

# **DEPARTMENT OF MECHATRONICS**

**DIGITAL ELECTRONICS for 3  
year**

# DIGITAL ELECTRONICS

## UNIT ONE

### NUMBER SYSTEM AND BASIC LOGIC

#### INTRODUCTION TO NUMBER SYSTEM

The study of number systems is important from the viewpoint of understanding how data are represented before they can be processed by any digital system including a digital computer.

It is one of the most basic topics in digital electronics.

In this chapter we will discuss different number systems commonly used to represent data.

We will begin the discussion with the decimal number system. This will then be followed by the more commonly used number systems such as the binary, octal and hexadecimal number systems.

# Decimal Number System

- The decimal number system is a radix-10 number system and therefore has 10 different digits or symbols.
- These are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. All higher numbers after '9' are represented in terms of these 10 digits only.

# Binary Number System

- The binary number system is a radix-2 number system with '0' and '1' as the two independent digits.
- All larger binary numbers are represented in terms of '0' and '1'. The procedure for writing higher-order binary numbers after '1' is similar to the one we know in the case of the decimal number system.
- For example, the first 16 numbers in the binary number system would be 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111.
- The next number after 1111 is 10000, which is the lowest binary number with five digits. This also proves the point made earlier that a maximum of only 16 ( $= 2^4$ ) numbers could be written with four digits.

# Octal Number System

- The octal number system has a radix of 8 and therefore has eight distinct digits.
- All higher-order numbers are expressed as a combination of these on the same pattern as the one followed in the case of the binary and decimal number systems.
- The independent digits in octal number system are 0, 1, 2, 3, 4, 5, 6 and 7.
- The next 10 numbers that follow '7', for example, would be 10, 11, 12, 13, 14, 15, 16, 17, 20 and 21 which means 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17 in decimal number system respectively.

# Hexadecimal Number System

- The hexadecimal number system is a radix-16 number system and its 16 basic digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. The decimal equivalent of A, B, C, D, E and F are 10, 11, 12, 13, 14 and 15 respectively, for obvious reason that the hexadecimal number system provides a condensed way of representing large binary numbers stored and processed inside the computer.

# Decimal-to-Binary Conversion

- For the integer part, the binary equivalent can be found by successively dividing the integer part of the number by 2 and recording the remainders until the quotient becomes '0'.
- The remainders written in reverse order constitute the binary equivalent.
- For the fractional part, it is found by successively multiplying the fractional part of the decimal number by 2 and recording the carry until the result of multiplication is '0'. The carry sequence written in forward order constitutes the binary equivalent of the fractional part of the decimal number

Example

*We will find the binary equivalent of  $(13.375)_{10}$ .*

***Solution***

- The integer part = 13

Divisor	Dividend	Remainder
2	13	—
2	6	1
2	3	0
2	1	1
—	0	1

- The binary equivalent of  $(13)_{10}$  is therefore  $(1101)_2$
- The fractional part = .375
- $0.375 \times 2 = 0.75$  with a carry of 0
- $0.75 \times 2 = 0.5$  with a carry of 1
- $0.5 \times 2 = 0$  with a carry of 1
- The binary equivalent of  $(0.375)_{10} = (.011)_2$
- Therefore, the binary equivalent of  $(13.375)_{10} = (1101.011)_2$



# Binary-to-Decimal Conversion

❖ The decimal equivalent of a binary number is given by the sum of all the digits multiplied by 2 the power of their respective place values.

The decimal equivalent of the binary number  $(1001.0101)_2$  is determined as follows:

- The integer part = 1001
- The decimal equivalent =  $1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 1 + 0 + 0 + 8 = 9$
- The fractional part = .0101
- Therefore, the decimal equivalent =  $0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0 + 0.25 + 0 + 0.0625 = 0.3125$
- Therefore, the decimal equivalent of  $(1001.0101)_2 = 9.3125$

# Decimal-to-Octal Conversion

- The process of decimal-to-octal conversion is similar to that of decimal-to-binary conversion.
- The progressive division in the case of the integer part and the progressive multiplication while working on the fractional part here are by '8' which is the radix of the octal number system.
- Again, the integer and fractional parts of the decimal number are treated separately.

# Conti.....

- Example

*We will find the octal equivalent of  $(73.75)_{10}$ .*

***Solution***

- The integer part = 73

Divisor	Dividend	Remainder
8	73	—
8	9	1
8	1	1
—	0	1

- The octal equivalent of  $(73)_{10} = (111)_8$
- The fractional part = 0.75
- $0.75 \times 8 = 6$  with a carry of 0
- The octal equivalent of  $(0.75)_{10} = (.6)_8$
- Therefore, the octal equivalent of  $(73.75)_{10} = (111.6)_8$

# Octal-to-Decimal Conversion

- ❖ The decimal equivalent of a binary number is given by the sum of all the digits multiplied by 8 the power of their respective place values.

The decimal equivalent of the octal number  $(137.21)_8$  is determined as follows:

- The integer part = 137
- The decimal equivalent =  $7 \times 8^0 + 3 \times 8^1 + 1 \times 8^2 = 7 + 24 + 64 = 95$
- The fractional part = .21
- The decimal equivalent =  $2 \times 8^{-1} + 1 \times 8^{-2} = 0.265$
- Therefore, the decimal equivalent of  $(137.21)_8 = (95.265)_{10}$


# Decimal-to-Hexadecimal Conversion

The process of decimal-to-hexadecimal conversion is also similar. Since the hexadecimal number system has a base of 16, the progressive division and multiplication factor in this case is

16. *Solution*

- The integer part = 82

Divisor	Dividend	Remainder
16	82	—
16	5	2
—	0	5



- The hexadecimal equivalent of  $(82)_{10} = (52)_{16}$
- The fractional part = 0.25
- $0.25 \times 16 = 0$  with a carry of 4
- Therefore, the hexadecimal equivalent of  $(82.25)_{10} = (52.4)_{16}$

# Binary Coded Decimal (BCD)

- The BCD equivalent of a decimal number is written by replacing each decimal digit in the integer and fractional parts with its four-bit binary equivalent.
- As an example, the BCD equivalent of  $(23.15)_{10}$  is written as  $(0010\ 0011.0001\ 0101)_{\text{BCD}}$ .

## The Gray Code

The Gray code is unweighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions. The important feature of the Gray code is that *it exhibits only a single bit change from one code word to the next in sequence*. This property is important in many applications, such as shaft position encoders, where error susceptibility increases with the number of bit changes between adjacent numbers in a sequence.

**Binary-to-Gray Code Conversion** Conversion between binary code and Gray code is sometimes useful. The following rules explain how to convert from a binary number to a Gray code word:

1. The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
2. Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 10110 to Gray code is as follows:

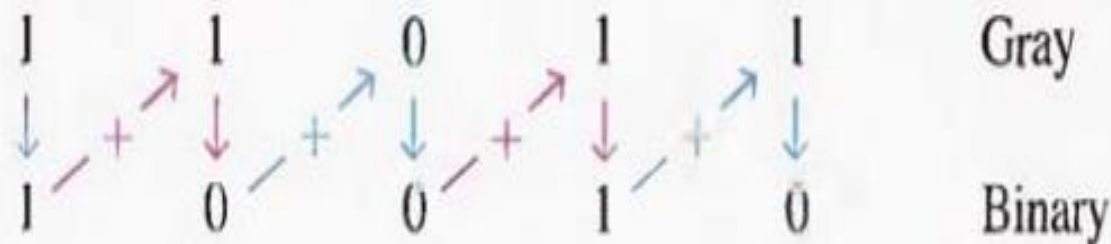
1	+	0	+	1	+	1	+	0	Binary
↓		↓		↓		↓		↓	
1		1		1		0		1	Gray

The Gray code is 11101.

**Gray-to-Binary Conversion** To convert from Gray code to binary, use a similar method; however, there are some differences. The following rules apply:

1. The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

For example, the conversion of the Gray code word 11011 to binary is as follows:



The binary number is 10010.



# Logic Gates

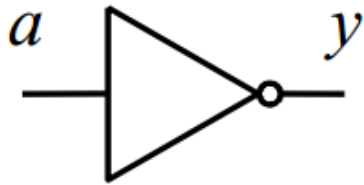
- Basic logic circuits with one or more inputs and one output are known as gates
- Gates are used as the building blocks in the design of more complex digital logic circuits

# Representing Logic Functions

- There are several ways of representing logic functions:
  - Symbols to represent the gates
  - Truth tables
  - Boolean algebra
- We will now describe commonly used gates

# NOT Gate

Symbol



Truth-table

$a$	$y$
0	1
1	0

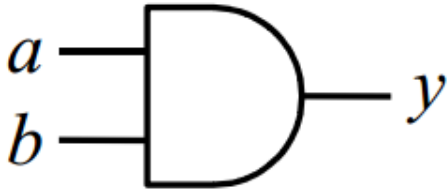
Boolean

$$y = \bar{a}$$

- A NOT gate is also called an 'inverter'.
- $y$  is only TRUE if  $a$  is FALSE.
- Circle (or 'bubble') on the output of a gate implies that it has an inverting (complemented) output.

# AND Gate

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	0
0	1	0
1	0	0
1	1	1

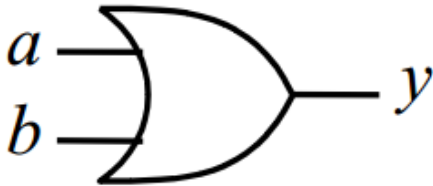
Boolean

$$y = a.b$$

- *y* is only TRUE only if *a* is TRUE and *b* is TRUE
- In Boolean algebra AND is represented by a dot .

# OR Gate

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	0
0	1	1
1	0	1
1	1	1

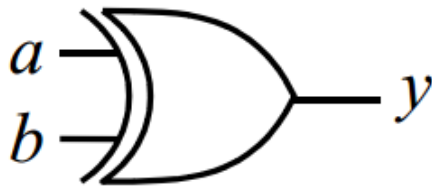
Boolean

$$y = a + b$$

- Y is TRUE if a is TRUE or b is TRUE (or both)
- In Boolean algebra OR is represented by a plus+ sign.

# EXCLUSIVE OR (XOR) Gate

Symbol



Truth-table

$a$	$b$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

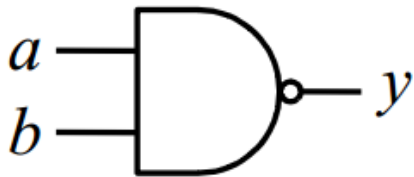
Boolean

$$y = a \oplus b$$

- $y$  is TRUE if  $a$  is TRUE or  $b$  is TRUE (but not both)
- In Boolean algebra XOR is represented by an  $\oplus$  sign.

# NOT AND (NAND) Gate

Symbol



Truth-table

<i>a</i>	<i>b</i>	<i>y</i>
0	0	1
0	1	1
1	0	1
1	1	0

Boolean

$$y = \overline{a.b}$$

- Y is TRUE if a is FALSE or b is FALSE (or both)
- y is FALSE only if a is TRUE and b is TRUE

# Boolean Algebra

- In this section we will introduce the laws of Boolean Algebra.
- We will then see how it can be used to design combinational logic circuits Combinational logic circuits do not have an internal stored state, i.e., they have no memory. Consequently the output is solely a function of the current inputs.
- Later, we will study circuits having a stored internal state, i.e., sequential logic circuits.



# Rules of Boolean Algebra

$$1. A + 0 = A$$

$$2. A + 1 = 1$$

$$3. A \cdot 0 = 0$$

$$4. A \cdot 1 = A$$

$$5. A + A = A$$

$$6. A + \bar{A} = 1$$

$$7. A \cdot A = A$$

$$8. A \cdot \bar{A} = 0$$

$$9. \bar{\bar{A}} = A$$

$$10. A + AB = A$$

$$11. A + \bar{A}B = A + B$$

$$12. (A + B)(A + C) = A + BC$$

# Simplification using Boolean Algebra

## EXAMPLE 4-8

Using Boolean algebra techniques, simplify this expression:

$$AB + A(B + C) + B(B + C)$$

**Solution** The following is not necessarily the only approach.

**Step 1:** Apply the distributive law to the second and third terms in the expression follows:

$$AB + AB + AC + BB + BC$$

**Step 2:** Apply rule 7 ( $BB = B$ ) to the fourth term.

$$AB + AB + AC + B + BC$$

**Step 3:** Apply rule 5 ( $AB + AB = AB$ ) to the first two terms.

$$AB + AC + B + BC$$

**Step 4:** Apply rule 10 ( $B + BC = B$ ) to the last two terms.

$$AB + AC + B$$

**Step 5:** Apply rule 10 ( $AB + B = B$ ) to the first and third terms.

$$B + AC$$

# De Morgan's Theorem

$$\overline{a + b + c + \dots} = \bar{a}.\bar{b}.\bar{c}.\dots$$

$$\overline{a.b.c.\dots} = \bar{a} + \bar{b} + \bar{c} + \dots$$

- In a simple expression like  $a + b + c$  (or  $a.b.c$ ) simply change all operators from OR to AND (or vice versa), complement each term (put a bar over it) and then complement the whole expression, i.e.,

$$a + b + c + \dots = \overline{\bar{a}.\bar{b}.\bar{c}.\dots}$$

$$a.b.c.\dots = \overline{\bar{a} + \bar{b} + \bar{c} + \dots}$$

# De Morgan's Theorem

- For 2 variables we can show  $\overline{a+b} = \bar{a}.\bar{b}$  and  $\overline{a.b} = \bar{a} + \bar{b}$  using a truth table.

$a$	$b$	$\overline{a+b}$	$\overline{a.b}$	$\bar{a}$	$\bar{b}$	$\bar{a}.\bar{b}$	$\bar{a} + \bar{b}$
0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1
1	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0

- Extending to more variables by induction

$$\overline{a+b+c} = \overline{(a+b).c} = (\overline{a.b}).\bar{c} = \bar{a}.\bar{b}.\bar{c}$$

# DeMorgan's Examples

- Simplify  $a.\bar{b} + a.(\overline{b+c}) + b.(\overline{b+c})$   
 $= a.\bar{b} + a.\bar{b}.\bar{c} + b.\bar{b}.\bar{c}$  (DeMorgan)  
 $= a.\bar{b} + a.\bar{b}.\bar{c}$  ( $b.\bar{b} = 0$ )  
 $= a.\bar{b}$  (absorbtion)

# Logic Minimization

- Any Boolean function can be implemented directly using combinational logic (gates)
- However, simplifying the Boolean function will enable the number of gates required to be reduced.  
Techniques available include:
  - Algebraic manipulation (as seen in examples)
  - Karnaugh (K) mapping (a visual approach)
  - Tabular approaches (usually implemented by computer, e.g., Quine-McCluskey)
- K mapping is the preferred technique for up to about 5 variables

# Sum-of-Products Boolean Expressions

- When two or more product terms are summed by boolean addition, the resulting expression is **sum of product(SOP)**.

- e.g

$$AB + ABC$$

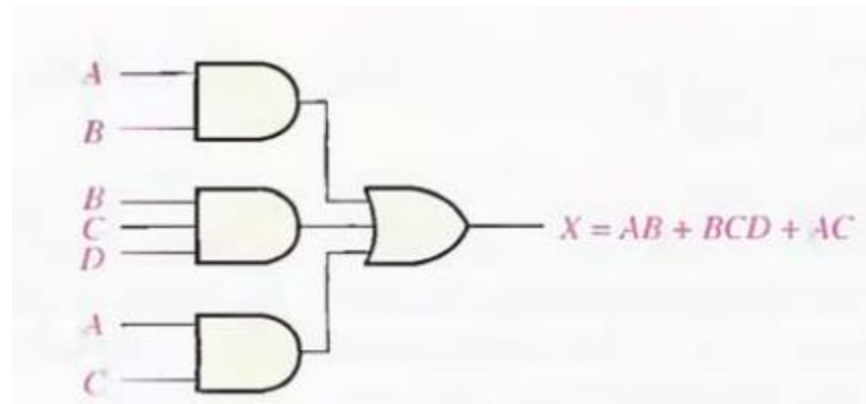
$$ABC + CDE + \overline{BCD}$$

$$\overline{AB} + \overline{ABC} + AC$$

- Implementation of **SOP** expression is simply **ORing** the output of two or more **ANDed** function.

- e,g

Implementation of the SOP  
expression  $AB + BCD + AC$ .



# Standard SOP form

- So far, you have seen SOP expression in which some of the product terms do not contain all of the variables in the domain of the expression, such expression is not standard SOP expression.
- standard SOP expression is one in which all the variables in the domain appear in each product term in the expression.
- To convert non standard expression to standard Sop expression

**Step 1.** Multiply each nonstandard product term by a term made up of the sum of a missing variable and its complement. This results in two product terms. As you know, you can multiply anything by 1 without changing its value.

**Step 2.** Repeat Step 1 until all resulting product terms contain all variables in the domain in either complemented or uncomplemented form. In converting a product term to standard form, the number of product terms is doubled for each missing variable, as Example 4–13 shows.



Convert the following Boolean expression into standard SOP form:

$$\overline{A}\overline{B}C + \overline{A}\overline{B} + AB\overline{C}\overline{D}$$

**Solution** The domain of this SOP expression is  $A, B, C, D$ . Take one term at a time. The first term,  $\overline{A}\overline{B}C$ , is missing variable  $D$  or  $\overline{D}$ , so multiply the first term by  $D + \overline{D}$  as follows:

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

In this case, two standard product terms are the result.

The second term,  $\overline{A}\overline{B}$ , is missing variables  $C$  or  $\overline{C}$  and  $D$  or  $\overline{D}$ , so first multiply the second term by  $C + \overline{C}$  as follows:

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

The two resulting terms are missing variable  $D$  or  $\overline{D}$ , so multiply both terms by  $D + \overline{D}$  as follows:

$$\begin{aligned}\overline{A}\overline{B} &= \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} = \overline{A}\overline{B}C(D + \overline{D}) + \overline{A}\overline{B}\overline{C}(D + \overline{D}) \\ &= \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D\end{aligned}$$

In this case, four standard product terms are the result.

The third term,  $AB\overline{C}\overline{D}$ , is already in standard form. The complete standard SOP form of the original expression is as follows:

$$\overline{A}\overline{B}C + \overline{A}\overline{B} + AB\overline{C}\overline{D} = \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + AB\overline{C}\overline{D}$$

Activate Windows

Go to Settings to activate Windows.

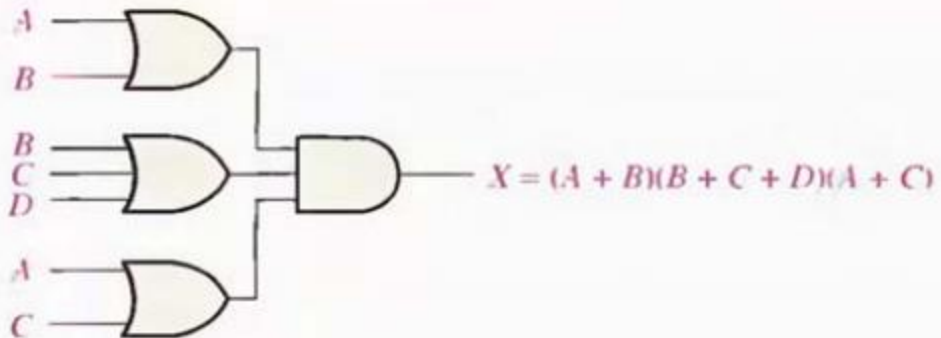
# Sum of product (POS)

- When two or more sum terms are multiplied the resulting expression is **product of sum(POS)** expression.

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

- E.g
- Implementation POS is AND ing the out put of ORed function.

- E.g Implementation of the POS expression  $(A + B)(B + C + D)(A + C)$ .



# Standard POS form

- POS expression in which some of the sum terms do not contain all variables in the domain is non standard POS expression.
- Eg.  $(A + \bar{B} + C)(A + B + \bar{D})(A + \bar{B} + \bar{C} + D)$

A standard POS expression is one in which *all* the variables in the domain appear in each sum term in the expression. For example,

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

- Steps to Convert sum term in to standard POS

**Step 1.** Add to each nonstandard product term a term made up of the product of the missing variable and its complement. This results in two sum terms. As you know, you can add 0 to anything without changing its value.

Step 2 apply  $A + BC = (A + B)(A + C)$

**Step 3.** Repeat Step 1 until all resulting sum terms contain all variables in the domain in either complemented or uncomplemented form.

# Example

Convert the following Boolean expression into standard POS form:

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

**Solution** The domain of this POS expression is  $A, B, C, D$ . Take one term at a time. The first term,  $A + \bar{B} + C$ , is missing variable  $D$  or  $\bar{D}$ , so add  $D\bar{D}$  and apply rule 12 as follows:

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

The second term,  $\bar{B} + C + \bar{D}$ , is missing variable  $A$  or  $\bar{A}$ , so add  $A\bar{A}$  and apply rule 12 as follows:

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

The third term,  $A + \bar{B} + \bar{C} + D$ , is already in standard form. The standard POS form of the original expression is as follows:

$$\begin{aligned} &(A + \bar{B} + C)(\bar{B} + C + \bar{D})(A + \bar{B} + \bar{C} + D) = \\ &(A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})(A + \bar{B} + C + \bar{D})(\bar{A} + \bar{B} + C + \bar{D})(A + \bar{B} + \bar{C} + D) \end{aligned}$$

# Karnaugh (K) mapping

- A Karnaugh map is a graphical representation of the logic system.
- It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions.
- Drawing a Karnaugh map from the truth table involves an additional step of writing the minterm or maxterm expression depending upon whether it is desired to have a minimized sum-of-products or a minimized product-of-sums expression.

# Conti...

- An n-variable Karnaugh map has  $2^n$  squares, and each possible input is allotted a square.
- In the case of a minterm Karnaugh map, '1' is placed in all those squares for which the output is '1'.
- In the case of a maxterm Karnaugh map, a '1' is placed in all those squares for which the output is '0'.
- K-map is an array of cells in which each cell represents a binary value of the input.
- K-map simplification can be used for expression with two, three and four variables.
- The number of cells in k-map is equal to total number of possible input variable combination.
- E.g for 3 variable, number of cells equal to  $2^3 = 8$  cells

# Cell adjacency

- Each cell is adjacent to the cells that are immediately next to it on any of its four sides.
- Note cells at the top row are adjacent to the corresponding cells in the bottom row and cells in the left column are adjacent to the corresponding cells in the right column.
- Cells are not adjacent to cells that are diagonally touch.



## Karnaugh Map Simplification of SOP Expressions

The process that results in an expression containing the fewest possible terms with the fewest possible variables is called **minimization**. After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map.

**Grouping the 1s** You can group 1s on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and to minimize the number of groups.

1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map,  $2^3 = 8$  cells is the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.



4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

### EXAMPLE 4-25

Group the 1s in each of the Karnaugh maps in Figure 4-29.

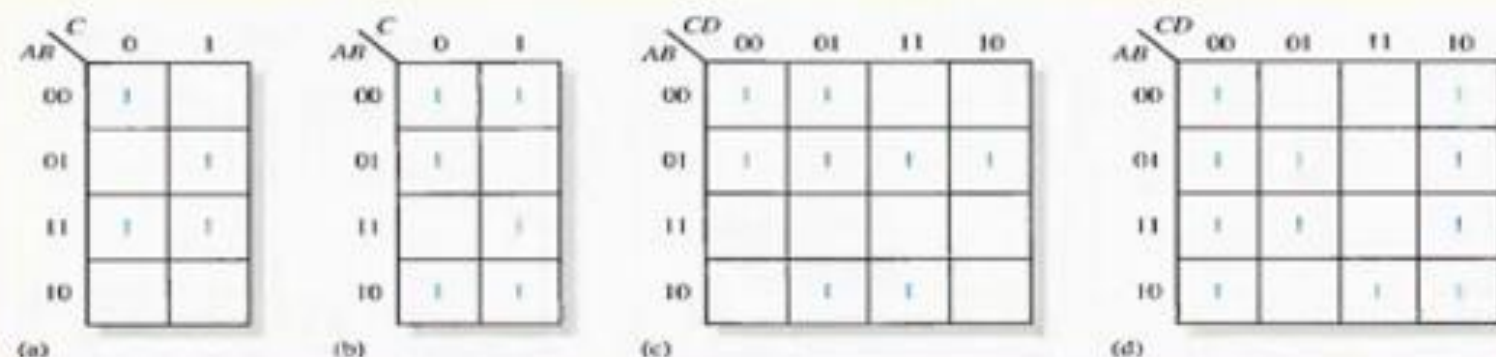
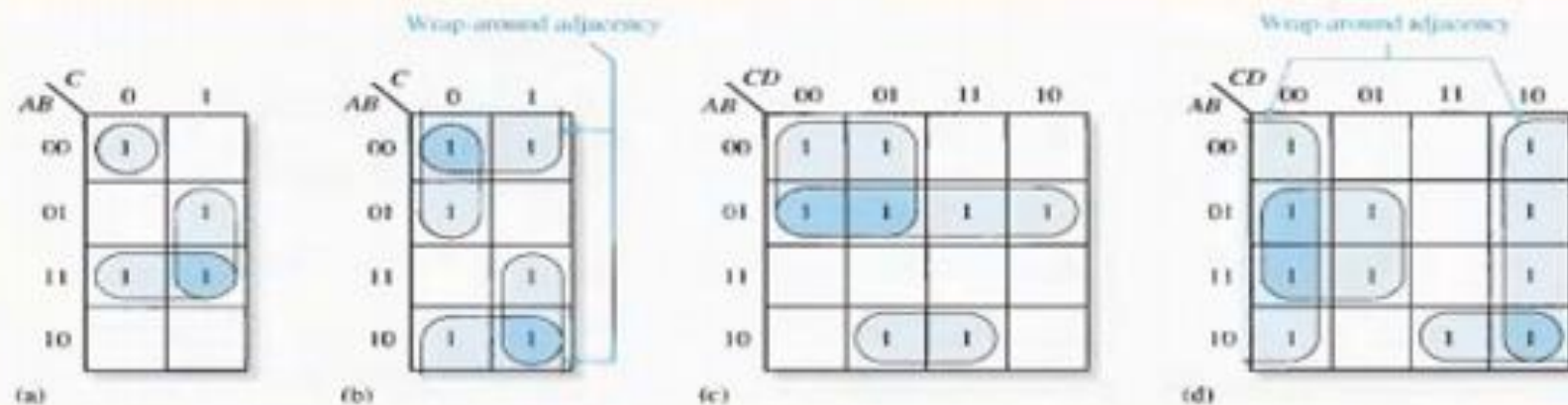


FIGURE 4-29

**Solution** The groupings are shown in Figure 4-30. In some cases, there may be more than one way to group the 1s to form maximum groupings.



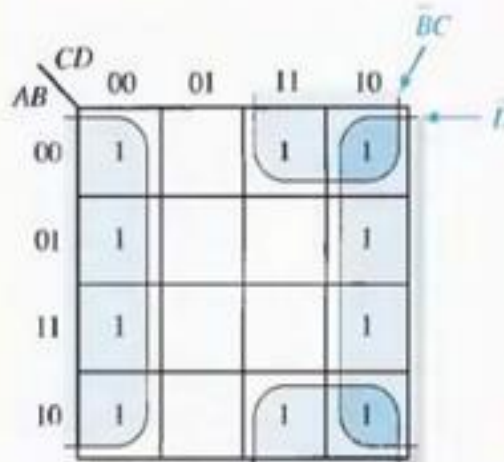
### EXAMPLE 4-29

Use a Karnaugh map to minimize the following SOP expression:

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

**Solution** The first term  $\overline{B}\overline{C}\overline{D}$  must be expanded into  $\overline{A}\overline{B}\overline{C}\overline{D}$  and  $A\overline{B}\overline{C}\overline{D}$  to get the standard SOP expression, which is then mapped; and the cells are grouped as shown in Figure 4-34.

► FIGURE 4-34



Notice that both groups exhibit “wrap around” adjacency. The group of eight is formed because the cells in the outer columns are adjacent. The group of four is formed to pick up the remaining two 1s because the top and bottom cells are adjacent. The product term for each group is shown. The resulting minimum SOP expression is

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

Keep in mind that this minimum expression is equivalent to the original standard expression.