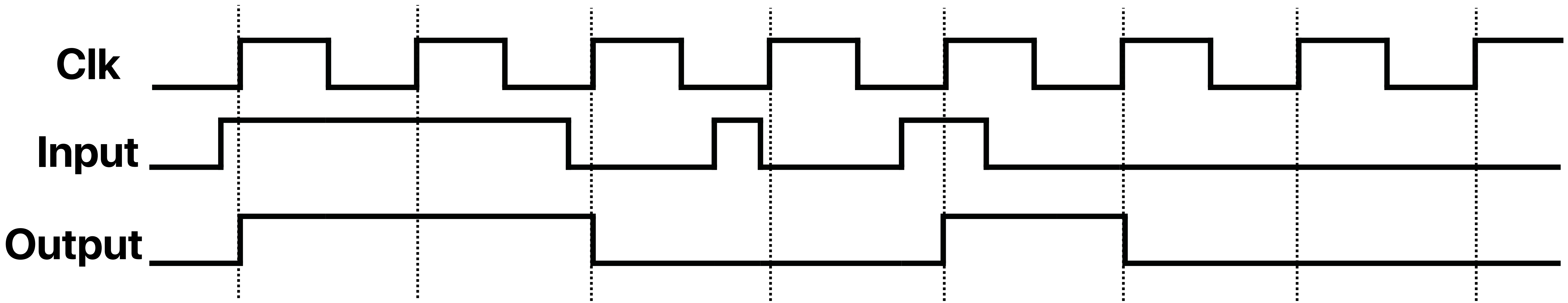
Only change Q/Q′ during positive clock edges



Output doesn’t hold for the whole cycle

D flip-flop


Input
D
Q
D-latch
Clk
D
Q
D-latch
Clk
Output
Clk
Master-Slave D Flip-flop


Clk
Input
Output

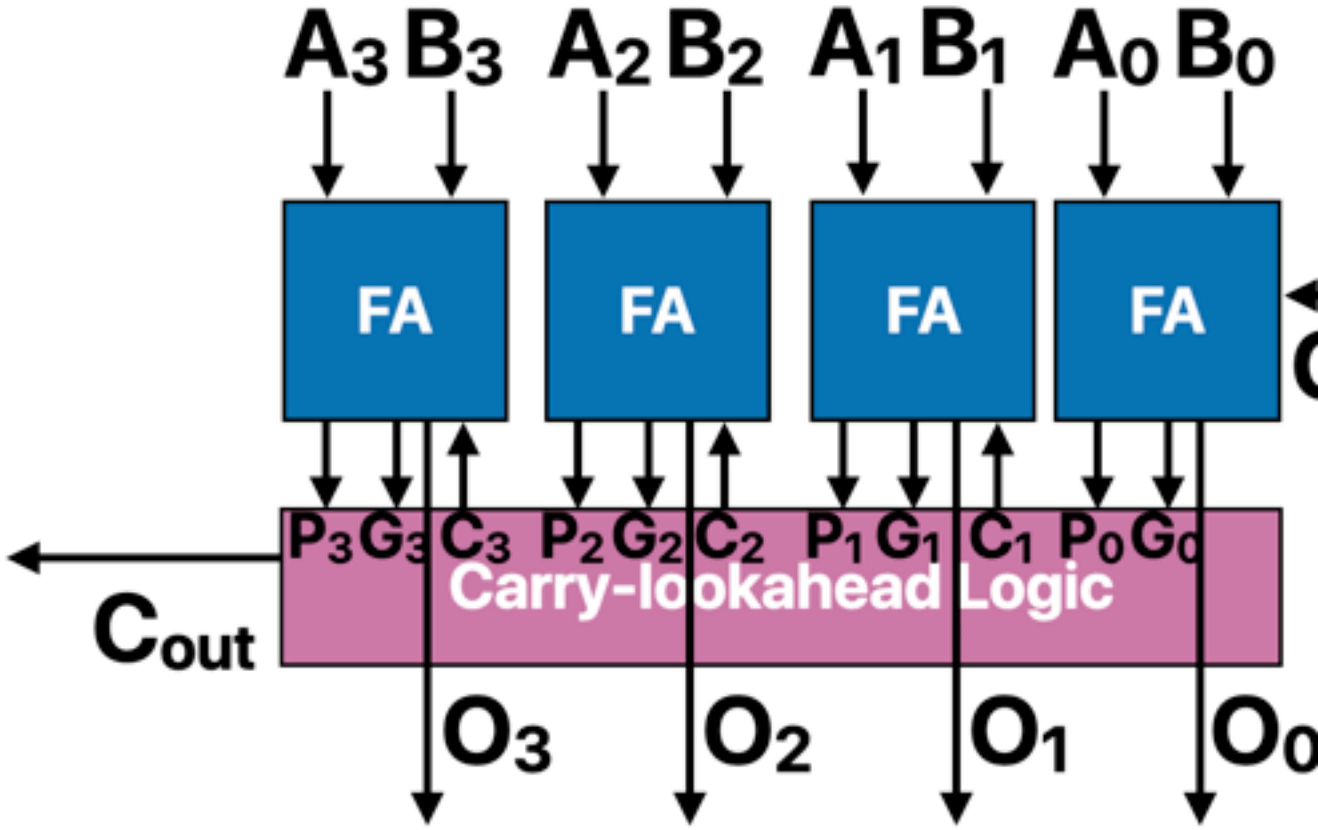
What do we need to physically implement the timer?

- A set of logic to display the remaining time **— we know how to do this already**
- A logic to keep track of the "current state" **— memory**
- A set of logic that uses the "current state" and "a new input" to transit to a new state and generate the output **— we also know how to build this**
- A control signal that helps us to transit to the right state at the right time **— clock**

The basic concept of “clock”

Poll close in 1:30

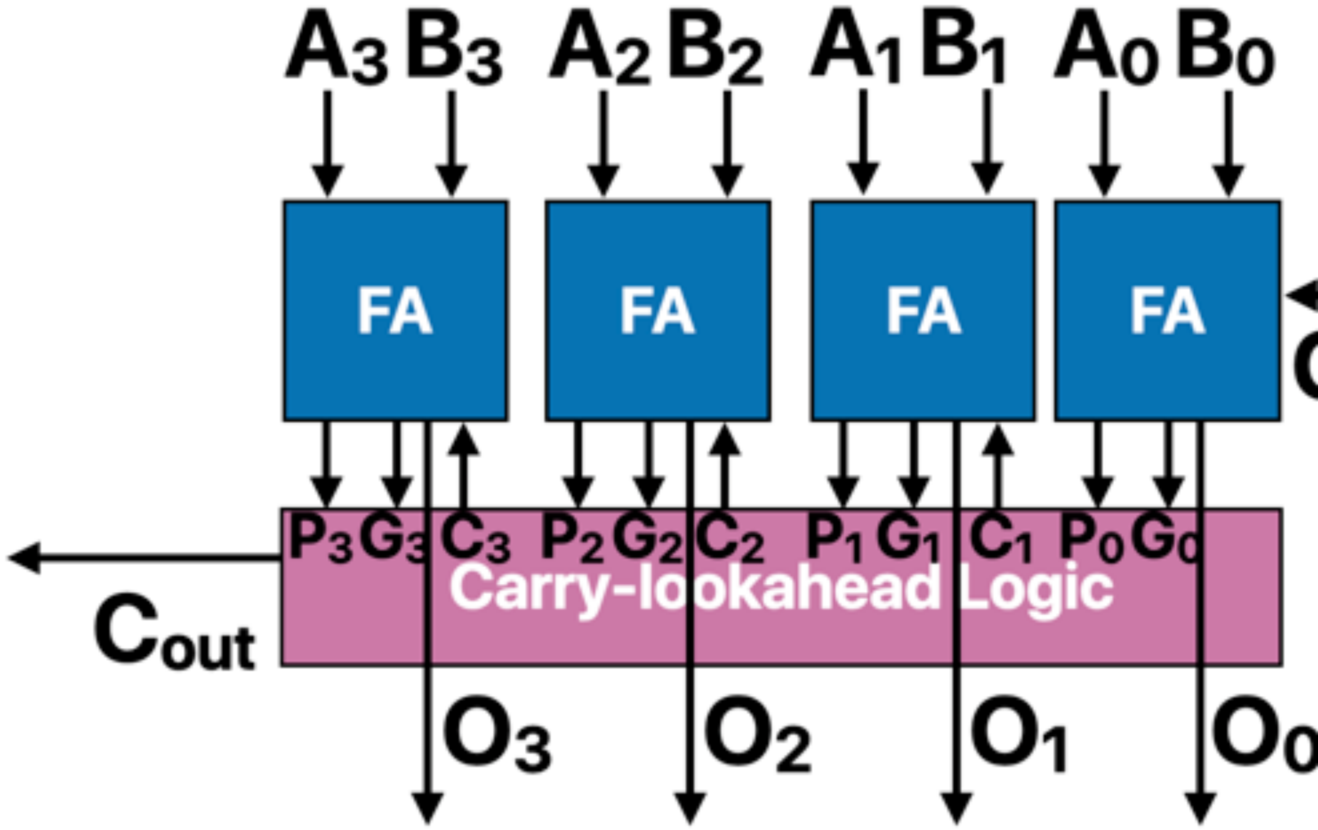
What if ?


A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
FA
FA
FA
FA
P_3 G_3 C_3 P_2 G_2 C_2 P_1 G_1 C_1 P_0 G_0
Carry-lookahead Logic
C_{out}
O_3
O_2
O_1
O_0

- Consider a 32-bit carry-lookahead adder built with 8 4-bit carry-lookahead adders. If we take the output after 4 gate delays and feed another input at that time, which of the following would be true?
  A. At the time we take the output, we can get the correct result
  B. At the time we take the output, we cannot get the correct result
  C. At the time we take the output, we cannot get the correct result, but we can get the correct result after another 8 gate delays

What if ?

- Consider a 32-bit carry-lookahead adder built with 8 4-bit carry-lookahead adders. If we take the output after 4 gate delays and feed another input at that time, which of the following would be true?



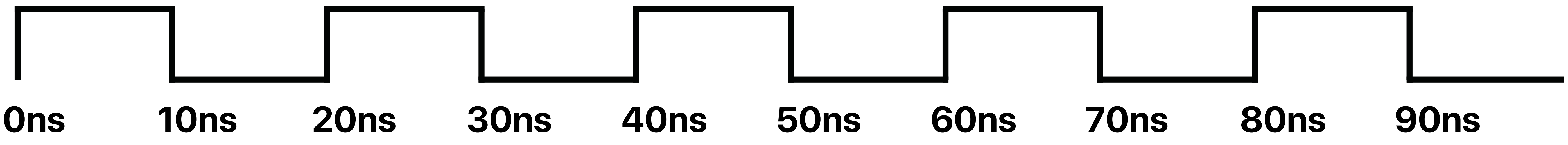
A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
FA
FA
FA
FA
P_3 G_3 C_3 P_2 G_2 C_2 P_1 G_1 C_1 P_0 G_0
Carry-lookahead Logic
C_{out}
O_3
O_2
O_1
O_0

A. At the time we take the output, we can get the correct result

[✓ B. At the time we take the output, we cannot get the correct result]

C. At the time we take the output, we cannot get the correct result, but we can get the correct result after another 8 gate delays

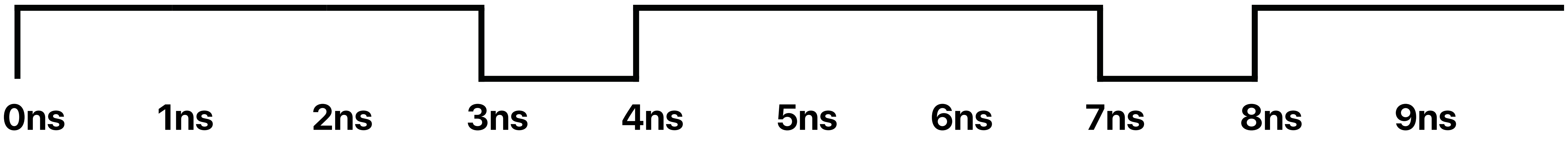
Clock signal



- Clock -- Pulsing signal for enabling latches; ticks like a clock
- Synchronous circuit: sequential circuit with a clock
- Clock period: time between pulse starts
  - Above signal: period = 20 ns
- Clock cycle: one such time interval
  - Above signal shows 3.5 clock cycles
- Clock duty cycle: time clock is high
  - 50% in this case
- Clock frequency: 1/period
  - Above : freq = 1 / 20ns = 50MHz;

Poll close in 1:30

Clock signal



- Regarding the above clock signal, please identify how many of the following statements are correct?

  ① Clock period of 4ns with 250MHz frequency

  ② Clock duty cycle 75%

  ③ Clock period of 1ns with 1GHz frequency

  ④ The above contains two complete clock cycles.

  A. 0

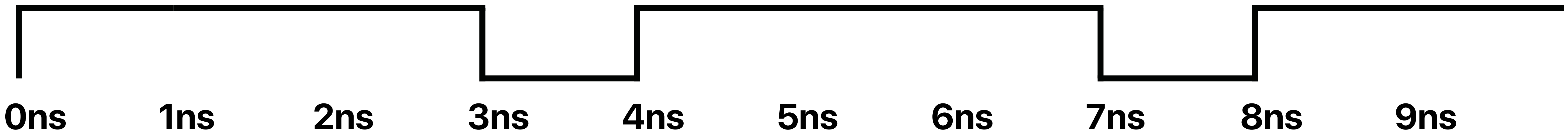
  B. 1

  C. 2

  D. 3

  E. 4

Clock signal



- Regarding the above clock signal, please identify how many of the following statements are correct?

  [✓ ①] Clock period of 4ns with 250MHz frequency

  [✓ ②] Clock duty cycle 75%

  ③ Clock period of 1ns with 1GHz frequency

  [✓ ④] The above contains two complete clock cycles.

  A. 0

  B. 1

  C. 2

  [✓ D. 3]

  E. 4

Announcement

- Lab 3 due 4/30
  - Watch the video and read the instruction BEFORE your session
  - There are links on both course webpage and iLearn lab section
  - Submit through iLearn > Labs
- Assignment #3 due next Tuesday — **Chapter 3.6-3.16 & 4.1-4.4 & 4.8-4.9**
- Midterm on 5/7 during the lecture time, access through iLearn
  - No late submission is allowed — make sure you will be able to take that at the time
  - Covers: Chapter 1, Chapter 2, Chapter 3.1 — 3.12, Chapter 3.15 & 3.16, Chapter 4.1— 4.9
  - Midterm review next Tuesday (5/5) will reveal more information (e.g., review on key concepts, test format, slides of a sample midterm)
- Lab 4 is up — due after final (5/12).
- Check your grades in iLearn

Electrical
Computer Science
Engineering

120A

つづく