Module - 2

## FLIP FLOPS

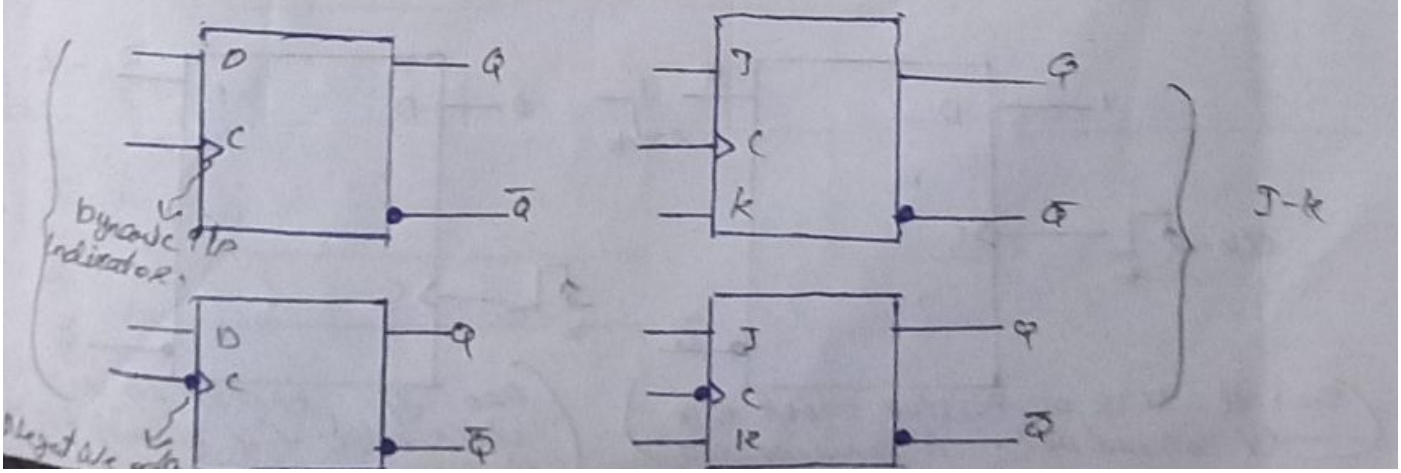
Flip flops are synchronous bistable devices, also known as bistable multivibrators.

The term synchronous means that the output changes states only at a specified point - (leading / trailing edge) on the triggering i/p called the clock (CLK), which is designated as a control i/p, C; that is, changes in the o/p occur in synchronization with the clock.

Flip flops are edge-triggered / edge-sensitive whereas gated latches are level-sensitive.

An edge-triggered flip-flop changes states either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse & is sensitive to its i/ps only at this transaction of the clock.

Two types of edge triggered flips-flops are covered here: D & J-K. The logic symbol for these flip-flops are given below.



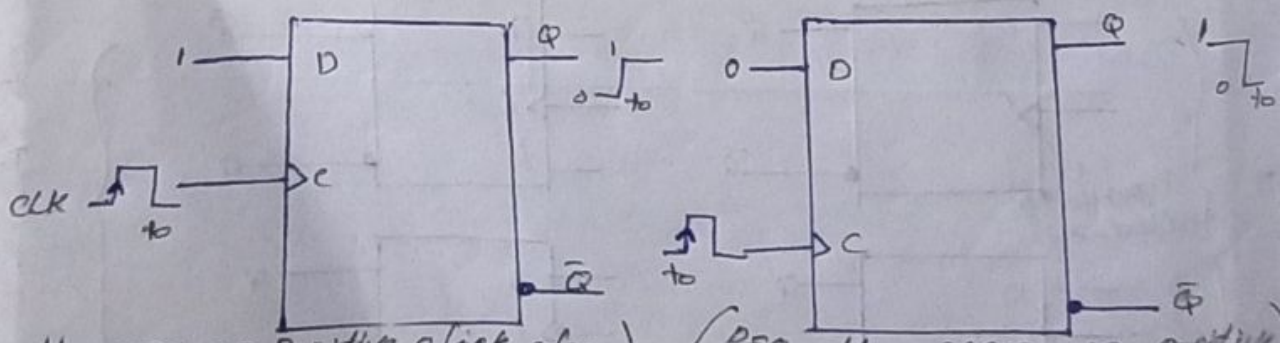
Each type can be either positive edge-triggered (no bubble @ clk) or negative edge-triggered (bubble @ clk).
The key to ~~indeed~~ identifying an edge-triggered flip-flop by its logic symbol is the small triangle inside the block at the clock (c) input. This triangle is called the dynamic input indicator.

## The D ff:

The D input of the D ff is a synchronous i/p because data on the i/p are transferred to the flip-flop's output only on the triggering edge of the clock pulse.

When D is ~~High~~ HIGH, the Q o/p goes HIGH on the triggering edge of the clock pulse, & the flip-flop is SET, when D is low, the Q output goes LOW on the triggering edge of the clock pulse & the ff is RESET.

This basic operation of a positive edge-triggered D-ff is given in below:




(D=1 ff SETS on Positive clock edge (if already SET. ff remains SET)

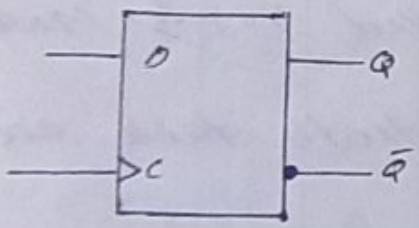
(D=0 ff RESETS on positive clock edge (if already RESET, it remains RESET)

The truth table for positive-edge-triggered D ff. is given below:

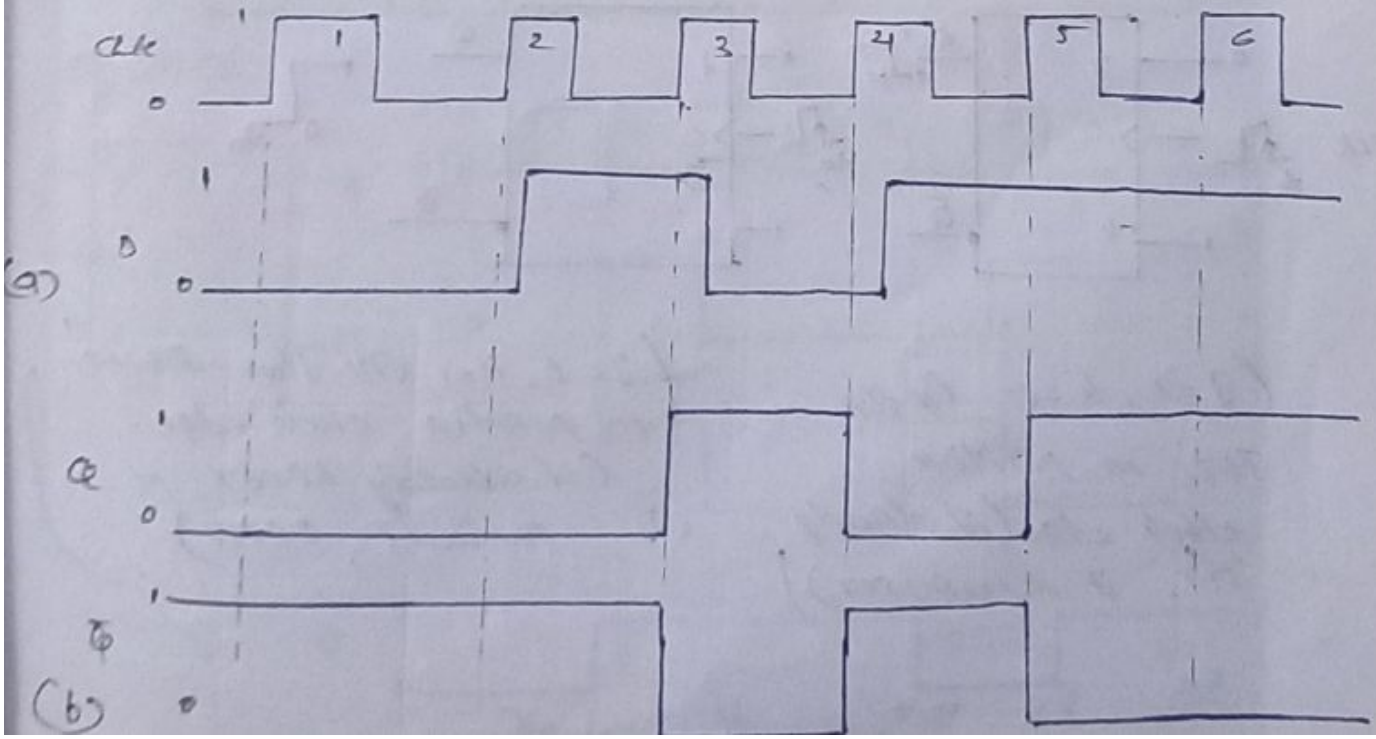
| Inputs | | outputs | | Comments |
|---|---|---|---|---|
| D | clk | Q | $\bar{Q}$ | |
| 0 | ↑ | 0 | 1 | RESET |
| 1 | ↑ | 1 | 0 | SET |

The operation and truth table for negative edge-triggered D ff. are the same as those for a positive edge-triggered device except that the falling edge of the clock pulse is the triggered edge.

Example:




(The clock pulse and restarts when a SET start it starts.)



# The JK ff :

The J and K i/p of the J-K ff are synchronous i/p because data on these i/ps are transferred to the flip-flop's o/p only on the triggering edge of the clock pulse.

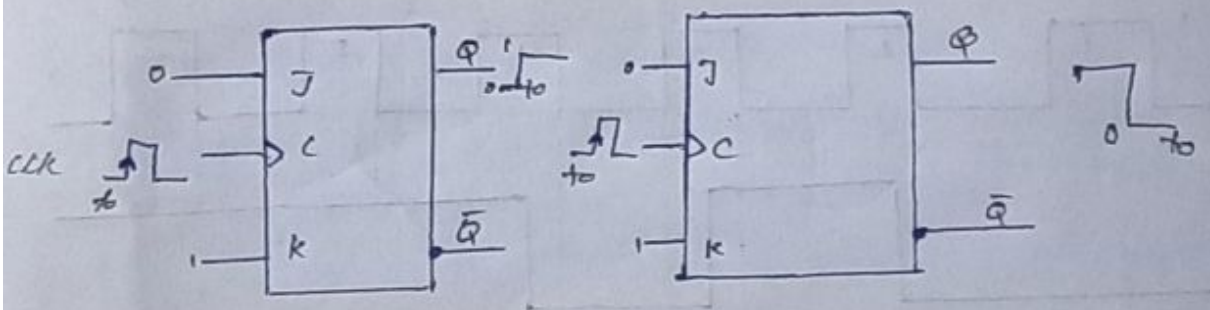
When J is HIGH & K is LOW, the Q o/p goes HIGH on the triggering edge of the clock pulse, & the flip-flop is SET.

When J is LOW & K is HIGH, the Q o/p goes LOW on the triggering edge of the clock pulse, & the flip flop is RESET.

When both J and K are low, the o/p does not change from its prior state.

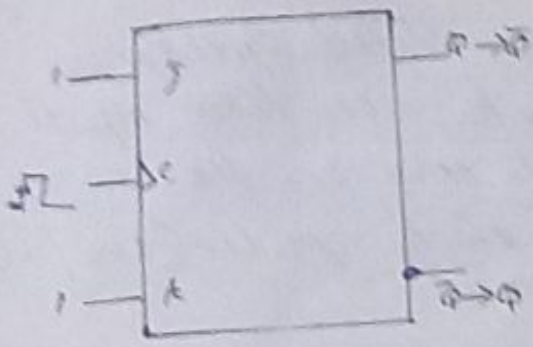
When J and K are both HIGH, the flip flop changes state, This is called Toggle mode

This operation and its truth table are as follows:

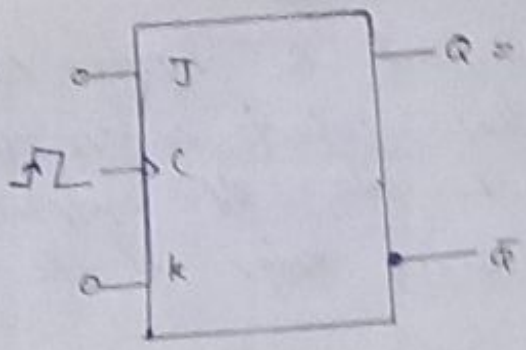



(J=1, K=0 flip-flop SETS on positive clock edge (if already SET, it remains SET.))

(J=0, K=1 flip-flop RESETS on positive clock edge. (if already RESET, it remains RESET))



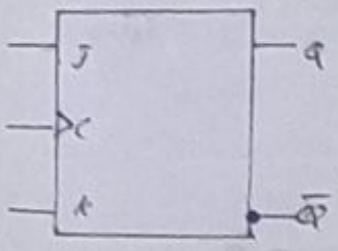

(J=1, k=1 flip flop changes state (toggle))

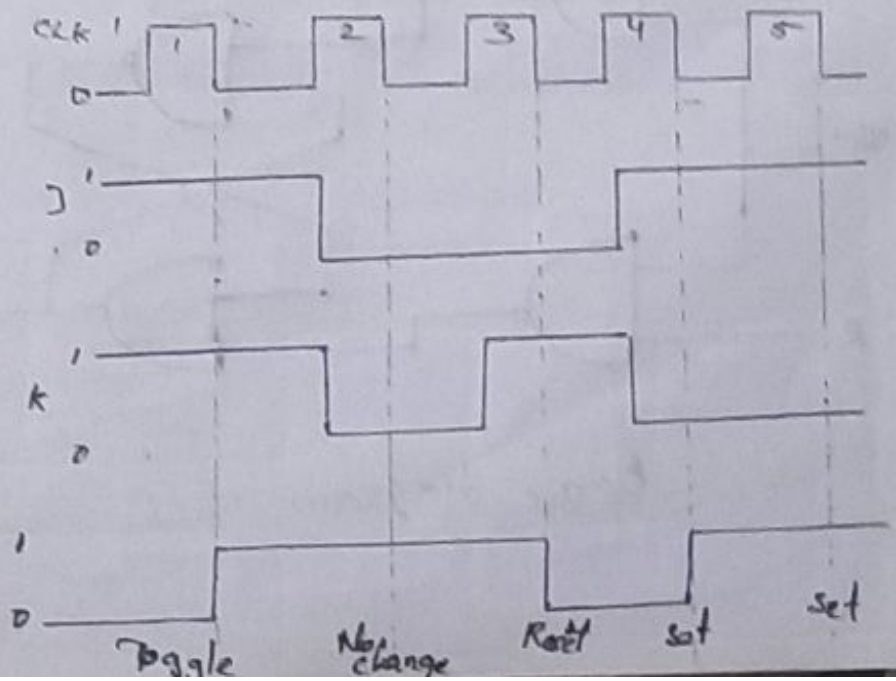



(J=0, k=0 flip-flop does not change (if set, it remains set; if reset, it remains reset))

| inputs | | | outputs | | comments |
|---|---|---|---|---|---|
| J | k | CLK | Q | Q̄ | |
| 0 | 0 | ↑ | $Q_0$ | $\bar{Q}_0$ | no change |
| 0 | 1 | ↑ | 0 | 1 | RESET |
| 1 | 0 | ↑ | 1 | 0 | SET |
| 1 | 1 | ↑ | $\bar{Q}_0$ | $Q_0$ | toggle. |

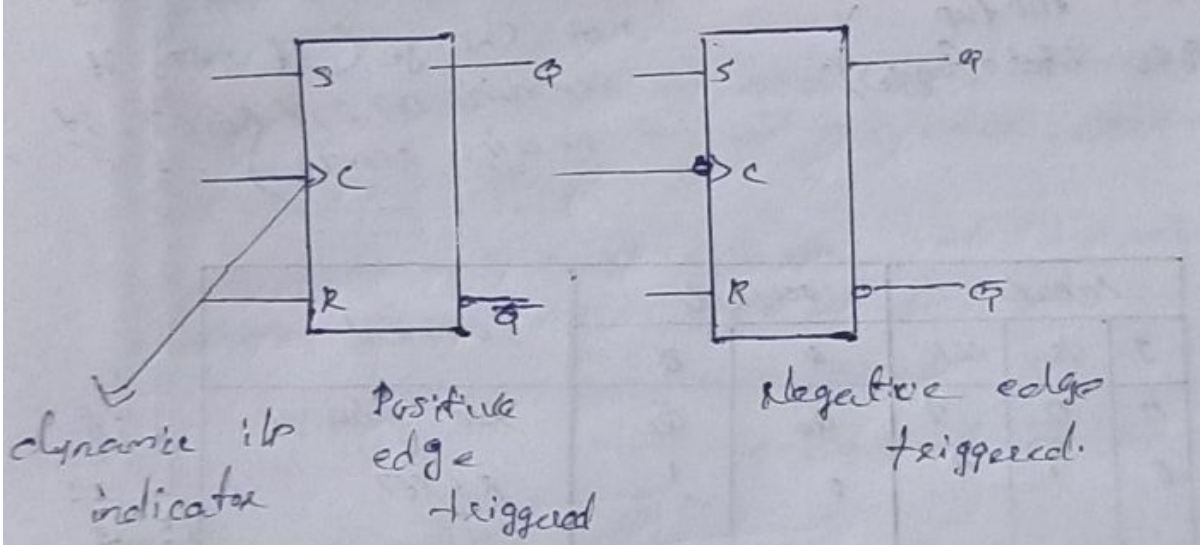
Example:

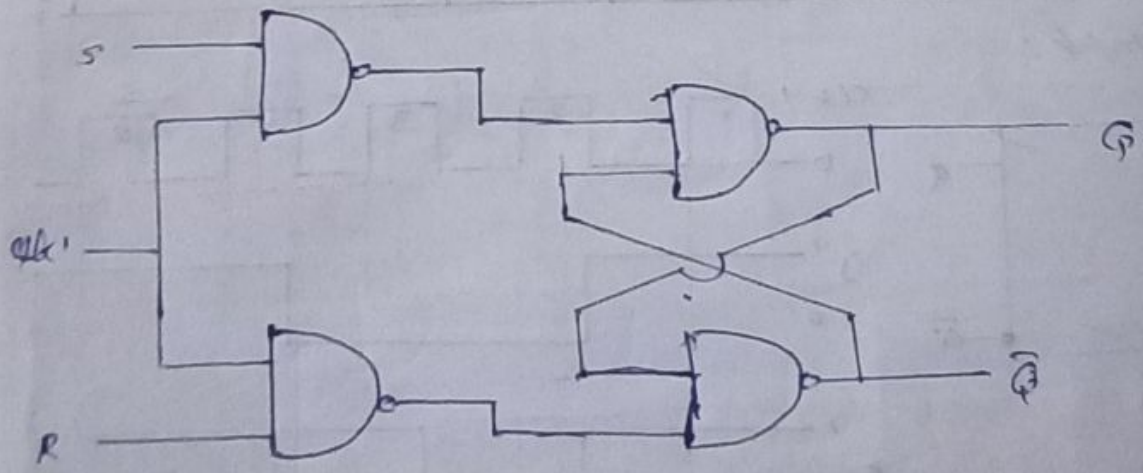


## SR Flip flops :

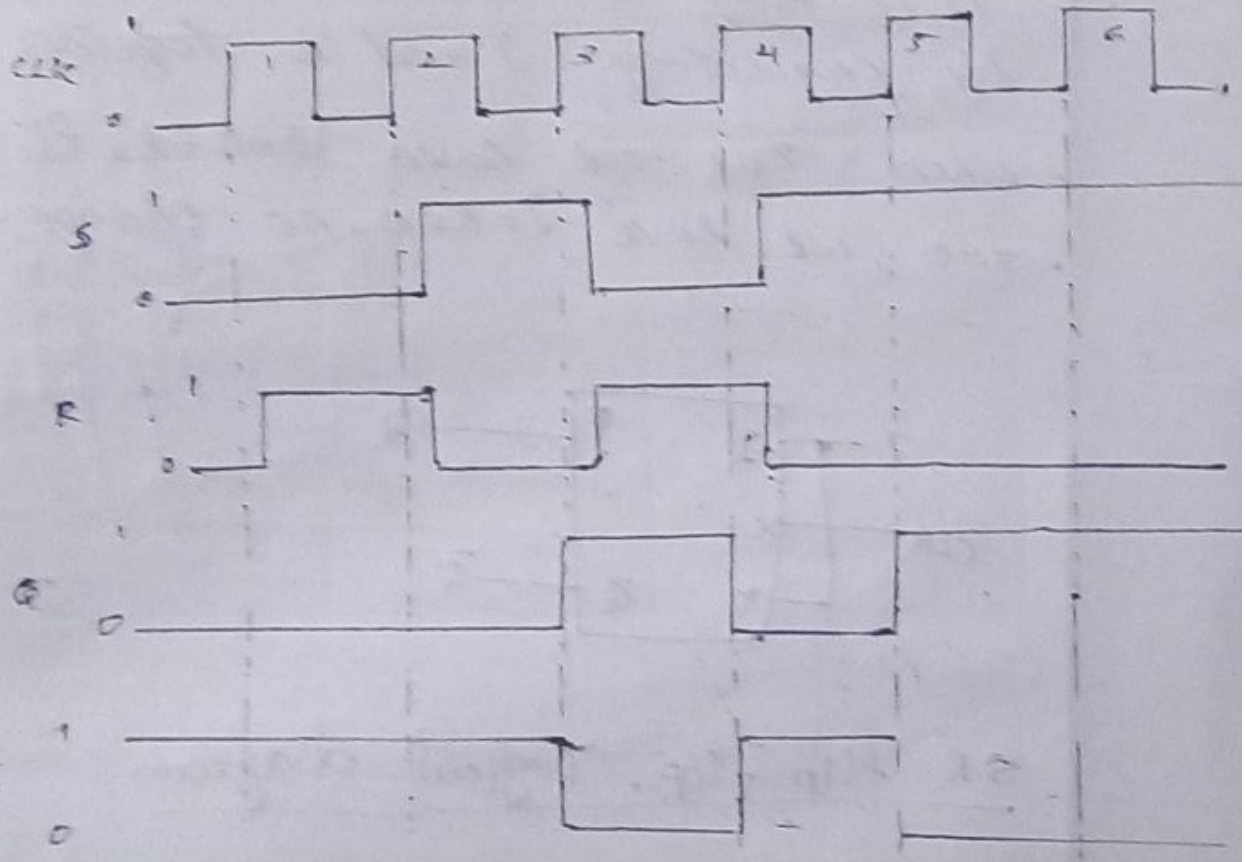
The S and R inputs are called synchronous control inputs → The data on these inputs affect the ff's output only on the triggering edge of the clock pulse (+ve / -ve)



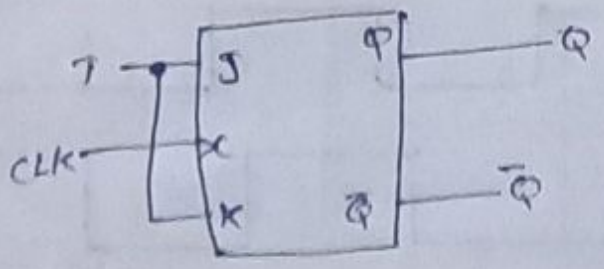

Logic diagram.

| Inputs | | | Outputs | | Comments |
|---|---|---|---|---|---|
| S | R | CLK | Q | $\overline{Q}$ | |
| 0 | 0 | X | $Q_0$ | $\overline{Q_0}$ | No change |
| 0 | 1 | ↑ | 0 | 1 | RESET |
| 1 | 0 | ↑ | 1 | 0 | SET |
| 1 | 1 | ↑ | ? | ? | Invalid. |

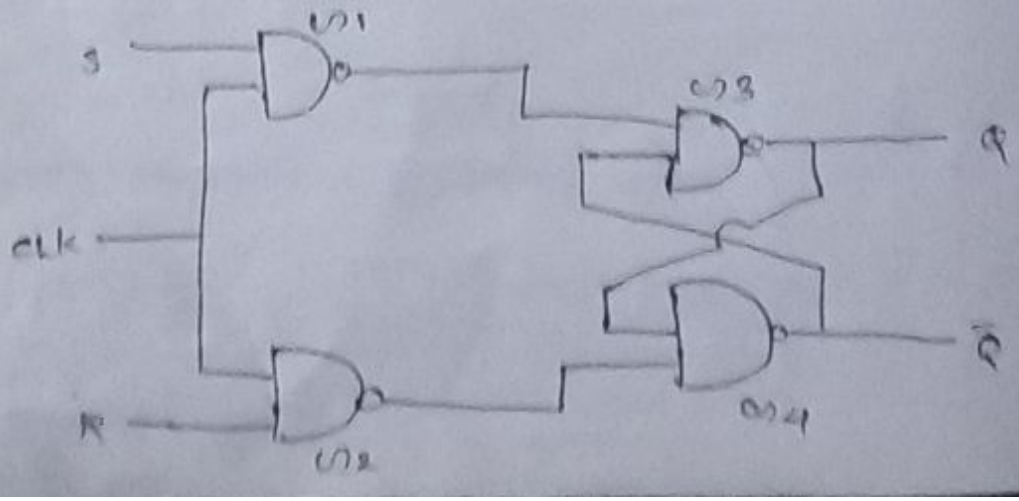


## T flip Flops:

- It has a single control input labelled T for Toggle.
- when a T is HIGH, the flip flop toggles on every new clock pulse.
- when T is LOW the flip flop remains in the same stage.
- It is easy to convert J-K ff to T ff by connecting J and K together
- when T=1, we have J=K=1, ff toggles.
- T=0, we have J=K=0, no change.




## SR flip flop logical diagram



| S | R | Q |
|---|---|---|
| 0 | 0 | No Change |
| 0 | 1 | RESET |
| 1 | 0 | SET |
| 1 | 1 | Invalide. |

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

Let $Q=0$

$g_1=1$ $g_2=1$

$(g_3=0)$ $g_4=1$

Let $Q=1$

$g_1=1$ $g_2=1$

$(g_3=1)$ $g_4=0$

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

let $Q=0$

$g_1=1$ $g_2=0$ RESET

$(g_3=1)$ $g_4=1$

Case:3 $S=1$ $R=1$ ~~$CLK=1$~~ $S=1$ $CLK=1$

$g_1=0$ $(g_3=1)$ $1\times1=0$ Nand

$g_2=1$ $g_4=0$

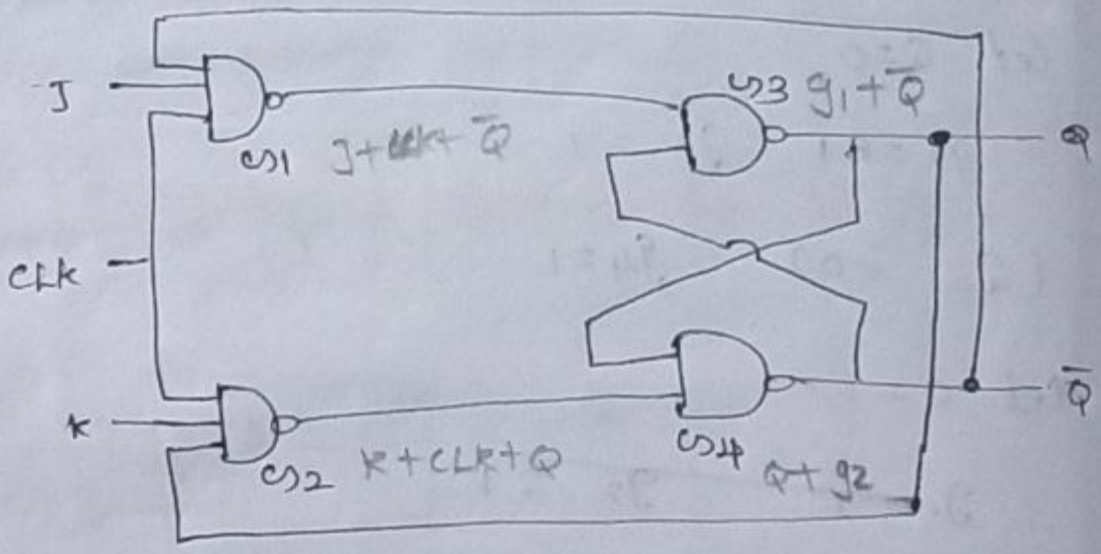
case :4

S = 1 R = 1 Clk = 1

$g_1 = 0$ $(g_3 = 1)$

$g_2 = 0$ $g_4 = 1$ $\bar{Q}$

## Jk flip flops Logic diagram




Case :1

J=0 K=0 Clk=1

Let Q=0 $\bar{Q}$=1

$G_1 = 1$ $G_3 = 0$

$G_2 = 1$ $G_4 = 1$

let Q=1 $\bar{Q}$=0

$G_1 = 1$ $G_3 = 1$

$G_2 = 1$ $G_4 = 0$

Case :2

J=1 K=0

let Q=0 $\bar{Q}=1$

$o_1=0$ $o_3=1$

$o_2=1$ $o_4=0$

let Q=1 $\bar{Q}=0$

$o_1=1$ $o_3=1$

$o_2=1$ $o_4=0$

Case :3

J=0 K=1

let Q=0 $\bar{Q}=1$

$o_1=1$ $o_3=0$

$o_2=1$ $o_4=1$

let Q=1 $\bar{Q}=0$

$o_1=1$ $o_3=0$

$o_2=0$ $o_4=1$

Case :4

J=1 K=1

let Q=0

$o_1=0$ $o_3=1$

$o_2=1$ $o_4=0$

let Q=1 $\bar{Q}=0$

$o_1=1$ $o_3=0$

$o_2=0$ $o_4=1$

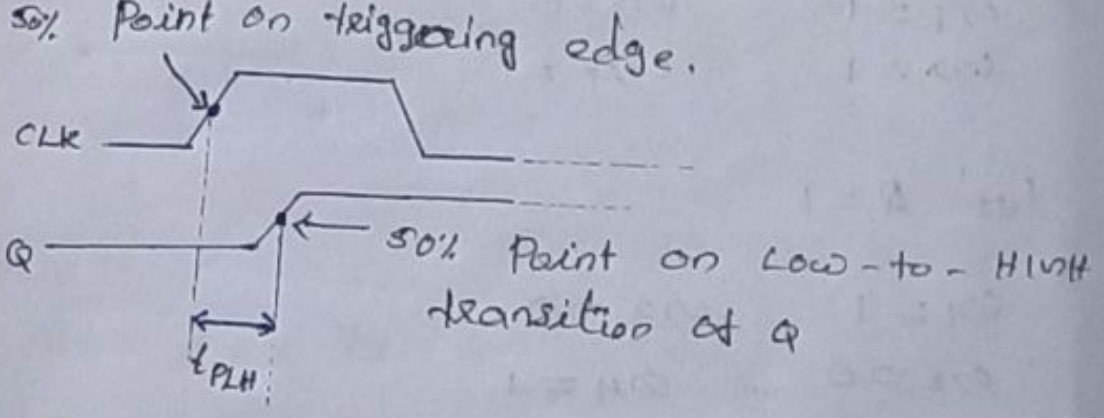
# flip-flop operating Characteristics

## Propagation Delay time:

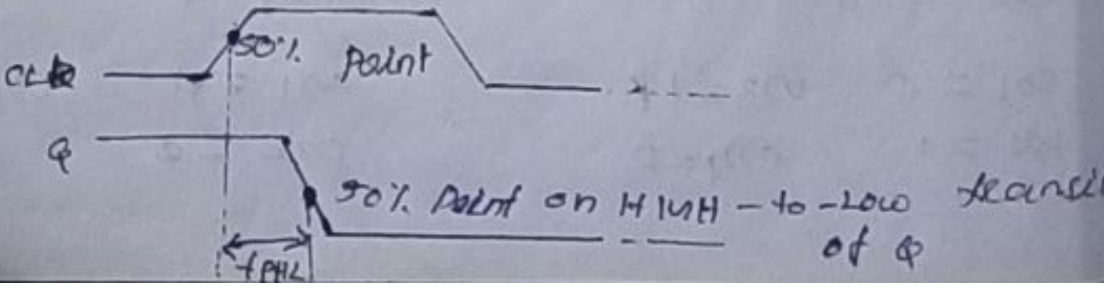
Propagation delay time is the interval time required after an i/p signal has been applied for the resulting o/p change to occure.

4 categories of propagation delay times are important in the operation of a ff.:

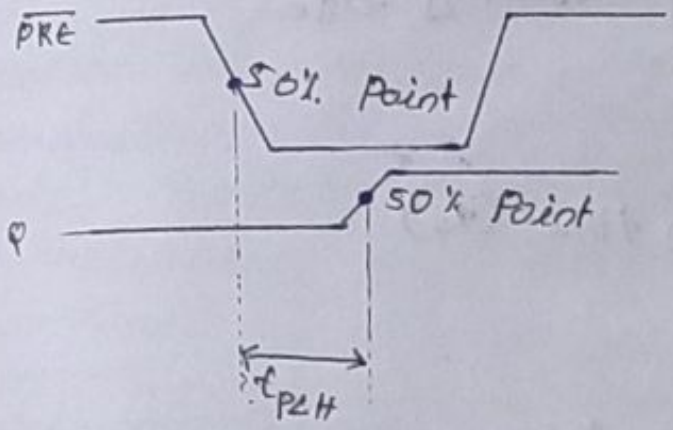
* Propagation delay $t_{PLH}$ as measured from the triggering edge of the clock pulse to the Low-to-High transition of the output.



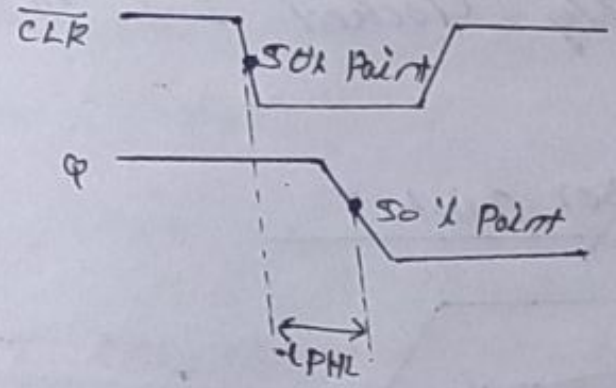

* Propagation delay $t_{PHL}$ as measured from the triggering edge of the clock to the High-to-Low transition of the o/p



* Propagation delay $t_{PLH}$ as measured from the leading edge of the present i/p to the LOW-to-HIGH transition of the o/p.

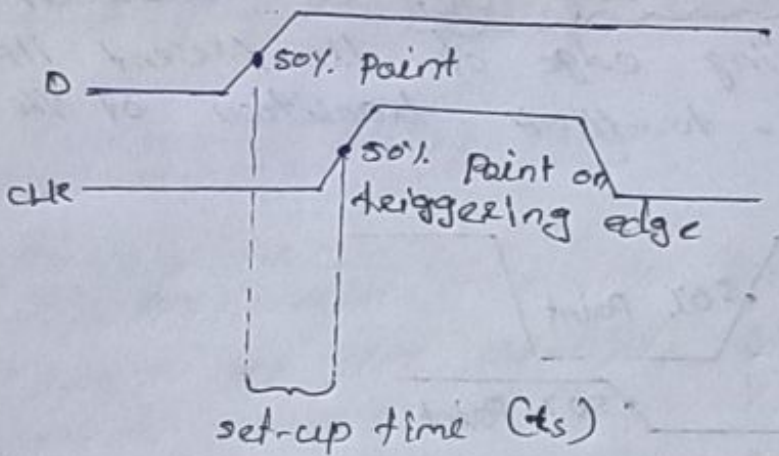



* Propagation delay $t_{PHL}$ as measured from the leading edge of the clear i/p to the HIGH-to-LOW transition of the o/p.



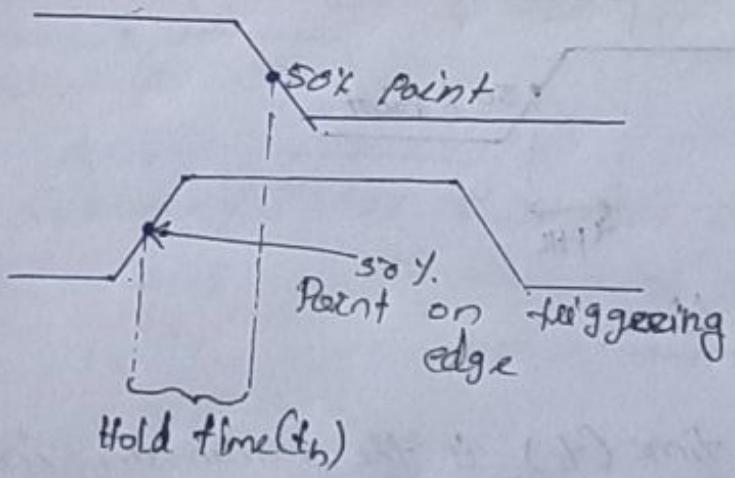

Set-up time:

The set-up time ($t_s$) is the minimum interval required for the logical levels to be maintained constantly on the inputs (J and K or D) prior to the triggering edge of the clock pulse in order for the levels to be reliably clocked into the FF.




## Hold time:

The hold time ($t_h$) is the minimum interval required for the logic level to remain on the inputs after the triggering edge of the clock pulse in order for the levels to be reliably clocked into the FF.




## Maximum clock frequency

The maximum clock frequency ($f_{max}$) is the highest rate at which a FF can be reliably triggered. At clock frequencies above the maximum, the FF would be unable to respond

quickly enough, and its operation would be impaired.

Pulse widths:

minimum pulse width ($t_w$) for reliable operation are usually specified by the manufacture for the clock, present, and clear inputs. Typically, the clock is specified by its minimum HIGH time and its minimum LOW time

Power dissipation:

The Power dissipation of any digital circuit is the total power consumption of the device. for example, if the IC. operates on a +5V dc source and draws 5 mA of current, the Power dissipation is

$$P = V_{cc} \times I_{cc} = 5V \times 5mA = 25mW$$

## SHIFT REGISTERS

Shift registers are a type of sequential logic circuits closely related to digital counters,

Registers are used primarily for the storage of digital data and typically do not possess a characteristic internal sequence of states as do counters.

Shift registers consist of arrangements of flip-flops which are used to store and are important in applications involving the storage and transfer of data in digital system.

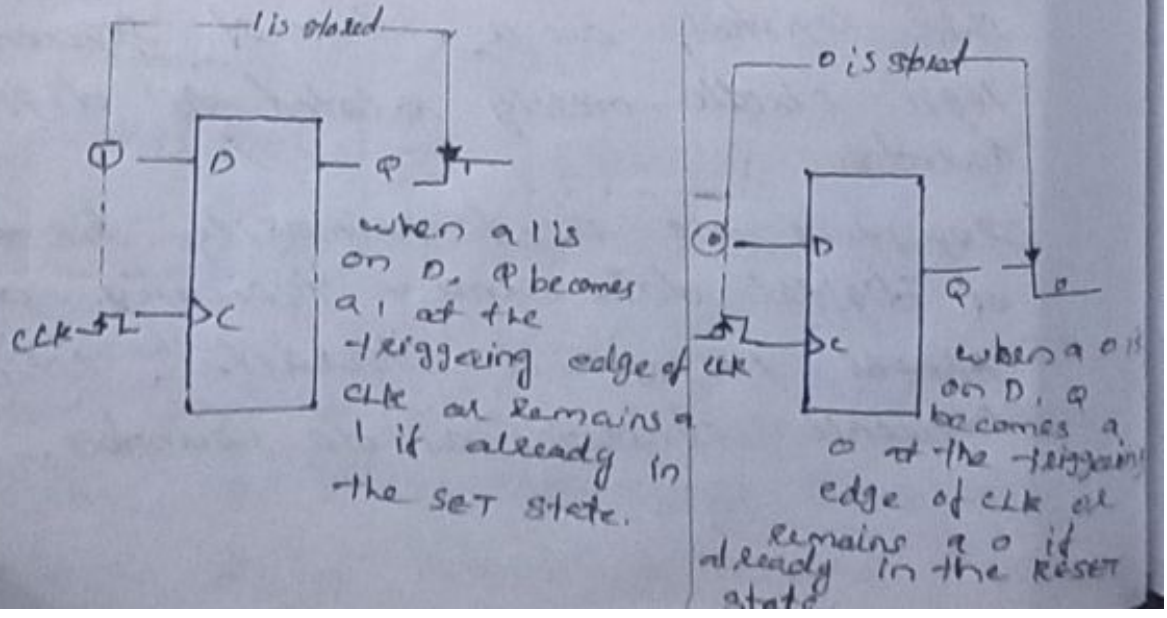
A registers, unlike a counter, has no specified sequence of states, except in certain very specific specialised application.

A register, in general, is used solely for storing and shifting data (1s & 0s) entered into it from a external source and typically possesses no characteristics internal sequence of states.

A register is a digital circuit with 2 basic functions: data storage and data movement.

It consists of one or more flip flops which are used to store & shift data.

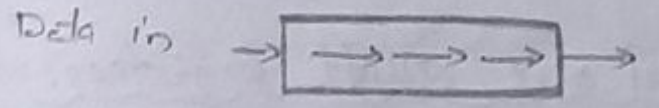
The storage capacity of registers makes it an important type of memory device.



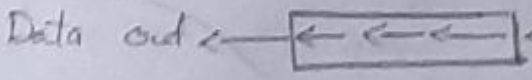
- The storage capacity of a register is the total number of bits (1s and 0s) of digital data it can retain.
- each stage (FF) in a shift register represents one bit of storage capacity; therefore, the number of states in a register determines its storage capacity.

- The shift capacity of a register permits the movement of data from stage to stage within the register or into or out of the register upon appl°. of clock pulse.

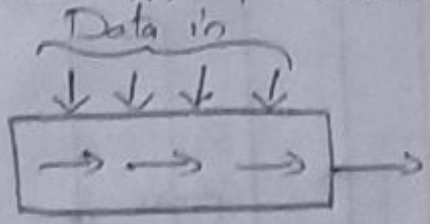
- Serial in / shift right / ~~Par~~ Serial out.



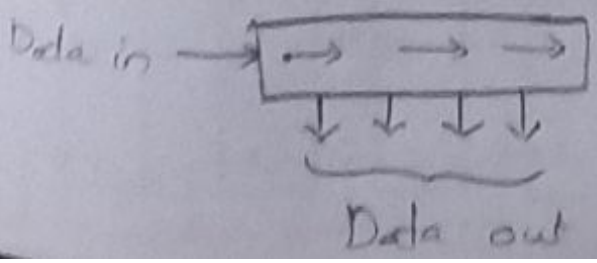

- Serial in / shift left / Serial out



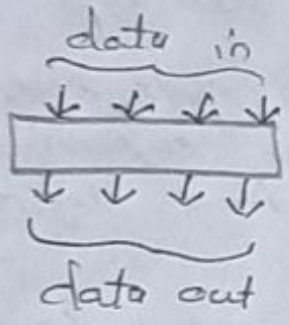

- Parallel in / serial out



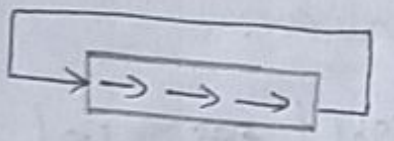

- Serial in / parallel out



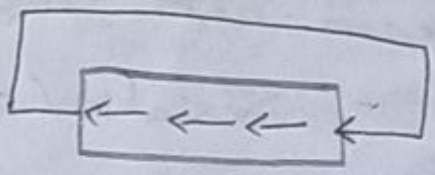
- Parallel in / parallel out




- Rotate right

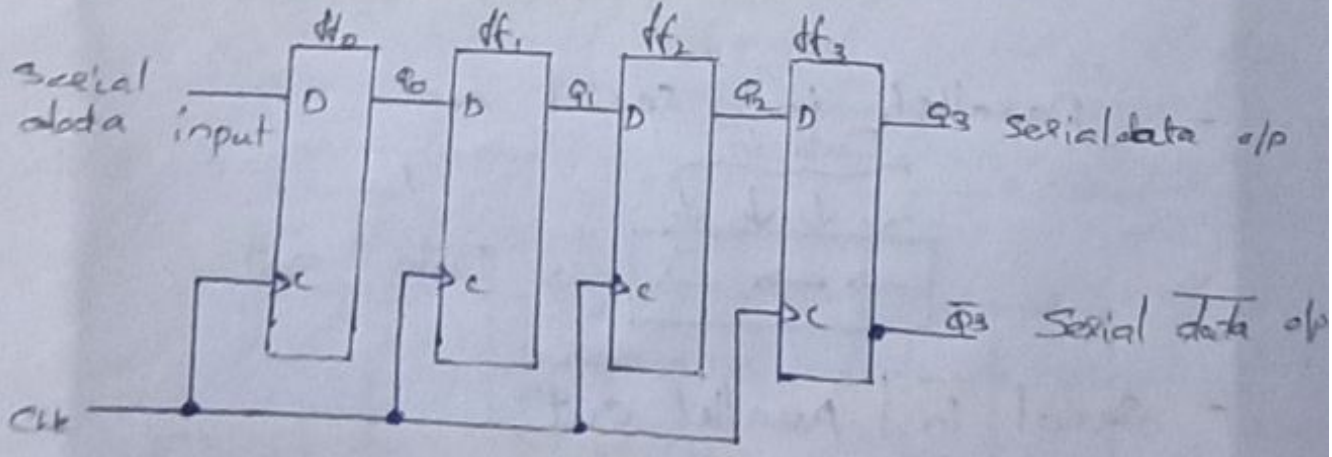


- Rotate left

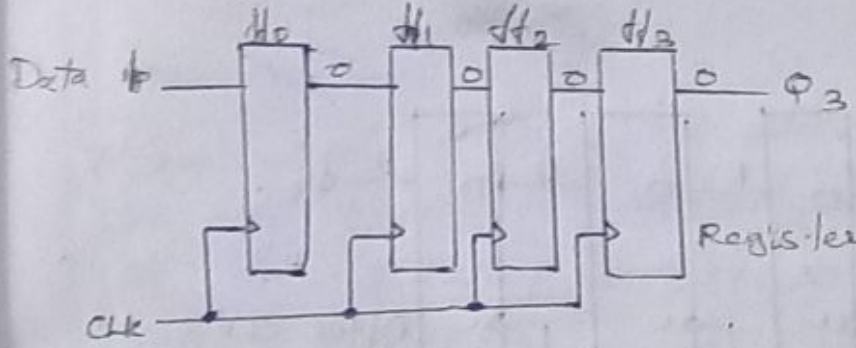


## Serial In / Serial out shift Registers:

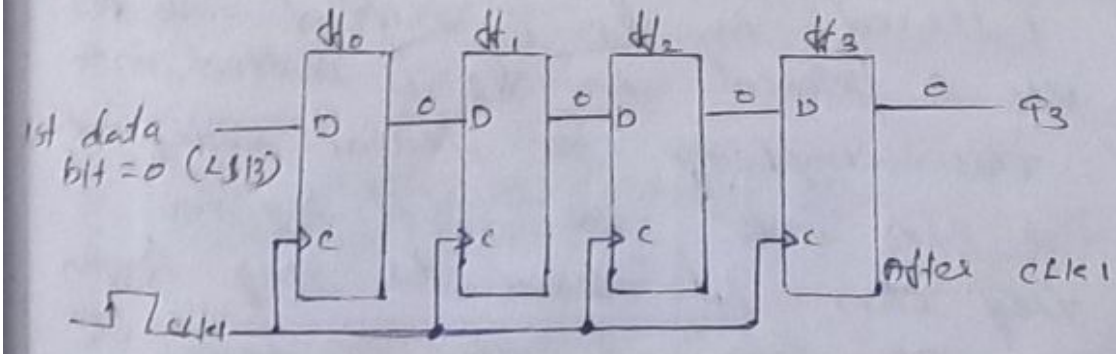
The serial in/serial out shift register accepts data serially - that is, one bit at a time on a single line. It produces the stored information on its output also in serial form

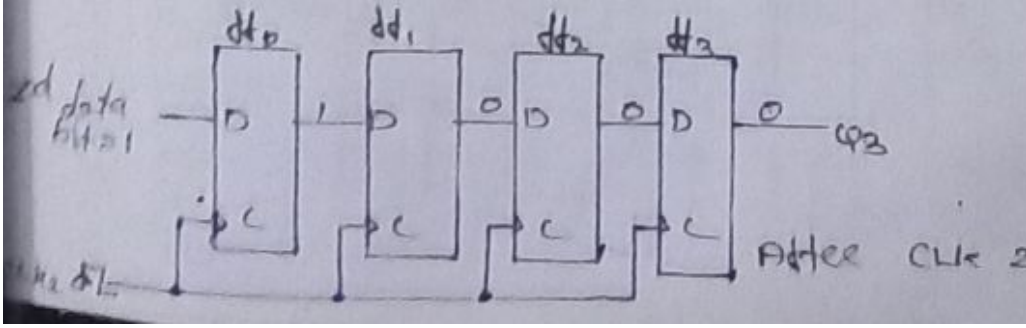


- The following figure illustrates entry of the four bits 1010 into the register, beginning with the level significant bit. The register is initially clear. The 0 is put onto the data input line, making D=0 for $ff_0$. When the first clock pulse is applied, $ff_0$ is reset, thus storing the 0.

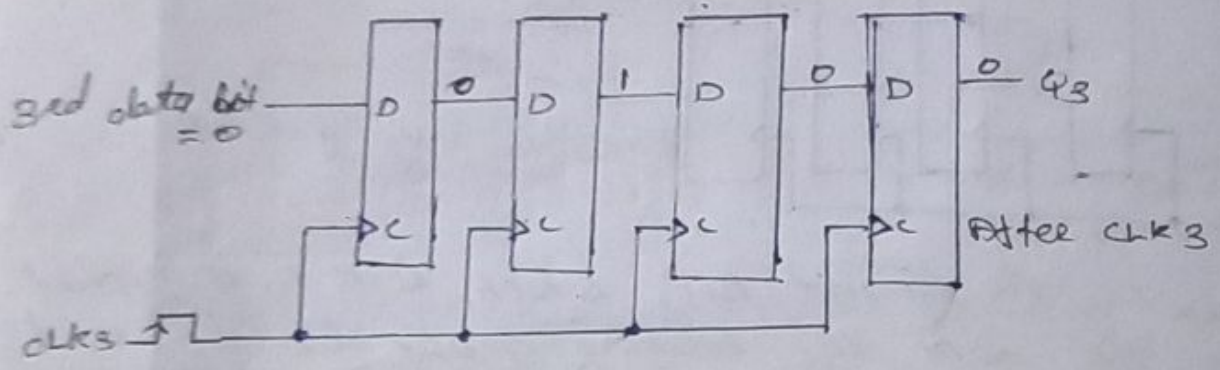



Next the second bit, which is a 1, is applied to the data i/p, making D=1 for $ff_0$ and D=0 for $ff_1$ because the D input of $ff_1$ is connected to the $Q_0$ output. When the second clock pulse occurs, the 1 on the data i/p is shifted into $ff_0$, causing $ff_0$ to set; and the 0 that was in $ff_0$ is shifted into $ff_1$.

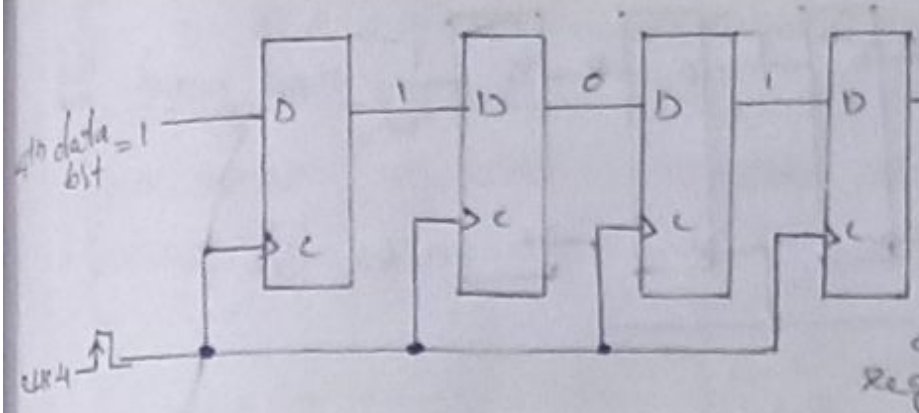


The third bit, a 0, is now put onto the data-input line, and a clock pulse is applied. The 0 is entered into $ff_0$, the 1 stored in $ff_0$ is shifted into $ff_1$, ~~the 0 is~~ and the 0 stored in $ff_1$ is shifted in to $ff_2$.



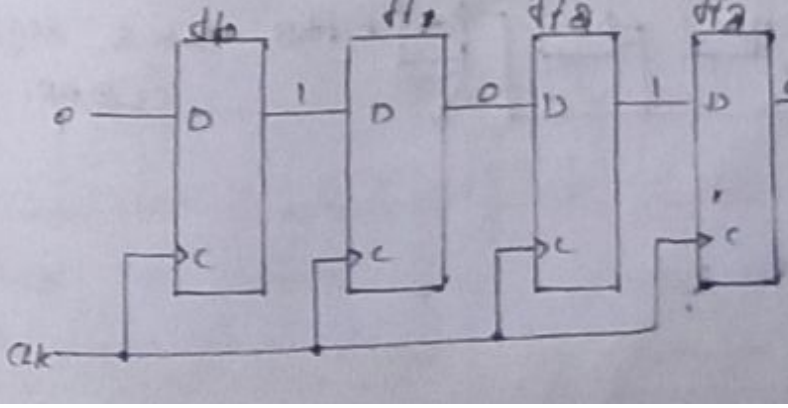

The last bit, a 1 is now applied to the data input, and a clock pulse is applied. This time the 1 is entered into $ff_0$, the 0 stored in $ff_0$ is shifted into $ff_1$. the 1 stored in $ff_1$ is shifted into $ff_2$. and the 0 stored in $ff_2$ is shifted into $ff_3$. This completes the serial entry of the 4 bits into the shift register. where they can be stored for any length of time as long as the $ff_s$ have dc power.

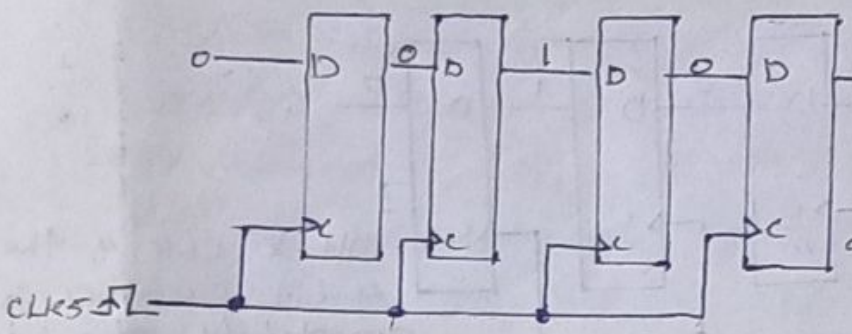


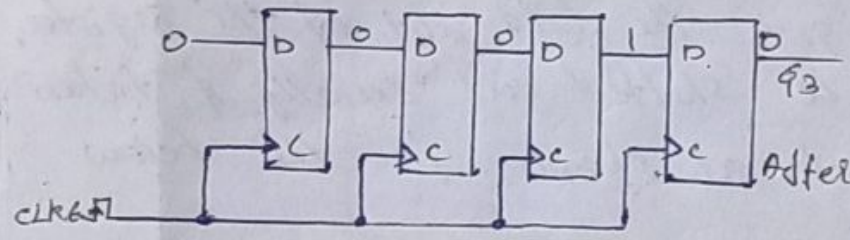

If we want to get the data out of the register, the bits must be shifted out serially & taken off the Q3 output, as shown in the below given figure.

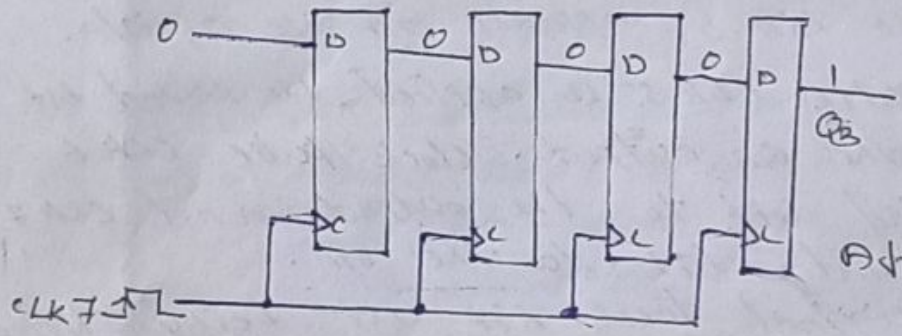
After clk 4 in the data-entry operation just discribed, the LSB i.e; appears on the Q3 o/p. when clock pulse clk 5 is applied, the second bit appears on the Q3 output. clock pulse clk 6 shift the third bit to the output, and clk 7 shifts the fourth-bit to the o/p.

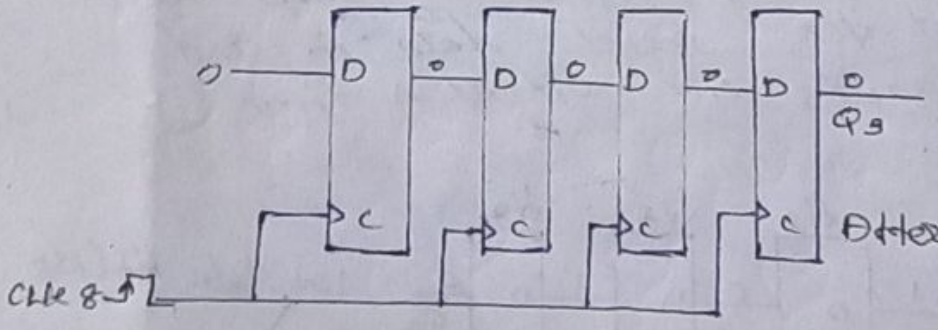
while the original four bit are being shifted out, more bit can be shifted in. All zeros are shown being shifted in. Thus we observe that for serial data in and serial data out, one bit at a time is transferred.



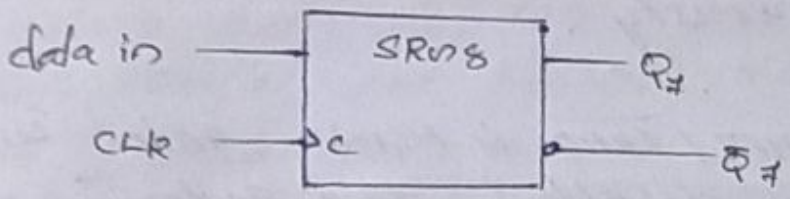

0
D
0
D
1
D
0
D
1
Q3.
2nd data bit
after CLK 5
CLK5


0
D
0
D
0
D
1
D
0
Q3
3rd data bit
After CLK 6
CLK6


0
D
0
D
0
D
0
D
1
Q3
4th data bit (MSB)
After CLK 7
CLK7

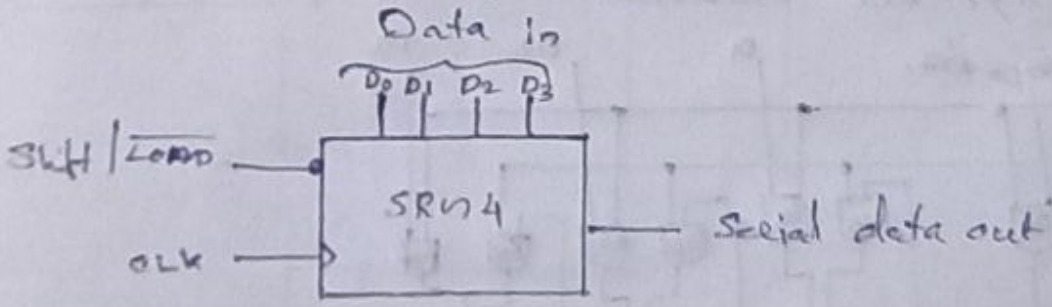

0
D
0
D
0
D
0
D
0
Q3
After CLK 8, register is CLEAR.
CLK 8

A ~~typical~~ traditional logic block symbol for an 8-bit serial in/serial out shift register is shown in below. The "SRG 8" designation indicates a shift register (SRG) is an 8-bit capacity.




## Parallel In / Serial out SRG:

For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs. The serial out is same as in serial in/ serial out SRG, once the data are completely stored in the register.

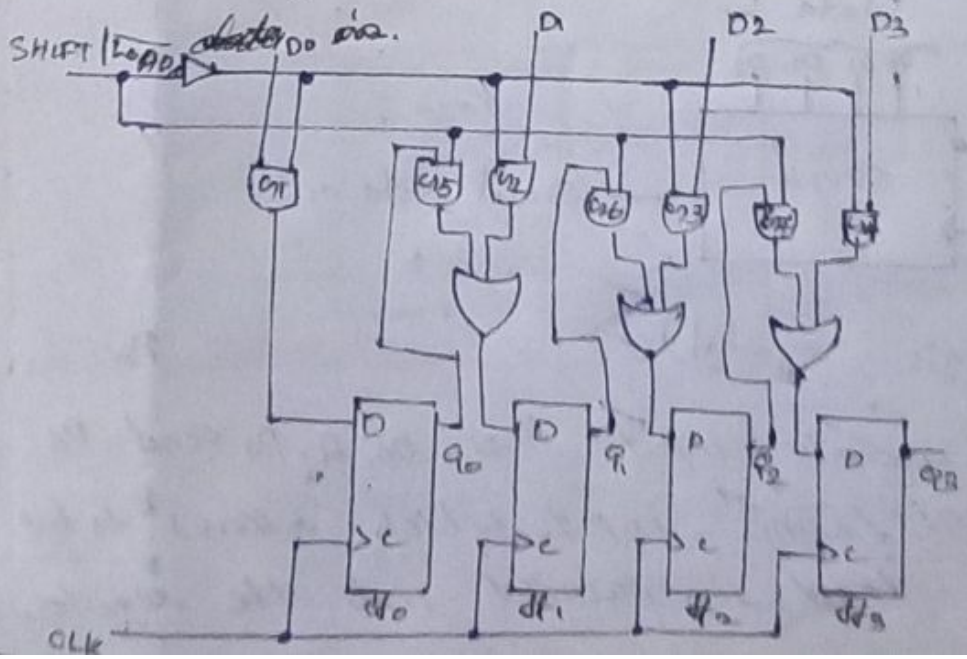



(Logic Symbol).

There are 4 'data' input lines, $D_0$, $D_1$, $D_2$ and $D_3$ and a SHIFT / $\overline{LOAD}$ input, which allows 4 bit of data to load in parallel into the register.

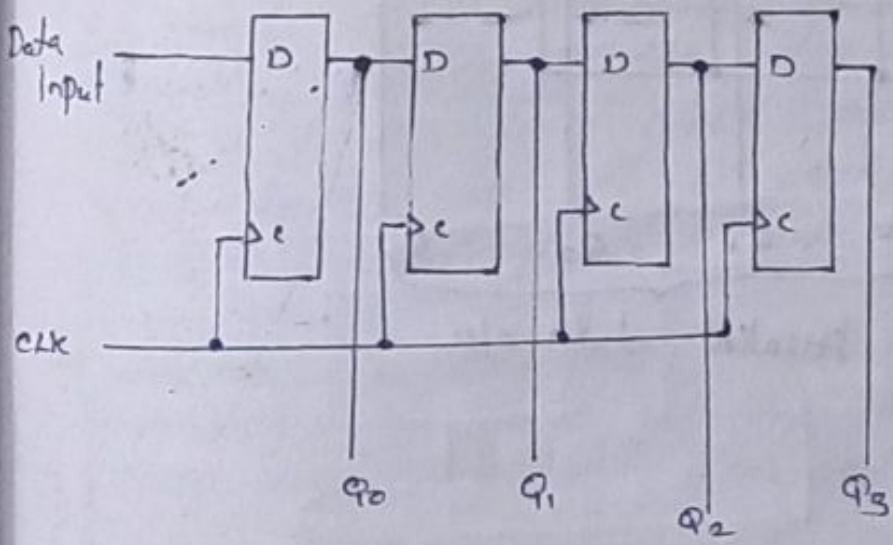
When SHIFT/$\overline{LOAD}$ is LOW, gates G1 through G4 are enabled, allowing each data bit to be applied to the D input of its respective FF. When a clock pulse is applied, the FF with D = 1 will set and those with D = 0 will reset, thereby storing all four bits simultaneously.

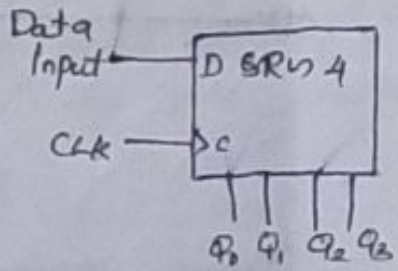
When SHIFT/$\overline{LOAD}$ is HIGH, gates G1 through G4 are disabled and gates G5 through G7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the SHIFT/$\overline{LOAD}$ input. Notice that FF0 has a single AND to disable the parallel input, D0. It does not require an AND/OR arrangement because there is no serial data in.

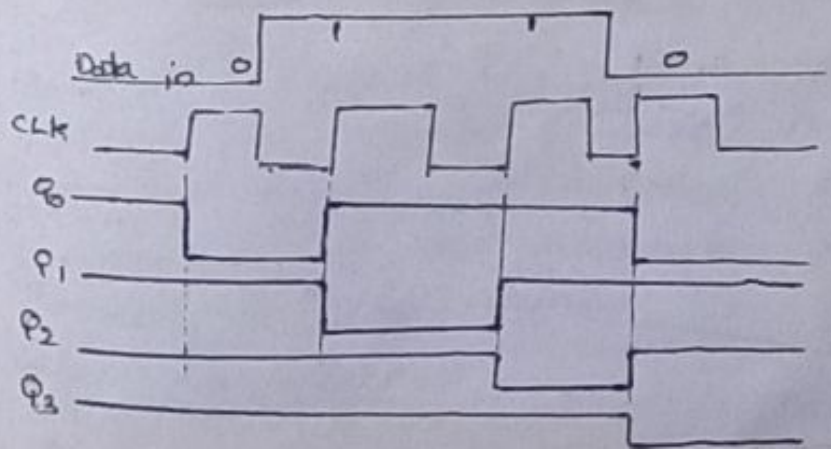


## Serial in / Parallel out SRn :

Data bit are entered serially (left significant bit first) into a serial in / parallel out SRn in same manner as in serial in / serial out registers. The difference is the way in which the data bits are taken out of the register; in the parallel out register, the o/p of each stage is available. Once the data are stored, each bits appears on its respective output line, and all bits are available simultaneously rather than on a bit-by-bit basis as with the serial out.

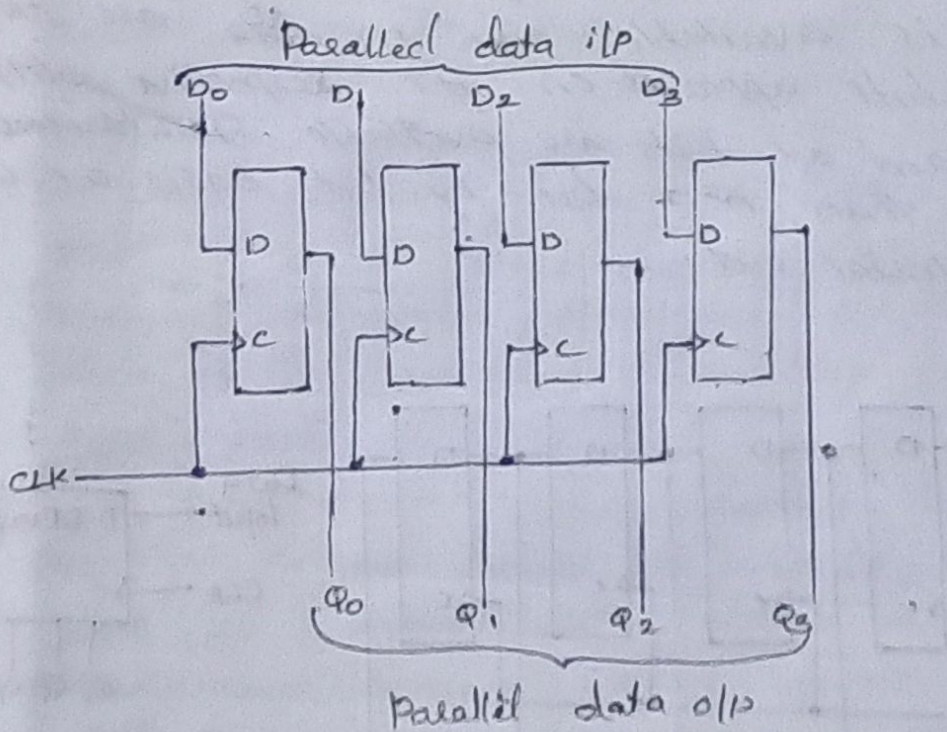


## Parallel in | Parallel out SRG :

Parallel entry and parallel output of data have been discussed. The parallel in/parallel out register employs both methods. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.



## COUNTERS

flip flop can be connected together to perform counting operations. Such a group of FFs is a counter, which is a type of finate state machine. The number of flip-flops used and the way in which they are connected determine the number of states (called the modulus) and also a specific sequence of states that the counter goes through during each complete cycle.

counters are classified into two broad categories according to the way they are clocked: asynchronous & synchronous.

A asynchronous counter, commonly called ripple counter, the first flip flop is clocked by the external clock pulse and then each successive FF is clocked by the output of the preceding flip flop.

In Syncheonous counters, the clock input is connected to all of the flip flops so that they are clocked simultaneously.

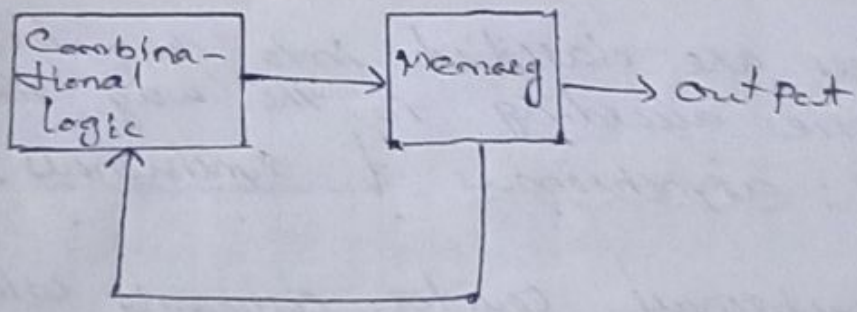
### finate state Machines:

A state machine is a sequential circuit having a limited (finate) number of states occuring in a prescribed order.
A counter is an example of a state machine; the number of states is called the modulus.

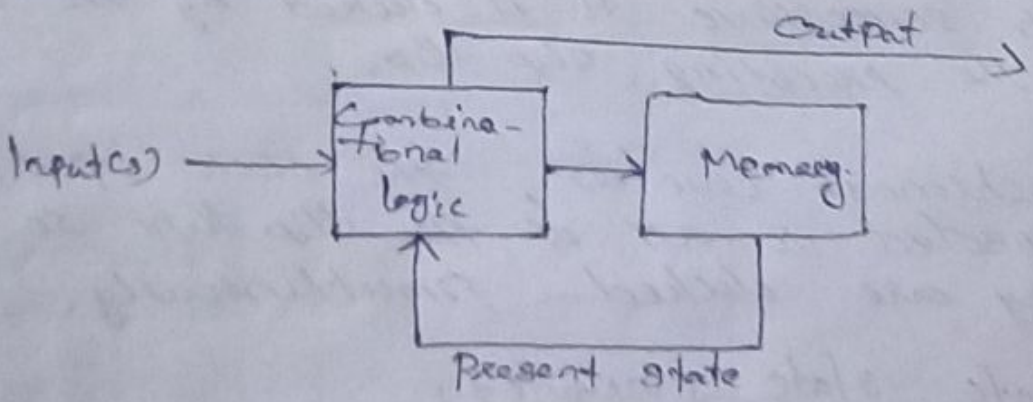
Two basic type of state machines are the Moore & the mealy.

Moore state machine is one where the outputs depend only on the internal Present state.

mealy state machine is one where the outputs depend on both the internal present state and on the inputs.




(Moore machine)




(Mealy Machine)

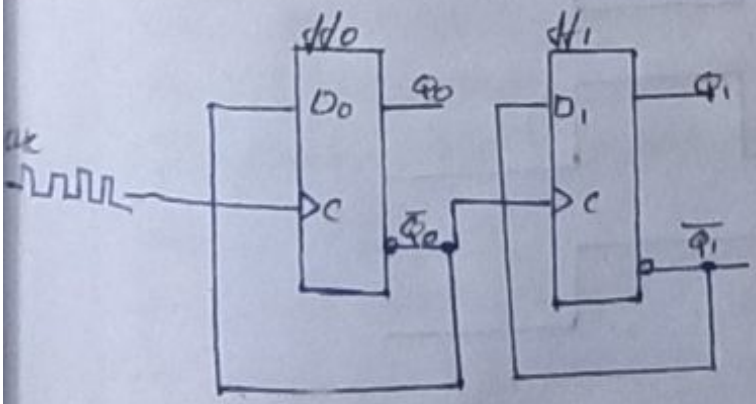
## => Asynchronous Counters:

The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time.

An Asynchronous counter is one in which the FF within the counter do not change states at exactly the same time because they do not have a common clock pulse.

The clock input of an asynchronous counter is always connected only to the LSB flip-flop.

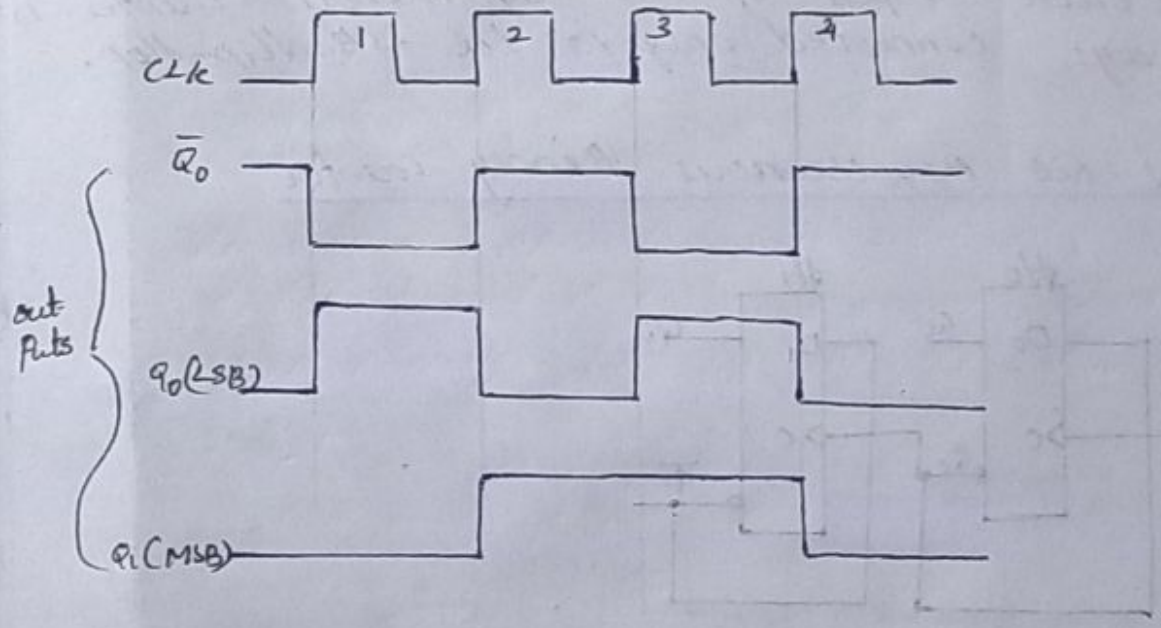
### A 2-bit Asynchronous Binary Counter




The given figure shows a 2-bit binary asynchronous counter connected for a asynchronous operation. The clk is applied to the clock i/p (↓) of only the first flip-flop, FF0 which is always the least significant bit (LSB).

Second flip-flop, FF1, is triggered by the $\overline{Q_0}$ o/p of FF0. FF0 changes state at the positive going edge of each clock pulse, but FF1 changes only when triggered by a positive-

going transition of the $\bar{Q}_0$ output of FF0. Because of the inherent propagation delay time through a flip flop, a transition of the i/p clk and a transition of the $\bar{Q}_0$ o/p of FF0 can never occure at exactly the same time. Therefore, the two flip-flop are never simultaneously triggered, so the counter operation is asynchronous.




Applying 4 clock pulses to FF0 and observing the Q output of each FF.

Both flip-flops are connected for toggle operation ($D = \bar{Q}$) and are assumed to be initially RESET (Q LOW).

The positive going edge of CLK1 causes the Q0 output of FF0 to $Q_0$ HIGH.

At the same time $\bar{Q}_0$ o/p goes low, but it has no effect on FF1 because a positive-going transition must occure to trigger the FF. After the leading edge of CLK1, $Q_0 = 1$ and $Q_1 = 0$. The positive-going edge of CLK2 causes ~~leading edge of~~ after, $Q_0$ to go low. output $\bar{Q}_0$ goes HIGH and triggers FF1, causing $Q_1$ to go HIGH. After the leading edge of CLK2, $Q_0 = 0$ and $Q_1 = 1$.

The positive-going edge of CLK 3 causes $Q_0$ to go HIGH again. o/p $\bar{Q}_0$ goes Low and has no effect on FF1. Thus, after the leading edge of CLK3, $Q_0 = 1$ and $Q_1 = 1$. The positive-going edge of CLK 4 causes $Q_0$ to go low, while $\bar{Q}_0$ goes HIGH and triggers FF1, causing $Q_1$ to go Low.

After leading edge of CLK4, $Q_0 = 0$ and $Q_1 = 0$. The counter has now recycled to its original state. (Both FFs are RESET).

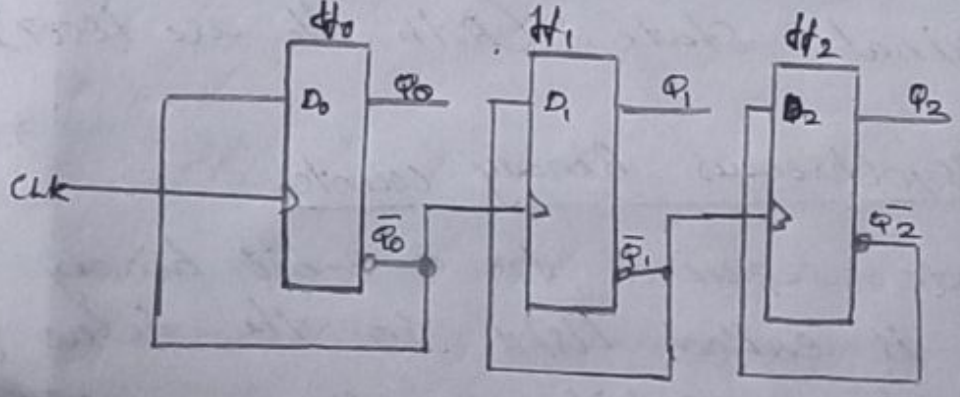
## 3 bit Asynchronous Binary counter

The state sequence for a 3-bit binary counter is ~~called~~ listed in the below given table and a 3-bit asynchronous binary counter is shown in below figure.

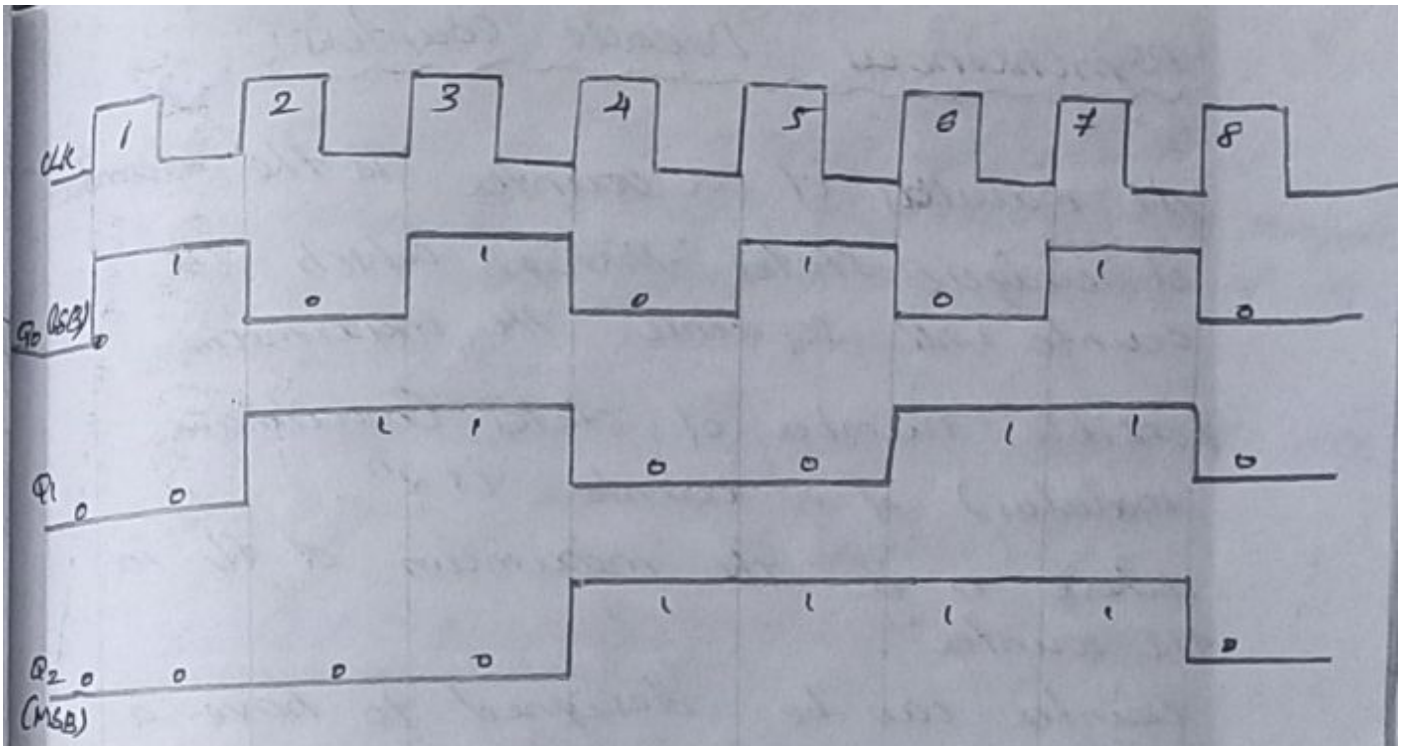
State sequence for a 3-bit binary counter

| Clock Pulse | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| Initially | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8(recycles) | 0 | 0 | 0 |

The basic operation is the same as that of the 2-bit counter except that the 3bit counter has 8 states, due to its 3 FFs.




A timing diagram is shown in below.




## Propagation delay:

Asynchronous counters are commonly refferred to as ripple counters for following reason:

The effect of the input clock pulse is first "felt" by FF0. This effects cannot get to FF1 immidiatly because of the propagation delay through FF0.

Then there is a propagation delay through FF1 before FF2 can be triggered. Thus, the effect of an input clock pulse "ripples" through the counter, taking some time, due to propagation delays, to reach the last flip-flop.

## Asynchronous Decade counters:

The modulus of a counter is the number of unique states through which the counter will sequence. The maximum possible number of states (maximum modulus) of a counter is $2^n$. where n is the maximum of flip flops in the counter.

counter can be designed to have a number of states in their sequence that is less than the maximum of $2^n$. This type of sequence is called a truncated sequence.

one common modulus for counters with truncated sequence is ten (called MOD10).

counters with 10 states in their sequence are called Decade counters.

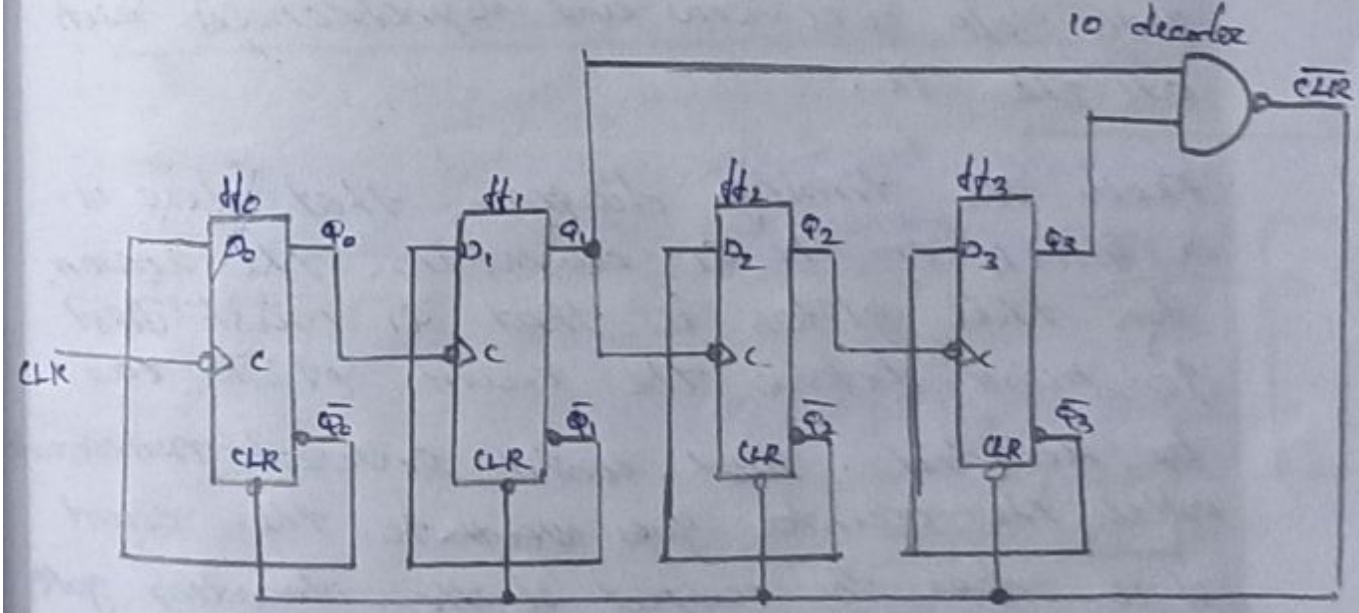
The decade counter with a count sequence of zero (0000) through nine (1001) is a BCD decade counter because its ten-state sequence produce the BCD counter. This type of counter is useful in display appl.n in which BCD is required for

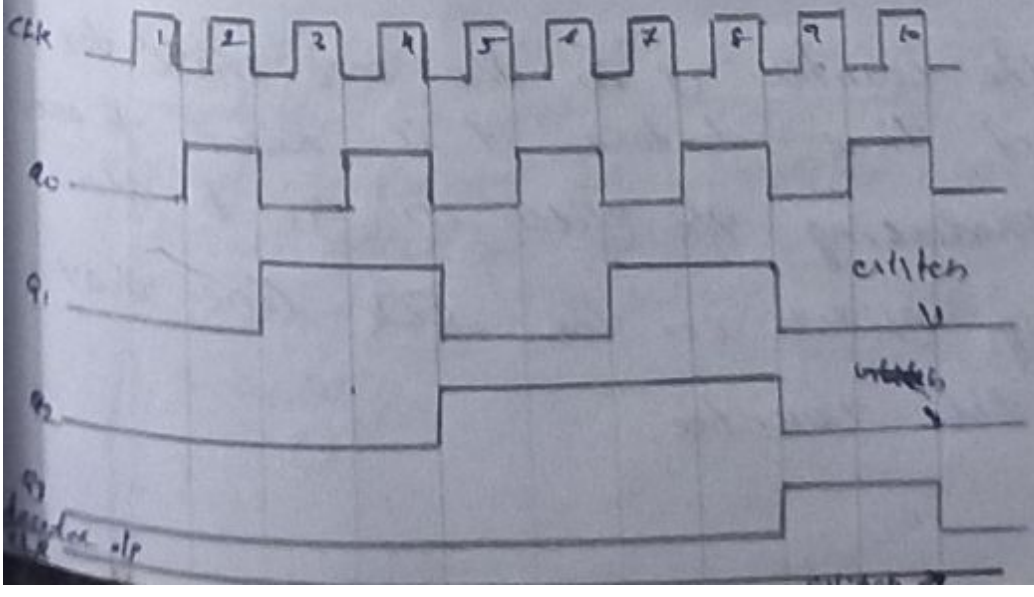
conversion to a decimal readout.

To obtain a truncated sequence, it is necessary to force the counter to recycle before going through all of its possible states.

A decade counter requires 4 FFs (3 FFs are insufficient because $2^3 = 8$)

## Partial Decoding :



In the following given figure, that only $Q_1$ & $Q_3$ are connected to the NAND gate i/p. This arrangement is an example of partial decoding, in which the 2 unique states ($Q_1 = 1$ and $Q_3 = 1$) are satisfied sufficient to decode the count of 10 because none of the other state (0 through 9) have both $Q_1$ & $Q_3$ HIGH at the same time. When the counter goes into count 10 (1010), the decoding gate o/p goes low and asynchronous resets all the FF.

From the timing diagram, that there is a glitch on the $Q_1$ waveform. The reason for this glitch is that $Q_1$ must first go HIGH before the count of 10 can be decoded. Not until several nanoseconds after the counter goes HIGH to the count of 10 does the output of the decoding gate go LOW (Both inputs are HIGH).

Thus the counter is in the 1010 state for a short time before it is reset to 0000, thus producing the glitch on $Q_1$ & the resulting glitch on the $\overline{CLR}$ line that reset the counter.

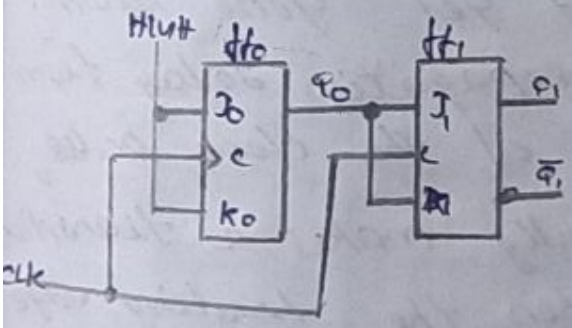
## => Synchronous Counters :

The term synchronous refers to events that have a fixed time relationship with each other.

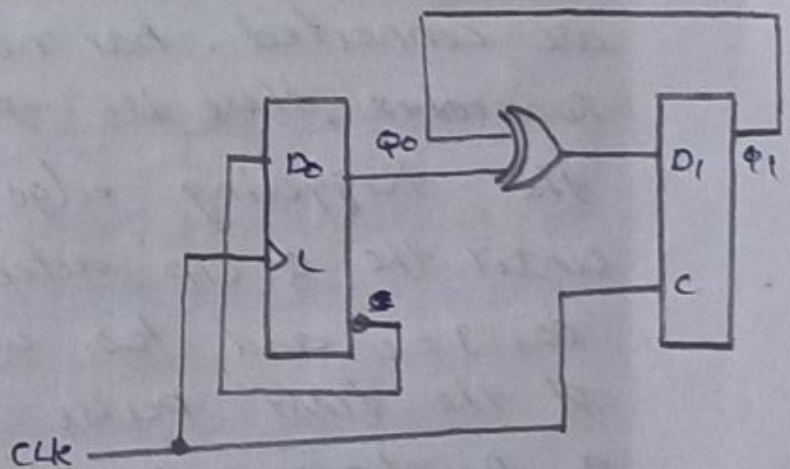
A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse.

J-K FFs are used to illustrate most synchronous counters.

## 2-Bit Synchronous Binary Counter




(JK FF)



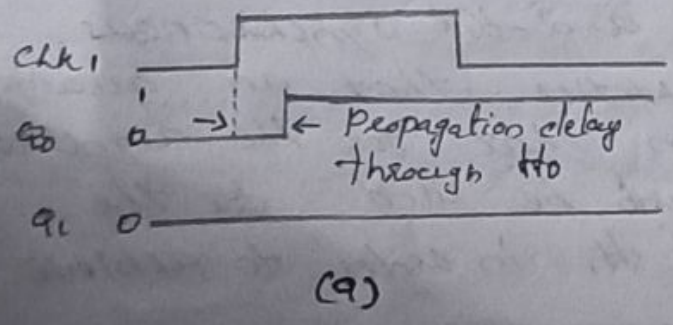

(D FF)

This figure shows a 2-bit synchronous binary counter. Notice that an arrangement different from that for the asynchronous counter must be used for the J1 and K1 I/P of FF1 in order to achieve a binary sequence.

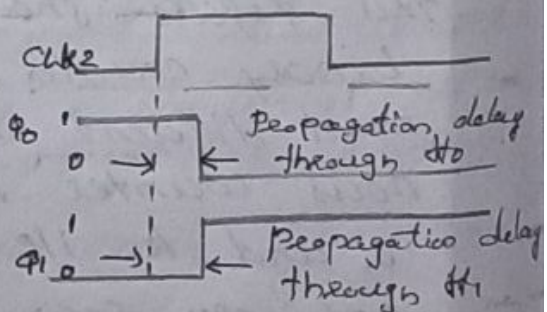
The operation of a J-K FF. Synchronous counter is as follows:

First, assume that the counter is initially in the binary 0 states; that is, both FF are RESET. When the positive edge of the first clock pulse is applied, FF0 will toggle and $Q_0$ will therefore go HIGH.

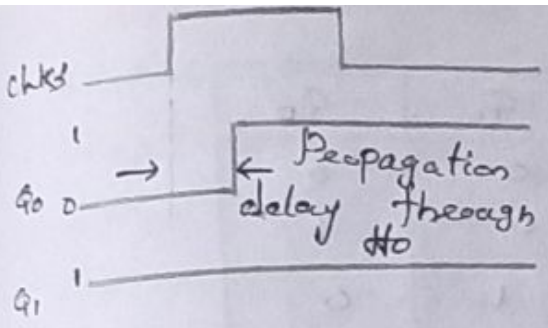
What happens to FF1 at the positive-going edge of CLK? To find out, let's look at the input conditions of FF1. Its J1 & K1 are both LOW because $Q_0$, to which they are connected, has not yet gone HIGH. Remember, there is propagation delay from the triggering edge of the clock pulse until the Q o/p actually makes a transition. So, J=0 and K=0 when the leading edge of the first pulse is applied. This is a no-change condition, and therefore FF1 does not change state



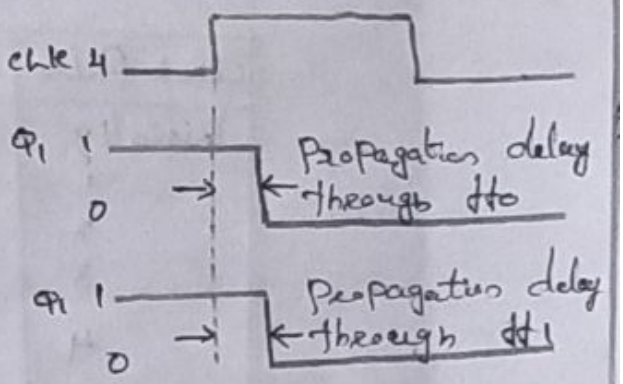

(a)




(b)



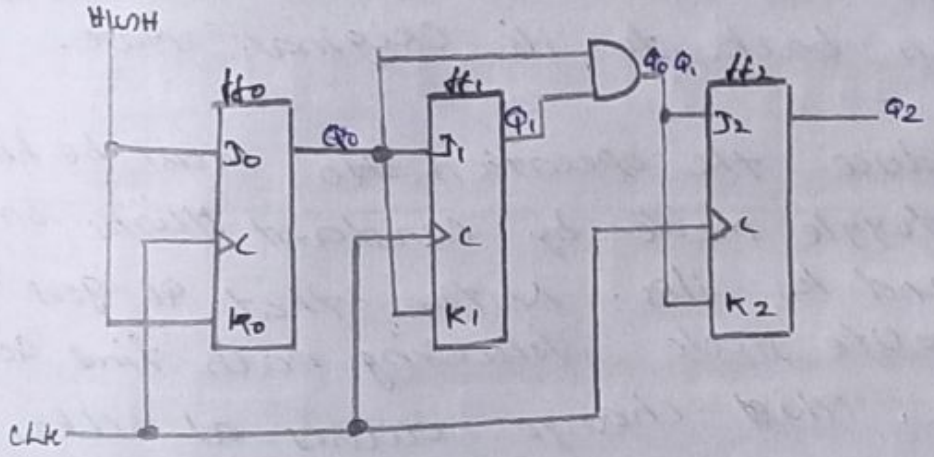

(c)

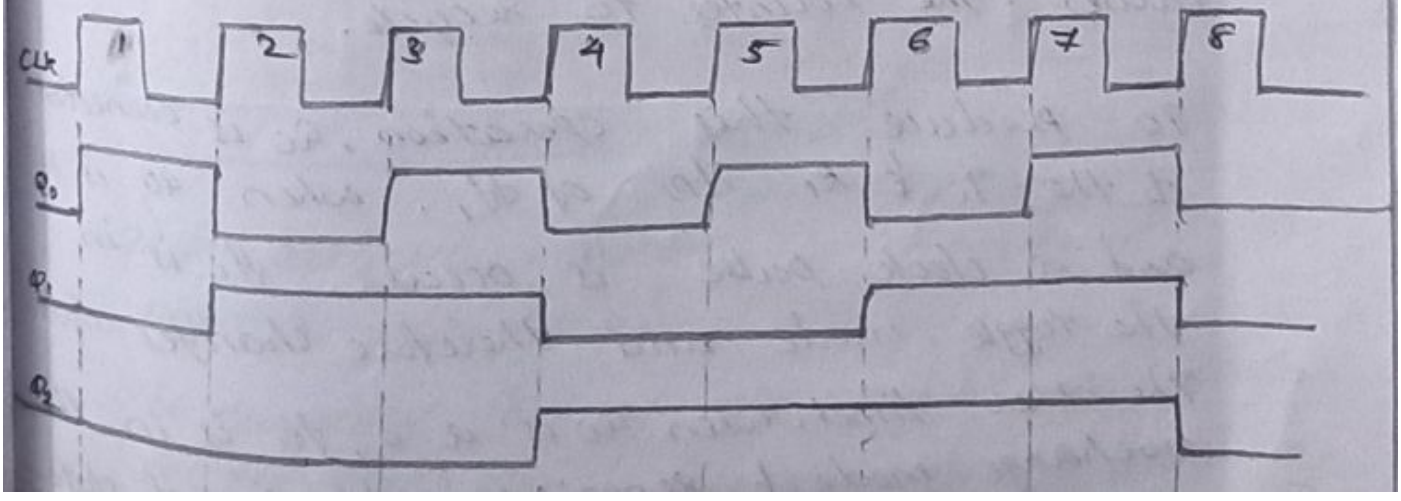



(d)

## 3bit Syncheronous Binary Counter

A 3-bit synchronous binary counter and it's timing diagram is shown in below.



| Clock Pulse | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| Initially | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 (Recycle) | 0 | 0 | 0 |

- First, let's look at $Q_0$. Notice that $Q_0$ changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state.

To produce the operation, $FF_0$ must be held in the toggle mode by constant HIGHs on its J0 and K0 i/Ps. Notice that $Q_1$ goes to the opposite state following each time $Q_0$ is a 1. This change occurs at clk 2, clk 4 clk 6 and clk 8. The clk 8 pulse causes the counter to recycle.

To produce this operation, $Q_0$ is connected to the $J_1$ & $K_1$ i/Ps of $FF_1$. When $Q_0$ is 1 and a clock pulse is occurs, $FF_1$ is in the toggle mode and therefore changes state. The other times, when $Q_0$ is a 0, $FF_1$ is in the no change mode & remains in its present state.

Next, let's see how $FF_2$ is made to change at the proper times according to the binary sequence.

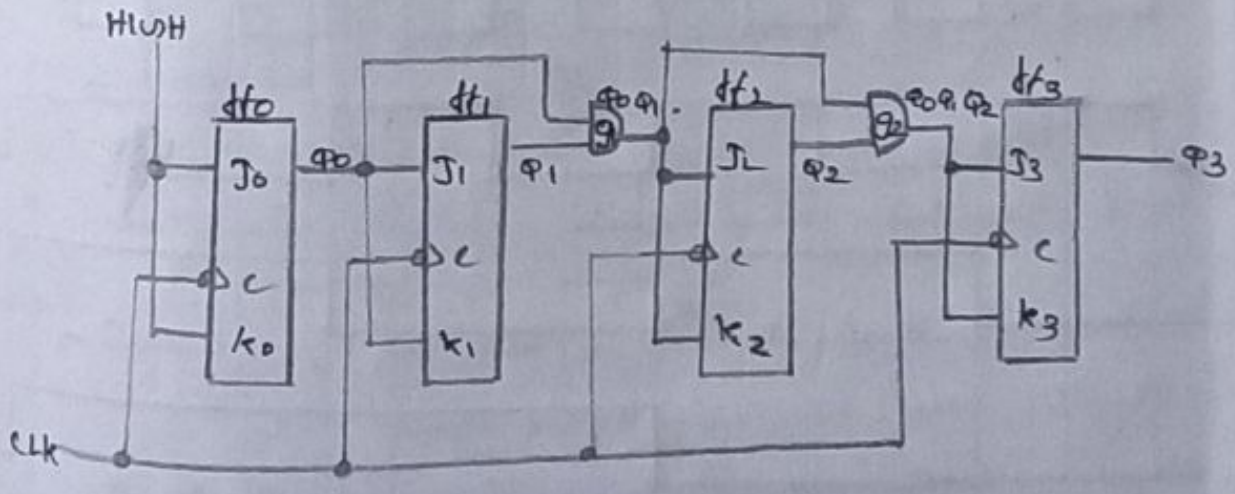
Notice that both times $Q_2$ changes state, it is preceded by the unique condition in which both $Q_0$ & $Q_1$ are HIGH.

This condition is detected by the AND gate and applied to the $J_2$ & $K_2$ inputs of $FF_2$.

Whenever both $Q_0$ & $Q_1$ are HIGH, the o/p of the AND gate makes the $J_2$ & $K_2$ inputs of $FF_2$ HIGH, and $FF_2$ toggles on the following clock pulse.

At all other times, the $J_2$ and $K_2$ inputs of $FF_2$ are held LOW by the AND gate output, and $FF_2$ does not change state.

## 4-bit Synchronous Binary Counter




This particular counter is implemented with negative edge-triggered flip-flops.

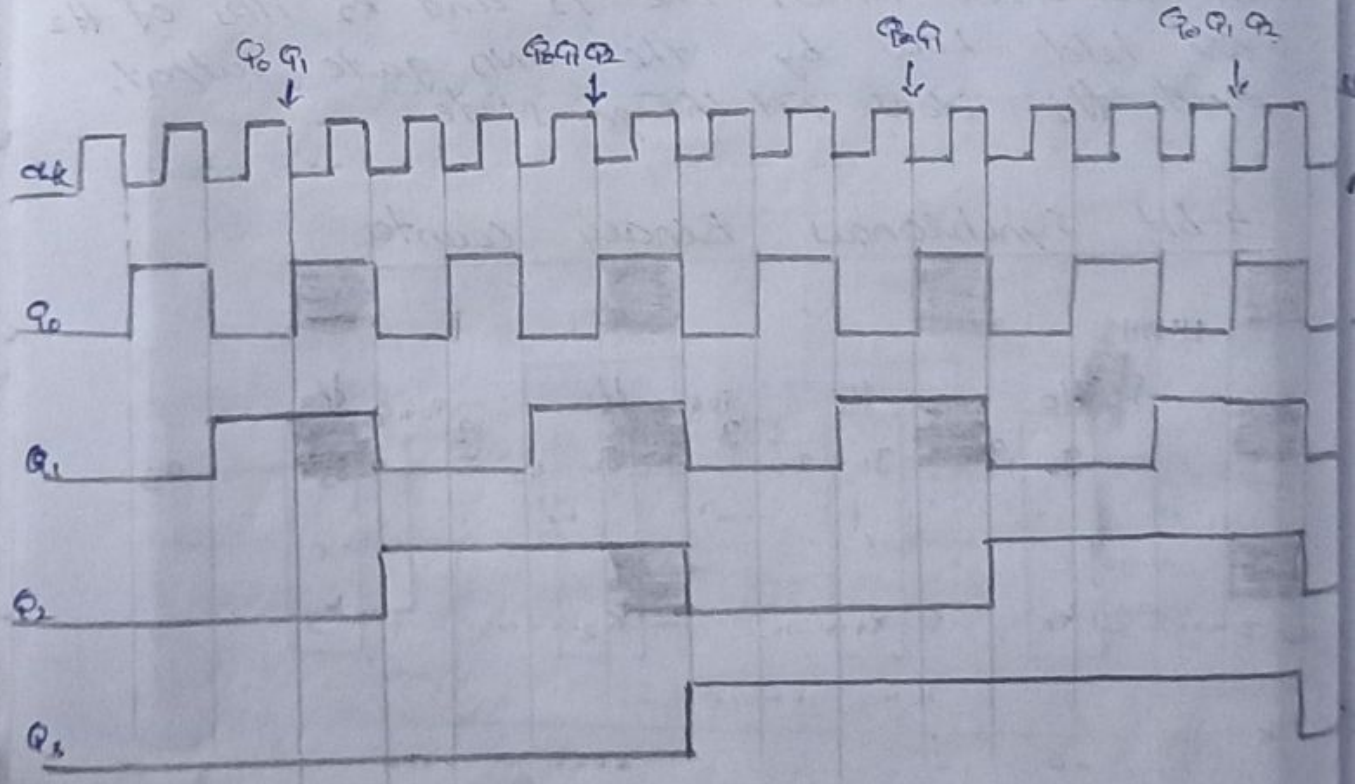
The reasoning behind the J & K input control for the first three flip-flops is the same as

3-bit counter.

The fourth stage, $FF_3$, changes only twice in the sequence. Notice that both of these transitions occur following the time that $Q_0$, $Q_1$ & $Q_2$ are all HIGH.

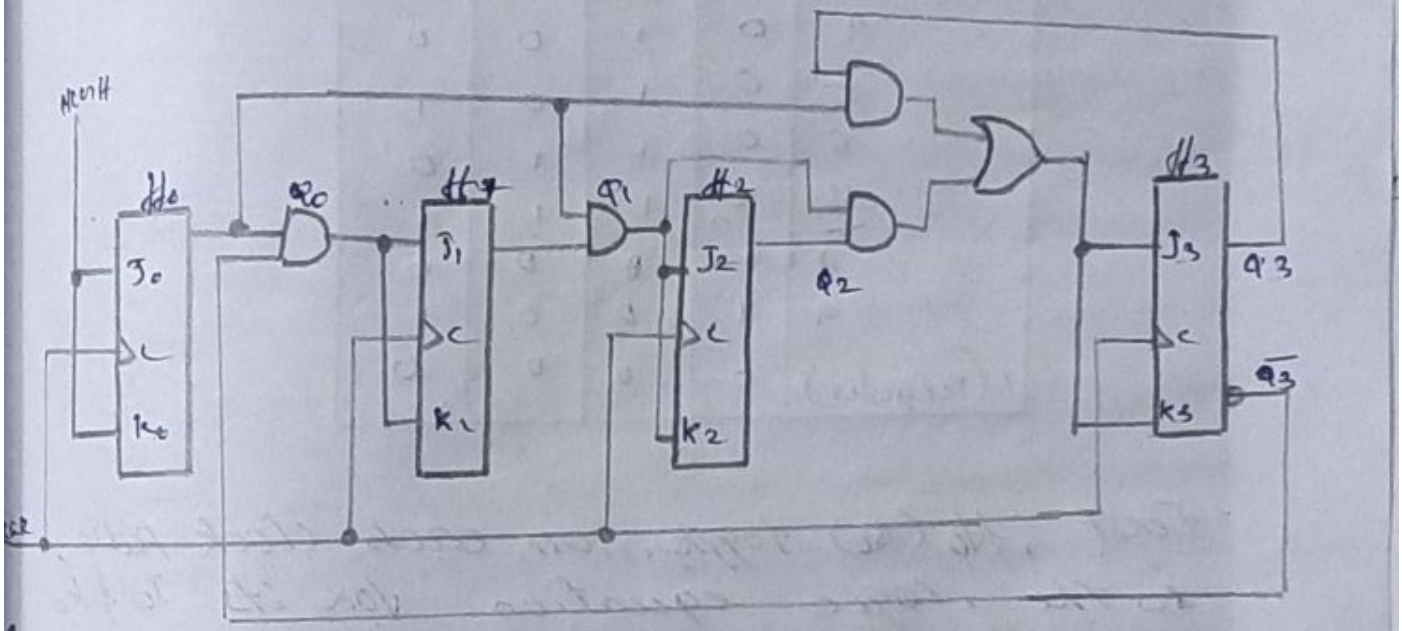
This condition is decoded by AND gate $G_2$ so that when a clock pulse occurs, $FF_3$ will change state.

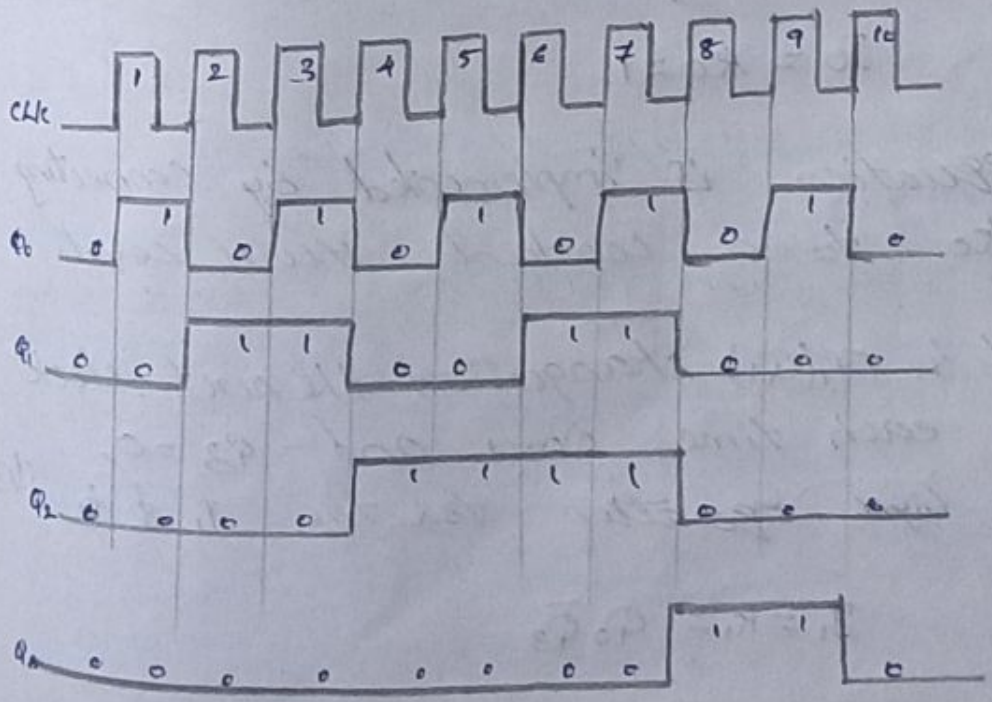
For all other times the $J_3$ & $K_3$ inputs of $FF_3$ are LOW, and it is in a no-change condition.



# 4-Bit Synchronous Decade Counter:

A BCD decade counter exhibits a truncated binary sequence and goes from 0000 through the 1001 state. Rather than going from the 1001 state to the 1010 state, it recycle to the 0000 state.



states of a BCD decade counter

| clock Pulse | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10(Recycles) | 0 | 0 | 0 | 0 |

- First, FF0 ($Q_0$) toggles on each clock pulse, so the logic equation for its $J_0$ & $k_0$ inputs is

$$J_0 = k_0 = 1$$

The equation is implemented by connecting $J_0$ & $k_0$ to a constant HIGH Level.

- Next, FF1($Q_1$) changes on the next clock pulse each time $Q_0 = 1$ and $Q_3 = 0$, so the logic equation for the $J_1$ & $k_1$ i/ps is

$$J_1 = k_1 = Q_0 \bar{Q}_3$$

This equation is implemented by ANDing $Q_0$ & $\bar{Q}_3$ and connecting the gate o/p to the

$J_1$ & $K_1$ inputs of $FF_1$.

- $FF_2$ ($Q_2$) changes on the next clock pulse each time both $Q_0 = 1$ and $Q_1 = 1$. This requires an I/P logic equation as follows:

$$J_2 = K_2 = Q_0 Q_1$$

This equation is implemented by ANDing $Q_0$ & $Q_1$ and connecting the gate o/p to the $J_2$ & $K_2$ inputs of $FF_2$.

- Finally, $FF_3$ ($Q_3$) changes to the opposite state on the next clock pulse each time $Q_0 = 1$, $Q_1 = 1$ and $Q_2 = 1$ (state 7) or when $Q_0 = 1$ & $Q_3 = 1$ (state 9). The equation for this is as follows:

$$J_3 = K_3 = Q_0 Q_1 Q_2 + Q_0 Q_3$$

This function is implemented with the AND/OR logic connected to the $J_3$ and $K_3$ inputs of $FF_3$.

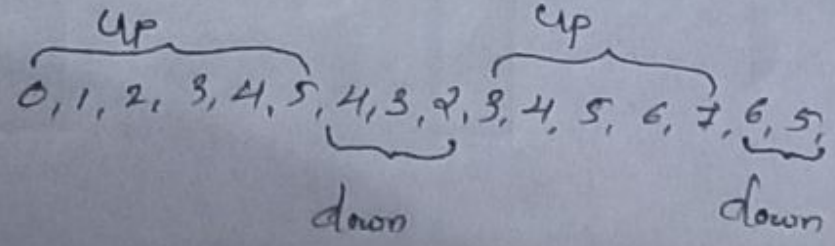
## Up/Down Synchronous Counters:

The up/down counter is one that is capable of progressing in either direction through a certain sequence.

An up/down counter, sometime called a bidirectional counter, can have any specified sequence of states.

A 3-bit binary counter that advance upward through its sequence (0,1,2,3,4,5,6,7) and then can be reversed so that it goes through the sequence in the opposit direction (7,6,5,4,3,2,1,0) is an illustration of up/down sequential operation.

In general, most up/down counters can be reversed at any point in their sequence: for instance, the 3-bit binary counter can be made to go through the following sequence:



up/down sequence for a 3-bit binary Counter

| Clock Pulse | up | $Q_2$ | $Q_1$ | $Q_0$ | Down |
|---|---|---|---|---|---|
| 0 | ↑ | 0 | 0 | 0 | ↓ |
| 1 | ↑ | 0 | 0 | 1 | ↓ |
| 2 | ↑ | 0 | 1 | 0 | ↓ |
| 3 | ↑ | 0 | 1 | 1 | ↓ |
| 4 | ↑ | 1 | 0 | 0 | ↓ |
| 5 | ↑ | 1 | 0 | 1 | ↓ |
| 6 | ↑ | 1 | 1 | 0 | ↓ |
| 7 | ↓ | 1 | 1 | 1 | ↓ |

- This table shows the complete up/down sequence for a 3-bit binary counter. The arrows indicate the state-to-state movement of the counter for both its up and its down modes of operation.
- An examination of $Q_0$ for both the up and down sequences shows that $ff_0$ toggles on each clock pulse. Thus, the $J_0$ & $k_0$ inputs of $ff_0$ are:

$$J_0 = k_0 = 1$$

- For the up sequence, $Q_1$ changes state on the next clock pulse when $Q_0 = 1$. For the down sequence, $Q_1$ changes on the next clock pulse when $Q_0 = 0$.

Thus, the $J_1$ & $k_1$ inputs of $ff_1$ must equal

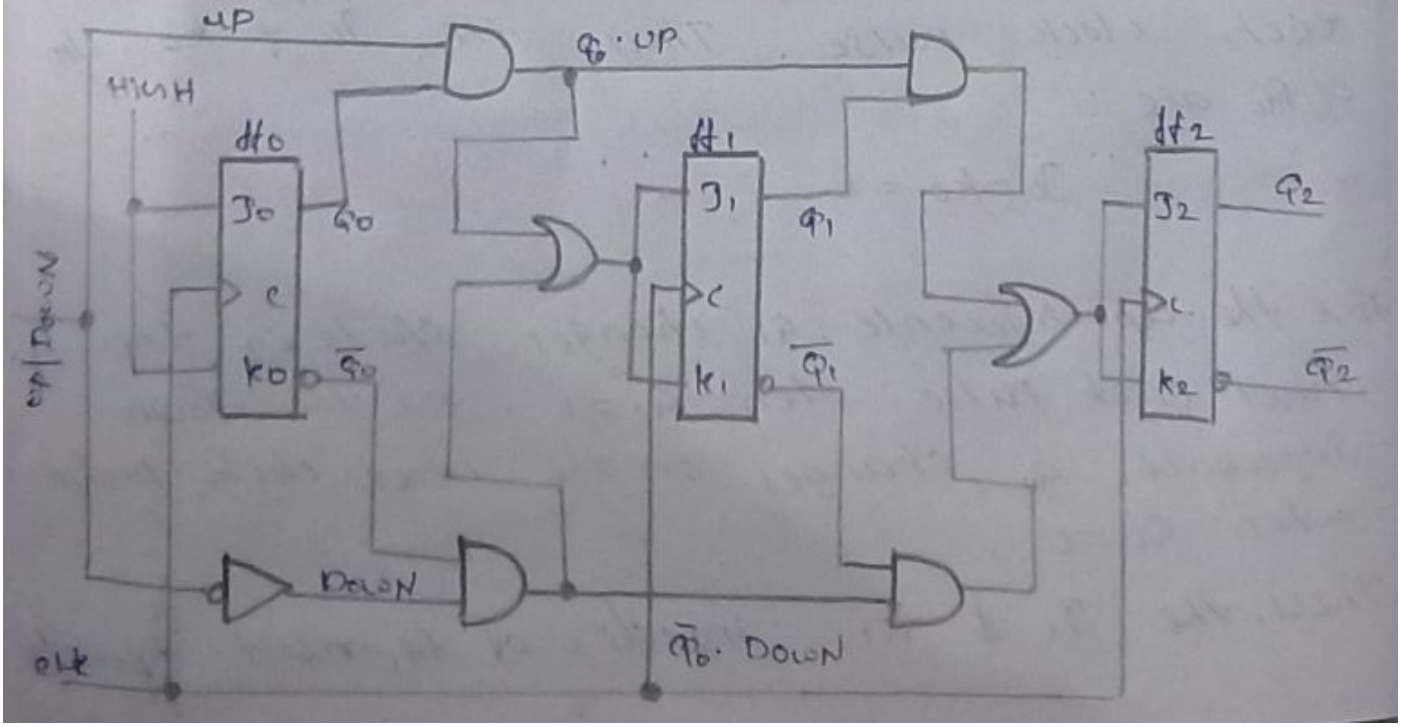
under the conditions expressed by the following equation:

$$J_1 = K_1 = (Q_0 \cdot UP) + (\bar{Q}_0 \cdot DOWN)$$

- For the up sequence, $Q_2$ changes state on the next clock pulse when $Q_0 = Q_1 = 1$. For the down sequence, $Q_2$ changes on the next clock pulse when $Q_0 = Q_1 = 0$. Thus, the J & K i/ps of $ff_2$ must equal 1 under the conditions expressed by the following equation:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot UP) + (\bar{Q}_0 \cdot \bar{Q}_1 \cdot DOWN)$$

- Each of the conditions for the J and K i/ps of each ff produces a toggle at the appropriate point in the counter sequence.



This figure shows a basic implementation of a 3-bit up/down binary counter using the logic equations just developed for the J & K I/Ps of each FFs. Notice that the UP/DOWN control I/P is HIGH for UP and LOW for DOWN.