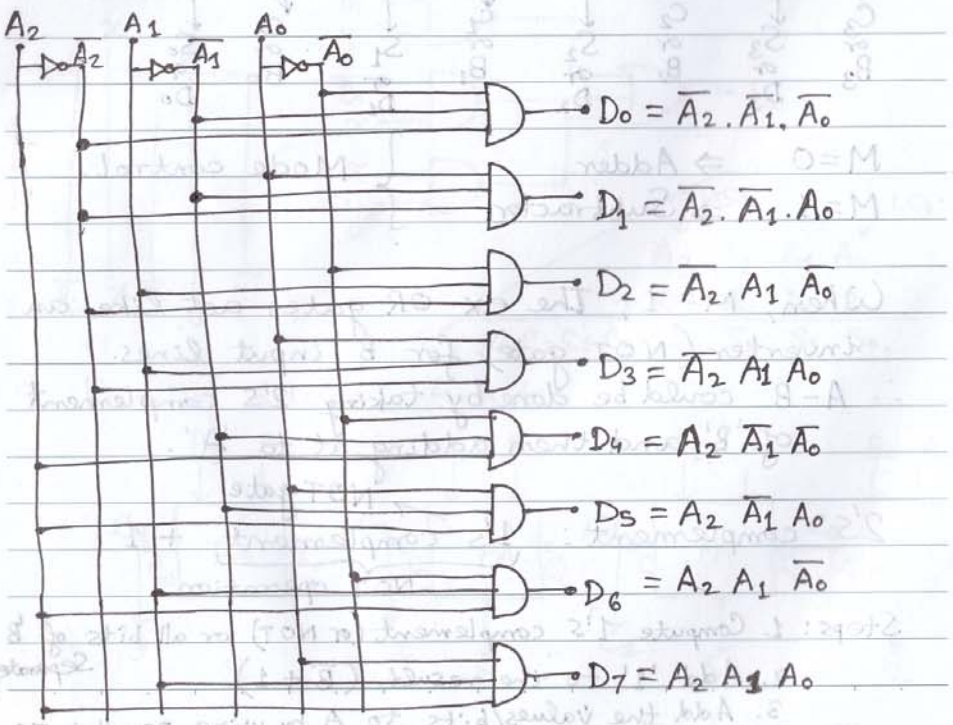
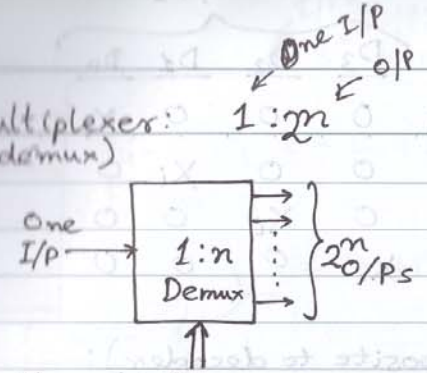
7. Decoder: 3:8 (3 I/Ps, 8 O/Ps) $n : 2^n$

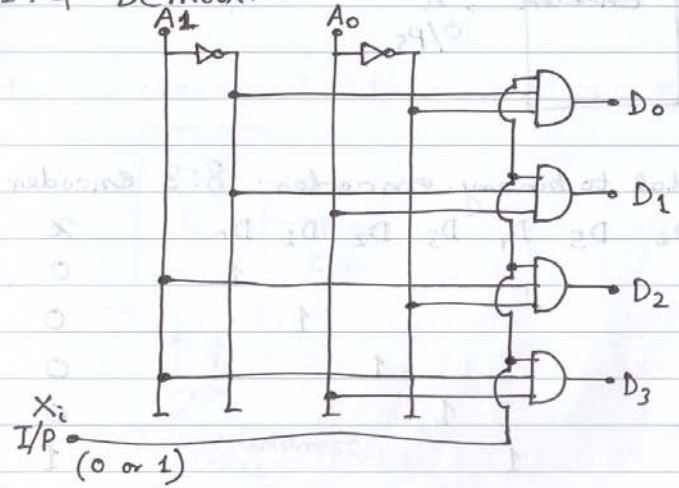
I/Ps			O/Ps							
A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



8. Demultiplexer: $1 : n$ $(n > 1)$ (demux)



1 : 4 Demux:



A_1, A_0 : Address line

X_i : I/P line

$D_3 - D_0$: Data O/P line

While, $A_1 = 0$, $A_0 = 0$ $\Rightarrow D_0 = X_i$ (I/P logic signal), $D_3 - D_1 = 0$

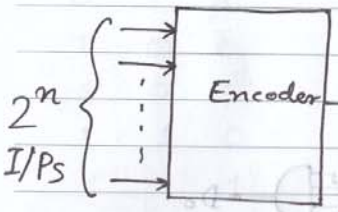
($X_i = 0 \Rightarrow D_0 = 0$, $X_i = 1 \Rightarrow D_0 = 1$)

& while, $A_1 = 1$, $A_0 = 1$ $\Rightarrow D_3 = X_i$, $D_2 - D_0 = 0$

(If $X_i = 0$, $D_3 = 0$; $X_i = 1$, $D_3 = 1$)

I/P	Select I/Ps		O/Ps			
X_i	A_1	A_0	D_3	D_2	D_1	D_0
0/1	0	0	0	0	0	X_i
0/1	0	1	0	0	X_i	0
0/1	1	0	0	X_i	0	0
0/1	1	1	X_i	0	0	0

9. Encoder (opposite to decoder):

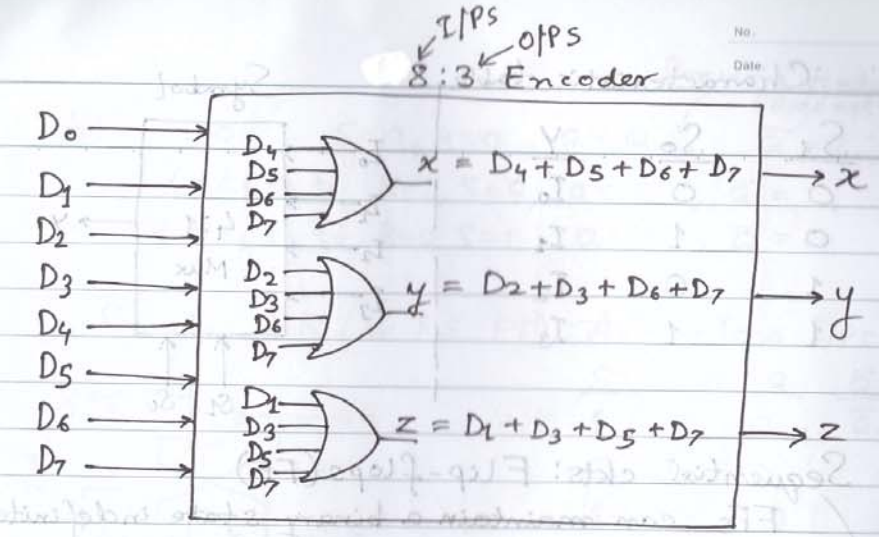


Octal to binary encoder: 8:3 encoder

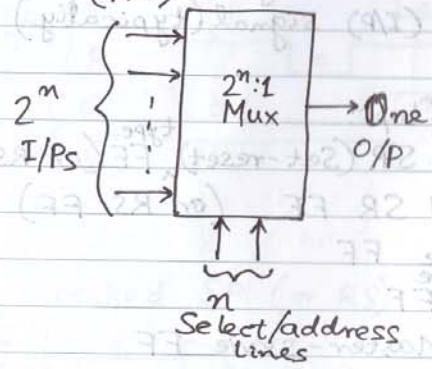
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	x	y	z
							1	0	0	0
						1		0	0	1
					1			0	1	0
				1				0	1	1
			1					1	0	0
		1						1	0	1
	1							1	1	0
1								1	1	1

Note: Only one of the I/Ps should be active at a time (mandatory requirement).

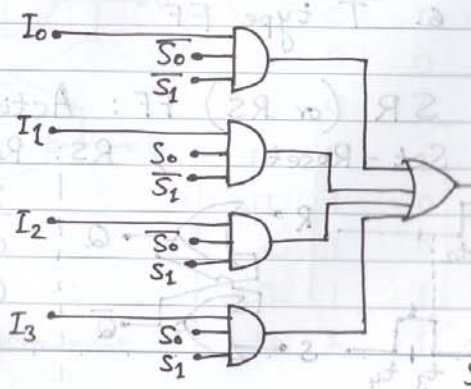
Priority encoder → Solves the prob.



10. Multiplexer (Mux): $2^n : 1$ (opp. of demux)



4:1 Mux

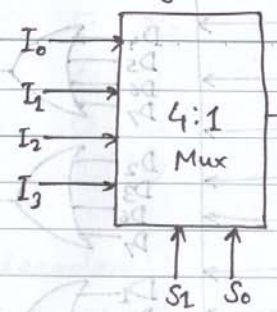


S_1 S_0 : Select lines (I/P)

Characteristic table:

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

Symbol



11. Sequential ckts: Flip-flops (FF)

FFs can maintain a binary state indefinitely until directed by an I/P signal to switch states. (Power supply must always be present)

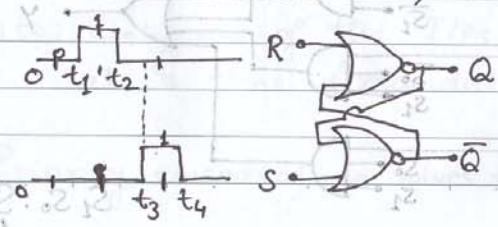
→ Ckts with clock (I/P) signal (typically)

Types of flip-flops:

1. Simple SR (Set-reset) type FF (or RS FF)
2. Clocked SR FF (or RS FF)
3. D-type FF
4. J-K type FF
5. J-K Master-slave FF
6. T type FF

12. i) Simple SR (or RS) FF: Active high I/Ps.

SR: Set-Reset ; RS: Reset-Set



S	R	Q	$\bar{Q}$
0	0	Q_{n-1}	$\bar{Q}_{n-1}$
0	1	0	1
1	0	1	0
1	1	X	X

X: Not allowed

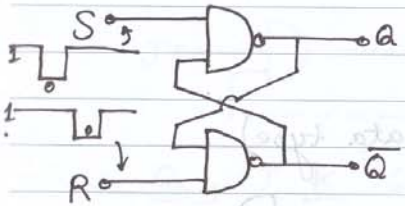
Previous state (does not change)

At $t=0$, $S=0, R=0$, $Q = Q_{n-1}$, $\bar{Q} = \bar{Q}_{n-1}$

$t = t_1$ to t_2, $R=1, S=0$, $Q=0$, $\bar{Q}=1$

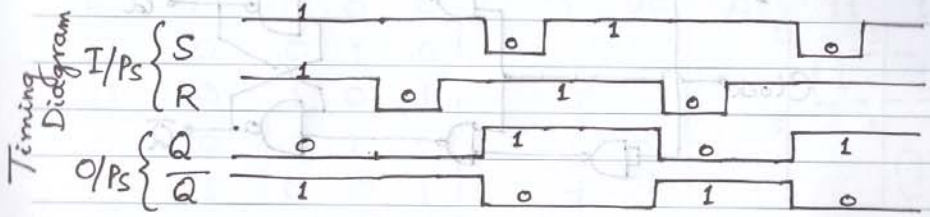
$t = t_3$ to t_4, $R=0, S=1$, $Q=1$, $\bar{Q}=0$

ii) Simple SR (or RS FF): Active low I/Ps.

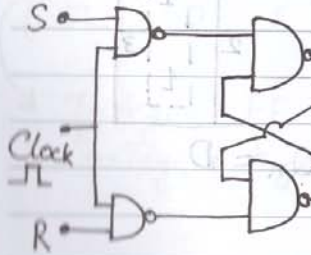


$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
1	1	Q_{n-1}	$\bar{Q}_{n-1}$
0	0	X	X
1	0	0	1
0	1	1	0

Timing Diagram

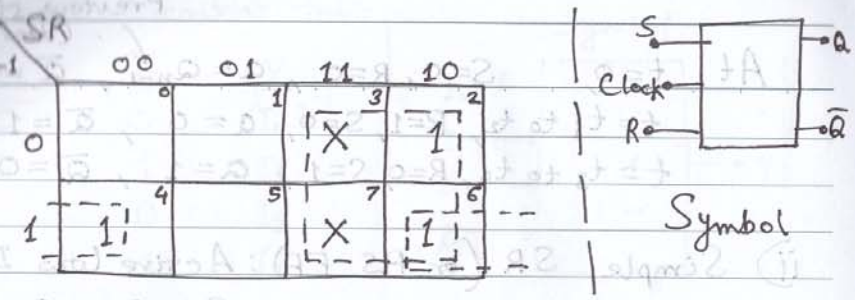


13. Clocked SR (or RS) FF:



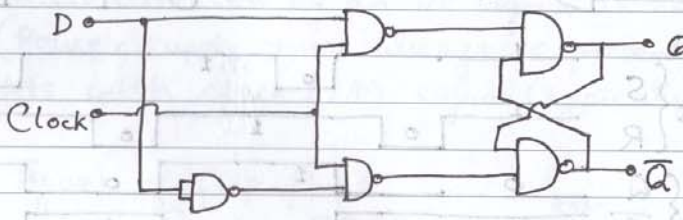
Clock	Q_{n-1} (previous o/p state)	S	R	Q_n (present o/p state)
⊓	0	0	0	0
⊓	0	0	1	0
⊓	0	1	0	1
⊓	0	1	1	X
⊓	1	0	0	1
⊓	1	0	1	0
⊓	1	1	0	1
⊓	1	1	1	X

X: Indeterminate

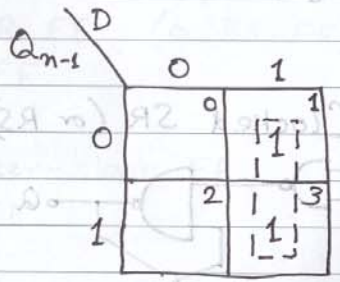


$$Q_n = S + \bar{R}.Q_{n-1}$$

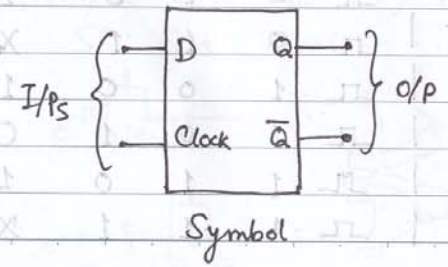
14. D-type FF (Data type)



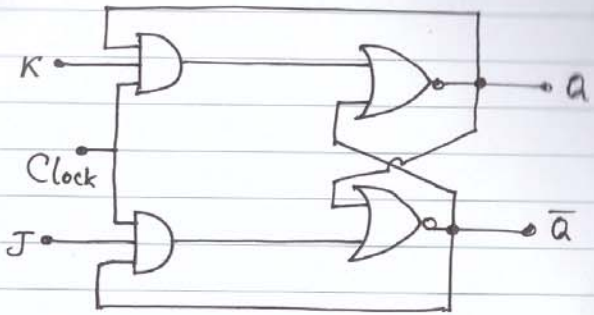
Clock	Q_{n-1}	D	Q_n
⎍	0	0	0
⎍	0	1	1
⎍	1	0	0
⎍	1	1	1



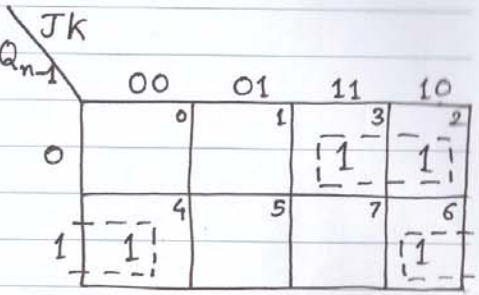
$Q_n = D$



15. J-K type FF:



Clock	Q_{n-1}	J	K	Q
⎍	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
⎍	0	1	1	1



$$Q = J.\overline{Q_{n-1}} + \bar{K}.Q_{n-1}$$

Present Logic O/P State

Previous Logic O/P state

↳ T-type FF operation
↳ Toggle type