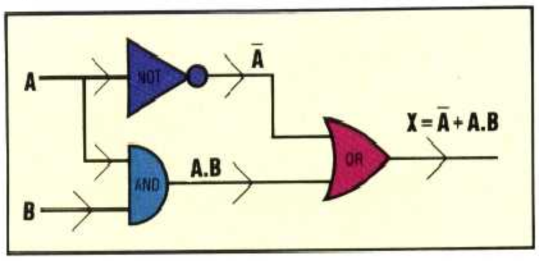
THE BUILDING BLOCKS OF ADDITION

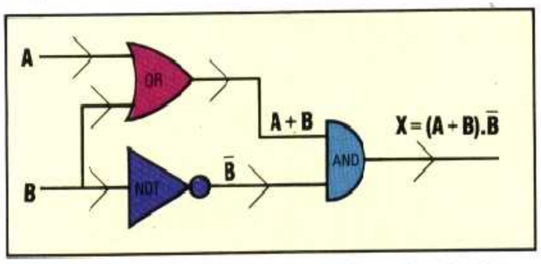
In the first instalment of this course, we looked at the three kinds of logical building blocks (AND, OR, and NOT) and saw how these may be combined together to produce simple logic circuits. Now we begin to investigate the way in which these logic circuits can be used to perform the function of addition.

The system of using algebraic notation to describe logical relationships is known as Boolean algebra, and is named after the mathematician George Boole (1815–1864). Boolean algebra is of great use in computer circuit design because it allows mathematical simplification of logic circuits. This means that fewer logic gates are required to perform a given function, which in turn increases the speed of operation of the machine.

We have already met the Boolean notation for the output from the three basic logic gates: AND (A.B), OR (A+B), and NOT ($\overline{A}$). More complex circuits can be represented by using these expressions. For example, the Boolean expression $X = \overline{A} + A.B$ represents this circuit:



It is important to notice that the order in which the operations AND and OR are carried out is significant. The simple rule is that AND has priority over OR (as well as over NOT). If it is wished to reverse the order of priority then brackets have to be used, as in this example: $X = (A + B) . \overline{B}$. For this expression the two inputs are ORed together before being ANDed with the negative of B. Here is the circuit diagram:



When drawing a logic circuit from its Boolean expression it is often best to start at the output and work backwards to the inputs. This method produces better circuit diagram layouts.

INCLUSIVE AND EXCLUSIVE OR

There are two possible meanings of the word OR in everyday English. The first meaning is the one that we have already met, namely:

One OR the other OR both.

The second meaning has important consequences for logic circuit design:

One OR the other but not both.

For example, to qualify for a competition for two-wheeled vehicle owners you could own a motorcycle OR a bicycle (or you may own both). This is a case of the *inclusive* use of OR. On the other hand, you may be tall OR you may be short (but you can't be both). In this case the use of the word OR *excludes* the possibility of both statements being true.

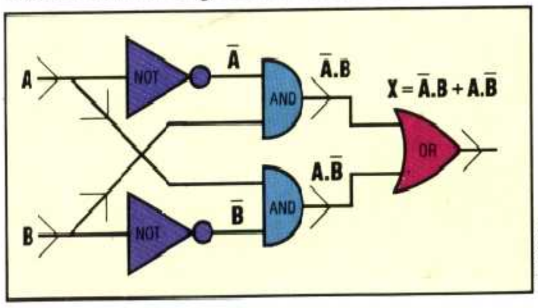
In logic circuits this *exclusive* OR (or XOR) operation is a useful tool and can be built up from the AND, OR and NOT set of gates. The truth table for XOR is:

INPUTS		OUTPUTS
A	B	$A \forall B$
0	0	0
0	1	1
1	0	1
1	1	0

As can be seen from the second and third rows of the truth table, a value of one can be produced as output if:

NOT(A) is ANDed with B
OR
A is ANDed with NOT(B).

This can be written as a Boolean expression in this way: $X = \overline{A}.B + A.\overline{B}$. A possible circuit to produce the Exclusive–OR operation from the above Boolean expression would be:



This would produce a five gate circuit. We shall see later how it is possible to simplify this circuit to one with only four gates using Boolean algebra.