Chapter3:Logic gates

Variable, complement, and literal are terms used in Boolean algebra. Boolean algebra is the **mathematic** of digital systems.

variable is a symbol used to represent a logical quantity. Any single variable can have a 1 or a 0 value.

The complement is the inverse of a variable and is indicated by a bar over variable (overbar). For example, the complement of the variable A is A. If A = 1, then A = 0. If A = 0, then A = 1. The complement of the variable A is read as "not A" or "A bar." Sometimes a prime symbol rather than an overbar is used to denote the complement of a variable; for example, B' indicates the complement of B.

A literal is a variable or the complement of a variable

In electronics, a ***logic gate*** **is an idealized or physical device implementing a Boolean function; that is, it performs a logical operation on one or more logical inputs, and produces a single logical output**. Depending on the context, the term may refer to an **ideal logic gate**.

Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements. With amplification, logic gates can be cascaded in the same way that Boolean functions can be composed, allowing the construction of a physical model of all of Boolean logic, and therefore, all of the algorithms and mathematics that can be described with Boolean logic.

Logic circuits include such devices as multiplexers, registers, arithmetic logic units (ALUs), and computer memory, all the way up through complete microprocessors, which may contain more than 100 million gates. **In practice, the gates are made from field-effect transistors (FETs), particularly MOSFETs (metal–oxide–semiconductor field-effect transistors).**

Compound logic gates AND-OR-Invert (AOI) and OR-AND-Invert (OAI) are often employed in circuit design because their construction using MOSFETs is simpler and more efficient than the sum of the individual gates.

To build a functionally complete logic system, relays, valves (vacuum tubes), or transistors can be used. The simplest family of logic gates using bipolar transistors is called ***resistor-transistor logic*** (RTL). Unlike simple diode logic gates (which do not have a gain element), RTL gates can be cascaded indefinitely to produce more complex logic functions. RTL gates were used in early integrated circuits. For higher speed and better density, the resistors used in RTL were replaced by diodes resulting in ***diode-transistor logic*** (DTL).

Transistor-transistor logic (TTL) then supplanted DTL. As integrated circuits became more complex, bipolar transistors were replaced with smaller ***field-effect transistors*** (MOSFETs). To reduce power consumption still further, most ***contemporary*** chip implementations of digital systems now use CMOS logic. CMOS uses complementary (both n-channel and p-channel) MOSFET devices to achieve a high speed with low power dissipation.

For small-scale logic, designers now use prefabricated logic gates from families of devices such as the TTL 7400 series by Texas Instruments, the CMOS 4000 series by RCA, and their more recent descendants. Increasingly, these fixed-function logic gates are being replaced by programmable logic devices, which allow designers to pack a large number of mixed logic gates into a single integrated circuit. The field-programmable nature of programmable logic devices such as

FPGAs has removed the 'hard' property of hardware; it is now possible to change the logic design of a hardware system by reprogramming some of its components, thus allowing the features or function of a hardware implementation of a logic system to be changed.

Electronic logic gates differ significantly from their relay-and-switch equivalents. They are much faster, consume much less power, and are much smaller (all by a factor of a million or more in most cases).

Also, there is a fundamental structural difference. The switch circuit creates a continuous metallic path for current to flow (in either direction) between its input and its output. The semiconductor logic gate, on the other hand, acts as a high-gain voltage amplifier, which sinks a tiny current at its input and produces a low-impedance voltage at its output. It is not possible for current to flow between the output and the input of a semiconductor logic gate.

Another important advantage of standardized integrated circuit logic families, such as the 7400 and 4000 families, is that they can be cascaded. This means that the output of one gate can be wired to the inputs of one or several other gates, and so on. Systems with varying degrees of complexity can be built without great concern of the designer for the internal workings of the gates, provided the limitations of each integrated circuit are considered.

The output of one gate can only drive a finite number of inputs to other gates, a number called the '*fanout limit*'.

Also, there is always a delay, called the '*propagation delay*', from a change in input of a gate to the corresponding change in its output.

When gates are cascaded, the total propagation delay is approximately the sum of the individual delays, an effect which can become a problem in high-speed circuits. Additional delay can be caused when a large number of inputs are connected to an output, due to the ***distributed capacitance*** of all the inputs and wiring and the finite amount of current that each output can provide.

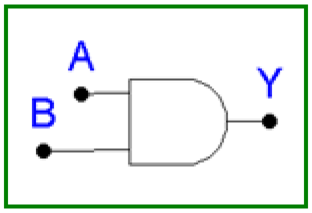
Logic gates can also be used to store data. A storage element can be constructed by connecting several gates in a "***latch***" circuit. More complicated designs that use ***clock signals*** and that change only on a rising or falling edge of the clock are called edge-triggered "***flip-flops***".

The combination of multiple flip-flops in parallel, to store a multiple-bit value, is known as a register. When using any of these gate setups the overall system has memory; it is then called a ***sequential logic*** system since its output can be influenced by its previous state(s).

These logic circuits are known as computer *memory*. They vary in performance, based on factors of speed, complexity, and reliability of storage, and many different types of designs are used based on the application.

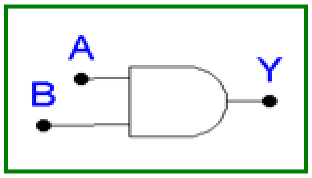
The AND Gate

- “All or Nothing Gate”
- Boolean Expression: A · B = Y



An AND gate produces A HIGH output *only w*hen all of the inputs are HIGH

Truth Table - AND Gate



B	A	Y
0	0	0
0	1	0
1	0	0
1	1	1

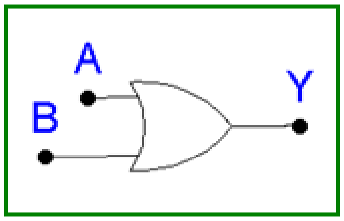
.

T
h
e

O
R

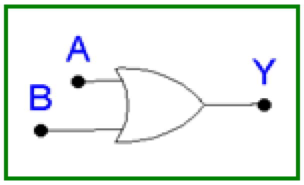
G
a
t
e

- "Any or All Gate"
- Boolean Expression: $A + B = Y$



An OR gate produces a HIGH on the output when *any* of the inputs are HIGH

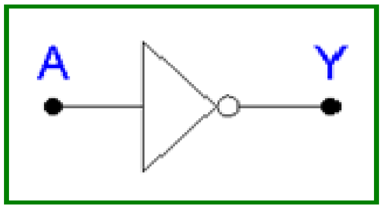
Truth Table - OR Gate



B	A	Y
0	0	0
0	1	1
1	0	1
1	1	1

The Inverter

- NOT Circuit
- Gives output that is not the same as the input.
- Boolean Expression: $Y = \overline{A}$ or $Y = A'$
- Double inverting: $\overline{\overline{A}} = A$
- NOT gate inverts, or complements, or negates

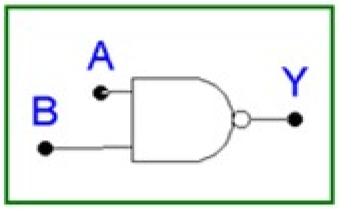


The Inverter (NOT circuit) performs the operation called *inversion or complementation* .

The inverter changes one logic level to the opposite level ,in terms of bits , it changes a 1to a 0 and a 0 to 1

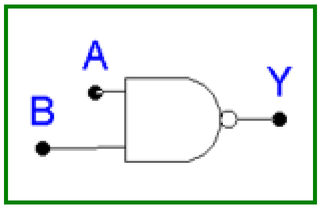
The NAND Gate

- NOT AND or inverted AND function.
- Boolean Expression: $\overline{A \cdot B} = Y$ or $(A \cdot B)' = Y$



The NAND Gate is popular logic element because it can be used as *a universal gate*: that is ,NAND gates can be used in combination to perform the AND,OR, and inverter operations.

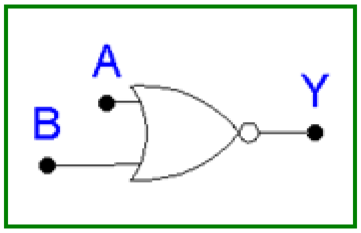
Truth Table - NAND Gate



B	A	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

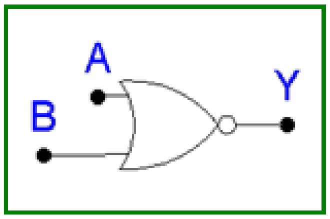
The NOR Gate

- NOT OR or Inverted OR
- Boolean Expression: $\overline{A + B} = Y$
  or $(A + B)' = Y$



The NOR gate, like the NAND gate, is a useful logic element because it can also be used as a universal gate.

Truth Table - NOR Gate

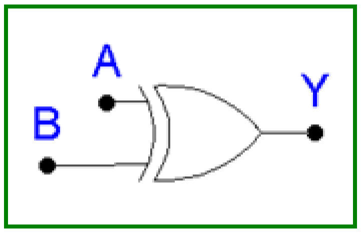


B	A	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

,

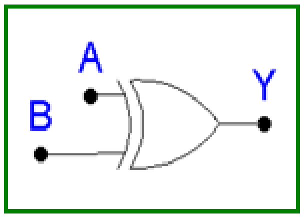
The XOR Gate

- Known as “Exclusive OR” Gate
- “Anything but not all” Gate
- Boolean Expression: $A \oplus B = Y$



Exclusive -OR and exclusive -NOR gates are formed by a combination of other gates , because of their importance in many applications , these gates are often treated as basic logic elements with their own unique symbols

Truth Table - XOR Gate

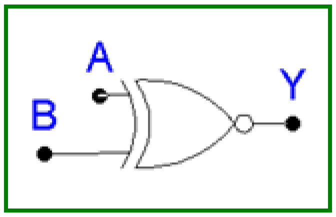


B	A	OR	XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

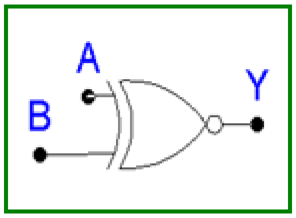
The XNOR Gate

- Known as the Exclusive NOR Gate
- The Inverted XOR
- Boolean Expression: $\overline{A \oplus B} = Y$

  or $(A \oplus B)' = Y$



Truth Table - XNOR Gate



B	A	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1