Lecture (05)
Boolean Algebra and Logic Gates

By:

Dr. Ahmed ElShafee

Minterms and Maxterms

- consider two binary variables *x* and *y* combined with an AND operation.
- Since eachv ariable may appear in either form, there are four possible combinations: *xy*, *x'y*, *xy'*, and *x'y'*.
- Each of these four AND terms is called a *minterm*, or a *standard product*
- *For any n* variables can be combined to form 2^n minterms.
- The binary numbers from 0 to 2^n - 1 are listed under the *n* variables
- *n* variables forming an OR term, provide 2^n possible combinations, called *maxterms*, or *standard sums*

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

- **A Boolean function can be expressed algebraically from a given truth table by forming**
  - **a minterm for each combination of the variables that produces a 1 in the function**
  - **and then taking the OR of all those terms.**

Any Boolean function can be expressed as a sum of minterms

- $f1$ in Table is determined by expressing the combinations 001, 100, and 111 as $x'y'z$, $xy'z'$, and xyz, respectively.
- Since each one of these minterms results in $f1 = 1$,

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

- *Functions of Three Variables*

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

- The complement of $f1$ is, sum of minterms that produce 0

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

- If we take the complement of f '

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

- *Functions of Three Variables*

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

$$f_2 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)$$
$$= M_0M_1M_2M_4$$

- **Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form* .**

Expressing Boolean function as sum of minterm

A. Sum of minterms

Methods 1:

- If the function is not in this form, it can be made so by first expanding the expression into a sum of AND terms.
- Each term is then inspected to see if it contains all the variables. If it misses one or more variables, it is ANDed with an expression such as *x* + *x'*, where *x* is one of the missing variables

Example 01

- Express the Boolean function *F* = *A* + *B'C* as a sum of minterms. The function has three variables: *A*, *B*, and *C*.

- The first term *A* is missing two variables

$$A = A(B + B') = AB + AB'$$

  - This function is still missing one variable, so

$$A = AB(C + C') + AB'(C + C')$$
$$= ABC + ABC' + AB'C + AB'C'$$

  - The second term *BC* is missing one variable;

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

- Combining all terms, we have

$$F = A + B'C$$
$$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$$

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

- But *AB'C* appears twice, and according to theorem (*x* + *x* = *x*),

$$F = A'B'C + AB'C + AB'C + ABC' + ABC$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$

- it is sometimes convenient to express the function in the following brief notation:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

- $\sum$ stands for the ORing

Method 2:

- An alternative procedure for deriving the minterms of a Boolean function is to obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table

Example 01

- Express the Boolean function *F* = *A* + *B'C* as a sum of minterms. The function has three variables: *A*, *B*, and *C*.

$$F = A + B'C$$

Truth Table for F = A + B'C

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- From the truth table, we can then read the five minterms of the function to be 1, 4, 5, 6, and 7.

B. Products of maxterm

- To express a Boolean function as a product of maxterms, done by using the distributive law,
- $x + yz = (x + y)(x + z)$.

Example 02

- Express the Boolean function $F = xy + x'z$ as a product of maxterms.

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

- The function has three variables: x, y, and z. Each OR term is missing one variable; therefore

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

- Combining all the terms and removing those which appear more than once

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

- convenient way to express this function is as follows

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

Conversion between Canonical Forms

- The complement of a function expressed as the sum of minterms equals the sum 0, which are missing from the original function

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

- if we take the complement of *F* by DeMorgan's theorem

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

- it is clear that the following relation holds

$$m_j' = M_j$$

maxterm with subscript *j* is a complement of the minterm with the same subscript *j* and vice versa

general conversion procedure:

- To convert from one canonical form to another, interchange the symbols $\sum$ and $\prod$ and list those numbers missing from the original form.
- Considering that the total number of minterms or maxterms is 2^n,

Example 03

- Express the Boolean function as a sum of minterms. And product of maxterms

$$F = xy + x'z$$

- $F = xy + x'z$

Truth Table for F = xy + x'z

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Minterms

Maxterms

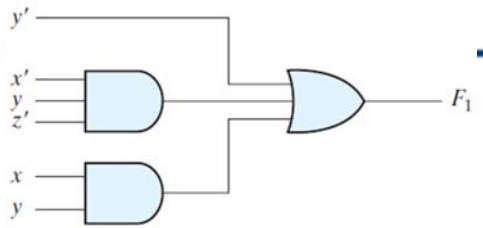
$F(x, y, z) = \Sigma(1, 3, 6, 7)$

$F(x, y, z) = \Pi(0, 2, 4, 5)$

c. Standard Forms

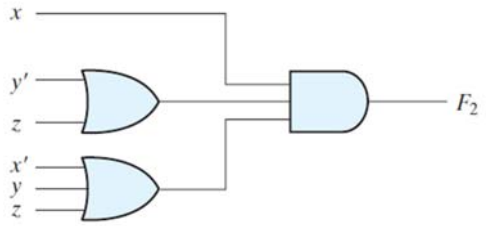
- function may contain one, two, or any number of literals
- Form sum of products or products of sums.
- logic diagram of a sum-of-products expression consists of a group of AND gates followed by a single OR gate.
- logic diagram of a product-of-sum expression consists of a group of OR gates followed by a single AND gate.

- $F_1 = y' + xy + x'yz'$



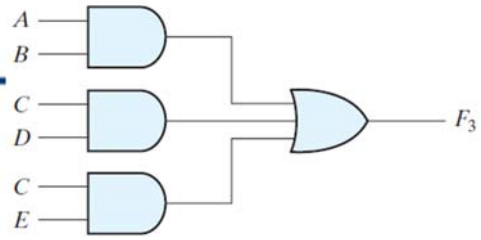
(a) Sum of Products

$F2 = (x)(y' + z)(x' + y + z)$



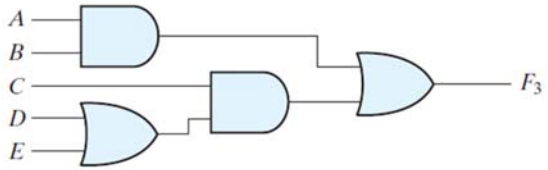
(b) Product of Sums

- (b) $AB + CD + CE$



Standard sum-of-product

(a) $AB + C(D + E)$



Nonstandard form

Other logic Operations

- there are 2^{2n} functions for n binary variables
- for two variables, $n = 2$, and the number of possible Boolean functions is 16.
- the AND and OR functions are only 2 of a total of 16 possible functions formed with two binary variables
- to find the other 14 functions, The truth tables for the 16 functions formed with two binary variables, each colulm contains one possible function for the two variables, *x* and *y.*

Truth Tables for the 16 Functions of Two Binary Variables

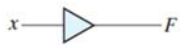
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

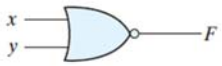
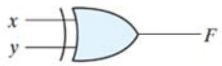
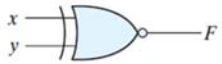
- The 16 functions listed can be subdivided into three categories:
- **1.** Two functions that produce a constant 0 or 1.
- **2.** Four functions with unary operations: complement and transfer.
- **3.** Ten functions with binary operators that define eight different operations: AND, OR, NAND, NOR, exclusive-OR, equivalence, inhibition, and implication.

Boolean Functions	Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

Digital logic gates

-

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1

Name	Graphic symbol	Algebraic function	Truth table
NAND		$F = (xy)'$	$x\ y\ \vert\ F$ $0\ 0\ \vert\ 1$ $0\ 1\ \vert\ 1$ $1\ 0\ \vert\ 1$ $1\ 1\ \vert\ 0$
NOR		$F = (x + y)'$	$x\ y\ \vert\ F$ $0\ 0\ \vert\ 1$ $0\ 1\ \vert\ 0$ $1\ 0\ \vert\ 0$ $1\ 1\ \vert\ 0$
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y\ \vert\ F$ $0\ 0\ \vert\ 0$ $0\ 1\ \vert\ 1$ $1\ 0\ \vert\ 1$ $1\ 1\ \vert\ 0$
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y\ \vert\ F$ $0\ 0\ \vert\ 1$ $0\ 1\ \vert\ 0$ $1\ 0\ \vert\ 0$ $1\ 1\ \vert\ 1$

- All gates except for the inverter and buffer can be extended to have more than two inputs.

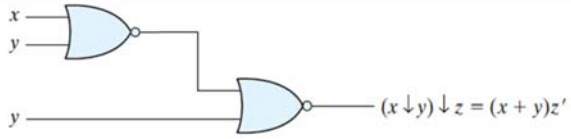
$$x + y = y + x \quad \text{(commutative)}$$

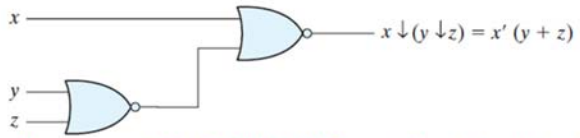
$$(x + y) + z = x + (y + z) = x + y + z \quad \text{(associative)}$$

- The NAND and NOR functions are commutative but not associative

$$(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)),$$

$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$
$$x \downarrow (y \downarrow z) = [x + (y + z)']' = x'(y + z) = x'y + x'z$$

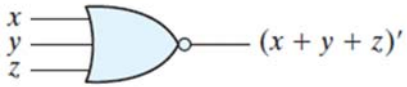




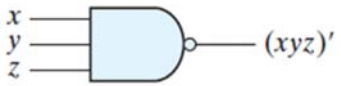
Demonstrating the nonassociativity of the NOR operator: $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

- he multiple NOR (or NAND) gate as a complemented OR (or AND) gate

$$x \downarrow y \downarrow z = (x + y + z)'$$
$$x \uparrow y \uparrow z = (xyz)'$$



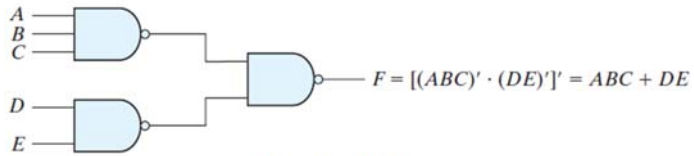
(a) 3-input NOR gate



(b) 3-input NAND gate

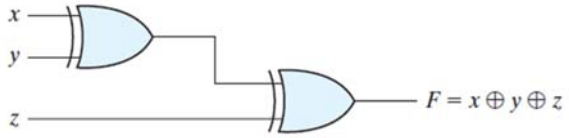
$$F = [(ABC)'(DE)']' = ABC + DE$$

- expression is obtained from one of DeMorgan's theorems.
- It also shows that an expression in sum-of-products form can be implemented with NAND gates.

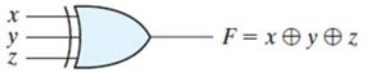


(c) Cascaded NAND gates

- exclusive-OR and equivalence gates are both commutative and associative and can be extended to more than two inputs.
- is normally implemented by cascading two-input gates,



(a) Using 2-input gates



(b) 3-input gate

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

- **INTEGRATED CIRCUITS**
- An integrated circuit (IC) is fabricated on a die of a silicon semiconductor crystal, called a *chip,* containing the electronic components for constructing digital gates.
- Digital ICs are often categorized according to the complexity of their circuits, according to the number of logic gates in a single package.
- ***Small-scale integration*** **(SSI)** devices contain several independent gates in a single package, The number of gates is usually fewer than 10 and is limited by the number of pins available in the IC.

- ***Medium-scale integration*** **(MSI)** devices have a complexity of approximately 10 to 1,000 gates in a single package.
- Like decoders, adders, multiplexers, registers and counters
- ***Large-scale integration*** **(LSI)** devices contain thousands of gates in a single package.
- such as processors, memory chips, and programmable logic devices.
- ***Very large-scale integration*** **(VLSI)** devices now contain millions of gates within a single package
- Like large memory arrays and complex microcomputer

- Digital integrated circuits are classified by the specific circuit technology to which they belong

TTL	transistor–transistor logic;
ECL	emitter-coupled logic;
MOS	metal-oxide semiconductor;
CMOS	complementary metal-oxide semiconductor.

- **TTL** is a logic family that has been in use for many years and is considered to be standard.
- **ECL** has an advantage in systems requiring high-speed operation.
- **MOS** is suitable for circuits that need high component density

- **CMOS** is preferable in systems requiring low power consumption, such as digital cameras, personal media players, and other handheld portable devices, are essential for VLSI design

IC specs:

- *Fan-out* specifies the number of standard loads that the output of a typical gate
- *Fan-in* is the number of inputs available in a gate.
- *Power dissipation* is the power consumed by the gate that must be available from the power supply.
- *Propagation delay* is the average transition delay time for a signal to propagate from input to output.

VLSI:

- The design of digital systems with VLSI circuits containing millions of transistors and gates is an enormous and difficult task.
- Task is impossible to develop and verify without the assistance of computer-aided design (**CAD**) tools,
- Electronic design automation (**EDA**) covers all phases of the design of integrated circuits.
- CAD systems include an editing program for creating and modifying schematic diagrams called *schematic capture*

Custom made ICs

- There are a variety of options available for creating the physical realization of a digital circuit in silicon.
- The designer can choose between an application-specific integrated circuit
- (**ASIC**), a field-programmable gate array (**FPGA**), a programmable logic device (**PLD**), and a full-custom IC.
- Hardware description language (**HDL**), a computer programming language, but is specifically oriented to describing digital hardware