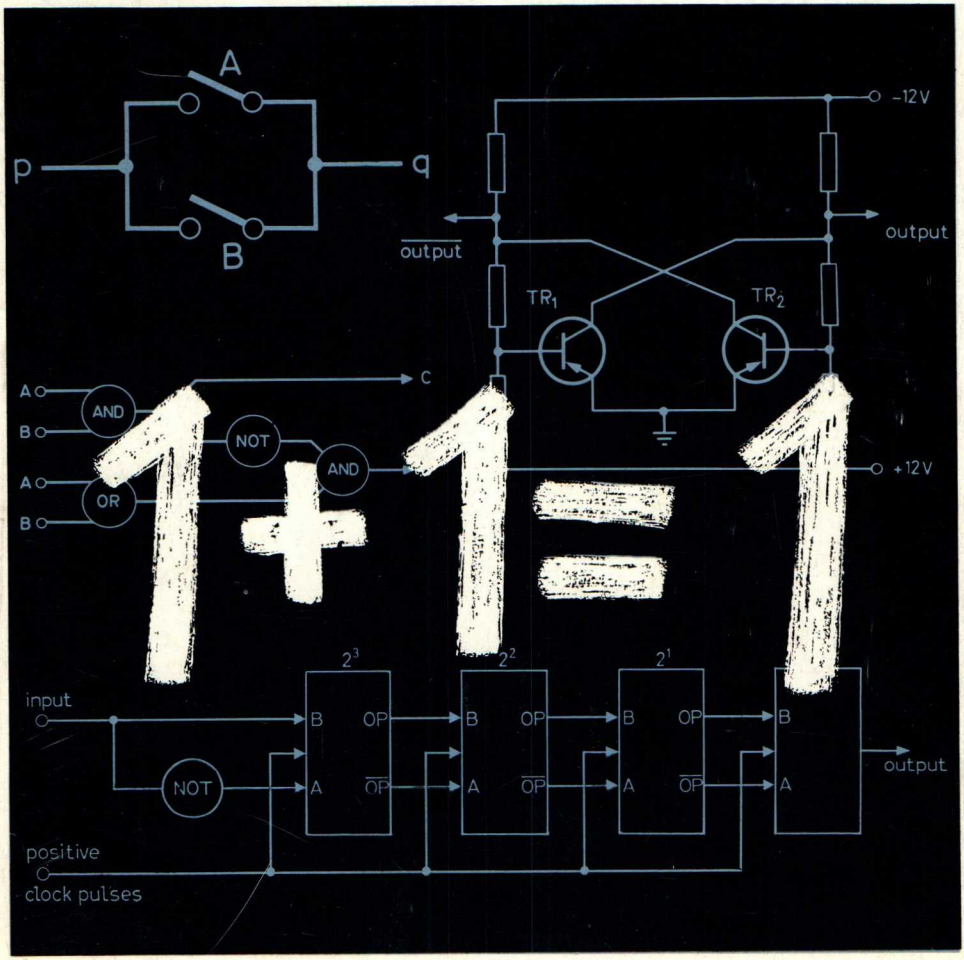
PHILIPS

APPLICATION BOOK

PHILIPS ELECTRONIC COMPONENTS AND MATERIALS DIVISION

INTRODUCTION TO DIGITAL TECHNIQUES



INTRODUCTION TO DIGITAL TECHNIQUES

The contents of this booklet are extracted from Part VIII of the Electronics Trainers Manual produced by the Educational Products and Systems Department, in close co-operation with R. H. GRIGG, Assoc. I.E.R.E. Royal Military College of Science, Shrivenham, England.

CONTENTS

Introduction

For centuries mankind has been interested in calculations, and naturally over the years mechanical aids have been invented and developed. One of the earliest calculating devices was the *abacus*, in which the numbers were represented by beads threaded on parallel wires. Essentially it was a pulse operated system, the pulses being supplied by the operator moving the beads.

A modern digital computer is also a pulse operated device, but since the pulses are electronic very fast operating speeds are possible. The range of potential applications for the modern machine is enormous, varying from scientific research on the one hand to the calculations necessary in banking and commerce on the other. In addition, the computer is becoming more and more important in the industrial field, not only for calculating purposes but also for the precise control of plant and machinery. Under these conditions it is obvious that there is an increasing need to extend the understanding of the basic principles and the electronic techniques used in digital computing.

CHAPTER 1

NUMBER SYSTEMS

CHAPTER CONTENTS

Decimal system

Throughout the world many different number systems are in daily use, but probably the most commonly used is the decimal system. By convention everyone readily understands that a number such as 1965 means one thousand, nine hundred and sixty five, because the notation employed is a *positional* one. Reading from the right the first digit gives the units, the next gives tens, the next hundreds and so on. The number can therefore be expressed in the following manner:

$$1965 = 1 \times 10^3 + 9 \times 10^2 + 6 \times 10^1 + 5 \times 10^0$$
$$= 1000 + 900 + 60 + 5$$

It should be noted that any number raised to the power of zero is equal to unity.

An integer, or whole number, is given in the example, but the notation can also be used for fractional quantities if negative indices are employed. For example:

$$1.965 = 1 \times 10^0 + 9 \times 10^{-1} + 6 \times 10^{-2} + 5 \times 10^{-3}$$
$$= 1 + \frac{9}{10} + \frac{6}{100} + \frac{5}{1000}$$

Since the system is based on powers of ten, a shift to the left multiplies by ten and a shift to the right divides by ten.

Binary system

In an electronic digital computer, the digits are usually represented by different values or levels of potential, and if the decimal number system were used it would mean that the circuits must be capable of differentiating accurately between the ten levels representing the various digits. Of course this is possible with careful circuit design, but the circuits may be complex and the chance of error is quite high. In order to simplify circuit design and improve reliability, the *binary system* of numbers is generally employed in digital computing. As the binary system is based on powers of two there are only two digits, usually referred to as *bits*, namely: 0 and 1. The machine has now only to recognise two separate levels of potential.

A positional notation is again used, as shown below, where the binary equivalents are tabulated for numbers up to ten.

Number	Binary Equivalent				
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0×2^0
1	0	0	0	1	1×2^0
2	0	0	1	0	$1\times2^1+0\times2^0$
3	0	0	1	1	$1\times2^1+1\times2^0$
4	0	1	0	0	$1\times2^2+0\times2^1+0\times2^0$
5	0	1	0	1	$1\times2^2+0\times2^1+1\times2^0$
6	0	1	1	0	$1\times2^2+1\times2^1+0\times2^0$
7	0	1	1	1	$1\times2^2+1\times2^1+1\times2^0$
8	1	0	0	0	$1\times2^3+0\times2^2+0\times2^1+0\times2^0$
9	1	0	0	1	$1\times2^3+0\times2^2+0\times2^1+1\times2^0$
10	1	0	1	0	$1\times2^3+0\times2^2+1\times2^1+0\times2^0$
etc.	etc.				etc.

It may be seen that the least significant digit is on the right and moving from right to left each digit represents in ascending order a higher power of two. If negative indices are used, fractional quantities can also be recorded, but the digits to the right of the binary point are in a descending scale, each being half the value of its predecessor. Thus the binary number

$$\begin{aligned}1.1101 &= 1\times2^0+1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}\\ &= 1 + \tfrac{1}{2} + \tfrac{1}{4} + 0 + \tfrac{1}{16}\\ &= 1.8125\end{aligned}$$

It should be appreciated that integers can be expressed precisely in binary form, but some approximation might be necessary for certain fractional quantities. The actual error will depend on the number of binary digits in use, and in practice it is usually very small.

Since the binary system is based on powers of two, a shift to the left multiplies by two and a shift to the right divides by two. Consider for example the binary number

$$1001 = 9$$

With a shift to the left it becomes:–

$$\begin{aligned}10010 &= 1\times2^4+0\times2^3+0\times2^2+1\times2^1+0\times2^0\\ &= 16 + 0 + 0 + 2 + 0\\ &= 18\end{aligned}$$

i.e. the number is doubled.

With a shift to the right it becomes:–

$$\begin{aligned}100.1 &= 1\times2^2+0\times2^1+0\times2^0+1\times2^{-1}\\ &= 4 + 0 + 0 + \tfrac{1}{2}\\ &= 4\tfrac{1}{2}\end{aligned}$$

i.e. the number is halved.

The normal processes of arithmetic can be carried out on numbers in any scale, but in the binary system the rules are extremely simple. For instance there are only three rules to consider for addition:–

$0+0=0$
$1+0=1$
$1+1=\text{Two}=0$ and 1 to carry.

To add two numbers, say 7 and 5:

$7 = 00111$
$5 = 00101$
01100 Sum = 12
01110 Carry

Note that the *carry* is transferred into the next significant column.

It is important to realise that within the machine all computations are reduced to a series of simple steps, and basically the computer need only be capable of addition or subtraction. Multiplication can be achieved by a process of repetitive addition combined with a shifting procedure, and division may be carried out by repeated subtraction. The reduction of a problem to a form suitable for the computer and the preparation of the sequence of basic operations for any particular computation is called *programming*.

In general, conversion from decimal to binary and vice versa is a requirement to ease communication between human beings and the computer. To represent ten distinct symbols for 0 to 9, four binary digits (or bits) are necessary, and this leads to some redundancy because sixteen combinations are possible with four bits. Ten of the sixteen possible combinations are chosen to represent the decimal digits, and this allows a coded system to be used. Only a few of the possible codes have been investigated, and a detailed discussion is outside the scope of this booklet. However, the factors affecting the choice of such a code are:–

(a) convenience of arithmetical operations,
(b) ability to detect errors,
(c) ability to correct errors,
(d) efficiency of storage requirements,
(e) convenience to human beings.

The list is not given in any particular order of significance.

CHAPTER 2

BOOLEAN ALGEBRA

CHAPTER CONTENTS

2.1. THE LOGIC OF CLASSES AND PROPOSITIONS

Introduction

A little over a hundred years ago, *logical* or *Boolean algebra* was developed by George Boole (Ref. 1) to simplify complex logical propositions, but its application to the analysis and simplification of switching circuits was realised by Shannon (Ref. 2) much later. Since a digital computer contains a number of complex switching circuits, Boolean algebra plays an important part in the analysis and logic design of the system. The basic *gating elements* may be combined in many ways to achieve a particular function, and the logic designer must choose the way best suited to his purpose. The choice is often influenced by the manner in which the elements are packaged during manufacture, but obviously an arrangement which involves a minimum number of elements is attractive.

The logic of classes

In mathematical logic, a *class* is defined as a group of elements all of which possess at least one characteristic in common. From this definition it follows that a complementary class can exist in which no member possesses the common characteristic. Both classes together may be taken to form the particular universal class under consideration. It is convenient to illustrate these ideas with the aid of a *Venn diagram* as shown in fig. 1.

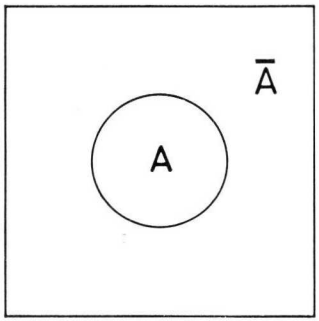


Fig. 1a.

The area within the square represents the universal class being considered. This class may be divided into a number of sub-classes: two in this case, i.e. class A contained within the circle and class $\bar{A}$ outside the circle (fig. 1a). A bar is placed above the variable to denote the complement.

The *logic of classes* is concerned with the relations between various sub-classes that may exist within a particular universal class. The complementary class has been introduced already, but it will be clear that the operations of intersection and union can also occur between sub-classes.

In order to clarify these ideas consider the following example:
Suppose the universal class is defined as a group of students, and is represented by the total area within the square. Two sub-classes exist in the group of students as shown in the *Venn* diagrams of fig. 1b, c, d, e and f.

Some are boys with brown hair.
Shaded area = class A

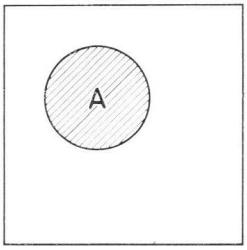


Fig. 1b.

The remainder are NOT boys with brown hair.
Shaded area = class $\bar{A}$.

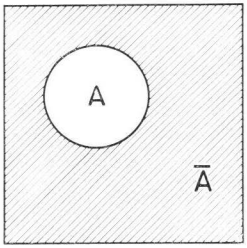


Fig. 1c.

Some are boys with blue eyes.
Shaded area = class B.

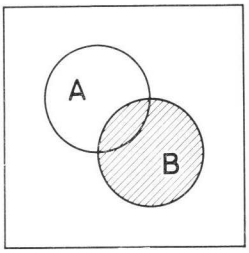


Fig. 1d.

Some are boys with brown hair AND blue eyes.
Shaded area = intersection of A and B.

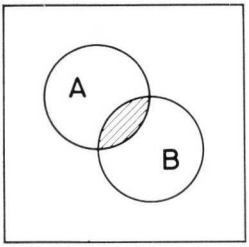


Fig. 1e.

Some are boys with brown hair OR blue eyes OR both.
Shaded area = union of A and B.

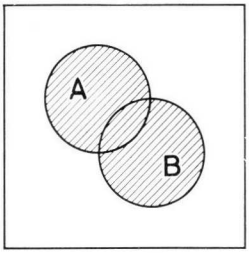


Fig. 1f.

Notice that when intersection occurs, the students are members of both A and B. Intersection is denoted by a full stop thus: $A \,.\, B$. Since there is no change in the intersecting area it is equally true to state that the students are members of both B and A, hence

$$A \,.\, B = B \,.\, A$$

For the case of union, the students are members of A or B or both, and this function is represented by a plus sign: $A + B$. Clearly, the area remains unaltered for $B + A$, therefore

$$A + B = B + A$$

Obviously the order in which the variables are placed has no significance. In the diagram for intersection the area NOT shaded represents NOT (A AND B) which may be written as $\overline{A \,.\, B}$. The area shaded in the following diagram is NOT A OR NOT B (written as $\bar{A} + \bar{B}$) and is equivalent to the area NOT shaded in the diagram for intersection.

Thus $\bar{A} + \bar{B} = \overline{A \,.\, B}$

Shaded
$\bar{A} + \bar{B}$

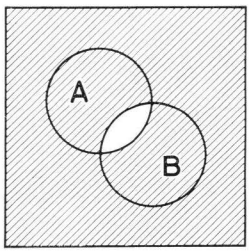
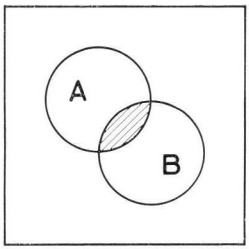


Fig. 1g.



Unshaded
$\overline{A \cdot B}$

Fig. 1h.

Both diagrams are shown side by side to aid comparison. A similar proof can be obtained for $\bar{A} \cdot \bar{B} = \overline{A + B}$ as shown by the diagrams below.

Shaded
$\bar{A} \cdot \bar{B}$

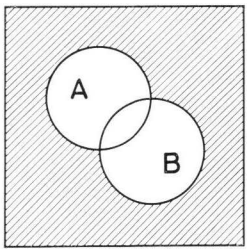
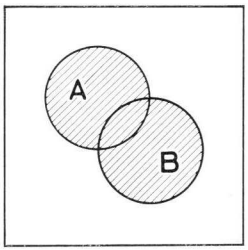


Fig. 1i.



Unshaded
$\overline{A + B}$

Fig. 1j.

These two proofs confirm De Morgan's theorem which is extremely important in Boolean algebra.

The logic of propositions

The theorems of Boolean algebra apply equally either to the logic of classes or to the logic of propositions, but in the latter the emphasis is placed on wether a given statement is true or false. In compound logical propositions two basic connectives are used, and since these are similar to the operations of intersection and union previously described the same symbols will be used. The basic connectives are:–

(a) AND formally known as conjunction–written as $A \cdot B$
(b) OR formally known as disjunction–written as $A + B$

(Students should realise that in textbooks and current literature many different symbols are used for the basic connectives.)

Conjunction is the proposition that both A and B are true, while *disjunction* is the proposition that A or B or both are true. For example, the statement "Today is Tuesday and it is raining" can be represented as follows:

Let A = "Today is Tuesday" and B = "it is raining"
If C = the complete statement then $C = A \cdot B$

When is the complete statement true? This can be established by tabulating all the possible combinations of truth and falsity for A and B, and deciding the particular conditions under which C is true as shown:-

A	B	C
False	False	False
False	True	False
True	False	False
True	True	True

Values can be assigned to these two possibilities and normally a 1 is used to denote the value of a true statement, and a 0 the value of a false statement. Under these conditions the table becomes:-

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

Such a table is known as a "Truth Table", and the proof is called a proof by the method of perfect induction. Clearly in this case the complete statement is true only when A AND B are true.

A similar procedure can be adopted for disjunction, and the equation $A+B=C$ may be obtained from a suitable proposition such as "There is a voltage on terminal number 3 or on terminal number 7". The truth table is given below.

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

For the operation of complementing or negating, a bar is placed above the variable, and the truth table is quite simply:-

A	$\bar{A}$
0	1
1	0

2.2. POSTULATES AND THEOREMS

Postulates

Only a summary of the *postulates* (i.e. fundamental conditions) is given since the truth of these is self-evident.

A variable may assume one of two values: 0 or 1.

The complement of 0 is 1 and vice versa.

$$1 \cdot 1 = 1 \qquad 0+0=0$$
$$1 \cdot 0 = 0 \cdot 1 = 0 \qquad 0+1=1+0=1$$
$$0 \cdot 0 = 0 \qquad 1+1=1$$

The postulates form the basis of a number of theorems which allow expressions to be manipulated and simplified. Most theorems in Boolean algebra fall into pairs, each expression being the *dual* of the other. In order to obtain the dual of an expression, exchange AND for OR and exchange 0 for 1. An example will serve to illustrate the procedure, but in fact the postulates given above are presented in dual pairs.

The dual of $\quad 0 \cdot A + B \cdot \bar{C} + 1$

is $\quad (1+A) \cdot (B+\bar{C}) \cdot 0$

It should be noted that normally the *AND function takes precedence over the OR function* in the absence of any other information. It is, therefore, necessary to add brackets to the dual expression to avoid any ambiguity.

Theorems

In an expression such as $A \cdot B + \bar{A} \cdot C + D(\bar{B}+\bar{C}) + \bar{D} \cdot E$ there are five variables, *A B C D* and *E*, and nine *literals*.

A literal is defined as each appearance of a variable or of its complement.

To obtain the complement of an expression, exchange AND for OR, exchange 0 for 1 and complement all literals.

The complement of $\quad 0 \cdot A + B \cdot C + 1$

is $\quad (1+\bar{A}) \cdot (\bar{B}+C) \cdot 0$

Where possible the following theorems are presented in dual pairs. It may then be shown that, if the first expression is true, the dual must also be true. In order to prove this point, both expressions for theorem 1 will be verified using the method of perfect induction.

1a $A \cdot 0 = 0$ $\qquad$ 1b $A+1=1$

Proof for 1a: $\quad A \cdot 0 = 0$

if $A=0$ then $0 \cdot 0 = 0$

if $A=1$ then $1 \cdot 0 = 0$

Both are postulates and are true.

Proof for 1b: $\quad A+1=1$

if $A=0$ then $0+1=1$

if $A=1$ then $1+1=1$

Both are postulates and are true.

A similar proof may be obtained for the following simple theorems:

2a $A \,.\, 1 = A$ 2b $A + 0 = A$

3a $A \,.\, A = A$ 3b $A + A = A$

4a $A \,.\, \bar{A} = 0$ 4b $\bar{A} + A = 1$

Theorem 5, the "*Commutative Law*", states that the order in which the variables are placed is not significant. This was shown to be true by means of Venn diagrams in the previous section.

5a $A \,.\, B = B \,.\, A$ 5b $A + B = B + A$

The remaining theorems can all be proved by constructing a truth table which shows all the possible combinations of the values of the variables involved. Theorem 6, the "*Associative Law*" will be taken as an example to demonstrate the method.

6a $A \,.\, B \,.\, C = A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$

6b $A + B + C = A + (B + C) = (A + B) + C$

To prove: $A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$

A	B	C	$(B \,.\, C)$	$(A \,.\, B)$	$A \,.\, (B \,.\, C)$	$(A \,.\, B) \,.\, C$
0	0	0	0	0	0	0
1	0	0	0	0	0	0
0	1	0	0	0	0	0
1	1	0	0	1	0	0
0	0	1	0	0	0	0
1	0	1	0	0	0	0
0	1	1	1	0	0	0
1	1	1	1	1	1	1

The first three columns provide all the possible combinations of the variables, and the other columns are derived by appropriate combinations of the values of the variables involved. Since the last two columns are identical in all respects the proof is valid.

7a $\overline{A \,.\, B \,.\, C} = \bar{A} + \bar{B} + \bar{C}$ 7b $\overline{A + B + C} = \bar{A} \,.\, \bar{B} \,.\, \bar{C}$

In the previous section, Venn diagrams were used to verify De Morgan's theorem, which is stated above. The result is very important and should be noted carefully. It may be stated in a more general form as follows:

$$\bar{f}\ (A,B,C,\ \text{AND OR}) = f\ (\bar{A},\bar{B},\bar{C},\ \text{OR AND})$$

The symbol $\bar{f}$ denotes that the complement of the function is required, and applying the normal rules, AND is exchanged for OR and each literal is complemented. For example:

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

Two more theorems are:

8a $A \,.\, (A+B) = A$ 8b $A+(A \,.\, B) = A$

9a $(A+B) \,.\, (A+\bar{B}) = A$ 9b $(A \,.\, B)+(A \,.\, \bar{B}) = A$

The "*Distributive Law*" given in theorem 10, below, is also very important, and requires careful attention:

10a $A \,.\, B+A \,.\, C = A \,.\, (B+C)$

10b $(A+B) \,.\, (A+C) = A+B \,.\, C$

In 10a the procedure is similar to the normal algebraic process of factoring, but 10b shows that the operations of AND and OR do not have precisely the same meaning as multiplication and addition in classical algebra, although the functions are frequently described by the terms "product" and "sum" respectively. Furthermore an expression such as $A \,.\, B+C \,.\, D$ is termed a "sum of products", while $(A+B) \,.\, (C+D)$ is falled a "product of sums".

In Boolean algebra there are four forms of expression that are of particular interest:–

Expanded sum of products.
Minimum sum of products.
Expanded product of sums.
Minimum product of sums.

The expanded forms are often useful for the analysis and simplification of Boolean functions and their associated circuitry, while the minimum forms are of interest because these are most frequently used as the actual basis for the required circuits. In practice, however, the minimum may be defined in several different ways, largely dependent upon the type of logical circuit being used. For instance the emphasis in diode circuits might be on the least number of logic inputs, but in transistor circuits the minimum number of terms might be more desirable. If the Boolean function is complex it is not always easy to determine a minimum by algebraic manipulation, and other methods of simplification must be adopted.

In order to obtain the expanded sum of products, theorem 9b is used and all the possible combinations of the missing variables are supplied to each product as shown in the example below.

$$\begin{aligned} A.\bar{B}.C + A.\bar{C}.D &= A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.\bar{C}.D \\ &= A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D + A.\bar{B}.\bar{C}.D \end{aligned}$$

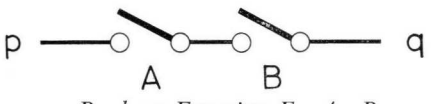
On the first line, $A.\bar{B}.C$ expands into two terms when the missing variable is supplied both as D and $\bar{D}$, and in the same way two further terms are formed from $A.\bar{C}.D$ by adding B and $\bar{B}$.

A similar operation is carried out for the expanded product of sums, in this case all the possible combinations of the missing variables being added to each sum. Thus the expansion of $(A+\bar{C}) \,.\, (B+\bar{C})$ becomes $(A+B+\bar{C}) \,.\, (A+\bar{B}+\bar{C}) \,.\, (A+B+\bar{C}) \,.\, (\bar{A}+B+\bar{C})$

2.3. CONTACT SWITCHING CIRCUITS

In any logic switching circuit the particular conditions representing 0 and 1 must be clearly defined since the assignment of values has a profound effect, not only on the Boolean function representing a given circuit but also on the circuit associated with a particular function. The choice does not however affect the algebra or its manipulations, but merely the interpretation we choose to place on the variables and their values.

In general, the Boolean function describing a circuit sets down conditions defining the transmission capability of the system. The normal convention adopted for a contact switching circuit is that a closed circuit is represented by 1 and an open circuit by 0. Under these conditions, two switches wired in series will perform the AND function as shown in fig. 2 and it may be seen that the maximum transmission is obtained between p and q when the circuit is closed.



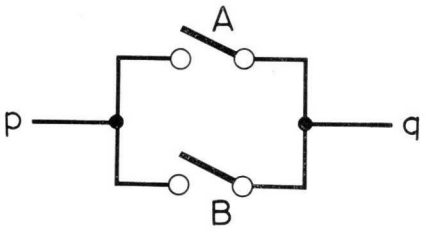
Boolean Function $F = A \cdot B$
Fig. 2. AND Circuit

Truth Table

A	B	Circuit State
0	0	0 = Open
1	0	0 = Open
0	1	0 = Open
1	1	1 = Closed

The action of the circuit is described by the Truth Table, and clearly maximum transmission implies a low impedance path between p and q permitting an easy flow of current through the circuit.

Fig. 3 shows the parallel arrangement of switches required for the OR function.

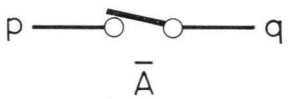


Boolean Function $F = A + B$
Fig. 3. OR Circuit

Truth Table

A	B	Circuit State
0	0	0 = Open
1	0	1 = Closed
0	1	1 = Closed
1	1	1 = Closed

The operation of complementing means that the circuit is closed when the switch is NOT operated as illustrated in fig. 4.



Truth Table

A	Ā	Circuit State
0	1	1 = Closed
1	0	0 = Open

Fig. 4. NOT Circuit

It is now possible to interpret the theorems of Boolean algebra in terms of switching circuits, as the following examples show.

Theorem 3 is shown in fig. 5.

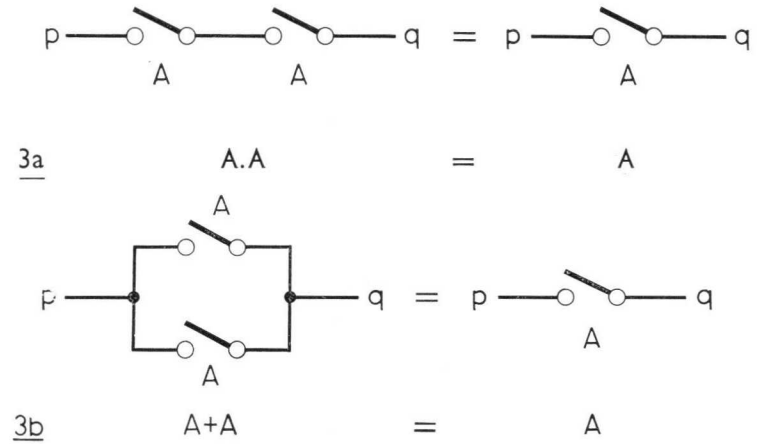


Fig. 5. Circuit for Theorem 3

From fig. 5 it may be seen that Theorem 3 states that identical switches wired in series or in parallel can be replaced by a single switch. It should be emphasised that the literals used in any theorem can represent single variables or complex expressions. For instance, Theorem 3 may be employed to simplify the following expression:

$$\begin{aligned}(\bar{A}.C.C+\bar{B}).(\bar{A}.C+\bar{B}+\bar{B}) &= (\bar{A}.C+\bar{B}).(\bar{A}.C+\bar{B}+\bar{B}) \\ &= (\bar{A}.C+\bar{B}).(\bar{A}.C+\bar{B}) \\ &= (\bar{A}.C+\bar{B})\end{aligned}$$

Theorem 8 is illustrated in fig. 6.

Theorem 8a states that any circuit consisting of a switch wired in series with a parallel circuit containing the same switch, may be replaced by a circuit consisting of the switch alone. Theorem 8b states that any circuit consisting of a switch wired in parallel with a series circuit containing the same switch, may be replaced by a circuit consisting of the switch alone.

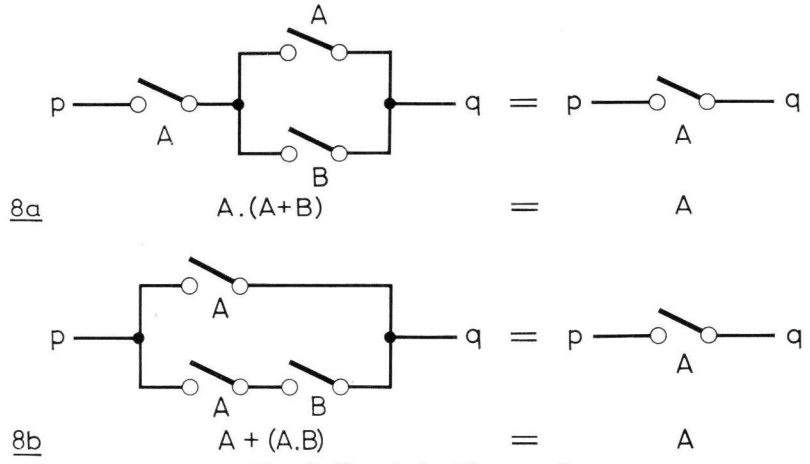


Fig. 6. Circuit for Theorem 8

If the logic convention employed is inverted, that is to say a closed circuit is represented by 0 and an open circuit by 1, then for a given circuit arrangement the AND and OR functions are interchanged. For example, the parallel arrangement of switches in fig. 7 is required for the AND function, while a series circuit is necessary for the OR function as shown in fig. 8.

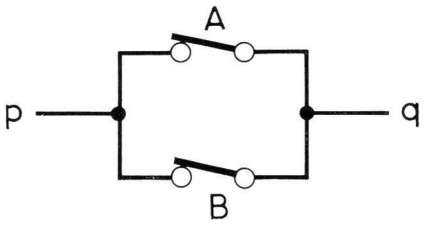


Fig. 7. AND Circuit – Inverted Logic

Truth Table

A	B	Circuit State
0	0	0 = Closed
1	0	0 = Closed
0	1	0 = Closed
1	1	1 = Open

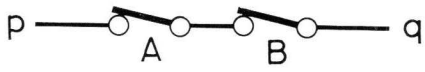


Fig. 8. OR Circuit – Inverted Logic

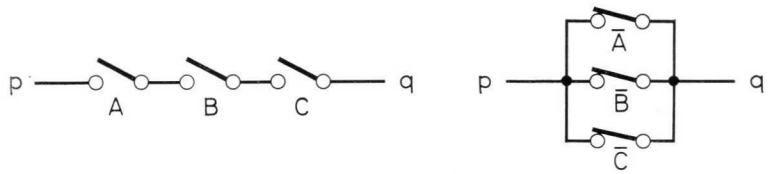
Truth Table

A	B	Circuit State
0	0	0 = Closed
1	0	1 = Open
0	1	1 = Open
1	1	1 = Open

From the Truth Tables given it is clear that in this case minimum transmission or maximum hindrance is obtained when the circuit is open, and this implies a high impedance path between p and q, which does not permit an easy flow of current through the circuit.

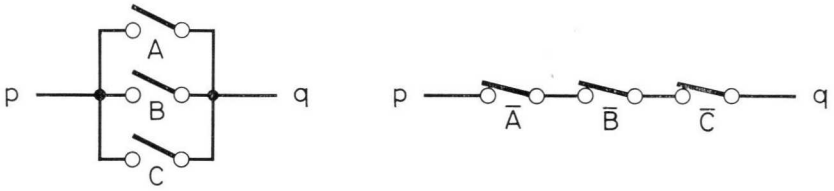
A similar result can be obtained without inverting the logic if the Boolean function is complemented, as illustrated by De Morgan's Theorem in fig. 9 below.

Thus the complement of is



7a $\overline{A.B.C} = \bar{A}+\bar{B}+\bar{C}$

And the complement of is



7b $\overline{A+B+C} = \bar{A}.\bar{B}.\bar{C}$

Fig. 9. Circuit for de Morgan's Theorem

Worked Examples

Example 1. Simplify $(A+\bar{B}+C)\,.\,(A+\bar{B}+\bar{C})\,.\,(A+B+C)$ and express as a sum of products.

Let

$$S=(A+\bar{B}+C)\,.\,(A+\bar{B}+\bar{C})\,.\,(A+B+C)$$

Then by De Morgan's Theorem 7a

$$\begin{aligned}\bar{S} &= \bar{A}.B.\bar{C}+\bar{A}.B.C+\bar{A}.\bar{B}.\bar{C}\\ &= \bar{A}.B(\bar{C}+C)+\bar{A}.\bar{B}.\bar{C} \quad \text{by theorem 10a}\\ &= \bar{A}.B.1+\bar{A}.\bar{B}.\bar{C} \quad \text{since } (\bar{C}+C)=1 \text{ by theorem 4b}\\ &= \bar{A}.B+\bar{A}.\bar{B}.\bar{C} \quad \text{since } \bar{A}.B.1=\bar{A}.B \text{ by theorem 2a}\end{aligned}$$

Now

$$\begin{aligned}S &= \text{the complement of } \bar{S}\\ &= \overline{\bar{A}.B+\bar{A}.\bar{B}.\bar{C}}\\ &= (A+\bar{B})\,.\,(A+B+C) \quad \text{by theorem 7b}\\ &= A+\bar{B}(B+C) \quad \text{by theorem 10b}\\ &= A+\bar{B}.B+\bar{B}.C \quad \text{but } \bar{B}\,.\,B=0 \text{ by theorem 4a}\end{aligned}$$

Thus $S=A+\bar{B}.C$

The solution given above is deliberately arranged to include as many theorems as possible.

An alternative solution is as follows:

$$\begin{aligned} S &= (A+\bar{B}+C)\,.\,(A+\bar{B}+\bar{C})\,.\,(A+B+C) \\ &= (A+\bar{B})+C\ .\ (A+\bar{B})+\bar{C}\ .\ (A+B+C) \\ &= (A+\bar{B})\,.\,(A+B+C) \quad \text{by theorem 9a} \\ &= A+\bar{B}.C \quad \text{by theorems 10b and 4a as before.} \end{aligned}$$

Thus $S = A+\bar{B}.C$

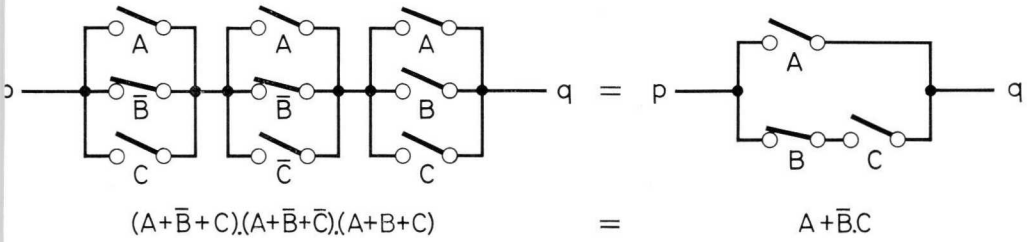


Fig. 10. Circuit for Example 1

Example 2. Simplify $B.\bar{C}.A+B.\bar{B}.C+B.\bar{C}.\bar{C}+\bar{B}.C+\bar{A}.\bar{B}.C+A.C.\bar{C}$ and express as a product of sums.

Let

$$S = B.\bar{C}.A+B.\bar{B}.C+B.\bar{C}.\bar{C}+\bar{B}.C+\bar{A}.\bar{B}.C+A.C.\bar{C}$$

Now

$$B\,.\,\bar{B} \text{ and } C\,.\,\bar{C}=0 \quad \text{by theorem 4a}$$

Therefore

$$B.\bar{B}.C = 0.C = 0 \quad \text{and}$$
$$A.C.\bar{C} = A.0 = 0 \quad \text{by theorem 1a}$$

Also

$$\bar{C}.\bar{C} = \bar{C} \quad \text{by theorem 3a}$$

Thus

$$S = B.\bar{C}.A+B.\bar{C}+\bar{B}.C+\bar{A}.\bar{B}.C$$

But

$$B.\bar{C}+B.\bar{C}.A = B\,.\,\bar{C} \text{ and}$$
$$\bar{B}.C+\bar{B}.C.\bar{A} = \bar{B}\,.\,C \quad \text{by theorem 8b}$$
$$\textit{Thus } S = B.\bar{C}+\bar{B}.C$$

The function is required in the form of a product of sums, and this may be achieved as follows:

Add the terms $B\,.\,\bar{B}$ and $C\,.\,\bar{C}$ to the expression. Since both these are equivalent to zero, the function will not be changed.

Then

$$\begin{aligned} S &= B.\bar{C}+B.\bar{B}+\bar{B}.C+C.\bar{C} \\ &= B(\bar{B}+\bar{C})+C(\bar{B}+\bar{C}) \end{aligned}$$
$$\textit{Thus } S = (B+C)\,.\,(\bar{B}+\bar{C})$$

Alternatively, the product of sums can be obtained by employing De Morgan's Theorem.
Since

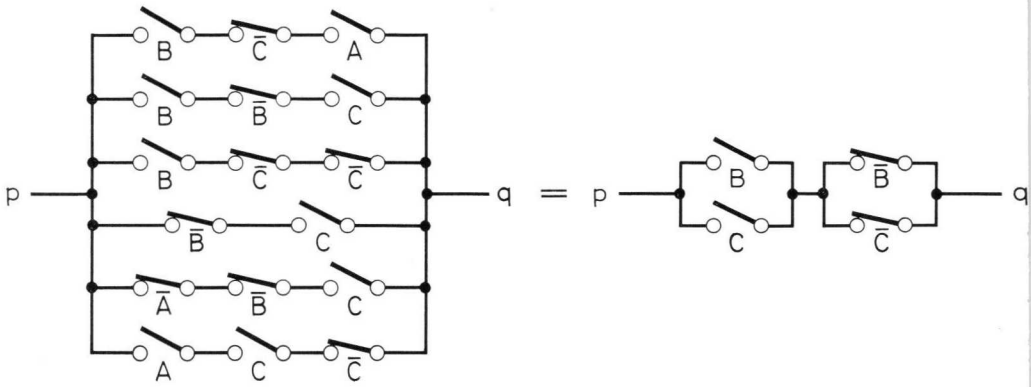
$$S = \overline{B.\bar{C} + \bar{B}.C}$$

$$\bar{S} = B.\bar{C} + \bar{B}.C$$
$$= (\bar{B} + C).(B + \bar{C})$$
$$= \bar{B}.B + \bar{B}.\bar{C} + B.C + C.\bar{C}$$
$$= \bar{B}.\bar{C} + B.C$$

Now

$$S = \text{the complement of } \bar{S}$$
$$= \overline{\bar{B}.\bar{C} + B.C}$$

Thus $S = (B + C).(\bar{B} + \bar{C})$ *as before.*



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

Fig. 11. Circuit for Example 2

CHAPTER 3

BASIC LOGIC CIRCUITS

CHAPTER CONTENTS

3.1. DIODE GATES

Introduction

Although logic circuits may be constructed from any device which has two stable states, modern computers invariably use semiconductors rather than electro-mechanical relays or thermionic valves. The choice of solid state components leads to a smaller physical size, longer life, high speed, generally better reliability and a comparatively low power requirement. For convenience in manufacture, a few types of basic elements are mass-produced as standard packages, and these form the logical building blocks for a given system, whether it be a computer or a complicated switching circuit to control an industrial process.

The basic elements are usually called "gates" and fall broadly into two main classes:

(a) passive networks providing an output power less than the input needed to drive them. Semiconductor diodes are widely used in this category.

or (b) active networks in which transistors perform logic functions and also give power amplification.

In both cases, the semiconductors are employed as high speed electronic switches, but it should be noted that the switching action is not perfect. For instance, in the conducting state a diode has a small forward resistance, and when it is cut off, a quite high back resistance, but not infinitely high. In general this leads to small fluctuations in the output potentials, which are obviously undesirable, but despite these limitations circuits can be designed to function satisfactorily. It is important to realise that in electronic circuits the values 0 and 1 are usually assigned to two different levels of potential, and a "signal" is normally taken to mean the particular potential used to represent a 1.

Circuit Descriptions

Before describing the action of the basic diode gating circuits, the conditions under which a diode will conduct should be clearly understood. Fig. 12 shows the circuit symbol for a semiconductor diode, and to avoid any possible ambiguity the anode and cathode are marked on the diagram. In practice, the manufacturer usually indicates the polarity by means of a coloured spot or band near the cathode terminal.

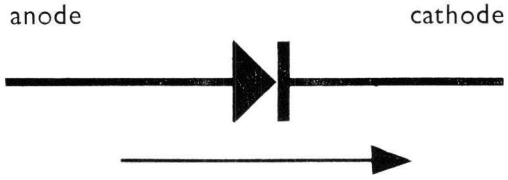
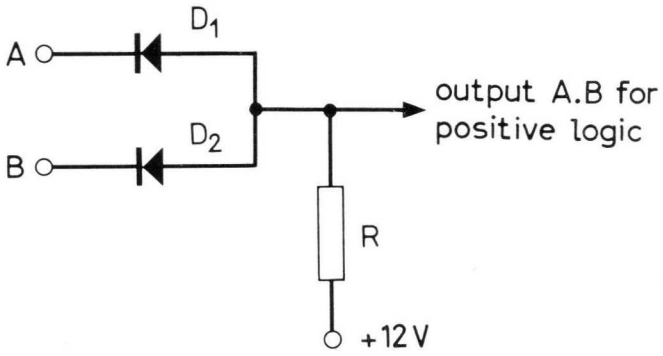
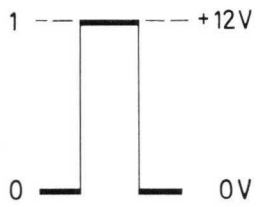


Fig. 12. Diode Circuit Symbol

Conventional current flows in the direction of the arrow when the anode is at a potential positive with respect to the cathode, or conversely when the cathode is negative with respect to the anode. In this condition the diode is said to be "forward" biased, and its resistance is low, thus allowing an easy flow of current. If the anode is negative with respect to the cathode (or the cathode positive with respect to the anode), the diode is "reverse" biased and its resistance becomes very high, and hence the reverse current is small.

The basic diode gate is shown in fig. 13 and the action of the circuit is illustrated by the Truth Tables. It may be seen that the circuit functions as an AND gate when the higher voltage level represents a 1. Since the



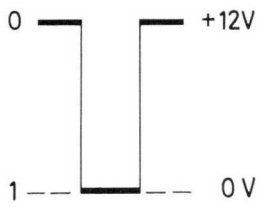


Positive Logic

Truth Table

A	B	Output	Diodes
0	0	0	Both conduct
1	0	0	D2 conducts
0	1	0	D1 conducts
1	1	1	Both cut off

AND Gate



Negative Logic

Truth Table

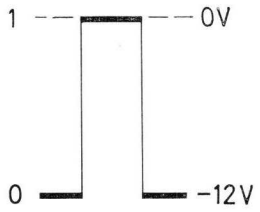
A	B	Output	Diodes
0	0	0	Both cut off
1	0	1	D1 conducts
0	1	1	D2 conducts
1	1	1	Both conduct

OR Gate

Fig. 13. Basic Diode Gate (Positive AND)

signals are then indicated by positive going pulses, this convention is known as *positive logic*. If the logic is inverted the circuit functions as an OR gate.

The signals could be applied to the diode gate using the following potentials.



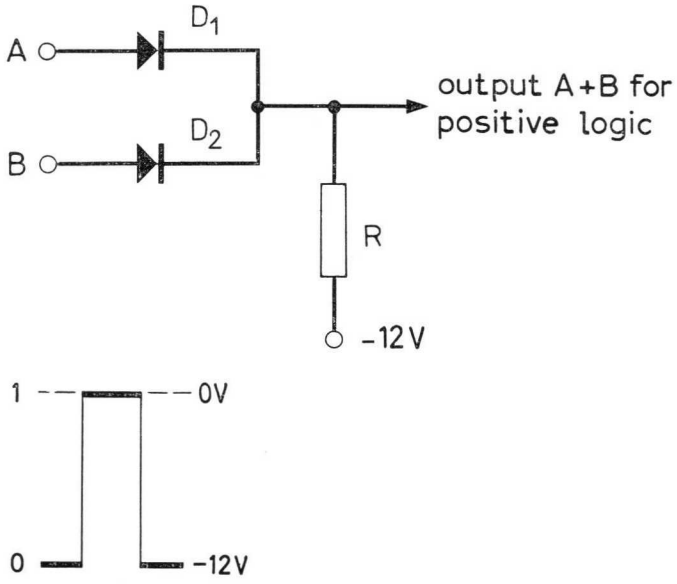
In this case, however, although the circuit still performs as an AND gate for positive logic (or an OR gate for negative logic), the diodes are never cut off simultaneously, assuming there is no loading effect due to any following circuit. Clearly when the output potential is at -12V, the current flowing through R is a maximum, and this value is approximately halved when the output potential is at zero volts.

If the diodes are reversed and the supply voltage changed to -12 volts, the circuit of fig. 14 is obtained, and in this case positive logic results in an OR gate.

If the signal potential is $+12$V and 0 is represented by zero volts, the circuit will still perform as a positive logic OR gate (or negative logic AND).

Under these conditions, the diodes are never cut off simultaneously, assuming there is no loading effect due to any following circuit, and the circuit action is similar to that noted previously for the positive logic AND gate.

In both circuits (figs. 13 and 14), the value of R is chosen to be large compared to the diode forward resistance, but small in relation to the back resistance. Thus, although the volt drop across the forward resistance of a conducting diode is small, the output voltage must be less than that applied at the input. Furthermore, if both diodes are conducting then the effective forward resistance is reduced, allowing a slight increase in the output potential. Clearly this leads to small fluctuations in the output signal which are undesirable. In addition, however, the signal is degraded each time it passes through a diode gate and it becomes necessary to include transistor amplifiers at salient points in a long logical chain of passive elements, in order to restore the original level.



Positive Logic

Truth Table

A	B	Output	Diodes
0	0	0	Both cut off
1	0	1	D1 conducts
0	1	1	D2 conducts
1	1	1	Both conduct

OR Gate

Negative Logic

Truth Table

A	B	Output	Diodes
0	0	0	Both conduct
1	0	0	D2 conducts
0	1	0	D1 conducts
1	1	1	Both cut off

AND Gate

Fig. 14. Basic Diode Gate (Positive OR)

So far the AND and OR gates described have been shown with two inputs only, but more diodes may be added so that several signals can be gated in the same circuit. However a practical limit is reached at five or six diodes for any one gate, but of course if the number of inputs demanded exceeds this practical limit, the circuit must be rearranged to perform the gating in several smaller groups.

3.2. TRANSISTOR GATES

The symbols commonly used for p-n-p and n-p-n transistors are shown in fig. 15, from which it may be seen that they are complementary to each other, since both the polarity of the supply voltage and the direction of current flow are reversed. In the diagram the arrows on the emitter indicate the direction of conventional current flow.

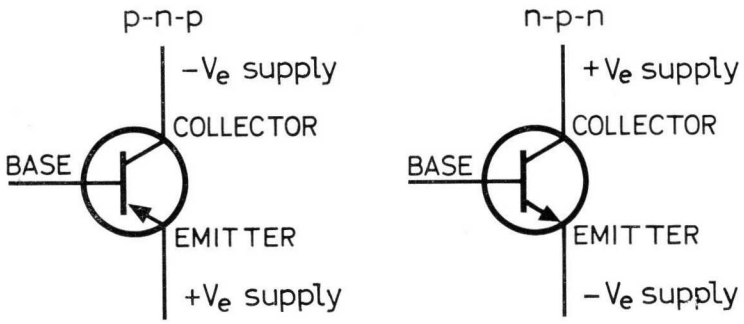


Fig. 15. Transistor Symbols

Normally a transistor conducts when the emitter-base junction is forward biased, and the collector-base junction reverse biased. For a p–n–p transistor, this occurs when the base is negative with respect to the emitter. Due to the forward bias, the emitter-base junction has a low impedance, and hence small changes of potential give rise to comparatively large current variations, and owing to the geometry of the structure more than 90% of the emitter current flows out of the collector terminal. Since the collector-base junction is reverse biased and has a high impedance, the transistor provides a power gain. A similar argument can be developed for the n–p–n transistor except that the base must be positive with respect to the emitter to allow current to flow in the collector circuit.

Since the device has three terminals, there are three possible ways of connecting the transistor in a circuit and these are: –

(a) Grounded or common base,
(b) Grounded or common emitter,
(c) Grounded or common collector.

The method of connection for each case is illustrated in fig. 16 and although p–n–p transistors are shown, similar arrangements can be used for n–p–n devices if the supply polarity is reversed.

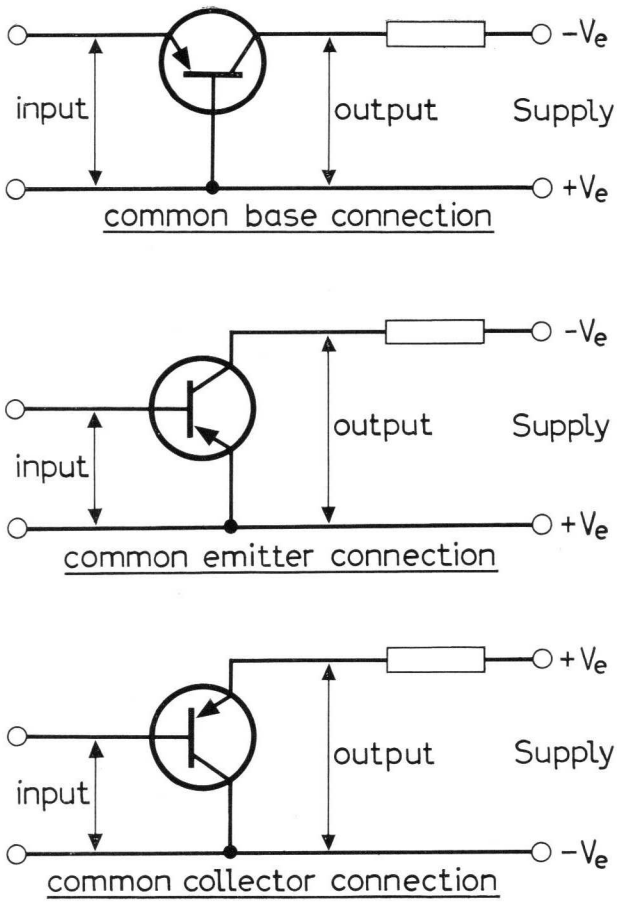


Fig. 16. Methods of connecting a p-n-p Transistor

In switching circuits, the common collector configuration is quite frequently employed, but the common emitter is by far the most widely used circuit and provides the basis for a number of transistor gates. In general, the complement of any p–n–p circuit may be obtained by reversing the polarity of the supply and substituting an n–p–n transistor, and therefore only one type need be considered in detail. A simple p–n–p inverter is shown in fig. 17.

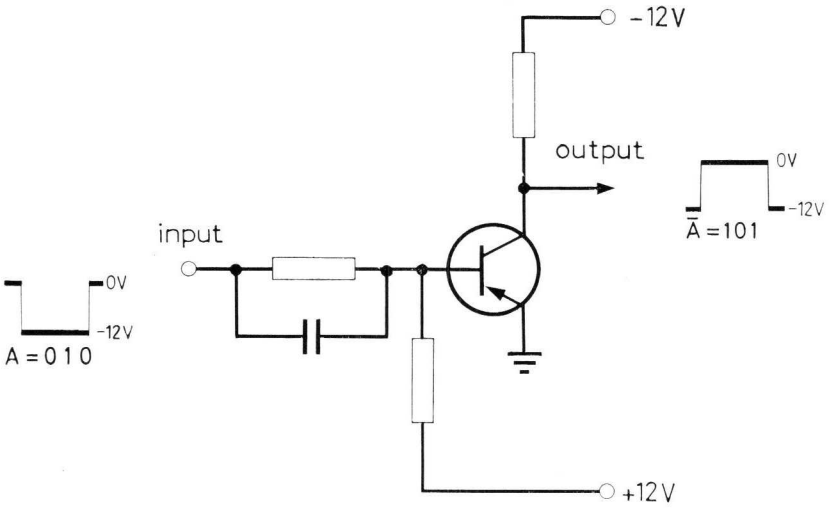


Fig. 17. Simple p-n-p Inverter

If the input is A the output is $\bar{A}$, thus the inverter provides a means of obtaining the complement of a Boolean expression. When the transistor is turned off (base positive to the emitter), the collector current is very small, and since the collector impedance is high, the output voltage is virtually that of the supply -12V. On the other hand, when the base is negative with respect to the emitter, the collector current is large and the output voltage rises. If the transistor saturates, that is to say a further increase in the base current provides little or no corresponding increase in the collector current, then the collector voltage remains constant. Typically this is about -0.2 volts with respect to the emitter and hence in this case the output voltage is almost at ground potential. If $A = 0\ 1\ 0$ and the supply (-12V) represents a 1, the output waveform will be as shown in fig. 17. In practice, it is important to ensure that the base is positive to the emitter when the transistor is turned off, in order to avoid the effects of high collector leakage current. It should also be realised that the transistor is essentially a current amplifier, which therefore requires an input current rather than an input voltage.

A common method of packaging logical elements is to combine diode gating with a transistor inverting amplifier. Consider the circuit given in fig. 18.

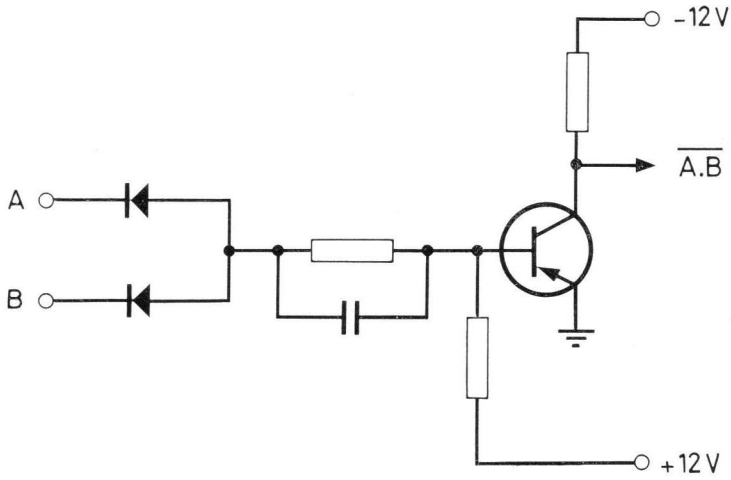


Fig. 18. Positive AND Gate with Inverter

If either input is 0 (−12V), one of the diodes is conducting, driving the base of the transistor negative. Thus the transistor saturates and the output voltage from the collector rises almost to ground potential. When both inputs are at 1, the potential applied to the base of the transistor is sufficiently positive to reduce the collector current to zero. The inverter output is therefore at −12 volts. Clearly under these conditions the combined element performs the Boolean function $\overline{A \,.\, B}$ and is therefore known as a NOT/AND or NAND gate.

For negative logic: 0 = 0 volts.
1 = −12 volts.

The circuit action is very similar to that described above except that the diode gate now provides an OR function, and hence the combined element performs the Boolean function $\overline{A+B}$ and is known as a NOT/OR or NOR gate.

A NOR gate can also be realised with the configuration shown in fig. 19 if positive logic is employed.

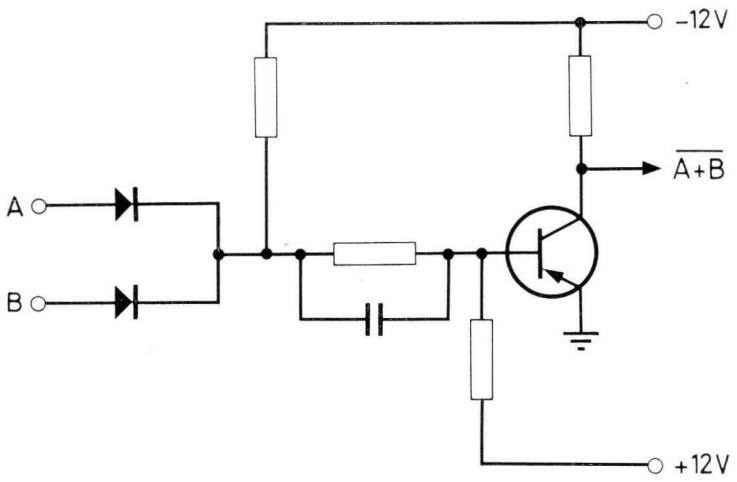


Fig. 19. Positive logic NOR Gate

If both inputs are at 0 (−12V), the diodes are reversed biased and the base of the transistor goes negative. The transistor saturates driving its collector almost to ground potential. Thus the inverter output is a 1 when both the inputs are at 0. If either or both the inputs are 1, the transistor base is driven positive and the collector current is cut off, allowing the inverter output to fall to −12 volts. Hence the combined element performs the Boolean function $\overline{A+B}$. If negative logic is applied the circuit behaves as a NAND gate.

In the design of the inverter stage, the collector load resistor should be kept to as low a value as possible, since the following stages may draw current through it when the transistor is cut off, and the volt drop across the load resistor must be as small as possible. For a similar reason the base current required for each stage can be minimised by making the base resistor comparatively large. However a compromise must be reached since the base current must be adequate under the worst loading condition likely to occur on the previous stage. The capacitor across the base input resistor is provided to improve the transient switching condition.

A considerable economy can be achieved if the input signals are applied through resistors instead of diodes, particularly when it is realised that generally the basic elements are used in large numbers for a typical installation. For obvious reasons this is known as "Resistance" logic and the circuit arrangement is shown in fig. 20.

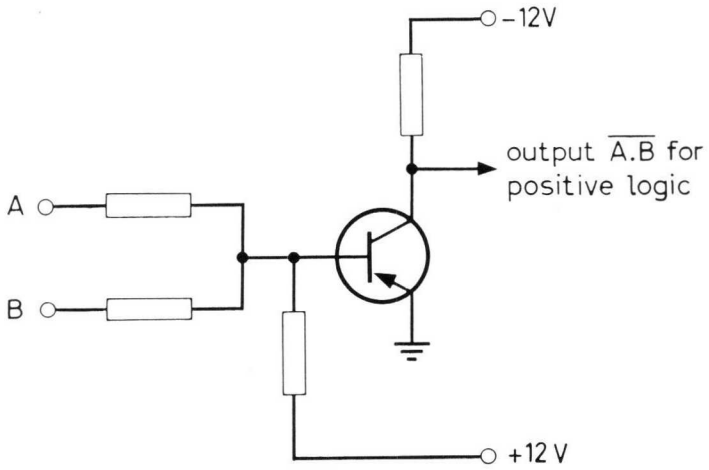


Fig. 20. Resistance logic gate

Normally, the input resistors are all equal, and may be chosen so that if any one input signal is at the lower voltage level, the base current is high enough to saturate the transistor, thus giving an output potential of almost zero volts. When more than one input is at the lower level, the base current increases driving the transistor further into the saturation region. This is a disadvantage since the switching speed is reduced under these conditions. A reverse bias is applied only when all the inputs are at ground potential, and as the transistor is cut off the output voltage falls to -12 volts. Hence the circuit is a NAND gate for positive logic, and a NOR for negative logic. The input resistors may also be chosen so that the transistor can conduct only when all the inputs are at -12V. In this case, any one input signal changing to zero volts is sufficient to apply a reverse bias to the transistor, and the gate now becomes a NOR for positive logic.

An important class of transistor logic circuits can be fabricated by direct simulation of a contact switching network, for example two transistors in series can be used as a NAND or a NOR gate depending on the logic convention employed. In general the practice is rather uneconomical since one transistor is necessary for each input signal fed to the gate as shown in fig. 21.

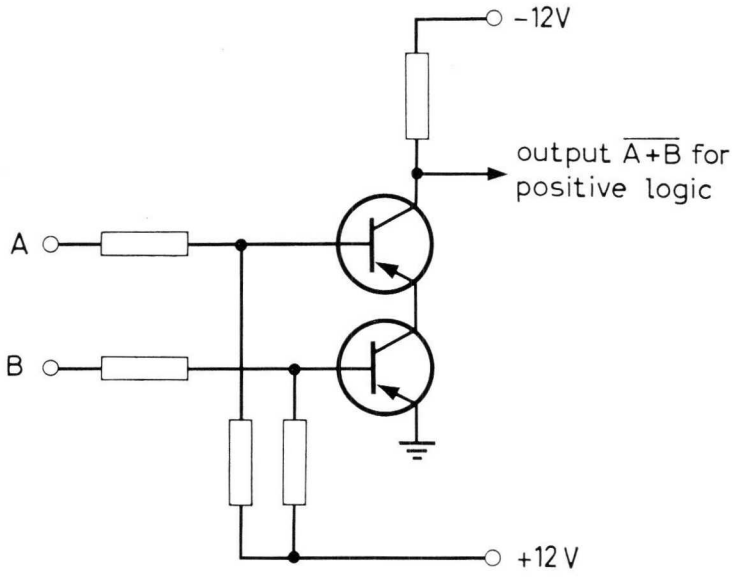


Fig. 21. Transistor series gate

Alternatively, a parallel arrangement can be used as shown in fig. 22. In this case, the output is almost at ground potential when either or both of the transistors conduct, and hence the circuit operates as a NOR gate for negative going signals. Positive logic will of course result in the circuit performing the NAND function.

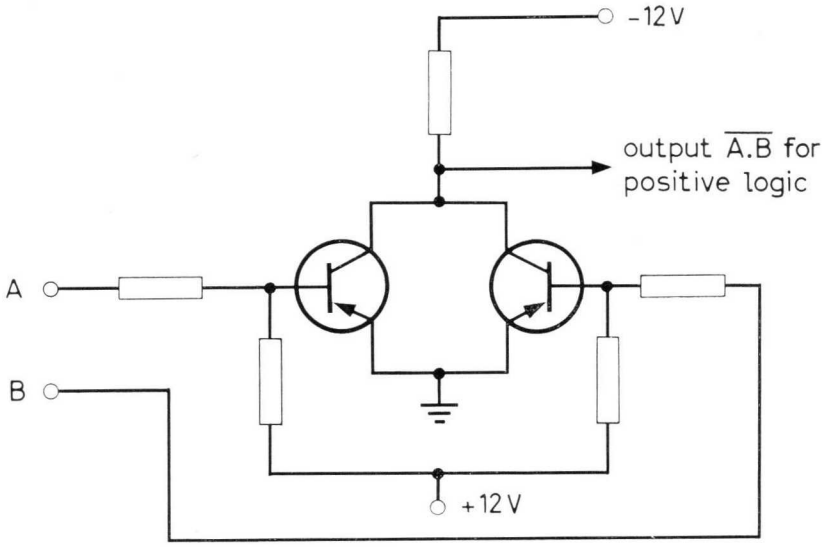


Fig. 22. Transistor parallel gate

The grounded collector, or as it is more often termed, the emitter follower, is primarily an impedance changing circuit and provides an output in phase with the input. Briefly, the main characteristics are high input impedance, low output impedance, a voltage gain of very nearly unity and a current gain similar to that obtained for the grounded emitter stage. Single emitter followers do not perform a logical function, but they can be used in multiples or combined with diode gates as shown in fig. 23.

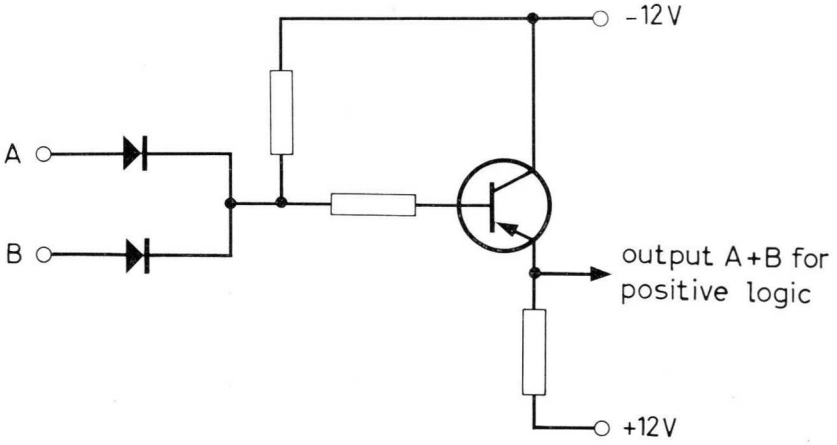


Fig. 23. Diode gate with Emitter Follower

Two emitter followers are shown in fig. 24 with the emitters connected to a common load resistor, and for positive logic the circuit provides an AND function.

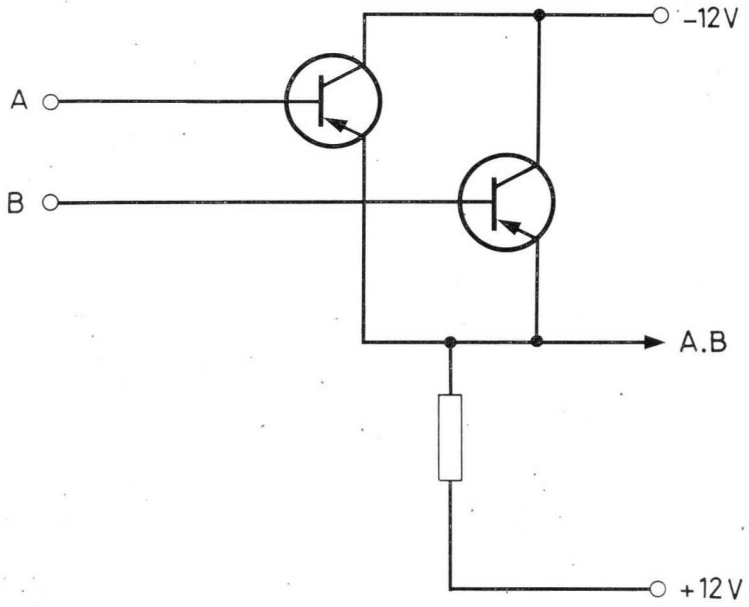


Fig. 24. Emitter Follower gate

3.3. BISTABLE CIRCUITS

Simply speaking a bistable circuit is a circuit which has two stable states, but in actual fact it is an extremely versatile device, capable of performing many of the functions necessary within a digital computer. Basically the circuit constitutes a rapid access single digit store, but it can also be used for counting and the production of control pulses. Consider the circuit given in fig. 25.

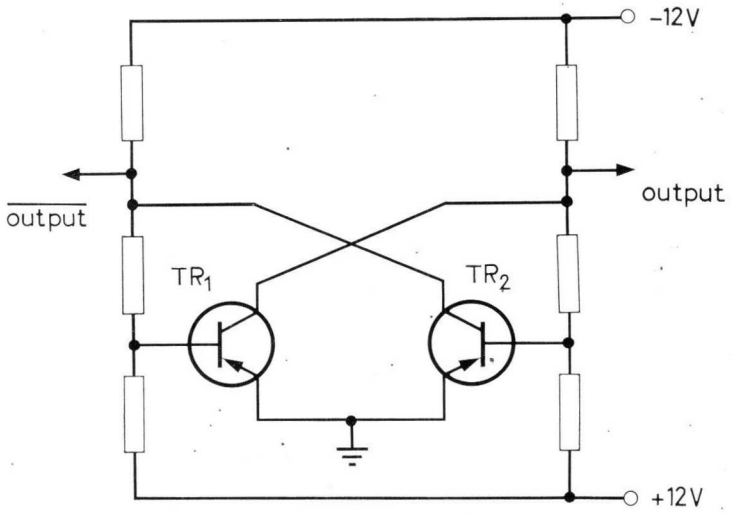


Fig. 25. Bistable Circuit

If the transistor TR_1 conducts, the potential at the output terminal is almost at zero volts, and due to the resistor network the base of TR_2 is held positive with respect to the emitter. Transistor TR_2 is therefore cut off, and the $\overline{\text{output}}$ terminal is almost at the negative supply potential. Under these conditions, the base of TR_1 remains sufficiently negative to hold this transistor on and the state is stable. A change of state can be initiated by driving TR_1 towards cut off by applying a positive pulse to its base. The initial fall in potential at the output terminal causes a proportional fall to occur at the base of TR_2, which then commences to conduct driving the $\overline{\text{output}}$ terminal and therefore the base of TR_1 positive. The action is cumulative, resulting in a second stable state being rapidly attained, with transistor TR_2 conducting and TR_1 cutoff. Clearly the $\overline{\text{output}}$ terminal is now almost at ground potential.

Since either transistor may be initially in the conducting state, the positive trigger pulse must be properly routed, and this is achieved by

means of a steering circuit. In fig. 26 the steering circuit is formed by the components R_1, R_2, C_1, and C_2 together with the diodes D_1 and D_2. The junction of C_1 and C_2 is the common input point to which positive going pulses are supplied, while d.c. levels are applied at *A* and *B* representing 0 and 1 respectively.

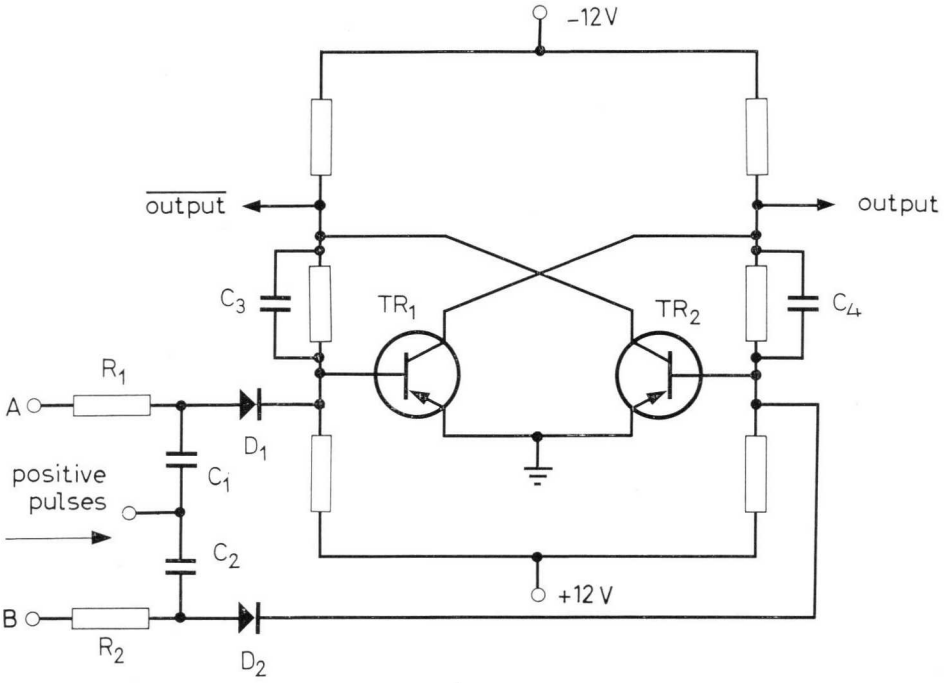


Fig. 26. Bistable shift register stage

Assume positive logic:– 1 = 0 volts
0 = –12 volts

Consider the case when the transistor TR_1 conducts, TR_2 is cut off and the output potential is at zero volts representing a 1. The d.c. potentials on the bases of the transistors only vary about earth by a small amount, the cathode of the diode D_1 being slightly negative, whereas that of D_2 is somewhat positive. Thus the bias condition of the diodes will depend largely on the complementary d.c. input levels applied at *A* and *B*. If *A* is 0 (−12V), D_1 as cutoff but since *B* is then at 1 (0V), the reverse bias on D_2 is comparatively small. When a positive pulse is applied to the junction of the capacitors C_1 and C_2, D_1 remains cut off but D_2 conducts allowing the trigger pulse to reach the base of TR_2. No change takes place since

this transistor is already cut off, and hence the output potential remains at the level representing a 1. However, if A is 1 and B is 0, the diode D_1 will conduct to transfer the positive pulse to the base of TR_1, which is then driven towards cutoff. Thus TR_2 is turned on and the bistable changes state, the output potential now being at $-12V$ representing a 0. Clearly if the trigger pulse is applied to the base of TR_1 a 0 is set, and conversely when it is transferred to the base of TR_2 a 1 is set. Since the transfer occurs when the signal is present on the appropriate terminal, the input at A can be called "Set 0", and similarly that at B "Set 1". The rate at which alternate 0 and 1 conditions can be transferred is limited by the switching speed of the transistors and also the time constant of the components R_1 and C_1 in the steering circuit. Usually R_1 equals R_2 and C_1 equals C_2. The capacitors C_3 and C_4 are provided to improve the transient switching conditions. It can readily be seen that two consecutive positive pulses must be applied in order to pass a 1 through the bistable stage, and of necessity the time required is precisely the interval between two such pulses. Since, within a computer, a central oscillator is used to generate short duration pulses to initiate and synchronise the movement of digits from one part of the machine to another, the time interval is called a "digit period". One bistable element is capable of storing one digit, and hence several stages connected in series are required to retain a complete number. The interconnection of one element to the next is accomplished by means of the steering circuit as shown in fig. 27 and the trigger pulses are applied simultaneously to all stages. When this occurs, the state of each stage is transferred to the next succeeding stage, and after several pulses the number stored may be shifted out of the register.

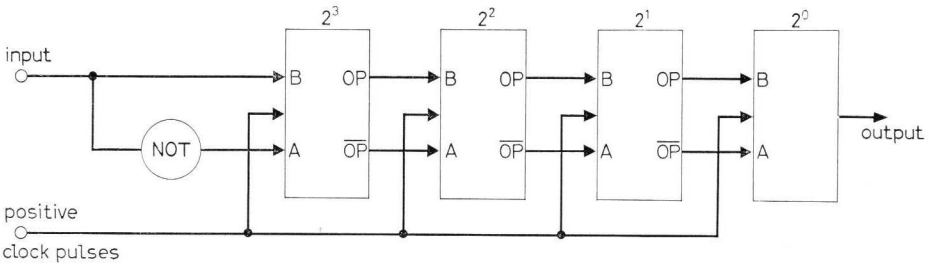


Fig. 27. Block diagram of Four-stage shift register

By means of the shifting action, the output from the final stage is a series of pulses appearing in a time sequence synchronised to the repetition speed of the central oscillator which is known as the "clock". Thus the number is represented by a train of pulses passed along the output wire, and this method of representation is known as "serial". Normally the least

significant digit is the first in the sequence, the remainder following at intervals of one digit period. Since serial operation is envisaged, only one wire is required to carry the output signal, and in order to maintain compatibility the input should also be by means of a single connection. However the steering circuit described needs dual complementary inputs, and it is therefore necessary to employ an inverter before the first stage of a shift register as illustrated in fig. 27.

From the foregoing remarks it is clear that each stage of the register introduces a time delay of one digit period, and therefore the same circuit can be used for delay purposes, the total delay being proportional to the number of bistable elements employed.

If the triggering arrangements are modified as shown in fig. 28, the circuit can be employed as a binary counter. It may be seen that the dual complementary inputs to the steering circuits are now derived from the collectors of the bistable transistors in the same stage, and the input stimulus is applied to the junction of the capacitors C_1 and C_2. Two input pulses are required for each output pulse, so that each stage operates as a "divide-by-two" circuit. It is convenient to arrange the circuit so that the count is displayed with the least significant digit on the right, as shown.

Notice that the output is used to give a positive pulse from the first bistable to drive the second stage. If both stages are initially at zero when the first pulse is received a 1 is set into the first bistable. The next pulse to arrive will set this stage to zero, and the $\overline{\text{output}}$ provides a positive pulse to set a 1 into the second bistable. The procedure is repeated, the third pulse setting stage one to 1 and the fourth resetting both stages to zero. Additional elements can of course be used to increase the capacity of the counter.

Binary counters are extremely important in a computer, not only for arithmetical processes, but also for control and timing purposes. For example, a predetermined count can be detected by a suitable gate, which then provides an output pulse to initiate some further action at precisely the right time. Alternatively, the counter can be arranged specifically to produce control waveforms, and this may be achieved by applying the inverted output to the input. Such a device is known as a "ring" counter, although the individual stages are wired as a shift register rather than a binary counter. If initially all the stages of a ring counter are set to zero, a 1 will be set into the first stage on receipt of the first clock pulse, and successive pulses finally cause all stages to be at 1. When this occurs, the next clock pulse sets a 0 into the first stage, and subsequently all the stages are at 0. In this manner a particular digit pattern circulates around the counter, and the repetitive waveforms thus produced may be used for control purposes.

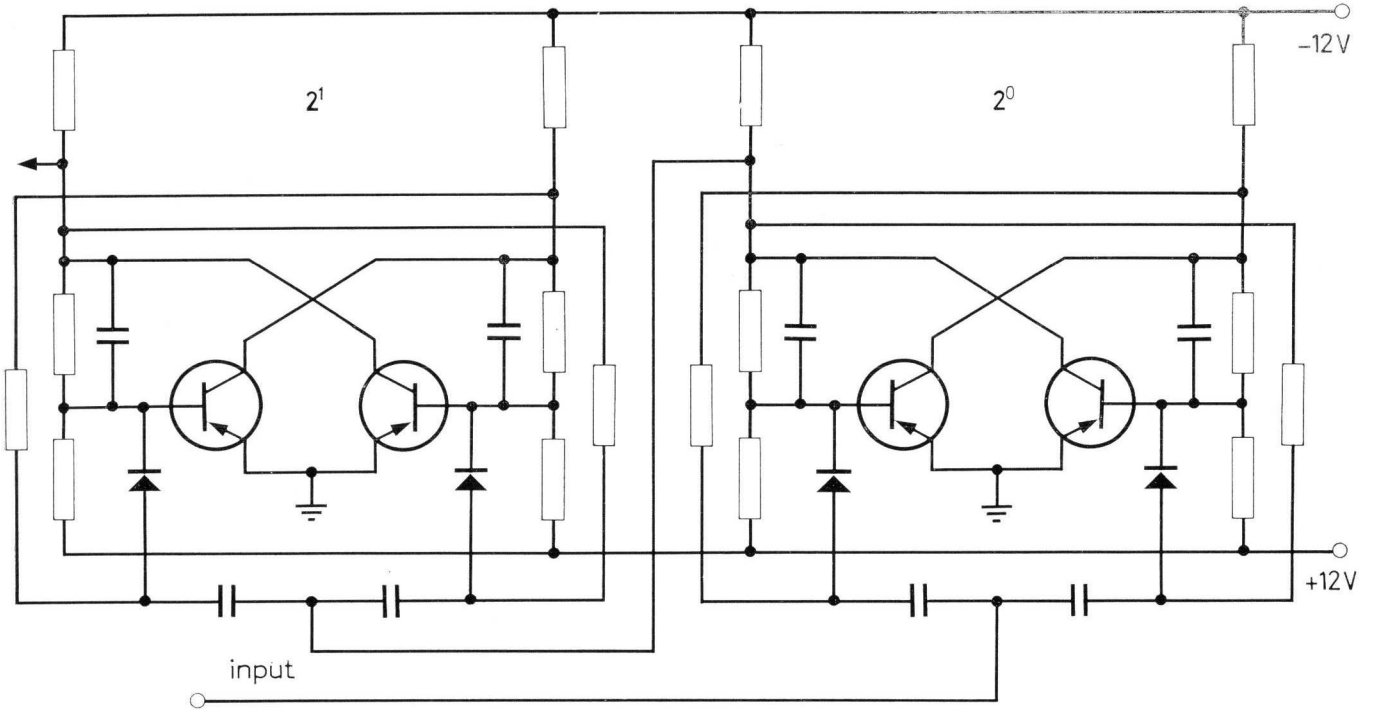


Fig. 28. Two stages of a binary counter

CHAPTER 4

LOGIC DESIGN

CHAPTER CONTENTS

The logic elements described can be used as basic building blocks in a system, and it is now a matter of logic design to establish the particular configurations required.

Clearly it is essential for a computer to perform arithmetic, and if the numbers are represented serially, the digits will appear in a time sequence. Hence the process of addition, for example, is carried out just as one would add two numbers on paper, commencing with the least significant digits and dealing with pairs of digits at successive time intervals. If a pair of digits are represented by A and B respectively and their sum by S, all the possible values for A, B, and S can be tabulated in a Truth Table as shown below.

A	B	S	
0	0	0	
1	0	1	Boolean Function
0	1	1	$S = A$ OR B AND NOT (A AND B)
1	1	0	$= (A+B) \,.\, \overline{A.B}$

Truth Table for addition of A and B

An examination of the Truth Table reveals that S is at 1 when A or B is 1, but not when both A and B are 1. Thus the Boolean function for S may be written:

$$S = (A+B) \,.\, \overline{A.B}$$

It should be noted that the Boolean function is formed by considering only the input signals, that is to say the input conditions representing 1 in the Truth Table. When the Boolean expression is obtained, it must be translated into a logical arrangement of elements which will provide the required function, in this case the sum of the two digits A and B.

$$\text{Let } S = X \,.\, Y \text{ where } X = (A+B) \text{ and } Y = \overline{A \,.\, B}$$

Clearly the sum S will be the output from an AND gate whose inputs are X and Y, and in turn these will be the outputs from elements forming $(A+B)$ and $\overline{A \,.\, B}$ respectively. Thus X will be the output from an OR gate whose inputs are A and B, and in a similar manner a NAND element is necessary to obtain Y. Rather than draw the complete circuit, it is convenient to produce a logic diagram using symbols for the gating elements. Since many different symbols are employed in current literature and textbooks, and to avoid any possible confusion, in this booklet all logic elements will be denoted by a circle inscribed with the appropriate function as shown in fig. 29.

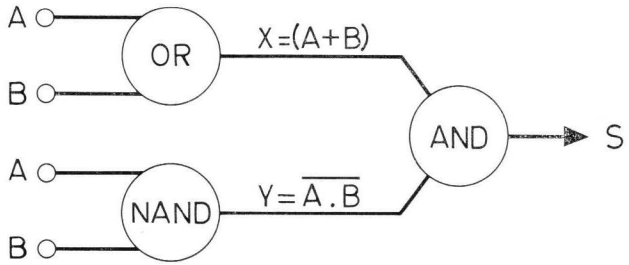


Fig. 29. Logic Diagram for the sum of A and B

An alternative arrangement can be obtained by manipulation of the Boolean expression as follows:

$$
\begin{aligned}
S &= (A+B)\,.\,\overline{A.B} \\
&= (A+B)\,.\,(\bar{A}+\bar{B}) \quad \text{by De Morgan's Theorem 7a} \\
&= A.\bar{A}+A.\bar{B}+\bar{A}.B+B.\bar{B} \\
&= A.\bar{B}+\bar{A}.B \quad \text{since both } A.\bar{A} \text{ and } B.\bar{B} = 0 \\
\textit{Thus } S &= A.\bar{B}+\bar{A}.B
\end{aligned}
$$

If $C = A.\bar{B}$ and $D = \bar{A}.B$ then $S = C+D$ and the logic diagram given in fig. 30 may be deduced from these relationships.

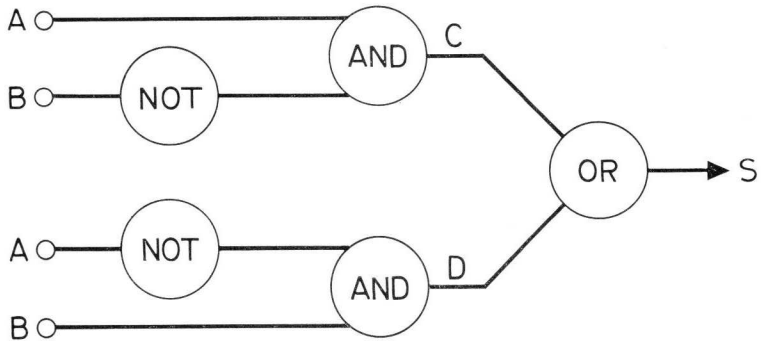


Fig. 30. Logic Diagram for $S = A.\bar{B}+\bar{A}.B$

Inverters are necessary if $\bar{A}$ and $\bar{B}$ are not directly available. Notice that fig. 30 provides a solution involving the basic functions AND, OR and NOT, whereas fig. 29 uses AND, OR and NAND elements.

The logical statement made by both arrangements is "*A* OR *B* AND NOT (*A* AND *B*)" and thus the function is known as an 'Exclusive OR'.

In contrast, the basic OR function which states "*A* OR *B* OR both" is often called an "Inclusive OR".

The summation of two digits in any scale can produce a carry, which must be taken into account for the process to be accurate. Employing the rules for binary addition, the complete Truth Table is given below.

A	*B*	Sum	Carry
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth Table for addition of A and B with carry

An examination of the Truth Table shows that the carry is available when coincidence occurs between *A* and *B*, and hence an AND gate is required to provide the carry signal. Under these conditions, the arrangement is called a "Half Adder" because it can deal with the summation of any pair of binary digits, but not with the complete addition of two numbers. For the Half Adder the expressions for the sum and carry must be satisfied simultaneously.

$$\text{Sum } S = (A+B) \,.\, \overline{A \,.\, B}$$
$$\text{Carry } C = A \,.\, B$$

Fig. 31 shows one method of achieving a Half Adder.

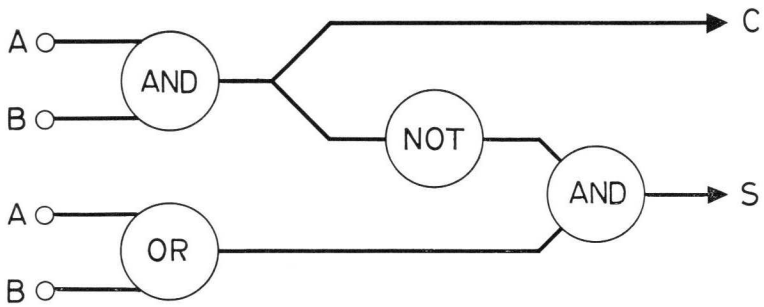


Fig. 31. Half Adder

Of course many other configurations are possible, for example a Half Adder can be constructed using only NAND gates, or alternatively only NOR gates, since all the basic logic functions can be simulated by

either of these elements. This versatility makes either the NAND or the NOR gate eminently suitable as a standard logic building block. Some manipulation of the Boolean expression is necessary, and a NAND solution for the Half Adder can be obtained as follows:

$$S = (A+B) \,.\, \overline{A \,.\, B}$$

Now $\overline{A \,.\, B} = (\bar{A}+\bar{B})$

Therefore $(A+B) = \overline{\bar{A} \,.\, \bar{B}}$

Hence $S = \overline{\bar{A} \,.\, \bar{B}} \,.\, \overline{A \,.\, B}$

$$\bar{S} = \overline{\overline{\bar{A} \,.\, \bar{B}} \,.\, \overline{A \,.\, B}}$$

$$S = \overline{\overline{\bar{A} \,.\, \bar{B}} \,.\, \overline{A \,.\, B}}$$

Now $C = A \,.\, B$

$$\bar{C} = \overline{A \,.\, B}$$

Therefore $C = \overline{\overline{A \,.\, B}}$

Six NAND gates are necessary for the sum circuit, one of which may be included in the carry signal path since the term $\overline{A \,.\, B}$ is common, and hence a total of seven elements is required altogether. Notice that in both the sum and carry circuits, a double inversion is necessary to retain the AND function. While this is unavoidable in the case of the carry, it does indicate some redundancy in the sum circuit, and a solution involving fewer elements should be possible.

Consider the following:

$$\begin{aligned} S &= (A+B) \,.\, \overline{A \,.\, B} \\ &= A \,.\, \bar{B} + \bar{A} \,.\, B \quad \text{as shown previously} \\ &= A \,.\, \bar{A} + A \,.\, \bar{B} + \bar{A} \,.\, B + B \,.\, \bar{B} \quad \text{since both } A \,.\, \bar{A} \text{ and } B \,.\, \bar{B} = 0 \\ &= A \,.\, (\bar{A}+\bar{B}) + B \,.\, (\bar{A}+\bar{B}) \\ &= A \,.\, (\overline{A \,.\, B}) + B \,.\, (\overline{A \,.\, B}) \quad \text{by De Morgan's Theorem} \end{aligned}$$

Therefore

$$\begin{aligned} \bar{S} &= \overline{A \,.\, (\overline{A \,.\, B}) + B \,.\, (\overline{A \,.\, B})} \\ &= \overline{A \,.\, (\overline{A \,.\, B})} \,.\, \overline{B \,.\, (\overline{A \,.\, B})} \\ S &= \bar{\bar{S}} \\ &= \overline{\overline{A \,.\, (\overline{A \,.\, B})} \,.\, \overline{B \,.\, (\overline{A \,.\, B})}} \end{aligned}$$

Again

$$C = \overline{\overline{A \,.\, B}}$$

Since each NAND element provides one inversion, some indication of the total number of gates required is given by the number of "bars" in the

Boolean expression, but in this case only one element is necessary for the term $\overline{A \,.\, B}$ which appears several times. The circuit is given in fig. 32 and it may be seen that only five gates are needed.

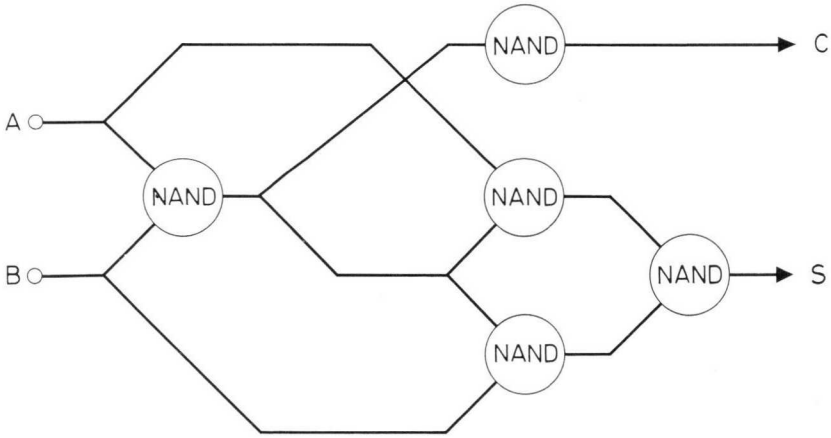


Fig. 32. Half Adder using NAND elements.

It may be shown that the serial addition of two binary numbers requires two half adders and a delay element arranged so that any carry produced from either half adder is taken into account correctly in the next significant digit place. Such a device is called a "Full Adder". It can also be shown that fundamentally all the processes of binary arithmetic can be carried out by means of addition or subtraction. As an alternative, therefore, the arithmetic unit of the computer could just as easily be designed to perform subtraction as the basic operation. The Truth Table for subtracting B from A is given below, where D is the difference between A and B, and C represents a "borrow" rather than a carry signal.

A	B	D	C	
0	0	0	0	Boolean Function
0	1	1	1	$D = A$ OR B AND NOT (A AND B)
1	0	1	0	$= (A+B) \,.\, \overline{A \,.\, B}$
1	1	0	0	$C = B$ AND NOT $A = B \,.\, \bar{A}$

Truth Table for Subtraction.

An examination of the Truth Table shows that the function required for D is precisely the same as that previously obtained for the sum circuit of the half adder, but the borrow function is different. An arrangement

using the basic connectives to achieve the functions given above is relatively straightforward, but some manipulation is necessary to obtain a solution involving only one type of element. It is proposed to find a solution using NOR logic:–

$$D = (A+B) \,.\, \overline{A \,.\, B}$$

Now

$$\overline{A \,.\, B} = \bar{A} + \bar{B}$$

Therefore

$$A \,.\, B = \overline{\bar{A} + \bar{B}}$$

Thus

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

The circuit for this arrangement is given in fig. 33.

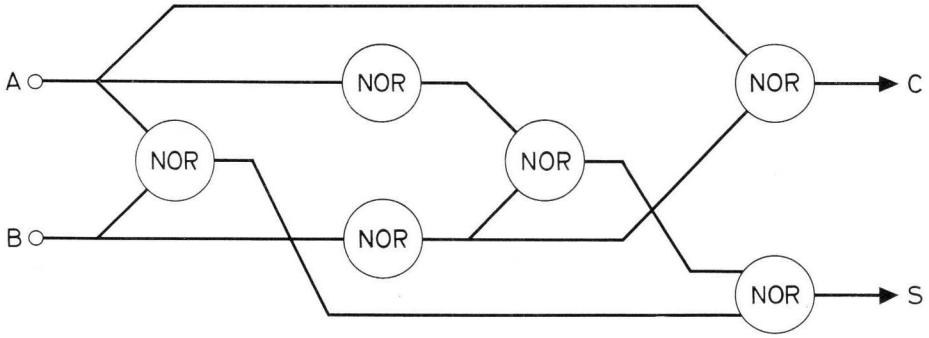


Fig. 33. Half Subtractor using NOR *elements*

Conclusion

Although the theory given is considered more than adequate to cover the work on basic logic, this booklet is not intended to be a complete treatise on the entire field of digital computing. For more advanced study, a selected bibliography is shown in Appendix A.

APPENDICES A AND B

Appendix A – References and Bibliography.

1. George Boole. "The Mathematical Analysis of Logic", Cambridge, 1947.
"An Investigation of the Laws of Thought", London, 1854.
2. C. E. Shannon. "A Symbolic Analysis of Relay and Switching Circuits", Trans. A.I.E.E. Vol. 57 pp. 713–723, 1938.

Bibliography

Hollingdale, S.H. "High Speed Computing – Methods and Applications", The English Universities Press, Ltd., London, 1959.

Gould, I. H. and Ellis, F. S. "Digital Computer Technology" Chapman and Hall, Ltd., London, 1963.

Siegel, P. "Understanding Digital Computers" John Wiley and Sons, Inc., New York, 1961.

Caldwell, S. H. "Switching Circuits and Logical Design" John Wiley and Sons, Inc., New York, 1958.

Marcus, M. P. "Switching Circuits for Engineers" Prentice-Hall International, Inc., London, 1962.

Karnaugh, M. P. "The Map Method for Synthesis of Combinational Logic Circuits" Trans. A.I.E.E. Vol. 72, Pt. 1, pp. 593–599, 1953.

Appendix B *Glossary of Terms used in Digital Computing.*

AND Gate	A device which produces an output only when all the inputs are present. Sometimes called a "Coincidence" gate.
Base.	Number of digits used in a positional number system.
Binary.	Involving the integer two.
Binary Point.	The point which separates the integral from the fractional powers of two in the binary system.
Bistable.	A device which remains in one of two stable states until an external stimulus triggers it into the other stable state.
Bit.	An abbreviation for "binary digit".
Carry.	The digit to be taken into account at the next higher place when the sum of the digits in a column is greater than the base.
Clock.	A timing device which produces pulses at a steady rate for synchronising purposes.
Complement.	The inverse of a logical expression.
Digit.	A symbol used to represent one of the elements in a positional number system.
Exclusive OR.	A device which produces an output only when either of two inputs is present, but not when both are present.
Gate.	Term used to describe a switching circuit.
Half Adder.	A device which produces signals representing the sum and carry for the addition of two binary numbers.
Memory.	An initial store controlled by the computer.
OR Gate.	A device which produces an output if any or all the inputs are present. Also known as an "Inclusive OR" gate.
Parallel Mode.	When all the bits of a computer word are available at the same time, they are said to be in parallel.
Programme.	A detailed set of instructions for the solution of a problem.
Register.	A device capable of storing a group of digits, usually one computer word.
Serial Mode.	When all the bits of a computer word are available in a time sequence, they are said to be in serial.
Store.	That part of the computer in which information is held.
Word.	The maximum number of bits which the computer can handle in one group. Such a group may represent an instruction or numerical data for a problem.