SUM OF PRODUCTS & PRODUCT OF SUMS

Logical functions are generally expressed in terms of logical variables. An arbitrary logic function can be expressed in the following forms:

1) **Sum of products (SOP)**
2) **Product of sums (POS)**

Product term: The logical product of several variables on which a function depends.

Sum term: The logical sum of several variables on which a function depends.

By : Ms. Kanika Dhingra

SOP: The logical sum of 2 or more logical product terms.

$$Y = AB + BC + \overline{A}C$$

POS : The logical product of 2 or more logical sum terms.

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

Minterm: A product term containing all the K variables of the function in either complemented or uncomplemented form.

- **The main property of the minterm is that it possesses the value 1 for only one combination of k input variables.**

Canonical Sum of Product Expression

It is defined as the logical sum of all the minterms derived from the rows of a truth table , for which the value of the function is 1.

Using the following procedure, the canonical sum of product form of a logic function can be obtained:

1. Examine each term in the given logic function. Retain it if it is a minterm; continue to examine the next term in the same manner.
2. Check for variables that are missing in each product which is not a minterm. Multiply the product by $(X + \overline{X})$, for each variable X that is missing.
3. Multiply all the products and omit the redundant terms.

By : Ms. Kanika Dhingra

Example 2.22 Obtain the canonical sum of product form of the function

$$Y(A, B) = A + B$$

Solution The given function containing the two variables A and B has the variable B missing in the first term and the variable A missing in the second. Therefore, the first term has to be multiplied by $(B + \overline{B})$, the second term by $(A + \overline{A})$ as given below:

$$\begin{aligned} A + B &= A \cdot 1 + B \cdot 1 \\ &= A \cdot (B + \overline{B}) + B \cdot (A + \overline{A}) \\ &= AB + A\overline{B} + BA + B\overline{A} \\ &= AB + A\overline{B} + \overline{A}B \quad (\because AB + AB = AB) \end{aligned}$$

$$Y(A, B) = A + B = AB + A\overline{B} + \overline{A}B$$

Example 2.23 Obtain the canonical sum of product form of the function

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

Solution Here, neither the first term nor the second term is a minterm. The variables B and C in the first term and the variable A in the second term are missing.

Therefore, the first term has to be multiplied by $B + \overline{B}$ and $C + \overline{C}$, the second term by $(A + \overline{A})$ as shown below:

$$\begin{aligned} A + BC &= A(B + \overline{B})(C + \overline{C}) + BC(A + \overline{A}) \\ &= (AB + A\overline{B})(C + \overline{C}) + BCA + BC\overline{A} \\ &= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + BCA + BC\overline{A} \\ &= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}BC \qquad (\because ABC + ABC = ABC) \end{aligned}$$

Therefore, the canonical sum of product form of $Y = A + BC$ is given by

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

Maxterm: A sum term containing all the k variables of the function in either complemented or uncomplemented form.

- **The most important property of a maxterm is that it possesses the value 0 for only one combination of k input variables.**

Canonical Product of sum

It is defined as the logical product of all the maxterms derived from the rows of truth table, for which the value of function is 0.

The following procedure can be used to obtain the canonical product of the sum form of a logic function:-

1. Examine each term in the given logic function. Retain it if it is a maxterm; continue to examine the next term in the same manner.

2. Check for variables that are missing in each sum, which is not a maxterm. Add $(x\bar{x})$ to the sum term for each variable x that is missing.

3. Expand the expression using the distributive property and eliminate the redundant terms.

Example 2.25 Obtain the canonical product of the sum expression of $Y(ABC) = (A+\overline{B})(B+C)(A+\overline{C})$.

Solution In the given expression, the variable C is missing in the first term, the variable A in the second term and the variable B in the third term. Therefore, $C\overline{C}$, $A\overline{A}$ and $B\overline{B}$ have to be added with the first, second and third terms respectively as shown below:

$$\begin{aligned} Y(ABC) &= (A+\overline{B})(B+C)(A+\overline{C}) \\ &= (A+\overline{B}+0)(B+C+0)(A+\overline{C}+0) \\ &= (A+\overline{B}+C\overline{C})(B+C+A\overline{A})(A+\overline{C}+B\overline{B}) \end{aligned}$$

Now, using the distributive property, each sum term can be expanded as

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

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

$$[\because (A+\overline{B}+\overline{C})(A+\overline{B}+\overline{C}) = (A+\overline{B}+\overline{C})]$$

This is called the maxterm canonical form or the canonical product of sum expression.

Deriving Sum of Product & Product of Sum Expression from a Truth Table

Inputs			Output	Product terms	Sum terms
A	B	C	Y		
0	0	0	0		$(A+B+C)$
0	0	1	0		$(A+B+\overline{C})$
0	1	0	1	$\overline{A}B\overline{C}$	
0	1	1	1	$\overline{A}BC$	
1	0	0	0		$(\overline{A}+B+C)$
1	0	1	1	$A\overline{B}C$	
1	1	0	0		$(\overline{A}+\overline{B}+C)$
1	1	1	1	ABC	

Now, the final SOP expression for the output Y is obtained by summing (OR operation of) the four product terms as follows:

$$Y = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

Now the final POS expression for the output Y is obtained by the AND operation of the four sum terms as follows:

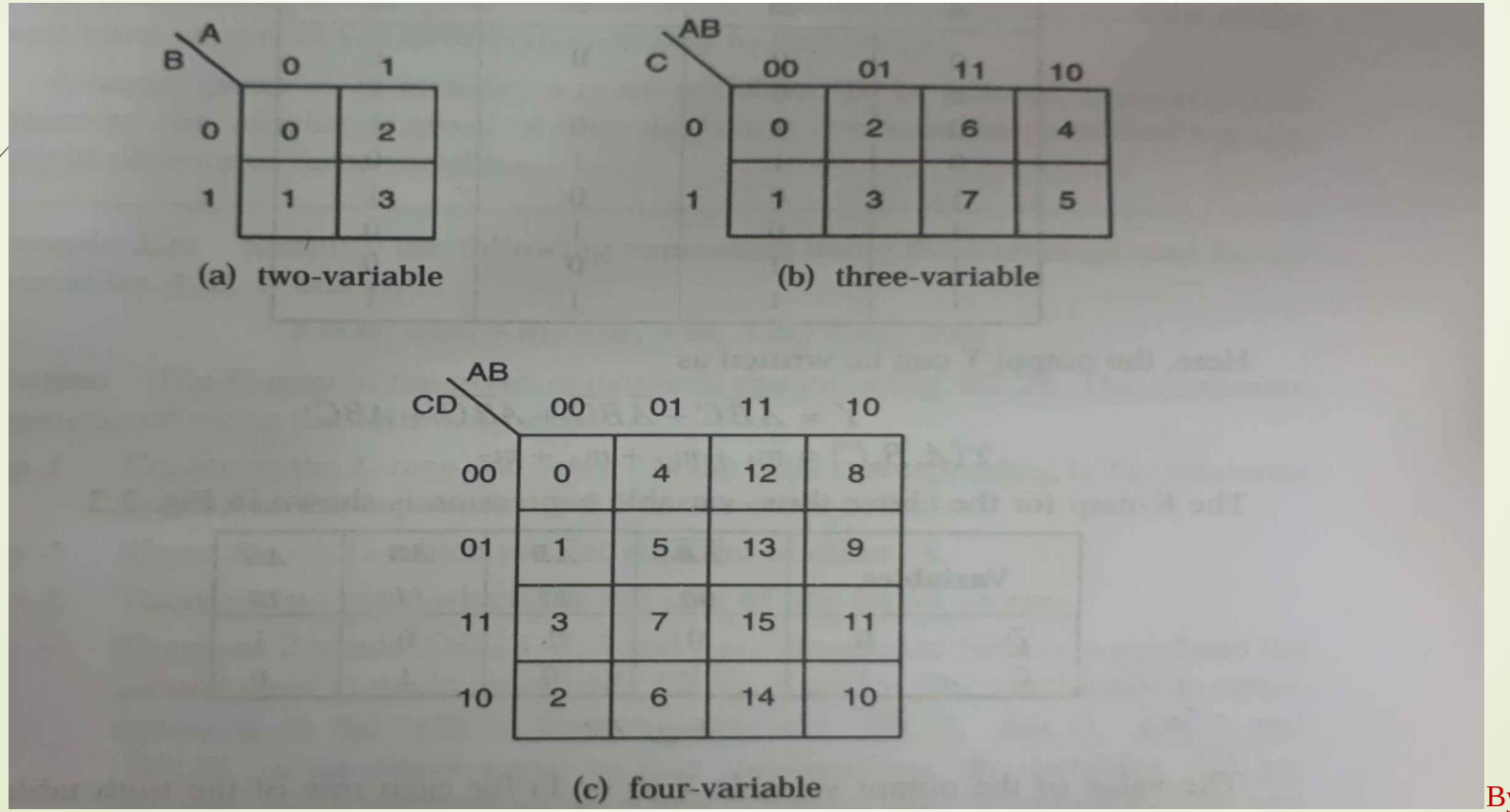
$$Y = (A+B+C)(A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$$

By : Ms. Kanika Dhingra

Karnaugh Map

The k- map is actually a modified form of a truth table. In this technique, the information contained in the truth table or available in the POS or SOP form is represented on the karnaugh map.

In an n variable k-map, there are 2^n cells. Each cell corresponds to one combination of n variables.



B \ A	0	1
0	0	2
1	1	3

(a) two-variable

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

(b) three-variable

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

(c) four-variable

By : Ms. Kanika Dhingra

Representation of Entries of truth table in k-map

Table 2.8 Truth table of a digital system

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Here, the output Y can be written as

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

$$Y(A,B,C) = m_1 + m_2 + m_4 + m_7$$

The K-map for the above three-variable expression is shown in Fig. 2.2.

Variables		$\overline{A}\,\overline{B}$ 00	$\overline{A}B$ 01	AB 11	$A\overline{B}$ 10
$\overline{C}$	0	0	1	0	1
C	1	1	0	1	0

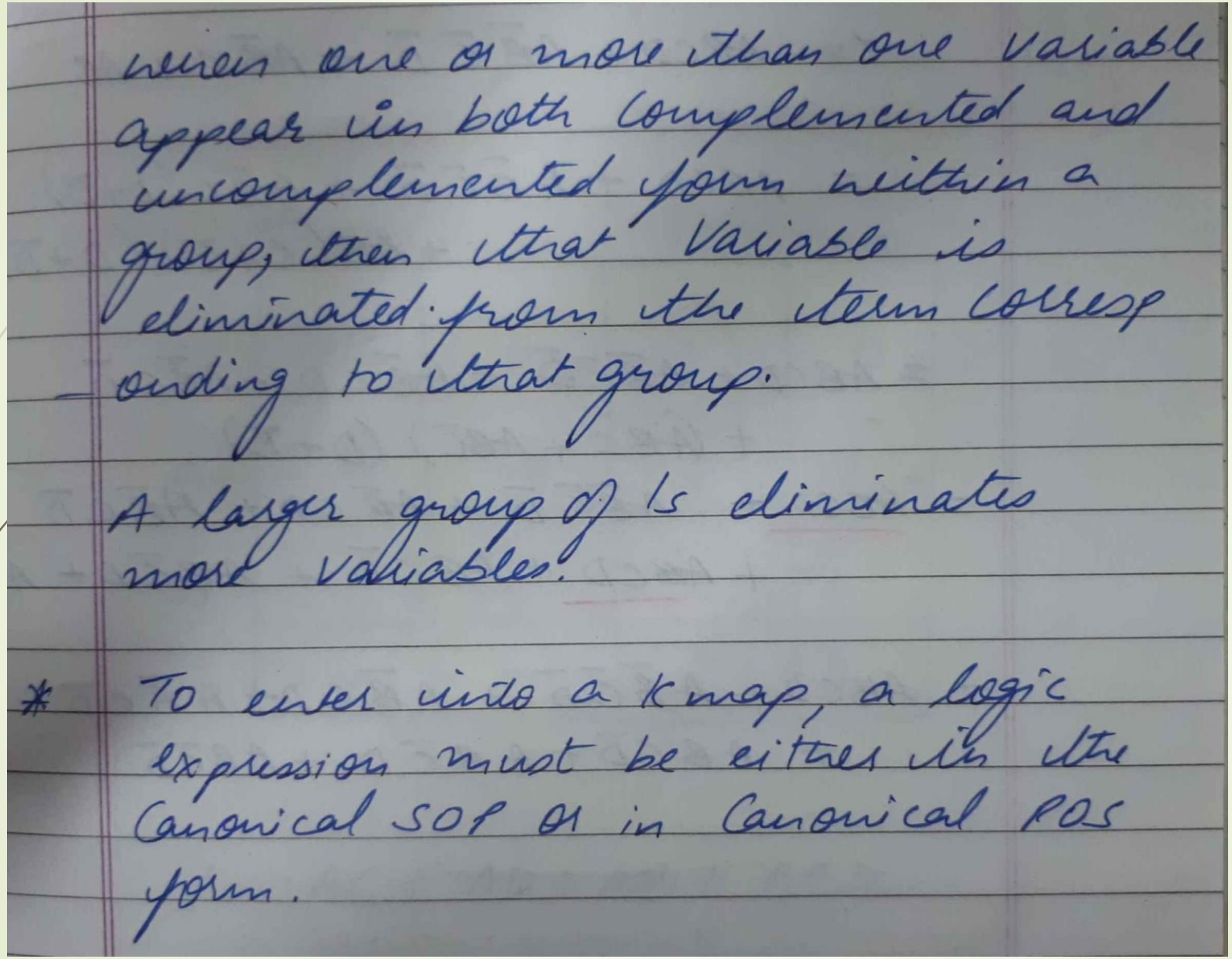
Fig. 2.2

By : Ms. Kanika Dhingra

SIMPLIFICATION OF BOOLEAN EQUATIONS USING K-map

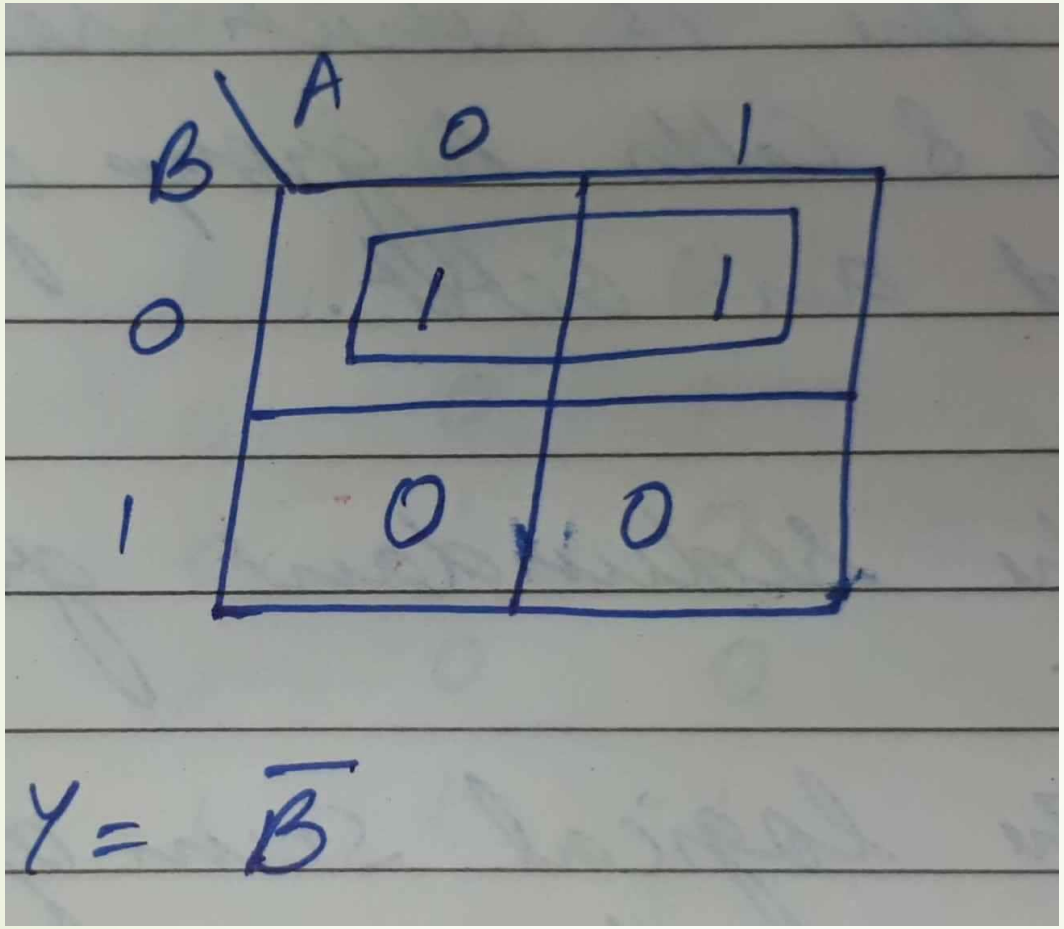
1. Construct the K-map and enter the 1s in those cells corresponding to the combinations for which function value is 1, then enter the 0s in the other cells.
2. Examine the map for 1s that cannot be combined with any other 1 cells and form groups with such single 1.
3. Next, look for those 1s which are adjacent to only one other 1 and form groups containing only 2 cells and which are not part of any group of 4 or 8 cells. A group of 2 cells is called a pair.
4. Group the 1s which results in groups of 4 cells but are not part of an 8-cells group. A group of 4 cells is called a quad.
5. Group the 1s which results in groups of 8 cells. A group of 8 cells is called an octet.
6. Form more pairs, quads and octets to include these 1s that have not yet been grouped, and use only a minimum number of groups. There can be overlapping of groups if they include common 1s.
7. Omit any redundant group.
8. Form the logical sum of all the terms generated by each group.

By : Ms. Kanika Dhingra



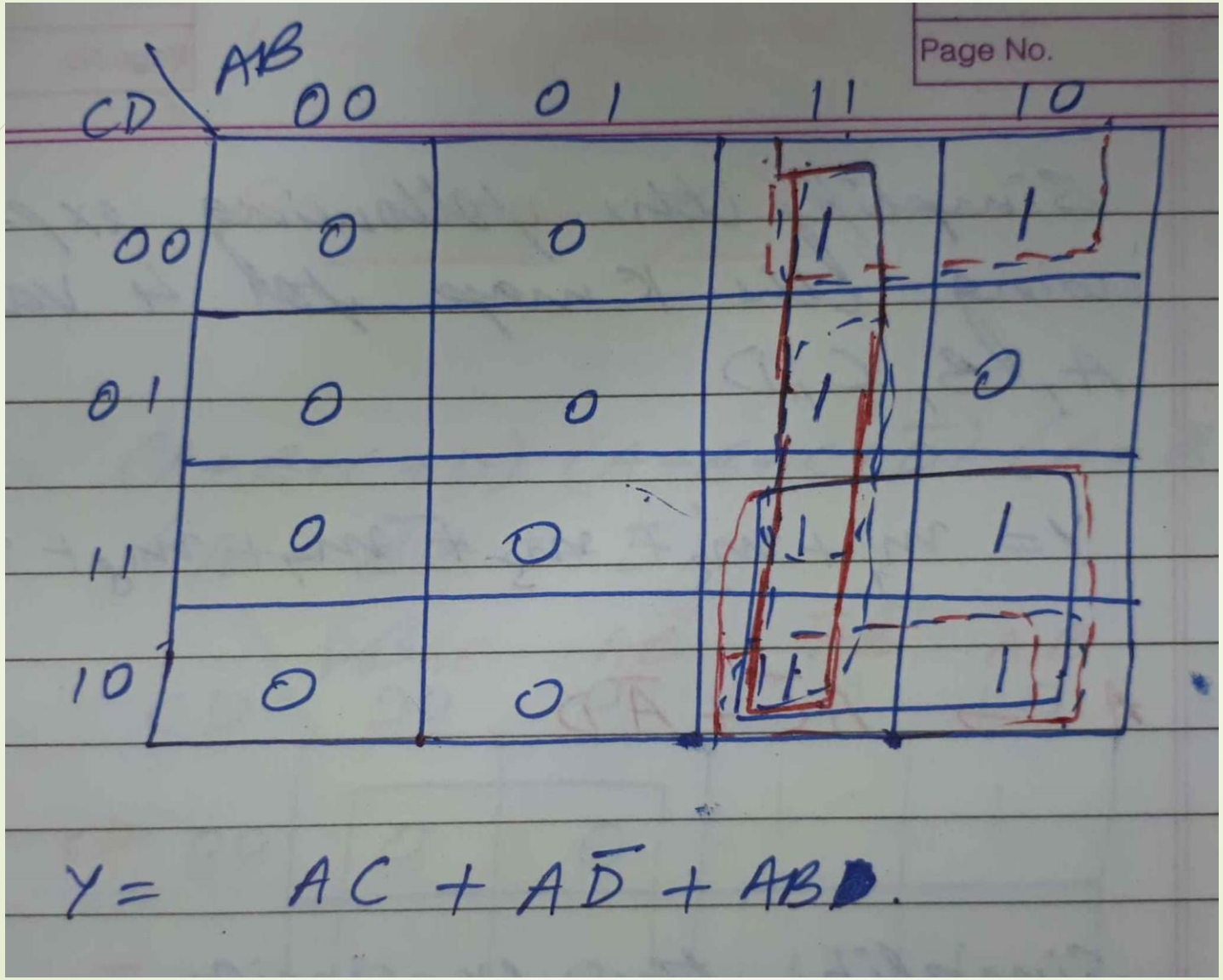
By : Ms. Kanika Dhingra

Simplify the expression $Y = \overline{A}\,\overline{B} + A\,\overline{B}$



By : Ms. Kanika Dhingra

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



By : Ms. Kanika Dhingra

Example 2.28 Simplify the following expression using the Karnaugh map for the 4-variables A, B, C and D.

$$Y = m_1 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{12} + m_{13}$$

Solution The K-map of the given equation is shown in Fig. E2.28. The expression is minimized using the following steps:

Step 1 Construct the K-map and enter 1 in the cells corresponding to the minterms present in the expression and 0 in the other cells.

Step 2 There are no 1s which are not adjacent to other 1s.

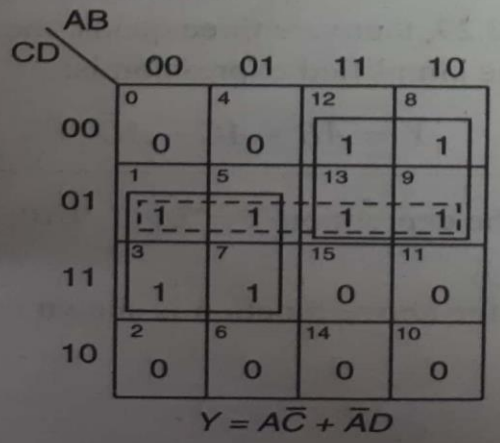
Step 3 There are no pairs which are not part of any larger groups.

Step 4 There are 2 quads. Cells 1, 3, 5 and 7 are grouped to form one quad and the second quad is made up of cells 12, 13, 8 and 9. The combinations corresponding to the cells in the first quad are $\bar{A}\bar{B}\bar{C}D$, $\bar{A}\bar{B}CD$, $\bar{A}B\bar{C}D$ and $\bar{A}BCD$. In the above group of four combinations, the variables $\bar{A}D$ are common in all the cells while B and C appear both in complemented and uncomplemented forms. From the preceding section, it is clear that only the variables that are the same in all the cells of the group must appear in the term corresponding to that group. Therefore, the minimized term for the first quad is $\bar{A}D$, and that of the second quad is $A\bar{C}$.

Step 5 There are no octets.

Step 6 All the 1s have already been grouped.

Step 7 The terms generated by the two groups are OR operated together to obtain the expression for Y as follows:



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

Example 2.29 Plot the logical expression $ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + AB$ on a 4-variable K-map; obtain the simplified expression from the map.

Solution To enter into a K-map, a logic expression must be either in the canonical SOP form or in the canonical POS form. The canonical SOP form of the given expression can be obtained as follows:

$$
\begin{aligned}
Y &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + AB \\
&= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C(D + \overline{D}) + AB(C + \overline{C})(D + \overline{D}) \\
&= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + (ABC + AB\overline{C})(D + \overline{D}) \\
&= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} \\
&= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} \\
&= m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12} \\
&= \Sigma_m(8,10,11,12,13,14,15)
\end{aligned}
$$

The K-map for the above expression is shown in Fig.E2.29.

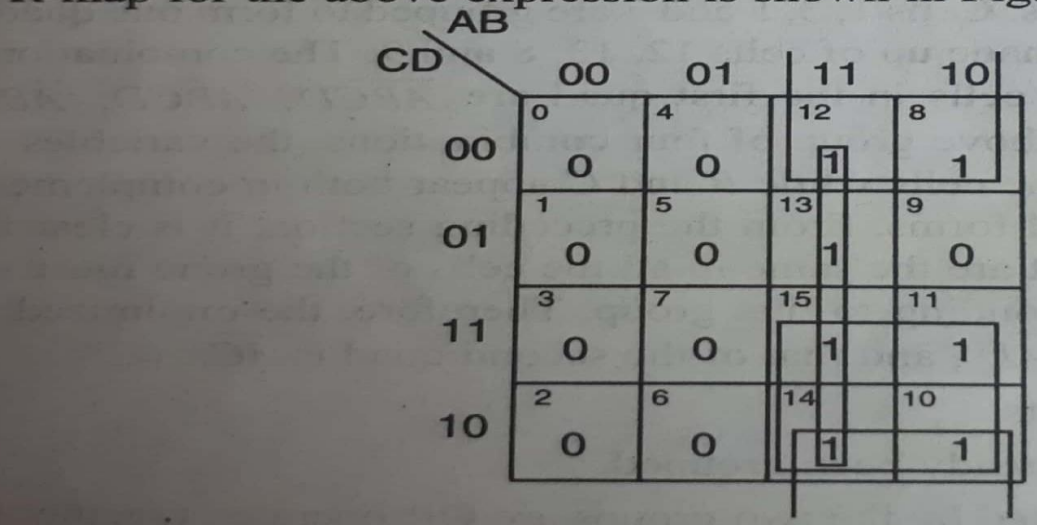


Fig. E2.29

In the K-map in Fig. E2.29, there are three quads; the minimised terms for them are AB, AC and $A\overline{D}$ and the simplified expression is:

$$Y = AB + AC + A\overline{D}$$

Example 2.31 Simplify the expression $Y = m_1 + m_5 + m_{10} + m_{11} + m_{12} + m_{13} + m_{15}$, using the K-map method.

Solution The K-map for the above expression is shown in Fig. E2.31(a).

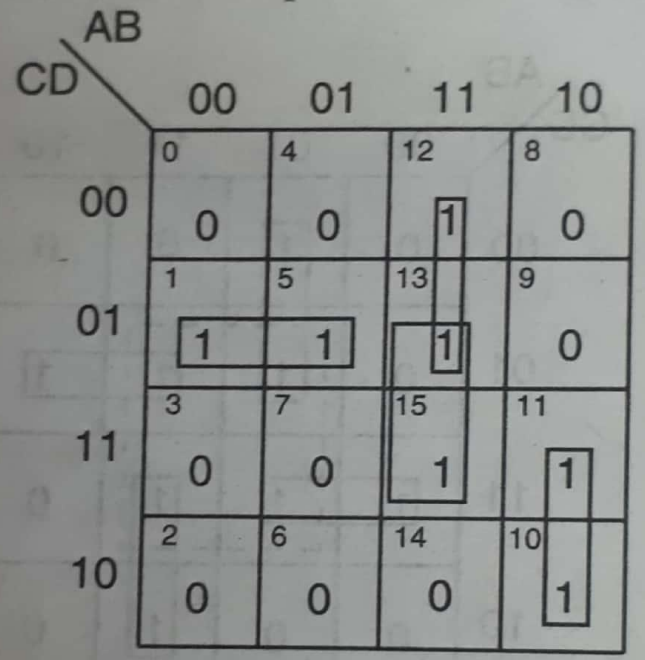


Fig. E2.31(a)

As shown in Fig. E2.31(a), the K-map contains four pairs but no quads or octets; the corresponding simplified expression is given by

$$Y = \overline{A}\,\overline{C}D + AB\overline{C} + ABD + A\overline{B}C \qquad (1)$$

It is important to note that the simplified expression obtained from the K-map is not unique. This can be explained by grouping the pairs in a different manner as shown in Fig. E2.31(b).

Example 2.32 Simplify the expression $Y = \Sigma_m(3,4,5,7,9,13,14,15)$, using the K-map method.

Solution The K-map for the above function is shown in Fig. E2.32.

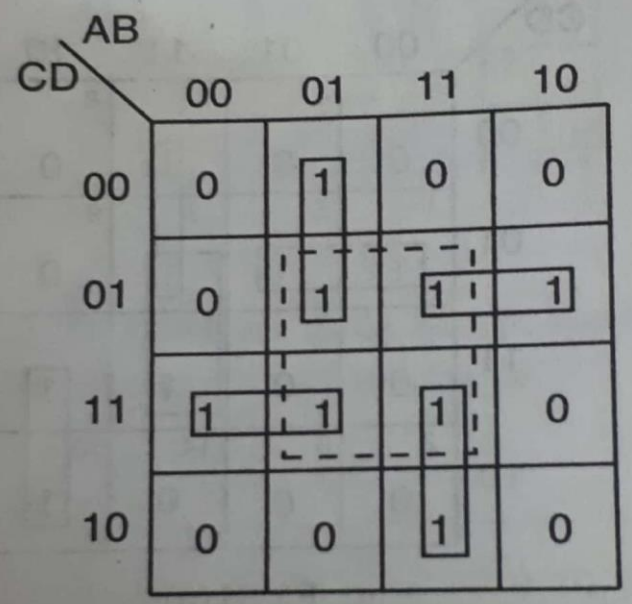


Fig. E2.32

In the above K-map, the cells 5,7,13 and 15 can be grouped to form a quad as indicated by the dotted lines. In order to group the remaining 1s, four pairs have to be formed as shown in Fig. E2.32. However, all the four 1s covered by the quad are also covered by the pairs. So, the quad in the above K-map is redundant. Therefore, the simplified expression will be

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

{ 0 → Uncomplement form of Variable
  1 → complemented

Date:
Page No.

For Product of Sum.

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

CD \ AB	AB 00	$A\bar{B}$ 01	$\bar{A}\bar{B}$ 11	$\bar{A}B$ 10
CD 00	0	0	1	1
$C\bar{D}$ 01	0	0	1	1
$\bar{C}\bar{D}$ 11	1	1	1	1
$\bar{C}D$ 10	1	1	1	1

$Y = (A+C)$

In sum of product: We were adding the result of each group.

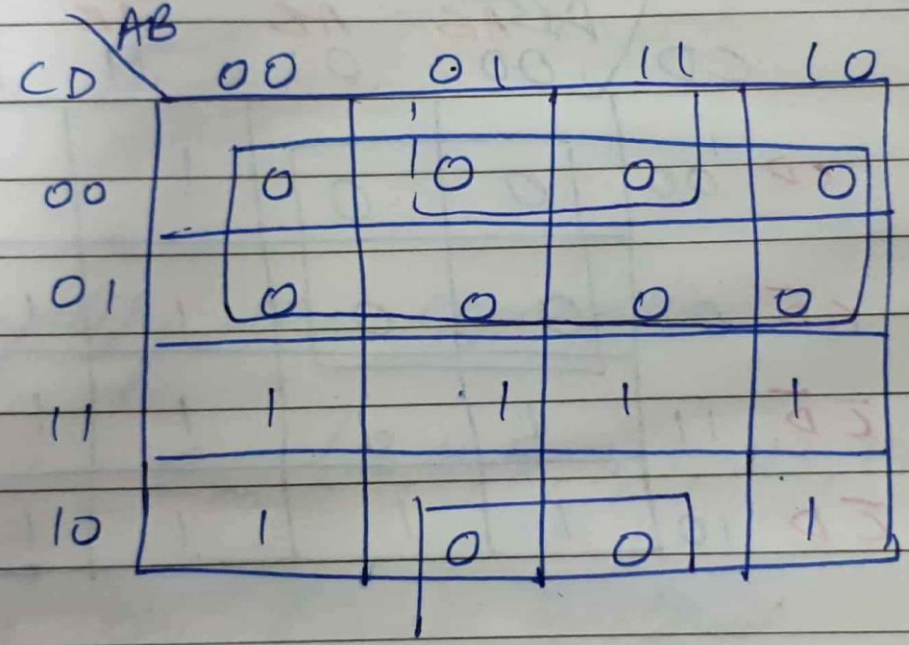
In Product of Sum: we are multiplying the result of each group.

By : Ms. Kanika Dhingra

Example: Simplify the expression

$$Y = \pi(0, 1, 4, 5, 6, 8, 9, 12, 13, 14)$$

using K map method.



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

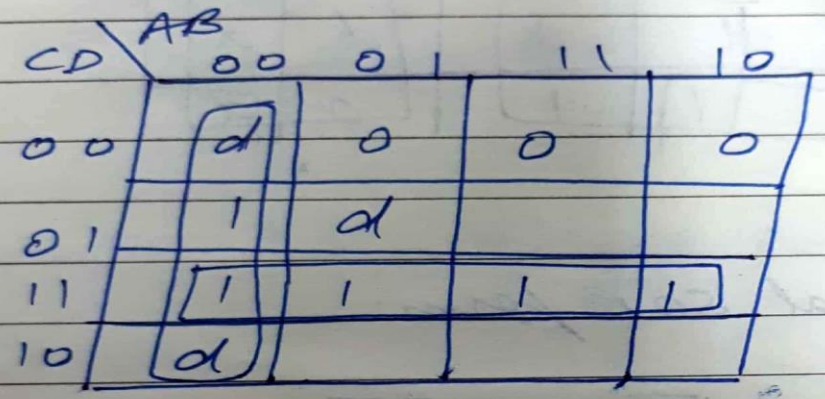
By : Ms. Kanika Dhingra

Don't Care Combinations

Example: Simplify the Boolean function

$F(A,B,C,D) = \Sigma_m (1,3,7,11,15)$

$+ \Sigma_d (0,2,5)$



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

The d in cell 5 is left free since it doesnot contribute in increasing the size of any group.

Example 2.38 *Using the K-map method, simplify the following Boolean function and obtain (i) minimal SOP and (ii) minimal POS expressions:*

$$Y = \Sigma_m (0,2,3,6,7) + \Sigma_d (8,10,11,15)$$

Solution The K-map for the above function is shown in Fig.E2.38.

Minimal SOP form By combining the 1s and **d** s as shown in the K-map, there are two quads; the simplifed SOP expression is given by

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

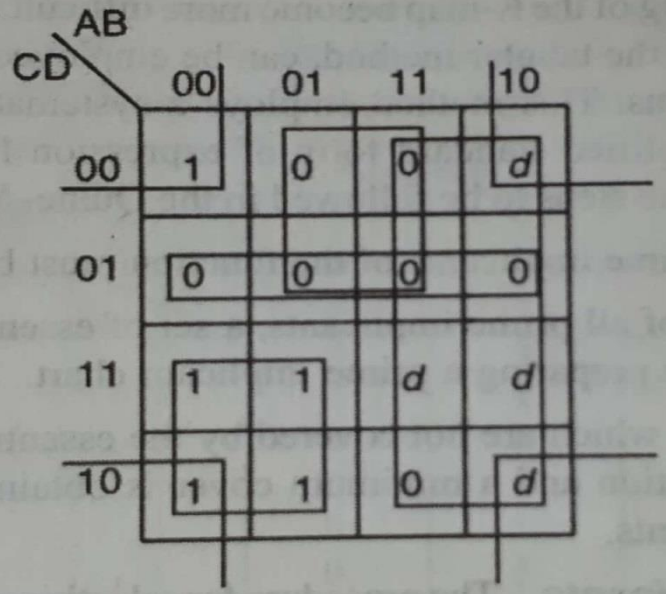
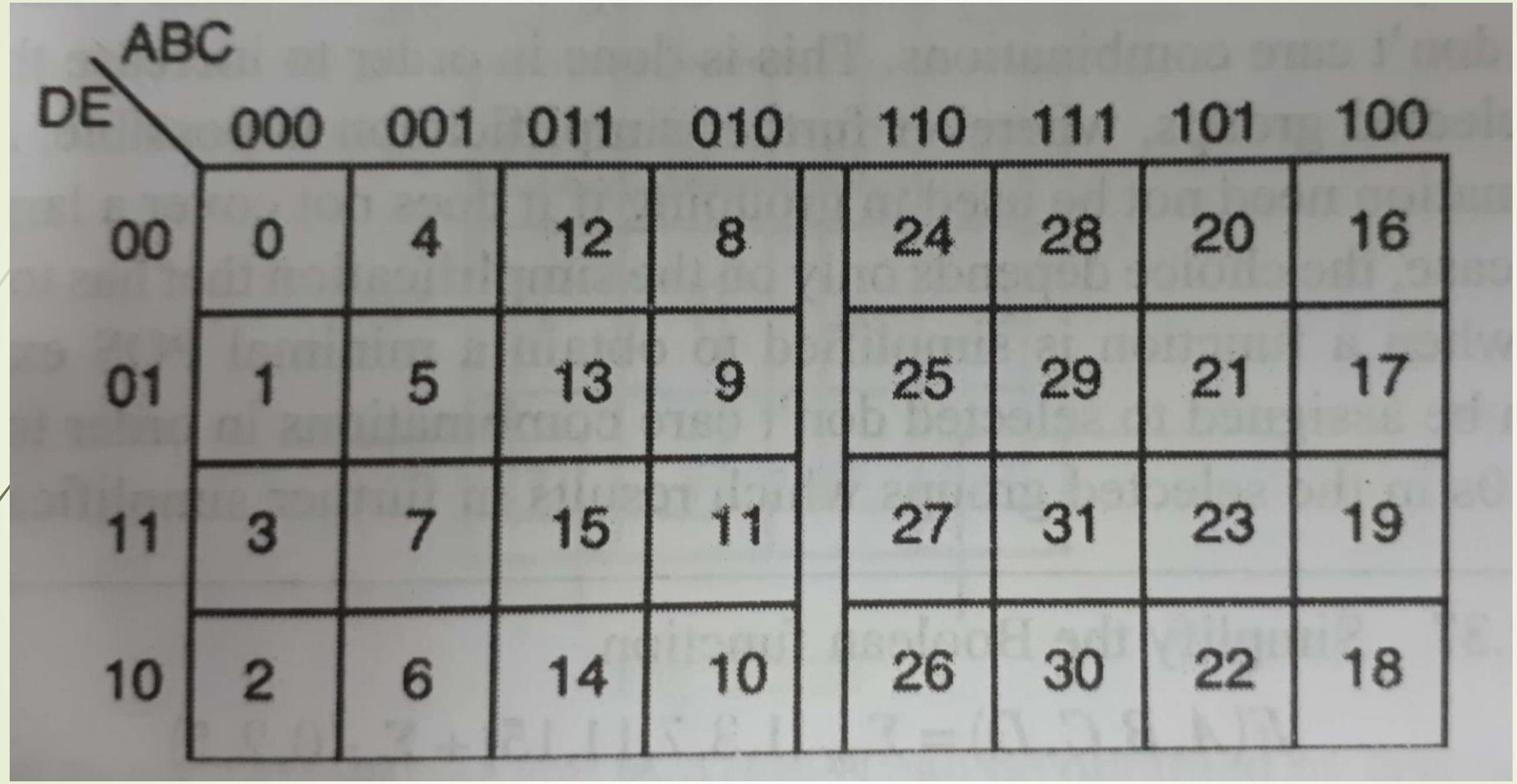


Fig. E2.38

Minimal POS form One octet and two quads can be obtained by combining the 0s and d s as shown in the K-map; the simplifed POS expression is given by

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

Five Variable K map



DE \ ABC	000	001	011	010	110	111	101	100
00	0	4	12	8	24	28	20	16
01	1	5	13	9	25	29	21	17
11	3	7	15	11	27	31	23	19
10	2	6	14	10	26	30	22	18

By : Ms. Kanika Dhingra

Example 2.36 Simplify $Y = \Sigma_m(3,6,7,8,10,12,14,17,19,20,21,24,25,27,28)$ using the K-map method.

Solution In the 5-variable K-map shown in Fig.E2.36, there are three quads and three pairs; the simplified expression is given by

$$Y = B\overline{D}\overline{E} + A\overline{C}E + \overline{A}B\overline{E} + A\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}DE$$

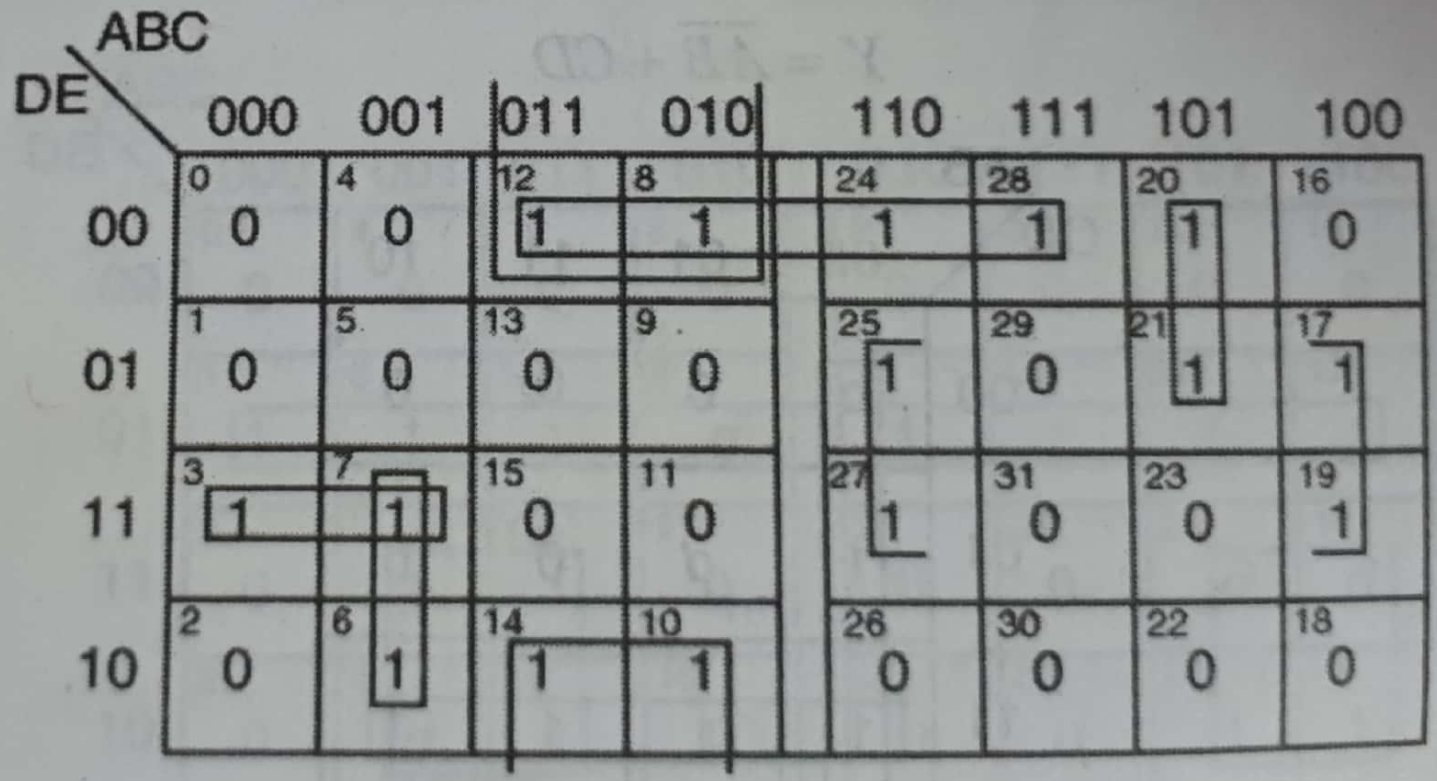


Fig. E2.36 5-variable K-map

Arithmetic Circuits

A digital system consists of two types of circuits

i) Combinational Circuits

ii) Sequential Circuits

- **In a combinational circuit output at any time depends only on the input values at that time.**
- **In a sequential circuit output at any time depends not only on the present input values but also on the past output values.**
- **The arithmetic circuits such as adders, subtractors, multiplier, divider are combinational circuits.**
- **The basic building blocks of the arithmetic unit in a digital system are adders.**

By : Ms. Kanika Dhingra

Procedure for the design of Combinational Circuits

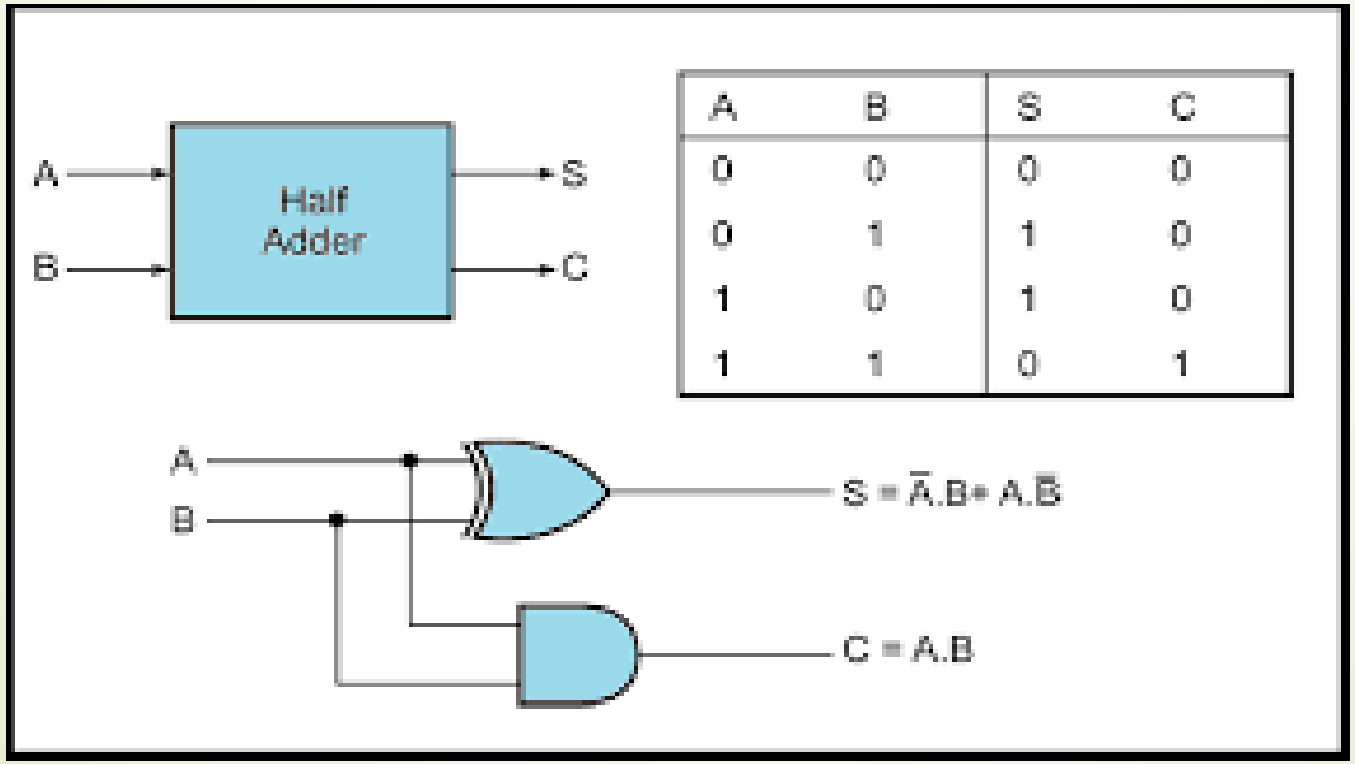
Any combinational circuit can be designed by following the design procedure given below:

1. From the word description of the problem, identify the inputs and outputs and draw a block diagram.
2. Draw a truth table such that it completely describes the operation of the circuit for different combinations of inputs.
3. Write 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

By : Ms. Kanika Dhingra

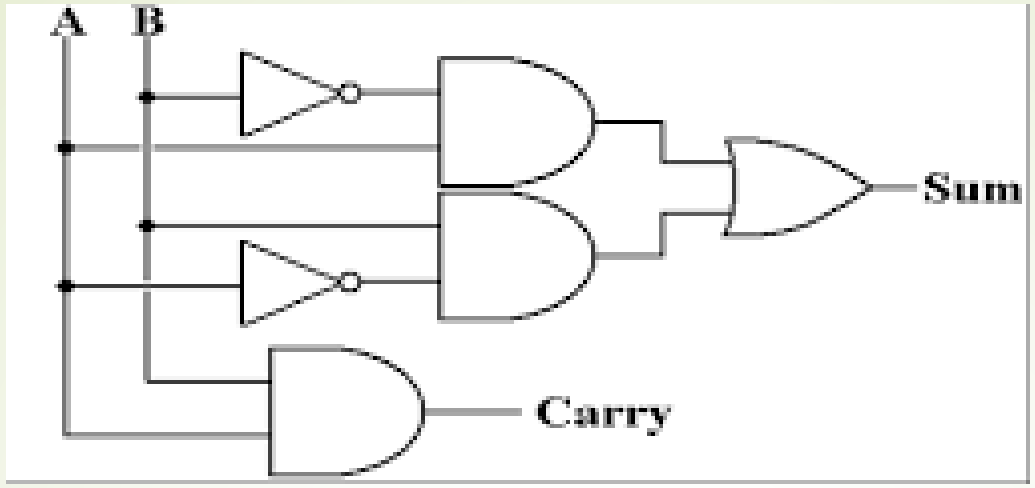
Half Adder

- **The simplest combinational circuit which performs the arithmetic addition of two binary digits is called a half adder.**
- **It has two inputs and two outputs. Inputs are two bits A and B and outputs are Sum(S) and Carry(C).**



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

By : Ms. Kanika Dhingra



Using Basic Gates

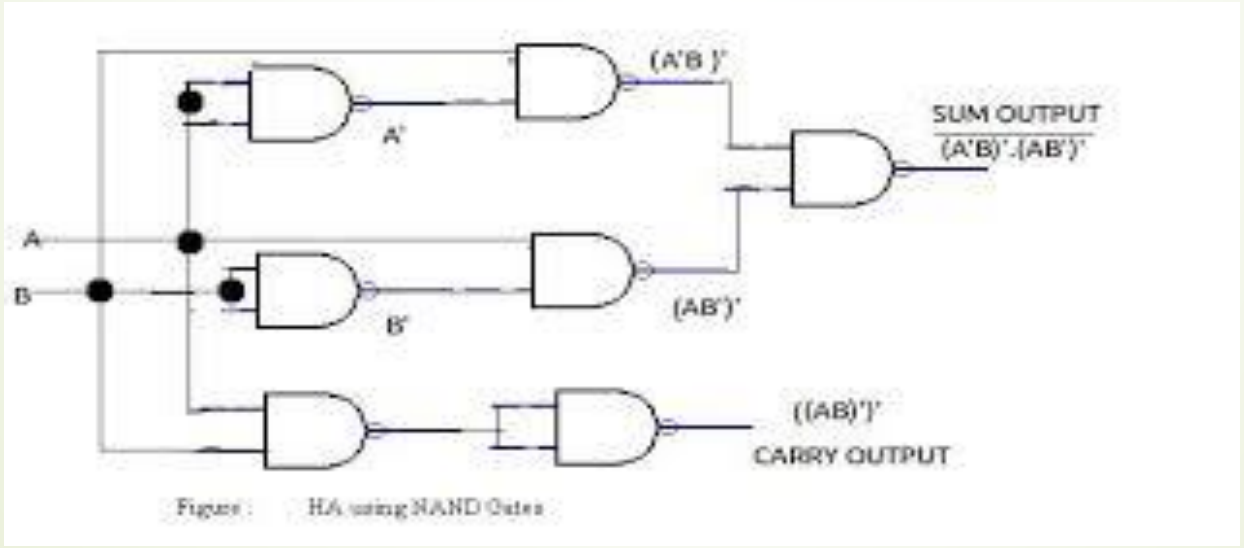


Figure : HA using NAND Gates

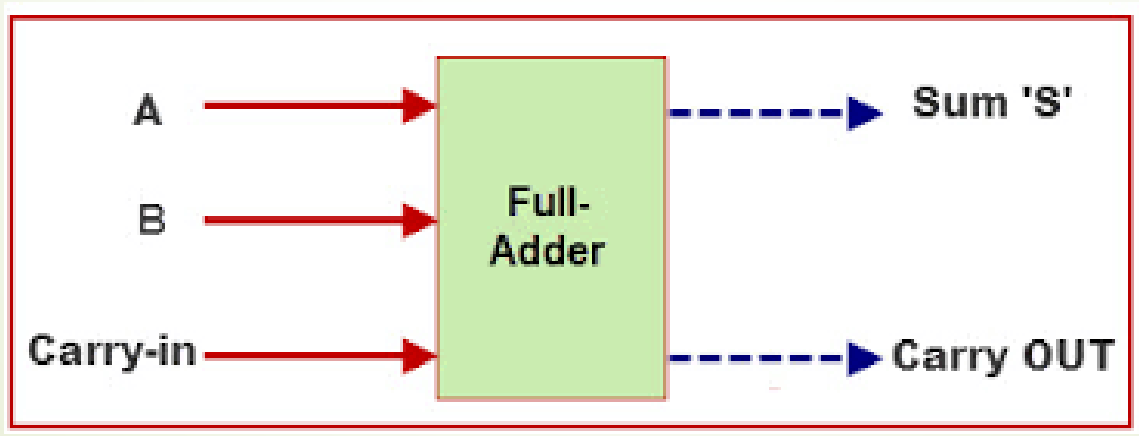
Using NAND Gates

By : Ms. Kanika Dhingra

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, full adder is designed.

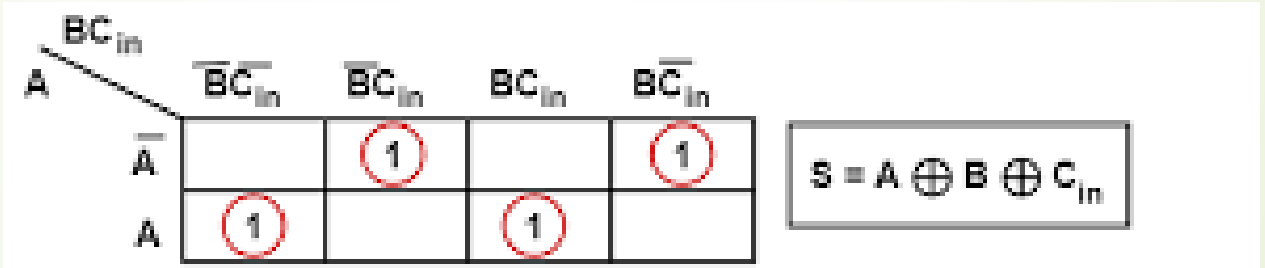
- A Full Adder has three inputs ie. Augend bit, Addend bit and a carry input; it has two outputs a sum and a carry output(Cout).



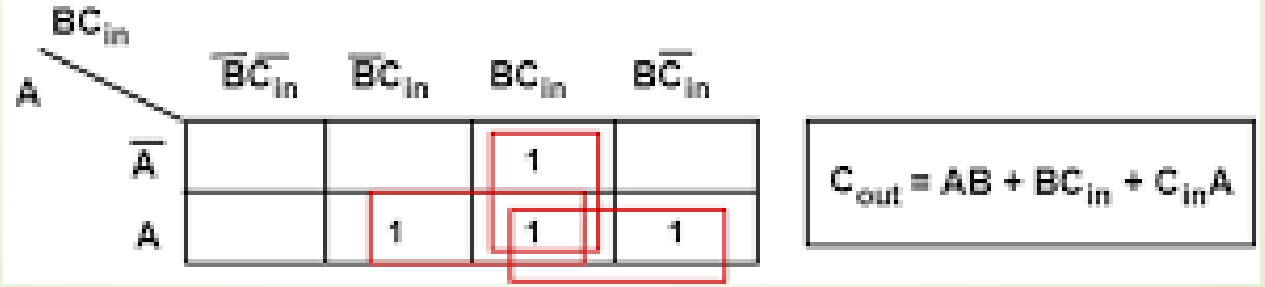
By : Ms. Kanika Dhingra

Inputs			Outputs	
A	B	C – IN	Sum	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

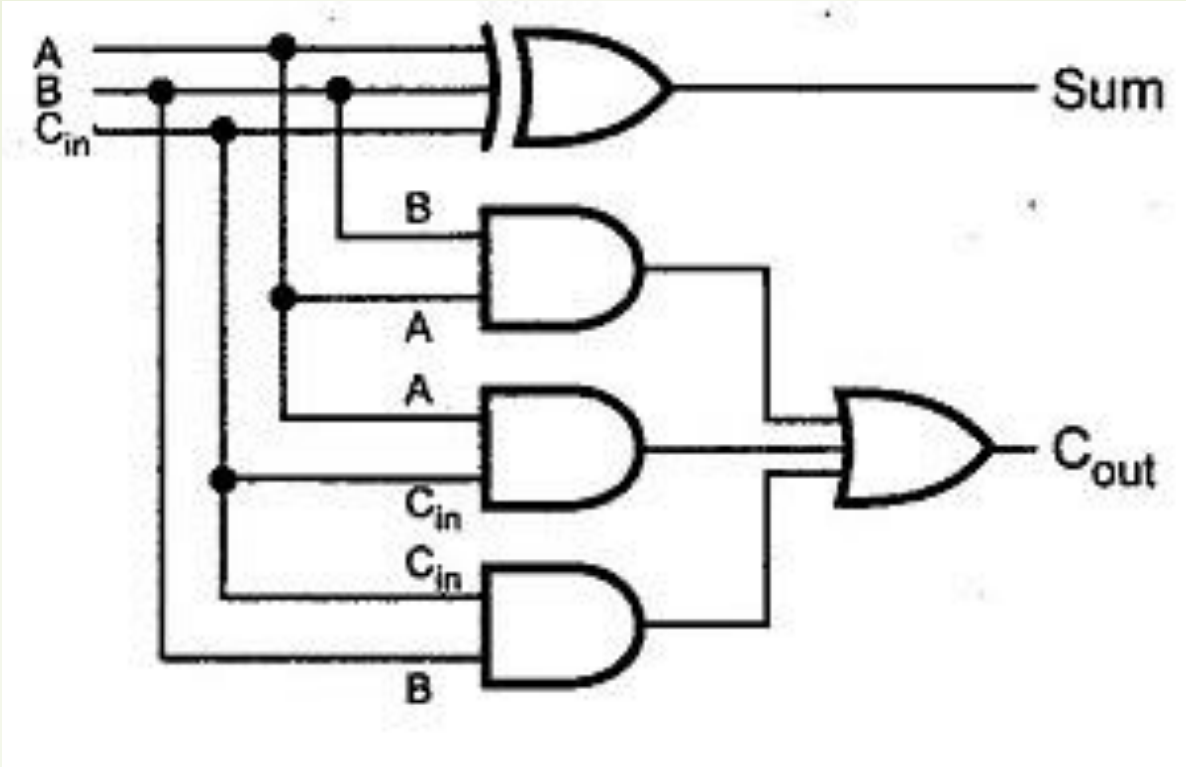
For Sum:



For Cout:



By : Ms. Kanika Dhingra



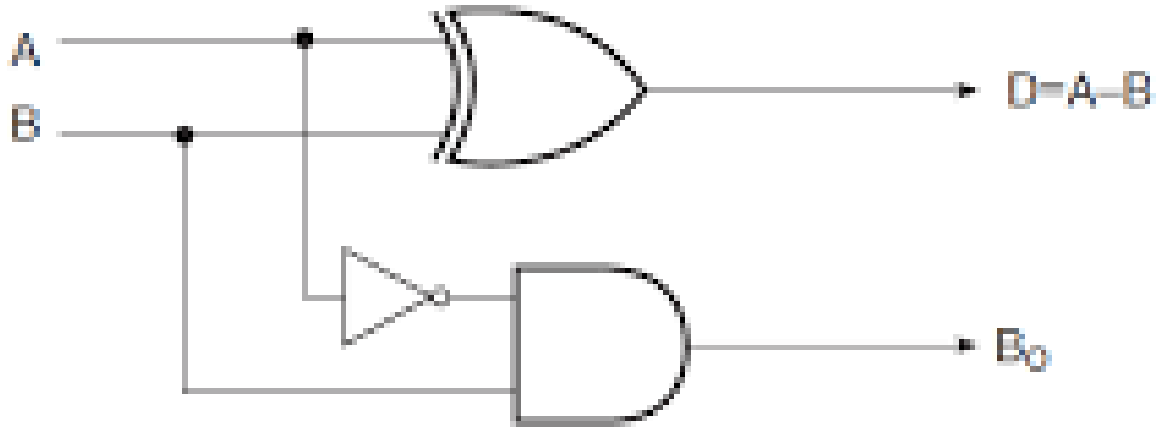
By : Ms. Kanika Dhingra

Half Subtractor

Minuend

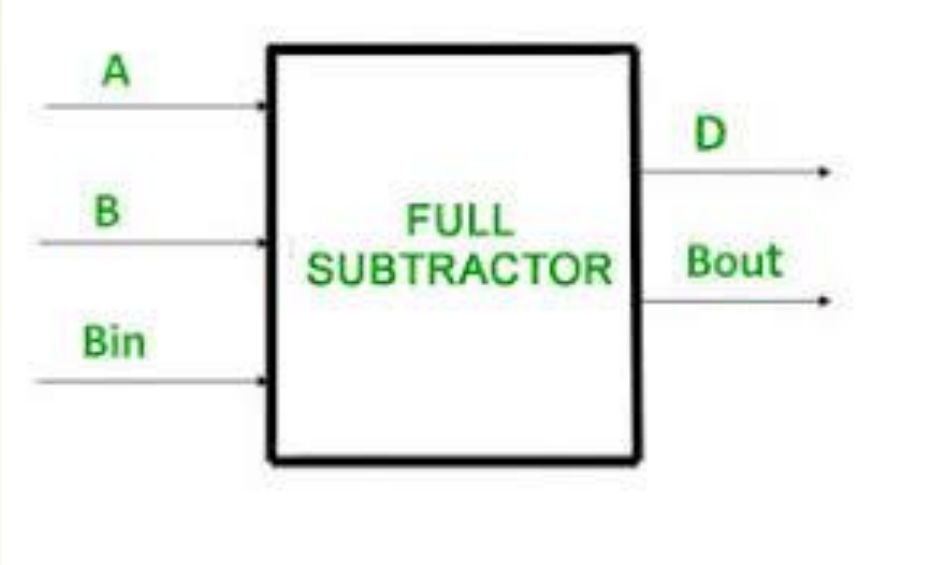
Subtrahend

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



By : Ms. Kanika Dhingra

Full Subtractor



By : Ms. Kanika Dhingra

Minuend

Subtrahend

INPUT			OUTPUT	
A	B	Bin	D	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

By : Ms. Kanika Dhingra

For D

A \ BB_{in}	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

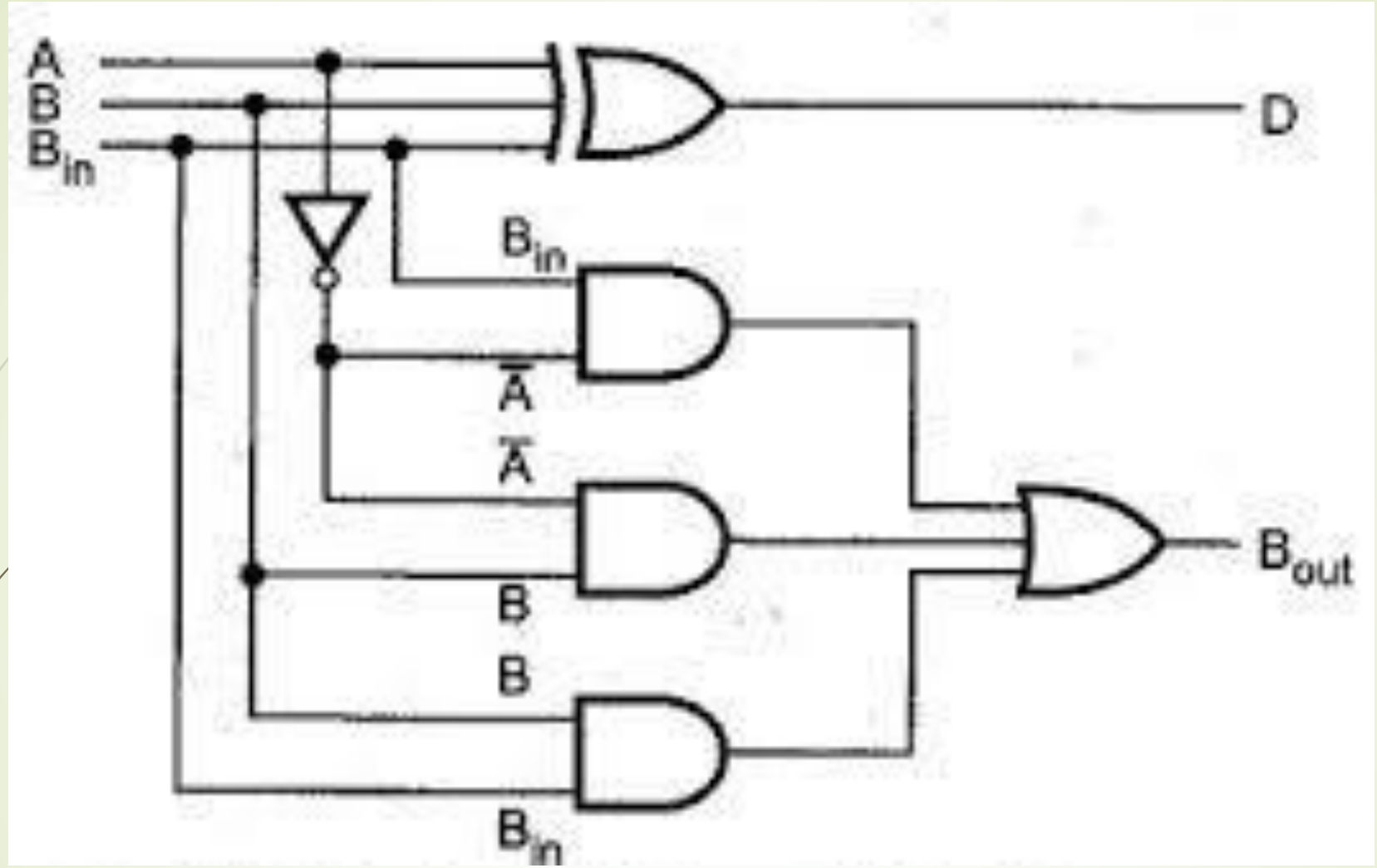
$D = \overline{A}\overline{B}B_{in} + \overline{A}B\overline{B}_{in} + A\overline{B}\overline{B}_{in} + ABB_{in}$

For B_{out}

A \ BB_{in}	00	01	11	10
0	0	1	1	1
1	0	0	1	0

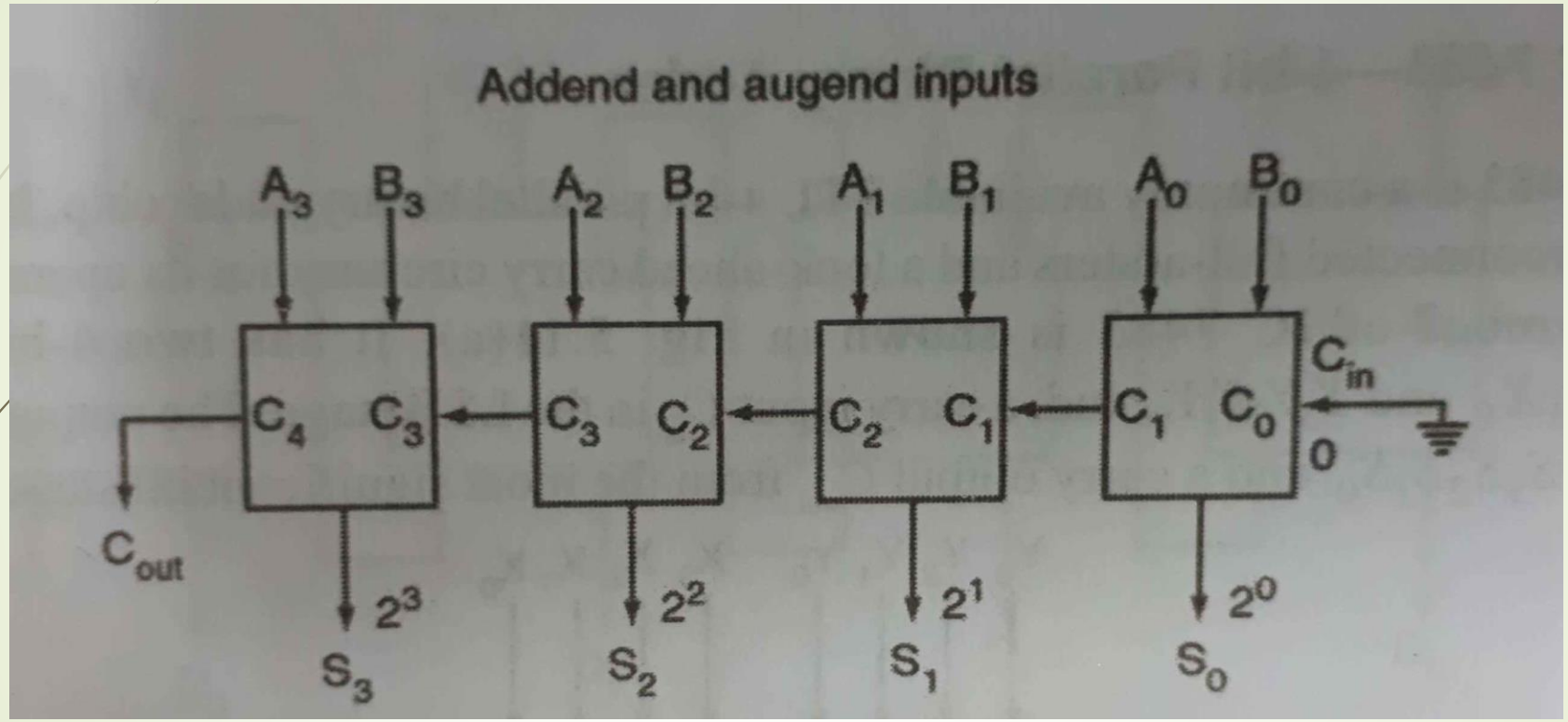
$B_{out} = \overline{A}B_{in} + \overline{A}B + BB_{in}$

By : Ms. Kanika Dhingra



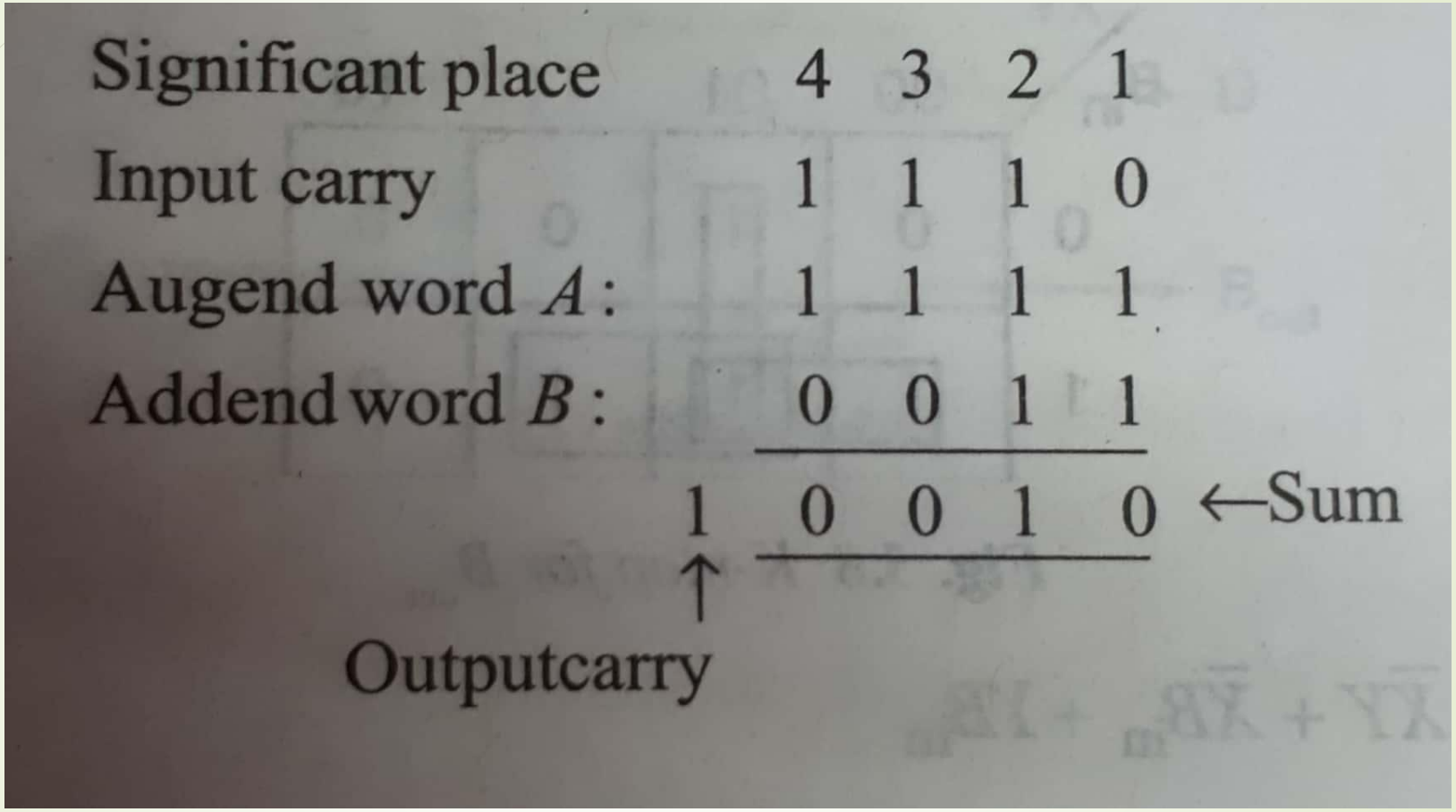
By : Ms. Kanika Dhingra

4-bit Parallel Binary Adder



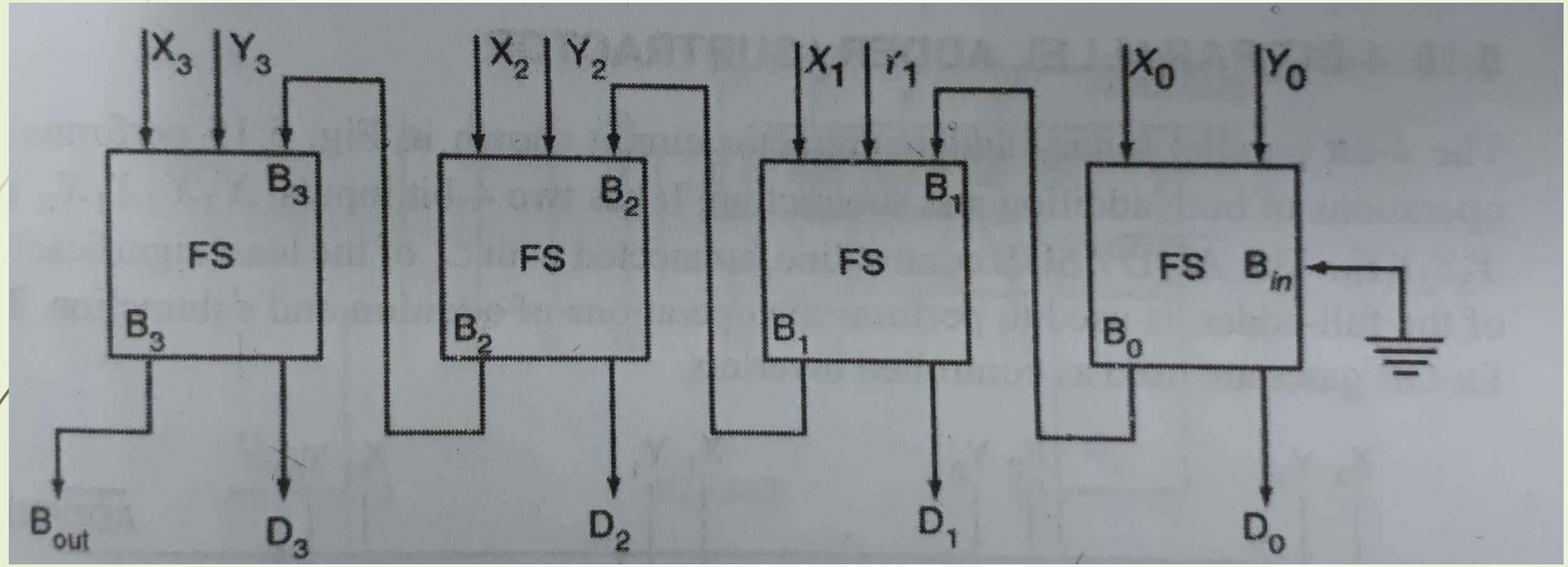
By : Ms. Kanika Dhingra

Let the 4-bit words to be added be represented by $A_3A_2A_1A_0$ = 1111 and $B_3B_2B_1B_0$ = 0011



By : Ms. Kanika Dhingra

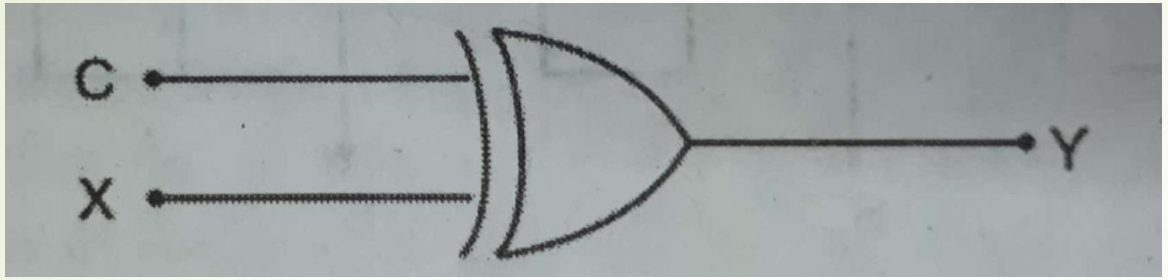
4-bit Parallel Binary Subtractor

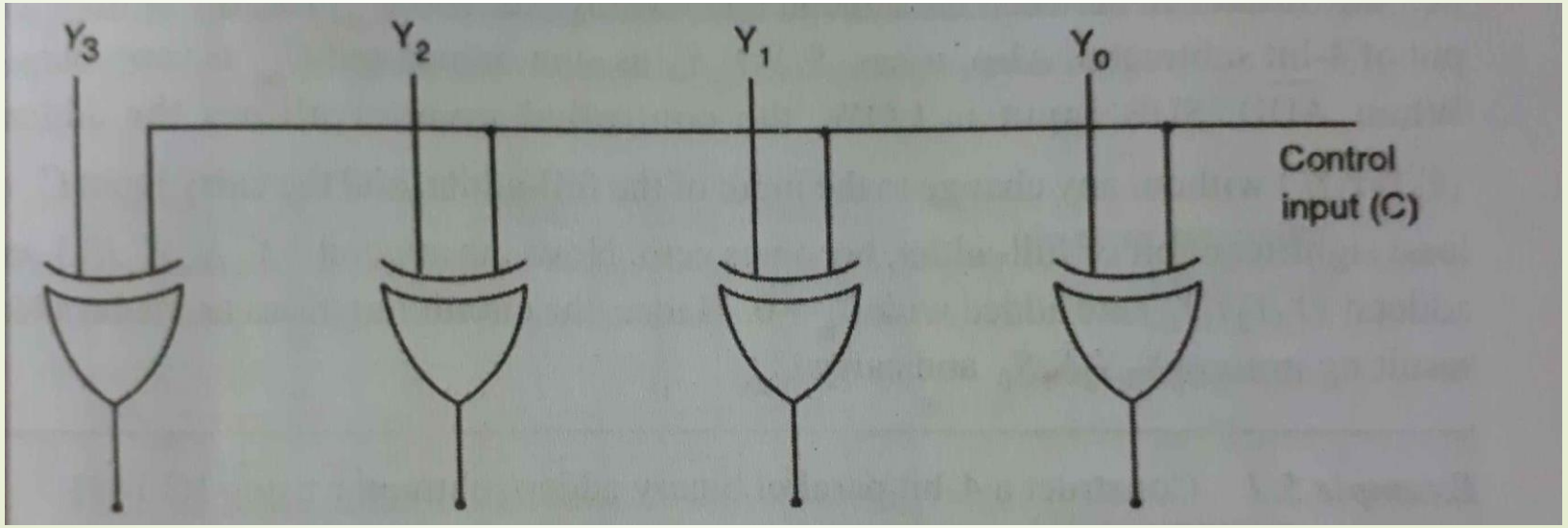


By : Ms. Kanika Dhingra

Controlled Inverter

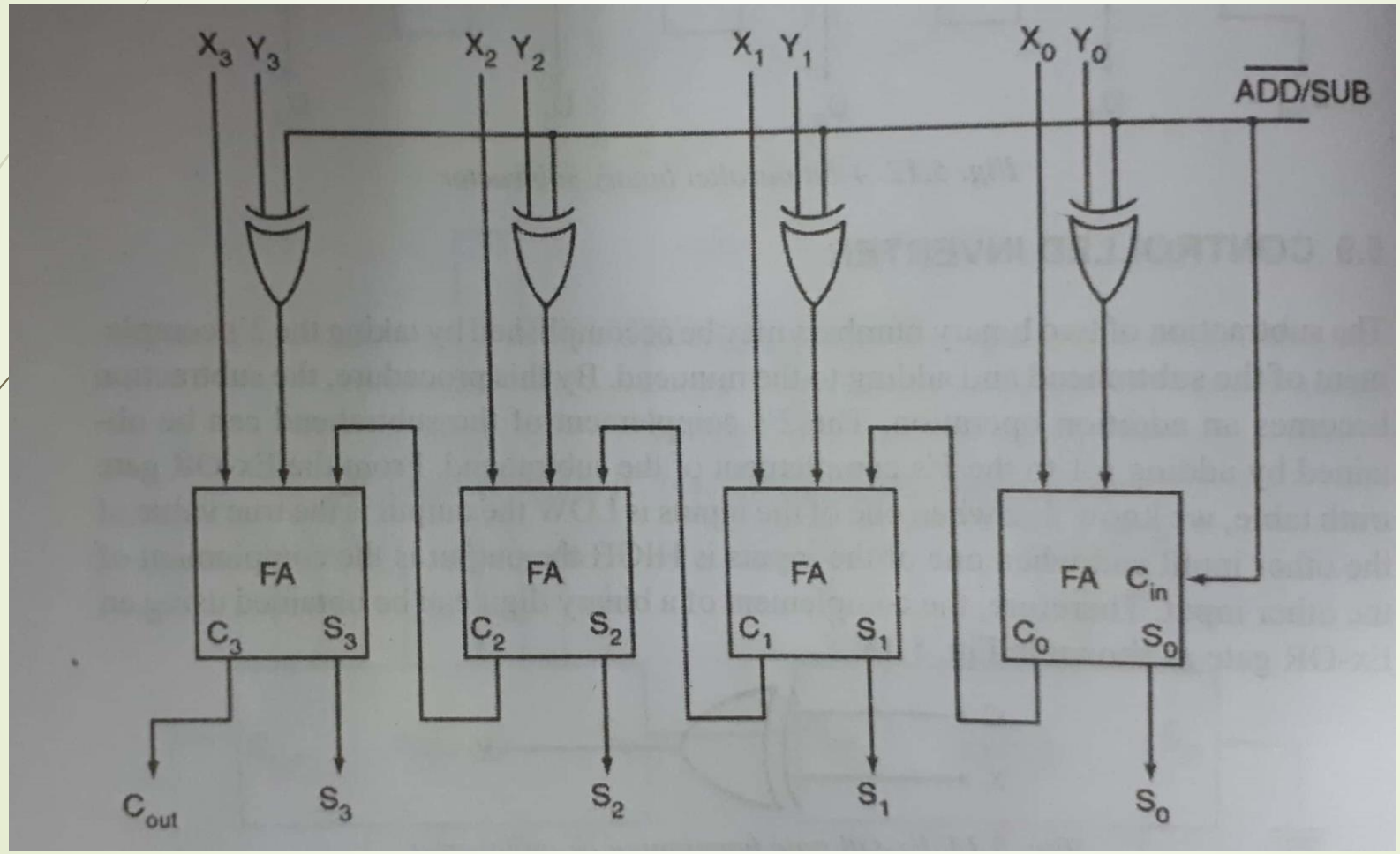
In case of Ex-OR gate when one of the inputs is low the output is true value of the other input and when one of the inputs is high the output is the complement of the other input.





By : Ms. Kanika Dhingra

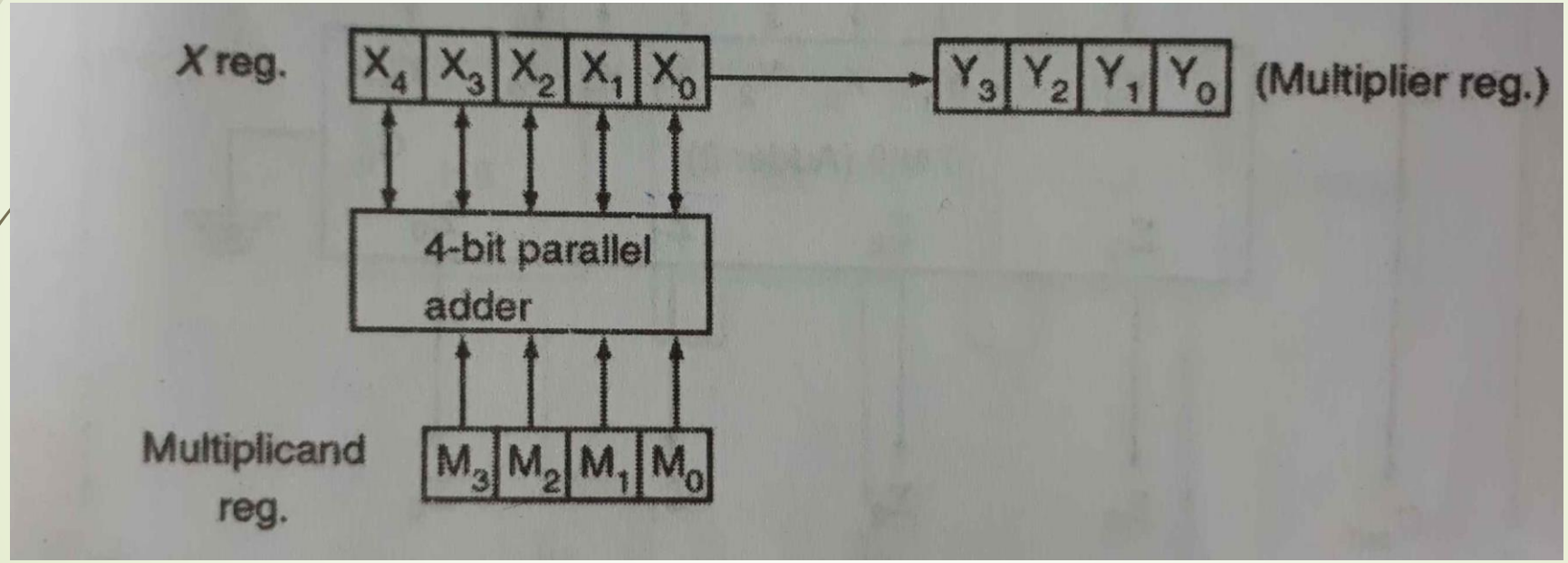
4-Bit Parallel Adder/Subtractor



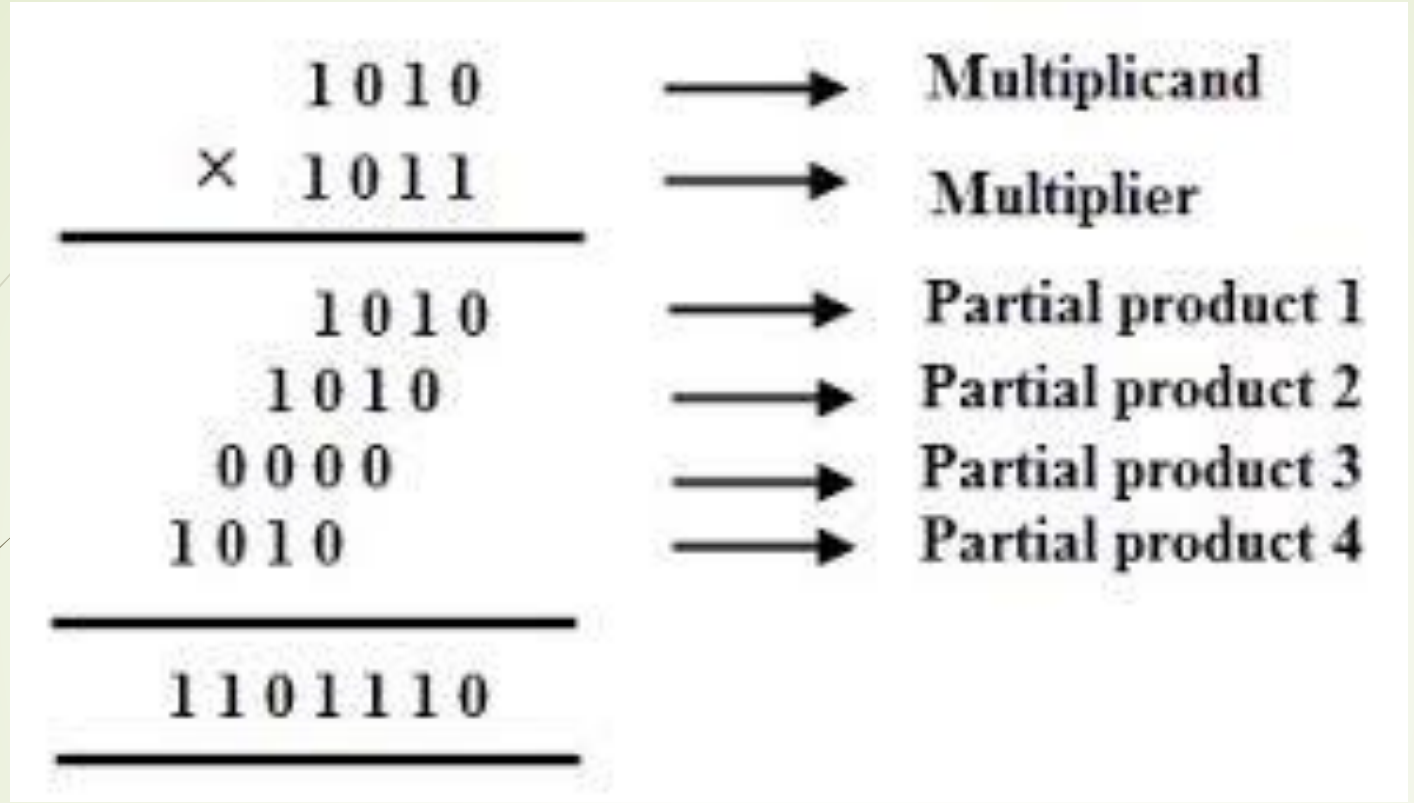
By : Ms. Kanika Dhingra

Binary Multiplier

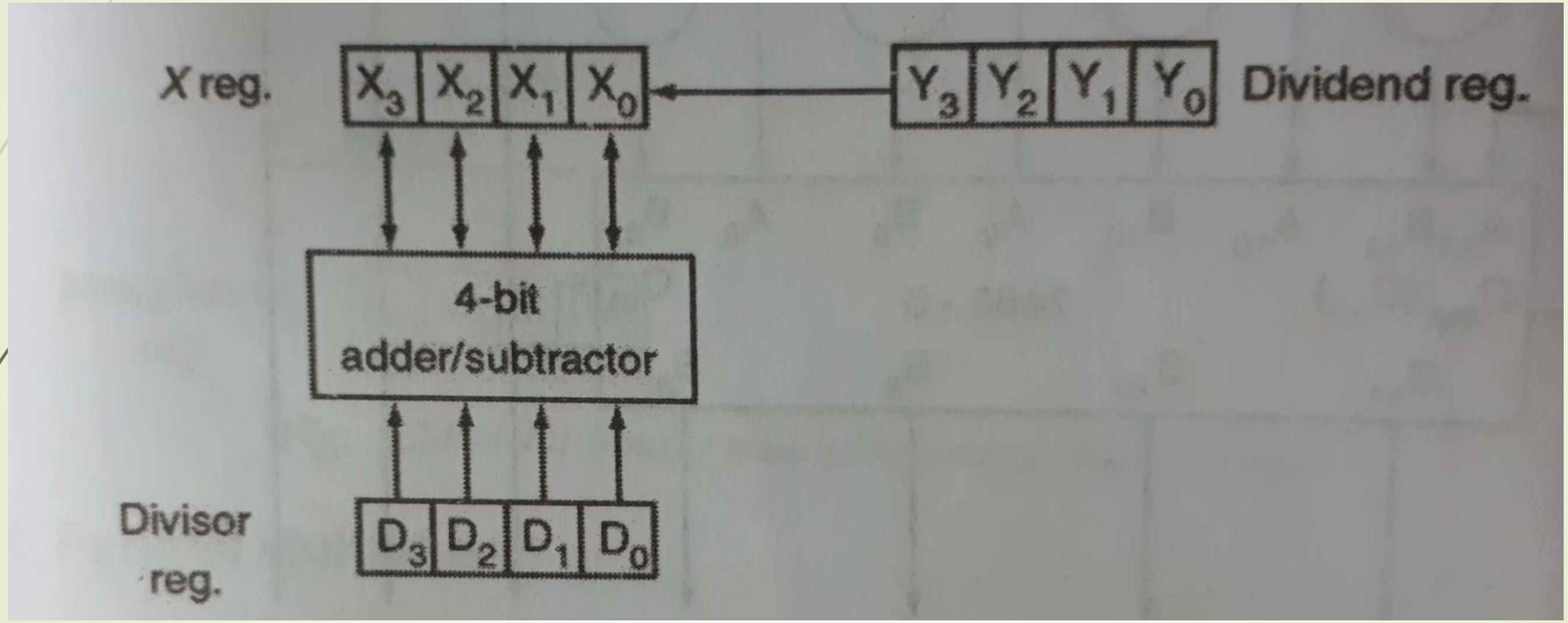
Multiplication Using Shift Method



By : Ms. Kanika Dhingra


1 0 1 0
× 1 0 1 1
1 0 1 0
1 0 1 0
0 0 0 0
1 0 1 0
1 1 0 1 1 1 0
Multiplicand
Multiplier
Partial product 1
Partial product 2
Partial product 3
Partial product 4

Binary Divider



By : Ms. Kanika Dhingra

Procedure for Division operation:

1. Shift the combined content of X and Y registers to the left by one bit.
2. Perform trial subtraction by subtracting the content of D register from the content of X register.
3. If there is no borrow in the previous subtraction, put 1 in the LSB of Y register, else restore the original content of X register by adding the contents of D register with the contents of X register.
4. Repeat steps 1 to 3 for n times, where n is the number of bits in the dividend. For a 4-bit division, $n = 4$.

Now the quotient will be available in the Y register and the remainder will be in the X register.

By : Ms. Kanika Dhingra

Consider the division of 1011 (11) by 0011(3). The dividend and divisor are stored in Y and D registers respectively, and the X register is initially cleared to 0. Therefore,

$$Y_3Y_2Y_1Y_0 = 1011$$
$$X_3X_2X_1X_0 = 0000$$
$$D_3D_2D_1D_0 = 0011$$

Here, both divisor and dividend are 4-bit numbers. Therefore, steps 1–3 have to be repeated four times.

I cycle

Step 1 Shift the combined contents of X and Y to the left by one bit. Therefore,

$$XY = 0001 \quad 0110$$

Step 2 Subtract dividend 0011 from register X resulting in

$$X = 1110 \quad \text{with borrow} = 1$$

Step 3 Since borrow = 1, restore the original content in X register by adding dividend 0011 with the content of X register 1110. Now,

$$XY = 0001 \quad 0110$$

II cycle

Step 1 Shift the combined contents of X and Y to the left by one bit. Therefore,

$$XY = 0010 \quad 1100$$

Step 2 Subtract dividend 0011 from X register resulting in

$$X = 1111 \quad \text{with borrow} = 1$$

Step 3 Since borrow = 1, restore the original content in X register by adding dividend 0011 with the content of X register 1111. Now,

$$XY = 0010 \quad 1100$$

III cycle

Step 1 Shift the combined contents of X and Y to the left by one bit. Therefore,

$$XY = 0101 \quad 1000$$

Step 2 Subtract dividend 0011 from register X which results in

$$X = 0010 \text{ with borrow} = 0$$

Step 3 Since borrow = 0, put 1 in the LSB of *Y* register (Y_0). Therefore,

$$XY = 0010 \quad 1001$$

IV cycle

Step 1 Shift the combined contents of *X* and *Y* to the left by one bit. Therefore,

$$XY = 0101 \quad 0010$$

Step 2 Subtract dividend 0011 from *X* register resulting in

$$X = 0010 \text{ with borrow} = 0$$

Step 3 Since borrow = 0, put 1 in the LSB of *Y* register (Y_0). Therefore,

$$XY = 0010 \quad 0011$$

Now, the quotient 0011(3) is available in the *Y* register and the remainder 0010(2) is available in the *X* register.

MULTIPLEXERS

- **Multiplex means “many to one”.**
- **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.**
- **Multiplexer has several data input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines.**

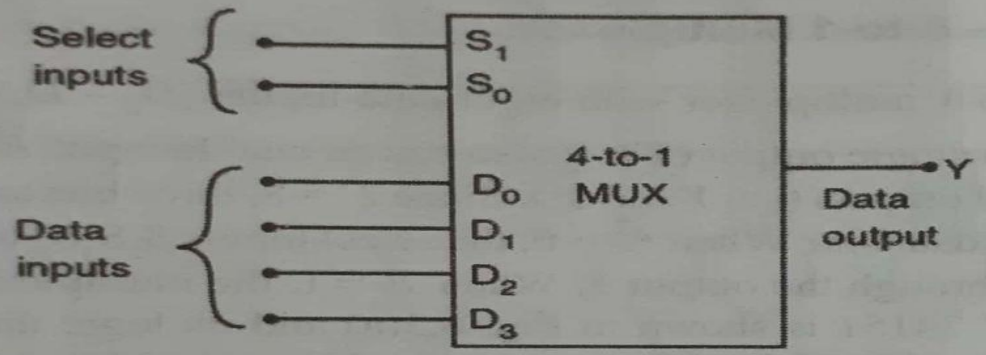
By : Ms. Kanika Dhingra

4 – to – 1 Multiplexer

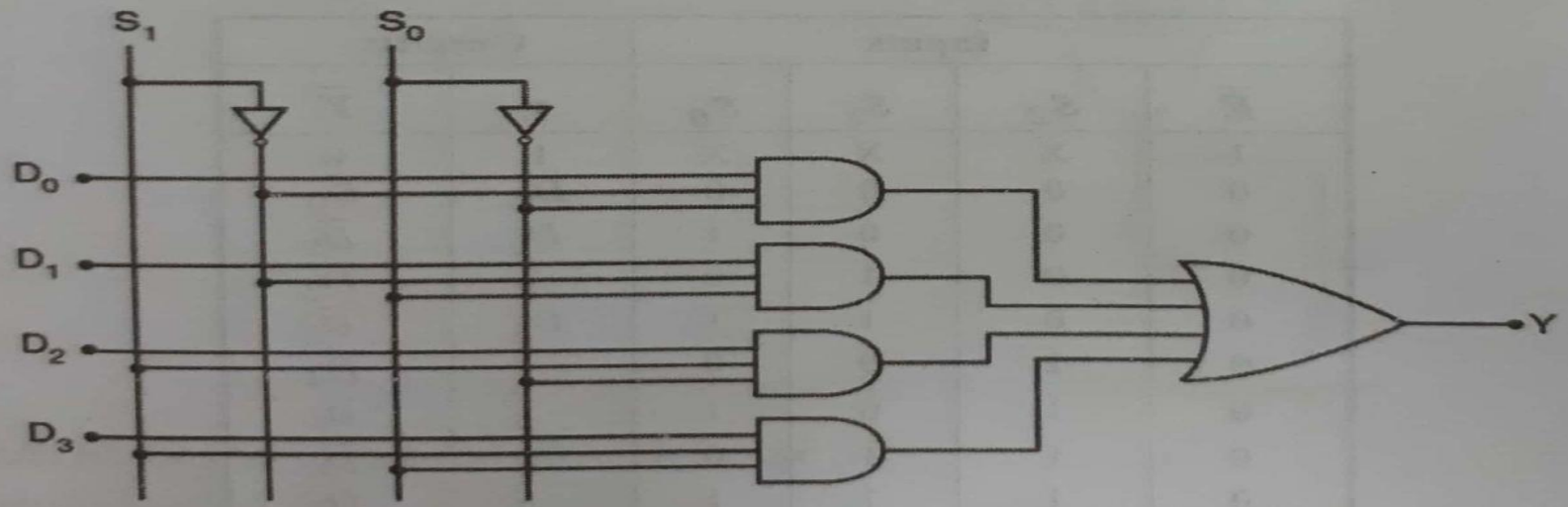
By : Ms. Kanika Dhingra

Table 6.1 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) Logic symbol



(b) Logic diagram

Fig. 6.2 4-to-1 multiplexer

8-to-1 Multiplexer

By : Ms. Kanika Dhingra

Table 6.2 *Truth table of IC 74151 — 8-to-1 multiplexer*

Inputs				Outputs	
$\overline{E}$	S_2	S_1	S_0	Y	$\overline{Y}$
1	X	X	X	1	0
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	4	$\overline{D_4}$
0	1	0	1	5	$\overline{D_5}$
0	1	1	0	6	$\overline{D_6}$
0	1	1	1	7	$\overline{D_7}$

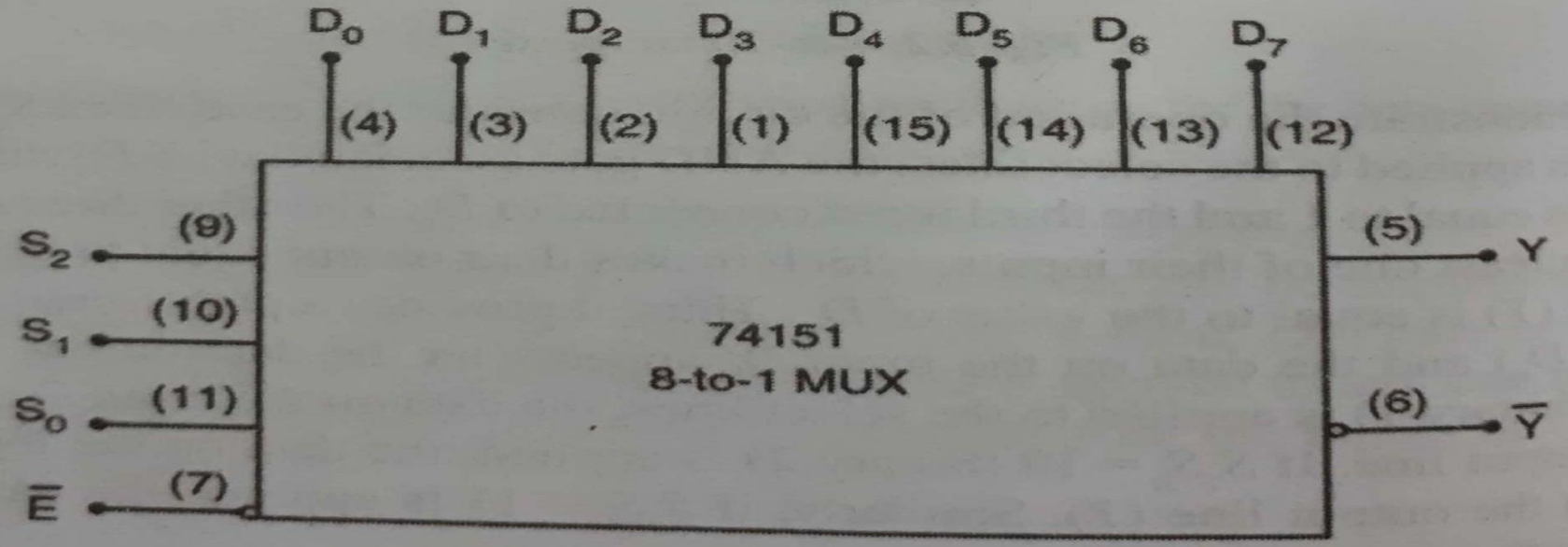


Fig. 6.3 (a) Logic symbol of IC 74151 — 8-to-1 multiplexer

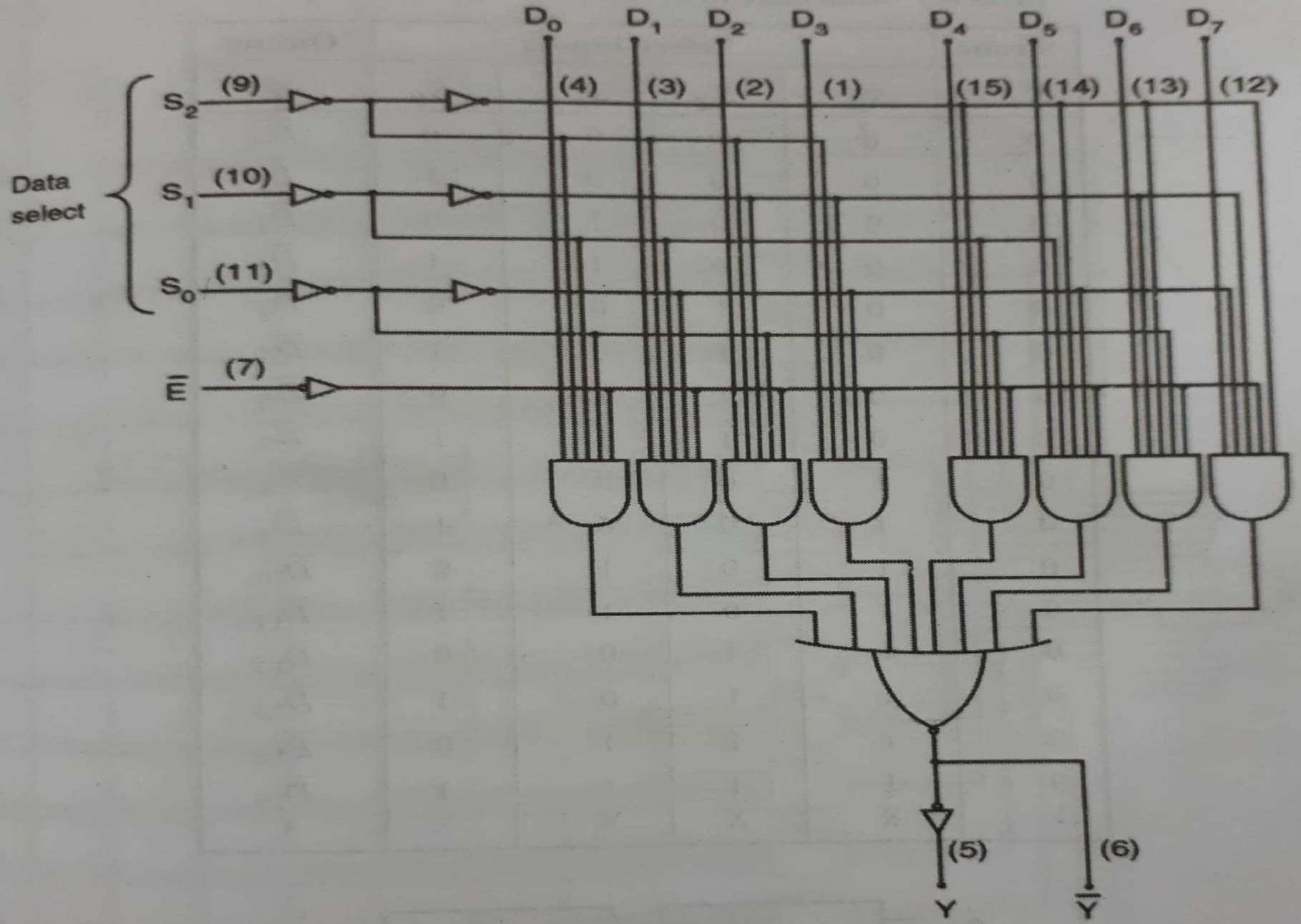


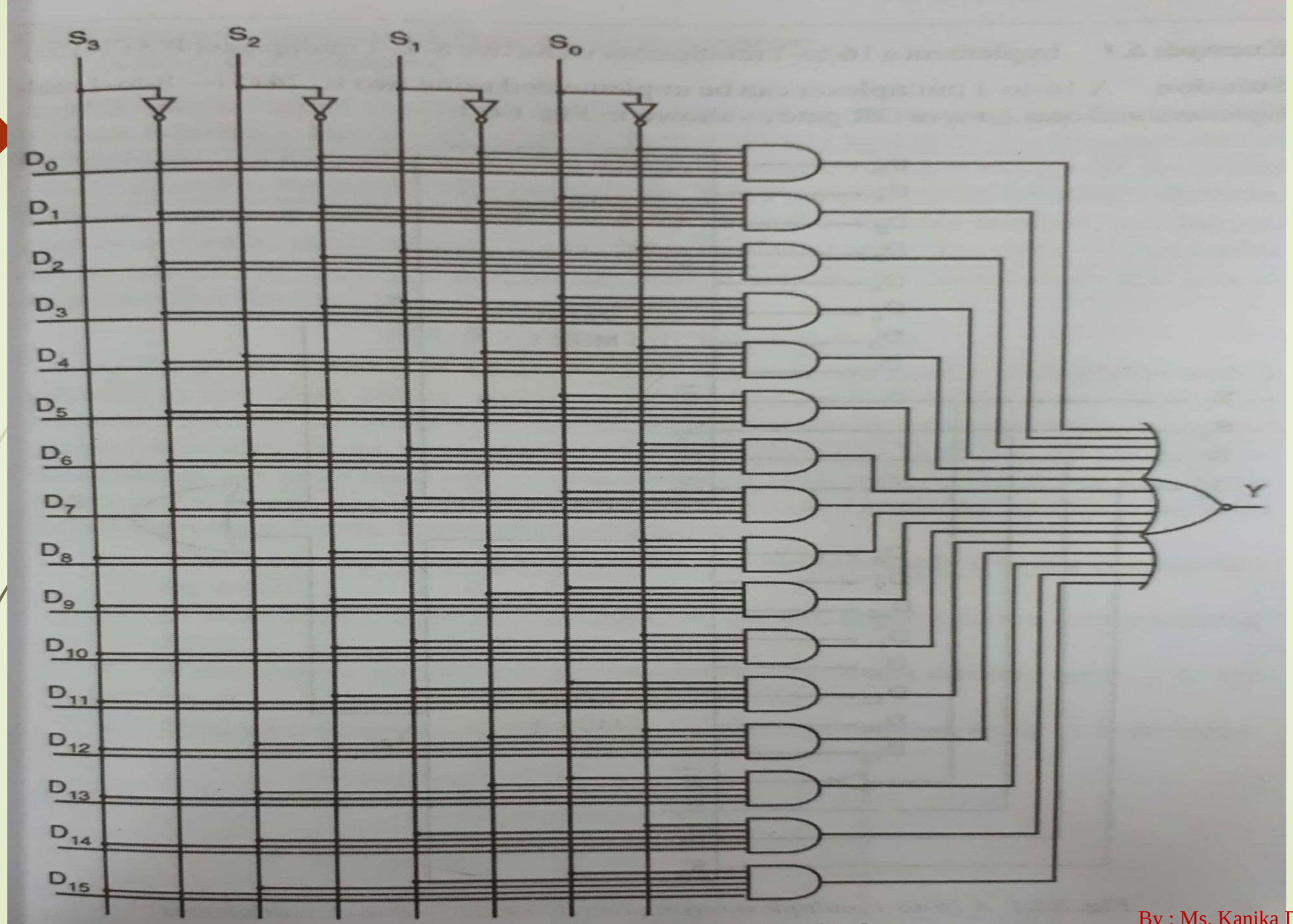
Fig. 6.3 (b) Logic diagram of IC 74151 — 8-to-1 multiplexer

16- to- 1 Multiplexer

By : Ms. Kanika Dhingra

Table 6.3 *Truth table of IC 74150 — 16-to-1 multiplexer*

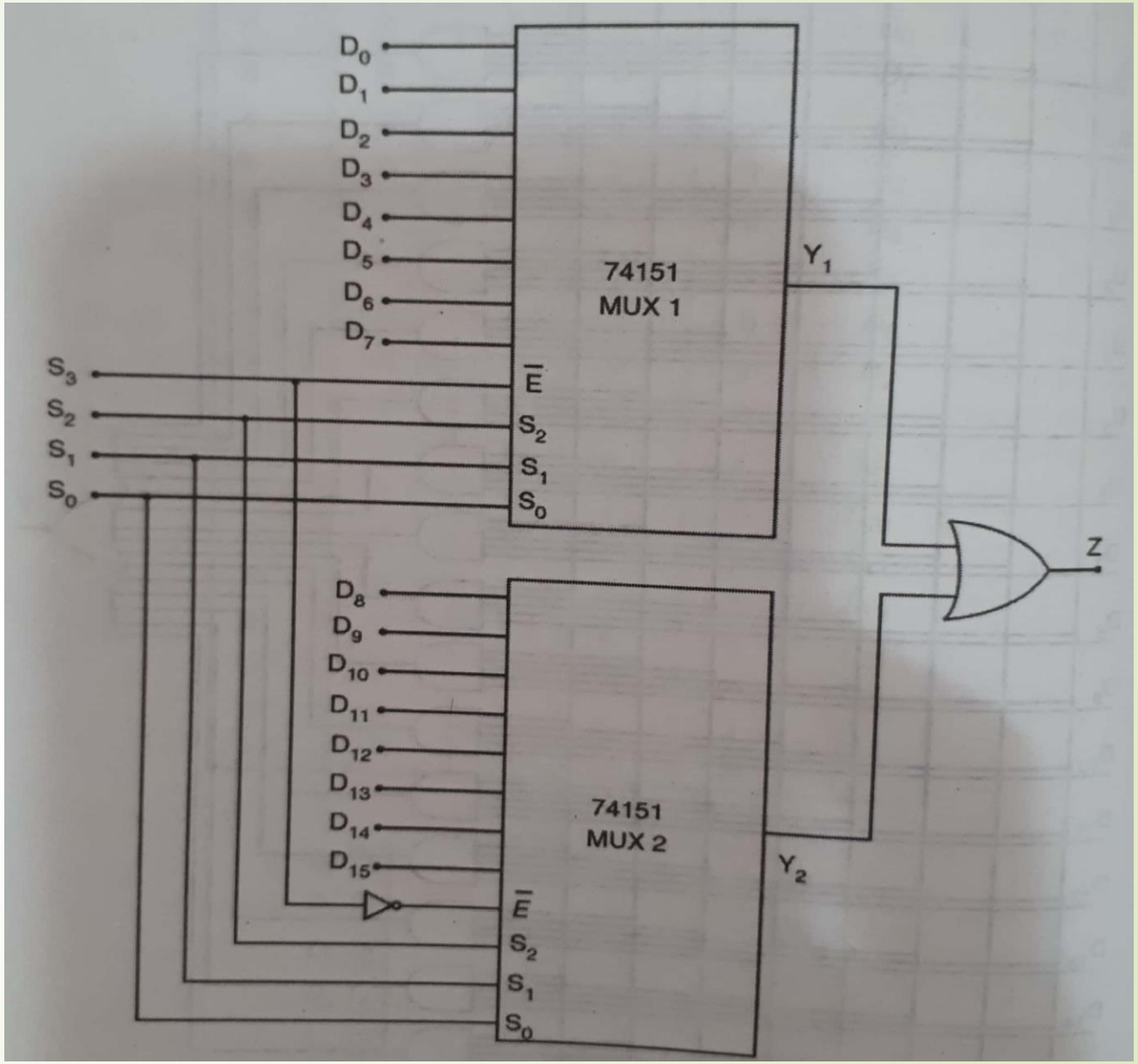
Strobe	Select inputs				Output
$\overline{S}$	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	$\overline{D}_0$
0	0	0	0	1	$\overline{D}_1$
0	0	0	1	0	$\overline{D}_2$
0	0	0	1	1	$\overline{D}_3$
0	0	1	0	0	$\overline{D}_4$
0	0	1	0	1	$\overline{D}_5$
0	0	1	1	0	$\overline{D}_6$
0	0	1	1	1	$\overline{D}_7$
0	1	0	0	0	$\overline{D}_8$
0	1	0	0	1	$\overline{D}_9$
0	1	0	1	0	$\overline{D}_{10}$
0	1	0	1	1	$\overline{D}_{11}$
0	1	1	0	0	$\overline{D}_{12}$
0	1	1	0	1	$\overline{D}_{13}$
0	1	1	1	0	$\overline{D}_{14}$
0	1	1	1	1	$\overline{D}_{15}$
1	X	X	X	X	1



(b) Logic diagram of 16-to-1 multiplexer

By : Ms. Kanika Dhingra

Implement a 16 – to – 1 multiplexer using two 8 – to – 1 multiplexers.



By : Ms. Kanika Dhingra

DEMULTIPLEXERS

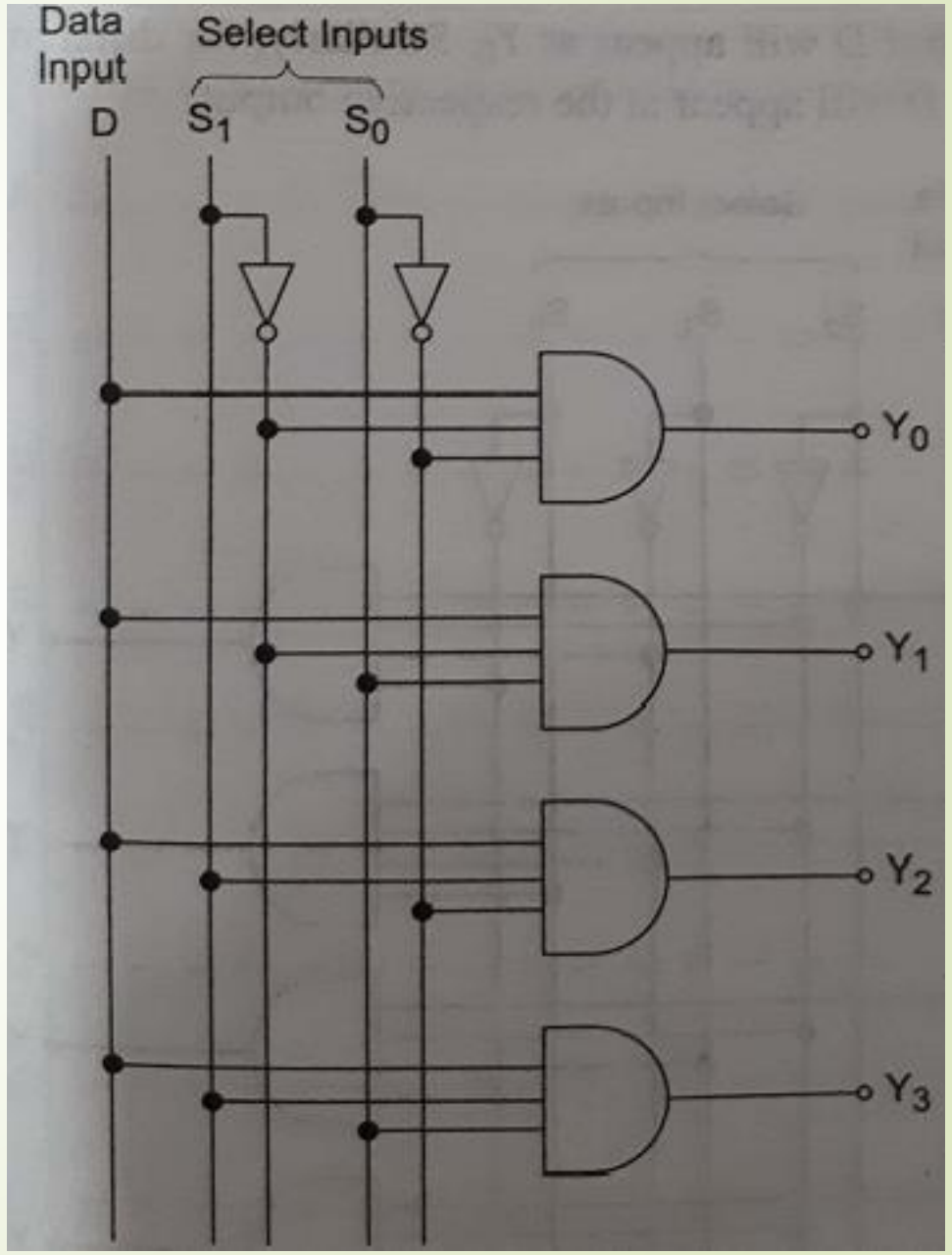
- **Demultiplex means “ one to many ”.**
- **A digital demultiplexer is a logic circuit that receives information on a single input and transmits the same information over one of the several (2^m) output lines.**
- **A multiplexer is also called a Data Distributor.**
- **Demultiplexer has single data input line and several (2^m) output lines. The selection of a particular output line is controlled by a set of selection lines.**

By : Ms. Kanika Dhingra

1 to 4 Demultiplexer

Data input	Select inputs		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

$$Y_0 = \overline{S}_1 \overline{S}_0 D$$
$$Y_1 = \overline{S}_1 S_0 D$$
$$Y_2 = S_1 \overline{S}_0 D$$
$$Y_3 = S_1 S_0 D$$



By : Ms. Kanika Dhingra

1 to 8 Demultiplexer

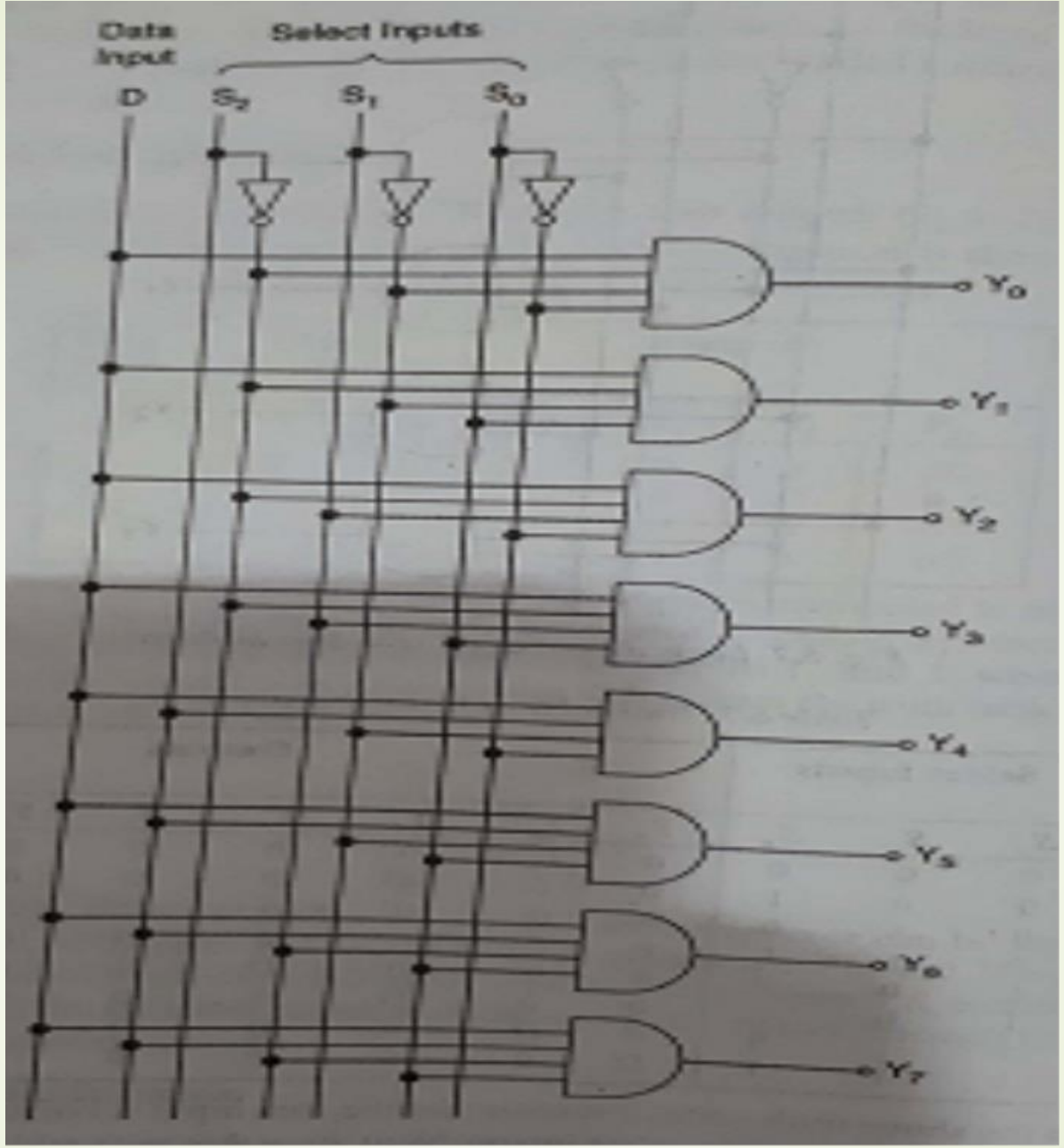
Data input	Select inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

$$Y_0 = \bar{S}_2 \bar{S}_1 \bar{S}_0 D;\ Y_1 = \bar{S}_2 \bar{S}_1 S_0 D;\ Y_2 = \bar{S}_2 S_1 \bar{S}_0 D;\ Y_3 = \bar{S}_2 S_1 S_0 D$$

$$Y_4 = S_2 \bar{S}_1 \bar{S}_0 D;\ Y_5 = S_2 \bar{S}_1 S_0 D;\ Y_6 = S_2 S_1 \bar{S}_0 D;\ Y_7 = S_2 S_1 S_0 D$$

Logic diagram of a 1-to-8 demultiplexer

By : Ms. Kanika Dhingra



By : Ms. Kanika Dhingra

Decoders

- A decoder is similar to demultiplexer but without any data input.
- A decoder is a logic circuit that converts an n-bit binary input code (data) into 2^n output lines, such that each output line will be activated for only one of the possible combinations of inputs.
- In a decoder, the number of output is greater than the number of inputs.

Note: If the number of inputs and outputs are equal in a digital system then it can be called converters.

By : Ms. Kanika Dhingra

3 - to – 8 Decoder

Table 6.10 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

By : Ms. Kanika Dhingra

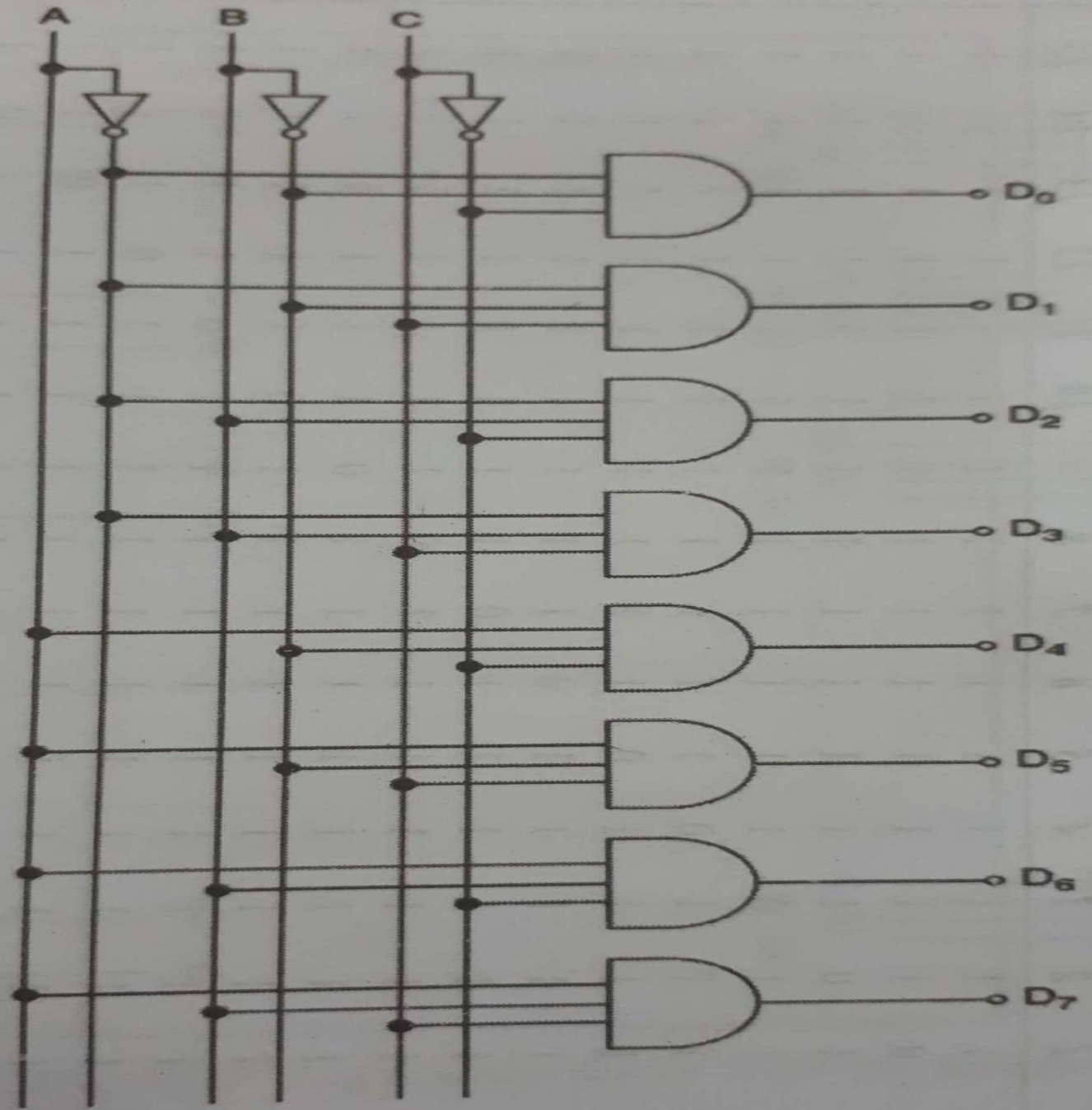


Fig. 6.12 Logic diagram of 3-to-8 decoder

By : Ms. Kanika Dhingra

4-to-16 Decoder

Table 6.11 Truth table of 4-to-16 decoder

Decimal digit	Binary inputs				Logic function	Outputs														
	D	*C*	*B*	*A*		*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	0	0	0	0	$\bar{D}\bar{C}\bar{B}\bar{A}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\bar{D}\bar{C}\bar{B}A$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\bar{D}\bar{C}B\bar{A}$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\bar{D}\bar{C}BA$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\bar{D}C\bar{B}\bar{A}$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\bar{D}C\bar{B}A$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\bar{D}CB\bar{A}$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\bar{D}CBA$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$D\bar{C}\bar{B}\bar{A}$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$D\bar{C}\bar{B}A$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$D\bar{C}B\bar{A}$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$D\bar{C}BA$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$DC\bar{B}\bar{A}$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$DC\bar{B}A$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$DCB\bar{A}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$DCBA$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

By : Ms. Kanika Dhingra

BCD to Decimal Decoder

A decoder that takes a 4-bit BCD as the input code and produces 10 outputs corresponding to the decimal digits .

Inputs				Outputs									
D	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

By : Ms. Kanika Dhingra

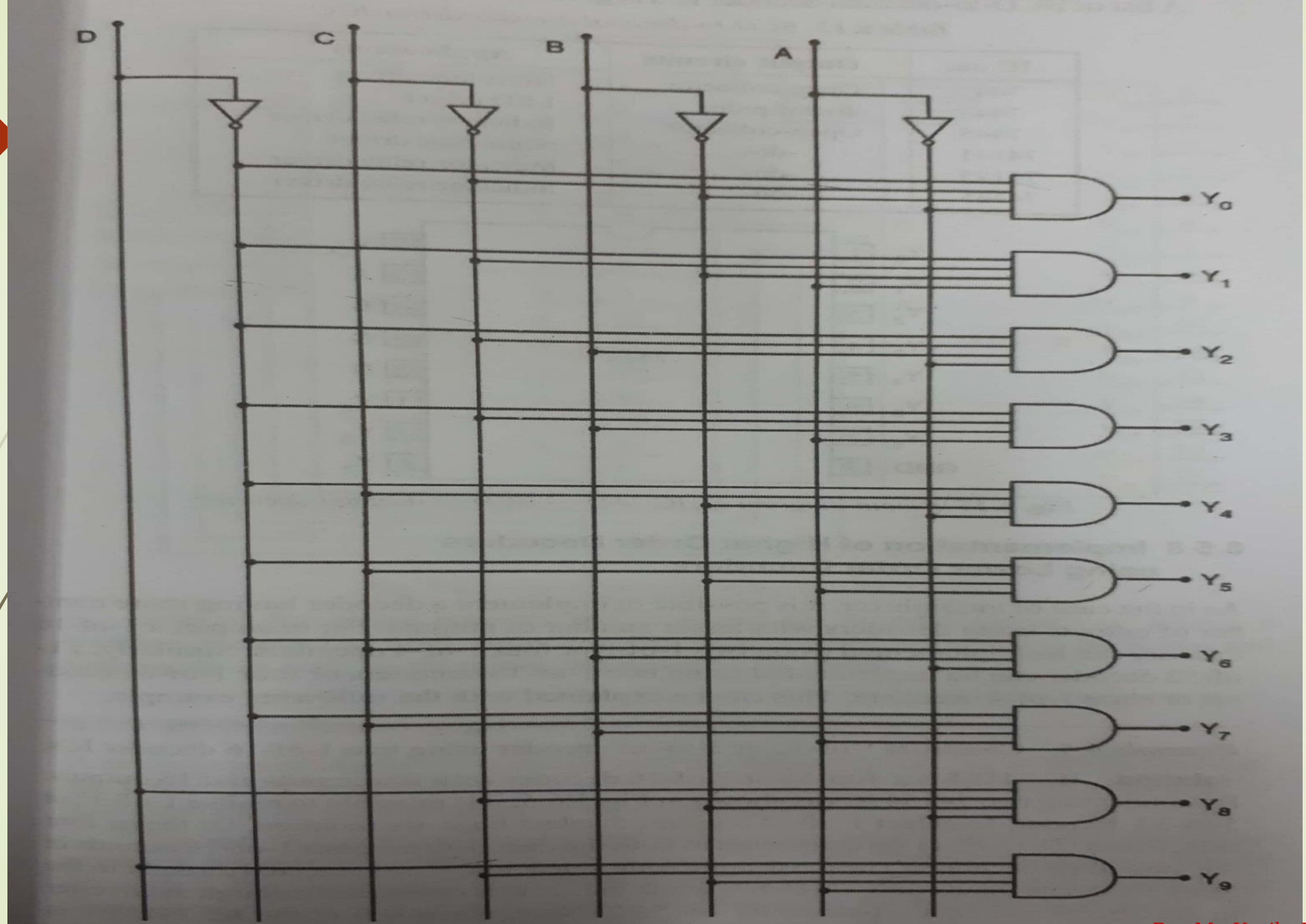


Fig. 6.16 Logic diagram of BCD-to-decimal decoder

BCD- to- Seven Segment Decoder

A seven segment display is 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.

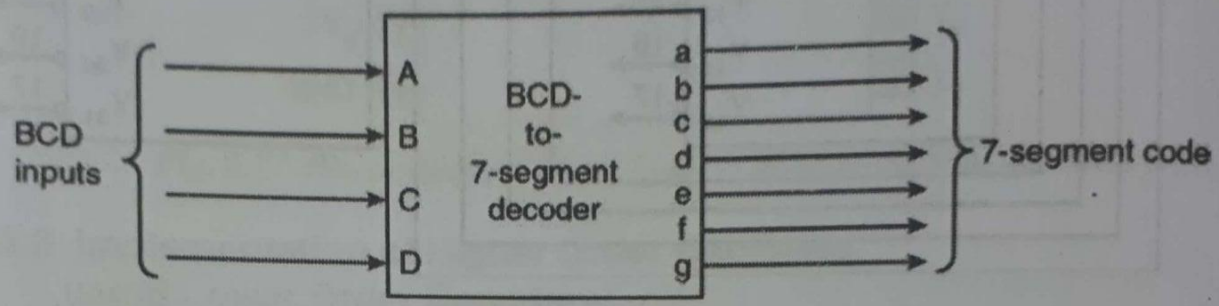


Fig. 6.19 Block diagram of BCD-to-7-segment decoder

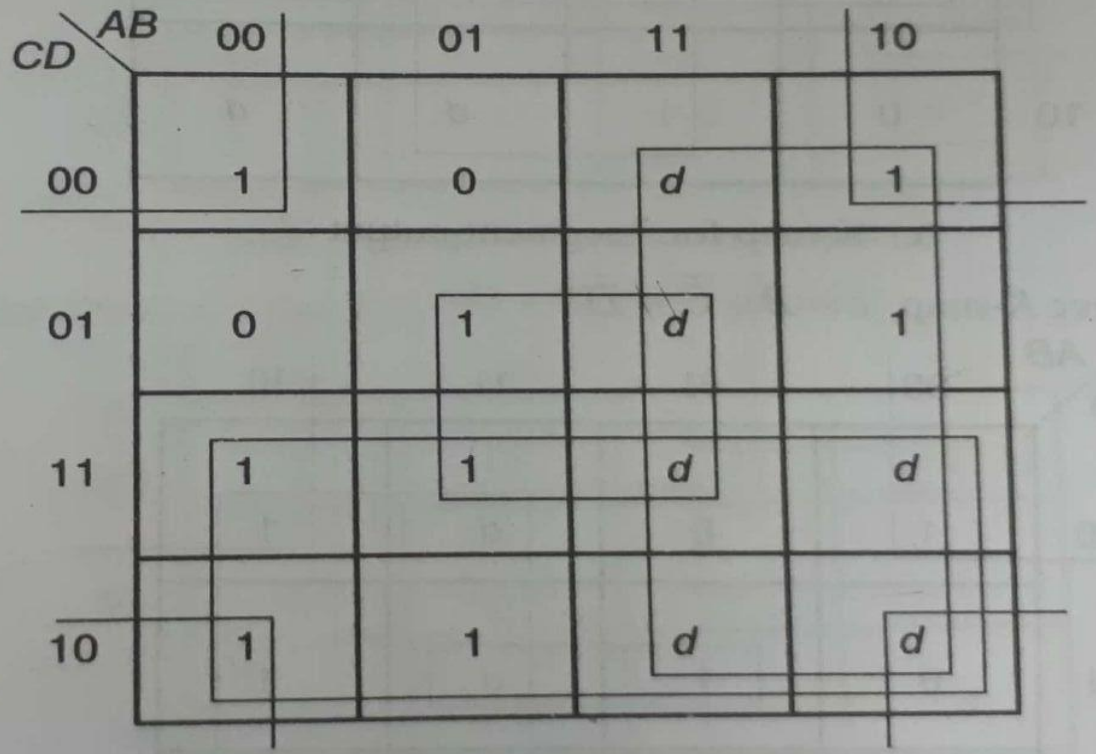
Table 6.13 Truth table of BCD-to-7-segment decoder

BCD inputs				Seven segment outputs						
A	*B*	*C*	*D*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

By : Ms. Kanika Dhingra

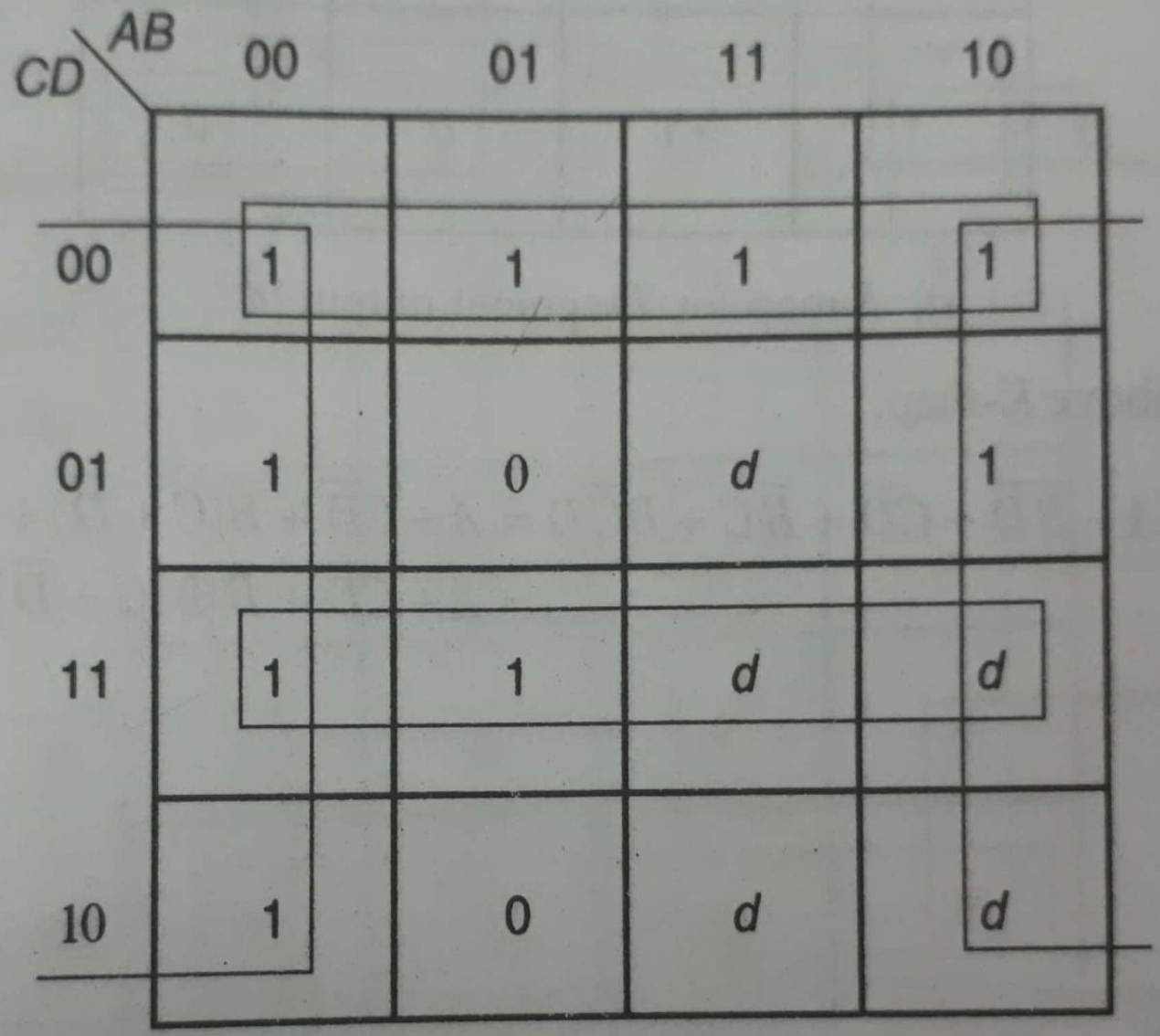
$$a = \Sigma_m(0,2,3,5,6,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$b = \Sigma_m(0,1,2,3,4,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$c = \Sigma_m(0,1,3,4,5,6,7,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$d = \Sigma_m(0,2,3,5,6,8,9) + \Sigma_d(10,11,12,13,14,15)$$
$$e = \Sigma_m(0,2,6,8) + \Sigma_d(10,11,12,13,14,15)$$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

The above expressions can be simplified using *K*-map method as shown in Fig. 6.20.



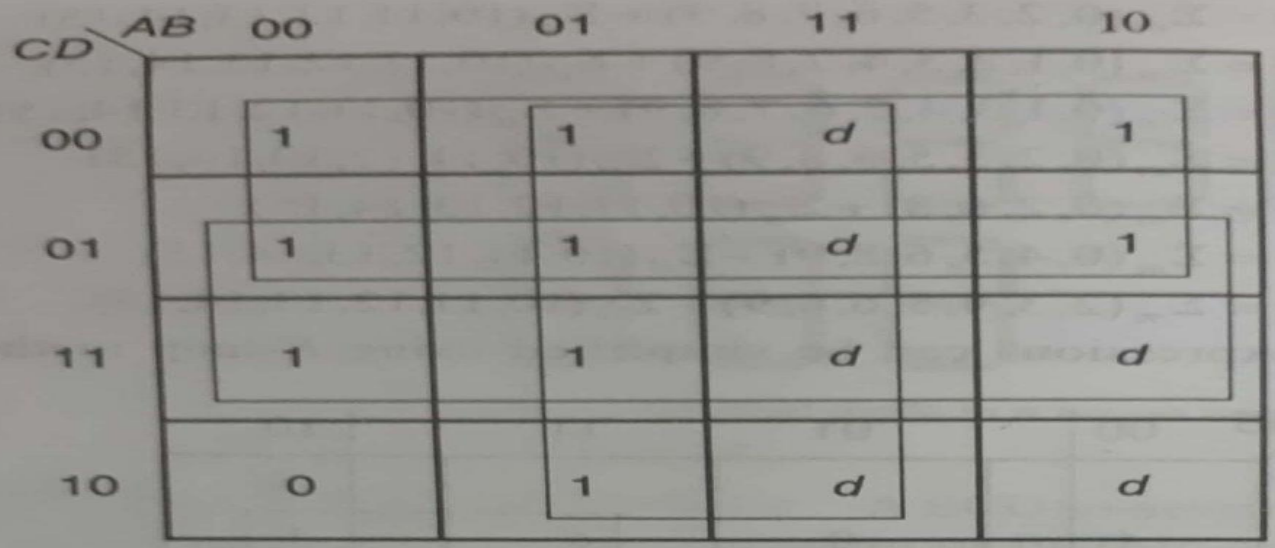
(a) *K*-map for 7-segment output '*a*'

From the above *K*-map, $a = A + C + BD + \overline{B}\,\overline{D} = A + C + \overline{B \oplus D}$



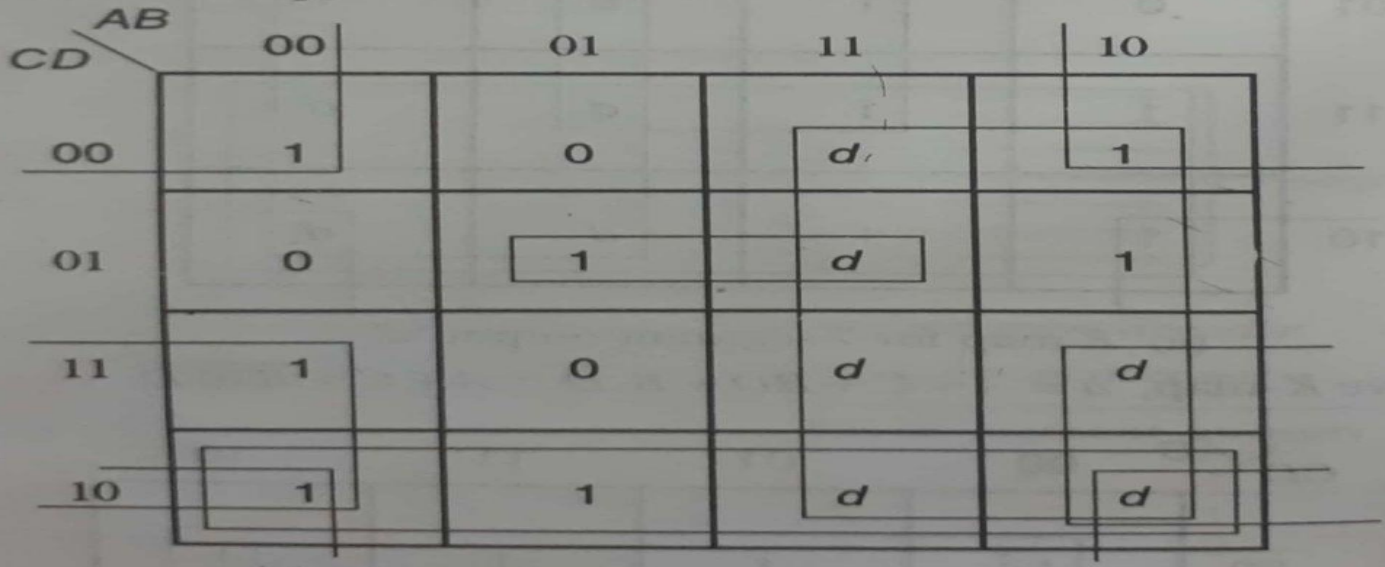
(b) *K*-map for 7-segment output '*b*'

From the above *K*-map, $b = \overline{B} + CD + \overline{C}\,\overline{D} = \overline{B} + \overline{C \oplus D}$



(c) K-map for 7-segment output 'c'

From the above K-map, $c = B + \overline{C} + D$

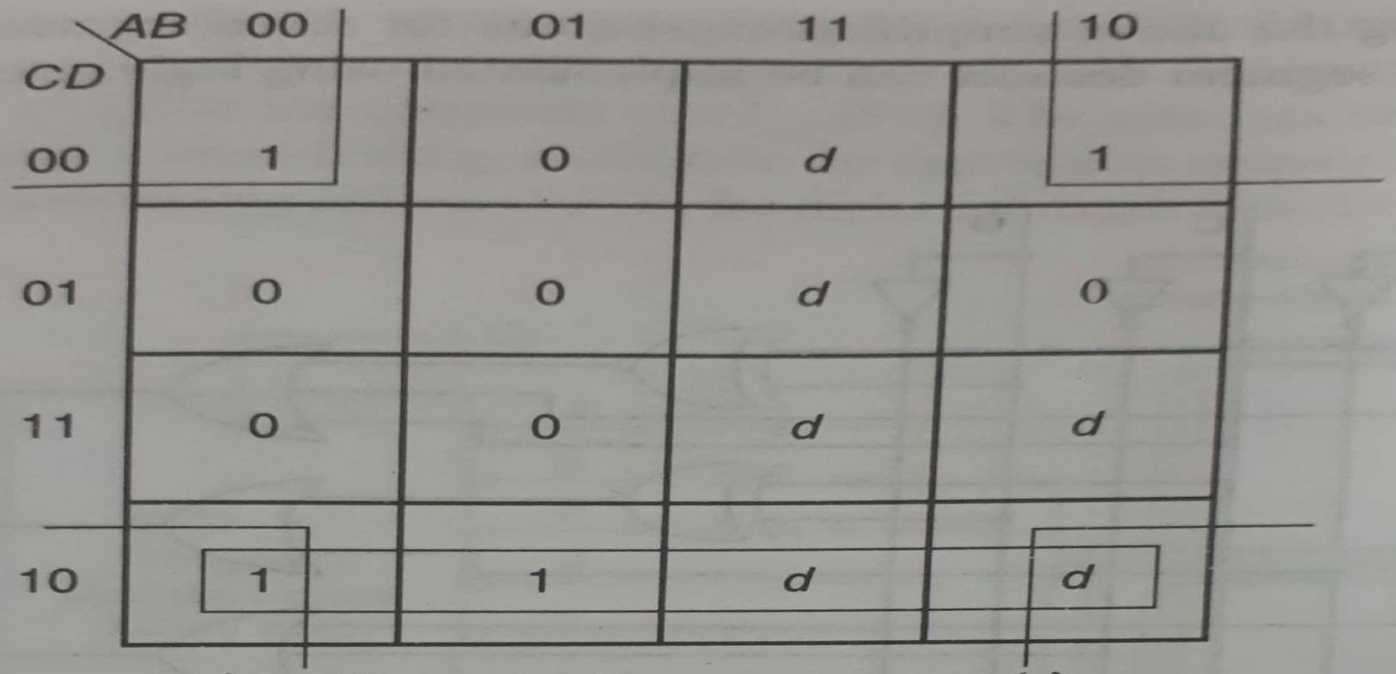


(d) K-map for 7-segment output 'd'

From the above K-map,

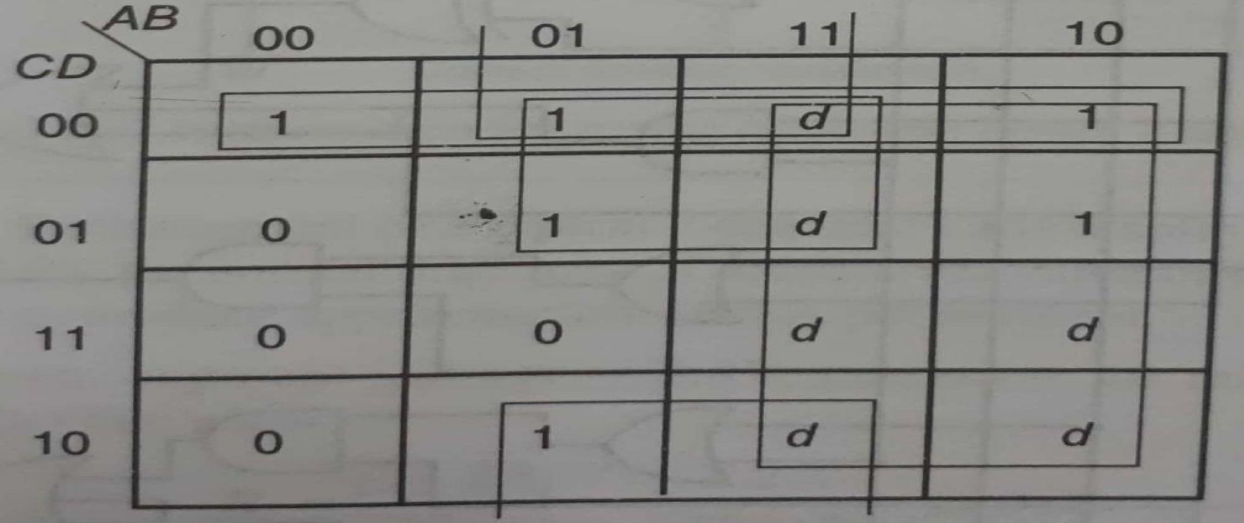
$$d = A + \overline{B}\,\overline{D} + C\overline{D} + \overline{B}C + B\overline{C}D = A + C\overline{D} + \overline{B}(C + \overline{D}) + B\overline{C}D$$
$$= A + C\overline{D} + B \oplus (C + \overline{D})$$

By : Ms. Kanika Dhingra



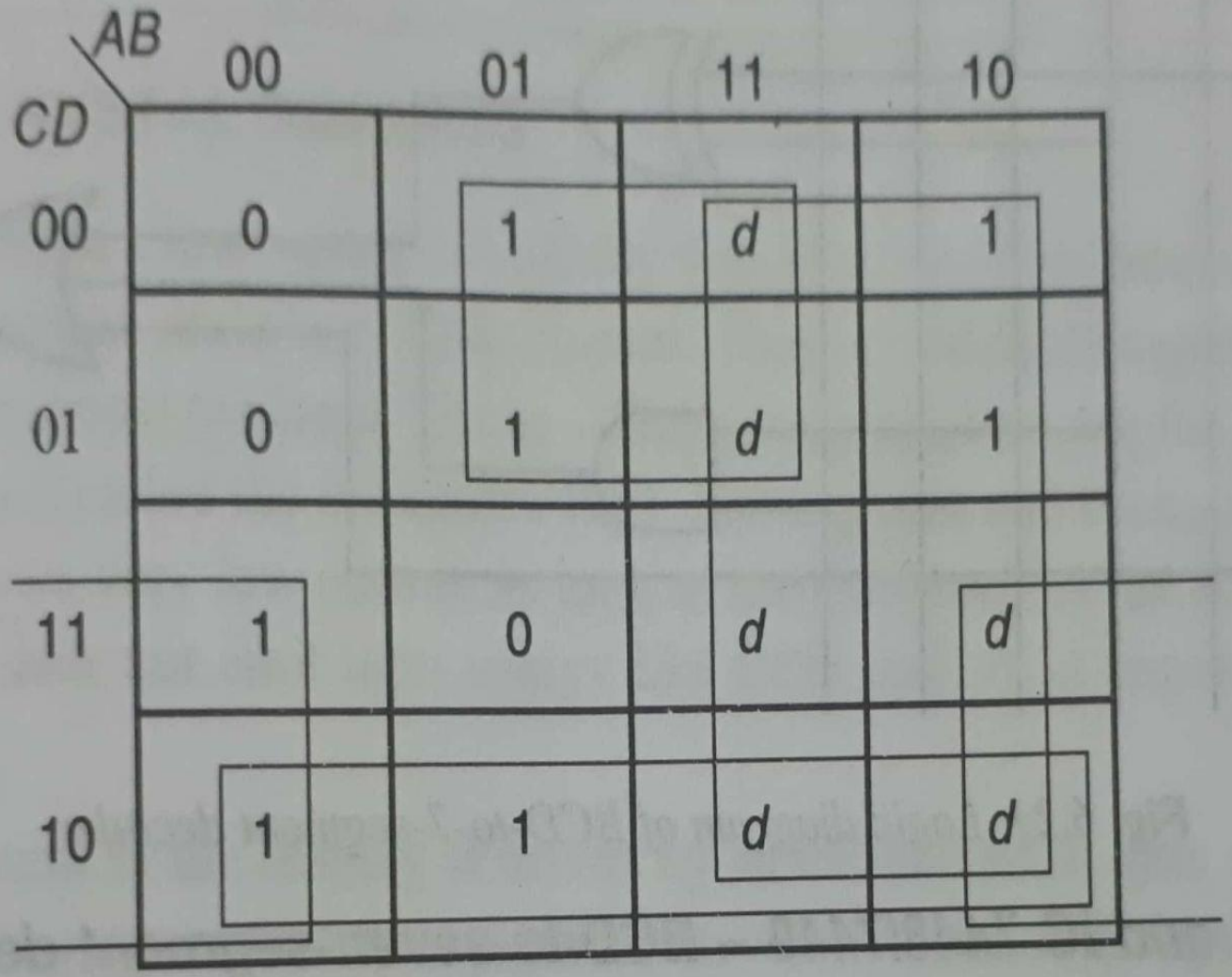
(e) *K*-map for 7-segment output '*e*'

From the above *K*-map, $e = \overline{B}\,\overline{D} + C\overline{D} = \overline{D}(\overline{B} + C)$



(f) *K*-map for 7-segment output '*f*'

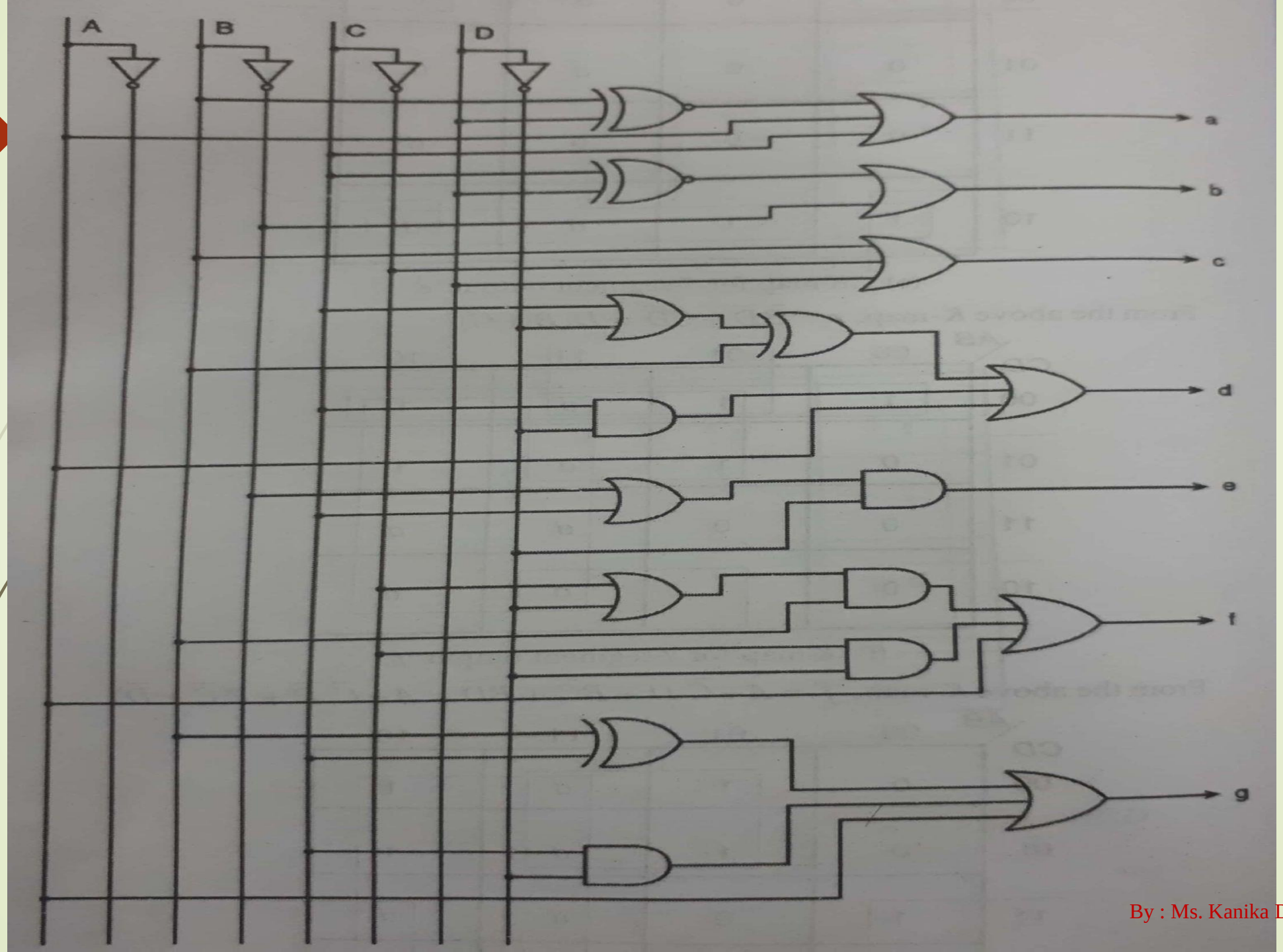
From the above *K*-map, $f = A + \overline{C}\,\overline{D} + B\overline{C} + B\overline{D} = A + \overline{C}\,\overline{D} + B(\overline{C} + \overline{D})$



(g) *K*-map for 7-segment output '*g*'

From the above *K*-map, $g = A + C\overline{D} + B\overline{C} + \overline{B}C = A + C\overline{D} + (B \oplus C)$

Fig. 6.20 *K-map simplification for BCD-to-7-segment decoder*



By : Ms. Kanika Dhingra

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 n input lines ,only one of which is active at any time and m output lines.
- It encodes one of the active inputs to a coded binary output with m bits.
- In an encoder, the number of outputs is less than the number of inputs.

By : Ms. Kanika Dhingra

Table 6.14 Truth table of octal-to-binary encoder

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

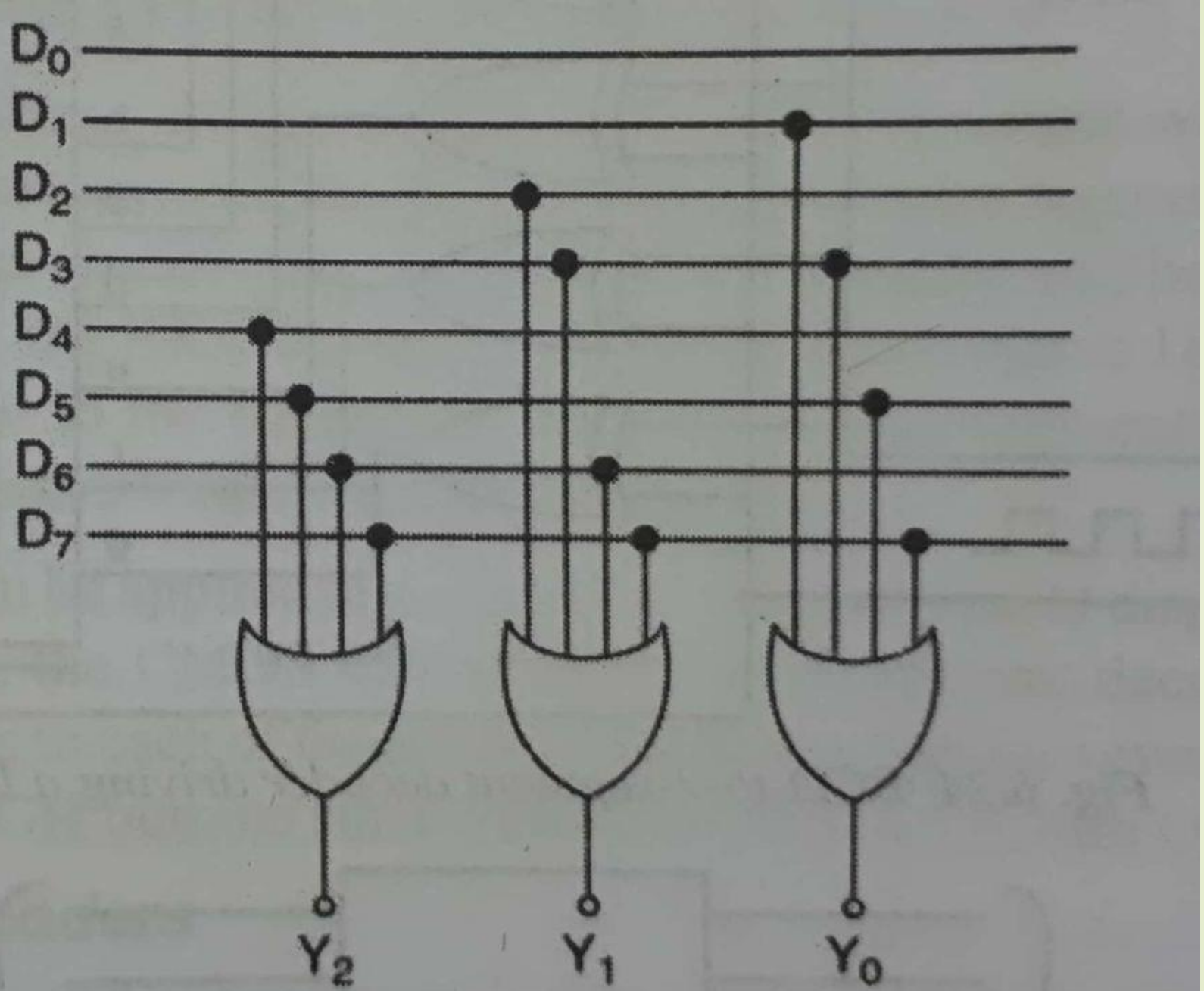


Fig. 6.26 Octal-to-binary encoder

Decimal- to- BCD encoder

Table 6.15 Truth table of decimal-to-BCD encoder

Decimal inputs										BCD outputs			
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

By : Ms. Kanika Dhingra

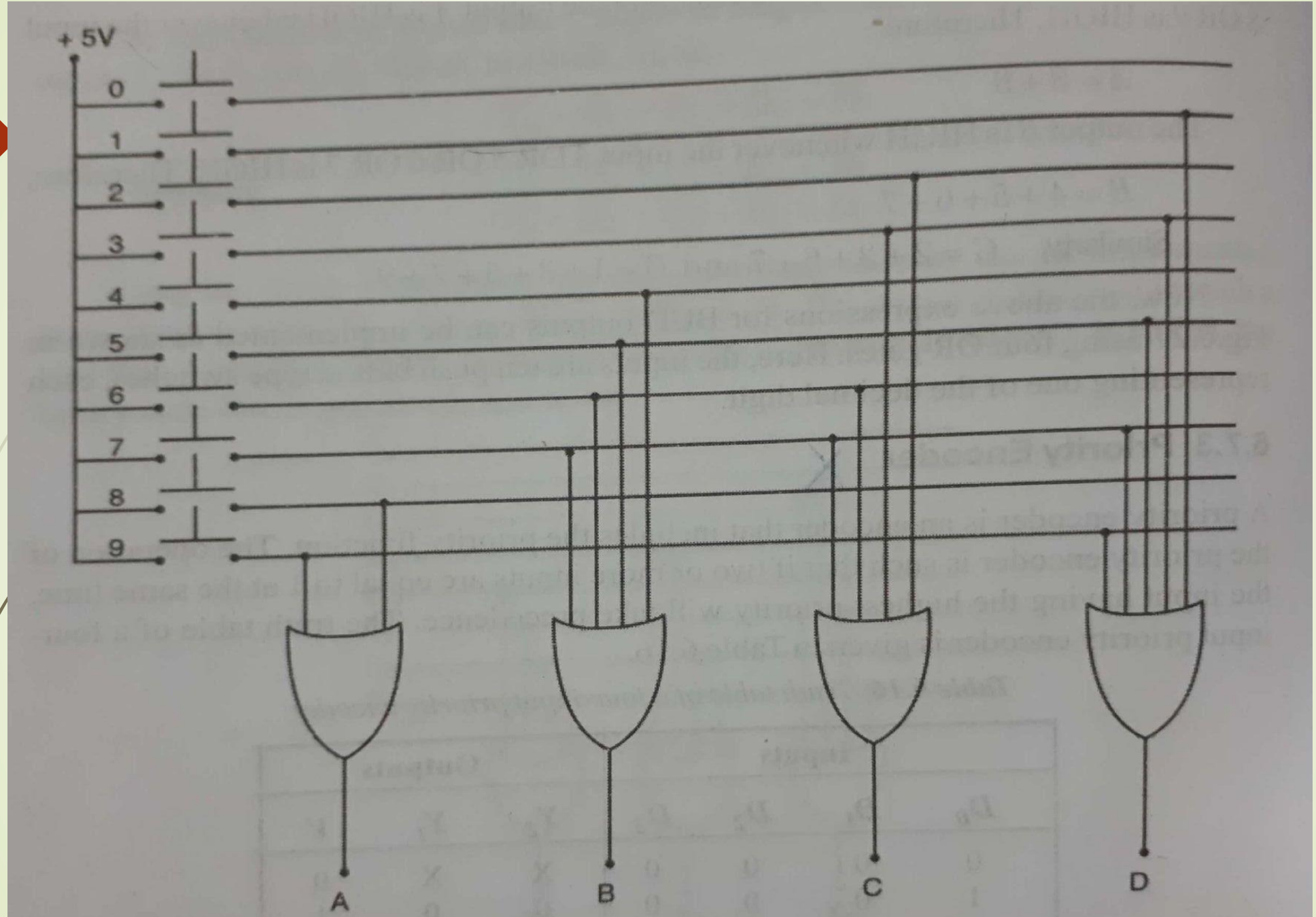


Fig. 6.27 Decimal-to-BCD encoder

By : Ms. Kanika Dhingra

Binary to Gray code Converter

Table 6.23 *Truth table of binary-to-gray code converters*

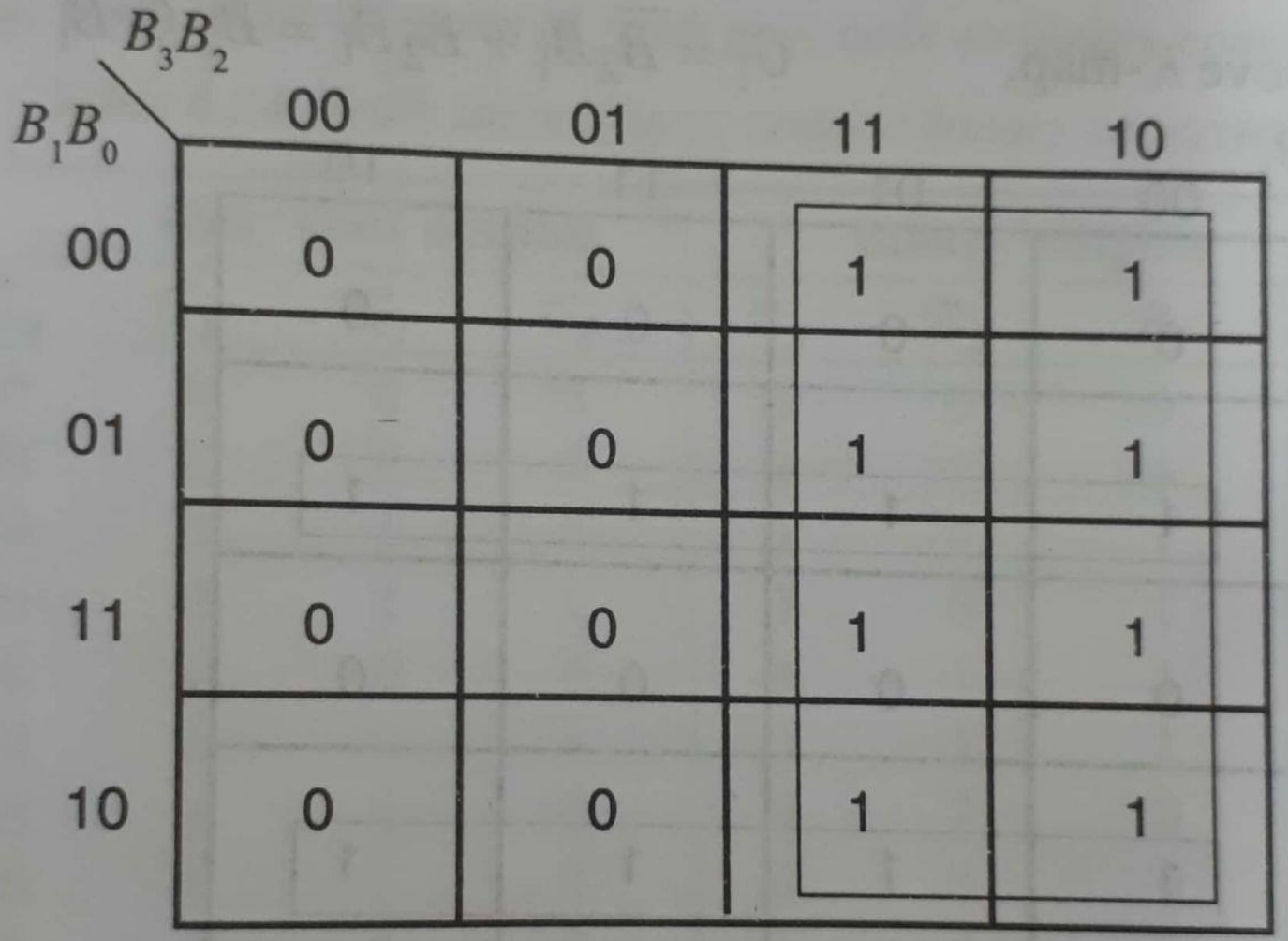
Binary inputs				Gray code outputs			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

From the truth table shown in Table 6.23, the logic expressions for the gray code outputs can be written as:

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

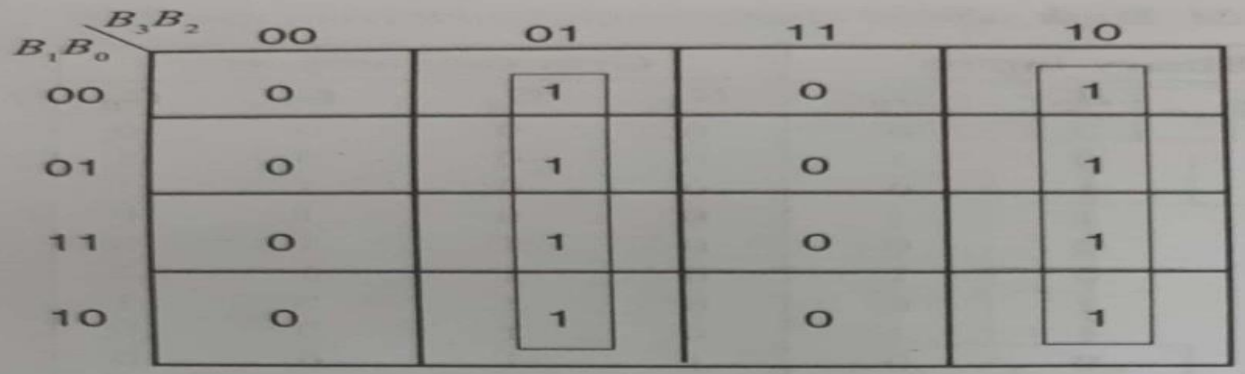
By : Ms. Kanika Dhingra

Fig. 6.40



(a) *K*-map for G_3

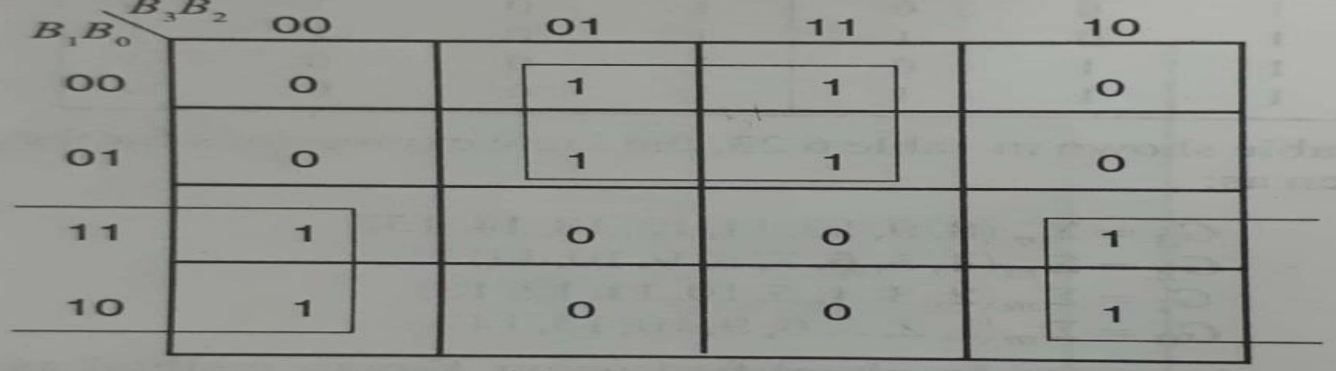
From the above *K*-map, $G_3 = B_3$



B_1B_0 \ B_3B_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

(b) *K*-map for G_2

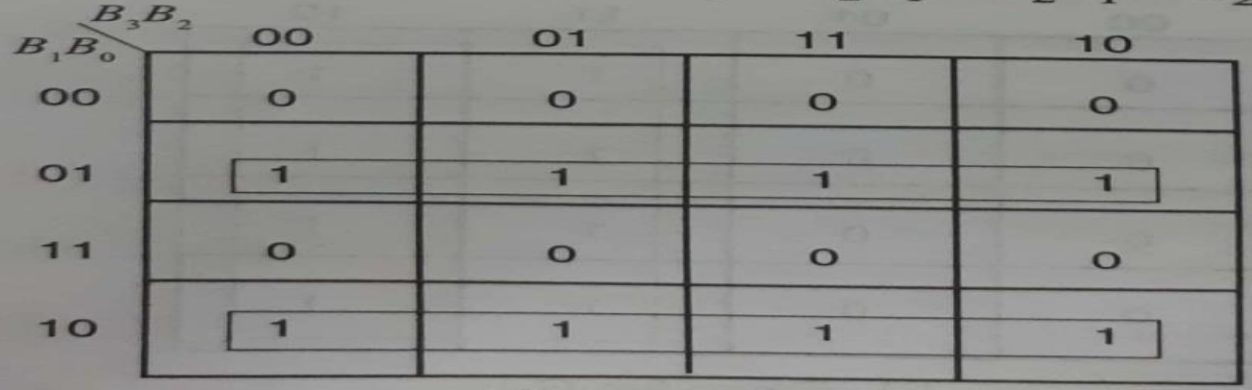
From the above *K*-map, $G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2 = B_3 \oplus B_2$



B_1B_0 \ B_3B_2	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

(c) *K*-map for G_1

From the above *K*-map, $G_1 = \overline{B}_2 B_1 + B_2 \overline{B}_1 = B_2 \oplus B_1$



B_1B_0 \ B_3B_2	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

(d) *K*-map for G_0

From the above *K*-map, $G_0 = \overline{B}_1 B_0 + B_1 \overline{B}_0 = B_1 \oplus B_0$

Fig. 6.40 *K-map simplification for binary-to-gray code converter*

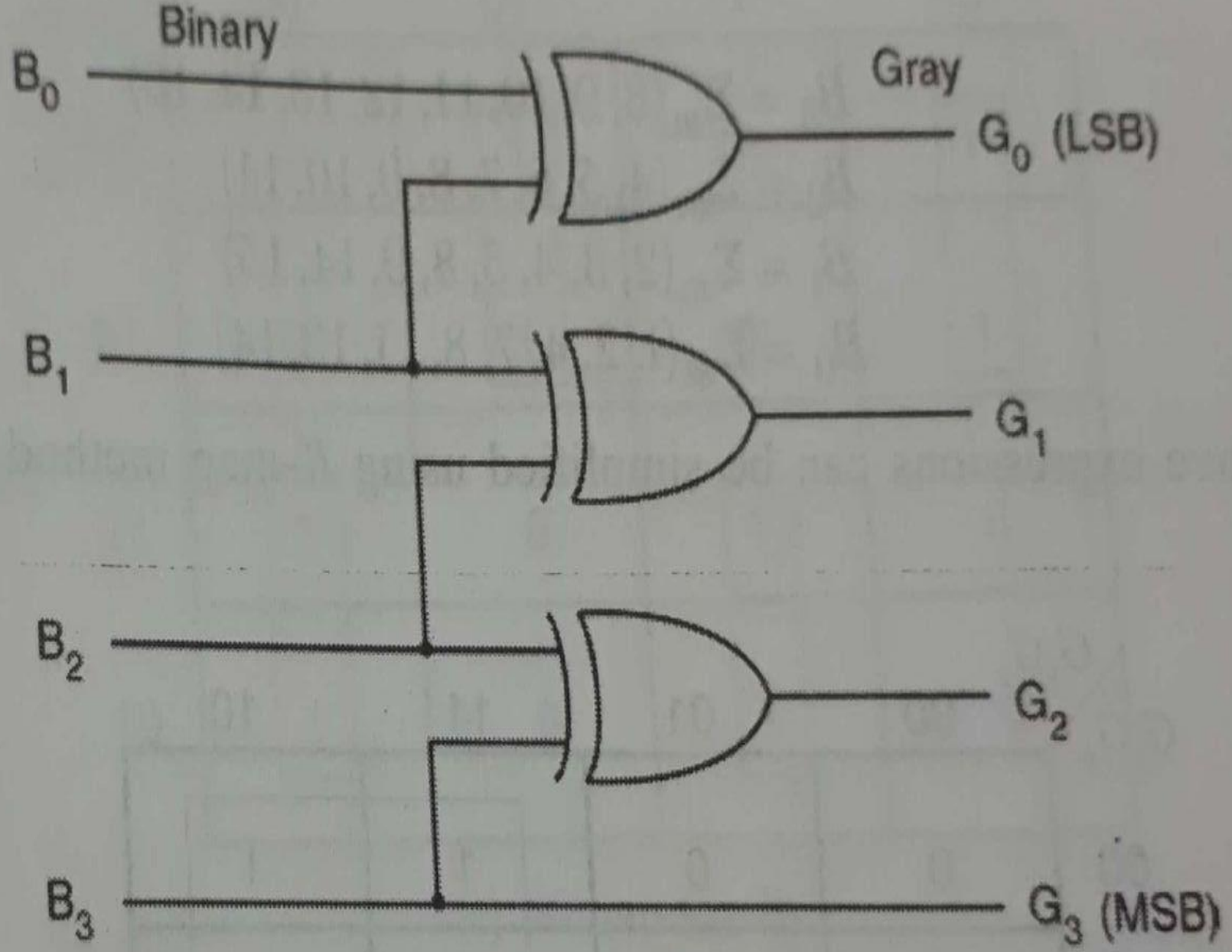


Fig. 6.41 Logic diagram of 4-bit binary-to-gray code converter

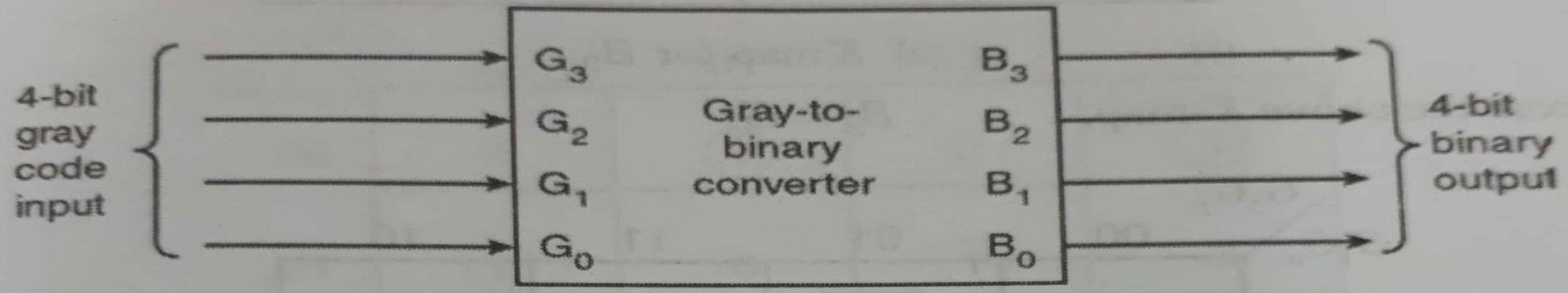


Fig. 6.42 Block diagram of 4-bit gray code-to-binary converter

Table 6.24 *Truth table of gray code-to-binary converter*

Gray code outputs				Binary outputs			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

From the above truth table, the logic expressions for the binary outputs can be written as:

$$B_3 = \Sigma_m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$B_2 = \Sigma_m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$B_1 = \Sigma_m(2, 3, 4, 5, 8, 9, 14, 15)$$
$$B_0 = \Sigma_m(1, 2, 4, 7, 8, 11, 13, 14)$$

The above expressions can be simplified using *K*-map method as shown in Fig. 6.43

G_1G_0 \ G_3G_2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

(a) *K*-map for B_3

From the above *K*-map, $B_3 = G_3$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

(b) *K*-map for B_2

From the above *K*-map, $B_2 = \overline{G}_3G_2 + G_3\overline{G}_2 = G_3 \oplus G_2 = B_3 \oplus G_2$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

(c) K-map for B_1

From the above K-map,

$$B_1 = \overline{G_3}\,\overline{G_2}\,G_1 + \overline{G_3}\,G_2\,\overline{G_1} + G_3\,G_2\,G_1 + G_3\,\overline{G_2}\,\overline{G_1}$$
$$= \overline{G_3}\,(\overline{G_2}G_1 + G_2\,\overline{G_1}) + G_3\,(G_2\,G_1 + \overline{G_2}\,\overline{G_1})$$
$$= \overline{G_3}\,(G_2 \oplus G_1) + G_3(\overline{G_2 \oplus G_1})$$
$$= G_3 \oplus G_2 \oplus G_1$$
$$B_1 = B_2 \oplus G_1$$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0
11	0	(1)	0	(1)
10	(1)	0	(1)	0

(d) K-map for B_0

From the above K-map,

$$B_0 = \overline{G_3}\,\overline{G_2}\,\overline{G_1}\,G_0 + \overline{G_3}\,\overline{G_2}\,G_1\,\overline{G_0} + G_3\,G_2\,\overline{G_1}\,G_0 + G_3\,G_2\,G_1\,\overline{G_0}$$
$$+ \overline{G_3}\,G_2\,\overline{G_1}\,\overline{G_0} + G_3\,\overline{G_2}\,\overline{G_1}\,\overline{G_0} + \overline{G_3}\,G_2\,G_1\,G_0 + G_3\,\overline{G_2}\,G_1\,G_0$$
$$= \overline{G_3}\,\overline{G_2}\,(\overline{G_1}\,G_0 + G_1\,\overline{G_0}) + G_3\,G_2(\overline{G_1}\,G_0 + G_1\,\overline{G_0}) + \overline{G_1}\,\overline{G_0}\,(\overline{G_3}\,G_2 + G_3\,\overline{G_2})$$
$$+ G_1\,G_0\,(\overline{G_3}\,G_2 + G_3\,\overline{G_2})$$
$$= \overline{G_3}\,\overline{G_2}\,(\overline{G_0 \oplus G_1}) + G_3\,G_2(G_0 \oplus G_1) + \overline{G_1}\,\overline{G_0}\,(G_2 \oplus G_3) + G_1\,G_0\,(G_2 \oplus G_3)$$
$$= (G_0 \oplus G_1)(\overline{G_3}\,\overline{G_2} + G_3\,G_2) + (G_2 \oplus G_3)(\overline{G_1}\,\overline{G_0} + G_1\,G_0)$$
$$= G_0 \oplus G_1 \oplus G_2 \oplus G_3$$
$$B_0 = G_0 \oplus B_1$$

binary converter

Now, the above expressions can be implemented using EX-OR gates as shown in Fig. 6.44.

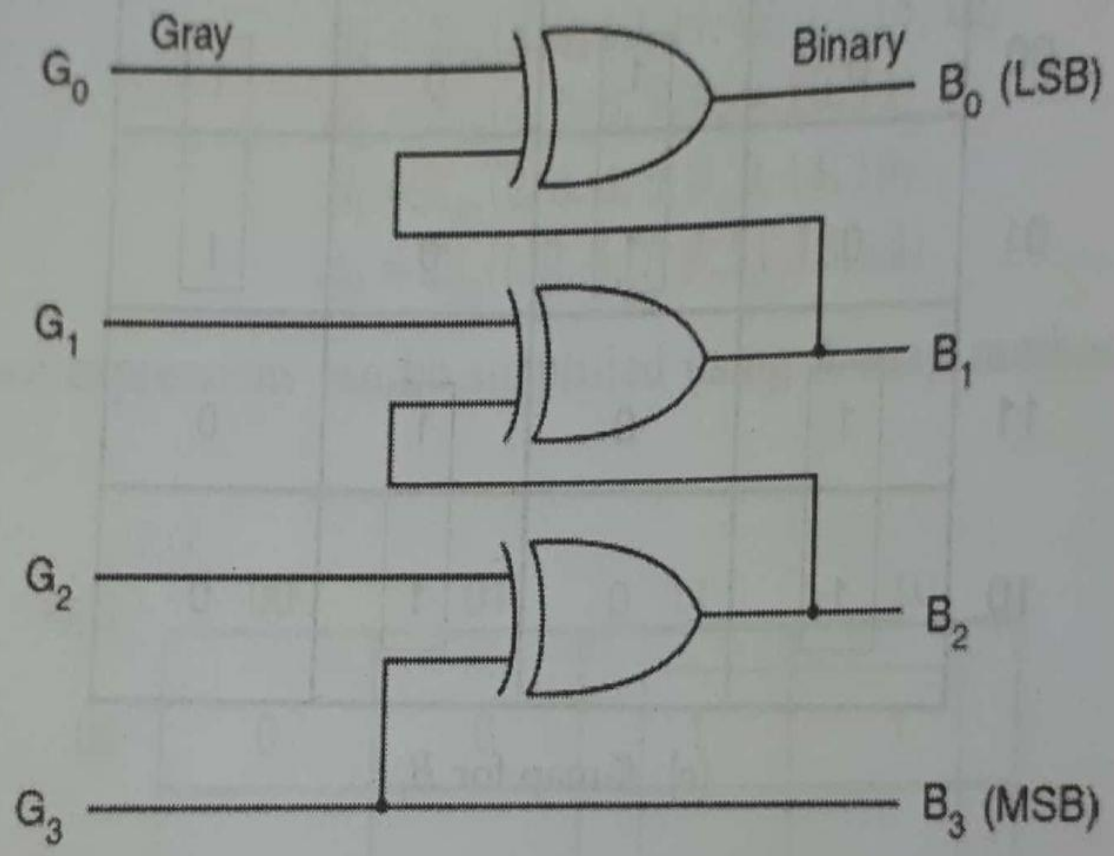


Fig. 6.44 *Logic diagram of 4-bit gray code-to-binary converter*