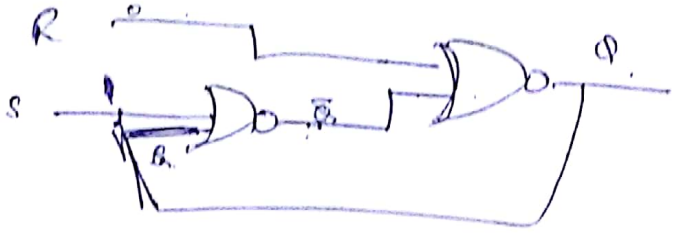
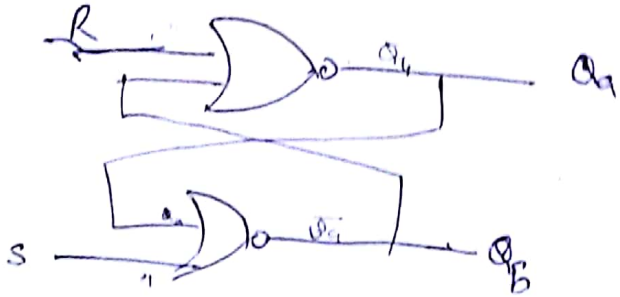
Gated SR Simple Latch



R	S	q
0	0	no change
0	1	1
1	0	0
1	1	



R	S	Q_a	Q_b
0	0	no change	
0	1	1	0
1	0	0	1
1	1	0	0

$Q_a = Q_b = 0$

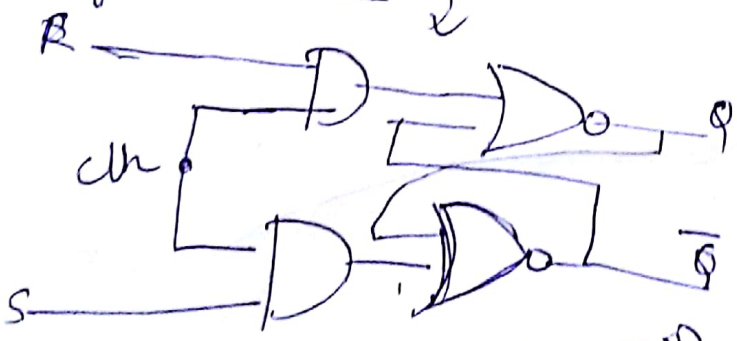
At a time $S = 1$ $R = 0$ $Q_a = 1$ $Q_b = 0$.

Now if $S = 1$ $R = 1$ Q_a 0 $Q_b = 0$.

Now if $R = 0$ $S = 0$ $Q_a =$ 0 / $Q_b = 0$. //
$Q_a = 0$

If delay b/w the NOR gates is the same the o/p will oscillate indef.

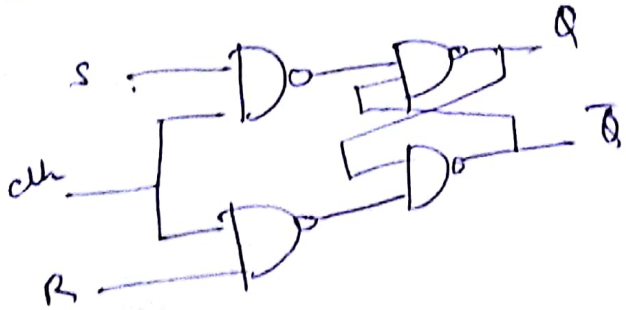
Gated Latches



clk	S	R	Q_{t+1}
0	X	X	Q(t)
1	0	0	Q(t)
1	0	1	0
1	1	0	1
1	1	1	X

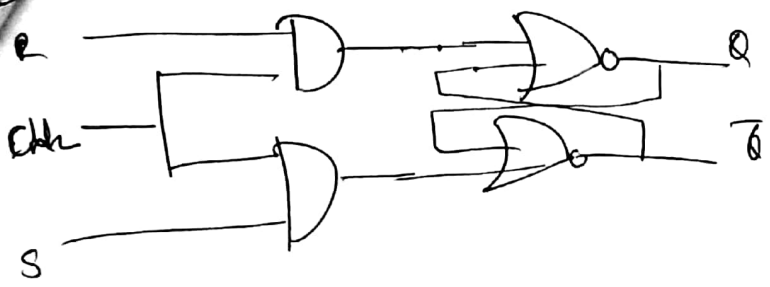
Gated SR with NAND

NOR Q =



ated SR latch

→ SR latch with an Enable signal.



Clk	S	R	Q(t+1)
0	x	x	Q(t) no change
1	0	0	Q(t) no.
1	0	1	0
1	1	0	1
1	1	1	X

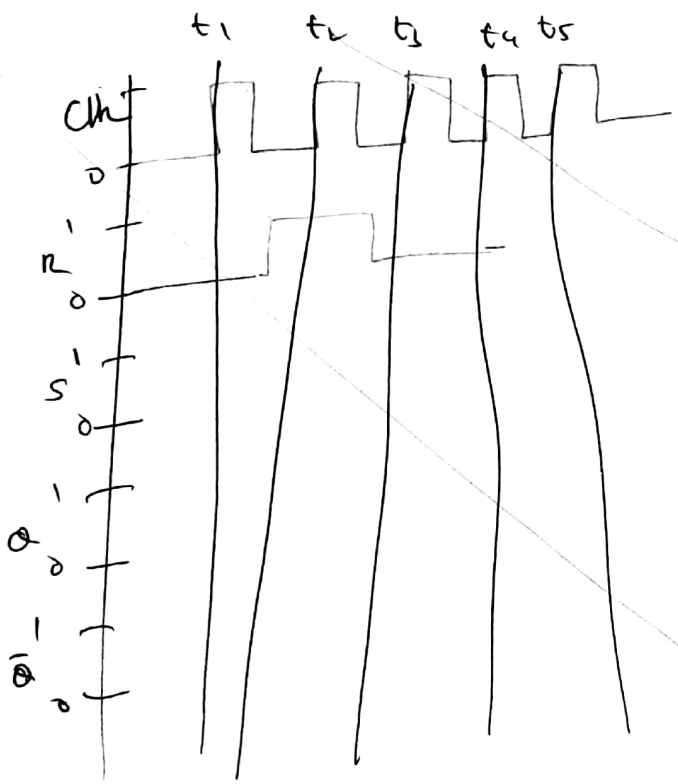
Clk :- allows changes in the states of mem elements to occur at well defined intervals

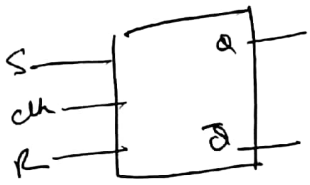
Ckt which have control signal are called gated latches.

Last state where S = 1 R = 1 & the o/p is undefined

When clk goes from 1 to 0 both ~~S=R~~ ~~race~~ o/p oscillates.

→ For proper operation of SR latch ~~i/p = 1 for both~~ S = R = 1 must be avoided when clock changes from 1 to 0.





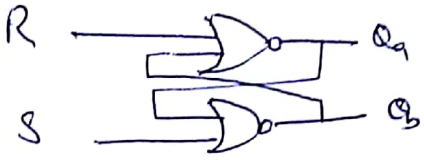
SR Latch

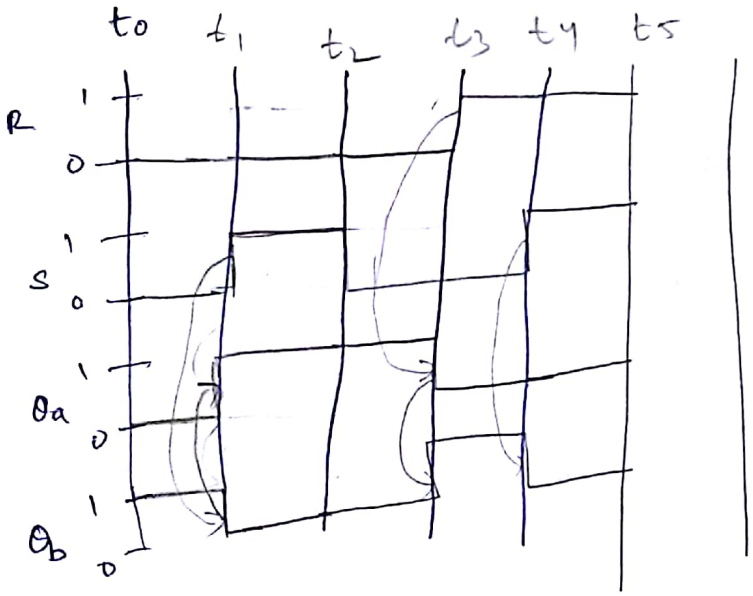
let $Q_a = 0$ $Q_b = 1$

R	S	Q_a	Q_b
0	0	0	1
0	1	1	0
0	0	1	0
1	0	0	1
1	1	0	0

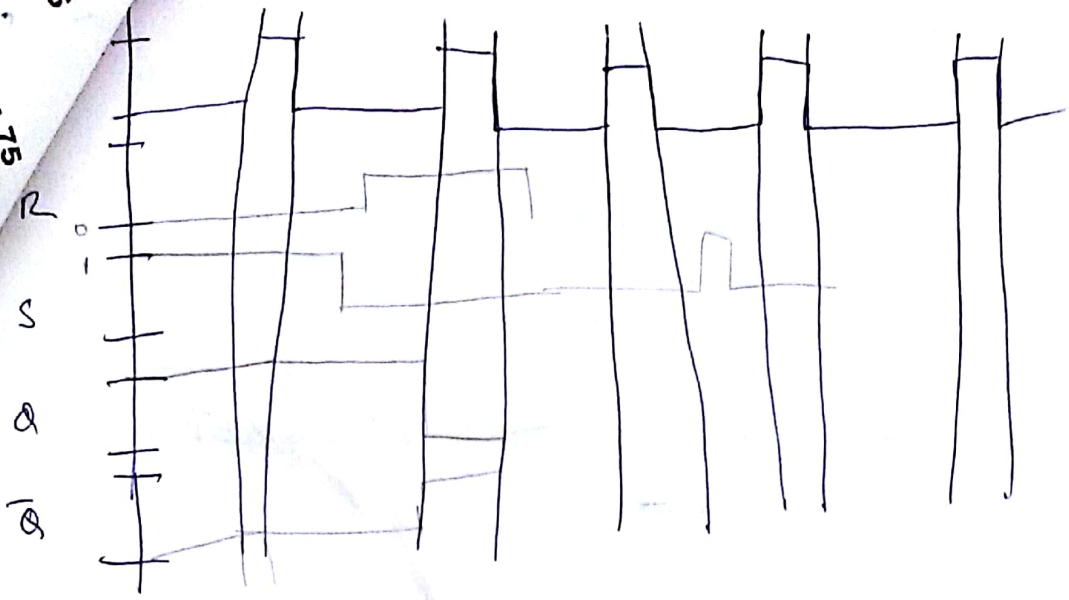
$$Q_a = \overline{R + Q_b}$$

$$Q_b = \overline{S + Q_a}$$

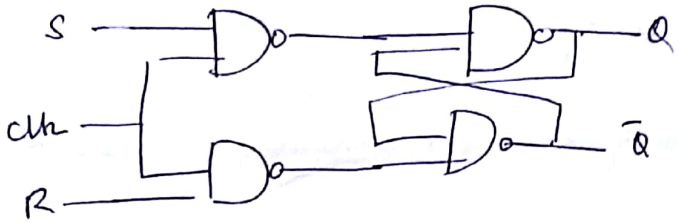




SR latch contd.

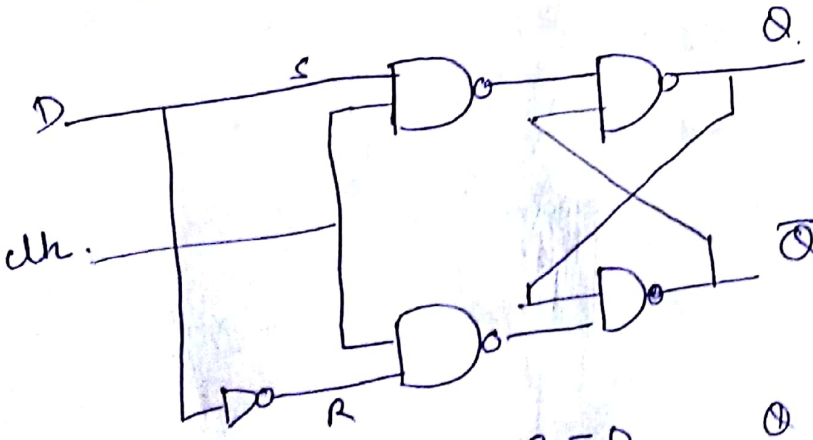


Gated SR Latch with NAND gates

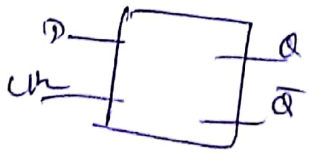


Gated D latch

→ Single data input called D
- Stores the i/p value, under the control of a clk signal.



clk	D	Q(t+1)
0	x	Q(t)
1	0	0
1	1	1



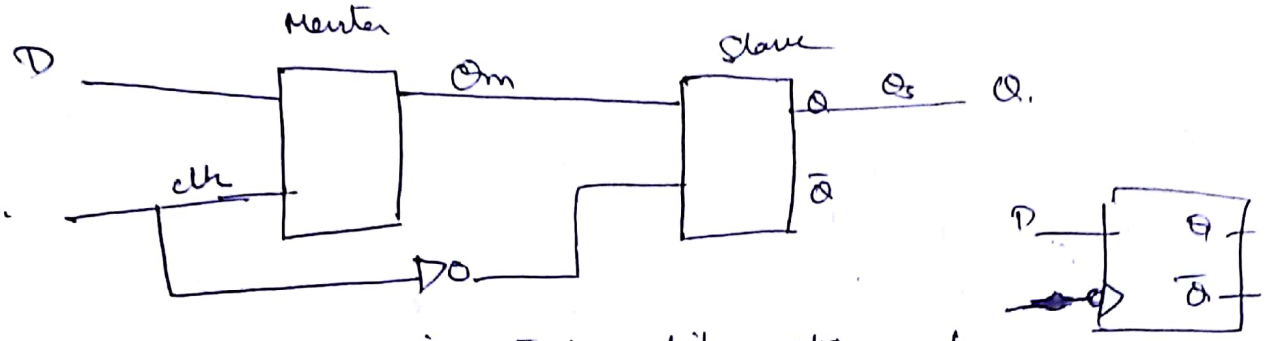
D = 1 S = 1 & R = 0 Q = 1 } clk = 1.
D = 0 S = 0 & R = 1 Q = 0

→ O/P of the gated D latch is controlled by the level of the clk i/p, the latch is said to be level sensitive.

Master Slave & Edge Tuggered D Flip-Flop

→ Level sensitive latches, the state of the latch keeps changing according to the values of i/p signals during the period when the clk signal is active (equal to 1)

Master Slave D F/F



Master - chages its state while clk = 1
Slave " " " " = 0

→ when clk is high the master tracks the value of D i/p & slave does not change

∴ Qm follows all chges in D
Qs is const.

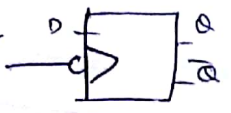
when clk = 0 Qm is const ∴ Qs will chg at most 1 time when clk chge from 1 to 0.

→ ∴ o/p will change at the -ve going edge of the clk.

→ Negative edge :- edge where clk signal changes from 1 to 0

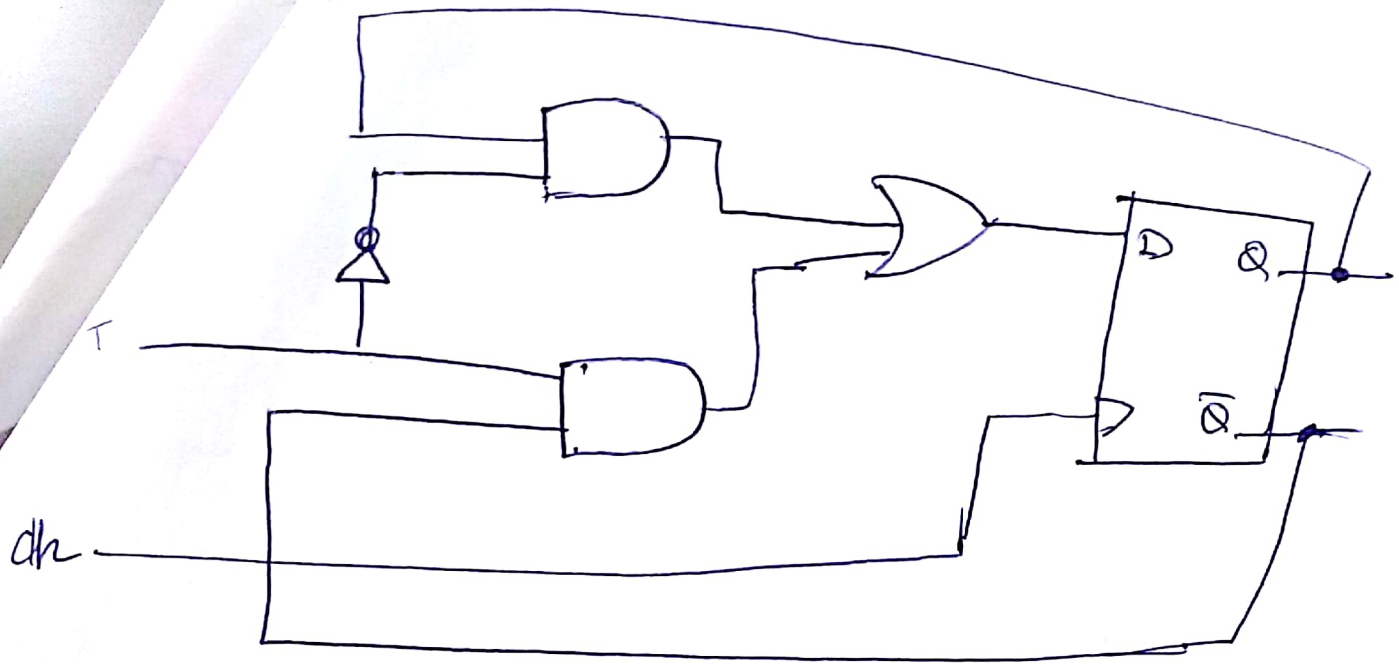
→ overall o/p is only one at the -ve edge of clk.

→ Flip flop :- denotes a storage element that changes its o/p state at the edge of a controlling clk signal.

⇒ The clk 

where > denotes that F/F response to 'active edge' of the clk.
—O denotes active edge is -ve edge

T Flip Flop



T	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

→ +ve edge triggered D FF.

→ D = Q or $\overline{Q}$ with ~~clock~~ control from T.

→ If T = 0 ⇒ D = Q ⇒ no change of state Q(t+1) = Q(t)

" T = 1 → D = $\overline{Q}$ ⇒ new state Q(t+1) = $\overline{Q(t)}$

→ used for toggle counter ckts.

$Q_b = 0$.

$\bar{Q}_b = 0$ //

$Q_b = 0$ // invalid

JK Flip Flop

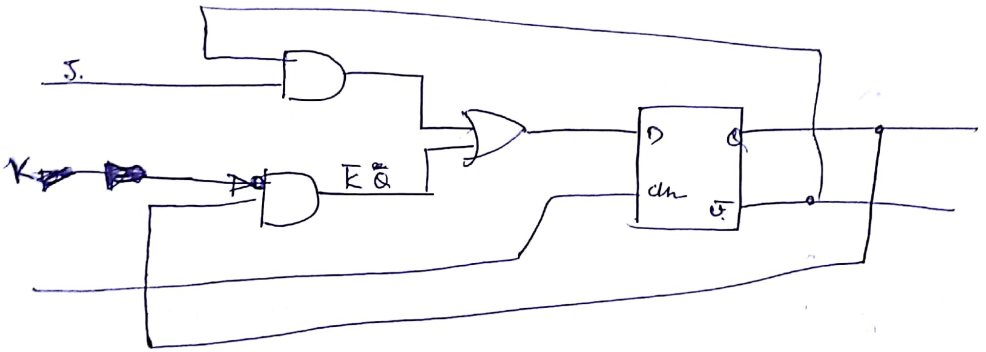
$$D = J\bar{Q} + \bar{K}Q.$$

uses SR & T F/F.

it behaves as SR F/F when $J = S$ & $K = R$ for all i/p. values except $J = K = 1$

In this case

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\bar{Q}$(t)



JK F/F toggles its state like T F/F when $J = K = 1$.

→ storage purpose also serves as T F/F. by connecting J & K together.

R	S	Q_a	Q_b
0	0	0	1
0	1	1	0
0	0	1	0
1	0	0	1
1	1	0	0
1	0	0	1
1	1	0	0

$$Q_a = \overline{R + Q_b}$$

$$Q_b = \overline{S + Q_a}$$

At a time only one input bit must change.

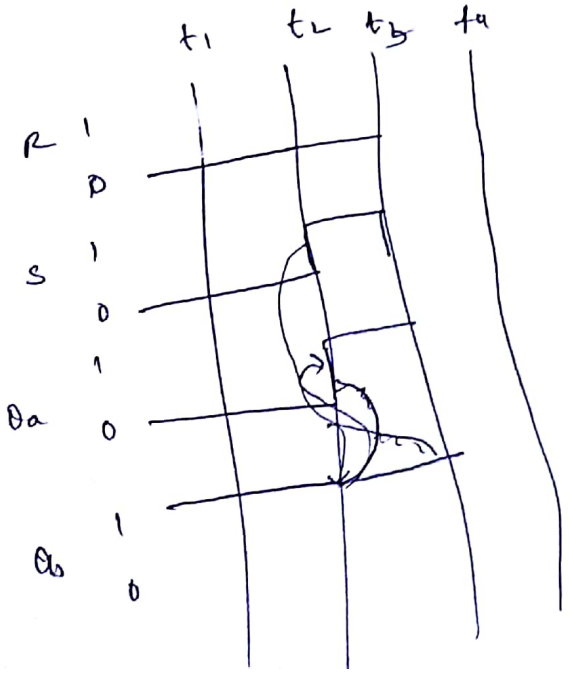
0	0	1	1
0	0	0	0
0	0	1	1

Ideally oscillate, if delay in both the NOR gates is exactly the same then it oscillate. Practically, there will be some diff so it will settle on some value.

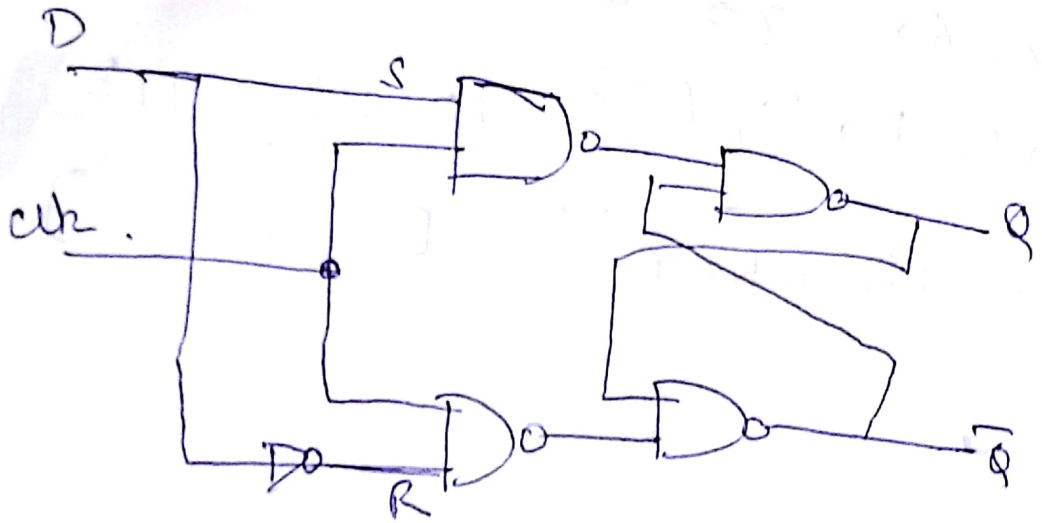
R	S	Q_a	Q_b
1	0	0	1
			0
0	1		

Q. previous step

R	S	Q_a	Q_b
1	1	0	0
1	1		0



single data i/p called D & it stores the value on its o/p under the control of a clock signal.



Clk	D	Q(t+1)
0	x	Q(t)
1	0	0
1	1	1

Latch → storage element.

At:

Now

Now

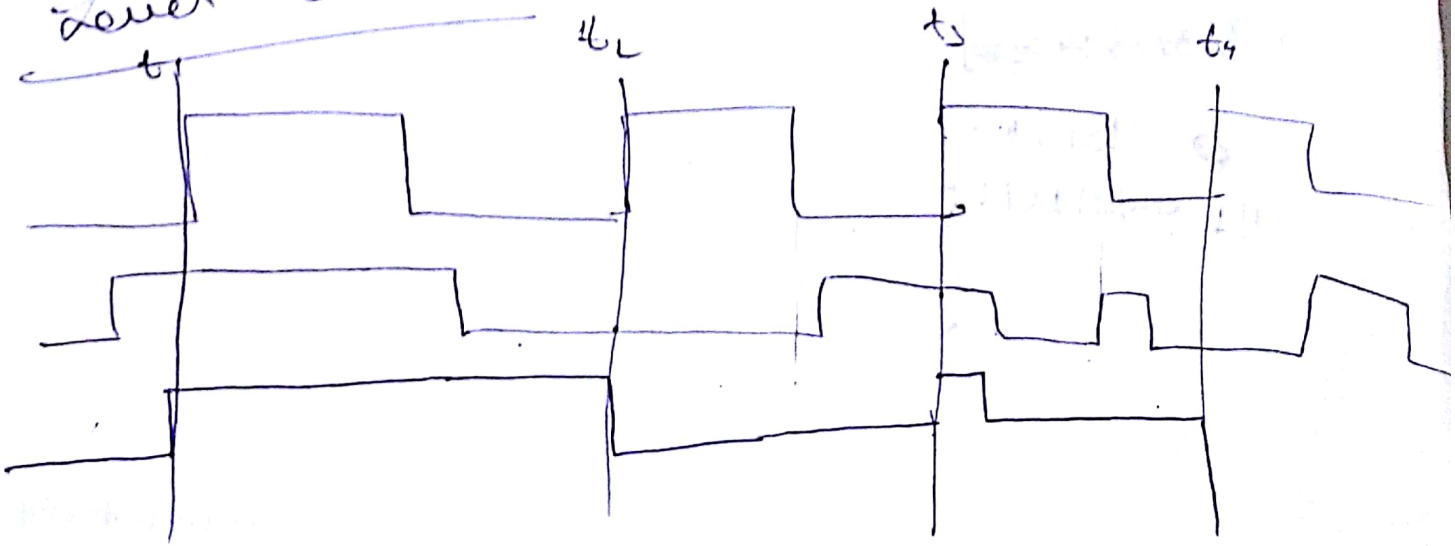
If clk oscillate

gated

R

clk

Level Sensitive



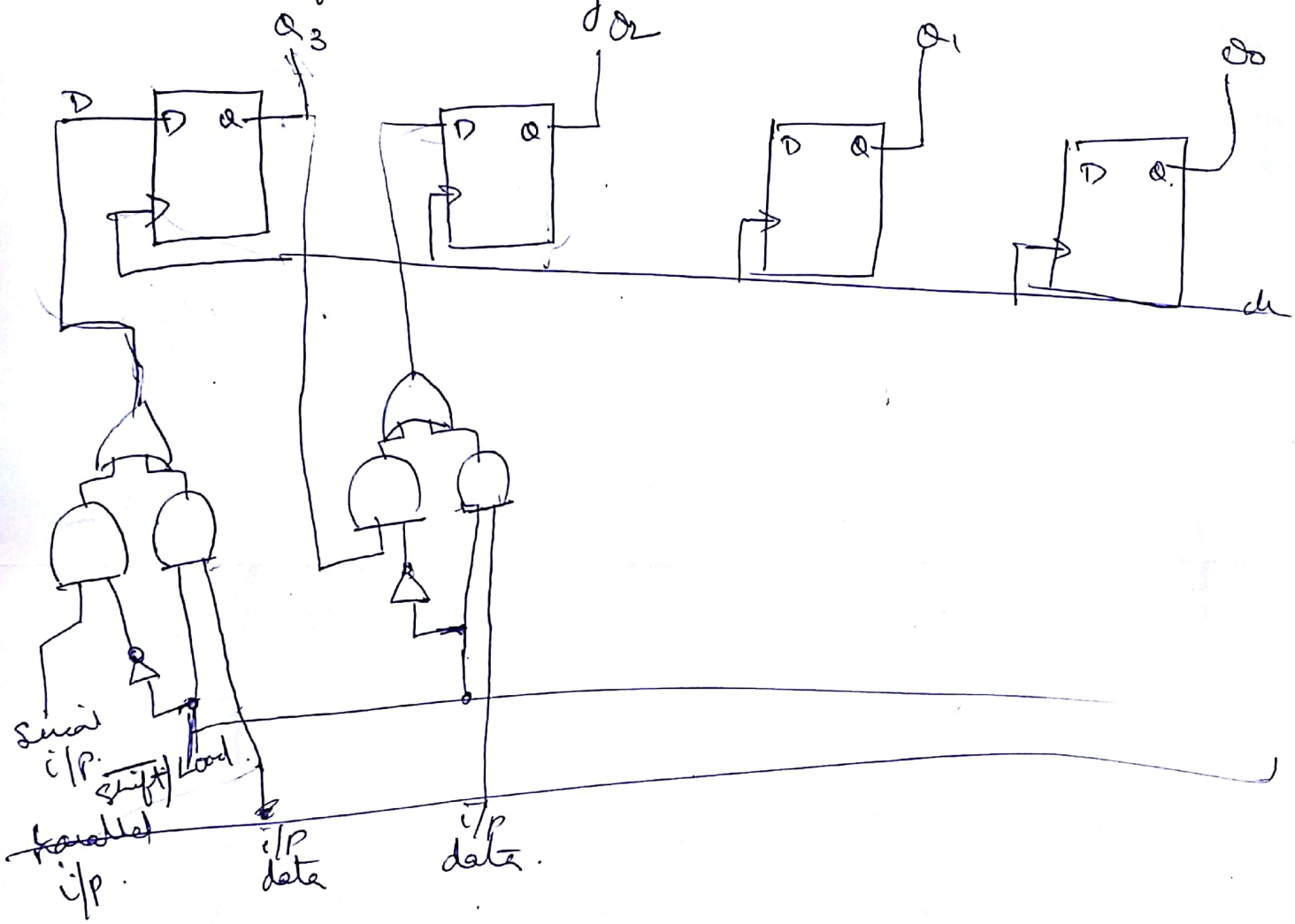
	$Q(t)$
x	

Parallel Access Shift reg

→ Parallel data Xfer

→ Serial " "

Xfer n-bit data serially, use a shift reg. where all n bits are loaded parallely and content xferred serially

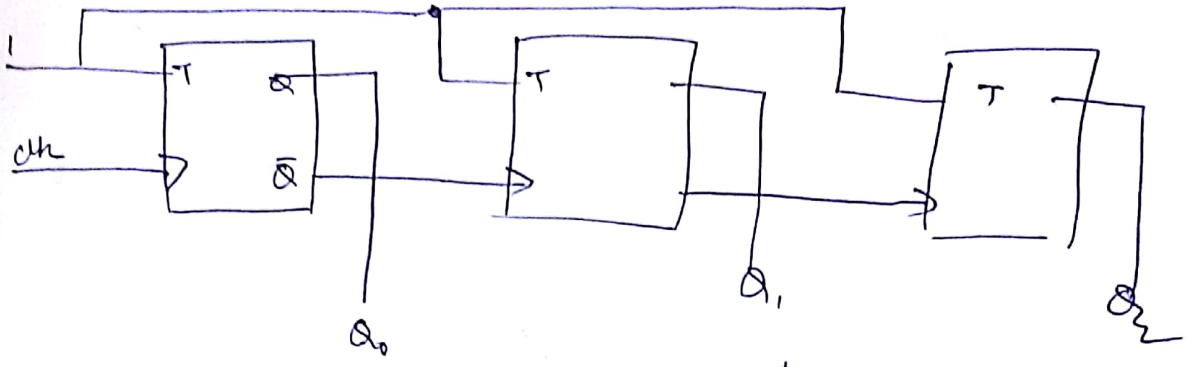


This ckt can do both serial to parallel & // to serial.

Counters

Ripple Counter / Asynchronous Counter

upcounter with T F/F

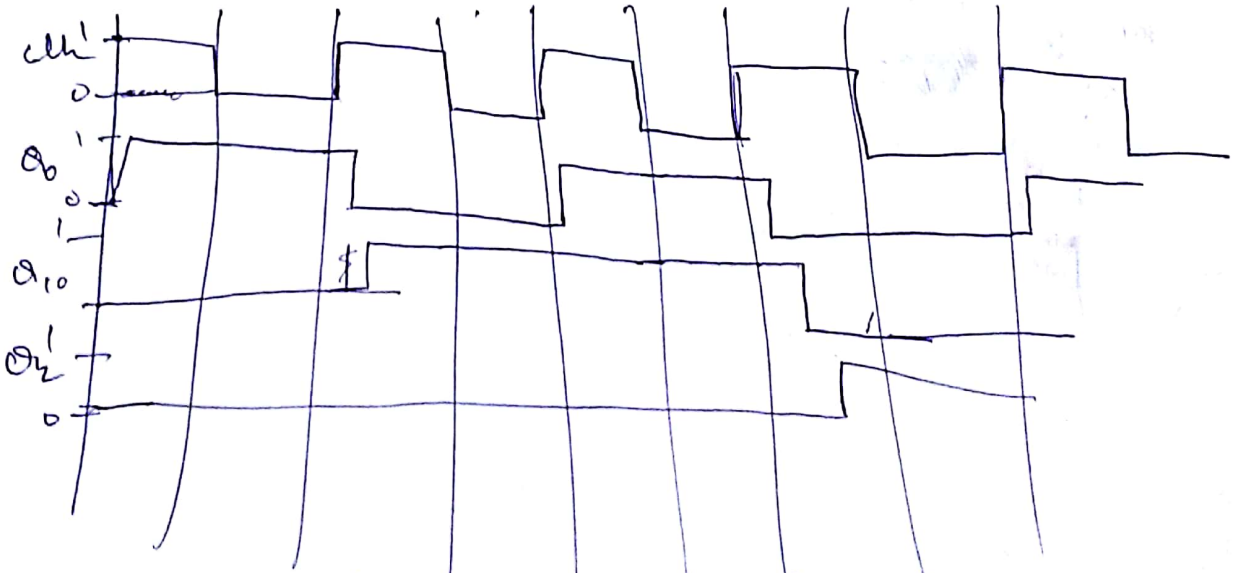


counts from 0 to 7 upcounter

→ i/p is connected to 1 ⇒ F/F will toggle at each the clock pulse.

→ Clk to count the no. of pulses at the clk.

→ 2nd & 3rd F/F toggle their state after the preceding F/F changes its state from $Q = 1$ to $Q = 0$. (+ve edge at $\bar{Q}$)



→ Only the 1st stage is synchronized with the clock. The other two stages respond after an additional delay.

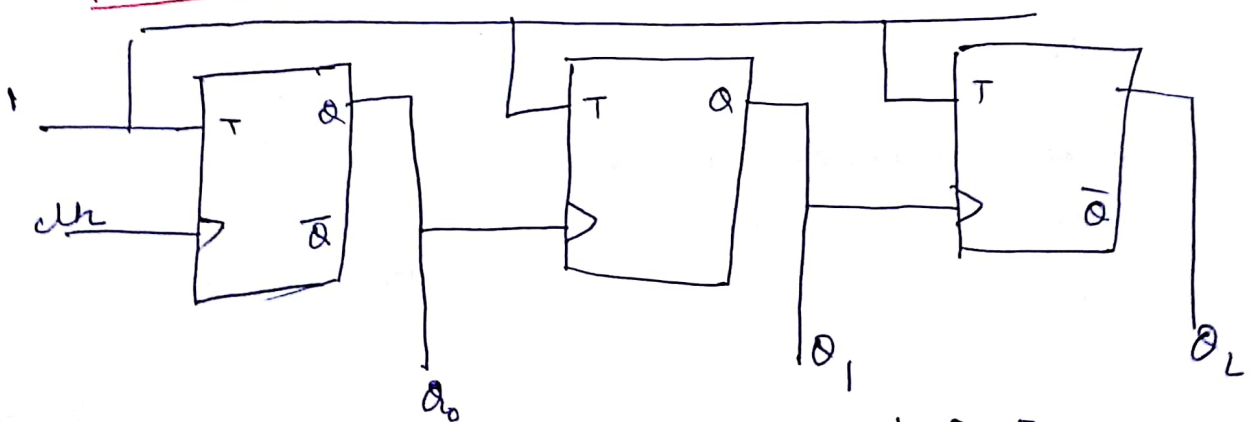
Asyn. counter contd

→ The change in Q_0 is observed only after a propagation delay from the the edge of clock. Q_1 & Q_2 FlF have yet not changed.

→ change in Q_1 → appears after propagation delay → (∵ at t_3 O/p is 000)

→ F.F O/p of Q_2 change after a third delay at which point the stable state of the ckt is reached & the count is 100.

Down counter with T FlF

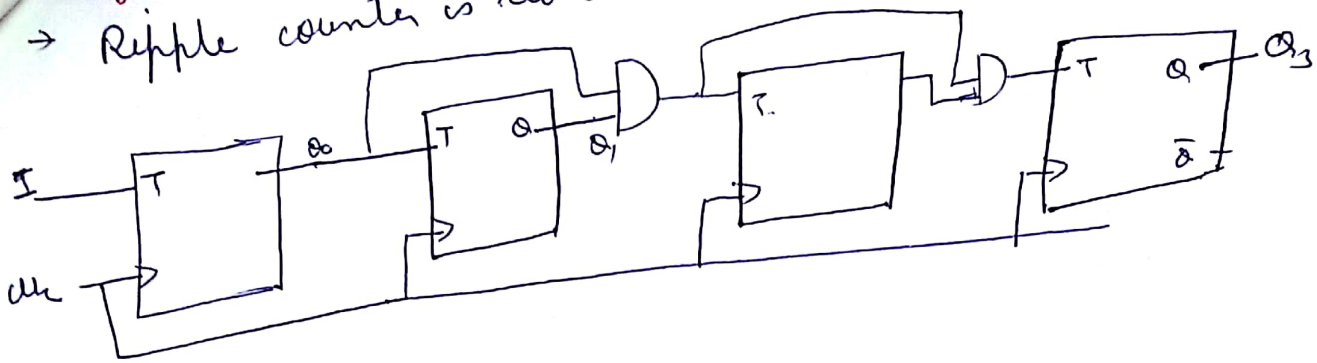


∴ counts from 0, 7, 6, 5, 4 . . . 1, 0, 7 . .

→ combination of up/down counter.

Synchronous Counter

→ Ripple counter is slow ⇒ more no of bits even slower.



→ Q_1 change when $Q_0 = 1$

→ Q_2 " " both $Q_1 = 1$ & $Q_0 = 1$

⇒ : n bit up counter, a given flf changes its state only when all the preceding flf are in the state Q = 1

→ Instead of using AND gates of increased size for each stage, factored arrangement as shown is used.

→ All flf change state after propagation delay from the true edge of the clk.

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