UNIT - 6

UNIT-6

Combinational Logic Circuits - III

Introduction :

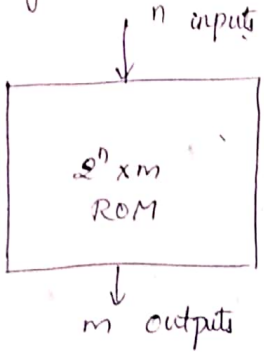
There are many applications for digital logic where the market is not great enough to develop a special-purpose MSI & LSI chip. This situation has led to the development of Programmable Logic Devices (PLDs) which can be easily configured by the individual user for specialised applications.

Basically, there are three types of PLDs.

→ Read Only Memory (ROM)
→ Programmable Logic Array (PLA)
→ Programmable Array Logic (PAL)

Read Only Memory (ROM) :

A read only memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package.

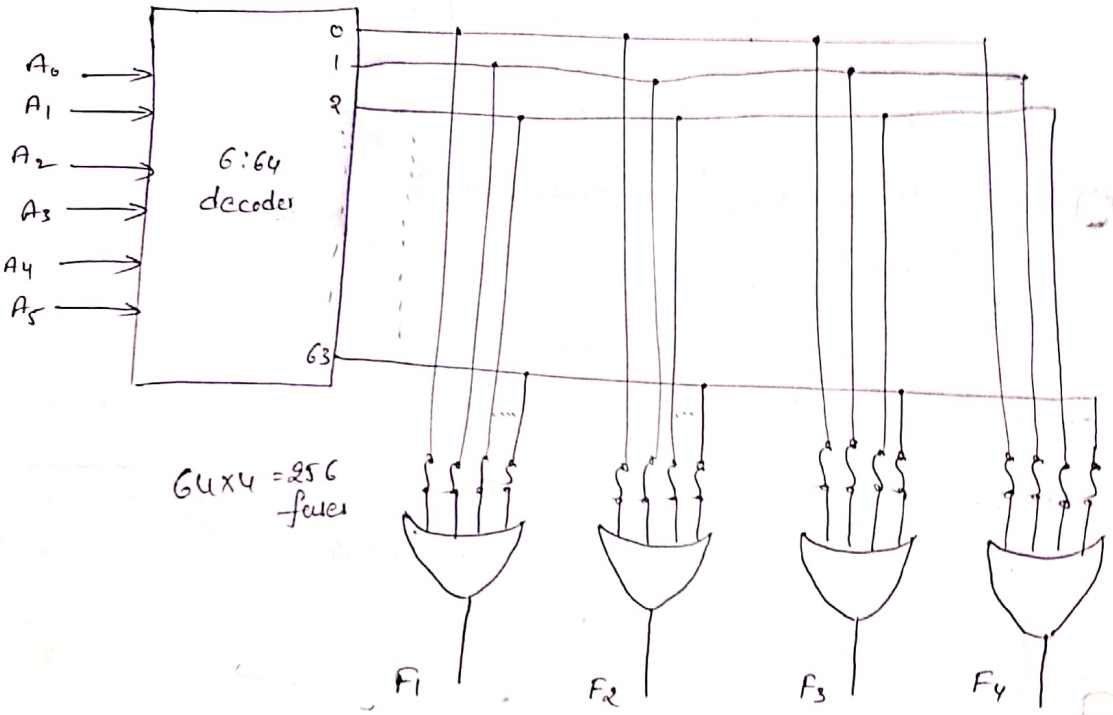


The ROM consists of 'n' input lines and 'm' output lines. Each combination of the input variables is called address. Each bit combination that comes out of the o/p lines is called word. The no. of distinct addresses possible with 'n' input variables is 2^n. An output word can be selected by a unique address, and since there are 2^n distinct addresses in a ROM, there are 2^n distinct words in the ROM.

64 x 4 ROM :

Let us consider 64 x 4 ROM consists of 4 o/p lines and particular word from 64 words and four o/p lines. Here 64 words are determined from the six i/p lines. There are 6 input lines in a 64 x 4 ROM because $2^6 = 64$.

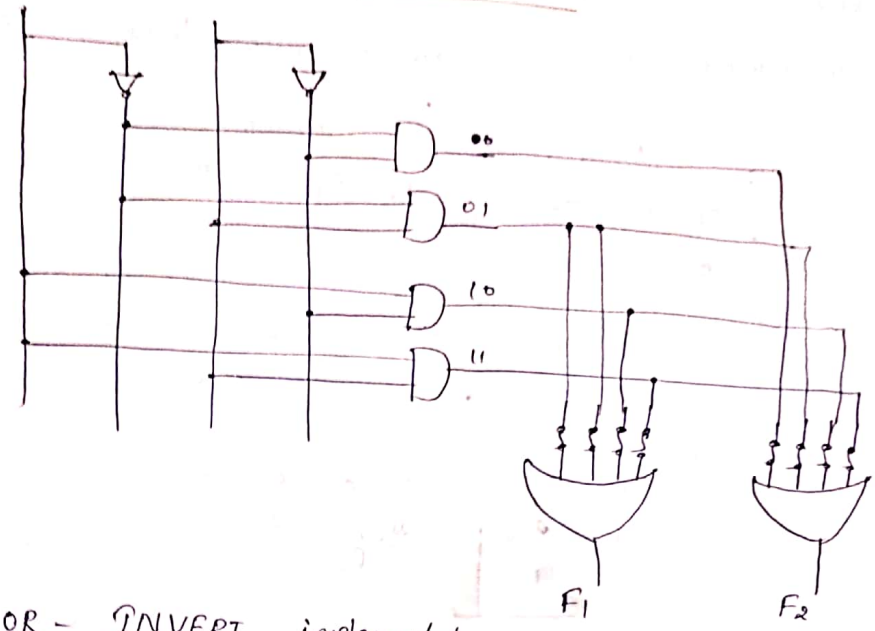
The internal logic construction of 64 x 4 ROM is shown below.



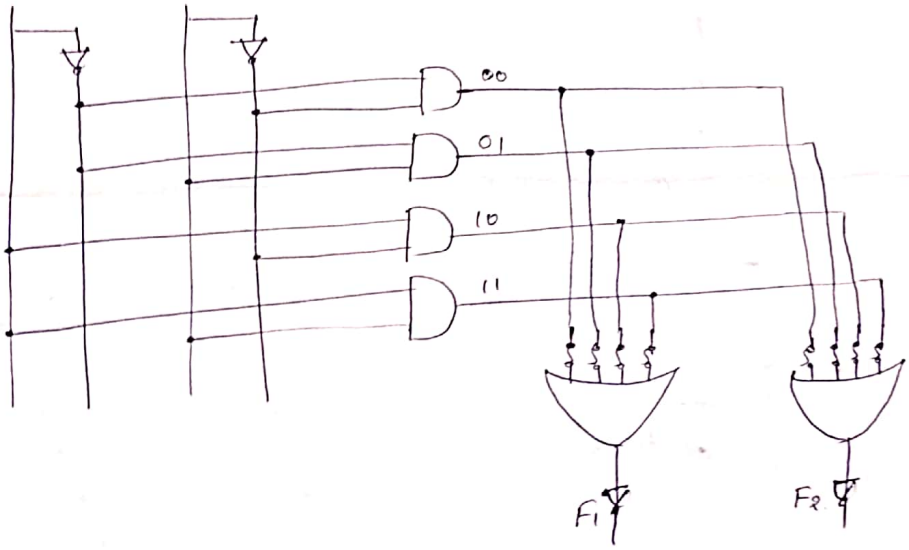
The figure shows 64 x 4 ROM, the six input variables are decoded in 64 lines by means of 64 AND gates & 6 inverters. Each o/p of the decoder represents one of the minterms of a function of six variables. The 64 outputs of the decoder are connected through fuses to each OR gate. Only four of these shown in figure.

The ROM is a two level implementation in Sum of minterms form. Here we can design AND - OR and AND - OR - Inverter implementation of ROM.

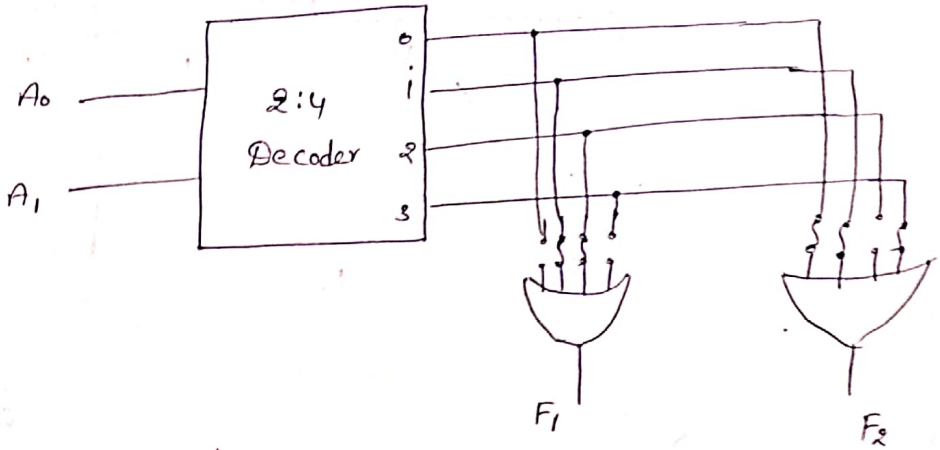
AND - OR implementation of 4x2 ROM :



AND - OR - INVERT implementation of 4x2 ROM :



Implement the $F_1(A_1, A_0) = \Sigma m(1, 2)$, $F_2(A_1, A_0) = \Sigma m(0, 1, 3)$ using ROM.



→ Design a Combinational Ckt using a ROM. The circuit accepts a binary number and generates its equivalent - Excess - 3 code.

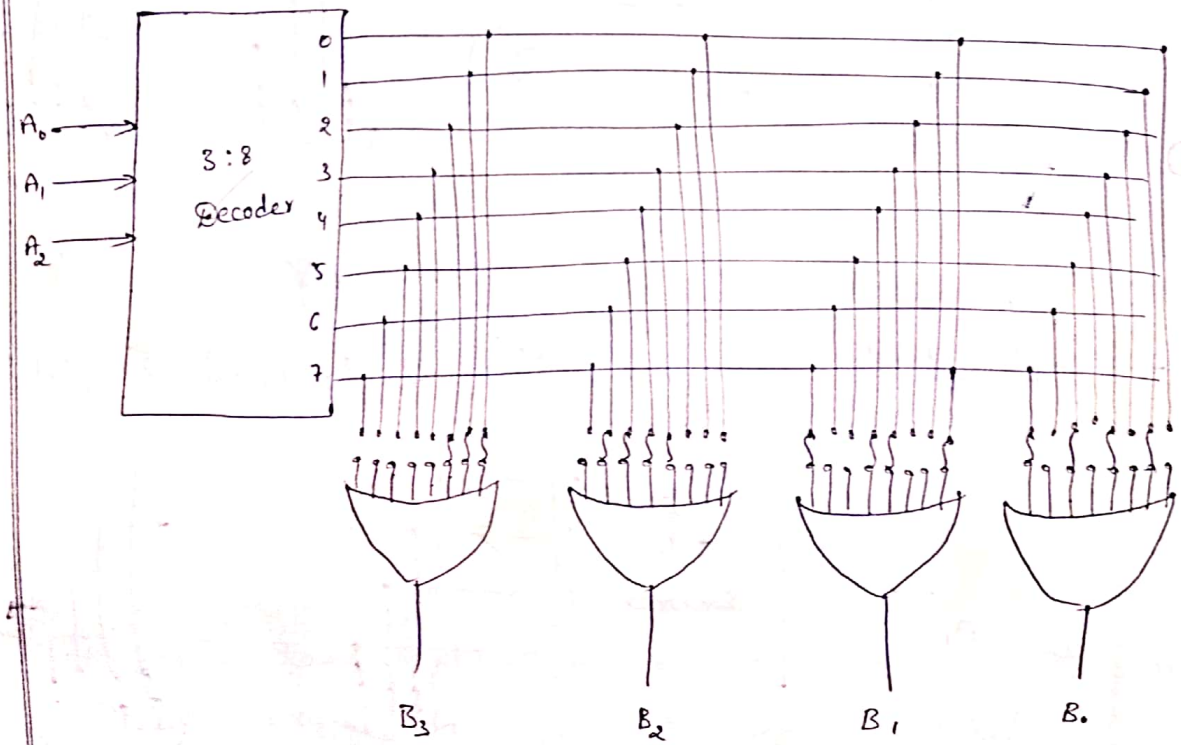
Inputs			Outputs			
A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	0	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

$B_0 = \Sigma m\ (0, 2, 4, 6)$

$B_1 = \Sigma m\ (0, 3, 4, 7)$

$B_2 = \Sigma m\ (1, 2, 3, 4)$

$B_3 = \Sigma m\ (5, 6, 7)$



Types of ROMs :

PROM (Programmable Read Only Memory) :

PROM allows user to store program. PROM use the fuses with material like nichrome and polycrystalline. The blowing of fuses according to the truth table is called programming of ROM. The user can program PROMs with special PROM programmer. The PROMs are one time programmable. Once programmed, the information stored is permanent.

EPROM (Erasable Programmable Read Only Memory) :

EPROMs use MOS circuitry. EPROMs can be programmed by the u with a special EPROM programmer. In this we can erase the stored data in the EPROMs by exposing the chip to ultraviolet light through its quartz window for 15 to 20 minutes.

In EPROMs, it is not possible to erase selective information; when erased, the entire information is lost. The chip can be reprogrammed.

EEPROM (Electrically Erasable Programmable Read Only Memory) :

Electrically erasable programmable ROMs also use MOS circuit very similar to that of EPROM. EEPROM allows selective erasing at the register level rather than erasing all the information since the information can be changed by using electrical signals. The EEPROM memory also has a special chip erase mode by which entire chip can be erased in 10 ms. This time is quite small as compared to time required to erase EPROM, and it can be erased and reprogrammed with device right in the circuit. However, EEPROMs are most expensive and the least dense ROMs.

Programmable Logic Array (PLA):

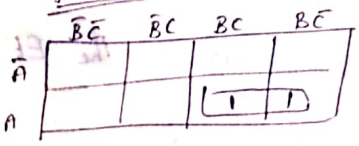
The combinational circuit do not use all the minterms every time. The result is that not all the bit patterns available in the ROM are used, which may be considered a waste of available equipment.

A programmable Logic Array (PLA) is similar to a ROM in concept; however it does not provide full decoding of the variables and does not generate all the minterms as in the ROM. In PLA, both AND and OR gates have fuses at the inputs, therefore in PLA both AND and OR gates are programmable.

→ Derive the PLA programming table for the combinational circuit that squares a 3-bit number.

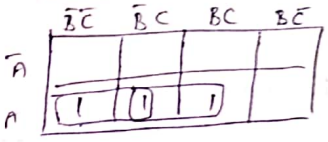
Input A B C	Decimal	Squared Decimal	f_1	f_2	f_3	f_4	f_5	f_6
0 0 0	0	0	0	0	0	0	0	0
0 0 1	1	1	0	0	0	0	0	1
0 1 0	2	4	0	0	0	1	0	0
0 1 1	3	9	0	0	1	0	0	1
1 0 0	4	16	0	1	0	0	0	0
1 0 1	5	25	0	1	1	0	0	1
1 1 0	6	36	1	0	0	1	0	0
1 1 1	7	49	1	1	0	0	0	1

for f_1



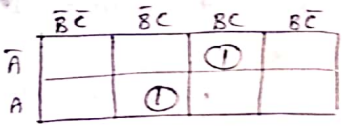
$f_1 = AB$

for f_2



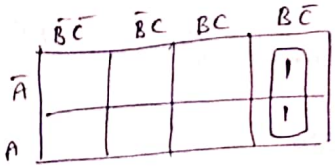
$f_2 = A\bar{B} + AC$

for f_3



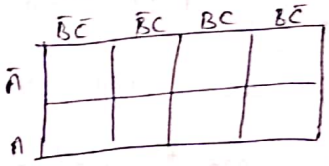
$f_3 = A\bar{B}C + \bar{A}BC$

for f_4



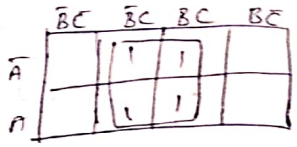
$f_4 = B\bar{C}$

for f_5



$f_5 = 0$

for f_6



$f_6 = C$

UNIT - 7

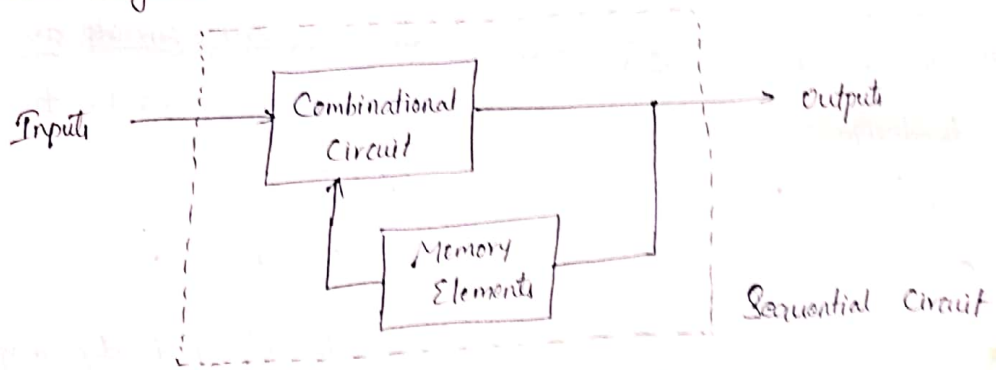
Sequential Circuits - I

Introduction :

The combinational digital circuit constitutes only a part of digital systems. The other major aspect of digital system is analysis and design of sequential circuits.

There are many applications in which digital outputs are required to be generated in accordance with the sequence in which the input signals are received. This requirement cannot be satisfied using a combinational logic system. These applications require outputs to be generated that are not only dependent on the present input conditions but they also depend upon the past history of these inputs. The past history is provided by feedback from the output back to the input.

Block diagram :



In above figure memory elements are connected to the combinational circuit as a feedback path.

The information stored in the memory elements at any given time defines the present state of the sequential circuit. The present state and the external inputs determine the outputs and the next state of the sequential circuit. Thus we can specify the sequential circuit by a external inputs, internal state (present & next), and outputs.

Comparison between Combinational & Sequential Circuits :

Combinational Circuits	Sequential Circuits
1) In combinational circuits, the output variables are at all times dependent on the combination of input variables.	1) In Sequential circuits, the o/p variables dependent not only on the present inputs variables but they also depend upon the past history of these input variables.
2) Memory unit is not required in Combinational Circuit	2) Memory unit is required to store the past history of i/p variables in the sequential circuit.
3) Combinational circuits are faster in speed because the delay b/w input & output is due to propagation delay of gates	3) Sequential circuits are slower than the combinational circuits.
4) Combinational circuits are easy to design	4) Sequential circuits are comparatively harder to design
5) Parallel adder is a Combinational Circuit.	5) Serial adder is a Sequential circuit.

The sequential circuits can be classified depending on the timing of their signals :

- Synchronous Sequential Circuits
- Asynchronous sequential Circuits

In synchronous sequential circuits, signals can affect the memory elements only at discrete instants of time.

In asynchronous sequential circuits, change in input signals can affect memory element at any instant of time.

The memory elements used in both synchronous & asynchronous sequential circuits are flip-flops which are capable of storing 1-bit binary information.

Comparison between Synchronous & Asynchronous Sequential Circuits :

Synchronous Sequential Circuits	Asynchronous Sequential Circuits
1. In synchronous circuits, memory elements are clocked flip-flops.	1) In asynchronous circuits, memory elements are either unclocked flip-flops or time delay elements.
2) In synchronous circuits, the change in i/p signals can affect memory element upon activation of clock signal.	2) In asynchronous circuits change in input signals can affect memory element at any instant of time.
3) The maximum operating speed of clock depends on time delays involved	3) Because of absence of clock, asynchronous circuits can operate faster than synchronous ckts
4) Easier to design	4) More difficult to design.

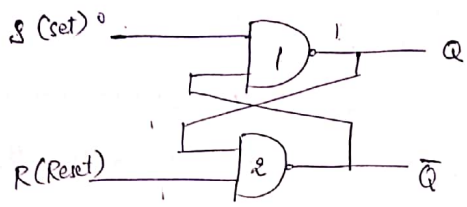
Latches and Flip-Flops :

Latches and flip-flops both are bistable elements. These are the basic building blocks of most sequential circuits. The main difference between latches and flip-flop is in the method used for changing their state. We use the name flip-flop for a sequential device that normally samples its inputs and changes its outputs only at time determined by clocking signal. We use the name latch for a sequential device that checks all of its inputs continuously and changes its outputs accordingly at any time independent of a clocking signal.

RS latch :

The simplest type of latch is the Set - reset (SR) latch. It can be constructed from either two NAND gates or two NOR gates.

RS latch using NAND gates :



0	0	1
0	1	1
1	0	1
1	1	0

In above figure two NAND gates are cross coupled so that the output of NAND gate 1 is connected to one of the inputs of NAND gate 2 and vice - versa. The latch has two outputs Q and $\bar{Q}$, and two inputs Set and Reset.

Case 1 : $S = 0$ and $R = 1$

Here NAND gate 1 input $S = 0$, hence the output of Q is at logic '1'. Both inputs to NAND gate 2 are at logic '1'. So the o/p $\bar{Q}$ is logic '0'.

Hence the output $Q = 1$ & $\bar{Q} = 0$

Case 2 : $S = 1$ and $R = 0$

Here R input of NAND gate 2 is '0'. So the output $\bar{Q}$ is logic '1'. Then the inputs of NAND gate '1' are at logic '1' So the output $Q = 0$.

Hence the output $Q = 0$ & $\bar{Q} = 1$.

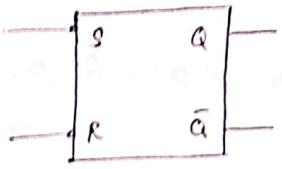
Case 3 : $S = 0$ and $R = 0$.

Here both the outputs of NAND gates are at logic '1'. So we call this an indeterminate or prohibited state, and represent this condition in the truth table as *. This condition also violates the basic definition of a latch that requires Q to be the complement of $\bar{Q}$.

Case 4: When $S=1$ and $R=1$.

Assume initially $Q=1$ & $\bar{Q}=0$. Then both the inputs of NAND gate 2 is at logic 1. So its output $\bar{Q}$ is zero. With $\bar{Q}=0$, one of the i/p of NAND gate 1 is at logic 0. So the output of NAND gate 1 is at logic 1. So the o/p $Q=1$ & $\bar{Q}=0$ means output state does not changes.

Symbol:



Truth Table:

S	R	Q_n	Q_{n+1}	State
0	0	0	X	Indeterminate
0	0	1	X	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No change
1	1	1	1	

→ When $S=0$, $R=0$ the o/p is unpredictable. This is called indeterminate condition

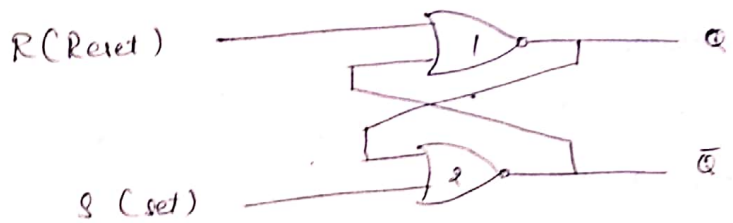
→ When $S=0$, $R=1$ the o/p of latch is Set

→ When $S=1$, $R=0$ the o/p of latch is reset.

→ When $S=1$, $R=1$ the output does not change.

* Note: The RS latch using NAND is also known as $R'S'$ latch.

RS latch using NOR gate:



Case (i): $S=1$ and $R=0$

Here NOR gate 2 input $S=1$, hence the output of gate $\bar{Q}$ is at logic 0. Both the inputs to NOR gate 1 are at logic 0. So that its output Q is at logic 1.

Hence, the output $Q=1$ & $\bar{Q}=0$

Case 2 : S=0 and R=1

Here R input of NOR gate 1 is logic '1' hence the output gate 1 Q is logic 0. So both the inputs of NOR gate 2 are now at logic 0. Hence the output of NOR gate 2 $\bar{Q}$ is at logic 1.

Hence, the outputs Q=0 & $\bar{Q}$=1

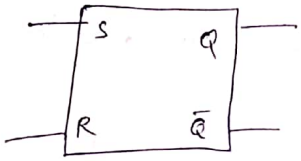
Case 3 : When S=0 & R=0

Assume initially Q=1 & $\bar{Q}$=0. Then both the inputs of NOR gate 1 are at logic 0. So its output, Q is at logic 1. With Q=1, one of the i/p of NOR gate 2 is at logic 1. So the o/p of NOR gate 2 is at logic 0. So the o/ps Q=1 & $\bar{Q}$=0 means o/p state does not changes.

Case 4 : S=1 and R=1

When R and S both are at logic 1, they force the outputs of both NOR gates to the low state (Q=0 and $\bar{Q}$=0). So we call this an indeterminate & prohibited state, and represent this condition as *.

Symbol:



Truth Table

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	X	Indeterminate
1	1	1	X	

→ When S=0, R=0 the o/p does not change

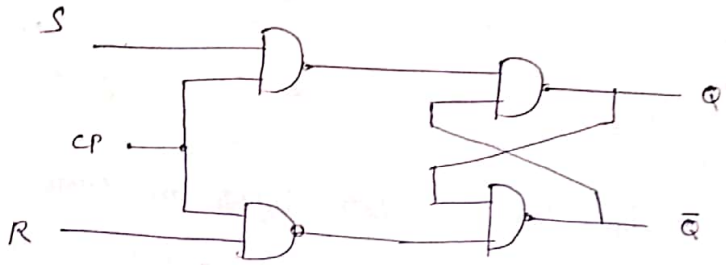
→ When S=1, R=0 then Q output of latch is set.

→ When S=0, R=1 then Q output of latch is reset.

→ When S=1, R=1 the output is unpredictable. This is called indeterminate condition.

Clocked SR Flip-Flop :

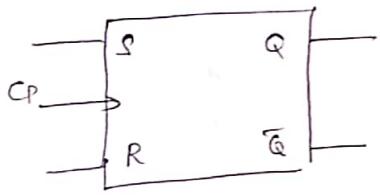
The clocked SR flip-flop is shown below.



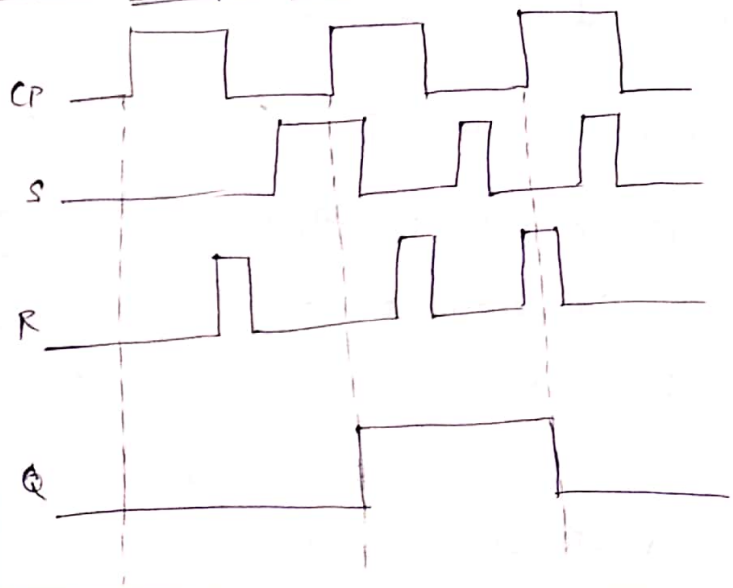
The circuit is similar to the SR latch. Here we are giving clock pulse (CP). On the positive edge of the clock pulse the circuit responds to the S and R inputs. At remaining places of clock the output is not changed it remains in the previous state only.

CP	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	x	Indeterminate
↑	1	1	1	x	
0	x	x	0	0	No change
0	x	x	1	1	

Symbol

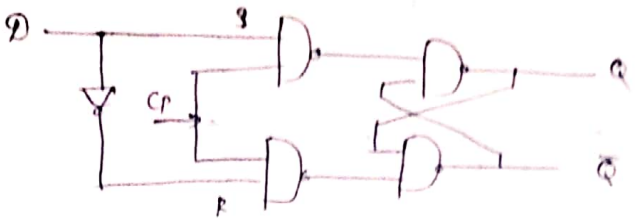


Input & Output waveforms :

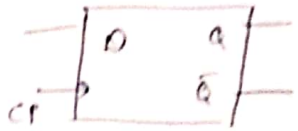


Clocked D Flip Flop :

In D flip-flop the basic SR flip-flop is used with complem...



Symbol :



In SR flip-flop when both inputs are same the output either does not change & it is invalid. In many practical applications, these input conditions are not required. These input conditions can be avoided by making them complement of each other.

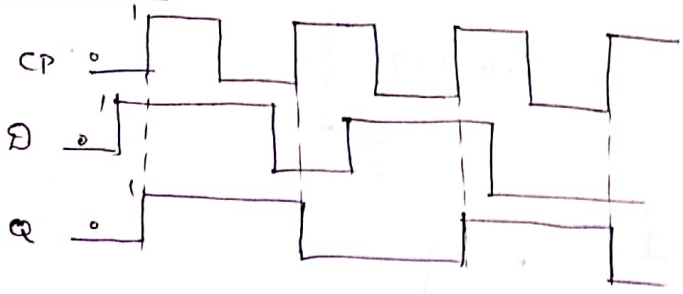
D input goes directly to the S input, and its complement is applied to the R input. Therefore only two i/p conditions exists, either $S=0$ & $R=1$ & $S=1$ & $R=0$.

Truth Table :

CP	D	Q_n	Q_{n+1}	State
↑	0	x	0	Reset
↑	1	x	1	Set
0	x	x	Q_n	Nochange

As shown in truth-table, the Q output follows the D input for this reason D flip-flop is sometimes called transperent flip-flop.

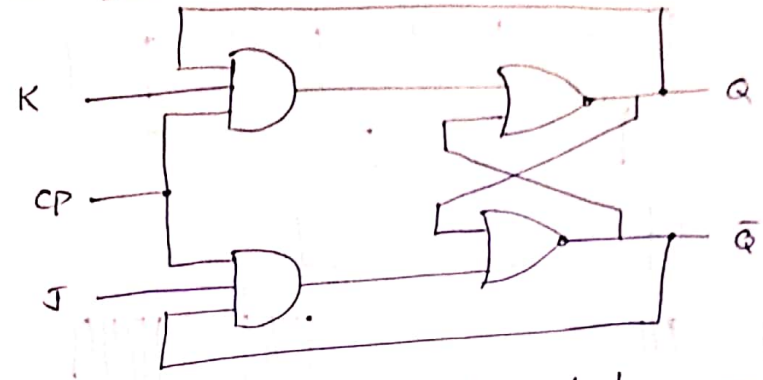
Wave forms :



In above wave-forms the Q_{n+1} function follows D input at the positive going edges of the clock pulses. Hence the Characteristic equation for D flip flop is

$$Q_{n+1} = D.$$

Clocked JK flip-flop :



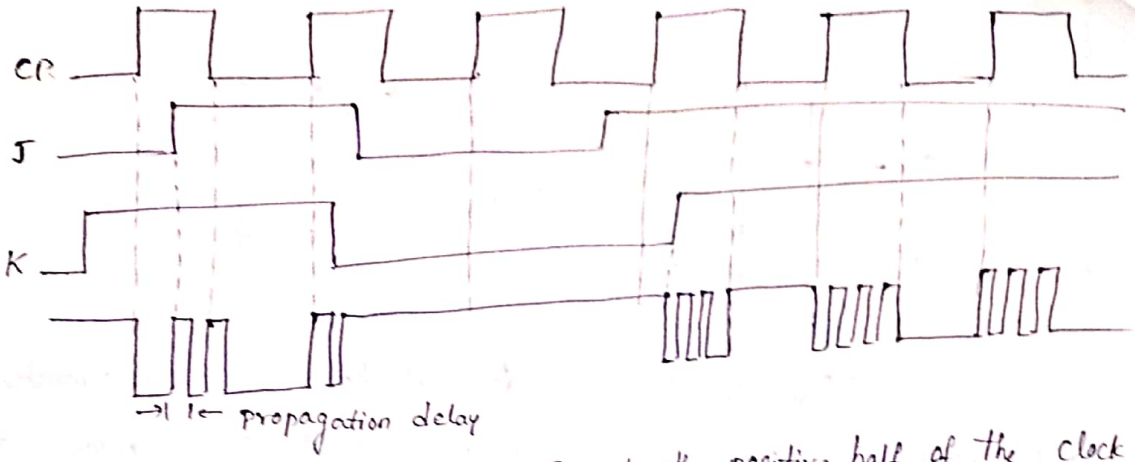
This JK flip-flop does not have an indeterminate input combination. The J input is analogous to the S (set) input and the K input to the R (reset). The indeterminate (SR = 11) i/p condition of the SR latch causes the JK latch to toggle ; when J = K = 1 then $Q_{n+1} = \bar{Q}_n$. By connecting the $\bar{Q}$ and Q outputs back to the J & K excitation input, the indeterminate condition that exists with the SR latch is no longer information.

When J and K are both low, both AND gates are disabled. Therefore, clock pulses have no effect and Q and $\bar{Q}$ retain their last values.

When J is low and K is High, the lower AND gate is disabled, so there is no way to set the flip-flop. The only possibility is reset. When Q is high, the upper gate passes through RESET trigger as soon as the next positive clock pulse arrives. This forces Q to become low. Therefore when J = 0, and K = 1 means that the next positive clock pulse resets the flip-flop.

When J is High and K is low, the upper gate is disabled, so there is no way to reset the flip-flop. The only possibility is to set the flip-flop if it is not previously set. With Q = 0 & $\bar{Q}$ = 1 and hence lower gate passes SET trigger on the next +ve clock pulse. This drives Q in to high. Therefore J = 1 & K = 0 means that next +ve clock pulse sets the flip-flop unless Q is already set.

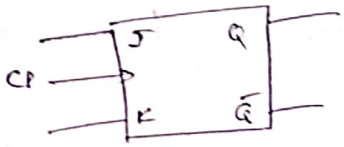
Input and Output waveforms :



In the above figure, Due to the positive half of the clock pulse and J and K are both high then output toggels continuously. This condition is known as 'race around condition' and must be avoided.

The solution to avoid race around condition in the JK is to create an edge triggered & pulse triggered JK flip flop.

Logic Symbol :



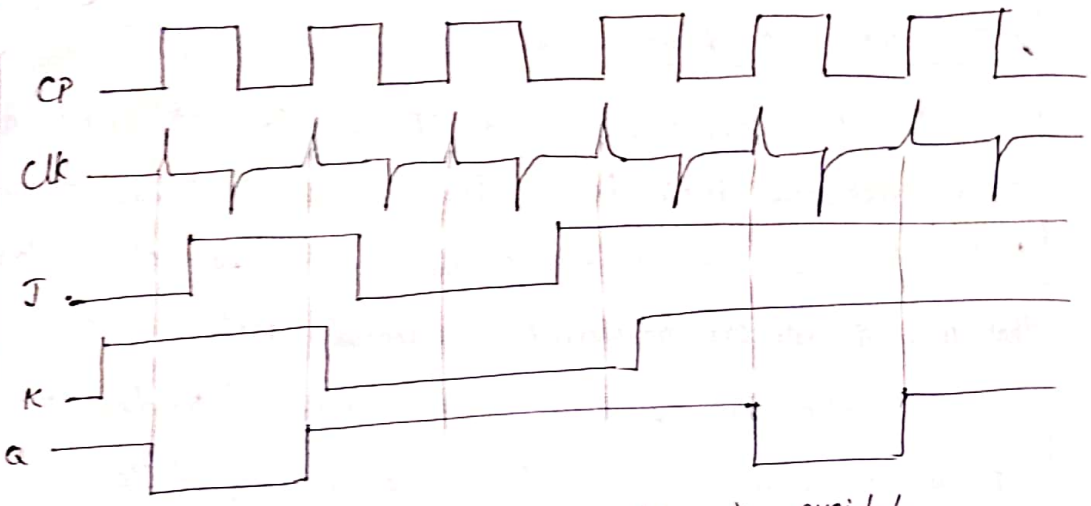
Truth Table :

CP	J	K	Q_n	Q_{n+1}
↑	0	0	0	0
↑	0	0	1	1
↑	0	1	0	0
↑	0	1	1	0
↑	1	0	0	1
↑	1	0	1	1
↑	1	1	0	1
↑	1	1	1	0
0	x	x	x	Q_n

≅

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

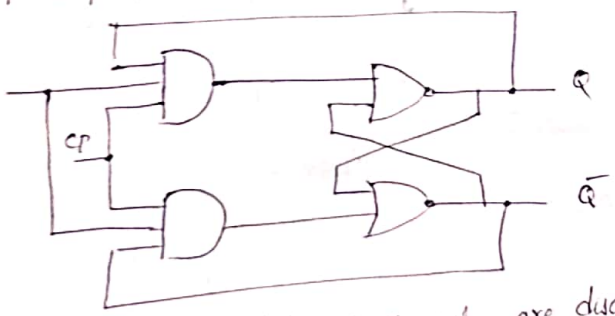
In case of pulse & edge triggering the flip flop changes state either at the positive edge (raising edge) & at the negative edge (falling edge) of the clock pulse.



Here race around condition is avoided.

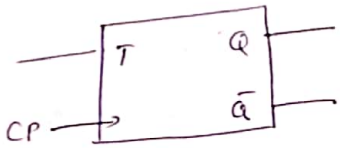
T flip-flop:

T- flip flop is also called as "Toggele flip-flop. The T flip flop is a modification of the JK flip-flop. T flip-flop is obtained from a JK flip-flop by connecting both inputs J and K together.



When T=0, both AND gates are disabled and hence there is no change in the O/p. When T=1, (i.e., J=1 & K=1) output toggels.

Logic Symbol:



Truth table:

CP	T	Q_n	Q_{n+1}
↑	0	0	0
↑	0	1	ϕ
↑	1	0	1
↑	1	1	0
↑			Q_n
0	x	x	

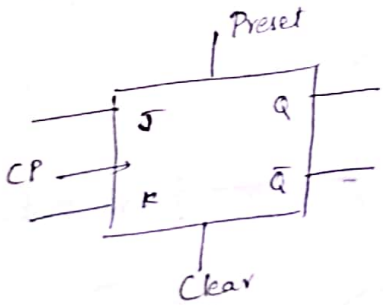
$\simeq$

T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

Asynchronous & Direct inputs:

The RS, D, JK and T flip flops, the inputs are called synchronous inputs because data on these inputs are transferred to the flip-flop's output only on the triggering edge of the clock pulse that is, the data are transferred synchronously with the clock.

Flip-flops available in IC packages sometimes provide special inputs for setting (preset) & clearing (clear) the flip f asynchronously. These inputs are called asynchronous & direct inputs. These inputs are connected directly into the latch portion of the flip-flop so that they override the effect of the synchronous inputs J, K and the clock.



Flip - Flop Conversions:

It is possible to convert one flip flop into another flip flop with some additional gates & simply doing some extra connections.

SR flip flop to D - flip flop:

Input (D)	Q_n	Q_{n+1}	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

For S

$\bar{D}$ \ $\bar{Q}_n$ Q_n	0	0
D	1	x

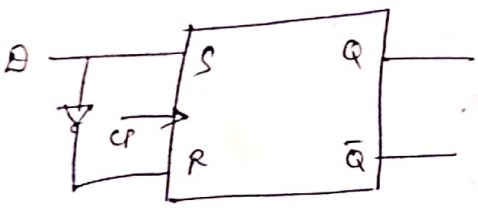
$S = D$

For R

$\bar{D}$ \ $\bar{Q}_n$ Q_n	x	1
D	0	0

$R = \bar{D}$

Logic Diagram:



SR-flip-flop to JK-flip-flop :

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1

For S

	$\bar{Q}_n\bar{K}$	$\bar{K}Q_n$	KQ_n	$K\bar{Q}_n$
$\bar{J}$	0	x	0	0
J	1	x	0	1

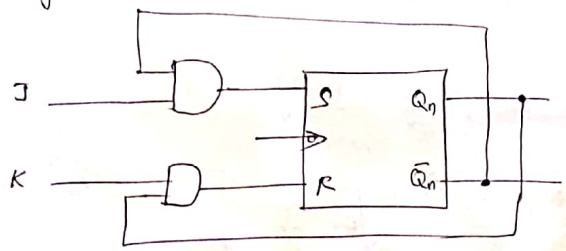
$S = J\bar{Q}_n$

For R

	$\bar{K}\bar{Q}_n$	$\bar{K}Q_n$	KQ_n	$K\bar{Q}_n$
$\bar{J}$	x	0	1	x
J	0	0	1	0

$R = KQ_n$

Logic Diagram :



SR-flip flop to T-flip flop :

T	Q_n	Q_{n+1}	S	R
0	0	0	0	x
0	1	1	x	0
1	0	1	1	0
1	1	0	0	1

For S

	$\bar{Q}_n$	Q_n
$\bar{T}$	0	x
T	1	0

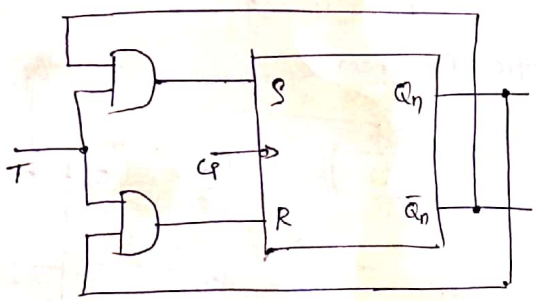
$S = T\bar{Q}_n$

For R

	$\bar{Q}_n$	Q_n
$\bar{T}$	x	0
T	0	1

$R = TQ_n$

Logic Diagram :



JK - flip flop to T - flip flop :

T	Q_n	Q_{n+1}	J	K
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1

For J

T \ Q_n	$\bar{Q}_n$	Q_n
$\bar{T}$	0	x
T	1	x

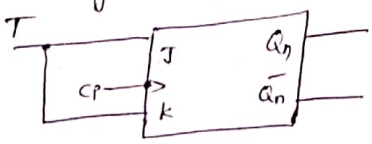
$J = T$

For K

T \ Q_n	$\bar{Q}_n$	Q_n
$\bar{T}$	x	0
T	x	1

$K = T$

Logic Diagram :



JK - flip flop to D - flip flop :

D	Q_n	Q_{n+1}	J	K
0	0	0	0	x
0	1	0	x	1
1	0	1	1	x
1	1	1	x	0

For J

D \ Q_n	$\bar{Q}_n$	Q_n
$\bar{D}$	0	x
D	1	x

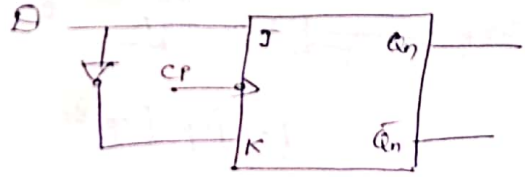
$J = D$

For K

D \ Q_n	$\bar{Q}_n$	Q_n
$\bar{D}$	x	1
D	x	0

$K = \bar{D}$

Logic Diagram :



D - flip flop to T - flip flop :

T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

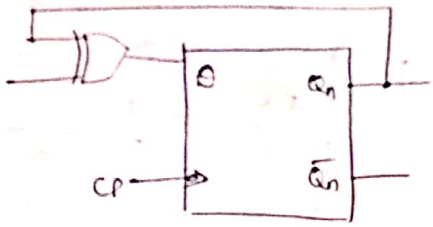
For D

T \ Q_n	$\bar{Q}_n$	Q_n
$\bar{T}$	0	1
T	1	0

$D = \bar{T} Q_n + T \bar{Q}_n$

$= T \oplus Q_n$

Logic Diagram :



T - flip-flop to D - flip-flop :

D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

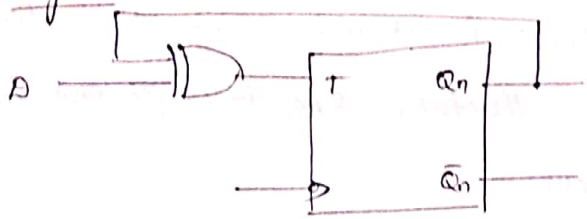
For T

D \ $\bar{Q}_n$	$\bar{Q}_n$	Q_n
0	0	1
1	1	0

$T = \bar{D} Q_n + D \bar{Q}_n$

$= D \oplus Q_n$

Logic Diagram :



Flip-flop Excitation / Application Tables :

During the design process, from the transition table, the sequence of states i.e., the transition from each present state to its corresponding next state. So we need a table that lists the required inputs for a given change of state. Such a table is known as an Excitation table of the flip-flop.

The excitation tables for flip-flops from their truth-tables. The excitation table consists of two columns Q_n & Q_{n+1}, and a column for each i/p show how the required transition can be achieved.

RS flip-flop Excitation table :

Truth table :

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	*

Excitation table

Q_n	Q_{n+1}	R	S
0	0	x	0
0	1	0	1
1	0	1	0
1	1	0	x

0 → 0 Transition :

The present state of flip-flop is 0 and is to remain 0 when a clock pulse is applied. From the truth table, when R = S = 0 or when R = 1 and S = 0. Thus, S has to be at 0, but R can be at 0, but R can be at either level. We indicate that condition with 'x'.

0 → 1 Transition :

The present state is 0 and is to change to 1. This can happen only when S=1 & R=0. Therefore, S has to be 1 and R has to be 0 for this transition to occur.

1 → 0 Transition :

The present state is 1 and is to change to 0. This can happen only when S=0 & R=1. Therefore, S has to be 0 and R has to be 1 for this transition to occur.

1 → 1 Transition :

The present state is 1 and is to remain 1. This can happen either when S=1 and R=0 or when S=0 and R=0. Thus R has to be 0 but S can be at either level. The table indicates this with a 'x' under S and '0' under R.

JK - flip flop Excitation table :

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

D - flip flop Excitation table :

D	Q_{n+1}
0	0
1	1

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

T - flip flop Excitation table :

T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

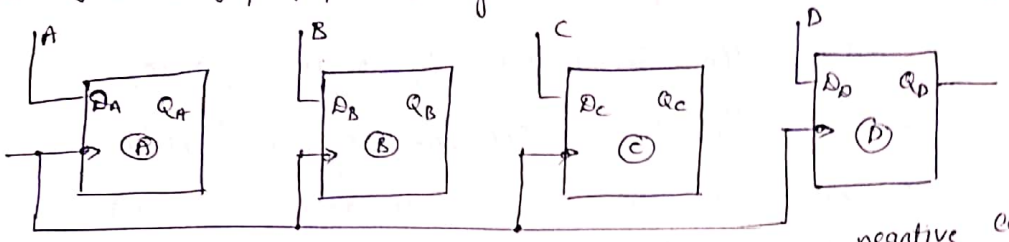
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Registers :

A register is a group of flip-flops. A flip-flop can store 1-bit information. So an n-bit register has a group of n flip-flops and is capable of storing any binary information / number containing n-bits.

Buffer Register :

The following figure shows the simplest register constructed with four D flip-flops. This register is also called as buffer register.

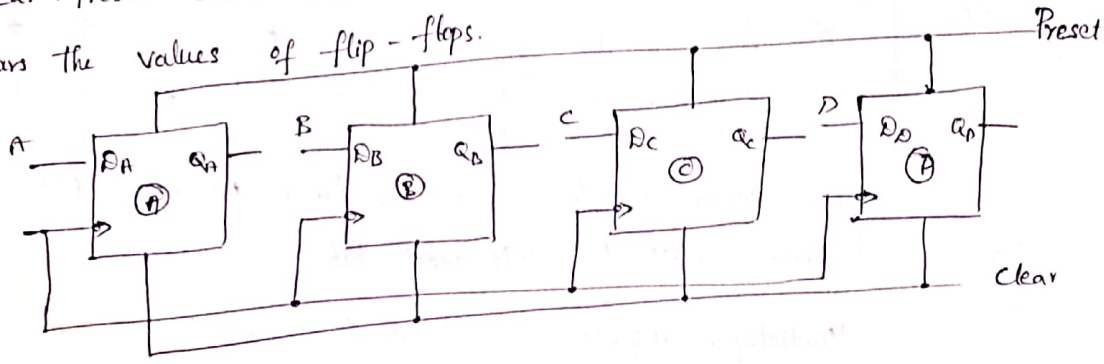


Each D-flip-flop is triggered with a common negative edge clock pulse. The input x bits set up the flip-flops for loading. Therefore, when the first negative clock edge arrives, the stored binary information becomes,

$$Q_A Q_B Q_C Q_D = ABCD$$

Control Buffer Register :

In this we are using two control inputs preset and clear. Preset control sets all the flip-flops and clear control clears the values of flip-flops.



Shift Registers :

The binary information (data) in a register can be moved from stage to stage, within the register & into & out of the register upon application of clock pulses.

This type of bit movement & shifting is essential for certain arithmetic and logic operations used in microprocessors. This gives rise to a group of registers called "Shift registers".

Shift registers are very important in applications involving storage and transfer of data in a digital system.

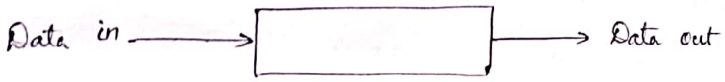
There are two types of Shift registers.

1. Shift Right register
2. Shift Left register.

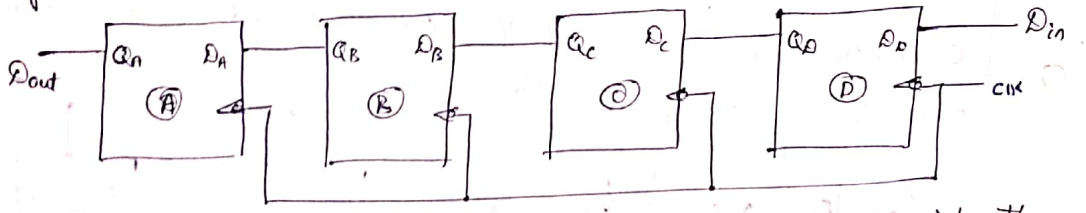
According to the data movement in a memory i.e., register, there are some types of shift registers. Those are

1. Serial In Serial Out Shift register (SISO)
2. Serial In Parallel Out Shift register (SIPO)
3. Parallel In Serial Out Shift register (PISO)
4. Parallel In Parallel Out Shift register (PIPO).

1) Serial In Serial Out Shift register :



The following figure shows serial in serial out left shift register.



The entry of the four bit binary number 1111 into the register beginning with the left most bit.

Initially, register is Cleared. So

$Q_A Q_B Q_C Q_D = 0000$

1) When data 1111 is applied serially, i.e.,

Left most 1 is applied as Din

$D_{in} = 1$, $Q_A Q_B Q_C Q_D = 0000$

The arrival of the first falling clock edge sets the right most flip flop, and the stored word becomes,

$Q_A Q_B Q_C Q_D = 0001$

2) When the next negative clock edge hits, the Q_A flip-flop sets and the register contents become

$$Q_A\ Q_B\ Q_C\ Q_D = 0011$$

3) The third negative clock edge results in

$$Q_A\ Q_B\ Q_C\ Q_D = 0111$$

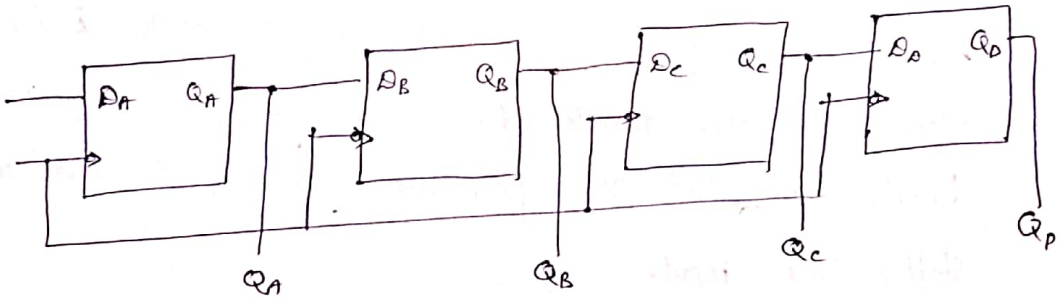
4) The fourth falling clock edge gives

$$Q_A\ Q_B\ Q_C\ Q_D = 1111$$

Here the serial in Serial out transmission can be done using Shift Right register also.

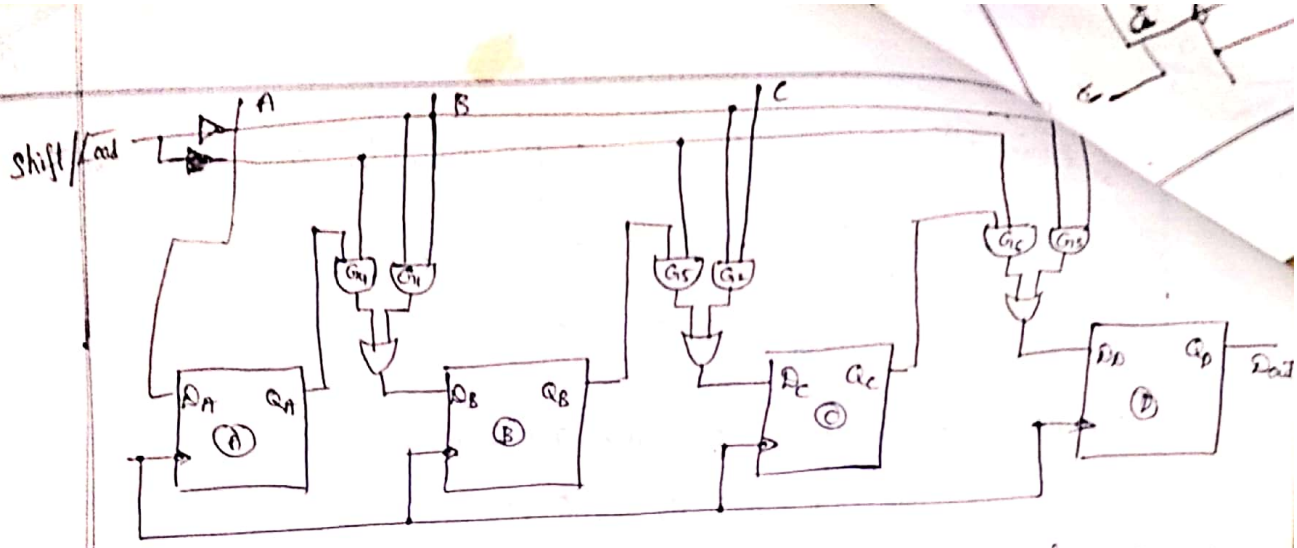
Serial In Parallel Out Shift register :

In this case, the data bits are entered into the registers serially. But the output is taken in parallel. Once the data are stored, each bit appears on its respective output line and all bits are available simultaneously, instead of on a bit-by bit basis as with the serial output.



Parallel In Serial Out Shift register :

The bits are entered here in parallel that is simultaneously into their respective stage on parallel lines. The following figure represents a 4-bit parallel in serial out register.



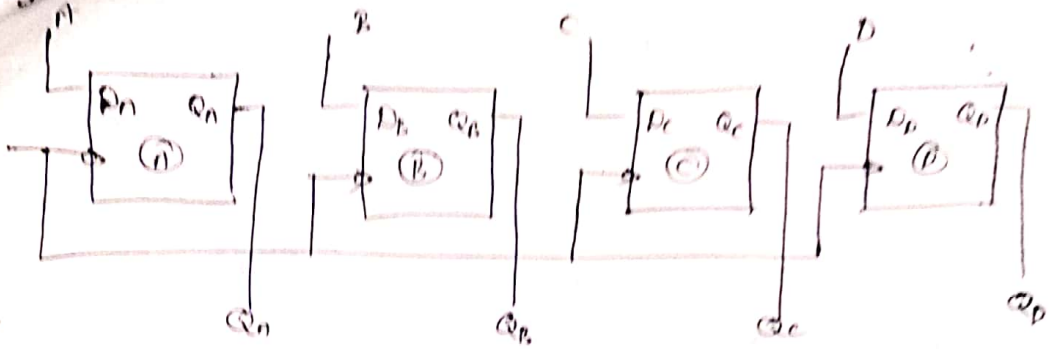
In the above figure, there are four input line A, B, C, D for entering data in parallel into the register. Shift / $\overline{Load}$ is the control input which allows shift & loading data operation of the register. When Shift / $\overline{Load}$ is low, gates G_1, G_2, G_3 are enabled, allowing each input data bit to be applied to D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with D = 1 will set and those with D = 0 will Reset. Thus all four bits are stored simultaneously.

When shift / $\overline{Load}$ is high, gates G_1, G_2, G_3 are disabled and gates G_4, G_5, G_6 are enabled. This allows the data bits to shift left from one stage to the next. The OR gates at the D flip-flop allow either the parallel data entry operation & shift operation, depending on which AND gates are enabled by the level on the shift / $\overline{Load}$ input.

Parallel In Parallel Out Shift register :

Here the data is entered in parallel i.e., all bits simultaneously into the register, and data out in parallel from the register

In parallel in parallel out register, there is simultaneous entry of all data bits and the bits appears on parallel outputs simultaneously.

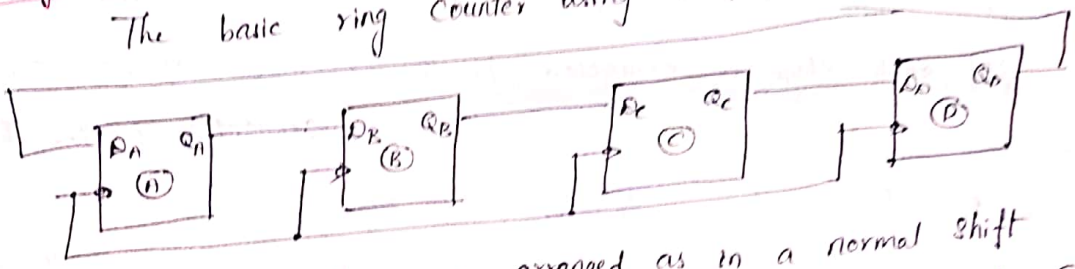


Shift Register Counters :

A shift register can also be used as a counter. A shift register with the serial output connected back to the serial input is called shift register counter. The most common shift register counters are the ring counter and the Johnson counter.

Ring Counter :

The basic ring counter using D-flip-flop shown in figure

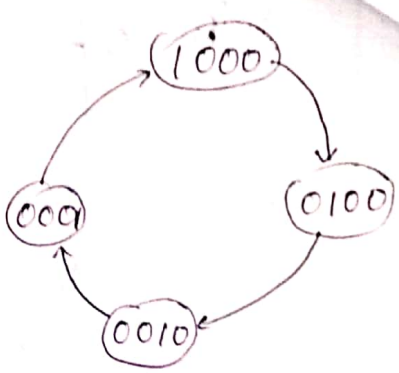


The flip-flops are arranged as in a normal shift register i.e., the Q output of each stage is connected to the D input of the next stage. But the Q output of last flip-flop is connected back to the D input of the first flip-flop such that the array of flip-flops is arranged in a ring and therefore the name ring counter.

A ring counter is a circular shift register with only one flip-flop being set at any particular time, all others are cleared.

The single bit is shifted from flip-flop to the next to produce the sequence of timing signals. The initial value of the register is 1000. The single bit is shifted right with every clock pulse and circulated back from Q_4 to Q_1.

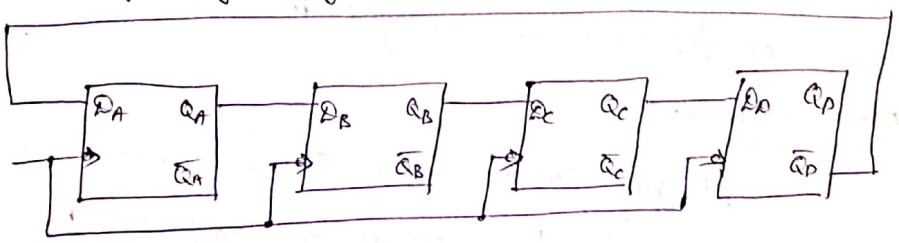
CP	Q_1	Q_2	Q_3	Q_4
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1



Johnson Counter :

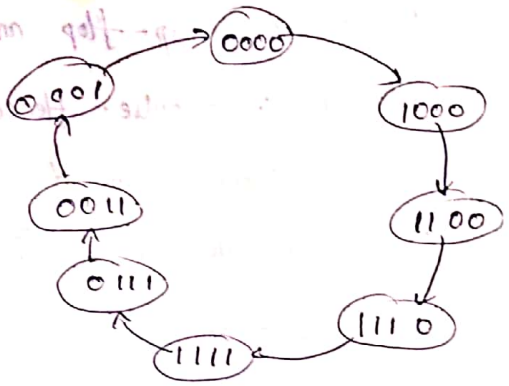
This counter is obtained from a serial in serial out shift register by providing a feedback from the inverted output of the last flip flop to the D input of the first flip flop. The Q o/p of each stage is connected to the D input of next stage but the $\overline{Q}$ output of the last stage is connected to the D input of first stage, therefore, the name twisted ring counter.

The number of states can be doubled if the register is connected as a switch-tail ring counter. The logic diagram of a 4 bit Johnson counter using D-flip flop is shown in following figure.



Let initially all flip flops be reset, i.e., the state of the counter be 0000. After each clock pulse, the level of Q_1 is shifted to Q_2, the level of Q_2 to Q_3 to Q_4 and the level of $\overline{Q_4}$ to Q_1. This sequence is repeated after every clock pulses.

CP	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0
9	1	0	0	0



State diagram

Counters :

Counter is an sequential circuit capable of counting the no. of clock pulses arriving at its clock input. A counter is set of flip-flops whose state change in response to pulses applied at the input to the counter. The flip-flops are interconnected such that their combined state at any time is the binary equivalent of the total no. of pulses that have occured upto that time.

A counter that follows the binary sequence is called a "binary counter". An n-bit binary counter consists of n-flip flops and can count in binary from 0 to $2^n - 1$. It is also called as modulus counter or MOD-N counter where N = no. of states as the counter output = 2^n.

Types of Counters :

1. Ripple or Asynchronous or non-Synchronous or Serial counters.
2. Synchronous or Parallel Counters.

In synchronous counters, the common clock i/p is connected to all the flip-flops and thus they are clocked simultaneously.

In asynchronous counters, each flip-flop is triggered by the previous flip-flop and the first flip-flop is triggered by the external clock pulse. Hence in asynchronous counters, the flip-flops are not clocked simultaneously and each successive flip-flop is clocked by the Q & $\bar{Q}$ output of the previous flip-flop.

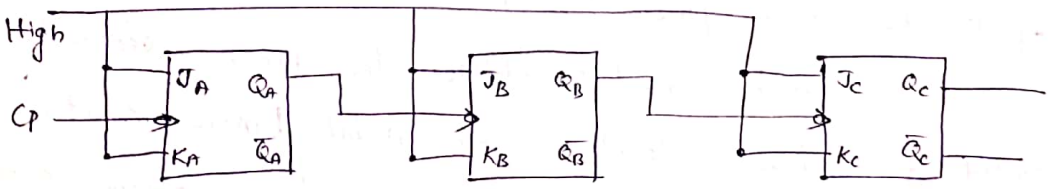
Asynchronous Counter :

To design the ripple counter, the no. of flip-flops required depends on the no. of states. The no. of o/p states of the counter is called "Modules" (MOD) of the counter. The maximum no. of states of a counter is 2^n, where n is the no. of flip-flops in the counter.

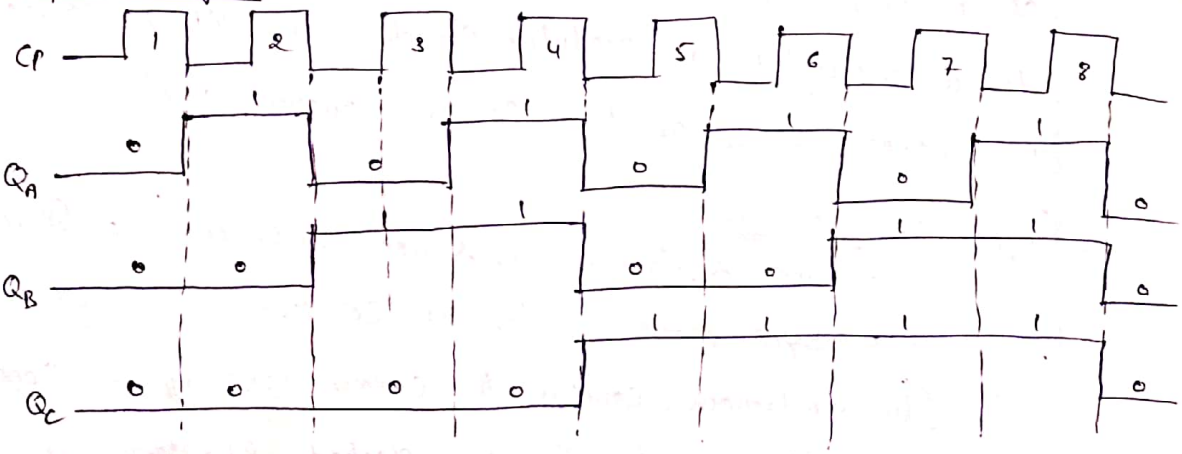
MOD-8 Counter :

A counter built with JK flip-flops. It can be seen that the J and K inputs of all the flip-flops are connected to high, so that all the flip-flops are in toggle mode. And also all the JK flip-flops are negative edge triggered.

MOD-8 Counter is a 3-bit asynchronous counter, the count sequence starts from 000 to 111 and uses three flip flops.



Output waveforms :

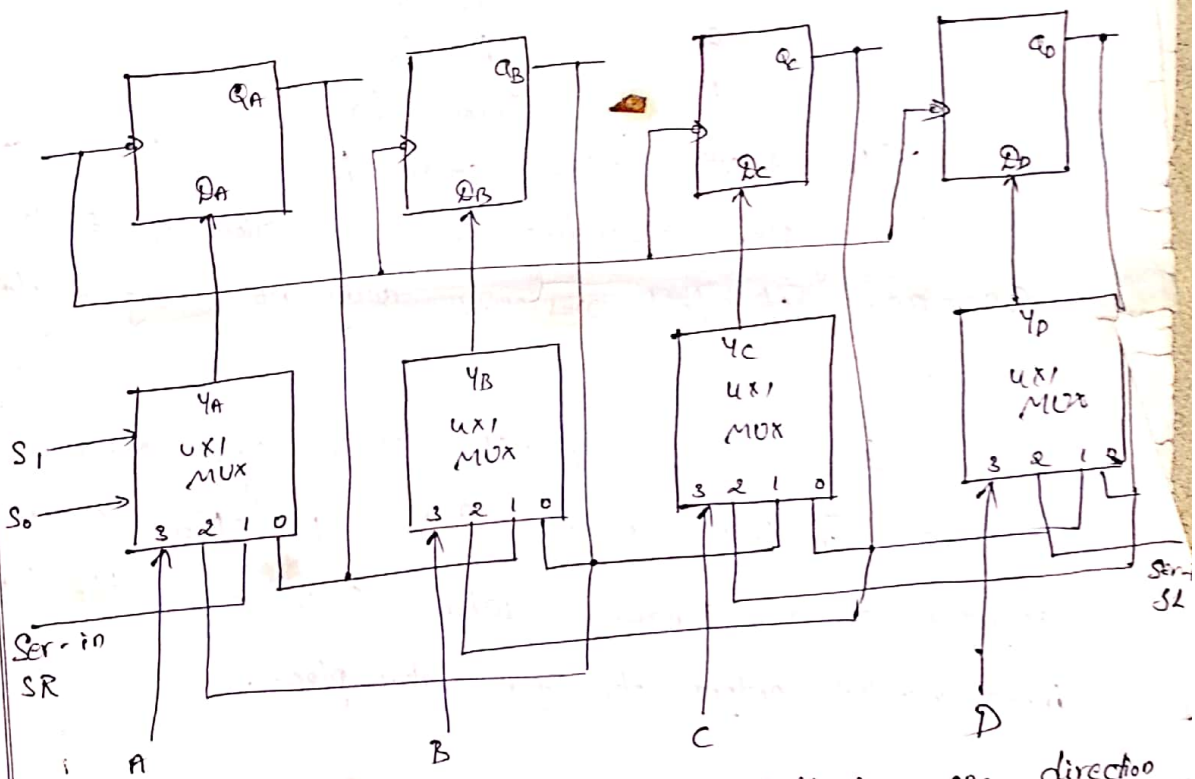


pulse occurs, then data is effectively shifted one place to the left.

Thus, a HIGH on the RIGHT/LEFT control input allows the data to be shifted to the right and a LOW on the R/$\bar{L}$ enables the data to be shifted to the left.

Universal Shift register:

An universal shift register is one that has both serial and parallel inputs.



A register capable of shift in one direction only is ~~else~~ unidirectional shift register one that can shift in both direction is a birectional shift register If register has both shifts are parallel load capabilities it is refered to as universal shift register.

A universal shift register consists of four D-flip flops and 4 multiplexers. The multiplexers have two common selection inputs S_1 and S_0. Input '0' in each multiplexer is selected when $S_1S_0 = 00$, Input '1' is selected when $S_1S_0 = 01$ and input '2' is selected when $S_1S_0 = 10$ & input 3 is selected when $S_1S_0 = 11$.

S_1	S_0	Mode of operation
0	0	No change
0	1	Shift right
1	0	Shift Left
1	1	Parallel load

The selection inputs control the mode of operation of the register according to the function of entries when $S_1S_0 = 00$. It held previously and no change state occurs.

When $S_1S_0 = 01$ the shift right operation occurs. Similarly when $S_1S_0 = 10$ then shift left operation occurs. Finally, when $S_1S_0 = 11$ then parallel input parallel output transfer takes place.

Here state a and b leads to common state c when input is 0. Therefore, we can eliminate state a & b.

2. If the state transition from two different states of the same input does not lead to same state then state o/p becomes the output corresponds to each input transition of that state.