

## **Analog versus digital:**

Analog devices and systems process time-varying signals can take on any value across a continuous range of voltage, current, or other metric, so do digital circuits and systems, the difference that we can pretend that they don't a digital signal is modeled as taking on, at any time, only one as two discrete values, which we call 0 and 1 [low and high, false and true].

Digital computers have been around since the 1940s and have been in widespread commercial use since the 1960s. Yet only in the past 10-20 years has the digital revolution spread to many other aspects of life. Examples of once analog systems that now "gone digital" include the following:

1. **Still picture**, the increased density of digital memory chips has allowed the development of digital cameras which record a picture as a 640x480 or longer array of pixels, where each pixel stores the intensities of its red green, and blue color components as 8 bits each. JPEG format compresses the picture down to as little as 5% of the original storage size.
2. **Video recording**, (DVD) stores video is a highly compressed digital format called MPEG-2. It encodes each other frame as the difference between it and the previous one. The capacity of a single-layer, single sided DVD is about 35 billion bits, 2 hours of high quality video, and a two layer double sided disk four times that capacity.
3. **Audio recordings**, once made exclusively by impressing analog waveforms on to vinyl or magnetic tape, audio recordings now use digital compact discs (CD's). Stores music as a sequence of 16 bit no. . . A full length CD recording (73min) contains over six billion bits of information.

#### 4. Telephone system.

5. **Traffic lights**, stop lights used to be controlled by electromechanical timers that would give the green light to each direction for a predetermined amount of time. Later relays were used in controllers that could activate the light according to the pattern of traffic detected by sensor embedded in the pavement. Today's controllers use microprocessors and can control the lights in ways that maximize vehicle throughput.

#### 6. Movie effects.

### Why digital:

- a) **Reproducibility of results:** Given the same set of input (in both value and time sequence), a properly designed digital circuit always produces exactly the same results. The outputs of an analog circuit vary with temperature, power-supply voltage, aging of components and other factors.
- b) **Ease of design.** Digital design often called logic design “is logical no special math’s skills are needed and the behavior of small logic circuits can be visualized mentally without any special insights about the operation of capacitors, transistors or other devices that require calculus to model.
- c) **Programmability.** Much of digital design is carried out today by writing programs. HDL (Hardware Description languages), simulation and synthesis programs. These software tools are used to test the hardware models behavior before and real hardware is built.

- d) **Speed**, today's digital devices are very fast individual transistors in the fastest integrated circuits can switch in less than 10 Pico seconds and a complete, complex device built from these transistors can examine its inputs and produce an output in less than 2 nanoseconds. This means that such a device can produce 500 million or more results per second.

## **Number Systems And Codes**

Digital systems are built from circuit that process binary digits 0s and 1s yet very few real life problems are based on binary numbers. Digital system designer must establish some correspondence between the binary digits processes by digital circuits.

### **Positional number system**

The traditional number system that we learned in school and use every day in business is called a positional number system. In such a system a number is represented by a string of digits, where each digit position has an associated weight.

For example

$$1734 = (1)(1000)+(7)(100)+(3)(10)+(4)(1)$$

Each weight is a power of 10 corresponding to the digits position. A decimal point allows negative as well as positive powers of 10 to be used.

$$5185.68 = (5)(1000)+(1)(100)+(8)(10)+(5)(1)+(6)(.1)+(8)(.01)$$

in general, a number D of the form  $d_1d_0.d_{-1}d_{-2}$  has the value

$$D = d_1.10^1+d_0.10^0+d_{-1}.10^{-1}+d_{-2}.10^{-2}$$

Here, 10 is called the base or radix of the number system.

In a binary number, the radix point is called the binary point. We use a subscript to indicate the base or radix of each number

$$10011_2 = (1)(16)+(0)(8)+(0)(4)+(1)(2)+(1)(1) = 19_{10}$$

$$\begin{aligned} 101.001_2 &= (1)(4)+(0)(2)+(1)(1)+(0)(.5)+(0)(.025)+(1)(.125) \\ &= 5.125_{10} \end{aligned}$$

The left bit of binary number is called the high order or most significant bit(MSB). The right most is the Low-order or least significant bit (LSB).

**Example.**

The decimal equivalent of the binary number 11010.11 is 26.75 as shown

$$(1)(2^3)+(1)(2^2)+(0)(2^1)+(1)(2^0)+(1)(2^{-1})+(1)(2^{-2}) = 26.75$$

**Octal And Hexadecimal Number:**

Base 10 is important because we use it in every day, business and radix 2 is important because binary numbers can be processed by digital circuit. Other bases have their uses but not as important as the first two. Base 8 and 16 provide convenient shorthand representation for multibit number in a digital system.

The octal number system uses base 8 while the hexadecimal number system uses base 16. The octal system needs 8 digits, so it uses digits 0 – 7 of the decimal system. The hexadecimal system needs 16 digits so it supplements decimal digits 0 – 9 with the letters A- F.

**Examples,**

$$(127.4)_8 = (1)(8)^2+(2)(8)^1+(7)(8)^0+(4)(8)^{-1} = (87.5)_{10}$$

$$(B65F)_{16} = (11)(16)^3+(6)(16)^2+(5)(16)^1+(15)(16)^0 = (46687)_{10}$$

Remember:

A = 10,  
 B = 11,  
 C = 12,  
 D = 13,  
 E = 14,  
 F = 15.

### Number Base Conversions:

We have already discussed how to convert binary, octal and hexadecimal numbers to base 10. It is very easy to convert a binary number to octal. Starting at the binary point and working left, we simply separate the bits into groups of three and replace each group with the corresponding octal digit.

#### Examples,

$$(100011001110)_2 = 100\ 011\ 001\ 110 = (4316)_8$$

$$(11101101110101001)_2 = 011\ 101\ 101\ 110\ 101\ 001 =$$

$$= (35565)_8$$

The procedure for binary-to-hexadecimal conversion is similar, except we use groups of four bits.

#### Example,

$$(100011001110)_2 = 1000\ 1100\ 1110 = (8CE)_{16}$$

$$(11101101110101001)_2 = 0001\ 1101\ 1011\ 1010\ 1001$$

$$= (1DBA9)_{16}$$

In these examples we have freely added zeroes on the left to make the total number bits a multiple of 3 or 4.

If a binary number contains digits to the right of the binary point we can convert them to octal or hexadecimal by starting at the binary point and working right, both the left hand side and right hand sides can be added with zeroes to get multiples of three or four bits.

**Example.**

$$(10.1011001011)_2 = 010 . 101 100 202 100 = (2.5454)_8$$

Converting in the reverse direction:

$$(1357)_8 = 001 011 101 111_2$$

$$(2046.178)_8 = 010 000 100 110.001 111_2$$

$$(BEAD)_{16} = 1011 1110 1010 1101_2$$

$$(9F.46C)_{16} = 1001 1111.0100 0110 1100_2$$

**Complements:**

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation, while the signed-magnitude system negates a number by changing its sign a complement number system negates a number by taking its complement as defined by the system.

**Twos – complement Representation:**

For binary numbers, the base complement is called the twos complement. The MSB of a number in this system serves as the sign bit; a number is called a negative number if and only if its MSB is 1.

**Example,**

|                         |                         |
|-------------------------|-------------------------|
| $17_{10} = 00010001_2$  | $-99_{10} = 10011101_2$ |
| 11101110 complements    | 01100010 complements    |
| <u>+1</u>               | <u>+1</u>               |
| $11101111 = -17_{10}$   | $01100011_2 = 99_{10}$  |
|                         |                         |
| $119_{10} = 01110111_2$ | $-128_{10} = 1000 0000$ |
| 10001000 complements    | 0111 1111 complement    |
| <u>+1</u>               | <u>+1</u>               |
| $10001001_2 = -119$     | $1000 0000 = -128_{10}$ |

Note that in 2s complement there is no negative output and the last previous example gave us -ve no. .In this case we pad the MSB with zero this is caused sign extension.

### Ones-Complement Representation:

The diminished radix-complement system for binary numbers is called the “ones” complement. As in twos complement the MSB is the sign output +ve and 1 if -ve.

#### Example,

$$\begin{array}{ll} 17_{10} = 000\ 1\ 000\ 1 & -99_{10} = 100\ 111\ 00 \\ = 111\ 0\ 111\ 0 = -17_{10} & 01100011 = 99_{10} \\ 119_{10} = 01110111_2 & -127_{10} = 10000000 \\ 10001000 = -119 & 01111111 = 127_{10} \end{array}$$

The main advantages of ones complement system are its symmetry and the ease of complementation. However, the adder design for ones complement numbers is somewhat trickier than the twos- complement added. Also zero detecting circuits in a ones – complement system either must check for both representation of zero, or must always convert 11...11 to 00...00.

### Binary Codes:

Binary coded decimal (BCD). The table bellow gives the 4-bit code for one decimal digit. A number with K decimal digits will require 4K bits in BCD. Decimal 396 is represented in 5CD with 12 bits as 0011 1001 0110.

Binary Coded Decimal (BCD):

| Decimal symbol | BCD digit |
|----------------|-----------|
| 0              | 0000      |
| 1              | 0001      |
| 2              | 0010      |
| 3              | 0011      |
| 4              | 0100      |
| 5              | 0101      |
| 6              | 0110      |
| 7              | 0111      |
| 8              | 1000      |
| 9              | 1001      |

A (BCD) number greater than 10 looks different than its equivalent binary number for example

$$10 = (0001\ 000) \text{ BCD} = (1010)_2$$

$$15 = (0001\ 0101) \text{ BCD} = (1111)_2$$

$$185 = (0001\ 1000\ 0101) \text{ BCD} = (10111001)_2$$

It is important to realize that BCD numbers are decimal numbers and not binary numbers.

### **Binary Storage And Registers**

The binary information in a digital computer must have a physical existence in some information storage medium for storing individual bits. A binary cell is a device that possesses two stable states and is capable of storing one bit of information 0 or 1.

#### **Registers:**

A register is a group of binary cells. A register with  $n$  cells can store any discrete quantity of information that contains  $n$  bits. A 16-bit register has the following content

1100001111001001.

A register with 16 cells can be in one of  $2^{16}$  possible states. If a binary integer then the register can store any binary number from 0 to  $2^{16} - 1$ .

#### **Gray code:**

The output data of many physical systems produce quantities that are continuous. These data must be converted into digital form before they are applied to a digital system. Continuous or analog information is converted into digital form by means of an analog-to-digital converter. It is sometimes convenient to use the gray code shown below to represent the digital data when it is converted from analog data. The advantages of the gray code over the straight binary number sequence is that only one bit changes in the code group changes when going from one number to the next.

Example,

7 → 8

Gray    0100 → 1100    only one bit changes

Binary   0111 → 1000    four bits changes



A typical application of the gray code occurs when analog data are represented by continuous change of a shaft position. The shaft is partitioned into segments and each segment is a signed a number. If adjacent segments are made to correspond with the gray code sequence ambiguity is eliminated.

The gray code is used in applications where the normal sequences of binary number may produce an error or ambiguity during the translation from one number to the next. If binary numbers are used a change from 0111 to 1000 may produce an intermediate erroneous number 1001 if the right most bit takes longer to change in value than the other three bits. The gray code eliminates this problem since only one-bit changes in value during the transition between two number.

**Gray – code**

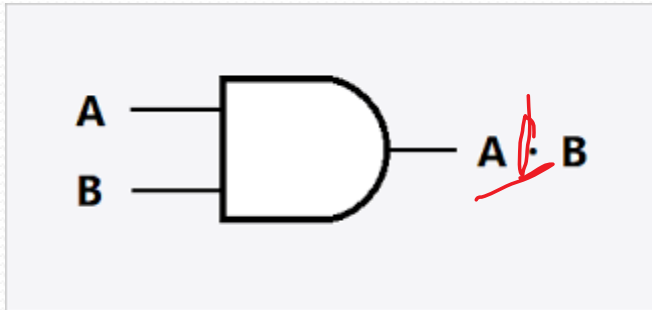
| Gray code | Decimal equivalent |
|-----------|--------------------|
| 0000      | 0                  |
| 0001      | 1                  |
| 0011      | 2                  |
| 0010      | 3                  |
| 0110      | 4                  |
| 0111      | 5                  |
| 0101      | 6                  |
| 0100      | 7                  |
| 1100      | 8                  |
| 1101      | 9                  |
| 1111      | 10                 |
| 1110      | 11                 |
| 1010      | 12                 |
| 1011      | 13                 |
| 1001      | 14                 |
| 1000      | 15                 |



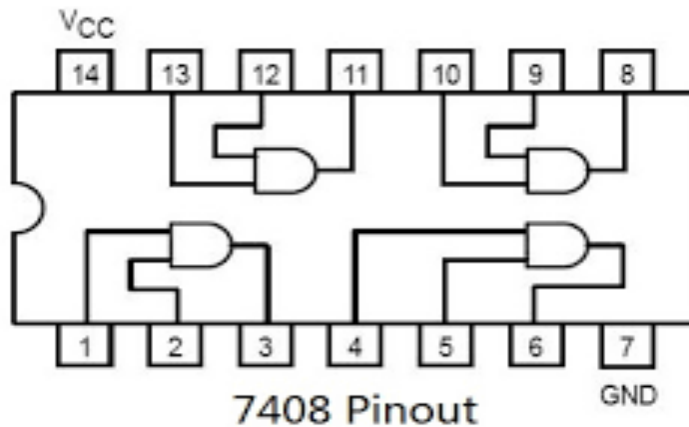
# Digital Logic and Electronic Logical gate (Ch3)

# AND Gate

- Symbol



- IC example (7408)

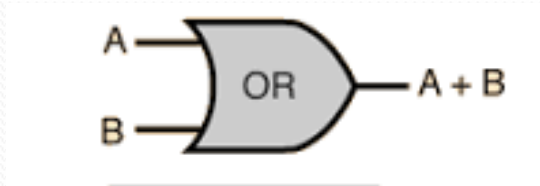


## Truth Table

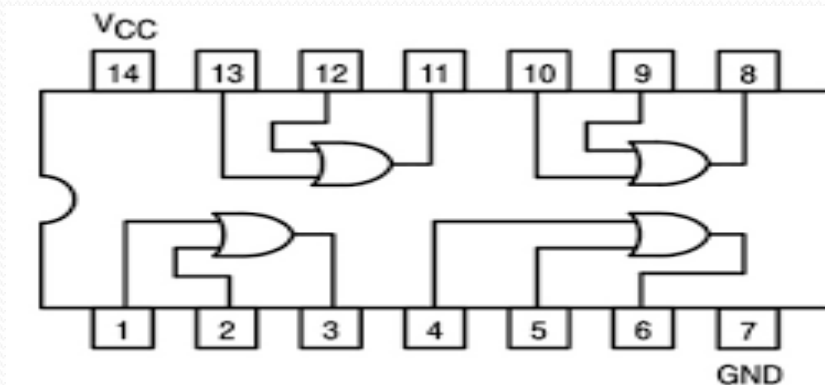
| A | B | Z=(A.B) |
|---|---|---------|
| 0 | 0 | 0       |
| 1 | 0 | 0       |
| 0 | 1 | 0       |
| 1 | 1 | 1       |

# OR Gate

- Symbol



- IC example : 7432

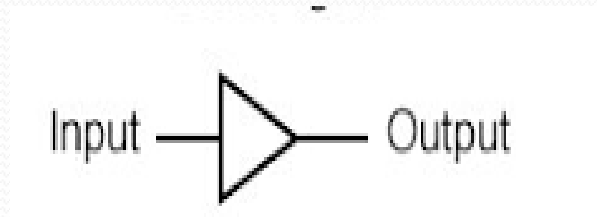


## Truth Table

| A | B | Z=(A+B) |
|---|---|---------|
| 0 | 0 | 0       |
| 1 | 0 | 1       |
| 0 | 1 | 1       |
| 1 | 1 | 1       |

# Buffer

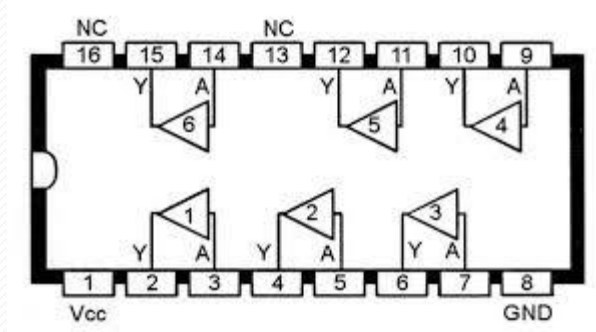
- Symbol



## Truth Table

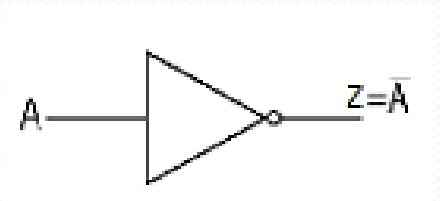
| A | Z=A |
|---|-----|
| 0 | 0   |
| 1 | 1   |

- IC example :



# NOT Gate (Inverter)

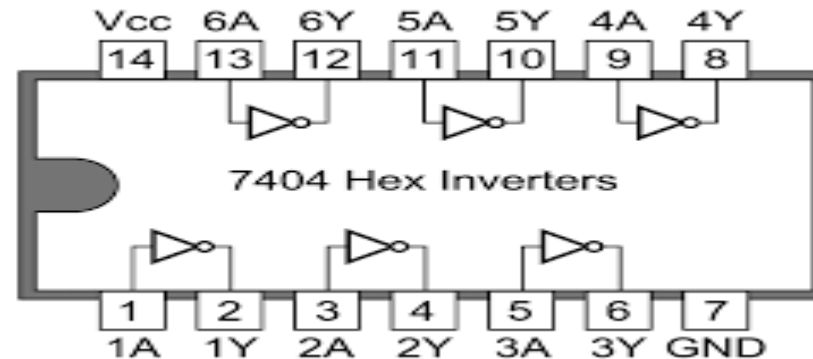
- Symbol



Truth Table

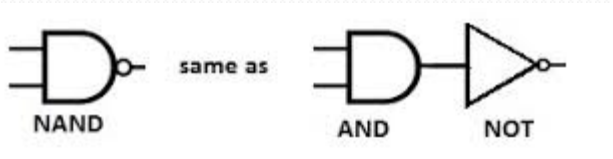
| A | Z = $\bar{A}$ |
|---|---------------|
| 0 | 1             |
| 1 | 0             |

- IC example : 7404

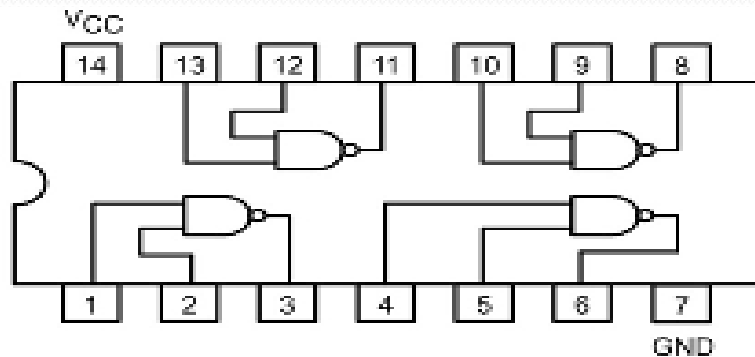


# NAND Gate

- Symbol



- IC example : 7400



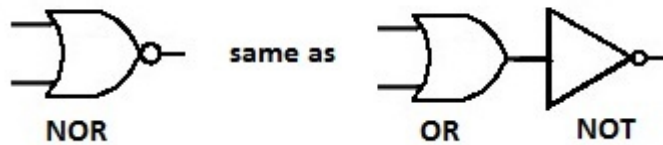
## Truth Table

| A | B | $Z = (\overline{A \cdot B})$ |
|---|---|------------------------------|
| 0 | 0 | 1                            |
| 0 | 1 | 1                            |
| 1 | 0 | 1                            |
| 1 | 1 | 0                            |



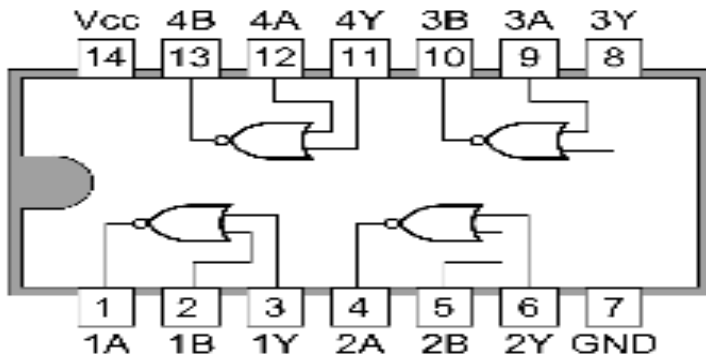
# NOR Gate

- Symbol



- IC example : 7402

7402 Quad 2-input NOR Gates



## Truth Table

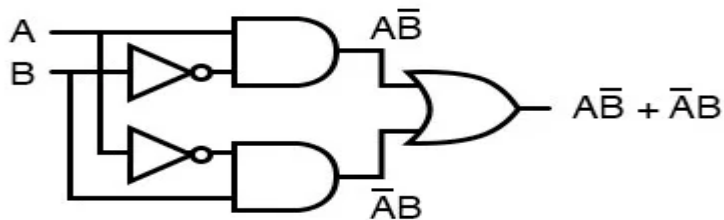
| A | B | $Z = \overline{A + B}$ |
|---|---|------------------------|
| 0 | 0 | 1                      |
| 0 | 1 | 0                      |
| 1 | 0 | 0                      |
| 1 | 1 | 0                      |

# Exclusive Or (EXOR) Gate

- Symbol



... is equivalent to ...

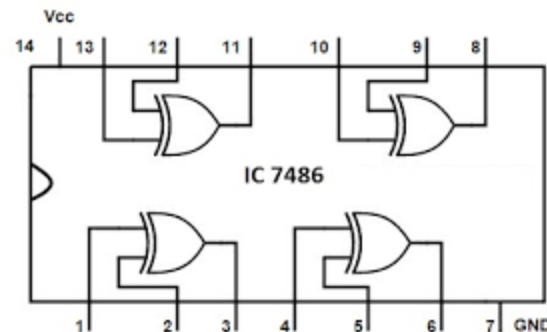


$$A \oplus B = A\bar{B} + \bar{A}B$$

- IC example : 7486

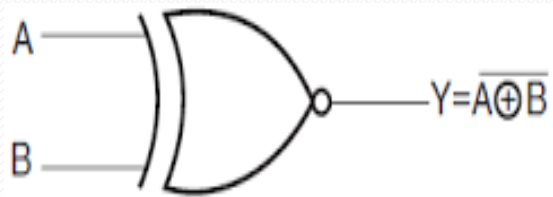
## Truth Table

| A | B | $Z = A.\bar{B} + \bar{A}.B$ |
|---|---|-----------------------------|
| 0 | 0 | 0                           |
| 0 | 1 | 1                           |
| 1 | 0 | 1                           |
| 1 | 1 | 0                           |

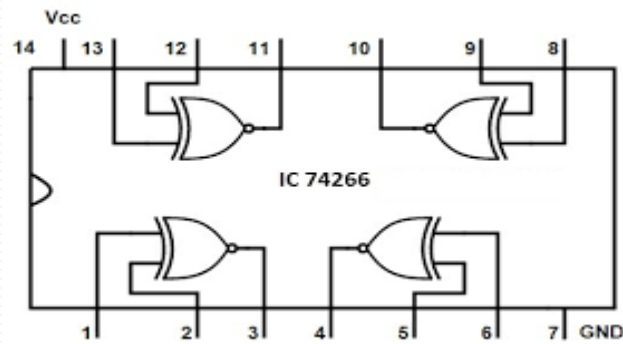


# EXNOR Gate

- Symbol



- IC example : 74266



## Truth Table

| A | B | $Z = A \cdot B + \bar{A}\bar{B}$ |
|---|---|----------------------------------|
| 0 | 0 | 1                                |
| 0 | 1 | 0                                |
| 1 | 0 | 0                                |
| 1 | 1 | 1                                |

- 14 pin IC



# Boolean Algebra

## CH(4)

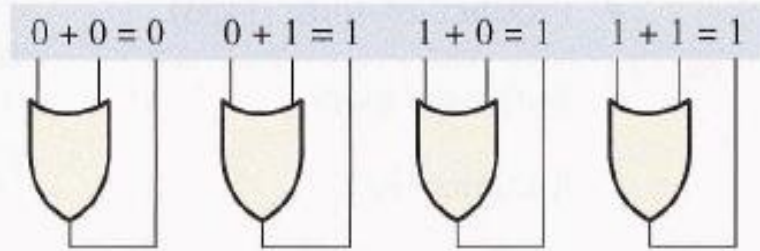
# Boolean Algebra

- Boolean algebra was introduced by George Boole in his first book *The Mathematical Analysis of Logic* (1847).
- **Boolean algebra** is the branch of algebra in which the values of the variable are the Truth values *true* (*one*) and *false* (*zero*),

# Boolean operation and expression

## Boolean Addition

- It is equivalent to the OR operation



OR

- In Boolean algebra the sum term is sum(+), while in circuit it OR gate.

# Boolean operation and expression

## Boolean Addition

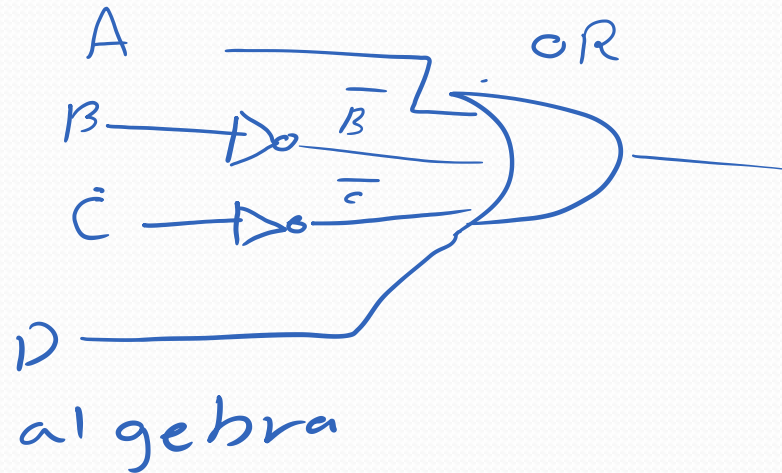
- What is the Value of A,B,C and D? if the sum term is given as:

$$A + \bar{B} + \bar{C} + D = 0$$

$$0 + \bar{1} + \bar{1} + 0 = 0$$

for Linear

$$1_2 + 1_2 = \underline{10}_2$$



X



# Boolean operation and expression

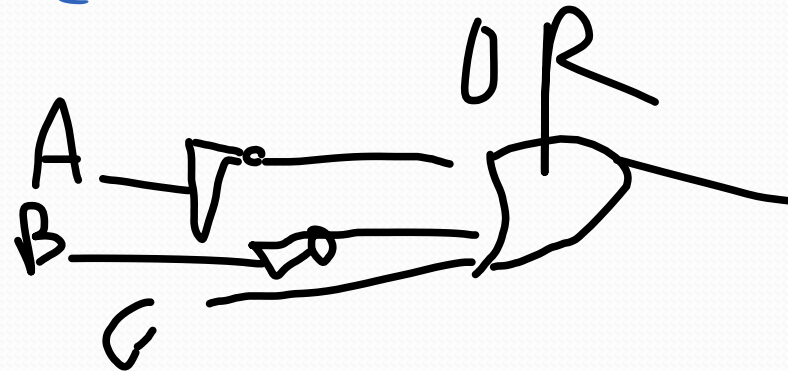
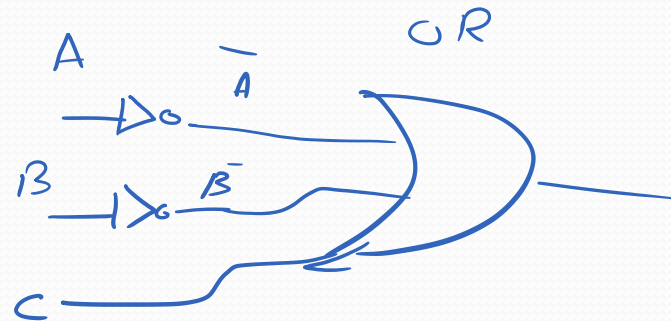
## Boolean Addition

- If  $A=0$ ,  $B=1$  and  $C=1$ , What is the sum term for  $\bar{A}, \bar{B}$  and  $C$

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

$$\bar{0} + \bar{1} + 1 = 1$$

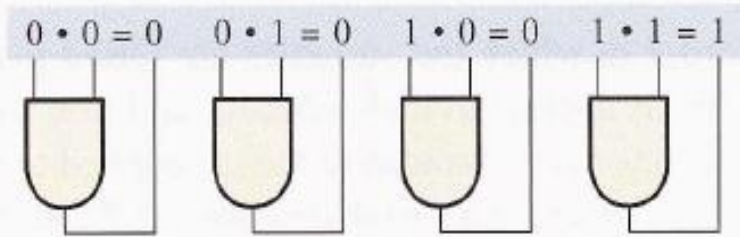
$$1 + 0 + 1$$



# Boolean operation and expression

## Boolean Multiplication

- It is equivalent to the AND operation



- In Boolean algebra the product term is the product in literal ( $\cdot$ ), while in circuit it AND gate.

# Boolean operation and expression

## Boolean Multiplication

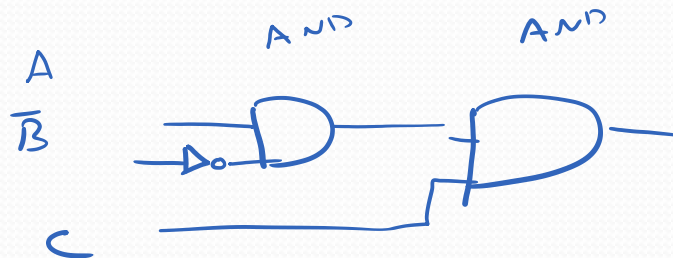
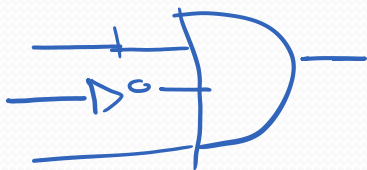
- What is the value of A, B, C and D that make the product term  $A, \bar{B}, C,$  and  $\bar{D}$  equal to 1

# Boolean operation and expression

## Boolean Multiplication

- What is the value product term of  $A, \bar{B}, C$ , if  $A=1, b=0$   
AND  $C=0$

$$(A \cdot \bar{B} \cdot C) = (1 \cdot 1 \cdot 0) = 0$$



# Laws and ruled of Boolean algebra

- The basic laws of Boolean algebra is :
  - 1- The commutative law for addition and multiplication
  - 2- The associative law for addition and multiplication
  - 3- The distributive law

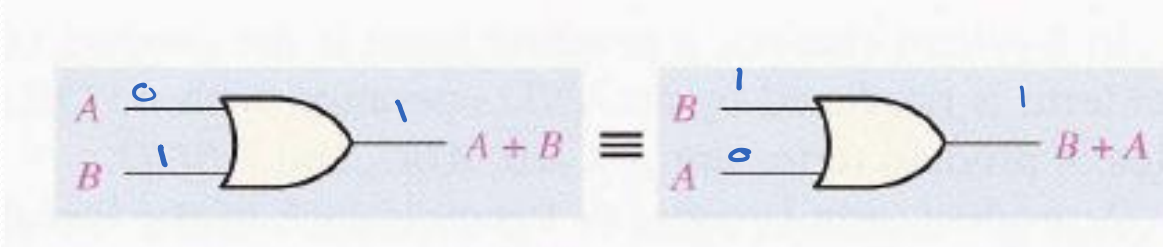
# Laws and ruled of Boolean algebra

## The commutative law for addition

- It is written as

$$A+B=B+A$$

- The commutative law applied to OR gate



$$A = 0$$

$$B = 1$$

# Laws and ruled of Boolean algebra

## The commutative law for multiplication

- It is written as

$$A.B=B.A$$

- The commutative law applied to AND gate



$$A = 1$$

$$B = 0$$

# Laws and ruled of Boolean algebra

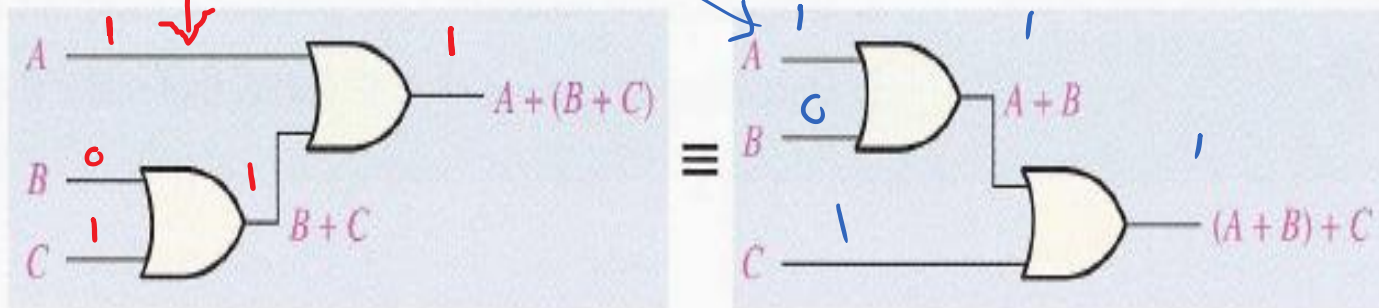
## The associative law for addition

- It is written as

$$(A+B)+C = A+(B+C)$$

- The associative law applied to OR gate

A = 1  
B = 0  
C = 1





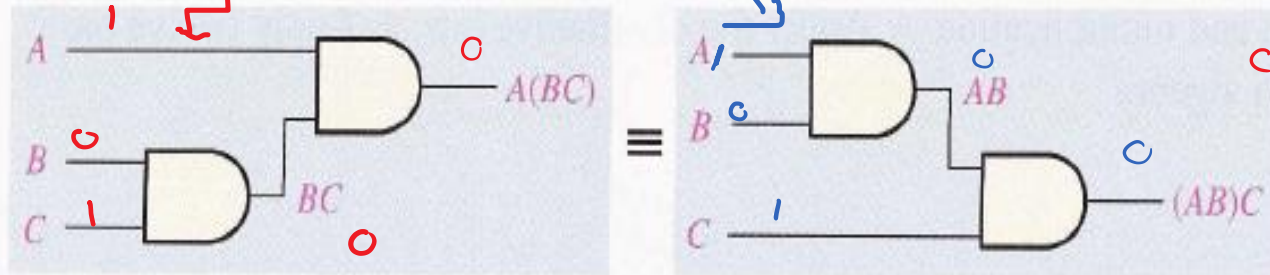
# Laws and ruled of Boolean algebra

## The associative law for Multiplication

- It is written as

$$(AB)C = A(BC)$$

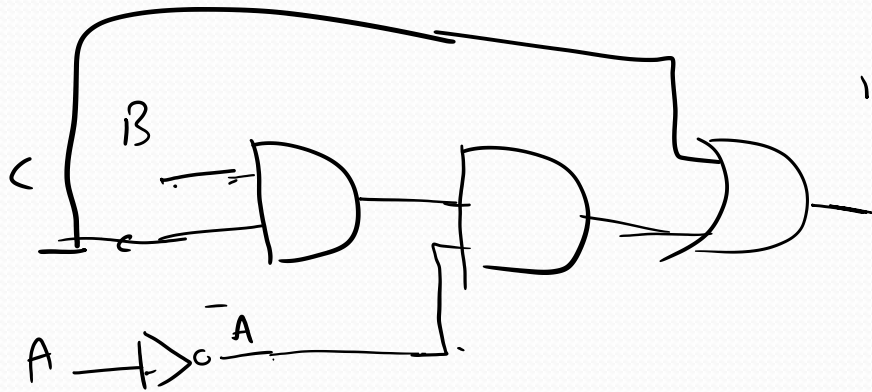
- The associative law applied to AND gate



$$\underline{(\bar{A} \cdot B \cdot C)} + C = ?$$

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

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



# Laws and ruled of Boolean algebra

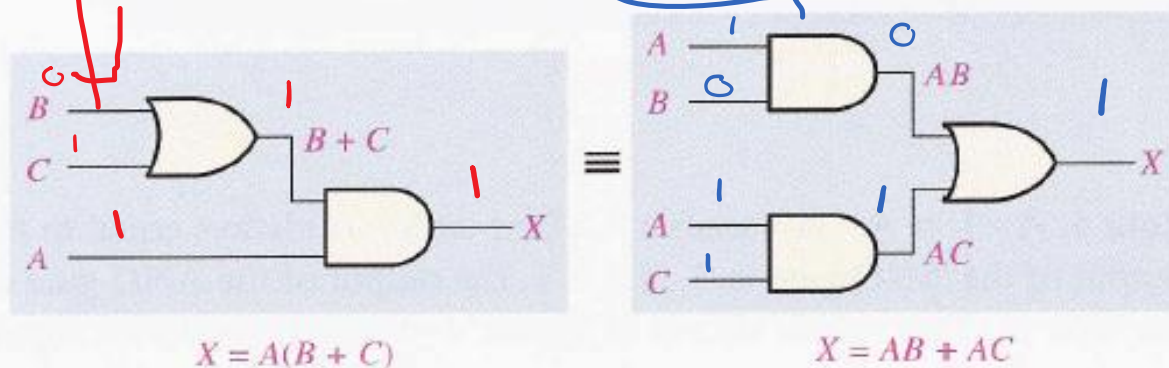
## The distributive law

- It is written for three variables as as

$$\underline{A(B+C)} = \underline{AB+AC}$$

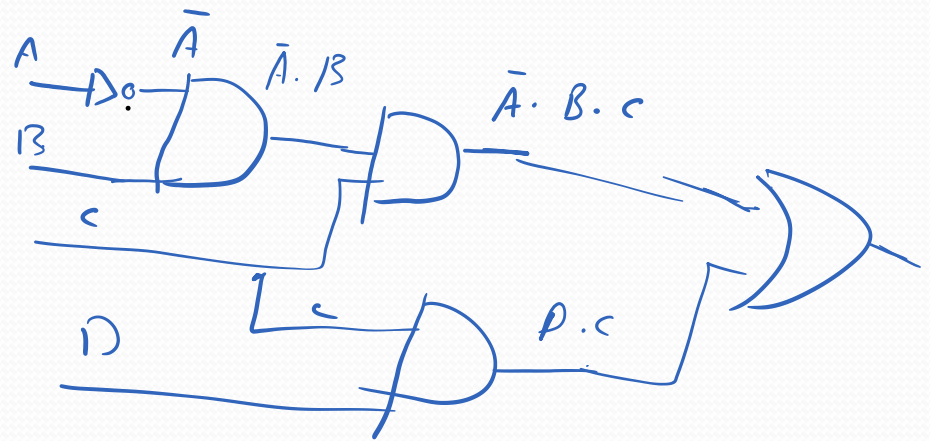
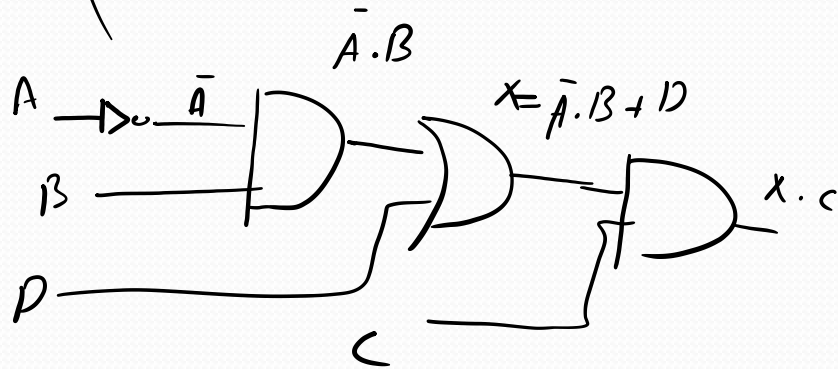
A = 1  
B = 0  
C = 1

- The associative law applied to OR and AND gate



- $(\overline{A}.B)C + D.C$

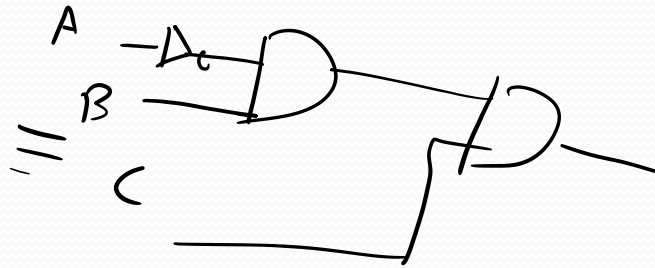
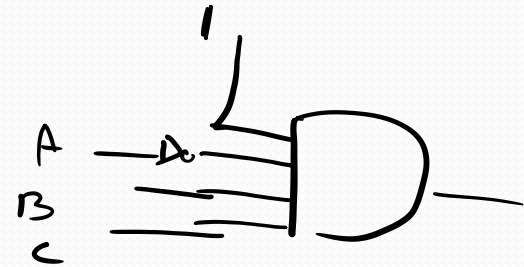
$$\left( \overline{A}.B + D \right).C = \overline{A}.B.C + D.C$$



Λ

$$\bar{A} \cdot B \cdot C \cdot 1$$

assume 4 input and gate



# Laws and ruled of Boolean algebra

- The basic Rule that simplifies Boolean expression is given in the table below :

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

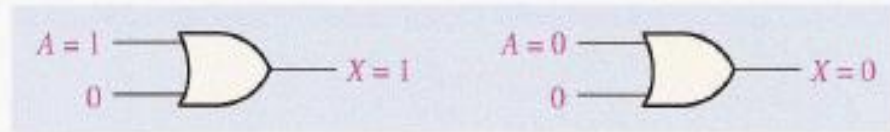
12.  $(A + B)(A + C) = A + BC$

---

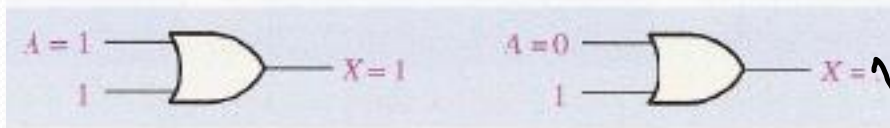
*A, B, or C* can represent a single variable or a combination of variables.

# Laws and ruled of Boolean algebra

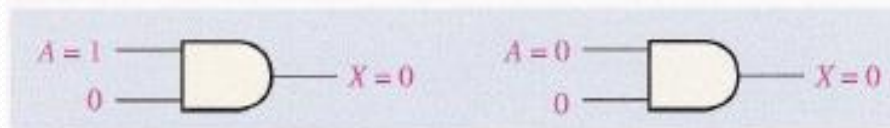
- Rule 1:  $A+0=A$
- Rule 2:  $A+1=1$
- Rule 3:  $A \cdot 0=0$
- Rule 4:  $A \cdot 1=A$



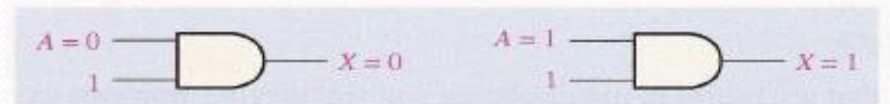
$$X = A + 0 = A$$



$$X = A + 1 = 1$$



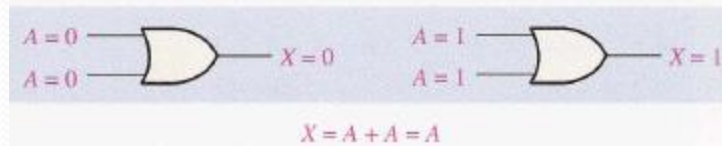
$$X = A \cdot 0 = 0$$



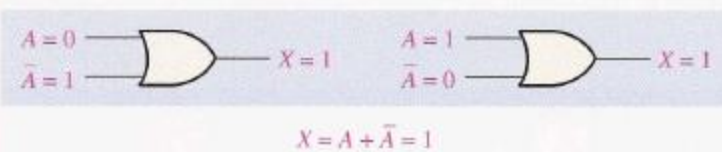
$$X = A \cdot 1 = A$$

# Laws and ruled of Boolean algebra

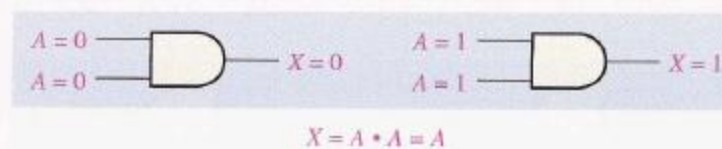
- Rule 5:  $A+A=A$



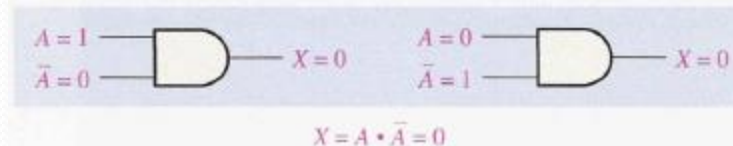
- Rule 6:  $A + \bar{A} = 1$



- Rule 7:  $A \cdot A = A$



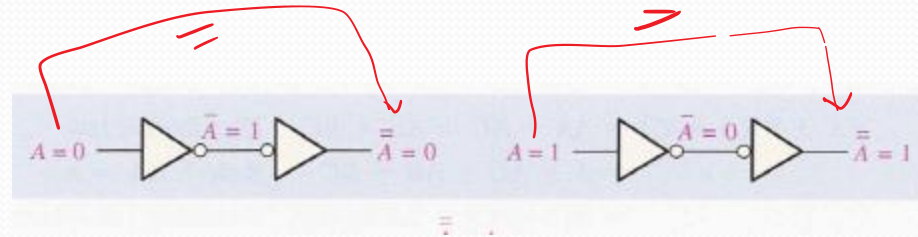
- Rule 8:  $A \cdot \bar{A} = 0$





# Laws and ruled of Boolean algebra

- Rule 9:  $\overline{\overline{A}}=A$



- Rule 10:  $A+AB=A$

| A | B | AB | A + AB |
|---|---|----|--------|
| 0 | 0 | 0  | 0      |
| 0 | 1 | 0  | 0      |
| 1 | 0 | 0  | 1      |
| 1 | 1 | 1  | 1      |

equal

*Proof:*

$=A(1+B)$  *Using distributive law*

$=A.1=A$  *Using rule 2*

*1+B=1*

# Laws and ruled of Boolean algebra

- Rule 11:  $A + B\bar{A} = A + B$

| A | B | $\bar{A}\bar{B}$ | $A + \bar{A}\bar{B}$ | $A + B$ |
|---|---|------------------|----------------------|---------|
| 0 | 0 | 0                | 0                    | 0       |
| 0 | 1 | 1                | 1                    | 1       |
| 1 | 0 | 0                | 1                    | 1       |
| 1 | 1 | 0                | 1                    | 1       |

equal

*Proof:*

$$\begin{aligned}
 &= A + B\bar{A} + B\bar{A} \quad (\text{rule 10}) \\
 &= A + B(A + \bar{A}) \quad (\text{distributive law}) \\
 &= A + B \cdot 1 \quad (\text{Rule 6}) \\
 &= A + B \quad (\text{Rule 4})
 \end{aligned}$$

# Laws and ruled of Boolean algebra

- Rule 12:  $(A+B)(A+C) = A+BC$

Proof :

$$= AA + AB + AC + BC \quad (\text{distributive law})$$

$$= A + AB + AC + BC \quad (\text{rule 7})$$

$$= A + AC + BC \quad (\text{rule 10})$$

$$= A + BC \quad (\text{rule 10})$$



| A | B | C | A + B | A + C | (A + B)(A + C) | BC | A + BC |
|---|---|---|-------|-------|----------------|----|--------|
| 0 | 0 | 0 | 0     | 0     | 0              | 0  | 0      |
| 0 | 0 | 1 | 0     | 1     | 0              | 0  | 0      |
| 0 | 1 | 0 | 1     | 0     | 0              | 0  | 0      |
| 0 | 1 | 1 | 1     | 1     | 1              | 1  | 1      |
| 1 | 0 | 0 | 1     | 1     | 1              | 0  | 1      |
| 1 | 0 | 1 | 1     | 1     | 1              | 0  | 1      |
| 1 | 1 | 0 | 1     | 1     | 1              | 0  | 1      |
| 1 | 1 | 1 | 1     | 1     | 1              | 1  | 1      |

↑ equal ↑

# Boolean Algebra

## CH(4)

# DeMorgan's theorems

- DeMorgan is a mathematician who proposed two theorems in Boolean algebra.
- The theorems provide mathematical proof of the equivalency between NAND and negative-OR, also the equivalency between NOR and negative AND gate.

# DeMorgan's theorems

## First theorem

- The complement of product of variables is the sum of the complement of the variables.

Or :

- The complement of two or more Aneded variables is the equivalent to the OR of the complement of the individual variables.

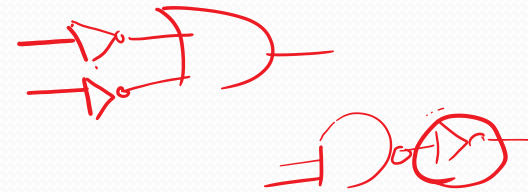
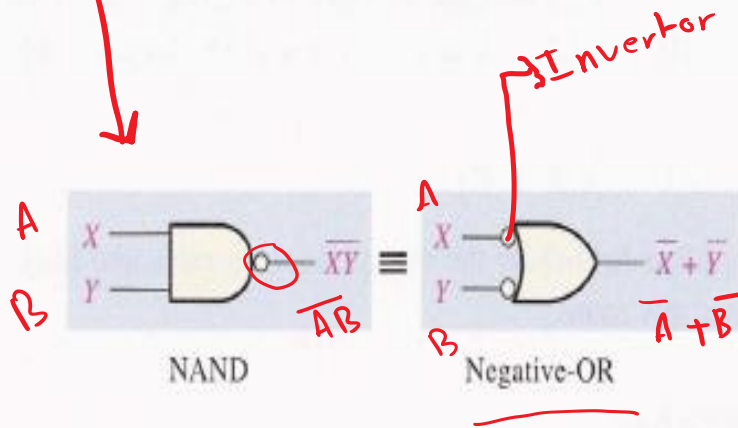
# DeMorgan's theorems

## First theorem

- Mathematical expression :

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

- Logical circuit and truth table



| Inputs |   | Output          |                  |
|--------|---|-----------------|------------------|
| X      | Y | $\overline{XY}$ | $\overline{X+Y}$ |
| 0      | 0 | 1               | 1                |
| 0      | 1 | 1               | 1                |
| 1      | 0 | 1               | 1                |
| 1      | 1 | 0               | 0                |

$$2^2 = 4$$

BCD

$$\overline{xy} = \bar{x} \cdot \bar{y}$$

# DeMorgan's theorems

## Second theorem

- The complement of the sum of the variables is equivalent to product of the complement of the individual variables.

Or:

- The complement of two or more ORed variables is equivalent to negative-And of the complement of the individual variable.



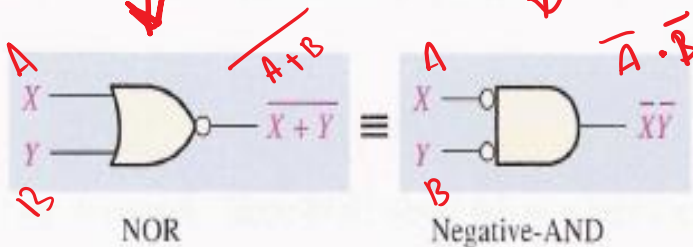
# DeMorgan's theorems

## Second theorem

- Mathematical expression :

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

- Logical circuit and truth table



| Inputs |   | Output           |                  |
|--------|---|------------------|------------------|
| X      | Y | $\overline{X+Y}$ | $\bar{X}\bar{Y}$ |
| 0      | 0 | 1                | 1                |
| 0      | 1 | 0                | 0                |
| 1      | 0 | 0                | 0                |
| 1      | 1 | 0                | 0                |

0 = 0

# DeMorgan's theorems

## EXAMPLE 4-3

Apply DeMorgan's theorems to the expressions  $\overline{XYZ}$  and  $\overline{X + Y + Z}$ .

*Solution*

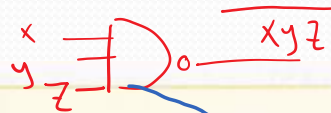
$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{X + Y + Z} = \overline{X} \overline{Y} \overline{Z}$$

*Related Problem*

Apply DeMorgan's theorem to the expression  $\overline{\overline{X} + \overline{Y} + \overline{Z}}$ .

$$X \cdot Y \cdot Z = \overline{\overline{X} + \overline{Y} + \overline{Z}}$$



## EXAMPLE 4-4

Apply DeMorgan's theorems to the expressions  $\overline{WXYZ}$  and  $\overline{W + X + Y + Z}$ .

*Solution*

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{W + X + Y + Z} = \overline{W} \overline{X} \overline{Y} \overline{Z}$$

*Related Problem*

Apply DeMorgan's theorem to the expression  $\overline{\overline{\overline{WXYZ}}}$ .

$$= \overline{\overline{\overline{W} + \overline{\overline{X} + \overline{\overline{Y} + \overline{\overline{Z}}}}} = W + X + Y + Z$$



# DeMorgan's theorems

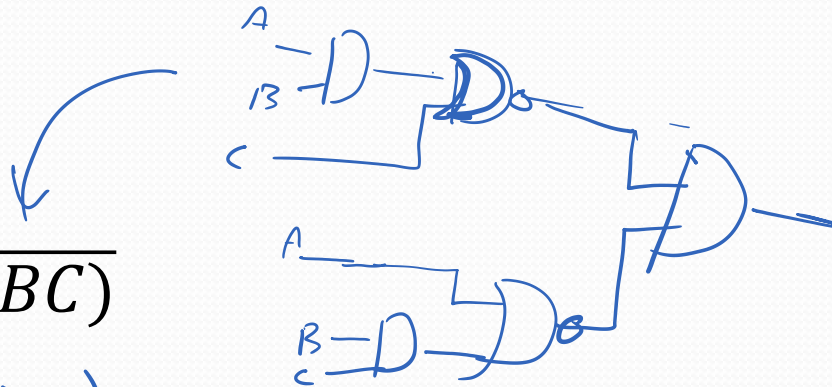
• Example:

$$\overline{(AB + C)} \cdot \overline{(A + BC)}$$

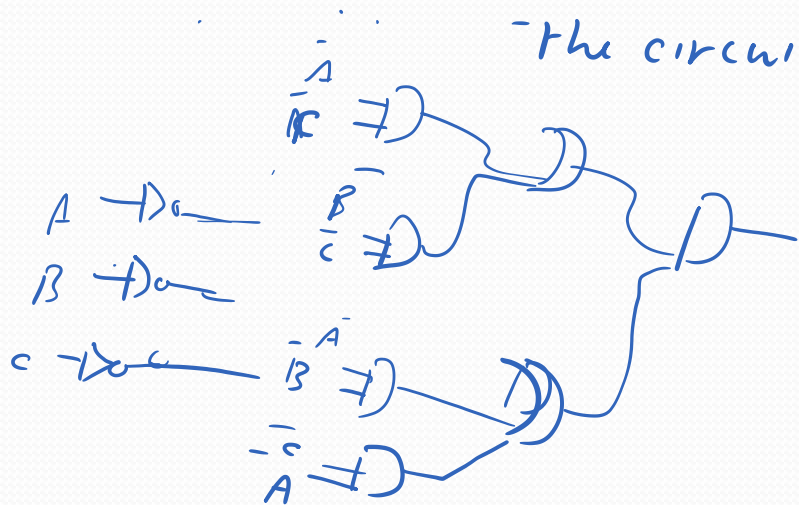
$$(\overline{AB} \cdot \overline{C}) \cdot (\overline{A} \cdot \overline{BC})$$

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

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



H<sub>1</sub>  
 =  
 simplify & draw the circuit



# DeMorgan's theorems

- Example :

$$\overline{\overline{A + BC} + D(E + \overline{F})}$$

*(Note: In the original image, red brackets under 'A + BC' and 'D(E + F-bar)' are labeled 'x' and 'y' respectively.)*

$$\overline{x + y} = \overline{x} \cdot \overline{y} = \overline{(A + BC)} \cdot \overline{D(E + \overline{F})}$$

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

$$= (A + B\overline{C}) \cdot (\overline{D} + (E + \overline{F}))$$

*H.W*  
*Draw*  
*the logical circuit*  
*for both expressions*

# DeMorgan's theorems

- Example :

b, c }  $\Rightarrow$  H.C

Apply DeMorgan's theorems to each of the following expressions:

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

$\overline{x \cdot D}$

$\overline{x} + \overline{D}$

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

(b)  $\overline{ABC + DEF}$

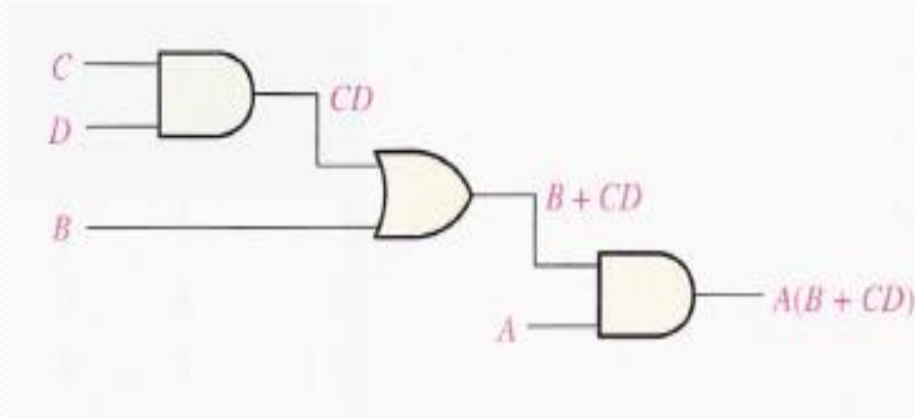
$(\overline{A} \cdot \overline{B} \cdot \overline{C}) + \overline{D}$

(c)  $\overline{\overline{AB} + \overline{CD} + EF}$

$\overline{x} \cdot \overline{y} \cdot \overline{z}$

# Truth Table FOR Logic Circuit

- Logic circuit



- The Boolean expression  $F = A(B + CD)$

# Truth Table FOR Logic Circuit

**Evaluating the Expression** To evaluate the expression  $A(B + CD)$ , first find the values of the variables that make the expression equal to 1, using the rules for Boolean addition and multiplication. In this case, the expression equals 1 only if  $A = 1$  and  $B + CD = 1$  because

$(B + CD) = 1$   
 $A(B + CD) = 1 \cdot 1 = 1$

Now determine when the  $B + CD$  term equals 1. The term  $B + CD = 1$  if either  $B = 1$  or  $CD = 1$  or if both  $B$  and  $CD$  equal 1 because

$$B + CD = 1 + 0 = 1$$

$$B + \underline{CD} = 0 + 1 = 1$$

$$B + CD = 1 + 1 = 1$$

The term  $CD = 1$  only if  $C = 1$  and  $D = 1$ .

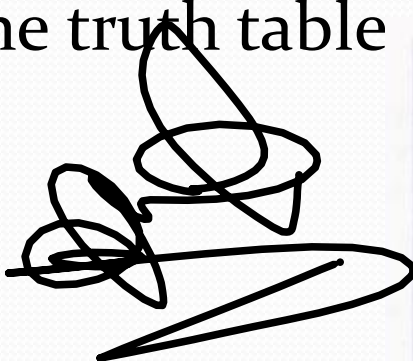
To summarize, the expression  $A(B + CD) = 1$  when  $A = 1$  and  $B = 1$  regardless of the values of  $C$  and  $D$  or when  $A = 1$  and  $C = 1$  and  $D = 1$  regardless of the value of  $B$ . The expression  $A(B + CD) = 0$  for all other value combinations of the variables.

| INPUTS |   |   |
|--------|---|---|
| A      | B | C |
| 0      | 0 | 0 |
| 0      | 0 | 0 |
| 0      | 0 | 1 |
| 0      | 0 | 1 |
| 0      | 1 | 0 |
| 0      | 1 | 0 |
| 0      | 1 | 1 |
| 0      | 1 | 1 |
| 1      | 0 | 0 |
| 1      | 0 | 0 |
| 1      | 0 | 1 |
| 1      | 0 | 1 |
| 1      | 1 | 0 |
| 1      | 1 | 0 |
| 1      | 1 | 1 |
| 1      | 1 | 1 |



# Truth Table FOR Logic Circuit

- The truth table



| INPUTS |   |   |   | OUTPUT      |
|--------|---|---|---|-------------|
| A      | B | C | D | $A(B + CD)$ |
| 0      | 0 | 0 | 0 | 0           |
| 0      | 0 | 0 | 1 | 0           |
| 0      | 0 | 1 | 0 | 0           |
| 0      | 0 | 1 | 1 | 0           |
| 0      | 1 | 0 | 0 | 0           |
| 0      | 1 | 0 | 1 | 0           |
| 0      | 1 | 1 | 0 | 0           |
| 0      | 1 | 1 | 1 | 0           |
| 1      | 0 | 0 | 0 | 0           |
| 1      | 0 | 0 | 1 | 0           |
| 1      | 0 | 1 | 0 | 0           |
| 1      | 0 | 1 | 1 | 1           |
| 1      | 1 | 0 | 0 | 1           |
| 1      | 1 | 0 | 1 | 1           |
| 1      | 1 | 1 | 0 | 1           |
| 1      | 1 | 1 | 1 | 1           |

$$2^4 = 16$$

$$(0 - 15)_{16}$$

$$2^4 - 1 = 15_{10}$$

# Simplification Using Boolean Algebra

## EXAMPLE 4-8

Using Boolean algebra techniques, simplify this expression:

$$\underline{AB + A(B + C) + B(B + C)}$$

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

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

$$AB + \overbrace{AB}^{AB} + AC + \overbrace{BB}^B + BC$$

**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.

$$AB + AC + \overbrace{B + BC} = B$$

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

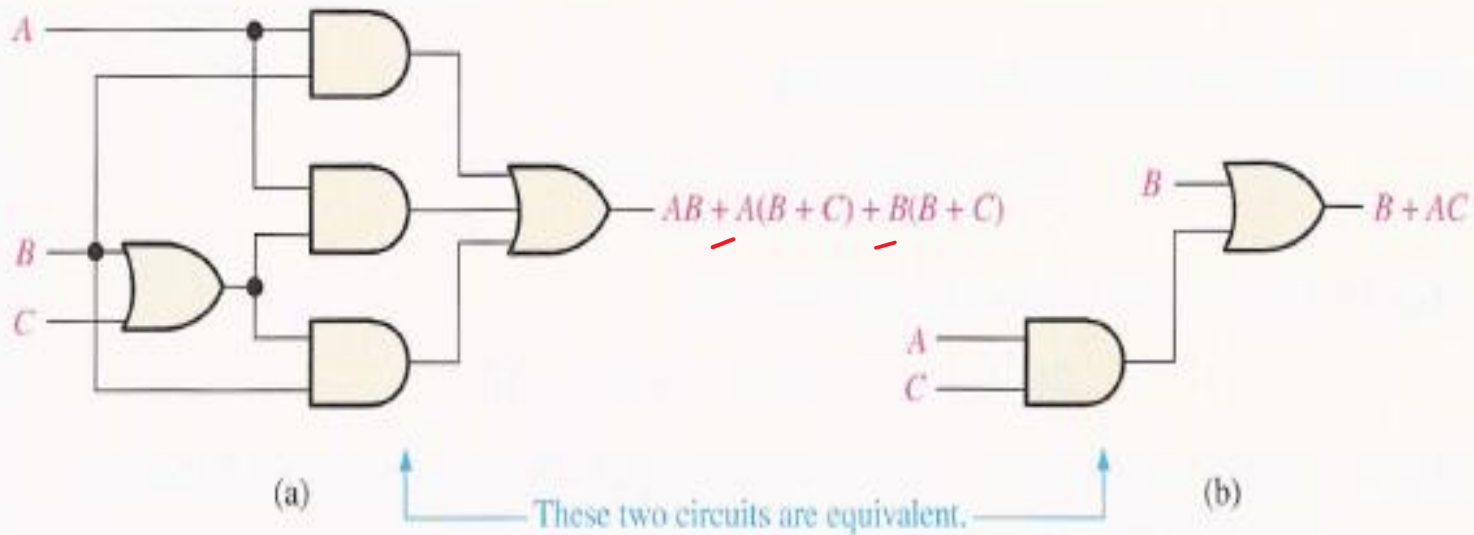
$$\underline{AB} + AC + \underline{B}$$

$$AB + B = B$$

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

$$B + AC$$

# Simplification Using Boolean Algebra



# Simplification Using Boolean Algebra

Simplify the following Boolean expression:

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

Note that brackets and parentheses mean the same thing: the term inside is multiplied (ANDed) with the term outside.

**Solution** **Step 1:** Apply the distributive law to the terms within the brackets.

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

**Step 2:** Apply rule 8 ( $\overline{B}B = 0$ ) to the second term within the parentheses.

$$(\overline{A}\overline{B}C + A \cdot 0 \cdot D + \overline{A}\overline{B})C$$

**Step 3:** Apply rule 3 ( $A \cdot 0 \cdot D = 0$ ) to the second term within the parentheses.

$$(\overline{A}\overline{B}C + 0 + \overline{A}\overline{B})C$$

**Step 4:** Apply rule 1 (drop the 0) within the parentheses.

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

**Step 5:** Apply the distributive law.

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

**Step 6:** Apply rule 7 ( $CC = C$ ) to the first term.

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

**Step 7:** Factor out  $\overline{B}C$ .

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

**Step 8:** Apply rule 6 ( $A + \overline{A} = 1$ ).

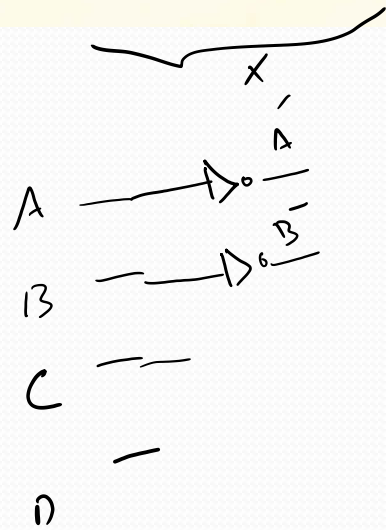
$$\overline{B}C \cdot 1$$

**Step 9:** Apply rule 4 (drop the 1).

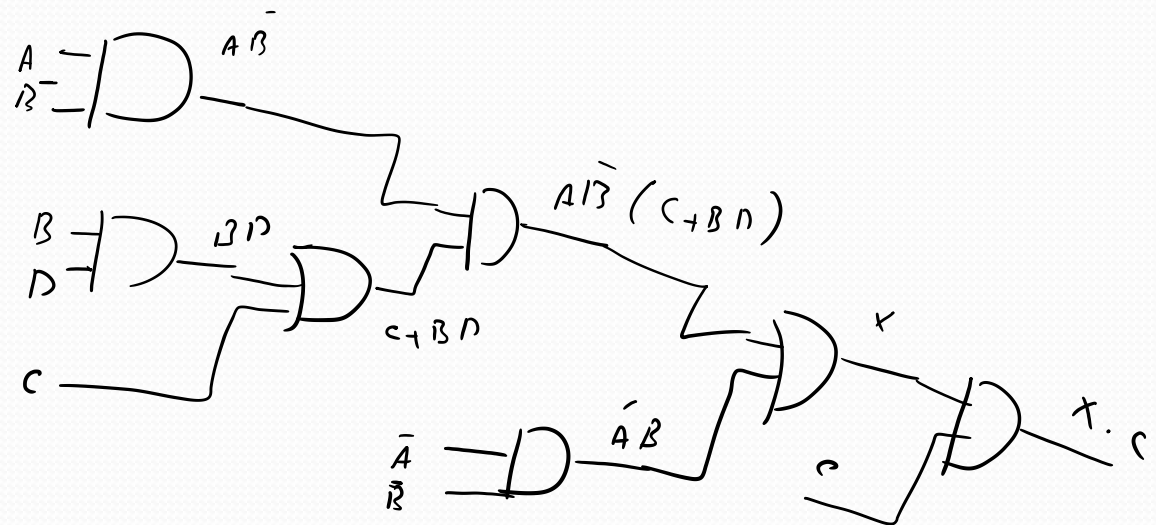
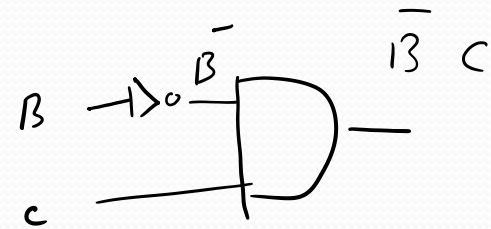
$$\overline{B}C$$

# Simplification Using Boolean Algebra

$$[A\bar{B}(C + BD) + \bar{A}\bar{B}]C$$



$$\bar{B}C$$



# Simplification Using Boolean Algebra

## EXAMPLE 4-11

Simplify the following Boolean expression:

$$\overline{AB + AC} + \overline{A}BC$$

**Solution** **Step 1:** Apply DeMorgan's theorem to the first term.

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

**Step 2:** Apply DeMorgan's theorem to each term in parentheses.

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

**Step 3:** Apply the distributive law to the two terms in parentheses.

$$\overline{A} \underbrace{\overline{A}\overline{A} + \overline{A}\overline{C} + \overline{A}\overline{B} + \overline{B}\overline{C} + \overline{A}BC}$$

**Step 4:** Apply rule 7 ( $\overline{A}\overline{A} = \overline{A}$ ) to the first term, and apply rule 10 [ $\overline{A}\overline{B} + \overline{A}BC = \overline{A}\overline{B}(1 + C) = \overline{A}\overline{B}$ ] to the third and last terms.

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

**Step 5:** Apply rule 10 [ $\overline{A} + \overline{A}\overline{C} = \overline{A}(1 + \overline{C}) = \overline{A}$ ] to the first and second terms.

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

**Step 6:** Apply rule 10 [ $\overline{A} + \overline{A}\overline{B} = \overline{A}(1 + \overline{B}) = \overline{A}$ ] to the first and second terms.

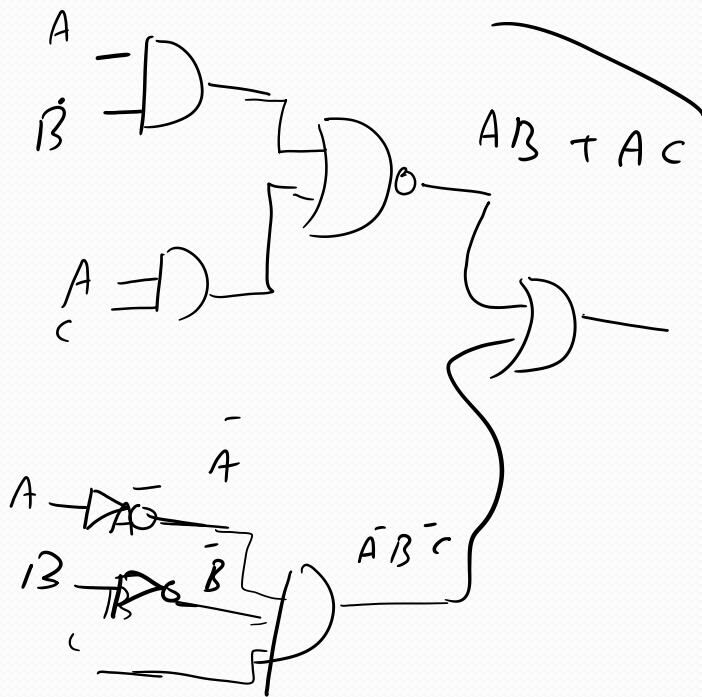
$$\overline{A} + \overline{B}\overline{C}$$

$$A + AC = A$$

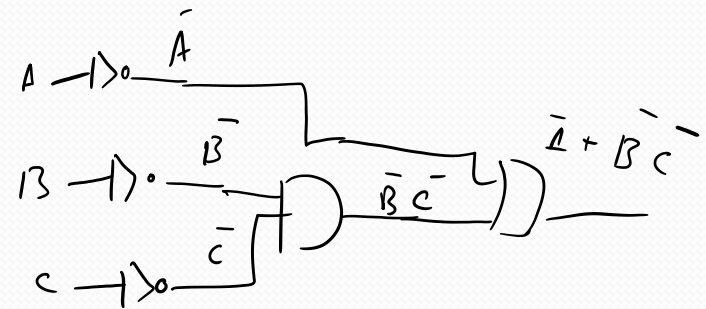
$$A\overline{A}$$

# Simplification Using Boolean Algebra

$$\overline{AB} + AC + \overline{A}BC$$



$$\overline{A} + \overline{BC}$$



# Boolean Algebra

## CH(4)



# Standard forms of Boolean Algebra

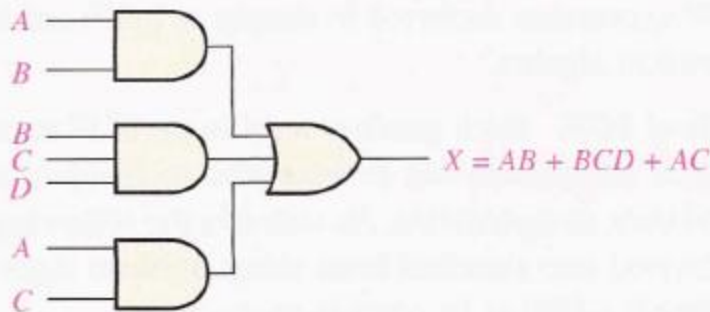
- The two standard forms of Boolean expression :
  - Sum of Product (SOP)
  - Product of Sum (POS)
- All Boolean expressions can be written in one of the forms either SOP or POS.

# Standard form of Boolean Algebra

- The SOP : when two or more product variables are summed .

- Example : 
$$AB + ABC$$
$$ABC + CDE + \overline{BCD}$$
$$\overline{AB} + \overline{ABC} + AC$$

- Circuit example

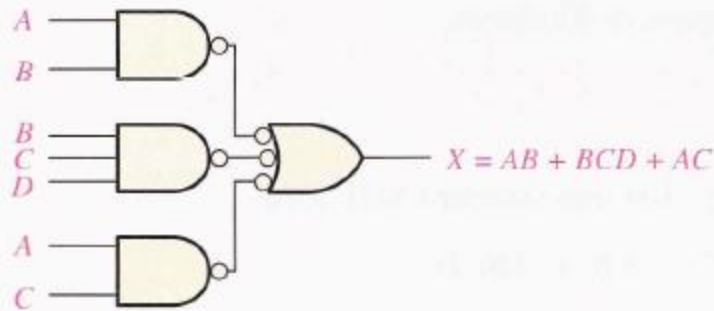


◀ **FIGURE 4-18**

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

# Standard form of Boolean Algebra

- Circuit example



◀ **FIGURE 4-19**

This NAND/NAND implementation is equivalent to the AND/OR in Figure 4-18.

# Standard form of Boolean Algebra

- Conversion of general expression to SOP

## EXAMPLE 4-12

Convert each of the following Boolean expressions to SOP form:

$$(a) AB + B(CD + EF) \quad (b) (A + B)(B + C + D) \quad (c) \overline{\overline{A + B}} + C$$

*Solution* (a)  $AB + B(CD + EF) = AB + BCD + BEF$

(b)  $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

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

# Standard form of Boolean Algebra

- The standard form of SOP, in which all variables in the domain appear in each product.
- Example :  $\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D}$
- It is important in constructing truth tables, and in Karnaugh map simplification.

# Standard form of Boolean Algebra

- Converting Boolean expression to the standard form of SOP:
  - 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.

# Standard form of Boolean Algebra

## EXAMPLE 4-13

Convert the following Boolean expression into standard SOP form:

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

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

$$\overline{A}BC = \overline{A}BC(D + \overline{D}) = \overline{A}BCD + \overline{A}BC\overline{D}$$

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:

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

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

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

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

# Standard form of Boolean Algebra

- Binary representation for SOP

*Binary Representation of a Standard Product Term* A standard product term is equal to 1 for only one combination of variable values. For example, the product term  $\overline{A}BC\overline{D}$  is equal to 1 when  $A = 1, B = 0, C = 1, D = 0$ , as shown below, and is 0 for all other combinations of values for the variables.

$$\overline{A}BC\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

In this case, the product term has a binary value of 1010 (decimal ten).

Remember, a product term is implemented with an AND gate whose output is 1 only if each of its inputs is 1. Inverters are used to produce the complements of the variables as required.

**An SOP expression is equal to 1 only if one or more of the product terms in the expression is equal to 1.**



# Standard form of Boolean Algebra

- Binary representation for SOP

## EXAMPLE 4-14

Determine the binary values for which the following standard SOP expression is equal to 1:

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

**Solution** The term  $ABCD$  is equal to 1 when  $A = 1, B = 1, C = 1,$  and  $D = 1.$

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term  $\overline{A}\overline{B}\overline{C}D$  is equal to 1 when  $A = 1, B = 0, C = 0,$  and  $D = 1.$

$$\overline{A}\overline{B}\overline{C}D = 1 \cdot \overline{0} \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The term  $\overline{A}\overline{B}C\overline{D}$  is equal to 1 when  $A = 0, B = 0, C = 0,$  and  $D = 0.$

$$\overline{A}\overline{B}C\overline{D} = \overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

The SOP expression equals 1 when any or all of the three product terms is 1.

# Boolean Algebra

## CH(4)

# Standard forms of Boolean Algebra

- The two standard forms of Boolean expression :
  - Sum of Product (SOP)
  - Product of Sum (POS)
- All Boolean expressions can be written in one of the forms either SOP or POS.

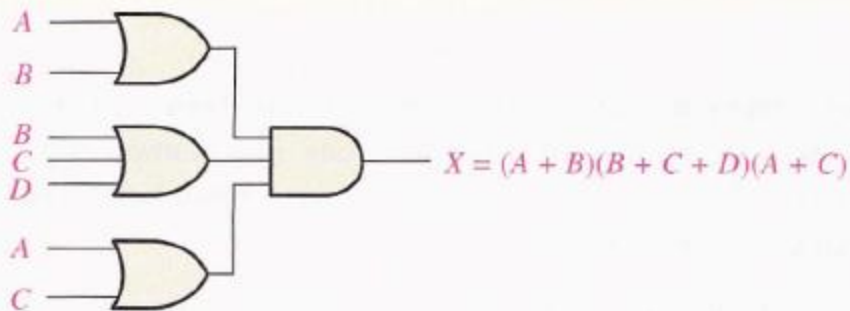
# Standard form of Boolean Algebra

- The POS : when two or more summed terms are multiplied.

- Example :

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

- Circuit example



◀ **FIGURE 4-20**

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

# Standard form of Boolean Algebra

- The standard form of POS, in which all variables in the domain appear in each sum.
- Example :  $(\bar{A} + \bar{B} + \bar{C} + \bar{D})(A + \bar{B} + C + D)(A + B + \bar{C} + D)$
- It is important in constructing truth tables, and in Karnaugh map simplification.

# Standard form of Boolean Algebra

$$(A + 0) \quad (A + \underline{0}) \quad (B + \underline{0}) \quad \textcircled{C \cdot \bar{C} = 0}$$

- Converting Boolean expression to the standard form of POS:

*Converting a Sum Term to Standard POS* Each sum term in a POS expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their complements. As stated in the following steps, a non-standard POS expression is converted into standard form using Boolean algebra rule 8 ( $A \cdot \bar{A} = 0$ ) from Table 4-1: A variable multiplied by its complement equals 0.

**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 rule 12 from Table 4-1.  $A + BC = (A + B)(A + C)$

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

# Standard form of Boolean Algebra

## EXAMPLE 4-15

$$\begin{array}{l}
 A \\
 B \\
 C \\
 D
 \end{array}
 \begin{array}{l}
 A + B + C \\
 = (A + B)(A + C)
 \end{array}$$

$$\begin{array}{l}
 x + D\bar{D} \\
 = (x + D)(x + \bar{D}) \\
 = (A + \bar{B} + C + D)(\dots + \bar{D})
 \end{array}$$

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:

$$A + \bar{B} + C = A + \bar{B} + C + \underbrace{D\bar{D}}_0 = (A + \bar{B} + C + D)(A + \bar{B} + C + \bar{D})$$

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:

$$\bar{B} + C + \bar{D} = \bar{B} + C + \bar{D} + \underbrace{A\bar{A}}_0 = (A + \bar{B} + C + \bar{D})(\bar{A} + \bar{B} + C + \bar{D})$$

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:

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

# Standard form of Boolean Algebra

- Binary representation for SOP

SOP  
A → 0  
Ā → 1

Binary Representation of a Standard Sum Term A standard sum term is equal to 0 for only one combination of variable values. For example, the sum term  $A + \bar{B} + C + \bar{D}$  is 0 when  $A = 0, B = 1, C = 0,$  and  $D = 1$ , as shown below, and is 1 for all other combinations of values for the variables.

$$A + \bar{B} + C + \bar{D} = 0 + \bar{1} + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$$

0 1 0 1 = 0 (0101)

In this case, the sum term has a binary value of 0101 (decimal 5). Remember, a sum term is implemented with an OR gate whose output is 0 only if each of its inputs is 0. Inverters are used to produce the complements of the variables as required.

**A POS expression is equal to 0 only if one or more of the sum terms in the expression is equal to 0.**



# Standard form of Boolean Algebra

- Binary representation for POS

$$A B \bar{C} + \dots$$

$$1 1 0$$

## EXAMPLE 4-16

Determine the binary values of the variables for which the following standard POS expression is equal to 0:

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

**Solution** The term  $A + B + C + D$  is equal to 0 when  $A = 0, B = 0, C = 0,$  and  $D = 0$ .

$$A + B + C + D \quad (0 \ 0 \ 0 \ 0)$$

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

The term  $A + \bar{B} + \bar{C} + D$  is equal to 0 when  $A = 0, B = 1, C = 1,$  and  $D = 0$ .

$$\begin{matrix} (0 & 1 & 1 & 0) \\ A & \bar{B} & \bar{C} & D \end{matrix} \rightarrow$$

$$A + \bar{B} + \bar{C} + D = 0 + \bar{1} + \bar{1} + 0 = 0 + 0 + 0 + 0 = 0$$

The term  $\bar{A} + \bar{B} + \bar{C} + \bar{D}$  is equal to 0 when  $A = 1, B = 1, C = 1,$  and  $D = 1$ .

$$\begin{matrix} 1 & 1 & 1 & 1 \\ \bar{A} & \bar{B} & \bar{C} & \bar{D} \end{matrix}$$

$$\bar{A} + \bar{B} + \bar{C} + \bar{D} = \bar{1} + \bar{1} + \bar{1} + \bar{1} = 0 + 0 + 0 + 0 = 0$$

The POS expression equals 0 when any of the three sum terms equals 0.

# Standard form of Boolean Algebra

- Converting standard SOP to standard POS

**Step 1.** Evaluate each product term in the SOP expression. That is, determine the binary numbers that represent the product terms.

**Step 2.** Determine all of the binary numbers not included in the evaluation in Step 1.

**Step 3.** Write the equivalent sum term for each binary number from Step 2 and express in POS form.

Using a similar procedure, you can go from POS to SOP.

# Standard form of Boolean Algebra

- Converting standard SOP to standard POS

SOP  $A \rightarrow 1$   
 $\bar{A} \rightarrow 0$   
 POS  $A \rightarrow 0$   
 $\bar{A} \rightarrow 1$

## EXAMPLE 4-17

Convert the following SOP expression to an equivalent POS expression:

$$\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = 1$$

8 options

**Solution** The evaluation is as follows:

$$000 + 010 + 011 + 101 + 111$$

Since there are three variables in the domain of this expression, there are a total of eight ( $2^3$ ) possible combinations. The SOP expression contains five of these combinations, so the POS must contain the other three which are 001, 100, and 110.

= 0

Remember, these are the binary values that make the sum term 0. The equivalent POS expression is

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

$$(101)(100)(110)$$

0 0 1 = 0  
 $(A + B + \bar{C}) = 0$   
 1 0 0 = 0  
 $(\bar{A} + B + C) = 0$   
 1 1 0 = 0  
 $(\bar{A} + \bar{B} + C) = 0$

# Standard form of Boolean Algebra

## SECTION 4-6 REVIEW

1. Identify each of the following expressions as SOP, standard SOP, POS, or standard POS:

*SOP (M4)*

(a)  $AB + \bar{A}BD + \bar{A}C\bar{D}$

(b)  $(A + \bar{B} + C)(A + B + \bar{C})$  //

(c)  $ABC + ABC$

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

2. Convert each SOP expression in Question 1 to standard form.

3. Convert each POS expression in Question 1 to standard form.

$b \Rightarrow 010 \quad 001 = 0$

$A = 1$   
 $\bar{A} = 0$

$011$

$100$

$101$

$110$

$111$

$= 1$

$\bar{A}BC$

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

$+ A\bar{B}C$

$+ AB\bar{C}$

$+ ABC$

//





1

$$A, B, c \Rightarrow \underbrace{A}_{B, c} ( \dots ) ( \dots )$$

$$A = A + B \bar{B} = \underline{(A + B)} \underline{(A + \bar{B})}$$

$$A + B = (A + B + c\bar{c}) = (A + B + c) (A + B + \bar{c})$$

$$A + \bar{B} = (A + \bar{B} + c\bar{c}) = (A + \bar{B} + c) (A + \bar{B} + \bar{c})$$

$$A = (A + B + c) (A + B + \bar{c}) (A + \bar{B} + c) (A + \bar{B} + \bar{c})$$

# Boolean Algebra

## CH(4)



# Standard forms of Boolean Algebra

- The two standard forms of Boolean expression :
  - Sum of Product (SOP)
  - Product of Sum (POS)
- All Boolean expressions can be written in one of the forms either SOP or POS.

# Boolean expression and truth table

- Converting SOP expression to truth table

## EXAMPLE 4-18

Develop a truth table for the standard SOP expression  $\overline{A}\overline{B}C + A\overline{B}\overline{C} + ABC$ .

### Solution

There are three variables in the domain, so there are eight possible combinations of binary values of the variables as listed in the left three columns of Table 4-6. The binary values that make the product terms in the expressions equal to 1 are  $\overline{A}\overline{B}C$ : 001;  $A\overline{B}\overline{C}$ : 100; and  $ABC$ : 111. For each of these binary values, place a 1 in the output column as shown in the table. For each of the remaining binary combinations, place a 0 in the output column.

TABLE 4-6

| INPUTS |   |   | OUTPUT | PRODUCT TERM                |
|--------|---|---|--------|-----------------------------|
| A      | B | C | X      |                             |
| 0      | 0 | 0 | 0      |                             |
| 0      | 0 | 1 | 1      | $\overline{A}\overline{B}C$ |
| 0      | 1 | 0 | 0      |                             |
| 0      | 1 | 1 | 0      |                             |
| 1      | 0 | 0 | 1      | $A\overline{B}\overline{C}$ |
| 1      | 0 | 1 | 0      |                             |
| 1      | 1 | 0 | 0      |                             |
| 1      | 1 | 1 | 1      | $ABC$                       |

$\overline{A}\overline{B}C$   
001  
 $A\overline{B}\overline{C}$   
100  
 $ABC$   
111

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100

$2^3 = 8$  options  
 $(0-7)_{10} = 7_{10}$   
 $max = 2^3 - 1 = 2^3 - 1 = 7_{10}$



# Boolean expression and truth table

- Converting POS expression to truth table

## EXAMPLE 4-19

Determine the truth table for the following standard POS expression:

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

*(Handwritten annotations: wavy lines under each sum term with a '0' below them, and a '- 0' at the end of the expression)*

**Solution** There are three variables in the domain and the eight possible binary values are listed in the left three columns of Table 4-7. The binary values that make the sum terms in the expression equal to 0 are  $A + B + C$ : 000;  $A + \bar{B} + C$ : 010;  $A + \bar{B} + \bar{C}$ : 011;  $\bar{A} + B + \bar{C}$ : 101; and  $\bar{A} + \bar{B} + C$ : 110. For each of these binary values, place a 0 in the output column as shown in the table. For each of the remaining binary combinations, place a 1 in the output column.

*(Handwritten annotations: A → 1, Ā → 0)*

# Boolean expression and truth table

- Converting POS expression to truth table

▶ TABLE 4-7

| INPUTS |   |   | OUTPUT | SUM TERM                  |
|--------|---|---|--------|---------------------------|
| A      | B | C | X      |                           |
| 0      | 0 | 0 | 0      | $(A + B + C)$             |
| 0      | 0 | 1 | 1      |                           |
| 0      | 1 | 0 | 0      | $(A + \bar{B} + C)$       |
| 0      | 1 | 1 | 0      | $(A + \bar{B} + \bar{C})$ |
| 1      | 0 | 0 | 1      |                           |
| 1      | 0 | 1 | 0      | $(\bar{A} + B + \bar{C})$ |
| 1      | 1 | 0 | 0      | $(\bar{A} + \bar{B} + C)$ |
| 1      | 1 | 1 | 1      |                           |

Notice that the truth table in this example is the same as the one in Example 4-18. This means that the SOP expression in the previous example and the POS expression in this example are equivalent.

# Boolean expression and truth table

- Determining truth table from Boolean expression

## EXAMPLE 4-20

From the truth table in Table 4-8, determine the standard SOP expression and the equivalent standard POS expression.

SOP

$$\bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$$

| INPUTS |   |   | OUTPUT |
|--------|---|---|--------|
| A      | B | C | X      |
| 0      | 0 | 0 | 0      |
| 0      | 0 | 1 | 0      |
| 0      | 1 | 0 | 0      |
| 0      | 1 | 1 | 1      |
| 1      | 0 | 0 | 1      |
| 1      | 0 | 1 | 0      |
| 1      | 1 | 0 | 1      |
| 1      | 1 | 1 | 1      |

POS

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

# Boolean expression and truth table

- Determining truth table from Boolean expression

**Solution** There are four 1s in the output column and the corresponding binary values are 011, 100, 110, and 111. Convert these binary values to product terms as follows:

$$011 \longrightarrow \bar{A}BC$$

$$100 \longrightarrow A\bar{B}\bar{C}$$

$$110 \longrightarrow AB\bar{C}$$

$$111 \longrightarrow ABC$$

The resulting standard SOP expression for the output  $X$  is

$$X = \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$$

For the POS expression, the output is 0 for binary values 000, 001, 010, and 101. Convert these binary values to sum terms as follows:

$$\underline{000} \longrightarrow A + B + C$$

$$001 \longrightarrow A + B + \bar{C}$$

$$010 \longrightarrow A + \bar{B} + C$$

$$101 \longrightarrow \bar{A} + B + \bar{C}$$

The resulting standard POS expression for the output  $X$  is

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



| INPUTS |   |   | O |
|--------|---|---|---|
| A      | B | C |   |
| 0      | 0 | 0 |   |
| 0      | 0 | 1 |   |
| 0      | 1 | 0 |   |
| 0      | 1 | 1 |   |
| 1      | 0 | 0 |   |
| 1      | 0 | 1 |   |
| 1      | 1 | 0 |   |
| 1      | 1 | 1 |   |

**Problem** By substitution of binary values, show that the SOP and the POS expressions derived

# Boolean expression and truth table

## SECTION 4-7 REVIEW

min no = 0  
max no = 31

1. If a certain Boolean expression has a domain of five variables, how many binary values will be in its truth table?  $A, B, C, D, E$   $2^5 = 32$
2. In a certain truth table, the output is a 1 for the binary value 0110. Convert this binary value to the corresponding product term using variables  $W, X, Y,$  and  $Z$ .
3. In a certain truth table, the output is a 0 for the binary value 1100. Convert this binary value to the corresponding sum term using variables  $W, X, Y,$  and  $Z$ .

MSB  $W$   $X$   $Y$   $Z$  LSB  
0 1 1 0  
 $\bar{W} + \bar{X} + Y + Z$

MSB LSB

$W$   $X$   $Y$   $Z$   
1 1 0 0  
 $\bar{W} + \bar{X} + Y + Z$



# Boolean Algebra

## CH(4)

# The Karnugh Map

- Provide systematic method to find simplifying Boolean expression
- Produce the simplest SOP or POS expression. Known as the minimum expression.
- By the end of this section you should be able to :
  - Construct the Karnugh map
  - Determine the binary value for each cell of the map
  - Determine the standard product term for each cell in the map
  - Explain cell adjacency and identify adjacent cell.

# The Karnugh Map

- Is an array of cells:

|        |        |
|--------|--------|
| Cell 1 | Cell 2 |
| Cell 3 | Cell 4 |

*No. of cell = 2*  
*2 input  $\rightarrow 2^2 = 4$  cells*  
*3 inputs  $\rightarrow 2^3 = 8$  cells*  
*4 inputs  $\rightarrow 2^4 = 16$  cells*  
*5 inputs  $\rightarrow 2^5 = 32$  cells*

- Each cell represent a binary value of the input.

Example for two inputs A,B:

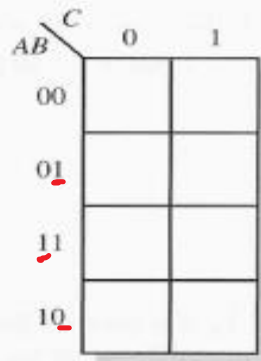
|   |       |                      |                |
|---|-------|----------------------|----------------|
|   | A \ B | 0                    | 1              |
| 0 |       | 00 $\bar{A} \bar{B}$ | 01 $\bar{A} B$ |
| 1 |       | 10 $A \bar{B}$       | 11 $A B$       |

# The Karnugh Map

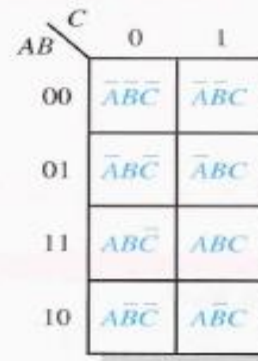
- The Karnugh For three Inputs:  $2^3 = 8$  cells
- There are eight cells with eight different binary representation

► **FIGURE 4-21**

A 3-variable Karnaugh map showing product terms.



(a)



(b)

# The Karnugh Map

- The Karnugh for four Inputs:
- There are 16 cells with 16 different binary represe

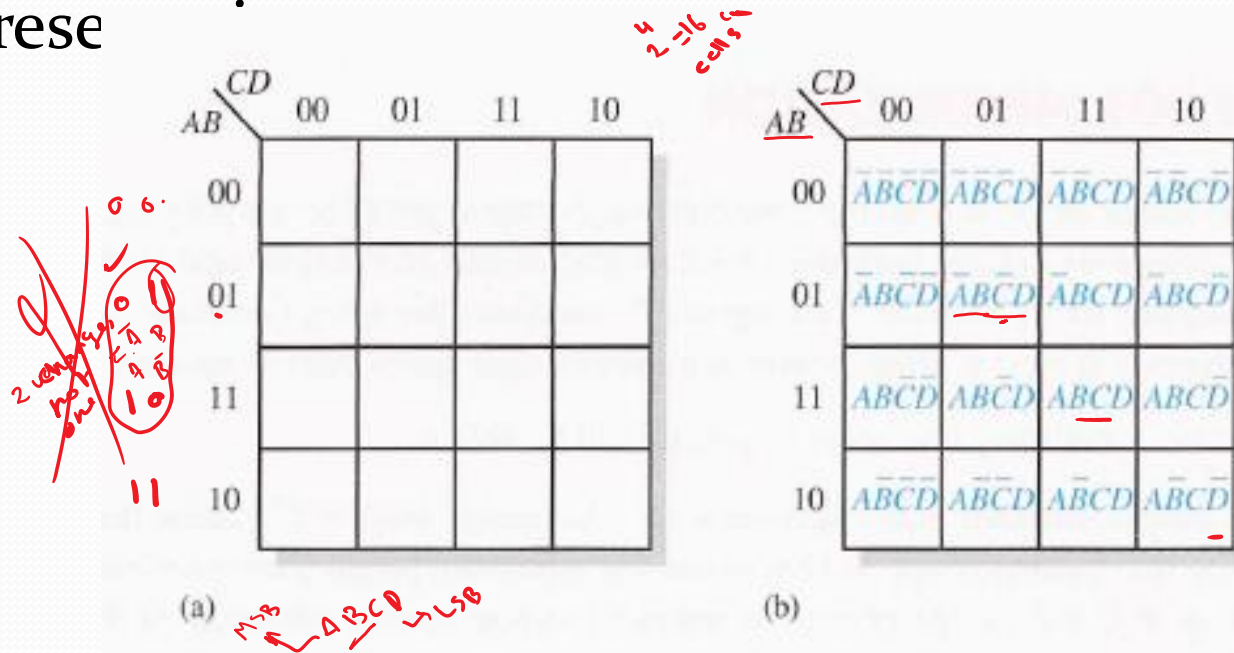
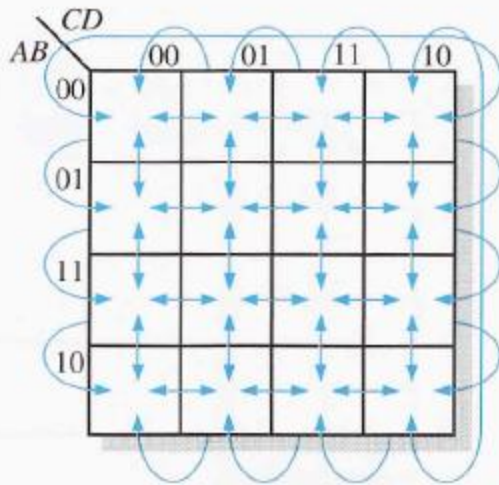


FIGURE 4-22  
A 4-variable Karnaugh

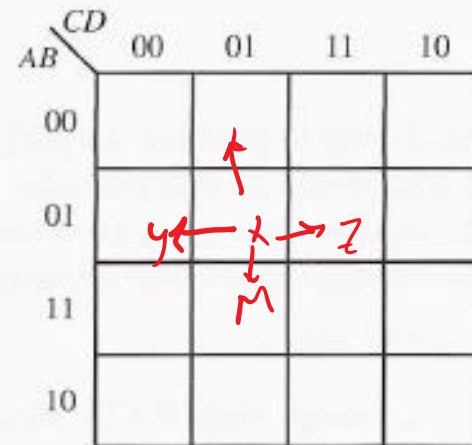
# The Karnugh Map

- Cell adjacency : is defined as single-variable change



◀ **FIGURE 4**

Adjacent cells are those that vary by one variable. Arrow adjacent cells.



# The Karnugh Map

- Karnugh map SOP minimization

A minimized SOP expression contains the least number of terms and least number of variable per terms.

By the end of this section you should be able to:

- Map a standard SOP expression on a Karnugh map.
- Combine the one's cell into max group
- Combine the min product terms to form the min SOP expression .

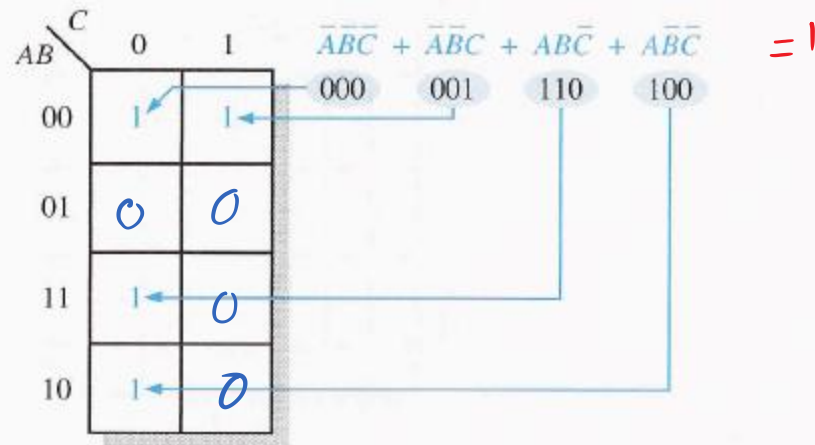
# The Karnaugh Map

- Mapping a standard SOP expression

- Step 1.** Determine the binary value of each product term in the standard SOP expression. After some practice, you can usually do the evaluation of terms mentally.
- Step 2.** As each product term is evaluated, place a 1 on the Karnaugh map in the cell having the same value as the product term.

► **FIGURE 4-24**

Example of mapping a standard SOP expression.





# The Karnaugh Map

- Mapping a standard SOP expression

## EXAMPLE 4-21

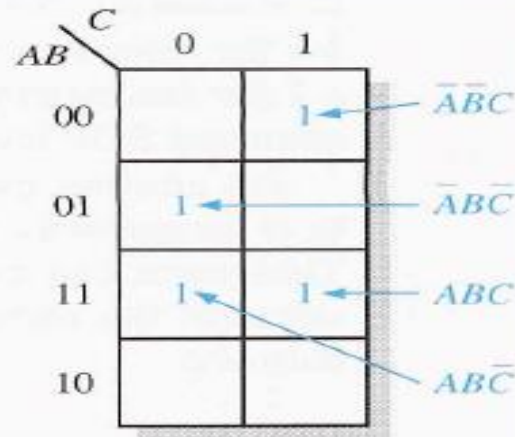
Map the following standard SOP expression on a Karnaugh map:

$$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

**Solution** Evaluate the expression as shown below. Place a 1 on the 3-variable Karnaugh map in Figure 4-25 for each standard product term in the expression.

$$\begin{array}{cccc} \bar{A}\bar{B}C & + & \bar{A}B\bar{C} & + & A\bar{B}\bar{C} & + & ABC \\ 001 & & 010 & & 110 & & 111 \end{array}$$

► FIGURE 4-25



# The Karnaugh Map

- Mapping a standard SOP expression

## EXAMPLE 4-22

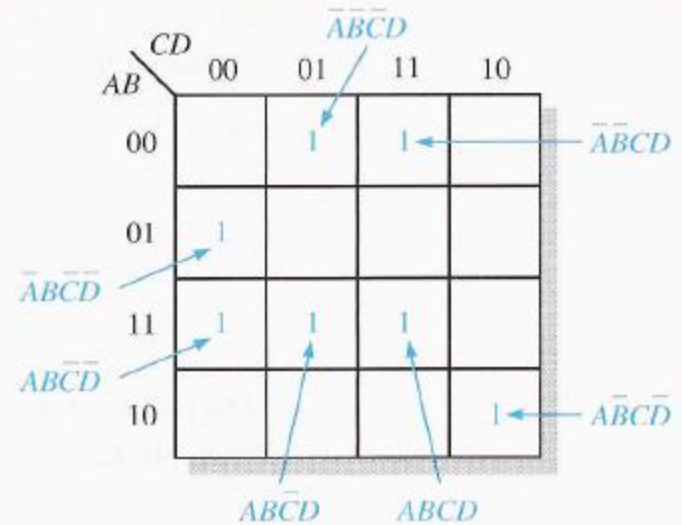
Map the following standard SOP expression on a Karnaugh map:

$$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}D + ABCD + A\bar{B}C\bar{D} + \bar{A}B\bar{C}D + A\bar{B}C\bar{D}$$

**Solution** Evaluate the expression as shown below. Place a 1 on the 4-variable Karnaugh map in Figure 4-26 for each standard product term in the expression.

$$\begin{array}{ccccccc} \bar{A}\bar{B}CD & \bar{A}B\bar{C}\bar{D} & A\bar{B}\bar{C}D & ABCD & A\bar{B}C\bar{D} & \bar{A}B\bar{C}D & A\bar{B}C\bar{D} \\ 0011 & 0100 & 1101 & 1111 & 1100 & 0001 & 1010 \end{array}$$

► FIGURE 4-26



# The Karnaugh Map

- Mapping non standard SOP expression

## EXAMPLE 4-23

Map the following SOP expression on a Karnaugh map:  $\bar{A} + \bar{A}\bar{B} + ABC\bar{C}$ .

### Solution

The SOP expression is obviously not in standard form because each product term does not have three variables. The first term is missing two variables, the second term is missing one variable, and the third term is standard. First expand the terms numerically as follows:

$$\begin{array}{l} \bar{A} + \bar{A}\bar{B} + ABC\bar{C} \\ \hline 000 \quad 100 \quad 110 \\ \hline 001 \quad 101 \\ \hline 010 \\ \hline 011 \end{array}$$

Handwritten notes: A circled list of binary values:  $\bar{A}\bar{B}\bar{C}$ ,  $\bar{A}\bar{B}C$ ,  $A\bar{B}\bar{C}$ ,  $A\bar{B}C$ . To the right, a handwritten expansion:  $A\bar{B}C + A\bar{B}\bar{C}$  with a circled  $A\bar{B}$  and  $(C + \bar{C})$  below it, with a '1' underneath the parentheses.

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map in Figure 4-27.

► FIGURE 4-27

|    |    |   |   |
|----|----|---|---|
|    |    | C |   |
|    |    | 0 | 1 |
| AB | 00 | 1 | 1 |
|    | 01 | 1 | 1 |
|    |    |   |   |

# The Karnaugh Map

- Mapping non standard SOP expression

## EXAMPLE 4-24

Map the following SOP expression on a Karnaugh map:

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

*Solution*

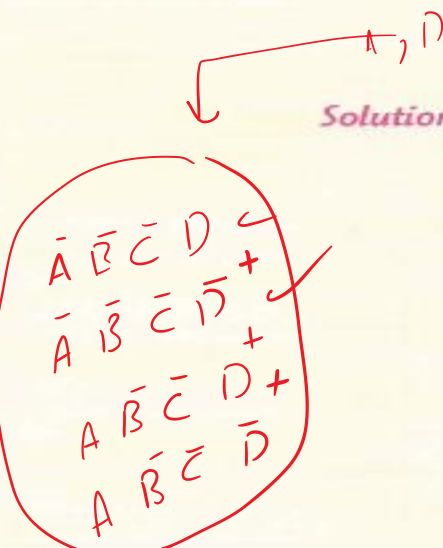
The SOP expression is obviously not in standard form because each product term does not have four variables. The first and second terms are both missing two variables, the third term is missing one variable, and the rest of the terms are standard. First expand the terms by including all combinations of the missing variables numerically as follows:

$$\begin{array}{cccccc} \overline{B}\overline{C} & \overline{A}\overline{B} & + & \overline{A}B\overline{C} & + & \overline{A}\overline{B}C\overline{D} & + & \overline{A}B\overline{C}D & + & \overline{A}B\overline{C}D \\ 0000 & 1000 & & 1100 & & 1010 & & 0001 & & 1011 \\ 0001 & 1001 & & 1101 & & & & & & \\ 1000 & 1010 & & & & & & & & \\ 1001 & 1011 & & & & & & & & \end{array}$$

Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4-variable Karnaugh map in Figure 4-28. Notice that some of the values in the expanded expression are redundant.

► FIGURE 4-28

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | 1  | 1  |    |    |
| 01      |    |    |    |    |
| 11      | 1  | 1  |    |    |



# The Karnugh Map

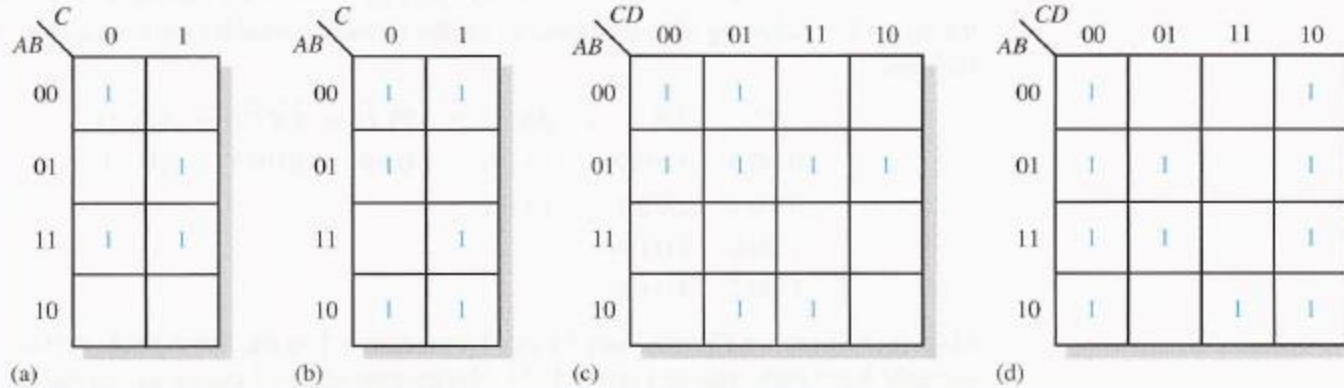
- Karnugh map simplification of SOP expression

*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.

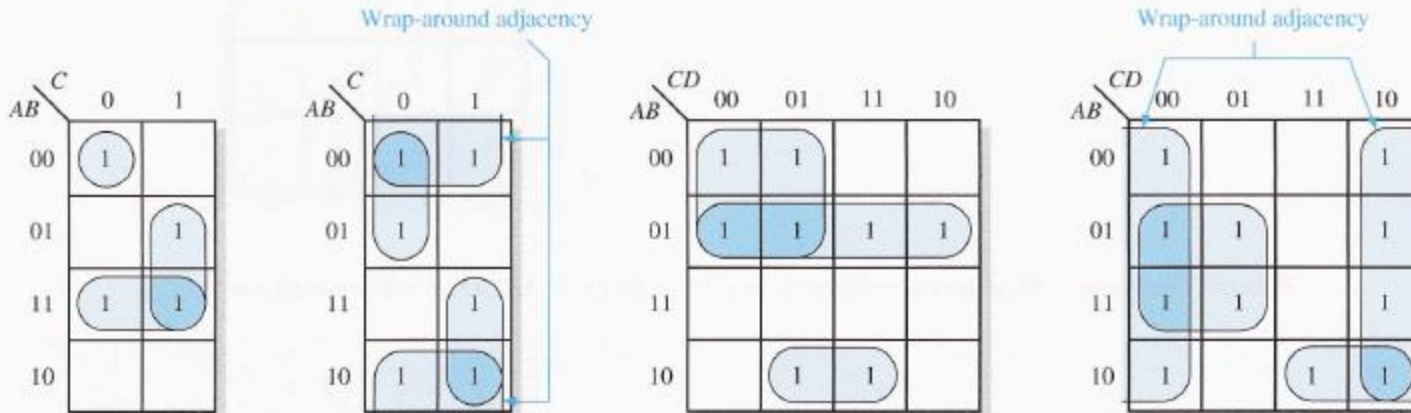
### EXAMPLE 4-25

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.



# The Karnaugh Map

- Karnaugh map simplification of SOP expression
- Determining the min SOP expression

1. Group the cells that have 1s. Each group of cells containing 1s creates one product term composed of all variables that occur in only one form (either uncomplemented or complemented) within the group. Variables that occur both uncomplemented and complemented within the group are eliminated. These are called *contradictory variables*.
2. Determine the minimum product term for each group.
  - a. For a 3-variable map:
    - (1) A 1-cell group yields a 3-variable product term
    - (2) A 2-cell group yields a 2-variable product term
    - (3) A 4-cell group yields a 1-variable term
    - (4) An 8-cell group yields a value of 1 for the expression
  - b. For a 4-variable map:
    - (1) A 1-cell group yields a 4-variable product term
    - (2) A 2-cell group yields a 3-variable product term
    - (3) A 4-cell group yields a 2-variable product term
    - (4) An 8-cell group yields a 1-variable term
    - (5) A 16-cell group yields a value of 1 for the expression
3. When all the minimum product terms are derived from the Karnaugh map, they are summed to form the minimum SOP expression.

3 inputs

1 cell    2    4    8

4 input

1    2    4    8    16

# The Karnaugh Map

## EXAMPLE 4-26

Determine the product terms for the Karnaugh map in Figure 4-31 and write the resulting minimum SOP expression.

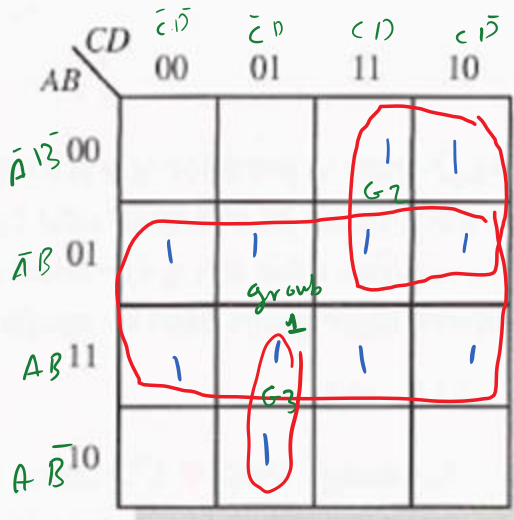
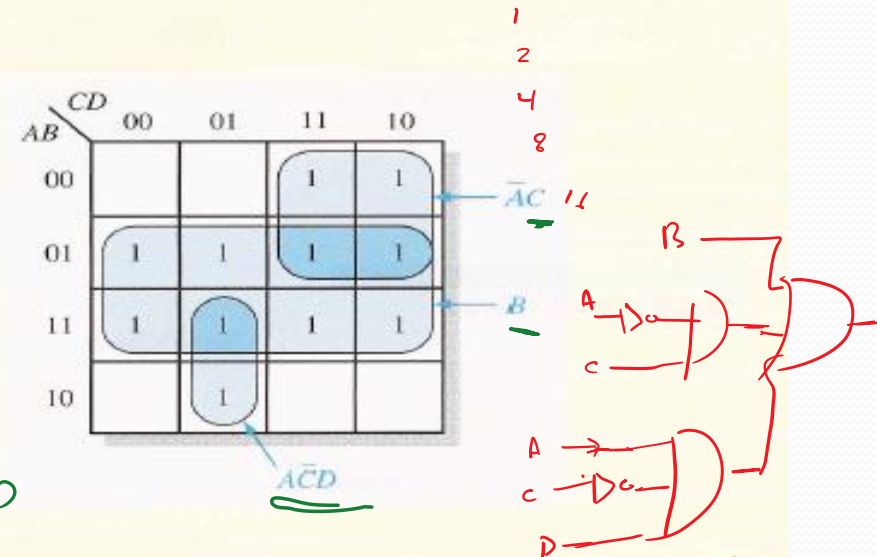


FIGURE 4-31

For group 1  $\Rightarrow B$   
 $A, \bar{A} \Rightarrow$  eliminate  
 $c, \bar{c} \Rightarrow$  eliminate  
 $D, \bar{D} \Rightarrow "$

Group 2  $\Rightarrow \bar{A}C$   
 $B, \bar{B} \Rightarrow$  eliminate  $B$   
 $D, \bar{D} \Rightarrow$  eliminate  $D$



For Group 3

$B, \bar{B} \rightarrow$  eliminate  $B$

### Solution

Eliminate variables that are in a grouping in both complemented and uncomplemented forms. In Figure 4-31, the product term for the 8-cell group is  $B$  because the cells within that group contain both  $A$  and  $\bar{A}$ ,  $C$  and  $\bar{C}$ , and  $D$  and  $\bar{D}$ , which are eliminated. The 4-cell group contains  $B, \bar{B}, D,$  and  $\bar{D}$ , leaving the variables  $\bar{A}$  and  $C$ , which form the product term  $\bar{A}C$ . The 2-cell group contains  $B$  and  $\bar{B}$ , leaving variables  $A, C,$  and  $D$  which form the product term  $ACD$ . Notice how overlapping is used to maximize the size of the groups. The resulting minimum SOP expression is the sum of these product terms:

$$B + \bar{A}C + \bar{A}CD$$

### Related Problem

For the Karnaugh map in Figure 4-31, add a 1 in the lower right cell (1010) and



# The Karnugh Map

- Karnugh map simplification of SOP expression
- Determining the min SOP expression

3 Inputs

$\bar{A}\bar{B}\bar{C}$   
 $\bar{A}B$   
 $AB$   
 $AB$

$\bar{A}\bar{B}\bar{C} + BC + AB$

$\bar{A}\bar{C}$   
 $\bar{A}\bar{B}$   
 $\bar{A}B$   
 $AB$   
 $\bar{A}\bar{B}$

$\bar{B} + \bar{A}\bar{C} + AC$

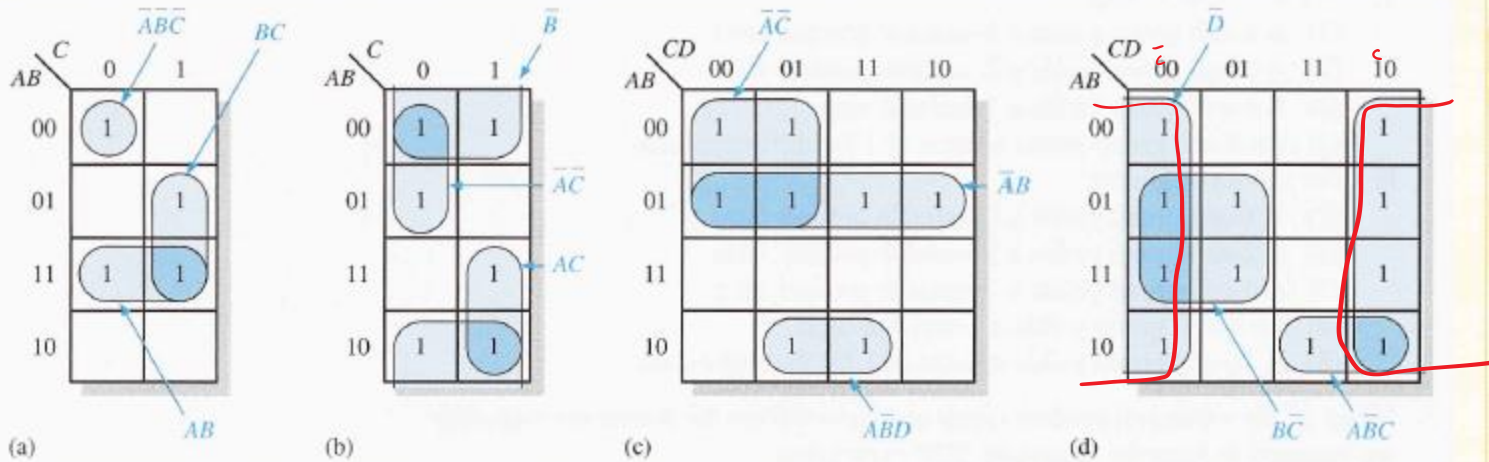
$CD$   
 $00$   $01$   $11$   $10$   
 $AB$   
 $00$   
 $01$   
 $11$   
 $10$

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

# The Karnaugh Map

## EXAMPLE 4-27

Determine the product terms for each of the Karnaugh maps in Figure 4-32 and write the resulting minimum SOP expression.



▲ FIGURE 4-32

**Solution** The resulting minimum product term for each group is shown in Figure 4-32. The minimum SOP expressions for each of the Karnaugh maps in the figure are

(a)  $AB + BC + \overline{A}\overline{B}\overline{C}$       (b)  $\overline{B} + \overline{A}\overline{C} + AC$

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

**Related Problem** For the Karnaugh map in Figure 4-32(d), add a 1 in the 0111 cell and determine the resulting SOP expression.

1, 2, 4, 8, 16  
Group!  
A, B, C, D

**EXAMPLE 4-29**

9 AND gate  
1 OR gate  
Inverters (2 Inverters)  
Solution:  $AB = \bar{D}$   
 $A\bar{B}CD$

Use a Karnaugh map to minimize the following SOP expression:

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

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

FIGURE 4-34

|         |    |    |    |    |
|---------|----|----|----|----|
| CD \ AB | 00 | 01 | 11 | 10 |
| 00      |    |    |    |    |
| 01      |    |    |    |    |
| 11      |    |    |    |    |
| 10      |    |    |    |    |

A, B, C

|         |    |    |    |    |
|---------|----|----|----|----|
| CD \ AB | 00 | 01 | 11 | 10 |
| 00      | 1  |    | 1  | 1  |
| 01      | 1  |    |    | 1  |
| 11      | 1  |    |    | 1  |
| 10      | 1  |    | 1  | 1  |

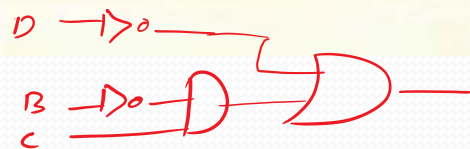
$0 \rightarrow n = \Rightarrow$  NO of variables  
2

- 1
- 2
- 4
- 8
- 16

A, B, C, D

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

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



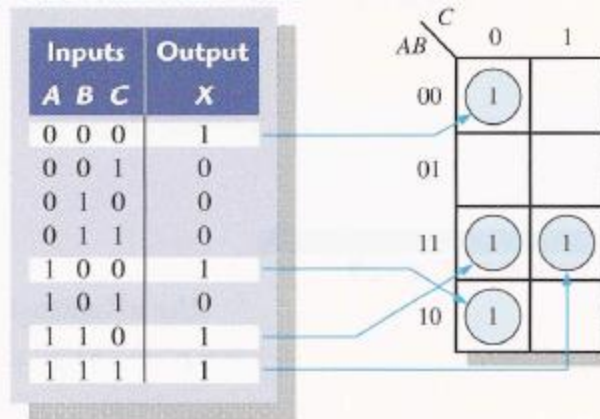
# The Karnaugh Map

- Mapping directly from the truth table to Karnaugh map

► **FIGURE 4-35**

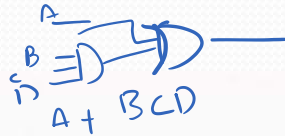
Example of mapping directly from a truth table to a Karnaugh map.

$$X = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}\overline{C} + ABC$$



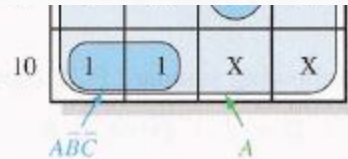
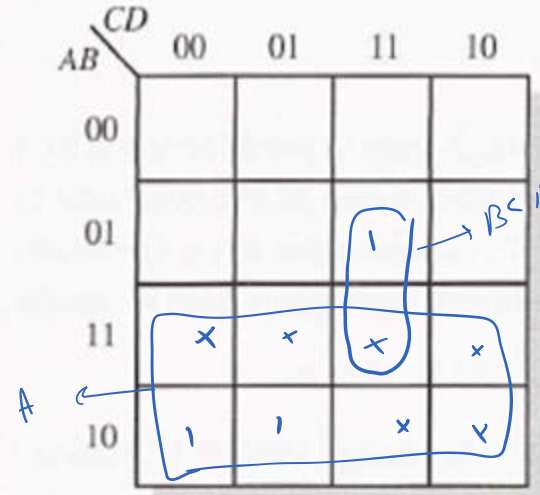
# The Karnugh Map

- Don't care condition



| Inputs<br>ABCD | Output<br>Y |
|----------------|-------------|
| 0 0 0 0        | 0           |
| 0 0 0 1        | 0           |
| 0 0 1 0        | 0           |
| 0 0 1 1        | 0           |
| 0 1 0 0        | 0           |
| 0 1 0 1        | 0           |
| 0 1 1 0        | 0           |
| 0 1 1 1        | 1           |
| 1 0 0 0        | 1           |
| 1 0 0 1        | 1           |
| 1 0 1 0        | X           |
| 1 0 1 1        | X           |
| 1 1 0 0        | X           |
| 1 1 0 1        | X           |
| 1 1 1 0        | X           |
| 1 1 1 1        | X           |

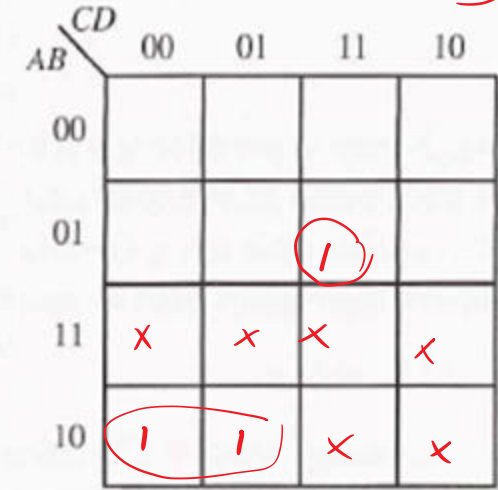
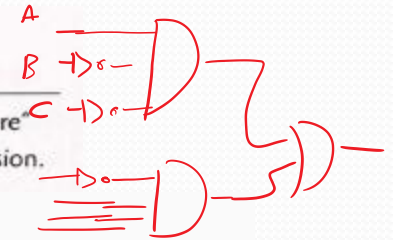
(a) Truth table



(b) Without "don't cares"  $Y = A\bar{B}\bar{C} + \bar{A}BCD$   
 With "don't cares"  $Y = A + BCD$

FIGURE 4-36

Example of the use of "don't care" conditions to simplify an expression.



$A\bar{B}\bar{C}$

$A\bar{B}\bar{C} + \bar{A}BCD$

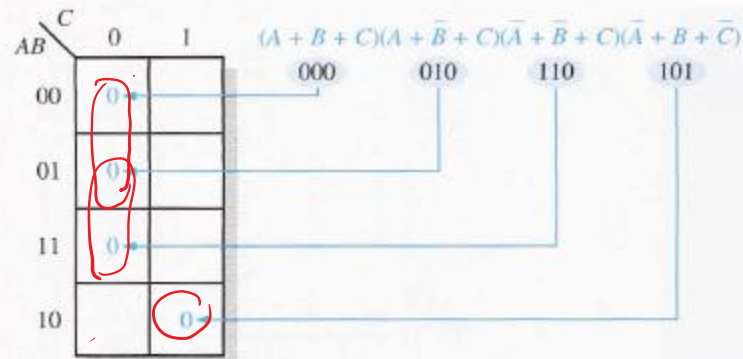
# The Karnugh Map

- Karnugh map simplification of POS expression

- Step 1.** Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.
- Step 2.** As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.

► **FIGURE 4-37**

Example of mapping a standard POS expression.



Handwritten notes in red ink:

2<sup>e</sup> 1  
 2 2  
 2 4  
 2 8

# The Karnaugh Map

- Karnaugh map simplification of POS expression

## EXAMPLE 4-30

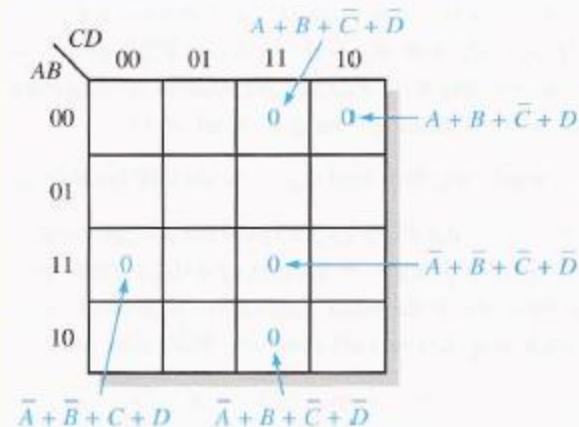
Map the following standard POS expression on a Karnaugh map:

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

**Solution** Evaluate the expression as shown below and place a 0 on the 4-variable Karnaugh map in Figure 4-38 for each standard sum term in the expression.

$$\begin{array}{cccccc}
 (\bar{A} + \bar{B} + C + D) & (\bar{A} + B + \bar{C} + \bar{D}) & (A + B + \bar{C} + D) & (\bar{A} + \bar{B} + \bar{C} + \bar{D}) & (A + B + \bar{C} + \bar{D}) & \\
 1100 & 1011 & 0010 & 1111 & 0011 & 
 \end{array}$$

► FIGURE 4-38



# The Karnugh Map

- Karnugh map simplification of POS expression

## EXAMPLE 4-30

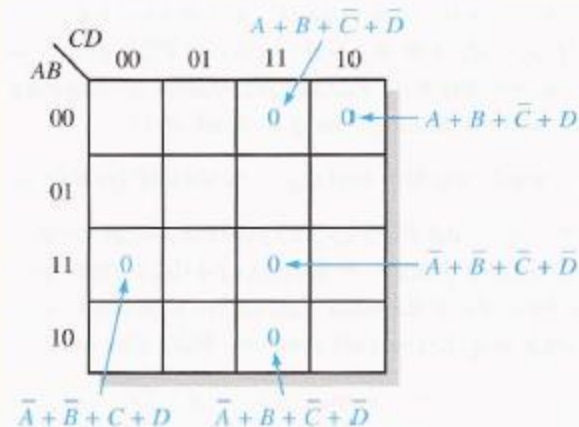
Map the following standard POS expression on a Karnaugh map:

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

**Solution** Evaluate the expression as shown below and place a 0 on the 4-variable Karnaugh map in Figure 4-38 for each standard sum term in the expression.

$$\begin{array}{cccccc}
 (\bar{A} + \bar{B} + C + D) & (\bar{A} + B + \bar{C} + \bar{D}) & (A + B + \bar{C} + D) & (\bar{A} + \bar{B} + \bar{C} + \bar{D}) & (A + B + \bar{C} + \bar{D}) & \\
 1100 & 1011 & 0010 & 1111 & 0011 & 
 \end{array}$$

► FIGURE 4-38





# The Karnugh Map

- Karnugh map simplification of POS expression

## EXAMPLE 4-31

Use a Karnugh map to minimize the following standard POS expression:

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

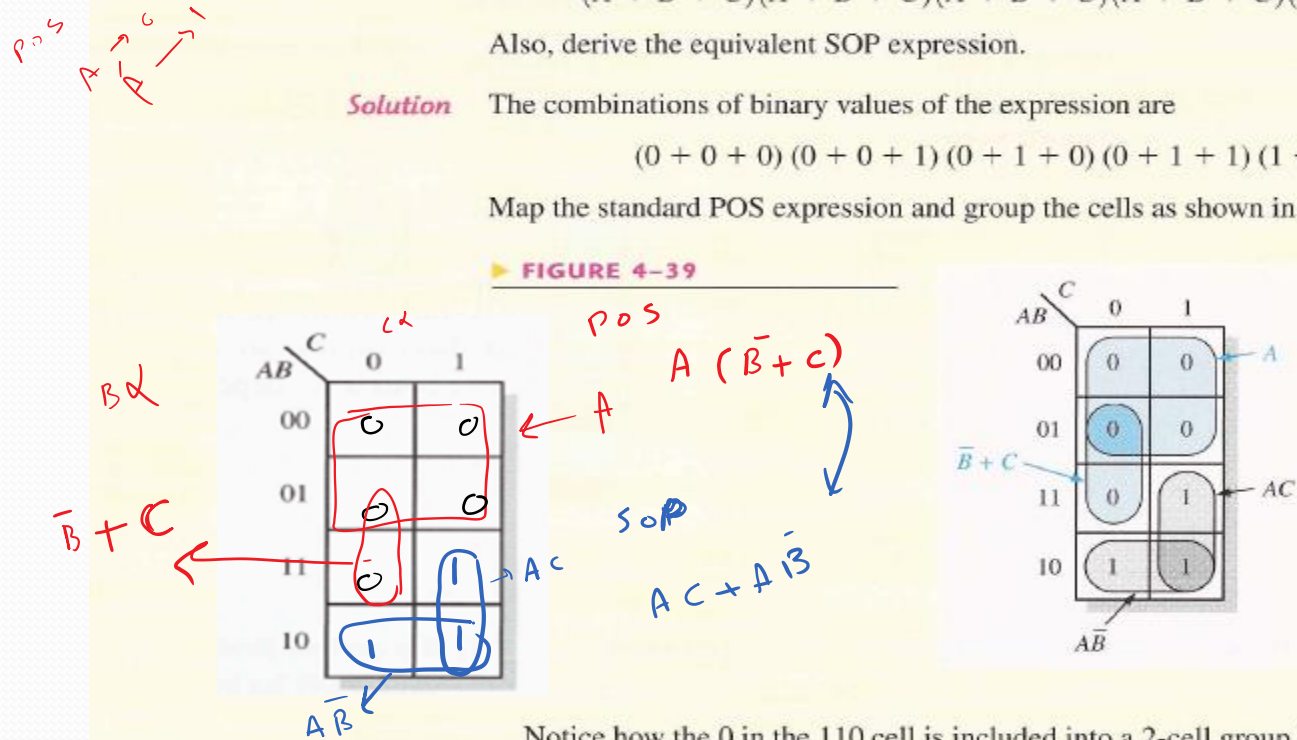
Also, derive the equivalent SOP expression.

**Solution** The combinations of binary values of the expression are

$$(0 + 0 + 0) (0 + 0 + 1) (0 + 1 + 0) (0 + 1 + 1) (1 + 1 + 0)$$

Map the standard POS expression and group the cells as shown in Figure 4-39.

► FIGURE 4-39



Notice how the 0 in the 110 cell is included into a 2-cell group by utilizing the 0 in the 4-cell group. The sum term for each blue group is shown in the figure and the resulting minimum POS expression is

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

# The Karnugh Map

- Karnugh map simplification of POS expression

## EXAMPLE 4-32

Use a Karnaugh map to minimize the following POS expression:

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

A/P

**Solution** The first term must be expanded into  $\bar{A} + B + C + D$  and  $A + B + C + D$  to get a standard POS expression, which is then mapped; and the cells are grouped as shown in

Group

1

2

4

8

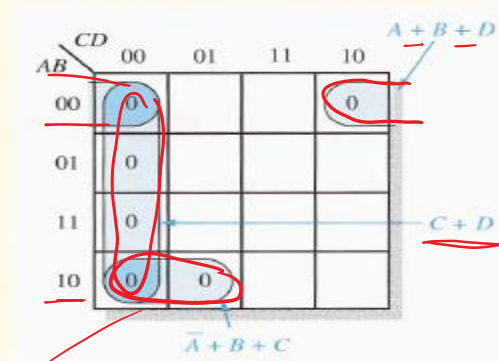
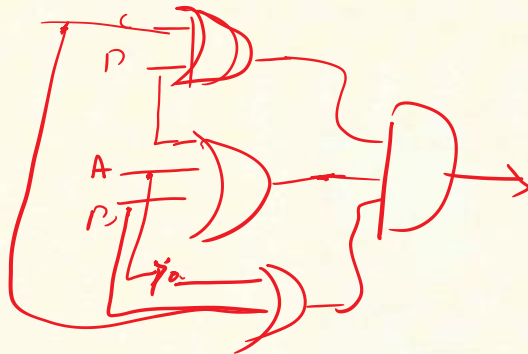
16

Figure 4-40. The sum term for each group is shown and the resulting minimum POS expression is

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

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

► FIGURE 4-40



C + D

A, B + C  
C + D

DOX

# The Karnugh Map

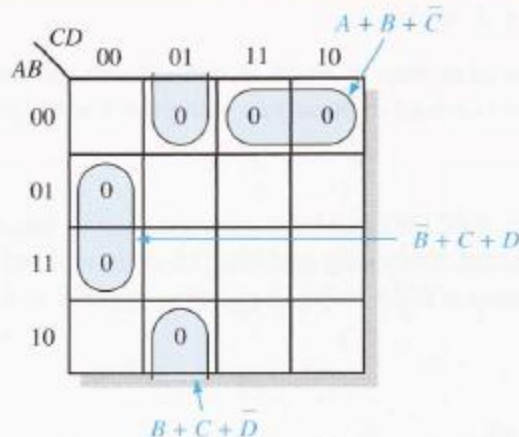
- Converting between SOP and POS using Karnugh map

## EXAMPLE 4-33

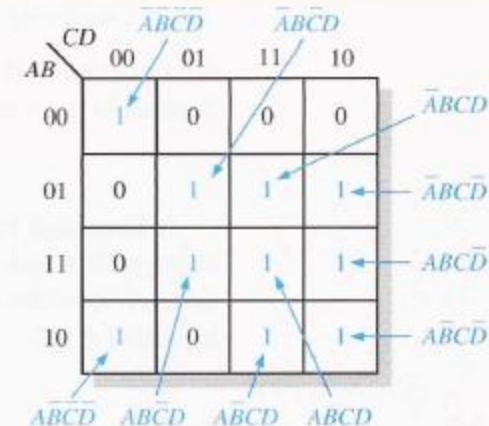
Using a Karnugh map, convert the following standard POS expression into a minimum POS expression, a standard SOP expression, and a minimum SOP expression.

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

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

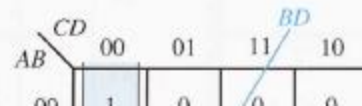


(a) Minimum POS:  $(A + B + C)(\bar{B} + \bar{C} + D)(B + C + \bar{D})$



(b) Standard SOP:

$$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}CD$$



# The Karnugh Map

- Karnugh map for 5 variables

# Boolean Algebra

# The Boolean expression

- Boolean expression of multiple variables can be written
- $F(A,B,C) = \sum(1,3,4) = 1$  *this is equivalent to SOP*  
 $= A'B'C + A'BC + AB'C'$   
 $= 001 + 011 + 100$
- Each term called **minterm**
- Or
- $F(A,B,C) = \prod(0,2,5,6,7)$  *this is equivalent to POS*  
 $= (A+B+C)(A+B'+C)(A'+B+C')(A'+B'+C)(A'+B'+C')$   
 $= (000)(010)(101)(110)(111)$
- Each term is called **maxterm**

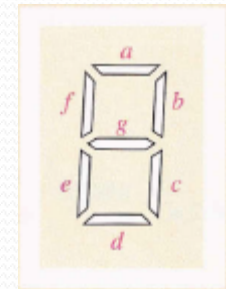
# Digital system application

ch4

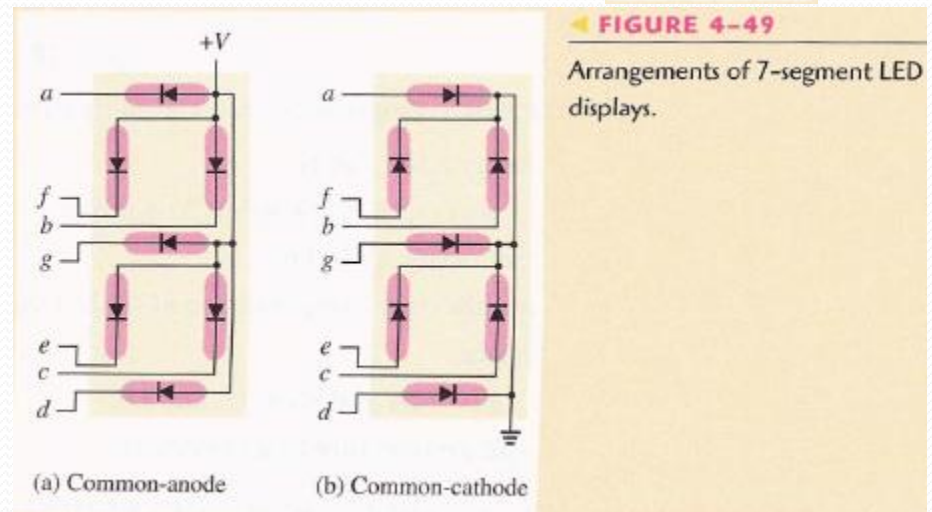
# Digital system application

## Seven segment display

- It is a display for the number, each one can display the number from 1 to 9, thus the seven segments display takes BCD as an input .



- Consists of seven LEDs
- The goal of this example is to design a logic circuit for each segment(a,b,c,d,e,f) .





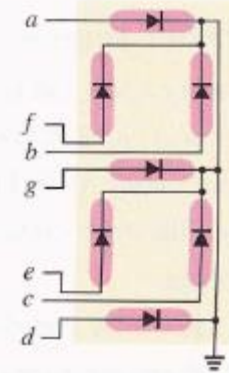
# Digital system application

## Seven segment display

► **TABLE 4-9**

Active segments for each decimal digit.

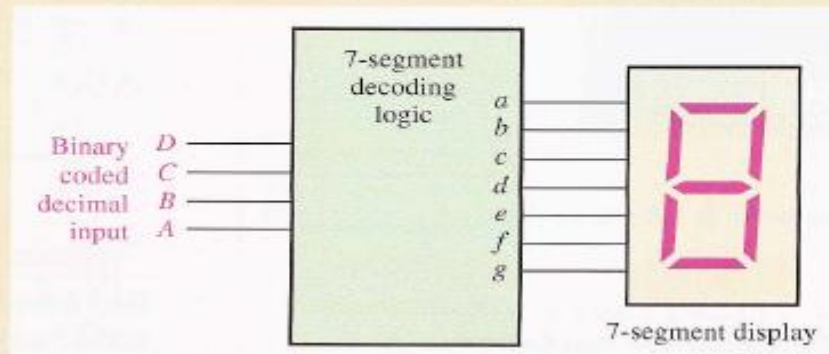
| DIGIT | SEGMENTS ACTIVATED         |
|-------|----------------------------|
| 0     | <i>a, b, c, d, e, f</i>    |
| 1     | <i>b, c</i>                |
| 2     | <i>a, b, d, e, g</i>       |
| 3     | <i>a, b, c, d, g</i>       |
| 4     | <i>b, c, f, g</i>          |
| 5     | <i>a, c, d, f, g</i>       |
| 6     | <i>a, c, d, e, f, g</i>    |
| 7     | <i>a, b, c</i>             |
| 8     | <i>a, b, c, d, e, f, g</i> |
| 9     | <i>a, b, c, d, f, g</i>    |



(b) Common-cathode

► **FIGURE 4-50**

Block diagram of 7-segment logic and display.



# Digital system application

## Seven segment display

- Design procedure

- 1- Construct the truth table for the segments.
- 2- Mapping the truth table to Karnugh map.
- 3- Find the minimized Boolean expression in the form of SOP or POS.
4. Convert the Boolean expression to digital circuit.

Note there will be for each segment output (a,b,c,d,e,f) digital circuit to convert the BCD input to the appropriate activation level for each output.

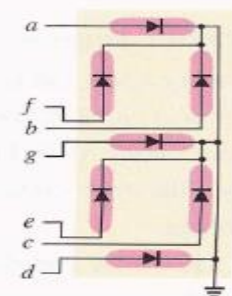
# Digital system application

## Seven segment display

- Truth table

| DECIMAL<br>DIGIT | INPUTS |   |   |   | SEGMENT OUTPUTS |   |   |   |   |   |   |
|------------------|--------|---|---|---|-----------------|---|---|---|---|---|---|
|                  | D      | C | B | A | a               | b | c | d | e | f | g |
| 0                | 0      | 0 | 0 | 0 | 1               | 1 | 1 | 1 | 1 | 1 | 0 |
| 1                | 0      | 0 | 0 | 1 | 0               | 1 | 1 | 0 | 0 | 0 | 0 |
| 2                | 0      | 0 | 1 | 0 | 1               | 1 | 0 | 1 | 1 | 0 | 1 |
| 3                | 0      | 0 | 1 | 1 | 1               | 1 | 1 | 1 | 0 | 0 | 1 |
| 4                | 0      | 1 | 0 | 0 | 0               | 1 | 1 | 0 | 0 | 1 | 1 |
| 5                | 0      | 1 | 0 | 1 | 1               | 0 | 1 | 1 | 0 | 1 | 1 |
| 6                | 0      | 1 | 1 | 0 | 1               | 0 | 1 | 1 | 1 | 1 | 1 |
| 7                | 0      | 1 | 1 | 1 | 1               | 1 | 1 | 0 | 0 | 0 | 0 |
| 8                | 1      | 0 | 0 | 0 | 1               | 1 | 1 | 1 | 1 | 1 | 1 |
| 9                | 1      | 0 | 0 | 1 | 1               | 1 | 1 | 1 | 0 | 1 | 1 |
| 10               | 1      | 0 | 1 | 0 | X               | X | X | X | X | X | X |
| 11               | 1      | 0 | 1 | 1 | X               | X | X | X | X | X | X |
| 12               | 1      | 1 | 0 | 0 | X               | X | X | X | X | X | X |
| 13               | 1      | 1 | 0 | 1 | X               | X | X | X | X | X | X |
| 14               | 1      | 1 | 1 | 0 | X               | X | X | X | X | X | X |
| 15               | 1      | 1 | 1 | 1 | X               | X | X | X | X | X | X |

| DIGIT | SEGMENTS ACTIVATED  |
|-------|---------------------|
| 0     | a, b, c, d, e, f    |
| 1     | b, c                |
| 2     | a, b, d, e, g       |
| 3     | a, b, c, d, g       |
| 4     | b, c, f, g          |
| 5     | a, c, d, f, g       |
| 6     | a, c, d, e, f, g    |
| 7     | a, b, c             |
| 8     | a, b, c, d, e, f, g |
| 9     | a, b, c, d, f, g    |



(b) Common-cathode

# Digital system application

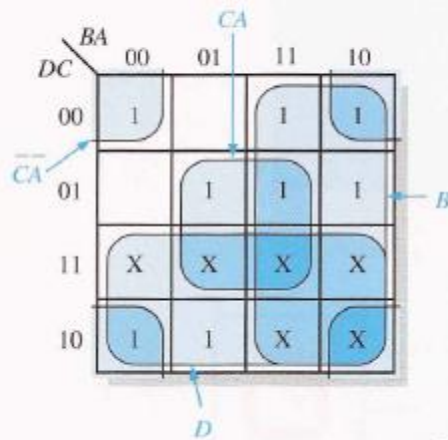
## Seven segment display

- Karnugh map

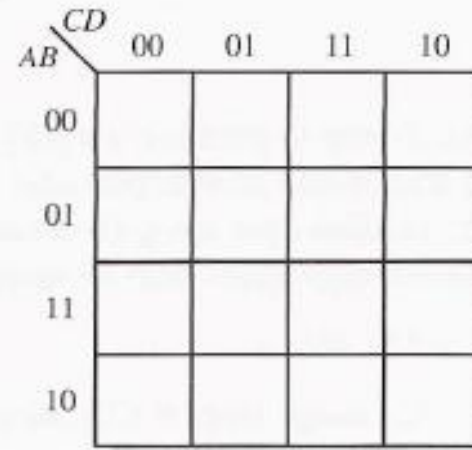
Digital circuit

Standard SOP expression:

$$\overline{DCBA} + \overline{DC}\overline{B}A + \overline{DC}B\overline{A} + \overline{DC}B\overline{A} + \overline{DC}B\overline{A} + \overline{DC}B\overline{A} + \overline{DC}B\overline{A} + \overline{DC}B\overline{A}$$



Minimum SOP expression:  $D + B + CA + \overline{CA}$

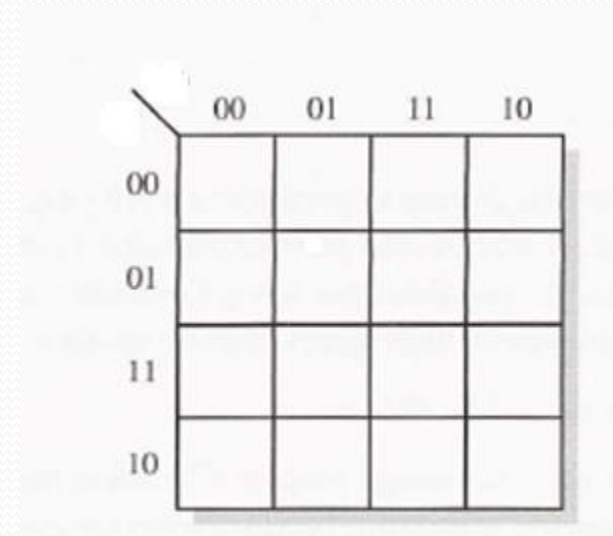


# Digital system application

## Seven segment display

- Karnugh map for output b

| INPUTS |   |   |   | S |
|--------|---|---|---|---|
| D      | C | B | A | b |
| 0      | 0 | 0 | 0 | 1 |
| 0      | 0 | 0 | 1 | 1 |
| 0      | 0 | 1 | 0 | 1 |
| 0      | 0 | 1 | 1 | 1 |
| 0      | 1 | 0 | 0 | 1 |
| 0      | 1 | 0 | 1 | 0 |
| 0      | 1 | 1 | 0 | 0 |
| 0      | 1 | 1 | 1 | 1 |
| 1      | 0 | 0 | 0 | 1 |
| 1      | 0 | 0 | 1 | 1 |
| 1      | 0 | 1 | 0 | X |
| 1      | 0 | 1 | 1 | X |
| 1      | 1 | 0 | 0 | X |
| 1      | 1 | 0 | 1 | X |
| 1      | 1 | 1 | 0 | X |
| 1      | 1 | 1 | 1 | X |



A Karnaugh map for output b. The map is a 4x4 grid with columns labeled 00, 01, 11, 10 and rows labeled 00, 01, 11, 10. The map is currently empty, with no 1s or Xs placed in the cells.

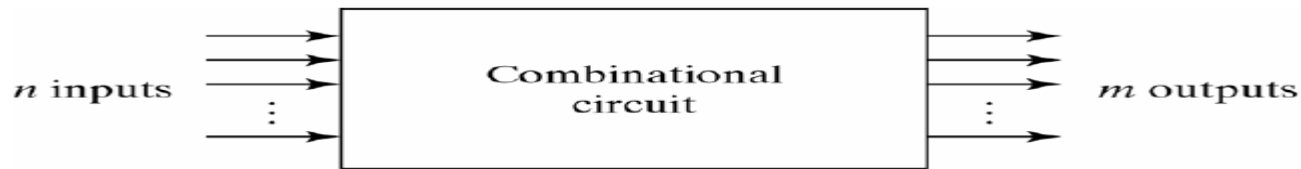
|    | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 |    |    |    |    |
| 01 |    |    |    |    |
| 11 |    |    |    |    |
| 10 |    |    |    |    |

# Logic circuit

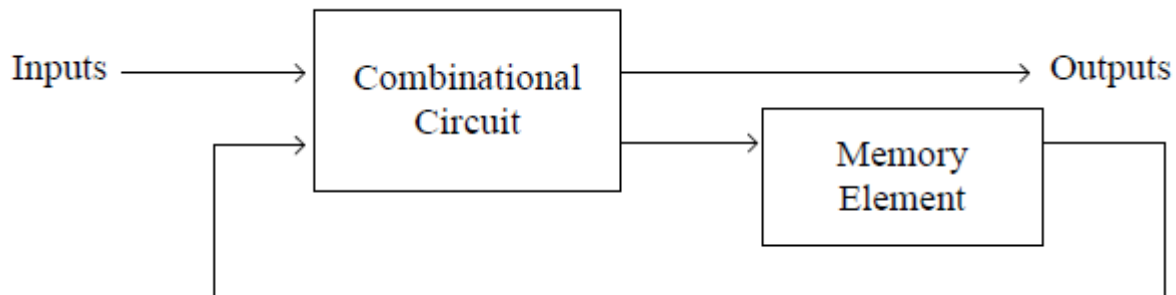
ch5

# Logic Circuit

- The combinational logic circuit : is a logical gate circuit in which the output will be presented immediately upon input present.



- The sequential logic circuit : the circuit employ memory element beside the logical gate.



# Combinational Logic Circuit

- Example of combinational logic circuit

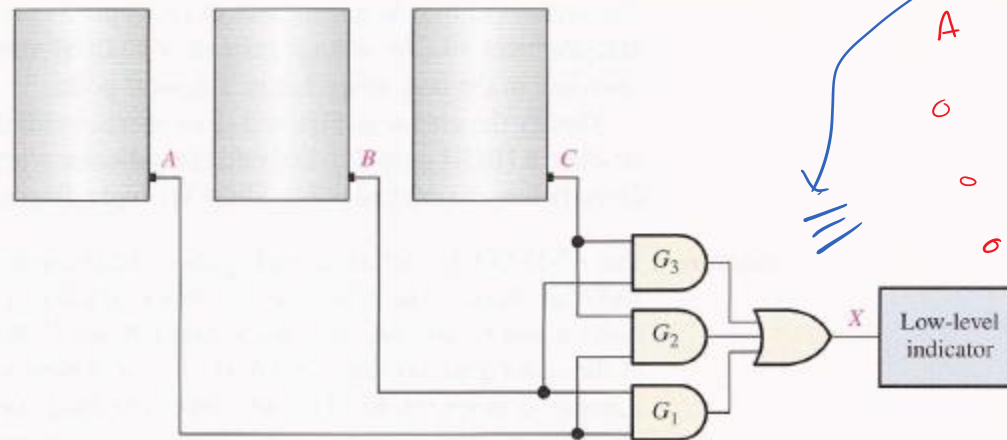
## EXAMPLE 5-1

In a certain chemical-processing plant, a liquid chemical is used in a manufacturing process. The chemical is stored in three different tanks. A level sensor in each tank produces a HIGH voltage when the level of chemical in the tank drops below a specified point.

Design a circuit that monitors the chemical level in each tank and indicates when the level in any two of the tanks drops below the specified point.

### Solution

The AND-OR circuit in Figure 5-2 has inputs from the sensors on tanks A, B, and C as shown. The AND gate  $G_1$  checks the levels in tanks A and B, gate  $G_2$  checks tanks A and C, and gate  $G_3$  checks tanks B and C. When the chemical level in any two of the tanks gets too low, one of the AND gates will have HIGHS on both of its inputs, causing its output to be HIGH; and so the final output X from the OR gate is HIGH. This HIGH input is then used to activate an indicator such as a lamp or audible alarm, as shown in the figure.



$$X = A'B'C + AB'C + A'BC + ABC$$

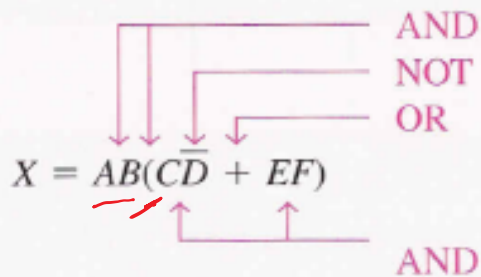
| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



# Combinational Logic Circuit

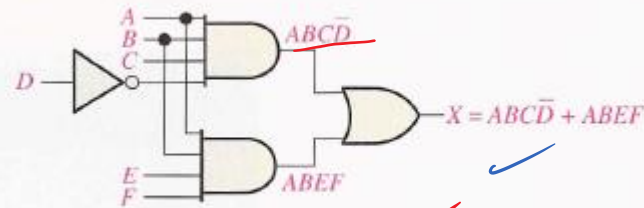
- Implementation of combinational logic

$$X = AB(\overline{CD} + EF)$$

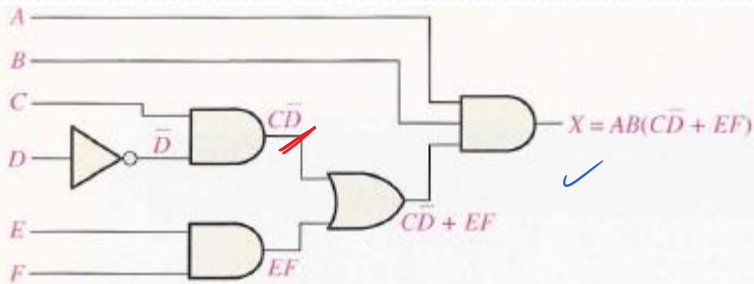


$$AB(\overline{CD} + EF) = \overline{ABCD} + ABEF$$

Handwritten annotations: "NOT" above the bar over C, "AND" under the first term, "OR" under the plus sign, and "AND" under the second term.



(b) Sum-of-products implementation of the circuit in part (a)



(a)

# Combinational Logic Circuit

- Combinational logic circuit from truth table

| INPUTS |   |   | OUTPUT | PRODUCT TERM      |
|--------|---|---|--------|-------------------|
| A      | B | C | X      |                   |
| 0      | 0 | 0 | 0      |                   |
| 0      | 0 | 1 | 0      |                   |
| 0      | 1 | 0 | 0      |                   |
| 0      | 1 | 1 | 1 ✓    | $\bar{A}BC$       |
| 1      | 0 | 0 | 1 ✓    | $A\bar{B}\bar{C}$ |
| 1      | 0 | 1 | 0      |                   |
| 1      | 1 | 0 | 0      |                   |
| 1      | 1 | 1 | 0      |                   |

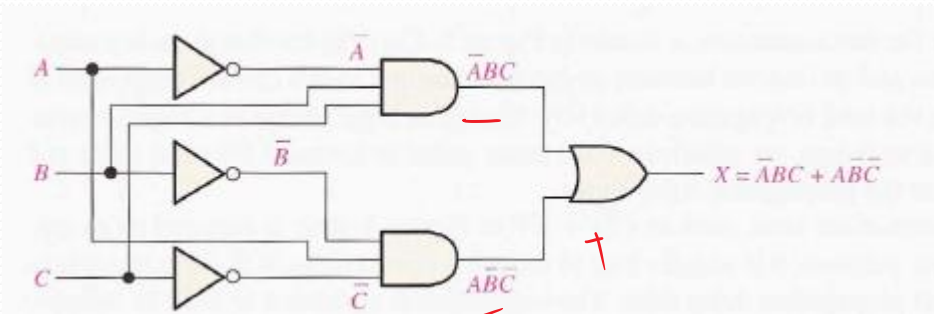
3 Inverters

$$X = \bar{A}BC \oplus A\bar{B}\bar{C} \Rightarrow \text{SOP}$$

AND      OR      NAND More simple

- In SOP form (2 terms)
- In POS form (6 terms)

↳ 6 or gates  
1 and gate



# Combinational Logic Circuit

- Combinational logic circuit from truth table

Design a logic circuit to implement the operation specified in the truth table of Table

TABLE 5-4

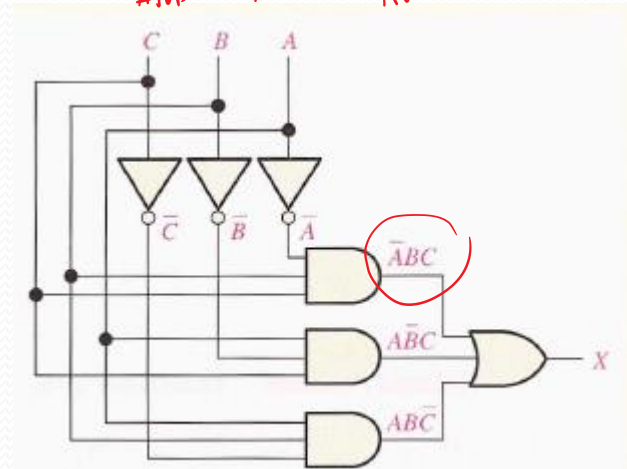
| INPUTS |   |   | OUTPUT | PRODUCT TERM |
|--------|---|---|--------|--------------|
| A      | B | C | X      |              |
| 0      | 0 | 0 | 0      |              |
| 0      | 0 | 1 | 0      |              |
| 0      | 1 | 0 | 0      |              |
| 0      | 1 | 1 | 1      | $\bar{A}BC$  |
| 1      | 0 | 0 | 0      |              |
| 1      | 0 | 1 | 1      | $A\bar{B}C$  |
| 1      | 1 | 0 | 1      | $ABC\bar{C}$ |
| 1      | 1 | 1 | 0      |              |

my choice is in SOP form

OR

$$X = \bar{A}BC + A\bar{B}C + ABC$$

3 Inverter



# Combinational Logic Circuit

- Combinational logic circuit from truth table

## EXAMPLE 5-4

Develop a logic circuit with four input variables that will only produce a 1 output when exactly three input variables are 1s.

**Solution** Out of sixteen possible combinations of four variables, the combinations in which there are exactly three 1s are listed in Table 5-5, along with the corresponding product term for each.

A B C D      x  
 0 1 1 1  
 1 0 1 1  
 1 1 0 1  
 1 1 1 0

TABLE 5-5

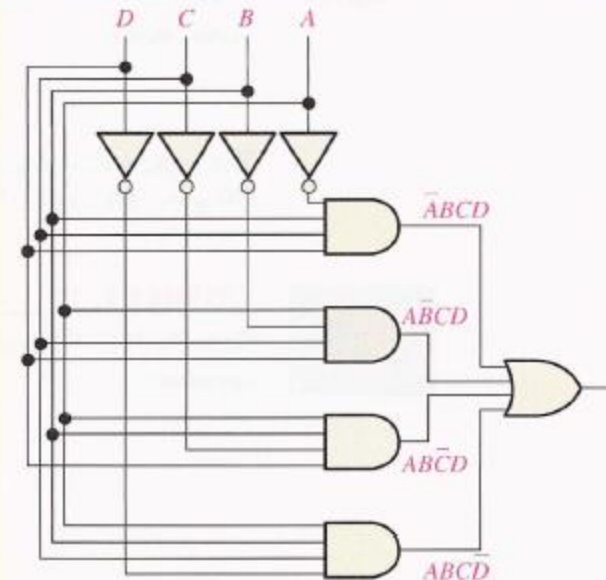
| A | B | C | D | PRODUCT TERM  |
|---|---|---|---|---------------|
| 0 | 1 | 1 | 1 | $\bar{A}BCD$  |
| 1 | 0 | 1 | 1 | $A\bar{B}CD$  |
| 1 | 1 | 0 | 1 | $ABC\bar{D}$  |
| 1 | 1 | 1 | 0 | $ABCD\bar{D}$ |

1 - OR gate  
 4 AND gate  
 4 Inverter

The product terms are ORed to get the following expression:

$$X = \bar{A}BCD + A\bar{B}CD + ABC\bar{D} + ABCD\bar{D}$$

This expression is implemented in Figure 5-11 with AND-OR logic.

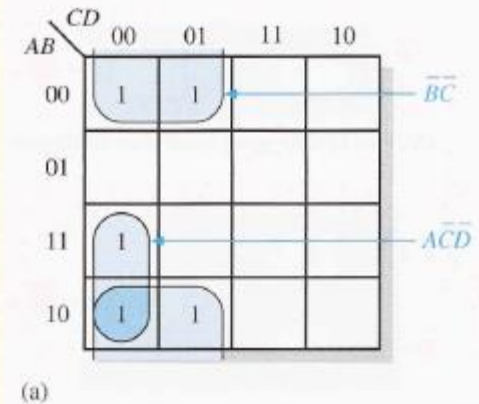
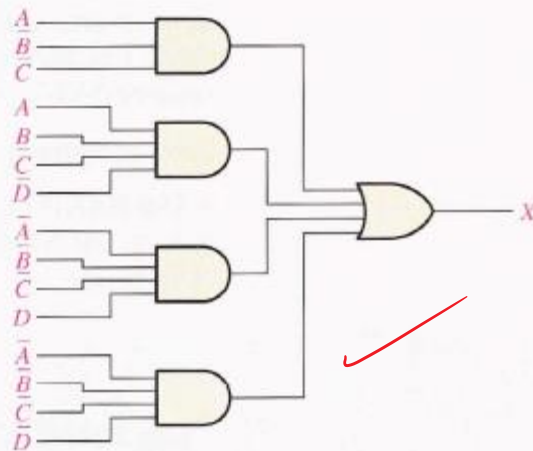


# Combinational Logic Circuit

- Combinational logic circuit from truth table

Minimize the combinational logic circuit in Figure 5–14. Inverters for the complemented variables are not shown.

► FIGURE 5–14

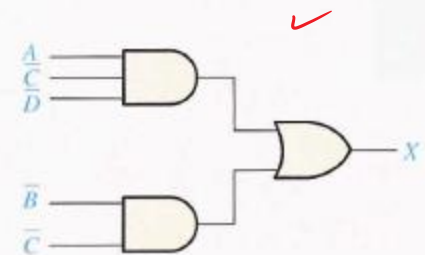


**Solution** The output expression is

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

Expanding the first term to include the missing variables  $D$  and  $\overline{D}$ ,

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



# Combinational Logic Circuit

- Self study sections

## **5** Combinational Logic Analysis 244

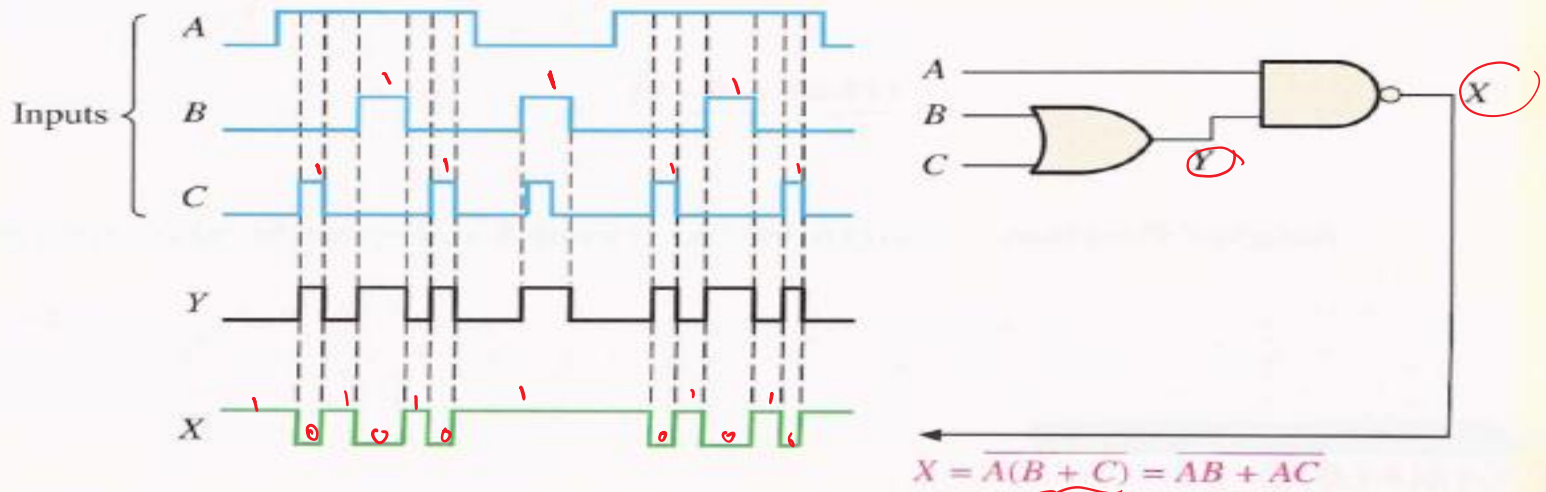
5-3 The Universal Property of NAND and NOR Gates 256

5-4 Combinational Logic Using NAND and NOR Gates 258

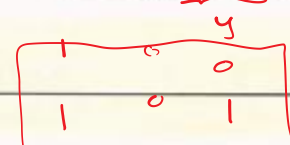
# waveform input

- The operation of the logical gate for pulse input is similar to the input of constant input.

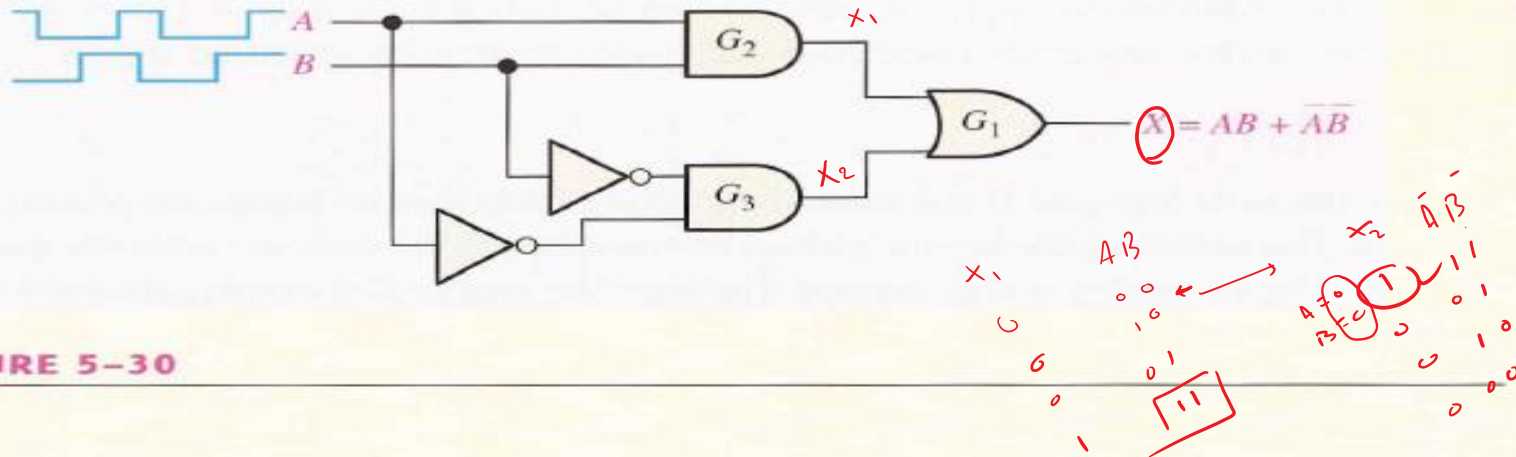
Determine the final output waveform  $X$  for the circuit in Figure 5-29, with input waveforms  $A$ ,  $B$ , and  $C$  as shown.



▲ FIGURE 5-29



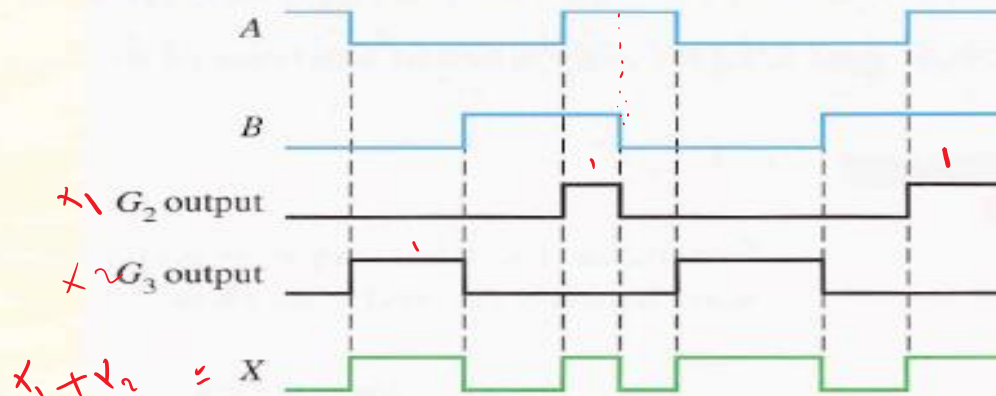
# Logic circuit operation with pulse waveform input



▲ FIGURE 5-30

## Solution

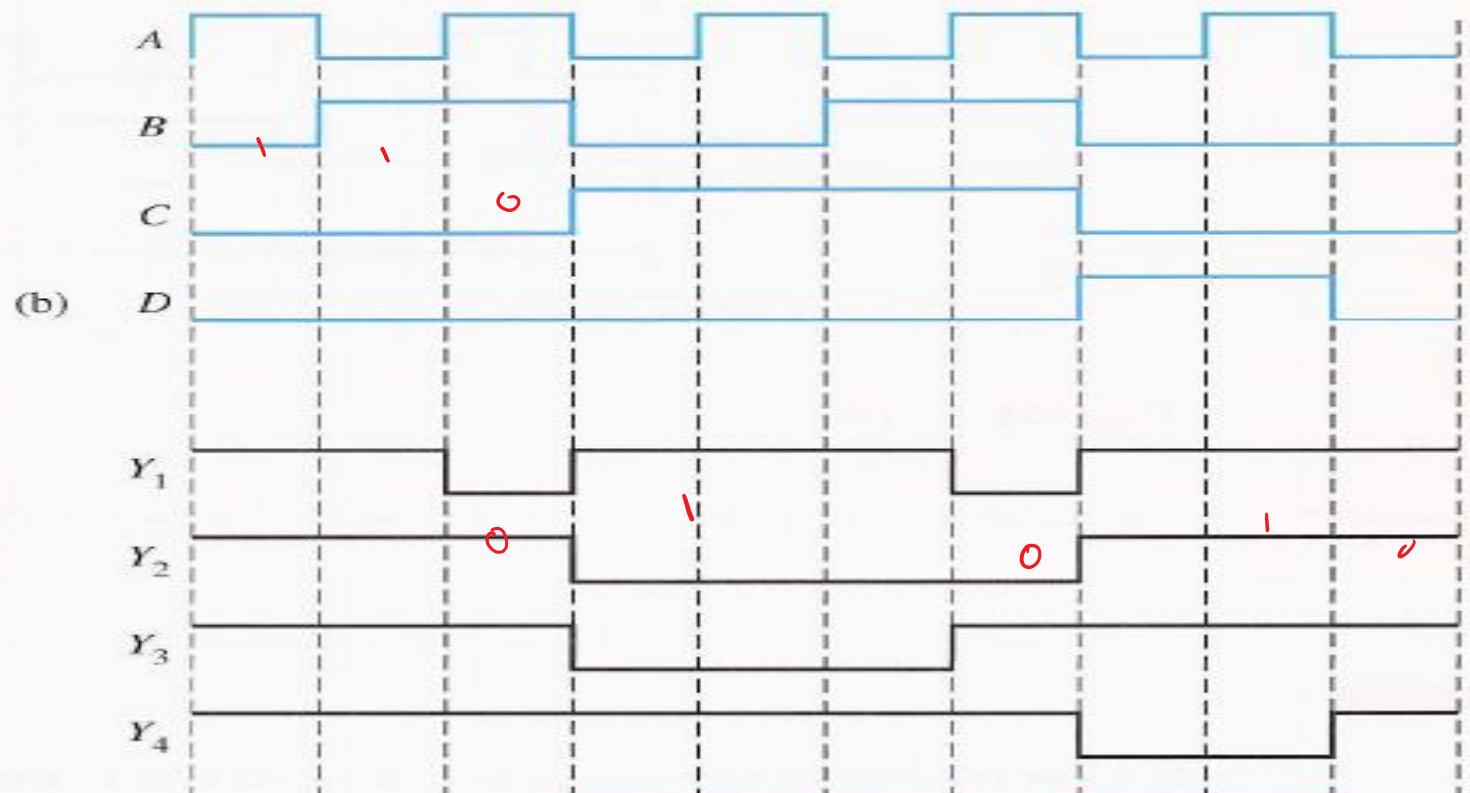
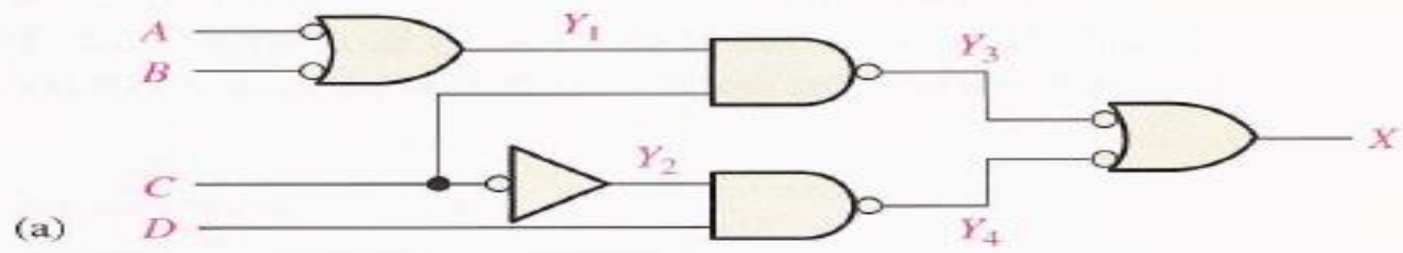
When both inputs are HIGH or when both inputs are LOW, the output  $X$  is HIGH as shown in Figure 5-31. Notice that this is an exclusive-NOR circuit. The intermediate outputs of gates  $G_2$  and  $G_3$  are also shown in Figure 5-31.



▲ FIGURE 5-31



# Logic circuit operation with pulse waveform input



Handwritten notes:  $Y_1$ ,  $Y_2$ ,  $Y_3$ ,  $Y_4$  and a boxed output  $110$ .

Handwritten notes:  $Y_2$  and a boxed output  $110$ .

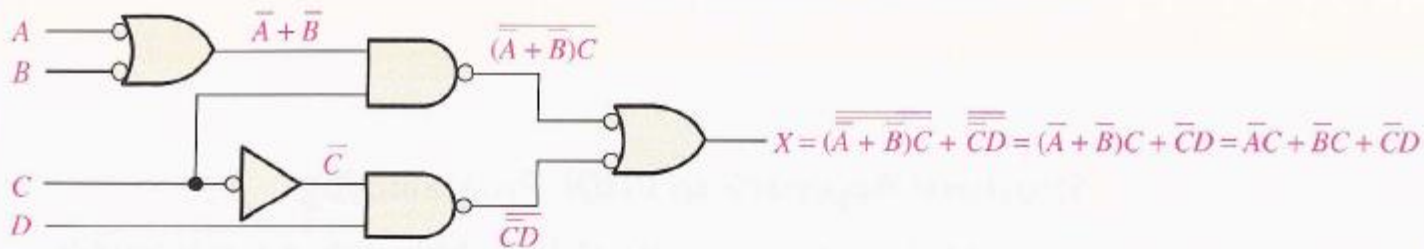
Handwritten notes:  $Y_3$ ,  $Y_4$ , and a boxed output  $110$ .

# waveform input

## EXAMPLE 5-13

Determine the output waveform  $X$  for the circuit in Example 5-12, Figure 5-32(a), directly from the output expression.

**Solution** The output expression for the circuit is developed in Figure 5-33. The SOP form indicates that the output is HIGH when  $A$  is LOW and  $C$  is HIGH or when  $B$  is LOW and  $C$  is HIGH or when  $C$  is LOW and  $D$  is HIGH.



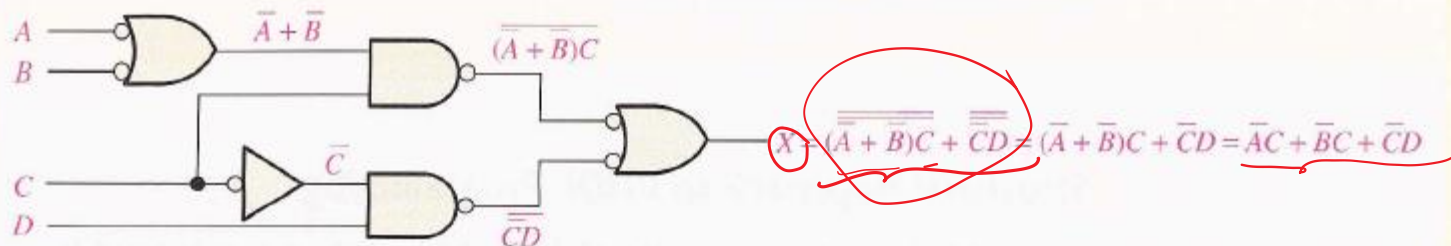
▲ FIGURE 5-33

# waveform input

## EXAMPLE 5-13

Determine the output waveform  $X$  for the circuit in Example 5-12, Figure 5-32(a), directly from the output expression.

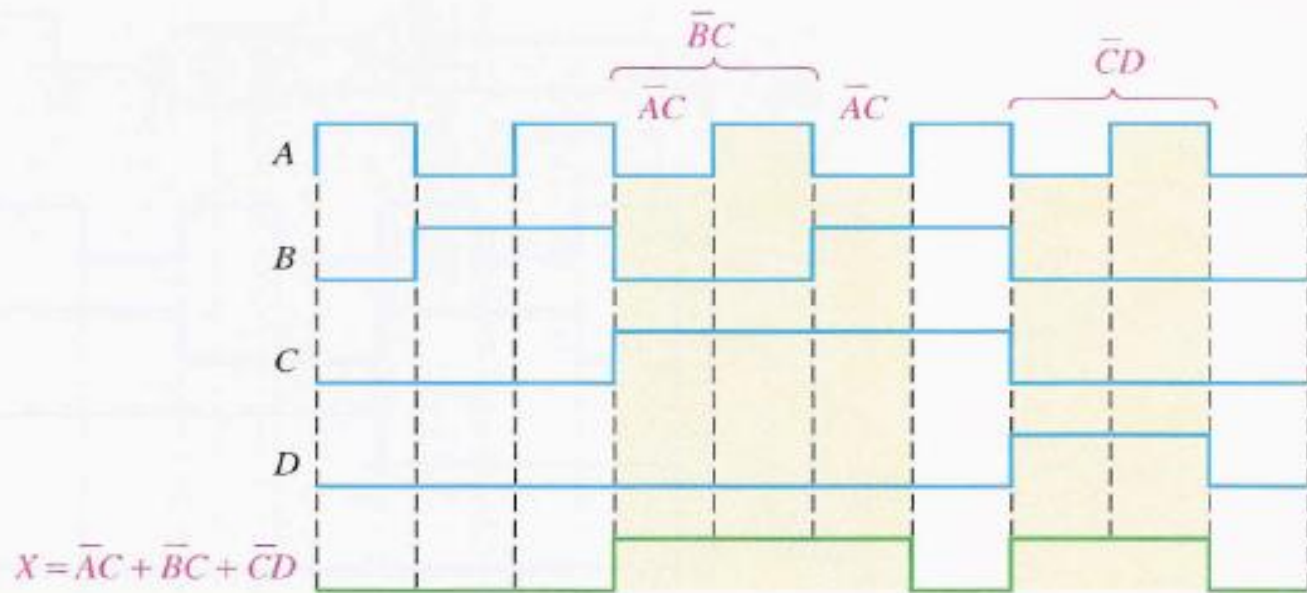
**Solution** The output expression for the circuit is developed in Figure 5-33. The SOP form indicates that the output is HIGH when  $A$  is LOW and  $C$  is HIGH or when  $B$  is LOW and  $C$  is HIGH or when  $C$  is LOW and  $D$  is HIGH.



▲ FIGURE 5-33

# waveform input

The result is shown in Figure 5-34 and is the same as the one obtained by the intermediate-waveform method in Example 5-12. The corresponding product terms for each waveform condition that results in a HIGH output are indicated.



▲ FIGURE 5-34

# Function and Combinational Logic ch6

# Logic Function and Function Combinational

- Adders : Half-adder, Full-adder
- Binary adding

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

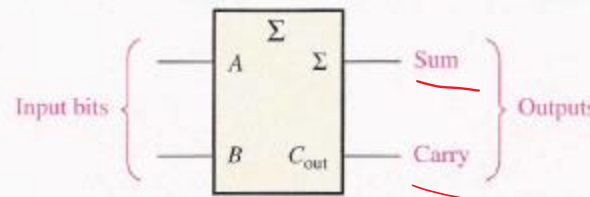
$$1+1=10$$

$$\begin{array}{r} \boxed{1} \\ 1 \\ + \\ 1 \\ \hline \underline{0} \end{array} \text{ sum result}$$

1- Half-adder : where it accepts two binary digit input and results two output sum bit and carry bit

► FIGURE 6-1

Logic symbol for a half-adder. Open file F06-01 to verify operation.



| A | B | C <sub>out</sub> | Σ |
|---|---|------------------|---|
| 0 | 0 | 0                | 0 |
| 0 | 1 | 0                | 1 |
| 1 | 0 | 0                | 1 |
| 1 | 1 | 1                | 0 |

Σ = sum  
C<sub>out</sub> = output carry  
A and B = input variables (operands)

# Logic Function and Function Combinational

▶ **TABLE 6-1**

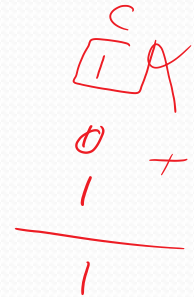
Half-adder truth table.

| $A$ | $B$ | $C_{out}$ | $\Sigma$ |
|-----|-----|-----------|----------|
| 0   | 0   | 0         | 0        |
| 0   | 1   | 0         | 1        |
| 1   | 0   | 0         | 1        |
| 1   | 1   | 1         | 0        |

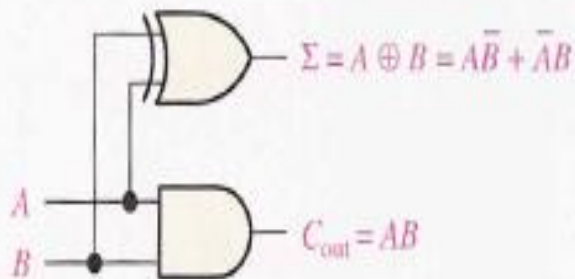
$\Sigma$  = sum

$C_{out}$  = output carry

$A$  and  $B$  = input variables (operands)



- Half-adder circuit



◀ **FIGURE 6-2**

Half-adder logic diagram.



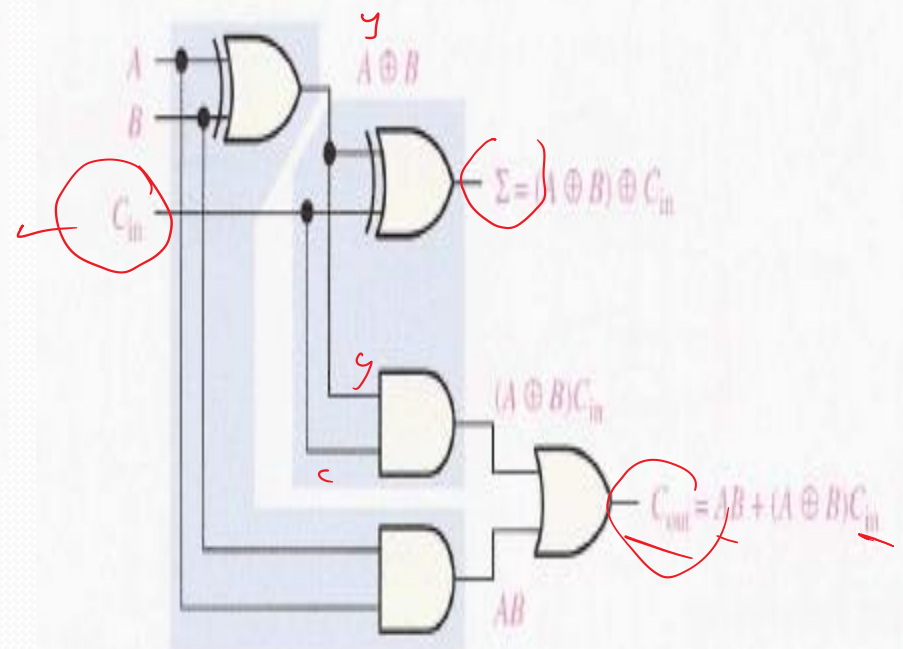


# Logic Function and Function Combinational

| A | B | $C_{in}$ | $C_{out}$ | $\Sigma$ |
|---|---|----------|-----------|----------|
| 0 | 0 | 0        | 0         | 0        |
| 0 | 0 | 1        | 0         | 1        |
| 0 | 1 | 0        | 0         | 1        |
| 0 | 1 | 1        | 1         | 0        |
| 1 | 0 | 0        | 0         | 1        |
| 1 | 0 | 1        | 1         | 0        |
| 1 | 1 | 0        | 1         | 0        |
| 1 | 1 | 1        | 1         | 1        |

$C_{in}$  = input carry, sometimes designated as  $CI$   
 $C_{out}$  = output carry, sometimes designated as  $CO$   
 $\Sigma$  = sum  
 A and B = input variables (operands)

TABLE 6-2  
Full-adder truth table

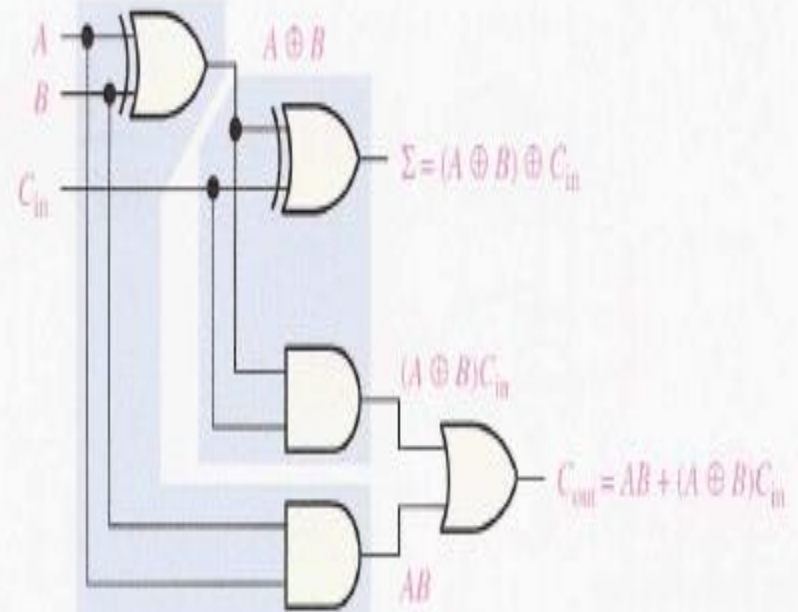


# Logic Function and Function Combinational

| $A$ | $B$ | $C_{in}$ | $C_{out}$ | $\Sigma$ |
|-----|-----|----------|-----------|----------|
| 0   | 0   | 0        | 0         | 0        |
| 0   | 0   | 1        | 0         | 1        |
| 0   | 1   | 0        | 0         | 1        |
| 0   | 1   | 1        | 1         | 0        |
| 1   | 0   | 0        | 0         | 1        |
| 1   | 0   | 1        | 1         | 0        |
| 1   | 1   | 0        | 1         | 0        |
| 1   | 1   | 1        | 1         | 1        |

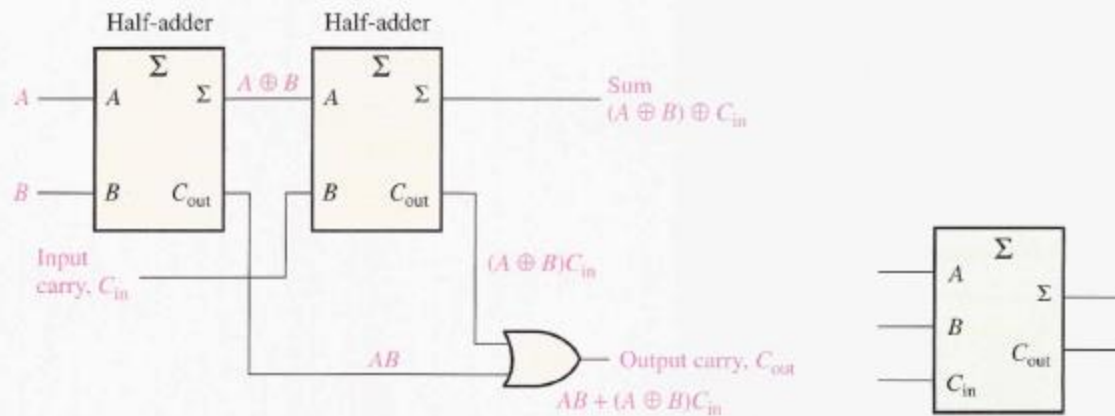
$C_{in}$  = input carry, sometimes designated as  $CI$   
 $C_{out}$  = output carry, sometimes designated as  $CO$   
 $\Sigma$  = sum  
 $A$  and  $B$  = input variables (operands)

TABLE 6-2  
Full-adder truth t

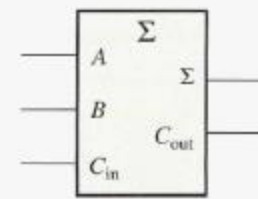


# Logic Function and Function Combinational

- Construct full-adder from two half adder



(a) Arrangement of two half-adders to form a full-adder



(b) Full-adder logic symbol

▲ **FIGURE 6-5**

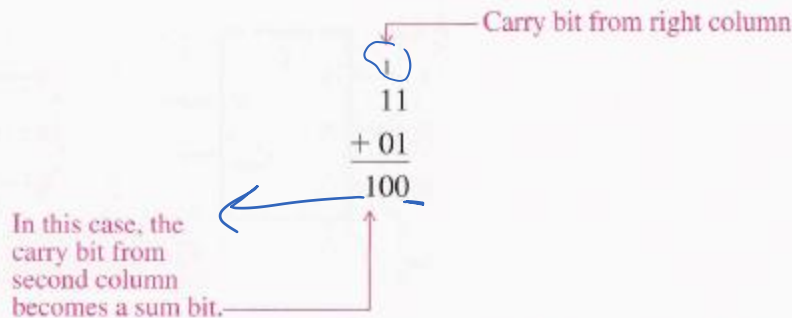
Full-adder implemented with half-adders.

# Logic Function and Function

## Combinational

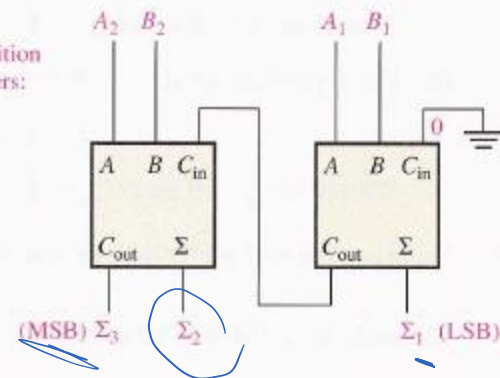
- Parallel binary adder: to add two binary number with number of bit more than one , a number of full adders equal to the number of bits.

*n-bit  
N-full adder*



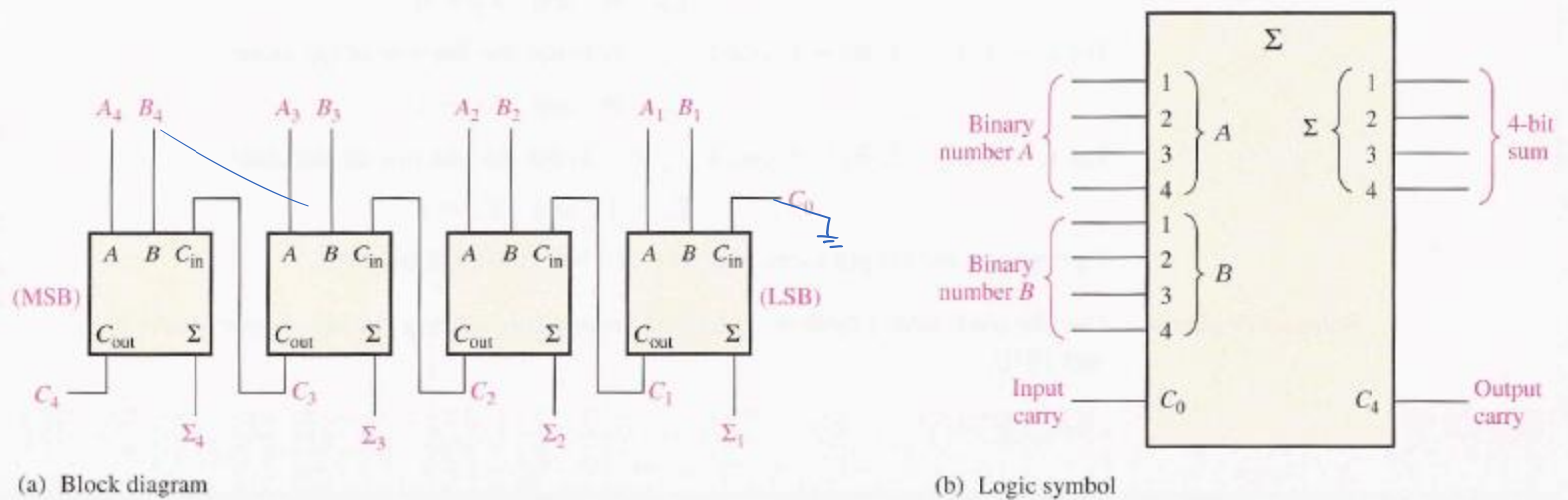
General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ + B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$



# Logic Function and Function Combinational

- Four-bit parallel adder



▲ FIGURE 6-9

A 4-bit parallel adder.

# Logic Function and Function Combinational

- Four-bit parallel adder truth table

| $C_{n-1}$ | $A_n$ | $B_n$ | $\Sigma_n$ | $C_n$ |
|-----------|-------|-------|------------|-------|
| 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     |

TABLE 6-3

Truth table for each stage of a 4-bit parallel adder.

- Example of 4bits adder

# Logic Function and Function Combinational

- Four-bit parallel adder

## EXAMPLE 6-3

Use the 4-bit parallel adder truth table (Table 6-3) to find the sum and output carry for the addition of the following two 4-bit numbers if the input carry ( $C_{n-1}$ ) is 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{and} \quad B_4B_3B_2B_1 = 1100$$

**Solution** For  $n = 1$ :  $A_1 = 0$ ,  $B_1 = 0$ , and  $C_{n-1} = 0$ . From the 1st row of the table,

$$\Sigma_1 = \mathbf{0} \quad \text{and} \quad C_1 = 0$$

For  $n = 2$ :  $A_2 = 0$ ,  $B_2 = 0$ , and  $C_{n-1} = 0$ . From the 1st row of the table,

$$\Sigma_2 = \mathbf{0} \quad \text{and} \quad C_2 = 0$$

For  $n = 3$ :  $A_3 = 1$ ,  $B_3 = 1$ , and  $C_{n-1} = 0$ . From the 4th row of the table,

$$\Sigma_3 = \mathbf{0} \quad \text{and} \quad C_3 = 1$$

For  $n = 4$ :  $A_4 = 1$ ,  $B_4 = 1$ , and  $C_{n-1} = 1$ . From the last row of the table,

$$\Sigma_4 = \mathbf{1} \quad \text{and} \quad C_4 = 1$$

$C_4$  becomes the output carry; the sum of 1100 and 1100 is 11000.

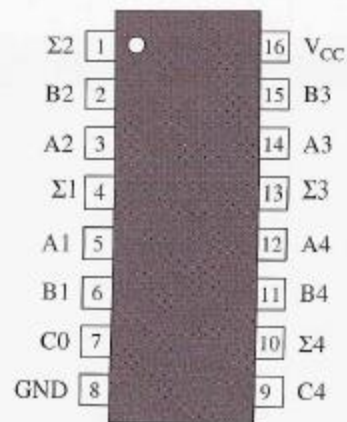
**Related Problem** Use the truth table (Table 6-3) to find the result of adding the binary numbers 1011 and 1010.

# Logic Function and Function Combinational

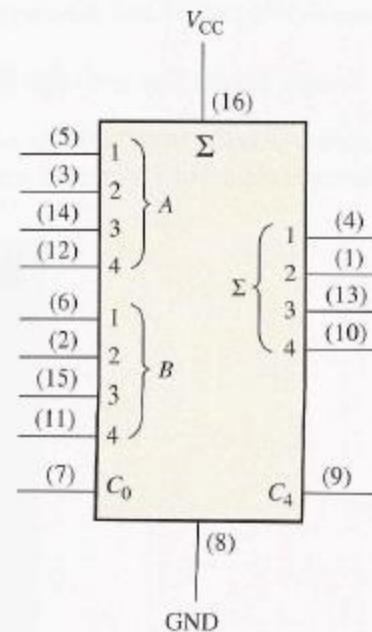
- Four-bit parallel adder

► FIGURE 6-10

Four-bit parallel adder.



(a) Pin diagram of 74LS283

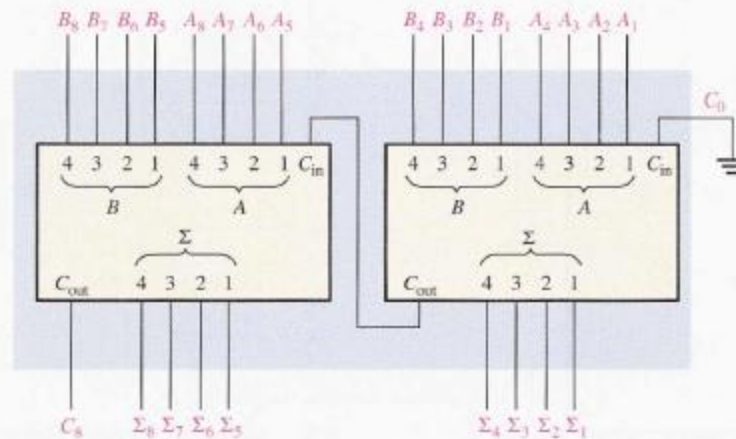


(b) 74LS283 logic symbol

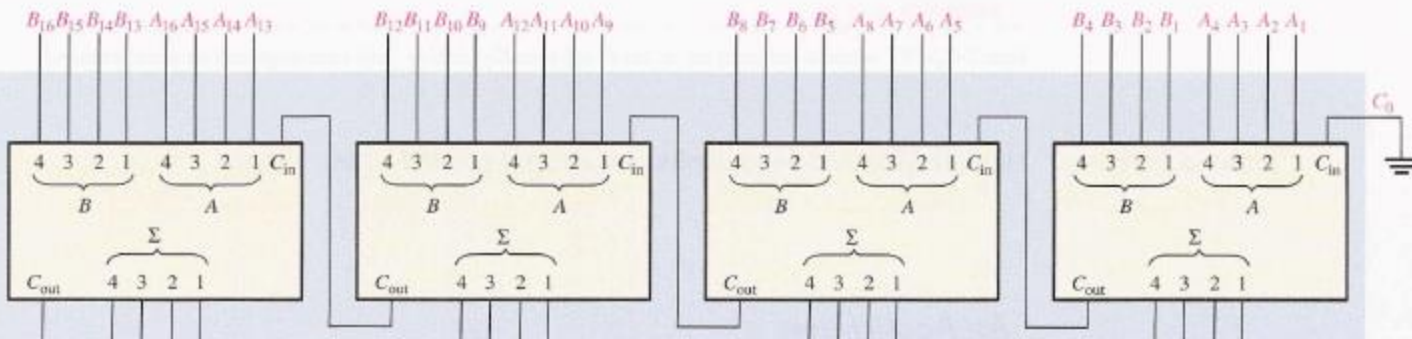


# Logic Function and Function Combinational

- Four-bit parallel adder



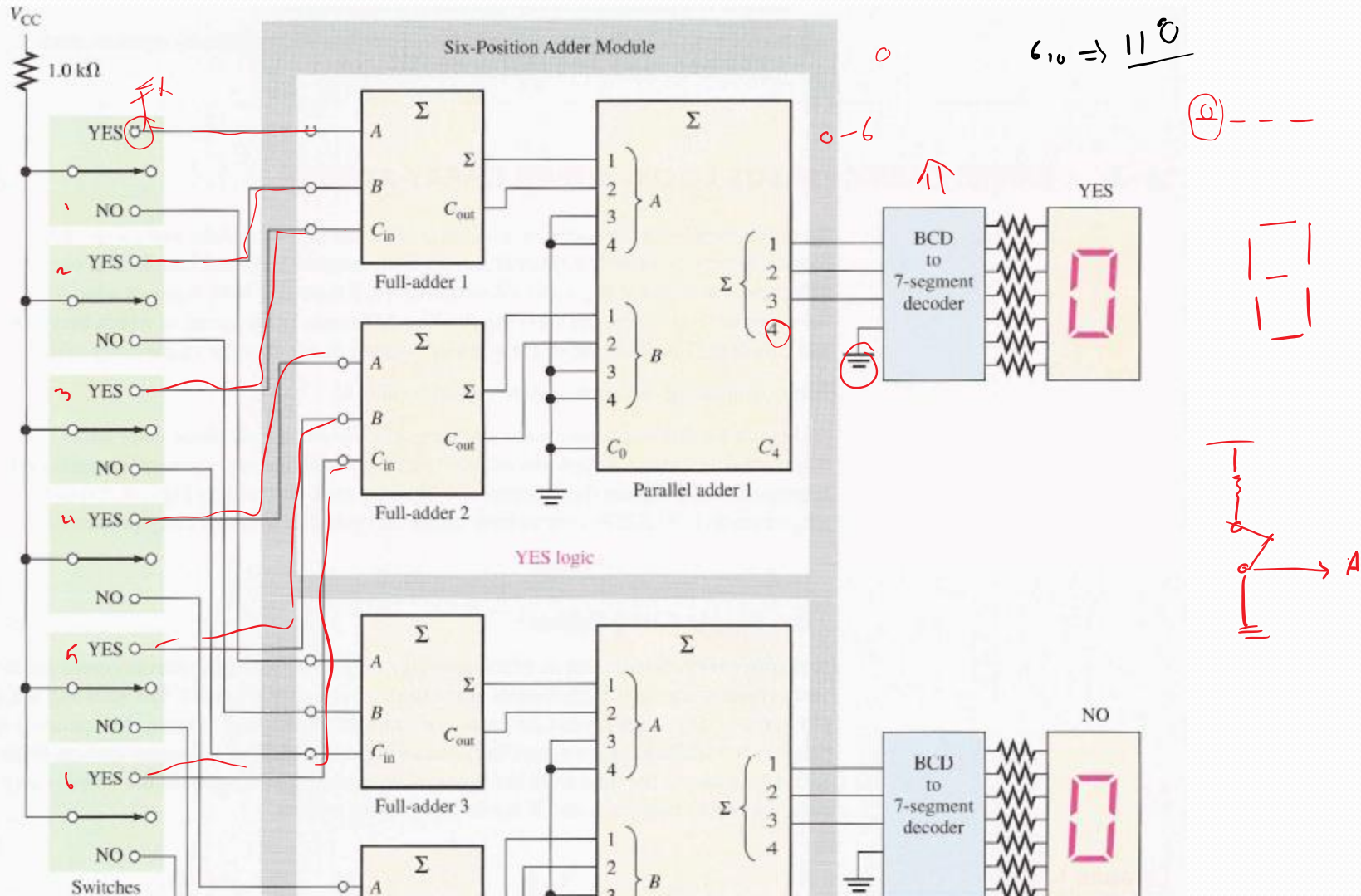
(a) Cascading of two 4-bit adders to form an 8-bit adder



# Logic Function and Function

## Combinational

- Adder application : simple voting system

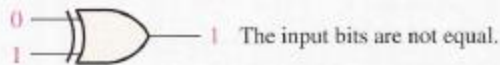
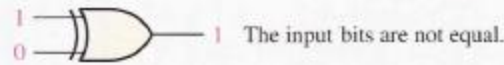


# Logic Function and Function Combinational

- Comparator (<, =, >.)

1- Equality (A=B)

EXOR gate can be used as comparator



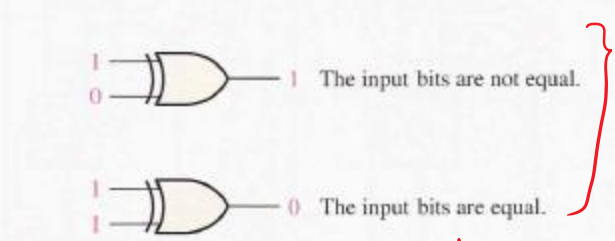
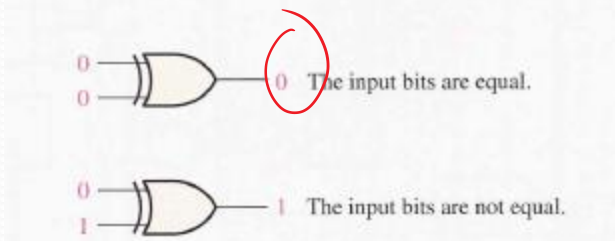
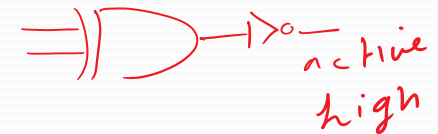
# Logic Function and Function

## Combinational

- Comparator (<, =, >)

1- Equality (A=B)

EXOR gate can be used as comparator



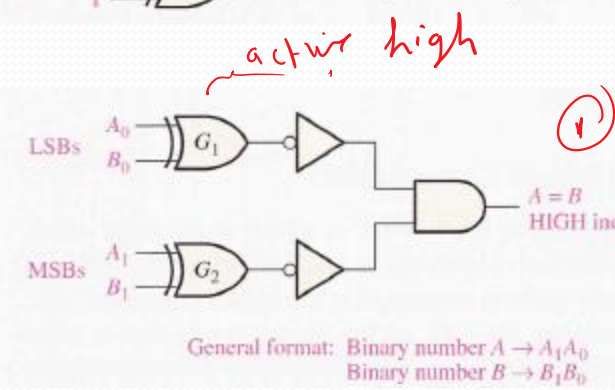
active Low

Two bit comparator



$A_0 \equiv B_0$  ✓

$A_1 \equiv B_1$  ✓



A = B output => 1

active high

A ≠ B output => 0

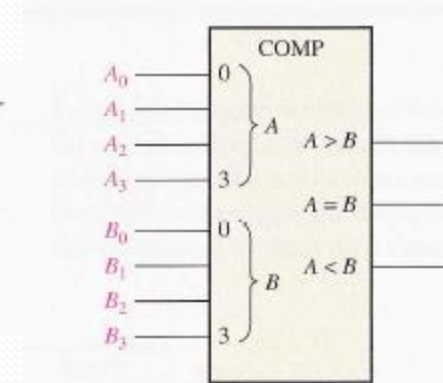
# Logic Function and Function Combinational

- Comparator (<, =, >)

## 1- Inequality ( $A > B$ ) or ( $A < B$ )

To determine an inequality of binary numbers  $A$  and  $B$ , you first examine the highest-order bit in each number. The following conditions are possible:

1. If  $A_3 = 1$  and  $B_3 = 0$ , number  $A$  is greater than number  $B$ .
2. If  $A_3 = 0$  and  $B_3 = 1$ , number  $A$  is less than number  $B$ .
3. If  $A_3 = B_3$ , then you must examine the next lower bit position for an inequality.



# Logic Function and Function

## Combinational

- Comparator ( $<$ ,  $=$ ,  $>$ ,  $\leq$ ,  $\geq$ ,)
- 1- Inequality ( $A > B$ ) or ( $A < B$ )

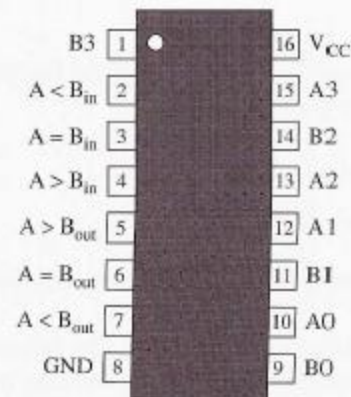
### THE 74HC85 4-BIT MAGNITUDE COMPARATOR



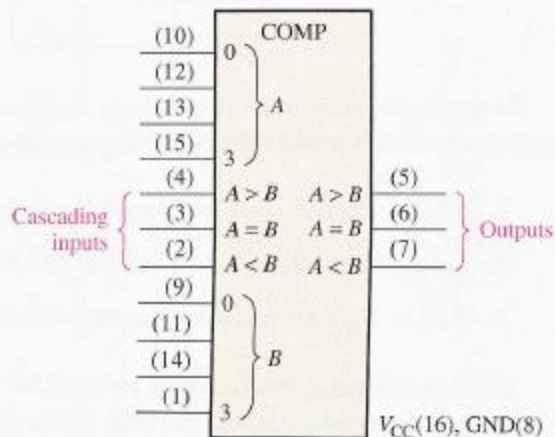
The 74HC85 is a comparator that is also available in other IC families. The pin diagram and logic symbol are shown in Figure 6–24. Notice that this device has all the inputs and outputs of the generalized comparator previously discussed and, in addition, has three cascading inputs:  $A < B$ ,  $A = B$ ,  $A > B$ . These inputs allow several comparators to be cascaded for comparison of any number of bits greater than four. To expand the comparator, the  $A < B$ ,

► FIGURE 6–24

Pin diagram and logic symbol for the 74HC85 4-bit magnitude comparator (pin numbers are in parentheses).



(a) Pin diagram



(b) Logic symbol

# Logic Function and Function Combinational

- Comparator ( $<$ ,  $=$ ,  $>$ ,  $\leq$ ,  $\geq$ ,)
- 1- Inequality ( $A > B$ ) or ( $A < B$ )

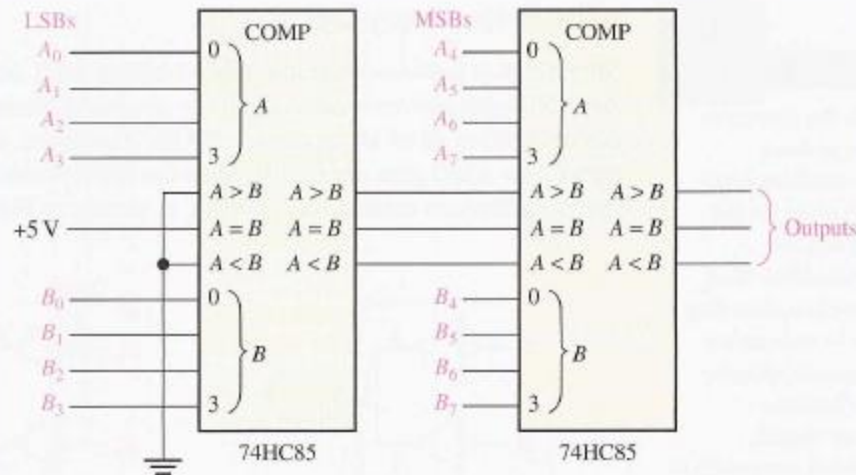
## EXAMPLE 6-7

Use 74HC85 comparators to compare the magnitudes of two 8-bit numbers. Show the comparators with proper interconnections.

**Solution** Two 74HC85s are required to compare two 8-bit numbers. They are connected as shown in Figure 6-25 in a cascaded arrangement.

### ► FIGURE 6-25

An 8-bit magnitude comparator using two 74HC85s.



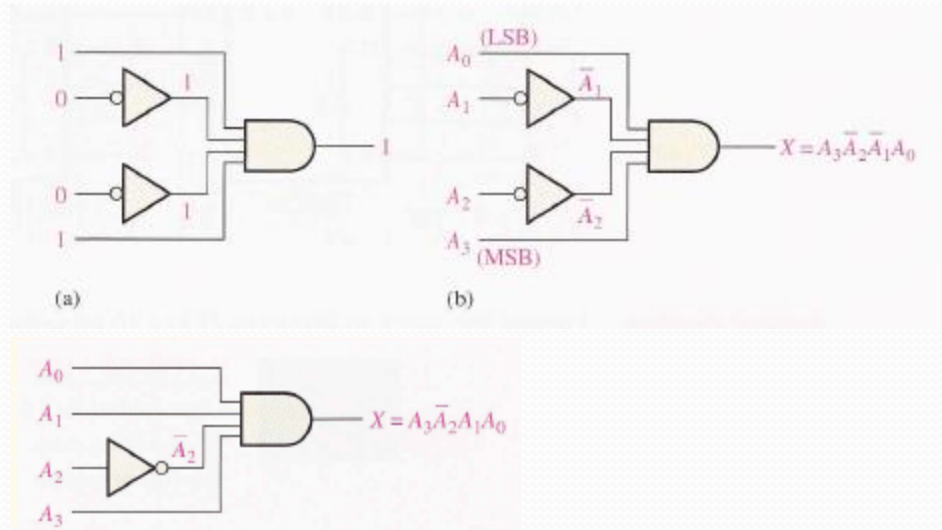
**Related Problem** Expand the circuit in Figure 6-25 to a 16-bit comparator.

# Logic Function and Function

## Combinational

- Decoder : a digital circuit that can detect the presence of certain binary combination.

Examples :







# Logic Function and Function Combinational

## THE 74HC154 1-OF-16 DECODER



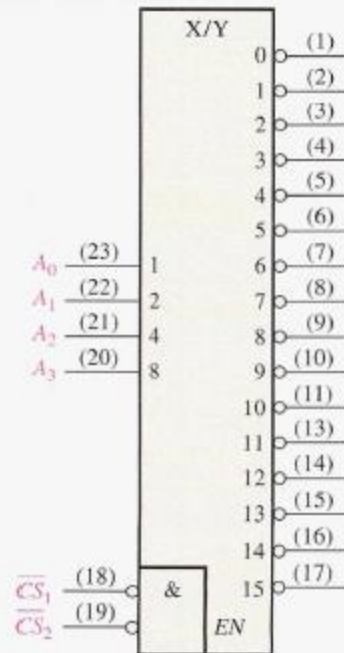
The 74HC154 is a good example of an IC decoder. The logic symbol is shown in Figure 6-29. There is an enable function ( $EN$ ) provided on this device, which is implemented with a NOR gate used as a negative-AND. A LOW level on each chip select input,  $\overline{CS}_1$  and  $\overline{CS}_2$ , is required in order to make the enable gate output ( $EN$ ) HIGH. The enable gate output is

► FIGURE 6-29

Pin diagram and logic symbol for the 74HC154 1-of-16 decoder.



(a) Pin diagram



(b) Logic symbol

# Logic Function and Function

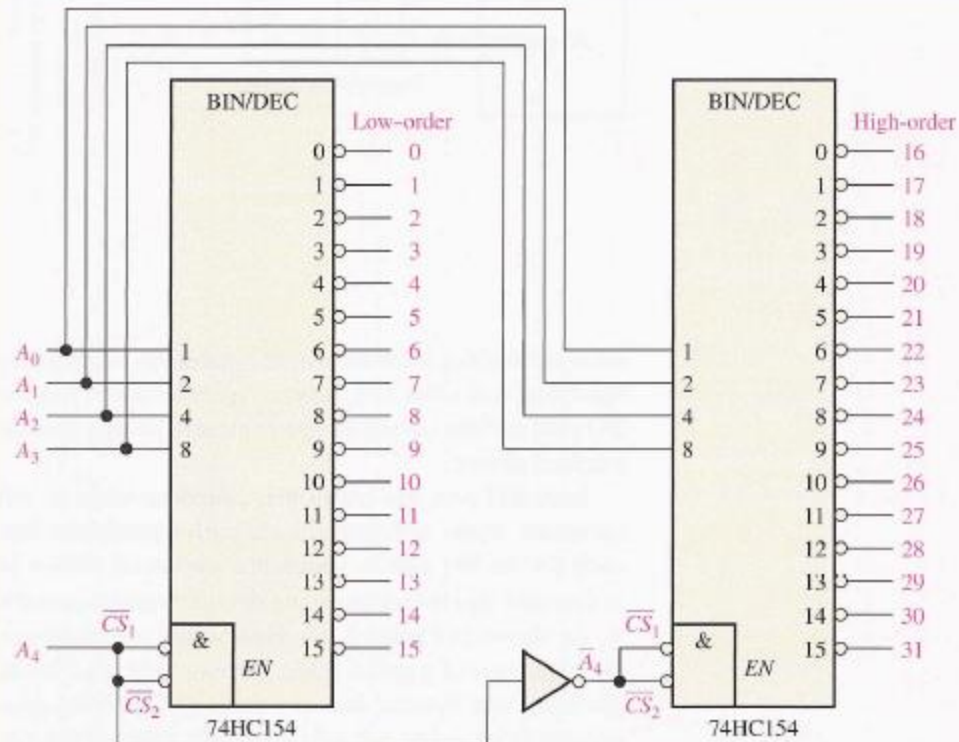
## EXAMPLE 6-9

A certain application requires that a 5-bit number be decoded. Use 74HC154 decoders to implement the logic. The binary number is represented by the format  $A_4A_3A_2A_1A_0$ .

**Solution** Since the 74HC154 can handle only four bits, two decoders must be used to decode five bits. The fifth bit,  $A_4$ , is connected to the chip select inputs,  $\overline{CS}_1$  and  $\overline{CS}_2$ , of one decoder, and  $\overline{A}_4$  is connected to the  $\overline{CS}_1$  and  $\overline{CS}_2$  inputs of the other decoder, as shown in Figure 6-30. When the decimal number is 15 or less,  $A_4 = 0$ , the low-order decoder is enabled, and the high-order decoder is disabled. When the decimal number is greater than 15,  $A_4 = 1$  so  $\overline{A}_4 = 0$ , the high-order decoder is enabled, and the low-order decoder is disabled.

► FIGURE 6-30

A 5-bit decoder using 74HC154s.



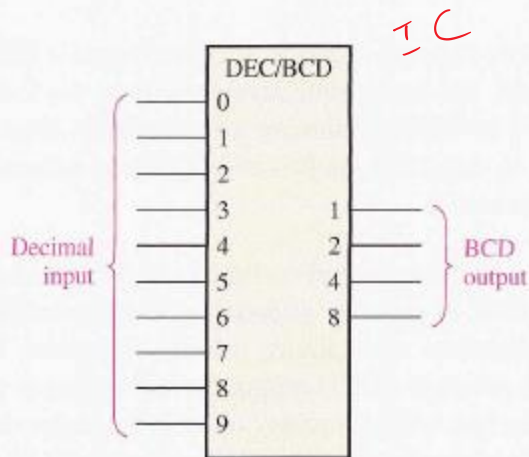
# Logic Function and Function

## Combinational

- Encoder : is a digital logic circuit that reverse the decoder function

Example : The Decimal to BCD

$2 @ 10$   $(000d \ 0100)_2$   
 $(\underline{00} \ \underline{10} \ \underline{0000})_{BCD}$



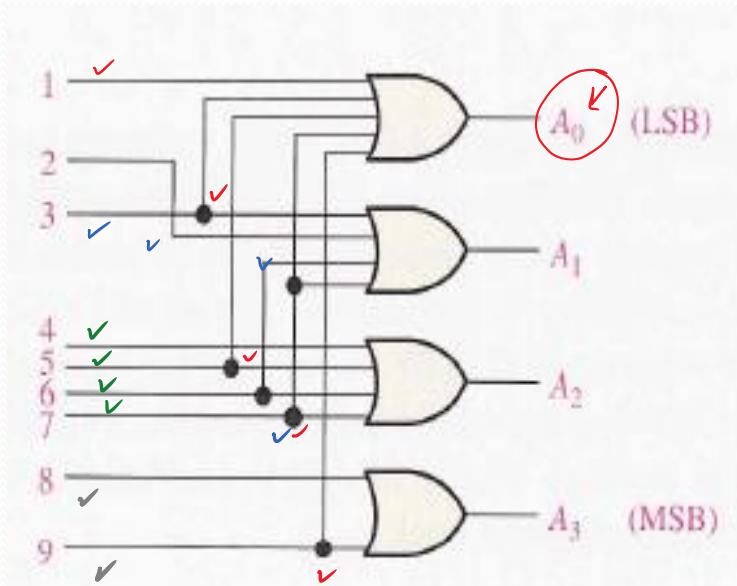
| DECIMAL DIGIT | BCD CODE |       |       |       |
|---------------|----------|-------|-------|-------|
|               | $A_3$    | $A_2$ | $A_1$ | $A_0$ |
| 0             | 0        | 0     | 0     | 0     |
| 1             | 0        | 0     | 0     | 1     |
| 2             | 0        | 0     | 1     | 0     |
| 3             | 0        | 0     | 1     | 1     |
| 4             | 0        | 1     | 0     | 0     |
| 5             | 0        | 1     | 0     | 1     |
| 6             | 0        | 1     | 1     | 0     |
| 7             | 0        | 1     | 1     | 1     |
| 8             | 1        | 0     | 0     | 0     |
| 9             | 1        | 0     | 0     | 1     |

# Logic Function and Function

## Combinational

- Decimal to BCD Encoder
- Digital circuit

*OR gate*

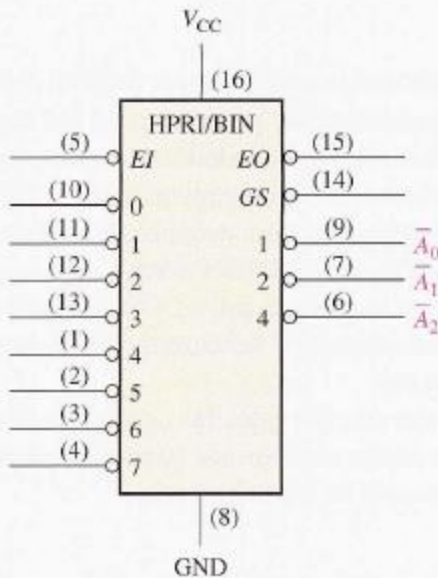


| DECIMAL DIGIT | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---------------|-------|-------|-------|-------|
| 0             | 0     | 0     | 0     | 0     |
| 1             | 0     | 0     | 0     | 1 ✓   |
| 2             | 0     | 0     | 1 ✓   | 0     |
| 3             | 0     | 0     | 1 ✓   | 1 ✓   |
| 4             | 0     | 1 ✓   | 0     | 0     |
| 5             | 0     | 1 ✓   | 0     | 1 ✓   |
| 6             | 0     | 1 ✓   | 1 ✓   | 0     |
| 7             | 0     | 1 ✓   | 1 ✓   | 1 ✓   |
| 8             | 1 ✓   | 0     | 0     | 0     |
| 9             | 1 ✓   | 0     | 0     | 1 ✓   |

# Logic Function and Function Combinational

- 8 lines to 3 lines encoder (74LS148)

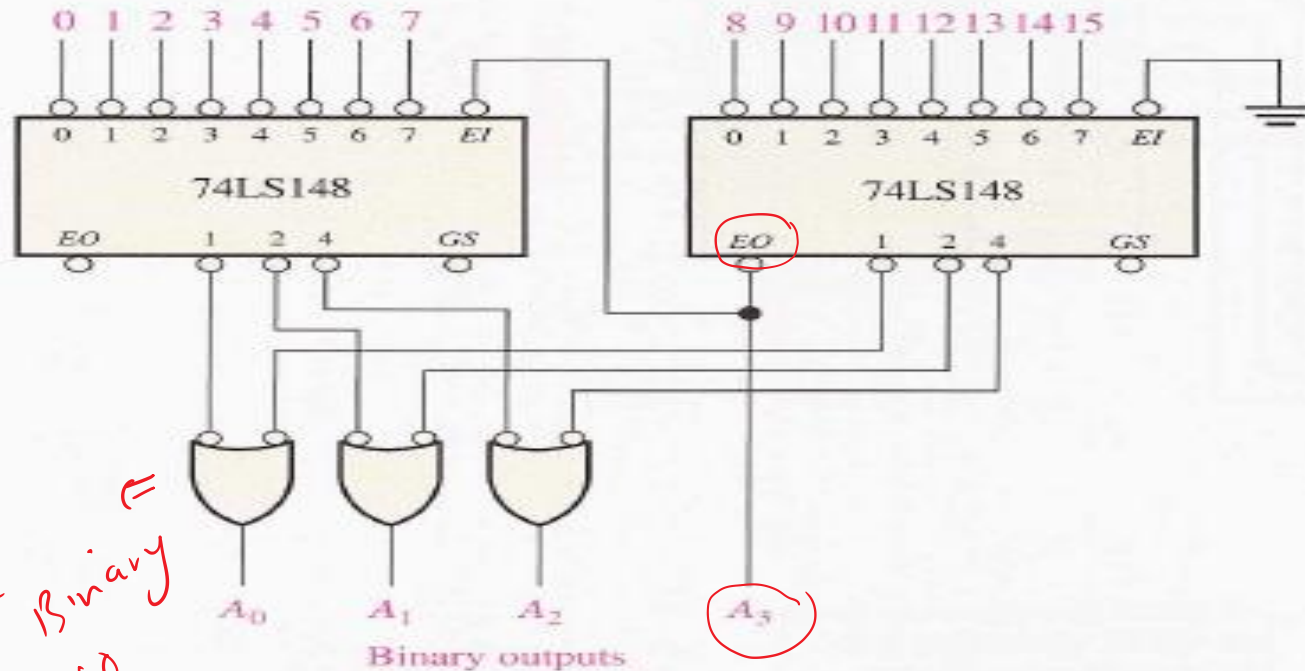
↳ dec → octal



The 74LS148 is a priority encoder that has eight active-LOW inputs and three active-LOW binary outputs, as shown in Figure 6-40. This device can be used for converting octal inputs (recall that the octal digits are 0 through 7) to a 3-bit binary code. To enable the device, the  $EI$  (enable input) must be **LOW**. It also has the  $EO$  (enable output) and  $GS$  output for expansion purposes. The  $EO$  is **LOW** when the  $EI$  is **LOW** and none of the inputs (0 through 7) is active.  $GS$  is **LOW** when  $EI$  is **LOW** and any of the inputs is active. This device may be available in other TTL or CMOS families. Check the Texas Instruments website at [www.ti.com](http://www.ti.com) or the TI CD-ROM accompanying this book.

# Logic Function and Function Combinational

- Expand the 8 lines to 3 lines encoder to 16 lines to 4 lines



$A_3$  (8-15)  
 |  
 -----  
 $A_3(0)$  (0-7)  
 =

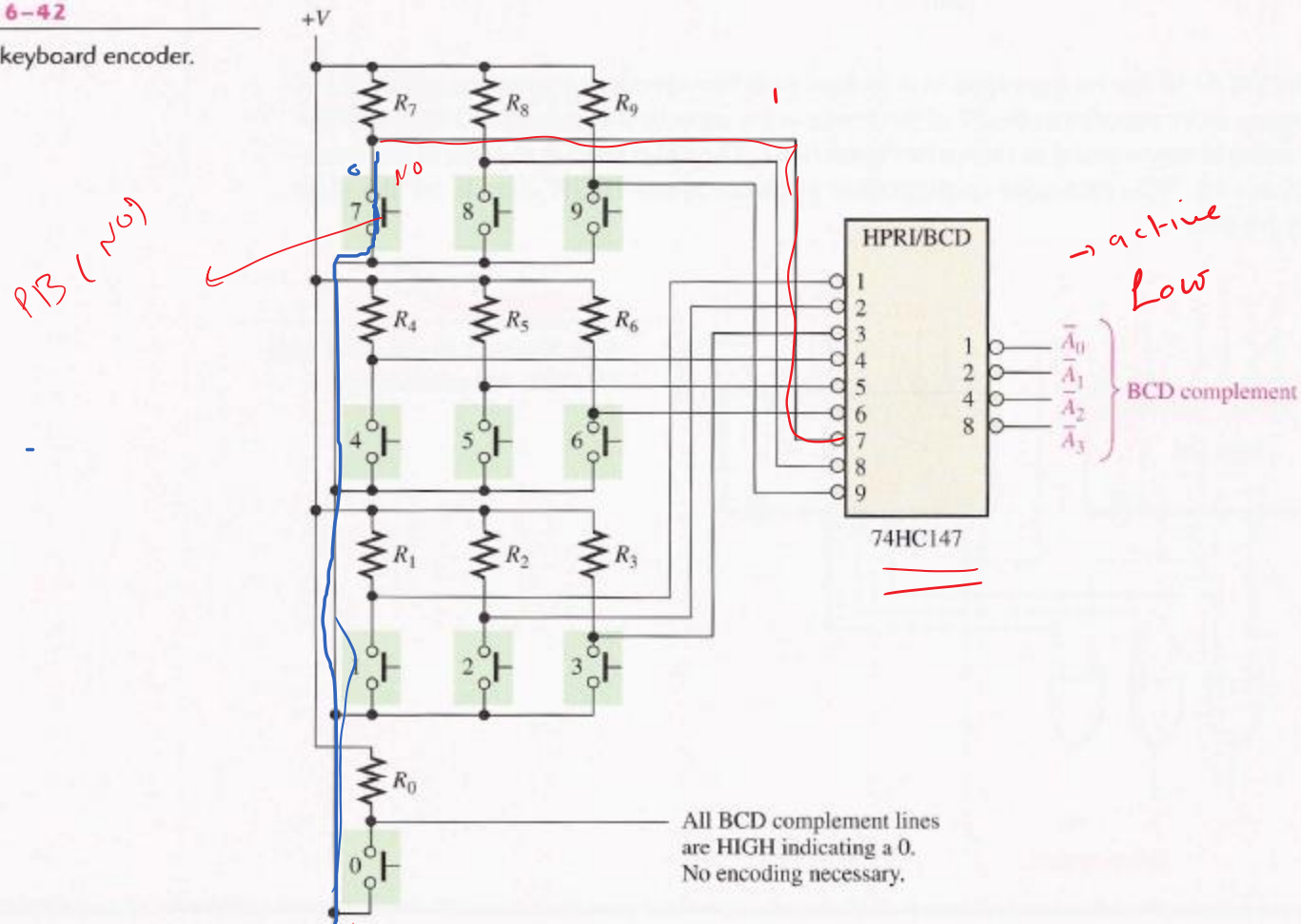
Not OR  
 active  
 4 bit  
 Binary  
 No

# Logic Function and Function Combinational

- Encoder Application (Key board)

▶ FIGURE 6-42

A simplified keyboard encoder.





# Logic Function and Function

## Combinational

- Code converter : is a logic circuit that convert from one code to another .

### Example : BCD to Binary conversion

One method of BCD-to-binary code conversion uses adder circuits. The basic conversion process is as follows:

1. The value, or weight, of each bit in the BCD number is represented by a binary number.
2. All of the binary representations of the weights of bits that are 1s in the BCD number are added.
3. The result of this addition is the binary equivalent of the BCD number.

| BCD BIT | BCD WEIGHT | (MSB) BINARY REPRESENTATION (LSB) |    |    |   |   |   |   |
|---------|------------|-----------------------------------|----|----|---|---|---|---|
|         |            | 64                                | 32 | 16 | 8 | 4 | 2 | 1 |
| $A_0$   | 1          | 0                                 | 0  | 0  | 0 | 0 | 0 | 1 |
| $A_1$   | 2          | 0                                 | 0  | 0  | 0 | 0 | 1 | 0 |
| $A_2$   | 4          | 0                                 | 0  | 0  | 0 | 1 | 0 | 0 |
| $A_3$   | 8          | 0                                 | 0  | 0  | 1 | 0 | 0 | 0 |
| $B_0$   | 10         | 0                                 | 0  | 0  | 1 | 0 | 1 | 0 |
| $B_1$   | 20         | 0                                 | 0  | 1  | 0 | 1 | 0 | 0 |
| $B_2$   | 40         | 0                                 | 1  | 0  | 1 | 0 | 0 | 0 |
| $B_3$   | 80         | 1                                 | 0  | 1  | 0 | 0 | 0 | 0 |

# Logic Function and Function

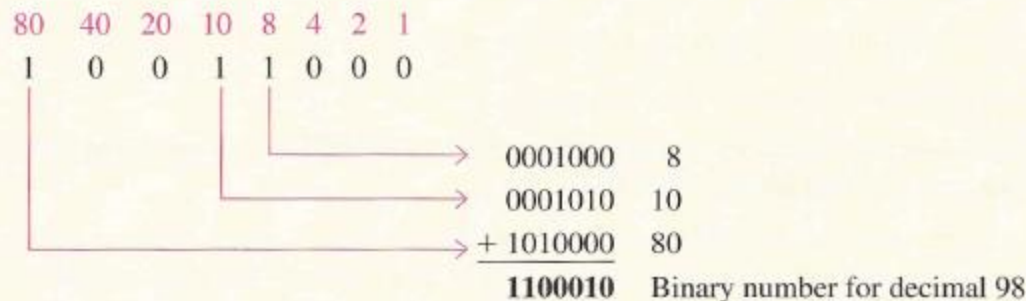
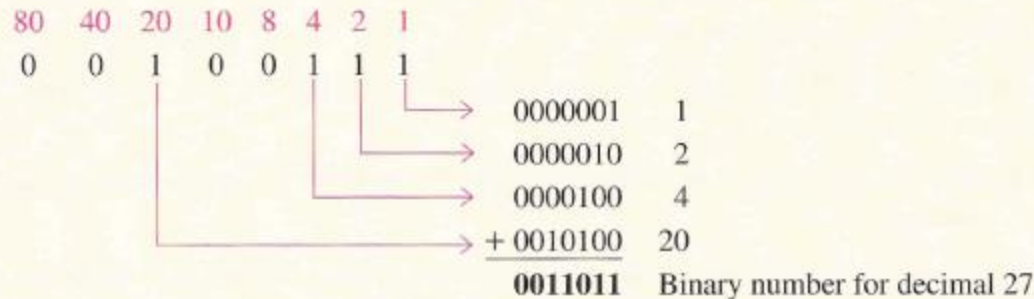
## Combinational

- Code converter : is a logic circuit that convert from one code to another .

### EXAMPLE 6-12

Convert the BCD numbers 00100111 (decimal 27) and 10011000 (decimal 98) to binary.

**Solution** Write the binary representations of the weights of all 1s appearing in the numbers, and then add them together.



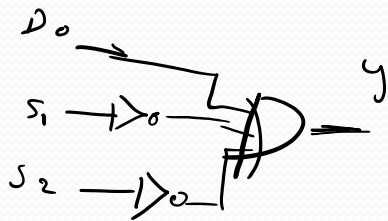
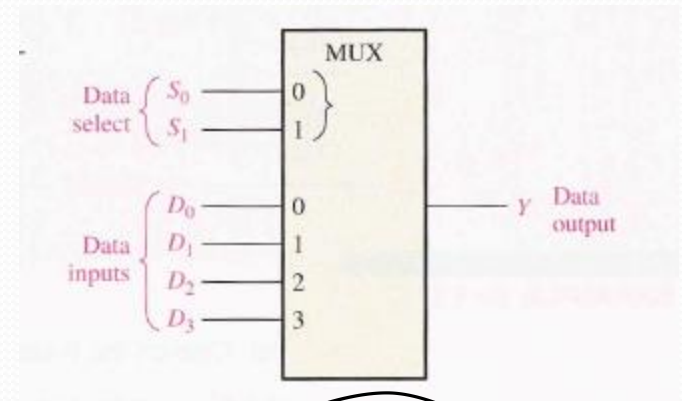
# Logic Function and Function

## Combinational

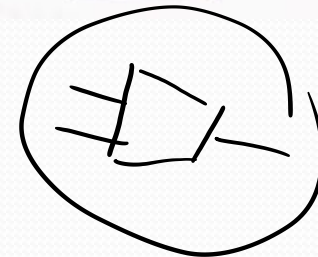
- Multiplexers (MUX): it is a digital device that allows digital information from different sources to be routed into a single line

Example : 1 of 4 data selector

| DATA-SELECT INPUTS |       | INPUT SELECTED |
|--------------------|-------|----------------|
| $S_1$              | $S_0$ |                |
| 0                  | 0     | $D_0$          |
| 0                  | 1     | $D_1$          |
| 1                  | 0     | $D_2$          |
| 1                  | 1     | $D_3$          |



$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$



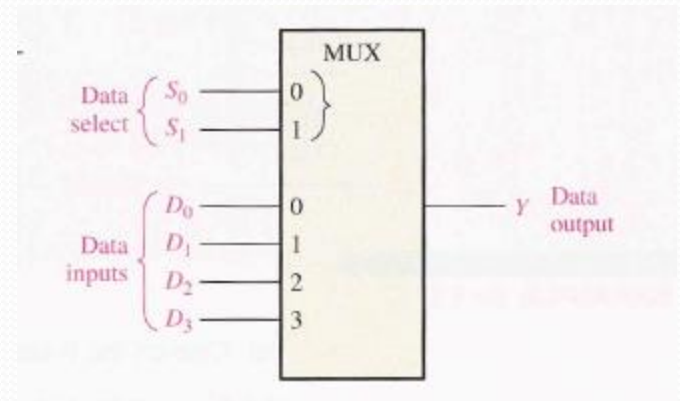
# Logic Function and Function

## Combinational

- Multiplexers (MUX): it is a digital device that allows digital information from different sources to be routed into a single line

Example : 1 of 4 data selector

| DATA-SELECT INPUTS |       | INPUT SELECTED |
|--------------------|-------|----------------|
| $S_1$              | $S_0$ |                |
| 0                  | 0     | $D_0$          |
| 0                  | 1     | $D_1$          |
| 1                  | 0     | $D_2$          |
| 1                  | 1     | $D_3$          |



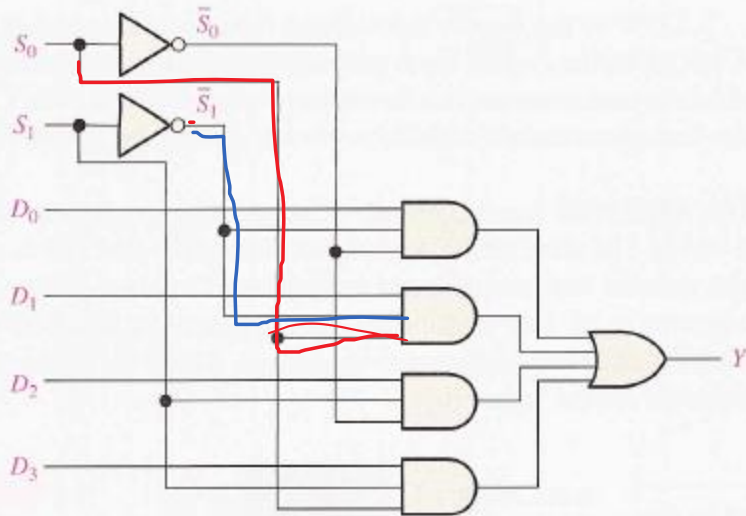
$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$

# Logic Function and Function

## Combinational

- Multiplexers (MUX): it is a digital device that allows digital information from different sources to be routed into a single line

Example : 1 of 4 data selector

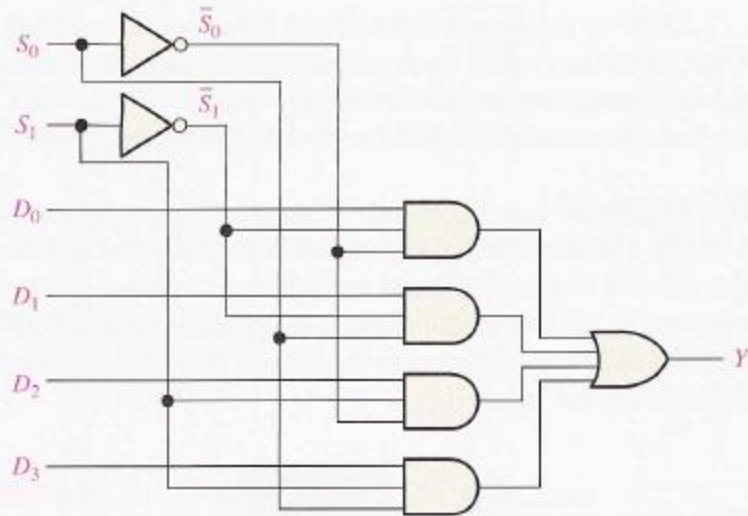


| DATA-SELECT INPUTS |       | INPUT SELECTED |
|--------------------|-------|----------------|
| $S_1$              | $S_0$ |                |
| 0                  | 0     | $D_0$          |
| 0                  | 1     | $D_1$          |
| 1                  | 0     | $D_2$          |
| 1                  | 1     | $D_3$          |

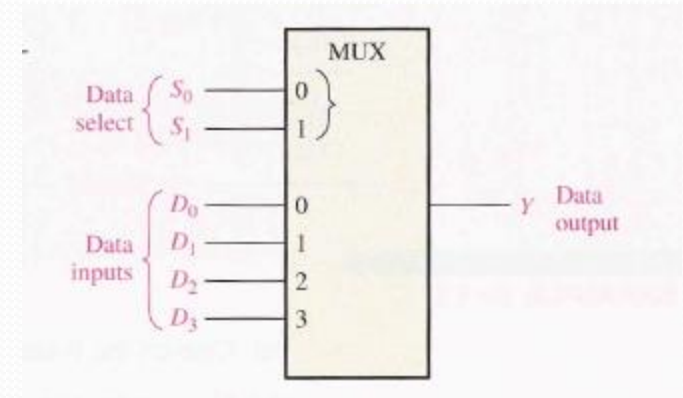
$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$

# Logic Function and Function Combinational

- Example : 1 of 4 data selector



$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$



# Logic Function and Function

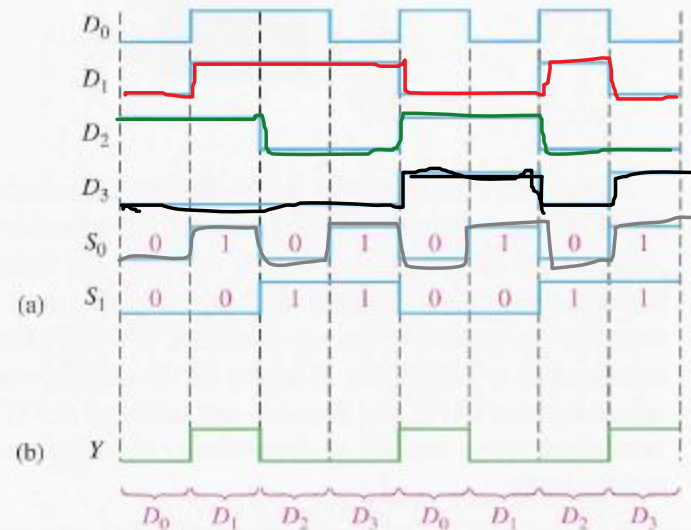
## Combinational

- Example : 1 of 4 data selector

### EXAMPLE 6-14

The data-input and data-select waveforms in Figure 6-48(a) are applied to the multiplexer in Figure 6-47. Determine the output waveform in relation to the inputs.

FIGURE 6-48



**Solution** The binary state of the data-select inputs during each interval determines which data input is selected. Notice that the data-select inputs go through a repetitive binary sequence 00, 01, 10, 11, 00, 01, 10, 11, and so on. The resulting output waveform is shown in Figure 6-48(b).

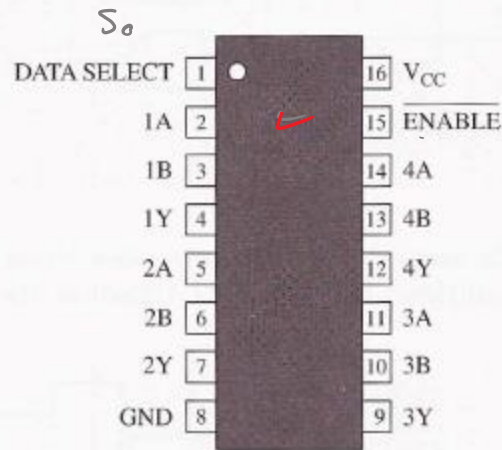
# Logic Function and Function Combinational

- Example : 74HC157

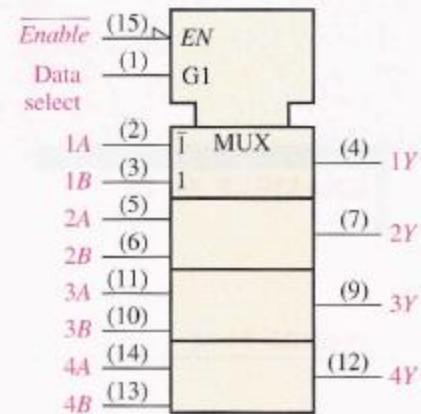
$S = 0 \Rightarrow A \Rightarrow 4 \text{ bits information}$   
 $S = 1 \Rightarrow B \Rightarrow 4 \text{ bits information}$

► FIGURE 6-49

Pin diagram and logic symbol for the 74HC157 quadruple 2-input data selector/multiplexer.



(a) Pin diagram

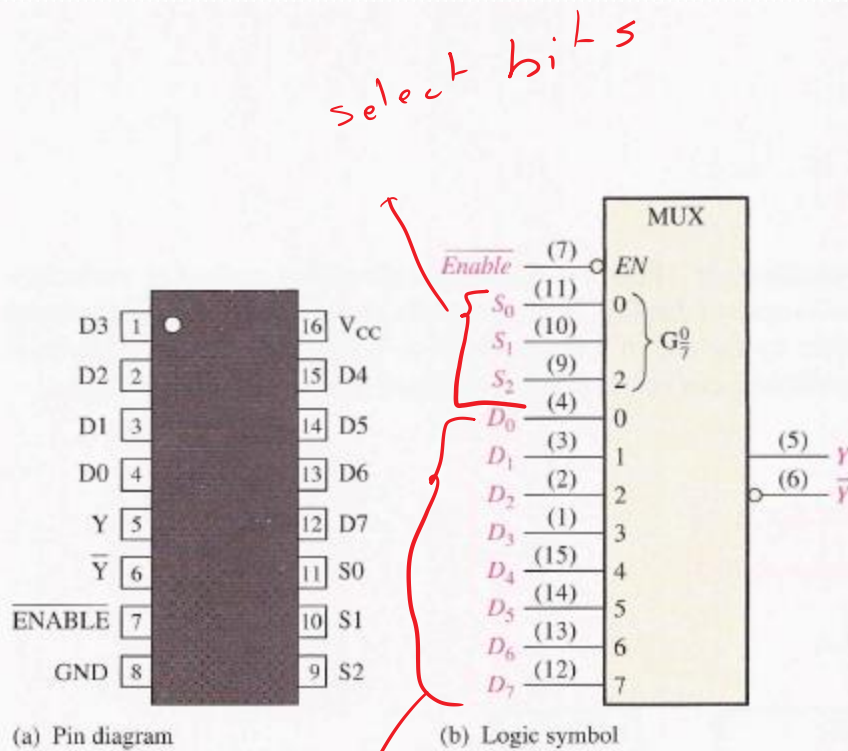


(b) Logic symbol



# Logic Function and Function Combinational

- Example : 74LS151 8-input data selector



MULTIPLEXERS (DATA SELECTC

◀ FIGURE 6-50

Pin diagram and logic symbol for the 74LS151 8-input data selector/multiplexer.

*DATA (1 bit data)*

# Logic Function and Function Combinational

## EXAMPLE 6-15

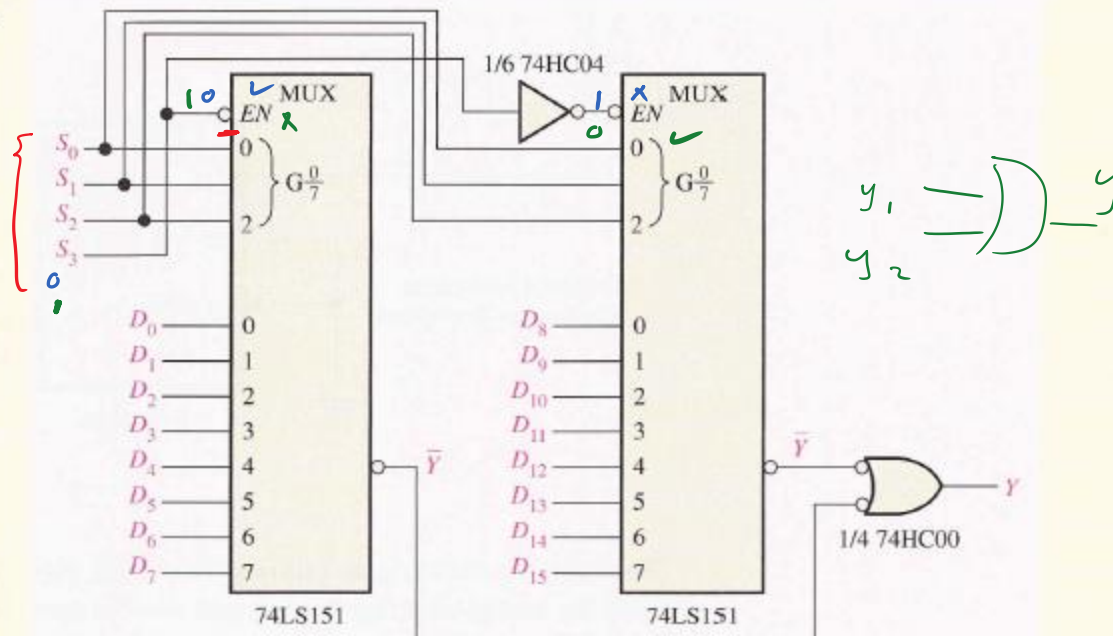
Use 74LS151s and any other logic necessary to multiplex 16 data lines onto a single data-output line.

**Solution** An implementation of this system is shown in Figure 6-51. Four bits are required to select one of 16 data inputs ( $2^4 = 16$ ). In this application the *Enable* input is used as the most significant data-select bit. When the MSB in the data-select code is LOW, the left 74LS151 is enabled, and one of the data inputs ( $D_0$  through  $D_7$ ) is selected by the other three data-select bits. When the data-select MSB is HIGH, the right 74LS151 is enabled, and one of the data inputs ( $D_8$  through  $D_{15}$ ) is selected. The selected input data are then passed through to the negative-OR gate and onto the single output line.

► FIGURE 6-51

A 16-input multiplexer.

$S_3 \ S_2 \ S_1 \ S_0$   
 - - - -  
 0 x x x (0-7)  
 1 x x x (8-15)

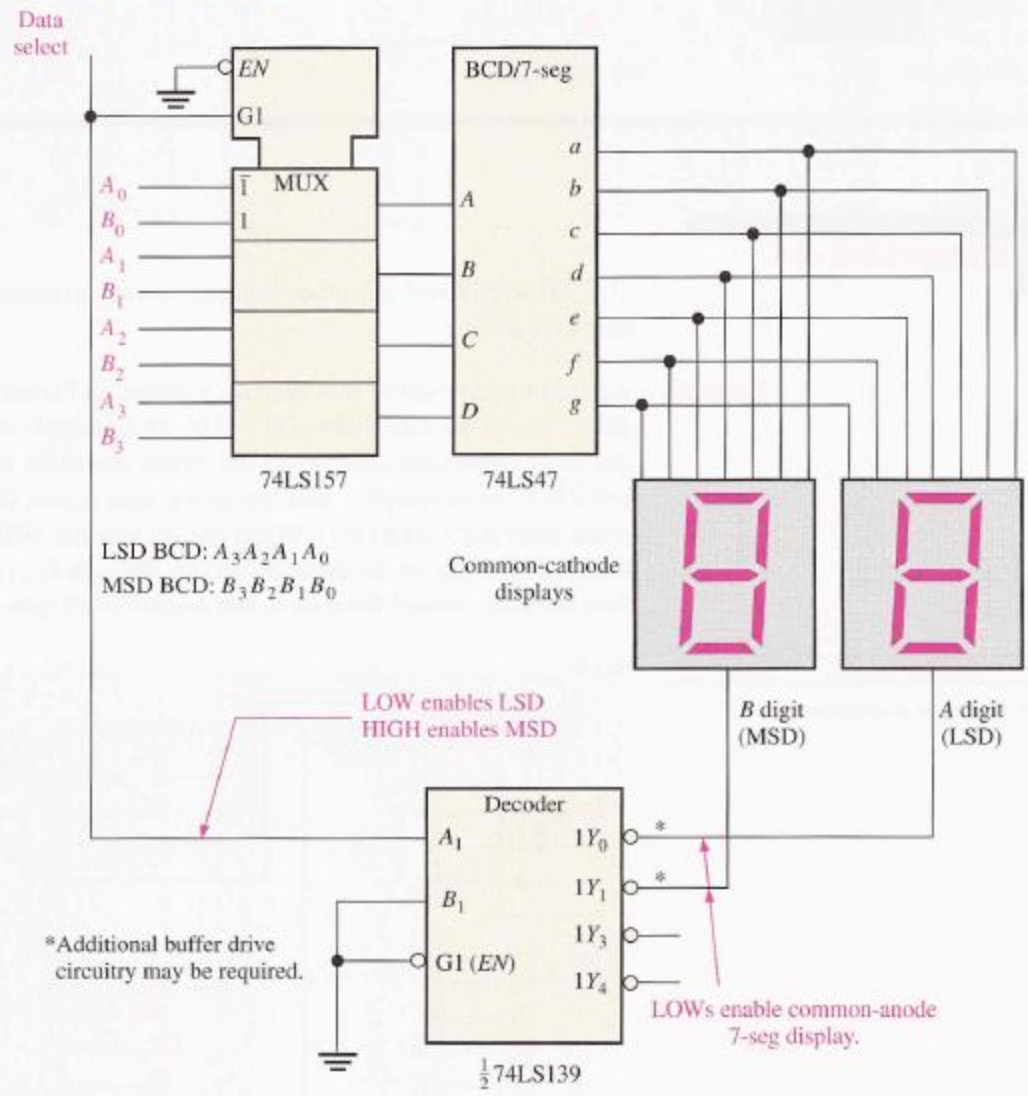


► **FIGURE 6-52**

Simplified 7-segment display multiplexing logic.

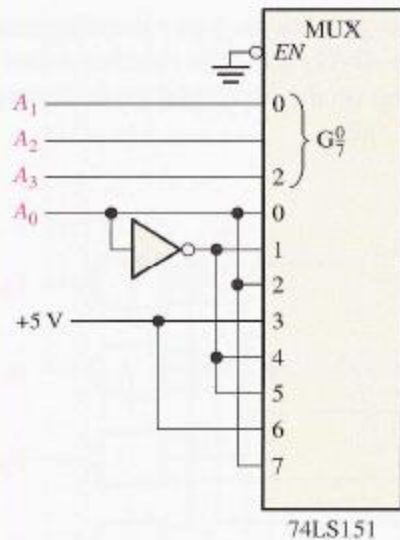


A



# Logic Function and Function Combinational

## Application



$$Y = \bar{A}_3\bar{A}_2\bar{A}_1A_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0 + \bar{A}_3A_2\bar{A}_1A_0 + \bar{A}_3A_2A_1\bar{A}_0 + \bar{A}_3A_2A_1A_0 + A_3\bar{A}_2\bar{A}_1\bar{A}_0 + A_3\bar{A}_2\bar{A}_1A_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3\bar{A}_2A_1A_0 + A_3A_2\bar{A}_1\bar{A}_0 + A_3A_2\bar{A}_1A_0 + A_3A_2A_1\bar{A}_0 + A_3A_2A_1A_0$$

| DECIMAL<br>DIGIT | INPUTS         |                |                |                | OUTPUT<br>Y |
|------------------|----------------|----------------|----------------|----------------|-------------|
|                  | A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> | A <sub>0</sub> |             |
| 0                | 0              | 0              | 0              | 0              | 0           |
| 1                | 0              | 0              | 0              | 1              | 1           |
| 2                | 0              | 0              | 1              | 0              | 1           |
| 3                | 0              | 0              | 1              | 1              | 0           |
| 4                | 0              | 1              | 0              | 0              | 0           |
| 5                | 0              | 1              | 0              | 1              | 1           |
| 6                | 0              | 1              | 1              | 0              | 1           |
| 7                | 0              | 1              | 1              | 1              | 1           |
| 8                | 1              | 0              | 0              | 0              | 1           |
| 9                | 1              | 0              | 0              | 1              | 0           |
| 10               | 1              | 0              | 1              | 0              | 1           |
| 11               | 1              | 0              | 1              | 1              | 0           |
| 12               | 1              | 1              | 0              | 0              | 1           |
| 13               | 1              | 1              | 0              | 1              | 1           |
| 14               | 1              | 1              | 1              | 0              | 0           |
| 15               | 1              | 1              | 1              | 1              | 1           |

# Logic Function and Function

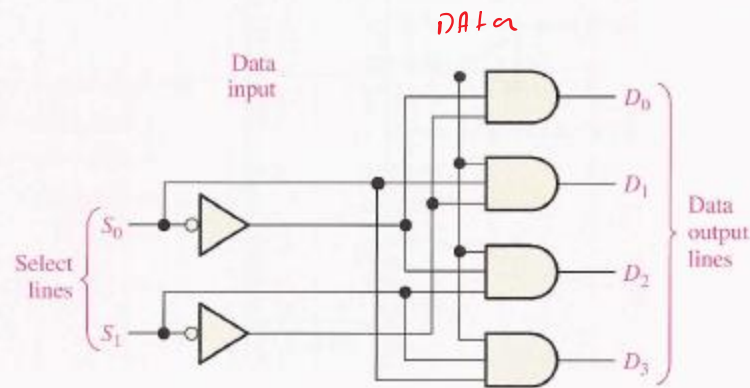
## Combinational

Demultiplexer (DEMUX) : It takes digital information from one line and distributes it to a given no. of out put lines.

Example:

► FIGURE 6-55

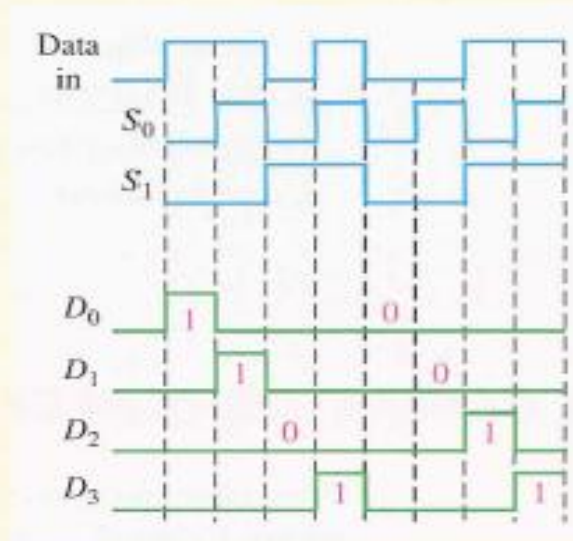
A 1-line-to-4-line demultiplexer.



# Logic Function and Function Combinational

Example:

► FIGURE 6-56



**Solution** Notice that the select lines go through a binary sequence so that each successive input bit is routed to  $D_0$ ,  $D_1$ ,  $D_2$ , and  $D_3$  in sequence, as shown by the output waveforms in Figure 6-56.

# Logic Function and Function Combinational

Example: 74HC154

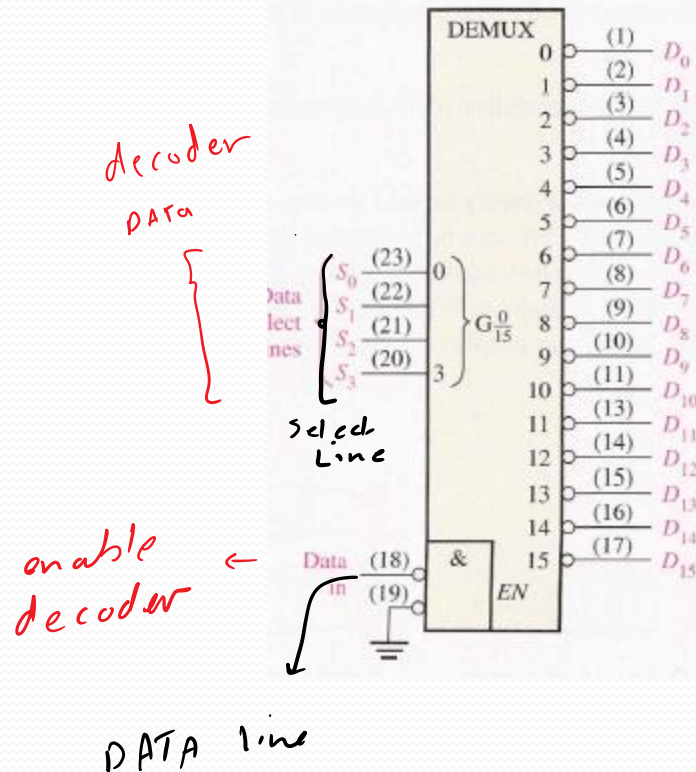


FIGURE 6-57

The 74HC154 decoder used as a demultiplexer.

# Logic Function and Function

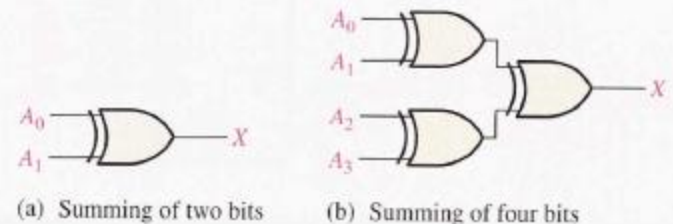
## Combinational

- Parity generator/checker:
- Parity : is the number of 1's in digital information either even or odd.
- Used to detect the error in transmission.

Basic parity logic :

- In this circuit the out put 1 if the parity is odd  
0 if the parity is even.

FIGURE 6-58





# Logic Function and Function Combinational

A  $\longrightarrow$  B

D = 10010

even no. of ones

D = 10101

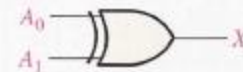
odd no. of ones.

D = 100110110  
odd parity

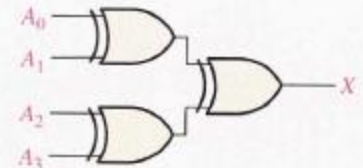
parity bit

0

FIGURE 6-58



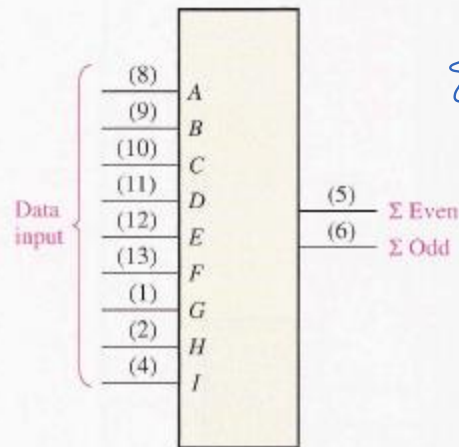
(a) Summing of two bits



(b) Summing of four bits

# Logic Function and Function Combinational

## Example 74LS280



(a) Traditional logic symbol

*Σ even*  $\int$  if even *output 1*  
 $\int$  if odd *output 0*

*Σ odd*  $\int$  if odd *output 1*  
 $\int$  if even *output 0*

| Number of Inputs<br>A-I That Are HIGH | Outputs |       |
|---------------------------------------|---------|-------|
|                                       | Σ Even  | Σ Odd |
| 0, 2, 4, 6, 8                         | H       | L     |
| 1, 3, 5, 7, 9                         | L       | H     |

(b) Function table

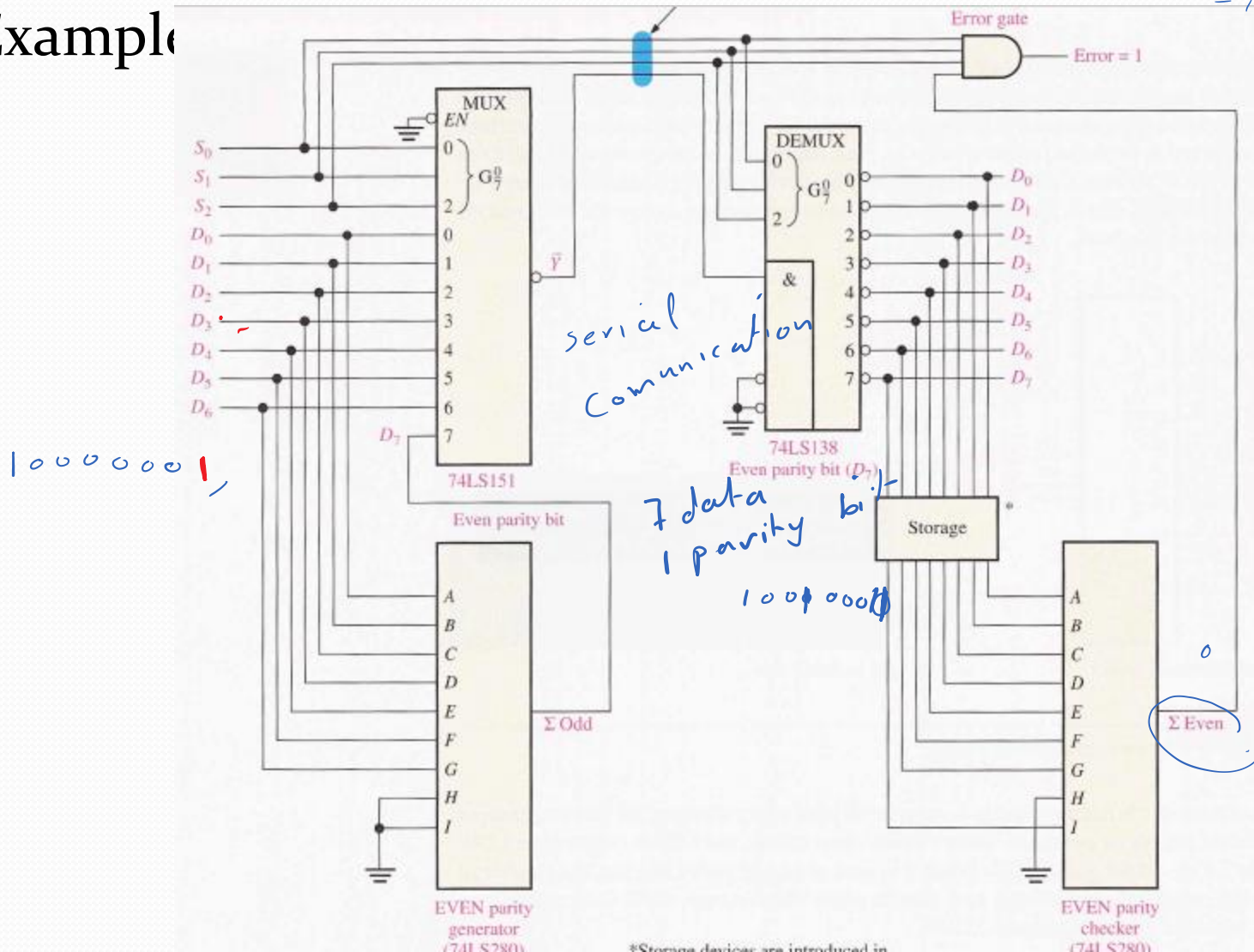
▲ **FIGURE 6-59**

The 74LS280 9-bit parity generator/checker.

# Logic Function and Function Combinational

1  $\Rightarrow$  if- correct  
0  $\Rightarrow$  if- not

Example



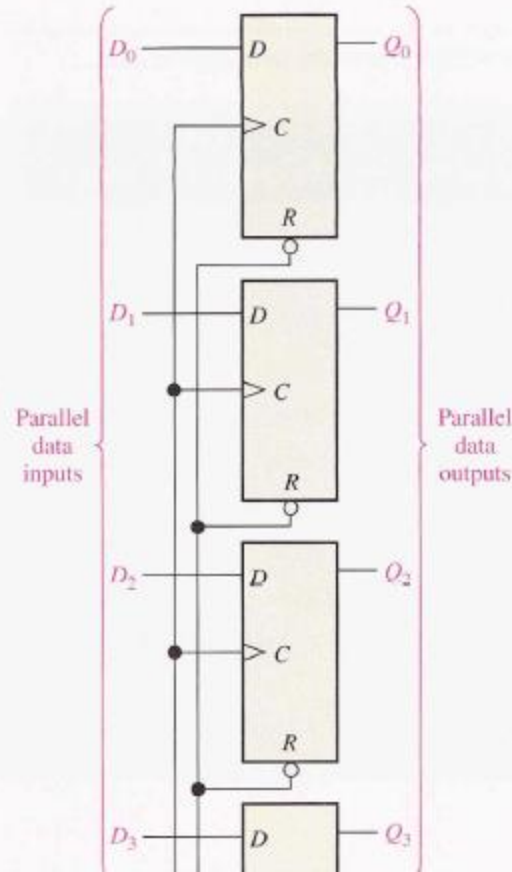
\*Storage devices are introduced in

# Latches , Flip-Flop and timers

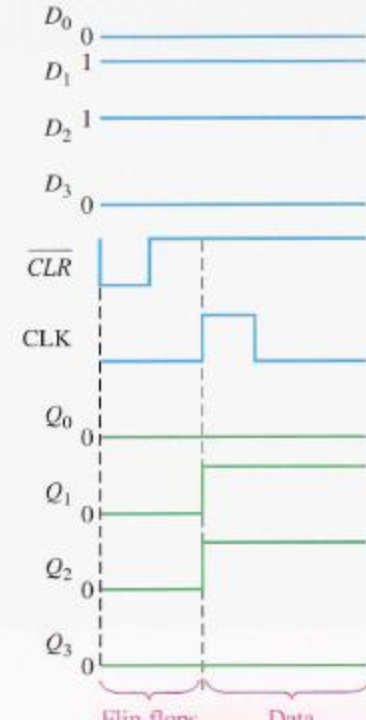
- The Flip-flop application
- Parallel data storage

► FIGURE 7-36

Example of flip-flops used in a basic register for parallel data storage.



Register Memory.



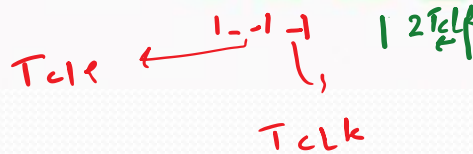
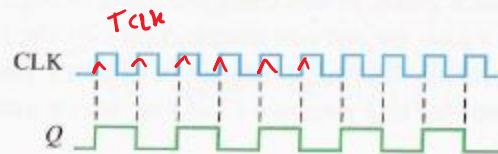
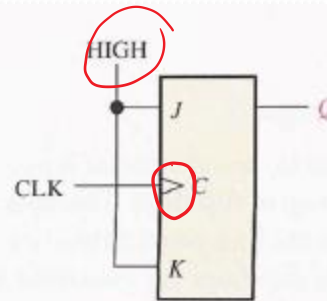
# Latches , Flip-Flop and timers

- The Flip-flop application
- Frequency division

from toggle

► FIGURE 7-37

The J-K flip-flop as a divide-by-2 device. Q is one-half the frequency of CLK.



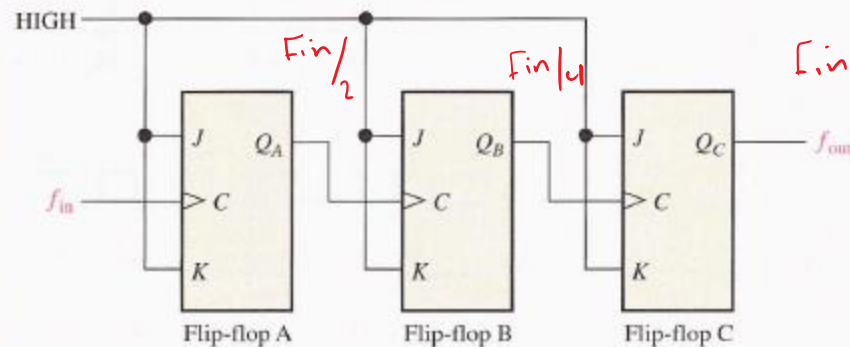
$$F_q = \frac{F_{clk}}{2}$$

# Latches , Flip-Flop and timers

- The Flip-flop application
- Frequency division

## EXAMPLE 7-10

Develop the  $f_{out}$  waveform for the circuit in Figure 7-39 when an 8 kHz square wave input is applied to the clock input of flip-flop A.



▲ FIGURE 7-39

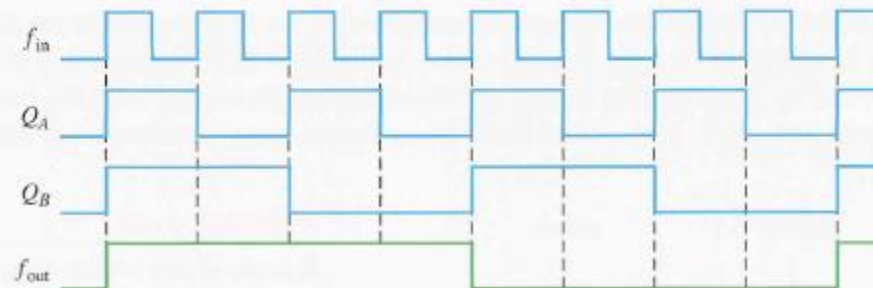
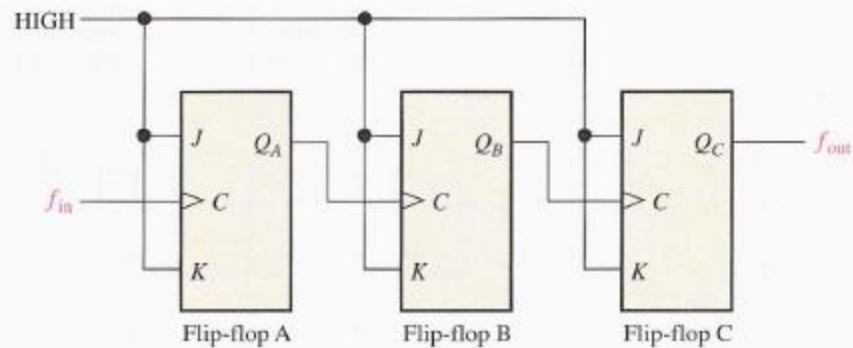
**Solution** The three flip-flops are connected to divide the input frequency by eight ( $2^3 = 8$ ) and the  $f_{out}$  waveform is shown in Figure 7-40. Since these are positive edge-triggered flip-flops, the outputs change on the positive-going clock edge. There is one output pulse for every eight input pulses, so the output frequency is 1 kHz. Waveforms of  $Q_A$  and  $Q_B$  are also shown.

# Latches , Flip-Flop and timers

## Frequency division

### EXAMPLE 7-10

Develop the  $f_{out}$  waveform for the circuit in Figure 7-39 when an 8 kHz square wave input is applied to the clock input of flip-flop A.

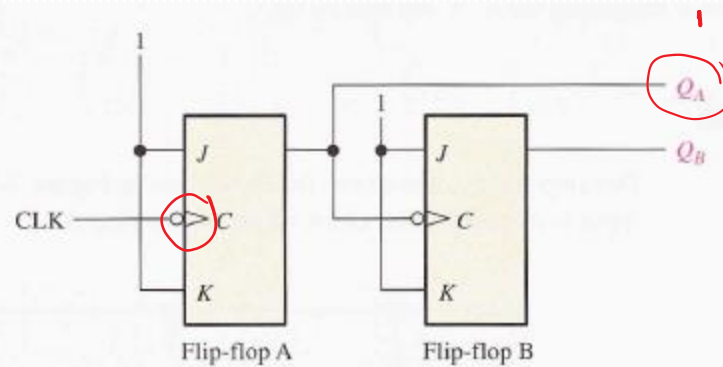


# Latches , Flip-Flop and timers

## Counting

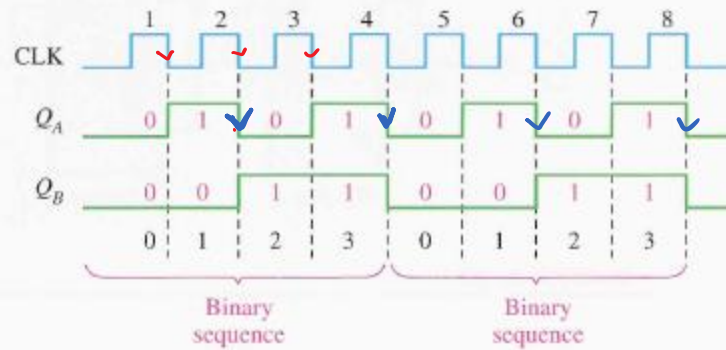
► FIGURE 7-41

Flip-flops used to generate a binary count sequence. Two repetitions (00, 01, 10, 11) are shown.



B A  
0 0  
0 1  
1 0  
1 1

toggle



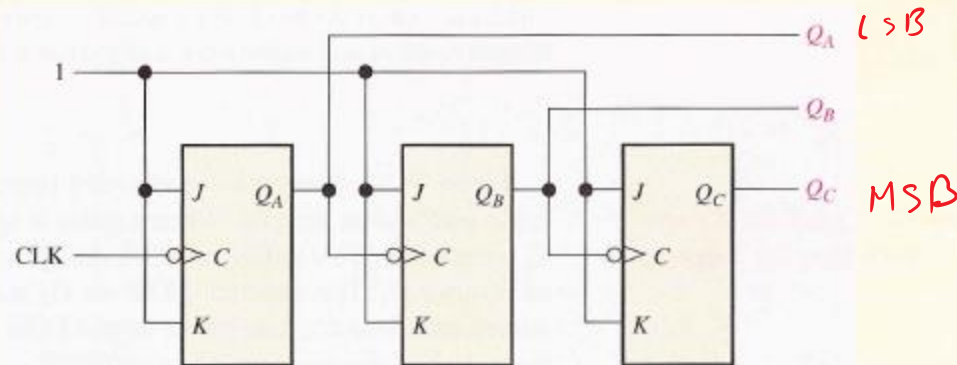


# Latches , Flip-Flop and timers

## Counting

### EXAMPLE 7-11

Determine the output waveforms in relation to the clock for  $Q_A$ ,  $Q_B$ , and  $Q_C$  in the circuit of Figure 7-42 and show the binary sequence represented by these waveforms.



▲ FIGURE 7-42

**Solution** The output timing diagram is shown in Figure 7-43. Notice that the outputs change on the negative-going edge of the clock pulses. The outputs go through the binary sequence 000, 001, 010, 011, 100, 101, 110, and 111 as indicated.

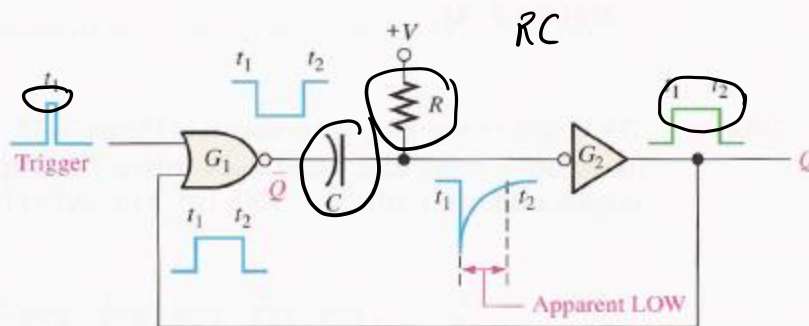


# Latches , Flip-Flop and timers

One-shot :

Is monostable multivibrator

Figure 7-44 shows a basic one-shot (monostable multivibrator) that is composed of a logic gate and an inverter. When a pulse is applied to the **trigger** input, the output of gate  $G_1$  goes LOW. This HIGH-to-LOW transition is coupled through the capacitor to the input of inverter  $G_2$ . The apparent LOW on  $G_2$  makes its output go HIGH. This HIGH is connected back into  $G_1$ , keeping its output LOW. Up to this point the trigger pulse has caused the output of the one-shot,  $Q$ , to go HIGH.

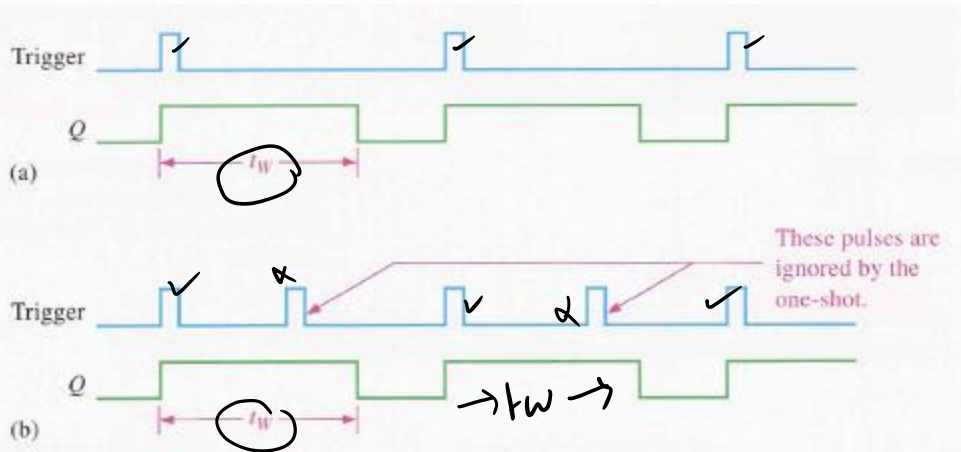


The capacitor immediately begins to charge through  $R$  toward the high voltage level. The rate at which it charges is determined by the  $RC$  time constant. When the capacitor charges to a certain level, which appears as a HIGH to  $G_2$ , the output goes back LOW.

To summarize, the output of inverter  $G_2$  goes HIGH in response to the trigger input. It remains HIGH for a time set by the  $RC$  time constant. At the end of this time, it goes LOW. A single narrow trigger pulse produces a single output pulse whose time duration is controlled by the  $RC$  time constant. This operation is illustrated in Figure 7-44.

# Latches , Flip-Flop and timers

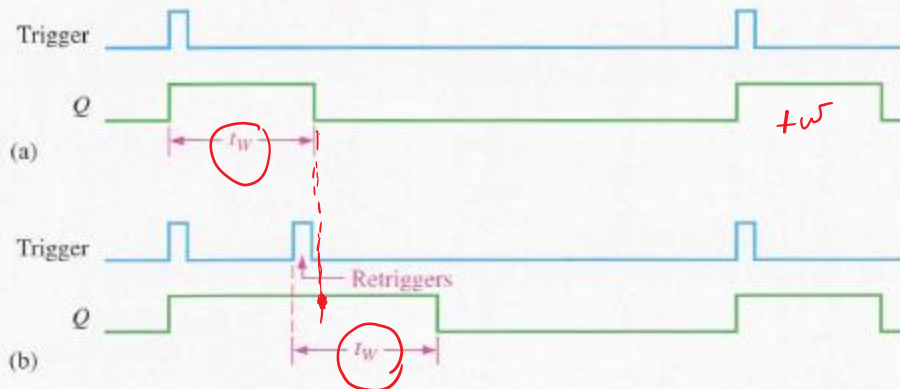
One-shot: Device



non-retriggerable  
one-shot

RC

A retriggerable one-shot can be triggered before it times out. The result of retriggering is an extension of the pulse width as illustrated in Figure 7-47.



# Latches , Flip-Flop and timers

## One-shot :

### THE 74121 NONRETRIGGERABLE ONE-SHOT

The 74121 is an example of a nonretriggerable IC one-shot. It has provisions for external  $R$  and  $C$ , as shown in Figure 7-48. The inputs labeled  $A_1$ ,  $A_2$ , and  $B$  are gated trigger inputs. The  $R_{INT}$  input connects to a  $2\text{ k}\Omega$  internal timing resistor.

**Setting the Pulse Width** A typical pulse width of about 30 ns is produced when no external timing components are used and the internal timing resistor ( $R_{INT}$ ) is connected to  $V_{CC}$ , as shown in Figure 7-49(a). The pulse width can be set anywhere between about 30 ns and 28 s by the use of external components. Figure 7-49(b) shows the configuration using the internal resistor ( $2\text{ k}\Omega$ ) and an external capacitor. Part (c) shows the configuration using an external resistor and an external capacitor. The output pulse width is set by the values of the resistor ( $R_{INT} = 2\text{ k}\Omega$ , and  $R_{EXT}$  is selected) and the capacitor according to the following formula:

$$t_w = 0.7RC_{EXT}$$

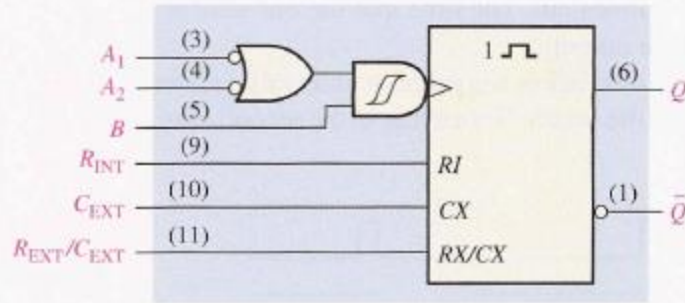
where  $R$  is either  $R_{INT}$  or  $R_{EXT}$ . When  $R$  is in kilohms ( $\text{k}\Omega$ ) and  $C_{EXT}$  is in picofarads (pF), the output pulse width  $t_w$  is in nanoseconds (ns).



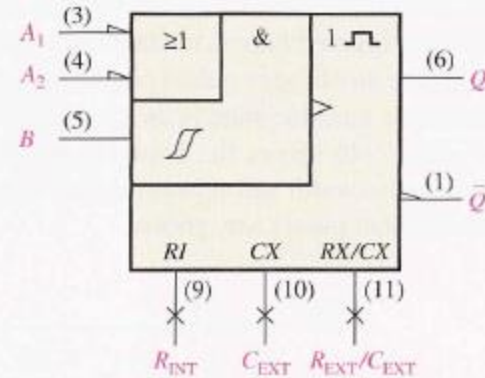
Equation 7-1

# Latches , Flip-Flop and timers

Or



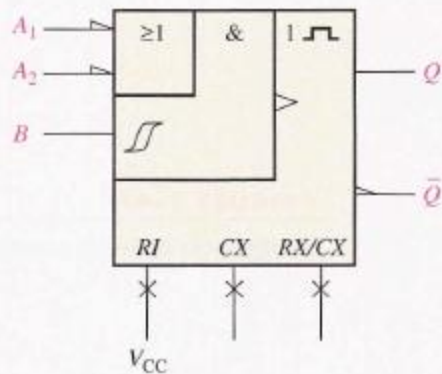
(a) Traditional logic symbol



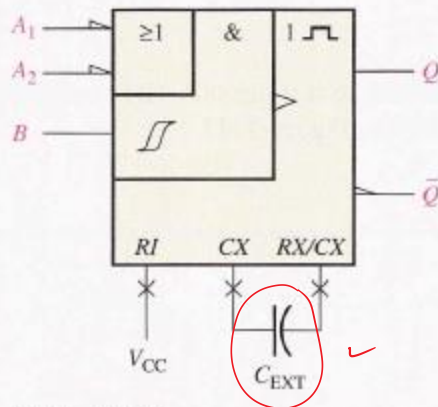
(b) ANSI/IEEE std. 91-1984 logic symbol  
(X = nonlogic connection). "1" is the qualifying symbol for a nonretriggerable one-shot.

▲ FIGURE 7-48

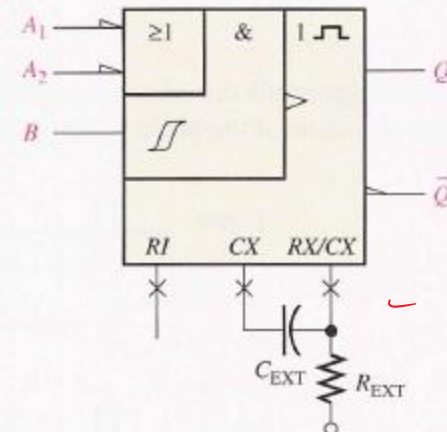
Logic symbols for the 74121 nonretriggerable one-shot.



(a) No external components

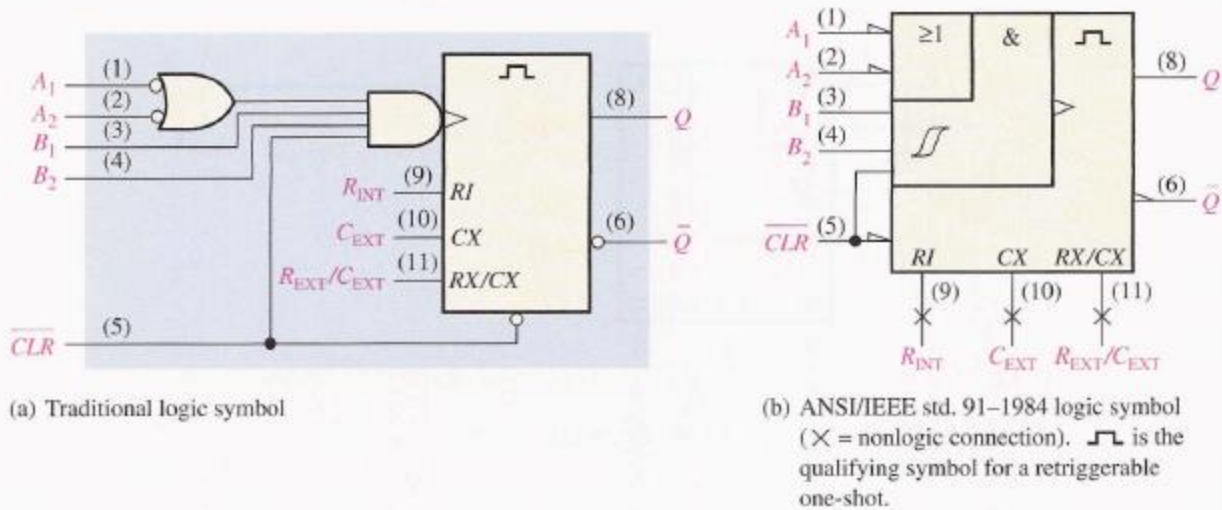


(b)  $R_{INT}$  and  $C_{EXT}$



# latches Flin-Flon and timers

0.



▲ FIGURE 7-50

Logic symbol for the 74LS122 retriggerable one-shot.

A minimum pulse width of approximately 45 ns is obtained with no external components. Wider pulse widths are achieved by using external components. A general formula for calculating the values of these components for a specified pulse width ( $t_w$ ) is

$$t_w = 0.32RC_{EXT} \left( 1 + \frac{0.7}{R} \right) \Rightarrow \text{DATA sheet}$$

where 0.32 is a constant determined by the particular type of one-shot,  $R$  is in  $k\Omega$  and is either the internal or the external resistor,  $C_{EXT}$  is in pF, and  $t_w$  is in ns. The internal resistance is  $10 k\Omega$  and can be used instead of an external resistor. (Notice the difference between this formula and that for the 74121, shown in Equation 7-1.)

Equation 7-2

DATA sheet.

# Latches , Flip-Flop and timers

## One-shot :

### EXAMPLE 7-13

Determine the values of  $R_{EXT}$  and  $C_{EXT}$  that will produce a pulse width of  $1 \mu s$  when connected to a 74LS122.

*Solution* Assume a value of  $C_{EXT} = 560 \text{ pF}$  and then solve for  $R_{EXT}$ . The pulse width must be expressed in ns and  $C_{EXT}$  in pF.  $R_{EXT}$  will be in  $k\Omega$ .

$$\begin{aligned}t_w &= 0.32R_{EXT}C_{EXT}\left(1 + \frac{0.7}{R_{EXT}}\right) = 0.32R_{EXT}C_{EXT} + 0.7\left(\frac{0.32R_{EXT}C_{EXT}}{R_{EXT}}\right) \\&= 0.32R_{EXT}C_{EXT} + (0.7)(0.32)C_{EXT} \\R_{EXT} &= \frac{t_w - (0.7)(0.32)C_{EXT}}{0.32C_{EXT}} = \frac{t_w}{0.32C_{EXT}} - 0.7 \\&= \frac{1000 \text{ ns}}{(0.32)560 \text{ pF}} - 0.7 = 4.88 \text{ k}\Omega\end{aligned}$$

Use a standard value of  $4.7 \text{ k}\Omega$ .

*Related Problem* Show the connections and component values for a 74LS122 one-shot with an output pulse width of  $5 \mu s$ . Assume  $C_{EXT} = 560 \text{ pF}$ .

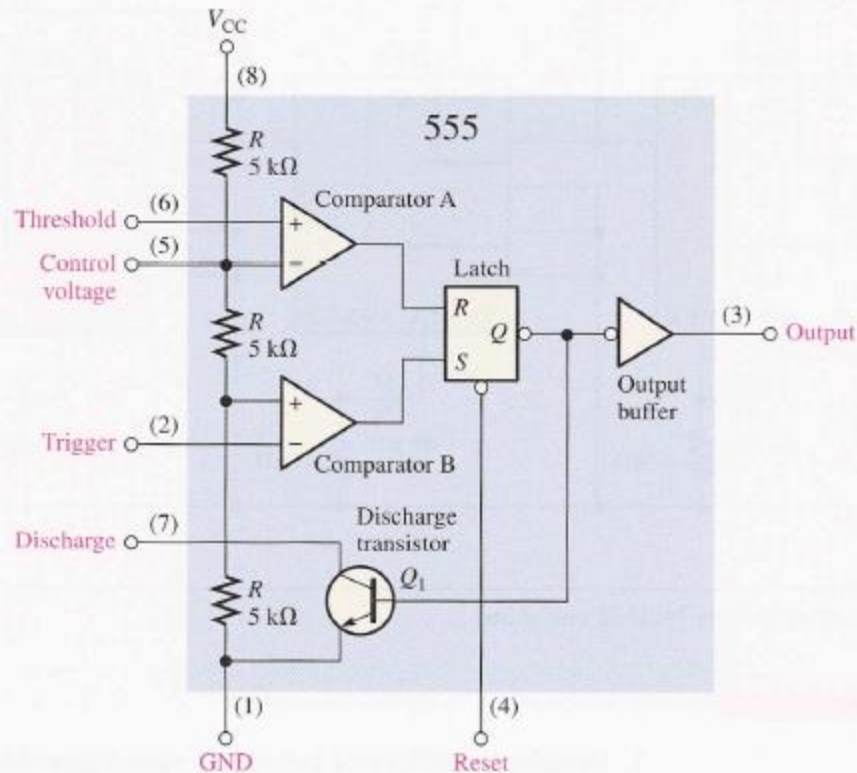
# Latches , Flip-Flop and timers

Timer 555:

Is a device can be used as either mono-stable multi-vibrator or as an stable multi-vibrator (oscillator).

► **FIGURE 7-53**

Internal functional diagram of a 555 timer (pin numbers are in parenthesis).



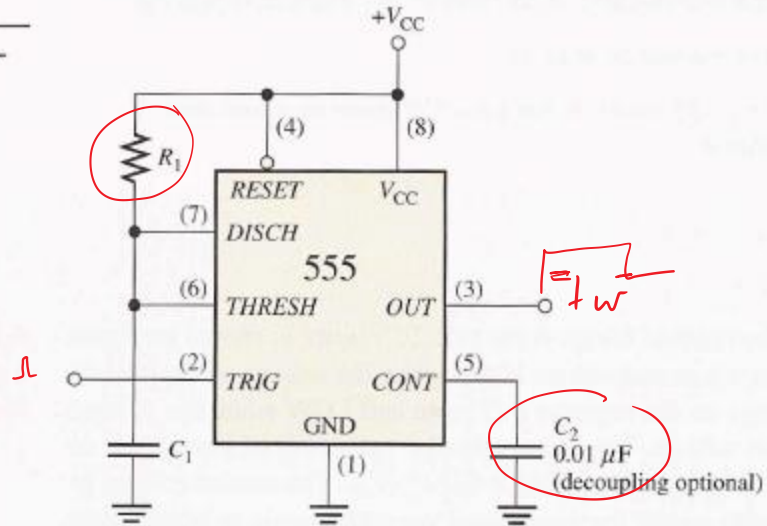


# Latches , Flip-Flop and timers

## Timer 555:

► FIGURE 7-54

The 555 timer connected as a one-shot.



### EXAMPLE 7-14

What is the output pulse width for a 555 monostable circuit with  $R_1 = 2.2 \text{ k}\Omega$  and  $C_1 = 0.01 \text{ }\mu\text{F}$ ?

DATA sheet Solution

From Equation 7-3 the pulse width is

$$t_w = 1.1R_1C_1 = 1.1(2.2 \text{ k}\Omega)(0.01 \text{ }\mu\text{F}) = 24.2 \text{ }\mu\text{s}$$

Related Problem

For  $C_1 = 0.01 \text{ }\mu\text{F}$ , determine the value of  $R_1$  for a pulse width of 1 ms.

# Latches , Flip-Flop and timers

## Timer 555:

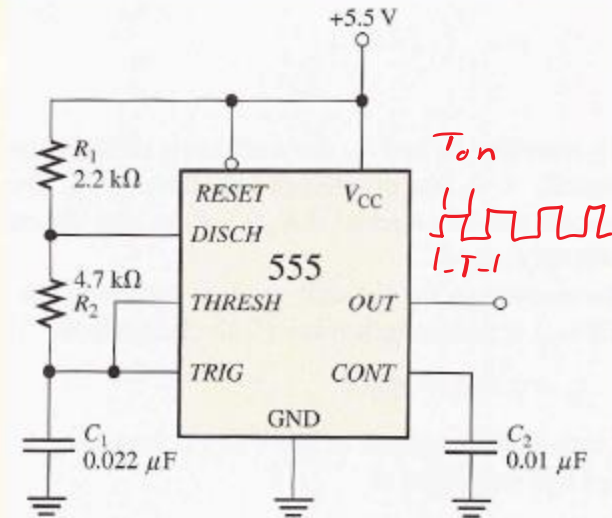
### EXAMPLE 7-15

A 555 timer configured to run in the astable mode (oscillator) is shown in Figure 7-60. Determine the frequency of the output and the duty cycle.



► **FIGURE 7-60**

Open file F07-60 to verify operation.



**Solution** Use Equations 7-4 and 7-7.

$$f = \frac{1.44}{(R_1 + 2R_2)C_1} = \frac{1.44}{(2.2 \text{ k}\Omega + 9.4 \text{ k}\Omega)0.022 \text{ }\mu\text{F}} = 5.64 \text{ kHz}$$

$$\text{Duty cycle} = \left( \frac{R_1 + R_2}{R_1 + 2R_2} \right) 100\% = \left( \frac{2.2 \text{ k}\Omega + 4.7 \text{ k}\Omega}{2.2 \text{ k}\Omega + 9.4 \text{ k}\Omega} \right) 100\% = 59.5\%$$

**Related Problem** Determine the duty cycle in Figure 7-60 if a diode is connected across  $R_2$  as indicated in Figure 7-59.

$\tau \Rightarrow RC$   
Duty cycle  
 $\frac{T_{ON}}{T} \neq 100\%$

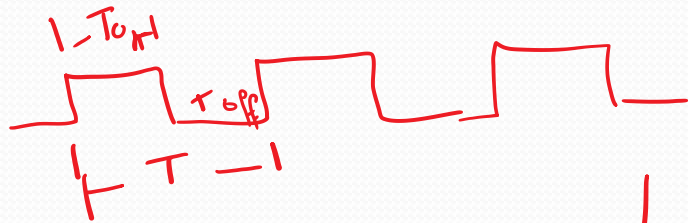
$2 \times 2.7$   
2.1

56%

constant.

$T_{ON}$   
1-T-1

$$\text{Duty cycle} = \frac{T_{\text{on}}}{T} \times 100\%$$



50% duty cycle

$$T = 1 \text{ sec}$$

$$T_{\text{on}} = 0.5 \text{ sec.}$$

ex: 60%

$$T = 1 \text{ sec}$$

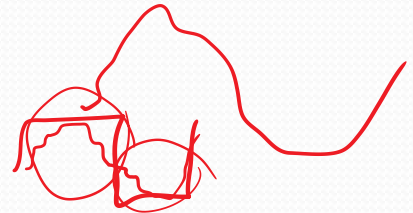
$$T_{\text{on}} = 0.6 \text{ sec}$$

PWM  
pulse width  
modulation

⇒ Control (Motor)

⇒ Inverter

DC ⇒ AC



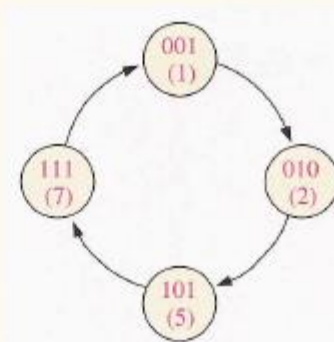
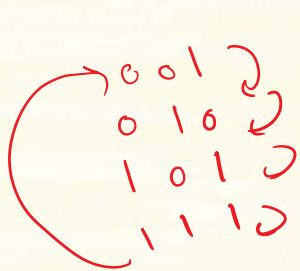
# Sequential logic system design Ch,8

# Sequential logic system design

## EXAMPLE 8-5

Design a counter with the irregular binary count sequence shown in the state diagram of Figure 8-32. Use J-K flip-flops.

► FIGURE 8-32



3 J-K Flip flop  
=> state diagram

**Solution Step 1:** The state diagram is as shown. Although there are only four states, a 3-bit counter is required to implement this sequence because the maximum binary count is seven. Since the required sequence does not include all the possible binary states, the invalid states (0, 3, 4, and 6) can be treated as "don't cares" in the design. However, if the counter should erroneously get into an invalid state, you must make sure that it goes back to a valid state.

**Step 2:** The next-state table is developed from the state diagram and is given in Table 8-9.

# Sequential logic system design

**Step 2:** The next-state table is developed from the state diagram and is given in Table 8-9.

**TABLE 8-9**

Next-state table.

| PRESENT STATE |       |       | NEXT STATE |       |       |
|---------------|-------|-------|------------|-------|-------|
| $Q_2$         | $Q_1$ | $Q_0$ | $Q_2$      | $Q_1$ | $Q_0$ |
| 0             | 0     | 1     | 0          | 1     | 0     |
| 0             | 1     | 0     | 1          | 0     | 1     |
| 1             | 0     | 1     | 1          | 1     | 1     |
| 1             | 1     | 1     | 0          | 0     | 1     |

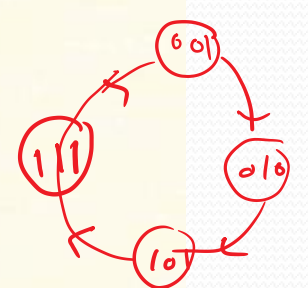
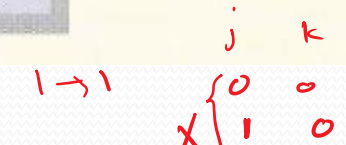
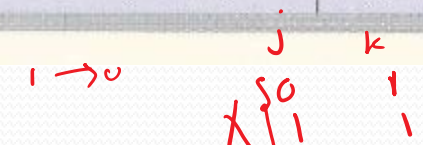
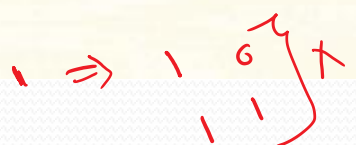
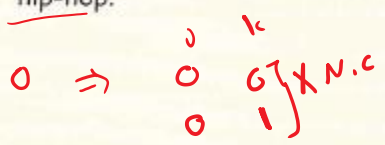


**Step 3:** The transition table for the J-K flip-flop is repeated in Table 8-10.

**TABLE 8-10**

Transition table for a J-K flip-flop.

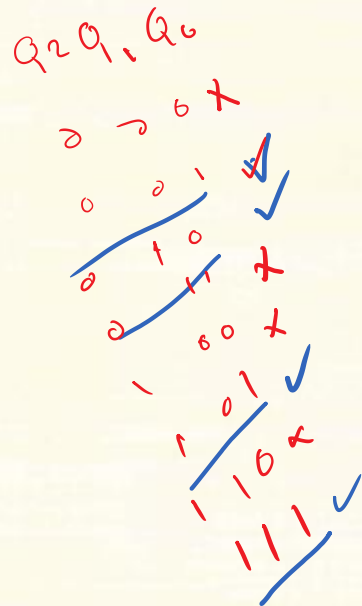
| OUTPUT TRANSITIONS |           | FLIP-FLOP INPUTS |   |
|--------------------|-----------|------------------|---|
| $Q_N$              | $Q_{N+1}$ | J                | K |
| 0                  | → 0       | 0                | X |
| 0                  | → 1       | 1                | X |
| 1                  | → 0       | X                | 1 |
| 1                  | → 1       | X                | 0 |



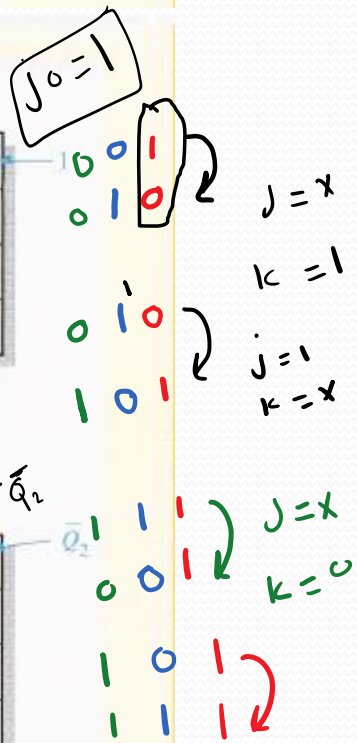
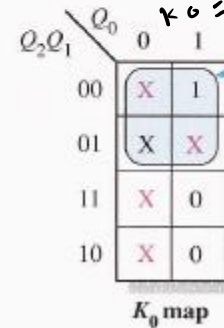
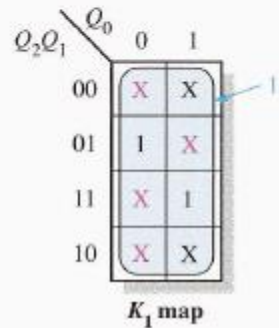
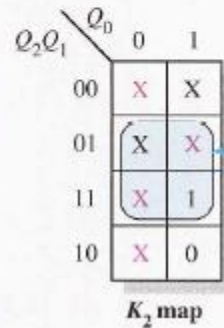
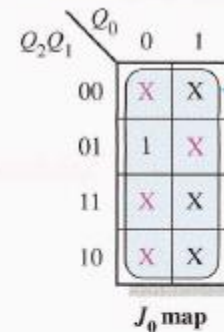
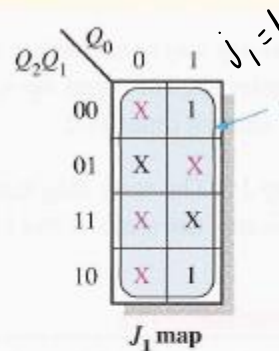
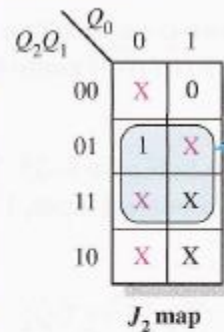
# Sequential logic system design

**Step 4:** The  $J$  and  $K$  inputs are plotted on the present-state Karnaugh maps in Figure 8-33. Also "don't cares" can be placed in the cells corresponding to the invalid states of 000, 011, 100, and 110, as indicated by the red Xs.

► FIGURE 8-33



$j_2 = Q_1$



# Sequential logic system design

**Step 5:** Group the 1s, taking advantage of as many of the “don’t care” states as possible for maximum simplification, as shown in Figure 8–33. Notice that when *all* cells in a map are grouped, the expression is simply equal to 1. The expression for each *J* and *K* input taken from the maps is as follows:

$$J_0 = 1, K_0 = \overline{Q_2}$$

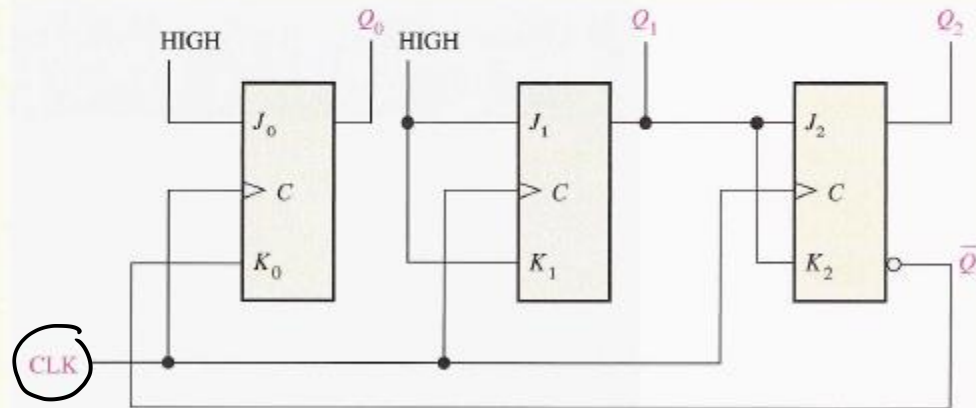
$$J_1 = K_1 = 1$$

$$J_2 = K_2 = Q_1$$

**Step 6:** The implementation of the counter is shown in Figure 8–34.

► FIGURE 8–34

Handwritten notes:  
Synchronous Counter (irregular)



An analysis shows that if the counter, by accident, gets into one of the invalid states (0, 3, 4, 6), it will always return to a valid state according to the following sequences:  $0 \rightarrow 3 \rightarrow 4 \rightarrow 7$ , and  $6 \rightarrow 1$ .



# Sequential logic system design

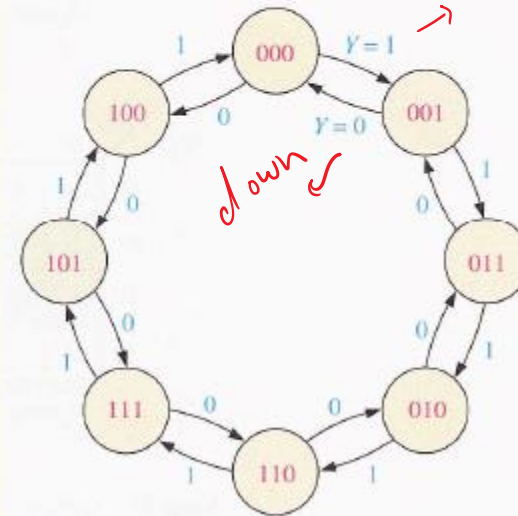
## EXAMPLE 8-6

Develop a synchronous 3-bit up/down counter with a Gray code sequence. The counter should count up when an UP/DOWN control input is 1 and count down when the control input is 0.

**Solution** **Step 1:** The state diagram is shown in Figure 8-35. The 1 or 0 beside each arrow indicates the state of the UP/DOWN control input,  $Y$ .

► **FIGURE 8-35**

State diagram for a 3-bit up/down Gray code counter.



# J-K Flip flop  
→ 3 J-K Flip flops  
3 - J-K " " the  
Synch } Same clock

# Sequential logic system design

**Step 2:** The next-state table is derived from the state diagram and is shown in Table 8-11. Notice that for each present state there are two possible next states, depending on the UP/DOWN control variable,  $Y$ .

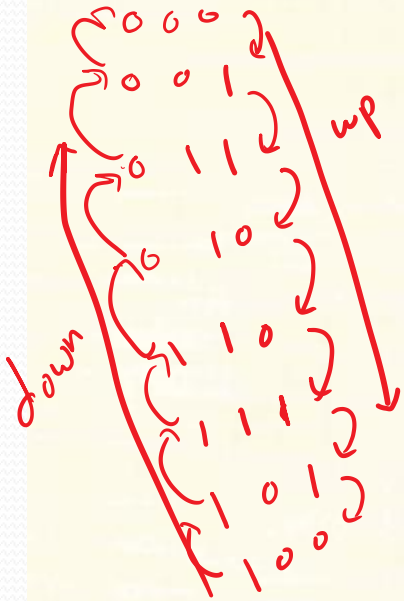
▼ **TABLE 8-11**

Next-state table for 3-bit up/down Gray code counter.

| PRESENT STATE |       |       | NEXT STATE     |       |       |              |       |       |
|---------------|-------|-------|----------------|-------|-------|--------------|-------|-------|
|               |       |       | $Y = 0$ (DOWN) |       |       | $Y = 1$ (UP) |       |       |
| $Q_2$         | $Q_1$ | $Q_0$ | $Q_2$          | $Q_1$ | $Q_0$ | $Q_2$        | $Q_1$ | $Q_0$ |
| 0             | 0     | 0     | 1              | 0     | 0     | 0            | 0     | 1     |
| 0             | 0     | 1     | 0              | 0     | 0     | 0            | 1     | 1     |
| 0             | 1     | 1     | 0              | 0     | 1     | 0            | 1     | 0     |
| 0             | 1     | 0     | 0              | 1     | 1     | 1            | 1     | 0     |
| 1             | 1     | 0     | 0              | 1     | 0     | 1            | 1     | 1     |
| 1             | 1     | 1     | 1              | 1     | 0     | 1            | 0     | 1     |
| 1             | 0     | 1     | 1              | 1     | 1     | 1            | 0     | 0     |
| 1             | 0     | 0     | 1              | 0     | 1     | 0            | 0     | 0     |

$Y = \overline{\text{UP/DOWN}}$  control input.

**Step 3:** The transition table for the J-K flip-flops is repeated in Table 8-12.



# Sequential logic system design

Step 3: The transition table for the J-K flip-flops is repeated in Table 8-12.

▶ TABLE 8-12

Transition table for a J-K flip-flop.

| OUTPUT TRANSITIONS |   |           | FLIP-FLOP INPUTS |   |
|--------------------|---|-----------|------------------|---|
| $Q_N$              |   | $Q_{N-1}$ | J                | K |
| 0                  | → | 0         | 0                | X |
| 0                  | → | 1         | 1                | X |
| 1                  | → | 0         | X                | 1 |
| 1                  | → | 1         | X                | 0 |



$0 \rightarrow 0$

N.ch  
 $\left. \begin{matrix} j=0 \\ \underline{0} \end{matrix} \right\} k=1$   
 $\left. \begin{matrix} 0 \\ \underline{0} \end{matrix} \right\} 0$

$1 \rightarrow 0$

toggle  
 $\left. \begin{matrix} j=0 \\ \uparrow \\ X \end{matrix} \right\} k=1$

$1 \rightarrow 1$

N.ch  
 $X \left\{ \begin{matrix} j=0 \\ \uparrow \\ j=1 \end{matrix} \right. \left. \begin{matrix} k=0 \\ \uparrow \\ k=0 \end{matrix} \right\}$

$0 \rightarrow 1$

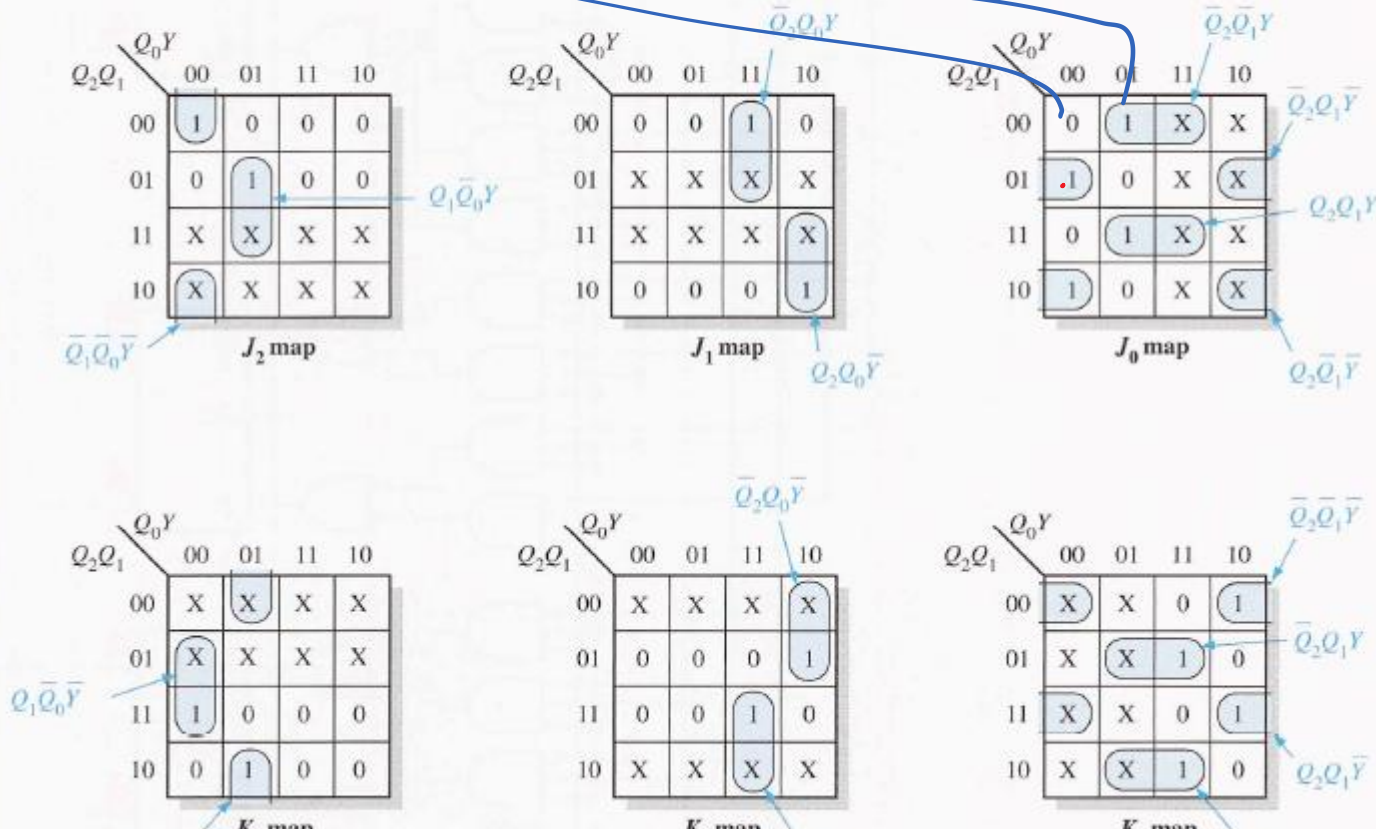
toggle  
 $\left. \begin{matrix} j=1 \\ \underline{1} \\ = \end{matrix} \right\} k=0$

# Sequential logic system design

$Q_2$   $Q_1$   $Q_0$   $Y$   
 0 0 0 0  
 0 0 0 1  
 0 0 1 0  
 0 0 1 1  
 0 1 0 0  
 0 1 0 1  
 0 1 1 0  
 0 1 1 1  
 1 0 0 0  
 1 0 0 1  
 1 0 1 0  
 1 0 1 1  
 1 1 0 0  
 1 1 0 1  
 1 1 1 0  
 1 1 1 1

down  
 up

**Step 4:** The Karnaugh maps for the  $J$  and  $K$  inputs of the flip-flops are shown in Figure 8-36. The UP/DOWN control input,  $Y$ , is considered one of the state variables along with  $Q_0$ ,  $Q_1$ , and  $Q_2$ . Using the next-state table, the information in the "Flip-Flop Inputs" column of Table 8-12 is transferred onto the maps as indicated for each present state of the counter.



down  
 0 0 0 0  
 0 0 0 1  
 0 1 0 0  
 0 1 0 1  
 0 1 1 0  
 0 1 1 1  
 1 0 0 0  
 1 0 0 1  
 1 0 1 0  
 1 0 1 1  
 1 1 0 0  
 1 1 0 1  
 1 1 1 0  
 1 1 1 1

up  
 down

# Sequential logic system design

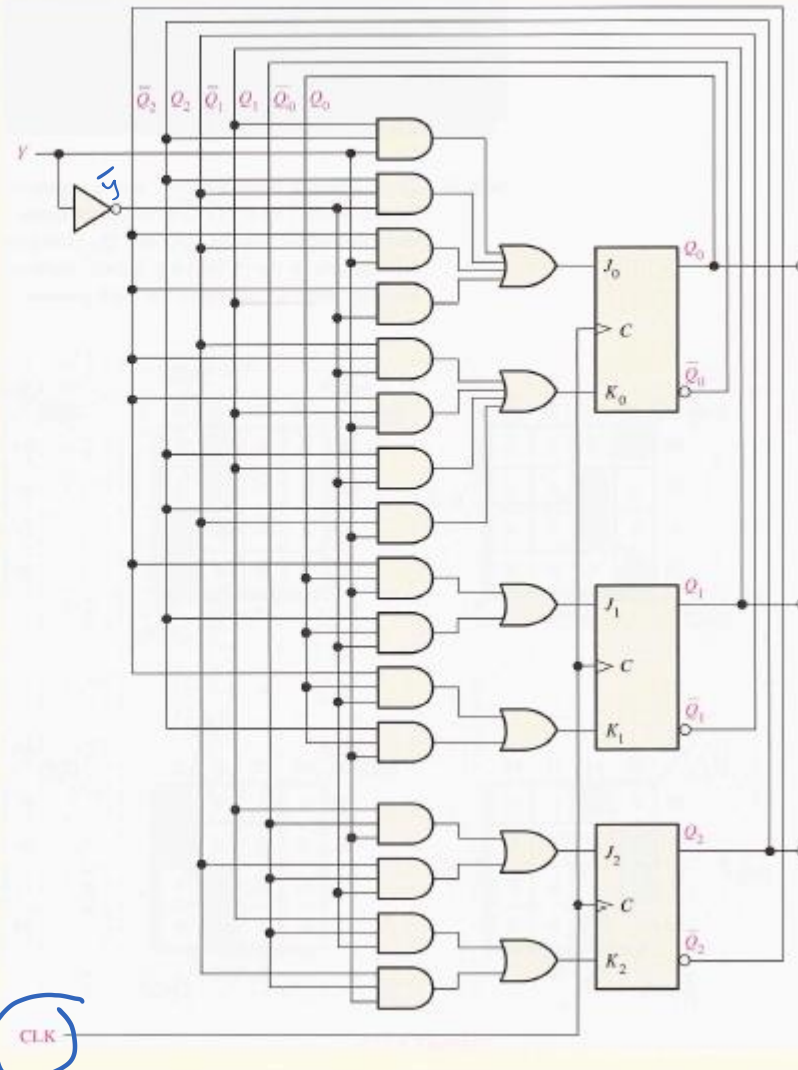
**Step 5:** The 1s are combined in the largest possible groupings, with “don’t cares” (Xs) used where possible. The groups are factored, and the expressions for the  $J$  and  $K$  inputs are as follows:

$$\begin{aligned} J_0 &= Q_2 Q_1 Y + Q_2 \bar{Q}_1 \bar{Y} + \bar{Q}_2 \bar{Q}_1 Y + \bar{Q}_2 Q_1 \bar{Y} & K_0 &= \bar{Q}_2 \bar{Q}_1 \bar{Y} + \bar{Q}_2 Q_1 Y + Q_2 \bar{Q}_1 Y + Q_2 Q_1 \bar{Y} \\ J_1 &= \bar{Q}_2 Q_0 Y + Q_2 Q_0 \bar{Y} & K_1 &= \bar{Q}_2 Q_0 \bar{Y} + Q_2 Q_0 Y \\ J_2 &= Q_1 \bar{Q}_0 Y + \bar{Q}_1 \bar{Q}_0 \bar{Y} & K_2 &= Q_1 \bar{Q}_0 \bar{Y} + \bar{Q}_1 \bar{Q}_0 Y \end{aligned}$$

Se

**Step 5:** The 1s are combined in the largest possible groupings, with “don’t cares” (Xs) used where possible. The groups are factored, and the expressions for the  $J$  and  $K$  inputs are as follows:

$$\begin{aligned} J_0 &= Q_2 Q_1 Y + Q_2 \bar{Q}_1 \bar{Y} + \bar{Q}_2 \bar{Q}_1 Y + \bar{Q}_2 Q_1 \bar{Y} & K_0 &= \bar{Q}_2 \bar{Q}_1 \bar{Y} + \bar{Q}_2 Q_1 Y + Q_2 \bar{Q}_1 Y + Q_2 Q_1 \bar{Y} \\ J_1 &= \bar{Q}_2 Q_0 Y + Q_2 Q_0 \bar{Y} & K_1 &= \bar{Q}_2 Q_0 \bar{Y} + Q_2 Q_0 Y \\ J_2 &= Q_1 \bar{Q}_0 Y + \bar{Q}_1 \bar{Q}_0 \bar{Y} & K_2 &= Q_1 \bar{Q}_0 \bar{Y} + \bar{Q}_1 \bar{Q}_0 Y \end{aligned}$$



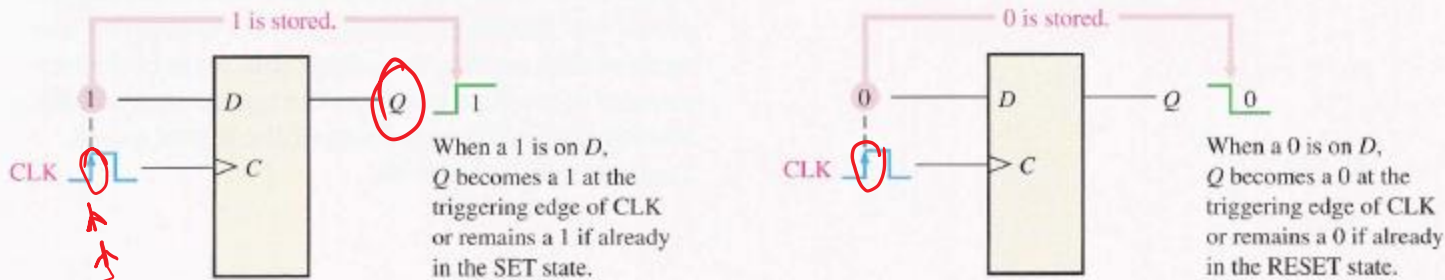
synch

# Shift Register

## Ch,9

# Shift Register

- A register is digital circuit with two basic function: data storage and data movement. Usually D-flip flop is used
- Data storage example



▲ FIGURE 9-1

The flip-flop as a storage element

$$X = \begin{matrix} Q_2 & Q_1 & Q_0 \\ 0 & 0 & 1 \end{matrix}$$

$$\begin{matrix} X_P = 100 \\ X_N = 001 \end{matrix}$$

$$\begin{matrix} = 4_{10} \\ = 2_{10} \\ = 1_{10} \end{matrix} \left. \begin{matrix} \\ \\ \end{matrix} \right\} \begin{matrix} \text{shift } R \\ \text{1 bit } X/2 \end{matrix}$$



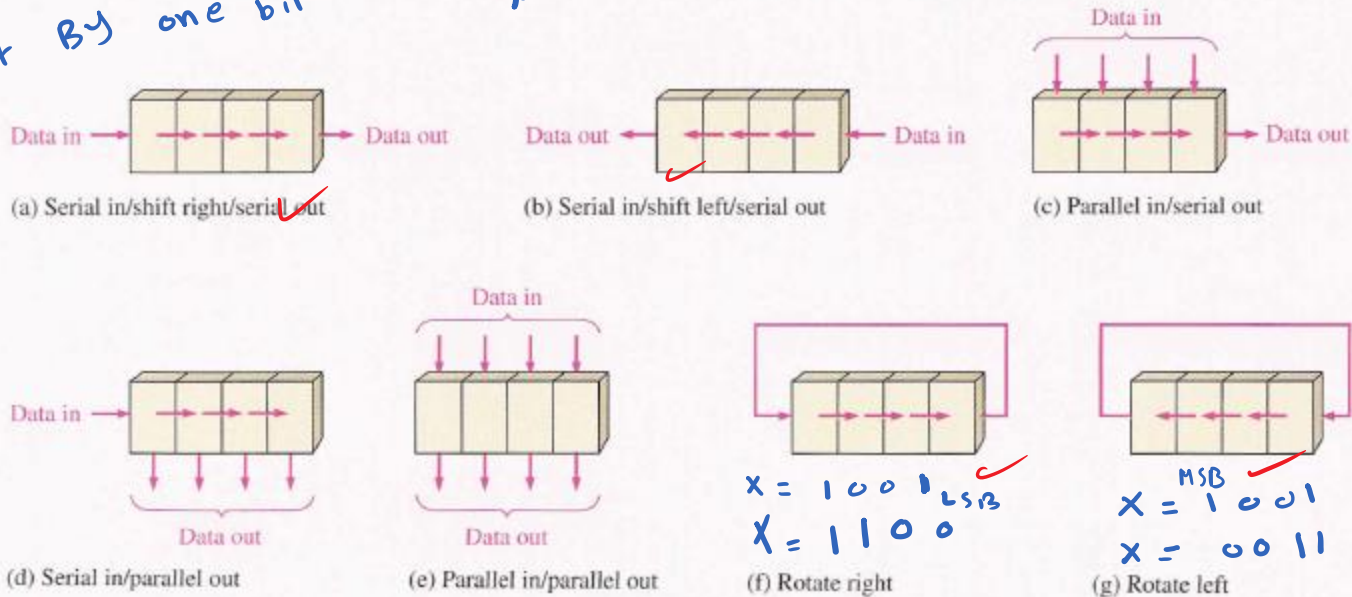
# Shift Register

shift left by one bit  
 $X_B = 10001$   
 $X_N = 00010$

ex shift right by one bit  
 $X_B = 10001$  (LSB)  
 $X_N = 01000$

- Shift register

Shift left by one bit only. ( $X_N = 2 \times X_B$ )  
 $X = 0100$  (410)  
 $X = 1000 = 8_{10}$

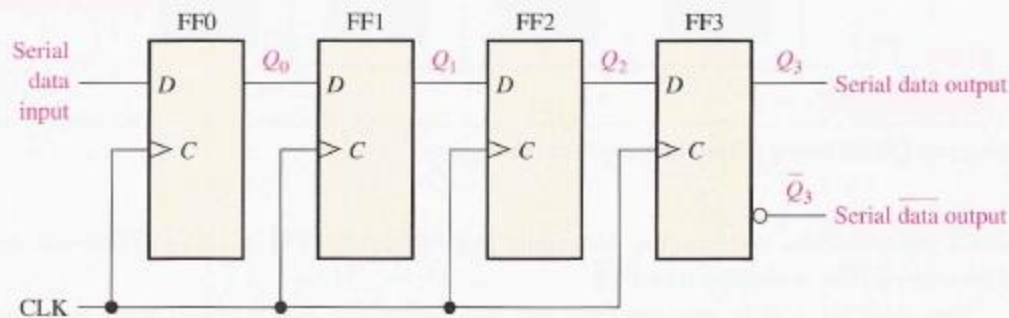


▲ FIGURE 9-2

Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

# Shift Register

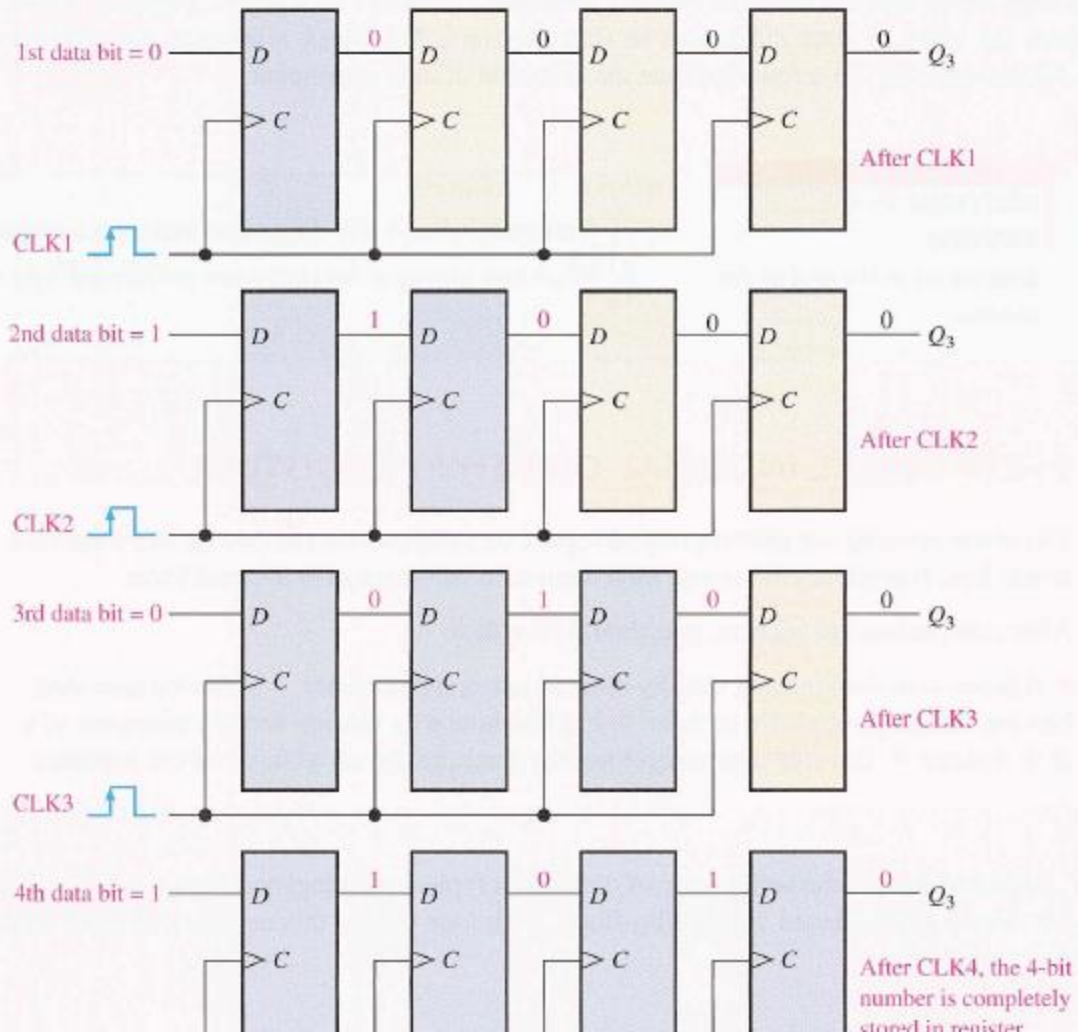
- Serial in/ serial out shift reg.



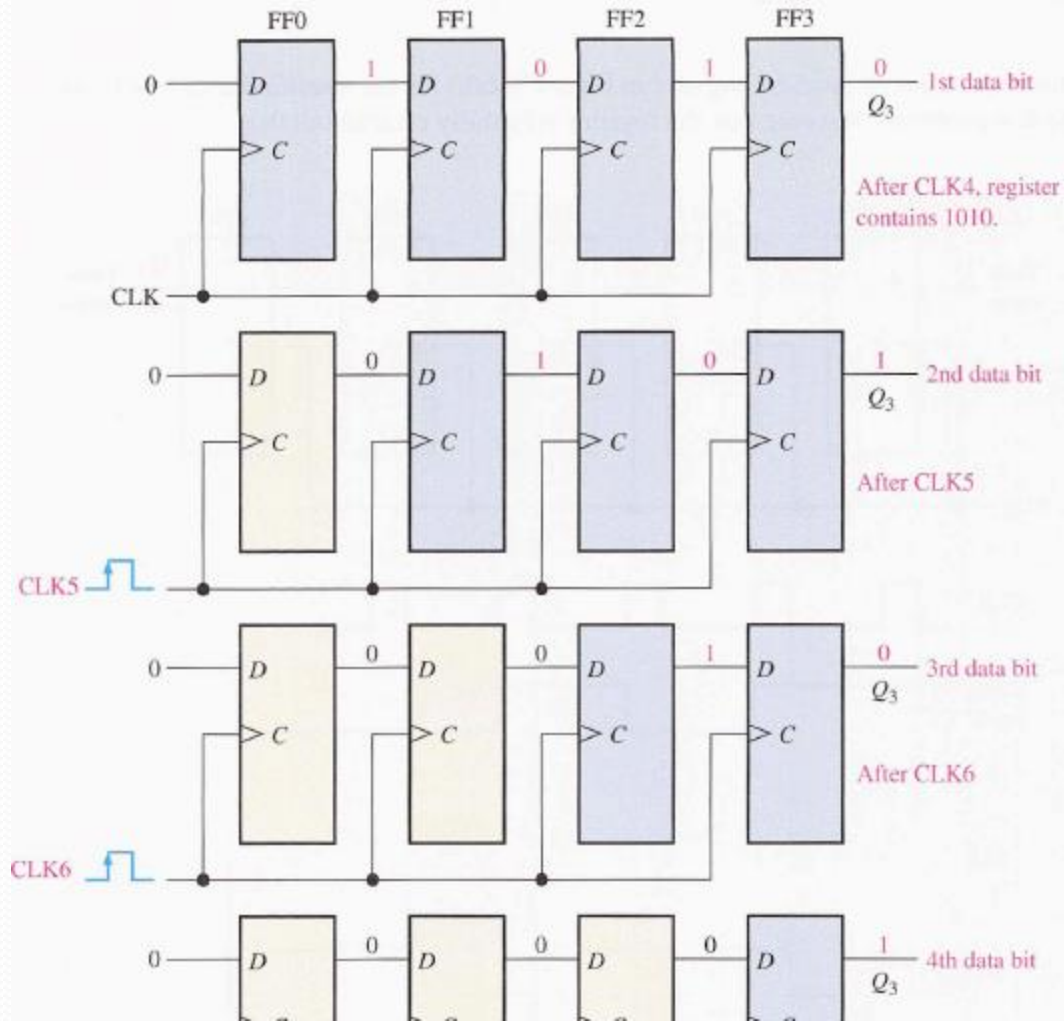
▲ FIGURE 9-3

Serial in/serial out shift register.

# Shift Register



# Shift Register

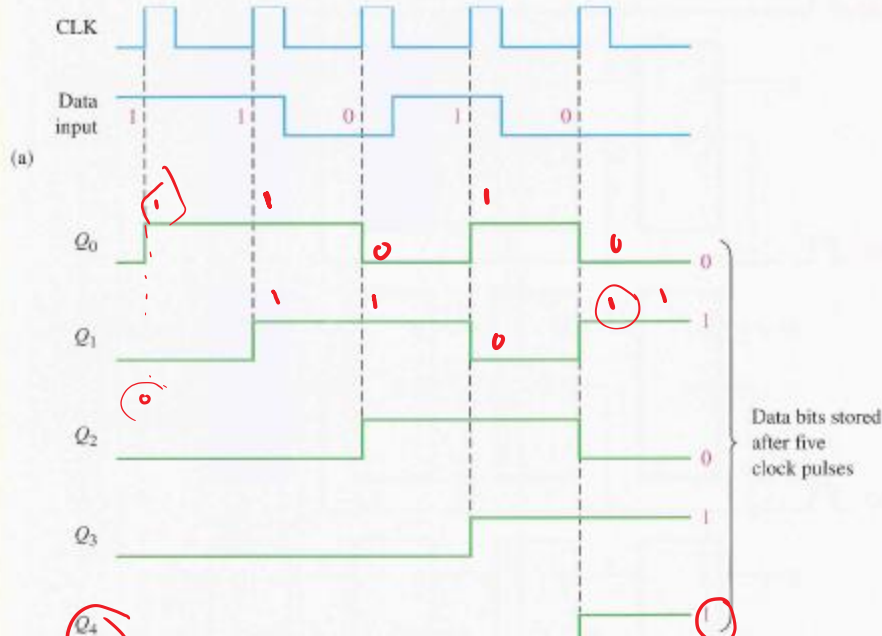
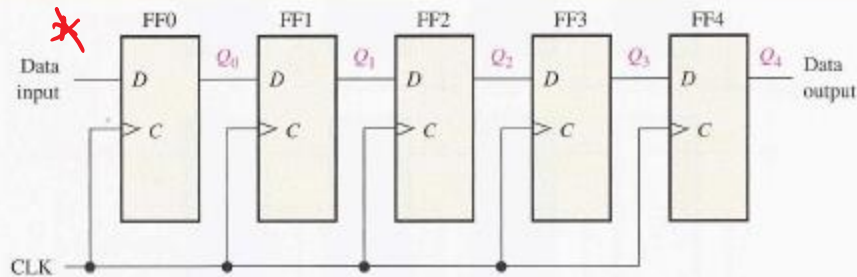


# Shift Register

Show the states of the 5-bit register in Figure 9-6(a) for the specified data input and clock waveforms. Assume that the register is initially cleared (all 0s).

FIGURE 9-6

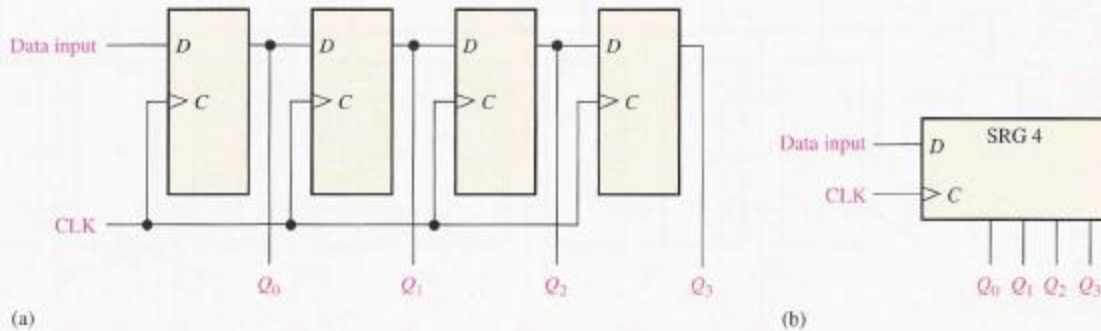
Open file F09-06 to verify operation.



*last data*  
*c/01111*  
*1st data*

# Shift Register

- Serial in /parallel out

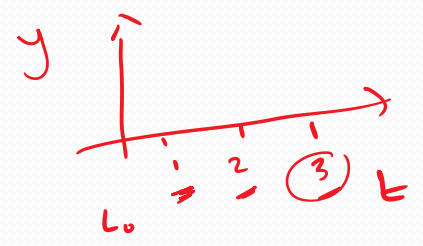


▲ FIGURE 9-8

A serial in/parallel out shift register.

# Shift Register

- Serial in /parallel out



## EXAMPLE 9-2

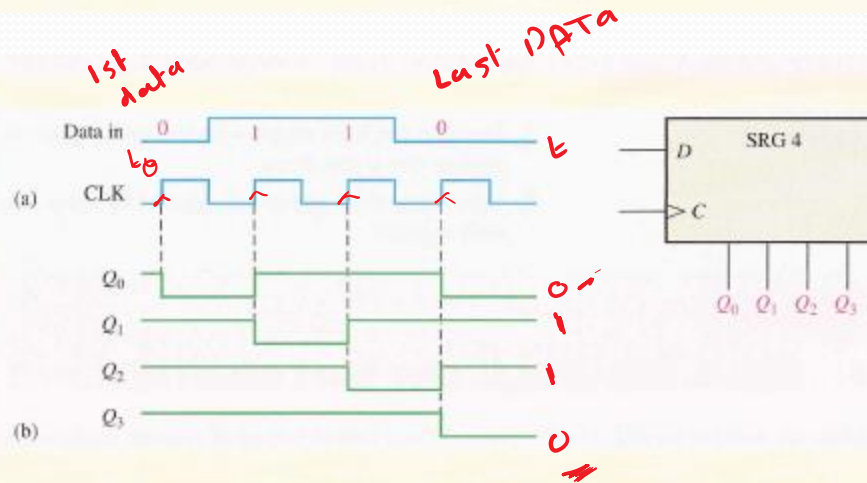
Show the states of the 4-bit register (SRG 4) for the data input and clock waveforms in Figure 9-9(a). The register initially contains all 1s.

**Solution** The register contains 0110 after four clock pulses. See Figure 9-9(b).

**Related Problem** If the data input remains 0 after the fourth clock pulse, what is the state of the register after three additional clock pulses?

0 1 1 0

FIGURE 9-9



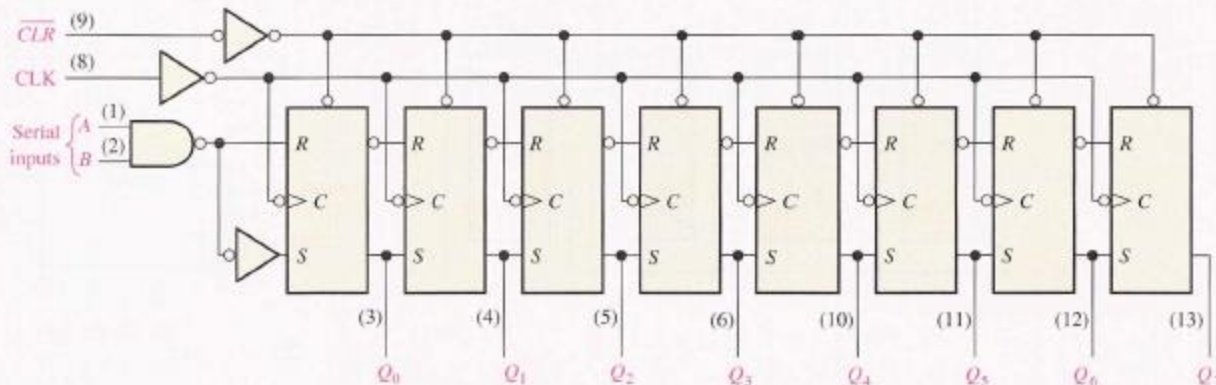
# Shift Register

- Serial in /parallel out

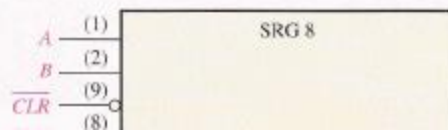
## THE 74HC164 8-BIT SERIAL IN/PARALLEL OUT SHIFT REGISTER



The 74HC164 is an example of an IC shift register having serial in/parallel out operation. The logic diagram is shown in Figure 9-10(a), and a typical logic block symbol is shown in part (b). Notice that this device has two gated serial inputs,  $A$  and  $B$ , and a clear ( $\overline{CLR}$ ) input that is active-LOW. The parallel outputs are  $Q_0$  through  $Q_7$ .



(a) Logic diagram

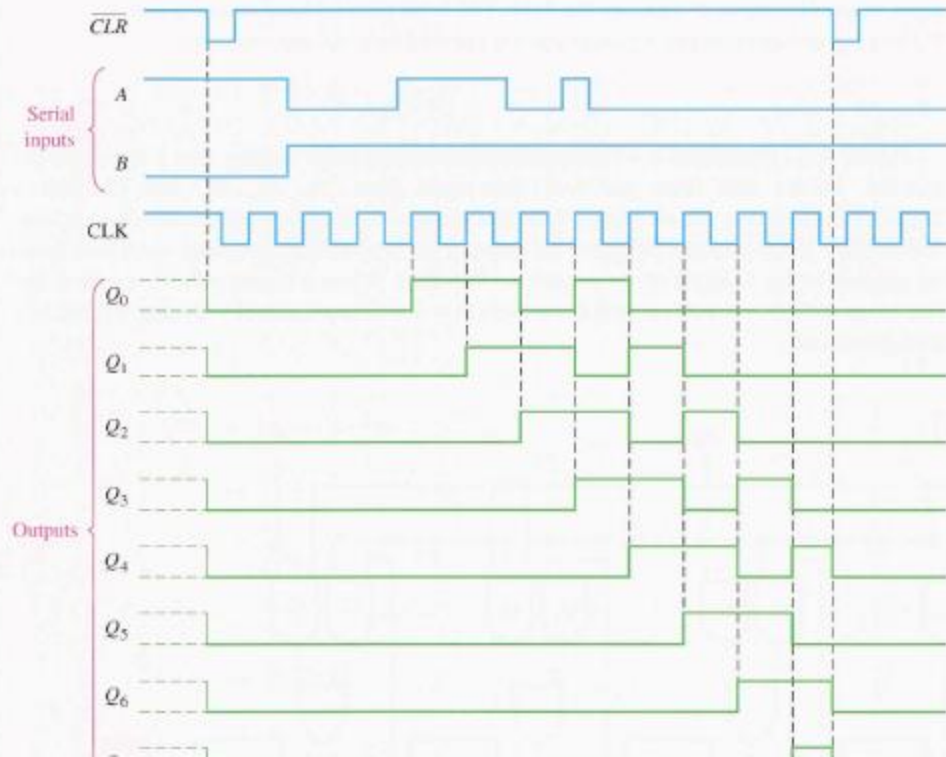




# Shift Register

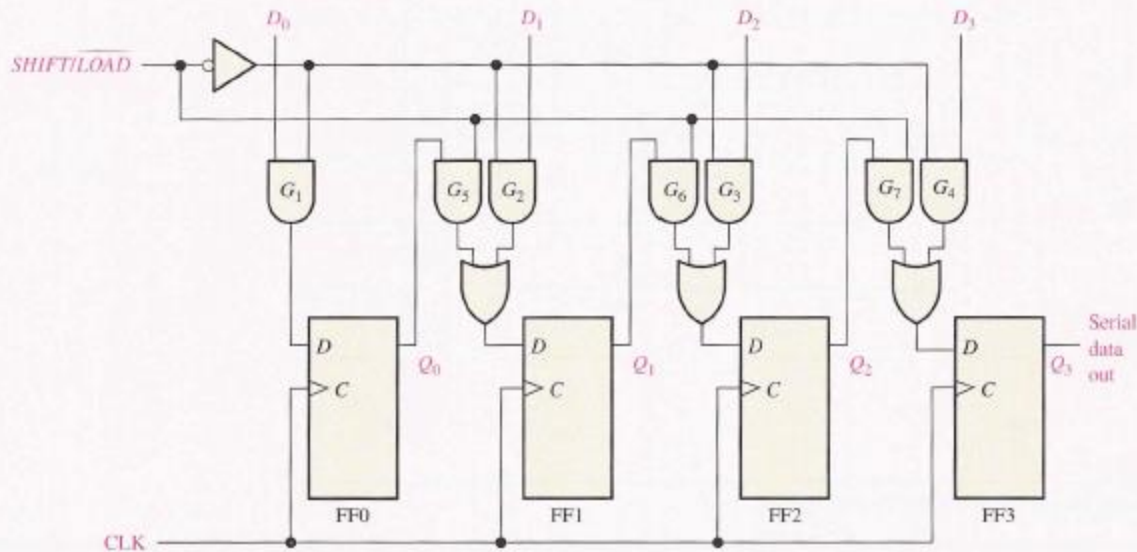
- Serial in /parallel out

A sample timing diagram for the 74HC164 is shown in Figure 9-11. Notice that the serial input data on input *A* are shifted into and through the register after input *B* goes HIGH.

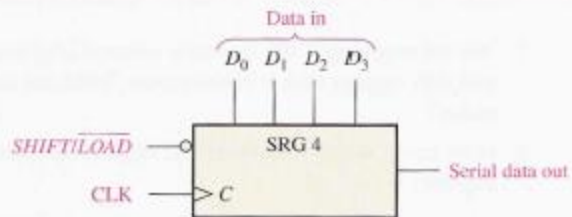


# Shift Register

- Parallel in /serial out



(a) Logic diagram



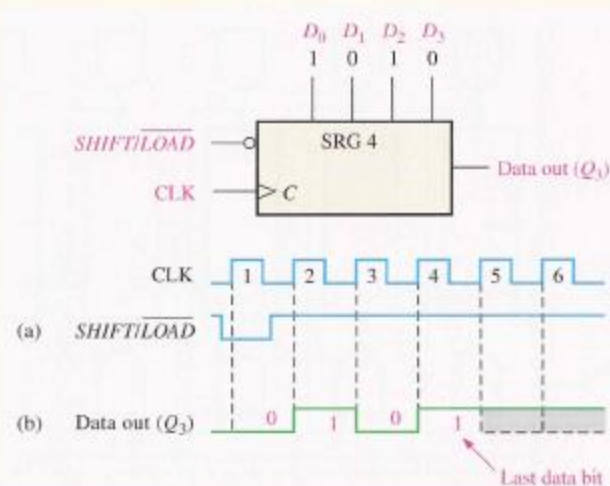
(b) Logic symbol

# Shift Register

- Parallel in /serial out

## EXAMPLE 9-3

Show the data-output waveform for a 4-bit register with the parallel input data and the clock and *SHIFT/LOAD* waveforms given in Figure 9-13(a). Refer to Figure 9-12(a) for the logic diagram.



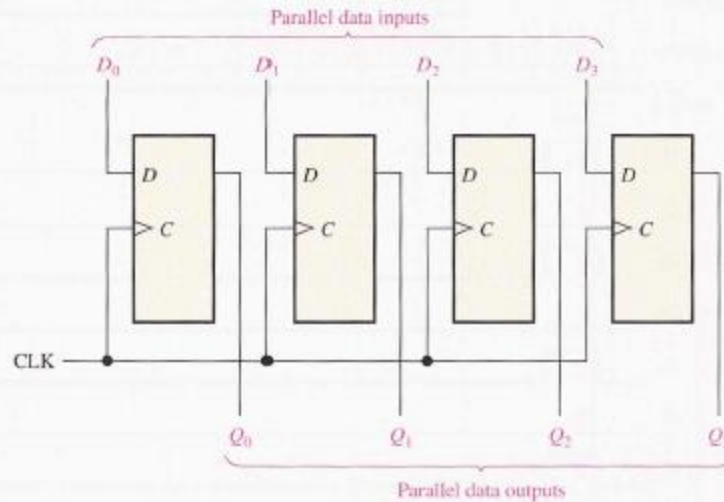
▲ FIGURE 9-13

**Solution** On clock pulse 1, the parallel data ( $D_0D_1D_2D_3 = 1010$ ) are loaded into the register, making  $Q_3$  a 0. On clock pulse 2 the 1 from  $Q_2$  is shifted onto  $Q_3$ ; on clock pulse 3 the 0 is shifted onto  $Q_3$ ; on clock pulse 4 the last data bit (1) is shifted onto  $Q_3$ ; and on clock pulse 5, all data bits have been shifted out, and only 1s remain in the register.

# Shift Register

- Parallel in /parallel out

Figure 9-16 shows a parallel in/parallel out register.



▲ FIGURE 9-16

A parallel in/parallel out register.

# Shift Register

- Parallel in /parallel out

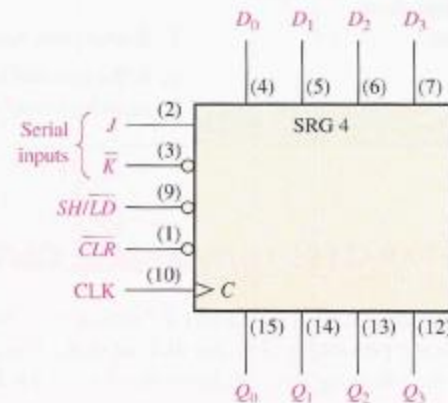
## THE 74HC195 4-BIT PARALLEL-ACCESS SHIFT REGISTER



The 74HC195 can be used for parallel in/parallel out operation. Because it also has a serial input, it can be used for serial in/serial out and serial in/parallel out operations. It can be used for parallel in/serial out operation by using  $Q_3$  as the output. A typical logic block symbol is shown in Figure 9-17.

► FIGURE 9-17

The 74HC195 4-bit parallel access shift register.

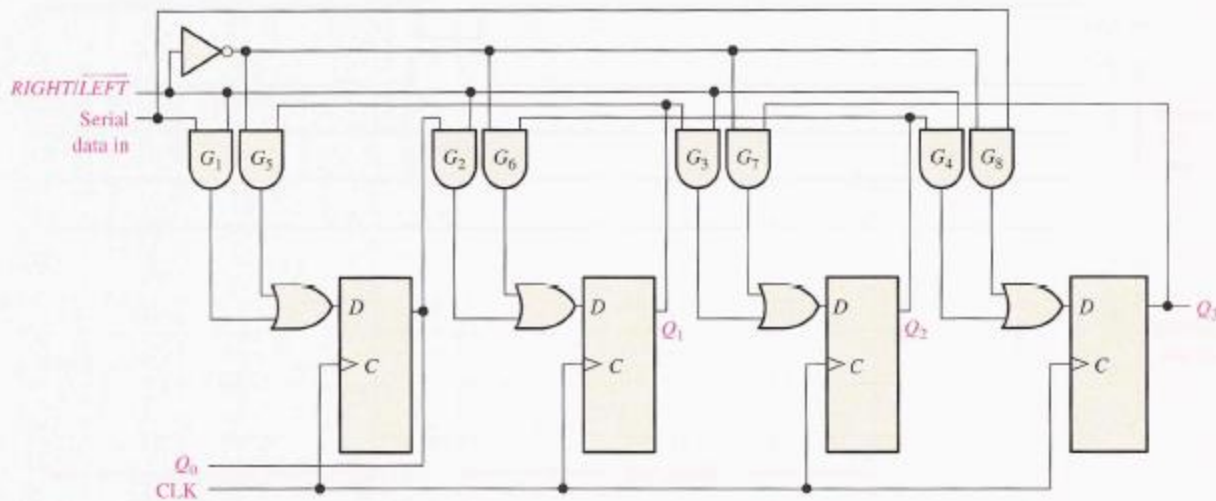


When the  $SHIFT/LOAD$  input ( $SH/\overline{LD}$ ) is LOW, the data on the parallel inputs are entered synchronously on the positive transition of the clock. When  $SH/\overline{LD}$  is HIGH, stored data will shift right ( $Q_0$  to  $Q_3$ ) synchronously with the clock. Inputs  $J$  and  $\overline{K}$  are the serial data inputs to the first stage of the register ( $Q_0$ );  $Q_3$  can be used for serial output data. The active-LOW clear input is asynchronous.

The timing diagram in Figure 9-18 illustrates the operation of this register.

# Shift Register

- Bidirectional shift register



▲ FIGURE 9-19

Four-bit bidirectional shift register. Open file F09-19 to verify the operation.

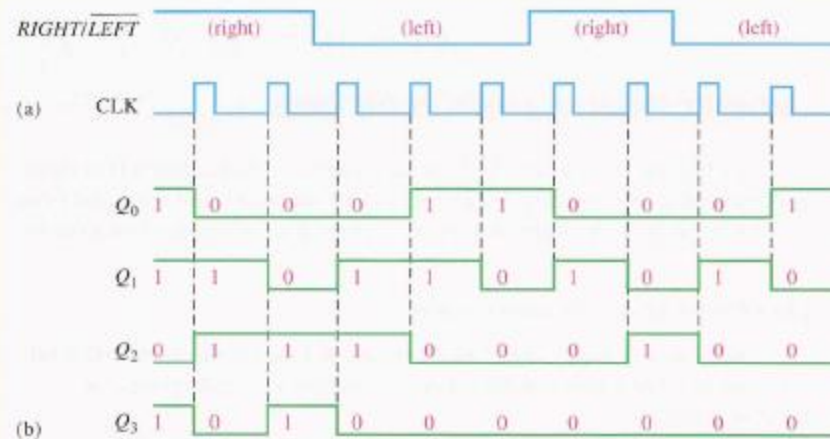
# Shift Register

- Bidirectional shift register

## EXAMPLE 9-4

Determine the state of the shift register of Figure 9-19 after each clock pulse for the given  $RIGHT/\overline{LEFT}$  control input waveform in Figure 9-20(a). Assume that  $Q_0 = 1$ ,  $Q_1 = 1$ ,  $Q_2 = 0$ , and  $Q_3 = 1$  and that the serial data-input line is LOW.

► FIGURE 9-20



**Solution** See Figure 9-20(b).

**Related Problem** Invert the  $RIGHT/\overline{LEFT}$  waveform, and determine the state of the shift register in Figure 9-19 after each clock pulse.

# Shift Register

- Bidirectional shift register

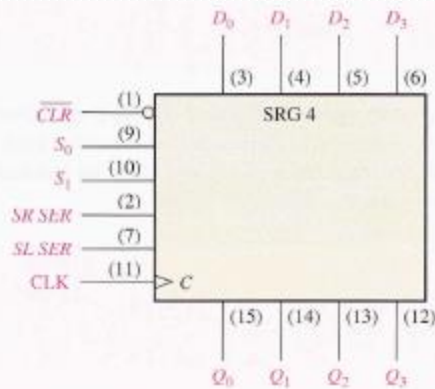
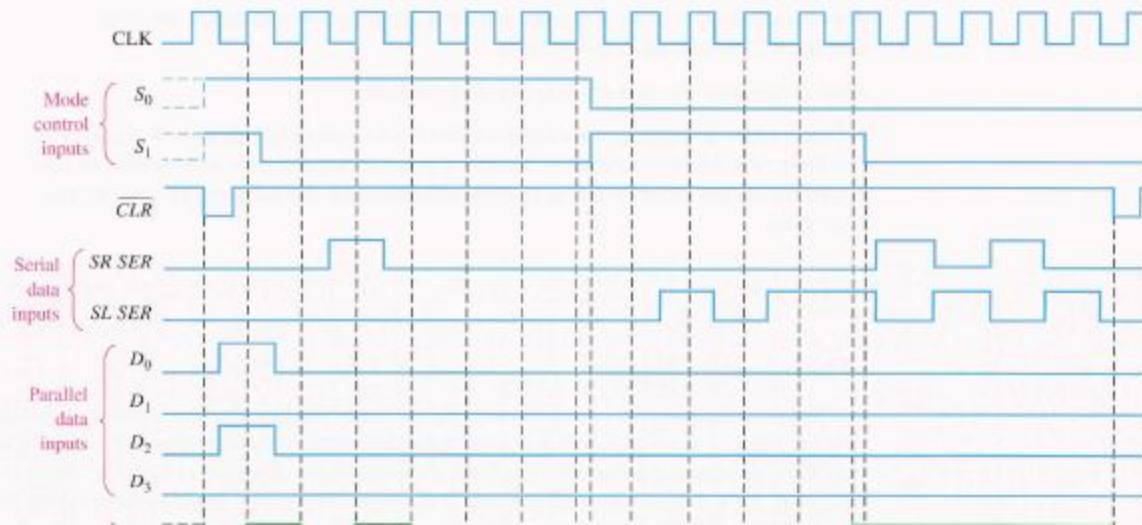


FIGURE 9-21

The 74HC194 4-bit bidirectional universal shift register.





# Shift Register

- shift register as counter ( Self study )