17/07/2018

Electronics and digital System

Digital circuits are also called logic circuit because it obeys certain set of code programs. It is used to in computation and data processing, control system, communication.

Advantages of digital electronics -

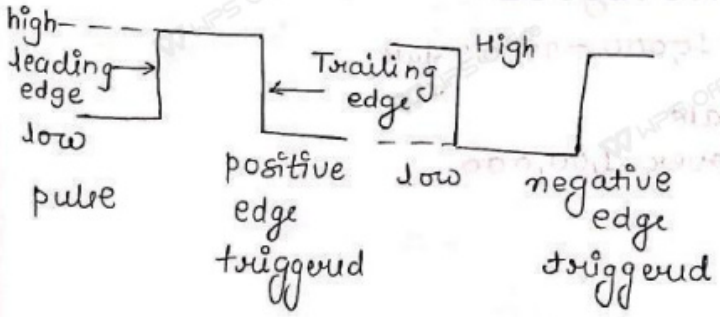
1) easier to design
2) Information storage (is higher)
3) accuracy and precision are greater
4) more versatile (operations are controlled by a set of code instruction called Program)
5) digital circuits are less effected by noise.
6) More digital digital circuit can be fabricated in a single IC.
7) realiability is more.

disadvantage -

1) The real world is analog.

The digital system roughly divided into 3 stages -

1) System design
2) logic design
3) Circuit design



In digital circuits voltage level generally changes from high to low state so pulse are very important in it

elements of digital logic -

Using flip-flops more complex logic circuits like, counter, encoder, decoder, multiplexer, demultiplexer can be constructed.

functions of digital logics -

1) Arithmatic operations
2) encoding
3) decoding
4) storage
5) counting

levels of integration -

1) small scale integrator (SSI)
   - 12 gates on a chip
2) medium scale integration. (MSI)
   - 12 to 99 ~~str~~ gates
3) Large scale integration
   - 100 to 9999 gate (microprocessor)
4) very large scale integration
   - 10,000 - 99,999 gate
5) ultra large scale
   - over 1,00,000

123 in 9's co

```
  999
- 123
-----
  876  — 9's co
+  1
```


782.54 in 9's compliment

```
  999.99
- 782.54
--------
  217.45  — 9's
```

+ 1

217.46 — 10's compliment

Sign bits - 18/07/18

0 1 1 0 0 1 1 → +51

1 0 1 1 0 0 1 1 → -51

for 0 ⊕

for 1 ⊖

①

$367.52_8 \longrightarrow$

8	367	
8	45	7
8	5	5
8		

367 . 5 2

011 110 111 101 010

$(011110111.101010)_2$

$(367.52)_8$

1 0
2
3
4
85
6
7
8
9

$(4057.06)_8 \longrightarrow (\ )_{10}$

$0.06 \times 8 = 0.48$

$0.48 \times 8 = 3.84$

8	4057	
8	507	1
8	63	3
8	7	7
8	0	7

$(7731.03)_{10}$

~~0.6~~

② $(4057.06)_8 \longrightarrow (\ )_{10}$

$= 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 \cdot 08 \times 8^{-1} + 6 \times 8^{-2}$

$= 2095.0937_{10}$

③ $(378.93) = (572.7341)$

8	378
8	47 — 2
8	5 — 7
	5

$0.93 \times 8 = 7.44$
$0.44 \times 8 = 3.52$
$0.52 \times 8 = 4.16$
$0.16 \times 8 = 1.25$

$0.93 = 0.7341$

$(572.7341)_8$

④ - $5497_{10} = (\quad)$

2	5497	
2	2748	1
2	1374	0
2		

8	5497	
8	687	1
8	45	7
	5	5

$(571)_8$

2	571	
2	285	1
2	142	1
2	71	0
2	35	1
2	17	1
2	8	1
2	4	0
2	2	0
	1	

$(100111011)_2$

④ -

8	5497	
8	687	1
8	85	7
8	10	5
	1	2

$(12571)_8 \longrightarrow (001\ 010\ 101\ 111\ 001)_2$

⑤ convert $(1011\ 1101\ 0001)_2$ (11 10 9 8 7 6 5 4 3 2 1 0)

$1 + 0 + 0 + 2^4 + 0 + 2^6 + 2^7 + 2^8 + 2^9 + 0 + 2^{11}$

$= 2048 + 512 + 256 + 128 + 64 + 16 + 1$

$= (3025)_{10}$

Or

$(\underset{5}{1011}\ \underset{7}{1101}\ \underset{2}{0}\ \underset{1}{001})_2$

$5 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0$

$= 3025_{10}$

Hexadecimal = 2^4 = replace with 4 digits

1) $00(1011\ 0111\ 011)_2 \longrightarrow (\quad)_{16}$

2 13 11

$(2DB)_{16}$

0
1
1
1
1
9 —
10 — A
11 — B
12 — C
13 — D
14 — E
15 — F

2) $0(0101\ 1111\ 1011 \cdot 0111\ 1100)_2 = (\quad)_{16}$

2 15 11

$(2FB.7C)_{16}$ ans

0 —
1
2
3
4
5
6

3) $(4BAC)_{16} \longrightarrow (\quad)_2$

$(0100\ 1001\ 1010\ 1100)_2$

7	
8	
9	
10	A
11	B
12	C
13	D
14	E
15	F
16	

4) $5C7_{16} \longrightarrow (\quad)_{10}$

$5\times16^2 + C\times16^1 + 7\times16^0$

$= 1280 + 192 + 7 = 1479$

5) $(A059.0EB)_{16} - (x)_{10}$

$0\times16^{-1} + 14\times16^{-2} + 11\times16^{-3} + 10\times16^3 + 0\times16^2 + 5\times16 + 9\times16^0$

~~$= (41209.0572)_{10}$~~ $(41049.0576)_{10}$ ans

6) $(2598.675)_{10} \longrightarrow (x)_{16}$

16	2598	
16	162	6
16	10	2

A26

$0.675 \times 16 = 10.8$

$0.8 \times 16 = 12.8$

$0.8 \times 16 = 12.8$

$0.8 \times 16 = 12.8$

$(A26.ACCC)_{16}$

7) $(49056)_{10} \longrightarrow (\quad)_2$

16	49056	
16	3066	0
16	191	10
16	11	15

BFA0

$(1011\ 1111\ 1010\ 0000)_2$ ans

8) $(756.603)_8 \longrightarrow (\ )_{16}$

~~$7 \times 8^2 + 5$~~

000 111 101 110 . 110 000 011 000

1 14 14 . 12 1 8

E E C

~~$(1DD.C19)_{16}$~~

$(1EE.C18)_{16}$ ans

9) $(B9F.AE)_{16} = (\ )_8$

1011 1000 1111 . 1010 1100

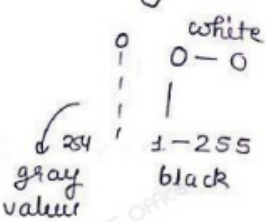
$(5637.534)_8$

$(B9F.AE)_{16} = (5637.534)_8$ ans

19/July/2018

Binary Code

1- Numeric → represents numeric information (only 0's and 1's)
→ 8421, XS-3, Gray Code (used in image processing)
→ Number code is used to represent the decimal digits are called Binary Code Decimal (BCD) codes.



2- Alpha numeric –

is used to represent alphanumeric codes ASCII code.

The 8421 BCD code (Natural BCD code) – It is widely used and each decimal point from zero to 9 is coded by 4 bit binary counter, 8421 weight attach to it.

* Decimal addition using 8421 code

1) 25 + 13

```
  25          0010 0101
+ 13    ⇒     0001 0011
  38          0011 1000
```

If no carry then sum term is not illegal code, if this is carry sum term is illegal, add 6_{10} (0110) if carry and resulting carry is added to next group.

Illegal combination – 1010, 1011, 1100, 1101, 1110, 1111

eg) 1) 679.6 + 536.8 = 1216.4

```
  0110  0111  1001 . 0110
  0101  0011  0110 . 1000
  1011  1010  1111 . 1110
+ 0110  0110  0110 . 0110
  0001  0000  0101
```

add 6

If carry exist then propagate & add.

```
  0001  0000  0101 . 0100
+    1     1     1     1
-------------------------
  0001  0010  0001  0110.0100
-------------------------
   1     2     1     6  .  4
```

This is called BCD addition.

BCD Substraction –

If there is borrow, no correction is needed, if there is borrow then substraction 6's, from the diffrence. this is done

add 6 to skip illegal steps.

a) $\begin{array}{r} 38 \\ -15 \\ \hline 23 \end{array}$ ⇒ $\begin{array}{r} 0011\ \ 1000 \\ -\ 0001\ \ 0101 \\ \hline 0010\ \ 0011 \end{array}$ ← Borrow

b) $\begin{array}{r} 206.7 \\ -\ 147.8 \\ \hline 058.9 \end{array}$

$\begin{array}{r} {}^{1}0\ {}^{1}0\ {}^{1}0 \\ 0010\ \ 0000\ \ 0110\ .\ 0111 \\ 0001\ \ 0100\ \ 0101\ .\ 1000 \\ \hline 0000\ \ 1100\ \ 0000\ .\ 1111 \\ 0000\ \ 0110\ \ 0110\ .\ 0110 \\ \hline 0101\ \ 1000\ .\ 1001 \end{array}$

58.9

again -

$\begin{array}{r} 0010\ \ 0000\ \ 0110\ .\ 0111 \\ 0001\ \ 0100\ \ 0111\ .\ 1000 \\ \hline 0000\ .\ 1111 \end{array}$

$\begin{array}{r} 0010\ \ 0000\ \ 0110\ .\ 0111 \\ 0001\ \ 0100\ \ 0111\ .\ 1000 \\ \hline 0000\ \ 1011\ \ 1110\ .\ 1111 \\ -0110\ -0110\ .\ 0110 \end{array}$

borrows are present on subtrahend

BCD substraction using 9's & 10's complement -

1) $\begin{array}{r} 305.5 \\ -\ 168.8 \\ \hline 136.7 \end{array}$ ⇒ $\begin{array}{r} 305.5 \\ 831.1 \\ \hline 1136.6 \\ +\ 1 \\ \hline 136.7 \end{array}$ (9's complement)

2) $\begin{array}{r} 342.7 \\ -\ 108.9 \\ \hline 233.8 \end{array}$ ⇒ $\begin{array}{r} 342.7 \\ 891.1 \\ \hline 1233.8 \\ \hline 233.8 \end{array}$ (10's complement) ignore the carry

9's complement in BCD -

$\begin{array}{r} 305.5 \\ +\ 831.1 \\ \hline 1136.6 \end{array}$ ⇒ $\begin{array}{r} 0011\ \ 0000\ \ 0101\ .\ 0101 \\ 1000\ \ 0011\ \ 0001\ .\ 0001 \end{array}$

1011 0011 0110 . 0110
+ 0110
0001 0011 0110 . 0110
1 3 6 . 7 ans

{ Illegal 1011 add 0110 it only

24/07/18

BCD substraction using 10's compliment -

342.7
- 108.9
233.8

342.7
+ 891.1 → (10's compliment)
① 233.8 (ignore carry)

0011 0100 0010 . 0111
1000 1001 0001 . 0001
1011 1101 0011 . 1000
+ 0110 + 0110
0010 0011 0011 . 1000
2 33.8
① 0001 ① 00110 0011 . 1000
(propagate carry)
0010 0011 0011 . 1000 (ignore)
2 3 3 . 0

Scanned with CamScanner

Excess-3 ~~(EX-3)~~ C (XS-3) Code -

→ It is a non weighted BCD code.
→ Each binary word is the corresponding 8421 code plus 0011 (3)
→ Six invalid state: 0000, 0001, 1101, 1110, 1111

XS-3 Addition -

1) 37 + 28 = 65 + 33 = 98 ⇒ 6 10 / 5 11 ⇒

0110 1010
0101 1011 XS-3 code
1011 ① 0101
+1
1100 0101
− 0011 + 0011
1001 1000
9 8

If no carry from the addition of any four group subs.
0011 and if

if carry then add 0011.

2) 247.6 + 359.4 = 607.0 + 333.3 = 940.3 → 9310.3

5 7 10.9
6 8 12.7 ⇒

0101 0111 1010.1001
0111 1000 1100.0111
1100 1111 0110.0000
1100 0000 0111.0000
1101 0000 0111.0000
1001 1101 0011.1101
1001 0011 1010.0011
9 3 10.3

XS-3 substraction-

*no invalid step will come.

ans

3) 267 − 175 = 092 + 333 = 3125

5 9 10 − 4 10 8 ⇒

0101 1001 1010
− 0100 1010 1000
0000 1111 0010
+0011 −0011 +0011
0011 1100 0101
3 12 5

carry)

If illegal if borrow substract 0011
no borrow add 0011

4) 57.6 − 27.8 = 29.8 + 33.3 = 5 12 11

8 10.9
5 10.11 ⇒

1000 1010.1001
− 0101 1010.1011
0010 1111.1110
+0011 −0011.−0011
0101 1100.1011
5 12.11

25/07/18

Binary to Gray conversion-

- If an n-bit binary number is represented by $B_n, B_{n-1}, \dots, B_1$ & its gray code equivalent is G_n G

$G_n, G_{n-1} \dots G_1$, while B_n & G_n are MSBs

then $G_n = B_n$, $G_{n-1} = B_n \oplus B_{n-1}$; $G_{n-2} = B_{n-1} \oplus B_{n-2}$ ---

---; $G_1 = B_2 \oplus B_1$

* convert the binary 1001 to the gray code.

Binary : 1 ⊕ 0 ⊕ 0 ⊕ 1

Gray : 1 1 0 1

other method-

Binary - 1 0 0 1

shift Binary - 1 0 0 1

1 1 0 1 ① (ignore this)

1101

Gray to Binary-

$B_n = G_n$; $B_{n-1} = B_n \oplus G_{n-1}$; $B_{n-2} = B_{n-1} \oplus G_{n-2}$ ------

$B_1 = B_2 \oplus G_1$

* convert Gray code 1101 to binary.

G - 1 1 0 1

B - 1 0 0 1

Q)- Convert the no. into gray.

$3A7_{16}$ 0011 1010 0111

grey - 0010 0111 0100

by shifting method-

0011 1010 0111

0010 1101 0011 ① (ignore)

0010 0111 0100

(don't take

(don't take carry and borrow)

Error detecting Code -

When Binary data is dtected error may occur due to noise which distord its content, so detection of error is very important. Parity is the simplest technique to dtecting error by adding an extra bit - There are two types of parity -

1) odd parity - Total no. of 1 bit in one's word.
2) even parity - Total no. of 1 bit in one's word.

* On the even scheme, find the error.

10101010 ; 11110110 ; 10111001

No error No error error word.

Error correcting code - A code is said to be an error correcting code, if the correct-code word can always be deduced from an erroneous word.

for a single bit error correcting code must be three. It corrects single bit error ~~but~~ and also detected 2 bit error.

eg) hamming Code -

7-bit hamming Code -

- The word format

P1 P2 D3 P4 D5 D6

P - Parity bits
D - Data bits

- P1 is to be set a '0' or '1' so that it establishes every parity over bits 1, 3, 5, 7 (P1, D3, D5, D7)
- P2 " " for 2, 3, 6, 7 (P2, D3, D6, D7)
- P3 " " for 4, 5, 6, 7 (P4, D5, D6, D7)

	C_3	C_2	C_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

- If there is an error, bit can be located by forming a 3-bit binary number C_3, C_2, C_1 where $C_1 = P_1 \oplus D_3 \oplus D_5 \oplus D_7$

$C_2 = P_2 \oplus D_3 \oplus D_6 \oplus D_7$

$C_3 = P_4 \oplus D_5 \oplus D_6 \oplus D_7$

* Encode data bits 1101 into the 7 bit even parity hamming code.

Soln- Bit Pattern P_1 P_2 D_3 P_4 D_5 D_6 D_7

1 1 0 1

Bits 1,3,5,7 (ie P_1 1 1 1) so P_1 must '1'.

Bits 2,3,6,7 (ie P_2 1 0 1) so P_2 must '0'.

Bits 4,5,6,7) (ie P_4 1 0 1) so P_4 must be '0'.

final Code- 1 0 1 0 1 0 1

* The message below coded in the 7-bit hamming code, is transmitted through a noisy channel. Decode the message assuming that at most a single error occurd in each code word. 1001001, 0111001, 1110110, 0011011

Solu^n Total 28-bit, four groups of 7-bit each & correct for error $P_1, P_2, P_3, D_4, D_5, D_6, D_7$

1st group- 1,3,5,7 (1 0 0 1) → No error, so put '0' in 1's position $C_1 = 0$.

$C_1 = 0$

2,3,6,7 (0 0 0 1) → error, put '1' in its position $C_2 = 1$

4, 5, 6, 7 (1001) → no error '0' $C_3 = 0$.

→ so error word is $C_3 C_2 C_1 = 010$

$= 2_{10}$

Thus complement 2nd bit from left i.e 1101001

1 2 3 4 5 6 7
0 1 1 1 0 0 1

2nd group - 1, 3, 5, 7 (0101) — no error $C_1 = 0$

2, 3, 6, 7 (1101) — error $C_2 = 1$

4, 5, 6, 7 (1001) — no error $C_3 = 0$.

So error word is $C_3 C_2 C_1 = 010$

$= 2_{10}$

Thus compliment 2nd bit from [illegible]

3rd group 1 2 3 4 5 6
1 1 1 0 1 1 0

1 2 3 4 5 6 7

1, 3, 5, 7 (1 1 1 0) — error $C_1 = 1$

2, 3, 6, 7 (1 1 1 0) — error $C_2 = 1$

4, 5, 6, 7 (0 1 1 0) — no error $C_3 = 0$

011

$= 3_{10}$

= 1100110.

4th group 1 2 3 4 5 6 7
0 0 1 1 0 1 1

1, 3, 5, 7 (0, 0, 1, 1) error $C_1 = 0$

2, 3, 6, 7 (0 1 1 1) error $C_2 = 1$

4, 5, 6, 7 (1 0 1 1) error $C_3 = 1$

= 110

$= 6_{10}$

The compliment of 7th bit from left -

= 0011010

= 0011001.

So decoded message is

1101001 , 0011001, 1100110, 0011001

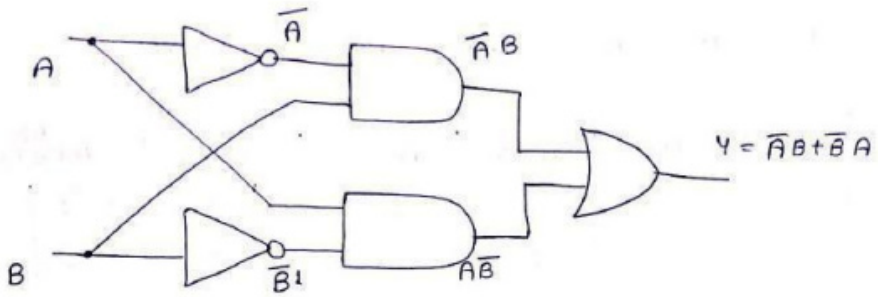
XOR gate -

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

Truth table

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

mplementation -



XNOR-

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

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

* Find logical equivalent (prove them too)

(a) $A \oplus 0 = A$

(b) $A \oplus 1 = \bar{A}$

(c) $A \odot 0 = \bar{A}$

(d) $A \odot 1 = A$

(e) $1 \oplus \bar{A} = 1 \cdot \bar{A} + \bar{1} \cdot \bar{A} = A + 0 = A$

a)

A	0	$A \oplus 0$
0	0	0
0	0	0
1	0	1
1	0	1

b)

A	1	$A \oplus 1$
0	1	1
0	1	1
1	1	0

1 | 1 | 0

* Show that $A \oplus B = A\bar{B} + B\bar{A}$ & construct the logic diagram

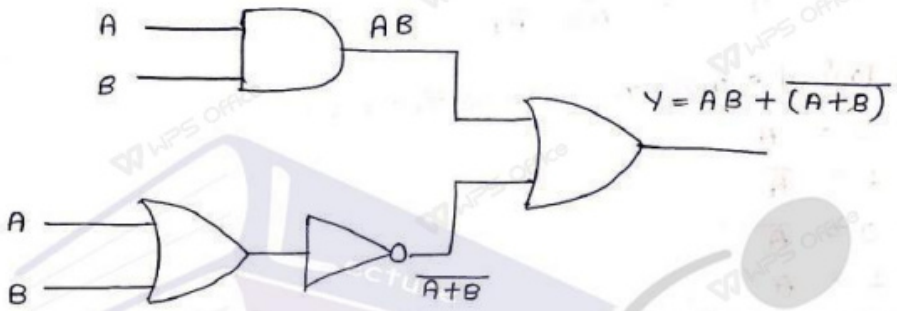
A	$\bar{A}$	B	$\bar{B}$	$A \oplus B$	$A\bar{B}$	$B\bar{A}$	$A\bar{B} + B\bar{A}$
0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	1	1	0	1
1	0	1	0	0	0	0	0

In exam make two separate table of RHS and LHS

* Show that $AB + \overline{(A+B)} = A \odot B$

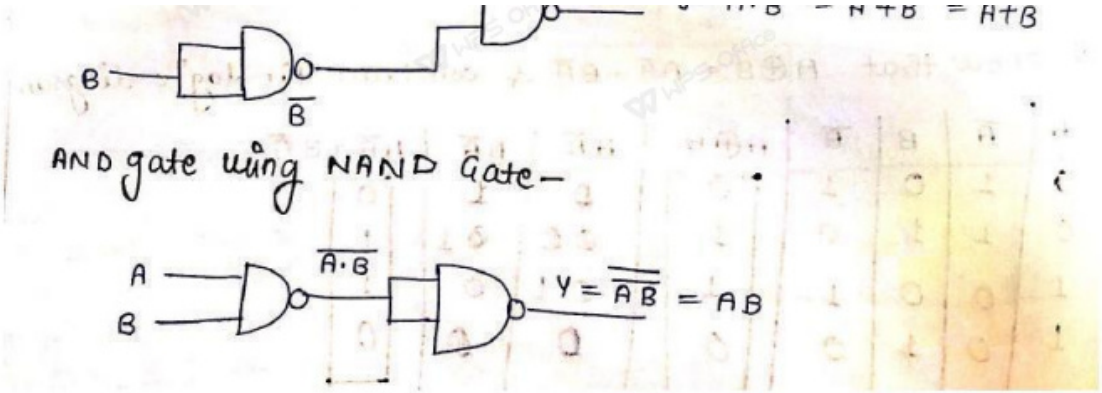
~~A B~~ ~~$\bar{A}$ $\bar{B}$~~ ~~$\bar{A}$ B~~ ~~$A\bar{B}$~~ ~~$A\bar{B} + \bar{A}B$~~

A	B	AB	A+B	$\overline{A+B}$	$A \odot B$	$\overline{(A+B)} + \overline{AB}$
0	0	0	0	1	1	1
0	1	0	1	0	0	0
1	0	0	1	0	0	0
1	1	1	1	0	1	1



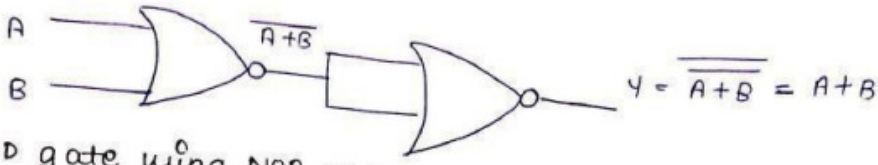
31/7/18

Design or gate using NAND gate-

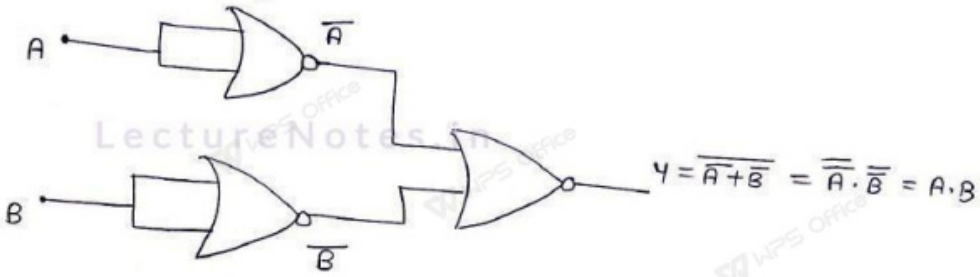


AND gate using NAND Gate-

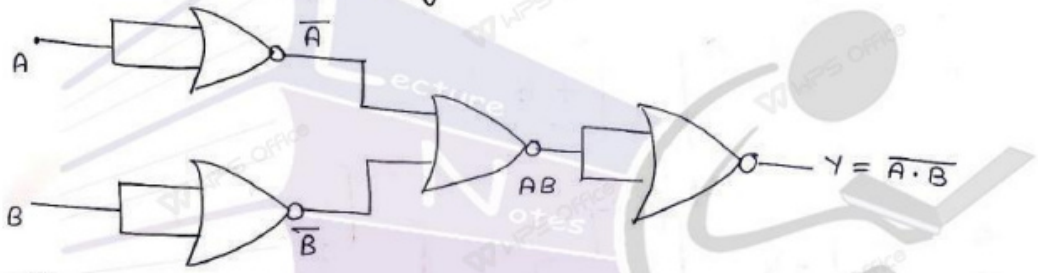
OR Gate using NOR gate-



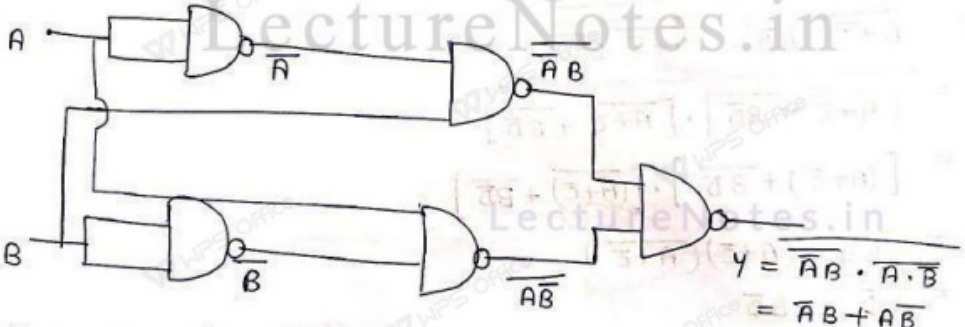
AND gate using NOR gate-



NAND Gate using NOR gate-



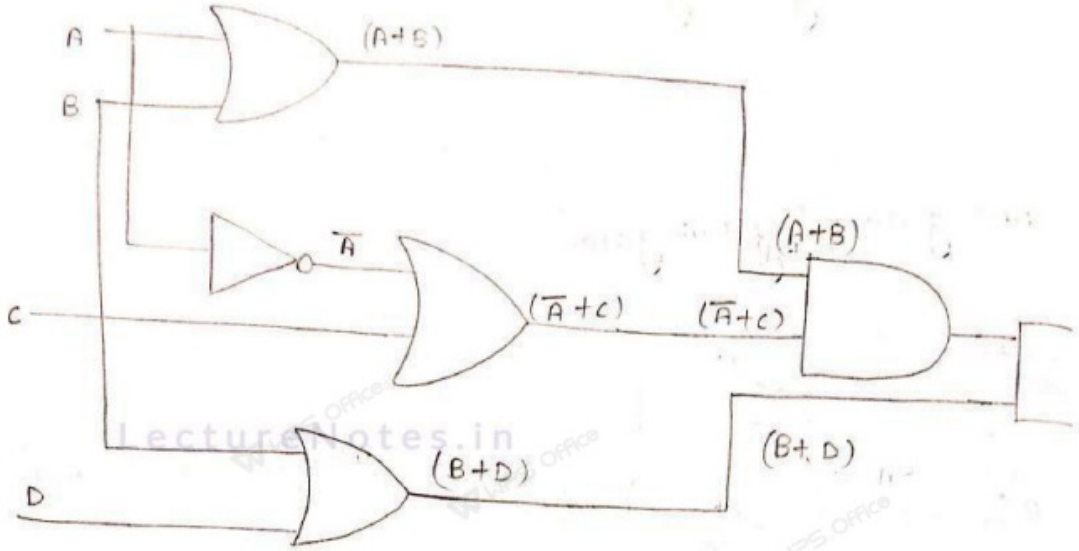
X-OR gate using NAND Gate-



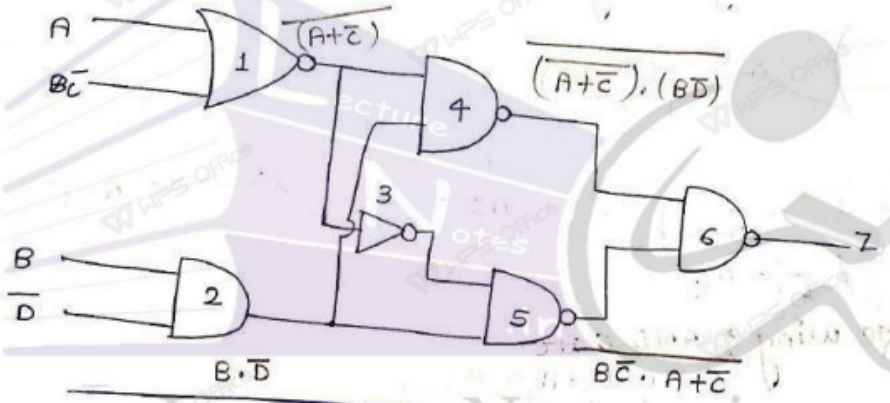
* Realize the logic

Realize the logic expression using basic gates

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



* Simplify the logic circuit –



$$Z = \overline{\overline{A+\bar{C}.B\bar{D}} \cdot \overline{A+\bar{C}\ B\bar{D}}}$$

(Do it again)

$$= \overline{\left[\overline{\overline{A+\bar{C}} + \overline{B\bar{D}}}\right] \cdot \left[\overline{A+\bar{C}} + \overline{B\bar{D}}\right]}$$

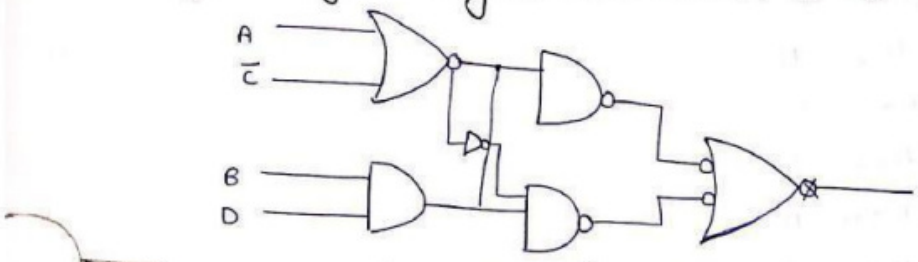
$$= \left[(A+\bar{C}) + \overline{B\bar{D}}\right] \cdot \left[\overline{(A+\bar{C})} + \overline{B\bar{D}}\right]$$

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

$$= \overline{\overline{B\bar{D}}} = B\bar{D}$$

* using logic gate

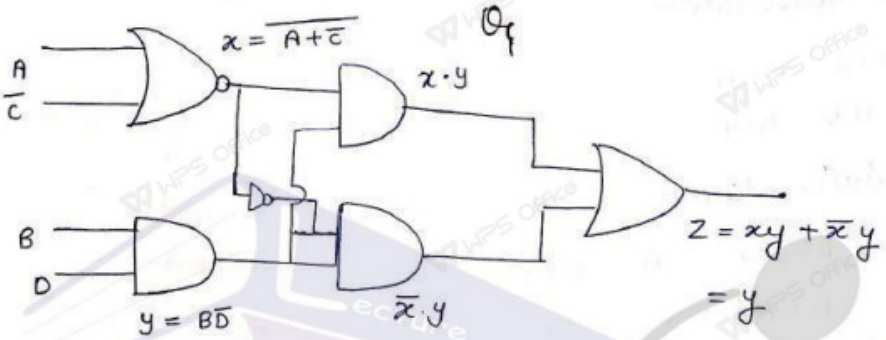
Replace gate 6 by an OR gate with bubbled at its i/p.



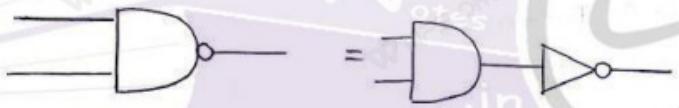
$Y = (A+B)(\bar{A}+C)(B+D)$

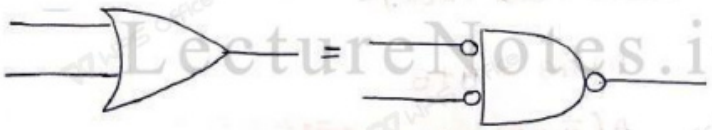
Now using $\overline{\bar{A}} = A$

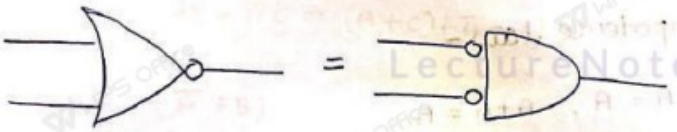
Q)

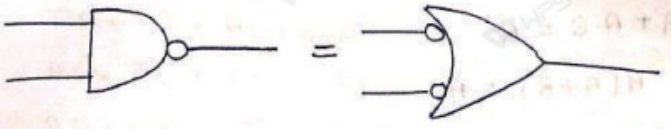


01|08|18









Boolean algebra -

1) $A \cdot 0 = 0$
2) $A \cdot 1 = A$
3) $A + 0 = A$
4) $A + 1 = 1$
5) $A + A = A$
6) $A + \overline{A} = 1$

commutative law -

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

Associative law -

$$(A+B)+C = A+(B+C)$$
$$(A \cdot B) \cdot C = A \cdot (BC)$$

Distributive law -

$$A(B+C) = AB + AC$$
$$A + BC = (A+B)(A+C)$$

Redundant Literal Rule (RLR) -

law 1 - $A + \overline{A}B = A + B$

law 2 - $A(\overline{A} + B) = AB$

Idempotence law -

$$A \cdot A = A, \quad A + A = A$$

Absorption law -

$$A + A \cdot B = A$$
$$A(A+B) = A$$

Consensus law -

$$AB + \overline{A}C + BC = AB + \overline{A}C$$

LHS $= AB + \bar{A}C + BC$

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

$= AB + \bar{A}C + ABC + BC\bar{A}$

$= AB(1+C) + \bar{A}C(1+B)$

$= AB + \bar{A}C$ R.H.S.

Extended Theorm –

① $AB + \bar{A}C + BCD = AB + \bar{A}C$

② $(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$

L.H.S $= (A+B)(\bar{A}+C)(B+C)$

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

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

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

$= ABC + ACC + \bar{A}BB + \bar{A}BC + BBC + BCC$

$= ABC + AC + \bar{A}B + \bar{A}BC + BC + BC$

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

$= BC(A+\bar{A}) + AC + \bar{A}B + BC$

$= BC + AC + \bar{A}B$

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

R.H.S

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

$= AC + B\bar{A} + BC$

L.H.S = R.H.S

Transposition theorm –

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

RHS

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

$= A\bar{A} + AB + C\bar{A} + CB$

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

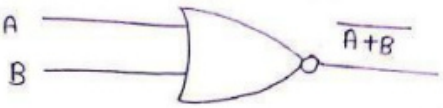
$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB + ABC + \bar{A}C + \bar{A}BC$

$= AB + \bar{A}C$ LHS

De-Morgan's theorm -

Law 1 - $\overline{A+B} = \overline{A} \cdot \overline{B}$



A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

=

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Law 2 - $\overline{AB} = \overline{A} + \overline{B}$

=

LHS -

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

RHS -

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Duality Theorm - $A+0 = A$, $A \cdot 1 = A$

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

$AB + \overline{A}C + BC = AB + \overline{A}C$

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

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

* Prove that -

$$A(\bar{A}+C)(\bar{A}B+C)(\bar{A}BC+\bar{C}) = 0$$

L.H.S

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

$= (A\cdot\bar{A}+AC)(\bar{A}B+C)(\bar{A}BC+\bar{C})$

$= (AC\cdot\bar{A}B+AC\cdot C)(\bar{A}BC+\bar{C})$

$= AC(\bar{A}BC+\bar{C})$

$= A\bar{A}\cdot BCC + \bar{C}\cdot AC$

$= 0$

R.H.S

* Simplify $y = (A+B)\overline{(\bar{A}\cdot(\bar{B}+\bar{C}))} + \bar{A}(B+C)$

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

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

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

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

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

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

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

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

$= A+AB(C+1)+\bar{A}(B+C)+BC$

$= A+AB(C+1)+\bar{A}(B+C)+BC$

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

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

$= A+ABC+B+\bar{A}C+BC \Rightarrow A+C+B+AB = A+BC$ proved

$= A+BC(A+1)+\bar{A}C+B$

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

$= A+B(C+1)+\bar{A}C = A+B+\bar{A}C$

* Simplify $AB + \overline{AC} + A\overline{B}C\,(AB+C)$

$= AB + \overline{A}C + A\overline{B}C \cdot A \cdot B + A\overline{B}C \cdot C$

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

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

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

$= AB + AC + \overline{A}C$

$= C(A + \overline{A}) + AB$

$= AB + C$

$AB + \overline{AC} + A\overline{B}C \cdot AB + A\overline{B}C$

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

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

$= (A+B) +$

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

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

$= AB + AC + \overline{A} + \overline{C}$

$=$

solution -

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

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

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

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

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

$= A + \overline{A} + \overline{C}$

$= 1 + \overline{C}$

$= 1$

* $\overline{A\overline{B} + ABC} + A(B + A\overline{B})$

solution -

$= \overline{(A\overline{B} + ABC)} \cdot \overline{A(A+B)}$

$= (A\overline{B} + ABC)\left(\overline{A} + \overline{(A+B)}\right)$

$= (A\overline{B} + ABC)\left(\overline{A} + \overline{A} \cdot \overline{B}\right)$

$= \overline{A}\left(A\overline{B} + ABC\right)$

$= A \cdot \overline{A}\left(\overline{B} + BC\right)$

$= 0$.

* Reduce the Boolean expression .

$f = A\left[B + \bar{C}\left(\overline{AB + A\bar{C}}\right)\right]$

Solution-

$= A\left[B + \bar{C}\left(\overline{AB} \cdot \overline{A\bar{C}}\right)\right]$

$= A\left[B + \bar{C}\left(\overline{AB} \cdot (\bar{A} + C)\right)\right]$

$= A\left[B + \bar{C}\,(A+B)(\bar{A}+C)\right]$

$= A\left[B + \bar{C}\left(A \cdot \bar{A} + AC + B\bar{A} + BC\right)\right]$

$= AB$ Ans.

Show that $AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$

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

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

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

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

$= AB + AC + B\bar{C}$

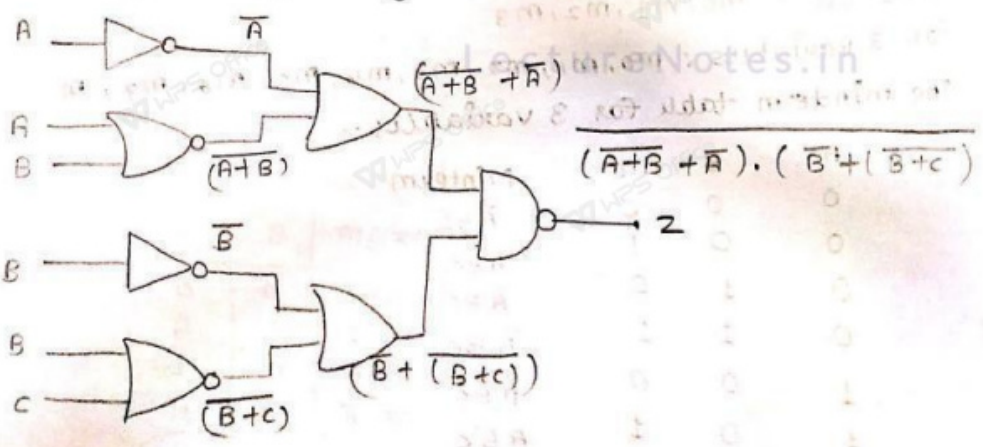
$= AB(C+\bar{C}) + AC + B\bar{C}$

$= ABC + AB\bar{C} + AC + B\bar{C}$

$= AC(B+1) + B\bar{C}(A+1)$

$= AC + B\bar{C}$

* Simplify the logic diagram & draw the reduced one-



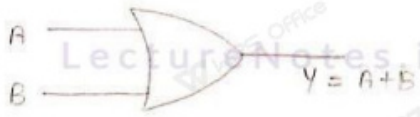
$= \overline{(\overline{A+B} + \bar{A})\,(\bar{B} + \overline{B+C})}$

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

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

$= \overline{\bar{A}(\bar{B}+1)} + \overline{\bar{B}(\bar{C}+1)}$

$= A + B$



Reduced logic diagram

Sum of Product & Product of sum–

SOP POS

SOP ; $y = AB + BC + AC$

POS ; $y = (A+B)(B+C)(A+\bar{C})$

Minterm– a product term containing k variable of the function in either complimented or uncomplimented form is known as minterm. A 2 variable function has 4 possibilities $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$, AB, these are called standard product, fundamental product or minterm-product.

2 variable – m_0, m_1, m_2, m_3

for 3 variables : $m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7$,

The minterm table for 3 variables –

A	B	C	Minterm.
0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	$\bar{A}\bar{B}C$
0	1	0	$\bar{A}B\bar{C}$
0	1	1	$\bar{A}BC$
1	0	0	$A\bar{B}\bar{C}$
1	0	1	$A\bar{B}C$
1	1	0	$AB\bar{C}$
1	1	1	ABC

Canonical Sum of Products expression also called minterm canonical form.

- A 3-variable logic function Y has 3 minterm $\bar{A}\bar{B}\bar{C}$, $A\bar{B}C$ & $AB\bar{C}$
- corresponding minterm.

$Y = \Sigma m\ (0, 5, 6)$

$= m_0 + m_5 + m_6$

$= \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + AB\overline{C}$

* obtain the canonical SOP form of the function

$Y(A,B) = A+B$

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

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

$Y(A,B) = AB + A\overline{B} + \overline{A}B$

obtain the SOP form-

$Y = AB + ACD$

$y = AB(C+\overline{C})(D+\overline{D}) + A(B+\overline{B})CD$

$y = AB(CD + C\overline{D} + \overline{C}D + \overline{C}\,\overline{D}) + (AB + A\overline{B})CD$

$y = ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} + A\overline{B}CD$

$y = ABCD + ABC\overline{D} + ABC\overline{D} + AB\overline{CD}$

$Y = \Sigma m = (15, 14, 13, 12, 11)$ (minterm)

$\Sigma m = (11, 12, 13, 14, 15)$

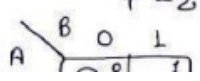
maxterm :-

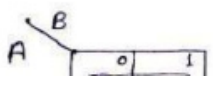
2-variable-

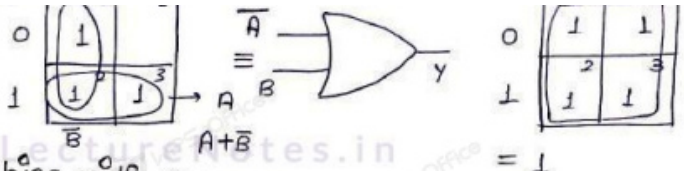
A	B	maxterm
0	0	$A+B$
0	1	$A+\overline{B}$
1	0	$\overline{A}+B$
1	1	$\overline{A}+\overline{B}$

2-variable K map

$f = \Sigma m(0, 2, 3)$







mapping with POS-

$f = \pi M(0,2,3)$ $\quad f = \Sigma m(0,2,3)$

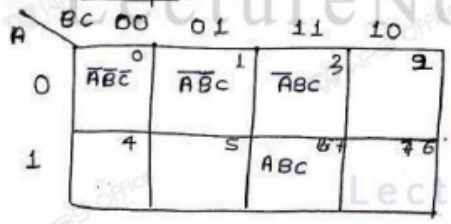
minterm $\Sigma m(1)$ $\quad m(0,1,2,3)$

maxterm πM $\quad = m_0 + m_1 + m_2 + m_3$ — SOP

$M(0,1,2,3)$

$M_0 \cdot M_1 \cdot M_2 \cdot M_3 \rightarrow$ POS

3 variable K-map -



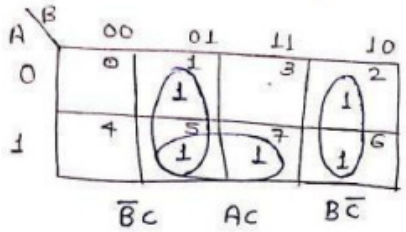
They are in grey code to ~~adjust~~ adjacent squares are really adjecent ie minterm and maxterm differ by one variable. (sequence of 0,1,3,2 is different, answer is above line)

* Map the expression -

$f = \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC$

001 101 010 110 111

$f = \Sigma m(1,2,5,6,7)$

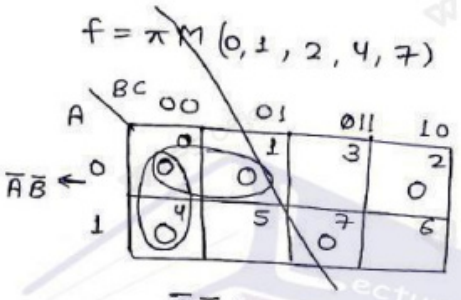


$f = AC + \bar{B}C + B\bar{C}$

* $f = (A+B+c)(\overline{A}+B+\overline{c})(\overline{A}+\overline{B}+\overline{c})(\overline{A}+\overline{B}+c)(\overline{A+\overline{B}+c})$

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

$f = \pi M(0,1,2,4,7)$



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

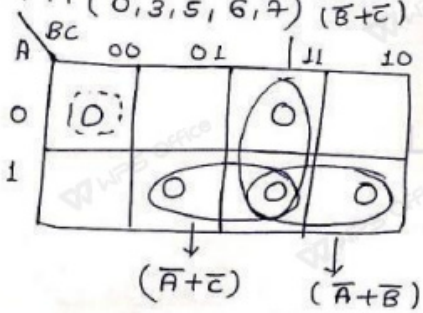
Solution –

$f = (A+B+c)(\overline{A}+B+\overline{c})(\overline{A}+\overline{B}+\overline{c})(A+\overline{B}+\overline{c})(\overline{A}+\overline{B}+c)$

$= (000)(101)(111)(011)(110)$

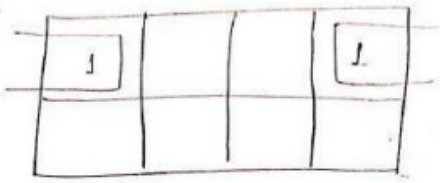
$= 0,5,7,3,6$

$f = \pi M(0,3,5,6,7)$



$(A+B+c)(\overline{A}+\overline{c})(\overline{B}+\overline{c})(\overline{A}+\overline{B})$

Note



possible to pair.

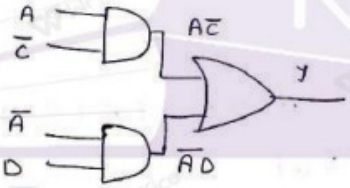
* Simply using K-map –

$y = m_1 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{12} + m_{13}$

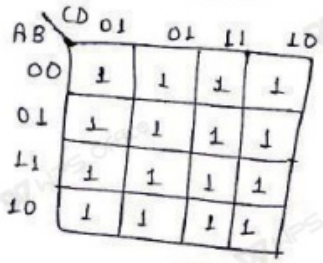
$y = \sum m(1,3,5,7,8,9,12,13)$

$A\bar{C}$

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



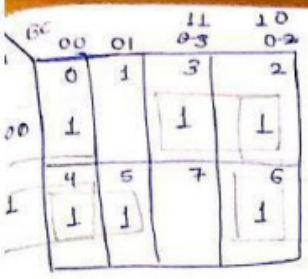
* If all 16 are true -



$Y = 1$

09|08|18

* Reduce the expression $f = \sum m(0,2,3,4,5,6)$ using mapping and impliment in AOI logic as well as AND OR INVERTER NAND logic



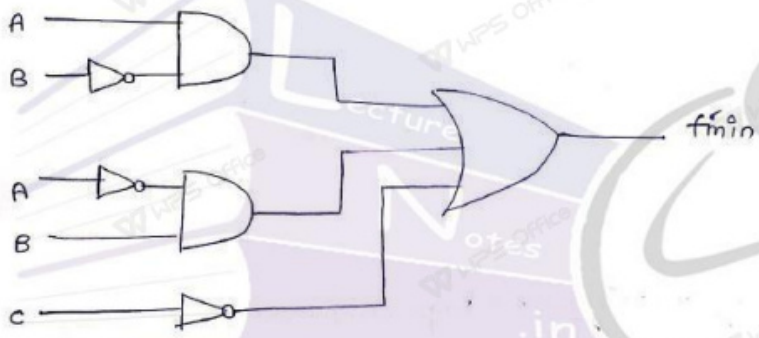
$\bar{A}\bar{B}\bar{C}$
$A\bar{B}\bar{C}$
$A\bar{B}C$
$A\bar{B}\bar{C}$

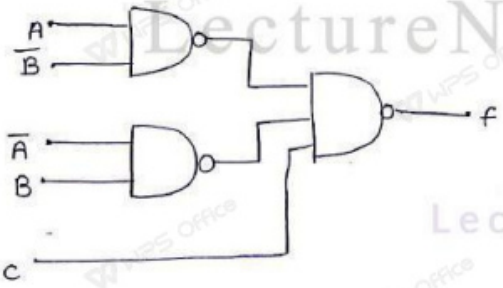
$\bar{A}\bar{B}\bar{C}$
$A\bar{B}\bar{C}$
$\bar{A}B\bar{C}$
$AB\bar{C}$
$+\bar{C}$

$A\bar{B}\bar{C}$
$A\bar{B}C$
$A\bar{B}$

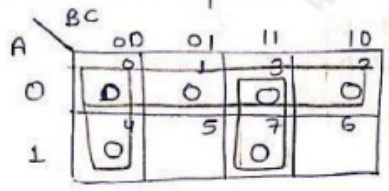
$A\bar{B}\bar{C}$ $\bar{A}BC$
$\bar{A}B\bar{C}$
$\bar{A}B$

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





* Reduce the expression $f = \pi M(0,1,2,3,4,7)$ using AOI & NOR gate.



$AB\bar{C} + A\bar{B} + \bar{B}\bar{C}$

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

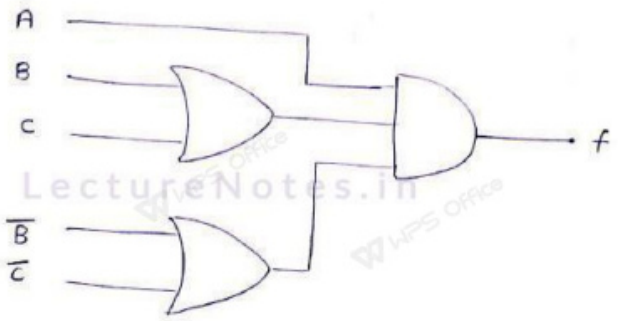
$\overline{A\bar{B}C + \bar{A}BC + AB\bar{C}}$

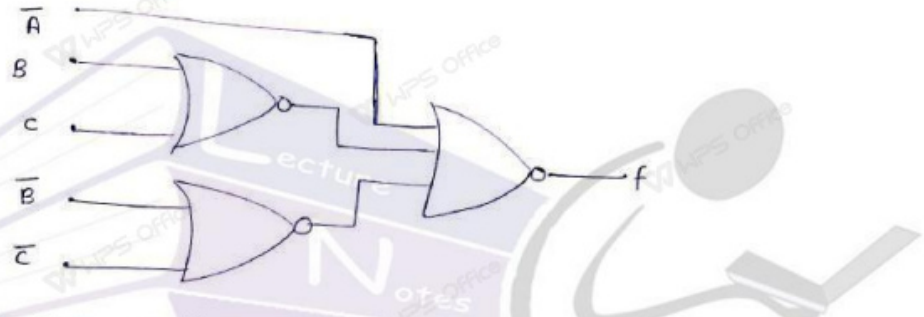
$\bar{A} + (\bar{B}+C) + (B+\bar{C})$

(minimization) $f_{min} = A(B+C)(\bar{B}+\bar{C})$

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

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





* Reduce using mapping and implument in universial logic.

$f = \pi M\ (2, 8, 9, 10, 11, 12, 14)$

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

$A\overline{B}\overline{C}\overline{D}$, $A\overline{B}\overline{C}\overline{D}$ → $A\overline{C}\overline{D}$

$A\overline{B}\overline{C}D$, $A\overline{B}CD$, $A\overline{B}C\overline{D}$ → $A\overline{B}$

$ABC\overline{D}$, $A\overline{B}C\overline{D}$ → $AC\overline{D}$

~~$\overline{A}\overline{B}C\overline{D}$~~ $ABCD$

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

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

POS $fmin = (\overline{A}+D)(\overline{A}+B)(B+\overline{C}+D)$

SOP

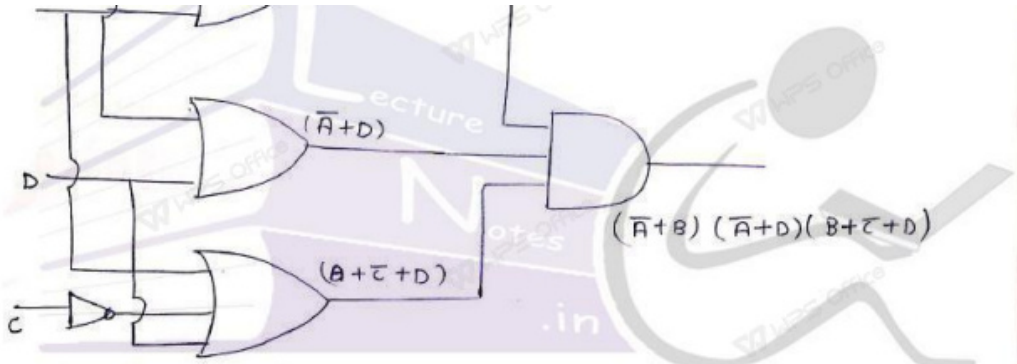
$f = \Sigma m\ (0, 1, 3, 4, 5, 6, 7, 13, 15)$

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

for 1's or 0's

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

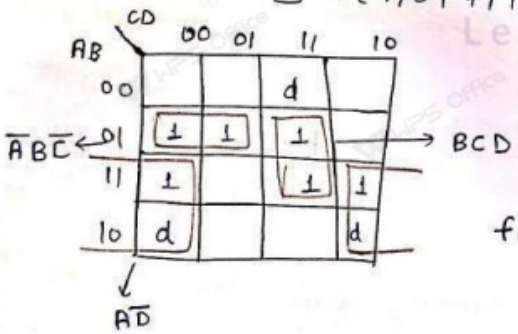
SOP $fmin = \overline{A}\overline{C} + \overline{A}D + \overline{A}B + BD$



14|8|18

Don't care condition- The combination for which the value of the expression are not specified are called don't care condition.

* $F(A,B,C,D) = \sum m(4,5,7,12,14,15) + \sum d(3,8,10)$



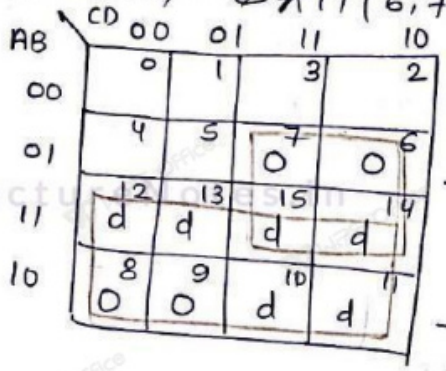
$fmin = A\bar{D} + \bar{A}B\bar{B}\bar{C} + BCD$

(n

* obtain the simplified expression in POS form -

$F(A,B,C,D) = \sum m(0,1,2,3,4,5) + d(10,11,12,13,14,15)$

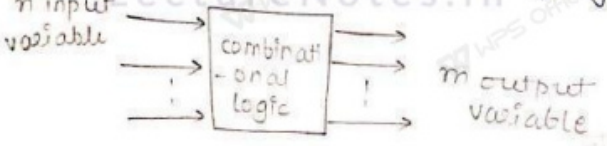
$F(A,B,C,D) = \prod M(6,7,8,9) \cdot d(10,11,12,13,14,15)$



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

Digital Circuit
- combinational circuit
- sequential circuit

combinational circuits - It performs specific processing operation fully specified logically by a set of Boolean functions.
It consists input variable, logic gates and output variable.



example- Adder, Substractor, encoder, decoder, multiplexer, demultiplexer.

Its output depends only upon present input.

characteristic-

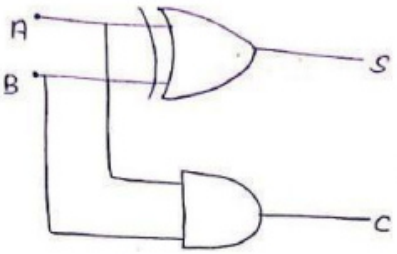
1) fast speed
2) easy to design
3) uses only logic gate
4) do not have ~~flop~~ clock and no memory.

Sequential circuit - its output depends only upon present and past input.

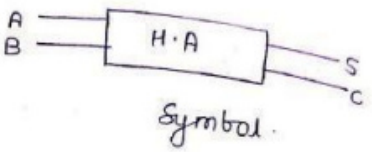
Characteristick -

1) slow speed
2) difficult to design
3) uses flip flops.
4) it needs triggering and have memory elements.

(iii) Half Adder -



logic circuit



Symbol.

Truth table -

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

It is used to add two bits it has two

addend) and two binary outputs (sum, carry).

$S = A \oplus B$

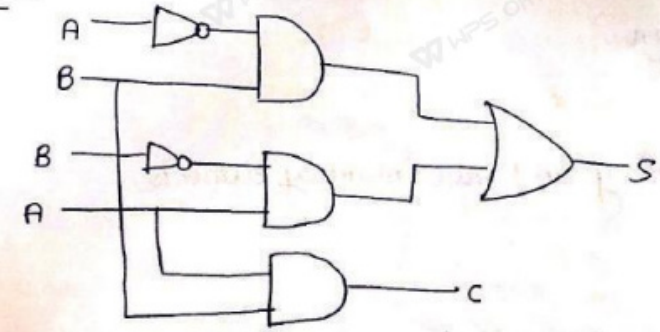
$= A\bar{B} + \bar{A}B$

$C = A \cdot B$

A \ B	0	1
0	0	1
1	1	0

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

AoI –



NAND Logic –

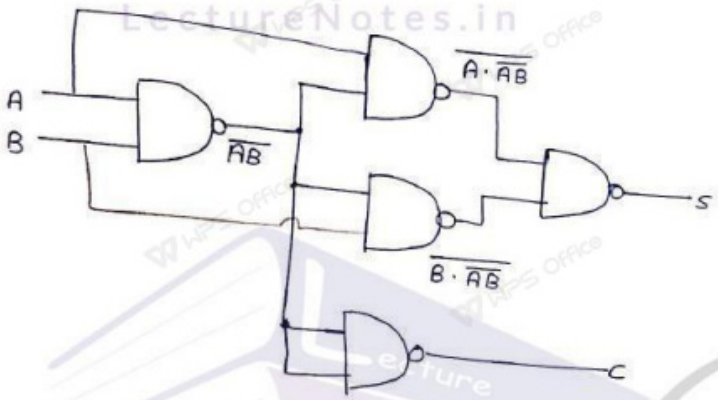
$S = A\bar{B} + \bar{A}B = A\bar{B} + A\bar{A} + \bar{A}B + B\bar{B}$

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

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

$= \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$

$C = AB = \overline{\overline{AB}}$

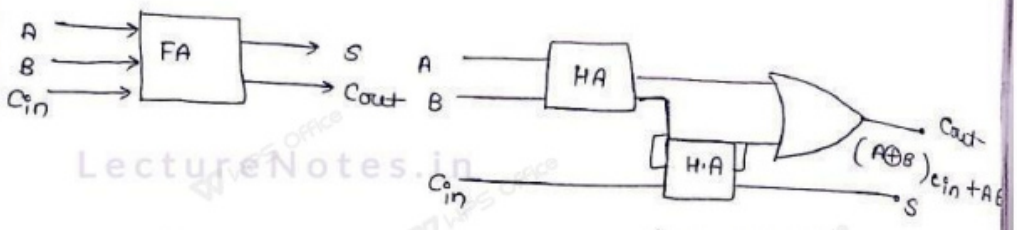


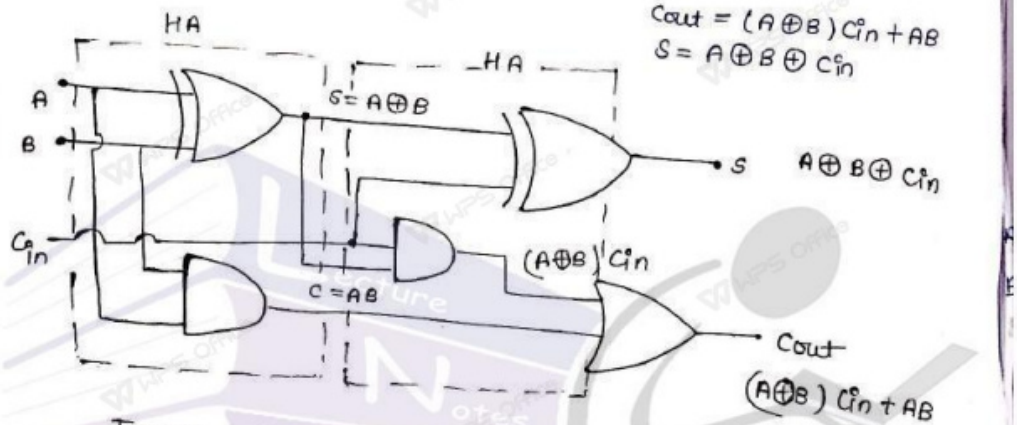
16/8/18

full adder -

- It can add 3 bit at a time.

The third bit is the carry from a lower column. It requires 3 input & has 2 outputs.



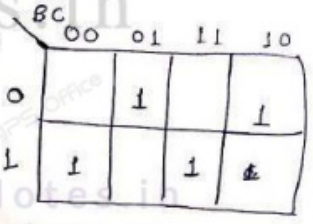


Cout = (A⊕B)Cin + AB
S = A⊕B⊕Cin

Truth table -

A	B	Cin	S	Cout

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



K-map for Sum term

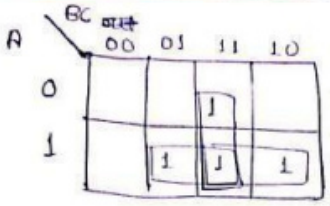
$S = A\bar{B}\bar{C_{in}} + \bar{A}\bar{B}C_{in} + ABC_{in} + \bar{A}B\bar{C_{in}}$

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

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

$= A \oplus B \oplus C_{in}$

Scanned with CamScanner



K map for C_{out}

$C_{out} = AC_{in} + BC_{in} + AB$

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

$= AB C_{in} + A\bar{B}C_{in} + ABC_{in} + \bar{A}BC_{in} + AB$

$= A\bar{B}C_{in} + \bar{A}BC_{in} + AB(C_{in} + 1)$

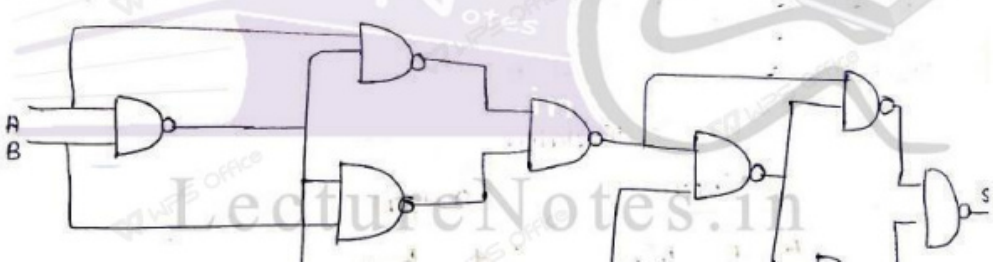
$C_{out} = (A \oplus B)C_{in} + AB$

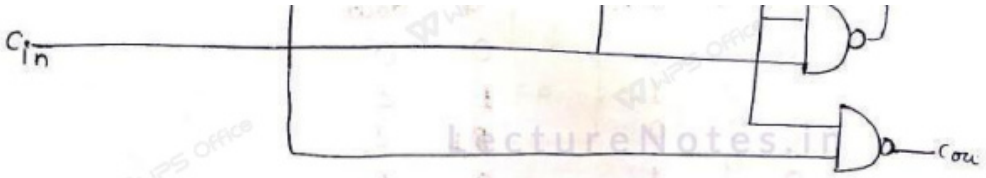
full adder using NAND –

$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$

$S = A \oplus B \oplus C_{in} = \overline{\overline{A \oplus B \cdot \overline{(A \oplus B)C_{in}}} \cdot \overline{A\bar{B}C_{in} \cdot \overline{(A \oplus B)C_{in}}}}$

$C_{out} = C_{in}(A \oplus B) + AB = \overline{\overline{C_{in}(A \oplus B)} \cdot \overline{AB}}$

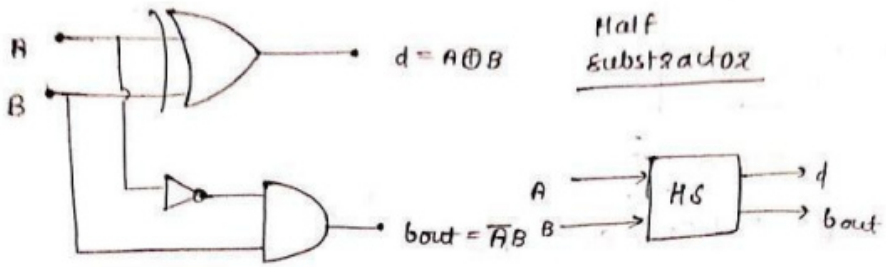




logic diagram of FA using NAND gate.

Subtractor –

- It is a combinational circuit used to subtract two bits. It has two inputs X (minuend) & Y (subtrahend) and two output D (difference) & Bout (Borrow out)



Truth table

A	B	d	bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

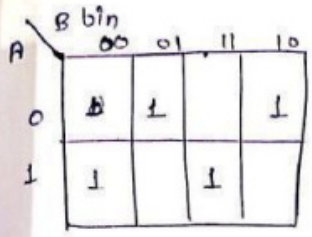
full - Substractor –

- It performs substraction using three bits (minuend, substrahend & borrow)

Truth table

Input			Output	
A	B	bin	d	bout
0	0	0	0	0

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



K map for diffuence

$d = A\bar{B}\bar{b_{in}} + \bar{A}\bar{B}\,b_{in} + AB\,b_{in} + \bar{A}B\,b_{in}$

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

$= b_{in}(\overline{A \oplus B}) + \overline{b_{in}}(A \oplus B)$

$d = A \oplus B \oplus b_{in}$

K map for bout

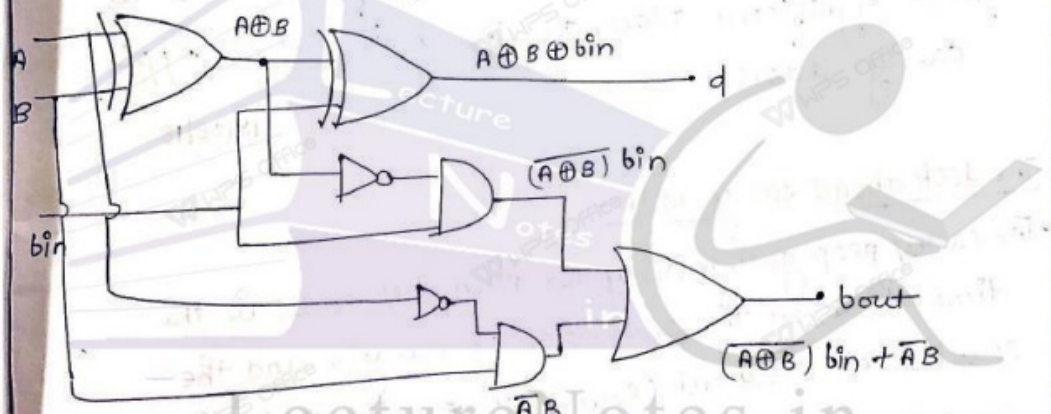
$b_{out} = \bar{A}\,b_{in} + \bar{A}B + B\,b_{in}$

$= \bar{A}\,b_{in}(B + \bar{B}) + \bar{A}B(b_{in} + \bar{b_{in}}) + (A + \bar{A})\,b_{in}$

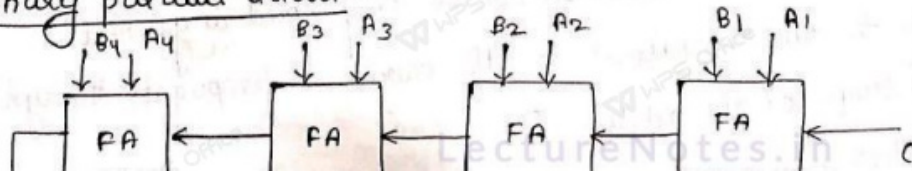
$= \bar{A}B\,b_{in} + \bar{A}\bar{B}\,b_{in} + AB\,b_{in} + \bar{A}B\,b_{in} + \bar{A}B$

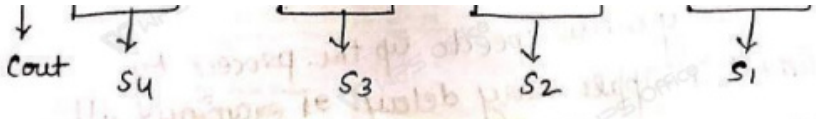
$= b_{in}(\bar{A}\bar{B} + AB) + \bar{A}B\,b_{in} + \bar{A}B$

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



Binary parallel adder -





Logic diagram of 4-bit binary parallel adder.

→ Let 4 bit words to be added represented by

$A_3\ A_2\ A_1\ A_0 = 1\ 1\ 1\ 1$

$B_3\ B_2\ B_1\ B_0 = 0\ 0\ 1\ 1$

Significant 4 3 2 1

I/P carry = 1 1 1 0

Augend = 1 1 1 1

Addend = 0 0 1 1

1 0 0 1 0

output carry

It add 2 binary no in II form and produces the Arithmetic sum of those no. in II form

It consists of full adders connected in a chain with output carry from each full adder connected to the input carry of the next full adder.

The parallel adder in which the carry out of each full adder is the carry in to the next most significant adder as full most significant adder as shown is called ripple carry adder.

18/08/18

The look ahead carry adder-

- The carry propogation delay for each full adder is the time between the application of carry in and the occurence of carry out (C_{out}). In parallel adder the speed in which an addition can be performed is govned by the time required for the carries to propogate through all stages of the adder.
- The look ahead carry adder speeds up the process by eliminating the ripple carry delay. It examines all

the inputs simultaneously and also generates the carry in bits for all stages simultaneously.

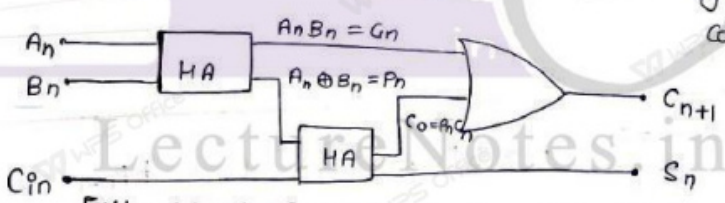
- The method of speeding up the addition process is based on two additional functions of the full adder called carry generate and carry propagate.

Truth table of full adder emphasising on carry generation

Row	A	B	Cin	S	Cout	
0	0	0	0	0	0	No carry generate i.e Cout = 0
1	0	0	1	1	0	
2	0	1	0	1	0	carry propogate i.e Cout = Cin
3	0	1	1	0	1	
4	1	0	0	1	0	
5	1	0	1	0	1	
6	1	1	0	0	1	carry generate i.e Cout = 1
7	1	1	1	1	1	

1) for row 0 and 1 Cout is 0 and independent of Cin.
2) In row 6 and 7 Cout is 1 and independent of Cin.
3) In row 2, 3, 4, 5 the Cout = Cin. There carry propagate combination.

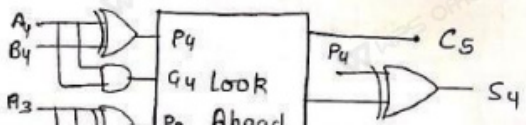
carry generate combination.

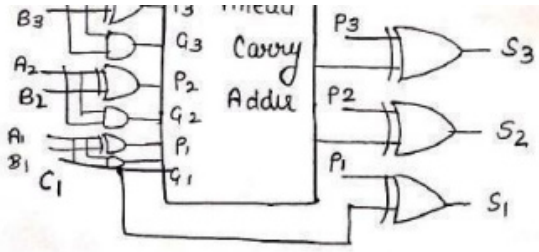


Full adder (nth stage of a parallel adder)

$C_{n+1} = C_{out_n} = (A_n \oplus B_n) C_n + A_n B_n$

$S_n = A_n \oplus B_n \oplus C_n$





Consider one F.A. stage as shown in figure if both bits A_n and B_n are 1 an a carry has to be generated regardless whether C_{in} is 0 or 1 this is called generated carry and expressed as $G_n = A_n B_n$

The Boolean exp. for sum and carry output of nth stage is given by

$$S_n = P_n \oplus C_n$$

$$C_{o_n} = C_{n+1} = G_n + P_n C_n \quad \text{where } G_n = A_n . B_n$$

→ The expression for the carry out of full adder is as follows:

for $n = 0$ $\quad C_1 = G_0 + P_0 C_0$

for $n = 1$ $\quad C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$

$$= G_1 + P_1 G_0 + P_1 P_0 G_0$$

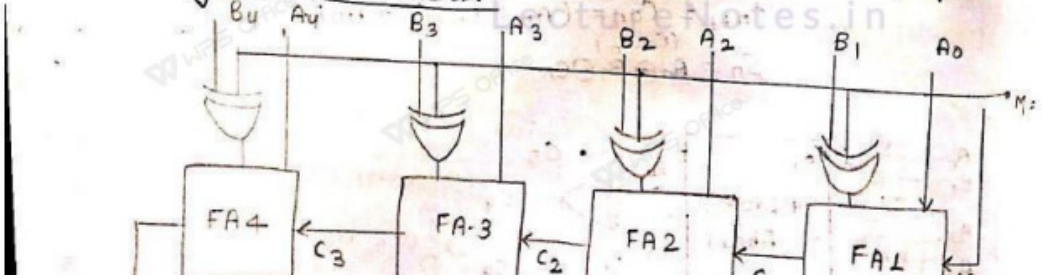
for $n = 2$, $\quad C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 G_0 P_1 + P_2 P_1 P_0 G_0$

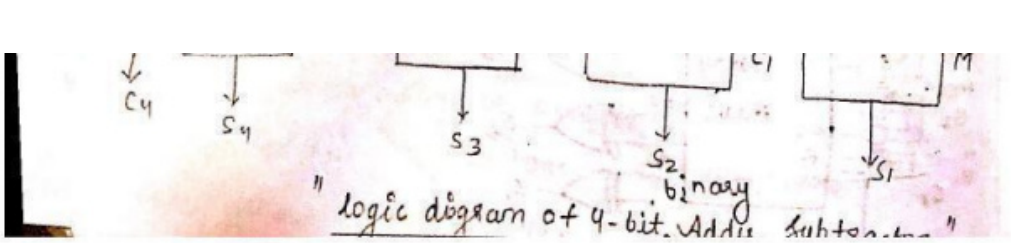
In general -

$$C_n = G_{n-1} + P_{n-1} G_{n-2} + P_{n-2} P_{n-1} G_{n-3} + \cdots + P_{n-1} \cdots P_0 C_0$$

21/08/18

Binary Adder Substractor -



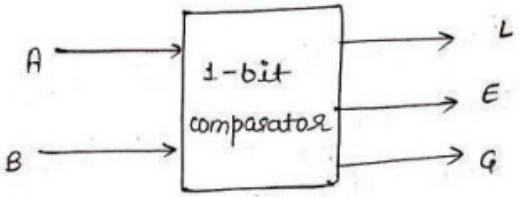


"logic diagram of 4-bit Addr Subtractor"

→ The mode input M controls the operations when $M=0$, the circuit is an adder & when $M=1$, the circuit is a subtractor.

→ Here addition & substraction operations combined into one circuit with one common binary adder. This is done by including XOR gate in each full adder. Each x-OR gate receives input M and input B when M is 0 ~~then~~ we have $B \oplus 0 = B$, the full adder receives the value of B, & input carry is zero and performs $A+B$ when M is 1 ~~then~~ we have $B \oplus 1 = \overline{B}$ and $C_1 = 1$ (carry), the B inputs are complimented and 1 is added through input carry which is 1, through the input carry the circuit performs the operation A plus 2's compliment

Comparators –



It is a logic circuit used to compare the magnitude of two binary no. depends on the design it may provide the output i.e high (active).

Note –
[X-NOR is a basic comparator because its output is 1 if its two input bits are equal.]

→ Implementation of the logic, for a 4-bit binary numbers $A_3 A_2 A_1 A_0$ & $B_3 B_2 B_1 B_0$

EQUALITY = $(A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$

For a 1 bit magnitude comparator –

let 1-bit number be $A = A_0$ & $B = B_0$

If $A_0 = 1$ & $B_0 = 0$ then $A > B$

therefore, $A > B$: $G = \overline{A_0} B_0$

Truth table

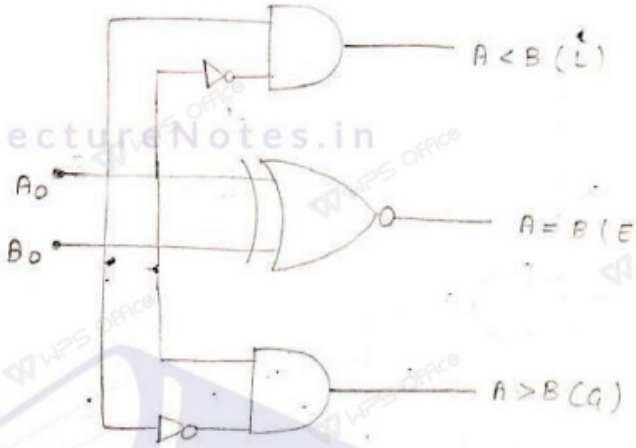
A	B	L	E	G
0	0	0	1	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0

1) If $A_0 = 0$ & $B_0 = 1$ then $A < B$

so $A < B$: $L = \overline{A_0} B_0$

2) If $A_0 = B_0 = 0$ & $A_1 = B_1 = 1$ then

$A = B$: $E = A_0 \odot B_0$



2-bit comparator-

Let 2-bit numbers be $A = A_1 A_0$

$B = B_1 B_0$

0 1	1	A > B
0 0	0	

1) If $A_1 = 1$ & $B_1 = 0$, then $A > B$

2) If A_1 & B_1 coincide and $A_0 = 1$ & $B_0 = 0$ then $A > B$

so logic expression for $A > B$

$A > B$: $G = A_1 \overline{B_1} + (A_1 \odot B_1) A_0 \overline{B_0}$

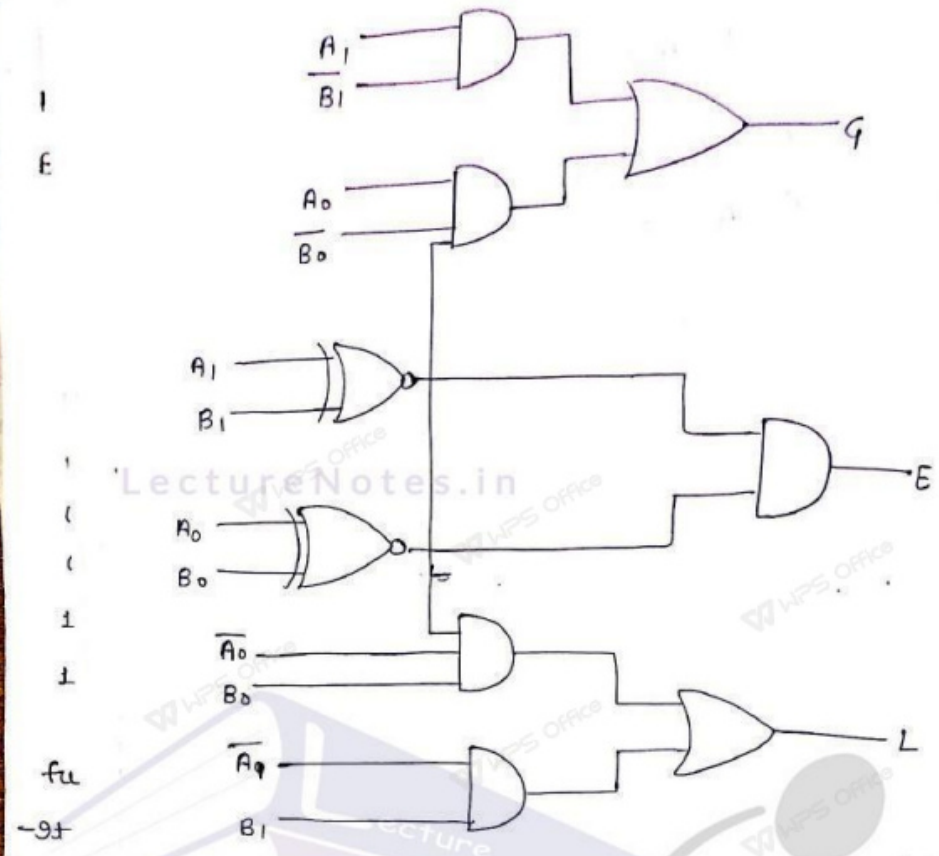
3) If $A_1 = 0$ & $B_1 = 1$ ~~there~~ then $A < B$ for

If A_1 & B_1 coincide & $A_0 = 0$ & $B_0 = 1$, then $A < B$

then $A < B$: $L = \overline{A_1} B_1 + (A_1 \odot B_1)(\overline{A_0} B_0)$

4) If A_1 & B_1 coincide and A_0 & B_0 coincide then $A = B$, then.

$A = B$: $E = (A_1 \odot B_1)(A_0 \odot B_0)$



Logic diagram of 2-bit comparator

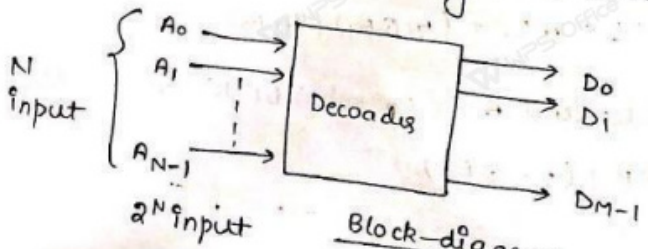
23/08/18

Decoadis –

Encoadis – encrypted & transmitted

Decoadis – decrypted & displayed

Q: It is an logic circuit that converts N bit binary input code into M output lines such that only one output line is activated for each one of the possible combination of inputs thus we can say it detects a particular code.



2^N input code

Block diagram

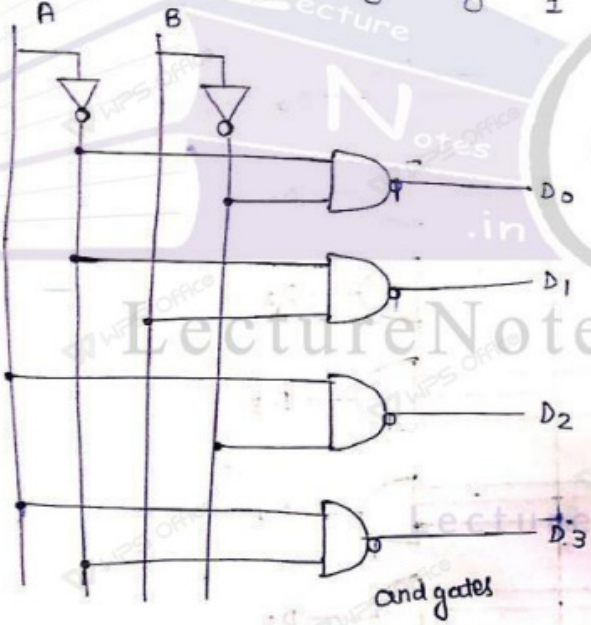
for each input combination, M outputs will be active (High) and ~~on~~ other outputs will be ~~high~~ low.

2 to 4 line
3-to-8 line
4-to-16 line

2-to-4 line decoder – 2 x 4 or 2:4

Truth table

Inputs		Outputs			
A	B	D_0	D_1	D_2	D_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



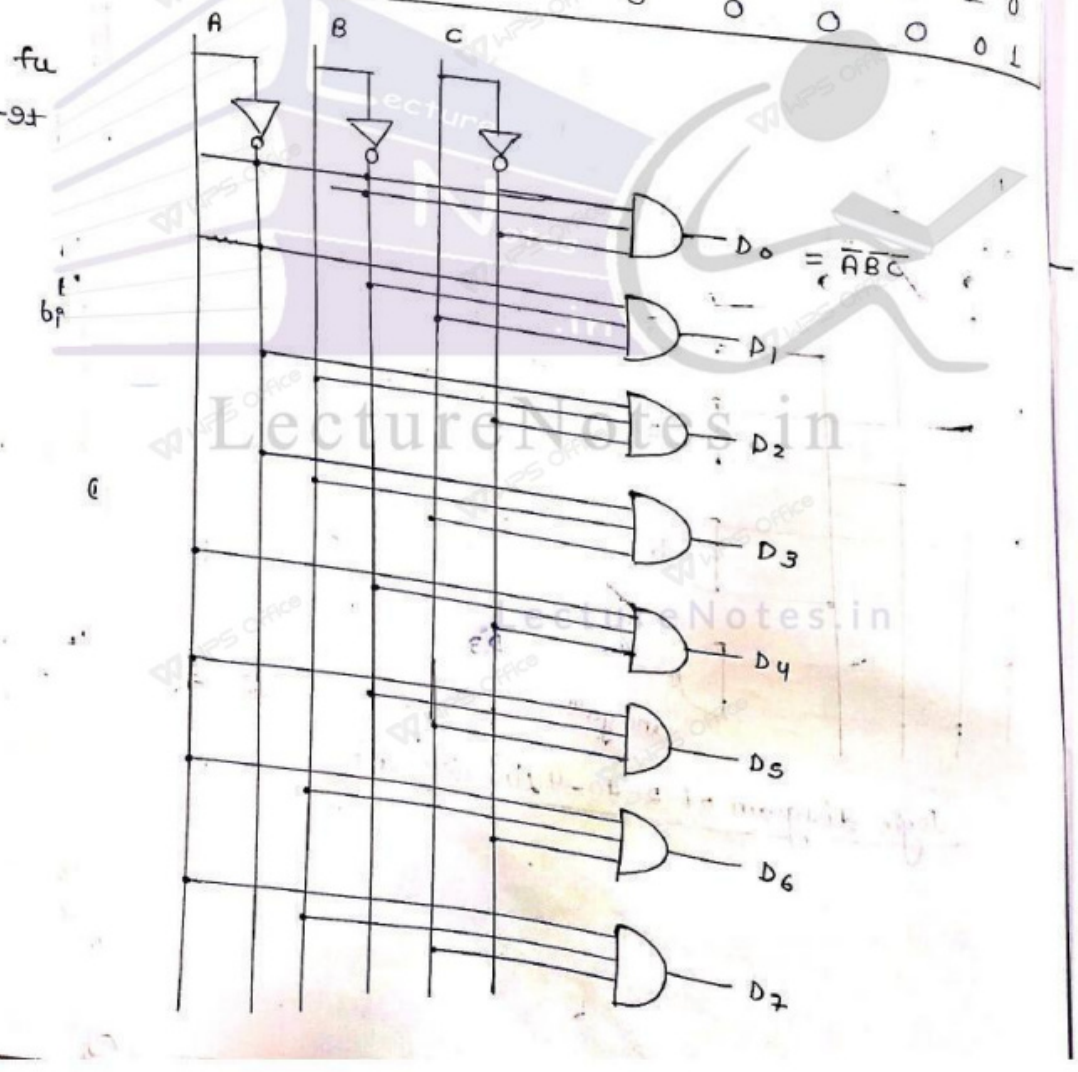
Logic diagram of 2-to-4 line decoder

3-to-8 line decoder –

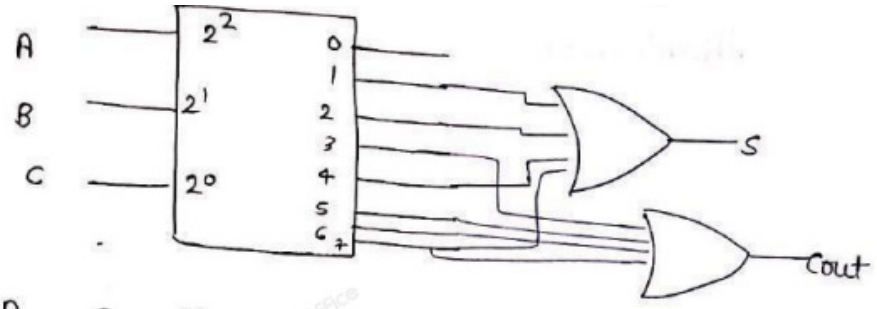
It has 3 input and 8 output.

It is also called binary to octal decoader.

Input			Output							
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



Combinational logic implementation-



A	B	Cin	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \sum m(1,2,4,7)$

$Cout = \sum m(3,5,6,7)$

$S = \bar{A}\bar{B}Cin + \bar{A}B\bar{Cin} + A\bar{B}\bar{Cin} + ABCin$

$= \sum m(1,2,4,7)$

$Cout = \bar{A}BCin + A\bar{B}Cin + AB\bar{Cin} + ABCin$

$= \sum m(3,5,6,7)$

Till decoder it will be coming in mid sem.

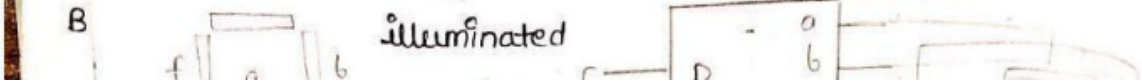
logic diagram of full adder using 3-to-8 line decoder-

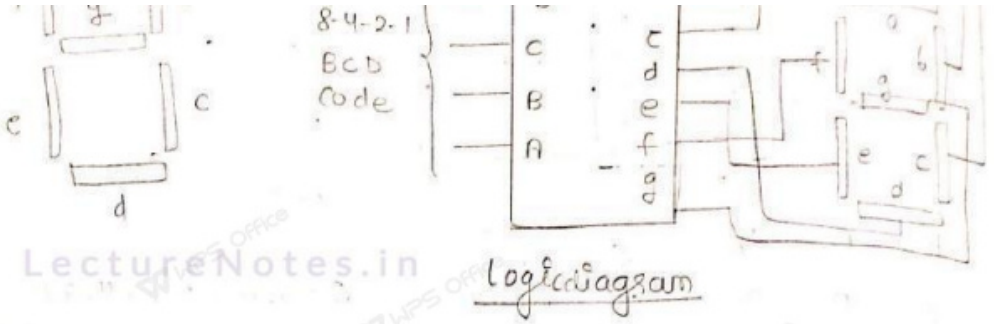
25/08/18

BCD-to-Seven Segment decoder-

B

illuminated





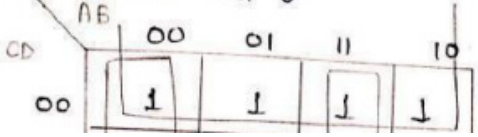
logic diagram

A seven segment display is normally used to display any one of the decimal digit 0 to 9. Sometimes hex character a through f may be produce. Each segment is made up of a material that emits light when ~~ek~~ current is passed through it. Commonly used materials led's, ~~incln~~ incandecent filaments.

Decimal digit	BCD inputs A B C D	Seven-segment output a b c d e f g
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	0 1 1 0 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	1 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 1 0 1 1

→ The K map is used to simplify the logic expression.

example for b



~~for anp~~

	00	01	11	10
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

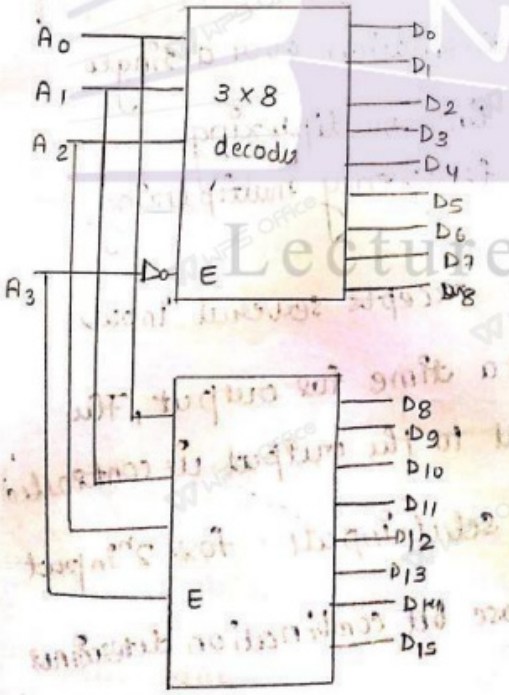
for empty boxes we don't care condition.

$b = \overline{D} + \overline{A}\,\overline{B} + AB$

$= \overline{D} + A \odot B$

$b = \sum m\,(0,1,2,3,4,7,8,9) + \sum d\,(10,11,12,13,14,15)$

4-to-16 decoder from two 3-to-8 decoder-



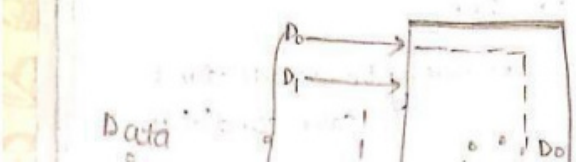
* When E=0 upper decoder will be activated and D_0 — D_7
* When E=1, 1 ~~outer~~ lower decoder will be activated D_8 – D_{15}

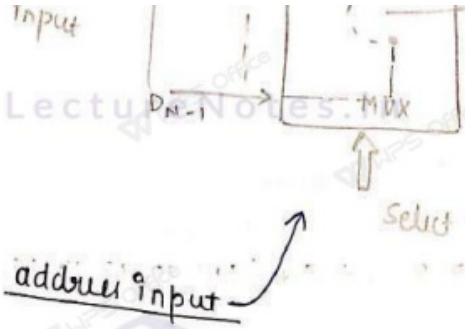
Binary input				output
A_3	A_2	A_1	A_0	
0	0	0	0	D_0
0	1	1	1	D_7
1	0	0	0	D_8
1	1	1	1	D_{15}

28/8/18

Multiplexers (Data Selector)

MUX



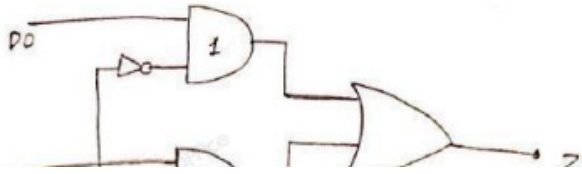


The term multiplexer means many into one. It is a process of transmitting a large no. of information over a single line. It is of two type- time multiplexing
frequency multiplexing

a MUX is a logic circuit that excepts several inputs and allows one of them ata time for output, The routing of desired data input to the output is controlled by select outputs also called select inputs. for 2^n input lines and n select line whose bit combination determines which input is selected.

Basic 2-input Multiplexer- It has two data inputs D_0 and D_1 and one select input's' and one output line.

Select	output
S	Z
0	D_0
1	D_1



Output-

$Z = \bar{S} D_0 +$

, when $S=0$, AND gate 1 is enable so gate 2 is disable.
so $Z = D_0$

, when $S=1$, gate 1 is disable, gate 2 is enable.
$Z = D_1$

procedure of

single

using

inputs

, the

controlled

2^n input

determines

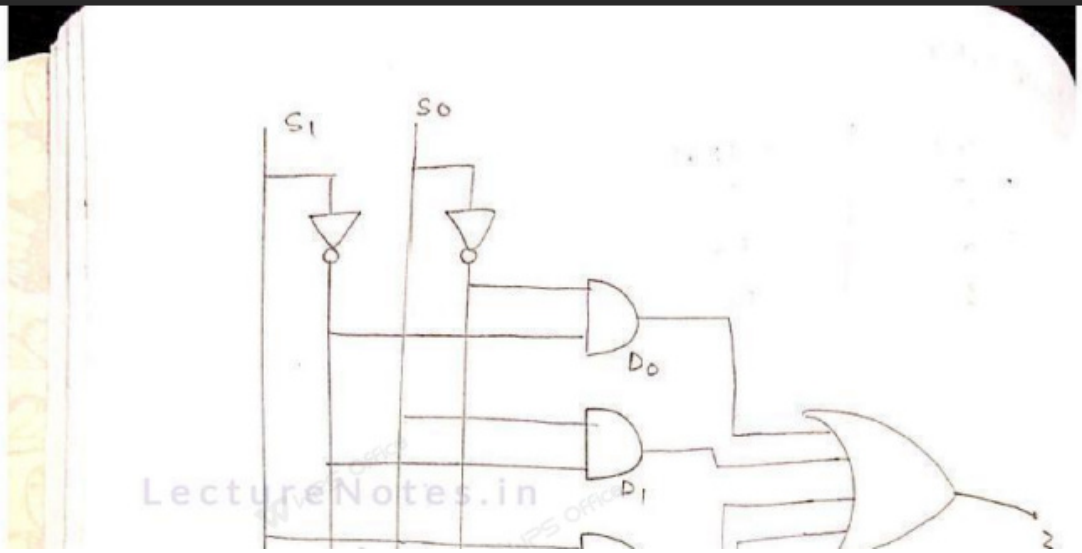
puts

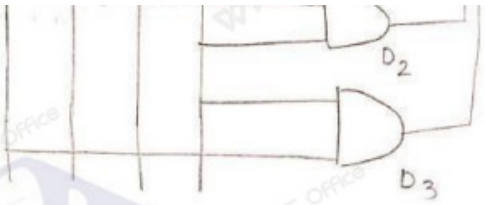
put

4:1 MUX –

truth table –

Select input		output
S_1	S_0	Z
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3





logic diagram of 4:1 MUX

The expression for the data output –

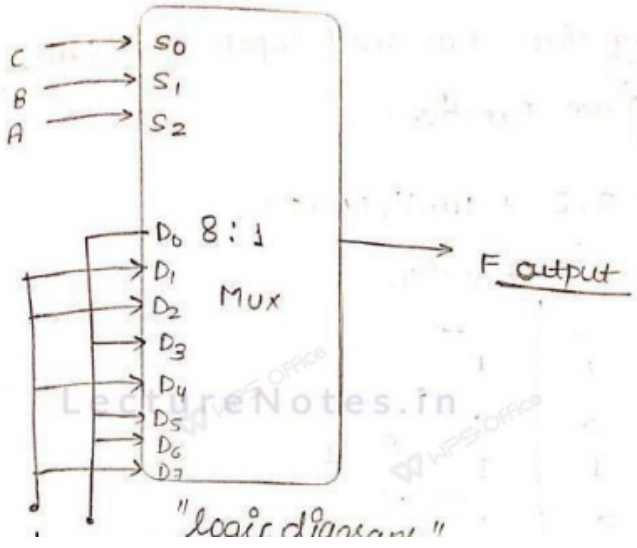
$$Z = D_0 \overline{S_0} S_1 + D_1 S_0 \overline{S_1} + D_2 \overline{S_0} S_1 + D_3 S_0 S_1$$

Application of multiplexer –

* use of MUX to implement the logic function $F = A \oplus B \oplus C$

8:1 –

S_2	S_1	S_0	$F = A \oplus B \oplus C$
A	B	C	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



"logic diagram"

1 0

since there are 3 input variable we can used MUX with three data select input that is 8:1 MUX. Truth table shows the use of data select inputs S_0, S_1, S_2 for 3 input variables A, B, C. We connect logic one to D_1, D_2, D_4, D_7 and logic 0 to D_0, D_3, D_5, D_6. Then MUX behaves exactly the same way that a set of logic gates implementing the function f.

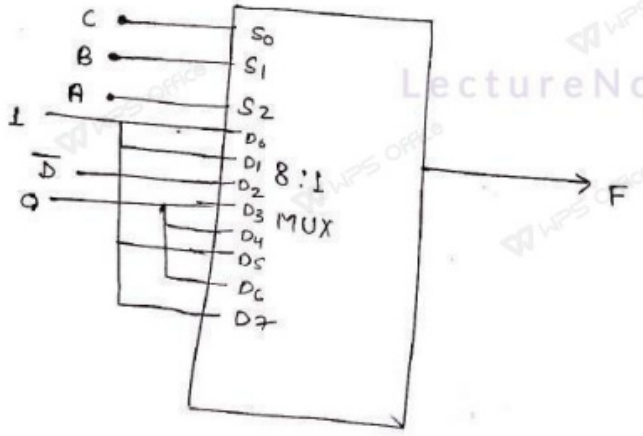
Date - 12/09/19

* Use a MUX having three data select inputs to implement the logic for the given function.

$F = \sum m\,(0,1,2,3,4,10,11,14,15)$

	Data				Function	
	S_2	S_1	S_0			
	A	B	C	D	F	
D_0	0	0	0	0	1	$F = 1$
	0	0	0	1	1	
D_1	0	0	1	0	1	$F = 1$
	0	0	1	1	1	
D_2	0	1	0	0	1	

	0	0	0	1	$F = \bar{D}$
	0	1	0	0	
D_3	0	1	1	0	$F = 0$
	1	0	1	0	
D_4	1	0	0	0	$F = 0$
	1	0	0	1	
D_5	1	0	1	0	$F = 1$
	1	0	1	1	
D_6	1	1	0	0	$F = 0$
	1	1	0	1	
D_7	1	1	1	0	$F = 1$
	1	1	1	1	

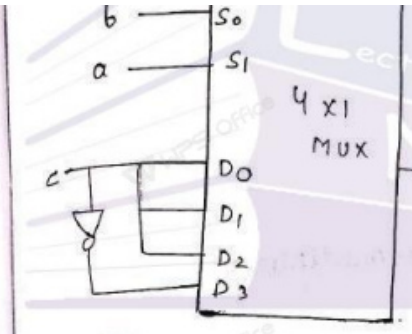


logic diagram

* Implement the following function with a MUX.

$F(a,b,c) = \sum m(1,3,5,6)$, choose a & b as select inputs.

S_1	S_0		
a	b	c	F
0	0	0	0
0	0	1	1
			$F = c$
0	1	0	0
0	1	1	1
			$F = c$
1	0	0	0
1	0	1	1
			$F = c$
1	1	0	1
1	1	1	0
			$F = c$



* Implement the following Boolean function using 8:1 MUX considering 8:1 MUX considering D as the input & A, B, C as the select lines.

$$F(A,B,C,D) = A\bar{B} + BD + \bar{B}C\bar{D}$$

$$= 10xx + x1x1 + x010$$

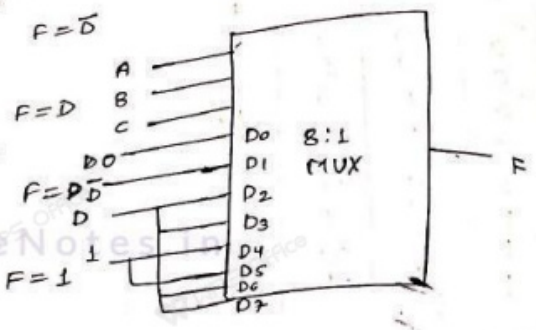
$$= 1000 + 1010 + 1001 + 1011 + 0101 + 1101 + 0111 + 1111 + 0010 + 1010$$

$$= \Sigma m(8,9,10,11,5,7,13,15,2,10)$$

$$= \Sigma m(2,5,7,8,9,10,11,13,15)$$

	S_2 S_1 S_0 A B C D	F
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	1
3	0 0 1 1	0
4	0 1 0 0	0
5	0 1 0 1	1
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	1
9	1 0 0 1	1
10	1 0 1 0	1
11	1 0 1 1	1
12	1 1 0 0	0
13	1 1 0 1	1
14	1 1 1 0	0
15	1 1 1 1	1

F = 0
F = $\bar{D}$
F = D
F = $D\bar{D}$
F = 1
F = 1
F = D
F = D



when CD = 00, Do
F2, D8 will app
F4

when CD = 01, D1
F4,

when CD = 10, D2

when

when CD = 11, D3
respectively

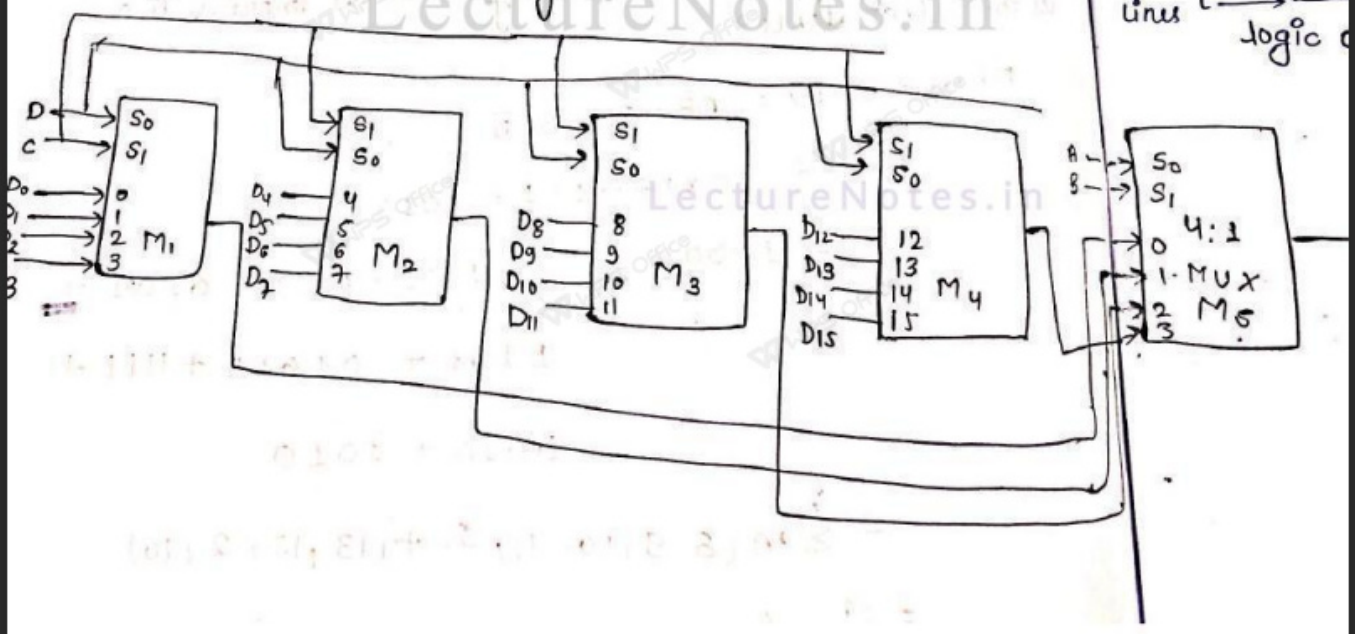
depending upo
will appear

* Design of a
* Design of a

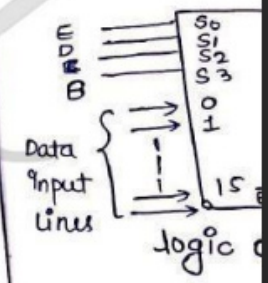
Course- Multiplexer, multiplexer tree, de-multiplexer.

Multiplexer tree-

* Design of a 16:1 MUX using 4:1 MUX.



2:1 MUX.



when CD = 00, D_0 will appear at F_1, D_4 will appear at F_2, D_8 will appear at F_3 and D_{12} will appear at F_4

when CD = 01, D_1, D_5, D_9, D_{13} appears at F_1, F_2, F_3, F_4,

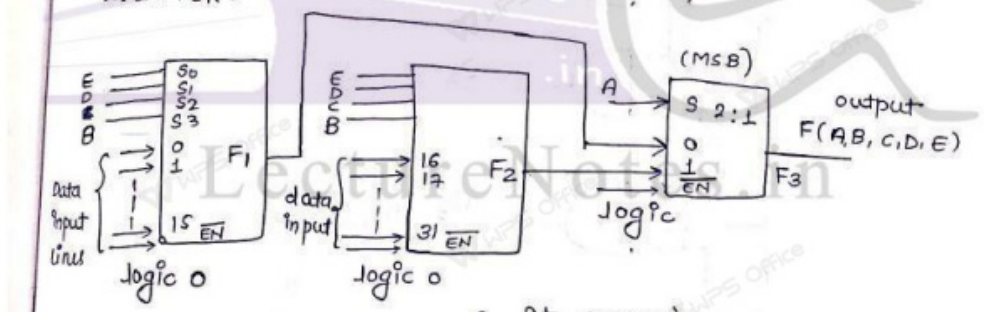
when CD = 10, D_2, D_6, D_{10}, D_{14} appears at F_1, F_2, F_3, F_4.

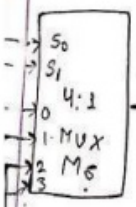
when CD = 11, D_3, D_7, D_{11}, D_{15} will appear at $F_1, F_2, F_3 F_4$ respectively.

depending upon the value of AB, either F_1 F_2, F_3 or F_4 will appear at output.

* Design of a 16:1 MUX using 4:1 MUX.
* Design of a 32:1 MUX using two 16:1

2:1 MUX.





- It requires 5 data select lines, 4 ~~data~~ select line BCDE are connected to 16:1 MUX and MSB A 2:1 MUX.
- for the value of B,C,D,E (0000, – 1111), input 0 to 15 will appear at the input terminal zero of 2:1 MUX

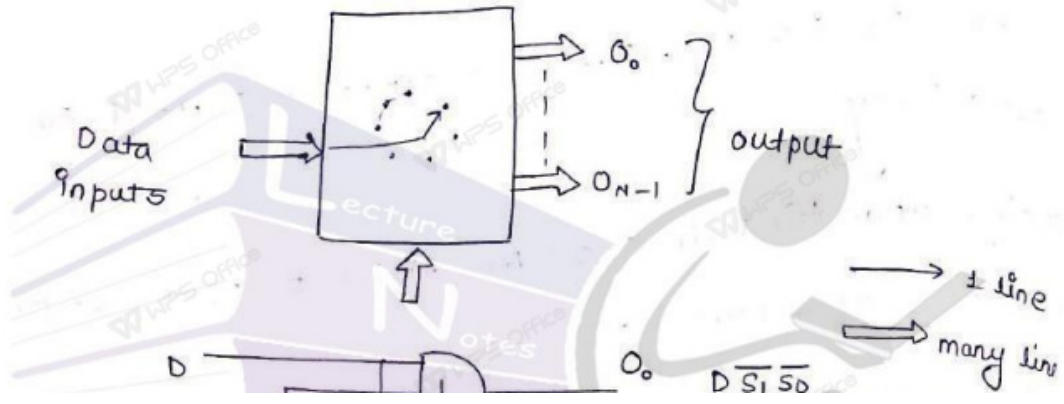
through F_1 and ~~16~~ 16 to 31 will appear at the input terminal of 1 of 2×1 MUX through output F_2

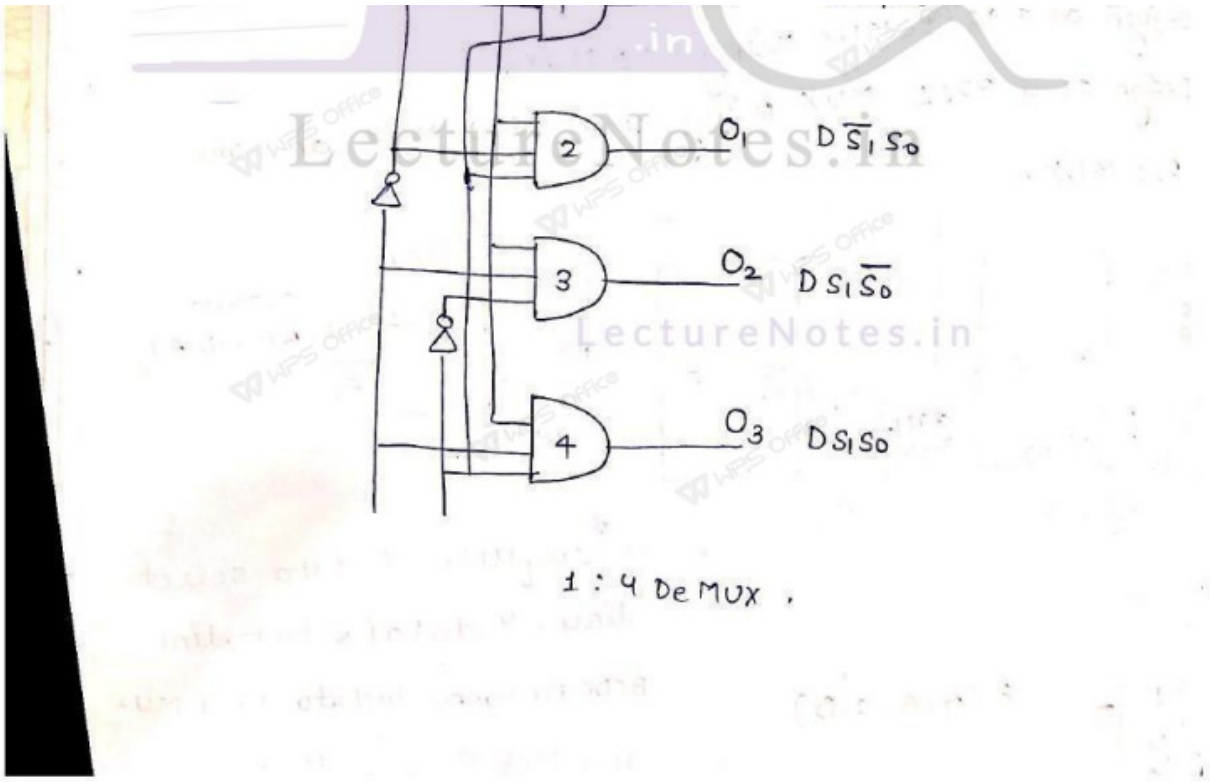
for A = 0, Output F = F_1 and

A = 1, output F = F_2.

De-Multiplexer (Data distributor)

1) 1 line – 4 – line –

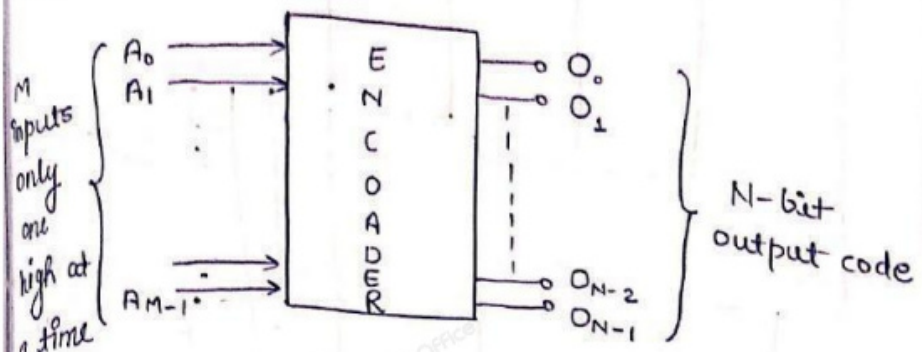




1 : 4 DeMUX .

Encoder–

19|Sep|2018

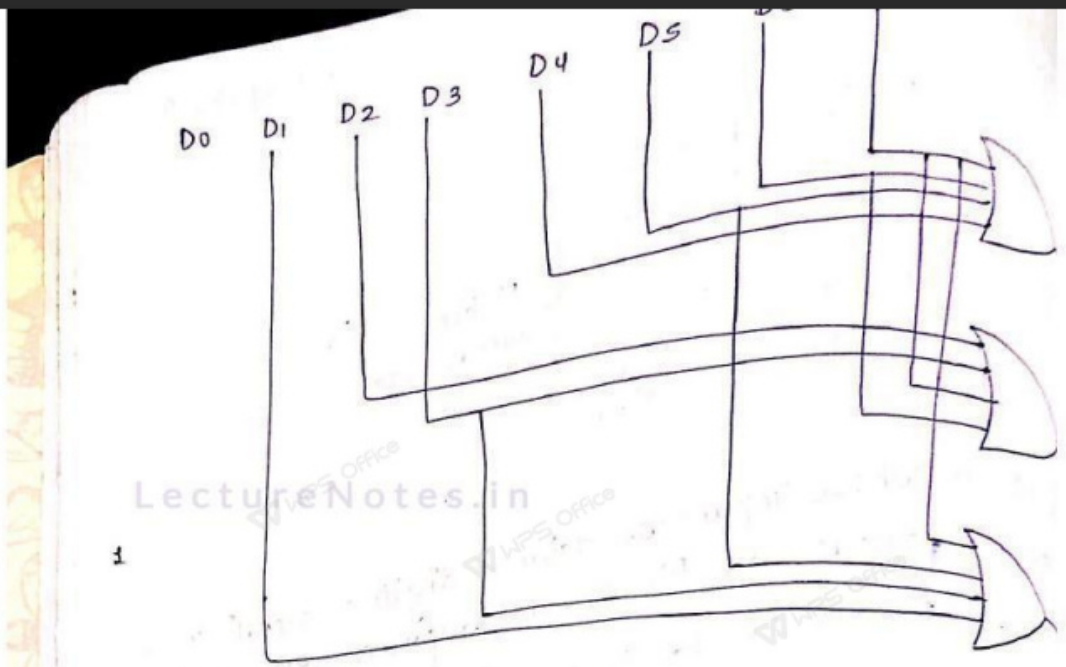


It is a device whose inputs are decimal digits or alphatic characters and outputs are the coaded representation of the input. It converts no. or symbols into coaded format. It performs reverse operations of decoder. It has a no. of input lines and only one of which activated at a given time and produce an N inp bit output code.

1) Octal to Binary Encoder– (8 to 3 lines)

Octal digit	Binary		
	A_2	A_1	A_0
D_0	0	0	0
D_1	0	0	1
D_2	0	1	0
D_3	0	1	1
D_4	1	0	0
D_5	1	0	1
D_6	1	1	0
D_7	1	1	1

truth table



logic diagram

$A_2 = D_4 + D_5 + D_6 + D_7$

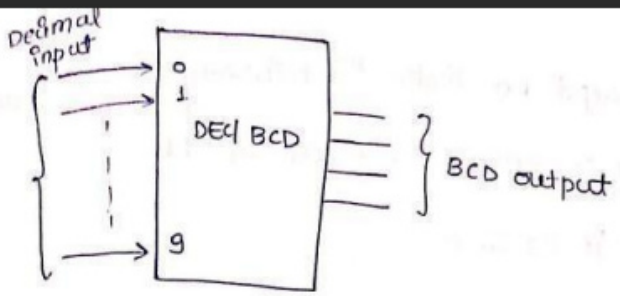
$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7$

D_0 = don't care

2) Decimal to BCD Encoder -

Input		BCD			
		A_3	A_2	A_1	A_0
0	D_0	0	0	0	0
1	D_1	0	0	0	1
2	D_2	0	0	1	0
3	D_3	0	0	1	1
4	D_4	0	1	0	0
5	D_5	0	1	0	1
6	D_6	0	1	1	0
7	D_7	0	1	1	1
8	D_8	1	0	0	0
9	D_9	1	0	0	1



Priority Encoder–

① Encoder operates correctly provided that only one decimal input is HIGH at any given time. ② Practically two or more decimal input may become HIGH at the same time. The priority encoder is a logic circuit that responds to just one input in accordance to some priority system.

③ It is based on decimal input, the largest, is 1 that is which decimal is encoded. eg)- 1234, 4 will be prioritise.

1) 4-input Priority Encoder–

Inputs				Outputs		
D_0	D_1	D_2	D_3	A	B	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1

x	x	1	0		1	0	1
x	x	x	1		1	1	1

from the truth table in addition of output A & B, the circuit has 3rd out put that ie V, which is a valid bit indicator that is set to 1 with 1 or more inputs are equal to one.

If all is 0 ie 1st condition, no valid input, V is zero, A, B is X.

Higher the subscript no. higher the priority, input D_3 highest priority so output for A, B is 1,1.

D_2 is the next if D_3 is 0.

K map for A–

D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

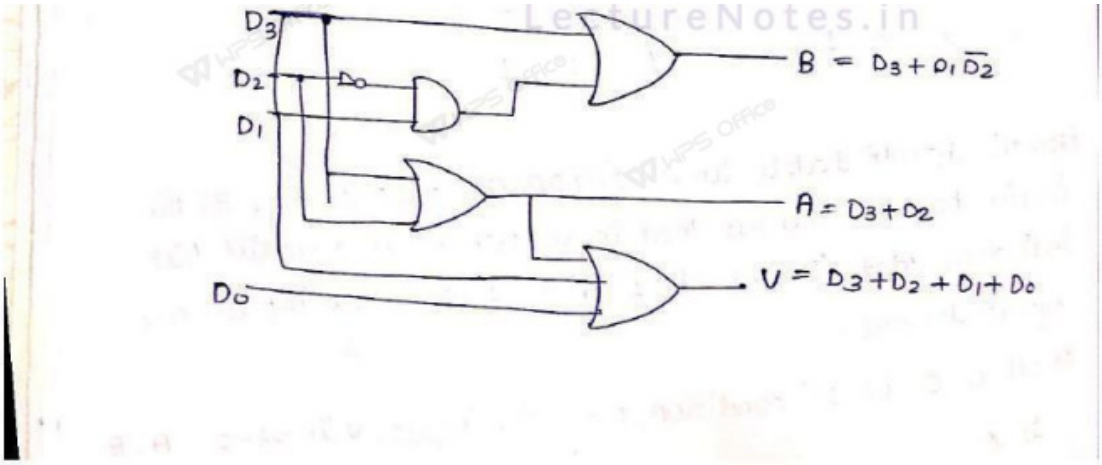
$A = D_2 + D_3$

K map for B–

D_0D_1 \ D_2D_3	00	01	11	10
00		1	1	
01	1	1	1	
11	1	1	1	
10		1	1	

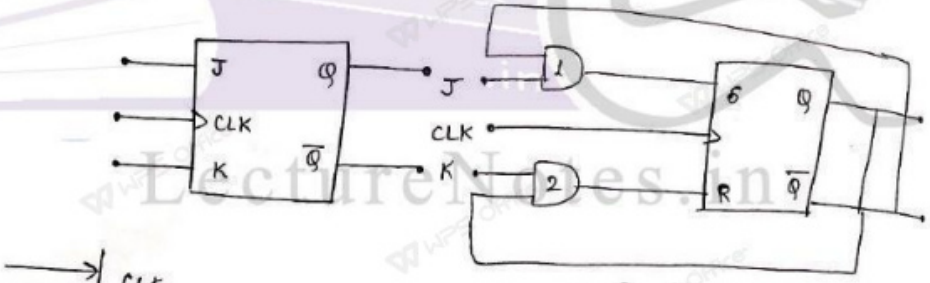
$B = D_3 + D_1\overline{D_2}$

$V = D_3 + D_2 + D_1 + D_0$



PS	Clock	data input		NS	Action
Q_n	CLK	S	R	Q_{n+1}	
0	1	0	0	0	no change
1	1	0	0	1	
0	1	0	1	0	Reset
1	1	0	1	0	
0	1	1	0	1	Set
1	1	1	0	1	
0	1	1	1	X	Forbidden / undetermine
1	1	1	1	X	
0	X	X	X	0	No change.
1	X	X	X	1	

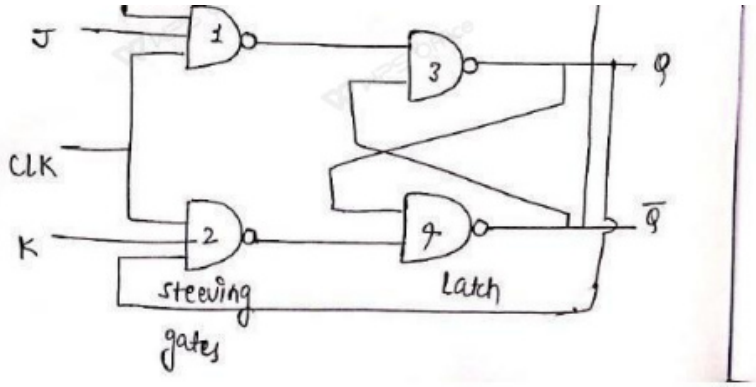
JK Flip Flop–



logic diagram

→| CLK

+ve edge triggered.



CLK	Input J	Input K	output Q_{n+1}	Action
x	0	0	Q_n	No change
1	0	0	Q_n	"
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	$\overline{Q_n}$	Toggle

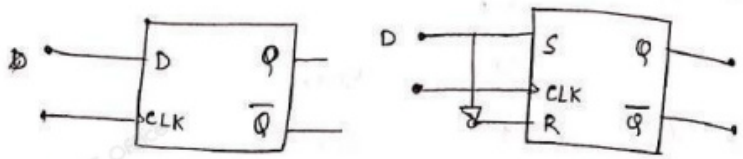
case I- when J = 0, whatever value of Q and $\overline{Q}$, the output of 1st nand gate is zero.

~~when K = 0~~ when K = 0, whatever value of Q and $\overline{Q}$ output of IInd nand gate is zero.

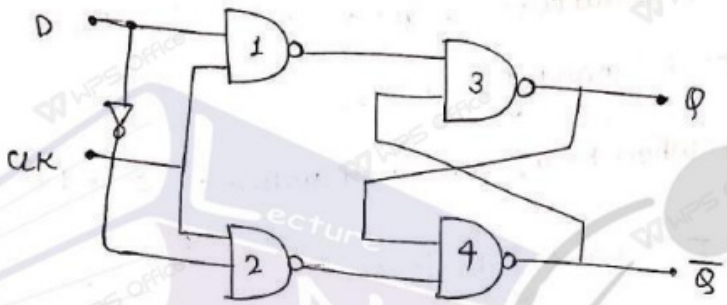
mention all the case.

D Flip Flop–

→ delay (D) flip flop.



Symbol

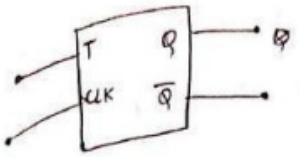


logic diagram using NAND gate

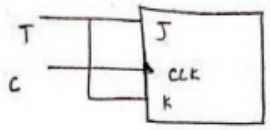
Ps		Input	Output	
Q_n	CLK	D	Q_{n+1}	Action / State
0	1	0	0	
1	1	0	0	Rest
0	1	1	1	Set
1	1	1	1	
0	↑	0	0	
1	X	0	1	NC

T Flip Flop–

It is a single control input labelled as T for Toggle or triggered FF.



Symbol



T Flip Flop using JK FF

Q_n	clk	T	Q_{n+1}	Action
0	↑	0	0	NC
1	↑	0	1	
0	↑	1	0 1	Toggle.
1	↑	1	0	

operating characteristics of flip flop– 26/09/18

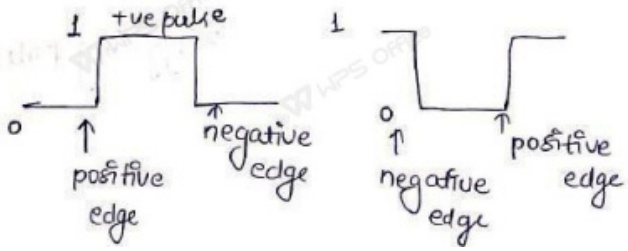
1) Triggering of FFs

1) level triggering < positive level (HIGH)
negative level (LOW)

When clock pulse goes HIGH, the flip flop is said to level triggered flip flop. When the clock is positive then it is said positive level triggered flip flop.

If FF change its states only when clock is negative (LOW) and then it is called -ve level triggered Flip Flop.

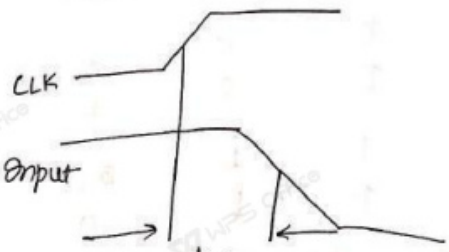
2) Edge triggered



a clock pulse goes through two signal transition that is 0-to-1 and return from 1-to-0 as shown in figure

2) Propagation Delay

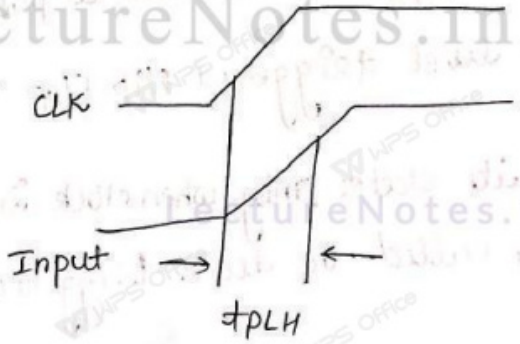
t_{PHL} is measured from the triggering of the pulse high to low transition (high-low) of the output.



triggering pulse high-low

defi- The time interval between time of application of triggering's and time of which the output makes a transition is called a propogation delay time.

Its range is few ns to 1 μs



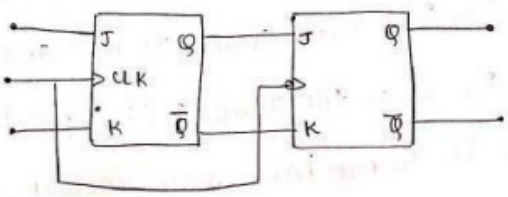
triggering pulse low-high

Race around condition (Clock skew and time race)-

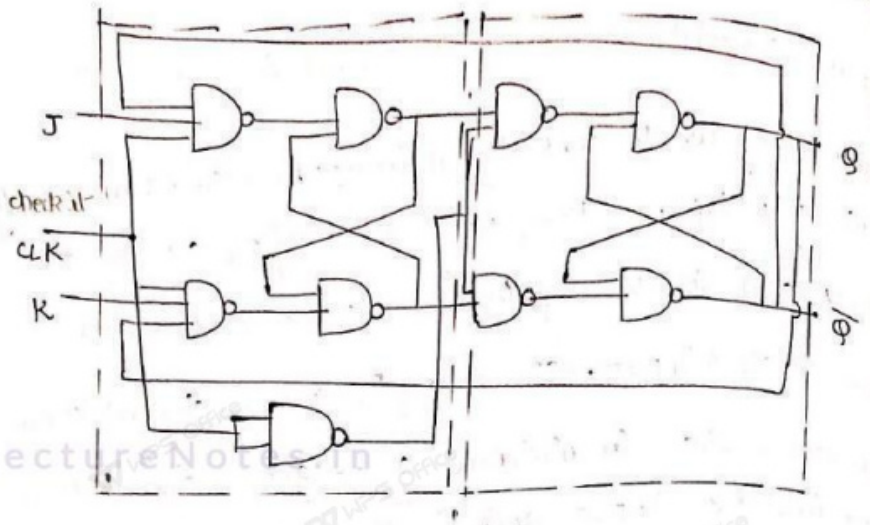
In synchronous system the clock signal is applied simultaneously to all the flip flops, it may undergo varying degree of delay due to wiring between components and arrival of clock input, this delay is called clocked skew. If the clock's skew is minimal a FF get clocked before it receives new input and the clock pulse is delayed significantly the flipflop may changed before the clock pulse arrived. In these situations we may have a kind of race between two competing signals that are attempting to accomplish effects. This is called time race.

Consider the JK FF, when $J = K = 1$, if the width of the clock pulse is too long the state of flip flop will keep on changing (toggling) and at the end of the clock pulse its state will be uncertain. This phenomenon is called race around condition.

Master-slave flip-flop



Logic diagram



Truth table –

Input			Output	Action
J	K	CLK	Q_{n+1}	
0	0	⊓	Q_n	no change
0	1	⊓	0	Reset
1	0	⊓	1	Set
1	1	⊓	$\overline{Q_n}$	Toggle

The 1st FF is called master is driven by the +ve edge of the clock, the second flip flop is called edge slave and is driven by –ve edge.

Therefore when clock input has positive ends the master acts according to its JK inputs but slave do not and when clock input are negative edge the slave & FF copies the master output, so it do not have race around condition

Setup time (t_s) – The minimum time required for which the control level need to be maintained constant on the input terminal of the FF to the arrival of the triggering edge of the clock pulse in order to enable FF.

Hold time (t_h) – The minimum time required for which the control level needs to be maintained constant after the arrival of the triggering edge of the clock pulse in order to enable FF.

f_{max} (max. clock frequency) – It is the highest frequency at which the FF can be reliably triggering.

Power Dissipation – It is the total power consumption of the device. $P = VI$

or

$P = V_{cc} \cdot I_{cc}$, mW

Flip Flop Excitation table –

SR truth table

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

Excitation table

PS	NS	Required input	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK FF truth table

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

Excitation table

PS	NS	Required input	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

DFF

D	Q_{n+1}
0	0
1	1

Excitation table

PS Q_n	NS Q_{n+1}	Required Input D
0	0	0
0	1	1
1	0	0
1	1	1

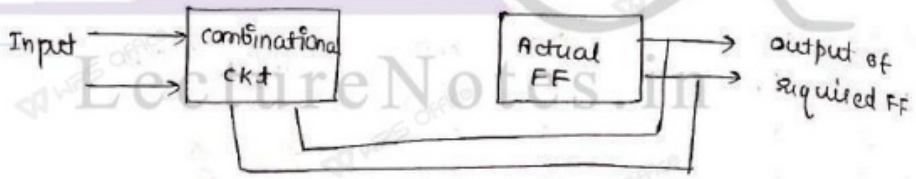
TFF

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

Excitation table

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

Conversion of FFs-



to convert 1 type to another type we have to obtain the expression for the input of the existing FF in terms of required FF.

Conversion of SR to JK -

external inputs J	K	PS Q_n	NS Q_{n+1}	FF inputs S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X

1 1 0 0 1

0	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1
1					

Kmap for S & R –

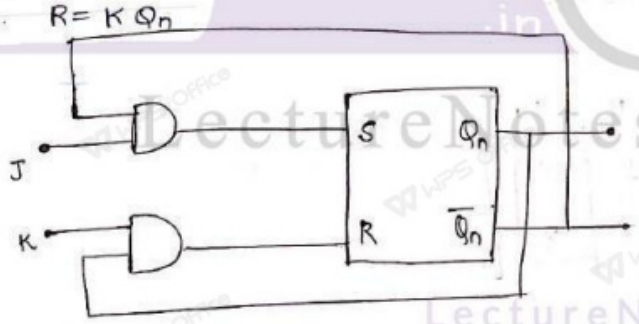
JQn \ KQn	00	01	11	10
0	0	x	0	0
1	1	x	0	1

~~$S = JK\overline{Q_n}$~~

$S = J\overline{Q_n}$

KQn \ JQn	00	01	11	10
0	x	0	1	x
1	0	0	1	0

$R = KQ_n$



logic diagram

conversion of JK Flip Flop to S-R flip flop –

External	PS	NS	FF Inputs

Inputs		Qn	Qn+1	J	K
S	R				
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0

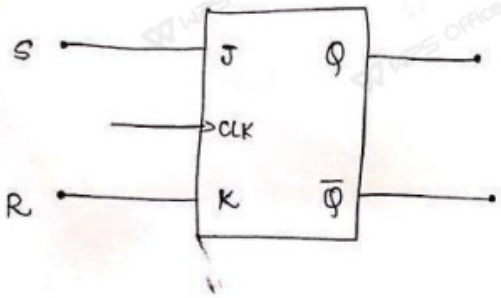
K map for J & K–

S \ RQn	00	01	11	10
0	0	X	X	0
1	1	X	X	X

J = S

S \ RQn	00	01	11	10
0	X	0	1	X
1	X	0	X	X

K = R



J-K to D FF–

External D	PS Q	NS Qn+1	FF Input J	K
	0		0	X

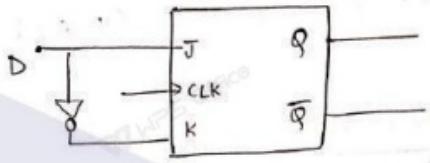
0	0	0	X	1
0	1	0	1	X
1	0	1	X	0
1	1	1		

Qn / D	0	1
0	0	X
1	1	X

$J = D$

Qn / D	0	1
0	X	1
1	X	0

$K = \overline{D}$



Logic diagram.

03|10|2018

Shift Register–

1) A shift register is a set of FFs, suitable for storing binary information.

2) An bit register has a group of n FFs.

3) It is a type of sequential circuit and an important building block used in digital system like multiplier, divider, microprocessor, ~~in buffer reg~~

Buffer Register–

1) It is a simplest register may be used for binary words/ binary information.

Shift register

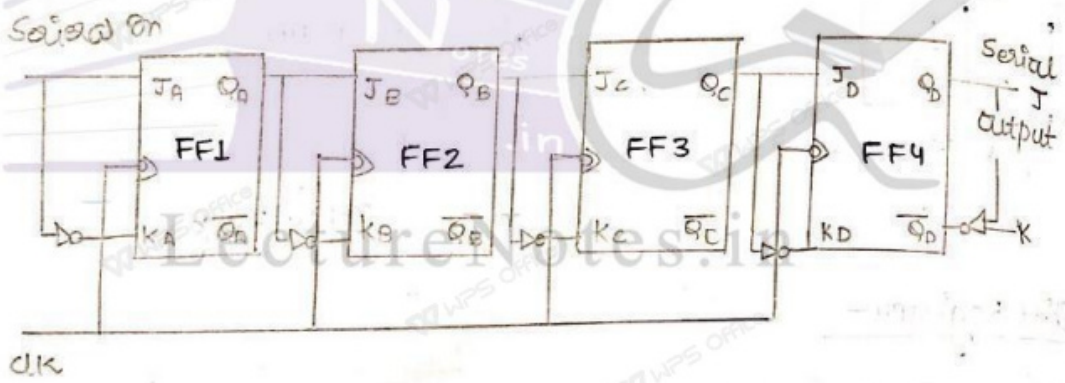
~~It~~ A no. of flipflop is connected together such data may be shifted into or shifted out of them is called a shift register.

The data may be shifted in parallel or serial

the types of shift register are -

1) Serial in serial out (SISO)
2) Serial in Parallel out (SIPO)
3) Parallel in serial out (PISO)
4) Parallel in parallel out (PIPO)

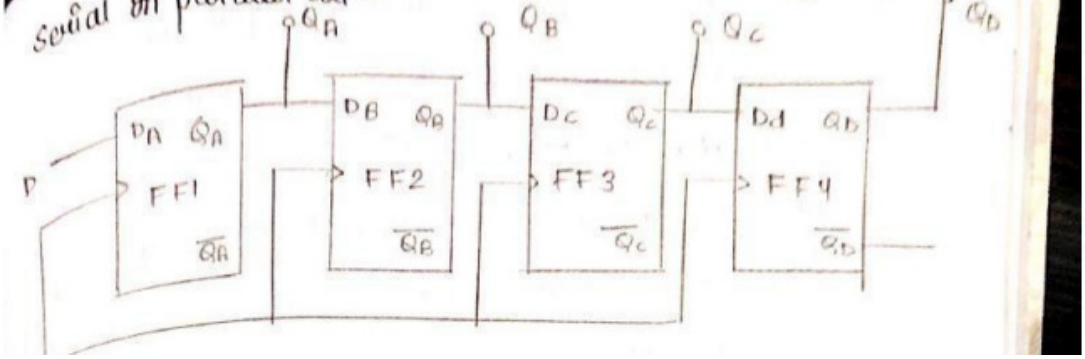
Serial in serial out -

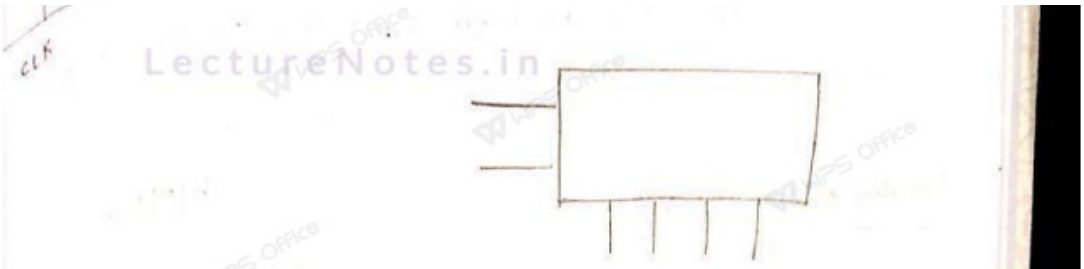


Shifted Pulse	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	Ø 1	0	1	0
4	Ø 1	Ø 1	0	1

input bit is
1101

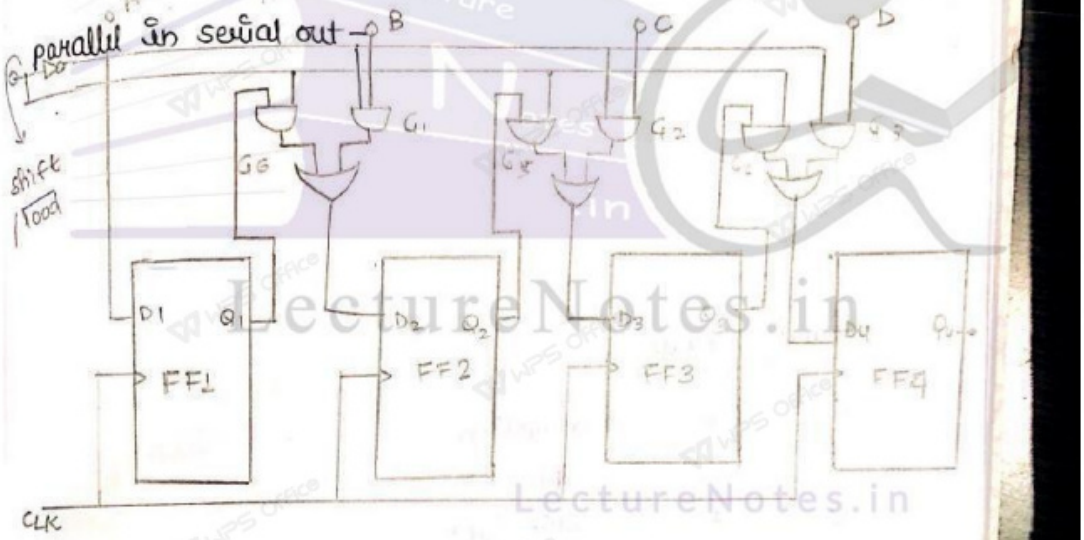
Serial in parallel out -



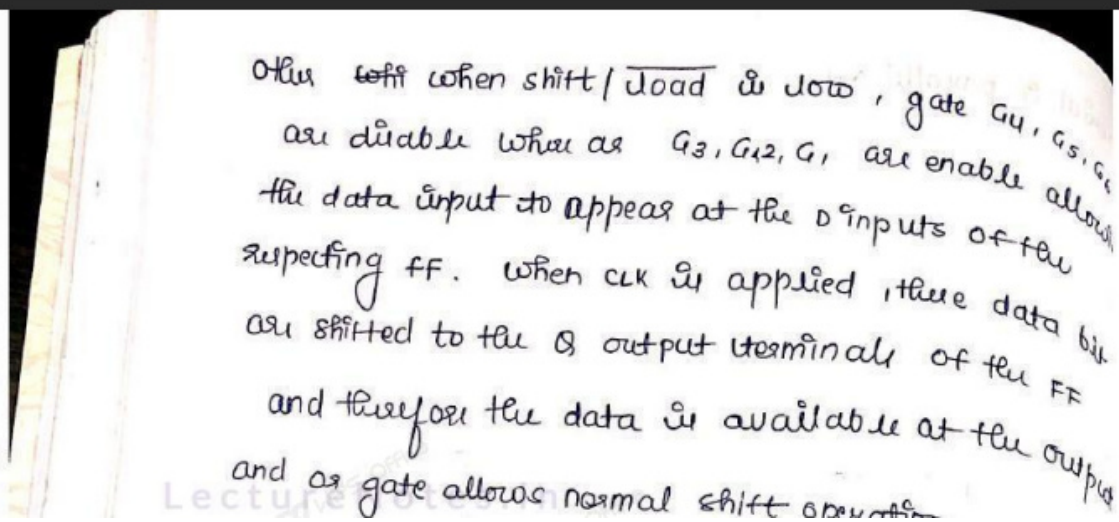


On this type of shift register the bits are entered serially but stored data in the register is shifted in parallel form

parallel in serial out



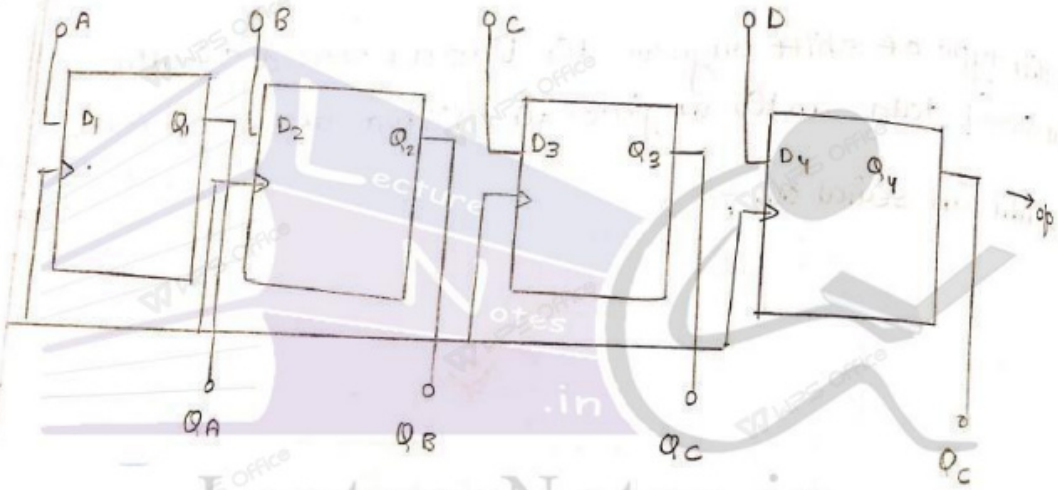
The data bits are entered simultaneously into the register in parallel form but data bit are transferred out serially, the signal shift / $\overline{load}$ allows the data to be enter. in parallel form when shift / $\overline{load}$ is HIGH gate G_1, G_2, G_3 are disable and G_4, G_5, & G_6 are enabled allowing the data bit to shift right, from one stage to



Otherwise when shift / $\overline{load}$ is low, gate G_4, G_5, G_6 are disable where as G_3, G_2, G_1 are enable allowing the data input to appear at the D inputs of the respecting FF. When CLK is applied, these data bits are shifted to the Q output terminals of the FF and therefore the data is available at the output and as gate allows normal shift operation

Parallel in parallel out –

06/11/2018



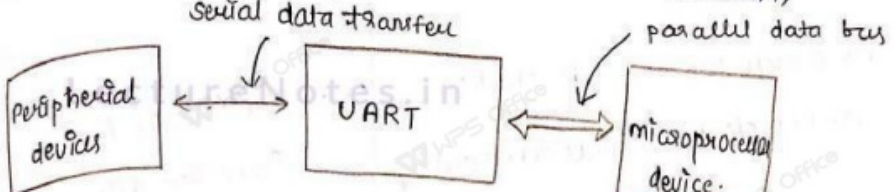
Function table

S_1	S_0	operation
0	0	NC
0	1	shift right
1	0	shift left
1	1	parallel load

Study
Universal

Application of Shift Register –

1) Time delay
2) Serial / parallel data conversion
3) Ring counter
4) UART (universal asynchronous Receiver Transmission)



UART as an interface device

It is a special design interfacing device used to accomplish the communication with external device either transmitting or receiving. Data Receive in serial form and converts transmit in parallel form and vice versa.

counter:-

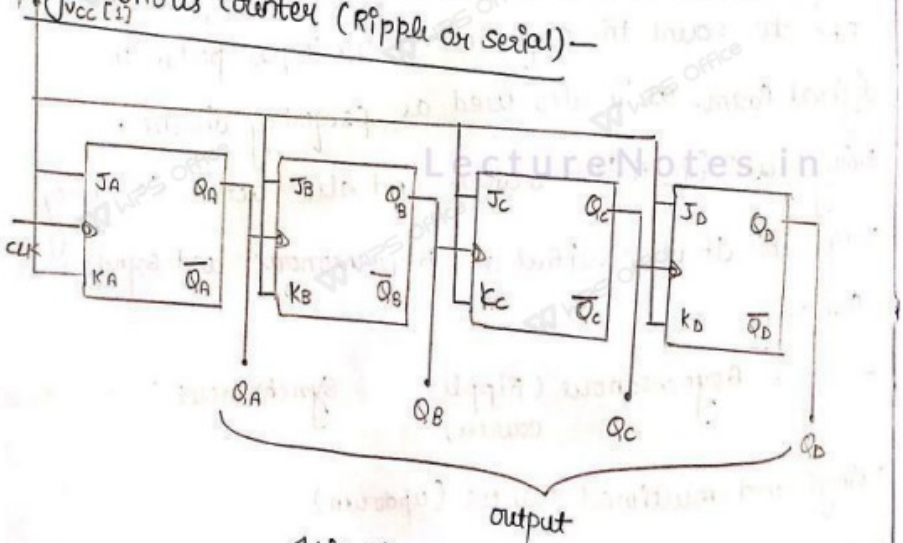
It is a ~~sync~~ sequential circuits consist of a set of ~~ff~~ FFs to count the sequences of the input pulse in digital form. It is also used as frequency divider, timing fun in digital watches and also act as frequency counter and it is classified as Asynchronous and synchronous Counter.

- Asynchronous (Ripple counter) — Synchronous (write diff
- single and multimod counter (updown)
- Modulus counter

Asynchronous	Synchronous
1) The FFs are connected in such way that the output of the first FFs drives the clock for the second FF and so on.	No connection b/w first and second FF and so on
2) For asynchronous all clocks are not clocked simultaneously.	FFs are clocked simultaneously.
3) Design	

...gn and implementation is very simple.	...of is very complex.
4) main drawback is low speed as clock is propagated through a no. of FFs.	FFs are clocked simultaneously, thus propogation delay is less thus very fast.

Asynchronous Counter (Ripple or serial)—

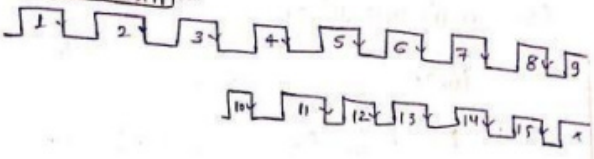


4bit Ripple counter

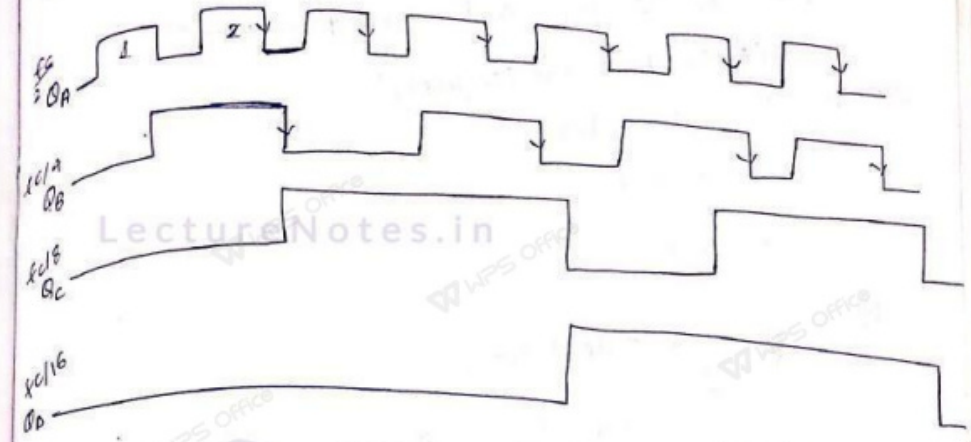
Mod = 2^{n-1}

n = no. of FFs

waveform—



The counter



draw it

properly.

An n bit binary counter repeats the counting sequency for every 2^n clockpulse and has decimate state from 0 to 2^n-1, states = no. of counter.

Truth table

state	Q_D	Q_{BC}	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

It's a up counter

(just study how to know whether the counter is up or down counter)

Qn) A binary ripple counter is required to count upto $16,383_{10}$ decimal counter.

1) How many FFs are required?

2) What is the frequency at the output of MSB?

(8.192 MHz = clock frequency)

Solⁿ ① $2^n - 1 = 16383 - 16388$

$2^n = 16384$

$n = 14$ FF

② frequency at output stage

$$f_{14} = \frac{f_c}{2^{14}} = \frac{8.192 MHz}{2^{14}} = 500\ Hz$$

30/10/18

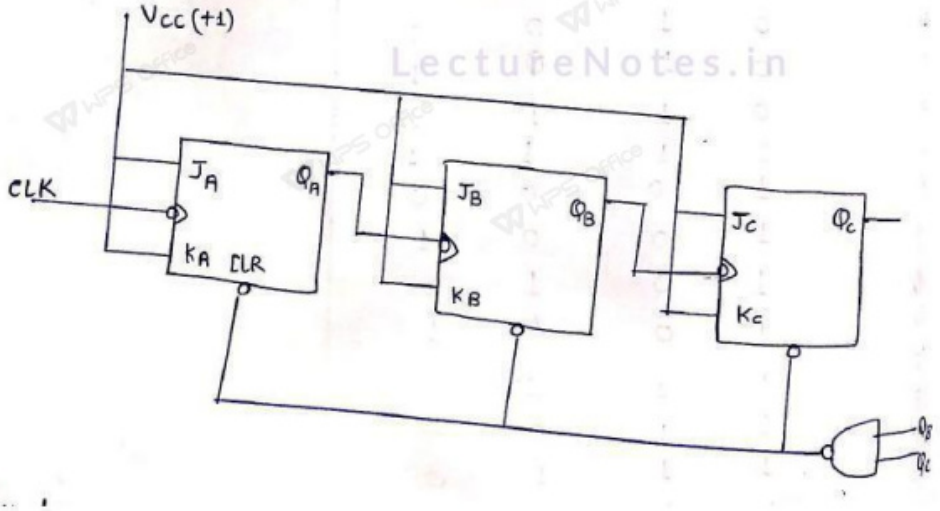
Design of Mod 6 Ripple Counter – (Up counter)

Mod 16 – 4 bit flipflop

0 ... 15

$N \leq 2^n$

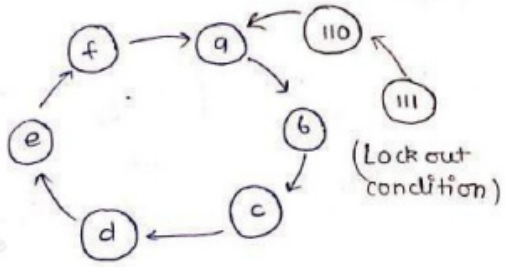
State	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1



Design of a mod 6 Synchronous Counter (Parallel counter)

- no. of FFs required – 3FFs
- state diagram
- state assignment

a = 000, b = 001,
c = 010, d = 011
e = 100, f = 101



state diagram

- excitation table –

PS			NS			Excitation required					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1

1	0	0		1	0	1		x	0	0	x	1	x
1	0	1		0	0	0		1	x	0	x	x	1

- Minimization using K map.

Q_3 \ Q_2Q_1	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_3 = Q_2Q_1$

Q_3 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	0	1	X	X

$K_3 = Q_1$

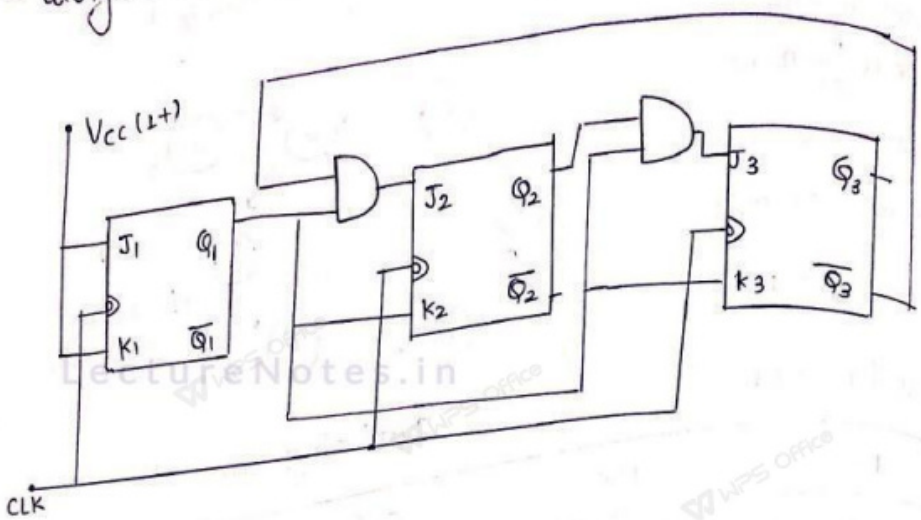
$J_2 = \overline{Q_3}\,Q_1$

$K_2 = Q_1$

$J_1 = 1$

Scanned with CamScanner

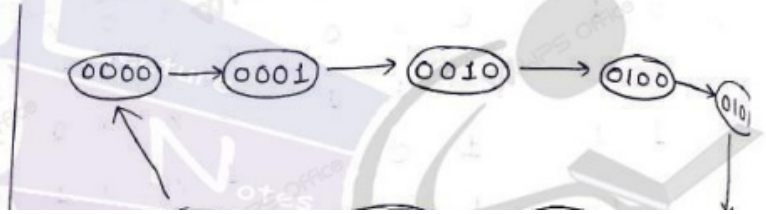
- design of circuit-



* Design of a synchronous BCD counter using JK FF's.

mod 10 / Decade

a 0000
b 1
c 1
d 1



e 1
f 1
g 1
h 1
i 1001

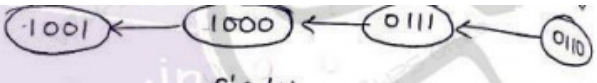


fig (1)

Complete it at previous.

1) no. of Flip Flops –
no. of flip Flops ~~$10 = 2^{n-1}$~~ $N \leq 2^n$

$10 \leq 2^4$

$n = 4FF$

2) States diagram – in fig 1 (a)

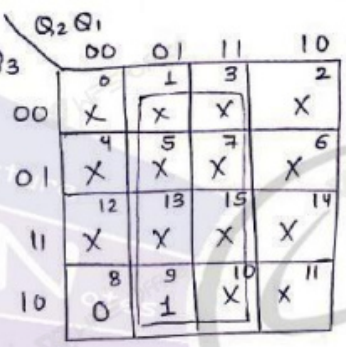
3) excitation table

PS NS Required Excitation

Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	1	0	0	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	1	0	X	0	X	X	1	X	1
0	1	0	0	0	1	1	0	0	X	1	X	0	X	1	X
0	1	0	1	0	1	1	1	0	X	X	0	1	X	X	1
0	1	1	0	1	0	0	0	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	1	0	X	X	0	X	1	X	1
0	0	0	0	0	0	0	0	1	X	X	1	0	X	1	X
1	0	0	1					X	0	X	X	0	X	X	1
1								X	1	0	X				
										0	X				

4) minimization of K map –

$J_4 = Q_3Q_2Q_1$

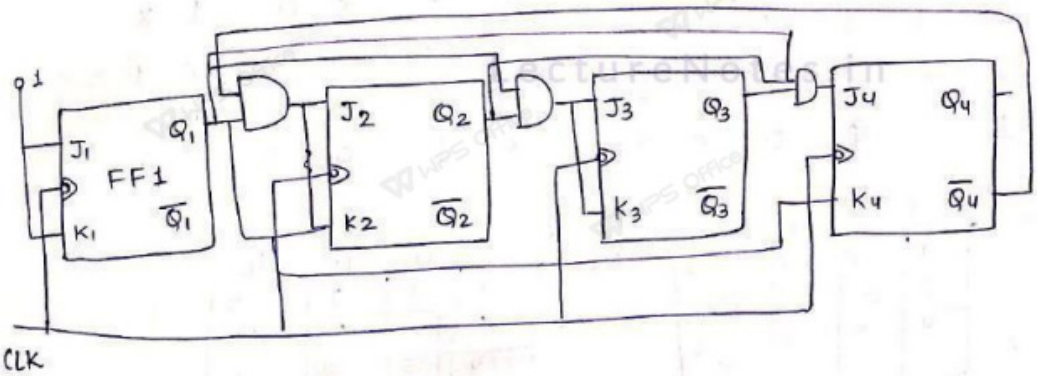


$K_4 = Q_1$

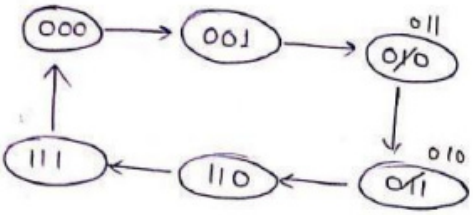
Similarly solving for others we get

$J_3 = Q_2 Q_1$ $\quad K_3 = Q_2 Q_1$

$J_2 = \overline{Q_4} Q_1$ $\quad K_2 = Q_1$

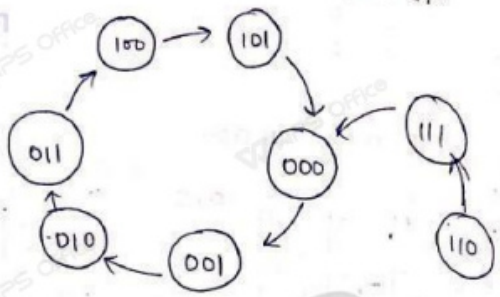


Mod 6 Gray Counter -



	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	
111	

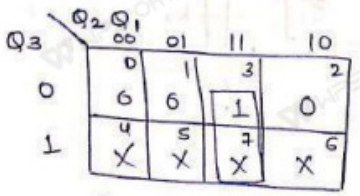
complete it



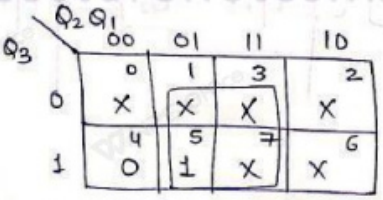
excitation table -

PS			NS			Required excitation					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1

0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	10	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	01	0	0	0	x	1	0	x	x	1



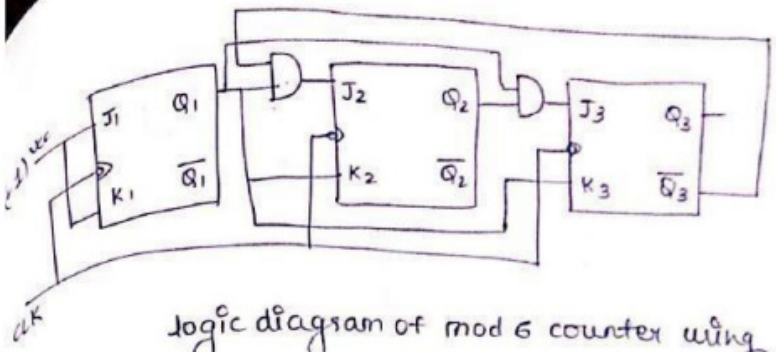
$J_3 = Q_2 Q_1$



$K_3 = Q_1$

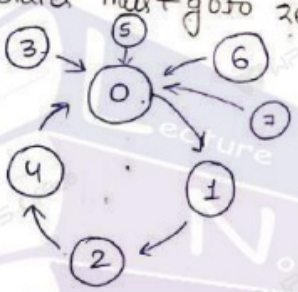
Similarly, $J_2 = \overline{Q_3} Q_1$ $K_2 = Q_1$

$J_1 = 1$, $K_1 = 1$



logic diagram of mod 6 counter using JK Flip Flops.

* Design a type D counter that goes through 0, 1, 2, 4, 0 ---- The undesired states must goto zero (000) in next clock. (Question is based on lock out condition)



PS	NS	Excitation D_3	D_2	D_1
000	001	0	0	1
001	010	0	1	0
010	100	1	0	0

1 0 0	0 0 0	0	0	0

excitation table

Q_3	Q_2Q_1 = 00	01	11	10
	0	1	3	2
0	0	0	0	1
1	4	5	7	6
	0	0	0	0

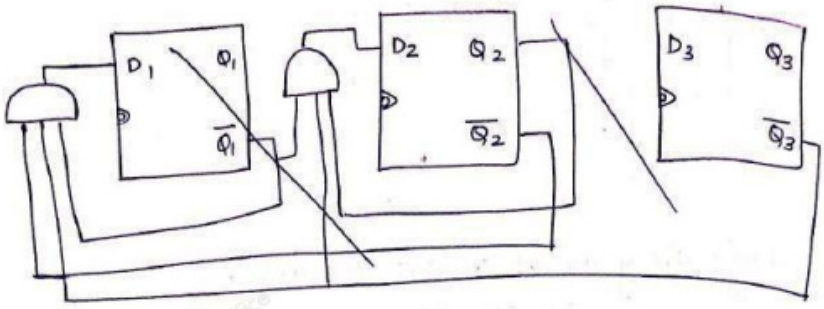
$D_3 = \overline{Q_3} Q_2 \overline{Q_1}$

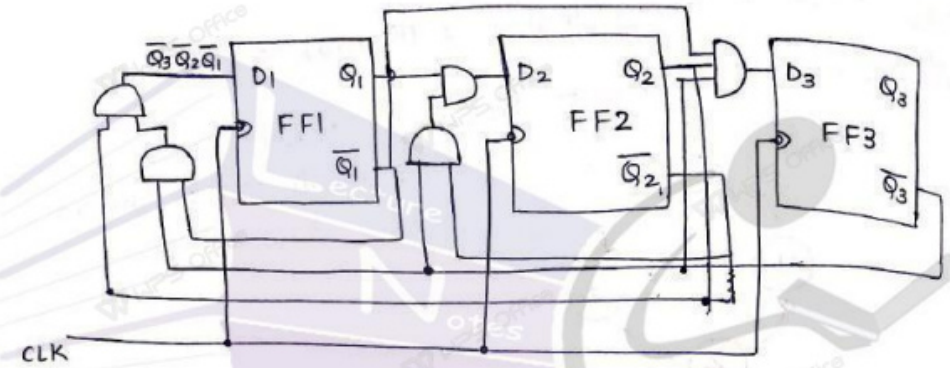
Q_3	Q_2Q_1 = 00	01	11	10
	0	1	3	2
0	1	0	0	0
1	4	5	7	6
	0	0	0	0

$D_1 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1}$

Q_3	Q_2Q_1 = 00	01	11	10
	0	1	3	2
0	0	1	0	0
1	4	5	7	6
	0	0	0	0

$D_2 = \overline{Q_3}\,\overline{Q_2} Q_1$





logic diagram of type D counter that goes through states 0, 1, 2, 4, 0, _____

31|10|18

Lock-out condition –

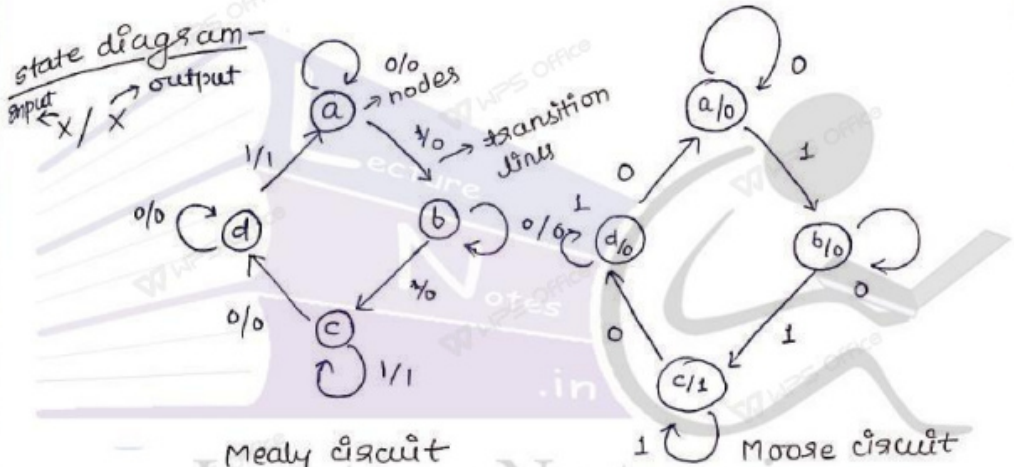
when counter is switch on, during counting the counter may find itself in some invalid state. Subsequent pulses may cause the counter to move from one unused states to another state, the counter may never come to a valid state so counter becomes useless.

31/10/18

Finite State Machine Model(FSM) – [Sequential circuit Part I]

1) It is an abstract model describing the synchronous sequential machine.
2) Those machine whose past history can affect there future behaviour only in a finite no. of ways. Hence called FSM. It is represented by
   I) Moore Model/circuit – ouput depends only on present state of FF.
   II) Mealy Model / circuit – ~~NS of FFs as well as~~ In mealy circuit it is both PS and input state.

State diagram –
input ← x/z → output



State table –

PS	NS, O/P	
	I/P x	
	x=0	x=1
a	a, 0	b, 0
b	d, 0	c, 0
c	c, 1	d, 0
d	b, 0	a, 0

PS	NS I/P(x)		O/P (z)
	x=0	x=1	
a	a	b	0
b	d	c	0
c	c	d	1
d	b	a	0

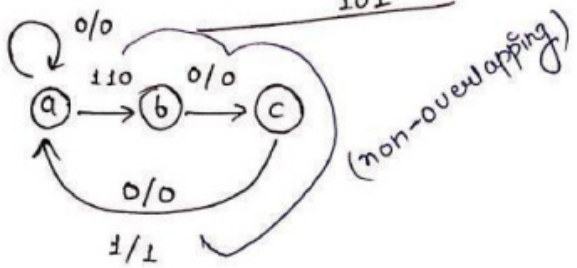
	x=0	x=1
a	a, 0	b, 0
b	b, 1	c, 0
c	d, 0	d, c, 1
d	d, 0	a, 1

Mealy circuit's table

a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	0

Moore table

Sequence detector 06/11/18



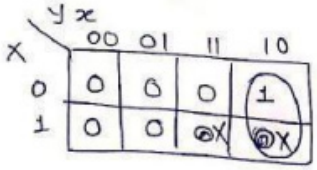
PS	NS		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	b	0	0
c	a	a	0	1

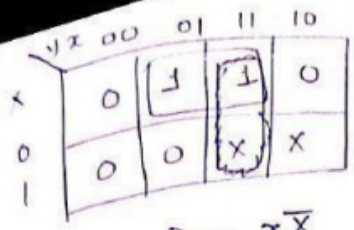
assign a = 0, 0
b = 0 1
c = 1 0

excitation table-

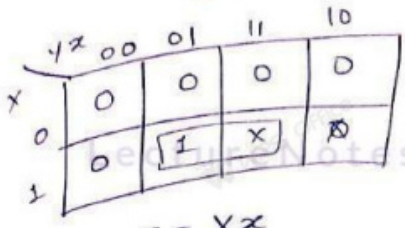
PS		Input	NS		Output	Required Excitation	
x	y	X	x'	y'	Z	Dx	Dy
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	1	0	0	1

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

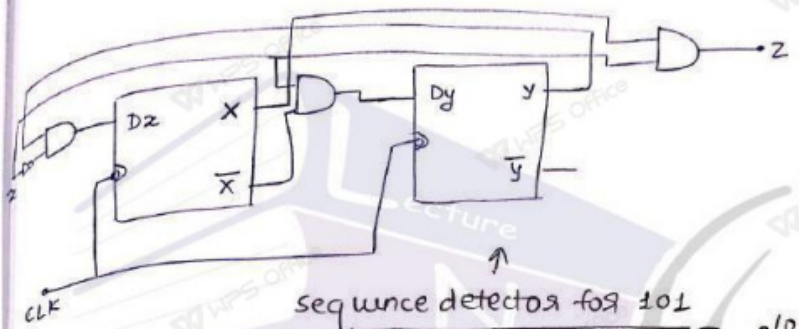




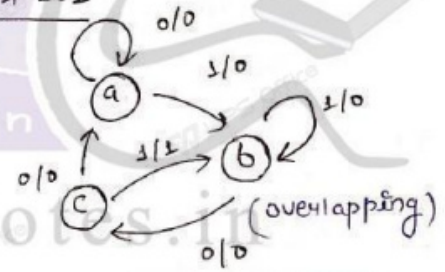
$D_y = x\bar{X}$



$Z = Xx$

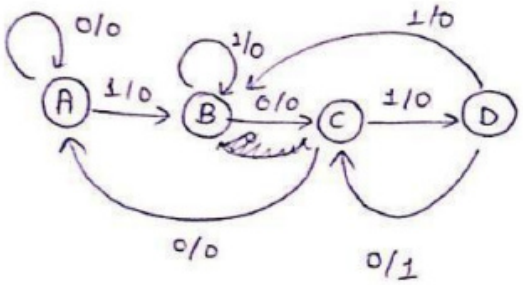


sequence detector for 101



PS	NS		Z	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	b	0	0
c	a	b	0	1

* Design of a sequence detector of 1010



A = '-' seen

B = '1' seen

C = '10' seen

D = '101' seen

Let A = 00

B = 01

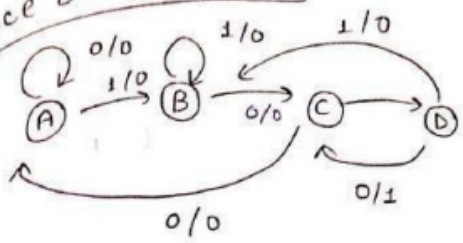
C = 10

D = 11

At C if it receives '0' the sequence is 100 which is not part of valid sequence and can not be used to start so machine goes to initial state A to search at the sequance

at D if it receives '1' the last 4 bit is 1011 which is not a valid sequence, since overlapping is permitted then machine will utilites 10 the last 2 bits to get another sequence 10 10

Sequence detector 1010 –



State table

PS	NS	
	x=0	x=1
A	A, 0	B, 0
B	C, 0	B, 0
C	A, 0	D, 0
D	C, 1	B, 0

Transition table –

PS	NS		Output	
	x=0	x=1	x=0	x=1
A → 0 0	0 0	0 1	0	0
B → 0 1	1 0	0 1	0	0
C → 1 0	0 0	1 1	0	0
D → 1 1	1 0	0 1	1	0

Excitation table –

PS y_1 y_2 (Q_2 Q_1)	Input x	NS	IP to FFs D_1 D_2	Output z
0 0	0	0 0	0 0	0
0 0	1	0 1	0 1	0
0 1	0	1 0	1 0	0
0 1	1	0 1	0 1	0
1 0	0	0 0	0 0	0
1 0	1	1 1	1 1	1
1 1	0	1 0	1 0	0

1 1					
1 1	1	0 1	0 1		

K Map for D_1 and D_2 –

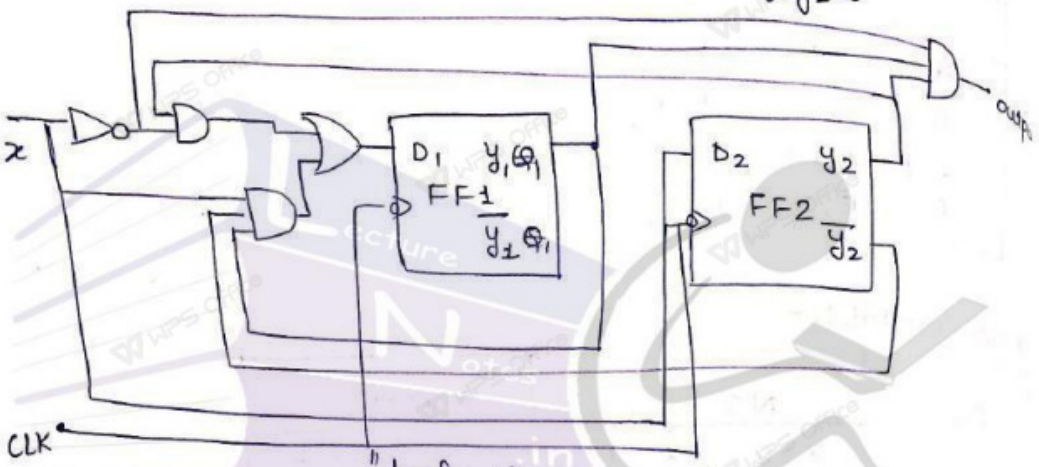
y_1 \ y_2x	00	01	11	10
0	0	0	0	1
1	0	1	0	1

$D_1 = y_2\bar{x} + y_1\bar{y}_2x$

y_1 \ y_2x	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$D_2 = x$

Output – $y_1y_2\bar{x}$



"logic diagram of the sequence (1010) detectector using D FlipFlop"

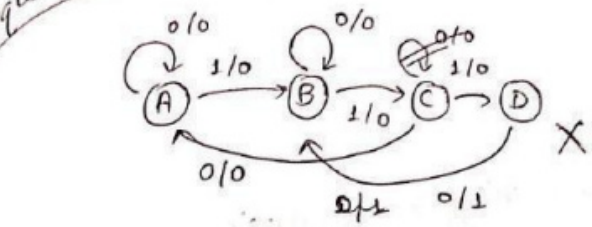
Rule

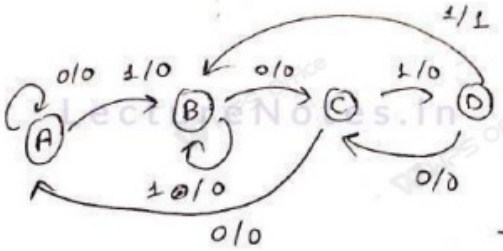
08/11/18

Mealy model –

1) For non OL the last bit to the next.
2) For 1 bit OL, compare the last bit to 1 bit state.
3)

Sequence detector (1011) -





1 0 1 1

overlapping

state table

PS	NS $x=0$	NS $x=1$
A	A, 0	B, 0
B	C, 0	B, 0
C	A, 0	D, 0
D	C, 0	B, 1

transition table -

PS	NS $x=0$	NS $x=1$	Output $x=0$	Output $x=1$
A → 00	00	01	0	0
B → 01	10	01	0	0
C → 10	00	11	0	0
D → 11	10	01	0	1

excitation table -

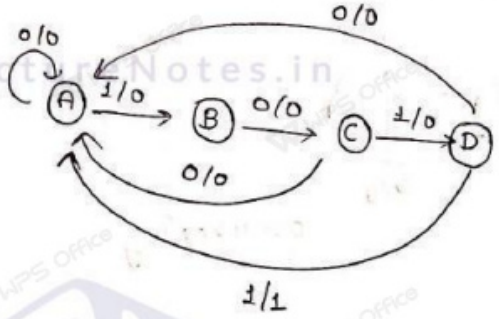
PS	Input	NS
00	0	00
00	1	01
01	0	10
01	1	01
10	0	00
10	1	11
11	0	10
11	1	01

Input of FF D_1	Input of FF D_2	Z
0	0	0
0	1	0
1	0	0
0	1	0
0	0	0
1	1	0
1	0	0
0	1	1

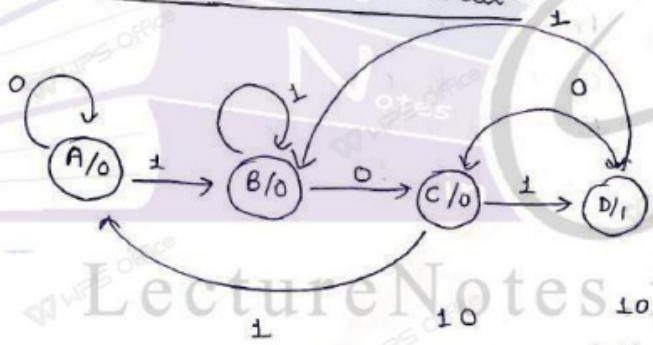
Same for $D_1 = y_2\bar{x}_1 + y_1\bar{y}_2 x$, $D_2 = x$

output $(z) = y_1 y_2 x$

Sequence detector (1011) ~~using~~ non overlapping -



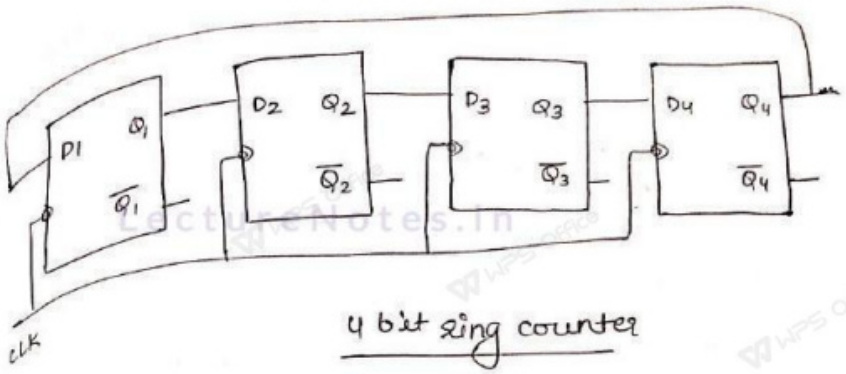
Sequence detector (101) Moore model -



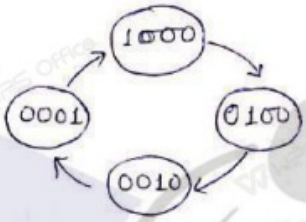
PS	NS		Z
	$x=0$	$x=1$	
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1

14/11/18

Ring counter –



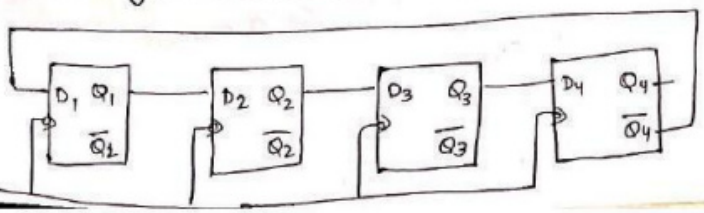
4 bit ring counter



State diagram

Q1	Q2	Q3	Q4	After clock pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

Twisted Ring counter (Johnson counter) –



~~X1~~

N bit
for ring counter, can count only N bits but in twisted ring counter, N FF can have 2N unique states and can count upto 2N pulses. So it is called mod-2N counter.

Logic families

IC's are fabricated using various technologies such as TTL, ECL, IIL. which are by bipolar transistors while as MOS and CMOS technology are unipolar ~~Most~~ MOSFET, FET, BJT → Bipolar

unipolar

TTL- T^2L- Transistor-2 logic

EC- Emitter coupled

IIL- Integrated Injection logic

MOS- Metal oxide semiconductor

CMOS-

power dissipation- $P_D = V_{CC} \times I_{CC}$

Characteristics of ICs-

1) power dissipation
2) propogation delay
3) threshold voltage - the voltage at which the input of a gate that changes in the ~~stage~~ o state of a output.
4) Fan-in
5) Fan-out
6) Noise margin

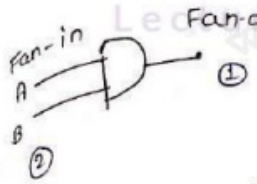
it is the no. of gates inputs that the gate is designed to handle

Fanin-

" output " ". Also called

Fan-out

loading factor of a logic gate, and is defined as the max. of load that the output of the gate can drive without affecting its normal operation.



noise margin - the noise immunity of a logic suffers to the circuits ability to tolerate noise voltage as its input. A quantitative measure of immunity is called noise margin.

operating temp. of ICs = 0-70°C - commertial

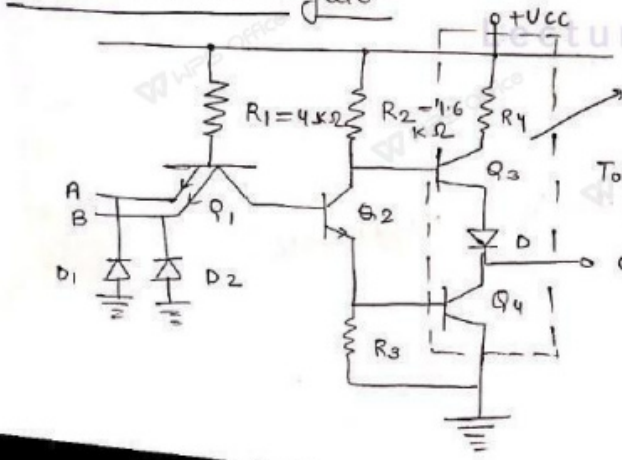
0-85°C - Industrial

-55°C - 125°C - Defence

Transistor-Transistor (TTL or T^2L) logic-

→ It is commonly used bipolar digital ICs family.

Two input TTL Nand gate-



Case I- When both inputs A, B is high the base, emitter junc of Q_1 are reverse bias so current flows to the

ope

that makes Q_2 turnson.

Current from Q_2 flows towards the base of Q_4 so Q_4 turnson.

therefore Q_3 is off and V_0 is its low level

So output is logic 0.

Case II – When A & B both are low the base emitter junction is forward bias and collector base junction is reverse bias of Q_1 therefore current flows to the ground to Q_1 and the base of Q_1 is 0.7 V that makes Q_2 off.

When Q_2 is off Q_4 does not get the required base derived voltage so Q_4 is off so Q_3 is on the output voltage V_0 is given by $V_{cc} - R_1 V_{R2} - V_{BE3} - V_0$

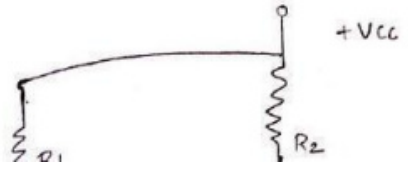
$= 3.4$ to 3.8 V, which is logic 1.

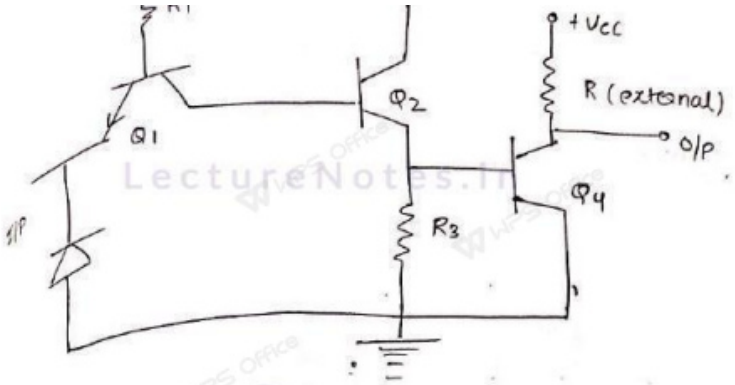
$$V_0 = V_{cc} - V_{R2} - V_{BE3} - V_D$$

So circuit acts as two input NAND gate.

15/11/18

Open collector TTL NAND Gate–





External pull transistor

In order to get proper high or low logic levels and external pull transistor is connected to Vcc from Q_4, and the output is at the collector of Q_4 so name as open collector. When Q_4 is off the output is pulled to Vcc (1) through R. When Q_4 is ON, the output is connected near ground. Therefore the value of R must be choosen when 1, output goes to low and while others are high.

Emitter coupled Logic (ECL)-

It is also called current mod logic and is the fastest in all the logic families. It use BJT's that are couplet at their emitter. It operates on the principle of current switching where by a fixed bias current less than I_C (saturation) is switched from one transistor collector to an other. It main drawbacks 1) high cost 2) low noise margin & 3) high power dissipation.

advantage- 1) never saturates
2) switching speed is higher
3) large fanout
4) logic levels are negative
-0.9 V for logic 1
-1.7 V for logic 0

ECL OR/NOR gates-

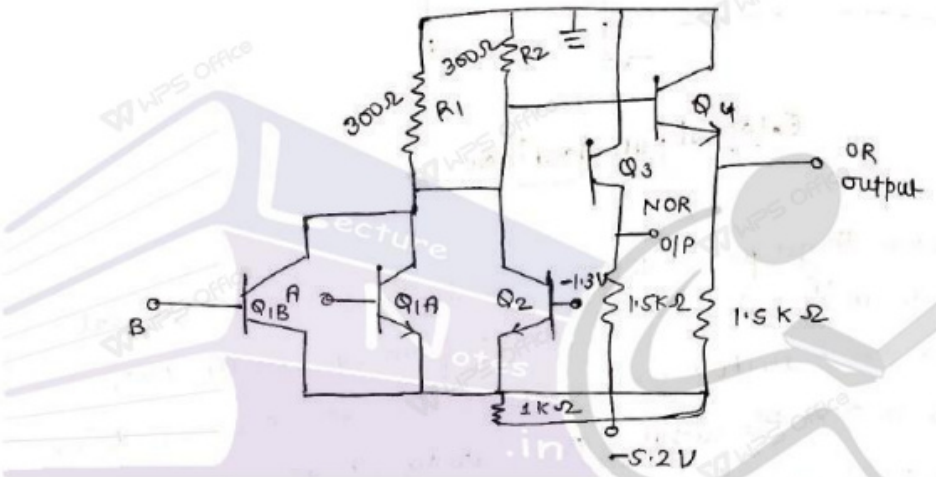


Fig - circuit diagram of ECL OR/NOR gate

Symbol.

OR			NOR		
A	B	Y	A	B	Y
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	0

transistors Q_2 and Q_1A → form differential amplifier

Q_3 and Q_4 ↔ are emitter followers

(Phase is same)

when A & B are low (-1.7V) Q_2 is more FB than Q_1, so Q_2 is ON and Q_1 is off. The value of R_2 is such that the current flowing through Q_2 puts the collector is ~~above~~ about -0.9V therefore the Q_2 is at $(-0.9 - 0.8) = -1.7V$ so OR gate is low.

then the base current Q_3 paying through R_1 is very

small the collector current of Q_1 is about $(-0.1V - 0.8V) = -0.9V$

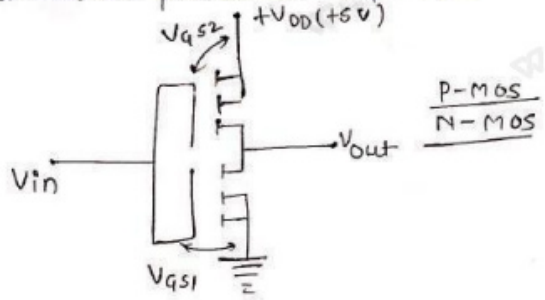
that is the nor output which is logic 1.

When A and B are high or ~~but~~ A is high & B is high the corresponding transistors are ~~not~~ on and they are more FB than Q_2 so Q_2 is off, Q_1 is on so collector of Q_1A & Q_1B are at $-0.9V$ which makes nor output $-0.9 - 0.8V = -1.7V$ which is logic 0.

Similary only small base current of Q_4 flows through R_2 so collector of Q_4 is at $-1 - 0.8 = -0.9$ V which is logic 1.

as per the process the circuit used works ~~at and~~ as OR and NOR.

CMOS Logic inverter – (Complimentary mos) – Two families are faster in operation and consume less power ~~and~~ but fabrication process is complicated.



It consists a N-MOS transistor Q_1 and PMOS. Q_2 the input is connected to common gate and output is taken from drain. The +ve voltage is connected to the source of Q_2 and source of Q_1 is grounded.

I/P

① $V_{in} = 0V$

$V_{GS2} = -5V$

$V_{GS1} = 0V$

So Q_2 = ON && Q_1 is OFF

output $V_{out} = 5V$ (logic 1)

(ii) $V_{in} = +5V$

$V_{GS2} = 0V$

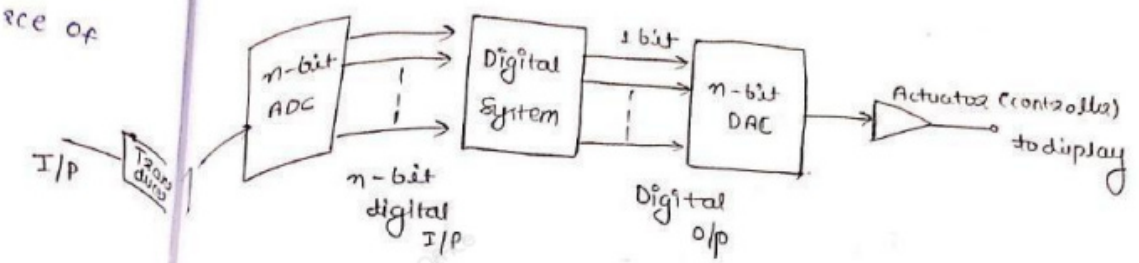
$V_{GS1} = 5V$

So Q_1 = ON && Q_2 is OFF

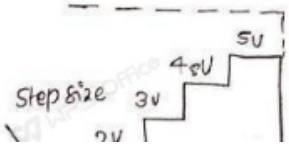
So $V_{out} = 0V$ so (logic 0)

the input is
from
ece of

Analog to digital (ADC) & Digital analog converter DAC-



Digital System



% Resolution = $\frac{\text{step size}}{\text{Full scale}} \times 100\%$

Full scale = No. of steps × step size

% Resolution = $\frac{\text{step size}}{\text{No of steps} \times \text{step size}} \times 100\%$

$= \frac{1}{\text{No. of steps}} \times 100\%$.

% Resolution = $\frac{1}{2^N - 1} \times 100\%$.

Qu) For 6 bit DAC find Resolution.

% Resolution = $\frac{1}{2^6 - 1} \times 100\%$

$= 1.587\%$ ans

DAC

* A 6-bit DAC has step size of 50mV. Determine Full scale o/p voltage and %age of resolution.

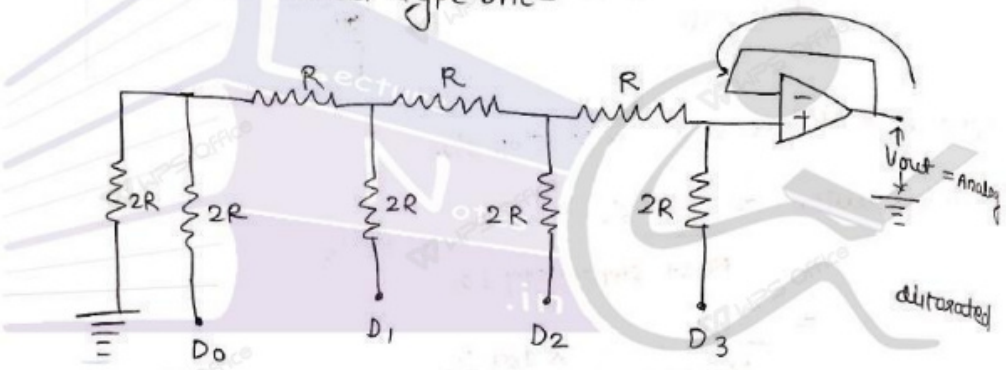
Full scale = No. of steps × step size

$\Rightarrow$ no. of steps = $2^N - 1 = 2^6 - 1 = 63$

Full scale = $63 \times 50mV = 3.15\ V$

% Resolution = $\frac{1}{63} \times 100 = 1.587\%$.

① The R–2R Ladder type DAC–

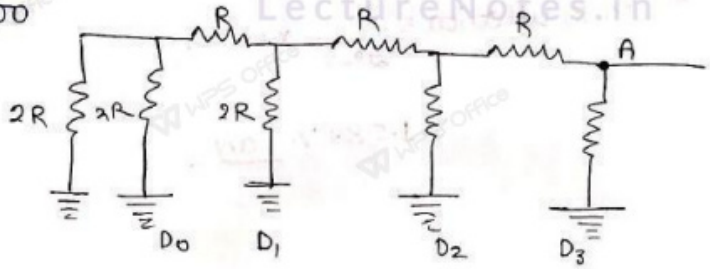


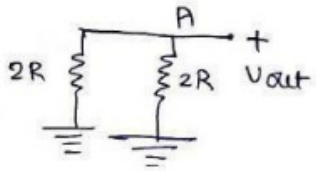
LSB $V_{out} = E\left[\frac{2R}{2R+2R}\right] = \frac{E}{2} = 2.5\,V$ MSB

of E = 5V

– The operation–

Case 1000





Case 0100, $n=2$, $V_{out} = \frac{E}{4}$

0010, $n=1$, $V_{out} = \frac{E}{8}$

0001, $n=0$, $V_{out} = \frac{E}{16}$

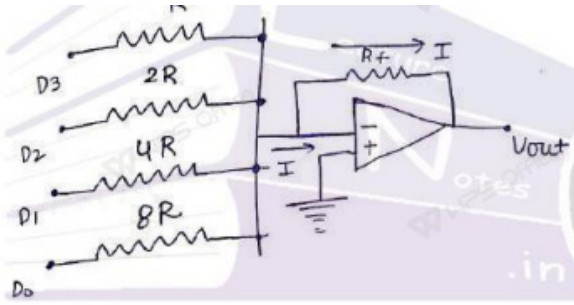
general eqn–

$$V_{out} = \frac{E}{2^{N-n}}$$

N = total no. of bit

n = bit position

② weighted Resistor type DAC–



→ op-amp adds and inverts

$$V_{out} = -\left[D_3 + \frac{D_2}{2} + \frac{D_1}{4} + \frac{D_0}{8} \right] \times \frac{R_f}{R}$$

D_3 is amplified by $\frac{R_f}{R}$

D_2 " " by $\frac{R_f}{2R}$

D_1 " $\frac{R_f}{4R}$

D_0 " " $\frac{R_f}{8R}$

Qu1 What are the output voltage caused by logic 1, in each bit position in an 8 bit ladder if the input levels for 0 is zero volt and for 1 is 10 volt.

~~$D_E = \frac{E}{2^N - 1}$~~ $D_r = \frac{E}{2^{N-n}}$

10000000

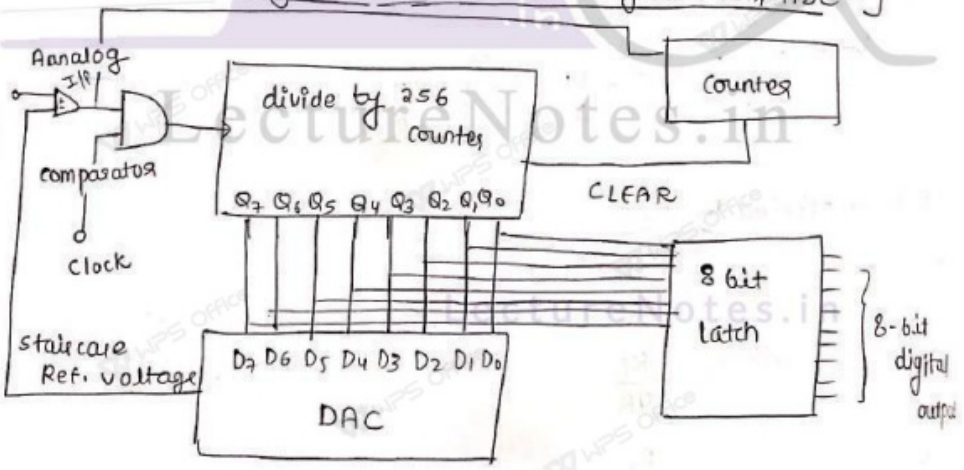
$D_7 = \frac{E}{2^{8-7}} = E/2 = 5\,V$

$D_6 = \frac{E}{2^{8-6}} = \frac{10}{4} = 2.5\,V$

$D_0 = \frac{E}{2^{8-0}} = \frac{E}{2^8} = \frac{10}{256} = 0.039\,V$

ADC

① The counter type ADC – also called Digital Ramp ADC



$$\% \text{ Resolution} = \frac{FSR}{2^N}$$ (full scale reading)

N = no. of bits in a counter.