# Shift Registers

A Flip-flop can store only one bit of data, a 0 or a 1.

When more bits of data are to be stored, a number of FFs are to be used.

So a register is a set of FFs used to store binary data or a Group of Flip flops used to store a word is called a register.

## Loading:

Loading a register means setting or resetting the individual FFs. ie inputting data into the register so that their states correspond to the bits of data to be stored.

Loading may be serial or parallel

In serial loading, data is transfered into the register in parallel form, that is the FFs are triggered into their new states at the same time.

Parallel i/p requires that the SET and/or RESET Control of every FF be accessed.

In a register may output data in serial form or in parallel form.

Serial output means that the data transferred out of the register, one bit at time serially.

Parallel output means that the entered data stored in the register is available in parallel form, and can be transferred out at the same time.

## Buffer Register:

Some registers do nothing more than storing a binary word. Such registers called Buffer registers. It simply stores binary word. Most of the buffer registers use D-FFs.

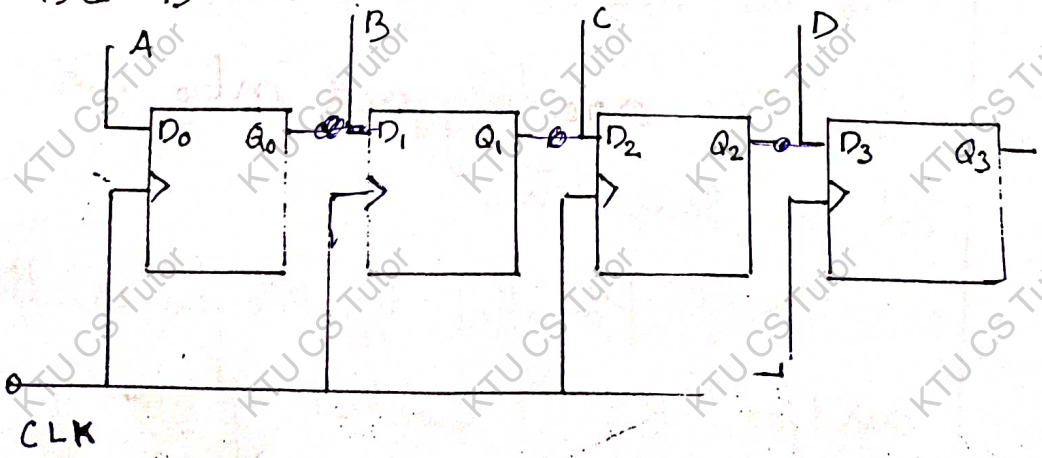


Fig shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals.

On the application of clock pulse, the o/p word becomes the same as the word applied at the i/p terminals.

ie the i/p word is loaded into the register by the application of clock pulse.

When the +ve edge of clock arrives, the stored word becomes

$$Q_3\ Q_2\ Q_1\ Q_0 = D\ C\ B\ A$$

## Shift Registers

A number of FFs connected together such that data may be shifted into and shifted out of them is called a Shift register.

Data may be shifted into or out of the register either in serial form or in parallel form.

So there are 4 basic type of Shift registers.

① Serial in, Serial out

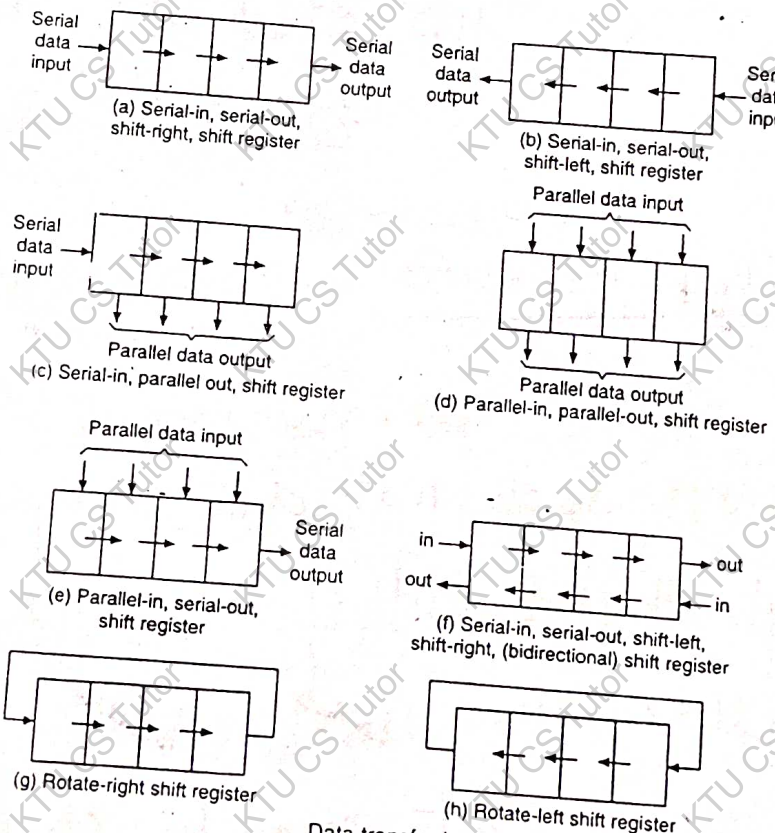
② Serial in, Parallel out

③ Parallel in, Serial out

④ Parallel in, Parallel out

Note :- ① Data may be shifted from left to right or right to left.

② Data may be rotated left or right




Data transfer in registers.

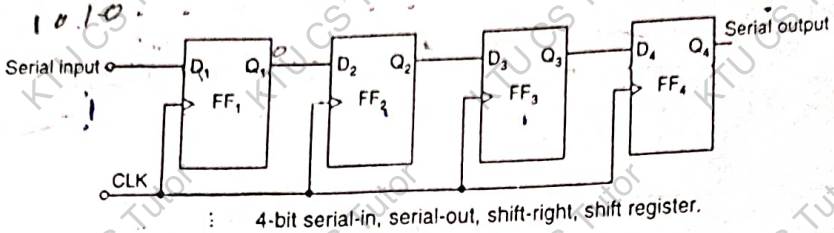
## Serial-in, Serial out shift register

This type of shift register accepts data serially, ie one bit at a time and also outputs data serially.

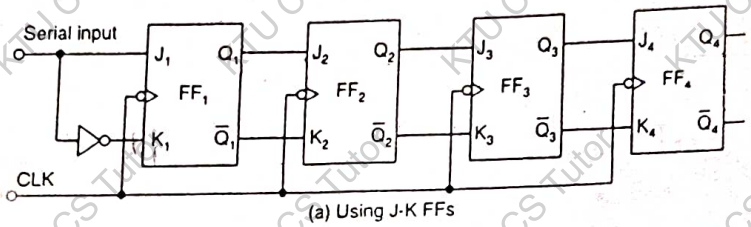
With four stages, ie four FFs, the register can store upto four bits of data. Serial data is applied to the D i/p of first FF The Q output of first FF is connected to the D i/p of the second FF & so on.

When serial data is transferred into a register, each new bit is clocked into the first FF at the +ve going edge of each clock pulse The bit that was previously stored by the first FF is transferred to the second FF, and so on.

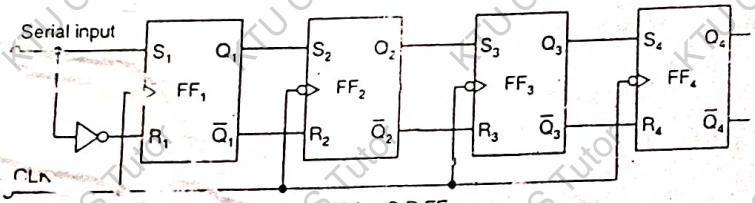
A shift register can also be constructed using J-K FFs or S-R FFs, as shown in figure below




4-bit serial-in, serial-out, shift-right, shift register.



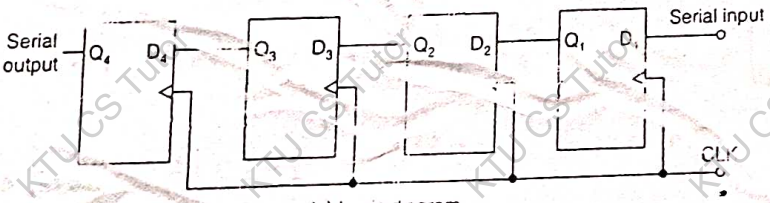

(a) Using J-K FFs



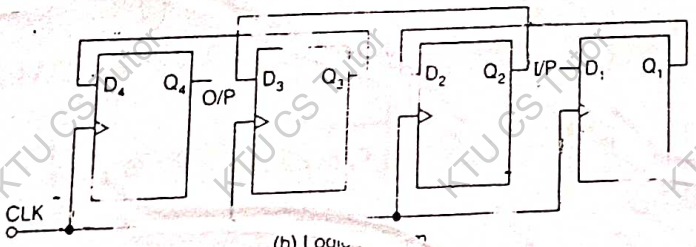

(b) Using S-R FFs

A 4-bit serial-in, serial-out, shift register.

logic diagrams of a 4-bit serial-in, serial-out, shift-left, shift register.




(a) Logic diagram

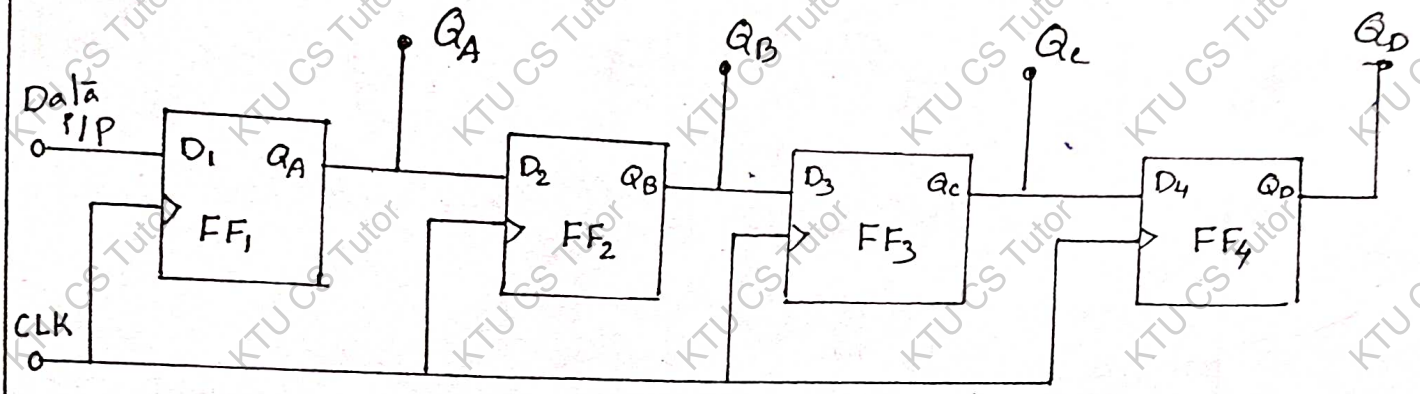


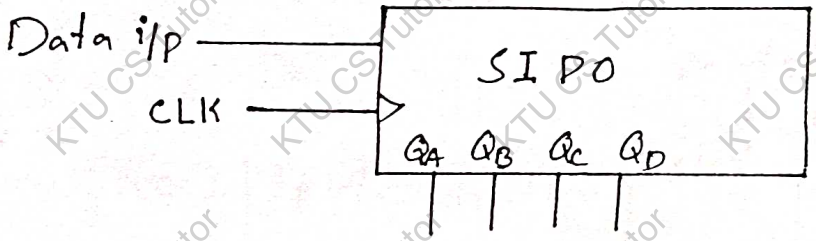

(b) Logic

A 4-bit serial-in, serial-out, shift-left, shift register.

## Serial in, parallel out shift Register

In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form.




## Parallel-in, Serial-out shift register

In this type of register, the data 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 i/ps.

The data bits are transferred out of the register serially. ie on a bit by bit basis over a single line.

Figure below illustrates a 4-bit parallel in, serial out, shift register using DFFs: There are four data lines A, B, C, D through which the data is entered in to the register in parallel form.

The signal shift/$\overline{LOAD}$ allows

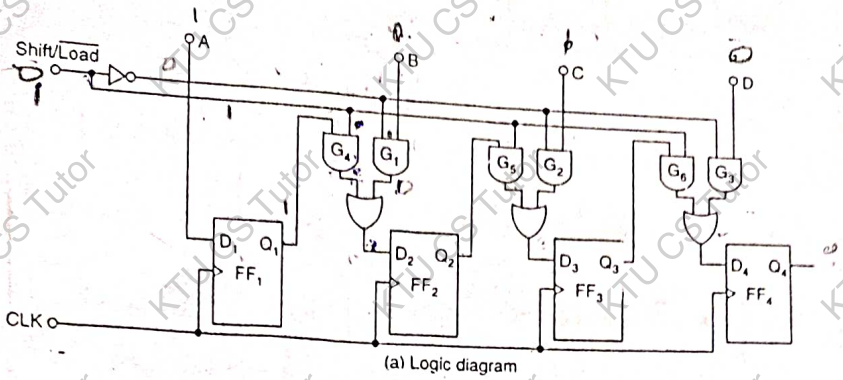
(a) the data to be entered in parallel form in to the register

(b) the data to be shifted out serially from terminal $Q_4$.

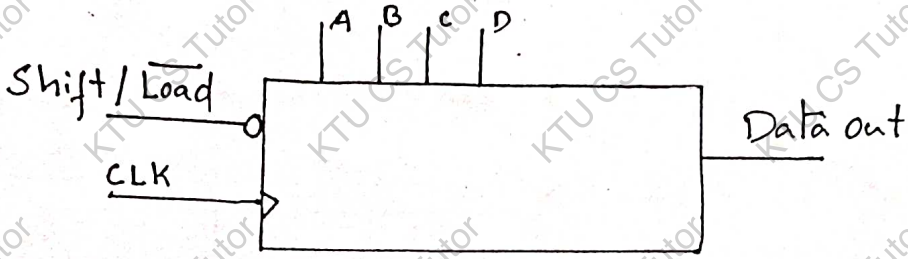
When shift/$\overline{LOAD}$ line is HIGH, Gates $G_1$, $G_2$, $G_3$ are disabled, but Gates $G_4$, $G_5$ & $G_6$ are enabled allowing the data bits to shift-from one stage to the next stage.

When shift/$\overline{LOAD}$ line is LOW, Gates $G_4$, $G_5$ & $G_6$ are disabled, whereas Gates $G_1$, $G_2$ & $G_3$ are enabled allowing data i/p to appear at the D i/ps of the respective FFs.

Load ⇒ 0 ⇒ Stores data
Shift ⇒ 1 ⇒ Outputs the data.



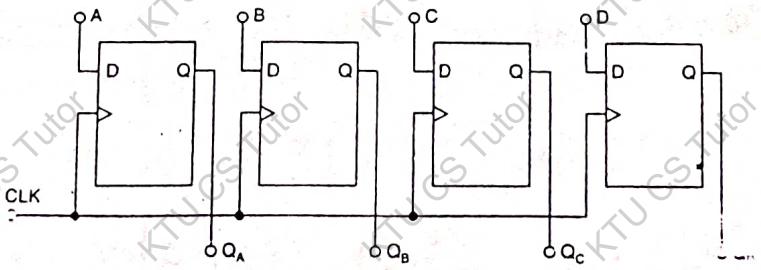
(a) Logic diagram



Logic Symbol

## Parallel-in, Parallel out Shift register

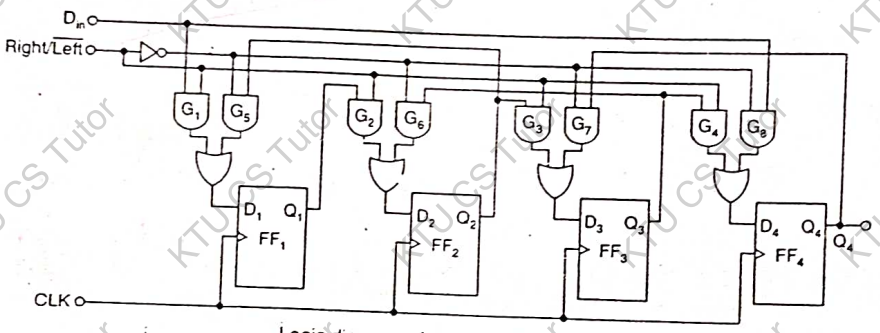
In this type of register, the data is entered in to the register in parallel form and also the data is taken out of the register in parallel form.



Logic diagram of a 4-bit parallel-in, parallel-out, shift register

## Bidirectional Shift Register

A bidirectional shift register is one in which the data bits can be shifted from left to right or from right to left.




Logic diagram of a 4-bit bidirectional shift register.

Right/$\overline{Left}$ is the mode signal.

When R/$\overline{L}$ is a 1, the logic circuit works as a shift-right shift register.

When R/$\overline{L}$ is a 0, it works as a shif-left shift register.

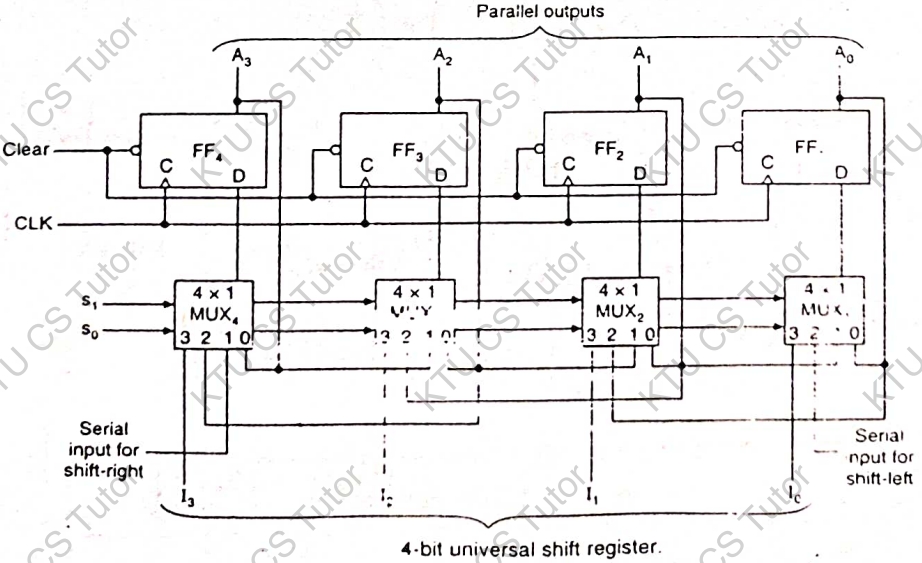
The bidirectional operation is achieved by using the mode s/g and two AND Gates and one OR Gate for each stage

# Universal Shift Registers

A register capable of shifting in one direction only is a unidirectional shift register. One that can shift in both directions is a bidirectional shift register.

If the register has both shifts and parallel load capabilities, it is refessed to as a universal shift register.

So a universal shift register is a bidirectional register, whose i/p can be either in serial form or in parallel form and whose o/p also can be either in serial form or in parallel form.




4-bit universal shift register.

A General shift register has the following capabilities

1. A clear control to clear the register.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and serial i/p and output lines associated with the shift-right.
4. A shift-left control to enable the shift-left operation and the serial i/p and output lines associated with the shift-left.
5. A parallel load control to enable a parallel transfer and the n i/p lines associated with the parallel transfer.
6. n parallel output lines
7. A control state that leaves the information in the register unchanged in the presence of the clock.

A universal shift register can be realized using multiplexers.

The above figure shows the logic diagram of a 4-bit universal shift register that has all the capabilities listed above.

It consists of four D-FFs and four Multiplexers. The four multiplexers have two common selection inputs $S_1$ and $S_0$.

Input 0 in each multiplexer is selected when $S_1 S_0 = 00$,

Input 1 is selected when $S_1 S_0 = 01$

Input 2 " " $S_1 S_0 = 10$

Input 3 " " $S_1 S_0 = 11$

The selection inputs control the mode of operation of the register according to the function entries in Table.

| Mode Control | | |
|---|---|---|
| $S_1$ | $S_0$ | Register Operation |
| 0 | 0 | No change |
| 0 | 1 | Shift right |
| 1 | 0 | Shift Left |
| 1 | 1 | Parallel Load |

When $S_1S_0 = 0$, the present value of the register is applied to the D-inputs of Flip flops. This condition forms a path from the output of each flipflop into the i/p of the same flipflop.

The next clock edge transfers into each flip flop, the binary value it held previously, and no change of state occurs.

When $S_1S_0 = 01$, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial i/p transferred into $FF_4$.

When $S_1S_0 = 10$, a shift left operation results with the other serial i/p going into $FF_1$.

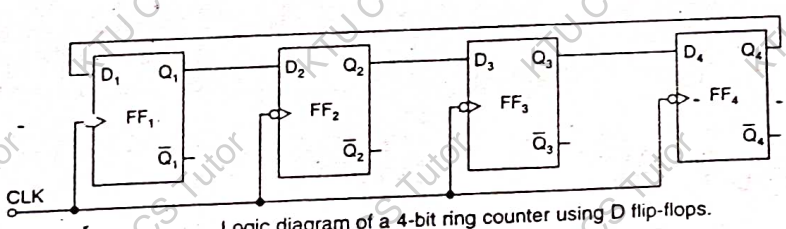
When $S_1S_0 = 11$, the binary information on the parallel i/p lines is transferred into the register simultaneously during the next clock edge.

# Application of Shift registers

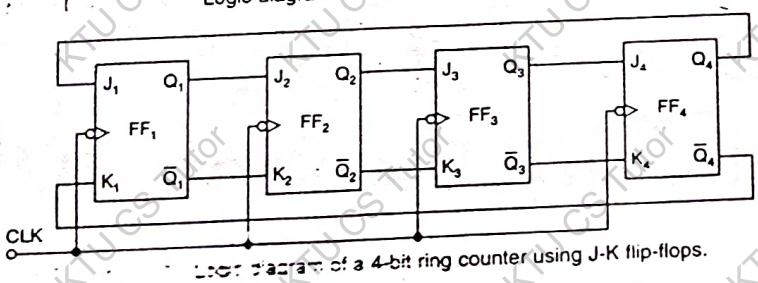
## Shift register Counters

### (a) Ring Counter:

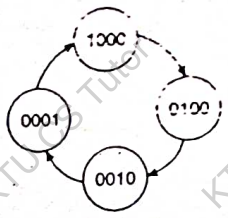
The FFs are arranged as in a normal shift register, ie the Q output of each stage is connected to the D i/p of the next stage, but the o/p of the last FF is connected back to the D i/p of first FF such that the array of FFs is arranged in a ring and therefore, the name ring Counter.




Logic diagram of a 4-bit ring counter using D flip-flops.




Logic diagram of a 4-bit ring counter using J-K flip-flops.



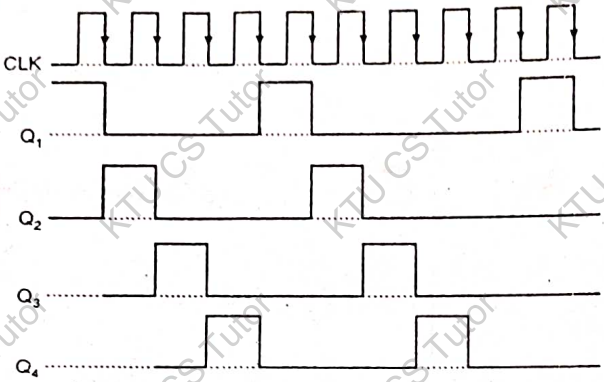

(a) State diagram

| $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | After clock pulse |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 1 | 3 |
| 1 | 0 | 0 | 0 | 4 |
| 0 | 1 | 0 | 0 | 5 |
| 0 | 0 | 1 | 0 | 6 |
| 0 | 0 | 0 | 1 | 7 |

(b) Sequence table

State diagram and sequence table of a 4-bit ring counter.

around the register

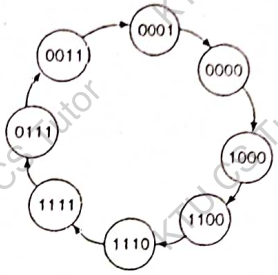


Timing diagram of a 4-bit ring counter.

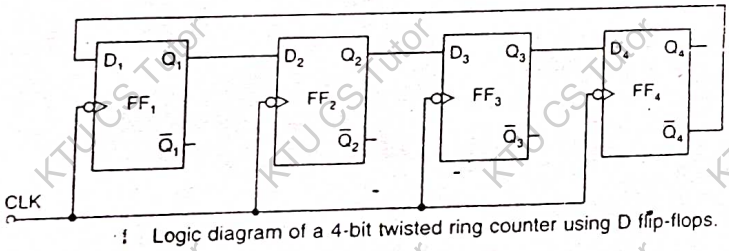
## Twisted Ring Counter (Johnson Counter)

The Q output of each sta
is connected to the D-i/p of the next stage
but the $\bar{Q}$ output of the last stage is
Connected to the D input of 1st stage,
Therefore the name twisted ring Counter.

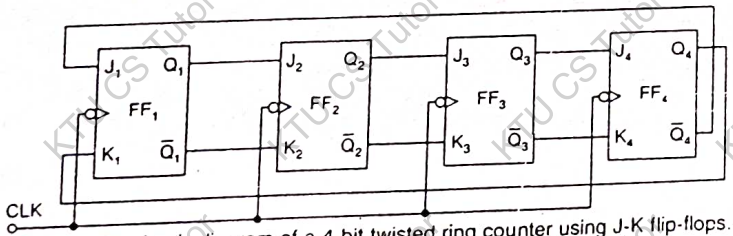
| CLK | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |



(a) State diagram



Logic diagram of a 4-bit twisted ring counter using D flip-flops.



Logic diagram of a 4-bit twisted ring counter using J-K flip-flops.

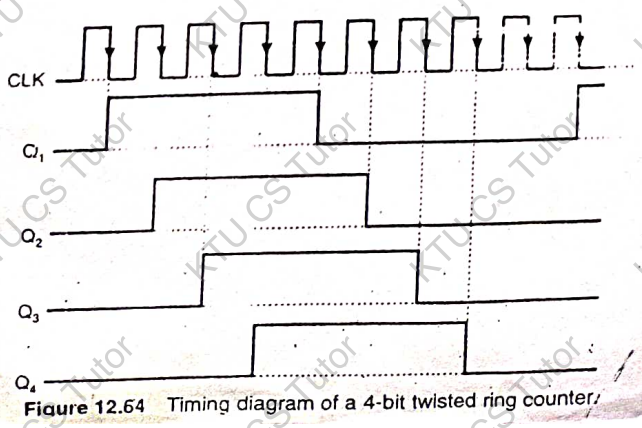


**Figure 12.64** Timing diagram of a 4-bit twisted ring counter.

## Serial Adder

The Serial Adder is used to add binary numbers in serial form. The two binary numbers to be added serially are stored in two shift registers, A & B. Bits are added one pair at a time through a single Full Adder Circuit.

The Carry out of the FA is transferred to a D-FF. The output of this FF is then used as the Carry i/p for the next pair of significant bits. The Sum bit from the S output of the Full Adder could be transferred to a third shift register. By shifting the Sum into A, while the bits of A are shifted out, it is possible to use one register for storing both augend and the Sum bits. The serial i/p register B can be used to transfer a new binary number while the addend bits are shifted out during the addition.

## Operation [Working]:

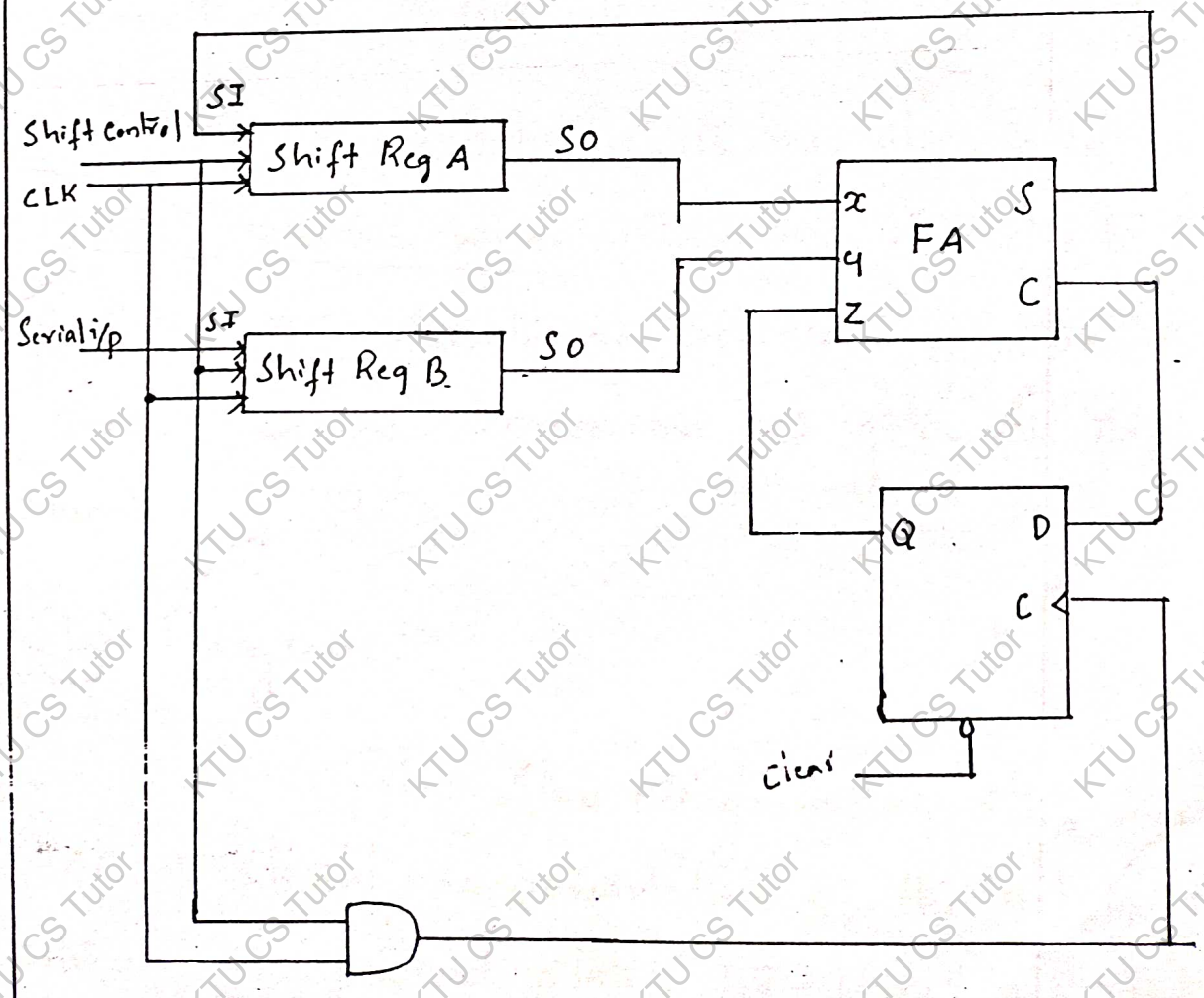
Initially register A holds the augend, register B holds the addend and Carry flip flop is cleared to '0'. The output (SO) of A & B provide a pair of significant bits for the full adder at 'x' & 'y'.

The Shift Control enables both registers and Carry flip flop; So at the clock pulse both registers are shifted one to the right, the Sum bit from S enters the left most FF of A and the o/p Carry is transferred into Flip flop Q.

The Shift Control enables the registers for a number of clock pulses equal to the number of bits of the regis For each Succeeding Clock pulse a ne Sum bit is transferred to A, a new Ca is transferred to Q and both register are shifted one to the right.

This process continues until the shift control is disabled.

Thus the addition is accomplished by passing each pair of bits together with the previous carry through a single full adder circuit and transferring the sum, one bit at a time, into register A.



Initially, register A and the Carry F
are cleared to 0 and then the 1st number
added from B. While B is shifted through
the Full Adder, a second number is transf
to it through its Serial i/p. The second
number is then added to the Content of
register A, while a third number is
transferred Serially into register B.

## Counters

A digital Counter is a set of Flip flops (FFs) whose states change in response to pulses applied at the input to the Counter.

The FFs are interconnected such that their Combined state at any time is the binary equivalent of the total number of pulses that have occurred upto that time.

Thus, as its name implies, a Counter is used to Count clock pulses.

Counters are classified into two types

1. Asynchronous Counters or ripple Counters
2. Synchronous Counters.

In Synchronous Counters the Common clock is connected to all the flip flops and thus they are clocked Simultaneously.

In asynchronous Counters, the flip flops is triggered by external clock pulse Hence in asynchronous Counters, the flip-flops are not clocked Simultaneously and each Successive flip-flop is clocked by the Q or $\bar{Q}$ output of the previous flip flop.

Asynchronous Counters are also Called ripple Counters, because the transition of the first stage ripples through to the last Stage.

| Asynchronous Counters | Synchronous Counters |
|---|---|
| 1. FFs are Connected in such a way that the output of first FF drives the clock for the Second FF, the output of the Second, the clock of the 3rd & So on. | 1. There is connec between the o/p of first FF and clock of next FF |
| 2. All the FFs are not clocked Simultaneously | All the FFs are clocl Simultaneously |

| | |
|---|---|
| 3. Design and implementation is very simple even for more number of states | 3. Design and implementation becomes tedious and complex as the number of states increases. |
| 4. Main drawback is their low speed as the clock is propagated through a number of FFs before it reaches the last FF | 4. Since clock is applied all the FFs simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster. |

A Counter may be

(i) up-Counter (increasing order counter)

(ii) down-Counter (decreasing order counter)

An up-Counter is a Counter which counts in the upward direction

ie 0, 1, 2, - - - N

A down Counter is a Counter which counts in the downward direction

ie N, N-1, N-2, - - - 1, 0

Each of the Counts of the Counter is Called the State of the Counter.

The number of states through which the Counter passes before returning to the starting state is called the modulus of the Counter.

Hence, the modulus of a Counter is equal to the total number of distinct states (Counts) including Zero, that a Counter can store.

ie [The number of i/p pulses that Causes the Counter to reset to its initial Count is called. the modulus of the Counter.

Eg: a 2-bit Counter has 4 states
it is called a mod-4 Counter

It divides the i/p clock signal frequency by 4.

∴ it is also called divide-by-4 Counter

111[4] a 3-bit Counter divides the clock freq by 8

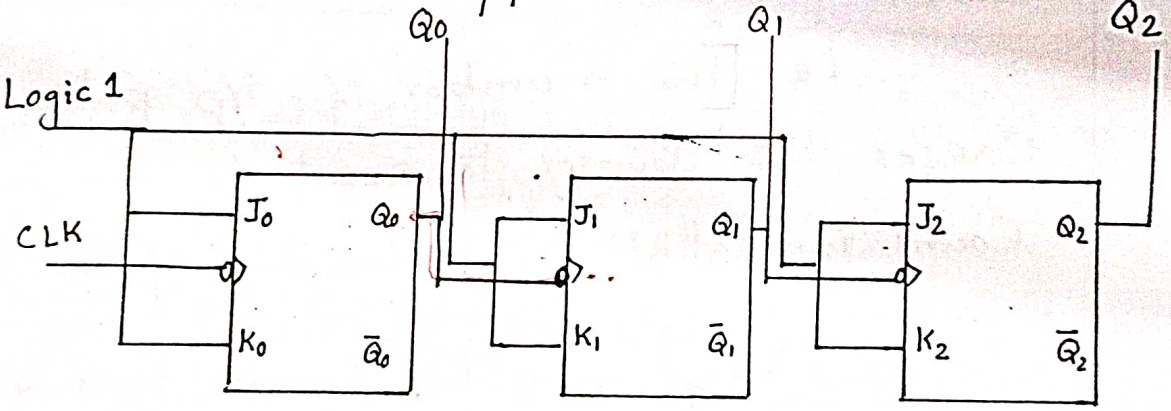
- a 4-bit Counter divides by 16 & so on

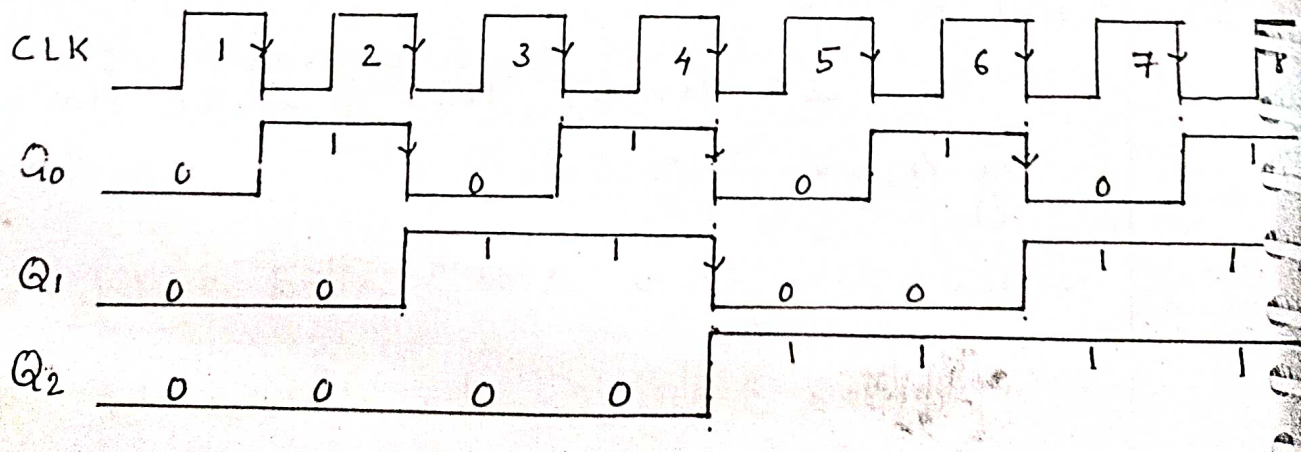
Note:- an n-bit Counter, will have n-FFs and $2^n$ states and divides the i/p clock frequency by $2^n$.

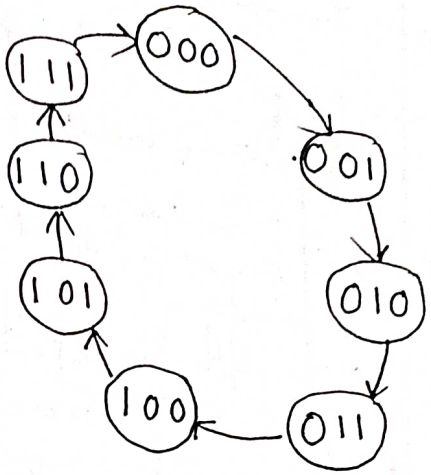
Hence it is called a divide-by-$2^n$ Counter.

Note:- A mod-N Counter divides the clock frequency by N, hence it is called a divide-by-N Counter.

Mod 8 or 3-bit ripple Counter (3-bit up Counter):



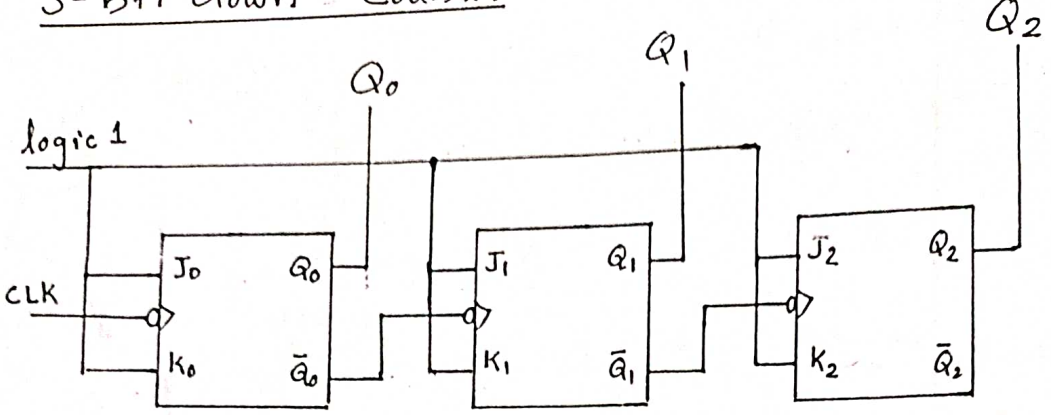



State diagram

## Truth Table

| CLK | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 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 |

## 3-bit down Counter




decreasing order
$\overline{Q}$ as clock

### Truth Table

| CLK | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 1 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 |

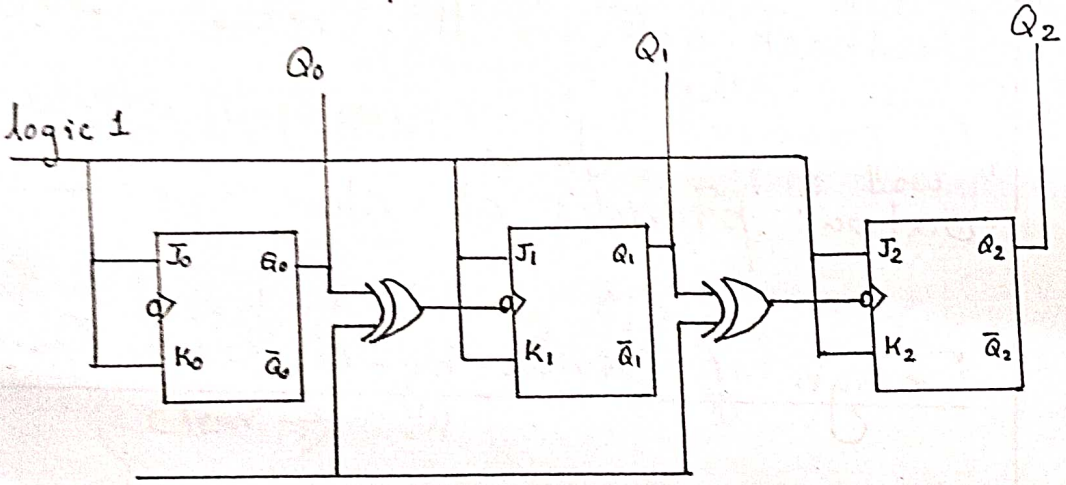
## 3-bit up/down Counter

The direction of Counting Seq
is decided by a mode control i/p M.

An XOR Gate between flip flops functioning
Controlled inverter connects either of Q or $\bar{Q}$

to the clock i/p of the succeeding flip flop as decided by the logic state at M.

When mode control is 0, Q outputs get connected to the clock i/ps of the succeeding flip flops and counter counts up. When mode control is 1, $\bar{Q}$ outputs are getting connected to the clock i/ps and counter counts down.




M = 0 for counting up.
= 1 for counting down.

## Counters with Mod-numbers less than $2^n$

Counters can also be designed to have a number of states in their sequence less than $2^n$.

The basic counter is modified

to produce the required Mod number less than $2^n$ by allowing the Counter to skip states that are to part of the Counting Sequence.

One of the most common method, for doing this is making all flip-flops to reset after the Count N with the help of feedback gate in the Circuit. Such feedback is provided by a NAND Gate, in which the output provides all clear i/ps in parallel.

## Design of Divide-by-N Ripple Counter

Procedure:

1. Determine the number n flip flop required based on number N
2. Connect the n-flip flops as a ripple Counter
3. Find the representation of N
4. Connect all flip-flops outputs that 1 at the Count N of the Counter, an i/p to the NAND Gate.

5. Connect the NAND Gate output to the Clear i/ps of all the flip-flops.

Note:

For designing Counters with mod number less than $2^n$, two more i/ps are introduced called Preset & clear

Both Preset & clear are active low i/ps.

Active Low i/ps → Ckt will be active only whe input is Low.

- Preset → Making o/p 1
- Clear → " " 0

| Preset | clear | o/p(Q) |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | Normal opr$^n$ |
| 0 | 0 | Invalid |

Note: When Preset & clear is present, the Output (Q) follows preset & clear irrespective of i/ps (J & K) & clock.

## Mode - 6 Ripple Counter

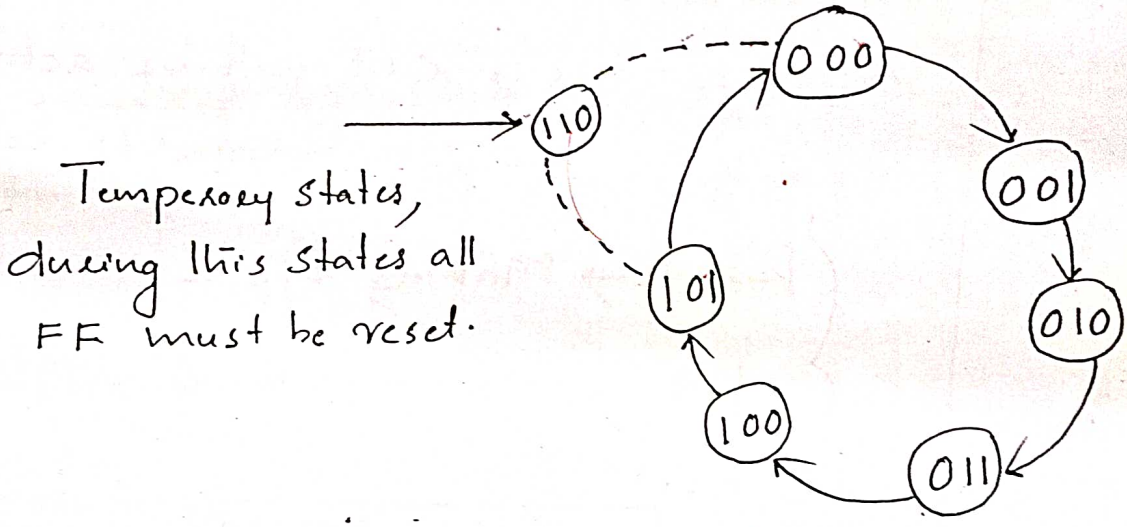
Here $N = 6$

∴ Number of flip-flops required

$n = 3 \quad (N \leq 2^n)$

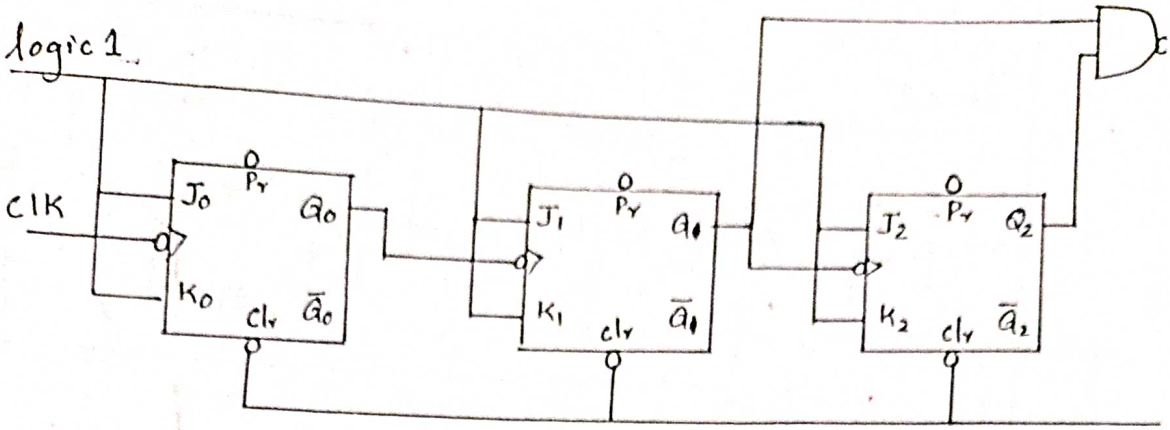
3 FFs are required.

Counts from 000 to 101




Temperory states, during this states all FF must be reset.

| CLK | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 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 | 0 | 0 | 0 |




Note :- The same circuit can be implemented using T FF also.

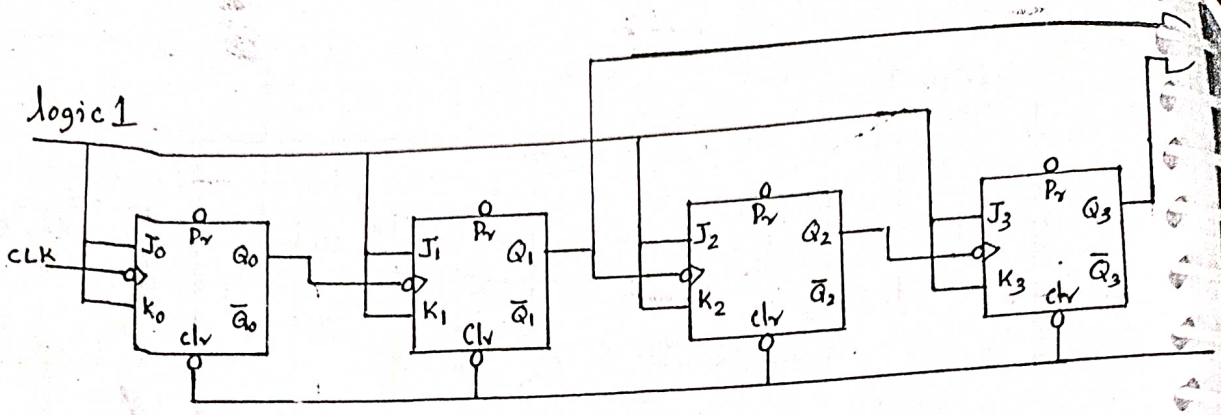
Design a Mod-10 Asynchronous Counter

OR

Design a Decade Counter

| CLK | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | 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 | 0 | 0 | 0 | 0 |

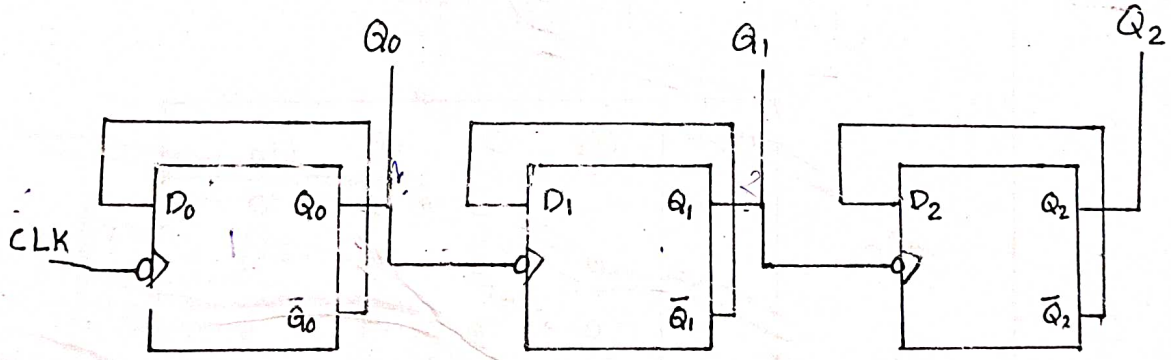
1001
1010 Do
$Q_3$ $Q_1$
1010
0000
1010




Implement a 3-bit ripple Counter using D FFs.

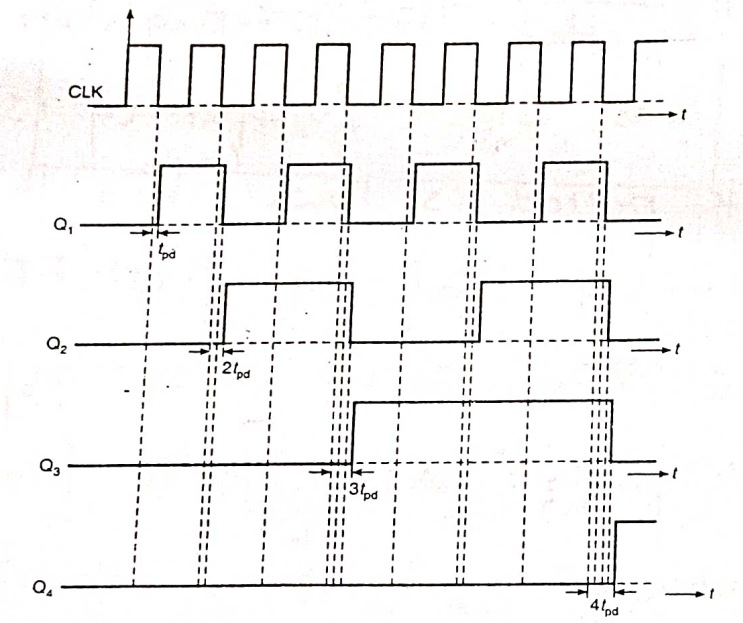
For ripple Counters, the FF used must be in toggle mode.

The D FFs may be used in toggle mode by connecting the $\bar{Q}$ of each FF to its D terminal



# Effects of Propagation delay in Ripple Counters

In ripple counters, each FF is toggled by the changing state of the preceding FF. So no FF can change state until after the propagation delay of the FF that precedes it. ie until all preceding FFs Complete their transition.




Timing diagram considering propagation delay

Eg: For a 4-bit Counter, all the FFs change states when the Count goes from 0111 to 1000.

So after the eighth clock pulse is

applied, $Q_1$ will not change state instantaneously, but goes from a 1 to a 0 after a propagation delay of tpd, $Q_2$ changes state after another propagation delay time of 2 tpd. $Q_3$ changes state after 3 tpd and $Q_4$ changes state after 4tpd.

If the propagation delay is large and clock frequency is high ie clock period is Low, then there is a possibility that the first FF responds to the new clock pulse before the previous clock pulse has effected transition of the fourth FF.

When the last FF finaly responds, the Counter may read 1001 having skipped 1000. Thus the Count goes directly from 0111 to 1001.

Thus the propagation delays

The FFs of a ripple counter impose a limit on the frequency at which the counter can be clocked.

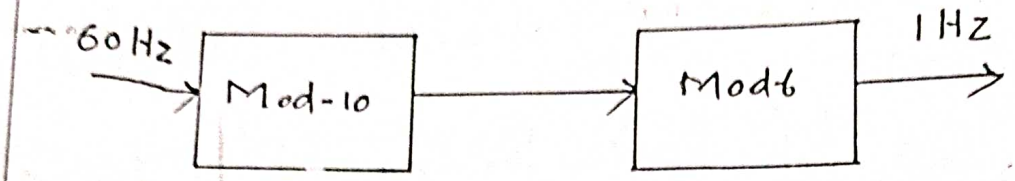
If $T_c$ is the period of the clock pulse, 'n' the number of stages and '$t_{pd}$' the propagation delay in each stage, then the clock frequency $f_c$ is constrained by

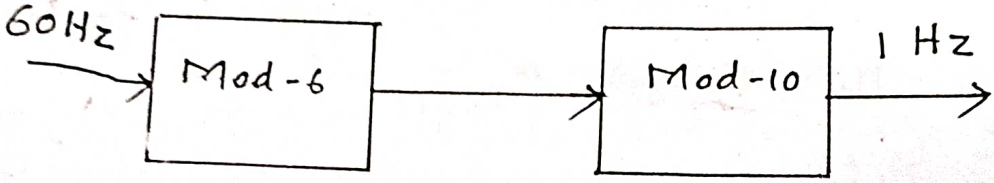
$$\frac{1}{T_c} = f_c < \frac{1}{n t_{pd}}$$

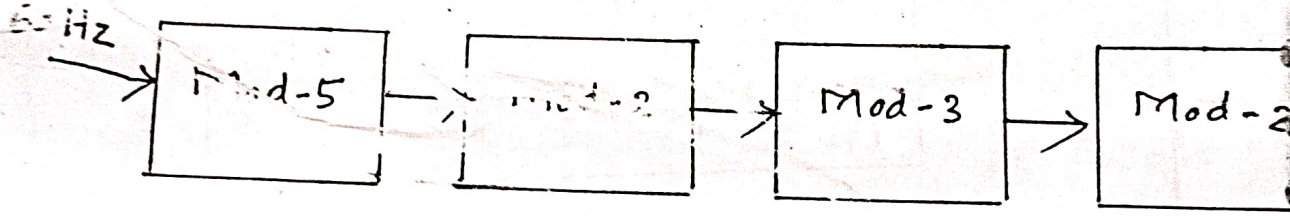
## Cascading of Ripple Counters

Ripple Counters can be connected in cascade to increase the modulus of the Counter.

A mod-M and a mod-N Counter in Cascade give a mod-MN Counter.


60Hz
Mod-10
Mod-6
1Hz


60Hz
Mod-6
Mod-10
1 Hz


Mod-5
Mod-3

# Synchronous Counters

## Design of Synchronous Counters

Step 1: Identify the no of FF required

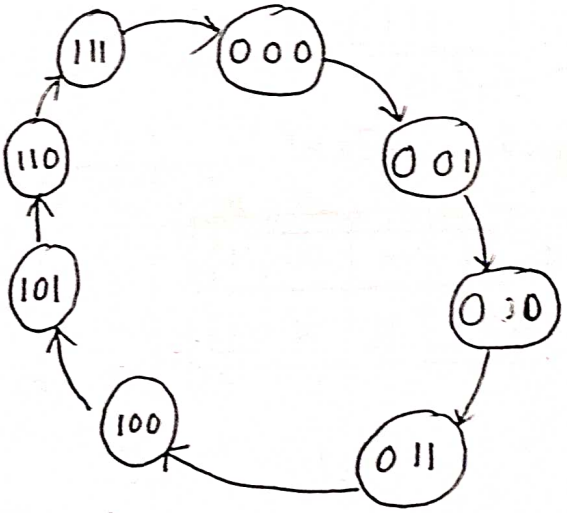
2: Identify the states of FF & draw the State diagram.

3: Prepare the excitation table of Counter Showing the present state, next state and i/p to the FF.

4: Simplify the i/p to the FF using k-ma

5: Implement using Gates & Flip flops.

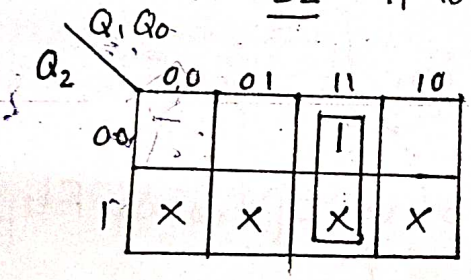
## Design a 3-bit up Counter



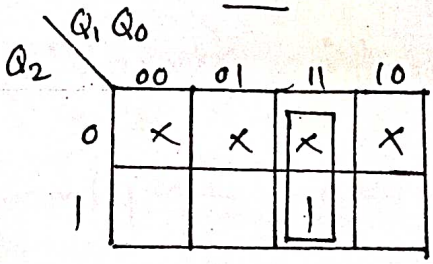

State diagram

| Present State | | | Next State | | | Required i/ps | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | x | 0 | x | 1 | x |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | x | 1 | x | x | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | x | x | 0 | 1 | x |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | x | x | 1 | x | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | x | 0 | 0 | x | 1 | x |
| 1 | 0 | 1 | 1 | 1 | 0 | x | 0 | 1 | x | x | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | x | 0 | x | 0 | 1 | x |
| 1 | 1 | 1 | 0 | 0 | 0 | x | 1 | x | 1 | x | 1 |

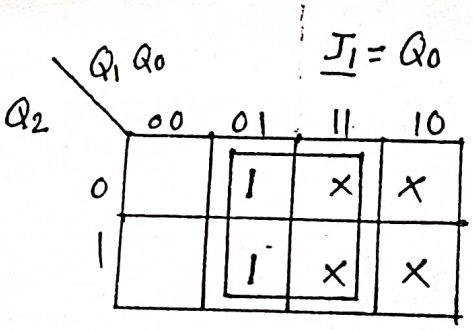
$J_2 = Q_1 Q_0$



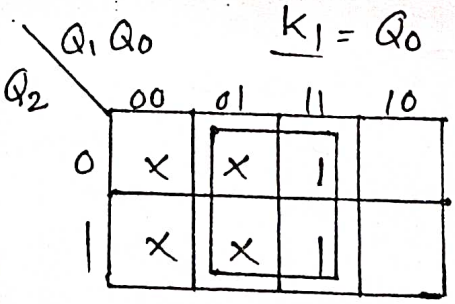

$K_2 = Q_1 Q_0$



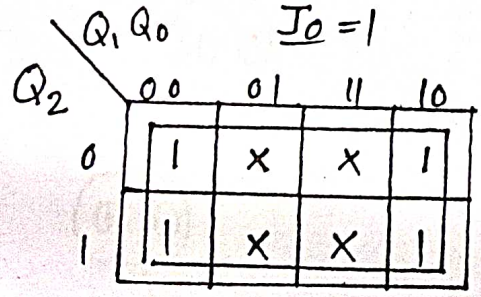

$J_1 = Q_0$



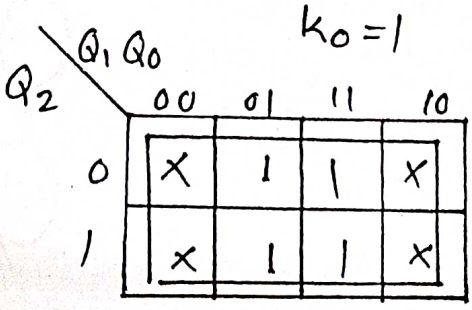

$K_1 = Q_0$

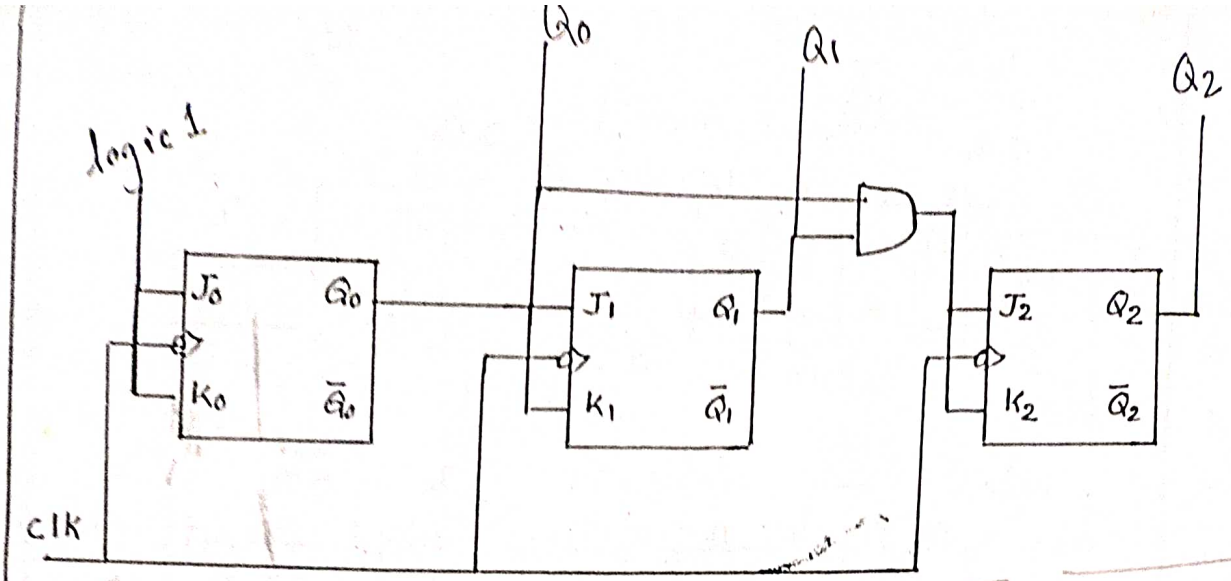



$J_0 = 1$



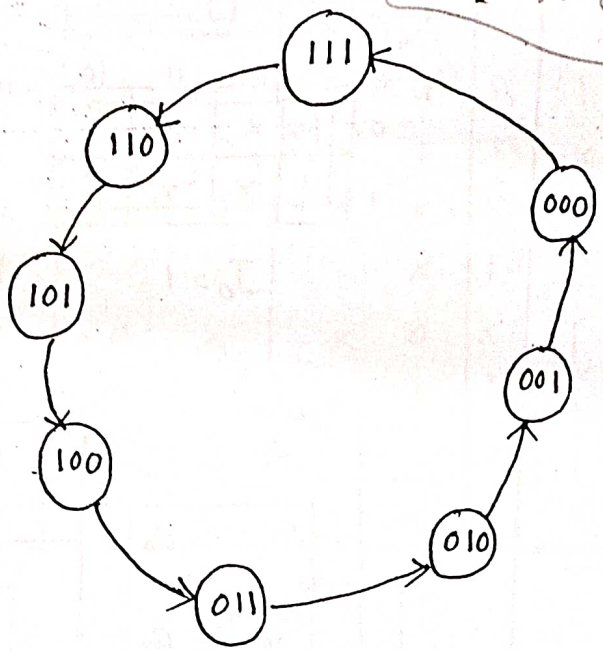

$K_0 = 1$






When zero comes it will get enable Preset the FF → Active low signal.

## Design a 3-bit Down Counter




| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | x | 0 | x | 0 | x | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | x | 0 | x | 1 | 1 | x |
| 1 | 0 | 1 | 1 | 0 | 0 | x | 0 | 0 | x | x | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | x | 1 | 1 | x | 1 | x |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | x | x | 0 | x | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | x | x | 1 | 1 | x |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | x | 0 | x | x | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | x | 1 | x | 1 | x |

## $J_2$

| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | X | X | X | X |

$J_2 = \bar{Q}_1 \bar{Q}_0$

## $K_2$

| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | X | X |
| 1 | 1 | 0 | 0 | 0 |

$K_2 = \bar{Q}_1 \bar{Q}_0$

## $J_1$

| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | X | X |
| 1 | 1 | 0 | X | X |

$J_1 = \bar{Q}_0$

## $K_1$

| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | X | 0 | 1 |
| 1 | X | X | 0 | 1 |

$K_1 = \bar{Q}_0$

## $J_0$

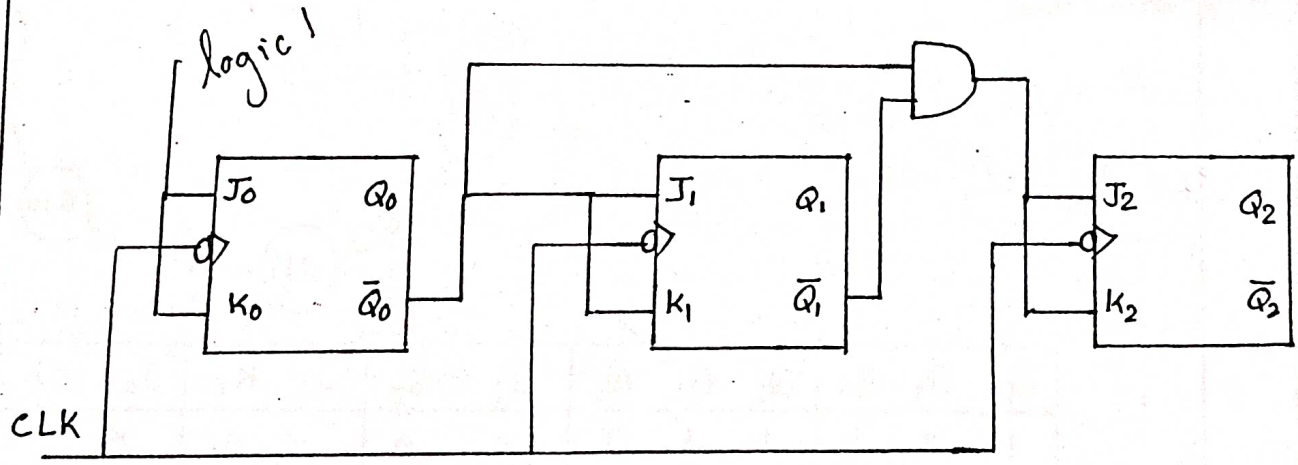
| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | X | X | 1 |
| 1 | 1 | X | X | 1 |

$J_0 = 1$

## $K_0$

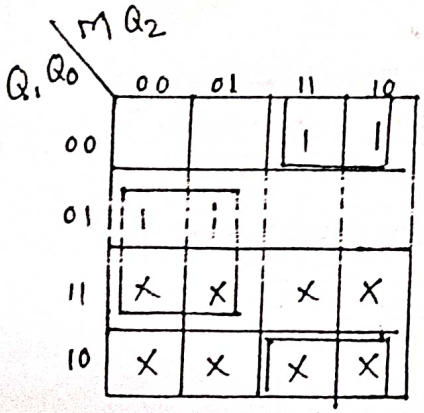
| $Q_2$ \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | X | 1 | 1 | X |
| 1 | X | 1 | 1 | X |

$K_0 = 1$

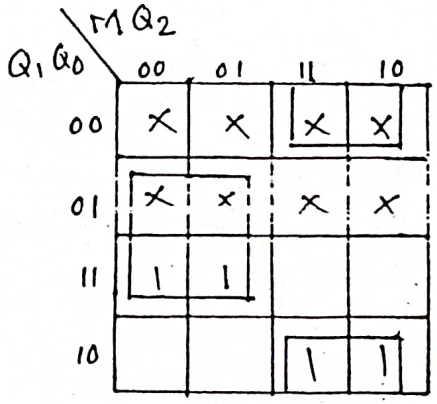


# Design a 3-bit up/down Counter

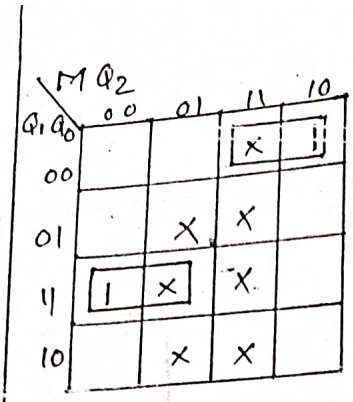
| Mode | $Q_2$ | $Q_1$ | $Q_0$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | 1 | X |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | X | 1 | X | X | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | X | 0 | 1 | X |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | X | X | 1 | X | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | X | 0 | 0 | X | 1 | X |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | X | 0 | 1 | X | X | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | X | 0 | X | 0 | 1 | X |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | X | 1 | X | 1 | X | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | X | 0 | X | 0 | X | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | X | 0 | X | 1 | 1 | X |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | X | 0 | 0 | X | X | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | X | 1 | 1 | X | 1 | X |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | X | X | 0 | X | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | X | X | 1 | 1 | X |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | X | 0 | X | X | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | X | 1 | X | 1 | X |



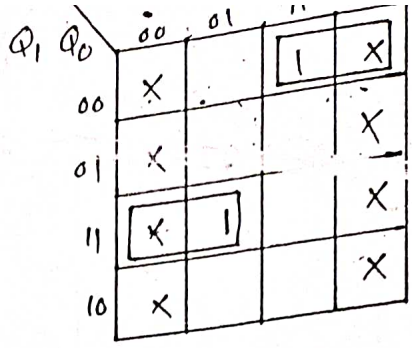

$$J_1 = \overline{M} Q_0 + M \overline{Q_0}$$




$$K_1 = \overline{M} Q_0 + M \overline{Q_0}$$




$J_2 = \bar{M} Q_1 Q_0 + M \bar{Q}_1 \bar{Q}_0$

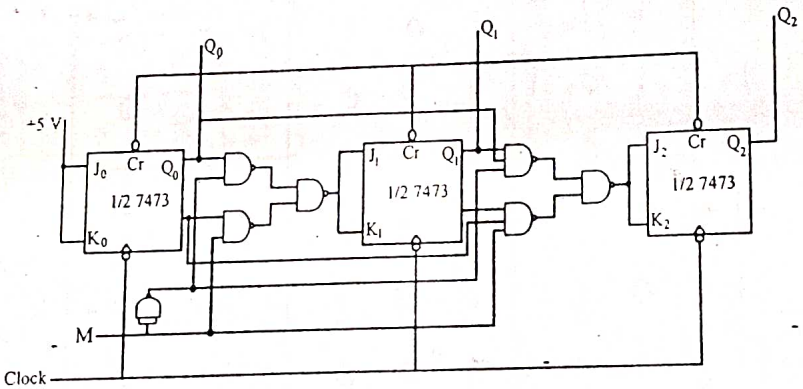



$K_2 = \bar{M} Q_1 Q_0 + M \bar{Q}_1 \bar{Q}_0$

From the direct inspection, we get

$J_0 = K_0 = 1$

Circuit diagram for 3-bit up/down counter

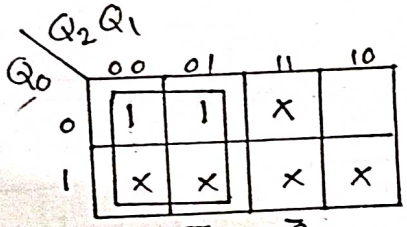



Condition: Preset 0 → output — 1 (set)
clear 0 → output — 0 (cleared)

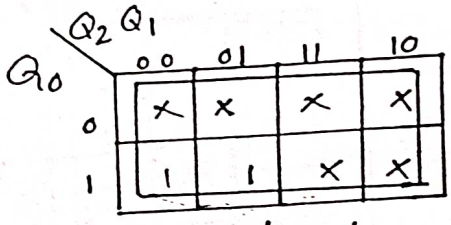
# Design a Mod-5 up Counter

| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | x | 0 | x | 1 | x |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | x | 1 | x | x | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | x | x | 0 | 1 | x |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | x | x | 1 | x | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | x | 1 | 0 | x | 0 | x |

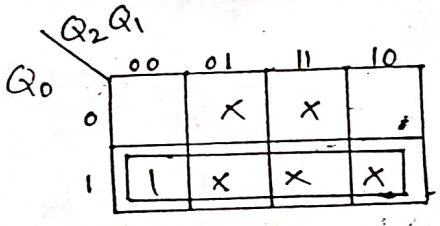
Put 101, 110, 111 (5) (6) (7) as don't care conditions



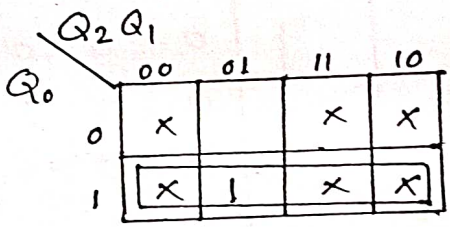

$J_0 = \bar{Q}_2$



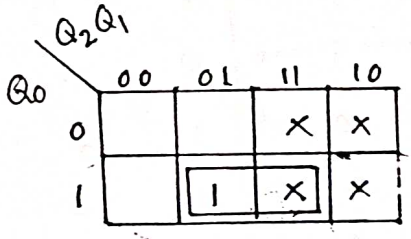

$K_0 = 1$



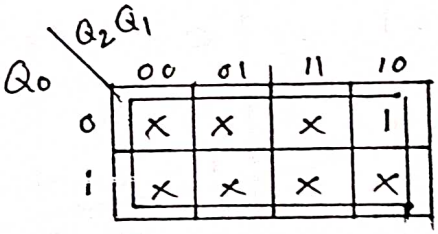

$J_1 = Q_0$




$K_1 = Q_0$

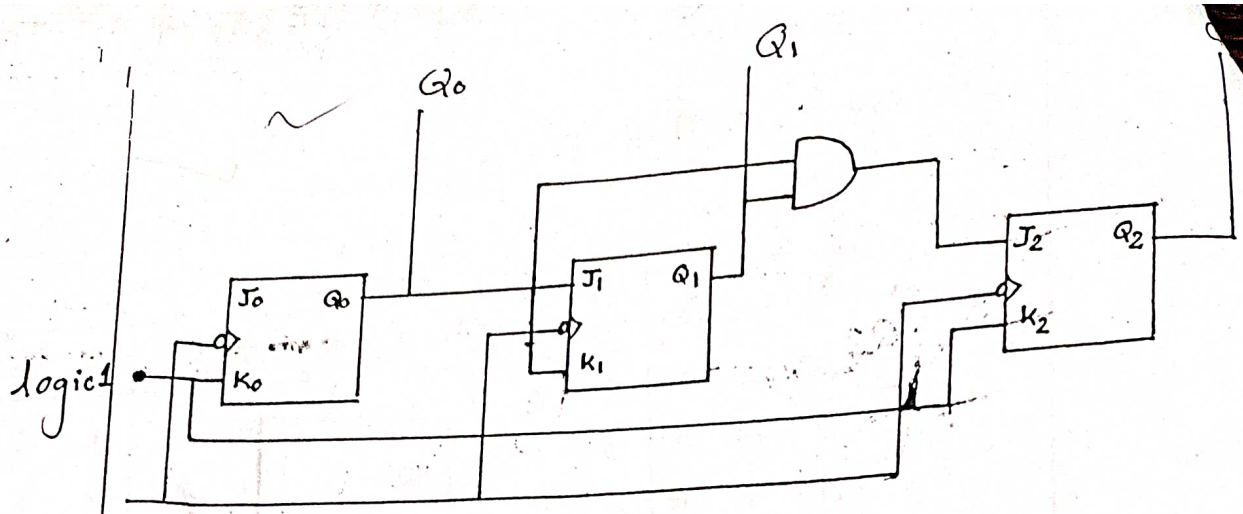



$J_2 = Q_0 Q_1$




$K_2 = 1$

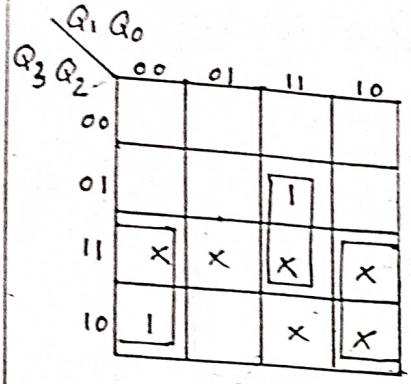
00 0 x
01 1 x
10 x 1
11 x 0



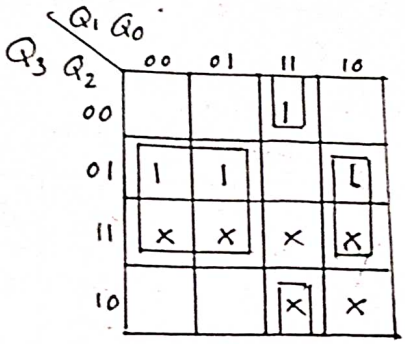

Design a BCD or decade Counter using D Flip flop. [Mod-10 Counter]

| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3^+$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

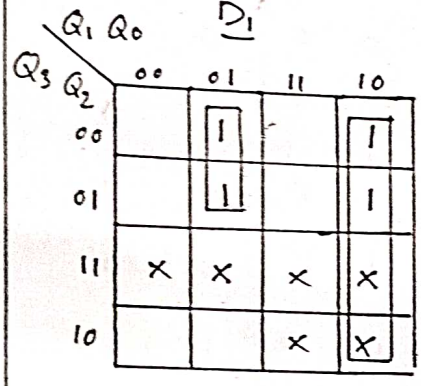
$D_3$



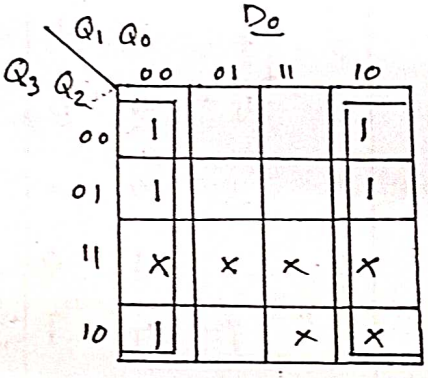

$D_2$




$D_1$




$D_0$




$$D_3 = Q_2 Q_1 Q_0 + Q_3 \bar{Q}_0$$

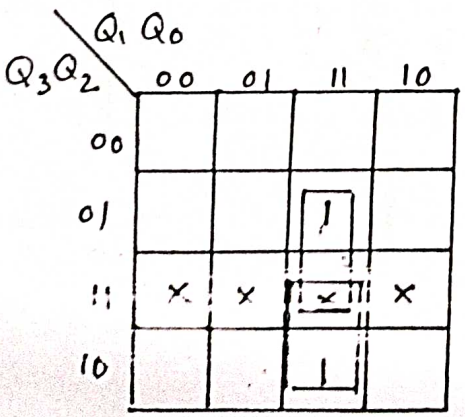
$$D_2 = Q_2 \bar{Q}_1 + Q_2 Q_1 \bar{Q}_0 + \bar{Q}_2 Q_1 Q_0$$

$$D_1 = \bar{Q}_3 \bar{Q}_1 Q_0 + Q_1 \bar{Q}_0$$

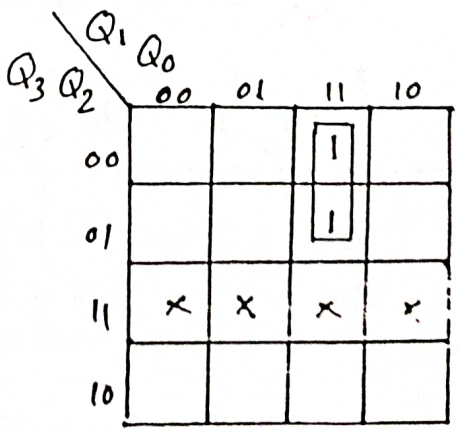
$$D_0 = \bar{Q}_0$$

# Design a mod-12 Counter using T-FF

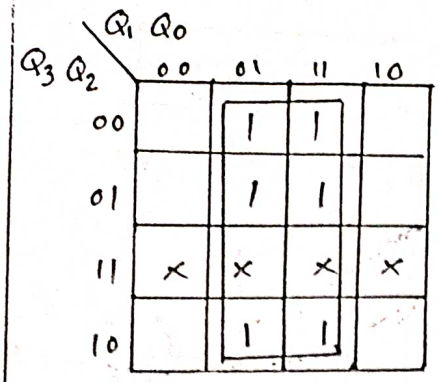
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3^+$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $T_3$ | $T_2$ | $T_1$ | $T_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |



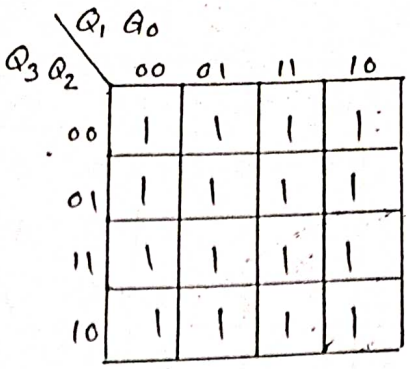

$T_3 = Q_2 Q_1 Q_0 + Q_3 Q_1 Q_0$




$T_2 = \bar{Q}_3 Q_1 Q_0$




$T_1 = Q_0$




$T_0 = 1$

$T_3 = Q_2 Q_1 Q_0 + Q_3 Q_1 Q_0$

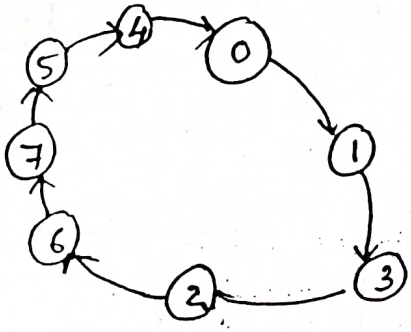
$T_2 = \bar{Q}_3 Q_1 Q_0$

$T_1 = Q_0$

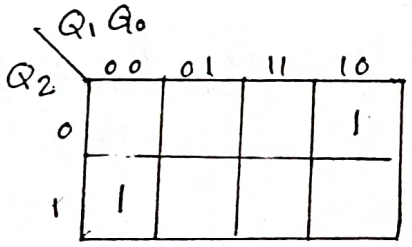
$T_0 = 1$

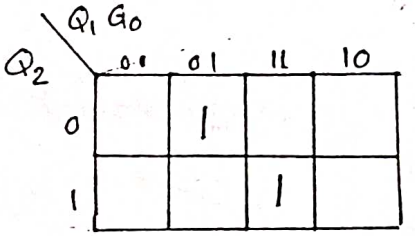
Implement using FFs & logic Gates

Design a Counter to Count the following Seque
0, 1, 3, 2, 6, 7, 5, 4 [OR design a 3-bit Graycode Counter]



| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $T_2$ | $T_1$ | $T_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

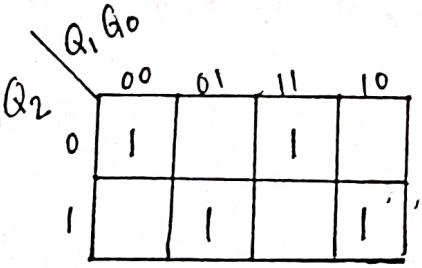



$$T_2 = Q_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_2 Q_1 \bar{Q}_0$$

$$T_1 = \bar{Q}_2 \bar{Q}_1 Q_0 + Q_2$$

$Q_1 Q_0$




$$\bar{Q}_2 \bar{Q}_1 \bar{Q}_0 + \bar{Q}_2 \bar{Q}_1 Q_0 + Q_2 \bar{Q}_1 Q_0 + Q_2$$

$Q_1 Q_0$

$$\bar{Q}_0 \left[ \bar{Q}_2 \bar{Q}_1 + Q_2 Q_1 \right] + Q_0 \left[ \bar{Q}_2 Q_1 + Q_2 \right]$$

$Q_1 Q_0$

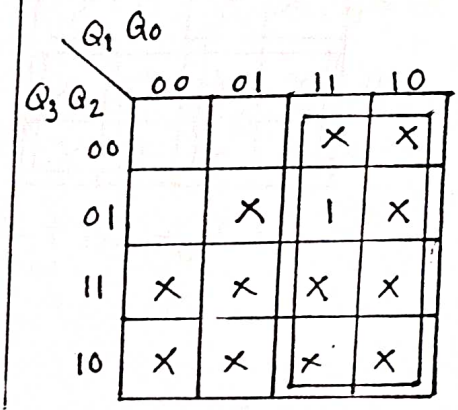
$$\bar{Q}_0 \left[ \overline{Q_2 \oplus Q_1} \right] + Q_0 \left[ Q_2 \oplus Q_1 \right]$$

A B

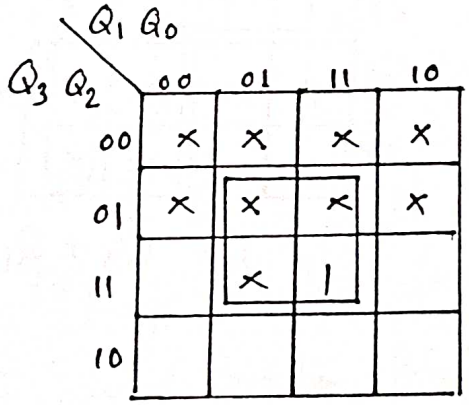
$$Q_0 \oplus Q_2 \oplus Q_1$$

Design a Counter to Count the Sequence 0, 1, 4, 7, 11, 12, 14, 15 using J-K Flip flop.

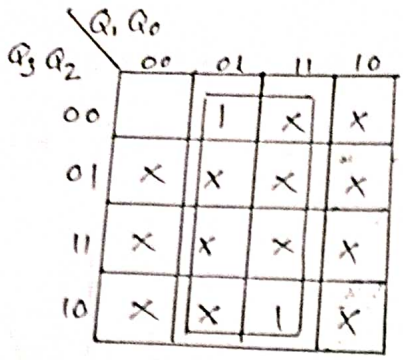
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $Q_3^+$ | $Q_2^+$ | $Q_1^+$ | $Q_0^+$ | $J_3$ | $K_3$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | 0 | X | 0 | X | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | X | 1 | X | 0 | X | > |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | X | X | 0 | 1 | X | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | X | X | 1 | X | 0 | > |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | X | 0 | 1 | X | X | 1 | > |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | X | 0 | X | 0 | 1 | X | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | X | 0 | X | 0 | X | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | X | 1 | X | 1 | X | 1 | X |



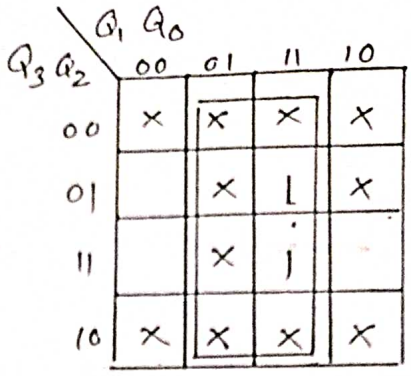

$J_3 = Q_1$



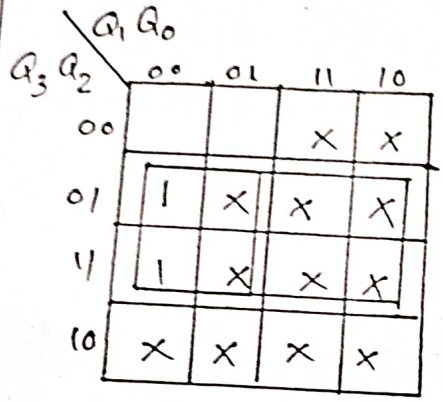

$K_1 = Q_2 Q_0$



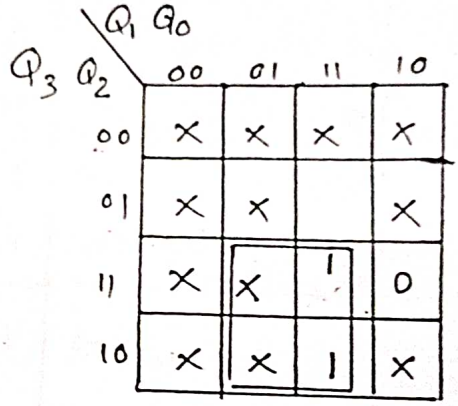

$J_2 = Q_0$



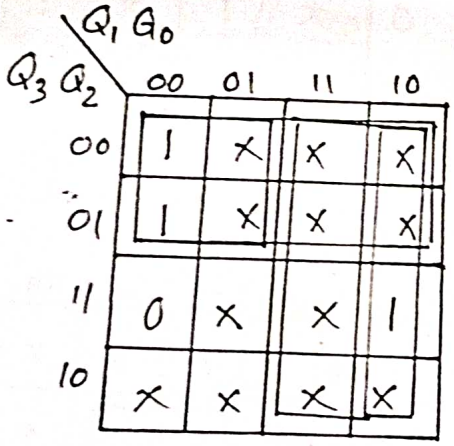

$K_2 = Q_0$



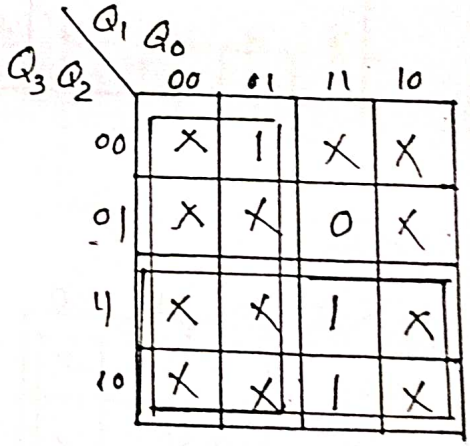

$J_1 = Q_2$




$K_1 = Q_3 Q_0$




$J_0 = \bar{Q}_3 + Q_1$




$K_0 = \bar{Q}_1 + Q_3$