THE LOGICAL ANSWER

We have now covered some substantial ground in the Logic course, having concentrated in particular on the areas of basic logical principles and Boolean algebra. In this instalment, we take an overview of the rules explained so far, and provide a series of revision questions and answers.

Let us undertake a short review of the work we have covered in the previous instalments. The principles of logic apply to computers in the design of hardware that must carry out certain special jobs. We have already designed an adder circuit, which when combined with other adder circuits allows the addition of binary numbers (see page 48). This circuit models the human method of addition, allowing digits to be carried from one column to the next.

We used three basic elements in the design of this circuit, called logic gates. The functions that these gates could perform were indicated by the names we gave them (AND, OR and NOT), and were each defined by a truth table. The truth table is a simple way of writing down the output(s) from a circuit for any possible combination of inputs. Two inputs gave us four (2^2) possible input combinations, three inputs gave us eight (2^3) combinations, and so on. We also saw how logic gates can be linked together to give certain desired outputs. These more complicated circuits could also be described by their truth tables. In particular, we designed a five-gate Exclusive-OR circuit, which gave a true output when only *one* of its inputs was true (see page 32).

BOOLEAN ALGEBRA

The combination of logical elements can be described on paper by a set of symbols rather like that of normal algebra. The branch of mathematics concerned with the representation of logic is known as Boolean algebra, after the English mathematician George Boole (1815-64) who first defined its principles. Each of the three basic elements of logic has its own special symbol:

A AND B	A.B
A OR B	A+B
NOT A	$\overline{A}$

Just as there are laws that govern the manipulation of figures in arithmetic and letters in algebra, so there are special laws that govern the simplification of logical expressions. The laws of Boolean algebra are summarised in the table that follows.

SPECIAL RELATIONS	
Relation	**Dual**
A.A = A	A + A = A
$A.\overline{A} = 0$	$A + \overline{A} = 1$
A.0 = 0	A + 1 = 1
A.1 = A	A + 0 = A
A.(A + B) = A	A + A.B = A
$A.(\overline{A} + B) = A.B$	$A + \overline{A}.B = A + B$
DE MORGAN'S LAWS	
1) $\overline{A + B} = \overline{A}.\overline{B}$	
2) $\overline{A.B} = \overline{A} + \overline{B}$	
ASSOCIATIVE LAW	
A.(B.C) = (A.B).C = A.B.C	
A + (B + C) = (A + B) + C = A + B + C	
COMMUTATIVE LAW	
A.B = B.A	
A + B = B + A	
DISTRIBUTIVE LAW	
A.(B + C) = A.B + A.C	

Using these rules it is possible to simplify logical expressions and reduce the number of gates required in the final circuit. In addition to the algebraic method, we have also discussed the use of Karnaugh maps in logic circuit simplification (see page 92). Karnaugh maps represent a significant advance. Although they do not replace algebraic simplifications, they do reduce the amount of effort inherent in dealing with Boolean algebra. These maps, which are really extensions of Venn diagrams (see page 46), allow the grouping of expressions extracted from a truth table into twos, fours or eights. These groups represent simpler Boolean expressions, and so simplification is achieved. In practice, a combination of k-maps and algebra is often required to produce the most efficient circuit.

We have also discussed the use of the logical AND and OR operations in BASIC programming. We saw how these commands can be used to combine conditions in BASIC IF... THEN statements, and how they enable the programmer to turn on and off isolated bits within a register (see page 66).

As the culmination of this first stage of the course, we used all our knowledge of truth tables, Karnaugh maps, Boolean algebra and logic gate notation to design circuits that would give the desired outputs for certain defined tasks (see page 106). The following Review Exercise covers all the aspects of the course so far. Once you feel confident in using these rules, we can proceed to tackle more advanced logical theory.