

# Logic

## 1. Why use binary

In our everyday lives, we use a number system that uses the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. This is called a 'Denary System'. The denary system has 10 separate numbers.

However, inside a computer, the CPU is made up of millions of tiny switches that can only be in one of two states, either 'ON' or 'OFF'. While the processing of data is taking place, the switches will alternate between these two states.

A denary counting system is not well-suited to this type of work. So instead we use a **binary** counting system, where the only numbers are 1 and 0. A 1 represents a switch in the CPU that is ON and 0 represents a switch that is OFF.



This allows us to describe the state of the CPU numerically. This section will discuss how we represent the comparison operations that a CPU makes - a topic called 'binary logic'.

# Logic

## 2. What is Logic

In general, you use 'logic' to work out whether doing something causes the result to be 'True' or 'False'.

The outcome of applying **logic** is always either true or false

Consider the painful experiment below (don't try this at home!):

*'If I stick my finger in boiling water then I will burn my finger'.*

Is this a True statement?

You work out the truth of this statement by applying some known rules such as:

Rule 1: I am human

Rule 2: Human fingers are harmed by contact with boiling water

Therefore the two rules are applied one after another. 'I am human' (True), 'I've stuck my finger in boiling water (True) therefore (logically) my finger will be burnt



It is very important to note that normal logic does not have anything in-between, there is no 'slightly true' or 'slightly false' result.

It is either **True** or **False**.

The early designers of computers quickly realised that *logic* can be used within a computer to work out problems. After all, computer data is either 1 (Logic True) or 0 (Logic False).

## Logic

### 3. Binary Logic

Binary logic is much the same as general logic, except it works on 1's and 0's. Given one or more binary inputs a logical output will result.

It is common practice to consider binary 1 to be 'True' and binary '0' to be 'False'.

There are only a few simple logic operations but they do lead to incredibly complex devices such as a CPU.

The logic operations are: NOT, AND, OR

#### *Invert or NOT operation*

This operation operates on a single input, let's call it input A to produce a single output Q

In English the rule is

*"If A is True then the output Q is False, if the input A is False then the output Q is True"*

So this operation produces an output that is opposite to the input

The shorthand way of writing this is  **$Q = \text{NOT } A$**  where 'A' is the input and 'Q' is the output.

It does not have to be A and Q, you could use any letter, but Q is a popular choice to describe a logic output.

#### *The AND operation*

This operation acts upon at least two inputs, say A, B to produce a single output Q.

In English, the rule is

*"If both A and B are True then the output Q is also True, otherwise it is False".*

The shorthand for this is  **$Q = A \text{ AND } B$**

#### *The OR operation*

Again, this needs at least two inputs A, B to produce a single output.

In English the rule is

*"If either or both A, B are True then the output Q is also True."*

The shorthand for this is  **$Q = A \text{ OR } B$**

The table below is a summary

# Binary Logic

Inputs and Output names	Type of Logic	Equivalent Statement
Input A, Output Q	NOT	$Q = \text{NOT } A$
Inputs A, B, Output Q	AND	$Q = A \text{ AND } B$
Inputs A, B, Output Q	OR	$Q = A \text{ OR } B$

*Extra Fact:*

There is one other popular logic operation called the 'Exclusive OR' but you do not need to know the details for this syllabus. But basically it says "If A OR B is true then the output is true, however if both of them are true at the same time, then the output is false"

## Logic

### 4. Truth Tables

You have seen the statements on the previous page about the NOT, AND, OR logic operations. However, it is not simple to work out in your head the result for every combination of input, especially if more than two inputs are present.

A very popular method of showing exactly what will be the output for every possible input is to use a 'Truth Table'.

A **truth table** lists every possible combination of input and the resulting output.

We shall use A and B as inputs and Q as the output in the truth tables below  
*NOT Truth Table*

This is the simplest truth table of all as it is

A	Q
---	---

A	Q
1	0
0	1

This truth table is the equivalent of the statement **Q = NOT A**

You can immediately see that the output is the opposite of the input

### *AND Truth Table*

This is the truth table for two input AND logic

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

This truth table is the equivalent of the statement **Q = A AND B**

With two inputs there are 4 possible combinations (you work it out by  $2^{\text{number of inputs}}$   $2^2$   $2 \times 2$   $4$ ). The truth table above shows that the output Q is true only when all inputs are also true.

For example 3 inputs has  $2^3$  or 8 combinations

4 inputs has  $2^4$  or 16 combinations

### *OR Truth table*

This is the truth table for two input OR logic

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

This truth table is the equivalent of the statement  **$Q = A \text{ OR } B$**

With two inputs there are 4 possible combinations. The truth table above shows that the output Q is True if any input is logic True.

### *Larger Truth tables*

The tables above are for the basic two input logic operations. However, in practical logic there could be many more inputs, A,B,C,D. The truth table will have to itemise every possible combination of inputs.

This will be explored a little further after explaining Logic gates.

## Logic

### 5. Logic Gate

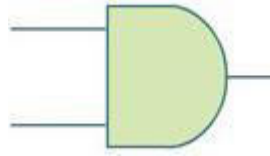
We have discussed logic in a fairly abstract way. No mention has been made of the actual hardware needed to carry out these logic operations.

Electronic engineers have designed hardware that can carry out the fundamental logic operations (AND, OR, NOT).

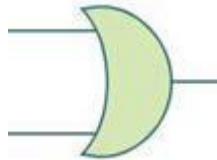
This hardware is called a 'logic gate'.

A logic gate is actually made up of a number of transistor switches arranged in a certain way. But for this syllabus you do not need to concern yourself about the details. It is enough to know that a logic gate carries out a logic operation.

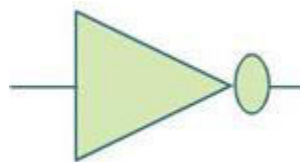
A logic gate that carries out the AND operation is called the AND Gate. When drawing an AND gate, use the following symbol



A logic gate that carries out the OR operation is called the OR Gate



A logic gate that carries out the NOT operation is called the NOT Gate or 'Inverter'



## Logic

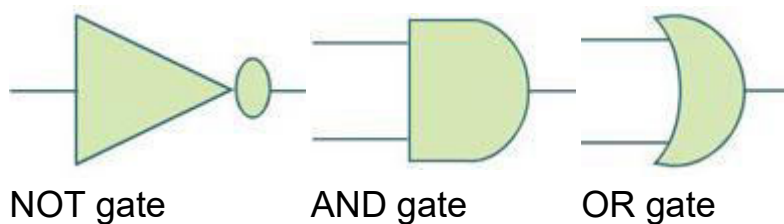
### 6. Logic circuits

Digital chips such as a CPU make use of billions of these gates in various combinations.

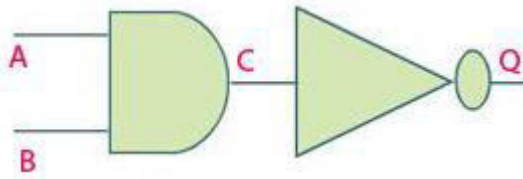
A group of connected logic gates is called a **logic circuit**.

To design a logic circuit, you must first create a logic circuit diagram to represent the gates used and their connections.

Each type of gate has its own symbol, shown below.



For example if the logic circuit is made up of an AND gate whose output is connected to a NOT gate, it looks like this:



In addition to setting out a logic circuit in graphical form, the same circuit can be set out as a boolean expression. For example the diagram above has the equivalent statement

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

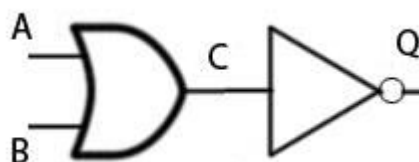
## Logic

### 7. Combining logic gates (2)

The OR gate followed by a NOT gate

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

**Truth table**



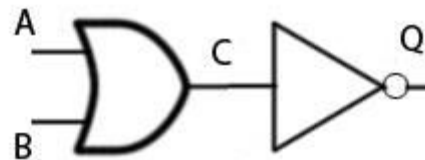
**Logic gate**



To work out what is happening, start off with the truth table for the OR gate, with the output set to C like this:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

**Partial Truth table**

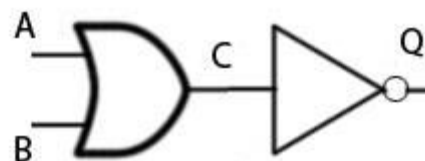


**Logic gate**

This shows that C will only be true (1) if either A or B or both are 1. The next step is to feed the output C into a NOT gate and list its output Q in the completed truth table. Like this:

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

**Completed Truth table**



**Logic gate**

This shows that the final output Q is the opposite (inverse) of the in-between value C.

The next page makes use of 3 inputs to show how diagrams are built up.

Logic Statement

A logic statement describes the output in terms of all the inputs and it can be used instead of a truth table to work out the result of every combination of input.

For instance you can also write a logic statement for the above logic diagram like this:

$$Q = \text{NOT} (A \text{ OR } B)$$

To do this start from the final output and work backwards. So the output Q is

$$Q = \text{NOT}(C)$$

$$\text{but } C = A \text{ OR } B$$

Replace C in the first statement to end up with the complete statement

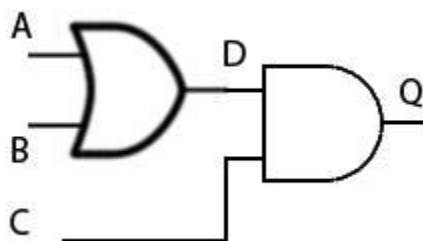
$$Q = \text{NOT} (A \text{ OR } B)$$

This statement describes the output in terms of the inputs without any in-between values. A logic statement should always end up with a single output and all the inputs.

## Logic

### 8. Combining logic gates (3)

The previous page had only two inputs. Let's now consider a logic diagram with 3 inputs A,B and C.



This logic diagram shows three inputs, with A and B going into an OR gate, its output D is then fed into an AND gate along with input C to produce a single output Q.

Let's work out the truth table for this. We know from the start that the truth table will have 8 rows because there are 8 possible combinations of 3 inputs. ( $2 \times 2 \times 2 = 8$ )

The truth table for A,B and D is that of an OR table, like this

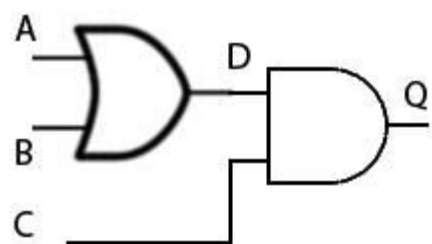
A	B	D
0	0	0
0	1	1
1	0	1
1	1	1

Now D goes into an AND gate along with input C. We already have all the possible states of D, so let's consider the output when C is always 0.

A	B	D	C	Q
0	0	0	0	0
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0

AND gate inputs C, D

**Partial Truth table**



**Logic gate**

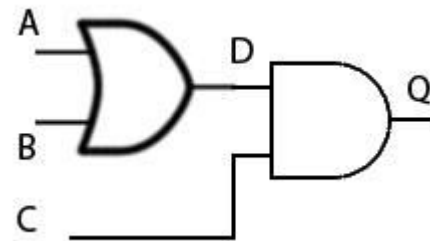
The partial truth table above shows what Q will be when C is always zero (0) AND every combination of D. In this case it shows that output Q is always zero.

But now let's consider the truth table when C is always 1. Like this

A	B	D	C	Q
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	1	1

AND gate inputs C, D

**Partial Truth table**




**Logic gate**

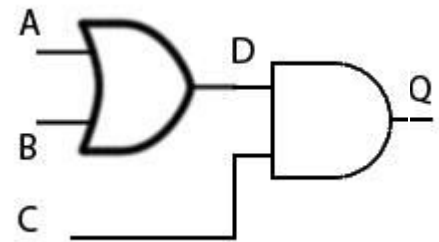
You can see that the output Q will be 1 when D AND C are 1.

These two partial truth tables have considered every possible combination of A,B,C. Now combine the two partial truth tables into a complete one as shown below. The top half is when C is 0 and the bottom half is when C is 1.

A	B	D	C	Q
0	0	0	0	0
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	1	1


 OR gate inputs A,B      AND gate inputs C, D

**Completed Truth table**



**Logic gate**

Logic Statement

The logic statement for this diagram can be determined by working from the output towards the inputs A,B,C

Step 1. The final gate is

$$Q = C \text{ AND } D$$

but D is also  $D = A \text{ OR } B$

Swap D in the first statement for that of the second statement

$$Q = C \text{ AND } (A \text{ OR } B)$$

This statement describes the output given all the inputs.

**Logic**

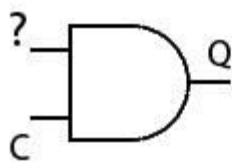
## 9. From statement to logic diagram

If you were given a statement in an exam, you could work out the logic diagram by taking it one step at a time.

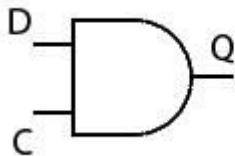
For example, work out the logic diagram for this statement

$$Q = C \text{ AND } (A \text{ OR } B)$$

Step 1. You can see that the input C has an AND operation with another input, so the final gate in the diagram will be an AND gate like this:

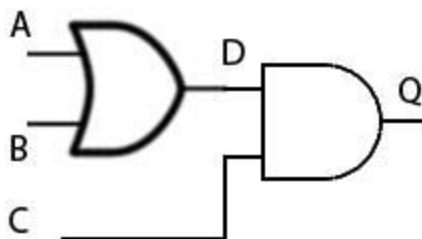


Let's call the question mark 'D', like this



The statement for this gate is  $Q = C \text{ AND } D$

From the statement you can see that D could be swapped for (A OR B) in the original statement which means D is the result of inputs A and B being fed through an OR gate. So draw an OR gate joined up to the AND gate like this



This is the logic diagram for the statement  $Q = C \text{ AND } (A \text{ OR } B)$

## Logic

### 10. Logic symbols

In this introduction to logic we have used the words AND NOT OR to describe the logic operations. However this is not how logic expressions are written down normally.

Instead there are a set of agreed symbols to be used. The table below shows how they are used with the letters A B

Logic operation	OCR Exam symbols	Engineering symbols
NOT A	$\neg A$	$\bar{A}$
A AND B	$A \wedge B$	$A . B$
A OR B	$A \vee B$	$A + B$

The OCR uses the standard mathematical symbols and this is what you will see in your exam. For example the expression for A AND NOT B AND (C OR D) looks like this

$$A \wedge \neg B \wedge (C \vee D)$$

However engineers rarely use this notation as it can be difficult to work with when simplifying long complicated expressions, especially when doing it by hand. Instead they use a bar over a letter to mean NOT then a + to mean OR and a dot between letters to mean AND. Therefore the same expression A AND NOT B AND (C OR D) looks like this

$$A . \bar{B} . (C + D)$$

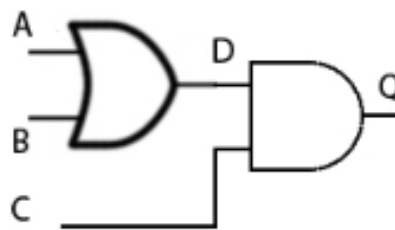
The + symbol does not mean 'plus' it means OR when used in logic expressions. For some reason it seems a lot easier to read and manipulate the expression in this form rather than the very unfamiliar symbols that mathematicians use.

Perhaps because the standard maths symbols are as big as the letters they are operating upon and so it tends to be harder to see the actual input and output letters.

## Logic

### 11. Converting logic diagrams into expressions

Here is a two gate logic diagram



In OCR symbol format the expression for Q is

$$Q = (A \vee B) \wedge C$$

In engineering symbols the expression for Q is

$$Q = (A + B).C$$

## Logic

### 12. Summary

- The hardware of a CPU is made up of billions of tiny switches called transistors
- A switch has only two states - ON and OFF
- The ON can be represented by binary 1
- The OFF can be represented by binary 0
- In terms of logic, binary 1 is considered as logic TRUE



- In terms of logic, binary 0 is considered as logic FALSE
- Logic is used within a CPU to carry out its functions
- The three basic logic functions are NOT, AND, OR
- There is a fourth logic function called EXCLUSIVE OR
- A logic statement sets out the function of a logic operation such as  $Q = A \text{ AND } B$
- The hardware used to carry out a logic operation is called a logic gate
- There are three types of logic gate an AND gate, an OR gate and a NOT gate
- Each logic gate type has a standard symbol shape
- Logic gates are combined to make more complicated logic operations
- A truth table lists all the possible combinations of inputs along with the single output