DATE: PAGE:

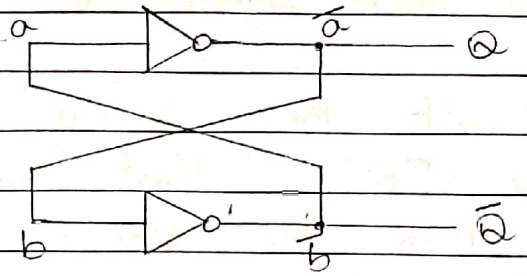
MODULE-3

FLIP-FLOPS AND THIER APPLICATIONS

* In sequential circuit the o/p at a given time depends upon the i/p at that time as well as on the previous i/p's. Hence the previous i/p sequence needs to be stored thereby we can say that sequential circuits always have a memory associated with them.
* Normally the operation of a combinational ckt is represented by a truth table.
* The operation of a sequential ckt is expressed by their states.
* The states are internal state is a set of signals at given points in the sequential ckt.
* for eg: In a counter, the counts will represent the states, in this case a 3-bit counter will be discribed by the sequence of its 8 possible states, where the present state & the present i/p will decide the next state.

* The basic memory element in a sequential circuit is the flip-flop.
* flip-flop is a sequential circuit which can store 1 or 0 indefinately.
* Thus a flip-flop is said to posses two stable states, it continues indefinately in one of these two stable states, depending on the i/p & o/p states.

Basic bistable element:



The digital logic ckt which can indefinately store a logic 0 or logic 1 is called as basic bistable element as shown as above.

Now let us assume that when power is switched on, on the ckt the i/p is at logic 1. $\therefore \bar{a} = 0$ & thus $Q = 0$. This implies $b = 0$ & that gives $\bar{b} = 1$ (or) $\bar{Q} = 1$. Hence $Q = 0$ & $\bar{Q} = 1$. & the ckt continues in this state until power is switched off. This is one of the two stable states of the ckt.

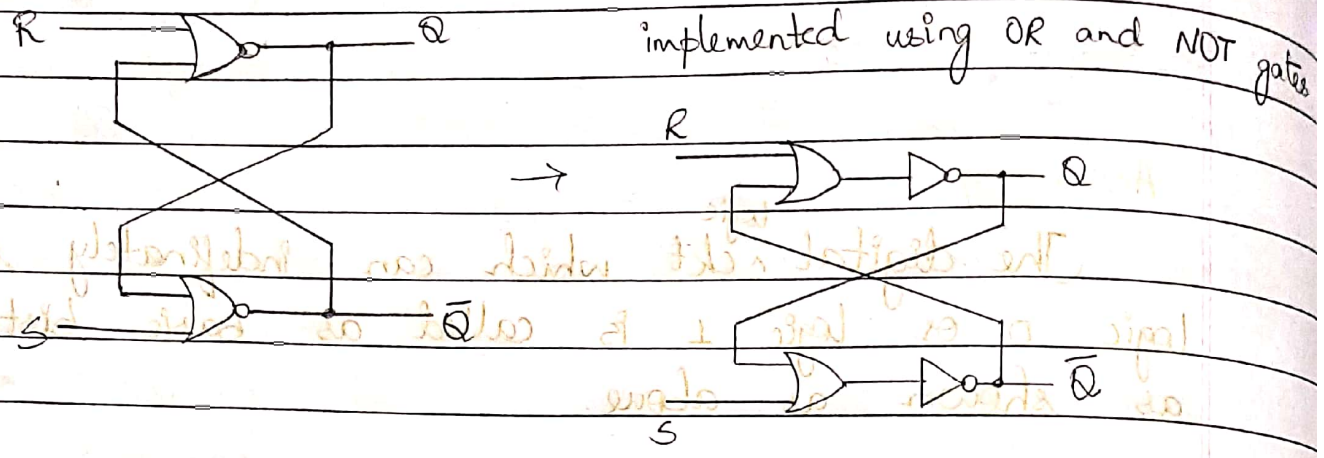
On the other hand, let $a = 0$ when the power is switched ON, on the ckt the i/p is at logic 0 $\therefore \bar{a} = 1$ & thus $Q = 1$. $\Rightarrow b = 1$ & that gives $\bar{b} = 0$ (or) $\bar{Q} = 0$. Hence $Q = 1$ & $\bar{Q} = 0$. & this is the other stable state of the ckt in which it is indefinately remains until power to the ckt is switched off. Hence the above ~~bita~~ bistable element is said to be in 1-state when it stores $Q = 1$. & it is said to be in 0-state when it stores $Q = 0$.

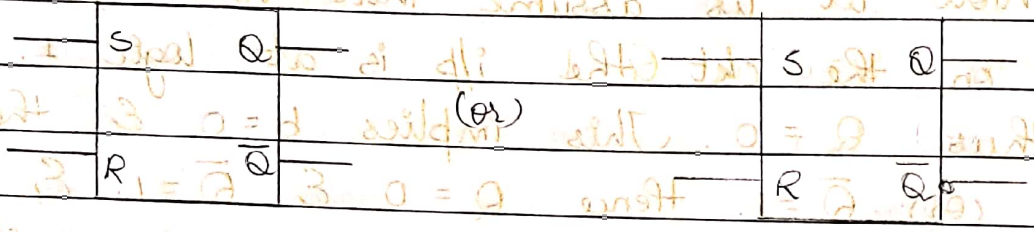
Latches:

A latch is a class of flip-flop whose o/p responds immediately to appropriate changes in the i/p inspite of the presence of an enable i/p or clock i/p. There are different types of latches based on

types of i/p's & enable lines.

1) S-R latch:





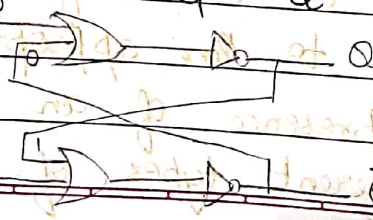
Row no.	S	R	Q^*_{t+}	$\bar{Q}^*_{t+}$
0	0	0	Q	$\bar{Q}$
1	0	1	0	1
2	1	0	1	0
3	1	1	0	0*

* → forbidden

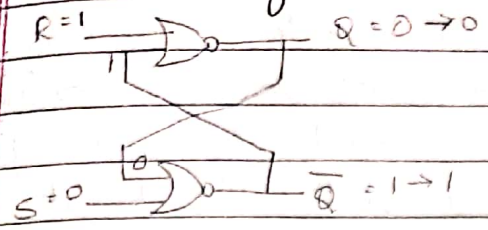
An S-R latch is obtained by cross-coupling two NOR gates where S & R stands for set & reset inputs of the latch while Q & $\bar{Q}$ are the two outputs.

In S-R latch, S=R=0 i.e., SR=00.

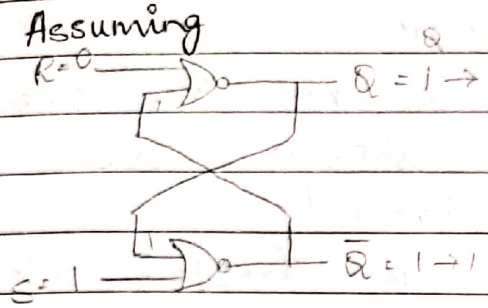
Assuming Q=0 & $\bar{Q}$=1.



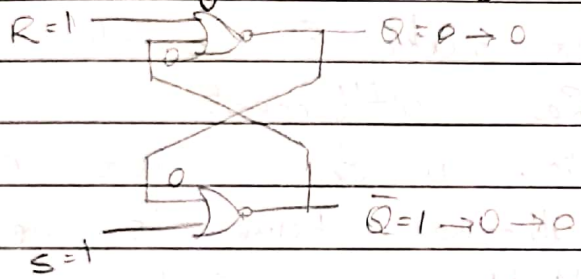
Assuming $\bar{Q} = 1$ & $Q = 0$



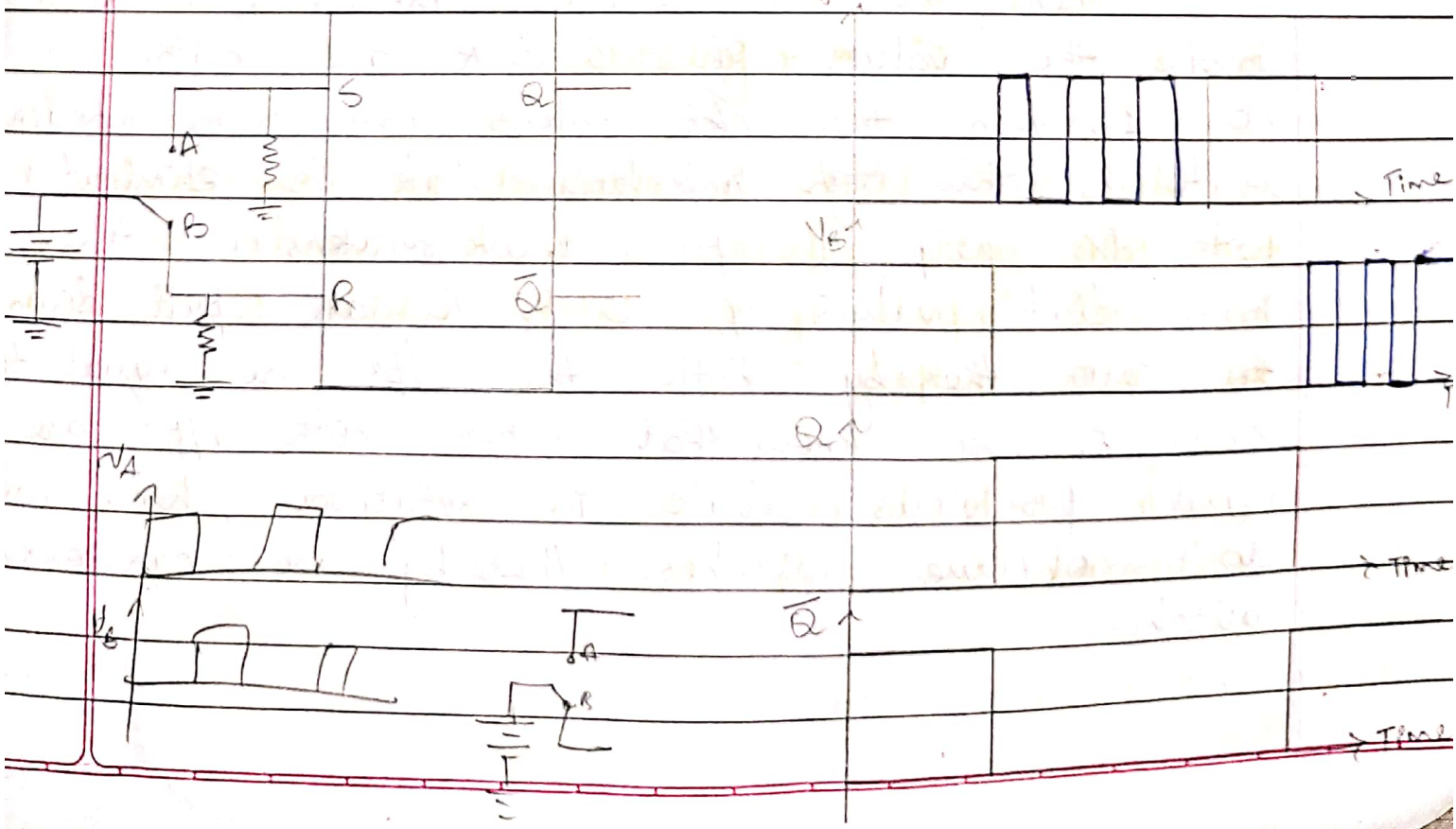
Assuming



Assuming $Q = 0$ & $\bar{Q} = 1$



Application of SR latch :



In the normal switch the center contact is at 'B', a vtg of 'V' volts is connected to point B. Its waveform is shown below. Later on the switch is moved towards point A now. point B is have '0' volts & point A is having 'V' volts. If this action is performed continously we get '0V0V'.

But in reality this is not the sinario. The real picture is as shown, coz we known that metals have elasticity.

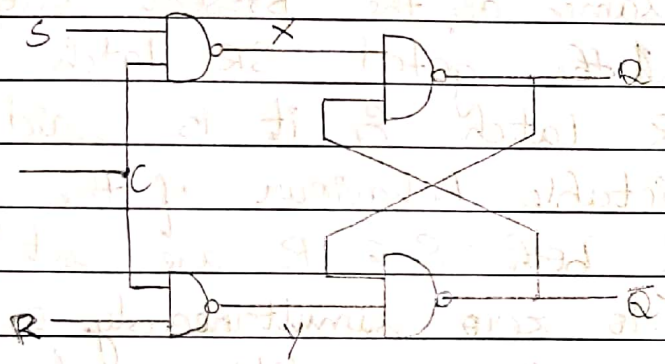
for eg: A ball which is dropped onto floor. As soon as the ball is thrown towards the floor, the ball will not settle on the floor immediately, many make & break action takes place before the settlement.

Between the logic 0 & 1 we observe the glitches, these (contact bounce) glitches have to be eliminated with the help of switch debouncer ckt using SR latch.

When the centre contact moves from B to A inblw the values of S & R are neither 0 nor 1, then the ckt enters into high impedence state & this high impedence state is eliminated with the help of pull down resistors i.e., the intermediate values of S & R are pull down to zero, thereby both the i/ps are equal to zero & we know that when both i/p's are equal perivious state is retained, hence we don't observe glitches. Thereby we can overcome glitches.

Eg1: Sketch the waveforms at Q & $\bar{Q}$ if the following signals are applied at the S & R inputs of a SR latch.

Gated SR latch:



In the latches discussed so far, the output responds immediately to the changes in the input. The input of such latches are said to be asynchronous. In many digital systems it is desirable that the circuit responds only at or during some prescribed time, decided by another i/p called as Enable, gate or clock input.

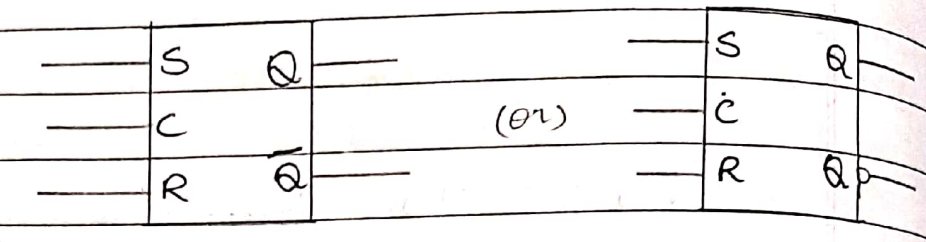
Inputs which cause a response only in synchronous with an enable or clock inputs are called as synchronous inputs.

A gated SR latch has synchronous inputs S & R along with a clock or enable input 'C'.

A gated SR latch is constructed using $\bar{S}\bar{R}$ latch together with NAND gates as shown above.

S	R	C	Q^+	$\bar{Q}^+$
0	0	1	Q	$\bar{Q}$
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1*
x	x	0	Q	$\bar{Q}$

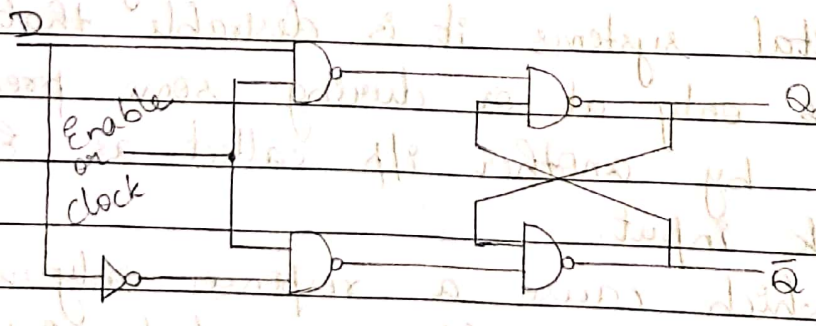
*forbidden

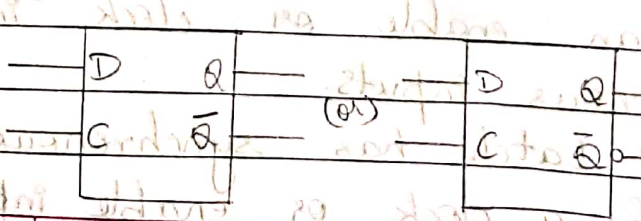


When clock C = 0, X & y = 1. whatever may be S & R output [S = x, R = x], then the next outputs are same as the present outputs.

When C = 1, the gated SR latch behaves like a normal SR latch & it is said to be enabled. The unpredictable behaviour of the gated SR latch results if both S & R are set to 1. or if S & R go to zero simultaneously or if clock 'C' goes to zero. These conditions should be avoided.

Gated D-latch:





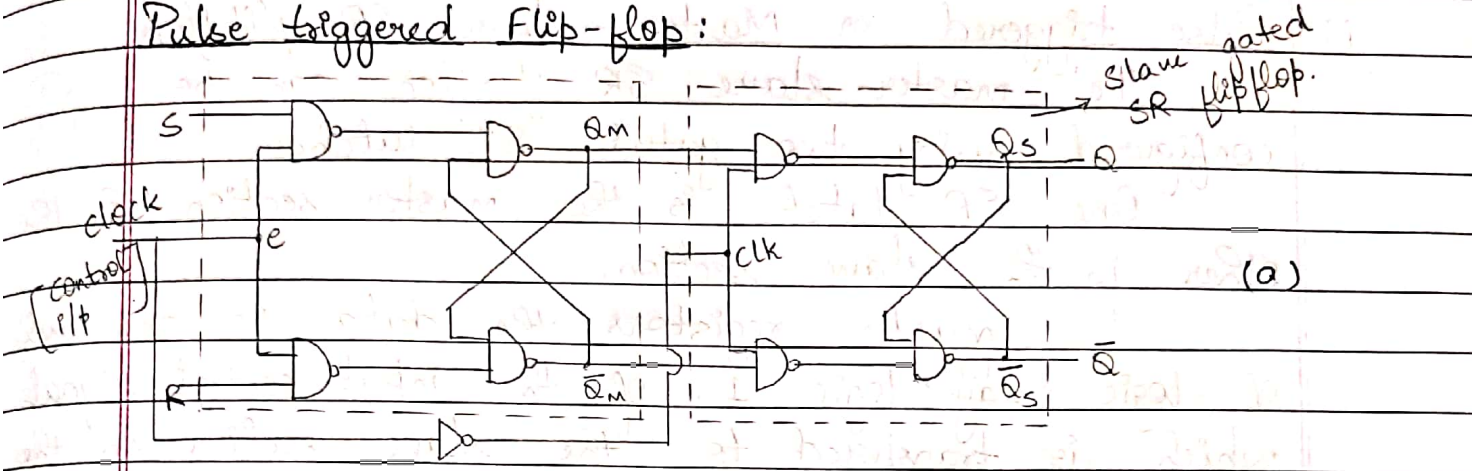
D	C	Q^+	$\bar{Q}^+$
0	1	0	1
1	1	1	0
x	0	Q	$\bar{Q}$

DATE: PAGE:

The forbidden o/p condition in S=1 & R=1 in a $\bar{S}\bar{R}$ latch can be avoided if the ckt is slightly modified. This modified ckt is called gated D-latch.

—+—

Pulse triggered Flip-flop:



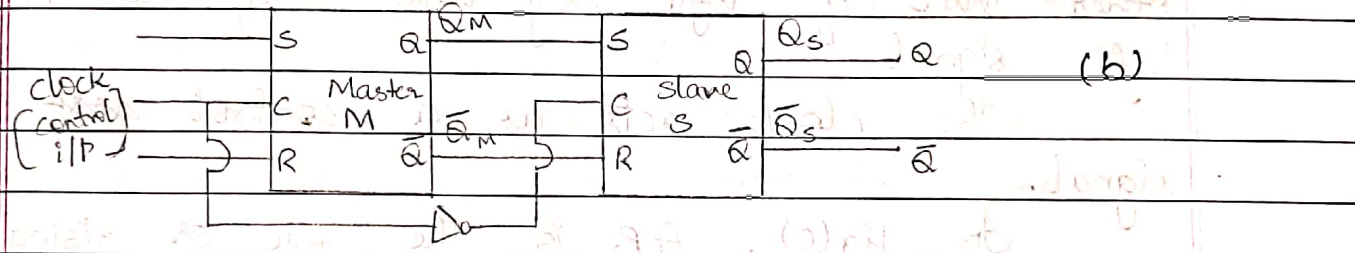


fig- Typical clock pulse

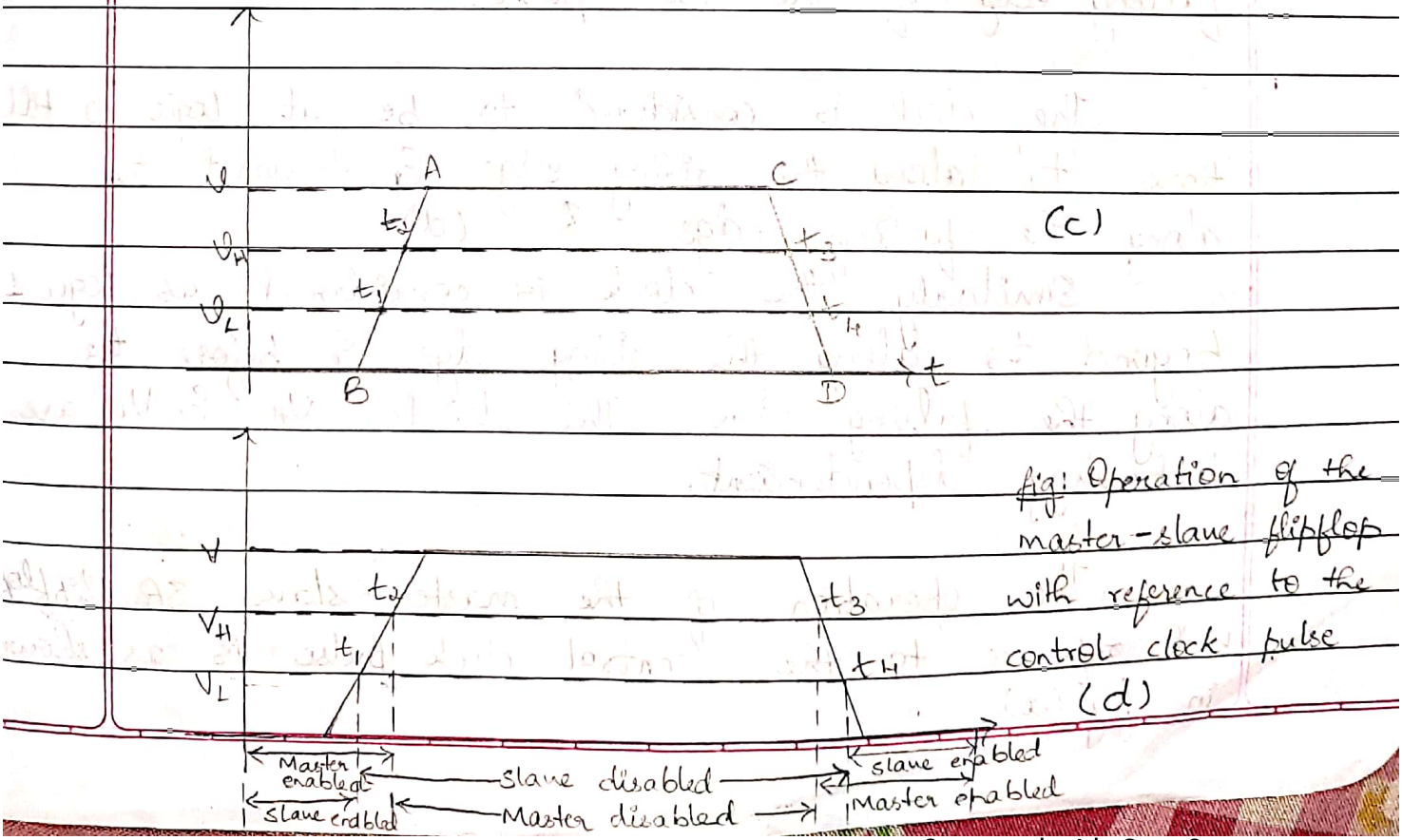


fig: Operation of the master-slave flipflop with reference to the control clock pulse

The pulse triggered flip flops are also called as Master-slave flip-flops. The outputs of the pulse triggered flip-flops responds only when changes takes place on their control input line.

(i) Pulse triggered or Master-slave SR flipflop:

The master-slave SR flipflop can be configured using two gated SR latches.

One SR latch is the master section & the other is the slave section.

The master registors the data on one level of logic i.e., say logic 1 of the input control signal which is transfered to the slave section on the other level of logic say logic 0 of the input logic signal.

The clock acts as a control i/p signal.

In fig(c), AB is the +ve or rising edge of the clock pluse & CD is the -ve or falling edge of the clock pulse.

The clock is considered to be at logic 0 till time 't_1' along the rising edge & beyond t_4 along the falling edge. & (d)

Similarly the clock is considered as logic 1 beyond t_2 along the rising edge & before t_3 along the falling edge. The levels V_H & V_L are technology dependendent.

The operation of the master-slave SR flipflop with reference to the control clock pulse is as shown in fig (d).

DATE: PAGE:

We have the s

S	R	C	Q^+	$\bar{Q}^+$
0	0	1	Q	$\bar{Q}$
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1*
x	x	0	Q	$\bar{Q}$

*forbidden

M=0

D

fig(d)

During the period when clock is at logic '0' the master is disabled while the slave is enabled. coz the inverted clock is applied to the slave flipflop. The Q_M & $\bar{Q}_M$ appear at S & R inputs of the salve. which is enabled. with Q_S equal to Q_M, while $\bar{Q}_S$ equal to $\bar{Q}_M$.

In other words the state of the master is transfered to the slave when the clock is at logic 0.

With reference to the clock pulse in fig(c) during the rising edge of the clock, the slave gets disabled at time t_1. coz beyond t_1 clock 'C' is no longer in logic 0.

Beyond t_1 in the rising edge the slave is disabled, the master gets enabled at t_2. when the clock is recognised to be at logic 1 level. Now the master flipflop responds to it S & R inputs while the slave retains it previous state. coz it is already disabled.

The master gets disabled at t_3 on the falling edge of the clock while the slave gets enabled at t_4. when once again the clock is set at logic 0. Logic 0 at the master means the clock

is at logic 1 of the slave.

The state of the master is once again transfered to slave.

We observee that [from fig(d)] both the master & slave are disabled b/w t_1 & t_2 and also b/w t_3 & t_4.

We can observe that the master can change its state as long as its clock input remains at logic 1, while the slave changes its state only along the falling edge of the clock at t_4.

The state table for the master-slave SR flipflop can be written as below.

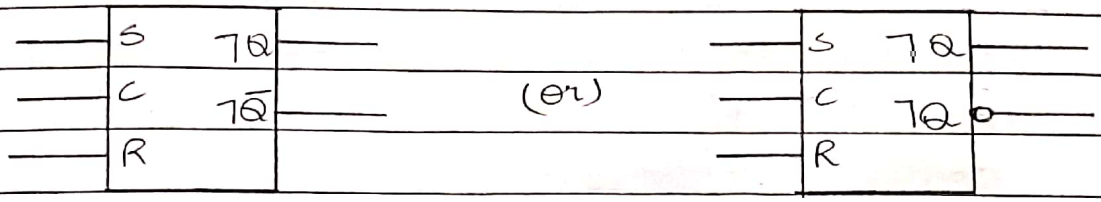
S	R	C	Q^+	$\bar{Q}^+$
0	0	⊓	Q	$\bar{Q}$
0	1	⊓	0	1
1	0	⊓	1	0
1	1	⊓	Not defined	
x	x	0	Q	$\bar{Q}$

The pulses in the table represent the fact that the master is enabled as long as the clock is at 1 & the state of the master is transfered to slave when the clock begins to go to zero.

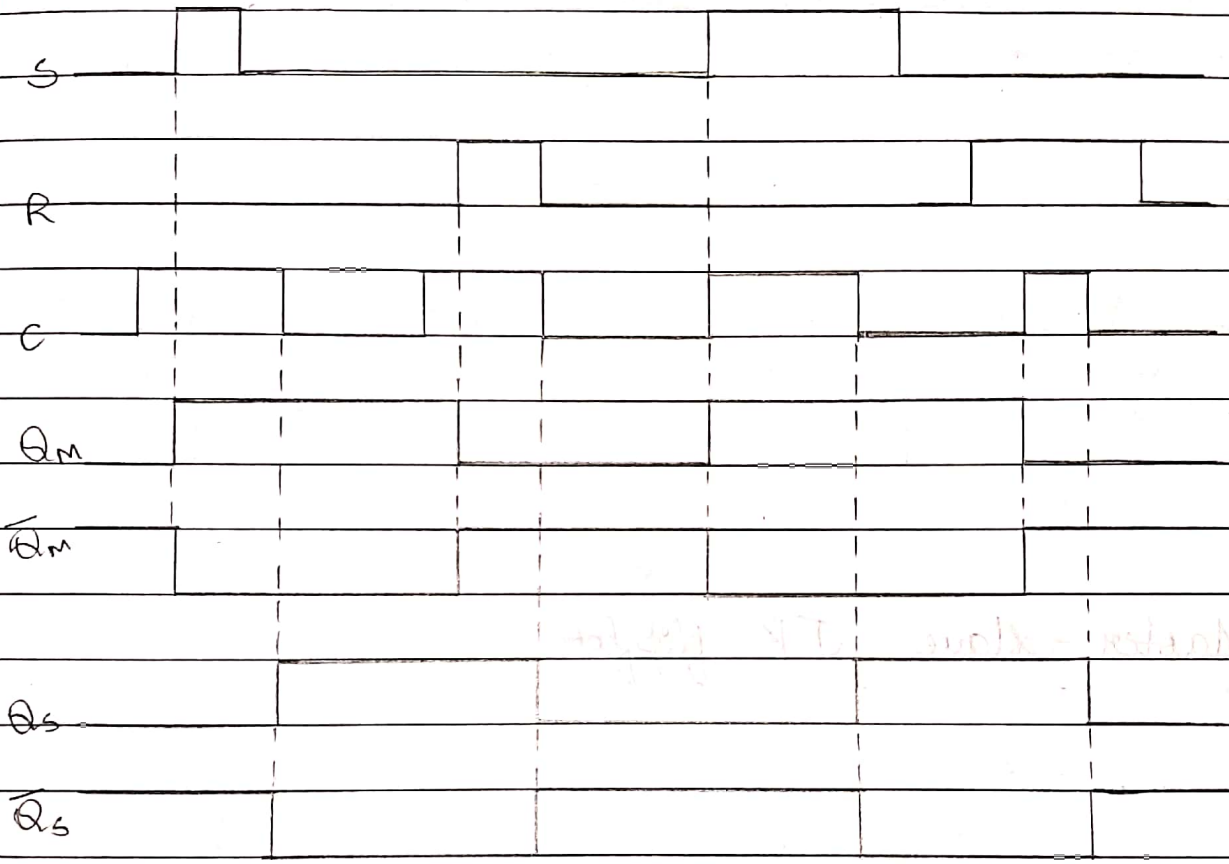
When S = R = 1, the state of the master & consequently the state of the slave becomes unpredictable. Since the state of the master depends on the period of time for which clock is at logic 1 & the state of the slave changes at the falling edge. These flipflops are also called as Master-slave flipflops or Pulse triggered flipflops.

DATE: PAGE:

Symbol & waveforms:

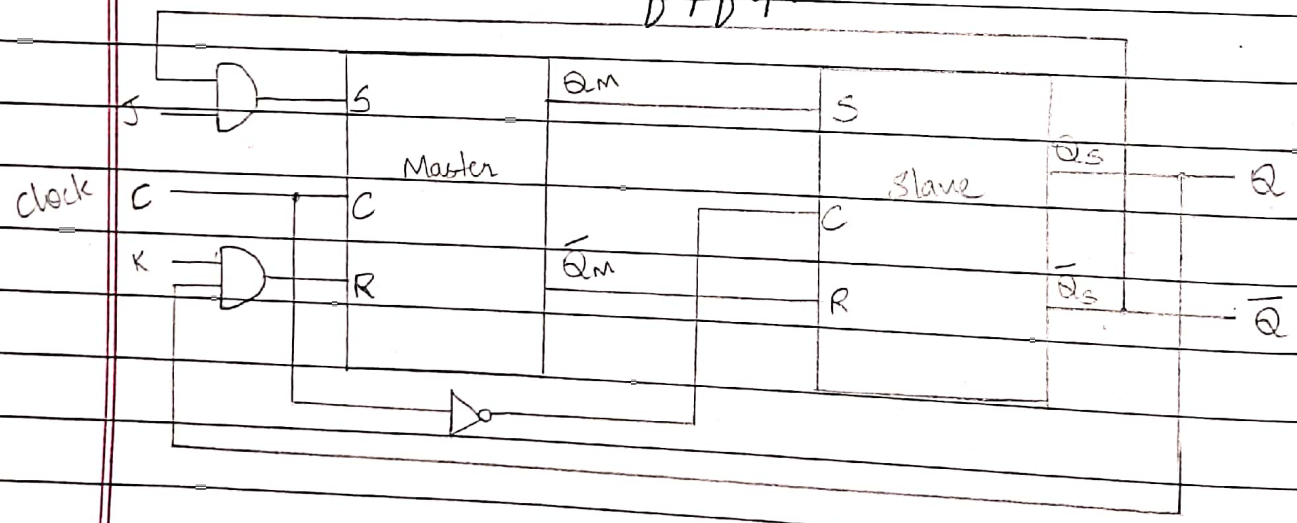


1)



2)

Master-slave JK flipflop:



Row no.	J	K	C	Q^+	$\bar{Q}^+$
0	0	0	⊓	Q	$\bar{Q}$
1	0	1	⊓	0	1
2	1	0	⊓	1	0
3	1	1	⊓	$\bar{Q}$	Q
4	X	X	0	Q	$\bar{Q}$

DATE: PAGE:

We have seen that a gated SR latch & even master-slave SR flipflops outputs are undefined when the inputs are S=R=1 this undefined condition must be eliminated.

We have seen that the undefined condition of a gated SR latch is eliminated by gated D latch. We shall now see how a master-slave JK flipflop eliminates the undefined condition of a master-slave SR F/F.

Construction:

The master-slave JK F/F consists of two gated SR F/F along with two AND gates, one for the J input & the other for the K input. The symbol 'T' is called the postponed output indicator. It is used to indicate that the o/p change is postponed until the end of the clock pulse period. The function table & state table is as shown above.

Consider J=K=1 & let Q=1 & $\bar{Q}$=0, thus o/p of AND gate 'A' goes to zero. & o/p of AND gate 'B' goes to 1. i.e., S=0 & R=1 at Master flipflop. Hence when clock goes high, the master resets to Q_M=0 & $\bar{Q}_M$=1.

On the next falling edge of the clock, the state of the master is transfered to slave thereby Q_S=0 & $\bar{Q}_S$=1. i.e., Q=0 or $\bar{Q}$=1 i.e., compliment of the initially assumed values of Q & $\bar{Q}$. Hence J=K=1 input has caused the o/p to toggle.

Now for J=K=1 we initially assume Q=0 & $\bar{Q}$=1. this makes the o/p of AND gate 'A' to be equal to 1

& o/p of AND gate 'B' to be equal to zero. Thus the i/p to master SR flipflop is now equal to S=1 & R=0, and when clock is high i.e., C=1 the master sets $Q_m = 1$ & $\bar{Q}_m = 0$. & next when clock goes low [to zero] this state of the master is now transfered to slave making $Q_s = 1$ & $\bar{Q}_s = 0$ i.e., $Q = 1$ & $\bar{Q} = 0$. [This is the compliment of the initially assumed values of Q & $\bar{Q}$, this toggles due to J=K=1.

Now with J=K=0, both AND gates A & B o/p's have zero. & it corresponds to S=R=0 in master when clock goes high the master will register the o/p as Q_m & $\bar{Q}_m$. & when clock goes to low the Q_m & $\bar{Q}_m$ goes to low [slave] then slave o/p becomes Q & $\bar{Q}$.

Since there is no change in the next o/p.

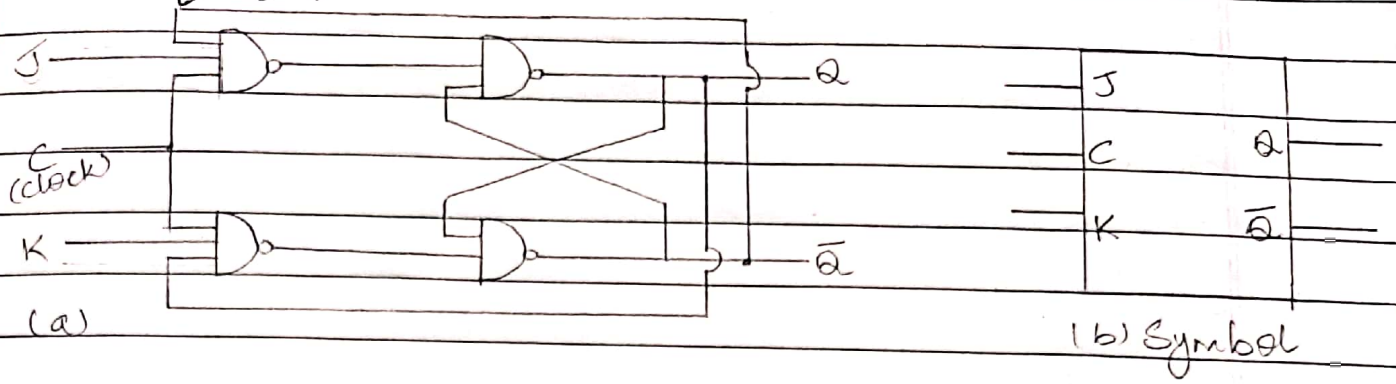
Now with i/p's J=0 & K=1 similar to S=0 & R=1 condition of the gate.

With inputs J=1 =K=1

We shall analyze the operation of MS JK F/F by assuming initially $Q = 1$, $\bar{Q} = 0$. Thus both the AND gates o/p is equal to zero, the o/p remains unchanged. Now Let J=1 & K=0 but assuming initially condition of Q=0 & $\bar{Q} = 1$. AND gates of J=1 & K=0, this makes S=1, R=0 then master sets $Q_m = 1$ & $\bar{Q}_m = 0$. this same o/p is transfered to slave i.e., $Q^+ = 1$ & $\bar{Q}^+ = 0$.

DATE: PAGE:

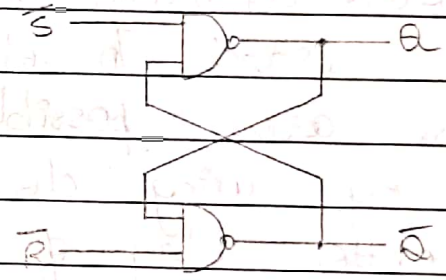
J-K flipflops:



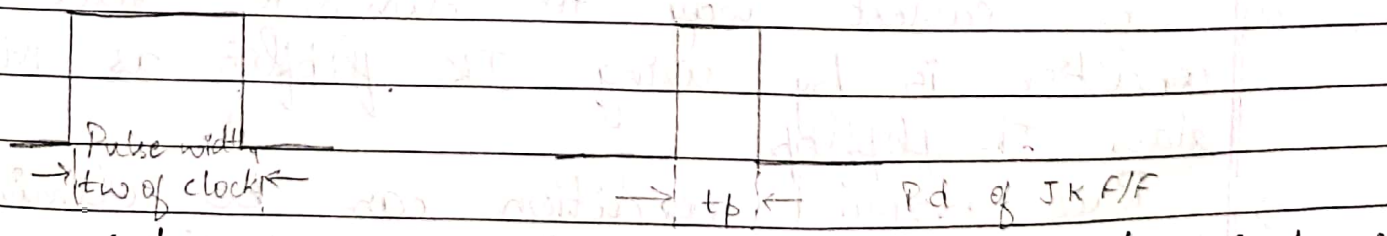
J	K	Q^+	$\bar{Q}^+$	
0	0	Q	$\bar{Q}$	→ with clock enabled i.e., C = 1
0	1	0	1	
1	0	1	0	
1	1	$\bar{Q}$	Q	

→ SR flipflops:-

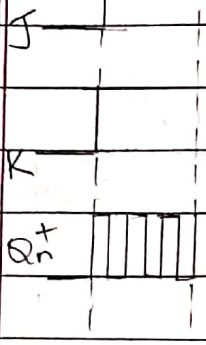
Row no.	$\bar{S}$	$\bar{R}$	Q^+	$\bar{Q}^+$	
0	0	0	1	1*	→ forbidden
1	0	1	1	0	
2	1	0	0	1	
3	1	1	Q	$\bar{Q}$	



The main cause for the race arround condition of the JK flipflop is that the propogation delay of the JK flipflop is less than the clock pulse width.



Let tw be the width of clock pulse & tp is propogation pulse delay. The race arround condition at J=K=1 occurs when tp ≤ tw. The race arround condition is shown in the waveform above.



When J = K = 1 & clockpulse width $t_w > t_p$ of JK flipflop, then the o/p changes several times within a single clockpulse period is called race arround condition. This race arround condition should be avoided. To eliminate race arround condition there are 3 possible ways:

1) By using clock pulse of smaller width than propogation delay of JK flipflop i.e., $t_w < t_p$. Since propogation delay is very small & it is interms of nanoseconds. To adjust $t_w < t_p$ is very difficult for eg: If t_p = 1 nano second the clock signal frequency require to avoid the race arround condition is $f = \frac{1}{2t_w} = \frac{1}{2 \times 10^{-9}} = 0.5$ GHz. (gegaHz)

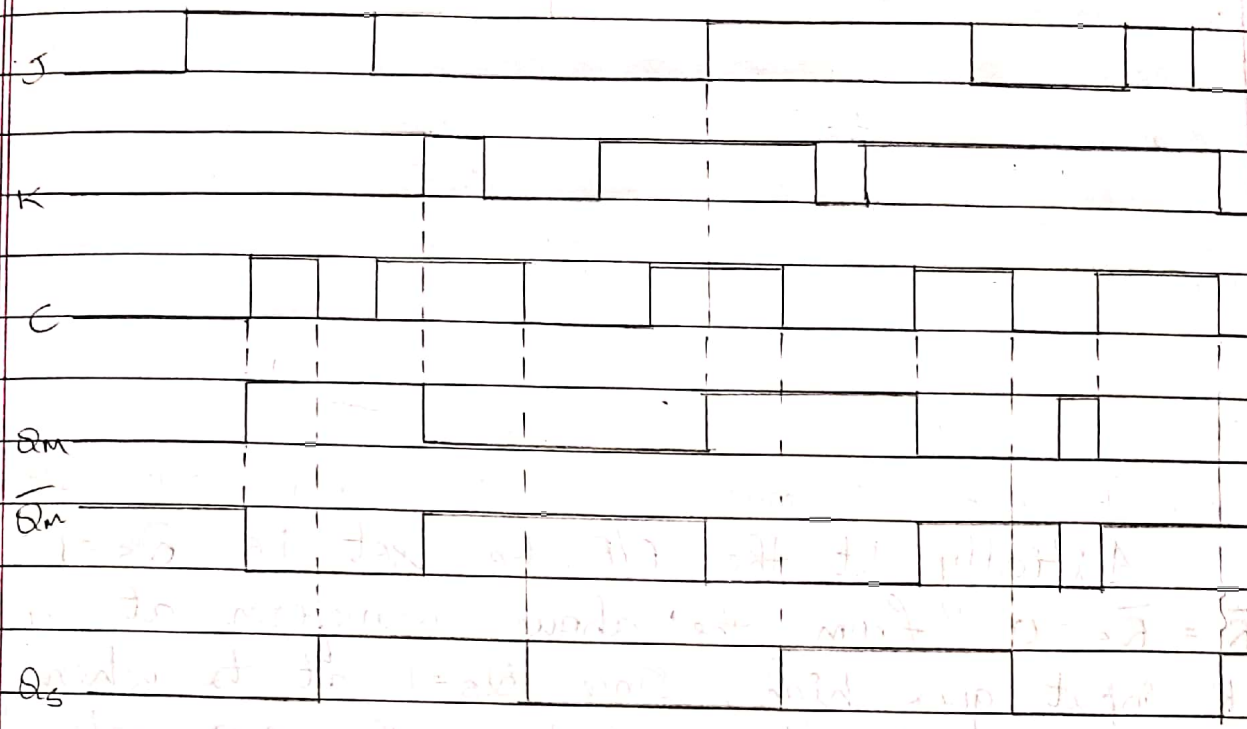
2) The easiest way to eliminate race arround condition is by using JK flipflop as Master-slave JK flipflop.

3) Race arround condition can be eliminated by using edge triggered JK flipflops coz there is no enough time to race but in JK flipflop we can have edge triggered JK F/F but not pulse triggered JK F/F. But only we can

DATE: PAGE:

have pulse triggered Master-slave JK F/F.

Continuation of Master-slave JK F/F:



Waveform of M-S JK F/F

Zero's & One's catching problem in JK-flipflop:

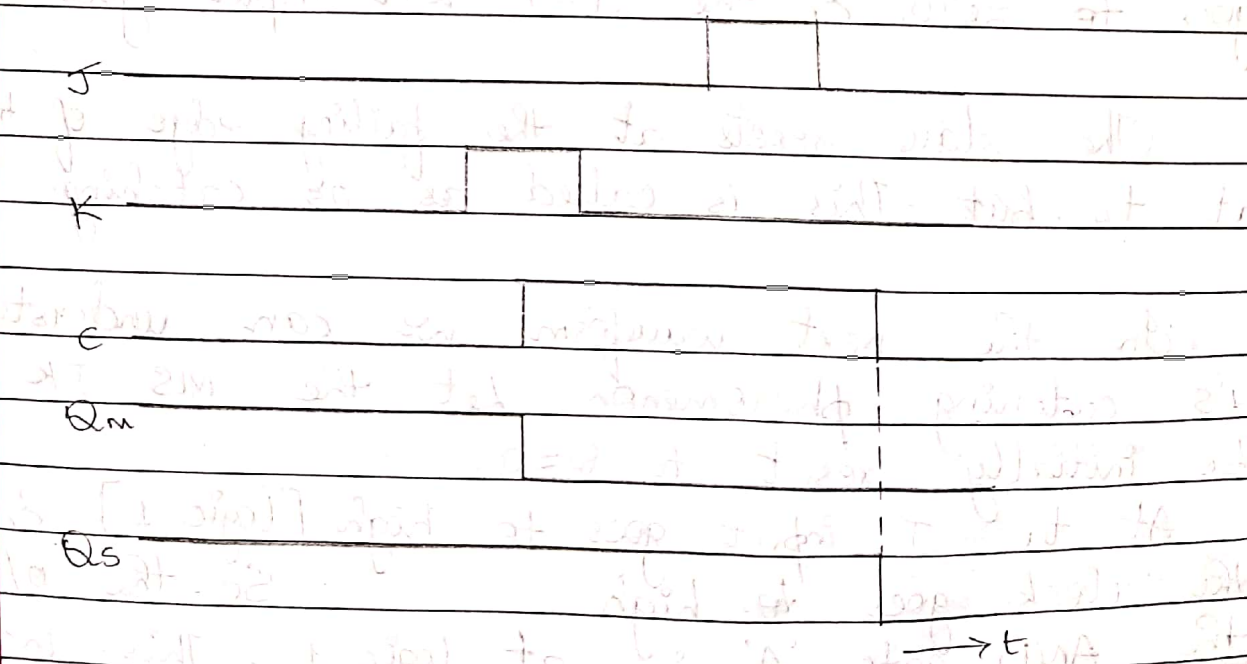


fig: Response of a Master-Slave F/F to illustrate 0's catching.

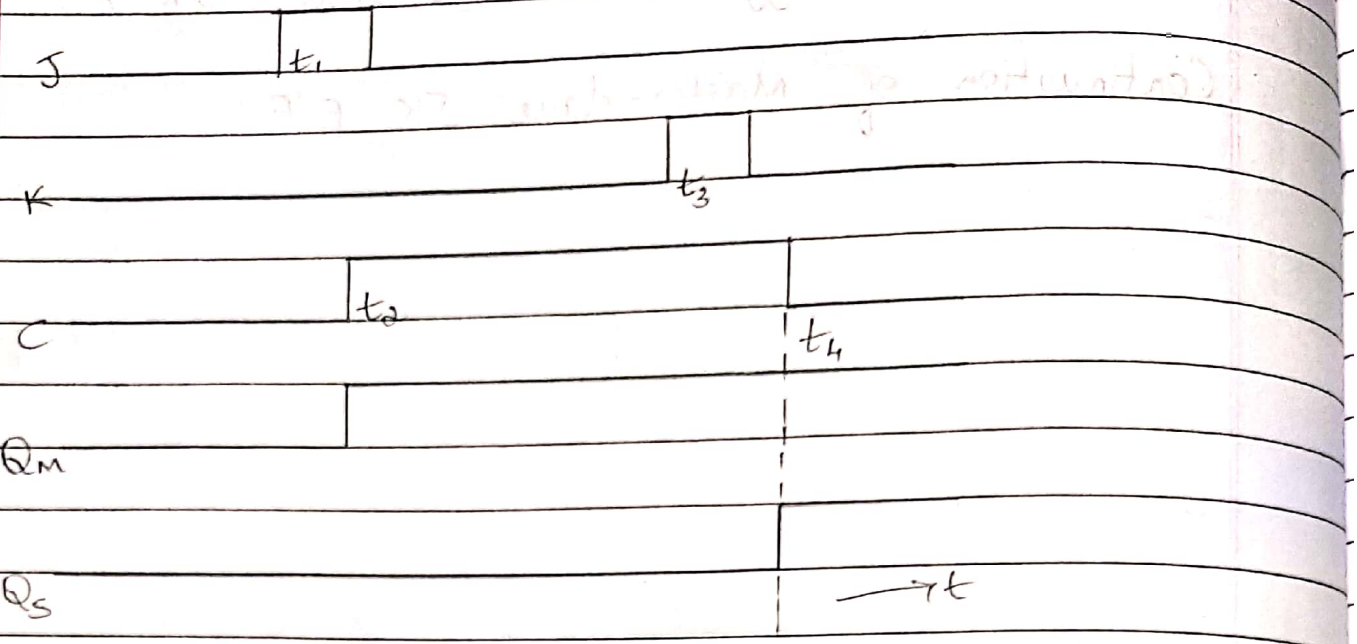


fig: Response of master-slave f/f to illustrate 1's catching

Initially let the F/F be set i.e., $Q_S = 1 = Q$ & $\overline{Q} = \overline{Q}_S = 0$. from the above waveform at t_1 the K input goes high. Since $Q_S = 1$ at t_2 when the clock goes high, the output of the AND gate 'b' goes to 1 & the master resets as shown in the Q_M waveform. Note down that the Q_S is still high i.e., at logic 1, now at t_3 'J' input goes high & $\overline{Q}_S$ is still zero, the o/p of the AND gate 'A' goes to zero. & the J=1, K=0 input goes unrecognised.

The slave resets at the falling edge of the clock at t_4. But This is called as 0's catching.

In the next waveform we can understand 1's catching phenomenon. Let the MS JK F/F be initially reset to Q=0.

At t_1, J input goes to high [logic 1] & at t_2 the clock goes to high. So the o/p of the AND gate 'A' is at logic 1. This indicates the master sets. $\therefore Q_M = 1$, now Q_S is still zero

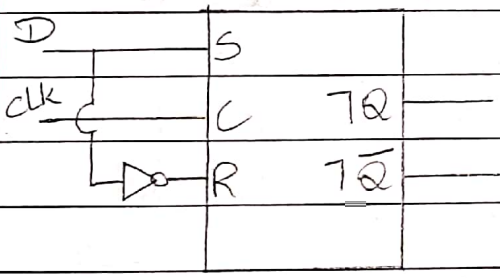
DATE: PAGE:

Thus it keeps AND gate 'b' disabled when 'k' becomes 1 at t_3. Thus at t_3, J=0 & k=1. So the output goes unrecognized. The slave sets at t_4 during the falling edge of the clock, this is called as 1's catching. Inorder to avoid 0's & 1's catching problems, the J & k input values ~~should~~ are kept constant when clock is at logic 1 i.e., when master is enabled.

Additional flipflops:

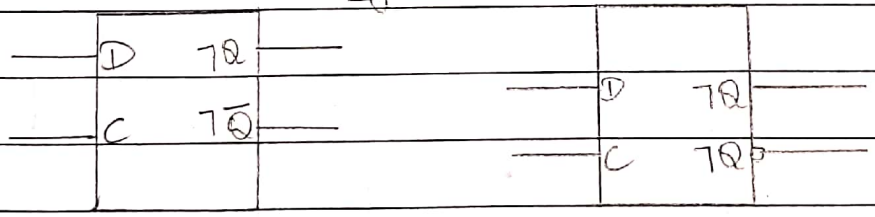
1) Master-slave D-F/F.
2) Master-slave T-F/F.

Master-slave D f/f:

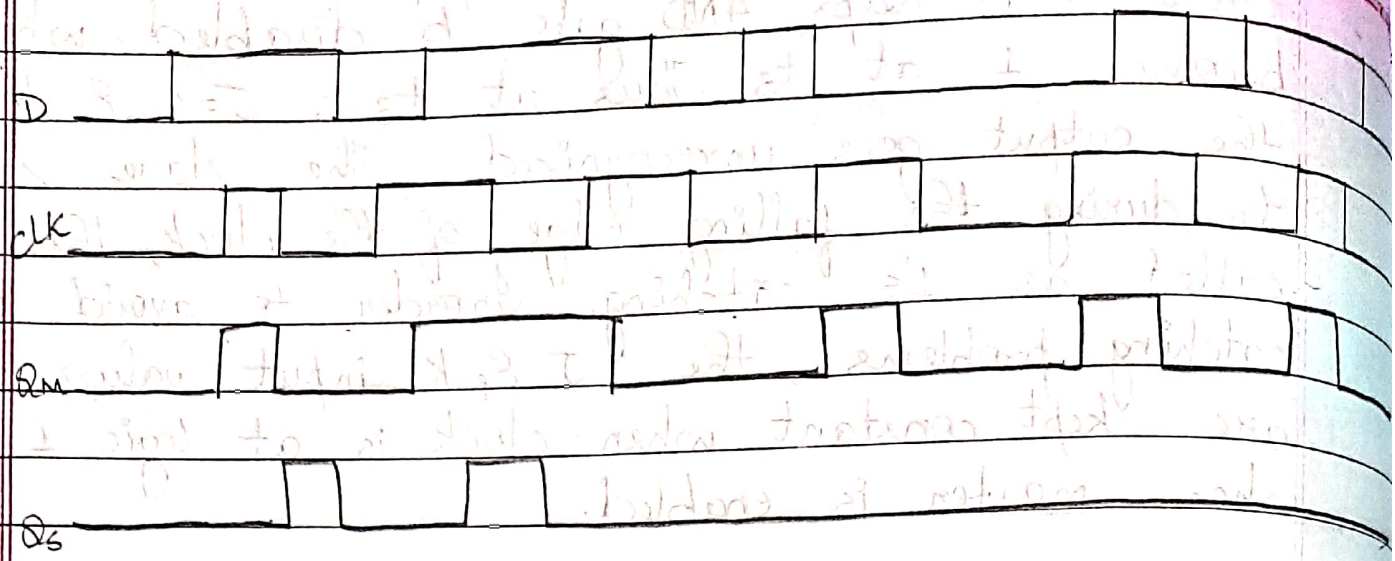


D	C	Q^+	$\bar{Q}^+$
0	⊓	0	1
1	⊓	1	0
x	0	Q	$\bar{Q}$

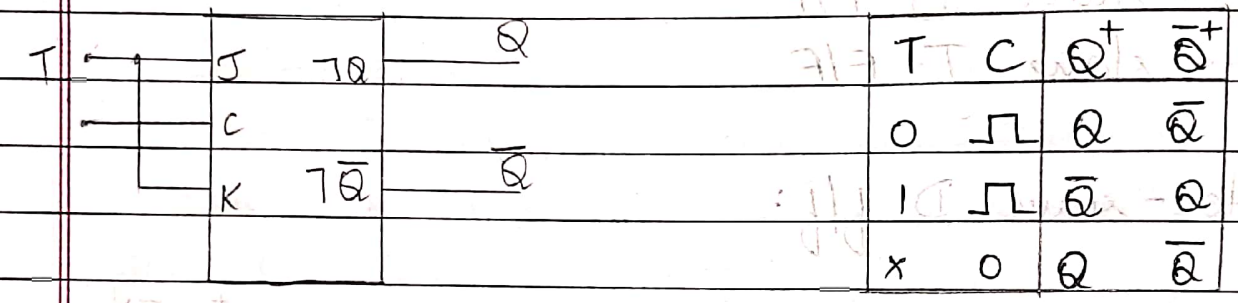
Symbol



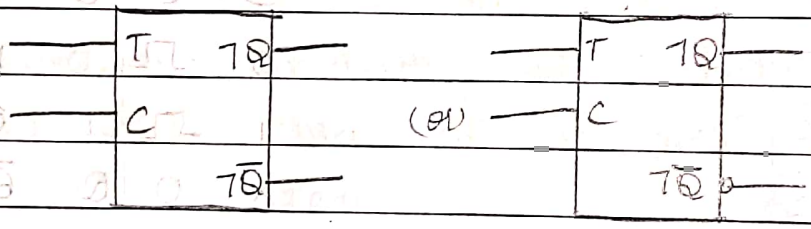
A master-slave D-f/f can be configured from master-slave SR f/f as shown above. The master f/f registers the i/p 'D' during the period when the clock is high & the state of the master is transfered to the slave when the clock goes low.



JK flf configured as T flf:



T	C	Q^+	$\bar{Q}^+$
0	↓	Q	$\bar{Q}$
1	↓	$\bar{Q}$	Q
x	0	Q	$\bar{Q}$



DATE: PAGE:

Characteristic eq^n for the next stage

We shall consider the function tables of the SR f/f, JK f/f, D f/f & T f/f. with the assumption that the control clock is applied. This means that we assume that these f/f's responds only with reference to a clock wheather it is pulse triggered or edge triggered.

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	— → Unpredictable

SR f/f

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

JK f/f

D	Q^+
0	0
1	1

D f/f

T	Q^+
0	Q
1	$\bar{Q}$

T f/f

Since the next state of these F/F depends on their present states, the functional tables will be written in the form of the next state tables as bellow.

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	—
1	1	1	—

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

The characteristic eqn of the F/F can be obtained by means of k-maps.

S \ RQ	00	01	11	10
0	0	1	0	0
1	1	1	X	X

J \ KQ	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$G_1 = \bar{S}\bar{R}Q + S\bar{R}Q$

$G_1 = \bar{R}Q$

$G_2 = S\bar{R}\bar{Q} + S\bar{R}Q + SRQ + SR\bar{Q}$

$G_2 = S\bar{R} + SR$

$G_2 = S$

$Q^+ = S + \bar{R}Q$

$G_1 = \bar{J}\bar{K}Q + J\bar{K}Q$

$G_1 = \bar{K}Q$

$G_2 = J\bar{K}\bar{Q} + JKQ$

$G_2 = J\bar{Q}$

$Q^+ = J\bar{Q} + \bar{K}Q$

D \ Q	0	1
0	0	0
1	1	1

T \ Q	0	1
0	0	1
1	1	0

$G_1 = D\bar{Q} + DQ$

$Q^+ = D$

$G_1 = \bar{T}Q + T\bar{Q}$

$Q^+ = T \oplus Q$

Registors:

Registors & counters are important applications of clocked F/F.

A collection of F/F in cascade is called as a registor.

Registors are used to store data in a digital system.

DATE: PAGE:

A cascade of 4 F/F configured as a registor can store 1 nibble of data [4 bits]

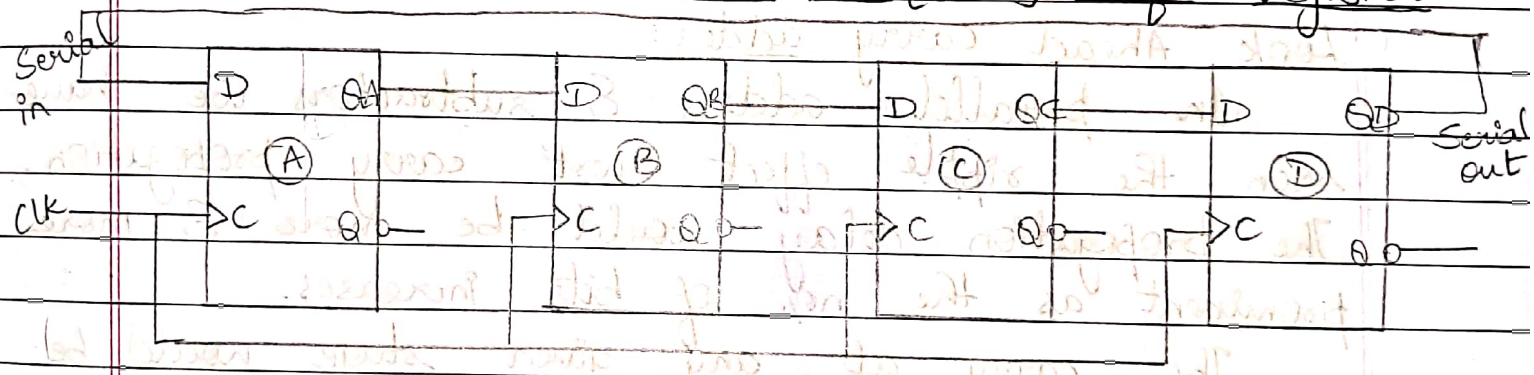
A 4 bit registor can store binary values from 0000 to 1111. These are called as contents or states of a registor, thus a 4 bit registor has 16 possible states.

Shift registors:

These are the registors which have the capicity to shift the data.

Shift is easier than rotate.

① 4 bit serial in serial out [SISO] shift registor:



The D inputs of each F/F is connected to the Q output of the perivious stage to the left.

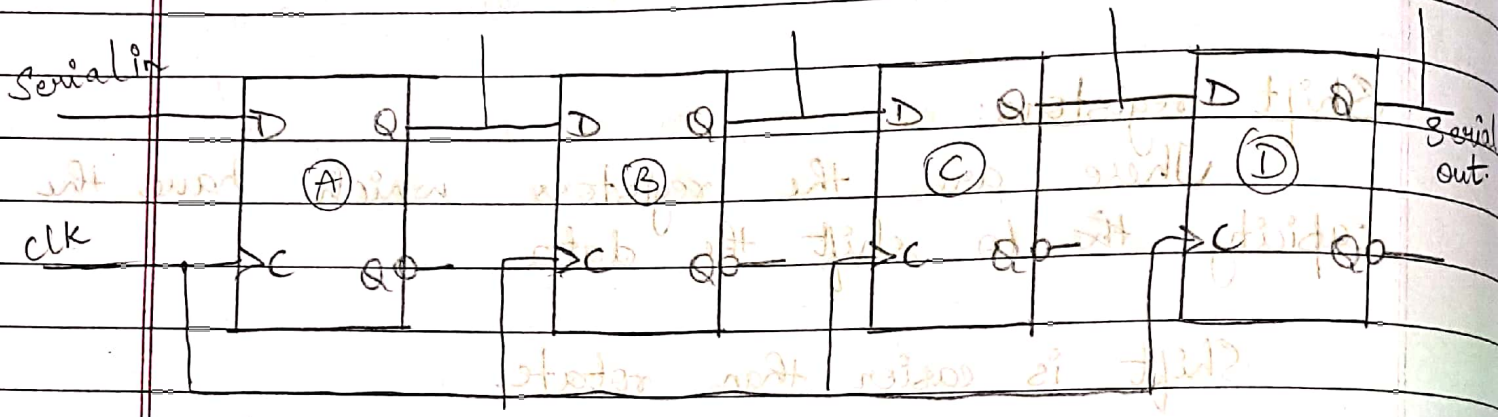
When a clock pulse is applied to the C input of F/F A, the data on the serial in line gets stored in F/F A. & appears at Q_A.

Every bit is shifted one position towards right The serial input is shifted into F/F A & the contents of F/F D gets shifted out & is lost.

In application where the data 1's input must be preserved then serial out should be completed to serial in, this is in the form of circular shift registors.

by the operation of rotation.
Eg: 1001 is stored in shift register after shifting process it becomes 1100.

4 bit serial in parallel out unidirectional shift registors:



Look Ahead carry adder:

In parallel adders & subtractors we have seen the ripple effect wrt carry propogation. The propogation delay would be more & more prominent as the no. of bits increases.

The carry at any given stage would be available only after the carry of the previous stage has been generated.

If we could ensure that the sum & carry output of any stage is not dependent on the previous stage then the ripple effect is eliminated & speed of the addition remarkably increases.

Sum eqn: $S = a_i \oplus b_i \oplus c_i$ & Carry at $(i+1)$ stage $C_{i+1} = a_ib_i + b_ic_i + a_ic_i$ then modifing this carry eqn we can write

$$C_{i+1} = a_ib_i + c_i(a_i + b_i) \rightarrow (1)$$

DATE: PAGE:

The term $a_i b_i$ relates to the carry formed at the i^{th} stage & is refered to as the carry generating function g_i. i.e., 'g_i' $= a_i b_i \rightarrow (2)$

The term $a_i + b_i$ relates to the carry C_i generated at the previous stage & thus $a_i + b_i$ is refered to as the carry propogating function, 'P_i'.

$$P_i = a_i + b_i \rightarrow (3)$$

Now observing that both P_i & g_i are functions of only the parallel inputs a_i & b_i and comparing eq^n's (1), (2), (3) we can write,

$$C_{i+1} = g_i + P_i C_i \rightarrow (4)$$

Now let us look at the carry generated at every stage of 4 bit parallel adder by setting $i=0$,

$$C_1 = g_0 + P_0 C_0 \rightarrow (5)$$

Now setting $i=1$ in eq^n (4) we get,

$$C_2 = g_1 + P_1 C_1.$$

Now substituting for C_1 from eq^n (5)

$$C_2 = g_1 + P_1 (g_0 + P_0 C_0)$$

$$C_2 = g_1 + P_1 g_0 + P_1 P_0 C_0 \rightarrow (6)$$

According to eq^n's (5) & (6), C_1 & C_2 are functions of only the parallel inputs & C_0. Now setting $i=2$ in eq^n (4) we get,

$$C_3 = g_2 + P_2 C_2$$

~~Now substitu setting i_2 in eq^n (4)~~

Now substituting for C_2 value.

$$C_3 = g_2 + P_2 (g_1 + P_1 g_0 + P_1 P_0 C_0)$$

$$C_3 = g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0 \rightarrow (7)$$

Now set $i = 3$ in eq (4).

$$C_4 = g_3 + P_3 C_3$$

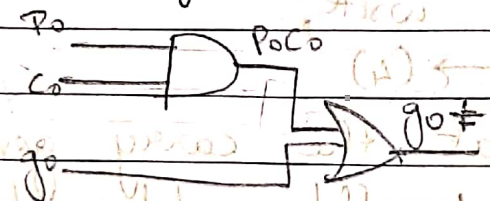
$$C_4 = g_3 + P_3(g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0)$$

$$C_4 = g_3 + P_3 g_2 + P_3 P_2 g_1 + P_3 P_2 P_1 g_0 + P_3 P_2 P_1 P_0 C_0 \rightarrow (8)$$

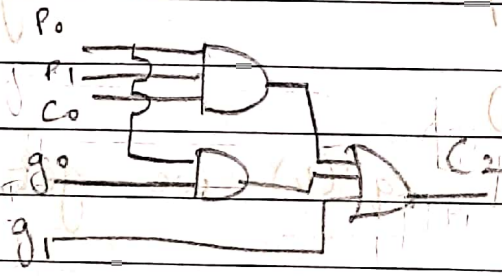
Eq^n (7) & (8) indicate that they are the functions of only parallel inputs & C_0)

The eq's for C_1 to C_4 can be implemented using logic gates

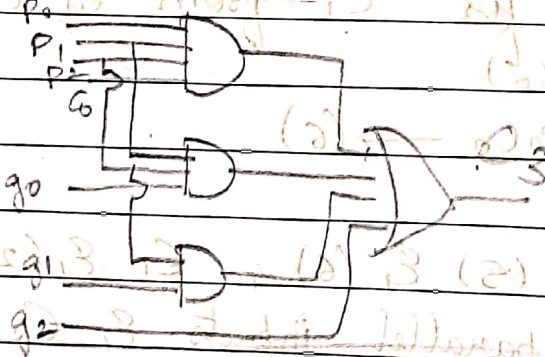
C_1



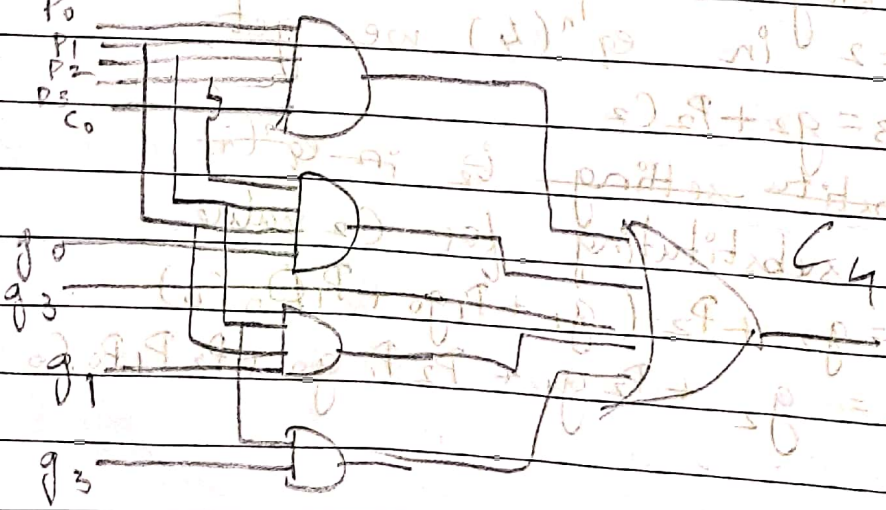
C_2



C_3

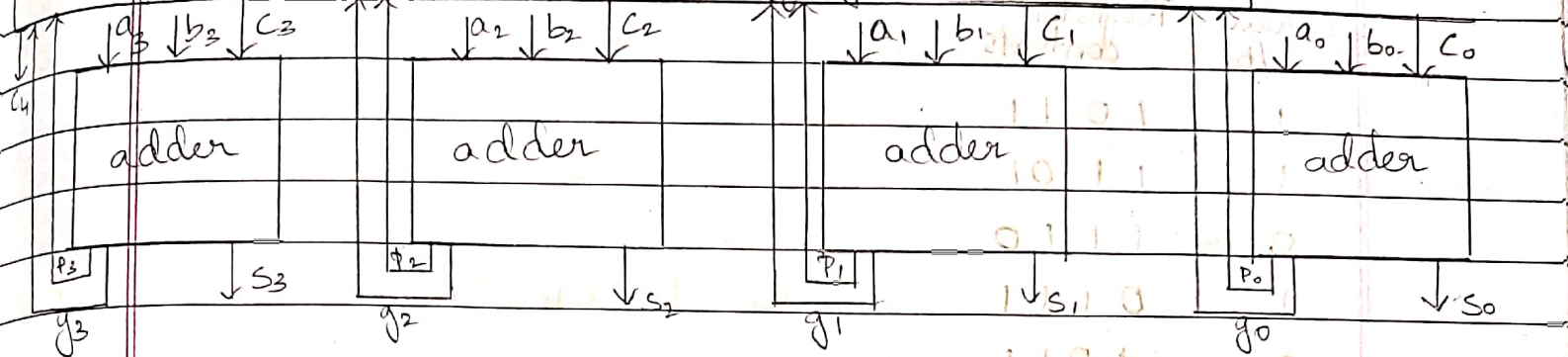


C_4

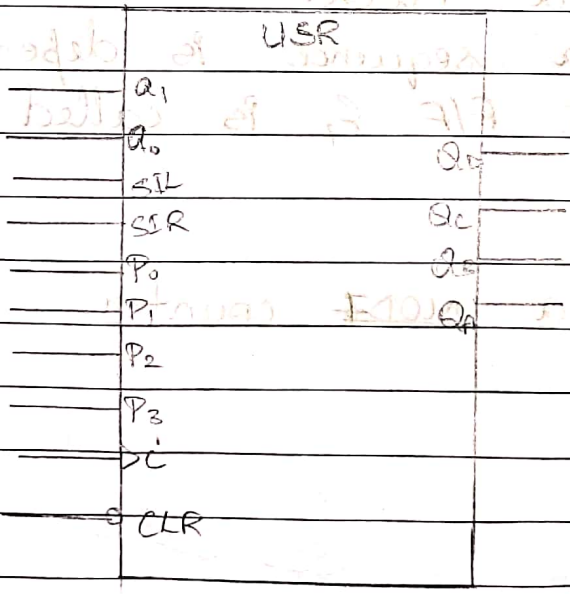


DATE: PAGE:

Look Ahead Carry adder:



a_1	a_0	Data line selected	Operation
0	0	d_0	Hold
0	1	d_1	shift right
1	0	d_2	Shift left
1	1	d_3	Parallel load



The given serial data is supplied at each +ve edge of the clock.

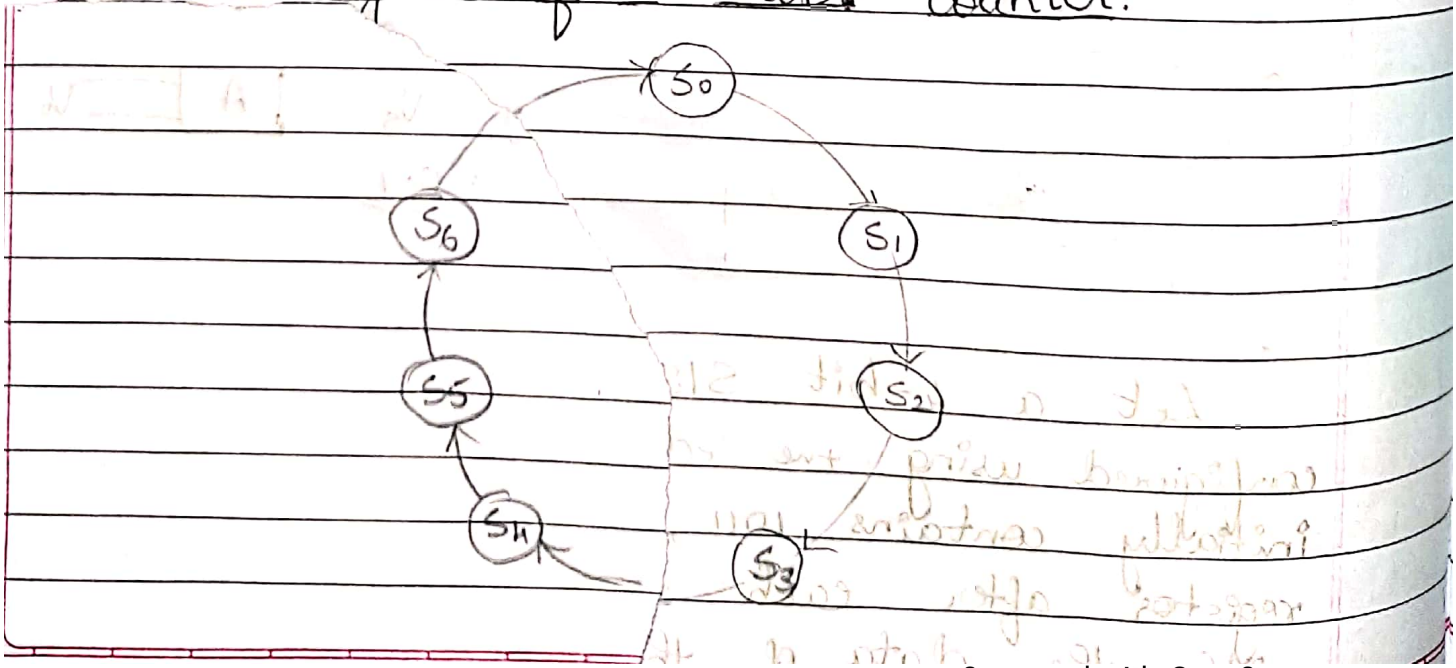
Serial i/p	Register contents
1	1011
1	1101
0	1110
1	0111
0	1011
1	0101
0	1010
	0101

Counters:

A counter is also a cascade of F/F configured to output as a specific sequence on application of a clock.

Each o/p of a sequence is dependent on the contents of the F/F & is called as state of the counters.

State diagram of a MOD7 counter.



A counter which counts from 0000 to 1001 [0-9] in decimal & then resets is called as Modulus 10 counter.

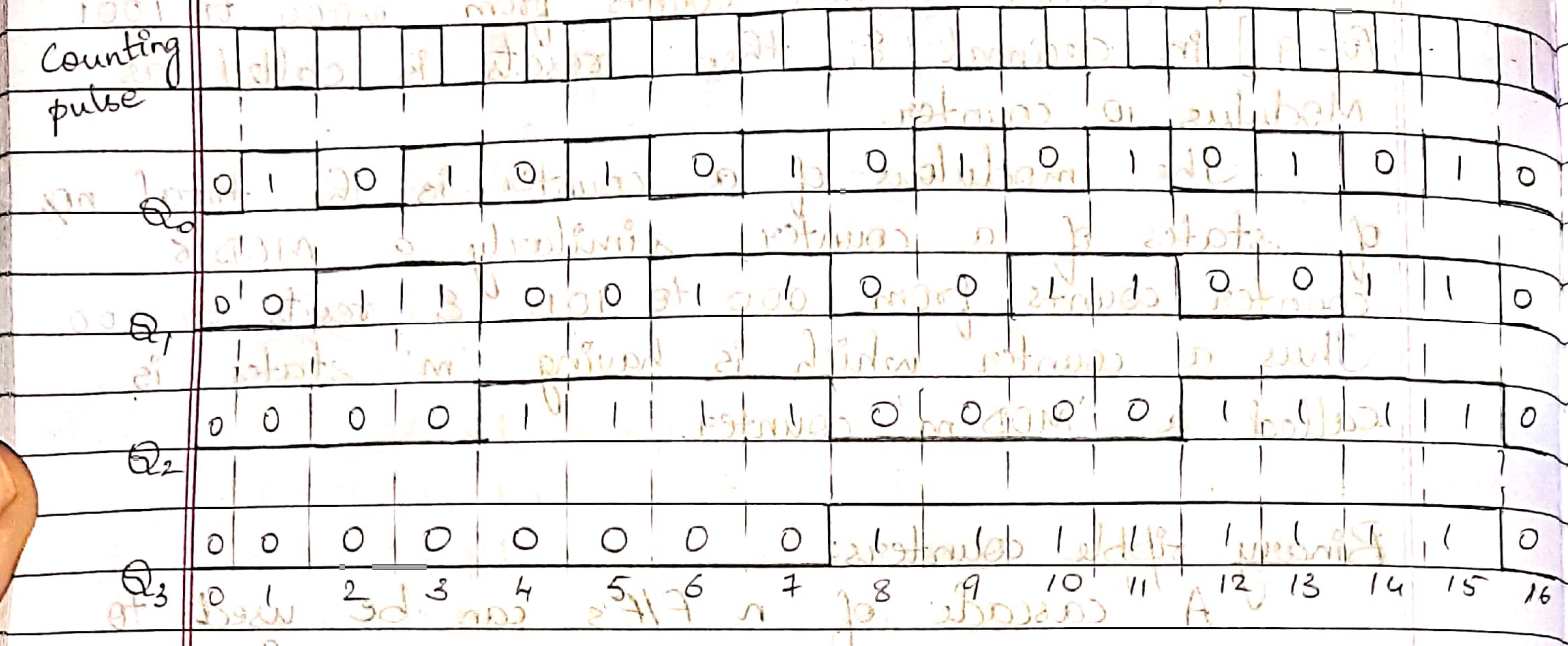
The modulous of a counter is the total no. of states of a counter, similarly a MOD 6 counter counts from 000 to 101 & resets to 000. Thus a counter which is having 'm' states is called a 'MOD m' counter.

Binary ripple counters:

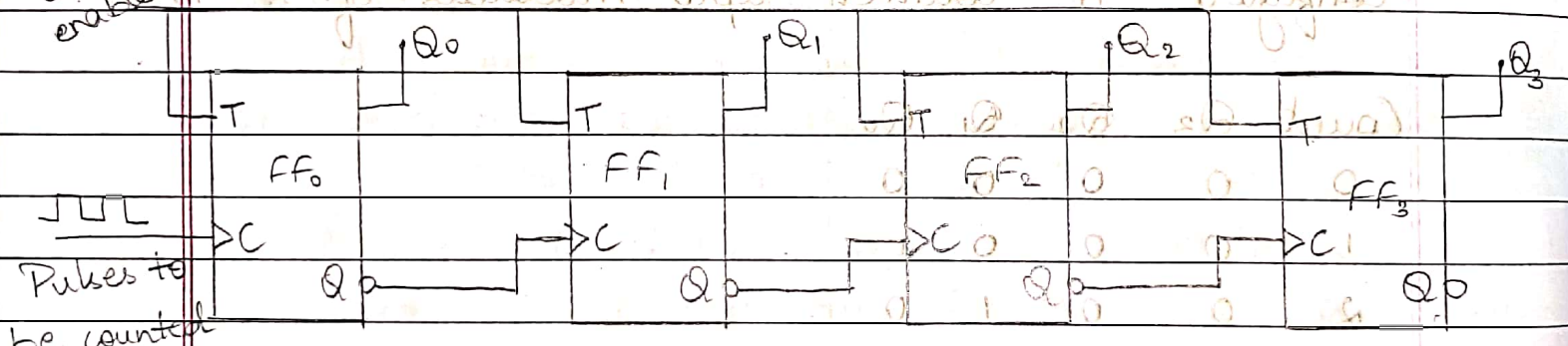
A cascade of n F/F's can be used to configure a counter upto modulus of 2^n.

A cascade of 4 F/F's can be used to configure a counter upto modulus of $2^4 = 16$.

Count	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	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



Positive edge trigger T F/F:



The fig. shows a 4-bit binary up counter configured using +ve edge triggered T F/F's along with its count sequence wrt clock with the count enable are T i/p's held at logic 1 the o/p of each flipflop toggles for every 0 to 1 transition of its clock i/p or at every +ve edge of its clock i/p.

The clock i/p's of FF_1, FF_2 & FF_3 are connected to the $\bar{Q}$ of the previous F/F. These F/F's will change the states on the 1 to 0 transition of the Q o/p's which corresponds to 0 to 1 transition of the $\bar{Q}$ o/p's.

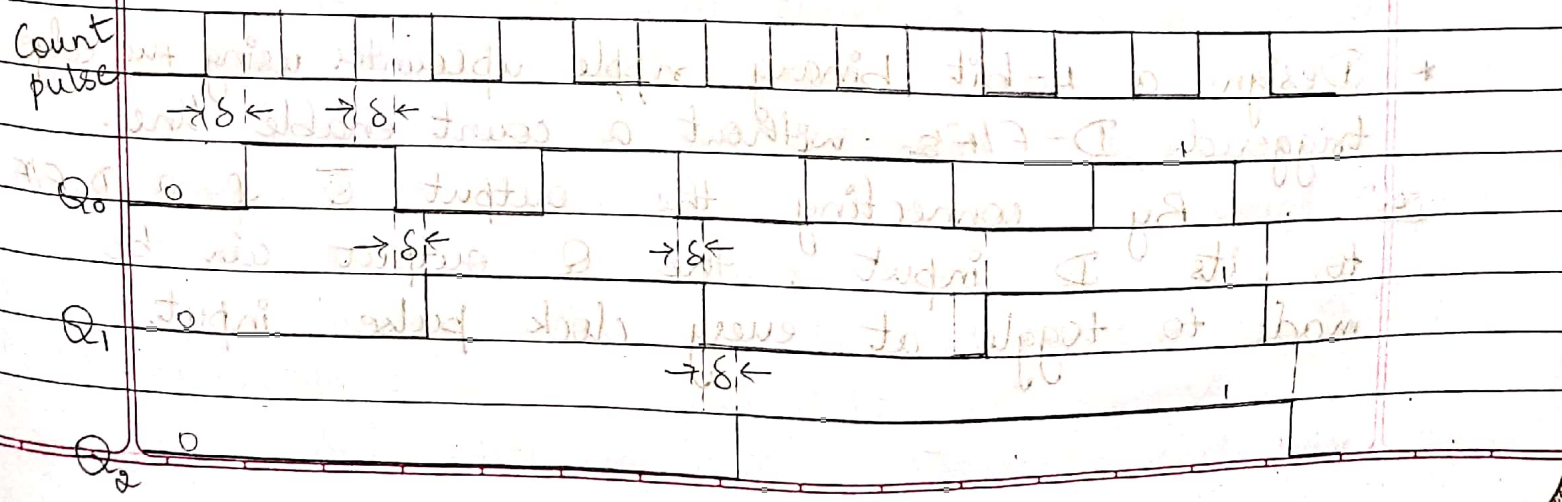
We observe that FF_0 toggles at every +ve edge of its clock & so all other F/F's i.e., FF_1, FF_2 & FF_3 & +ve edge at $\bar{Q}$ corresponds to a -ve edge at Q. These binary counters are also called as ripple counters coz the i^{th} F/F is toggled by a change in the $(i-1)^{th}$ F/F.

The pulses to be counted will ripple through the counter & since the clock pulses are not simultaneously applied to all the F/F these counters are also called as asynchronous counters.

If we take account the propogation delay i.e., tpd b/w the i/p & o/p of a F/F a finite time must lags before a correct count that appears at that o/p of the F/F. This delay will be maximum when all the F/F's need to toggle together as in the 1111 to 0000 transition.

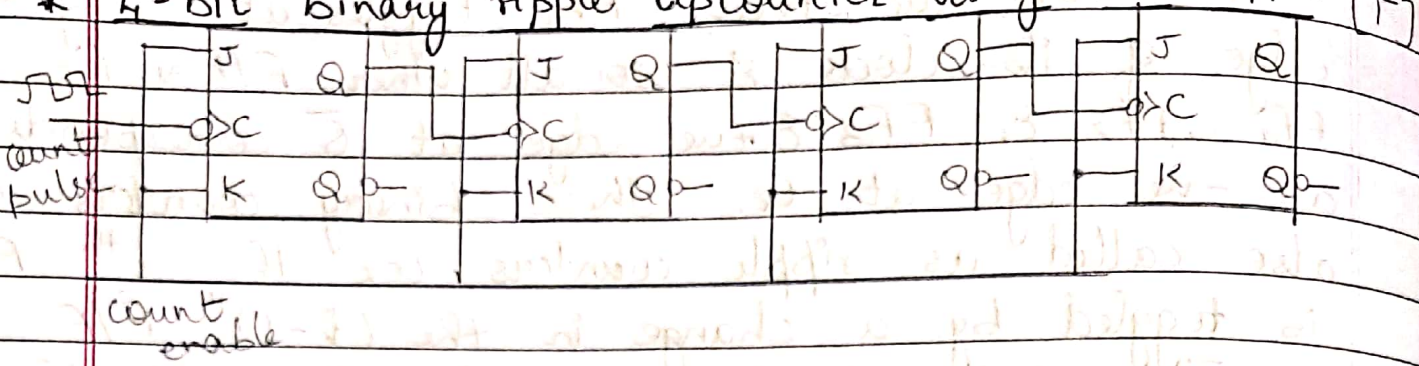
In this case the worst case for the settling time is 4 x tpd. In genaral in an 'n' stage binnary ripple counter is 'n x tpd'.

Draw the timing diagram of a 3 bit binary ripple up-counter configurate using T-flf each with the propogation of δ.



DATE: PAGE:

* 4-bit binary ripple upcounter using JK F/F: (T)



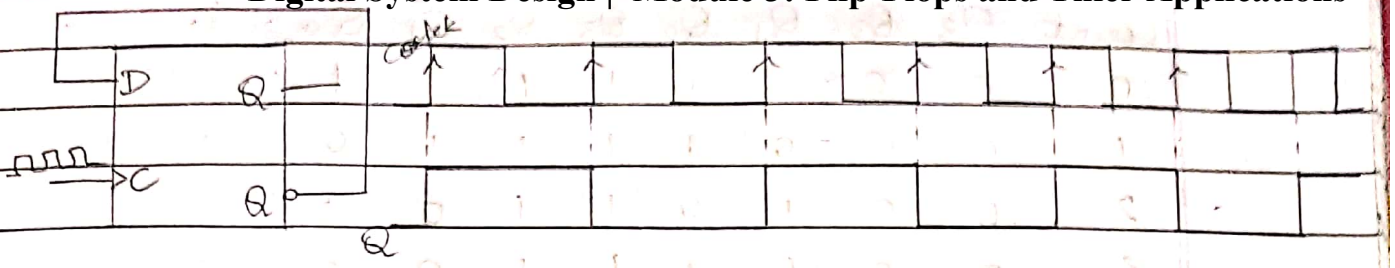
Let us make use of the fact that with J = K = 1, the JK F/F behaves like a toggle F/F which toggles at every negative edge or at every +ve edge of its clock inputs.

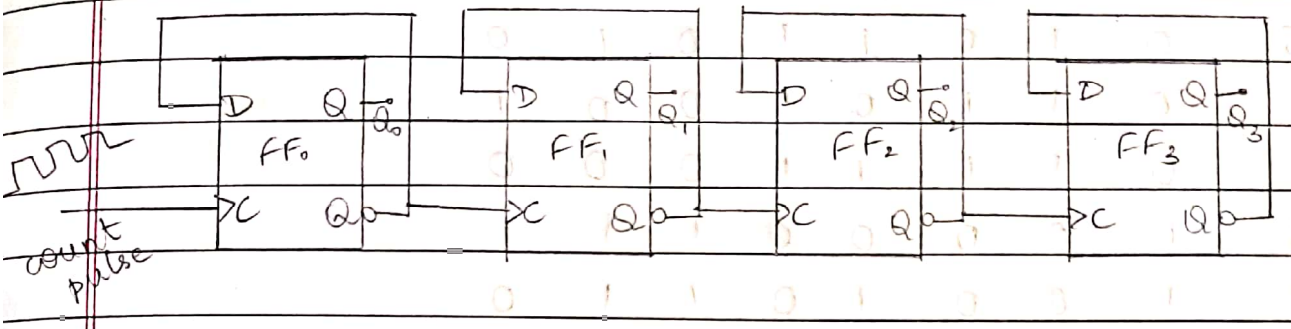
When the count enable line is held at 1. All the J & K inputs are at 1 & each F/F toggles with a -ve transition at its clock input.

Same waveform written as (*)

* Design a 4-bit binary ripple upcounter using +ve edge triggered D-F/Fs. without a count enable line.

Sol: By connecting the output $\bar{Q}$ of a D F/F to its D input, the Q output can be made to toggle at every clock pulse input.





Extra space

Synchronous binary counter:

If the clock pulses to be counted are applied simultaneously to the control i/p of all the F/F's in the cascade connection, such counters are called as Synchronous counters.

The effect of propogation delay 'δ' is not cummulative. Since all F/F's in the counter change the state at the same time.

Suppose if there is delay that is due to the delay of a single f/f.

Now let us look at the 4-bit binary sequence as shown in the fig.

Count	Q_3	Q_2	Q_1	Q_0	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	$\bar{Q}_0$
0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	0
2	0	0	1	0	1	1	0	1
3	0	0	1	1	1	1	0	0
4	0	1	0	0	1	0	1	1
5	0	1	0	1	1	0	1	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	0	1	1	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	0	1
11	1	0	1	1	0	1	0	0
12	1	1	0	0	0	0	1	1
13	1	1	0	1	0	0	1	0
14	1	1	1	0	0	0	0	1
15	1	1	1	1	0	0	0	0

Observe that Q_0 toggles with every clock pulse the other o/p's will toggle whenever all the lower order F/F's are at 1.

Eg: At count 4, Q_2 toggles coz both the lower order F/F's are at high [1] i.e., at count 3. This is true with all counts in the table. Thus the lowest significant F/F must toggle at every clockpulse & the rest must toggle whenever every lower order F/F is at 1. ∴ All the lower order F/F outputs can be ANDed to enable the toggle of a given F/F.

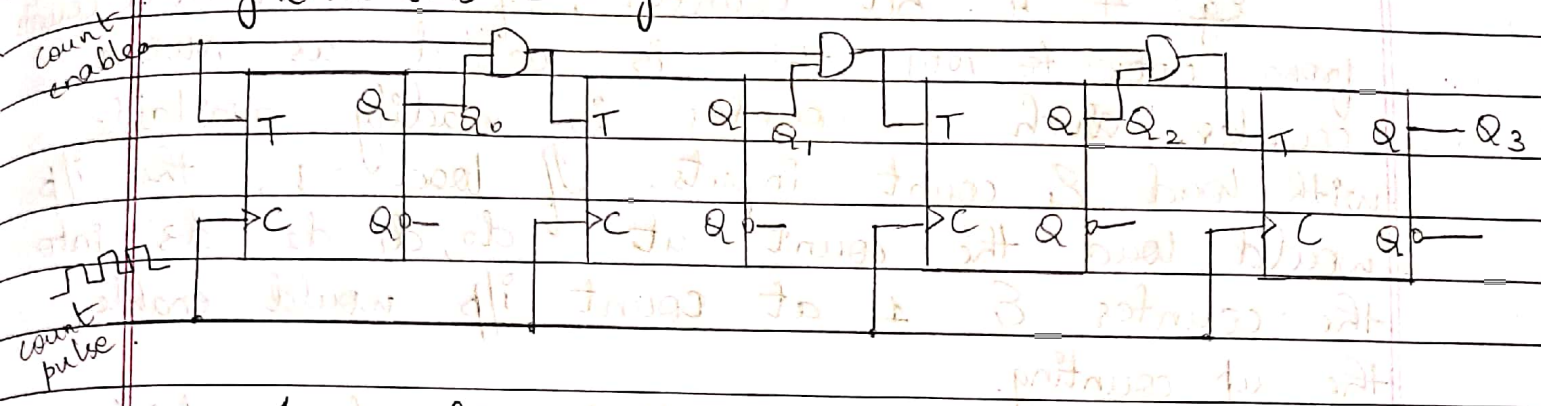
When count enable is at logic 1, the AND gate outputs 1 at T input when all the

DATE: PAGE:

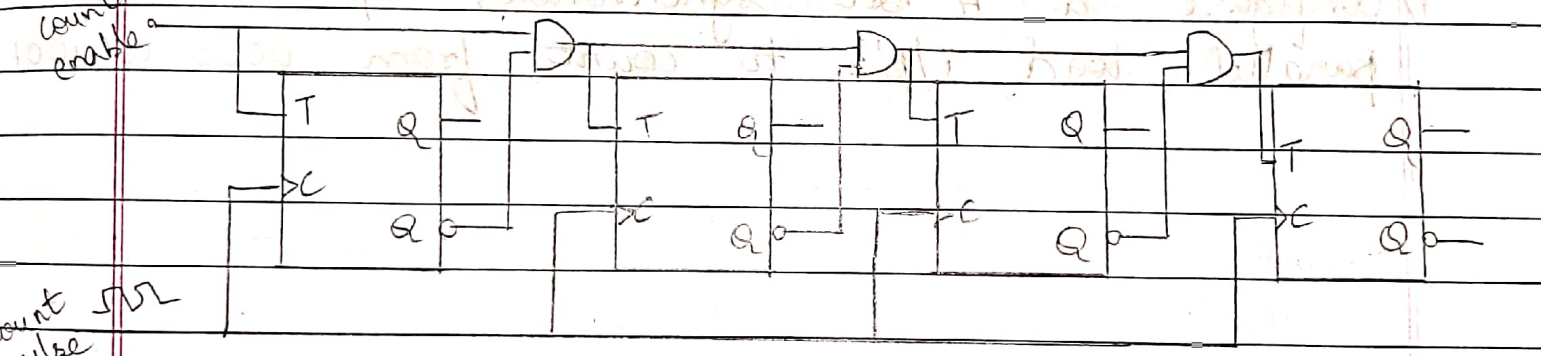
previous FF outputs are at high [1].

The FF whose T input is at logic 1 will toggle at the next clock pulse.

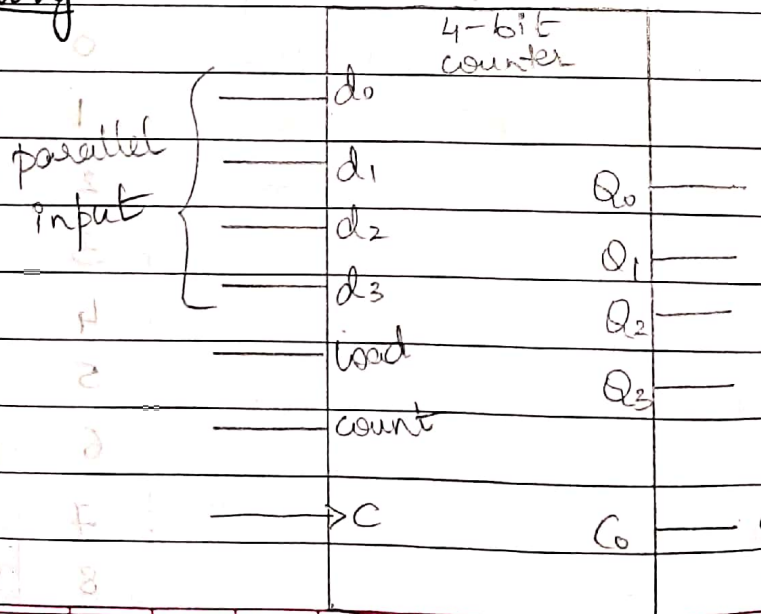
Synchronous binary



In order to configure synchronous binary down counter the input to the AND gate $\bar{Q}$ should be removed & take the output from $\bar{Q}$.



4 bit synchronous counter with parallel times load facility:

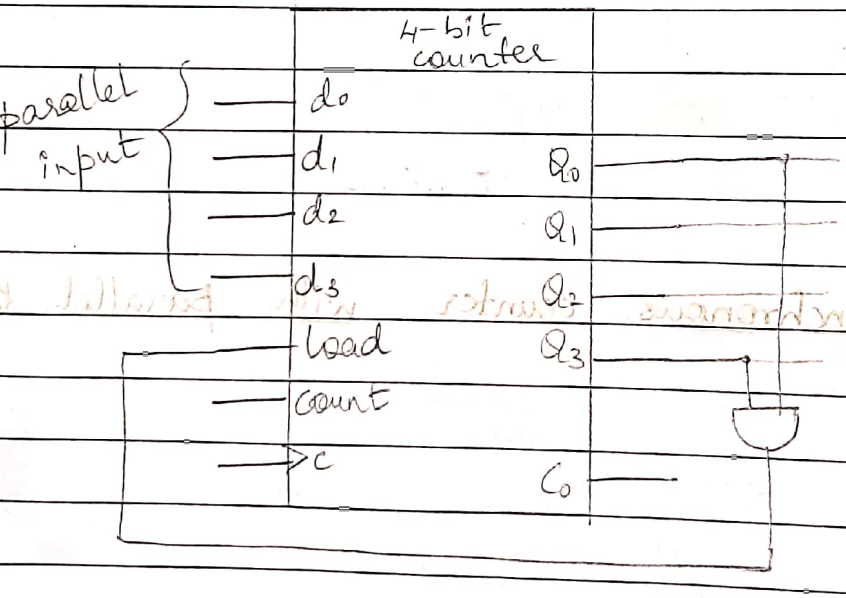


WKT an n-bit counter can be used as a MOD 2^n counter. We can configure a MOD m counter where $m < 2^n$ if we include a facility to parallel load & initial counter.

Eg: A 4 bit counter, if this counter counts from 0000 to 1001 this is called as MOD 10 counter. Such a counter is readily available with load & count inputs. If load = 1, the i/p would load the count at d_0, d_1, d_2, d_3 into the counter & 1 at count i/p would enable the up counting.

We have carry output C_0 also which goes high during the 1111 to 0000 transition.

1) Configure a 4-bit synchronous up counter with parallel load i/p's. to count from 0000 to 1001.

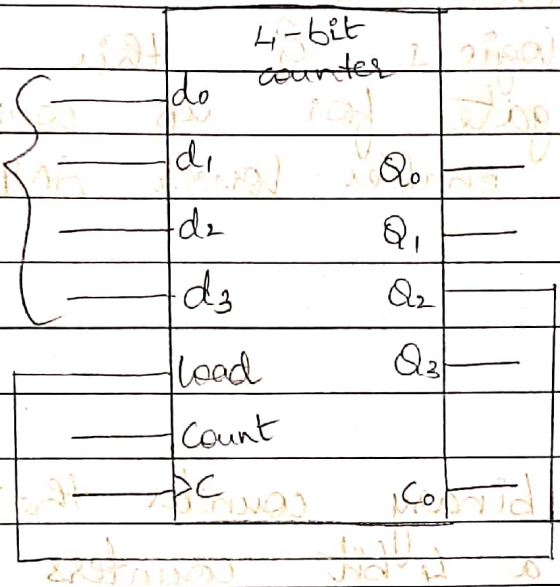


Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	1	0	0	1
0	repeats			

Design

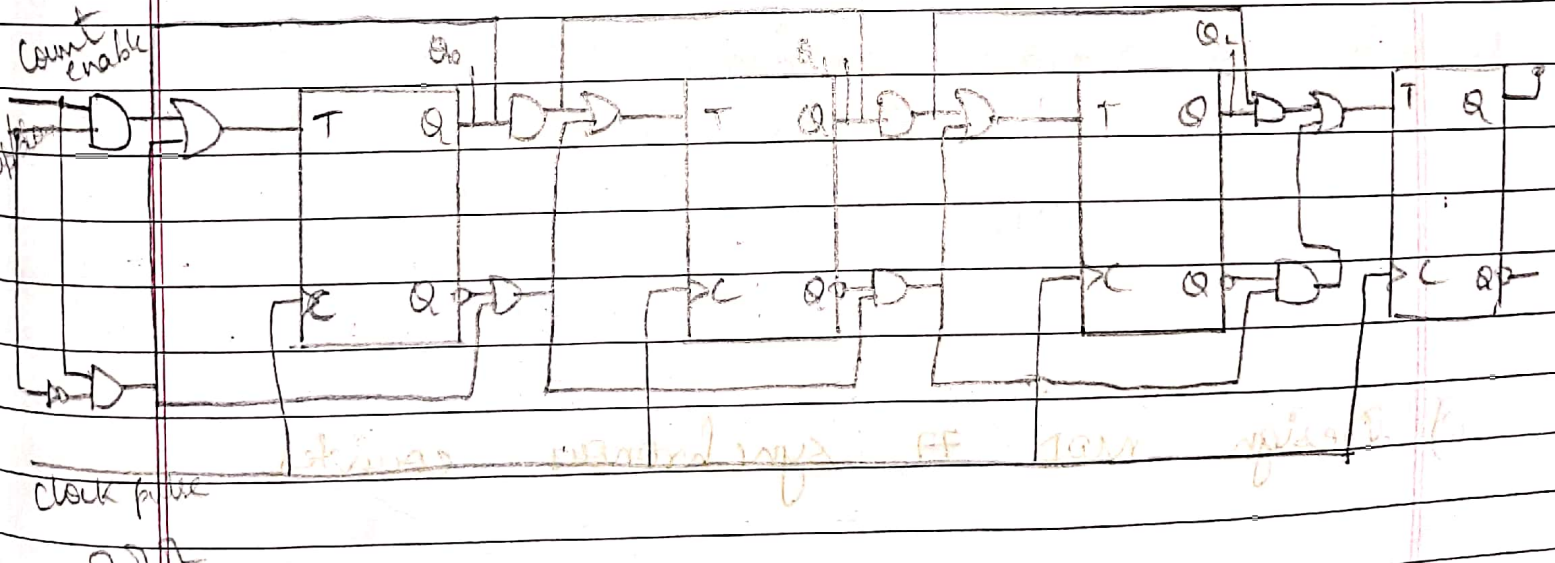
DATE: PAGE:

2) Design a MOD 5 synchronous counter which counts from 0000 to 0100 & repeats using a 4 bit synchronous counter



count	d_0	d_1	d_2	d_3
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
0 repeat	0	0	0	0

3) Design a 4-bit synchronous binary up down counter using T F/F having a count enable i/p.

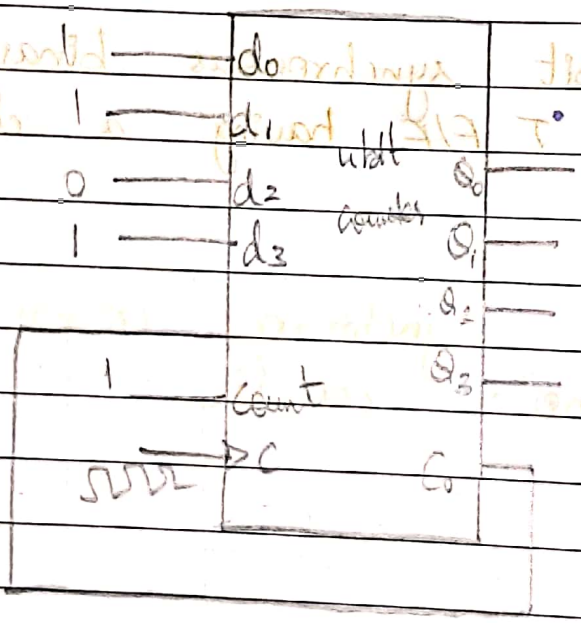


to accept all the previous ANDed o/p as well as previous ANDed $\overline{Q}$, o/p.

* When count enable is at logic 1, the up/down at logic 1 & this enables the upper AND gate for up counting. up/down at logic 0. enable lower AND gates for down count.

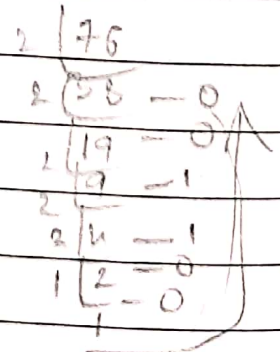
Problems:-

1) Configure a binary counter that counts last 5 states of a 4 bit counters & repeats.

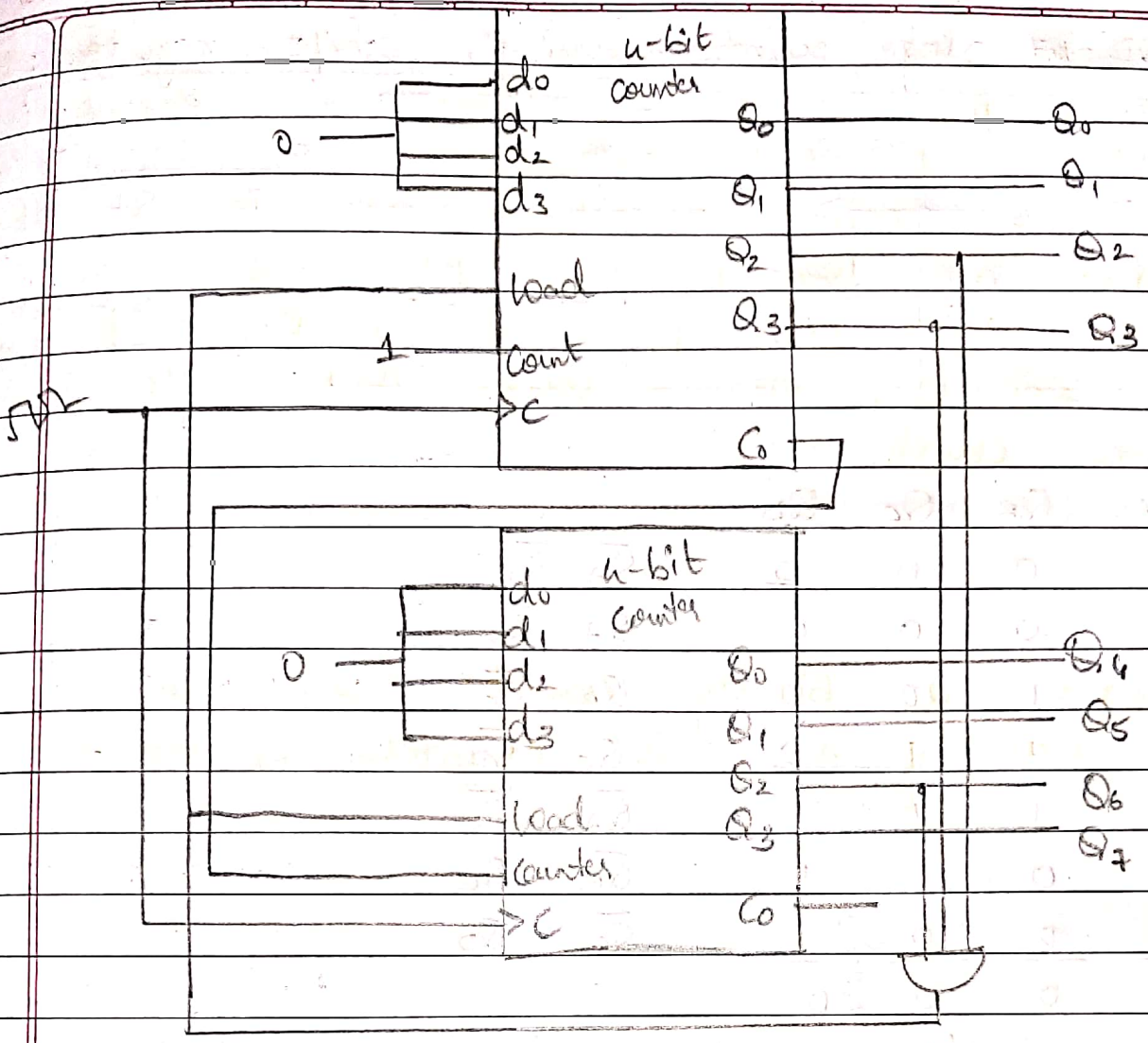


2) Design MOD 77 synchronous counter

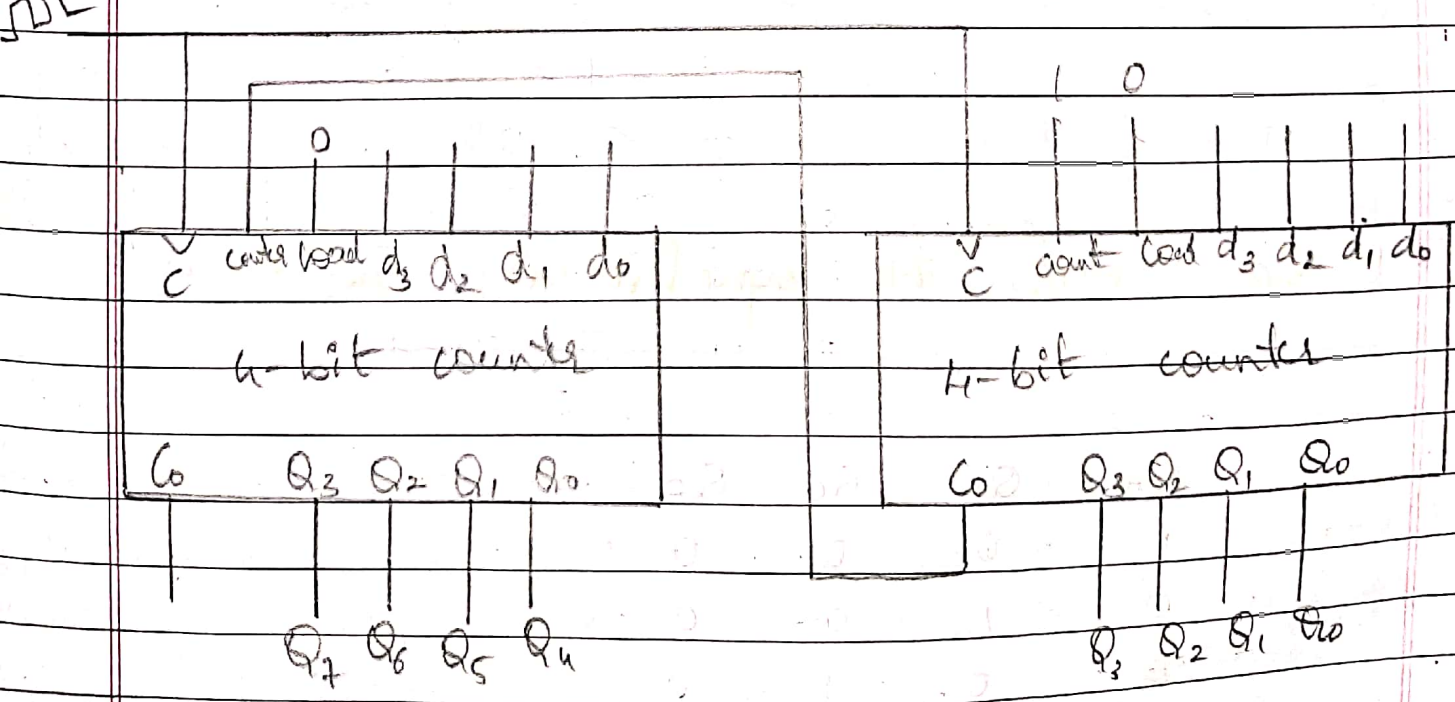
0-76



DATE: PAGE:

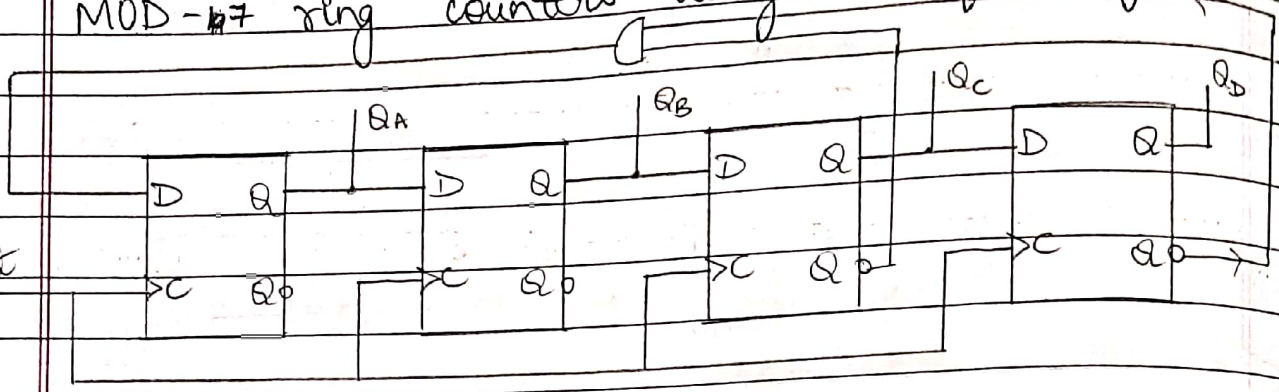


3) Configure 8-bit synchronous counter by using 2, 4-bit synchronous counter.



twisted

MOD-8 ring counters using shift registors:

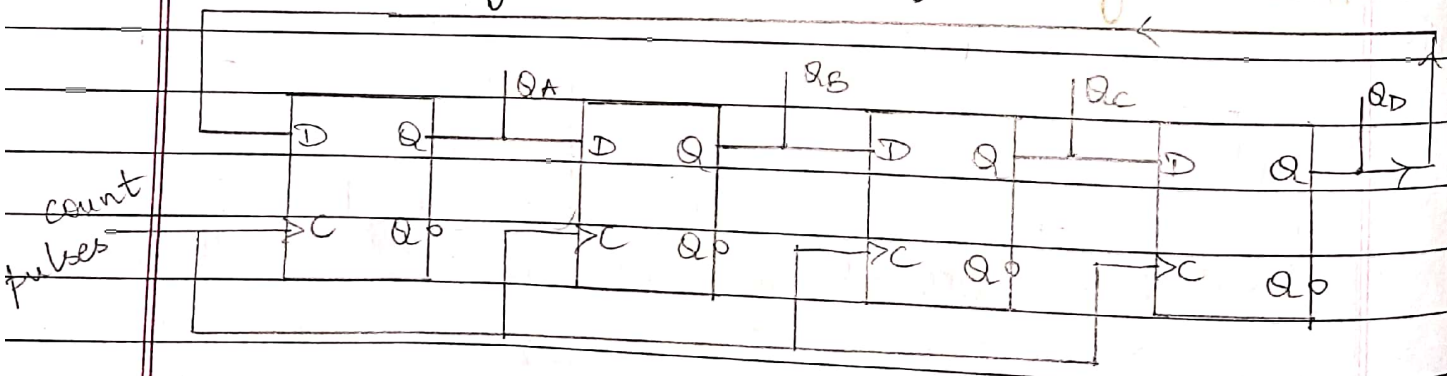


Q_A	Q_B	Q_C	Q_D	
0	0	0	0	$\bar{Q}_A$ $\bar{Q}_D$
1	0	0	0	Q_A $\bar{Q}_B$
1	1	0	0	Q_B $\bar{Q}_C$
1	1	1	0	Q_C $\bar{Q}_D$
0	1	1	1	$\bar{Q}_A$ $\bar{Q}_B$
0	0	1	1	$\bar{Q}_B$ Q_C
0	0	0	1	$\bar{Q}_C$ Q_D
0	0	0	0	

etc.

Theory refer text book

MOD-4 ring counter: using SR

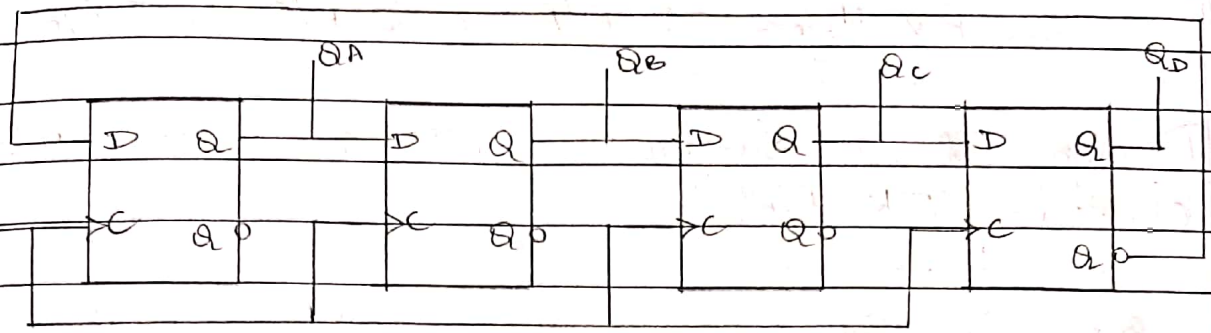


Q_A	Q_B	Q_C	Q_D
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

etc.

DATE: PAGE:

MOD-8 twisted ring counter or also called as switch tale counter or also called Johnson counter



Q_A	Q_B	Q_C	Q_D	AND-gate inputs
0	0	0	0	$\bar{Q}_A$ $\bar{Q}_D$
1	0	0	0	Q_A $\bar{Q}_B$
1	1	0	0	Q_B $\bar{Q}_C$
1	1	1	0	Q_C $\bar{Q}_D$
1	1	1	1	Q_A Q_D
0	1	1	1	$\bar{Q}_A$ Q_B
0	0	1	1	$\bar{Q}_B$ Q_C
0	0	0	1	$\bar{Q}_C$ Q_D
0	0	0	0	

etc

A variation of the ring counter is the switch tale counter also known as Johnson counter. Assuming the counter starts at Q_A, Q_B Q_C Q_D = 0000 state. In this counter the compliment of the right most F/F serves as the i/p to the left most F/F & sequence starts.