DIGITAL ELECTRONICS

Ed Clarvis 2018

PART ONE

Logic Gates

In this booklet:

Part 1: Logic gates

In the second booklet (to be downloaded separately):

Part 2: Combining Logic Gates

In the third booklet (to be downloaded separately):

Part 3: Sequential Circuits

Logic gates are circuits with one output and one or more inputs. They are used to make decisions based on the state of their inputs. Unlike analogue circuits, such as audio amplifiers, a logic circuit only has two states, either on or off.

Figures 2.1 & 2.2 show the differences between an analogue and digital signal. The voltage of an analogue signal can vary between any voltage without any distinct boundaries. The digital signal however only has two states: on and off.

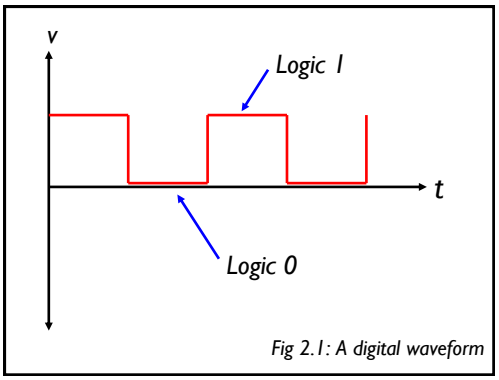

Fig 2.1: A digital waveform

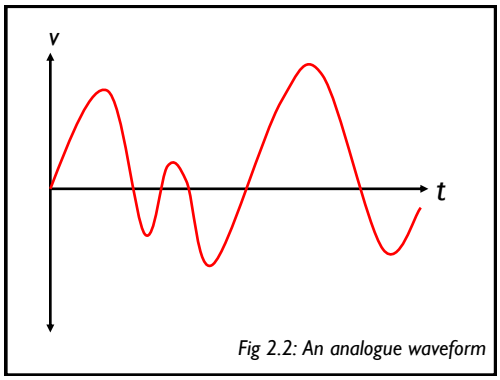

Fig 2.2: An analogue waveform

The correct names for the on and off states are called logic states and are referred to *as logic 0* (for off) and *logic 1* (for on). As well as this the terms high and low are sometimes used depending on the context:

Logic 1	=	High	=	On
Logic 0	=	Low	=	Off

Logic families and logic levels

Each logic level is represented by a certain voltage, or more accurately, a range of voltages. The voltage boundaries depend on the type of integrated circuit. The two main *families* are TTL (Transistor Transistor Logic) and CMOS (Complimentary Metal Oxide Semiconductor). The boundaries for these two technologies are shown in the table below. Note that while the TTL logic level voltages are preset, the voltages representing either logic level in CMOS integrated circuits is a percentage of the supply voltage (Vcc).

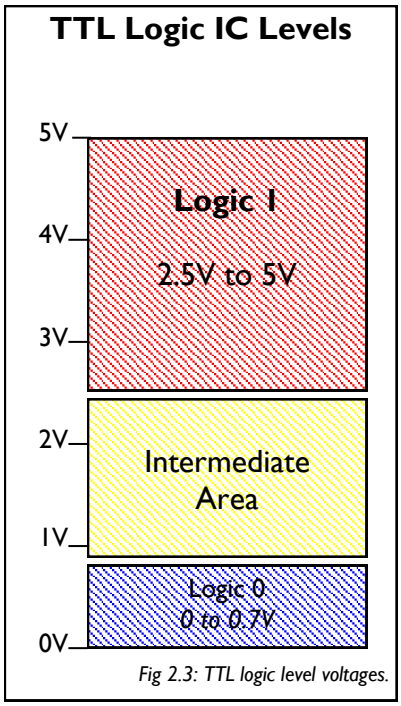

Fig 2.3: TTL logic level voltages.

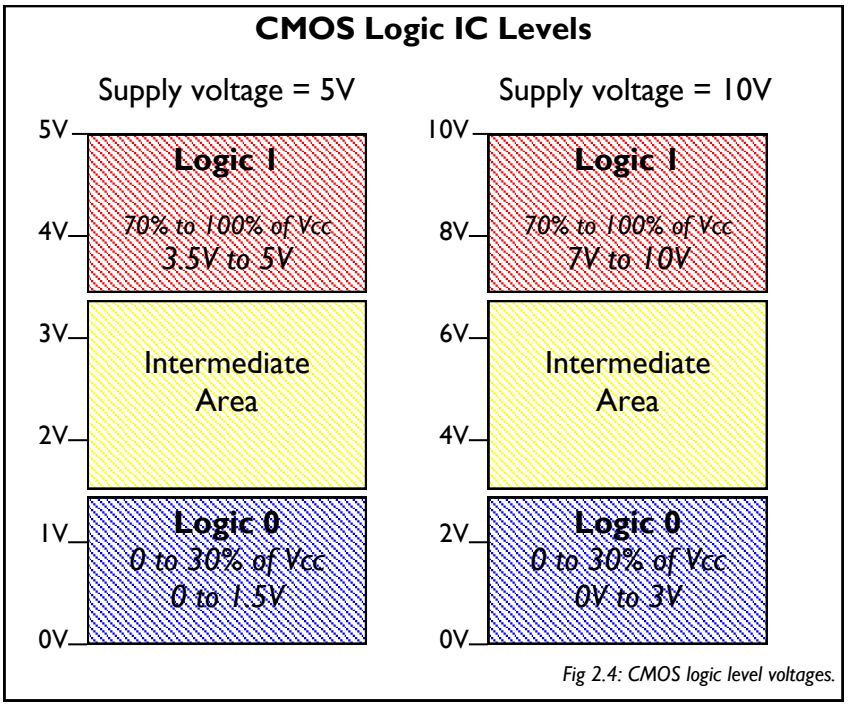

Fig 2.4: CMOS logic level voltages.

Logic gate symbols

There are seven basic types of logic gate, each of which has its own symbol to represent it. In actual fact each logic gate could be drawn as a circuit diagram but this would make things very complicated and difficult to understand.

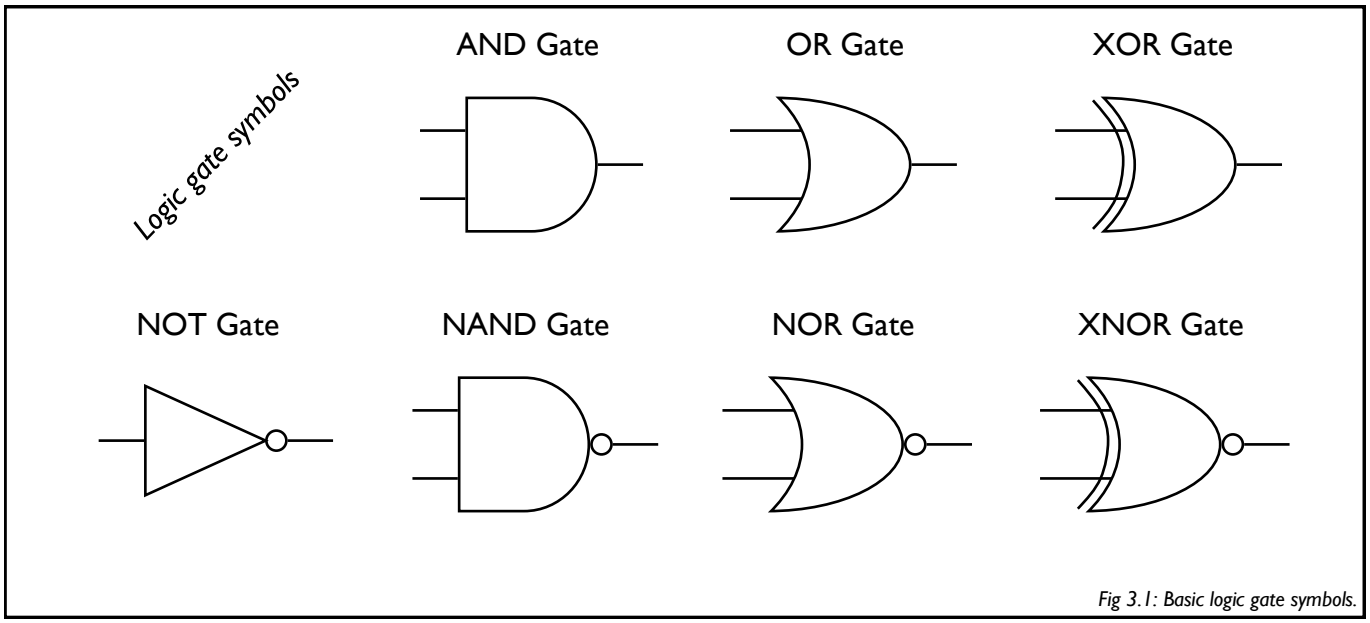

Fig 3.1: Basic logic gate symbols.

Truth Tables

A truthtable is a method of representing how a logic circuit is working. Each logic gate can be represented in terms of one of these and they can also be used to represent much more complicated circuits including more than one logic gate.

The inputs are generally labelled A,B, C &c and the output is conventionally labelled "Q". O for output would seem more logical but it is easily confused with a zero.

Inputs

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Output

This pattern of inputs represents every single combination that could exist for a given circuit.

Fig 3.2: Example truthtable (OR gate).

Introducing Boolean Algebra

All logic circuits can be represented in an algebraic form. Boolean Algebra is used to explain logical operations and is a vital tool in the design and development of more complex circuits consisting of many logic gates connected together. The Boolean expressions for the basic logic gates are shown below:

The line above the letter is called a negator and indicates that the output is inverted. This is also indicated by the circle on the output of a logic gate.

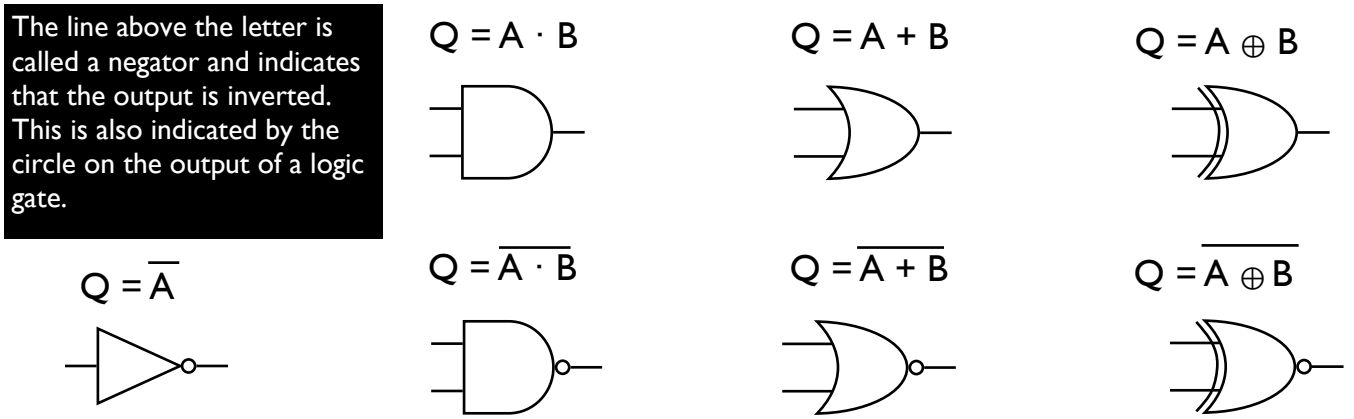

Fig 3.3: Logic gate Boolean Expressions.

The AND Gate

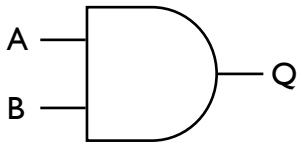

Fig 4.1: AND gate symbol.

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Fig 4.2: AND gate truthtable.

An AND gate can have a number of inputs and one output. The symbol and truth table shown is for a 2-input AND gate. As can be seen, that output is only high when:

A **AND** B are high.

This can be written in Boolean algebraic form:

$$Q = A \cdot B$$

The OR Gate

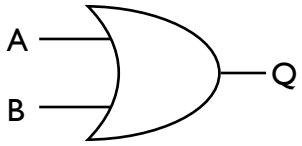

Fig 4.3: OR gate symbol.

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

Fig 4.4: OR gate truthtable.

An OR gate can have a number of inputs and one output. The symbol and truth table shown is for a 2-input OR gate. As can be seen, that output is only high when:

A **OR** B **OR** both are high.

This can be written in Boolean algebraic form:

$$Q = A + B$$

The XOR Gate—Exclusive OR

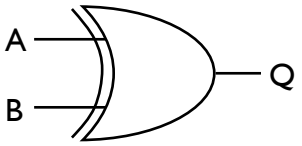

Fig 4.5: XOR gate truthtable.

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

Fig 4.6: XOR gate truthtable.

An XOR gate can only have two inputs and one output. . As can be seen, that output is only high when:

A **OR** B but not when both **are high**.

This can be written in Boolean algebraic form:

$$Q = A \oplus B$$

The NAND Gate

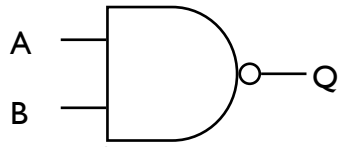

Fig 5.1: NAND gate symbol.

A NAND gate can have a number of inputs and one output. The symbol and truth table shown is for a 2-input NAND gate. As can be seen, that output is only **low** when:

A **AND** B are high.

This can be written in Boolean algebraic form:

$$Q = \overline{A \cdot B}$$

The negator in this case indicates that gate is a NAND gate rather than an AND gate. This is also indicated on the gate by the circle on the output, this is also called a negator. You would read this expression as: "Q equals not A and B".

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

Fig 5.2: NAND gate truthtable.

The NOR Gate

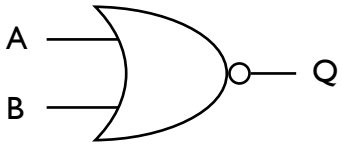

Fig 5.3: NOR gate truthtable.

A NOR gate can have a number of inputs and one output. The symbol and truth table shown is for a 2-input NOR gate. As can be seen, that output is only **low** when:

A **OR** B **OR** both are high.

This can be written in Boolean algebraic form:

$$Q = \overline{A + B}$$

The negator in this case indicates that the gate is a NOR gate rather than an OR gate. You would read this expression as: "Q equals not A or B".

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

Fig 5.4: NOR gate truthtable.

The XNOR Gate—Exclusive NOR

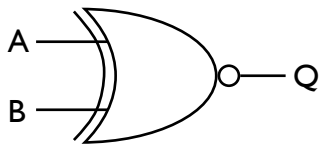

Fig 5.5: XNOR gate truthtable.

An XNOR gate can only have two inputs and one output. . As can be seen, that output is only **low** when:

A **OR** B but not when both **are high**.

This can be written in Boolean algebraic form:

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

The negator in this case indicates that the gate is a XNOR gate rather than an XOR gate. You would read this expression as: "Q equals not A xor B".

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

Fig 5.6: NAND gate truthtable.

The NOT Gate

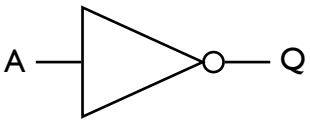

Fig 6.1: NOT gate truthtable.

Input **A**	Output **Q**
0	1
1	0

Fig 6.2: NOT gate truthtable.

A NOT gate has only one input and one output and is the most simple of logic gates.

The output state is always the opposite to the input. Therefore if the input is high then the output will be low and vice versa. Put another way, the output is NOT the input.

$$Q = \overline{A}$$

You would read this expression as: "Q equals not A".

Logic gates can be connected together to form more complex systems. Circuits can be designed which can perform mathematical functions such as addition, subtraction, multiplication and division. They can also be combined to make *flip-flops* which are circuits which are able to remember logic states and are the basis of computer memory. In essence a whole computer can be manufactured from basic logic gate building blocks you have seen so far.

Combining two NOT gates to make a buffer

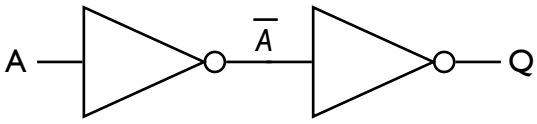

Fig 6.3: Two NOT gates combined.

Input **A**	Midpoint $\overline{A}$	Output **Q**
0	1	0
1	0	1

Fig 6.4: NOT gate truthtable.

This circuit shows how two NOT gates can be connected together. On the face of it this is a pretty useless circuit since the output is effectively the same as the input. A circuit like this is called a buffer.

$$Q = A$$

Buffers enable the output of a circuit to be isolated from the proceeding circuit or output transducer and can be used to drive loads requiring a larger current

In very complex systems a signal from one part of a circuit may need holding back while another can catch up before being processed. When one signal is out of synchronisation with another *glitches* can occur which can seriously effect the function of a circuit. Inverters can be used to facilitate this.

The topic of combinational logic (combining more than one logic gate) is to be found in the second chapter of this book. This is downloaded separately.