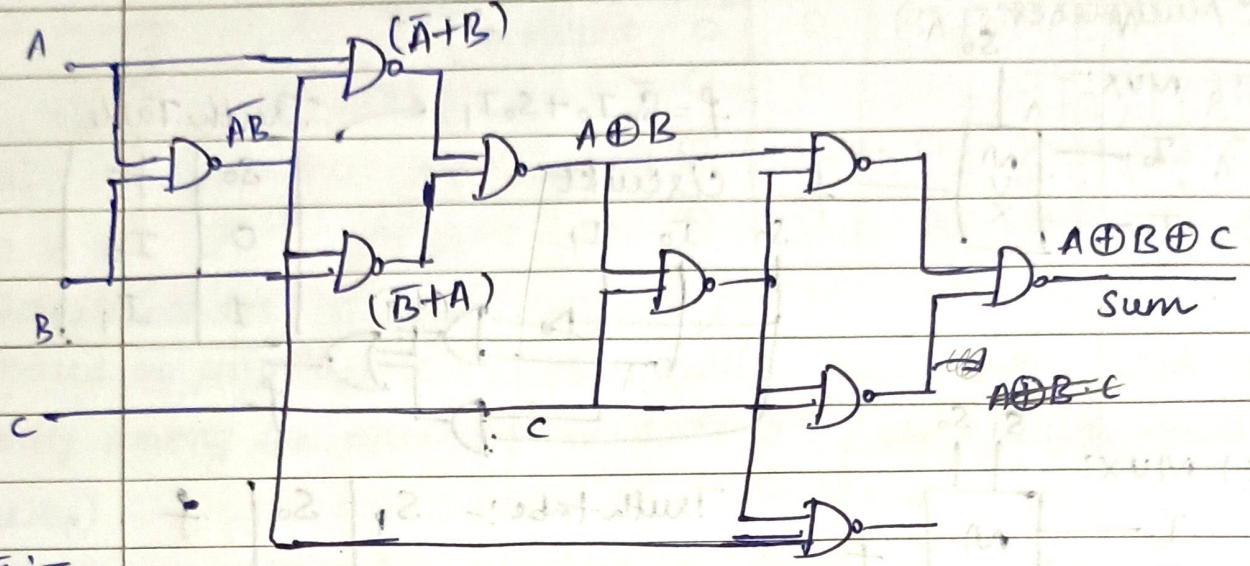
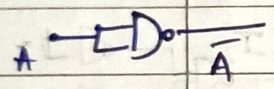
DDCO

Implementing Full adder using only NAND Gates

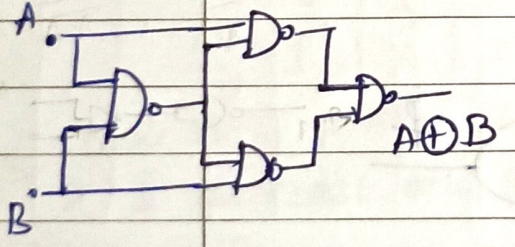


Note:-

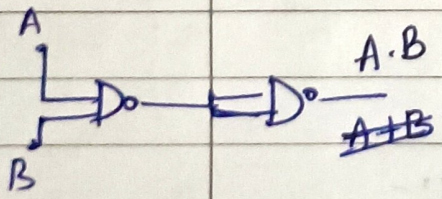
① A to $\bar{A}$ (using NAND)



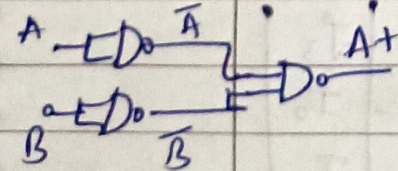
② A, B to $A \oplus B$ (using NAND)



③ A, B to $A \cdot B$ (using NAND)



④ A, B to $A+B$ (using NAND)



Carry:- $\overline{\overline{(A \oplus B) \cdot C} \cdot \overline{AB}}$

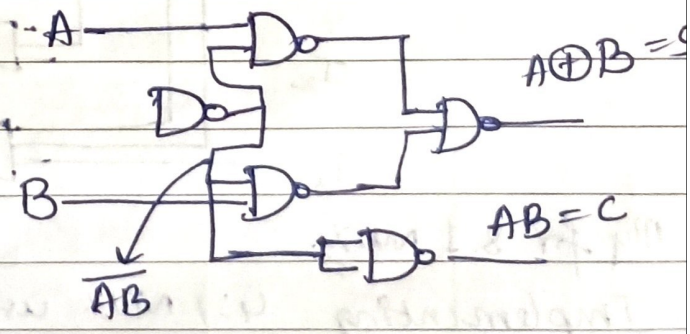
Sum:- $A \oplus B \oplus C$

Carry:- $((\overline{A \oplus B}) \cdot C) \cdot (\overline{AB})$

Half Adder:-

$S = A \oplus B$

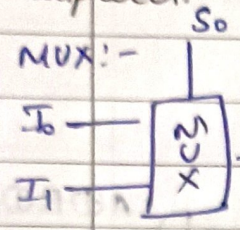
$C = AB$



__/__/____

* Multiplexer:-

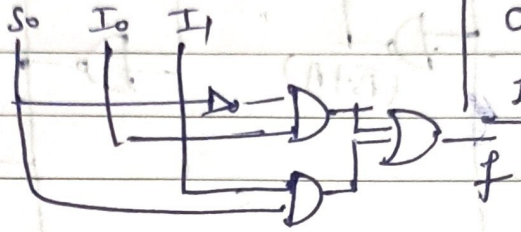
2:1 MUX:-



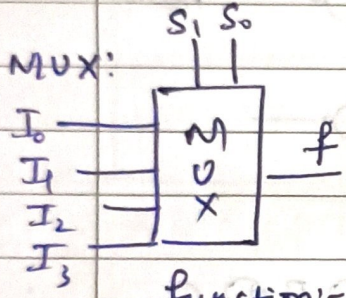
$f = \bar{S_0} I_0 + S_0 I_1$ ← Truth Table:-

S_0	f
0	I_0
1	I_1

circuit



4:1 MUX:

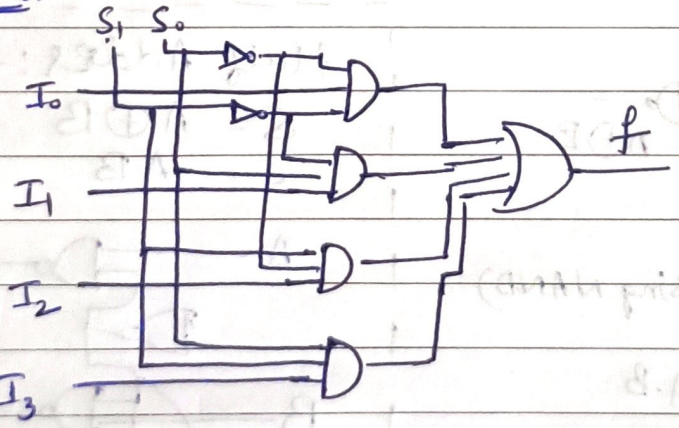


Truth table:-

S_1	S_0	f
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

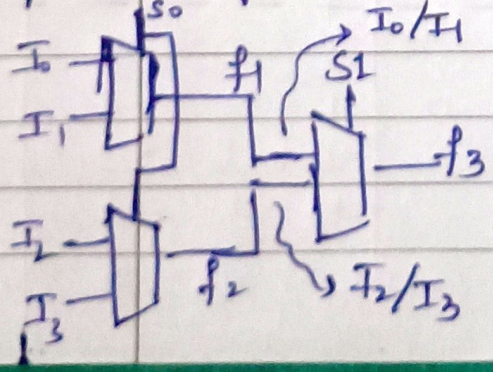
function:- $\bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$

Ckt:-



Ntly for 8:1 MUX

Implementing 4:1 MUX using 2:1 MUX:-



Truth Table:

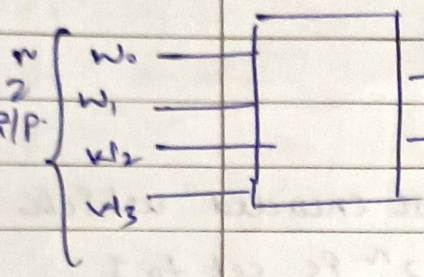
S_0	S_1	f_1	f_2	f_3
0	0	I_0	I_2	I_0
0	1	I_0	I_2	I_2
1	0	I_1	I_3	I_1
1	1	I_1	I_3	I_3

* Encoder

→ used to convert large information to small information.

$y_0 = 1$ when either w_1 or $w_3 = 1$

$y_1 = 1$ when either w_2 or $w_3 = 1$



w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

* priority encoder.

→ based on priority, this shows which input with highest priority among the ones set to 1 is 1. (check truth table for clarity)

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

i_0 i_1 i_2 i_3

→ dont care (not meaning full)

Note: if $z = 0$ then none of the inputs are set. If $z = 1$ at least one input line is set.

→ We define a set of intermediate signals, i_0, i_1, i_2, i_3 which are defined as:-

$$i_0 = \overline{w_3}\,\overline{w_2}\,\overline{w_1}\,w_0$$
$$i_1 = \overline{w_3}\,\overline{w_2}\,w_1$$
$$i_2 = \overline{w_3}\,w_2$$
$$i_3 = w_3$$

→ using the intermediate signals, the rest of the circuit of the priority encoder has the same structure as the binary encoder in Fig. 6.23, namely

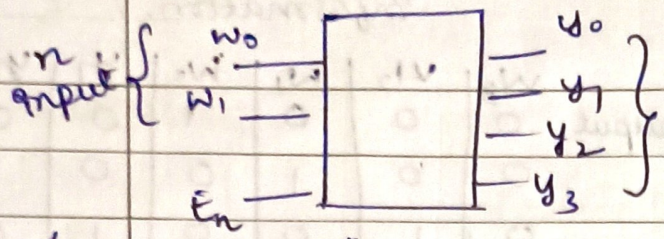
$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$

And output z is given by

$$z = i_0 + i_1 + i_2 + i_3$$

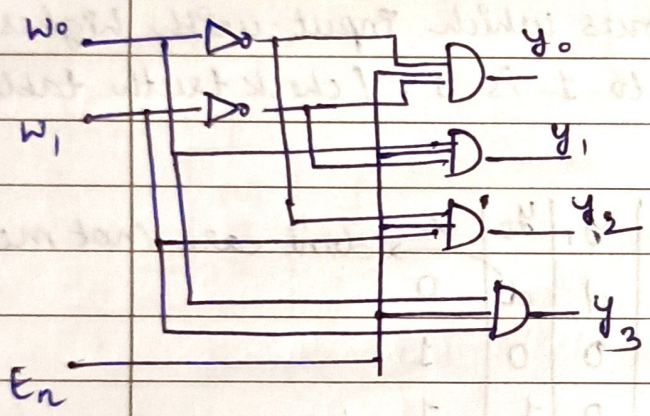
* Decoders.

__/__/____



→ The output (2^n outputs) are "one-hot encoded" which means at any given time only 1 among 2^n Ps set to 1.

CKT (for 2-4 decoder)



output expressions:-

$y_0 = \overline{w_0}\,\overline{w_1}$

$y_1 = w_0\,\overline{w_1}$

$y_2 = \overline{w_0}\,w_1$

$y_3 = w_0\,w_1$

Truth Table for (2-4 decoder)

E_n	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

* 3-8 decoder:-

Truth Table

E_n	w_2	w_1	w_0	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1
0	X	X	X	0	0	0	0	0	0	0	0

Expressions:-

$y_0 = \overline{w_2}\,\overline{w_1}\,\overline{w_0}$

$y_1 = \overline{w_2}\,\overline{w_1}\,w_0$

$y_2 = \overline{w_2}\,w_1\,\overline{w_0}$

$y_3 = \overline{w_2}\,w_1\,w_0$

$y_4 = w_2\,\overline{w_1}\,\overline{w_0}$

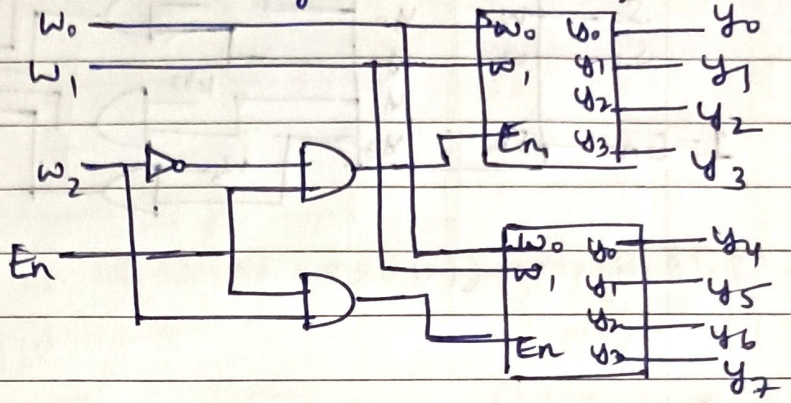
$y_5 = w_2\,\overline{w_1}\,w_0$

$y_6 = w_2\,w_1\,\overline{w_0}$

$y_7 = w_2\,w_1\,w_0$

Ckt:- Note:- can draw ps using basic gates just like 2-4 decoder previous by.

Now, using 2-4 decoder?



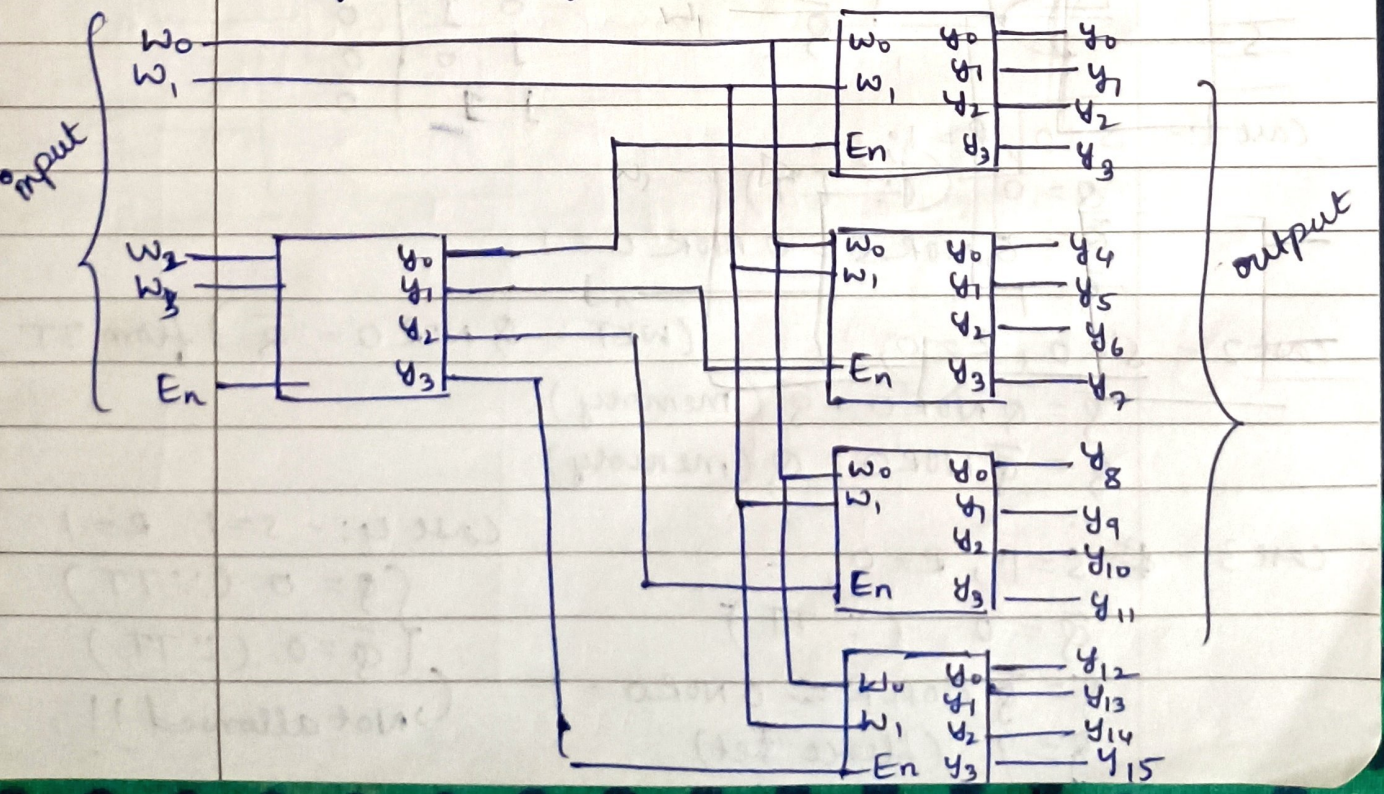
For constructing 3-8 decoder from a two 2-4 decoders.

Observations from truth table:-

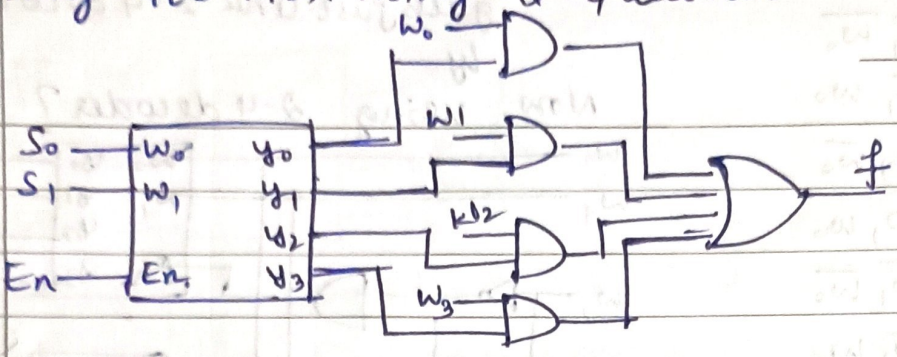
1) if $w_2 = 0$; "y_4, y_5, y_6, y_7 are always 0" and y_0, y_1, y_2, y_3 behave as 2-4 decoder with w_0, w_1 ~~playing~~ as inputs.

2) if $w_2 = 1$; "y_0, y_1, y_2, y_3 are always 0" and y_4, y_5, y_6, y_7 constitute a 2-4 decoder with w_0, w_1 as inputs.

* Similarly by observing truth table of 4-16 decoder:-



* designing 4to1 MUX using 2-4 decoder



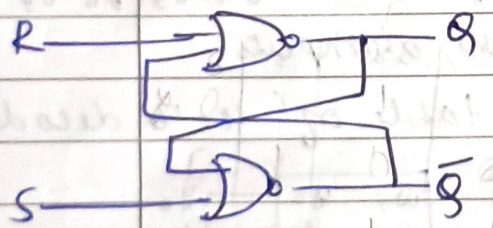
@Pg.18 diagram (textbook)
@Pg.54 (notes).

Latches:-

Doubt:- When using NOR gate SR latch when S=1, R=1, both $Q_a=0$ $Q_b=0$. How is this valid? Why the fluctuation in the timing diagram in the textbook (Chapter 7 DD - Pg No 384)

SR latch:-

CKT (using NOR gate)



T-T for NOR gate -

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Case 1:- S=0, R=1;
$Q=0$ (∵ T-T)
$\bar{Q}=Q$ NOR S = 0 NOR 0 = 1
$\bar{Q}=1$

Case 2:- S=0, R=0; (WKT Q NOR 0 = $\bar{Q}$) from TT
$\bar{Q}=Q$ NOR 0 = $\bar{Q}$ (memory)
$Q=\bar{Q}$ NOR 0 = Q (memory)

Case 3:- S=1, R=0;
$\bar{Q}=0$ (∵ TT)
$Q=\bar{Q}$ NOR R = 0 NOR 0 = 1
$Q=1$ (Hence set)

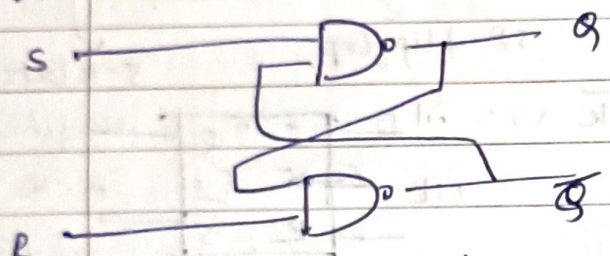
Case 4:- S=1 R=1
$Q=0$ (∵ TT)
$\bar{Q}=0$ (∵ TT)
Not allowed !!

Characteristic Table

S	R	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	(Memory)
0	1	0	1	(Reset)
1	0	1	0	(set)
1	1	Not allowed		

For NAND Gate; Similar explaination:-

Ckt:-



S-R Latch using NAND Gate

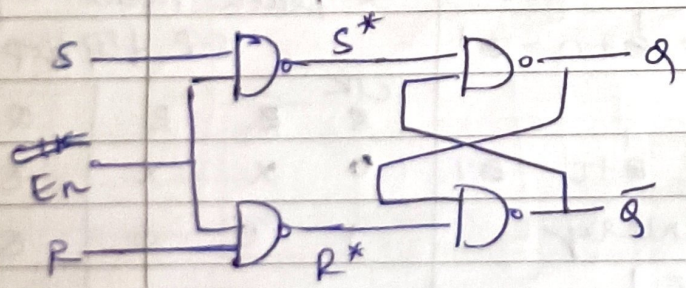
Characteristic Table

S	R	Q	$\bar{Q}$
0	0	Not allowed	
0	1	1	0
1	0	0	1
1	1	Q	$\bar{Q}$

(same Memory)

* Difference btw latch & Flip Flop:-

- Latch is ~~when~~ level triggered:-



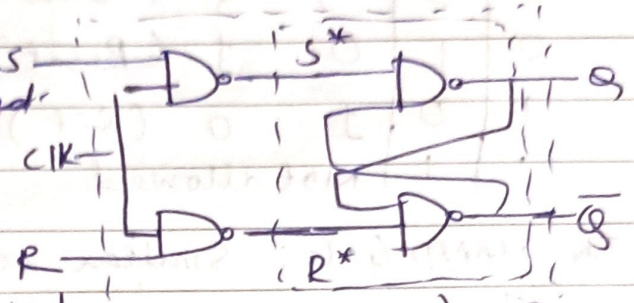
This is a latch.

Only when

* S-R Flip Flop:

CLK	S	R	Q	$\bar{Q}$
0	X	X	Memory	
1	0	0	~~Not allowed~~ Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Not Allowed	

ckt:-



SR-flipflop. S-R latch.

Case 1:- CLK = 0;

$S^* = \overline{S \cdot CLK} = \bar{S} + \overline{CLK}$

$= \bar{S} + 1$

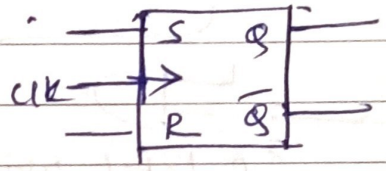
$\boxed{S^* = 1}$

$R^* = \overline{R \cdot CLK} = \bar{R} + \overline{CLK}$

$= \bar{R} + 1$

$\boxed{R^* = 1}$

If $S^* = 1$, $R^* = 1 \Rightarrow$ Memory ($\because$ TT of S-R latch)



SR-flip flop.

Case 2:- CLK = 1; S = 0, R = 0

$S^* = \bar{S} + \overline{CLK} = \bar{S} + 0 = \bar{0} + 0 = 1$

$S^* = 0$

$R^* = \bar{R} + \overline{CLK} = \bar{R} + 0 = \bar{0} + 0 = 1$

$\Rightarrow$ ~~Not allowed~~ Memory

Case 3:- CLK = 1; S = 0; R = 1

$S^* = \bar{S} + \overline{CLK} = \bar{S} + 0 = 1 + 0 = 1$

$R^* = \bar{R} + \overline{CLK} = \bar{R} + 0 = 0 + 0 = 0$

$\Rightarrow \boxed{Q = 0}$ & $\boxed{\bar{Q} = 1}$

Case 4:- CLK = 1; S = 1, R = 0

$S^* = \bar{S} + \overline{CLK} = 0 + 0 = 0$

$R^* = \bar{R} + \overline{CLK} = 1 + 0 = 1$

$\Rightarrow \boxed{Q = 1}$ & $\boxed{\bar{Q} = 0}$

* Truth Table of S-R flip flop *

CLK	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Invali

Case 5:- CLK = 1; S = 1; R = 1

$S^* = \bar{S} + \overline{CLK} + 0 + 0 = 0$; $R^* = \bar{R} + \overline{CLK} = 0 + 0 = 0 \Rightarrow$ Not Allowed

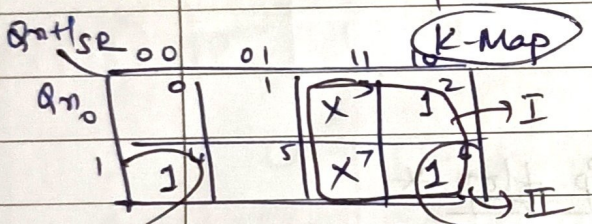
Characteristic Table:- (SR FF)

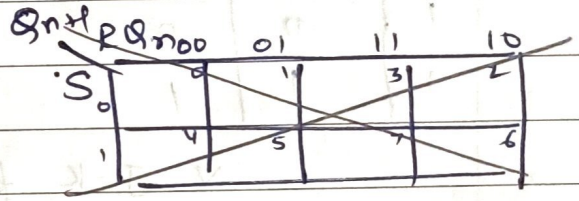
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Not Allowed (x)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Not Allowed (x)

(SR FF) Excitation Table

Q_n	Q_{n+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

K-Map

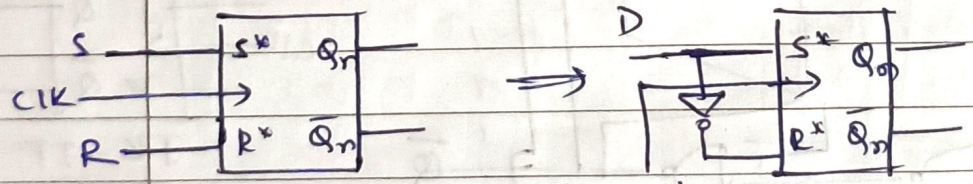




$$Q_{n+1} = S + Q_n \bar{R}$$

$$\boxed{Q_{n+1} = S + Q_n \bar{R}}$$

* D-flip flop *



SR Flip Flop ⟹ D-flip flop.

Note:- The reason to convert SR flip flop to D flip flop is to look at TT of SR flip flop, if remove the invalid case and observe. R is always $\bar{S}$. Therefore we use D & $\bar{D}$ instead of SR

T.T of D-FF →

CLK	D	Q_{n+1}	
0	x	Q_n	
1	0	0	(Reset)
1	1	1	(Set)

* Characteristic Table for D-flip flop.

__/__/____

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

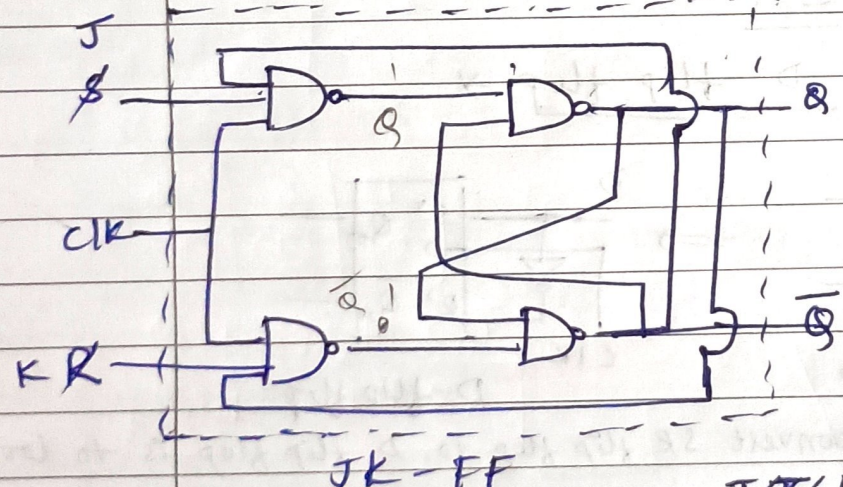
Note :- $\boxed{D = Q_{n+1}}$

* Excitation Table for D-flip flop.

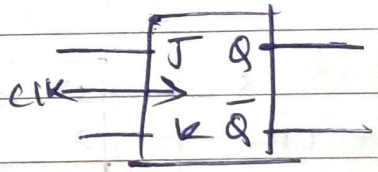
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Note ⇒ $\boxed{D = Q_{n+1}}$

* J-K flip flop *



JK-FF



JK-FF

~~T/T for S/R~~

$Q_n = 0$ $\bar{Q}_n = 1$

$Q_{n+1} = Q_n = 0$

$Q_n = 1$ $\bar{Q}_n = 0$

$Q_{n+1} = 0$

Analysis:-

Case 1: $CLK = 0 \Rightarrow S^* = 1, R^* = 1 \Rightarrow$ Memory

Case 2:- $CLK = 1$; $J = 1$; $K = 0$,

Assume Q_n as previous state; $S^* = \overline{Q_n} \cdot 1 \cdot 1 = Q_n$

$R^* = \overline{0} = 1$ → Now, If $Q_n = 0$, $S^* = 0$, $R^* = 1 \Rightarrow Q_{n+1} = 1$

If $Q_n = 1$, $S^* = 1$, $R^* = 1 \Rightarrow Q_{n+1} = Q_n = 1$

$\therefore \boxed{J = 1, K = 0, Q_n = 1}$

Case 3: $C|K = 1$; $J = 0$, $K = 1$

Similar Analysis $\Rightarrow$ $\boxed{Q_{n+1} = 0}$

__/__/____

Case 4: $C|K = 1$; $J = 1$, $K = 1$

case 1: Assume $Q_n = 0$, $\overline{Q_n} = 1$

$R^* = 1$ $S^* = 0$ $\Rightarrow$ $Q_{n+1} = 1$

Assume $Q_n = 1$, $\overline{Q_n} = 0$

$R^* = 0$ $S^* = 1$ $\Rightarrow$ $Q_{n+1} = 0$

$\therefore$ $Q_{n+1} = \overline{Q_n}$ (Toggle) (Actually raising)

TT for JK-FF

C|K	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q_n}$ (Toggle)

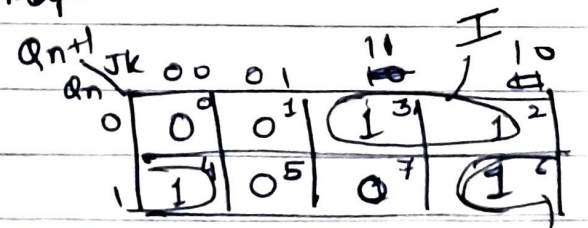
Characteristic Table for JK-FF

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation Table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

K-Map



$Q_{n+1} = I + II$

$\boxed{Q_{n+1} = \overline{Q_n} J + Q_n \overline{K}}$

for J

Q_n \ Q_{n+1}	0	1
0	1	1
1	x	x

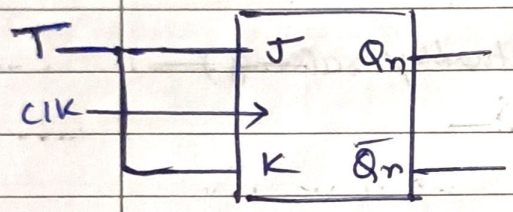
$J = Q_{n+1}$

for K.

Q_n \ Q_{n+1}	0	1
0	x	x
1	1	0

$K = \overline{Q_{n+1}}$

* T-flip flop.



T.T for T-FF

CLK	T	Q_{n+1}
0	X	Q_n (Memory)
1	0	Q_n (Memory)
1	1	$\overline{Q_n}$ (Toggle)

$J = K = 0$

$J = K = 1$

Characteristic Table of T-FF

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

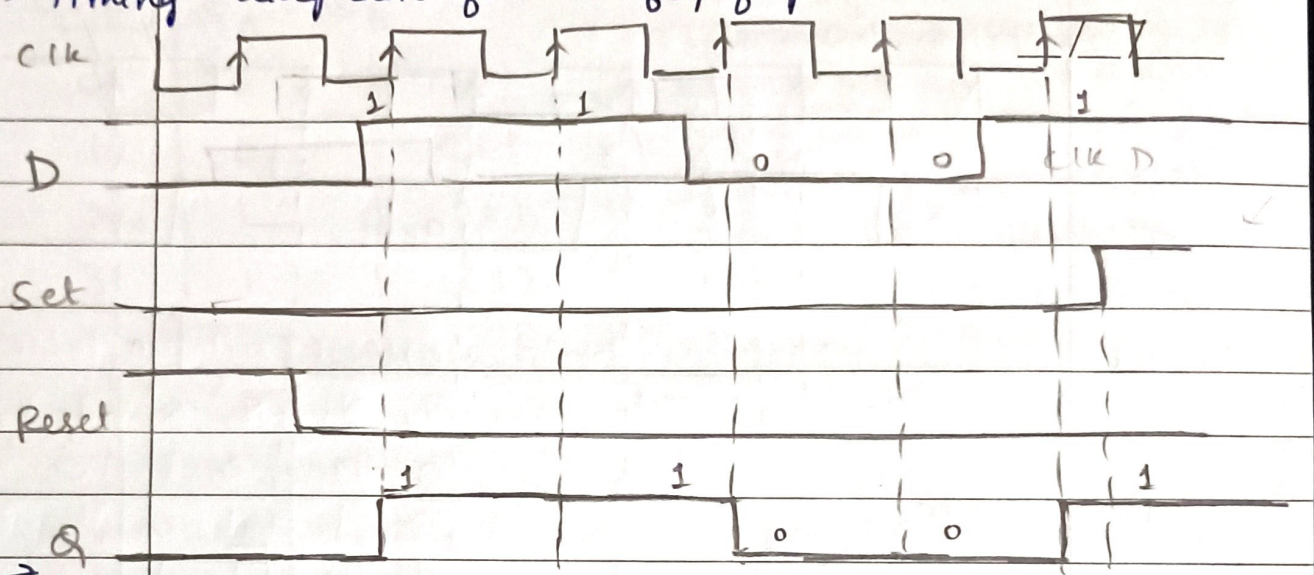
K Map

Q_n \ Q_{n+1}, T	0	1
0	0	(1)
1	(1)	0

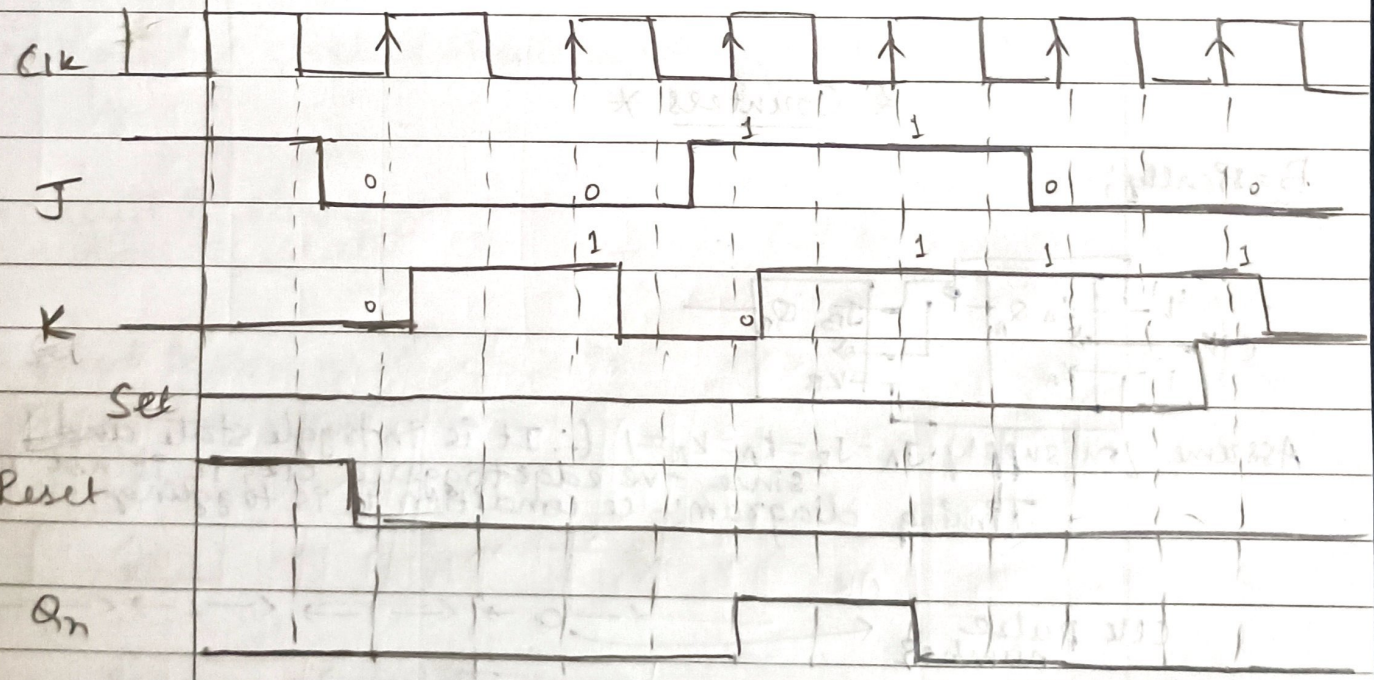
$Q_{n+1} = T\overline{Q_n} + Q_n\overline{T}$

$Q_{n+1} = T \oplus Q_n$

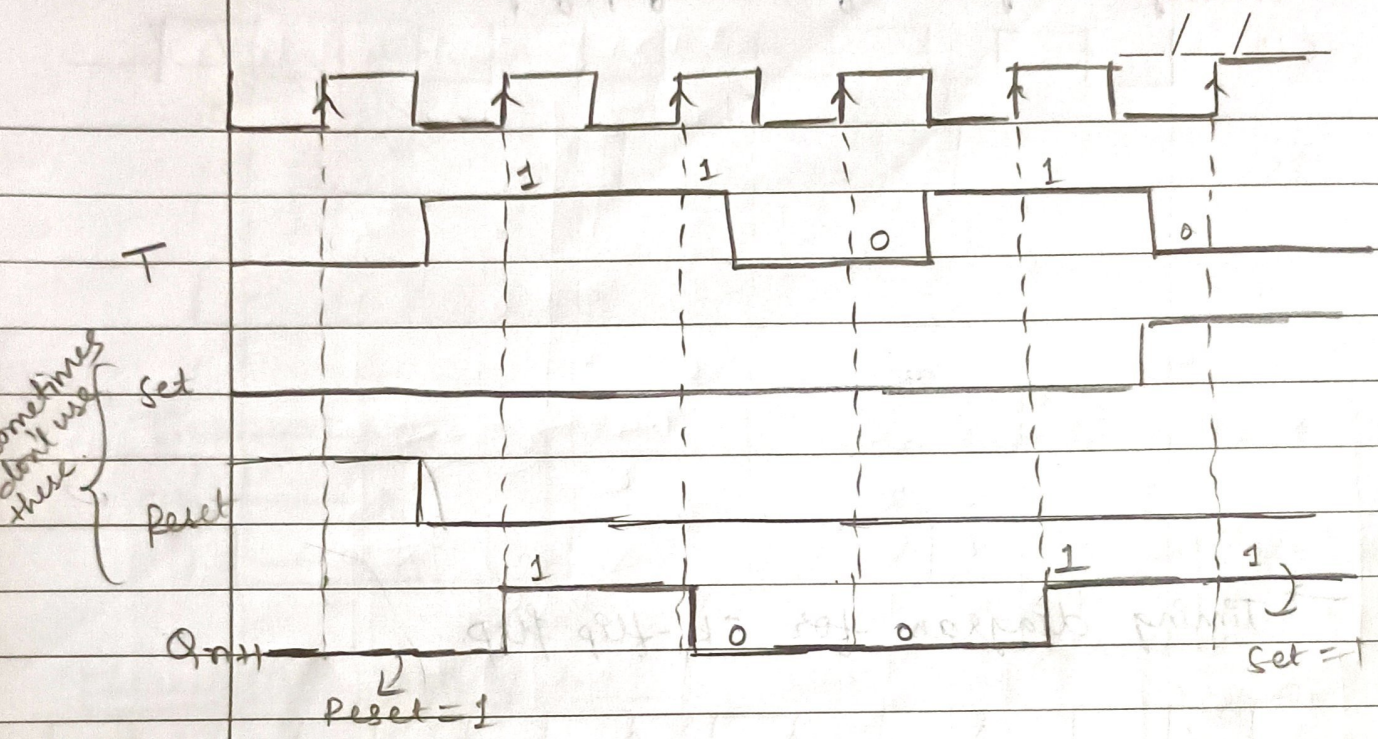
→ Timing diagram for D-flip flop.



→ Timing diagram for JK-flip flop.

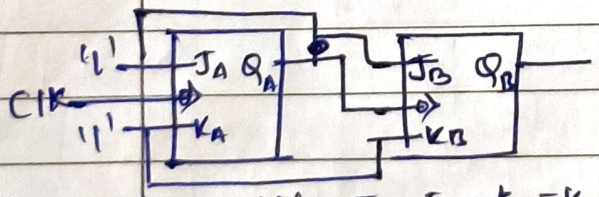


→ Timing diagram for T-flip flop.



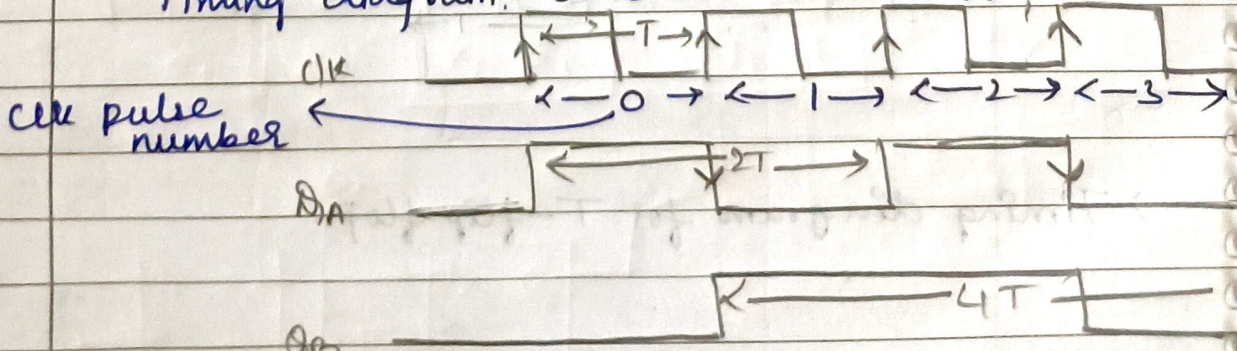
* Counters *

Basically;



Assume you supply $J_A = J_B = K_A = K_B = 1$ ∴ It is in toggle state and since -ve edge triggered clk, it is not ra ce condition it is toggling

Timing diagram:-



∴ It is evident that $T_A = 2T$ & $T_B = 2T_A = 4T$

∴ If $f = f_{clock}$ then $f_A = f/2$ & $f_B = f_A/2 = f/4$.

C_{CLK} (clk pulse number)	Q_A	Q_B	
0	0	0	(0)
1	0	1	(1)
2	1	0	(2)
3	1	1	(3)

Note :- If we want to count from 0 to 3 we use 2 ff bcz $2^2 = 4$ and we are counting 4 numbers. This implies If we want to count from 0 to 15 i.e 16 numbers we use $\log_2 16 = 4$ ff.

Difference btw Synchronous & Asynchronous Counters.

Asynchronous	Synchronous
• FF are connected in such a way that output of one FF drives the clk of another FF	• The o/p of FF are not connected to the clk of another FF.
• FF are not clocked simultaneously	• FF are clocked simultaneously
• Circuit is simple for more number of states	• Circuit becomes complicated as number of states increases.
• Speed is slow as the clock is propogated through no. of stages	• speed is faster as clock is given at the same time.

* Designing of Synchronous up counter.

Steps

Step 1: Decide the number of ff

Step 2: Excitation table of FF

Step 3: state diagram and circuit excitation table

Step 4: Obtain simplified eqn using k-map

Step 5: Draw the logic diagram (circuit).

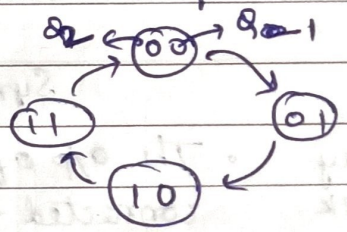
Q) Design 2-bit synchronous up counter

Step 1:- no. of bits = 2 hence 2 FF's. (JK FF)

Step 2:- Excitation table of JK FF:

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3:-

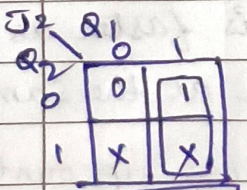


Circuit Excitation Table.

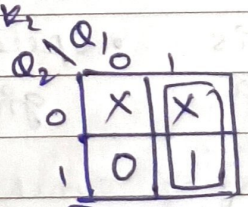
Q_2	Q_1	Q_2^*	Q_1^*	J_2	K_2
0	0	0	1	0	X
0	1	1	0	1	X
1	0	1	1	X	0
1	1	0	0	X	1

J_1	K_1
1	X
X	1
1	X
X	1

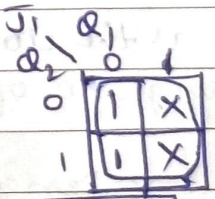
Step 4:- K Map.



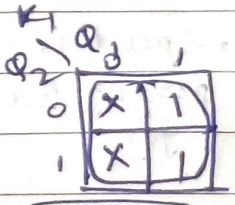
$J_2 = Q_1$



$K_2 = Q_1$

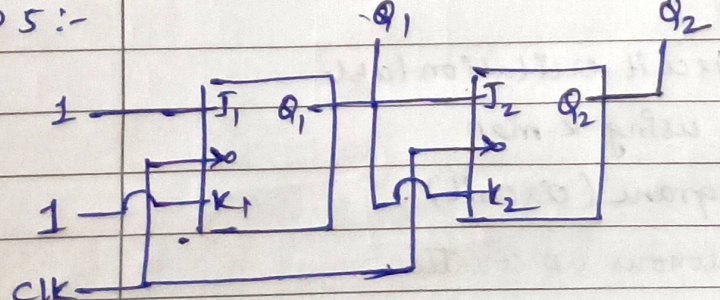


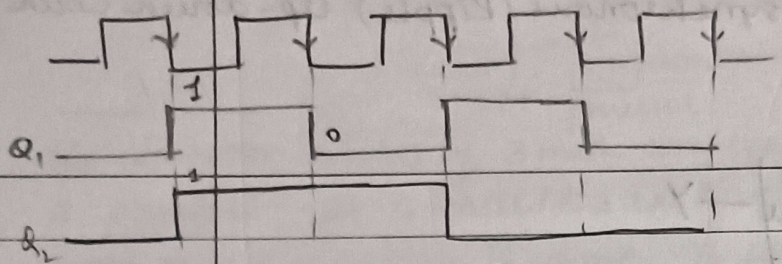
$J_1 = 1$



$K_1 = 1$

Step 5:-





__/__/____

WKT: $J_2 = Q_1$ $K_2 = Q_1$

case 1: $Q_1 = 0$; $J_2 = 0$; $K_2 = 0$ $\Rightarrow$ memory

$\therefore Q_2^* = Q_2$

case 2: $Q_1 = 1$; $J_2 = 1$; $K_2 = 1$ $\Rightarrow$ Toggle

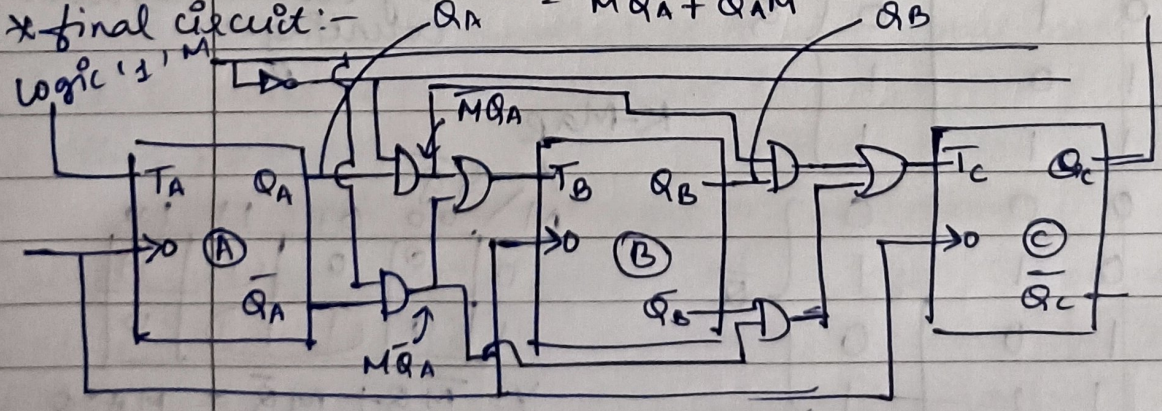
$\therefore Q_2^* = \overline{Q_2}$

* 3-bit up Down Synchronous counter

* follow the 5 steps similarly but use another control variable M with Q_A Q_B Q_C and draw the excitation table for the circuit. Your table will have 16 rows if T flip flop is used.
* follow the same procedure, use ~~to vaiia~~ 4 variable K-Map to obtain the following:

$$T_A = 1 \quad T_B = M \oplus Q_A = M\bar{Q}_A + Q_A\bar{M} \quad T_C = \bar{M}Q_BQ_A + M\bar{Q}_B\bar{Q}_A$$

* final circuit:-

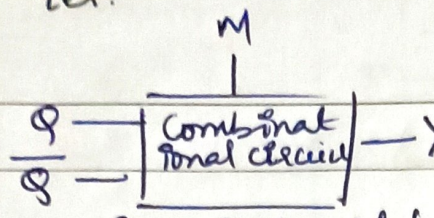


* For J-K flipflop:-

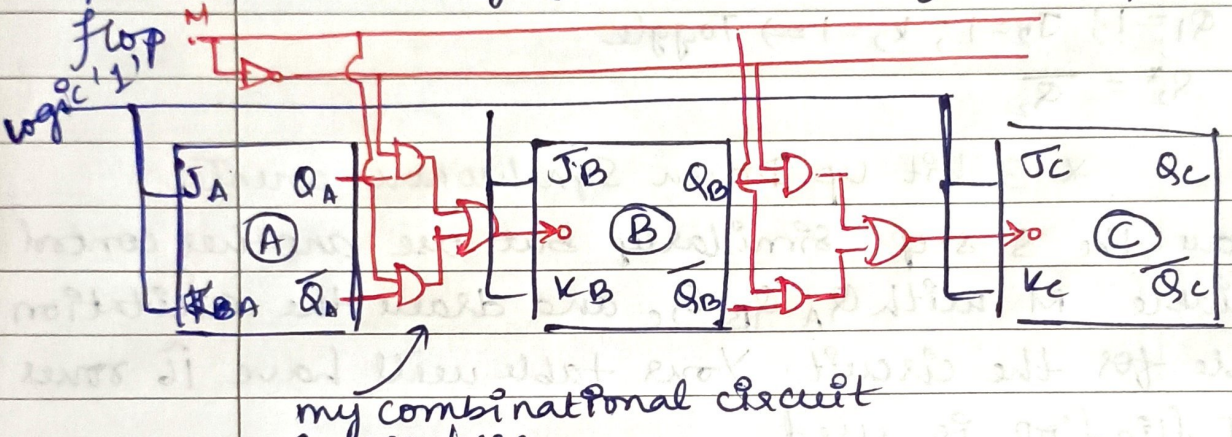
~~Use same~~

3-bit / 4-bit Asynchronous (Ripple) up-down Counter.

__/__/____



- For up counting Q should be clk for subsequent ~~out~~ flip flop
- For down counting $\overline{Q}$ should be clk for subsequent flip flop



T-T

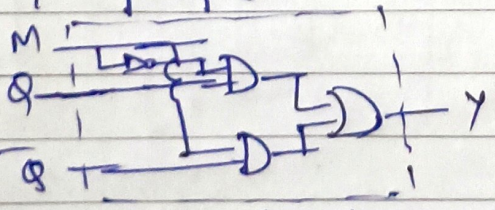
M	Q	$\overline{Q}$	Y	
0	0	0	0	Q
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	0	$\overline{Q}$
1	0	1	1	
1	1	0	0	
1	1	1	-1	

- Here we assume a rule if M=0 ⇒ uprounting else M=1 ⇒ down counting

K-Map

Y	$Q\overline{Q}$ = 00	01	11	10
M=0	0	0	1	1
M=1	0	1	1	Q

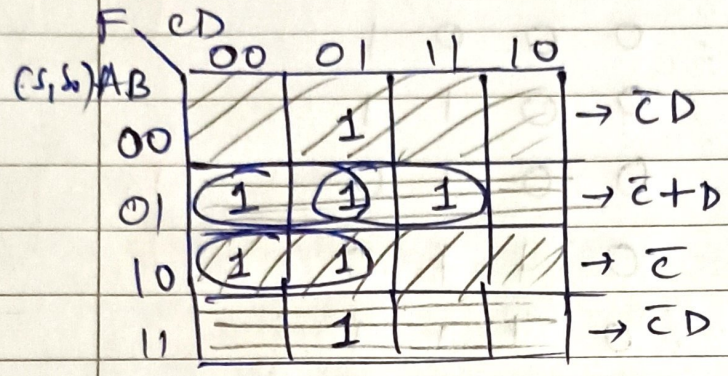
$Y = \overline{M}Q + M\overline{Q} = M \oplus Q$

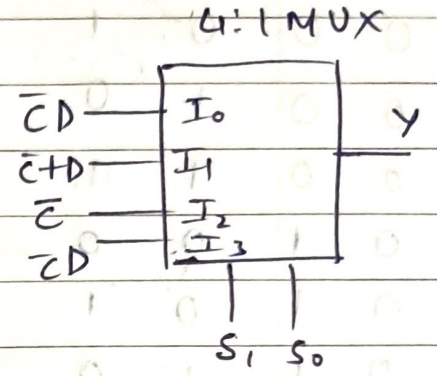


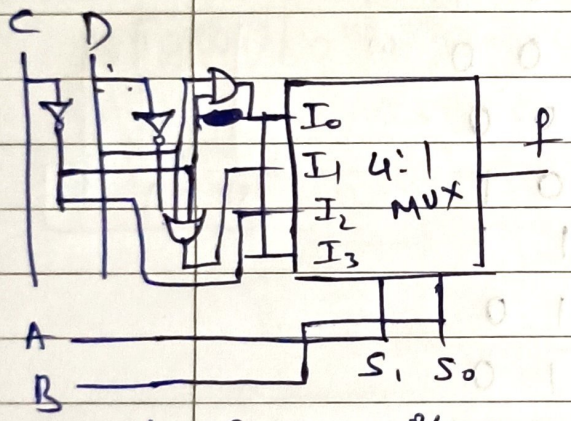
- Note:- for 4-bit also do same just add one more flipflop.

Implementing Boolean functions using MUX.

Q) $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$ using 4:1 MUX







S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

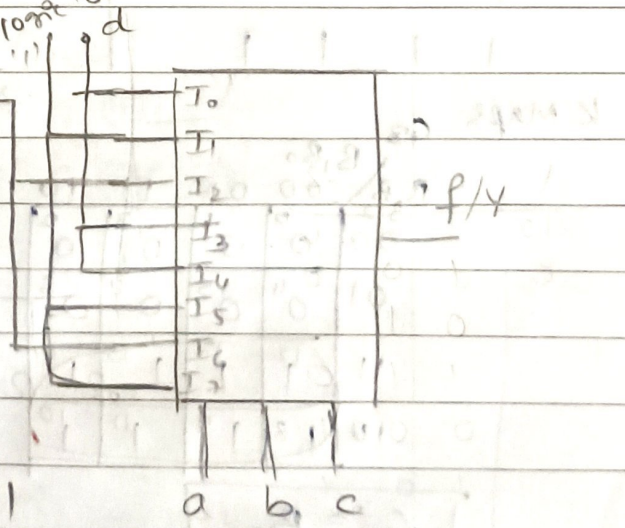
Let
~~A = S1~~
~~B = S0~~
~~A = S0~~
~~B = S1~~
A = S1
B = S0

Logic circuit.

* Using 8:1 MUX *

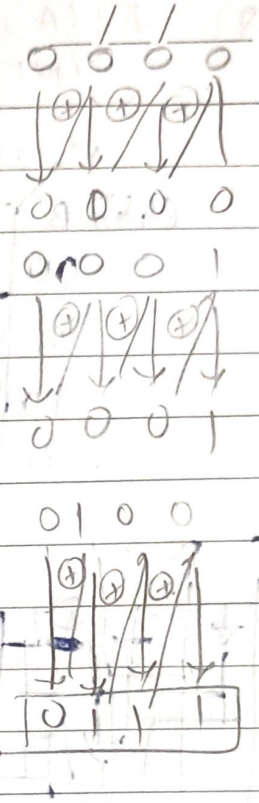
Minterms:-
$\Sigma m(1,4,5,7,9,12,13)$

	$\bar{d}$	d	
D_0	0	①	= d
D_1	2	3	= 0
D_2	④	⑤	$d+\bar{d}=1$
D_3	6	⑦	= d
D_4	8	⑨	= d
D_5	10	11	= 0
D_6	⑫	⑬	$= d+\bar{d}=1$
D_7	14	15	= 0



Binary to Gray Code converter

Inputs				Outputs				
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0	
0	0	0	0	0	0	0	0	(0)
0	0	0	1	0	0	0	1	(1)
0	0	1	0	0	0	1	1	(3)
0	0	1	1	0	0	1	0	(2)
0	1	0	0	0	1	1	0	(6)
0	1	0	1	0	1	1	1	(7)
0	1	1	0	0	1	0	1	(5)
0	1	1	1	0	1	0	0	(4)
1	0	0	0	1	1	0	0	
1	0	0	1	1	1	0	1	
1	0	1	0	1	1	1	1	
1	0	1	1	1	1	1	0	
1	1	0	0	1	0	1	0	
1	1	0	1	1	0	1	1	
1	1	1	0	1	0	0	1	
1	1	1	1	1	0	0	0	



K Maps

G_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	1 (12)	1 (13)	1 (15)	1 (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

$$G_3 = B_3$$

G_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	1 (4)	1 (5)	1 (7)	1 (6)
11	0 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

$$G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2$$

$$G_2 = B_3 \oplus B_2$$

G_1

B_3B_2 \ B_1B_0	00	01	11	10
00	0 (0)	0 (1)	1 (3)	1 (2)
01	1 (4)	1 (5)	0 (7)	0 (6)
11	1 (12)	1 (13)	0 (15)	0 (14)
10	0 (8)	0 (9)	1 (11)	1 (10)

$G_1 = B_2 \cdot \overline{B}_1 + B_1 \cdot \overline{B}_2$

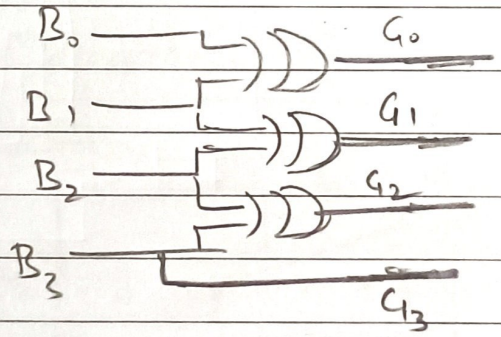
$\boxed{G_1 = B_1 \oplus B_2} = \boxed{B_2 \oplus B_1}$

G_0

B_3B_2 \ B_1B_0	00	01	11	10
00	0 (0)	1 (1)	0 (3)	1 (2)
01	0 (4)	1 (5)	0 (7)	1 (6)
11	0 (12)	1 (13)	0 (15)	1 (14)
10	0 (8)	1 (9)	0 (11)	1 (10)

$G_0 = B_1\overline{B}_0 + \overline{B}_1 B_0$

$\boxed{G_0 = B_0 \oplus B_1}$



For Gray code to binary code conversion :-

$B_3 = G_3$

$B_2 = B_3 \oplus G_2$

$B_1 = B_2 \oplus G_1$

$B_0 = B_1 \oplus G_0$

$\Rightarrow$

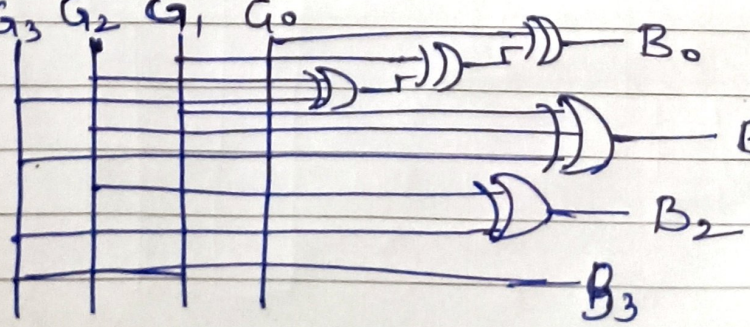
$B_3 = G_3$

$B_2 = G_3 \oplus G_2$

$B_1 = G_3 \oplus G_2 \oplus G_1$

$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$

Circuit :-



[instead of 1, 3 input XOR gate use 2, 2 input XOR gate]