

Lecture – 3

Chapter 2: Sections 1 and 2

Outline

- **Boolean Algebra**
- **Basic Theorems and Properties of Boolean Algebra**
- **Boolean Functions**

Digital circuit

- **Digital circuit:** hardware components that manipulate binary information.
- Each basic circuit: **logic gate**
- Each gate: performs a specific logical operation.
- **Boolean Algebras:**
 - describe the operational properties of digital circuits
 - specify the operation of each gate
 - design logic circuit through the manipulation of Boolean expressions

Binary Variables

- **Recall that the two binary values have different names:**
 - True/False
 - On/Off
 - Yes/No
 - 1/0
- **We use 1 and 0 to denote the two values.**
- **Variable identifier examples:**
 - A, B, y, z, or X_1 for now
 - RESET, START_IT, or ADD1 later

Logical Operations

- **The three basic logical operations are:**
 - AND
 - OR
 - NOT
 - NAND
 - NOR
 - XOR

- **AND is denoted by a dot (\cdot).**
- **OR is denoted by a plus ($+$).**
- **NOT is denoted by a single quote mark ($'$) or an overbar ($\bar{}$).**
- **NAND (\cdot)'**
- **NOR ($+$)'**
- **XOR (\oplus)**

Notation Examples

- **Examples:**

- **$Y = A \cdot B$ is read “Y is equal to A AND B.”**
- **$Z = x + y$ is read “z is equal to x OR y.”**
- **$X = A'$ is read “X is equal to NOT A.”**

Gates

Let's examine the processing of the following six types of gates

1. NOT
2. AND
3. OR
4. XOR
5. NAND
6. NOR

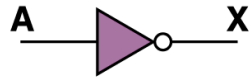
Typically, logic diagrams are black and white, and the gates are distinguished only by their shape

NOT Gate

A NOT gate accepts one input value and produces one output value

By definition, if the input value for a NOT gate is 0, the output value is 1, and if the input value is 1, the output is 0

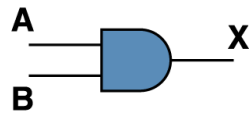
A NOT gate is sometimes referred to as an inverter because it inverts the input value

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

AND Gate

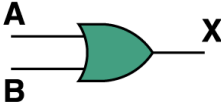
An AND gate accepts two input signals.

If the two input values for an AND gate are both 1, the output is 1; otherwise, the output is 0

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \cdot B$		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>X</th> </tr> </thead> <tbody> <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> </tbody> </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

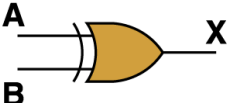
If the two input values are **both** 0, the output value is 0; otherwise, the output is 1

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A + B$		<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th>A</th> <th>B</th> <th>X</th> </tr> </thead> <tbody> <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> </tbody> </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

XOR, or *exclusive* OR, gate

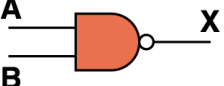
- An XOR gate produces 0 if its **two inputs are the same**, and a 1 otherwise
- **Note** the difference between the XOR gate and the OR gate; they differ only in one input situation
- When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \oplus B$		<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th>A</th> <th>B</th> <th>X</th> </tr> </thead> <tbody> <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> </tbody> </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															

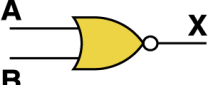
NAND and NOR Gates

The NAND and NOR gates are essentially the **opposite** of the AND and OR gates, respectively

NAND Gate

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

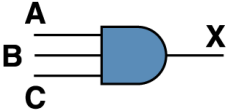
Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = (A + B)'$		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>X</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr>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

Review of Gate Processing

- A **NOT** gate **inverts** its single input value
- An **AND** gate produces 1 if both input values are 1
- An **OR** gate produces 1 if one or the other or both input values are 1
- An **XOR** gate produces 1 if one or the other (but not both) input values are 1
- A **NAND** gate produces the **opposite** results of an **AND** gate
- A **NOR** gate produces the **opposite** results of an **OR** gate

Gates with More Inputs

- Gates can be designed 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 border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> <th>X</th> </tr> </thead> <tbody> <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> </tbody> </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																																			

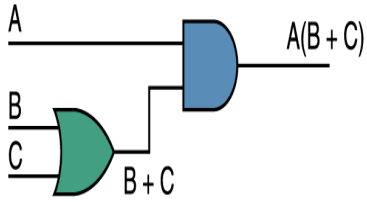
Combinational Circuits

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

Consider the following Boolean expression $A(B + C)$

Digital Logic

The Truth table and the draw is :



A	B	C	B + C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Boolean algebra

Boolean algebra allows us to apply provable mathematical principles to help us design logical circuits

An algebraic structure defined on a set of at least two elements $B=\{0, 1\}$, together with three binary operators (denoted $+$, \cdot and $'$) that satisfies the following basic identities:

Properties of Boolean Algebra

Closure: The structure is closed with respect to the operator $+$, \cdot , $'$

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	$A(A') = 0$	$A + (A') = 1$
DeMorgan's law	$(AB)' = A' \text{ OR } B'$	$(A + B)' = A'B'$

Theorem

- | | | |
|------------------------------------|------------------------------|-------------|
| 1. (a) $X + X = X$ | (b) $X \cdot X = X$ | |
| 2. (a) $X + 1 = 1$ | (b) $X \cdot 0 = 0$ | |
| 3. (a) $(X)' = X$ | | |
| 4. (a) $(X + Y) + Z = X + (Y + Z)$ | (b) $(XY)Z = X(YZ)$ | Associative |
| 5. (a) $(X + Y)' = X' \cdot Y'$ | (b) $(X \cdot Y)' = X' + Y'$ | DeMorgan |
| 6. (a) $X + XY = X$ | (b) $X(X + Y) = X$ | absorption |

Boolean Operator Precedence

The order of evaluation in a Boolean expression is:

- Parentheses
- NOT
- AND
- OR

Consequence: Parentheses appear around OR expressions

Example: $F = A(B + C)(C + D')$

Example 1 : Boolean Algebraic Proof

$x + \underline{x \cdot y} = x$	(Theorem 6(a))
Proof Steps	Justification
$x = x + x \cdot y$	Our assertion (after commutation)
$= x \cdot 1 + x \cdot y$	Identity Postulate
$= x \cdot (1+y)$	Distributive Postulate
$= x \cdot 1$	from definition of OR operation
$= x$	Identity Postulate

Example 2 : Boolean Algebra Proof

$x(x+y)=x$ (Theorem 6(b)): Let's use some truth tables to do this ... but first

Proof Steps

Justification

$x = x(x + y)$

Our assertion (after commutation)

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

Distributive Postulate

x	y	xx	xy	xx + xy
0	0	0	0	0
0	1	0	0	0
1	0	1	0	1
1	1	1	1	1

Some Properties of Identities & the Algebra

- If the **meaning is unambiguous**, we leave out the symbol “ \cdot ”
- **Duality principle:** The dual of a boolean algebraic expression is obtained by interchanging $+$ and \cdot and interchanging 0’s and 1’s.

Example: $z = x \cdot y$
Dual: $\rightarrow x + y$

AND	Dual
$0 \cdot 0 = 0$	$1 + 1 = 1$
$0 \cdot 1 = 0$	$1 + 0 = 1$
$1 \cdot 0 = 0$	$0 + 1 = 1$
$1 \cdot 1 = 1$	$0 + 0 = 0$

Example: $f = (a + c') \cdot b + 0$
Dual: $\rightarrow (a \cdot c' + b) \cdot 1$
 $\rightarrow a \cdot c' + b$

a	b	c	f	F dual
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

One more Dual

Example: $g = x \cdot y + (w + z)'$

Dual: $\rightarrow (x + y) \cdot (w \cdot z)'$

x	y	w	z	g	g dual
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	0

De Morgan's Laws (Theorem)

Formalization credited to [Augustus De Morgan](#), a British mathematician.

He is also credited with initial work on the rules of mathematical induction.

In plain English we might say, "Since it is false that two things are both true, at least one of them must be false."

The Laws:

The Not X or Y rule:

- $\text{NOR}(X, Y) = \text{AND}(X', Y')$
- $(X + Y)' = X' \cdot Y'$

The Not X and Y rule:

- $\text{NAND}(X, Y) = \text{OR}(X', Y')$
- $(X \cdot Y)' = X' + Y'$

Adders

- At the digital logic level, addition is performed in binary
- Addition operations are carried out by special circuits called, appropriately, **adders**
- The result of adding two binary digits could produce a **carry value**
- Recall that $1 + 1 = 10$ in base two

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

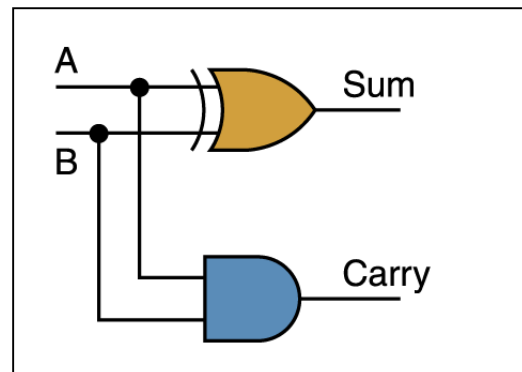
- A circuit that computes the sum of two bits and produces the correct carry bit is called a **half adder**

Circuit diagram representing a **half adder**

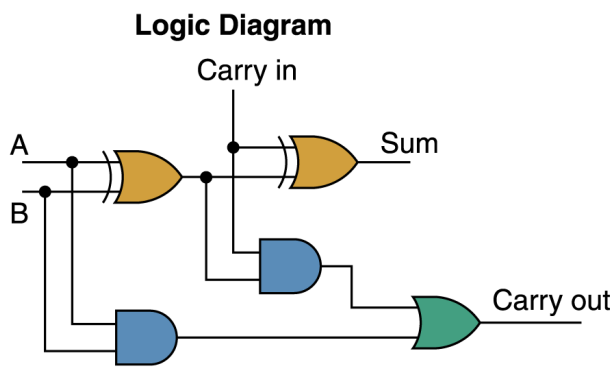
Two Boolean expressions:

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

$$\text{carry} = AB$$



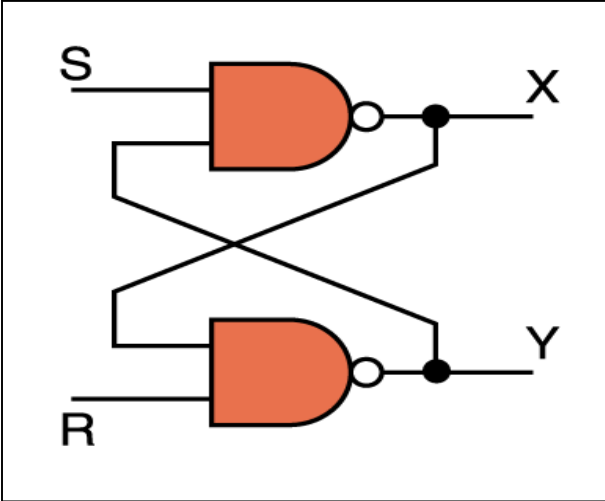
A circuit called a **full adder** takes the carry-in value into account



Truth Table

A	B	Carry-in	Sum	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

Circuits as Memory

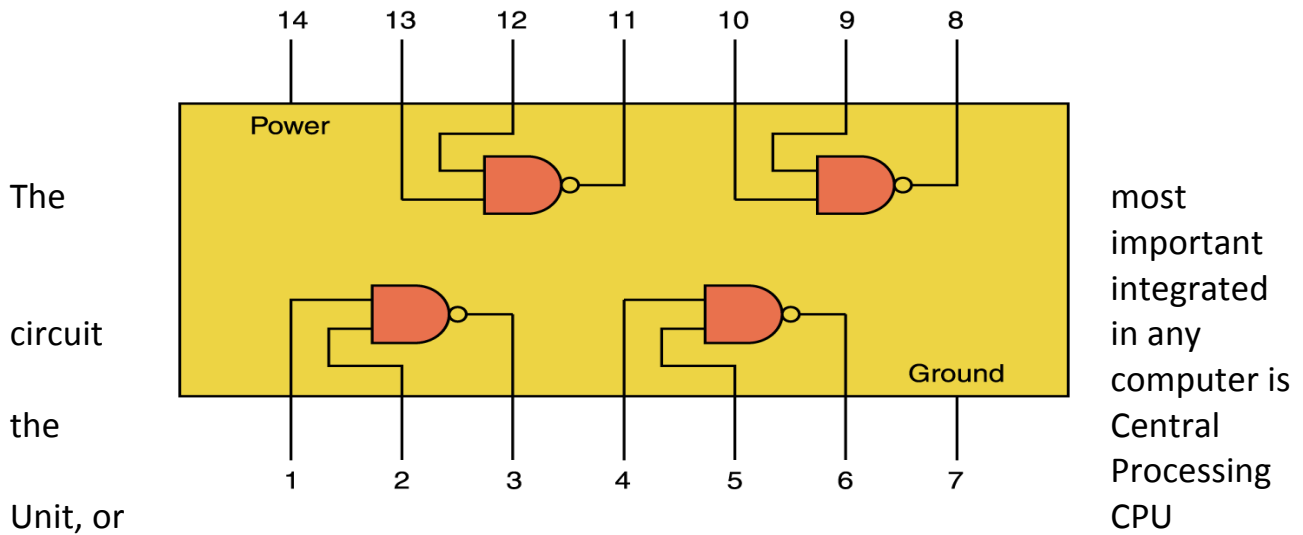
- Digital circuits can be used to store information
 - These circuits form a sequential circuit, because the output of the circuit is also used as input to the circuit
 - An S-R latch (Set-Reset) stores a single binary digit (1 or 0)
 - There are several ways an S-R latch circuit could be designed using various kinds of gates
 - The design of this circuit guarantees that the two outputs X and Y are always complements of each other
 - The value of X at any point in time is considered to be the current state of the circuit
- 
- Therefore, if X is 1, the circuit is storing a 1; if X is 0, the circuit is storing a 0

Integrated Circuits

- Integrated circuit (also called a chip) A piece of silicon on which multiple gates have been embedded
- These silicon pieces are mounted on a plastic or ceramic package with pins along the edges that can be soldered onto circuit boards or inserted into appropriate sockets
- Integrated circuits (IC) are classified by the number of gates contained in them

Abbreviation	Name	Number of Gates
SSI	Small-Scale Integration	1 to 10
MSI	Medium-Scale Integration	10 to 100
LSI	Large-Scale Integration	100 to 100,000
VLSI	Very-Large-Scale Integration	more than 100,000

Example : An SSI chip contains independent NAND gates



Each CPU chip has a large number of pins through which essentially all communication in a computer system occurs