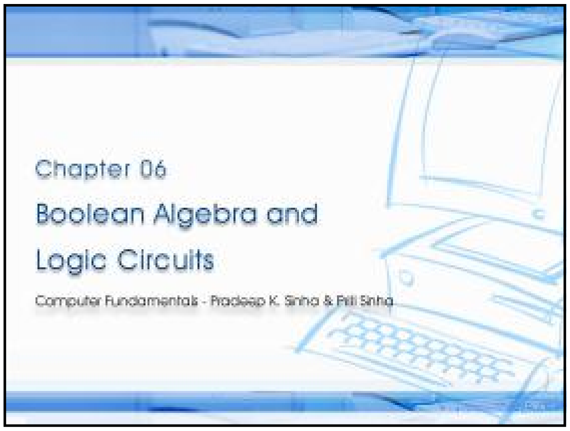
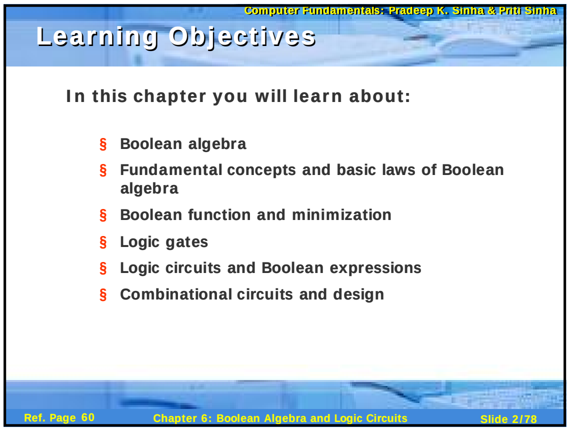


Chapter 06

Boolean Algebra and Logic Circuits

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha



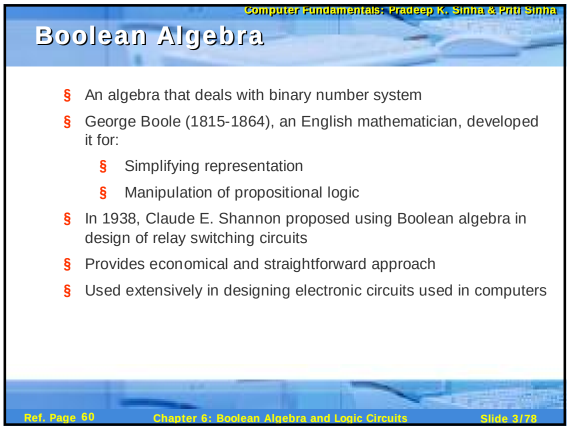
Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Learning Objectives

In this chapter you will learn about:

- **Boolean algebra**
- **Fundamental concepts and basic laws of Boolean algebra**
- **Boolean function and minimization**
- **Logic gates**
- **Logic circuits and Boolean expressions**
- **Combinational circuits and design**

Ref. Page 60 Chapter 6: Boolean Algebra and Logic Circuits Slide 2/78

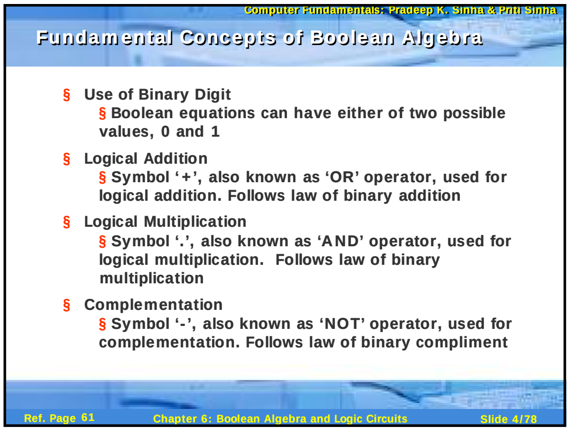


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Boolean Algebra

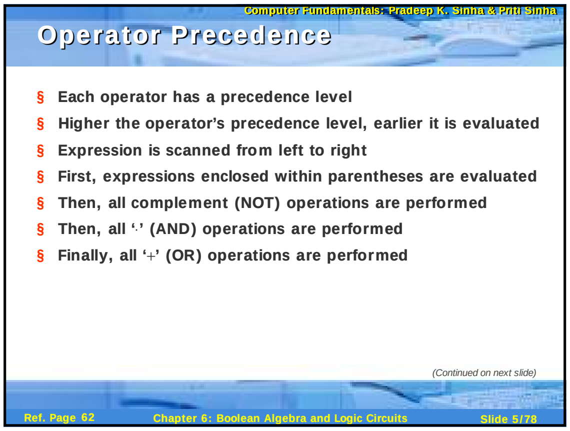
- An algebra that deals with binary number system
- George Boole (1815-1864), an English mathematician, developed it for:
  - Simplifying representation
  - Manipulation of propositional logic
- In 1938, Claude E. Shannon proposed using Boolean algebra in design of relay switching circuits
- Provides economical and straightforward approach
- Used extensively in designing electronic circuits used in computers

Ref. Page 60 Chapter 6: Boolean Algebra and Logic Circuits Slide 3/78



Fundamental Concepts of Boolean Algebra

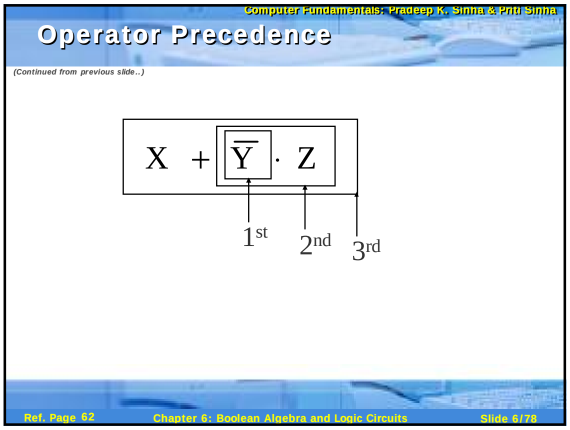
- Use of Binary Digit
  - Boolean equations can have either of two possible values, 0 and 1
- Logical Addition
  - Symbol '+', also known as 'OR' operator, used for logical addition. Follows law of binary addition
- Logical Multiplication
  - Symbol '.', also known as 'AND' operator, used for logical multiplication. Follows law of binary multiplication
- Complementation
  - Symbol '-', also known as 'NOT' operator, used for complementation. Follows law of binary compliment



Operator Precedence

- Each operator has a precedence level
- Higher the operator's precedence level, earlier it is evaluated
- Expression is scanned from left to right
- First, expressions enclosed within parentheses are evaluated
- Then, all complement (NOT) operations are performed
- Then, all '·' (AND) operations are performed
- Finally, all '+' (OR) operations are performed

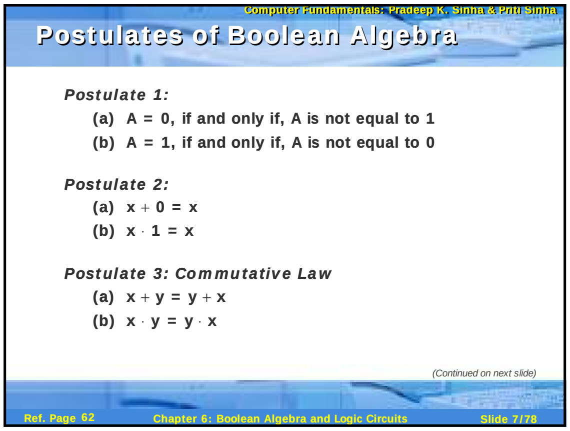
(Continued on next slide)



Operator Precedence

(Continued from previous slide..)

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



Postulates of Boolean Algebra

Postulate 1:

(a) A = 0, if and only if, A is not equal to 1

(b) A = 1, if and only if, A is not equal to 0

Postulate 2:

(a) $x + 0 = x$

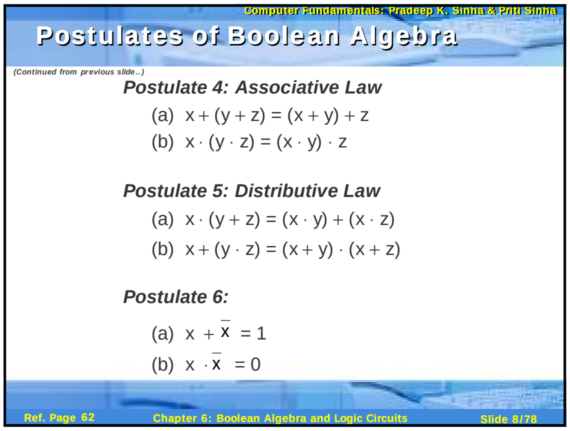
(b) $x \cdot 1 = x$

Postulate 3: Commutative Law

(a) $x + y = y + x$

(b) $x \cdot y = y \cdot x$

(Continued on next slide)



Postulates of Boolean Algebra

(Continued from previous slide..)

Postulate 4: Associative Law

(a) $x + (y + z) = (x + y) + z$

(b) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

Postulate 5: Distributive Law

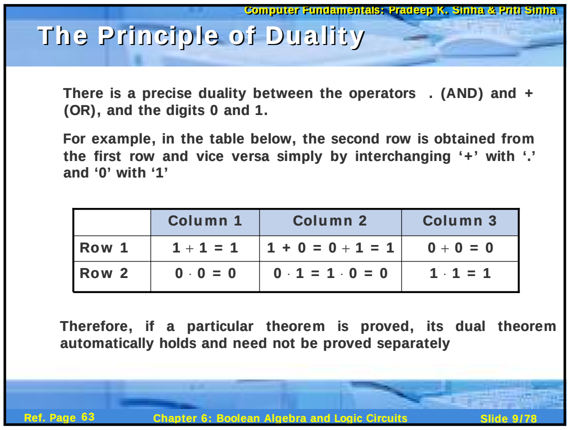
(a) $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

(b) $x + (y \cdot z) = (x + y) \cdot (x + z)$

Postulate 6:

(a) $x + \bar{x} = 1$

(b) $x \cdot \bar{x} = 0$



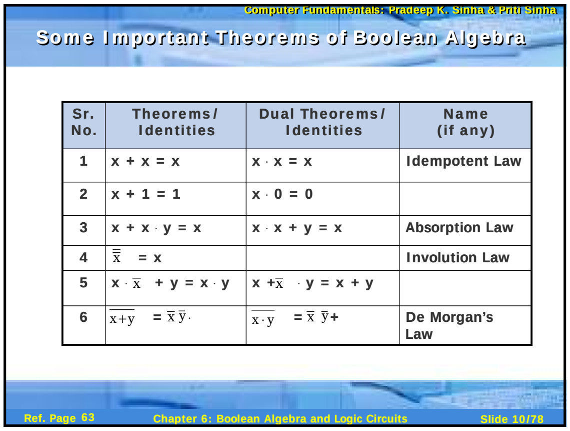
The Principle of Duality

There is a precise duality between the operators . (AND) and + (OR), and the digits 0 and 1.

For example, in the table below, the second row is obtained from the first row and vice versa simply by interchanging '+' with '.' and '0' with '1'

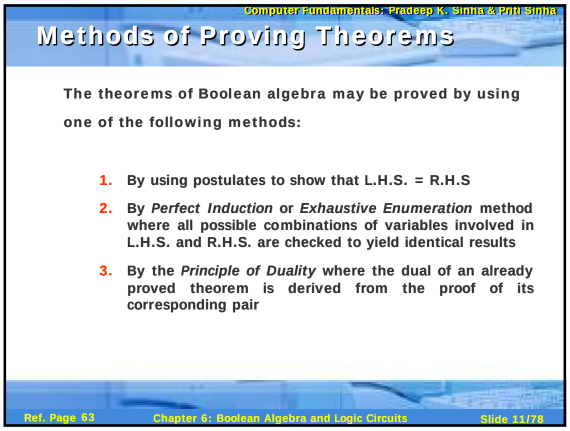
	Column 1	Column 2	Column 3
Row 1	$1 + 1 = 1$	$1 + 0 = 0 + 1 = 1$	$0 + 0 = 0$
Row 2	$0 \cdot 0 = 0$	$0 \cdot 1 = 1 \cdot 0 = 0$	$1 \cdot 1 = 1$

Therefore, if a particular theorem is proved, its dual theorem automatically holds and need not be proved separately



Some Important Theorems of Boolean Algebra

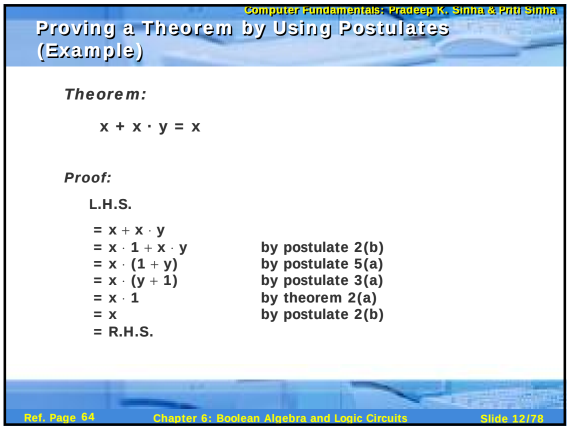
Sr. No.	Theorems/ Identities	Dual Theorems/ Identities	Name (if any)
1	$x + x = x$	$x \cdot x = x$	Idempotent Law
2	$x + 1 = 1$	$x \cdot 0 = 0$	
3	$x + x \cdot y = x$	$x \cdot x + y = x$	Absorption Law
4	$\overline{\overline{x}} = x$		Involution Law
5	$x \cdot \overline{x} + y = x \cdot y$	$x + \overline{x} \cdot y = x + y$	
6	$\overline{x+y} = \overline{x}\,\overline{y} \cdot$	$\overline{x \cdot y} = \overline{x}\,\overline{y} +$	De Morgan's Law



Methods of Proving Theorems

The theorems of Boolean algebra may be proved by using one of the following methods:

1. By using postulates to show that L.H.S. = R.H.S
2. By *Perfect Induction* or *Exhaustive Enumeration* method where all possible combinations of variables involved in L.H.S. and R.H.S. are checked to yield identical results
3. By the *Principle of Duality* where the dual of an already proved theorem is derived from the proof of its corresponding pair



Proving a Theorem by Using Postulates (Example)

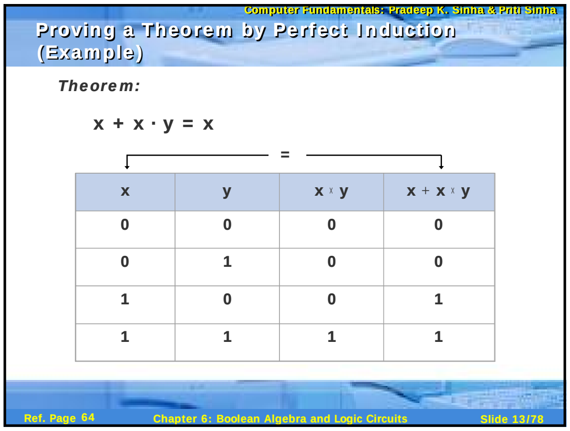
Theorem:

$x + x \cdot y = x$

Proof:

L.H.S.

$= x + x \cdot y$	
$= x \cdot 1 + x \cdot y$	by postulate 2(b)
$= x \cdot (1 + y)$	by postulate 5(a)
$= x \cdot (y + 1)$	by postulate 3(a)
$= x \cdot 1$	by theorem 2(a)
$= x$	by postulate 2(b)
= R.H.S.	



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Proving a Theorem by Perfect Induction (Example)

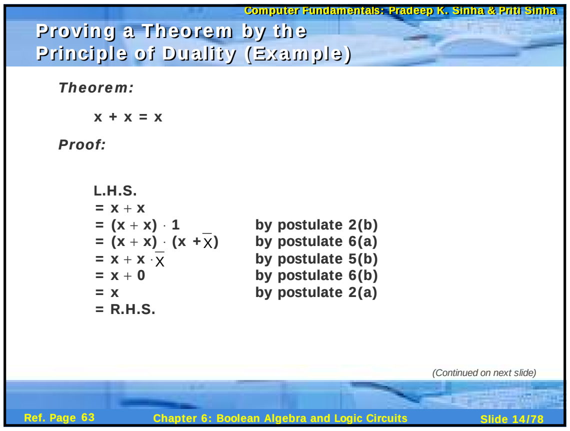
Theorem:

$$x + x \cdot y = x$$

=

x	y	$x \cdot y$	$x + x \cdot y$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Ref. Page 64 Chapter 6: Boolean Algebra and Logic Circuits Slide 13/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Proving a Theorem by the Principle of Duality (Example)

Theorem:

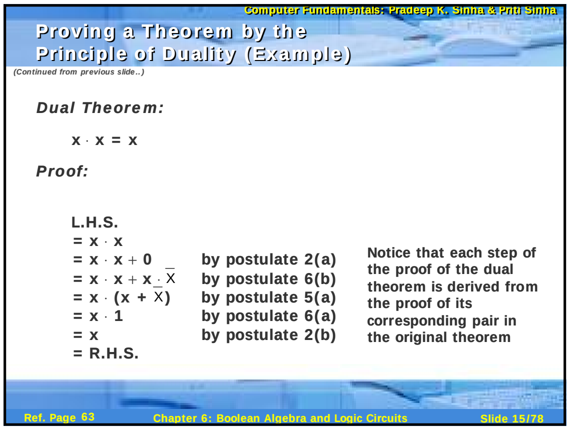
$$x + x = x$$

Proof:

L.H.S.	
$= x + x$	
$= (x + x) \cdot 1$	by postulate 2(b)
$= (x + x) \cdot (x + \overline{x})$	by postulate 6(a)
$= x + x \cdot \overline{x}$	by postulate 5(b)
$= x + 0$	by postulate 6(b)
$= x$	by postulate 2(a)
= R.H.S.	

(Continued on next slide)

Ref. Page 63 Chapter 6: Boolean Algebra and Logic Circuits Slide 14/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Proving a Theorem by the Principle of Duality (Example)

(Continued from previous slide..)

Dual Theorem:

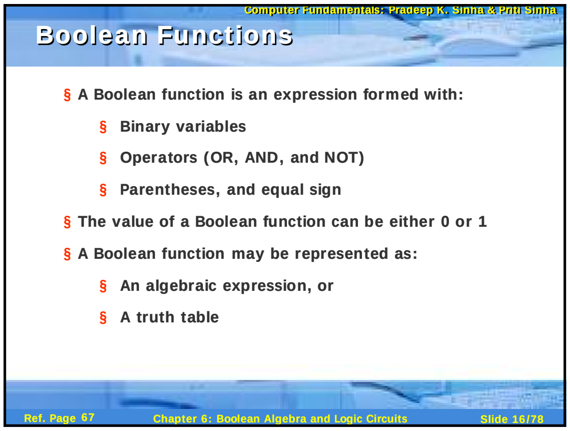
$$x \cdot x = x$$

Proof:

L.H.S.	
$= x \cdot x$	
$= x \cdot x + 0$	by postulate 2(a)
$= x \cdot x + x \cdot \overline{x}$	by postulate 6(b)
$= x \cdot (x + \overline{x})$	by postulate 5(a)
$= x \cdot 1$	by postulate 6(a)
$= x$	by postulate 2(b)
= R.H.S.	

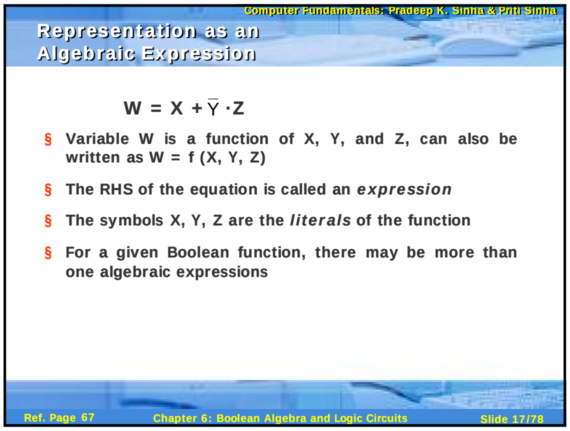
Notice that each step of the proof of the dual theorem is derived from the proof of its corresponding pair in the original theorem

Ref. Page 63 Chapter 6: Boolean Algebra and Logic Circuits Slide 15/78



Boolean Functions

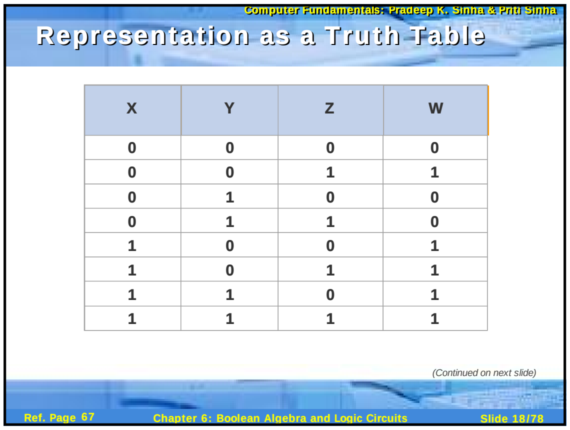
- A Boolean function is an expression formed with:
  - Binary variables
  - Operators (OR, AND, and NOT)
  - Parentheses, and equal sign
- The value of a Boolean function can be either 0 or 1
- A Boolean function may be represented as:
  - An algebraic expression, or
  - A truth table



Representation as an Algebraic Expression

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

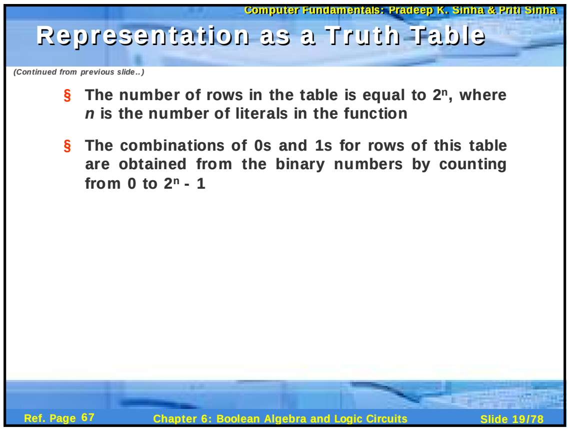
- Variable W is a function of X, Y, and Z, can also be written as W = f (X, Y, Z)
- The RHS of the equation is called an *expression*
- The symbols X, Y, Z are the *literals* of the function
- For a given Boolean function, there may be more than one algebraic expressions



Representation as a Truth Table

X	Y	Z	W
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

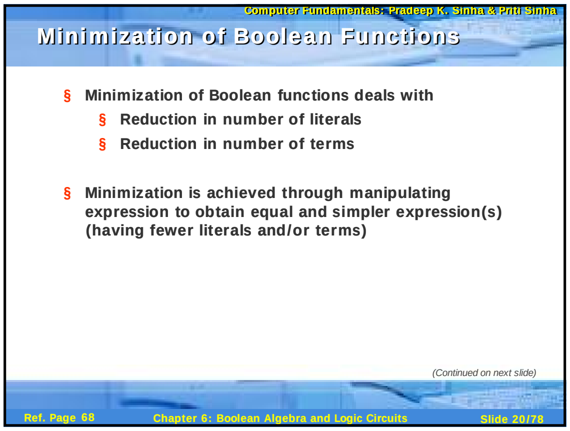
(Continued on next slide)



Representation as a Truth Table

(Continued from previous slide..)

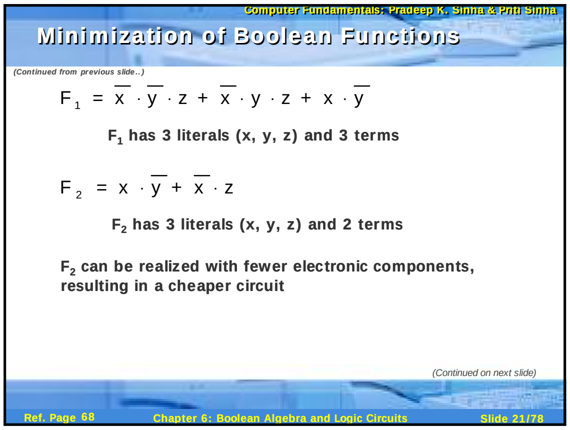
- The number of rows in the table is equal to 2^n, where *n* is the number of literals in the function
- The combinations of 0s and 1s for rows of this table are obtained from the binary numbers by counting from 0 to $2^n - 1$



Minimization of Boolean Functions

- Minimization of Boolean functions deals with
  - Reduction in number of literals
  - Reduction in number of terms
- Minimization is achieved through manipulating expression to obtain equal and simpler expression(s) (having fewer literals and/or terms)

(Continued on next slide)



Minimization of Boolean Functions

(Continued from previous slide..)

$$F_1 = \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot z + x \cdot \bar{y}$$

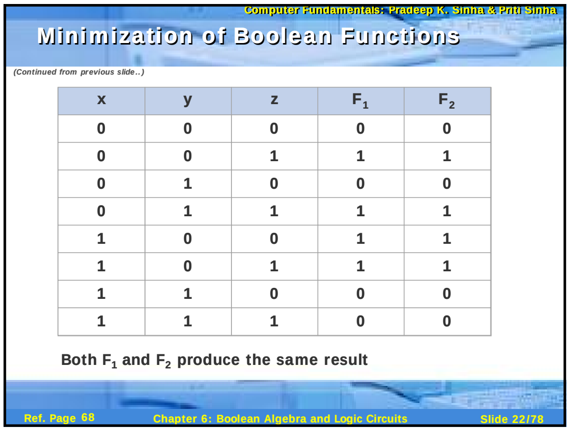
F_1 has 3 literals (x, y, z) and 3 terms

$$F_2 = x \cdot \bar{y} + \bar{x} \cdot z$$

F_2 has 3 literals (x, y, z) and 2 terms

F_2 can be realized with fewer electronic components, resulting in a cheaper circuit

(Continued on next slide)

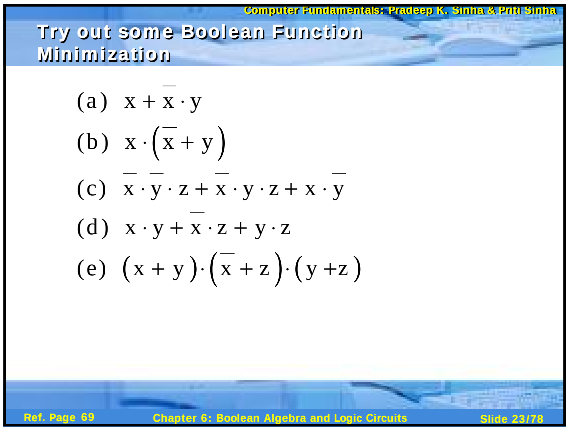


Minimization of Boolean Functions

(Continued from previous slide..)

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Both F_1 and F_2 produce the same result



Try out some Boolean Function Minimization

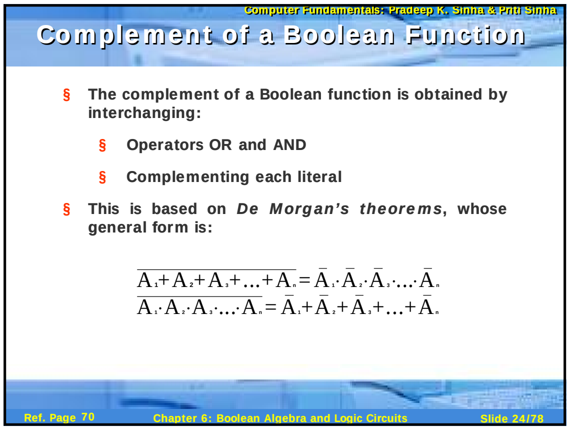
(a) $x + \bar{x} \cdot y$

(b) $x \cdot \left(\bar{x} + y\right)$

(c) $\bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot z + x \cdot \bar{y}$

(d) $x \cdot y + \bar{x} \cdot z + y \cdot z$

(e) $\left(x + y\right) \cdot \left(\bar{x} + z\right) \cdot \left(y + z\right)$

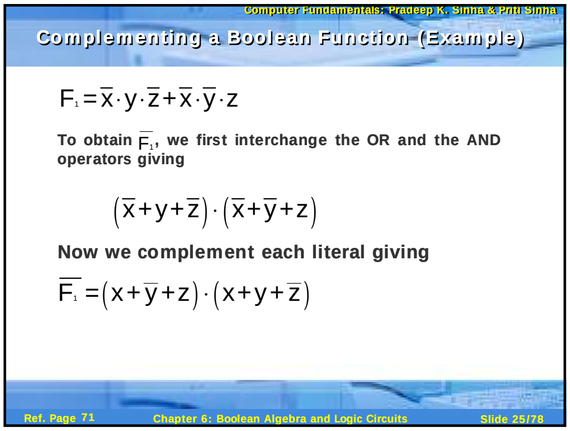


Complement of a Boolean Function

- The complement of a Boolean function is obtained by interchanging:
  - Operators OR and AND
  - Complementing each literal
- This is based on *De Morgan's theorems*, whose general form is:

$$\overline{A_1 + A_2 + A_3 + \ldots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdot \ldots \cdot \bar{A}_n$$

$$\overline{A_1 \cdot A_2 \cdot A_3 \cdot \ldots \cdot A_n} = \bar{A}_1 + \bar{A}_2 + \bar{A}_3 + \ldots + \bar{A}_n$$



Complementing a Boolean Function (Example)

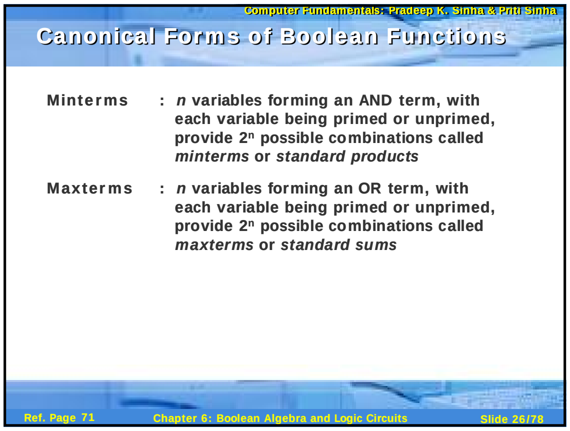
$$F_1 = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z$$

To obtain $\overline{F_1}$, we first interchange the OR and the AND operators giving

$$\left(\bar{x} + y + \bar{z}\right) \cdot \left(\bar{x} + \bar{y} + z\right)$$

Now we complement each literal giving

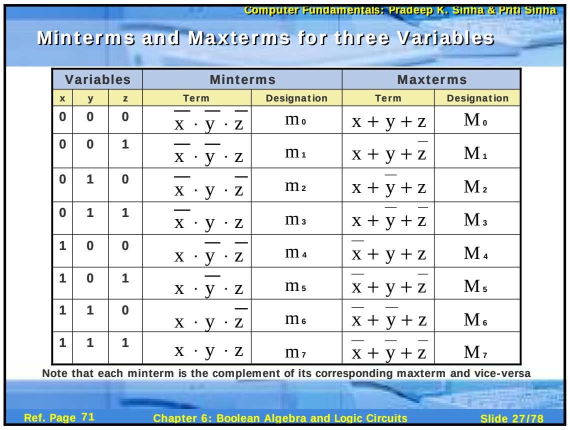
$$\overline{F_1} = \left(x + \bar{y} + z\right) \cdot \left(x + y + \bar{z}\right)$$



Canonical Forms of Boolean Functions

Minterms : *n* variables forming an AND term, with each variable being primed or unprimed, provide 2^n possible combinations called *minterms* or *standard products*

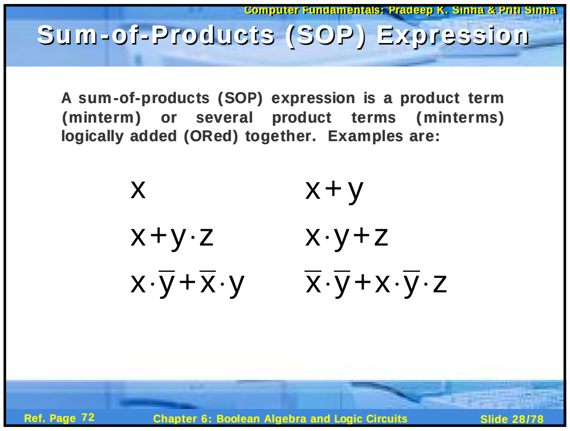
Maxterms : *n* variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations called *maxterms* or *standard sums*



Minterms and Maxterms for three Variables

Variables			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$\bar{x} \cdot \bar{y} \cdot \bar{z}$	m_0	$x + y + z$	M_0
0	0	1	$\bar{x} \cdot \bar{y} \cdot z$	m_1	$x + y + \bar{z}$	M_1
0	1	0	$\bar{x} \cdot y \cdot \bar{z}$	m_2	$x + \bar{y} + z$	M_2
0	1	1	$\bar{x} \cdot y \cdot z$	m_3	$x + \bar{y} + \bar{z}$	M_3
1	0	0	$x \cdot \bar{y} \cdot \bar{z}$	m_4	$\bar{x} + y + z$	M_4
1	0	1	$x \cdot \bar{y} \cdot z$	m_5	$\bar{x} + y + \bar{z}$	M_5
1	1	0	$x \cdot y \cdot \bar{z}$	m_6	$\bar{x} + \bar{y} + z$	M_6
1	1	1	$x \cdot y \cdot z$	m_7	$\bar{x} + \bar{y} + \bar{z}$	M_7

Note that each minterm is the complement of its corresponding maxterm and vice-versa



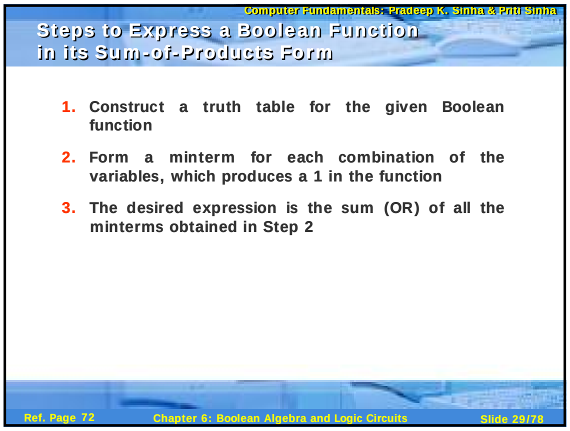
Sum-of-Products (SOP) Expression

A sum-of-products (SOP) expression is a product term (minterm) or several product terms (minterms) logically added (ORed) together. Examples are:

$x \qquad x+y$

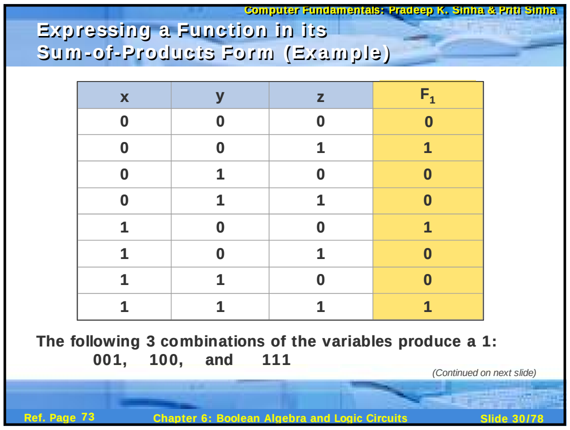
$x+y \cdot z \qquad x \cdot y+z$

$x \cdot \bar{y}+\bar{x} \cdot y \qquad \bar{x} \cdot \bar{y}+x \cdot \bar{y} \cdot z$



Steps to Express a Boolean Function in its Sum-of-Products Form

1. Construct a truth table for the given Boolean function
2. Form a minterm for each combination of the variables, which produces a 1 in the function
3. The desired expression is the sum (OR) of all the minterms obtained in Step 2

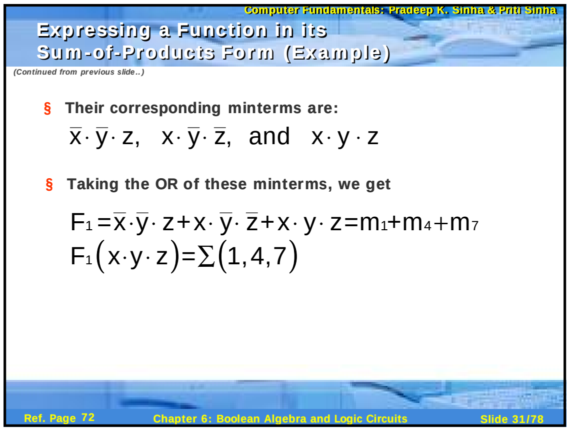


Expressing a Function in its Sum-of-Products Form (Example)

x	y	z	F_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

The following 3 combinations of the variables produce a 1:
001, 100, and 111

(Continued on next slide)



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Expressing a Function in its Sum-of-Products Form (Example)

(Continued from previous slide..)

- Their corresponding minterms are:

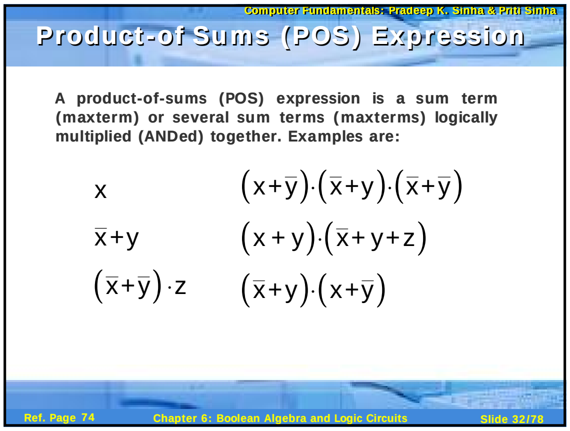
  $\bar{x} \cdot \bar{y} \cdot z, \quad x \cdot \bar{y} \cdot \bar{z}, \quad \text{and} \quad x \cdot y \cdot z$

- Taking the OR of these minterms, we get

$$F_1 = \bar{x} \cdot \bar{y} \cdot z + x \cdot \bar{y} \cdot \bar{z} + x \cdot y \cdot z = m_1 + m_4 + m_7$$

$$F_1\left(x \cdot y \cdot z\right) = \sum\left(1, 4, 7\right)$$

Ref. Page 72 Chapter 6: Boolean Algebra and Logic Circuits Slide 31/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Product-of Sums (POS) Expression

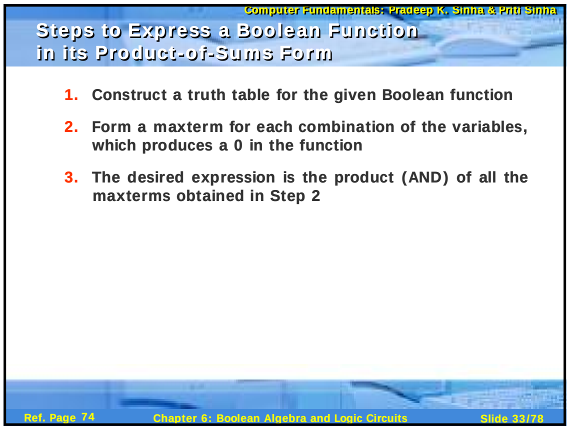
A product-of-sums (POS) expression is a sum term (maxterm) or several sum terms (maxterms) logically multiplied (ANDed) together. Examples are:

$$x \qquad \left(x + \bar{y}\right) \cdot \left(\bar{x} + y\right) \cdot \left(\bar{x} + \bar{y}\right)$$

$$\bar{x} + y \qquad \left(x + y\right) \cdot \left(\bar{x} + y + z\right)$$

$$\left(\bar{x} + \bar{y}\right) \cdot z \qquad \left(\bar{x} + y\right) \cdot \left(x + \bar{y}\right)$$

Ref. Page 74 Chapter 6: Boolean Algebra and Logic Circuits Slide 32/78

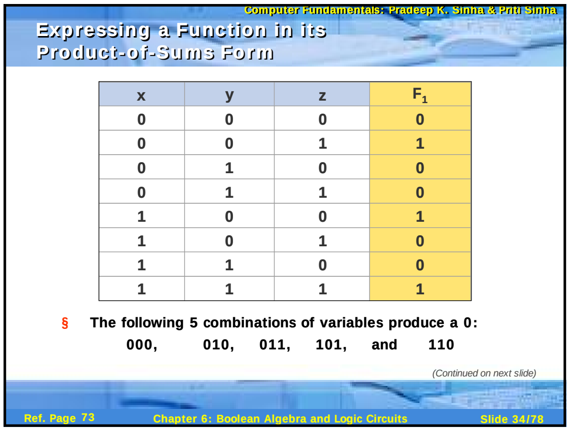


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Steps to Express a Boolean Function in its Product-of-Sums Form

1. Construct a truth table for the given Boolean function
2. Form a maxterm for each combination of the variables, which produces a 0 in the function
3. The desired expression is the product (AND) of all the maxterms obtained in Step 2

Ref. Page 74 Chapter 6: Boolean Algebra and Logic Circuits Slide 33/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Expressing a Function in its Product-of-Sums Form

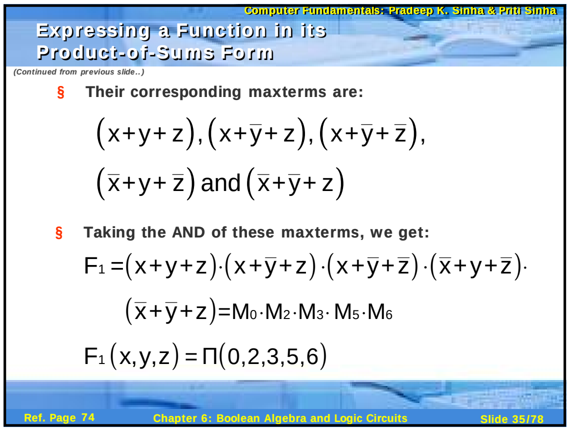
x	y	z	F_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- **The following 5 combinations of variables produce a 0:**

000, 010, 011, 101, and 110

(Continued on next slide)

Ref. Page 73 Chapter 6: Boolean Algebra and Logic Circuits Slide 34/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Expressing a Function in its Product-of-Sums Form

(Continued from previous slide..)

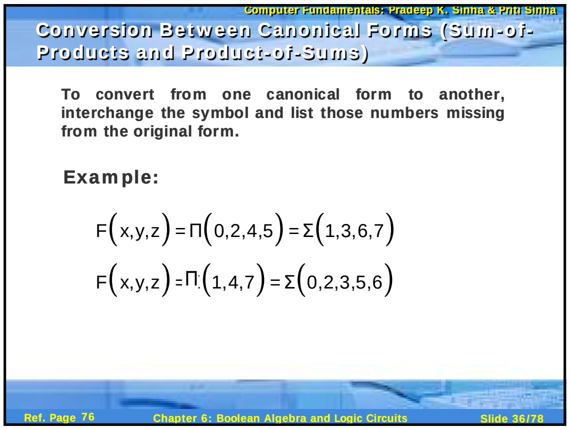
- **Their corresponding maxterms are:**

$$\left(x+y+z\right), \left(x+\bar{y}+z\right), \left(x+\bar{y}+\bar{z}\right),$$
$$\left(\bar{x}+y+\bar{z}\right) \text{ and } \left(\bar{x}+\bar{y}+z\right)$$

- **Taking the AND of these maxterms, we get:**

$$F_1=\left(x+y+z\right)\cdot\left(x+\bar{y}+z\right)\cdot\left(x+\bar{y}+\bar{z}\right)\cdot\left(\bar{x}+y+\bar{z}\right)\cdot$$
$$\left(\bar{x}+\bar{y}+z\right)=M_0\cdot M_2\cdot M_3\cdot M_5\cdot M_6$$
$$F_1\left(x,y,z\right)=\Pi\left(0,2,3,5,6\right)$$

Ref. Page 74 Chapter 6: Boolean Algebra and Logic Circuits Slide 35/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

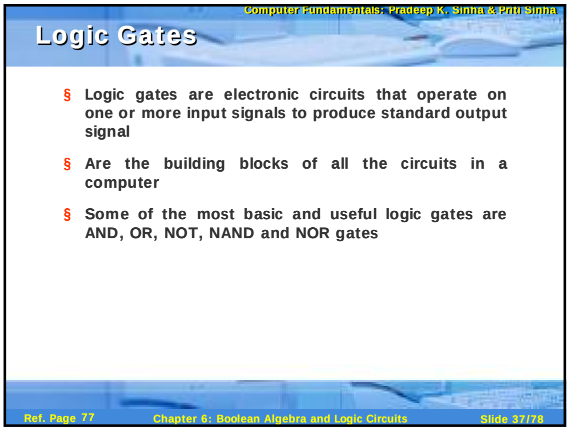
Conversion Between Canonical Forms (Sum-of-Products and Product-of-Sums)

To convert from one canonical form to another, interchange the symbol and list those numbers missing from the original form.

Example:

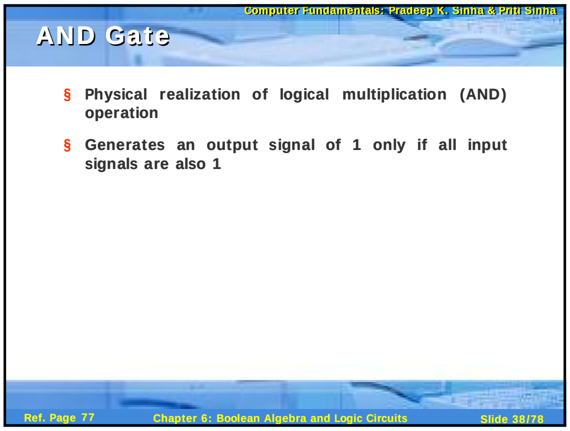
$$F\left(x,y,z\right)=\Pi\left(0,2,4,5\right)=\Sigma\left(1,3,6,7\right)$$
$$F\left(x,y,z\right)=\Pi\left(1,4,7\right)=\Sigma\left(0,2,3,5,6\right)$$

Ref. Page 76 Chapter 6: Boolean Algebra and Logic Circuits Slide 36/78



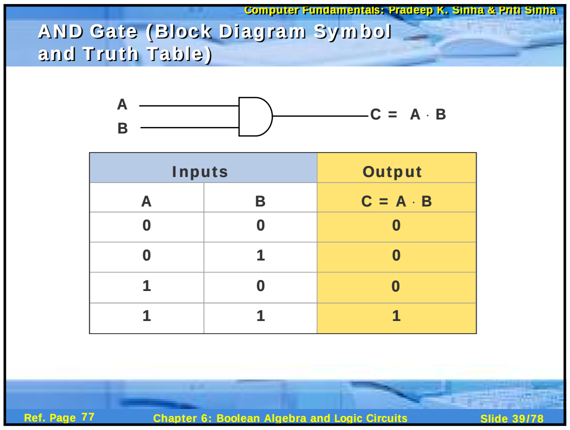
Logic Gates

- Logic gates are electronic circuits that operate on one or more input signals to produce standard output signal
- Are the building blocks of all the circuits in a computer
- Some of the most basic and useful logic gates are AND, OR, NOT, NAND and NOR gates



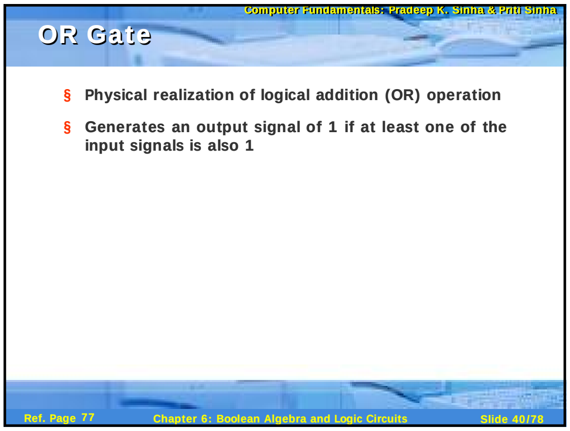
AND Gate

- Physical realization of logical multiplication (AND) operation
- Generates an output signal of 1 only if all input signals are also 1



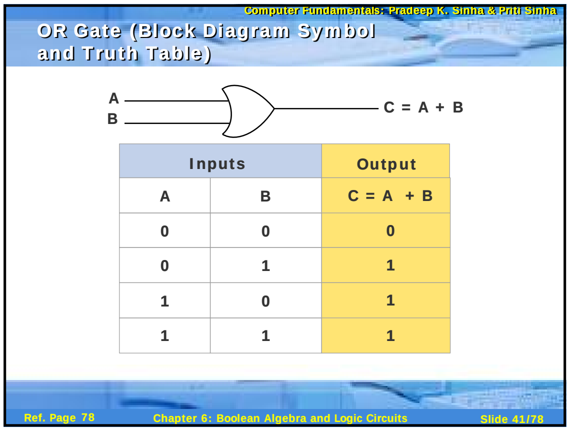
AND Gate (Block Diagram Symbol and Truth Table)

Inputs		Output
A	B	C = A · B
0	0	0
0	1	0
1	0	0
1	1	1



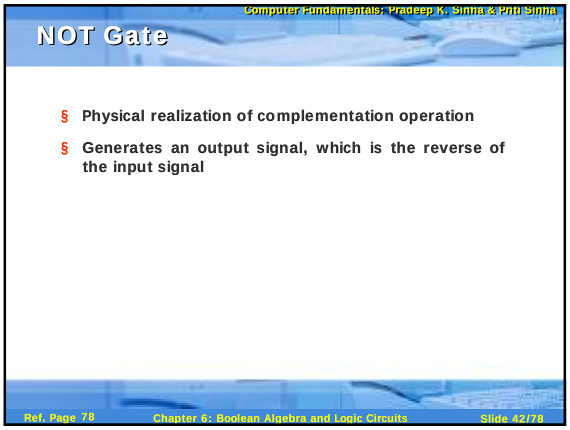
OR Gate

- Physical realization of logical addition (OR) operation
- Generates an output signal of 1 if at least one of the input signals is also 1



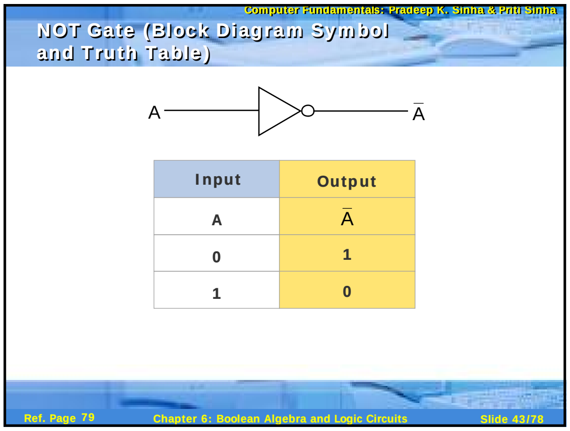
OR Gate (Block Diagram Symbol and Truth Table)

Inputs		Output
A	B	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1



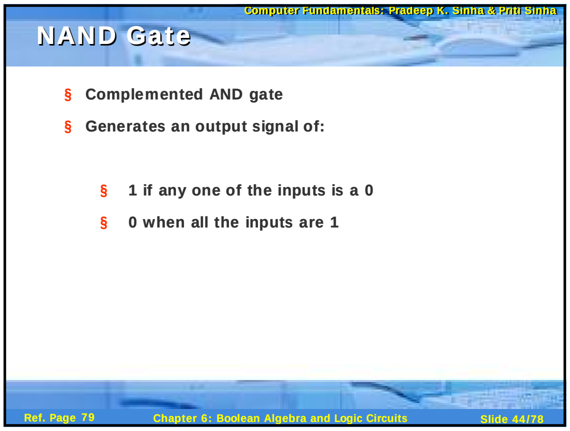
NOT Gate

- Physical realization of complementation operation
- Generates an output signal, which is the reverse of the input signal



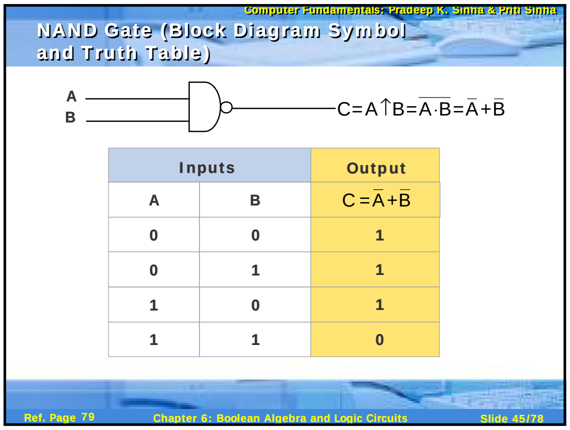
NOT Gate (Block Diagram Symbol and Truth Table)

Input	Output
A	$\bar{A}$
0	1
1	0



NAND Gate

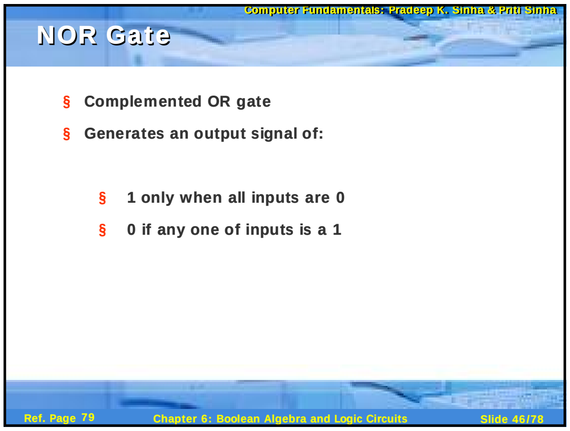
- Complemented AND gate
- Generates an output signal of:
  - 1 if any one of the inputs is a 0
  - 0 when all the inputs are 1



NAND Gate (Block Diagram Symbol and Truth Table)

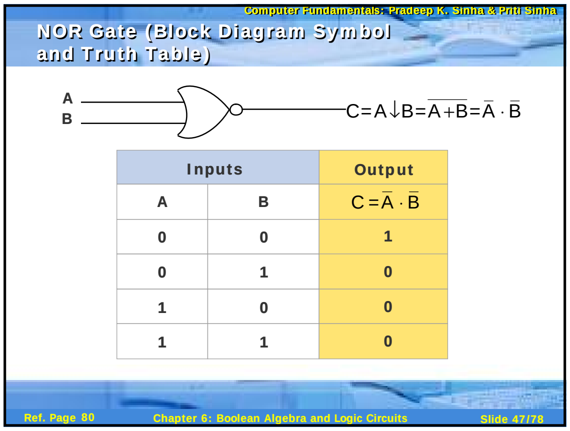
A
B
$C = A \uparrow B = \overline{A \cdot B} = \bar{A} + \bar{B}$

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



NOR Gate

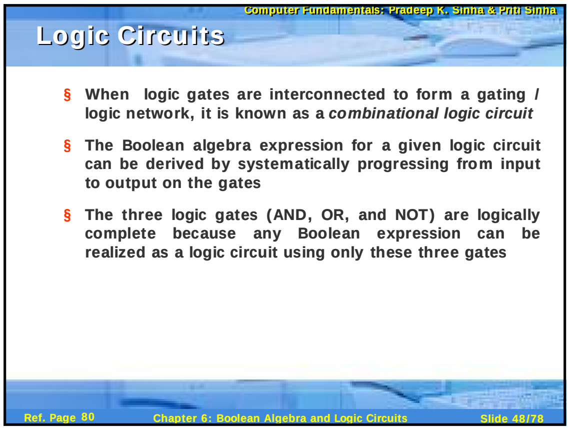
- Complemented OR gate
- Generates an output signal of:
  - 1 only when all inputs are 0
  - 0 if any one of inputs is a 1



NOR Gate (Block Diagram Symbol and Truth Table)

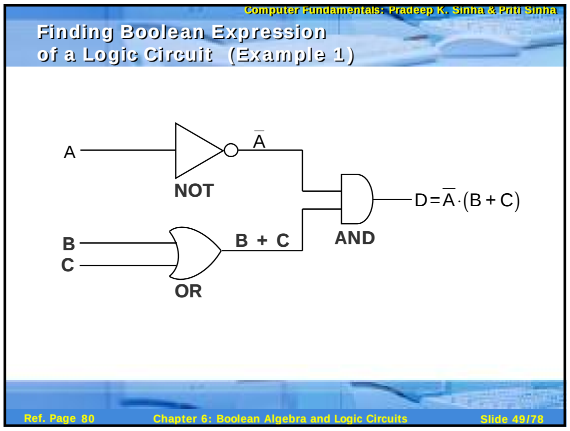
$C = A \downarrow B = \overline{A+B} = \bar{A} \cdot \bar{B}$

Inputs		Output
A	B	$C = \bar{A} \cdot \bar{B}$
0	0	1
0	1	0
1	0	0
1	1	0



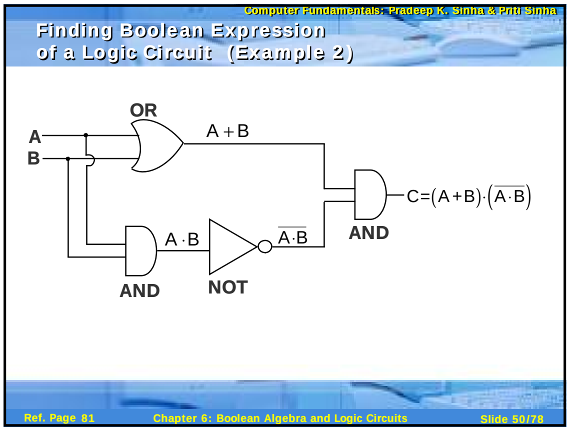
Logic Circuits

- When logic gates are interconnected to form a gating / logic network, it is known as a *combinational logic circuit*
- The Boolean algebra expression for a given logic circuit can be derived by systematically progressing from input to output on the gates
- The three logic gates (AND, OR, and NOT) are logically complete because any Boolean expression can be realized as a logic circuit using only these three gates


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

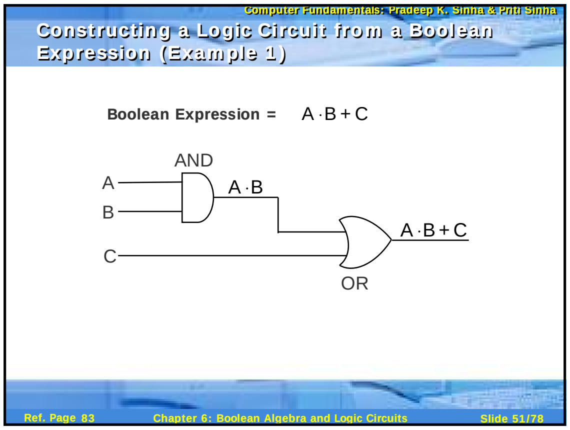
Finding Boolean Expression of a Logic Circuit (Example 1)

Ref. Page 80 Chapter 6: Boolean Algebra and Logic Circuits Slide 49/78


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Finding Boolean Expression of a Logic Circuit (Example 2)

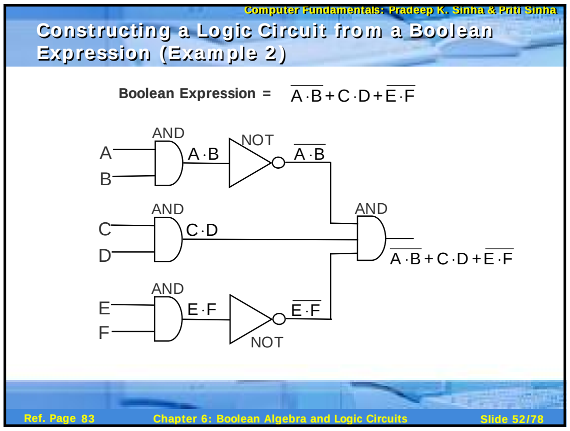
Ref. Page 81 Chapter 6: Boolean Algebra and Logic Circuits Slide 50/78


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Constructing a Logic Circuit from a Boolean Expression (Example 1)

Boolean Expression = $A \cdot B + C$

Ref. Page 83 Chapter 6: Boolean Algebra and Logic Circuits Slide 51/78

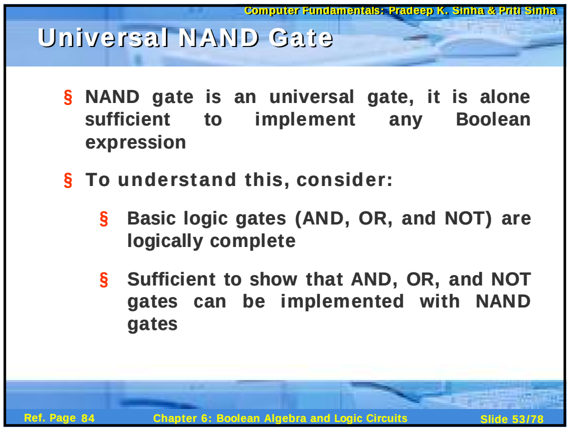


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Constructing a Logic Circuit from a Boolean Expression (Example 2)

Boolean Expression = $\overline{A \cdot B} + C \cdot D + \overline{E \cdot F}$

Ref. Page 83 Chapter 6: Boolean Algebra and Logic Circuits Slide 52/78

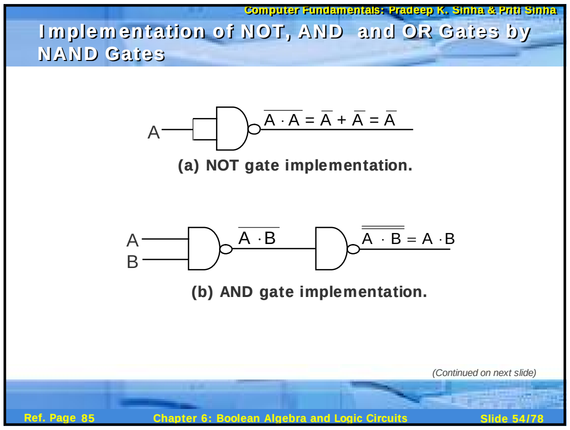


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Universal NAND Gate

- NAND gate is an universal gate, it is alone sufficient to implement any Boolean expression
- To understand this, consider:
  - Basic logic gates (AND, OR, and NOT) are logically complete
  - Sufficient to show that AND, OR, and NOT gates can be implemented with NAND gates

Ref. Page 84 Chapter 6: Boolean Algebra and Logic Circuits Slide 53/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Implementation of NOT, AND and OR Gates by NAND Gates

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

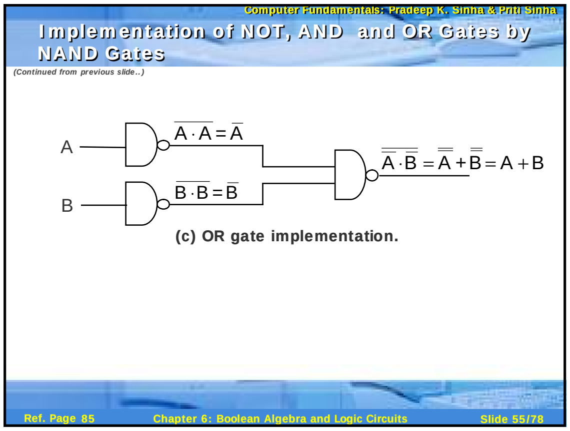
(a) NOT gate implementation.

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

(b) AND gate implementation.

(Continued on next slide)

Ref. Page 85 Chapter 6: Boolean Algebra and Logic Circuits Slide 54/78

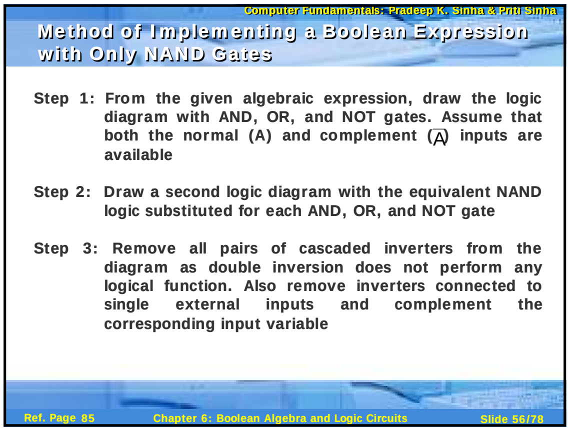


Implementation of NOT, AND and OR Gates by NAND Gates

(Continued from previous slide..)

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

(c) OR gate implementation.

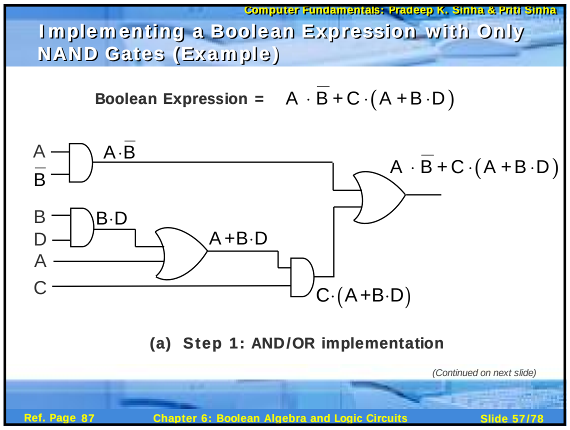


Method of Implementing a Boolean Expression with Only NAND Gates

Step 1: From the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal (A) and complement ($\overline{A}$) inputs are available

Step 2: Draw a second logic diagram with the equivalent NAND logic substituted for each AND, OR, and NOT gate

Step 3: Remove all pairs of cascaded inverters from the diagram as double inversion does not perform any logical function. Also remove inverters connected to single external inputs and complement the corresponding input variable

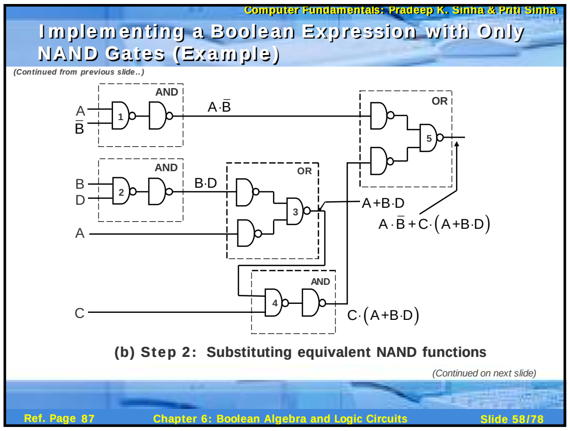


Implementing a Boolean Expression with Only NAND Gates (Example)

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

(a) Step 1: AND/OR implementation

(Continued on next slide)

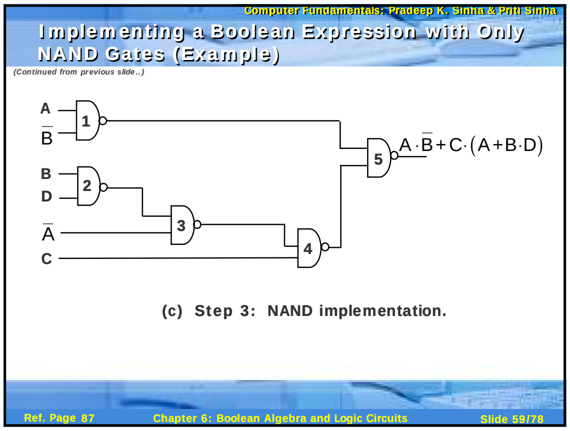


Implementing a Boolean Expression with Only NAND Gates (Example)

(Continued from previous slide..)

(b) Step 2: Substituting equivalent NAND functions

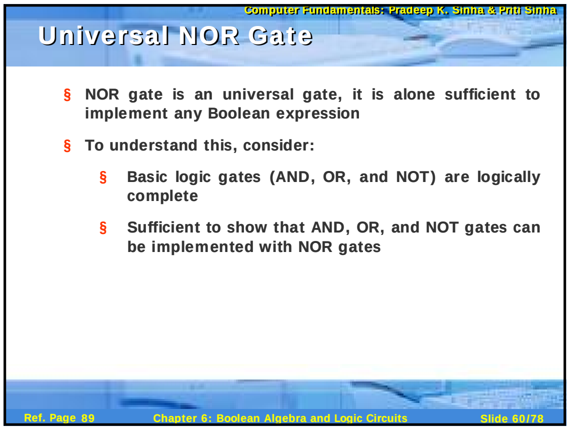
(Continued on next slide)



Implementing a Boolean Expression with Only NAND Gates (Example)

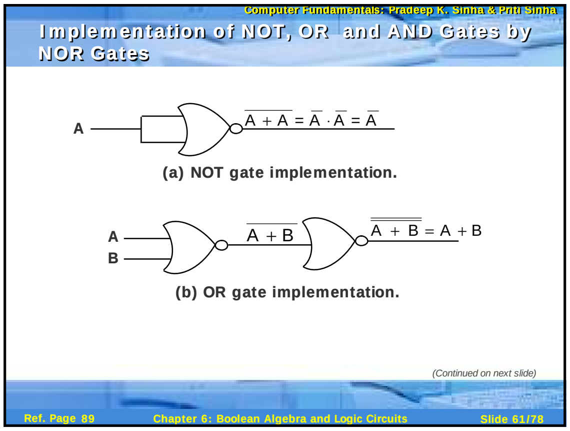
(Continued from previous slide..)

(c) Step 3: NAND implementation.



Universal NOR Gate

- NOR gate is an universal gate, it is alone sufficient to implement any Boolean expression
- To understand this, consider:
  - Basic logic gates (AND, OR, and NOT) are logically complete
  - Sufficient to show that AND, OR, and NOT gates can be implemented with NOR gates



Implementation of NOT, OR and AND Gates by NOR Gates

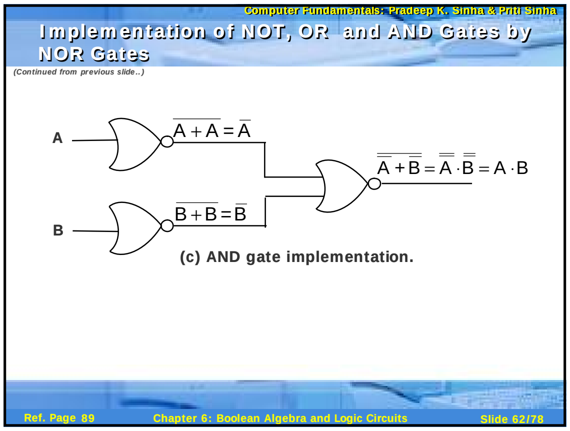
A
$\overline{A+A} = \overline{A}\cdot\overline{A} = \overline{A}$

(a) NOT gate implementation.

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

(b) OR gate implementation.

(Continued on next slide)

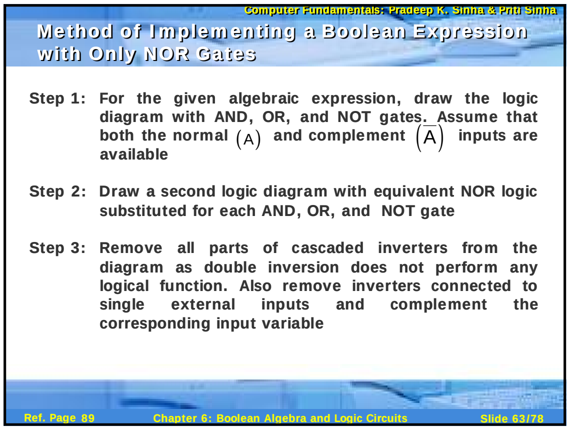


Implementation of NOT, OR and AND Gates by NOR Gates

(Continued from previous slide..)

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

(c) AND gate implementation.



Method of Implementing a Boolean Expression with Only NOR Gates

Step 1: For the given algebraic expression, draw the logic diagram with AND, OR, and NOT gates. Assume that both the normal (A) and complement $(\overline{A})$ inputs are available

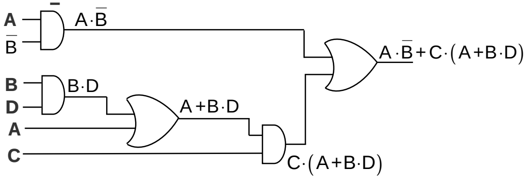
Step 2: Draw a second logic diagram with equivalent NOR logic substituted for each AND, OR, and NOT gate

Step 3: Remove all parts of cascaded inverters from the diagram as double inversion does not perform any logical function. Also remove inverters connected to single external inputs and complement the corresponding input variable

Implementing a Boolean Expression with Only NOR Gates (Examples)

(Continued from previous slide..)

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

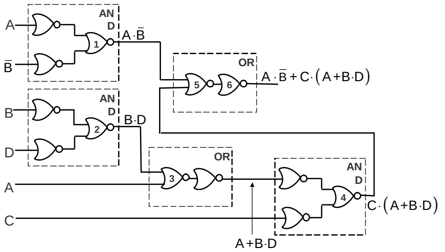


(a) Step 1: AND/OR implementation.

(Continued on next slide)

Implementing a Boolean Expression with Only NOR Gates (Examples)

(Continued from previous slide..)

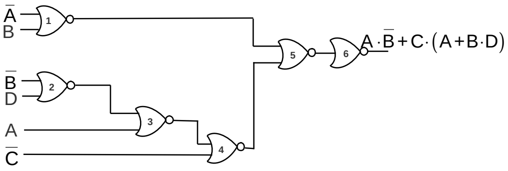


(b) Step 2: Substituting equivalent NOR functions.

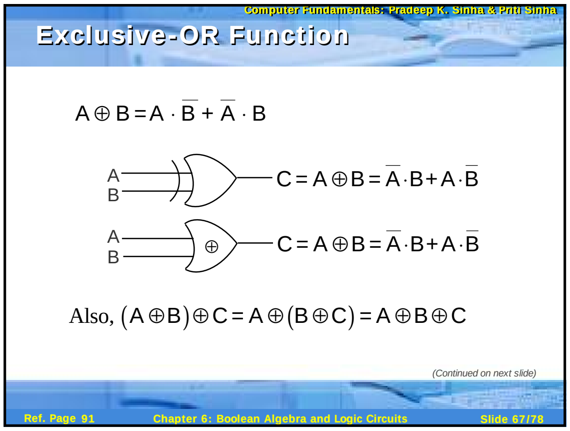
(Continued on next slide)

Implementing a Boolean Expression with Only NOR Gates (Examples)

(Continued from previous slide..)



(c) Step 3: NOR implementation.



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Exclusive-OR Function

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

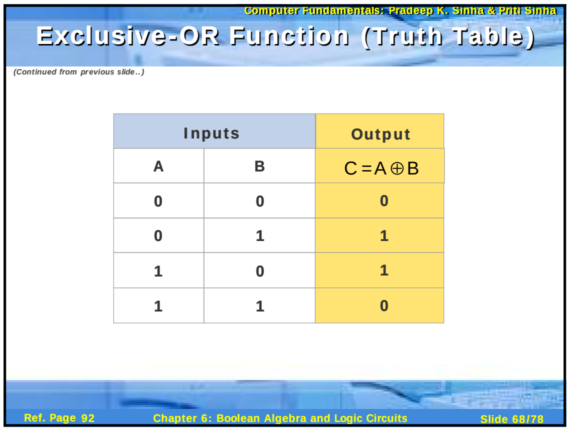
$C = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

$C = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

Also, $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

(Continued on next slide)

Ref. Page 91 Chapter 6: Boolean Algebra and Logic Circuits Slide 67/78



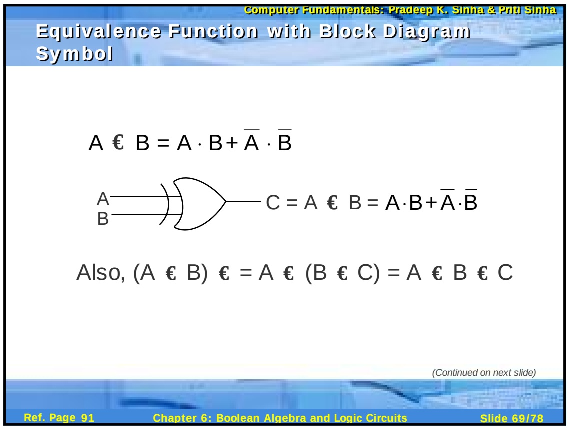
Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Exclusive-OR Function (Truth Table)

(Continued from previous slide..)

Inputs		Output
A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Ref. Page 92 Chapter 6: Boolean Algebra and Logic Circuits Slide 68/78



Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Equivalence Function with Block Diagram Symbol

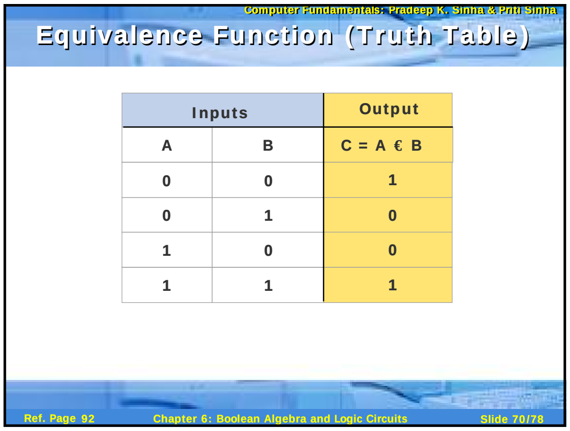
$A \text{ € } B = A \cdot B + \overline{A} \cdot \overline{B}$

$C = A \text{ € } B = A \cdot B + \overline{A} \cdot \overline{B}$

Also, $(A \text{ € } B) \text{ € } = A \text{ € } (B \text{ € } C) = A \text{ € } B \text{ € } C$

(Continued on next slide)

Ref. Page 91 Chapter 6: Boolean Algebra and Logic Circuits Slide 69/78

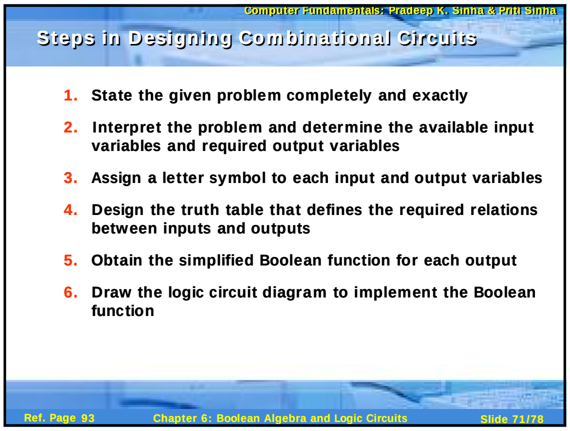


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Equivalence Function (Truth Table)

Inputs		Output
A	B	C = A € B
0	0	1
0	1	0
1	0	0
1	1	1

Ref. Page 92 Chapter 6: Boolean Algebra and Logic Circuits Slide 70/78

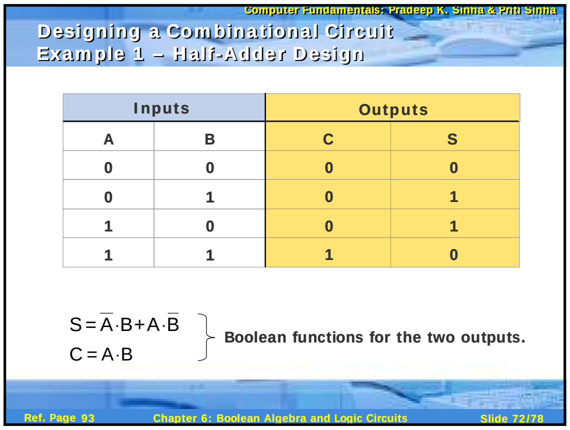


Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Steps in Designing Combinational Circuits

1. State the given problem completely and exactly
2. Interpret the problem and determine the available input variables and required output variables
3. Assign a letter symbol to each input and output variables
4. Design the truth table that defines the required relations between inputs and outputs
5. Obtain the simplified Boolean function for each output
6. Draw the logic circuit diagram to implement the Boolean function

Ref. Page 93 Chapter 6: Boolean Algebra and Logic Circuits Slide 71/78



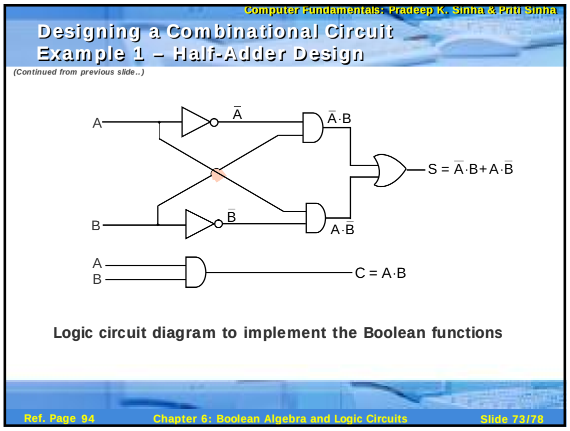
Computer Fundamentals: Pradeep K. Sinha & Priti Sinha

Designing a Combinational Circuit Example 1 – Half-Adder Design

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\left.\begin{array}{l} S = \overline{A}\cdot B + A\cdot\overline{B} \\ C = A\cdot B \end{array}\right\}$$ Boolean functions for the two outputs.

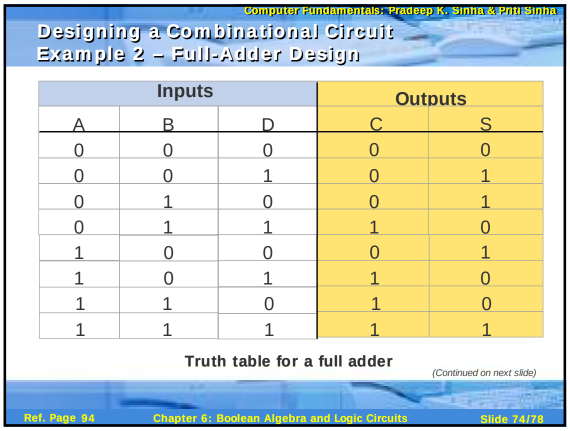
Ref. Page 93 Chapter 6: Boolean Algebra and Logic Circuits Slide 72/78



Designing a Combinational Circuit Example 1 – Half-Adder Design

(Continued from previous slide..)

Logic circuit diagram to implement the Boolean functions

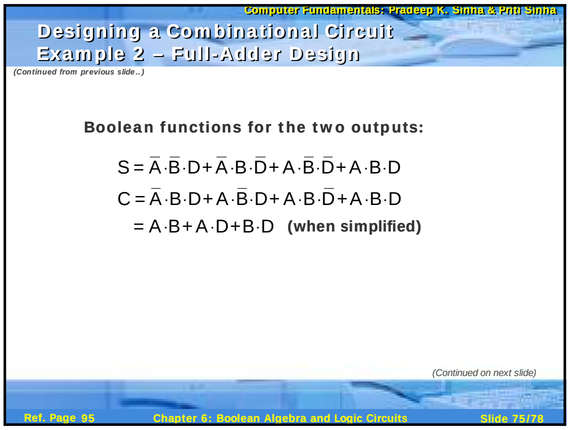


Designing a Combinational Circuit Example 2 – Full-Adder Design

Inputs			Outputs	
A	B	D	C	S
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

Truth table for a full adder

(Continued on next slide)



Designing a Combinational Circuit Example 2 – Full-Adder Design

(Continued from previous slide..)

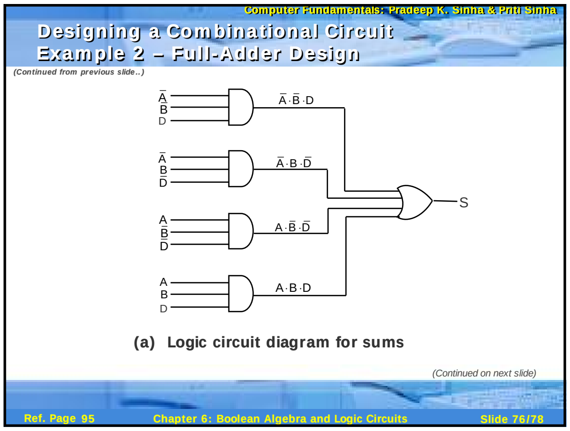
Boolean functions for the two outputs:

$$S = \bar{A}\cdot\bar{B}\cdot D + \bar{A}\cdot B\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{D} + A\cdot B\cdot D$$

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

$$= A\cdot B + A\cdot D + B\cdot D$$ (when simplified)

(Continued on next slide)



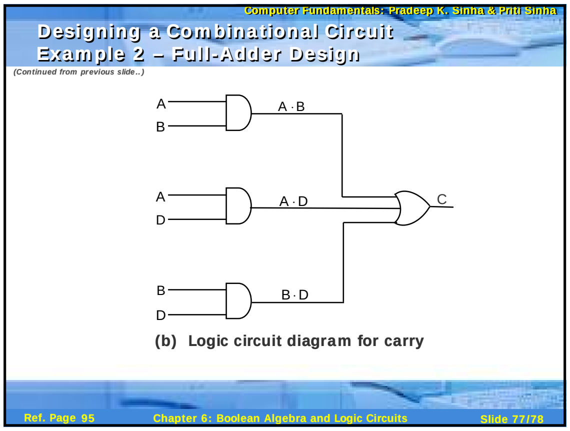
Designing a Combinational Circuit Example 2 – Full-Adder Design

(Continued from previous slide..)

$\bar{A}$
B
D
$\bar{A} \cdot \bar{B} \cdot D$
$\bar{A}$
B
$\bar{D}$
$\bar{A} \cdot B \cdot \bar{D}$
A
$\bar{B}$
$\bar{D}$
$A \cdot \bar{B} \cdot \bar{D}$
A
B
D
$A \cdot B \cdot D$
S

(a) Logic circuit diagram for sums

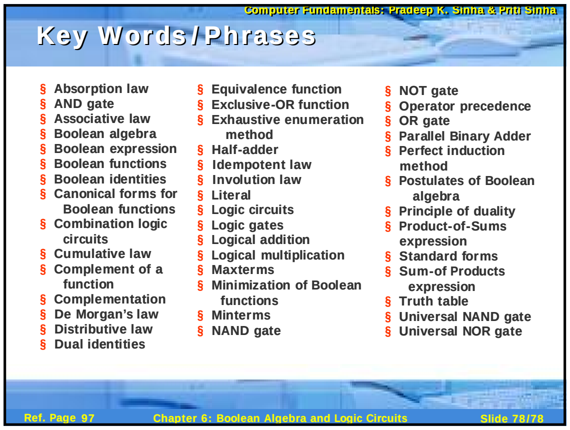
(Continued on next slide)



Designing a Combinational Circuit Example 2 – Full-Adder Design

(Continued from previous slide..)

(b) Logic circuit diagram for carry



Key Words / Phrases

- Absorption law
- AND gate
- Associative law
- Boolean algebra
- Boolean expression
- Boolean functions
- Boolean identities
- Canonical forms for Boolean functions
- Combination logic circuits
- Cumulative law
- Complement of a function
- Complementation
- De Morgan's law
- Distributive law
- Dual identities
- Equivalence function
- Exclusive-OR function
- Exhaustive enumeration method
- Half-adder
- Idempotent law
- Involution law
- Literal
- Logic circuits
- Logic gates
- Logical addition
- Logical multiplication
- Maxterms
- Minimization of Boolean functions
- Minterms
- NAND gate
- NOT gate
- Operator precedence
- OR gate
- Parallel Binary Adder
- Perfect induction method
- Postulates of Boolean algebra
- Principle of duality
- Product-of-Sums expression
- Standard forms
- Sum-of Products expression
- Truth table
- Universal NAND gate
- Universal NOR gate