Logic gate

From Wikipedia, the free encyclopedia

A **logic gate** is an arrangement of controlled switches used to calculate operations using Boolean logic in digital circuits. They are primarily implemented electronically but can also be constructed using electromagnetic relays, electronic diodes, fluidics, optical or even mechanical elements.

Contents

Basic logic gates and mechanical equivalents

While semiconductor electronic logic (see later) is preferred in most applications, relays and switches are still used in some industrial applications and for educational purposes. In this article, the various types of logic gates are illustrated with drawings of their relay-and-switch implementations, although the reader should remember that these are electrically different from the semiconductor equivalents that are discussed later.

Relay logic was historically important in industrial automation (see ladder logic and programmable logic controller). Since relay contacts conduct in both directions, complex logic designs must be checked for "sneak paths" that produce unintended logic paths.

Semiconductor logic gates are *not* conductive in both directions, as the input signal acts as a 'trigger' to allow current out of the output, rather than allowing current straight through from input to output. However, the following mechanical variations do show the basic principles of the gates without detailing the precise internal workings. For information about how modern semiconductors really work, see CMOS.

The three types of essential logic gate are the AND, the OR and the NOT gate. With these three, any conceivable

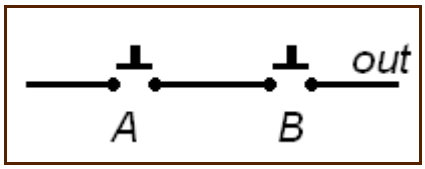
boolean equation can be implemented. However, for convenience, the derived types NAND, NOR, XOR and XNOR are also used, which often use fewer circuit elements for a given equation than an implementation based solely on AND, OR and NOT would do. In fact the NAND has the lowest component count of any gate apart from NOT when implemented using modern semiconductor techniques, and since a NAND can implement both a NOT and, by application of De Morgan's Law, an OR function, this single type can effectively replace AND, OR and NOT, making it the only type of gate that is needed in a real system. Programmable logic arrays will very often contain nothing but NAND gates to simplify their internal design.

[edit]

AND gates

The first example is the AND gate, whose truth table is shown below, left. The Boolean AND function can be implemented with two switches, A and B, as shown below, right. If a power lead is connected to switch A, and a wire connects switches A and B, then both A and B have to be "on" in order for the output of the circuit to conduct electricity and provide power.

INPUT		OUTPUT
A	B	A AND B
0	0	0
1	0	0
0	1	0
1	1	1



Switch circuit diagram for AND gate
[edit]

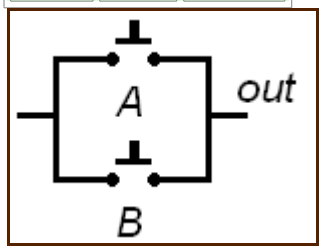
OR gates

Another important arrangement is the OR gate, whose truth table is shown below, left.

An OR gate can be constructed from two switches, arranged so that if either switch is "on", the output will also be "on". Note that the output will still be on even if both switches are on.

INPUT		OUTPUT
A	B	A OR B
0	0	0
1	0	1

0	1	1
1	1	1



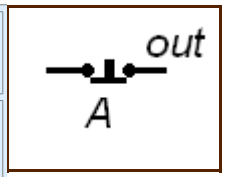
Switch circuit diagram for OR gate
[edit]

NOT gates

A simpler arrangement is the NOT gate, whose truth table is shown opposite.

This is a special switch that when pushed *breaks* the current when it is pressed. The normally-closed contact of a relay can be used for this purpose.

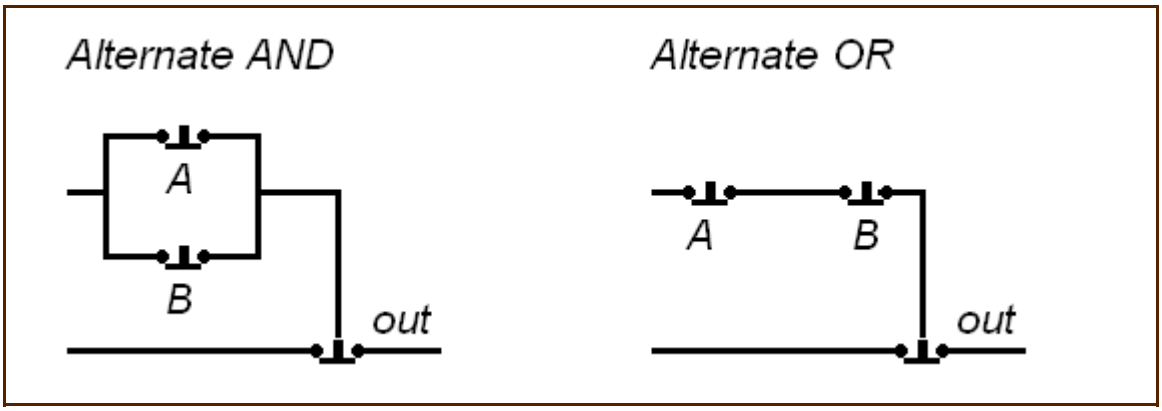
INPUT	OUTPUT
A	NOT A
0	1
1	0



Switch circuit diagram for NOT gate
[edit]

NAND and NOR gates

Using NOT gates, also called inverters, allows us to make alternate versions of the AND and OR gates, by virtue of De Morgan's Law. Note that the layout of the switches in the two circuits is swapped when we turn the switches "backwards". Also note how the output of the first pair controls the operation of the NOT gate.

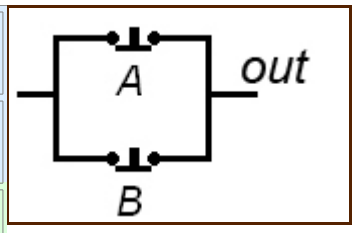


Switch diagram of alternate AND and OR gates

This may seem like an unnecessary complication, but in fact this is very useful. By removing the NOT gate from these alternate circuits, we create the so-called NAND (for NOT-AND) and NOR (for NOT-OR) gates.

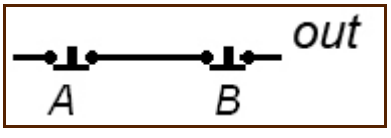
NAND

INPUT		OUTPUT
A	B	A NAND B
0	0	1
1	0	1
0	1	1
1	1	0



Switch circuit diagram of NAND gate
[edit]

NOR



INPUT		OUTPUT
A	B	A NOR B
0	0	1
1	0	0
0	1	0
1	1	0

Switch circuit diagram of NOR gate
[edit]

XOR and XNOR gates

There exist two other 'basic' logic gates - XOR (exclusive-OR)and XNOR (exclusive-NOR).

XOR is a 'stricter' version of the OR gate. Rather than allowing the output to be HIGH when either one or both of the inputs are HIGH, an XOR gate has a HIGH output only when only *one* input is HIGH. As such it has the truth table shown to the right. This can also be interpreted (for a two-input gate) as "HIGH output when the inputs are different".

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR is an inverted version of the XOR gate. As such it has the truth table shown to the right. This can also be interpreted as "HIGH output when the inputs are same".

INPUT		OUTPUT
A	B	A XNOR B
0	0	1
1	0	0
0	1	0
1	1	1

The preceding simple logic gates can be combined to form more complicated boolean logic circuits. Logic circuits are often classified in two groups: combinatorial logic, in which the outputs are continuous-time functions of the inputs, and sequential logic, in which the outputs depend on information stored by the circuits as well as on the inputs.

[edit]

Background

The simplest form of electronic logic is diode logic. This allows AND and OR gates to be built, but not inverters, and so is an incomplete form of logic. To build a complete logic system, valves or transistors can be used. The simplest family of logic gates using bipolar transistors is called resistor-transistor logic, or RTL. Unlike diode logic gates, RTL gates can be cascaded indefinitely to produce more complex logic functions. These gates were used in early integrated circuits. For higher speed, the resistors used in RTL were replaced by diodes, leading to diode-transistor logic, or DTL. It was then discovered that one transistor could do the job of two diodes in the space of one diode, so transistor-transistor logic, or TTL, was created. In some types of chip, to reduce size and power consumption still further, the bipolar transistors were replaced with complementary field-effect transistors (MOSFETs), resulting in complementary metal-oxide-semiconductor (CMOS) logic.

For small-scale logic, designers now use prefabricated logic gates from families of devices such as the TTL 7400 series invented by Texas Instruments and the CMOS 4000 series invented by RCA, and their more recent descendants. These devices usually contain transistors with multiple emitters, used to implement the AND function, which are not available as separate components. Increasingly, these fixed-function logic gates are being replaced by programmable logic devices, which allow designers to pack a huge 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 standardised semiconductor logic gates, such as the 7400 and 4000 families, is that they are cascadable. This means that the output of one gate can be wired to the inputs of one or several other gates, and so on *ad infinitum*, enabling the construction of circuits of arbitrary complexity without requiring the designer to understand the internal workings of the gates.

In practice, the output of one gate can only drive a finite number of inputs to other gates, a number called the 'fanout limit', but this limit is rarely reached in the newer CMOS logic circuits, as compared to TTL circuits. 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.

[edit]

Logic levels

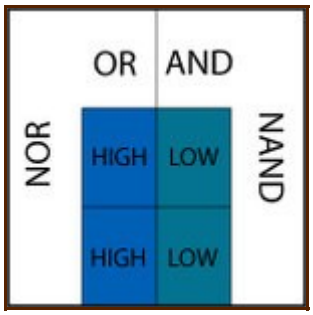
The two logic levels in binary logic circuits can be described as two voltage ranges, "zero" and "one", or "high" and "low". Each technology has its own requirements for the voltages used to represent the two logic levels, to ensure that the output of any device can reliably drive the input of the next device. Usually, two non-overlapping voltage ranges, one for each level, are defined. The difference between the high and low levels ranges from 0.7 volts in ECL logic to around 28 volts in relay logic.

[edit]

Logic gates and hardware

NAND and NOR logic gates are the two pillars of logic, in that all other types of Boolean logic gates (i.e., AND, OR, NOT, XOR, XNOR) can be created from a suitable network of just NAND or just NOR gate(s). They can be built from relays or transistors, or any other technology that can create an inverter and a two-input AND or OR gate.

These functions can be seen in the table below. As you look at this table, start with the name of a gate and read across the table while you apply the following statement:



Basic Logic Functions

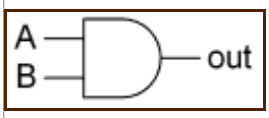
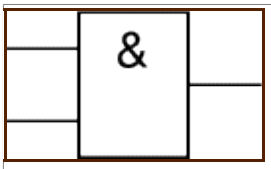
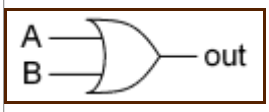
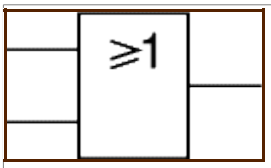
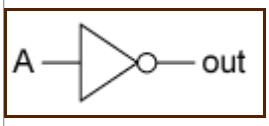
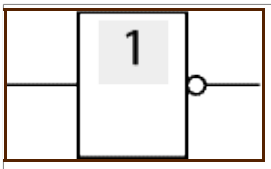
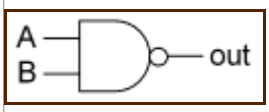
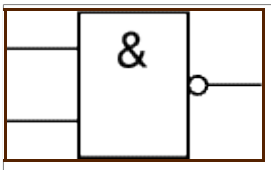
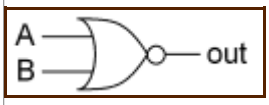
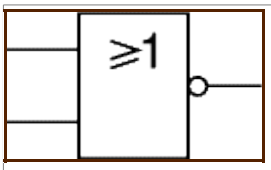
OR: Any high in will create a high output
NOR: Any high in will create a low output
AND: Any low in will create a low output
NAND: Any low in will create a high output

Logic gates are a vital part of many digital circuits, and as such, every kind is available as an IC. For examples, see the 4000 series of CMOS logic chips.

[edit]

Symbols

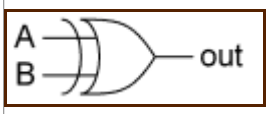
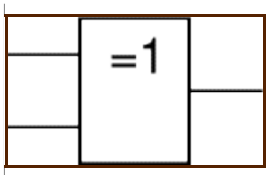
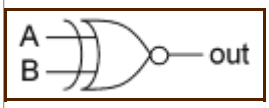
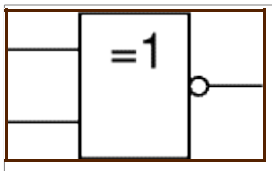
There are two sets of symbols in common use, both now defined by ANSI/IEEE Std 91-1984 and its supplement ANSI/IEEE Std 91a-1991. The "distinctive shape" set, based on traditional schematics, is used for simple drawings and is quicker to draw by hand. It is sometimes unofficially described as "military", reflecting its origin if not its modern usage. The "rectangular shape" set, based on IEC 60617-12, has rectangular outlines for all types of gate, and allows representation of a much wider range of devices than is possible with the traditional symbols. The IEC's system has been adopted by other standards, such as EN 60617-12:1999 in Europe and BS EN 60617-12:1999 in the United Kingdom.

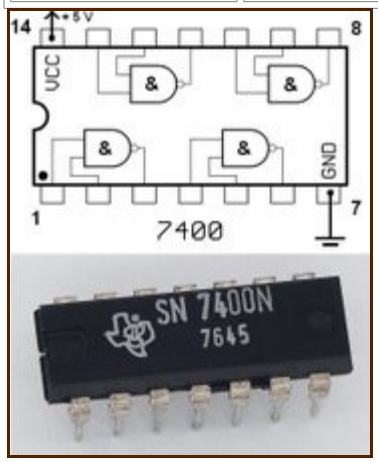
Type	Distinctive shape	Rectangular shape
AND		
OR		
NOT		
In electronics a NOT gate is more commonly called an inverter. The circle on the symbol is called a *bubble*, and is generally used in circuit diagrams to indicate an inverted input or output.		
NAND		
NOR		

In practice, the cheapest gate to manufacture is usually the NAND gate. Additionally, Charles Peirce showed that NAND gates alone (as well as NOR gates alone) can be used to reproduce all the other logic gates.

Symbolically, a NAND gate can also be shown using the OR shape with bubbles on its inputs, and a NOR gate can be shown as an AND gate with bubbles on its inputs. This reflects the equivalency due to De Morgans law, but it also allows a diagram to be read more easily, or a circuit to be mapped onto available physical gates in packages easily, since any circuit node that has bubbles at both ends can be replaced by a simple bubble-less connection and a suitable change of gate. If the NAND is drawn as OR with input bubbles, and a NOR as AND with input bubbles, this gate substitution occurs automatically in the diagram (effectively, bubbles "cancel"). This is commonly seen in real logic diagrams - thus the reader must not get into the habit of associating the shapes exclusively as OR or AND shapes, but also take into account the bubbles at both inputs and outputs in order to determine the "true" logic function indicated.

Two more gates are the exclusive-OR or XOR function and its inverse, exclusive-NOR or XNOR. The two input Exclusive-OR is true only when the two input values are *different*, false if they are equal, regardless of the value. If there are more than two inputs, the gate generates a true at its output if the number of trues's at its input is *odd* ([1]). In practice, these gates are built from combinations of simpler logic gates.

XOR		
XNOR		



The 7400 chip, containing four NAND's. The two additional contacts supply power (+5 V) and connect the ground. It was a very popular chip, widely used by amateurs

[edit]

Storage of bits

Related to the concept of logic gates (and also built from them) is the idea of storing a bit of information. The gates discussed up to here cannot store a value: when the inputs change, the outputs immediately react. It is possible to make a storage element either through a capacitor (which stores charge due to its physical properties) or by feedback. Connecting the output of a gate to the input causes it to be put through the logic again, and choosing the feedback correctly allows it to be preserved or modified through the use of other inputs. A set of gates arranged in this fashion is known as a "latch", and more complicated designs that utilise clocks (signals that oscillate with a known period) and change only on the rising edge 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.

These registers or capacitor-based 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.

[edit]

Three-state logic gates

Three-state, or 3-state, logic gates are a form of electronic logic gate in which the output has three possible states: high (H), low (L) and high-impedance (Z). The Z state exists merely to help the circuit designer and, unlike the H and L states, carries no information. This feature is used in circuits that have two or more logic outputs connected to a single logic input. A control circuit enables one output at a time depending on the logic function that is required, the other outputs being held in the Z state (also called 'disabled').

'Tri-state', a widely-used synonym of 'three-state', is a trademark of the National Semiconductor Corporation.

[edit]

Miscellaneous

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

In reversible logic, Toffoli gates are used.

[edit]

History and development

The earliest logic gates were made mechanically. Charles Babbage, around 1837, devised the Analytical Engine. His logic gates relied on mechanical gearing to perform operations. Electromagnetic relays were later used for logic gates. In 1891, Almon Strowger patented a device containing a logic gate switch circuit (U.S. Patent 447918). Strowger's patent was not in widespread use until the 1920s. Starting in 1898, Nikola Tesla filed for patents of devices containing logic gate circuits (see List of Tesla patents). Eventually, vacuum tubes replaced relays for logic operations. Lee De Forest's modification, in 1907, of the Fleming valve can be used as AND logic gate. Claude E. Shannon introduced the use of Boolean algebra in the analysis and design of switching circuits in 1937. Walther Bothe, inventor of the coincidence circuit, got part of the 1954 Nobel prize in physics, for the first modern electronic AND gate in 1924.

[edit]

See also

- Digital circuit
- Digital logic
- Fanout
- Karnaugh map
- List of Boolean algebra topics
- Families of logic devices
- NMOS
- Race hazard
- Venn diagram

[edit]

External links and references

- *Symbols for logic gates*. Twenty First Century Books, Breckenridge, CO.
- *Tesla's invention of the AND logic gate*. Twenty First Century Books, Breckenridge, CO.
- *Wireless Remote Control and the Electronic Computer Logic Gate*. Twenty First Century Books, Breckenridge, CO.
- Anderson, Leland I., "*Nikola Tesla — Guided Weapons & Computer Technology*". ISBN 0-9636012-5-3
- Bigelow, Ken, "*How logic gates work internally (for several logic families)*", play-hookey.com.

- C. E. Shannon, "*A symbolic analysis of relay and switching circuits*," Transactions American Institute of Electrical Engineers, vol. 57, pp. 713-723, March 1938.
- The IEC symbols are defined in IEC 60617-12 (1997-12), *Graphical symbols for diagrams - Part 12: Binary logic elements*
- "*LEGO Logic Gates*". goldfish.org.uk, 2005.

[edit]

Further reading

- Bostock, Geoff, "*Programmable logic devices : technology and applications*". New York, McGraw-Hill, c1988. ISBN 0070066116
- Brown, Stephen D. et. al., "*Field-programmable gate arrays*". Boston, Kluwer Academic Publishers, c1992. The Kluwer international series in engineering and computer science. ISBN 0792392485
- Awschalom, D., D. Loss, and N. Samarth, "*Semiconductor spintronics and quantum computation*". Berlin, Springer, c2002. ISBN 3540421769

Category: Digital electronics