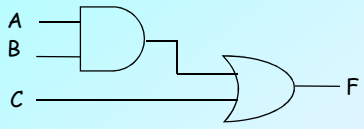
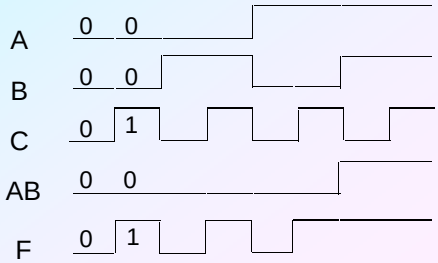
Timing Diagrams

- They describe the temporal behavior (input/output relation) of digital circuits.
- Time is drawn on the horizontal axis (x-axis), output of the digital circuit (0/1 or L/H) is drawn on the vertical axis.
- In more detailed timing diagrams, values of the outputs are written in terms of electrical voltage or current.
- Some of the physical phenomena cannot be shown on the truth table. In these cases timing diagrams should be used to describe the behavior of the circuit.

Example:



In this diagram only logical behavior of the circuit is shown and time delays (described in the next slides) are ignored.



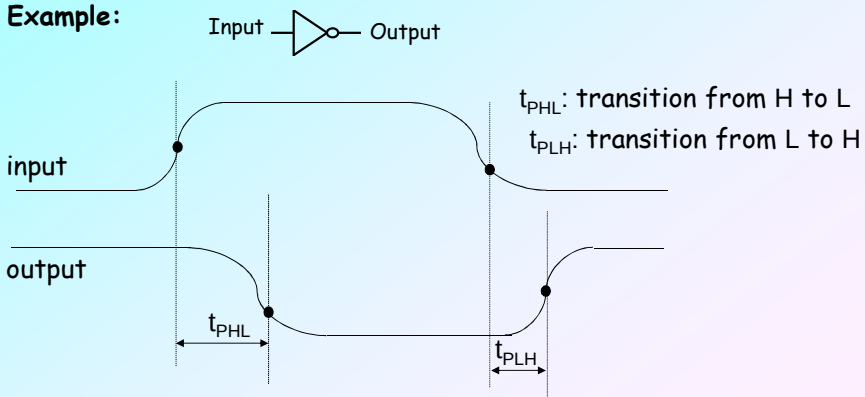
Propagation Delay

Because of the electronic structure of the logic gates, there is time difference between the input (of the logic gate) and the response of the gate (to this input).

The time required for the input signal to propagate inside the logic gate until its effect can be seen at the output is called propagation delay.

Propagation delay determines the speed of the logic circuit.

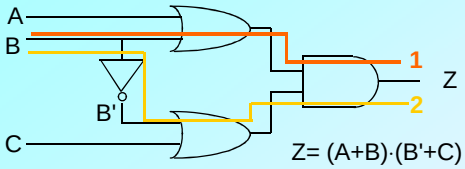
Example:



Hazards caused by propagation delays

If an input propagates through multiple paths to the output, unexpected output values (hazards) may happen at the output.

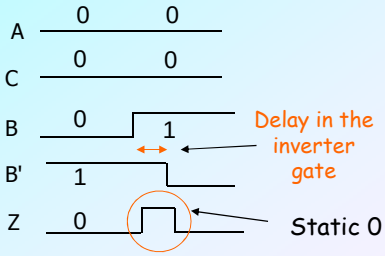
For example, input B has two different paths (shown with red and yellow lines) to the output Z.



When the inputs A, B, and C are all zero (A=0, B=0,C=0), the output will also be zero (Z=0).

In this state, if the value of input B is changed to 1, the output should not change its value (Z=0).

However due to the different propagation delays in the paths from B to Z, a "short" change (hazard) appears at the output.

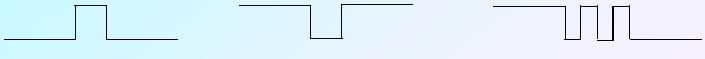


There are three types of hazards:

a) Static 0: The output becomes 1 for a short time and gets back to 0.
Static 0 hazard happens when a function is implemented with product of sums.

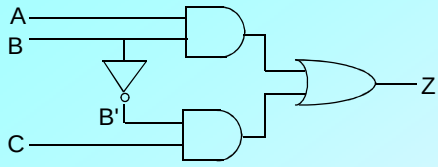
b) Static 1: The output becomes 0 (from 1) and gets back to 1 after a short time.
Static 1 hazard happens when a function is implemented with sum of products.

c) Dynamic: The output oscillates when it changes its value.



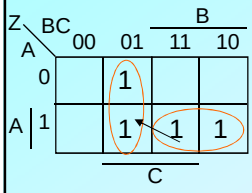
Static 0 Static 1 Dynamic

Avoiding hazards:



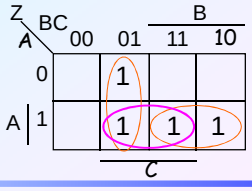
Circuit at the left (implemented as SoP) has output Z=1 when A=1, B=1 and C=1.
In this state if B becomes 0 (1→0), the output should stay as Z=1.
However, **static 1 hazard** occurs at the output.

Possible hazards can be foreseen from the Karnaugh diagram.



Z= AB + B'C

Change in B (1→0) causes a transition from a prime implicant to another. These type of transitions cause hazards because of delays.
To avoid all possible hazards, consensus of the transitions are added to the design which increase the cost.



Z= AB + B'C + **AC**

SEQUENTIAL CIRCUITS

- In the first part of the course, **combinational circuits** are covered.
  The outputs of the combinational circuits depend only to the inputs.
- In **sequential circuits**, the outputs depend both on the inputs and the "state" of the circuit.
  Memory units are required to store (remember) the state of the circuit.
- For example, vending machines keep track of (remember) the coins that were thrown into the machine. With each coin, the state of the machine (total amount of thrown coins) is updated.
- There two types of sequential circuits:
  - **Synchronous sequential circuits**: Their state can change at a discrete instance of time.
    All memory elements are synchronized by a common clock signal. Therefore these circuits are also called "clocked synchronous sequential" circuit.
  - **Asynchronous sequential circuit:** Their state can change at any instant of time depending upon the input signals.
  - In this course we will deal only with clocked synchronous sequential circuits, because nearly all sequential logic today is clocked synchronous.
  - For example microprocessors are clocked synchronous sequential circuits.

Finite State Machine (FSM) Model

Sequential circuits are designed using "finite state machine – FSM" model.

This model is also used in the design of many other systems.

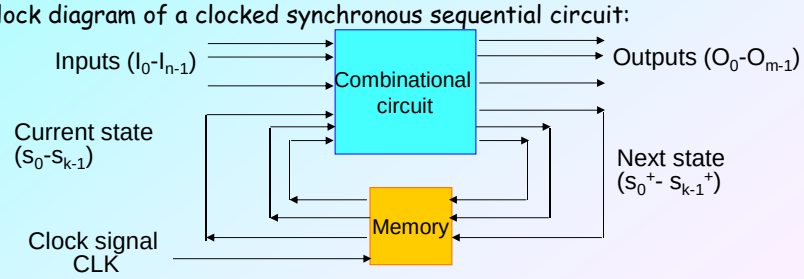
- When the machine is started, it is in a certain state (initial state: s_0).
- An output is produced depending on the inputs and the current state.
- Transition into a new state happens depending on the input and the current state.

A FSM has two parts:

a) Combinational circuit for logical operations

b) Memory unit to remember the current state.

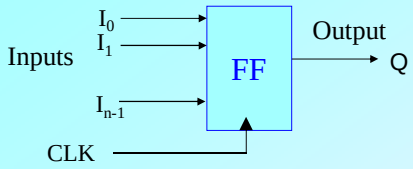
Block diagram of a clocked synchronous sequential circuit:



We will get back to FSM and clocked synchronous sequential circuits (in chapters 7 and 8) after we cover memory units.

Memory Units

'Flip-flop': One-bit memory unit. They are designed as logical circuits with multiple inputs and a single output.



$Q(t^+) = f(Q(t), I_0, I_1, \ldots, I_{n-1})$

Q(t): Current value

$Q(t^+)$: Next value

Q output shows the current value of the flip-flop (0,1). This value is the state of the flip-flop.

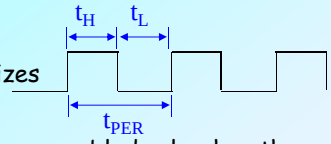
The next value of the output Q (denoted by $Q(t^+)$ or Q^+) is a function of the current state (denoted by Q(t) or Q) and the current inputs.

Clock signal (denoted as CLK) determines the time when the next state function is evaluated and the output of the flip-flop changes its value.

The output of the flip-flop can only change when the clock signal is active (the definition of being active will be described in the next slides).

If the clock signal is not active, flip-flop output will not change even if the input values change.

Clock Signal:



Clock signal is a periodic square wave that synchronizes the gates in the circuit.

Logic units with clock signal input (such as flip-flops) are enabled only when the clock signal is active. If clock signal is not active, they preserve their state.

There are two types of units in terms of clock signal activation

a) Level-triggered units b) Edge-triggered units

Level-triggered units use a level of the clock signal (1 in positive logic) as active.

These units activate and change their outputs when the clock signal is at **high** level.

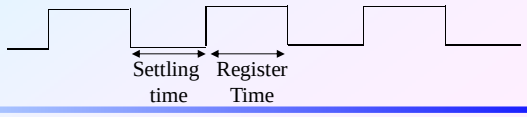
They preserve their state when the clock signal is at **low** level.

When the clock signal is at high level ("1"), the inputs should not change as they are processed.

Otherwise the output of the sequential circuit is undetermined (random).

This time is called **register time**.

The inputs can change when the clock signal is 0. This time is called **settling time**.



Edge-triggered units:

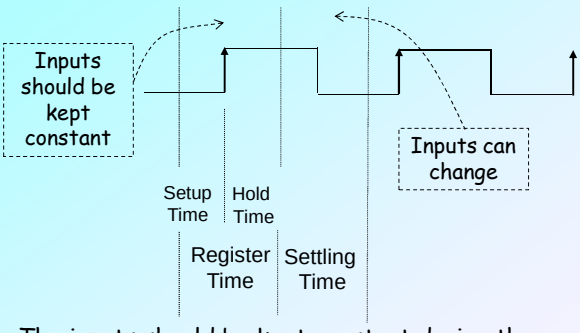
These units use an edge (rising edge in positive logic) of the clock signal as active.

Positive edge-triggered units use 0→1 transition of the clock signal (rising edge) to change their state and output. At other times, they preserve their state.

In negative logic, all transactions happen at 1→0 transition (falling edge).

As the inputs are used (processed) during 0→1 transition, inputs should be kept constant for certain time before and after the transition.

Otherwise, the output of the sequential unit is undetermined (random).



T he register time is the addition of the setup and hold times.

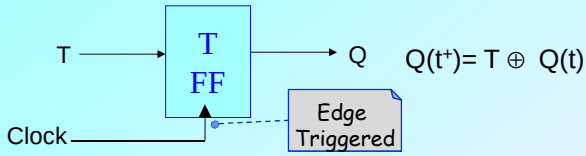
Setup time is the minimum amount of time the data signal should be held steady **before** the clock transition.

Hold time is the minimum amount of time the data signal should be held steady **after** the clock transition.

The inputs should be kept constant during the register time so that the sequential circuit works correctly.

Example: Edge-triggered Toggle (T) Flip-Flop

Before detailed explanation of flip-flops, a T flip-flop is used as an example.



$Q(t^+) = T \oplus Q(t)$

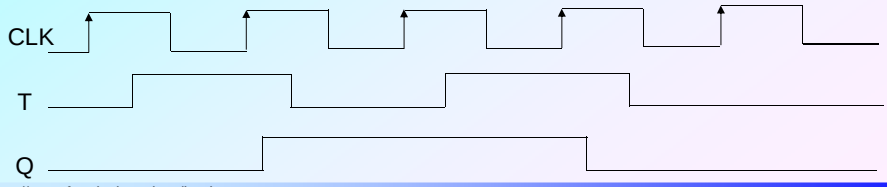
The next output (state) of the T flip-flop $Q(t^+)$ is equal to its current output XOR its input (T).

According to this, the output of the flip-flop does not change when T=0 as:

$0 \oplus x = x$

The output of the flip-flop is complemented when T=1 as:

$1 \oplus x = x'$



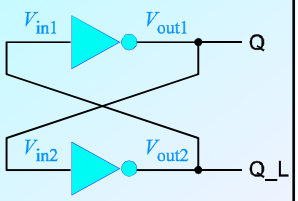
Bistable (Two stable states) Circuit

A bistabe circuit can be implemented with two inverters.

The inputs of the gates are connected to the outputs (feedback) of each other.

This bistable circuit has no external inputs.

This circuit will always be in one of the **two** possible stable states.



1. If the output of the inverter at the top is V_{out1} (Q) = 0, the input of the inverter at the bottom becomes V_{in2} = 0, and its output becomes V_{out2} (Q_L) = 1.
   This is a stable state as this state requires Q = 0 and V_{in1} = 1.

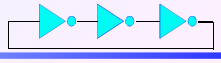
   State 1: $V_{in1} = 1$, $V_{out1} = V_{in2} = 0$, $V_{out2} = 1$

2. If the output of the inverter at the top is V_{out1} (Q) = 1, the input of the inverter at the bottom becomes V_{in2} = 1, and its output becomes (Q_L) = 0.
   This is also a stable state as it requires V_{in1} = 0 and Q = 1.

   State 2: $V_{in1} = 0$, $V_{out1} = V_{in2} = 1$, $V_{out2} = 0$

This circuit has two stable states: Q=0 and Q=1

It is impossible to control (change) the state of the circuit as it has no inputs.
When this circuit is turned on, it takes a random state.

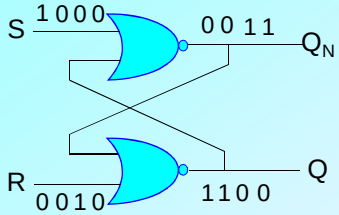
Examples of **unstable** circuits:  or 

S-R (Set-Reset) Latch

S-R Latch is a one-bit memory built with two NAND or two NOR gates.

All other latches and flip-flops can be built upon this fundamental memory unit with certain extensions.

S-R latch with NOR gates:



S: Set
R: Reset
Q: Output (State)
Q_N: Complemented output (Q')

Recall: When an input of a NOR gate is "1", the output will be "0" regardless of the other input.

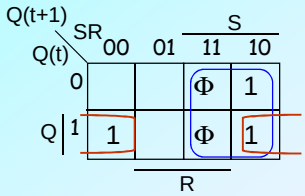
S	R	Q	Q_N	
1	0	1	0	
0	0	1	0	After S=1, R=0
0	1	0	1	
0	0	0	1	After S=0, R=1
1	1	0	0	Forbidden input

- S input is used to write (store) "1" to the latch, R input is used to write "0".
- If both inputs are "0", SR latch preserves its state.
- Both inputs should not be "1" at the same time.

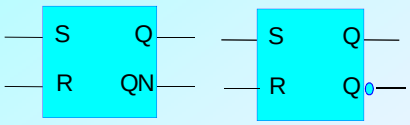
Next value of the output (state) Q(t+1) depends on the inputs and current output Q(t).

The truth table and the logical expression of the SR latch can be expressed as follows:

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

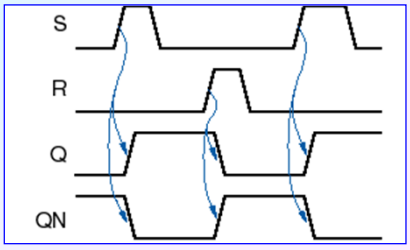


$Q(t+1) = S + Q(t)R', \quad SR=0$



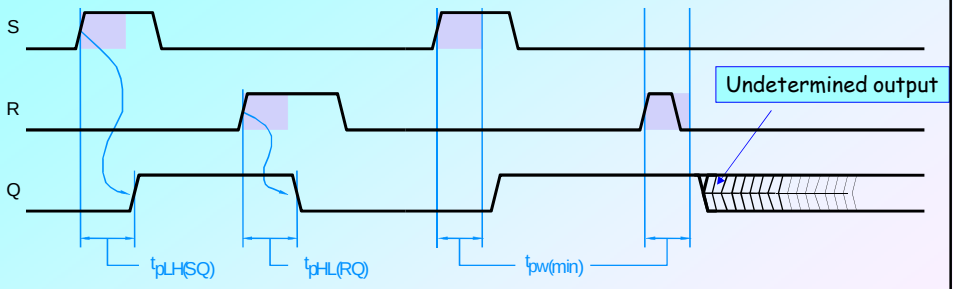
Latches are not triggered by a clock signal.

Flip-flops are clock-triggered memory units.



Because of the propagation delay, the changes in the S and R inputs will not affect the output simultaneously.

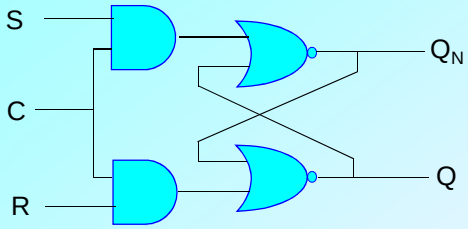
During the delay, the inputs should be kept constant. Otherwise, the value of the output is undetermined (random).



$t_{pLH(SQ)}$: Delay in the 0-1 transition of the output when S changes.
$t_{pHL(RQ)}$: Delay in the 1-0 transition of the output when R changes.
$t_{pW(min)}$: Minimum duration when the inputs should be kept constant.

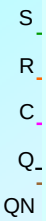
S-R Latch with Enable Input

AND gates are connected to S and R inputs so that the latch is active only when it is enabled.

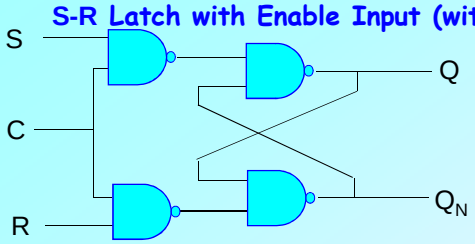


S: Set
R: Reset
Q: Output (State)
Q_N: Complemented output (Q')
C: Enable input

The value of the latch can only be changed when C=1. When C=0, the value of the latch is preserved regardless of the inputs.



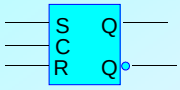
S-R Latch with Enable Input (with only NAND Gates)

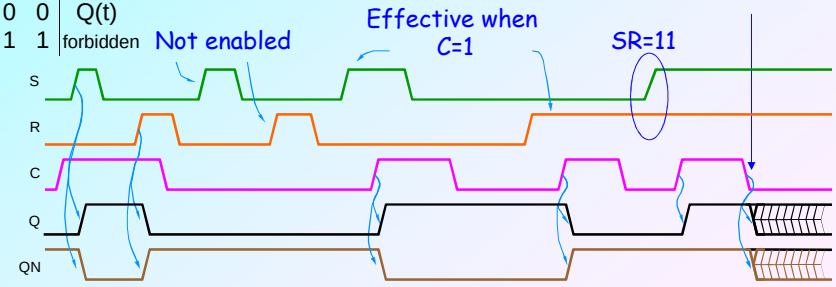


S-R Latch implemented with NOR and AND gates (slide 6.16) can also be implemented using only NAND gates.

If the forbidden input (SR=11) is applied to the latch, both Q and Q' outputs will be 1.
If the latch is disabled in this state, the value in the latch is undetermined.

C	S	R	Q(t+1)
0	X	X	Q(t)
1	1	0	1
1	0	1	0
1	0	0	Q(t)
1	1	1	forbidden





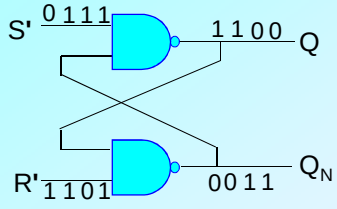
Difference between Latch and Flip-flop:

Latches are not triggered by clock signal. The value of the latch can be changed whenever it is enabled.

Memory units that are triggered by clock signal are called **flip-flop**.

S-R Latch with NAND Gates (S'-R' Latch)

S-R latches can be implemented using NAND gates instead of NOR. These units are called **S'-R'** latch.



S': Complement of Set
R': Complement of Reset
Q: Output (State)
Q_N: Complemented output (Q')

This latch is different than the one in slide 6.17.

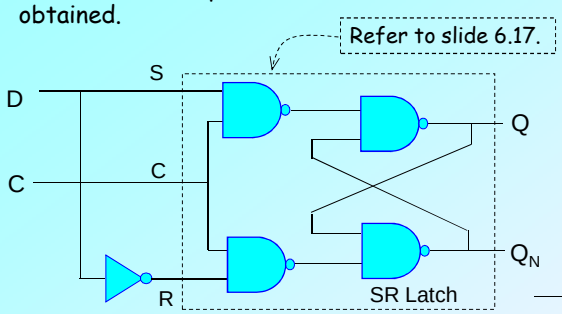
S'	R'	Q	Q_N	
0	1	1	0	
1	1	1	0	After S'=0, R'=1
1	0	0	1	
1	1	0	1	After S'=1, R'=0
0	0	1	1	Forbidden inputs

Recall: If an input of a NAND gate is "0", the output will be "1" regardless of the other input.

Delay (D) Latch

Other types of latches can be built upon S-R latch with some additions.

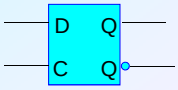
If the S and R inputs of the SR latch are connected with an inverter, D-latch is obtained.



C=1	D	S	R	$Q(t^+)$
	0	0	1	0 (Reset)
	1	1	0	1 (Set)

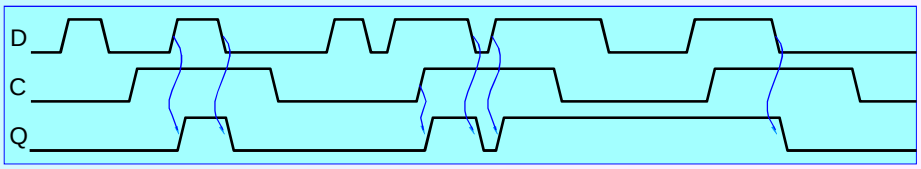
If C=1, the value of D input is written to the latch.

If C=0, latch preservers its previous value.



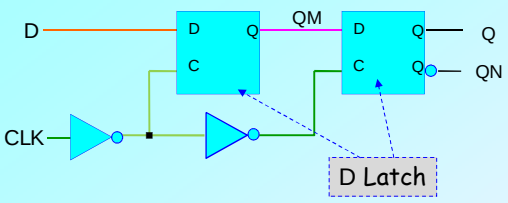
C	D	$Q(t^+)$	$Q_N(t^+)$
1	0	0	1
1	1	1	0
0	X	Q(t)	$Q_N(t)$

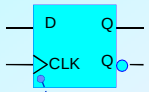
Q(t+1)=D



Rising Edge Triggered D Flip-flop

Latches change their value (depending on their input) when they are enabled.
Flip-flops on the other hand, change their value when the clock signal is active.





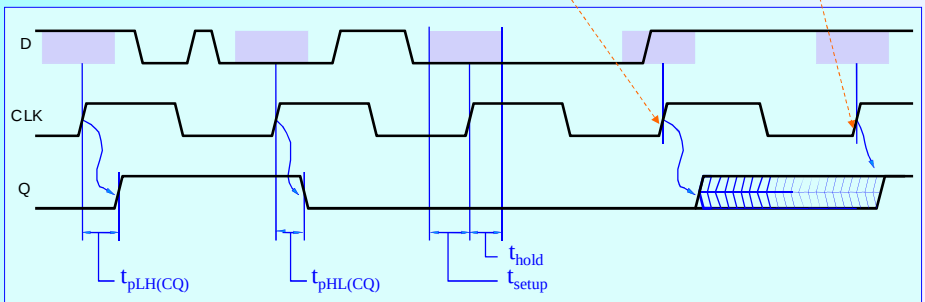
Edge triggered

D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	prev Q	prev QN
x	1	prev Q	prev QN

Timing properties of the rising edge triggered D flip-flop

Output is undetermined as the inputs are not kept constant during the *setup time*.

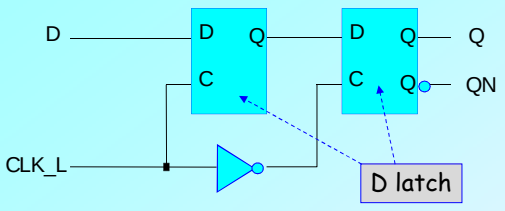
Output gets a valid value (1 in this example).

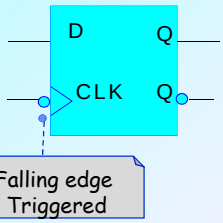


$t_{pLH(CQ)}$: Delay between the active edge of clock and 0-1 transition of output.
$t_{pHL(CQ)}$: Delay between the active edge of clock and 1-0 transition of output.
t_{setup} : Time before active edge where inputs should be kept constant.
t_{hold} : Time after active edge where inputs should be kept constant.

Falling Edge Triggered D Flip-flop

Data is written to D flip-flop during the falling edge of the clock signal.



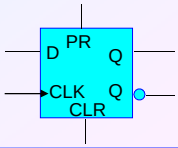


D	CLK_L	Q	QN
0	↓	0	1
1	↓	1	0
x	0	prev Q	prev QN
x	1	prev Q	prev QN

In flip-flops (especially to write an initial value), there may be asynchronous (independent of clock signal) inputs.

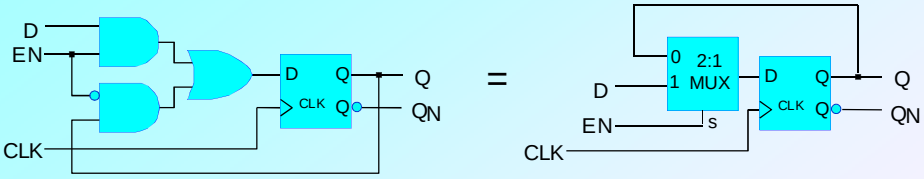
To write 1 to the flip-flop PR (*Preset*), to write 0 CLR (*Clear*) inputs can be used.

Asynchronous inputs change the value of flip-flop even if the clock signal is not active.

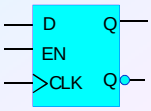


Edge Triggered D Flip-flop with Enable Input

Flip-flops may also have an enable input. If the enable input is active, the value of the flip-flop can be changed. Otherwise, it preserves its value regardless of its inputs.



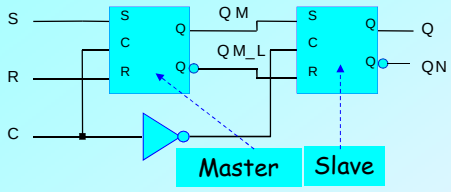
D	EN	CLK	Q	QN
0	1	↑	0	1
1	1	↑	1	0
x	0	↑	prev Q	prev QN
x	x	0	prev Q	prev QN
x	x	1	prev Q	prev QN



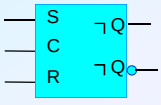
Master/Slave SR Flip-flop

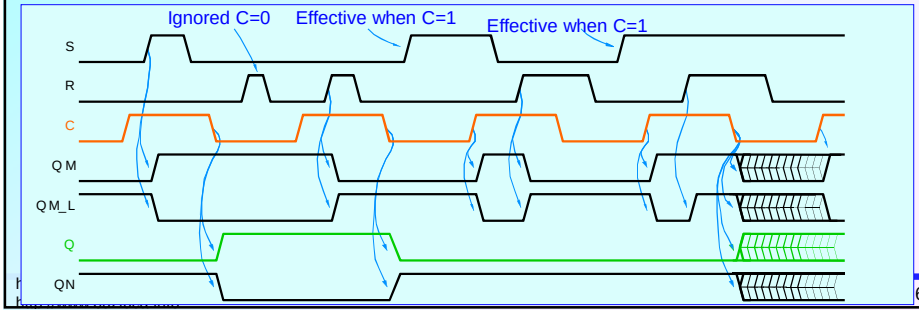
Master/slave flip-flops are pulse triggered units. The value (output) of this type of flip-flops changes only at the falling edge of the clock signal.

However, next value of the flip-flop can be determined while the clock signal is 1.



S	R	C	Q	QN
x	x	0	prev Q	prev QN
0	0	↓	prev Q	prev QN
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	undefined	undefined



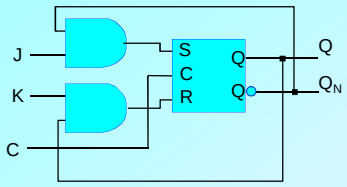


JK Latch

The problem of forbidden inputs of the SR latches (S=1,R=1) is solved in JK latches.

These units function similar to SR latches. Input J sets and input K resets the output.

When both inputs are "1" (J=1, K=1), the value (output) of the latch is complemented.

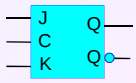


C=1

J	K	Q(t)	S	R	Q(t+1)	Operation
0	0	0	0	0	0	Don't change
0	0	1	0	0	1	
0	1	0	0	0	0	Reset
0	1	1	0	1	0	
1	0	0	1	0	1	Set
1	0	1	0	0	1	
1	1	0	1	0	1	Complement
1	1	1	0	1	0	

J	K	C	Q	QN
x	x	0	prev Q	prev QN
0	0	1	prev Q	prev QN
0	1	1	0	1
1	0	1	1	0
1	1	1	prev QN	prev Q

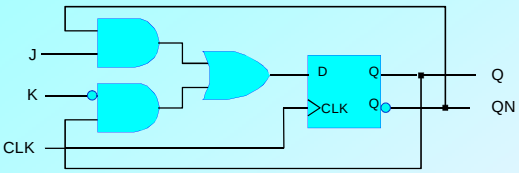
$$Q(t+1)=J \bullet Q(t)' + K' \bullet Q(t)$$



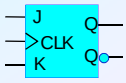
Edge Triggered JK Flip-flop

An edge-triggered JK flip-flop can be implemented using an edge triggered D flip-flop and logical gates.

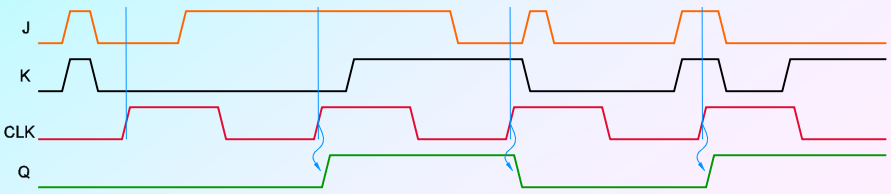
In this unit, the J and K inputs are processed only at the active edge of the clock signal.



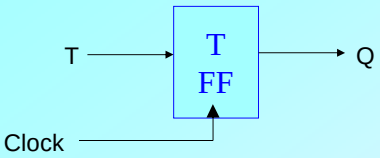
J	K	CLK	Q	QN
x	x	0	prev Q	prev QN
x	x	1	prev Q	prev QN
0	0	↑	prev Q	prev QN
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	prev QN	prev Q



$$Q(t+1)=J \bullet Q(t)' + K' \bullet Q(t)$$



Edge Triggered Toggle (T) Flip-flop



$Q(t+1) = T \oplus Q(t)$

The output (value) of a T flip-flop is determined by XOR operation of the input (T) and the current value.

Therefore, if the input is 0 (T=0), the value of the flip-flop is preserved as:

$0 \oplus x = x$

If the input is 1 (T=1), the value of the flip-flop is complemented (toggled) as:

$1 \oplus x = x'$