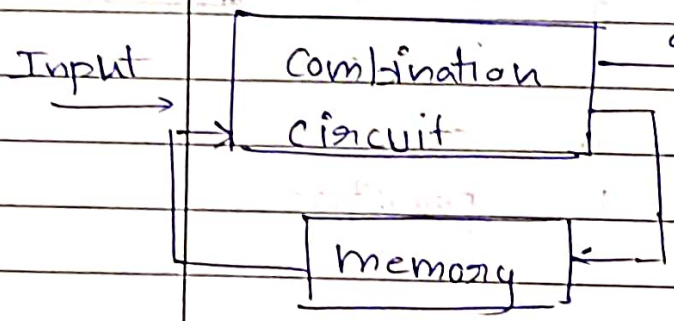
__/__/____

Sequential Circuit: → output depends on the past output



two types:

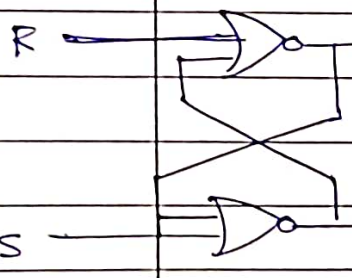
→ Synchronous sequential circuit
Responds to the inputs only at discrete time intervals.

→ asynchronous sequential circuit
Immediatly responds to the input level changes

memory element

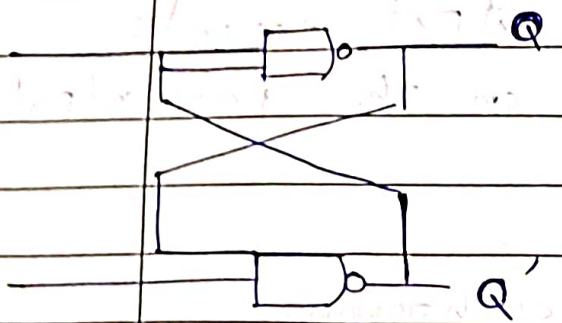
↳ latch → level triggered
↳ flip-flop → edge triggered.

SR latch (Asynchrous sc)



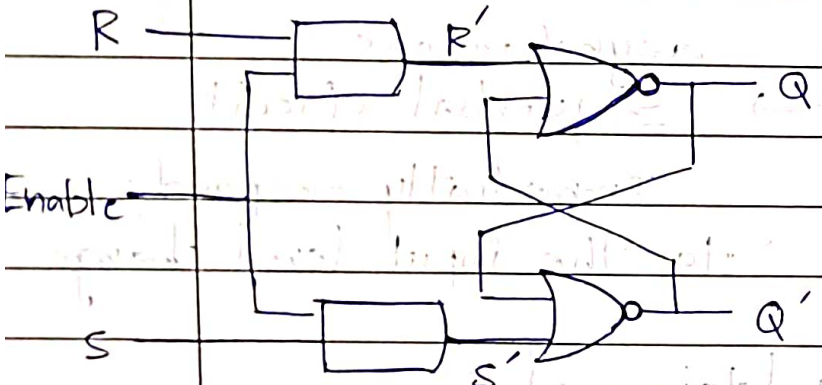
R	S	Q	Q'	
1	0	0	1	(Reset)
0	1	1	0	(set)
0	0	← PrevState →		(Retain)
1	1	Forbidden		

SR latch Using Nand gates:

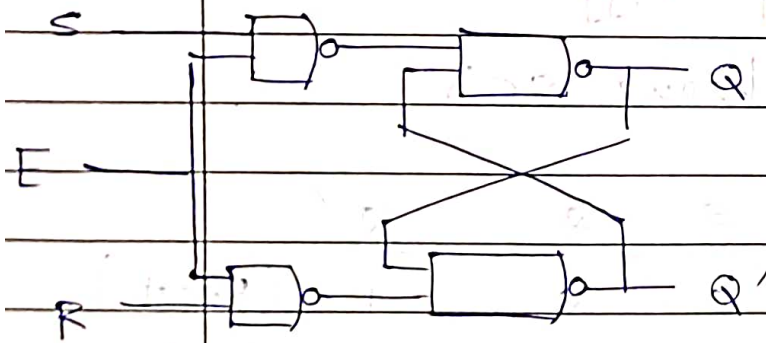


S	R	Q	Q'
0	0	1	1
0	1	1	0
1	0	0	1
1	1	prevstate	

Gated SR latch:



E	S	R	Q	Q'
0	x	x	prevstate	
1	0	0	prevstate	
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0



Gated SR latch using Nand gates.

__/__/____

Characteristic tables:

D	Q(t+1)
0	0
1	1

Data-flipflop

$Q(t+1) = D$

T	Q(t+1)
0	Q(t)
1	Q'(t)

T-flipflop

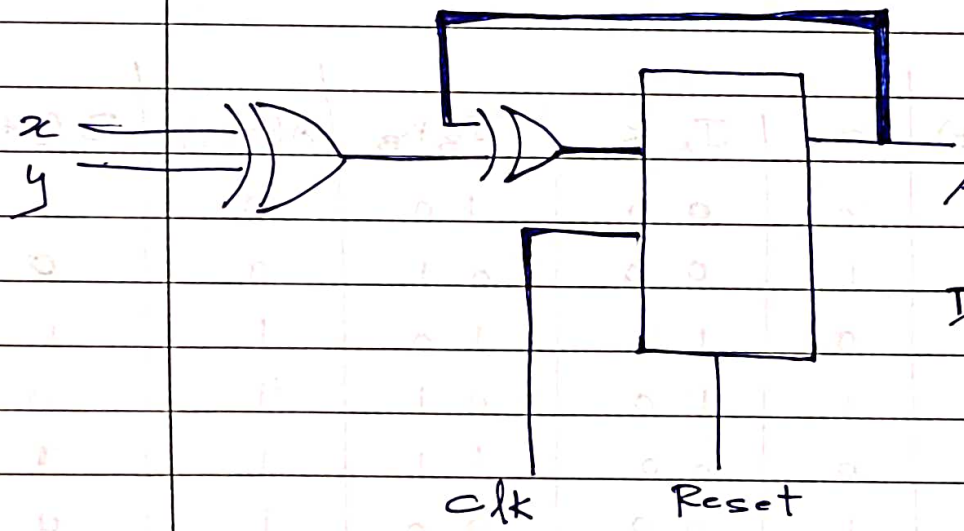
$Q(t+1) = T \oplus Q(t)$

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

JK flipflop

$Q(t+1) = J\,Q'(t) + K'\,Q(t)$

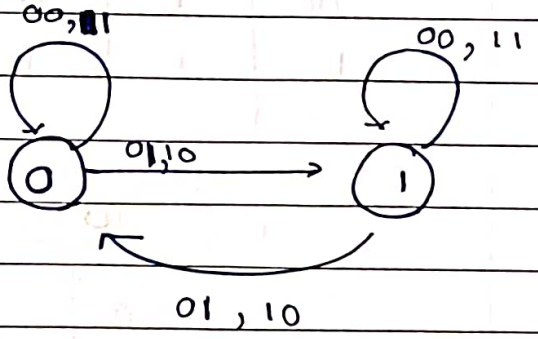
Example:



$A \rightarrow Q(t)$

$D_A = x \oplus y \oplus A$

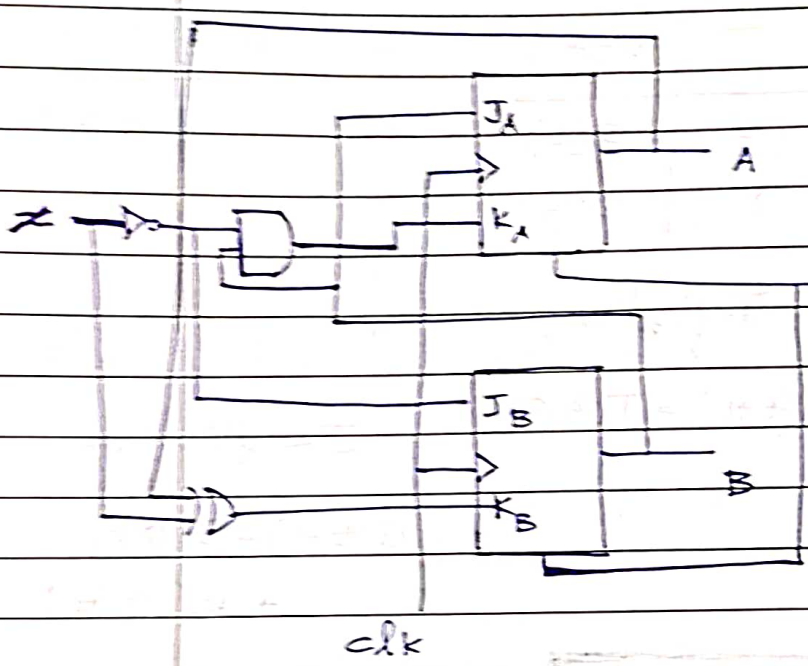
Q(t)	x	y	D_A	Q(t+1)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



transition-diagram

__/__/__

Example:



$J_A = B$

$K_A = Bx'$

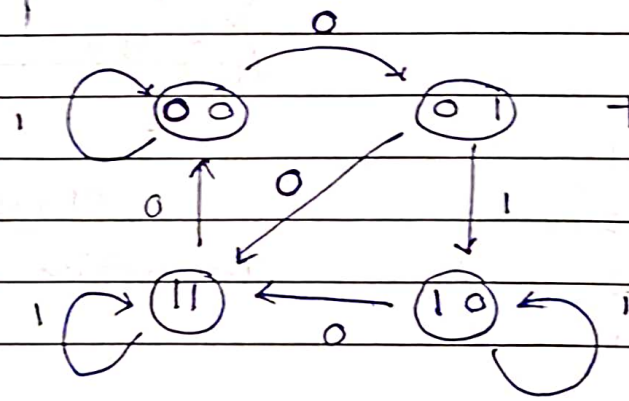
$J_B = x'$

$K_B = A \oplus x$

$A(t+1) = J_A A'(t) + K'_A A(t)$

$B(t+1) = J_B B'(t) + K'_B B(t)$

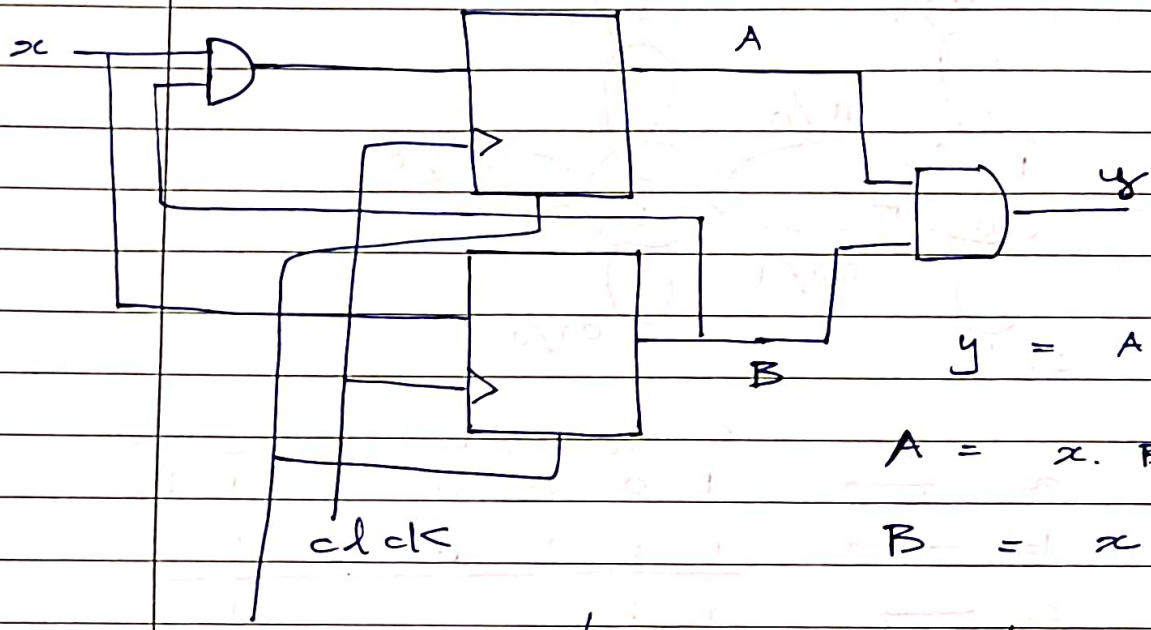
A (t)	B(t)	x	J_A K_A	J_B K_B	A(t+1)	B (t+1)
0	0	0	0 0	1 0	0	1
0	0	1	0 0	0 1	0	0
0	1	0	1 1	1 0	1	1
0	1	1	1 0	0 1	1	0
1	0	0	0 0	1 1	1	1
1	0	1	0 0	0 0	1	0
01	1	0	1 1	1 1	0	0
1	1	1	1 0	0 0	1	1



Transition diagram

__/__/____

Example:



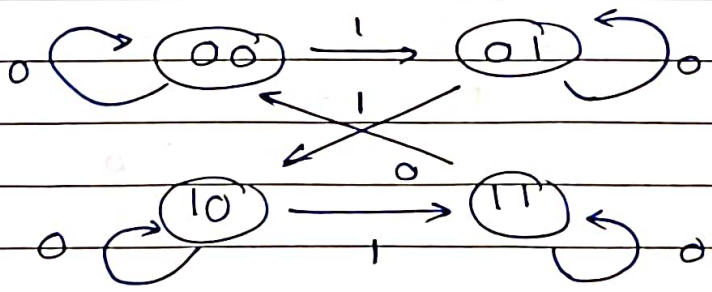
$y = A(t+1) \cdot B(t+1)$

$A = x \cdot B(t)$

$B = x$

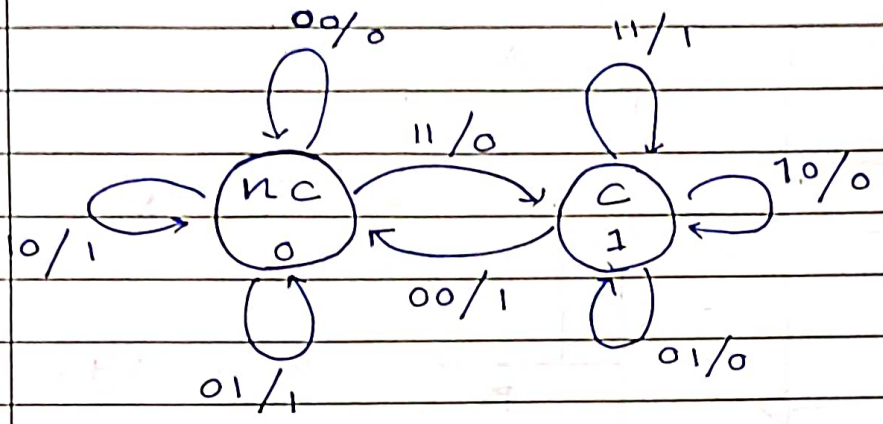
A(t)	B(t)	x	T_A	T_B	A(t+1)	B(t+1)
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	1	0	1	1	1
1	1	0	0	0	1	1
1	1	1	1	1	0	0

Transtion diagram



__/__/____

2-bit sequential adder:

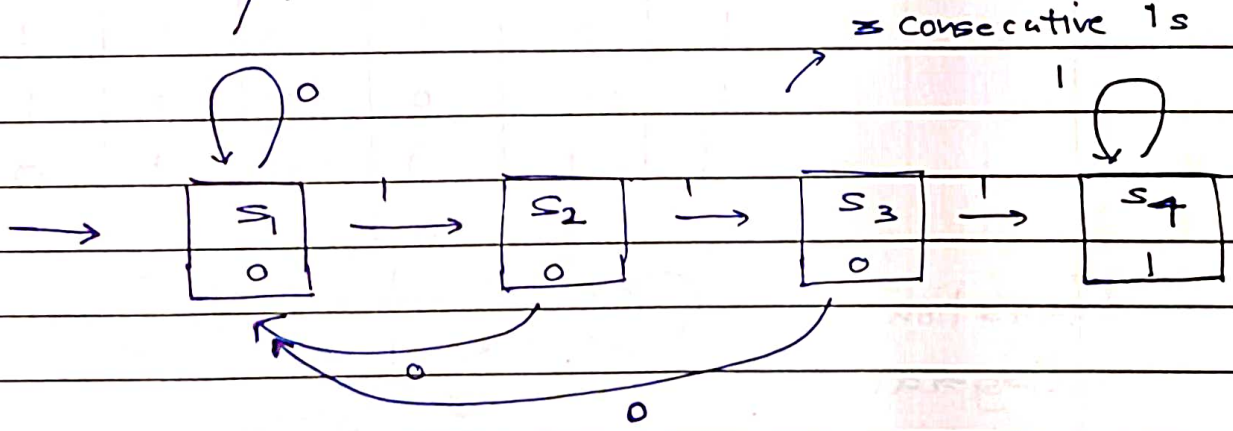


	C	NC	NC	NC	C	C	NC
a →	1	0	0	0	0	1	1
b →	1	1	1	0	0	1	1
c →	0	1	1	0	1	1	0

Patter recogniser:

x → 0011110011011110

o/p → 0000110000000010

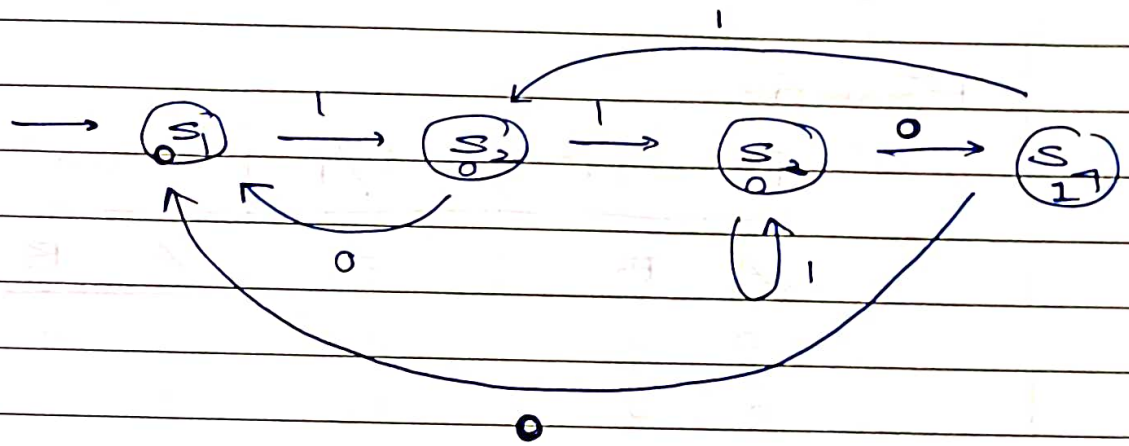


11011101

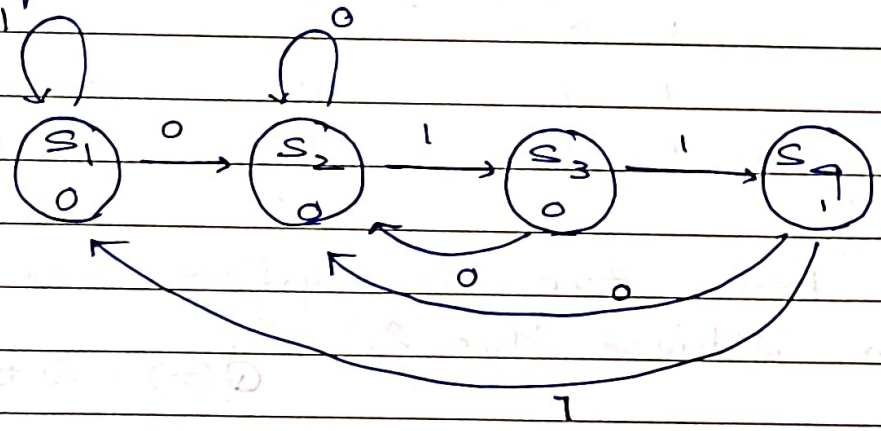
00100010

__/__/____

now for pattern 110,



now for patter 011,



__/__/____

Designing a Circuit:

Example:

Present		I/P	Next	
A	B	x	A	B
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

↓

here two JK flipflops are required to achieve this.

Characteristic table for JK →

Q(t)	Q t+1	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

↓ So

We implement two JK flip flops.

EAXAM

__/__/____

Present		I/P	Next					
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	1	0	1	x	x	1
0	1	1	0	1	0	x	x	0
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

$$J_A = \sum(2) + \sum_d(4,5,6,7)$$

x \ A/B	00	01	11	10
0	0	1 (2)	x (6)	x (4)
1	0 (1)	0 (3)	x (7)	x (5)

$= x'B$

$$K_A = \sum(7) + \sum_d(0,1,2,3)$$

x \ AB	00	01	11	10
0	x (0)	x (2)	0 (6)	0 (4)
1	x (1)	x (3)	1 (7)	0 (5)

$= xB$

$$J_B = \sum(1,5) + \sum_d(2,3,6,7)$$

x \ AB	00	01	11	10
0	0	x (2)	x (6)	4
1	1 (1)	x (3)	x (7)	1 (5)

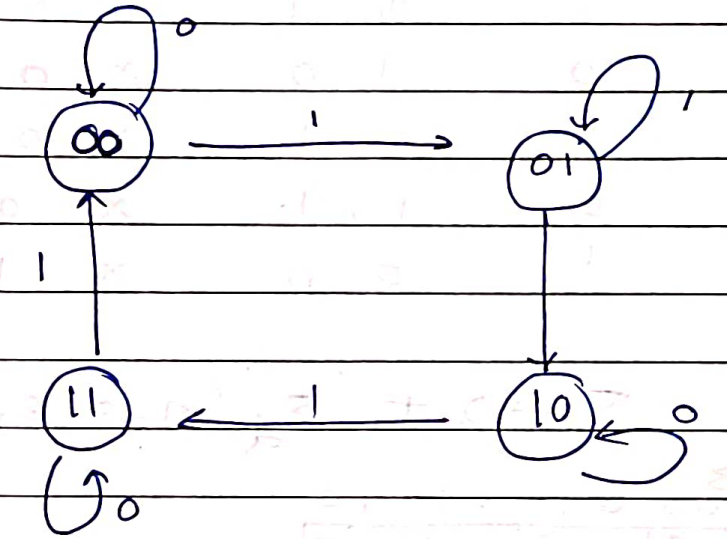
$= x$

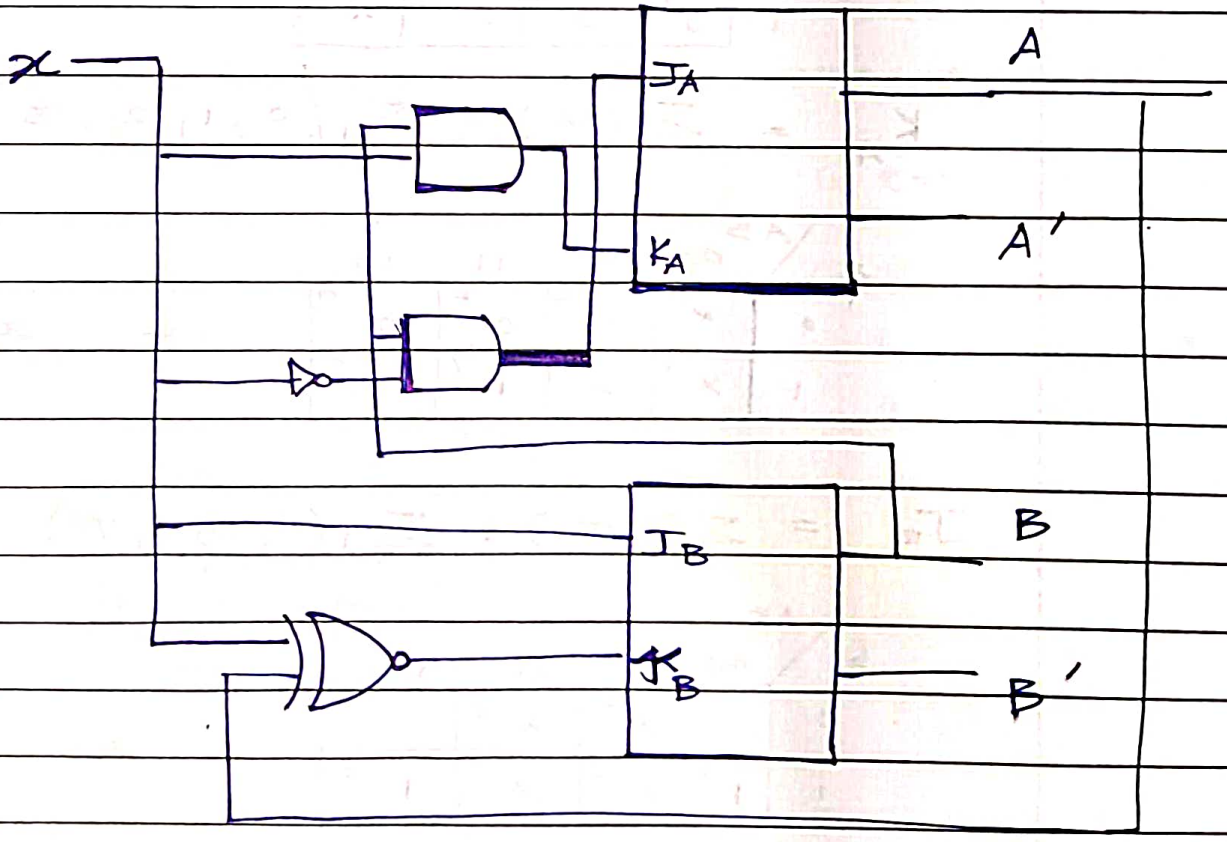
$$K_B = \Sigma(2,7) + \Sigma_d(0,1,4,5)$$

x \ AB	00	01	11	10
0	x 0	1 2	0 6	x 4
1	x 1	0 3	1 7	x 5

$= A'x' + Ax$

$= A \overline{\oplus} x$





__/__/____

Note :

Moore and mealy machine.

Example :

solution ↓

Present A	B	C	NS A	B	C	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Characteristic table of T flip flop

↳

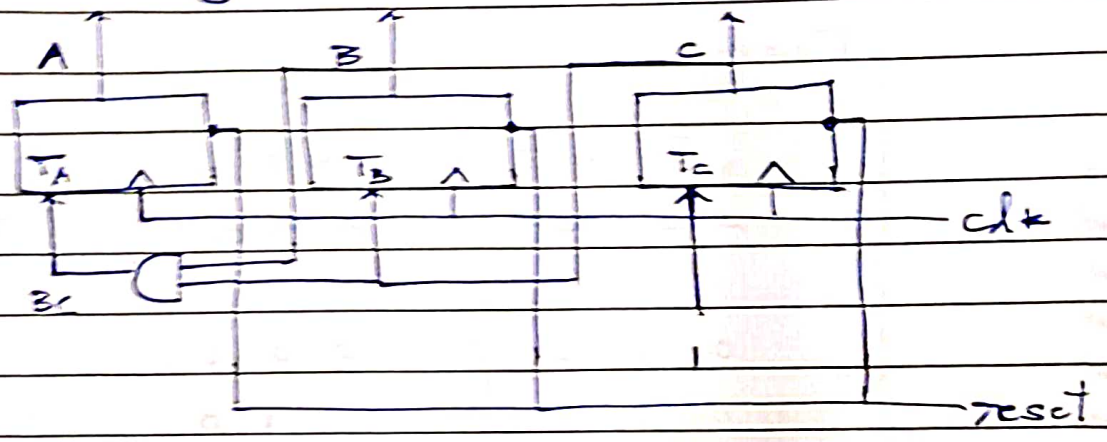
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

$T_A = \Sigma(3,7) = BC$

$T_B = \Sigma(1,3,5,7) = C$

$T_C = 1$

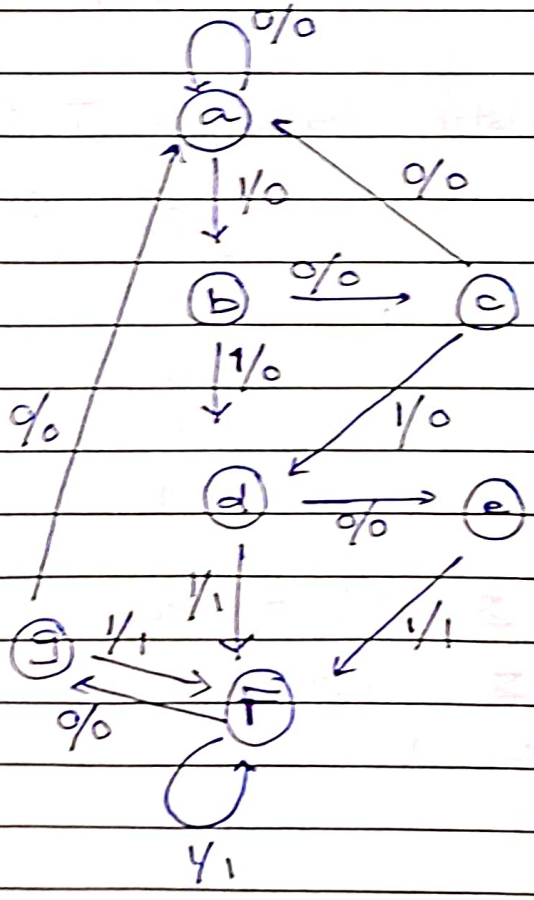
Circuit diagram:



→ State Minimization :

Steps to be Followed:

Design state diagram



7 states

∞

3 flip flops.

__/__/____

PS	NS		O/P	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

As you see, state g is identical to state e, so g is eliminated.
. . . repeat.

This is state-reduction
Now assigning binary values to the states.

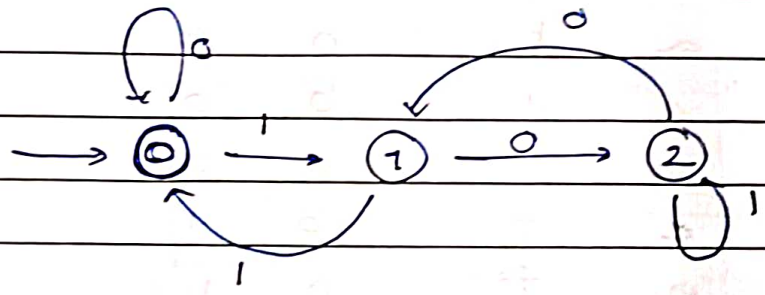
	binary	grey	Unique Code
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

01001

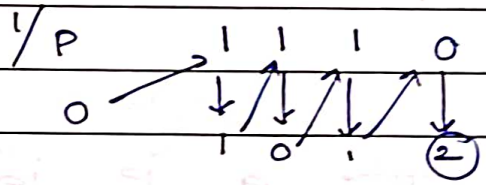
__/__/____

Example:

B mod 3



lets take an example,



→ 14

14 mod 3 = 2

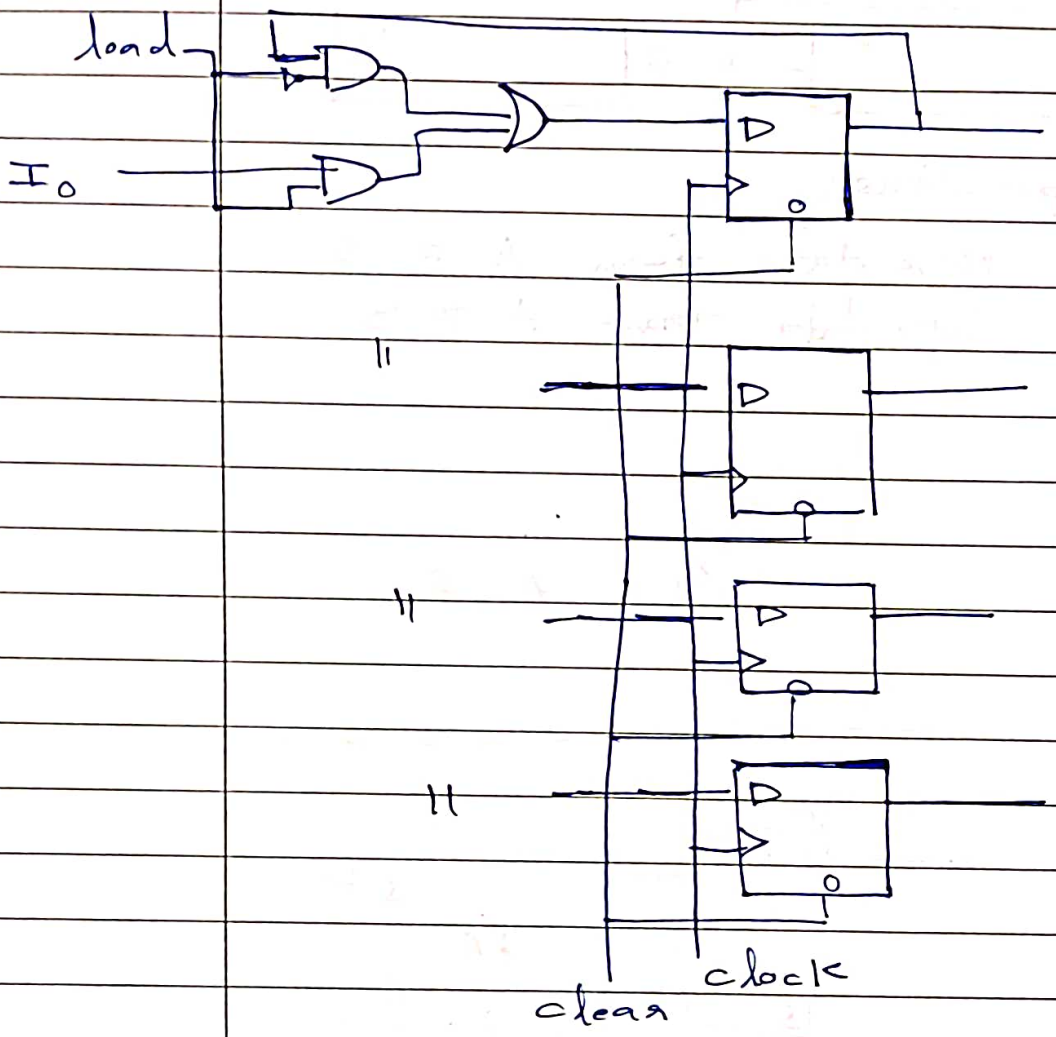
CT 2 → latches
flip flops
analysis (excitation table)
~~state~~ minimization
~~state~~ assignment

group of flip flop
Function

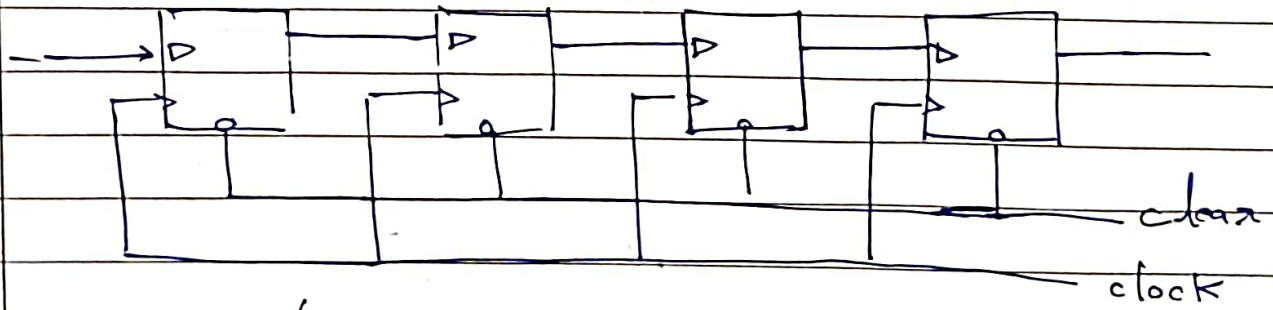
Register.
1. Storage
2. shift
SL SR

Counters

4 bit register with gates :



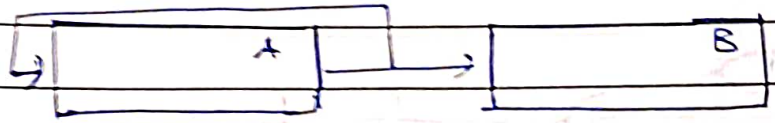
shift right :



↓

For shift left, the reverse.

two registers:

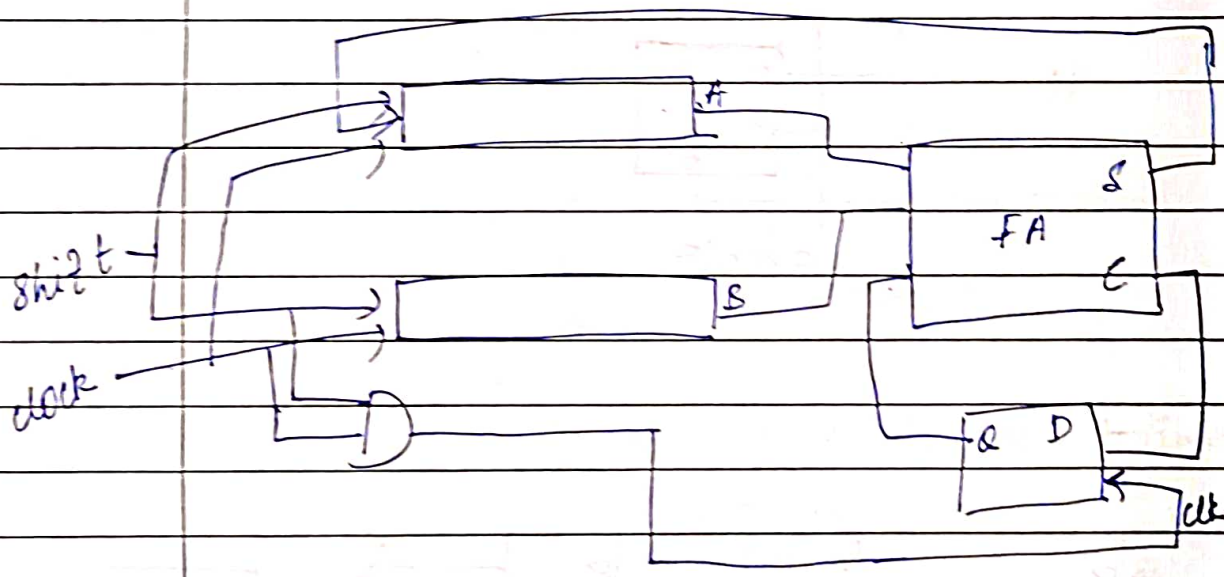


two operations:

move data from A to B

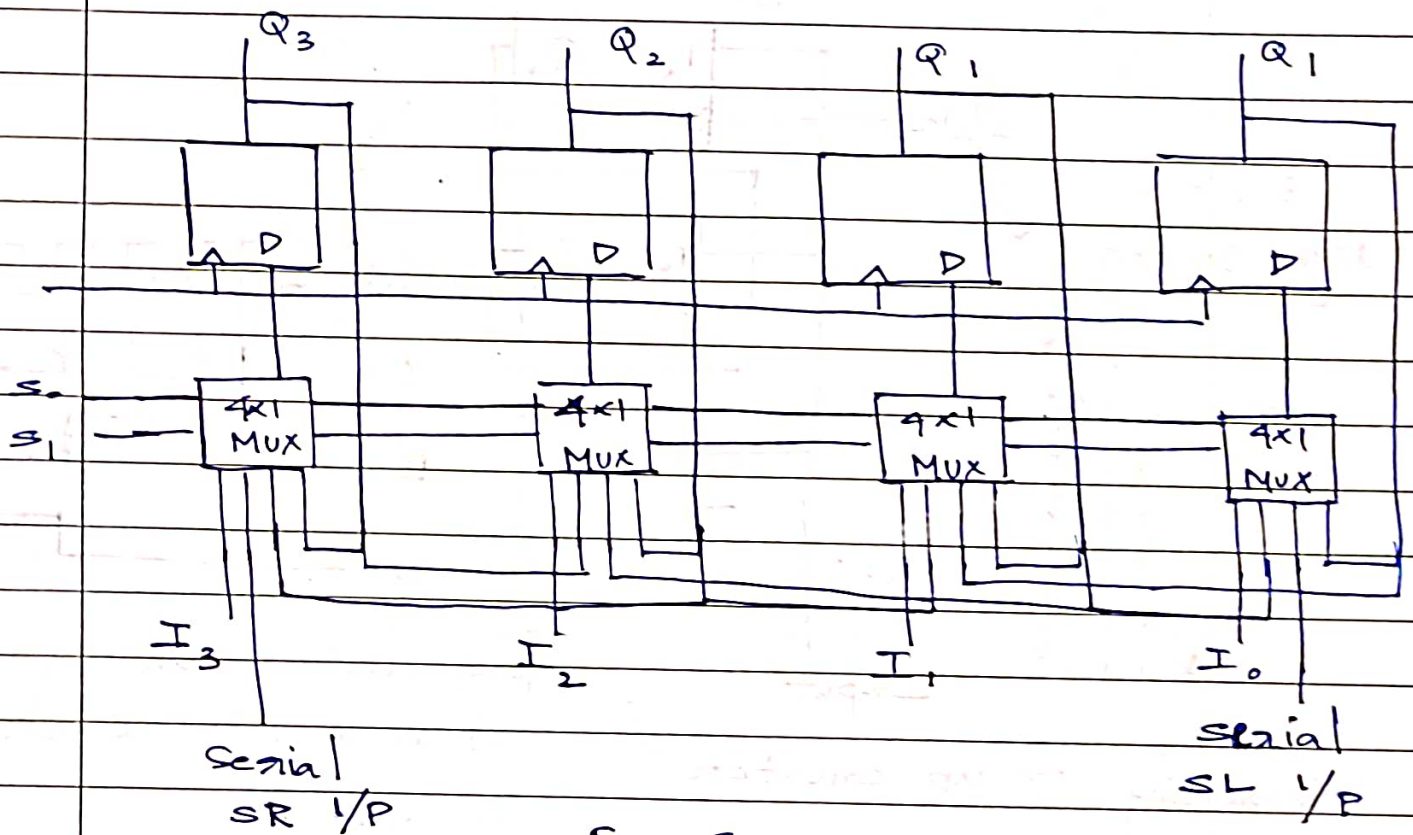
Copy data from A to B.

A + B → A //Add A, B.



Universal Register :

__/__/____

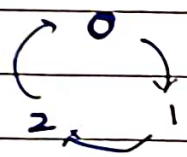


S_0	S_1	
0	0	→ no change
0	1	→ shift left
1	0	→ shift right
1	1	→ parallel load.

Counters → predefined seq of states

- sync
- nsync → Use T flipflops.

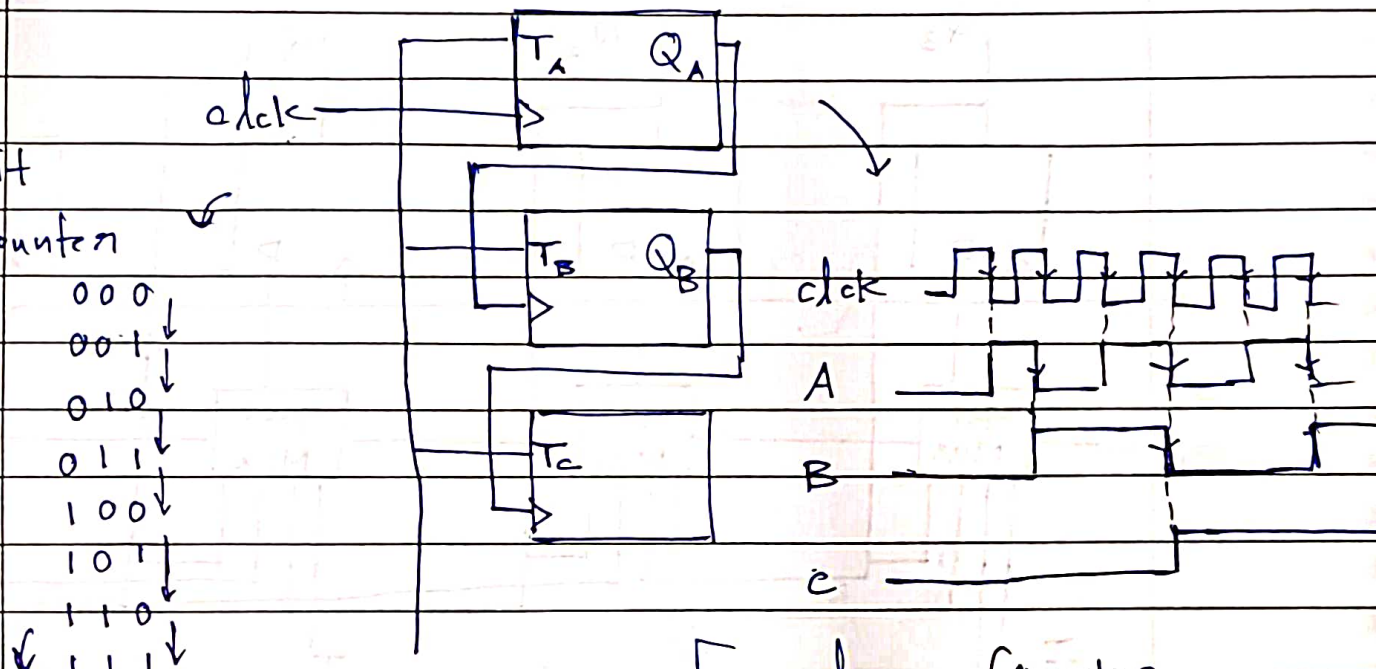
Ex:



Use
ps ns Dflipflops.

__/__/____

Async Counters:



3 bit async Counter

000
001
010
011
100
101
110
111

Input

numbers increasing → up counter

- for down counter, just negate clock.

For 4-bit Counter : (Decade Counter)

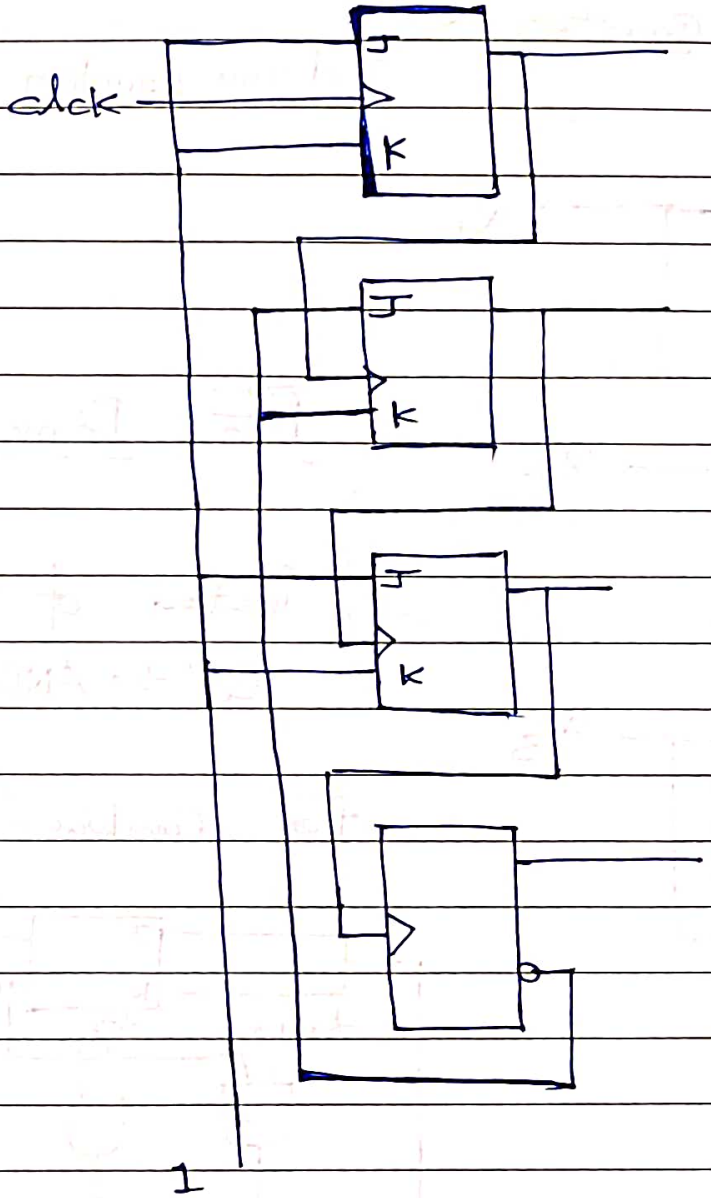
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

Q_0 → clock

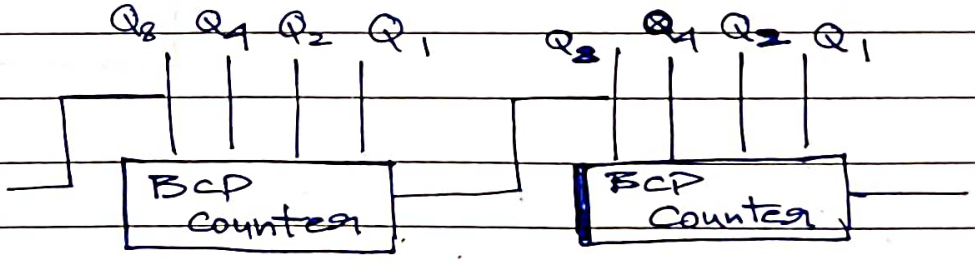
Q_1 →
For $Q_3 = 0$, normally change for Q_0 1 to 0

For $Q_3 = 0$, should not change for Q_0 1 to 0

Q_2 →
normally chang for Q_1 1 to 0



Remember to connect preset.



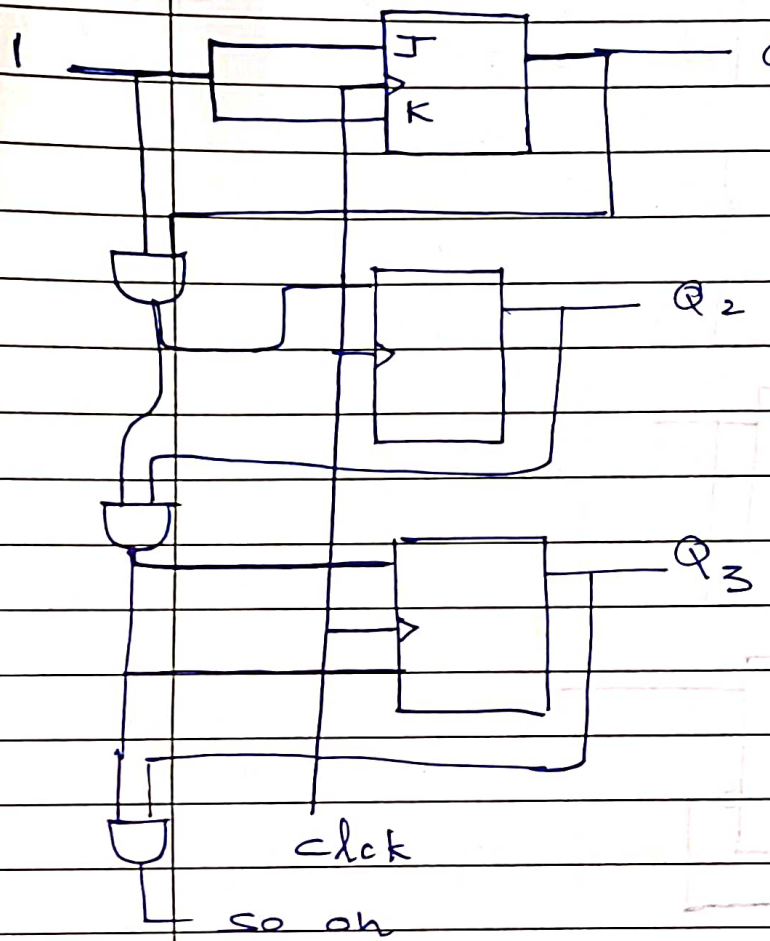
10^1 10^0

99 - 0 0 - 9

Counters.

__/__/__

Synchronous Counter : Up Counter / down Counter

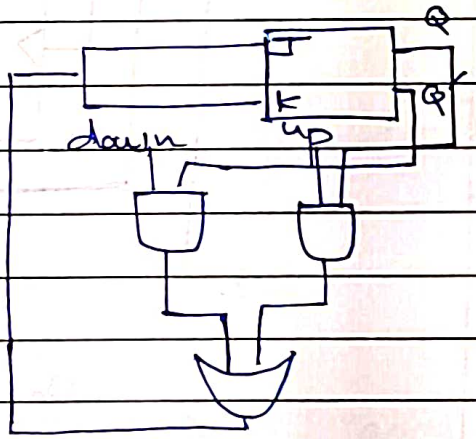


Up Counter

For Down Counter,

instead of Q_i connect Q_i' to AND Gate.

For Combined,



Synchronous decade Counter:

Ps				Ns			
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
A	B	C	D	A	B	C	D

T_A	T_B	T_C	T_D
0	0	0	1
0	0	1	1
0	0	0	1
0	1	1	1
0	0	0	1
0	0	1	1
0	0	0	1
1	1	1	1
0	0	0	1
1	0	0	1

$T_D = 1$

$T_C = \Sigma(1, 3, 5, 7)$

$= A'D$

$T_B = \Sigma(1, 7)$

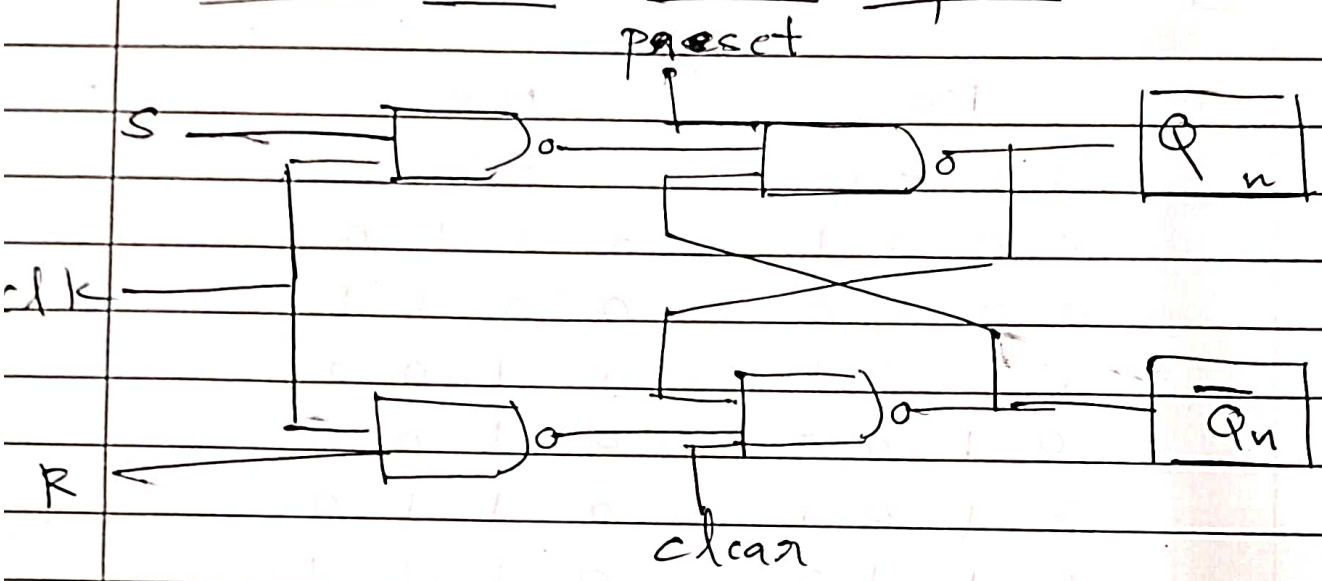
$T_A = \Sigma(7, 9)$

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	x	x	x	x
10	0	0	x	x

T_C

__/__/__

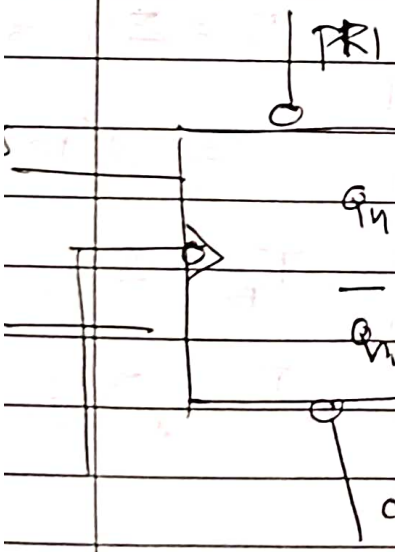
Preset and clear Inputs :



for preset = 0 → $Q_n = 1$

for clear = 0 ⇒ $Q_n = 0$ (because $Q_n' = 0$)

preset and clear are overidding and asynchsous inputs. The synchsohous inputs are S, R, J, K, D, T



Preset	clear	Q_n
0	0	not used
0	1	1
1	0	0
1	1	FF will perform normally

PR and clear are active low signal.