DIGITAL ELCTRONICS

II B.Tech
I SEM

UNIT I :- Number System and Boolean Algebra

Digital systems, Binary numbers, Number Base conversions, octal and Hexadecimal numbers, complements, signed Binary Numbers, Binary codes, Binary storage and Registers, Binary logic, Axiomatic Definition of Boolean Algebra, Basic theorems and properties of Boolean Algebra, Boolean Functions, canonical and standard Forms, other logic operations, Digital logic Gates, Integrated circuits, Gate-level minimization; The Map method, Four-variable K-Map, Five-variable K-Map, product of sums simplification, Don't-care conditions, NAND & NOR Implementation, Exclusive-OR Functions.

UNIT-II :- combinational logic

combinational circuits, Analysis procedure, Design procedure, code-conversion, Binary Adder-Subtractor, carry propagation, Half Subtractor, Full subtractor, Binary Subtractor, Decimal Adder, BCD adder, Binary Multiplier, Magnitude comparator, Decoders, Encoders and multiplexers.

UNIT-III :- Sequential logic

Synchronous; sequential circuits, storage elements: Latches, introduction to multivibrators, Flip-Flops, Analysis of clocked sequential circuits, State reduction and Assignment, Asynchronous: Analysis procedure, circuits with latches, Design procedure, Reduction of state and flow tables,

Race-Free State Assignment, Hazards.

UNIT-IV :- Registers and counters

Registers with parallelload, shift registers; Serial Transfr, Serial Addition, Universal shift Register, Ripple counters; Binary Ripple counter, BCD Ripple counter, Synchronous counters; Binary counter, UP-Down counter, BCD counter, Binary counter with Parallel load, counter with Unused States, Ring counter, Johnson counter.

UNIT-V :- Memory and Programmable Logic

Random-Access Memory, write and Read operations Timing waveform, Types of Memories, Memory Decoding; Internal construction, coincident Decoding, Address Multiplexing, Error Detection & correction; Hamming code, Single-Error correction, Double-Error Detection, Read only Memory; combinal circuit Implementation, Types of ROM's, combinational PLD's, Programmable logic Array, Programmable Array logic, Sequential, Programmable Devices.

09/07/2014

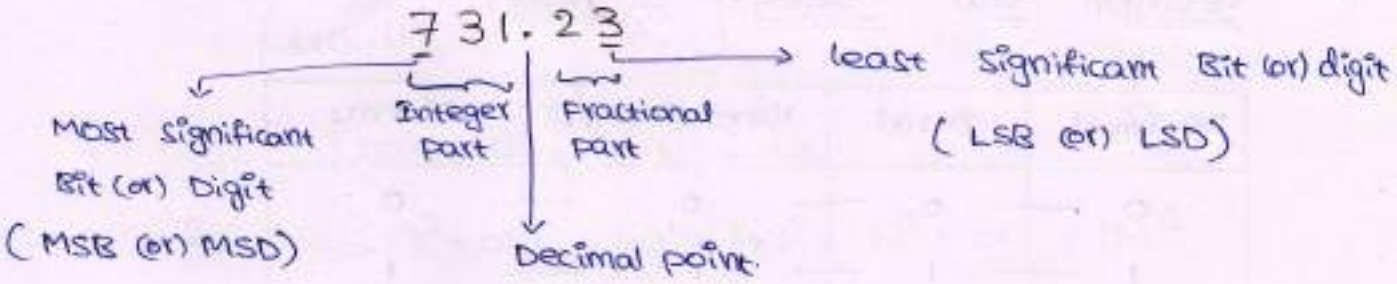
UNIT-I

1. NUMBER SYSTEMS AND BOOLEAN ALGEBRA

1. Decimal number System:-

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (10 digits)

Ex:- 73_{10}, 48.92_{10} → Base (or) Radix (or) Index.



10^2 10^1 10^0 10^{-1} 10^{-2} → positional values (or) weights.
7 3 1 . 2 3

2. Binary number System:-

0, 1 (2 digits)

Ex:- 1101_2, 11001.0_2

2^2 2^1 2^0 2^{-1} 2^{-2} → positional values (or) weights
1 0 1 . 0 1
↓
Binary point

3. Octal number System:-

0, 1, 2, 3, 4, 5, 6, 7 (8 digits)

Ex:- 73_8, 4072_8, 123.65_8

8^2 8^1 8^0 8^{-1} 8^{-2} → weights (or) positional values.
1 2 3 . 6 5
↓
Octal point.

4. Hexa decimal number system:-

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (16 digits)

Ex:- $B73_{16}$, $57A.C3_{16}$

16^2 16^1 16^0 16^{-1} 16^{-2} → weights (or) positional values.

1 2 A . 1 3

↓

Hexa decimal point.

* Relation b/w number systems:-

Decimal	Octal	Hexa decimal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111
16	20	10	10000
17	21	11	10001

Number Base conversion :-

1. Binary to Decimal conversion :-

Ex:- ① $(101.11)_2 = (\ ?\ )_{10}$

$2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$

Sol:- $1\ 0\ 1 . 1\ 1$

$= 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$

$= 4 + 0 + 1 + 0.5 + 0.25 = 5.75$

$\boxed{(101.11)_2 = (5.75)_{10}}$

② $(1011.101)_2 = (\ ?\ )_{10}$

Sol:- $1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 8 + 2 + 1 + \frac{1}{2} + \frac{1}{8}$

$= (11.625)_{10}$

$\boxed{(1011.101)_2 = (11.625)_{10}}$

2. Octal to Decimal conversion :-

Ex:- ① $(75.3)_8 = (\ ?\ )_{10}$

Sol:- $= 7\times8^1 + 5\times8^0 + 3\times8^{-1}$

$= 56 + 5 + \frac{3}{8} = (61.375)_{10}$

$\boxed{(75.3)_8 = (61.375)_{10}}$

② $(624.712)_8 = (\ ?\ )_{10}$

Sol:- $6\times8^2 + 2\times8^1 + 4\times8^0 + 7\times8^{-1} + 1\times8^{-2} + 2\times8^{-3}$

$= 6\times64 + 16 + 4 + \frac{7}{8} + \frac{1}{64} + \frac{2}{64\times8}$

$= 384 + 20 + 0.875 + 0.015625 + 0.00390625$

$= (404.894)_{10}$

$\boxed{(624.712)_8 = (404.894)_{10}}$

Ex:- ③ $(482.31)_8 = (\ ?\ )_{10}$

It is not an octal number.

So, conversion is not possible.

③ Hexadecimal to Decimal conversion :-

Ex:- ① $(7A2.C9)_{16} = (\ ?\ )_{10}$

Sol:- $7\times16^2 + A\times16^1 + 2\times16^0 + C\times16^{-1} + 9\times16^{-2}$

$= 7\times256 + 10\times16 + 2\times1 + \frac{12}{16} + \frac{9}{256}$

$= 1792 + 160 + 2 + \frac{3}{4} + 0.3515625$

$= (1954.78)_{10}$

$\boxed{(7A2.C9)_{16} = (1954.78)_{10}}$

② $(CD3.B7)_{16} = (\ ?\ )_{10}$

Sol:- $C\times16^2 + D\times16^1 + 3\times16^0 + B\times16^{-1} + 7\times16^{-2}$

$= 12\times256 + 13\times16 + 3\times\frac{11}{16} + \frac{7}{256}$

$= (3283.715)_{10}$

$\boxed{(CD3.B7)_{16} = (3283.715)_{10}}$

④ Base 5 no. to Decimal conversion :-

Ex:- ① $(431.23)_5 = (116.52)_{10}$

Sol:- $4\times5^2 + 3\times5^1 + 1\times5^0 + 2\times5^{-1} + 3\times5^{-2}$

$= 100 + 15 + 1 + \frac{2}{5} + \frac{3}{25}$

$= (116.52)_{10}$

$\boxed{(431.23)_5 = (116.52)_{10}}$

⑤ $(231.12)_3 = (\ ?\ )_{10}$

Sol:- It is not a Base 3 no.

Base 3 no. digits are 0, 1, 2.

⑥ Decimal to Binary conversion :-

Ex:- ① $(13.25)_{10} = (\ ?\ )_2$

Sol:- Integer part conversion

$$\begin{array}{r|l} 2 & 13 \\ 2 & 6 - 1 \\ 2 & 3 - 0 \\ 2 & 1 - 1 \\ & 0 - 1 \end{array}$$

↑ all remainders from bottom to top

$(13)_{10} = 1101_2$

Fractional part conversion

$0.25 \times 2 = 0.5$

$0.5 \times 2 = 1.0$ ↓

Integer parts of all product terms from top to bottom.

$0.25_{10} = 0.01_2$

$\boxed{(13.25)_{10} = (1101.01)_2}$

② $(5.6)_{10} = (?)_2$

Sol:- I.P.C.

$$\begin{array}{r|l} 2 & 5 \\ 2 & 2 - 1 \\ 2 & 1 - 0 \\ & 0 - 1 \end{array}$$

↑

$(5)_{10} = (101)_2$

F.P.C

$0.6 \times 2 = 1.2$

$0.2 \times 2 = 0.4$

$0.4 \times 2 = 0.8$

$0.8 \times 2 = 1.6$

$0.6 \times 2 = 1.2$

$0.2 \times 2 = 0.4$

$0.4 \times 2 = 0.8$

$0.8 \times 2 = 1.6$.

↓ $0.\overline{1001}$

$(5.6)_{10} = (101.\overline{1001})_2$

Note :- If there is no repetation in that case the process of multiplication is to be stopped after 4 or 5 places.

10/07/2014

7. Decimal to octal conversion :-

Ex:- (i) $(73.625)_{10} = (\ ?\ )_8$

Sol:- Integer conversion

8	73	
8	9	-1 ↑
8	1	-1
	0	-1

$(73)_{10} = (111)_8$

Fractional part convertion

$0.625 \times 8 = 5.000$ ↓

$(0.625)_{10} = (0.5)_8$

$\therefore \boxed{(73.625)_{10} = (111.5)_8}$

(ii) $(296.198)_{10} = (\ ?\ )_8$

Sol:- Integer part conversion

8	296	
8	37	-0 ↑
8	4	-5
	0	-4

$(296)_{10} = (450)_8$

Fractional part convertion

$0.198 \times 8 = 1.584$
$0.584 \times 8 = 4.672$
$0.672 \times 8 = 5.376$ ↓
$0.376 \times 8 = 3.008$
$0.008 \times 8 = 0.064$
$0.064 \times 8 = 0.512$

$(0.198)_{10} = (0.145300)_8$

$\boxed{(296.198)_{10} = (450.145300)_8}$

8. Decimal to Hexa Decimal :-

Ex:- (i) $(1954.785)_{10} = (\ ?\ )_{16}$

Sol:- Integer part conversion

16	1954	
16	122	-2 ↑
16	7	-(10)A
	0	-7

$(1954)_{10} = (7A2)_{16}$

Fractional part conversion

$0.785 \times 16 = 12.56$ C
$0.56 \times 16 = 8.96$ 8
$0.96 \times 16 = 15.36$ F
$0.36 \times 16 = 5.76$ 5
$0.76 \times 16 = 12.16$ C

$(0.785)_{10} = (0.C85FC)_{16}$

$\therefore \boxed{(1954.785)_{10} = (7A2.C8F5C)_{16}}$

(ii) $(3283.715)_{10} = (\ ?\ )_{16}$

Sol:- Integer part conversion

16 | 3283		
16 | 205	- 3	↑
16 | 12	- (13) D	↑
0	- (12) C	↑

$(3283)_{10} = (CD3)_{16}$

Fractional part conversion

$0.715 \times 16 = 11.44$	B
$0.44 \times 16 = 7.04$	7
$0.04 \times 16 = 0.64$	0
$0.64 \times 16 = 10.24$	A

$(0.715)_{10} = (0.B70A)_{16}$

$\therefore \boxed{(3283.715)_{10} = (CD3.B70A)_{16}}$

9. Base 3 to Base 5 :-

Ex:- (i) $(21.1)_3 = (\ ?\ )_5$

Sol:- Base 3 ⟶ decimal ⟶ Base 5

$2\times3^1 + 1\times3^0 + 1\times3^{-1} = 7.333$

$\therefore (21.1)_3 = (7.333)_{10}$

$(7.333)_{10} = (\ ?\ )_5$

5 | 7		
5 | 1	- 2	↑
0	- 1	↑

$(7)_{10} = (12)_5$

$0.333\times5 = 1.665$
$1.665\times5 = 3.325$
$0.325\times5 = 1.625$
$0.625\times5 = 3.125$
$0.125\times5 = 0.725$
$0.725\times5 = 3.625$

$(0.333)_{10} = (0.131\overline{30})_5$

$\therefore \boxed{(7.333)_{10} = (12.131\overline{30})_5 = (21.1)_3}$

10. Octal to Binary conversion :-

Ex:- (i) $(73.2)_8 = (\ ?\ )_2$

Sol:-

7	3	.	2
↓	↓		↓
111	011		010

$\therefore \boxed{(73.2)_8 = (111011.010)_2}$

(ii) $(23.46)_8 = (\ ?\ )_2$

Sol:-

2	3	.	4	6
↓	↓		↓	↓
010	011		100	110

$\therefore \boxed{(23.46)_8 = (10011.10011)_2}$

11. Hexadecimal to Binary conversion:-

Ex:- (i) $(7A.2C)_{16} = (\ ?\ )_2$

Sol:-

7	A	.	2	C
↓	↓		↓	↓
0111	1010		0010	1100

$\boxed{(7A.2C)_{16} = (1111010.00101100)_2}$

(ii) $(D2.E9)_{16} = (\ ?\ )_2$

Sol:-

D	2	.	E	9
↓	↓		↓	↓
1101	0010		1110	1001

$\boxed{(D2.E9)_{16} = (11010010.11101001)_2}$

12. Binary to octal conversion:-

16 8 4 2 1

Ex:- (i) $(10111.101)_2 = (\ ?\ )_8$

Sol:- $\underset{\leftarrow 2}{0}\underset{\leftarrow 1}{1}\underset{\leftarrow 7}{0}\underset{\leftarrow 1}{1}\underset{\leftarrow 1}{1}.\underset{\rightarrow 5}{1}\underset{\rightarrow}{0}\underset{\rightarrow}{1} = (27.5)_8$

$\therefore \boxed{(10111.101)_2 = (27.5)_8}$

(ii) $(1011.1)_2 = (\ ?\ )_8$

Sol:- $\underset{\leftarrow 1}{0}\underset{\leftarrow 3}{0}\underset{\leftarrow}{1}\underset{\leftarrow}{1}.\underset{\rightarrow 4}{1}\underset{\rightarrow}{0}\underset{\rightarrow}{0} = (13.4)_8$

$\therefore \boxed{(001011.100)_2 = (13.4)_8}$

13. Binary to Hexadecimal :-

Ex:- (i) $(01110111.1010)_2 = (\ ?\ )_{16}$

Sol:- $\underset{7}{\overleftarrow{0111}}\ \underset{7}{\overleftarrow{0111}} \cdot \underset{A}{\overrightarrow{1010}} = (77.A)_{16}$

$\therefore \boxed{(01110111.1010)_2 = (77.A)_{16}}$

(ii) $(10111.1000)_2 = (\ ?\ )_{16}$

Sol:- $\underset{1}{\overleftarrow{0001}}\ \underset{7}{\overleftarrow{0111}} \cdot \underset{8}{\overrightarrow{1000}} = (17.8)_{16}$

$\therefore \boxed{(10111.1)_2 = (17.8)_{16}}$

14. Octal to Hexadecimal :-

Ex:- (i) $(35.7)_8 = (\ ?\ )_{16}$

Sol:- Octal ⟶ Binary ⟶ Hexadecimal.

3 5 . 7
↓ ↓ ↓
011 101 111

$(35.7)_8 = (\underset{1}{\overleftarrow{011}}\ \underset{D}{\overleftarrow{101}} \cdot \underset{E}{\overrightarrow{111}})_2$

$\therefore \boxed{(35.7)_8 = (1D.E)_{16}}$

(ii) $(73.2)_8 = (\ ?\ )_{16}$

Sol:- 7 3 . 2
111 011 010

$(73.2)_8 = (\underset{3}{\overleftarrow{111}}\ \underset{B}{\overleftarrow{011}} \cdot \underset{4}{\overrightarrow{010}})_2$

$\therefore \boxed{(73.2)_8 = (3B.4)_{16}}$

15. Hexadecimal to octal :-

Hexadecimal → Binary → octal

Ex:- ① $(1D.E)_{16} = (\ ?\ )_8$.

Sol:-

1	D	.	E
0001	1101		1110

$\therefore (1D.E)_{16} = (\underset{3}{0}\underset{5}{1}\underset{7}{1}01.\underset{7}{111})_2$

$= (35.7)_8$

$\therefore \boxed{(1D.E)_{16} = (35.7)_8}$

② $(3B.4)_{16} = (\ ?\ )_8$

Sol:-

3	B	.	4
0011	1011		0100

$= (\underset{7}{111}\underset{3}{011}.\underset{2}{010})_2$

$\therefore \boxed{(3B.4)_{16} = (73.2)_8}$

16/07/2014

* Binary addition

Rules

	Sum	Carry	Result
0+0	0	0	0
0+1	1	0	1
1+0	1	0	1
1+1	0	1	10

1 → +1 → 10 → +1 → 11 → +1 → 100 → +1 → 101

Ex:- (i) perform $1011_2 + 1111_2$

Sol:-

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 0\ 1\ 0 \end{array}$$

Carry Sum

$1011_2 + 1111_2 = 11010_2$

11 + 15 = 26

(ii) perform $1011_2 + 1101_2 + 1110_2$

Sol:
$$\begin{array}{r} 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,0\,0\,1\,1\,0 \end{array}$$
carry, Sum

$\therefore$ $1011_2 + 1101_2 + 1110_2 = 100110_2$

(iii) $1001.1_2 + 1101.01_2$

Sol:
$$\begin{array}{r} 1\,0\,0\,1\cdot 1\,0 \\ 1\,1\,0\,1\cdot 0\,1 \\ \hline 1\,0\,1\ 1\,0\cdot 1\,1 \end{array}$$
carry, Sum

$\therefore$ $1001.1_2 + 1101.01_2 = 10110.11_2$

* Binary multiplication :-

Ex:- (i) $101_2 \times 111_2$

Sol:-
$$\begin{array}{r} 101 \\ \times 111 \\ \hline 101 \\ 101 \\ 101 \\ \hline 100011 \end{array}$$

$\therefore$ $101_2 \times 111_2 = 100011_2$

(ii) $1010.1_2 \times 1011.1_2$

Sol:-
$$\begin{array}{r} 1010.1 \times 1011.1 \\ \hline 10101 \\ 10101 \\ 10101 \\ 00000 \\ 10101 \\ \hline 1111000.11 \end{array}$$

$\therefore$ $1010.1_2 \times 1011.1_2 = 1111000.11$

* Binary division :-

Ex:- (i) $111101_2 \div 100_2$

Sol:-
$$\begin{array}{r} 100\overline{)111101}\,(1111 \\ 100 \\ \hline 111 \\ 100 \\ \hline 110 \\ 100 \\ \hline 101 \\ 100 \\ \hline 1 \end{array}$$
Q R

$\therefore$ $111101_2 \div 100_2 = 1111.1$

Ex:- (ii) $1010.1 \div 101.01$

Sol:-
$$\frac{1010.1}{101.01} = \frac{10101/10}{10101/100}$$

$$= \frac{101010}{10101}$$

$$= \frac{10}{1}$$

* Binary subtraction & complements :-

Note:-

$A - B = A +$ r's complement of B (Subtraction using radix complement)

$A - B = A +$ (r-1)'s complement of B + 1 (Subtraction using diminished radix complement)

r → radix (or) Base of A & B no.'s

If A & B are Binary no.'s then

$A - B = A +$ 1's comp. of B + 1

$= A +$ 2's comp. of B.

where
A → Minuend
B → Subtrahend.

1's complement:-

1's complement of any number can be obtained by changing each '0' to '1' and '1' to '0'.

∴ 1's complement of 1101001 is 0010110.

1's complement subtraction:-

Rules:-

(i) Add minuend and 1's complement of subtrahend.

(ii) If there is carry after addition, then the result is '+ve'. else result is '+ve'. else result is '-ve'

(iii) If result is '+ve', add carry to sum to get the actual value.

(iv) If result is '-ve' take 1's complement of sum to get the actual value.

Ex:- (1) Perform unsigned binary subtraction using 1's complement.

(i) $1101_2 - 1001_2$

Sol:-

1101 → minuend
1001 → subtrahend
0110 → 1's complement of subtrahend.

1101 → minuend
0110 → i's comp. of Sub
10011
carry Sum

∴ there is a carry. ∴ Result is 'tve'.

To get the actual value (difference) add carry to Sum.

0011
+ 1
0100 → difference.

∴ $1101_2 - 1001_2 = 0100_2$

(ii) $1001_2 - 1101_2$

Sol:- 1001 → minuend
1101 → Subtrahend.

1001 → minuend
0010 → i's complement of Subtrahend
1011
Sum.

there is no carry. ∴ Result is '-ve'.

To get the actual value take i's complement of Sum.

i's complement of 1011 is 0100

∴ $1001_2 - 1101_2 = -0100_2$

2's complement :-

2's complement of A = 1's complement of A + 1

Ex:- Find 2's complement of 1010.

Sol:- = 1's complement of 1010 + 1
= 0101 + 1
= 0110

0101
+ 1
0110

∴ 2's complement of 1010 = 0110

2's comp of 101100 is 010100 (write no. as it is from right to left until u found 1. if u found 1 then write that first 1 as it is. then change the remaining no.s from 0's to 1's and 1's to 0's)

2's complement subtraction rules :-

(i) Add minuend and 2's complement of subtrahend.

(ii) After addition if there is carry then the result is '+ve', else '-ve'.

(iii) If the result is '+ve' to get the actual value discard the carry.

(iv) If the result is '-ve' to get the actual values take 2's complement of sum.

Ex:- 1. Perform unsign binary subtraction using 2's complement.

& (i) $1101_2 - 1010_2$

Sol:- 1101 → minuend
1010 → subtrahend.

0101
+ 1
0110

2's comp. of subtrahend (1010) = 0101 + 1 = 0110.

1101 → minuend
0110 → 2's comp. of sub.
10011
carry sum

∴ It has carry. ∴ Result is '+ve'.

∴ To get actual value discard the carry.

$= 0011_2$

∴ $\boxed{1101_2 - 1010_2 = 0011_2}$

(ii) $1010_2 - 1101_2$

Sol:- 1010 → min.
1101 → sub.

0011 → 2's comp. of sub.
1010 → min
1101
Sum.

∴ It has no carry. ∴ Result is '-ve'.

To get actual value take 2's comp. of sum.

2's comp. of $1101 = 0010 + 1 = 0011_2$

∴ $\boxed{1010_2 - 1101_2 = -0011_2}$

9's complement :-

9's complement of a decimal number can be obtained by the subtracting each digit from 9.

Ex:- 9's comp. of 73_{10} is

$$\begin{array}{r} 99 \\ -73 \\ \hline 26 \end{array}$$

∴ 9's comp. of 73_{10} is 26

9's complement subtraction :-

Rules same as is complement but instead of is complement take 9's complement.

Ex:- (i) $73_{10} - 26_{10}$

Sol:- 73 → min

26 → sub.

73 → min.

73 → 9's comp. of sub.

$$\begin{array}{r} 99 \\ -26 \\ \hline 73 \end{array}$$

$$\begin{array}{r} 146 \\ \text{carry sum} \end{array}$$

∴ It has carry. ∴ Result is '+ve'.

To get actual value add carry to sum.

$$\begin{array}{r} 46 \\ 1 \\ \hline 47_{10} \end{array}$$

∴ ~~26_{10}~~ ⇒ ∴ $73_{10} - 26_{10} = 47_{10}$

(ii) $26_{10} - 73_{10}$

Sol:- 26 → min.

73 → sub.

$$\begin{array}{r} 99 \\ 73 \\ \hline 26 \end{array}$$ → 9's comp of sub.

$$\begin{array}{r} 26 \\ 26 \\ \hline 53 \end{array}$$

It has no carry. Result is '-ve'.

To get actual values Take 9's comp. of sum.

$$\begin{array}{r} 99 \\ -52 \\ \hline 47 \end{array} \longrightarrow 9\text{'s comp. of } 52.$$

$\therefore$ $\boxed{26_{10} - 73_{10} = -47_{10}}$

10's complement :-

10's comp. of A = 9's comp. of A + 1

Ex:- Find 10's comp. of 23_{10}

Sol:- 10's comp. of 23 = 9's comp. of 23 + 1

= 76 + 1

= 77

$$\begin{array}{r} 99 \\ -23 \\ \hline 76 \end{array}$$

$\therefore$ $\boxed{\text{10's comp. of } 23 = 77}$

10's complement Subtraction :-

Rules Same as 2's Subtraction complement but take 10's complement instead of 2's comp.

Ex:- $45_{10} - 16_{10}$

Sol:- 45 → min.

16 → Sub.

10's comp. of 16 = 9's comp of 16 + 1

= 83 + 1 = 84

$$\begin{array}{r} 99 \\ -16 \\ \hline 83 \end{array}$$

$\therefore$ 84 → 10's comp. of Sub.

45 → min.

$$\begin{array}{r} 129 \\ \hline \end{array}$$

carry · Sum

It has Carry. $\therefore$ Result is '+ve'.

To get actual value discard the carry.

$\therefore$ $\boxed{45_{10} - 16_{10} = 29_{10}}$

Ex:- (i) $16_{10} - 45_{10}$.

Sol:- 16 → min.
45 → sub.

10's comp. of 45 = 9's comp. of 45 + 1
= 54 + 1 = 55.

$\begin{array}{r} 99 \\ -45 \\ \hline 54 \end{array}$

55 → 10's comp. of 45
16 → min

$\begin{array}{r} 71 \\ \hline \end{array}$
Sum

It has no carry. ∴ Result is '-ve'.

To get actual value take 10's comp. of Sum.

10's comp. of 71 is = 9's comp. of 71 + 1
= 28 + 1 = 29

$\begin{array}{r} 99 \\ -71 \\ \hline 28 \end{array}$

∴ $\boxed{16_{10} - 45_{10} = -29_{10}}$

Octal Subtraction :-

If A & B are octal no.'s then

A − B = A + 8's comp. of B
= A + 7's comp. of B + 1

7's comp. of 23_8 is

$\begin{array}{r} 77 \\ -23 \\ \hline 54 \end{array}$

Hexadecimal Subtraction:-

If A & B are octal no.'s then,

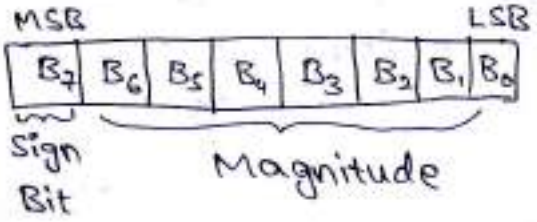
A − B = A + 16's comp. of B
= A + 15's comp. of B + 1

15's comp. of $A2_{16}$ is

$\begin{array}{r} FF \\ -A2 \\ \hline 5D \end{array}$

17/07/2014

Signed Binary numbers :-



Sign bit = 0 ⟶ +ve binary number

Sign bit = 1 ⟶ -ve binary number.

Ex:- Represent -5_{10}, -9_{10}, $+17_{10}$ in signed magnitude form.

Sol:- $5 \longrightarrow 101$

$-5_{10} = (1\,101)_{\text{Signed binary}}$ (∵ add 1 in MSB place)

(Sign bit, Magn.)

$9 \longrightarrow 1001$

$-9_{10} = (1\,1001)_{\text{signed binary}}$

$+17_{10} = 0\,10001$

$+9_{10} = (0\,1001)_{\text{signed binary}}$.

Ex:- Represent -5_{10}, -9_{10}, $+17_{10}$ in 8 bit signed magnitude form.

Sol:- $-5_{10} =$ | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

$-5_{10} = (10000101)_{\text{Signed Binary (2)}}$

$-9_{10} = (1\,0001\,001)_{\text{signed binary}}$

$+17_{10} = (0\,0010001)_{\text{Signed binary}}$.

Ex:- Represent +7, -7 in 8bit signed binary, signed 1's complement, signed 2's complement form.

Sol:- (i) +7 = 0111 (signed binary representation)

+7 = 0111 (signed 1's complement ")

+7 = 0111 (signed 2's complement ").

+7 = 00000111 (8-bit signed binary ")

+7 = 00000111 (signed 1's comp. ")

+7 = 00000111 (signed 2's " ").

(ii) -7 = 1111 (signed magnitude Representation)

-7 = 1000 (signed 1's comp. ") (+7 is complement)

-7 = 1001 (" 2's " ") (+7 2's comple = is comp + 1.

-7 = 10000111 (8-Bit Signed magnitude rep)

-7 = 11111000 (signed 1's comp. rep)

-7 = 11111001 (" 2's comp. ")

Ex:- In signed 1's complement '-7' is obtained by complementing all the bits of +7, including the sign bit.

In signed 2's complement '-7' is obtained by taking 2's complement of +7.

* Arthametic Addition :-

Ex:- perform the following using 8-Bit Binary arthametic.

(i) (+6) + (+13) (ii) (-6) + (+13) (iii) (+6) + (-13)

(iv) (-6) + (-13).

(i) sol:- +6 → 00000110

+13 → 00001101

```
  00000110
  00001101
  --------
  00010011
```

Signed bit is '0'.

∴ Result is 'tve' number.

(+6) + (+13) = +19

(ii) sol:- +6 → 00000110

(-6 → 10000110)

Represent -ve' no's in signed 2's comp. form

-6 = (2's comp. of +6)

-6 = 11111010

+13 = 00001101

```
  11111
  11111010
  00001101
  --------
 100000111
```

Carry. 7

Sign bit = 0 , ∴ Result is 'tve' no.

If carry is generated, discard that carry.

(-6) + (+13) = +7

(iii) $+6 \longrightarrow 00000110$

$-13 \longrightarrow 11110011$ (2's comp. of +13)

$$\begin{array}{r} 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\ \hline 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1 \end{array}$$

Sign bit

Sign bit = 1, ∴ Result is '-ve' no.

'-ve' results are already in 2's comp. form.

To get result take 2's comp. of Sum including Sign bit.

2's comp. of 11111001 is $\underbrace{00000111}_{7}$.

∴ $\boxed{(+6) + (-13) = -7}$

(iv) $-6 \longrightarrow 11111010$

$-13 \longrightarrow 11110011$

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0 \quad (2\text{'s comp of } -6) \\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \quad (2\text{'s comp of } -13) \\ \hline 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1 \end{array}$$

carry Sign bit

carry is discard

Sign bit = 1 ∴ Result is '-ve' no.

'-ve' results are already in 2's comp. form.

To get result take 2's comp. of Sum including Sign bit.

2's comp. of 11101101 is $\underbrace{00010011}_{19}$

∴ $\boxed{(-6) + (-13) = -19}$

Binary codes

- Weighted codes
  - Binary
  - BCD codes
    - 8421
    - 2421
    - 3321
    - 4221
    - 5211
    - 5311
    - 5421
    - 6311
    - 7421
    - 74$\bar{2}\bar{1}$ (or) 74-2-1
    - 84$\bar{2}\bar{1}$ (or) 84-2-1
- non weighted codes
  1. Excess-3
  2. Gray
  3. 5-bit BCD codes
- Reflective codes
  1. 2421
  2. 5211
  3. Excess-3
- Sequential codes
  1. 8421
  2. Excess-3
- Alphanumeric codes
  1. ASCII
  2. EBCDIC
  3. Hollerith
- Error detecting & correcting codes
  1. Parity
  2. Hamming code

Binary codes ..

Weighted codes :-

In weighted codes each digit position of the number representing a specific weight.

1. Straight Binary code

$\ldots\ldots 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

MSB — LSB

↳ depends upon no. of digits.

2. Binary coded decimal (BCD)

BCD is a numeric code in which each digit of a decimal number is represented by a seperate group of bits.

The most common BCD code is 8 4 2 1

Decimal	BCD codes		
	8421	4221	7421 (or) 7 4 -2 1
0	0000	0000	0000
1	0001	0001	0111
2	0010	0100 (or) 0010	0110
3	0011	0011 (or) 0101	0101
4	0100	1000 (or) 0110	1011
5	0101	1001 (or) 0111	1010
6	0110	1100 (or) 1010	1001
7	0111	1101 (or) 1011	1000
8	1000	1110	1100
9	1001	1111	1110

Ex:- Represent 65_{10} in straight binary & BCD code

Sol:- S.B $65_{10} = (1000001)_2$

2	65
2	32 - 1
2	16 - 0
2	8 - 0
2	4 - 0
2	2 - 0
2	1 - 0
	0 - 1

BCD code

6 5

0110 0101

$65_{10} = (01100101)_{BCD}$ (8421)

Ex:- ① convert $2A_{16}$ to BCD code.

Sol:- Hexa decimal ⟶ decimal ⟶ BCD.

$= 2\times16^1 + A\times16^0$

$= 32 + 10 = 42$

$(2A)_{16} = (42)_{10}$

4 2

0100 0010

$(2A)_{16} = (42)_{10} = (01000010)_{BCD\ 8421}$

② convert $(01000010)_{BCD} = (\ ?\ )_{16}$

BCD → decimal → Hexadeci

Sol:- 01000010

4 2

$(01000010)_{BCD} = (42)_{10}$

16	42
16	2 - (10)A
	0 - 2

$(42)_{10} = (2A)_{16}$

$\therefore (01000010)_{BCD} = (42)_{10} = (2A)_{16}$

22/07/2019

BCD Addition rules :-

(i) Add two BCD numbers using Binary addition rules

(ii) If a 4 bit sum is greater than 9 or if a carry is generated from a 4 bit sum, the sum is Invalid

(iii) Add 6 to the 4 bit sum inorder to skip the invalid states.

Ex: perform the following using BCD arthametic.

(i) $1273_{10} + 9587_{10}$

Sol:- $1273_{10} \longrightarrow \overbrace{0001}^{1}\overbrace{0010}^{2}\overbrace{0111}^{7}\overbrace{0011}^{3}_{BCD}$

$9587_{10} \longrightarrow \overbrace{1001}^{9}\overbrace{0101}^{5}\overbrace{1000}^{8}\overbrace{0111}^{7}_{BCD}$

```
  0001 0010 0111 0011
+ 1001 0101 1000 0111
  ---------------------
  1010 1111 1111 1010
```

1010 1111 1111 1010

0110 0110 0110

10000 1000 0110 0000

1 0 8 6 0

} in the first level of addition add 0110 (6) to the groups which are >9 (1001) (or) carry generated out of 4-bit group

From second level onwards add 6 to the groups which are >9

$1273_{10} + 9587_{10} = 10860_{10}$

(ii) $999_{10} + 989_{10}$

Sol:- $999_{10} \longrightarrow 1001\ 1001\ 1001_{BCD}$

$989_{10} \longrightarrow 1001\ 1000\ 1001_{BCD}$

```
  1001 1001 1001
  1001 1000 1001
  ---------------
  10011 0010 0010
carry
  0110 0110 0110
  ---------------
  11001 1000 1000
  ---------------
```

1 9 8 8

→ In the first level of addition carry is generated so, add 6 (0110) to all 4-bit groups.

$999 + 989 = 1988_{10}$

(iii) $7762_{10} + 3838_{10}$

Sol:- $7762_{10} \longrightarrow$ 0111 0111 0110 0010 BCD

$3838_{10} \longrightarrow$ 0011 1000 0011 1000 BCD.

```
 0111   0111   0110   0010
 0011   1000   0011   1000
 ----   ----   ----   ----
 1010   1111   1001   1010
 0110   0110          0110
 ----   ----   ----   ----
10001   0101   1000   0000
                0110
                ----
10001   0110   0000   0000
  1      1      6      0      0
```

In the first level of addition add 0110 to the groups which are > 1001 (or) carry generated out of 4-bit group

from 2nd level onwards add 6 to the groups which are > 9 only

$7762_{10} + 3838_{10} = 11600_{10}$

* Non-weighted (or) unweighted codes :-

In weighted codes are not assigned with any weight to each digit position.

Ex:- EXCESS-3, GRAY codes.

EXCESS-3 code :- It can be derived from the natural BCD code by adding '3' to each coded number.

Decimal	BCD 8421	Excess-3 (BCD + 0011)
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Ex:- Find EXCESS-3 code for 592_{10}.

Sol:-

5 9 2_{10}

1000 1100 0101 → EXCESS-3 form.

$592_{10} = 100011000101_{\text{EXCESS-3}}$.

→ Binary to Gray code conversion :-

rules :-

B_1 B_2 B_3 B_4 ------ Binary no.

G_1 G_2 G_3 G_4 ------ Gray code no.

$G_1 = B_1$

$G_2 = B_1 \oplus B_2 = B_1\bar{B}_2 + \bar{B}_1 B_2$

$G_3 = B_2 \oplus B_3$

$G_4 = B_3 \oplus B_4$

Ex:- $(1100)_2 = (\ ?\ )_{\text{Gray}}$

Sol:- MSB is same.

Remaining digits are replaced with opposite digits ~~1's~~ → 0's

~~0's~~ → 1's.

Remaining are as below table.

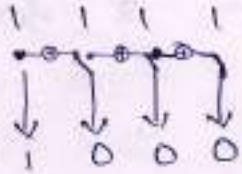
EXOR (or) XOR (or) Exclusive or gate.

$C = \bar{A}B + A\bar{B} = A \oplus B$.

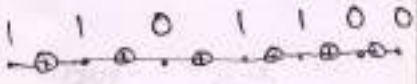
A	B	C = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Ex:- (i) $(1111)_2 = (\ ?\ )_{\text{Gray}}$

Sol:-



$(1111)_2 = (1000)_{\text{Gray}}$.



(ii) $(101010)_2 = (\ ?\ )_{\text{Gray}}$.

Sol:- $(101010)_2 = (111111)_{\text{Gray}}$.

Gray to Binary :-

rules :-

G_1 G_2 G_3 ---- Gray code no.

B_1 B_2 B_3 ---- Binary code no.

$B_1 = G_1$

$B_2 = B_1 \oplus G_2$

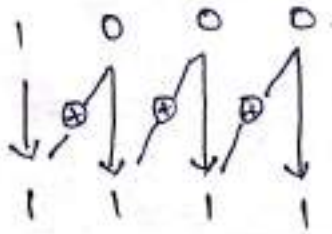
$B_3 = B_2 \oplus G_3$

$B_4 = B_3 \oplus B_4$

:

Ex:- (i) $(1000)_{Gray} = (\ ?\ )_2$

Sol:-



$(1000)_{Gray} = (1111)_2$

Reflective codes :-

A code is said to be reflective when the code for '9' is the complement for the code for '0', '8' for '1', '7' for '2', '6' for '3', '5' for '4'.

Ex:- 2421, 5211, EXCESS-3.

Sequential codes :-

In sequential codes each succeeding code is one binary no. greater than it's preceding code.

Alphanumeric codes :-

The codes which represent the alphabetic characters, numbers are called alpha numeric codes.

Ex:- ASCII. (American standard code for Information Interchange)

EBCDIC (Extended Binary coded Decimal Interchange code)

→ **Binary to Gray code :-**

unit distance code is Gray code.

decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

2. BOOLEAN ALGEBRA (or) SWITCHING ALGEBRA

Boolean algebra is used to simplify boolean equations, to make simple logic circuits.

Representation of Boolean function (or) switching function (or) logical function :-

Ex:- $y = ab + bc$

→ a, b, c are called variables (or) literals. which are inputs for the logic gates.

→ To implement a logic function with less no. of gates we have to minimise variables & the number of terms.

Boolean algebra rules :-

(i) commutative law

$$A + B = B + A$$

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

(ii) Associative law

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

$$(AB)C = A(BC)$$

(iii) Distributive law

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

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

(iv) Additive law

$$A + 0 = A$$

$$A + A = A$$

$$A + 1 = 1$$

$$A + \bar{A} = 1$$

(v) multiplication property.

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

$A \cdot \bar{A} = 0$

(vi) Double negation.

$\overline{(\bar{A})} = A$

(vii) Absorption law

$A(A+B) = A$

$A + AB = A$

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

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

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

(viii) Duality property. $AB + \bar{A}\bar{B} = (A+B)(\bar{A}+\bar{B})$

(ix) Demorgan's theorem. $\overline{A \cdot B} = \bar{A} + \bar{B}$

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

(x) consensus theorem. $AB + \bar{A}C + BC = AB + \bar{A}C$.

① Prove that $(A+B)(A+C) = A + BC$.

Proof:- L.H.S $= (A+B)(A+C)$

$= AA + AC + BA + BC$ $(\because AA = A)$

$= A + AC + BA + BC$

$= BC + (1 + C + B)A$ $(\because 1 + A = 1)$

$= A + BC$ (1 + Anything = 1 in logical gates).

$= R.H.S.$

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

Ex:- Simplify following boolean expressions to a min. no. of variables or literals. ($\bar{B}$ (or) B' both are same)

① $(A+B)(A+\bar{B})$

Sol:- $(A+B)(A+\bar{B}) = A\cdot A + A\cdot\bar{B} + BA + B\cdot\bar{B}$

$= A + A\cdot\bar{B} + B\cdot A + 0$

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

$= A(1+1)$ ($\because$ 1 + anything = 1)

$= A$

② $xy + xyz + xy\bar{z} + \bar{x}yz$

Sol:- $xy + xyz + xy\bar{z} + \bar{x}yz$

$= xy[1 + z + \bar{z}] + \bar{x}yz$

$= xy.1 + \bar{x}yz$ ($\because$ 1 + anything = 1)

$= \bar{x}yz + xy$ ($\because$ $A+BC = (A+B)(A+C)$)

$= y[\bar{x}z + x] = y[(x+\bar{x}) + (x+z)]$

$= y[x+z]$

23/07/2014

③ $x'y + xy' + xy + x'y'$

Sol:- $x'[y+y'] + x[y+y']$

$= x'[1] + x[1] = x + x' = 1$

④ $\bar{A} + AB + A\bar{C} + A\bar{B}\bar{C}$

Sol:- $\bar{A} + AB + A\bar{C}(A + A\bar{B})$

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

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

$= 1 - A + AB + A(1 - C)$

$= 1 - A + AB + A - AC$

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

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

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

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

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

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

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

Sol:- $ABC + AB\bar{C} + \bar{A}B$

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

$= AB + \bar{A}B$

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

(6) $x'y' + xy + x'y$

Sol:- $x'(y'+y) + xy$

$= x' + xy$

$= (x'+x)(x'+y)$ (:: distributive law)

$= x' + y$

(7) $a + a'b + a'b'c + a'b'c'd + \cdots\cdots$

Sol:- $a + a'[b + b'c + b'c'd + \cdots\cdots]$ (absorption law) (:: $A + \bar{A}B = A + B$)

$= a + b + b'c + b'c'd + \cdots$

$= a + b + b'[c + c'd + c'd'e + \cdots]$ (:: absorption law $A + \bar{A}B = A + B$)

$= a + b + c + c'd + c'd'e + \cdots\cdots$

$\vdots$

$= a + b + c + d + e + \cdots\cdots$

(8) prove that $A(\bar{A}+C)(\bar{A}B+\bar{C}) = 0$

Sol:- $(A\cdot\bar{A} + AC)(\bar{A}B + \bar{C})$

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

$= AC\cdot\bar{A}B + A(C\cdot\bar{C})$ (:: $A\cdot\bar{A} = 0$)

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

$= 0 + 0 = 0$

⑨ P.T $(A+C)(A+D)(B+C)(B+D) = AB+CD$

Sol:- LHS $= [A\cdot A + AD + CA + CD][B\cdot B + B\cdot D + C\cdot B + C\cdot D]$

$= [A + AD + CA + CD][B + BD + CB + CD]$

$(A+C)(A+D)(B+C)(B+C)$ $\left[\because A+BC = (A+B)(A+C)\right]$

$= (A+CD)(B+CD)$ $\quad \because A$

$= CD + AB$

⑩ P.T $AB + \bar{A}\bar{C} + A\bar{B}C(AB+C) = 1$

Sol:- $AB + \bar{A} + \bar{C} + \underbrace{A\bar{B}C\cdot AB}_{0} + A\bar{B}C\cdot C$

$(\because B\cdot\bar{B} = 0)$

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

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

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

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

$= A[(B+\bar{B})(B+C)] + \bar{A}\bar{C}$ $\quad (\because B+\bar{B} = 1)$

$= AB + \underbrace{AC + \bar{A}\bar{C}}_{1}$

$= 1 + AB$ $\quad (\because 1 + \text{anything} = 1)$

$= 1$

⑪ Prove that $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C} = \bar{C}$

Sol:- $\bar{A}\bar{C}[B+\bar{B}] + A\bar{C}[B+\bar{B}]$

$(\because A+\bar{A} = 1)$

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

$= \bar{C}[A+\bar{A}] = \bar{C}$

Ex:- Reduce the following Boolean expression to indicated no. of variables or literals.

(12) $\bar{A}\bar{C} + ABC + A\bar{C}$ to 3 literals.

Sol:- $(\bar{A}+A)\bar{C} + ABC$

$= \bar{C} + ABC.$

$= \bar{C} + C.AB$ $\quad (\because A+BC = (A+B)(A+C))$

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

$= \bar{C} + AB$

(13) $(\overline{\bar{x}y + \bar{z}}) + z + xy + wz$ to 3 variables.

Sol:- $(\overline{\bar{x}y}) \cdot \bar{z} + z + xy + wz$ $\quad (\because \overline{A+B} = \bar{A}\cdot\bar{B})$

$x y \bar{z} + z + xy + wz$

$= z + \bar{z}.xy + xy + wz$ $\quad$ (or) $\quad (A+BC = (A+B)(A+C))$

$= (z+\bar{z})(z+xy) + xy + wz$

$= z + xy + xy + wz$

$= z(1+w) + xy(1+1)$

$= z + xy.$

$= xy(1+\bar{z}) + z(1+w)$

$= xy + z$ $\quad (\because 1 + \text{anything} = 1)$

(13)(i) $(\overline{\bar{x}\bar{y} + z}) + z + xy + wz$ to 3 variables.

Sol:- $\overline{\bar{x}\cdot\bar{y}}\cdot\bar{z} + z + xy + wz$

$(\overline{\bar{x}} + \overline{\bar{y}})\bar{z} + z(1+w) + xy$

$= xy\bar{z} + xy$

$= x\bar{z} + y\bar{z} + z + xy.$

$= z + x\bar{z} + y\bar{z} + xy$

$= (z+x)(z+\bar{z}) + y\bar{z} + xy$

$= x + z + xy + x\bar{z}$ $\quad = x(1+y) + (z+y)(z+\bar{z})$

$= x + y + z$ (∵ 1 + anything = 1, $A + \bar{A} = 1$).

(14) $A'B(C'D + D') + B(A + A'CD)$ to 1 literal

Sol:- $A'BC'D + A'BD' + AB + A'BCD$

$= A'BD(C + C') + AB + A'BD'$ (∵ $A + BC = (A+B)(A+C)$, $A + \bar{A} = 1$)

$= A'BD + AB + A'BD'$

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

$= A'B + AB \quad = B(A + A') = \underline{B}$

Ex: Find the compliment of the Boolean function and reducing to a minimum number of variables.

Sol:- (15) $(B\bar{C} + \bar{A}D)(A\bar{B} + C\bar{D})$

Sol: after taking complement

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

$= \overline{(AB\bar{B}\bar{C} + B\bar{C}C\bar{D} + A\bar{A}D\bar{B} + \bar{A}DC\bar{D})}$ (∵ $A \cdot \bar{A} = 0$)

$= \bar{0} = 1$

(16) $x'(y' + z')(x + y + z')$

Sol:- $\overline{x'(y' + z')(x + y + z')}$

$= \overline{(x'y' + x'z')(x + y + z')}$

$= \overline{(x'y'x + x'z'x + x'y'y + x'y'z' + x'z'y + x' \cdot z' \cdot z')}$

$= \overline{x'y'z' + x'z'y + x' \cdot z'}$

$= \overline{x'z'(y + y') + x' \cdot z'} \quad = \overline{x'z' + x'z'}$ (∵ $x(y+z)$)

$= \overline{x'z'(1+1)} \quad = \overline{x'} = (\bar{x})$

$= \overline{\bar{x}\bar{z}} \quad = x$

$= (\bar{x}) + (\bar{z}) = x + z$

(17) Find dual of $\bar{F} = \bar{A}B + Bc + A\bar{c}$

Sol:- Duality of $F = (\bar{A}+B)(B+c)(A+\bar{c})$

Sub. x's → +'s
+'s → x's
0's → 1's
1's → 0's

(18) Find dual of $xy + x'z = 0$

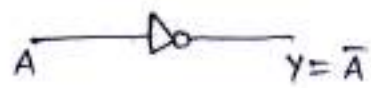
Sol:- Duality $\Rightarrow (x+y)(x'+z) = 1$

LOGIC GATES :-

1. Inverter (or) NOT gate :-

'NOT' Gate performs inversion operation. It has 1 input and one output.

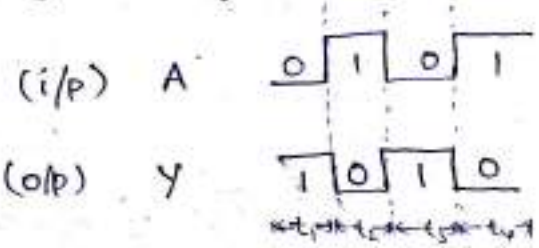
Logic Symbol :-



Truth Table :- Truth table indicates outputs for different possibilities of input.

i/p	o/p
A	$Y = \bar{A}$
0	1
1	0

Timing diagram :- the input & output wave forms showing time relationship is called timing diagram.



2. Buffer gate :-

output is same as the i/p

i/p = o/p.

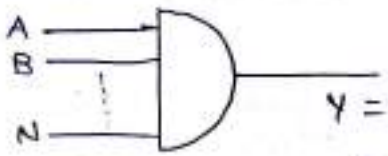
It used for time delay.

3. AND gate :-

'AND' gate performs logical multiplication.

It has two or more i/p's and one output.

Logic symbol :-



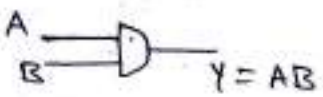
Y = A and B and ..... N

= A · B · ..... N

= AB ..... N.

2 i/p AND gate :-

Logic symbol :-

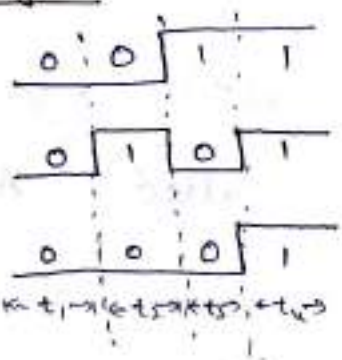


Truth table :-

i/p's		o/p
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

AND gate o/p is high when all i/p's are high.

Timing diagram :-

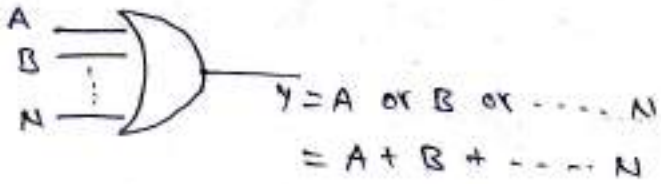


1 i/p AND gate :-

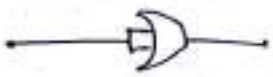
④ OR gate :-

OR gate performs logical Addition. It has two or more inputs & one output.

Logic symbol :-



1 i/p OR gate :-



2 i/p OR gate :-

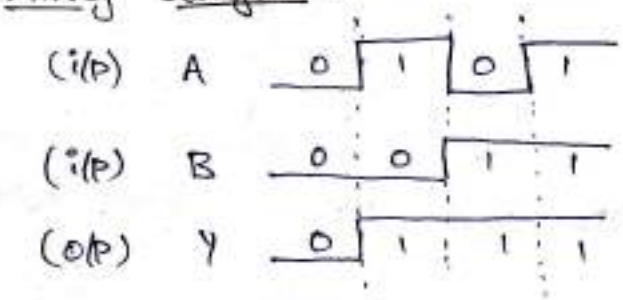
Logic symbol :-

Truth table :-

i/p's		o/p
A	B	Y
0	0	0
1	0	1
0	1	1
1	1	1

OR gate o/p is low when all the inputs are low.

Timing diagram :-



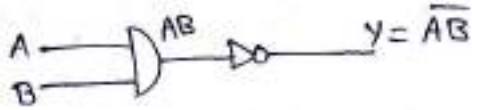
Note:- Logic circuits which use AND, OR, NOT gates only are called AOI logic circuits

Universal gates :-

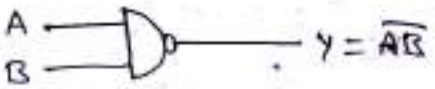
NAND & NOR are called universal gates because by using either NAND or NOR gates we can implement any logic circuit.

NAND gate :- The term NAND is a combination of AND & NOT gates.

Logic Symbol :-



2 i/p NAND gate :-



Truth table :-

i/p's		o/p
A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

NAND gate o/p is low when the all the i/p's are high.

NOR gate :- The term NOR is a combination of OR & NOT gates.

Logic Symbol :-

2 i/p's NOR gate :-

Truth table :-

i/p's		o/p
A	B	y
0	0	1
1	0	0
0	1	0
1	1	0

NOR gate o/p is high when all i/p's are low.

EXOR (or) XOR (or) EXCLUSIVE-OR gate :-

logic symbol :-

$y = A \text{ XOR } B$

$= A \oplus B$

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

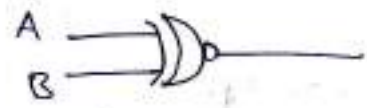
Truth table :-

i/p's		o/p
A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

XOR o/p is low when all i/p's are same.

EXNOR (or) XNOR (or) EXCLUSIVE NOR gate :-

logic symbol :-



$y = A \text{ XNOR } B$

$= A \odot B = \overline{A \oplus B}$

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

Truth table :-

i/p's		o/p
A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

EXNOR o/p is high when all i/p's are same.

Note :- dual of XOR gate is XNOR.

30/07/2014

* Dual of XOR gate is XNOR.

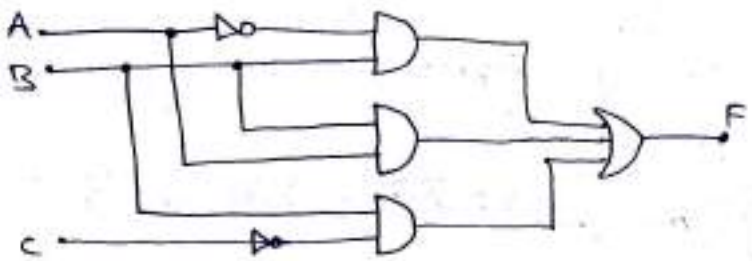
dual of $\bar{A}B + A\bar{B} = (\bar{A}+B)(A+\bar{B})$

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

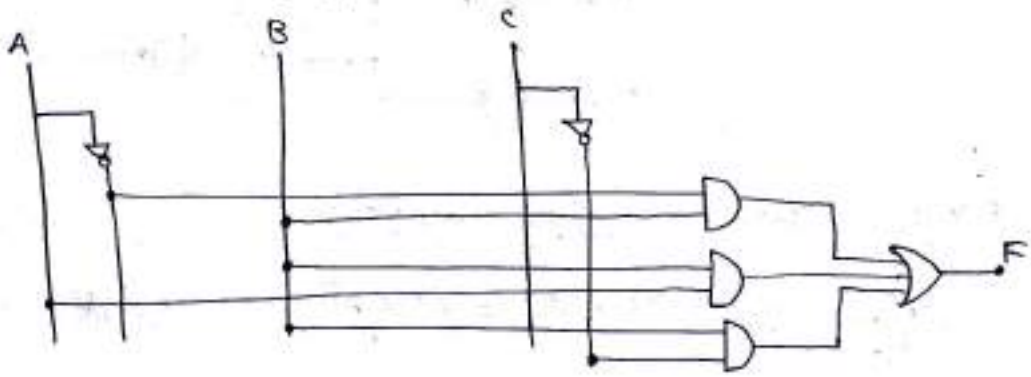
$= \bar{A}\bar{B} + AB$

Ex: Implement $F = \bar{A}B + AB\bar{C} + B\bar{C}$ using basic logic gates.

Sol :-

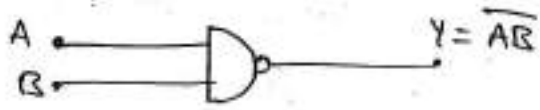


(or)



Ex: Implement NOT, AND, OR, NOR, EXNOR, EXOR gates using NAND gate.

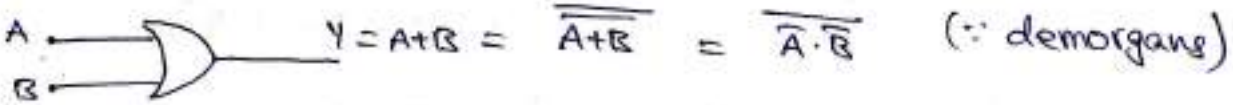
Sol:- NAND gate :-

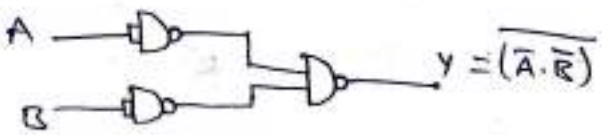


NOT gate using NAND gate :-

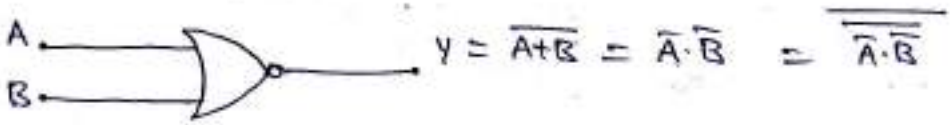
AND gate using NAND :-

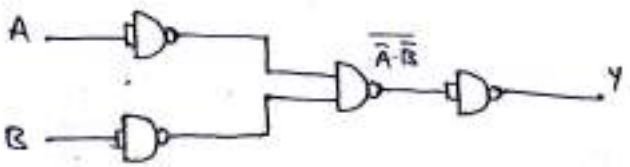
OR gate using NAND :-



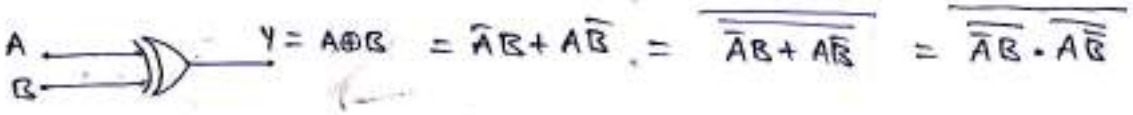


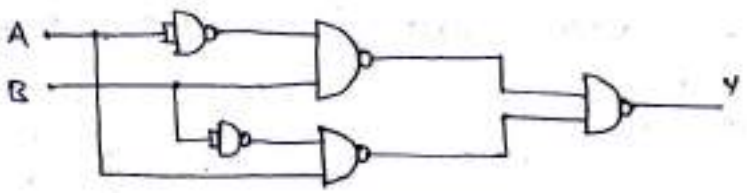
NOR gate using NAND :-





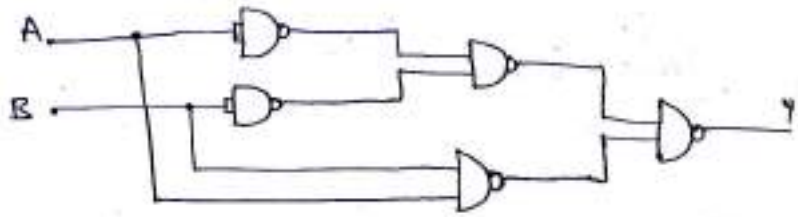
EXOR gate using NAND :-





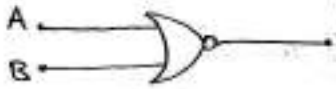
EXNOR gate using NAND :-

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



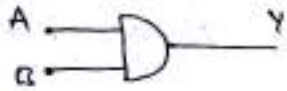
Ex: Implement all logic gates expect NOR gate using NOR gate.

Sol:- NOR gate :-

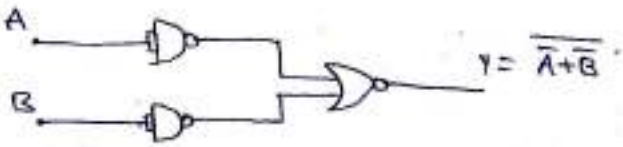


NOT gate using NOR gate :-

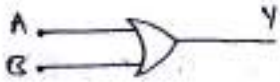
AND gate using NOR gate :-



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



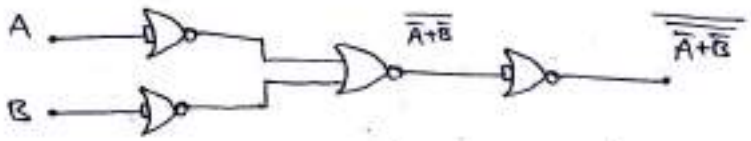
OR gate using NOR gate :-



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

NAND gate using NOR gate :-

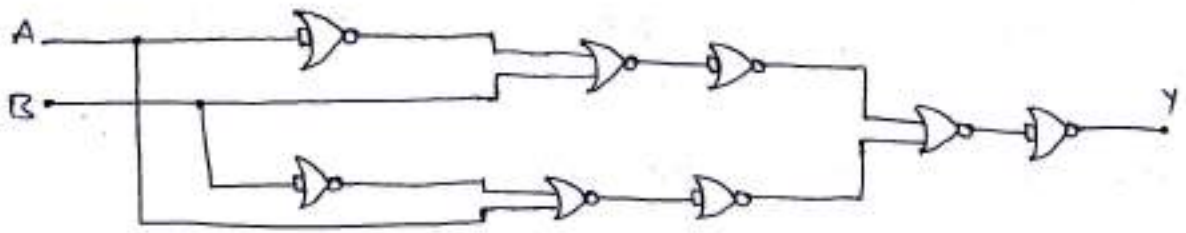
$$\overline{AB} = \bar{A}+\bar{B} = \overline{\overline{A}+\overline{B}}$$



EXOR gate using NOR gate :-

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

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

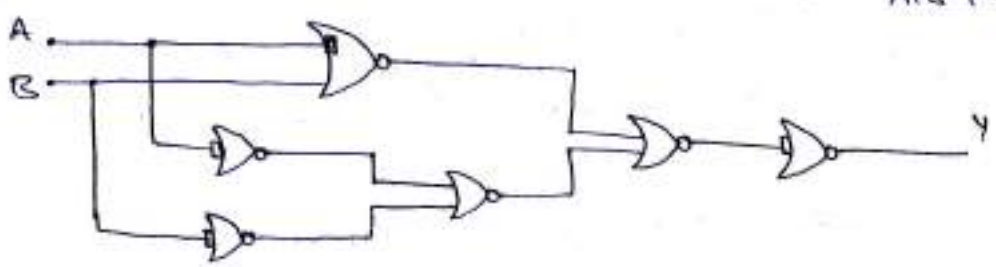


EXNOR gate using NOR gate :-

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

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

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



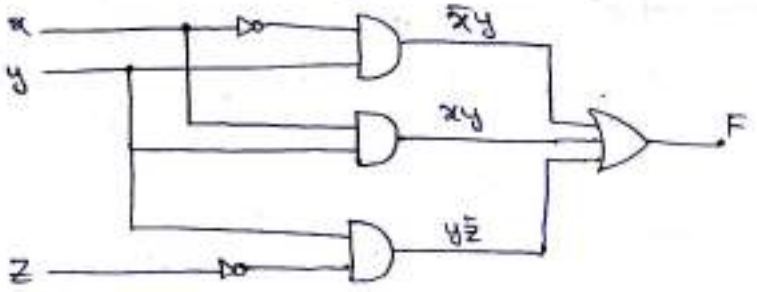
Ex: Implement $F = \bar{x}y + x\bar{y} + y\bar{z}$ using (i) Basic gates

(ii) AND & NOT gates (iii) OR & NOT gates

(iv) only NAND gate (iv) only NOR gate.

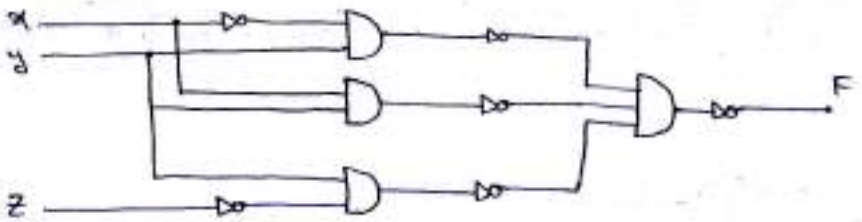
Sol:- $F = \bar{x}y + xy + y\bar{z}$

(i) using Basic gates.



(ii) using AND & NOT gates

$$F = \bar{x}y + xy + y\bar{z} = \overline{\overline{\bar{x}y + xy + y\bar{z}}} = \overline{\overline{\bar{x}y} \cdot \overline{xy} \cdot \overline{y\bar{z}}}$$



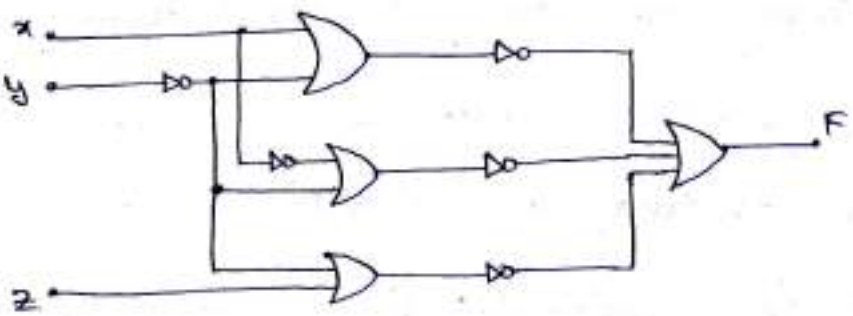
(iii) using OR & NOT gates.

$$F = \bar{x}y + xy + y\bar{z}$$

$$= \overline{\overline{\bar{x}y}} + \overline{\overline{xy}} + \overline{\overline{y\bar{z}}}$$

$$= \overline{\overline{\bar{x}} + \overline{y}} + \overline{\overline{x} + \overline{y}} + \overline{(\overline{y} + \overline{\bar{z}})}$$

$$= \overline{x + \overline{y}} + \overline{\overline{x} + \overline{y}} + \overline{\overline{y} + z}$$

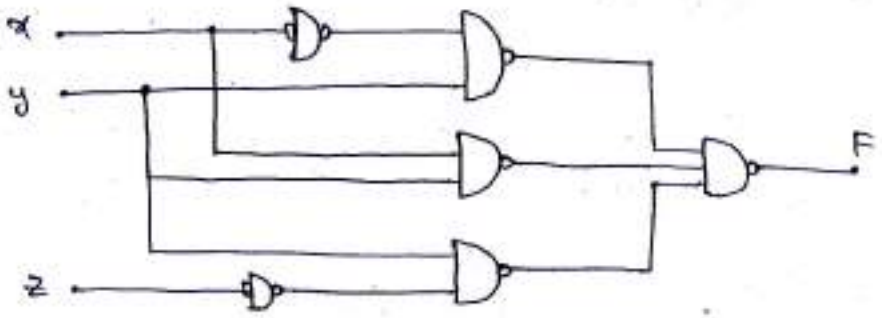


(iv) using NAND gate

$$F = \bar{x}y + xy + y\bar{z}$$

$$= \overline{\overline{\bar{x}y + xy + y\bar{z}}} \quad (\because \text{demorgans})$$

$$= \overline{\overline{\bar{x}y} \cdot \overline{xy} \cdot \overline{y\bar{z}}}$$



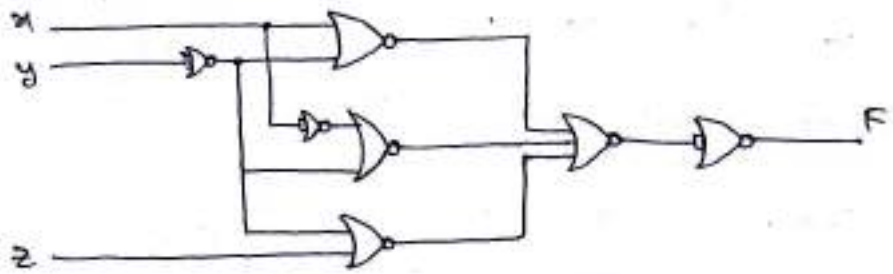
(v) Using NOR gates.

$$F = \bar{x}y + x\bar{y} + y\bar{z}$$

$$= \overline{\overline{\bar{x}y} + \overline{x\bar{y}} + \overline{y\bar{z}}}$$

$$= \overline{\overline{\bar{x}+\bar{y}} + \overline{\bar{x}+\bar{y}} + \overline{\bar{y}+\bar{z}}}$$

$$= \overline{x+\bar{y}} + \overline{\bar{x}+\bar{y}} + \overline{\bar{y}+z} = \overline{\overline{x+\bar{y}} + \overline{\bar{x}+\bar{y}} + \overline{\bar{y}+z}}$$



* Representation of switching functions (or) logic functions (or) Boolean function :-

There are two standard forms in which logic functions can be expressed.

(i) Sum of products form (SOP)

(ii) Product of Sum's (POS).

(i) SOP :- SOP is a group of product terms ORed together.

Ex:- $F = \bar{x}y + x\bar{y} + yz$

(ii) POS :- POS is a group of Sum terms ANDed together.

Ex:- $F = (\bar{x}+\bar{y})(x+y)(y+z)$

Standard and canonical SOP & POS forms :-

In the standard SOP & POS forms all the individual terms do not contain all variables. If each term is SOP & POS form contains all variables then it is canonical ~~standard~~ SOP & POS forms.

Ex. of CSOP is,

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

Ex. of CPOS is,

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

Ex:- convert the given expression is into canonical ~~standard~~ SOP form.

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

Sol:- $= \bar{A}B(C+\bar{C}) + A\bar{B}C \quad (\because C+\bar{C}=1)$

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

(ii) $Y(A,B,C) = AB + BC + CA$

Sol:- $= AB(C+\bar{C}) + Bc(A+\bar{A}) + CA(B+\bar{B})$

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

$= ABc(1+1+1) + AB\bar{C} + A\bar{B}C + \bar{A}BC \quad (\because 1+\text{anything} = 1)$

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

(iii) $A(X,Y,Z) = X + XY + XYZ$

Sol:- $= X(Y+\bar{Y})(Z+\bar{Z}) + XY(Z+\bar{Z}) + XYZ$

$= X(YZ + Y\bar{Z} + \bar{Y}Z + \bar{Y}\bar{Z}) + XY(Z+\bar{Z}) + XYZ$

$= XYZ + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z} + XYZ + XY\bar{Z} + XYZ$

$= XYZ(1+1+1) + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z} + XY\bar{Z}$

$= XYZ + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z} + XY\bar{Z}$

$= XYZ + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z}$

05/08/2014

Ex:- Convert the given expression into canonical POS form.

① $F(A,B,C) = (A+B)(B+C)(A+C)$

Sol:- $= (A+B+C\bar{C})(B+C+A\bar{A})(A+C+B\bar{B})$ ($\because A+BC=(A+B)(A+C)$) ($\because$ distributive)

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

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

(ii) $F(x,y,z) = x(x+y)(x+y+\bar{z})$

Sol:- $= (x+y\bar{y}+z\bar{z})(x+y+z\bar{z})(x+y+\bar{z})$

$= (x+y\bar{y}+z)(x+y\bar{y}+\bar{z})(x+y+z)(x+y+\bar{z})(x+y+\bar{z})$

$= (x+y+z)(x+\bar{y}+z)(x+y+\bar{z})(x+\bar{y}+\bar{z})(x+y+z)(x+y+\bar{z})$

$= (x+y+z)(x+\bar{y}+z)(x+y+\bar{z})(x+\bar{y}+\bar{z})$

* Minterms and Maxterms :-

→ Each individual term in CSOP is called Minterm.

→ Each individual term in CPOS is called Maxterm.

* For 'n' variable logic function there are 2^n minterms and 2^n maxterms.

Minterms and Maxterms for 3 Binary variables :-

Variables			Minterms (m_i)		Maxterm (M_i)	
A	B	C	Term	designation	Term	designation
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0	$A+B+C$	M_0
0	0	1	$\bar{A}\bar{B}C$	m_1	$A+B+\bar{C}$	M_1
0	1	0	$\bar{A}B\bar{C}$	m_2	$A+\bar{B}+C$	M_2
0	1	1	$\bar{A}BC$	m_3	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$A\bar{B}\bar{C}$	m_4	$\bar{A}+B+C$	M_4
1	0	1	$A\bar{B}C$	m_5	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$AB\bar{C}$	m_6	$\bar{A}+\bar{B}+C$	M_6
1	1	1	ABC	m_7	$\bar{A}+\bar{B}+\bar{C}$	M_7

→ the minterm is represented by m, Maxterm by M

→ the subscript 'i' is the decimal number equivalent of binary input.

* With these short hand notations, logic function can be represented as follows.

Ex:- (i) $Y = \overset{1}{A}\overset{1}{B}\overset{1}{C} + \overset{0}{\bar{A}}\overset{1}{B}\overset{1}{C} + \overset{1}{A}\overset{0}{\bar{B}}\overset{1}{C} + \overset{1}{A}\overset{0}{\bar{B}}\overset{0}{\bar{C}}$

$= m_7 + m_3 + m_5 + m_4$

$= \sum m(3,4,5,7)$

$= \sum(3,4,5,7)$

Σ denotes SOP

(ii) $Y = (\overset{0}{A}+\overset{0}{B}+\overset{0}{C})(\overset{1}{\bar{A}}+\overset{0}{B}+\overset{1}{\bar{C}})(\overset{1}{\bar{A}}+\overset{1}{\bar{B}}+\overset{1}{\bar{C}})$

$= M_0 \cdot M_5 \cdot M_7$

$= \pi M(0,5,7)$

$= \pi(0,5,7)$

π denotes POS

Note:- If o/p func is '1' then it corresponds to Minterm.
If o/p func is '0' then it corresponds to Maxterm.

Ex:- Find SOP and POS expressions from given truthtable

i/p's			o/p
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Note:-

o/p is '1' ⟶ Minterm

o/p is '0' ⟶ Maxterm.

Sol:- SOP

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

$= m_0 + m_2 + m_3 + m_7 = \sum m(0,2,3,7)$

POS

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

$= M_1 \cdot M_4 \cdot M_5 \cdot M_6 = \pi M(1,4,5,6)$

* From the above expressions, there is a complementary

Ex:- Simplify the following 3 variable expressions using boolean algebra.

(i) $Y = \sum m(1,3,5,7)$ considering A, B, C as 3 i/p variables.

Sol:- $Y = m_1 + m_3 + m_5 + m_7$

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

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

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

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

(ii) $Y = \pi M(3,7)$

Sol:- $Y = M_3 \cdot M_7$

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

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

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

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

$= \bar{B} + \bar{C} + \bar{B} + \bar{C}$ $(\because 1+\text{anything} = 1)$

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

Ex:- Convert the given expression into Minterms using complementary property. and simplify the expression.

(i) $F = \pi M(3,5,7)$

Sol:- $F = M_3 \cdot M_5 \cdot M_7$ ⟶ This is POS form.

∴ SOP form is,

$F(A,B,C) = \sum m(0,1,2,4,6)$

$= m_0 + m_1 + m_2 + m_4 + m_6$

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

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

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

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

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

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

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

Ex:- Express the following functions in sum of minterms (canonical SOP) and product of max. terms (canonical POS) forms.

(i) $F(w, x, y, z) = w'y + xy' + yz + xyz$.

Sol:-

$w'y$

w'	x'	y	z'	
0	0	1	0	m_2
0	0	1	1	m_3
0	1	1	0	m_6
0	1	1	1	m_7

xy'

w'	x	y'	z'	
0	1	0	0	m_4
0	1	0	1	m_5
1	1	0	0	m_{12}
1	1	0	1	m_{13}.

yz

w	x	y	z	
0	0	1	1	m_3
0	1	1	1	m_7
1	0	1	1	m_{11}
1	1	1	1	m_{15}.

xyz

w	x	y	z	
0	1	1	1	m_7
1	1	1	1	m_{15}.

CSOP (or) Sum of minterms form,

$F(w,x,y,z) = m_2 + m_3 + m_4 + m_5 + m_6 + m_7 + m_{11} + m_{12} + m_{13} + m_{15}$

$= \sum m(2,3,4,5,6,7,11,12,13,15)$

Remaining terms are maxterms.

CPOS (or) Product of Maxterms form.

$F(w,x,y,z) = M_0 \cdot M_1 \cdot M_8 \cdot M_9 \cdot M_{10} \cdot M_{14}$

$= \prod M(0,1,8,9,10,14)$

(2) $F(A,B,C) = (AB+C)(AC+B)$

Sol:- $F(A,B,C) = ABC + AB + AC + BC$.

ABC

A	B	C	
1	1	1	$\rightarrow m_7$

AB

A	B	C	
1	1	0	$-m_6$
1	1	1	$-m_7$

BC

A	B	C	
0	1	1	$-m_3$
1	1	1	$-m_7$

AC

A	B	C	
1	0	1	$-m_5$
1	1	1	$-m_7$

Sum of minterms $F = \Sigma m(3,5,6,7)$.

Product of maxterms $F = \pi M(0,1,2,4)$

06/08/2014

* Karnaugh Map (K-Map) :-

K-Map gives a systematic approch for simplifying a bodean expression. The basis of this method is a graphical chart known as "K-Map". It contains boxes called cells. Each of the cell represents one of the 2^n possible products that can be formed from 'n' variables.

2-variable K-Map

(i) Representation using minterms

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

~~2-variable K-Map representation using minterms~~

A \ B	0	1
0	m_0 00	m_1 01
1	m_2 10	m_3 11

~~2-variable K-Map representation using Maxterms~~

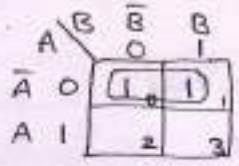
(ii) Representation using maxterms.

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

A \ B	0	1
0	M_0 00	M_1 01
1	M_2 10	M_3 11

Ex:- (i) Simplify $F(A,B) = \Sigma m(0,1)$ using K-Map.

Sol:-



$F(A,B) = \Sigma m(0,1)$

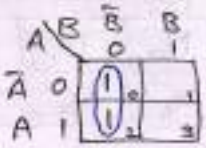
$= m_0 + m_1$

Fill minterms with '1'.

$\therefore \boxed{F(A,B) = \bar{A}}$ $\left(\because F(A,B) = \bar{A}\bar{B} + \bar{A}B = \bar{A}(B+\bar{B}) = \bar{A}\right)$

(ii) Simplify $F(A,B) = \Sigma m(0,2)$ using K-map.

Sol:-



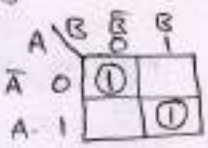
$F(A,B) = \Sigma(m(0,2))$

$= m_0 + m_2$

$\therefore \boxed{F(A,B) = \bar{B}}$

(iii) Simplify $F(A,B) = \Sigma(0,3)$ using K-map

Sol:-



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

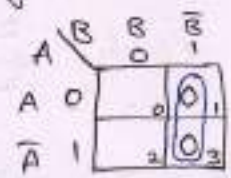
$= \Sigma m(0,3)$

$= m_0 + m_3$

$\therefore \boxed{F(A,B) = \bar{A}\bar{B} + AB}$

(iv) Simplify $F(A,B) = \Pi M(1,3)$ using K-Map.

Sol:-



$F(A,B) = \Pi M(1,3)$

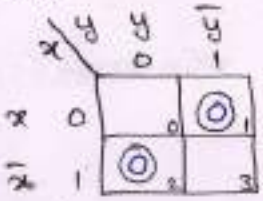
$= M_1 \cdot M_3$

Fill maxterms with 0's.

$\therefore \boxed{F(A,B) = \bar{B}}$

(v) Simplify $F(x,y) = \Pi(1,2)$ using K-map.

Sol:-



$F(x,y) = \Pi M(1,2)$

$= M_1 M_2$

Fill maxterms with 0's.

$\therefore \boxed{F(x,y) = (x+\bar{y})(\bar{x}+y)}$

3- Variable K-Map :-

Representation using Minterms

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

A \ BC	00	01	11	10
0	m_0 0	m_1 1	m_3 3	m_2 2
1	m_4 4	m_5 5	m_7 7	m_6 6

Representation using Maxterms

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

A \ BC	00	01	11	10
0	M_0 0	M_1 1	M_3 3	M_2 2
1	M_4 4	M_5 5	M_7 7	M_6 6

Ex: ① simplify $F(A,B,C) = \sum m(1,5,2,6)$ using K-Map.

Sol:-

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0	1 1	3	1 2
A 1	4	1 5	7	1 6

$F(A,B,C) = \sum m(1,5,2,6)$

$= m_1 + m_2 + m_5 + m_6$.

Fill minterms with 1's.

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

$\therefore \boxed{F(A,B,C) = B \oplus C}$

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

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

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

$= B \oplus C.)$

(ii) Simplify $F(A,B,C) = \sum(1,3,5,7,6)$ using K-Map.

Sol:-

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0	1 1	1 3	2
A 1	4	1 5	1 7	1 6

$F(A,B,C) = \sum(1,3,5,7,6)$

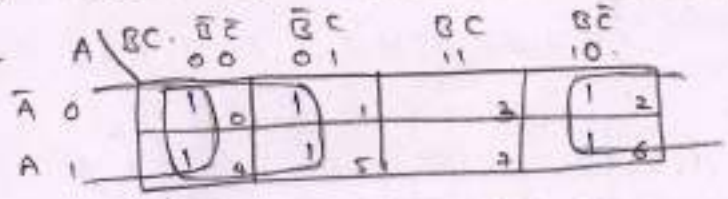
$= m_1 + m_3 + m_5 + m_6 + m_7$

Fill minterms with 1's.

$\therefore \boxed{F(A,B,C) = C + BA}$

(iii) $F(A,B,C) = \Sigma(0,1,4,5,2,6)$.

Sol:-

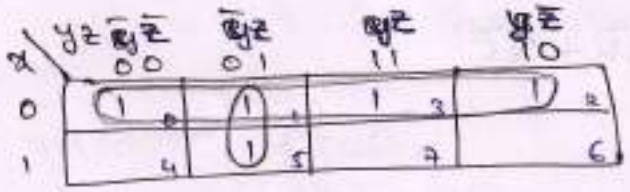


$F(A,B,C) = m_0+m_1+m_2+m_4+m_5+m_6$.

$\therefore F(A,B,C) = \bar{B} + \bar{C}$.

(iv) $F(x,y,z) = \Sigma m(0,1,2,3,5)$.

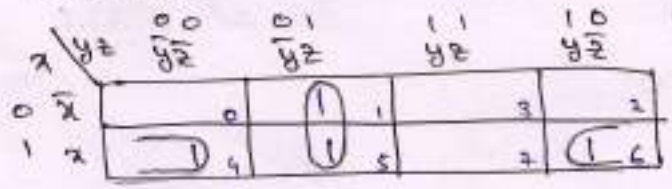
Sol:-



$\therefore F(x,y,z) = \bar{x} + \bar{y}z$

(v) $F(x,y,z) = \Sigma(1,4,5,6)$.

Sol:-



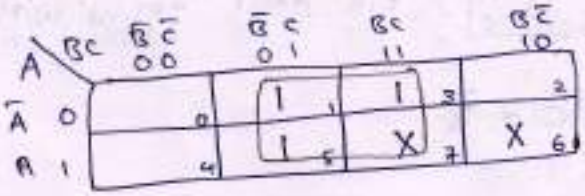
$\therefore F(x,y,z) = \bar{y}z + x\bar{z}$

<u>don't care condition</u>

X → don't care symbol, it can be 0 or 1.

(vi) $Y(A,B,C) = \Sigma m(1,3,5) + \Sigma d(6,7)$.

Sol:-



$\therefore Y(A,B,C) = C$

$Y(A,B,C) = m_1+m_3+m_5+d_6+d_7$

Fill mixterms with '1'.

Fill don't terms with 'X'

It may be '0' or '1'.

consider $d_7 = 1$.

Ex:- Simplify the following boolean expression and draw logic circuit.

(vii) $Y(A,B,C) = \Sigma(0,1,2,6)$

Sol:-

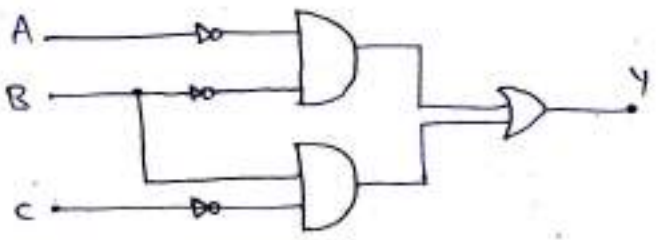
A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1 (0)	1 (1)	(3)	1 (2)
A 1	(4)	(5)	(7)	1 (6)

$Y(A,B,C) = \Sigma(0,1,2,6)$

$= m_0 + m_1 + m_2 + m_6$

$\therefore Y(A,B,C) = \bar{A}\bar{B} + B\bar{C}$.

Logic circuit :-



Ex:- Simplify the following boolean expression and draw logic circuits using NAND gate only.

(viii) $F = \Sigma(0,1,5,6) + \Sigma d(4,3)$

Sol:-

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1 (0)	1 (1)	X (3)	(2)
A 1	X (4)	1 (5)	(7)	1 (6)

$F = m_0 + m_1 + m_5 + m_6 + d_3 + d_4$

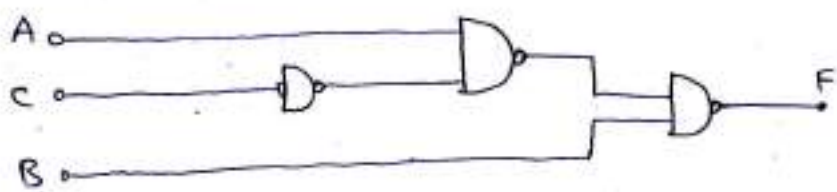
Fill minterms with 1's

Fill dont terms with x's.

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

Logic circuit:- $F = A\bar{C} + \bar{B} = \overline{\overline{A\bar{C} + \bar{B}}} = \overline{\bar{\bar{B}} \cdot \overline{A\bar{C}}}$

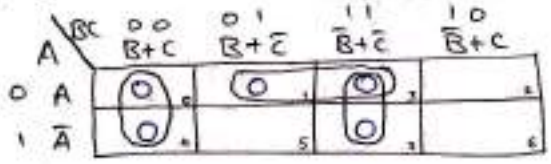
$= \overline{B \cdot \overline{A\bar{C}}}$



07/08/2014

(ix) $F(A,B,C) = \pi M(0,1,3,4,7)$

Sol:-



$F(A,B,C) = \pi M(0,1,3,4,7)$

$= M_0 \cdot M_1 \cdot M_3 \cdot M_4 \cdot M_7$

Fill Maxterms with 0's.

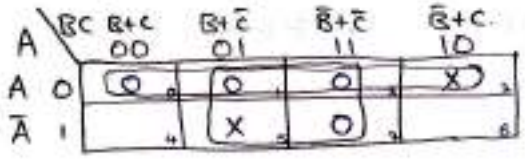
$\therefore F(A,B,C) = (B+C)(\bar{B}+\bar{C})(A+\bar{C})$

(or)

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

(x) $F = \pi M(0,1,3,7) + \pi d(2,5)$

Sol:-



$F = \pi M(0,1,3,7) + \pi d(2,5)$

$= M_0 \cdot M_1 \cdot M_3 \cdot M_7 + d_2 \cdot d_5$.

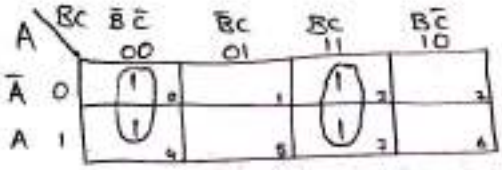
Fill Maxterms with 0's.

Fill don't terms with 'x's consider d_2, d_5 as 0's.

$\therefore F(A,B,C) = A\bar{C}$.

(xi) Given $F = \Sigma(0,3,4,7)$. Find (a) F (b) dual of F (c) F'.

Sol:-



$F = \Sigma m(0,3,4,7)$

$= m_0 + m_3 + m_4 + m_7$

Fill minterms with 1's.

$F \rightarrow$ ~~POS~~ SOP

(i) $F(A,B,C) = \bar{B}\bar{C} + BC = B \odot C$

(ii) Dual of F is $(\bar{B}+\bar{C})(B+C)$

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

$= B\bar{C} + \bar{B}C = B \oplus C$.

$\therefore$ Dual of $F = B\bar{C} + \bar{B}C = B \oplus C$

(iii) $F' \rightarrow$ SOP.

$\therefore F' = \pi(1,2,5,6)$

$\therefore F' = (\bar{B}+C)(B+C')$. ($\because F'$ = dual of F).

4-variable K-Map :-

Representation using Minterms

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

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

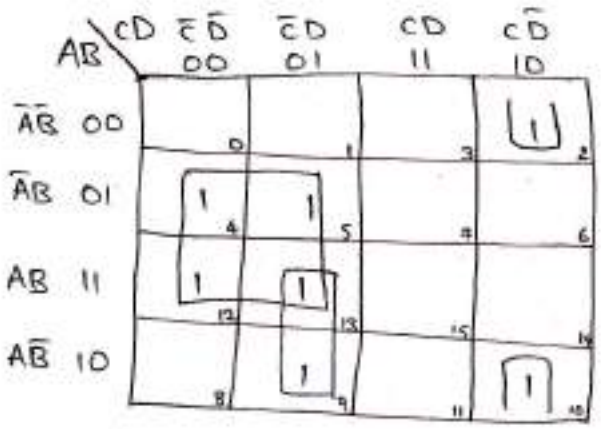
Representation using Maxterms

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

AB \ CD	00	01	11	10
00	M_0	M_1	M_3	M_2
01	M_4	M_5	M_7	M_6
11	M_{12}	M_{13}	M_{15}	M_{14}
10	M_8	M_9	M_{11}	M_{10}

Ex:- (i) $F(A,B,C,D) = \sum m(2,4,5,9,10,12,13)$

Sol:-



$F(A,B,C,D) = \sum m(2,4,5,9,10,12,13)$

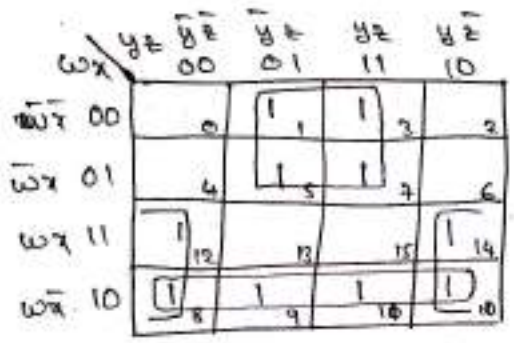
$= m_2 + m_4 + m_5 + m_9 + m_{10} + m_{12} + m_{13}$

Fill Minterms with 1's.

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

(ii) $F(w,x,y,z) = \sum m(1,3,5,7,8,9,10,11,12,14)$.

Sol:



$F(w,x,y,z) = m_1+m_3+m_5+m_7+m_8+m_9+m_{10}+m_{11}+m_{12}+m_{14}$

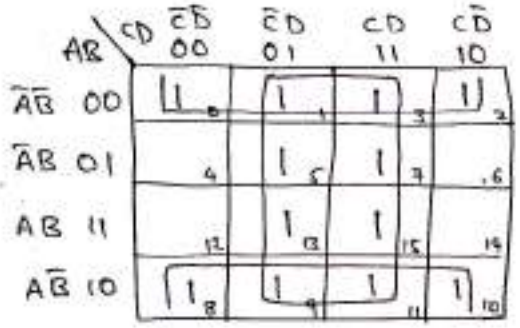
Fill minterms with 1's.

$F(w,x,y,z) = z\bar{w} + w\bar{z} + w\bar{x}$

$= w\bar{x} + w\bar{z} + \bar{w}z$

(iii) $F(A,B,C,D) = \sum(0,1,2,3,5,7,8,9,10,11,13,15)$.

Sol:



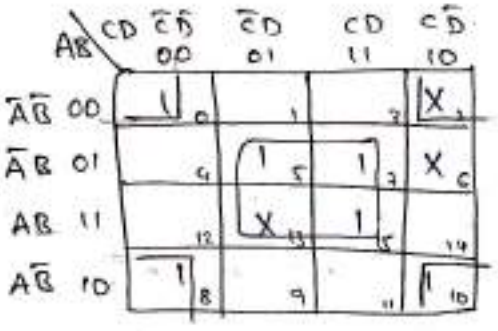
$F(A,B,C,D) = m_0+m_1+m_2+m_3+m_5+m_7+m_8+m_9+m_{10}+m_{11}+m_{13}+m_{15}$

Fill minterms with 1's.

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

(iv) $F = \sum(0,5,7,8,10,15) + \sum d(2,6,13)$.

Sol:



$F = \sum(0,5,7,8,10,15) + \sum d(2,6,13)$

$= m_0+m_5+m_7+m_8+m_{10}+m_{15} + d_2+d_6+d_{13}$.

Fill Minterms with 1's.
Fill don't terms with X's.
Consider d_{13}, d_2 as 1's.

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

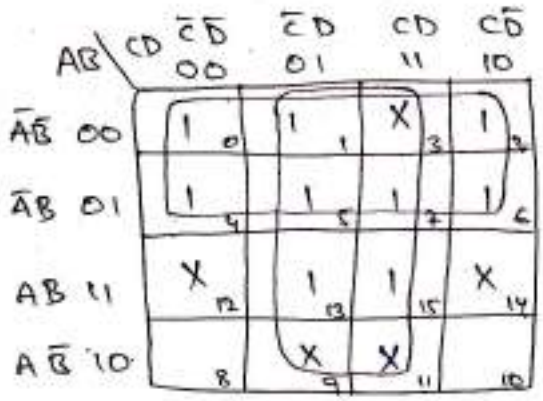
(v) $F = \sum(0,1,2,4,5,6,7,13,15) + \sum\phi(3,12,14,9,11)$.

Sol:

$F = m_0+m_1+m_2+m_4+m_5+m_6,m_7,m_{13}+m_{15} + \phi_3+\phi_{12}+\phi_{14}+\phi_9+\phi_{11}$

Fill Minterms with 1's.

Consider ϕ_3, ϕ_{11} as 1's.



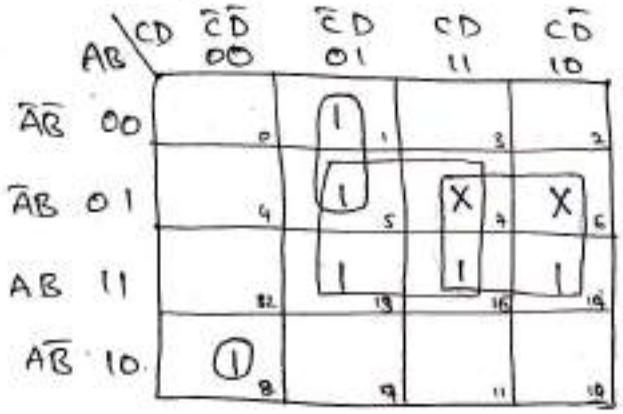
$\therefore F(A,B,C,D) = \bar{A} + D.$

(or)

$\bar{A} + B.$

(vi) $F = \sum m(1,5,8,15,13,14) + \sum d(6,7).$

Sol:-



$F = m_1 + m_5 + m_8 + m_{15} + m_{13} + m_{14} + d_6 + d_7$

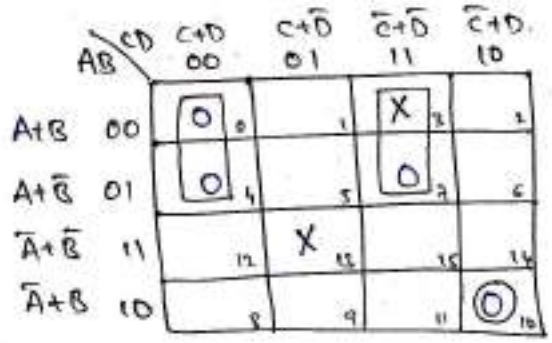
Fill 'd' terms with x's

consider d_6, d_7 as is.

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

(vii) $F = \pi M(0,4,7,10) + \pi d(3,13)$

Sol:-



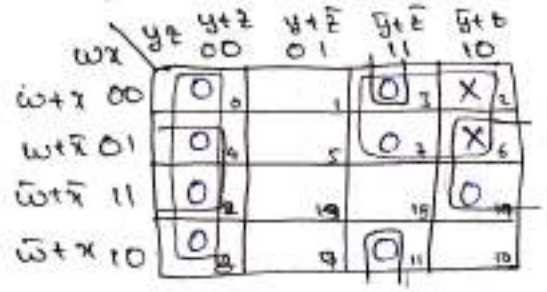
$F = M_0 \cdot M_4 \cdot M_7 \cdot M_{10} + d_3 \cdot d_{13}$

consider d_3 as 0.

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

(viii) $F(w,x,y,z) = \pi(0,3,4,7,8,11,12,14) + \pi\phi(2,6)$.

Sol:-



$F(w,x,y,z) = M_0 \cdot M_3 \cdot M_4 \cdot M_7 \cdot M_8 M_{11} \cdot M_{12} \cdot M_{14} + \phi_2 \cdot \phi_6$

Fill Maxterms with 0's.
Fill ϕ terms with x's.
consider ϕ_2, ϕ_6 as 0's.

$\therefore F(w,x,y,z) = (y+z)(\bar{x}+z)(x+\bar{y}+\bar{z})$

12/08/2014

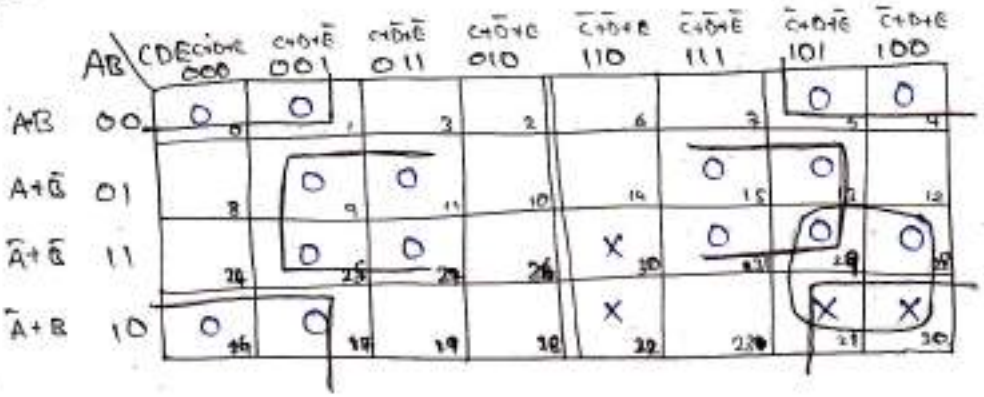
5-variable K-Map :-

Representing using minterms & maxterms

Ex:- Simplify the following boolean expressions using K-map.

(i) $F = \pi M(0,1,4,5,9,11,12,15,16,17,25,27,28,29,31) + \pi d(20,21,22,30)$

Sol:-



$\therefore F = (\bar{B}+\bar{E})(B+D)(\bar{A}+\bar{C}+D)$

(ii) $F = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

Sol:-

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

Note:- (i) In K-Map pair is a group of two adjecent cells which cancels one variable.

(ii) Quad is a group of 4 adjecent cells which cancels two variables.

(iii) octet is a group of 8 adjecent cells which cancels three variables.

Ex:- ① Using demorgan's law simplify the following.

(i) $Y_1 = ((a+b')(c'+d))'$

Sol:- $Y_1 = \overline{((a+\bar{b})(\bar{c}+d))}$ (∵ demorgan's law $\overline{X+Y} = \bar{X}\cdot\bar{Y}$, $\overline{X\cdot Y} = \bar{X}+\bar{Y}$)

$= (\overline{a+\bar{b}})+(\overline{\bar{c}+d})$

$= (\bar{a}\cdot\bar{\bar{b}})+(\bar{\bar{c}}\cdot\bar{d})$

$= (\bar{a}\cdot b)+(c\cdot\bar{d})$

$\therefore \boxed{Y_1 = \bar{a}\cdot b + c\cdot\bar{d}}$

(ii) $Y_2 = (a+b')'$

Sol:- $Y_2 = (a+b')'$

$= \bar{a}\cdot\overline{b} = \bar{a}\cdot b$

$\boxed{Y_2 = \bar{a}\cdot b}$

Ex: ② simplify the following expressions and implement using 2-level NAND gate.

(i) $AB' + ABD + ABD' + A'C'D' + A'BC'$ $(\because A+BC = (A+B)(A+C))$

Sol: $AB' + AB(D+D') + A'C'D' + A'BC'$

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

$= A + A'C'D' + A'BC'$ $= (A+A')(A+C'D') + A'BC'$

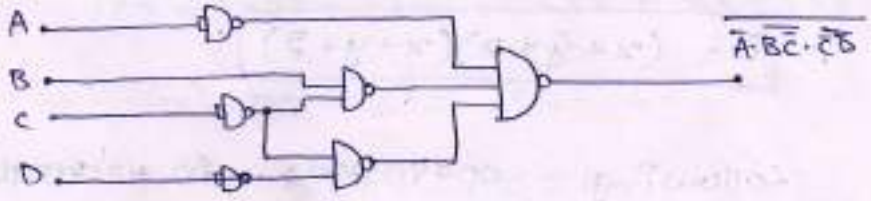
$= (A + A'C'D' + A'BC')''$ $= A + C'D' + A'BC'$

$= (A+BC')(A+BC') + C'D'$

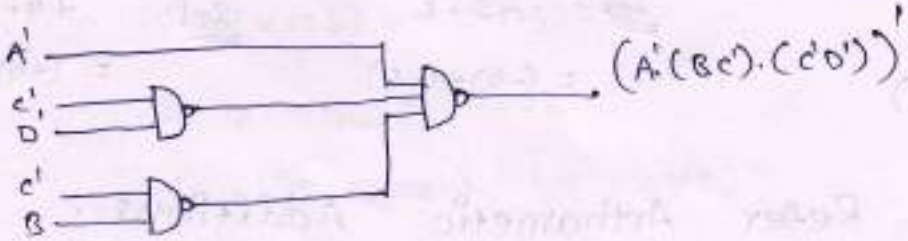
$= A + BC' + C'D'$

$= ((A+BC'+C'D')')'$

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



(or)



Note:-

i/p variables — n

∴ Total number of Maxterm $= 2^n$

" " " Minterms $= 2^n$

" " " Boolean expressions $= 2^{2^n}$.

Ex:- ③ The minimum number of NAND gates required to implement $A + AB' + ABC'$.

Sol:- $A(1 + \bar{B} + B\bar{C})$ $(\because 1 + \text{anything} = 1)$

$= A$ — A

$\therefore$ the no. of NAND gates required $= 0$.

④ Find complement of $F = \bar{x}y\bar{z} + \bar{x}\bar{y}z$

Sol:- $F = \bar{x}y\bar{z} + \bar{x}\bar{y}z$

complement of F is $\bar{F} = \overline{\bar{x}y\bar{z} + \bar{x}\bar{y}z}$

$= \overline{\bar{x}y\bar{z}} \cdot \overline{\bar{x}\bar{y}z}$

$= (\bar{\bar{x}} + \bar{y} + \bar{\bar{z}})(\bar{\bar{x}} + \bar{\bar{y}} + \bar{z})$

$= (x + \bar{y} + z)(x + y + \bar{z})$

$\therefore \boxed{\bar{F} = (x + \bar{y} + z)(x + y + \bar{z})}$

⑤ Perform the following operations in 2's compliment form. 8-bit Binary arthametic.

(i) $+6 - 3$ (ii) $-2 - 6$ (iii) $+4 - 7$

Sol:- $+6 - 3$

$= (+6) + (-3)$

Sol:- $-2 - 6$

$= (-2) + (-6)$

Sol:- $+4 - 7$

$= (+4) + (-7)$

Refer Arthametic Additions.

⑥ Determine the value of Base x.

(i) $(211)_x = (152)_8$ convert both numbers into decimal.

Sol:- $2\times x^2 + 1\times x^1 + 1\times x^0 = 1\times 8^2 + 5\times 8 + 2\times 8^0$

$\Rightarrow 2x^2 + x + 1 = 64 + 40 + 2$

$\therefore 2x^2 + x - 105 = 0$

$2x^2 + 15x - 14x - 105 = 0$

$2x(x-7) + 15(x-7) = 0$

$(x-7)(2x+15) = 0$

$\therefore x = 7$ (or) $\frac{-15}{2}$

It can't be -ve

$\therefore \boxed{x = 7}$

(ii) $(193)_x = (623)_8$

Sol:- $1\times x^2 + 9\times x^1 + 3\times x^0 = 6\times 8^2 + 2\times 8^1 + 3\times 8^0$

$\Rightarrow x^2 + 9x + 3 = 384 + 16 + 3$

$\therefore x^2 + 9x - 400 = 0$

$x^2 + 25x - 16x - 400 = 0$

$x(x+25) - 16(x+25) = 0$

$\therefore x = 16$ (or) -25

It can't be -ve

$\therefore \boxed{x = 16}$

(iii) $(\sqrt{41})_x = 5_{10}$

Sol:- $(\sqrt{41})_x = 5$

$(\sqrt{41})^2 = 5^2$

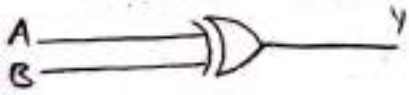
$\therefore (41)_x = 25$

$4\times x + 1\times x^0 = 25$

$4x = 24$

$\therefore \boxed{x = 6}$

EXOR function :-



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

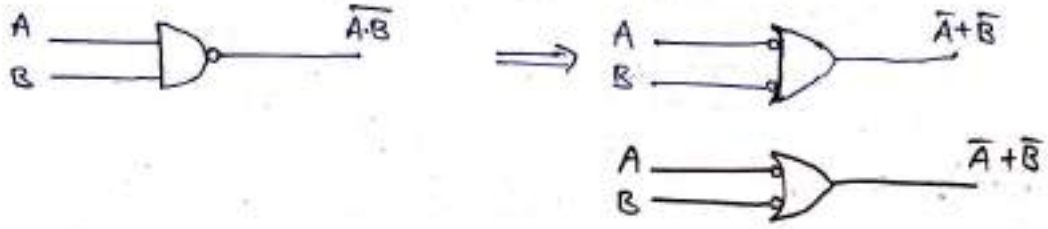
EXOR properties :-

1. $A \oplus A = 0$
2. $A \oplus \bar{A} = 1$
3. $A \oplus 1 = \bar{A}$ (Inverter)
4. $A \oplus 0 = A$ (Non - Inverter)

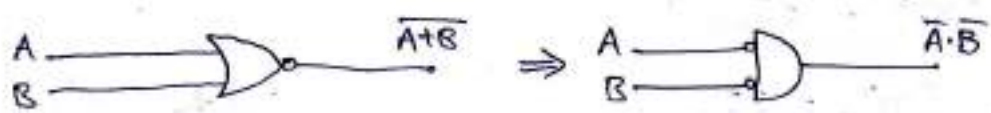
Alternative gate representation :-

1. NOT Gate :-

2. NAND Gate

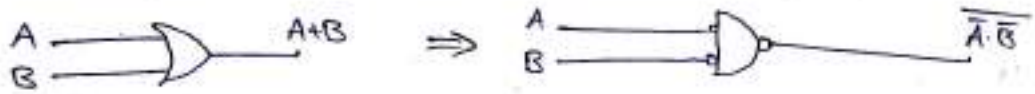


3. NOR Gate



4. AND Gate

5. OR Gate



Convertion of AND, OR, NOT logic to NAND/NOR logic using Graphical procedure :-

Rules :-

(i) Draw logic circuit using basic gates.

(ii) If it is an AND/OR gate place a bubble at output side.

(iii) If it is an OR/AND gate place a bubble at input side.

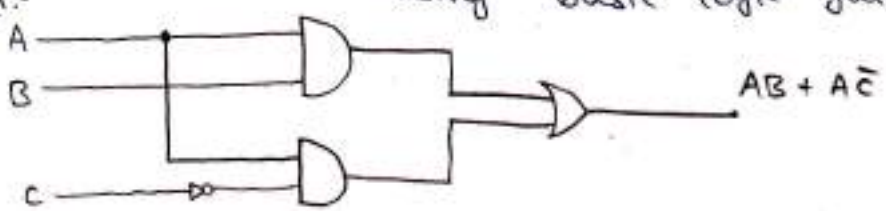
(iv) Place a NOT gate where we placed the bubble.

(v) Replace bubbled AND/OR by using NAND/NOR.

(vi) Eliminate double inverters.

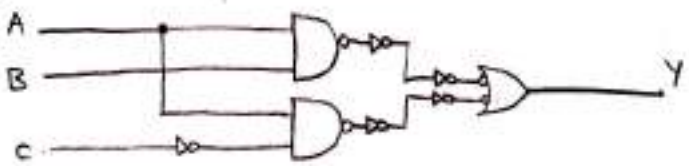
Ex:- $Y = AB + A\bar{C}$ using NAND logic.

Sol:- Step 1:- using basic logic gates.

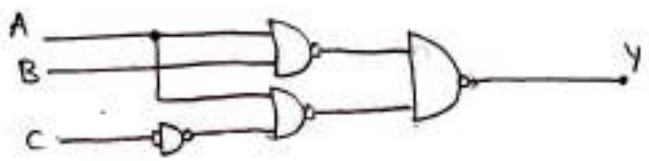


Graphical method.

Step 2:-

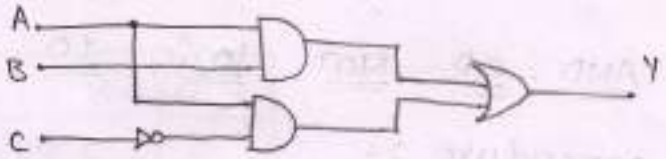


Step 3:-

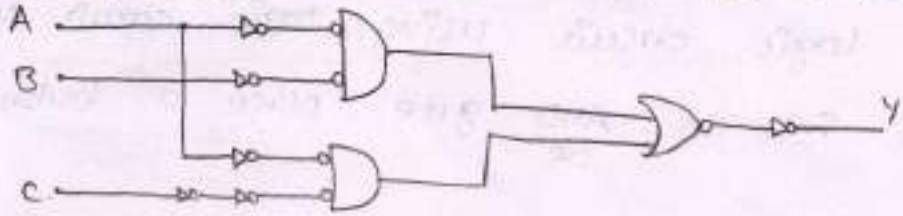


Ex:- $Y = AB + A\bar{C}$ using NOR Gate.

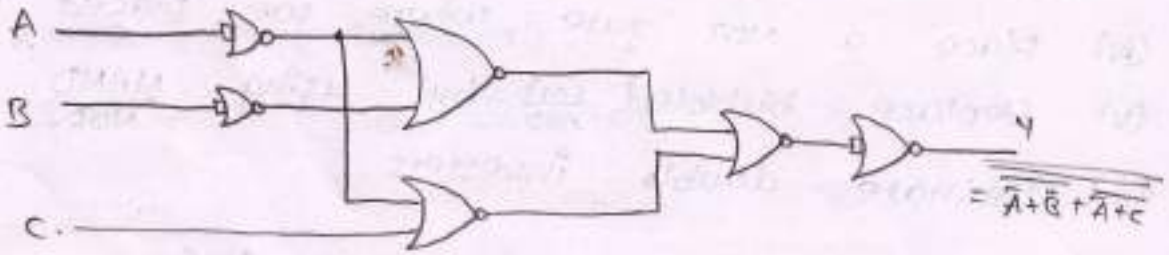
Sol:- Step 1:-



Step 2:- In OR gate place a bubble at o/p side & place a NOT gate.
In AND gate place a bubble at i/p side & place a NOT gate.



Step 3:- Replace bubble AND gate by using NOR.



13-08-2014

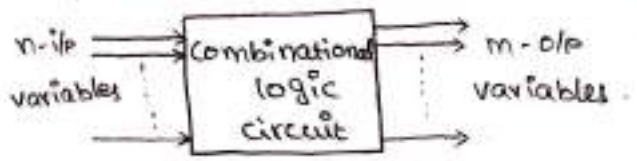
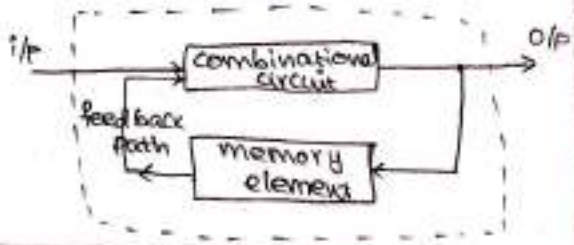
UNIT-II

COMBINATIONAL LOGIC

There are two types of digital logic circuits.

(i) combinational circuit

(ii) sequential circuit

combinational circuits	sequential circuit
* o/p depends on present i/p's. * Contains no internal memory * It consists of logic gates * Block diagram. 	* o/p depends on present i/p's & past o/p's. * contains internal memory *It consists of logic gates & memory elements. *Block diagram 

Design procedure of a combinational circuit :-

Step 1 :- From the specifications of the circuit, determine the required number of i/p's & o/p's and assign a symbol to each.

Step 2 :- Derive the truth table that defines the required relation b/w i/p's & o/p's.

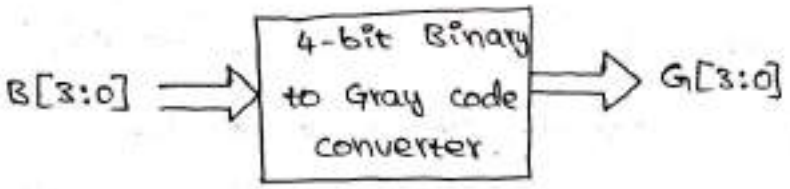
Step 3 :- Obtain the simplified boolean functions for each o/p as a function of the i/p variables.

Step 4 :- Draw the logic diagram and verify the correctness of the design.

* Code converters :-

→ Construct 4-bit binary to Gray code converter

4-bit Binary to Gray code converter converts 4-bit binary to 4 bit Gray code.



$B[3:0] \longrightarrow$ 4-bit Binary number

$B = B_3 B_2 B_1 B_0$

$G[3:0] \longrightarrow$ 4-bit Gray number

$G = G_3 G_2 G_1 G_0$

Truth Table :-

	i/p (Binary)				o/p (Gray)			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

In Gray numbers, 1's are minterms, 0's are Maxterms.

$G_0 = \Sigma m(1,2,5,6,9,10,13,14)$

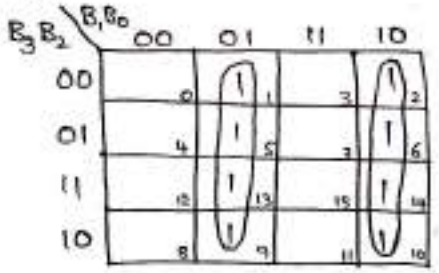
$G_1 = \Sigma m(2,3,4,5,10,11,12,13)$

$G_2 = \Sigma m(4,5,6,7,8,9,10,11)$

$G_3 = \Sigma m(8,9,10,11,12,13,14,15)$

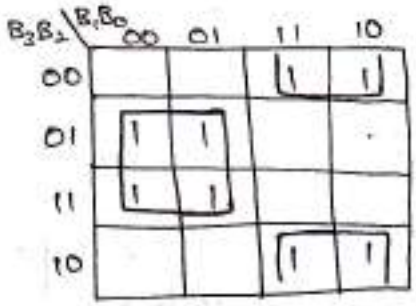
K-map simplification :-

For G0 :-



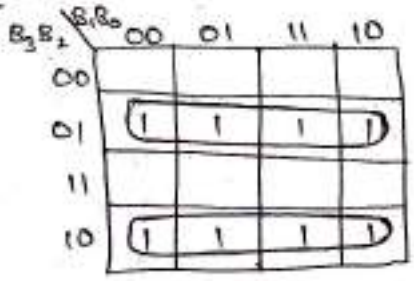
$\therefore \boxed{G_0 = \bar{B}_1 B + B_1 \bar{B}_0 = B_1 \oplus B_0}$

For G_1 :-



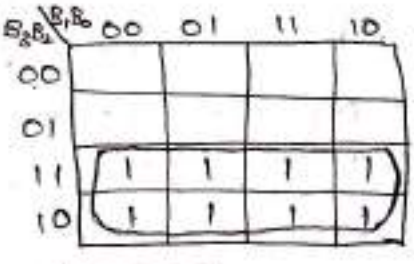
$\therefore \boxed{G_1 = B_1 \bar{B}_2 + \bar{B}_1 B_2 = B_2 \oplus B_1}$

For G_2 :-

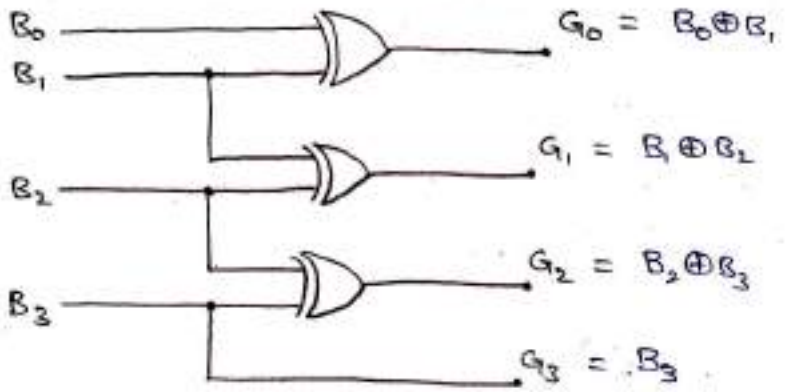


$\therefore \boxed{G_2 = B_3 \bar{B}_2 + \bar{B}_3 B_2 = B_3 \oplus B_2}$

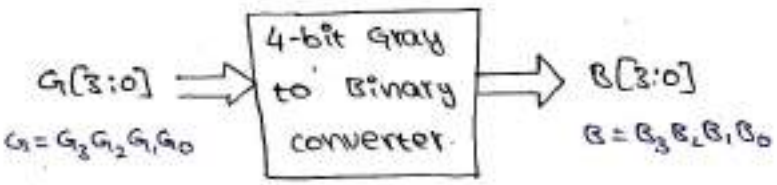
For G_3 :-



logic circuits :-



* Gray to Binary converter :-



Truth table :-

	i/p (Gray)				o/p (Binary)			
	G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

	i/p (Gray)				o/p (Binary)			
	G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	0
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	0	1	0

$B_0 = \sum m(1,3,5,7,9,11,13,15) = \sum m(1,2,4,7,8,11,13,14)$

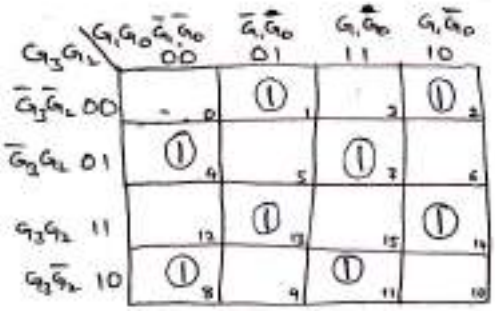
$B_1 = \sum m(2,3,6,7,10,11,14,15) = \sum m(2,3,4,5,8,9,14,15)$

$B_2 = \sum m(4,5,6,7,12,13,14,15) = \sum m(4,5,6,7,8,9,10,11)$

$B_3 = \sum m(8,9,10,11,12,13,14,15) = \sum m(8,9,10,11,12,13,14,15)$

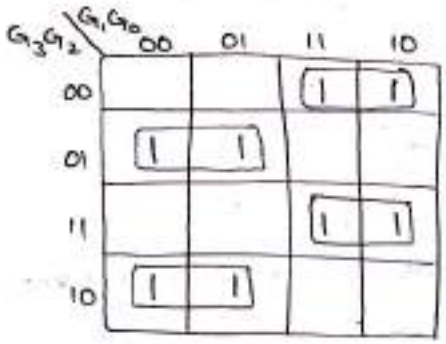
K-Map Simplification :-

For B_0 :-



$\therefore B_0 = (G_0 \oplus G_1) \oplus (G_2 \oplus G_3)$

For B_1 :-

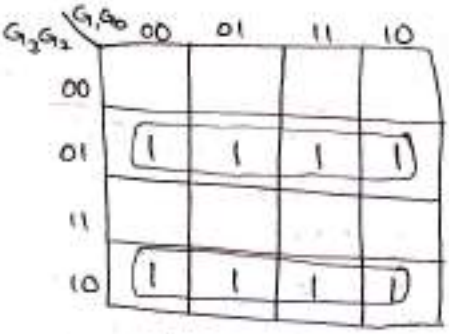


$\therefore B_1 = \bar{G}_2\bar{G}_3G_1 + G_2\bar{G}_3\bar{G}_1 + G_2G_3G_1 + \bar{G}_2G_3\bar{G}_1$

$= \bar{G}_3(G_1\bar{G}_2 + \bar{G}_1G_2) + G_3(G_1G_2 + \bar{G}_1\bar{G}_2)$

$\therefore B_1 = G_1 \oplus G_2 \oplus G_3$

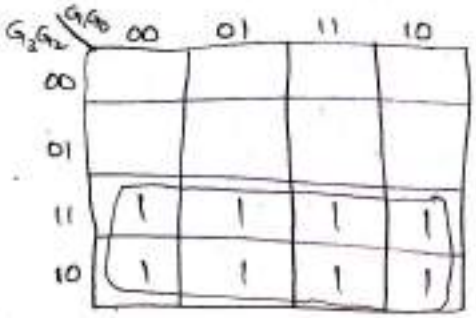
For B_2 :-



$\therefore B_2 = G_2\bar{G}_3 + \bar{G}_2G_3$

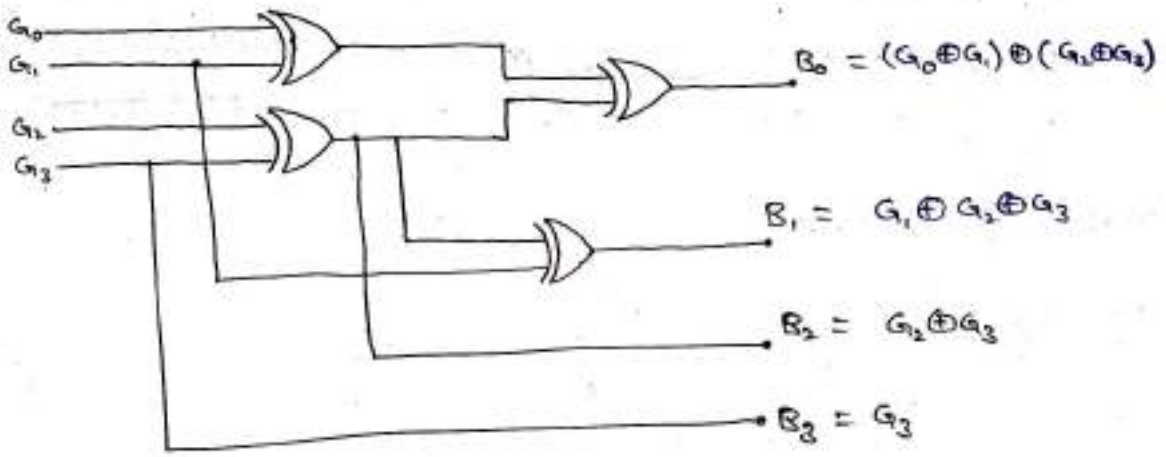
$\therefore B_2 = G_2 \oplus G_3$

For B_3 :-

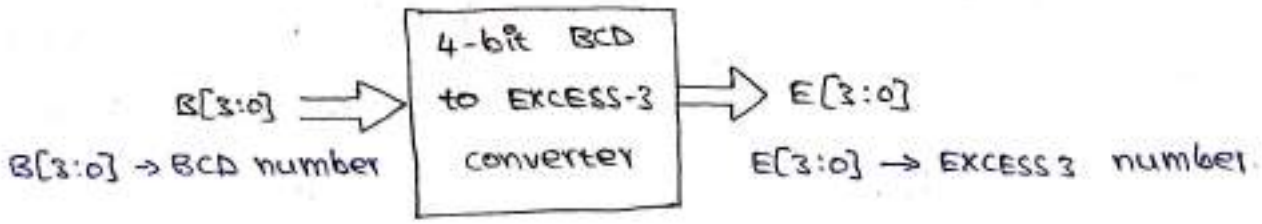


$\therefore B_3 = G_3$

logic circuit :-



* → BCD to EXCESS-3 converter :-



Truth table :-

i/p (BCD)				o/p (EXCESS-3)			
B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	0	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

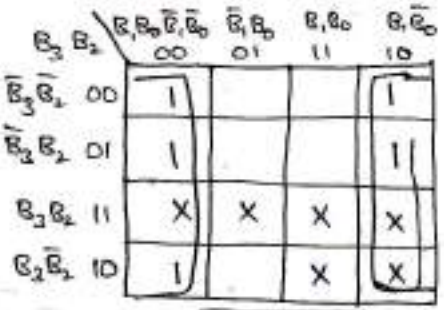
BCD numbers are 0-9

∴ consider 10-15 as don't cares.

$$E_0 = \Sigma m(0,2,4,6,8) + \Sigma d(10,11,12,13,14,15)$$
$$E_1 = \Sigma m(0,3,4,7,8) + \Sigma d(10,11,12,13,14,15)$$
$$E_2 = \Sigma m(1,2,3,4,9) + \Sigma d(10,11,12,13,14,15)$$
$$E_3 = \Sigma m(5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$$

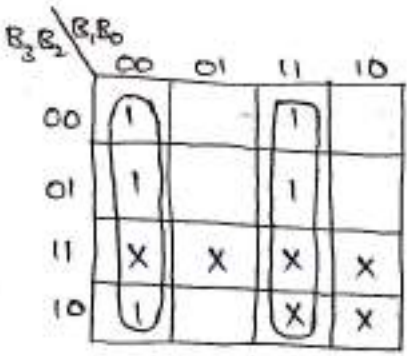
K-MAP Simplification :-

For E_0 :-



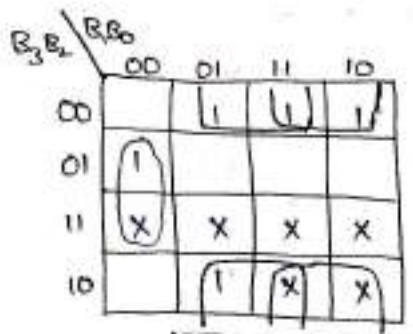
$\therefore$ $E_0 = \bar{B}_0$

For E_1 :-



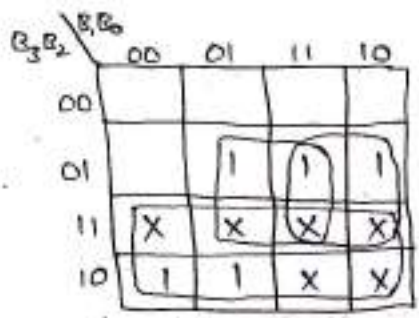
$\therefore$ $E_1 = B_1 \odot B_0 = \bar{B}_0\bar{B}_1 + B_0B_1$

For E_2 :-



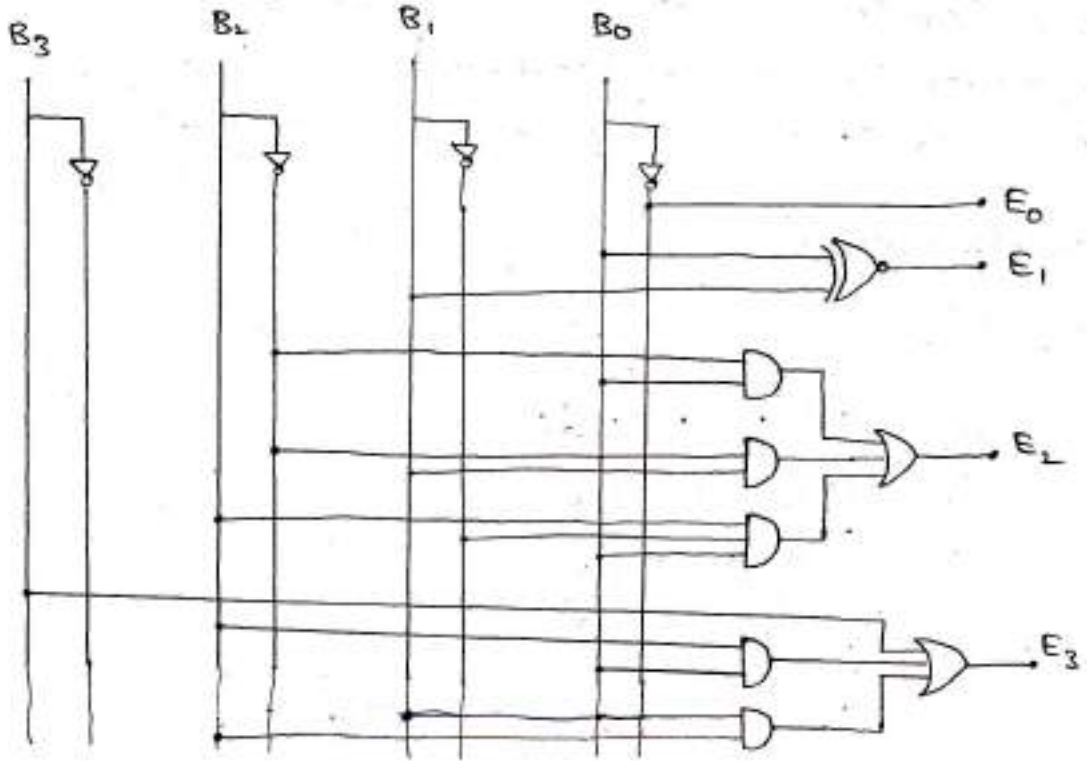
$\therefore$ $E_2 = \bar{B}_2B_0 + \bar{B}_2B_1 + B_2\bar{B}_1\bar{B}_0$

For E_3 :-



$\therefore$ $E_3 = B_3 + B_2B_0 + B_2B_1$

Logic circuit :-



2/08/2014

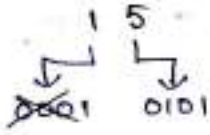
* → EXCESS-3 to BCD converter :-

Truth table :-

i/p (Excess-3)				o/p (BCD)			
E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	0	0	x	x	x	x
0	0	0	1	x	x	x	x
0	0	1	0	x	x	x	x
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Binary to BCD converter :-

Ex :-



for 10 - 15 no.'s 3 MSB's are 0's.

∴ neglect 3 MSB's.

Truth table :-

i/p (Binary)				o/p (BCD)				
B_3	B_2	B_1	B_0	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0	1

BCD to Binary converter :-

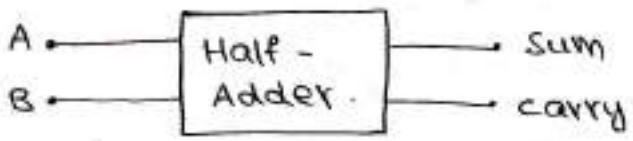
input (BCD)					o/p (Binary)			
D_4	D_3	D_2	D_1	D_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	1	0	0
0	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	0	0	0
0	1	0	0	1	1	0	0	1
1	0	0	0	0	1	0	1	0
1	0	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	0
1	0	0	1	1	1	1	0	1
1	0	1	0	0	1	1	1	0
1	0	1	0	1	1	1	1	1

* **ADDERS :-** Adders is a digital circuit which performs addition operations.

1. **Half Adders :-**

The logic circuit which performs addition of 2bits is called a "Half Adder". It contains two Binary i/p (Augend & Addend) and two Binary o/p (sum & carry).

Block diagram :-

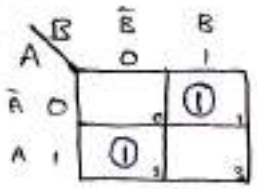


Truth table :-

i/p's		o/p's	
A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-Map Simplification :-

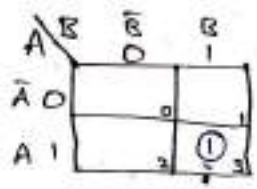
For Sum :-



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

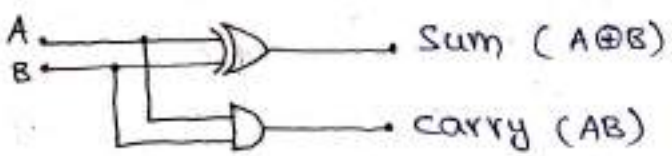
$= A \oplus B$

For carry :-



$\therefore$ carry $= AB$.

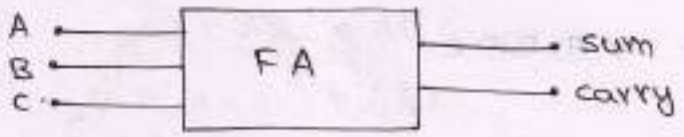
Logic circuit :-



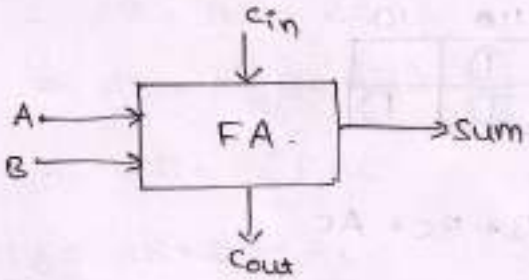
2. Full Adder :-

Full Adder is a combinational circuits that forms the arithmetic sum of 3-binput bits. It consists of 3 ilp's & 2 olp. The third input is a Cin which represents the carry from the previous significant position.

Block diagram :-



(or)



Truth table :-

ilp's			olp's	
A	B	C	Sum	carry
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

K-Map simplifications:-

For sum

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

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

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

$= \bar{A}(B \oplus C) + A(\overline{B \oplus C})$ $\quad (\because \bar{x}y + x\bar{y} = x \oplus y)$

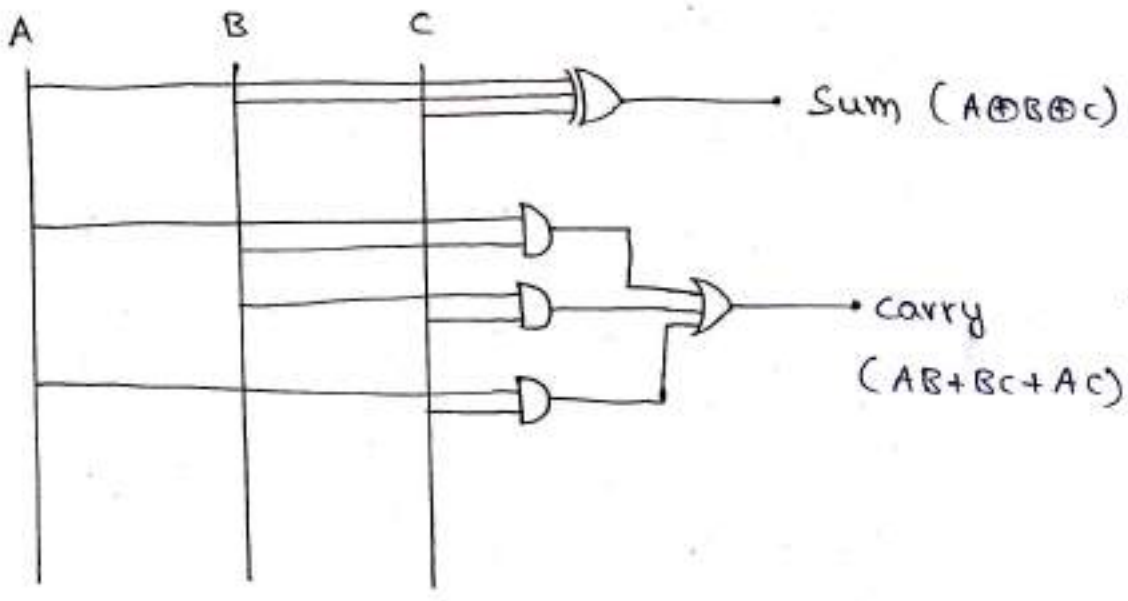
$= A \oplus B \oplus C$

For carry

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

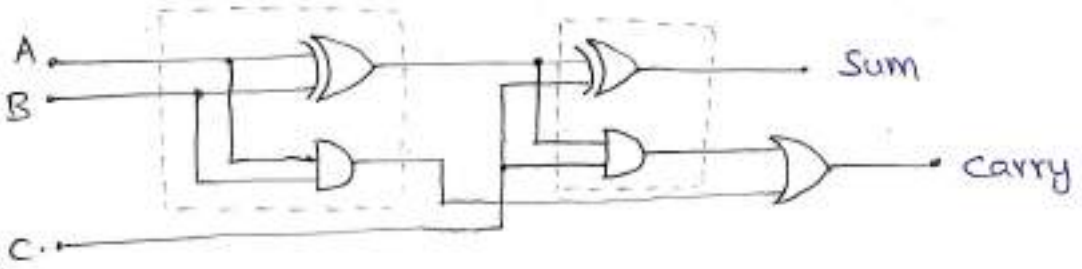
$\therefore$ carry $= AB + BC + AC$

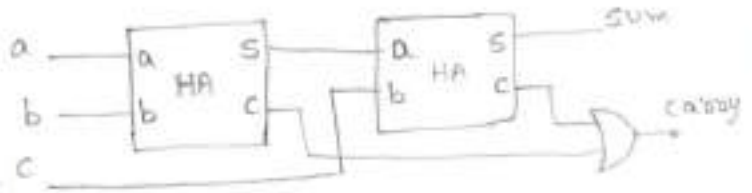
Logic circuit:-



Note:- A full adder can also be implemented by using two half-adders and 1 OR gate.

Full Adder using Half Adders :-





$\therefore$ Sum = $A \oplus B \oplus C$.

$$\begin{aligned}
\text{Carry} &= AB + (A \oplus B)C \\
&= AB + (\bar{A}B + A\bar{B})C \\
&= AB + \bar{A}BC + A\bar{B}C \\
&= B(A + \bar{A}C) + A\bar{B}C \\
&= B((A+\bar{A})(A+C)) + A\bar{B}C \\
&= AB + BC + A\bar{B}C \\
&= AB + C(\bar{B} + A\bar{B}) = AB + C((B+A)(B+\bar{B})) \\
&= AB + BC + AC.
\end{aligned}$$

$\therefore$ carry = $AB + BC + AC$.

3. Parallel Adder (Binary Adder) :-

It is a combinational circuit which performs addition of 2, n-bit number.

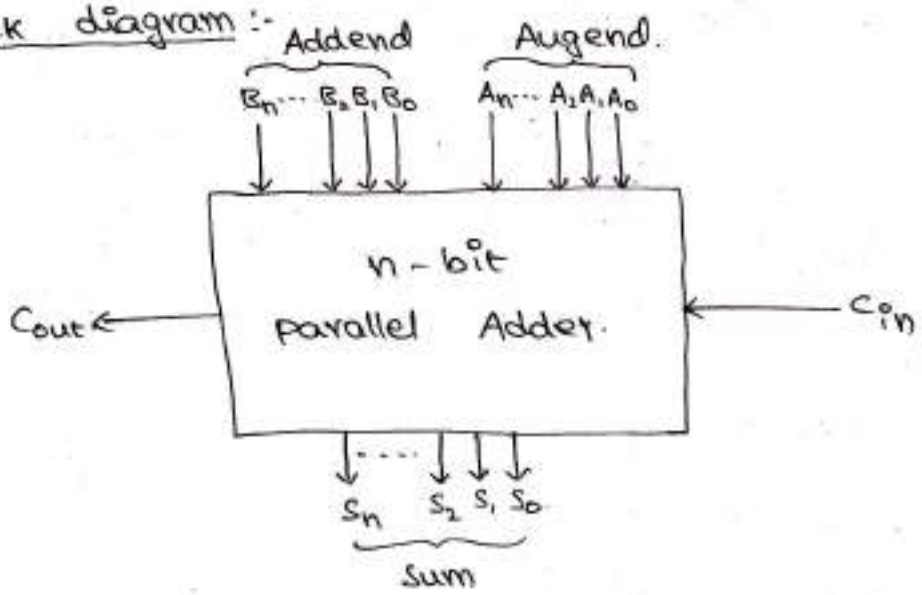
A & B are n+1 bit numbers.

$A \longrightarrow A_n \cdots A_3 A_2 A_1 A_0$

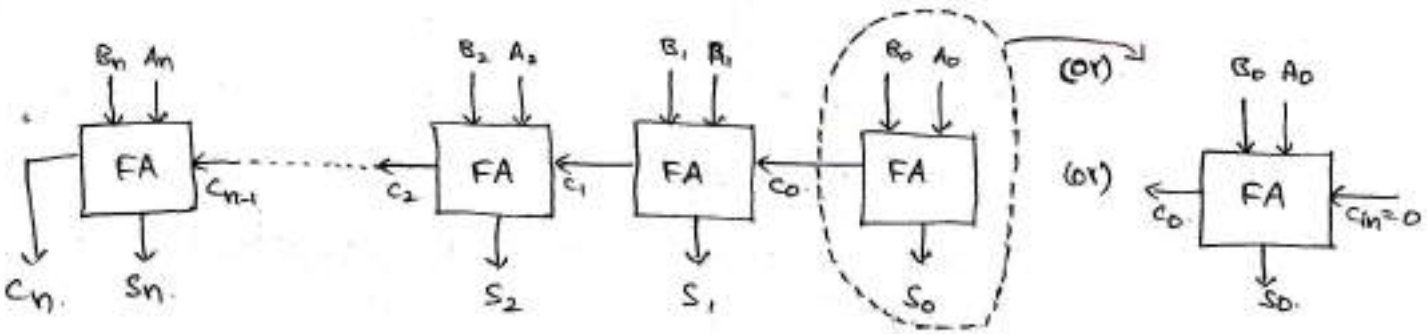
$B \longrightarrow B_n \cdots B_3 B_2 B_1 B_0$

$$\begin{array}{l}
\overset{c_{n-1}}{A_n} \cdots\cdots \overset{c_2}{A_2}\ \overset{c_1}{A_1}\ \overset{c_0}{A_0} \longrightarrow \text{Augend} \\
+\ B_n \cdots\cdots B_2\ B_1\ B_0 \longrightarrow \text{Addend} \\
\hline
c_n\ S_n \qquad\qquad S_2\ S_1\ S_0 \longrightarrow \text{Sum}.
\end{array}$$

Block diagram :-



Logic diagram :-



26/08/2014

Ex:- Design 4-bit parallel adder using full-Adders.

Sol:- A, B → 4-bit Binary numbers.

$A = A_3 A_2 A_1 A_0$

$B = B_3 B_2 B_1 B_0$

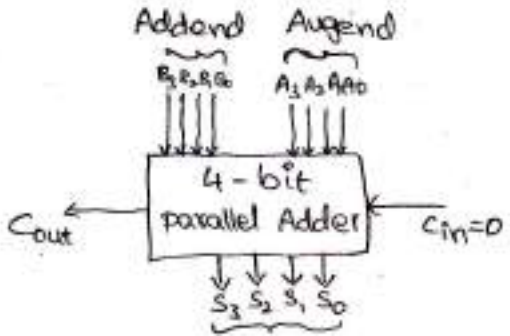
c_2 c_1 c_0 $c_{in} \rightarrow$ '0'

A_3 A_2 A_1 A_0 → Augend

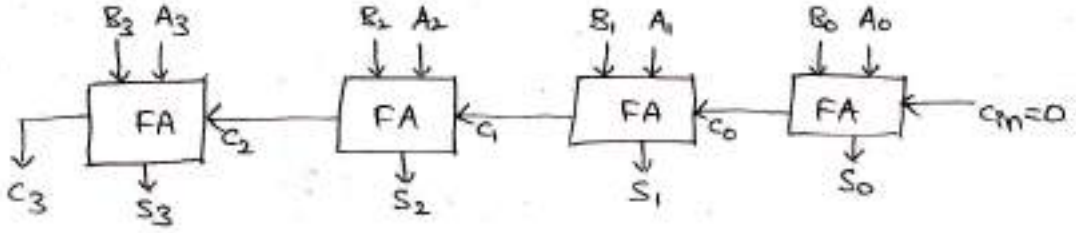
B_3 B_2 B_1 B_0 → Addend

C_3 S_3 S_2 S_1 S_0 → result.

Block diagram :-



Logic diagram :-

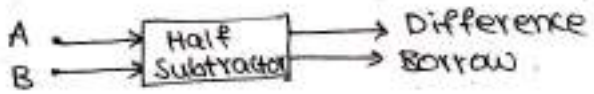


* **Subtractors:-** Subtractor is a digital circuit which performs subtraction operations.

1. **Half Subtractor :-**

It is a combinational circuit which performs subtraction of two binary bits. It's has two inputs (minuend & subtrahend) and two outputs (difference & borrow).

Block diagram :-

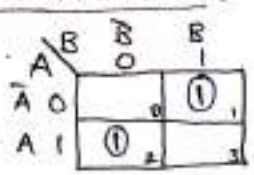


Truth table :-

i/p's		o/p's	
A	B	Diff.	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

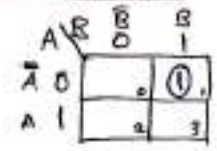
K-Map Simplification :-

for Difference :-



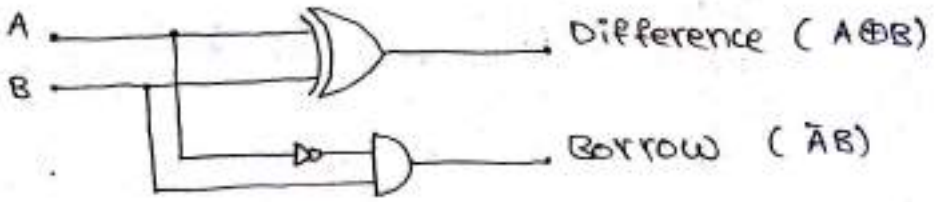
$\therefore$ Diff. $= \bar{A}B + A\bar{B}$

for Borrow :-



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

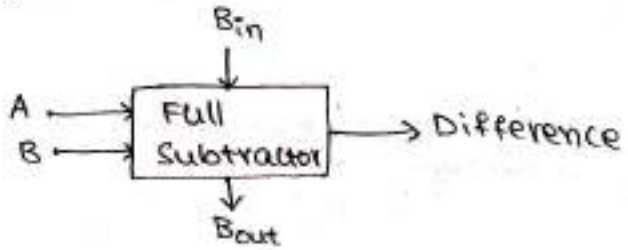
Logic circuit :-



2. Full Subtractor :-

Full Subtractor is a combinational circuit which performs subtraction of 2 binary bits by considering borrow of the previous stage. It has 3 inputs and 2 outputs.

Block diagram :-

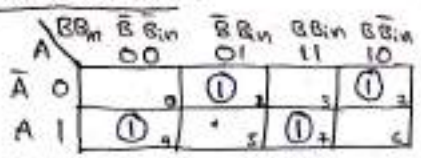


Truth table :-

i/p's			o/p's	
A	B	B_{in}	Diff	B_{out}
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 simplifications :-

For Difference :-



$\therefore$ Difference = $A \oplus B \oplus B_{in}$

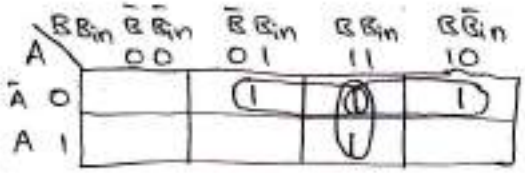
$\therefore$ Difference = $\bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$= \bar{A}(\bar{B}B_{in} + B\bar{B}_{in}) + A(\bar{B}\bar{B}_{in} + BB_{in})$

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

$= A \oplus B \oplus B_{in} = A \oplus B \oplus B_{in}$.

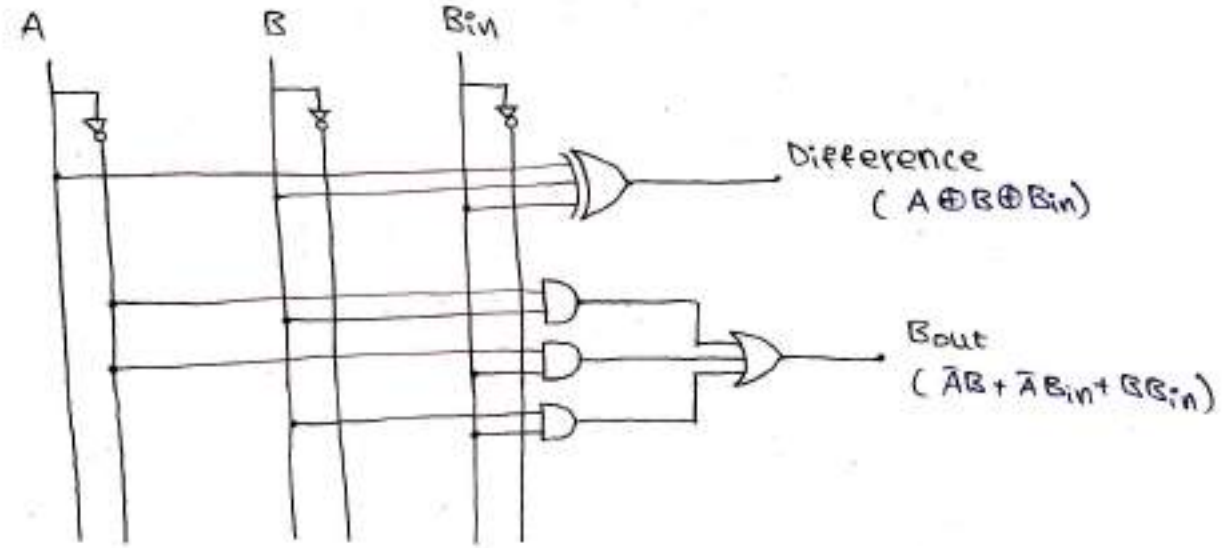
For B_{out} :-



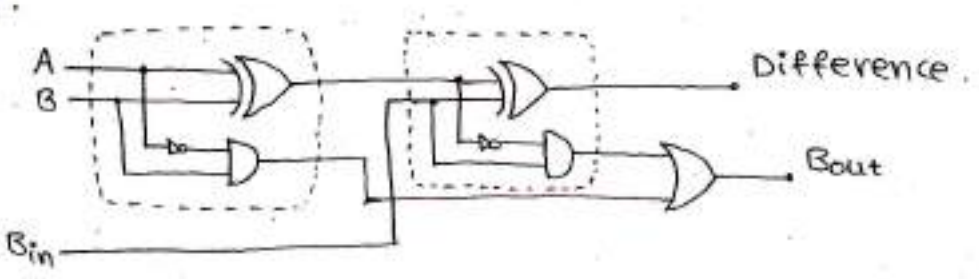
$\therefore$ $B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$

$\therefore$ $B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$

logic circuit :-



Note :- A full Subtractor can also be implemented with two half - Subtractors and 1 OR gate.



$\therefore$ Difference $= A \oplus B \oplus B_{in}$

$\therefore$ $B_{out} = (\overline{A \oplus B})B_{in} + \bar{A}B$

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

$= \bar{A}\bar{B}B_{in} + AB B_{in} + \bar{A}B$

$= \bar{A}\bar{B}B_{in} + B(AB_{in} + \bar{A})$ $\quad (\because A + BC = (A+B)(A+C))$

$= \bar{A}\bar{B}B_{in} + B((A+\bar{A})(B_{in}+\bar{A}))$ $\quad (\because A + \bar{A} = 1)$

$= \bar{A}\bar{B}B_{in} + BB_{in} + B\bar{A}$

$= B_{in}(\bar{A}\bar{B} + B) + B\bar{A}$ $\quad (\because A + BC = (A+B)(A+C))$

$= B_{in}((\bar{A}+B)(B+\bar{B})) + B\bar{A}$ $\quad (\because B + \bar{B} = 1)$

$= \bar{A}B_{in} + BB_{in} + B\bar{A}$.

Ex: Design 4-bit (Binary) parallel (or) subtractor using full adders.

Sol: A, B $\rightarrow$ 4-bit Binary no.'s.

$A = A_3\ A_2\ A_1\ A_0$

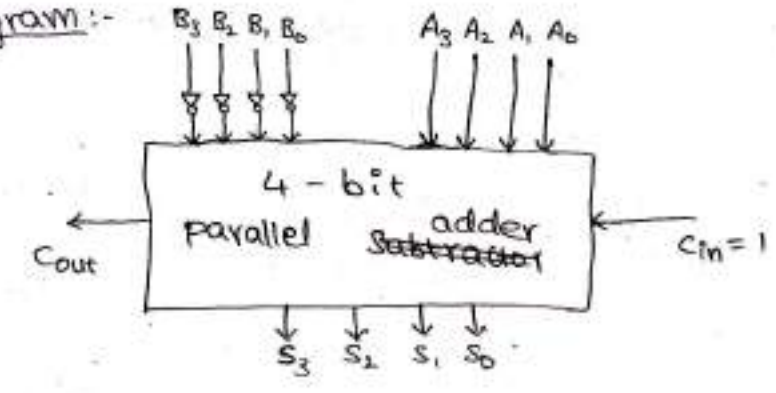
$B = B_3\ B_2\ B_1\ B_0$

$A - B =$ A + 2's complement of B

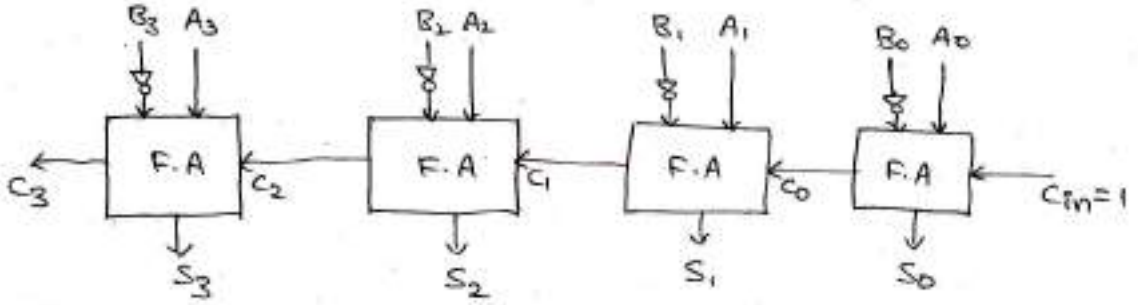
$=$ A + 1's comp. of B + 1

$$
\begin{array}{cccccl}
 & c_2 & c_1 & c_0 & & \\
 & A_3 & A_2 & A_1 & A_0 & \longrightarrow \text{Minuend} \\
 & \bar{B}_3 & B_2 & \bar{B}_1 & \bar{B}_0 & \longrightarrow \text{1's comp. of B} \\
+ & & & & 1 & \longrightarrow 1. \\
\hline
 & C_3 & S_3 & S_2 & S_1 & S_0
\end{array}
$$

Block diagram:-



Logic diagram :-



Parallel adder / Subtractor (or) Binary adder / Subtractor :-

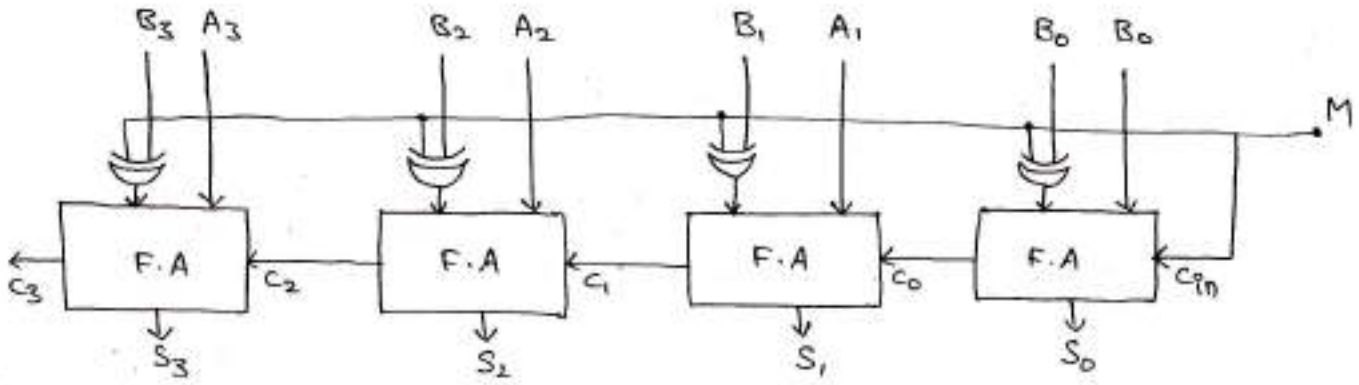


Fig:- 4-bit parallel adder / subtractor.

where M - controlling input.

if $M=0$ then output $= A_3A_2A_1A_0 + B_3B_2B_1B_0 + C_{in}$ ($\because A \oplus 0 = A$)

$= A_3A_2A_1A_0 + B_3B_2B_1B_0$ ($\because C_{in} \overset{M=0}{=} 0$)

$= A + B$.

$\therefore$ (i.e) this circuit act as adder.

If $M=1$ then output $= A_3A_2A_1A_0 + \bar{B}_3\bar{B}_2\bar{B}_1\bar{B}_0 + C_{in}$ ($\because A \oplus 1 = \bar{A}$)

$= A_3A_2A_1A_0 + \bar{B}_3\bar{B}_2\bar{B}_1\bar{B}_0 + 1$ ($\because M=1 \Rightarrow C_{in}=0$)

$= A - B$

$\therefore$ (i.e) this circuit act as Subtractor.

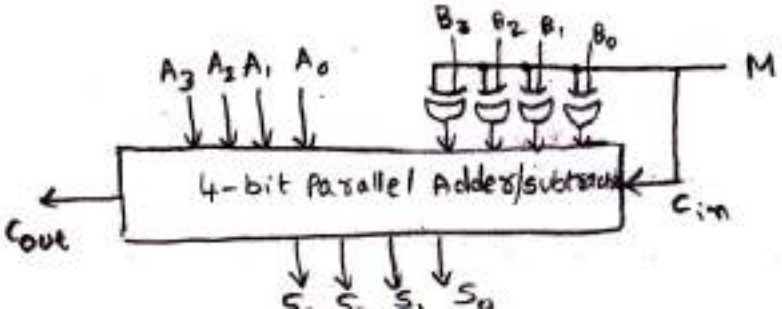


fig: Block diag

Look ahead carry generator :-

* The sum and carry outputs of any stage of n-bit parallel adder cannot be produced until the input carry occurs. This leads to time delay in addition process. This delay is known as "carry propagation delay."

* LACG is used to eliminate the carry propagation delay.

* LAC addition uses two functions.

(i) carry generate (G_i)

(ii) carry propagate (P_i)

$$\begin{array}{rrrr} & C_2 & C_1 & C_0 \\ & A_2 & A_1 & A_0 \\ + & B_2 & B_1 & B_0 \\ \hline C_3 & S_2 & S_1 & S_0 \end{array}$$

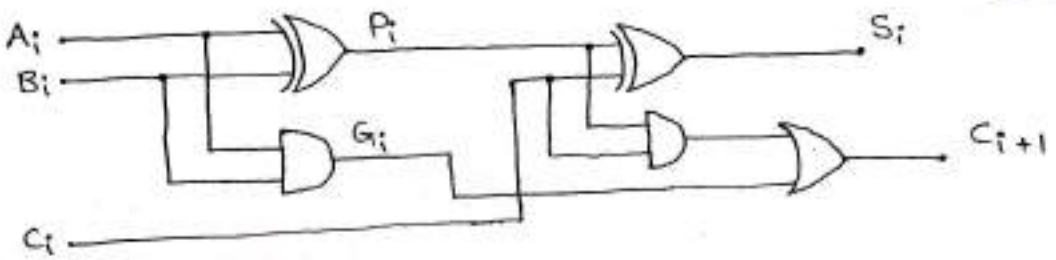


Fig:- FA using 2 HA's and OR gate with P & G shown.

$$\therefore \quad P_i = A_i \oplus B_i$$

$$G_i = A_i \cdot B_i$$

using P & G, Sum & carry can be expressed as

$$S_i = P_i \oplus C_i$$

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

$\therefore$ C_0 = i/p carry.

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$

$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

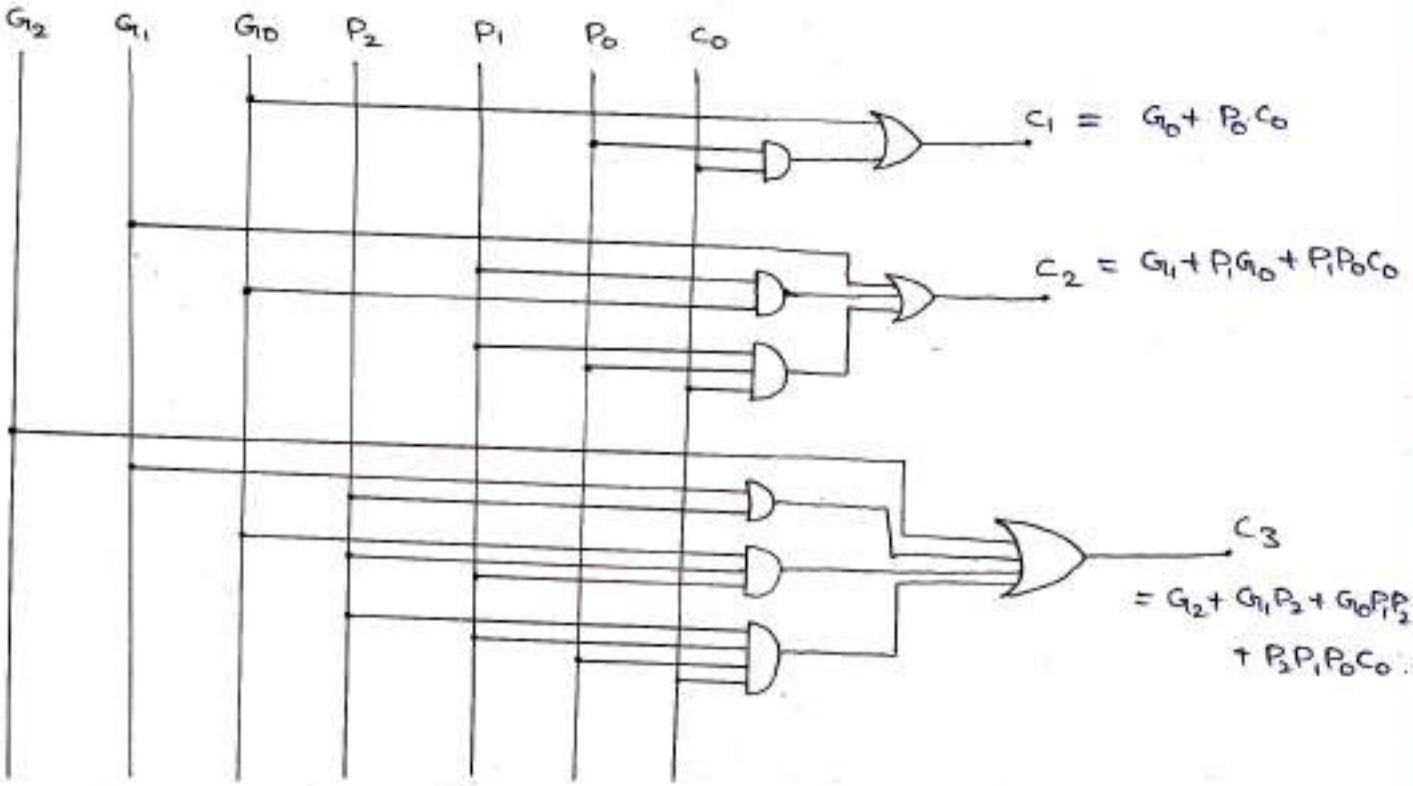
$$\therefore C_3 = G_2 + P_2 C_2$$

$$= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$$

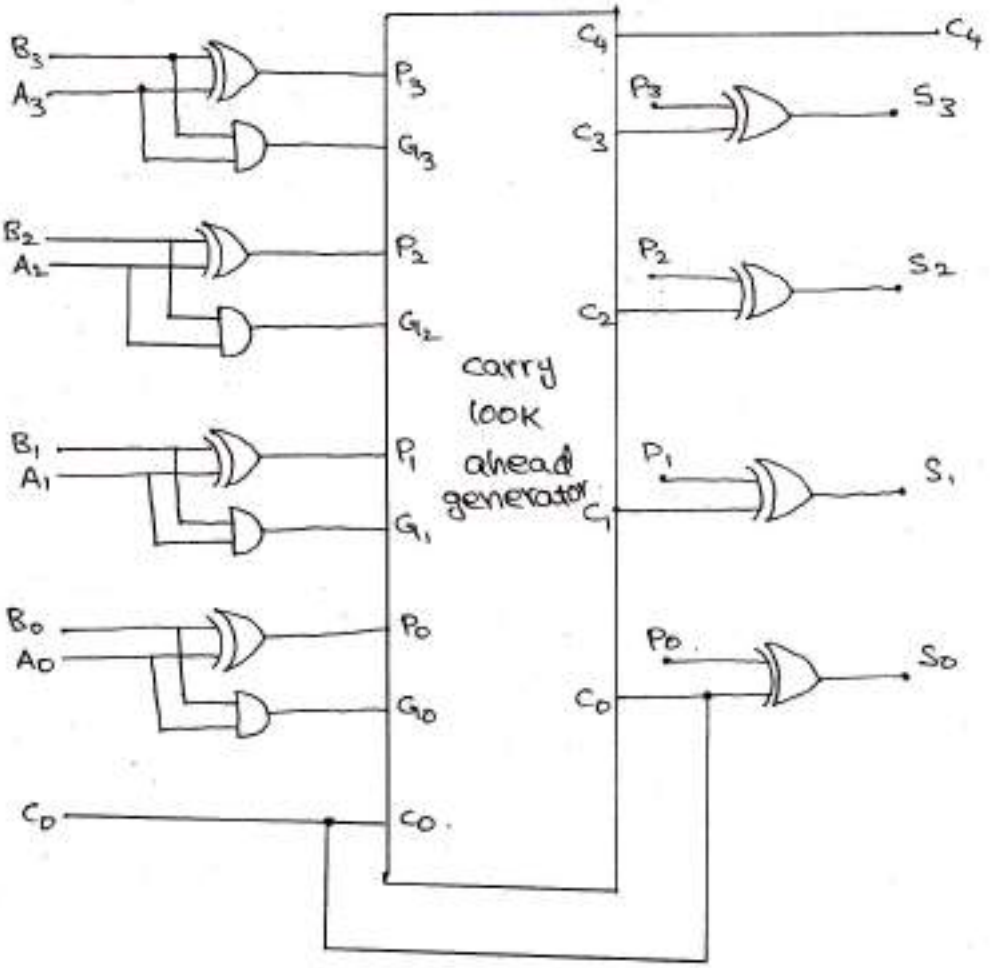
$$= G_2 + G_1 P_2 + G_0 P_1 P_2 + P_2 P_1 P_0 C_0$$

From the above boolean equations (Expressions) it can be seen that C_1, C_2, C_3 propagate at the same time and depends only on C_0.

3-bit look ahead carry generator logic gen diagram :-



4-bit adder with carry look ahead :-



* BCD adder / Decimal adder :-

* The digital systems handles the decimal no. in the form of BCD number.

* A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD.

* To implement BCD adder we required.

(i) 4-bit binary adder for initial Addition.

(ii) logic circuit to detect sum is greater then 9 (or) not.

(iii) 1 more 4-bit binary adder to add $(0110)_2$

to the sum if sum is >9 or carry is 1.

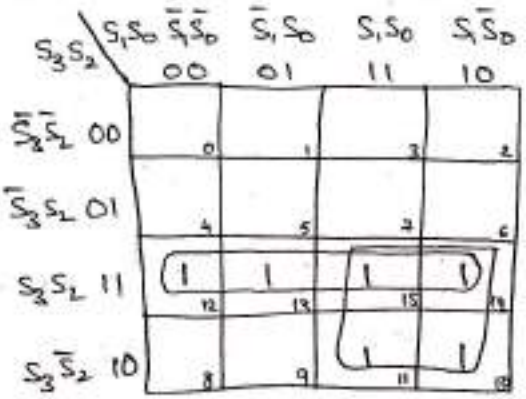
Truth table to implement logic circuit to detect Sum is >9 or not :-

i/p's				o/p
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

o/p is '1' when i/p is >9.

K-map simplification :-

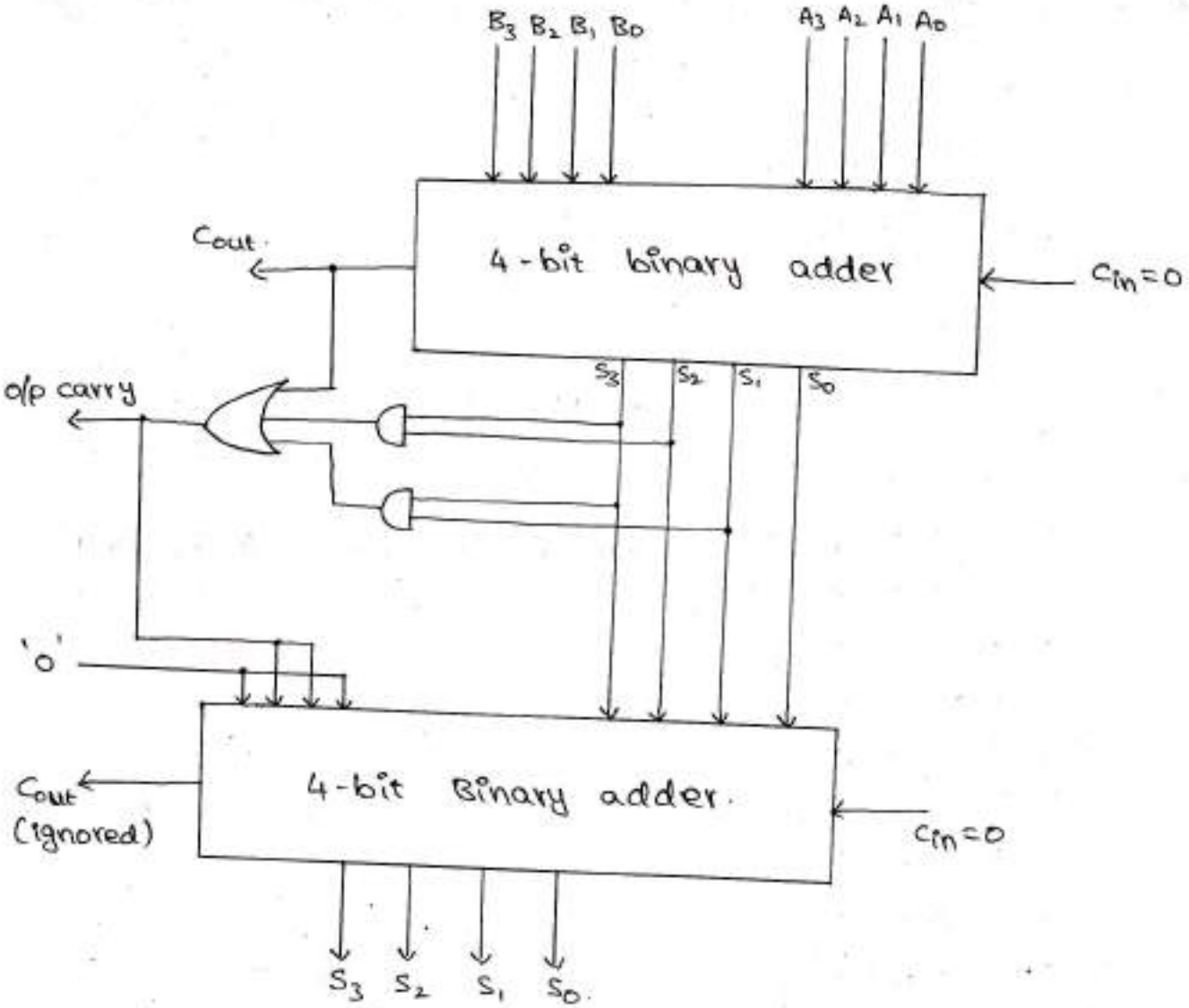
for y :-



$\therefore \quad Y = S_3S_2 + S_3S_1$

Q1

logic diagram :-



* 4-bit Binary Incrementer :-

$A = A_3 A_2 A_1 A_0$ 4-bit binary no.

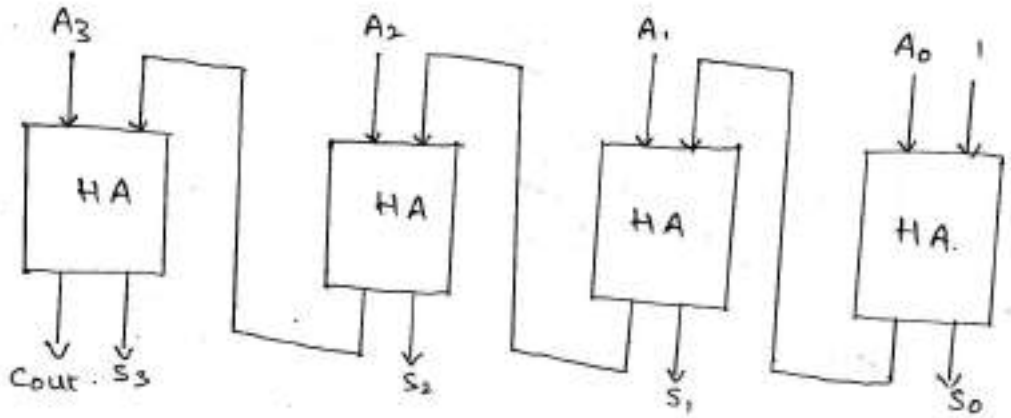
$A + 1 = A_3 A_2 A_1 A_0 + 1$

$C_2\ C_1\ C_0$

$A_3\ A_2\ A_1\ A_0$

$+\quad\quad\quad\quad 1$

$C_{out}\ S_3\ S_2\ S_1\ S_0$



* 4-bit Binary Decrementer :-

$A = A_3 A_2 A_1 A_0$ 4-bit Binary number.

$A - 1 = A_3 A_2 A_1 A_0 - 0001$

$= A_3 A_2 A_1 A_0 +$ 2's comp. of 0001

$= A_3 A_2 A_1 A_0 +$ 1's comp. of 0001 + 1

$= A_3 A_2 A_1 A_0 +$ 1110 + 1

** Binary Multiplier :-

(i) 2X2 Binary Multiplier :-

$B = B_1 B_0$ $A = A_1 B_0$ 2-bit binary no.'s.

$B \times A \Rightarrow$

$B_1 \; B_0 \longrightarrow$ Multiplicand

$\times A_1 \; A_0 \longrightarrow$ Multiplier

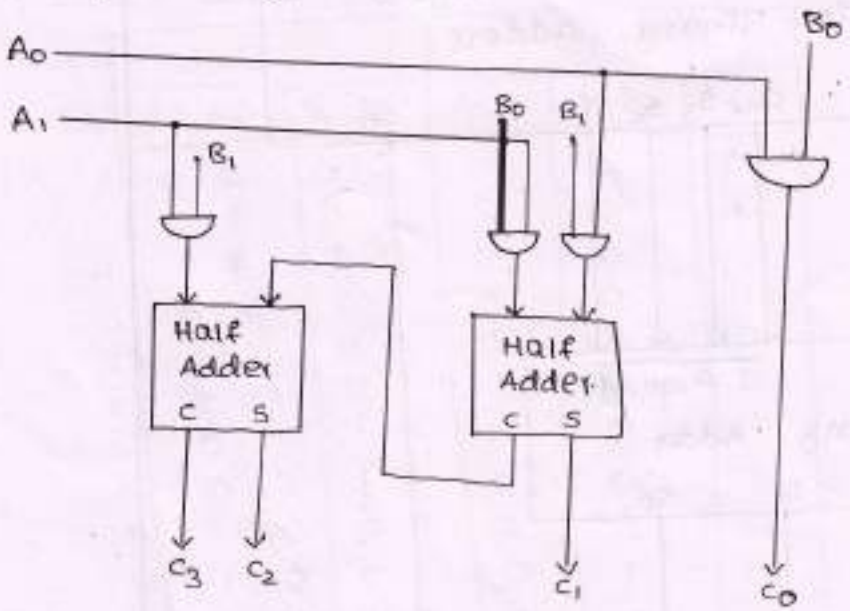
$A_0 B_1 \quad A_0 B_0$ } Partial products.

$A_1 B_1 \quad A_1 B_0$

$C_3 \; C_2 \; C_1 \; C_0 \longrightarrow$ Product

(B X A = Simple multiplication)

logic diagram :-



(ii) 4×3 Binary Multiplier :-

$B = B_3\ B_2\ B_1\ B_0$ → 4-bit Binary no.

$A = A_2\ A_1\ A_0$ → 3-bit Binary no.

B X A

$B_3\ B_2\ B_1\ B_0$ ⟶ Multiplicand

$\times\ A_2\ A_1\ A_0$ ⟶ Multiplier

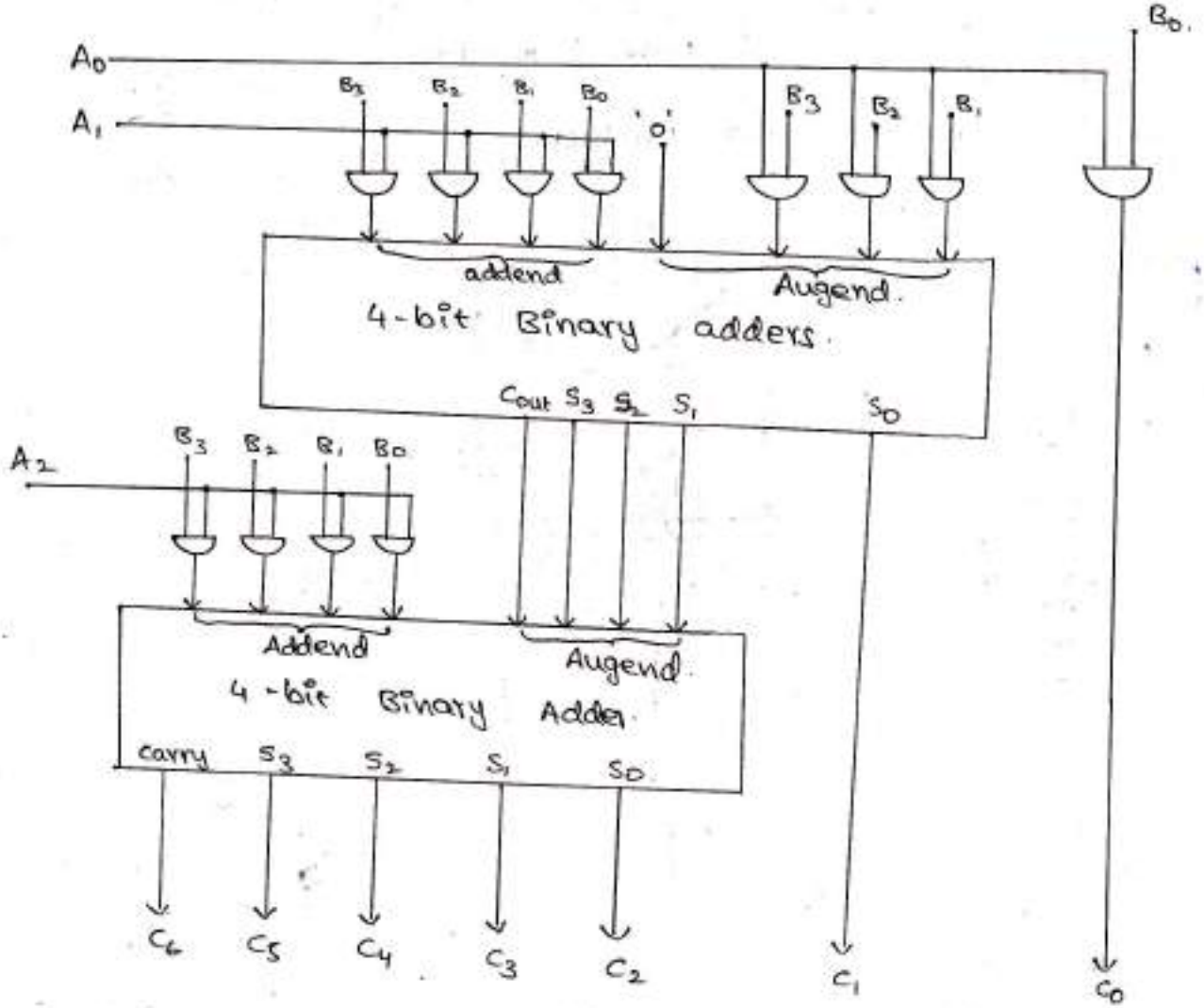
$A_0B_3\ \ A_0B_2\ \ A_0B_1\ \ A_0B_0$

$A_1B_3\ \ A_1B_2\ \ A_1B_1\ \ A_1B_0$ } partial products.

$A_2B_3\ \ A_2B_2\ \ A_2B_1\ \ A_2B_0$

$C_6\ C_5\ C_4\ C_3\ C_2\ C_1\ C_0$ → Product

Logic diagram :-



16/08/2014

* Magnitude comparator (Digital comparator) :-

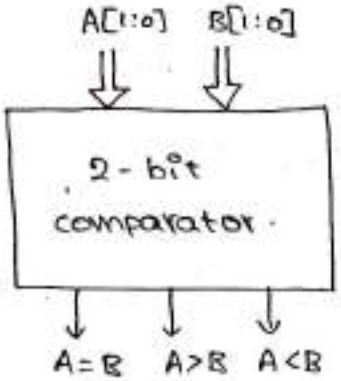
* A comparator is a special combinational circuit designed primarily to compare the relative magnitudes of two Binary numbers.

* It receives two n-bit numbers, A & B are inputs and produces 3 outputs A>B, A<B, A=B.

Design 2-bit comparator using logic gates.

2-bit comparator compares 2, 2-bit binary numbers.

Block diagram:-



A, B are 2 bit Binary numbers.

Truth table:-

	i/p's				o/p's		
	A_1	A_0	B_1	B_0	A=B	A>B	A<B
0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	0	1	0
5	0	1	0	1	1	0	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	0	1	0
9	1	0	0	1	0	1	0
10	1	0	1	0	1	0	0
11	1	0	1	1	0	0	1
12	1	1	0	0	0	1	0
13	1	1	0	1	0	1	0
14	1	1	1	0	0	1	0
16	1	1	1	1	1	0	0

$(A=B) = \sum m(0,5,10,15)$

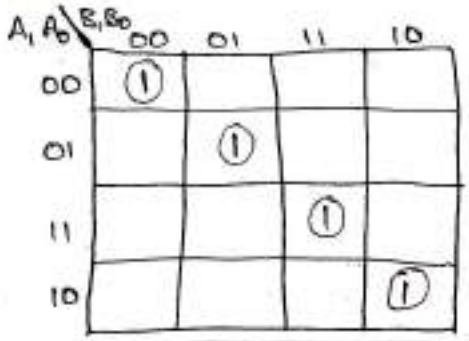
$(A>B) = \sum m(4,8,9,12,13,14)$

$(A<B) = \sum m(1,2,3,6,7,11)$

K-map simplification :-

For A=B :-

$(A=B) = \sum m(0,5,10,15)$



$(A=B) = \bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$

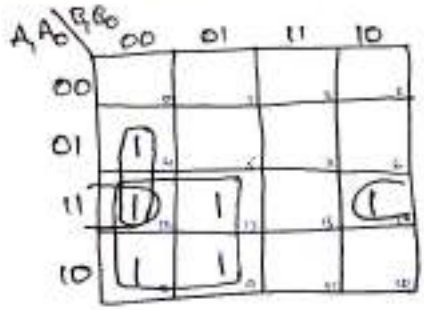
$= \bar{A}_0 \bar{B}_0 (\bar{A}_1 \bar{B}_1 + A_1 B_1) + A_0 B_0 (\bar{A}_1 \bar{B}_1 + A_1 B_1)$

$= (\bar{A}_1 \bar{B}_1 + A_1 B_1)(\bar{A}_0 \bar{B}_0 + A_0 B_0)$.

$\therefore \boxed{(A=B) = (A_0 \odot B_0)(A_1 \odot B_1)}$

For (A>B) :-

$(A>B) = \sum m(4,8,9,12,13,14)$

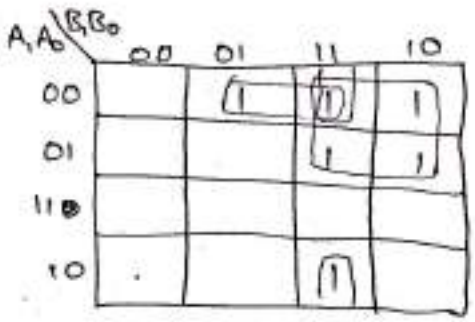


$(A>B) = A_1 \bar{B}_1 + \bar{B}_1 \bar{B}_0 A_0 + A_1 A_0 \bar{B}_0$

$\therefore \boxed{(A>B) = A_1 \bar{B}_1 + A_1 A_0 \bar{B}_0 + A_0 \bar{B}_1 \bar{B}_0}$

For (A<B) :-

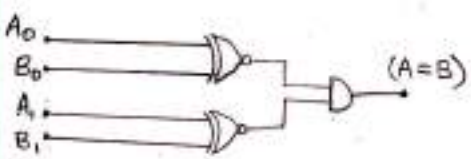
$(A<B) = \sum m(1,2,3,6,7,11)$

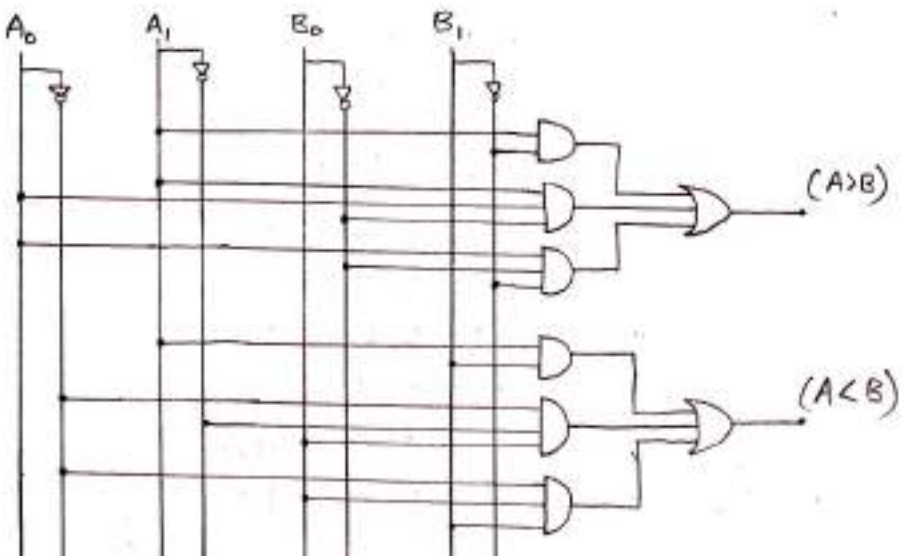


$(A<B) = \bar{A}_1 B_1 + B_0 \bar{A}_1 \bar{A}_0 + B_1 B_0 \bar{A}_0$

$\therefore \boxed{(A<B) = \bar{A}_1 B_1 + \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_1 B_0}$

Logic circuit :-





* **Decoder** :- A Decoder is a multiple input, multiple o/p circuit, which converts coded inputs into coded outputs, where the i/p & o/p codes are different.

* the i/p code generally has fewer bits than them o/p codes.

logic diagram :-

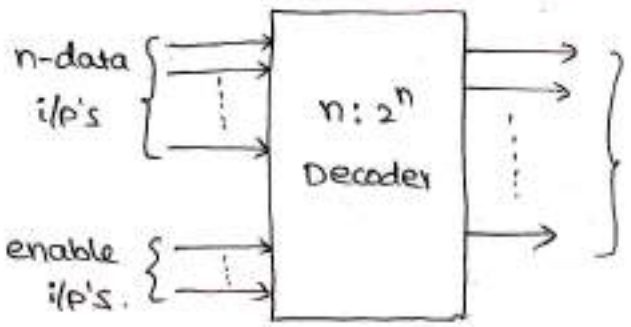
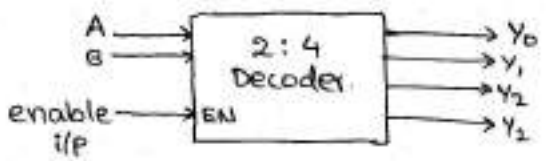


Fig:- General Structure of Decoder.

(i) 2 - to - 4 (2:4) Binary Decoder :-.

Block diagram :-



Truth table :-

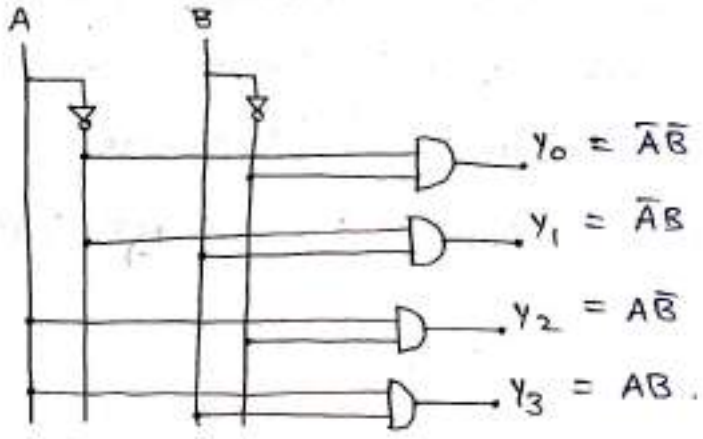
i/p's			o/p's.			
EN	A	B	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(∵ enable i/p is '0' then there is no i/p's.)

from the above truth table, $Y_0 = \bar{A}\bar{B}$, $Y_1 = \bar{A}B$ (∵ k-map Simplification)

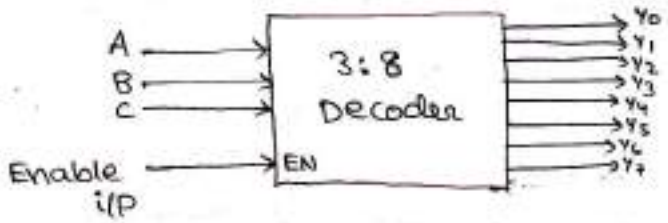
$Y_2 = A\bar{B}$, $Y_3 = AB$

Logic diagram :-



(ii) 3-to-8 (3:8) Binary decoder :-

Block diagram :-



Truth table :-

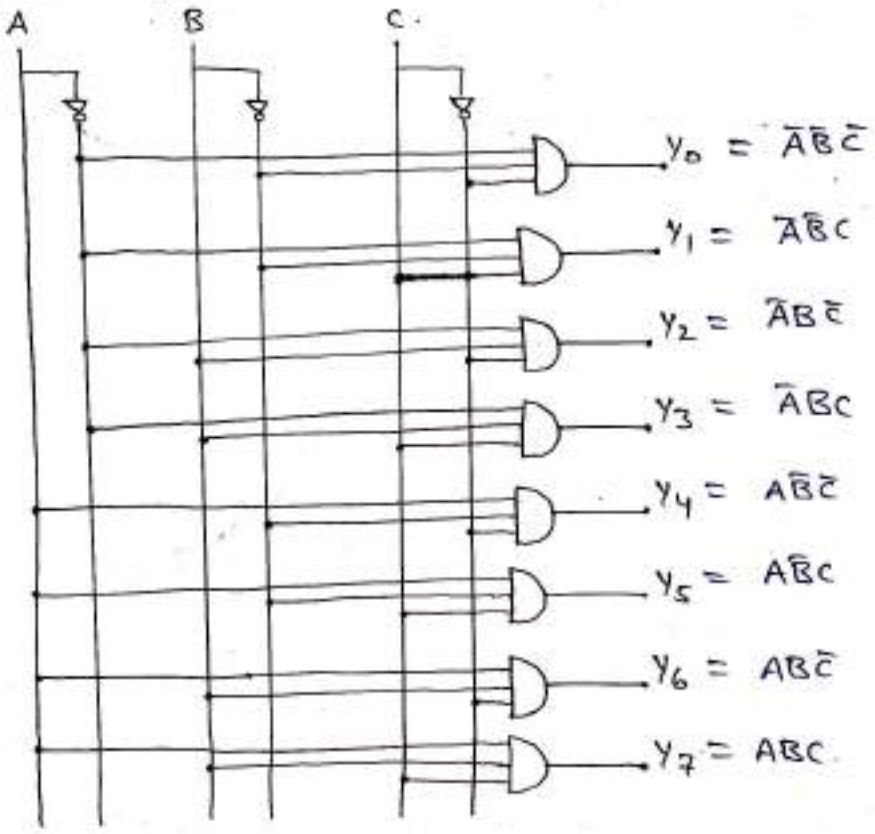
i/p's				o/p's							
EN	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

From the above truth table (using k-map simplification)

$Y_0 = \bar{A}\bar{B}\bar{C}$, $Y_1 = \bar{A}\bar{B}C$, $Y_2 = \bar{A}B\bar{C}$, $Y_3 = \bar{A}BC$

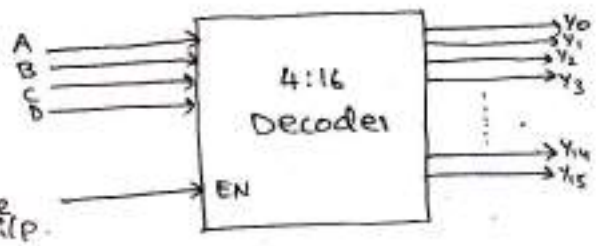
$Y_4 = A\bar{B}\bar{C}$, $Y_5 = A\bar{B}C$, $Y_6 = AB\bar{C}$, $Y_7 = ABC$

logic diagram :-



Ex:- construct 4:16 decoders using 3:8 decoders.

Sol:- Block diagram of 4:16 decoder :-



Truth table of 4:16 decoder :-

i/p's				o/p
A	B	C	D	Y (High)
0	0	0	0	Y_0
0	0	0	1	Y_1
0	0	1	0	Y_2
0	0	1	1	Y_3
0	1	0	0	Y_4
0	1	0	1	Y_5
0	1	1	0	Y_6
0	1	1	1	Y_7
1	0	0	0	Y_8
1	0	0	1	Y_9
1	0	1	0	Y_{10}
1	0	1	1	⋮
1	1	0	0	

4:16 Decoder using 3:8 Decoders :-

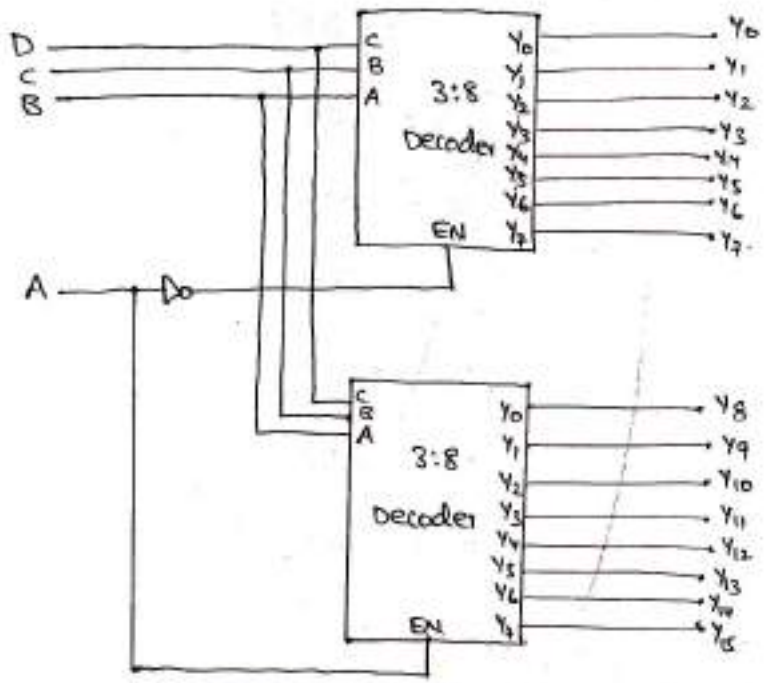
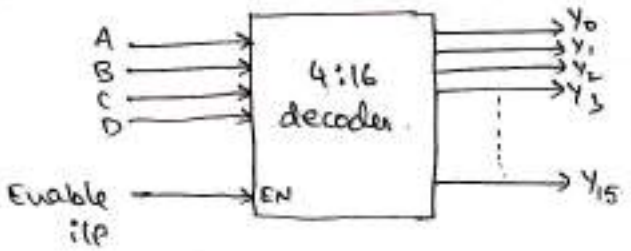


Fig:- 4:16 decoder using 3:8 decoders.

Ex:- Construct 4:16 Decoder using 2:4 decoder.

Sol:-

Block diagram of 4:16 decoder :-



Truth table of 4:16 decoder :-

i/p's				o/p
A	B	C	D	Y(High)
0	0	0	0	Y_0
0	0	0	1	Y_1
0	0	1	0	Y_2
0	0	1	1	Y_3
0	1	0	0	Y_4
0	1	0	1	Y_5
		⋮		⋮
1	1	1	0	Y_{14}
1	1	1	1	Y_{15}

4:16 decoder using 2:4 decoders :-

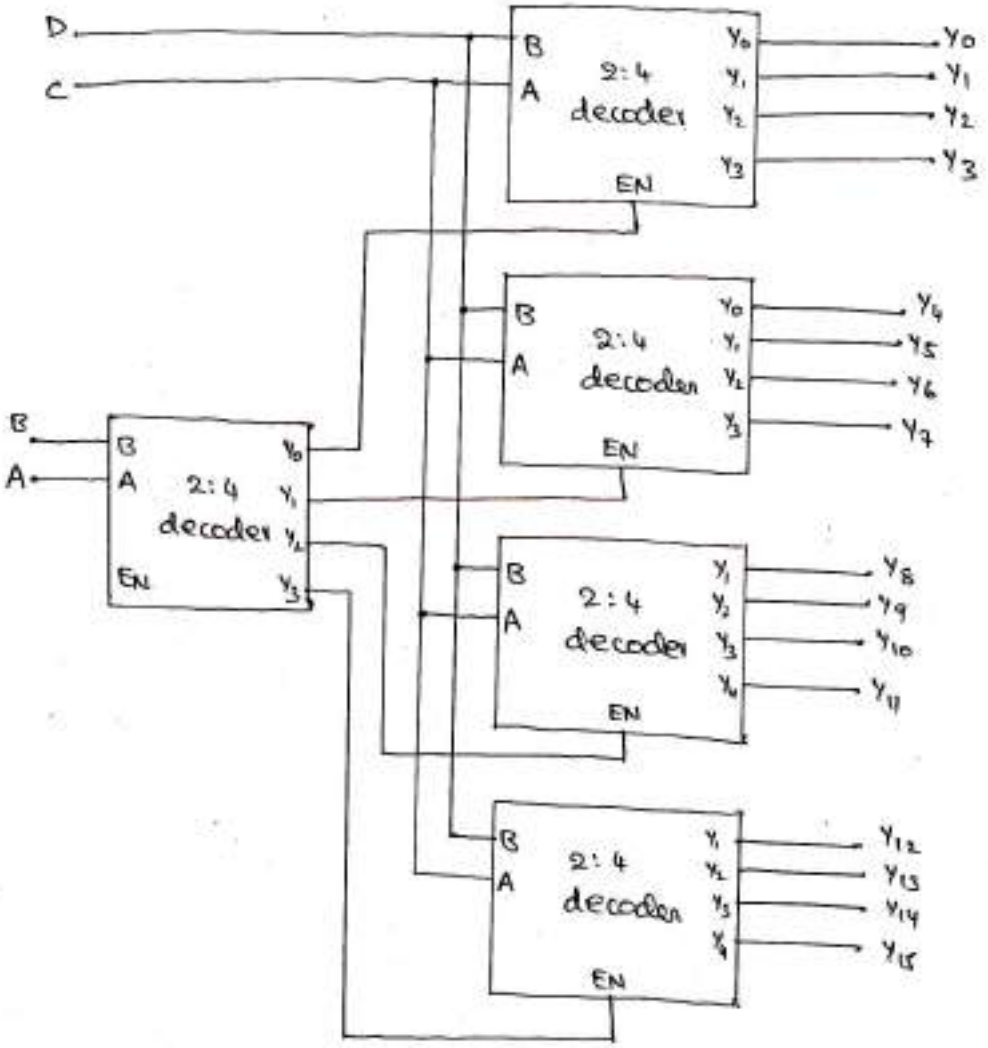
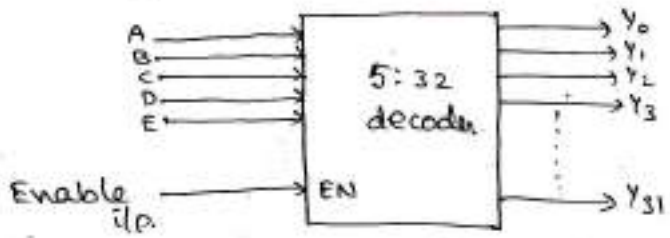


Fig:- 4:16 decoder using 2:4 decoders.

Ex:- construct 5:32 decoder using 4, 3:8 decoders & 1, 2:4 decoder.

Sol.

Block diagram of 5:32 decoder :-



Truth table of 5:32 decoder :-

i/p's					o/p
A	B	C	D	E	Y (High)
0	0	0	0	0	Y_0
0	0	0	0	1	Y_1
0	0	0	1	0	Y_2
⋮					⋮
1	1	1	1	0	Y_{30}
1	1	1	1	1	Y_{31}

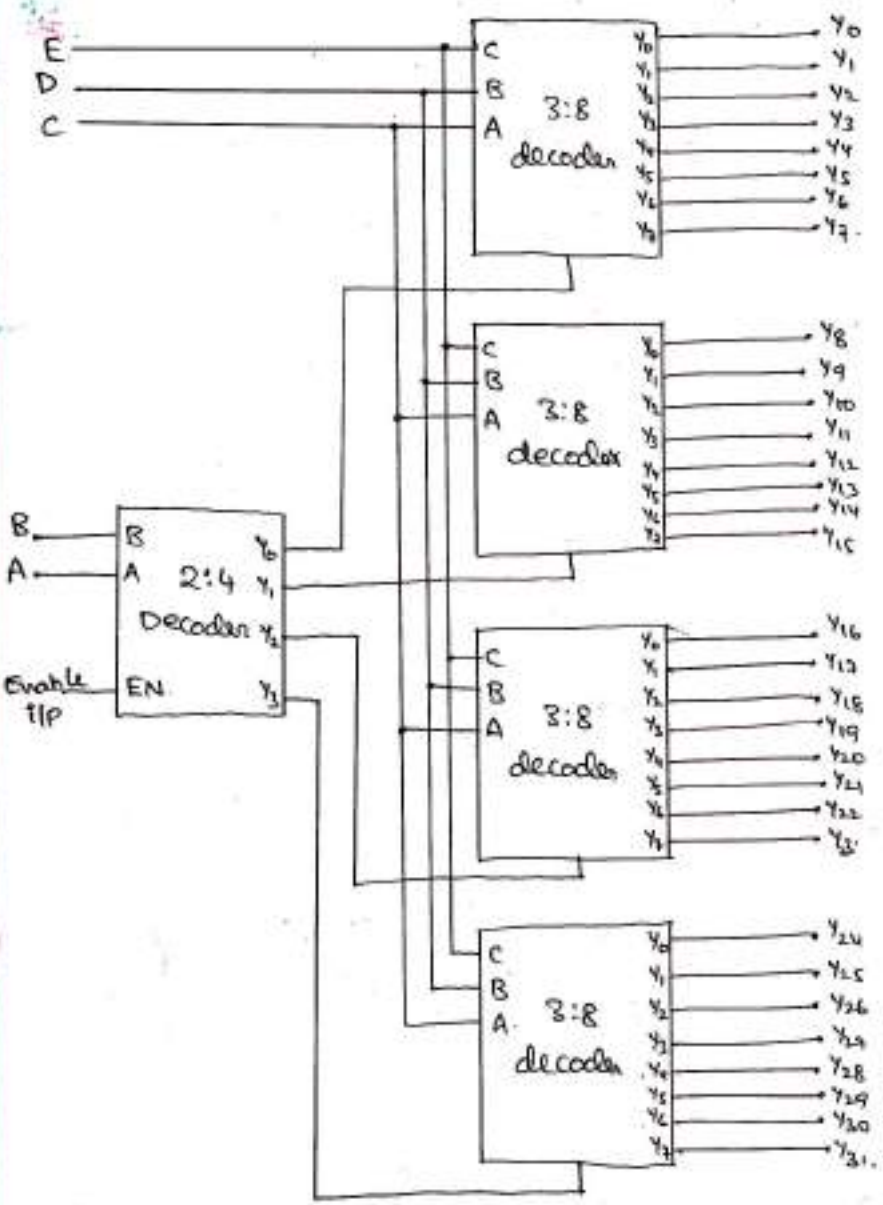


Fig:- 5:32 decoder using 4, 3:8 decoders & 1, 2:4 decoders.

* Realization of Boolean Expressions using Binary decoder:-

→ the combination of decoder and external logic gates can be used to implement single or multiple output functions.

→ The decoder generates minterms for input variables. thus by logically oring specified minterms we can implement the given function.

Ex:-

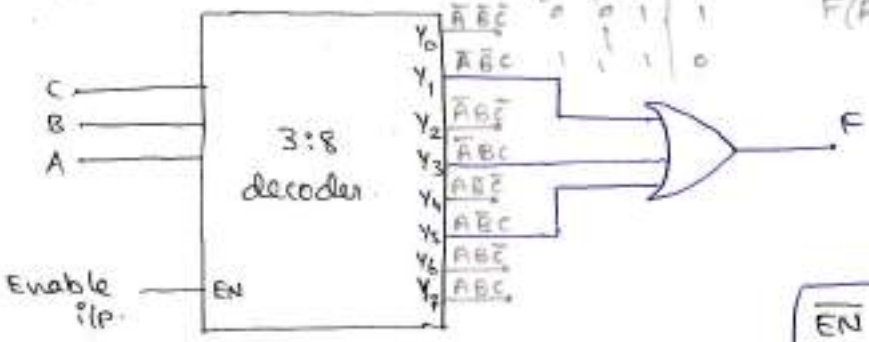
Ex:- $F(A,B,C) = \Sigma m(1,3,5)$ Implement by using decoder

for SM write TRUTH table

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

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

Sol:-

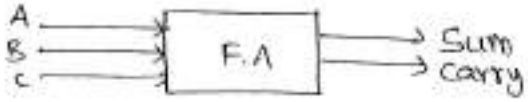


$\overline{EN} = 0 \rightarrow$ Active

$\overline{EN} = 1 \rightarrow$ disable

Ex:- Implement full adder circuit using decoder.

Sol:- Block diagram of full adder :-



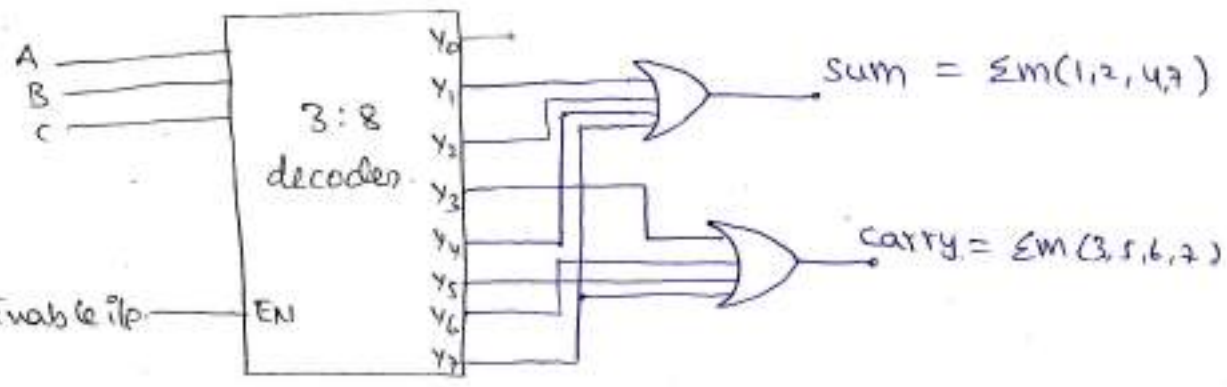
Truthtable of full adder:-

i/p's			o/p's	
A	B	C	sum	carry
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

$\therefore$ Sum $= \Sigma m(1,2,4,7)$

carry $= \Sigma m(3,5,6,7) = \Sigma m(3,5,6,7)$

Full adder using Decoder & external gates :-

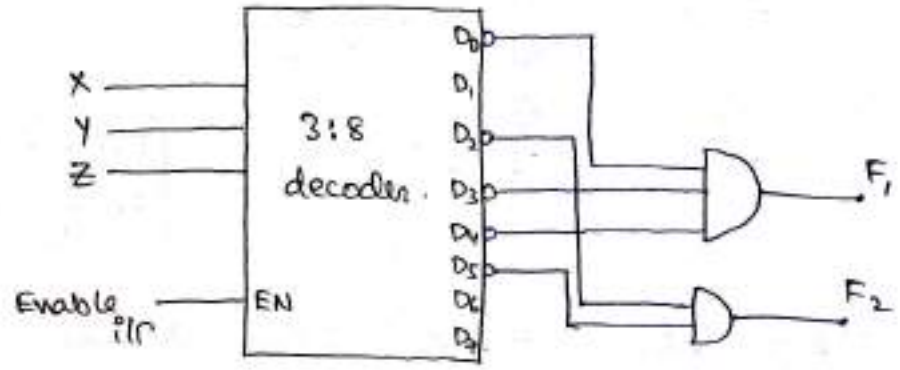


Ex:- Implement the following boolean expressions using decoder.

Sol:-

$F_1(X,Y,Z) = \pi M(0,3,4)$

$F_2(X,Y,Z) = \pi M(2,5)$



from the above circuit,

$F_1 = \pi M(0,3,4)$

$F_2 = \pi M(2,5)$.

02/09/2014

* BCD to 7-Segment Display Decoder :-

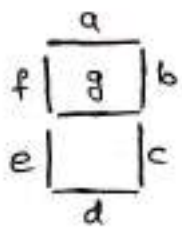


fig:- 7-segment display.

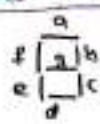
→ there are two types of 7-segment displays.

(i) common anode → Segment i/p must be active low to glow the segment.

(ii) common cathode → segment i/p must be active high to glow the segment.

→ most commonly used one is common cathode type.

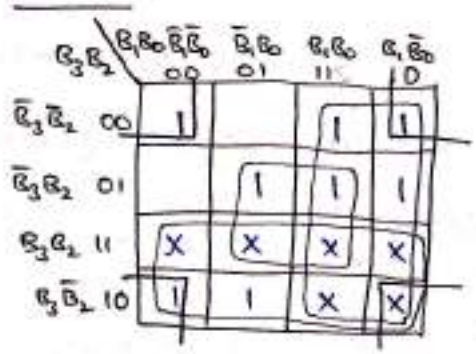
Truth table of common cathode 7-segment display decoder:-



decimal	i/p (BCD)				o/p							
	B_3	B_2	B_1	B_0	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	0	1	1	1	9

K-Map Simplification

for a :-



$\therefore \quad a = B_3 + B_2B_0 + B_1 + \bar{B}_2\bar{B}_0$

Similarly, $b =$

$c =$

$d =$

$e =$

$f =$

$g =$

Logic diagram:

* ENCODER :-

An encoder is a digital circuit that performs the inverse operation of the decoder. It has "2^n" (or) fewer) input lines and 'n' output lines. the o/p lines generate the binary code corresponding to the i/p value.

Block diagram :-

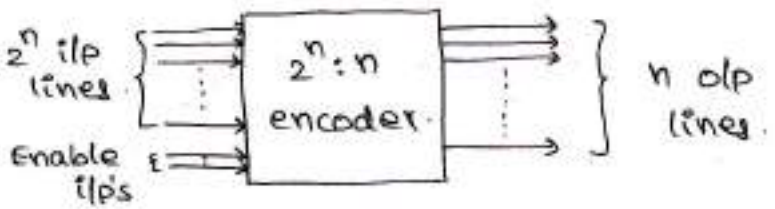


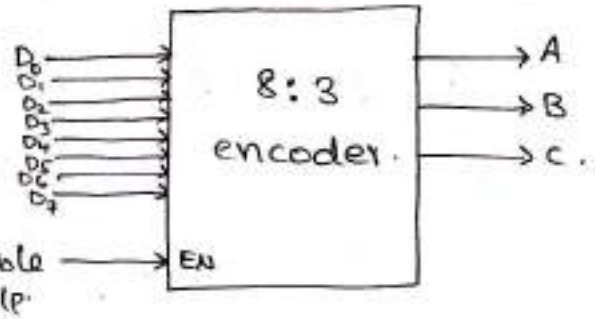
Fig:- General structure of Encoder.

Octal to Binary Encoder :- (8-to-3) (8:3 Encoder)

→ It has '8' inputs. (i.e) 1 for each octal digit and 3 o/p's that generate the corresponding Binary code.

→ In Encoders it is assumed that only one i/p has a value of '1' at any given time, otherwise the circuit is meaningless.

Block diagram :-



Truth table (or) Function table :-

	i/p's								o/p's		
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
	1	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	1
	0	0	1	0	0	0	0	0	0	1	0
	0	0	0	1	0	0	0	0	0	1	1
	0	0	0	0	1	0	0	0	1	0	0
	0	0	0	0	0	1	0	0	1	0	1
	0	0	0	0	0	0	0	0	1	1	0
	0	0	0	0	0	0	0	1	1	1	1
EN = 0	0	0	0	0	0	0	0	0	X	X	X

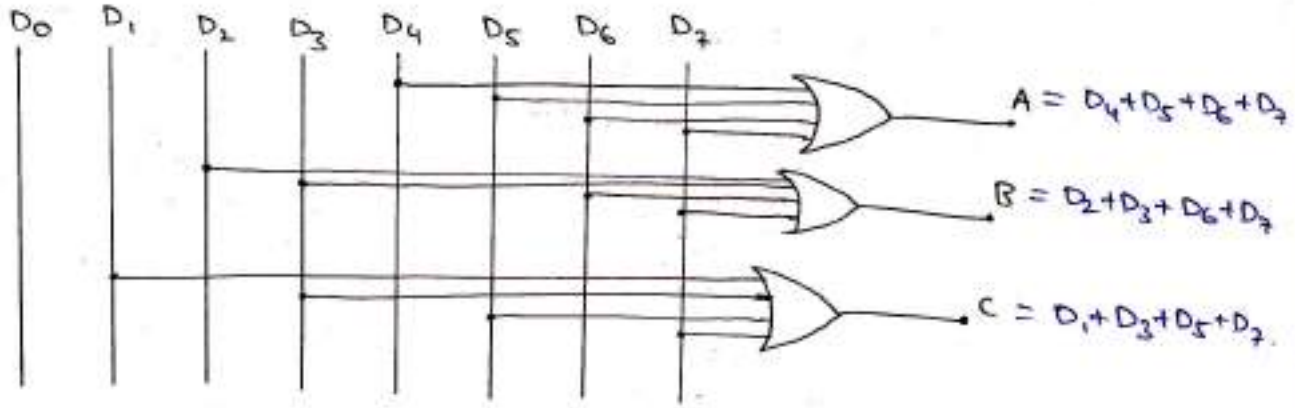
→ if Enable i/p is '0' there is no o/p.

$\therefore$ $A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

Logic diagram :-



* Priority Encoder :- (4-bit priority encoder)

A priority encoder is a encoder circuit that includes the priority function. In priority encoder, if two or more i/p's are equal to '1' at the same time, the i/p having the highest priority will take precedance.

Truth table:-

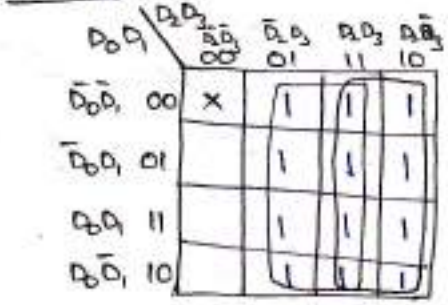
i/p's				o/p's		
D_0	D_1	D_2	D_3	Y_1	Y_0	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

→ D_3 i/p has highest priority and D_0 i/p has lowest priority. when D_3 i/p is high, regardless of other i/p's output is 11.

→ 'V' is a valid bit indicator, that is set to '1' when 1 or more i/p's are equal to '1'.

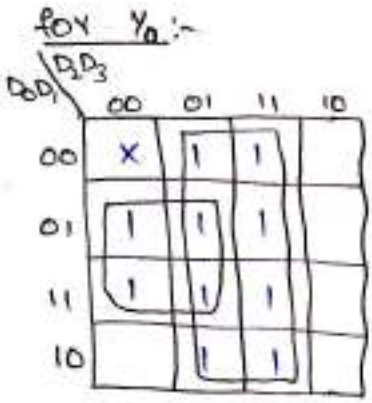
k-Map simplifications:-

for Y_1 :-



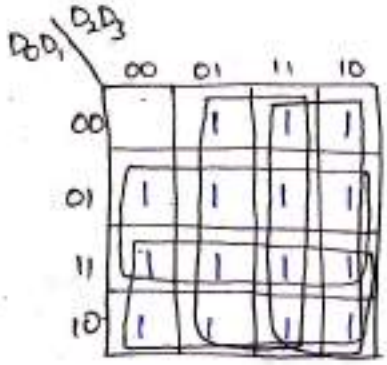
$\therefore \quad Y_1 = D_2 + D_3$

for Y_0 :-



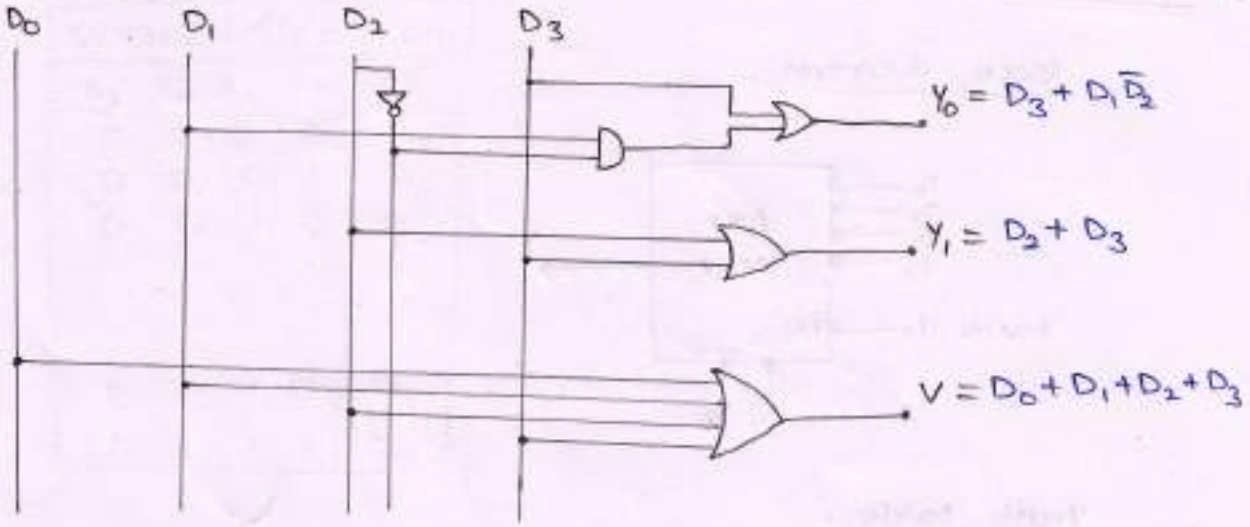
$\therefore \quad Y_0 = D_3 + D_1\bar{D}_2$

for V :-



$\therefore \quad V = D_0 + D_1 + D_2 + D_3$

logic circuit :-



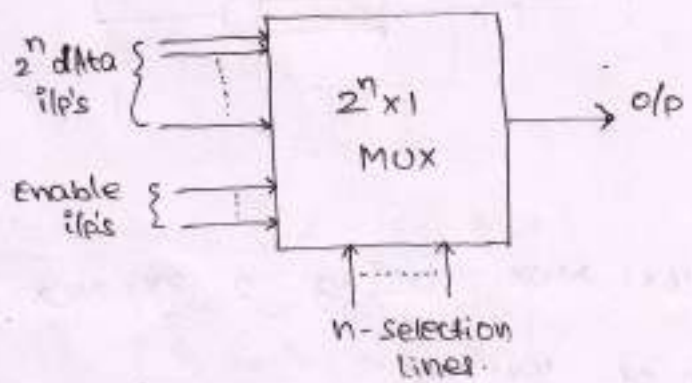
* Multiplexer (or) Data selector :-

→ Multiplexer is a combinational logic circuit that selects binary information from one of many input lines and directs it to a single o/p line.

→ the selection of particular i/p line is controled by a set of selection lines.

→ MUX has 2^n i/p lines, n selection lines & one o/p.

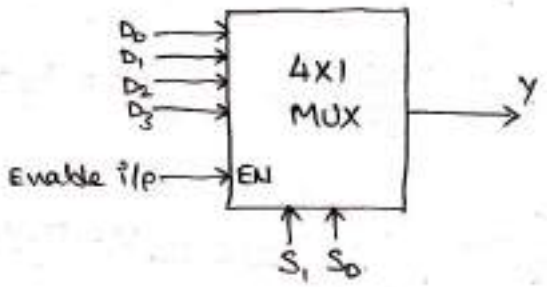
Block diagram :-



03/09/2014

4x1 MUX :-

Block diagram :-

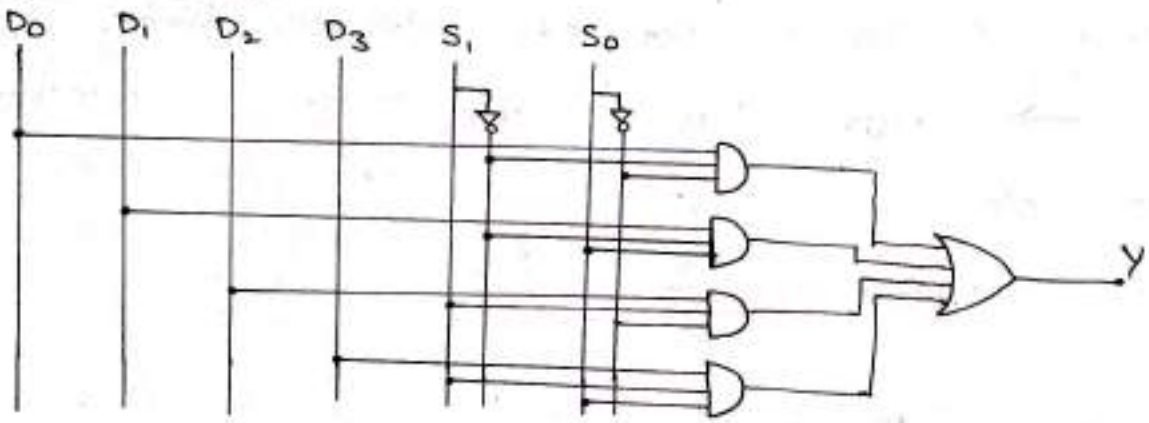


Truth table :-

	i/p's		o/p
EN	S_1	S_0	Y
0	x	x	x
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

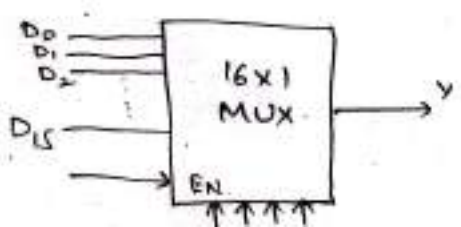
$\therefore Y = D_0 \cdot \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 \cdot S_1 S_0$

logic circuit :-



Ex:- Construct a 16x1 MUX using 2, 8x1 MUX & 1, 2x1 MUX.

Sol:- Block diagram of 16x1 MUX :-



Truth table :-

selection i/p's				o/p
S_3	S_2	S_1	S_0	Y
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
:	:	:	:	:
1	1	1	0	D_{14}
1	1	1	1	D_{15}

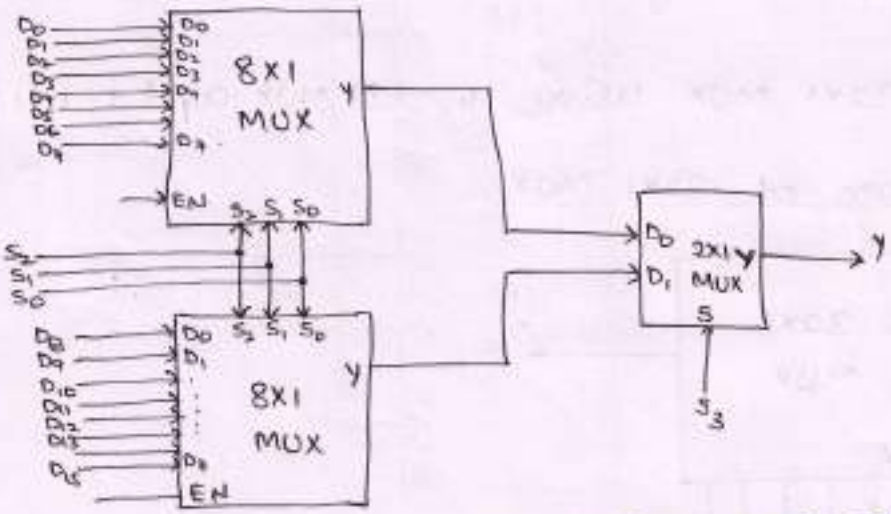
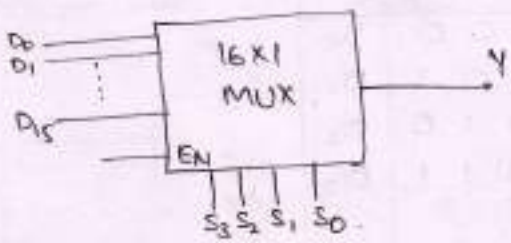


Fig:- 16X1 MUX using two 8X1 MUX & one 2X1 MUX.

Ex:- construct a 16X1 MUX using two 8X1 MUX and OR gate.

Sol:- Block diagram of 16X1 MUX :-



Truth table of 16X1 MUX :-

selection i/p's				o/p
S_3	S_2	S_1	S_0	Y
0	0	0	0	D_0
0	0	0	1	D_1
:	:	:	:	:
1	1	1	0	D_{14}
1	1	1	1	D_{15}

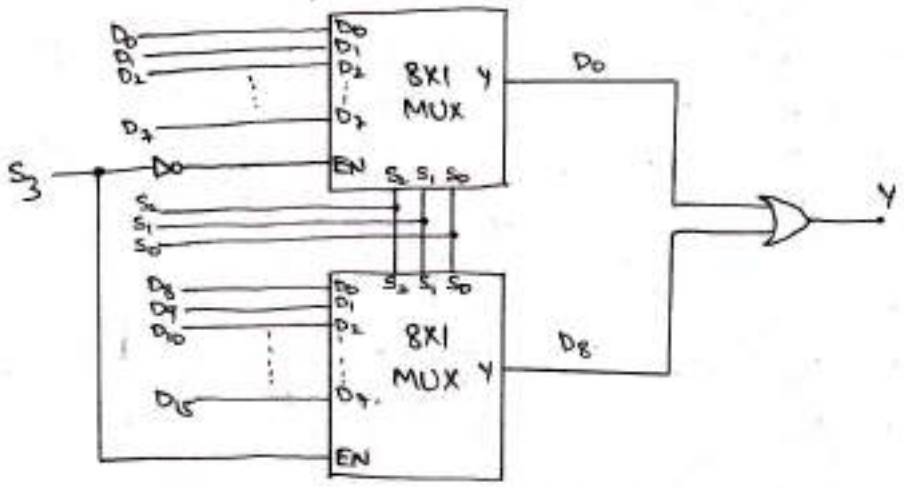
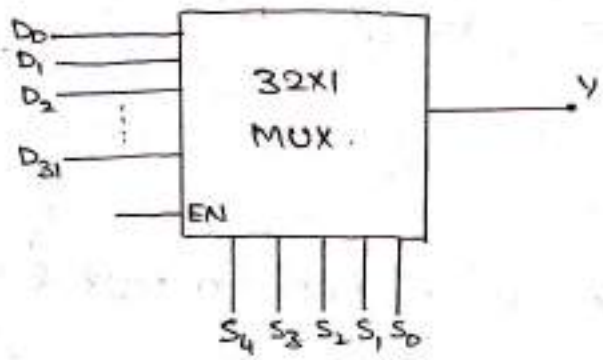


fig:- 16X1 MUX using two 8X1 MUX and one OR gate.

Ex:- construct a 32X1 MUX using 4, 8X1 MUX and 1, 4X1 MUX.

Sol:- Block diagram of 32X1 MUX:-



Truth table of 32X1 MUX:-

selection i/p's					o/p
S_4	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	D_0
0	0	0	0	1	D_1
0	0	0	1	0	D_2
0	0	0	1	1	D_3
					.
					.
					.
					.
					.
					.
1	1	1	0	1	D_{29}
1	1	1	1	0	D_{30}
1	1	1	1	1	D_{31}

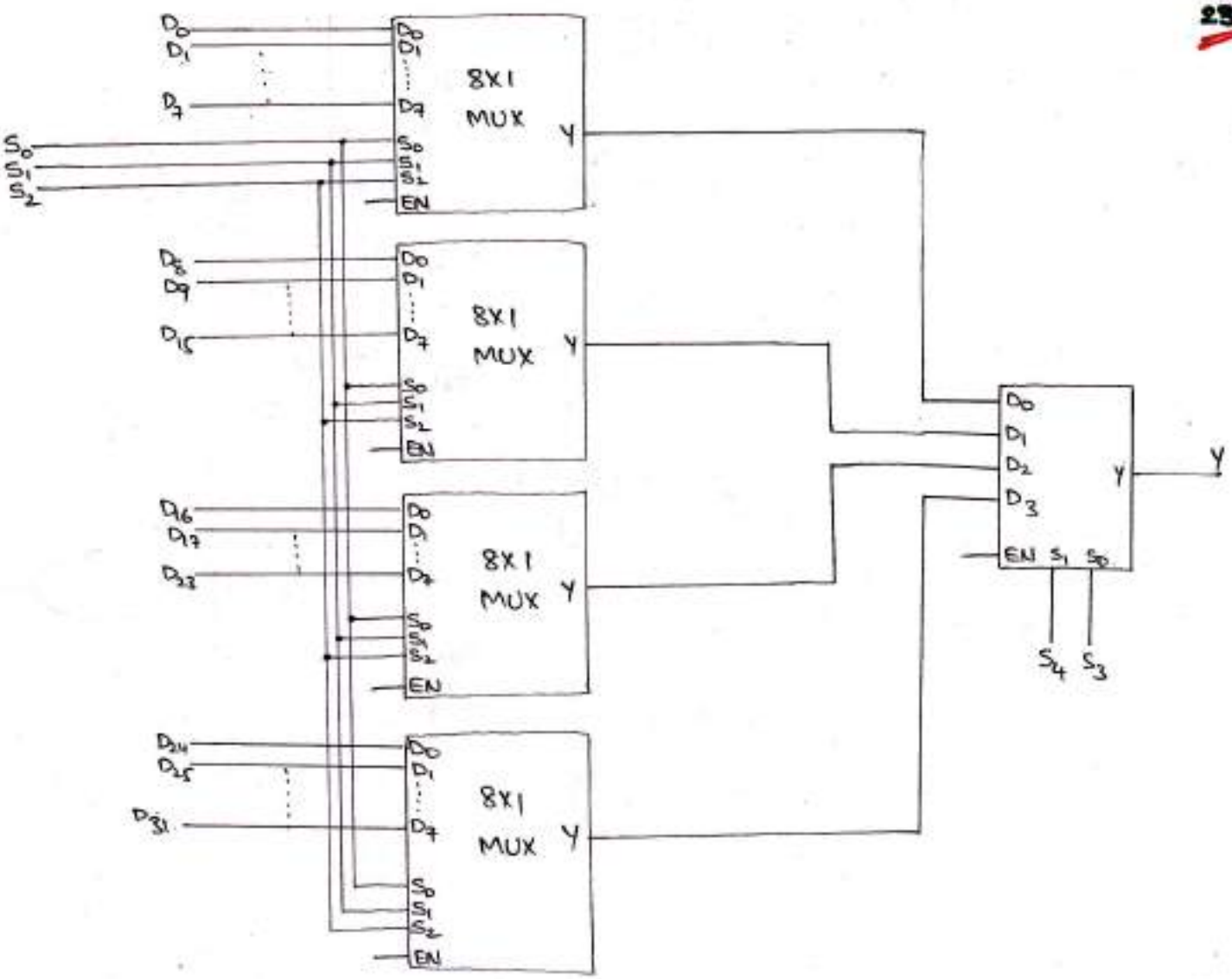
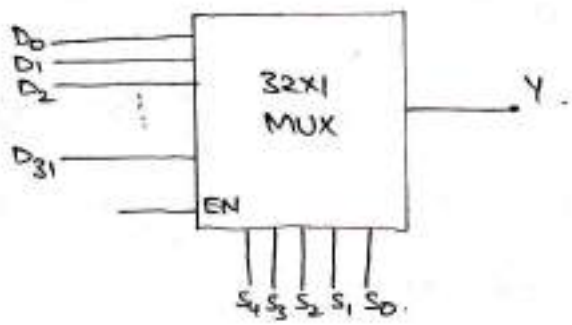


Fig:- 32X1 MUX using four 8X1 MUX & one 4X1 MUX.

Ex:- Construct 32X1 MUX 4. 8X1 MUX & 1. 2-to-4 decoder, 1 OR-gate.

Sol:- Block diagram of 32X1 MUX :-



Truth table of 32X1 MUX :-

selection i/p's					o/p
S_4	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	D_0
0	0	0	0	1	D_1
		⋮			⋮
1	1	1	1	0	D_{30}

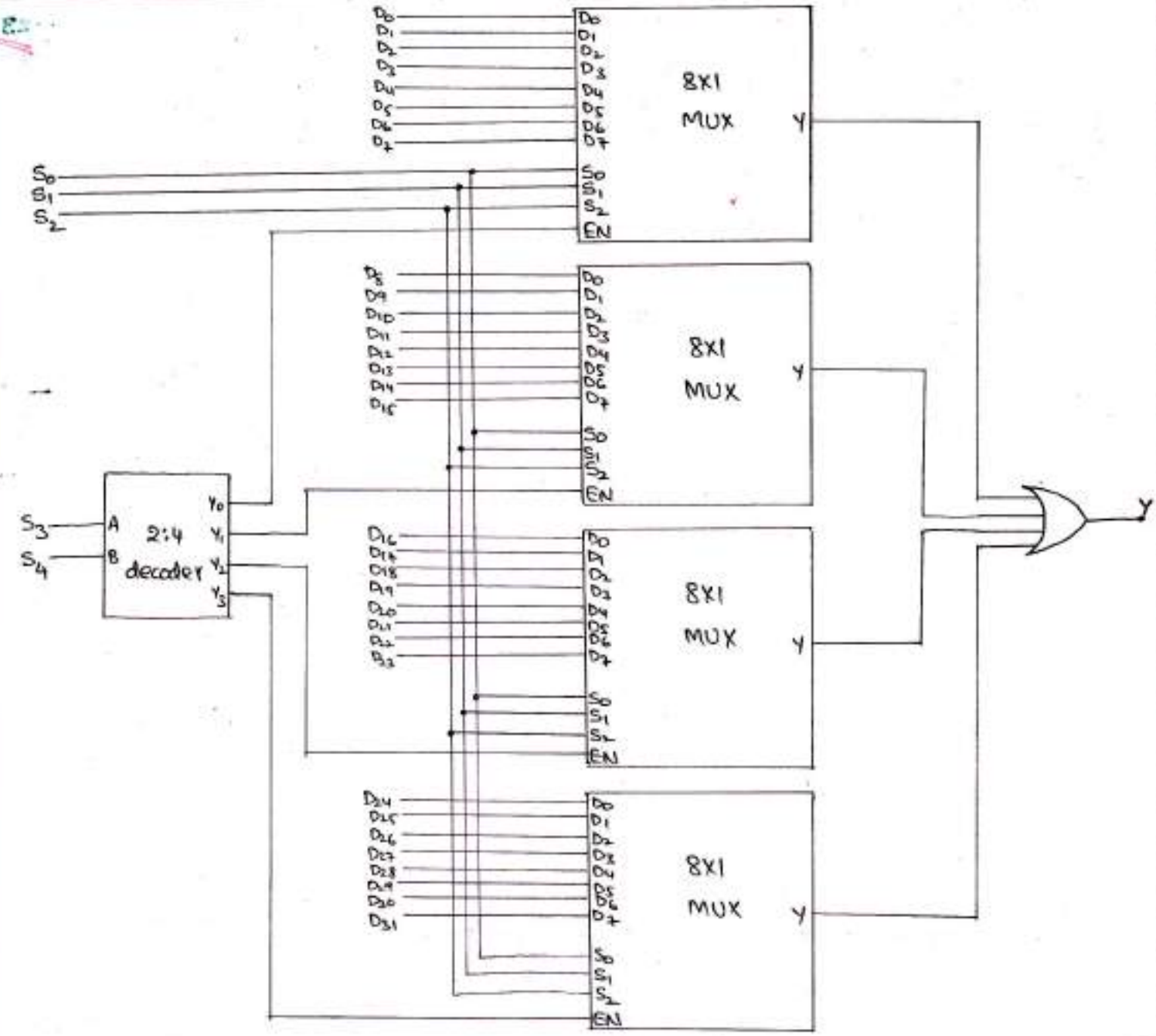


Fig:- 32x1 MUX using four 8x1 MUX, one 2:4 decoder & one OR-gate.

* Implementation of combinational logic using Multiplexer:-

Ex:- Implement the following boolean function using

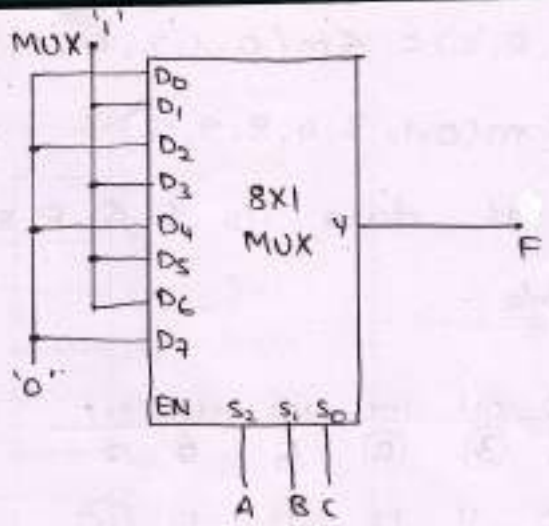
(a) 8x1 MUX (b) 4x1 MUX

(i) $F(A,B,C) = \sum m(1,3,5,6)$

Sol:- Truth table:-

I/P's			o/p
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1

(i) using 8X1 MUX

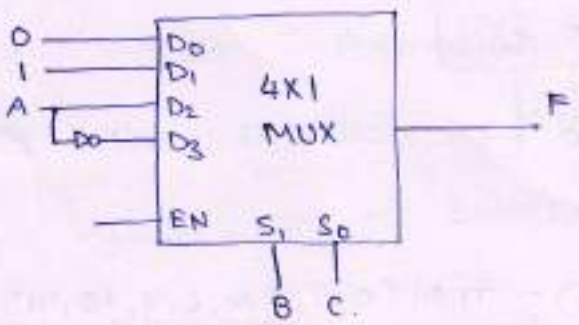


(ii) using 4X1 MUX

→ considering 'A' as data i/p & B,C as selection i/p's.

implementation table :-

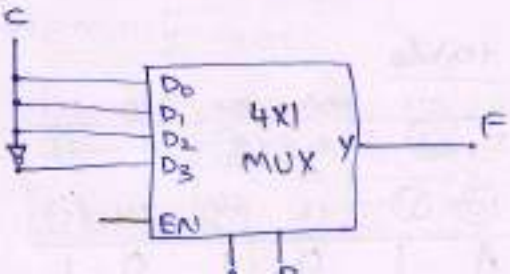
A	BC	D_0 $B\bar{C}$ 00	D_1 $\bar{B}C$ 01	D_2 $B\bar{C}$ 10	D_3 BC 11
$\bar{A}$	0	0	①	2	③
A	1	4	⑤	⑥	7
		0	1	A	$\bar{A}$



→ considering 'c' as data i/p & A,B as selection i/p's.

implementation table :-

C	AB	D_0 $\bar{A}\bar{B}$ 00	D_1 $\bar{A}B$ 01	D_2 $A\bar{B}$ 10	D_3 AB 11
$\bar{C}$	0	0	2	4	⑥
C	1	①	③	⑤	7
		C	C	C	$\bar{C}$



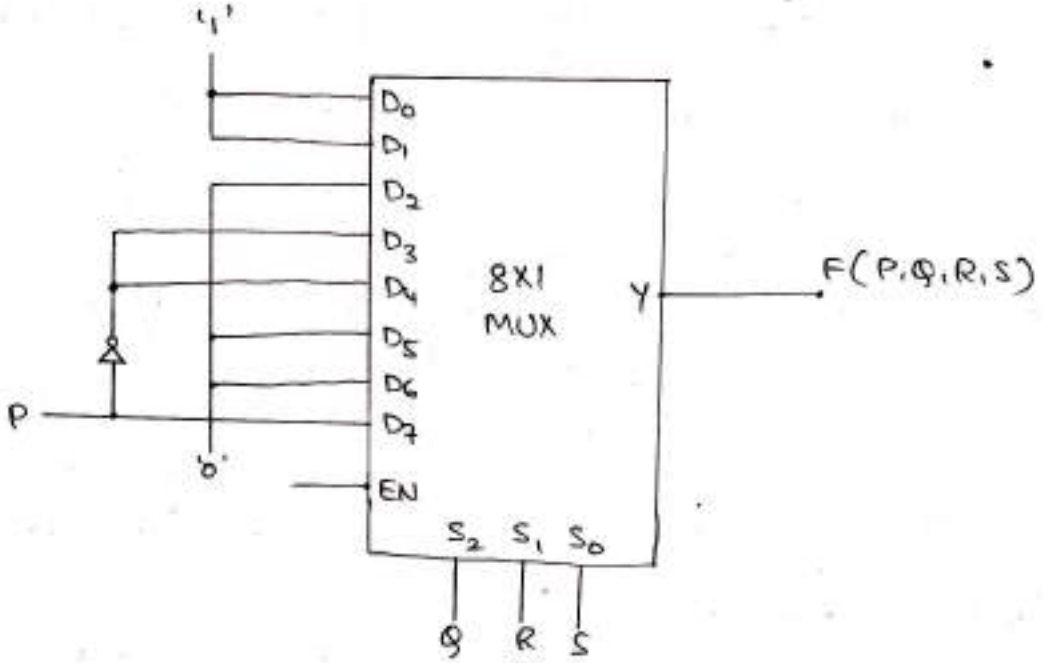
Ex: Implement $F(P,Q,R,S) = \Sigma m(0,1,3,4,8,9,15)$ using 8x1 MUX.

Sol:- $F(P,Q,R,S) = \Sigma m(0,1,3,4,8,9,15)$.

considering 'P' as data i/p & Q, R, S as Selection i/p's.

Implementation table :-

P \ QRS	000	001	010	011	100	101	110	111
$\bar{P}$ 0	(0)	(1)	2	(3)	(4)	5	6	7
P 1	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{P}$	$\bar{P}$	0	0	P



Ex: Implement $F(A,B,C,D) = \Pi M(0,1,2,4,6,9,12,14)$ using 8x1 MUX.

Sol:- $F(A,B,C,D) = \Pi M(0,1,2,4,6,9,12,14)$.

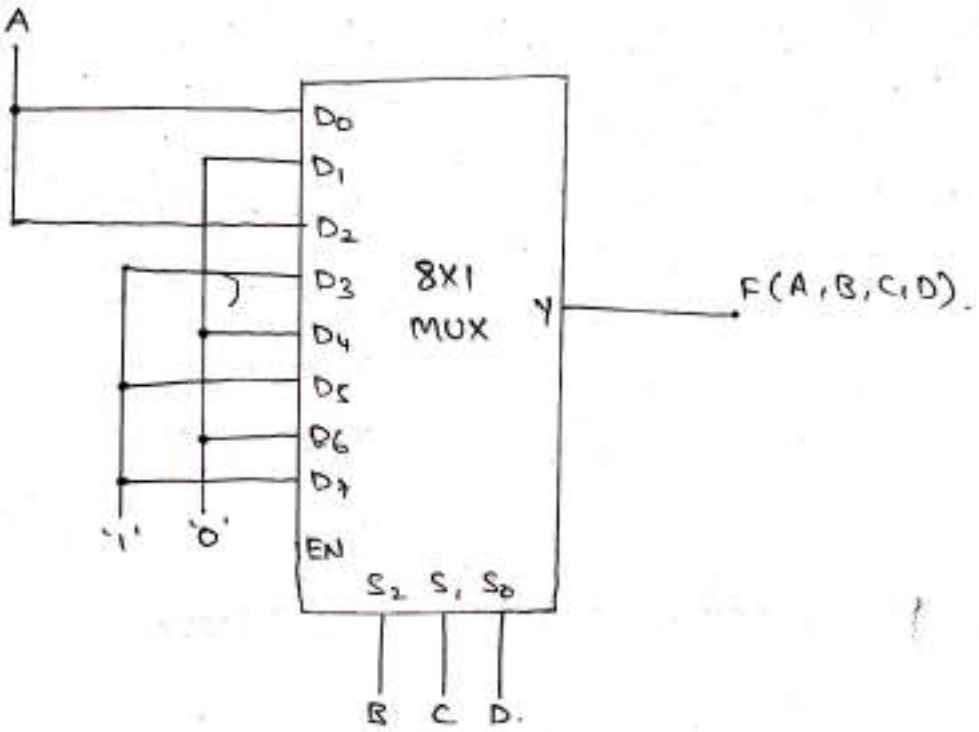
considering 'A' as data i/p & B, C, D as selection i/p's.

Here instead of minterms, Maxterms are Specified. Thus we have to circle Maxterms which are not included in the boolean function.

$\therefore F(A,B,C,D) = \Pi M(0,1,2,4,6,9,12,14) = \Sigma m(3,5,7,8,10,11,13,15)$

implementation table :-

A \ BCD	000	001	010	011	100	101	110	111
$\bar{A}$ 0	0	1	2	(3)	4	(5)	6	(7)
A 1	(8)	9	(10)	(11)	12	(13)	14	(15)
	A	0	A	1	0	1	0	1

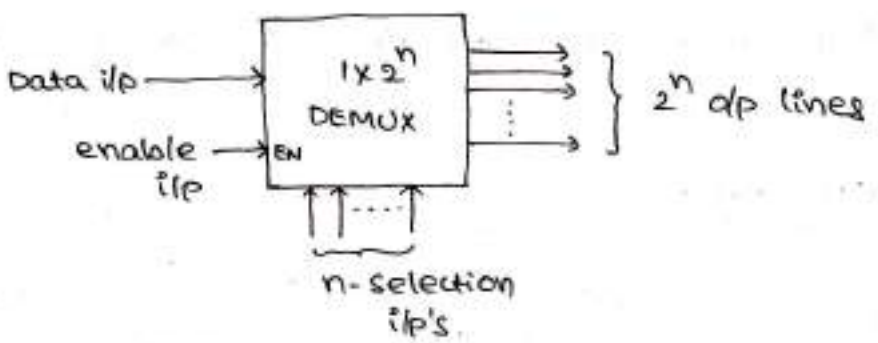


05/09/2014

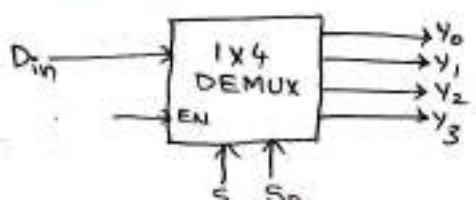
Demultiplexer (or) Data distributor :-

A demultiplexer is a combinational circuit that receives information on a single i/p line and transmits that information on one of 2^n possible o/p lines. The selection of specific o/p line is controlled by the values of 'n' selection lines.

Block diagram :-



(i) 1x4 demultiplexer



Truth table:-

i/p's			o/p's			
EN	S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	D_{in}
1	0	1	0	0	D_{in}	0
1	1	0	0	D_{in}	0	0
1	1	1	D_{in}	0	0	0

$\therefore Y_0 = D_{in} \cdot \bar{S}_1 \bar{S}_0$

$Y_1 = D_{in} \cdot \bar{S}_1 S_0$

$Y_2 = D_{in} \cdot S_1 \bar{S}_0$

$Y_3 = D_{in} \cdot S_1 S_0$

Ex:- Construct 1x4 DEMUX using 2, 1x2 DEMUX.

Sol:-

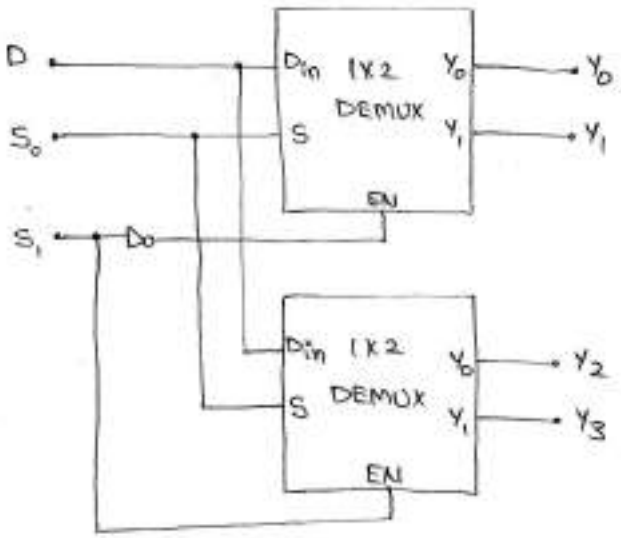


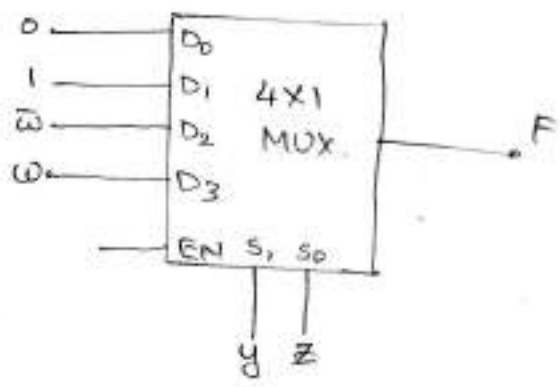
fig:- 1x4 DEMUX using two 1x2 demux.

Ex:- Implement $F(w,x,y,z) = \sum m(1,2,5,6,9,11,13,15)$ using 4x1 MUX.

Sol:- Considering w, x as data i/p's & y, z as section i/p's of 4x1 MUX.

Implementation table

wx \ yz	D_0 00	D_1 01	D_2 10	D_3 11
$\bar{w}\bar{x}$ 00	0	(1)	(2)	3
$\bar{w}x$ 01	4	(5)	(6)	7
$w\bar{x}$ 10	8	(9)	10	11
wx 11	12	(13)	14	15
	0	1	$\bar{w}\bar{x} + \bar{w}x = \bar{w}$	$w\bar{x} + wx = w$



09/09/2014

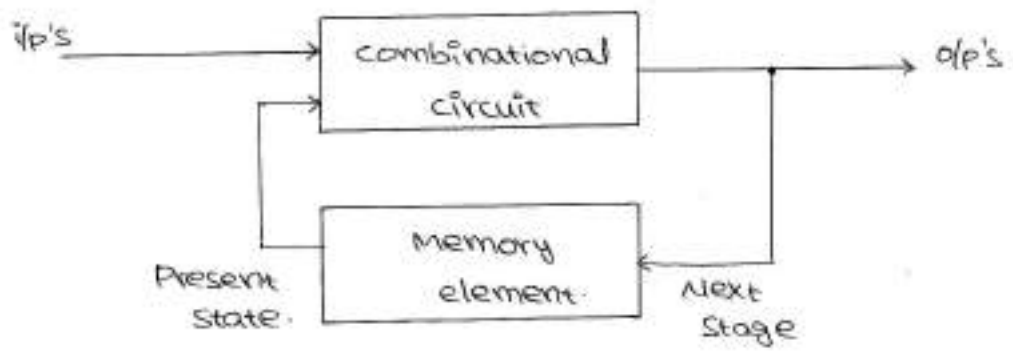
UNIT-III

SEQUENTIAL LOGIC

Differences b/w combinational circuit and sequential circuit:

combinational circuit	sequential circuit
(i) Output depends on only present inputs.	(i) Output depends on present i/p's and past outputs.
(ii) Memory unit is not required.	(ii) Memory unit is required to store the past history of o/p's.
(iii) Faster in speed.	(iii) slower than combinational circuits
(iv) Easy to design.	(iv) compared to combinational circuits harder to design.
(v) Ex:- Encoders, Decoders, MUX, DEMUX.	(v) Ex:- flip-flops, Registers, counters.

Block diagram of sequential circuit :-



* One of the most foundamental sequential circuit is latch / flip-flop.
* A latch / flip-flop is a bistable multivibrator, that can store 1 bit of binary information (either 0 or 1).
* Because of their ability to retain a given state, these elements are useful as storage elements.

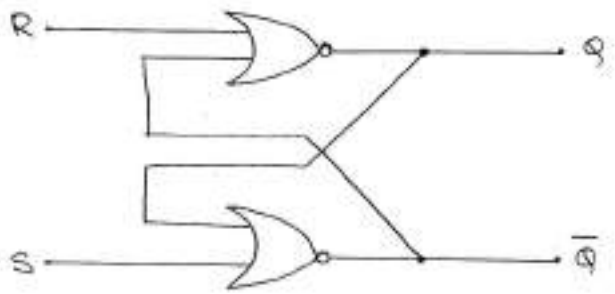
* LATCHES :-

1. RS latch / SR latch / Set - Reset latch / Reset - set latch :-

* The most fundamental type of storage element is an SR latch.
* SR latch as two i/p's (S and R), two o/p's (Q & $\bar{Q}$) which are complemented to each other.
* The SR latch can be constructed from either NAND (or) NOR gates.

Logic symbol of SR latch :-

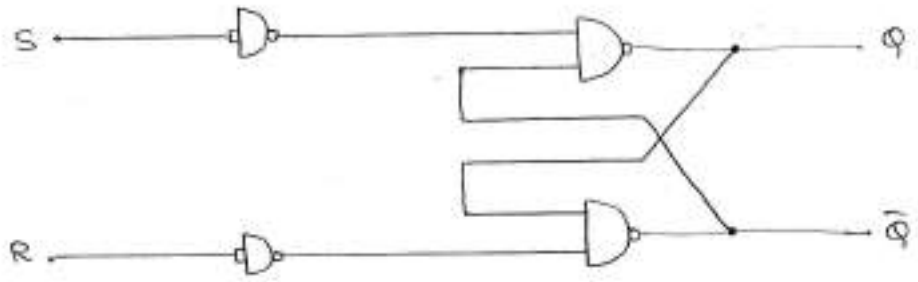
logic circuit of RS latch using NOR gates :-



Truth table / Function table :-

i/p's			o/p	State
R	S	Q_n	Q_{n+1}	
0	0	0	0	No change
0	0	1	1	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	X	Indeter-minate.
1	1	1	X	

Logic circuit of RS latch using NAND Gate :-

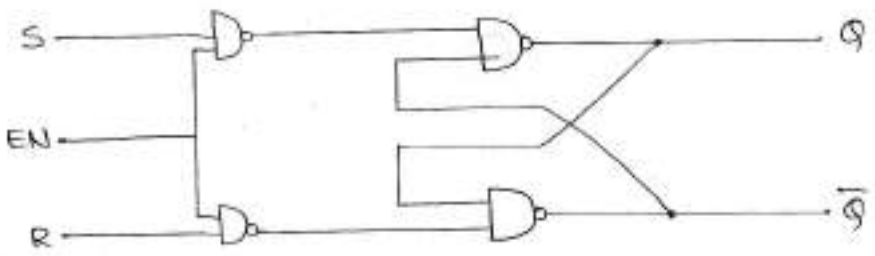


2. Gated SR latch :-

SR latch with enable i/p is known as "Gated SR latch"

Block diagram / Logic symbol :-

Logic circuit :-



Truth table / Function table :-

	i/p's			o/p	State
EN	R	S	Q_n	Q_{n+1}	
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	1	Set
1	0	1	1	1	
1	1	0	0	0	Reset
1	1	0	1	0	
1	1	1	0	X	Indeter-minate
1	1	1	1	X	
0	X	X	0	0	No change
0	X	X	1	1	

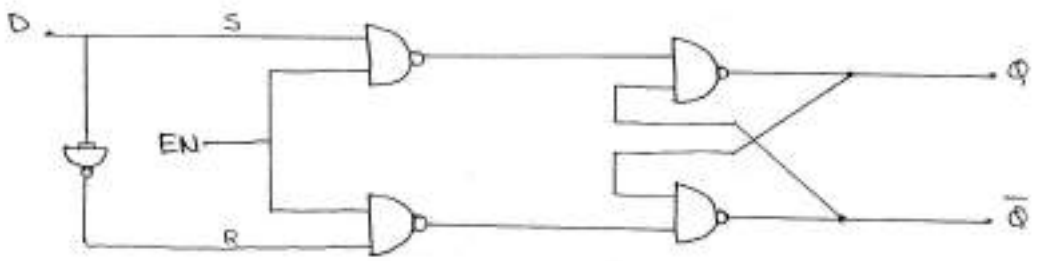
3. D latch / Data latch :-

* From the truth table of SR latch it is clear that when both i/p's are same, the o/p either doesn't change or it is invalid.
* In many practical applications these i/p conditions are not required.
* These i/p conditions can be avoided by making i/p's complement to each other.
* This modified SR latch is known as "D latch".

Block diagram / logic symbol :-

EN → Enable i/p.

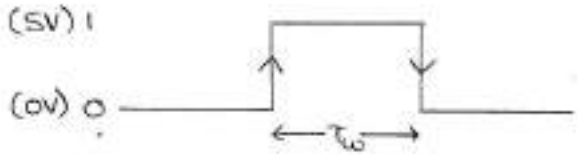
logic circuit :-



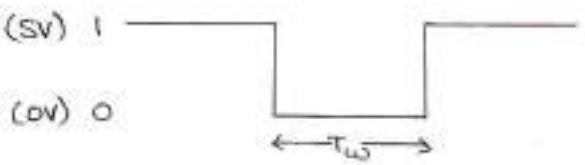
Truth table / Function table :-

i/p's			o/p	State
EN	D	Q_n	Q_{n+1}	
0	X	0	0	No change
0	X	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set.
1	1	1	1	

Positive pulse (0 to 1)

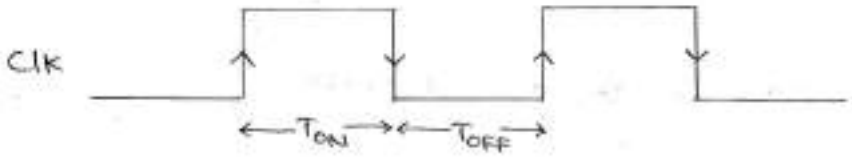


Negative pulse (1 to 0)



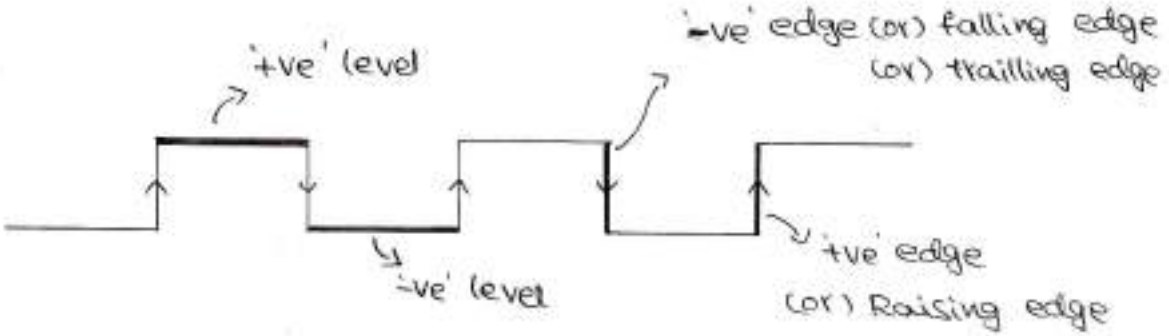
T_w → pulse width

Clk → clock is a periodic pulse train



Time period of clk signal $T = T_{ON} + T_{OFF}$

frequency of clk $(f) = \frac{1}{T}$



* Types of Triggerings :-

1. Level Triggering.
   (a) '+ve' level triggering
   (b) '-ve' level triggering
2. Edge Triggering
   (a) '+ve' edge Triggering
   (b) '-ve' edge Triggering → '-ve' edge triggered circuit responds to i/ps only at '-ve' edges of clk signal.

* Latches uses level triggering.

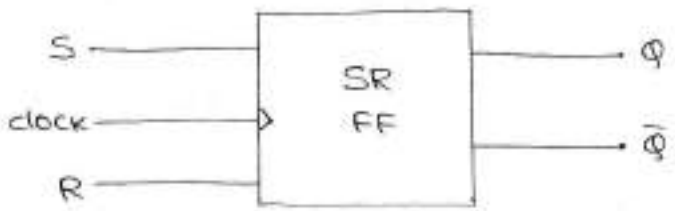
* Flip-flops uses edge triggering.

* Flip-flop :-

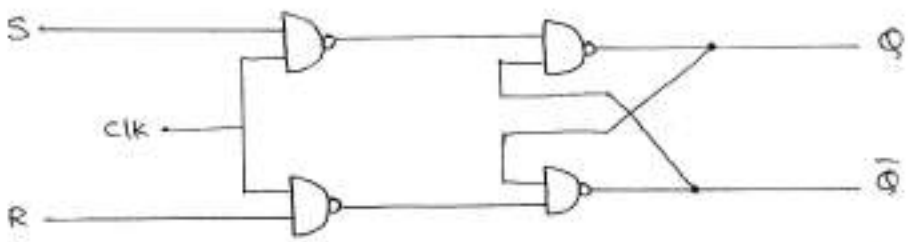
* Flip-flop is a memory element stores 1 bit of binary information.
* . Flip-flop has two stable states (1 or 0).
* State of flip-flop i.e bit stored is indicated by Q output.
* Other names of flip-flop are 1 bit memory cell. Bistable circuit, Bistable Multivibrator.
* on power, FF can be 0 or 1 state

1. Clocked SR flip-flop :-

(i) logic symbol of '+ve' edge triggered SR flip-flop :-



logic circuit :-



Q_n → state of FF before applying clk pulse

Q_{n+1} → state of FF after applying clk

Truth table :-

i/p's				o/p	State
clk	S	R	Q_n	Q_{n+1}	
↓ (or) ↑	0	0	0	0	No change
↓ (or) ↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	x	Indeter-minate
↑	1	1	1	x	
↑	x	x	0	0	No change
↑	x	x	1	1	

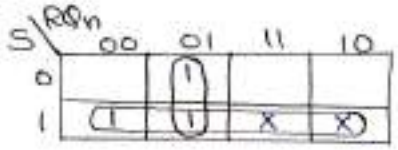
≡

characteristic table

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

characteristic equation of SR flip-flop :-

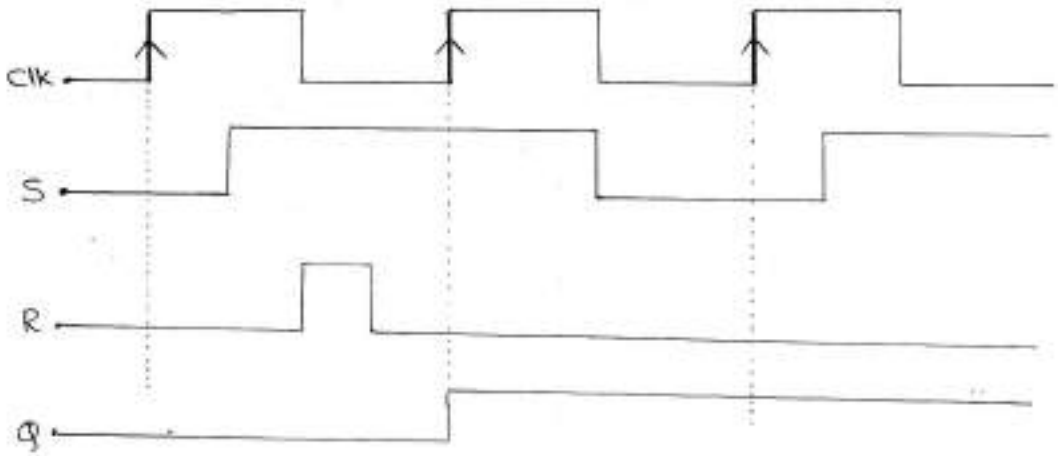
K-map for Q_{n+1}



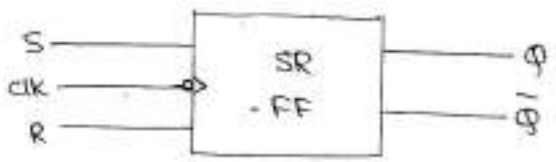
$\therefore Q_{n+1} = S + \bar{R}Q_n$

∴ characteristic equation of SR flip-flop is $Q_{n+1} = S + \bar{R}Q_n$

i/p & o/p waveforms of 'tve' edge triggered SR flip-flop:-

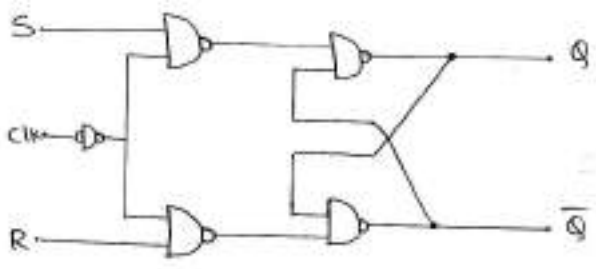


(ii) logic symbol of '-ve' edge triggered SR flip-flop :-



clock signal symbol
↓ (or) ↓

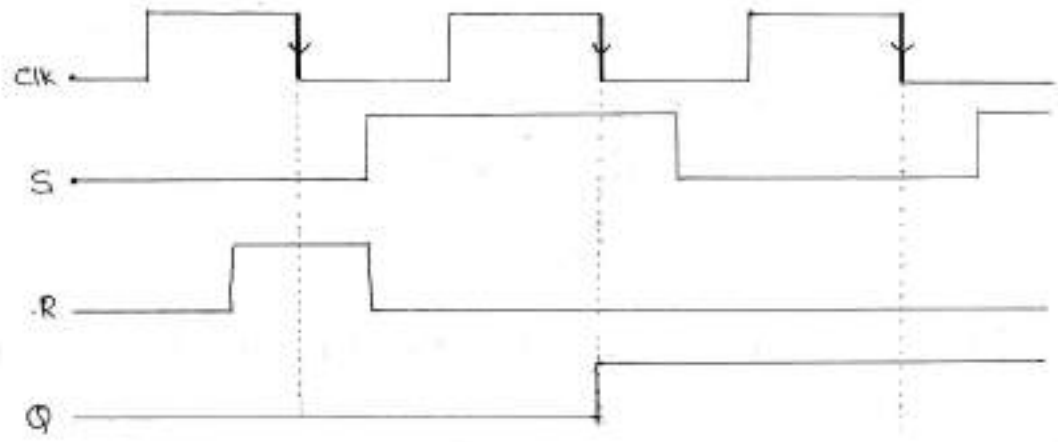
logic circuit :-



Truth table :-

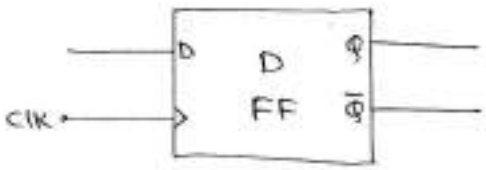
i/p's				o/p	
Clk	S	R	Q_n	Q_{n+1}	State
↓ (or) ↓	0	0	0	0	No change
↓ (or) ↓	0	0	1	1	
↓ (or) ↓	0	1	0	0	Reset
↓ (or) ↓	0	1	1	0	
↓	1	0	0	1	Set
↓	1	0	1	1	
↓	1	1	0	X	Indeter-minate
↓	1	1	1	X	
↓	X	X	0	0	No change
↓	X	X	1	1	

i/p & o/p waveforms of '-ve' edge triggered SR flip-flop :-

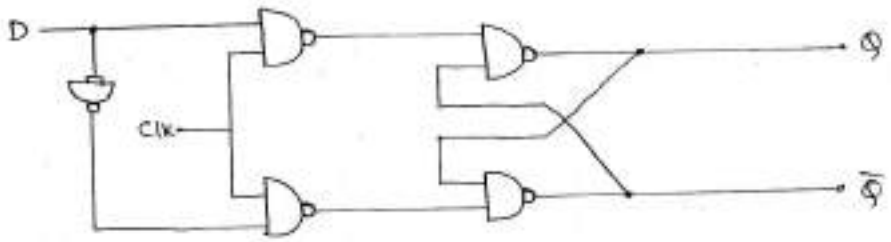


2. Clocked D flip-flop :- / Data flip-flop / Delay flip-flop :-

(i) Logic symbol of '+ve' edge triggered D flip-flop :-



Logic circuit :-



Truth table :-

i/p's			o/p	State
Clk	D	Q_n	Q_{n+1}	
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
0	x	0	0	No change
0	x	1	1	

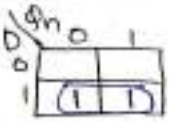
≡

characteristic table :-

D	Q_{n+1}
0	0
1	1

characteristic equation of D flip-flop :-

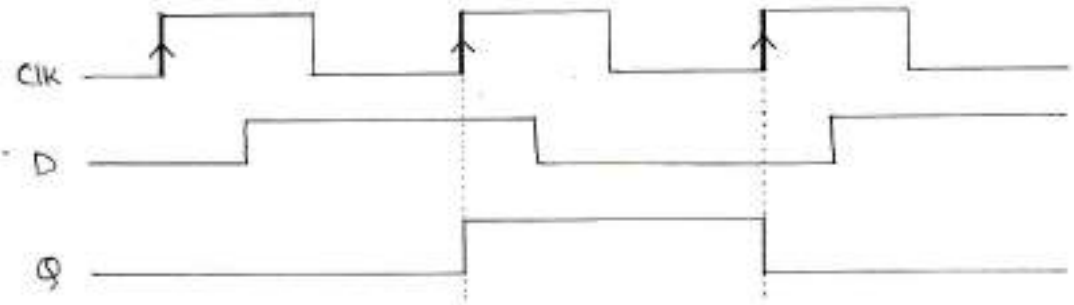
K-map for Q_{n+1}



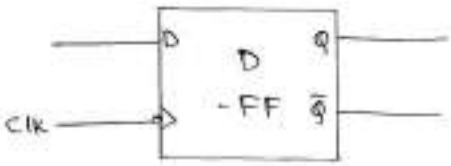
$\therefore Q_{n+1} = D$

$\therefore$ characteristic equation of D flip-flop is $Q_{n+1} = D$

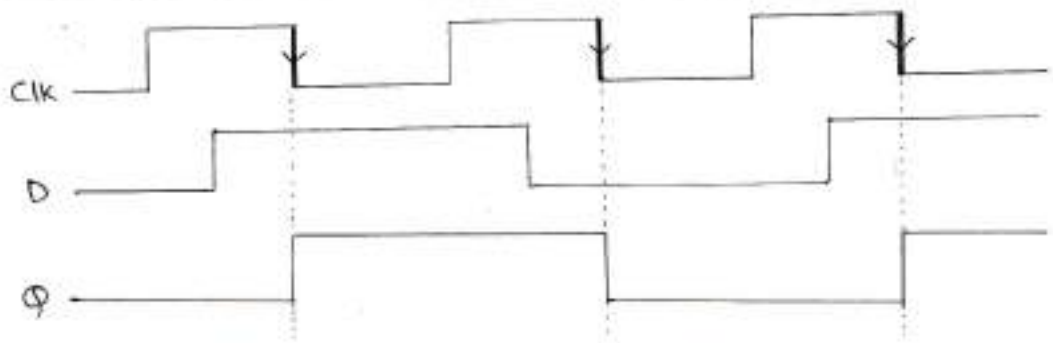
i/p & o/p waveforms of 'tve' edge triggered D flip-flop:-



(ii) Logic symbol of '-ve' edge triggered D flip-flop :-



i/p & o/p waveforms of '-ve' edge triggered D flip-flop:-

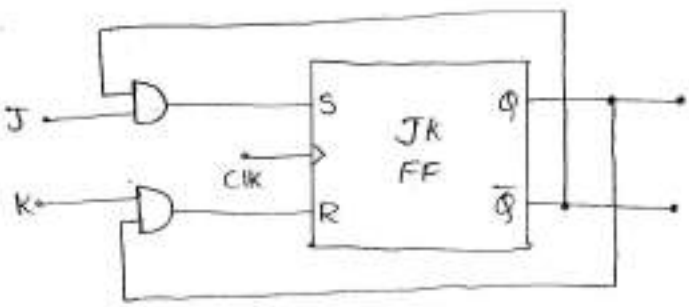


11/09/2014

3. Clocked JK flip-flop :-

the uncertinity in the state of an SR FF when S = R = 1 can be elimated by converting it into a JK flip-flop

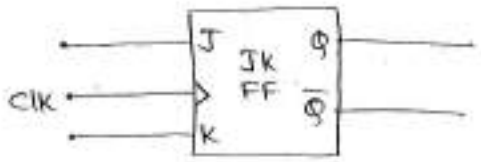
JK flip flop using SR flip-flop :-



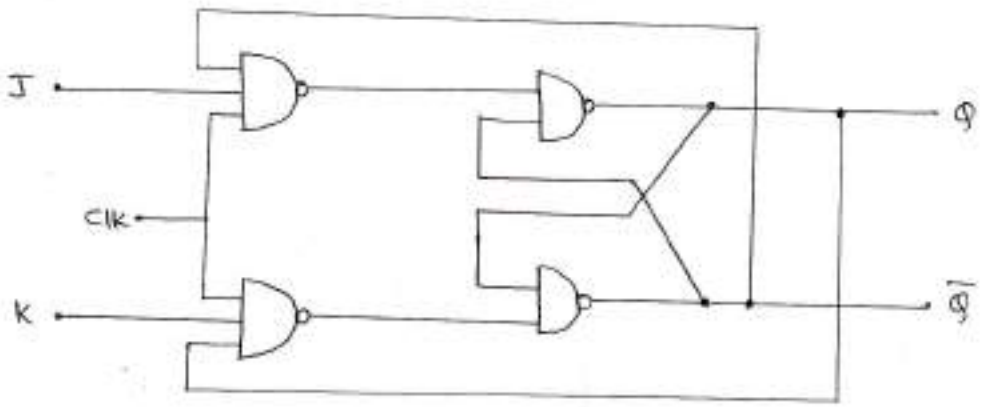
$S = J\bar{Q}$

$R = KQ$

logic symbol of JK Flip-flop :-



logic circuit of JK flip-flop using NAND gates :-



Truth table :-

i/p's				o/p
clk	J	K	Q_n	Q_{n+1}
0	x	x	0	0
0	x	x	1	1
↑	0	0	0	0
↑	0	0	1	1
↑	0	1	0	0
↑	0	1	1	0
↑	1	0	0	1
↑	1	0	1	1
↑	1	1	0	1
↑	1	1	1	0

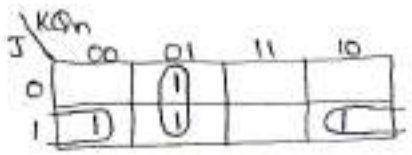
≡

characteristic table :-

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

characteristic equation of JK flip-flop :-

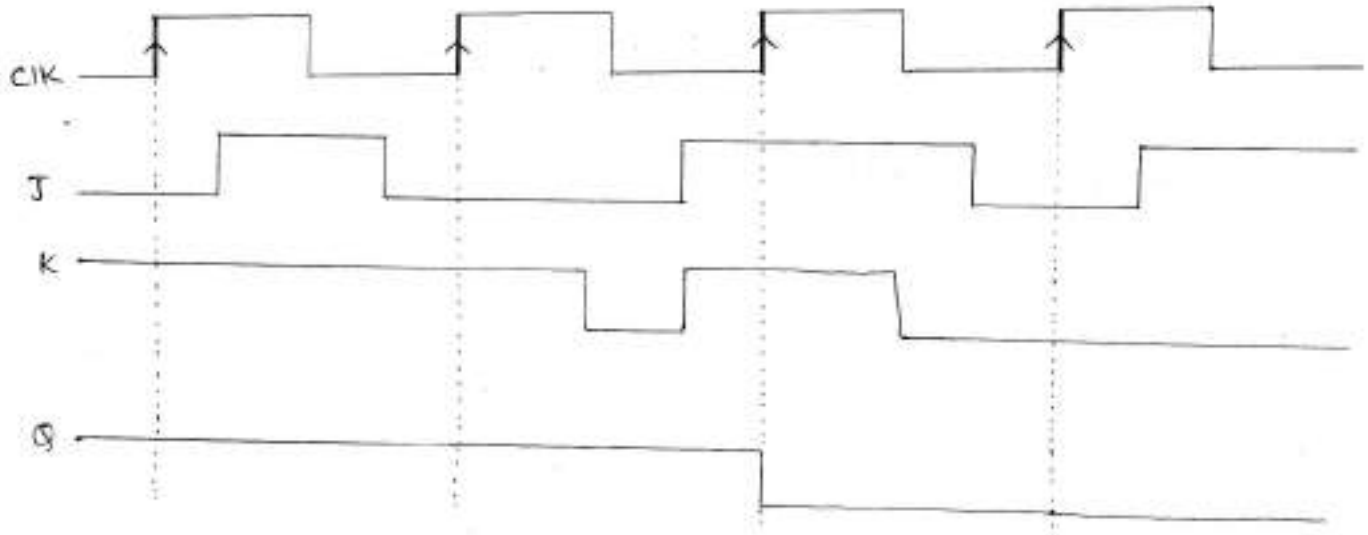
K-Map for Q_{n+1}



$\therefore Q_{n+1} = \bar{K}Q_n + J\bar{Q}_n$

characteristic equation of JK flip-flop is $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

i/p & o/p waveforms of 'tve' edge triggered JK flip-flop :-



16/09/2014

* Race around condition :-

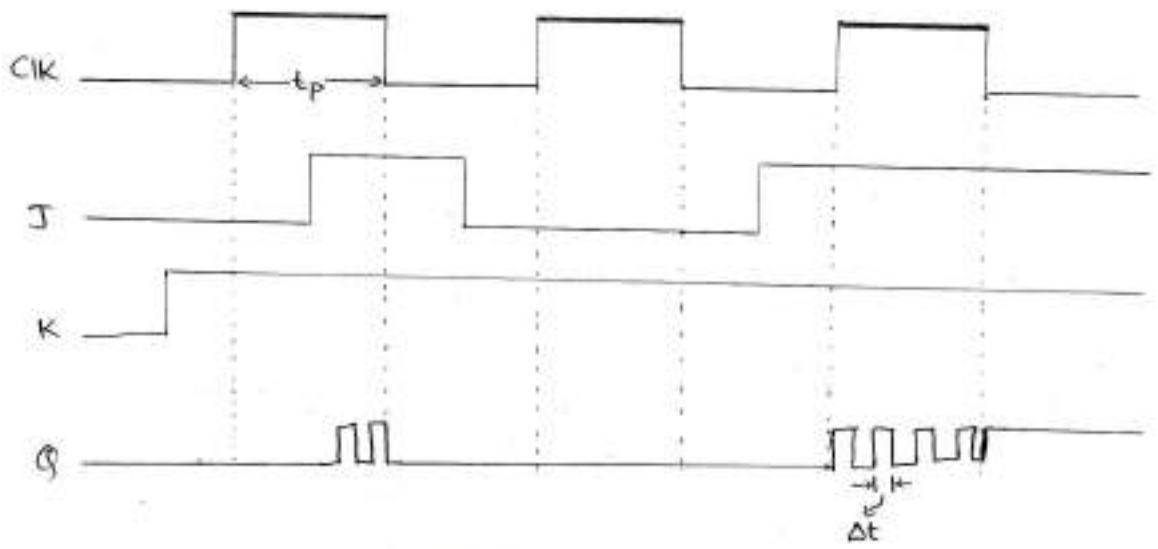


fig:- 'tve' level triggered JK FF i/p & o/p waveforms

$t_p \rightarrow$ pulse width

$\Delta t \rightarrow$ propagation delay of 2 level NAND gates (considering JK FF using NAND gates circuit)

* In JK flip-flop, when J = K = 1, the o/p continously toggles in that region (o/p changes either from 0 to 1 (or) from 1 to 0). which creates distribence in o/p this situation is referred to as the

Race around condition.

* There are 3 solutions for reducing race around condition.

(i) use of edge triggering.

(ii) RAC exists when $t_p \geq \Delta t$. thus By keeping $t_p < \Delta t$, we can avoid RAC.

(iii) Use of Master slave flip-flop configuration.

* Master - slave JK flip-flop :-

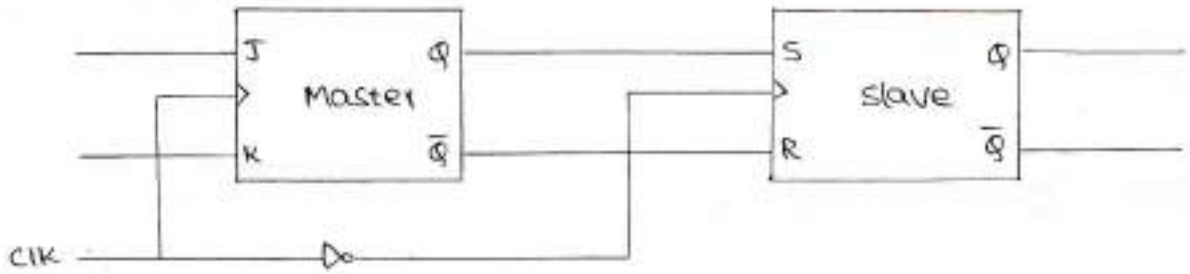


Fig:- logic diagram of Master - slave JK flipflop

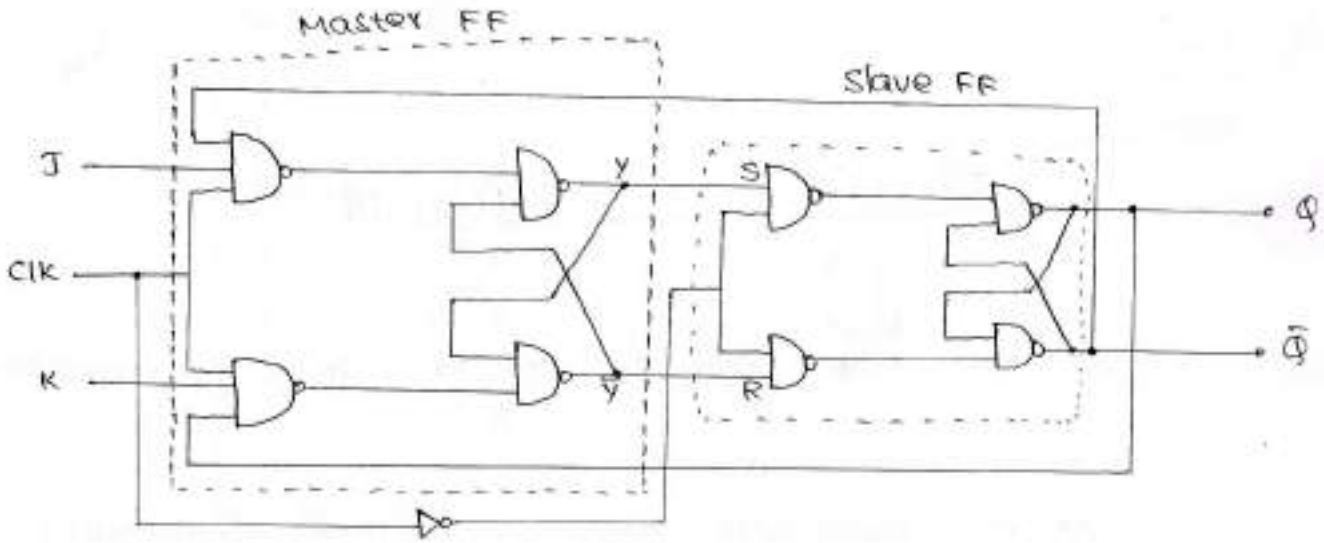


Fig: Logic circuit of Master slave JK FF.

* Master slave JK flip-flop consists of clocked JK flip-flop as the Master and clocked SR flip-flop as a slave.

* the o/p of the Master flip-flop is fed as an i/p to the slave FF.
* clock signal is connected directly to the Master flip-flop But it is connected through inverter to the slave flip-flop.

∴ the information present at the J & K i/p's is transmitted to the o/p of master flip flop on the 'tve' clock pulse & it is held their until the '-ve' clock pulse occurs; After which it is allowed to pass through to the o/p of slave FF. The o/p of the slave FF is connected to the third i/p of the JK Master JK FF.

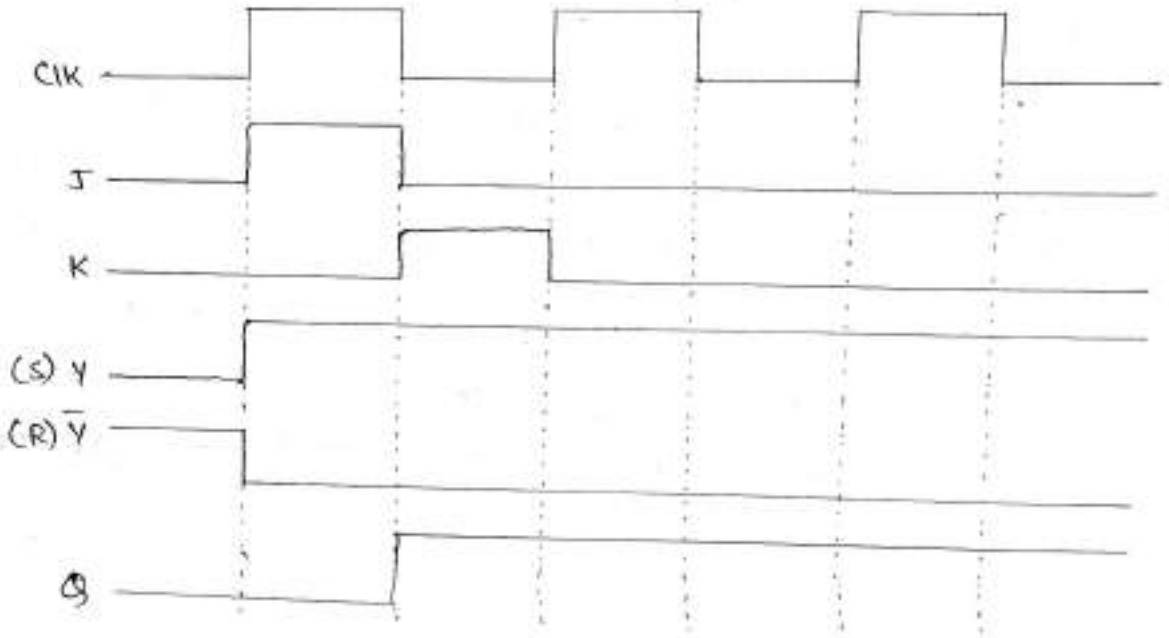
Truth table:-

Clk	J	K	Q_n	Master FF o/p Y	Slave FF o/p Q_{n+1}
↑	0	0	0	0	No change
↓	0	0	0	NC	0
↑	0	0	1	1	NC
↓	0	0	1	No change	1
↑	0	1	0	0	NC
↓	0	1	0	NC	0
↑	0	1	1	0	NC
↓	0	1	1	NC	0
↑	1	0	0	1	NC
↓	1	0	0	NC	1
↑	1	0	1	1	NC
↓	1	0	1	NC	1
↑	1	1	0	1	NC
↓	1	1	0	NC	1
↑	1	1	1	0	NC
↓	1	1	1	NC	0

* When J=1 & K=1 master flip-flop toggles on 'tve' clock and slave then copies the o/p of master on the '-ve' clock. at this instant, feedback i/p's to the Master flip-flip are complemented but as it is '-ve' half of the clock pulse Master FF is inactive.

This prevents Race around condition.

i/p & o/p waveforms of Master-slave JK FF :-



* JK FF Responds to +ve clk Pulses
* SR FR Responds to -ve clk Pulses

Differences b/w Latch & Flip-flop :-

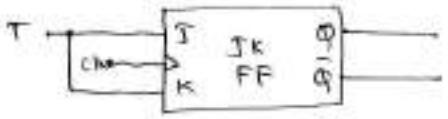
Latch	Flip-flop
* A latch checks all its i/p's continuously & changes its o/p's accordingly at any time.	* flip-flop samples its i/p's & changes ~~change~~ its o/p's only at a time as determine by a clock signal
* No clock is used.	* A clock is used.

12/09/2014

Clocked T Flip-flop / Toggle FF :-

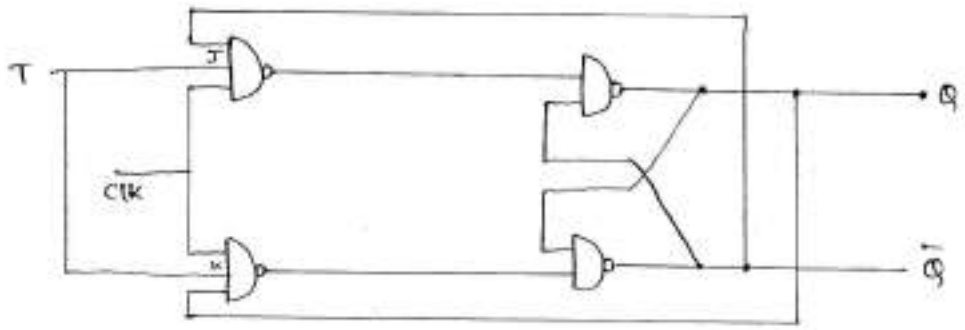
* the T flip-flop is a modification of the JK flip-flop.
* T flip-flop is obtained from a JK flip-flop by connecting J & K i/p's together.

logic diagram of T FF using JKFF :-



logic diagram of T FF :-

logic circuit :-



Truth table :-

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

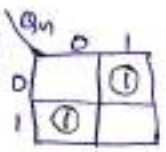
$\equiv$

characteristic table :-

T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

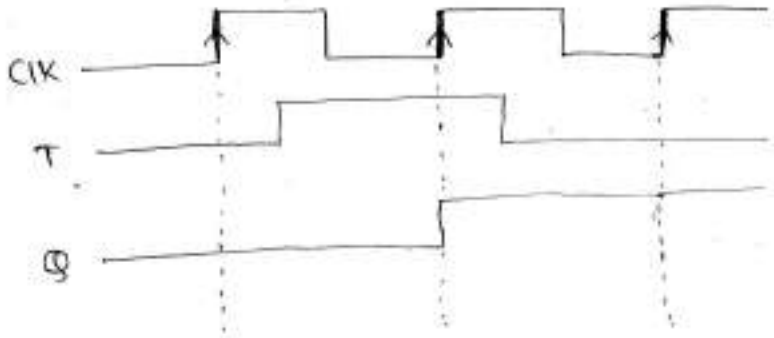
characteristic equation of T flip flop :-

K Map for Q_{n+1}



$\therefore Q_{n+1} = \bar{T}Q_n + T\bar{Q}_n$

I/p and o/p waveforms of +ve edge triggered T-FF:-



* Flip-flop excitation tables :-

1. SR flip-flop :-

characteristic table :-

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	* → Indeterminate

Excitation table :-

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

2. D flip-flop :-

characteristic table :-

D	Q_{n+1}
0	0
1	1

Excitation table :-

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

3. JK flip-flop :-

characteristic table :-

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Excitation table :-

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

4. T - flip - flop :-

Characteristic table :-

T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

Excitation table :-

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

Realisation of one flip-flop using other flip-flop :-

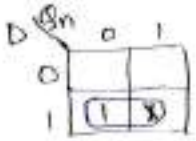
Ex:- convert SR FF to D FF.

Sol:- Excitation table for SR to D FF conversion :-

i/p	present state	next state	FF i/p's	
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

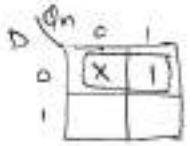
K-Map simplification :-

for S



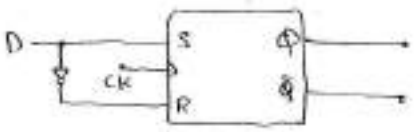
$\therefore S = D$

for R



$\therefore R = \bar{D}$

logic diagram of DFF using SR FF :-



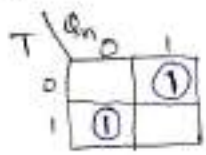
2. Convert D FF to TFF :-

Excitation table for D to T FF conversion :-

i/p	Present State	Next State	FF i/p
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

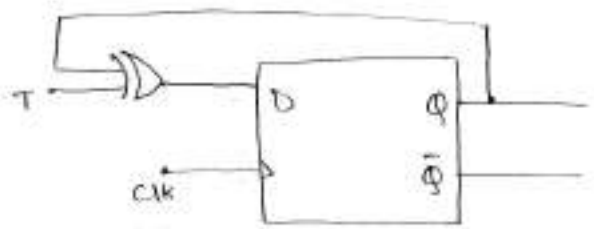
K-Map Simplification :-

for D :-



$$\therefore D = \bar{T}Q_n + T\bar{Q}_n = T \oplus Q_n$$

Logic diagram of T FF using D FF :-



3. Convert SR FF to JK FF

Excitation table for SR FF to JK FF conversion :-

i/p		Present State	Next State	FF i/p	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1

k-map simplification :-

for S

J \ KQ_n	00	01	11	10
0		X		
1	1	X		1

$\therefore S = J\bar{Q}_n$

for R

J \ KQ_n	00	01	11	10
0	X		1	X
0			1	

$\therefore R = KQ_n$.

logic diagram of Jk flip-flop using SR FF :-

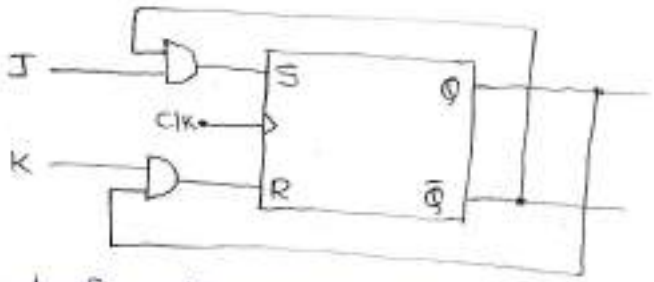


Fig:- logic diagram of JK FF using SR FF.

21/10/2014

Analysis and Design of Synchronous (or) clocked Sequential circuits

The synchronous or clocked sequential circuits are represented by two models.

(i) Moore circuit (or) Moore model (or) Moore Mechine

(ii) Mealy circuit

1. Moore circuit :-

(i) Moore circuit o/p depends only on the present state of FF's.

(ii) I/p changes doesn't effect the o/p.

(iii) It requires more no. of states for implementing function which is implemented using Mealy circuit.

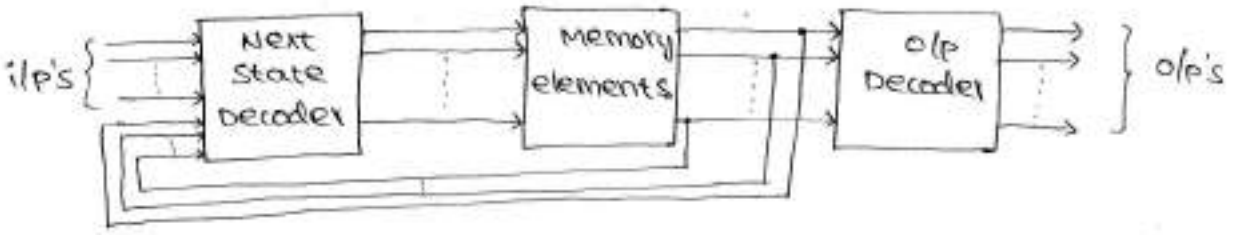


Fig:- B.D. of Moore circuit

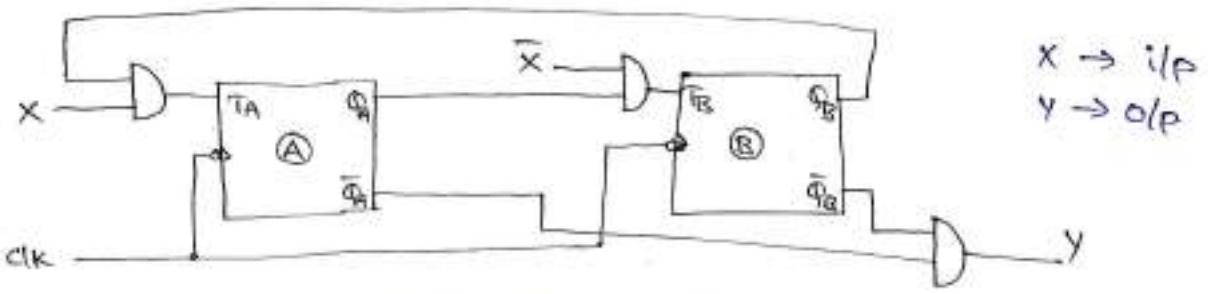


Fig:- Example of moore circuit.

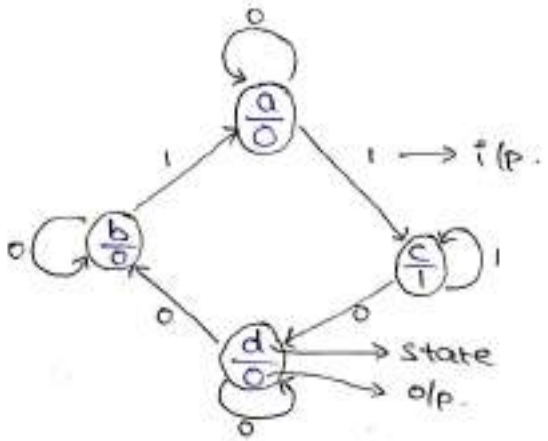


Fig:- Example state diag. for moore circuit.

2. Mealy circuit :-

(i) The o/p depends on both the present state of the FF's and on the i/p's.

(ii) I/p changes may effect the o/p of the ckt.

(iii) It requires less no. of states for implementing the function which is implemented using Moore circuit.

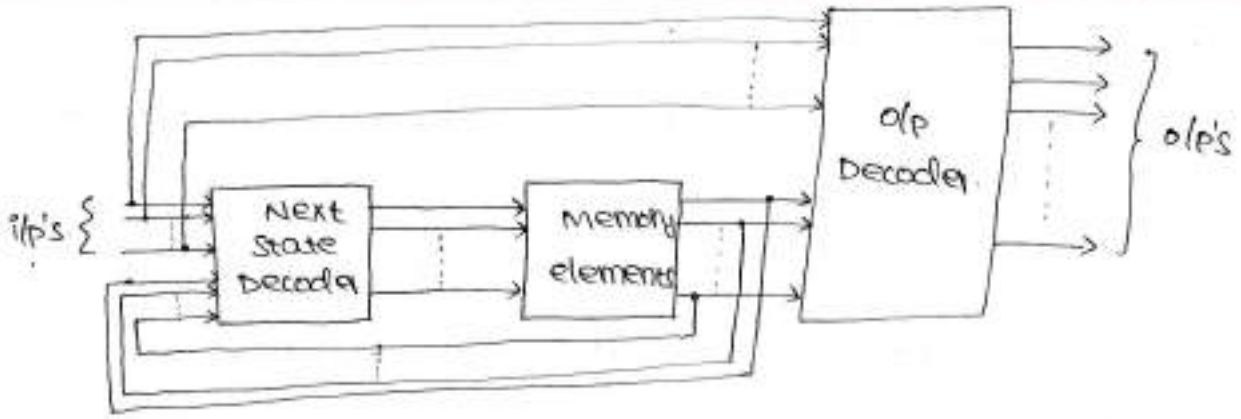
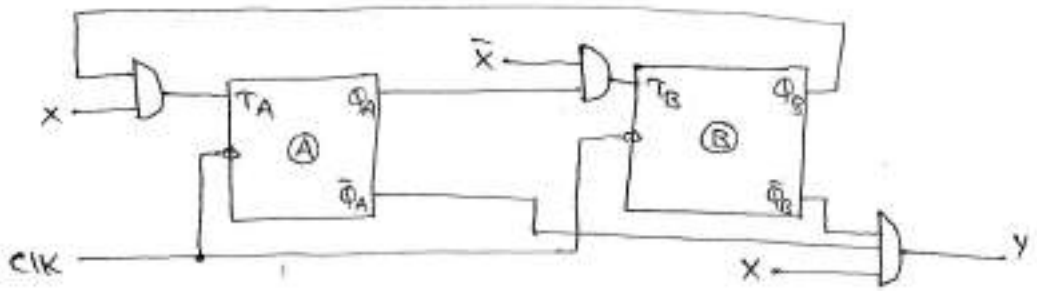


Fig:- B.D. of Melay circuit.



x → i/p
y → o/p.

Fig:- Example of mealy circuit

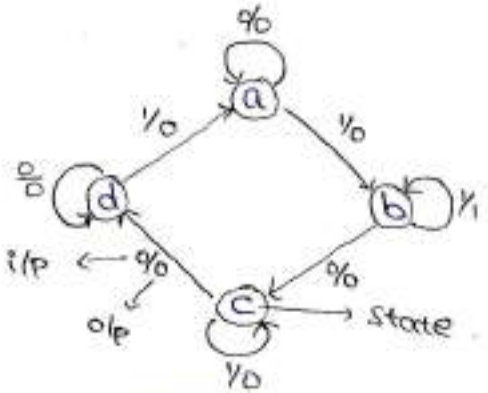


Fig:- example State diag. for mealy circuit.

Steps for the Design of a Synchronous (or) clocked sequential circuits :-

Step 1 : Obtain the state table from the given circuit information such as state diagram.

Step 2 : Reduce no. of states if possible by state reduction technique.

Step 3 : Assign binary values to each state in the state table

Step 4 : Determine the no. of FF's needed and

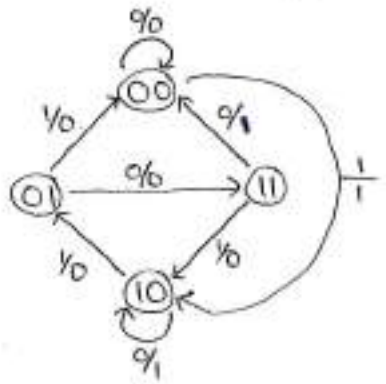
choose the type of FF to be used.

Step 5: From the state table derive the circuit exitation & o/p tables.

Step 6: Using K-Map, derive the circuit o/p functions and the FF i/p functions.

Step 7: Draw the logic diagram.

Q:- A sequential circuit has 1 i/p & 1 o/p. that ckt state diagram is shown below. Design the sequential circuit with D-FF.



Sol:- Step 1: state diagram

The given state diagram has 4 states with one i/p and one o/p. considering X → i/p
Y → o/p

Present state		Next state				o/p (Y)	
		X=0		X=1		X=0	X=1
Q_A	Q_B	Q_{A+1}	Q_{B+1}	Q_{A+1}	Q_{B+1}	Y	Y
0	0	0	0	1	0	0	1
0	1	1	1	0	0	0	0
1	0	1	0	0	1	1	0
1	1	0	0	1	0	1	0

Step 2: State Reduction and state Assignment

Here binary values are already assigned to states and it is not possible to reduce no. of states because no two present states, Next states and o/p's are same.

Step3 : Determine no. of FF's required.

Determine no. of FF's required using eq.,

$2^n \geq N$ $\quad n \rightarrow$ no. of FF's.
$N \rightarrow$ no. of states.

$2^n \geq 4$

$\therefore n = 2$

FF's required $\longrightarrow$ 2
FF's using $\longrightarrow$ D F.F's

Step4: Exitation table

present state		i/p	Next State		FF i/p's		o/p
Q_A	Q_B	X	Q_{A+1}	Q_{B+1}	D_A	D_B	Y
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

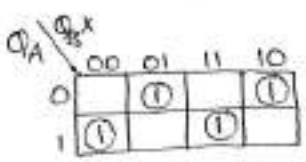
(∵ D-FF exitation table

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

Step5: K-Map simplification.

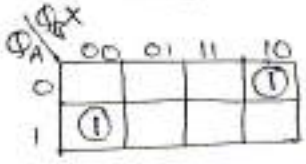
K-map simplification for FF i/p functions & sequential circuit o/p function.

for D_A



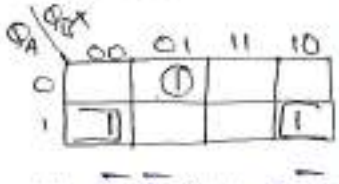
$\therefore D_A = Q_A \oplus Q_B \oplus X$

FOR D_B



$\therefore D_B = \bar{Q}_A Q_B \bar{X} + Q_A \bar{Q}_B \bar{X}$

FOR Y



$\therefore Y = \bar{Q}_A \bar{Q}_B X + Q_A \bar{X}$

Step 6: logic diagram

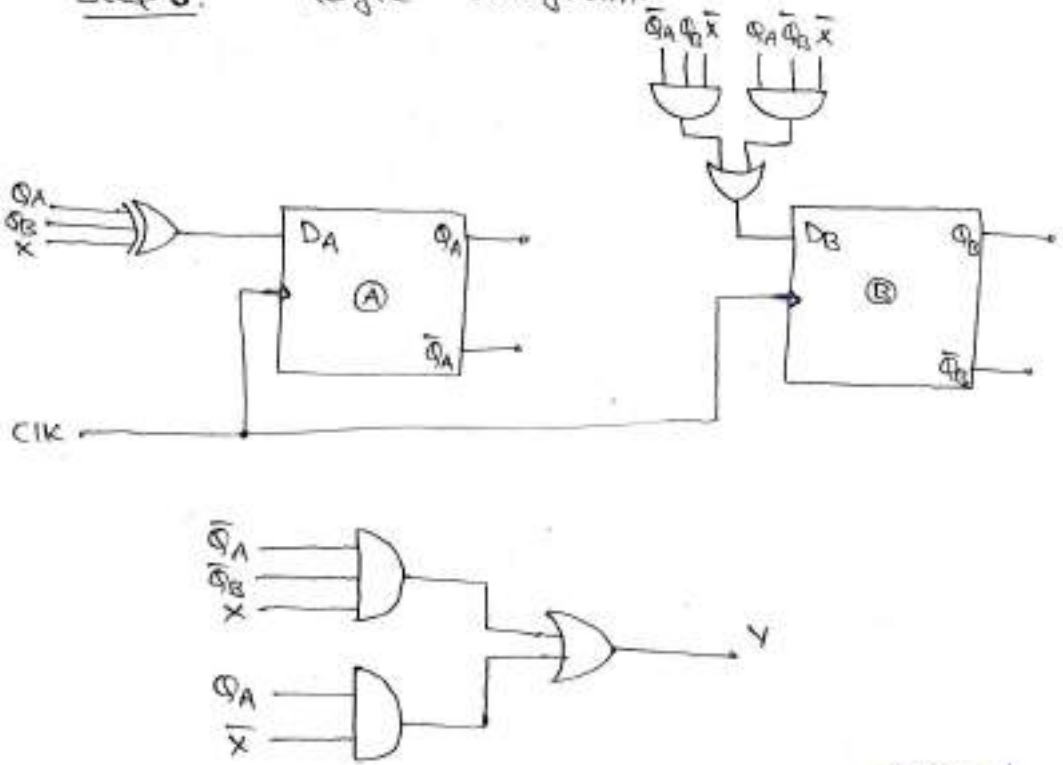
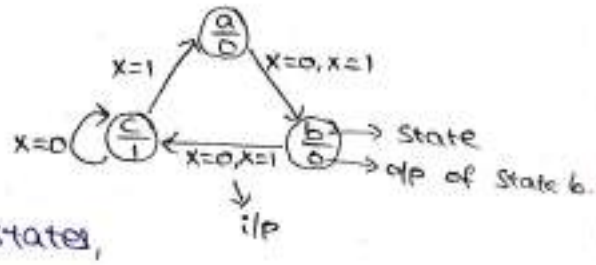


Fig: logic diagram of mealy sequential circuit for given State diagram.

22/10/2014

Q: Realize the sequential circuit for the state diagram shown Below.



Sol: Given state diag. has 3 states, one i/p, one o/p.

considering $x \rightarrow$ i/p

$Y \rightarrow$ o/p

Step 1: State table

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

Step 2: State Reduction

there is no possibility of State Reduction since no two states, Next States & o/p's are same.

Step 3: No. of FF's Required

Determine no. of FF's required using eq.

$2^n \geq N$ $\quad n \rightarrow$ no. of FF's, $N \rightarrow$ no. of States

$2^n \geq 3$

$\therefore n = 2$

$\therefore$ FF's required = 2
FF's using → T FF's

Step 4: State Assignment

Assign Binary values to states

$a = 00$, $b = 01$, $c = 10$

Step 5: Exitation table

Present State		i/p	Next State		FF i/p's		o/p
Q_A	Q_B	x	Q_{A+1}	Q_{B+1}	T_A	T_B	y
0	0	0	0	1	0	1	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	1	0	1	1	0
1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	1

∵ T-FF exitation table

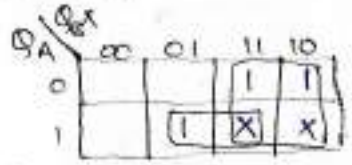
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State 11 is unused state. ∴ Treat that state as don't care.

Step6: K-Map simplification

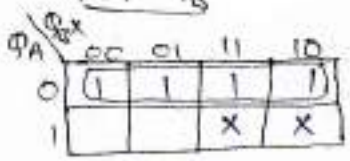
K-Map simplification for FF's i/p functions and sequential circuit o/p function

for T_A



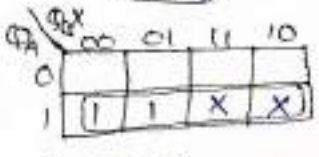
$\therefore T_A = Q_A x + Q_B$

For T_B



$\therefore T_B = \bar{Q}_A$

for y



$\therefore y = Q_A$

Step6: LOGIC DIAGRAM

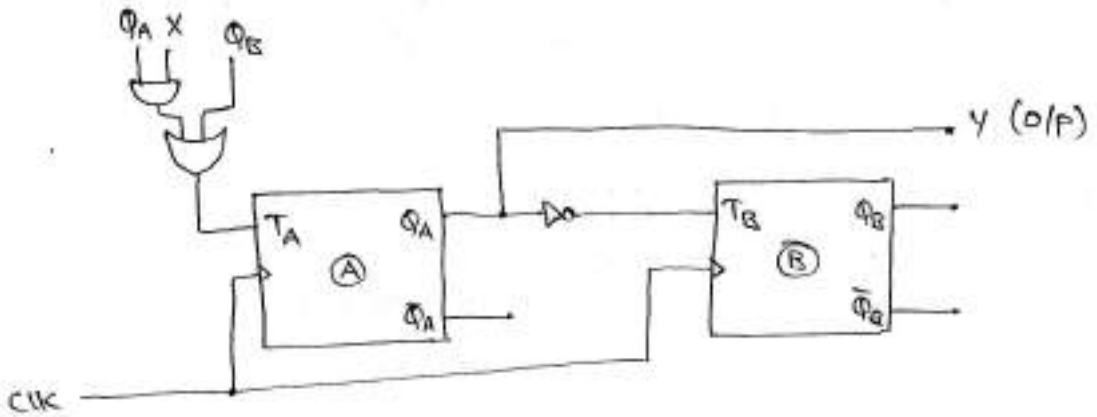
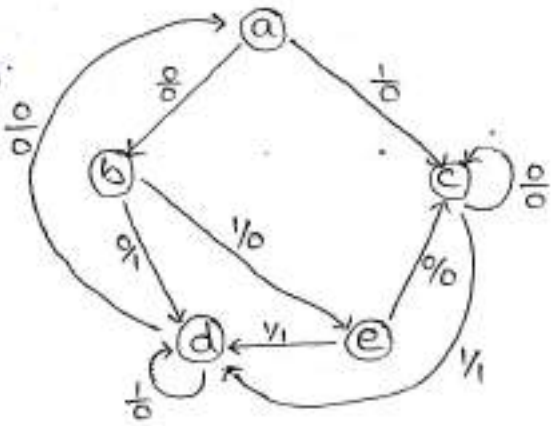


Fig:- logic diagram of moore sequential ckt for given state diagram.

Q:- Design a clocked sequential circuit for state diagram shown below.



Sol:- Given state diag. has 5 states with one i/p and one o/p.

considering.

$X \longrightarrow$ i/p

$Y \longrightarrow$ o/p.

Step1: State table

Present State	Next State X=0	Next State X=1	o/p (Y) X=0	o/p (Y) X=1
a	b	c	0	0
b	d	~~e~~ c	1	0
✓ c	c	d	0	1
d	a	d	0	0
x ~~e~~	c	d	0	1

c & e states Next states and o/p's are same.
∴ we can cancel either c (or) e state. if e is cancelled then replace e with c.

Reduced state table

present state	Next State x=0	Next State x=1	o/p (y) x=0	o/p (y) x=1
a	b	c	0	0
b	d	c	1	0
c	c	d	0	1
d	a	d	0	0

Step 2: Determine no. of FF's required.

no. of FF's required to design seq. ckt can be determined by the eq.,

$2^n \geq N$ n → no. of FF's, N → no. of states.

$2^n \geq 4$ (∵ After state reduction N=4)

∴ n=2

∴ no. of FF's required = 2

FF's using → D-FF's.

Step 3: State Assignment

Assign Binary values to states

a=00 , b=01 , c=10 , d=11

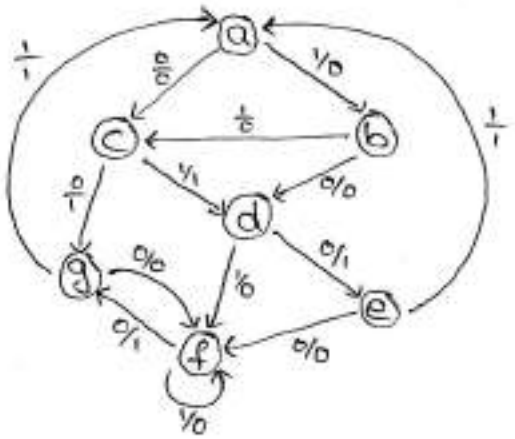
Step 4: Exitation table

(for procedure refer previous problem)

Step 5: K-map simplification

Step 6: Logic diagram

Q:- Design a clocked seq. circuit for state diagram shown below.



Sol:- The given state diag. has 7 states with one i/p & one o/p.

considering $x \rightarrow i/p$

$y \rightarrow o/p$

Step1: State table

present state	Next State x=0	Next State x=1	O/P(y) x=0	O/P(y) x=1
a	c	b	0	0
b	d	c	0	0
c	~~g~~ e	d	1	1
d	e	f	1	0
✓ e	f	a	0	1
f	~~g~~ e	f	1	0
x ~~g~~	f	a	0	1

e & g states Next states and o/p's are same.

∴ we can cancel either e (or) g state. If 'g' is cancelled then replace g with e.

reduced state table

present state	Next State x=0	Next State x=1	O/P(y) x=0	O/P(y) x=1
a	c	b	0	0
b	d	c	0	0
c	e	d	1	1
✓ d	e	~~f~~ d	1	0
e	~~f~~ d	a	0	1
x ~~f~~	e	f	1	0

d and f states Next states and o/p's are same.

∴ we can cancel either d or f state. If 'f' is cancelled then replace 'f' with 'd'

final reduced state table

present state	Next state x=0	Next state x=1	o/p (y) x=0	o/p (y) x=1
a	c	b	0	0
b	d	c	0	0
c	e	d	1	1
d	e	d	1	0
e	d	a	0	1

Step 2:

no. of FF's required = 3 ($\because 2^n \geq 5$, $n=3$)

FF's using → D FF's.

Step 3: State Assignment

Assign binary values to states

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

101, 110, 111 are unused states.

∴ treat them as don't cares.

(i.e) for 101, 110, 111 states, Next states and i/p and o/p's are don't cares.

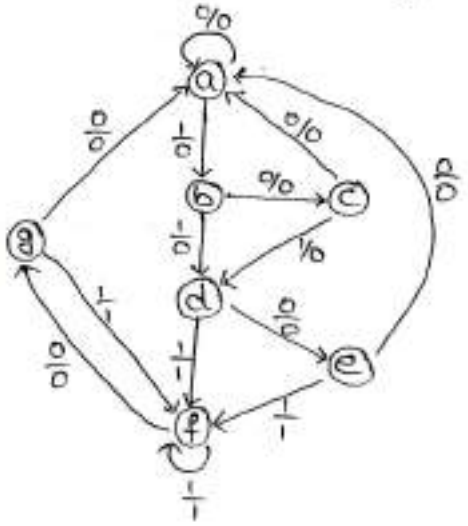
Step 4: Excitation table

Step 5: K-map simplification

Step 6: Logic diagram

25/10/2014

Q. :- Obtain the reduced state table and reduced state diagram for a sequential ckt whose state diagram shown below.



Sol:- The give state diagram has 7 states, one i/p & 1 o/p.

considering,

$x \longrightarrow$ i/p

Step 1: State table

Present State	Next State		o/p	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
✓ e	a	f	0	1
f	~~g~~ e	f	0	1
x ~~g~~	a	f	0	1

∴ e & g states Next states and o/p's are same

∴ We can cancel either e (or) g state. If 'g' is cancelled then replace g with e.

reduced state table

Present State	Next State		o/p	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
✓ d	e	~~f~~ d	0	1
e	a	~~f~~ d	0	1
x f	e	f	0	1

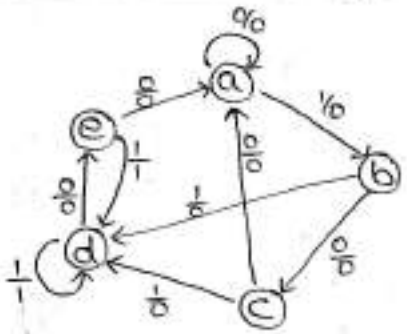
∴ d & f States Next states and o/p's are same.

∴ We can cancel either d (or) f state. If 'f' is cancelled then replace 'f' with 'd'.

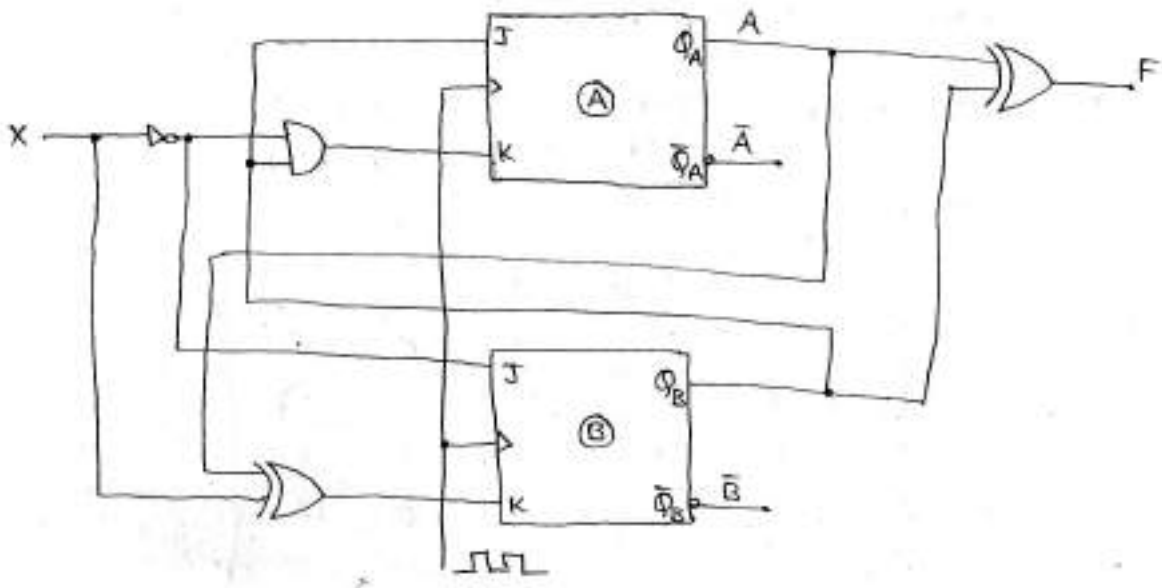
Final reduced state table

Present State	Next State X=0	Next State X=1	O/P X=0	O/P X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Reduced State diagram:



Q. :- Derive the transition table, State table & State diagram for the Moore Sequential circuit given below.



Sol:- Step 1: Determine the FF i/p Eq.'s & the o/p eq.'s of sequential circuit from the above seq. ckt.

FF i/p eq.'s

$J_A = Q_B$

$K_A = \bar{x} Q_B$

$J_B = \bar{x}$

$K_B = Q_A \oplus x$

Seq. circuit o/p

$F = Q_A \oplus Q_B$

Step 2: Derive the transition eq.'s.

The transition eq.'s for JK FF's can be derived from the characterstic eq. of JK FF.

char. eq. of JK FF is $Q_{n+1} = J\bar{Q}_n + \bar{K} Q_n$

$Q_{A+1} = J_A \bar{Q}_A + \bar{K}_A Q_A$

After substituting J_A & K_A eqn's,

Above eqn. becomes,

$\therefore \boxed{Q_{A+1} = Q_B \bar{Q}_A + (\overline{\bar{x} Q_B}) Q_A}$

$Q_{B+1} = J_B \bar{Q}_B + \bar{K}_B Q_B$

After substituting J_B & K_B eqn's,

Above eqn. becomes,

$\therefore \boxed{Q_{B+1} = \bar{x} \bar{Q}_B + (\overline{Q_A \oplus x}) Q_B}$

Step 3: Transition table

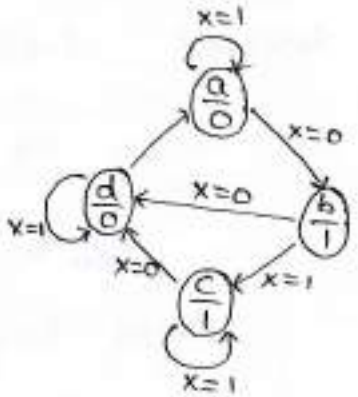
Present State		Next State x=0		x=1		Output
Q_A	Q_B	Q_{A+1}	Q_{B+1}	Q_{A+1}	Q_{B+1}	F
0	0	0	1	0	0	0
0	1	1	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0

Step 4: State table

considering a=00, b=01, c=10, d=11.

present state	Next State x=0	Next State x=1	O/P (F)
a	b	a	0
b	d	c	1
c	d	c	1
d	a	d	0

Step 5: State diagram



A	(or)	Q_A				
A^+	(or)	Q_{A+1}	(or)	A(t+1)	(or)	Q_A^+

} Different representations.

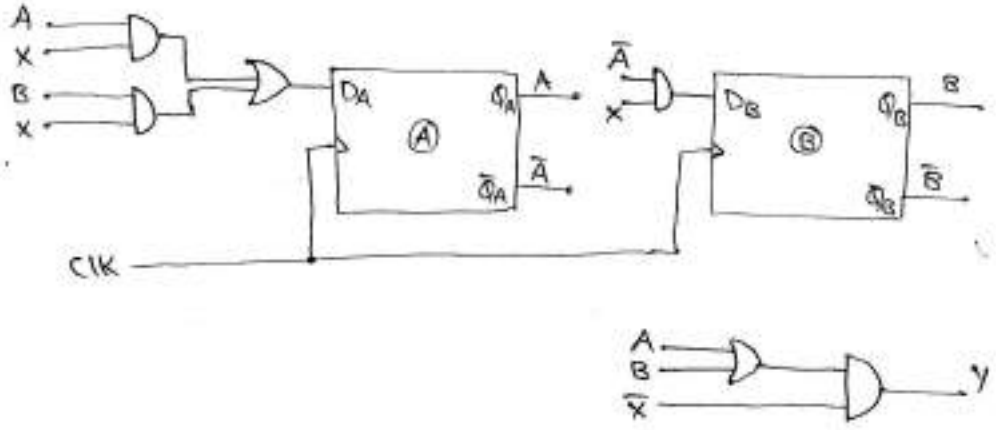
Q.:- A sequential circuit with two D-FF's A & B and i/p x & o/p Y. is specified by the following Next State and o/p equations.

$$A(t+1) = Ax + Bx$$

$$B(t+1) = A'x \quad ; \quad Y = (A+B)x'$$

(i) Draw the logic diagram of the ckt.
(ii) Derive the State table
(iii) Derive the State diagram.

Sol:- (i) logic diagram



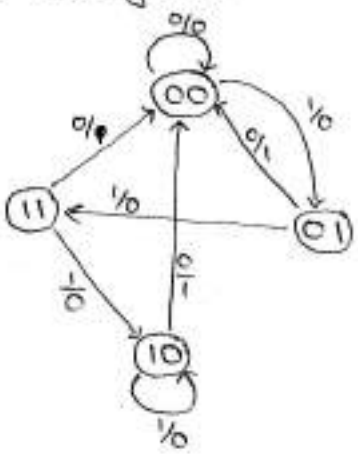
for D-FF,
O/P = I/P.
$\therefore D_A = A(t+1)$
$D_B = B(t+1)$

(ii) State table

Present State		Next State				o/p (Y)	
		X=0		X=1			
A	B	A(t+1)	B(t+1)	A(t+1)	B(t+1)	X=0	X=1
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

(iii) State diagram

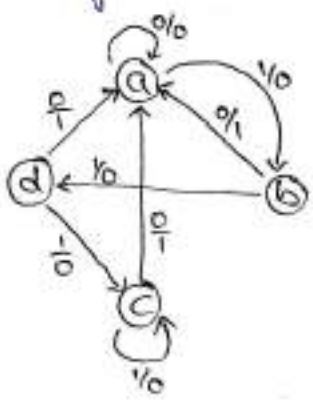
considering a=00, b=01
c=10, d=11.



present state	Next state		o/p(y)	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	a	d	1	0
c	a	c	1	0
d	a	c	1	0

(or)

considering a=00, b=01, c=10, d=11



28/10/2014

* Hazards :-

→ Hazards are unwanted switching transients that may appear at the o/p of a circuit because different paths exibit different propagation delays.

→ Hazards occur in combinational circuits, where they may cause a temparary false o/p value. When this condition occurs in sequential circuits, it may result in a transition to a wrong stable state.

Classification of Hazards :-

There are 3 different types of Hazards

(i) Static Hazard
- Static-0 Hazard
- Static-1 Hazard

(ii) Dynamic Hazard

(iii) Essential Hazard

1. (i) Static-1 Hazard :-

When a circuit output goes to momentarily '0', where it has to remain at constant '1', then we say that circuit has "Static-1 Hazard".

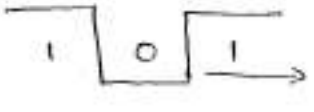


Fig:- Static-1 Hazard.

(ii) Static-0 Hazard :-

When a circuit output goes to momentarily '1', where it has to remain at constant '0',

then we say that circuit has "Static-0 Hazard".

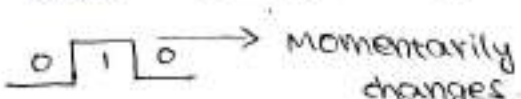


Fig:- Static-0 Hazard.

2. Dynamic Hazard :-

When the o/p is supposed to change from '0' to '1' (or) '1' to '0', if o/p changes more no. of times then this transient is called "Dynamic Hazard."

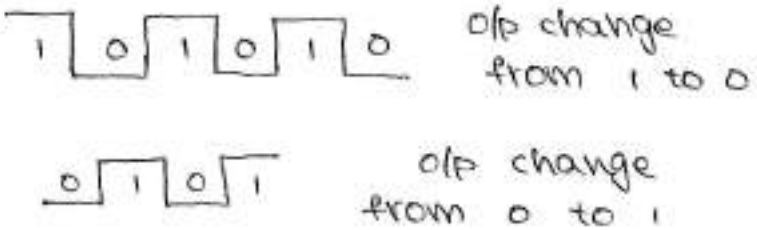
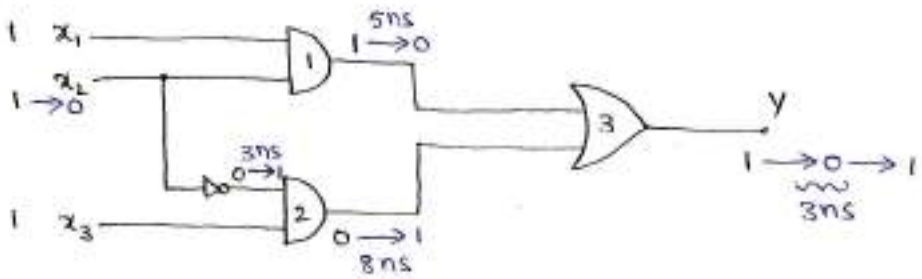


Fig:- Dynamic Hazard.

Combinational circuit with Static-1 Hazard



propagation delay of AND gate = 5 ns

NOT gate = 3 ns

∴ propagation delay of 1st path is 5 ns

2nd path is 8 ns.

This ckt displays wrong o/p '0' for 3ns when i/p changes from 111 ($x_1x_2x_3$) to 101 ($x_1x_2x_3$)

3. Essential Hazards:-

→ the Static and dynamic Hazards can occur in combinational as well as sequential logic circuits.

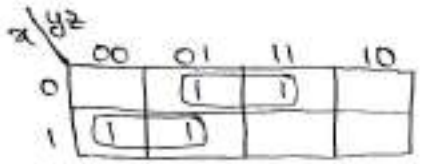
→ Essential Hazards occur in sequential circuits only.

→ An Essential Hazard is caused by unequal delays along two or more paths that originate from the same input.

Q.:- Realize the boolean function $T(x,y,z) = \Sigma(1,3,4,5)$ using logic gates for Hazard free.

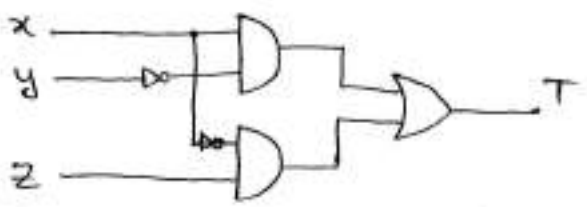
Sol:- K-Map simplification & logic circuit of above function T with Hazard are :

K-Map



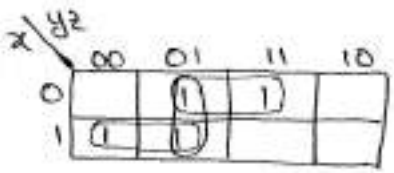
$\therefore T = x\bar{y} + \bar{x}z$

logic circuit



For designing Hazard free circuit

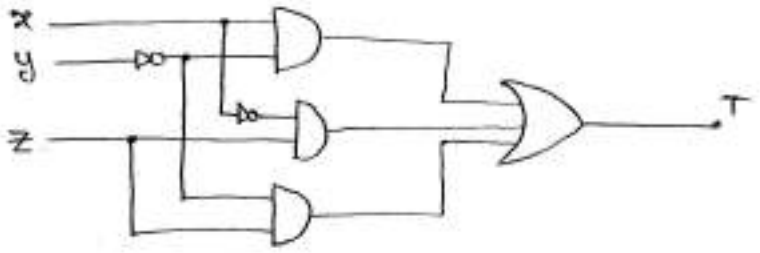
K-Map



(∵ combined all the possible groups)

$\therefore T = x\bar{y} + \bar{x}z + \bar{y}z$

Hazard free logic circuit:



22/09/2014

UNIT-IV

REGISTERS AND COUNTERS

Applications of Flip-flops :-

(i) It can be used as a memory element.

(ii) It can be used to eliminate key debounce.

(iii) It is used as a basic building block in sequential circuits such as counters and Registers.

(iv) It can be used as a delay element

(v)

REGISTERS :-

(i) Register is a group of flip-flops.

(ii) n-bit Register consists of 'n' number of flip-flops and it stores 'n' bit binary information.

Types of Shift Registers :-

1. Serial In serial OUT shift register (SISO) :-

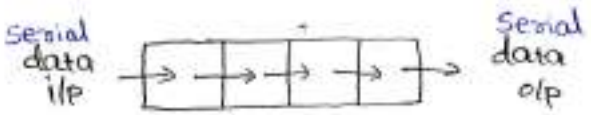


Fig: Data flow in SISO shift right register

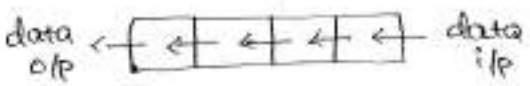


Fig: Data flow in SISO shift left register.

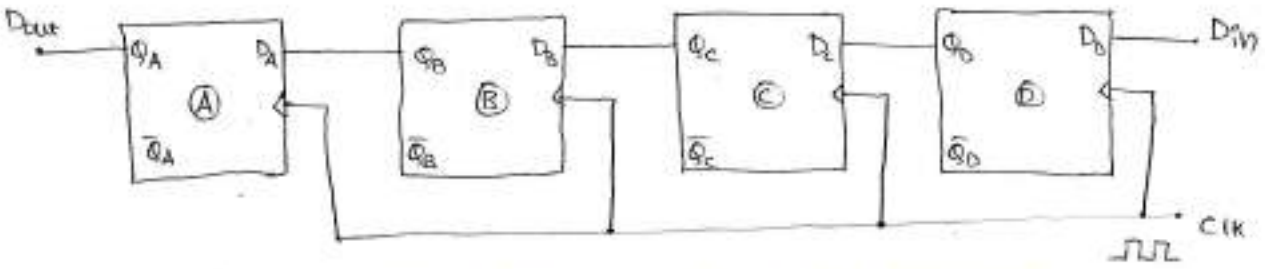


Fig: 4-bit SISO shift left register

i/p to reg 1011.

CLK	reg. o/p (Q_{out}) Q_A	Q_B	Q_C	Q_D	
0	0	0	0	0	
↑ (LSB)	0	0	0	1	
↑	0	0	1	0	
↑	0	1	0	1	
↑	1	0	1	1	→ i/p data stored in register.
↑	0	1	1	0	
↑	1	1	0	0	
↑	1	0	0	0	→ all 4 data i/p bits are transfered to o/p side
↑	0	0	0	0	→ reg. cleared.

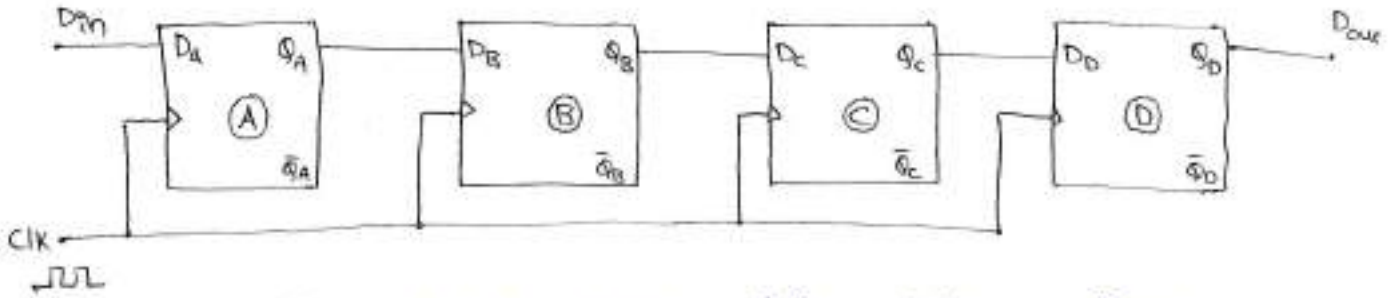


Fig:- 4-bit SISO shift right register.

2. Serial IN Parallel OUT shift Register (SIPO):-

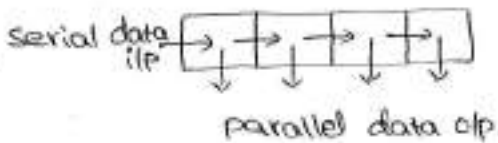


Fig:- data flow in SIPO shift right register.

In this case, data bits enters into register in serial, but o/p is taken in parallel.

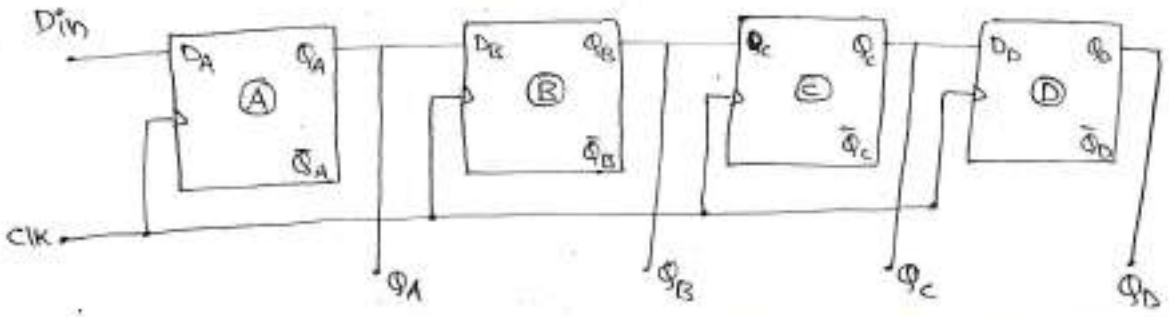


Fig: 4-bit SIPO shift right register.

D_{in} → Serial data i/p

$Q_A\ Q_B\ Q_C\ Q_D$ → 4-bit parallel data o/p.

Register with parallel load :-

* the transfer of new information into a register is refered to as loading Register.

* If all the bits of Register are loaded simultaneously with a common clock pulse, we say that the loading is done in parallel.

3. Parallel IN parallel OUT shift register (PIPO) :-

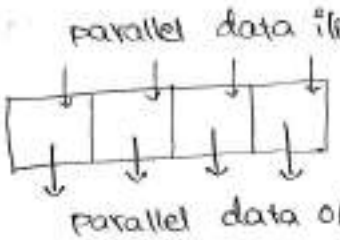


Fig: data flow in PIPO shift register.

Data bits are entered into reg in parallel and o/p is taken in parallel.

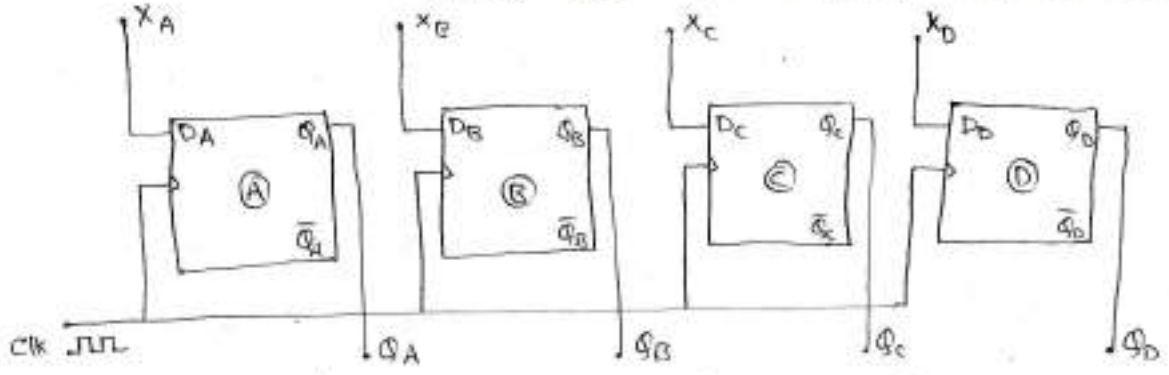


Fig: 4-bit PIPO shift register.

$X_A X_B X_C X_D$ → parallel data i/p's

$Q_A\ Q_B\ Q_C\ Q_D$ → 4-bit parallel data o/p.

07/10/2014

4. Parallel In serial Out shift Register (PISO) :-

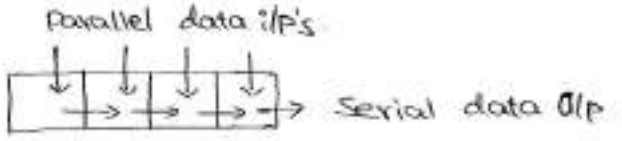


Fig: Dataflow in PISO shift right register.

I/P data entered into register parallely but o/p is taken serially.

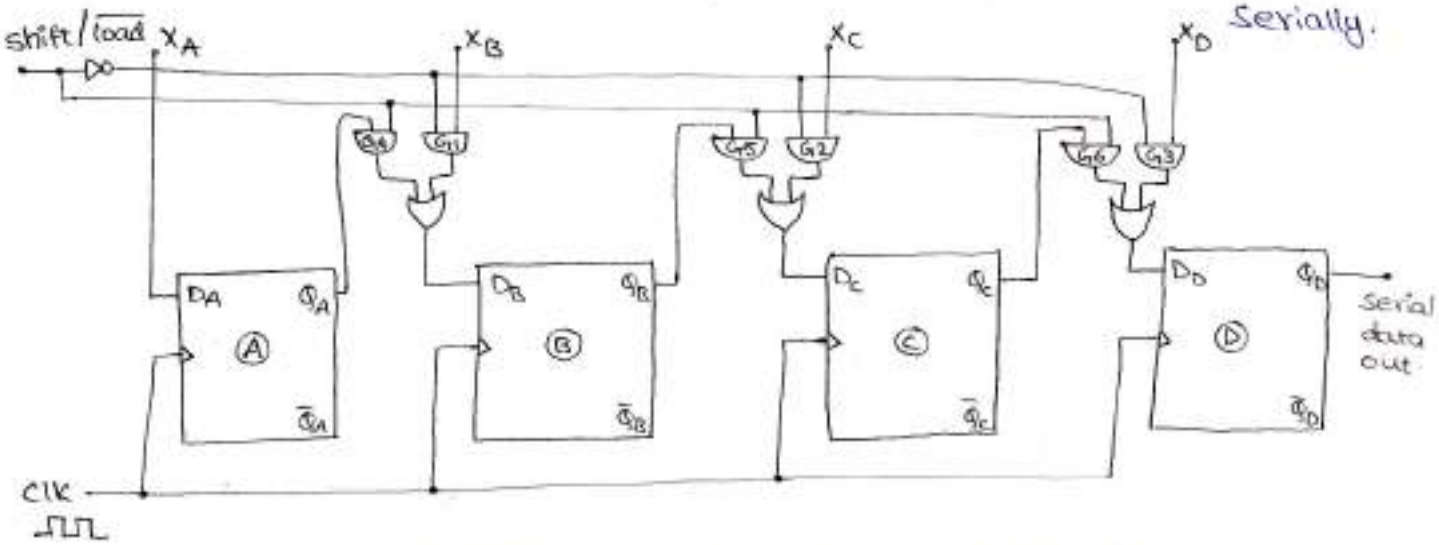


Fig: 4-bit PISO shift right register.

$X_A X_B X_C X_D$ → parallel data i/p's

If shift/$\overline{load}$ = 0 then parallel data loads into reg.

when shift/$\overline{Load}$ = 1, reg shifts the stored data to right

* Asynchronous (or) direct i/p's of flip-flops :-

(i) Set (or) Pre-Set :-

If set i/p is high then flip-flop o/p is '1' (for any data i/p) irrespective of data i/p's & clk i/p

If set is high ⇒ 1 ⇒ then flip-flop will set
If $\overline{set}$ is low ⇒ 0 ⇒ then flip-flop will set.

(ii) clear (or) reset :-

If reset i/p is high then flip-flop o/p is '0' (for any data i/p) irrespective of clk & data i/p's

* Universal shift register :-

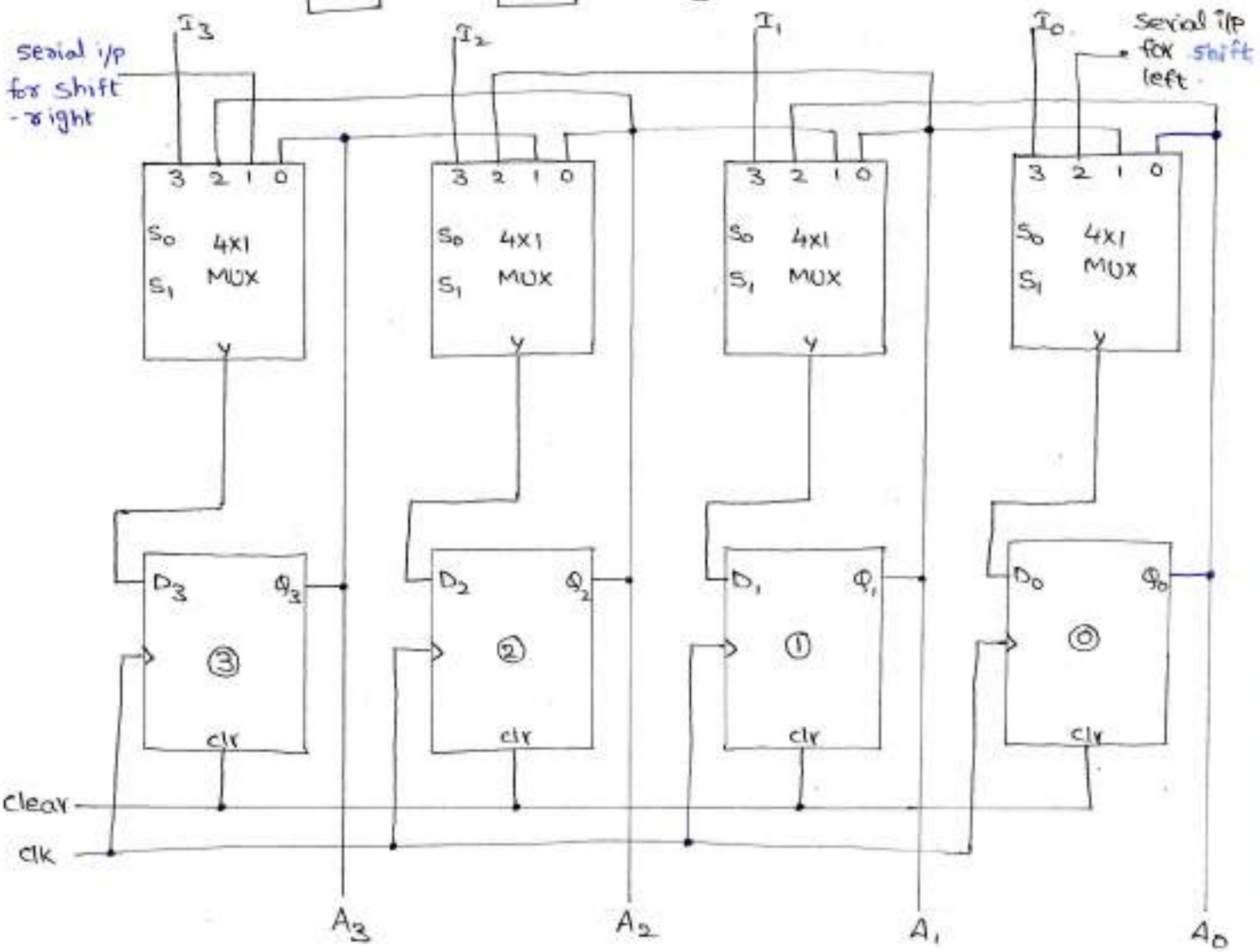
If the register has both shift (left & Right) and Parallel load capabilities, it is refered to as universal shift register.

→ acts like PISO, PIPO, SISO, SIPO

→ acts like shift (right & left)

→ (i.e) performs all operations.

Note :- clr ≡ $\overline{clr}$ } both are same.



S_0, S_1 → are common selection i/p's, but not o/p's of previous mux. For simplicity we have drawn in the above way.

Fig:- 4-bit universal shift register.

Function table :-

Mode control S_1	S_0	Reg. operation
0	0	NO change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

$I_3 I_2 I_1 I_0 \rightarrow$ parallel i/p's

$A_3 A_2 A_1 A_0 \rightarrow$ parallel o/p's.

for '00' we observe all '0' connections in MUX, gives o/p to D_1, D_2, D_3, D_0.

Similarly, for '11' we observe all '3' connections in MUX, gives o/p to D_0, D_1, D_2, D_3.

* Serial Adder :-

It performs serial addition.

Block diagram :-

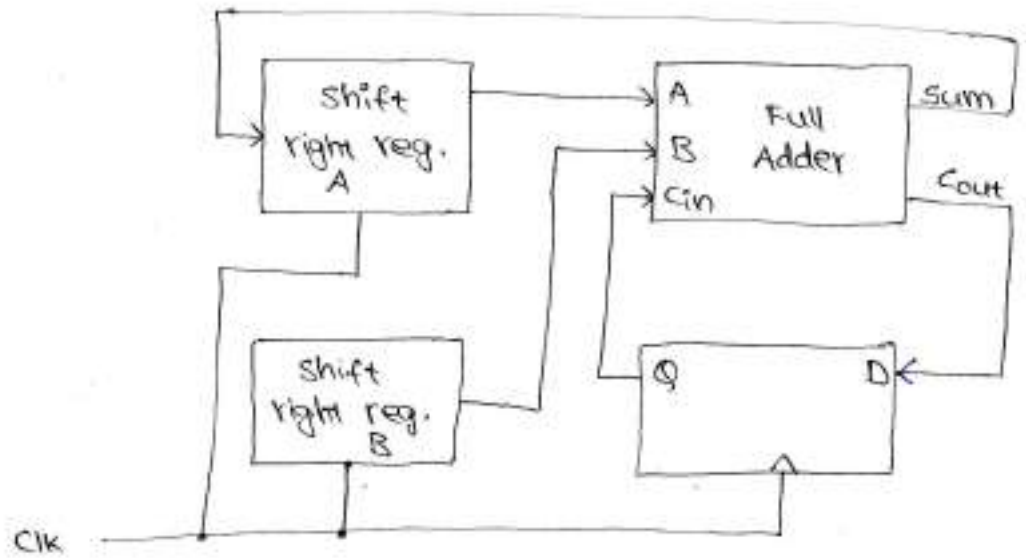


Fig: Serial Adder

let us consider two 4-bit numbers →

```
  0 1 1  ← Intermediate carry
  1 0 1 1
+ 1 0 0 1
-------
1 0 1 0 0
end carry  sum
```

considering A & B are 4 bit registers (SISO shift right reg.)

A reg. content is 1011.

B reg. content is 1001.

serial adder performs operation.

$A \leftarrow A+B$ Arrow indicates dataflow direction.

when clk is applied Reg's performs shift right operation.

During shift right, LSB comes out & goes to full Adder.

and FF adds A, B, C_{in} bits and transfers sum to A reg. & C_{out} to D-FF.

After performing serial addition contents of

reg A → 0100 (sum)

B → 0000

FF D → 1 (end carry)

Serial Transfer :-

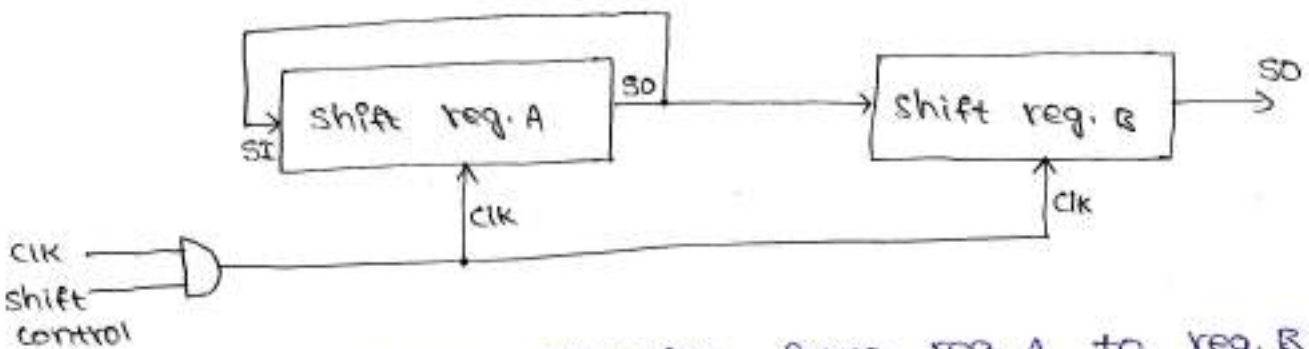


Fig:-(a) serial transfer from reg. A to reg. B.

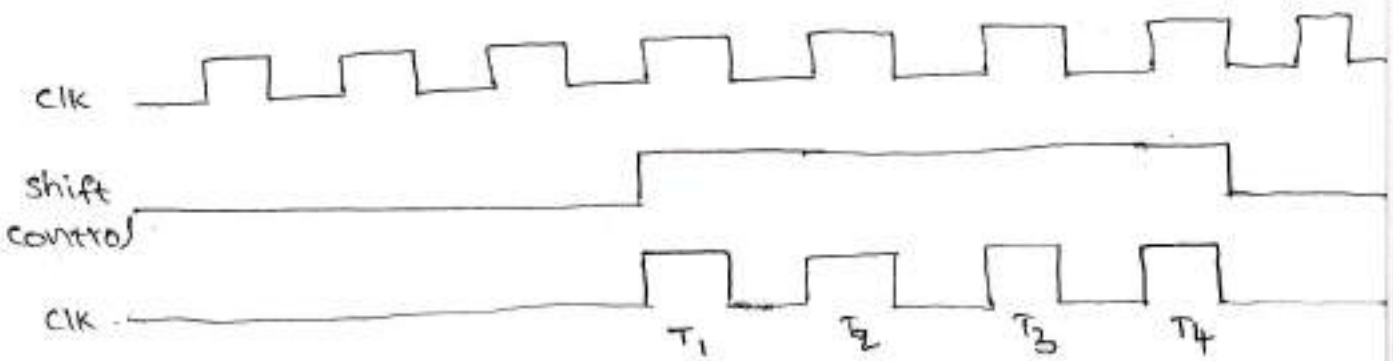


Fig:-(b) Timing diagram

considering A & B are 4-bit SISO shift right reg.'s.

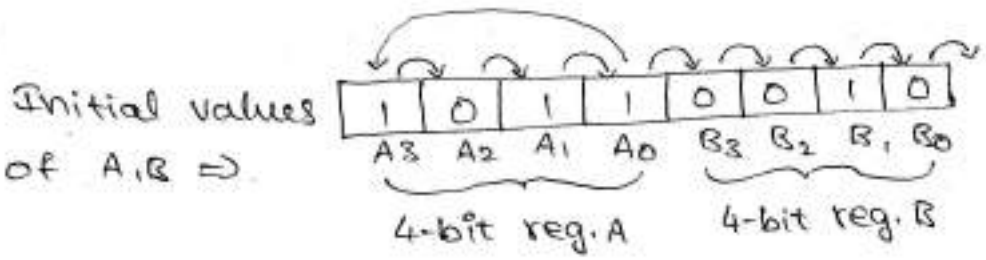
SO → Serial out ; SI → Serial In.

The above ckt performs operation

$B \leftarrow A$ (or) $A \rightarrow B$ Arrow indicates direction of dataflow.

After performing register transfer operation $A = B$

Fig (a) performs shift operation only when shift control is high because reg.'s receives clk i/p only when shift control = 1.



Serial transfer

Timing pulse	Shift reg. A	Shift reg. B
Initial value of reg.	1011	0010
After T_1	1101	1001
T_2	1110	1100
T_3	0111	0110
T_4	1011	1011

After T_4, both A & B reg.'s content is same.

(i.e) $A = B$.

COUNTERS :-

A counter is a register capable of count-ing the number of clock pulses arriving at it clock input. Count represents the number of clock pulses arrived.

(or)

A counter is an essentially a register that goes through a predetermined sequence of binary states.

* counters are available in two categories.

1. Asynchronous counters.
2. Synchronous counters

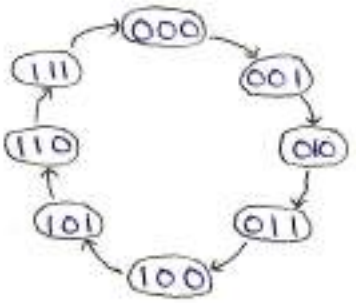
Asynchronous counters	synchronous counters
1. In this type of counter, flip-flops are connected in such a way that o/p of 1st FF drives the clk for the next FF.	1. In this type, there is no connection b/w o/p of 1st FF and clk i/p of the next FF.
2. All the FF are not clocked simultaneously.	2. All the FF are clocked simultaneously.
3. logic circuit is very simple even for more number of states	3. Design involves complex logic circuits as no. of states increases.
4. Main drawback of these counters is their low speed as the clk is propagated through no. of FF before it reaches last FF.	4. As the clk is simultaneously given to all FFs there is no problem of propagation delay. Hence they are high speed counters

Asynchronous (or) Ripple (or) serial counters (or) Non-synchronous counter:-

1. 3-bit Asynchronous up counter :-

An upcounter is a counter which counts in the upward direction. (i.e) 0, 1, 2, 3, ... n

State diagram :-



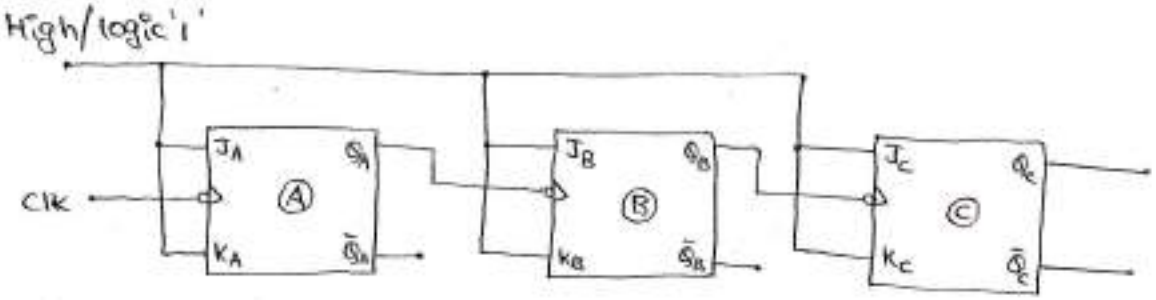
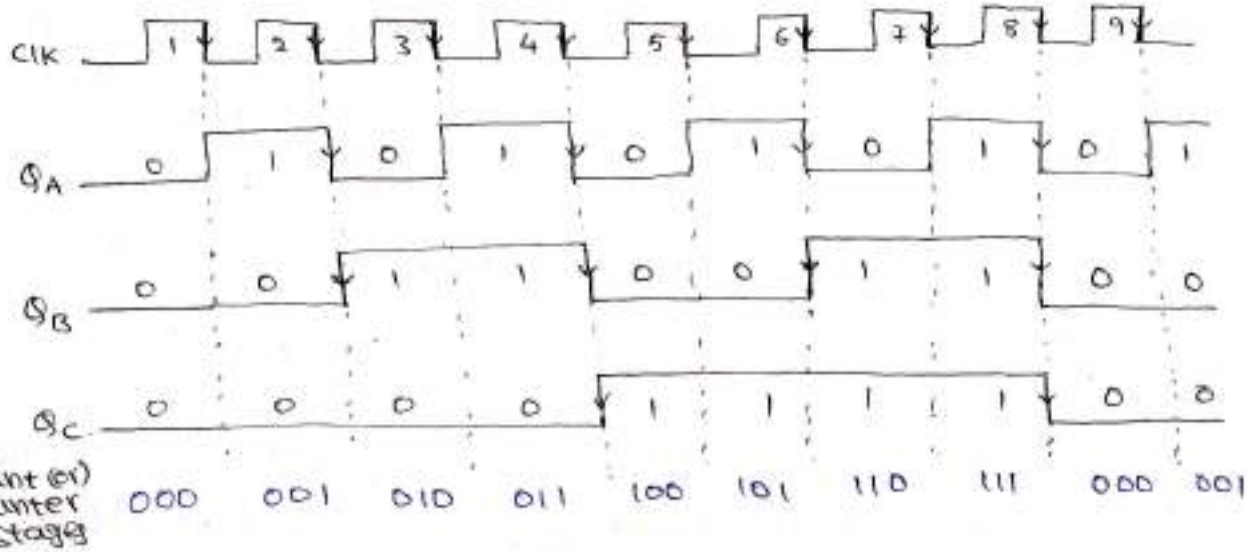


Fig:- 3-bit ripple up counter using -ve edge triggered Jk FF's.

counter o/p → Q_C Q_B Q_A
MSB LSB

o/p waveforms



* the number of states through which the counter passes before returning to the starting state is called the Modulus of the counter.

(or)

The Mod number of a counter is the total no. of states it sequences through in each complete cycle

* Since a 3-bit counter has 8 states it is called a Mod-8 (or) Modulus-8 counter. (or) Modulo-8 counter.

3-bit ripple up counter using T-FF's.

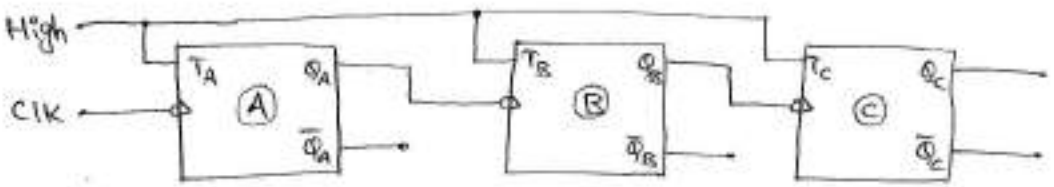


Fig:- 3-bit Asynchronous up counter using -ve edge triggered T-FF's.

counter o/p → Q_C Q_B Q_A
MSB LSB.

Note:- For designing ripple up counter using +ve edge triggered FF's connect $\bar{Q}$ o/p's to clk i/p's of next flip-flops.

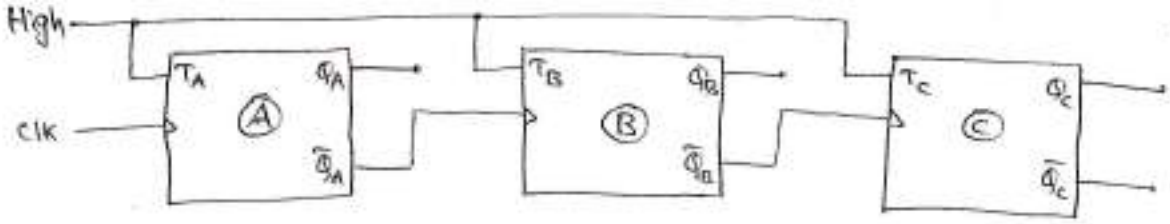


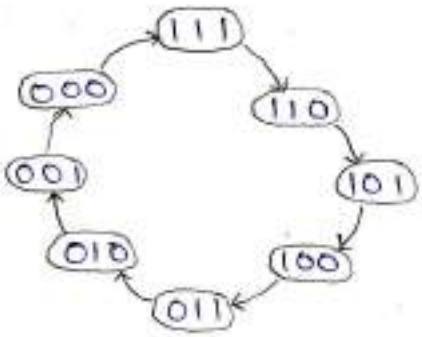
Fig:- 3-bit ripple up counter using +ve edge triggered T-FF's.

Counter o/p → Q_C Q_B Q_A
MSB LSB.

2. 3-bit Asynchronous down counter :-

Down counter counts in downward direction. (i.e) n, n-1, n-2, ..., 2, 1, 0.

State diagram:-



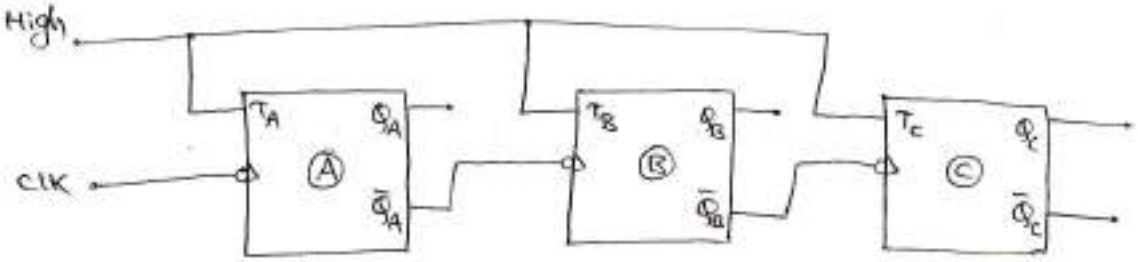
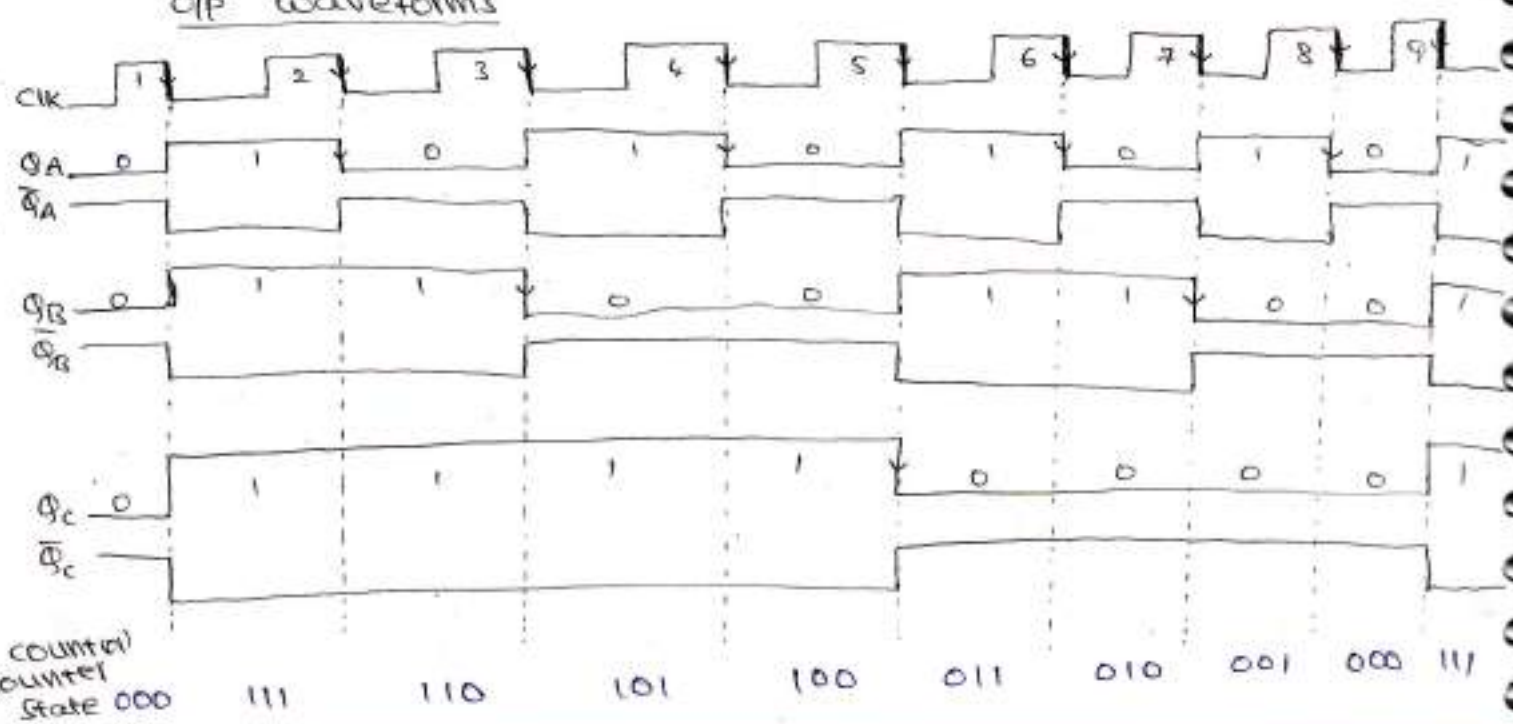


Fig:- 3-bit ripple down counter using -ve edge triggered T-FF's

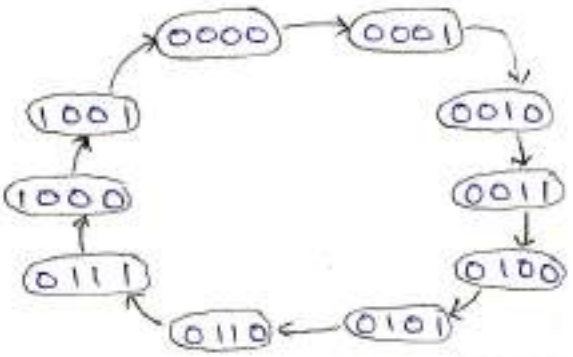
counter o/p → Q_C Q_B Q_A
MSB LSB

O/p waveforms



3. Asynchronous Mod-10 counter (or) Decade counter (or) BCD counter :-

State diagram :-



for NAND gate draw bubbles for clr i/p

for AND gate Don't draw bubbles

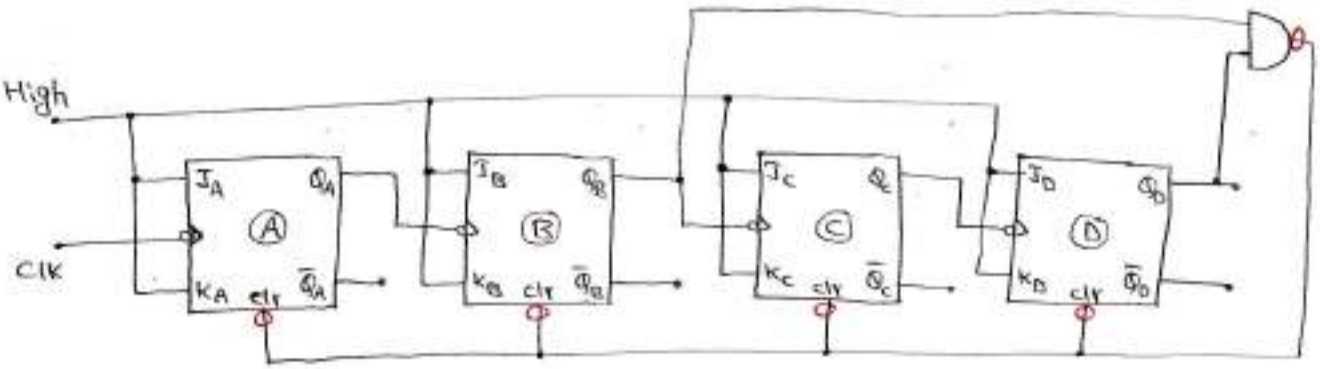


Fig:- MOD-10 ripple counter.

counter o/p → $Q_D\ Q_C\ Q_B\ Q_A$ (MSB ... LSB)

* When counter o/p is $Q_D Q_C Q_B Q_A$ 1010, o/p of And gate becomes 1. Since o/p of And gate is connected to reset (or) clear i/p's of FF's, counter enters into 0000 state after 1010 state.

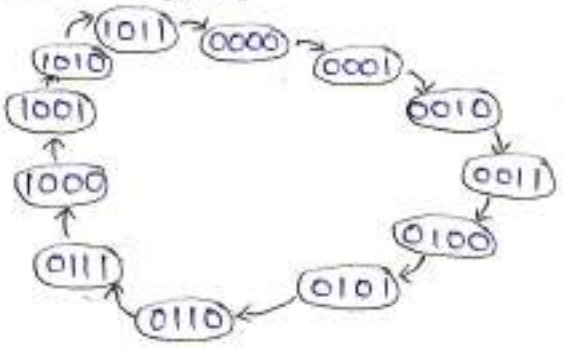
Truth table :-

CLK	Q_D	Q_C	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

CLK	Q_D	Q_C	Q_B	Q_A
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0
11	0	0	0	1
12	0	0	1	0
13	0	0	1	1
14	0	1	0	0

4. Asynchronous Mod-12 counter :-

State diagram :-



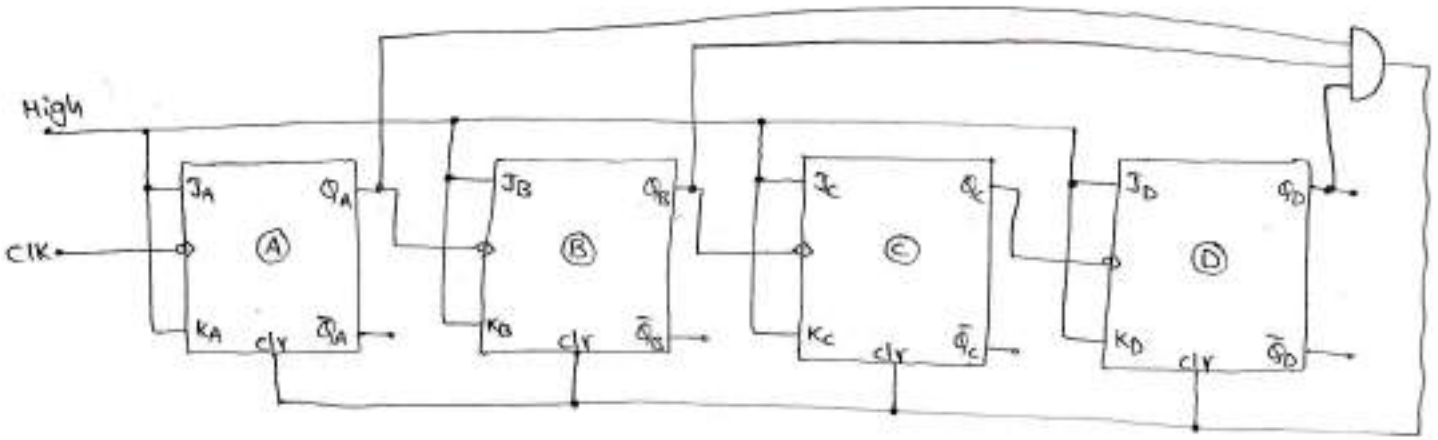


Fig: MOD-12 ripple counter

counter o/p → $Q_D Q_C Q_B Q_A$
MSB LSB

* When counter o/p is $Q_D Q_C Q_B Q_A$ 1011, o/p of And gate becomes 1. Since o/p of And gate is connected to reset (or) clear i/p's of FF's, counter enters into 0000 state after 1011 state.

Truth table :-

CLK	Q_D	Q_C	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

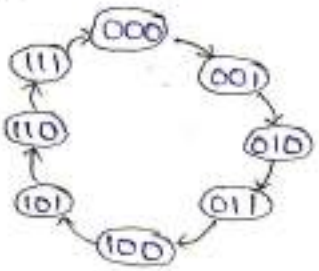
CLK	Q_D	Q_C	Q_B	Q_A
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	0	0	0	0
13	0	0	0	1
14	0	0	1	0
15	0	0	1	1
16	0	1	0	0
17	0	1	0	1

14/10/2014

Synchronous (or) Parallel counters :-

1. 3-bit synchronous Binary up counter (or) Mod-8 Sync. counter :-

State diagram :-



Logic diagram :-

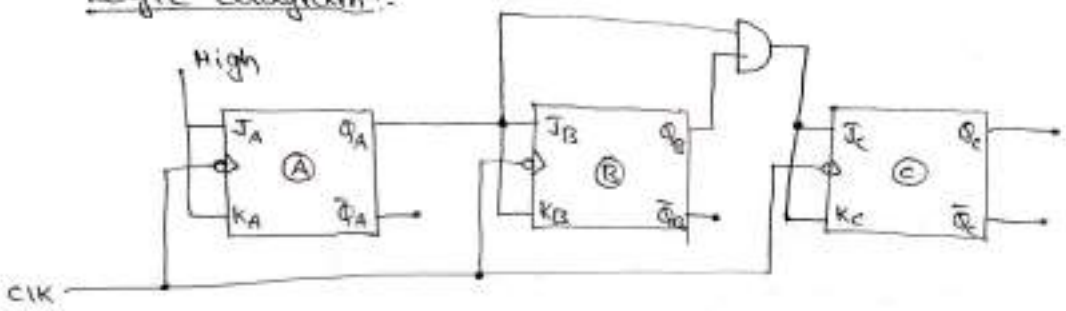


Fig :- MOD-8 Sync. counter

FF I/P's

$J_A = K_A = 1$

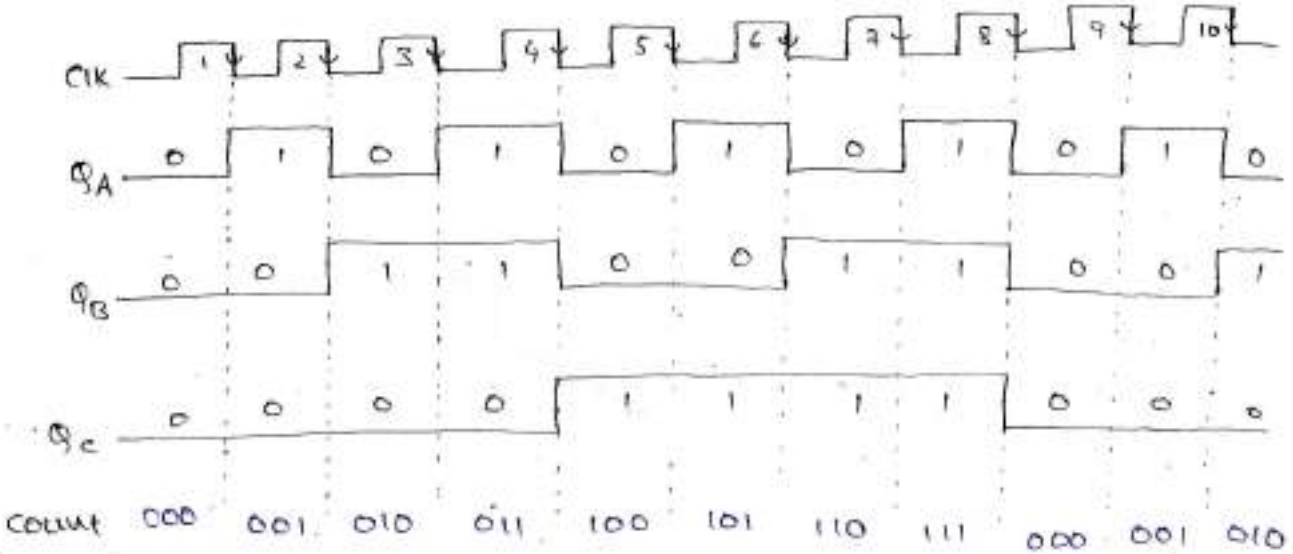
$J_B = K_B = Q_A$

$J_C = K_C = Q_A \cdot Q_B$

counter o/p $\longrightarrow$ Q_C Q_B Q_A

MSB LSB

output waveforms :-



2. 4-bit Synchronous Binary up counter (or) MOD-16 counter:-

Block diagram:-

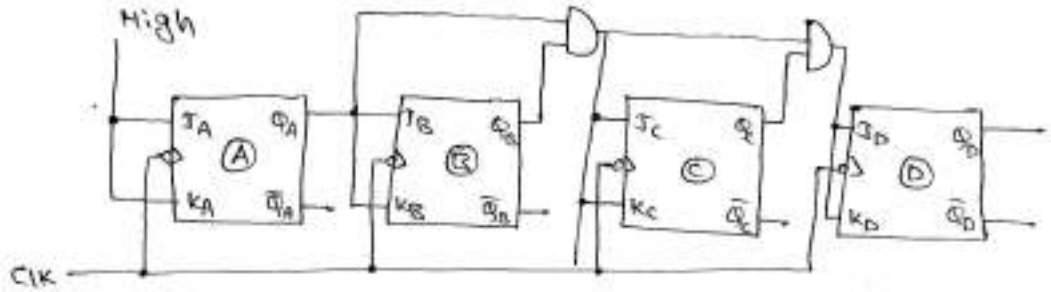


Fig:- MOD-16 Binary up counter

FF i/p's

$J_A = K_A = 1$

$J_B = K_B = Q_A$

$J_C = K_C = Q_A \cdot Q_B$

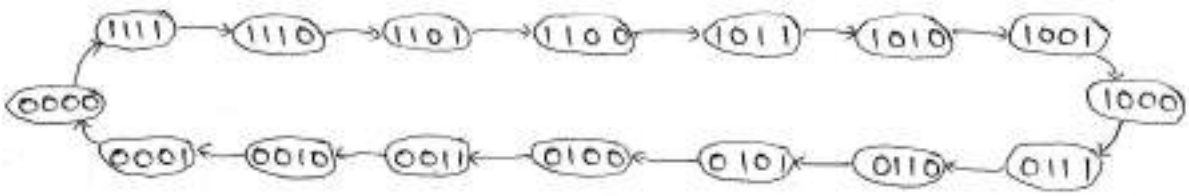
$J_D = K_D = Q_A \cdot Q_B \cdot Q_C$

Counter o/p → Q_D Q_C Q_B Q_A

MSB LSB

3. 4-bit Synchronous Down counter (or) MOD-16 Down counter:-

State diagram:-



Logic diagram:-

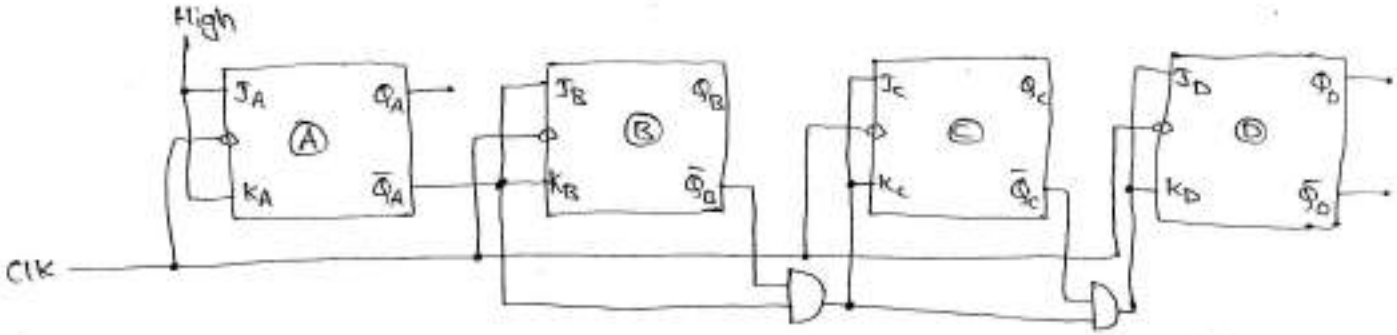


Fig:- MOD-16 Down counter

FF i/p's

$J_A = K_A = 1$

$J_B = K_B = \bar{Q}_A$

$J_C = K_C = \bar{Q}_A \bar{Q}_B$

$J_D = K_D = \bar{Q}_A \bar{Q}_B \bar{Q}_C$

counter o/p → Q_D Q_C Q_B Q_A
MSB LSB

Truth table (or) Function table :-

CLK	counter o/p Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1

CLK	counter o/p Q_D	Q_C	Q_B	Q_A
10	0	1	1	0
11	0	1	0	1
12	0	0	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0
17	1	1	1	1
18	1	1	1	0
19	1	1	0	1

* Design of Synchrous counters :-

Q. Design synchrous MOD-5 counter using JK flip-flops.

Sol:- Note:- If counter type is not mentioned then design up counter

Step 1:- Determine number of flip-flops.

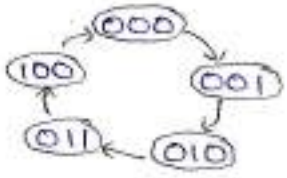
The no. of FF's required to design MOD-5 Synchronous up counter can be determine by the equation, $2^n \geq N$ where $n \rightarrow$ no. of FF's
$N \rightarrow$ MOD no.

the possible value of 'n' which satisifies the above equation is 3.

thus MOD-5 counter uses 3 FF's.

Step 2:- Flip-flop type - JK FF.

Step 3 :- State diagram



Step 4 :- Excitation table

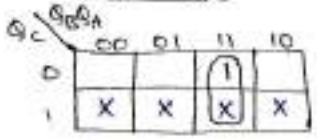
Present stage			Next stage			FF i/p's					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

∵ JK FF Excitation table

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

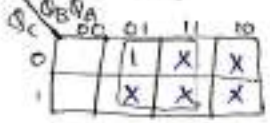
Step 5 :- K-Map simplification

for J_C



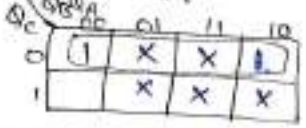
$\therefore J_C = Q_B Q_A$

for J_B



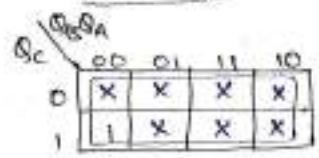
$\therefore J_B = Q_A$

for J_A



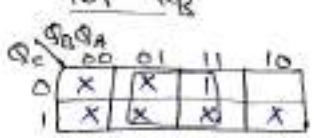
$\therefore J_A = \bar{Q}_C$

for K_C



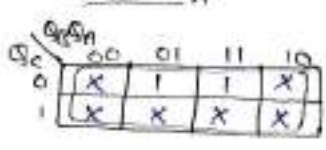
$\therefore K_C = 1$

for K_B



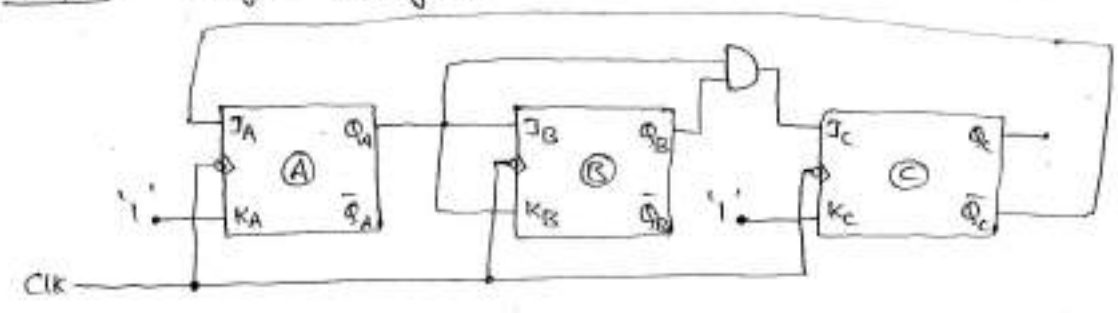
$\therefore K_B = Q_A$

for K_A



$\therefore K_A = 1$

Step 6 :- Logic diagram



Solved

Q. Design MOD-10 synchronous up counter using T-Flip flops (Decade counter / BCD counter / MOD-10 counter).

Sol:

Step 1: Determine no. of FF's required.

$2^n \geq N$ where $N \rightarrow$ Mod No.
$n \rightarrow$ no. of FF's.

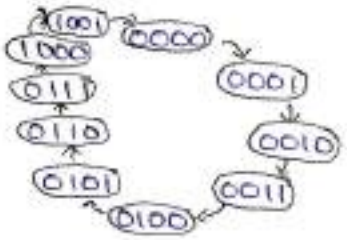
$2^n \geq 10$

$n = 4$

∴ No. of FF's required = 4

FF Type $\longrightarrow$ T-FF

Step 2: State diagram



Step 3: Excitation table

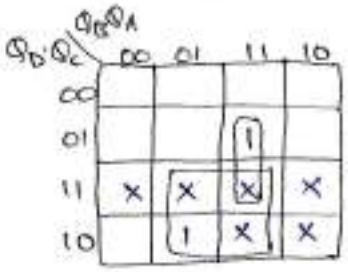
Present Stage				Next Stage				FF i/p's			
Q_D	Q_C	Q_B	Q_A	Q_{D+1}	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	x	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x

∵ T-FF excitation table

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

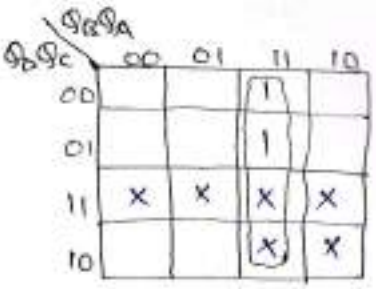
Step 4 : K-Map simplification

for T_D



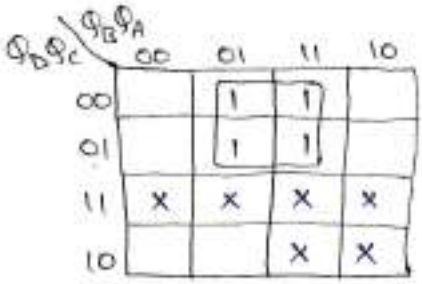
$\therefore T_D = Q_D Q_A + Q_C Q_B Q_A$

for T_C



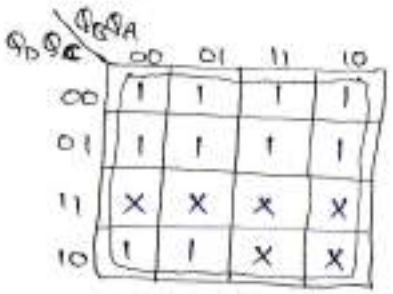
$\therefore T_C = Q_B Q_A$

for T_B



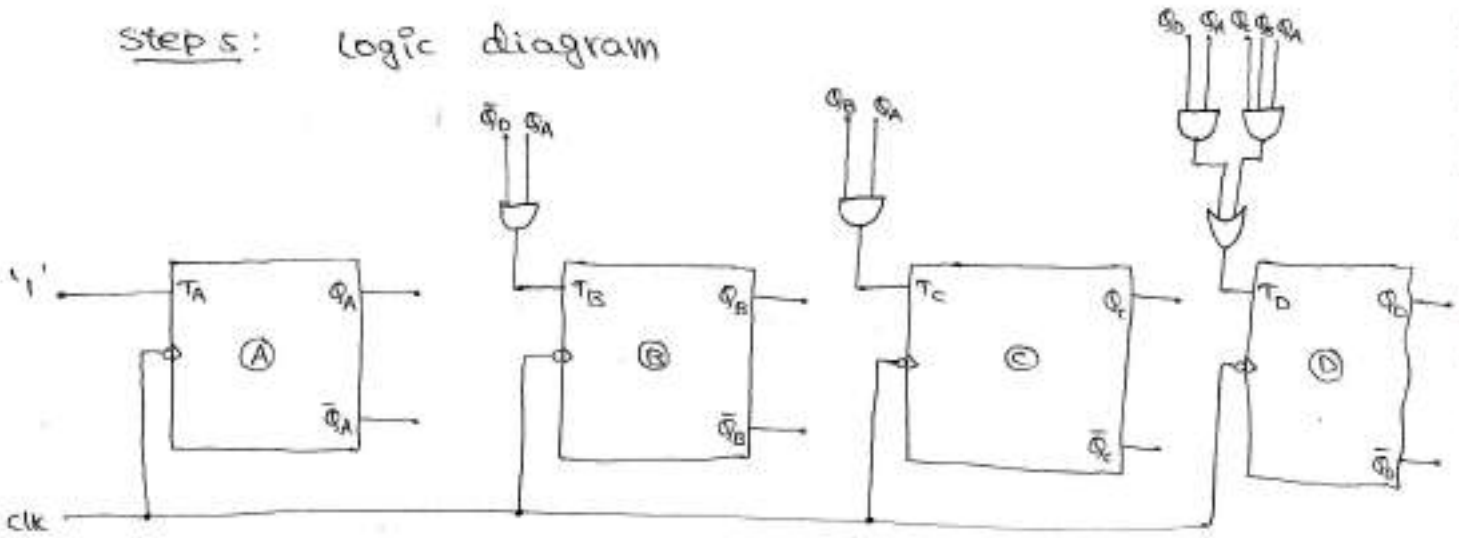
$\therefore T_B = \bar{Q}_D Q_A$

for T_A



$\therefore T_A = 1$

Step 5 : Logic diagram



Counter O/P → Q_D Q_C Q_B Q_A

MSB LSB

Counter with unused states :-

Q. Design a modulo-5 counter to count the sequence 0,1,3,7,6. Design should include circutary to ensure that if we end up in an unwanted state, the next clock pulse will reset the counter to $Q_2 \cdot Q_1 \cdot Q_0 = 000$ use T-Flipflops.

Sol:- **Step 1:** no. of FF's required to design MOD-5 counter can be determined by equation,

$2^n \geq N$ where $n \rightarrow$ no. of FF's
$N \rightarrow$ MOD no.

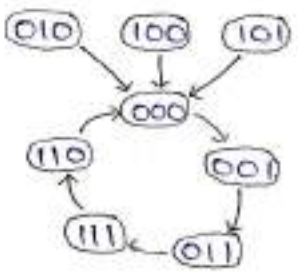
$\therefore 2^n \geq 5$

$\therefore n = 3$

$\therefore$ no. of FF's required $\longrightarrow$ 3
FF's using $\longrightarrow$ T FF's

Step 2: State diagram

used states $\longrightarrow$ 0,1,3,7,6
unused states $\longrightarrow$ 2,4,5



(∵ for unused states next stage is 000)

Step 3: Excitation table

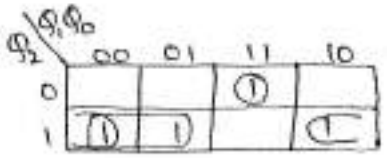
Present State			Next State			FF i/p's		
Q_2	Q_1	Q_0	Q_{2+1} (00) Q_{2+}	Q_{1+1} Q_{1+}	Q_{0+1} Q_{0+}	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	0	0
1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	1	0	1
1	1	0	0	0	0	1	1	0
1	1	1	1	1	0	0	0	1

∵ T-FF excitation table

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

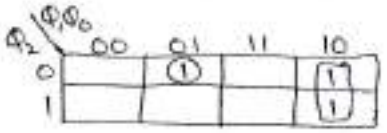
Step 4 : K-Map simplification

for T_2



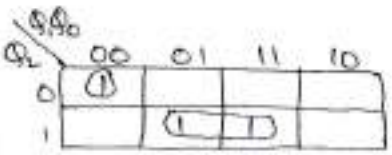
$\therefore T_2 = Q_2\bar{Q}_1 + Q_2\bar{Q}_0 + \bar{Q}_2Q_1Q_0$

for T_1



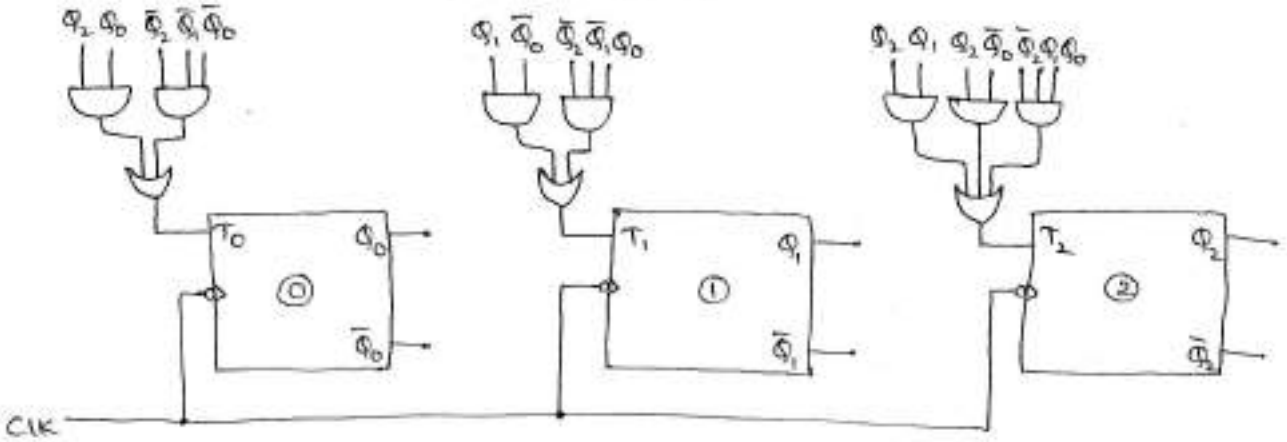
$\therefore T_1 = Q_1\bar{Q}_0 + \bar{Q}_2\bar{Q}_1Q_0$

for T_0



$\therefore T_0 = \bar{Q}_2\bar{Q}_1\bar{Q}_0 + Q_2Q_0$

Step 5 : Logic diagram



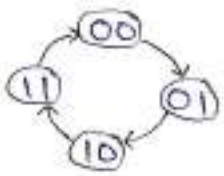
counter o/p → Q_2 Q_1 Q_0
MSB LSB

Q. Design a synchronous counter with states 0,1,2,3, 0,1, ... using JK FF's.

Sol: counter states are 0,1,2,3,0,1, ...

∴ It is MOD-4 counter.

State diagram :-



Q. Design synchronous counter using JK FF. to count 12 the following sequence.

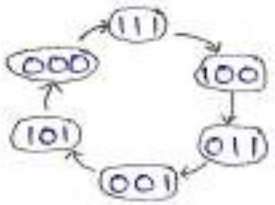
7, 4, 3, 1, 5, 0, 7, ...

Sol:- Total no. of states → 6

∴ It is MOD-6 counter.

Used states → 7, 4, 3, 1, 5, 0.

State diagram:-



Unused states → 2, 6.

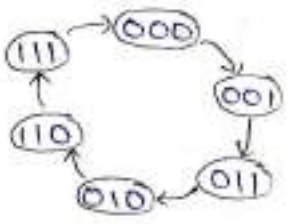
Treat unused states as don't cares.

Q. Design a synchronous gray code MOD-6 up counter.

Sol:-

	Binary	Gray
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1

State diagram:-



Unused states are 4, 5.

∴ treat them as don't cares.

* Synchronous UP/Down counter (or) Multimode counter (or) Bidirectional counter :-

Q. Design 3-bit synchronous up/down counter.

Sol: → the counter which is capable of progressing in either direction (i.e) in ascending order (incrementing order) (or) descending order (decreasing order) through a certain counting sequence is known as up/down counter.

→ Usually, up/down operation of the counter is controlled by up/$\overline{\text{down}}$ (M) signal.

→ When M=1, the counter goes through up sequence (0, 1, 2, ---, n).

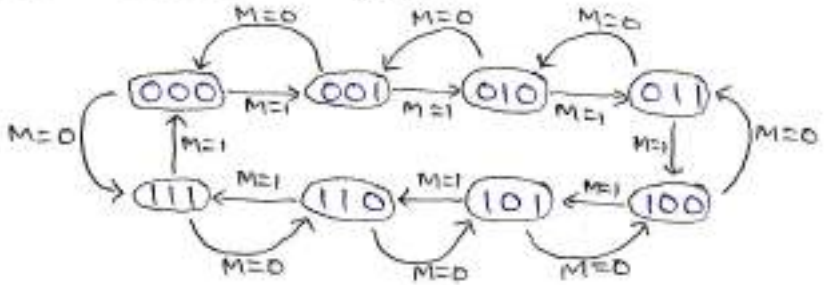
→ When M=0, the counter goes through Down sequence (n, n-1, ----, 2, 1, 0)

Step 1 :

no. of FF's required → 3 (∵ it is 3-bit counter)

FF's using → T FF's.

Step 2 : State diagram



Step 3: Excitation table

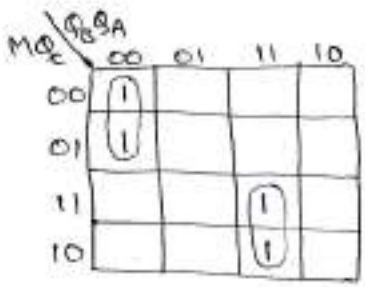
control i/p UP/$\overline{down}$ (M)		present State Q_C	Q_B	Q_A	Next Stage Q_{C+1}	Q_{B+1}	Q_{A+1}	FF i/p's T_C	T_B	T_A
Down	0	0	0	0	1	1	1	1	1	1
	0	0	0	1	0	0	0	0	0	1
	0	0	1	0	0	0	1	0	1	1
	0	0	1	1	0	1	0	0	0	1
	0	1	0	0	0	1	1	1	1	1
	0	1	0	1	1	0	0	0	0	1
	0	1	1	0	1	0	1	0	1	1
	0	1	1	1	1	1	0	0	0	1
UP	1	0	0	0	0	0	1	0	0	1
	1	0	0	1	0	1	0	0	1	1
	1	0	1	0	0	1	1	0	0	1
	1	0	1	1	1	0	0	1	1	1
	1	1	0	0	1	0	1	0	0	1
	1	1	0	1	1	1	0	0	1	1
	1	1	1	0	1	1	1	0	0	1
	1	1	1	1	0	0	0	1	1	1

∵ T-FF excitable table

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

Step 4: K-map simplification

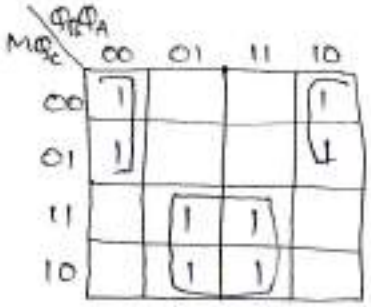
for T_C



$\therefore T_C = \bar{M}\bar{Q}_B\bar{Q}_A + MQ_BQ_A$

$= M \odot Q_B \odot Q_A$

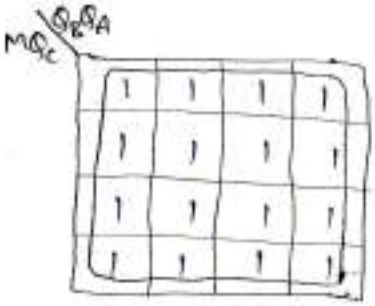
for T_B



$\therefore T_B = \bar{M}\bar{Q}_A + MQ_A$

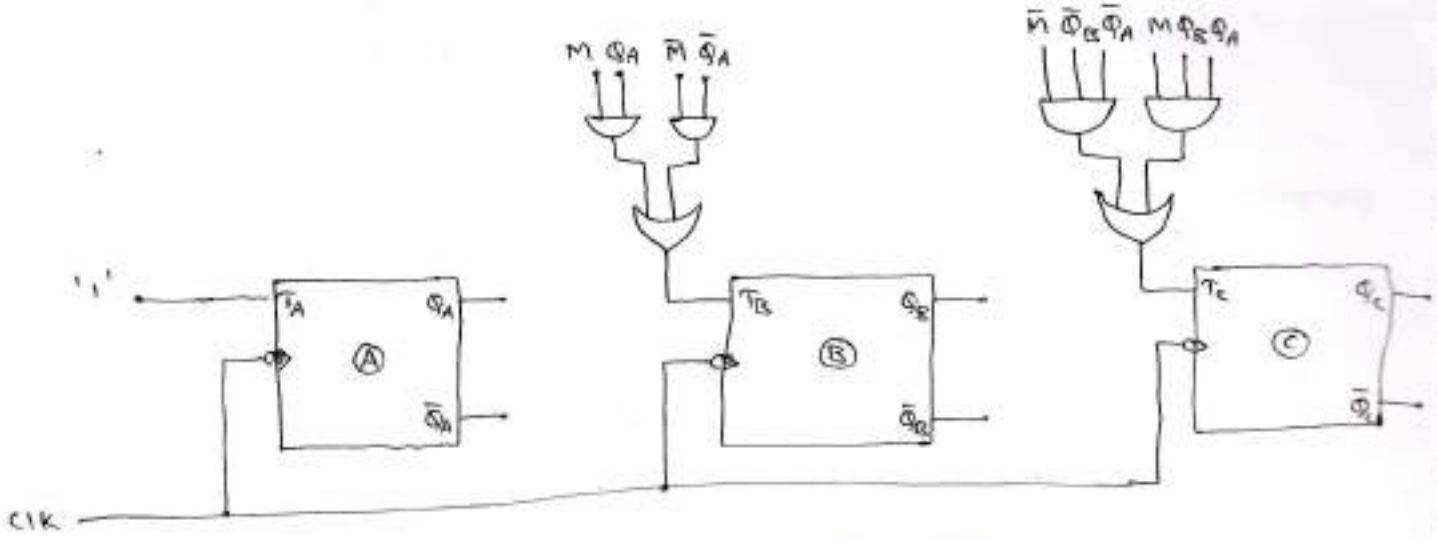
$= M \odot Q_A$

for T_A



$\therefore T_A = 1$

logic diagram :-



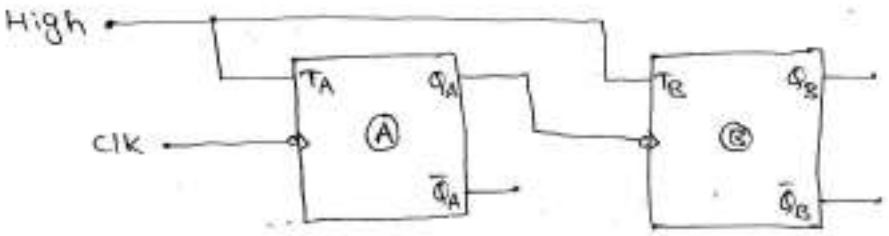
counter o/p → Q_C Q_B Q_A
MSB LSB

16/10/2014

* Frequency dividers :-

2-bit Asynchronous counter (or) MOD-4 counter (or) Divide-by-4 counter :-

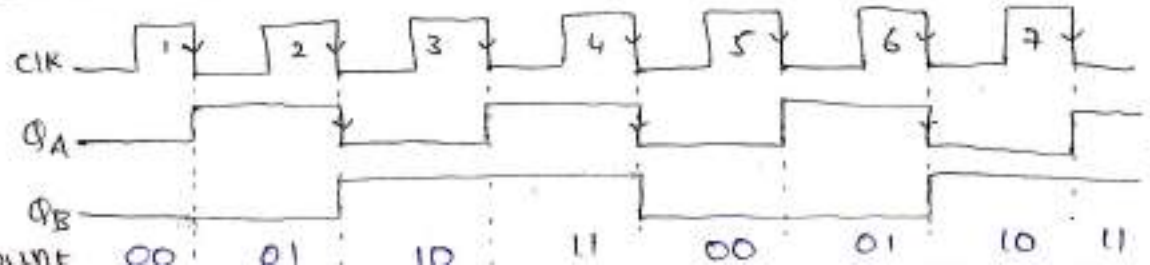
logic diagram :-



counter o/p → Q_B Q_A
MSB LSB

fig:- 2-bit ripple counter.

o/p waveform :- (or) Timing diag. of 2-bit ripple counter:-



* From the timing diagram it is clear that the frequency of Q_A is one half of the frequency of clk signal, Q_B is one half of Q_A and ~~th~~ is one fourth of the clk frequency.

* If the clk frequency is 1000 Hz, then Q_A = 500 Hz & Q_B = 250 Hz. Hence the 2-bit ripple counter is also called as divide-by-4 counter.

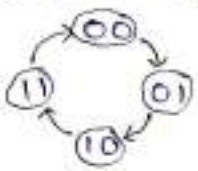
Note:- Each FF acts as a divide by 2 (÷2) frequency divider. (i.e) Each FF divides the incoming clk signal frequency by 2.

3-bit counter → MOD-8 counter → Divide-by-8 counter
4-bit counter → MOD-16 counter → Divide-by-16 counter
MOD-5 counter → Divide-by-5 counter
MOD-6 counter → Divide-by-6 counter

State table :-

The state table represents the state diagram in tabular form.

State diag:-



State table :-

Present State	Next State
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

* Lock out condition :-

In a counter if the next state of some unused state is again an unused state and if by chance the counter happen to find itself in the unused states and never arrived at a used state then the counter is said to be in the "lockout condition".

the circuit that goes in lockout condition is called "bustless circuit"

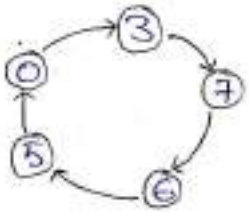


Fig:- a) Desired sequence

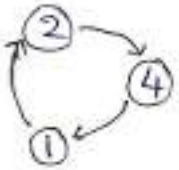


Fig:- b) Unused states forming lockout.

To avoid lockout condition, the unused states are introduced infront of the used states

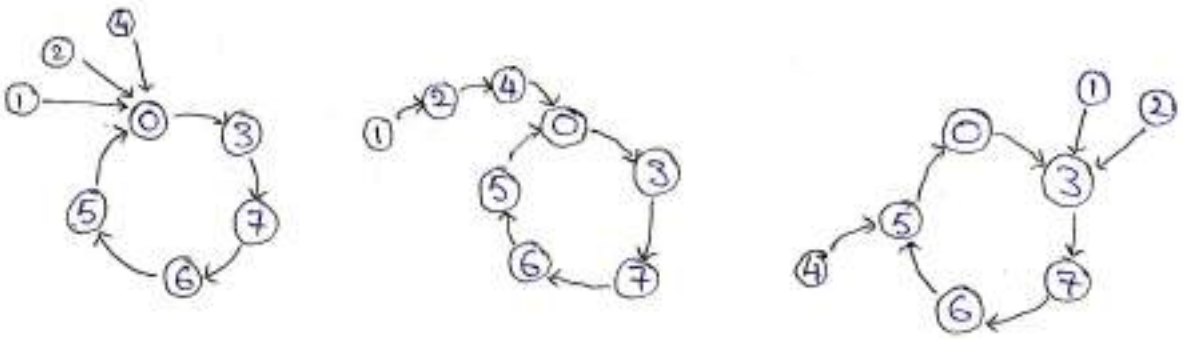
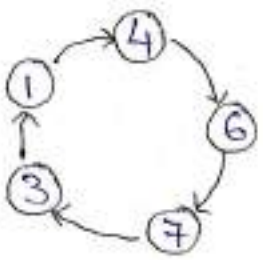


Fig:- state diagrams for removing lockout.

Q:- Design a synchronous counter for the following sequence. $4 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 4 \ldots\ldots$

Avoid lockout condition use JK type of design.

Sol:- State diagram:-



Used states $\rightarrow$ 1, 3, 4, 6, 7

Unused states $\rightarrow$ 0, 2, 5

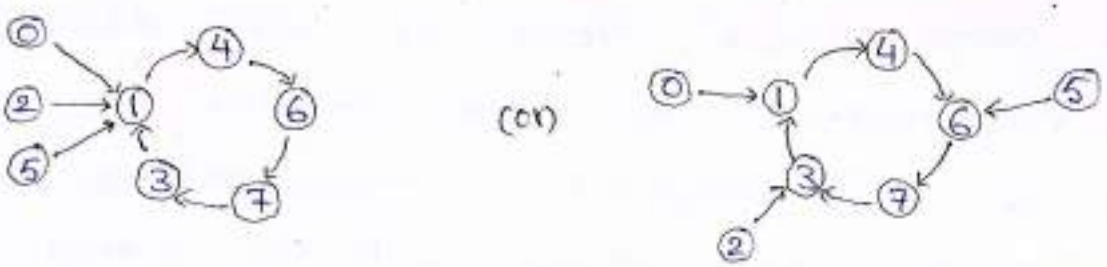


Fig:- state diagrams for Avoiding lockout condition.

* Shift Register counters :-

A shift Register counter is basically a shift register wch with the serial o/p connected back to the serial i/p to produce special sequences. these devices are classified as counters because they exhibit a special sequence of states.

Two of the most common types of shift Register counters are

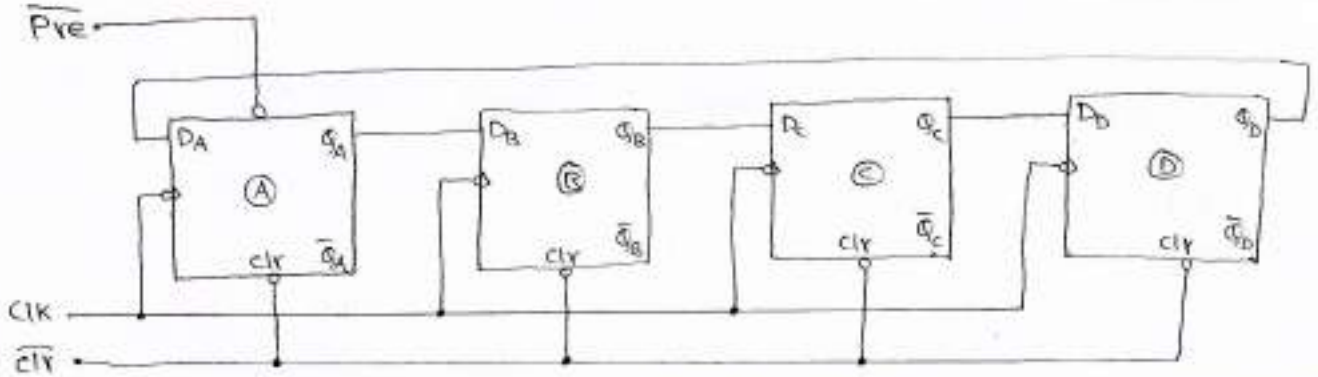
(i) Ring counter

(ii) Johnson counter

1. Ring counter :-

A Ring counter is a circular shift Register with only 1 FF being set at any particular time, all others are cleared.

The single bit is shifted from one FF to the next to produce the sequence of timing signals.



counter o/p → Q_D Q_C Q_B Q_A (MSB … LSB)

clr → clear (or) Reset i/p

Pre → set (or)

Fig:- 4-bit ring counter.

The above figure shows the logic diagram of 4-bit ring counter. As shown in the fig., the Q output of each stage is connected through the D input of the next stage and the output of the last stage is fedback to the i/p of first stage. the $\overline{clr}$ & $\overline{pre}$ i/p's are used to make the o/p of first stage to 1 and remaining o/p's to zeros. (i.e) $Q_A Q_B Q_C Q_D = 1000$.

Truth table:-

Clk	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

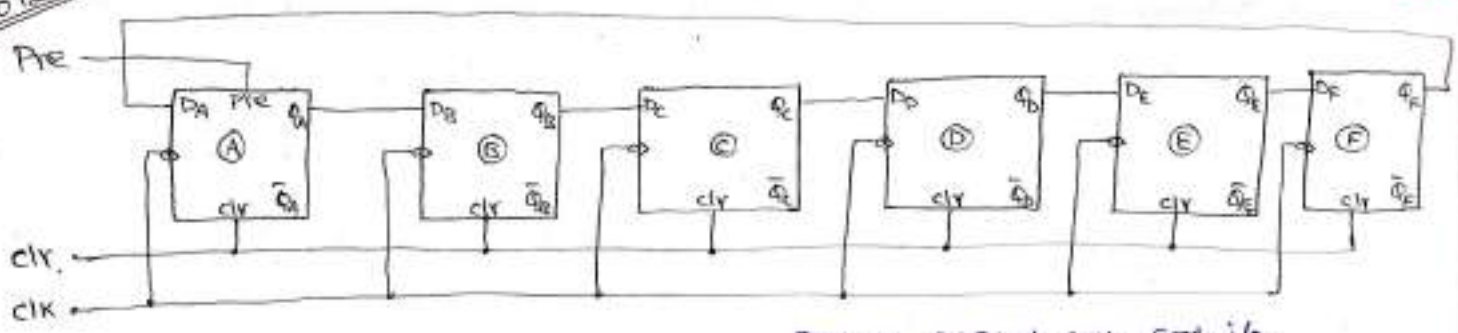
Since the 4-bit ring count has 4 distinct states, it is also known as a MOD-4 counter.

Note:- A MOD-N ring counter will require 'N' no. of flip-flops connected together to circulate a single data bit providing 'N' different o/p states.

Q:- Implement a MOD-6 Ring counter using suitable FF's.

Sol:-

21/10/2014



Pre → preset (or) set i/p

clr → clear (or) Reset i/p

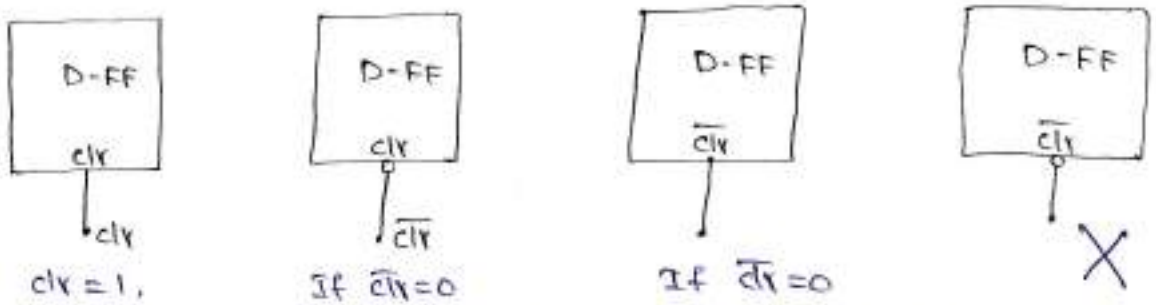
counter o/p → $Q_F\ Q_E\ Q_D\ Q_C\ Q_B\ Q_A$ (MSB … LSB)

Fig: MOD-6 Ring counter

Truth table :-

clk	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	1	0	0	0	0	0

*



If clr = 1, then FF clears.

If $\overline{clr}$ = 0 then FF clears.

If $\overline{clr}$ = 0 then FF clears.

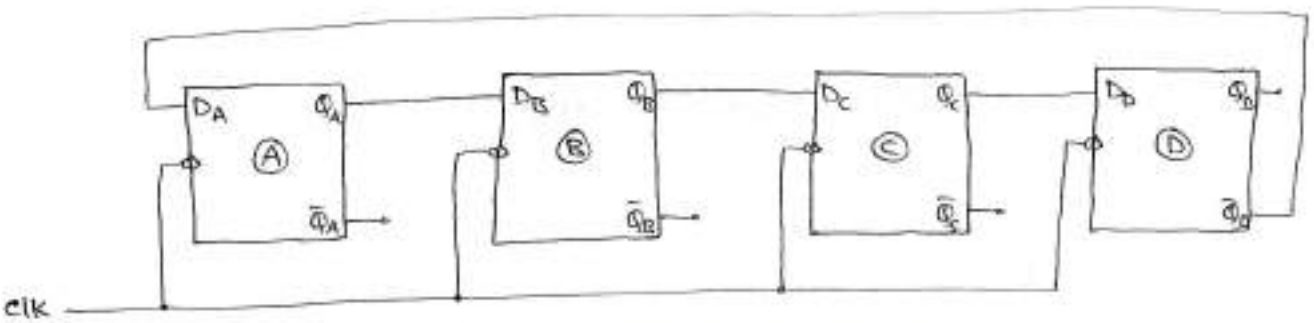
* Johnson counter (or) Twisted ring counter (or) Switch tile ring counter :-

(i) the 'n' bit ring counter circulates a single bit among the FF's to provide 'n' different states.

(ii) the no. of states can be doubled if the shift register is connected as a switch tile counter.

(iii) Johnson counters have basic counting cycles of length 2n. where 'n' is the no. of FF's.

(iv) In Johnson counter, the compliment of the o/p of the last FF is connected back to the i/p of the first FF.



counter o/p → Q_D Q_C Q_B Q_A
MSB LSB

Fig:- 4-bit Johnson ring counter.

Truth table on state sequence :-

clk	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

UNIT - 5

Memory and programmable Logic

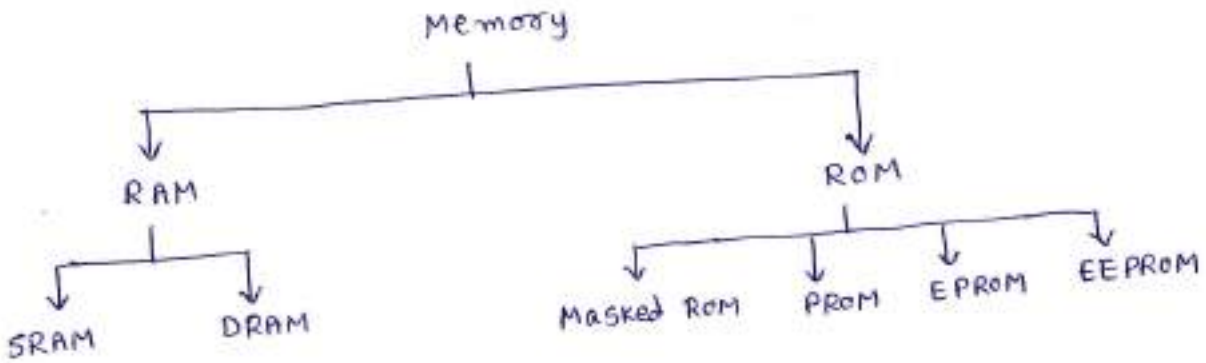
Memory

* Memory is a set of registers which holds instructions and data for processing.

* Memory unit stores instructions and data in Binary form

* There are 2 types of Memories that are used in digital systems. They are RAM, ROM

classification of Memories



Random Access Memory (RAM)

* RAM is called "Random Access Memory" because any storage location can be accessed directly

* the process of storing new information into Memory is referred to as a Memory write operation

* the process of transferring the stored information out of Memory is referred to as a Memory Read operation

* RAM performs both Read & write operations that's why it is called as Read/write Memory

* RAM is volatile Memory i.e if Power is OFF, the stored information will be lost. that's why we store only temporary data in RAM. So, RAM is called as data Memory

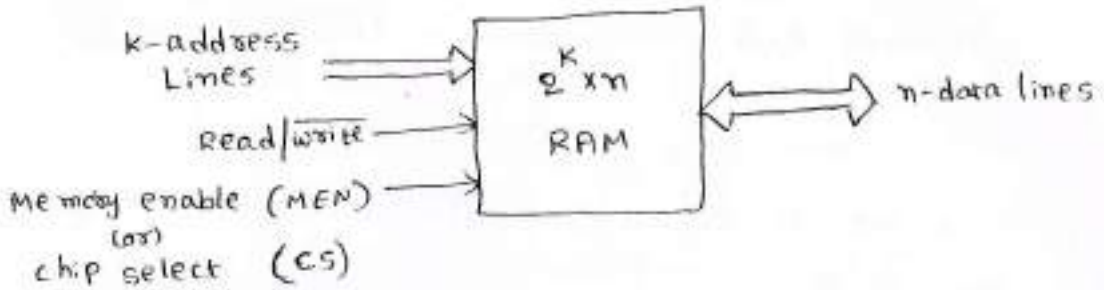


fig: B.D of $2^K \times n$ RAM

* In the above fig, size of RAM is $2^K \times n$, it means RAM consists of 2^K memory locations and each Memory location size is n-bits
* the communication between a Memory and other devices can be achieved through data lines. each data line carries one bit of binary information.
* Data lines are Bidirectional, but at any time they acts as either i/p or o/p lines. During Memory write operation, data lines acts as i/p lines.
* address lines carries desired Memory location address for memory read (or) write operation.
* Read/$\overline{\text{write}}$ is control signal. It is used to select either Read or write operation. if Read/$\overline{\text{write}}$ = 0, the RAM performs write operation
* RAM enables and performs either Read (or) write operation only when its CS (or) MEN i/p is high. if CS=0, RAM disables.

Static RAM (SRAM)

SRAM consists of flip-flops to store the binary information.

Dynamic RAM (DRAM)

* DRAM consists of CMOS transistors and capacitors
* DRAM stores the Binary information in the form of electric charges on capacitors

Read operation

The process of transferring the stored information out of memory is referred to as a memory read operation

Memory Read operation steps

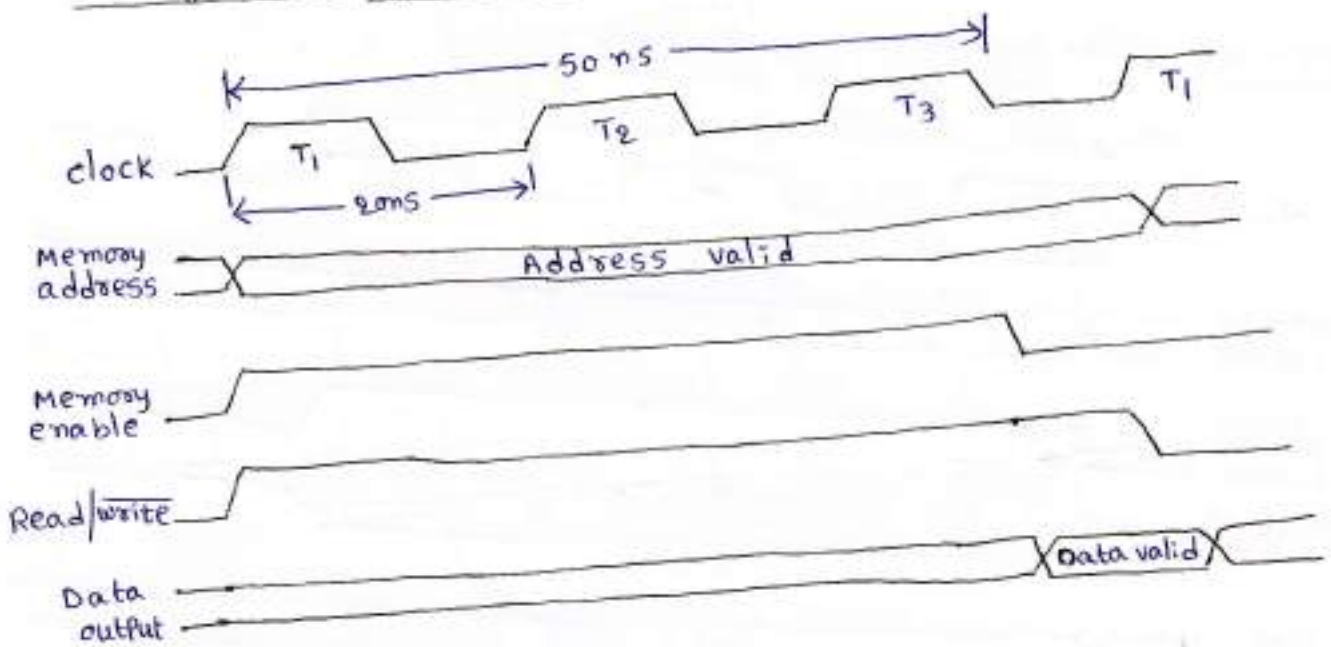
Step 1 : Apply the binary address of the desired word to the address lines

Step 2 : Activate the read input

The memory unit will then take the bits from the word that has been selected by the address and apply them to the output data lines.

The content of the selected word does not change after reading

Memory Read cycle Timing diagram



clock Time period = 20ns

Access time = 50ns

Access time

It is a memory device operating speed. It is defined as the time required to perform the read operation. (or) write

Write operation

The process of storing new information into memory is reffered to as a memory write operation

Memory write operation steps

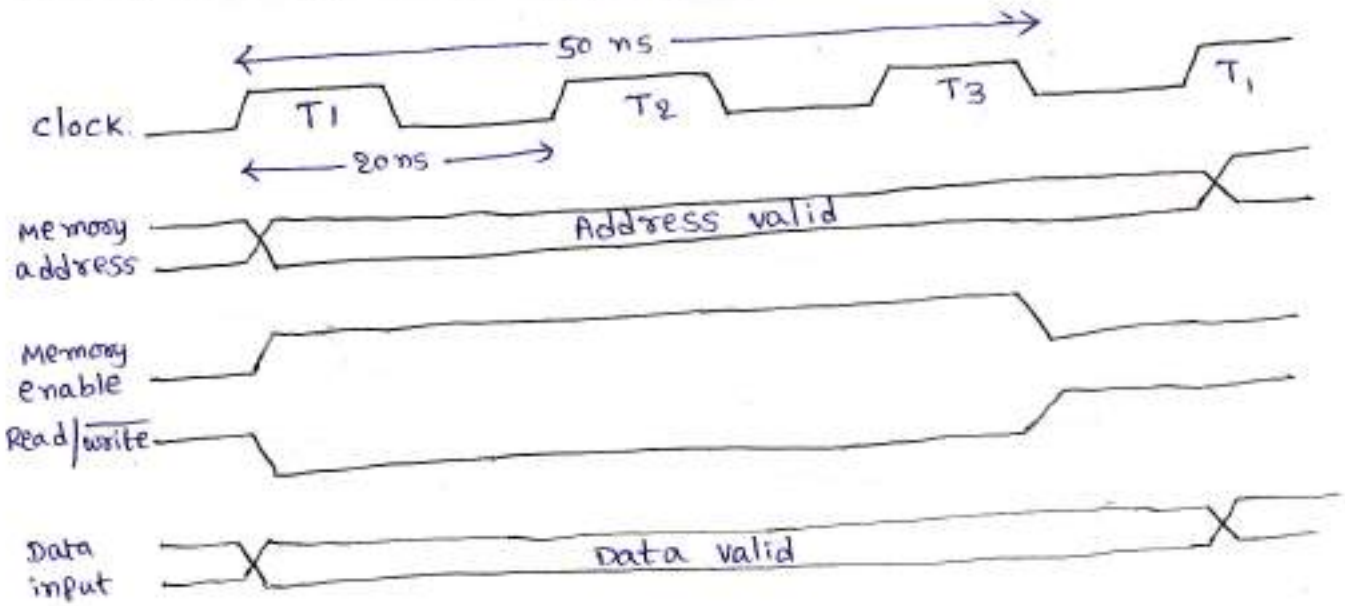
Step 1 : Apply the binary address of the desired word to the address lines

Step 2 : Apply the data bits that must be stored in memory to the data input lines

Step 3 : Activate the write input.

The memory unit will then take the bits from the input data lines and store them in the word specified by the address lines.

Memory write cycle Timing diagram



Read only Memory (ROM)

* ROM Performs only Read operation
* ROM is non-volatile memory i.e stored information will not get erased even after power is turned OFF. that's why we store permanent programs (eg: operating system programs) in ROM. ∴ ROM is called as program memory

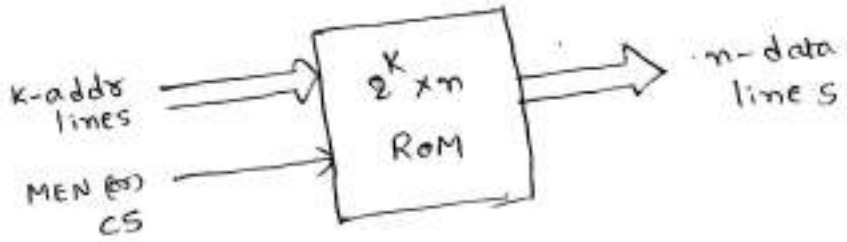


fig: B.D of $2^K \times n$ ROM

* for ROM data lines are always o/p lines because it performs only Read operation
* CS (or) MEN i/p is used to select ROM Ic. if CS = 0, then n-data lines will be in high impedance state.

Types of ROM

Masked ROM (or) Simple ROM

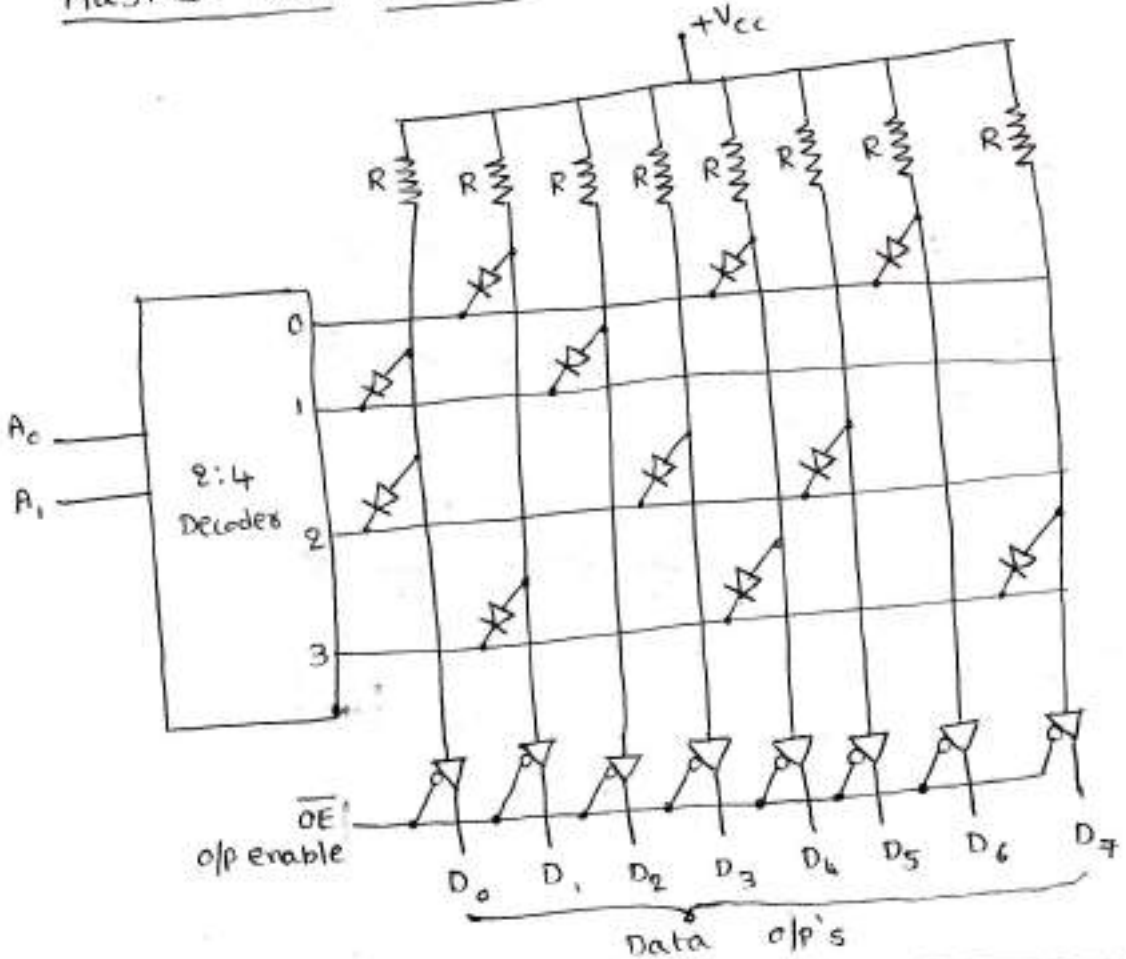


fig: simple 4 byte diode ROM

Address in Binary	Binary Data D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7
0 0	1 0 1 1 0 1 0 1
0 1	0 1 0 1 1 1 1 1
1 0	0 1 1 0 1 0 1 1
1 1	1 0 1 1 0 1 1 0

fig: contents of ROM

Diode ROM consists of only diodes and a decoder. as shown in the above fig. address lines A_0 and A_1 are decoded by 2:4 decoder and used to select one of the 4 rows. Data is available on the o/p data lines only when output enable ($\overline{OE}$) signal is Low.

Masked ROMs are used in microprocessor based toys, TV games, home computers.

PROM (Programmable Read only Memory (or) field PROM)

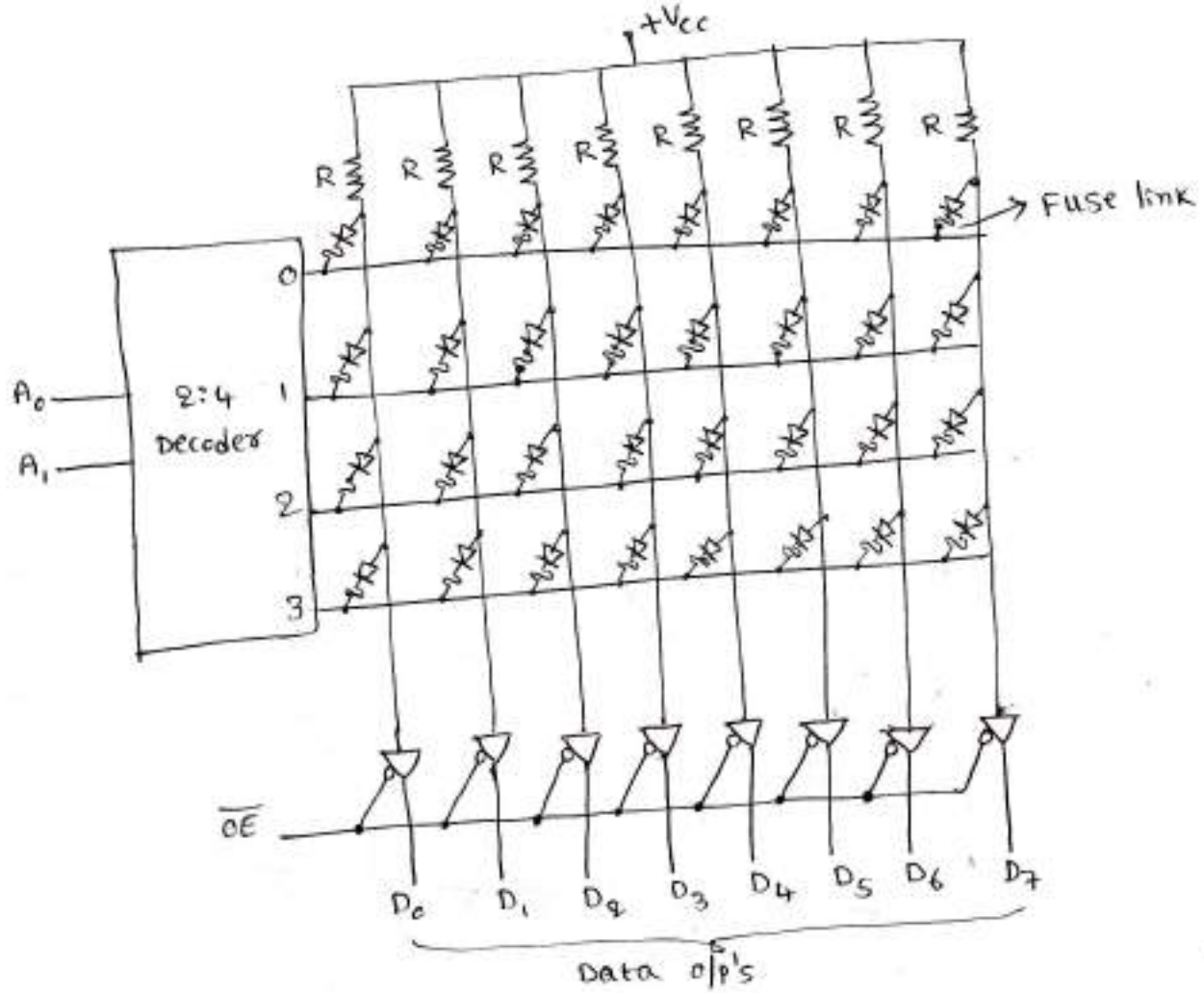


fig: 4-byte PROM

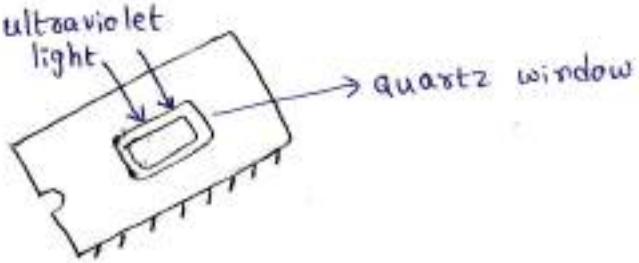
The PROM chip comes 'blank' and can be programmed only once by a user

above fig shows 4-byte PROM. It has diodes in every bit position; therefore, the o/p is initially all o's. each diode, however has a fusible link in series with it. By addressing bit and

applying proper current pulse at the corresponding o/p, we can blow out the fuse, storing logic 1 at that bit position.

The PROMs are one time programmable. once programmed, the information stored is permanent.

EPROM (Erasable programmable Read only Memory)



* EPROMs use MOS circuitry. they store 1's and 0's as a packet of charge in a buried layer of the Ic chip.
* EPROM chips can be erased and rewritten a no. of times.
* Stored data in the EPROM can be erased by exposing the chip to ultraviolet light through its quartz window for 15 to 20 minutes. (before erasing the data we have to remove the Ic from its socket)
* It is not possible to erase selective information, when erased the entire information is lost. the chip can be reprogrammed
* EPROM is ideally suitable for product development, experimental projects and college Laboratories, since this chip can be reused many times.
* In EPROM, it is possible to program any location at any time- either individually, sequentially, or at random.

EEPROM (Electrically Erasable programmable ROM)

* EEPROM is like the EPROM except that the previously programmed connections can be erased with an electrical signal instead of ultraviolet light
* the advantage of EEPROM over EPROM is that the device can be erased without removing it from its socket.

30/10/2014

* Logic devices :- Logic devices can be broadly categoried as (i) Fixed logic devices
(ii) Programmable logic devices

1. Fixed logic devices :- (FLD's)

As the name indicates, the circuits in the FLD are permanent., they perform one function or set of functions. ones manufactured, they cannot be erased.

2. programmable logic devices (PLD's) :-

* PLD is an IC that contains large no. of gates, FF's and registers that are inter connected on the chip.

* PLDs can be reconfigured to perform any no. of functions at any time.

* Types of PLD's are (i) simple PLD's / sequential PLD's.
(a) PROM
(b) PLA
(c) PAL
(ii) complex PLD's (CPLD's)
(iii) Field programmable gate Arrays (FPGA).

(i) PROM (programmable Read only memory) :-

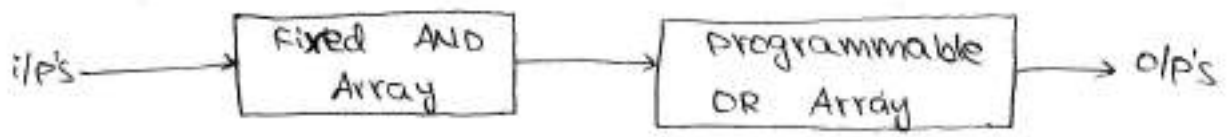
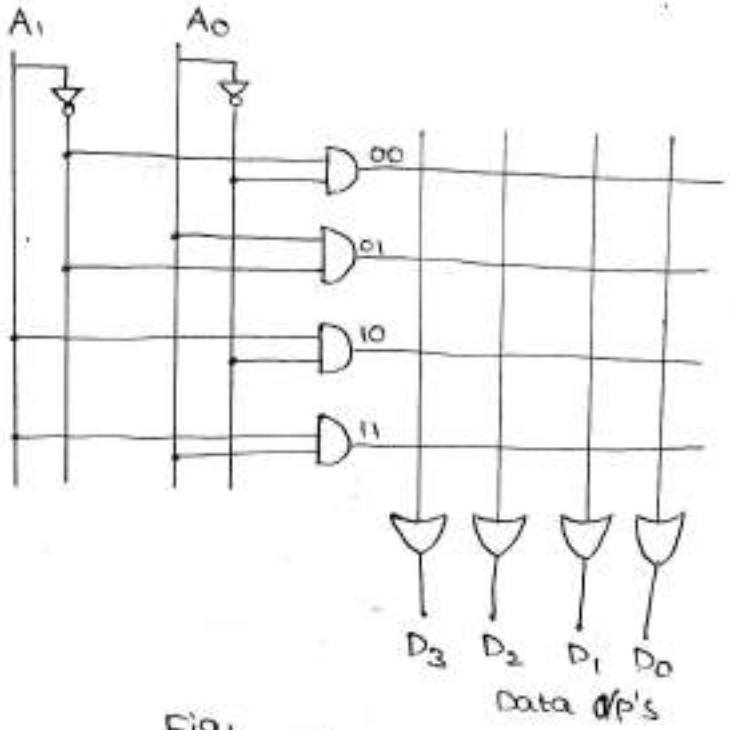


Fig:- Basic configuration of PROM



PROM has fixed AND Array & programmable OR Array

Fig:- Internal logic of 4X4 ROM (00) AND-OR implementation of 4X4 ROM

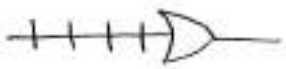


Fig:- Array logic Symbol for OR gate

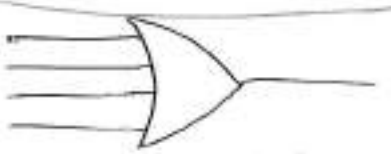


Fig:- conventional logic Symbol for OR gate.

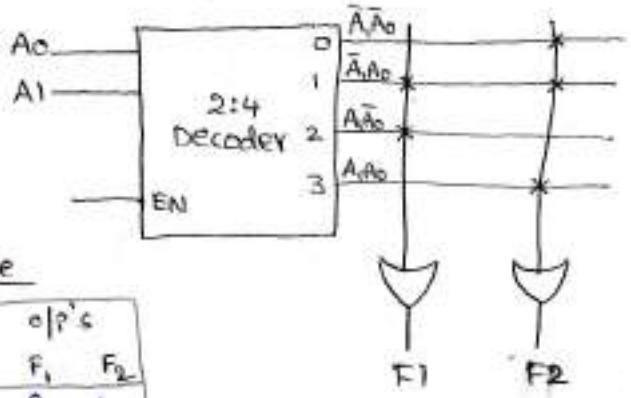
Q.:- Implement the following boolean functions using PROM.

$F1(A_1, A_0) = \Sigma m(1,2)$; $F2(A1, A0) = \Sigma m(0,1,3)$.

Sol:- Logic diagram

I/P variables are A_1, A_0 (2 variables)

∴ use 2:4 Decoder.



$F1(A_1, A_0) = \Sigma m(1,2)$

$= \bar{A}_1 A_0 + A_1 \bar{A}_0$

$F2(A_1, A_0) = \Sigma m(0,1,3)$

$= \bar{A}_1 \bar{A}_0 + \bar{A}_1 A_0 + A_1 A_0$

Truth table

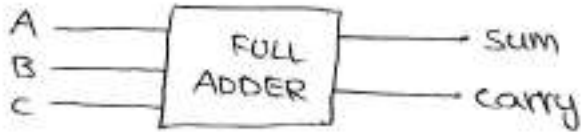
i/p's		o/p's	
A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	1
1	0	1	0
1	1	0	1

fig: Implementation of F1, F2 using PROM

Q.:- Implement Full adder using PROM.

Sol: Full adder :-

Block diagram :



Truth table :

i/p's			o/p's	
A	B	C	sum	carry
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

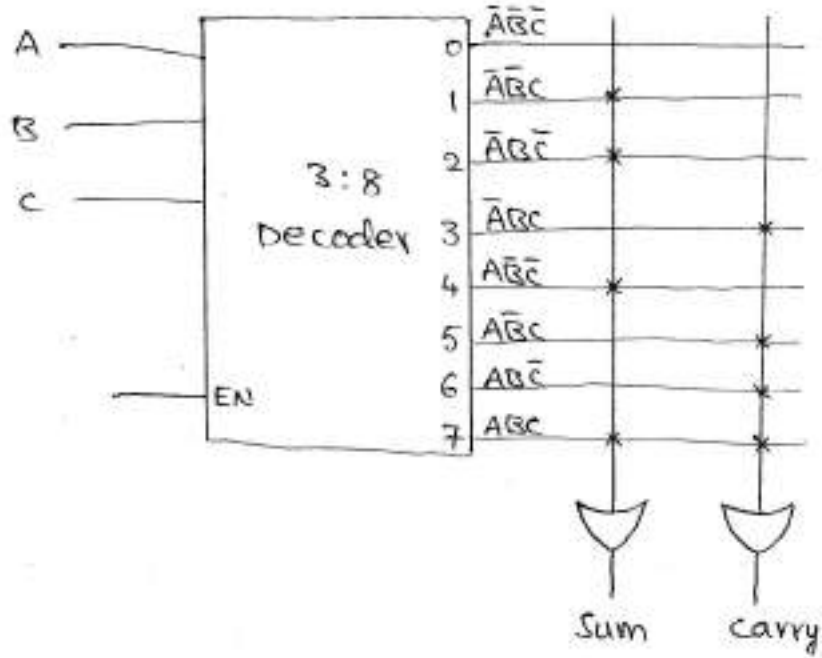
$\therefore$ Sum $= \Sigma m(1,2,4,7)$

$\therefore$ carry $= \Sigma m(3,5,6,7)$

Logic Diagram

$\therefore$ I/p variables = A, B, C (3 variables)

$\therefore$ use 3:8 decoder.



$\therefore$ Sum $= \Sigma m(1,2,4,7)$

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

$\therefore$ carry $= \Sigma m(3,5,6,7)$

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

fig: Full adder using PROM

Q.:- Design a combinational circuit using ROM. The circuit accepts 3-bit binary no. and Generates its equivalent EXCESS-3 code.

Sol:

EXCESS-3 :-

Truth table :-

I/P (3 bit binary no.)			O/P (EXCESS-3 code of I/P)			
B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	0	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

$\therefore$ $E_0(B_2, B_1, B_0) = \Sigma m(0, 2, 4, 6)$

$E_1(B_2, B_1, B_0) = \Sigma m(0, 3, 4, 7)$

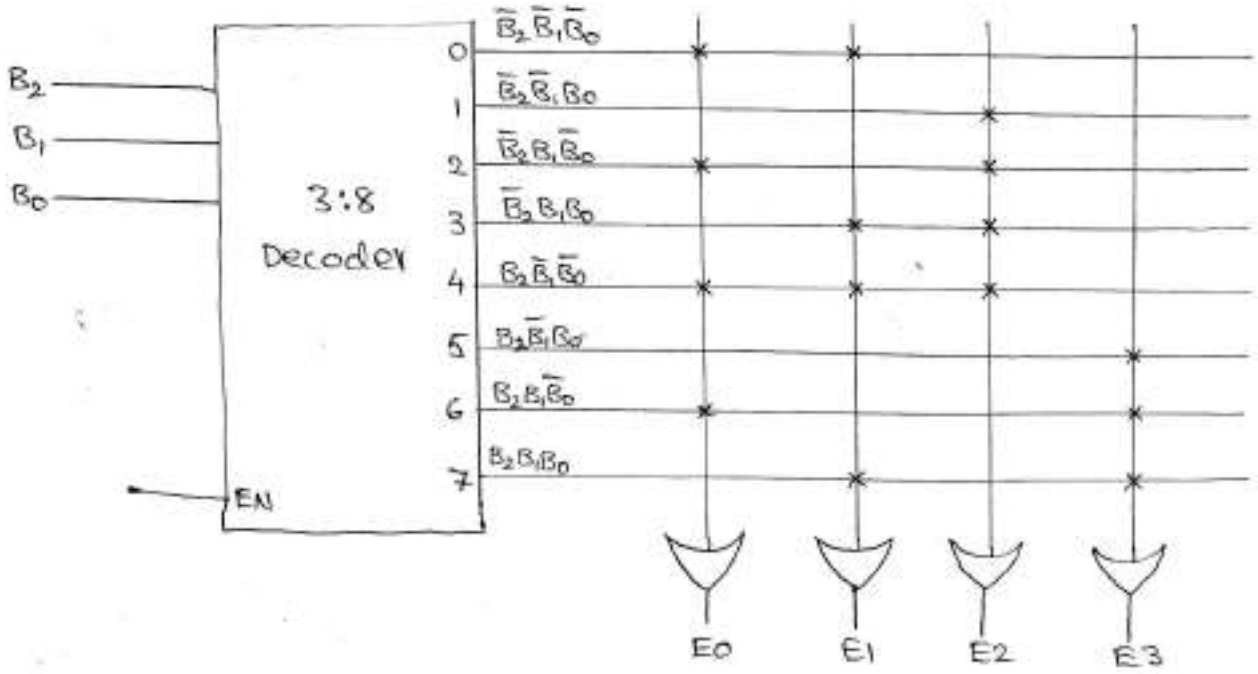
$E_2(B_2, B_1, B_0) = \Sigma m(1, 2, 3, 4)$

$E_3(B_2, B_1, B_0) = \Sigma m(5, 6, 7)$

Logic Diagram

$\therefore$ I/P variables $= B_2, B_1, B_0$ (3 variables)

$\therefore$ use 3:8 decoder.



Q.:- Design a combinational circuit using ROM. The circuit accepts a 3-bit number and generates An o/p binary no. equal to the square of the i/p no.

Sol:- Truth table

i/p (3-bit binary no.)			o/p (square of i/p binary no)						Decimal no.
B_2	B_1	B_0	S_5	S_4	S_3	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

$\therefore S_0(B_2, B_1, B_0) = \Sigma m(1, 3, 5, 7)$

$S_1(B_2, B_1, B_0) = \Sigma m($

$S_2(B_2, B_1, B_0) = \Sigma m(2, 6)$

$S_3(B_2, B_1, B_0) = \Sigma m(3, 5)$

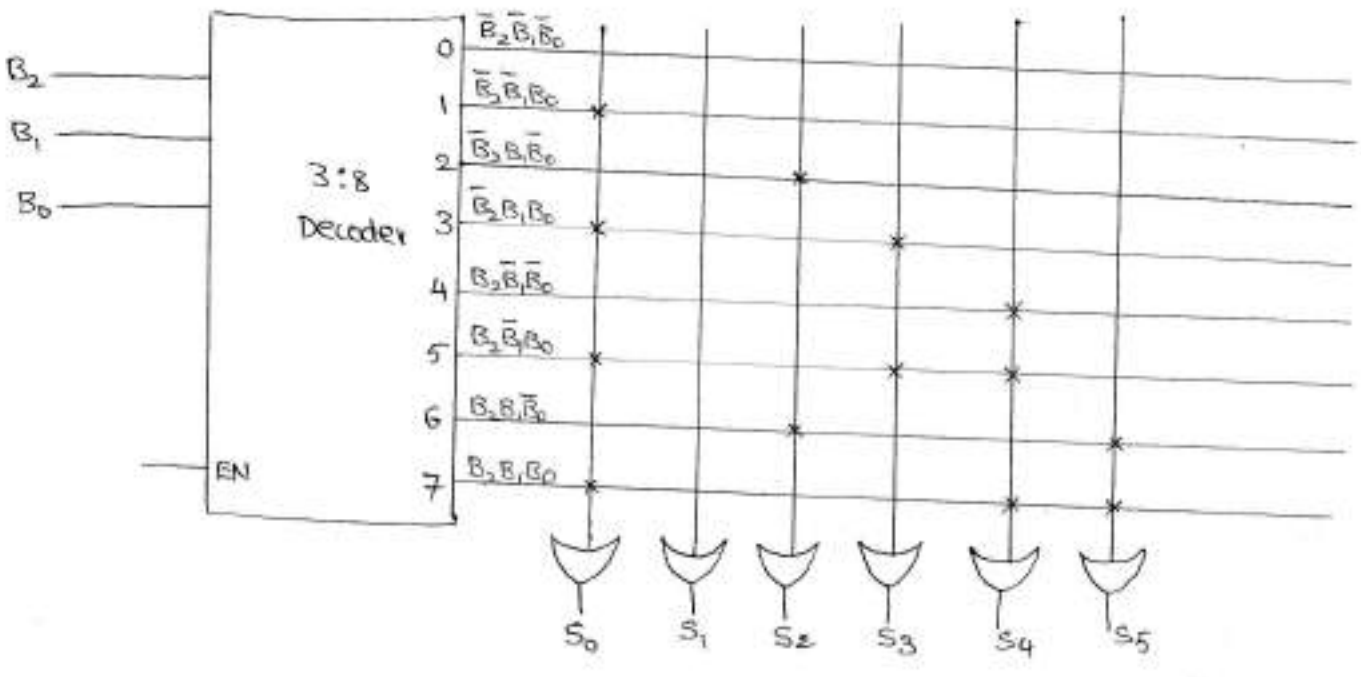
$S_4(B_2, B_1, B_0) = \Sigma m(4, 5, 7)$

$S_5(B_2, B_1, B_0) = \Sigma m(6, 7)$

Logic Diagram

$\therefore$ I/P variables = B_2, B_1, B_0 (3 variables).

$\therefore$ Use 3:8 decoder.



05/11/2014

Programmable logic Array (PLA):-

Fig: Basic configuration of Programmable logic array.

Q.:- A combinational circuit is defined by the function $F_1 = \Sigma m(3,5,7)$, $F_2 = \Sigma m(4,5,7)$. Implement the ckt with a PLA having 3 i/p's, 3 product terms and two o/p's.

Sol:- Given, $F_1 = \Sigma m(3,5,7)$

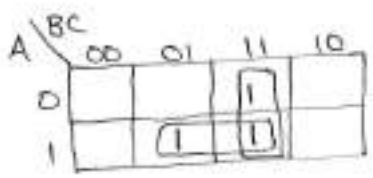
$F_2 = \Sigma m(4,5,7)$

Truth table for boolean functions F_1 & F_2.

i/p's			o/p's	
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

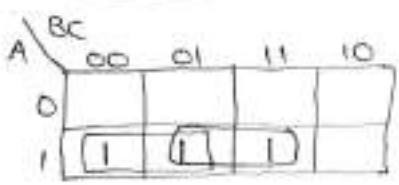
K-Map simplification :-

for F_1



$\therefore F_1 = AC + BC$

for F_2



$\therefore F_2 = A\bar{B} + AC$

PLA program table :-

	Product term	i/p's A	B	C	o/p's F_1	F_2
1	AC	1	–	1	1	1
2	BC	–	1	1	1	–
3	$A\bar{B}$	1	0	–	–	1

no. of i/p's → 3
no. of product terms → 3
no. of o/p's → 2

Logic diagram :-

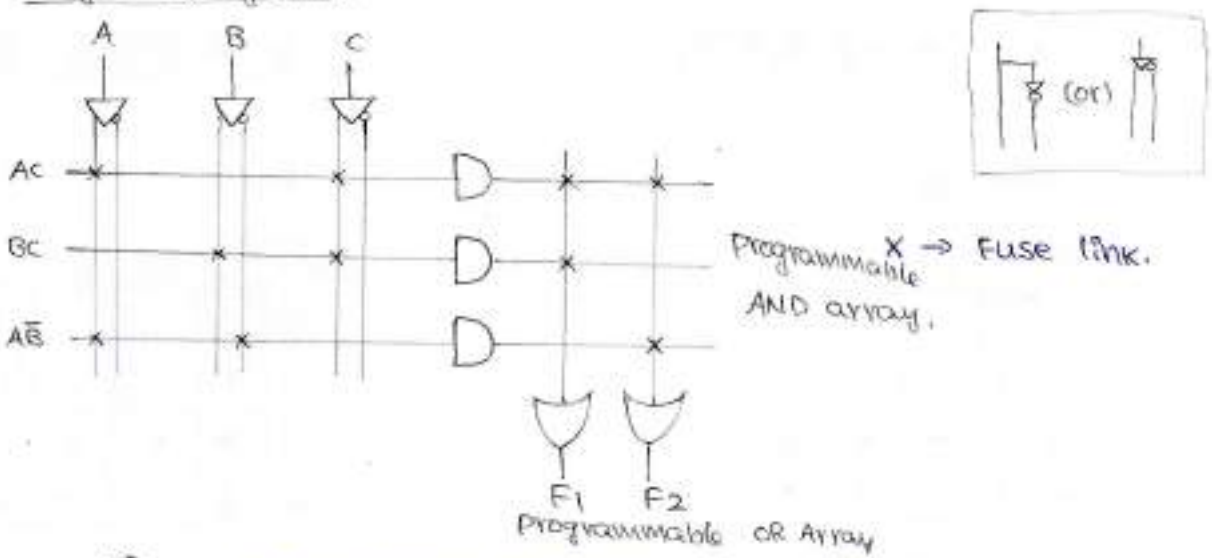


Fig :- Implementation of F_1, F_2 using PLA

Q.:- Design a BCD to EXCESS-3 code converter and implement using suitable PLA.

Sol:- Truth table for BCD to EXCESS-3 code converter :-

	i/p's B_3	B_2	B_1	B_0	o/p's E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

(∵ BCD digits are 0-9. So consider 10-15 as don't care)

$E_3(B_3, B_2, B_1, B_0) = \Sigma m(5,6,7,8,9)$

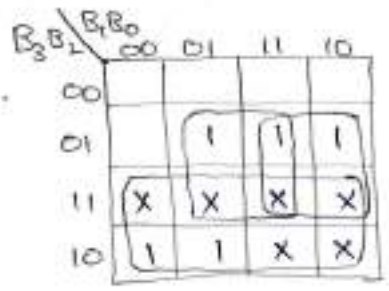
$E_2(B_3, B_2, B_1, B_0) = \Sigma m(1,2,3,4,9)$

$E_1(B_3, B_2, B_1, B_0) = \Sigma m(0,3,4,7,8)$

$E_0(B_3, B_2, B_1, B_0) = \Sigma m(0,2,4,6,8)$

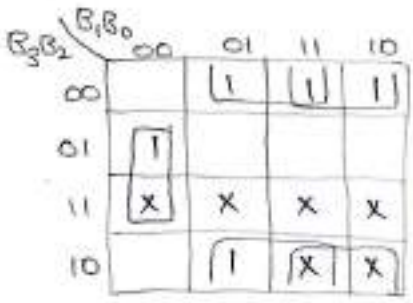
K-map simplification :-

for E_3



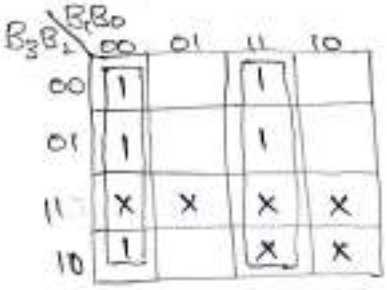
$\therefore E_3 = B_3 + B_1B_0 + B_2B_1$

For E_2



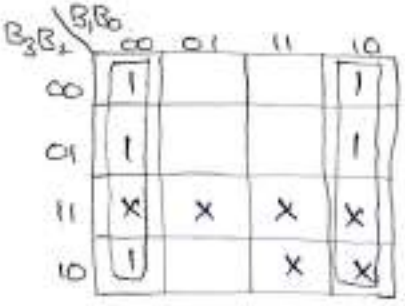
$\therefore E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2B_0 + \bar{B}_2B_1$

for E_1



$\therefore E_1 = \bar{B}_1\bar{B}_0 + B_1B_0$

For E_0

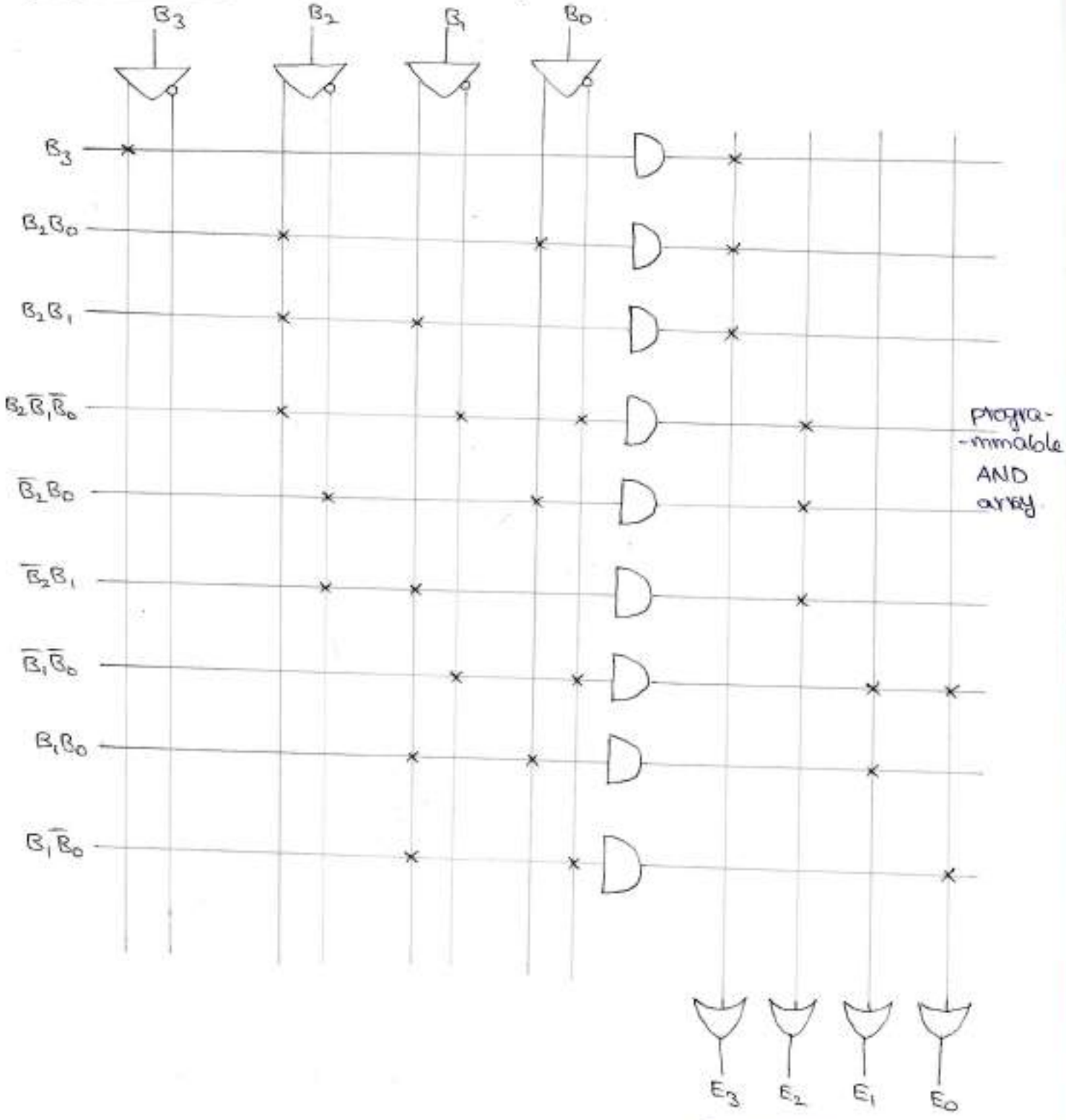


$\therefore E_0 = \bar{B}_1\bar{B}_0 + B_1\bar{B}_0$

PLA program table :-

Product terms	i/p's B_3	B_2	B_1	B_0	o/p's E_3	E_2	E_1	E_0
B_3	1	–	–	–	1	–	–	–
B_2B_0	–	1	–	1	1	–	–	–
B_2B_1	–	1	1	–	1	–	–	–
$B_2\bar{B}_1\bar{B}_0$	–	1	0	0	–	1	–	–
$\bar{B}_2B_0$	–	0	–	1	–	1	–	–
$\bar{B}_2B_1$	–	0	1	–	–	1	–	–
$\bar{B}_1\bar{B}_0$	–	–	0	0	–	–	1	1
B_1B_0	–	–	1	1	–	–	1	–
$B_1\bar{B}_0$	–	–	1	0	–	–	–	1

logic diagram :-



* Programmable Array Logic (PAL) :-

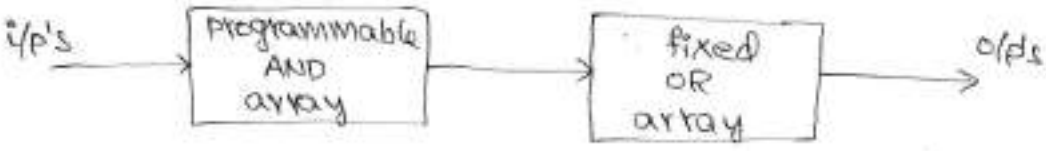


fig:- Basic configuration of PAL.

Q:- Implement the following boolean expressions using PAL.

$F_1 = \sum m(3,5,7)$, $F_2 = \sum m(4,5,7)$.

Sol:- Given $F_1 = \sum m(3,5,7)$

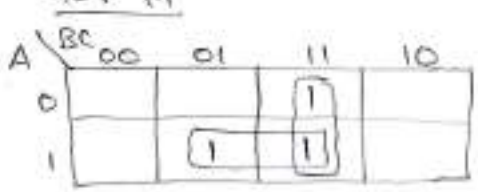
$F_2 = \sum m(4,5,7)$

Truth table for boolean functions F_1 & F_2

i/p's			o/p's	
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

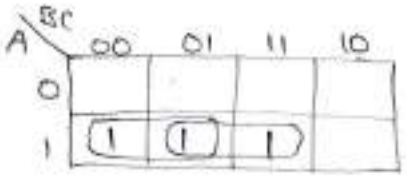
K-map simplification:-

for F_1



$\therefore F_1 = AC + BC$

for F_2



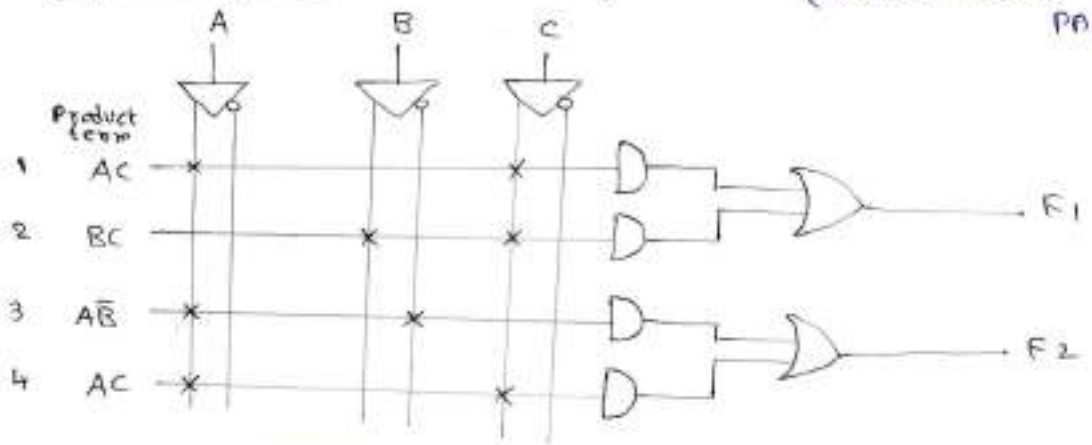
$\therefore F_2 = A\bar{B} + AC$

PAL program table :-

	Product term	AND i/p's A	B	C	o/p's
1	AC	1	-	1	$F_1 = AC + BC$
2	BC	-	1	1	
3	$A\bar{B}$	1	0	-	$F_2 = AC + A\bar{B}$
4	AC	1	-	1	

Logic diagram :-

(this is 2-wide AND-OR structure PAL)



Q :- Implement the following Boolean expressions using PAL

$$A(x,y,z) = \Sigma m(1,2,4,6)$$
$$B(x,y,z) = \Sigma m(0,1,6,7)$$
$$C(x,y,z) = \Sigma m(2,6)$$
$$D(x,y,z) = \Sigma m(1,2,3,5,7)$$

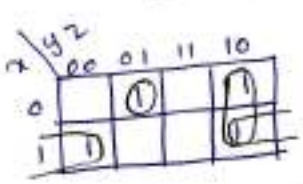
Solu

Truth table for Boolean functions A, B, C, D

i/p's			o/p's			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	0	0	1
1	0	0	1	0	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	0	1

K-map simplification

for A

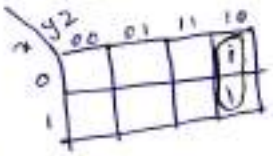


$A = x\bar{z} + \bar{x}\bar{y}z + y\bar{z}$

for B

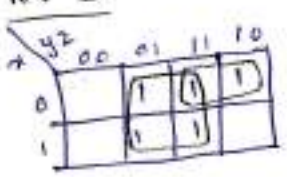
$B = \bar{x}\bar{y} + xy$

for C



$C = y\bar{z}$

for D



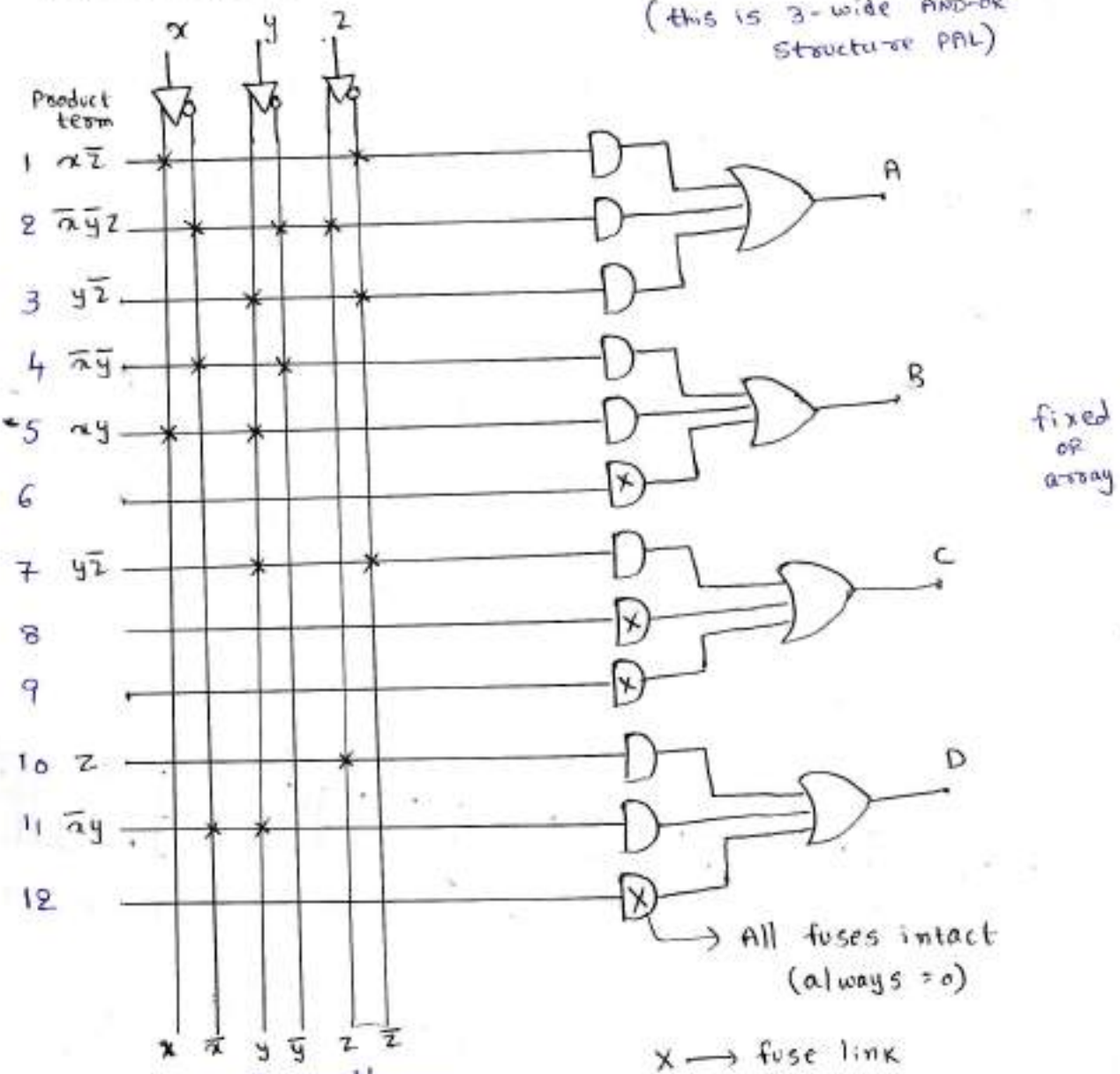
$D = z + \bar{x}y$

PAL Program Table

	Product terms	AND inputs x	y	z	outputs
1	$x\bar{z}$	1	–	0	$A = x\bar{z} + \bar{x}\bar{y}z + y\bar{z}$
2	$\bar{x}\bar{y}z$	0	0	1	
3	$y\bar{z}$	–	1	0	
4	$\bar{x}\bar{y}$	0	0	–	$B = \bar{x}\bar{y} + xy$
5	xy	1	1	–	
6					
7	$y\bar{z}$	–	1	0	$C = y\bar{z}$
8					
9					
10	z	–	–	1	$D = z + \bar{x}y$
11	$\bar{x}y$	0	1	–	
12					

Logic diagram

(this is 3-wide AND-OR structure PAL)



Q Implement the following Boolean functions using 3 wide AND-OR structure PAL

$w(A,B,C,D) = \sum m(0,2,6,7,8,9,12,13)$

$x(A,B,C,D) = \sum m(0,2,6,7,8,9,12,13,14)$

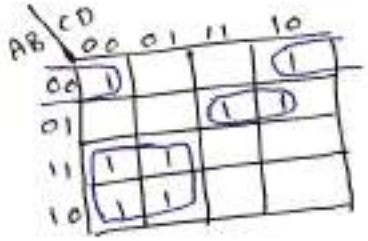
$y(A,B,C,D) = \sum m(2,3,8,9,10,12,13)$

$z(A,B,C,D) = \sum m(1,3,4,6,9,12,14)$

Solu

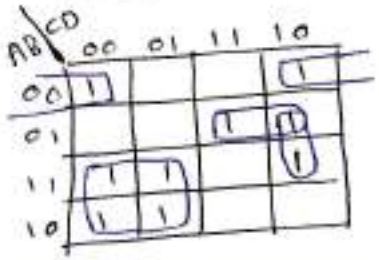
K-map simplification

for w



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

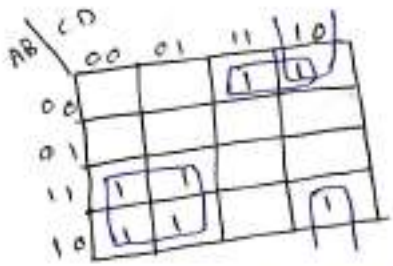
for x



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

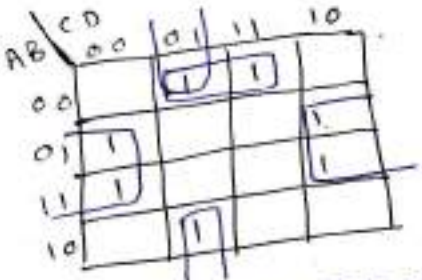
$= w + BC\bar{D}$

for y



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

for z



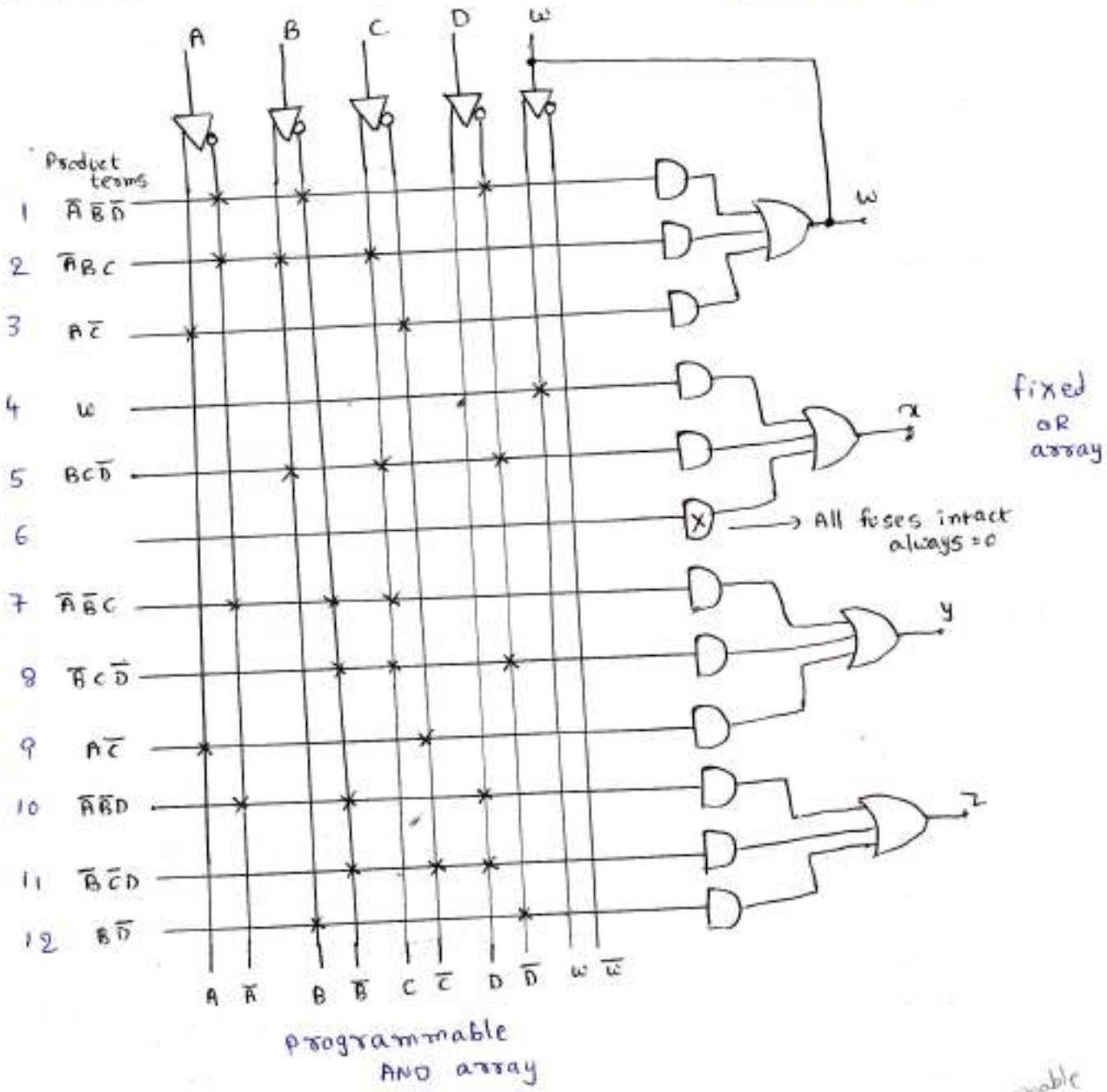
$z = \bar{A}\bar{B}D + \bar{B}\bar{C}D + B\bar{D}$

PAL program Table

	Product terms	AND inputs A	B	C	D	w	outputs
1	$\bar{A}\bar{B}\bar{D}$	0	0	–	0	–	$w = \bar{A}\bar{B}\bar{D} + \bar{A}BC + A\bar{C}$
2	$\bar{A}BC$	0	1	1	–	–	
3	$A\bar{C}$	1	–	0	–	–	
4	w	–	–	–	–	1	$x = w + BC\bar{D}$
5	$BC\bar{D}$	–	1	1	0	–	
6							
7	$\bar{A}\bar{B}C$	0	0	1	–	–	$y = \bar{A}\bar{B}C + \bar{B}C\bar{D} + A\bar{C}$
8	$\bar{B}C\bar{D}$	–	0	1	0	–	
9	$A\bar{C}$	1	–	0	–	–	
10	$\bar{A}\bar{B}D$	0	0	–	1	–	$z = \bar{A}\bar{B}D + \bar{B}\bar{C}D + B\bar{D}$
11	$\bar{B}\bar{C}D$	–	0	0	1	–	
	$B\bar{D}$	–	1	–	0	–	

Logic diagram

This is 3-wide AND-OR structured PAL



Comparison Between PROM, PLA and PAL

(PROM: Programmable OR, fixed AND; PAL: Programmable AND, fixed OR)

	PROM	PLA	PAL
1	AND array is fixed and OR array is programmable	Both AND and OR arrays are programmable	OR array is fixed and AND array is programmable
2	cheaper and simple to use	costliest and complex than PAL and PROMs	cheaper and simpler
3	All minterms are decoded	AND array can be programmed to get desired minterms	AND array can be programmed to get desired minterms
4	only Boolean functions in standard SOP form can be implemented using PROM	Any Boolean functions in SOP form can be implemented using PLA	Any Boolean function in SOP form can be implemented using PAL

Error Detecting and correcting codes

when the digital information in the binary form is transmitted from one ckt to another ckt an error may occur. this means signal corresponding to 0 may change to 1 (or) vice versa due to presence of noise.

To maintain the data integrity between Transmitter and Receiver, extra bit or more than one bit are added in the data. These extra bits allow the detection and some times correction of error in the data. The data along with the extra bit/bits forms the code.

codes which allow only error detection are called error detecting codes.

codes which allow error detection and correction are called error detecting and corresponding codes.

example of error detecting code is parity code. and error detecting and correction code is Hamming code.

Hamming code

Hamming code not only provides the detection of a bit error, but also identifies which bit is in error so that it can be corrected. thus Hamming code is called error detecting and correcting code.

Hamming code

Q: encode the binary word 11000100 into even parity Hamming code

Sol^{n}: H·C comprises of certain parity bits. The no. of parity bits used by the H·C depends on the no. of data bits. Thus, we can derive the no. of parity bits used by the H·C for the given no. of data bits from the eqn given below

$$2^p \geq n + p + 1$$

where $p \rightarrow$ no. of parity bits $\quad n \rightarrow$ no. of data bits

given binary word = 11000100

$n = 8$, $\therefore p = 4$

$\therefore$ total no. of bits in H·C = 8 + 4 = 12

Place 4 parity bits in positions $2^0, 2^1, 2^2, 2^3$ respectively $(1, 2, 4, 8)$

So, the H·C is of the form

bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_0	P_1	1	P_2	1	0	0	P_3	0	1	0	0

To determine parity bit P_0, consider 1, 3, 5, 7, 9, 11 bits

for $P_0 \rightarrow$ 1 3 5 7 9 11 $\longrightarrow P_0$ 1 1 0 0 0

we are using even parity. $\therefore$ To make no. of is in the above group even P_0 must be '0'.

$\therefore P_0 = 0$

for $P_1 \rightarrow$ 2 3 6 7 10 11 $\longrightarrow P_1$ 1 0 0 1 0

$\therefore P_1 = 0$

for $P_2 \rightarrow$ 4 5 6 7 12 $\rightarrow P_2$ 1 0 0 0

$\therefore P_2 = 1$

for $P_3 \rightarrow$ 8 9 10 11 12 $\rightarrow P_3$ 0 1 0 0

$\therefore P_3 = 1$

	P_3	P_2	P_1	P_0
0	0	0	0	0
1	0	0	0	1 ✓ (for P_0 consider)
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 ✓

$\therefore$ 12-bit even parity H.C for 8-bit data word 11 000 100 is $\rightarrow$

00 111 0 0 10 1 00

Single error Detection

Q Detect and correct error in the Hamming code received as 000 11 0010100 with even parity

Consider 12-bit H.C shown above, but 3rd bit error. We have to detect this

Solu

Received even parity H.C is 000110010100

bit position	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	1	1	0	0	1	0	1	0	0

To detect error bit, find error word $C = C_3 C_2 C_1 C_0$

if $C = 0000$, that indicates no error has occurred.
if $C \neq 0$, then the 4-bit binary no. gives the position of the erroneous bit

Total no. of bits in error word depends on formula

$2^C \geq n+P$ where $n+P$ = no. of bits in H.C
C = no. of bits in error word

$n+P = 12$, $\therefore C = 4$ i.e error word has 4-bits

$C = C_3 C_2 C_1 C_0$

To determine C_0, consider 1, 3, 5, 7, 9, 11 bits

$C_0 \rightarrow$ 1 3 5 7 9 11 $\rightarrow$ 0 0 1 0 0 0

we are using even parity H.C
but, Total no. of ones in above group are odd. ∴ there is an error bit in the above group.

$$\therefore C_0 = 1$$

for $C_1 \rightarrow$ 2 3 6 7 10 11 $\rightarrow$ 0 0 0 0 1 0

there is no even no. of 1's
∴ parity checks for even parity is wrong.

$$\therefore C_1 = 1$$

for $C_2 \rightarrow$ 4 5 6 7 12 $\rightarrow$ 1 1 0 0 0

there is even no. of 1's. correct parity

$$\therefore C_2 = 0$$

for $C_3 \rightarrow$ 8 9 10 11 12 $\rightarrow$ 1 0 1 0 0

correct parity

$$\therefore C_3 = 0$$

Now, $C = C_3\ C_2\ C_1\ C_0 = 0\ 0\ 1\ 1$

∴ 3rd bit in the received message is wrong.
This bit then should be complemented, so that the correct Code will be 0 0 1 1 1 0 0 1 0 1 0 0

Note: for even parity, we use XOR gates for generating parity bits
for ODD " " XNOR " " "
i.e for $P_0 \rightarrow$ XOR of bits (1, 3, 5, 7, 9, 11)

Double-Error Detection

The Hamming code can detect and correct only a single error. Multiple errors are not detected.

By adding another parity bit to the coded word, the Hamming code can be used to correct a single error and detect double errors.

if we include this additional parity bit, then the previous 12-bit Hamming coded word becomes $0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,P_{13}$.

where P_{13} makes no. of 1's in the message even if we are using even parity H.C

$0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,P_{13}$

(for even parity) $P_{13} = 1$

∴ 13-bit H.C word = $0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,1$

Transmitter then sends this 13-bit word to Receiver.

after receiving this 13-bit word, Receiver calculates error word (C) and also calculates the parity (P) over the entire 13-bits.

after calculating C & P four cases can occur:

if $C=0$ and $P=0$, no error occured

if $C \neq 0$ and $P=1$, a single error occured that can be corrected

if $C \neq 0$ and $P=0$, a double error occured that is detected but that cannot be corrected.

if $C=0$ and $P=1$, An error occured in the P_{13} bit.