COE 202- Digital Logic

Binary Logic and Gates

Dr. Abdulaziz Y. Barnawi
COE Department
KFUPM

Outline

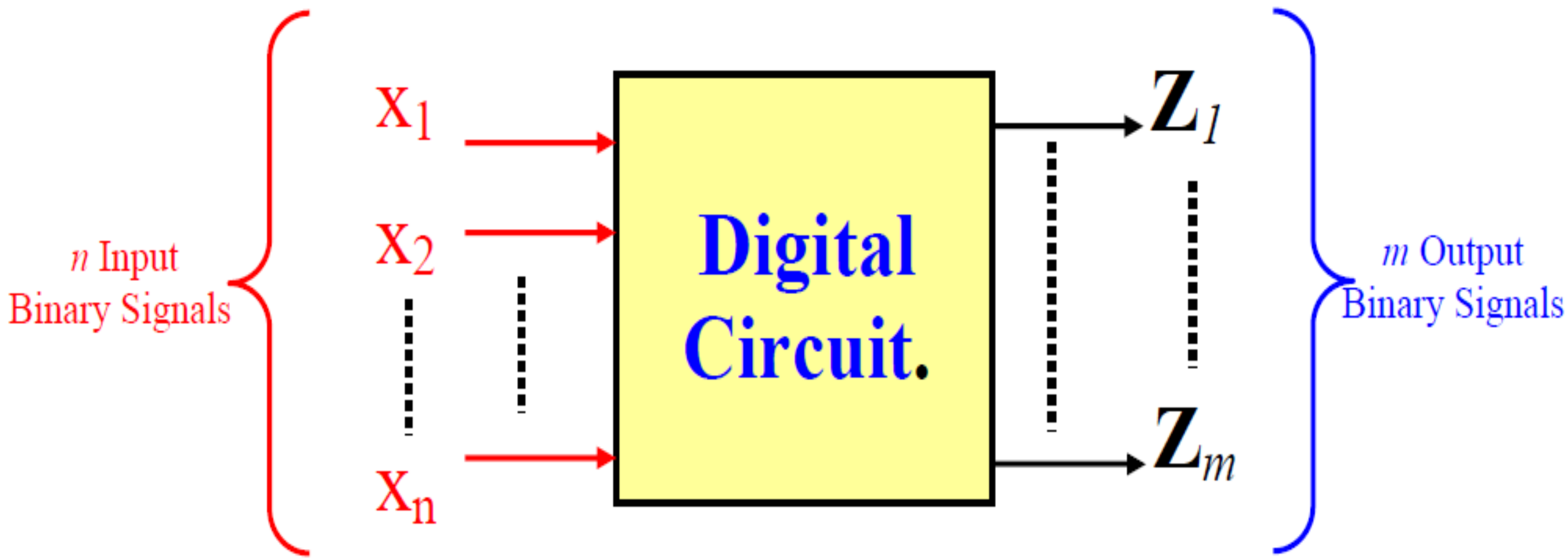
- Introduction
- Boolean Algebra
- Elements of Boolean Algebra (Binary Logic)
- Logic Operations & Logic Gates
- Logic Circuits and Boolean Expressions
- Operator Precedence
- Basic Identities of Boolean Algebra
- Duality Principle
- Algebraic Manipulation

Introduction

- Our objective is to learn how to design ***digital circuits***
- These circuits use ***binary systems***
- Signals in such binary systems may represent only one of 2 possible values 0 *or* 1
- Physically, these signals are electrical voltage signals
- These signals may assume either a High or a Low voltage value
- The High voltage value typically equals the voltage of the power supply (e.g. 5 volts or 3.3 volts), and the Low voltage value is typically 0 volts (or Ground)
- When a signal is at the High voltage value, we say that the signal has a Logic 1 value
- When a signal is at the Low voltage value, we say that the signal has a Logic 0 value

Introduction - Digital Circuits

- Digital circuits process (or manipulate) input binary signals and produce the required output binary signals



Introduction - Digital Circuits

- Generally, the circuit will have a number of input signals (say *n*) as x_1, x_2, ..., x_n, and a number of output signals (say m) Z_1, Z_2, ..., Z_m
- The value assumed by the i^{th} output signal Z_i depends on the values of the input signals x_1, x_2, ..., x_n
- In other words, we can say that Z_i is a function of the n input signals x_1, x_2, ..., x_n. Or we can write:

$$Z_i = F_i(x_1, x_2, \ldots\ldots, x_n) \text{ for } i = 1, 2, 3, \ldots., m$$

- The m output functions (F_i) are functions of binary signals and each produces a single binary output signal
- Thus, these functions are binary functions and require binary logic algebra for their derivation and manipulation

Boolean Algebra

- The binary system algebra is commonly referred to as Boolean Algebra after the mathematician George Boole
- The functions are known as Boolean functions while the binary signals are represented by Boolean variables
- To be able to design a digital circuit, we must learn how to derive the Boolean function implemented by this circuit
- Systems manipulating Binary Logic Signals are commonly referred to as Binary Logic systems
- Digital circuits implementing a particular Binary (Boolean) function are commonly known as Logic Circuits

Elements of Boolean Algebra (Binary Logic)

- As in standard algebra, Boolean algebra has 3 main elements:
  - Constants,
  - Variables, and
  - Operators.

- Logically
  - Constant values are either 0 or 1 Binary Variables $\in$ { 0, 1}
  - 3 Possible Operators: The AND operator, the OR operator, and the NOT operator.

Elements of Boolean Algebra (Binary Logic)

- Physically
  - Constants ⇒
    - Power Supply Voltage (Logic 1)
    - Ground Voltage (Logic 0)
  - Variables ⇒ Signals (High = 1, Low = 0)
  - Operators ⇒ Electronic Devices (Logic Gates)
    - AND - Gate
    - OR - Gate
    - NOT - Gate (Inverter)

Logical Operations

- Three basic logical operations can be applied to binary variables: **AND, OR, NOT**
- The electronic devices which perform the above operations are called: **AND** gate**, OR** gate**, NOT** gate
- A ***truth table*** of a logical operation is a table of all possible combinations of input variable values, and the corresponding value of the output

The AND Operation

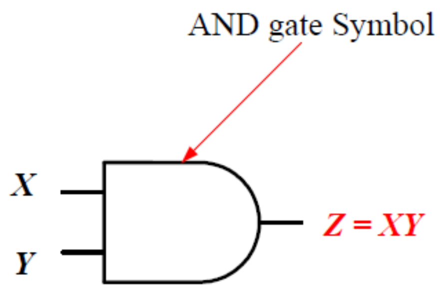
- In Boolean expressions, the AND operation is represented either by a “dot” or by the absence of an operator. Thus, X AND Y is written as X·Y or just XY

AND: $Z = X \cdot Y$ or $Z = XY$

- The AND operation works as follows: If both X and Y have a value of 1, the output Z will be 1 else Z will be 0
- The electronic device which performs the AND operation is called the AND gate

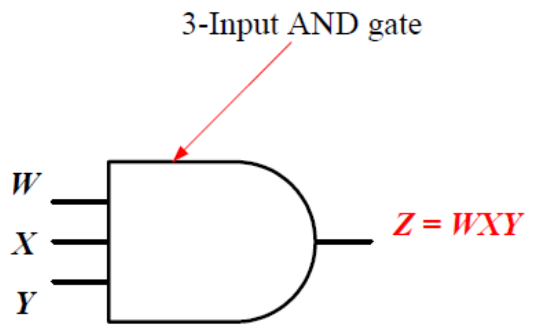
The AND Gate

- 2-input and 3-input AND gates, symbols and truth tables



Truth table of 2-input AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



Truth table of 3-input AND

W	X	Y	Z
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

The OR Operation

- In Boolean expressions, the AND operation is represented either by a “plus” sign. Thus, X OR Y is written as X + Y

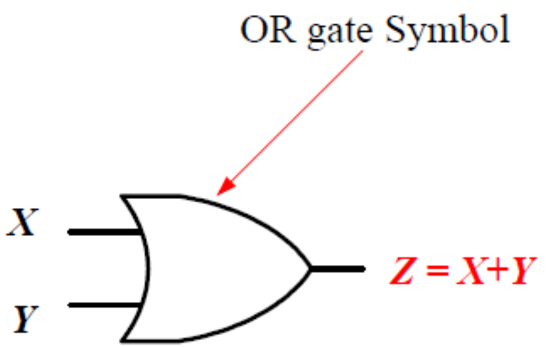
OR: Z = X + Y

- The OR operation implies: If either X or Y have a value of 1, the output Z will be 1

- The electronic device which performs the OR operation is called the OR gate.

The OR Gate

- 2-input and 3-input OR gates, symbols and truth tables

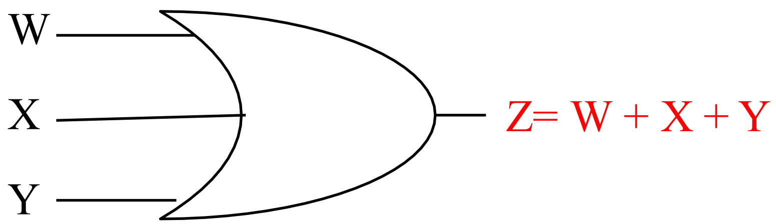


Truth table of 2-input OR

X	Y	Z = X +Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth table of 3-input OR

W	X	Y	Z = W + X + Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



The NOT Operation

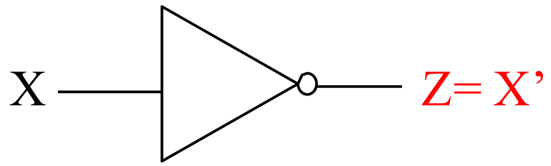
- NOT is a unary operator, meaning there can only be 1 input
- In Boolean expressions, the NOT operation is represented by either a bar on top of the variable (e.g. Z=$\overline{X}$) or a prime (e.g. Z = X').

NOT: Z = X

- If the value of X is a 0, Z is a 1, and if X = 1, Z= 0
- X is also referred to as the *complement* of Z
- The electronic device which performs the NOT operation is called the NOT gate, or simply INVERTER

The NOT Gate (Inverter)

- NOT gate symbol and truth table



Truth table of NOT

NOT	
X	Z = X'
0	1
1	0

Logic Circuits and Boolean Expressions

- Boolean Algebra (summary)

	Regular Algebra	**Boolean Algebra**
Values	Numbers Integers Real numbers Complex Numbers	1 (True, High) 0 (False, Low)
Operators	+, -, x, /, … etc	AND (·) OR (+) NOT (')

Logic Circuits and Boolean Expressions

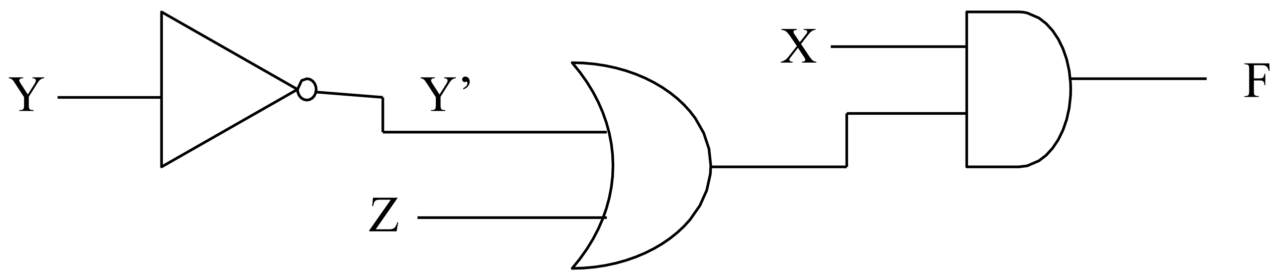
$$F(X, Y) = XY + Y'$$

variable operator literals

- A **Boolean expression** is made of Boolean variables and constants combined with logical operators: AND, OR and NOT
- A *literal* is each instance of a variable or constant
- Boolean expressions are fully defined by their truth tables
- A Boolean expression can be represented using interconnected logic gates
  - Literals correspond to the input signals to the gates
  - Constants (1 or 0) can also be input signals
  - Operators of the expression are converted to logic gates

Logic Circuits and Boolean Expressions

- Example: Consider the function ***F = X · (Y' + Z)***
- This function has three inputs X, Y, Z and the output is given by F
- Logic circuit diagram of ***F = X · (Y' + Z)*** shows the gates needed to construct this circuit are: 2 input AND, 2 input OR and NOT



Logic Circuits and Boolean Expressions

- Another representation of the function F = X · (Y' + Z) is the **truth table**

F = X · (Y' + Z)					
X	Y	Z	Y'	Y' + Z	F = X.(Y'+Z)
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	1	1

Operator Precedence

- Given a Boolean expression, the order of operations depends on the precedence rules given by:

1.	Parenthesis	Highest Priority
2.	NOT	
3.	AND	
4.	OR	Lowest Priority

- Example: XY + WZ will be evaluated as:
  1. XY
  2. WZ
  3. XY + WZ

Basic Identities of Boolean Algebra

Basic Identities of Boolean Algebra

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

Duality Principle

- The principle of duality states that given a Boolean Algebra identity, its dual is obtained by replacing all 1's with 0's and 0's with 1's, all AND operators with OR, and all OR operators with AND

- The dual of an identity is also an identity

AND Identities:	Replacing	OR Identities (Dual):
$X \cdot 1 = X$	*1 with 0*	$X + 0 = X$
$X \cdot 0 = 0$	*0 with 1*	$X + 1 = 1$
$X \cdot X = X$	*+ with ·*	$X + X = X$
$X \cdot \overline{X} = 0$	*· with +*	$X + \overline{X} = 1$

DeMorgan's Theorem

$$\overline{X + Y} = \overline{X} \bullet \overline{Y} \longleftrightarrow \overline{X \bullet Y} = \overline{X} + \overline{Y}$$

Truth Tables to Verify DeMorgan's Theorem

A)	X	Y	X + Y	$\overline{X+Y}$	B)	X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X}\cdot\overline{Y}$
	0	0	0	1		0	0	1	1	1
	0	1	1	0		0	1	1	0	0
	1	0	1	0		1	0	0	1	0
	1	1	1	0		1	1	0	0	0

use truth tables to prove that two Boolean expressions are equal !

Extended DeMorgan's Theorem: $\overline{X_1 + X_2 + \ldots + X_n} = \overline{X_1}\,\overline{X_2}\ldots\overline{X_n}$

$\overline{X_1 X_2 \ldots X_n} = \overline{X_1} + \overline{X_2} + \ldots + \overline{X_n}$

Basic Identities of Boolean Algebra

- Identities of Boolean Algebra can be easily proved using truth tables.
- The only difference between the dual of an expression and the complement of that expression is that in the dual variables are not complemented while in the complement expression, all variables are complemented.
- Using the Boolean Algebra Identities, complex Boolean expressions can be manipulated into a simpler forms resulting in simpler logic circuit implementations.
- Simpler expressions are generally implemented by simpler logic circuits which are both faster and less expensive.

Algebraic Manipulation

- The objective here is to acquire some skills in manipulating Boolean expressions into simpler forms for more efficient implementations.
- Example: Prove that X + XY = X
- Proof: X + XY = X·1 + XY =X·(1 + Y) = X.1 = X

• This is called absorption
• Y has been absorbed

- Example: Prove that X + X\`Y= X + Y
- Proof: X + X\`Y= (X+ X\`) (X + Y)= 1·(X + Y)= X + Y
- OR X + X\`Y= X.1 + X\`Y= X·(1+Y) + X\`Y= X + XY + X\`Y= X + (XY +X\`Y)= X + Y(X +X\`)= X + Y

Algebraic Manipulation

- Example: $F = X`YZ + X`YZ` + XZ$

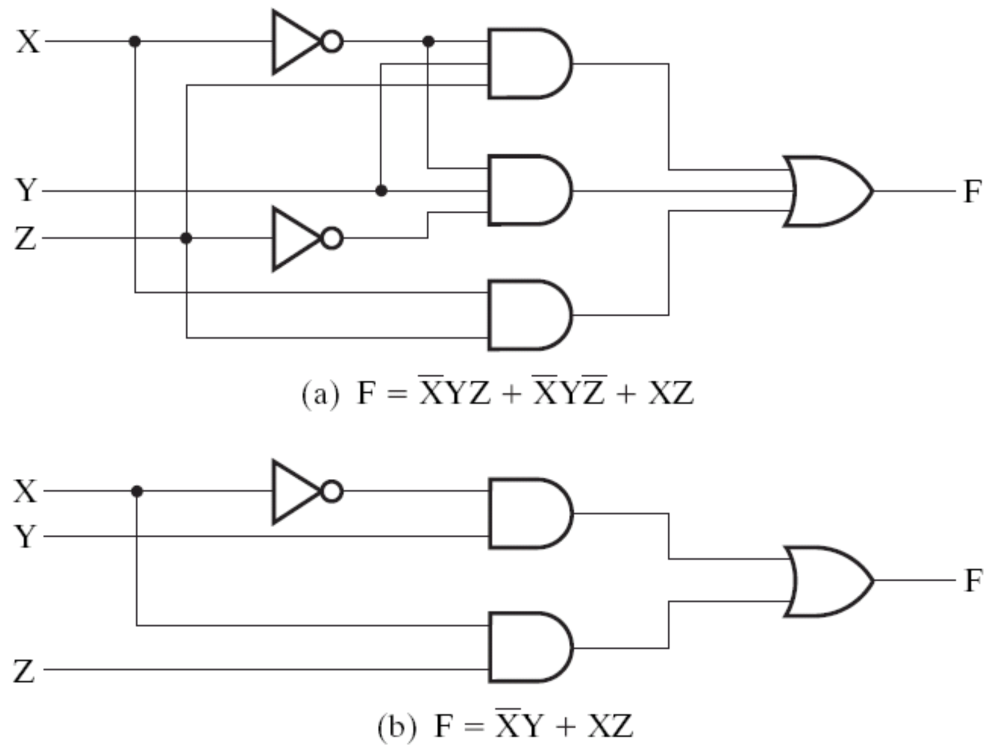
$$= X`Y(Z+Z`) + XZ$$

$$= X`Y \cdot 1 + XZ$$

$$= X`Y + XZ$$

Algebraic Manipulation

- Example: F = X\`YZ + X\`YZ\` + XZ
  = X\`Y(Z+Z\`) + XZ
  = X\`Y·1 + XZ
  = X\`Y + XZ



- Need to verify?

X	Y	Z	(a) F	(b) F
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

- We have reduced the circuit size from 4 gates + 2 inverters to 3 gates + 1 inverter

Consensus Theorem

XY + X'Z + YZ = XY + X'Z

- Y and Z are associated with X and X' in the first two terms and appear together in the third.
- YZ is redundant and can be eliminated.

Prove:

LHS = XY+X'Z+YZ = XY+X'Z+YZ.1
= XY+X'Z+YZ(X+X')
= XY+X'Z+XYZ+X'YZ
= XY+XYZ+X'Z+X'YZ
= XY(1+Z)+X'Z(1+Y)
= XY·1 + X'Z·1
= XY + X'Z = RHS

Adding an extra step to acquire the desired output

Algebraic Manipulation

- Example: Reduce F1 = (A + B’ + AB)’ (AB + A’C + BC)’

Using DeMorgan’s Theorem,

F1 = (A’.B.(A’+B')).(A’+B’).(A+C’).(B’+C’)

= (A’.B.A’ + A’.B.B’).(A’+B’)(A+C’) .(B’+C’)

= (A’B + 0).(A’+B’)(A+C’) .(B’+C’)

= (A’BA’ + A’BB’) (A+C’) .(B’+C’)

= (A’B) (A+C’) .(B’+C’)

= (A’BA+A’BC’)(B’+C’)

= (0+A’BC’)(B’+C’)

= (A’BC’B’ + A’BC’C’)

= (0 + A’BC’) = <u>A’BC’</u>

Algebraic Manipulation

- Example: Simplify the function $\mathbf{G} = \overline{\left(\overline{(A+\overline{B}+C)}.(\overline{AB}+\overline{C}\,\overline{D}) + \overline{ACD} \right)}$

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

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

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

$$= (ACD + ACD\overline{B}) + (ACDB + ACDB)$$

$$= ACD + ACDB$$

$$= ACD$$

Complement of a Function

- The complement of a function F is $\overline{F}$
- $\overline{F}$ is obtained in two ways:
  1. Truth Table: Change 1s to 0s and 0s to 1s
  2. Boolean Expression: Apply DeMorgan's Theorem (Be Careful, before begin, surround all AND terms with parentheses (easy to make mistake!))
     - Short cut: Take the dual and complement the literals.

Truth table

A	B	F	F'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Conclusion

- Boolean Algebra is a mathematical system to study logic circuits
- Boolean identities and algebraic manipulation can be used to simplify Boolean expressions.
- Simplified Boolean expressions result in simpler and, generally, faster circuits.