⇒ Standard representation for Logic Expression

Logic expression → SOP → $y = AB + BC + CA$
→ POS → $y = (A+B)\cdot(\bar{A}+\bar{B})\cdot(\bar{A}+B)$

Standard form of SOP form	Non standard form of SOP form
$y = ABC + AB\bar{C} + \bar{A}\bar{B}\bar{C}$	$y = A\bar{B} + BC + A\bar{C}$

SOP is also known as Minterms (m)
POS is ——— Maxterms (M)

Non standard ⟶ Standard for SOP

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

Non stand ⟶ standard for POS

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

Minterm Each individual term in the Standard SOP form is called as minterm.

Maxterm Each individual terms in the standard POS form is called Maxterm.

Minterm 0 → —
Max 1 → =

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

Ex

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

$Y = m_7 + m_3 + m_4$

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

Ex

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

$Y = M_2 \; M_0 \; M_6$

$Y = \underbrace{\prod M(0,2,6)}_{POS}$

Ex

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Sol Consider only those combinations of inputs which correspond to $Y=1$

0	1	1	$Y_1 = \bar{A}B$
1	0	1	$Y_2 = A\bar{B}$

$Y = Y_1 + Y_2$

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

$= m_1 + m_2$

$Y = \sum m(1,2)$

⇒ Write a std POS Exp for given Truth table

A	B	C	Y	
0	0	0	0	$A+B+C$ M_0
0	0	1	1	
0	1	0	1	
0	1	1	0	$A+\bar{B}+\bar{C}$ M_3
1	0	0	1	
1	0	1	0	$\bar{A}+B+\bar{C}$ M_5
1	1	0	0	$\bar{A}+\bar{B}+C$ M_6
1	1	1	1	

Write the max term for combination of input produce Y = 0

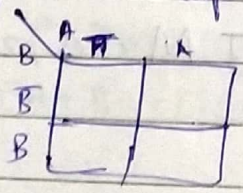
$Y = (A+B+C)(A+\bar{B}+\bar{C})(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$

$= M_0 M_3 M_5 M_6$

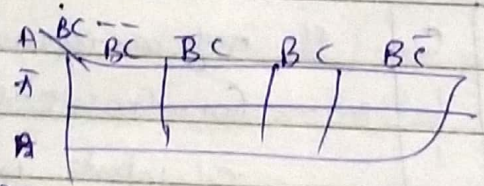
$Y = \prod M(0,3,5,6)$

⇒ Karnaugh - Map

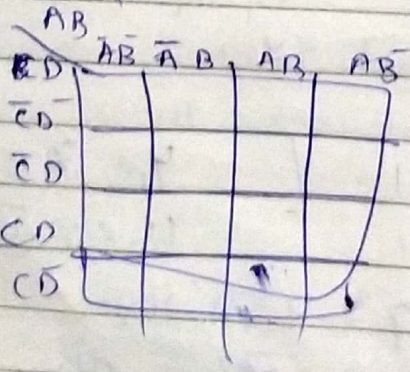
2 variable



3 Variable



4 Variable



2. On top of that, sometimes we are not sure whether the further simplification is possible or not. That means whether we have really obtained the minimized expression or not.

3.6 Karnaugh-Map Simplification :

- This is another simplification technique to reduce the Boolean equation.
- It overcomes all the disadvantages of the algebraic simplification technique.
- K-map (short form of Karnaugh map) is a graphical method of simplifying a Boolean equation.
- K-map is a graphical chart made up of rectangular boxes.
- The information contained in a truth table or available in the SOP or POS form can be represented on a K-map.
- The K-map can be used for systematic simplification of Boolean expression.
- K-maps can be written for 2, 3, 4 ... upto 6 variables. Beyond that the K-map technique becomes very cumbersome.

K-map is ideally suitable for designing the combinational logic circuits using either a SOP method or a POS method.

- The K-map is drawn for output Y and the input variables (A, B, C... etc.) are used for making the entries in the boxes.

3.6.1 K-map Structure :

- The structure of a 2 input (variable) Karnaugh is shown in Fig. 3.6.1. This K-map is drawn for output Y of any two input combinational circuit such as a logic gate with inputs A and B. i.e. AB = 00, 01, 10 and 11.

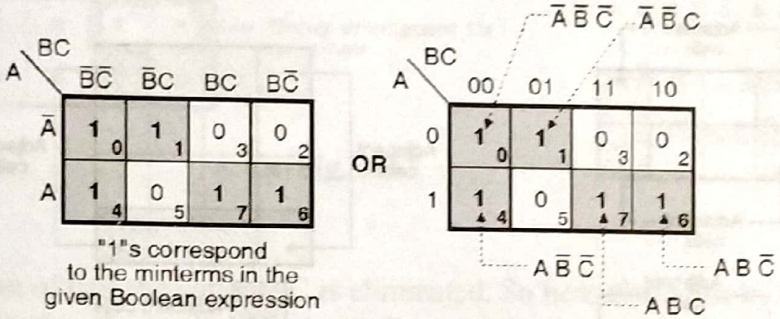
3.6.5 Representation of Standard SOP form on K-map :

- The logical expression in standard SOP form can be represented with the help of a K-map by simply entering 1's in the cells (boxes) of the K-map corresponding to each minterm present in the equation.
- The remaining cells (boxes) are filled with zeros.
- Ex. 3.6.1 illustrates the concept of transferring a standard SOP expression on K-map.

Ex. 3.6.1 : Represent the equation given below on Karnaugh map.

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

Soln. : The given expression is in the standard SOP form. Each term represents a minterm. We have to enter 1's in the boxes corresponding to each minterm, as shown in Fig. P. 3.6.1.

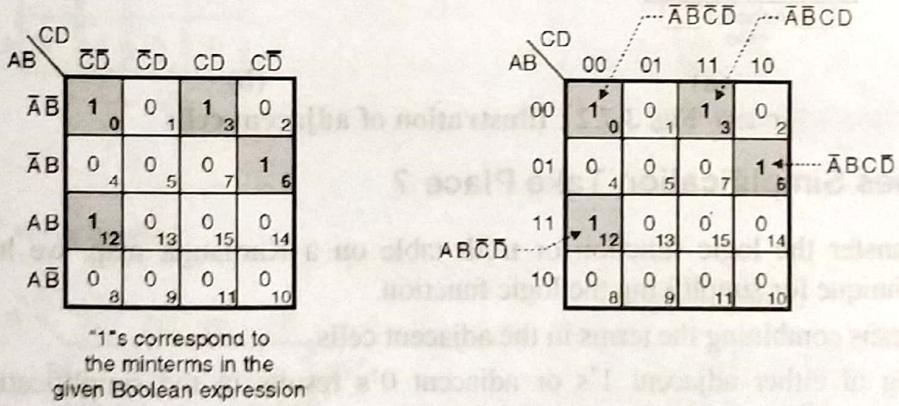


(C-223) **Fig. P. 3.6.1 : Representation of standard SOP on K-map**

Ex. 3.6.2 : Plot the following Boolean expression on K-map.

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

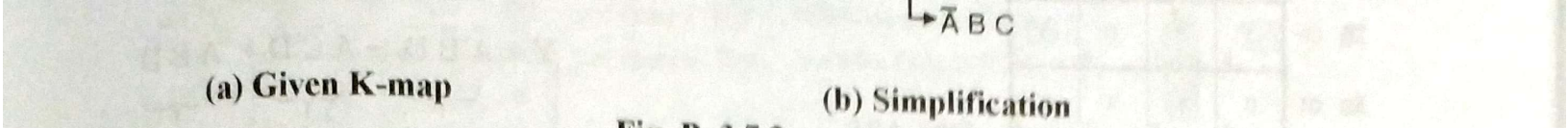
Soln. : Refer Fig. P. 3.6.2.



(C-224) **Fig. P. 3.6.2 : Representation of canonical SOP on Karnaugh map**

3.7 Simplification of Boolean Expressions using K-map :

- Simplification of Boolean expressions using K-map is based on combining or grouping the terms in the adjacent cells (or boxes) of a K-map.
- Two cells of a K-map are said to be adjacent if they differ in only one variable as shown in Fig. 3.7.1.



(a) Given K-map (b) Simplification

Fig. P. 3.7.2

3.8 Minimization of SOP Expressions (K Map Simplification) :

- The K-map can be used to simplify the logical expression to a level beyond which it cannot be further simplified.
- After such a simplification, it will require minimum number of gates with minimum number of inputs to the gates. Such an expression is called as a **minimized** expression.
- For minimizing the logical expression, follow the procedure given below.

Minimization procedure :

Step 1 : Prepare the K-map and place 1's according to the given truth table or logical expression. Fill the remaining cells by 0's.

Step 2 : Locate the isolated 1's i.e. the 1's which cannot be combined with any other 1. Encircle such 1's.

Step 3 : Identify the 1's which can be combined to form a pair in only one way and encircle them.

Step 4 : Identify the 1's which can form a quad in only one way and encircle them.

Step 5 : Identify the 1's which can form an octet in only one way and encircle them.

Step 6 : After identifying the pairs, quads and octets, check if any 1 is yet to be encircled. If yes then encircle them with each other or with the already encircled 1's (by means of overlapping).

- Note that the number of groups should be minimum.
- Also note that any 1 can be included any number of times without affecting the expression.

Let us solve some examples on this to make the concept clear.

Ex. 3.8.1 : A logical expression in the standard SOP form is as follows :

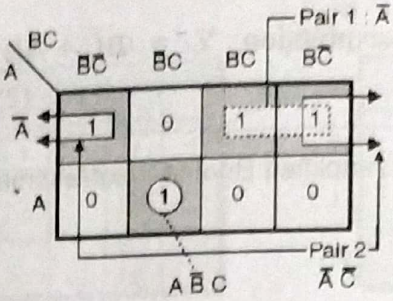
$$Y = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,B\,\bar{C} + \bar{A}BC + A\bar{B}C$$

Minimize it using the K-map technique.

Soln. :

$$Y = \Sigma m\,(0, 2, 3, 5)$$

The required k-map is as shown in Fig. P. 3.8.1.



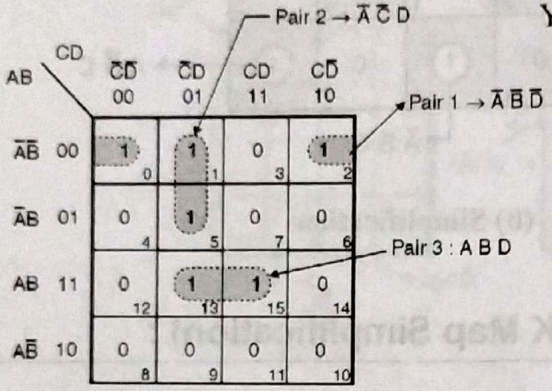
(C-250) **Fig. P. 3.8.1**

Ex. 3.8.2 : The logical expression representing a logic circuit is $Y = \Sigma m\,(0, 1, 2, 5, 13, 15)$. Draw the K-map and find the minimized logical expression.

Soln. : From the given expression, it is clear that the number of variables is 4.

$$Y = m_0 + m_1 + m_2 + m_5 + m_{13} + m_{15}$$

The required k-map is as shown in Fig. P. 3.8.2.



Minimized expression is

$$Y = \underbrace{\bar{A}\bar{B}\bar{D}}_{\text{Pair 1}} + \underbrace{\bar{A}\bar{C}D}_{\text{Pair 2}} + \underbrace{ABD}_{\text{Pair 3}}$$

(C-251) **Fig. P. 3.8.2**

Ex. 3.8.3 : For the logical expression given below draw the K-map and obtain the simplified logical expression.

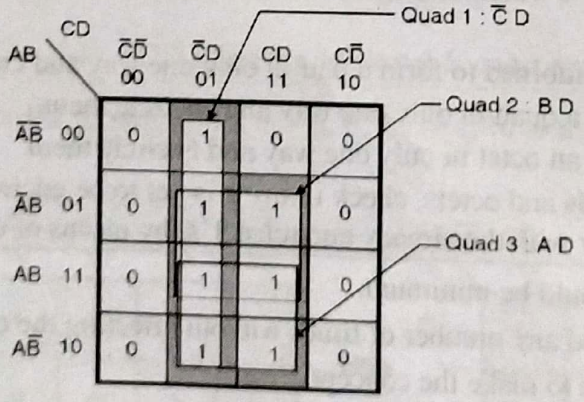
$$Y = \Sigma m\,(1, 5, 7, 9, 11, 13, 15).$$

Realize the minimized expression using the basic gates.

Soln. : The given expression is,

$$Y = m_1 + m_5 + m_7 + m_9 + m_{11} + m_{13} + m_{15}$$

It can be expressed on K-map as shown in Fig. P. 3.8.3(a).



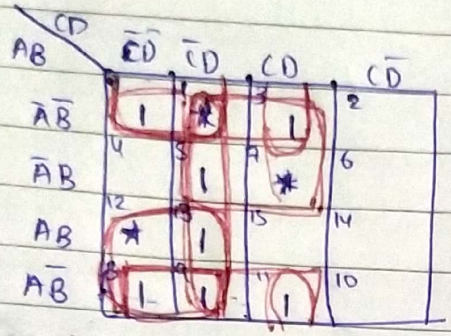
The simplified expression is as follows :

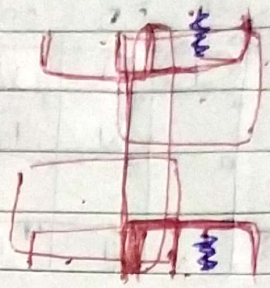
$$Y = \underbrace{\bar{C}D}_{\text{Quad 1}} + \underbrace{BD}_{\text{Quad 2}} + \underbrace{AD}_{\text{Quad 3}} \quad \ldots(1)$$

(C-252) **Fig. P. 3.8.3(a)**

Ex Minimize the following function using K Map

$f(A,B,C,D) = \Sigma m(0,3,5,8,9,11,13) + \Sigma d(1,7,12)$

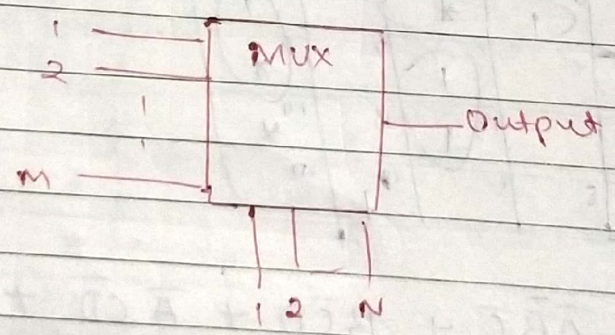




~~$Y = \bar{A}B\bar{C} + \bar{C}D + A\bar{C} + \bar{B}CD$~~

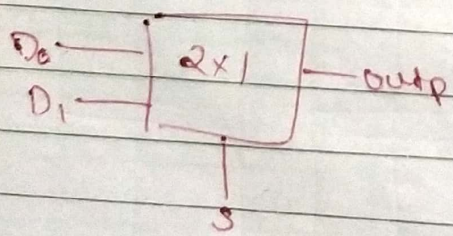
$Y = A\bar{C} + \bar{B}\bar{C} + \bar{C}D + \bar{A}D + \bar{B}D$

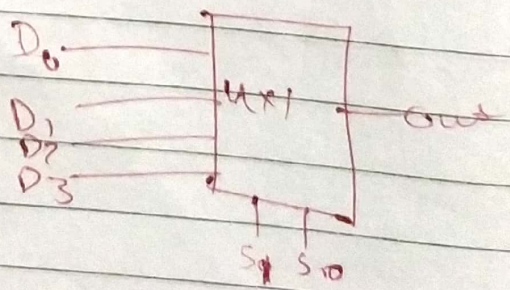
⇒ Multiplexer – the combinational Circuit which has M inputs, and one output. Apart of that it also has the N selection lines, So the block diagram of the multiplexer

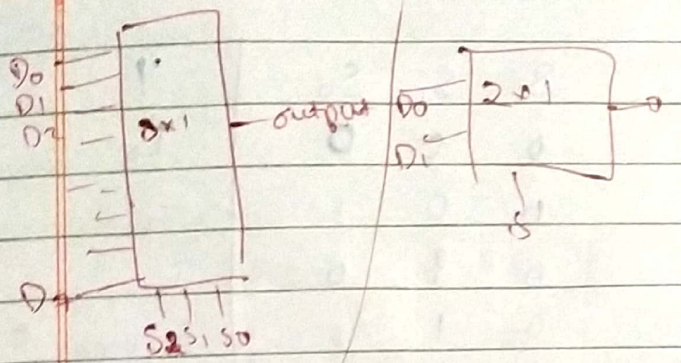


$M = 2^N$

$N = \log_2(m)$

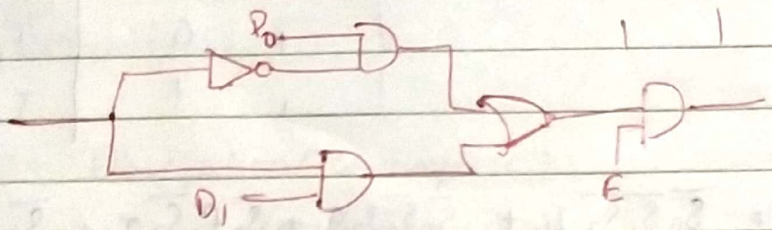




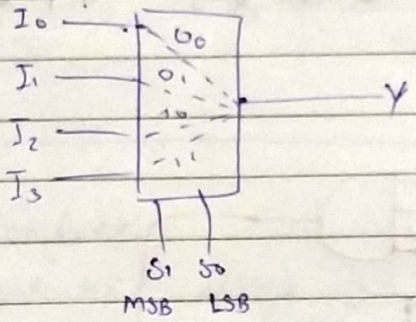


D1	D0	S	
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

S	Y
0	D0
1	D1



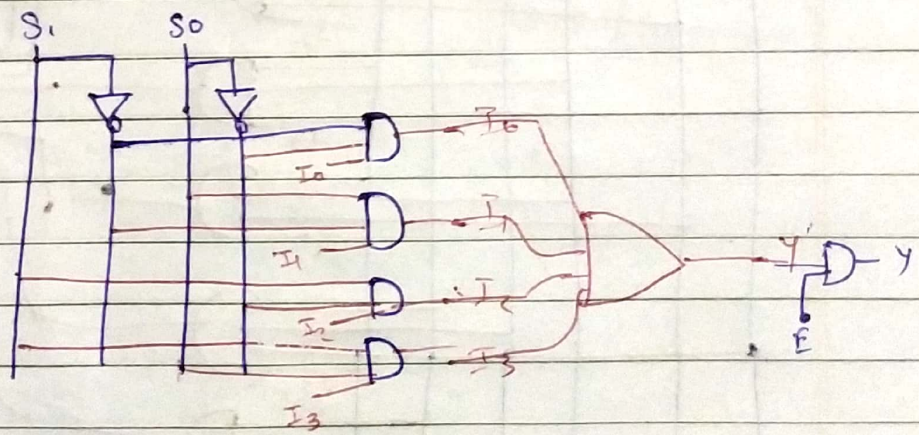
4×1



S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

$Y = \overline{S_1}\,\overline{S_0}\,I_0 + S_0\overline{S_1}\,I_1 + \overline{S_0}\,S_1 I_2 + S_0 S_1 I_3$

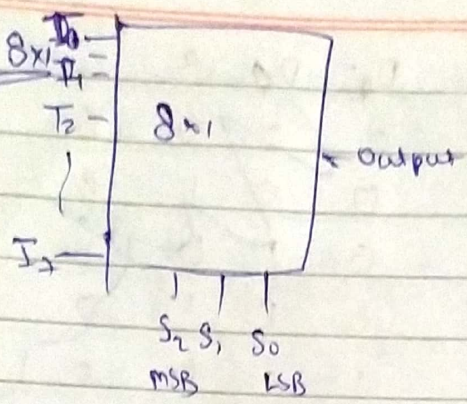
Logic circuit



E = 0 Disable

E = 1 Enable

8×1

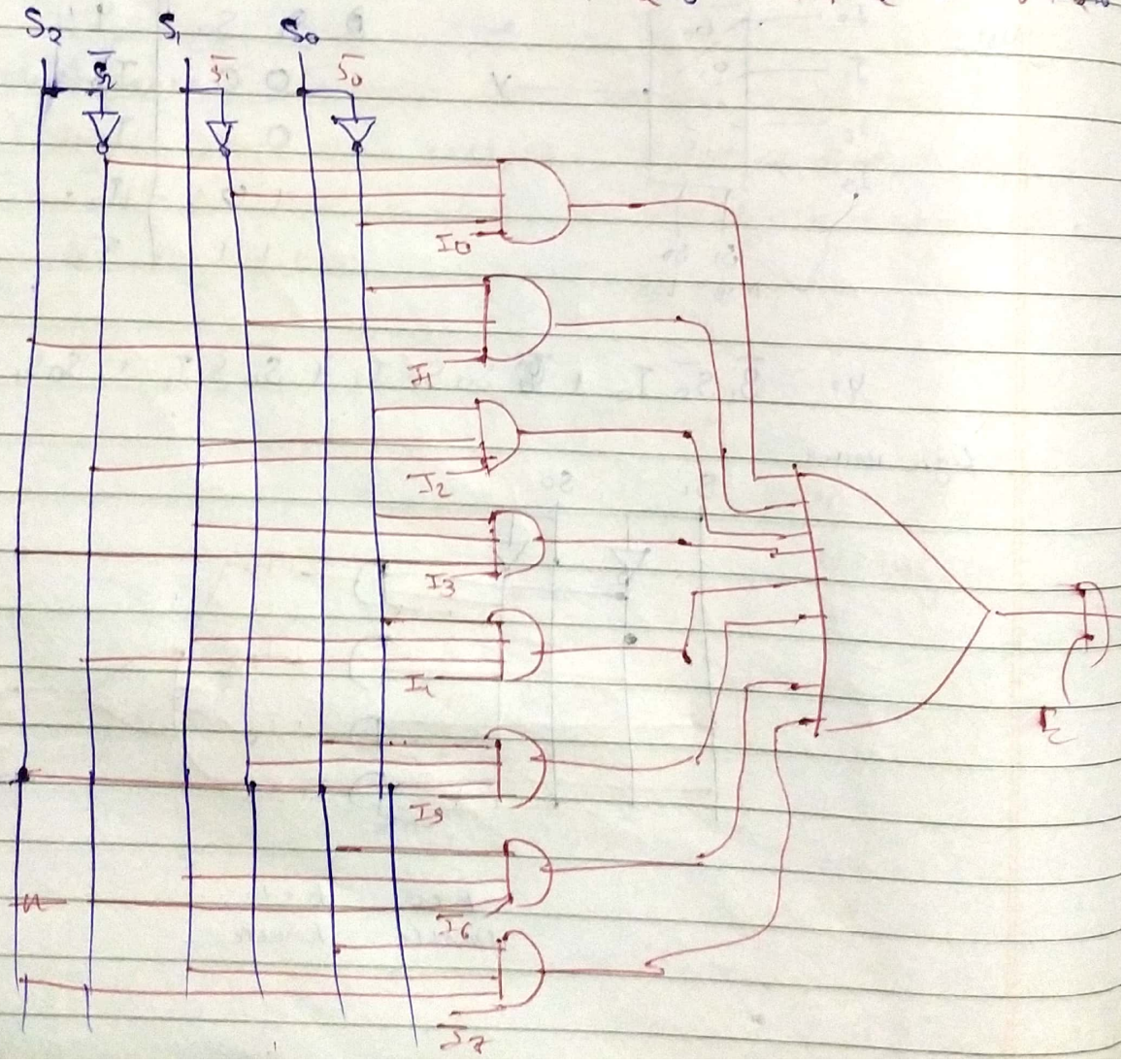


Truth table

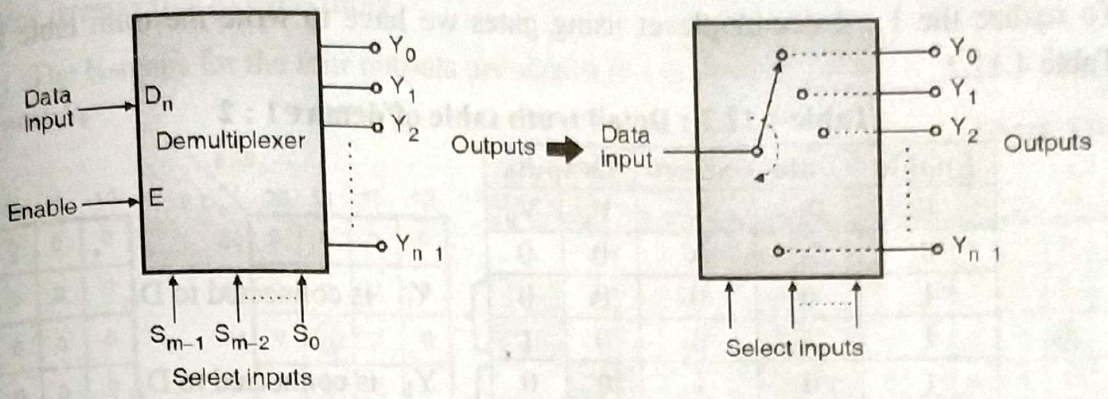
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Ckt logic :-

$$Y = \bar{S_0}\bar{S_1}\bar{S_2}I_0 + S_0\bar{S_1}\bar{S_2}I_1 + \bar{S_0}S_1\bar{S_2}I_2 + S_0S_1\bar{S_2}I_3 + \bar{S_0}\bar{S_1}S_2I_4 + S_0\bar{S_1}S_2I_5 + \bar{S_0}S_1S_2I_6 + S_0S_1S_2I_7$$



Demultiplexer



(a) 1 : n demultiplexer **(b) Equivalent circuit**

(C-447) **Fig. 4.12.1**

- It has only one input, "n" outputs, and "m" select inputs.
- A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs.
- At a time only one output line is selected by the select lines and the input is routed to the selected output line.
- Hence a demultiplexer is equivalent to a single pole multiple way switch as shown in Fig. 4.12.1(b). The enable input will enable the demultiplexer. If the enable (E) input is not active, then the demultiplexer does not work.
- The relation between the n output lines and m select lines is as follows :

$$n = 2^m$$

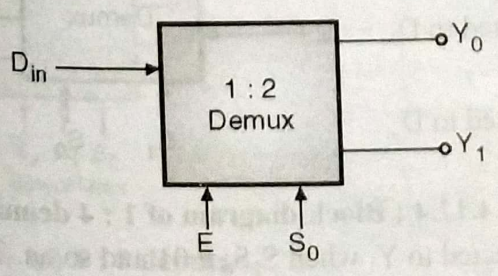
4.12.2 Types of Demultiplexers :

Similar to the multiplexers, the demultiplexers are classified as follows :

1. 1 : 2 demultiplexer
2. 1 : 4 demultiplexer
3. 1 : 8 demultiplexer
4. 1 : 16 demultiplexer

4.12.3 1 : 2 Demultiplexer :

- The block diagram of 1 : 2 demultiplexer is shown in Fig. 4.12.2. It has one data input D_{in}, one select input S_0, one enable (E) input and two outputs Y_0 and Y_1.
- D_{in} is connected to Y_0 if $S_0 = 0$ and $E = 1$. Similarly D_{in} is connected to Y_1 if $S_0 = 1$ and $E = 1$.
- If $E = 0$, then both the outputs will be 0 irrespective of the inputs, because the DEMUX is disabled.
- The truth table of the 1 : 2 demultiplexer is as follows :



(C-448) **Fig. 4.12.2**

Table 4.12.1 : Truth table of demux 1 : 2

Enable	Select	Outputs	
E	S_0	Y_1	Y_0
0	X	0	0
1	0	0	D_{in}
1	1	D_{in}	0

Realization of 1 : 2 demux :

Step 1 : To realize the 1 : 2 demultiplexer using gates we have to write the truth table as given in Table 4.12.2 :

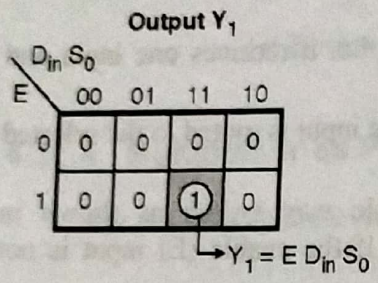
Table 4.12.2 : Detail truth table of demux 1 : 2

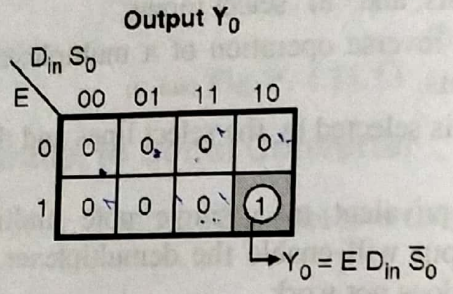
Enable	Data	Select	Outputs	
E	D_{in}	S_0	Y_1	Y_0
0	X	X	0	0
1	0	0	0	0
1	1	0	0	1
1	0	1	0	0
1	1	1	1	0

Y_0 is connected to D_{in}

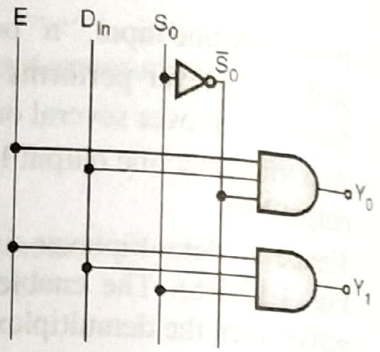
Y_1 is connected to D_{in}

Step 2 : Write the K maps : The k maps for the two outputs are as follows.





(C-449) **Fig. 4.12.2(a) : K maps**



(C-450) **Fig. 4.12.3 : 1 : 2 demux using gates**

- Thus the expressions for Y_0 and Y_1 are as follows :

$$Y_0 = ED_{in}\bar{S_0} \quad \text{and} \quad Y_1 = E\,D_{in}\,S_0$$

- The 1 : 2 demux using gates is shown in Fig. 4.12.2.

4.12.4 1 : 4 Demultiplexer :

- The 1 : 4 demultiplexer is shown in Fig. 4.12.3 and its truth table is given in Table 4.12.3.

Table 4.12.3 : Truth table for 1 : 4 demultiplexer

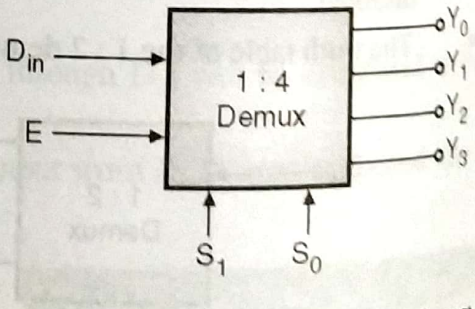
		Inputs		Outputs			
E	D_{in}	S_1	S_0	Y_0	Y_1	Y_2	Y_3
1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	0
1	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0
1	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1

Y_0 is connected to D_{in}

Y_1 is connected to D_{in}

Y_2 is connected to D_{in}

Y_3 is connected to D_{in}

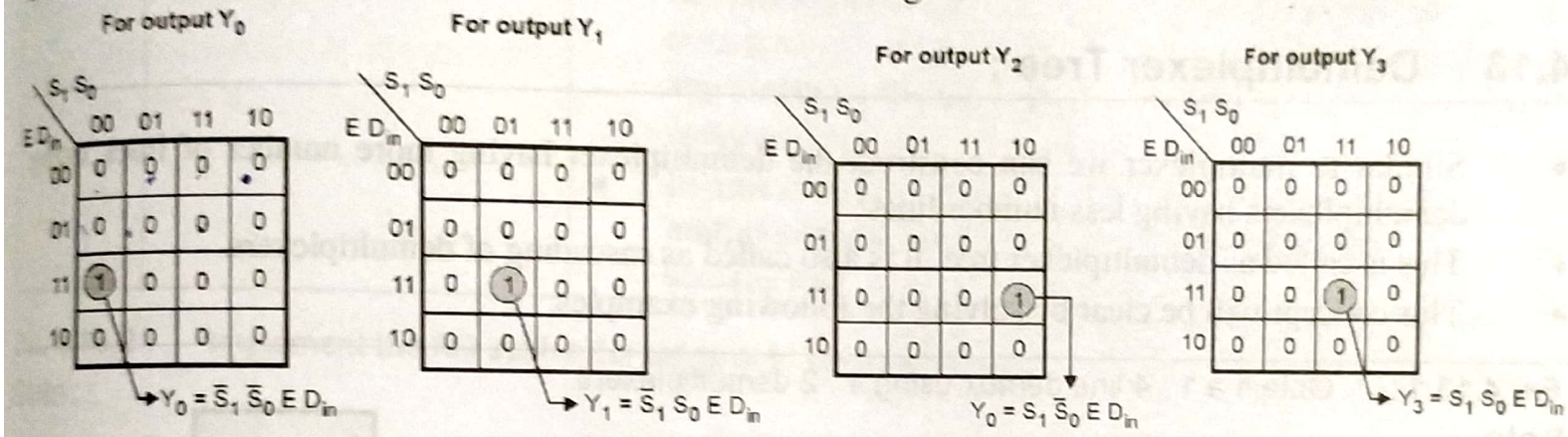


(C-451) **Fig. 4.12.4 : Block diagram of 1 : 4 demultiplexer**

- D_{in} is connected to Y_0 when $S_1S_0 = 00$, it is connected to Y_1 when $S_1S_0 = 01$ and so on. The other outputs will remain zero.

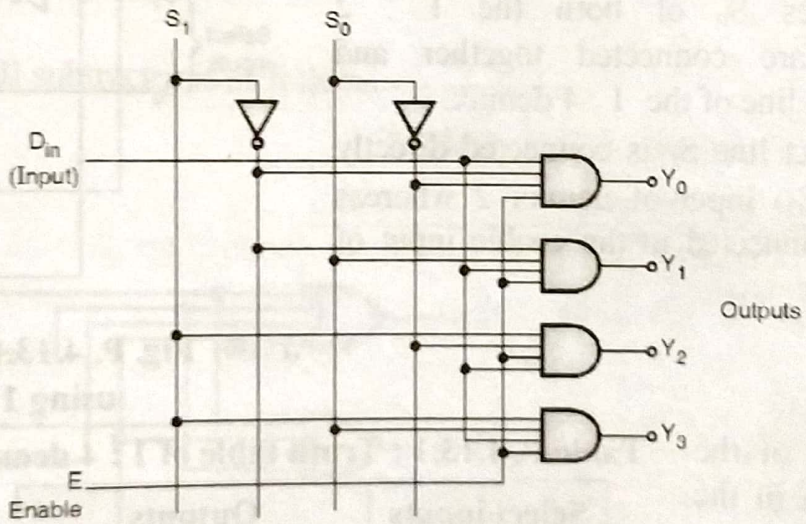
- The enable input needs to be high in order to enable the demux. If E = 0 then all the outputs will be low irrespective of everything.

K-maps : The K-maps for the four outputs are shown in Fig. 4.12.5.



(C-452) **Fig. 4.12.5 : K-maps for various outputs**

Implementation : The 1 : 4 demux is implemented as shown in Fig. 4.12.6.



(C-453) **Fig. 4.12.6 : 1:4 demultiplexer**

4.2 Binary Adders and Subtractors :

New Syll. : MDU : May 12

Addition of two binary digits is most basic operation performed by the digital computers.

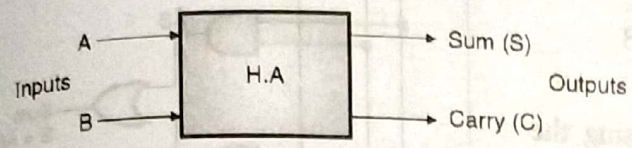
4.2.1 Types of Binary Adders :

- In this section we are going to learn digital circuits which are used to "add" two "binary" numbers. The binary adders are of two types :

  1. Half adder and 2. Full adder

4.2.2 Half Adder :

- Half adder is a combinational logic circuit with two inputs and two outputs. It is the basic building block for addition of two "single" bit numbers. This circuit has two outputs namely "carry" and "sum". The block diagram of half adder is as shown in Fig. 4.2.1(a).
- The half adder circuit is supposed to add two single bit binary numbers A and B. Therefore the truth table of a half adder is as shown in Fig. 4.2.1(b).



(a) Block diagram

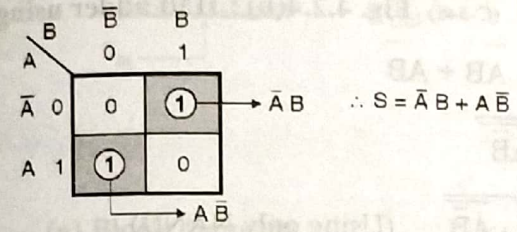
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(b) Truth table

(C-344) **Fig. 4.2.1 : Half adder**

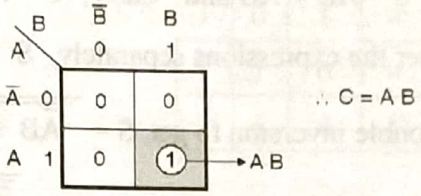
Karnaugh maps and simplified expressions for outputs :

- K-maps for carry and sum outputs are as shown in Figs. 4.2.1(a) and (b).



(a) K - map for sum output

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



$\therefore C = AB$

(b) K - map for carry output

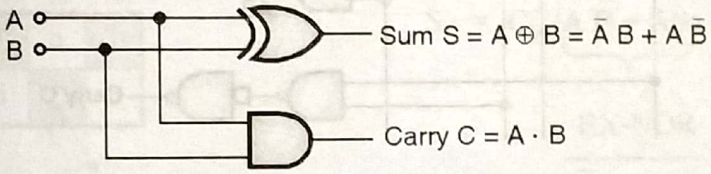
(C-345) **Fig. 4.2.2**

- Boolean expressions for the sum (S) and carry (C) output are obtained from the K-maps as follows :

$$\left.\begin{aligned} S &= A\bar{B} + \bar{A}B = A \oplus B \\ C &= AB \end{aligned}\right\} \quad \ldots(4.2.1)$$

The disadvantage of half adder is that addition of three bits is not possible to perform.

- Hence the half adder circuit is as shown in Fig. 4.2.3.



(C-346) **Fig. 4.2.3 : Half adder circuit**

Disadvantage of half adder :

- The principle of adding two 2-bit numbers A and B is as shown in Fig. 4.2.4(a).

Let Number A = $A_1 A_0$ and Number B = $B_1 B_0$

Then the addition should take place as shown in Fig. 4.2.4(a).

- A half adder can add A_0 and B_0 to produce S_0 and C_0. But the addition of next bits requires the addition of A_1, B_1 and C_0. The addition of three bits is not possible to perform by using a half adder. Hence we cannot use a half adder in practice.

A_1 A_0 ← Number A
\+ B_1 B_0 ← Number B
\+ C_0 ← Carry generated from the addition $(A_0 + B_0)$
C_1 S_1 S_0

(C-347) **Fig. 4.2.4(a) : Two bit binary addition**

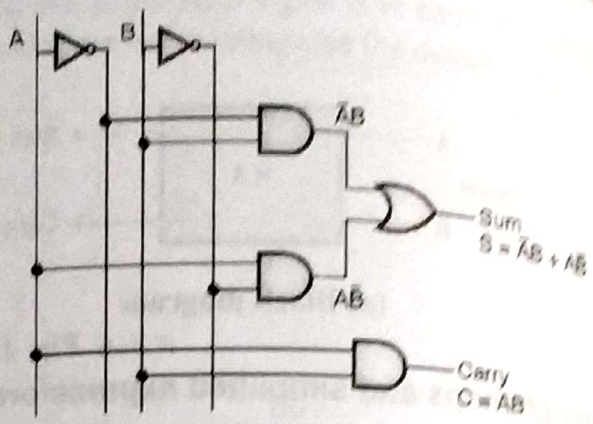
Half adder using basic gates :

- Refer Equation (4.2.1) which states that,

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

$$\text{and Carry } C = AB$$

- These equations are implemented using the basic gates as shown in Fig. 4.2.4(b).



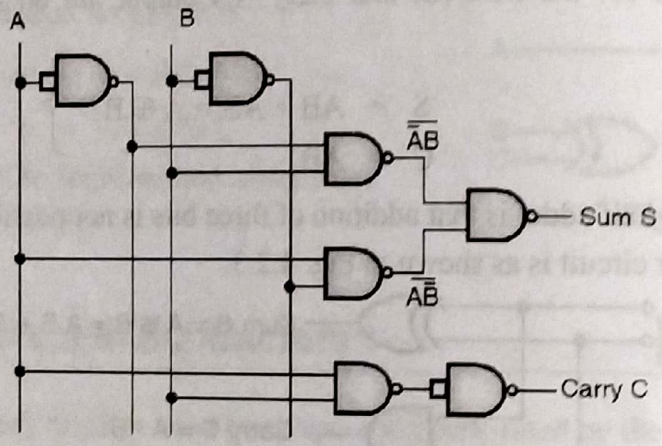
(C-348) **Fig. 4.2.4(b) : Half adder using basic gates**

Half adder using only NAND gates :

- Equation (4.2.1) states that,
- Sum, $S = \bar{A}B + A\bar{B}$ and Carry, $C = AB$
- Consider the expressions separately, $S = \bar{A}B + A\bar{B}$
- Take double inversion to get, $S = \overline{\overline{\bar{A}B + A\bar{B}}}$

Using the DeMorgan's theorem, $S = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}}$ ...(Using only NAND) ...(4.2.2)

- Similarly $C = AB = \overline{\overline{AB}}$ ...(Using only NAND) ...(4.2.3)



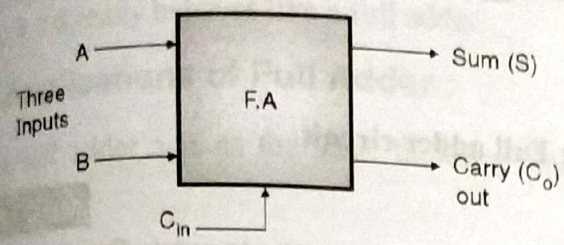
(C-349) **Fig. 4.2.4(c) : Half adder using only NAND gates**

- Equations (4.2.2) and (4.2.3) are implemented using only NAND gates as shown in Fig. 4.2.4(c).

4.2.3 Full Adder :

MDU : May 07, New Syll. : May 11

- To overcome the drawback of Half Adder circuit, a 3 single bit adder circuit called Full Adder is developed.
- It can add two one-bit numbers A and B, and carry C_{in}. The full adder is a three input and two output combinational circuit.
- The block diagram of a full adder is as shown in Fig. 4.2.5(a) and its truth table is given in Fig. 4.2.5(b).



(a) Block diagram

Inputs			Outputs	
A	B	C_{in}	S	C_o
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

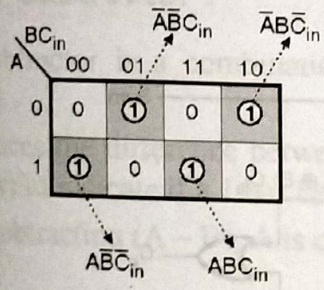
(b) Truth table

(C-350) **Fig. 4.2.5 : Full adder**

The K-maps :

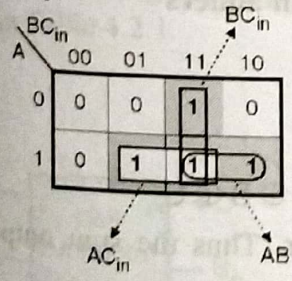
The K-maps for the sum (S) and carry out (C_o) outputs and the corresponding Boolean expressions are as shown in Fig. 4.2.6. Note that the K-maps have been written from the truth table of Fig. 4.2.5(b).

For the sum output



(C-351) **Fig. 4.2.6(a) : K-map for sum output**

For carry output



(C-352) **Fig. 4.2.6(b) : K-map for carry output**

Expression for sum output

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

$$S = C_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{C}_{in}\underbrace{(\bar{A}B + A\bar{B})}_{\text{EX-OR}}$$

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

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

$$\therefore S = C_{in}\bar{X} + \bar{C}_{in}X = C_{in} \oplus X$$

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

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

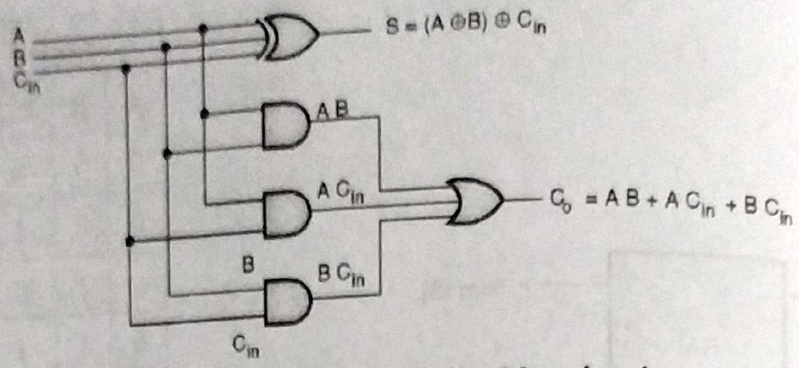
$$\therefore S = C_{in} \oplus A \oplus B \quad \ldots(4.2.4)$$

Expression for carry output

$$C_o = AB + AC_{in} + BC_{in} \quad \ldots(4.2.5)$$

We can use Equations (4.2.4) and (4.2.5) to draw the logic diagram of a full adder as shown in Fig. 4.2.6(c).

Logic diagram for full adder :

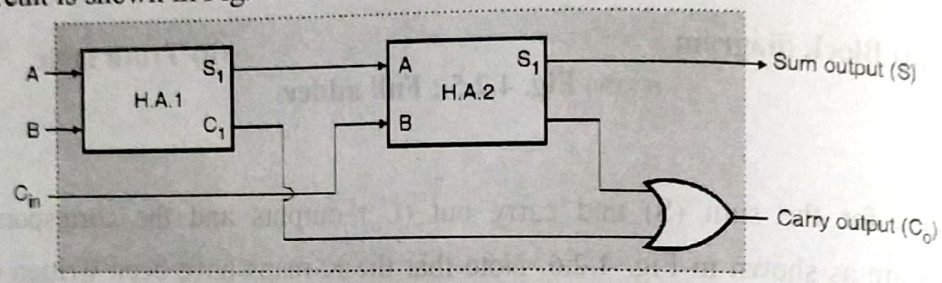


(C-353) **Fig. 4.2.6(c) : Full adder circuit**

4.2.4 Full Adder using Half Adder :

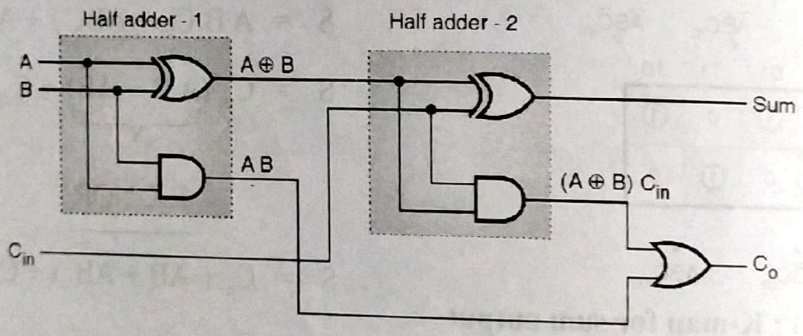
MDU : May 07

- The full adder circuit can be constructed using two half adders as shown in Fig. 4.2.7 and the detail circuit is shown in Fig. 4.2.8.



(C-354) **Fig. 4.2.7 : Full adder using half adders**

- A full adder can be implemented using two half adders and an OR gate as shown in Fig. 4.2.8.



(C-355) **Fig. 4.2.8 : Full adder using two half adders**

- Now let us prove that this circuit acts as a full adder.

Proof :

- Refer Fig. 4.2.8 and write the expression for sum output as,

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

This expression is same as that obtained for the full adder. Thus the sum output has been successfully implemented by the circuit shown in Fig. 4.2.8.

- Now write the expression for carry output C_o as,

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

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

1. Half subtractor 2. Full subtractor

4.7 Half Subtractor :

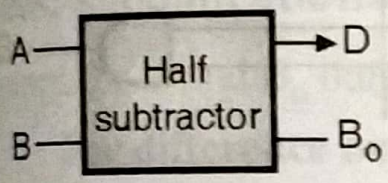
Half subtractor is a combinational circuit with two inputs and two outputs (difference and borrow).

It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed.

In the subtraction (A – B), A is called as **minuend bit** and B is called as **subtrahend bit**.

4.7.1 Truth table :

Truth table showing the outputs of a half subtractor for all the possible combinations of input are shown in Table 4.2.1.



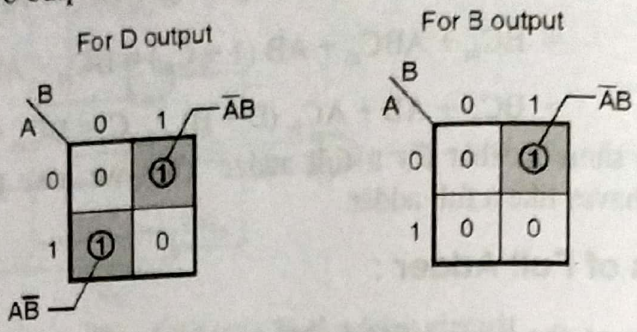
(C-3514)

Table 4.2.1 : Truth table for half subtractor

Inputs		Outputs	
A	B	Difference D (A – B)	Borrow B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-maps for difference and borrow outputs :

The K-maps for the two outputs of a half subtractor are as shown in Fig. 4.2.9.



(a) K-map and simplification for difference output **(b) K-map and simplification for the borrow output**

(C-356) **Fig. 4.2.9**

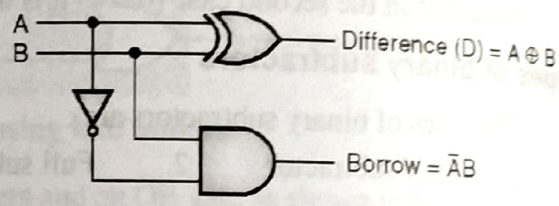
$\therefore$ Difference $D = \bar{A}B + A\bar{B} = A \oplus B$ $\therefore$ Borrow $B_o = \bar{A}B$

Logic diagram :

The logic diagram using these two Boolean expressions is as shown in Fig. 4.2.9(c).

Disadvantage of the half subtractor :

Half subtractor can only perform the subtraction of two binary bits. But while performing the subtraction, it does not take into account the borrow of the lower significant stage.



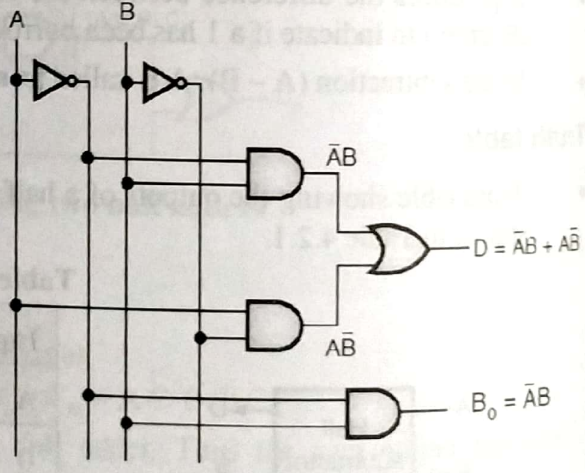
(C-357) **Fig. 4.2.9(c) : Half subtractor circuit**

Half subtractor using basic gates :

The expressions for the difference and borrow outputs are :

$$D = \bar{A}B + A\bar{B} \quad \text{and} \quad B_o = \bar{A}B$$

The half subtractor using basic gates is as shown in Fig. 4.2.10.



(C-358) **Fig. 4.2.10 : Half subtractor using basic gates**

Half subtractor using NAND gates :

- The expression for difference output is,

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

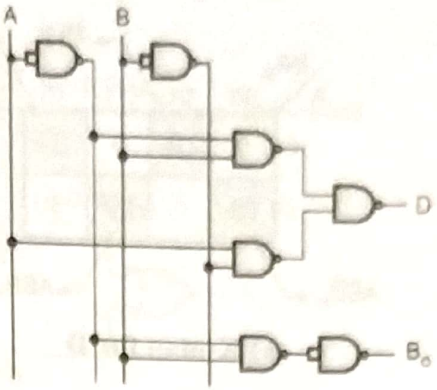
$$D = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}} \quad \text{...Double inversion}$$

$$D = \overline{AB} \cdot \overline{AB} \quad \ldots(4.2.6)$$

Similarly the other output B_o is given by,

$$B_o = \overline{A}B = A\overline{B} \quad \ldots(4.2.7)$$

Equations (4.2.6) and (4.2.7) can be implemented as shown in Fig. 4.2.11.



(C-359) **Fig. 4.2.11 : Half subtractor using NAND gates**

4.2.8 Full Subtractor :

MDU : May 05, May 07

The disadvantage of a half subtractor is overcome if we use the full subtractor.

The full subtractor is a combinational circuit with three inputs A, B and B_{in} and two outputs D and B_o.

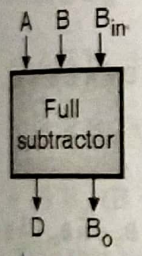
A is the minuend, B is subtrahend, B_{in} is the borrow produced by the previous stage. D is the difference output and B_o is the borrow output.

Truth table :

The truth table for full subtractor is shown in Table 4.2.2.

Table 4.2.2 : Truth table for a full subtractor

Inputs			Outputs	
A (Minuend)	**B (Subtrahend)**	**B_{in} Previous borrow**	**$(A - B - B_{in})$**	**B_o**
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



(C-3515)

K-maps and simplifications :

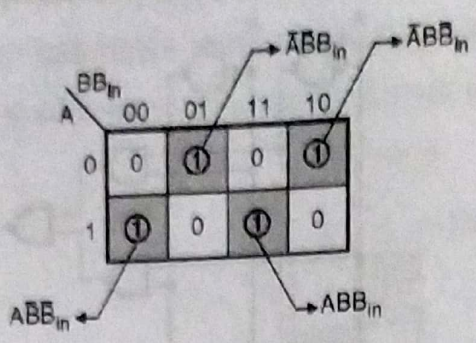
K-maps for D and B_o outputs are shown in Figs. 4.2.12(a) and (b).

For difference output

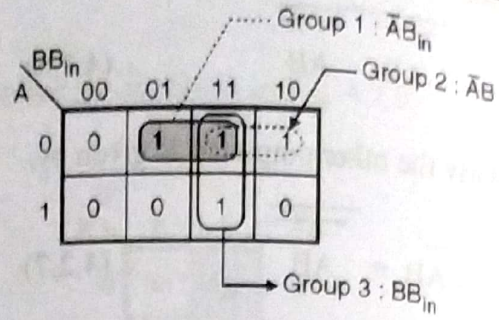
$$\therefore D = \overline{A}\,\overline{B}\,B_{in} + \overline{A}B\overline{B}_{in} + A\overline{B}\overline{B}_{in} + ABB_{in}$$

For borrow output

$$\therefore B_o = \overline{A}B_{in} + \overline{A}B + BB_{in}$$



(a) K-map for D



(b) K-map for B_o

(C-360) **Fig. 4.2.12**

Simplification for difference output :

From Fig. 4.2.12(a),

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

$$= B_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{B}_{in}\underbrace{(A\bar{B} + \bar{A}B)}_{\text{EX-OR}}$$

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

Let $A \oplus B = C$,

$$\therefore D = B_{in}\bar{C} + \bar{B}_{in}C = B_{in} \oplus C$$

$$\therefore D = B_{in} \oplus A \oplus B \quad \ldots(4.2.8)$$

Simplification for borrow output :

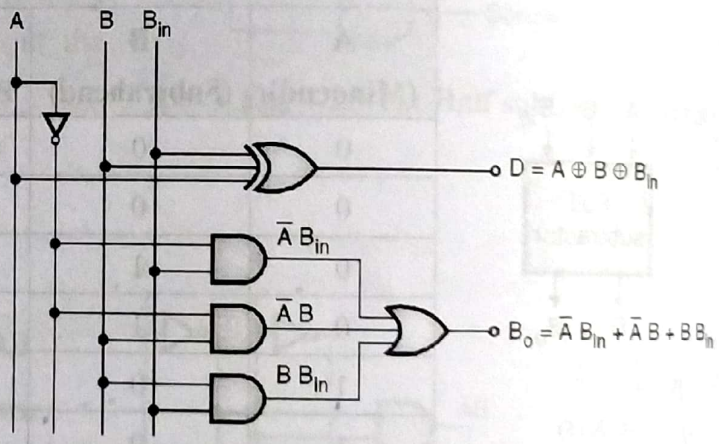
From Fig. 4.2.12(b),

$$B_o = \bar{A}B_{in} + \bar{A}B + BB_{in} \quad \ldots(4.2.9)$$

No further simplification is possible.

Logic diagram for full subtractor :

Logic diagram for the full subtractor is shown in Fig. 4.2.13. This has been drawn by using the Boolean equations of (4.2.8) and (4.2.9).



(C-361) **Fig. 4.2.13 : Logic diagram for a full subtractor**

4.2.9 Full Subtractor using Half Subtractors :

MDU : May 07

- Fig. 4.2.14 shows the implementation of a full subtractor using two half subtractors and an OR gate.