अङ्कीय वैद्युतक मति बृहत्स्तरीय संयोजन परिकल्पना च

DIGITAL ELECTRONICS AND VLSI DESIGN

3-08-2023

DIGITAL ELECTRONICS

Logic High — 1
Logic low — 0

Logic Gates:

NOT
AND
OR
NAND
NOR
XOR
XNOR

XOR: $Y = A\bar{B} + \bar{A}B$: $A \oplus B$ → (SOP)

Truth Table:

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

XNOR: $A \odot B$

$Y = A\bar{B} + \bar{A}B$ ⇝ SOP (minterm (m))

$Y = (\bar{A}+B) \cdot (A+\bar{B})$ ⇝ POS (maxterm (M))

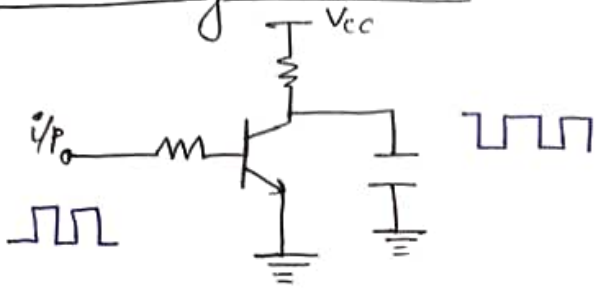
$m = \bar{M}$

de-Morgan's law:

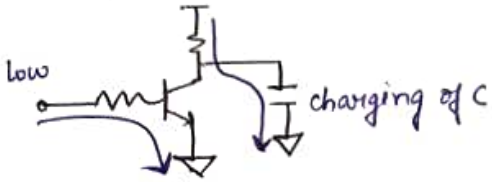
$\overline{AB} = \bar{A} + \bar{B}$

$\overline{A+B} = \bar{A} \cdot \bar{B}$

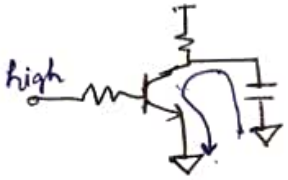
Inverter Using Transistor:



Low i/p:



High i/p:



Minimization of Logic Equations

1. Using Boolean Axioms / Theorems / Postulation
2. Karnaugh Map Method
3. Tabulation Method.

One Variable: A → 0 / 1

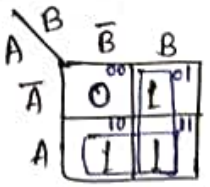
Two Variables: AB

A \ B	0	1
0	$\bar{A}\bar{B}$	$\bar{A}B$
1	$A\bar{B}$	AB

A \ B	0	1
0	0	1
1	1	0

0 1
1 0
} nothing in common

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



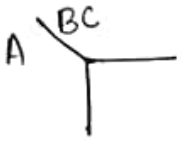
$\bar{A}B + AB$
$= B(A+\bar{A})$
$= B$

$B + A\bar{B}$ $\xrightarrow{\text{simplified form}}$ $A + B \rightarrow$ OR Gate

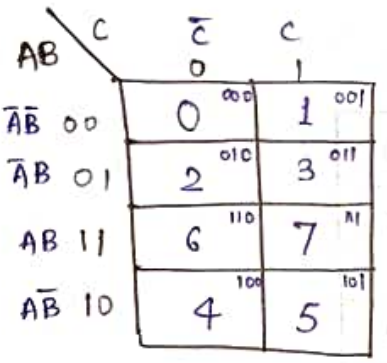
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

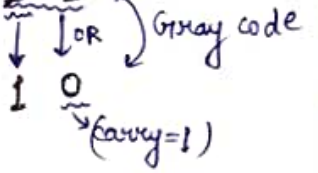
Three Variables

A, B C

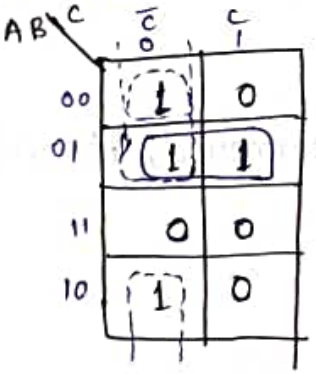


OR



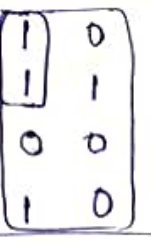


Eg. $F = \sum(0,2,3,4)$
↳ SOP
$= \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$
$\Pi \rightarrow$ POS
$= \bar{A}B + B\bar{C}$



→ Insert 1 at 0, 2, 3, 4 positions & 0 everywhere else.

Either



OR

are neighbours

4-08-2023

K-Map:

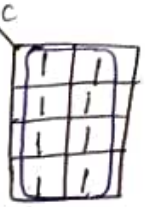
A B C

↳ We can only cornered 2^n (even) squares.

2 squares Karnaugh cornered → 1 variable gets reduced → ~~A~~ B C

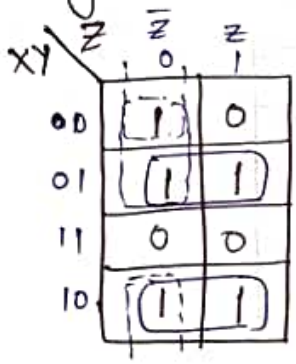
4 squares Karnaugh cornered → 2 variables reduced → ~~A~~ ~~B~~ C

8 squares Karnaugh cornered → o/p is 1 (always; irrespective of inputs) → ~~A B C~~ (1) → AB / C



Eg. $F(X,Y,Z) = \Sigma(0,2,3,4,5)$

Reduce using K-Map.



$F = X'Y'Z' + X'ZY' + X'YZ + XY'Z' + XY'Z$

$F = X'Z' + X'Y + XY'$

OR

$F = Y'Z' + X'Y' + XY'$

→ If any of the minterm is left to be cornered, it can be overlapped.

Four Variable Map

ABCD

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

→ Each square is called minterm.

$m_0 = A'B'C'D'$

No. of Squares that can be reduced:

Squares	Reduced to (Variables)
2	3 variables / literals
4	2 variables
8	1 variables
16	o/p = 1

Eg. $F(A,B,C,D) = \sum(1,3,7,11,15,12)$

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

$F = \bar{A}\bar{B}D + CD + AB\bar{C}\bar{D}$

→ 'Don't care' Condition: for undefined

In BCD code → 4 bits required → 16 variables possible → But represent only (0 to 9)

0 to 9 → output = 1

10 (to 16) → 1010 (10 in binary) → not defined

Introducing 'X' (don't care) to digital:

Logics: '0', '1', 'X'

possible values in digital

Eg. $F(A,B,C,D) = \Sigma(1,3,7,11,12,15)$

$D(A,B,C,D) = (4,13)$

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

↳ 1 in (1,3,7,11,12,15)
↳ X in (4,13)
↳ 0 elsewhere.

$F = \bar{A}\bar{B}D + CD + B\bar{C}\bar{D}$

OR

$F = \bar{A}\bar{B}D + CD + AB\bar{C}$

'Don't care' (X) can take value '0' or '1'

↓

helps to reduce no. of literals

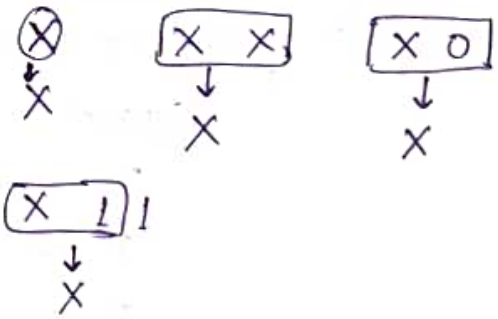
↓

should be cornered with only '1'

↓

Shouldn't be cornered alone or with itself

#



8-08-2023

5 variable Map

$F(A, B, C, D, E)$

A=0

BC \ DE	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A=1

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

Eg. $F(A,B,C,D,E) = \sum(0,2,4,6,9,13,21,23,25,29,31)$

A=0

BC \ DE	00	01	11	10
00	1			1
01	1			1
11		1		
10		1		

A=1

BC \ DE	00	01	11	10
00				
01		1	1	
11		1	1	
10		1		

(are neighbours; only 1 bit change)

$F = \bar{A}\bar{B}\bar{E} + ACE + B\bar{D}E$

Tabulation Method (Quine Mccluskey Method)

Eg. $F(A,B,C,D) = \sum(0,1,2,5,8,9,10)$

① convert the minterms into binary.
② Group them according to the no. of '1's.

	A B C D		A , B , C , D		A B C D	
0	0 0 0 0 ✓	0,1	0 0 0 _ ✓	0,1,8,9	_ 00 _	
1	0 0 0 1 ✓	0,2	0 0 _ 0 ✓	0,2 8 10	_ 0 _ 0	same
2	0 0 1 0 ✓	0,8	_ 0 0 0 ✓	0 8 2 10	_ 0 _ 0	same
8	1 0 0 0 ✓	1,5	0 _ 0 1	0 8 1 9	_ 00 _	
5	0 1 0 1 ✓	1,9	_ 0 0 1 ✓			
9	1 0 0 1 ✓	2,10	_ 0 1 0 ✓			
10	1 0 1 0 ✓	8,9	1 0 0 _ ✓			
		8,10	1 0 _ 0 ✓			

Essential one ⇓ needs to be mentioned ⇓ won't be included afterwards

$$F = \bar{A}\bar{C}D + \bar{B}\bar{C} + \bar{B}\bar{D}$$

↑ Essential prime implicant (1,5)

Q $F(A,B,C,D) = \sum(1,4,6,7,8,9,10,11,15)$.

Reduce the boolean equation using tabulation method.

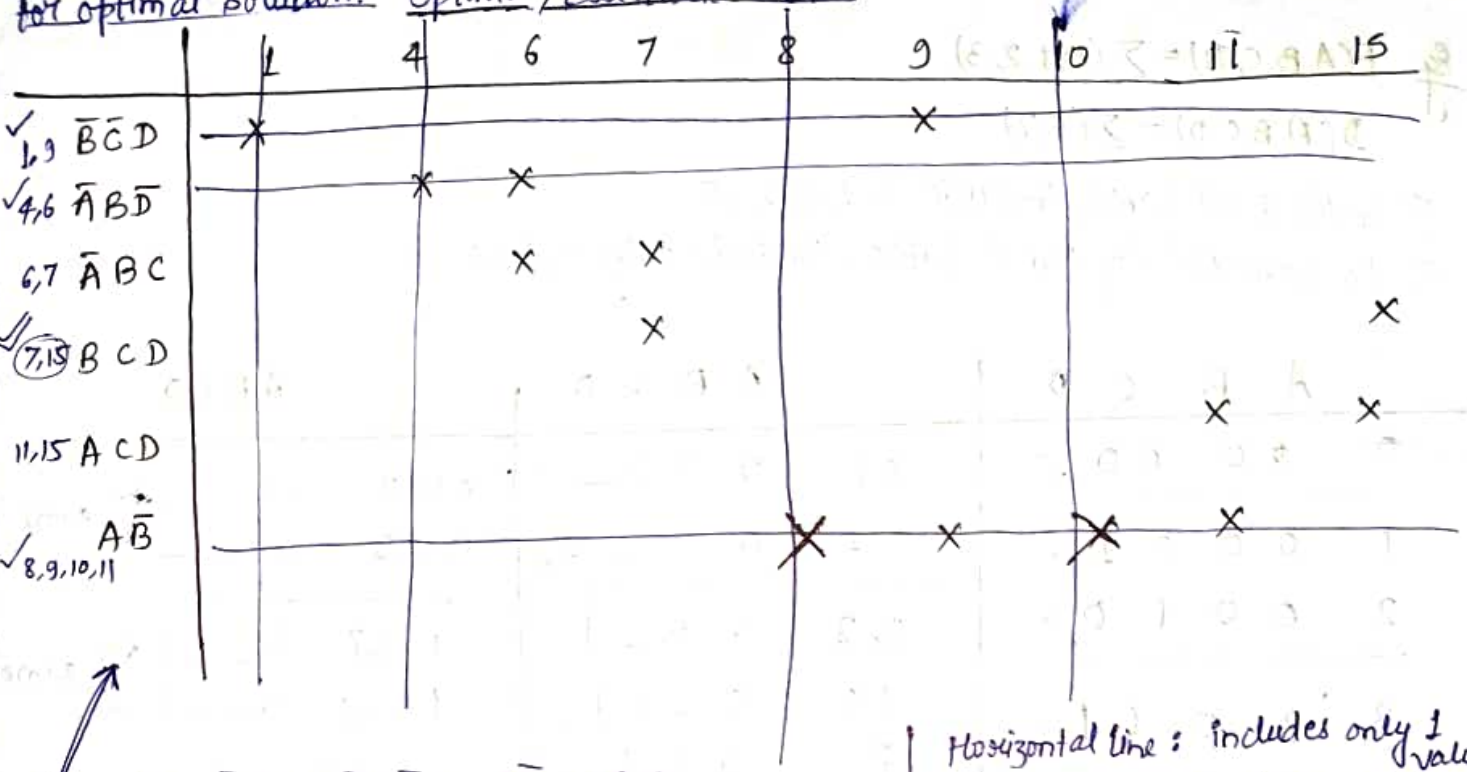
(eqn but given in numeric)

	A B C D		A B C D		A B C D	
(I)	1 0 0 0 1 ✓	1,9	_ 0 0 1	8,9,10,11	1 0 _ _	}
	4 0 1 0 0 ✓	4,6	0 1 _ 0	8,10,9,11	1 0 _ _	}
	8 1 0 0 0 ✓	8,9	1 0 0 _ ✓			
	6 0 1 1 0 ✓	8,10	1 0 _ 0 ✓			
(II)	9 1 0 0 1 ✓	6,7	0 1 1 _			
	10 1 0 1 0 ✓	9,11	1 0 _ 1 ✓			
	7 0 1 1 1 ✓	10,11	1 0 1 _ ✓			
(III)	11 1 0 1 1 ✓	7,15	_ 1 1 1			
(IV)	15 1 1 1 1 ✓	11,15	1 _ 1 1			

$$F = \underset{(1,9)}{\bar{B}\bar{C}D} + \underset{(4,6)}{\bar{A}B\bar{D}} + \underset{(6,7)}{\bar{A}BC} + \underset{(7,15)}{BCD} + \underset{(11,15)}{ACD} + \underset{(8,9,10,11)}{A\bar{B}}$$

include 'Don't Care' here

for optimal solution: Optimal/Essential Table



$F = \bar{B}\bar{C}D + \bar{A}B\bar{D} + A\bar{B} + BCD$

don't include 'Don't Care' here

to get (7,15) included as none of 7 or 15 are there

Horizontal line: includes only 1 value.

Vertical line: includes pair not already included
↳ e.g. (6,7) but not (7,15)

10-08-2023

Eg. $F(A,B,C,D) = \sum(0,1,2,3)$

$D(A,B,C,D) = \sum(5,7)$

① In the first table, include 0, 1, 2, 3, 5, 7

② In Essential/optimal table, include only 0, 1, 2, 3.

	A	B	C	D	
0	0	0	0	0	✓
1	0	0	0	1	✓
2	0	0	1	0	✓
3	0	0	1	1	✓
5	0	1	0	1	✓
7	0	1	1	1	✓

	A	B	C	D	
0,1	0	0	0	_	✓
0,2	0	0	_	0	✓
1,3	0	0	_	1	✓
1,5	0	_	0	1	✓
2,3	0	0	1	_	✓
3,7	0	_	1	1	✓
5,7	0	1	_	1	✓

	A	B	C	D	
0,1,2,3	0	0	_	_	same
0,2,1,3	0	0	_	_	
1,3,5,7	0	_	_	1	same
1,5,3,7	0	_	_	1	

$F = \bar{A}\bar{B} + \bar{A}D$

Optimal Solution : ('Don't care' won't be considered here)

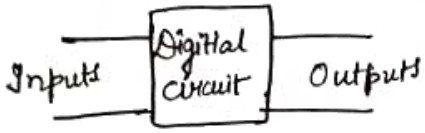
		0	1	2	3
(0,1,2,3) ✓	$\bar{A}\bar{B}$	×	×	×	×
(1,3,5,7)	$\bar{A}D$		×		×

$\therefore F = \bar{A}\bar{B}$

DIGITAL DESIGN

① Combinational Circuits
② Sequential Circuits

Combinational circuits: Output depends only on the present input.



Eg. Half Adder ⇝ Adding without carry

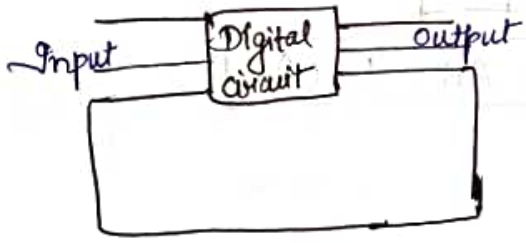
A + B

A	B	S (Sum)	C (Carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

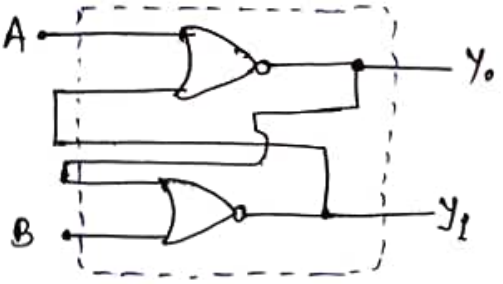
$C = AB$

$S = A \oplus B$

Sequential circuits: Output depends not only on the present input but also on the past output.



Eg.



Eg. Full Adder ⇝ Adding with carry

	A	B	C	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Sum:

(S)

AB \ C	0	1
00	0	1
01	1	0
11	0	1
10	1	0

$S = A \oplus B \oplus C$

Carry:

(C)

AB \ C	0	1
00	0	0
01	0	1
11	1	1
10	0	1

$C = AB + BC + AC$

$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

$= (\bar{A}\bar{B} + AB)C + (\bar{A}B + A\bar{B})\bar{C}$

$= (\overline{\bar{A}B + A\bar{B}})C + (\bar{A}B + A\bar{B})\bar{C}$

$= (\overline{A \oplus B})C + (A \oplus B)\bar{C}$

$= A \oplus B \oplus C$

Eg Use Full Adder circuit to add 2 four-bits numbers.

$A = \overset{MSB}{A_3}\ A_2\ A_1\ \overset{LSB}{A_0}$

$B = B_3\ B_2\ B_1\ B_0$

$C_2\ C_1\ C_0$ ← carries

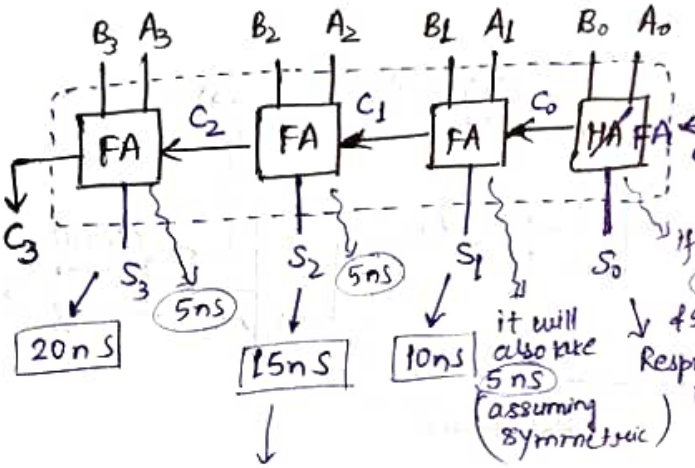
$A = A_3\ A_2\ A_1\ A_0$

$B = B_3\ B_2\ B_1\ B_0$

FA FA FA HA

$C_3\ S_3\ S_2\ S_1\ S_0$

2 HA circuits = 1 FA circuit



→ 4 bit Parallel Adder (Ripple Carry Adder)

∴ in 10ns, carry will be available for it.

→ For fast result, increase carry part efficiency.

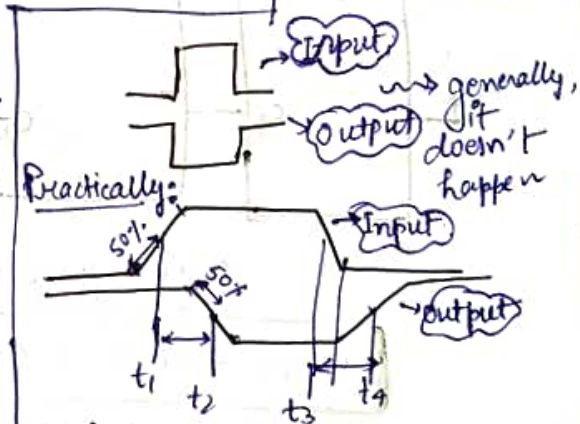
→ **Propagation Delay:**

↳ Time taken by circuit to acknowledge the change in state.

OR

finite time required to respond the change in state.

- Inverter is BJT or MOSFET
  - ↳ takes finite time to change from cutoff to saturation



$PD = t_2 - t_1$

propagation delay

2 PDs ↔ high to low, low to high

↔ Normally, we take average.

$$T_{pd} = \frac{t_{PD_{HL}} + t_{PD_{LH}}}{2}$$

11-08-2023

→ ABC

Full Adder: $S = A \oplus B \oplus C$

$C = AB + BC + AC$

AB \ C	0	1
00	0	0
01	0	1
11	1	1
10	0	1

$C = AB + AB'C + A'BC$

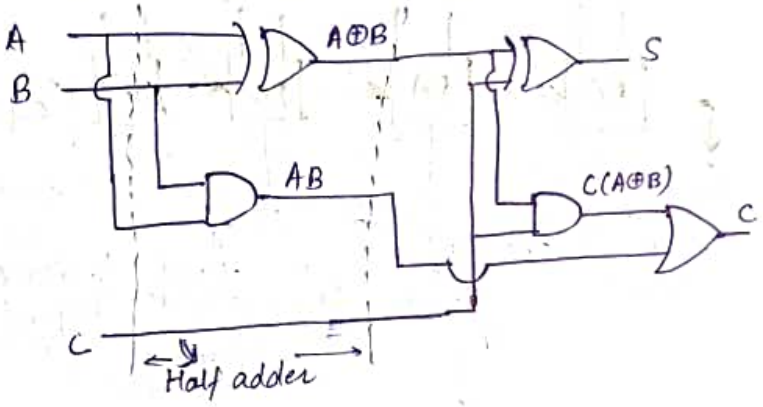
$= AB + C(AB' + A'B)$

$\Rightarrow C = AB + C(A \oplus B)$

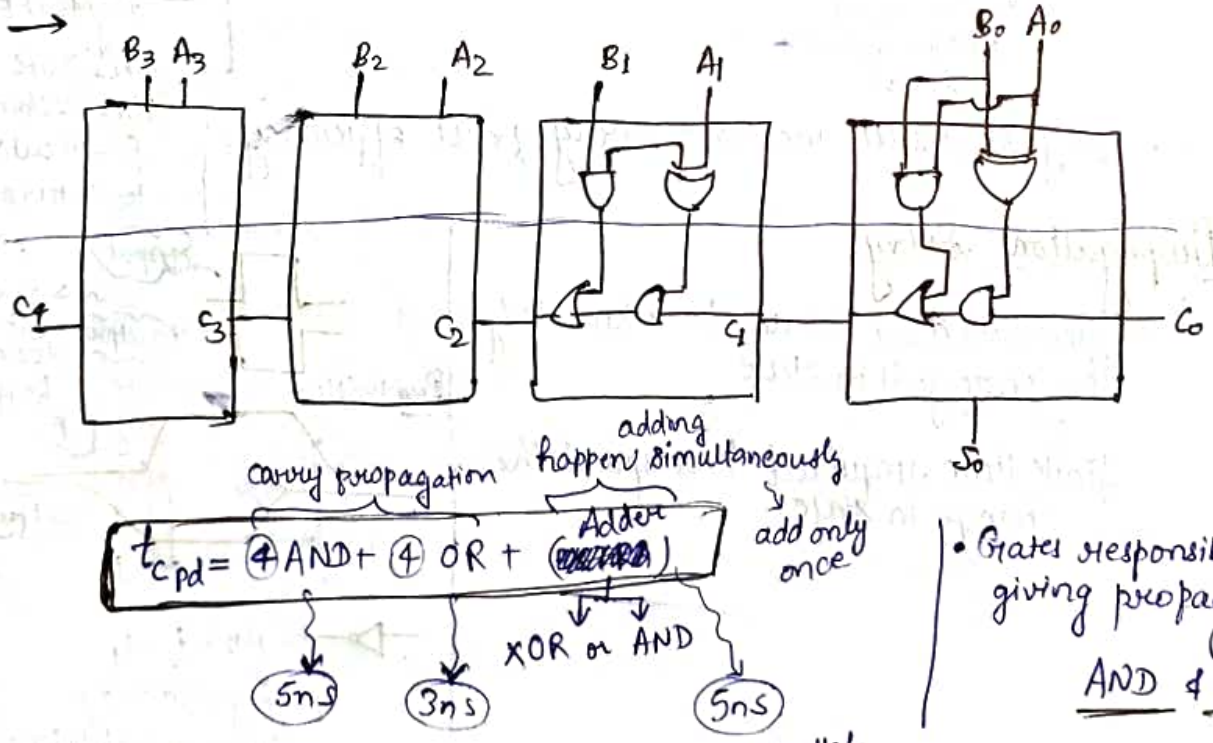
Half Adder:

$S = A \oplus B$

$C = AB$



→



carry propagation

adding happen simultaneously

add only once

t_{cpd} = ④ AND + ④ OR + (Adder)

XOR or AND

5ns 3ns 5ns

- Gates responsible for giving propagation delay: AND & OR

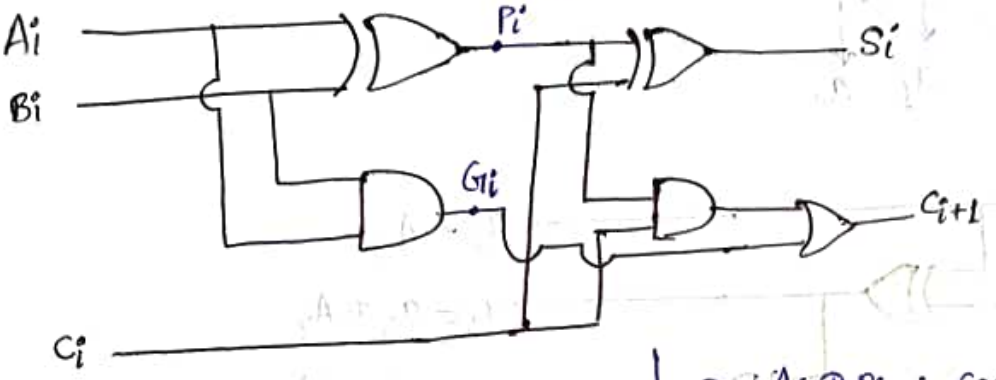
Propagation delay for the 8-bit parallel adder

$= 8 \times 5ns + 8 \times 3ns + 5ns$

$= 69ns$

Carry Lookahead Adder

↳ Aim: To reduce the propagation delay
  ↳ Area will increase.
  ↳ All the carries are generated simultaneously.



$P_i = A_i \oplus B_i$: carry propagator
$G_i = A_i B_i$: carry generator

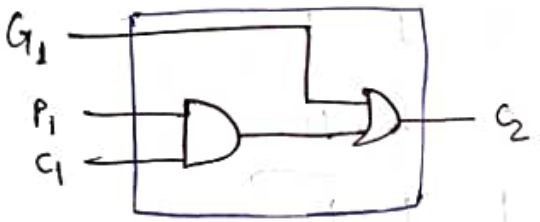
$S_i = P_i \oplus C_i$

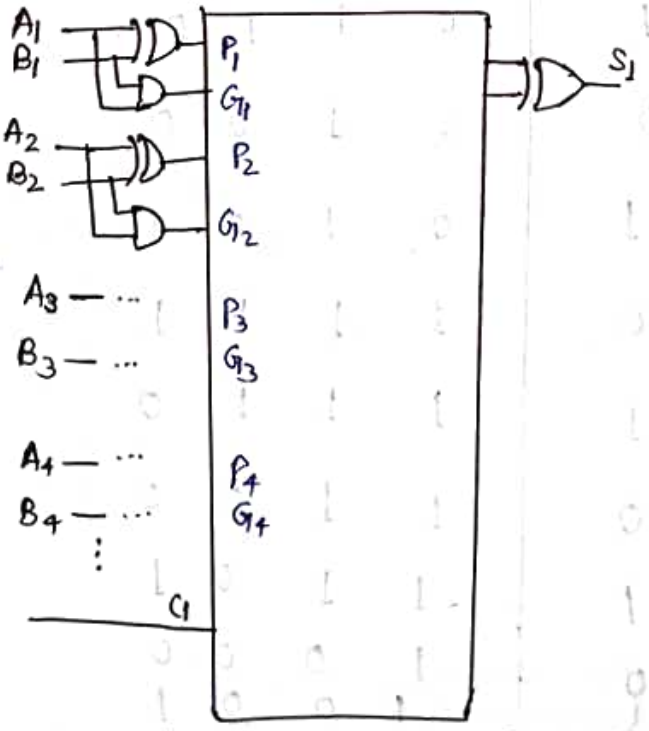
$C_{i+1} = G_i + P_i C_i$

$C_2 = G_1 + P_1 C_1$

$C_3 = G_2 + P_2 C_2$
$= G_2 + P_2 (G_1 + P_1 C_1)$

$C_4 = G_3 + P_3 C_3$
$= G_3 + P_3 (G_2 + P_2 (G_1 + P_1 C_1))$





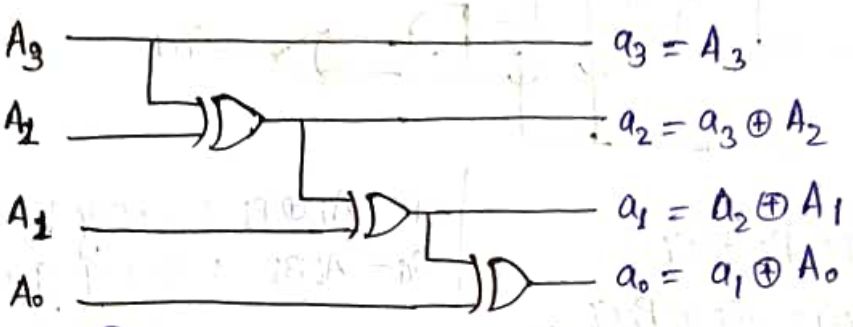
Q. Design a 4-bit gray to binary code converter and obtain the reduced boolean equation using K-Map.

Soln:

A_3 A_2 A_1 A_0

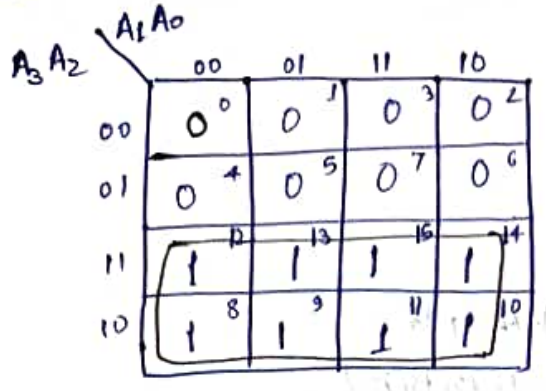
↓ ↗↓ ↗ ↓ ↗ ↓

a_3 a_2 a_1 a_0



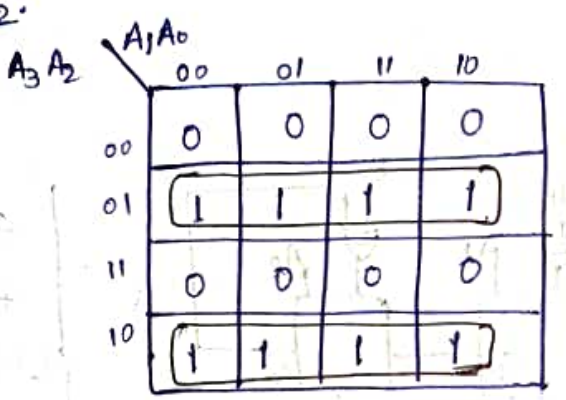
Gray				Binary			
A_3	A_2	A_1	A_0	a_3	a_2	a_1	a_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

For a_3 :



$a_3 = A_3$

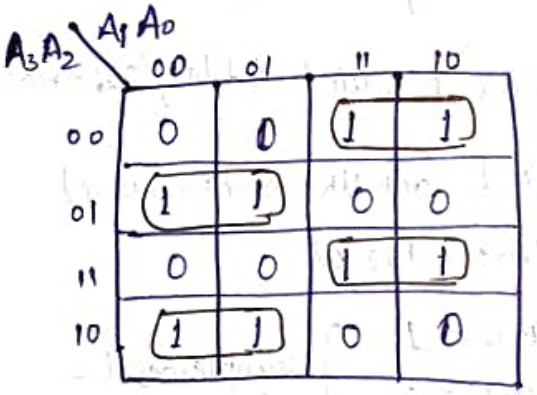
For a_2 :



$a_2 = \bar{A}_3 A_2 + A_3 \bar{A}_2$

$= A_2 \oplus A_3$

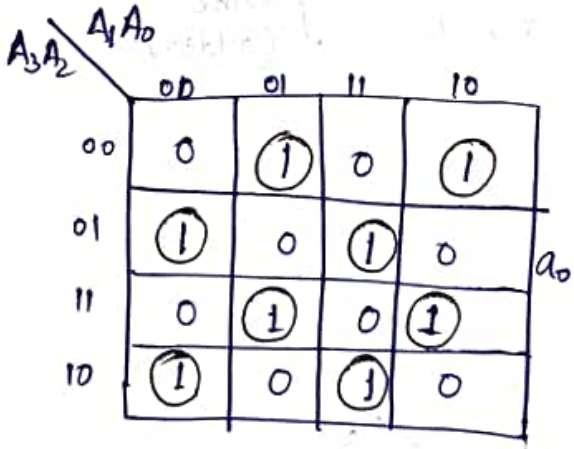
For a_1 :



$a_1 = \bar{A}_3 \bar{A}_2 A_1 + \bar{A}_3 A_2 \bar{A}_1 + A_3 A_2 A_1 + A_3 \bar{A}_2 \bar{A}_1$

$= A_1 \oplus A_2 \oplus A_3$

For a_0 :



$a_0 = \bar{A}_3 \bar{A}_2 \bar{A}_1 A_0 + \bar{A}_3 \bar{A}_2 A_1 \bar{A}_0 + \bar{A}_3 A_2 \bar{A}_1 \bar{A}_0 + \bar{A}_3 A_2 A_1 A_0 + A_3 A_2 \bar{A}_1 A_0 + A_3 A_2 A_1 \bar{A}_0 + A_3 \bar{A}_2 \bar{A}_1 \bar{A}_0 + A_3 \bar{A}_2 A_1 A_0$

$= A_3 \oplus A_2 \oplus A_1 \oplus A_0$

17-08-2023

Carry look Ahead Adder

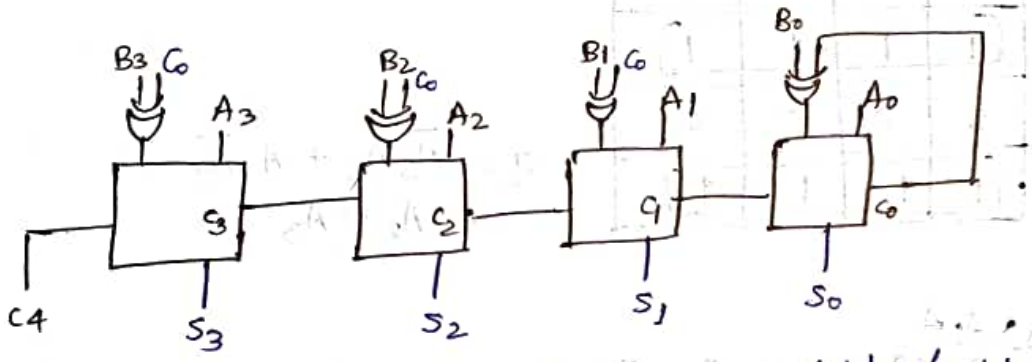
$$\begin{array}{r} +A \\ +B \\ \hline S = +A+B \end{array}$$

$$\begin{array}{r} +A \\ -B \\ \hline S = \dots \end{array}$$

A (4-bit) + 2's compl. B

$A_3 A_2 A_1 A_0$

2's com. $(B_3 B_2 B_1 B_0)$



4 → (100) 3 bits required

+4 → 4 bits required

(0100)

↓

std. no. of bits required

↳ 4 bit parallel Adder/Subtractor

→ If $C_0 = 1$, $B_0 \rightarrow$ complemented (act like subtractor)
If $C_0 = 0$, $B_0 \rightarrow B_0$ (act like adder)

→ If $C_0 = 1$, if $B_0 = 0$, $B_0 \oplus C_0 = 1$
if $B_0 = 1$, $B_0 \oplus C_0 = 0$
} complement (subtractor)

If $C_0 = 0$, if $B_0 = 0$, $B_0 \oplus C_0 = 0$
if $B_0 = 1$, $B_0 \oplus C_0 = 1$.
} same (adder)

Parity Generator / Parity Checker Circuit

↳ Due to noise in the system, some bits may be flipped.

$\underline{1}00 \rightarrow \underline{0}00$

↳ So, we send along with the original signal, an error checking signal; parity checker to check for any error in communication/transmission.

Parity → Even parity: No. of '1' in the signal is even.
→ Odd parity: No. of '1' in the signal is odd.

Parity checker → Checks whether the no. of '1' is even or not.
↳ o/p = 1, if even
↳ o/p = 0, if odd

A B C P → parity

1	0	0	1

	A	B	C	P	
0	0	0	0	0	⇝ no. of 1 = even (o/p = 0)
1	0	0	1	1	⇝ no. of 1 = odd (o/p = 1)
2	0	1	0	1	
3	0	1	1	0	
4	1	0	0	1	
5	1	0	1	0	
6	1	1	0	0	
7	1	1	1	1	

↳ we add '1' more
↳ we get even no. of '1' (0 0 1 1)

XOR → odd fn (for more than 2 variables)

$P(A,B,C) = \Sigma(1,2,4,7)$

Magnitude Comparator

↳ Compare if $A = B$ or $A < B$ or $A > B$.

Logic ⟶ if $(A_3 = B_3)$ & $(A_2 = B_2)$ & $(A_1 = B_1)$ & $(A_0 = B_0)$

For A = B

A $A_3\ A_2\ A_1\ A_0$

B $B_3\ B_2\ B_1\ B_0$

$Y_3\ Y_2\ Y_1\ Y_0$ → AND Gate

$Y_3 \& Y_2 \& Y_1 \& Y_0$

To check if $A_3 = B_3$, use XNOR Gate:

$$Y = AB + \bar{A}\bar{B}$$

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

For A > B

A : A_3 A_2 A_1 A_0

B : B_3 B_2 B_1 B_0

$$Y_i = A_i B_i + \overline{A_i B_i}$$

$$Y(A>B) = A_3\bar{B_3} + Y_3 A_2 \bar{B_2} + Y_3 Y_2 A_1 \bar{B_1} + Y_3 Y_2 Y_1 \overline{A_0 B_0}$$

For A < B

A : A_3 A_2 A_1 A_0

B : B_3 B_2 B_1 B_0

$$Y(A<B) = \bar{A_3} B_3 + Y_3 \bar{A_2} B_2 + Y_3 Y_2 \bar{A_1} B_1 + Y_3 Y_2 Y_1 \bar{A_0} B_0$$

To minimise the area (time doesn't matter) ⇝ Iteration method.

↳ check $A_3 > B_3$ → store the result

↳ check $A_2 > B_2$ → store the result

↳

⋮

} Take AND of all the results.

22-08-2023

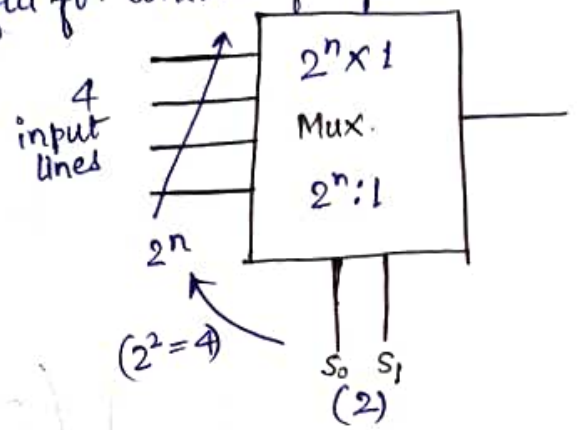
COMBINATIONAL CIRCUITS

Multiplexer

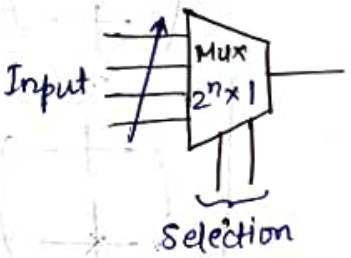
↳ Digital circuit used to ~~transmit~~ transfer multiple signals.

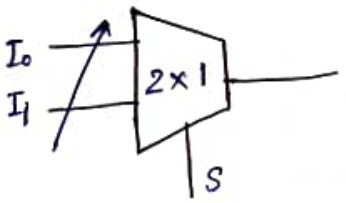
↳ Accepts ~~many~~ bunch of signals but outputs only one signal based on the selection lines.

↳ Useful for control purposes.



S_0	S_1	Input line taken
0	0	1st
0	1	2nd
1	0	3rd
1	1	4th





$F = I_0 \bar{S} + I_1 S$

$S = 0 \cdot I_0$

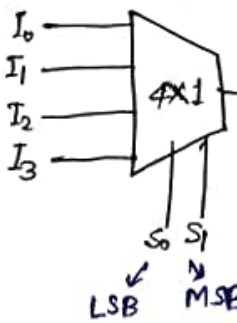
$S = 1 \cdot I_1$

2×1

$2^n = 2^1$ — 1 output

1 selection line — 2 inputs

Eg

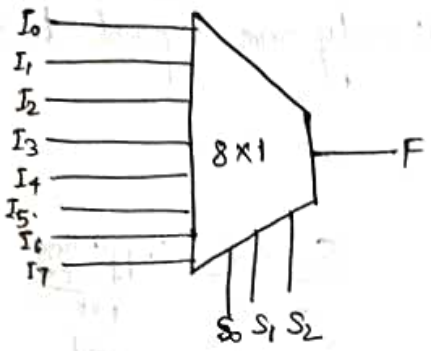


$F = \bar{S_0}\bar{S_1} I_0 + \bar{S_1} S_0 I_1 + S_1 \bar{S_0} I_2 + S_1 S_0 I_3$

S_1 S_0	F	
0 0	I_0	$\bar{S_1}\bar{S_0} I_0$
0 1	I_1	$\bar{S_1} S_0 I_1$
1 0	I_2	$S_1 \bar{S_0} I_2$
1 1	I_3	$S_1 S_0 I_3$

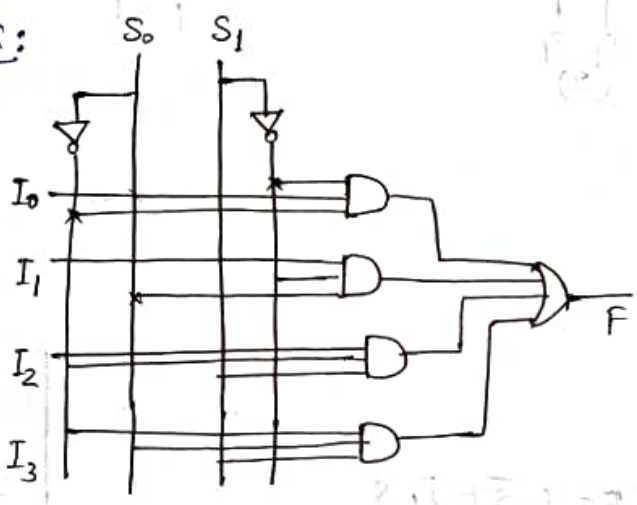
→ 8×1 circuit: $F = \bar{S}_2 \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_2 \bar{S}_1 S_0 I_1 + \bar{S}_2 S_1 \bar{S}_0 I_2 + \bar{S}_2 S_1 S_0 I_3$

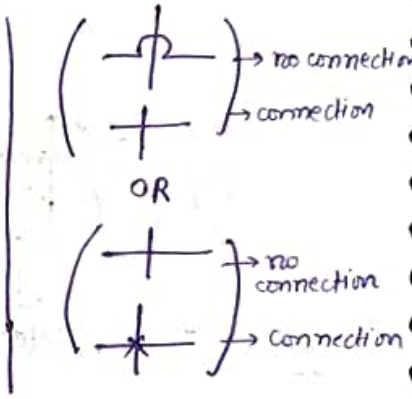
$+ S_2 \bar{S}_1 \bar{S}_0 I_4 + S_2 \bar{S}_1 S_0 I_5 + S_2 S_1 \bar{S}_0 I_6 + S_2 S_1 S_0 I_7$



→ Logic Circuit:

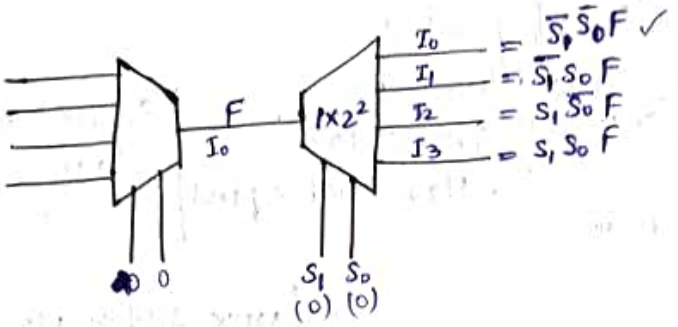
4×1 MUX:

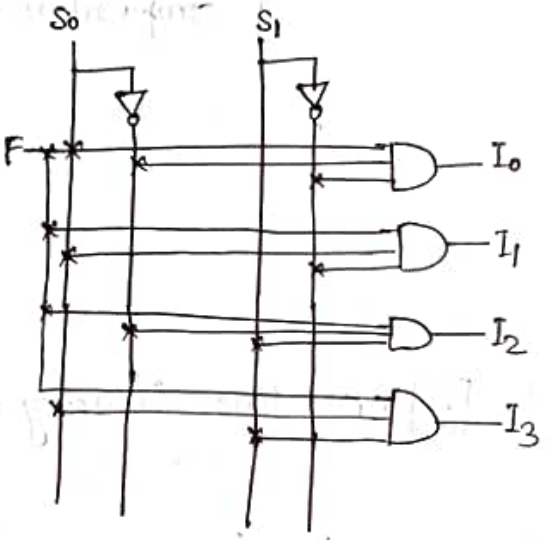




~~DeMux~~

DeMUX



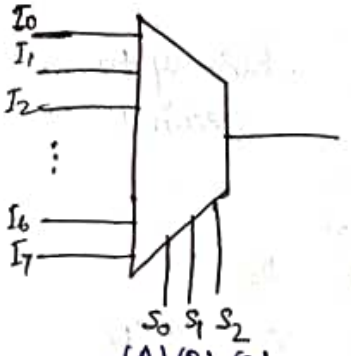


→ To have SOP (we have only AND Gate here), we can connect OR Gate externally.

→ MUX can act as a Full Adder.
Selection lines → inputs.

Full adder : A, B, C → S, C

MUX:



0 0 0 → I_0 gets 5V
0 0 1 → I_1 gets 5V.
⋮
1 1 1 → I_7 gets 5V.

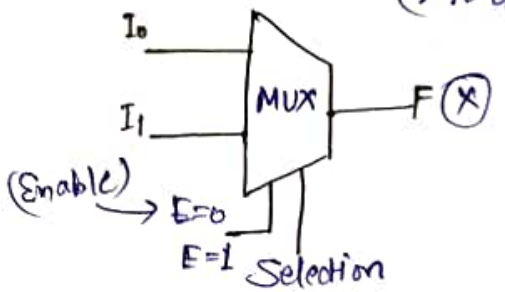
Decoder

MSI → ~~Middle~~ Medium Scale Integration

↳ Transistor level is more than 10 (> 10 Gates / Transistors)

↳ Have enable pins

Enabled signal
↳ Its output is considered



MUX, Adder, Magnitude comparator are MSI

$F = \bar{S} I_0 + S I_1$

$F = E\bar{S} I_0 + E S I_1$

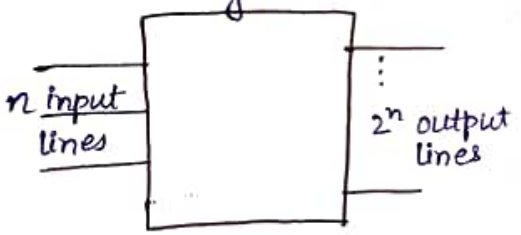
$F = 0$

24-08-2023

Decoder

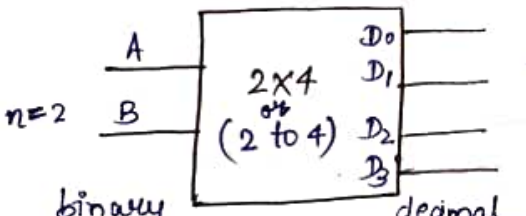
↳ Coder used to code/decode n input lines (binary information) to 2^n output lines.

Binary to decimal



2^n distinct output lines (but not necessary).

eg.



Binary to decimal

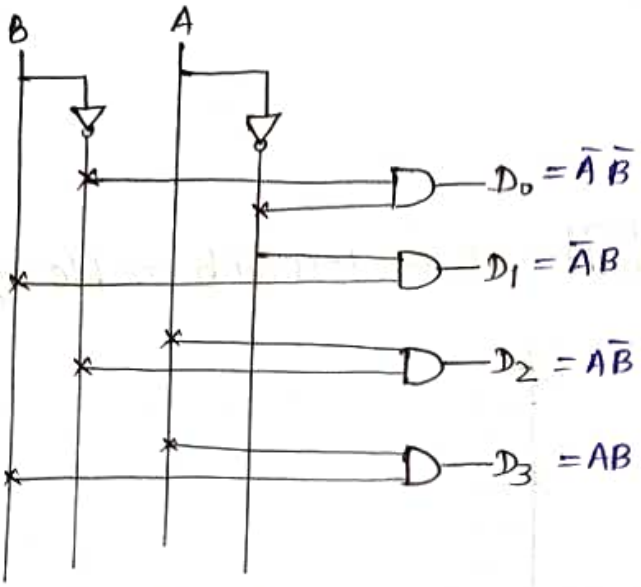
binary A	B	decimal D_0	D_1	D_2	D_3	
0	0	1	0	0	0	(0)
0	1	0	1	0	0	(1)
1	0	0	0	1	0	(2)
1	1	0	0	0	1	(3)

$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

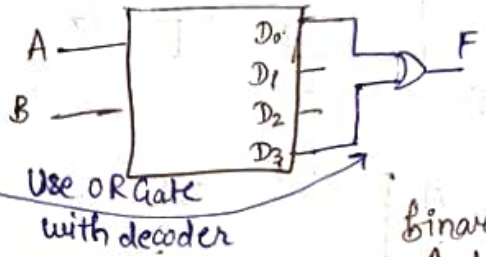
$D_2 = A\bar{B}$

$D_3 = AB$



Eg. $F(A, B) = \Sigma(0, 3)$

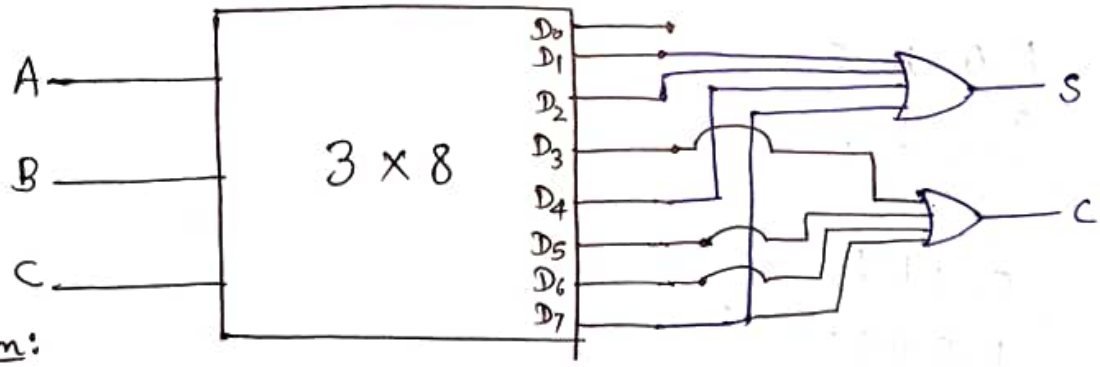
$F = \bar{A}\bar{B} + AB$



Q) Design a Full Adder circuit using 3 to 8 ^ binary decoder and necessary gates.

Soln: $S(A, B, C) = \sum(1, 2, 4, 7)$

$C(A, B, C) = \Sigma(3, 5, 6, 7)$



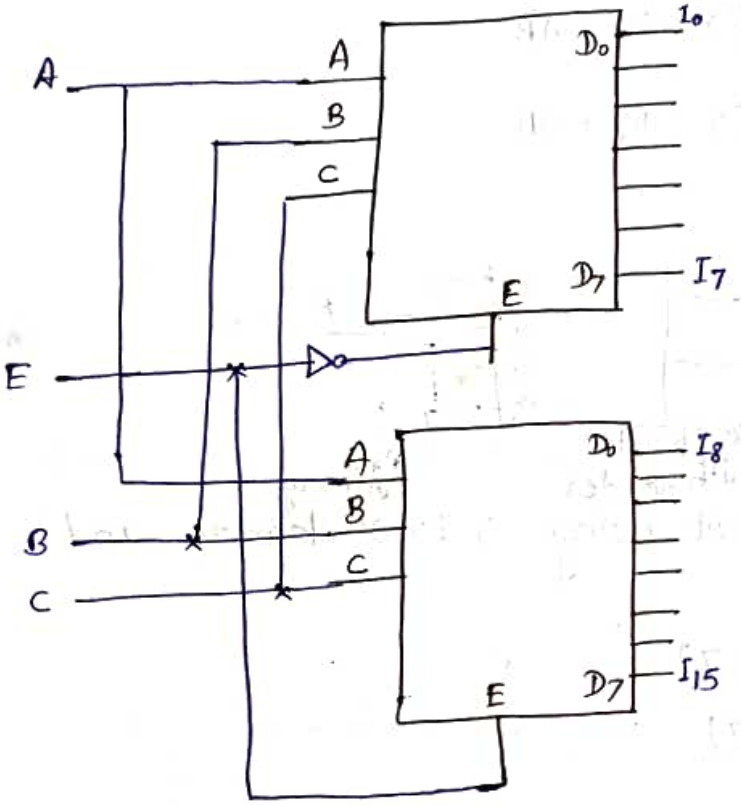
Sum:

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Decoder with Enable Signal

Q) Construct a 4×16 decoder using 3×8 decoders with enable signal.

Soln:



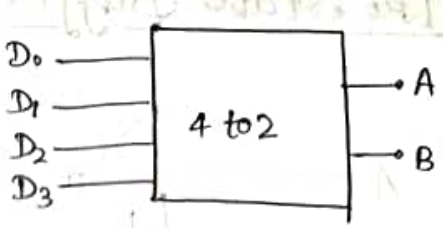
E	A	B	C	
0	0	0	0	
0	0	0	1	
⋮				
0	1	1	1	
1	0	0	0	
1	0	0	1	
⋮				

→ To construct 4×16 decoder using 2×4 decoders, we need 4 (2×4) decoders.

We can't use inverter (E) as with 3×8 decoders

Encoder

4 to 2 Encoder



D_0	D_1	D_2	D_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

↳ 2 values (1000 & 0000) are giving the same code (00)
↳ must be avoided.

↳ Ambiguity
↳ If input is (0110) it will give (11), but we wanted (01) or (10).

$A = D_2 + D_3$

$B = D_1 + D_3$

↳ ~~Priority Encoder~~

D_0	D_1	D_2	D_3	A	B	V
0	0	0	0	x	x	0
1	x	x	x	0	0	1
0	1	x	x	0	1	1
0	0	1	x	1	0	1
0	0	0	1	1	1	1

Varification ⇝ determines if we should consider the info
↳ consider, if V = 1 & not, if V = 0.

More than one info shouldn't be '1'

Don't care

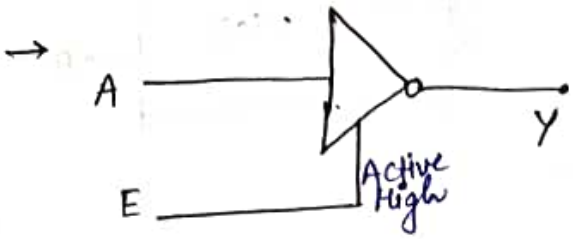
eg., parity checker

↳ V = 0, if $D_0D_1D_2D_3 = 0000$
= 1, else
↳ we should atleast consider only if one of info (D_0, D_1, D_2, D_3) is 1.

↳ Priority Encoder
↳ Giving priority to MSB or LSB.

31-08-2023

Tri State Buffer / Inverter

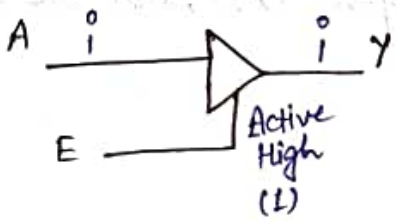


→ Inverting Buffer

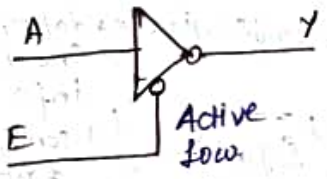
Z → High Impedence State
↳ floating
↳ no connection in output section

A	E	Y
X	0	'Z'
0	1	1
1	1	0

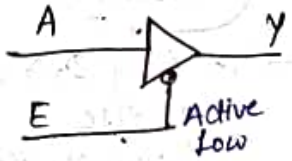
Active High : E=1
Active Low : E=0



→ Non-inverting Buffer
(A=Y)

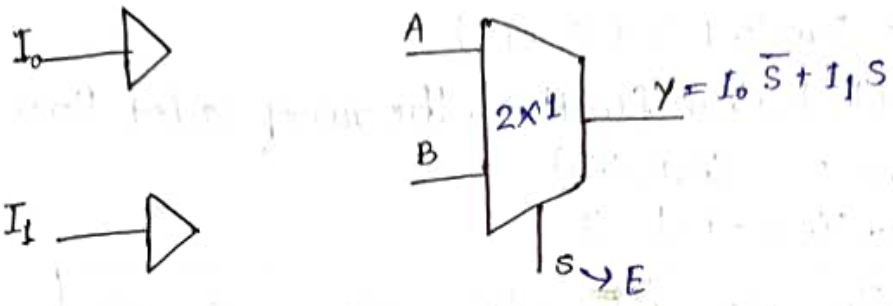


A	E	Y
X	1	High 'Z'
0	0	1
1	0	0

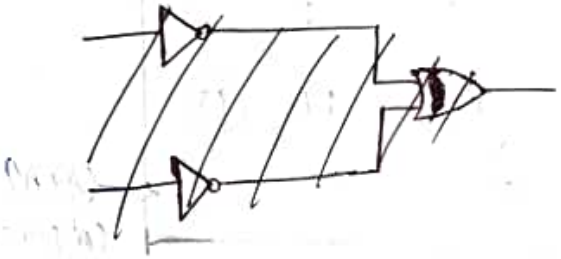


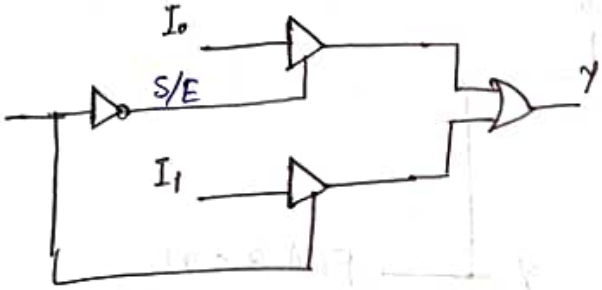
Q) Use tristate buffer to make 2×1 Mux.

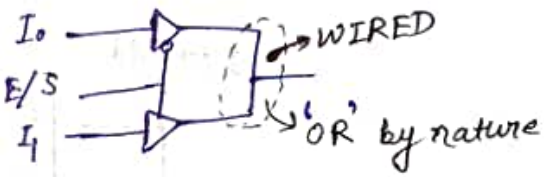
Soln:



S	Y
0	I_0
1	I_1

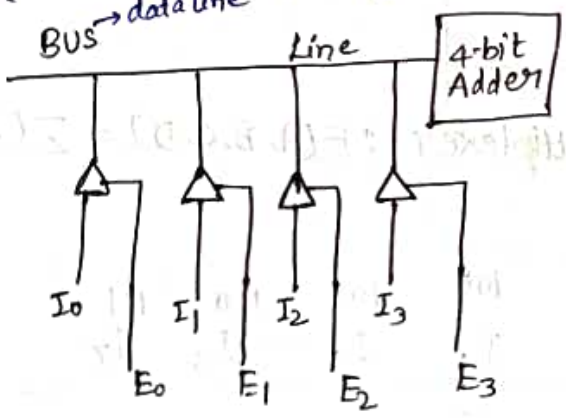






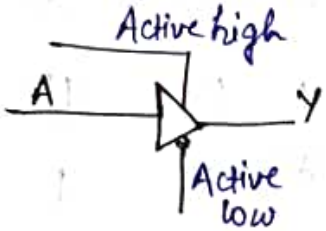
Application:
(4-bit adder of only one input line)

Decoder:



A B	E_0	E_1	E_2	E_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1

→ Buffer can also be like (Bidirectional switch)



1-09-2023

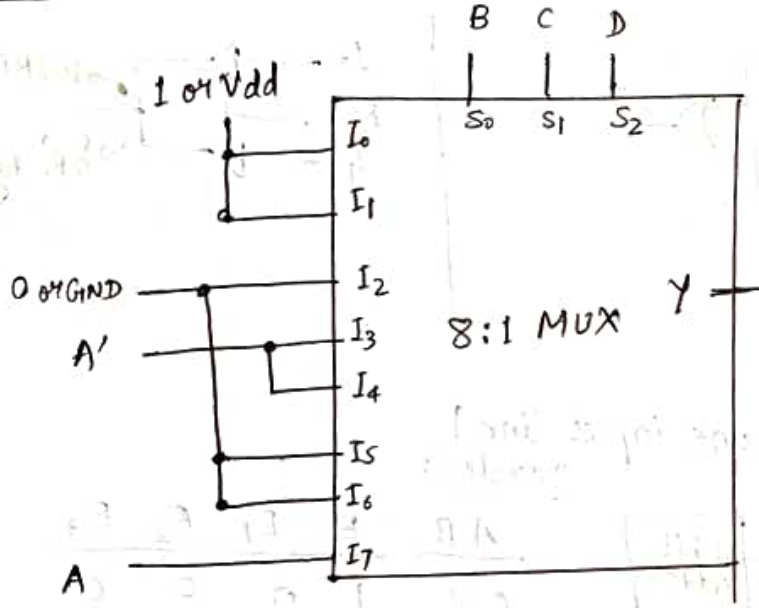
Combinational Circuit Design using MUX/DECODER

Eg. $F(A,B,C,D) = \sum m(0,1,3,4,8,9,15)$

- For n variable Boolean function, the no. of select lines would be (n-1).
- Variables, n = 4 (A, B, C, D)
- No. of select lines = n-1 = 3

	$B'C'D'$	$B'C'D$	$B'CD'$	$B'CD$	$BC'D'$	$BC'D$	BCD'	BCD
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	A'	A'	0	0	A

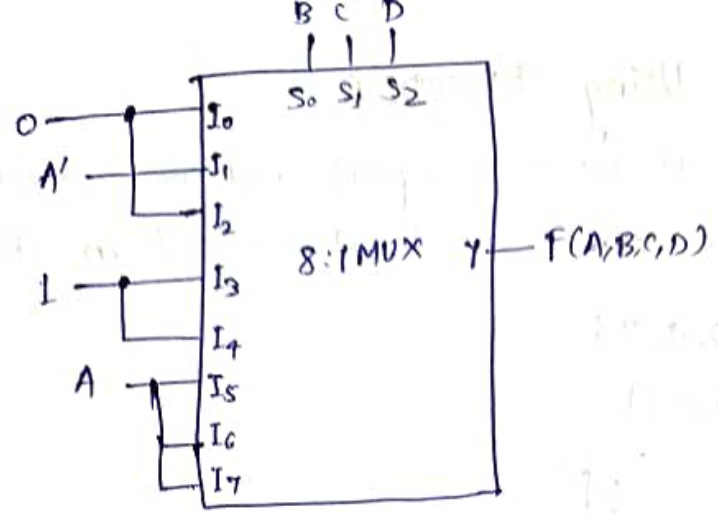
$(A+A')$ (A' for circled)



Eg. 4-input function with a multiplexer : $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$.

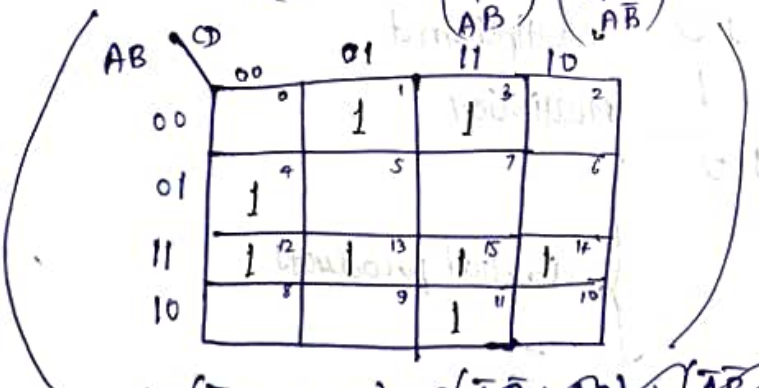
Soln:

	000 I_0	001 I_1	010 I_2	011 I_3	100 I_4	101 I_5	110 I_6	111 I_7
A'	0	(1)	2	(3)	(4)	5	6	7
A	8	9	10	(11)	(12)	(13)	(14)	(15)
	0	A'	0	1	1	A	A	A

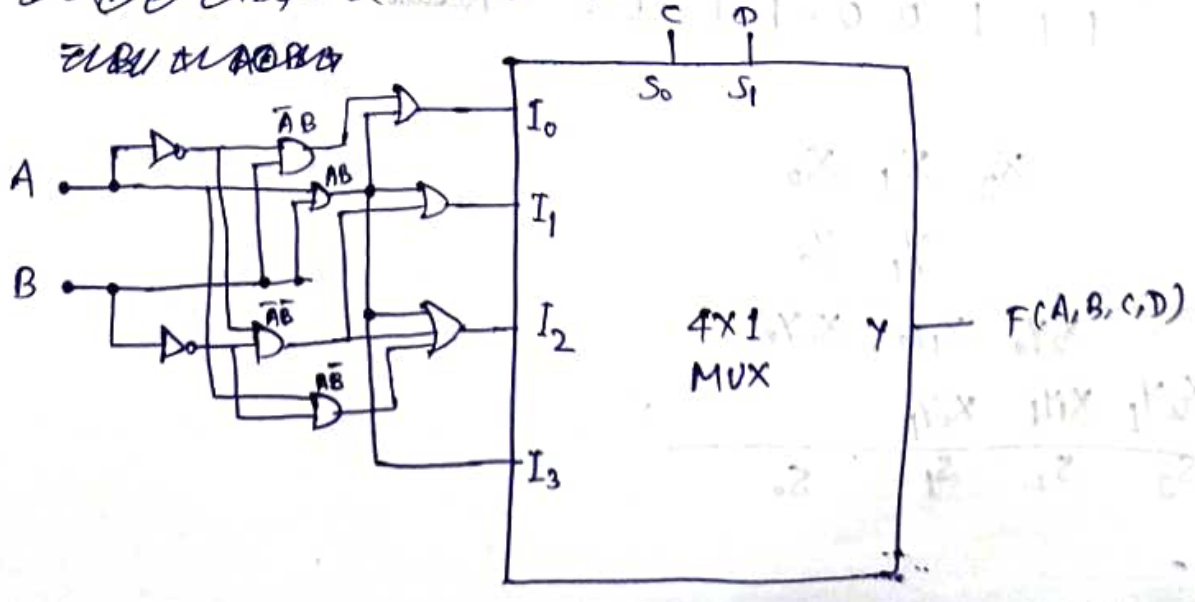


Eg. Implement $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$ using 4x1 MUX.

	$C'D'$ I_0	$C'D$ I_1	CD I_2	CD' I_3
$\bar{A}\bar{B}$	0	(1)	(3)	2
$\bar{A}B$	(4)	5	7	6
AB	(12)	(13)	(15)	(14)
$A\bar{B}$	8	9	(11)	10
	$(\bar{A}B + AB)$	$(\bar{A}\bar{B} + AB)$	$(A\bar{B} + AB + A\bar{B})$	(AB)



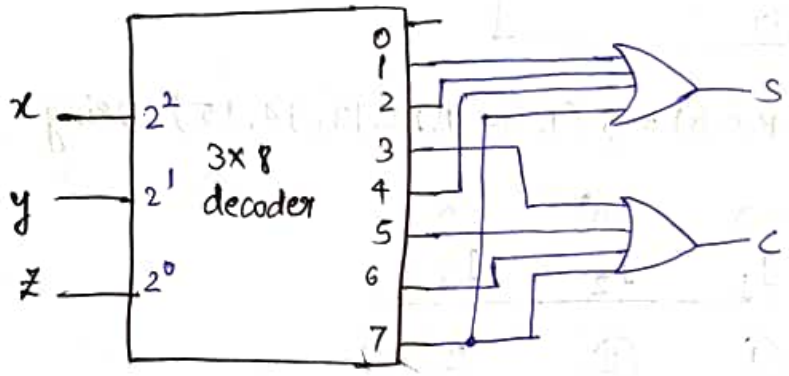
~~$F = (\bar{A}B + AB) + (\bar{A}\bar{B} + AB) + (A\bar{B} + AB + A\bar{B}) + AB$~~



Combinational Circuit Using DECODER

Any combinational circuit with n inputs and m outputs can be implemented with an $n \times 2^n$ line decoder and m OR gates.

Eg. $S(x,y,z) = \Sigma(1,2,4,7)$
$C(x,y,z) = \Sigma(3,5,6,7)$



Full adder using decoder

Multiplier Circuit

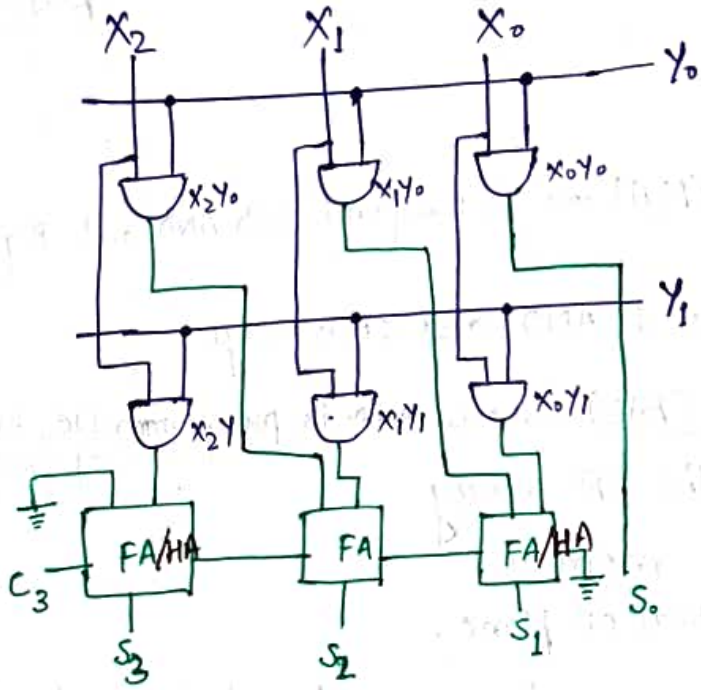
The Binary Multiplication:

```
          1 0 1 0 1 0   Multiplicand
      x       1 0 1 1   Multiplier
    ---------------------
          1 0 1 0 1 0
        1 0 1 0 1 0
      0 0 0 0 0 0       } Partial products
    + 1 0 1 0 1 0
    ---------------------
    1 1 1 0 0 1 1 1 0   Result
```

$X_2\ X_1\ X_0$
$Y_1\ Y_0$

$X_2Y_0\ \ X_1Y_0\ \ X_0Y_0$
$X_2Y_1\ \ X_1Y_1\ \ X_0Y_1$

$S_3\ \ S_2\ \ S_1\ \ S_0$



→ Optimally, only (2 HA + 1 FA are required.

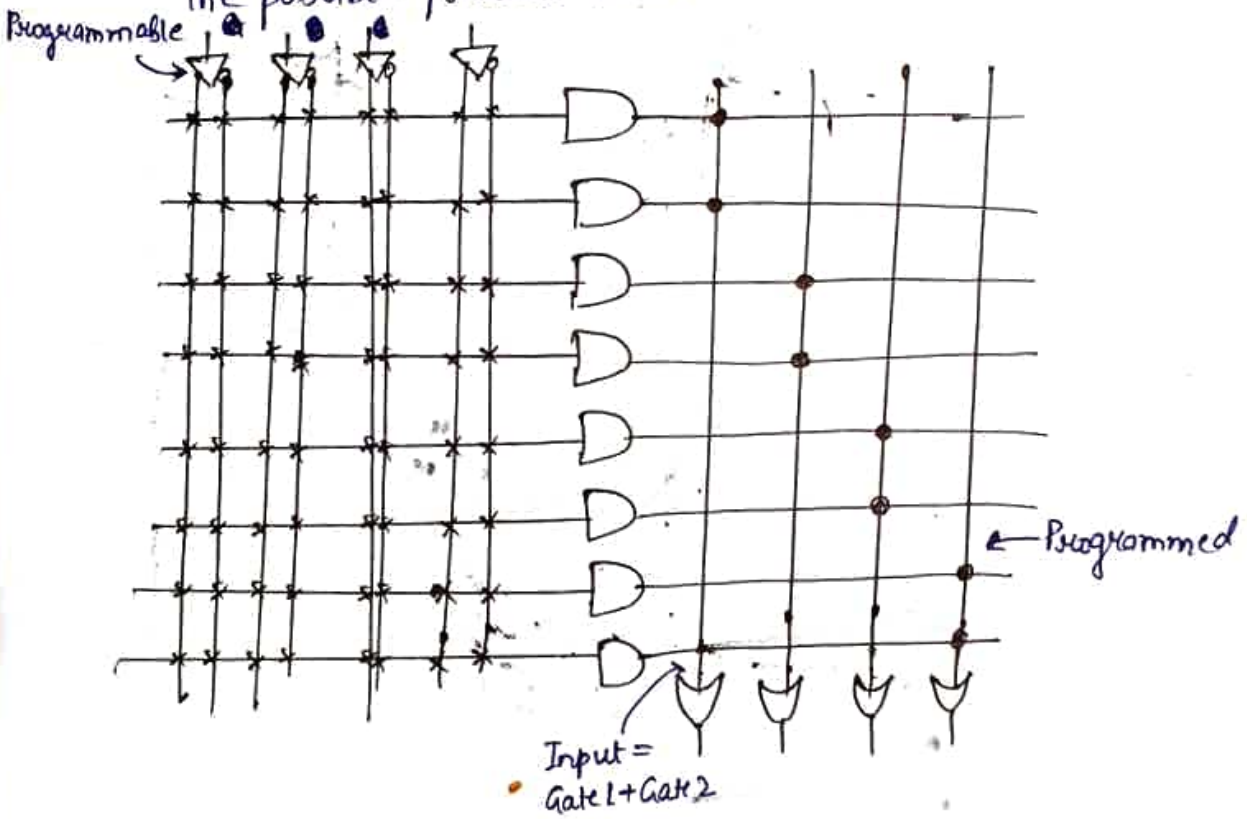
4-09-2023

PALs AND PLAs

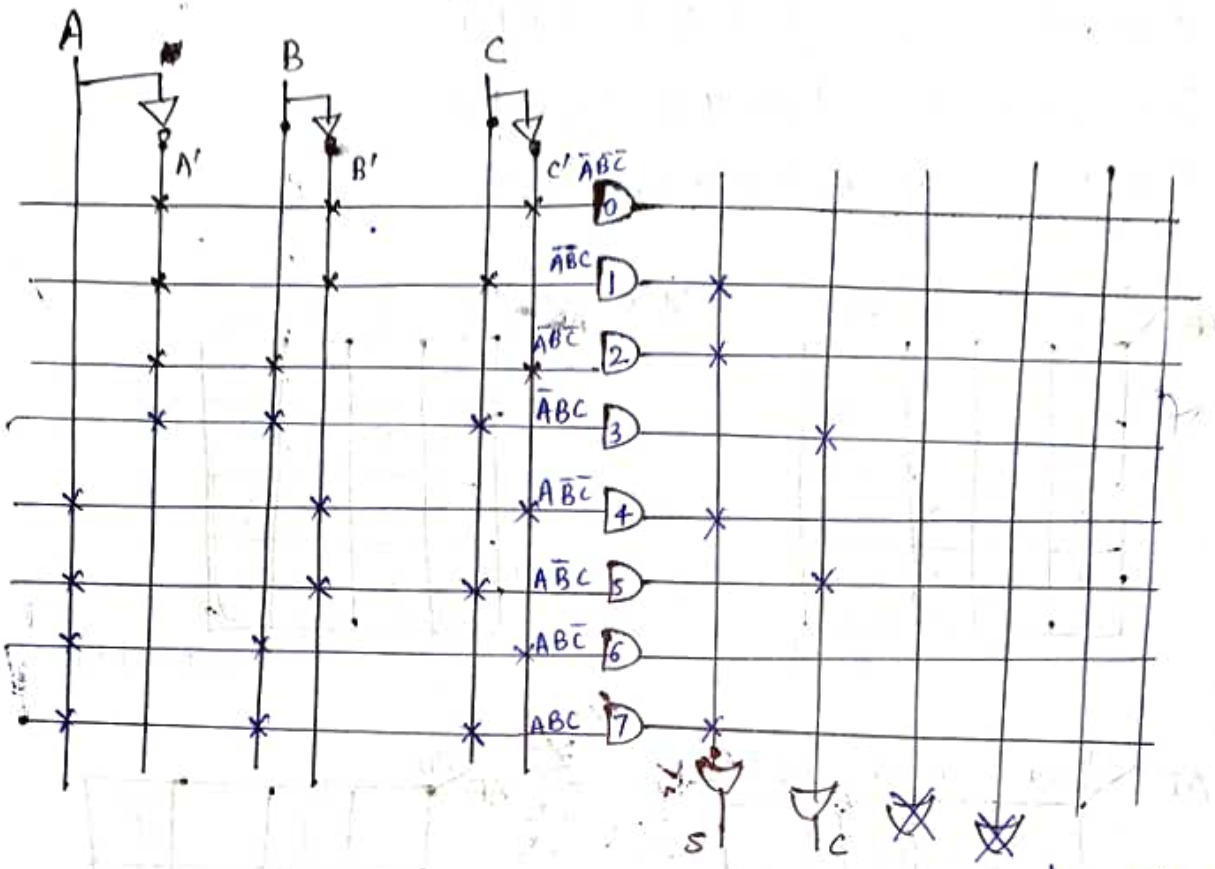
Programmable logic array (PLA) ⇝ can program both AND and OR gates.
- ↳ what we've seen so far
- ↳ Unconstrained fully-general AND and OR arrays.

Programmable array logic (PAL) ⇝ only AND is programmable, fixed no. of OR gates.
- ↳ constrained topology of the OR array
- ↳ Innovation by Monolithic memories
- ↳ faster and smaller (cheaper) OR plane.

→ A given column of the OR array has access to only a subset of the possible product terms.



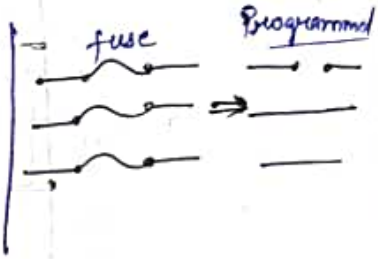
Eg. Design FA using programmable logic devices.
(PLA)



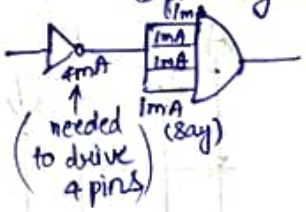
$6 \times 8 \times 3$

- 6 → Input lines → 3 A, B, C and 3 complements
- 8 → AND gates
- 3 → Output lines

- **Loading:** current driving capability to run a gate.



Load / power driving capability



Eg. BCD to gray code converter

BCD				Gray			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1

$W = \sum m(8,9) + d(10,11,12,13,14,15)$

$x = \sum m(4,5,6,7,8,9) + d(10,11,12,13,14,15)$

$y = \sum m(2,3,4,5) + d(10,11,12,13,14,15)$

$z = \sum m(1,2,5,6,9) + d(10,11,12,13,14,15)$

W:

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

$W = A$

x:

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

$x = A + B$

y:

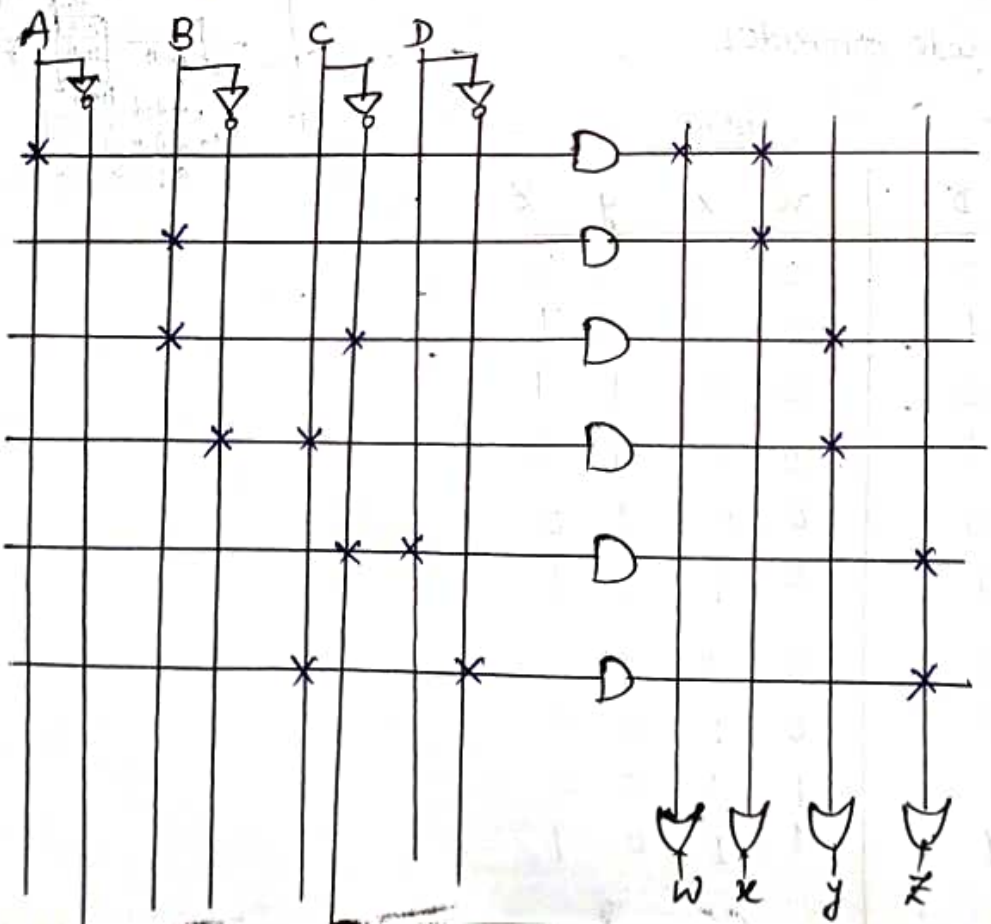
AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

$y = B\bar{C} + \bar{B}C$

z:

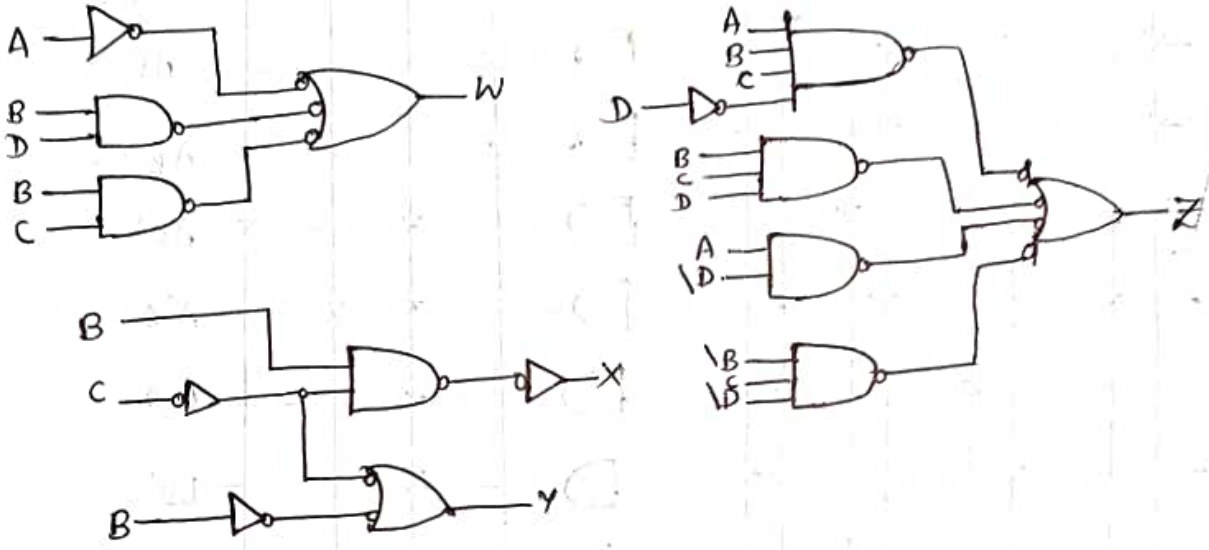
AB \ CD	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	X	X	X	X
10	0	1	X	X

$z = \bar{C}D + C\bar{D}$

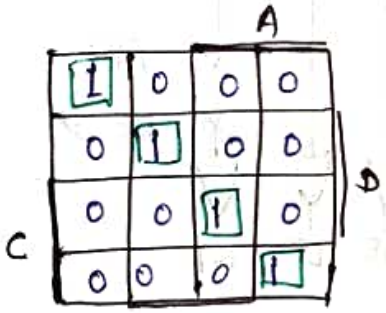


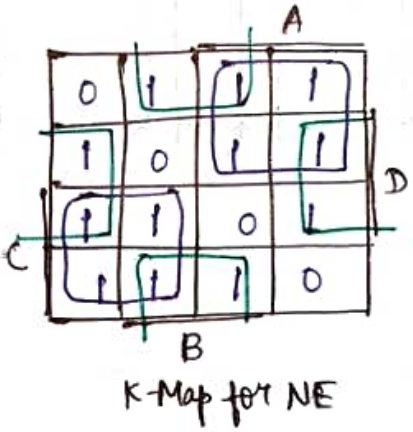
Code Converter : NAND Gate implementation

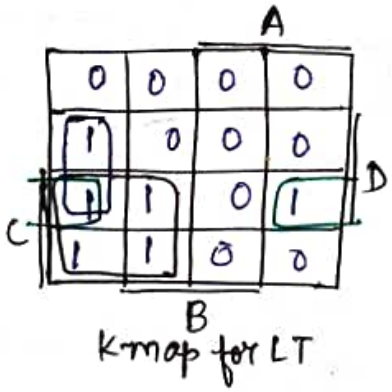
- → loss of regularity, harder to understand
- → harder to make changes.

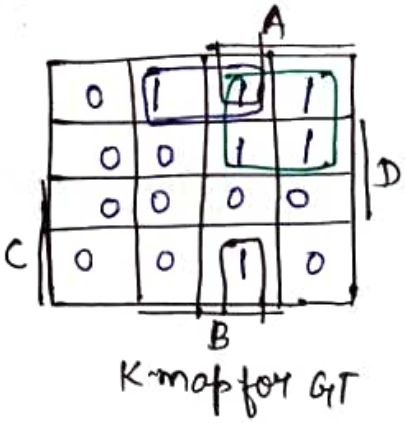


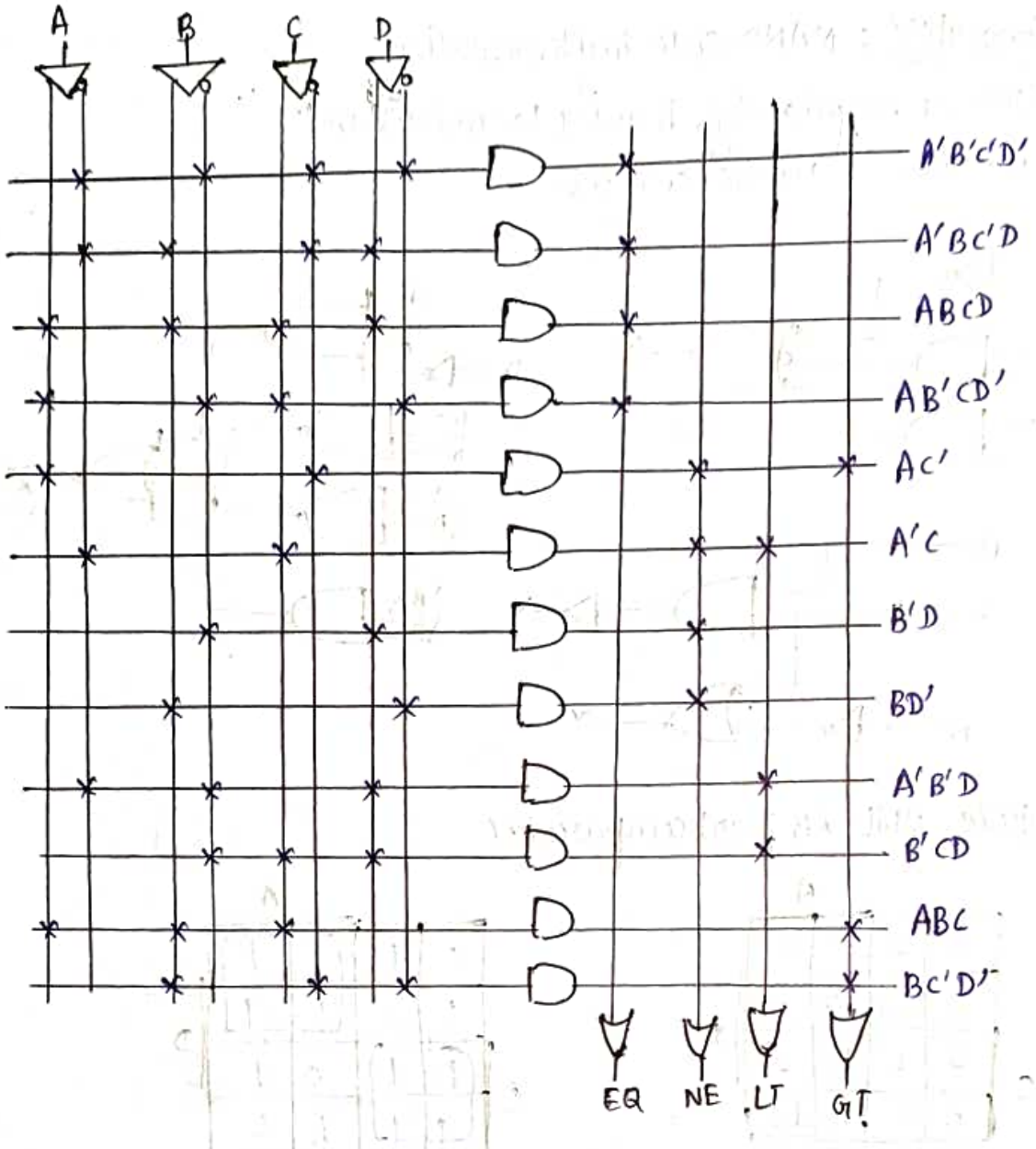
Magnitude 2bit (AB ? CD) comparator





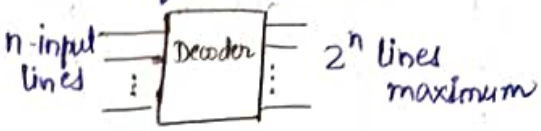





A
B
C
D
A'B'C'D'
A'BC'D
ABCD
AB'CD'
AC'
A'C
B'D
BD'
A'B'D
B'CD
ABC
BC'D'
EQ
NE
LT
GT

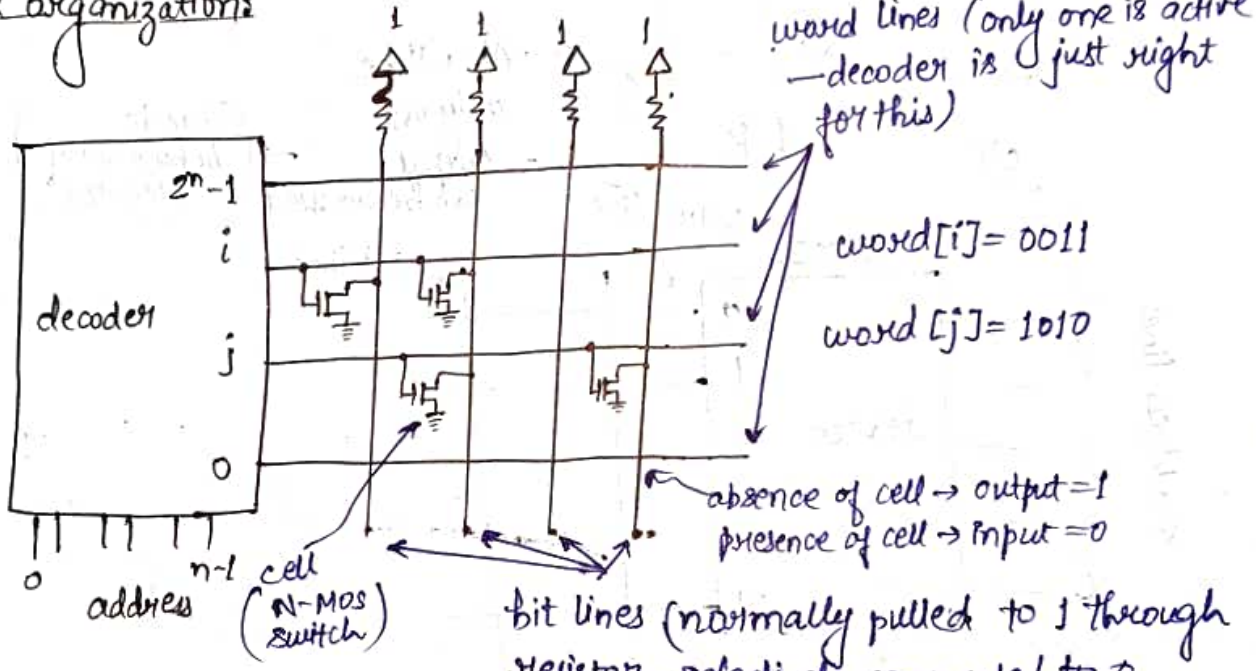
READ-ONLY MEMORIES (ROM)

- One-time programmable
- To store 16-bit information, we need 16 input lines.
- To store Mega, Giga, Tera bits, we require decoder.



Two dimensional array of 1s and 0s
- Entry (row) is called a "word"
- width of row = word-size
- index is called an "address".
- address is input
- selected word output

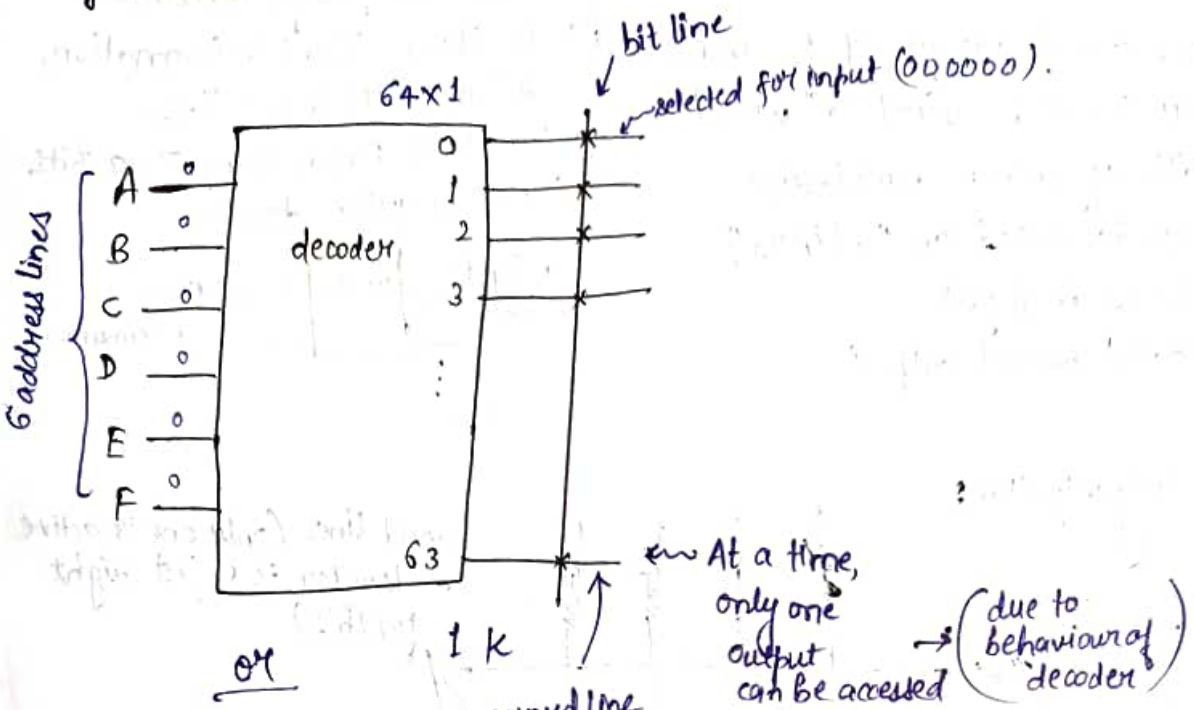
Internal organizations:



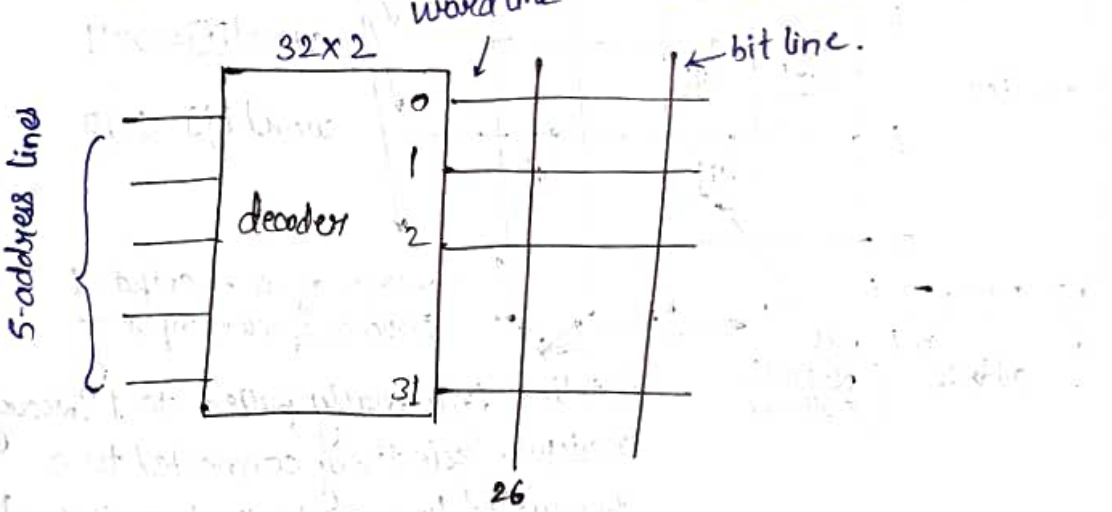
ROM
- Read-only memory (can only be read)
- OTP (One Time Programmable)
- made of static cells
- consist of decoder and memory array.

RAM
- can be re-programmed.
- made of flip-flops.

Eg 64-bit ROM



or



16x4

4 to 16 x 4 bits

5-09-2023

Combinational logic implementation (two-level canonical form) using a ROM

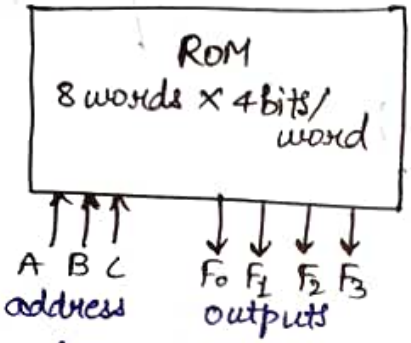
$F_0 = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$

$F_1 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC$

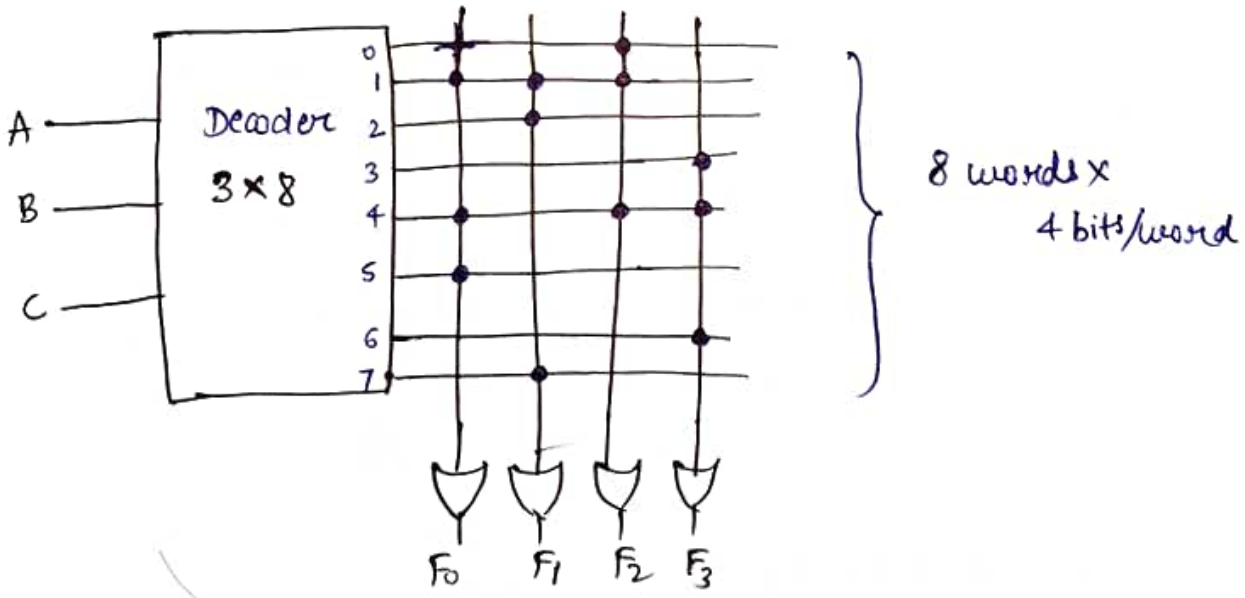
$F_2 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C}$

$F_3 = \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$

A B C	F_0	F_1	F_2	F_3
0 0 0	0	0	1	0
0 0 1	1	1	1	0
0 1 0	0	1	0	0
0 1 1	0	0	0	1
1 0 0	1	0	1	1
1 0 1	1	0	0	0
1 1 0	0	0	0	1
1 1 1	0	1	0	0



Block diagram



Memory

- Volatile ⇝ Erases data on power off (e.g., RAM, Cache)
- Non-volatile ⇝ Remains even after power is off (e.g., ROM, USB drives)
  - ↳ static

Programmable Devices:

AND-OR-array → programmable

- PROM (Programmable Read-only Memory)
- PLA (Programmable Logic Array)
- PAL (Programmable Array logic)
- CPLD (Complex Programmable Logic Device)
- FPGA (Field Programmable Gate Array)

RAM
↳ erases memory when power is off.

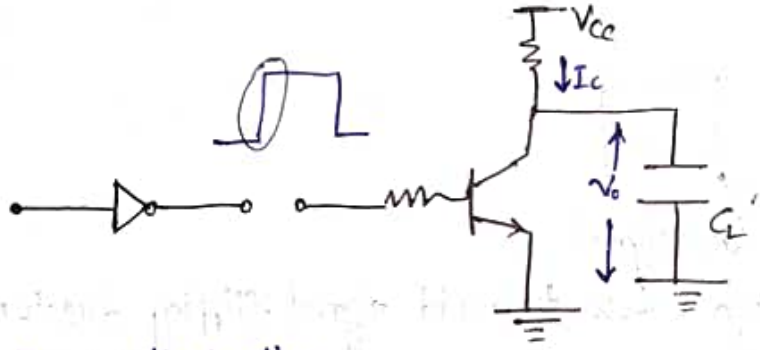
ROM
↳ stores memory even ~~when~~ if power is off.

ELECTRICAL CHARACTERISTICS OF LOGIC GATES

1. Power dissipation
2. Propagation delay
3. Noise margin
4. (Loading) Fan in / Fan out

Figure of ment = Power × Delay
↳ constant

- BJT supports frequency below radiofrequency
- For analog, Figure of ment = Gain × Bandwidth (range of freq. in which device works properly.)



- Static power: Power taken from source when source is ON.
- Dynamic power:

Power dissipation,

$P_D = P_{DP} + P_{SP}$, P_{DP} : dynamic power, P_{SP} : static power

$P_{SP} = V_{cc} I_c$

$i(t) = C\frac{dV}{dt}$; $\frac{\Delta V}{\Delta t} = \frac{V_{cc}}{t_d}$ (V_{cc} ⇝ max. V which C can be charged with; t_d ⇝ max. time)

$\Rightarrow i(t) = \frac{C V_{cc}}{t_d}$

$P_D = V_{cc}\, i(t) = \frac{V_{cc} \cdot C_L \cdot V_{cc}}{t_d} = \frac{V_{cc}^2 C_L}{t_d} \Rightarrow P_D \cdot t_D = \underbrace{V_{cc}^2 C_L}_{\text{fixed (based on device)}}$

$\Rightarrow P_D = V_{cc}^2 C_L \cdot f$, f : frequency of switching $= \frac{1}{t_d}$

$f \uparrow \Rightarrow P_D \uparrow \Rightarrow$ Less control over PD. (using high frequency switches).

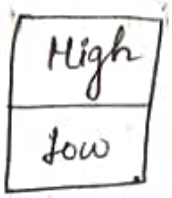
$\rightarrow P_D \propto V_{cc} \Rightarrow V_{cc} \downarrow \Rightarrow P_D \downarrow$

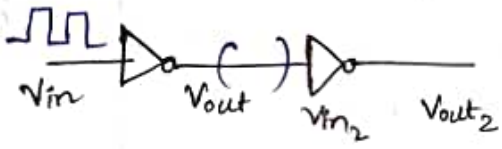
Propagation Delay

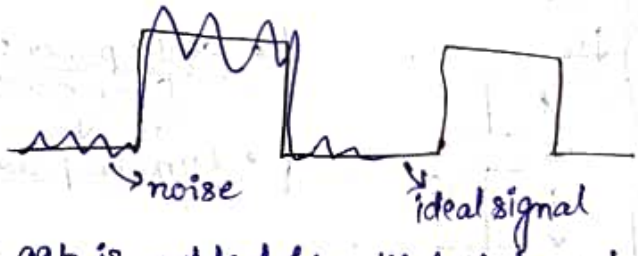
$$t_p = \frac{t_{LH} + t_{HL}}{2}$$

Noise Margin

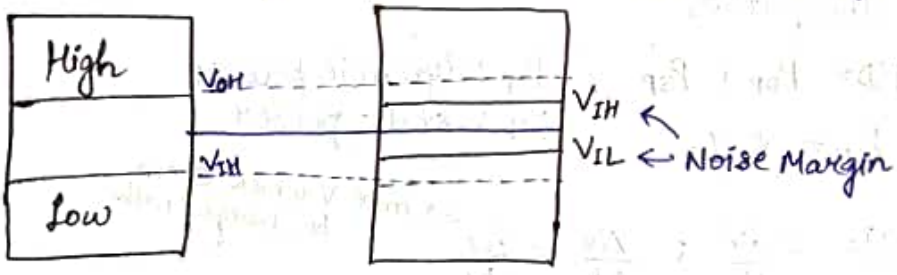
Only two levels in digital electronics:







→ Some gap is added b/w High & Low to avoid signal flipping due to noise.

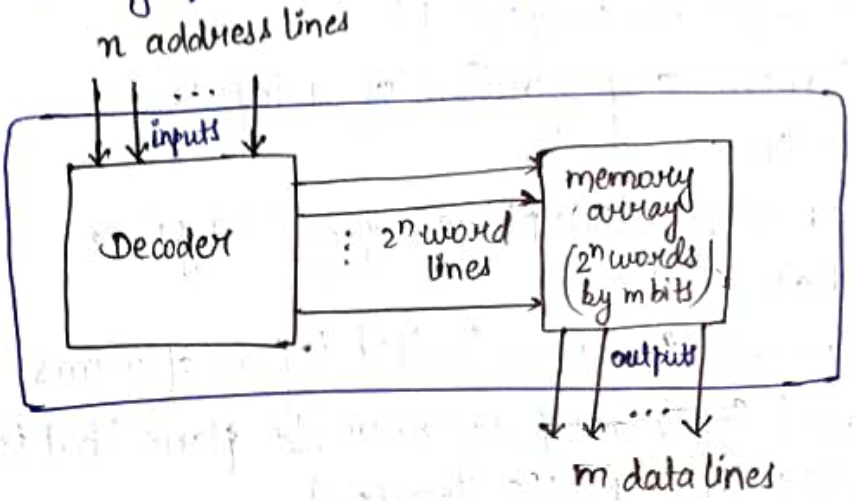


$$NM_L = V_{IL} - V_{OL}$$

$$NM_H = V_{OH} - V_{IH}$$

ROM structure:

↳ Similar to a PLA structure but with a fully decoded AND array
  ↳ completely flexible OR array (unlike PAL)



ROM vs. PLA

→ ROM approach advantageous when:
- design time is short (no need to minimize output functions)
- most output combinations are needed (e.g., code converters)
- little sharing of product terms among output functions.

→ ROM problems:
- size doubles for each additional input
- can't exploit don't cares

→ PLA approach advantageous when:
- design tools are available for multi-output minimization.
- there are relatively few unique minterm combinations.
- many minterms are shared among the output functions.

→ PAL problems:
- constrained fan-ins on OR plane.

Regular logic structures for two-levels logic:

→ ROM: full AND plane, general OR plane
- cheap (high-volume component)
- can implement any function of n inputs.
- medium speed.

→ PAL: programmable AND plane, fixed OR plane
- intermediate cost
- can implement functions limited by no. of terms.
- high speed (only one programmable plane that is much smaller than ROM's decoder).

→ PLA: programmable AND and OR planes
- most expensive (most complex in design, need more sophisticated tools).
- can implement any function upto a product term limit.
- slow (two programmable planes).

Regular logic structures for multi-level logic.

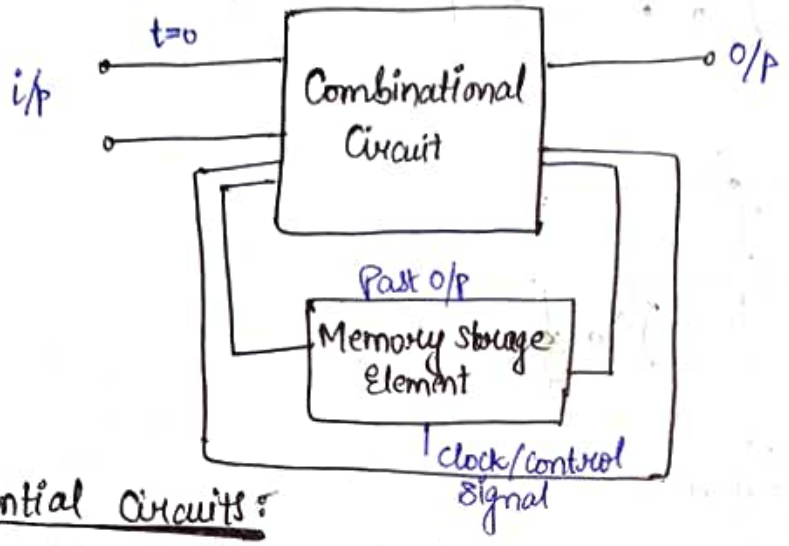
→ Difficult to devise a regular structure for arbitrary connections b/w a large set of different types of gates.
- efficiency/speed concerns for such a structure.
- field programmable gate ~~sta~~ arrays (FPGAs) are just such programmable multi-level structures.
  - programmable multiplexers for wiring.
  - lookup table for logic functions (programming fills in the table).
  - multi-purpose cells (utilization is the big issue).

→ Use multiple levels of PALs/PLAs/ROMs:
- output intermediate result.
- make it an input to be used in further logic.

12-09-2023

SEQUENTIAL CIRCUITS

clock ↓ controller enable



Sequential Circuits:

① Synchronous Sequential Circuit ⇝ clock is used

② Asynchronous Sequential Circuit

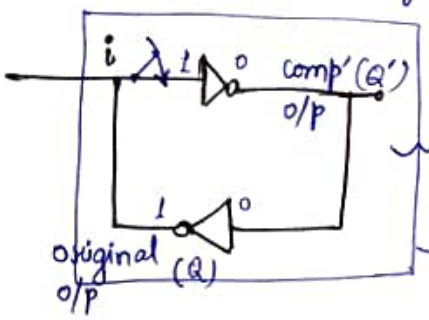
Memory Element:

Latch / Flip Flop (1 bit)

↳ Holds information indefinitely till power is given, or until input signal is disturbed.

↳ NOR or NAND Gate used

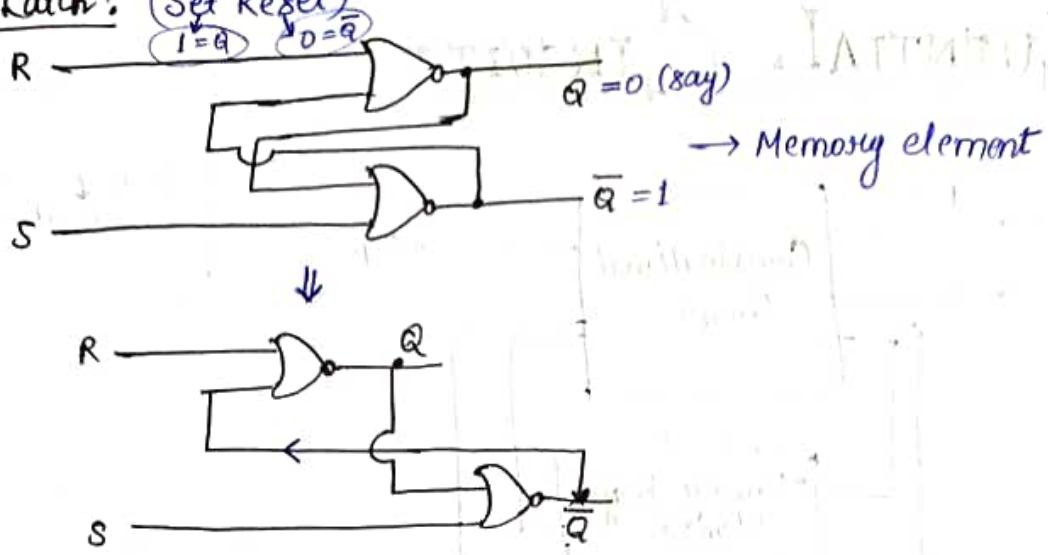
↳ To hold the info. indefinitely, a feedback is required.



⇝ Back-to-back inverter to store info indefinetly.

→ Cell for RAM

SR Latch: (Set Reset)



NOR Gate characteristics:

Inputs		output
0	0	1
1	0	0
0	1	0
1	1	0

⇝ o/p = 0, if one of the i/p = 1
(doesn't check other i/p)
↳ Responds quickly

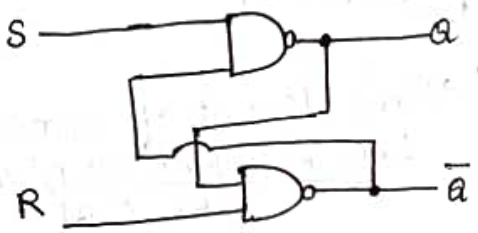
S	R	Q	Q̄
0	0	0	1
(1)	0	(1)	0
0	(1)	(0)	1
1	1	Undetermined state (1 0)	

⇝ When i/p = (0, 0), it maintains previous o/p.

S	R	Q	Q̄
0	0	0	1
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	Undetermined	

NAND Gate:

I/P		O/P
0	0	1
1	0	1
0	1	1
1	1	0



S	R	Q	$\bar{Q}$
1	1	Q(t)	$\bar{Q}(t)$
0	1	1	0
1	0	0	1
0	0	Indetermined	

~~not determined~~ → Maintains the previous state

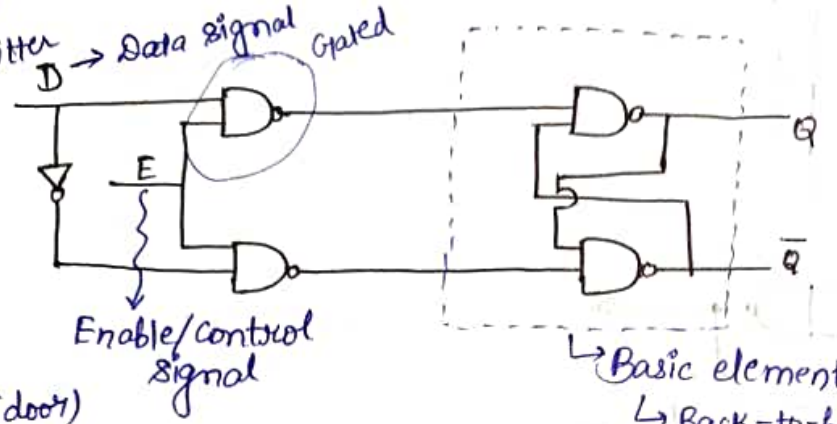
15-09-2023

D-Latch / D-FF

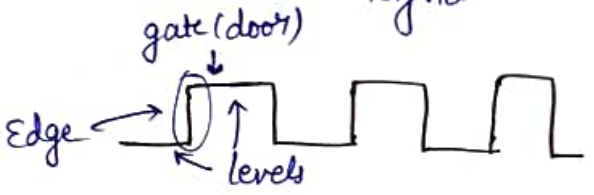
↳ Data-Latch

↓

Data selector/
Data Transmitter



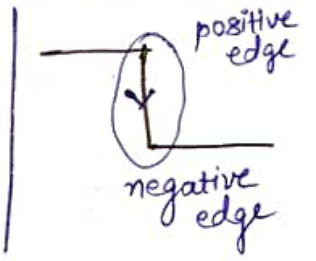
↳ Basic element of memory
↳ Back-to-back NOT/NAND gates. (SR)



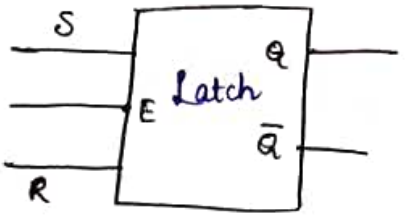
(high or low)

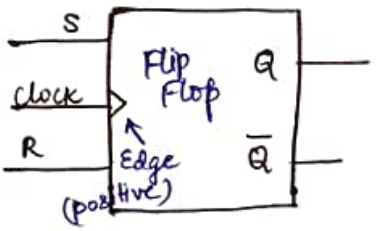
A device that works on levels → Latch
↳ Level-sensitive devices

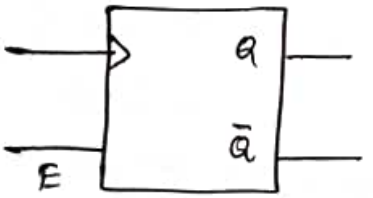
A device working on ~~at~~ edge → Flip-flop
↳ Edge-sensitive device

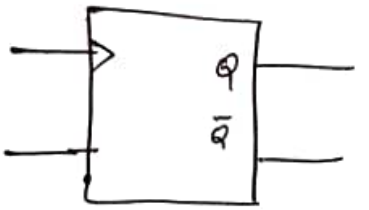


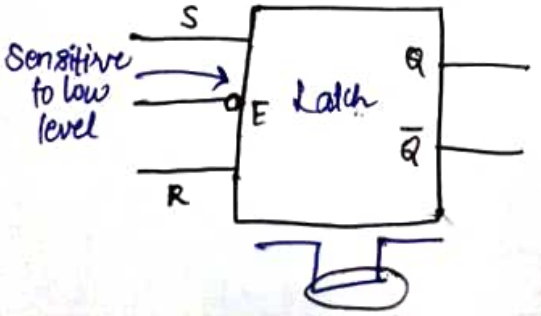
E	D	Q(t)	Q	$\bar{Q}$
⊓ (=1)	0	Q(t)	0	1
Indetermined	1	Q(t)	1	0

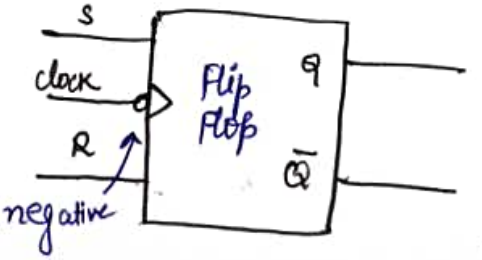




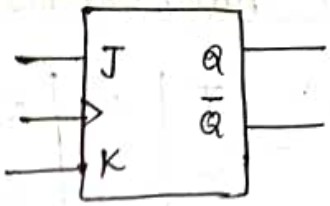








J K FlipFlop



F K
↓ ↓
S R

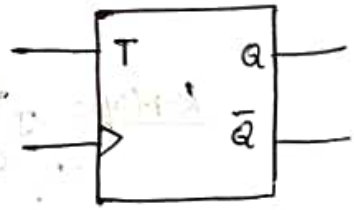
Truth Table:

S / J	R / K	Q(t)	Q(t+1) = Q(t+1)	$\bar{Q}$(t+1)	
0	0	0	0	1	
0	0	1	1	0	
0	1	0	0	1	Reset → doesn't depend on ~~input~~ Q(t).
0	1	1	0	1	
1	0	0	1	0	
1	0	1	1	0	
1	1	0	1	0	→ Toggle the present state Q(t).
1	1	1	0	1	

→ 2 input SR FF disadvantages are overcome by JK FF by toggling the present state when input = (1,1).

Toggle FF (TFF)

T ← Toggle	Q(t)	Q	$\bar{Q}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

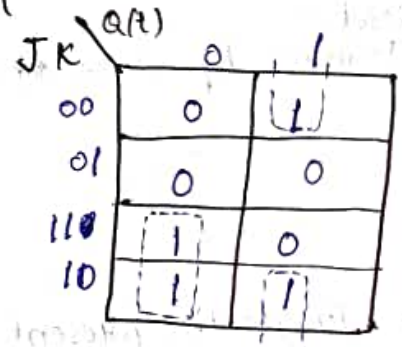


19-09-2023

JK FF characteristics Table:

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

K-Map:

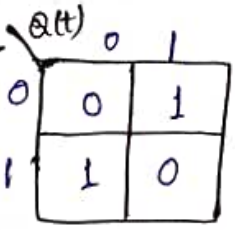


$\therefore Q(t+1) = J\overline{Q(t)} + \overline{K}Q(t)$

T FF characteristics Table:

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

K-Map:

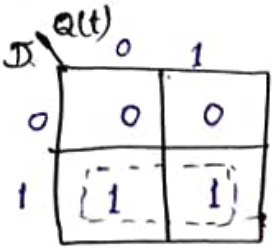


$\therefore Q(t+1) = T \oplus Q$
$= TQ' + T'Q$

D FF characteristics Table:

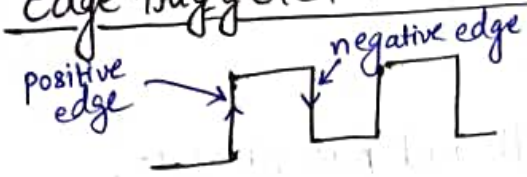
D	Q(t)	Q(t+1)
0	Q(t)	0
1	Q(t)	1

K-Map:



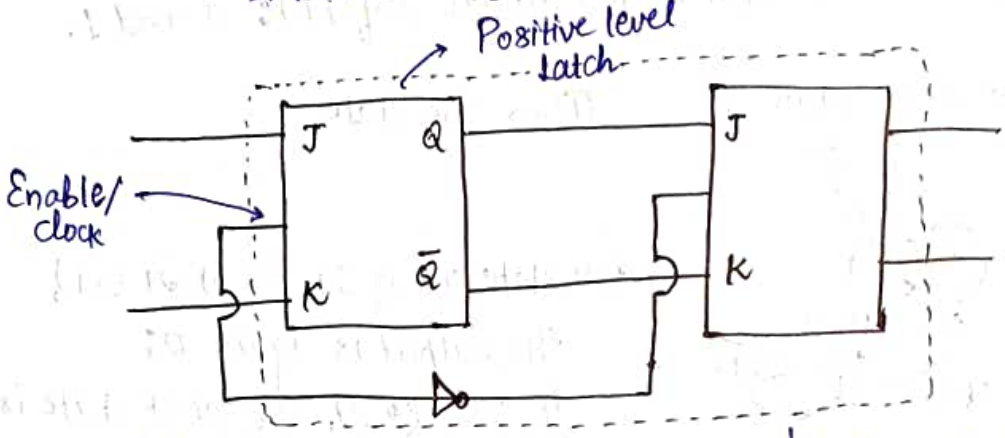
$\therefore Q(t+1) = D$

Edge Triggered Latch (FF)



→ Using Master and Slave
  ↳ 2 Latches required; one act as master, other as slave.

(Assume $t_p = 0$)



↳ Becomes Edge sensitive (Negative Edge Trigger Latch)

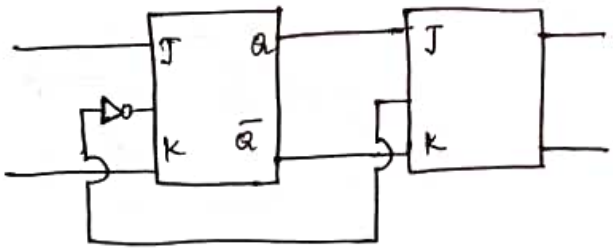
Device senses the latest input here

Device is sensitive at ove-to-ove edge (ON → OFF)

→ Output should be stable irrespective of the input coming

→ Timing (t_p) is important.

Positive Edge Trigger Latch:



Sequential Circuit Design

① State equation → Equation contains inputs and present state.

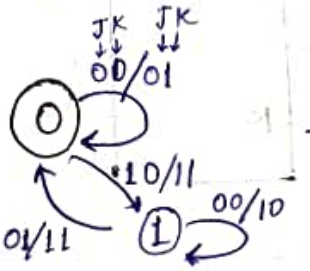
② State Table → One of the inputs is the present state.

③ State Diagram → Two states for inputs (of FF): 0 and 1.

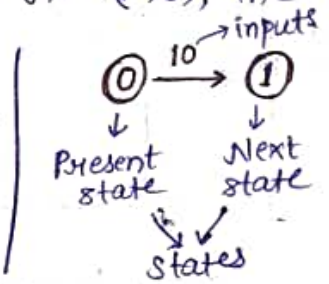
(0) → zero state ; (1) → one state

JK FF:

State Diagram:



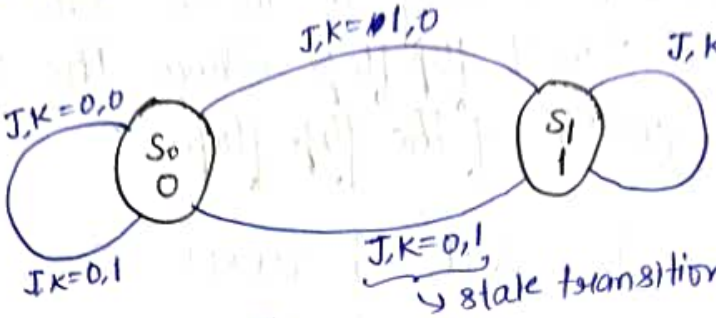
→ For state 0, if JK = (0,0) or (0,1), the output is again 0;
if JK = (1,0), the next state is 1.



21-09-2023

Moore Machine

↳ Moore machine output depends on the states only.



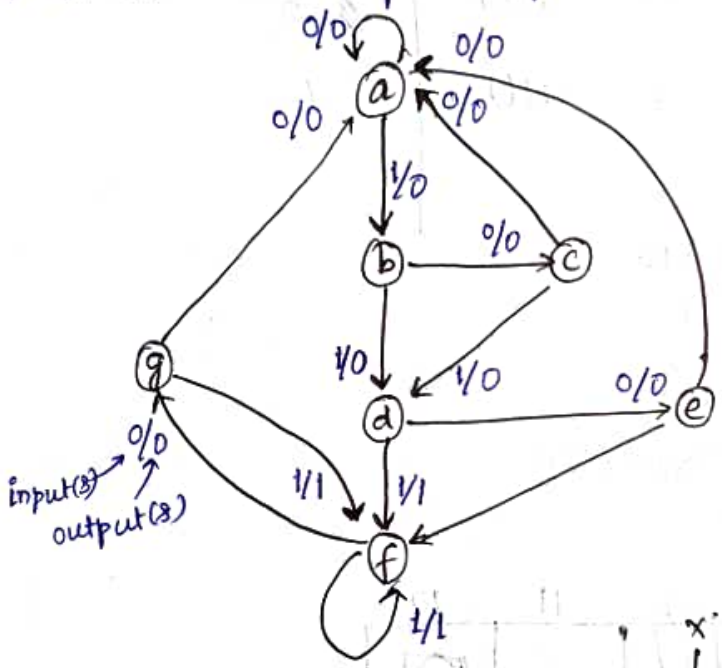
→ 2 inputs (J,K)

→ All FFs are moore machine.

↳ state transition

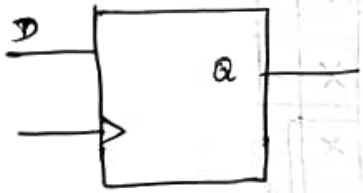
Mealy Machine

↳ Mealy machine output depends not only on the states but also on the external ~~point~~ input.

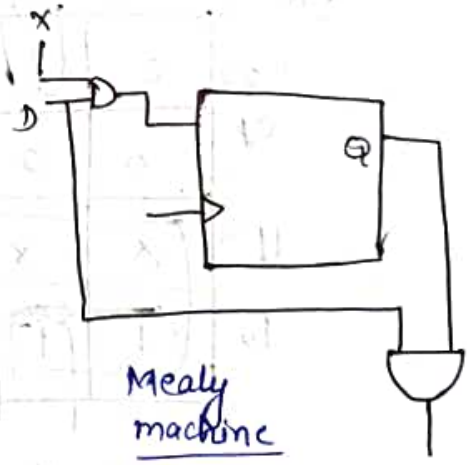


→ one input, one output

Eg.,



Moore machine



Mealy machine

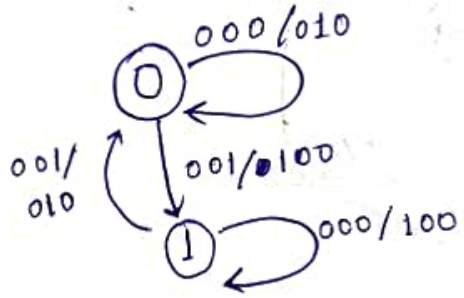
→ To get controlled output, Moore machine is used.
(We don't get output immediately; we have to wait.)

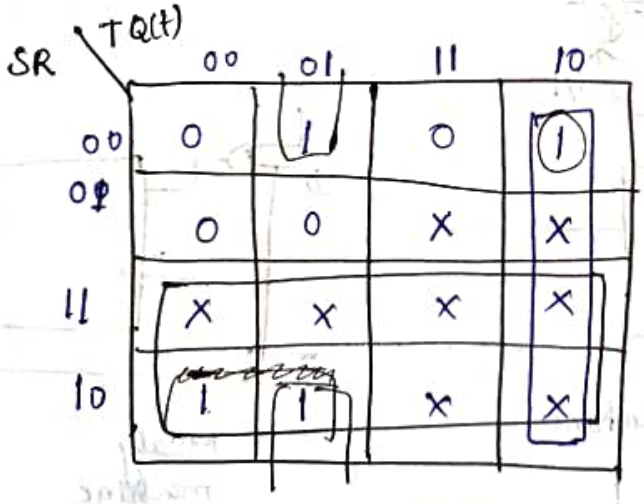
→ To get uncontrolled output, Mealy machine is used.
(We get output immediately after giving input.)

Q) We have a new FF with 3 inputs S, R and T. No more than one of these inputs may be 1 at anytime. The S and R inputs behave exactly as they do in an S,R. T input behaves as it does in T flip flop. Show the state diagram. Obtain the logic equation of the flip flop.

Soln:

S	R	T	Q(t)	Q(t+1)
0	0	0	Q(t)	$\bar{Q}(t)$
0	0	1	Q(t)	$\bar{Q}(t)$
0	1	0	Q(t)	0
1	0	0	Q(t)	1





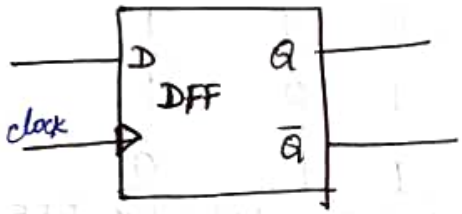
~~$Q(t+1) = S\bar{R}\bar{T} + \bar{R}TQ(t) + \bar{S}\bar{R}T\bar{Q}(t)$~~

$$Q(t+1) = S + \bar{R}\bar{T}Q(t) + T\bar{Q}(t)$$

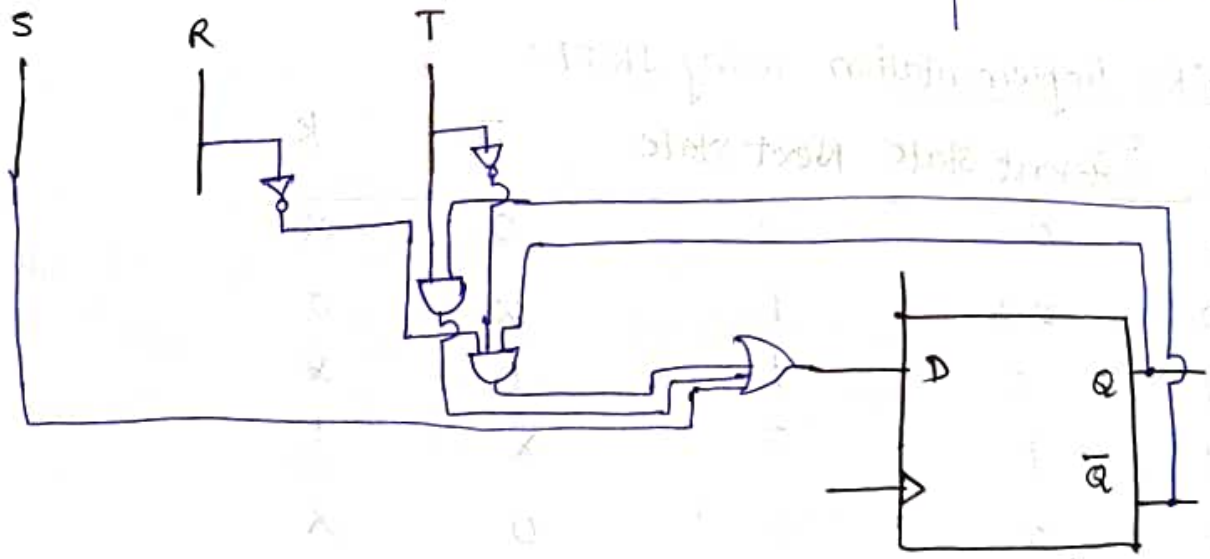
SRT	Present state	Next State	D
000	0	0	0
000	1	1	1
001	0	0	1
001	1	1	0
010	0	0	0
010	1	1	0
100	0	0	1
100	1	1	1

22-09-2023

SRT implementation Using D-FF:



$D = S + T\bar{Q} + \bar{R}\bar{T}Q$



For one SRT
↓
one DFF required

SRT → 2 states
D → 2 states

For 4 state FF
↓
2 DFF required

For 8 state FF
↓
3 DFF required

Excitation Table for Flip flops:

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

Excitation table for SR FF

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

Excitation table for D FF

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

Excitation table for JK FF

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

Excitation table for T FF

SRT Implementation using JK FF:

SRT	Present state	Next state	J	k
000	0	0	0	X
000	1	1	X	0
001	0	1	1	X
001	1	0	X	1
010	0	0	0	X
010	1	0	X	1
100	0	1	1	X
100	1	1	X	0

K-Map for J:

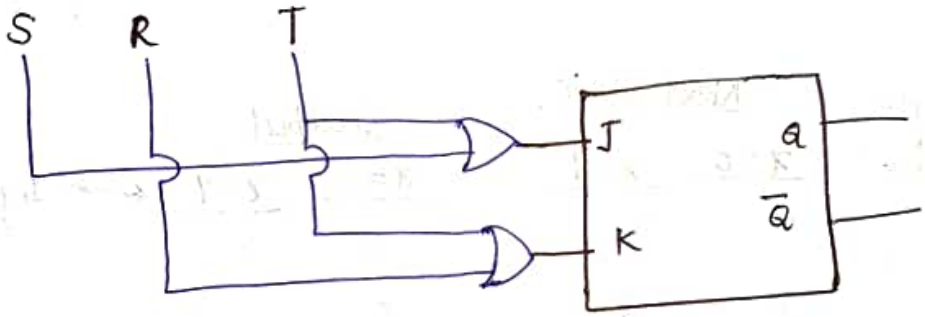
SR \ TQ	00	01	11	10
00	0	X	X	1
01	0	X	X	X
11	X	X	X	X
10	1	X	X	X

$J = S + T$

K-Map for K:

SR \ TQ	00	01	11	10
00	X	0	1	X
01	X	1	X	X
11	X	X	X	X
10	1	X	X	X

$K = T + R$



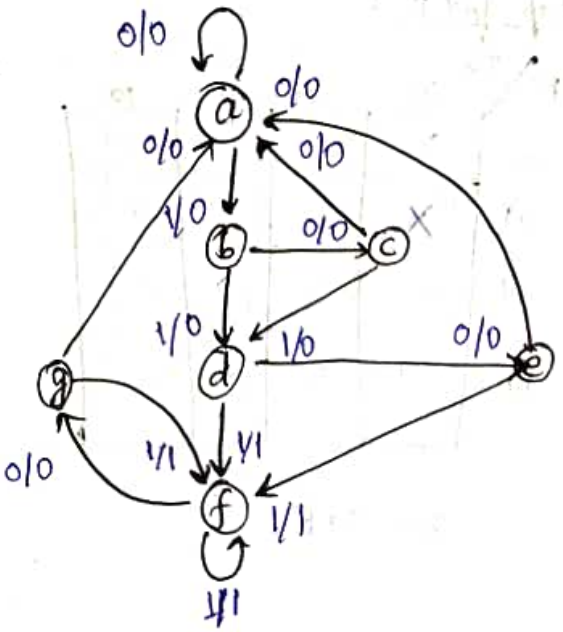
① The word description of the circuit is stated. This may be ~~accomplished~~ accompanied by a state diagram, a timing diagram, or other pertinent information.

② From the given information about the circuit, obtain the state table.

③ The no. of states may be reduced by state-reduction methods if the sequential circuit can be characteristiesed by input-output relationships independant.

26-09-2023

eg.



State Table:

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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

← Inputs

→ Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state.

→ When two states are equivalent, one of them can be removed without altering the input-output relationships.

b & c are equivalent, provided a & c are eq.

a & c are equivalent, provided b & d are eq. → Can't be made equivalent b/c o/p are different.

e & g are eq.

d & f are eq.

Reduced state Table

Present state	Next State x=0	Next State x=1	Output x=0	Output x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Three Possible Binary state Assignments

state	Assignment 1	Assignment 2	Assignment 3
a	001	000	000
b	010	010	100
c	011	011	010
d	100	101	101
e	101	111	011

One hot assignment : Only one bit is High.

One cold assignment : Only one bit is Low.

State	one hot assignment
a	00001
b	00010
c	00100
d	01000
e	10000

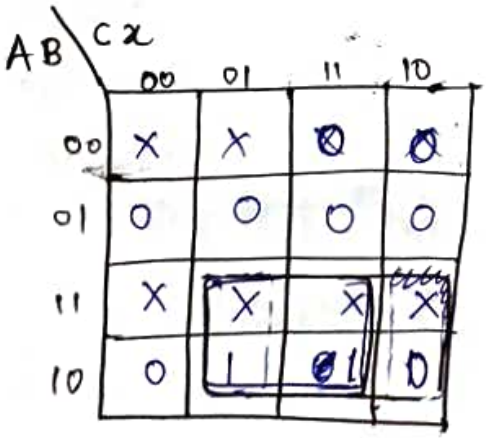
Assigning a, b, c, d, e = 001, 010, 011, 100, 101, in the reduced state table,

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
001	001	010	0	0
010	011	100	0	0
011	001	100	0	0
100	101	100	0	1
101	001	100	0	1

PS ABC	x	NS ABC	O/P	T_A	T_B	T_C
001	0	001	0	0	0	0
001	1	010	0	0	1	1
010	0	011	0	0	0	1
010	1	100	0	1	1	0
011	0	001	0	0	1	0
011	1	100	0	1	1	1
100	0	101	0	0	0	1
100	1	100	1	0	0	0
101	0	001	0	1	0	0
101	1	100	1	0	0	1

K-Map for Q, T_A, T_B, T_C.

K-Map for Q:



~~O/P = $\bar{A}\bar{B}\bar{C}x + \bar{A}\bar{B}C\bar{x} + A\bar{B}\bar{C}\bar{x} + A\bar{B}Cx$~~

O/P = Ax

K-Map for T_A:

AB \ Cx	00	01	11	10
00	X	X	0	0
01	0	1	1	0
11	X	X	X	X
10	0	0	0	1

$T_A = Bx + Ac\bar{x}$

K-Map for T_B:

AB \ Cx	00	01	11	10
00	X	X	1	0
01	0	1	1	1
11	X	X	X	X
10	0	0	0	0

$T_B = \bar{A}x + BC$

K-Map for T_C:

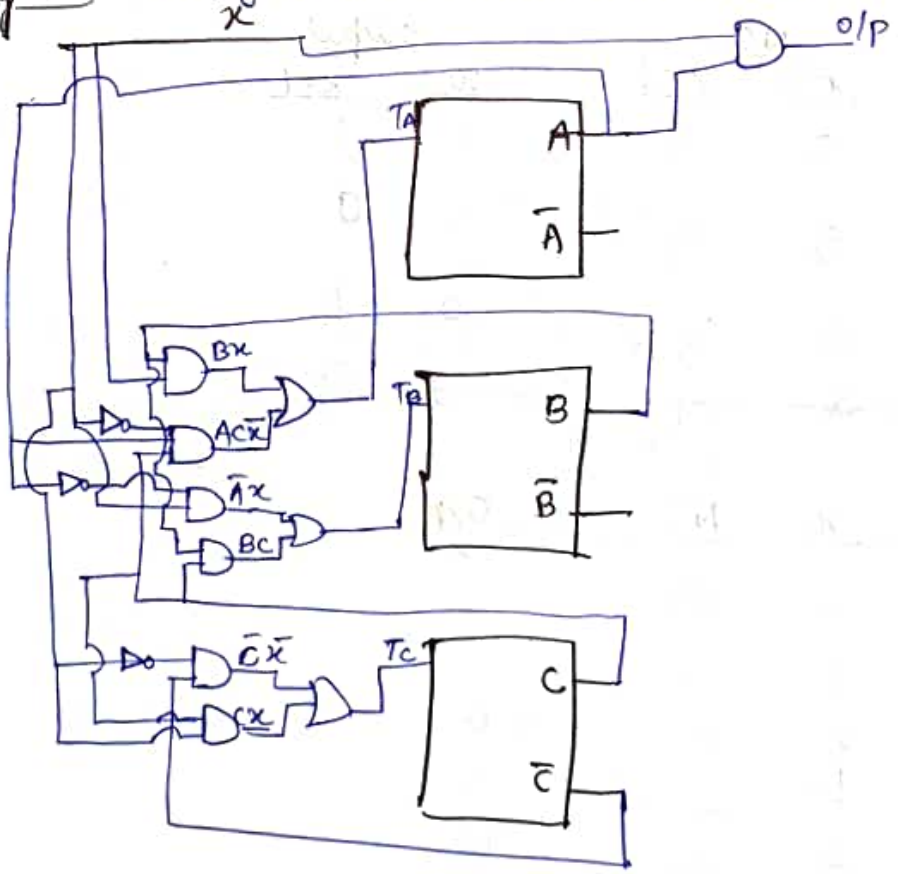
AB \ Cx	00	01	11	10
00	X	X	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	1	0

$T_C = \bar{c}\bar{x} + cx$

Steps:

State Table
↓
State Diagram
↓
Reduced state table
↓
Assignment
↓
Decide no. of flip flops required
↓
Excitation table
↓
K-Map
↓
Logic circuit

Logic Diagram: Mealy Machine — Finite State Machine (FSM)



29-09-2023

Sequence Detector

Detecting the sequence 101 using Mealy machine
- ↳ output = 1, if detected/encountered.
- ↳ output = 0, otherwise.

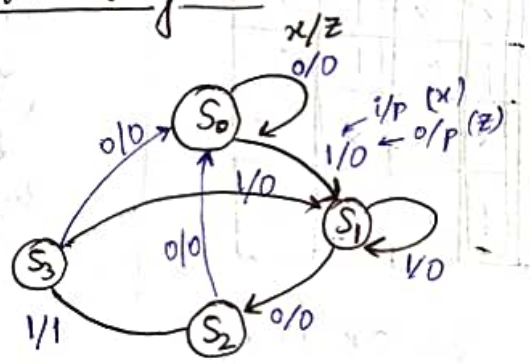
1 0 1 (non-overlapping)

x	1	0	1	1	1	0	1	0	1 ← Input
z	0	0	1	0	0	0	1	0	0 ← output

non-overlapping → 00100
overlapping → 00101

10101

State Diagram:



→ No useful info ⇒ Reset → 0 (same state)
→ When a useful info comes, force to a new state

State Table:

PS	NS		output	
	x=0	x=1	x=0	x=1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_0	0	1
~~S_3~~	~~S_0~~	~~S_1~~	~~0~~	~~0~~

PS (A B)	x	NS	O/P x
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_2	0
S_1	1	S_1	0
S_2	0	S_0	0
S_2	1	S_0	1

State Assignment:

S_0	0 0
S_1	0 1
S_2	1 0

(Using JK FF)

For SRT FF, use $S_A R_A T_A$, $S_B R_B T_B$.

PS A B	x	NS A B	O/P x	J_A	K_A	J_B	K_B
0 0	0	0 0	0	0	X	0	X
0 0	1	0 1	0	0	X	1	X
0 1	0	1 0	0	1	X	X	1
0 1	1	0 1	0	0	X	X	0
1 0	0	0 0	0	X	1	0	X
1 0	1	0 0	1	X	1	0	X

K-Map for O/P(Q):

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

$Q = Ax$

K-Map for J_A:

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

$J_A = B\bar{x}$

K-Map for K_A:

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

$K_A = 1$

K-Map for J_B:

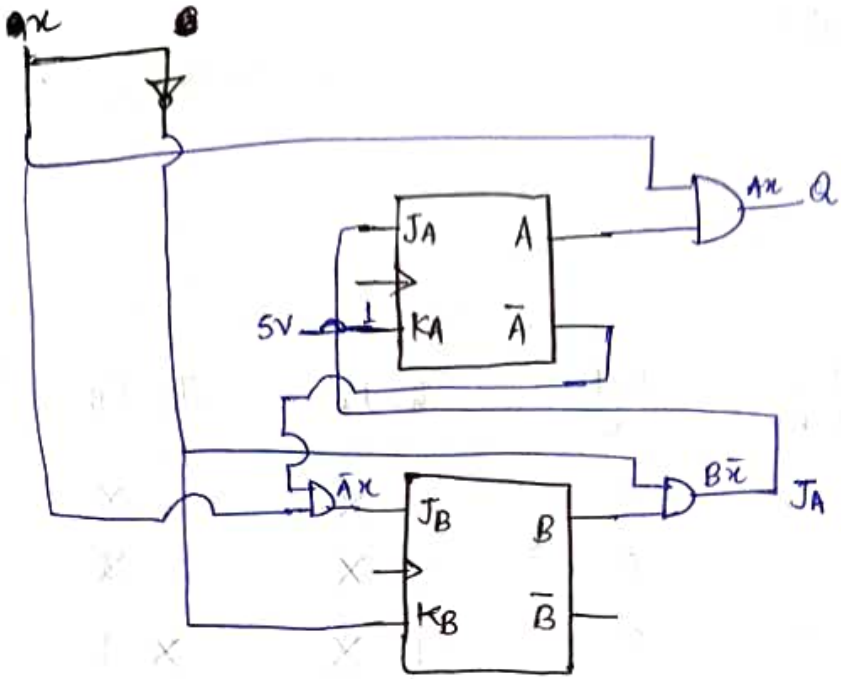
AB \ x	0	1
00	0	1
01	X	X
11	X	X
10	0	0

$J_B = \bar{A}x$

K-Map for K_B:

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

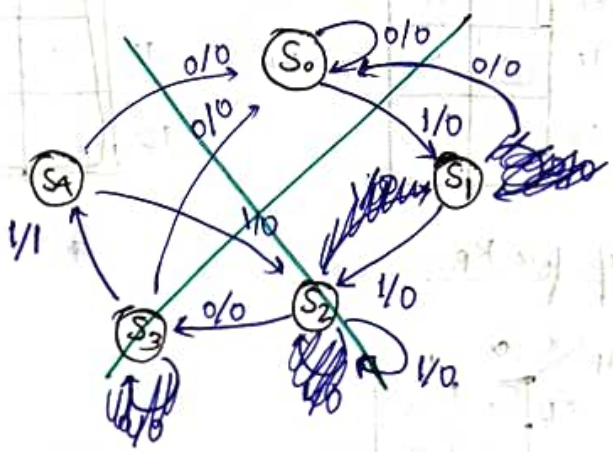
$K_B = \bar{x}$

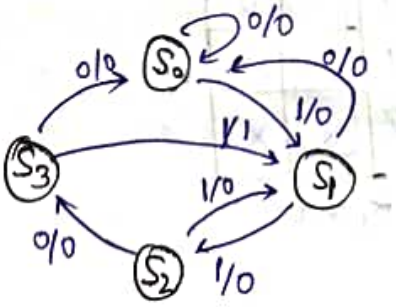


Q) Design a sequence detector which detects 1101 in overlapping mode using D-flip flop (Use binary assignment).

Soln: State Diagram:

1101





State Table:

PS	NS		O/P	
	x=0	x=1	x=0	x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_1	0	0
S_3	S_0	S_1	0	1

State Assignment:

$S_0 \longrightarrow 00$

$S_1 \longrightarrow 01$

$S_2 \longrightarrow 10$

$S_3 \longrightarrow 11$

PS			(D_A D_B) NS		O/P
	A	B	A	B	
(S_0)	0	0	0	0	0
(S_0)	0	0	0	1	0
(S_1)	0	1	0	0	0
(S_1)	0	1	1	0	0
(S_2)	1	0	1	1	0
(S_2)	1	0	0	1	0
(S_3)	1	1	0	0	0
(S_3)	1	1	0	1	1

K-Map:

O/P:

AB \ x	0	1
00	0	0
01	0	0
11	0	(1)
10	0	0

$O/p = ABx$

D_A:

AB \ x	0	1
00	0	0
01	0	(1)
11	0	0
10	(1)	0

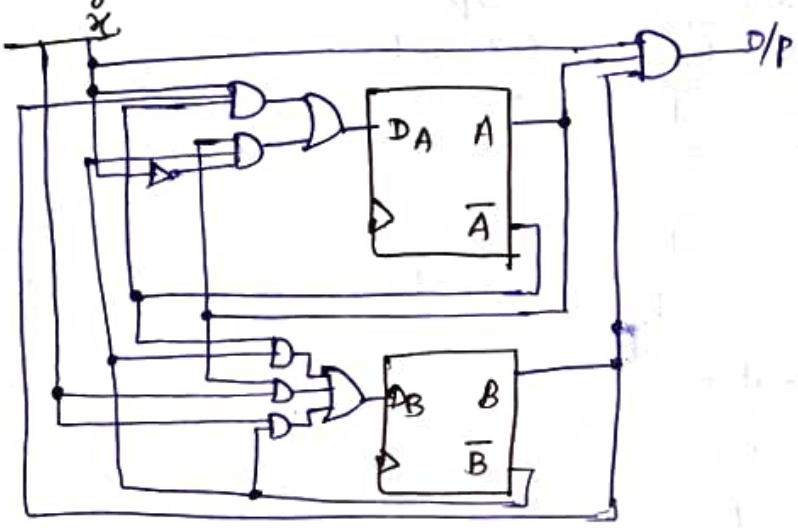
$D_A = \bar{A}Bx + A\bar{B}\bar{x}$

D_B:

AB \ x	0	1
00	0	[1]
01	0	0
11	0	[1]
10	[1]	[1]

$D_B = A\bar{B} + Ax + \bar{B}x$

Circuit diagram:



B-10-2023

Moore Machine

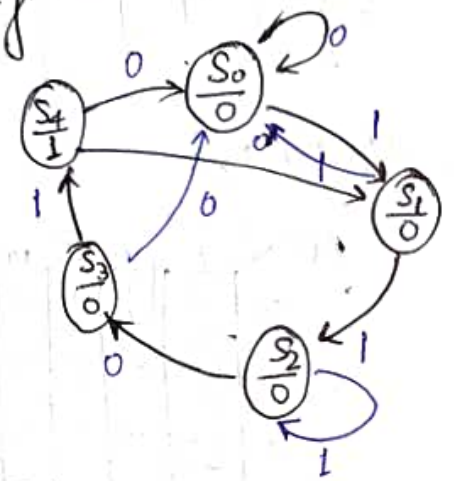
Sequence: 1101

x: 1 0 1 1 1 1 0 1 1 0 1 X

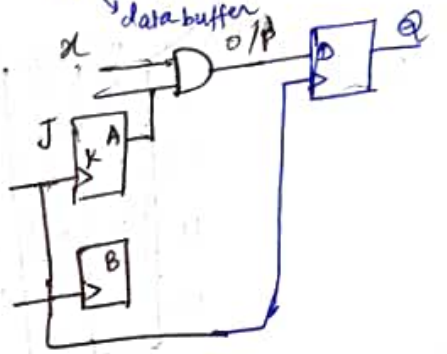
Z: 0 0 0 0 0 0 0 0 1 0 0 ↓ non-overlapping
1 for overlapping

x: 1101X
Z: 0001

State Diagram:



To convert a Mealy machine to Moore, add a DFF to o/p.



State Table:

PS	NS x=0	NS x=1	O/P x=0	O/P x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_4	0	0
S_4	S_0	S_1	0	1

State Assignment: (Binary)

S_0	000 (ABC)
S_1	001
S_2	010
S_3	011
S_4	100

PS(ABC)	x	NS ($D_A D_B D_C$)	O/P	D_A	D_B	D_C
(S_0) 000	0	000	0	0	0	0
(S_0) 000	1	001	0	0	0	1
(S_1) 001	0	000	0	0	0	0
(S_1) 001	1	010	0	0	1	0
(S_2) 010	0	011	0	0	1	1
(S_2) 010	1	010	0	0	1	0
(S_3) 011	0	000	0	0	0	0
(S_3) 011	1	100	0	1	0	0
(S_4) 100	0	000	1	0	0	0
(S_4) 100	1	001	1	0	0	1

K-Map for O/p :

AB \ Cx	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	X	X

O/p = A

K-Map for D_A :

AB \ Cx	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	0	0	X	X

$D_A = B\bar{C}x$

K-Map for D_B :

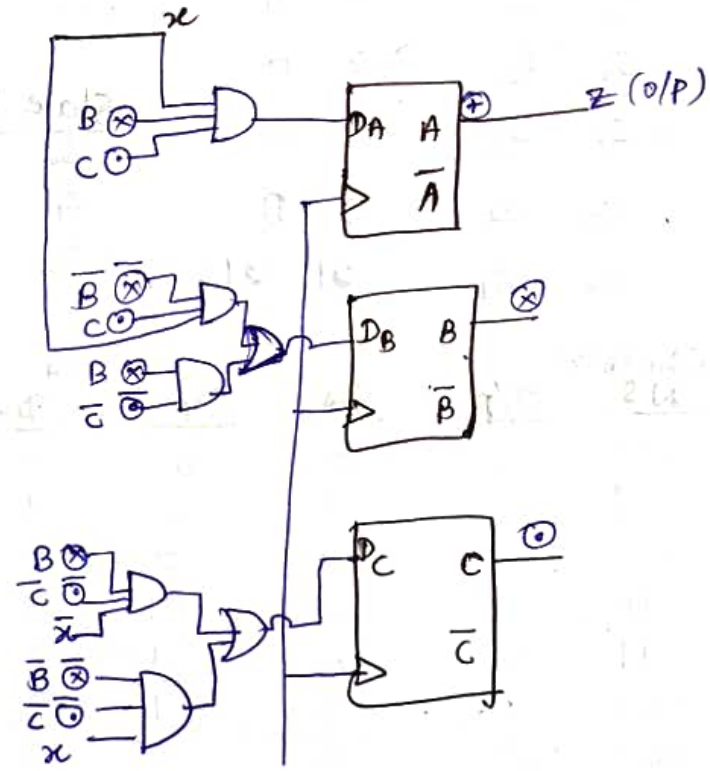
AB \ Cx	00	01	11	10
00	0	0	1	0
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

$D_B = B\bar{C} + \bar{B}Cx$

K-Map for D_C :

AB \ Cx	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

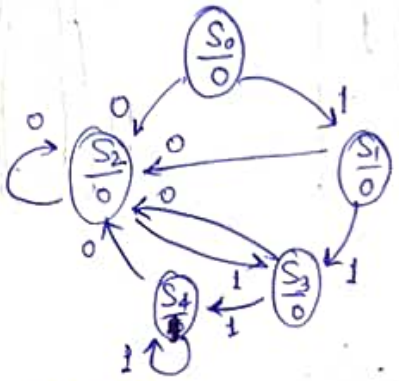
$D_C = B\bar{C}\bar{x} + \bar{B}\bar{C}x$



Sequence: 111 (overlapping)

Q] Obtain a sequence detector to detect the sequences 111 and 011 using JK FF (overlapping) in Moore Machine.

Soln:



State Table:

PS	NS x=0	NS x=1	O/P x=0	O/P x=1
S_0	S_2 S_1	S_1	0	0
S_1	S_2 S_1	S_3	0	0
S_2	S_2 S_1	S_3	0	0
S_3	S_2 S_1	S_4	0	0
S_4	S_2 S_1	S_4	1	1

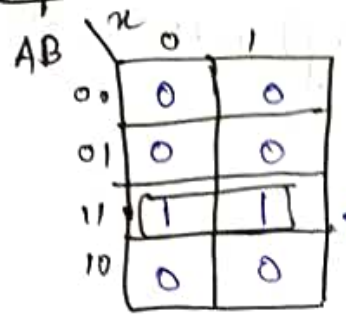
State Assignment

S_0	000
S_1	001
S_3	010
S_4	11

PS(AB)	x	NS	O/P	J_A K_A	J_B K_B
S_0 000	0	01	0	0 X	1 X
S_0 00	1	01	0	0 X	1 X
S_1 01	0	01	0	0 X	X 0
S_1 01	1	10	0	1 X	X 1
S_3 10	0	01	0	X 1	1 X
S_3 10	1	11	0	X 0	1 X
S_4 11	0	01	1	X 1	X 0
S_4 11	1	11	1	X 0	X 0

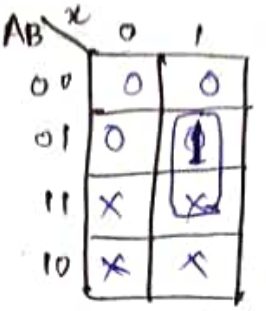
K-Map:

Op:



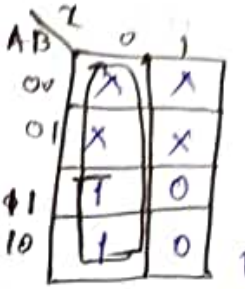
Op = AB

J_A :



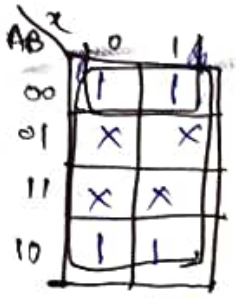
$J_A = Bx$

K_A:



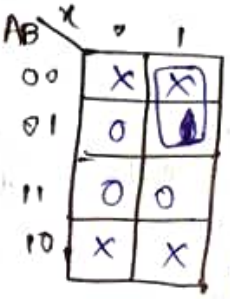
$K_A = x$

J_B:

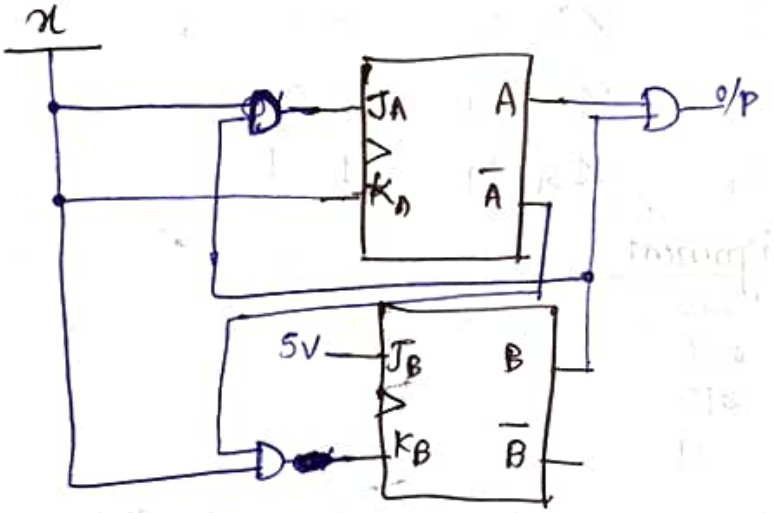


$J_B = 1$

K_B:



$K_B = \bar{A}x$

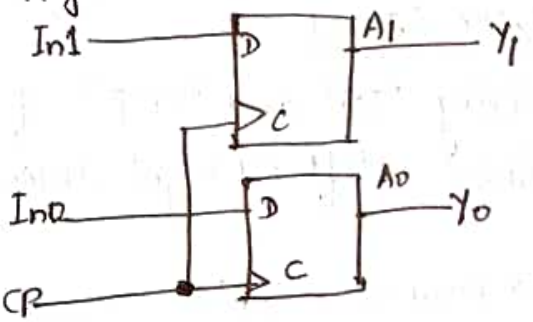


REGISTERS AND COUNTERS

Registers

- A group of binary cells suitable for holding binary info.
- Group of FF constitutes a register which is a sequential circuit.
- Registers handle simple data storage and data movement during processing operations.
- A counter is essentially a register that goes through a predetermined sequence of binary states.

Eg 2 bit Register



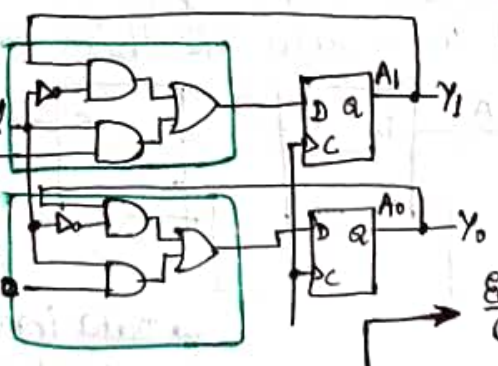
1 bit register → require 8 FFs

Register Storage

- A register can store information for multiple clock cycles.
- External control signals used to load and store information in registers.
- Load signal controls register storage and loading
  - Load=1: Load the values on the data inputs.
  - Load=0: Store the values in the register.

Load=0: holding past info
Load=1: Ready to store new info.

2-bit register with Load control:
For Load=0, loads register contents (holds current values)
For load=1, loads input values (load new values)



Registers with Clock Gating

- The Load signal enables the clock signal to pass through if 1 and prevents the clock signal from processing through if 0.

Eg. For +ve Edge-Triggered or -ve Pulse Master-Slave FF:

- Gated Clock = Clock + $\overline{\text{Load}}$

Eg

Shift Register

- A register capable of shifting its binary information in one or both direction
- All flip-flops receive common clock pulses, which activate the shift from one stage to the next.
- The simplest possible shift register is one that uses only flip-flops.

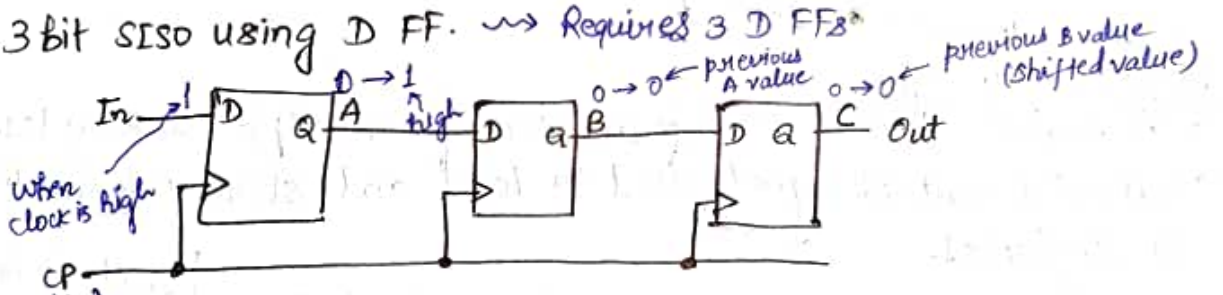
Shift Register Types :

- A register capable of shifting its binary information either to the right or to the left is called shift register.

→ **Serial in serial out Register (SISO)**

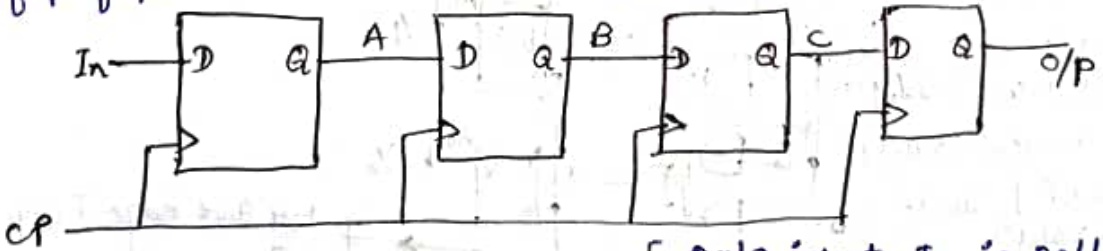
The data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

Eg. 3 bit SISO using D FF. ⇝ Requires 3 D FFs



→ Shift Registers move data laterally within the register toward its MSB or LSB position.

→ In the simplest case, the shift register is simply a set of D flip-flops connected in a row like this:



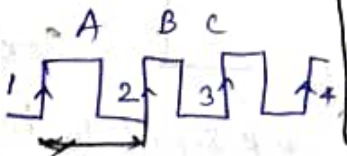
→ Data input, In, is called a serial input or shift right input.

→ Data output, Out, is often called the serial output.

3-bit SISO :

C = O/P

IN: 1010101

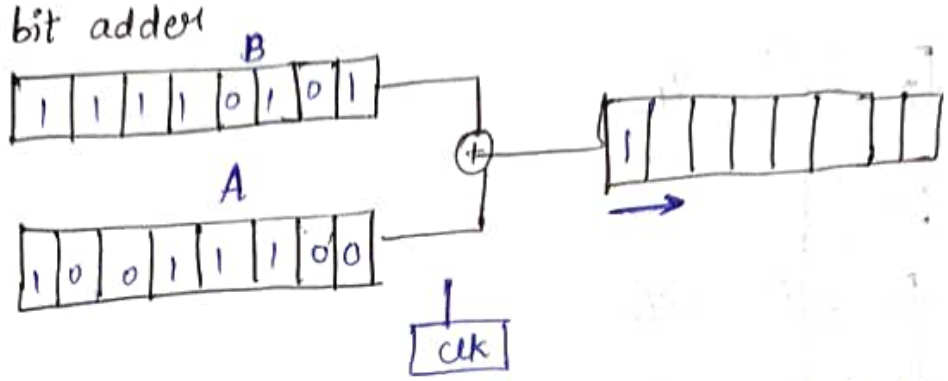


	A	B	C	
	0	0	0	← Initially
1 ↑	1	0	0	← 1st clock cycle
2 ↑	0	1	0	← 2nd clock cycle
3 ↑	1	0	1	

	A	B	C
4 ↑	0	1	0

Parallel IN serial OUT:

↳ 8 bit adder



all controlled by a single clock.

Shift registers:

- Serial IN serial Out
- Serial In Parallel out
- Parallel In Serial out
- Parallel In Parallel Out

SISO:

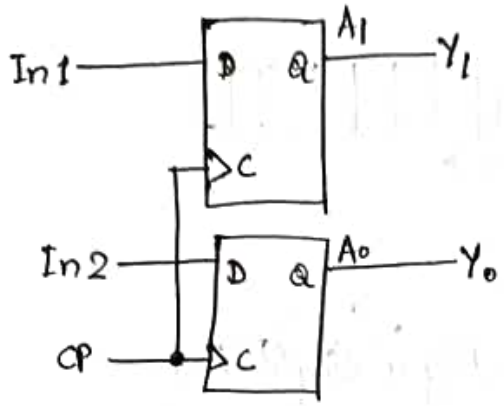
- Each clock pulse shifts the contents of the register one bit position to the right.
- The serial input determines what goes into the leftmost FF during the shift.
- The serial output is taken from the output of the rightmost flip-flop.

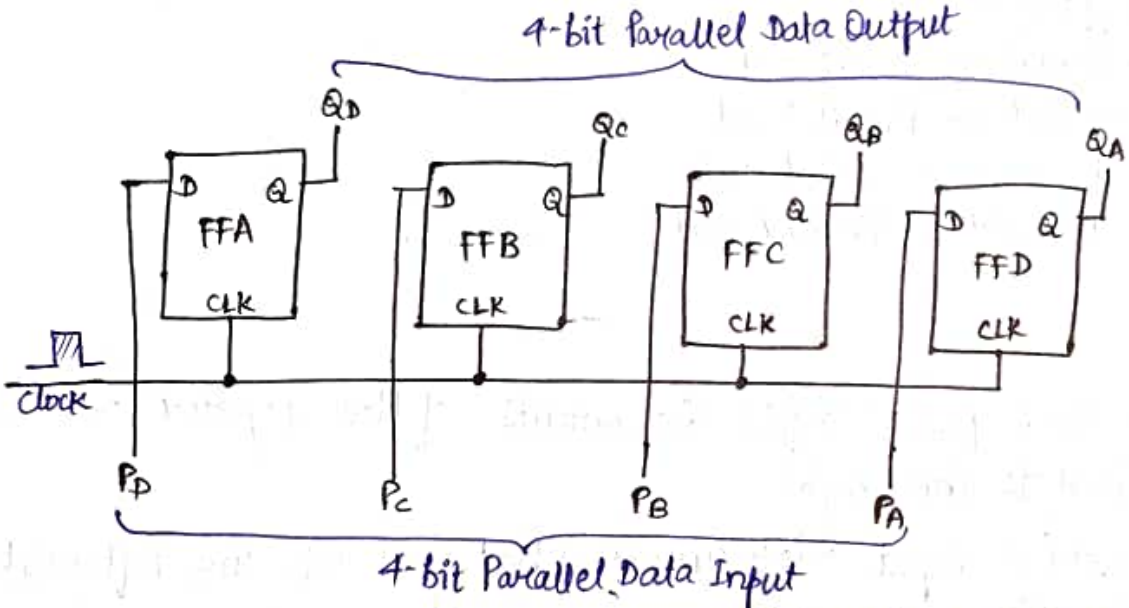
→ Serial-in to Parallel-out (SIPO): The register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.

→ Parallel-in to Parallel-out (PIPO): The parallel data in loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

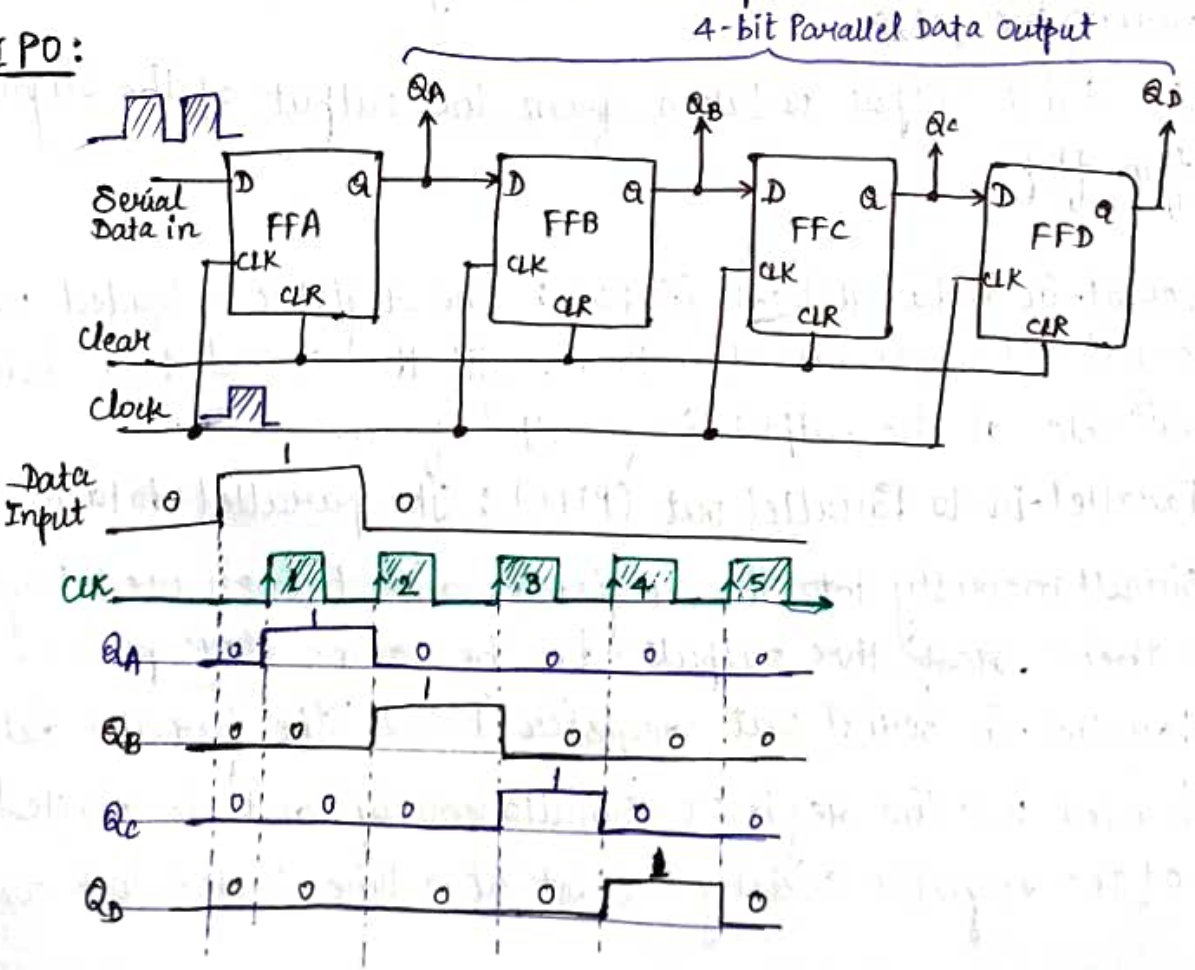
→ Parallel-in serial-out register (PISO): The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

PIPO:


In 1
D
Q
A1
Y1
C
In 2
D
Q
A0
Y0
CP
C

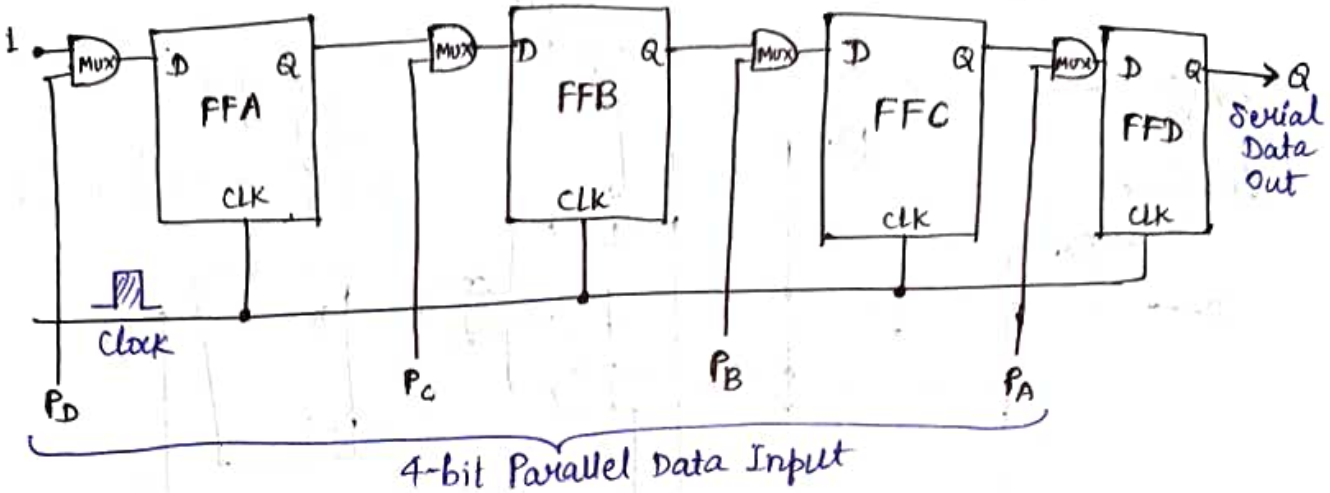

4-bit Parallel Data Output
QD
QC
QB
QA
D
Q
FFA
FFB
FFC
FFD
CLK
Clock
PD
PC
PB
PA
4-bit Parallel Data Input

SIPO:

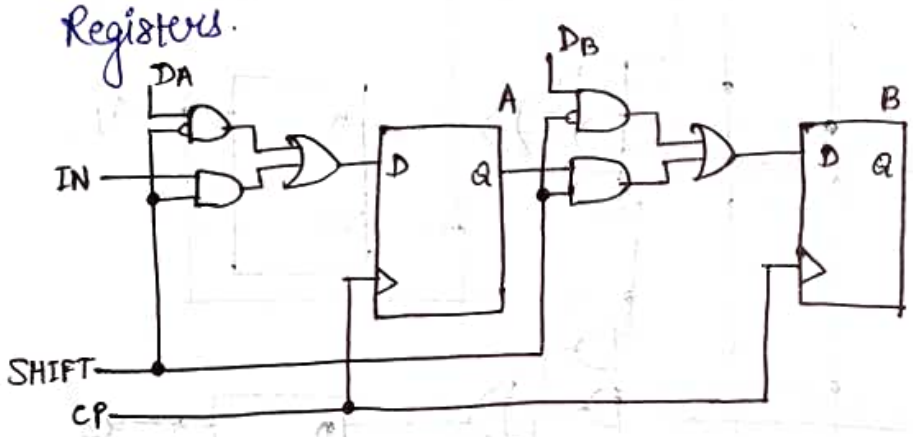

4-bit Parallel Data Output
QA
QB
QC
QD
Serial Data in
D
Q
FFA
FFB
FFC
FFD
CLK
CLR
Clear
Clock
Data Input
CLK
1
2
3
4
5
0
1

06-10-2023

PISO



→ IC's include the 74HC166 8-bit Parallel-In/Serial-out Shift Registers.



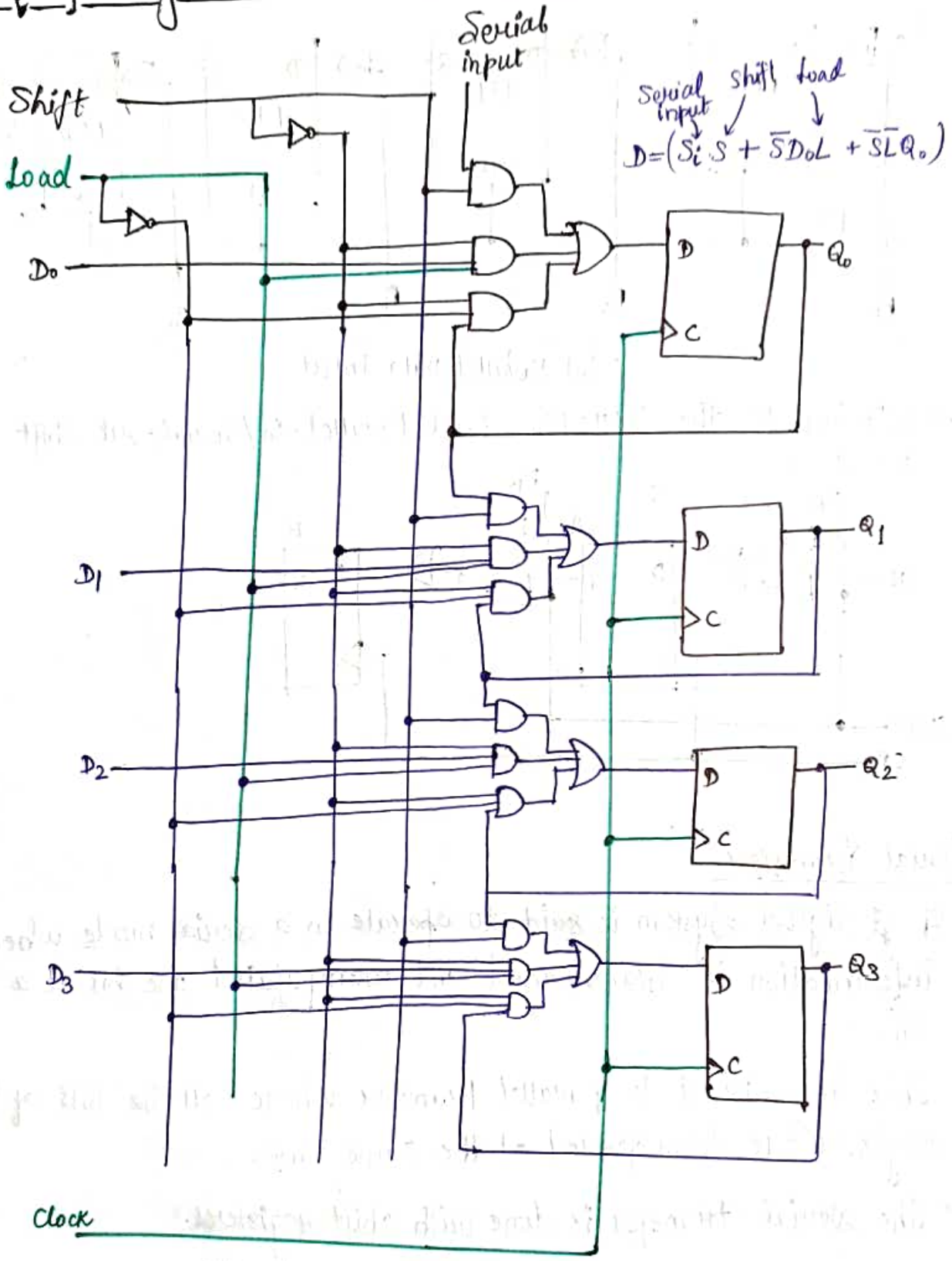
Serial Transfer

→ A digital system is said to operate in a serial mode when information is transferred and manipulated one bit at a time.

→ This in contrast to parallel transfer where all the bits of the register are transferred at the same time.

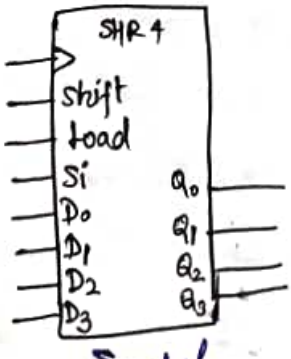
→ The serial transfer is done with shift registers.

Shift Register with Parallel Load



Serial input: S_i, Shift: S, Load: L

$$D = (S_i S + \bar{S} D_0 L + \bar{S} \bar{L} Q_0)$$

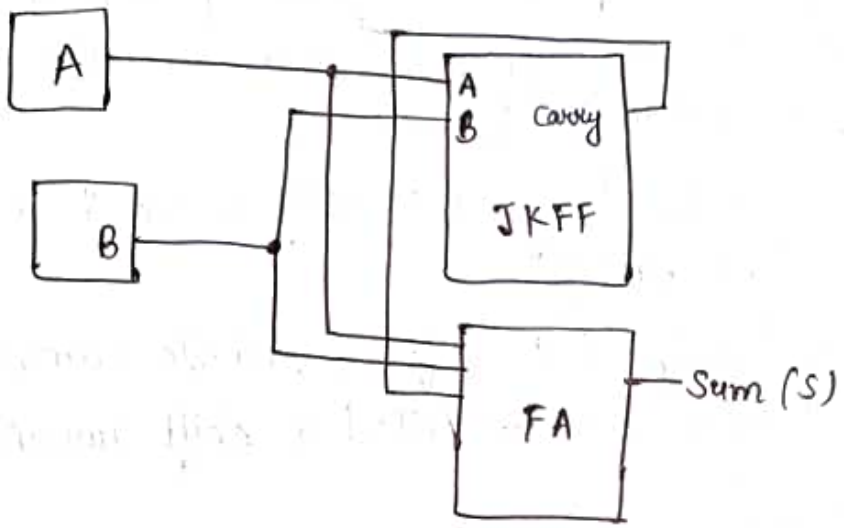


Symbol

Function Table for the Register

Shift	Load	Operation
0	0	No change
0	1	load parallel data
1	X	Shift down from Q_0 to Q_3

4 bit (4+4) Serial (Full) Adder



A	B	(Carry) PS	Sum	$(C)_{out}$ NS	J_c	K_c
0	0	0	0	0	0	X
0	0	1	1	0	X	1
0	1	0	1	0	0	X
0	1	1	0	1	X	0
1	0	0	1	0	0	X
1	0	1	0	1	X	0
1	1	0	0	1	1	X
1	1	1	1	1	X	0

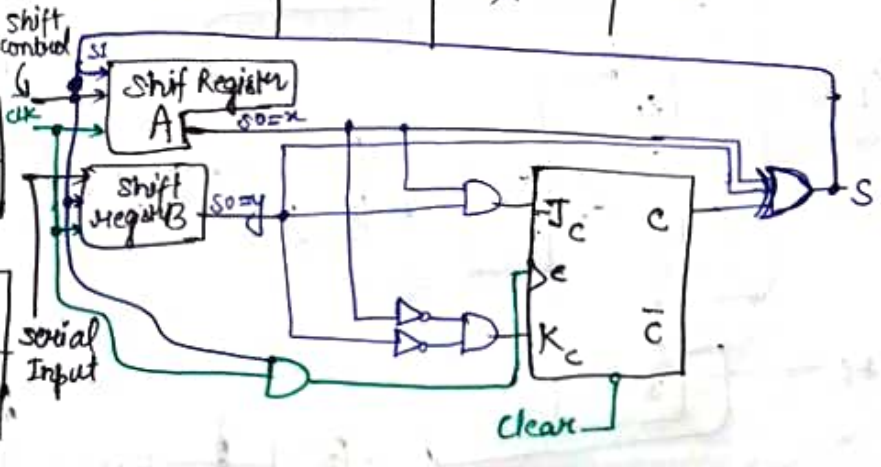
K-Map:

J_c:

A \ BC	00	01	11	10
0	0	X	X	0
1	0	X	X	1

K_c:

A \ BC	00	01	11	10
0	X	1	0	X
1	X	0	0	X



$J_c = AB$

$K_c = \bar{A}\bar{B} = \overline{(A+B)}$

$S = A \oplus B \oplus C$

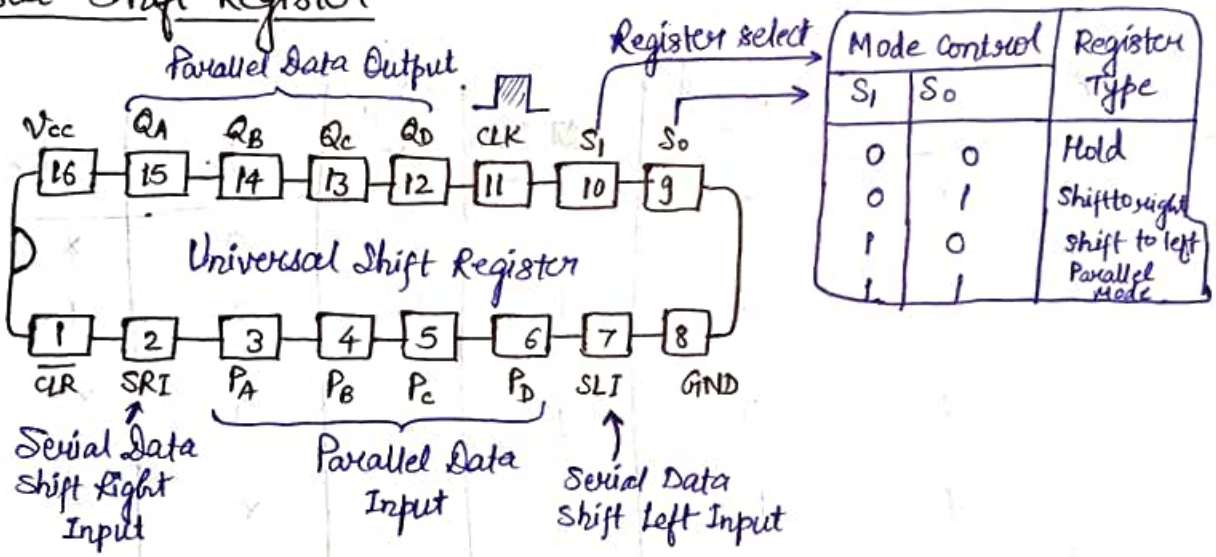
Shift Registers with Additional Functions

→ By placing a 4-input multiplexer in front of each D flip-flop in a shift register, we can implement a circuit with shifts right, shifts left, parallel load, hold.

→ Shift registers can also be designed to shift more than a shift bit position right or left.

→ Shift registers can be designed to shift a variable number of bit positions specified by a variable called a shift amount.

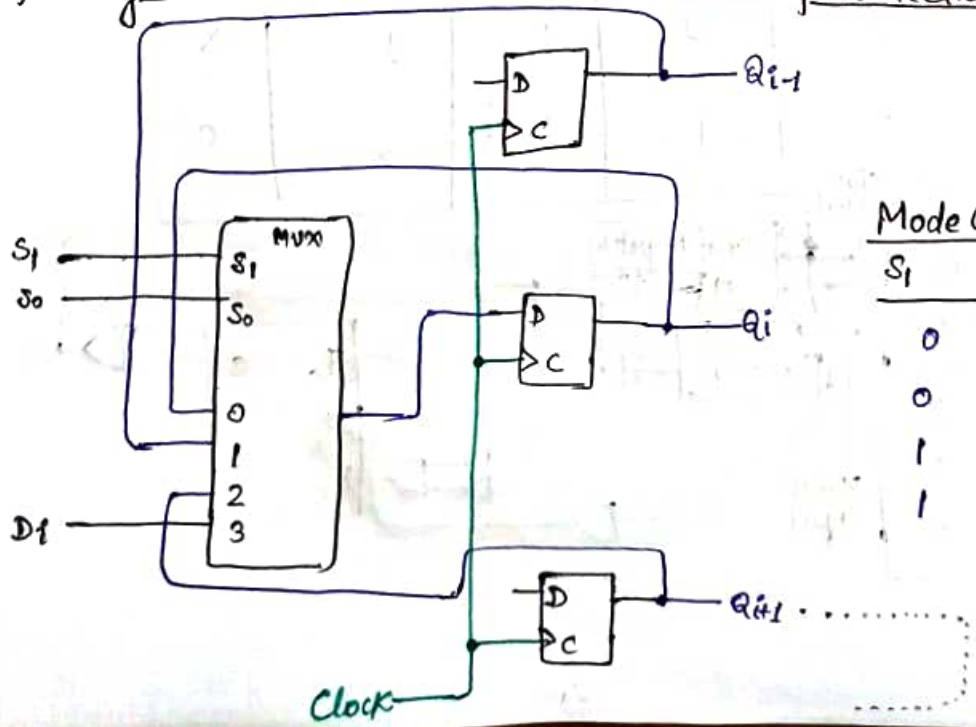
Universal Shift Register



Mode Control		Register Type
S_1	S_0	
0	0	Hold
0	1	Shift to right
1	0	shift to left
1	1	Parallel mode

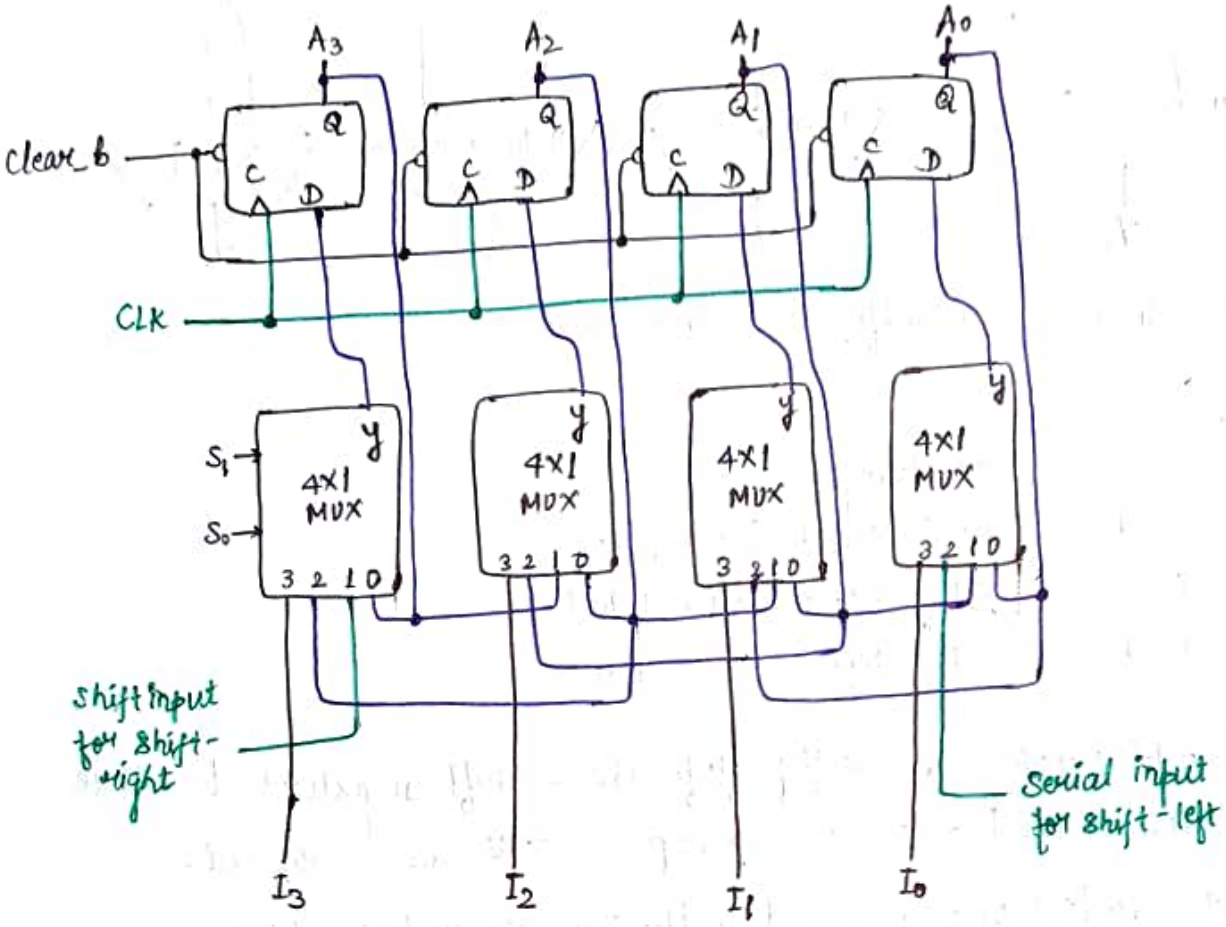
4-bit Universal Shift Register 74LS194

Shift Register with Parallel Load and Shift Direction (Universal)



Mode Control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift down
1	0	Shift up
1	1	Parallel load

Universal Shift Register

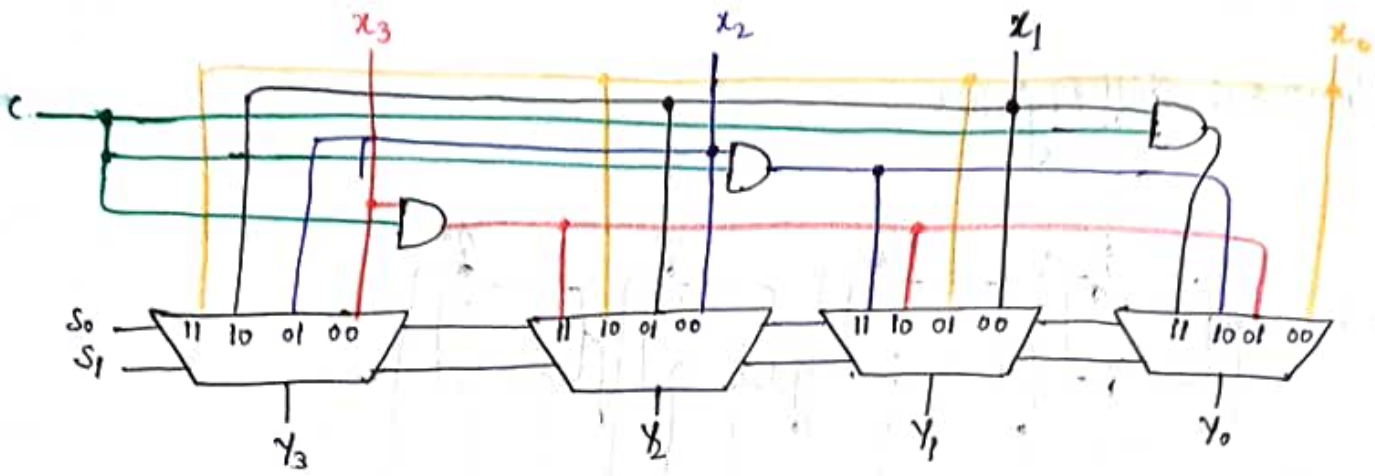


S_1	S_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

Barrel Shift Register

→ Shifting and rotating data is required in several applications including arithmetic operations.

→ Shift right logical, shift right arithmetic, rotate right, shift left logical, shift left arithmetic, and rotate left.



If c=0, then it's a shift, c=1 a rotate

S_0	S_1	
0	0	no change
0	1	shift left 1 (or rotate)
1	0	shift left 2 (or rotate)
1	1	shift left 3 (or rotate)

→ Serial Adder using a JK flip-flop use 2 shift registers to store two numbers and one JK flip-flop to store the carry out.

→ Bits are added one pair at a time using a full adder.

- The carry out is transferred to a JK FF.
- The result will be stored in shift register A.
- The circuit can be used to sum a set of numbers.
- Register B can be used to transfer a new binary number.
- Initially, register A and carry flip-flop are cleared to 0.

12-10-2023

COUNTERS

→ Counters are sequential circuits which "count" through a specific state sequence. They can count up, count down, or count through other ~~specify~~ fixed sequences.

Two distinct types are in common usage:

① Ripple Counters / Asynchronous Counters:

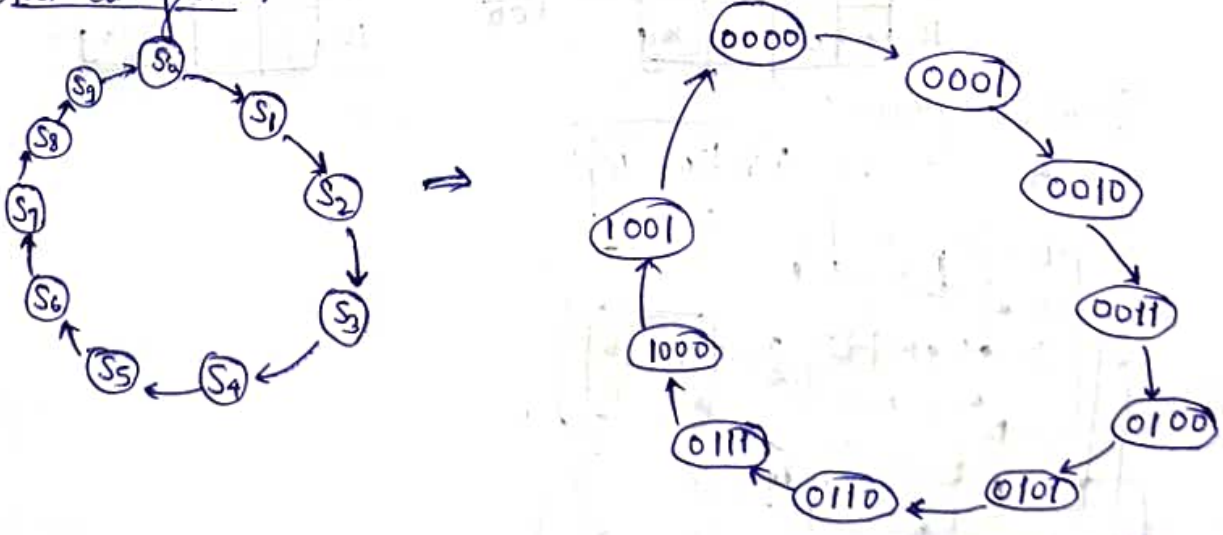
- Clock connected to the flip-flop clock input on the LSB bit flip-flop.
- For all other bits, a flip-flop output is connected on the clock input, thus circuit is not truly synchronous!
- Output change is delayed more for each bit towards the MSB.
- Resurgent because of low power consumption.

② Synchronous Counters:

- Clock is directly connected to the flip-flop clock inputs.
- Logic is used to implement the desired state sequencing.
- Faster than asynchronous counters.

Eg. Design 0 to 9 counters using D FF.

State diagram:

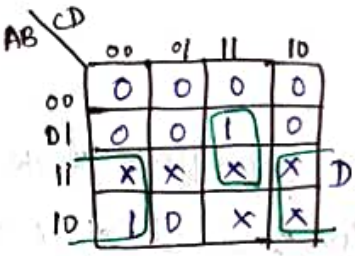


State table:

	PS				D_A D_B D_C D_D NS			
	A	B	C	D	A	B	C	D
(S_0)	0	0	0	0	0	0	0	1
(S_1)	0	0	0	1	0	0	1	0
(S_2)	0	0	1	0	0	0	1	1
(S_3)	0	0	1	1	0	1	0	0
(S_4)	0	1	0	0	0	1	0	1
(S_5)	0	1	0	1	0	1	1	0
(S_6)	0	1	1	0	0	1	1	1
(S_7)	0	1	1	1	1	0	0	0
(S_8)	1	0	0	0	1	0	0	1
(S_9)	1	0	0	1	0	0	0	0

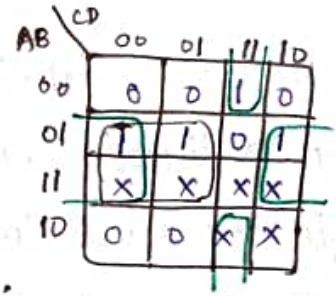
K-Map:

D_A:



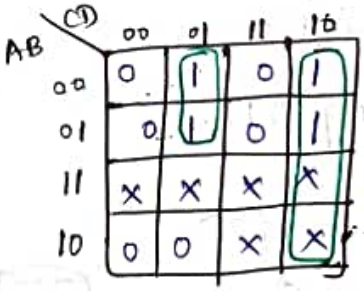
$D_A = B\bar{C}D + A\bar{D}$

D_B:



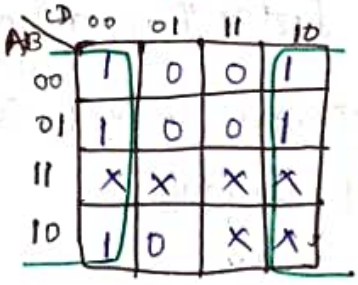
$D_B = B\bar{C} + B\bar{D} + \bar{B}CD$

D_C:



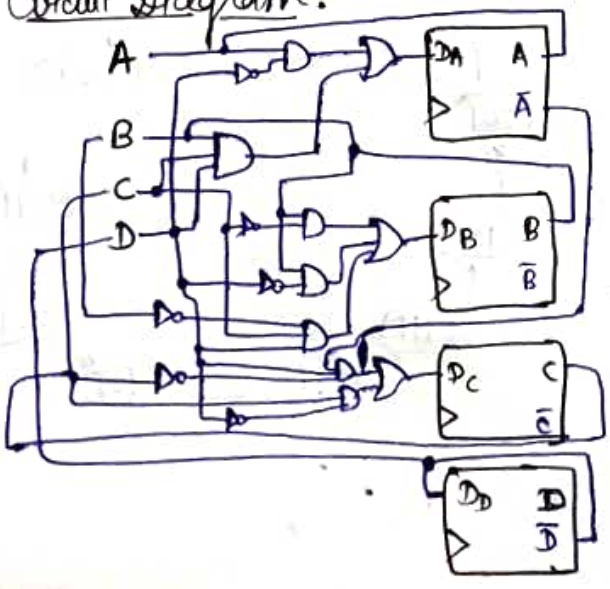
$D_C = \bar{A}\bar{C}D + C\bar{D}$

D_D:

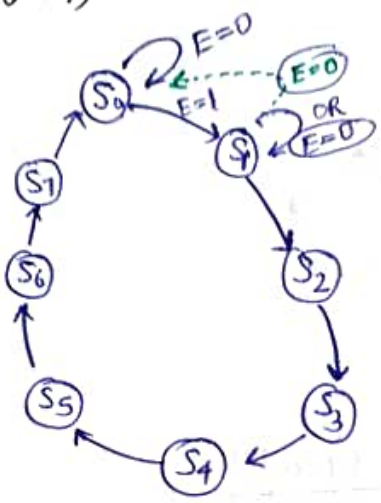


$D_D = \bar{D}$

Circuit Diagram:

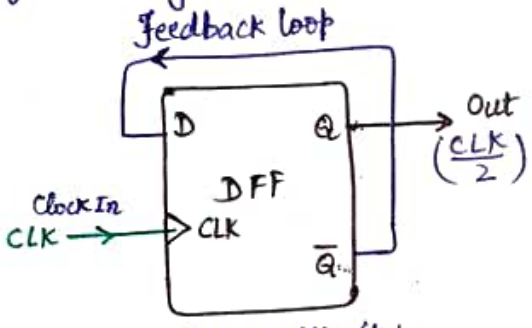


Eg 3-bit (0→7) Counter with Enable Pin

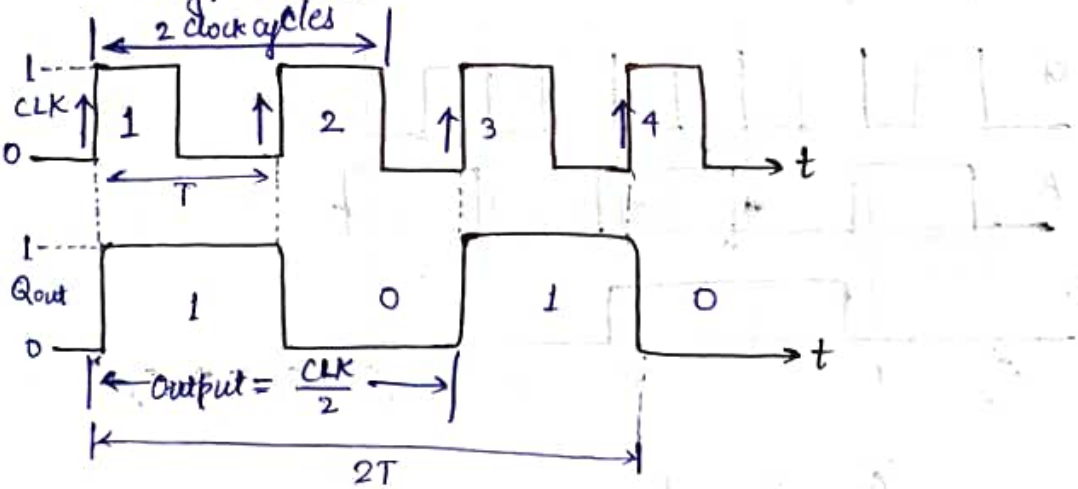


Clock Frequency Divider (Divide-by-two Counter)

↳ A useful feature of the D-type FF is as a binary divider, for frequency division or as a "divide-by-2" counter



D-type flip flop

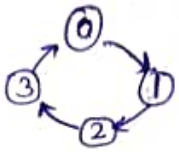


↳ Divides the clock frequency by 2: $\left(\frac{f_{CLK}}{2}\right)$

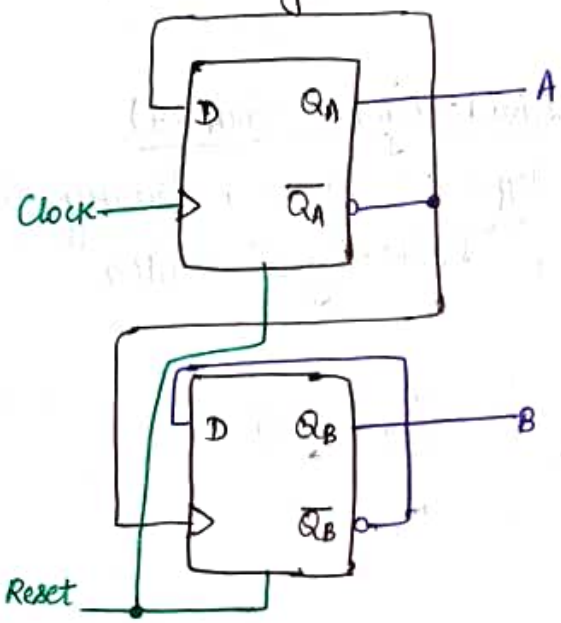
Eg 2-bit signal

MSB	LSB
0	0
0	1
1	0
1	1

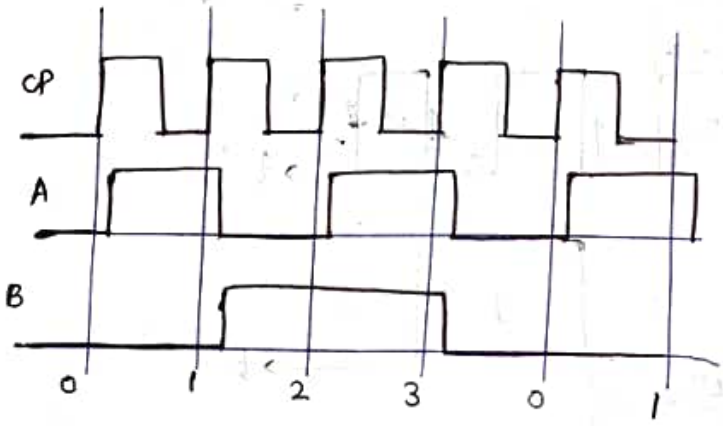
f_{CLK} / 2



Ripple Counter — Asynchronous Counter

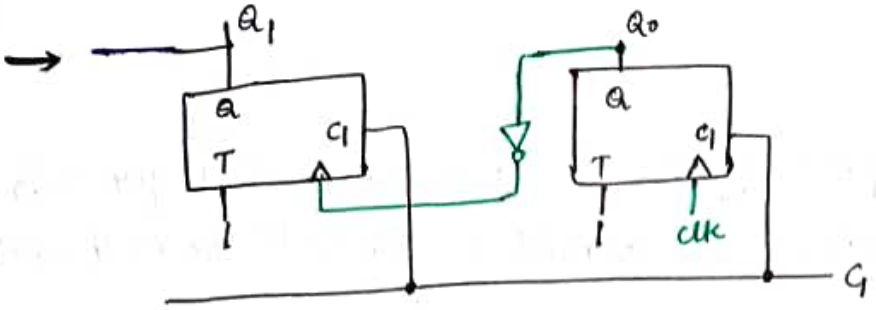


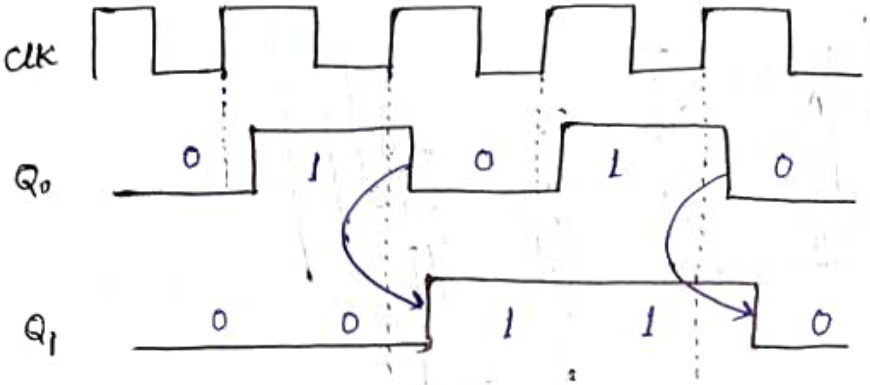
(Initially, $Q_A = Q_B = 0$)



	$\overline{A}$	A	B	
	1	0	0	← Initially
CLK ↑	0	1	0	
↑	1	0	1	
↑	0	1	1	
		↓	↓	
		$f_{CLK}/2$	$f_{CLK}/4$	

(Clock is Positive transition
B changes its value)

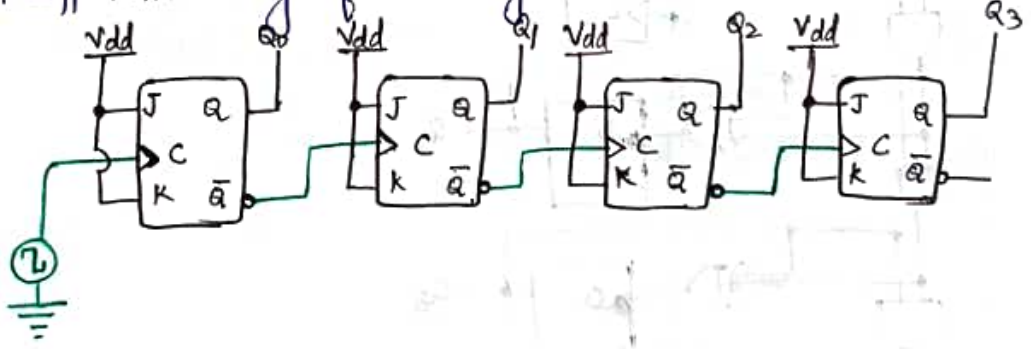


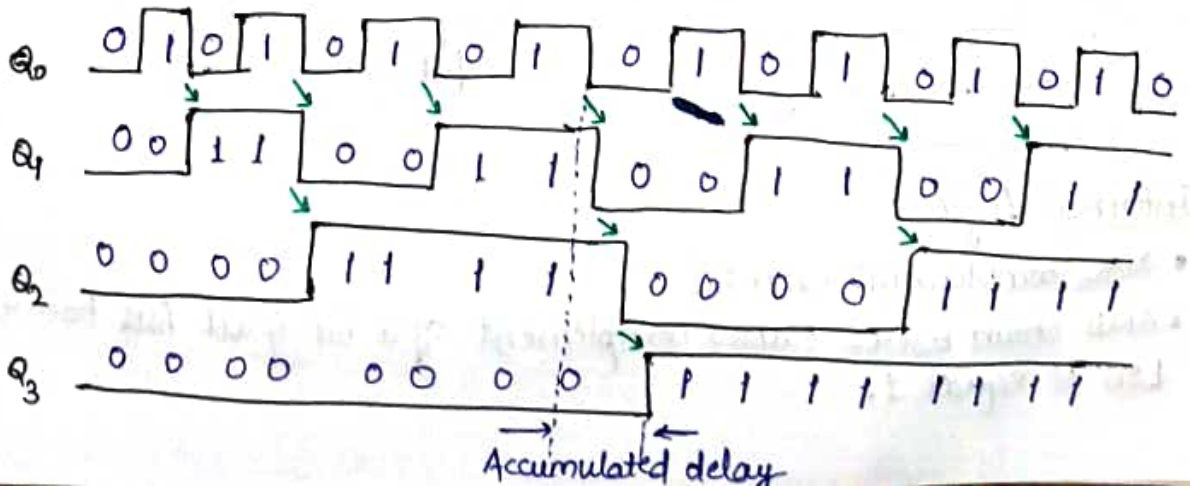


Ripple Counter

- These circuits are called ripple counters because each edge sensitive transition (positive in the example) causes a change in the next flip-flop's state.
- The changes "ripple" upward through the chain of flip-flops, i.e., each transition occurs after a clock-to-output delay from the stage before.

→ A different way of making a 4-bit "up" counter.

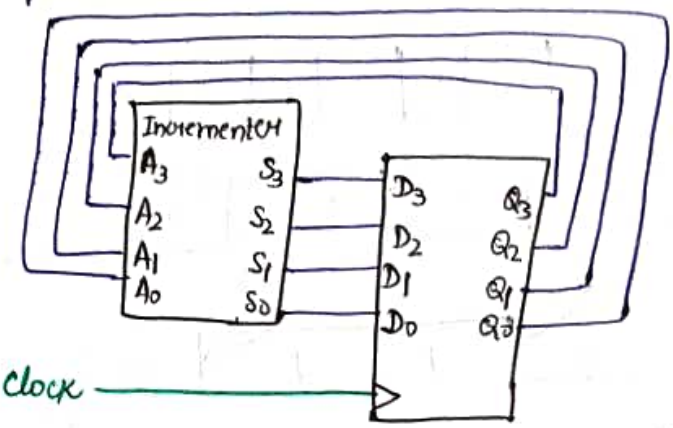




Synchronous Counters

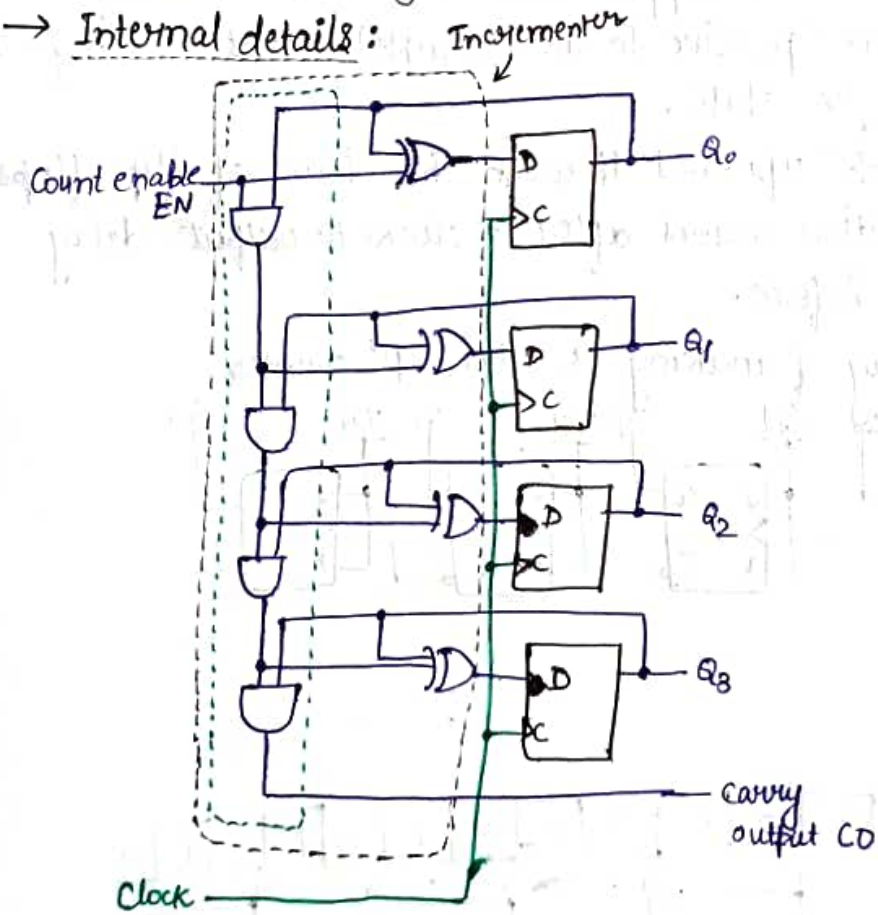
→ To eliminate the "ripple" effects, use a common clock for each flip-flop and a combinational circuit to generate the next state.

→ For an up-counter, use an incrementer.



Note: Synchronous counter design is same as that of sequential circuit design.

→ Internal details:



→ Internal Logic:

- XOR complements each bit.
- AND chain ~~uses~~ causes complement of a bit if all bits toward LSB is equal 1.

→ Count Enable:

- Forces all outputs of AND chain to 0 to "hold" the state.

→ Carry Out:

- Added as part of incrementer
- Connect to Count Enable of additional 4-bit counters to form larger counters.

13-10-2023

Mod N Counter

→ A modulo N counter goes through a repeated sequence of N counts which required N number of Flip Flops.

→ The "Modulo" or "Modulus of a counter is the number of states the counter counts or sequences through before repeating itself.

Eg A 3-bit binary counter is a Mod 8.

A BCD counter is a Mod 10 counter.

→ MOD counters have a modulus value that is an integral power of 2.

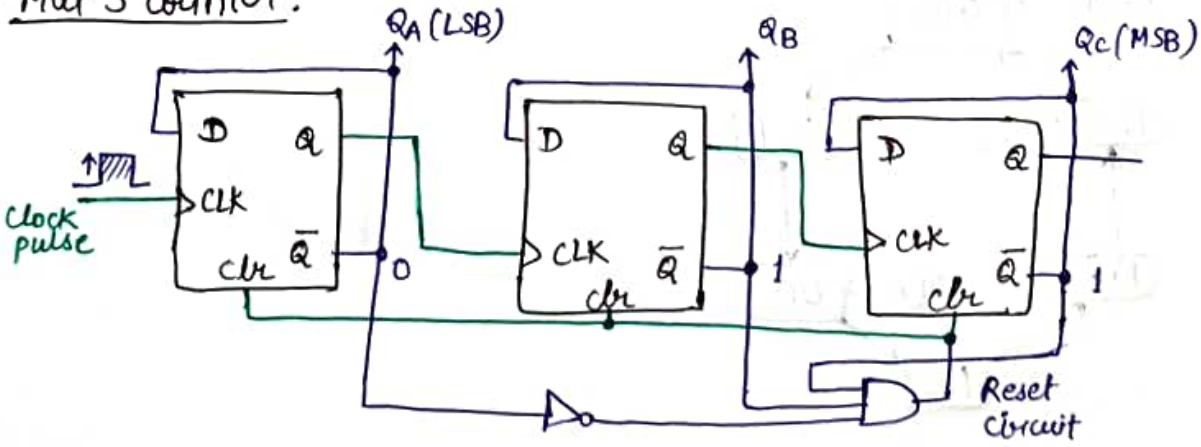
$$2^n \geq N$$

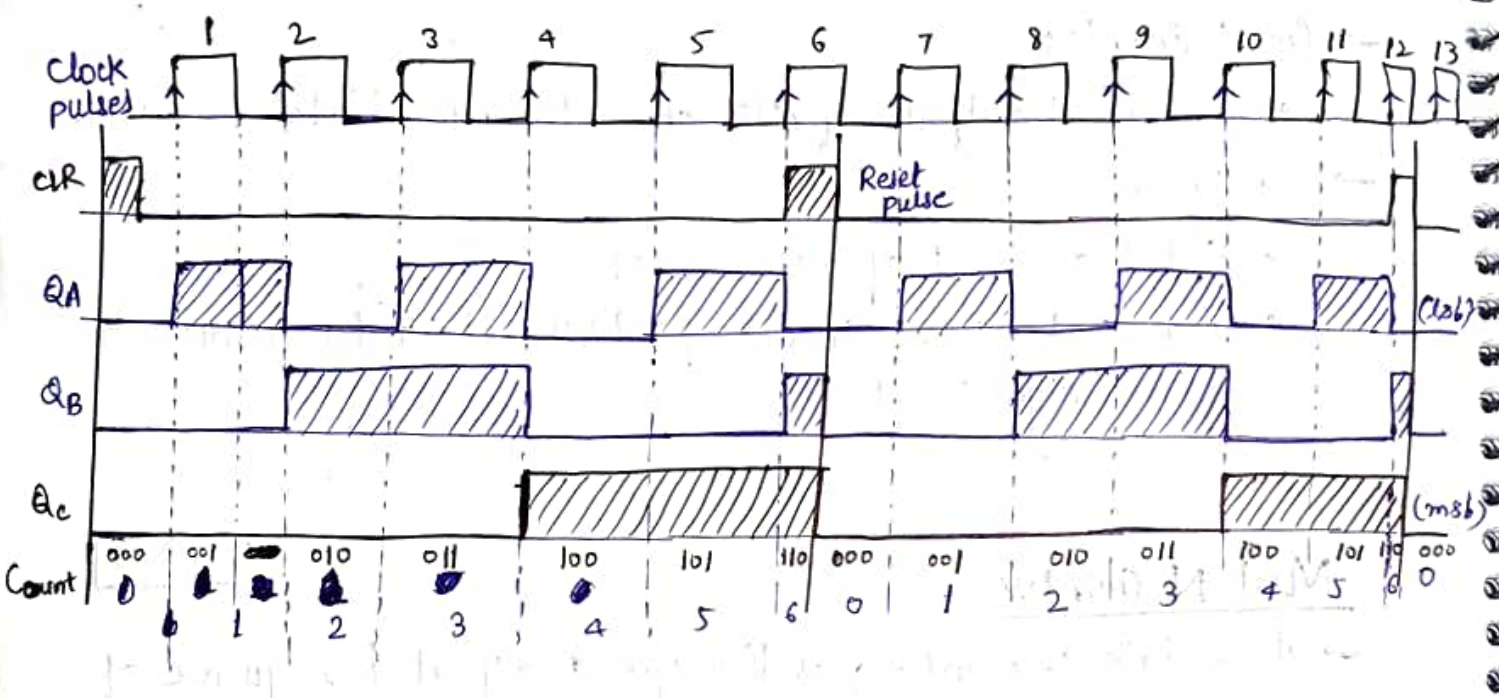
for Mod 10 counter,

min^m no. of FF required = 4

$(2^4 = 16 \geq 10)$.

→ Mod-5 counter:



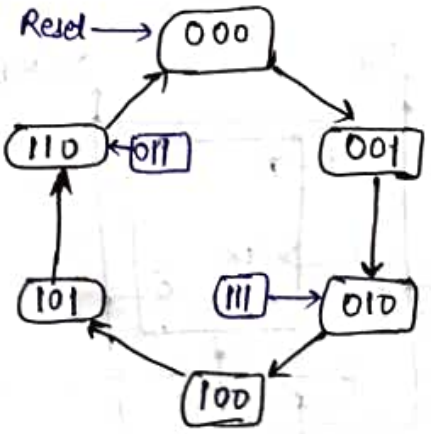


Arbitrary Count

→ Counter goes through an arbitrary sequence.

Eg.

Present state			Next state		
A	B	C	$D_A = A(t+1)$	$D_B = B(t+1)$	$D_C = C(t+1)$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

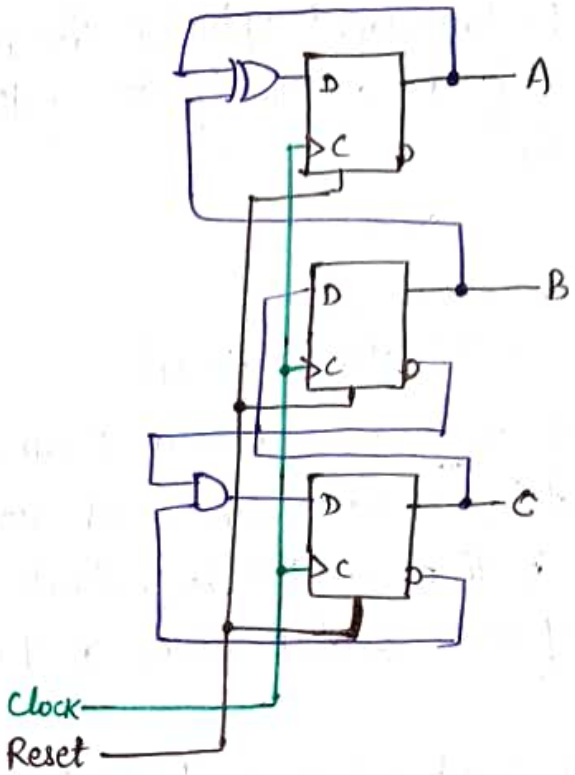


Circuit:

$D_A = (A \oplus B)$

$D_B = C$

$D_C = (\bar{B} \cdot \bar{C})$



Design example: Synchronous BCD

- Use the sequential logic model to design a synchronous BCD counter with D flip-flops.
- State Table:

Current State Q_8 Q_4 Q_2 Q_1	Next State Q_8 Q_4 Q_2 Q_1
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

- Input combinations 1010 through 1111 are don't cares.

- Use K-Maps to two-level optimize the next state equations and manipulate its form containing XOR gates:

$$D_1 = \overline{Q_1}$$

$$D_2 = Q_2 \oplus Q_1 \overline{Q_8}$$

$$D_4 = Q_4 \oplus Q_1 Q_2$$

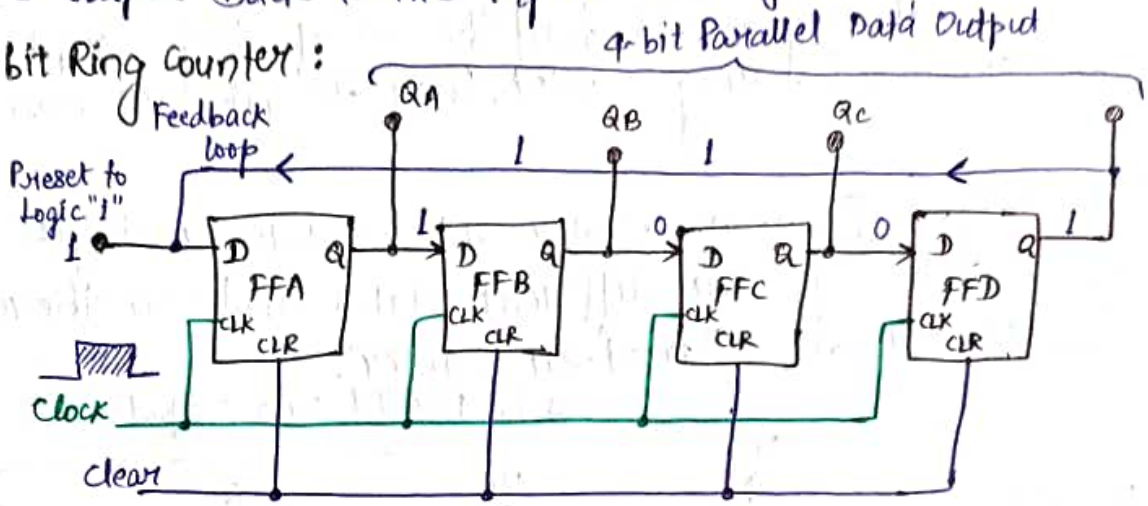
$$D_8 = Q_8 \oplus (Q_1 Q_8 + Q_1 Q_2 Q_4)$$

- The logic diagram can be drawn from these equations. Anasynchronous or synchronous reset should be added.
- What happens if the counter is perturbed by a power disturbance or other interference and it enters a state other than 000 through 1001?
- For the BCD counter design, if an invalid state is entered, return to a valid state occurs within two clock cycles.
- Is this adequate? If not:
  - Is a signal needed that indicates than an invalid state has been entered? What is the equation for such a signal?
  - Does the design need to be modified to return from an invalid state to a ~~valid~~ specific state ~~in one clock cycle?~~ (such as 0)?
  - Does the design need to be modified to return from an invalid state to a valid state in one clock cycle?
- The action to be taken depends on:
  - the application of the circuit.
  - design group policy.

Ring Counter

↳ A standard shift register circuit into a ring counter by looping the output back to the input. (Initially o/p is set 1)

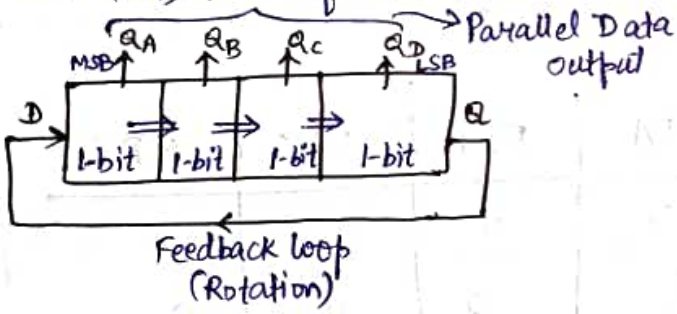
Eg: 4-bit Ring Counter :



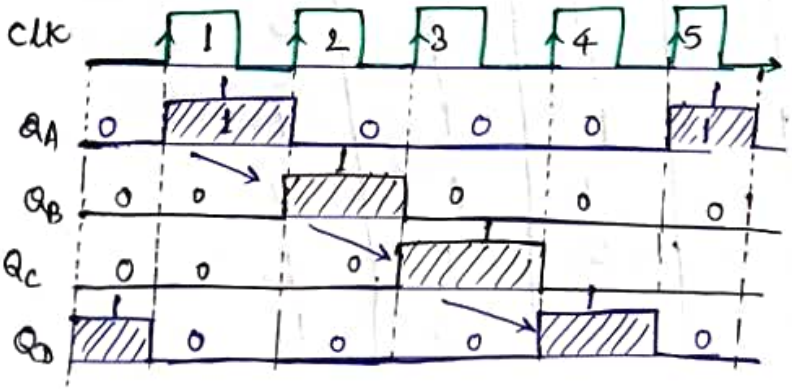
Rotational Movement of Shift register

↳ It has 4 distinct states

↳ Also known as "modulo-4" or "mod-4" counter with each flip-flop output having a frequency value equal to one-fourth or a quarter (1/4) that of the main clock frequency.



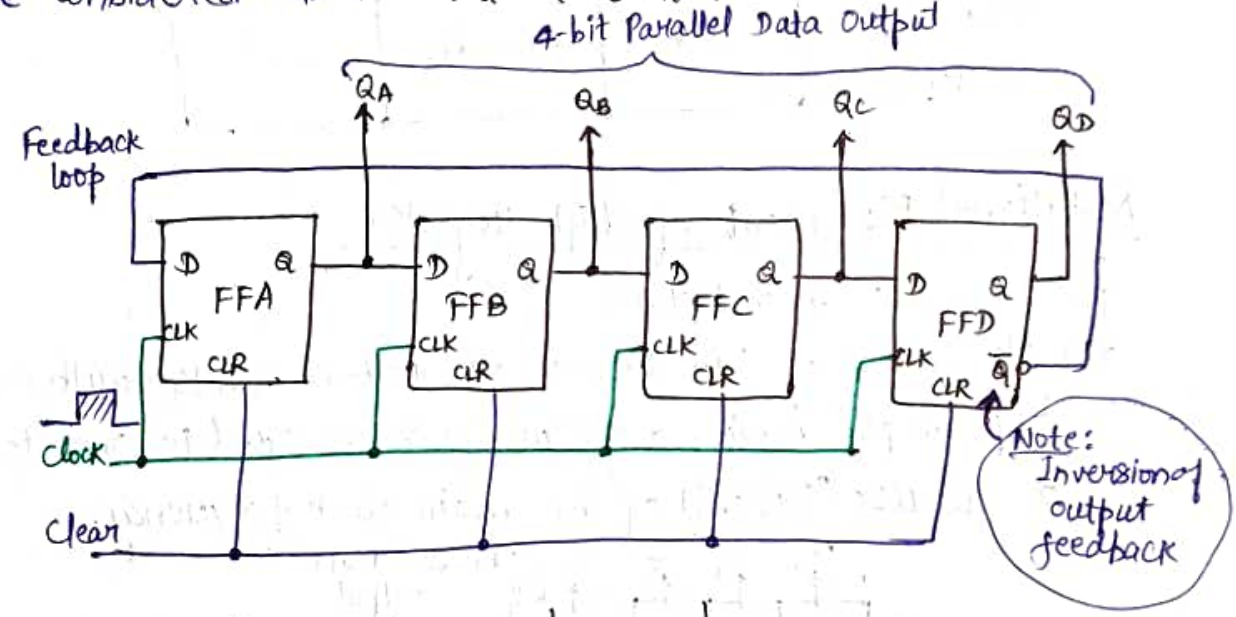
Set, Reset inputs ⇓ Asynchronous inputs



Johnson Ring Counter / Twisted Ring Counters

- In this, the inverted output $\bar{Q}$ of the last flip-flop is now connected to the input D of the first flip-flop.
- Advantage: It only needs half the no. of flip-flops compared to the standard ring counter, then its modulo number is halved.
- A "n-stage" Johnson counter will circulate a single data bit giving sequence of $2n$ different states and can therefore be considered as a "mod-$2n$ counter".

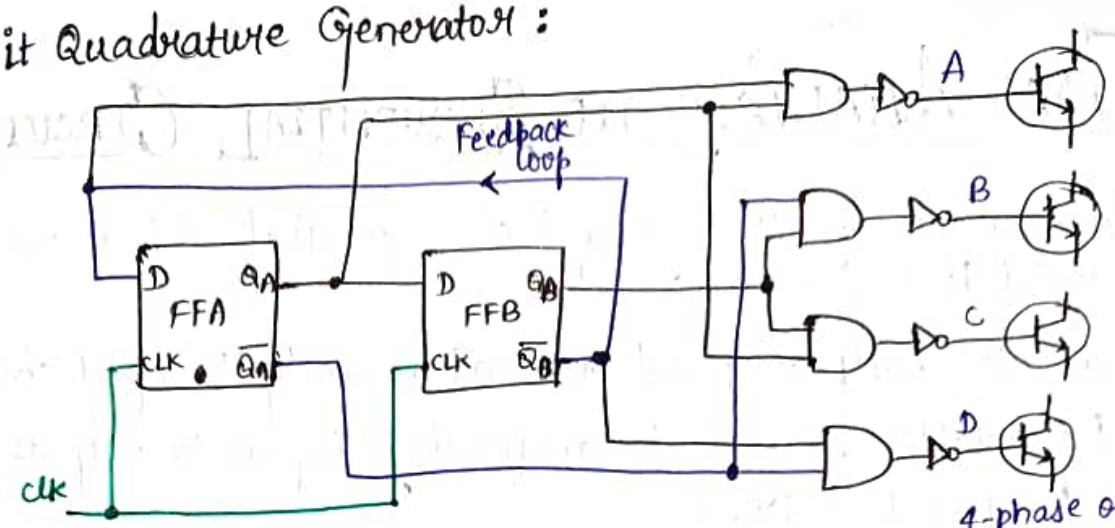
Eg.



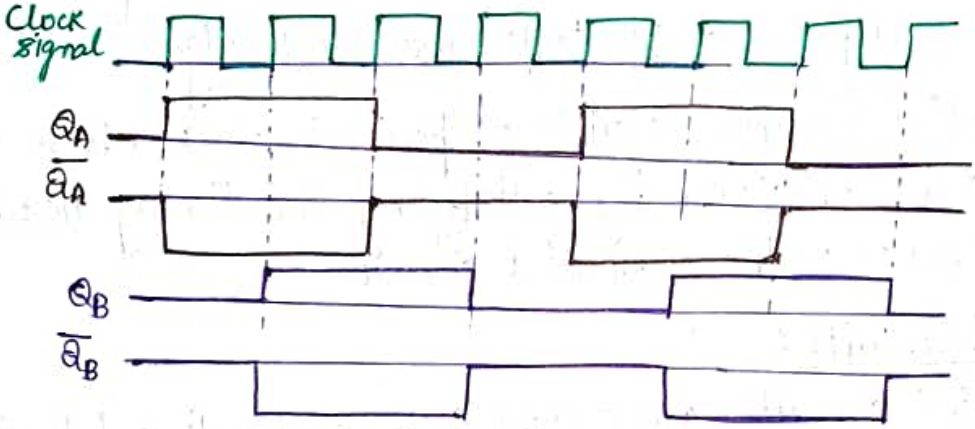
Clock Pulse No.	FFA	FFB	FFC	FFD
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

Application: To divide the frequency of the clock signal by varying their feedback connections — Mod-3, Mod-5 counters.

→ 2-bit Quadrature Generator :



4-phase output for Bi-polar, Unipolar or Universal Motors



- A to D are phase shifted by 90° with each other
- can be used to position control or the ability to rotate a motor to a particular location with some additional circuitry.

→ Duty cycle = $\frac{t_{ON} + t_{OFF}}{2}$, t_{ON} & t_{OFF} : same for symmetric waveforms.

TIME ANALYSIS IN SEQUENTIAL CIRCUITS

→ The correct functioning of the sequential circuit involves several timing issues:

- Circuits don't respond instantaneously to input changes. Predictable delay in transferring inputs to outputs — propagation delay.
- Sequential circuits require certain amount of time the input need to be stable/after the clock edge.
- Sequential circuits require a periodic clock whose period need to be obtained by satisfying the timing needs of the components in the sequential circuit.

Sequential Circuits:

→ Sequential circuits can contain both combinational logic and edge-triggered flip-flops.

→ A clock signal determines when data is stored in flip-flops.

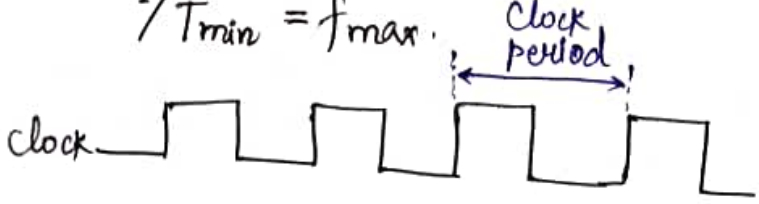
→ Goal: How fast can the circuit operate?

Minimum clock period : T_{min}

Maximum clock frequency: f_{max}.

→ Maximum clock frequency is the inverse of the minimum clock period.

$$1/T_{min} = f_{max}$$



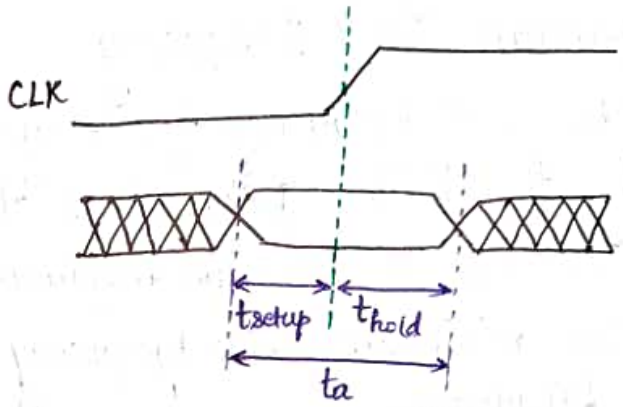
Input Timing Constraints

→ **Setup time**: t_{setup} = time before the clock edge that data must be stable (i.e., not changing).

→ **Hold time**: t_{hold} = time after the clock edge that data must be stable.

→ **Aperture time**: t_a = time around clock edge that data must be stable.

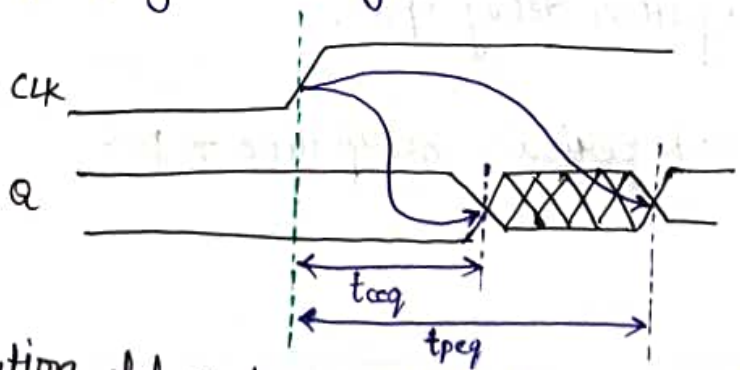
$t_a = t_{setup} + t_{hold}$



Output Timing Constraints

→ **Propagation delay**: t_{pcq} = max^m time after clock edge by which output Q is guaranteed to have stabilized (i.e., not changing anymore).

- Propagation delays for transition from Low to High and High to Low are different/same.
  t_{phl} is the propagation delay during output transition from high to low.
- Any sequential circuit has minimum and maximum propagation delay
  Min^m propagation delay is called as contamination delay.



→ **Contamination delay**: t_{ccq} = min^m time after clock edge during which Q will not have started changing yet.

Dynamic Discipline

→ The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge.

→ Specifically, the input must be stable.
- at least t_{setup} before the clock edge.
- at least until t_{hold} after the clock edge.

Maximum Clock Frequency

→ The clock frequency for a synchronous sequential circuit is limited by the timing parameters of its flip-flops and gates. This limit is called the maximum clock frequency for the circuit.

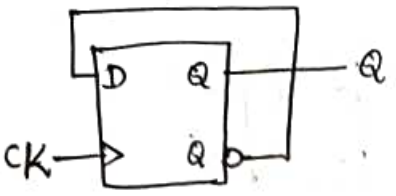
→ The minimum clock frequency is the reciprocal of this frequency.

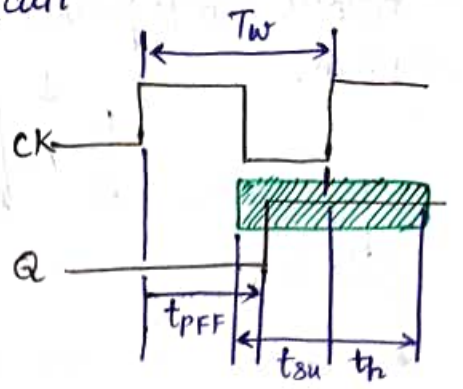
→ Relevant timing parameters:
- Gates:
  - ↳ Propagation delays: min t_{PLH}, min t_{PHL}, max t_{PLH}, max t_{PHL}.
- Flip-flops:
  - → Propagation delays: min t_{PLH}, min t_{PHL}, max t_{PLH}, max t_{PHL}.
  - → Setup time: t_{su}.
  - → Hold time: t_h.

4-timing parameters:
- → Setup time → time allocated to make the input stable before the application of clock.
- → Hold time
- → Aperture time
- → propagation delay (pd).

Min^m clock period = setup time + pd

Eg: Analyze this sequential circuit





Tw: clock period

T_{su}: setup time of the FF

t_h: hold time of the FF

t_{phl}, t_{hl}: propagation delays.

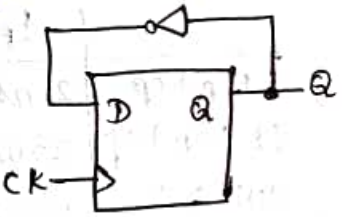
$T_W \geq \max t_{PFF} + t_{su}$

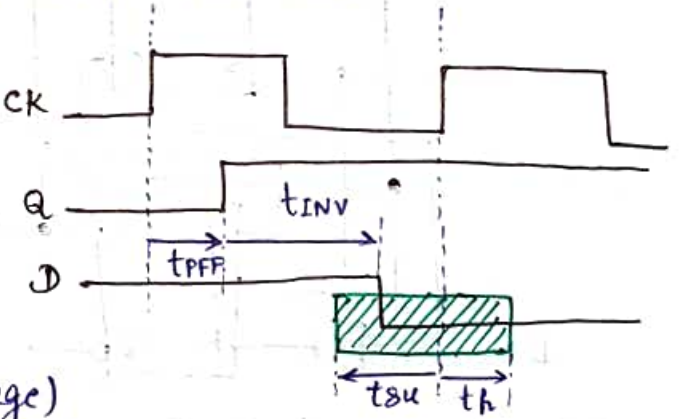
For the 7474, $\max t_{PLH} = 25\,ns$, $\max t_{PHL} = 40\,ns$, $t_{su} = 20\,ns$.

$\therefore T_w \geq \max(\max t_{PLH} + t_{su}, \max t_{PHL} + t_{su})$

$\Rightarrow T_w \geq \max(25+20, 40+20) = 60.$

Eg. Analyze the sequential circuit with the combination circuit.





t_{PFF}: Propagation delay of FF (average)

t_{su}: setup time of FF

t_{inv}: propagation delay of inverter

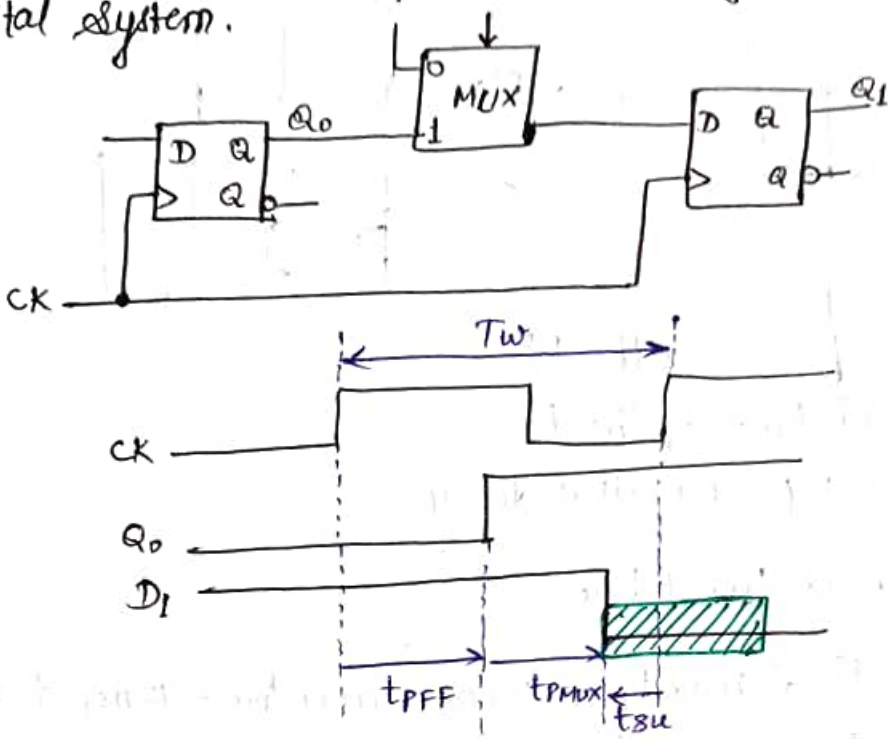
Minm clock period required,

$(T_w)_{min} = \max t_{PFF} + \max t_{pinv} + t_{su}$

$\therefore T_w \geq \max t_{PFF} + \max t_{pinv} + t_{su}$

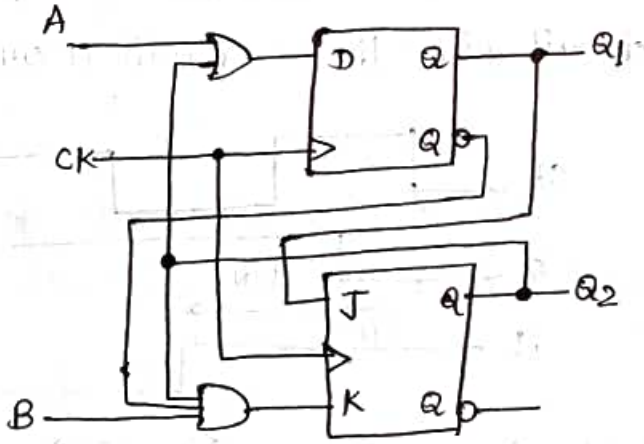
17-10-2023

Eg. Obtain the min^m clock period of the logic circuit or digital system.



$T_W \geq \max t_{PFF} + \max t_{PMUX} + t_{SU}$

Eg.



	t_P	t_{SU}
D Flip-Flop	20nS	5nS
JK Flip-Flop	25nS	10nS
AND Gate	12nS	
OR Gate	10nS	

Paths from Q_1 to Q_1 : None

Paths from Q_1 to Q_2 : $T_W \geq \max t_{PD\,FF} + t_{JKSU} = 20 + 10 = 30nS$

$T_W \geq \max t_{PD\,FF} + t_{AND} + t_{JKSU}$

$= 20 + 12 + 10 = 42nS$

Paths from Q_2 to Q_1 : $T_W \geq \max t_{PJKFF} + t_{OR} + t_{D\,SU}$

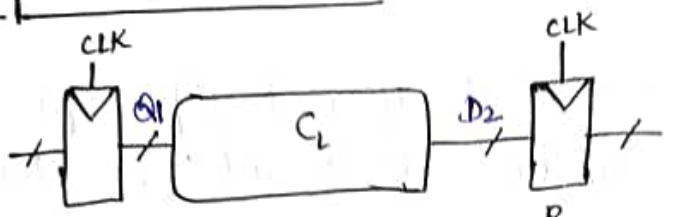
$= 25 + 10 + 5 = 40nS.$

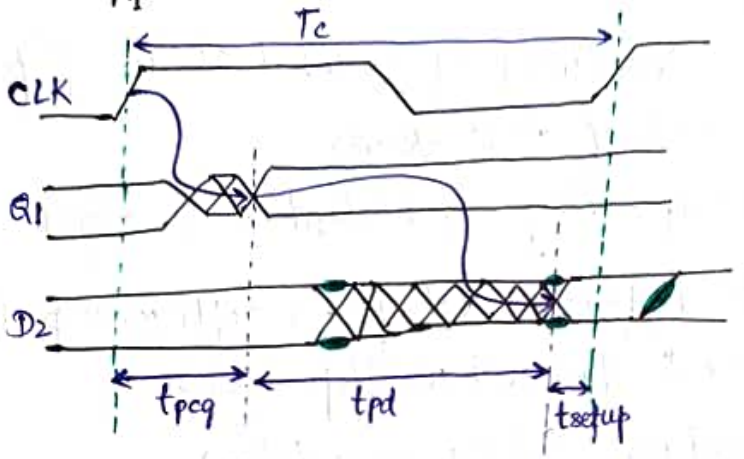
Paths from Q_2 to Q_2 : $T_W \geq \max t_{PJKFF} + \max t_{AND} + t_{JKSU}$

$= 25 + 12 + 10 = 47\,nS.$

$\therefore T_W \geq 47nS.$

Setup Time Constraint

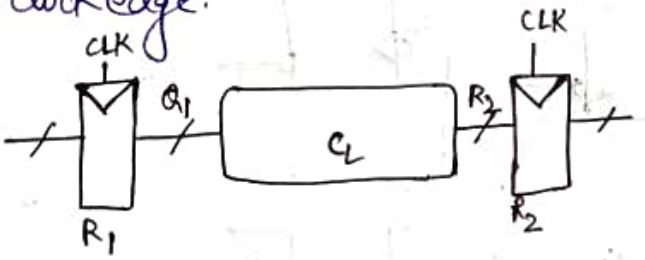


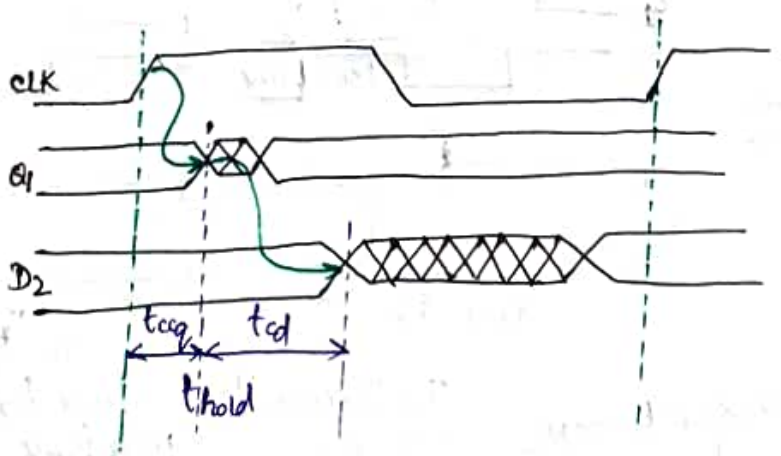


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$
$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

Hold Time Constraint

- Depends on the minimum delay from register R_1 through the combinational logic.
- The input to register R_2 must be stable for atleast t_{hold} after the clock edge.





$$t_{hold} < t_{ccq} + t_{cd}$$
$$t_{cd} > t_{hold} - t_{ccq}$$

18-10-2023

Clock Skew

→ If a clock edge does not arrive at different flip-flops at exactly the same time, then the clock is said to be skewed between these flip-flops.

→ The difference between the times of arrival at the flip-flops is said to be the amount of clock skew.

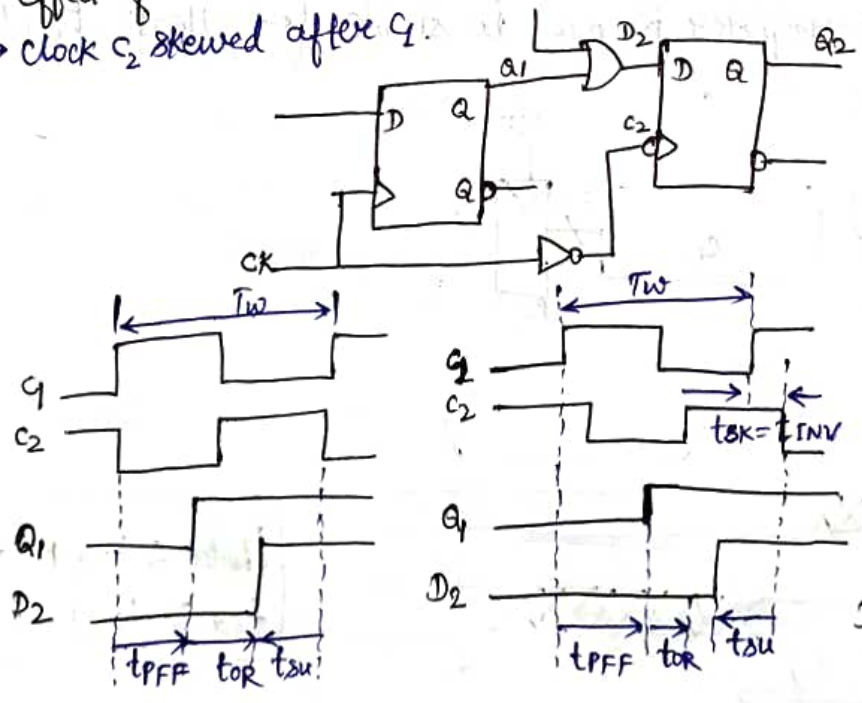
→ Spatial variation in temporally equivalent clock edges.

→ Clock skew is due to different delays on different paths from the clock generator to the various flip-flops.

- Different length wires (wires have delay).
- Gates (buffers) on the paths.
- Flip-flops that clock on different edges (need to invert clock for some flip-flops).
- Gating the clock to control loading of registers (a very bad idea).

Eg. Effect of clock skew on clock rate.

→ Clock C_2 skewed after C_1.



Positive Clock Skew.

$T_w \geq \max t_{PFF_1} + \max t_{OR} + t_{su_{FF_2}}$

(if clock not skewed, i.e., $t_{INV} = 0$)

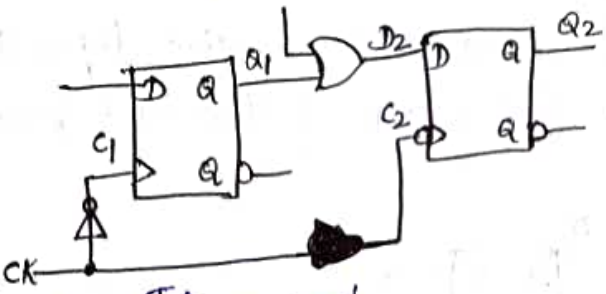
$T_w \geq \max t_{PFF} + \max t_{OR} + t_{su} - \min t_{INV}$

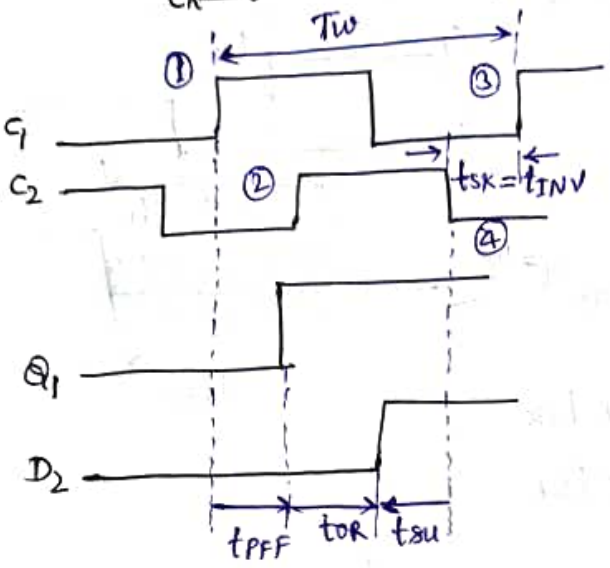
(if clock skewed, ie, $t_{INV} > 0$)

$T_w + \min t_{INV} \geq \max t_{PFF} + \max t_{OR} + t_{su}$

→ clock C_2 skewed after C_1.

→ There is a change in clock period when there is skew.

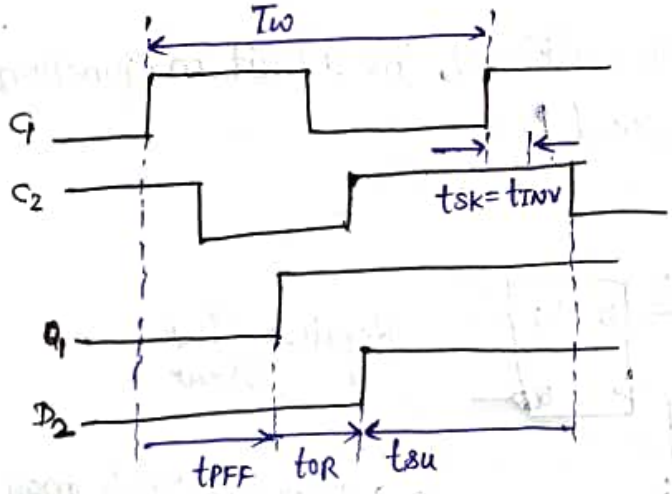


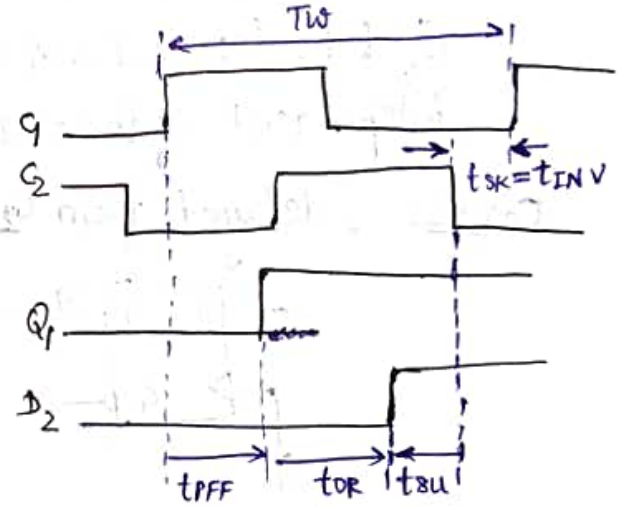


$T_w \geq \max T_{PFF} + \max t_{OR} + t_{su}$
(if clock not skewed, ie., $t_{INV} = 0$)

$T_w \geq \max T_{PFF} + \max t_{OR} + t_{su} + \max t_{INV}$
(if clock skewed, ie., $t_{INV} > 0$)

→ Summary of maximum clock frequency calculations.





C_2 skewed after C_1: $T_w \geq \max T_{PFF} + \max t_{NET} + t_{su} - \min t_{INV}$

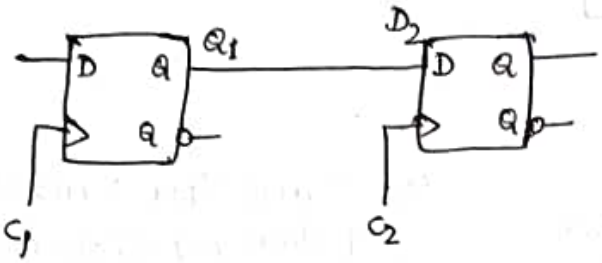
C_2 skewed before C_1: $T_w \geq \max T_{PFF} + \max t_{NET} + t_{su} + \max t_{INV}$

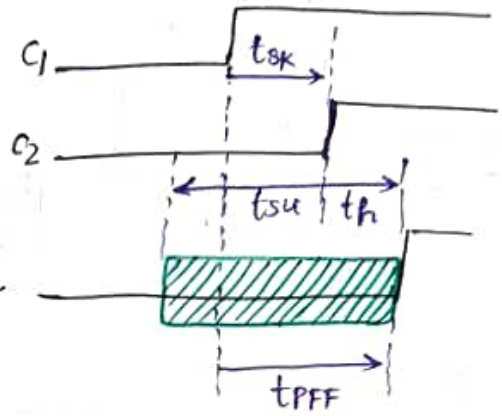
→ Positive clock skew: If data path and clock path are in same forward direction. ↳ clock period gets reduced (due to inverter at C_2).

→ Negative clock skew: If data path and clock path are in opposite dirn. ↳ clock period gets increased.

Maximum Allowable Clock Skew

↳ The clock skew can be positive or negative depending upon the routing direction and position of the clock source.





Positive clock skew

$T_w \geq \max T_{PFF} + t_{su} + t_{SK}$

$T_w + t_{SK} \geq \max t_{PFF} + t_{su}$

$t_{PFF} > t_h + t_{SK}$

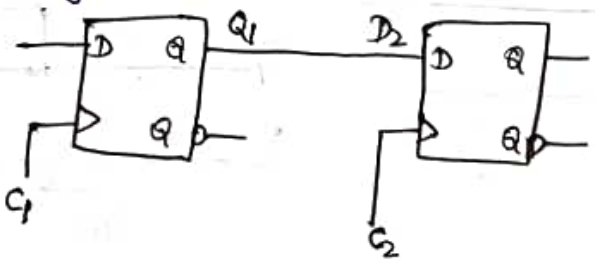
$t_{SK} < \min t_{PFF} - t_h$

$t_h < t_{PFF} - t_{SK}$

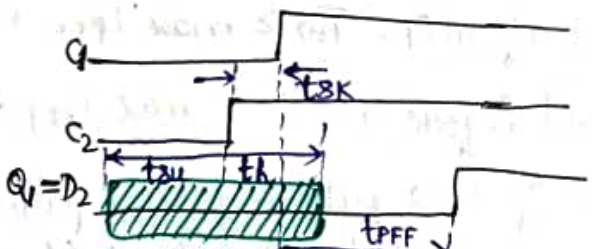
→ Improves performance (Clock Period reduces), but makes the hold harder to meet.

If t hold is not met (race conditions), the circuit malfunctions independent of the clock period!

Case-2: C_1 delayed from C_2



Negative Clock Skew



→ Degrades performance (Clock Period increases) but thold is easier to meet (eliminating race conditions).

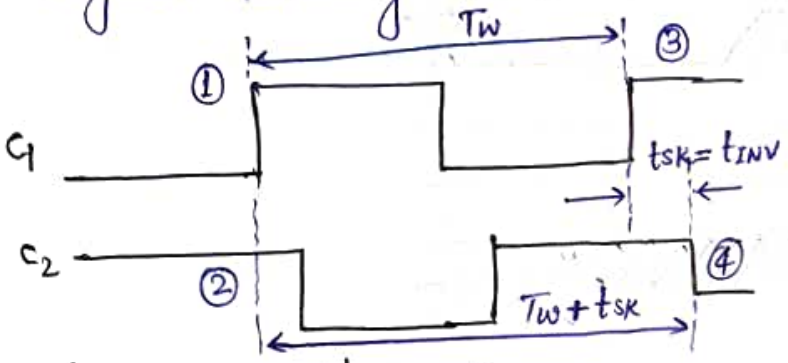
$T_w \geq \max T_{PFF} + t_{su} + t_{SK}$ (if clock skewed, ie, $t_{INV} > 0$)

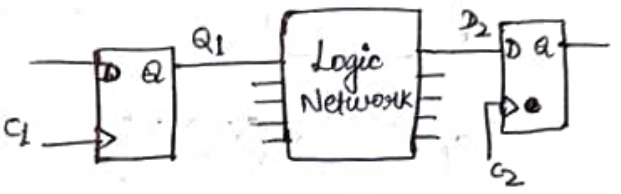
$T_w - t_{SK} \geq \max T_{PFF} + t_{su}$

$t_h < t_{PFF} + t_{SK}$

Race Conditions

If input In is sampled on the rising edge of C_1 at edge 1 into D1. The new value at the output D1 (Q1) propagates through the combinational logic and should be valid before edge 4 at C_2. However, if the minimum delay of the combinational logic block is small, the inputs to D2 may change before the clock edge 2, resulting in incorrect evaluation.



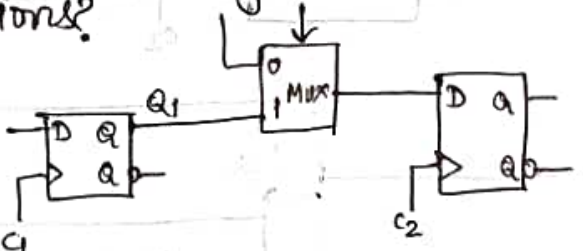


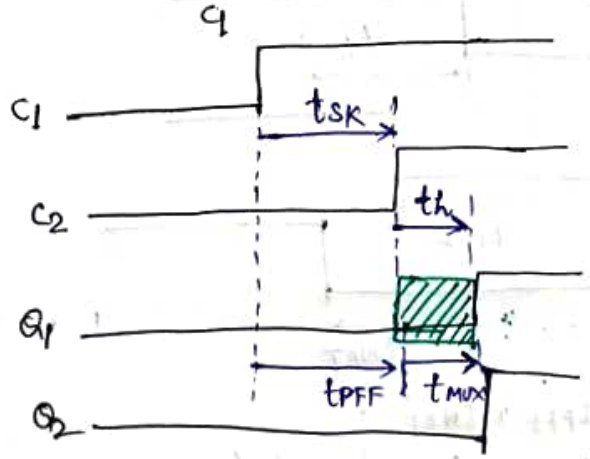
How to avoid races:

One must ensure that the minimum propagation delay through the register and logic is long enough that the inputs to D2 are valid for a ~~long~~ hold time after edge D_2.

→ To overcome skew problems, it is necessary to satisfy the hold-time constraints at design-time.

→ How does additional delay between the flip-flops affect the skew calculations?

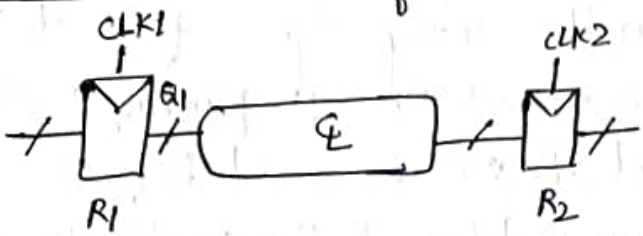




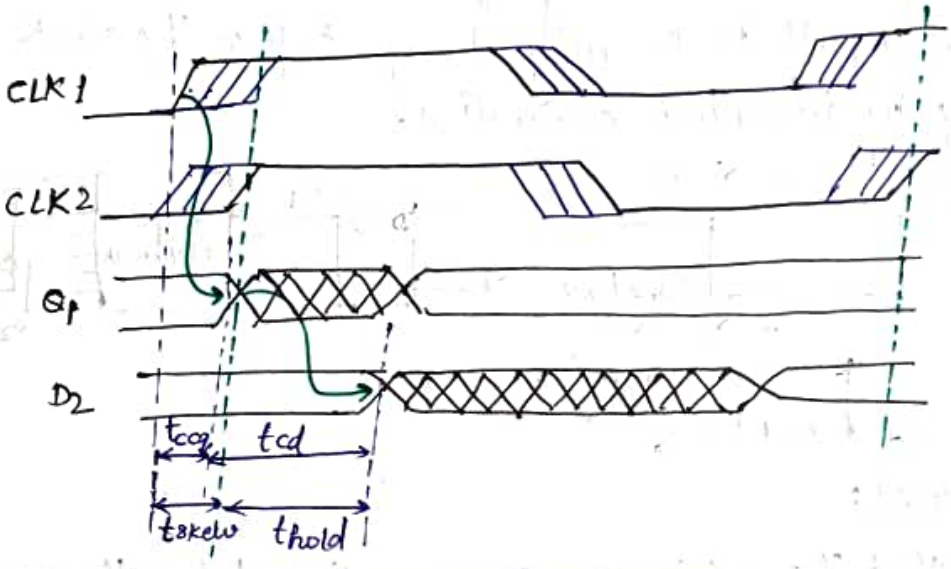
$t_{SK} \leq \min t_{PFF} - t_h$

$t_{SK} \leq \min t_{PFF} + \min t_{MUX} - t_h$

HOLD Time Constraint for Clock Skew

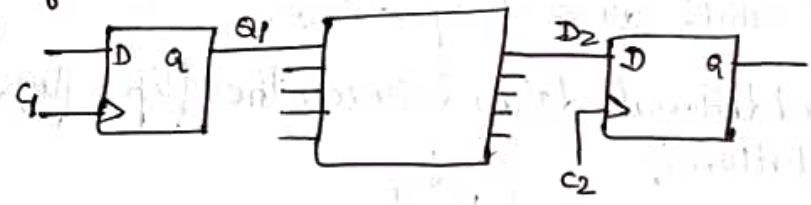


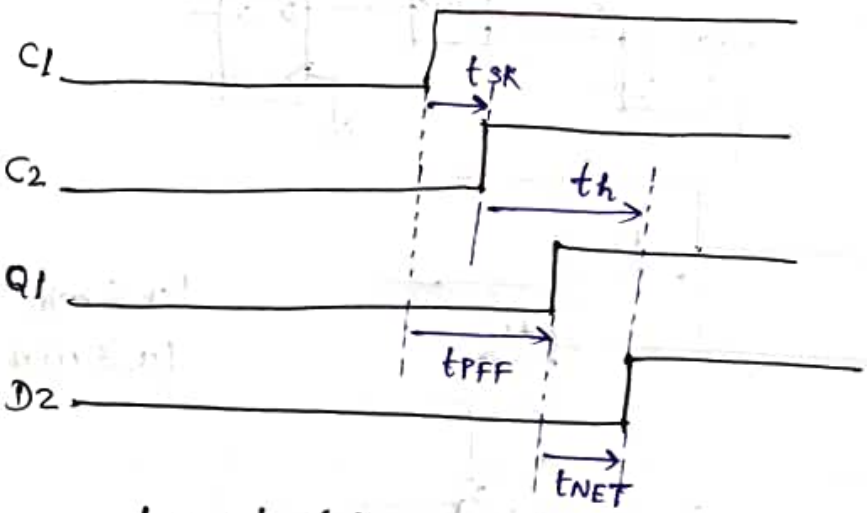
In the worst case, CLK2 is later than CLK1



$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$
$$t_{cd} > t_{hold} + t_{skew} - t_{ccq}$$

Summary of allowable clock skew calculations





$$t_{SK} + t_h \leq t_{PFF} + t_{NET}$$
$$t_{SK} \leq \min t_{PFF} + \min t_{NET} - t_h$$

Summary:

→ Maximum clock frequency is a fundamental parameter in sequential computer systems.

→ Possible to determined clock frequency from propagation delays and setup time.

→ The longest path determines the clock frequency.

→ All flip-flop to flip-flop paths must be checked.

26-10-2023

VERY LARGE SCALE INTEGRATION (VLSI)

CMOS (Complementary Metal Oxide Semiconductor)

Logic family :

- CMOS → All digital designs are made using CMOS ⇝ Uses MOSFET
- BJT → TTL (Transistor - Transistor logic)
- Hybrid CMOS + BJT

Schottky
MS (Metal semiconductor)

↳ Zener diode

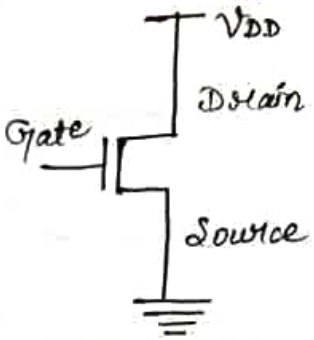
In BJT, reverse saturation current ↑ for every 10°C rise in temperature.

↳ OFF to ON ⇝ due to forward current
↳ ON to OFF
↳ memory

CMOS
↳ Easy to manufacture (sophisticated fabs)

27-10-2023

→ CMOS advantages:
↳ Density ↑ (No. of transistors per unit area)
↳ Miniaturization is possible
↳ Low power consumption
↳ Sophisticated fabs

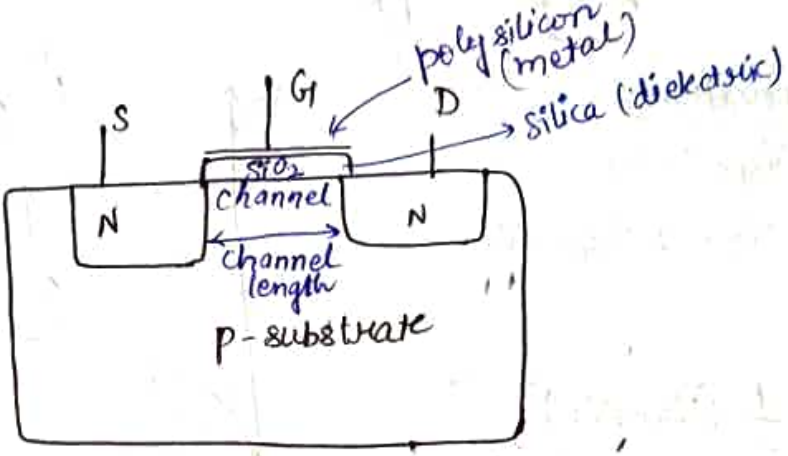


MOSFET
↳ NMOS → n-type material used in Drain & Source
↳ Conduction due to majority charge carrier only.
↳ Gate is made of metal.
↳ separated from D & S by a dielectric (SiO_2).

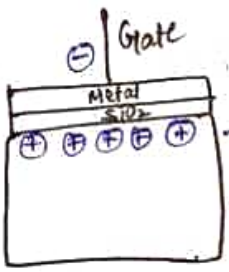
n-type material (doping): Taking a lightly-doped (p or n) semiconductor and implanting ions to make n- (or p-) type material in a controlled manner.

Diffusion: Implanting free-electrons to semiconductor to make n- or p-type material.

CMOS:



→ $V_G < 0$



Accumulation Region

V_G = ⊕ve

Acts as capacitor

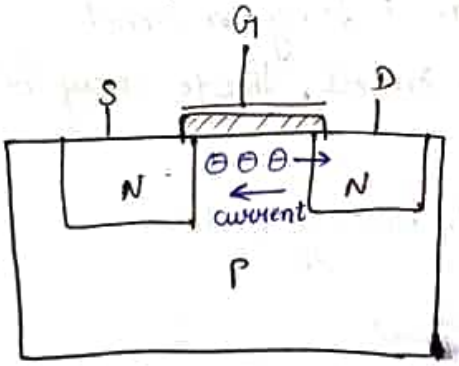
→ channel is depleted of charges

Depletion Region ⇒ "Inversion"

↳ no ⊕ve ions present beneath the channel

($V_G > V_T$)

→ Threshold voltage (V_T): Minimum voltage required to form the channel (negative voltage).



- V_{GS} = ⊕ve is applied.

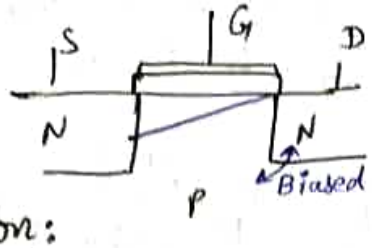
  $V_{GS} < V_T$: No channel is formed (cutoff region)

  $V_{GS} = V_T$: channel starts to form.

  $V_{GS} \geq V_T$: Channel is formed.

  ↳ ON

- V_{GD} is applied.
  ↳ Channel gets ~~amplified~~ pinched off.

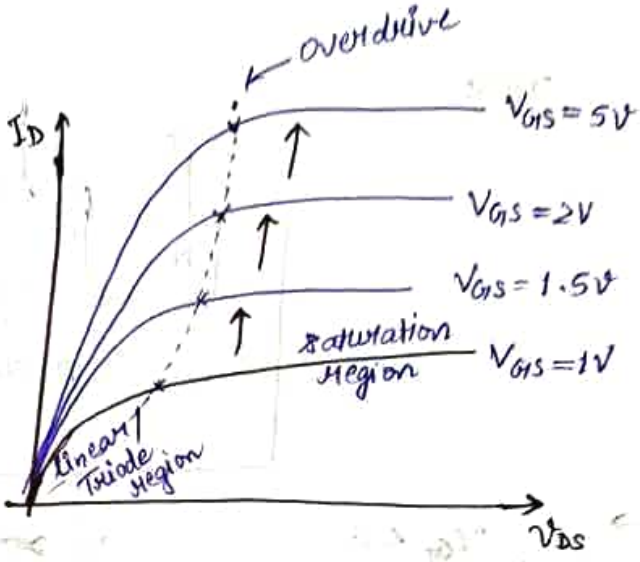


CMOS operation:

① $V_{GS} \geq V_T$

② $V_{DS} \geq V_{GS} - V_T$

→ $I_{DS} = \frac{1}{2} \underbrace{(V_{GS}-V_T)^2}_{\text{overdrive voltage}}$



→ When input voltage, $V_{GS} < V_T$, then also there is some current (due to minority charge carrier) called as subthreshold leakage.
  ↳ Not a good thing in digital electronics.

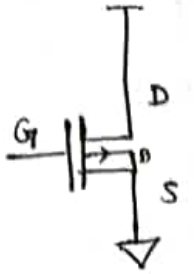
→ Channel length: Separation b/w drain and source.
  ↳ plays important role in conduction of current.

- Channel length > 1 μm : long channel

↳ When channel is too small, there may be conduction through the gate.

→ Channel length defines technology node.

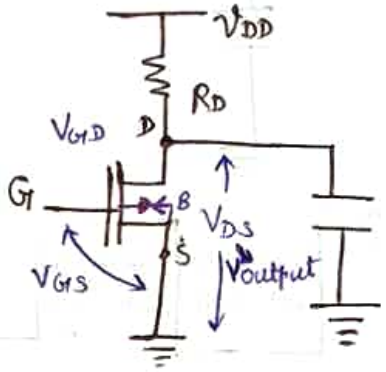
NMOS: Threshold voltage : V_T



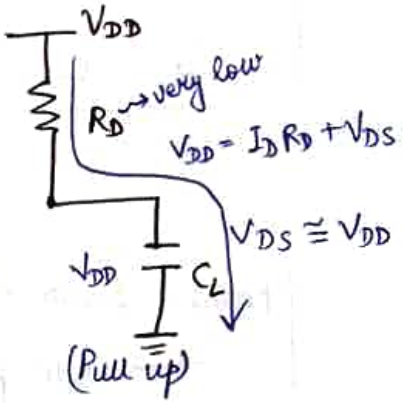
S : Ground
D : ~~GND~~ HIGH
Body (B) / Substrate : Source

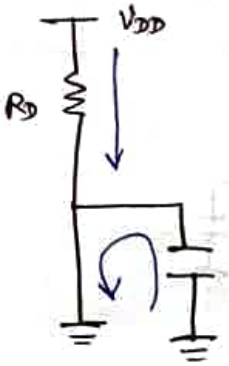
PMOS:

Threshold voltage : $|V_T|$



→ Inverter

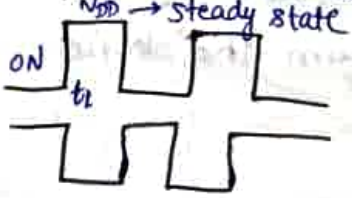




R_D → very low

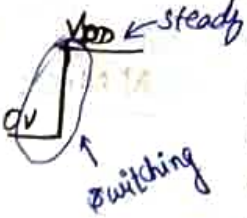
Pull Down : (NMOS) → Active

Pull Up : Circuit responsible for ~~making~~ driving Voutput go to zero.



Circuit responsible for driving output to High.
↳ Resistive device
↳ Passive

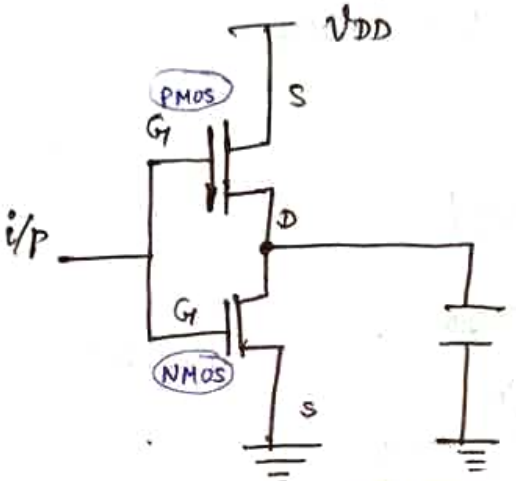
Dynamic power: Power consumption during switching :
Static power : Power consumption during steady state.



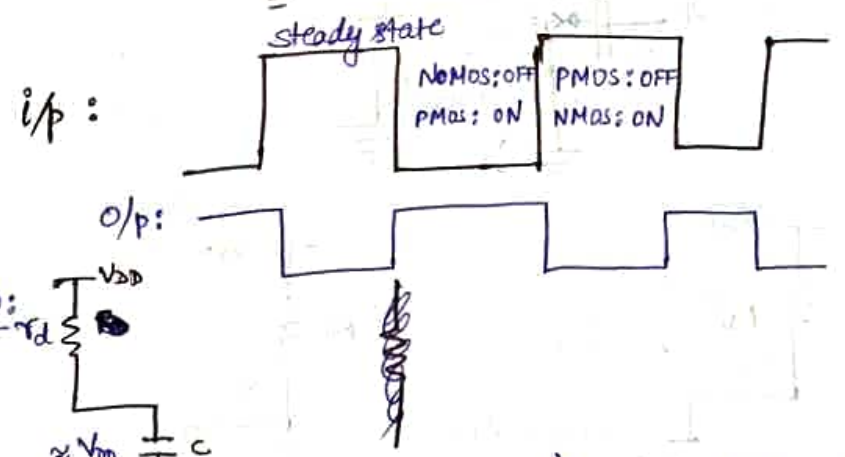
↳ To reduce static power dissipation:

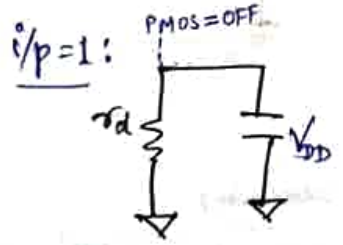
Replace resistive Pull Up circuit (Passive) to Active circuit which is complement of the Pull Down circuit.

[Complement of NMOS : PMOS]



→ CMOS Logic circuit
↳ contains both NMOS and PMOS in a single circuit.





PMOS : LOAD (here)
↳ Static power dissipation = 0 (ideally)

Sink current : Current into the device.
Source current: Current out from the device.

CMOS

↓

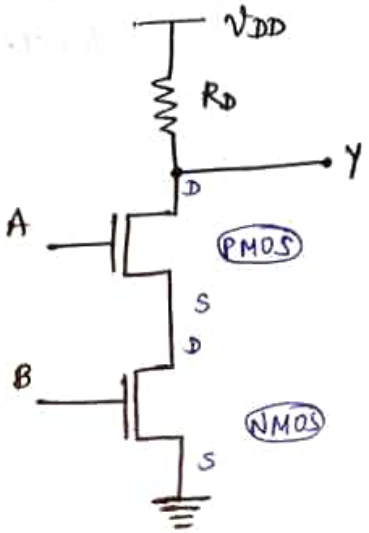
Realisation:
NAND
NOR

NAND:

$Y = \overline{AB}$ [NMOS + PMOS]

$\overline{Y} = AB$

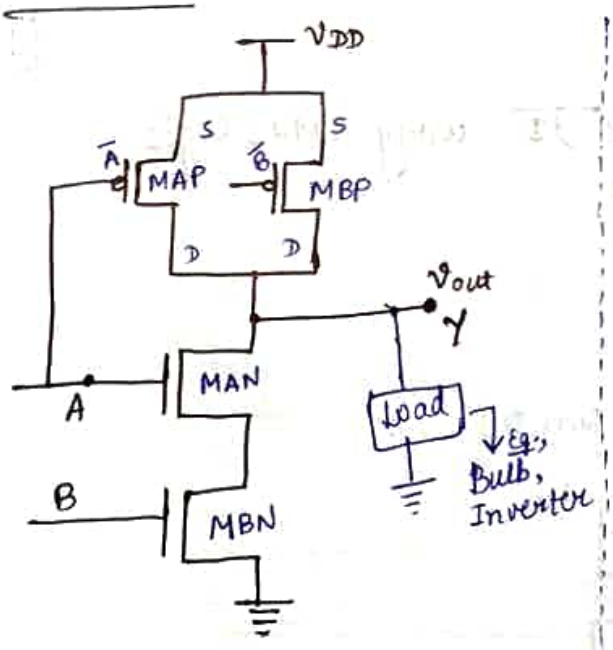
↓ NMOS ↘ series connection



Realisation as ~~using~~ NAND Gate:

A	B	Y	
0	0	1	→ PMOS: OFF, NMOS: OFF
0	1	1	→ PMOS: OFF, NMOS: ON
1	0	1	
1	1	0	

$Y = \overline{A} + \overline{B}$



AB 00: $\frac{r_{dp}}{2}$, C_L — $\tau = \frac{r_{dp}}{2} C_L$

AB 11: $2r_{dN}$ — $\tau = 2 r_{dN} C_L$

AB 01: r_{dp}, o/p, C_L — $\tau = r_{dp} C_L$

AB 10: $\tau = r_{dp} C_L$

A	B	MAP	MBP	MAN	MBN	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

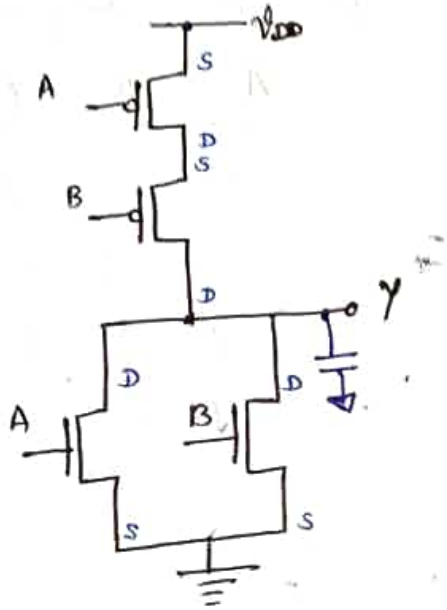
Eg Draw the CMOS logic circuit for NOR.

NOR: $Y = \overline{A+B}$

$\overline{Y} = \overline{\overline{A+B}} = \underbrace{A+B}_{NMOS}$

PMOS + NMOS } CMOS

Two Types:
Series : AB → AND
Parallel : A+B → OR



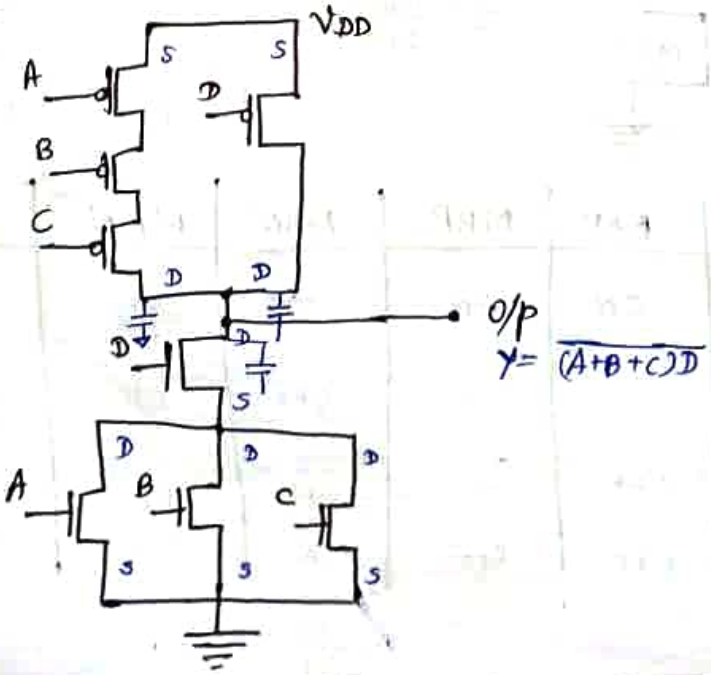
1-11-2023

Eg Implement $Y = \overline{(A+B+C)D}$ using CMOS logic.

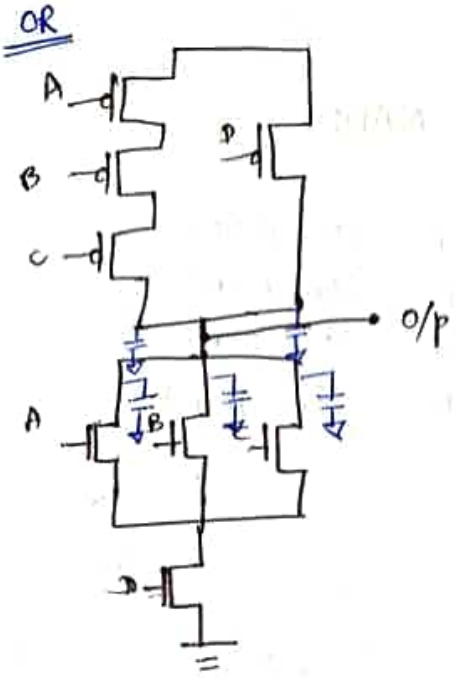
Soln: $Y = \overline{(A+B+C)D}$

$\overline{Y} = \underbrace{(A+B+C)}_{X} D$

NMOS ((NMOS || D) sum D).



OR



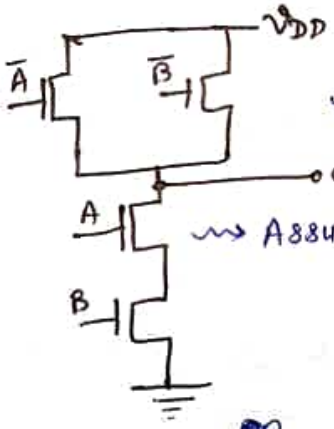
CMOS
↳ implements NAND, NOR
↳ To realise AND, OR
↓
we require extra inverter

[Speed ↑ ⇒ Power ↑]

Eg Implement $Y = A + B$.

$Y = A + B$

$\overline{Y} = \overline{A}\,\overline{B}$

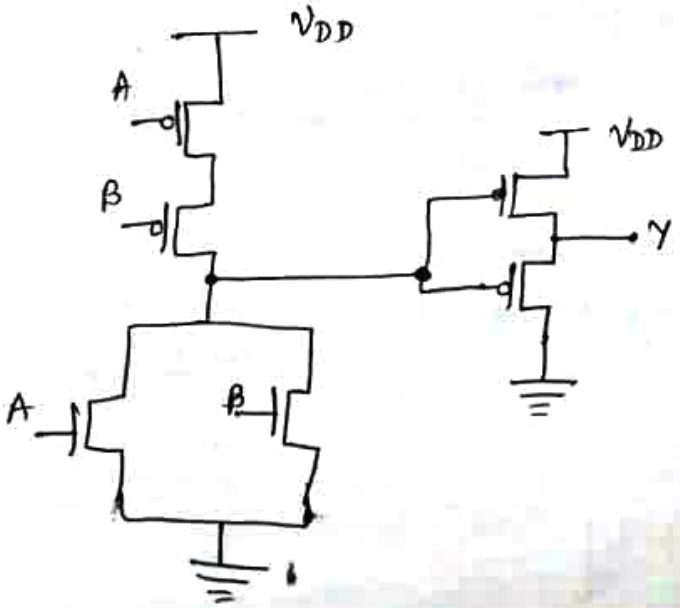


⇝ Requires extra inverter

⇝ Assumption : Inputs are comming from field

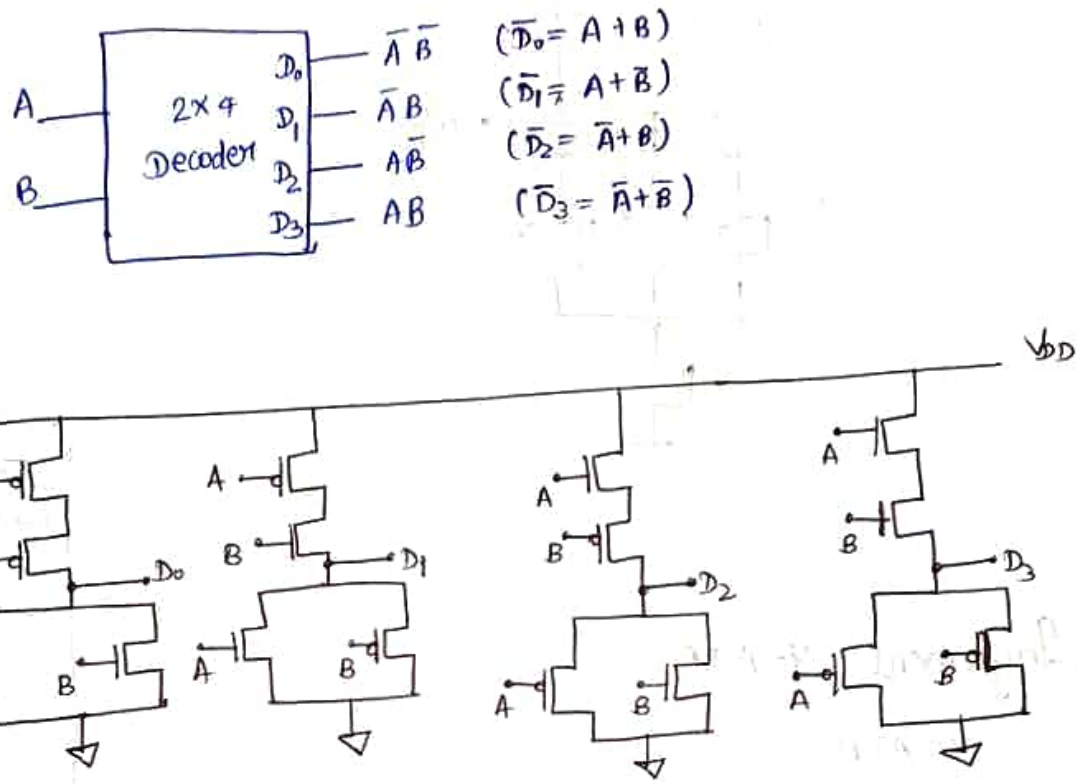
→ This way is done generally.

OR



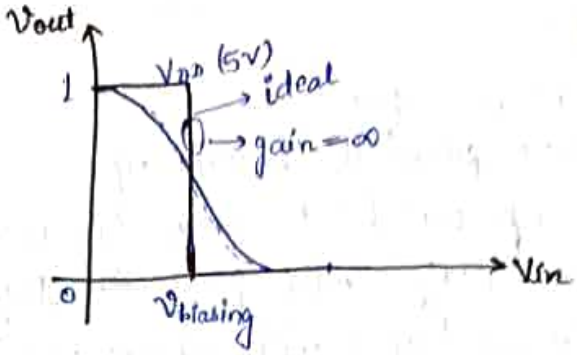
Q) Realise 2×4 decoder using NAND.

Soln:



2-11-2023

Transfer Characteristics



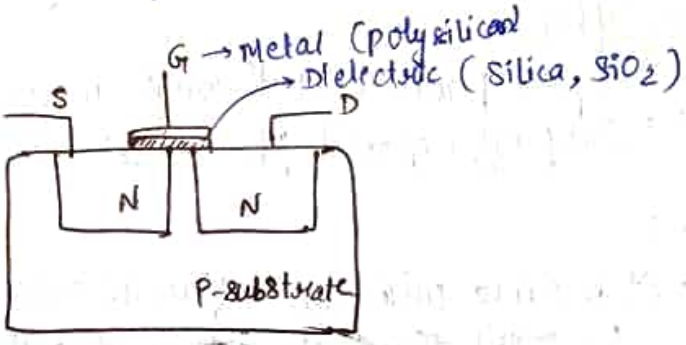
CMOS FABRICATION

Moore's Law: Every 24 months, no. of transistors available in a chip gets doubled.

↳ Not applicable after 7nm technology.

→ Technology node: The length b/w drain and source (channel length)

NMOS:



→ CMOS transistors are fabricated on silicon wafer.
→ Lithography process similar to printing press.
→ On each steps, different materials are deposited or etched.
→ Easiest to understand by viewing both top and cross-section of wafer in a simplified manufacturing process.
→ Chips are built in huge factories called fabs.
↳ Contain clean rooms as large as football fields, costing billions of dollars.

nwell: n-substrate required to house the PMOS.

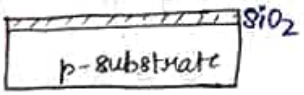
Fabrication Steps

→ Start with Blank wafer.
→ Build inverter from the bottom up.
→ ~~first~~ First step will be to form the n-well.
  ↳ Cover wafer with protective layer of SiO_2 (oxide).
  ↳ Remove layer ~~with~~ where n-well should be built.
  ↳ Implant or diffuse n dopants into exposed wafer.
  ↳ Strip off SiO_2.

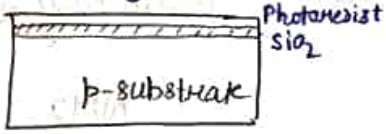
Oxidation:

↳ Grow SiO_2 on top of Si wafer.
  ↳ 900°–1200°C with H_2O or O_2 in oxidation furnace.

Photoresist:

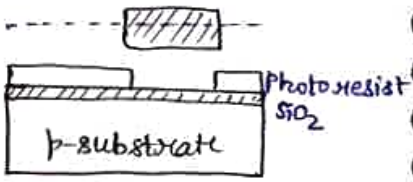


↳ Spin on photoresist
  ↳ Photoresist is a light-sensitive organic polymer.
  ↳ Softens where exposed to light.



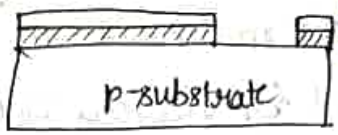
Lithography:

↳ Expose photoresist through n-well mask.
↳ Strip off exposed photoresist.



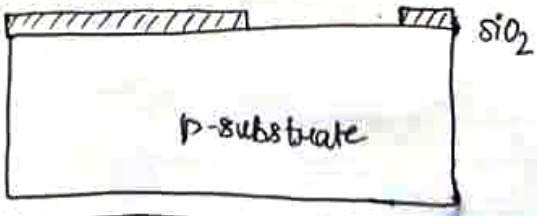
Etch:

↳ Etch oxide with hydrofluoric acid
  ↳ seeps through skin and eats bone; nasty stuff!!!
↳ Only attacks oxide where resist has been exposed.



Strip Photoresist:

↳ Strip off remaining photoresist.
  ↳ Use mixture of acids called piranah etch.
↳ Necessary so resist doesn't melt in next step.

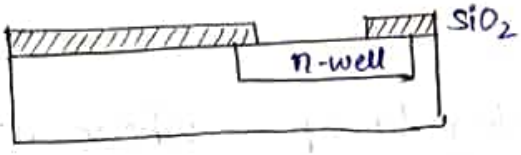


n-well:

↳ n-well is formed with diffusion or ion implementation.

↳ Diffusion:
- ↳ Place wafer in furnace with arsenic gas.
- ↳ Heat until As atoms diffuse into exposed Si.

↳ Ion Implanatation:
- ↳ Blast wafer with beam of As ions.
- ↳ Ions blocked by SiO_2, only enter exposed Si.

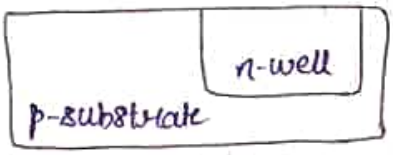


Strip Oxide:

↳ Strip off the Remaining oxide using HF.

↳ Back to bare wafer with n-well.

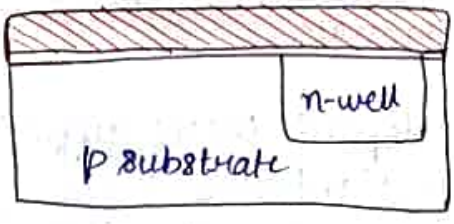
↳ Subsequent steps involve similar series of steps.



Polysilicon:

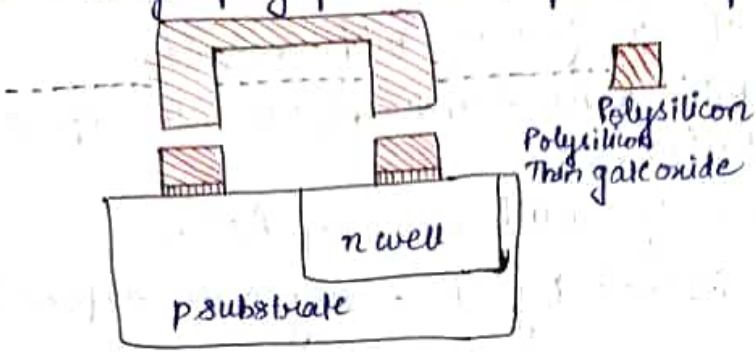
↳ Deposit very thin layer of gate oxide (<20 Å ; 6-7 atomic layers).

↳ Chemical vapour deposition (CVD) of silicon layer.
- ↳ Place wafer in furnace with Silane gas (SiH_4).
- ↳ Forms many small crystals called polysilicon.
- ↳ Heavily deposed to be good conductor.



Polysilicon Patterning:

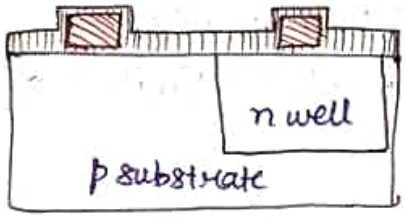
↳ Use same lithography process to pattern polysilicon.



Self-Aligned Process:

↳ Use oxide and masking to expose where n^+ dopants should be diffused or implanted.

↳ N-diffusion forms nMOS source, drain, and n-well contact.

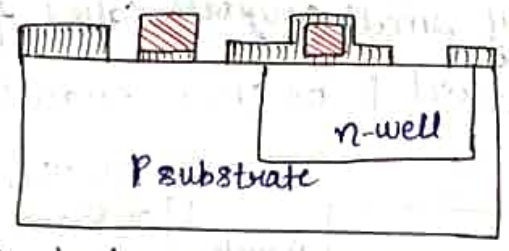


N-diffusion:

↳ Pattern oxide and form n^+ regions.

↳ Self-aligned process where gate blocks diffusion.

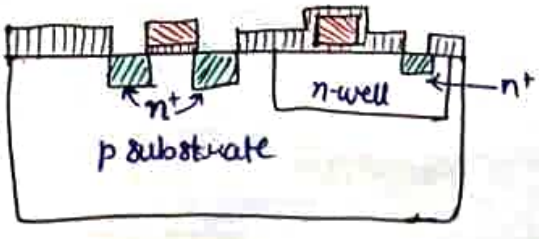
↳ Polysilicon is better than metal for self-aligned gates because it doesn't melt during later processing.



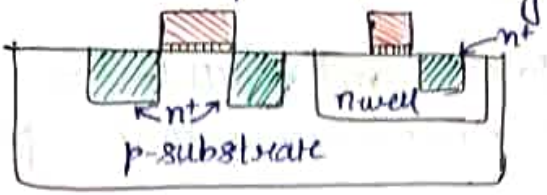
↳ Historically dopants were diffused.

↳ Usually ion implantation today.

↳ But regions are still called diffusion.

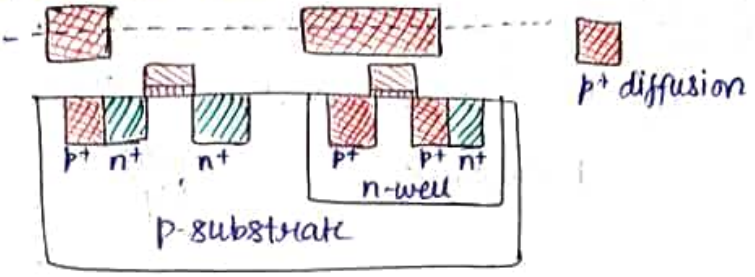


↳ Strip off oxide to complete patterning oxide.



P-diffusion:

↳ Similar set of steps form p+ diffusion regions for pMOS source and drain and substrate contact.

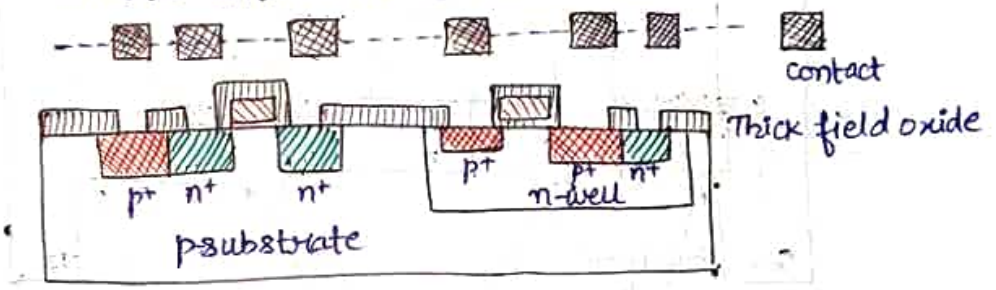


Contacts:

↳ Now, we need to wire together the devices.

↳ Cover chip with thick field oxide.

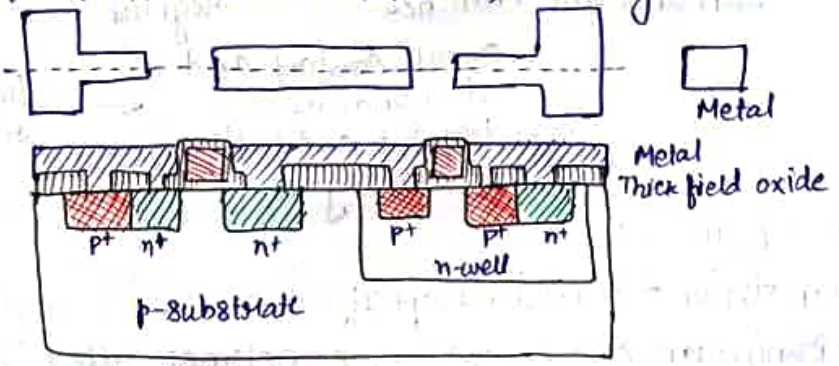
↳ Etch oxide where contacts cuts are needed.



Metalization

↳ Sputter on aluminium over whole wafer.

↳ Pattern to remove excess metal, leaving wires.

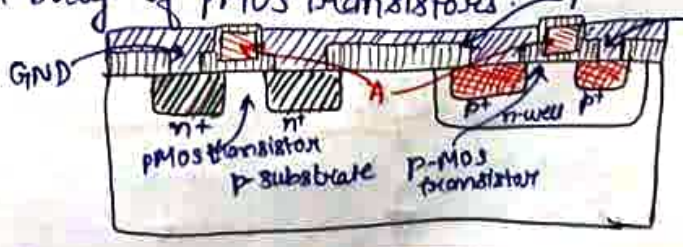


Inverter Cross-section:

↳ Typically use p-type substrate for nMOS transistors.

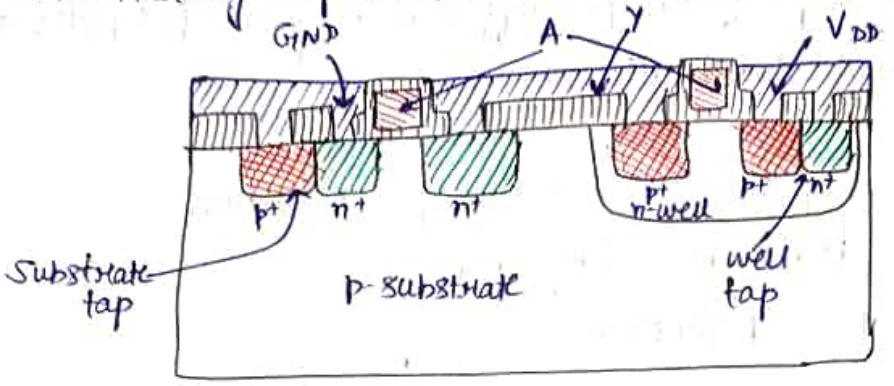
↳ Requires n-well for body of pMOS transistors.

↳ So, pMOS p-type source/drain doesn't short to p-type substrate.



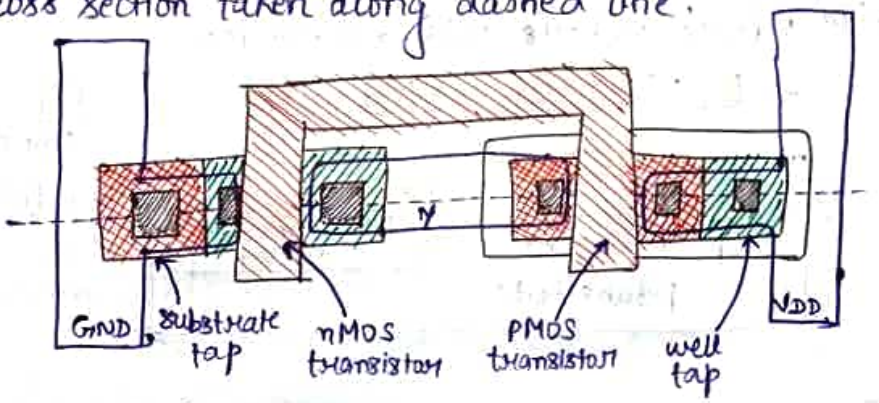
Well and Substrate Taps:

↳ Substrate must be tied to GND and n-well to V_{DD}.
↳ Metal to lightly-doped semiconductor forms poor connection called Shottky Diode.
↳ Use heavily doped well and substrate contacts/taps.



Inverter Mask Set:

↳ Transistors and wires are defined by masks.
↳ Cross section taken along dashed line.

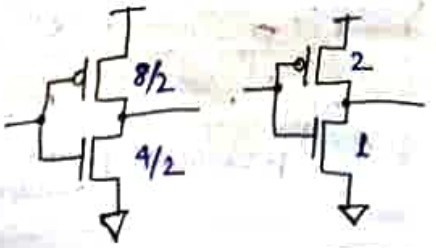


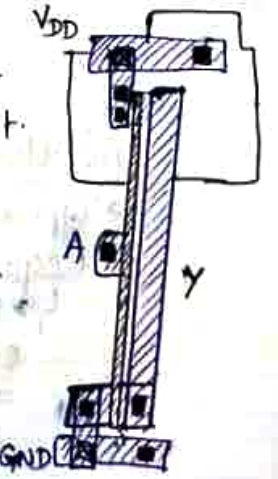
General CMOS Process Step:

Define active areas
Etch and fill trenches → Implant well regions → Deposit and pattern polysilicon layer
↓
Implant source and drain regions and substrate contacts
↓
Create contact and via windows
Deposit and pattern metal layers.

Inverter Layout:

↳ Transistor dimensions specified as Width/Length.
↳ Minimum size is 4λ/2λ, sometimes called 1unit.
↳ In $f=0.6\mu m$ process, this is $1.2\mu m$ wide, $0.6\mu m$ long, $\lambda = f/2$; f is called feature dimension.





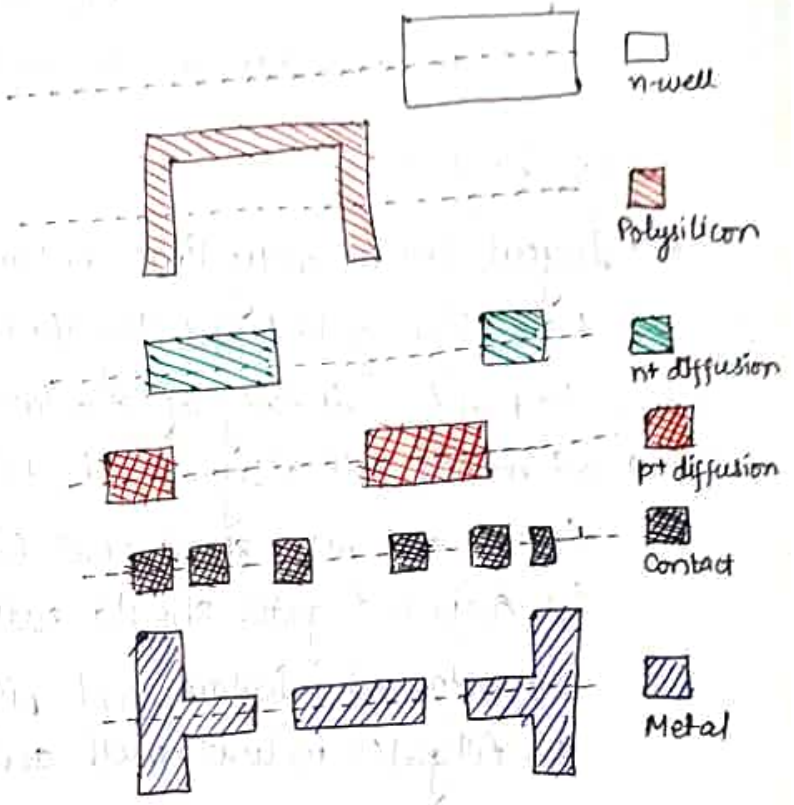
09-11-2023

VLSI CIRCUITS & LAYOUT

Detailed Mask Views

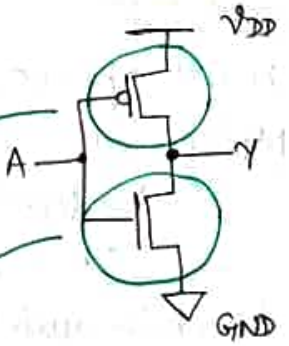
Six Masks:

- n-well
- Polysilicon
- n+ diffusion
- p+ diffusion
- Contact
- Metal

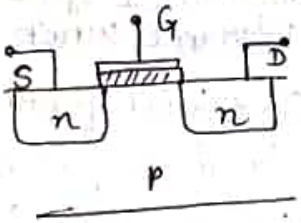


How to build CMOS circuits?

- CMOS Inverter
- Pull Up Network responsible for true output.
- Pull down network responsible for complementary output.



Layout



→ Chips are specified with set of masks

→ Minimum dimensions of masks determine transistor size (and hence speed, cost and power).

→ Feature size, f = distance between source and drain.

↳ Set by minimum width of polysilicon.

↳ Improves 30% every year or so

→ Normalise for feature size when describing design.

→ Express rules in terms of $\lambda = f/2$.

Eg. $\lambda = 0.3\,m$ in $0.6\,m$ process.

Gate Layout

→ Layout can be very time consuming.
  ↳ Design gates to fit together nicely.
  ↳ Build a library of standard cells.

→ Standard cell design methodology
  ↳ V_{DD} and GND about abut (standard height).
  ↳ Adjacent gates should satisfy design rules.
  ↳ nMOS at bottom and pMOS at top.
  ↳ All gates include well and substrate contacts.

Design Rules

Two major approaches:

MOSIS website: themosisservice.com

① Micron rules: stated as micron resolution
  ↳ Used for commercial purposes
  ↳ not scalable.

② Lambda rules: Simplified micron rules with limited scaling attributes. ↳ not used in commercial applications
  ↳ Scalable (one technology to another).
  ↳ very conservative and thus waste space.

→ Design rules represents a tolerance which insures:
  - very high probability of correct fabrication
  - scalable design rules: lambda parameter.
  - absolute dimensions (micron rules).

CMOS 'λ' Design Rules

→ MOSIS: MOS Implementation Service
  ↳ IC fabrication service available to universities for layout, simulation, and test the completed design.
  ↳ MOSIS rules are scalable λ rules.

CMOS-Design Rules

↳ Based on following components:

① Substrates or wells: p-type for NMOS devices
n-type for PMOS devices.

② Diffusion regions: At these regions, the transistors are formed.
↳ called as active layer.
↳ Defined by n^+ for NMOS and p^+ for PMOS transistors.

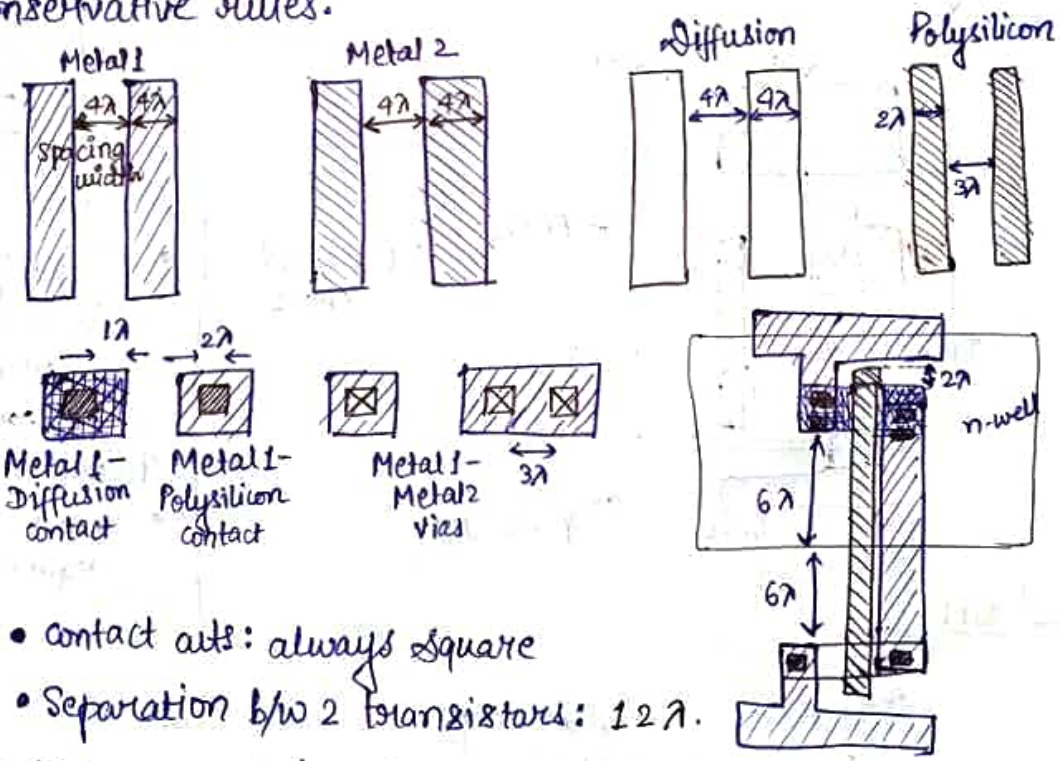
③ Polysilicon layers: Used to form the gate electrodes of the transistors.

④ Metal interconnects layers: Used to form the power supply and ground rails as well as input and output rails.

⑤ ~~Connect~~ contact and via layers: Used to form the inter-layer connections.

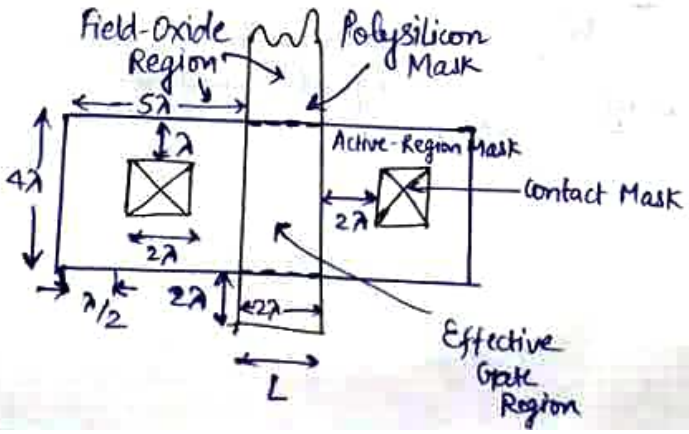
Simplified Design Rules:

Conservative rules:

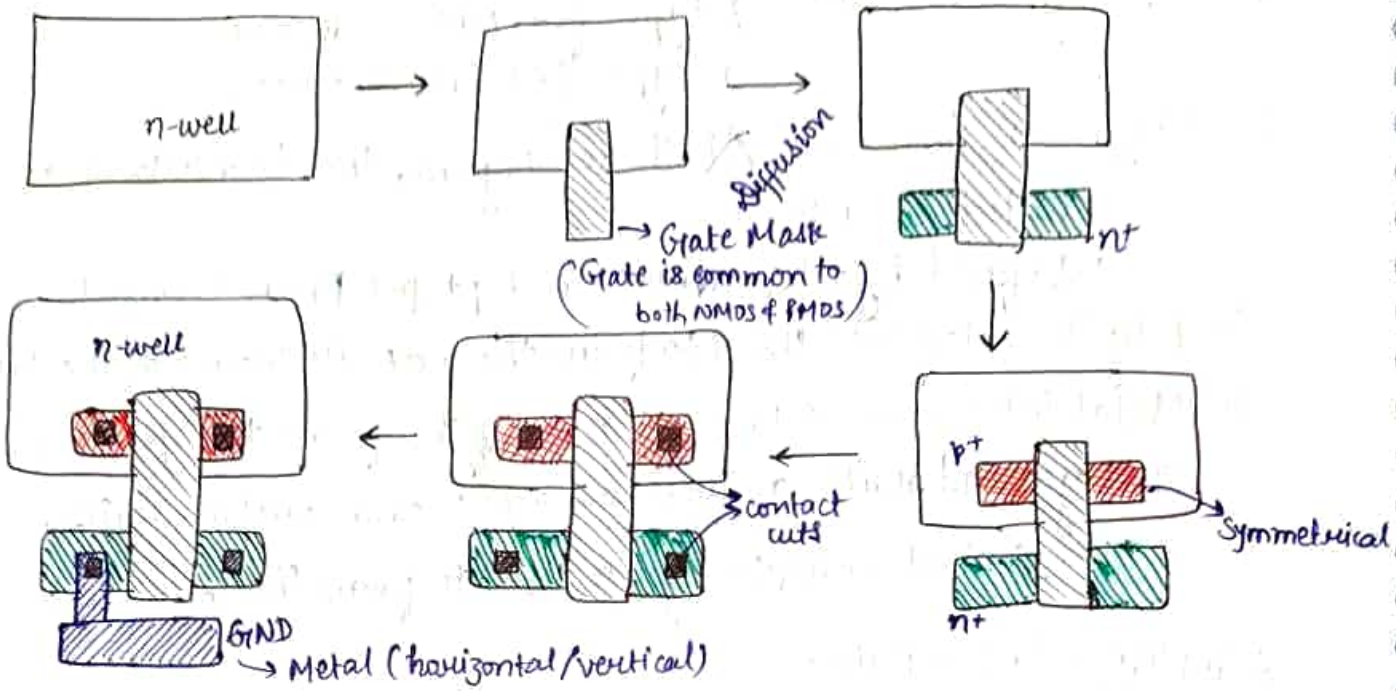


- Contact cuts: always square
- Separation b/w 2 transistors: 12λ.

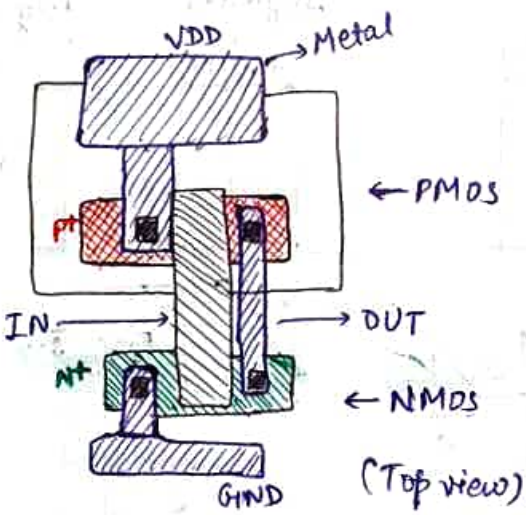
NMOS/PMOS layout based on λ-rules:



Layout of Inverter



4 Masks:
- n-well
- P+
- N+
- Gate

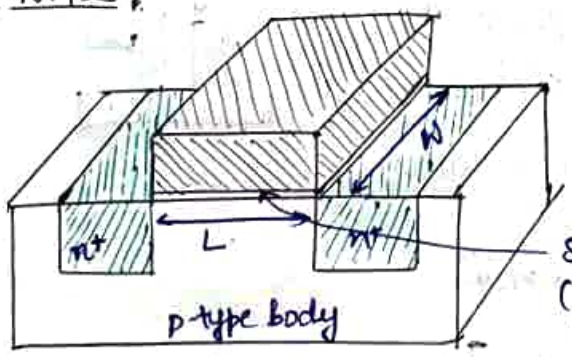


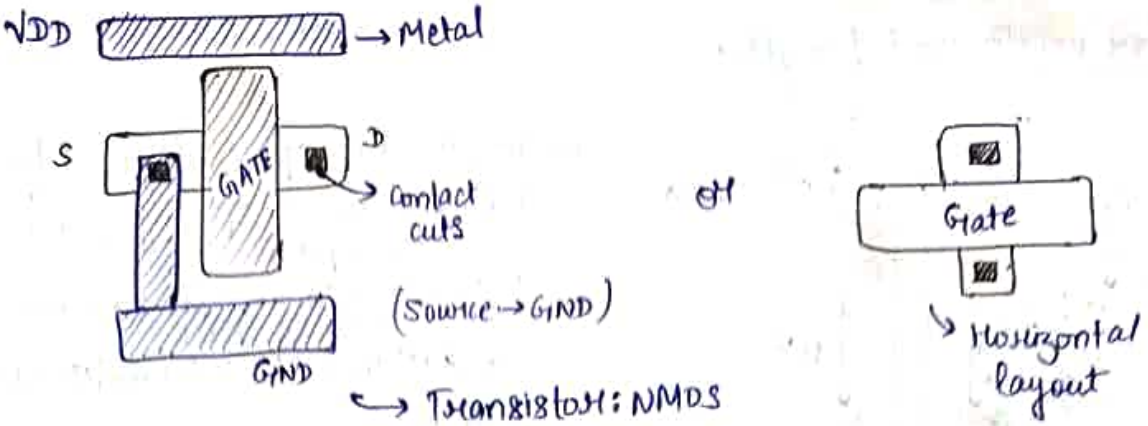
(Top view)

: Backend CMOS

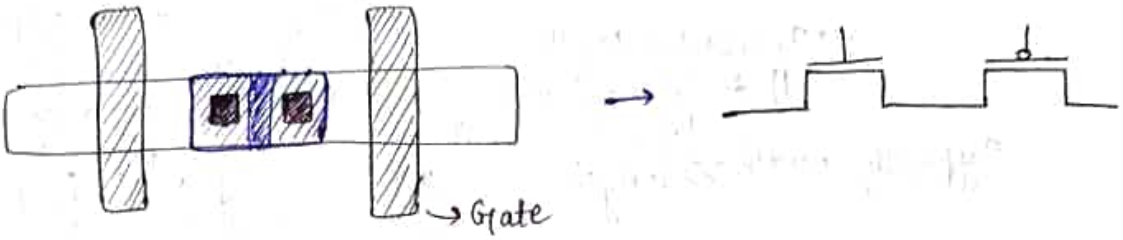
- P+ is bigger than N+ as P+ conduction: hole, N+ conduction: e⁻ and, some I_D flows in PMOS & NMOS if (hole mobility < e⁻)
- Contact cut equivalent: Parallel

Layout of NMOS:





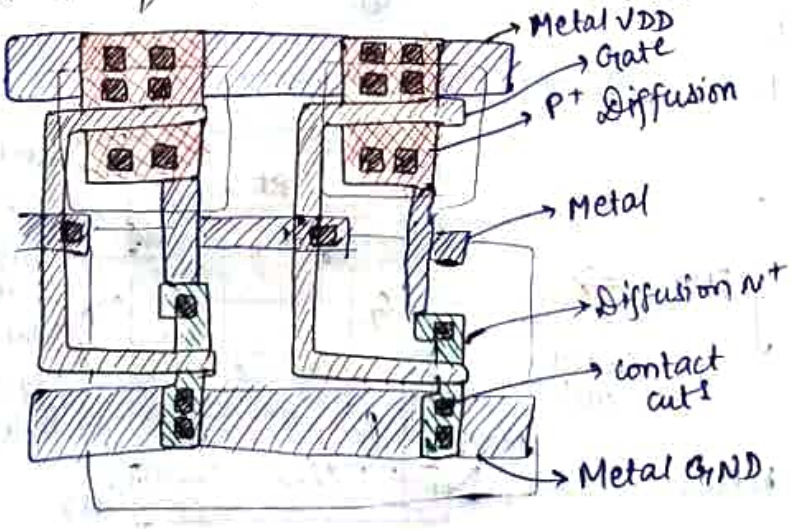
Layout:

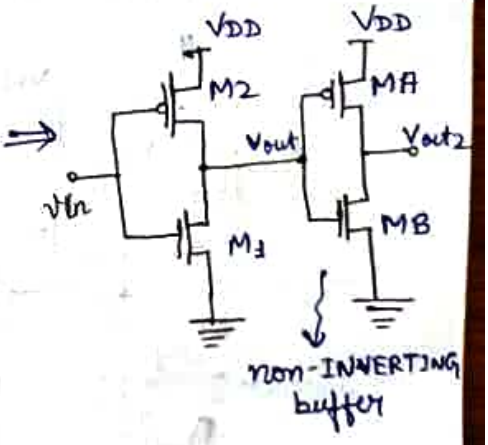


Connecting Poly and Diffusion
↳ can't connect poly to diffusion directly!

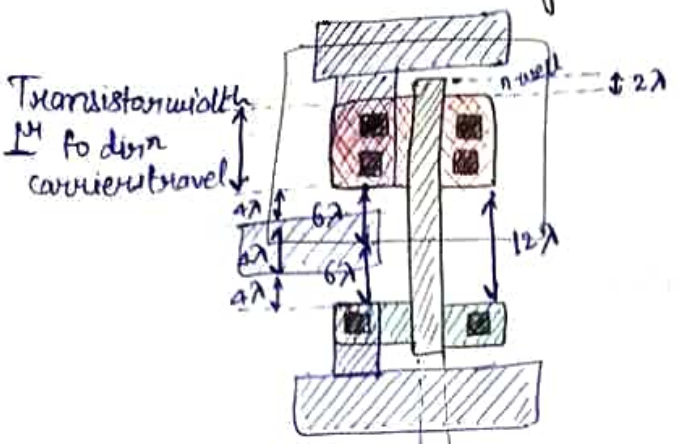
Layout Example:

Draw the equivalent transistor schematic.





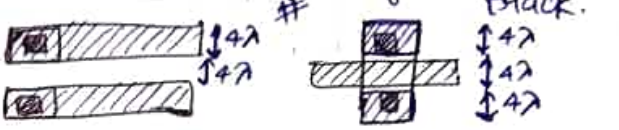
Transistor Width and Length:



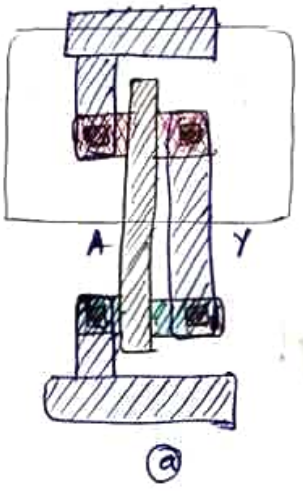
Typically width >> length

Wiring Tracks: Space required for a wire
- ↳ 4λ width, 4λ spacing from neighbour = 8λ pitch
- ↳ Transistors also consume one wiring track
- → Distance b/w 2 transistors : 12λ

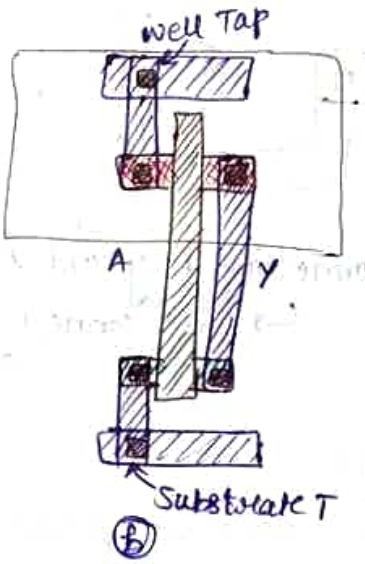
Well spacing: wells must surround transistors by 6λ
- ↳ Implies 12λ b/w opposite transistor flavours
- ↳ Leaves room for one wire track.



Eg. Inverter



(a)

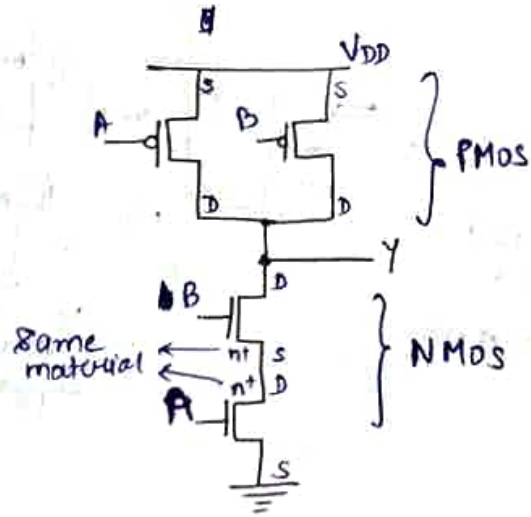


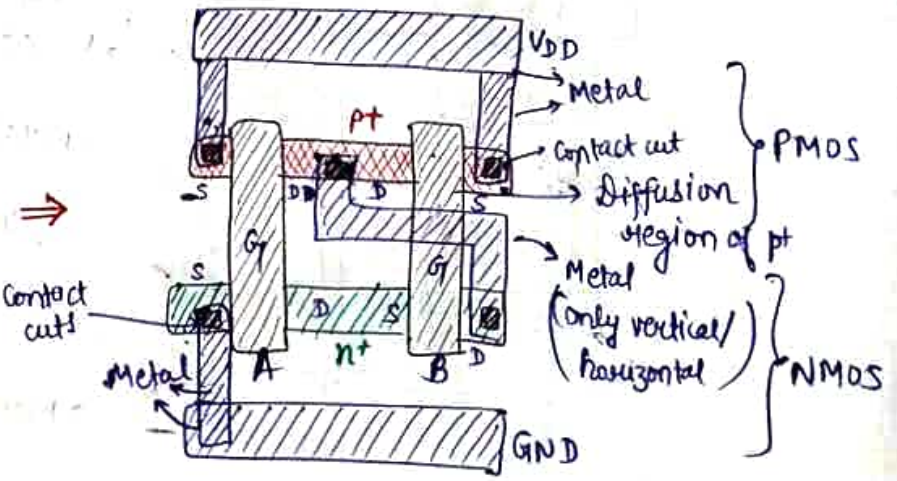
(b)

↔ Body is connected to V_{DD}, using well-tap (cannot be connected directly)

Eg. 2-input NAND Gate

$Y = \overline{AB} = \overline{A} + \overline{B}$





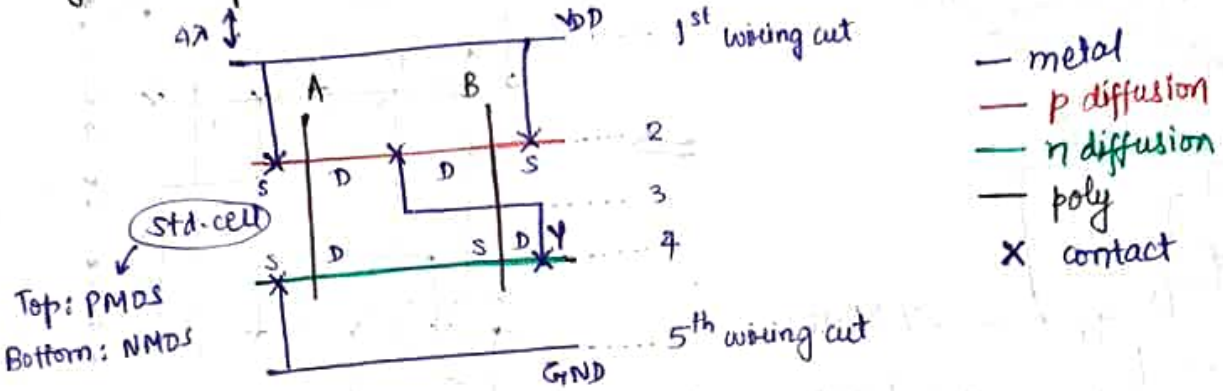
Stick diagrams

↳ Help plan layout quickly.

↳ Need not be to scale.

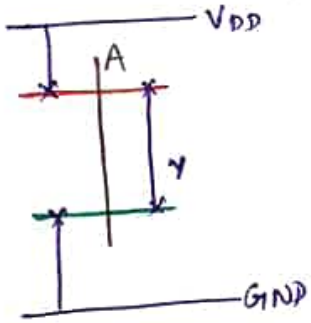
↳ Draw with colour pencils or dry-erase markers.

Eg. 2-input NAND



→ 5×8λ

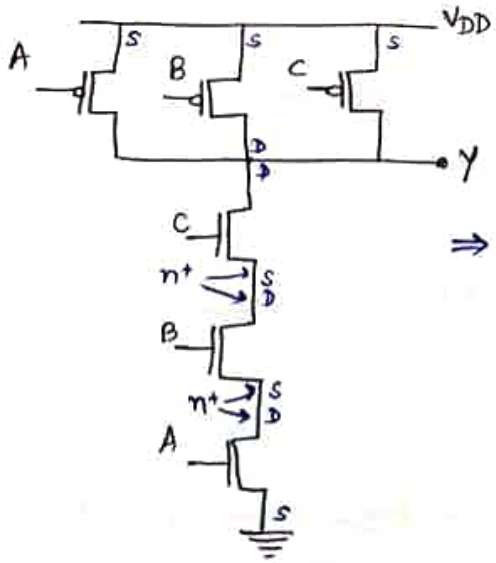
Eg. Inverter



10-11-2023

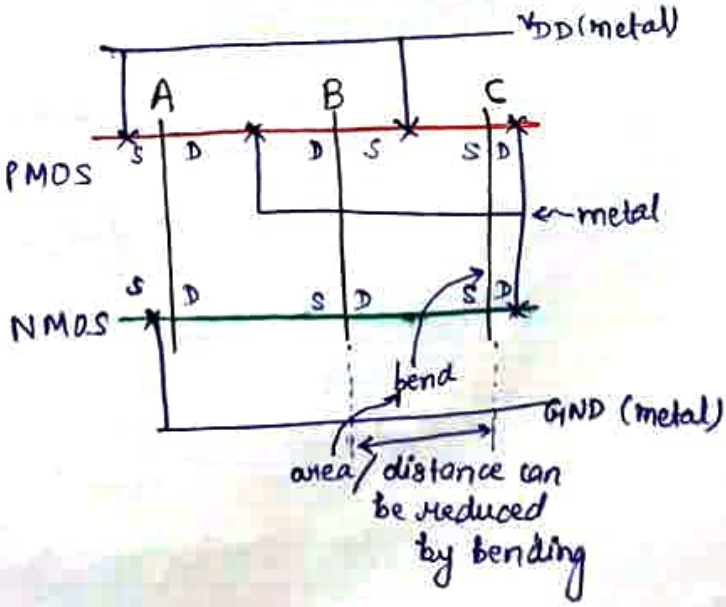
Eg. 3-input NAND Gate

$$Y = \overline{ABC} = \bar{A} + \bar{B} + \bar{C}$$



⇒

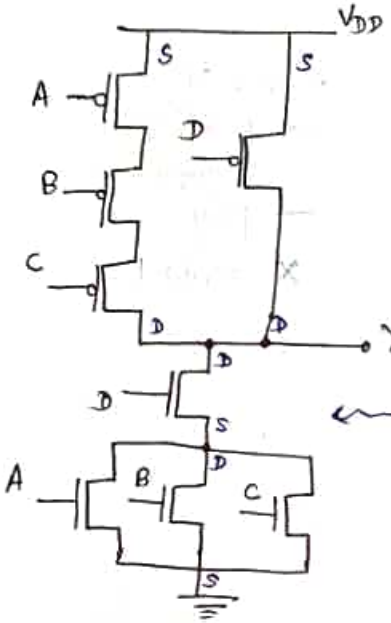
Stick diagram:



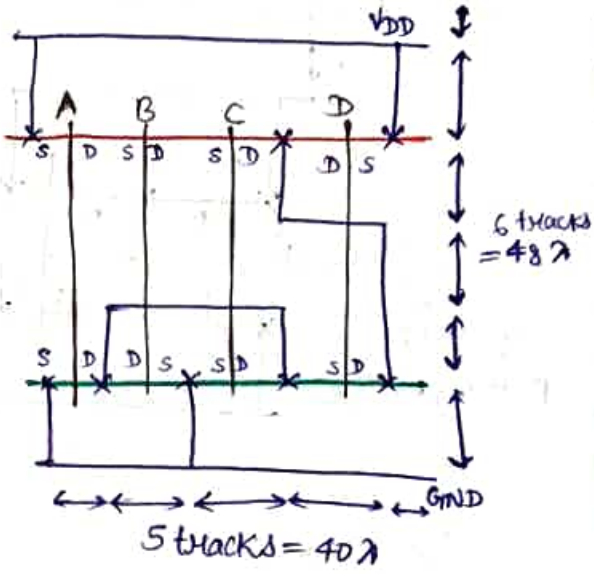
Area Estimation

↳ Estimate area by counting wiring tracks.
  ↳ Multiply by 8 to express in λ.

Eg. $Y = \overline{(A+B+C)D}$



⇒



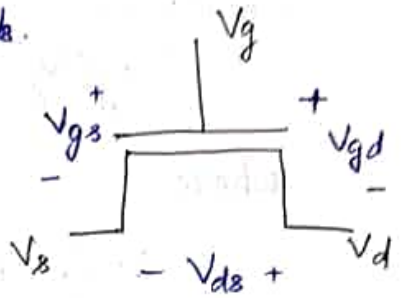
↳ Due to capacitance effect, D is placed above (A+B+C)

CMOS TRANSISTOR THEORY

Terminal Voltages

→ Mode of operation depends on V_g, V_d, V_s.

$$V_{gs} = V_g - V_s$$
$$V_{gd} = V_g - V_d$$
$$V_{ds} = V_d - V_s = V_{gs} - V_{gd}$$



→ Source and drain are symmetric diffusion terminals.

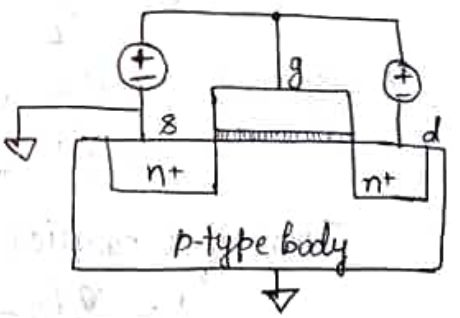
↳ By convention, source is terminal at lower voltage.
Hence, $V_{ds} \geq 0$.

→ nMOS body is grounded.

→ Three regions of operation:

- cutoff
- linear
- Saturation.

nMOS cutoff:



- $V_{gs} < V_t$
- No inversion, no channel
- $I_{ds} \approx 0$

nMOS linear:

- $V_{gs} > V_t$
- V_{ds}: small
- Channel forms.
- Current flows from d to s. ($e^{\ominus}$ from s to d)
- I_{ds} ↑ with V_{ds}.
- Similar to linear transistor.

nMOS Saturation:

- $V_{gs} > V_t$
- $V_{ds} > V_{gs} - V_t$
- Channel pinches off.
- I_{ds} is independent of V_{ds}.
- Current saturates.
- Similar to current source.

Channel Charge

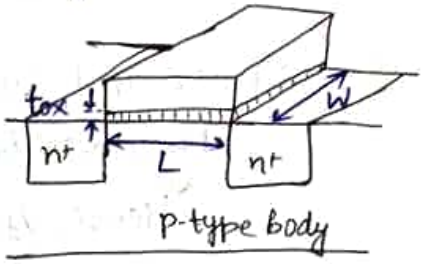
→ MOS structure looks like parallel plate capacitor while operating in inversions.

$Q_{channel} = CV$

$= C_g(V_{gc} - V_t)$; $V_{gc} - V_t$: amount of voltage attracting charge to the channel beyond the min.m required to invert from p to n.

where

$$C = C_g = \epsilon_{ox}\frac{WL}{t_{ox}} = C_{ox}WL, \qquad C_{ox} = \frac{\epsilon_{ox}}{t_{ox}}$$



$$V_{gc} = \frac{(V_{gs} + V_{gd})}{2} = V_{gs} - \frac{V_{ds}}{2} \qquad [V_{ds} = V_d - V_s = V_{gs} - V_{gd}]$$

$$\Rightarrow V_{gc} - V_t = (V_{gs} - V_{ds}/2) - V_t \qquad [V_{gd} = V_{gs} - V_{ds}]$$

→ Electrons are propelled by the ~~ele~~ lateral electric field b/w source and drain.

$$E = \frac{V_{ds}}{L}$$

→ Carrier velocity, v, proportional to lateral E field,

$v = \mu E$, μ: mobility

Time for carrier to cross channel,

$$t = \frac{L}{v} = \frac{L}{\mu E} = \frac{L^2}{\mu V_{ds}}$$

$$\therefore Q_{channel} = C_g[(V_{gs} - V_{ds}/2) - V_t]$$

$$= \epsilon_{ox}\frac{WL}{t_{ox}}[(V_{gs} - V_{ds}/2) - V_t]$$

$$\therefore I_{ds} = \epsilon_{ox}WL\left[(V_{gs} - V_{ds}/2) - V_t\right]\frac{\mu V_{ds}}{L^2 t_{ox}} \qquad \left[\because I_{ds} = \frac{Q_{channel}}{t}\right]$$

$$\Rightarrow \boxed{I_{ds} = \mu C_{ox}\left(\frac{W}{L}\right)(V_{gs} - V_t - V_{ds}/2)V_{ds}}$$

$$= \beta(V_{gs} - V_t - V_{ds}/2)V_{ds}, \text{ where } \beta = \mu C_{ox}\left(\frac{W}{L}\right)$$

↳ Process transconductance

If $V_{gd} < V_t$, channel pinches off near drain.

When $V_{ds} > V_{dsat} = V_{gs} - V_t$.

Now, drain voltage no longer increases current.

$$\therefore I_{ds} = \beta (V_{gs} - V_t - V_{dsat}/2) V_{dsat}$$
$$= \frac{\beta}{2} (V_{gs} - V_t)^2$$

→ ~~Set~~ Shockley 1st order transistor model:

$$I_{ds} = \begin{cases} 0 & , \ V_{gs} < V_t : \text{Cutoff} \\ \beta(V_{gs} - V_t - V_{ds}/2) V_{ds} & , \ V_{ds} < V_{dsat} : \text{Linear} \\ \frac{\beta}{2}(V_{gs} - V_t)^2 & , \ V_{ds} > V_{dsat} : \text{Saturation} \end{cases}$$

Transconductance:

$\beta = \mu_n C_{ox} \frac{W}{L}$: Device transconductance $\left(\beta \propto \frac{W}{L}\right)$

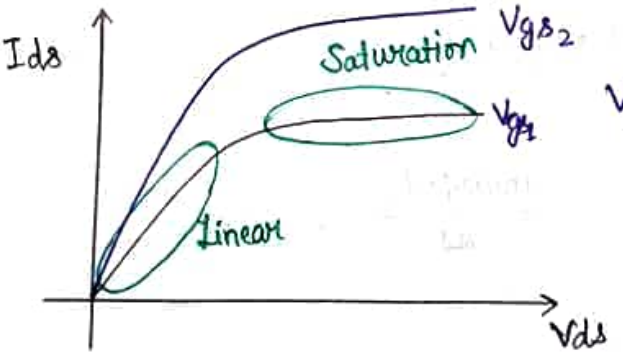
$C_{ox} = \frac{\epsilon_{ox}}{t_{ox}}$: Capacitance per unit area

$k_n = \mu_n \frac{\epsilon_{ox}}{t_{ox}}$: Process transconductance.

14-11-2023

Variable Resistance offered by MOSFET:

Application: Variable resistor by changing transconductance



$V_{gs_2} > V_{gs_1}$

$$R_d = \frac{V_{ds}}{I_{ds}}$$

In linear region,

$$I_{ds} = \beta (V_{gs} - V_t) V_{ds}$$

$$R_D = \frac{1}{\beta(V_{gs} - V_t)} = \frac{1}{\mu C_{ox} \frac{W}{L} (V_{gs} - V_t)} = \frac{L}{\mu C_{ox} W (V_{gs} - V_t)}$$

$\propto L$

$\Rightarrow L$ (technology) ↑ $\Rightarrow R_D$ ↑ & C ↑

$$R_{D,sat.} = \frac{2 V_{DS}}{\beta (V_{GS} - V_T)^2}$$

pMOS I-V

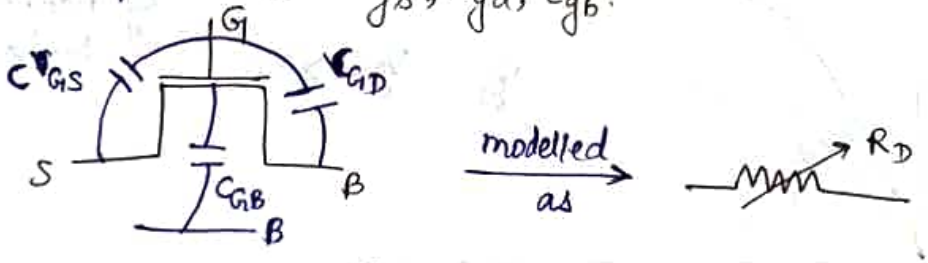
→ All dopings and voltages are inverted for pMOS.
  ↳ Source is more positive terminal.
→ Mobility is determined by holes.
  ↳ Typically 2–3× lower than that of electrons μ_n.
→ Thus, pMOS must be wider to provide same current.
  In this class, assume $\mu_n / \mu_p = 2$.

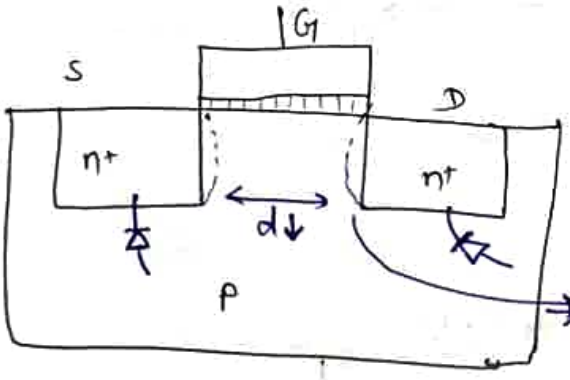
Capacitance

→ Any two conductors separated by an insulator have capacitance.
→ Gate to channel capacitor creates channel charge necessary for operation.
→ Source and drain have capacitance to body.
  ↳ Across reverse-biased diodes.
  ↳ called diffusion capacitance b/c it is associated with source/drain diffusion

Gate capacitance:

↳ Depends on the region of operation.
↳ 3 capacitances: C_{gs}, C_{gd}, C_{gb}.



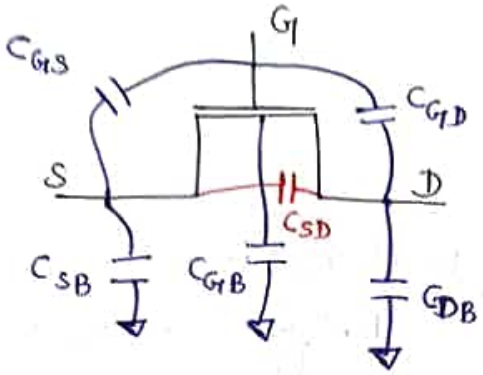


→ Lateral diffusion (due to fabrication)
  ↳ diffusion of D & D into G/Body.

Overlap capacitance: $C_{OVL} = C_{GSO} + C_{GDO}$

→ Diode effect ⇒ Reverse bias (leakage) ⇒ Transistor model
(Transistor: npn)

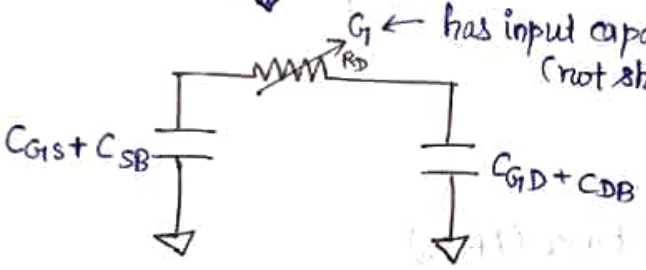
⇒ $I \propto e^{-\frac{V_B}{nV_T}}$ ⇒ Junction / Reverse Bias capacitance
↳ Depends on reverse saturation current.



C_{GS}, C_{GD}, C_{GB} : Contribute for G-capacitance (input)

C_{SD} : not dominant

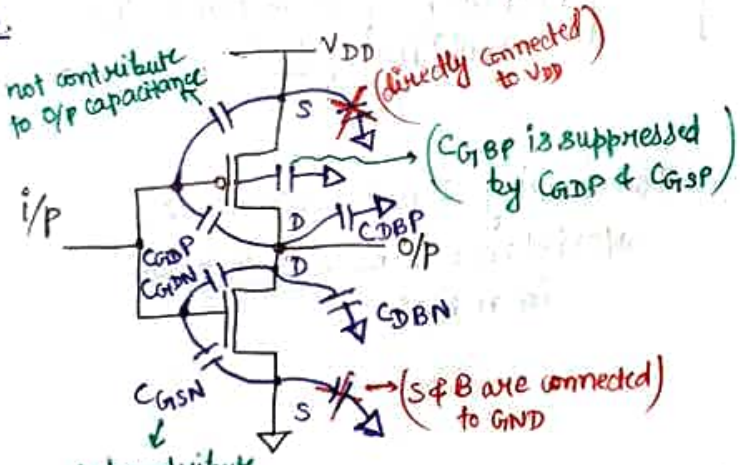
C_{GD}, C_{DB} : contribute for D-capacitance (input)

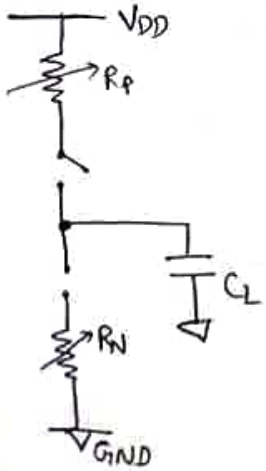


(RC Model of CMOS)

RC Equivalent circuit of CMOS Inverter:

Eg.





$C_L = C_{GDP} + C_{DBP} + C_{GDN} + C_{DBN}$

PMOS: S & B are connected to V_{DD}.

NMOS: S & B are connected to GND.

$R = \frac{L}{\mu C_{ox} W (V_g - V_t)}$; $C_g = \frac{\epsilon WL}{t_{ox}}$ $(\because C = \frac{\epsilon A}{d})$

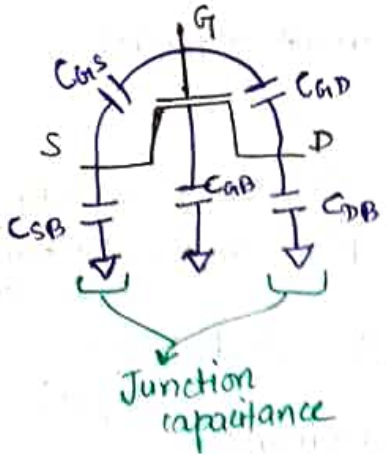
$\tau = RC_g = \frac{\epsilon L^2}{\mu C_{ox}^2 (V_g - V_t)} \propto \boxed{L^2 W^0}$

(Time constant)

↳ τ or propagation delay depends on L only (not on W)

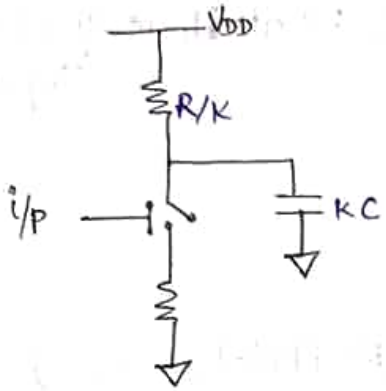
16-11-2023

→



$R \propto \frac{L}{W}$

$C \propto WL$



Scaling: If W changes to WK;

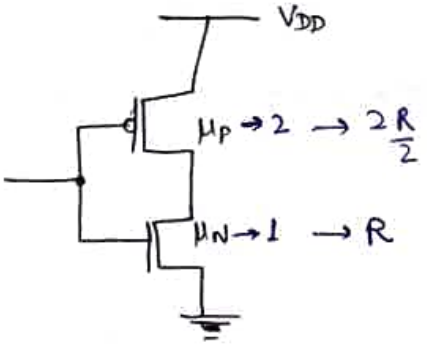
$R \propto \frac{L}{WK}$, $C \propto WKL$.

(K: scaling parameter)

→ Resistance (NMOS) < Resistance (PMOS).

$\mu_N = 2\mu_P$ (mobility)

$\left(\frac{W}{L}\right)_P = \frac{2}{1}$; $\left(\frac{W}{L}\right)_N = \frac{1}{1}$ [Mobility of e^- is twice the mobility of holes.]



→ For having same current I_D, resistance should increase in n-MOS.

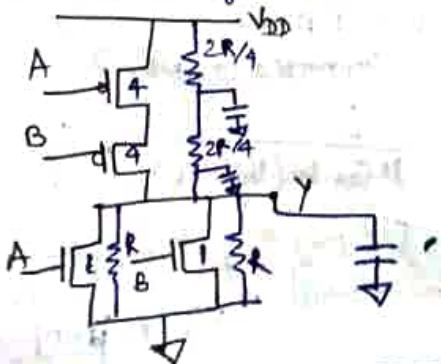
→ If width of NMOS is 1,

then width of PMOS : 2.

Resistance of PMOS : R

Resistance of NMOS : 2R.

Eg.



→ Transistors can be used as switches.

In case of NMOS,

if we give V_{DD} to d and zero (GND) to g, then output will be zero (as MOSFET is not conducting).

↳ So, it is strong zero.

If we give Vdd to g ($g=1$), then ideally o/p (or S) should be 1 (i.e., V_{DD}).

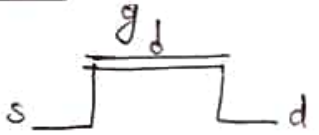
But if $V_s = V_{dd}$, then $V_s - V_g = 0$ → makes MOSFET non-conducting.

So, for conducting, $V_{sg} \geq V_t$.

$\therefore V_{max,s} = V_{DD} - V_t < V_{DD} \Rightarrow$ Weak 1.

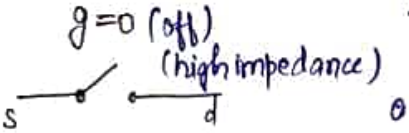
So, NMOS is Strong 0 and Weak 1.

PMOS

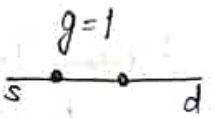


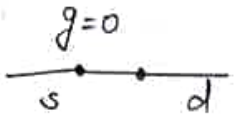
In case of PMOS,

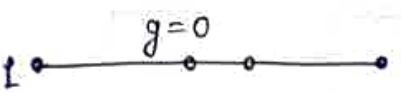
if we give V_{DD} to drain and Vdd to g ($g=1$), then since $V_g = V_{DD}$, i.e., $V_{gs}=0$, it is not-conducting.

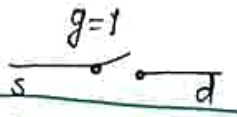


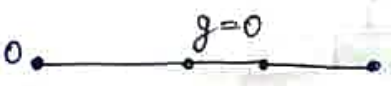
Input g=1 output











PMOS : strong 1, weak 0
NMOS : strong 0, weak 1

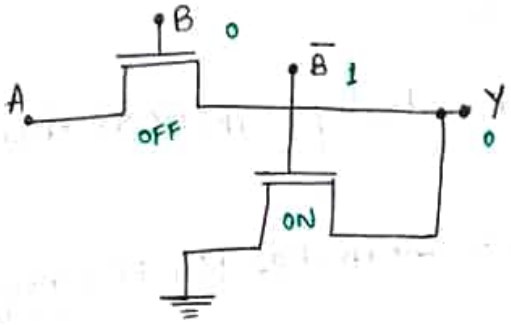
17-11-2023

Pass Transistor

NMOS : strong 0
Weak 1 (V_{DD})

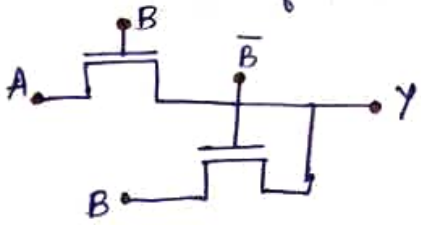
PMOS : Strong 1
Weak 0 ($|V_{TP}|$)

Eg.



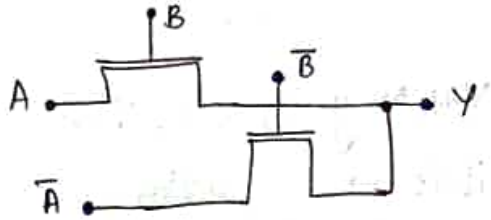
A	B	Y	
0	0	0	} AND
0	1	0	
1	0	0	
1	1	1	

For 2 i/p AND, in CMOS implementation,
no. of transistors = 6 (or >6).
↳ if made separately.



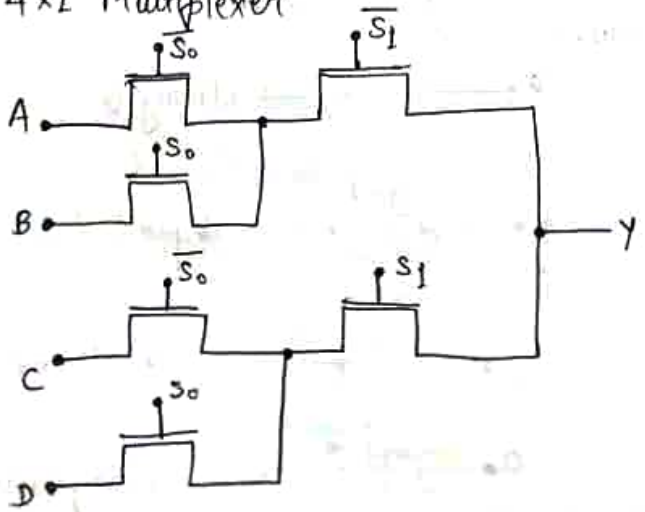
: AND

Eg.



A	B	Y	
0	0	1	
0	1	0	: XNOR
1	0	0	
1	1	1	

Eg. 4×1 Multiplexer

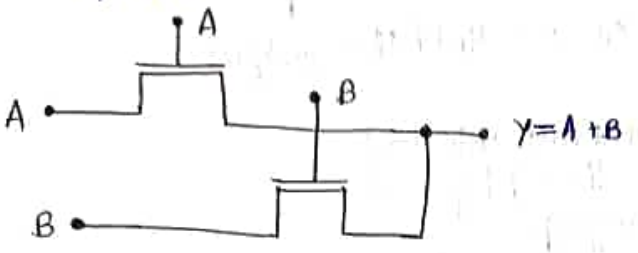


: Pass Transistor logic style

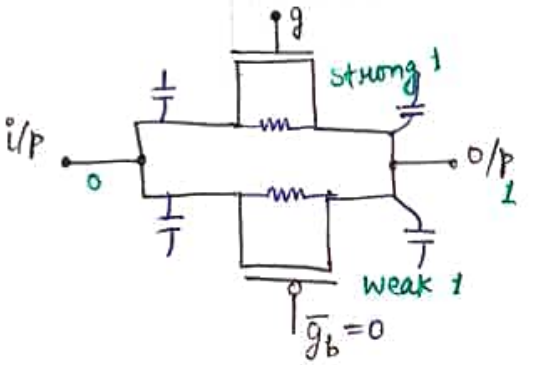
Eg 2 i/p OR Gate

$Y = A + B$

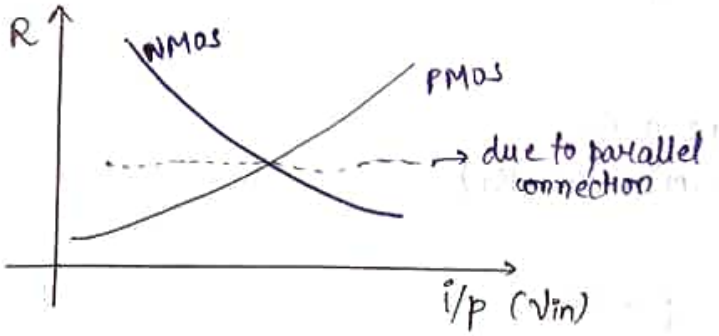
$\overline{Y} = \overline{A}\,\overline{B}$



Transmission Gate → To overcome the disadvantages of pass transistor.



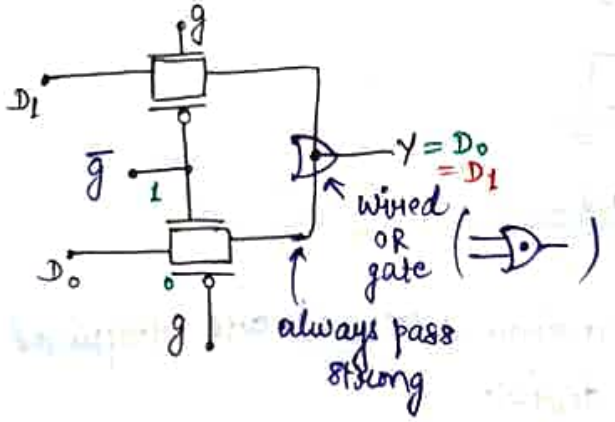
[For parallel connection, resistance/impedance is less]



[we get strong 0 & strong 1]

Symbols of T.G. :

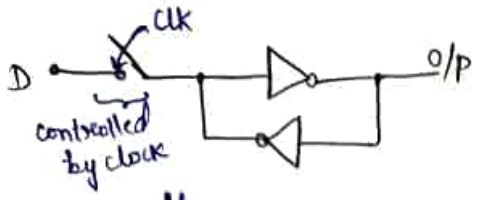
2×1 MUX



Book: Neil Weste
↳ Chapter-1

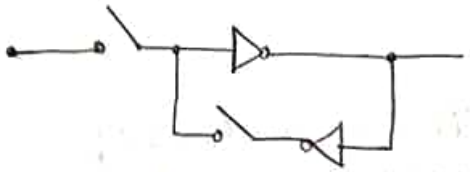
→ To reduce area, we use pass transistor.

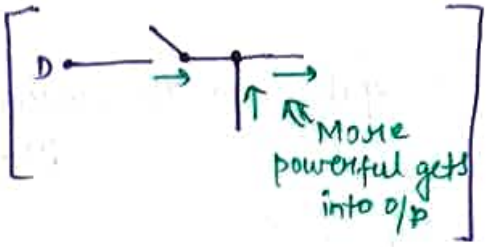
Latch



clk = 1 : Pass the input
clk = 0 : Hold the output.

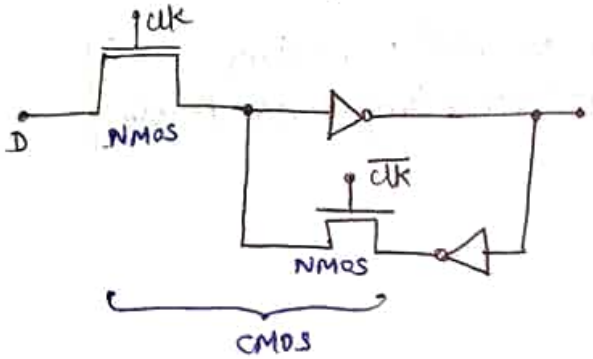
⇓ Introduce a switch (as we are not able to change the o/p by giving i/p)



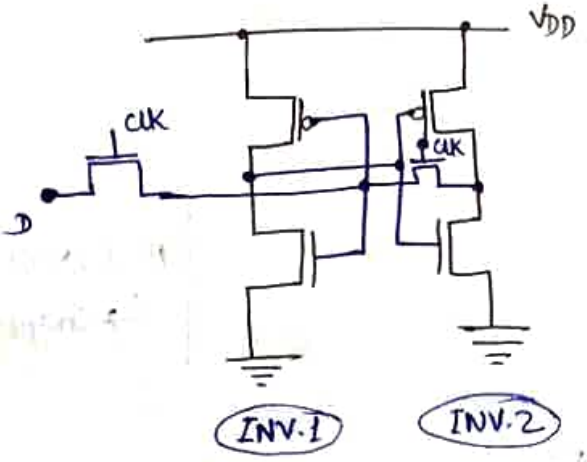
[

]

⇓ Replace switch by NMOS



⇓ Implement gates using CMOS (Inverter)



→ Min^m (6+6) = 12 transistors (CMOS) are required to design the edge triggered device.

→ 1 MB SRAM → 2^{20} × 6 transistors.