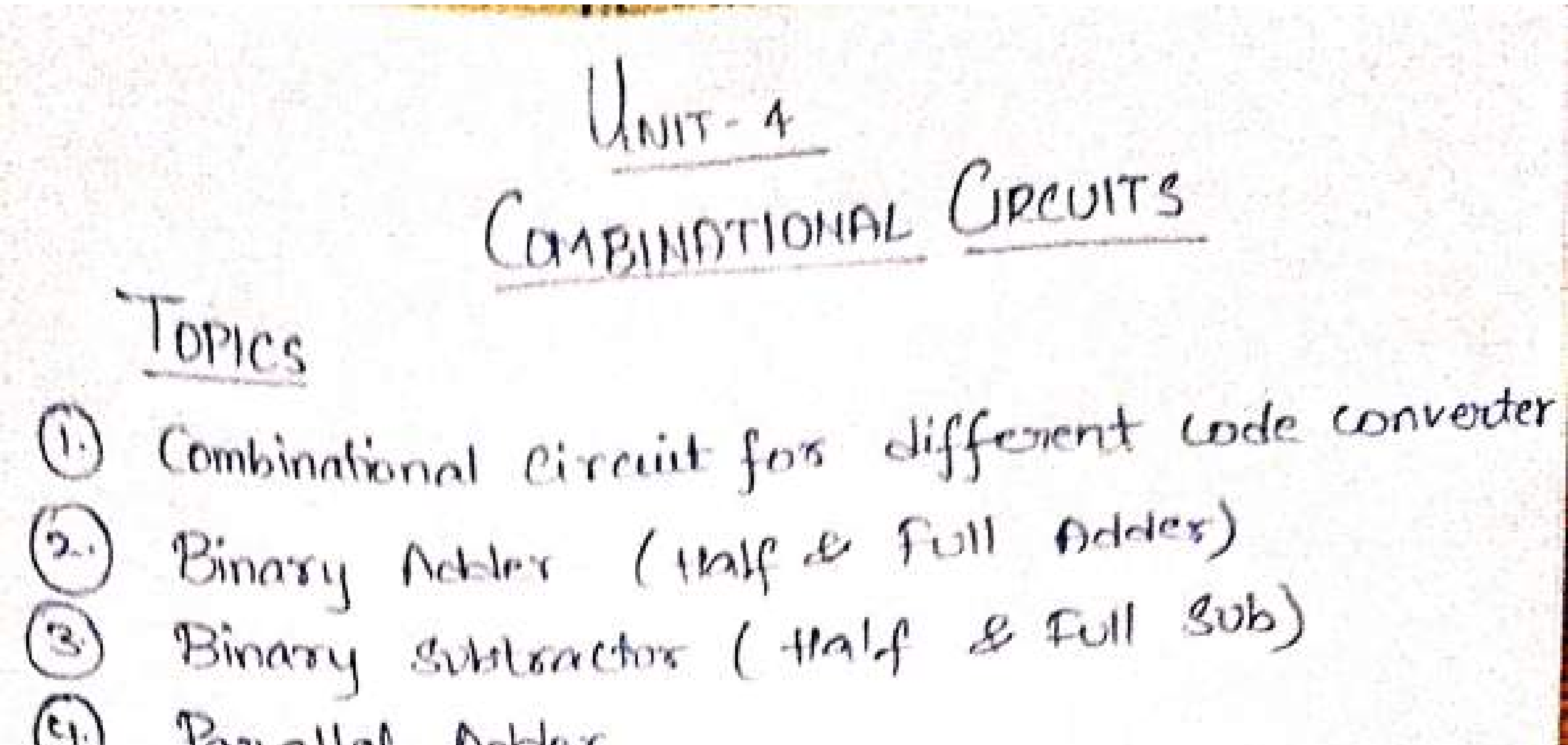


UNIT - 4

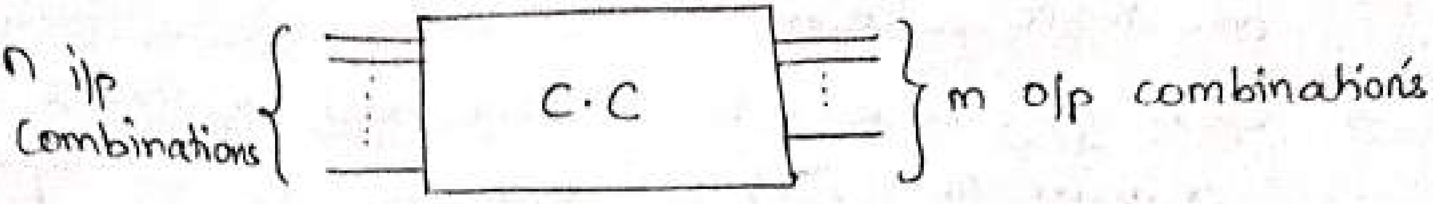
COMBINATIONAL CIRCUITS

TOPICS

1. Combinational circuit for different code converter
2. Binary Adder (Half & Full Adder)
3. Binary Subtractor (Half & Full Sub)
4. Parallel Adder
5. Decimal Adder
6. Binary Multiplier
7. Magnitude Comparator
8. Decoder,
9. Encoder
10. Multiplexer
11. Demultiplexer.

* Combinational Circuits

* A combinational circuit has 'n' input combination & 'm' output combinations



* Procedure to design a C·C:

(1.) Determine number of variables
(2.) Draw the truth table
(3.) Simplify expression in either SOP (or) POS form using K-map.
(4.) Draw logic circuit for minimised expression.

* Code Converters

* They are used for providing encryption to data.
* Few of the basic code converters are
  - (i) Binary to Grey Code (B2G)
  - (ii) Grey Code to Binary (G2B)
  - (iii) Excess-3 code ...

  ... etc..

NOTE: BCD (Binary Coded Decimal)

* Binary Codes for decimal numbers requires minimum of 4 bits.
* Most common BCD code is 8421 code in which each decimal digit is represented by a 4-bit binary number.

(i) BCD to Grey Code

Grey Code:

→ It is a special case of unit distance code.

→ Bit patterns for any 2 numbers differ in only 1-bit position.

→ Each grey code bit is obtained by applying exclusive OR (XOR) operation to the corresponding binary code bit & the next higher bit.

→ Expression for grey code.

$g_i = b_i \oplus b_{i+1}$ for $0 \leq i \leq n-1$

$g_{n-1} = b_{n-1}$ for MSB

→ Truth Table

	8	4	2	1				
decimal number	binary code				grey code			
	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

→ Grey code is a non-weighted, reflective code.

③ ④

→ From the truth table determine the min terms 1

→ $g_3 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$

$g_2 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$

$g_1 = \Sigma m(2, 3, 4, 5, 10, 11, 12, 13)$

$g_0 = \Sigma m(1, 2, 5, 6, 9, 10, 13, 14)$

→ K-map for above SOP expressions are obtained as below. -1

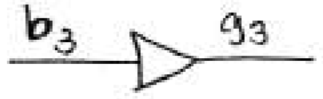
<u>g_3 k-map</u> is $m_8, m_9, m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15}$

b_3b_2 \ b_1b_0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12 1	13 1	15 1	14 1
10	8 1	9 1	11 1	10 1

→ b_3

Mathematical expression is $g_3 = b_3$

Circuit for g_3



<u>g_2 k-map</u> is $m_4, m_5, m_6, m_7, m_8, m_9, m_{10}, m_{11}$

b_3b_2 \ b_1b_0	00	01	11	10
00	0	1	3	2
01	4 1	5 1	7 1	6 1
11	12	13	15	14
10	8 1	9 1	11 1	10 1

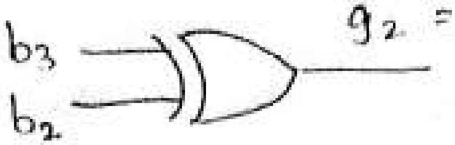
→ $\overline{b_3} b_2$

→ $b_3 \overline{b_2}$

Mathematical expression is $g_2 = \overline{b_3} b_2 + b_3 \overline{b_2}$

$g_2 = b_3 \oplus b_2$

circuit for g_2



$g_2 = b_3 \oplus b_2$

<u>g_1 k-map</u> $m_2, m_3, m_4, m_5, m_{10}, m_{11}, m_{12}, m_{13}$

b_3b_2 \ b_1b_0	00	01	11	10
00	0	1	3 1	2 1
01	4 1	5 1	7	6
11	12 1	13 1	15	14
10	8	9	11 1	10 1

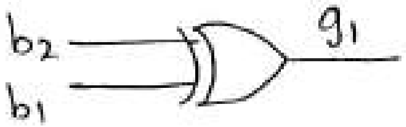
→ $b_2\bar{b}_1$

→ $\bar{b}_2 b_1$

Mathematical expression for $g_1 = b_2\bar{b}_1 + \bar{b}_2 b_1$

$g_1 = b_2 \oplus b_1$

circuit for g_1



<u>g_0 k-map</u> $m_1, m_2, m_5, m_6, m_9, m_{10}, m_{13}, m_{14}$

b_3b_2 \ b_1b_0	00	01	11	10
00	0	1 1	3	2 1
01	4	5 1	7	6 1
11	12	13 1	15	14 1
10	8	9 1	11	10 1

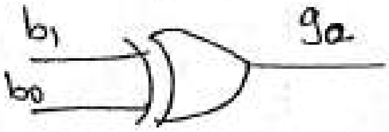
→ $b_1\bar{b}_0$

→ $\bar{b}_1 b_0$

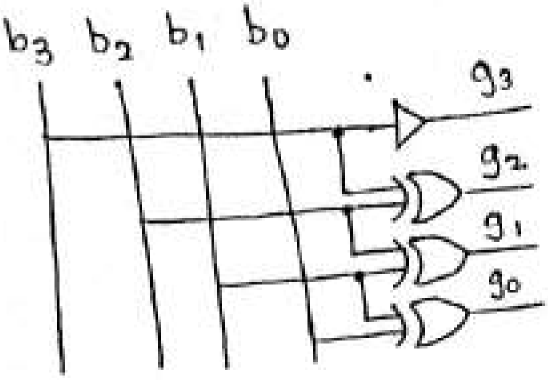
Mathematical expression for $g_0 = \bar{b}_1 b_0 + \bar{b}_0 b_1$

$g_0 = b_1 \oplus b_0$

circuit for g_0



C·C for Binary to Grey Converter



(ii) Grey Code to Binary

→ $b_i = b_{i+1} \oplus g_i$ except for MSB. for $0 \leq i \leq n-1$

Expression for binary code from grey code.

→ Truth table

decimal number	grey code g_3	g_2	g_1	g_0	binary number 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	0	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

→ From truth table determine the min terms

→ $b_3 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$

$b_2 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$

$b_1 = \Sigma m(2, 3, 4, 5, 8, 9, 14, 15)$

$b_0 = \Sigma m(1, 2, 4, 7, 8, 11, 13, 14)$

k-map obtained from above SOP expressions are

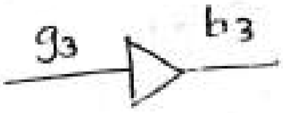
b_3 Kmap

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12 1	13 1	15 1	14 1
10	8 1	9 1	11 1	10 1

→ g_3

Mathematical exp. is $b_3 = g_3 \rightarrow \textcircled{1}$

circuit for b_3



b_2 kmap

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

Mathematical exp. is $b_2 = \overline{g_3}\, g_2 + g_3\, \overline{g_2} \rightarrow \textcircled{2}$

$b_2 = g_3 \oplus g_2 \rightarrow \textcircled{3}$

$b_2 = b_3 \oplus g_2 \rightarrow \textcircled{4}$

b_1 k-map

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00			1	1
01	1	1		
11			1	1
10	1	1		

Math. exp. is $b_1 = \overline{g_3}\,\overline{g_2}\, g_1 + \overline{g_3}\, g_2\, \overline{g_1} + g_3 g_2 g_1 + g_3\, \overline{g_2}\, \overline{g_1}$

$b_1 = g_3 \oplus g_2 \oplus g_1 \rightarrow \textcircled{5}$

$b_1 = b_2 \oplus g_1 \rightarrow \textcircled{6}$

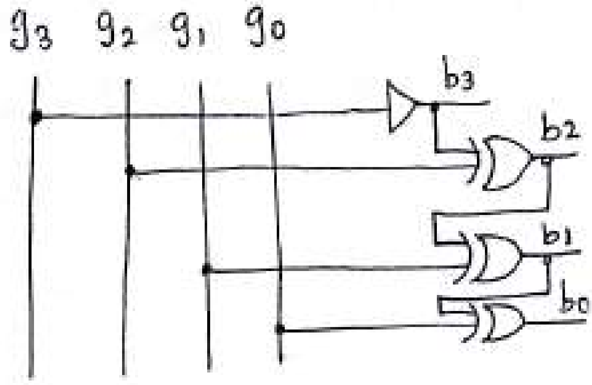
b_0 K-map

$g_3g_2 \backslash g_1g_0$	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$$b_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0.$$

$$b_0 = b_1 \oplus g_0 \rightarrow \textcircled{7}$$

C.C for Grey to Binary Converter



Adders:

Adders are of two types

① Half. Adder , ② Full Adder.

Half Adder: It has two inputs and two outputs.

TRUTH TABLE

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

Consider k-map of two variable.

Sum ⟹

A \ B	0	1
0	0	1 1
1	2 1	3

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

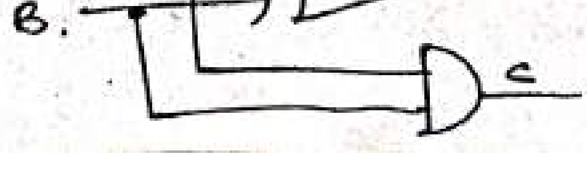
Sum. $= A \oplus B$

Carry ⟹

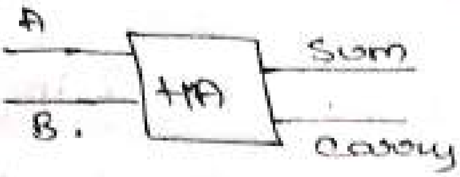
A \ B	0	1
0	0	1
1	2	3 1

$C = AB.$

Circuit



Top Module.



Full Adder (***):

Q] Design a full adder (or) design a full adder using 2 half adders.

Sol: Full Adder:

⇒ three inputs & two outputs

A	B	(input carry) Cin	sum (S)	Carry (C)
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

Sum K-map

A \ BCin	00	01	11	10
0	^{0}	^{1} 1	^{3}	^{2} 1
1	^{4} 1	^{5}	^{7} 1	^{6}

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

$= \bar{A}(\bar{B}C_{in} + B\bar{C}_{in}) + A(\bar{B}\bar{C}_{in} + BC_{in})$

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

$\Rightarrow \bar{A}(B \oplus C_{in}) + A(B \odot C_{in})$

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

Let $B \oplus C_{in} = X$

$\Rightarrow \bar{A}X + A\bar{X} \Rightarrow A \oplus X$.

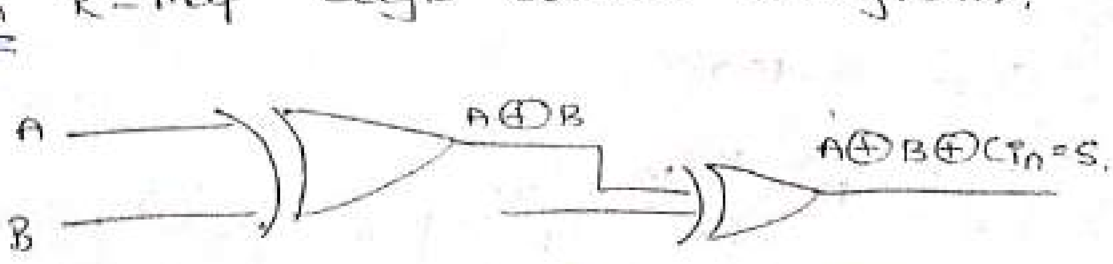
$\therefore S = A \oplus (B \oplus C_{in})$

Carry K-map

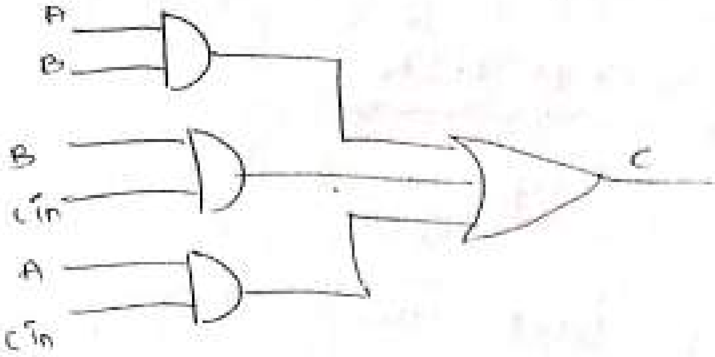
A \ BCin	00	01	11	10
0			1	
1		1	1	1

$C = AB + BC_{in} + AC_{in}.$

Sum K-map Logic Circuit Diagram:

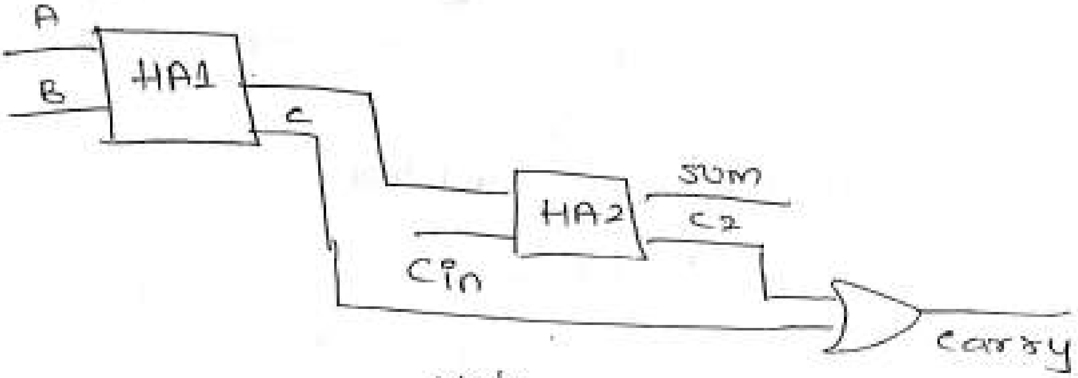


Carry K-map Logic Circuit Diagram:



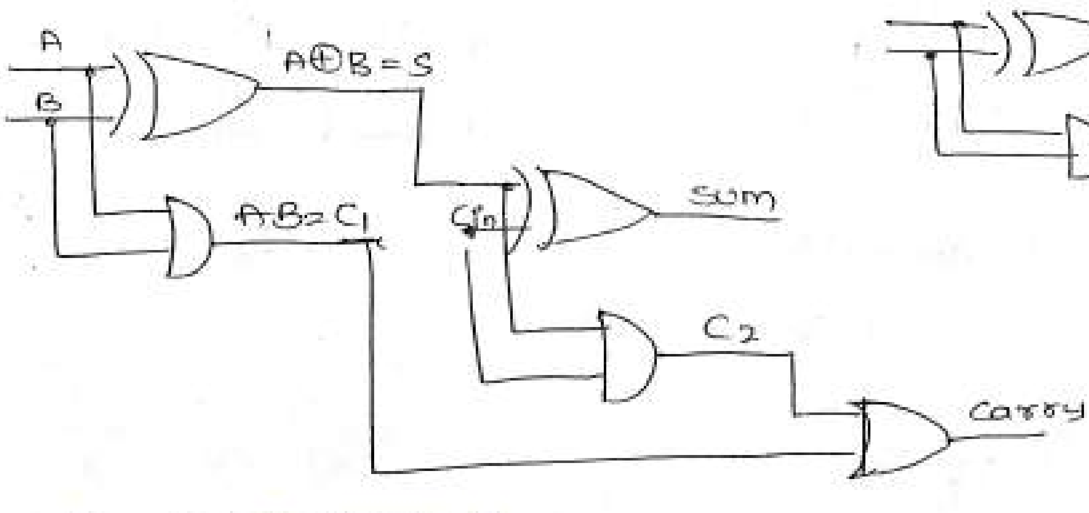
Design a full adder using two half adders:

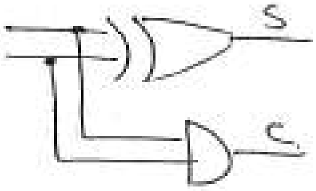
Let us consider three variables A B C



LOGIC CIRCUIT (for above)

Note
HA { sum $S = A \oplus B$
carry $C = AB$





SUBTRACTER: q 2 types

① Half subtractor

② Full subtractor

① HALF SUBTRACTOR

it has two inputs & two outputs

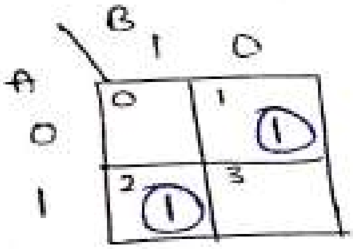
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

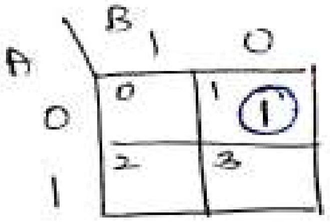
① Determine variables

② Truth table

③ Simplify SOP (or) POS

④ · Logic circuit

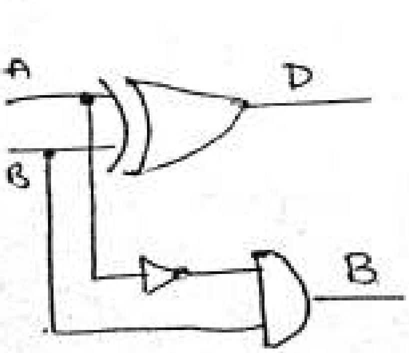




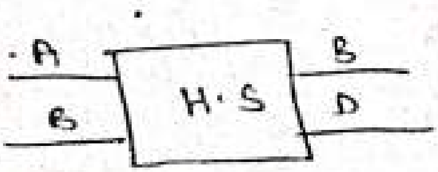
$D = A\bar{B} + \bar{A}B$

$B = \bar{A}B$

$D = A \oplus B$



⇒ HALF SUBTRACTOR CIRCUIT.



⇒ TOP MODULE

BCD to Excess-3 Convertor:

3 → 0011

	BCD				Excess-3			
	B_0	B_1	B_2	B_3	X_0	X_1	X_2	X_3
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
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

$X_0 = \Sigma m(5,6,7,8,9) + d(10,11,12,13,14,15)$

$X_1 = \Sigma m(1,2,3,4,9) + d(10,11,12,13,14,15)$

$X_2 = \Sigma m(0,3,4,7,8) + d(10,11,12,13,14,15)$

$X_3 = \Sigma m(0,2,4,6,8) + d(10,11,12,13,14,15)$

B_0B_1 \ B_2B_3	00	01	11	10
00	0	1	3	2
01	4	5 1	7 1	6 1
10	12 X	13 X	15 X	14 X
10	8 1	9 1	11 X	10 X

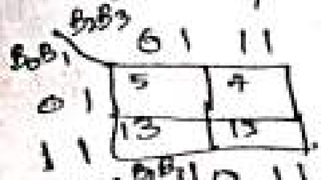
~~quad~~ octet is possible.

B_0B_1 B_2 B_3 changing

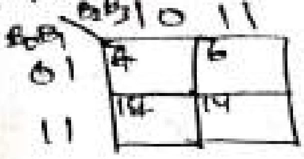
1 1 (i) B_0.

1 0

quad
~~two pair~~ is possible:



(ii) $B_1 B_3$.



(iii) $B_1 B_2$

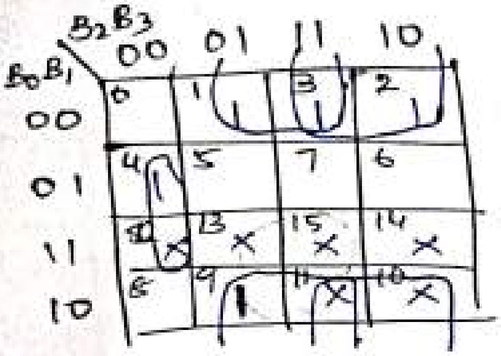
~~$X_0 = B_0 + \bar{B}_0 B_1 + B_3$~~

~~$+ B_1 + B_2 \bar{B}_3$~~

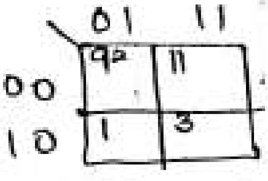
~~$X_0 = B_1(1+\bar{B}_0) + B_2\bar{B}_3 + B_3$~~

~~$X_0 = B_1 + B_2\bar{B}_3 + B_3$~~

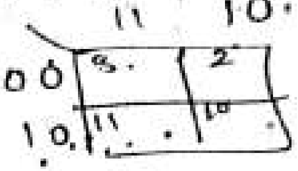
~~$X_0 =$~~ $X_0 = B_0 + B_1B_3 + B_1B_2.$



quad is possible.

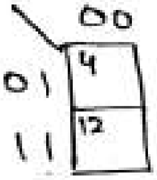


(i) $\bar{B}_1 B_3$



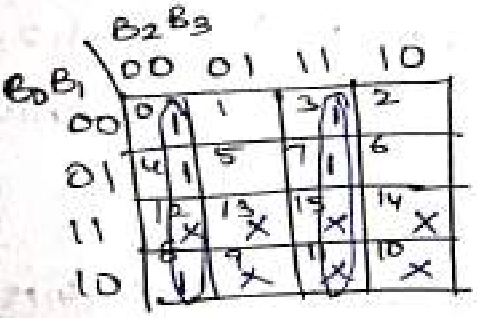
(ii) $\bar{B}_1 B_2$.

(iv) one pair



$B_1 \bar{B}_2 \bar{B}_3$.

$X_1 = \bar{B}_1 B_3 + \bar{B}_1 B_2 + B_1 \bar{B}_2 \bar{B}_3$.



2quads are possible:

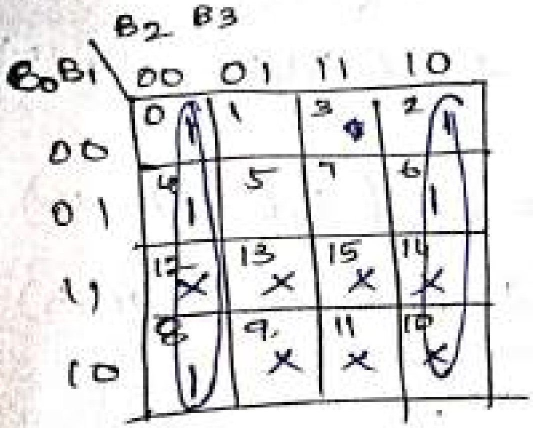
$B_0 B_1$ changing

$B_2 B_3$

0 0   1 1

$X_2 = \bar{B}_2 \bar{B}_3 + B_2 B_3$.

$X_2 = B_2 \odot B_3$.



octet ~~2 quads~~ is ~~are~~ possible

$B_0 B_1$ changing

$B_2 B_3$

0 0 $= \bar{B}_3$

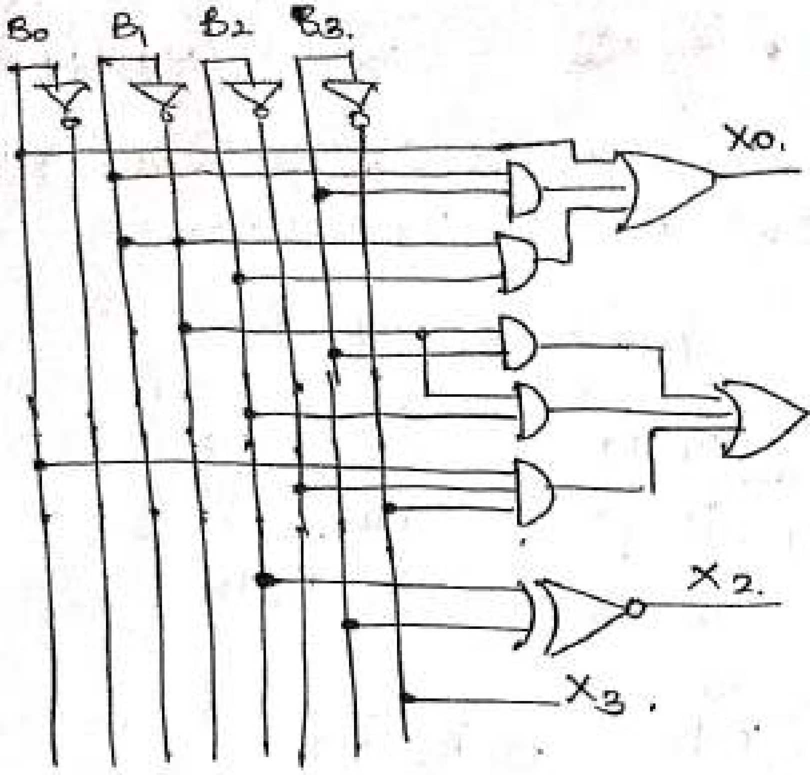
1 0

$X_3 = \bar{B}_3$

$X_0 = B_0 + B_1 B_3 + B_1 B_2$ $X_1 = \overline{B_1} B_3 + \overline{B_1} B_2 + B_1 \overline{B_2}\,\overline{B_3}$

$X_3 = \overline{B_3}$ $X_2 = B_2 \odot B_3$

~~$X_2 = \overline{B_2}\,\overline{B_3} + B_2 \overline{B_3}$~~



LOGIC

CIRCUIT

Excess- 3 to BCD Code Convertor:

	X_0	X_1	X_2	X_3	B_0	B_1	B_2	B_3
0	0	0	0	0	X	X	X	X
1	0	0	0	1	X	X	X	X
2	0	0	1	0	X	X	X	X
3	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	0	1	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	0	1	0	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	1	0	0	1
12	1	1	0	0	1	0	1	0
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

$B_0 = \Sigma m(11, 12) + d(0, 1, 2, 13, 14, 15)$

$B_1 = \Sigma m(7, 8, 9, 10) + d(0, 1, 2, 13, 14, 15)$

$B_2 = \Sigma m(5, 6, 9, 10, 12) + d(0, 1, 2, 13, 14, 15)$

$B_3 = \Sigma m(4, 6, 8, 10, 11) + d(0, 1, 2, 13, 14, 15)$

x_0x_1 \ x_2x_3	00	01	11	10
00	0 X	1 X	3	2 X
01	4	5	7	6
11	12 1	13 X	15 X	14 X
10	8	9	11 1	10

quad is possible

x_0 x_1 x_2x_3 is changing

1   1

(i) = x_0x_1

pair is possible

	11
11	15
10	11

$= x_0x_2x_3$.

$B_0 = x_0x_1 + x_0x_2x_3$.

x_0x_1 \ x_2x_3	00	01	11	10
00	0 X	1 X	3	2 X
01	4	5	7 1	6
11	12	13 X	15 X	14 X
10	8 1	9 1	11	10 1

quad is possible.

x_0x_1 \ x_2x_3	00	01
00	0	1
10	8	9

(i) $\bar{x_1}\,\bar{x_2}$

2 pairs

	11
01	7
11	15

(ii) $x_1x_2x_3$.

	10
11	14
10	10

(iii) $x_0x_2\bar{x_3}$

$B_1 = \bar{x_1}\,\bar{x_2} + x_1x_2x_3 + x_0x_2\bar{x_3}$

x_0x_1 \ x_2x_3	00	01	11	10
00	0 X	1 X	3	2 X
01	4	5 1	7	6 1
11	12 1	13 X	15 X	14 X
10	8	9 1	11	10 1

1 octet, quads are possible.

x_2x_3 changing

x_0x_1				
11	12 1	13 X	15 X	14 X

x_0x_1

x_0x_1 changing

x_2x_3				
01	1 X	5 1	13 X	9 1

$\bar{x_2}x_3$.

x_0x_1 changing

x_2x_3				
10	2 X	6 1	14 X	10 1

$x_2\bar{x_3}$

$B_2 = x_0x_1 + \bar{x}_2x_3 + x_2\bar{x_3}$

$B_2 = x_0x_1 + x_2 \oplus x_3$.

x_0x_1 \ x_2x_3	00	01	11	10
00	0 X	1 X	3	2 X
01	4 1	5	7	6 1
11	12	13 X	15 X	14 X
10	8 1	9	11 1	10 1

2 quads.

x_0x_1 \ x_2x_3	00	10
01	4	6
10	8	10

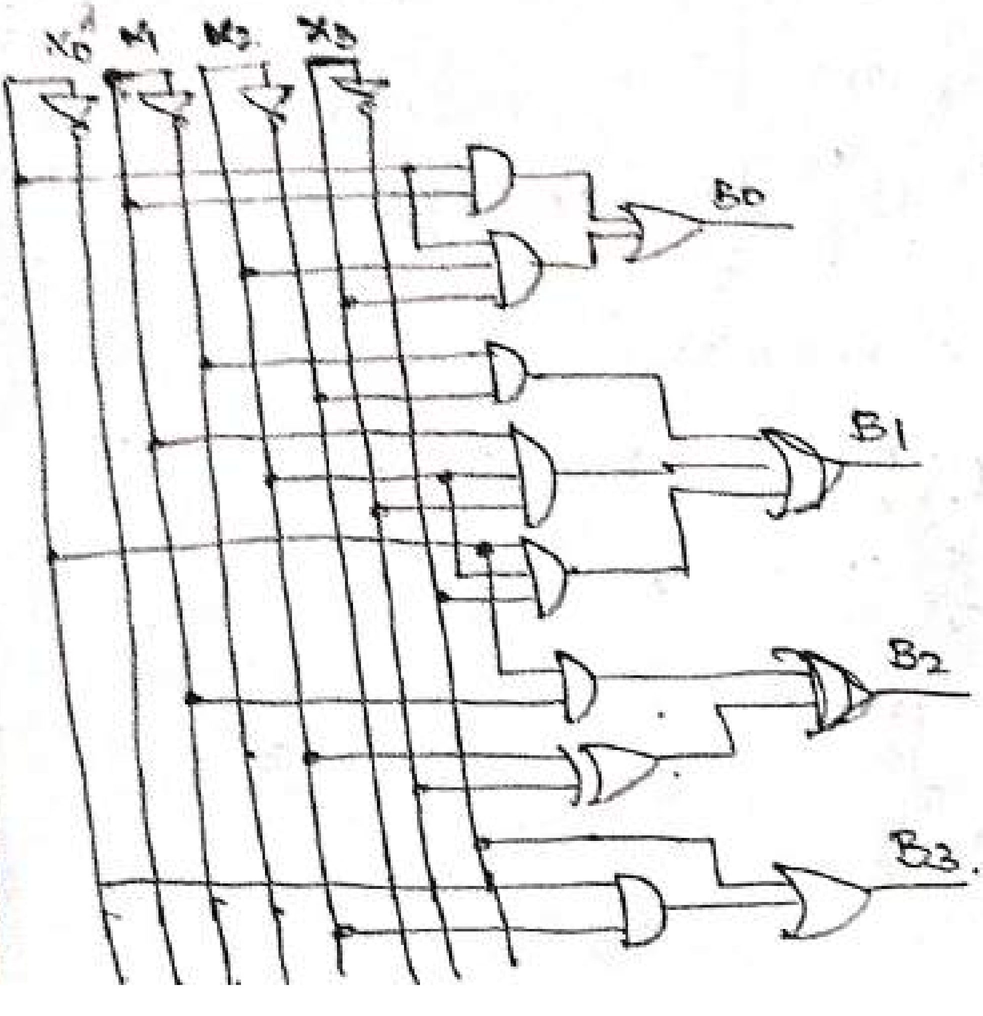
x_0x_1 \ x_2x_3	11	10
11	15	14
10	11	10

$\bar{x_3}$ x_0x_2.

$B_3 = \bar{x_3} + x_0x_2$.

$B_0 = x_0x_1 + x_0x_2x_3$ $B_1 = \overline{x_1}\,\overline{x_2} + x_1x_2x_3 + x_0x_2\overline{x_3}$

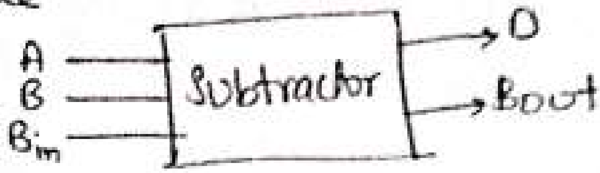
$B_2 = x_0x_1 + x_2 \oplus x_3$ $B_3 = \overline{x_3} + x_0x_2$



FULL SUBTRACTOR

- * A full subtractor is a combinational circuit that perform subtraction of 2bits, one is minuend and other is subtrahend, taking into account the borrow of the previous adjacent lower minuend bit.
- * This circuit has 3 i/p's & 2 o/p's.
- * The 3 i/p's A, B & Bin, denotes minuend, subtrahend & previous borrow respectively. The 2 o/p's D & Bout represent the defference and o/p borrow respectively. Although subtraction is usually achieved by adding the complement of subtrahend to the minuend we will do the subtraction operation.

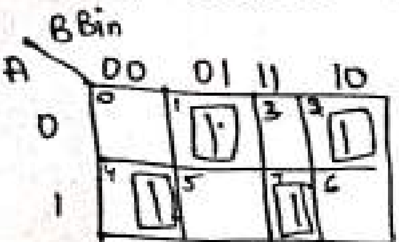
Top Module



Truth Table

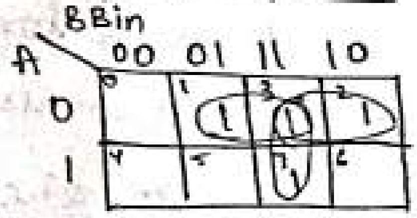
$(d)_{10}$	A	B	Bin	D	Bout
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

From above T.T we can draw K-map for D & Bout



A \ BBin	00	01	11	10
0		1		1
1	1		1	

$D: A\bar{B}\bar{B}_{in} + \bar{A}\bar{B}B_{in} + ABB_{in} + AB\bar{B}_{in}$



A \ BBin	00	01	11	10
0		1	1	1
1			1	

$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$

Logical Expression for Difference.

$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

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

$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$ Let $x = A \oplus B$

$= B_{in}\bar{x} + \bar{B}_{in}x$

$= x \oplus B_{in}$

$D = (A \oplus B) \oplus B_{in}$

Logical Expression for Borrow

$B_{out} = \bar{B}\bar{A} B_{in} + \bar{A} B\bar{B}_{in} + \bar{A}B.B_{in} + ABB_{in}$

$= \bar{A}\bar{B} B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}BB_{in} + \bar{A}BB_{in} + ABB_{in}$

$= \bar{A}B_{in}(B+\bar{B}) + \bar{A}B(B_{in}+\bar{B}_{in}) + BB_{in}(A+\bar{A})$

$= \bar{A}B_{in} + \bar{A}B + BB_{in}$

(or)

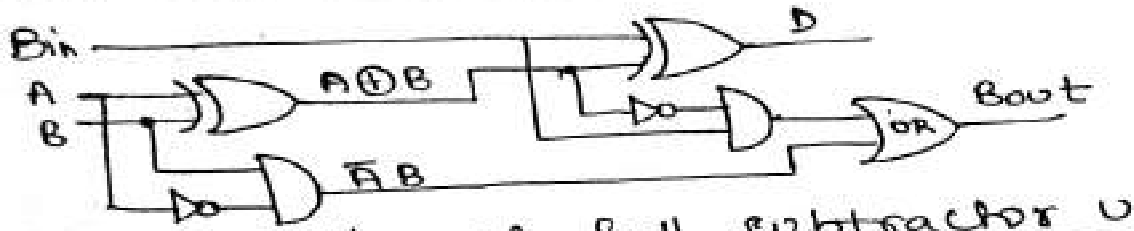
$B_{out} = \bar{A}\bar{B} B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}BB_{in} + ABB_{in}$

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

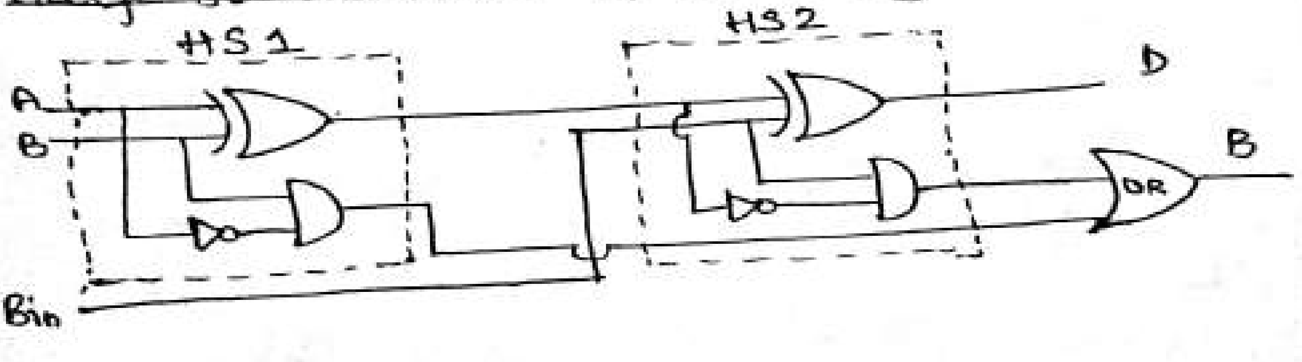
$= B_{in}(\overline{A\oplus B}) + \bar{A}B$

$= B_{in}(A\odot B) + \bar{A}B$

Logical circuit for full subtractor

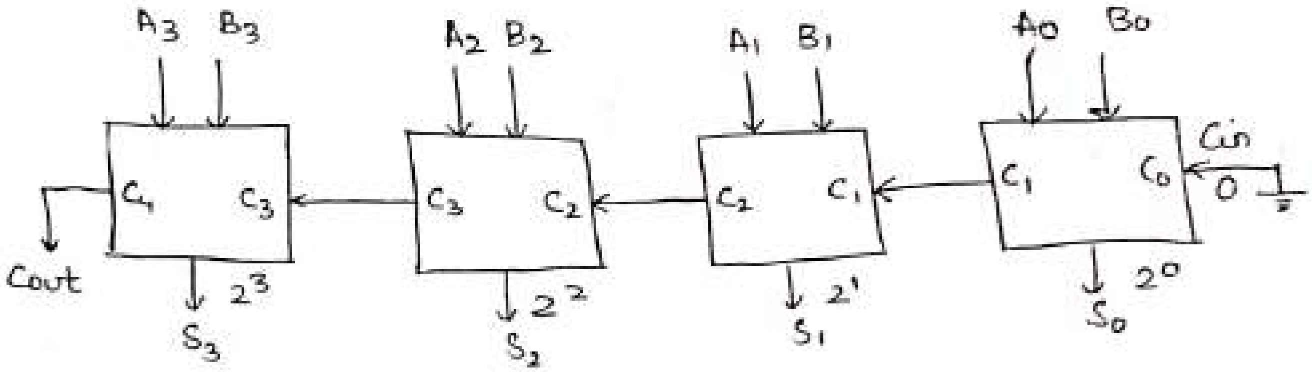


Implementation of full subtractor using 2 half subtractors & an OR gate



* BINARY ADDER (Parallel Binary Adder)

* Addition of multiple bit binary numbers can be accomplished by using full adders.
* The 4- bit adder using full adder circuits is capable of adding two 4-bit numbers resulting in a 4- bit sum and a carry output as shown below in figure.
* Since all bits of augend & addend are fed into the adder circuit simultaneously and the additions in each position are taking place at same time, the circuit is called parallel adder.



4- bit parallel adder

* Addition operation is illustrated below.

Let $A_4 A_3 A_2 A_1 = 1111$ $B_3 B_2 B_1 B_0 = 0011$.

Signficant Place		4	3	2	1
Input Carry		1	1	1	0
Augend word A:		1	1	1	1
Augend word B:		0	0	1	1
	1	0	0	1	0 ← Sum

↑
Cout (output carry)

* In a 4-bit parallel binary adder circuit, the i/p to each full adder will be Ai, Bi & Ci and o/p will be Si & C_{i+1} where i varies from 0 to 3.

* Also, the Cout of lower order is carried forward to next higher order state. Hence, this type of adder is called ripple carry adder.

* Disadvantage Though the parallel binary adder is said to generate o/p immediately after the i/p's are applied, its speed of operation is limited by the carry propogation delay through all stages.

* The propogation delay (tp) of a full adder is the time difference between the instants at which the inputs (Ai, Bi and Ci) are applied and the instant at which its o/p's (Si and C_{i+1}) are generated. ∴, the o/p in LSB stage is generated only after tp seconds. $||^{ly}$, the o/p in the second stage will be generated only after the tp seconds from the time the o/p's ~~are~~ of the first stage are generated. ie after 2tp seconds from the time the i/p's are applied and similarly for remaining stages. ∴, for 4-bit binary adder, where each F.A has a propogation delay of 50ns, the o/p of the 4th stage is $4tp = 4 \times 50ns = 200ns$.

* To overcome this disadvantage we go for carry look ahead adder.

* BINARY MULTIPLIER

* Multiplication operation can be carried out by
(i) multipliers using partial product addition and shifting
& (ii) Parallel multipliers.

(i) Multiplier Using Shift Method

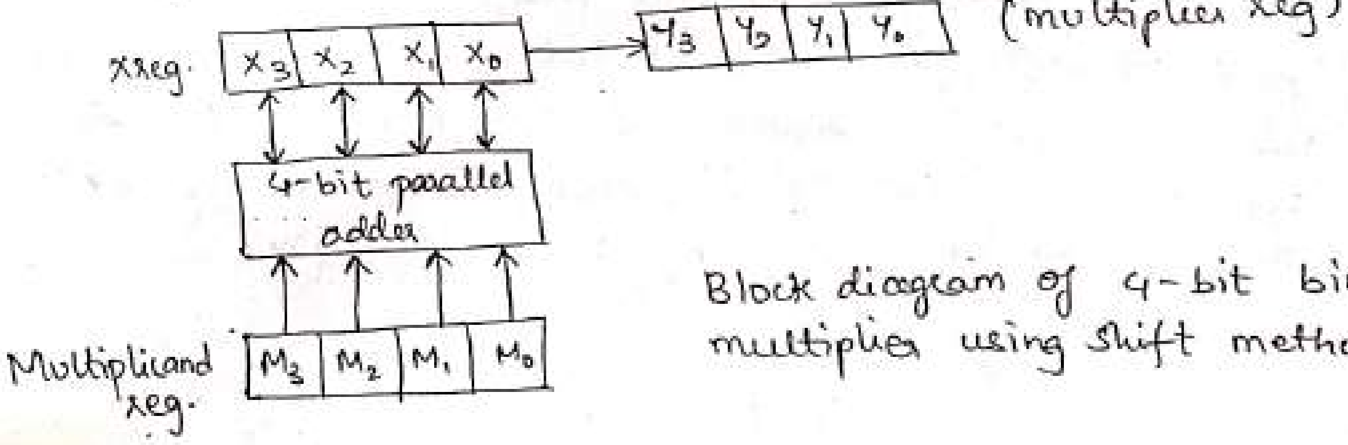
→ Consider, the multiplication of 2 4-bit binary no. 1010 & 1011, as an example.

```
       1 0 1 0    → Multiplicand
     x 1 0 1 1    → Multiplier
     ---------
       1 0 1 0    → Partial product 1
     1 0 1 0      →   "      "      2
   0 0 0 0        →   "      "      3
 1 0 1 0          →   "      "      4
 ---------------
 1 1 0 1 1 1 0    → Product
```

→ From above multiplication process,
if multiplier bit = 1, then multiplicand = partial prd.
= 0, then partial prd. = 0.

→ Whenever a P.P is obtained, it is shifted one bit to the left of the previous P.P. This process is continued until all the multiplier bits are checked, & then the P.P are added.

→ Implementation:



Block diagram of 4-bit binary multiplier using shift method.

→ In above ~~abov~~ diagram, the 4-bit multiplier is stored in register Y ($Y_3 Y_2 Y_1 Y_0$); the 4-bit multiplicand is stored in register M ($M_3 M_2 M_1 M_0$) & the X register (X_4, X_3, X_2, X_1, X_0) is initially cleared to 00000.

→ Here, the LSB of multiplier bit Y_0 is checked.

If $Y_0 = 1$ ⇒ the number in M is added with the LSB of X register and the combined X & Y register is shifted to the right by 1 bit.

If $Y_0 = 0$ ⇒ the combined X & Y register is shifted to right by 1 bit without performing addition.

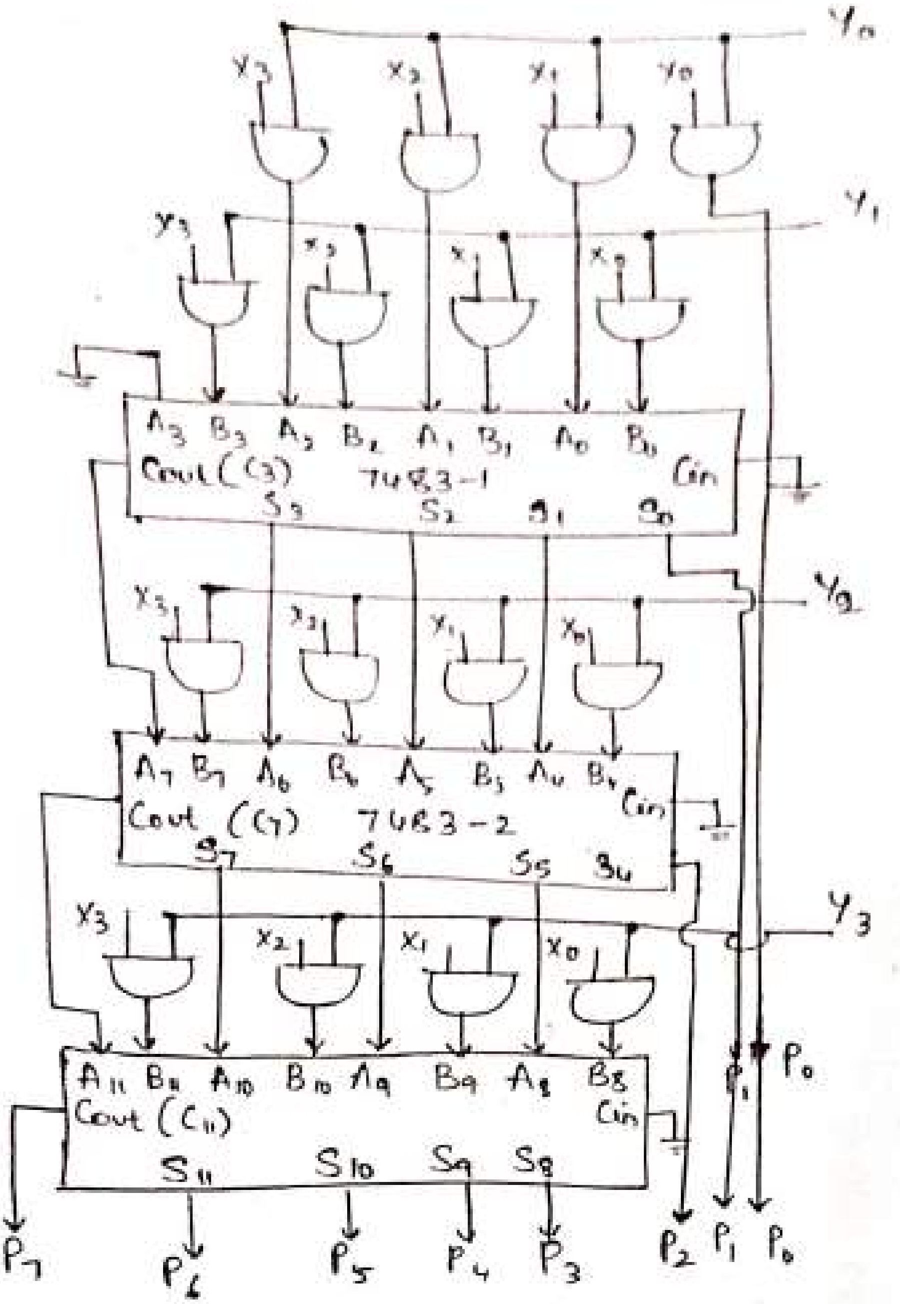
This process is repeated 4 times to perform a 4-bit multiplication.

→ The multiplication result ($R_7, R_6, R_5, R_4 R_3 R_2 R_1 R_0$) will be available in X & Y registers.

Parallel Multiplier

→ The 4-bit multiplier using shift method requires 4 cycles of addition & shifting operations, but it requires only a single 4-bit parallel adder.

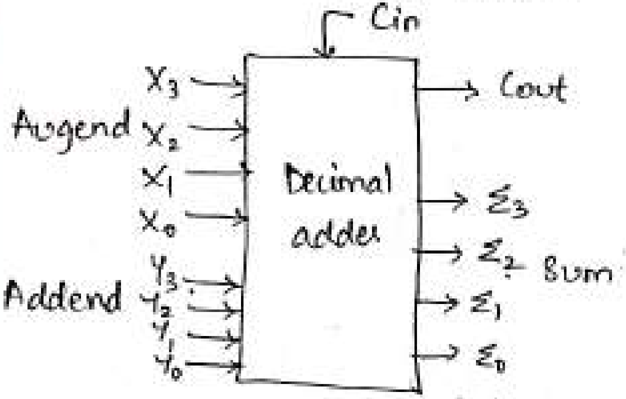
→ It requires 3 4-bit parallel binary adders & 16 numbers of 2-i/p AND gates. Here, each group of 4 AND gates is used to obtain p.p while 4-bit parallel adders are used to add the P.P. Since, the generation of P.P & their additions are performed in parallel in group of AND gates and 4-bit adders respectively, the multiplication result ($P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0$) will be available at the o/p respectively immediately after the propogation delay in the multiplier circuit.



4-bit parallel multiplier.

* BCD Adder

* A BCD adder is a circuit that adds 2 BCD digits in parallel & produces a sum digit which is also BCD.
* A BCD adder must include the correction logic in its internal construction.
* This adder has 2 4-bit BCD i/p's $X_3, X_2, X_1, X_0, Y_3, Y_2, Y_1, Y_0$ and a carry i/p C_{in}. It also has a 4·bit ~~B~~ sum o/p $Z_3 Z_2 Z_1 Z_0$ and a carry o/p Cout.



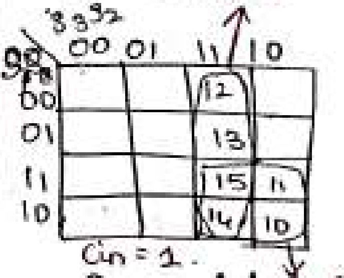
Block diagram of BCD Adder.

*

Decimal digit	Uncorrected BCD sum C_{in}	S_3	S_2	S_1	S_0	Corrected BCD Sum Cout	S_3	S_2	S_1	S_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	0
3		0	0	1	1		0	0	1	1
4		0	1	0	0		0	1	0	0
5		0	1	0	1		0	1	0	1
6		0	1	1	0		0	1	1	0
7		0	1	1	1		0	1	1	1
8		1	0	0	0		1	0	0	0
9		1	0	0	1		1	0	0	1
10		1	0	1	0	1	0	0	0	0
11		1	0	1	1	1	0	0	0	1
12		1	1	0	0	1	0	0	1	0
13		1	1	0	1	1	0	0	1	1
14		1	1	1	0	1	0	1	0	0
15		1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

No correction required

Correction Required



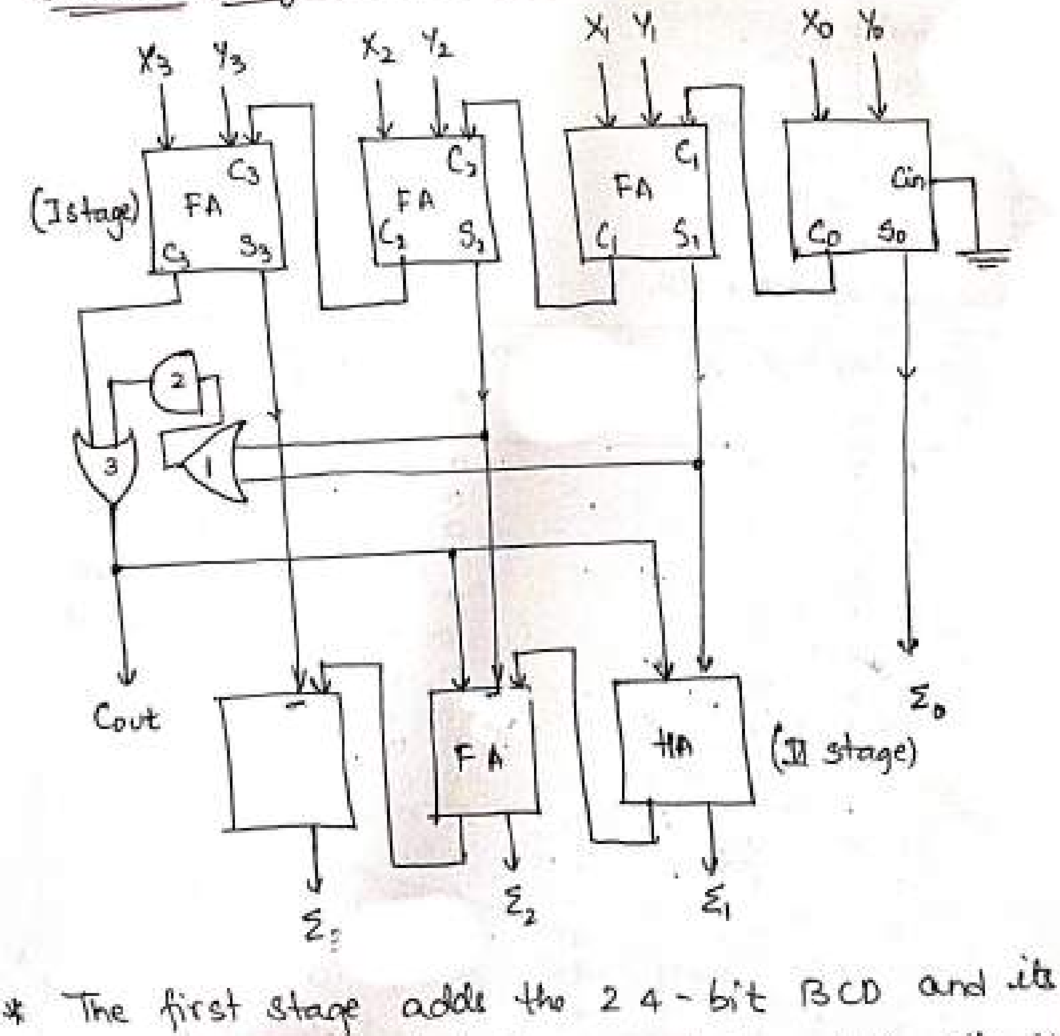
* When the sum o/p ($S_3 S_2 S_1 S_0$) is greater than 9, add 0110 to get BCD result. So the correction can be written as an expression as follows

$$C_n = S_3 S_2 + S_3 S_1 + C_{in}.$$

* A BCD adder circuit must be able to do ~~following~~ following:

1. Add two 4-bit BCD numbers using straight binary addition.
2. If the 4-bit sum is =< 9, the sum is in proper BCD form and no correction is required.
3. If the 4-bit sum is >9 9(or) if a carry is generated from the sum, the sum is not in BCD form. Then, the digit 6(0110) should be added to the sum to produce the BCD results. The carry maybe produced due to this addition and it is added to next decimal position.

Circuit diagram for BCD adder using full adders



* The first stage adds the 2 4-bit BCD and its sum and carry are checked to ascertain whether the result exceeds by 9 by AND-OR gate combinations. If the o/p of OR gate (3) is equal to 1, then correction is required & this is accomplished by adding 0110 in the second stage of adders.

COMPARATOR

Which compares the magnitude of two numbers.

1) One- bit comparator

A	B	A< B	A> B	A = B
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

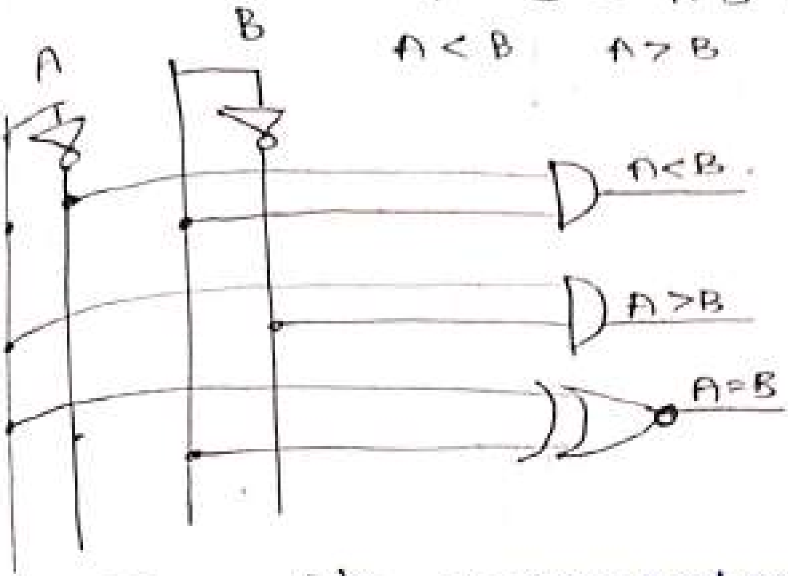
expressions

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

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

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

A<B A>B A=B → (X-NOR) gate

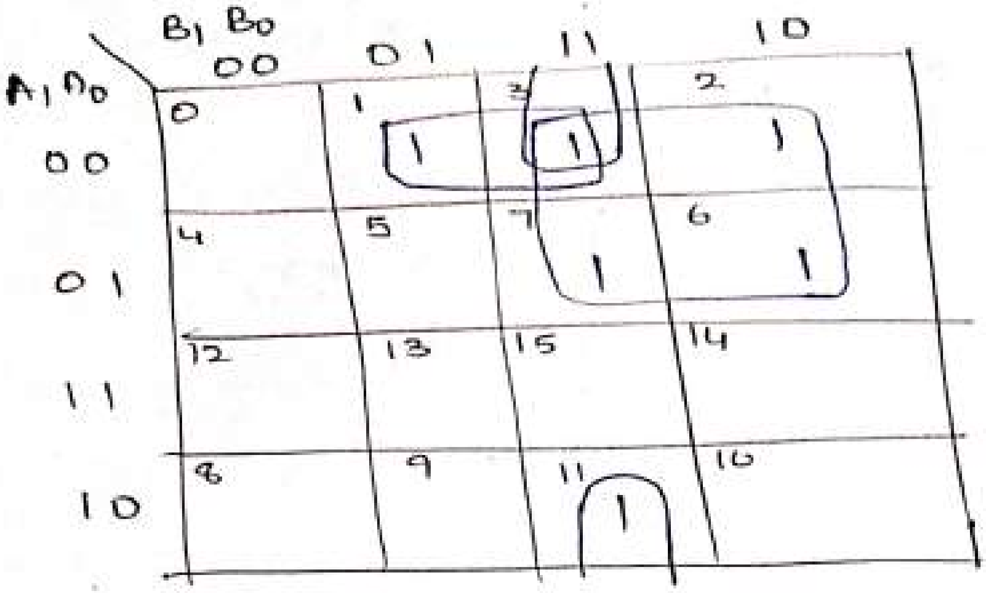


2) Two-bit comparator:

	A		B		A< B	A > B	A = B	
	A_1	A_0	B_1	B_0				
1	0	0	0	0	0	0	1	→ m_0
2	0	0	0	1	1	0	0	m_1
3	0	0	1	0	1	0	0	m_2
4	0	0	1	1	1	0	0	m_3
5	0	1	0	0	0	1	0	m_4
6	0	1	0	1	0	0	1	m_5
7	0	1	1	0	1	0	0	m_6
8	0	1	1	1	1	0	0	m_7
9	1	0	0	0	0	1	0	m_8
10	1	0	0	1	0	1	0	m_9
11	1	0	1	0	0	0	1	m_{10}
12	1	0	1	1	1	0	0	m_{11}
13	1	1	0	0	0	1	0	m_{12}
14	1	1	0	1	0	1	0	m_{13}
15	1	1	1	0	0	1	0	M_{14}
16	1	1	1	1	0	0	1	M_{15}

$\boxed{A < B}$

~~A<B=~~ Its a four variable K-map



A_1A_0 \ B_1B_0	00	01	11	10
00	0	1 1	3 1	2 1
01	4	5	7 1	6 1
11	12	13	15	14
10	8	9	11 1	10

	11	10
00	3 1	2 1
01	7 1	6 1

$A_1 = 0 \quad B_1 = 0$

$\Rightarrow \overline{A_1}\ \overline{B_1}$

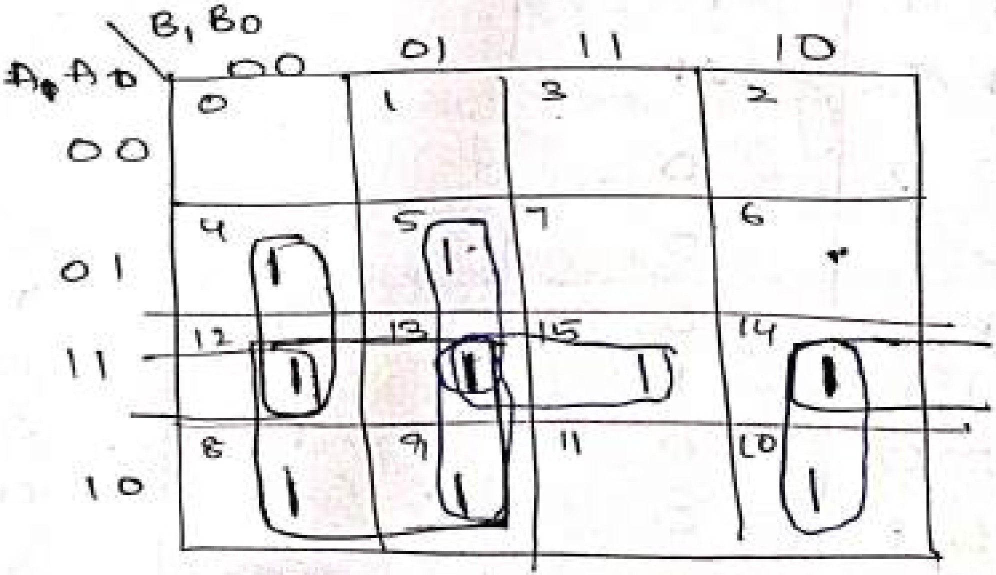
	01	11
00	1	3

$\overline{A_0}\ \overline{A_1}\ \overline{B_0}$

	11
00	3
10	11

$\overline{A_0}\ \overline{B_1}\ \overline{B_0}$.

$A < B = \overline{A_1}\,\overline{B_1} + \overline{A_1}\,\overline{B_0}\,\overline{A_0} + \overline{B_1}\,\overline{A_0}\,\overline{B_0}$



A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	3	2
01	4 1	5 1	7	6
11	12 1	13 1	15 1	14 1
10	8 1	9 1	11	10 1

Black V

Blue X.

	00	01
11	12	13
10	8	9

$A_1 = 1$ $B_1 = 0$

$A_1\overline{B_1}$

	00
01	4
11	12

$A_0 = 1$ $B_0 = 0$ $B_1 = 0$

$A_0\,\overline{B_0}\,\overline{B_1}$

	00	10
11	12	14

$A_0 = 1$ $A_1 = 1$ $B_0 = 0$.

$A_0 A_1 \overline{B_0}$.

$A > B = A_1\overline{B_1} + A_0\overline{B_0}\,\overline{B_1} + A_0A_1\overline{B_0}$.

$A = B$.

A_1A_0 \ B_1B_0	00	01	11	10
00	0 1	1	3	2
01	4	5 1	7	6
11	12	13	15 1	14
10.	8	9	11	10 1

$$(A=B) = \overline{A}_1\overline{A}_0\overline{B}_1\overline{B}_0 + \underbrace{\overline{A}_1A_0\overline{B}_1B_0 + A_1A_0B_1B_0}_{A_0B_0} + A_1\overline{A}_0B_1\overline{B}_0$$

$$\overline{A}_0\overline{B}_0$$

$$= A_0B_0(\overline{A}_1\overline{B}_1 + A_1B_1) + \overline{A}_0\overline{B}_0(\overline{A}_1\overline{B}_1 + A_1B_1)$$

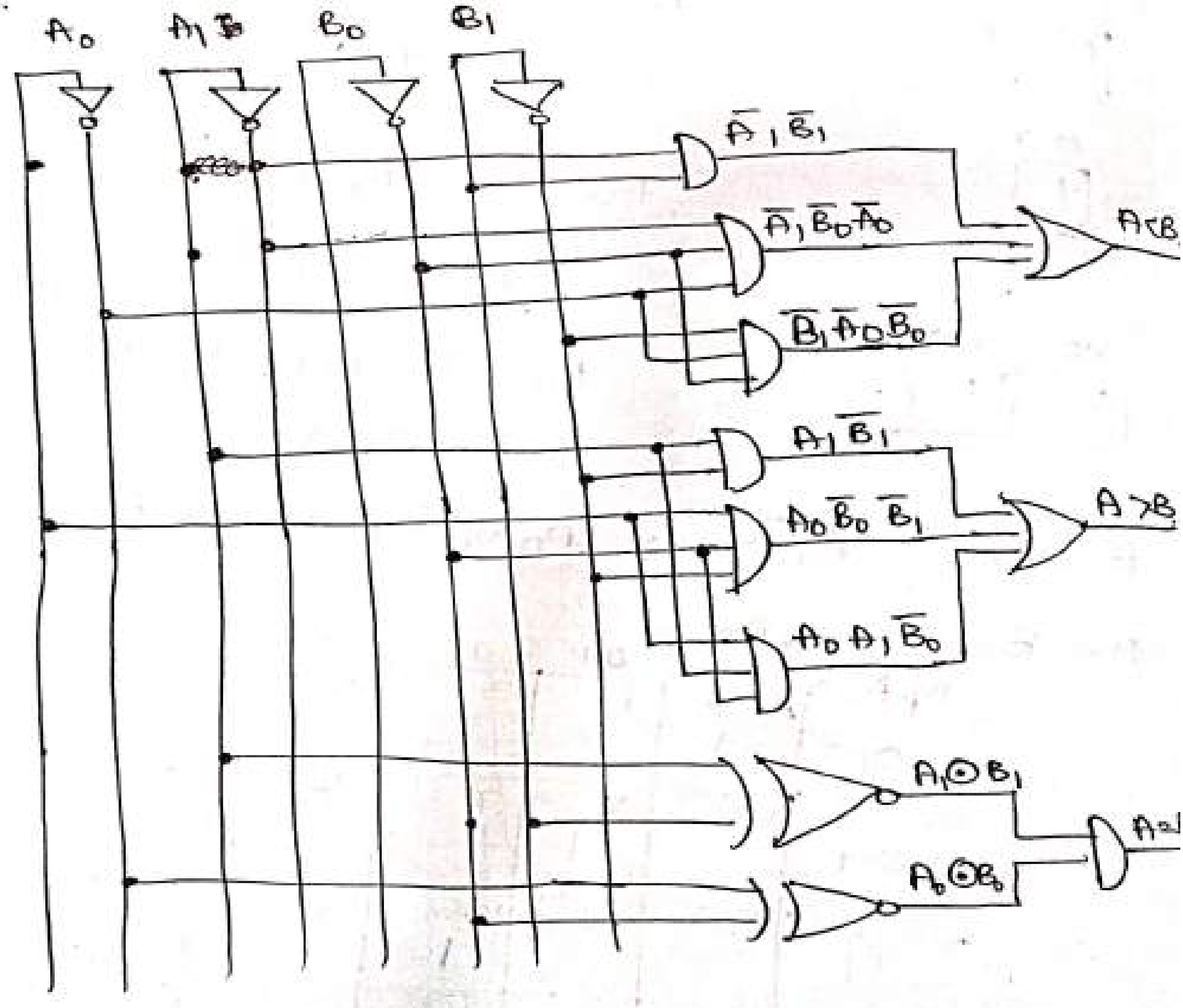
$$= (\overline{A}_1\overline{B}_1 + A_1B_1)(A_0B_0 + \overline{A}_0\overline{B}_0)$$

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

$$A<B = \overline{A}_1\overline{B}_1 + \overline{A}_1\overline{B}_0\overline{A}_0 + \overline{B}_1\overline{A}_0\overline{B}_0$$

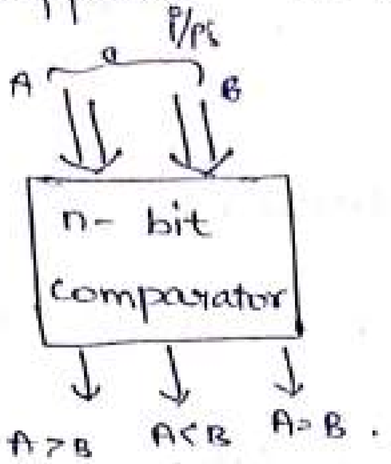
= ~~$A_0B_0(A_1$~~ ~~$A_0B_0(A_1+B$~~

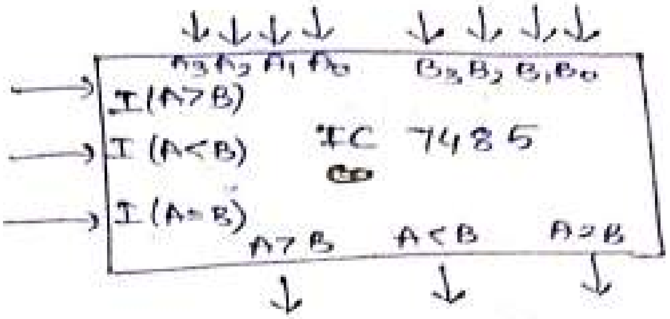
$$A>B = A_1\overline{B}_1 + A_0\overline{B}_0\overline{B}_1 + A_0A_1\overline{B}_0$$



A_0
A_1
B_0
B_1
$\overline{A}_1\overline{B}_1$
$\overline{A}_1\overline{B}_0\overline{A}_0$
$\overline{B}_1\overline{A}_0\overline{B}_0$
A<B
$A_1\overline{B}_1$
$A_0\overline{B}_0\overline{B}_1$
$A_0A_1\overline{B}_0$
A>B
$A_1\odot B_1$
$A_0\odot B_0$
A=

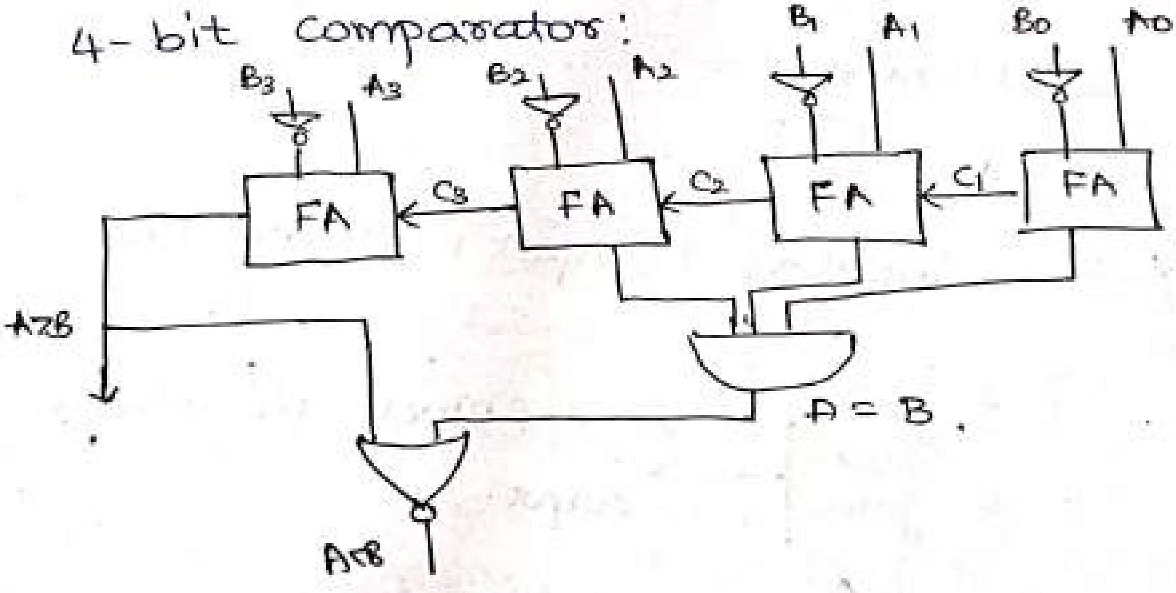
Suppose n- bit complements:





Comparing I/p A B	Cascading I/P.			O/p		
	I(A>B)	I(A<B)	I(A=B)	A>B	A<B	A=B
A>B	x	x	x	1	0	0
A=B	1	0	0	1	0	0
	X	1	X	D	1	0
	0	0	1	0	0	1
	0	0	0	1	0	1
	1	0	1	0	0	0
A<B	X	x	x	0	0	1

4- bit comparator:



MULTIPLEXER

(OR)

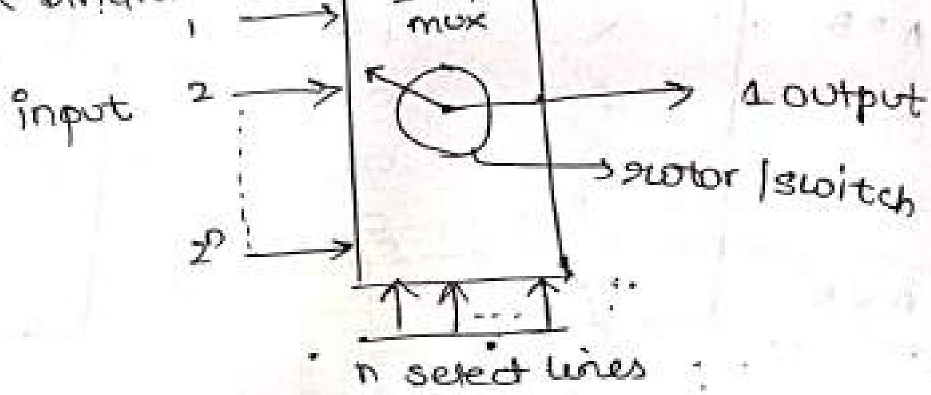
MUX (data Selector)

* Sharing of data through a single common line

* Multiplexing means all data lines are combined (or) punched together.

* A multiplexer has 2^n inputs, 1 output, and n select lines.

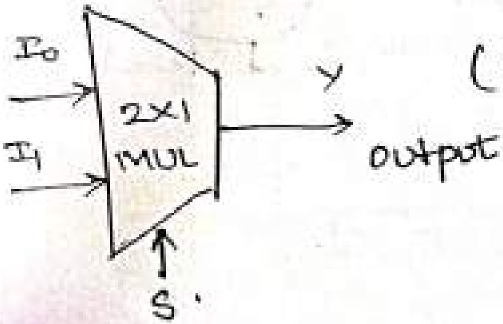
* BLOCK DIAGRAM



* 2 x 1 MUX

Sol: $n = 1$ $\boxed{2^n}$

⇒ 2 inputs, 1 output, 1 select line

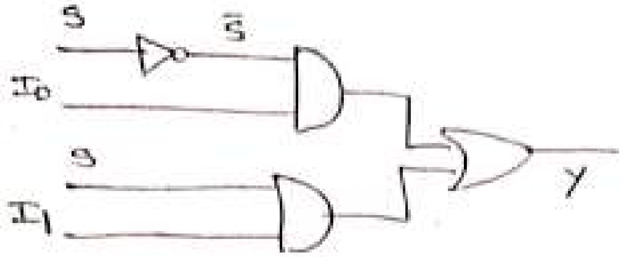


(Block diagram)

S	Y
0	I_0
1	I_1

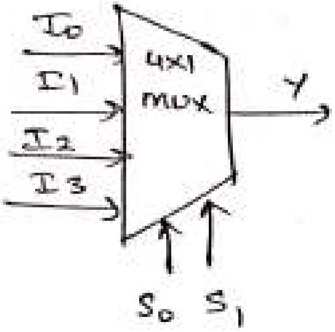
(Truth table).

expression $Y = \bar{S} I_0 + S I_1$



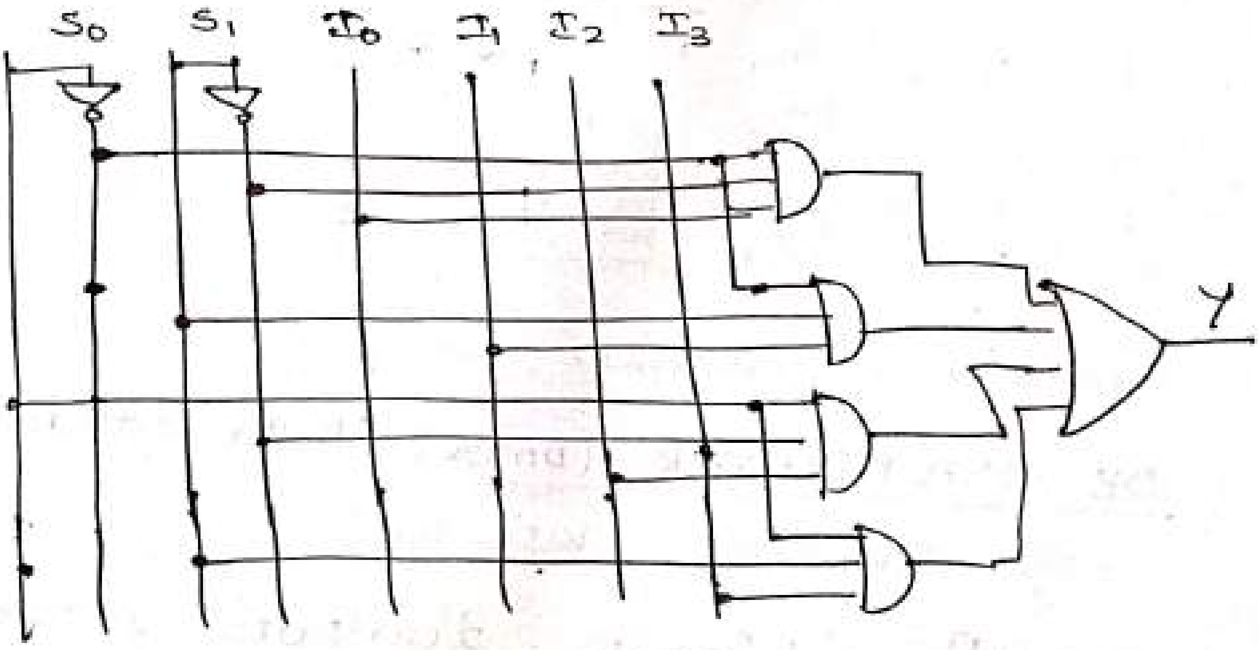
* 4 X 1 MUX.

$2^2 \times 1$ n = 2 Truth table



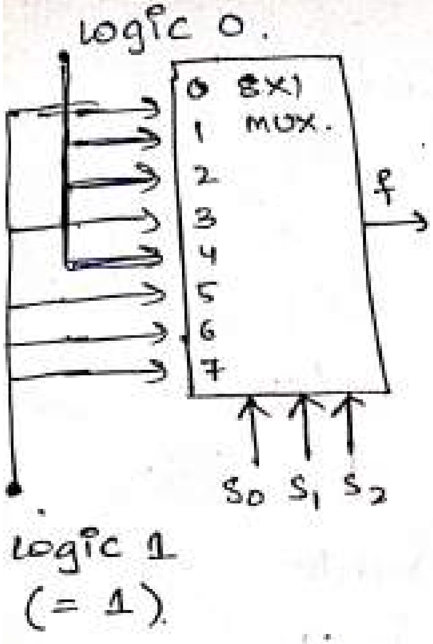
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$Y = \bar{S}_0 \bar{S}_1 I_0 + \bar{S}_0 S_1 I_1 + S_0 \bar{S}_1 I_2 + S_0 S_1 I_3$.



Q) Implement the following f^n using multiplexer. $f = \Sigma m\,(0,3,5,6,7)$.

Sol^n: When n = 3 $2^n = 2^3 = 8 \times 1$ MUX
↓
Select lines



TRUTHTABLE

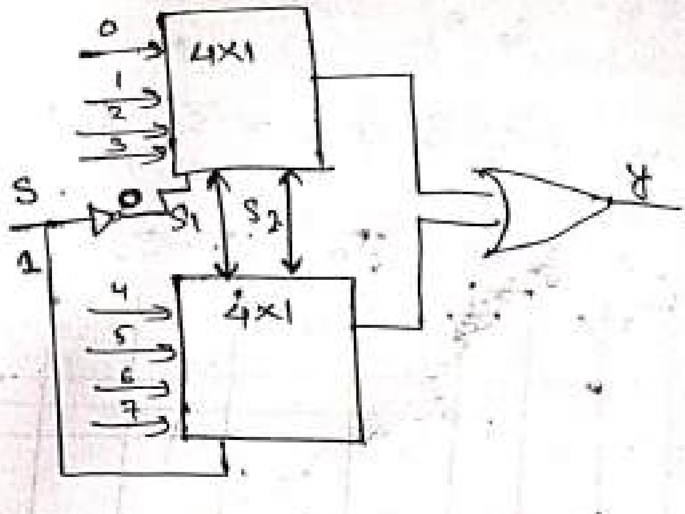
S0	S1	S2	f	logic.
0	0	0	I_0	1
0	0	1	I_1	0
0	1	0	I_2	0
0	1	1	I_3	1
1	0	0	I_4	0
1	0	1	I_5	1
1	1	0	I_6	1
1	1	1	I_7	1

Q) Design 8X1 MUX. using 4X1 MUX:
(U can use OR gate (or) 2X1 MUX).

8X1

S0	S1	S2	f
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

4 X1

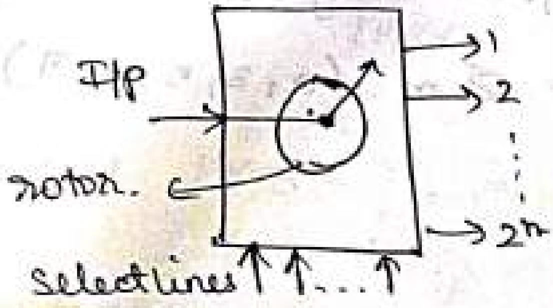


obtain expression for 'y'.

DE MULTIPLEXER (Dmux) (Data Distribution)

opposite of Multiplexer.

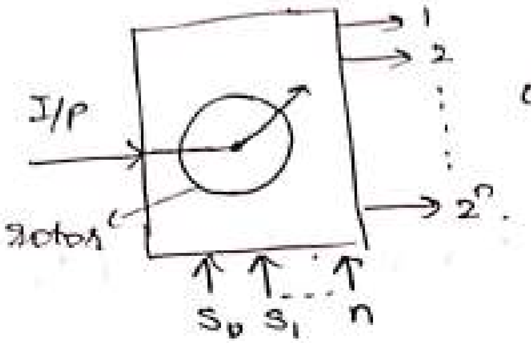
1 X 2^n 1 input 2^n outputs. n select lines



Contd in new book.

DE- MULTIPLEXER (Dmux) (Data Distribution)

* It is the opposite of a multiplexer. It has 1×2^n ie 1 input, 2^n outputs ; n select lines.

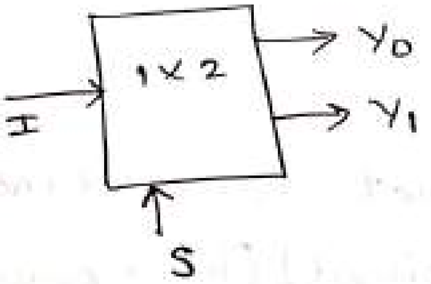


select lines

BLOCK DIAGRAM:

Ex: (a) 1 × 2 dmux:

$2^n = 2^1 \Rightarrow \boxed{n=1}$

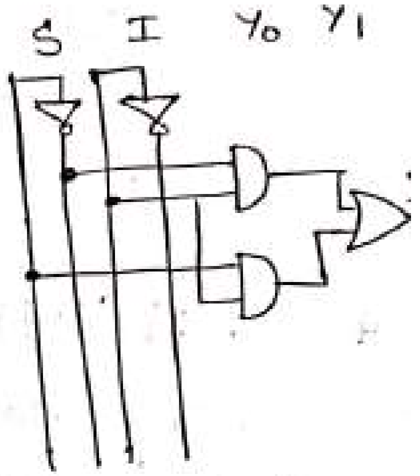


BLOCK DIAGRAM

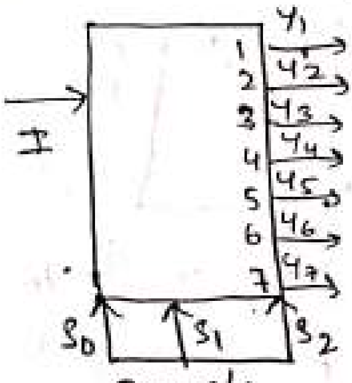
S	Y_0	Y_1
0	I	0
1	0	I

TRUTH TABLE

$Y = \bar{S}I + SI$



(b) 8 1 × 8 dmux $2^n = 2^3 \Rightarrow \boxed{n=3}$

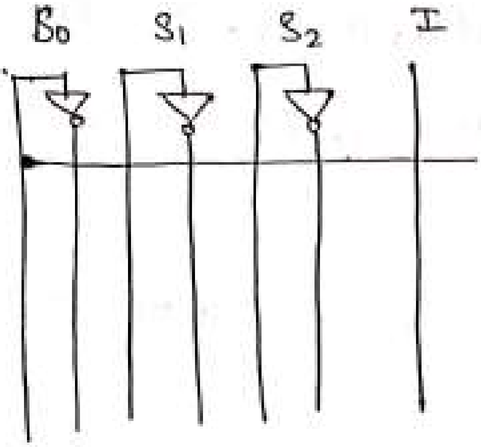


sample lines

BLOCK DIAGRAM.

S_0	S_1	S_2	Y_0	Y_1	Y_2	Y_3 ,	Y_4-	Y_5	Y_6	Y_7
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I.

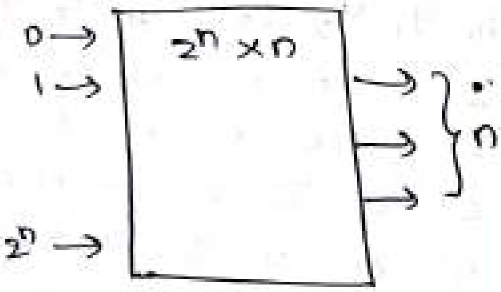
$Y_0 = \bar{S}_0 \bar{S}_1 \bar{S}_2 I + \bar{S}_0 \bar{S}_1 S_2 I + \bar{S}_0 S_1 \bar{S}_2 I + \bar{S}_0 S_1 S_2 I$,
$S_0 \bar{S}_1 \bar{S}_2 I + S_0 \bar{S}_1 S_2 I + S_0 S_1 \bar{S}_2 I + S_0 \bar{S}_1 S_2$



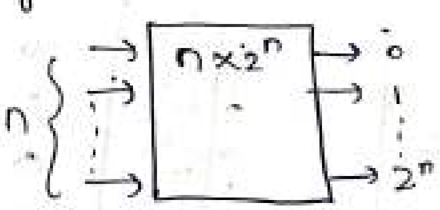
$Y = \bar{S}_0 \bar{S}_1 I[\bar{S}_2 + S_2] + \bar{S}_0 S_1 I[\bar{S}_2 + S_2] + S_0 \bar{S}_1 I[\bar{S}_2 + S_2]$
$+ S_0 S_1 I[\bar{S}_2 + S_2]$
$= \bar{S}_0 \bar{S}_1 I + \bar{S}_0 S_1 I + S_0 \bar{S}_1 I + S_0 S_1 I$
$= I \bar{S}_0 [\bar{S}_1 + S_1] + I S_0 [\bar{S}_1 + S_1]$
$= I \cdot [\bar{S}_0 + S]$
$Y = I$.

* ENCODER:

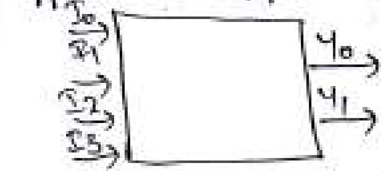
Coding data into other format is called encoder (Given the data is in decimal) (ie. decimal to binary (OR) any other like grey code etc).



Note: for DECODER:



4 to 2 logic. Encoder
O/P I/P lines.



I_0	I_1	I_2	I_3	Y_0	Y_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

y_0: $\bar{I_0}\,\bar{I_1}\,I_2\,\bar{I_3} + \bar{I_0}\,\bar{I_1}\,\bar{I_2}\,I_3$

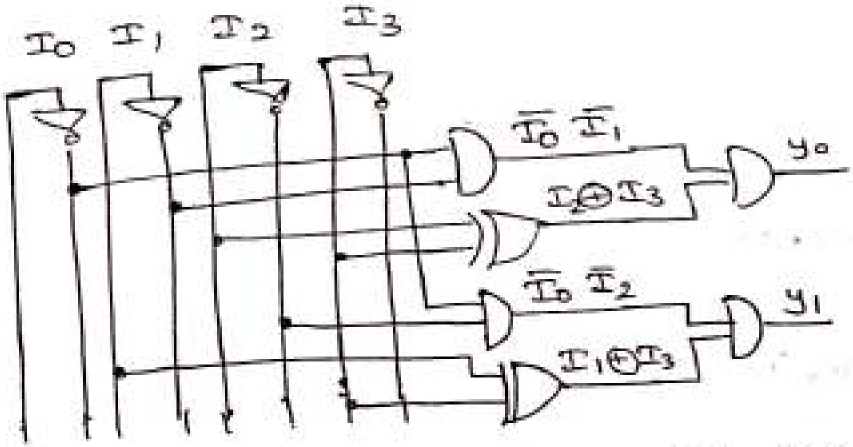
$y_0 = \bar{I_0}\,\bar{I_1}\,[I_2\bar{I_3} + \bar{I_2}I_3]$

$y_0 = \bar{I_0}\bar{I_1}\ [I_2 \oplus I_3]$

$y_1 = \bar{I_0}\,I_1\,\bar{I_2}\,\bar{I_3} + \bar{I_0}\,\bar{I_1}\,\bar{I_2}\,I_3$

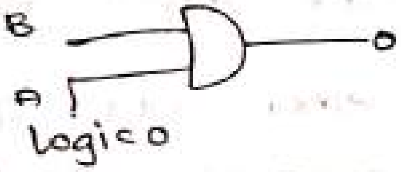
$= \bar{I_0}\,\bar{I_2}\,[I_1\bar{I_3} + \bar{I_1}I_3]$

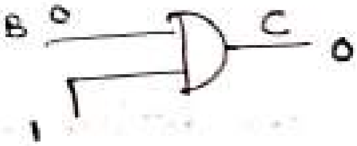
$= \bar{I_0}\bar{I_2}\ (I_1 \oplus I_3)$



DEGENERATE FORM OF GATES:

AND gate:





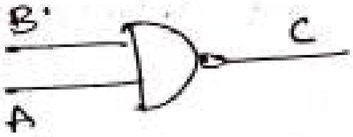
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

condition ① [When one of input is equal to logic 0 we get output as 0]

condition ② [When one of the input is logic 1 the output will be equal to other input 1]

∴ We can use AND gate as a "Buffer"

NAND gate:



A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

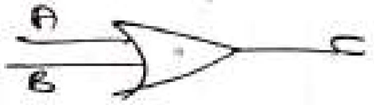
condition ①: When one of input is 0 we output as logic "1".

condition ②: When one of input is equal to logic 1 output is inverted.

∴ NAND gate can be used as "investor" where one of input is equal to logic 1.

OR gate :

A	B	C	
0	0	0	} Buffer
0	1	1	
1	0	1	} logic 1
1	1	1	

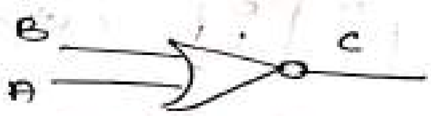


condition① : When one of input is 0 we get the input of other (ie. Buffer)

condition② : When one of the input is 1 the output is 1.

NOR gate :

A	B	C	
0	0	1	} inverter
0	1	0	
1	0	0	} logic 0
1	1.	0	



condition① When one of input is logic 0 we get output as inverter (hence we can use as inverter)

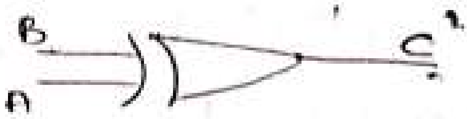
condition② When the input is equal 1, we get output as logic 0.

EX-OR gate (ANTI COINCEDENT GATE)

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

} Buffer

} invertor



condition ①: When one of input is 0, we get buffer EX-OR gate can be used as buffer

condition②: When one of input. is 1, we can use EX-OR gate is invertor.

EX-NOR gate (COINCEDENT GATE)

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

} invertor

} buffer

condition ①: when A=0, we can use EX-NOR as invertor.

condition ②: when A=1, we can use EX-NOR as buffer.

Q] Implement the following using decoder

$f_1(x,y,z) = x'y'z' + xz$,

$f_2(x,y,z) = xyz' + x'z$.

Sol. If given f^n is not in standard form then convert it into standard form (SOP

∴ $f_1(x,y,z) = x'y'z' + xz$

$= x'y'z' + xz(y+y')$

$= x'y'z' + xyz + xy'z$

$\quad 000 \quad 111 \quad 101$

$\quad \Downarrow \quad \Downarrow \quad \Downarrow$

$\quad 0 \quad 7 \quad 5$

$\Rightarrow f_1 = \Sigma m(0,7,5)$

$f_2(x,y,z) = xyz' + x'z$

$= xyz' + x'z(y+y')$

$= xyz' + x'yz + x'y'z$

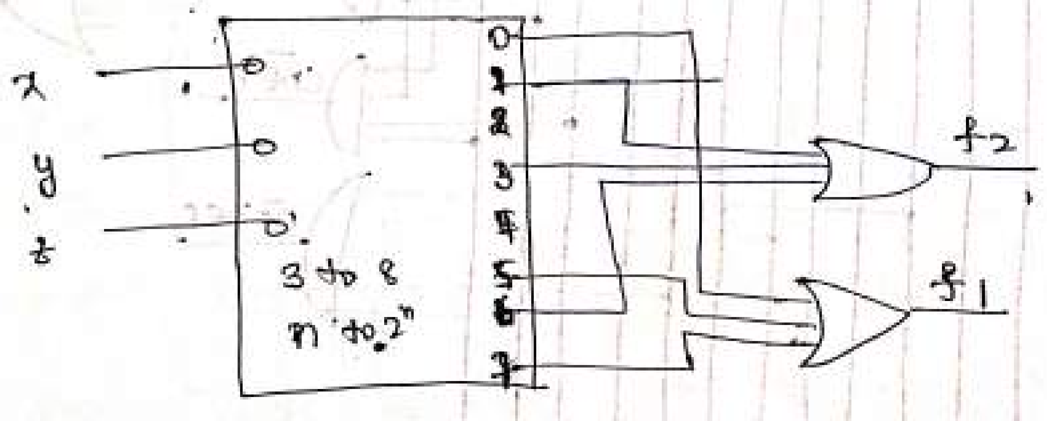
$\Rightarrow 110 \quad 011 \quad 001$

$\quad \Downarrow \quad \Downarrow \quad \Downarrow$

$\quad 6 \quad 3 \quad 1$

$f_2 = \Sigma m(6,3,1)$

Max number is 8 $2^3 \Rightarrow 2^n \Rightarrow n=$



Note: In case of decoder (or) encoder f(x,y, are input lines but in case of multiplexer

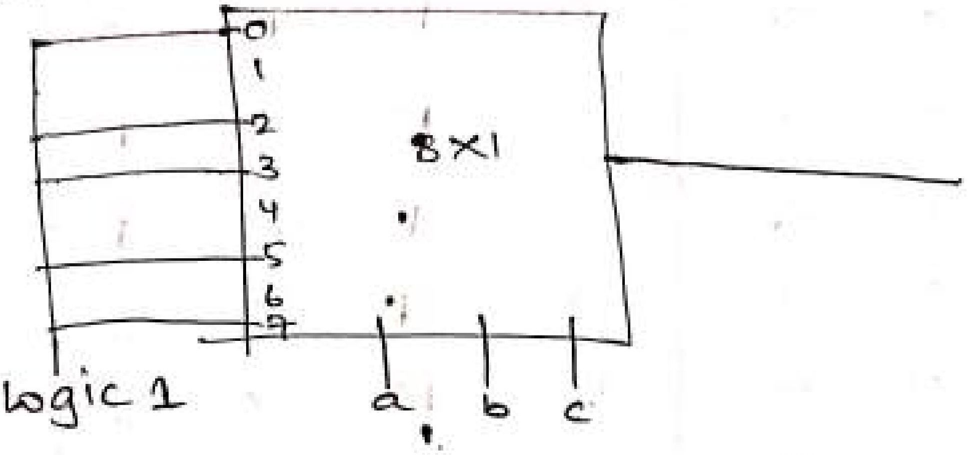
Demultiplexer. f (a,b,c) are select lines]

Q] Implement the following logic using multiplexer : [using 8×1 MUX]

f (a,b,c) = $\Sigma m\,(0,2,3,5,7)$

⇒ here we have 3 select lines

for mux we should have only 1. $2^n = 2^3 = 8$



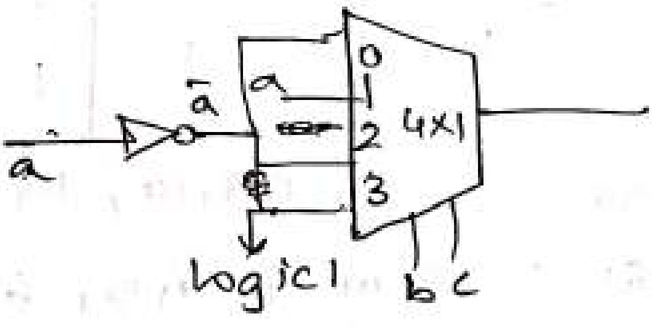
Using only single 4×1 MUX.

f (a,b,c) = $\Sigma m\,(0,2,3,5,7)$

	D0	D1	D2	D3
A = 0	^{0} 1	^{1} 0	^{3} 1	^{2} 1
A = 1	^{4} 0	^{5} 1	^{7} 0	^{6} 1
	D4	D5	D6	D7

→ [$A + \bar{A} = 1$]

$\bar{A}$	A	$\bar{A}$	1
0	1	2	3

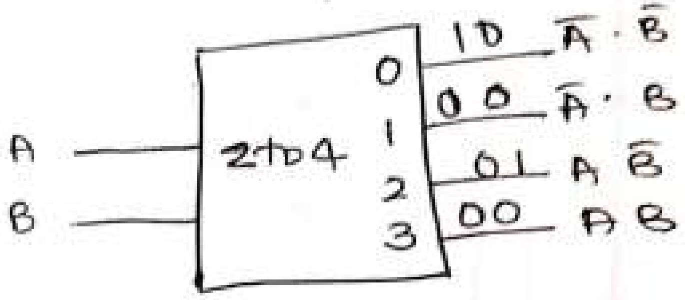


* DECODERS:

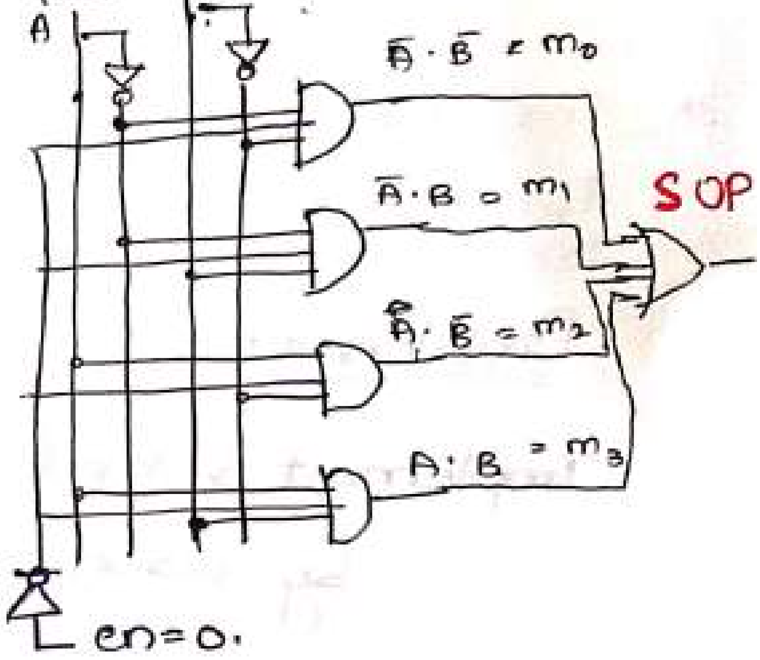
* Decoder o/p are min terms of the i/p's.

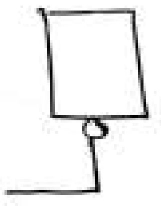
2 × 4 DECODER:

* For decoder for n i/p's we have 2^n o/p's.



* Gate B circuit for 2×4:

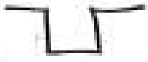


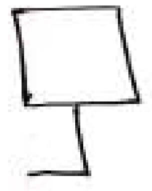


↳ en=0

↳ active low

↳ -ve logic

↳ 



↳ en=1

↳ active high

↳ +ve logic

↳