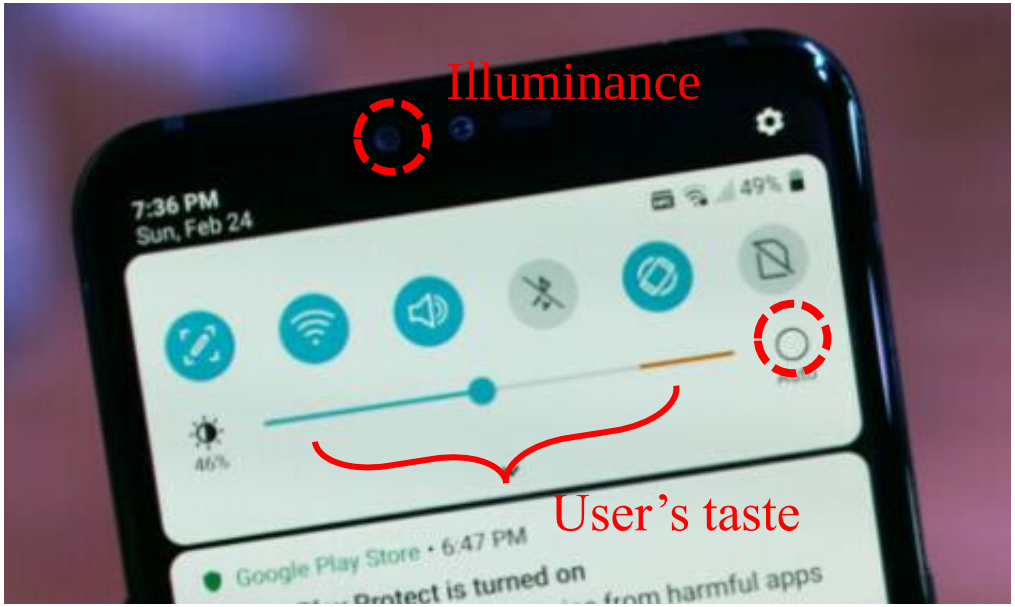
Design
Smartphone Auto Brightness Control!

- Which information?
- Which operation?

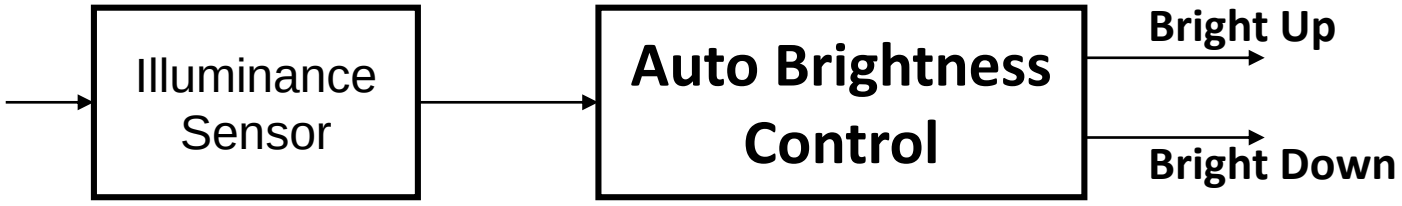


Now You Know Combinational Logic!

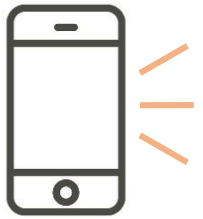
- Generally, combinational logic “compute” or “process”
- Given operation, you can implement it thru hardware

➔It is called **circuit design**

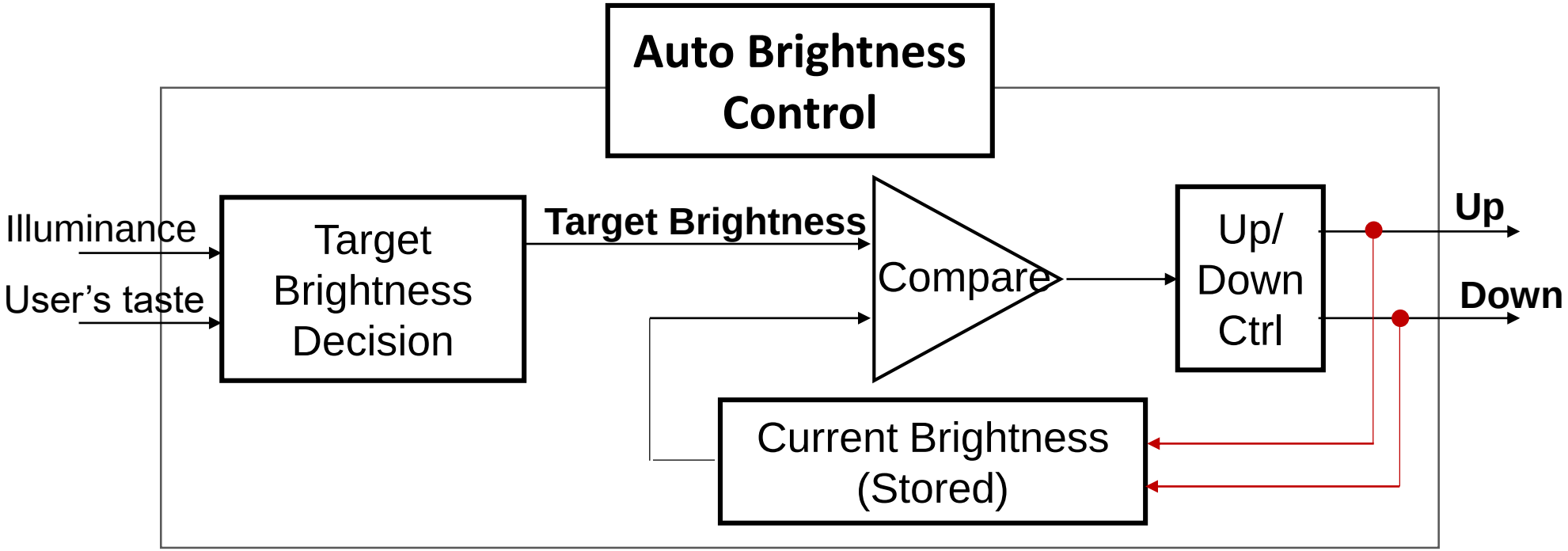
- Of course, illuminance is given with binary codes.
- Then, you should decide whether brightness is **up** or **down**.



Specifying Required Operation



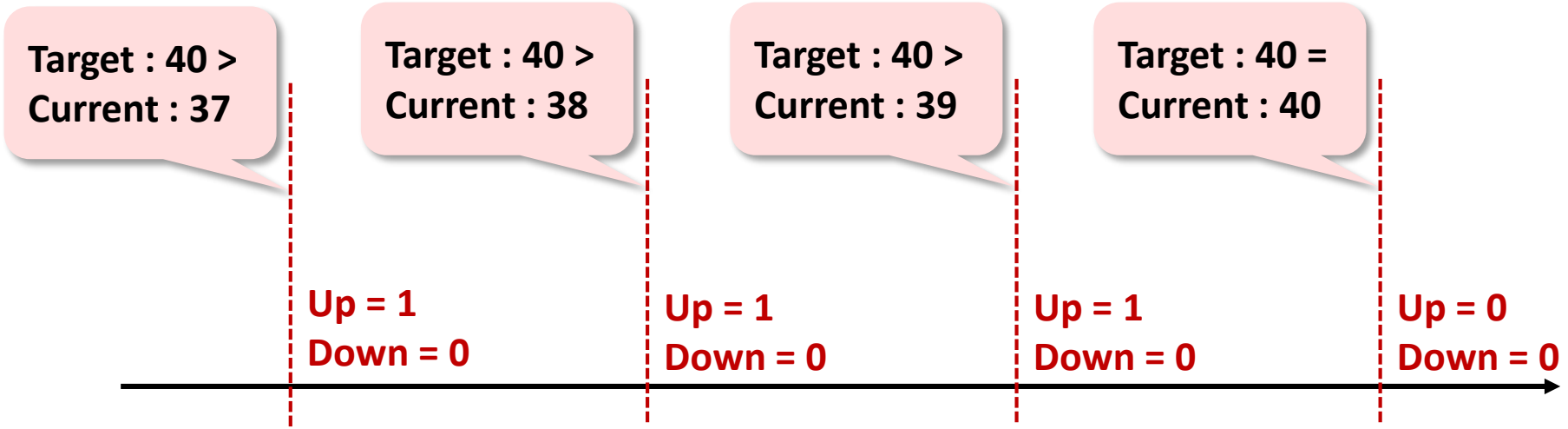
1) Given illuminance, the target brightness of smartphone monitor is determined based on **the user's taste**
2) Target brightness should be compared with the current brightness
3) Whether Up or Down should be determined
4) **Current brightness should be updated**



What Do We Need to Further Study?

① How we **store** & update information

② Operating based on **discrete times**





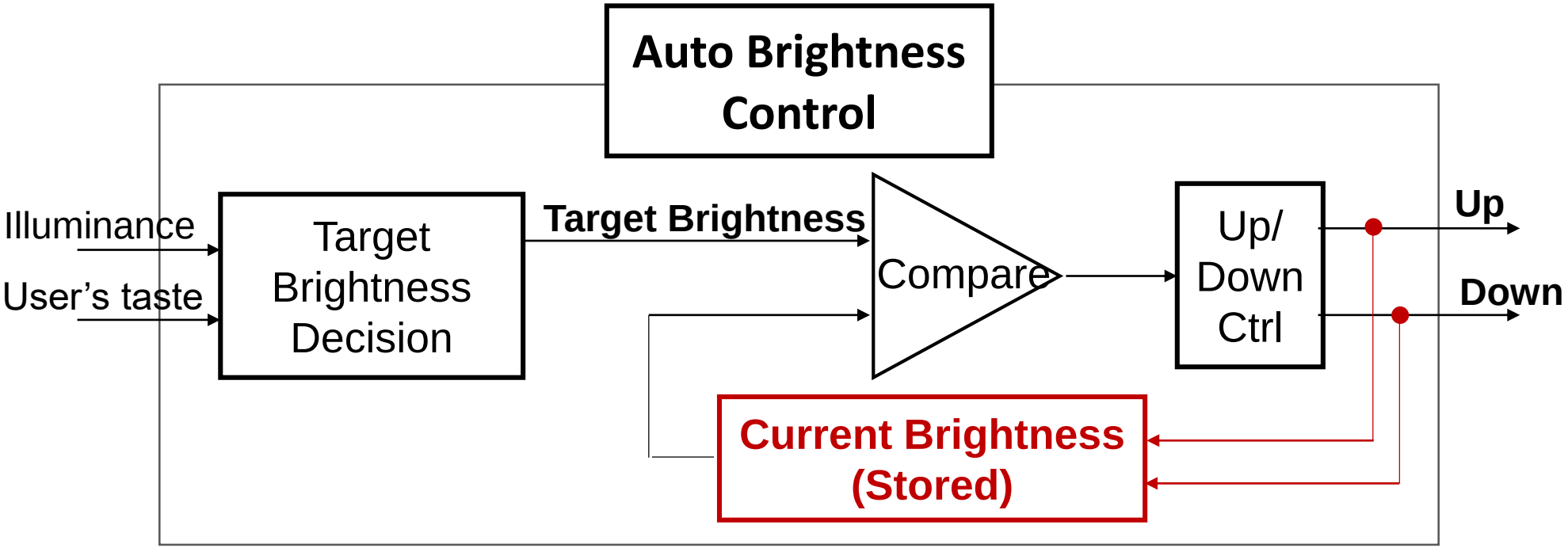
Chap 5
Synchronous Sequential Logic (1)

Hanwool Jeong

hwjeong@kw.ac.kr

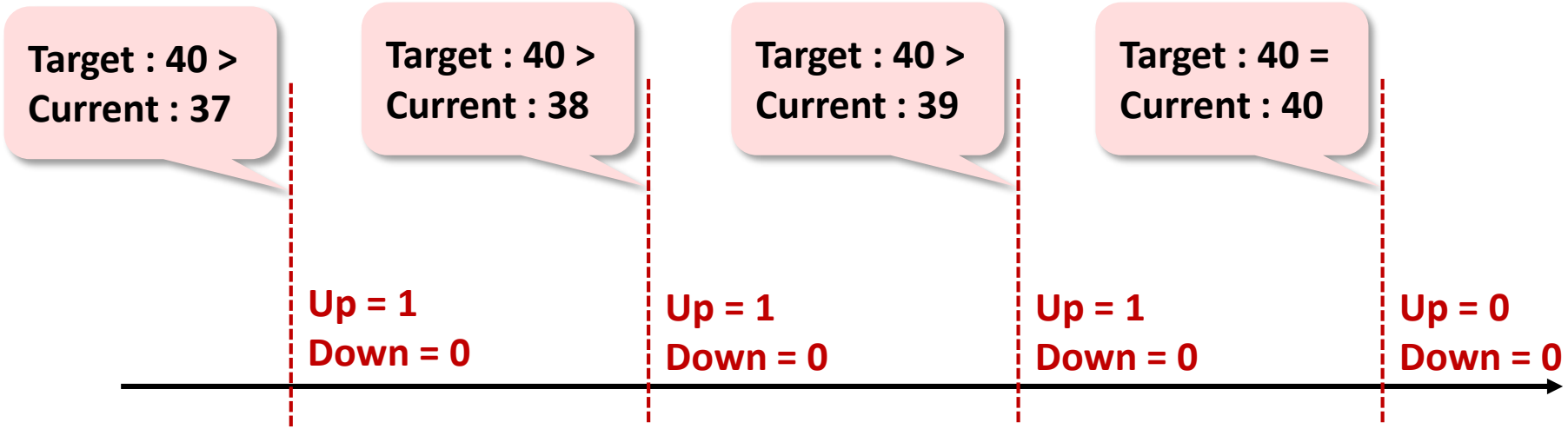
Concept of State

- Combinational logic : Input determines output
- But in brightness control, not only input but also “current brightness” also determines output
- This is the **state** of a circuit! It is stored and may be updated every time unit.



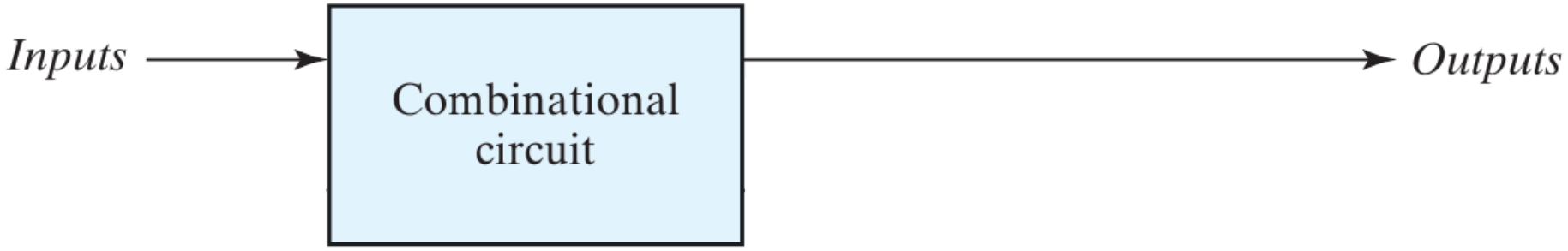
How is the State Determined?

- It is determined by sequence of inputs!



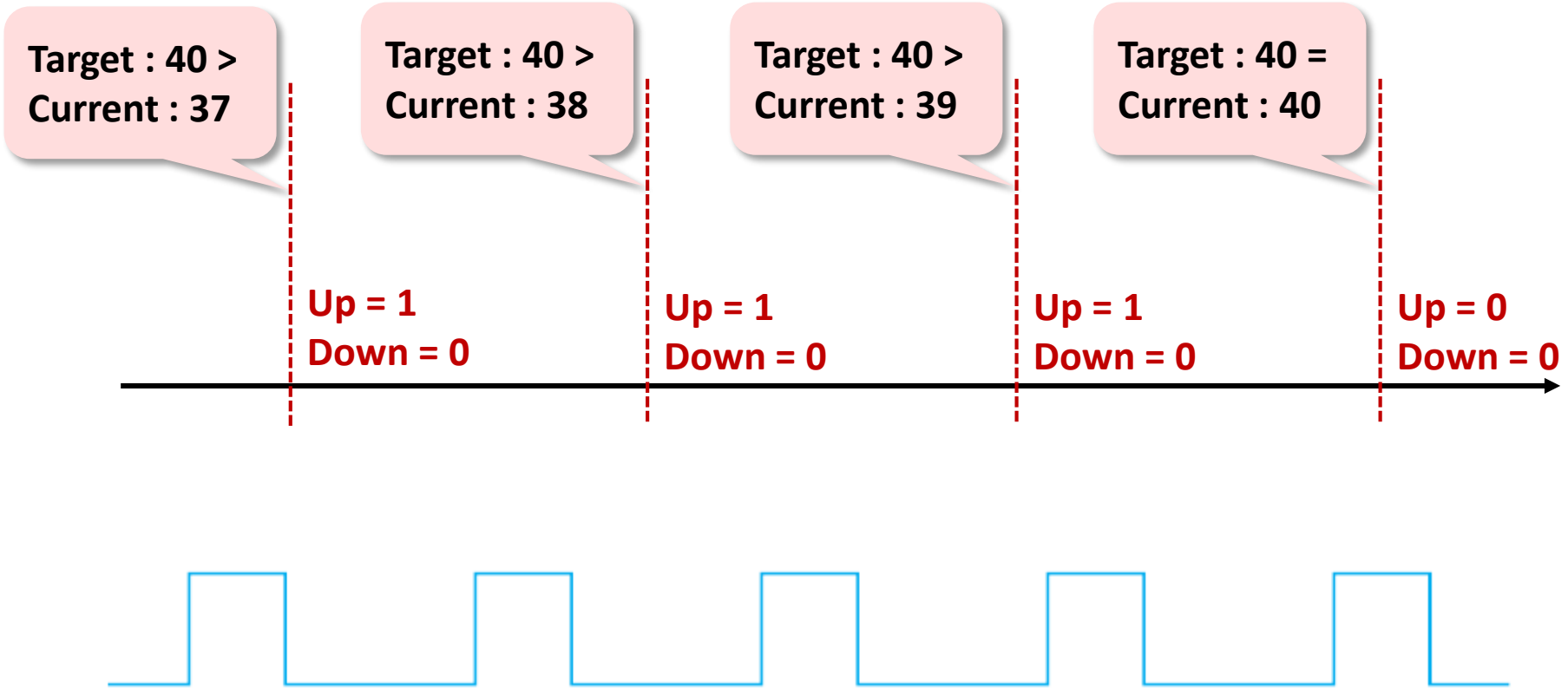
Sequential Circuit

- State is stored in memory, which is to be used at later time.
- Inputs/States/Outputs are all formed in binary.



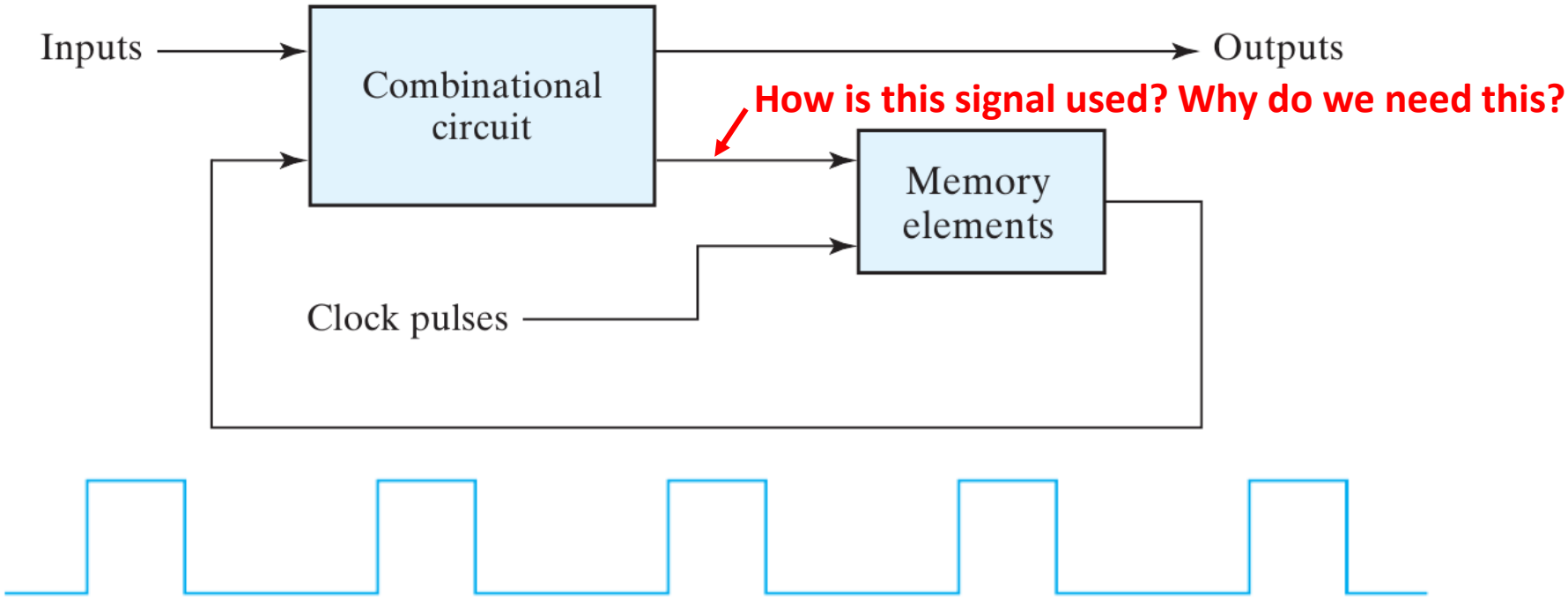
When is the State Updated?

- Not only state is updated, but all things starts something based on the discrete timing unit.



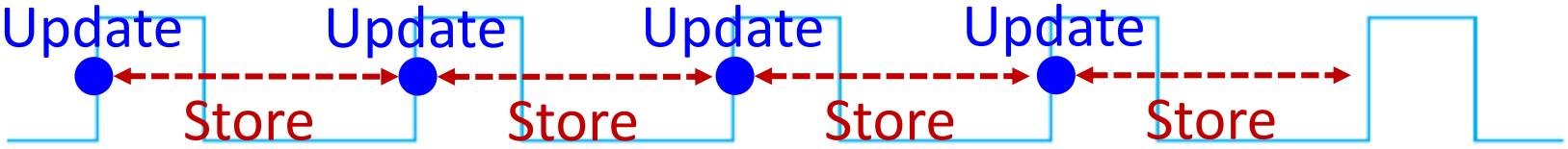
Synchronous Sequential Circuit

- Storage elements in clocked sequential circuits are called **flip-flops**, which is a binary storage device storing one bit.
- Flip-flop updates the state every rising edge of clock. Then, the state is feedback to the combinational logic.



How Can We Store States?

- States should be used mapped into binary numbers.
- If we have 50 possible states, then we should allocate 6 bits to represent a state. That is, we need six flip flops.
- Then, how does flip-flop or memory elements (Flip-flop) for storing state look like?
- Flip- flop should be able to perform three main operations
  - The state should be able to be changed to what is intended at the rising edge of clock
  - The state should be able to be stored
  - The state should be detected



Let's Divide Our Goals for Designing a Flip-Flop

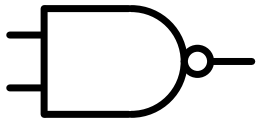
① Write/hold/read operation

② Updatable at clock edge

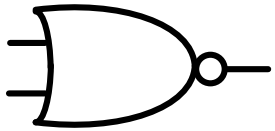
- Let's focus on ① first, which are revisited as follows
  - The state should be able to be changed to what is intended ~~at the rising edge of clock~~
  - The state should be able to be stored
  - The state should be detected

Revisit NAND2 or NOR2

- NAND2 and NOR2 become inverter!
- NAND or NOR yield fixed output independently to other inputs when



A	B	Y
0	0	
0	1	
1	0	
1	1	

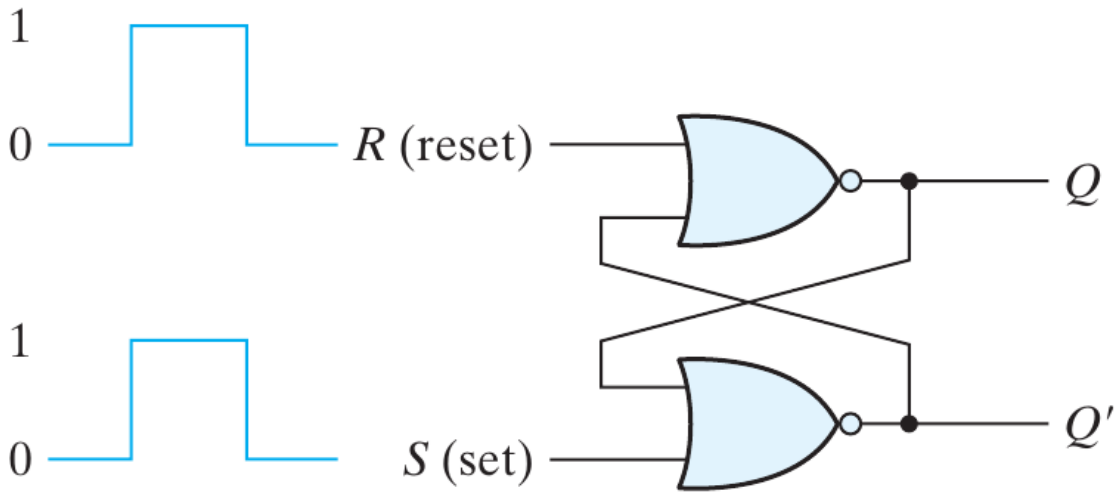


A	B	Y
0	0	
0	1	
1	0	
1	1	

Can You Make A Storage Element?

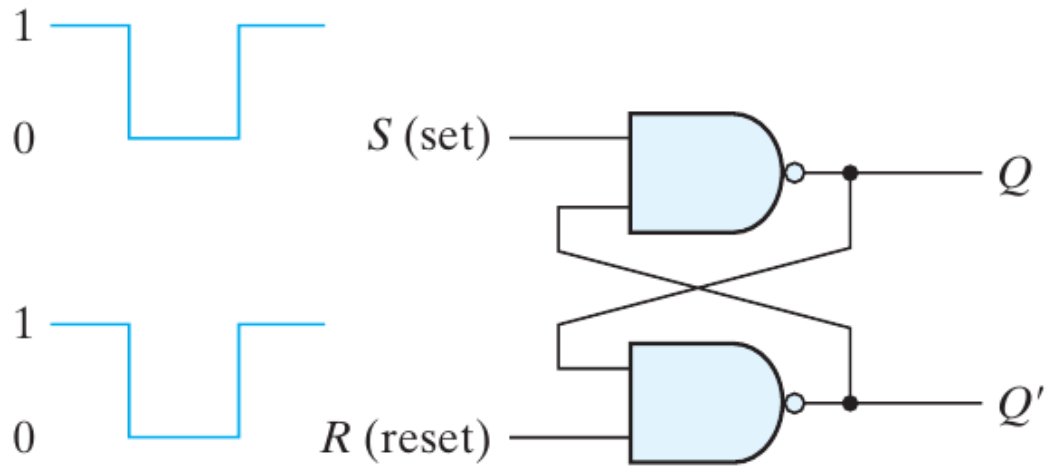
- Please bear in mind that write/hold/read operation should be able to be performed.
- We called such storage element as **latch**

SR Latch with NOR



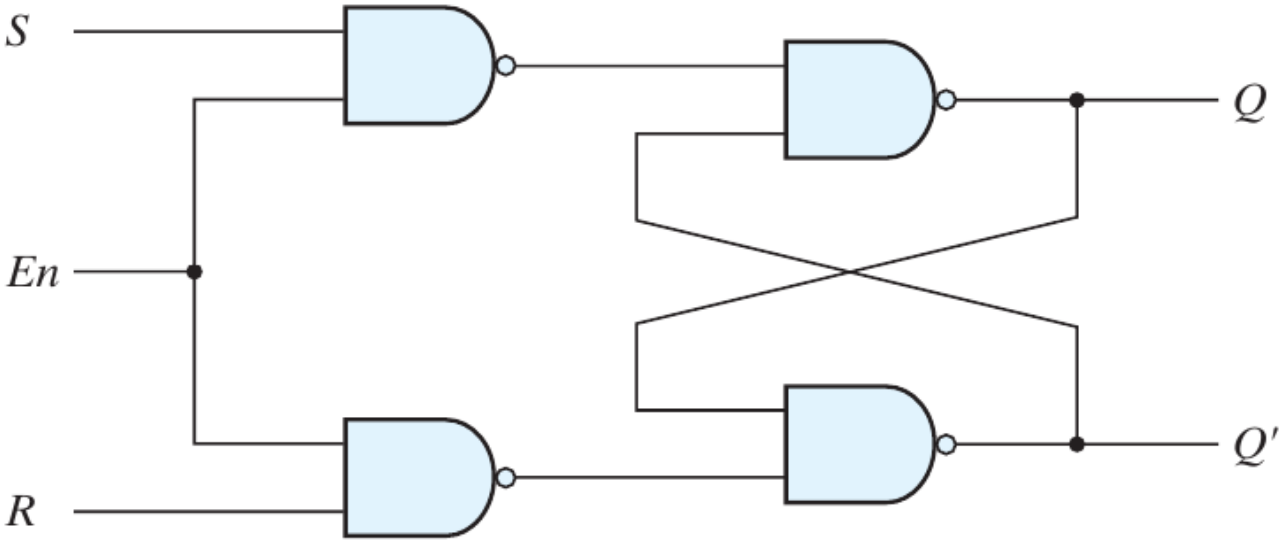
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

SR Latch with NAND



S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

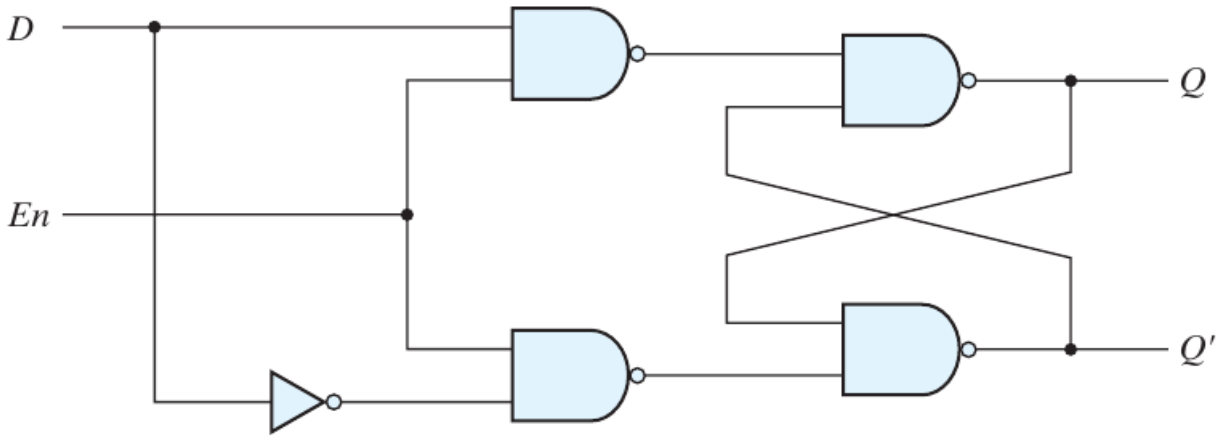
SR Latch with Control Input



En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

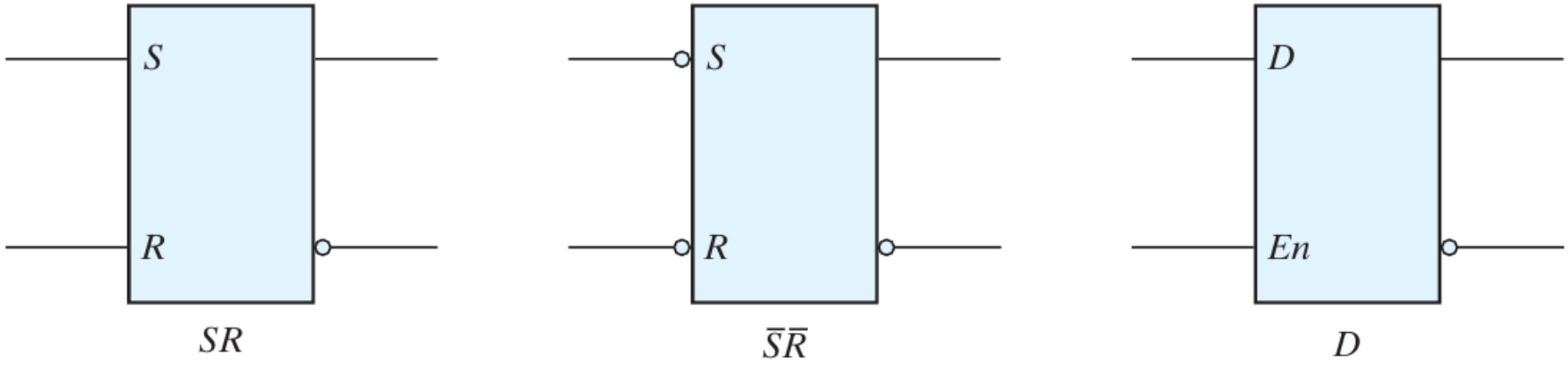
D Latch;
Eliminating Undesirable Condition

- Two inputs: D and En



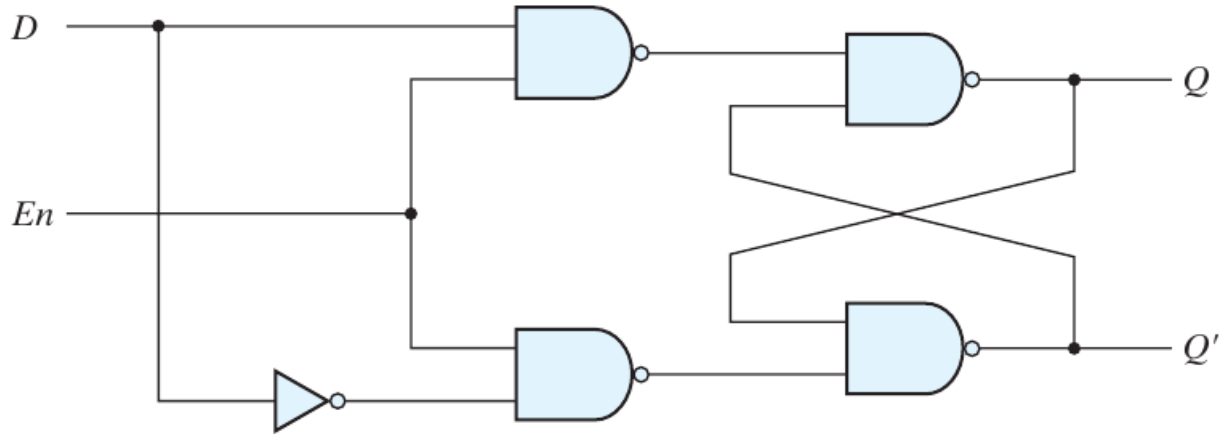
En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

Graphical Symbols for Latches



Additional Constraint of Flip Flop Compared to Latch

- Revisiting a D-latch, it changes the state whenever D changes, once En is enabled.



Revisit Our Goals for Designing a Flip Flop

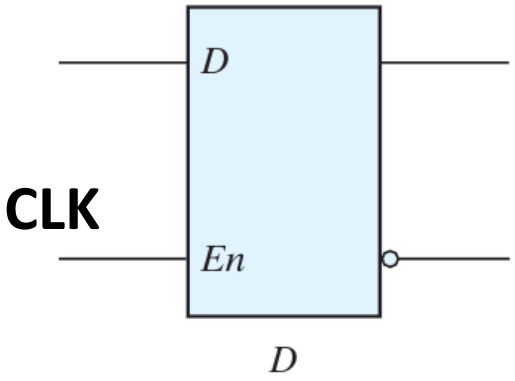
① Write/hold/read operation (We already did this!)

- The state should be able to be changed to what is intended ~~at the rising edge of clock~~
- The state should be able to be stored
- The state should be detected

② Updatable **at the clock edge**

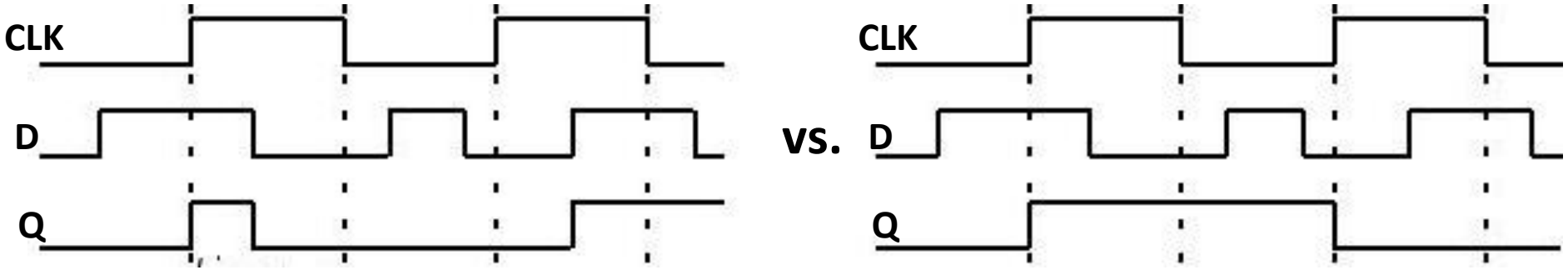
- Now, let's realize ②

Can't we use this?



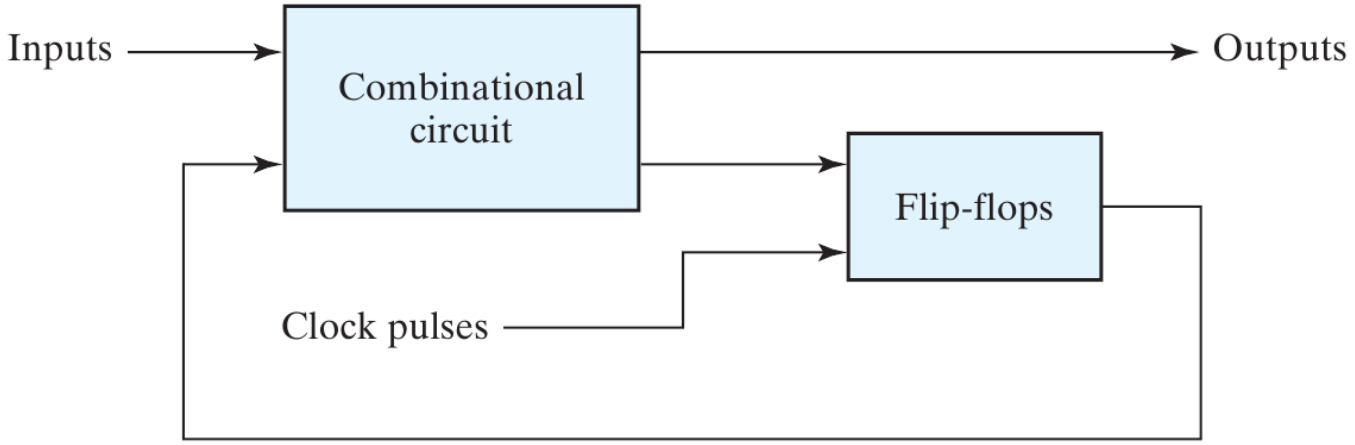
No! What We Want is

- Only at the rising edge, not during rising phase



Revisit Synchronous Sequential Circuit; We Cannot Use Latch For State Storage

- The latch cannot be used for state storage below. Why?
  - Suppose that latch input changes while the clock pulse is still at 1.
  - The latches will respond to new values and update a new state.
  - The result is an unpredictable situation, since the state of the latches may keep changing for as long as the clock is high
- Thus, the output of a latch cannot be applied directly or thru combinational logic to the input of the same or another latch when all the latches are triggered by a common clock source.



Requirement of Flip-Flop; Edge-Triggered

- Q is updated only at the edge. By how?



(a) Response to positive level



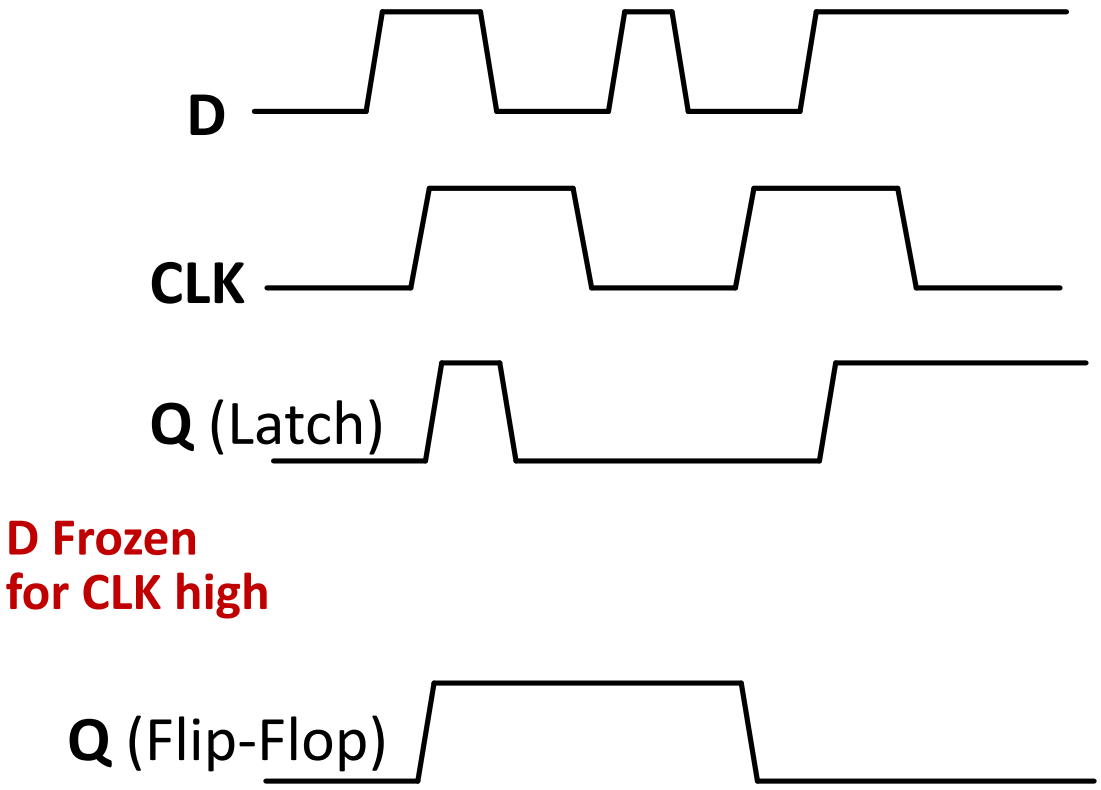
(b) Positive-edge response

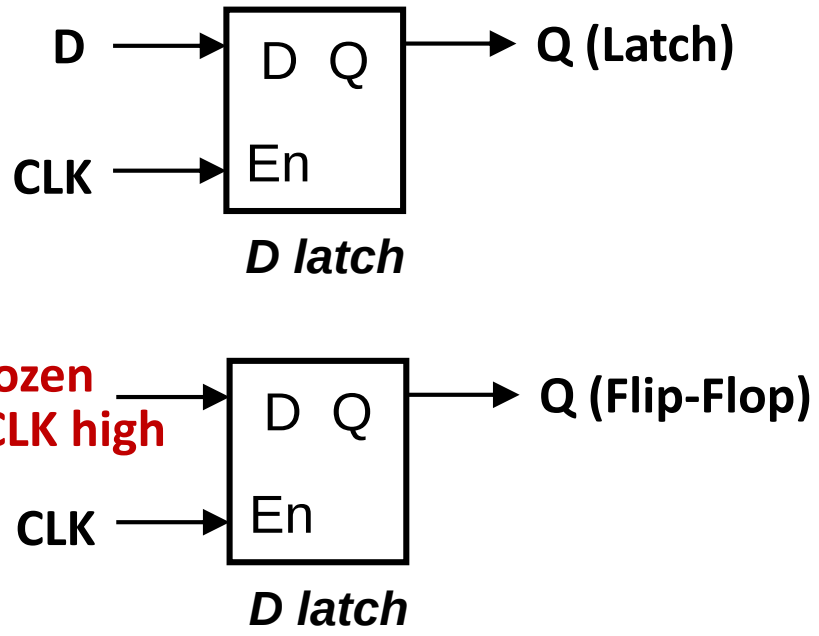


(c) Negative-edge response

Can You Design Flip Flop Using D-Latch?

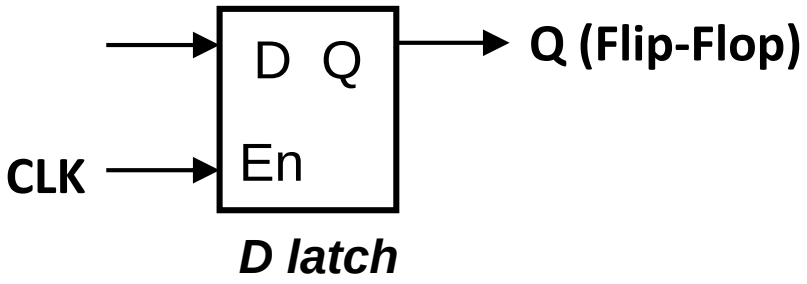
- Positive edge triggered flip-flop should “capture” D at the rising edge of clock.
- **Can you make this flip flop utilizing/modifying D-latches?**



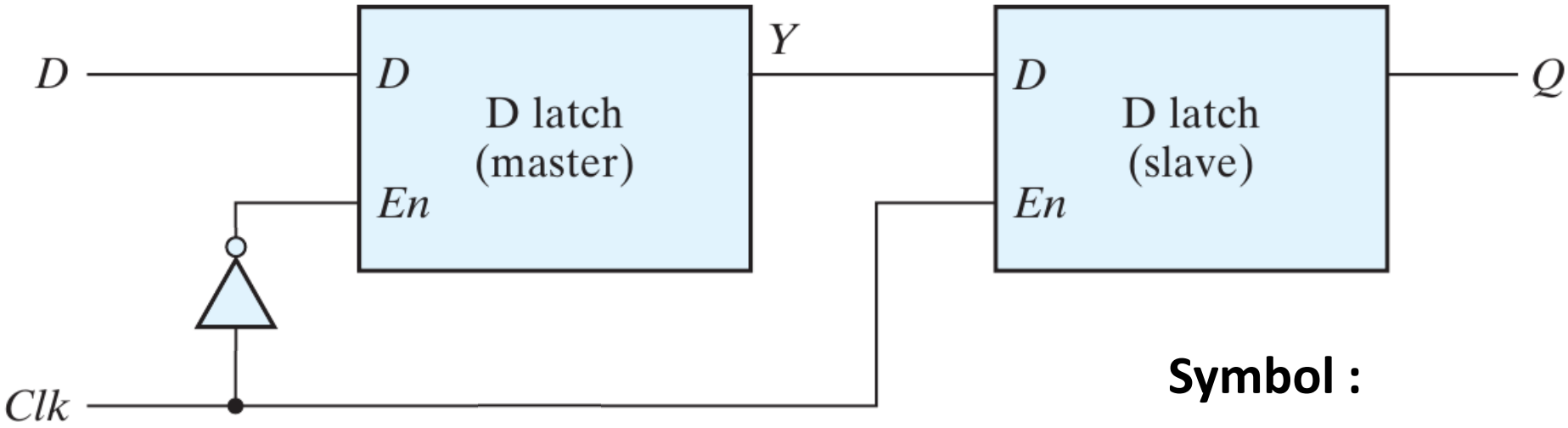


How Can We Freeze D for High CLK?

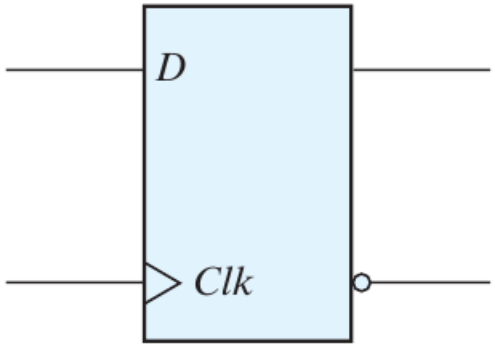
- You remember D latch can hold data. When?
- That is EN = 0!



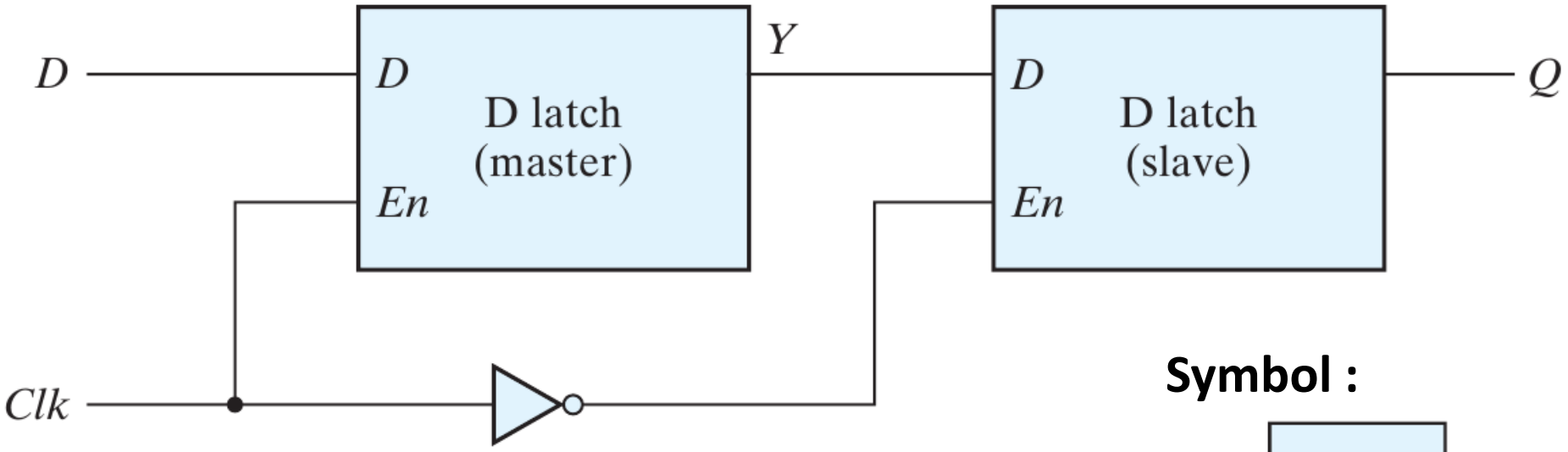
Positive Clock Triggered Master-Slave D Flip Flop



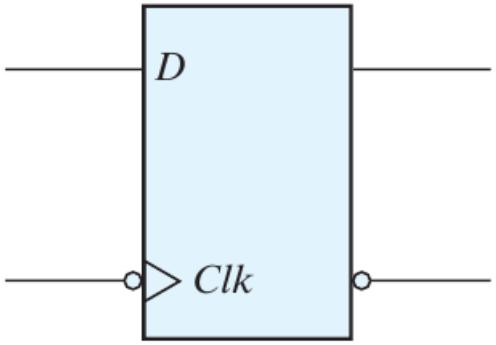
Symbol :



Negative Clock Triggered Master-Slave D Flip Flop

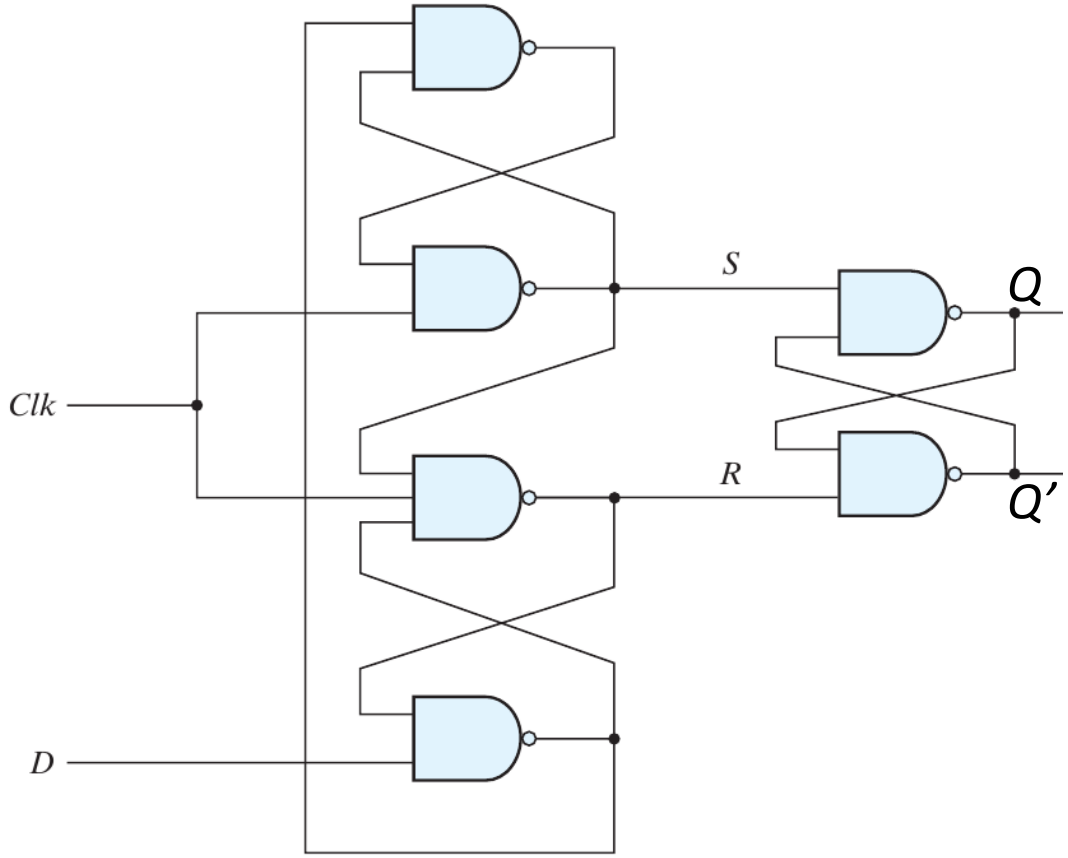


Symbol :



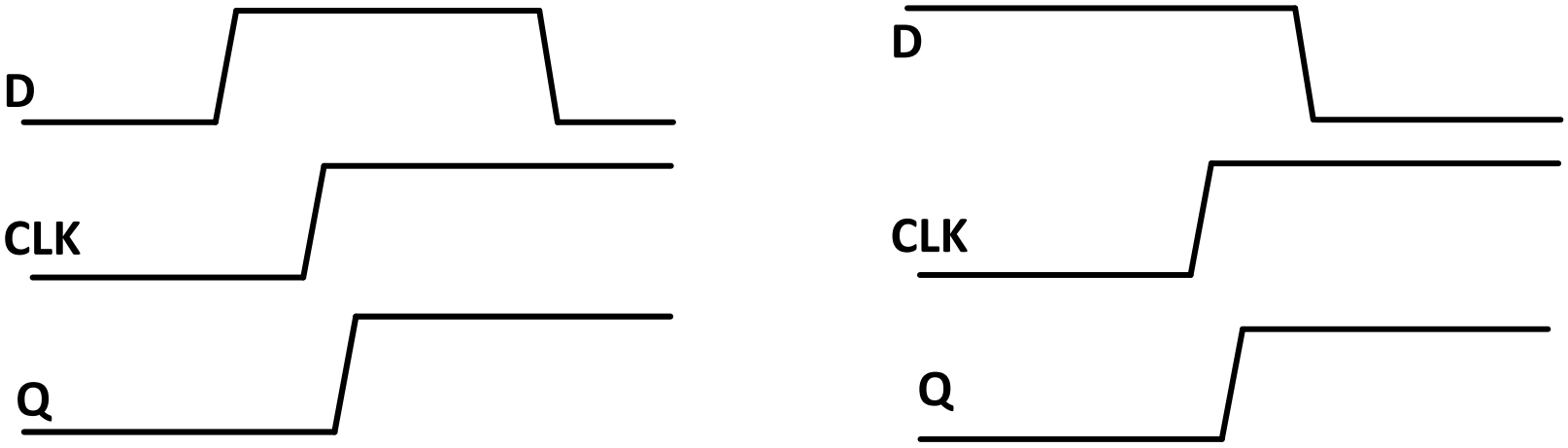
D Flip Flop Using Three SR Latches

- The S and R are high when Clk = 0 ➔ Output Hold
- If D = 0 when Clk=1, R becomes 0, making Q = 0.
- Even D changes for Clk = 1, R remains at 0. Thus, the flip-flop is locked out and is unresponsive to further changes in the input.
- When Clk returns to 0, R goes to 1 ➔ Output hold.
- Any change in D while Clk = 1 does not affect the output.



Setup Time & Hold Time in Flip Flop

- Ideally, FF should capture at the rising edge although D changes close to clock edge.
- Setup time is the minimum time constraint that the data input should be stable (kept unchanged) **before** the clock edge for correct operation.
- Hold time is the minimum time constraint that the data input should be stable (kept unchanged) **after** the clock edge for correct operation.



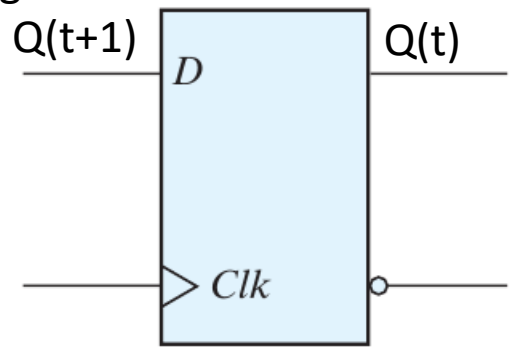
Other Flip Flops; JK Flip Flop

- There are four operations to be performed with a flip-flop: Hold the current Q, Set it to 1, reset it to 0, or complement Q.
- Can you design JK FF with D-FF?

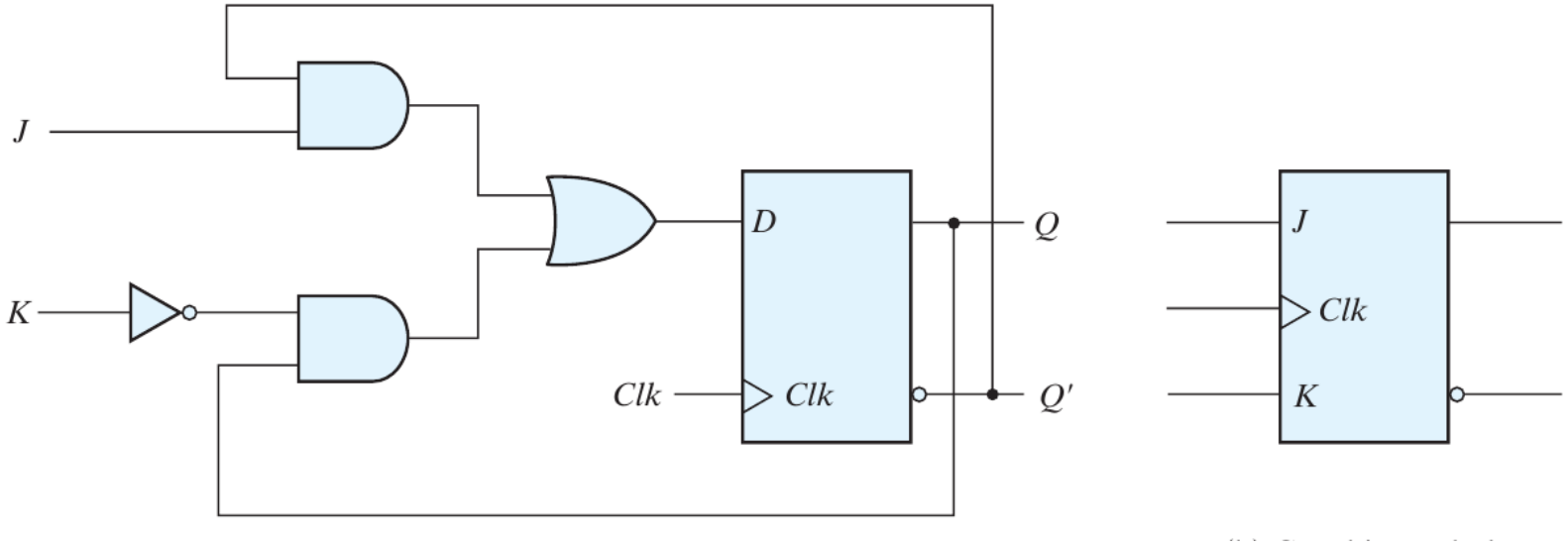
Table 5.1
Flip-Flop Characteristic Tables

***JK* Flip-Flop**			
J	***K***	***Q*(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

It is nothing but
Next state, Q(t+1)



Structure & Symbol of JK FF



Other Flip Flops; T Flip Flop

- Two operations are needed : Toggle or not
- T(Toggle) flip flop

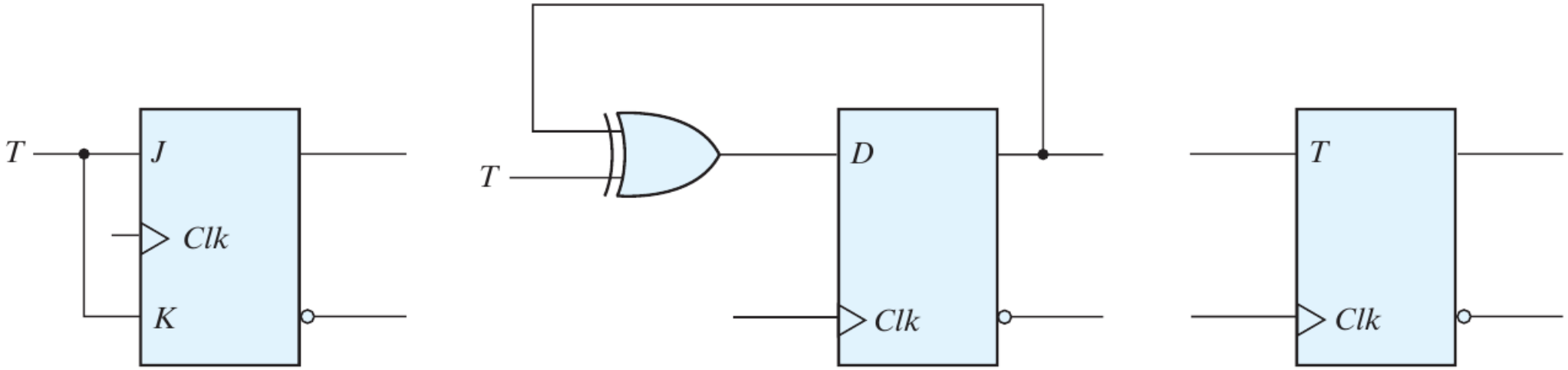
***D* Flip-Flop**

D	***Q*(*t* + 1)**	
0	0	Reset
1	1	Set

***T* Flip-Flop**

T	***Q*(*t* + 1)**	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Structure & Symbol of T-FF



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

Characteristic Equations

- D Flip-flop

$$Q(t+1) = D$$

- JK Flip-flop

$$Q(t+1) = JQ' + K'Q$$

- T flip Flop

$$Q(t+1) = T \oplus Q = TQ' + T'Q$$

What We Learned

- Why do we need and what is synchronous sequential logic
- Components composing of synchronous sequential logic
- Designing of a latch
- Designing of D Flip flop
- Setup time & hold time in FF
- JK FF and T FF