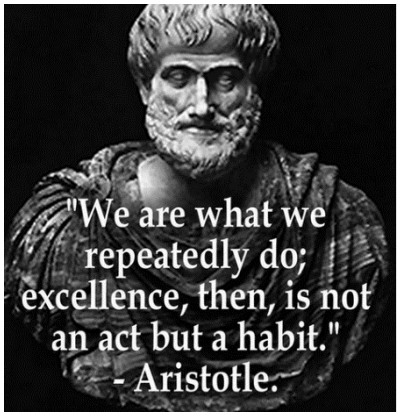
Contents

The First Age of Logic



Originally, logic dealt with arguments in the natural languages used by humans.

For example it would be used to demonstrate the correctness of language.

Thinking Rationally: Laws of Thought

Aristotle (≈450 B.C.) attempted to codify "right thinking"

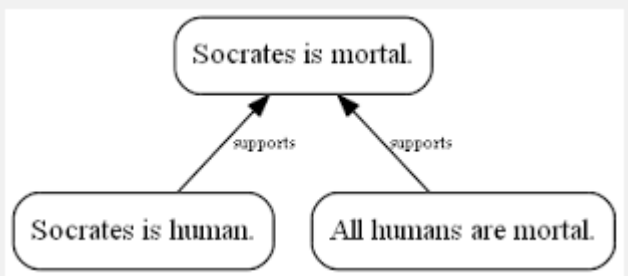
What are the correct argument/thought processes?

E.g.

All men are mortal

Socrates is a man

Therefore, Socrates is mortal



The Law of Thought approach initiated the field called LOGIC.

Historically logic developed within the domains of mathematics and philosophy, but here we will concentrate on the application of logic in computer science.

Recall, the idea of a general purpose computer, the Turing Machine, was invented in the course of research in logic. Computer programs are written in special, symbolic languages, e.g. Fortran, C++, Lisp, Prolog. These languages contain features of logical symbolism, and Lisp and Prolog are derived from formal languages for logic. Through such connections, the study of logic can help one in the design of programs.

Where is Logic applied in Computer Science?

Logic has numerous applications in computer science i.e. processes involving computer control and automated manufacturing machines. Efficient running of these processes require knowledge of the laws/rules of logic. These rules are used in the **design of computer circuits**, the **construction of computer programs**, the verification of the **correctness of programs**, **networks**, **algorithm design** and in many other ways.

The Genius of George Boole

https://bit.ly/2NbsvHH

This is definitely worth a watch, RTÉ's "The Genius of George Boole"

from George Booles 19th Century mathematics achievements to the amazing computing power we have today

1847 – Boolean Algebra

- Boole described an algebraic system of logic, now known as **Boolean algebra.**
- Boole's system was based on binary that consisted the 3 most basic operations:

  AND, **OR**, and **NOT**

- Early thinker into theory of AI

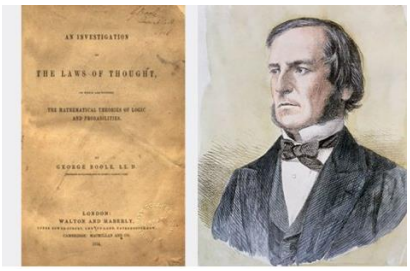
George Boole (1815-1864)

His legacy was **Boolean logic**, a theory of mathematics in which all variables are either on/off

Boolean algebra is ideal foundation for designing the electronic structure of computers, and for manipulating information within computers!!!

It is the basis to the design of digital computer circuits

In 1854, **George Boole** performed an investigation into the "laws of thought" which were based around a simplified version of the "group" or "set" theory, and from this Boolean Algebra (sometimes referred to as ***Boolean Logic*** or just ***Logic***) was developed.



His book was initially largely ignored, besides for his close acquaintances, until after Boole's death ten years later.

QUESTION: The insight of the young Claude Shannon was one of the greatest intellectual breakthroughs of the twentieth century. Who is Claude Shannon and what is his significance in relation to Boolean Logic?

Logic Gates are made by combining transistors. They enable to apply logic to small currents which are either turned on or off and represent **binary information inside a computer**. Computers are made by combining logic gates together.

Logic gates let computers make very simple decisions using the mathematical technique called **Boolean algebra**. These calculations performed by the transistor is an example of a logic function – and that idea is the foundation stone of computer programs: the logical series of instructions that make computers do things.

Boolean algebra deals mainly with the theory that both logic and set operations are either "TRUE" or "FALSE" but not both at the same time. Boolean logic reflects the binary logic of logic gates and transistors in a computer's CPU.

Logic gates

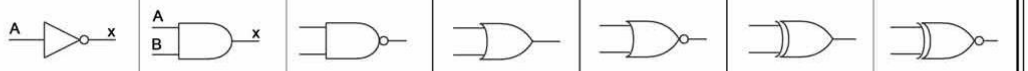
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

Figure 1: Table of logic gates

Many electronic circuits have to make decisions. A Digital Logic Gate is an electronic device that makes logical decisions based on the different combinations of digital signals present on its inputs.

Digital logic gates may have more than one input, (A, B, C, etc.) but generally only have one digital output, (Q).

Individual logic gates can be *connected* together to form combinational or sequential circuits, or larger logic gate functions. **Combining multiple conditions to form one True/False value is, in essence, the domain of Boolean Logic.**

Logic is therefore a simple and effective way of representing the switching action of standard Logic Gates.

Each input and output of the logic gates must be, as always, in one of two states:

1. _____
2. _____

For example, a light with *one* switch can be in one of two possible states:

if there's one input, then there's 2^1 possible outputs

on or ***off***

For example, if two signals are present, complete the following four possible combinations:

1. _0 and 0 _____
2. _____
3. _____
4. _____

However, if there is more than one signal, there are more than two possible states (i.e. 2 inputs=2^2 possible states, 3 inputs=2^3 possible outputs… *n* inputs =2^n possible outputs).

An example of basic logic

Suppose you want to build a light sensor which **only works at night**, you can treat this as giving off a signal that indicates the truth of the statement:

```
P = It is daytime.
```

Clearly `Not(P)` is true when it is night-time and we have our first practical use for Boolean logic!

A logic gate is an elementary building block of a digital circuit.

> They take one or more input values, and produces a single output value BUT these values can only be either true (1) or false(0).

They are a physical device implementing a Boolean function, that is, it performs a logical operation on one or more logic inputs and produces a single logic output

Voltage Levels

At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels.
In microprocessor chips (IC's), there is a pre-defined voltage range for the input and output voltage levels which define exactly what a logic **1** level is and what is a logic **0** level.

> The exact switching voltage required to produce either a logic "0" or a logic "1" depends upon the specific logic group or family, for example:
> Logic high, or 1 might be represented as voltage input between 2.0v and 5v
> Logic low, or 0, might be represented as any voltage input below 0.8v

Logic gate symbols, Boolean algebraic expression and Truth Tables

There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

AND OR and NOT are the building blocks for just about everything else we can do in Boolean Algebra.
XOR NAND NOR and XNOR what are called **derived operations**. These are essentially shortcuts for commonly used combinations of the basic operations

The behaviour of each logic gate and circuits can be described by:

1. its **boolean algebraic expressions** e.g **$\bar{A}$** is the expression for **not A**

2. its **symbol** 

   Each gate is represented by a specific symbol

3. its **truth table**

A	X
0	1
1	0

   Defines the function of a gate by listing all of its possible input combinations and the corresponding output.

These gates are each fairly simple but all digital electronics are composed of just these gates in various arrangements!

1. AND gate description, Boolean expression and truth table

The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator.

e.g. “If switch 1 is on AND switch 2 is on, turn on the light”

→ if A=ON **AND** B=ON, Y=lights

Boolean expression: **Y=A.B** (where the dot represents logical AND)

alternative expression **Y=AB**

The following illustration and truth table show the circuit symbol and logic combinations for an AND gate.
(In the symbol, the input terminals are at left and the output terminal is at right.)
The output is "true" when both inputs are "true." Otherwise, the output is "false."

AND gate

Circuit Symbol

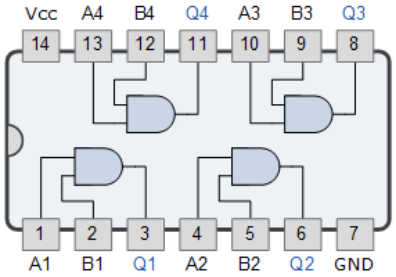
Truth Table

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Truth tables are used to help **show the function** of a logic gate
A logic gate has a truth table for each possible combination of inputs
For n inputs there will be 2^n outputs so two inputs: 2^2=4 outputs

The AND gate is an electronic circuit that gives a **high** output (1) only if all its inputs are high.

A dot (.) is used to show the AND operation i.e. A.B
Alternatively, it can be represented without the dot i.e. AB



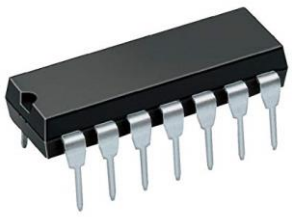


Figure 2: 7408 Quadruple 2-input AND Gate Schematic diagram

2. OR gate description, Boolean expression and truth table

The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or"

The output is "true" if either or both of the inputs are "true."
If both inputs are "false," then the output is "false."
→Turn on the headlights if it is dark ***or*** if it is raining

A=dark B=raining Y=output

Y=A+B

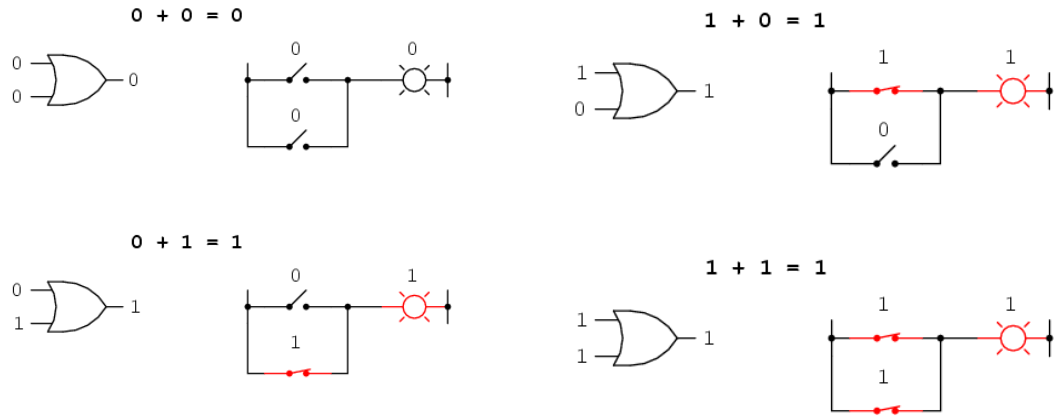


Figure 3: Analogue switches used as Multiplexers

OR gate

2 input OR gate		
A	**B**	**A+B**
(dark)	**(rain)**	**(lightsOn)**
0	**0**	
0	**1**	
1	**0**	
1	**1**	

Exercise 1. Complete the output on the truth table above for the 2 input OR gate

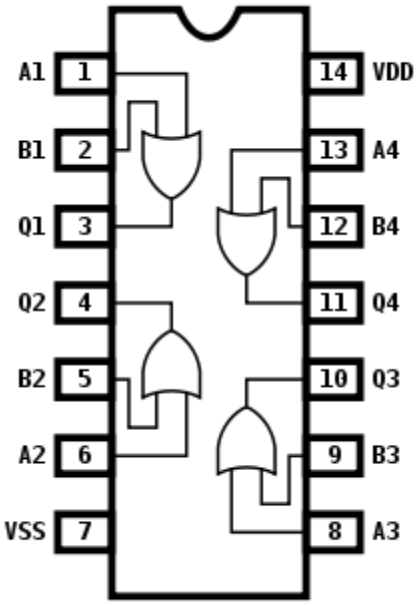


Figure 4: OR Gate Schematic Diagram

The schematic diagram shows the arrangement of four OR gates within a standard 4071 CMOS **integrated circuit**.

OR gates are equivalent to Boolean **addition**	**A+B=X**
AND gates are equivalent to Boolean **multiplication**	**A.B=X**
(return to the truth table to confirm this statement)	

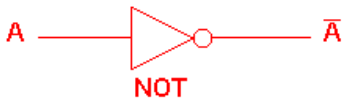
3. NOT gate description, Boolean expression and truth table

The NOT gate is an electronic circuit that (has only one input) produces an inverted version of the input at its output. It is also known as an ***inverter***.

NOT gates could be used to turn the heating on if it is ***not*** hot:

A=hot, **Y**=heating on → **$Y=\bar{A}$**

NOT gate



NOT gate	
A	**$\bar{A}$ (=A')**
0	
1	

If the input variable is A, the inverted output is known as NOT A.
NOT A is represented as the Boolean expression **$\bar{A}$** or **A'**

Exercise 2. Complete the output on the truth table above for the NOT gate

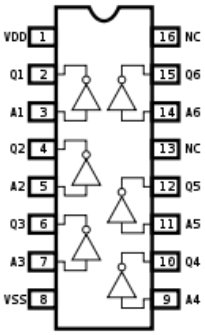


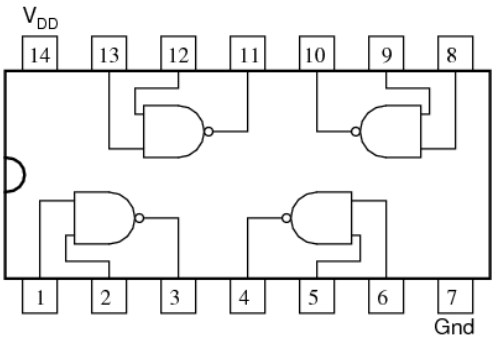
Figure 5: NOT gates within 4049 CMOS hex inverting buffer

4. NAND gate description, Boolean expression and truth table

The ***NAND*** gate (***Not AND*** gate) operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

NAND gate

"Pinout," or "connection" diagram for the 4011 quad NAND gate



The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

In a NAND gate the output is only 0 if all the inputs are 1.

Exercise 3. Complete the full truth table for the 2-input NAND gate $\mathbf{X=\overline{AB}}$

5. NOR gate description, Boolean expression and truth table

The *NOR gate* (*Not OR* gate) is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

The outputs of all NOR gates are low if ***any*** of the inputs are high (the opposite of an OR gate)

Exercise 4. Complete the full truth table for the 2-input NOR gate $\mathbf{X=\overline{A+B}}$

Universal Gates NAND and NOR

Being a *universal gate* means that any other gate, and therefore any digital machine, can be made of just NAND gates or just NOR gates.

This therefore implies that ***all*** logic circuits can be constructed using either of these gates.

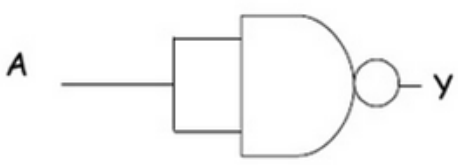
This has some significant benefits for manufacturing electronics. NAND gates are preferred over NOR gates in industry as they offer a lesser delay when compared to NOR gates

(Nmos transistors used in NAND gates allow double the current per channel area compared to Pmos transistors using NOR gates)

For Example:

If we tie the inputs of a NAND gate together, then we limit the possible input combinations to two: 1 1 and 0 0.

Exercise 5. Complete the truth table output to demonstrate the output of this NAND

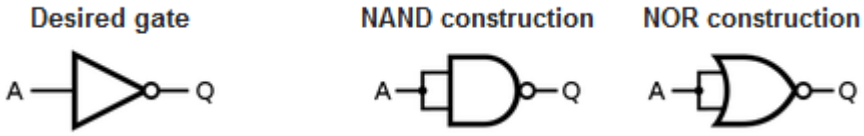


A	*B*	*Y*
0	0	
1	1	

Q: Why might a universal gate be used instead of a NOT gate?

Due to the fact that current flows through the resistor in one of the two states, the resistive-drain configuration is disadvantaged for power consumption and processing speed.

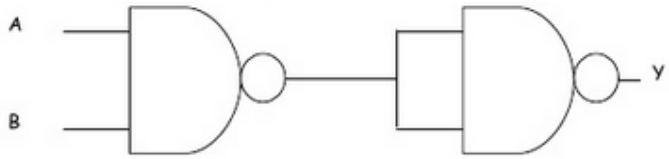
Alternatively, inverters can be constructed using two complementary transistors:



Exercise 6. Complete the truth table output to demonstrate the output of this NOR.

Exercise 7. With the use of a truth table, demonstrate what gate the following use of two NAND gates correspond to?

ANSWER: _____ gate



Exercise 8. Prove that, similar to exercise directly above, this is also the case with a NOR gate

Another Example:

If you tie the I/P of A for a NAND gate, the O/P is notA
If one I/P is A and the other I/P is 1, the O/P is notA

Exercise 9. What is the corresponding function when you invert the inputs and then NAND them?

A	B	$\bar{A}$	$\overline{B}$	$\overline{A}.\overline{B}$	$\overline{\overline{A}.\overline{B}}$
0	0				
0	1				
1	0				
1	1				

Recall:A+B

6. exclusive-OR gate description, Boolean expression and truth table

The ***XOR*** (*exclusive-OR, sometimes represented as* ***EXOR***) *gate* acts in the same way as the logical "either/or."

The output is "true" if either, but not both, of the inputs are "true." (≡a high output if **either, but not both**, of its two inputs are high i.e. an odd no. of inputs are true)

The output is "false" if both inputs are "false" or if both inputs are "true."

Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

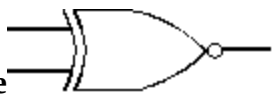
≡XOR gate

A	B	A⊕B
	1	
1		
1	1	

Exercise 10. Complete the above **A⊕B** truth table

7. exclusive-NOR gate description, Boolean expression and truth table

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.

XNOR gate 

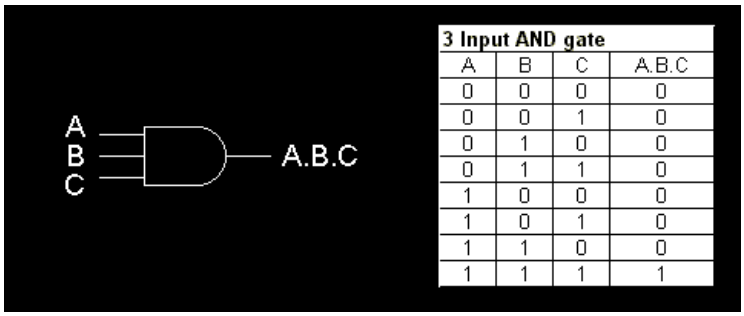
A	B	$\overline{A \oplus B}$

Exercise 11. Complete the truth table of the above $\overline{A \oplus B}$ logical expression

Multiple Input Gates

Three Input AND Gate

Here is an example of a three input AND gate. Notice that the truth table for the three input gate is similar to the truth table for the two input gate. It works on the same principle, this time all three inputs need to be high (1) to get a high output.

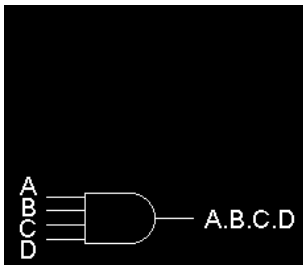


3 Input AND gate			
A	B	C	A.B.C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

RECALL an AND gate principle is that the output is "true" or “1” when both/all inputs are "true."

Four Input AND Gate

Here is an example of a four input AND gate. It also works on the same principle, all four inputs need to be high (1) to get a high output



4 Input AND gate				
A	B	C	D	A.B.C.D
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The same principles apply to 5, 6,..., n input gates.

In reality, the maximum number of gate inputs is limited due to electrical constraints. It depends on the technology being used.

For a high number of inputs, gates are typically composed of several smaller gates

Multi input gates can be made by joining gates of the same type with less inputs

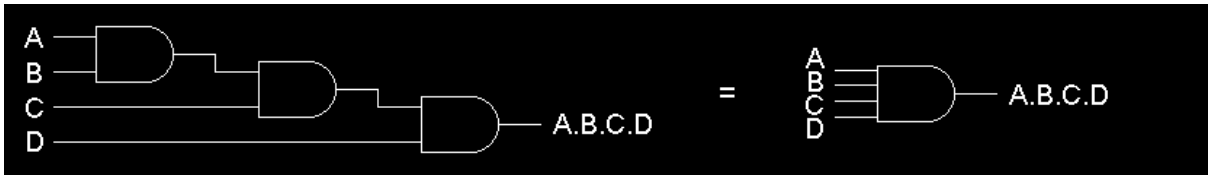
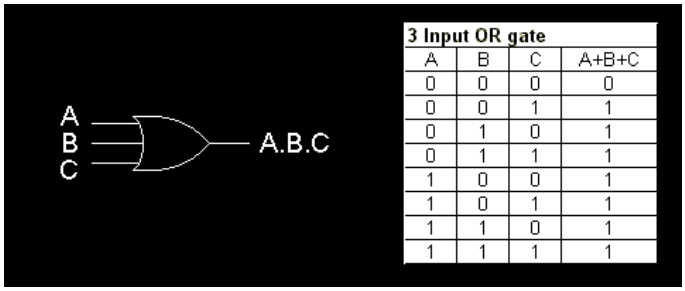


Figure 6: multiple 2 input AND gates used to make 4 input AND gate

Three Input OR Gate

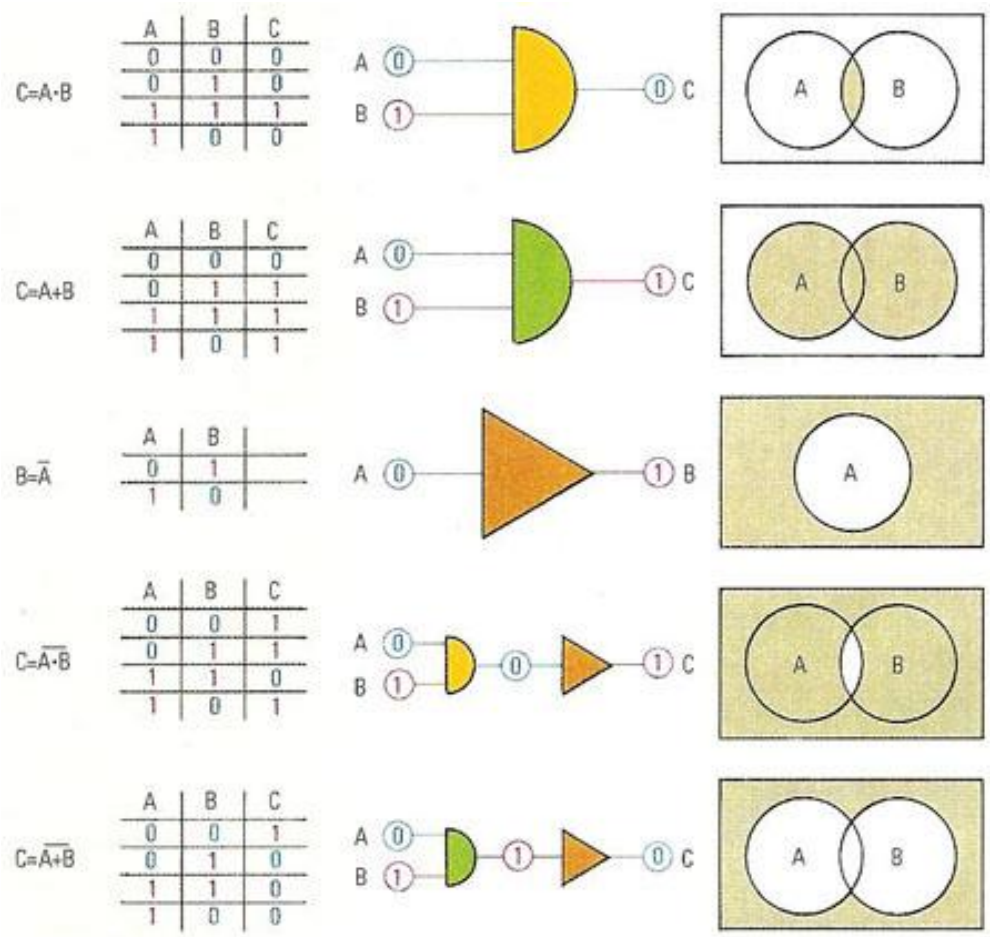


3 Input OR gate			
A	B	C	A+B+C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

RECALL an OR gate principle is that if both/all inputs are "false" or “0” then the output is "false", otherwise true outputs

Logic Gates Recap:

AND Gate	F=A.B	enables/disables device
OR Gate	F=A+B	car door alarm system
NOT Gate	$F=\overline{A}$	1’s complement
Universal NAND gate	$F=\overline{A \cdot B}$	
Universal NOR gate	$F=\overline{A + B}$	
XOR gate	$F= A \oplus B$	
XNOR gate	$F= \overline{A \oplus B}$	


C=A·B
A B C
0 0 0
0 1 0
1 1 1
1 0 0
A 0
B 1
0 C
A B
C=A+B
A B C
0 0 0
0 1 1
1 1 1
1 0 1
A 0
B 1
1 C
A B
B=Ā
A B
0 1
1 0
A 0
1 B
A
C=Ā·B̄
A B C
0 0 1
0 1 1
1 1 0
1 0 1
A 0
B 1
0
1 C
A B
C=Ā+B̄
A B C
0 0 1
0 1 0
1 1 0
1 0 0
A 0
B 1
1
0 C
A B