EE2000 Logic Circuit Design

Chapter 7 – Latch and Flip-Flog Circuits

Outline

- 7.1 Latch
  - Active High
  - Active Low
- 7.2 Flip-Flop (FF)
  - SR-, D-, JK-, T- FFs
- 7.3 Master-Slave FF
- 7.4 Edge-Triggered FF
- 7.5 Synchronous and Asynchronous Inputs

Two classes of logic circuits

- Combinational logic circuit

  Previous lectures have been discussed

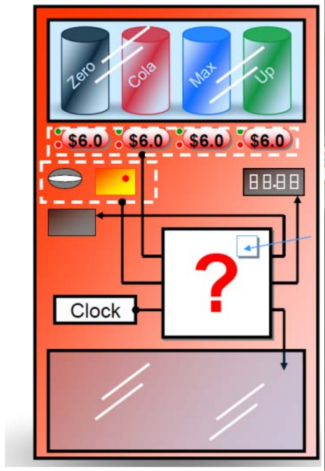
- Sequential logic circuit

  Output depends on present input + past history

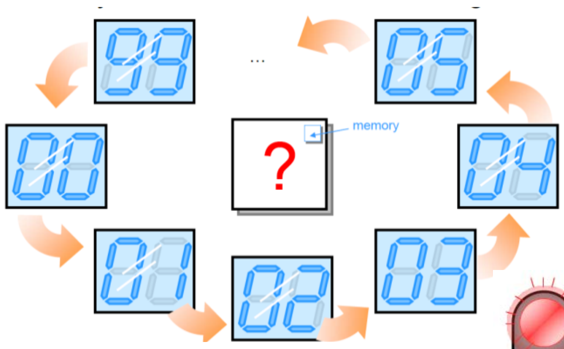
  Memory circuit (previous STATE information) is required

Sequential logic circuit

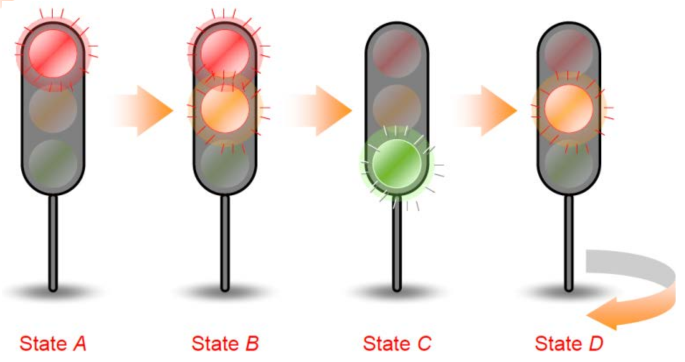
Examples



Vending Machine
Food/ drink / MTR tickets



Digital counter / clock

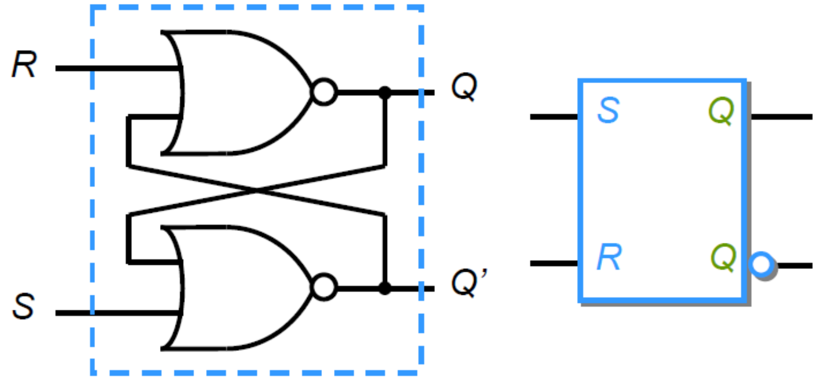


Traffic light controller

7.1 Latch

- The output state of a *latch* is controlled by its excitation input signals
- *Latch* and *flip-flop* are memory devices and implemented using *bistable* circuit - its output will remain in either 0 or 1 state

  - The formal logic diagram and symbol



Each label has different meanings:

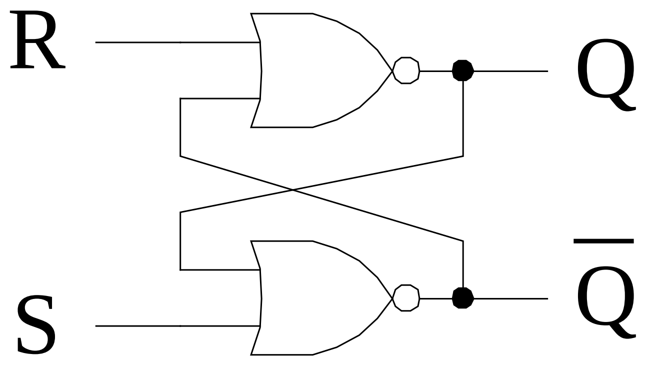
R stands for **reset input**
S stands for **set input**
Q and *Q'* define the **state** of the latch

  - A binary storage device
    - Q stores the state of the latch, or view as
    - Storing the data (a binary value) in Q

Latch

- <u>SR (reset-set) latch circuit</u>

  When S (set) is set to 1, the output Q will be set to 1. Likewise, when R (reset) is set to 1, the output Q will be set to 0. It is invalid to set both S and R to 1
- ***Active High Latch***



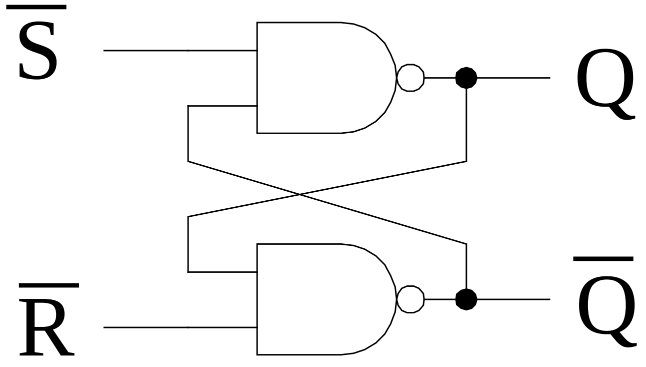
<u>NOR gate implementation</u>

Next state of Q

S	**R**	Q_{t+1}	Q'_{t+1}	**State**
0	0	Q_t	Q'_t	Hold state
0	1	0	1	Reset state
1	0	1	0	Set state
1	1	0	0	Undefined state

Latch

- S'R' latch circuit
  When S' (set) is set to 1, the output Q will be set to 0. Likewise, when R' (reset) is set to 1, the output Q will be set to 1. It is invalid to set both S' and R' to 1
- ***Active Low Latch***

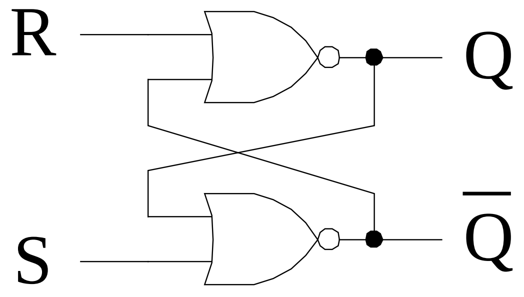


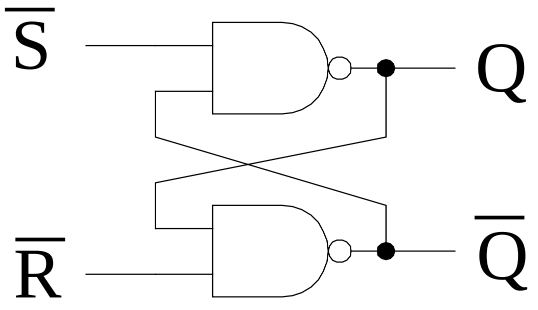
NAND gate implementation

S'	R'	Q_{t+1}	Q'_{t+1}	State
1	1	Q_t	Q'_t	Hold state
1	0	0	1	Reset state
0	1	1	0	Set state
0	0	1	1	Undefined state

Latch

- Note that the input is ***active high*** for NOR gate implementation, whereas the input is ***active low*** for NAND gate implementation





Require a 1 signal to change its state

Require a 0 signal to change its state

Drawbacks

- If inputs *S*, *R* are unstable (or noise), output produces a unstable short pulse

Solution:

- Prevention of input noise
- Using “Gated Latch”

- The latches will enter **undefined state** for some inputs (e.g. $R = S = 1$)

Solution:

- Prevent *R* and *S* set to 1 simultaneously
- Using “D Latch”

7.2 Flip-Flop (FF)

- The state of flip-flop changes on the transition of the clock
- Four types of FF

  SR FF

  D-type FF

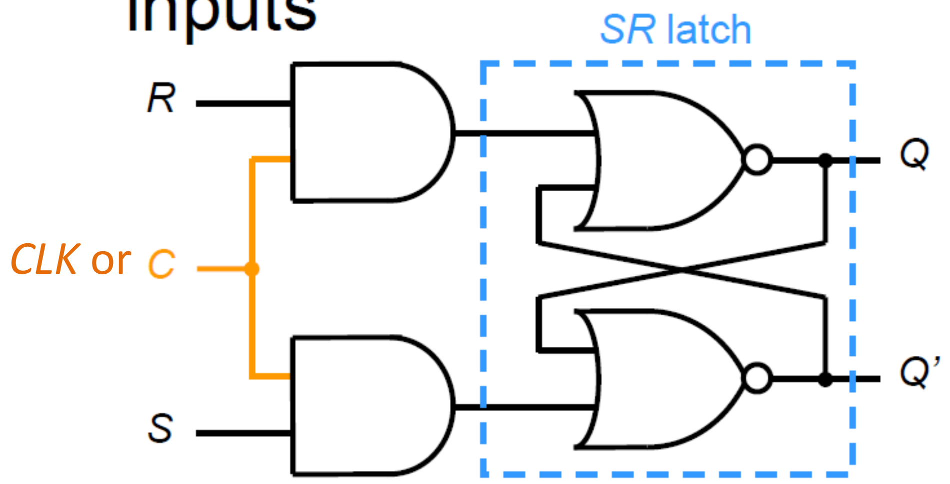
  JK FF

  T FF

Clocked SR FF

C can be presented by a clocked signal. Then the latch with the clocked signal is called Flip-flop

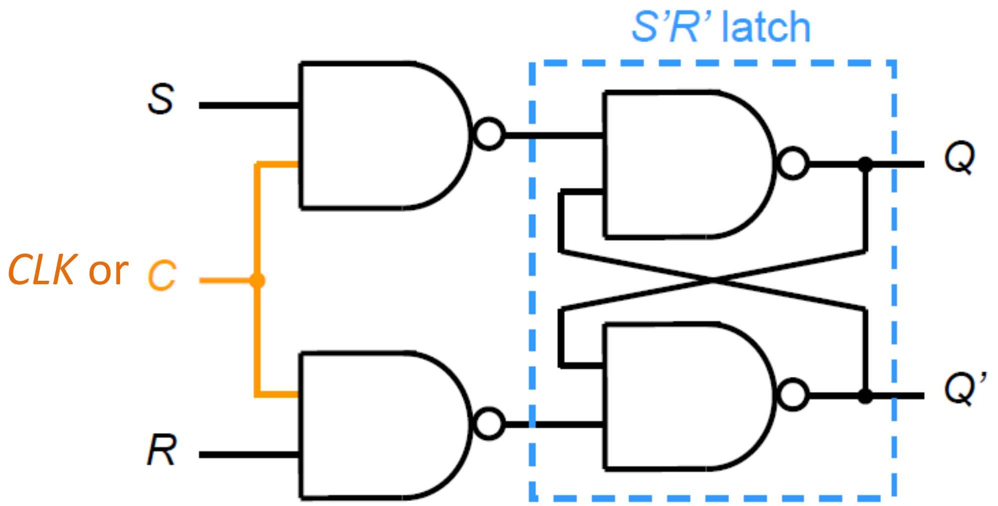
- An additional control input *C* is introduced to act as an enable signal for the *R*, *S* inputs



C	*S*	*R*	State
0	x	x	Disable (Hold)
1	0	0	Hold state
1	0	1	Reset state
1	1	0	Set state
1	1	1	Undefined state

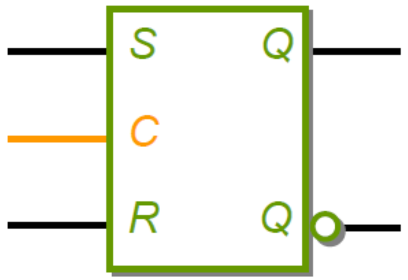
When $\underline{C = 0}$, the two inputs to the *SR* latch are (0, 0), output *Q* will be unchanged (state change disabled)
When $\underline{C = 1}$, the two inputs to the *SR* latch are (*S*, *R*), output *Q* will follow that of the *SR* latch (state change enabled)

Another Version



C	S'	R'	State
0	x	x	Disable (Hold)
1	1	1	Hold state
1	1	0	Reset state
1	0	1	Set state
1	0	0	Undefined state

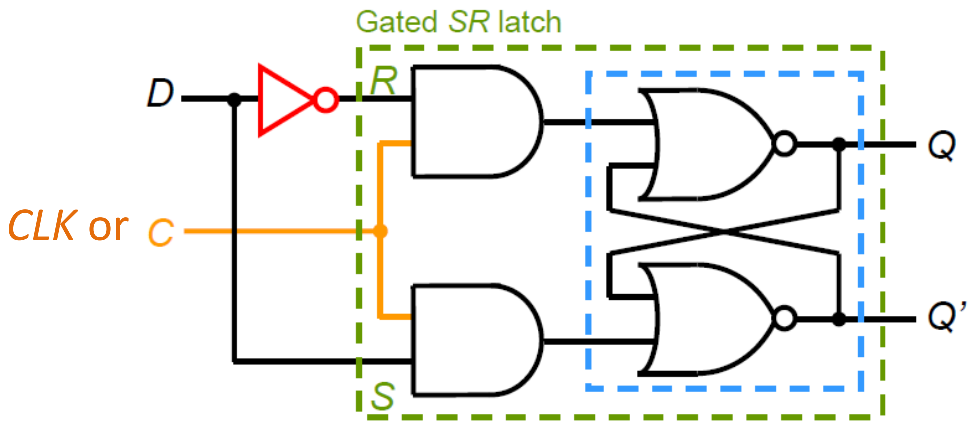
- The control input is also known as enable input, gate
- So this kind of latches are called gated latches

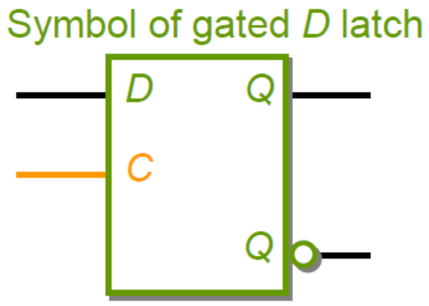


Logic symbol of gated *SR* latch

D-type FF

C can be presented by a clocked signal.
Inputs *R* and *S* are correlated with *D* but differentiate by a NOT-gate





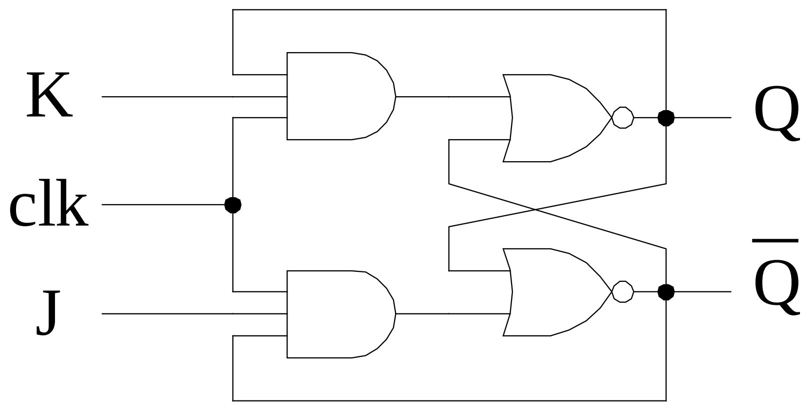
- *S* = *D*, *R* = *D*'
- When *D* = 0,
  - *S* = 0, *R* = 1 (reset state)
- When *D* = 1,
  - *S* = 1, *R* = 0 (set state)
- No undefined state!

C	D	Q	Next state of Q
0	x	*Q*	Hold state
1	0	0 (= *D*)	Reset state
1	1	1 (= *D*)	Set state

Remember in this way:
The value of *D* is transferred to the *Q* output

JK FF

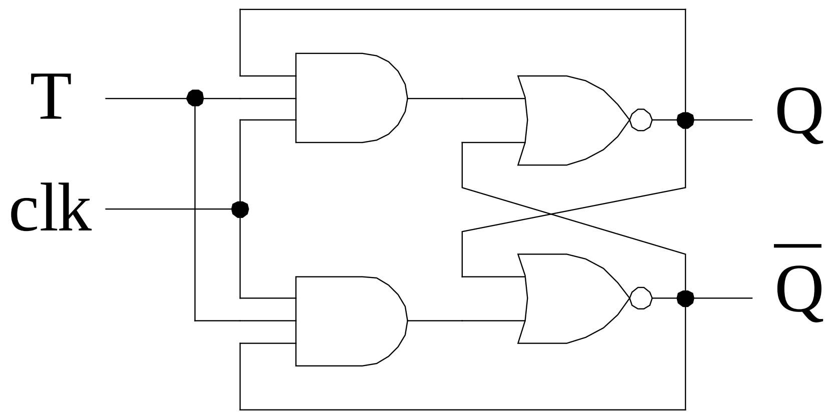
- The *JK* FF is a refinement of the *SR* FF in that the undetermined state of the *SR* type is defined in the JK type. Inputs *J* and *K* behave like inputs *S* and *R* to set and reset (clear) the FF, respectively. The input marked *J* is for *set* and the input marked *K* is *reset.*



		clk=0	clk=1
J	K	Q_{n+1}	Q_{n+1}
0	0	Q_n	Q_n
0	1	Q_n	0
1	0	Q_n	1
1	1	Q_n	$\overline{Q}_n$

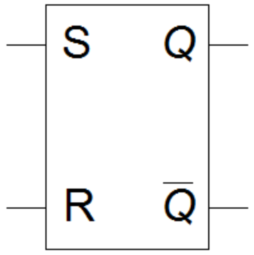
T-type FF

- The toggle (*T*) FF has a clock input which causes the output state changed for each clock pulse if *T* is in its active state. It is useful in counter design

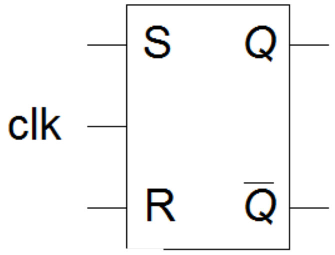


clk	T	Q_{n+1}
0	X	Q_n
1	0	Q_n
1	1	$\overline{Q}_n$

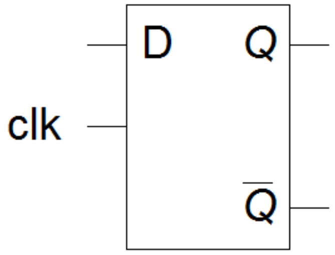
Logic symbols of various latch and flip-flops



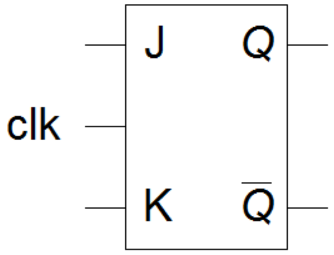
SR latch



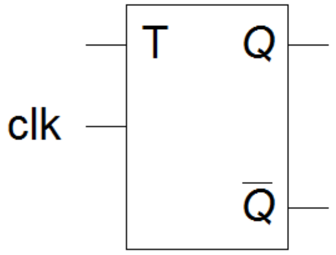
SR flip-flop



D flip-flop



JK flip-flop

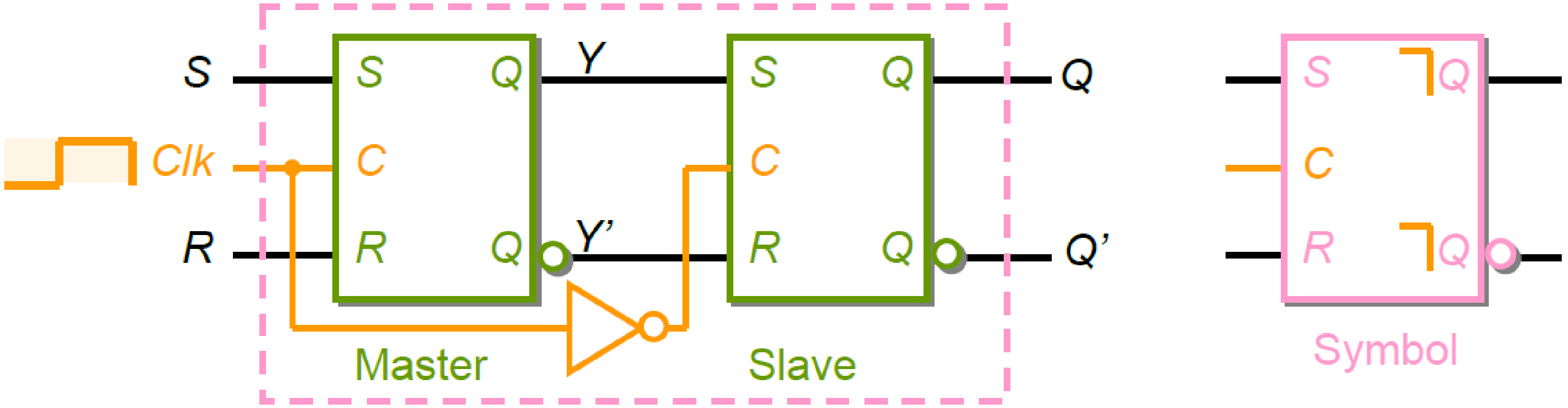


T flip-flop

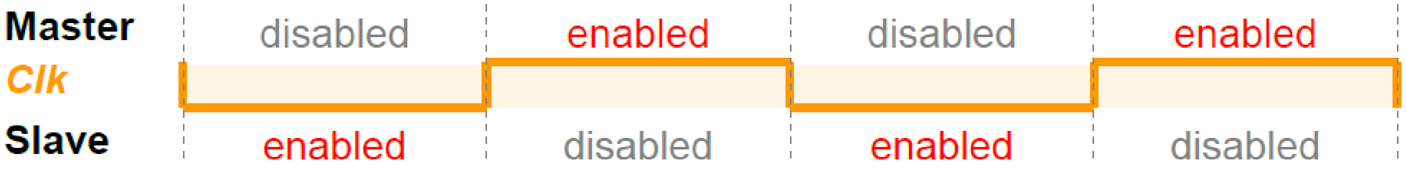
7.3 Master-Slave Flip-Flop

- Master-Slave SR Flip-Flop

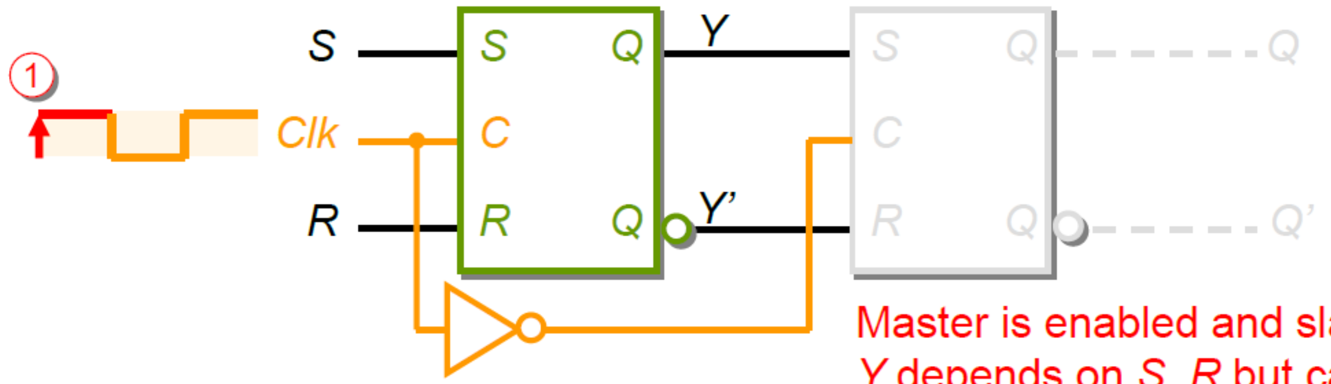
- Connect two gated *SR* latches like this



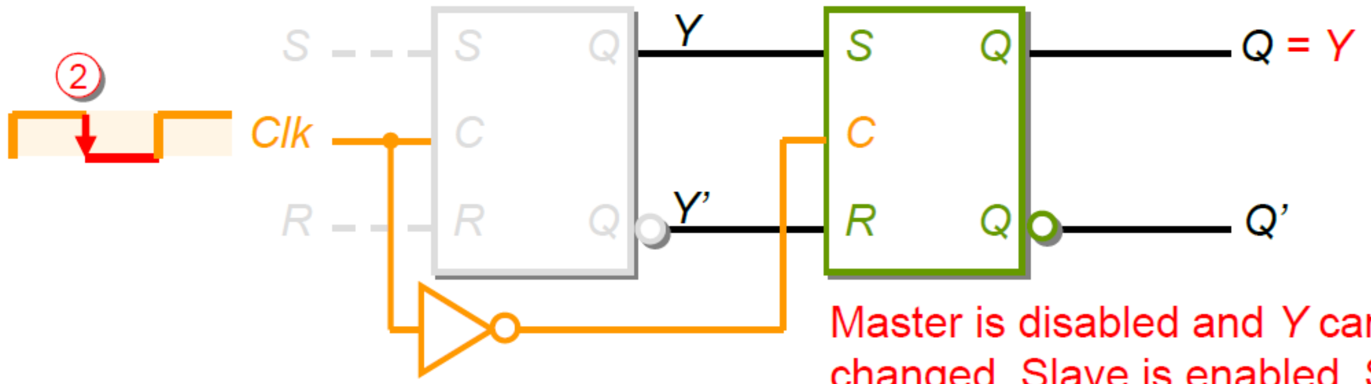
- The left clocked latch called **master**, the right is **slave**
- At any time, only one latch could be enabled



How Does It Work?

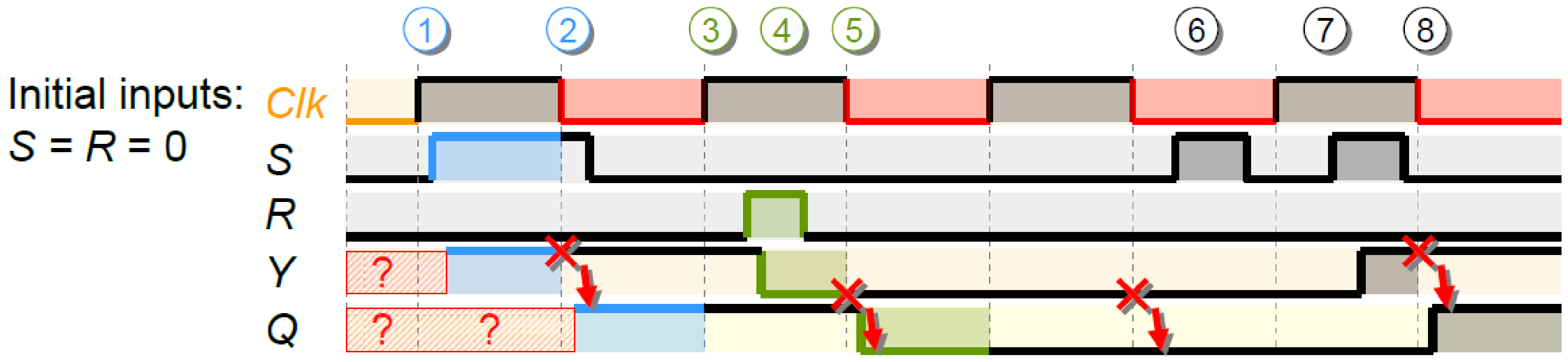


Master is enabled and slave is disabled. *Y* depends on *S*, *R* but cannot affect *Q*



Master is disabled and *Y* can't be changed. Slave is enabled. So the current value of *Y* is transferred to *Q*

Timing Diagram of MS *SR* FF



1. *Clk* goes to 1, *S* affects the output of *Y* (master enabled)
2. *Clk* goes to 0, the output of *Q* is enabled (slave enabled)

3. *Clk* goes to 1 again
4. *R* goes to 1, *Y is* 0 (reset enabled)
5. The new *Y* is transferred to Q when *Clk* goes back to 0

6. Now *Clk* is 0 (master disabled), so any change in *S*, *R* can't affect *Y*
7. *S* goes to 1, and back to 1, *Y* set to 1 again
8. *Q* follows *Y* until *Clk* falls from 1 to 0

Pulse-Triggered FF

Its behavior depends on:
The rising and the period of the pulse remain at high (*Y*) and the falling edges (*Q*)
So referred to as **pulse-triggered flip-flops**

- When the clock pulse goes to 1
  - Master is enabled
  - Output *Y* is updated at this moment
- When the clock pulse remains at 1
  - *Y* will continue be changed if *S* and *R* change
- When the clock pulse goes to 0
  - Slave is enabled
  - Output *Q* is updated now
  - The state of the FF is changed (i.e. triggered)
- When the clock pulse remains at 0
  - *Q* remain unchanged as *Y* can't be changed

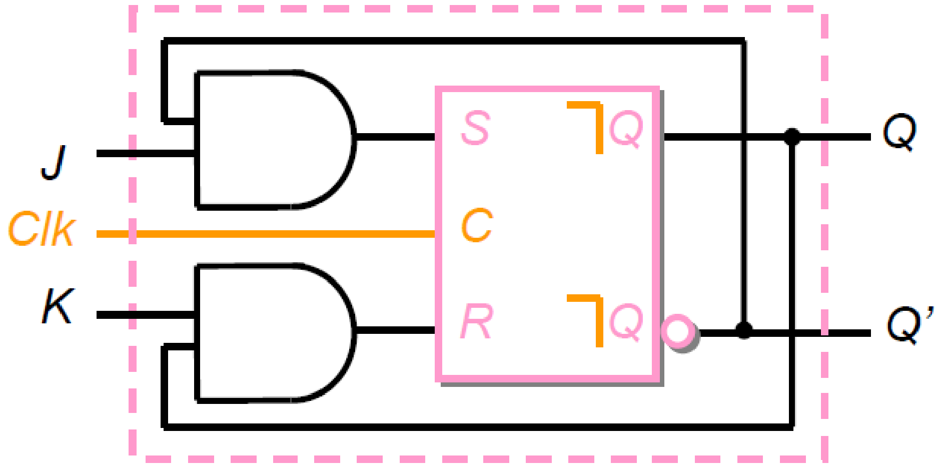
Behavior of the MS *SR* FF

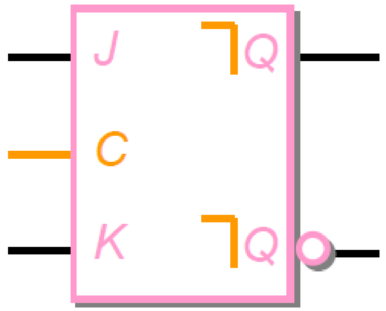
The function table

Inputs			**Outputs**		
S	***R***	***Clk***	**Next *Q***	**Next *Q'***	**Meaning**
x	x	0	*Q*	*Q'*	Hold
0	0	⎍	*Q*	*Q'*	Hold
0	1	⎍	0	1	Reset
1	0	⎍	1	0	Set
1	1	⎍	?	?	Undefined

Master-Slave *JK* Flip-Flop

- Extension of master-slave *SR* flip-flop





Symbol

- Want to eliminate the undefined state
  - $S = JQ'$, $R = KQ$

How to Operate?

- $S = JQ'$, $R = KQ$
- 1) When $J = K = 0$
  - $S = 0$, $R = 0$ (Hold state)
- 2) When $J = 0$, $K = 1$
  - $S = 0$, $R = Q$
  - Hold state if Q is 0
  - Reset if Q is 1 (but Q will be 0 in next clock pulse)
  - Therefore, reset state
- 3) When $J = 1$, $K = 0$
  - $S = Q'$, $R = 0$
  - Hold state if Q is 1
  - Set if Q is 0 (but Q will be 1 in next clock pulse)
  - Therefore, set state
- 4) When $J = 1$, $K = 1$
  - $S = Q'$, $R = Q$
  - Reset if Q is 1 (next $Q = 0$)
  - Set if Q is 0 (next $Q = 1$)
  - Therefore, toggle the value of Q

Behavior of the MS *JK* FF

The function table

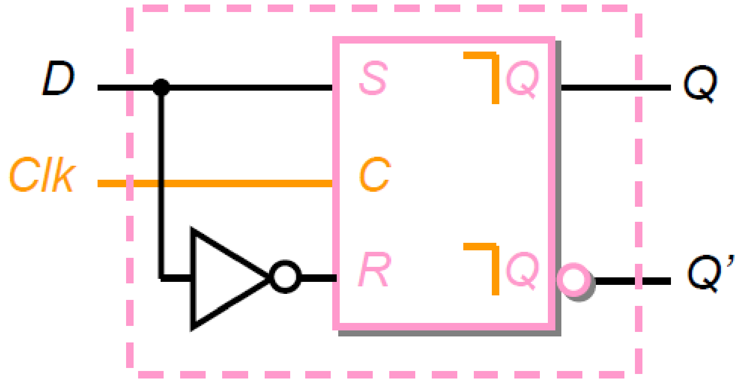
Inputs			Outputs		
J	***K***	***Clk***	**Next *Q***	**Next *Q'***	**Meaning**
x	x	0	Q	Q'	Hold
0	0	⎍	Q	Q'	Hold
0	1	⎍	0	1	Reset
1	0	⎍	1	0	Set
1	1	⎍	Q'	Q	Toggle

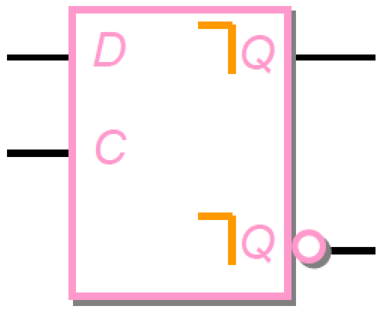
The same behaviors as master-slave *SR* flip-flop

Eliminated the undefined state

Master-Slave *D* Flip-Flop

- Extension of master-slave *SR* flip-flop



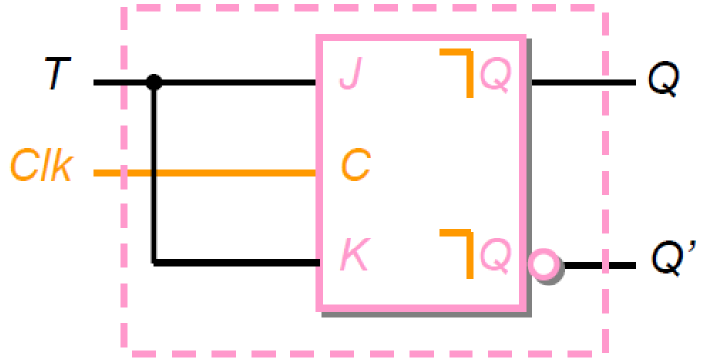


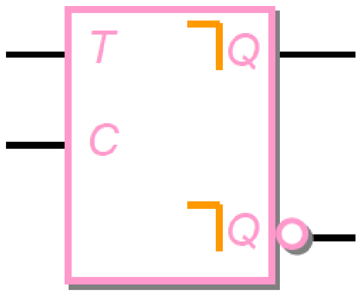
Symbol

- *D* stands for delay
  - *S* = *D*, *R* = *D'* (Set *Q* if *D* is 1, reset *Q* if *D* is 0)
  - The output *Q* is just the input delayed *D* until the next active clock transition

Master-Slave *T* Flip-Flop

- Extension of master-slave *JK* flip-flop





Symbol

- $J = K = T$
  - When $T = 0$, $J = K = 0$ (Hold)
  - When $T = 1$, $J = K = 1$ (Toggle)

MS *D* FF & MS *T* FF

The function tables

Inputs		**Outputs**		
D	***Clk***	**Next *Q***	**Next *Q’***	**Meaning**
x	0	Q	Q’	Hold
x	⎍	*D*	*D’*	Set / Reset

Inputs		**Outputs**		
T	***Clk***	**Next *Q***	**Next *Q’***	**Meaning**
x	0	Q	Q’	Hold
0	⎍	Q	Q’	Hold
1	⎍	Q’	Q	Toggle

Problems

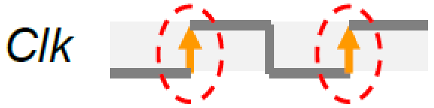
- For MSFF, master is enabled during the period of clock pulse is 1
- Undesired behavior produced when inputs values of *S* and *R* (or *J*, *K*, *D*, *T*) change at that period
- Better solution: Edge-triggered flip-flops

7.4 Edge-Triggered Flip-Flop

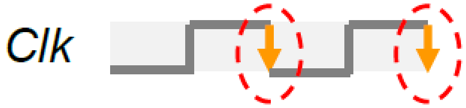
- Use one of the edges of the clock signal to read the input values
  - either positive or negative transition
  - triggering edge
- The response to the triggering edge at the output of the flip-flop is almost immediate
- The flip-flops remains unresponsive to the input change until the next triggering edge

Triggering Edge

- Triggered only during a signal transition from 0 to 1 (or from 1 to 0) on the clock
  - Positive (rising, leading) edge-triggering: 0-to-1 transition

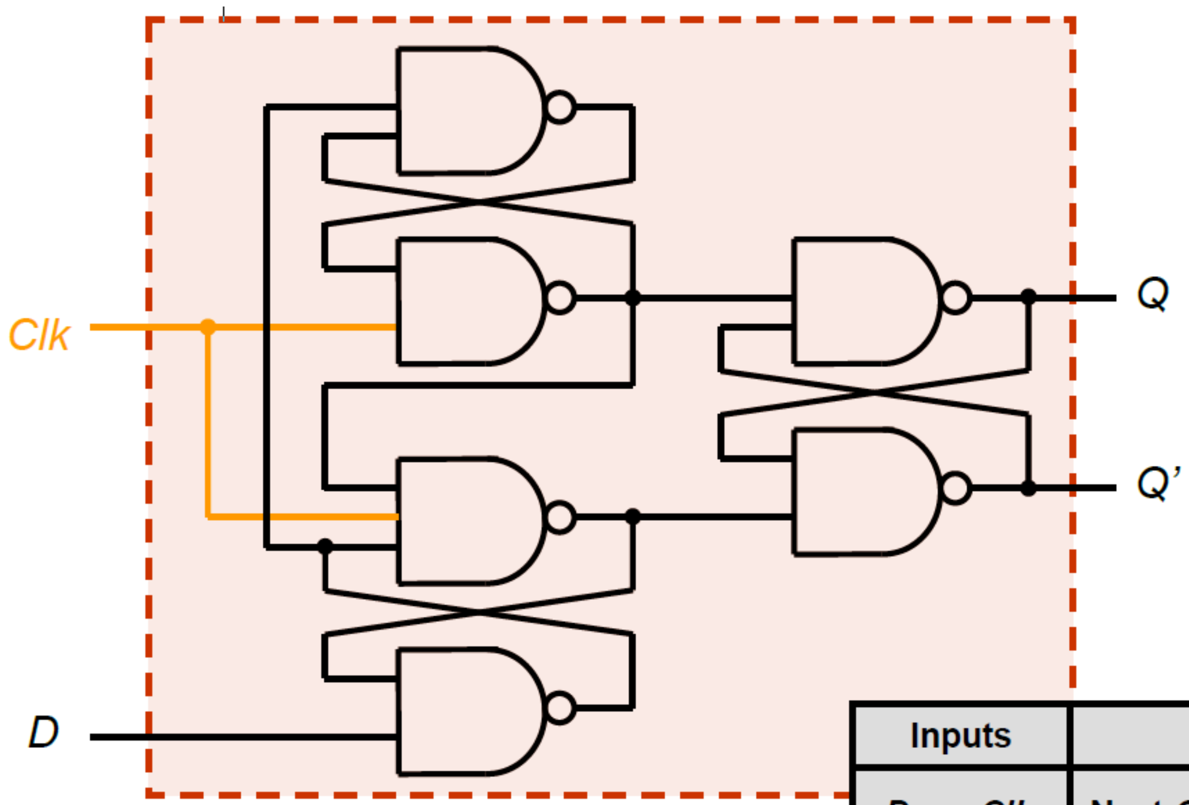


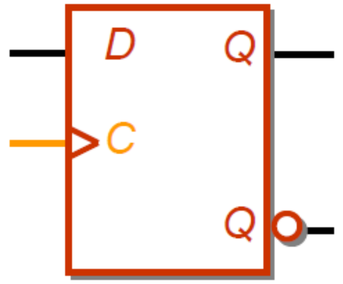
  - Negative (falling, trailing) edge-triggering: 1-to-0 transition



- Additional circuit is included to ensure it will only response to the input at the transition edge of the clock pulse

Positive-Edge-Triggered *D* FF



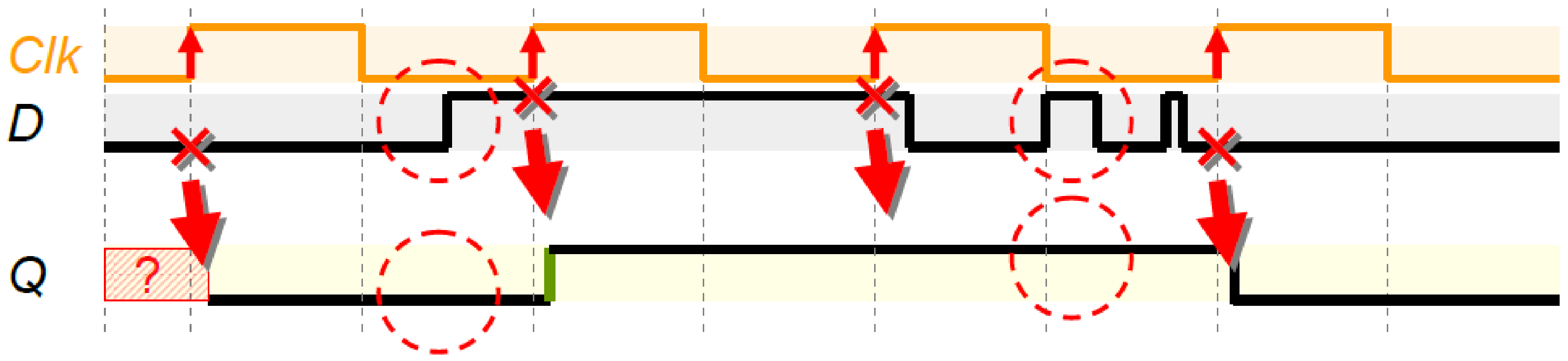


Remember the symbol instead of the configuration

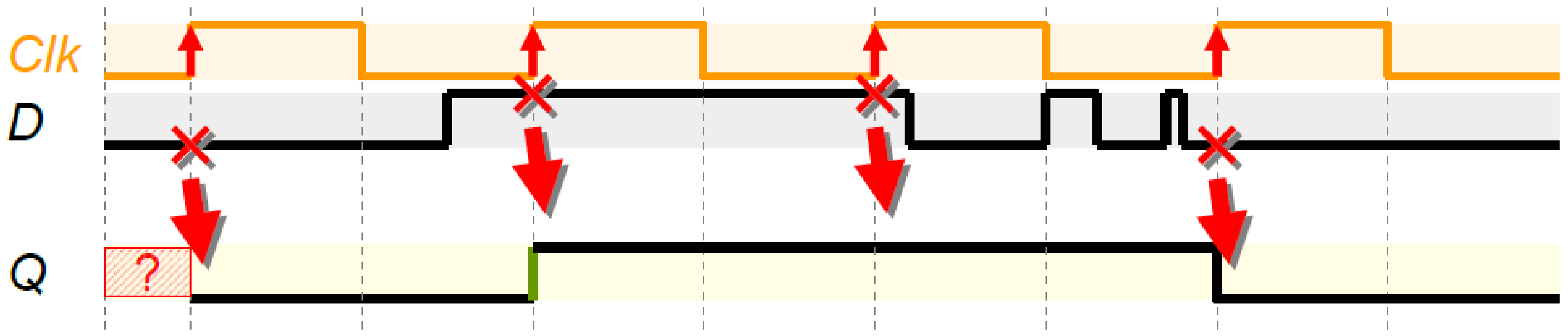
Inputs		Outputs		
D	*Clk*	Next *Q*	Next *Q'*	State
x	↓,0,1	*Q*	*Q'*	Hold
x	↑	*D*	*D'*	Set / Reset

Timing Diagram

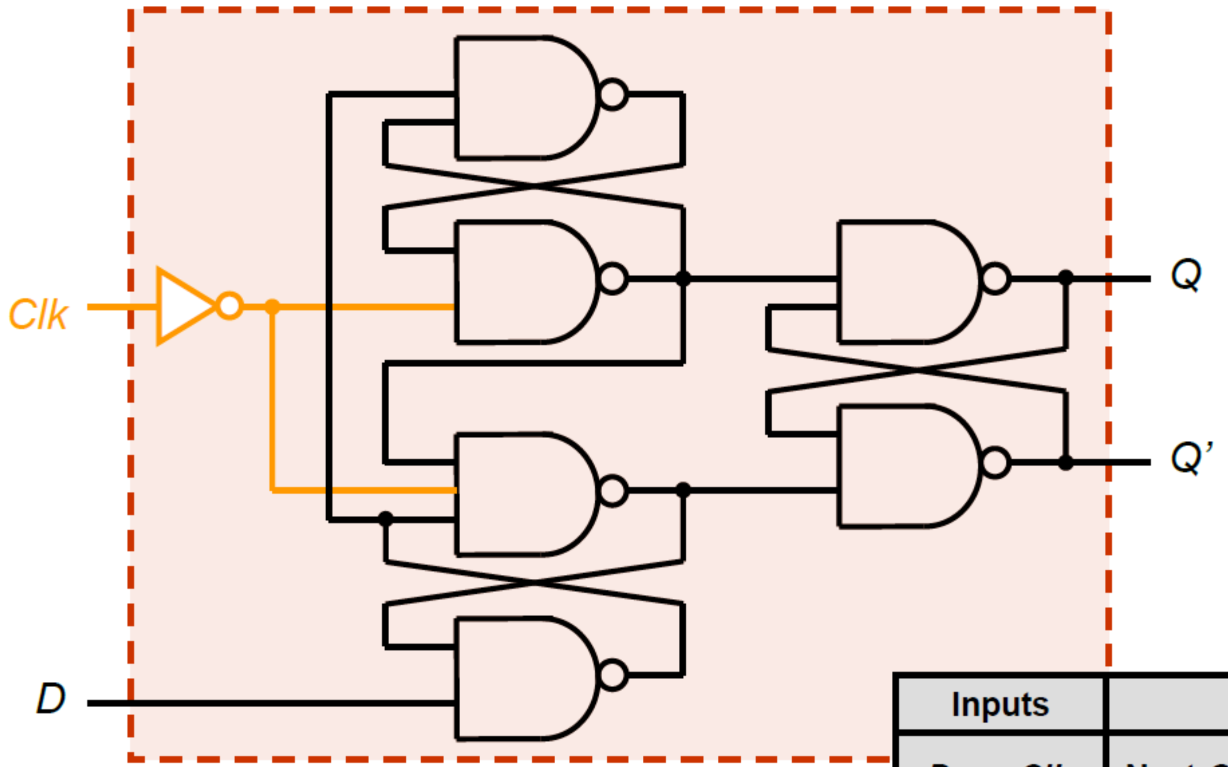
Timing Diagram of PET *D* FF

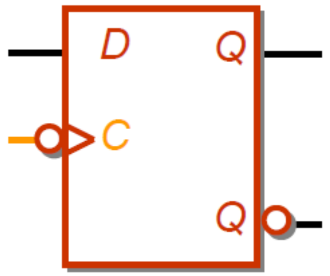


Timing Diagram of PET *D* FF (if no output delay)



Negative-Edge-Triggered *D* FF

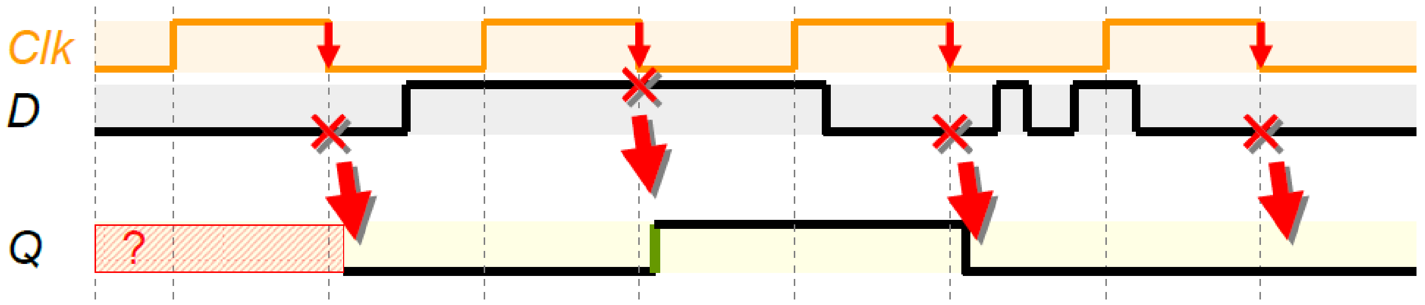




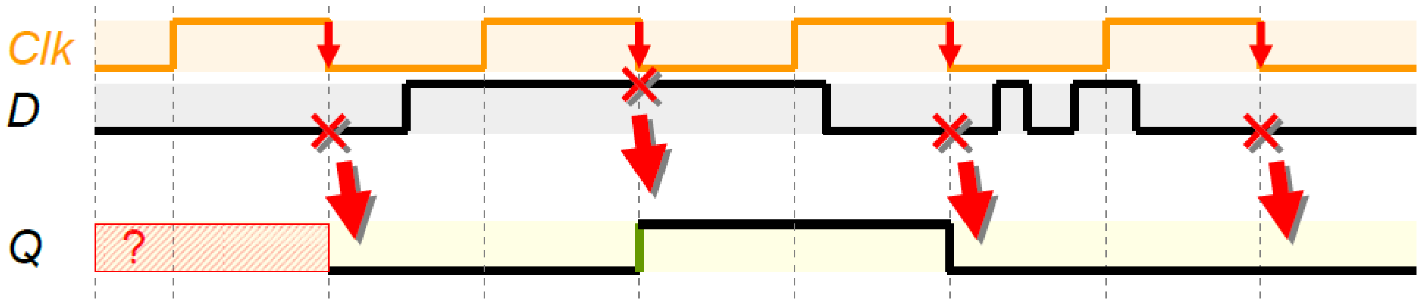
Inputs		Outputs		
D	*Clk*	Next *Q*	Next *Q’*	State
x	↑,0,1	*Q*	*Q’*	Hold
x	↓	*D*	*D’*	Set / Reset

Timing Diagram

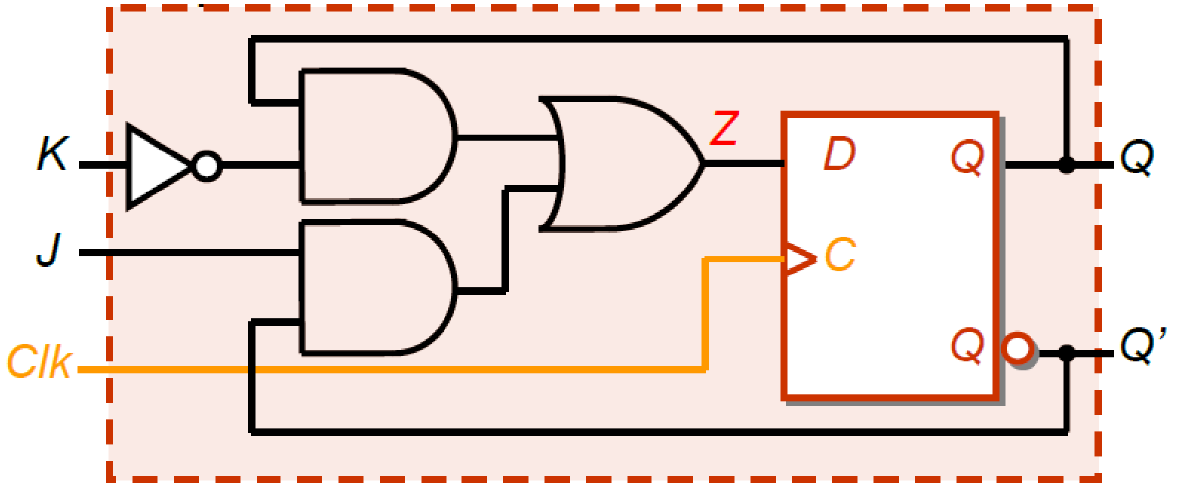
Timing Diagram of NET *D* FF

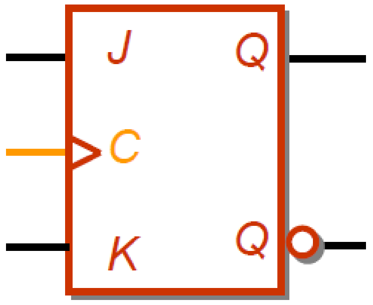


Timing Diagram of NET *D* FF (if no output delay)



Edge-Triggered *JK* Flip- Flop

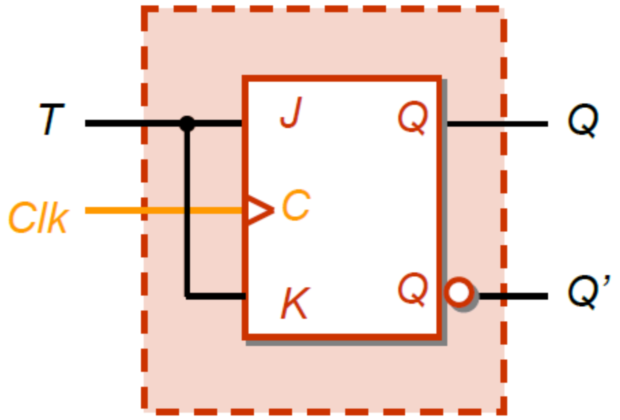


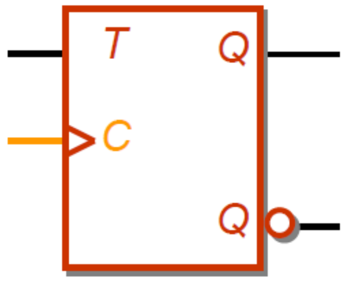


Inputs			Outputs		
J	***K***	***Clk***	**Next *Q***	**Next *Q'***	**State**
x	x	↓,0,1	Q	Q'	Hold
0	0	↑	Q	Q'	Hold
0	1	↑	0	1	Reset
1	0	↑	1	0	Set
1	1	↑	Q'	Q	Toggle

Edge-Triggered *T* Flip- Flop

- Assign *J* = *K* = *T*
  - When *T* = 0 (*J* = 0 and *K* = 0), hold state
  - When *T* = 1 (*J* = 1 and *K* = 1), toggle state
  - Remove the reset and set state

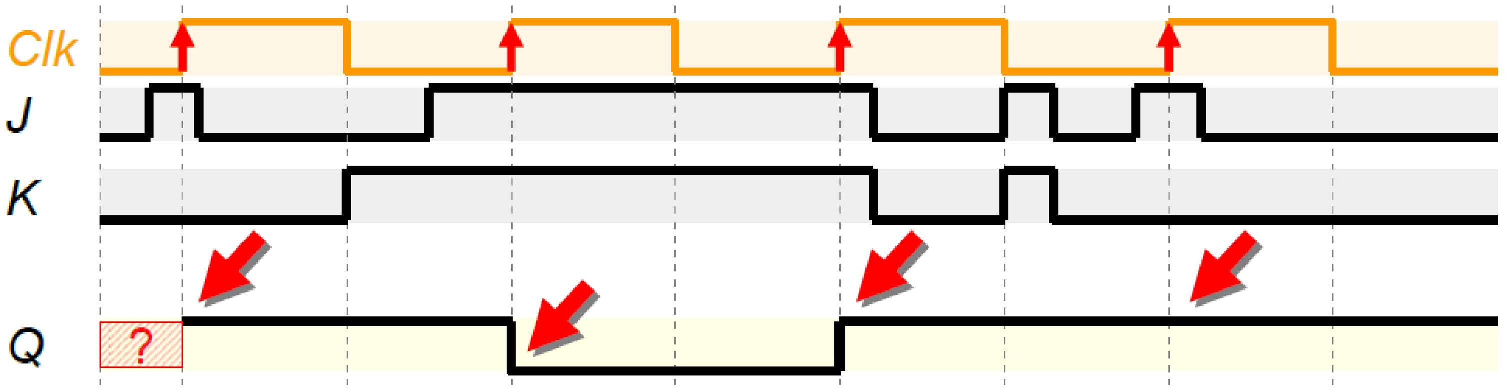




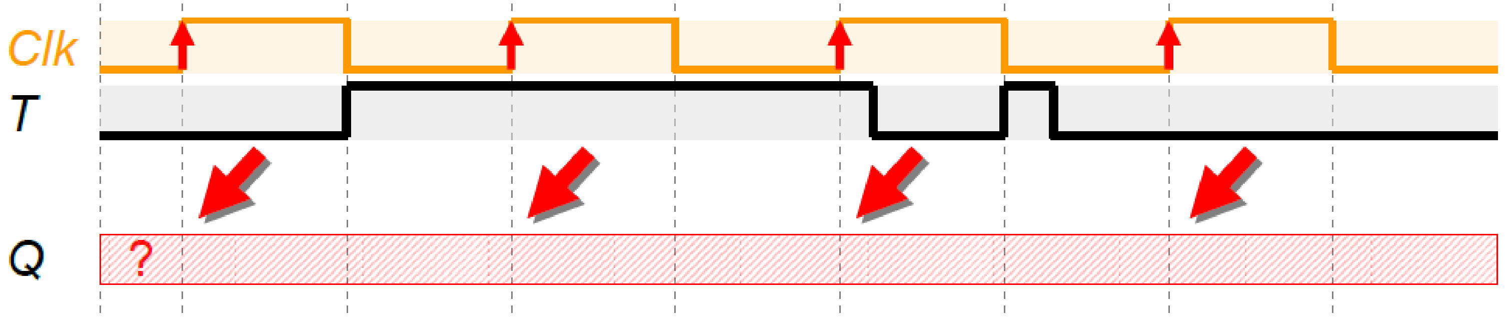
Inputs		Outputs		
T	*Clk*	Next *Q*	Next *Q'*	State
x	↓,0,1	Q	Q'	Hold
0	↑	Q	Q'	Hold
1	↑	Q'	Q	Toggle

Timing Diagram

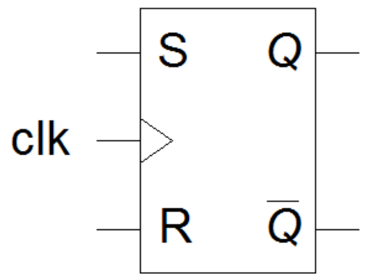
Timing Diagram of PET *JK* FF (if no output delay)



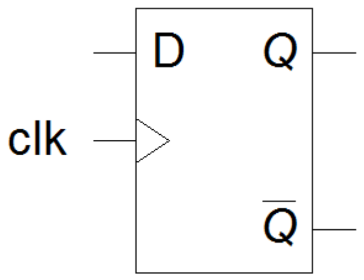
Timing Diagram of PET *T* FF (if no output delay)



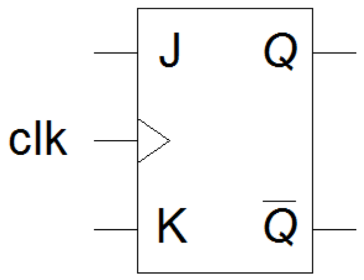
*Logic symbols of various **edge-triggered** flip-flops*



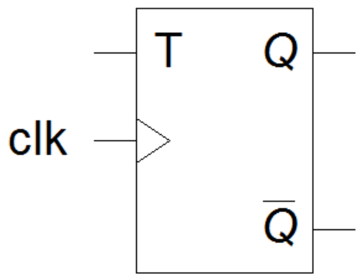
SR flip-flop



D flip-flop



JK flip-flop



T flip-flop

7.5 Synchronous and Asynchronous Inputs

- Inputs of flip-flops are categorized into two types
- **Synchronous** and **asynchronous**
- All inputs to FFs presented so far are synchronous inputs

Asynchronous Inputs

- The output will response to the sync. inputs only if all async. inputs are inactive
- Particularly useful for bringing a FF into a desired initial state prior to normal clocked operation
- The asynchronous (or direct) inputs usually called
  - **Preset** (denoted by *PRE*) and **clear** (denoted *CLR*)
  - Used to forcibly set and reset the state of flip-flop

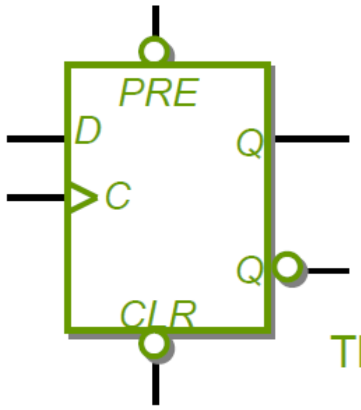
Example

The corresponding function table

Inputs				**Outputs**		
PRE'	***CLR'***	***Clk***	***D***	**Next *Q***	**Next *Q'***	**Meaning**
0	1	x	x	1	0	Preset
1	0	x	x	0	1	Clear
0	0	x	x	1*	1*	Undefined
1	1	↑	x	*D*	*D'*	Set / Reset
1	1	↓,0,1	x	Q	Q'	Hold

*Unpredictable behavior will result if both *PRE'* and *CLR'* set to 1 simultaneously

The same behaviors as *D* flip-flops



The logic symbol

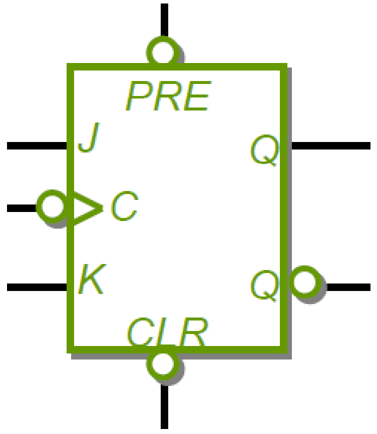
Example

The function table of an negative-edge-triggered *JK* flip-flop with asynchronous preset and clear inputs

Inputs					**Outputs**		
PRE'	***CLR'***	***Clk***	***J***	***K***	**Next *Q***	**Next *Q'***	**Meaning**
L	H	x	x	x	H	L	Preset
H	L	x	x	x	L	H	Clear
L	L	x	x	x	H*	H*	Undefined
H	H	↓	L	L	Q	Q'	Hold
H	H	↓	H	L	H	L	Set
H	H	↓	L	H	L	H	Reset
H	H	↓	H	H	Q'	Q	Toggle
H	H	↑,0,1	x	x	Q	Q'	Hold

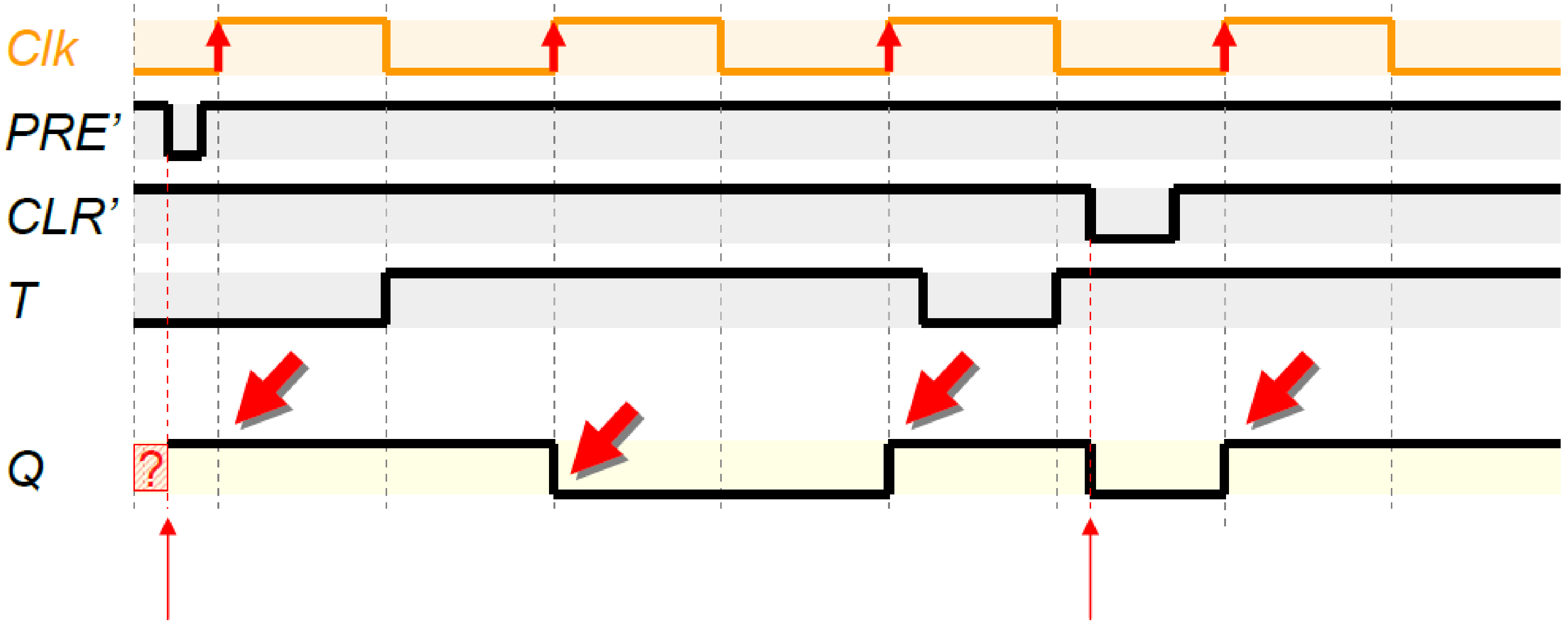
The same behaviors as *JK* flip-flops

The logic symbol



Timing Diagram

Timing Diagram of PET *T* FF with asynchronous preset and clear inputs (if no output delay)



Response to asynchronous input *PRE'*

Response to asynchronous input *CLR'*

Summary

- Flip-flips as memory elements
  - To store the value in *Q*
  - One FF can store 1-bit data

- Different types of FFs
  - *SR*-type, *D*-type, *JK*-type, *T*-type
  - By controlling the inputs (*S, R, D, J, K, T*) to change, set, reset or hold the value of *Q*

- Different Triggering Methods
  - Pulse-triggered
  - Edge-triggered