ECI 101 Basic Electronics Course

Module 4 - Digital Electronics

Text Book: T L Floyd and R P Jain, "Digital Fundamentals" Pearson

Department of Electronics Engineering

- Digital v/s Analog – what is the difference?
- Can you name some digital equipments?
- Why are they digital, as a matter of fact !!?

A system using analog & digital circuits

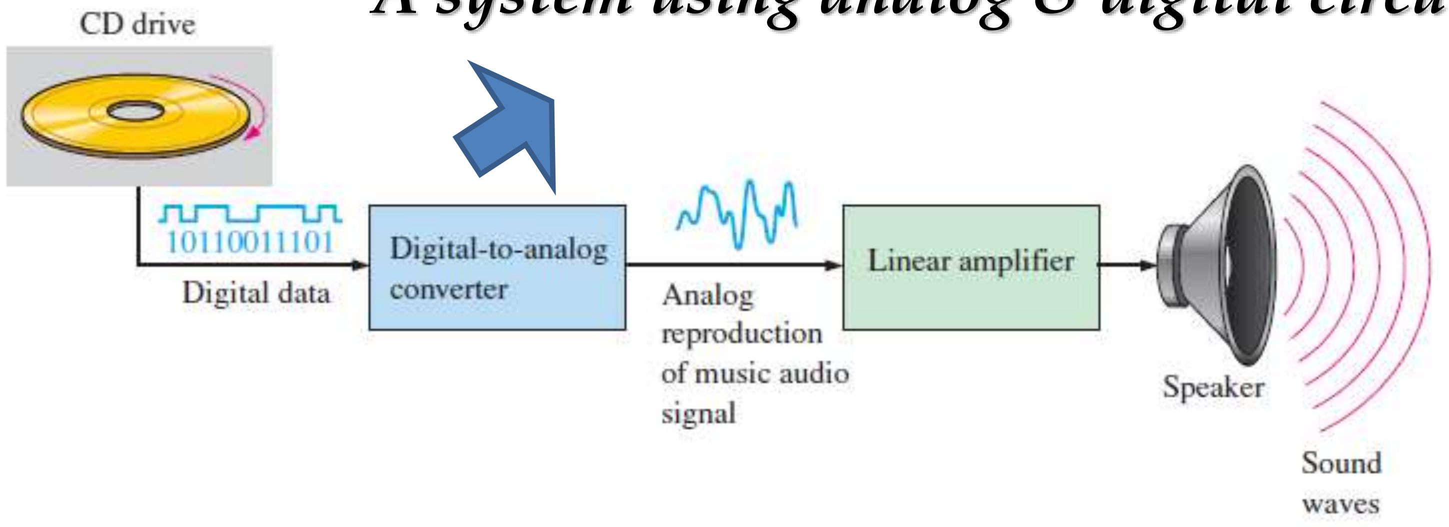

CD drive
10110011101
Digital data
Digital-to-analog converter
Analog reproduction of music audio signal
Linear amplifier
Speaker
Sound waves

FIGURE 1–4 Basic block diagram of a CD player. Only one channel is shown.

- Digital electronics involves circuits and systems in which there are only two possible states. These states are represented by two different voltage levels: A HIGH and a LOW
- If you look at a CD/DVD what are these levels??
- In digital systems such as computers, combinations of the two states, called codes, are used to represent numbers, symbols, alphabetic characters, and other types of information
- The two-state number system is called **binary**, and its two digits are 0 and 1
- A binary digit is called a **bit**

HIGH = 1 and LOW = 0 *Positive logic*

Negative logic ? **HIGH = 0 and LOW = 1**

- Groups of bits (combinations of 1s and 0s), called codes, are used to represent numbers, letters, symbols, instructions, and anything else required in a given application.
- The voltages used to represent a 1 and a 0 are called logic levels.
- Values depend on digital circuit technology – CMOS/ TTL

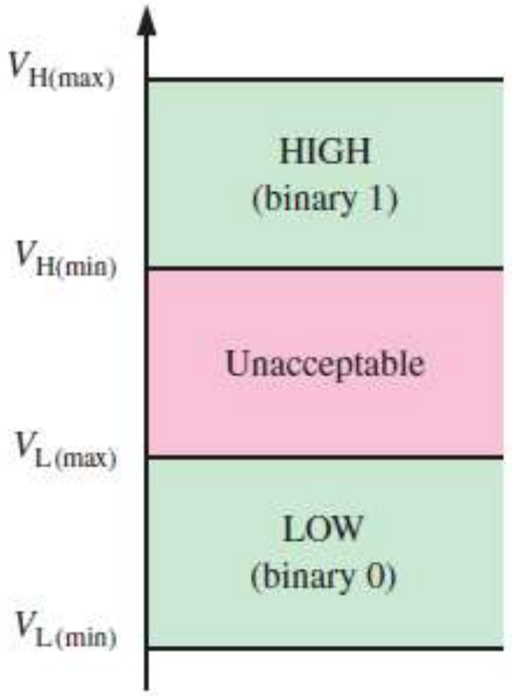
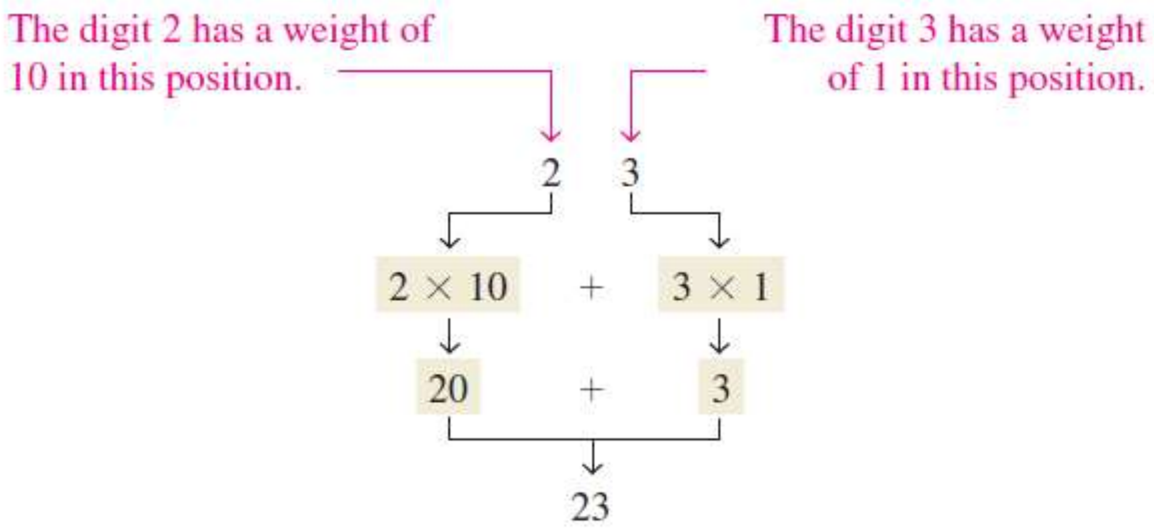


FIGURE 1–6 Logic level ranges of voltage for a digital circuit.

Number Systems & Basics

- The number system that we use in day-to-day life is called **Decimal Number System**.
- Ten digits, 0 through 9
- The ten symbols (digits) do not limit you to expressing only ten different quantities because you use the various digits in appropriate positions within a number to indicate the magnitude of the quantity


The digit 2 has a weight of 10 in this position.
The digit 3 has a weight of 1 in this position.
2 3
2 × 10 + 3 × 1
20 + 3
23

- What is **weight** in the number system??

$$10^2\ 10^1\ 10^0\underset{\substack{\uparrow\\ \text{Decimal point}}}{.}10^{-1}\ 10^{-2}\ 10^{-3}\ \ldots$$

$$\begin{aligned}
568.23 &= (5\times 10^2) + (6\times 10^1) + (8\times 10^0) + (2\times 10^{-1}) + (3\times 10^{-2})\\
&= (5\times 100) + (6\times 10) + (8\times 1) + (2\times 0.1) + (3\times 0.01)\\
&= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03}
\end{aligned}$$

- Can you explain using the number 123.456?

✓ **The decimal number system has a base of 10.**

✓ **The value of a digit is determined by its position in the number**

- **Binary Number System** – less complicated- only 2 digits!!
- Let us count in binary,

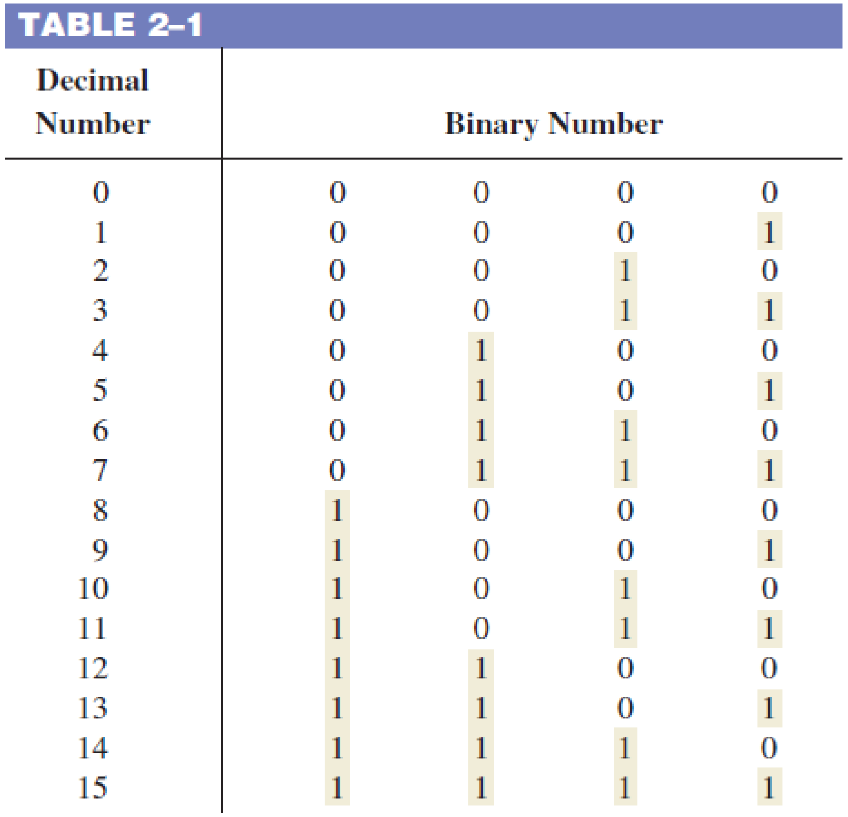


TABLE 2–1

Decimal Number	Binary Number			
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

- with n bits you can count up to a number equal to $2^n - 1$.

✓ **The binary number system has a base of 2.**
✓ **The value of a bit is determined by its position in the number**

- Binary **weights**

$$2^{n-1} \ldots 2^3\,2^2\,2^1\,2^0 \underset{\uparrow \text{ Binary point}}{.} 2^{-1}\,2^{-2} \ldots 2^{-n}$$

TABLE 2–2

Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

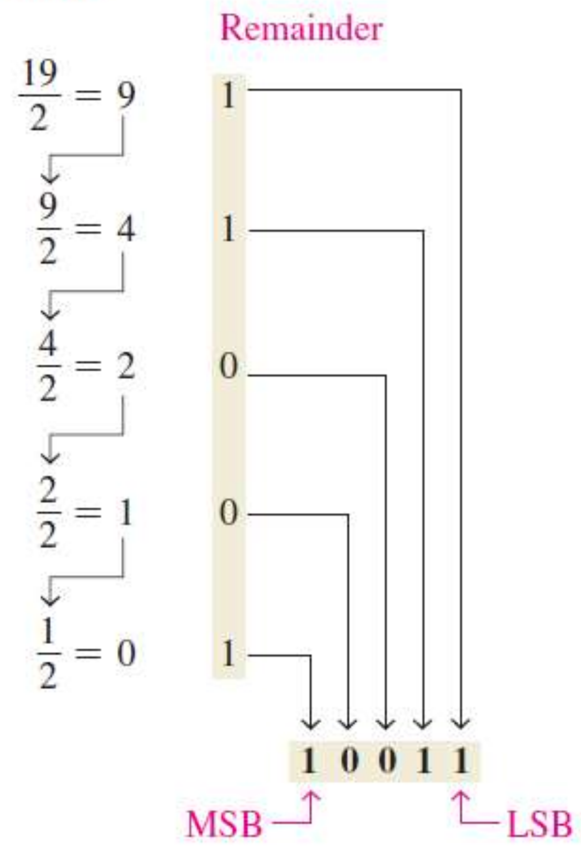
- Decimal numbers to binary

Convert the following decimal numbers to binary:

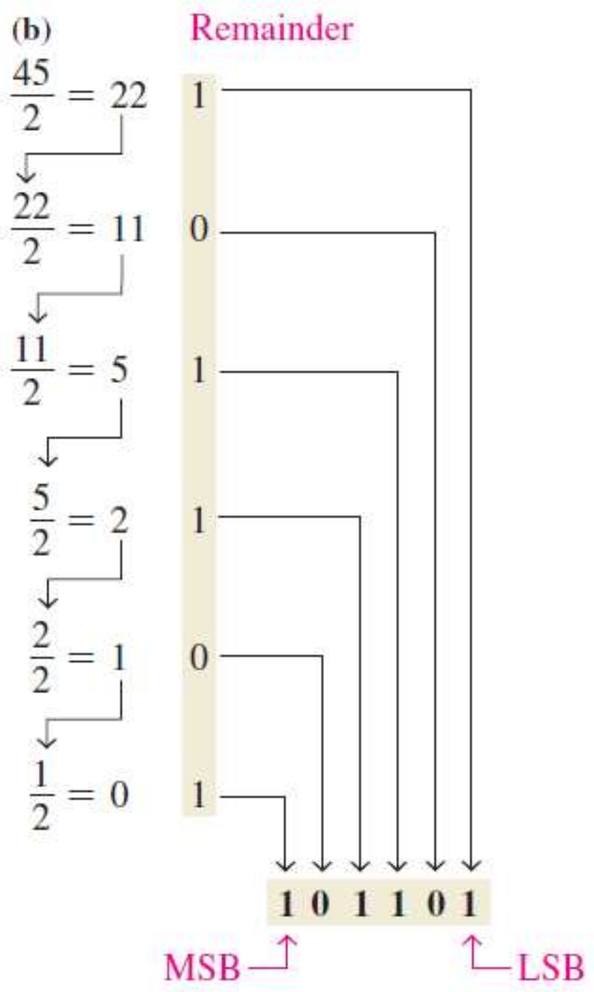
(a) 19 **(b)** 45

Solution

(a)



(b)

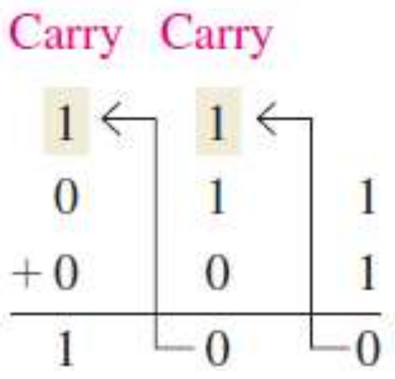


• *Binary Addition*

$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1

Basic Rules

addition of 11 + 1:



Carry bits

$1 + 0 + 0 = 01$	Sum of 1 with a carry of 0
$1 + 1 + 0 = 10$	Sum of 0 with a carry of 1
$1 + 0 + 1 = 10$	Sum of 0 with a carry of 1
$1 + 1 + 1 = 11$	Sum of 1 with a carry of 1

✓ **In binary 1+1 = 10, not 2**

- ***Binary Subtraction***

$0 - 0 = 0$	
$1 - 1 = 0$	
$1 - 0 = 1$	
$10 - 1 = 1$	$0 - 1$ with a borrow of 1

Basic Rules

Subtract 011 from 101.

Solution

$$\begin{array}{r} 101 \\ -011 \\ \hline \mathbf{010} \end{array} \qquad \begin{array}{r} 5 \\ -3 \\ \hline 2 \end{array}$$

✓ **In binary 10 - 1 = 1, not 9.**

- ***Binary Multiplication***

$$\begin{array}{l} 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{array}$$

Basic Rules

Perform the following binary multiplications:

(a) 11×11 **(b)** 101×111

Solution

(a)

$$\begin{array}{rr} & 11 \\ & \times\ 11 \\ \cline{2-2} \text{Partial products}\left\{\begin{array}{r} 11 \\ +11 \end{array}\right. \\ \cline{2-2} & \mathbf{1001} \end{array} \qquad \begin{array}{r} 3 \\ \times\ 3 \\ \hline 9 \end{array}$$

(b)

$$\begin{array}{rr} & 111 \\ & \times\ 101 \\ \cline{2-2} \text{Partial products}\left\{\begin{array}{r} 111 \\ 000\ \, \\ +111\ \ \ \ \end{array}\right. \\ \cline{2-2} & \mathbf{100011} \end{array} \qquad \begin{array}{r} 7 \\ \times\ 5 \\ \hline 35 \end{array}$$

✓ **Binary multiplication of two bits is the same as multiplication of the decimal digits 0 and 1.**

- ***Binary Division***

Perform the following binary divisions:

(a) 110 ÷ 11 (b) 110 ÷ 10

Solution

(a) $$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ \underline{11} \\ 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

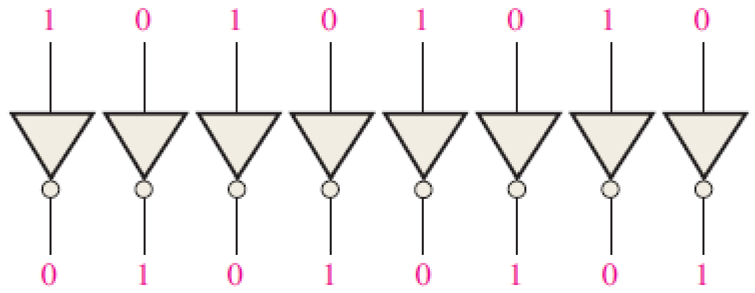
(b) $$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

Complements & Related Arithemtic

1’s complement

- The 1’s complement of a binary number is found by changing all 1s to 0s and all 0s to 1s

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	1’s complement



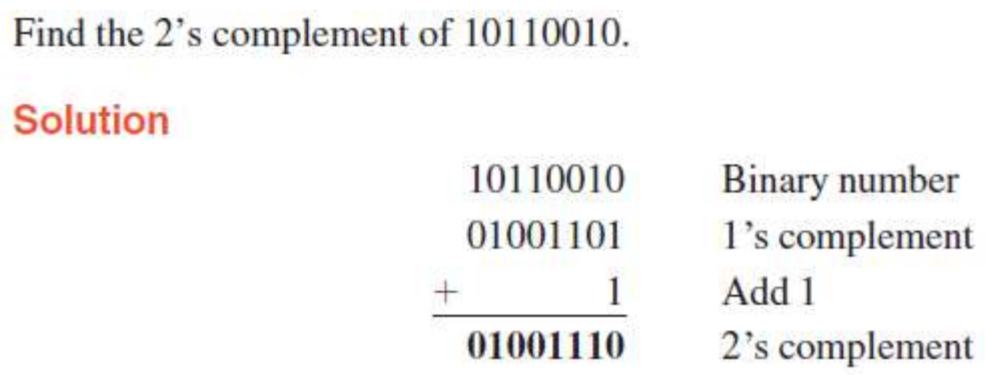
Example of inverters used to obtain the 1’s complement of a binary number.

✓ **Change each bit in a number to get the 1’s complement**

2’s complement

- The 2’s complement of a binary number is found by adding 1 to the LSB of the 1’s complement.

2’s complement = (1’s complement) + 1



Find the 2’s complement of 10110010.

Solution

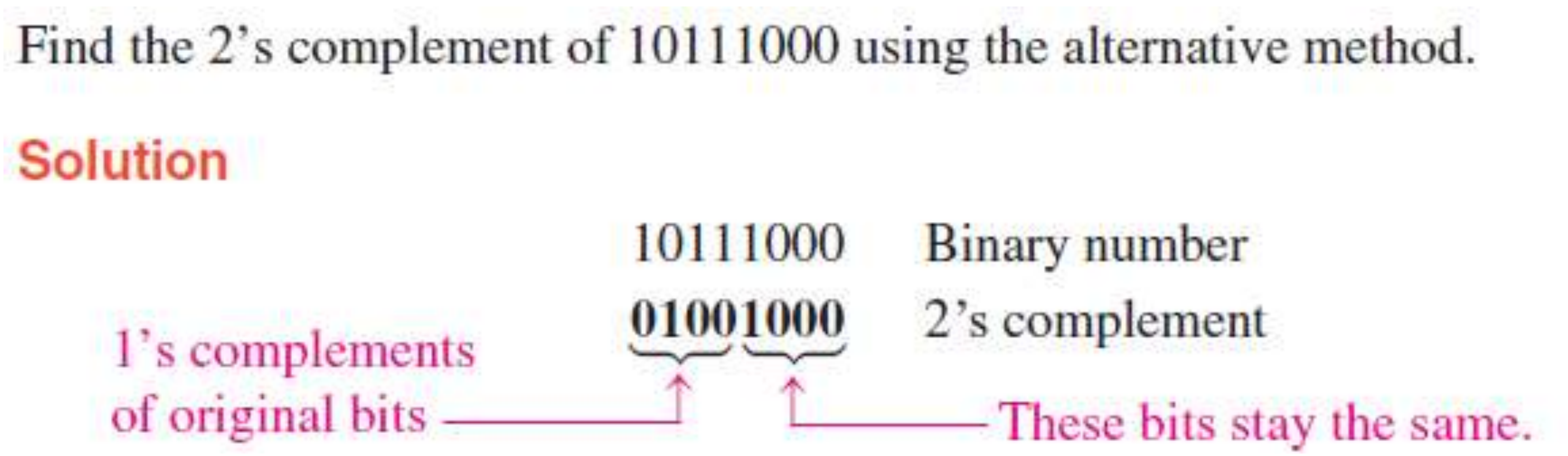
10110010	Binary number
01001101	1’s complement
+ 1	Add 1
01001110	2’s complement

✓ **Change all bits to the left of the least significant 1 to get 2’s complement**

2’s complement

- An alternative method of finding the 2’s complement of a binary number is as follows:

  1. Start at the right with the LSB and write the bits as they are up to and including the first 1.
  2. Take the 1’s complements of the remaining bits.



Find the 2’s complement of 10111000 using the alternative method.

Solution

$$\begin{array}{ll} 10111000 & \text{Binary number} \\ \underbrace{0100}_{\text{1's complements of original bits}}\underbrace{1000}_{\text{These bits stay the same.}} & \text{2's complement} \end{array}$$

2’s complement

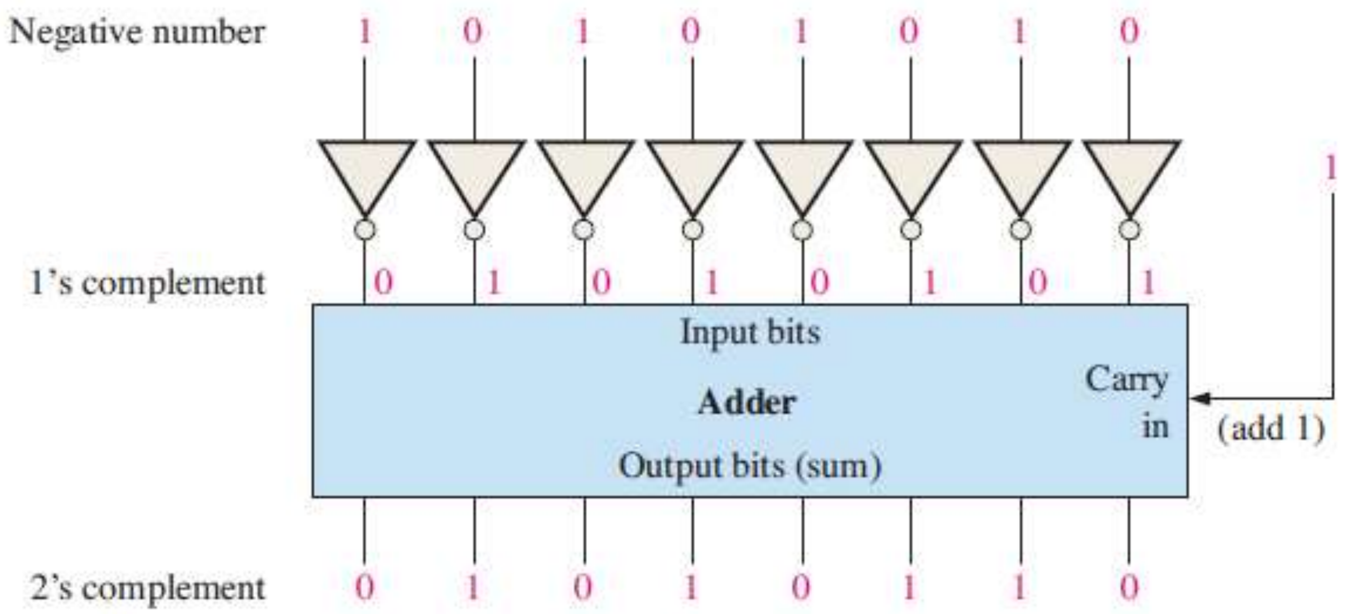
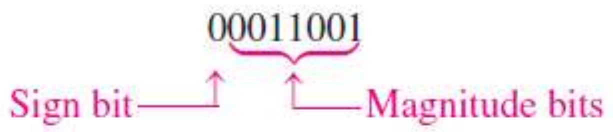


FIGURE 2–3 Example of obtaining the 2’s complement of a negative binary number.

Signed Numbers

The left-most bit in a signed binary number is the **sign bit**, which tells you whether the number is positive or negative.

A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

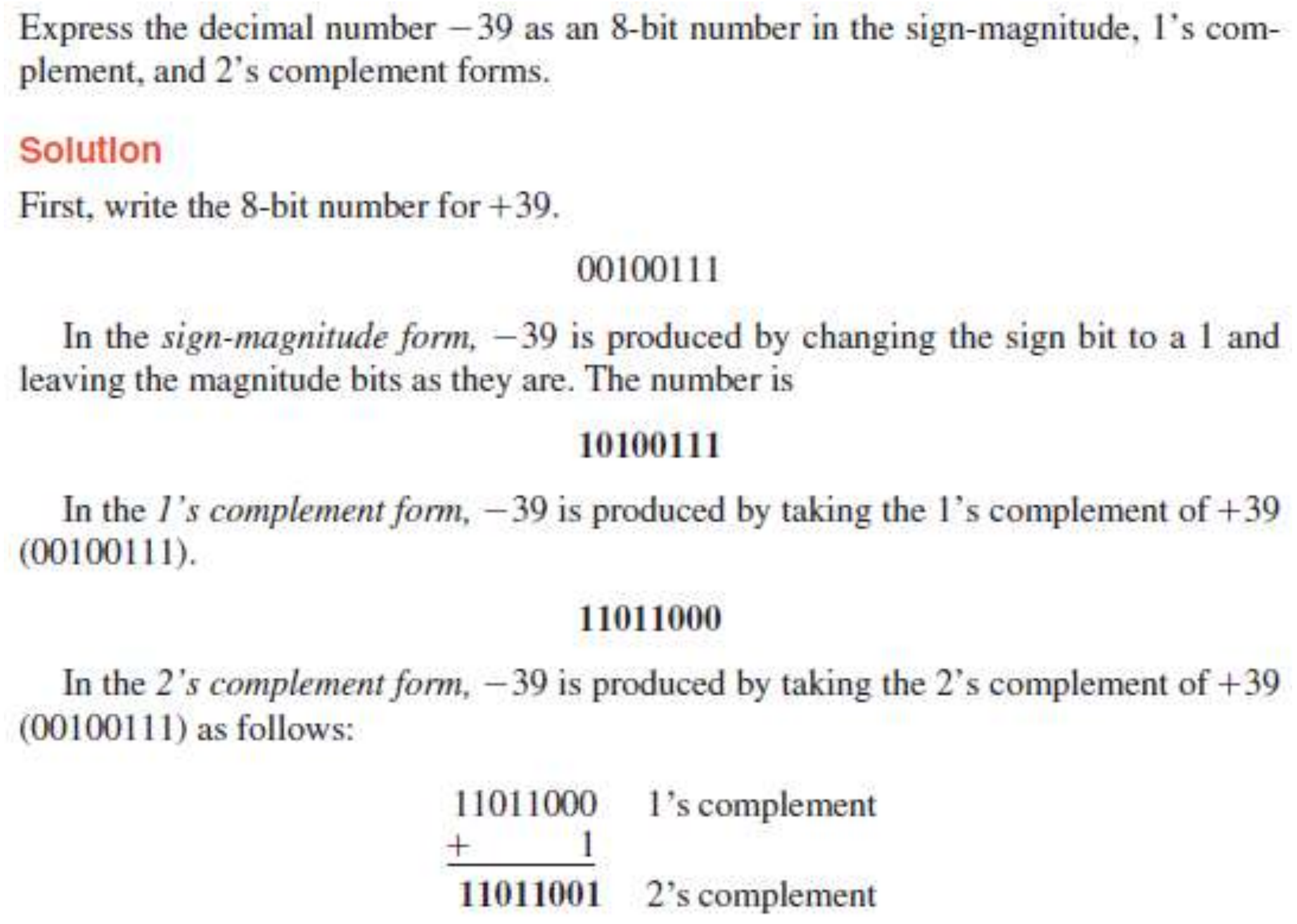


The decimal number −25 is expressed as

10011001

In the sign-magnitude form, a negative number has the same magnitude bits as the corresponding positive number but the sign bit is a 1 rather than a zero.

- **In the 1’s complement form, a negative number is the 1’s complement of the corresponding positive number**
- **In the 2’s complement form, a negative number is the 2’s complement of the corresponding positive number**



Express the decimal number −39 as an 8-bit number in the sign-magnitude, 1’s complement, and 2’s complement forms.

Solution

First, write the 8-bit number for +39.

00100111

In the *sign-magnitude form,* −39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the *1’s complement form,* −39 is produced by taking the 1’s complement of +39 (00100111).

11011000

In the *2’s complement form,* −39 is produced by taking the 2’s complement of +39 (00100111) as follows:

$$\begin{array}{rl} 11011000 & \text{1's complement} \\ +\qquad\quad 1 & \\ \hline \mathbf{11011001} & \text{2's complement} \end{array}$$

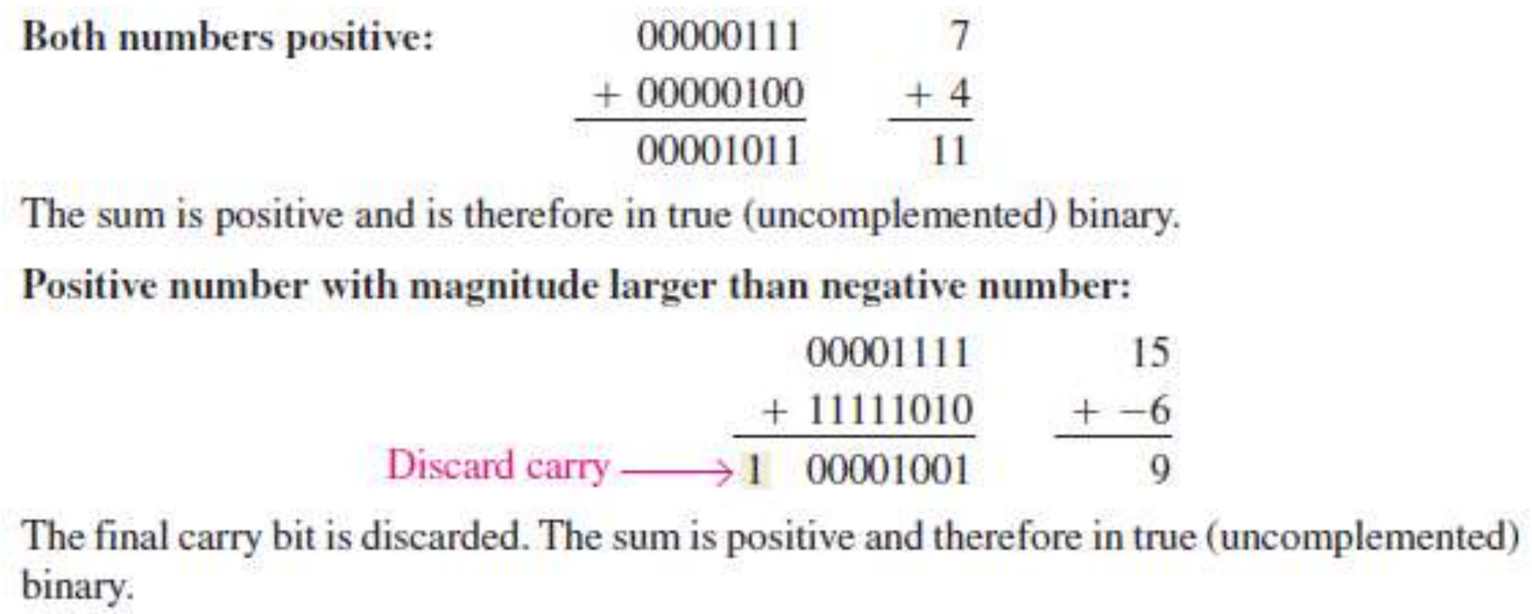
- The formula for finding the number of different combinations of n bits is

  Total combinations = 2n
- For 2's complement signed numbers, the range of values for n-bit numbers is

  Range = $-(2^{n-1})$ to $+(2^{n-1} - 1)$

where in each case there is one sign bit and n - 1 magnitude bits.

Addition



Both numbers positive:

$$\begin{array}{rr} 00000111 & 7 \\ +\ 00000100 & +\ 4 \\ \hline 00001011 & 11 \end{array}$$

The sum is positive and is therefore in true (uncomplemented) binary.

Positive number with magnitude larger than negative number:

$$\begin{array}{rrr} & 00001111 & 15 \\ & +\ 11111010 & +\ -6 \\ \hline \text{Discard carry} \longrightarrow 1 & 00001001 & 9 \end{array}$$

The final carry bit is discarded. The sum is positive and therefore in true (uncomplemented) binary.

Negative number with magnitude larger than positive number:

$$\begin{array}{rr}
00010000 & 16 \\
+\ 11101000 & +\ -24 \\
\hline
11111000 & -8
\end{array}$$

The sum is negative and therefore in 2's complement form.

Both numbers negative:

$$\begin{array}{rrr}
 & 11111011 & -5 \\
 & +\ 11110111 & +\ -9 \\
\hline
\text{Discard carry} \longrightarrow 1 & 11110010 & -14
\end{array}$$

The final carry bit is discarded. The sum is negative and therefore in 2's complement form.

Subtraction

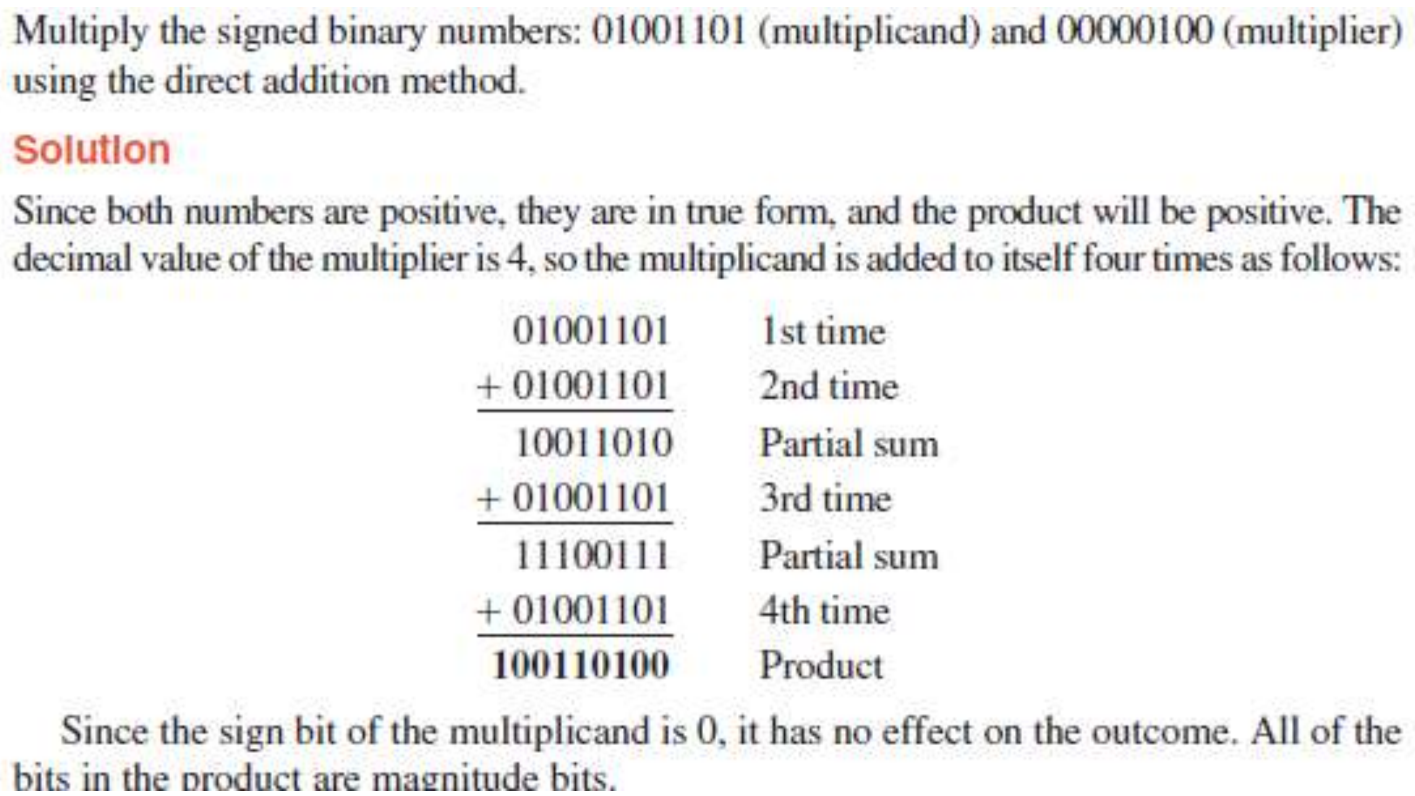
- To subtract two signed numbers, take the 2's complement of the subtrahend and add. Discard any final carry bit.

00001000 – 00000011

In this case, $8 - 3 = 8 + (-3) = 5$.

$$\begin{array}{rrl}
 & 00001000 & \text{Minuend } (+8) \\
 & +\ 11111101 & \text{2's complement of subtrahend } (-3) \\
\hline
\text{Discard carry} \longrightarrow & 1\ 00000101 & \text{Difference } (+5)
\end{array}$$

Multiplication



Multiply the signed binary numbers: 01001101 (multiplicand) and 00000100 (multiplier) using the direct addition method.

Solution

Since both numbers are positive, they are in true form, and the product will be positive. The decimal value of the multiplier is 4, so the multiplicand is added to itself four times as follows:

01001101	1st time
+ 01001101	2nd time
10011010	Partial sum
+ 01001101	3rd time
11100111	Partial sum
+ 01001101	4th time
100110100	Product

Since the sign bit of the multiplicand is 0, it has no effect on the outcome. All of the bits in the product are magnitude bits.

- **If the signs are the same, the product is positive.**
- **If the signs are different, the product is negative.**

Other number systems/codes

Hexadecimel

- ✓ The hexadecimal number system consists of digits 0–9 and letters A–F

TABLE 2–3

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Conversion

Convert the following binary numbers to hexadecimal:

(a) 1100101001010111 **(b)** 111111000101101001

Solution

(a) $$\underbrace{1100}_{\text{C}}\underbrace{1010}_{\text{A}}\underbrace{0101}_{5}\underbrace{0111}_{7} = \mathbf{CA57}_{16}$$

(b) $$\underbrace{0011}_{3}\underbrace{1111}_{\text{F}}\underbrace{0001}_{1}\underbrace{0110}_{6}\underbrace{1001}_{9} = \mathbf{3F169}_{16}$$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solution

(a) $$\overset{1}{1}\underbrace{0000}_{0}\underbrace{1010}_{\text{A}}\underbrace{0100}_{4} = \mathbf{1000010100100}$$

(b) $$\underbrace{1100}_{\text{C}}\underbrace{1111}_{\text{F}}\underbrace{1000}_{8}\underbrace{1110}_{\text{E}} = \mathbf{1100111110001110}$$

(c) $$\underbrace{1001}_{9}\underbrace{0111}_{7}\underbrace{0100}_{4}\underbrace{0010}_{2} = \mathbf{1001011101000010}$$

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

Octal

- The octal number system is composed of eight digits, which are 0, 1, 2, 3, 4, 5, 6, 7
- To count above 7, begin another column and start over:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21, etc

Octal/binary conversion.

Octal Digit	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

BCD – Binary Coded Decimal

8421 BCD code

- The 8421 code is a type of BCD (binary coded decimal) code.
- Binary coded decimal means that each decimal digit, 0 through 9, is represented by a binary code of four bits.
- The designation 8421 indicates the binary weights of the four bits (2^3, 2^2, 2^1, 2^0).
- The six code combinations that are not used—1010, 1011, 1100, 1101, 1110, and 1111—are invalid in the 8421 BCD code

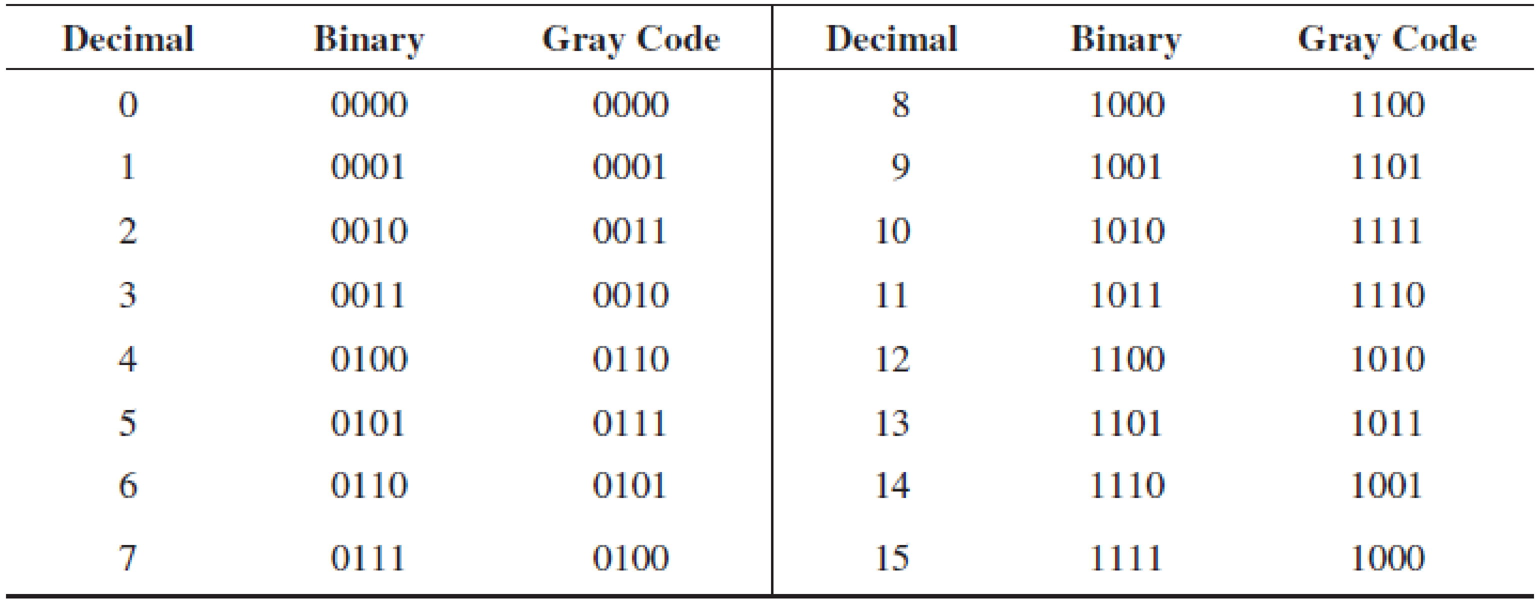
Decimal/BCD conversion.

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

Gray Code

- The Gray code is unweighted and is not an arithmetic code
- It exhibits only a single bit change from one code word to the next in sequence
  – Why is that so important??

Four-bit Gray code.



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

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

$$\begin{tikzcd}
1 \arrow[r, dashed, "+"] \arrow[d] & 0 \arrow[r, dashed, "+"] \arrow[d] & 1 \arrow[r, dashed, "+"] \arrow[d] & 1 \arrow[r, dashed, "+"] \arrow[d] & 0 \arrow[d] & \text{Binary} \\
1 & 1 & 1 & 0 & 1 & \text{Gray}
\end{tikzcd}$$

The Gray code is 11101.

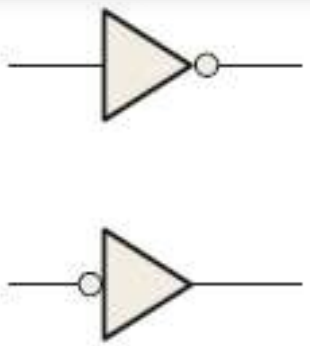
For example, the conversion of the Gray code word 11011 to binary is as follows:

$$\begin{tikzcd}
1 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & \text{Gray} \\
1 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 0 \arrow[ur, "+"] & 1 \arrow[ur, "+"] & 0 & \text{Binary}
\end{tikzcd}$$

The binary number is 10010.

Basic Logic Gates

Inverter

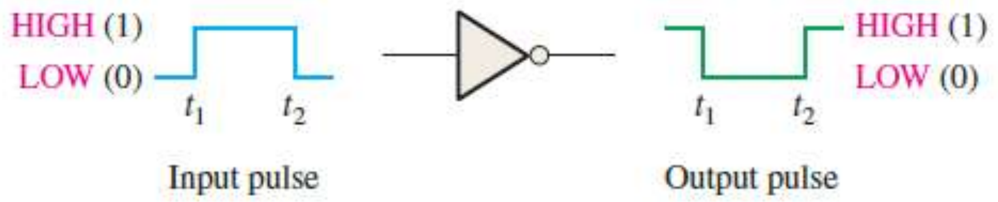


(a) Distinctive shape symbols with negation indicators

Inverter truth table.

Input	Output
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW, thereby producing an inverted output pulse.



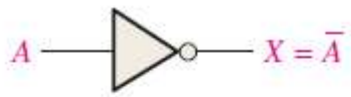
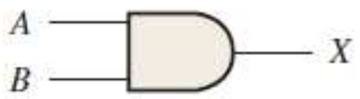


FIGURE 3–6 The inverter complements an input variable.

In Boolean algebra, $X = \overline{A}$

AND gate

- For a 2-input AND gate, output *X* is HIGH only when inputs *A* and *B* are HIGH
- *X* is LOW when either *A* or *B* is LOW, or when both *A* and *B* are LOW.

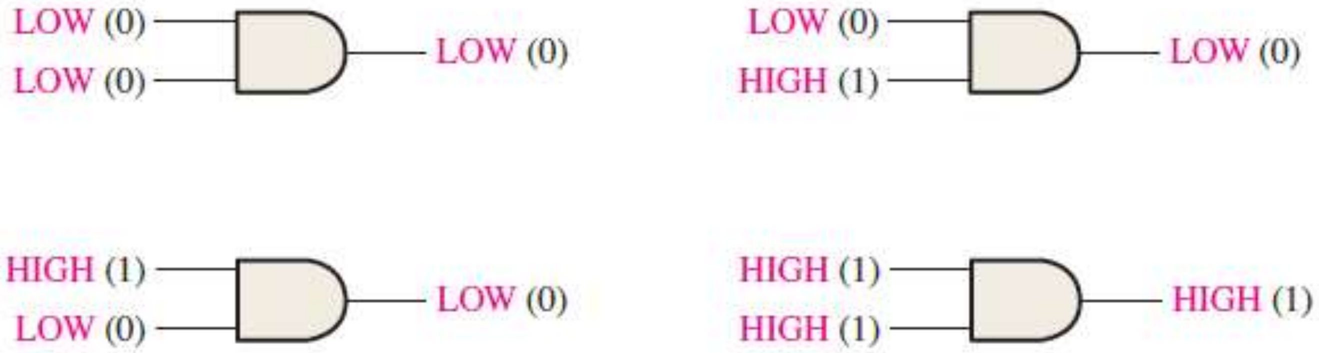


(a) Distinctive shape

Truth table for a 2-input AND gate.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

1 = HIGH, 0 = LOW



✓ **For an AND gate, all HIGH inputs produce a HIGH output**

If two waveforms, *A* and *B*, are applied to the AND gate inputs as in Figure 3–11, what is the resulting output waveform?

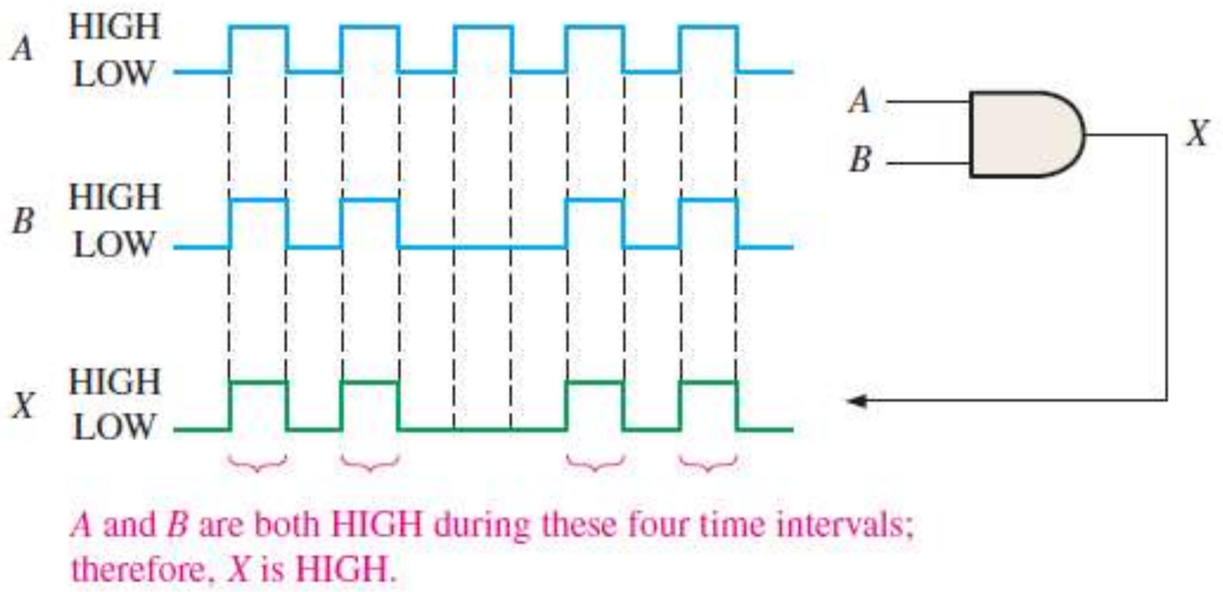

A
HIGH
LOW
B
HIGH
LOW
X
HIGH
LOW
A
B
X
A and B are both HIGH during these four time intervals; therefore, X is HIGH.

FIGURE 3–11

Solution

The output waveform *X* is HIGH only when both *A* and *B* waveforms are HIGH as shown in the timing diagram in Figure 3–11.

The total number of possible combinations of binary inputs to a gate is

$$N = 2^n$$

where *N* is the number of possible input combinations and *n* is the number of input variables

- Boolean multiplication follows the same basic rules governing binary multiplication
- **Boolean multiplication is the same as the AND function.**

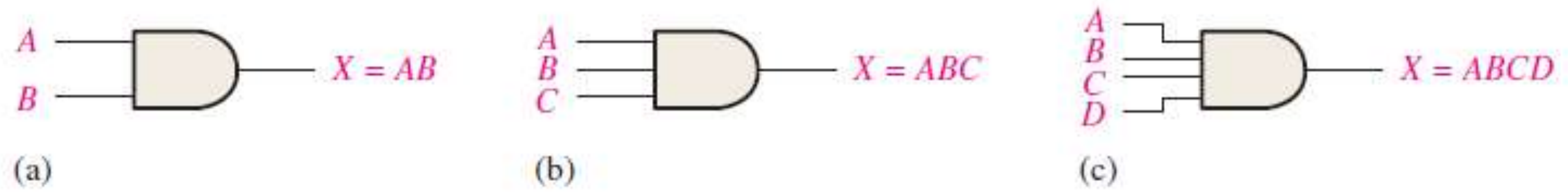
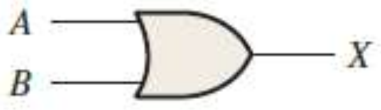


FIGURE 3–15 Boolean expressions for AND gates with two, three, and four inputs.

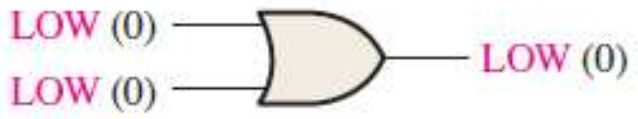
A	B	$AB = X$
0	0	$0 \cdot 0 = 0$
0	1	$0 \cdot 1 = 0$
1	0	$1 \cdot 0 = 0$
1	1	$1 \cdot 1 = 1$

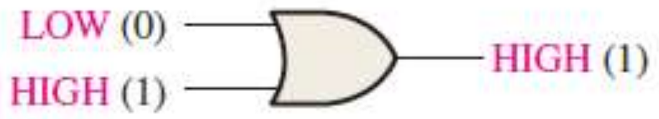
OR gate

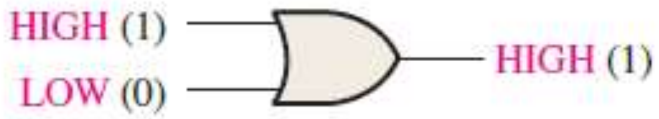


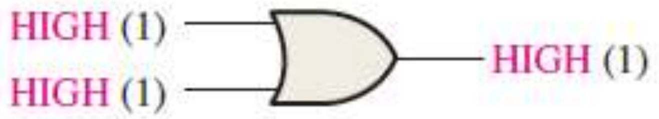
(a) Distinctive shape

- For a 2-input OR gate, output X is HIGH when either input A or input B is HIGH, or when both A and B are HIGH;
- X is LOW only when both A and B are LOW.





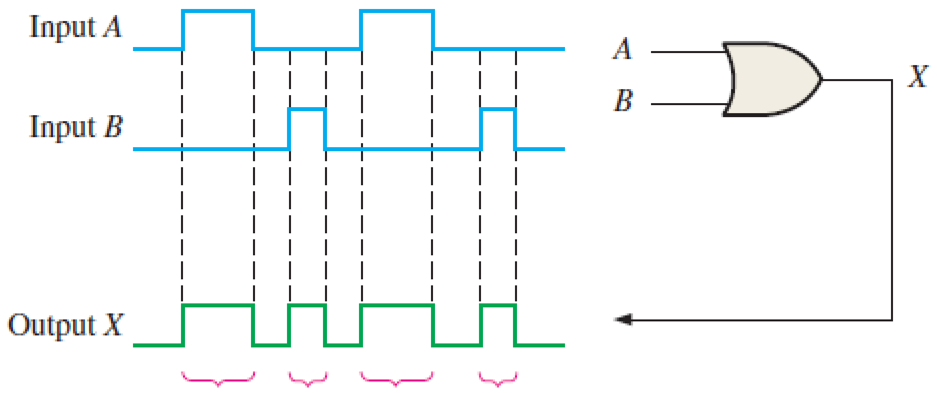




Truth table for a 2-input OR gate.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

1 = HIGH, 0 = LOW



When either input or both inputs are HIGH, the output is HIGH.

✓ **For an OR gate, at least one HIGH input produces a HIGH output**

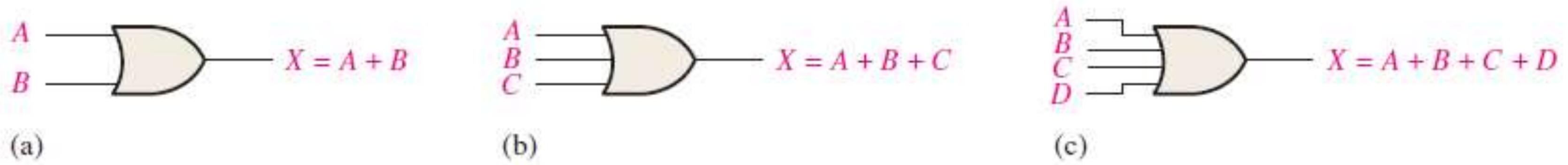


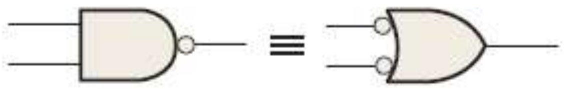
FIGURE 3–24 Boolean expressions for OR gates with two, three, and four inputs.

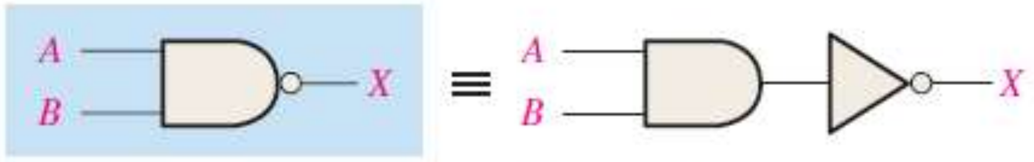
A	B	$A + B = X$
0	0	$0 + 0 = 0$
0	1	$0 + 1 = 1$
1	0	$1 + 0 = 1$
1	1	$1 + 1 = 1$

Boolean addition is the same as the OR function.

- ✓ **When variables are separated by +, they are Ored**
- ✓ **When variables are shown together like ABC, they are ANDed.**

NAND gate





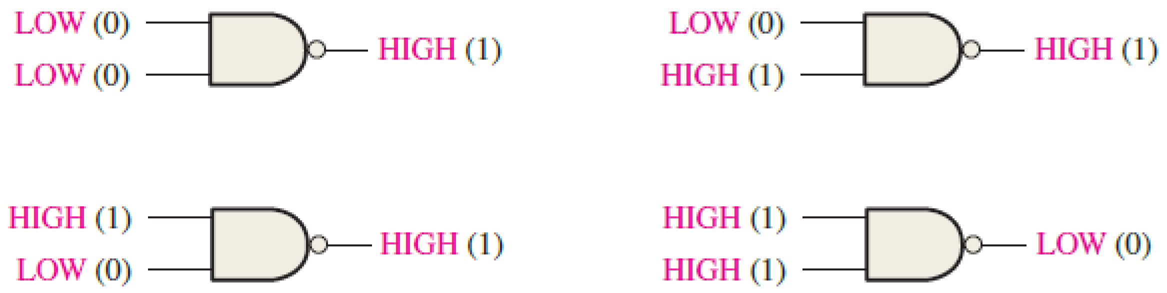
(a) Distinctive shape, 2-input NAND gate and its NOT/AND equivalent

- For a 2-input NAND gate, output X is LOW only when inputs A and B are HIGH;
- X is HIGH when either A or B is LOW, or when both A and B are LOW.

Truth table for a 2-input NAND gate.

Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

1 = HIGH, 0 = LOW.



✓ The NAND gate is the same as the AND gate except the output is inverted

If the two waveforms A and B shown in Figure 3–28 are applied to the NAND gate inputs, determine the resulting output waveform.

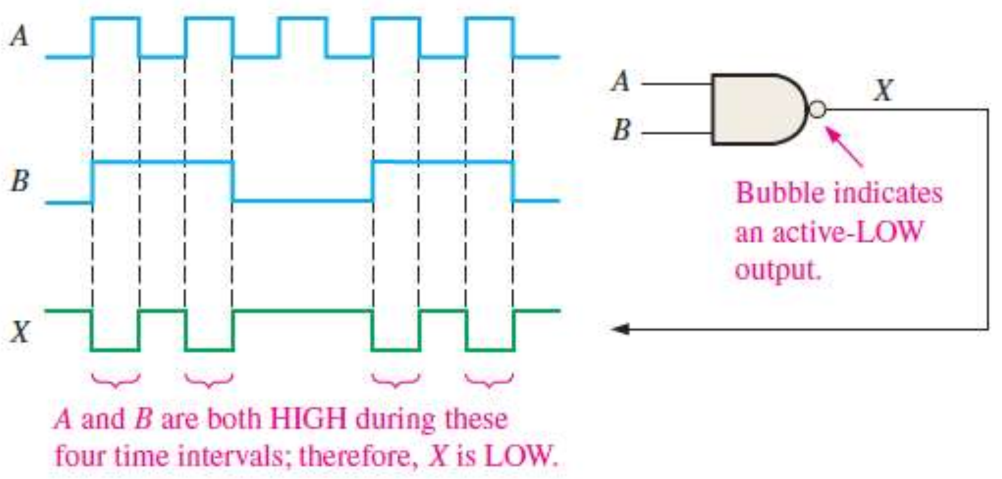

A
B
X
A
B
X
Bubble indicates an active-LOW output.
A and B are both HIGH during these four time intervals; therefore, X is LOW.

FIGURE 3–28

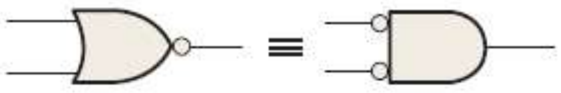
Solution

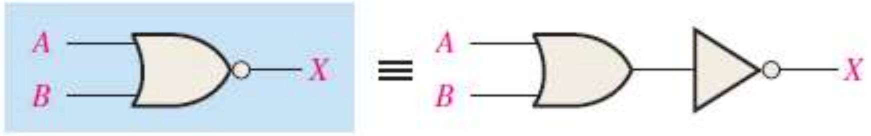
Output waveform X is LOW only during the four time intervals when both input waveforms A and B are HIGH as shown in the timing diagram.

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

- **The Boolean expression for the output of a 2-input NAND gate is** $X = \overline{AB}$
- **This expression says that the two input variables, A and B, are first ANDed and then complemented, as indicated by the bar over the AND expression.**

NOR gate





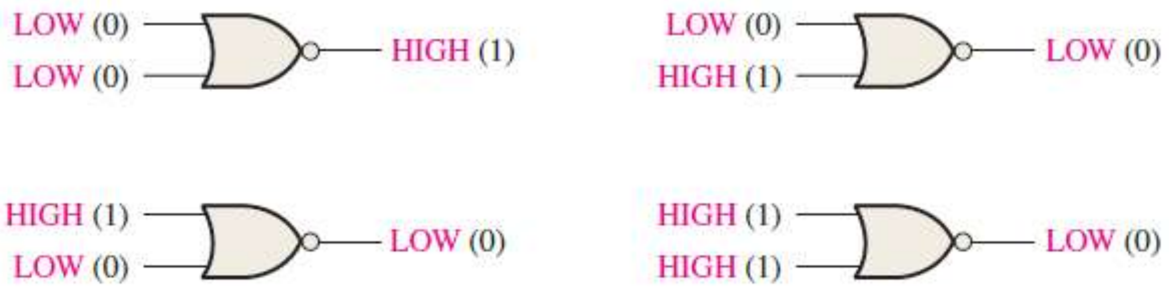
(a) Distinctive shape, 2-input NOR gate and its NOT/OR equivalent

- For a 2-input NOR gate, output X is LOW when either input A or input B is HIGH, or when both A and B are HIGH;
- X is HIGH only when both A and B are LOW

Truth table for a 2-input NOR gate.

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

1 = HIGH, 0 = LOW.



✓ **The NOR is the same as the OR except the output is inverted**

If the two waveforms shown in Figure 3–36 are applied to a NOR gate, what is the resulting output waveform?

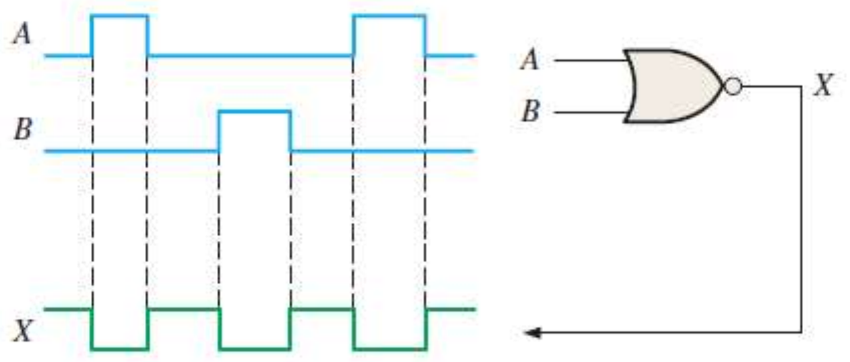


FIGURE 3–36

Solution

Whenever any input of the NOR gate is HIGH, the output is LOW as shown by the output waveform X in the timing diagram.

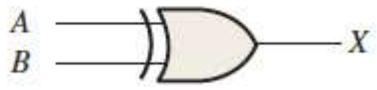
A	B	$\overline{A+B}=X$
0	0	$\overline{0+0}=\overline{0}=1$
0	1	$\overline{0+1}=\overline{1}=0$
1	0	$\overline{1+0}=\overline{1}=0$
1	1	$\overline{1+1}=\overline{1}=0$

- **The Boolean expression for the output of a 2-input NOR gate can be written as**

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

- **This equation says that the two input variables are first ORed and then complemented, as indicated by the bar over the OR expression**

XOR gate

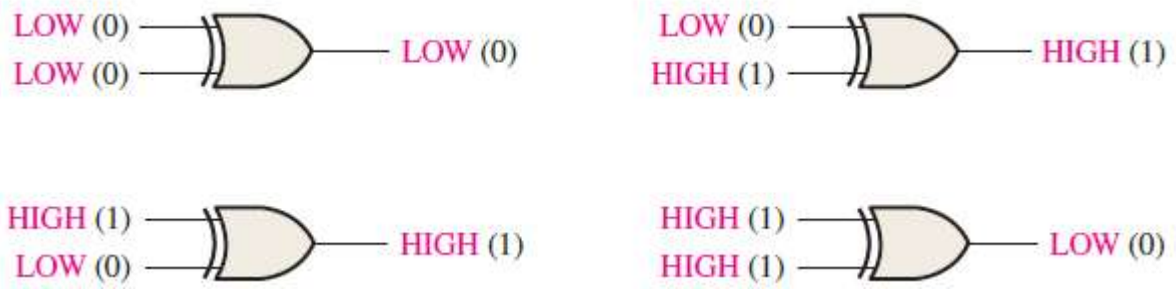


(a) Distinctive shape

Truth table for an exclusive-OR gate.

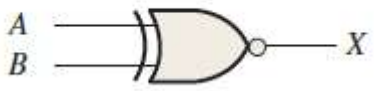
Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

- For an exclusive-OR gate, output *X* is HIGH when input *A* is LOW and input *B* is HIGH, or when input *A* is HIGH and input *B* is LOW;
- *X* is LOW when *A* and *B* are both HIGH or both LOW.



✓ **For an exclusive-OR gate, opposite inputs make the output HIGH**

exclusive-NOR (XNOR) gate

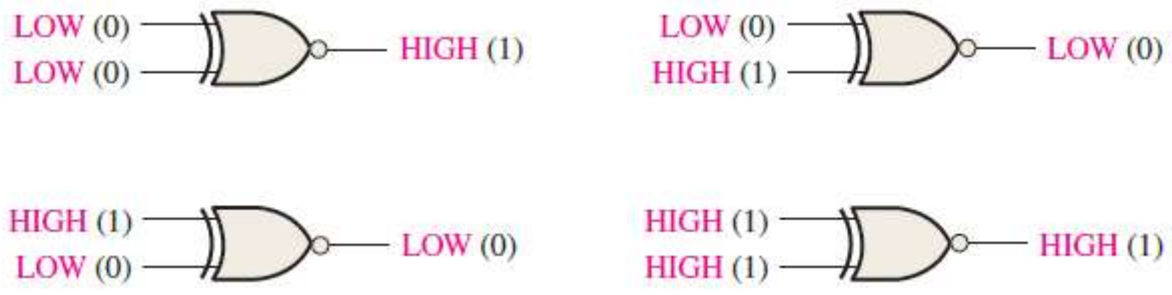


(a) Distinctive shape

Truth table for an exclusive-NOR gate.

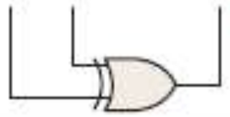
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

- **For an exclusive-NOR gate, output X is LOW when input A is LOW and input B is HIGH, or when A is HIGH and B is LOW;**
- **X is HIGH when A and B are both HIGH or both LOW.**



An XOR gate used to add two bits.

Input Bits		Output (Sum)
A	*B*	Σ
0	0	0
0	1	1
1	0	1
1	1	0 (without the 1 carry bit)



Determine the output waveforms for the XOR gate and for the XNOR gate, given the input waveforms, *A* and *B*, in Figure 3–48.

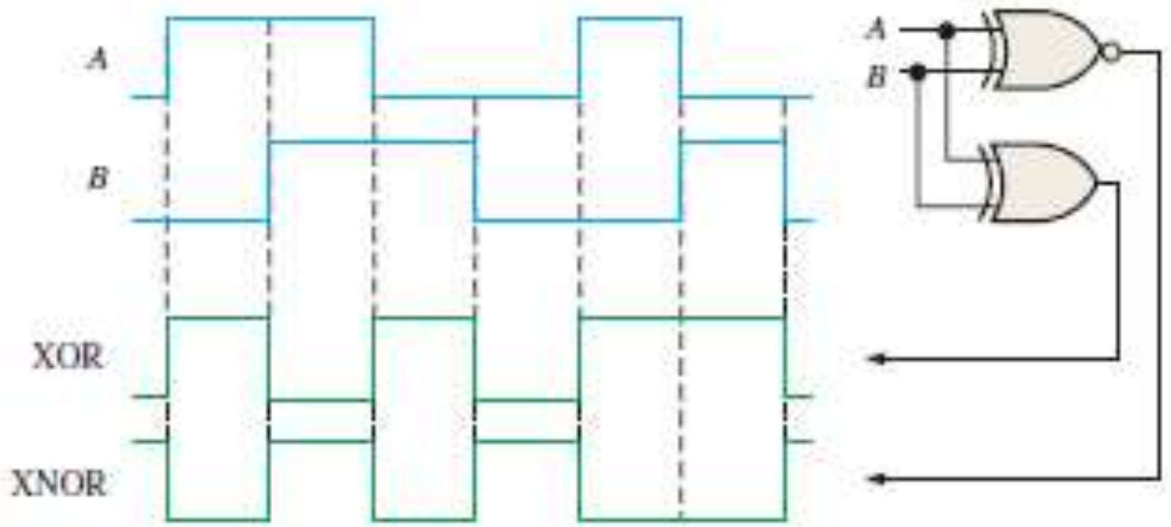


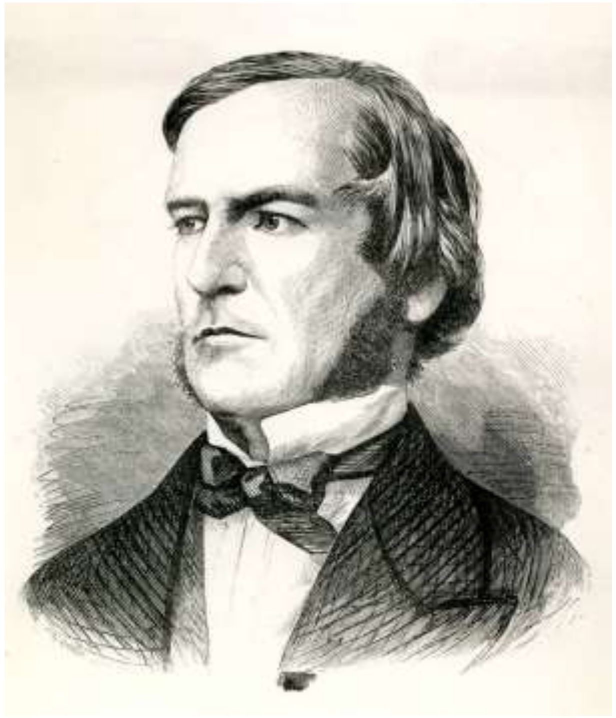
FIGURE 3–48

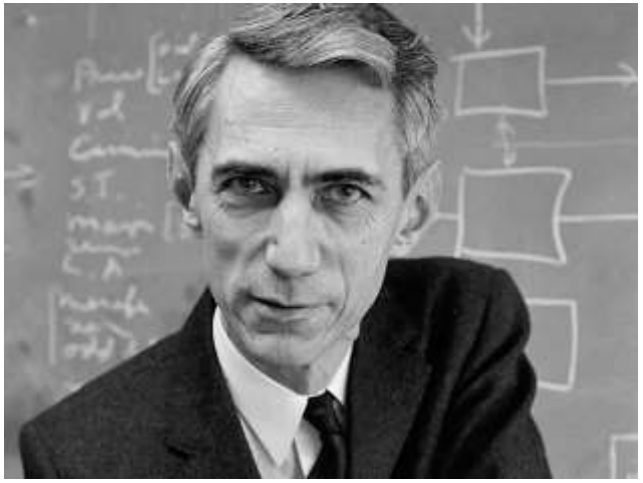
Solution

The output waveforms are shown in Figure 3–48. Notice that the XOR output is HIGH only when both inputs are at opposite levels. Notice that the XNOR output is HIGH only when both inputs are the same.

Boolean Algebra

- In 1854, George Boole published a work titled An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities.
- It was in this publication that a “logical algebra,” known today as Boolean algebra, was formulated.
- Boolean algebra is a convenient and systematic way of expressing and analyzing the operation of logic circuits.





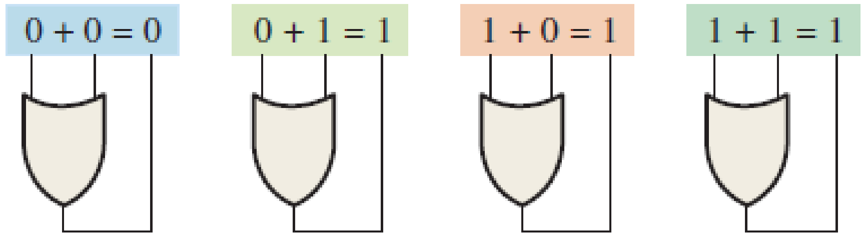
- An MIT student Claude Shannon applied Boole’s work to the analysis and design of logic circuits
- Made Boole’s work useful and famous
- Claude Shannon is also well known for something else – can you find??

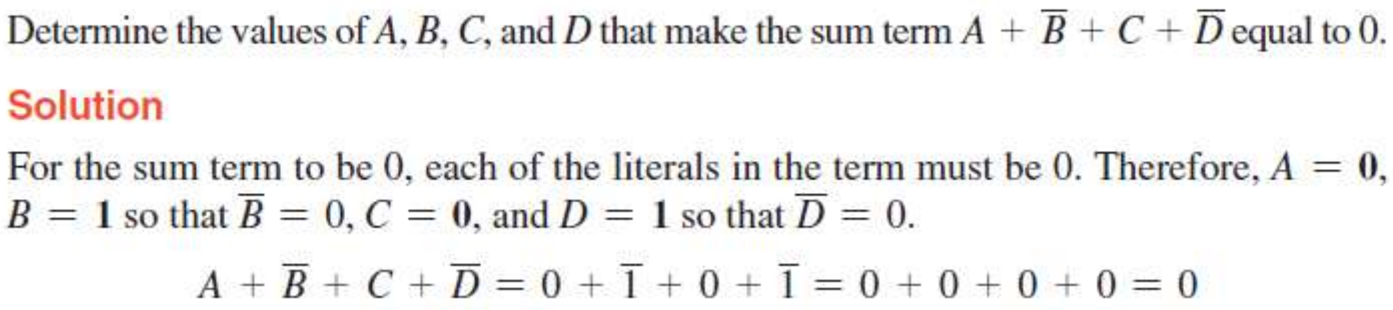
- A variable is a symbol (usually an italic uppercase letter or word) used to represent an action, a condition, or data. Any single variable can have only a 1 or a 0 value.
- The complement is the inverse of a variable and is indicated by a bar over the variable (overbar).
- A literal is a variable or the complement of a variable

Boolean Arithmetic

Boolean Addition

- **Boolean addition is equivalent to the OR operation**
- **In Boolean algebra, a sum term is a sum of literals**




Determine the values of A, B, C, and D that make the sum term $A+\overline{B}+C+\overline{D}$ equal to 0.

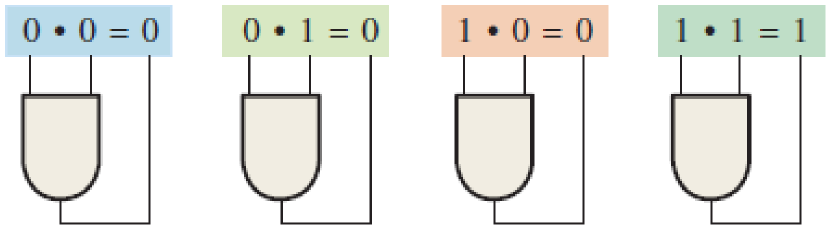
Solution

For the sum term to be 0, each of the literals in the term must be 0. Therefore, $A = \mathbf{0}$, $B = \mathbf{1}$ so that $\overline{B} = 0$, $C = \mathbf{0}$, and $D = \mathbf{1}$ so that $\overline{D} = 0$.

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

Boolean Multiplication

- **Boolean multiplication is equivalent to the AND operation**
- **In Boolean algebra, a product term is the product of literals**



Determine the values of A, B, C, and D that make the product term $A\overline{B}C\overline{D}$ equal to 1.

Solution

For the product term to be 1, each of the literals in the term must be 1. Therefore, $A = \mathbf{1}$, $B = \mathbf{0}$ so that $\overline{B} = 1$, $C = \mathbf{1}$, and $D = \mathbf{0}$ so that $\overline{D} = 1$.

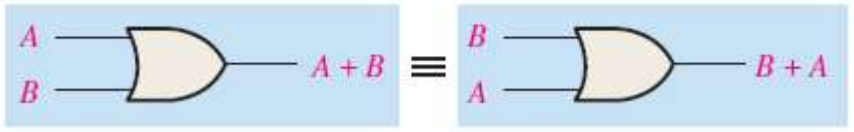
$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Laws of Boolean Algebra

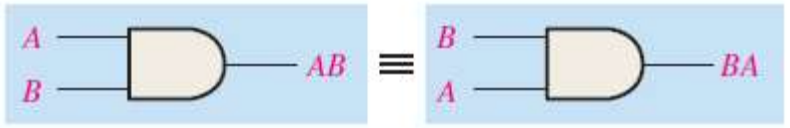
Commutative Law

Commutative law of addition $A + B = B + A$

Commutative law of multiplication $AB = BA$



Application of commutative law of addition.

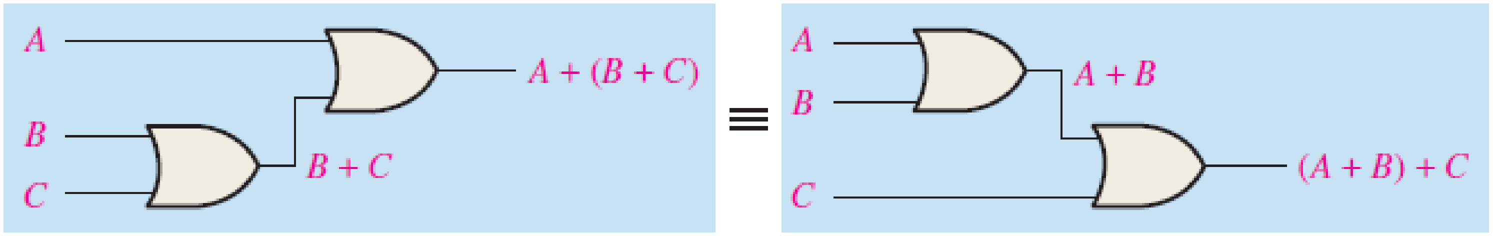


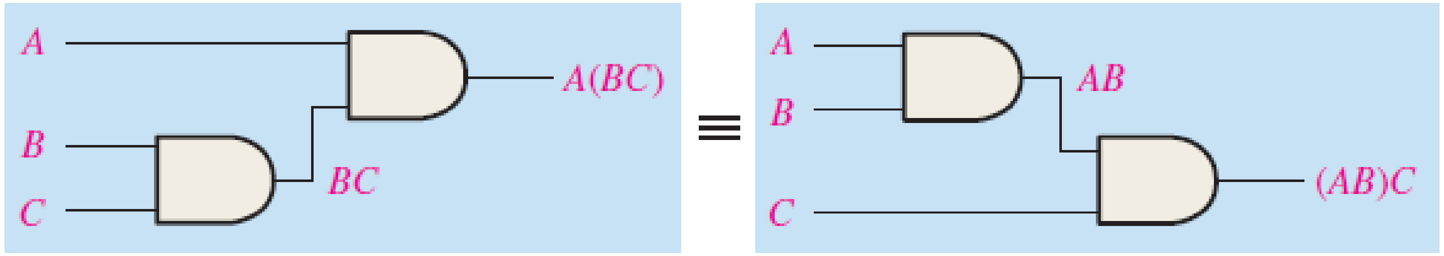
Application of commutative law of multiplication.

Associative Law

Associative law of addition $A + (B + C) = (A + B) + C$

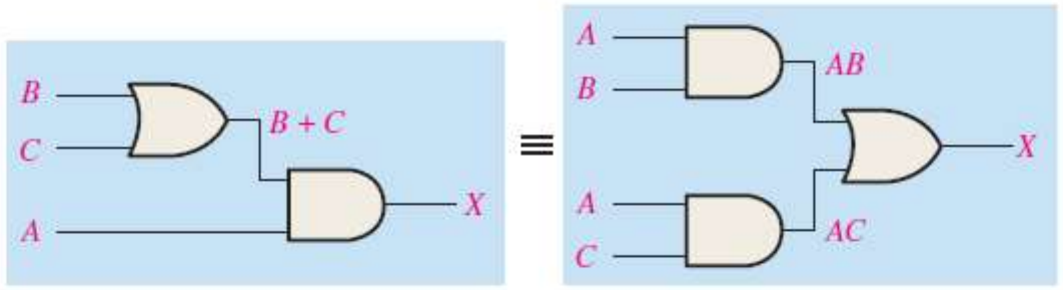
Associative law of multiplication $A(BC) = (AB)C$





Distributive Law

Distributive Law $A(B + C) = AB + AC$



$X = A(B + C)$ $X = AB + AC$

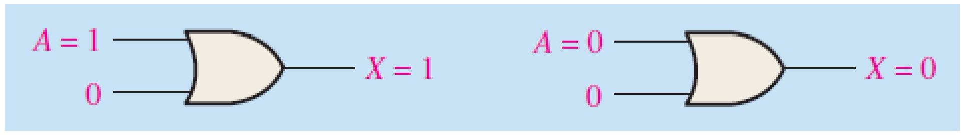
Basic Rules

Basic rules of Boolean algebra.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

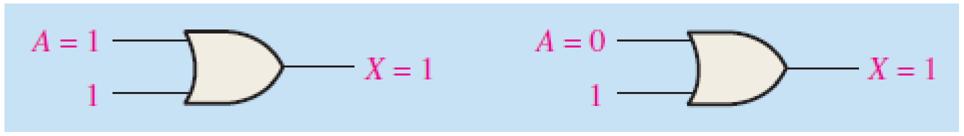
A, *B*, or *C* can represent a single variable or a combination of variables.

Rule 1: $A + 0 = A$



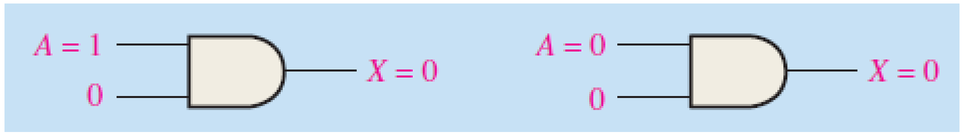
$X = A + 0 = A$

Rule 2: $A + 1 = 1$



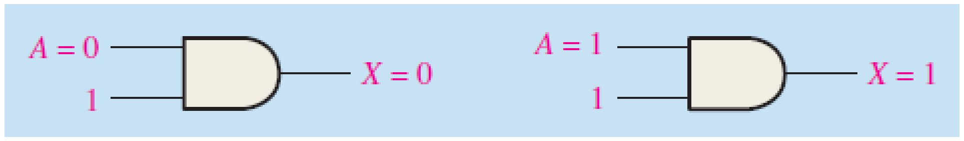
$X = A + 1 = 1$

Rule 3: $A \cdot 0 = 0$



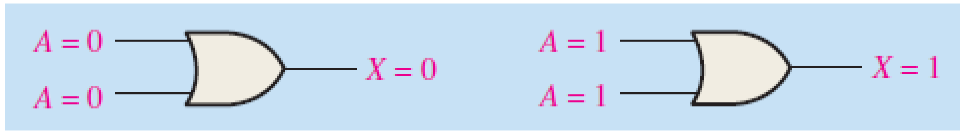
$X = A \bullet 0 = 0$

Rule 4: $A \cdot 1 = A$



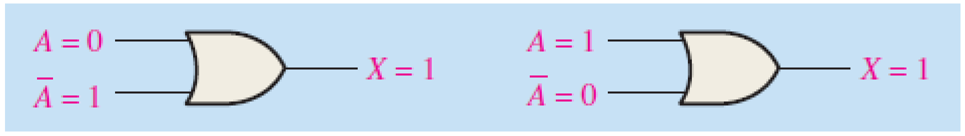
$X = A \bullet 1 = A$

Rule 5: $A + A = A$



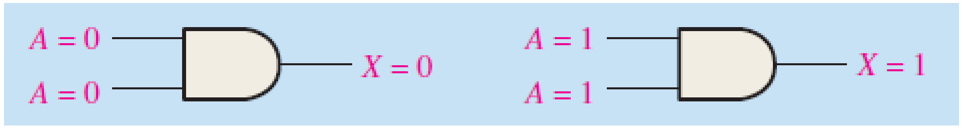
$X = A + A = A$

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



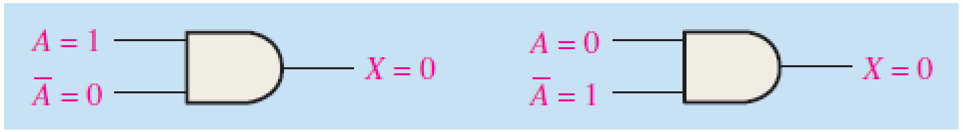
$X = A + \bar{A} = 1$

Rule 7: $A \cdot A = A$



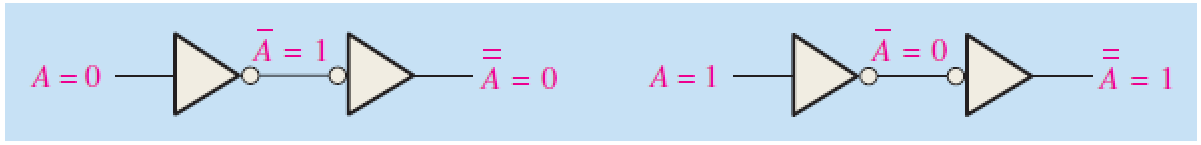
$X = A \cdot A = A$

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



$X = A \cdot \bar{A} = 0$

Rule 9: $\bar{\bar{A}} = A$



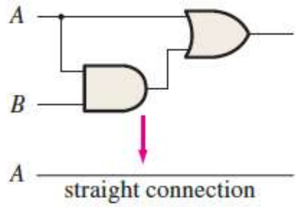
$\bar{\bar{A}} = A$

Rule 10: $A + AB = A$

$A + AB = A \cdot 1 + AB = A(1 + B)$	Factoring (distributive law)
$= A \cdot 1$	Rule 2: $(1 + B) = 1$
$= A$	Rule 4: $A \cdot 1 = A$

A	B	AB	$A + AB$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal

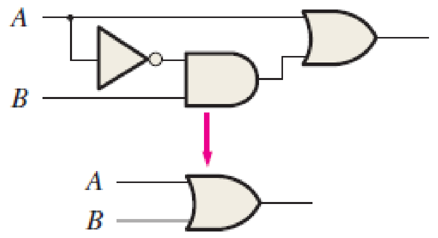


Rule 11: $A + \bar{A}B = A + B$

$A + \bar{A}B = (A + AB) + \bar{A}B$	Rule 10: $A = A + AB$
$= (AA + AB) + \bar{A}B$	Rule 7: $A = AA$
$= AA + AB + A\bar{A} + \bar{A}B$	Rule 8: adding $A\bar{A} = 0$
$= (A + \bar{A})(A + B)$	Factoring
$= 1 \cdot (A + B)$	Rule 6: $A + \bar{A} = 1$
$= A + B$	Rule 4: drop the 1

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

equal

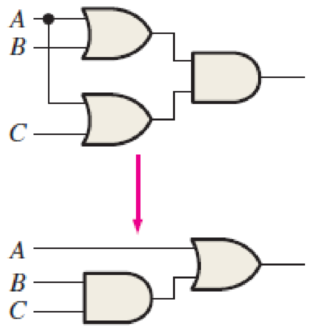


Rule 12: $(A + B)(A + C) = A + BC$

$$\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC && \text{Distributive law}\\
&= A + AC + AB + BC && \text{Rule 7: } AA = A\\
&= A(1 + C) + AB + BC && \text{Factoring (distributive law)}\\
&= A \cdot 1 + AB + BC && \text{Rule 2: } 1 + C = 1\\
&= A(1 + B) + BC && \text{Factoring (distributive law)}\\
&= A \cdot 1 + BC && \text{Rule 2: } 1 + B = 1\\
&= A + BC && \text{Rule 4: } A \cdot 1 = A
\end{aligned}$$

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



Demorgan's Theorems

Theorem 1 : **The complement of a product of variables is equal to the sum of the complements of the variables**

OR

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

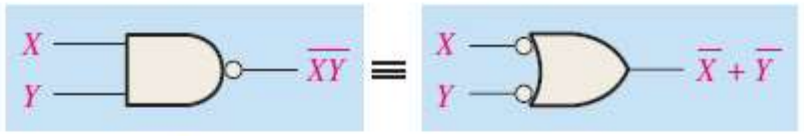
$$\overline{XY} = \overline{X} + \overline{Y}$$

Theorem 2 : **The complement of a sum of variables is equal to the product of the complements of the variables**

OR

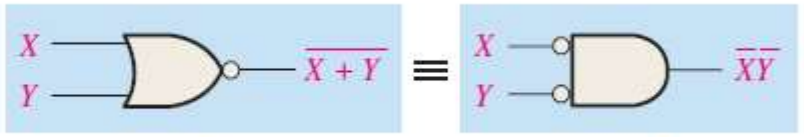
The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

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



NAND Negative-OR

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



NOR Negative-AND

Inputs		Output	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

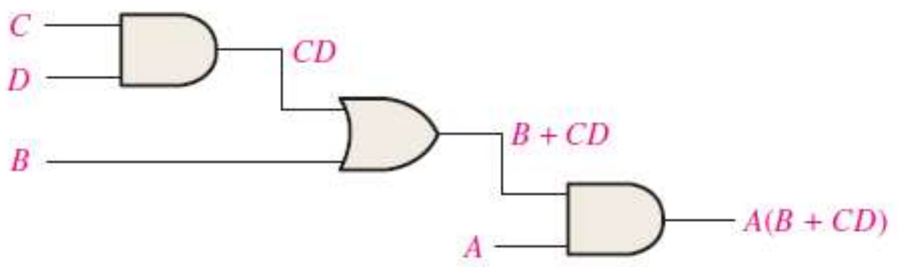
- Gate equivalencies and the corresponding truth tables that illustrate DeMorgan's theorems. Notice the equality of the two output columns in each table.
- This shows that the equivalent gates perform the same logic function

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

Solution

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

Consider the implementation of a Boolean Expression:



✓ **A combinational logic circuit can be described by a 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

Logic Simplification

Apply the laws, rules, and theorems of Boolean algebra to simplify general expressions

Using Boolean algebra techniques, simplify this expression:

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

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

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

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

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

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

$$AB + AC + B + BC$$

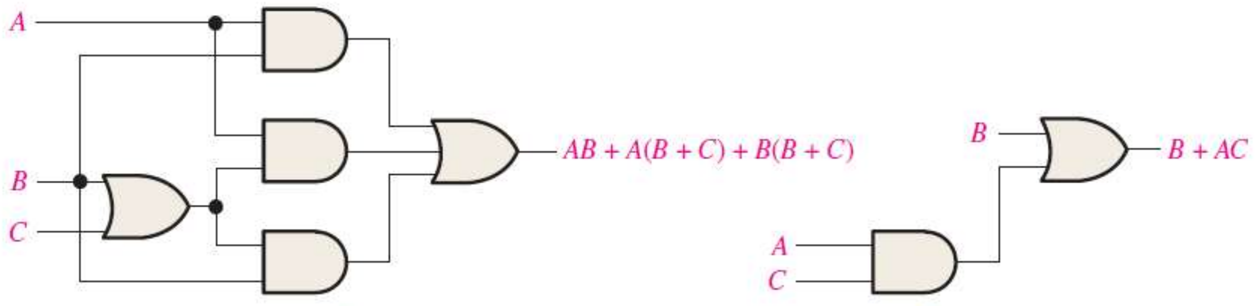
Step 4: Apply rule 10 ($B + BC = B$) to the last two terms.

$$AB + AC + B$$

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

$$B + AC$$

At this point the expression is simplified as much as possible. Once you gain experience in applying Boolean algebra, you can often combine many individual steps.



(a) These two circuits are equivalent. (b)

✓ **Simplification means fewer gates for the same function.**

Standard forms of Boolean Expressions

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

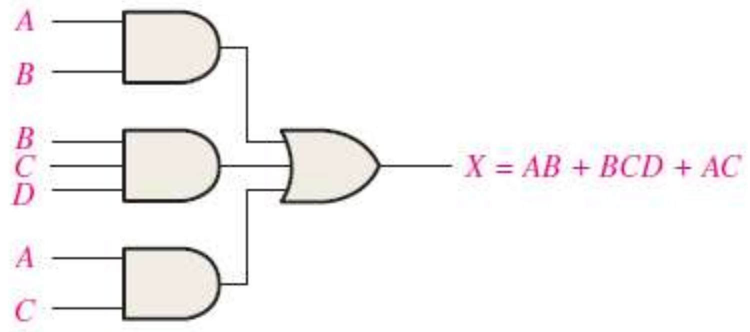
$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

- The **domain** of a general Boolean expression is the set of variables contained in the expression in either complemented or uncomplemented form

$\overline{A}B + \dot{A}\overline{B}C \longrightarrow$ Domain is set of variables A, B,C

$AB\overline{C} + C\overline{D}E + \overline{B}C\overline{D} \longrightarrow$ Domain is set of variables A, B,C, D, E

AND/OR implementation of SOP $\longleftarrow$



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

✓ **An SOP expression can be implemented with one OR gate and two or more AND gates**

- **Any logic expression can be changed into SOP** form by applying Boolean algebra techniques.
- For example, the expression $A(B + CD)$ can be converted to SOP form by applying the distributive law:

$$\boldsymbol{A(B + CD) = AB + ACD}$$

- **A standard SOP expression** is one in which all the variables in the domain appear in each product term in the expression.

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

Example:

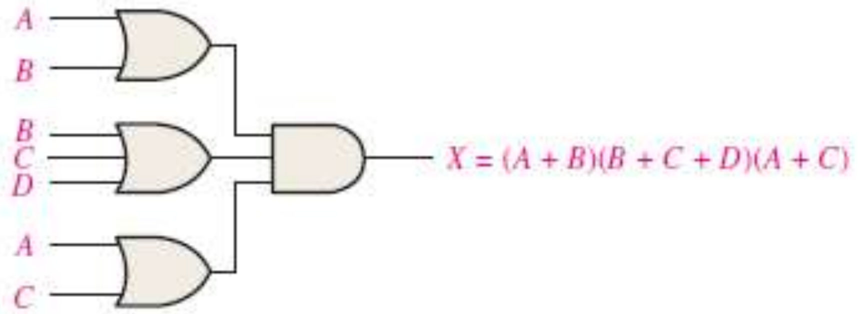
Convert the following Boolean expression into standard SOP form:

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

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

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

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



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

- **A standard POS expression** is one in which all the variables in the domain appear in each sum term in the expression

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

Example:

Convert the following Boolean expression into standard POS form:

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

Solution

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

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

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

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

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

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

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.

Convert the following SOP expression to an equivalent POS expression:

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

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

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

Boolean Expressions & Truth Tables

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

TABLE 4–6

Inputs			Output	
A	B	C	X	Product Term
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; 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.

Determine the truth table for the following standard POS expression:

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

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; $A + \overline{B} + C$: 010; $A + \overline{B} + \overline{C}$: 011; $\overline{A} + B + \overline{C}$: 101; and $\overline{A} + \overline{B} + C$: 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.

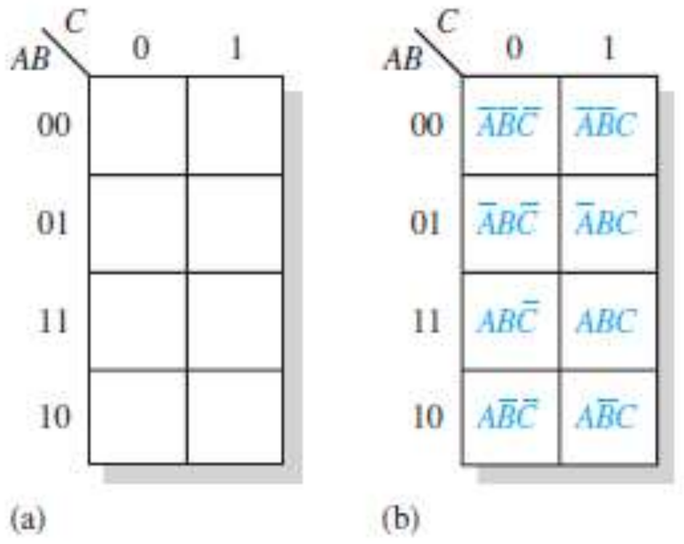
TABLE 4–7

Inputs			Output	
A	*B*	*C*	*X*	Sum Term
0	0	0	0	$(A + B + C)$
0	0	1	1	
0	1	0	0	$(A + \overline{B} + C)$
0	1	1	0	$(A + \overline{B} + \overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A} + B + \overline{C})$
1	1	0	0	$(\overline{A} + \overline{B} + C)$
1	1	1	1	

✓ **Similar way, you can determine Standard Expressions from a given Truth Table**

The Karnaugh Map

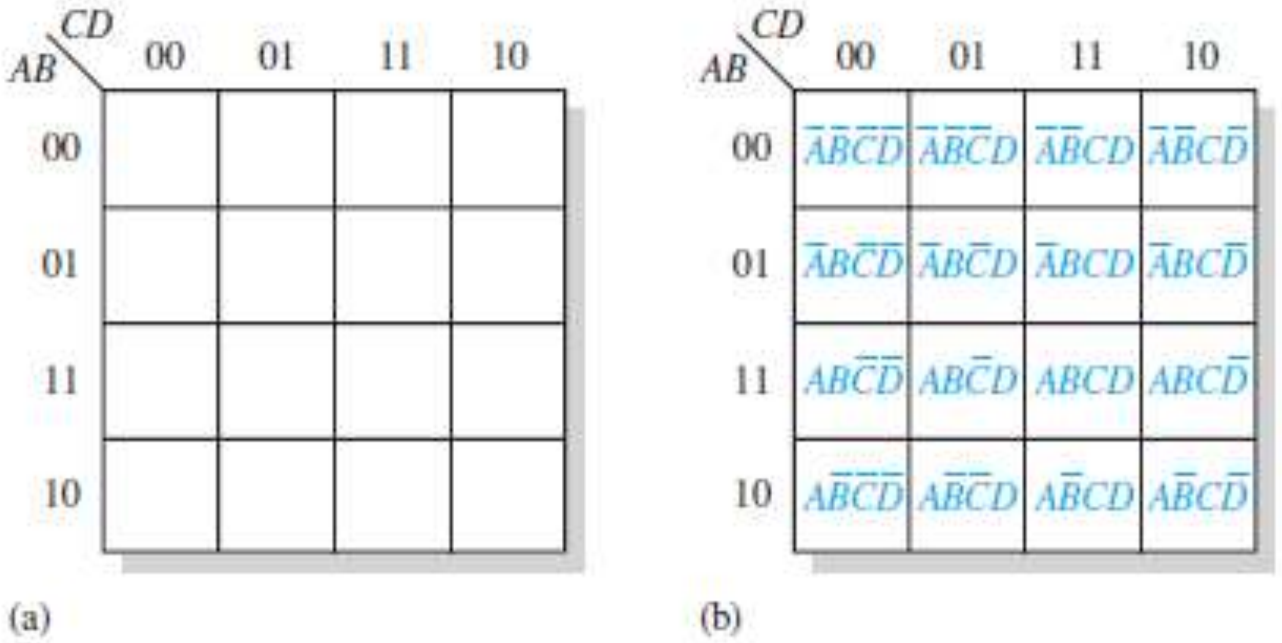
- **Karnaugh map** is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value.
- Instead of being organized into columns and rows like a truth table, the Karnaugh map is an array of cells in which each cell represents a binary value of the input variables.



C
AB
0
1
00
01
11
10
(a)
(b)
$\overline{A}\,\overline{B}\,\overline{C}$
$\overline{A}\,\overline{B}C$
$\overline{A}B\overline{C}$
$\overline{A}BC$
$AB\overline{C}$
ABC
$A\overline{B}\,\overline{C}$
$A\overline{B}C$

A 3-variable Karnaugh map showing Boolean product terms for each cell.

✓ **The purpose of a Karnaugh map is to simplify a Boolean expression**



CD
AB
00
01
11
10
(a)
(b)
$\overline{A}\overline{B}\overline{C}\overline{D}$
$\overline{A}\overline{B}\overline{C}D$
$\overline{A}\overline{B}CD$
$\overline{A}\overline{B}C\overline{D}$
$\overline{A}B\overline{C}\overline{D}$
$\overline{A}B\overline{C}D$
$\overline{A}BCD$
$\overline{A}BC\overline{D}$
$AB\overline{C}\overline{D}$
$AB\overline{C}D$
$ABCD$
$ABC\overline{D}$
$A\overline{B}\overline{C}\overline{D}$
$A\overline{B}\overline{C}D$
$A\overline{B}CD$
$A\overline{B}C\overline{D}$

A 4-variable Karnaugh map.

- ✓ **Cells that differ by only one variable are adjacent.**
- ✓ **Cells with values that differ by more than one variable are not adjacent**
- ✓ **Adjacent cells could be combined**
- ✓ **For more than 4 variables, other methods such as The Quine-McCluskey Method is used.**

Karnaugh Map SOP minimization

STEP 1 : Convert to Standard SOP

STEP 2 : Map standard SOP to K-map

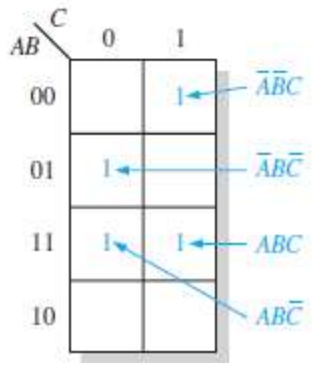
Map the following standard SOP expression on a Karnaugh map:

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

Solution

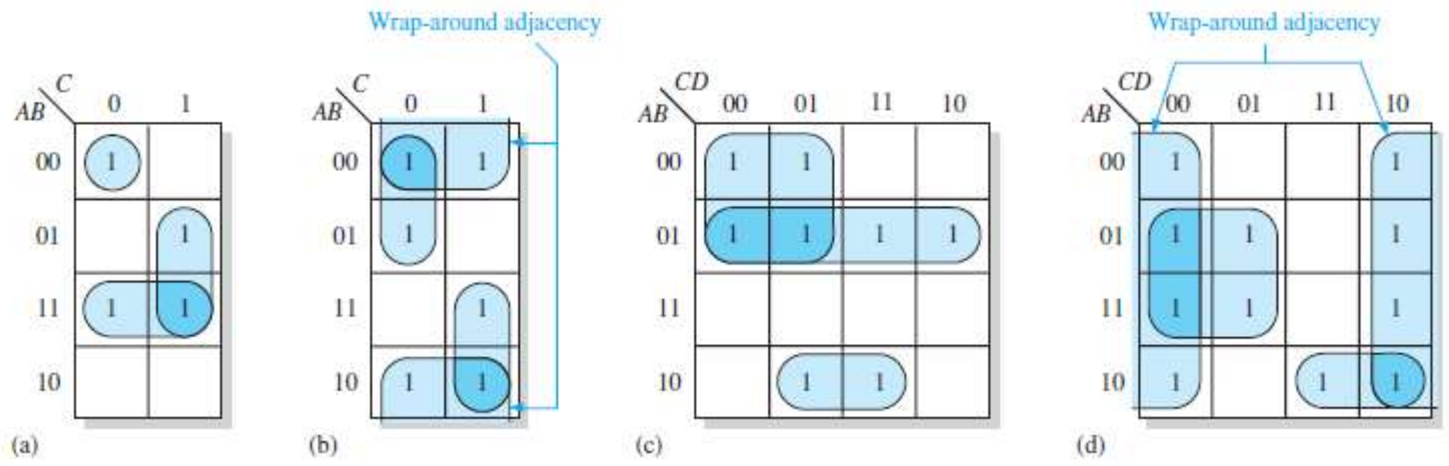
Evaluate the expression as shown below. Place a 1 on the 3-variable Karnaugh map in Figure 4–29 for each standard product term in the expression.

$$\begin{array}{ccccccc} \overline{A}\,\overline{B}C & + & \overline{A}B\overline{C} & + & AB\overline{C} & + & ABC \\ 001 & & 010 & & 110 & & 111 \end{array}$$



STEP 3 : Minimization

Grouping the 1s



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, 23 = 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.

Determining the Minimum SOP Expression from the Map

Determine the product terms for the Karnaugh map in Figure 4–35 and write the resulting minimum SOP expression.

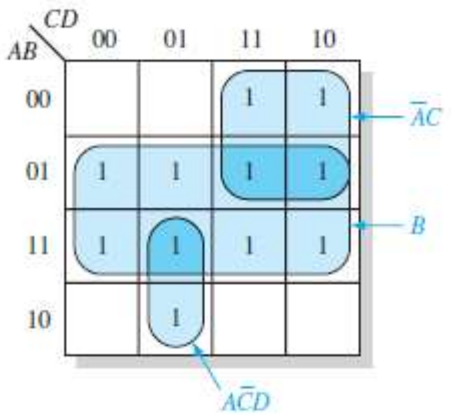


FIGURE 4–35

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

Resulting minimum SOP expression

Determining the Minimum SOP Expression from the Map

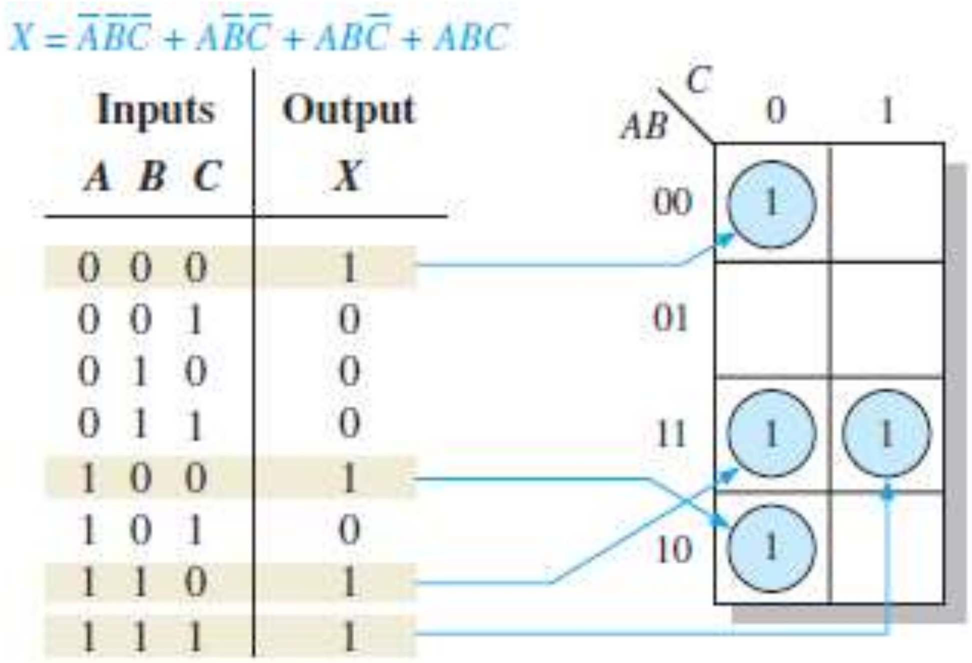
1. Group the cells that have 1s. Each group of cells containing 1s creates one product term composed of all variables that occur in only one form (either uncomplemented or complemented) within the group. Variables that occur both uncomplemented and complemented within the group are eliminated. These 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.

Mapping Directly from Truth Table



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

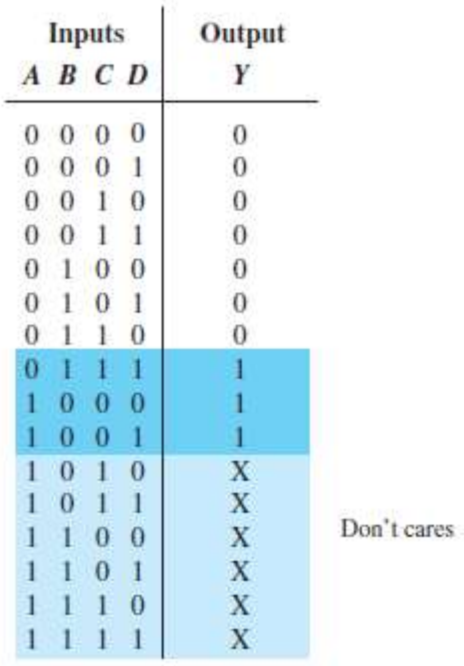
Inputs			Output
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\begin{array}{c|cc} AB \backslash C & 0 & 1 \\ \hline 00 & 1 & \\ 01 & & \\ 11 & 1 & 1 \\ 10 & 1 & \end{array}$$

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

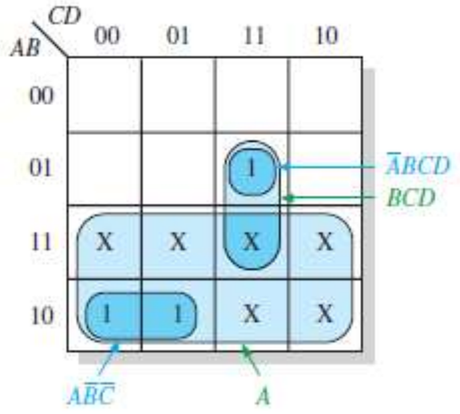
“Don’t Care” Conditions

- They can be treated as “don’t care” terms with respect to their effect on the output.
- That is, for these “don’t care” terms either a 1 or a 0 may be assigned to the output;
- It really does not matter since they will never occur. Eg: invalid codes of BCD



Inputs				Output
A	B	C	D	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” $Y = A + BCD$

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

Karnaugh Map POS minimization

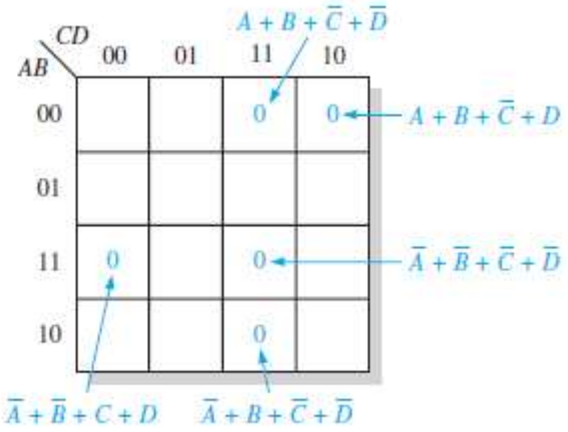
Map the following standard POS expression on a Karnaugh map:

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

Solution

Evaluate the expression as shown below and place a 0 on the 4-variable Karnaugh map in Figure 4–44 for each standard sum term in the expression.

$$\underset{1100}{(\overline{A} + \overline{B} + C + D)}\underset{1011}{(\overline{A} + B + \overline{C} + \overline{D})}\underset{0010}{(A + B + \overline{C} + D)}\underset{1111}{(\overline{A} + \overline{B} + \overline{C} + \overline{D})}\underset{0011}{(A + B + \overline{C} + \overline{D})}$$



CD
AB
00
01
11
10
0
$A+B+\overline{C}+\overline{D}$
$A+B+\overline{C}+D$
$\overline{A}+\overline{B}+\overline{C}+\overline{D}$
$\overline{A}+\overline{B}+C+D$
$\overline{A}+B+\overline{C}+\overline{D}$

FIGURE 4–44

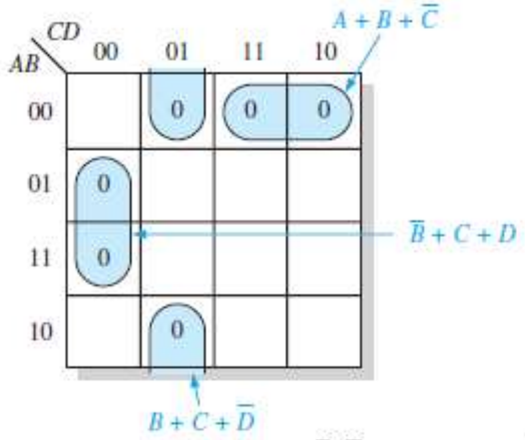
The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms.

Converting Between POS and SOP Using the Karnaugh Map

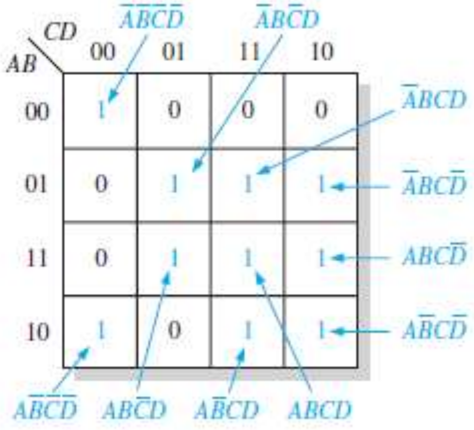
Using a Karnaugh map, convert the following standard POS expression into a minimum POS expression, a standard SOP expression, and a minimum SOP expression.

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

The 0s for the standard POS expression are mapped and grouped to obtain the minimum POS expression in Figure 4–47(a). In Figure 4–47(b), 1s are added to the cells that do not contain 0s. From each cell containing a 1, a standard product term is obtained as indicated. These product terms form the standard SOP expression. In Figure 4–47(c), the 1s are grouped and a minimum SOP expression is obtained.

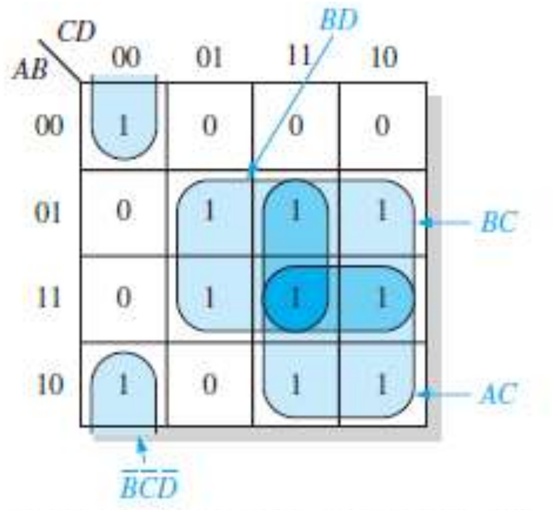


(a) Minimum POS: $(A + B + C)(\overline{B} + \overline{C} + D)(B + C + \overline{D})$



CD
AB
00
01
11
10
1
0
$\overline{A}\overline{B}\overline{C}\overline{D}$
$\overline{A}B\overline{C}D$
$\overline{A}BCD$
$\overline{A}BC\overline{D}$
$ABC\overline{D}$
$A\overline{B}C\overline{D}$
$A\overline{B}\overline{C}\overline{D}$
$AB\overline{C}D$
$A\overline{B}CD$
$ABCD$

(b) Standard SOP:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D} +$
$A\overline{B}\overline{C}\overline{D} + AB\overline{C}D + A\overline{B}CD + ABCD$

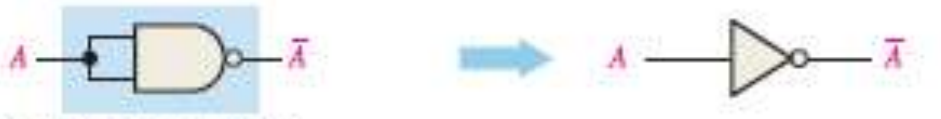


(c) Minimum SOP: $AC + BC + BD + \overline{B}\overline{C}\overline{D}$

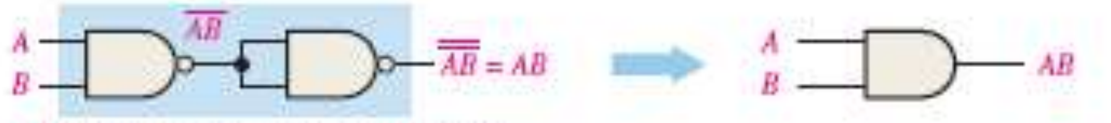
Universal Gates

- All Boolean functions can be implemented using the set {AND, OR, NOT}
- Gates which can implement any Boolean function without the need to use any other type of gate
- **NAND and NOR are universal gates**
- Really?

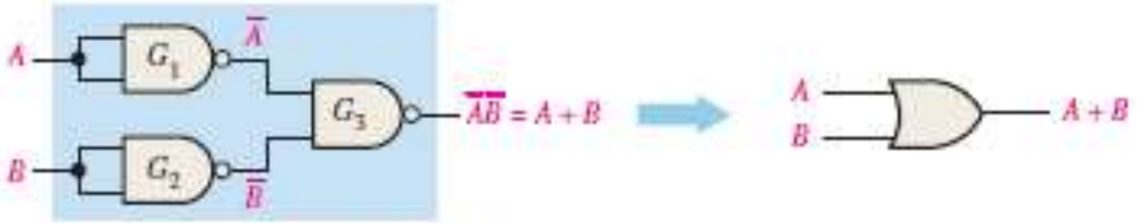
Universal Gates



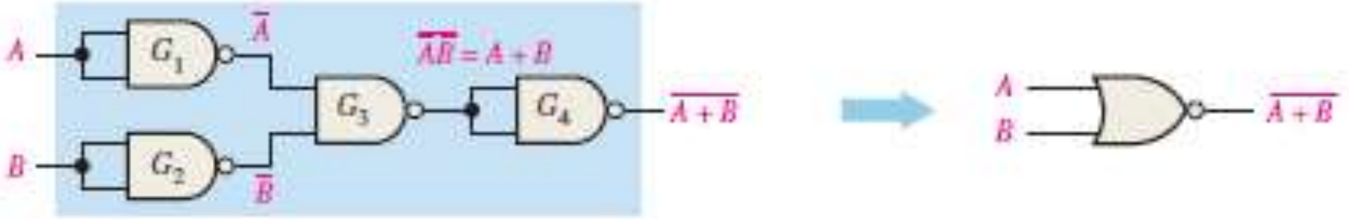
(a) One NAND gate used as an inverter



(b) Two NAND gates used as an AND gate



(c) Three NAND gates used as an OR gate

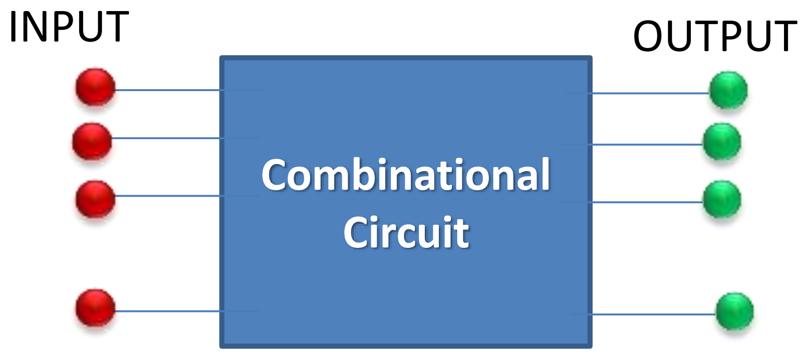


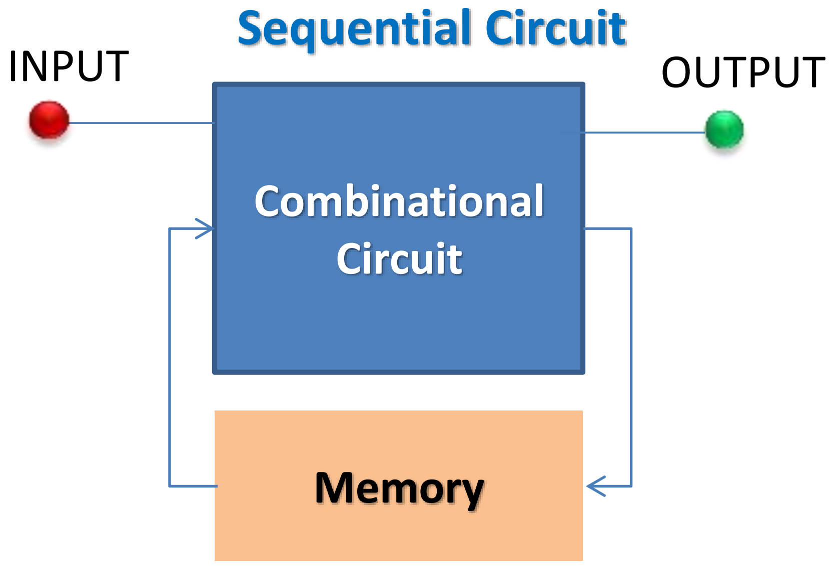
(d) Four NAND gates used as a NOR gate

- Please verify them yourself!!!
- In a similar way, prove that NOR gate is universal too

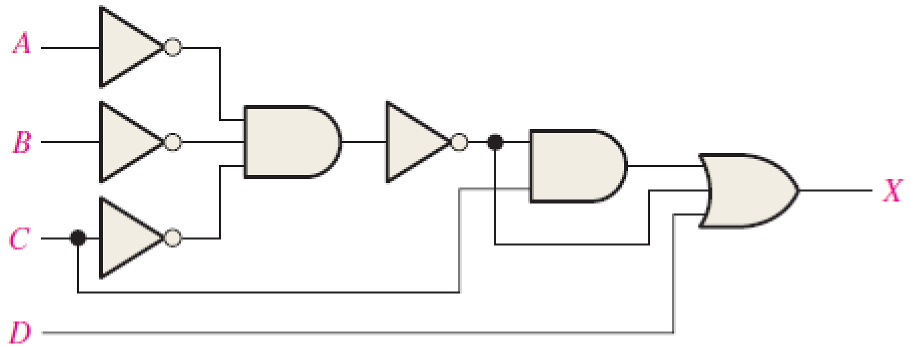
Combinational circuits

- Combinational logic refers to circuits whose output is a function of the *present value of the inputs* only.
- As soon as inputs are changed, the information about the previous inputs is lost, that is, combinational logic circuits have no memory
- Sequential logic circuits are those whose outputs are also dependent upon past inputs, and hence outputs.
- In other words the output of a sequential circuit may depend upon its previous outputs and so in effect has some form of "memory"





- **Reduce the combinational logic circuit to a minimum form**



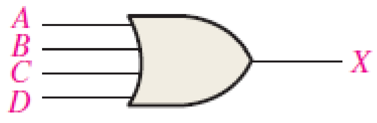
Solution

The expression for the output of the circuit is

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

Applying DeMorgan's theorem and Boolean algebra,

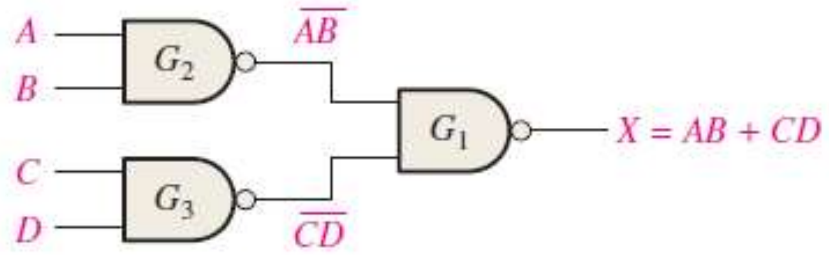
$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \cancel{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$



$\longrightarrow$ **Final Simplified Circuit**

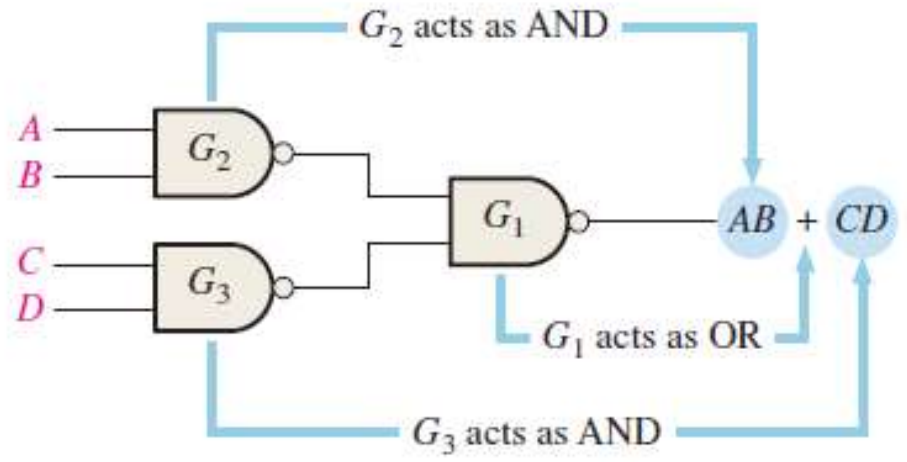
- **Combinational logic could be implemented using universal gates**

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

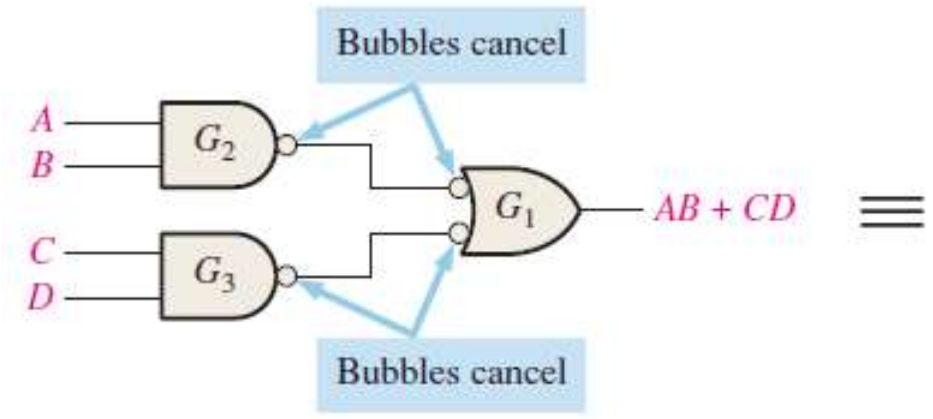


NAND logic for $X = AB + CD$.

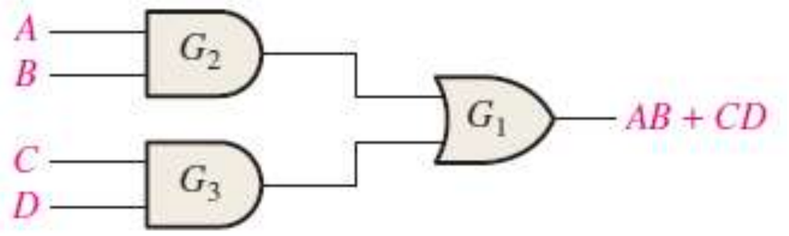
- **Universal Gate Implementation Understanding**



(a) Original NAND logic diagram showing effective gate operation relative to the output expression

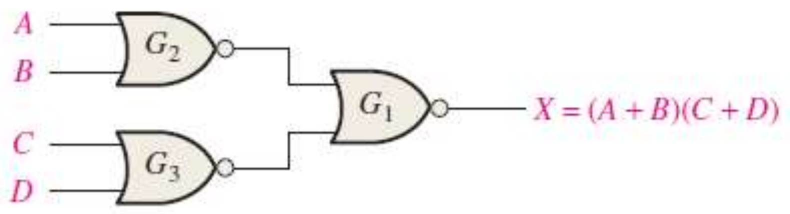


(b) Equivalent NAND/Negative-OR logic diagram

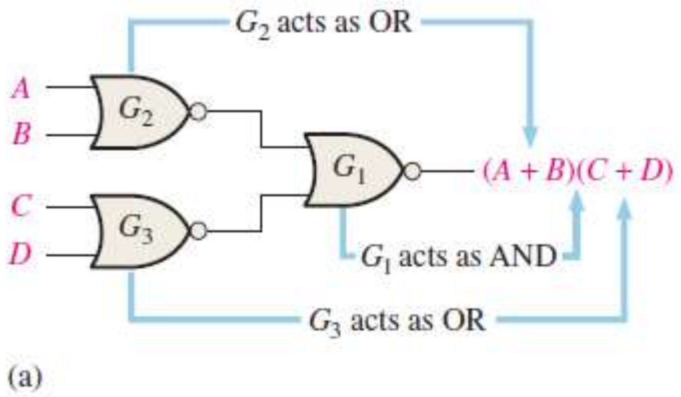


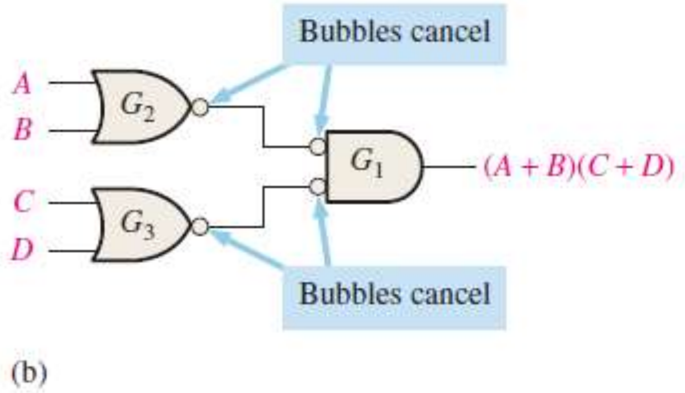
(c) AND-OR equivalent

- **NOR logic**



NOR logic for $X = (A + B)(C + D)$.


G_2 acts as OR
A
B
C
D
G_2
G_3
G_1
$(A + B)(C + D)$
G_1 acts as AND
G_3 acts as OR
(a)


Bubbles cancel
A
B
C
D
G_2
G_3
G_1
$(A + B)(C + D)$
Bubbles cancel
(b)