A logo of a university

Description automatically generated

**قــســــــــــم الامــــــــــــــــــــــــــن الــــــــــــــــــــــــــــــــــــسيبرانــــــــــــــــــــــــــــــــــــي**

**Department of Cyber Security**

**Subject:**

**COMPUTER ORGANIZATION & LOGIC DESIGN**

**Class:**

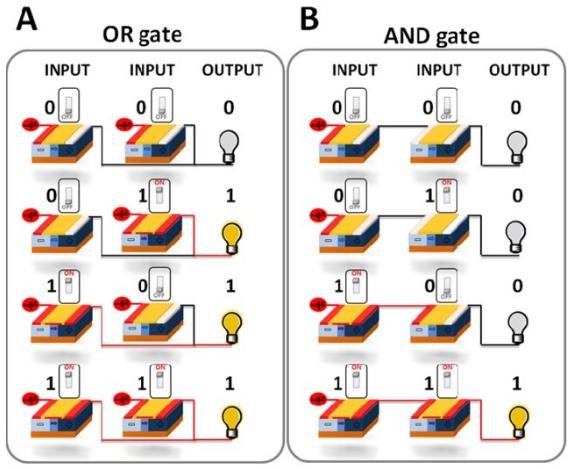
**First**

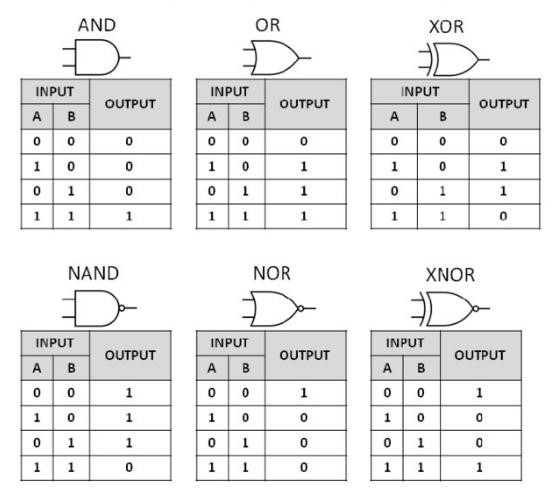
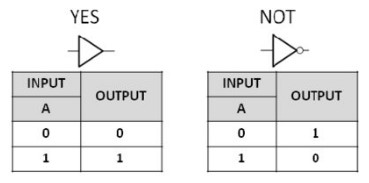
**Lecture: ( 4 )**

**(Numbers System)**

**Lecturer:**

**Msc :Muntather AL-mussawee**

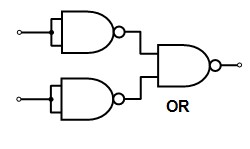
**Logic gates (AND, OR, NOT, NAND, NOR, XOR, XNOR)** 



**Making other logic gates from NAND gates**

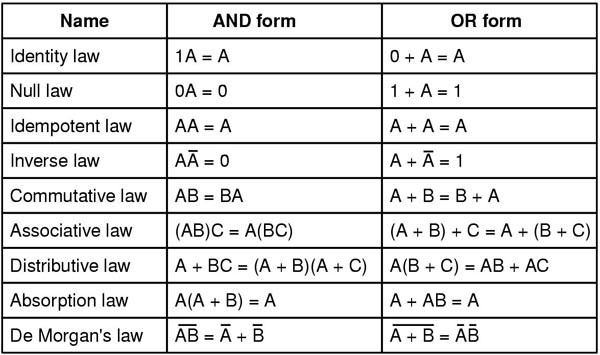
The circuits below show you how to make a NOT, OR, NOR and AND gate using NAND gates.

|  |
| --- |
|  |
|  |

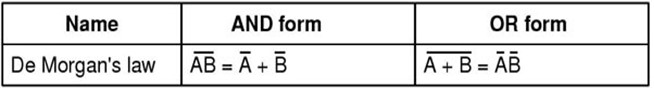


**Logic simplification (Boolean theorem) & (Demorgan’s theorem):**

|  |  |  |
| --- | --- | --- |
|  | **Boolean theorem** |  |
|  |
|  |  |  |



**Demorgan’s theorem**



Boolean Algebra simplification is not that difficult to understand if you realise that the use of the symbols or signs of: “**+**” and “**.**” represent the operation of logical functions.

 Logical functions test whether a **condition** or state is either *TRUE* or *FALSE* but not both at the same time. So depending on the result of that test, a digital circuit can then decide to do one thing or another.

As we saw in the **Laws of Boolean Algebra** tutorial, that Boolean algebra is the mathematics of logic and that the application of various switching theory rules can be applied to simplify long or complex switching algebra notation, and which can also be applied to logic gates and basic digital circuits.

let’s first remind ourselves of a few basic symbols, meanings and laws relating to the three main functions of: **AND**, **OR**, and **NOT**.

#### The Logic AND Operation

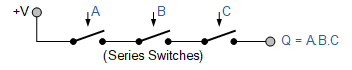
The Boolean expression of A and B is equivalent to A\*B. The AND operator is commonly denoted by a single dot or full stop symbol, ( . ). This gives us the Boolean expression of: A.B, or simple AB.

#### 2-input Logic AND Gate



Then it is clear that the logical AND function is used to compare two or more input conditions and returns TRUE only if all of the conditions occur together. The logical AND operation and Boolean expression of A.B.C can be shown in switching algebra as being:

#### Series (AND) Switching Representation



##### The Logic OR Operation

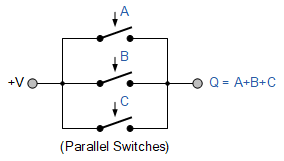
the Boolean expression of A or B is equivalent to A+B. The OR operator is commonly denoted by a plus sign, ( + ) giving us the Boolean expression of: A+B.

##### The 2-input Logic OR Gate



Then it is clear that the logical OR function is used to compare two or more input conditions and returns TRUE only if either one of the conditions occurs. The logical OR operation of addition represents a parallel connection with the order in which the switches are connected in parallel being unimportant as it can extend to any number of parallel-connected switches. So our Boolean expression of A+B+C can be shown in switching algebra as being:

#### Parallel (OR) Switching Representation



**The Logic NOT Operation**

The logical NOT operation is simply an inversion or complementation function of a Boolean value and is not considered as a separate variable. The *NOT function* is so called because its output state is “NOT” the same as its input state,

(hence its name as an inverter).

This means that if switch A is open, A means that the switch is closed. In other words, the Boolean expression for a NOT function is the output is “0” if the input is “1” and the output is “1” if the input is “0”.

#### NOT Representation

