OF THE
DEPARTMENT OF
INFORMATION ENGINEERING

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Logic Gates

Boolean Algebra

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering

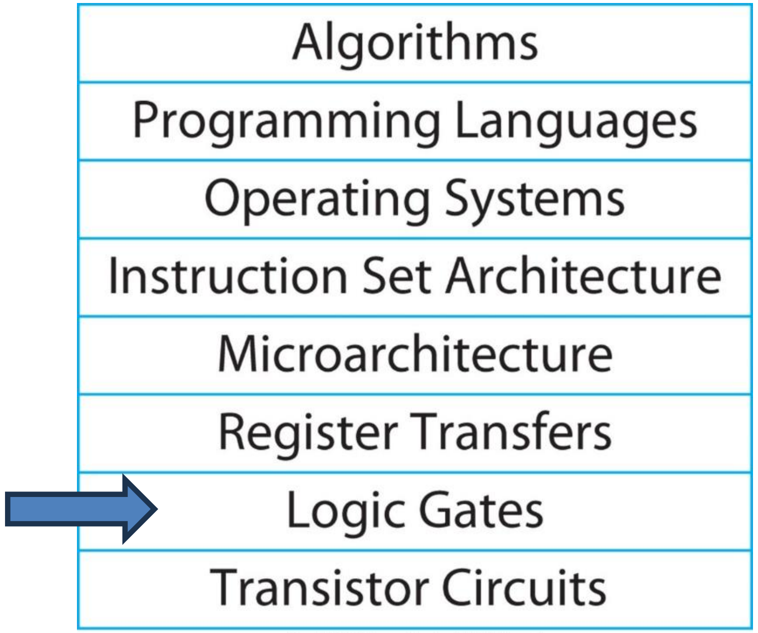
Academic Year 2023-2024

Purpose of the Lesson

- Study the **logic gates** and how to use them to realize digital circuits with given relations between inputs and outputs
- Introduce **Boolean algebra** and identify the rules to manipulate the logic functions in order to make them suitable for the design of a digital circuit

Digital Circuits and Logic Gates

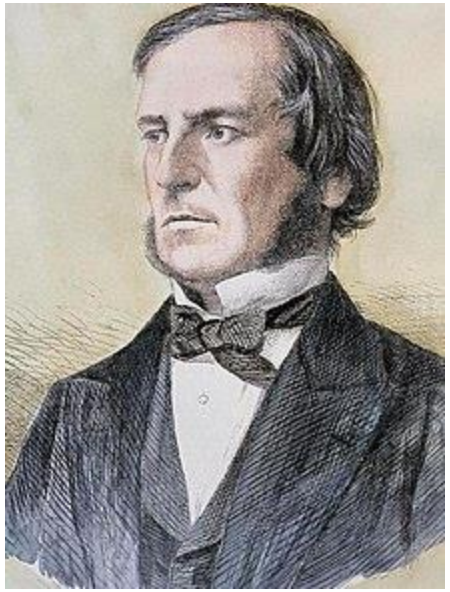
- **Digital circuits** are hardware components that process binary information
- The basic building blocks of these circuits are the **logic gates**
- The purpose of a logic gate is to **perform logic operations between its inputs (binary) and produce an output (binary)**



In turn, the logic gates are made with a set of transistors and interconnects

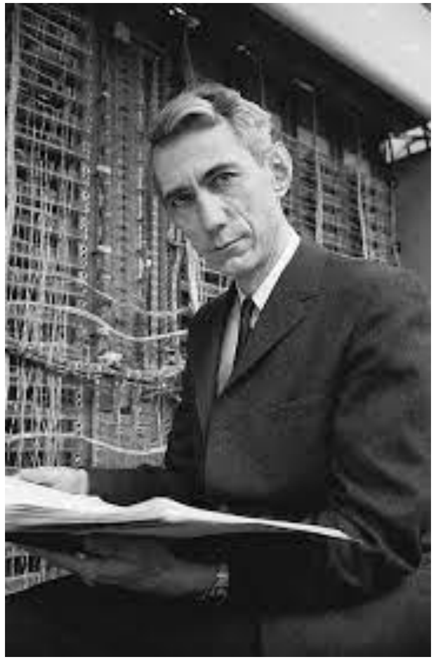
Boolean Algebra

- Algebra = discipline studying algebraic structures (i.e., sets equipped with one or more operations)
- Boolean algebra was proposed by **George Boole** (1815-1864) in two texts (“The Laws of Thought”, 1854)
  – Is characterized by the fact that the **variables can only assume 2 values: "true" - "false"**
  – It has properties, such as existence of minimum/maximum, existence of complement, distributive property, etc. (more on this later)
- In the following, after introducing some practical concepts (logic functions, logic gates, …), we will study the axioms of Boolean algebra



Boolean Algebra

- Claude E. Shannon (1916-2001) in his Master thesis in Electrical Engineering at MIT “A Symbolic Analysis of Relay and Switching Circuits” (1938) demonstrated that **electrical signals propagating through a network of switches (only 2 states: open/closed) follow the Boolean algebra rules**, with true/false values matching the open/closed state of a switch
- In general a **digital circuit can be described by a Boolean expression**, that can be handled with the rules of Boolean algebra
  – We use Boolean algebra for the analysis and synthesis (design) of digital systems
  – We will decompose a complex Boolean function into simpler functions, or obtain a normal expression from a truth table, etc.



Binary Logic

- Logic gates operate in **binary logic** (= Boolean logic), i.e. with variables that can only assume 2 discrete values: '**0**' and '**1**'
- There are **three fundamental operations in binary logic**
  - **AND**
  - **OR**
  - **NOT**

=> All Boolean expressions can be written in terms of these 3 operations

Truth Table of a Logic Function

- A logic function can be described by its **truth table**
- The **truth table of a logic function** is a table that lists the values of the output of the function for **<u>all</u>** the possible combinations of the inputs of the function
- The truth table of a function with **n inputs** has **2^n rows**

AND Logic Function

- AND is a logic function with two or more variables as input
- Notations: $Z = X \cdot Y$ $Z = XY$ $Z = X \wedge Y$ Z = X **AND** Y
- <u>Z = 1 if and only if X = 1 and Y = 1</u>

  <u>otherwise Z = 0</u>
- **Truth Table**

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Logic Function

- OR is a logic function with two or more variables as input
- Notations: Z = X **+** Y   Z = X **V** Y   Z = X **OR** Y
- <u>Z = 1 if at least one between X and Y is 1</u>

  <u>otherwise Z = 0</u>
- **Truth Table**

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT Logic Function

- NOT is a logic function with one input variable, also called complement function (negates the input)
- Notations: $Z = \overline{X}$ $Z = \sim X$ Z = **NOT** X
- Z = 1 if X = 0

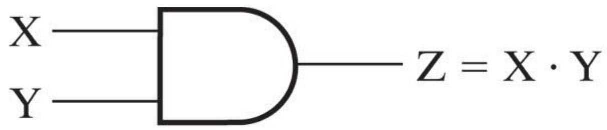
  Z = 0 if X = 1
- **Truth Table**

X	NOT X
0	1
1	0

Logic Gates

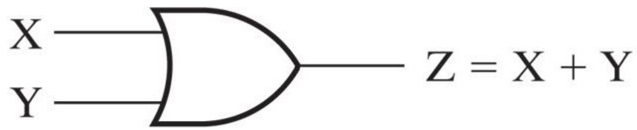
- The AND, OR, NOT logic gates are circuits producing as an output the value indicated by the corresponding truth table

- **AND gate**



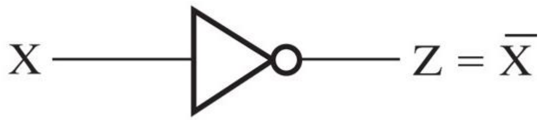
AND gate

- **OR gate**



OR gate

- **NOT gate** (inverter)

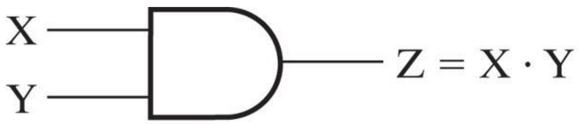


NOT gate or inverter

Logic Gates

AND

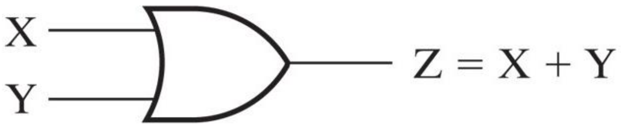
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1



AND gate

OR

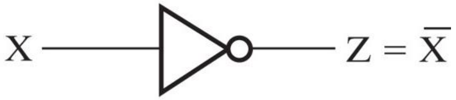
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1



OR gate

NOT

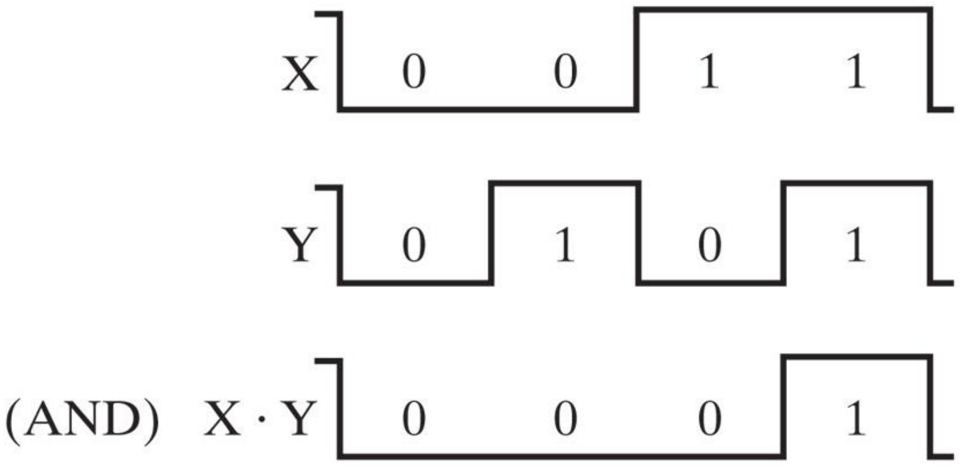
X	$Z = \overline{X}$
0	1
1	0



NOT gate or inverter

Time Diagram

- A time diagram describes the evolution of the **input signals** versus time and the corresponding **evolution of the output signals**
- The **X axis shows the time and the Y axis plots a binary signal** (which can only take the values '0' or '1')
  - Example: time diagram of an AND gate

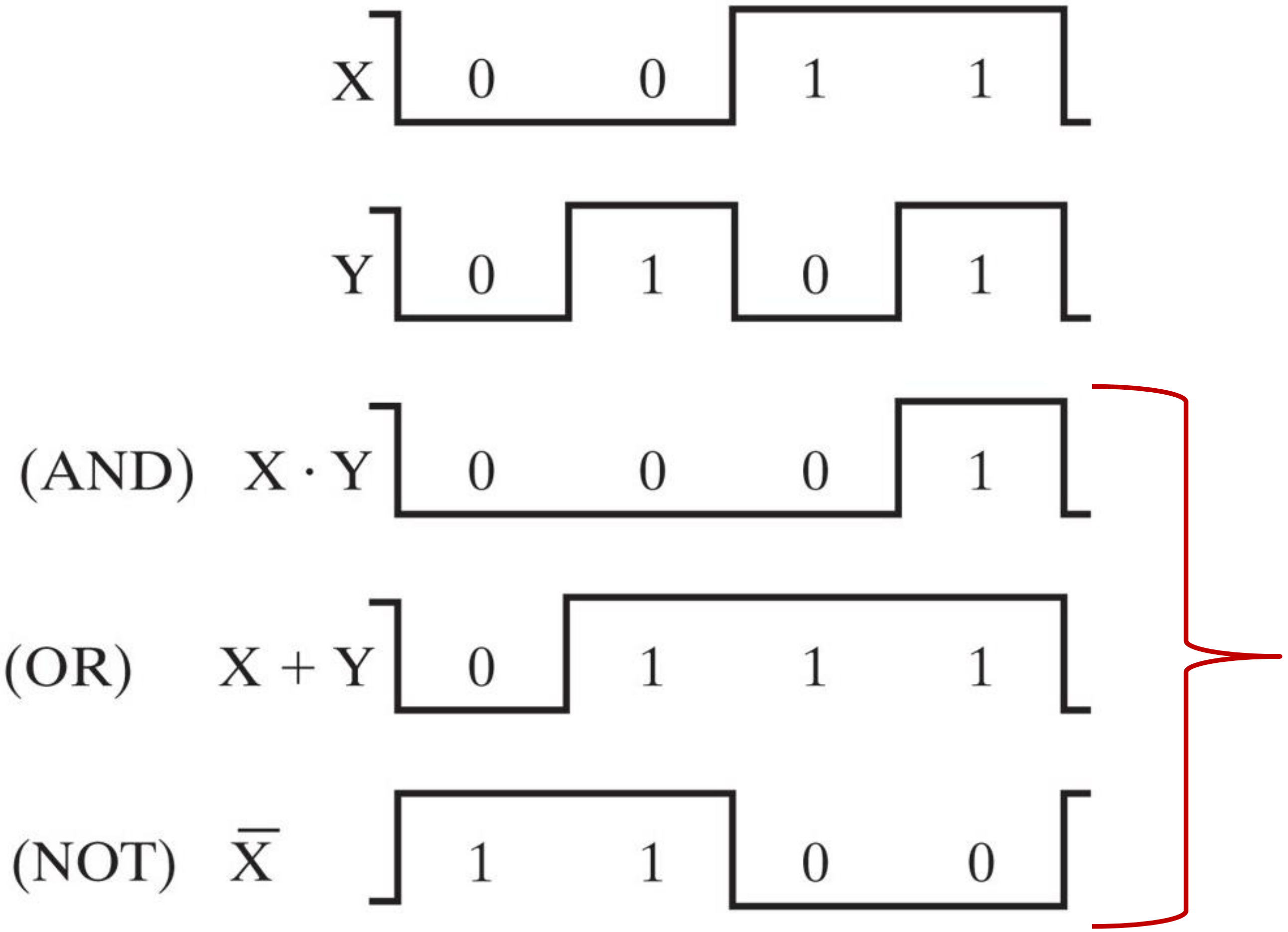


This is an ***ideal time diagram***: the output transition occurs ***instantaneously*** as the inputs change

Delay Time of a Logic Gate

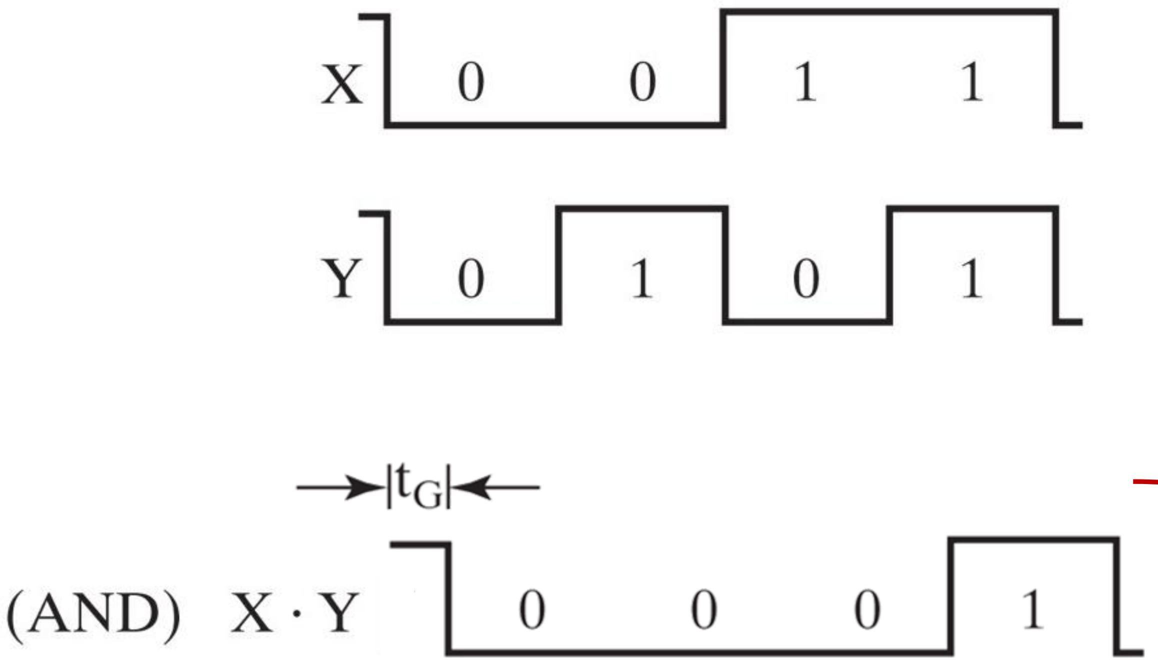
- An **ideal logic gate** has a zero delay time (a change in input produces an instantaneous change in the output)
- A **real logic gate** is characterized by a given **time delay** ($\boldsymbol{t_{G.}}$: gate delay)
- t_G is the time elapsing between **the change of the inputs and the consequent change in the outputs**
- The gate delay depends on the technology with which the logic gate is made
- The time to switch from logic low to logic high is not necessarily the same as the high to low transition

Time Diagrams of Ideal Logic Gates



Time diagrams
of **ideal gates**
(zero delay)

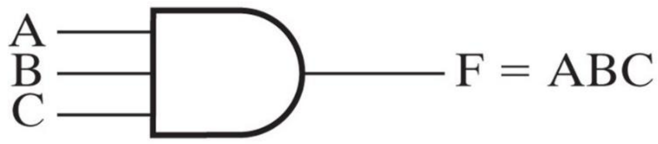
Time Diagrams of Real Logic Gates



AND gate with **delay $\boldsymbol{t_G}$**
($t_{G,\ 0\text{-}>1} = t_{G,\ 1\text{-}>0}$)

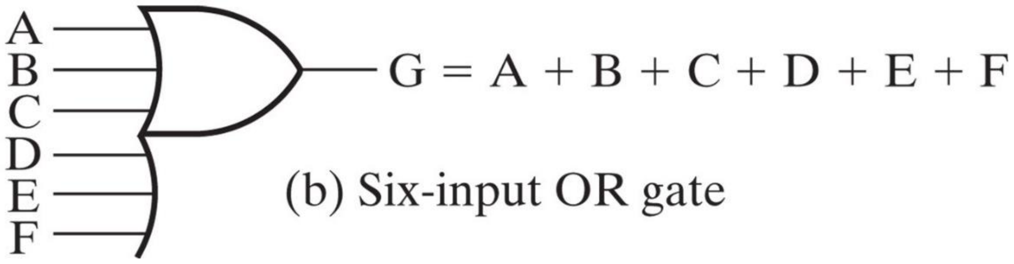
Multi-Input Logic Gates

- A logic gate can also have **more than two inputs**
  - Example 1: 3-input AND gate



(a) Three-input AND gate

  - Example 2: 6-input OR gate



(b) Six-input OR gate

Other Logic Gates

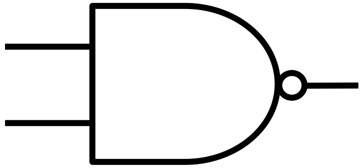
- There are other frequently used logic gates, besides the elementary gates (AND, OR, NOT):
  - **NAND**: complement of the AND gate
  - **NOR**: complement of the OR gate
  - **XOR**: exclusive OR
  - **XNOR**: exclusive NOR (complement of the XOR gate)

- Notation: the **bubble** at the output (or input) of a logic gate stands for a NOT operation (complement)

NAND Gate

- NAND gate has two or more inputs
- Notations: $Z = \overline{X \cdot Y}$ $\quad Z = \overline{X\,Y}$ $\quad Z = X$ **NAND** Y
- <u>Z = 1 if at least one between X and Y is equal to 0</u>

  <u>otherwise Z = 0</u>



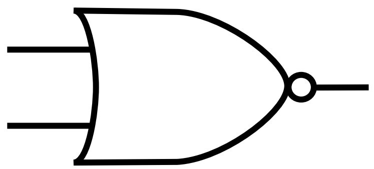
- **Truth table**

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

- NOR gate has two or more inputs
- Notations: $Z = \overline{X + Y}$ $Z = \overline{X \vee Y}$ Z = X **NOR** Y
- <u>Z = 1 only if both X and Y are equal to 0</u>

  <u>otherwise Z = 0</u>



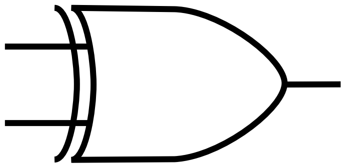
- **Truth table**

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate

- XOR gate has two or more inputs
- Notations: $Z = X \oplus Y$ Z = X **XOR** Y
- Z = 1 if only one between X and Y is 1 (i.e. if X and Y are different)

  otherwise Z = 0



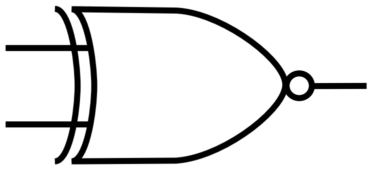
- **Truth table**

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate

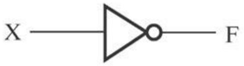
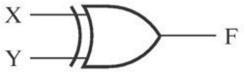
- XNOR gate has two or more inputs
- Notations: $Z = \overline{X \oplus Y}$ Z = X **XNOR** Y
- Z = 1 if both X and Y are equal to 0 or equal to 1 (if X is equal to Y)

  otherwise Z = 0



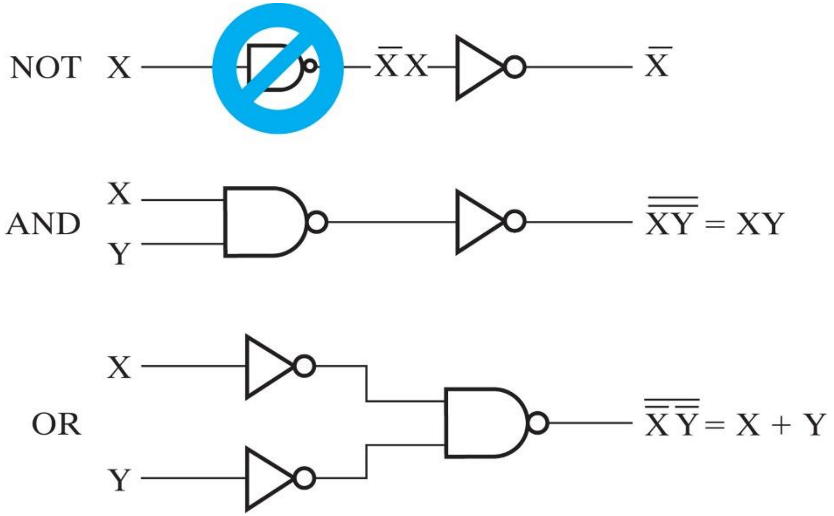
- **Truth table**

X	Y	X XNOR Y
0	0	1
0	1	0
1	0	0
1	1	1

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
AND		$F = XY$	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
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\,\overline{Y}$ $= \overline{X \oplus Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Universal Logic Gates

- All Boolean functions can be represented with **NAND gate only** (considering the inverter as a particular case of NAND gate with a single input)



The equivalence of these functions will be proved in the following slides (De Morgan's Theorem)

- The same is true for the **NOR gate**

=> NAND and NOR are called **universal gates** (or complete gates)

VHDL Representation of Logic Gates

Default **operators** to represent **logic gates in VHDL**:

VHDL logic operator	Example
`not`	`F <= not X;`
`and`	`F <= X and Y;`
`or`	`F <= X or Y;`
`nand`	`F <= X nand Y;`
`nor`	`F <= X nor Y;`
`xor`	`F <= X xor Y;`
`xnor`	`F <= X xnor Y;`

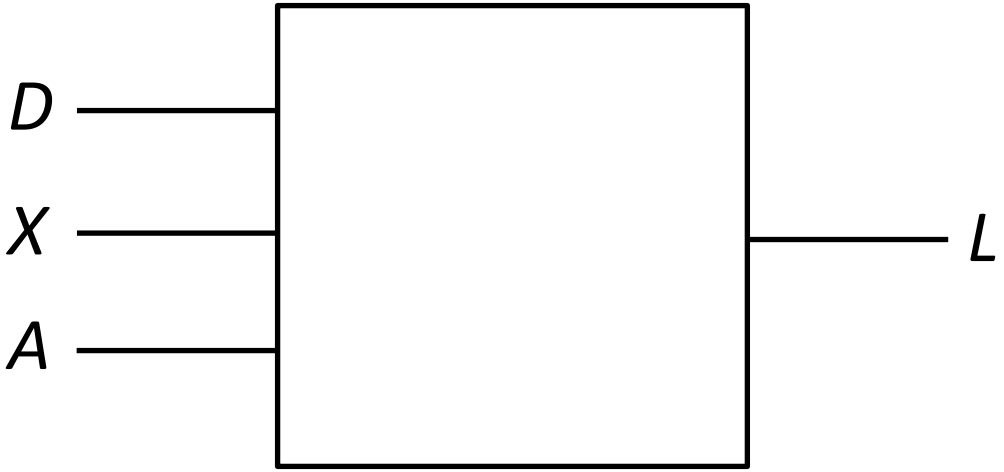
Boolean Functions

Boolean Expression

- A Boolean expression is an algebraic expression composed of **binary variables** and **binary constants** ('0' and '1'), **logic basic operation symbols (AND, OR, NOT)**, and parenthesis

Example of a Boolean Function

- Three-variable Boolean function for lowering a car electric window



$$\boldsymbol{L = D\bar{X} + A}$$

Example of a Boolean Function

- Three-variable Boolean function to control a car electric window: $\boldsymbol{L = D\overline{X} + A}$
  - Output *L* controls the engine that lowers the window: *L* = '1’ activates the motor, *L* = '0’ leaves the motor inactive
  - Input *D* = ‘1’: if the driver presses the window down button
  - Input *X* is connected to the output of a limit switch: *X* = '1' when the window is completely down
  - Input *A* (automatic window lowering) is a signal generated by the timing logic between *D* and *X*: if *D* stays at '1' for more than 0.5 s, *A* becomes '1' and does not change until *X* = '1'

➢ The window is lowered if at least one of the following is '1':

- *D* **AND NOT**(*X*): the driver presses the button <u>and</u> the window is not already completely down
- *A*: the driver presses the button to lower the window for more than 0.5 s

Example of a Boolean Function

- Truth table

Truth Table for the Function $L = D\overline{X} + A$

D	X	A	L
0	0	0	0
0	0	1	1
0	1	0	0
~~0~~	~~1~~	~~1~~	~~1~~
1	0	0	1
1	0	1	1
1	1	0	0
~~1~~	~~1~~	~~1~~	~~1~~

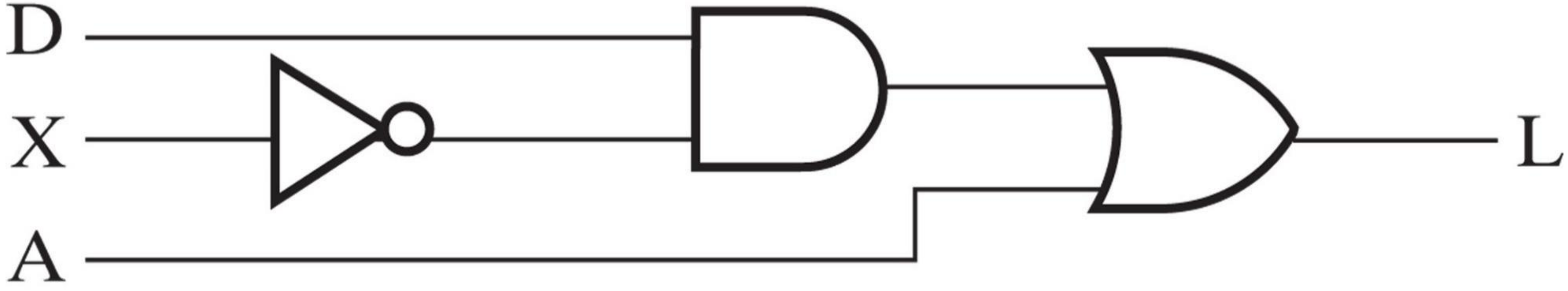
Some of the combinations are impossible! (if limit switch is ‘1’, A = ‘0’)

- The truth table of a n-variable function has 2^n rows (each corresponding to a combination of '0' and '1' assigned to the variables) and shows the value of the function corresponding to those values of the variables

Example of a Boolean Function

- The Boolean function can be translated into a circuit made of logic gates that operate on input variables to produce output variable(s)

- The logic circuit for lowering the car electric window seen in the example is:



$$L = D\overline{X} + A$$

Boolean Algebra

Boolean Algebra

- Boolean algebra (George Boole, 1854) includes a series of **postulates, properties, and theorems to manipulate Boolean functions**, which can be used to simplify expressions
- Our aim is **minimize** the number of logic gates in the circuit and the number of inputs
- **Simplifying** (getting simpler expressions) typically means using less hardware, resulting in faster circuits and less silicon area (=> lower $)

Duality of Boolean Algebra

- The **dual expression** of a logic expression is obtained
  - By replacing the **AND** gates with **OR** gates, and viceversa

  and
  - By replacing the constants **'0'** with **'1'** and **'1'** with **'0'**
- In Boolean algebra the duality principle holds: **if an equation is true, its dual is also true** (although it is different from the original one)

$$\cdot \rightleftarrows +$$

$$0 \rightleftarrows 1$$

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.	$\overline{\overline{X}} = X$		

- Relations between a variable, the complemented variable, and the logic constants '0' and '1'
- NOTE: the two columns contain **dual equations**

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. $\overline{\overline{X}} = X$	

- We can prove them with the **perfect induction principle**, i.e. we prove that they are valid in **all possible cases**:

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. $\overline{\overline{X}} = X$

- We can prove them with the **perfect induction principle**, i.e. we prove that they are valid in **all possible cases**:

$\mathbf{X}$	$\mathbf{X + 0}$	$\mathbf{X + 1}$	$\mathbf{X + X}$	$\mathbf{X + \overline{X}}$	$\overline{\overline{\mathbf{X}}}$
0	0	1	0	1	0
1	1	1	1	1	1

Properties of Boolean Algebra

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

- Identities **1.-8. and 10.-15.** are the **Boolean algebra axioms**, i.e. we are dealing with Boolean algebra <u>if and only if</u> these properties are verified
- In the absence of parentheses, **AND takes precedence over OR** (analogous to ordinary algebra, where multiplication takes precedence over addition)
- It is advisable to always use parentheses, even if they are implied!
- Some of these properties are also valid in ordinary algebra (10-14), others not (15-17)

De Morgan's Theorem

$$\overline{X + Y} = \overline{X} \cdot \overline{Y} \qquad\qquad \overline{X \cdot Y} = \overline{X} + \overline{Y}$$

- Very important theorem in Boolean logic, it can be used **to get the complement of an expression**
- It can be extended to more than two variables:

$$\overline{X_1 + X_2 + \ \ldots + Xn} = \overline{X_1} \cdot \overline{X_2} \cdots \overline{X_n}$$
$$\overline{X_1 \cdot X_2 \cdot \cdots \cdot X_n} = \overline{X_1} + \overline{X_2} + \cdots + \overline{X_n}$$

- **Generalized form**: the complement of a logic expression can be obtained by replacing AND with OR and vice versa and negating all variables and constants

De Morgan’s Theorem: Proof

$$\overline{\boldsymbol{X}+\boldsymbol{Y}}=\overline{\boldsymbol{X}}\cdot\overline{\boldsymbol{Y}} \qquad \overline{\boldsymbol{X}\cdot\boldsymbol{Y}}=\overline{\boldsymbol{X}}+\overline{\boldsymbol{Y}}$$

- It can be demonstrated replacing the variables with all possible combinations of their value (**perfect induction principle**)

De Morgan's Theorem: Proof

$$\overline{X+Y} = \overline{X} \cdot \overline{Y} \qquad\qquad \overline{X \cdot Y} = \overline{X} + \overline{Y}$$

- It can be demonstrated replacing the variables with all possible combinations of their value (**perfect induction principle**)

(a) X	Y	X + Y	$\overline{X+Y}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

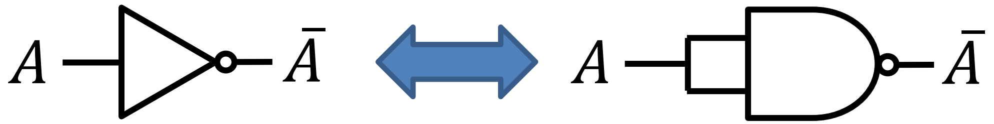
(b) X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X} \cdot \overline{Y}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

NAND and NOR as Universal Operators

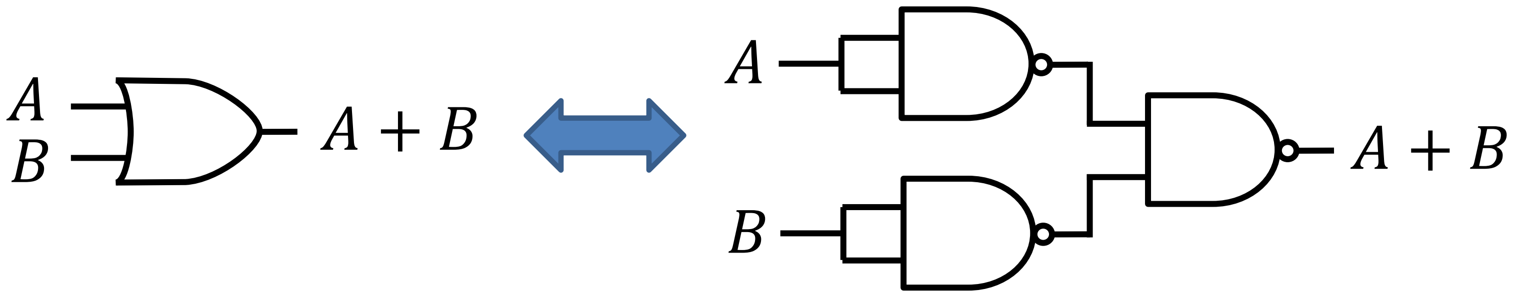
$$\overline{X+Y} = \bar{X} \cdot \bar{Y} \qquad\qquad \overline{X \cdot Y} = \bar{X} + \bar{Y}$$

- Every logic function can be expressed with only NAND or only NOR logic operators
- Ex: NOT and OR can be represented only with NOR gates

NOT:



OR: $A + B = \overline{\overline{A+B}} = \overline{\bar{A} \cdot \; \bar{B}}$ (De Morgan)



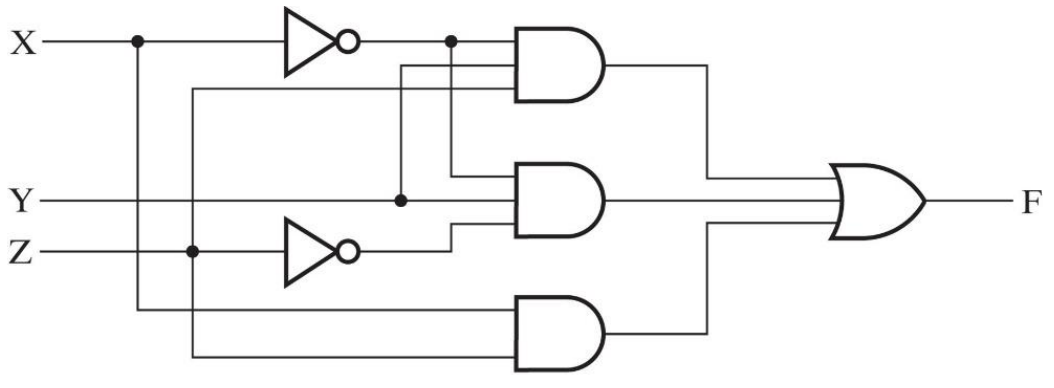
Example: Application of Properties

$$F = \boxed{\bar{X}YZ + \bar{X}Y\bar{Z} + XZ}$$

$= \bar{X}Y(Z + \bar{Z}) + XZ$ → distributive property

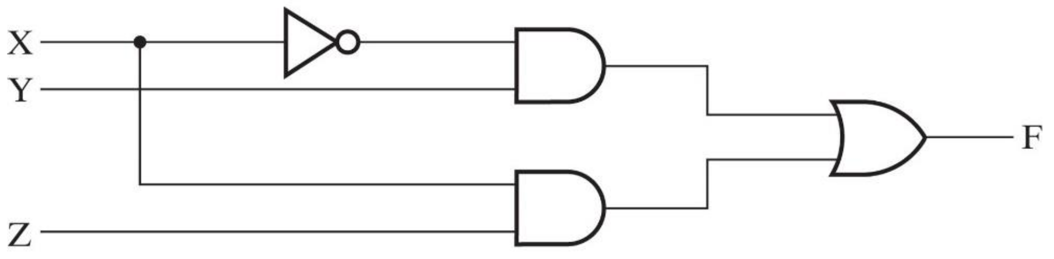
$= \bar{X}Y \cdot 1 + XZ$ → identity # 7: $X + \bar{X} = 1$

$= \boxed{\bar{X}Y + XZ}$ → identity # 2: $X \cdot 1 = \mathrm{X}$



(a) $F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$

=> The two expressions are equivalent, but (b) saves in terms of number of gates!



(b) $F = \bar{X}Y + XZ$

Logic Gates and Literals

- Let's consider the implementation of a Boolean equation with logic gates in the form of the sum of products
  - Every **product term** in the equation represents an AND logic gate
  - Every **variable** in the product term represents an input of the AND gate
- Every **separate occurrence of a variable, direct or complemented**, is defined **literal**
  - Examples:

(a) $F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$ → 3 terms, 8 literals

(b) $G = \bar{X}Y + XZ$ → 2 terms, 4 literals

Absorption Theorem

$$\boldsymbol{X + XY = X}$$

(intuitively: the second term can be omitted because it is redundant)

- Prove the theorem
    1. Using the perfect induction principle
    2. Using the properties of Boolean algebra

Absorption Theorem

$$X + XY = X$$

- Prove the theorem
  1. Using the perfect induction principle

X	Y	XY	X + XY
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Absorption Theorem

$$\boldsymbol{X + XY = X}$$

- Prove the theorem
  2. Using the properties of Boolean algebra

$X + XY = X(1 + Y)$ → distributive property

$= X \cdot 1$ → identity # 3: $X + 1 = 1$

$= X$ → identity # 1: $X \cdot 1 = \mathrm{X}$

Consensus Theorem

$$\boldsymbol{XY + \bar{X}Z + YZ = XY + \bar{X}Z}$$

- The third term can be omitted because it is redundant: Y and Z appear in AND with X and complemented X, respectively in the first and second term
- Proof:

$$\begin{aligned} XY + \bar{X}Z + YZ &= XY + \bar{X}Z + YZ(X + \bar{X}) && \rightarrow \text{identity \# 7: } X + \bar{X} = 1 \\ &= XY + \bar{X}Z + XYZ + \bar{X}YZ && \rightarrow \text{distributive property} \\ &= XY + XYZ + \bar{X}Z + \bar{X}YZ && \rightarrow \text{commutative property} \\ &= XY(1 + Z) + \bar{X}Z(1 + Y) && \rightarrow \text{distributive property} \\ &= XY + \bar{X}Z && \rightarrow \text{identity \# 3: } X + 1 = 1 \end{aligned}$$

Complement of a Function

- To get the complement of a function, **'0' and ‘1’ must be swapped in the output column of the truth table**
- There are two alternative ways to obtain the algebraic expression of the complement of a function
  a) Use the De Morgan theorem
  b) Find the dual expression and negate each literal (both variables and constants). In fact, the generalized De Morgan theorem states that the complement of a function can be obtained by exchanging AND with OR and making the complement of each literal

Example: Complement of a Function

- Exercise: find the complement of the following functions in the two ways described in the previous slide

$$F_1 = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z$$
$$F_2 = X(\bar{Y}\bar{Z} + YZ)$$

Example: Complement of a Function

$$F_1 = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z$$

a) Find the complement using De Morgan's theorem

$$\begin{aligned}
\bar{F_1} &= \overline{\bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z} && \rightarrow \text{Complement of } F_1 \\
&= \overline{\bar{X}Y\bar{Z}} \cdot \overline{\bar{X}\bar{Y}Z} && \rightarrow \text{De Morgan's theorem} \\
&= (\bar{\bar{X}} + \bar{Y} + \bar{\bar{Z}}) \cdot (\bar{\bar{X}} + \bar{\bar{Y}} + \bar{Z}) && \rightarrow \text{De Morgan's theorem} \\
&= (X + \bar{Y} + Z) \cdot (X + Y + \bar{Z}) && \rightarrow \text{Identity \# 9: } \bar{\bar{X}} = \mathrm{X}
\end{aligned}$$

Example: Complement of a Function

$$F_2 = X(\bar{Y}\bar{Z} + YZ)$$

a) Find the complement using De Morgan's theorem

$$\begin{aligned}
\overline{F_2} &= \overline{X(\bar{Y}\bar{Z} + YZ)} && \rightarrow \text{Complement of } F_2 \\
&= \bar{X} + (\overline{\bar{Y}\bar{Z} + YZ}) && \rightarrow \text{De Morgan's theorem} \\
&= \bar{X} + \left(\overline{\bar{Y}\bar{Z}}\right) \cdot (\overline{YZ}) && \rightarrow \text{De Morgan's theorem} \\
&= \bar{X} + (Y + Z) \cdot (\bar{Y} + \bar{Z}) && \rightarrow \text{De Morgan's theorem}
\end{aligned}$$

Example: Complement of a Function

$$F_1 = \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z$$

b) Find the dual expression and negate each literal

Dual of F_1: $(\bar{X} + Y + \bar{Z}) \cdot (\bar{X} + \bar{Y} + Z)$

$$\Rightarrow \overline{F_1} = (X + \bar{Y} + Z) \cdot (X + Y + \bar{Z})$$

Pay attention to parentheses: it is better to put them <u>before</u> replacing an expression with the dual!

Example: Complement of a Function

$$F_2 = X(\bar{Y}\bar{Z} + YZ)$$

b) Find the dual expression and negate each literal

Dual of F_2: $X + [(\bar{Y} + \bar{Z}) \cdot (Y + Z)]$

$$\Rightarrow \overline{F_2} = \bar{X} + (Y + Z) \cdot (\bar{Y} + \bar{Z})$$

Summary

- We can realize any logic function with the three fundamental logic gates: AND, OR, NOT
- There are other logic gates (NAND, NOR, XOR, etc.) derived from the fundamental ones. NOR and NAND are universal gates
- Boolean algebra operates on logic functions
  - Duality
  - Basic identity and properties (De Morgan theorem)
- Our aim is to simplify the expressions, in order to make logic circuits as simple as possible (fewer gates, fewer inputs, fewer connections)
- In the next lesson we will learn techniques to minimize logic functions

Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime| Martin