

BOOLEAN ALGEBRA AND LOGIC CIRCUITS

Introduction to Information and Communication Technologies

Dr. Muhammad Abdullah



Department of Data Science
Faculty of Computing and Information Technology (FCIT)
University of the Punjab, Lahore, Pakistan.

Boolean Algebra

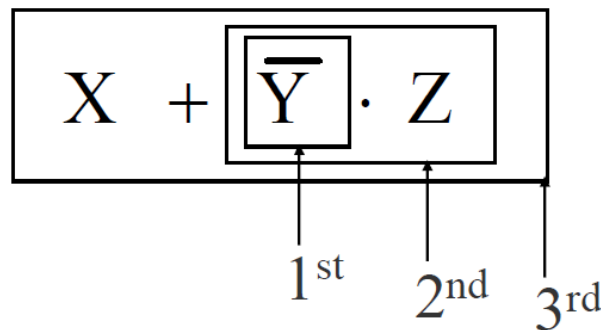
- An algebra that deals with binary number system
- George Boole (1815-1864), an English mathematician, developed it for:
 - Simplifying representation
 - Manipulation of propositional logic
- In 1938, Claude E. Shannon proposed using Boolean algebra in design of relay switching circuits
- Provides economical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

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 complement

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



Properties of Boolean Algebra

PROPERTY	AND	OR
Commutative	$AB = BA$	$A + B = B + A$
Associative	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive	$A(B + C) = (AB) + (AC)$	$A + (BC) = (A + B)(A + C)$
Identity	$A1 = A$	$A + 0 = A$
Complement	$A(A') = 0$	$A + (A') = 1$
De Morgan's law	$(AB)' = A' \text{ OR } B'$	$(A + B)' = A'B'$

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

Boolean Algebraic Expression

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

- 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

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

- 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 $2^n - 1$

Gates and Circuits

Computers and Electricity

- **Gate**
 - A device that performs a basic operation of electrical signals
- **Circuits**
 - Gates combined to perform more complicated tasks

Computers and Electricity

How do we describe the behavior of gates and circuits?

- **Boolean expressions:** Uses Boolean algebra, a mathematical notation for expressing two-valued logic
- **Logic diagrams:** A graphical representation of a circuit; each gate has its own symbol
- **Truth tables:** A table showing all possible input values and the associated output values

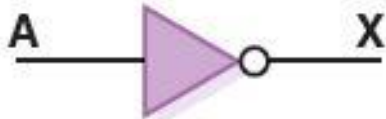
Gates

Six types of gates

- NOT
- AND
- OR
- XOR
- NAND
- NOR

NOT Gate

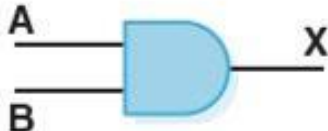
A NOT gate accepts one input signal (0 or 1) and returns the complementary (opposite) signal as output

Boolean Expression	Logic Diagram Symbol	Truth Table						
$X = A'$		<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0
A	X							
0	1							
1	0							

AND Gate

An AND gate accepts two input signals

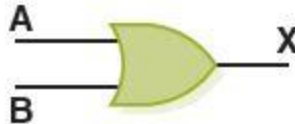
If both are 1, the output is 1; otherwise, the output is 0

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \cdot B$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	0	1	0	0	1	1	1
A	B	X															
0	0	0															
0	1	0															
1	0	0															
1	1	1															

OR Gate

An OR gate accepts two input signals


If both are 0, the output is 0; otherwise, the output is 1

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A + B$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	1
A	B	X															
0	0	0															
0	1	1															
1	0	1															
1	1	1															

XOR Gate

An XOR gate accepts two input signals

If both are the same, the output is 0; otherwise, the output is 1

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \oplus B$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	0
A	B	X															
0	0	0															
0	1	1															
1	0	1															
1	1	0															

XOR Gate

Note the difference between the **XOR** gate and the **OR** gate; they differ only in one input situation

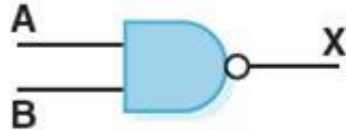
When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR is called the *exclusive OR* because its output is 1 if (and only if):

- *either* one input *or* the other is 1,
- *excluding* the case that they both are

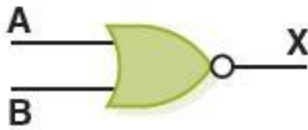
NAND Gate

The NAND (“NOT of AND”) gate accepts two input signals. If both are 1, the output is 0; otherwise, the output is 1.

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = (A \cdot B)'$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	1	1	0	1	1	1	0
A	B	X															
0	0	1															
0	1	1															
1	0	1															
1	1	0															

NOR Gate

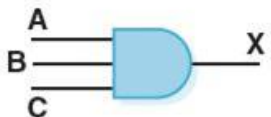
The NOR (“NOT of OR”) gate accepts two inputs. If both are 0, the output is 1; otherwise, the output is 0.

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = (A + B)'$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	0	1	0	0	1	1	0
A	B	X															
0	0	1															
0	1	0															
1	0	0															
1	1	0															

Gates with More Inputs

Some gates can be generalized to accept three or more input values

A three-input **AND** gate, for example, produces an output of **1** only if all input values are **1**

Boolean Expression	Logic Diagram Symbol	Truth Table																																				
$X = A \cdot B \cdot C$		<table><tr><th>A</th><th>B</th><th>C</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	A	B	C	X	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1
A	B	C	X																																			
0	0	0	0																																			
0	0	1	0																																			
0	1	0	0																																			
0	1	1	0																																			
1	0	0	0																																			
1	0	1	0																																			
1	1	0	0																																			
1	1	1	1																																			

Review of Gate Processing

Gate	Behavior
NOT	Inverts its single input
AND	Produces 1 if all input values are 1
OR	Produces 0 if all input values are 0
XOR	Produces 0 if both input values are the same
NAND	Produces 0 if all input values are 1
NOR	Produces 1 if all input values are 0

Constructing Gates

Transistor: A device that acts either as a wire that conducts electricity or as a resistor that blocks the flow of electricity, depending on the voltage level of an input signal

Transistors are used to build logical gates



Circuits

Combinational circuit: The input values explicitly determine the output

Sequential circuit: The output is a function of the input values and the existing state of the circuit

We describe the circuit operations using

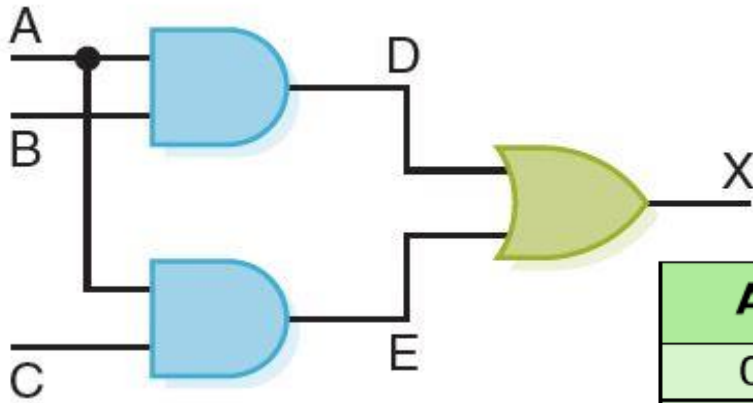
- Boolean expressions

- Logic diagrams

- Truth tables

Combinational Circuits

Gates are combined into circuits by using the output of one gate as the input for another



A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Adders

At the digital logic level, addition is performed in binary

Addition operations are carried out by special circuits called, appropriately, **adders**

Adders

The result of adding two binary digits could produce a *carry value*

Recall that $1 + 1 = 10$
in base two

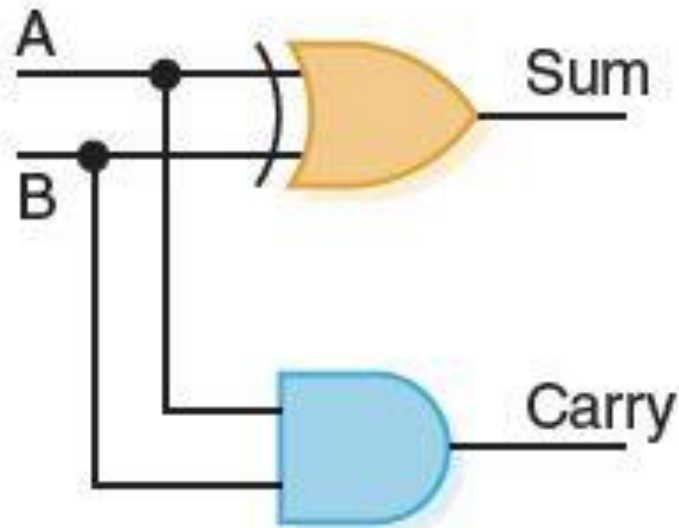
Half adder

A circuit that computes the sum of two bits and produces the correct carry bit

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table

Adders



Circuit diagram
representing
a half adder

Boolean expressions

$$\text{sum} = A \oplus B$$

$$\text{carry} = AB$$

Adders

Full adder

A circuit that takes the carry-in value into account

