SEQUENTIAL CIRCUITS

- In the first part of the course, **combinational circuits** were covered.

  The outputs of the combinational circuits depend only on the **current** inputs.

  Combinational circuit: Output = G (Input)

- In **sequential circuits**, the outputs depend both on the inputs and the “state” of the circuit.

  Sequential circuit: Output = G (Input , Current State)

  Next State = H (Input , Current State)

Memory units are required to store (remember) the state of the circuit.

For example, vending machines keep track of (remember) the coins that were inserted into the machine.

With each coin, the state of the machine (total amount of inserted coins) is updated.

Types of sequential circuits:

There are two types of sequential circuits :

A) 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.

B) 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.

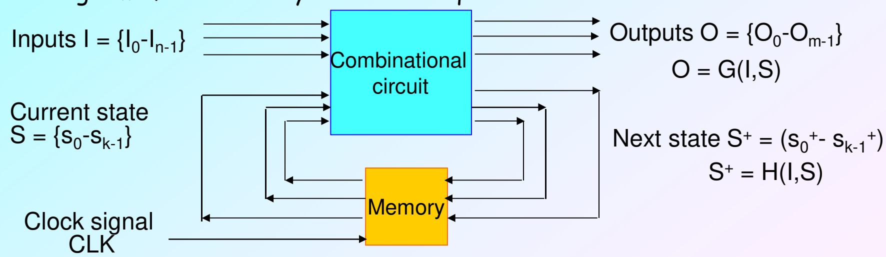
- When the machine is started, it is in a certain state (initial state: s_0).
- An output is produced depending on the inputs and the current state. $O = f\ (I,S)$
- Transition into a new state happens depending on the input and the current state.

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:

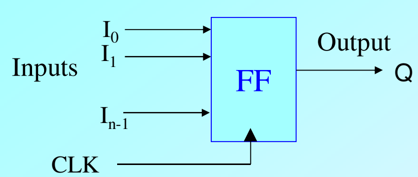


We will see details of the FSM and clocked synchronous sequential circuits (in chapters 7 and 8) after we cover memory units.

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.

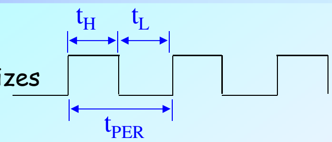
The next value of the output Q (denoted by $Q(t+1)$, $Q(t^+)$ or Q^+) is a function of the current state (denoted by $Q(t)$ or Q) and the current inputs.

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.

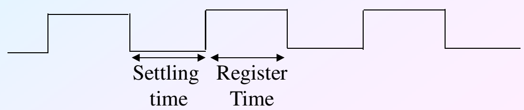
These units activate and change their outputs when the clock signal is at **high** level.

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**.

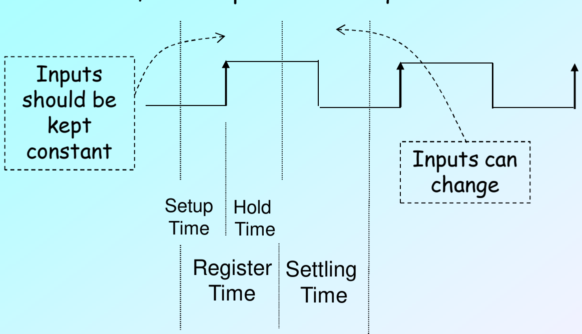


Edge-triggered units:

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.

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).



The **register time** is the sum 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.

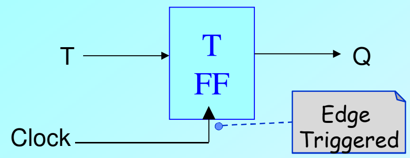
Hold time is the minimum amount of time the data signal should be held steady **after** the clock transition.

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

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

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

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

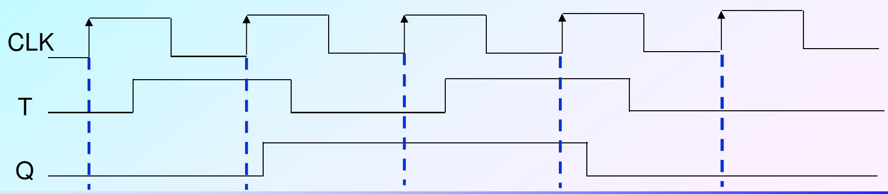


$Q(t^+) = T \oplus Q(t)$

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

According to this, the output of the flip-flop does not change when T=0 because $0 \oplus x = x$.

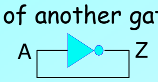
The output of the flip-flop is inverted (complemented) when T=1 because $1 \oplus x = \bar{x}$.



Feedback Connections

In order to construct a circuit that has memory, we must introduce feedback into the circuit.

By **feedback**, we mean that the output of one of the gates is connected back into the input of another gate in the circuit so as to form a closed loop.

Example: 

If, at some instant of time, the inverter input A is "0", this "0" will propagate through the inverter and cause the output Z to become "1".

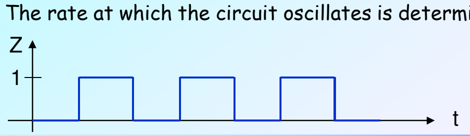
This 1 is fed back into the input, so after the propagation delay, the inverter output Z will become "0".

When this "0" feeds back into the input A, the output Z will again switch to "1", and so forth.

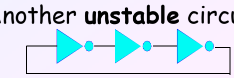
The inverter output Z will continue to oscillate back and forth between "0" and "1" as shown in the figure below, and it will never reach a stable condition (unstable).

This **unstable** circuit cannot be used as a memory.

The rate at which the circuit oscillates is determined by the propagation delay in the inverter.



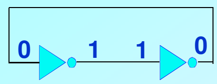
Another **unstable** circuit:



Feedback Loop with Two Inverters (Bistable Circuit)

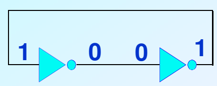
Next, consider a feedback loop which has two inverters in it, shown below.
In this case, the network has two stable conditions (bistable), often referred to as stable states.

Stable state 1:



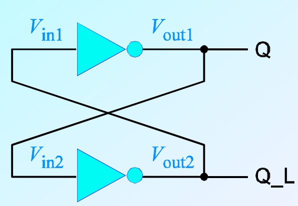
If the input to the first inverter is 0, its output will be 1.
Then, the input to the second inverter will be 1, and its output will be 0.
This 0 will feed back into the first inverter, but since this input is already 0, no changes will occur.
The circuit is then in a stable state.

Stable state 2:



A second stable state of the circuit occurs when the input to the first inverter is 1 and the input to the second inverter is 0.

Bistable (Two stable states) Circuit (cont'd)



The bistable circuit in slide 6.9 can be also drawn as shown at right.
Remember, this circuit will always be in one of the **two** possible **stable** states.

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

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

This circuit has two stable states: Q = 0 and Q = 1,
Q_L is complement of Q $Q_L = \bar{Q}$.
These are necessary properties for memory units.

But this circuit has no external inputs.

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.

Therefore, it cannot be used as a memory unit.

A memory unit must have

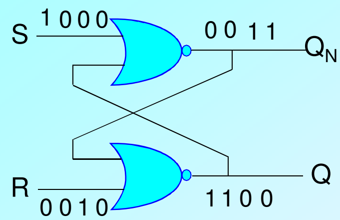
1. two stable states,
2. control input(s), which can be used to change or preserve the state of the unit.

S-R (Set-Reset) Latch

S-R Latch is a one-bit memory built by introducing feedback into two NAND or two NOR gates.

All other latches and flip-flops can be built from 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 one input of a NOR gate is "1", the output will be "0" regardless of the other input.

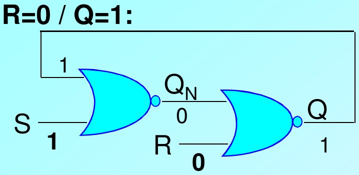
S	R	Q	Q_N	
1	0	1	0	
0	0	1	0	After S=1, R=0
0	1	0	1	
0	0	0	1	After S=0, R=1
1	1	0	0	Forbidden input

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

State Changes of the S-R (Set-Reset) Latch

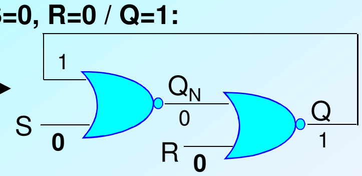
To show how states of an S-R latch change we can draw the circuits as seen below.

S=1, R=0 / Q=1:



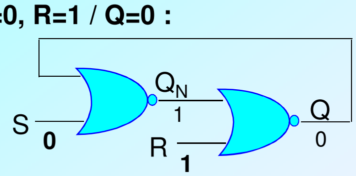
If S = 1, R = 0, Q_N will be 0. Since both inputs of the second gate are 0, Q will become 1.

S=0, R=0 / Q=1:



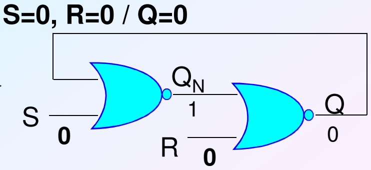
If S changed to 0, the latch will remain in the present state (1) because Q = 1 is fed back into the first gate will cause Q_N to remain 0, as shown above.

S=0, R=1 / Q=0 :



If we now change R to 1, Q will become 0 and Q_N will then change back to 1.

S=0, R=0 / Q=0



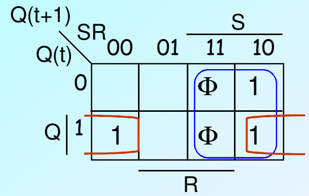
If we then change R back to 0, the latch remains in the present state (0).

Truth table and the characteristic equation for the S-R latch:

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

Truth table:

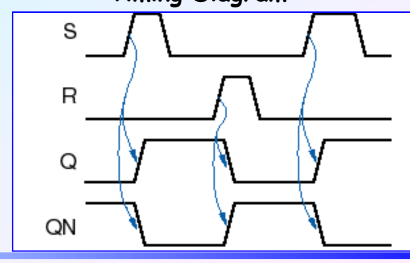
Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	forbidden(Φ)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	forbidden(Φ)



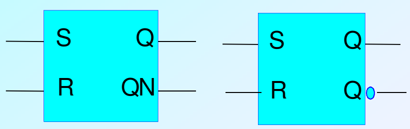
Characteristic Equation:

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

Timing Diagram:



Block Diagrams for the S-R Latch:



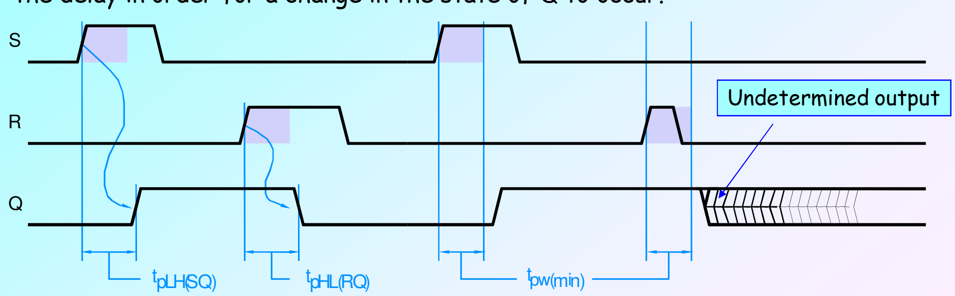
Timing Properties of the S-R latch:

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

During the delay, the inputs should be kept constant.

Otherwise, the value of the output is undetermined (random).

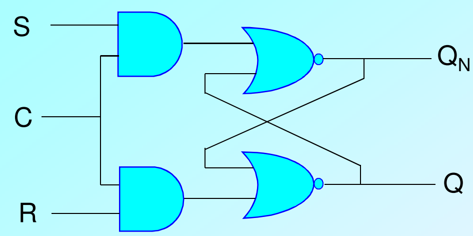
The duration of the S (or R) input pulse must normally be at least as great as the delay in order for a change in the state of Q to occur.



$t_{pLH(SQ)}$: 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 for which the inputs should be kept constant.

S-R Latch with Enable Input

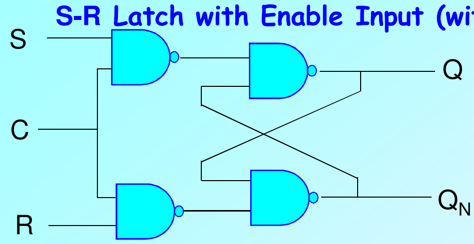
AND gates are connected to S and R inputs so that the latch is active only when it is enabled.



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.

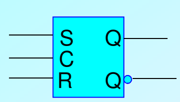
S-R Latch with Enable Input (with only NAND Gates)

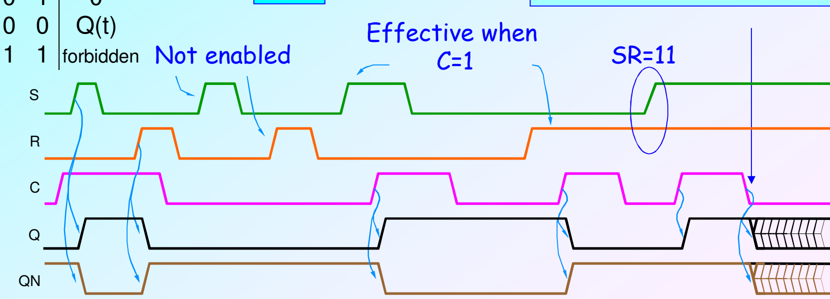


The S-R Latch implemented with NOR and AND gates (slide 6.15) can also be implemented using only NAND gates.

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 a Latch and a Flip-Flop:

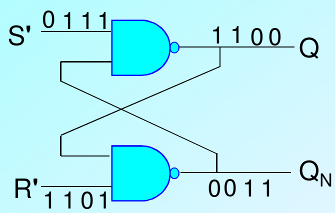
A latches is not triggered by a clock signal. The value of the latch can be changed whenever the latch is enabled.

A flip-flop is a memory units that is triggered by a clock signal.

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

An S-R latch can be implemented using NAND gates instead of NOR.

These results in an **S'-R'** latch.



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

This latch is different from the latches in slides 6.15 and 6.16.

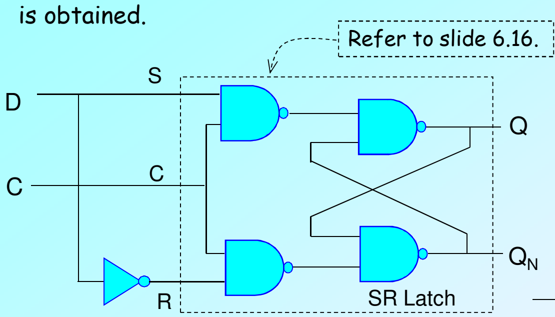
S'	R'	Q	Q_N	
0	1	1	0	
1	1	1	0	After S'=0, R'=1
1	0	0	1	
1	1	0	1	After S'=1, R'=0
0	0	1	1	Forbidden inputs

Recall: If one of the inputs 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 from the S-R latch by adding external gates.

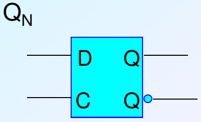
If an inverter is added to an S-R latch to make R the complement of S, the D-latch is obtained.



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

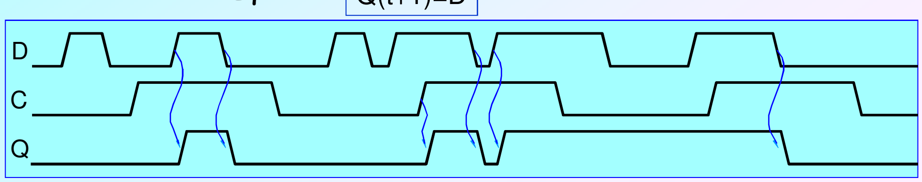
If C=1, the value of input D is written to the latch.

If C=0, the latch preserves its previous value.



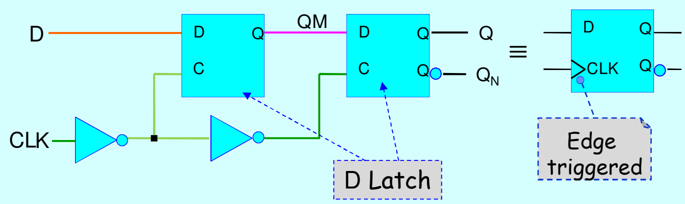
C	D	$Q(t^+)$	$Q_N(t^+)$
1	0	0	1
1	1	1	0
0	X	Q(t)	$Q_N(t)$

Characteristic Equation: $\boxed{Q(t+1)=D}$

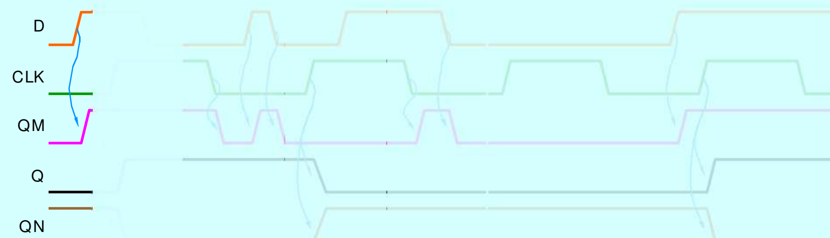


Rising Edge Triggered D Flip-Flop

A latch changes its value (depending on its input) when it is enabled.
A flip-flop, on the other hand, changes its value when the clock signal is active.



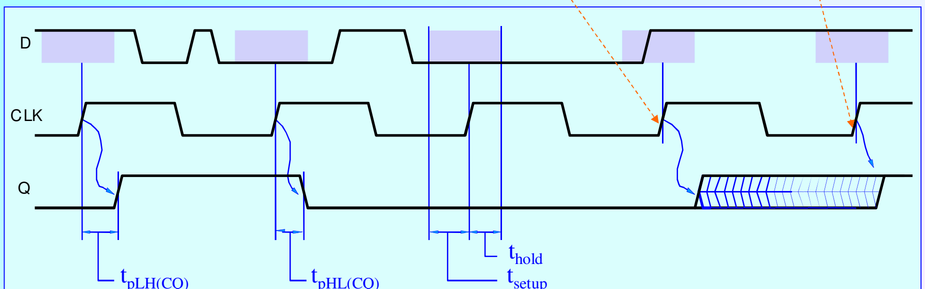
D	CLK	Q(t+1)	$Q_N(t+1)$
0	↑	0	1
1	↑	1	0
x	0	Q(t)	$Q_N(t)$
x	1	Q(t)	$Q_N(t)$



Timing properties of the rising edge triggered D flip-flop

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

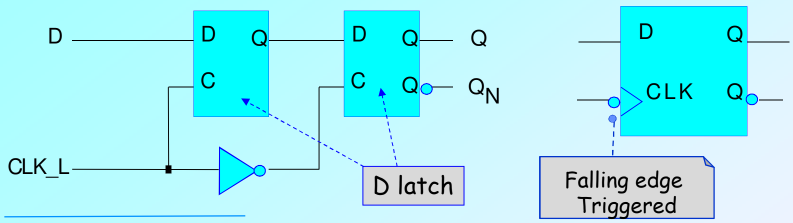
The output gets a valid value ("1", in this example).



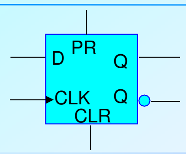
$t_{pLH(CQ)}$: Delay between the active edge of clock and 0-1 transition of the output.
$t_{pHL(CQ)}$: Delay between the active edge of clock and 1-0 transition of the output.
t_{setup} : Time before active edge when inputs should be kept constant.
t_{hold} : Time after active edge when inputs should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

Falling Edge Triggered D Flip-Flop

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



D	CLK_L	Q(t+1)	$Q_N(t+1)$
0	↓	0	1
1	↓	1	0
x	0	Q(t)	$Q_N(t)$
x	1	Q(t)	$Q_N(t)$



Clocked integrated circuit flip-flops often have additional inputs which can be used to set the flip-flop to an initial state independent of the clock (asynchronous).

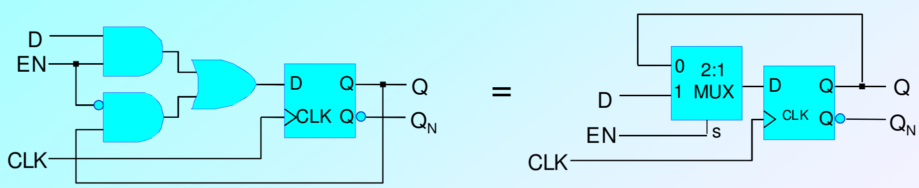
To write "1" to the flip-flop PR (*Preset*), to write "0" CLR (*Clear*) inputs can be used.

A "1" applied to the clear (CLR) input will reset the flip-flop to Q = 0, and a "1" applied to the preset (PR) input will set the flip-flop to Q = 1.

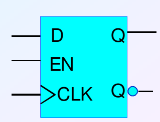
These asynchronous inputs override the clock and D inputs. That is, a "1" applied to the clear input will reset the flip-flop regardless of the values of D and the clock.

Edge Triggered D Flip-flop with Enable Input

A flip-flop may also have an enable input (EN).
If the enable input (EN) is active, the value of the flip-flop can be changed.
Otherwise, the flip-flop preserves its value regardless of its inputs.

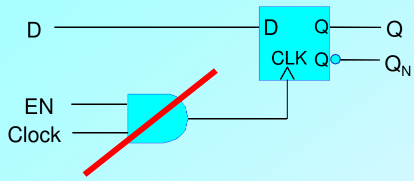


D	EN	CLK	Q(t+1)	$Q_N(t+1)$
0	1	↑	0	1
1	1	↑	1	0
x	0	↑	Q(t)	$Q_N(t)$
x	x	0	Q(t)	$Q_N(t)$
x	x	1	Q(t)	$Q_N(t)$



Edge Triggered D Flip-flop with Enable Input (cont'd)

Can we design a simpler circuit by connecting an AND gate to the CLK input of the D flip-flop?



This is **not a proper solution**.

This circuit has some problems.

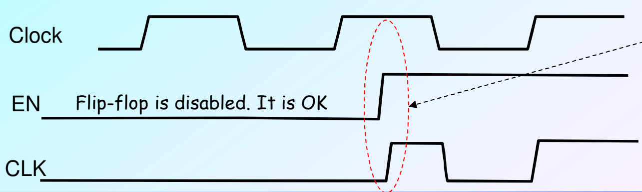
1. It is not a good idea to increase the delay at the CLK input.

   It can cause synchronization problems.

2. If Clock=1 and EN=0 then CLK=0 and the flip-flop is disabled (it's OK).

But if EN becomes 1 while clock=1, flip-flop will get a rising edge on the CLK input and it will process its input D.

However the real clock signal is 1; There is not a 0-1 transition on the clock input.



Problem:
There is not a real edge but the FF gets an rising edge.

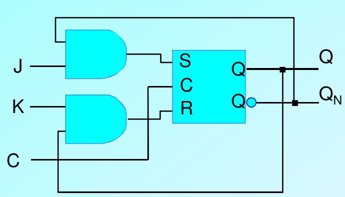
J-K Latch

The J-K latch combines the features of the S-R and T latches.

A "1" input applied to J or K alone acts exactly like an S or R input, respectively.

That is, if J = 1, the latch output is set to Q = 1; and if K = 1, the latch output is reset to Q = 0.

Unlike the S-R latch, it is permissible to apply a "1" input simultaneously to J and K, in which case the latch changes state (toggles) just like a T latch.



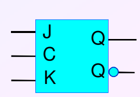
J	K	C	Q	QN
x	x	0	prev Q	prev QN
0	0	1	prev Q	prev QN
0	1	1	0	1
1	0	1	1	0
1	1	1	prev QN	prev Q

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	

Characteristic Equation:

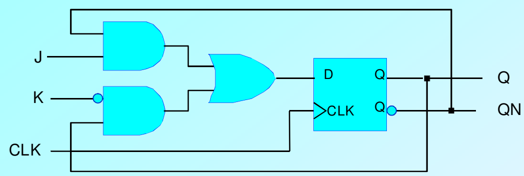
$$\boxed{Q(t+1)=J \cdot Q(t)' + K' \cdot Q(t)}$$



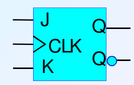
Edge Triggered J-K Flip-Flop

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

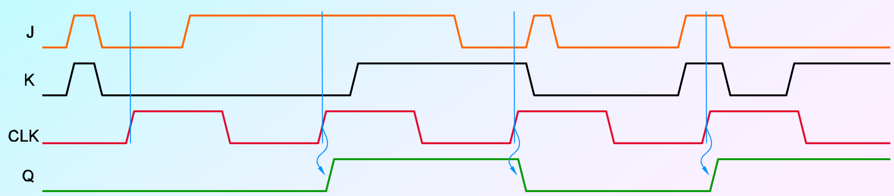
In this unit, the J and K inputs are processed only at the active edge of the clock signal.



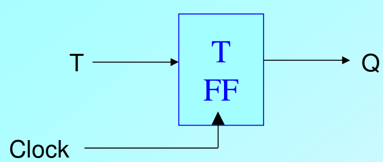
J	K	CLK	Q	QN
x	x	0	prev Q	prev QN
x	x	1	prev Q	prev QN
0	0	↑	prev Q	prev QN
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	prevQN	prev Q



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



Edge Triggered Toggle (T) Flip-Flop



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

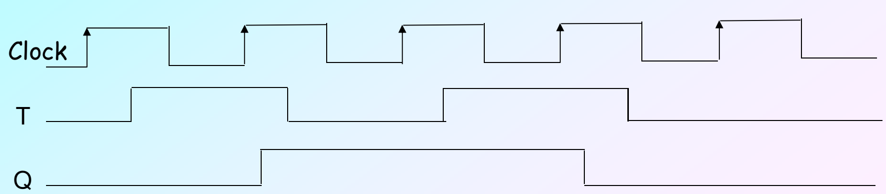
The output (value) of a T flip-flop is determined by applying the XOR operation to the input (T) and the current value.

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:

$1 \oplus x = x'$



Characteristic Equations of latches and flip-flops:

The functional behavior of a latch or flip-flop can be described by a **characteristic equation** that specifies the next state of the flip-flop (or latch) as a function of its inputs and current state.

Characteristic equations for the flip-flops:

S-R FF: $Q(t+1) = S + R' \cdot Q(t)$, $SR=0$

J-K FF: $Q(t+1) = J \cdot Q(t)' + K' \cdot Q(t)$

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

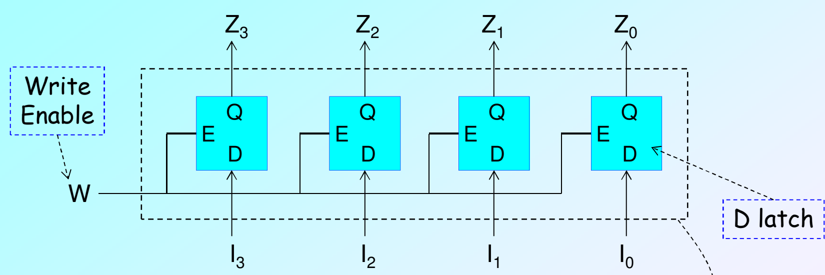
T FF : $Q(t+1) = T \oplus Q(t)$

Registers

Registers are memory units that can hold n-bit binary numbers.

Example: 4-bit parallel register

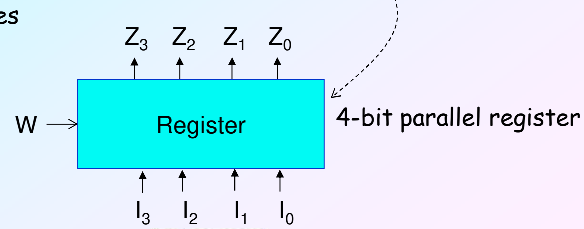
D latches are used.



If W=0, the register preserves its current state (contents).

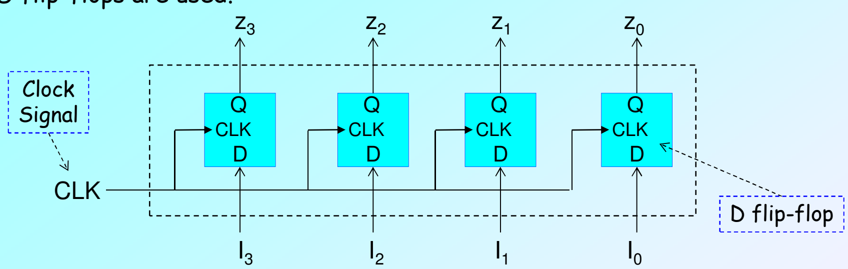
If W=1, the data at the input is written to the register.

$W = 0,\ Z(t+1) = Z(t)$
$W = 1,\ Z(t+1) = I(t)$



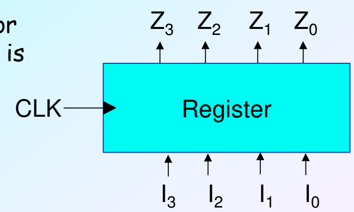
Example: 4-bit parallel register **with clock signal**

D flip-flops are used.



If the clock signal is active (for example rising edge), the data is written to the register.

Clock-triggered 4-bit parallel register:

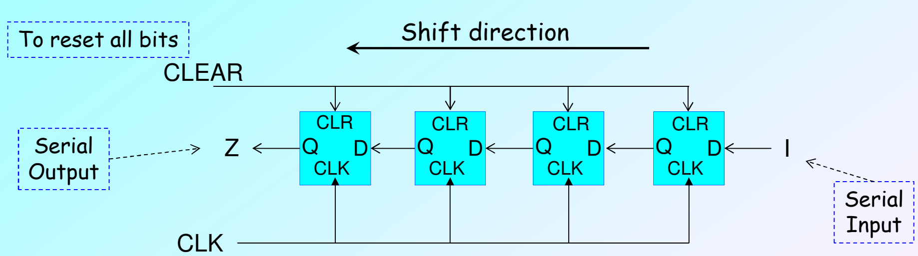


Note: If D flip-flops with enable (E) inputs are used then the register will contain both the clock signal (CLK) and write-enable (W) input.

Serial Shift Registers

Serial shift registers can hold and shift binary numbers to left or right one bit at each active transition of the clock signal.

Example: 4-bit serial left shift register



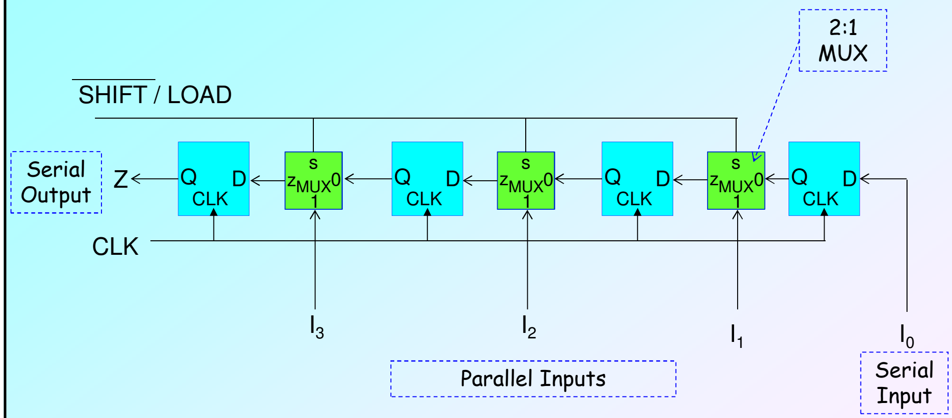
The data is shift left one bit at each active transition of the clock signal.

By changing the order of the flip-flops a right-shift register can be implemented.

In this design it is not possible to load an initial value to the register (except zero) in parallel.

Example: 4-bit serial left shift register with parallel load

In this design we can load an initial value to the register in parallel.



$\overline{\text{SHIFT}}$ / LOAD = 0 : shift

$\overline{\text{SHIFT}}$ / LOAD = 1 : load