

# **CSC 211: DIGITAL ELECTRONICS II**

- **Sequential Logic Circuits-Flip flops**

**Dr. Ronoh Saweh**

**Department of Computer Science**

**KIBABII UNIVERSITY**

# Introduction

- The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are combinatorial in nature, i.e., the output(s) depends only on the present values of the inputs and not on their past values.
- In sequential circuits, the “state” of the circuit is crucial in determining the output values.
- For a given input combination, a sequential circuit may produce different output values, depending on its previous state.
- In other words, a sequential circuit has a memory (of its past state) whereas a combinatorial circuit has no memory.
- Sequential circuits (together with combinatorial circuits) make it possible to build several useful applications, such as counters, registers, arithmetic/logic unit (ALU), all the way to microprocessors.

# NAND latch (RS latch)

- \* A, B: inputs,  $X_1$ ,  $X_2$ : outputs

- \* Consider  $A = 1$ ,  $B = 0$ .

$B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = A X_2 = 1 \cdot 1 = 0$ .

Overall, we have  $X_1 = 0$ ,  $X_2 = 1$ .

- \* Consider  $A = 0$ ,  $B = 1$ .

$\rightarrow X_1 = 1$ ,  $X_2 = 0$ .

- \* Consider  $A = B = 1$ .

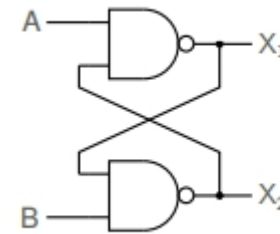
$X_1 = A X_2 = X_2$ ,  $X_2 = B X_1 = X_1 \Rightarrow X_1 = X_2$

If  $X_1 = 1$ ,  $X_2 = 0$  previously, the circuit continues to “hold” that state.

Similarly, if  $X_1 = 0$ ,  $X_2 = 1$  previously, the circuit continues to “hold” that state.

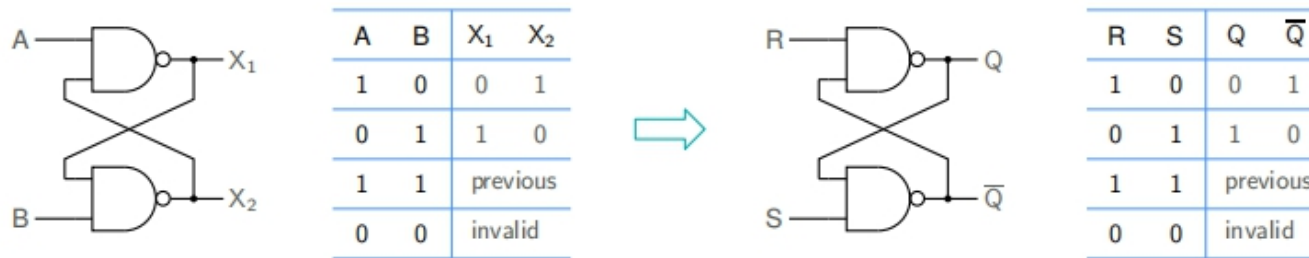
The circuit has “latched in” the previous state.

- \* For  $A = B = 0$ ,  $X_1$  and  $X_2$  are both 1. This combination of A and B is not allowed for reasons that will become clear later.



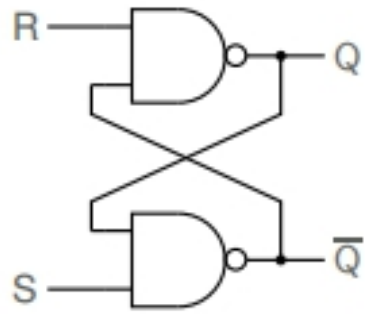
A	B	$X_1$	$X_2$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	1	1

# NAND latch (RS latch)

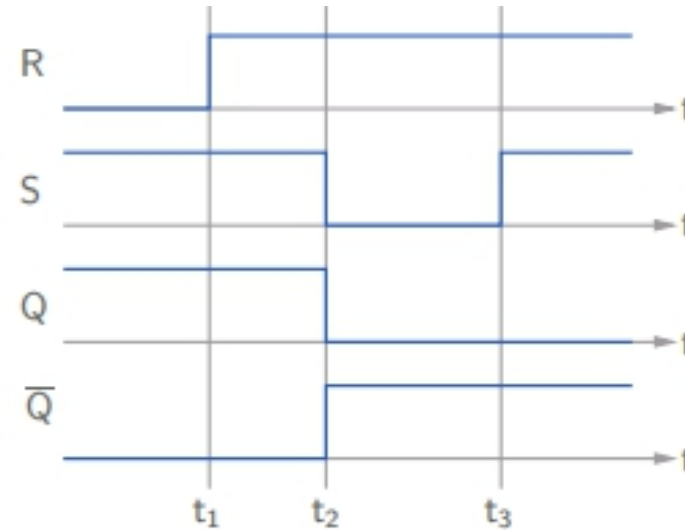


- \* The combination  $A = 1, B = 0$  serves to reset  $X_1$  to 0 (irrespective of the previous state of the latch).
- \* The combination  $A = 0, B = 1$  serves to set  $X_1$  to 1 (irrespective of the previous state of the latch).
- \* In other words,  
 $A = 1, B = 0 \rightarrow$  latch gets reset to 0.  
 $A = 0, B = 1 \rightarrow$  latch gets set to 1.
- \* The A input is therefore called the RESET (R) input, and B is called the SET (S) input of the latch.
- \*  $X_1$  is denoted by Q, and  $X_2$  (which is  $X_1$  in all cases except for  $A = B = 0$ ) is denoted by  $\overline{Q}$ .

# NAND latch (RS latch)

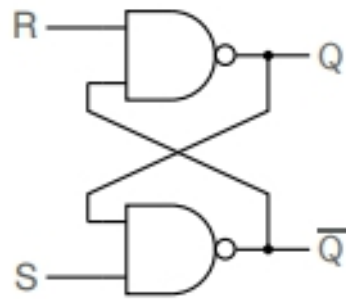


R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

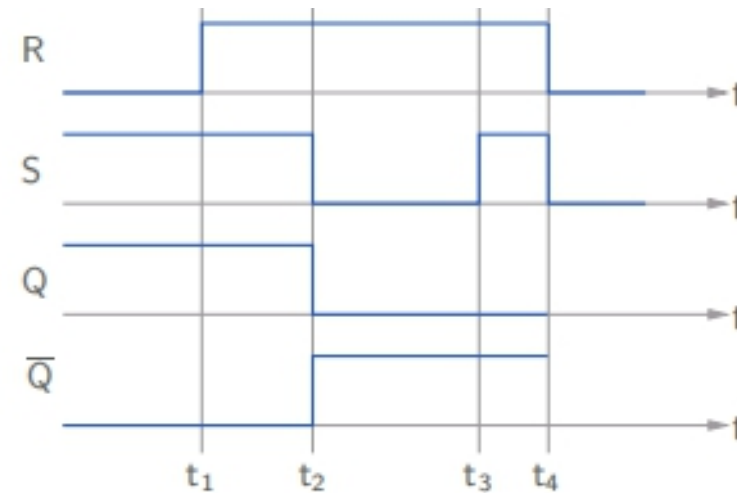


- \* Up to  $t = t_1$ ,  $R = 0$ ,  $S = 1 \rightarrow Q = 1$ .
- \* At  $t = t_1$ ,  $R$  goes high  $\rightarrow R = S = 1$ , and the latch holds its previous state  $\rightarrow$  no change at the output.
- \* At  $t = t_2$ ,  $S$  goes low  $\rightarrow R = 1$ ,  $S = 0 \rightarrow Q = 0$ .
- \* At  $t = t_3$ ,  $S$  goes high  $\rightarrow R = S = 1$ , and the latch holds its previous state  $\rightarrow$  no change at the output.

# NAND latch (RS latch)



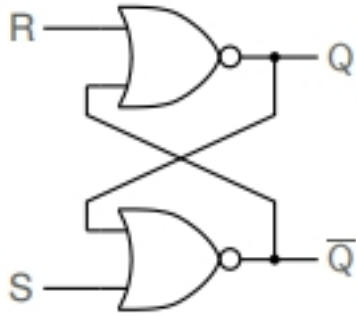
R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	



Why not allow  $R = S = 0$ ?

- It makes  $Q = \overline{Q} = 1$ , i.e.,  $Q$  and  $\overline{Q}$  are not inverse of each other any more.
- More importantly, when  $R$  and  $S$  both become 1 simultaneously (starting from  $R = S = 0$ ), the final outputs  $Q$  and  $\overline{Q}$  cannot be uniquely determined. We could have  $Q = 0$ ,  $\overline{Q} = 1$  or  $Q = 1$ ,  $\overline{Q} = 0$ , depending on the delays associated with the two NAND gates.

# NOR latch (RS latch)



R	S	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
0	0	previous	
1	1	invalid	

\* The NOR latch is similar to the NAND latch:

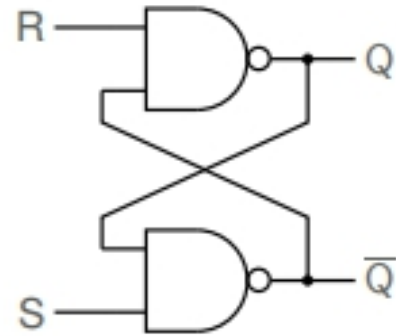
When  $R = 1$ ,  $S = 0$ , the latch gets reset to  $Q = 0$ .

When  $R = 0$ ,  $S = 1$ , the latch gets set to  $Q = 1$ .

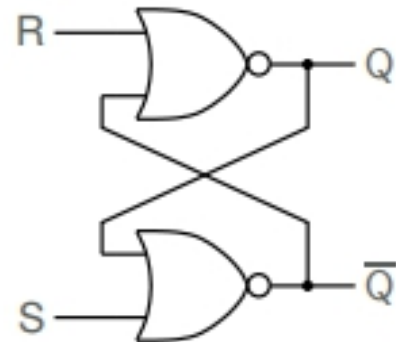
\* For  $R = S = 0$ , the latch retains its previous state (i.e., the previous values of  $Q$  and  $\bar{Q}$ ).

\*  $R = S = 1$  is not allowed for reasons similar to those discussed in the context of the NAND latch.

# Comparison of NAND and NOR latches



R	S	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	



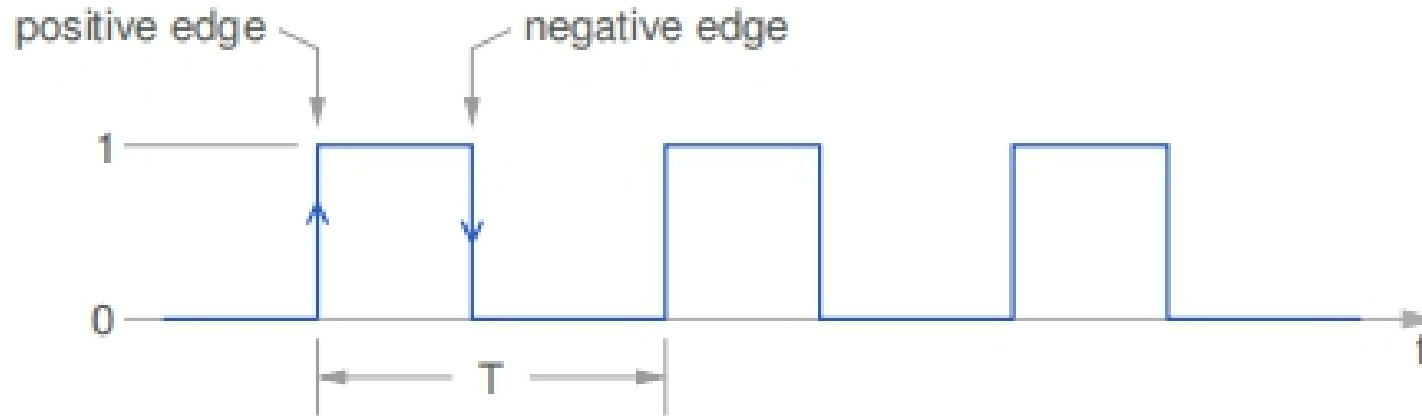
R	S	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
0	0	previous	
1	1	invalid	



# The “clock”

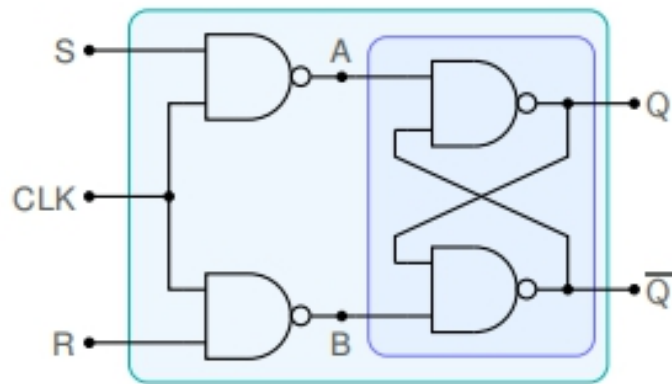
- \* Complex digital circuits are generally designed for synchronous operation, i.e., transitions in the various signals are synchronised with the clock.
- \* Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.
- \* A clock is a periodic signal, with a positive-going transition and a negative-going transition.

# The “clock”



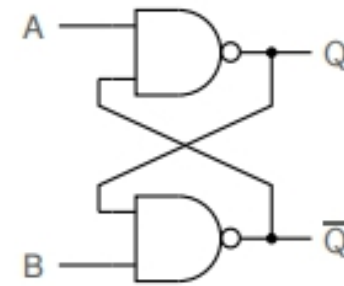
- The clock frequency determines the overall speed of the circuit. For example, a processor that operates with a 1 GHz clock is 10 times faster than one that operates with a 100 MHz clock.
- Intel 80286 (IBM PC-AT): 6 MHz
- Modern CPU chips: 2 to 3 GHz.

# The “clock”



Clocked RS latch

CLK	R	S	Q	$\bar{Q}$
0	X	X	previous	
1	1	0	0	1
1	0	1	1	0
1	0	0	previous	
1	1	1	invalid	

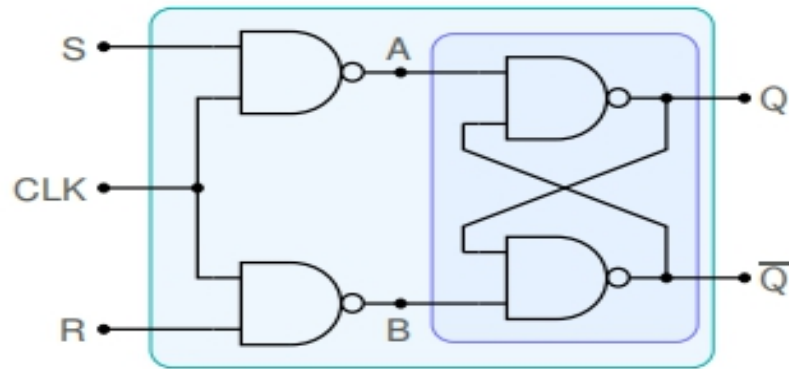


NAND RS latch

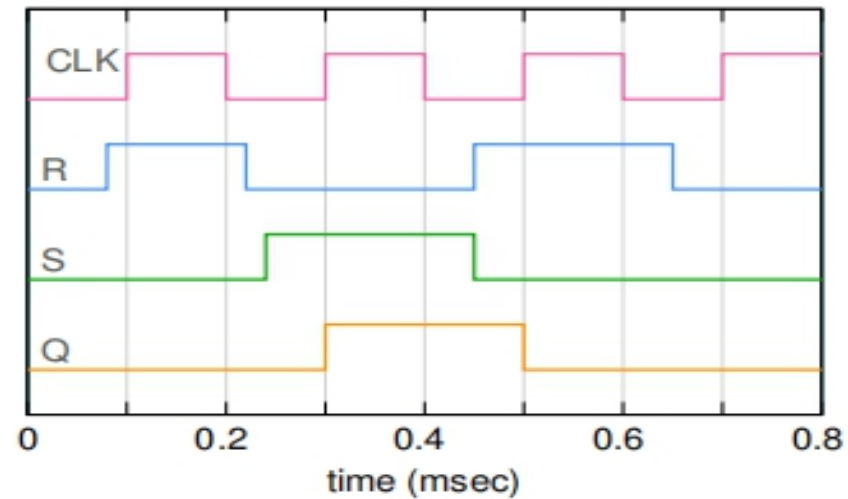
A	B	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

- \* When clock is inactive (0),  $A = B = 1$ , and the latch holds the previous state.
- \* When clock is active (1),  $A = S$ ,  $B = R$ . Using the truth table for the NAND RS latch (right), we can construct the truth table for the clocked RS latch.
- \* Note that the above table is sensitive to the level of the clock (i.e., whether CLK is 0 or 1).

# Clocked RS latch

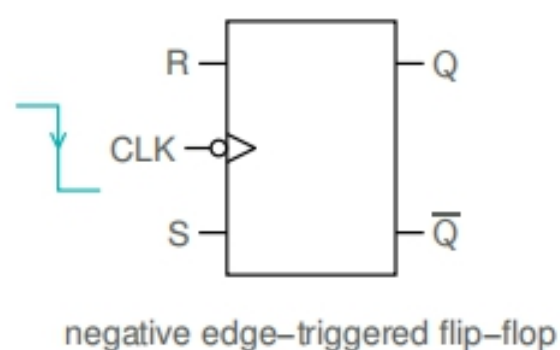
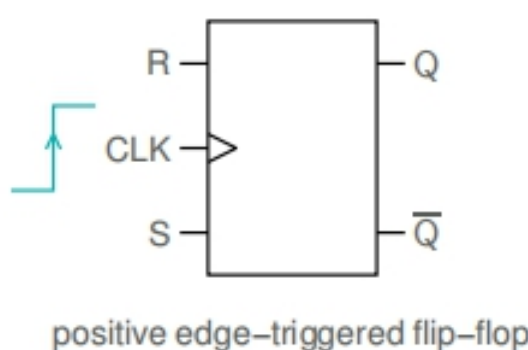


CLK	R	S	Q	$\bar{Q}$
0	X	X	previous	
1	1	0	0	1
1	0	1	1	0
1	0	0	previous	
1	1	1	invalid	



# Edge-triggered flip-flops

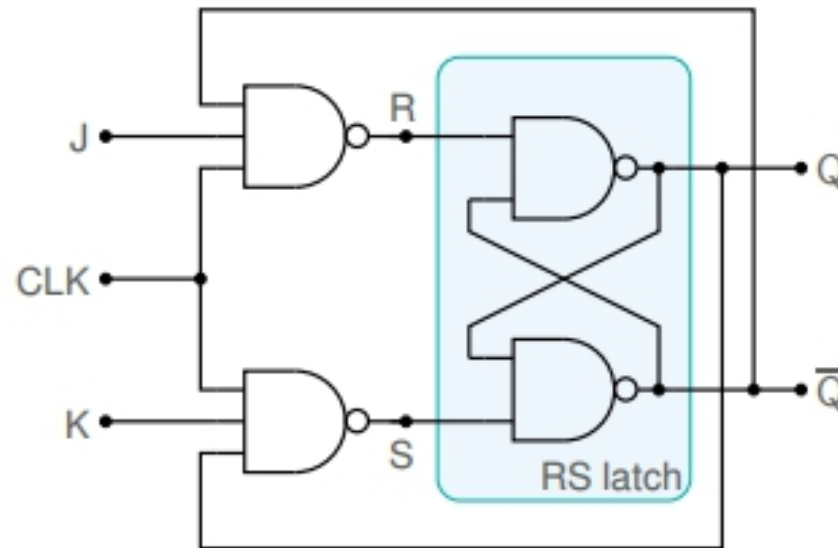
- \* The clocked RS latch seen previously is level-sensitive, i.e., if the clock is active (CLK = 1), the flip-flop output is allowed to change, depending on the R and S inputs.
- \* In an edge-sensitive flip-flop, the output can change only at the active clock edge (i.e., CLK transition from 0 to 1 or from 1 to 0).
- \* Edge-sensitive flip-flops are denoted by the following symbols:



# JK flip-flop: introduction

R	S	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

Truth table for RS latch



CLK	J	K	Q ( $Q_{n+1}$ )
0	X	X	previous ( $Q_n$ )
1	0	0	previous ( $Q_n$ )

Truth table for JK flip-flop

# JK flip-flop: introduction

\* When  $\text{CLK} = 0$ , we have  $R = S = 1$ , and the RS latch holds the previous  $Q$ . In other words, nothing happens as long as  $\text{CLK} = 0$ .

\* When  $\text{CLK} = 1$ :

-  $J = K = 0 \rightarrow R = S = 1$ , RS latch holds previous  $Q$ , i.e.,  $Q_{n+1} = Q_n$ , where  $n$  denotes the  $n^{\text{th}}$  clock pulse (This notation will become clear shortly).

-  $J = 0, K = 1 \rightarrow R = 1, S = Q_n$ .

Case (i):  $Q_n = 0 \rightarrow S = 1$  (i.e.,  $R = S = 1$ )  $\rightarrow Q_{n+1} = Q_n = 0$ .

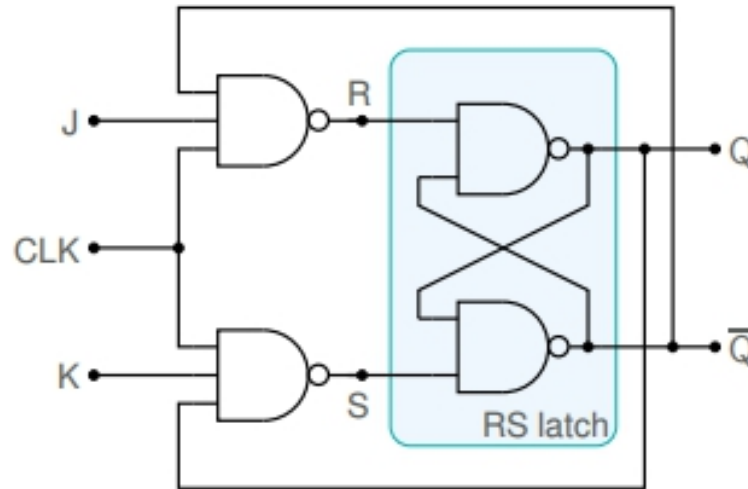
Case (ii):  $Q_n = 1 \rightarrow S = 0$  (i.e.,  $R = 1, S = 0$ )  $\rightarrow Q_{n+1} = 0$ .

In either case,  $Q_{n+1} = 0 \rightarrow$  For  $J = 0, K = 1, Q_{n+1} = 0$ .

# JK flip-flop: introduction

R	S	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
1	1	previous	
0	0	invalid	

Truth table for RS latch



CLK	J	K	Q ( $Q_{n+1}$ )
0	X	X	previous ( $Q_n$ )
1	0	0	previous ( $Q_n$ )
1	0	1	0
1	1	0	1
1	1	1	toggles ( $\bar{Q}_n$ )

Truth table for JK flip-flop

\* When CLK = 1:

- Consider  $J = 1, K = 0 \rightarrow S = 1, R = Q_n = Q_n$ .

Case (i):  $Q_n = 0 \rightarrow R = 0$  (i.e.,  $R = 0, S = 1$ )  $\rightarrow Q_{n+1} = 1$ .

Case (ii):  $Q_n = 1 \rightarrow R = 1$  (i.e.,  $R = 1, S = 1$ )  $\rightarrow Q_{n+1} = Q_n = 1$ .

$\rightarrow$  For  $J = 1, K = 0, Q_{n+1} = 1$ .

- Consider  $J = 1, K = 1 \rightarrow R = Q_n, S = Q_n$ .

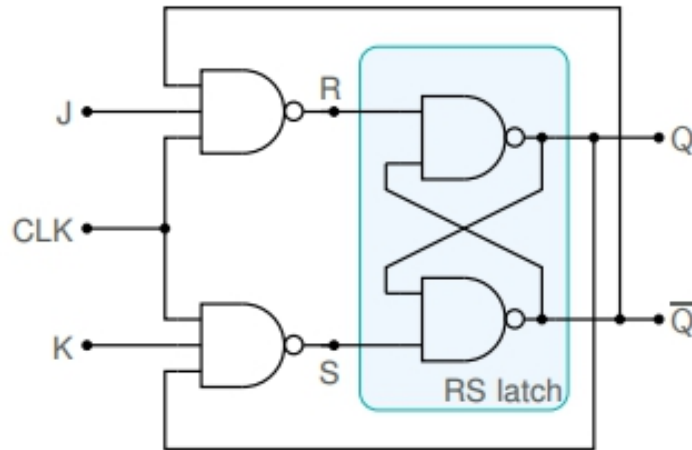
Case (i):  $Q_n = 0 \rightarrow R = 0, S = 1 \rightarrow Q_{n+1} = 1$ .

Case (ii):  $Q_n = 1 \rightarrow R = 1, S = 0 \rightarrow Q_{n+1} = 0$ .

$\rightarrow$  For  $J = 1, K = 1, Q_{n+1} = \bar{Q}_n$ .



# JK flip-flop: introduction



CLK	J	K	Q ( $Q_{n+1}$ )
0	X	X	previous ( $Q_n$ )
1	0	0	previous ( $Q_n$ )
1	0	1	0
1	1	0	1
1	1	1	toggles ( $\overline{Q_n}$ )

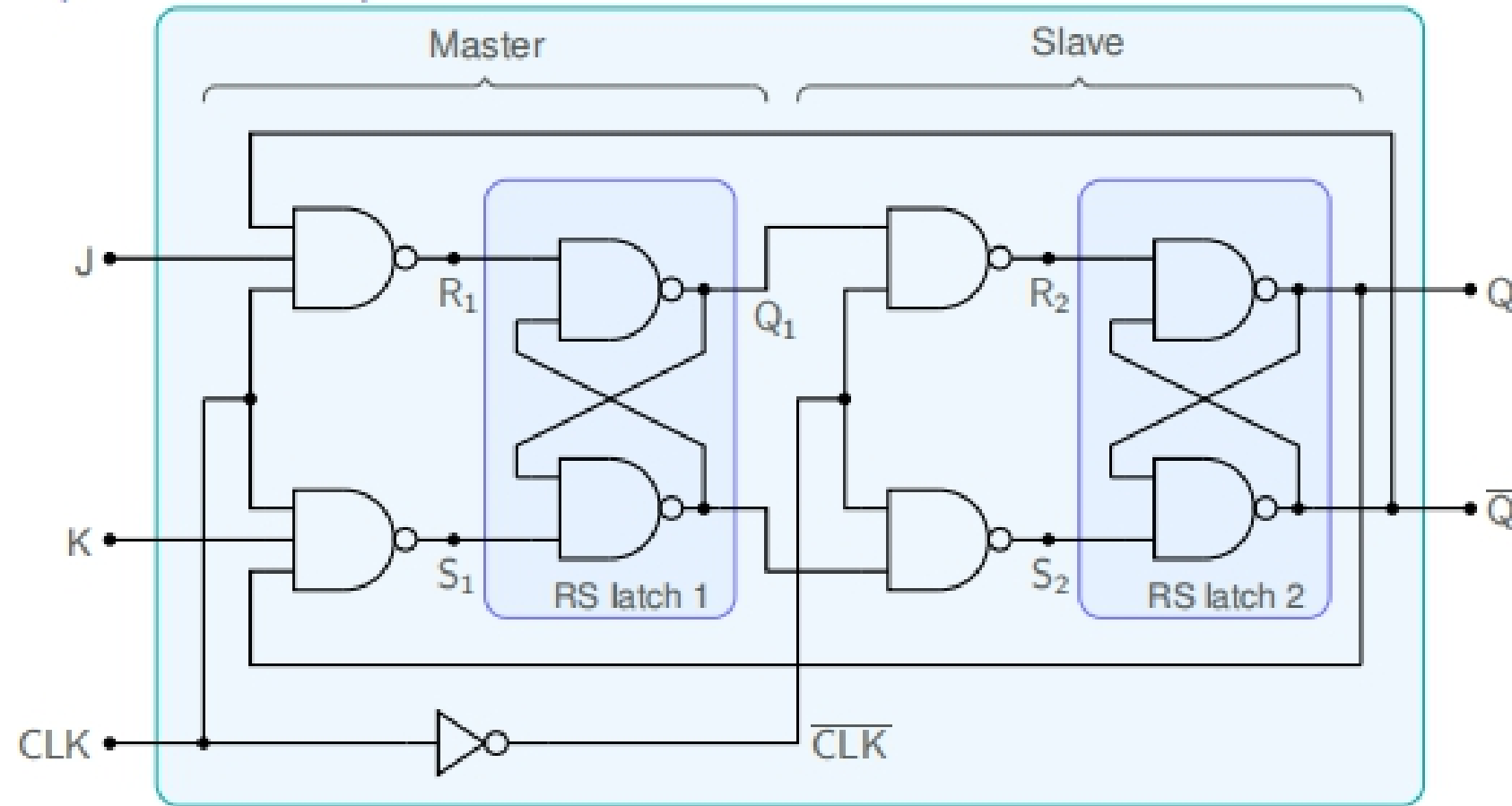
Truth table for JK flip-flop

Consider  $J = K = 1$  and  $CLK = 1$ .

As long as  $CLK = 1$ , Q will keep toggling! (The frequency will depend on the delay values of the various gates).

→ Use the “Master-slave” configuration.

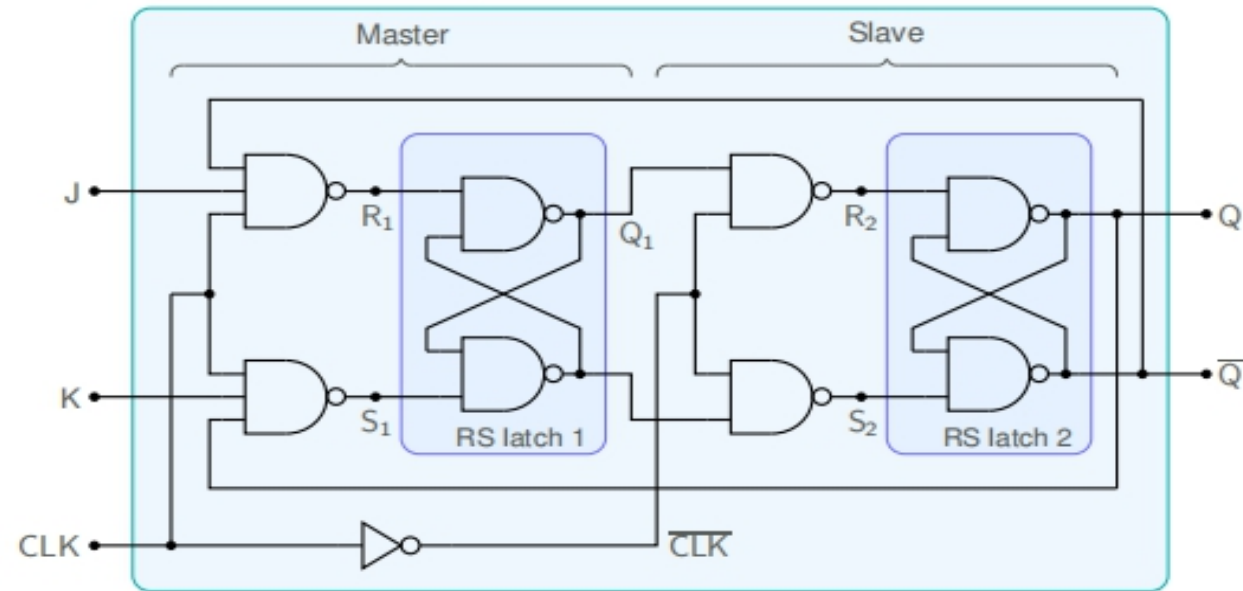
# JK flip-flop (Master-Slave)



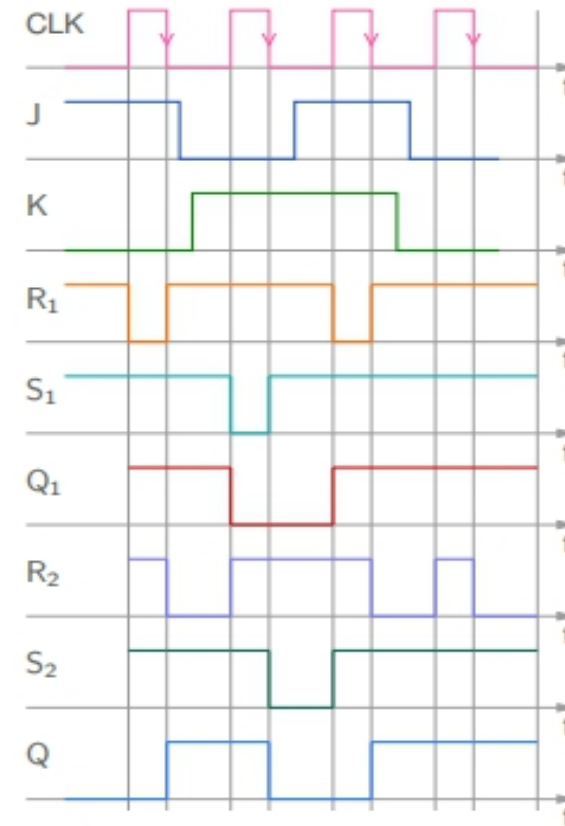
# JK flip-flop (Master-Slave)

- \* When CLK goes high, only the first latch is affected; the second latch retains its previous value (because  $\text{CLK} = 0 \rightarrow R_2 = S_2 = 1$ ).
- \* When CLK goes low, the output of the first latch ( $Q_1$ ) is retained (since  $R_1 = S_1 = 1$ ), and  $Q_1$  can now affect  $Q$ .
- \* In other words, the effect of any changes in J and K appears at the output Q only when CLK makes a transition from 1 to 0.
- This is therefore a negative edge-triggered flip-flop.
- \* Note that the JK flip-flop allows all four input combinations.

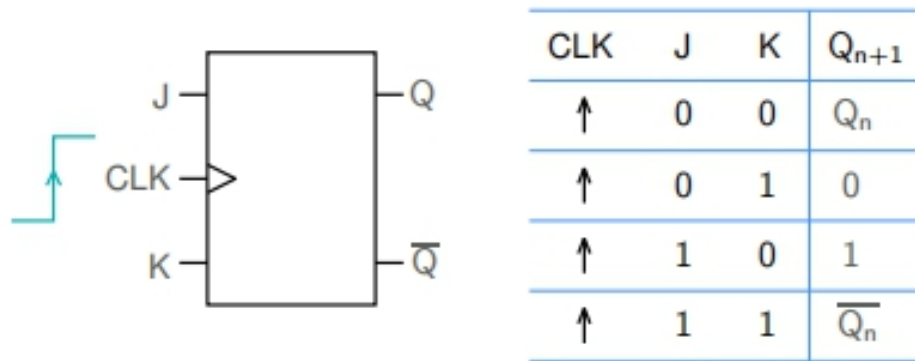
# JK flip-flop (Master-Slave)



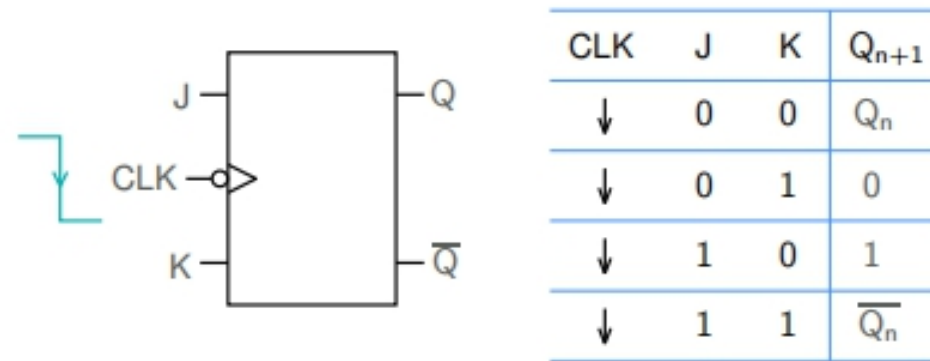
CLK	J	K	$Q_{n+1}$
↓	0	0	$Q_n$
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q_n}$



# JK flip-flop



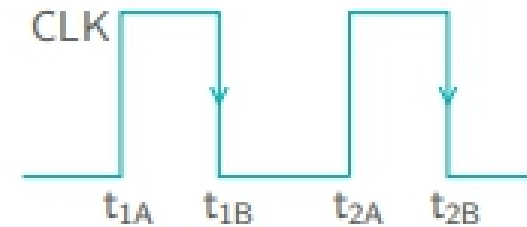
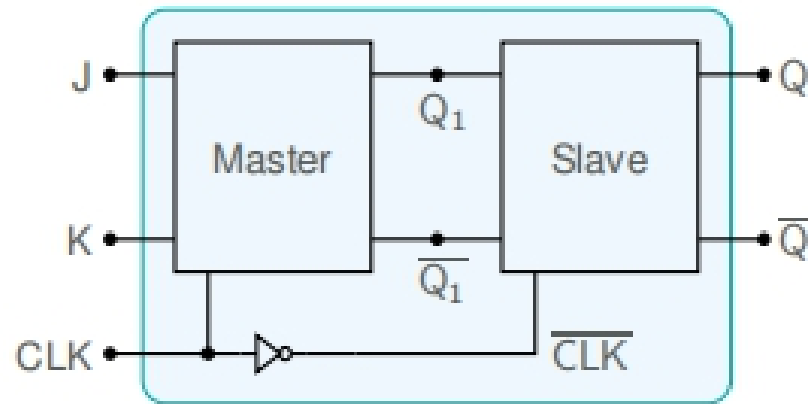
positive edge-triggered JK flip-flop



negative edge-triggered JK flip-flop

\* Both negative (e.g., 74ALS112A, CD54ACT112) and positive (e.g., 74ALS109A, CD4027) edge triggered JK flip-flops are available as ICs.

# JK flip-flop



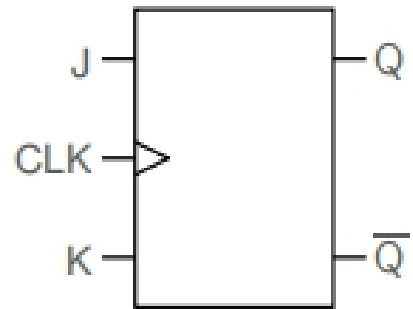
Consider a negative edge-triggered JK flip-flop.

- \* As seen earlier, when CLK is high (i.e.,  $t_{1A} < t < t_{1B}$ , etc.), the input J and K determine the Master latch output Q<sub>1</sub>. During this time, no change is visible at the flip-flop output Q.

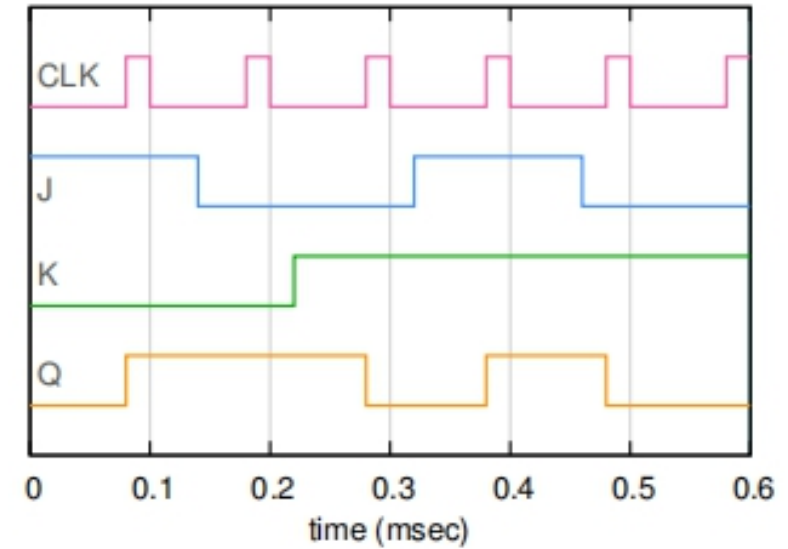
# JK flip-flop

- \* When the clock goes low, the Slave flip-flop becomes active, making it possible for Q to change.
- \* In short, although the flip-flop output Q can only change after the active edge, ( $t_{1B}$  ,  $t_{2B}$  , etc.), the new Q value is determined by J and K values just before the active edge.
  - This is a very important point!

# JK flip-flop

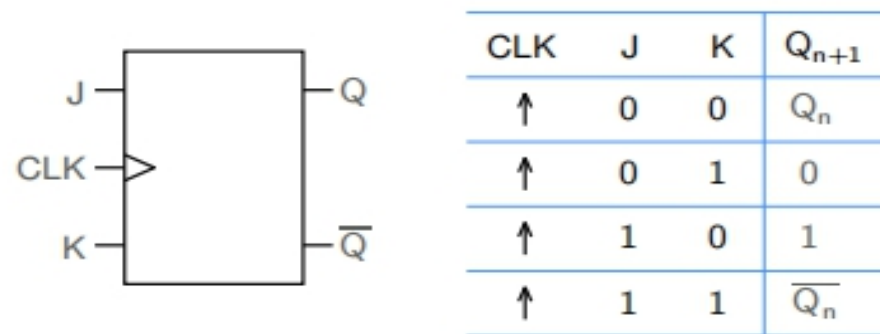


CLK	J	K	$Q_{n+1}$
↑	0	0	$Q_n$
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_n}$

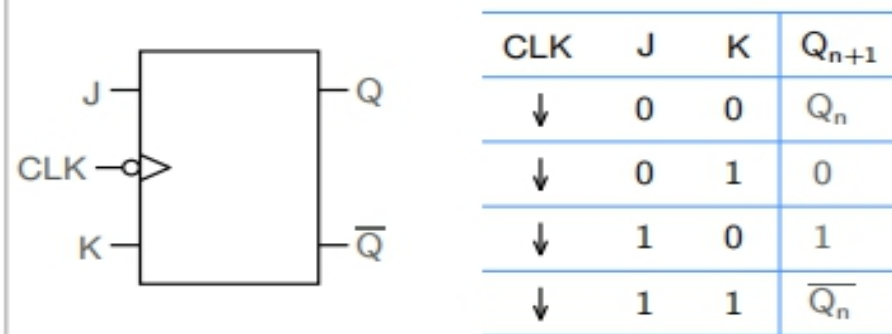
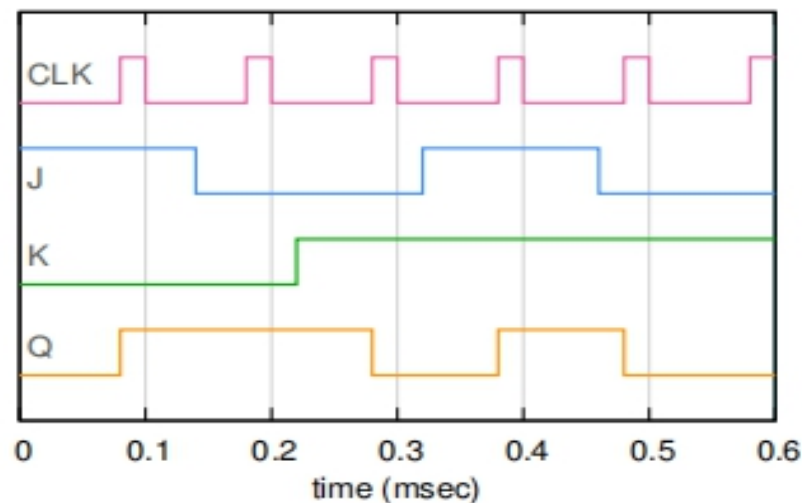




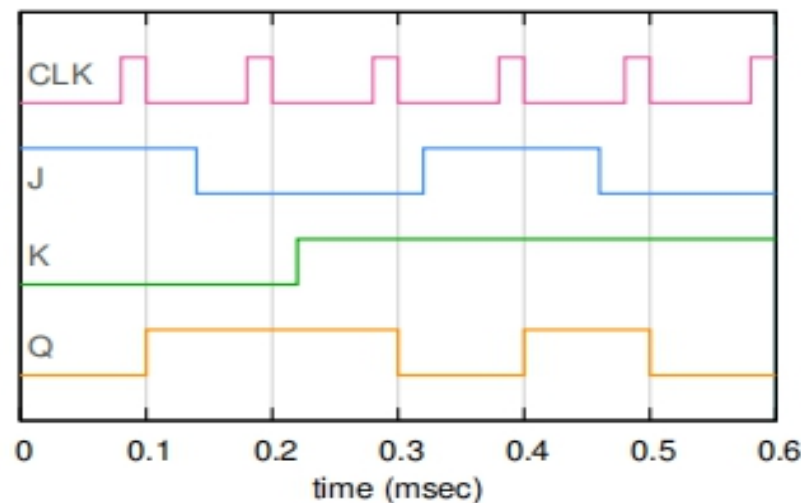
# JK flip-flop



positive edge-triggered JK flip-flop

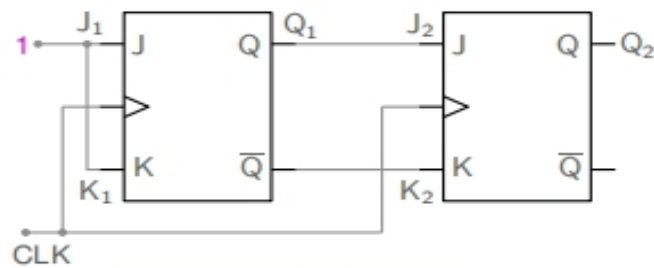


negative edge-triggered JK flip-flop

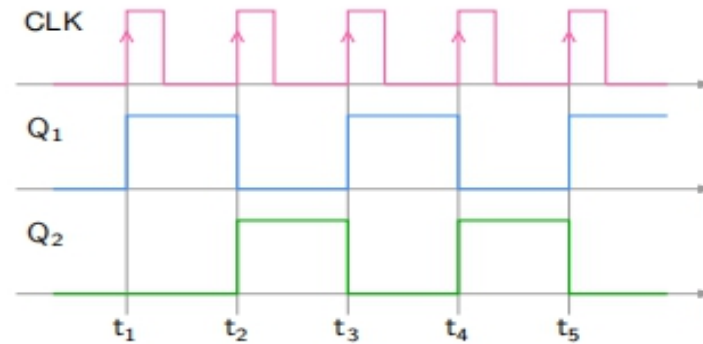


# JK flip-flop

$J_1 = K_1 = 1$ . Assume  $Q_1 = Q_2 = 0$  initially.



CLK	J	K	$Q_{n+1}$
↑	0	0	$Q_n$
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_n}$



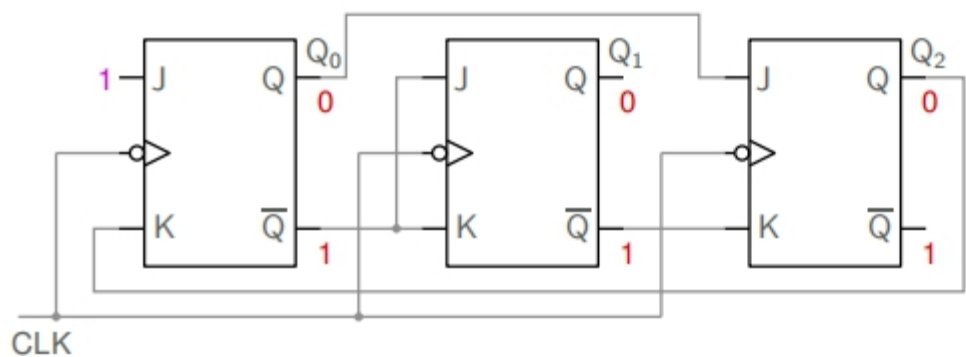
t	$J_2(t = t_k^-)$	$K_2(t = t_k^-)$	$Q_2(t = t_k^+)$
$t_1$	0	1	0
$t_2$	1	0	1
$t_3$	0	1	0
$t_4$	1	0	1
$t_5$	0	1	0

\* Since  $J_1 = K_1 = 1$ ,  $Q_1$  toggles after every active clock edge.

\*  $J_2 = Q_1$ ,  $K_2 = \overline{Q_1}$ . We need to look at  $J_2$  and  $K_2$  values just before the active edge, to determine the next value of  $Q_2$ .

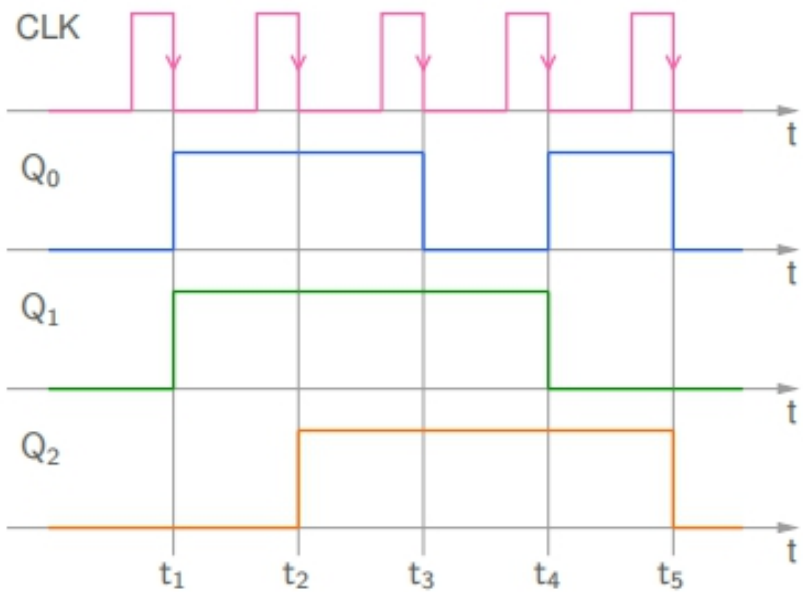
\* It is convenient to construct a table listing  $J_2$  and  $K_2$  to figure out the next  $Q_2$  value.

# JK flip-flop

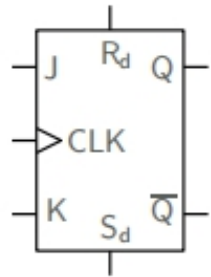


CLK	J	K	$Q_{n+1}$
↓	0	0	$Q_n$
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q_n}$

	$t_k^-$									$t_k^+$		
t	$Q_0$	$Q_1$	$Q_2$	$J_0$	$K_0$	$J_1$	$K_1$	$J_2$	$K_2$	$Q_0$	$Q_1$	$Q_2$
$t_1$	0	0	0	1	0	1	1	0	1	1	1	0
$t_2$	1	1	0	1	0	0	0	1	0	1	1	1
$t_3$	1	1	1	1	1	0	0	1	0	0	1	1
$t_4$	0	1	1	1	1	1	1	0	0	1	0	1
$t_5$	1	0	1	1	1	0	0	1	1	0	0	0



# JK flip-flop: asynchronous inputs

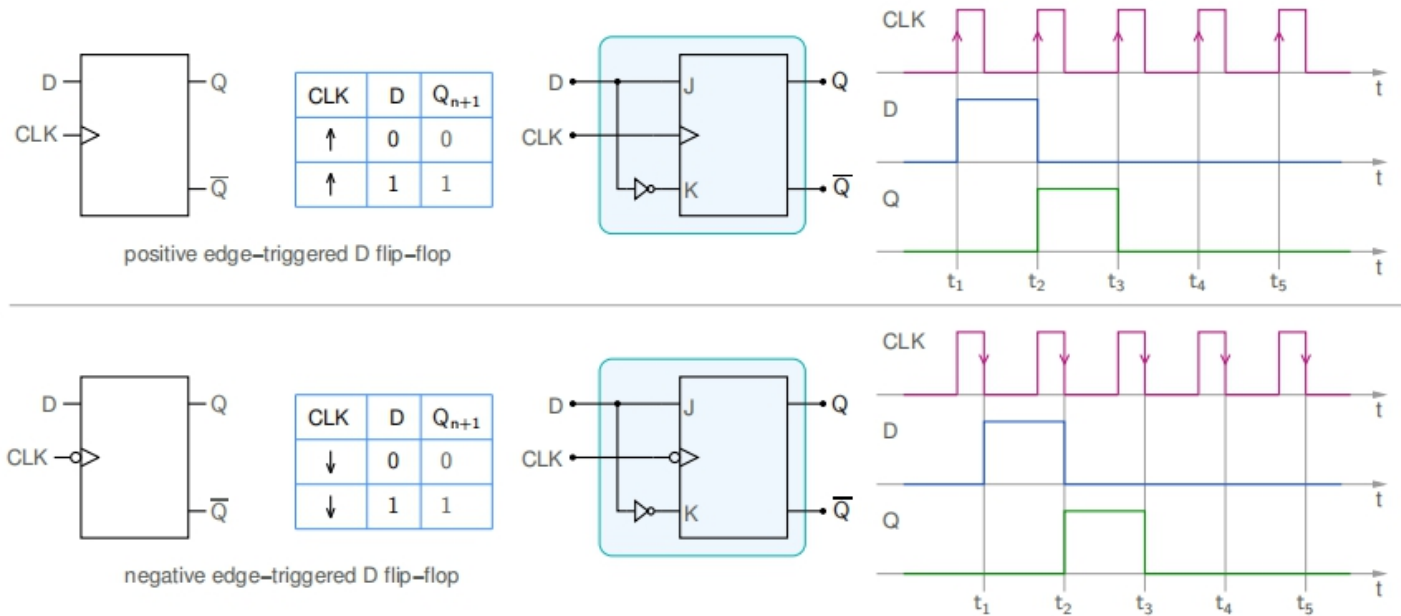


$S_d$	$R_d$	CLK	J	K	$Q_{n+1}$
0	1	X	X	X	0
1	0	X	X	X	1
1	1	X	X	X	invalid
0	0	↑	0	0	$Q_n$
0	0	↑	0	1	0
0	0	↑	1	0	1
0	0	↑	1	1	$\overline{Q_n}$

} normal operation

- \* Clocked flip-flops are also provided with asynchronous or direct Set and Reset inputs,  $S_d$  and  $R_d$ , (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).
- \* The  $S_d$  and  $R_d$  inputs may be active low; in that case, they are denoted by  $\overline{S_d}$  and  $\overline{R_d}$ .
- \* The asynchronous inputs are convenient for starting up a circuit in a known state.

# D flip-flop



- \* The D flip-flop can be used to delay the Data (D) signal by one clock period.
- \* With  $J = D$ ,  $K = \overline{D}$ , we have either  $J = 0$ ,  $K = 1$  or  $J = 1$ ,  $K = 0$ ; the next  $Q$  is 0 in the first case, 1 in the second case.
- \* Instead of a JK flip-flop, an RS flip-flop can also be used to make a D flip-flop, with  $S = D$ ,  $R = \overline{D}$ .