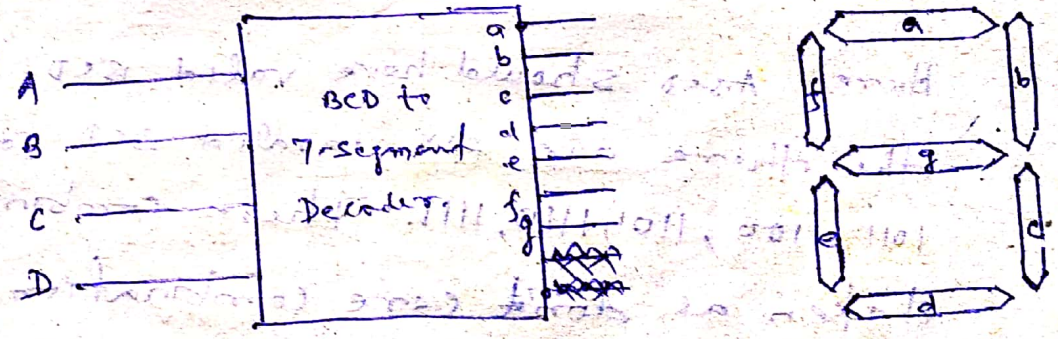
BCD to Seven Segment Decoder

BCD to seven segment decoder is a combinational ckt that has 4-input lines and accept BCD code and display corresponding decimal number on seven-segment display. The display screen consist of seven segment and each segment made of the material which can illuminate or emits light when current pass through it. Usually, these segments are LED and emits light when it is forward bias. The fig. as below indicates a basic block diagram for BCD-to-7-segment decoder.



From above fig. it is clear that a, b, f, d, f, g represents 7-segment of display in clockwise direction. If we want to display 2, corresponding to BCD – 0010, then the segments a, g, e, d should have to be illuminated. If we want to display '0' corresponding to BCD 0000, then segments a, b, c, d, e, f should be illuminated and g should be dead.

This can be understood in betterway by following truth table of BCD-to-seven segment decoder.

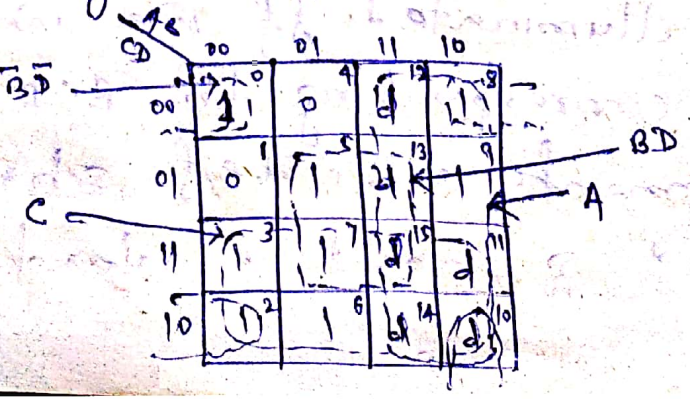
Inputs				Output of 7-segment display						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Here, A B C D should have valid BCD but we know that there are six invalid BCD codes as 1010, 1011, 1100, 1101, 1110, 1111. These combinations are taken as don't care combination when we derive the expression for a, b, c, d, e, f, g.

Boolean Expression for 'a'

$$a = \sum_m (0,2,3,5,6,7,8,9) + \sum_d (10,11,12,13,14,15)$$

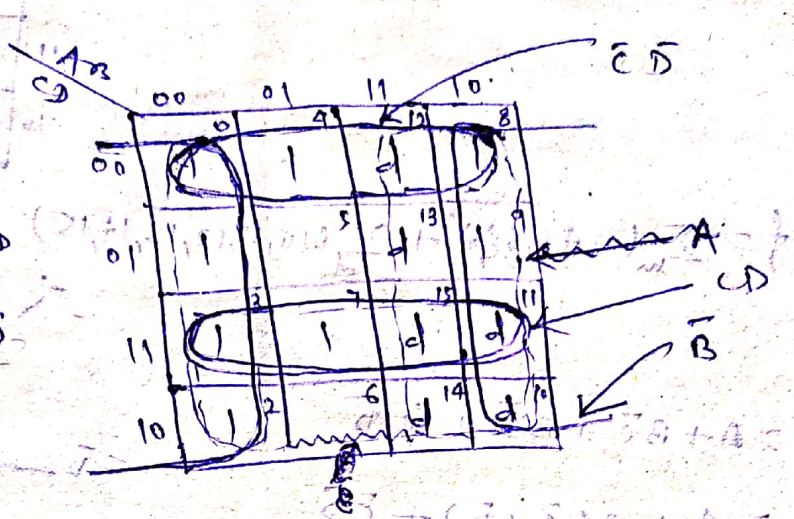
Simplifying by using k-map as —



$a = A + C + B\bar{D} + \bar{B}\bar{D} = A + C + \overline{B \oplus D}$

Boolean Expression for 'b'

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

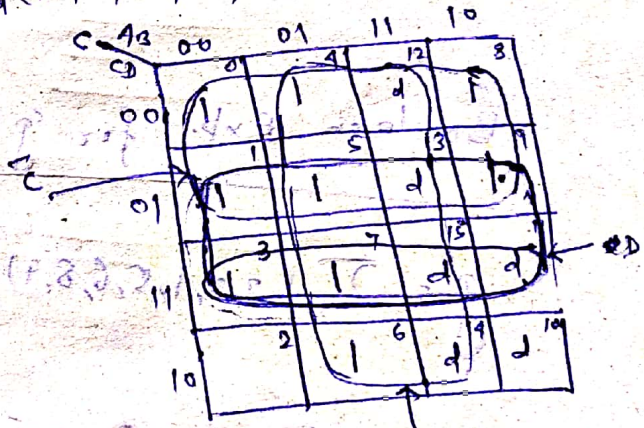


$b = \bar{B} + \bar{B}\bar{D} + B\bar{D}$

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

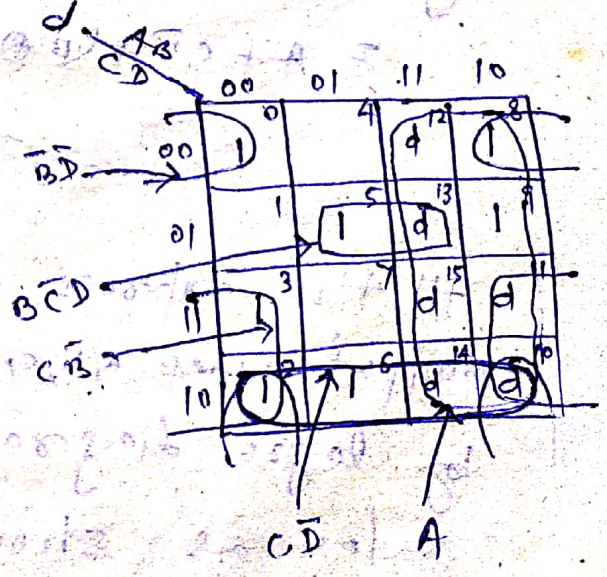
Boolean Expression for 'c'.

$c = \sum_m(0,1,3,4,5,6,7,8,9) + \sum_d(10,11,12,13,14,15)$



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

$d = \sum_m(0,2,3,5,6,8,9) + \sum_d(10,11,12,13,14,15)$



$\therefore d = A + \bar{B}C + C\bar{D} + B\bar{C}D + \bar{B}\bar{D}$

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

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

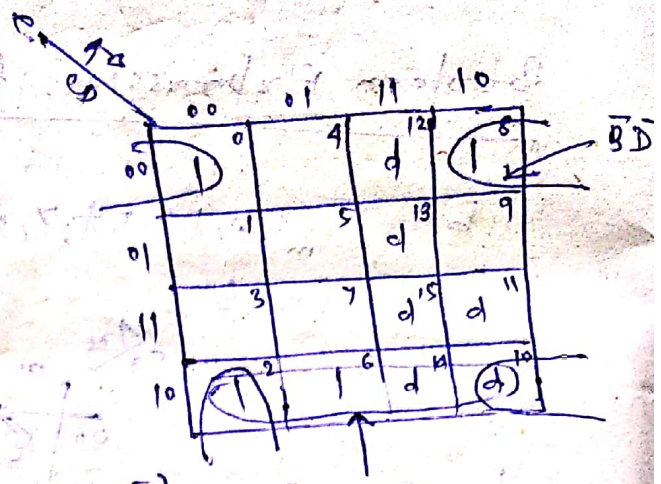
$= A + \bar{B}(C + \bar{D}) + B\bar{C}D + C\bar{D}$

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

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

$e = \sum_m(0,2,6,8) + \sum_d(10,11,12,13,14,15)$

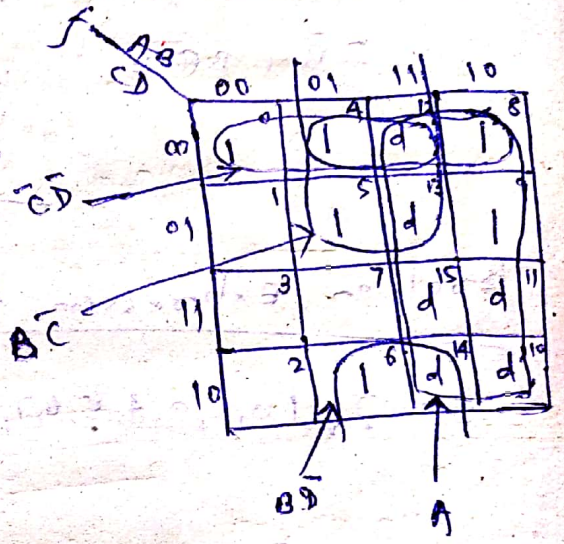
$e = \bar{B}\bar{D} + C\bar{D} = \bar{D}(C+\bar{B})$



$f = \sum_m(0,4,5,6,8,9) + \sum_d(10,11,12,13,14,15)$

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

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

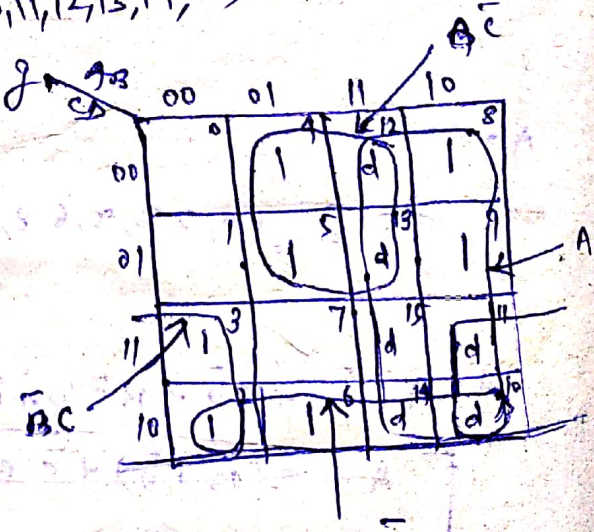


Boolean Exp. for 'g'

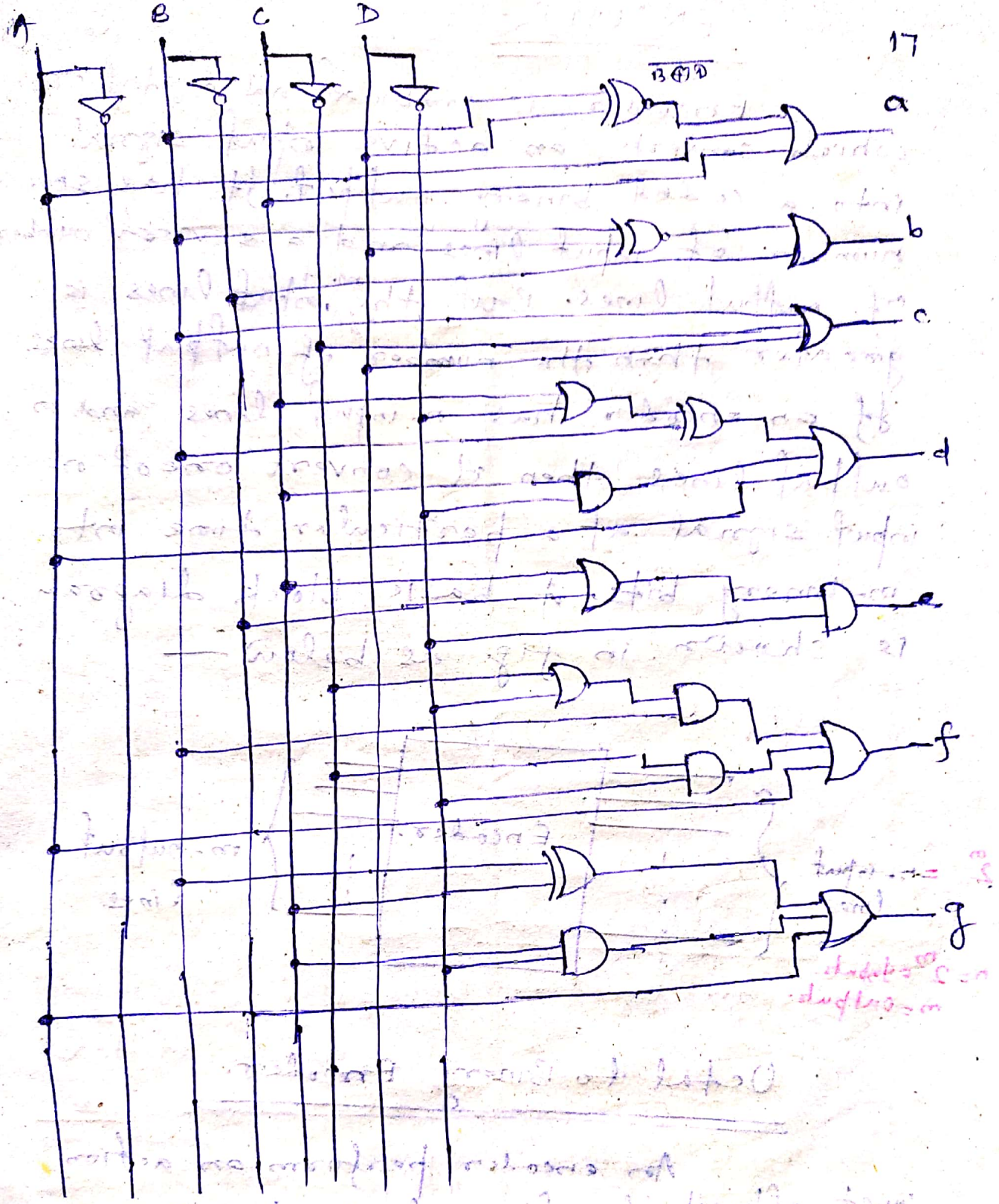
$g = \sum_m(2,3,4,5,6,8,9) + \sum_d(10,11,12,13,14,15)$

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

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



All the expressions of output lines a, b, c, d, e, f and g can be implemented by logic diagrams or logic ckt using logic gates as shown in fig-A.



Logic ckt of 7-segment BCD decoder.

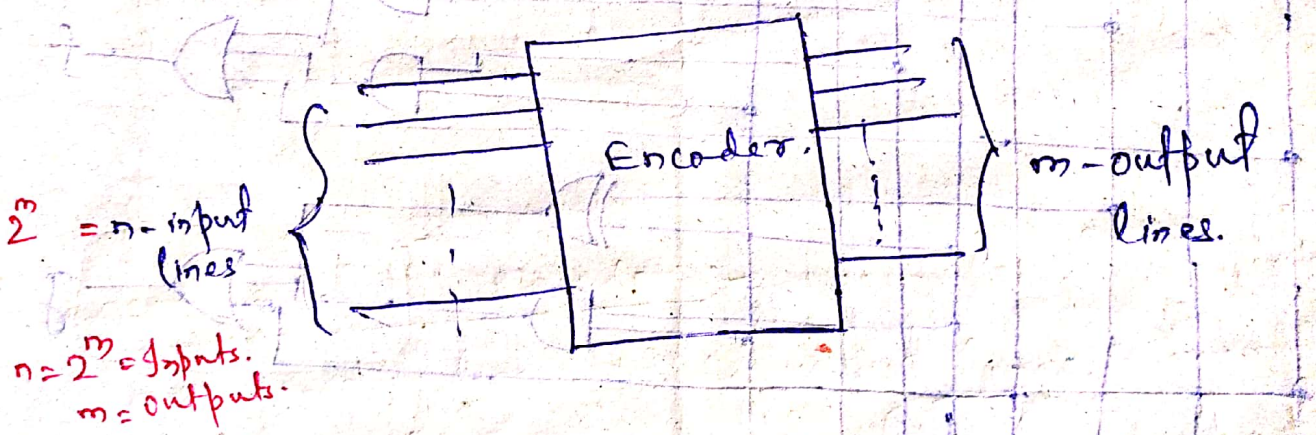
fig-1.

Application of Decoder —

1) Used in counter
2) Used in A/D converter
3) Used to drive display.

ENCODER

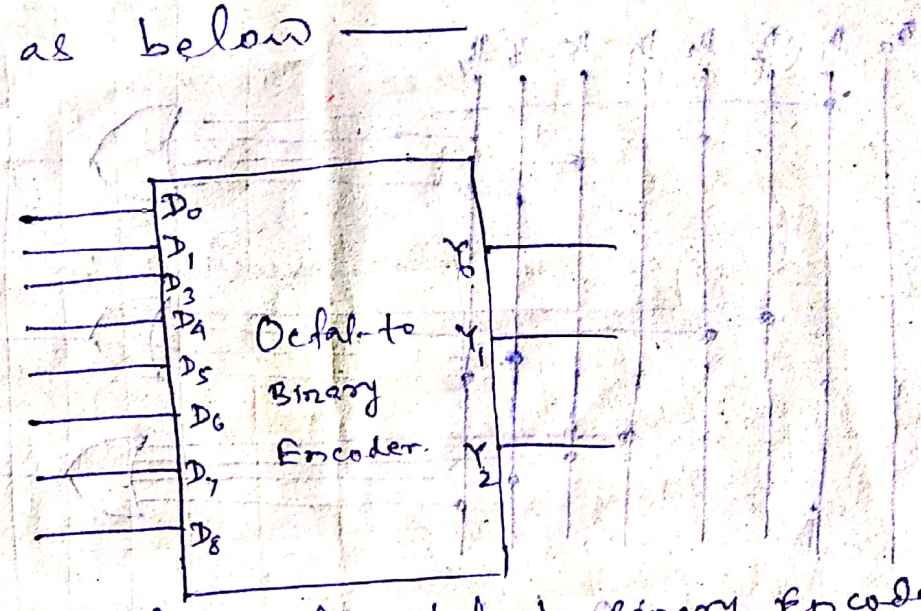
Encoder is combinational digital ckt which converts an active input signal into a coded binary output. It has several number of input lines and a several number of output lines. But, the no. of input lines is greater than the number of output lines. If an encoder has n-input lines and m output lines then it convert one of n-input signal at a particular time into m-binary bits. A basic block diagram is shown in fig. as below —



Octal to Binary Encoder.

An encoder performs an action just opposit function of decoder. For example, BCD decoder which is also ~~known~~ known as 3-to-8 decoder accepts a 3-bit binary code and activate only one of the eight output line. Here, Octal-to-Binary Encoder functions opposit to the func

of 3-to-8 decoder. Octal-to-Binary encoder has 8-input lines and three output lines and at a time only one input is converted or coded into 3-binary bit. A block diagram is shown in figure as below ——



The function of octal-to-Binary Encoder can be understood in betterway by the truth table which is as below ——

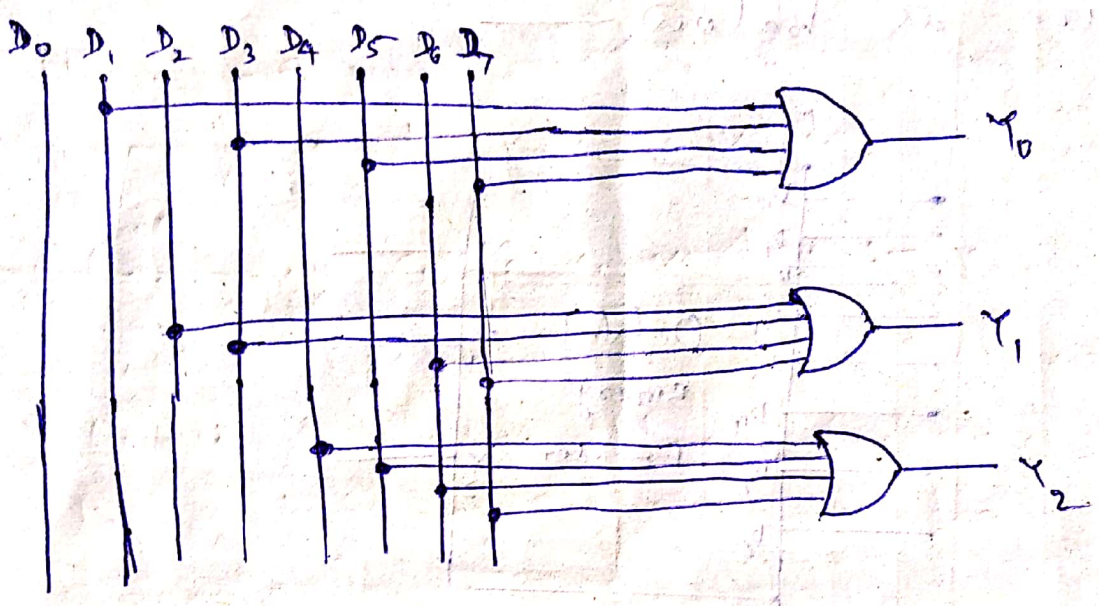
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

from above truth table, we can write boolean expression as below ——

$$Y_0 = D_1 + D_3 + D_5 + D_7 \quad , \quad Y_1 = D_2 + D_3 + D_6 + D_7$$

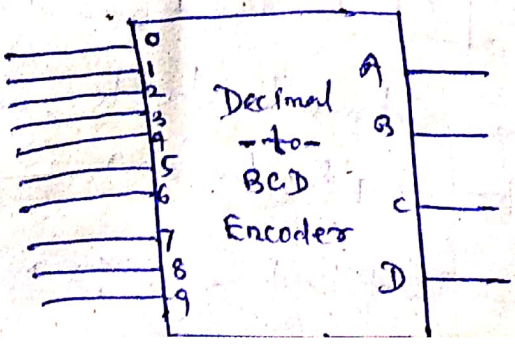
$$Y_2 = D_4 + D_5 + D_6 + D_7$$

By using these boolean expressions, Octal-to-binary encoder can be implemented by using three OR gates of 4-input lines. This logic ckt is shown in fig as below——



Octal-to-Binary Encoder

Decimal-to-BCD Encoder :- Decimal-to-BCD encoder has 10-input lines and 4-output lines. Decimal to BCD-encoder is a combinational ckt that converts any one of ~~an~~ active input of ~~the~~ ten input into a 4-bit, binary code. The fig shown in below represents a block diagram——



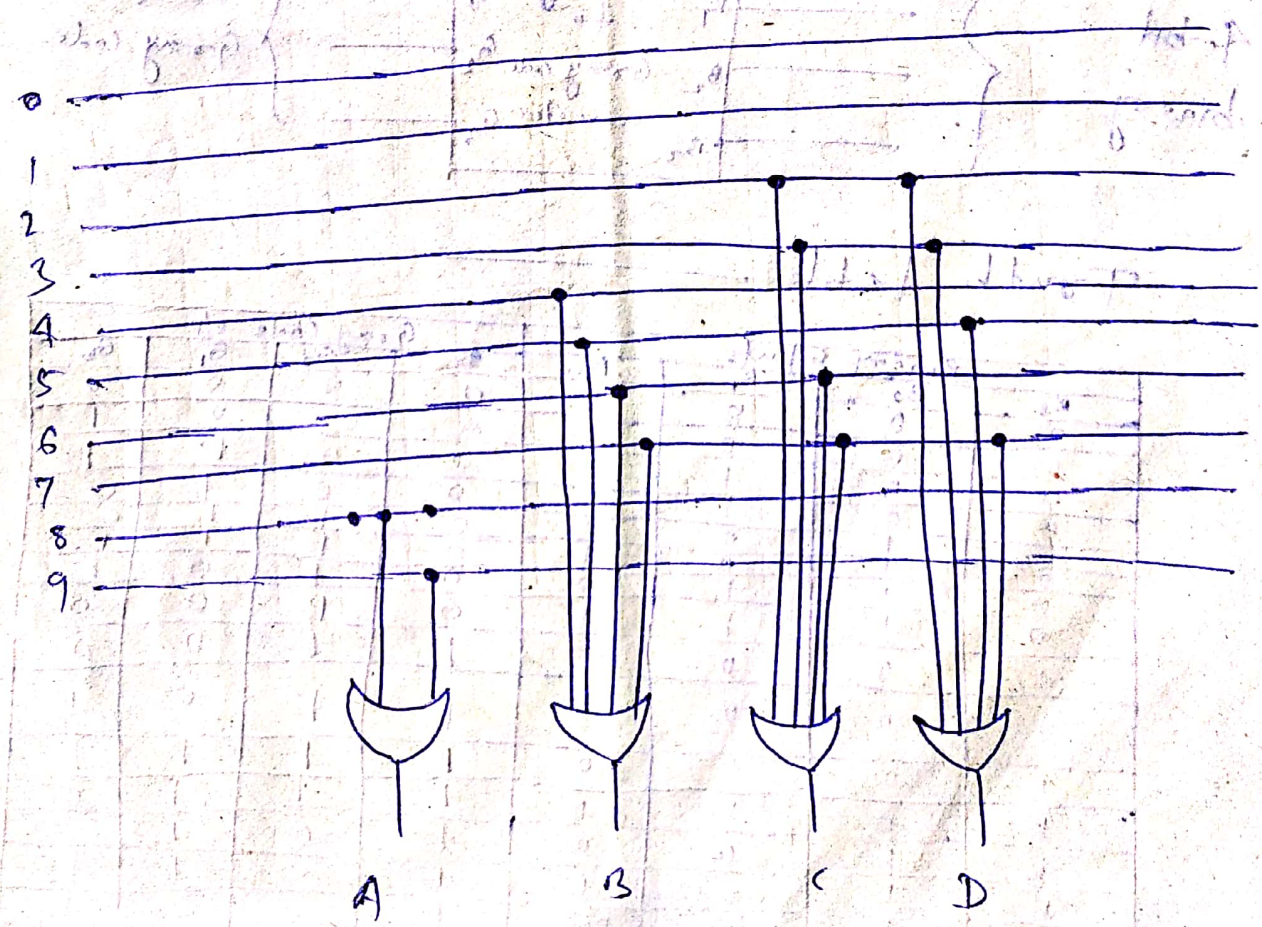
Truth table for decimal to BCD-Encoder.

Decimal input										BCD output			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

from above table —

$D = 1+3+5+7$, $C = 2+3+6+7$, $B = 4+5+6+7$, $A = 8+9$

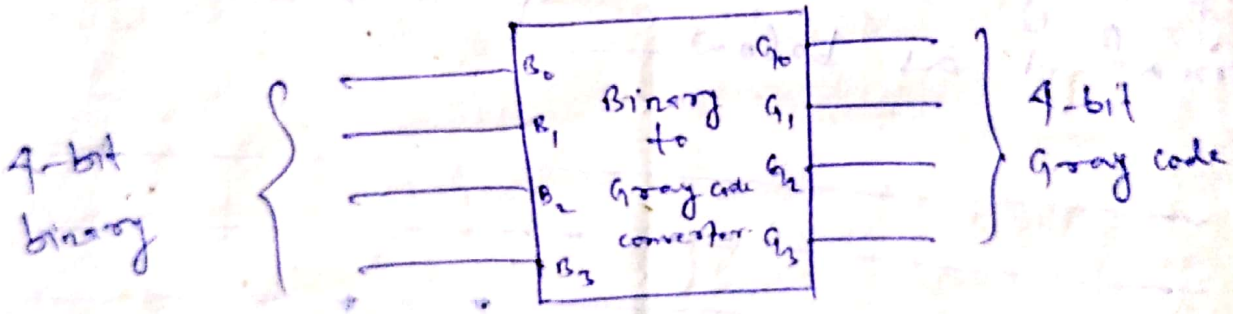
By using above boolean expression, we can implement it as below —



Code Converters

Code converters are the combinational type ckt that convert one binary code into other type of binary code. BCD-to-seven segment decoder is also a code converter. Here we are discussing some most commonly used code converters as ——

Binary-to-Gray code converter :- Binary to gray converter has equal no. of input and output lines. A basic block diagram of 4-bit binary-to-Gray code converter is shown in fig. 1.



Truth table ——

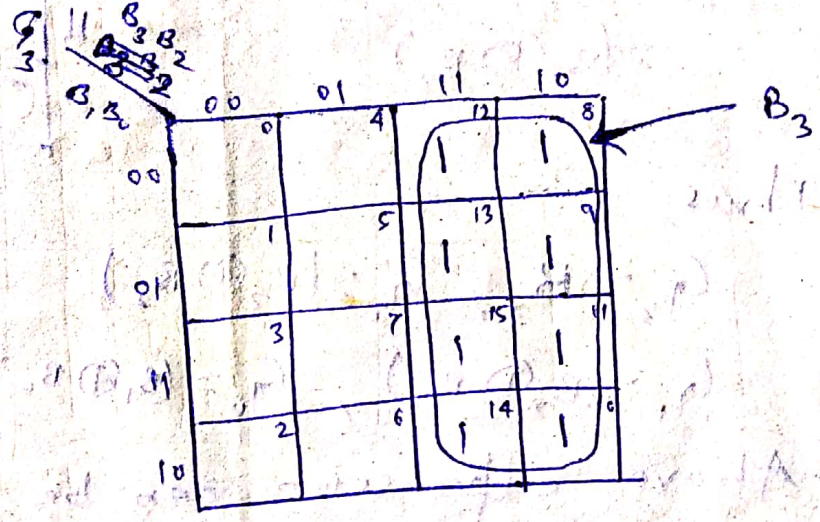
Binary inputs				Gray code o/p			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

from truth table ——

$G_3 = \sum_m(8,9,10,11,12,13,14,15)$, $G_2 = \sum_m(4,5,6,7,8,9,10,11)$,

$G_1 = \sum_m(2,3,4,5,10,11,12,13)$, $G_0 = \sum_m(1,2,5,6,9,10,13,14)$

K-map simplification for 'G_3'

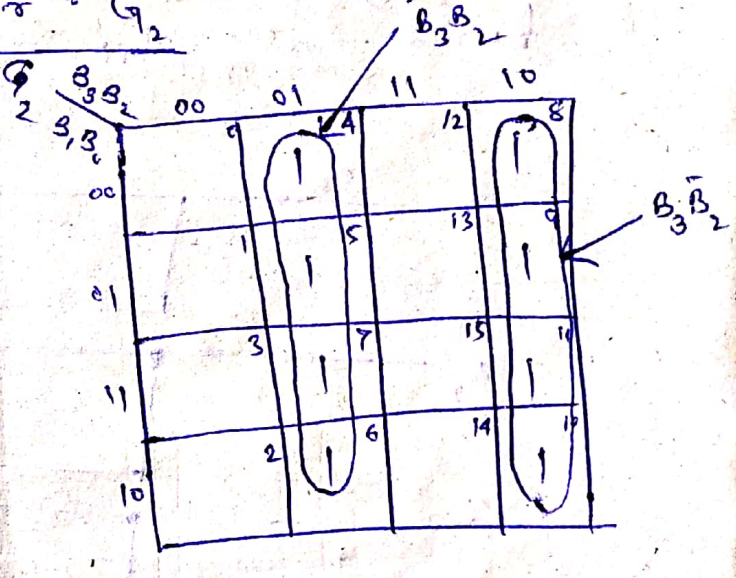


$\therefore G_3 = B_3$

K-map simplification for 'G_2'

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

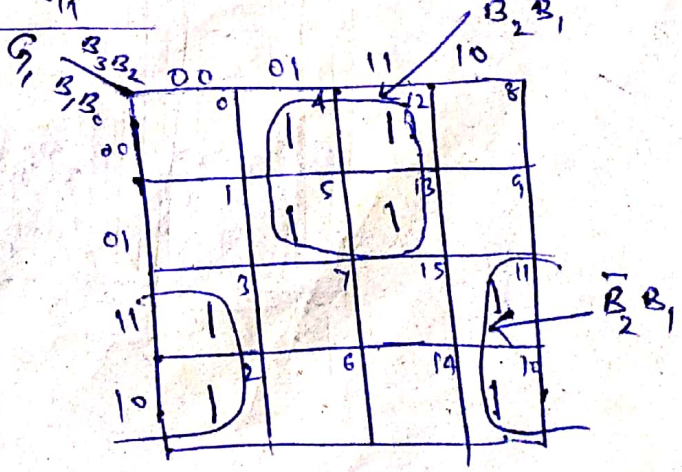
$= (B_3 \oplus B_2)$



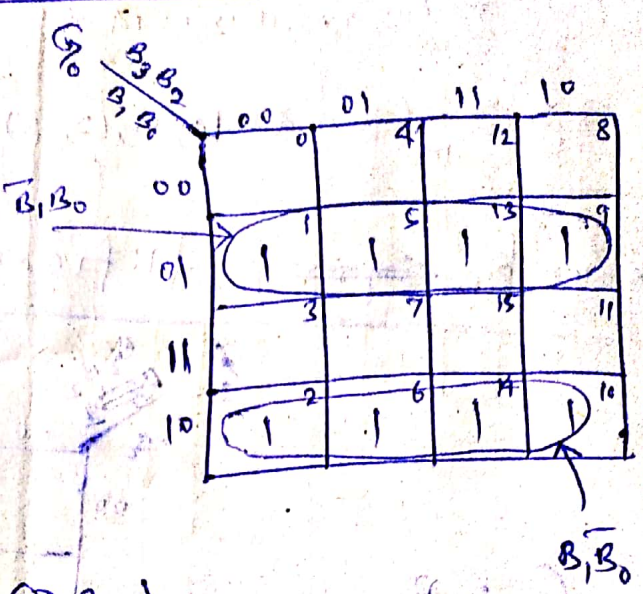
K-map simplification for 'G_1'

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

$= (B_2 \oplus B_1)$.



K-map simplification for G_0



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

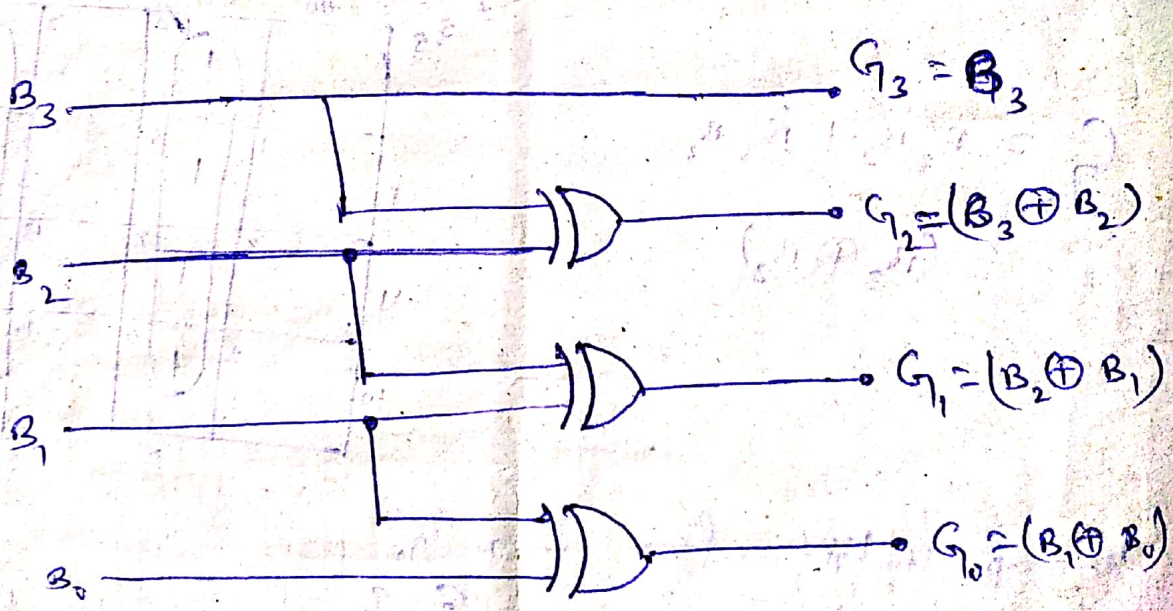
$= (B_1 \oplus B_0)$

Thus,

$G_3 = B_3$, $G_2 = (B_3 \oplus B_2)$

$G_1 = (B_2 \oplus B_1)$, $G_0 = (B_1 \oplus B_0)$.

Above expression can be implemented by using three X-OR gates with two-input lines as shown below —



Binary ($B_3 B_2 B_1 B_0$) to Gray code ($G_3 G_2 G_1 G_0$) Converter

Gray to Binary Converter:

Gray to binary converter is combinational ckt that convert the Gray code into binary number. A 4-bit Gray to binary converter block diagram is shown in fig-1.

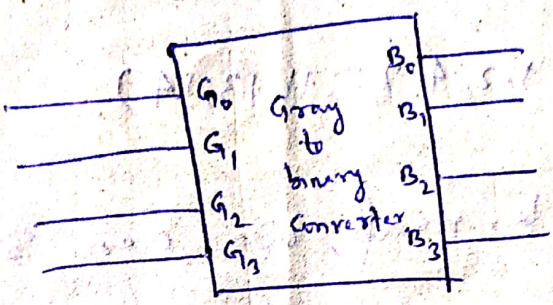


fig-1.

truth table ——

Gray code				Binary output			
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	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

from truth table —

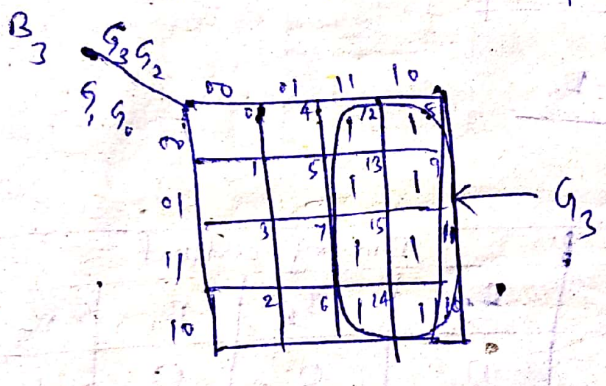
$B_3 = \sum_m (8, 9, 10, 11, 12, 13, 14, 15)$

$B_2 = \sum_m (4, 5, 6, 7, 8, 9, 10, 11)$

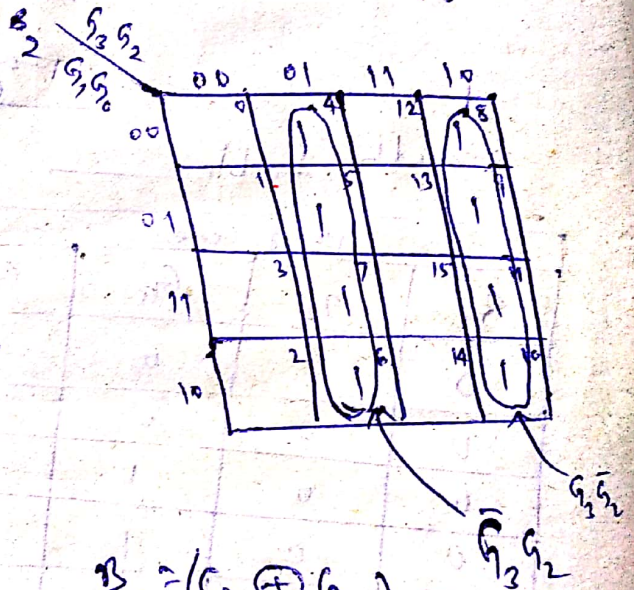
$B_1 = \sum_m (2, 3, 4, 5, 8, 9, 14, 15)$

$B_0 = \sum_m (1, 2, 4, 7, 8, 11, 13, 14)$.

By using k-map, we can simplify —

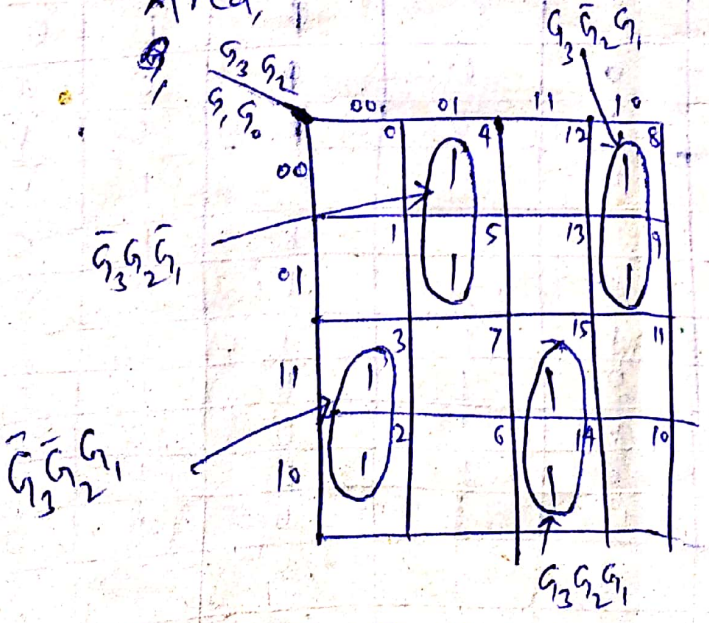


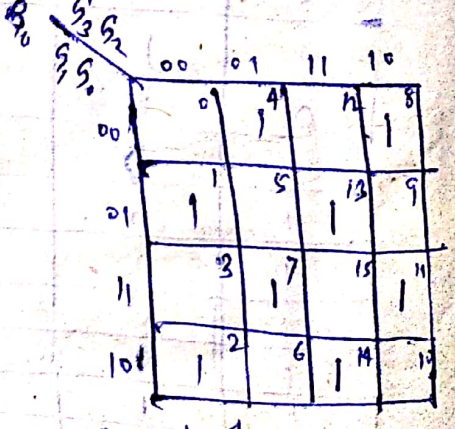
$B_3 = G_3$



$B_2 = (G_3 \oplus G_2)$

And,





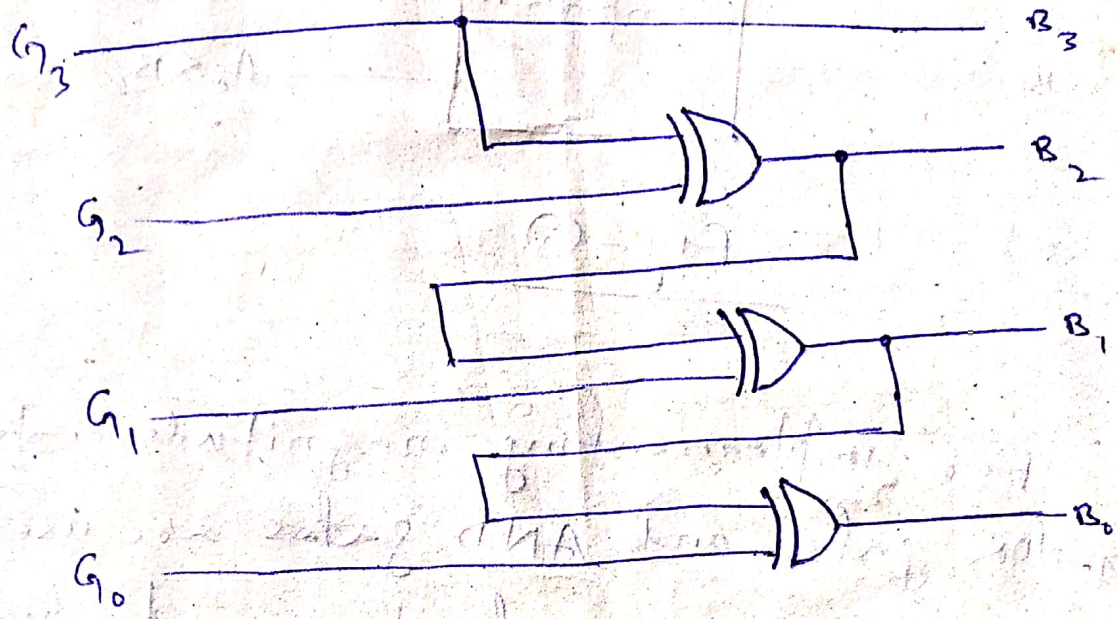
Similarly —

$B_0 = B_1 \oplus G_0$

$B_1 = \bar{G}_3\bar{G}_2G_1 + \bar{G}_3G_2\bar{G}_1 + G_3\bar{G}_2\bar{G}_1 + G_3G_2G_1$

$= \bar{G}_2(G_2 \oplus G_1) + G_3\overline{(G_2 \oplus G_1)} = \bar{G}_2 \oplus G_2 \oplus G_1 = B_2 \oplus G_1$

the boolean expression for B_3, B_2, B_1 and B_0 27 can be implemented by using x-or gates as below ——



"Gray to binary converter"

Magnitude Comparator :- A comparator is a combinational ckt that compares the two numbers A and B and give the output as whether ~~it is~~ they are equal, greater, then, or smaller then. i.e A=B, A>B or A<B. It is obvious that, it has two inputs but three outputs. A basic block diagram for single bit Comparator is shown in fig-(a)

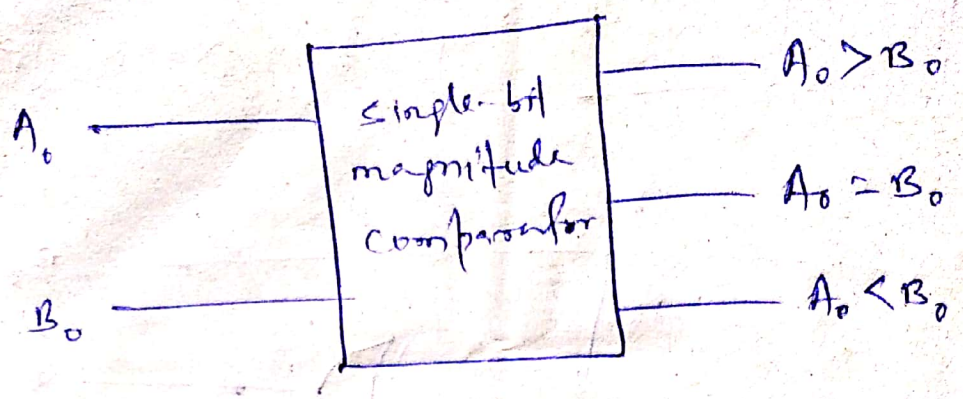


Fig-(a)

For implementing magnitude comparator, Ex-NOR gates and AND gates are used because Ex-NOR gate has a property that when both the input of Ex-NOR are equal then it produce always HIGH. It is shown in fig-(b).

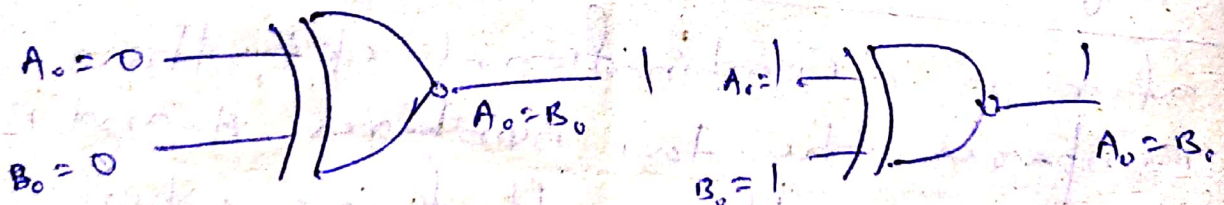


Fig-(b).

By using AND gate we can compare A_0 and B_0 whether it is $A_0 > B_0$ or $A_0 < B_0$.

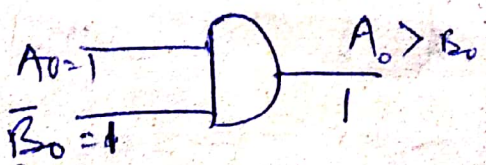


Fig-(i)

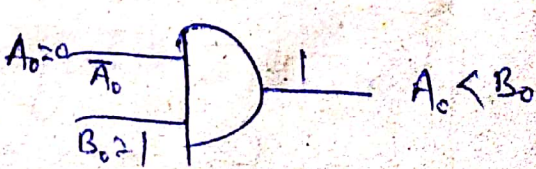


Fig-(ii)

Fig-① represent an AND gate with input line A_0 and $\bar{B}_0$. This AND gate produces a 'HIGH' or 1 if $A_0=1, B_0=0$ which means that $A_0 > B_0$.

Again, fig-(ii) represents AND gate with input line $\bar{A}_0$ and B_0. This produce a HIGH (1) when $A_0=0$ and $B_0=1$. i.e $A_0 < B_0$.

Thus, complete implementation of a single bit comparator is as below