Unit - 4
Combinational Circuits

Combinational logic circuit design, adder, subtractor, BCD adder, encoder, decoder, BCD to 7-segment decoder, multiplexer, demultiplexer.

- Digital computers and calculators consist of arithmetic and logic circuits, which contain logic gates and flip-flops that add, subtract, multiply and divide binary numbers.
- The basic building blocks of the arithmetic unit in a digital computer are adders.
- These circuits perform operations at speed less than 1 μs.
- A digital systems consists of two types of circuits, namely

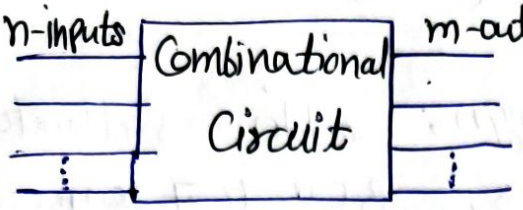
(i) **Combinational Logic Circuit**

- The output at any time depends only on the present value at that time.
- Examples:- - Half adder,
  - Full adder,
  - DeMultiplexer, Multiplexer.
  - Encoder, - Decoder, parallel adder,
- Memory element is absent.

(ii) **Sequential Logic Circuit**

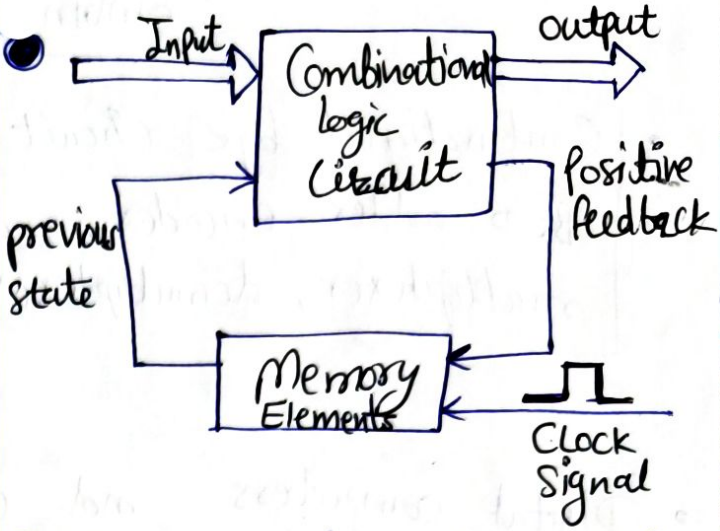
- The output at any time depends on the present values of input as well as the past output values.
- Examples:- - Flip-flops
  - Counters
  - Registers
  - Serial adder.
- Memory element is present to store the past history of i/p variables.

Combinational Circuits



- They are faster in speed because the delay between input & output is due to propagation delay of times. (all inputs are applied at the same time.)
- Easy to design & implement with the help of basic logic gates.

Sequential Circuits



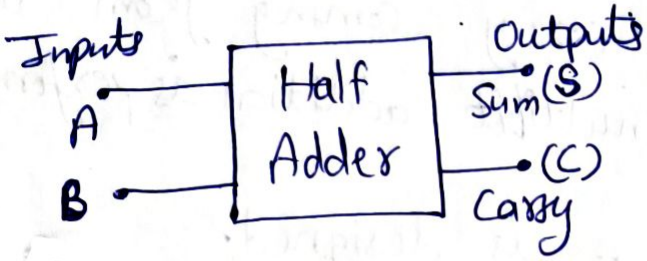
- They are slower, because of the secondary inputs. So, there is a delay in between inputs.
- Comparatively harder to design.

---

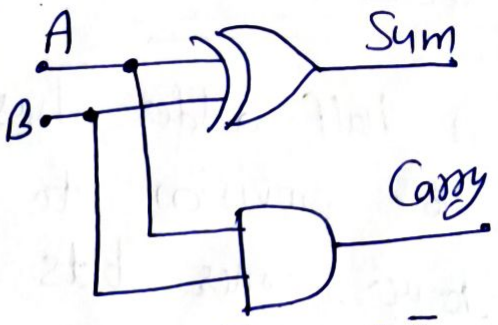
[1] **Half Adder** :- The Combinational circuit which performs the arithmetic addition of two binary digits is called a halfadder.

- when either of the inputs (A or B) is 1, the sum output is 1. & Carry output is one when both inputs (A & B) are 1.

(1) Logic Symbol



(2) Logic diagram



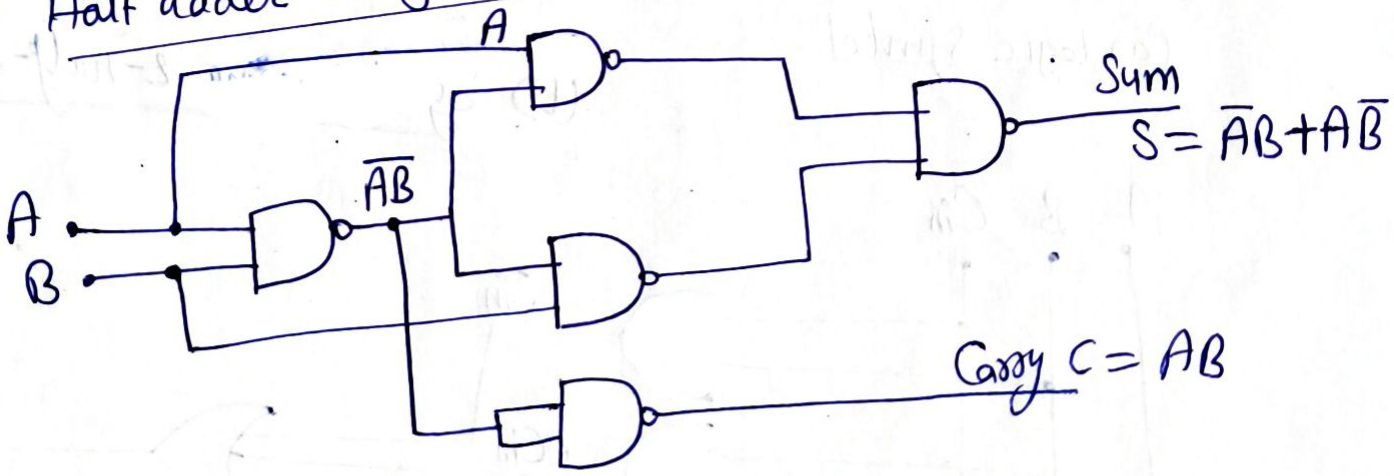
$S = \bar{A}B + A\bar{B}$

$C = AB$

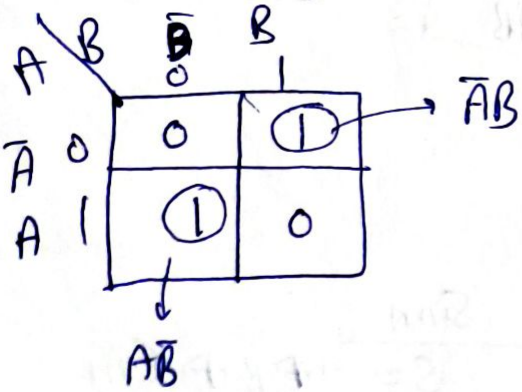
(3) Truth table of half-adder:-

Input		Output	
Augend A	Addend B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(4) Half adder using NAND Gate:-

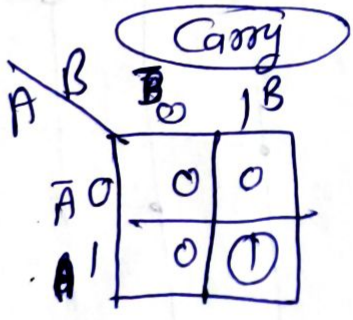


Sum



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

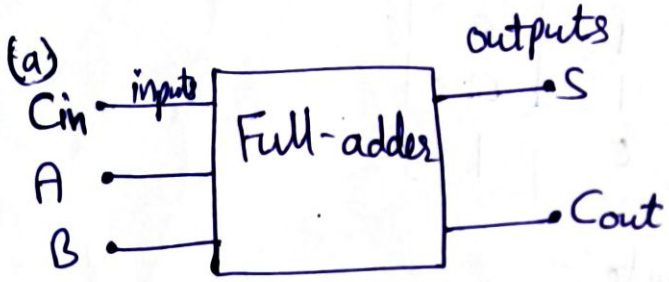
Carry



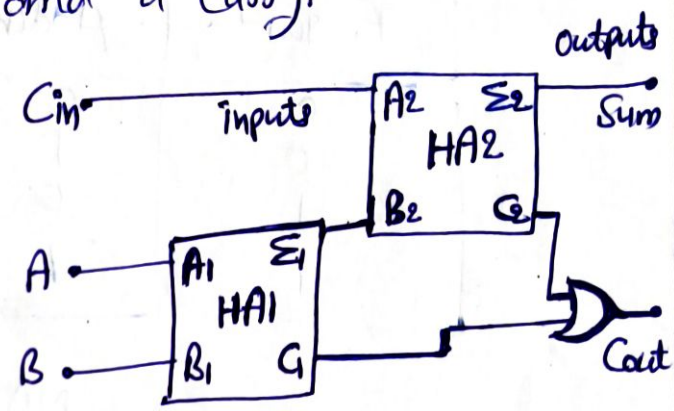
$Y = AB$

FULL Adder:-

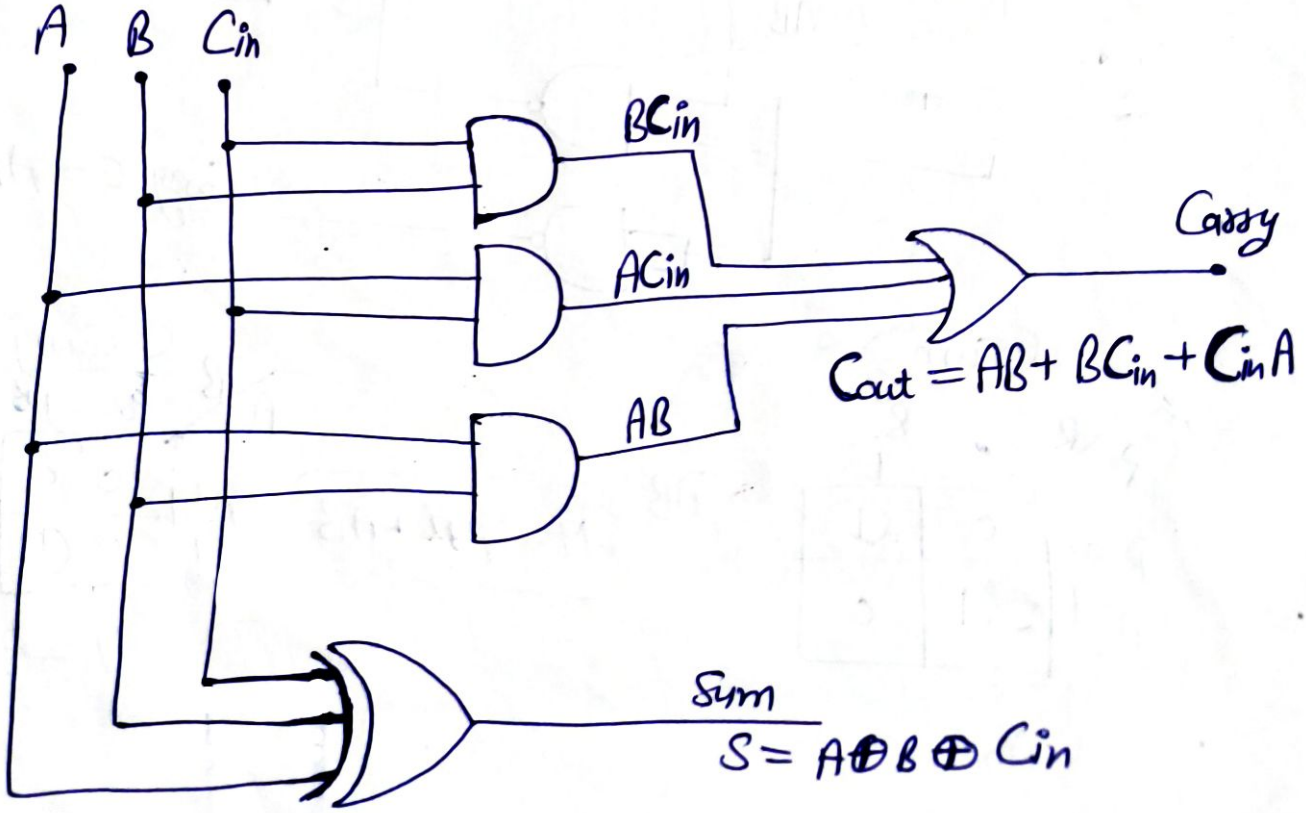
- A half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multibit addition is performed.
- For this purpose, a full-adder is designed.
- A full-adder is a combinational circuit that performs the arithmatic sum of three input bits and produces a sum output and a carry.



(a) Logic Symbol



(b) Symbol using 2-half-adders



(c) Logic diagram of full adder

Truth table of full adder

Inputs			outputs	
Augend bits A	Addend bit B	Carry input C_{in}	Sum S	Carry output C_{out}
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

- From truth table, the logic expression for S can be written by summing up the input combinations for which the sum output is 1 as:

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

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

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

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

Now, $S = \bar{A}X + A\bar{X} = A \oplus X$

Replacing X by $B \oplus C_{in}$ in above expression, then

Sum $\quad S = A \oplus B \oplus C_{in}$.

- Similarly, for Cout logic expression, summing up the input combinations for which Cout is 1.

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

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

$$C_{out} = BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in}$$

Now, ABC_{in} term is added twice for simplification

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

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

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

- From simplified expression of S and C_{out}, the full adder circuit can be implemented using one 3-input EX-OR gate, three 2-input AND gates and one 3-input OR gates.

K-Map Simplification:- It is used to simplify the logic expression for S & C_{out}.

Sum

A \ BC_{in}	00 ($\bar{B}\bar{C}_{in}$)	01 ($\bar{B}C_{in}$)	11 (BC_{in})	10 ($B\bar{C}_{in}$)
$\bar{A}$ 0	0	(1)	0	(1)
A 1	(1)	0	(1)	0

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

Carry

A \ BC_{in}	00 ($\bar{B}\bar{C}_{in}$)	01 ($\bar{B}C_{in}$)	11 (BC_{in})	10 ($B\bar{C}_{in}$)
$\bar{A}$ 0	0	0	(1)	0
A 1	0	(1)	(1)	(1)

$$Carry = BC_{in} + AC_{in} + AB$$

- Using the above expression, full adder can be implemented using basic logic gates →

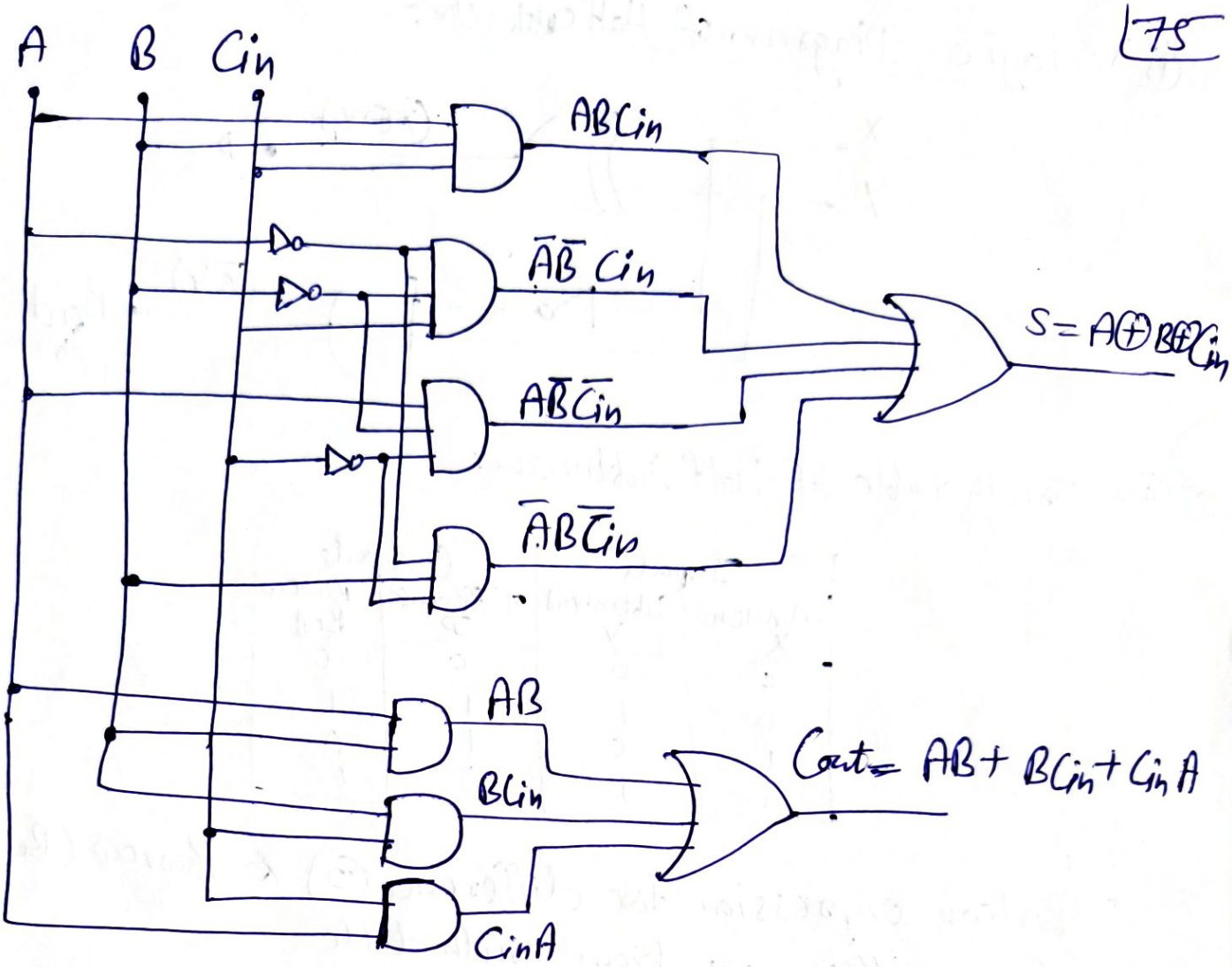
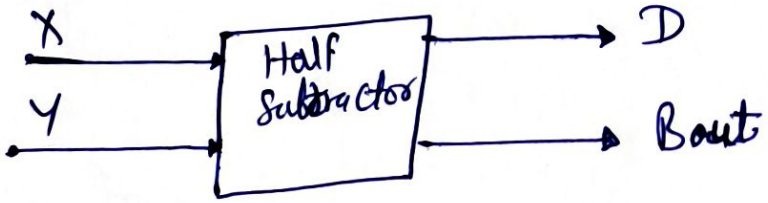


Fig:- Full adder using basic logic gates.

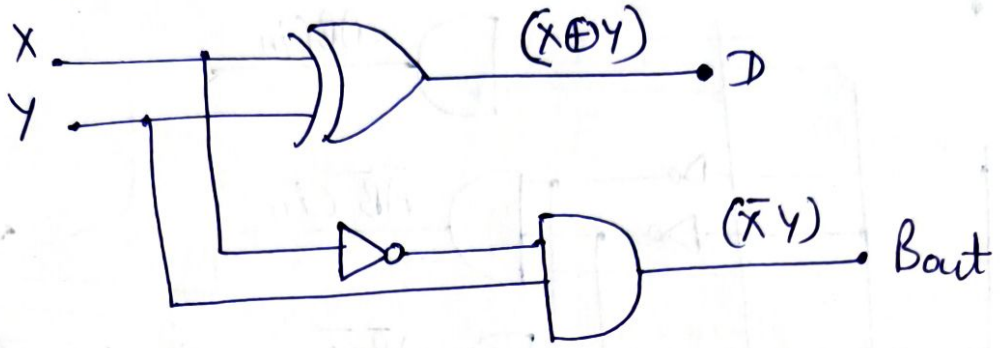
Half-Subtractor :-

- H.S. is a combinational circuit which is used to perform subtraction of two bits.
- It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and Bout (Borrow out).

- (i) Logic Symbol :-



(ii) Logic Diagram of Half Subtractor:-



(iii) Truth table of half subtractor:-

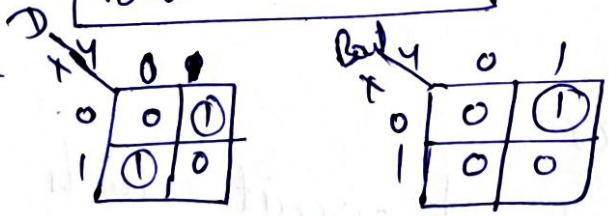
Inputs		Outputs	
Minuend X	Subtrahend Y	Difference D	Borrow Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- Boolean expression for difference (D) & Borrow (Bout) can be written as; from Truth table

$$D = \bar{X}Y + X\bar{Y} \Rightarrow X \oplus Y$$

$$Bout = \bar{X}Y$$

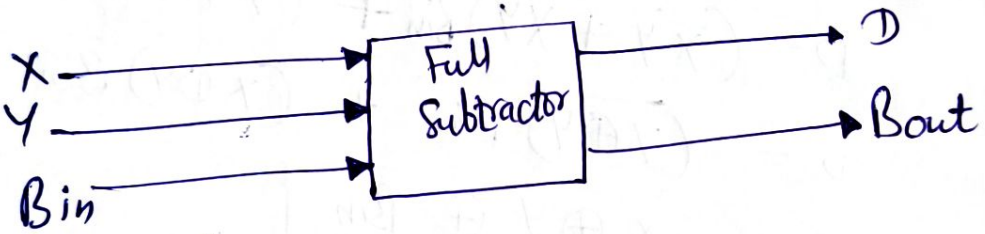
K-map :-



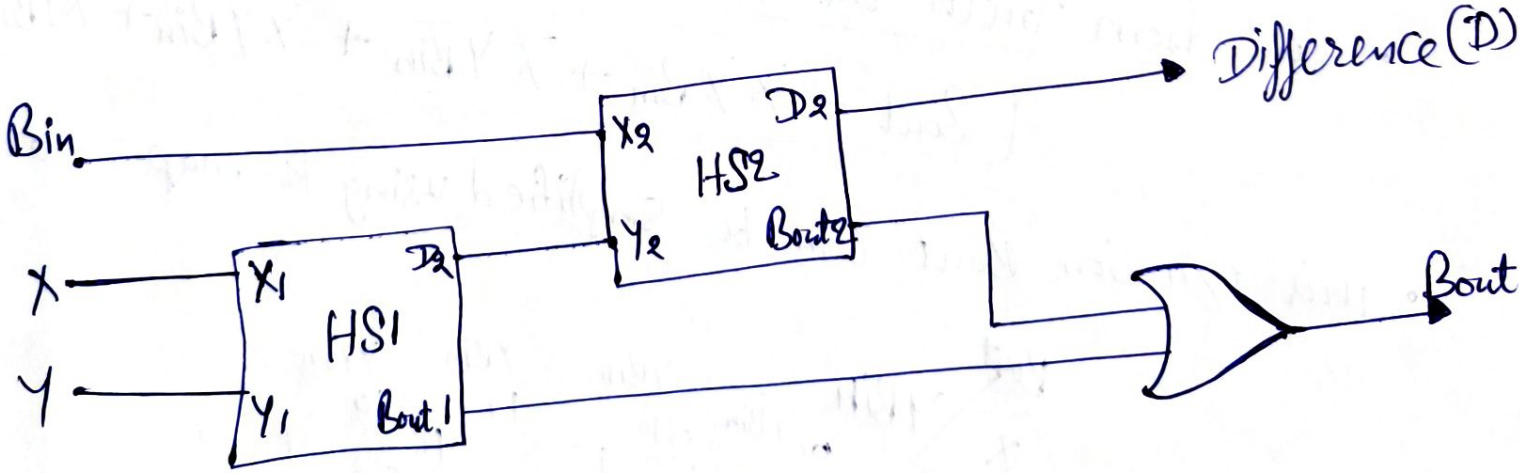
- From above expression, the half subtractor can be implemented using an Ex-OR gate, NOT gate & one AND gate as in logic diagram (ii).

✱ Full Subtractor :-

- A full subtractor is a combinational circuit that performs subtraction involving three bits, namely minuend bit, subtrahend bit and the borrow from the previous stage.
- It has three inputs, X (minuend), Y (subtrahend) & Bin (borrow from previous stage), and two outputs D (difference) and Bout (borrow out).
- The full-subtractor can be implemented using two half-subtractors and an OR gate.



(i) logic Symbol



(ii) Using half subtractors Full Subtractor

(iii) Truth table:-

Inputs			Outputs	
Minuend Bit X	Subtrahend Bit Y	Borrow in Bin	Difference D	Borrow out Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- From this table, SOP expression for the difference (D) is:-

$$D = \bar{X}\bar{Y}Bin + \bar{X}Y\bar{Bin} + X\bar{Y}\bar{Bin}$$

$$D = (\bar{X}\bar{Y} + XY)Bin + (\bar{X}Y + X\bar{Y})\bar{Bin}$$

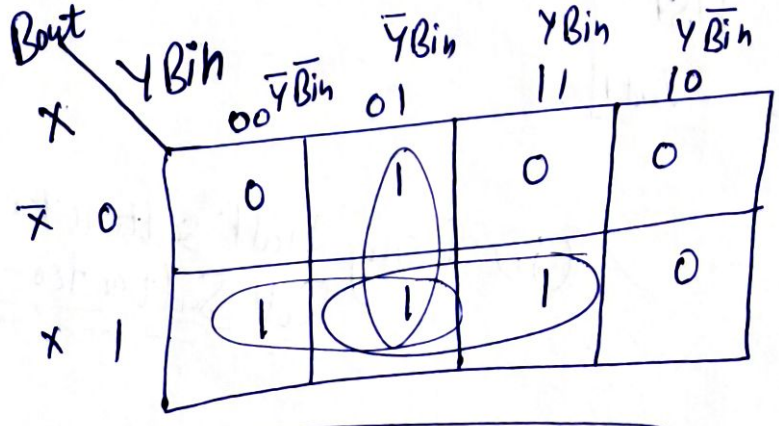
$$D = \overline{(X \oplus Y)}\,Bin + (X \oplus Y)\,\bar{Bin}$$

$$\boxed{D = X \oplus Y \oplus Bin}$$

- Bout from Truth table:-

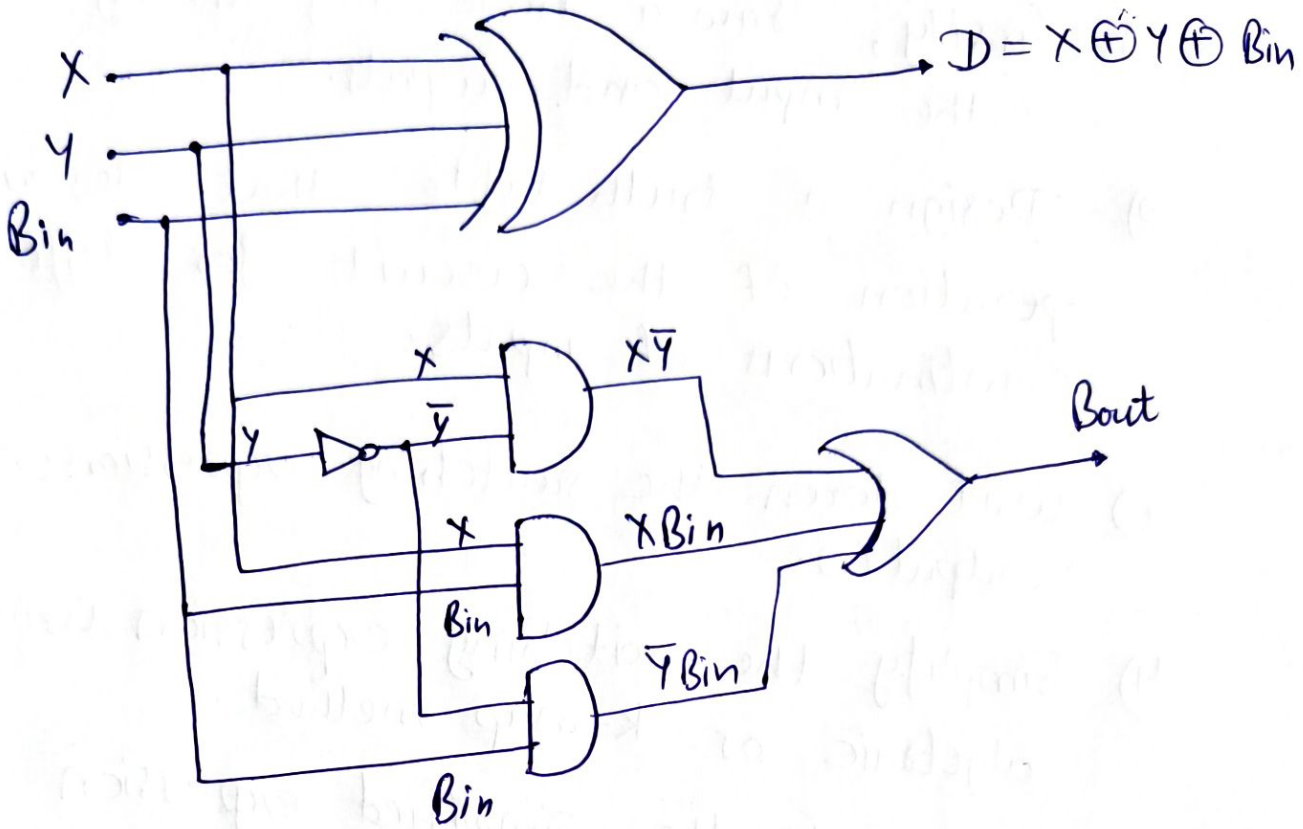
$$\boxed{Bout = \bar{X}\bar{Y}Bin + \bar{X}Y\bar{Bin} + \bar{X}YBin + XYBin}$$

- This equation Bout can be simplified using k-map →



$$\boxed{Bout = X\bar{Y} + XBin + \bar{Y}Bin}$$

- Realization of full subtractor using basic logic gates :-



- Note:- One can notice that equation for D is same as Sum output for a full adder, & borrow output Bout resembles the carry output for full adder except that one of the inputs is complemented. From these similarities, it is possible to convert a full adder into a full subtractor by merely complementing that input prior to its application to the input of gates which form the borrow output.

Procedure for the design of Combinational Circuits:-

1) firstly, Draw a block diagram by identifying the input and outputs.
2) Design a truth table that describes the operation of the circuit for different combinations of inputs.
3) Writ down the switching expression (s) for the output (s).
4) Simplify the switching expression using either algebraic or K-map method.
5) Implement the simplified expression using logic gates.

* Parallel Binary Adder or (4-bit binary parallel adder) (Ripple Carry Adder)

- Binary adder is the combinational circuit that is used to find the sum of two binary numbers of any length.
- Modern computers and calculators use no. ranging from 8 bit to 64 bits.
- The addition of multibit numbers can be accomplished using several full adders.
- The 4-bit adder using full-adder circuit is capable of adding two 4-bit numbers resulting in a 4-bit sum and a carry output.
- Since all the bits of the augend and addend are fed into the adder circuits simultaneously and additions in each position are taking place at the same time, this circuit is known as parallel adder.

Ex: Let the 4-bit words to be added be represented by $A_3 A_2 A_1 A_0 = 1111$ and $B_3 B_2 B_1 B_0 = 0011$.

→

Significant place	4	3	2	1	
Input carry	(1)	(1)	(1)	(0)	$C_3C_2C_1C_0$
Augend word A:	1	1	1	1	$A_3A_2A_1A_0$
Addend word B:	0	0	1	1	$B_3B_2B_1B_0$
	(1)	0	0	1 0	→ Sum

Cout → Output Carry

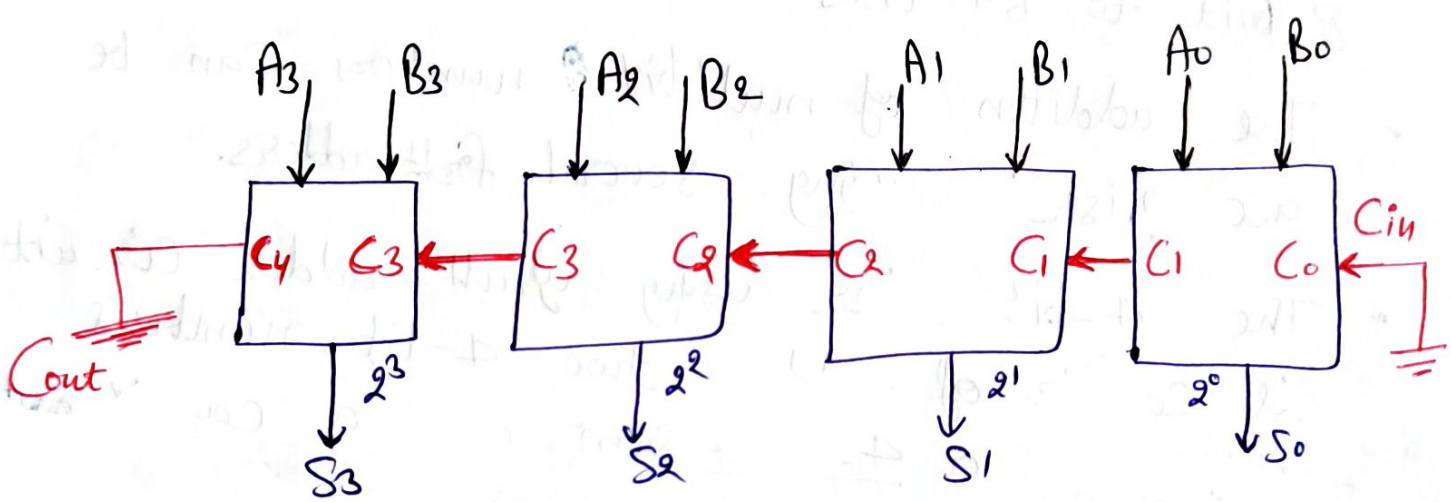


fig:- 4-bit binary parallel adder

- In this circuit, input to each full adder will be A_i, B_i & C_i & the output will be S_i & C_{i+1}, i where varies from 0 to 3.

- Also, the carry output of the lower order stage is connected to the carry input of the next higher order stage. This type of adder is called "Ripple-carry adder".

- In the least significant stage, A_0, B_0 and C_0 (which is 0) are added resulting in Sum S_0 and Carry C_1.
- This carry C_1 becomes the carry input to second stage. Similarly, in second stage, A_1, B_1 & C_1 are added resulting in S_1 and C_2;
- In third stage, A_2, B_2 & C_2 are added resulting in S_2 & C_3; in fourth stage, A_3, B_3 & C_3 are added resulting in S_3 & C_4 which is the o/p Carry.
- Thus, circuit results in a sum ($S_3S_2S_1S_0$) & Carry Output (C_{out}).

» The paralled binary adder generate its output immediately after the inputs are applied, but its speed of operation is limited by the Carry propagation delay through all stages.

» The propagation delay (t_p) of a full adder is the time differece between the instant at which the inputs (A_i, B_i & C_i) are applied & the instant at which its outputs (S_i and C_{i+1}) are generated.

» Therefore, in a 4-bit binary adder, output in LSB stage is generated after t_p second. In second stage, output generated in $2t_p$ second. In third stage, it will be $3t_p$ second. & in fourth stage it will be $4t_p$ seconds

- where each full adder has a propagation delay of 50 ns, the output in fourth stage will be generated only after $4t_p = 4 \times 50$ ns $\Rightarrow$ 200 ns. (nano second).

- This magnitude of such delay is not allowed for high speed computers. Therefore, several methods to reduce this delay.

- One of the methods of speeding up this process is look-ahead carry addition which eliminates the ripple-carry delay. This method is based on carry generate & carry propagate functions of the full adder.

⇓

This look-ahead carry addition scheme, utilises the logic gates to look at the lower order bits of the augend and addend if a higher order carry is to be generated. This requires extra circuitry for getting high speed adders.

- Advantages of 4-bit Ripple carry adder (Parallel binary adder):-
  - Simple & easy to implement.

Disadvantage:- Parallel binary adder slow for large no. of bits addition because each stage must wait for pevious carry.

4-bit Serial Adder :-

- Parallel adder performs the addition of two binary numbers at higher speed, but disadvantage is that it requires a large amount of logic circuitry.
- Serial adder performs bit by bit addition, So, it requires simple circuitry than parallel adder but results in a low speed of operation.

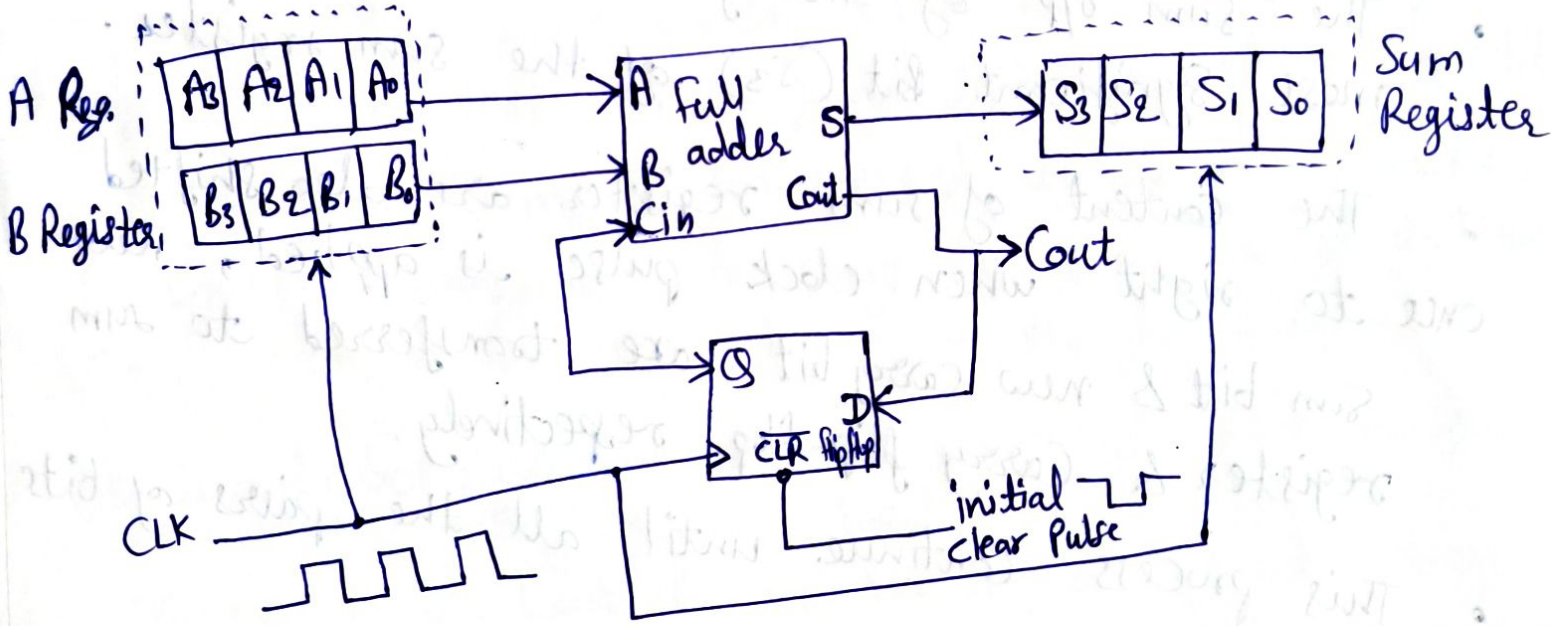


fig:- 4-bit Serial Adder

- The two shift registers A & B are used to store the no. to be added serially.
- A single full adder is used to add one pair of bits at a time along with the carry.

- D flip-flop (Delay) i.e. carry flip flop, is used to store the carry o/p of the full adder, to add it to next position of numbers in registers.
- The contents of the shift registers shifts from left to right & their o/p starting from A_0 & B_0 are fed into a single full adder along with the o/p of the carry flip-flop upon application of each clock pulse.
- The sum o/p of the full adder is fed to the most significant bit (S_3) of the sum register.
- The content of sum register are also shifted once to right when clock pulse is applied, new sum bit & new carry bit are transferred to sum register & carry flip-flop respectively.
- This process continue until all the pairs of bits are added.

Serial Adder	Parallel adder
1. Serial adder is less fast it requires fewer Components.	1. Parallel adder is generally faster. It requires more Components Compared to Serial Adder.
2). Addition is performed bit by starting from the LSB.	2) All the bits are added Simultaneously.

Example of serial adder / operation of serial adder :-

- Let the augend → $A_3 A_2 A_1 A_0$ is 0 1 1 1
  - → addend → $B_3 B_2 B_1 B_0$ is 0 0 1 0
  - → Stored in shift registers A & B, respectively.
  - → Carry flip-flop initially cleared to the 0 state, So that $C_{in} = 0$.

(i) First clock Pulse :- Before first clock pulse occurs, the full adder inputs are $A_0 = 1$, $B_0 = 0$ & $C_{in} = 0$. output of full adder will be $S = 1$ & $C_{out} = 0$. when first clock pulse occurs, the value in the A & B registers shift from left to right by one bit. In addition, Sum is transferred to S_3 of sum register, & Cout is transferred to carry flip flop, whose o/p become 0. which is the carry input of the full adder.

(ii) Second clock Pulse :- Now, $A_0 = 1$, $B_0 = 1$ & $C_{in} = 0$ at full adder input.

. Therfore $S = 0$, $C_{out} = 1$ when the second clock pulse occurs, A, B and Sum registers again shift right; $S = 0$ is transferred to S_3, and $C_{out} = 1$ is transferred to the carry flip flop.

(iii) Third clock Pulse : $A_0 = 1$, $B_0 = 0$, $C_{in} = 1$. These values produce $S = 0$ & $C_{out} = 1$ at full adder outputs. Third clock pulse occurs. the A, B & Sum register shift right, Sum = 0 goes to S_3 & $C_{out} = 1$ goes to carry flip flop.

(iv) fourth Clock Pulse :- Both A_0 & $B_0 = 0$, $C_{in} = 1$ Then produces $S = 1$, $C_{out} = 0$ After fourth clock $S = 1$ to S_3 & initiates all other transfers. Atend $S = 1001$ & $C_{out} = 0$

Serial Subtraction using 2's Complement:-

- A serial subtractor can be obtained by converting the serial adder using 2's complement system.
- For subtraction, the subtrahend is stored in the B register and must be 2's complemented before it is added to the minuend stored in the A register.
- One simple way to accomplish this is to complement the B register and make initial $C_{in} = 1$. Cinstead of 0 prior to the first clock pulse.
- This is easy by feeding inverted output $\bar{B}_0$ into full adder intead of B_0 & initally setting the carry flip-flop to 1 instead of clearing it.
- The remaining circuitry is same as serial adder.

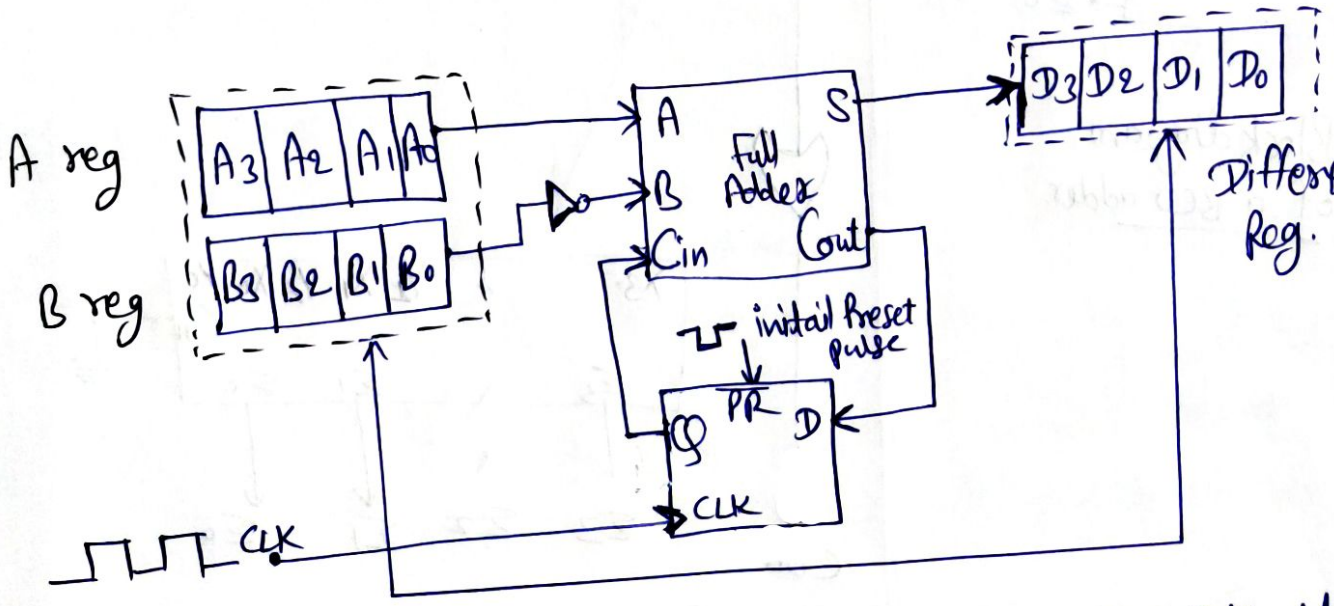
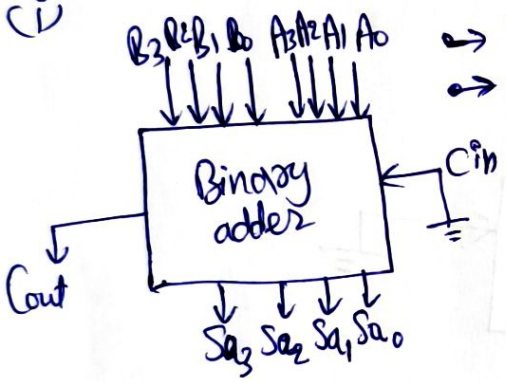


fig:- 4-bit serial Subtractor using Full adder.

BCD adder:-

- A binary coded decimal (BCD) adder is a circuit that adds two BCD digits in parallel and produces a sum digit which is also in BCD.
- This adder has two 4-bit BCD inputs $B_3\ B_2\ B_1\ B_0$, $A_3\ A_2\ A_1\ A_0$ and a carry input (Cin). It has also a 4 bit sum output ($S_{a_3}\ S_{a_2}\ S_{a_1}\ S_{a_0}$) and a carry output (Cout).
- Here the sum output is also in BCD form.

(i)



→ min value of set A & B → 0000 → 0
→ max value of set A & B → 1001 → 9
→ max value of Cin → 1

So, max value of Sum can be:-

(S) Sum = 9 + 9 + 1 ⇒ 19

(So, we have taken the values in table from (0000) to 19 → (10011).
↳ 0

Fig:- Block diagram of BCD adder

Example:- assume

$$\begin{array}{rl} & A \rightarrow \text{BCD no} \\ + & B \rightarrow \text{BCD no.} \\ \hline \text{Carry} & \text{Sum} \end{array}$$

Sum → S≤9 valid BCD no.
Sum → if S>9 invalid BCD no. add +6

$$\begin{array}{rl} & \text{Sum} \\ + & 6 \\ \hline \text{Cout} & \text{So} \end{array}$$

So → Correct/valid BCD Sum. (new carry So)

Cases: a) if Sum≤9 and Carry(C)=0 ; $\begin{array}{rl} 7 & \rightarrow 0111 \\ +1 & \rightarrow 0001 \\ \hline 8 & 1000 \end{array}$ → no carry

in this case no carry occurs. Answer is correct BCD number.

(ii) Case (b):- if Sum > 9 and Carry = 0 ; 7 → 0111
+4 → 0100
11 1011 → no carry invalid number
+0110 → +6 add
0001 0001

(iii) Case (c): if S ≥ 9, but Carry C = 1
take no: ; 9 → 1001
+8 → 1000
17 0001 0001 → carry generated invalid BCD no
+ 0110
0001 0111
1 7

- For invalid BCD no: 10 to 15, we generate a function that produce output 1.

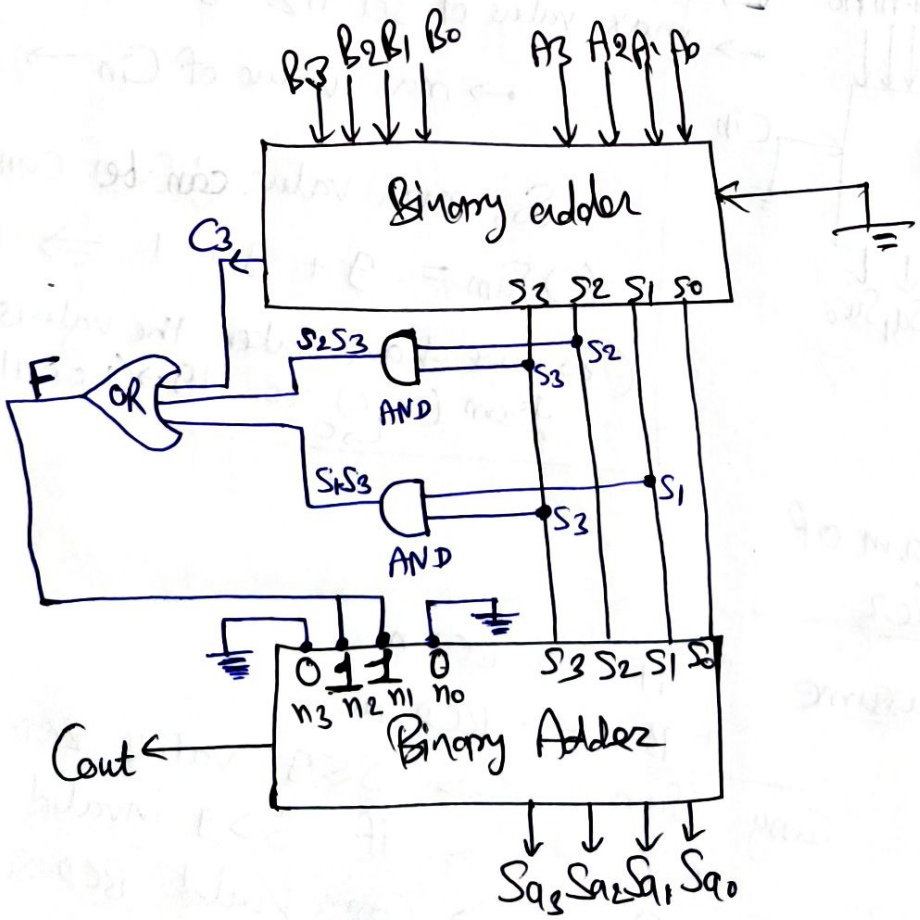


fig:- BCD adder using 7483 IC.

(ii) Case (b):- if Sum > 9 and Carry = 0 ;

$$\begin{array}{r} 7 \rightarrow 0111 \\ +4 \rightarrow 0100 \\ \hline 11 \quad 1011 \end{array} \rightarrow \text{no carry invalid number}$$

$$+0110 \rightarrow +6 \text{ add}$$

$$0001\ 0001$$

(iii) Case (c): if $S \geq 9$, but Carry $C = 1$

take no: ;

$$\begin{array}{r} 9 \rightarrow 1001 \\ +8 \rightarrow 1000 \\ \hline 17 \quad 0001\ 0001 \end{array} \rightarrow \text{carry generated invalid BCD no.}$$

$$+\ 0110$$

$$0001\ 0111$$

$$1 \quad 7$$

- For invalid BCD no. 10 to 15, we generate a function that produce output 1.

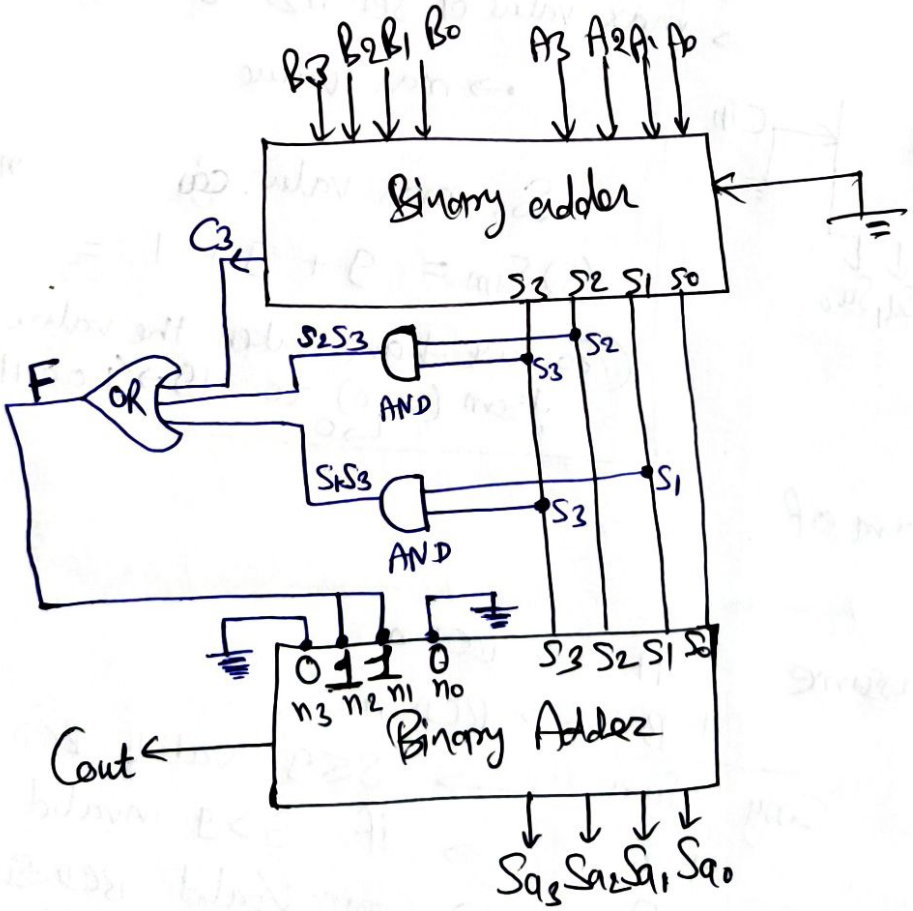


fig:- BCD adder using 7483 IC.

A BCD adder circuit must be able to do following:-

i) Add two 4-bit BCD no. using straight binary addition.

ii) If four bit sum is equal to or less than 9: $(S \leq)$, Sum is in proper BCD form & no correction is needed.

iii) If four bit sum is greather than $(S>9)$ or if carry is generated from sum, the sum is not in BCD form. Then, digit 6 (0110) should be added to sum and produce BCD results.

Table:- BCD addition result with Correction indicated.

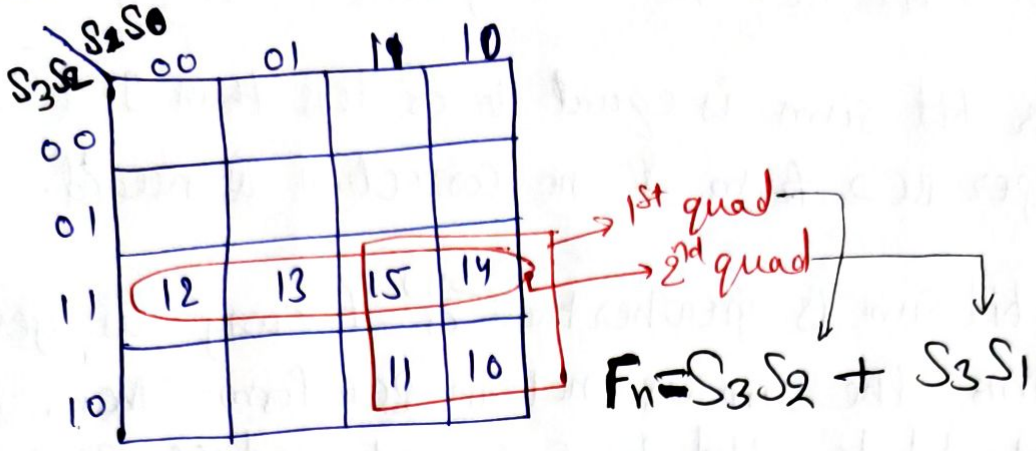
Decimal Digit	Uncorrected BCD Sum					Corrected BCD Sum					
	C_3	S_3	S_2	S_1	S_0	Cout	S_3	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	0	No Correction Required
1	0	0	0	0	1	0	0	0	0	1	
2	0	0	0	1	0	0	0	0	1	0	
3	0	0	0	1	1	0	0	0	1	1	
4	0	0	1	0	0	0	0	1	0	0	
5	0	0	1	0	1	0	0	1	0	1	
6	0	0	1	1	0	0	0	1	1	0	
7	0	0	1	1	1	0	0	1	1	1	
8	0	1	0	0	0	0	1	0	0	0	
9	0	1	0	0	1	0	1	0	0	1	
10	0	1	0	1	0	1	0	0	0	0	Correction Required
11	0	1	0	1	1	1	0	0	0	1	
12	0	1	1	0	0	1	0	0	1	0	
13	0	1	1	0	1	1	0	0	1	1	
14	0	1	1	1	0	1	0	1	0	0	
15	0	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	

Kmap

C_3

Cout

K-map Simplification for incorrect/invalid BCD no:-



Total F = $S_3S_2 + S_3S_1 + C_3$ Condition for correction.

Binary Multiplier:-

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

(1) Multiplier using Shift method:-

- To understand multiplication process using shift method, consider the multiplication of two 4-bit binary numbers 1010 and 1011.

example:-

```
    1 0 1 0   → multiplicand
  x 1 0 1 1   → Multiplier
  -----------
    1 0 1 0   } → Partial product 1
  1 0 1 0 x   } → Partial Product 2
  0 0 0 0 x x } → Partial Product 3
1 0 1 0 x x x } → Partial Product 4
  -----------
1 1 0 1 1 1 0 } → Product addition
```

- From this multiplication, it is clear that if multiplier bit is 1, then the multiplicand is simply copied as a partial product; if multiplier bit is 0; then partial product 0. For every

partial product, it is shifted one bit to the left of previous partial product. This process is continued till last bit. & then partial product are added.

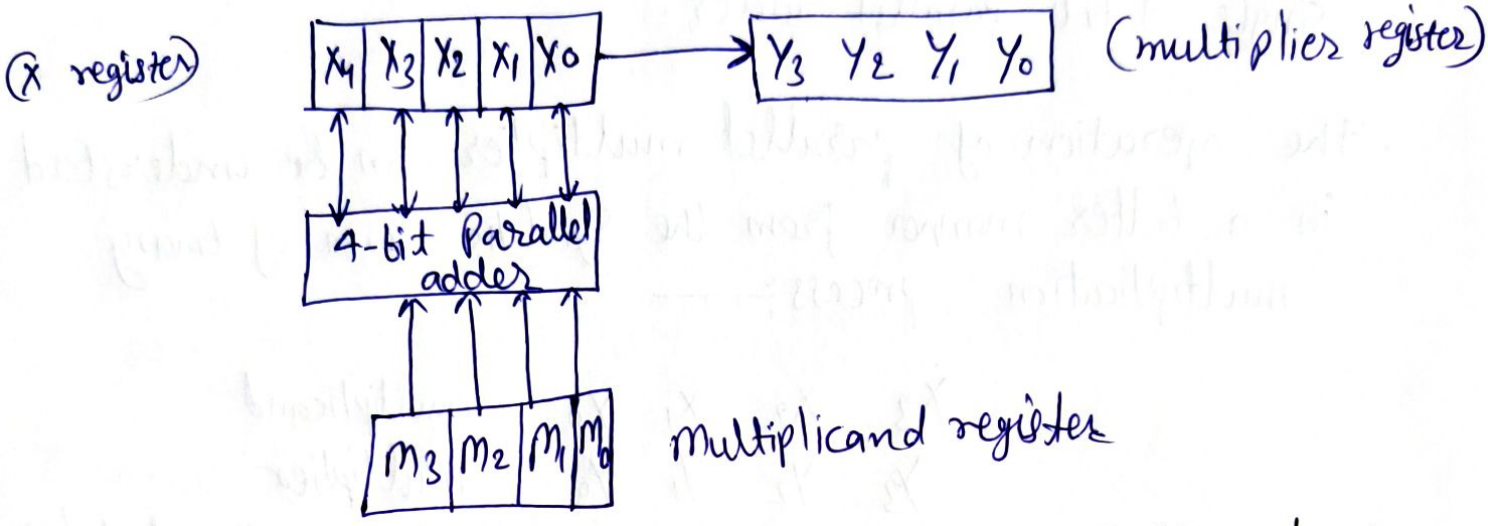


Fig:- 4-Bit binary multiplier using shift method.

- 4-bit multiplier bits are stored in Y register $(Y_3 Y_2 Y_1 Y_0)$ & 4-bit multiplicand is stored in register M $(M_3 M_2 M_1 M_0)$.
- X register $(X_4 X_3 X_2 X_1 X_0)$ is initially cleared to 00000.
- Perfor multiplication, the least significant bit of multiplier bit (Y_0) is checked wheather it is 0 or 1.

•If $Y_0 = 1$, no. in multiplicand register (M) is added with least significant 4-bit of X register $(X_3 X_2 X_1 X_0$; X_4 is to store carry in addition process).

- Combined X & Y register is shifted to right by 1 bit.
- if $Y_0 = 0$, Combined X & Y register is shifted to right by 1 bit without performing addition.
- This process repeated 4-times to perform 4-bit multiplication

(2) Parallel Multiplier :-

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

- The operation of parallel multiplier can be understood in a better manner from the symbolic form of binary multiplication process :—

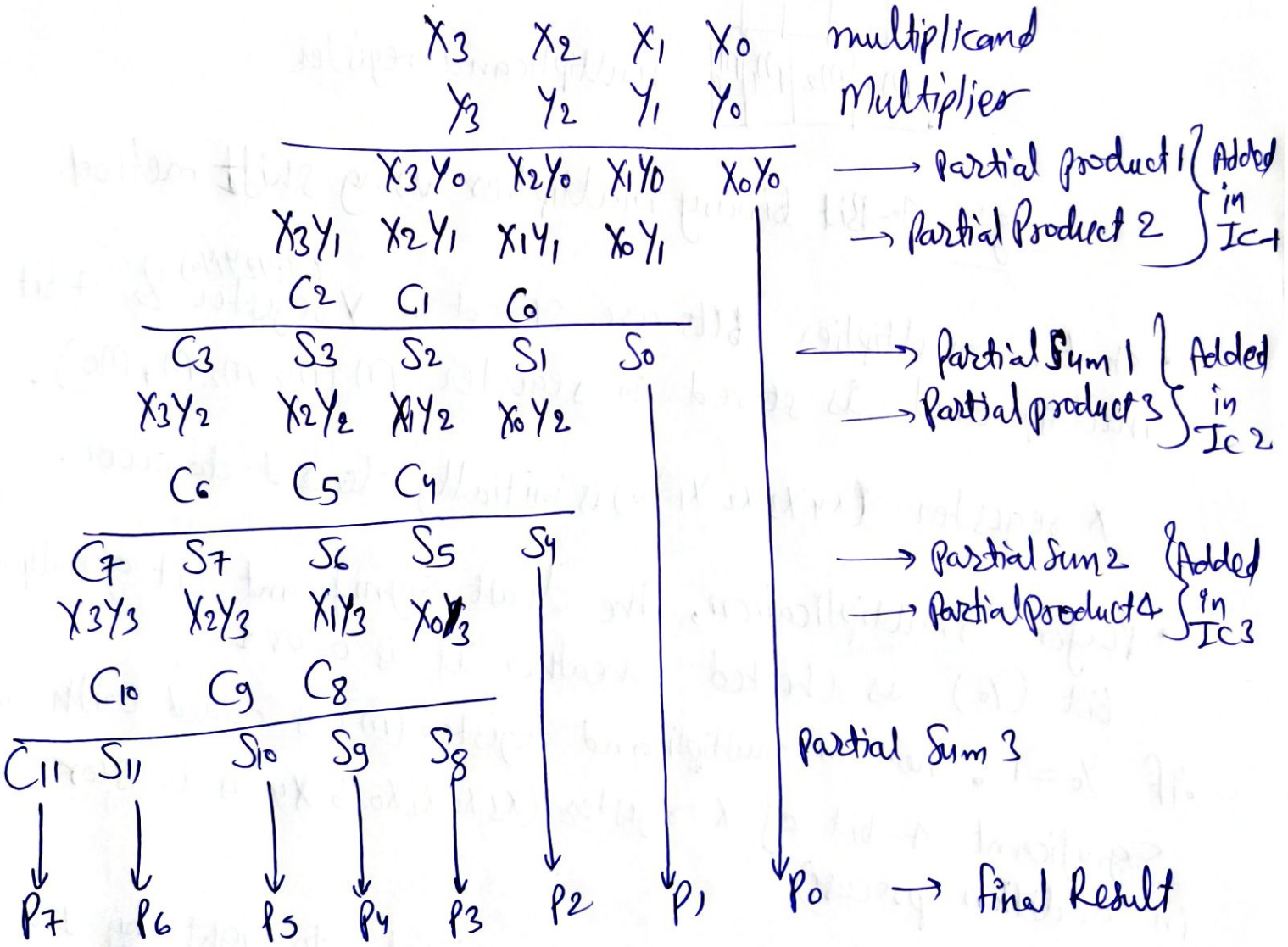
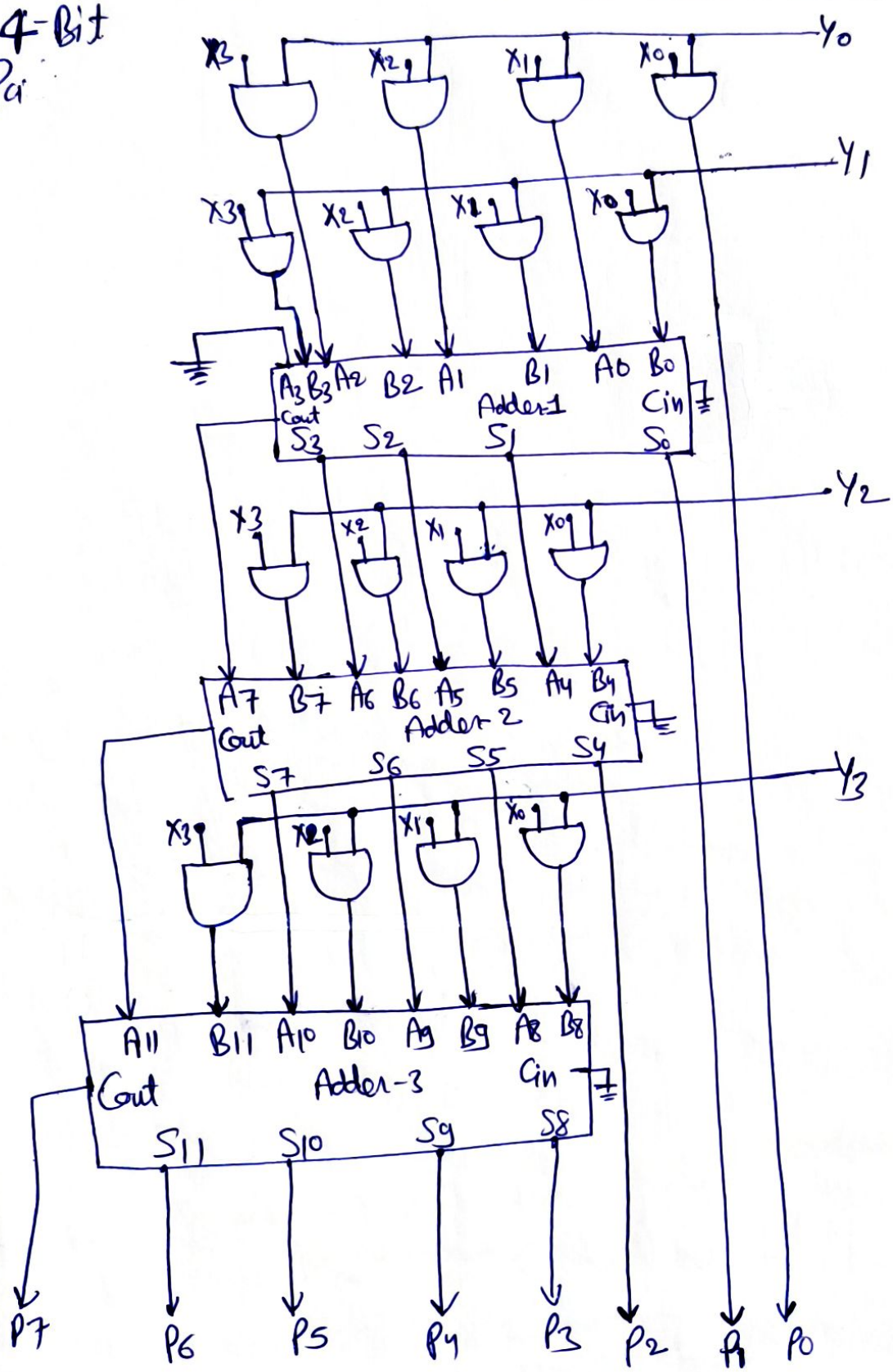


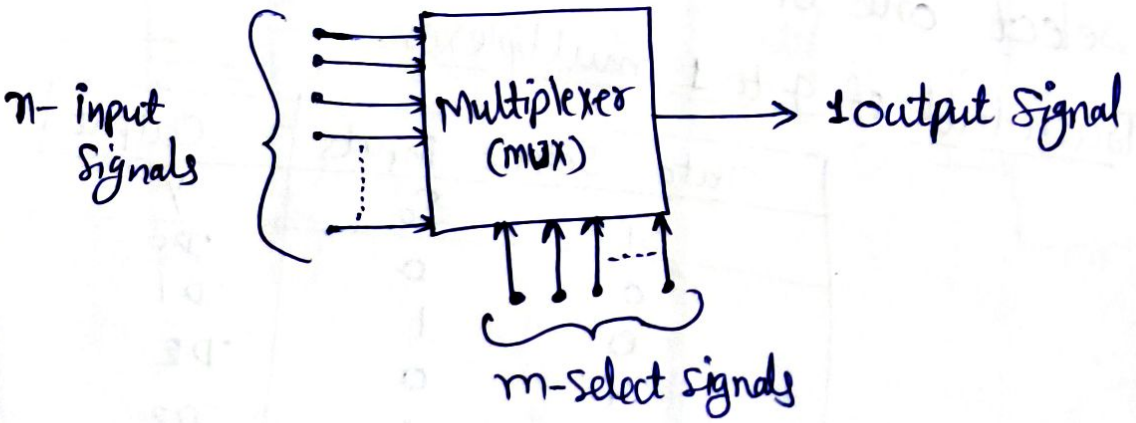
Fig: Symbolic form of Binary multiplication process

fig:- 4-Bit Pai


Y0
Y1
Y2
Y3
X3 X2 X1 X0
A3 B3 A2 B2 A1 B1 A0 B0
Cout
Adder-1
Cin
S3 S2 S1 S0
A7 B7 A6 B6 A5 B5 A4 B4
Adder-2
S7 S6 S5 S4
A11 B11 A10 B10 A9 B9 A8 B8
Adder-3
S11 S10 S9 S8
P7 P6 P5 P4 P3 P2 P1 P0

Multiplexer (Data Selectors) :- 'Multiplex' means "many to one".

- Multiplexing is the process of transmitting a large number of information over a single line.
- A digital multiplexer (MUX) is a combinational circuit that selects one digital information from several sources and transmits the selected information on a single output line.
- A multiplexer is also called a data selector because it selects one of many inputs and send the information to the output.
- Multiplexer is an electric switch that connects one out of n input to an output.
- Multiplexer is functionally complete i.e. all boolean functions can be realized using one multiplexer without any other gates. (Universal complete circuit).
- Block diagram of Multiplexer:-



- The multiplexer has several data-input lines & a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- The selection lines decide the number of input lines of a particular multiplexer.
- If the number of n input lines is equal to 2^m, then m select lines are required to select one of the n input lines. [$n = 2^m$]
- Example:- ① To select 1 out of 4 input lines, 2 select lines are required.
  ② To select 1 out of 8 input lines, 3 select lines are required & so on.....

4-input Multiplexer :-

- It has 4-input data lines ($D_0 - D_3$), a single output line (Y) and two select lines (S_0 and S_1) to select one of the 4-input lines.
  - Truth table of 4 to 1 multiplexer

Data Select inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

- A logical expression for the output in the term of the data input and the select inputs can be derived as follows :

- for $S_1=0, S_0=0$, $Y = D_0 \bar{S}_1 \bar{S}_0 = D_0 \cdot \bar{0} \cdot \bar{0} = D_0 \cdot 1 \cdot 1$

$$\boxed{Y = D_0}$$

$S_1 = 0, S_0 = 1$, $Y = D_1 \bar{S}_1 S_0 = D_1$

$S_1 = 1, S_0 = 0$, $Y = D_2 S_1 \bar{S}_0 = D_2$

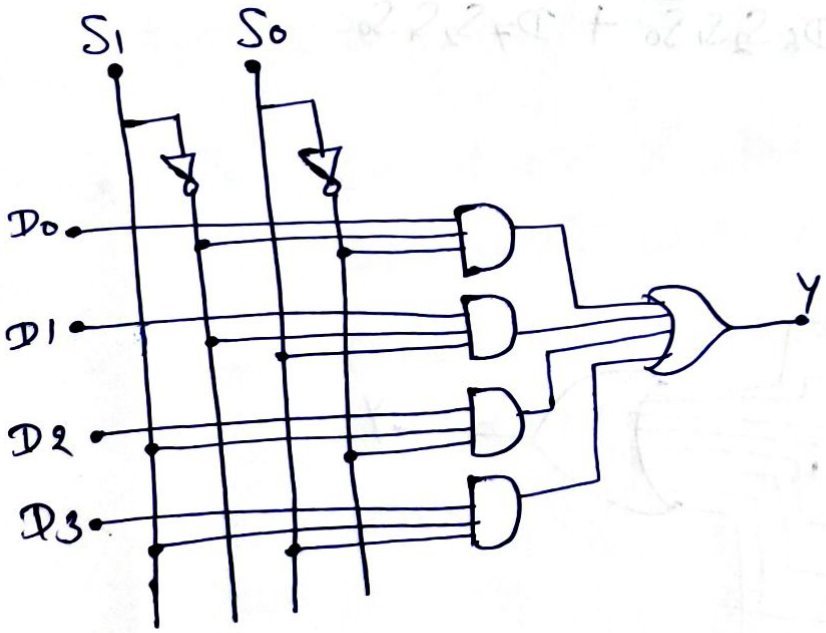
$S_1 = 1, S_0 = 1$, $Y = D_3 S_1 S_0 = D_3$

final expression for the data output —

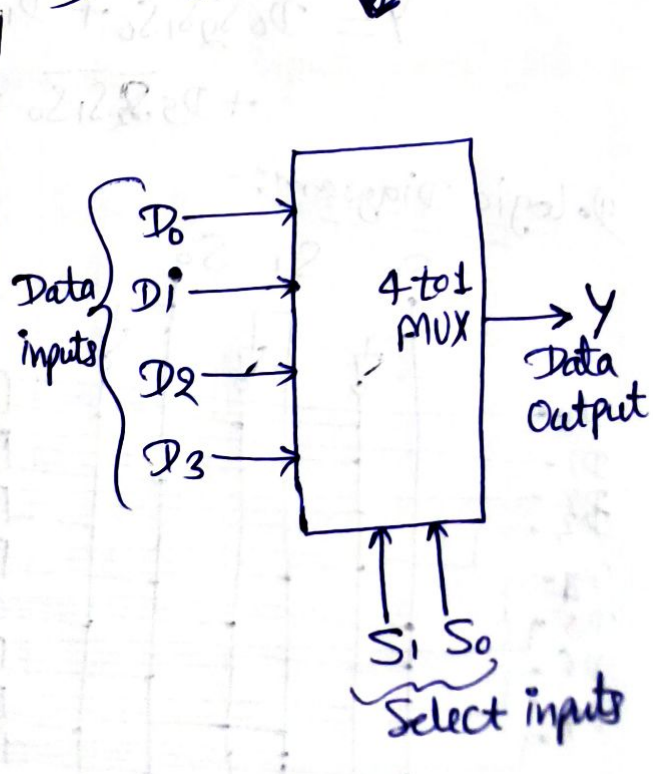
$$\boxed{Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0}$$

- The above expression (4 to 1 mux) can be implemented using
  - 2 NOT gates
  - 4 AND gates
  - 1 OR gate

a) Logic diagram



b) Logic Symbol



8 to 1 Multiplexer (IC 74151) :-

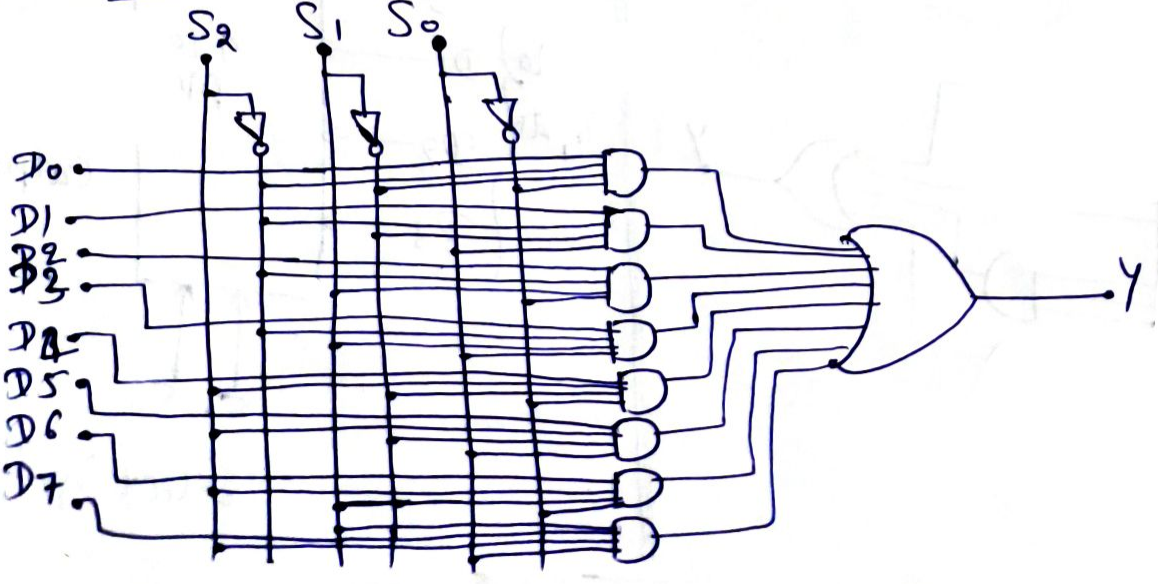
- 8 to 1 multiplexer has eight data inputs (D_0-D_7) three select input ($2^3 \rightarrow 8$) i.e. S_2 S_1 S_0, and a single output (Y).
- Since, $2^3 = 8$, three bits are required to select any one of the eight data bits.
- Truth table of 8 to 1 multiplexer :-

Inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

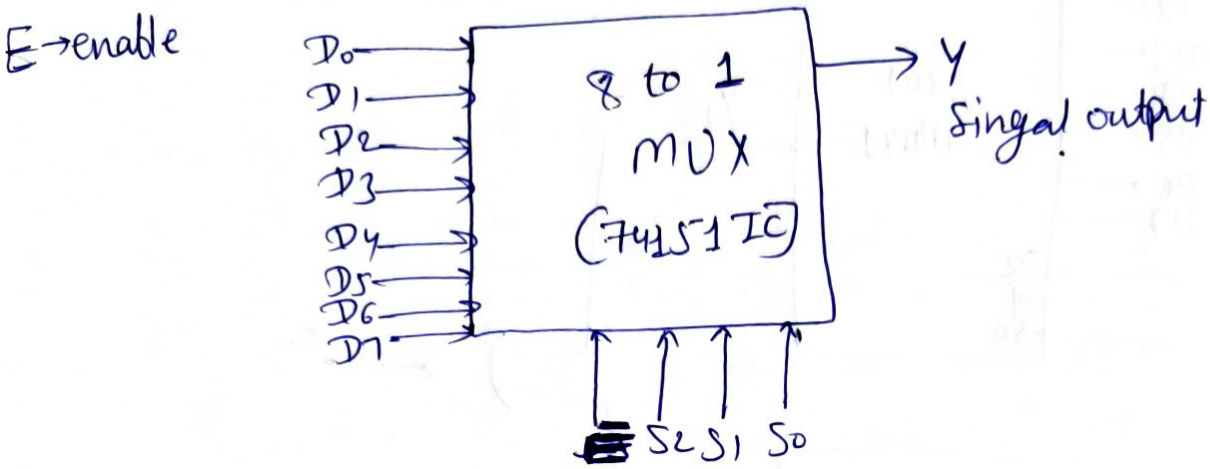
- Logic expression

$$Y = D_0 \bar{S}_2 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_2 \bar{S}_1 S_0 + D_2 \bar{S}_2 S_1 \bar{S}_0 + D_3 \bar{S}_2 S_1 S_0 + D_4 S_2 \bar{S}_1 \bar{S}_0 + D_5 S_2 \bar{S}_1 S_0 + D_6 S_2 S_1 \bar{S}_0 + D_7 S_2 S_1 S_0$$

- Logic Diagram:-



(b) logic symbol of IC 74151 (8 to 1 multiplexer)



- when $\bar{E}=0$, the select input $S_2S_1S_0$ will select one of the data input to pass through the output Y.
- When $\bar{E}=1$, the multiplexer is disabled.

(16 to 1 – MUX → Homework)

Q1. Implement a 16-to-1 multiplexer using two 8-to-1 multiplexer ICs (74151).

Ans:- A 16 to 1 multiplexer can be implemented using two 8 to 1 multiplexers and one 2-input OR gate.

- To select one of the 16 inputs, four select lines ($S_3S_2S_1S_0$) are required.
- From these four select lines, the least significant three select lines ($S_2S_1S_0$) are connected with three select inputs of both multiplexer ICs.

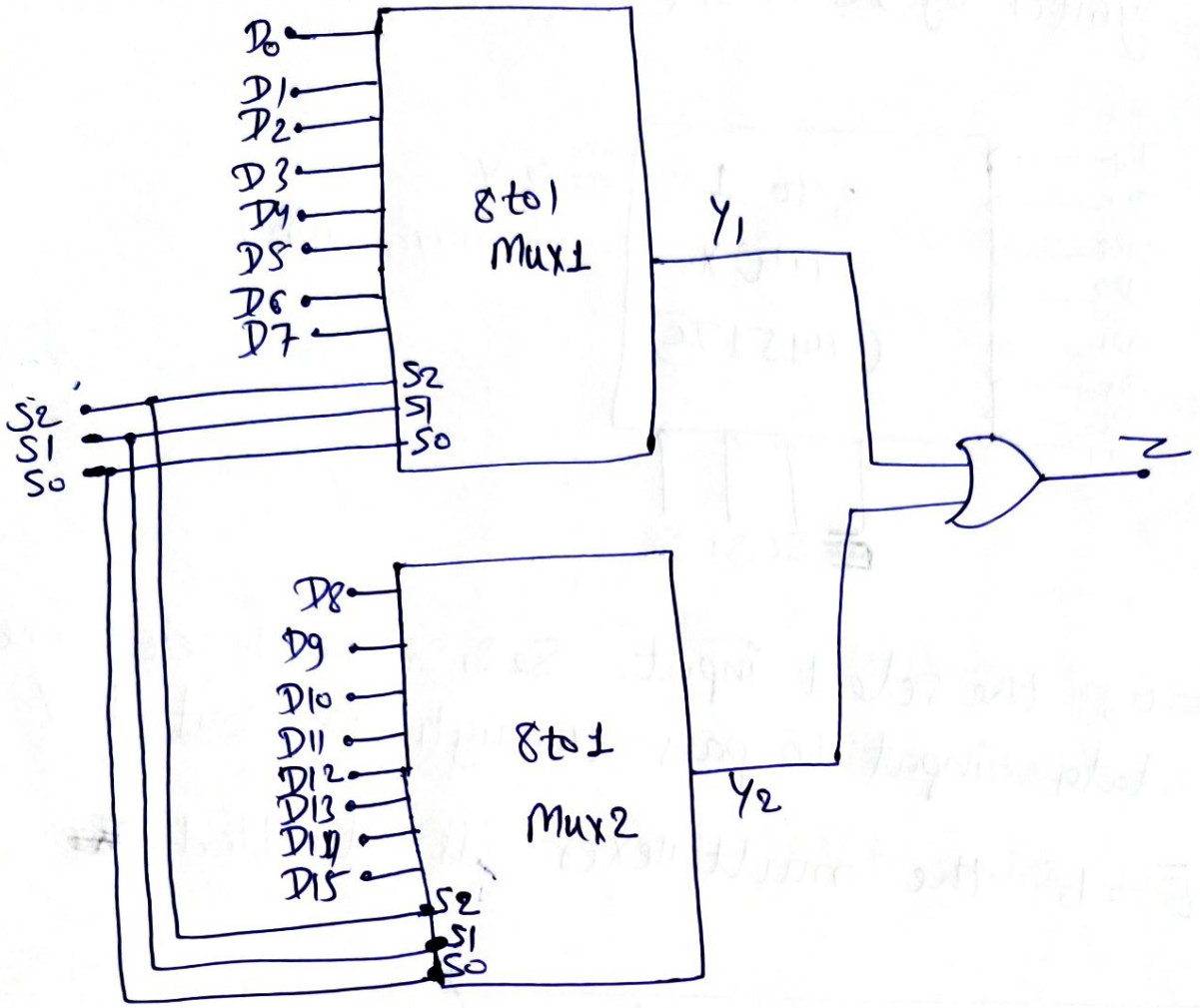


fig:- 16 to 1 MUX using two ICs 74151 — 8 to 1 MUXs.

Implementation of Boolean Expression Using Multiplexer

- if Boolean expression has (n+1) variables.

n variables → Connected to selectlines of multiplexer

1 → (single variable) → it can be '1' & '0' as input of multiplexer.

Ex: If A is singlevariable, input of multiplexer are A, $\bar{A}$, '1' and '0'.

Q1:- Implement the boolean expression multiplexer.

$$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$$

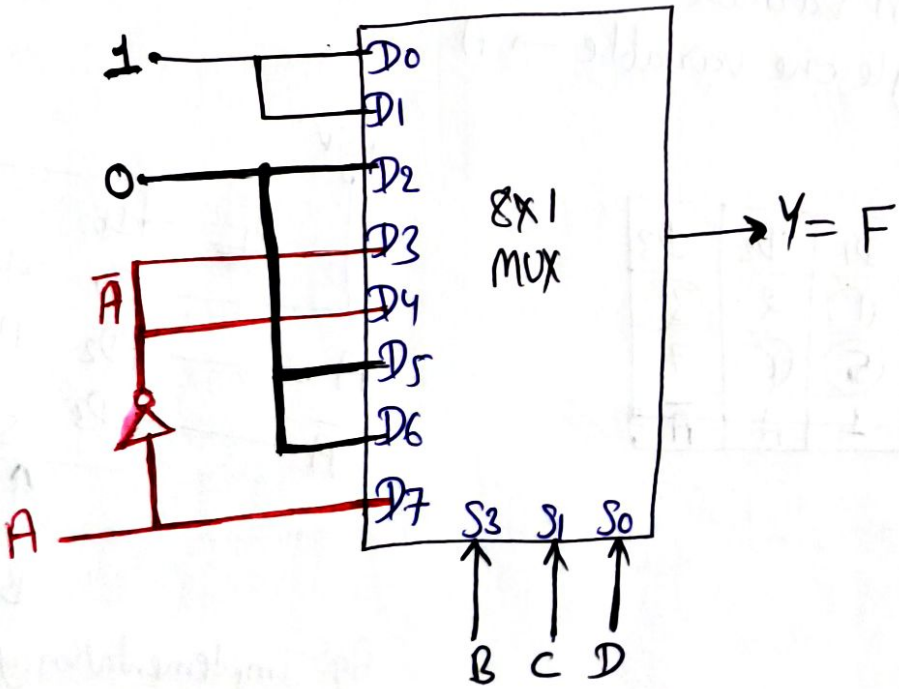
Ans:-

- Four variable A, B, C, D. → (n+1) variable
  - A → 1 single variable (A) → input
  - B, C, D → n variable → So, n = 3 variable (B, C, D)
    - means → 3 select lines (B, C, D)
    - 2^n inputs → 2^3 → 8×1 multiplexer

- The procedure for implementing the function are
  (i) List the input of multiplexer &
  (ii) List under them all minterms in two rows, like

Table: List for implementation of the function

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
$\bar{A}$	0	1	2	3	4	5	6	7	→ first half minterms
A	8	9	10	11	12	13	14	15	→ 2nd half minterms
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A	



* Following Rules to find the values for i/p of multiplexer. (how to implement function by circling the minterms of the function):-

(i) If both the minterms in a column are not circled, apply '0' to the corresponding input.

(ii) If both the minterms in a column are circled, apply 1 to the corresponding input.

(iii) If the bottom minterm is circled and the top is not circled, apply A to the input.

(iv) If the top minterm is circled and the bottom is not circled, apply $\bar{A}$ to the input.

Q2:- Implement the following function using a multiplexer:

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

Ans:- The given function has three variables → A, B, C } (n+1) variable

n variable → B, C ; (2 variable → 2 → 4 input mux); 2 select lines

single one variable → A

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	1	A	$\bar{A}$

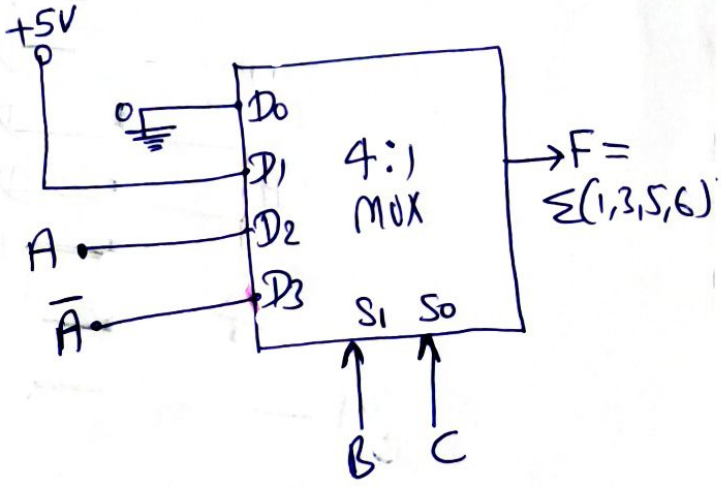


fig:- Implementation of $F(A,B,C) = \Sigma(1,3,5,6)$

Q3: Generate the logic function given in table 1 using IC74151 – 8 to 1 MUX.

Input C	B	A	output Y	
0	0	0	0	→ D0
0	0	1	1	→ D1
0	1	0	1	→ D2
0	1	1	0	→ D3
1	0	0	0	→ D4
1	0	1	0	→ D5
1	1	0	0	→ D6
1	1	1	1	→ D7

Ans: According to the question, 8 to 1 multiplexer is used to implement 3-variable (A, B, C) logic function.

- Let, A, B & C are the input variable connected to S_0, S_1 & S_2 respectively.
- Output Y is low when $ABC = 000$ (Low), means multiplexer input D_0 should be connected to low.

[0 (low) output Y at → D_0, D_3, D_4, D_5 & D_6
  1 (High) output Y at → D_1, D_2 & D_7]

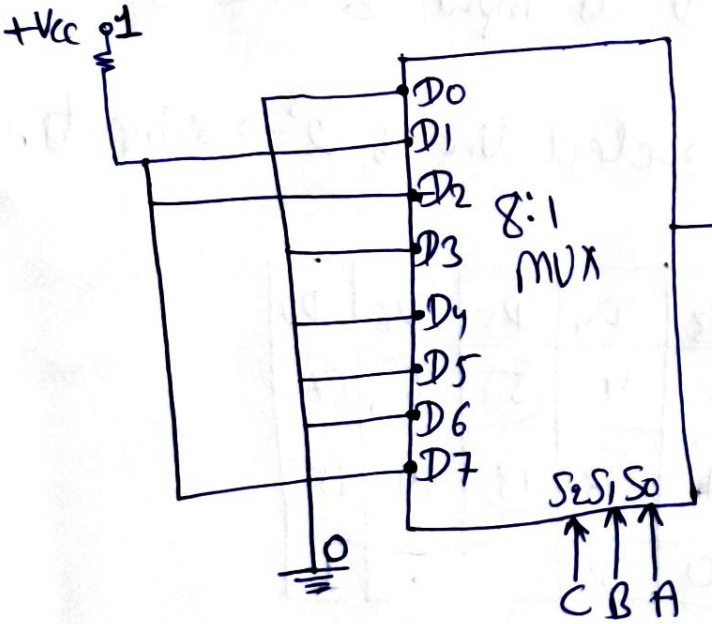


Fig:- 3 variable logic function generator using 8 to 1 MUX.

Q4) Implement the logic function in Table 2. using a 74151A eight input data selector/MUX.

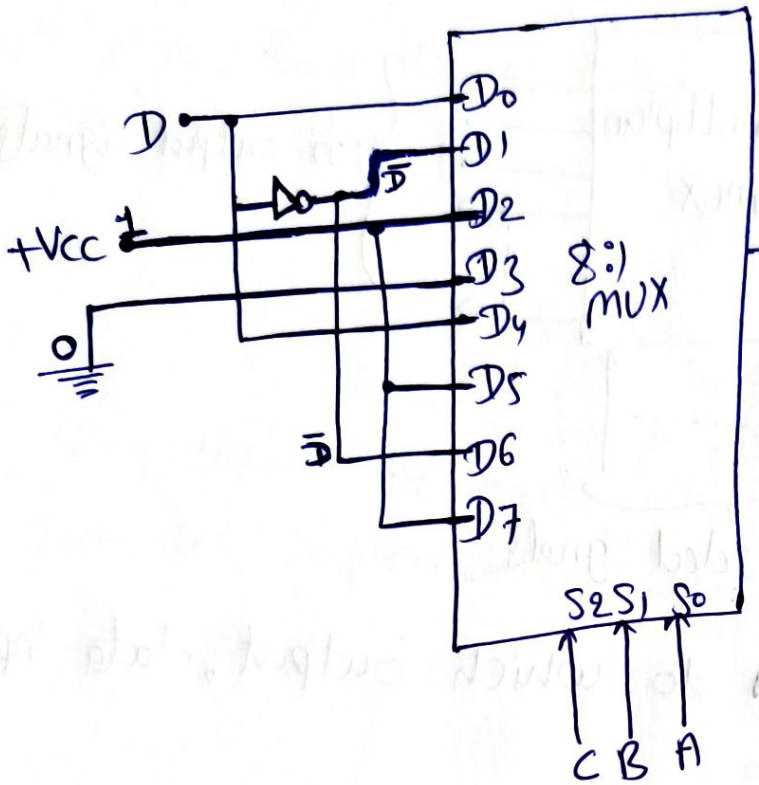
Input				Output
D	C	B	A	Y
0	0	0	0	0 ← D0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1 ← D15

Ans:- Let CBA (3 variable) be the select inputs. In first row of table CBA = 000 and therefor Y = D = 0.

- The single variable D is input & othe 3 variable CBA are n.
- 2^n input & n → select line ; 2^3 → 8 inputs MUX.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{D}$	0	(1)	(2)	3	4	(5)	(6)	(7)
D	(8)	9	(10)	11.	(12)	(13)	14	(15)
	D	$\bar{D}$	1	0	D	1	$\bar{D}$	1

Now, implementation of logic function using 8:1 MUX:-

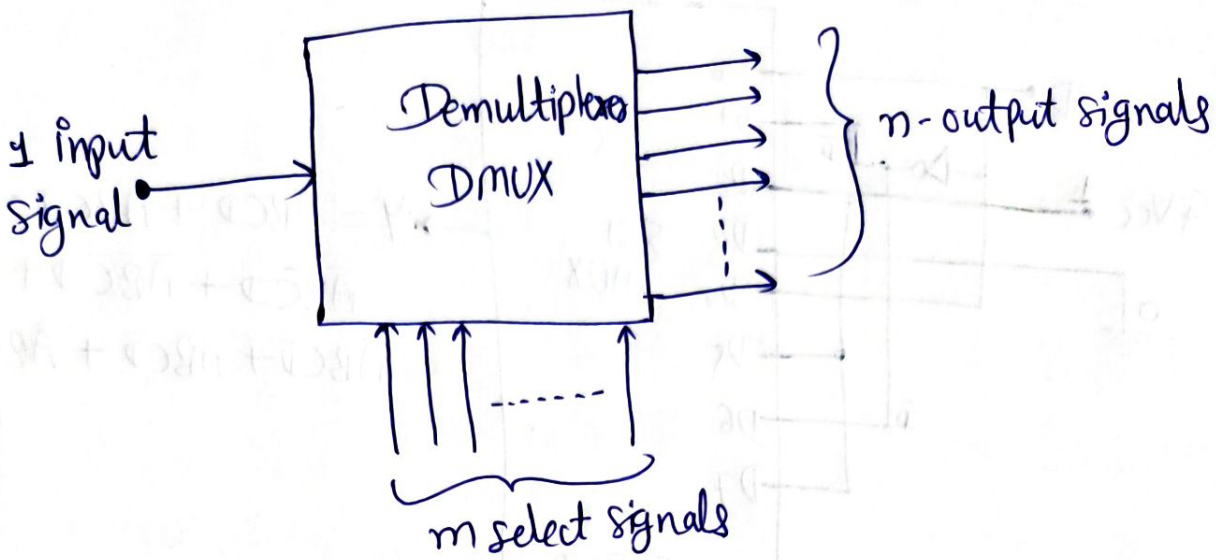


$Y = ABCD + A\bar{B}CD + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}B\bar{C}D + ABC\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D}$

DEMULTIPLEXER (Data Distributors):-

- The word 'demultiplex' means 'one into many'.
- Demultiplexing is the process of taking information from one input and transmitting the same over one of several outputs.
- A demultiplexer is a logic circuit that receives information on a single input and transmits the same information over one of several (2^m) output lines.

- Block diagram of demultiplexer:-



- Select inputs determine to which output, data input will be connected.

1: 1-to-4 Demultiplexer :-

- A 1 to 4 Demultiplexer has a single input D, four outputs (Y_0 to Y_3) and two select lines (S_1 and S_0).

Data input	Select lines		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

:- Truth table of 1 to 4 DMUX

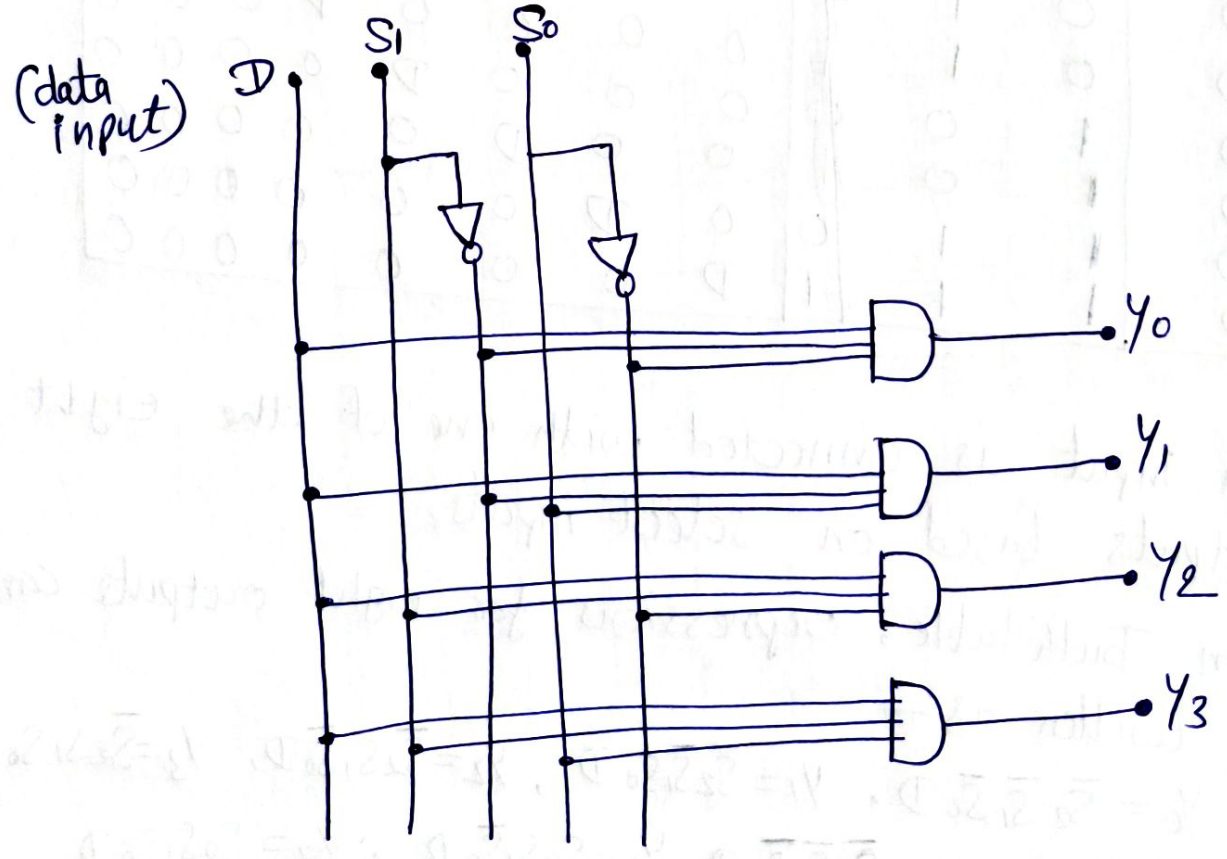
- when $S_1 = 0$ & $S_0 = 0$, the data input is connected to output Y_0.

- when $S_1=0$, $S_0=1$; the data input is connected to output Y_1. and So on.....
- Also, from truth table, expressions for outputs can be written as follows

$$Y_0 = \bar{S}_1 \bar{S}_0 D \quad ; \quad Y_2 = S_1 \bar{S}_0 D$$
$$Y_1 = \bar{S}_1 S_0 D \quad Y_3 = S_1 S_0 D$$

- Using these expressions, a 1 to 4 demultiplexer can be implemented using
  - 4 - three input AND gates.
  - 2 - Not gates.

- Logic diagram of 1 to 4 demux:-



[2] 1 to 8 Demultiplexer :-

- A 1 to 8 demultiplexer has a single output (D), eight outputs (Y_0 to Y_7) and three select inputs (S_2, S_1 & S_0).
- It distributes one input line to eight output lines based on the select inputs.
- Truth table of 1 to 8 demultiplexer:-

Data input	Select inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

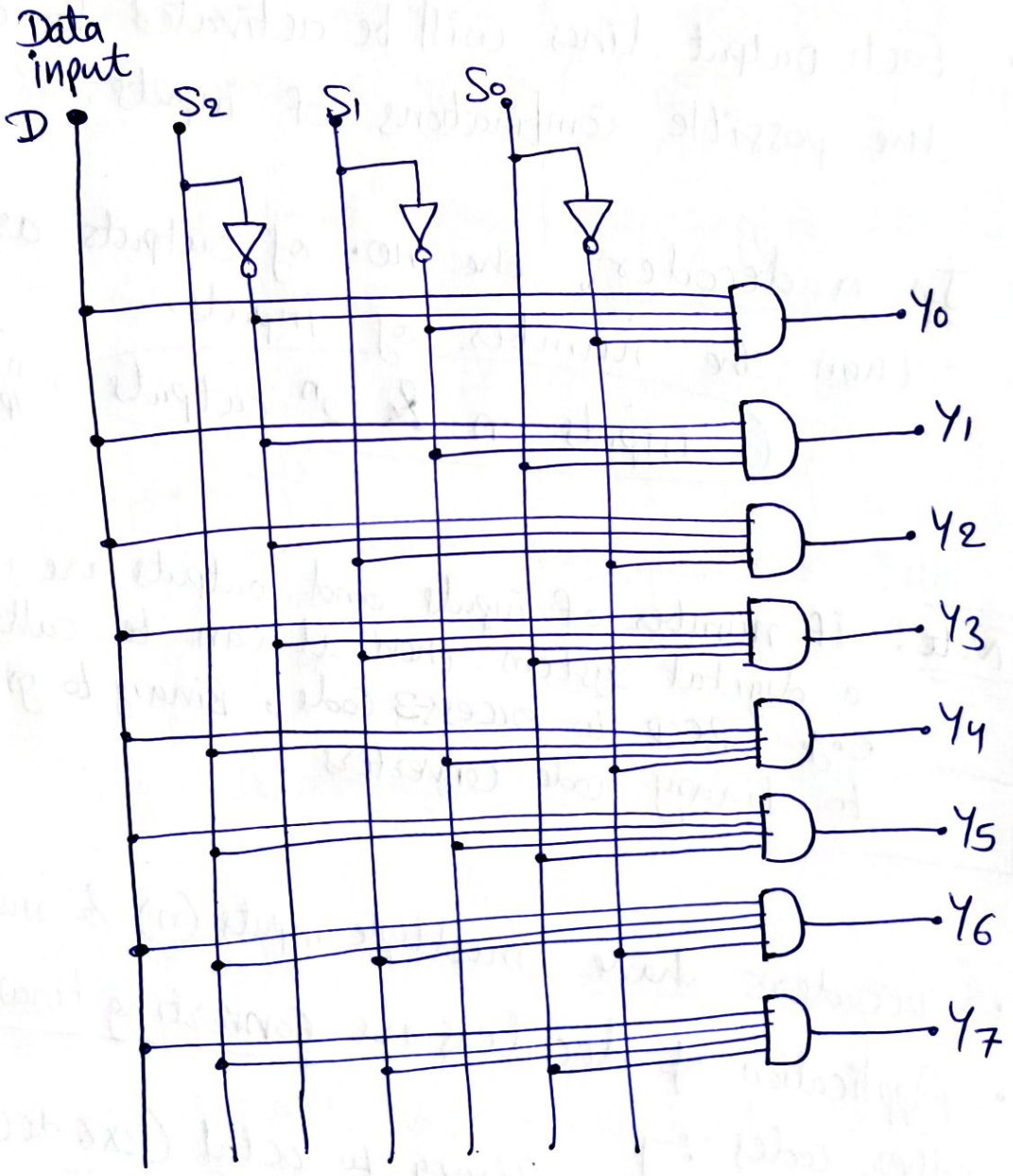
- Data input is connected with one of the eight outputs based on select inputs.
- From Truth table, expressions for eight outputs can be written as:-

$$Y_0 = \bar{S}_2 \bar{S}_1 \bar{S}_0 D,\ Y_1 = \bar{S}_2 \bar{S}_1 S_0 D,\ Y_2 = \bar{S}_2 S_1 \bar{S}_0 D,\ Y_3 = \bar{S}_2 S_1 S_0 D,$$
$$Y_4 = S_2 \bar{S}_1 \bar{S}_0 D,\ Y_5 = S_2 \bar{S}_1 S_0 D,\ Y_6 = S_2 S_1 \bar{S}_0 D,\ Y_7 = S_2 S_1 S_0 D$$

- Here, the single data input line D is connected to all the eight AND gates, but only one of eight AND gate will be enabled by the select input lines.

- For implementation 1 to 8 Demux, 3-NOT gates & four input 8-AND gates

- Logic diagram of 1 to 8 demultiplexer:—



1 to 16 Demultiplexer → Home work

DECODERS:- Digital systems require the decoding of data.

- Decoding is necessary in applications such as data multiplexing, digital display, digital to analog converter.
- A decoder is a logic circuit that converts an n-bit binary input code into 2^n output lines
- Each output lines will be activated for only one of the possible combinations of inputs.
- In a decoder, the no. of outputs are greater than the number of inputs.

inputs n & 2^n outputs

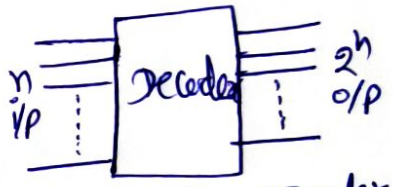


fig:- Block dia of Decoder

Note:- if number of inputs and outputs are equal in a digital system then it can be called converters. e.g. BCD to excess-3 code, Binary to gray & gray to binary code converters

- Decoders have multiple inputs (n) & multiple outputs (2^n).
- Application of decoders are converting binary code to other codes :-
  1) Binary to octal (3x8 decoder)
  2) Binary to Hexa-decimal (4x16 decoder)
  3) Binary/BCD to Decimal (4 x 10 decoder)

Basic Binary Decoder :-

- An AND gate can be used as the basic decoding element because its output is HIGH only when all the inputs are HIGH.
- If a NAND gate is used in place of the AND gate, then a LOW output is generated to indicate the presence of the proper binary code.

For example :- if input binary number is 1001, then to make all the input to AND gate HIGH, two middle bits (0) must be inverted by NOT Gate.

like :-

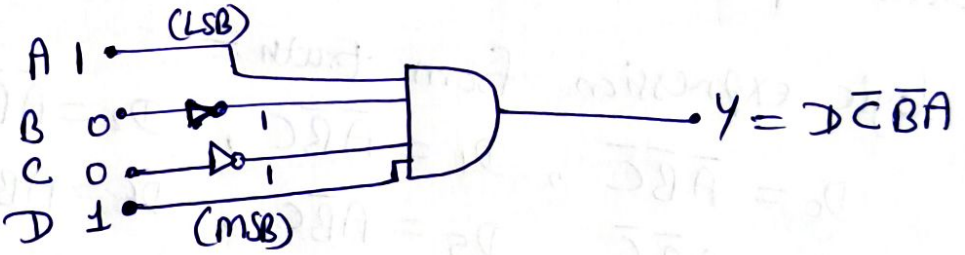


fig: Decoding logic for input 1001 with an active High O/P.

1. 3-to-8 Decoder :- or [1 to 8 Decoder]

- A 3 to 8 decoder has three inputs (A, B, C) & eight outputs (D_0 to D_7).
- Based on the 3 inputs, one of the eight outputs is selected.

- Truth table of 3 to 8 decoder:-

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

- From truth table, it is clear that only one of eight outputs (D_0 to D_7) is selected based on the three select inputs.

- logic expression from truth :-

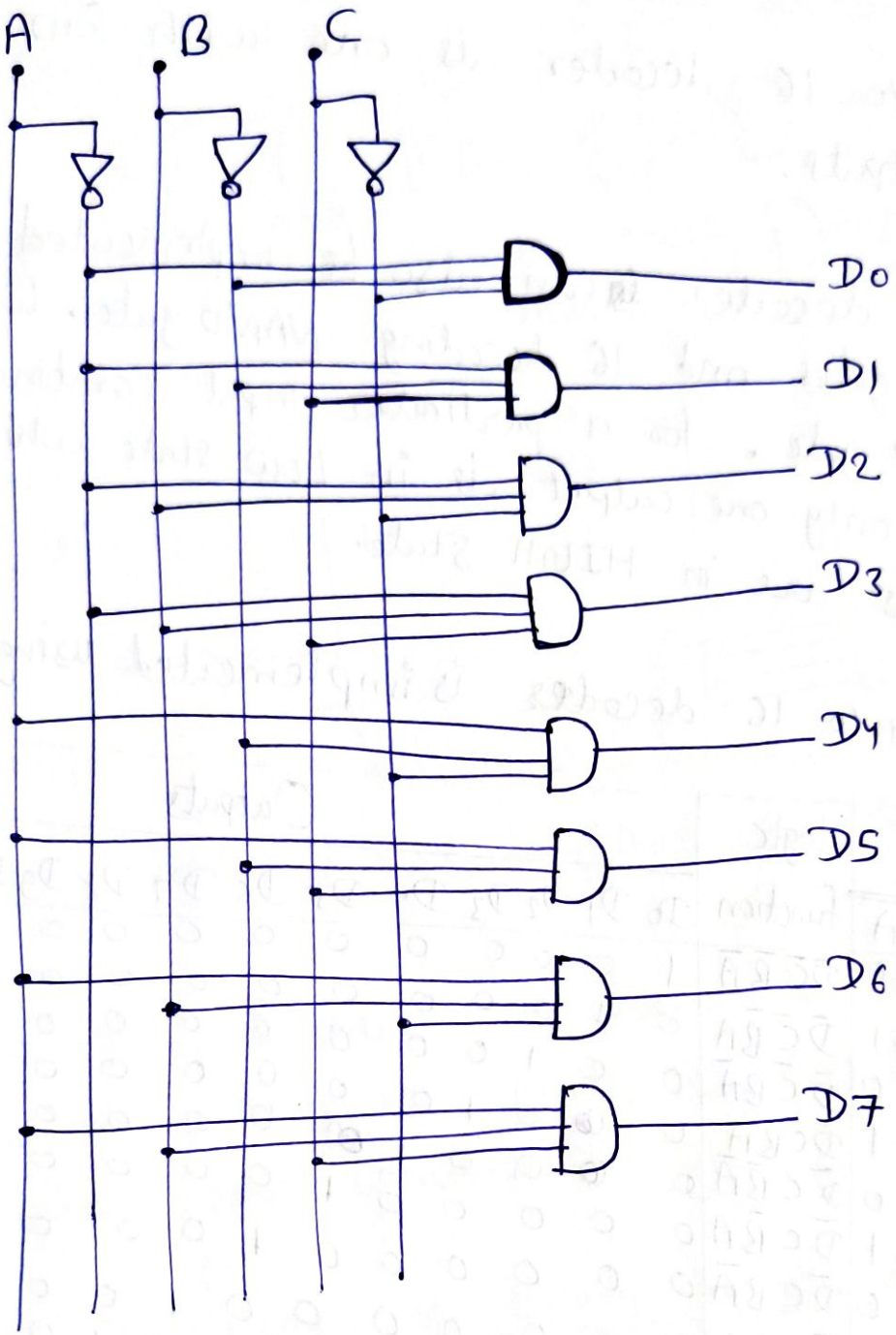
$D_0 = \bar{A}\bar{B}\bar{C}$, $D_1 = \bar{A}\bar{B}C$, $D_2 = \bar{A}B\bar{C}$, $D_3 = \bar{A}BC$

$D_4 = A\bar{B}\bar{C}$, $D_5 = A\bar{B}C$, $D_6 = AB\bar{C}$, $D_7 = ABC$

- Using these expression, 3 to 8 decoder can be implemented using
  - 3-NOT Gates.
  - eight 3-input AND gates.

- Input represent three bit binary numbers and outputs represent the eight digits in the octal number system, called binary-to-octal decoder.

- Logic diagram of 3 to 8 decoder:-



Note:- Some decoders have one or more enable inputs which are used to control the operation of decoder. With the enable line held HIGH, decoder functions normally & the input code ABC will determine which output is high. Hence, the decoder is enabled only if the enable line is HIGH.

[2] 4-to-16 Decoder :- or [1-to-16 Decoder]

- A 4 to 16 decoder is one with four inputs and 16 outputs.
- This decoder can also be implemented using four NOT gates and 16 decoding NAND gate. When we use NAND gate, for a particular input combination (in Truth table) only one output is in LOW state while remaining outputs are in HIGH State.
- Here, 4 to 16 decoder is implemented using AND gate.

Decimal digits	Binary Inputs D C B A	Logic function	Outputs D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 D_9 D_{10} D_{11} D_{12} D_{13} D_{14} D_{15}
0	0 0 0 0	$\bar{D}\bar{C}\bar{B}\bar{A}$	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1	0 0 0 1	$\bar{D}\bar{C}\bar{B}A$	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 1 0	$\bar{D}\bar{C}B\bar{A}$	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
3	0 0 1 1	$\bar{D}\bar{C}BA$	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
4	0 1 0 0	$\bar{D}C\bar{B}\bar{A}$	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
5	0 1 0 1	$\bar{D}C\bar{B}A$	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
6	0 1 1 0	$\bar{D}CB\bar{A}$	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
7	0 1 1 1	$\bar{D}CBA$	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8	1 0 0 0	$D\bar{C}\bar{B}\bar{A}$	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
9	1 0 0 1	$D\bar{C}\bar{B}A$	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
10	1 0 1 0	$D\bar{C}B\bar{A}$	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
11	1 0 1 1	$D\bar{C}BA$	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
12	1 1 0 0	$DC\bar{B}\bar{A}$	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
13	1 1 0 1	$DC\bar{B}A$	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
14	1 1 1 0	$DCB\bar{A}$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
15	1 1 1 1	$DCBA$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

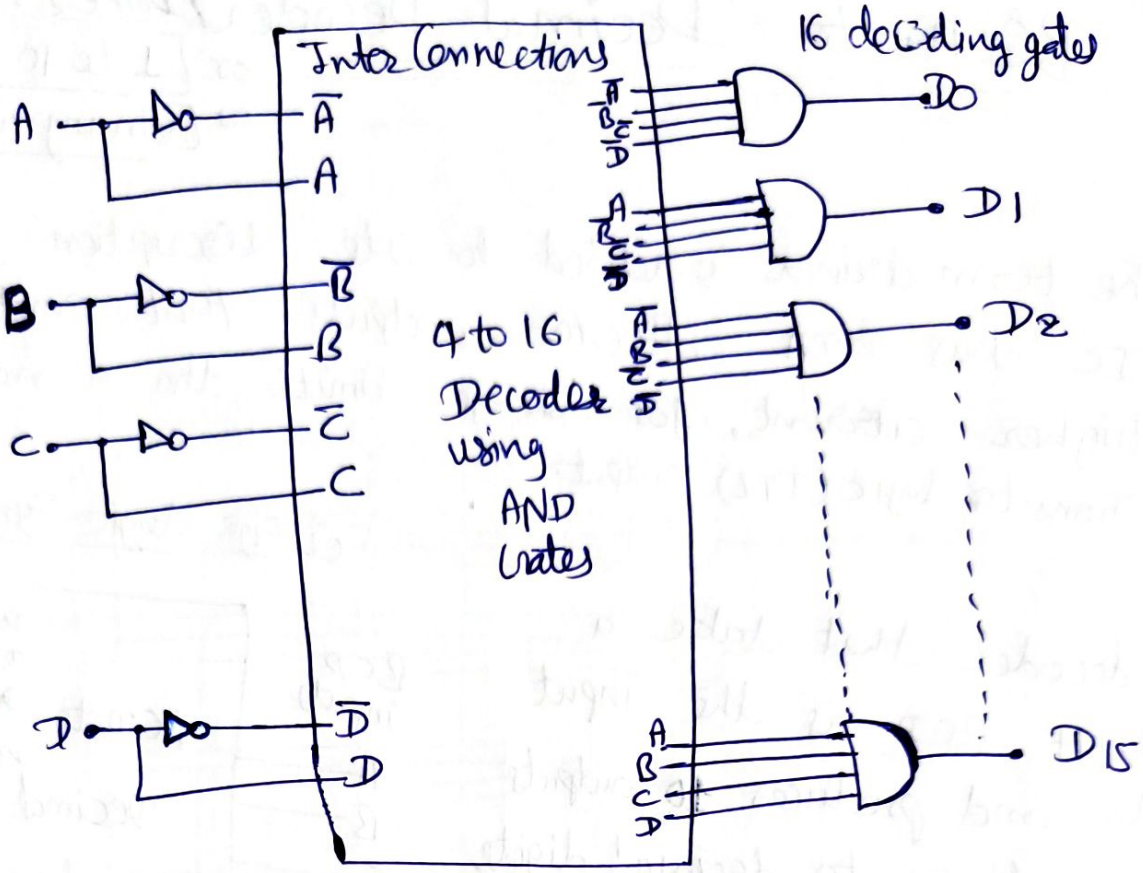


Fig: logic diagram of 4-to-16 Decoder

- The 4 to 16 decoder is also called a 1-to-16 decoder since only one of 16 outputs is selected based on a particular input combination.

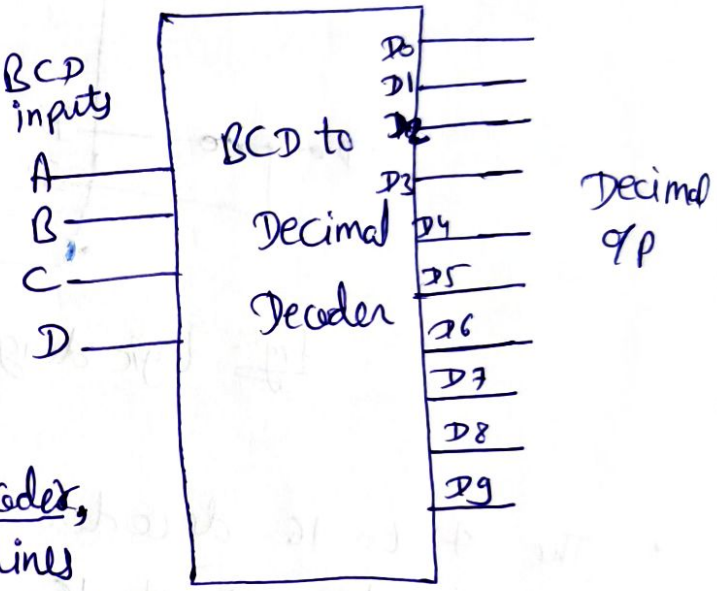
- It is also called as binary to hexa-decimal decoder since the inputs represents four-bit binary number and outputs represents the sixteen digits in the hexadecimal no. system.

3] # BCD to Decimal Decoder/Driver :-

or [1 to 10 Decoder]
or [Binary to decimal Decoder]

- The term driver is added to its description because this IC has been collector outputs that can operate at higher current, and voltage limits tha a normal Transistor Transitor logic (TTL) output.

Fig: logic Symbol



- A decoder that take a 4-bit BCD as the input code and produces 10 outputs corresponding to decimal digits, is called a BCD to Decimal Decoder.

- This is also called a 1 to 10 decoder, since only one of eight output lines is HIGH for a particular input combination.

- Truth table:-

Decimal Digits	inputs D	C	B	A	Logic function	Outputs D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	0	$\bar{D}\bar{C}\bar{B}\bar{A}$	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	$\bar{D}\bar{C}\bar{B}A$	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	$\bar{D}\bar{C}B\bar{A}$	0	0	1	0	0	0	0	0	0	0
3	0	0	1	1	$\bar{D}\bar{C}BA$	0	0	0	1	0	0	0	0	0	0
4	0	1	0	0	$\bar{D}C\bar{B}\bar{A}$	0	0	0	0	1	0	0	0	0	0
5	0	1	0	1	$\bar{D}C\bar{B}A$	0	0	0	0	0	1	0	0	0	0
6	0	1	1	0	$\bar{D}CB\bar{A}$	0	0	0	0	0	0	1	0	0	0
7	0	1	1	1	$\bar{D}CBA$	0	0	0	0	0	0	0	1	0	0
8	1	0	0	0	$D\bar{C}\bar{B}\bar{A}$	0	0	0	0	0	0	0	0	1	0
9	1	0	0	1	$D\bar{C}\bar{B}A$	0	0	0	0	0	0	0	0	0	1

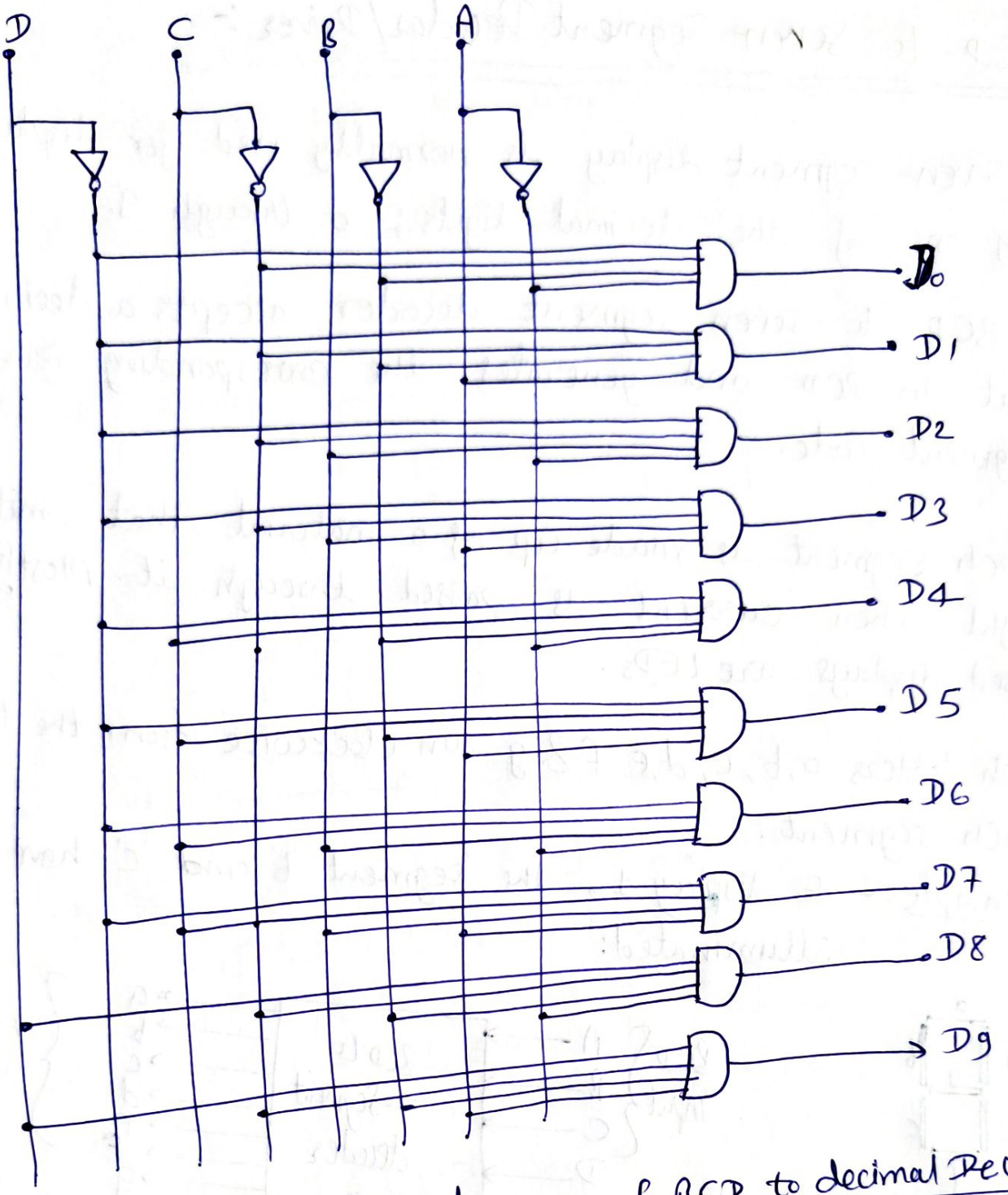


fig:- Logic diagram of BCD to decimal Decoder

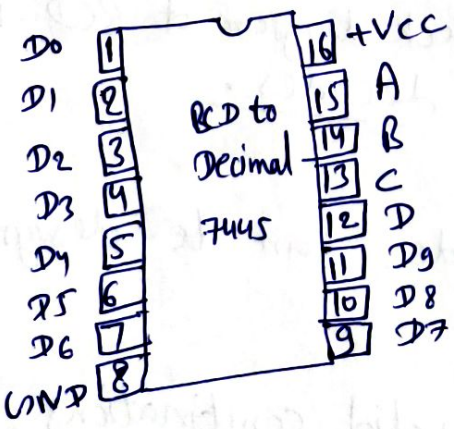
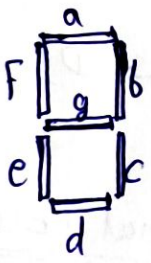


fig:- Pinout diagram of BCD to decimal Decoder.

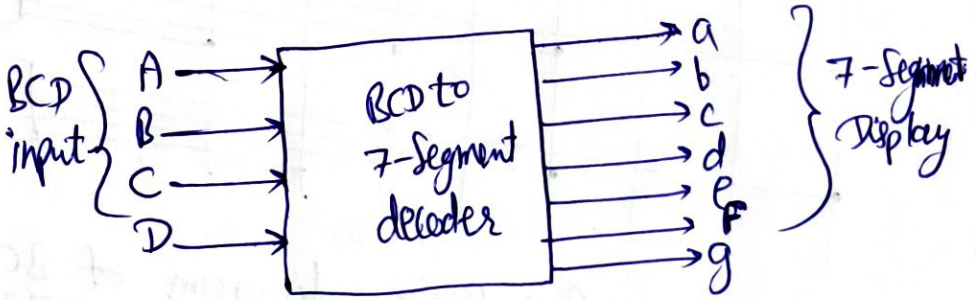
*Imp.

BCD-to-Seven-Segment Decoder/Driver :-

- A seven-segment display is normally used for displaying any one of the decimal digits, 0 through 9.
- A BCD-to-seven segment decoder accepts a decimal digit in BCD and generates the corresponding seven-segment code.
- Each segment is made up of a material that emits light when current is passed through it. Mostly used displays are LEDs.
- Each letters a, b, c, d, e, f & g run clockwise from the top of each segment.
- Example:- To display 1, the segment 'b' and 'c' have to be illuminated:



Fig(a). Seven-segment display



Fig(b): Block diagram of BCD-to-7 Segment decoder.

- A BCD-to-7 segment decoder can be designed using logic gates.

* Since only BCD inputs are valid combinations, the other input combination of four variables corresponding to

to 10, 11, 12, 13, 14, & 15 can be termed as **don't care** Combinations to aid the simplification of logic expressions.

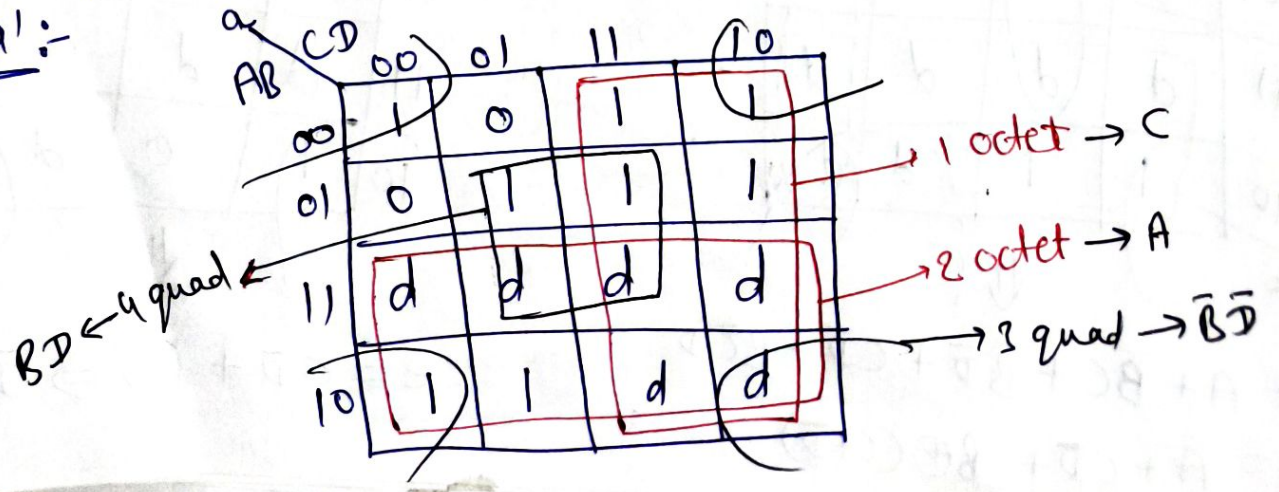
- Truth table of BCD-to-7 segment decoder :- 123456789

	BCD input				Seven segment Outputs						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	0	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

$$a = \Sigma_m(0,2,3,5,6,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$b = \Sigma_m(0,1,2,3,4,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$c = \Sigma_m(0,1,3,4,5,6,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$d = \Sigma_m(0,2,3,5,6,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$e = \Sigma_m(0,2,6,8) + \Sigma_d(10,11,12,13,14,15)$$
$$f = \Sigma_m(0,4,5,6,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$g = \Sigma_m(2,3,4,5,6,8,9) + \Sigma_d(10,11,12,13,14,15)$$

→ these expression are solved using k-map :——

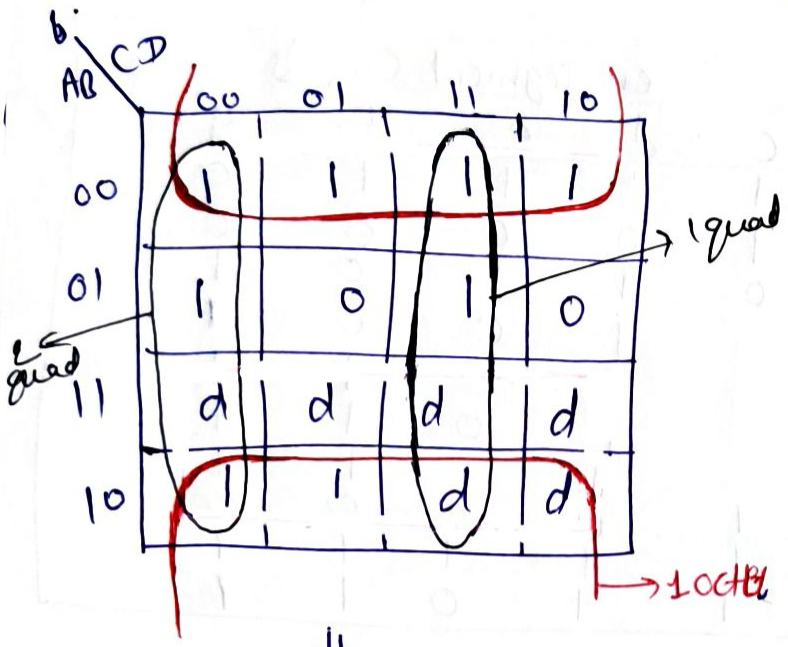
For 'a' :-



$$a = A + C + \overline{B}\,\overline{D} + BD \quad \rightarrow \text{EX-NOR}$$

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

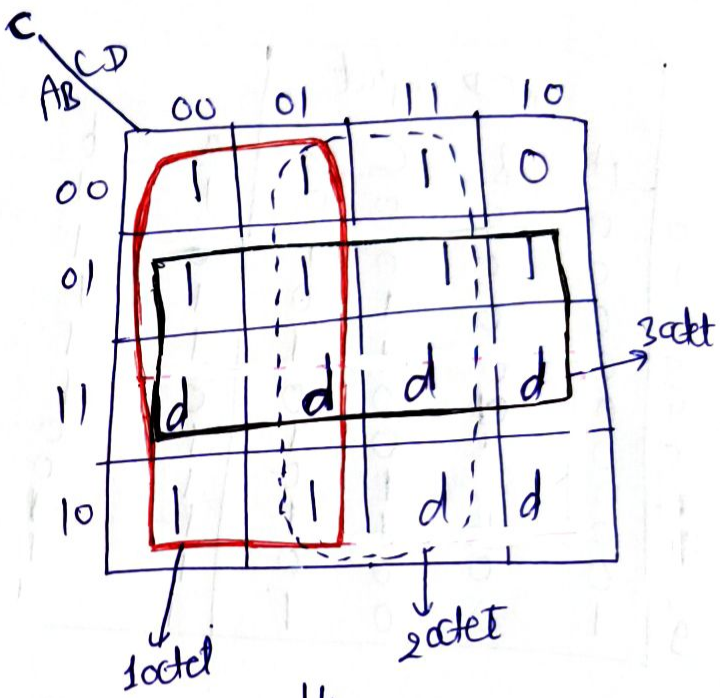
2) for 'b':-



$$b = \overline{B} + \overline{C}\,\overline{D} + CD$$

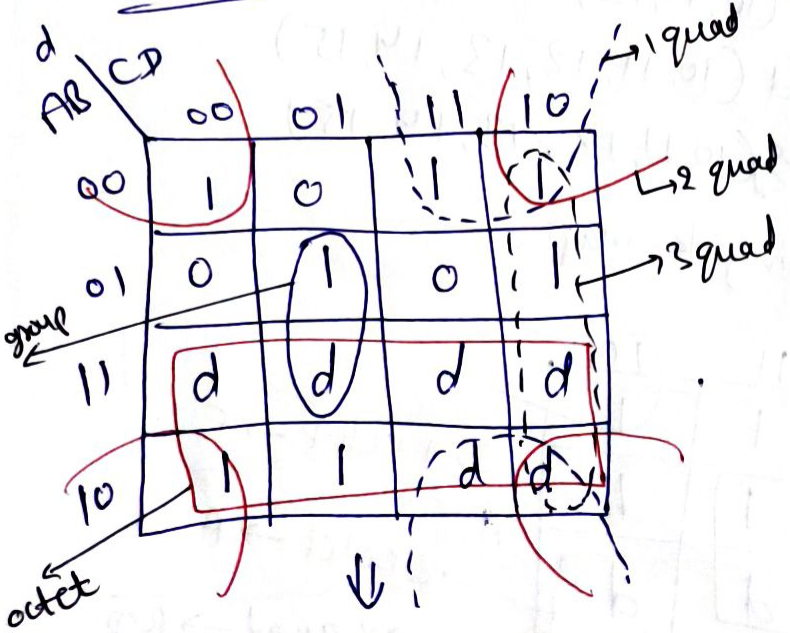
$$b = \overline{B} + \overline{C \oplus D}$$

3) for 'C':-



$$C = \overline{C} + D + B$$

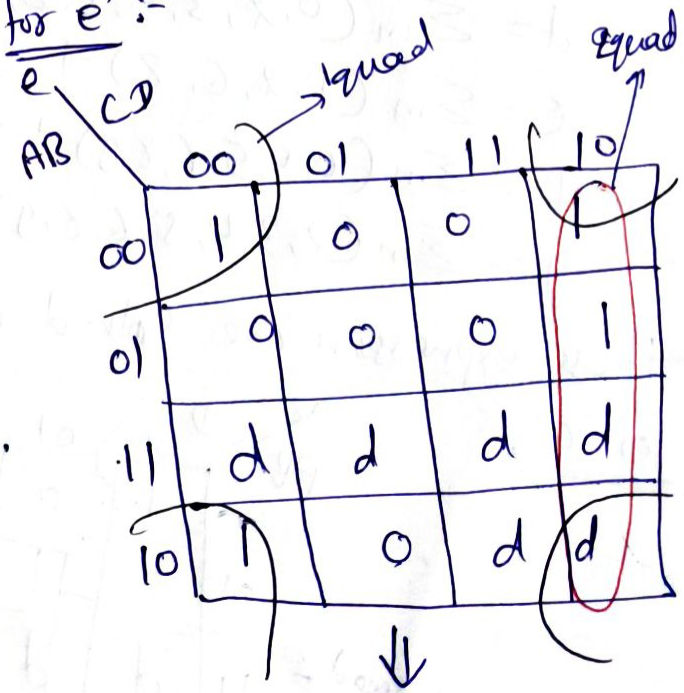
4) for 'd':-



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

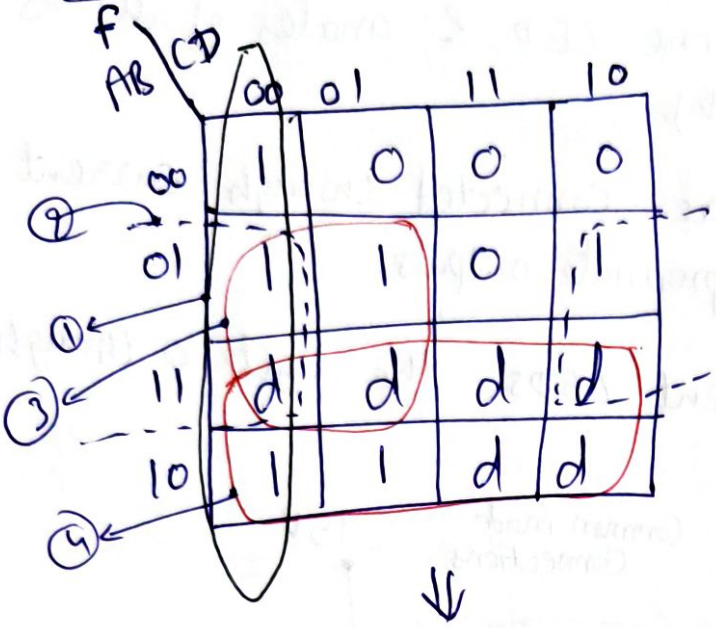
$$d = A + C\overline{D} + B \oplus (C + \overline{D})$$

5) for 'e':-



$$e = C\overline{D} + \overline{B}\,\overline{D} \Rightarrow \overline{D}(\overline{B} + C)$$

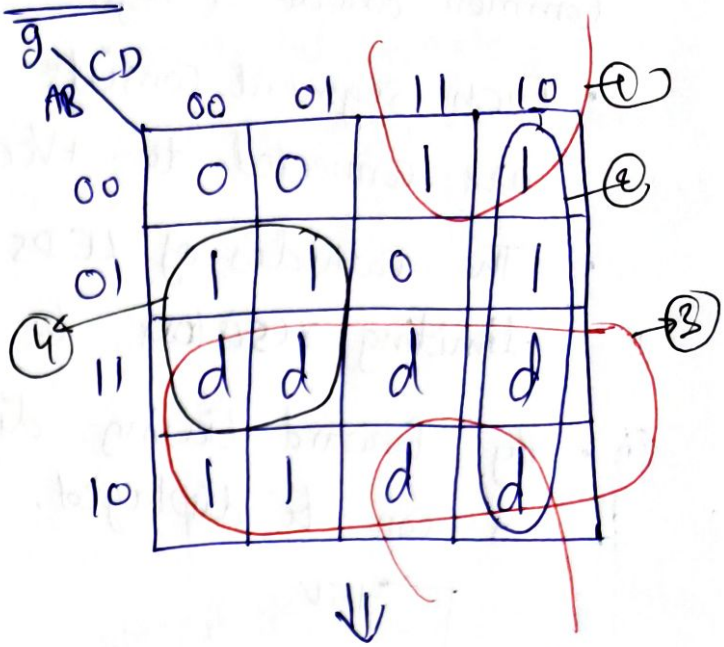
for 'f':-



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

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

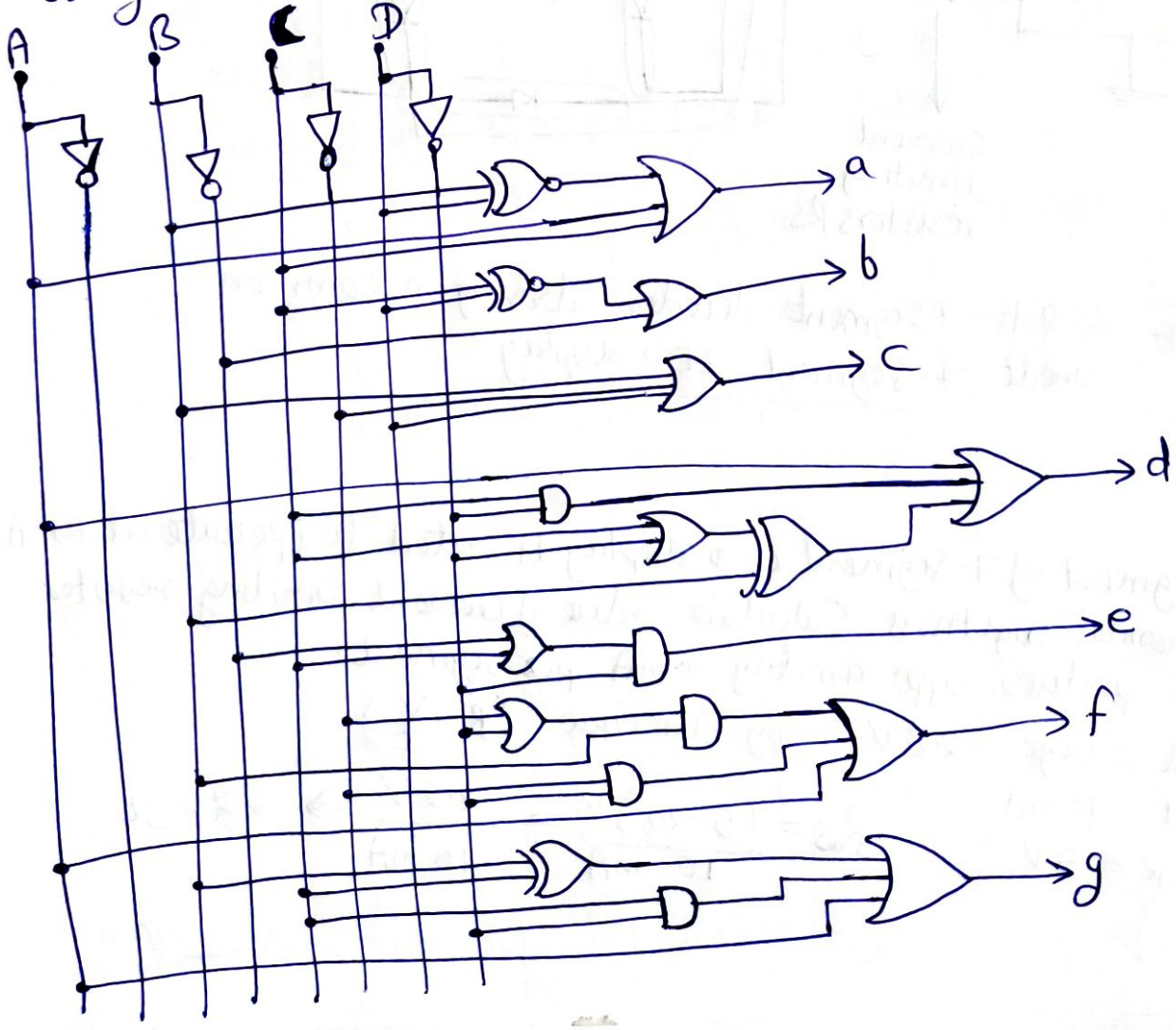
for 'g':-



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

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

- Using above expressions, BCD-to-7 segment decoder implemented:-



* BCD-to-7 segment decoder/driver used to drive a common anode 7-segment LED display.

- Each segment consists of one LED & anodes of all LEDs are connected to +Vcc (5V).
- The cathodes of LEDs are connected through current limiting resistors to appropriate outputs.
- By forward biasing different LEDs, the digits 0 through 9 can be displayed.

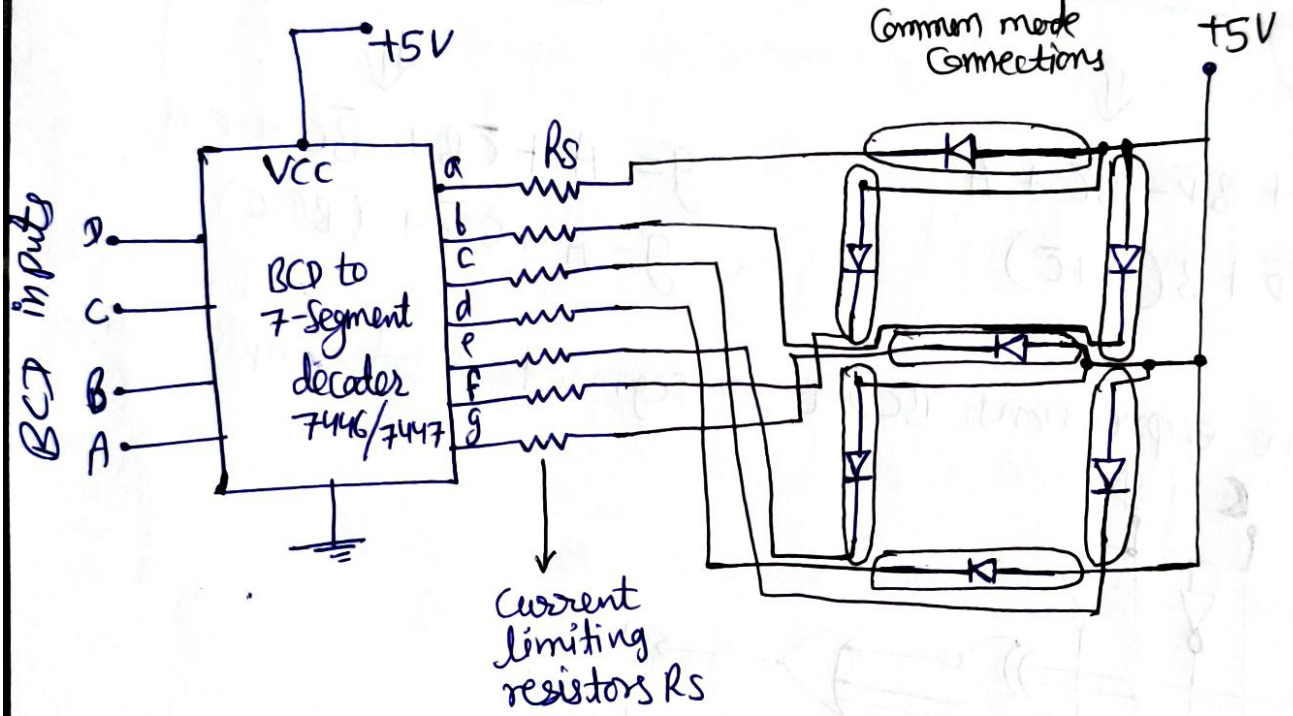


fig:- BCD-to-7 segment decoder driving a common anode 7-segment LED display.

Q1: Each segment of 7-segment LED display is rated to operate at 10mA & 2.8 V for normal brightness. Calculate value of current limiting resistor needed to produce approximately 10mA per segment.

Ans: Segment voltage = 2.8 V ; By ohm's law ($R = \frac{V}{I}$)

Current = 10 mA

Applied voltage = 5 V

$$R_S = \frac{(5-2.8)\,V}{10\ mA} = \frac{2.2\,V}{10\ mA} \Rightarrow 220\ \Omega$$

Ans

Application of decoders:-

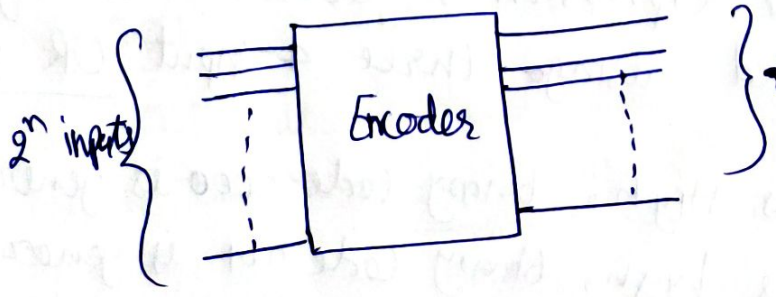
1. Decoders are used in counter systems.
2. They are used in analog-to-digital converters.
3. Decoder outputs can be used to drive a display system.

ENCODERS:-

- An encoder is a digital circuit that performs the inverse operation of a decoder.
- An encoder is a combinational logic circuit that converts an active input signal into a coded output signal.
- It has 2^n input lines, only one of which is active at any time & n output lines.
- In encoder, the number of outputs are less than the number of inputs.

in encoder → input 2^n & n output lines

- Block diagram of encoder:-



1 Octal-to-Binary Encoder :- (8-to-3 Encoder)

- 8-to-3 encoder performs the opposite function; it accepts eight inputs and produces a 3-bit output code corresponding to the activated input.

truth table :-

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

- From truth table, output Y_0 must be 1, when inputs D_1 OR D_3 OR D_5 OR D_7 will be HIGH.

$$Y_0 = D_1 + D_3 + D_5 + D_7$$
$$Y_1 = D_2 + D_3 + D_6 + D_7$$
$$Y_2 = D_4 + D_5 + D_6 + D_7$$

- Using these expression, octal to binary encoder can be implemented using three 4-input OR gates.

- when D_0 is High, binary code 000 is generated.
- when D_1 is high, binary code 001 is generated & so on..

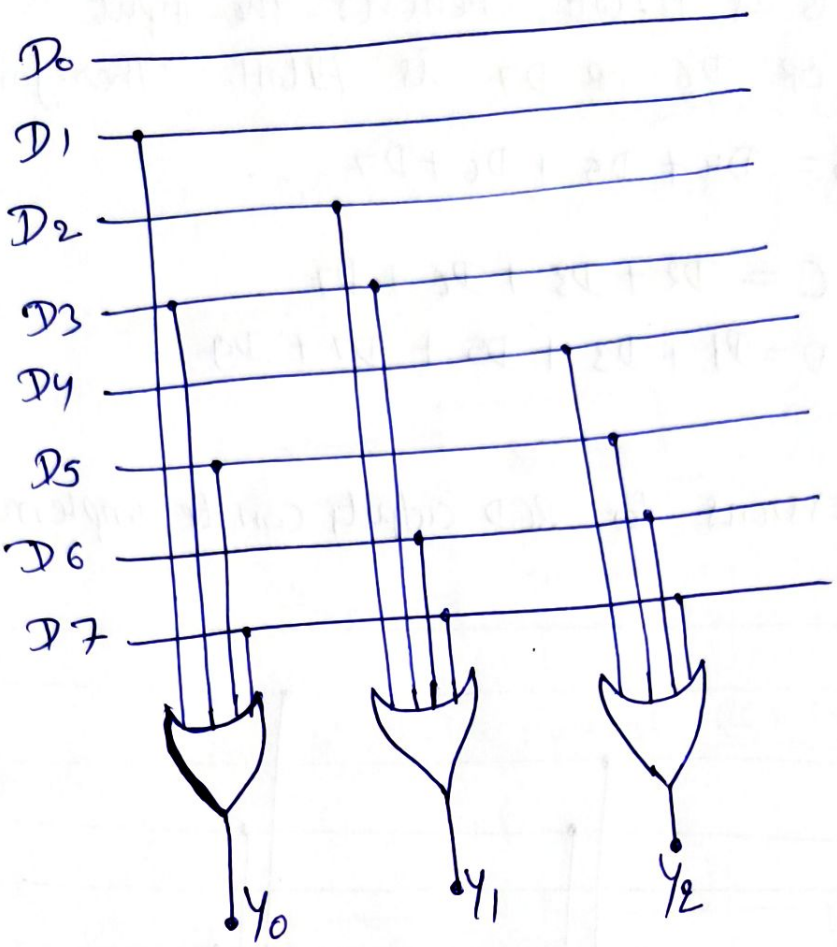


fig:- Octal-to-Binary Encoder

2] Decimal to BCD Encoder:- (10-to-4 encoder or 10 to 1 encoder)

- A decimal-to-BCD encoder is one with ten inputs corresponding to ten decimal digits (0 to 9) and four ouput (A, B, C, D) representing the BCD value of i/p decimal digit.

- Truth table (T.T.):-
  - From T.T., it is clear that the output A is HIGH whenever the input Y_8 OR Y_9 is HIGH.
    therefore $A = Y_8 + Y_9$

- The output B is HIGH, whenever the input D4 OR D5 OR D6 OR D7 is HIGH. Therefore

$$B = D_4 + D_5 + D_6 + D_7$$

Similarly, $C = D_2 + D_3 + D_6 + D_7$

$$D = D_1 + D_3 + D_5 + D_7 + D_9$$

- These expressions for BCD outputs can be implemented:-

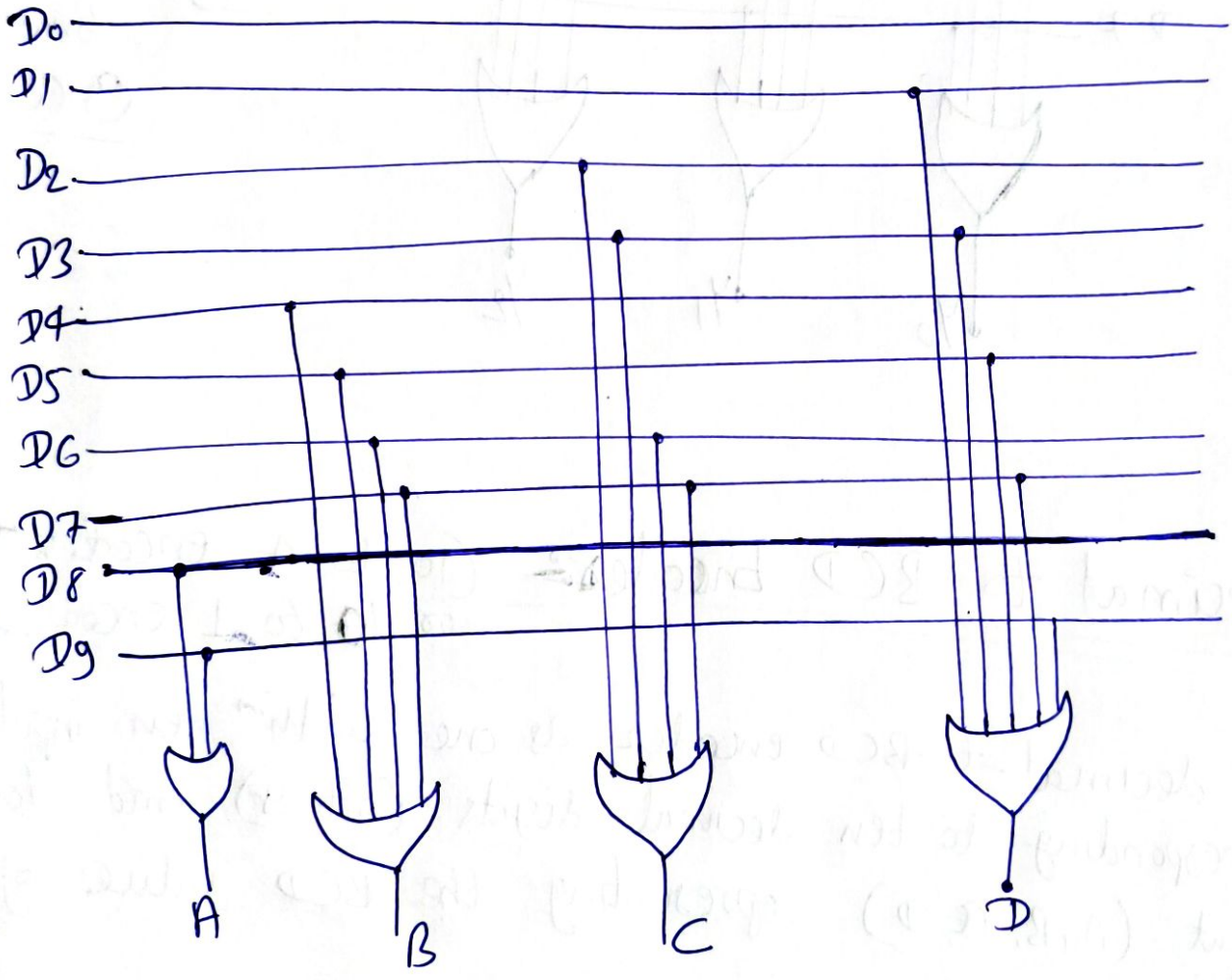


Fig:- Decimal to BCD encoder

BCD-to-Excess-3 Encoder :-

- The BCD-to-Excess-3 encoder converts 4-bit BCD input into 4 bit excess-3 code.

- Block diagram :-

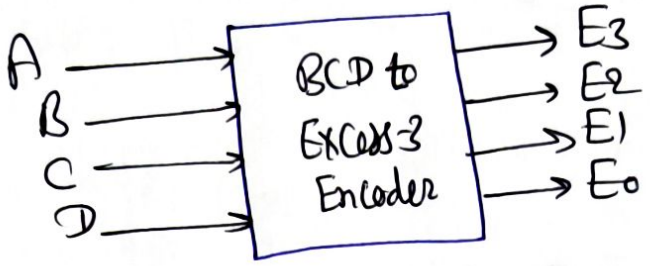


fig :- Block diagram of BCD to Excess-3 encoder.

- Truth Table :-

	Inputs				Outputs			
	A	B	C	D	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
Invalid BCD no. put 'd'	1	0	1	0	d	d	d	d
10	1	0	1	1	d	d	d	d
11	1	1	0	0	d	d	d	d
12	1	1	0	1	d	d	d	d
13	1	1	1	0	d	d	d	d
14	1	1	1	1	d	d	d	d
15								

Add [+3] in BCD no. (0011)

At the place of invalid BCD no. put don't care 'd' condition.

And solve K-map for Excess-3 outputs.

- According to the truth table using K-map simplification the boolean expressions for output are:-

$E_3 = A + BD + BC \longrightarrow A + B(C+D)$

$E_2 = \bar{B}D + \bar{B}C + B\bar{C}\bar{D} \longrightarrow \bar{B}(C+D) + B\bar{C}\bar{D} \Rightarrow \bar{B}(\overline{C+D}) + B\overline{C+D}$

$\Rightarrow \overline{B \oplus (C+D)}$

$E_1 = \bar{C}\bar{D} + CD \longrightarrow \overline{C \oplus D}$

$E_0 = \bar{D}$

- Logic Digram:-

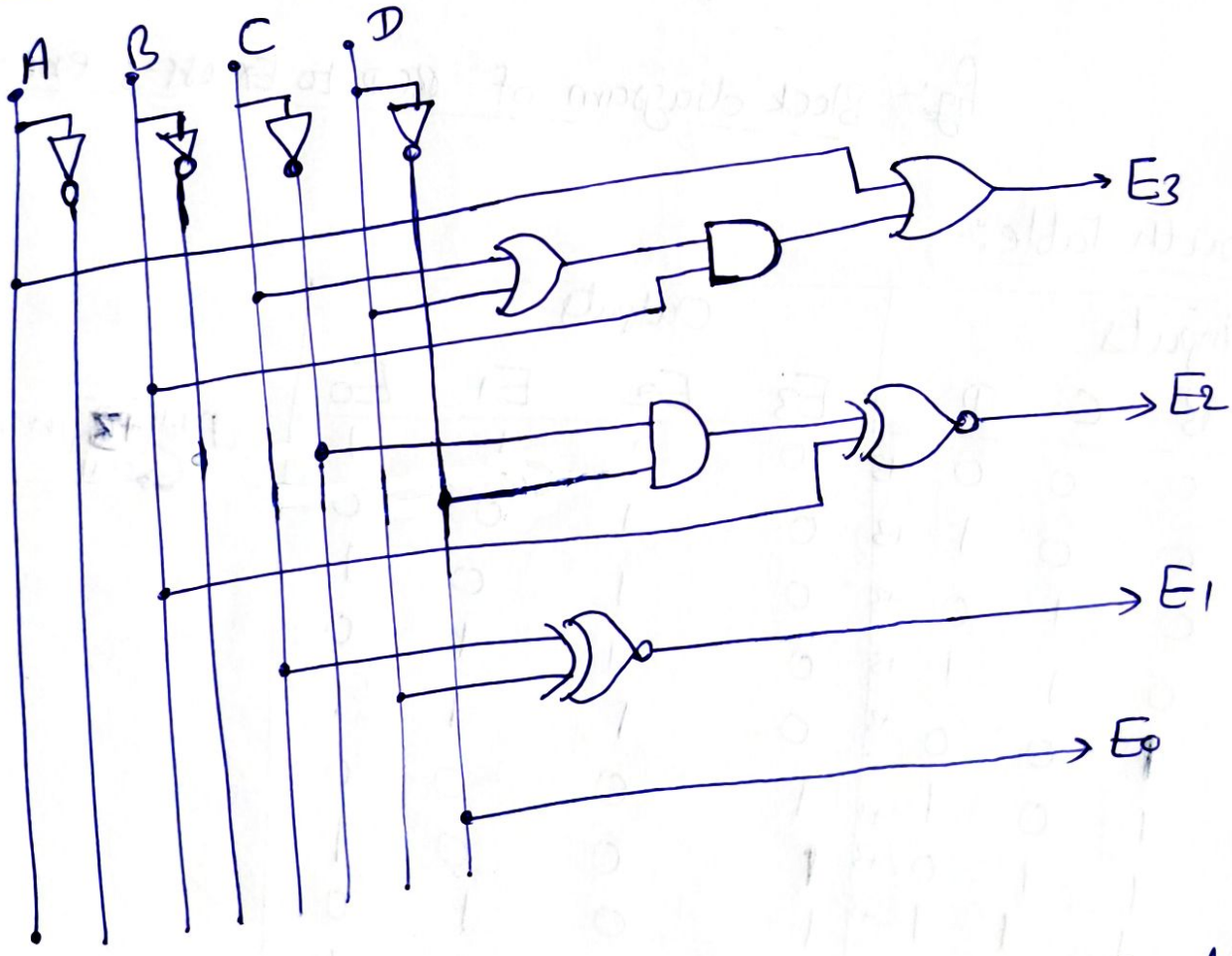


fig:- Logic Diagram of BCD to excess-3 encoder.

Binary-to-Gray Code Converters:-

- It has four input ($B_3 B_2 B_1 B_0$) representing 4-bit binary numbers and 4-outputs ($G_3 G_2 G_1 G_0$) representing 4-bit gray code.

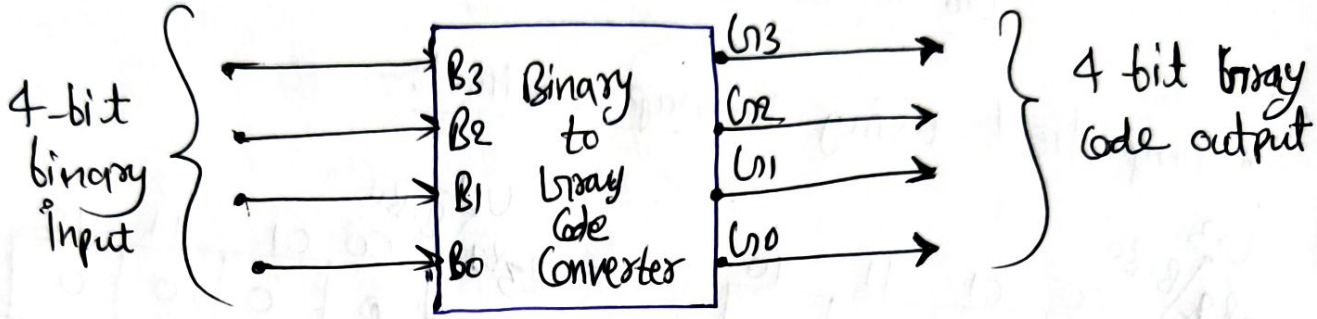


Fig:- 4-bit binary to gray code converter

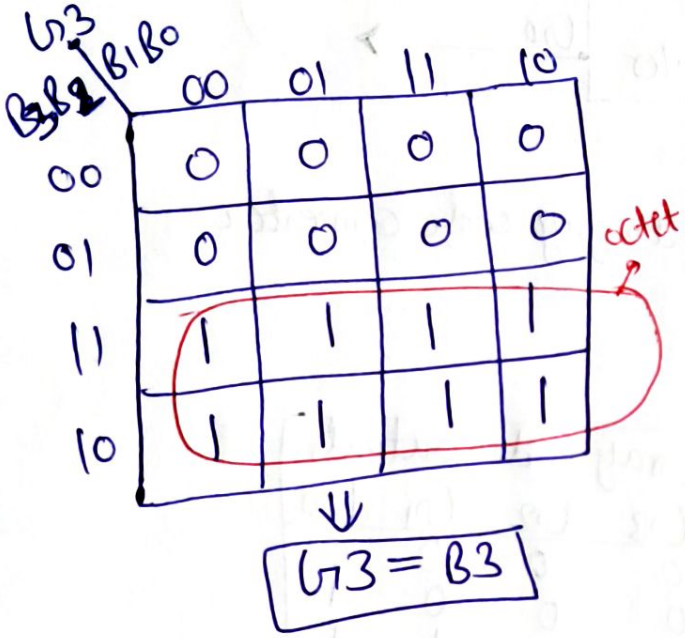
- Truth table:-

Binary inputs				Gray code outputs			
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

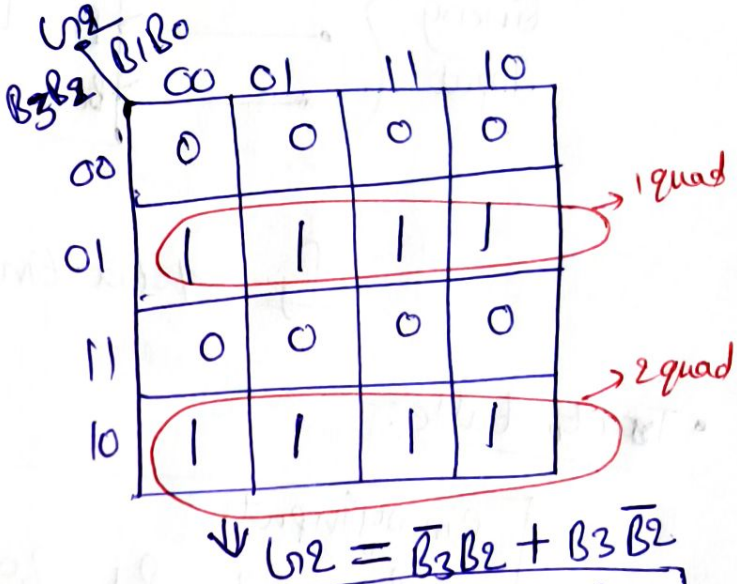
- From Truth table; logical expressions for gray code outputs can be written as:-

$$G_3 = \sum_m (8,9,10,11,12,13,14,15)$$
$$G_2 = \sum_m (4,5,6,7,8,9,10,11)$$
$$G_1 = \sum_m (2,3,4,5,10,11,12,13)$$
$$G_0 = \sum_m (1,2,5,6,9,10,13,14)$$

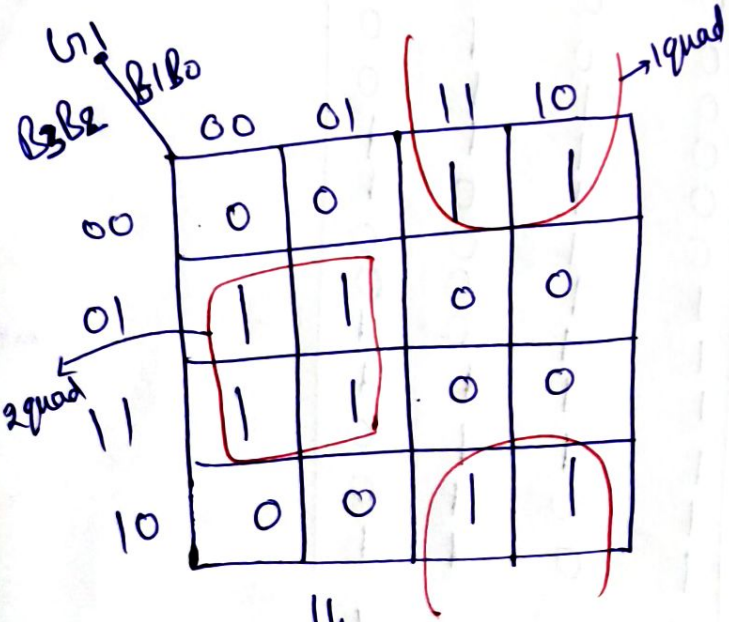
- Simplified using k-map method:-



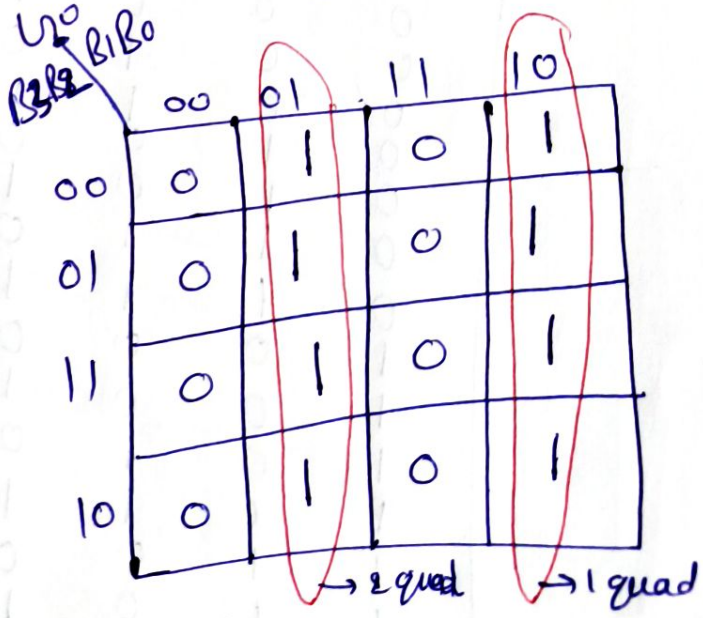
$$G_3 = B_3$$



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



$$G_1 = B_2 \bar{B}_1 + \bar{B}_2 B_1$$
$$G_2 = B_1 \oplus B_2$$



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

1130
Finish

- Now, above G_3, G_2, G_1 & G_0 expressions can be implemented using Ex-OR gates:-

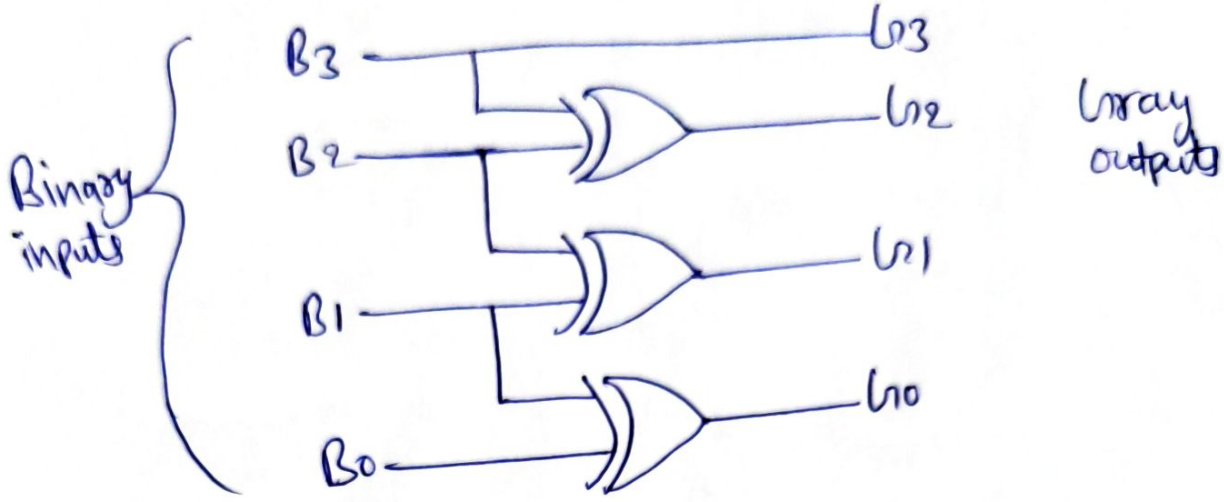
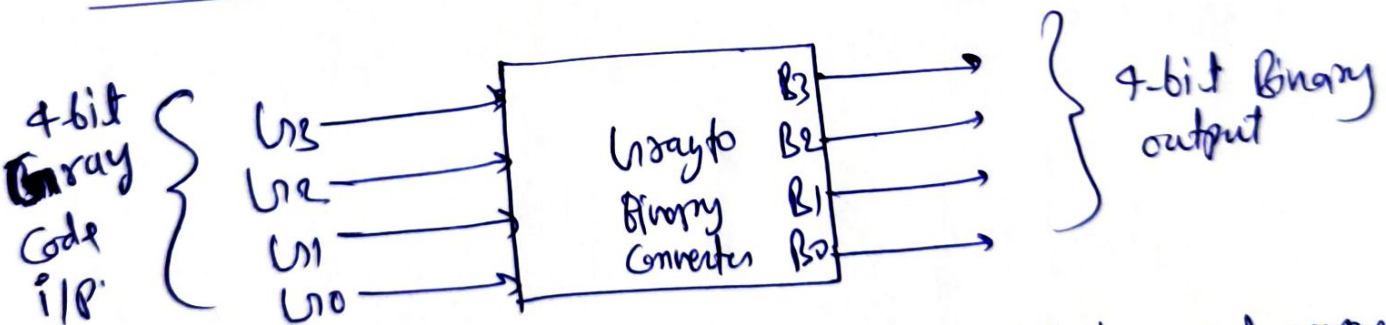


fig:- logic diagram of 4-bit binary-to-gray code converter.

Gray to Binary Code Converter:-

Block diagram of G → B



Truth table:- Same as (B → G) → opposite and simplify k-map for expressions.

① $B_3 = G_3$
② $B_2 = B_3 \oplus G_2$
③ $B_1 = B_2 \oplus G_1$
④ $B_0 = G_0 \oplus B_1$

⇒

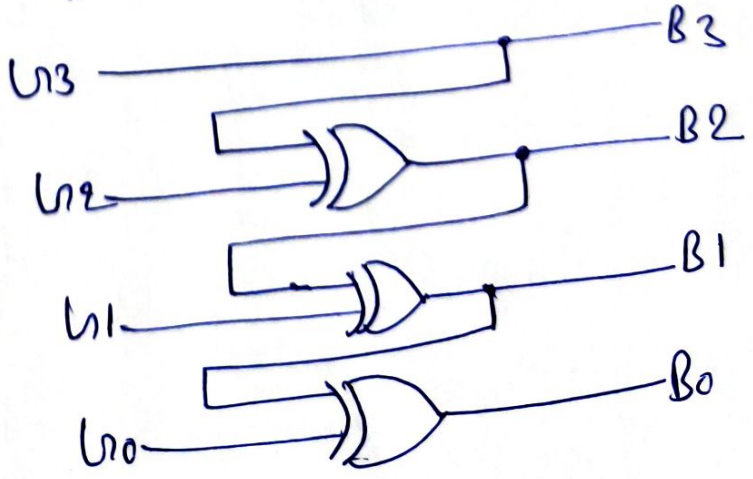


fig: logic diagram of 4-bit gray to binary code converter