AUSTRALIAN POST OFFICE

ETP 0072

TECHNICAL TRAINING PUBLICATION

Engineering Training Group, 6 Flinders Street, Melbourne, Victoria, 3000

ELECTRONIC LOGIC PRINCIPLES 1

PUBLISHED 1970

(PREVIOUSLY HGP 801)

1. INTRODUCTION.

1.1 In the past, relay circuits were used extensively in industrial control systems and automatic telephone exchanges to make decisions and control mechanical operations. Even the original digital computers were developed using relay logic circuits. Nowadays, electronic logic circuits are preferred because of their reliability, speed of operation, and size; they have allowed many systems to be developed that were previously only a theoretical possibility. The increasing use of electronic logic circuits in telephone, telegraph, and data transmission, automatic telephony, radio supervisory equipment, measuring instruments, and mail handling equipment, means that the technician of the future must have an understanding of electronic logic principles and techniques. This paper explains how electronic circuits can make the decisions previously entrusted to relay contact arrangements. The paper Electronic Logic Principles 2 explains how information is stored and transmitted in electronic logic systems.

1.2 At the time of writing, there is no universally accepted standard for the use of terms, symbols, and notations in logic equipment. Until a universal standard is adopted, and observed, the following practices generally apply in this, and other papers of the series:-

The word "logic" is used as a noun or adjective as required.

Tables variously termed "Tables of Combination", or "Tables of Possibilities" are termed "Truth Tables".

Logic 1 and logic 0 are used generally, although the use of H (high) and L (Low) as an alternative is briefly explained.

Since many different symbols exist in the field, one type only has been chosen for use throughout the paper; other equivalent symbols are shown on page 32.

2. INTRODUCTION TO LOGIC.

2.1 BINARY VARIABLES. Aristotle, the Greek philosopher, made a study of logic and developed it as a tool for solving philosophical problems. In the late 1930s bivalent, or two-state logic was adapted for analysing multi-contact networks in automatic telephone equipment, and it is now used in the design and understanding of electronic logic equipment.

Bivalent (two-state) logic uses a binary variable which:-

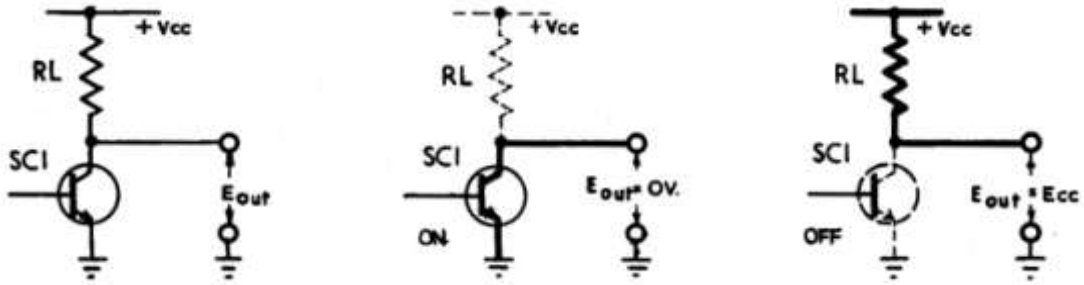
- Can have only two possible conditions (or states).
- Must be either in one condition or the other.
- Is never in both conditions at the same time.

2.2 Examples of the use of binary variables in various types of bivalent logic applications are:-

• To philosophers a statement is either	TRUE	or	FALSE
• Switches or relay contacts are either	CLOSED	or	OPEN
• An electronic logic signal is either at	ONE DEFINED VOLTAGE LEVEL	or	ANOTHER DEFINED VOLTAGE LEVEL

This paper is concerned with the application of bivalent logic or electronic circuits using two defined voltage levels for the binary variable.

2.3 DERIVATION OF VOLTAGE LEVELS IN ELCTRONIC LOGIC CIRCUITS. The circuit arrangement in Fig. 1a is often used to develop the defined voltage levels in electronic logic circuits. SC1 and RL form a voltage divider, where the resistance of SC1 determines the output voltage. SC1 is operated as a switching transistor which is either saturated (ON), or cut-off (OFF). When the transistor is saturated (Fig. 1b), its emitter to collector resistance is very small (assumed to be negligible), and 0 volts is extended to the output. This is one defined voltage level of the binary variable. When the transistor is cut-off (Fig. 1c) its emitter to collector resistance is high (assumed to be an open-circuit) and NO current flows through RL. Since there is no potential drop across RL, the supply voltage Vcc is extended to the output. This is the other defined voltage level of the binary variable.



(a) (b) (c)

FIG. 1. DERIVATION OF VOLTAGE LEVEL.

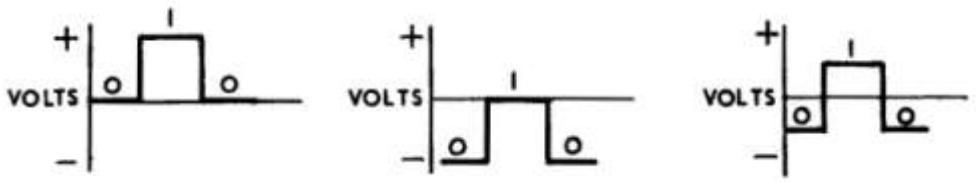
2.4 LOGIC 1 AND LOGIC 0. For convenience, the two states of a binary variable are assigned logic symbols, such as the binary notations 1 and 0, or H and L. In practice, the symbols 1 and 0 are commonly used. Either of the two states of the binary variable can be assigned logic 1, but it is usual for:-

Logic 1 to represent	(a)	a TRUE or SIGNIFICANT statement.
	(b)	a CLOSED switch or contact.
Logic 0 to represent	(a)	a FALSE or INSIGNIFICANT statement.
	(b)	an OPEN switch or contact.

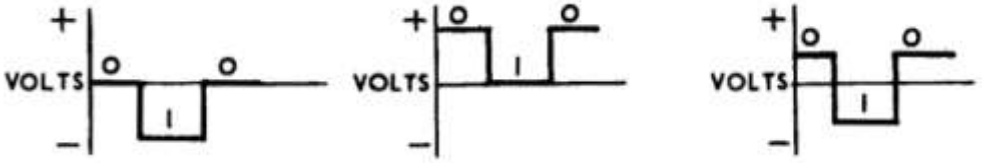
In electronic logic systems, one of the two defined voltage levels is assigned logic 1, and the other logic 0. Once the logic significance has been defined, it is usual for it to be maintained throughout a system.

2.5 POSITIVE AND NEGATIVE LOGIC. When the more positive voltage is assigned logic 1, and the less positive voltage is assigned logic 0, the system is said to use "positive" logic. Examples of positive logic are shown in Fig. 2a. When the more negative voltage is assigned logic 1, and the less negative voltage is assigned logic 0, the system is said to use "negative logic". Examples of negative logic are shown in Fig. 2b.

Some manufacturers avoid giving one voltage level more significance than the other, and designate the more positive voltage "High" (H) and the less positive voltage "Low" (L).



(a) Examples of Positive Logic.



(b) Examples of Negative Logic.

FIG. 2. POSITIVE AND NEGATIVE LOGIC.

2.6 BOOLEAN ALGRBRA. In normal algebra, lengthy word statements are written in shorthand form by using alphabetical characters to represent the variables, and standard arithmetical symbols to show the relationship between the variables. For example, Ohms law states that the current in a circuit is directly proportional to the applied voltage, and inversely proportional to the resistance of the circuit. Algebraically, this lengthy statement is written as $I = \frac{E}{R}$, where I, E and R are the variables; and since I depends on the values of E and R, it is the dependent variable. The "equals" and "division" signs indicate the relationship between the variables.

In electronic logic circuitry, the lengthy word statements used to describe logic relationships are expressed in shorthand form with Boolean algebra; named after the originator George Boole. In Boolean algebra, alphabetical characters such as A, B, C etc., are used to represent the binary variables, and arithmetical symbols show the relationships between the variables. The arithmetical symbols used in Boolean algebra have the following meanings:-

- The symbol (.) represents the word AND.
- The symbol (+) represents the word OR.
- The symbols ($\bar{}$) or (') represents the word NOT. The (') is termed a "Prime".

Note, that the (+) sign loses its normal arithmetical meaning when used in Boolean algebra.

The following example shows how Boolean algebra is used to express the relationship in a logic circuit. Fig. 3 is a block diagram representing a logic circuit having three inputs A, B and C. The logic condition on output D is dependent on the logic conditions on inputs A, B and C. Now assume that the following word statement describes the logic relationship of the circuit:-

D is logic 1 when A is logic 1 AND B is logic 1 AND C is logic 1.

This word statement is expressed in Boolean form by:-

D = A AND B AND C.

Replacing the AND with (.) the equation becomes:-

D = A.B.C.

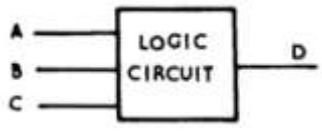


FIG. 3. LOGIC CIRCUIT.

Note, that the word "when" in the word statement is replaced by an equal sign (=) in the Boolean equation, because the condition on output D is dependent on the conditions on inputs A, B and C.

The term A.B.C is a Boolean expression which describes the function performed by the logic circuit preceding the output.

The conditions required to make the output of the logic circuit significant are more readily seen from the Boolean expression than from the word statement. It is essential that you should become familiar with interpreting the Boolean expressions used in logic diagrams. Word statements are very limited in their application, as they are too cumbersome when used to express the logic relationships in complete logic circuits.

3. THE **AND** FUNCTION.

3.1 The function of any circuit made up of switches, or contacts, is to provide a complete circuit path when specified conditions exist. Fig. 4 is the circuit of two switches (A and B) connected so that the circuit (C) is complete when switch A AND switch B are closed. It therefore performs the AND function.

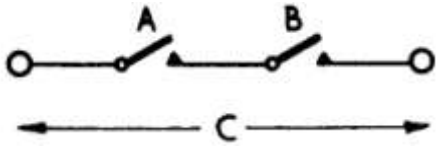


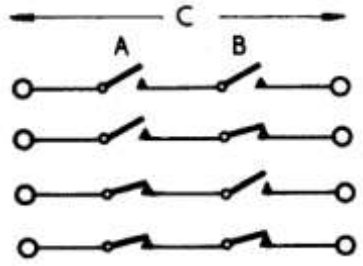
FIG. 4. SWITCHES CONNECTED TO PERFORM AND FUNCTION.

3.2 TRUTH TABLES. The possible combinations of the states of the variables in a circuit which performs a logic function are often tabulated in tables using the logic symbols 1 and 0. Tables of this type are called truth tables, because they show which combination, or combinations, make the dependent variable true (logic 1),. They are also known as tables of possibilities, or tables of combinations.

3.3 TRUTH TABLE FOR AND FUNCTION. Table 1 is the truth table which gives the circuit possibilities of the three binary variables shown in Fig. 4. These variables are:-

- A, which is either CLOSED (logic 1) or OPEN (logic 0),
- B, which is either CLOSED (logic 1) or OPEN (logic 0), and
- C, the complete circuit, which is either CLOSED (logic1) or OPEN (logic 0).

All possible combinations of A and B are listed in the first two columns of the truth table, and the condition of C for each combination is listed in the third column. Fig. 5 shows all the possible combinations of the variables in circuit form.



... C open (Logic 0) .....
... C open (Logic 0) .....
... C open (Logic 0) .....
... C closed (Logic 1) .....

FIG. 5 CIRCUIT POSSIBILITIES.

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

TABLE 1. TRUTH TABLE FOR AND FUNCTION.

3.4 WORD STATEMENT OF AND function. Table 1 is the truth table for an AND function, and it can be summed up with the following word statement:-

C is logic 1 when A is logic 1 AND B is logic 1.

3.5 BOOLEAN EQUATION OF AND function. Boolean algebra is used to express this statement in shorthand, as follows:-

$C = A.B$ where C is the dependent binary variable,
A and B are binary variables,
The symbol (.) is read as AND.

In practice the (.) may be omitted and the expression written as AB.

Note that the equation expresses the condition which applies to the circuit in its logic 1 state only. For C to be logic 0, it is not necessary for both A and B to be logic 0.

4. AND GATES.

4.1 The function of an electronic gate circuit is to provide a specified voltage level at its output, when some specified input voltage conditions exist. An AND gate is defined as an electronic logic circuit which provides a logic 1 voltage level on its output when all inputs are at the logic 1 voltage level. Fig. 6 shows the symbol used to represent a two input AND gate in logic diagrams in this paper. Other AND gate symbols are shown in page 32.

The lines on the symbol represent single wires. Those on the left are inputs to the gate, and the one on the right is the output. When this symbol is used the power supply wiring is not shown.

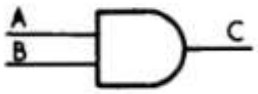


FIG. 6. ELECTRONIC AND GATE SYMBOL.

4.2 EXPRESSION OF AND GATE OPERATION. In the AND gate shown in Fig. 6 there are three binary variables, namely:-

- A and B, the inputs, which are either at the voltage level representing logic 1, or, the voltage level representing logic 0.
- C, the output and dependent variable, which is either at the voltage level representing logic 1, or the voltage level representing logic 0, depending on the logic levels on the inputs.

In Table 2, the possible combinations of input conditions are tabulated, together with the logic conditions which result on the output.

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 2. TRUTH TABLE FOR AND GATE.

Since output C is logic 1, only when input A is logic 1 AND input B is logic 1, the AND function is performed. Therefore, the operation of an AND gate can be expressed in Boolean form by:-

$C = A.B$ where C is the output, A and B are the inputs, The symbol (.) is read as AND.

4.3 AND GATE rule. The following rule allows the logic significance of the output of an electronic AND gate to be determined. It applies to all AND gates, irrespective of the number of inputs connected.

The output of an AND gate is:-

- logic 1 when ALL inputs are logic 1.
- logic 0 when ANY input is logic 0.

Question 1 on page 32 is an exercise on AND gates, and should be attempted before reading further.

5. THE OR FUNCTION.

5.1 Fig. 7 shows two switches connected so that the circuit C is complete, or significant, when switch A OR switch B is closed. It therefore performs the OR function.

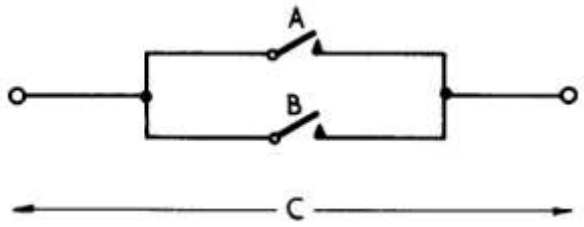


FIG. 7. SWITCHES CONNECTED TO PERFORM OR FUNCTION.

5.2 TRUTH TABLE FOR OR FUNCTION. Three binary variables are represented by the circuit in Fig. 7, these are:-

- A, which is either CLOSED (logic 1) or OPEN (logic 0),
- B, which is either CLOSED (logic 1) or OPEN (logic 0),
- C, the complete circuit, which is either CLOSED (logic1) or OPEN (logic 0), and is the dependent variable.

Table 3 shows the possible combinations of A and B and the condition of C for each combination

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

TABLE 3. TRUTH TABLE FOR OR FUNCTION.

5.3 WORD STATEMENT OF OR FUNCTION. A word statement summing up the conclusion reached from Table 3 states that:-

C is logic 1 when A is logic 1 OR B is logic 1.

5.4 BOOLEAN EQUATION FOR OR function. Boolean algebra is used to express this statement in shorthand form as follows:-

$C = A + B$ where C is the dependent binary variable, A and B are binary variables, The symbol (.) is read as AND.

Another way of expressing C = A + B is:-

$C = A$

$C = B$

This does not mean that C equals A, and C equals B; it means that C is dependent on the logic state of A only OR B only.

6. OR GATES.

6.1 An OR gate is defined as an electronic logic circuit which provides a logic 1 voltage level on its output when any input is at the logic 1 voltage level. OR gates are often represented in logic diagrams by the symbol shown in Fig. 8. Other OR gate symbols are given on page 32.

6.2 EXPRESSION OF OR GATE OPERATION. The behaviour of an electronic OR gate for all possible combinations of input logic levels is given in Table 4.

FIG. 8. ELECTRONIC OR GATE SYMBOL.

INPUTS A	INPUTS B	OUTPUT C
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 4. TRUTH TABLE FOR OR GATES.

FIG. 8. NAND GATES AS SR FLIP-FLOP.

Since the output C is logic 1 when either input A OR input B is logic 1, (or both are logic 1) the OR function is performed. Therefore, the operation of an OR gate can be expressed in Boolean form by:-

$C = A + B$ where C is the output, A and B are the inputs, The symbol (+) is read as OR.

6.3 OR GATE RULE. The output of an OR gate is:-

- logic 1 when ANY inputs are logic 1.
- logic 0 when ALL input is logic 0.

Question 2 on page 33 is an exercise on OR gates, and should be completed before reading further.

7. THE NOT FUNCTION.

7.1 The NOT function is the "negation" or the "inversion" of the state of a variable, and the electronic logic element which performs it is called an inverter. When the input of an inverter is logic 1, its output is logic 0 (that is, NOT logic 1). Conversely, when its input is logic 0, its output is logic 1 (that is, NOT logic 0). It therefore inverts the logic condition on its input.

Fig. 9 shows a symbol used for an inverter, and Table 5 is the Truth table. When the input of an inverter is designated A, the inverted output is designated $\bar{A}$ or A′, which is read as NOT A or A NOT. A and $\bar{A}$ are referred to as the complements of each other. Other inverter symbols are given on page 32.

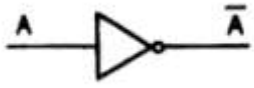


FIG. 9. INVERTER SYMBOL.

INPUT A	OUTPUT C
0	1
1	0

TABLE 4. TRUTH TABLE FOR INVERTER.

Question 3 on page 34 should now be completed.

8. TIMING DIAGRAMS.

8.1 In logic circuits, it often occurs that the signals appearing at the gate inputs are of various pulse lengths, and not repetitive. With AND gates it is necessary that all inputs be significant at the same time for the gate output to become significant. A timing diagram enables us to determine when the output of a gate becomes significant, and also the output pulse length.

Fig. 10 shows an example of the input conditions applied to an AND gate and the resulting output. It is assumed that the significant condition (logic 1) is a positive voltage level, and logic 0 is zero volts. (This is positive logic).

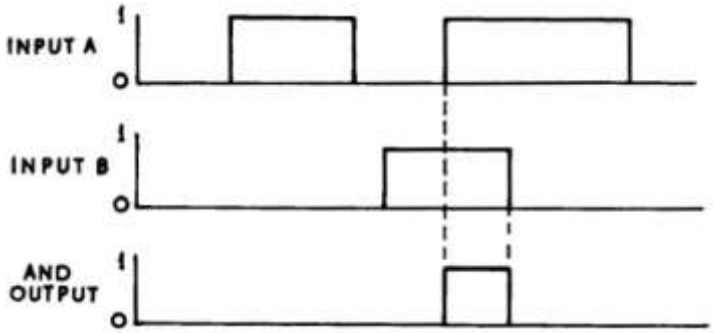


FIG. 10. AND GATE TIMING DIAGRAM.

The graphs in Fig. 10 show that the output of the AND gate is logic 1 only when both inputs are logic 1.

Now assume that the same signals are applied to an OR gate, as shown in Fig. 11. In this case, the output is significant when either input A OR input B is significant.

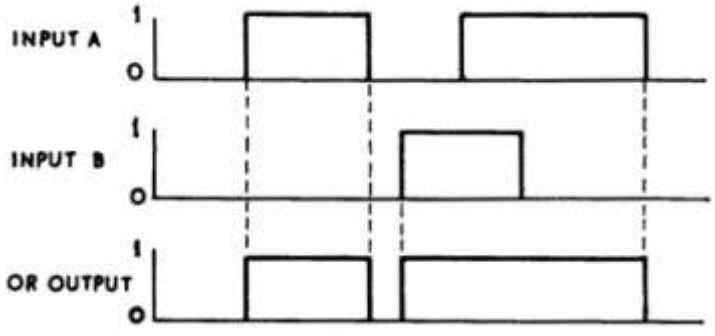


FIG. 11. OR GATE TIMING DIAGRAM.

9. COMBINED LOGIC FUNCTIONS.

9.1 GENERAL. Electronic logic circuits are required to provide a specified logic condition on their output when some special input conditions exist. Logic circuits are made up by combining elements which perform the basic AND, OR and NOT functions.

9.2 EXAMPLE OF COMBINED LOGIC FUNCTION. The following example shows how a combined logic circuit is developed. The logic circuit associated with the mechanical equipment in Fig. 12 is required to provide a pulse of logic 1 when an article exceeding four inches in height, or six inches in length, passes along the conveyor. The input information is generated by light beams, and the output pulse could be used to control the operation of a solenoid which initiates some mechanical operation.

The light detectors A, B and C provide a logic 1 to the logic circuit when an article breaks a beam. Light beam A is broken by any article higher than four inches. Light beams B and C are placed so that articles longer than 6″ break both beams at the same time.

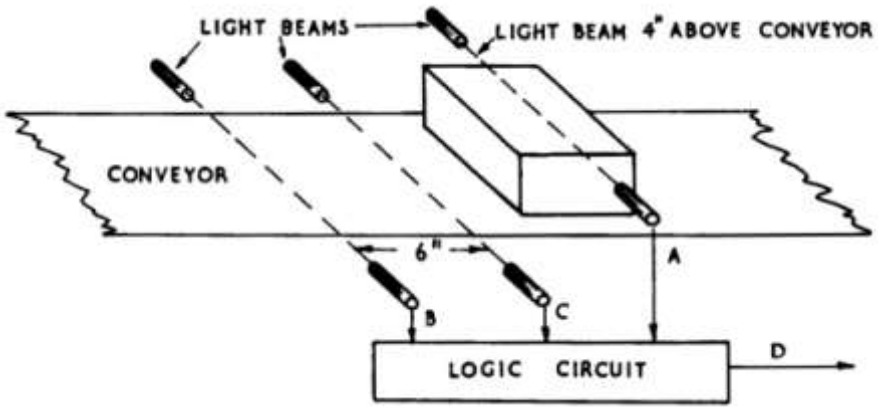


FIG. 12. DETECTION OF OVERSIZE ARTICLES.

A logic circuit which will perform the desired function is developed by making a logic word statement of the function to be performed, expressing it as a Boolean equation, and constructing a circuit from this Boolean equation.

The function required of the logic circuit is expressed by the following word statement:-

> The output D must be logic 1 whenever A is logic 1 OR both B AND C are logic 1.
>
> This means that D is logic 1 whenever the light beam to A is broken, OR when the light beams to B and C are both broken at the same time.

As logic circuits become more complex, the word statement of the function performed becomes more involved, and the advantage of expressing the function in shorthand is more obvious. In this case the word statement of the function is expressed in Boolean form as follows:-

$$D = A + B.C$$

ELECTRONIC LOGIC CIRCUIT. Fig. 13 is the logic diagram of an electronic circuit which uses an AND and an OR gate to perform the function required in Fig. 12. The logic diagram is drawn by arranging the gates and connections so that the signal A + B.C is produced at the output. To do this, B and C are connected to the AND gate so that the signal B.C (B AND C) is developed at its output. The signal B.C and the signal A are connected to the OR gate so that its output, which is the combined output of the circuit, is A + B.C (A OR B AND C).

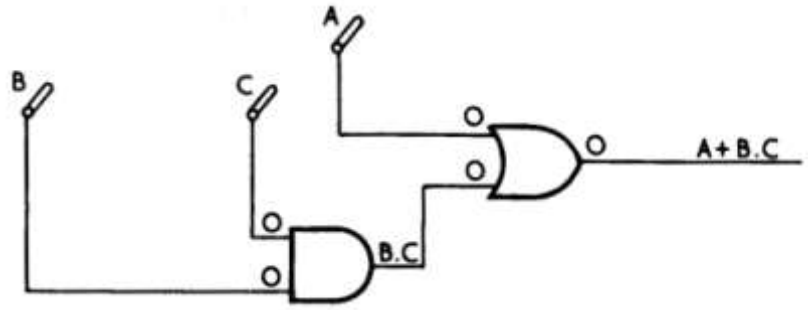


FIG. 13. ELECTRONIC LOGIC CIRCUIT.

OPERATION OF CIRCUIT. At normal, that is with no light beam broken, the logic levels indicated in Fig. 13 apply, and the output is at logic 0. Articles within the height specification pass without breaking light beam A, so signal A stays at logic 0. Although light beams B and C are both broken when an article within the length specification passes, they are broken at different times, so signals B and C are logic 1 at different times causing signal B.C to stay at logic 0. Normal articles, therefore, allow the output to remain at logic 0.

When a high article passes, A goes to logic 1, causing output D to go to logic 1. Also, a long article causes B and C to go to logic 1 together, thus producing logic 1 at output (B.C) of the AND gate, which in turn causes output D to go to logic 1. A logic 1 on output D activates a mechanical function which rejects the oversize article.

9.3 **INTERMEDIATE POINTS.** The intermediate points on logic diagrams are often designated with a Boolean expression which describes the logic function of the preceding equipment. For example B.C designates an intermediate point in Fig. 13, and indicates that B and C have been ANDed together in the previous stage. Although an intermediate point can be at logic 1, or logic 0, depending on the logic states of the inputs, the expression at an intermediate point can be used to determine when that point is at logic 1. For example, point B.C is logic 1 when B is logic 1 AND C is logic 1. If B and C do not meet these conditions the intermediate point B.C is at logic 0.

9.4 **CONVERTING BOOLEAN EXPRESSIONS TO LOGIC DIAGRAMS.** It is essential that you should be able to convert Boolean expressions to logic diagrams, and vice versa.

One method of converting Boolean expressions to logic diagrams is to use the following steps, in the order given:-

- Combine any bracketed terms with the type of gate indicated by the sign within the brackets.
- Combine any ANDed terms.
- Combine any ORed terms.

As an example, Fig. 14 is the logic diagram derived from the Boolean expression A.(B + C) + D. The first step in drawing this diagram is to combine the bracketed signals B and C in an OR gate (G1) to obtain the signal (B + C). Next, signal A and signal (B + C) are combined in an AND gate (G2) to obtain the signal A.(B + C). Then, signal A.(B + C) and signal D are combined in an OR gate (G3) to obtain the output signal A.(B + C) + D.

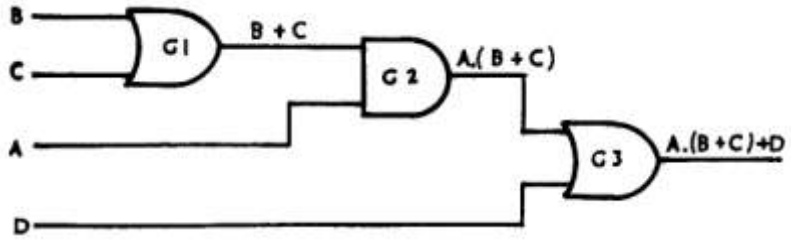


FIG. 14. LOGIC DIAGRAM REPRESENTING A.(B + C) + D.

Boolean expressions not containing bracketed expressions are converted to logic diagrams by considering ANDed functions first and then the ORed functions. For example, A + B.C is represented by the logic diagram shown in Fig. 12.

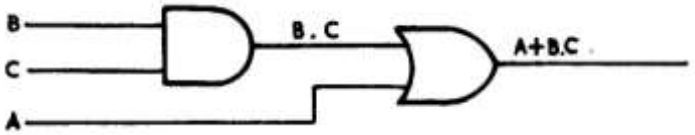


FIG. 15. LOGIC DIAGRAM REPRESENTING A + B.C.

9.5 **BOOLEAN EXPRESSIONS FROM LOGIC DIAGRAMS.** To derive a Boolean expression from a logic diagram, it is necessary to start from the inputs and progressively work towards the output, establishing a Boolean expression for each intermediate point. For example, consider the logic diagram shown in Fig. 16. First, the output of the inverter is established as $\bar{A}$. Signal $\bar{A}$ and signal B are ORed in gate G1 to establish the intermediate point $\bar{A} + B$. Next, signal $\bar{A} + B$ and signal C are ANDed in gate G2 to obtain the intermediate point $(\bar{A} + B).C$. The signal $(\bar{A} + B).C$ and signal D are then ORed in gate G3 to obtain the output signal $(\bar{A} + B).C + D$.

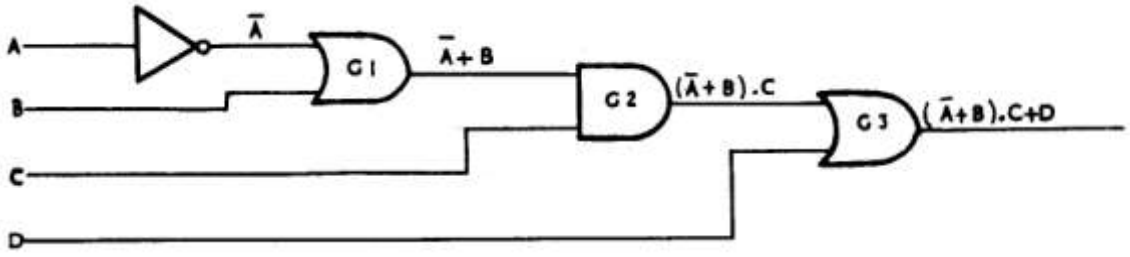
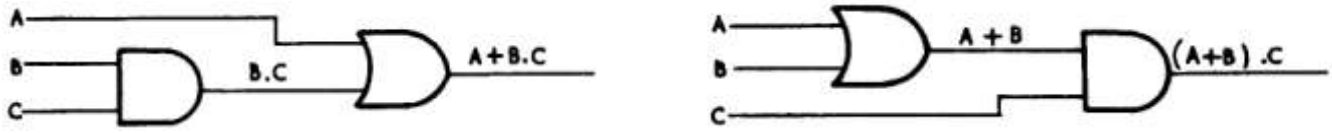


FIG. 16. $(\bar{A} + B).C + D$.

9.6 BRACKETS. In general, the following rule should be observed if brackets are to be used in a Boolean expression. If an OR function occurs, and the output of this is ANDed with another term, it is necessary to place brackets around the ORed terms. The presence or absence of brackets in a Boolean expression can completely change the logic circuit represented by the expression.

For example, the expression A + B.C has a different logic circuit to the expression (A + B).C, as shown in Figs 17(a) and 17(b) respectively.



(a) Logic circuit of A + B.C

(b) Logic circuit of (A + B).C

FIG. 17. THE IMPORTANCE OF BRACKETS.

Question 4 on page 34 should now be attempted.

10. CONSTRUCTION OF TRUTH TABLES.

10.1 GENERAL. Truth tables are used to record all the possible combinations of the logic conditions in a circuit, and the function of a logic circuit can easily be established by this means. Where the function of a logic circuit is not easily established from the Boolean expression, or from the circuit itself, a truth table should be developed. To assist in later studies of complex logic circuits, you should form the habit of developing and interpreting truth tables for all circuits.

10.2 This paragraph describes a simple method of constructing a truth table to ensure that all possible input combinations are included. When constructing truth tables, the number of input combinations is equal to 2^n, where n is the number of inputs. Table 6, which is the truth table for the circuit in Fig. 13, is used as an example.

1. Head the column inputs, intermediate points, and output, as shown in Table 6. (Intermediate points are not always shown in truth tables, but assist in determining the output conditions).
2. Designate a column for each input, the intermediate points, and the output.
3. Determine the number of combinations. In this case the number of combinations is $2^3 = 8$ because there are three inputs in Fig. 13. The number of combinations determines the number of horizontal lines in the truth table.
4. Commence all input columns with zeroes.
5. Complete the last input column (in this case C), by changing the condition of the binary variable on each line.
6. Complete the second last input column (in this case B) by changing the condition of the binary variable after every two combinations.

Inputs	Intermediate Point	Output (D)
A B C	B.C	A + B.C
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	1
1 0 0	0	1
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

Table 6. TRUTH TABLE FOR FIG. 13.

7. Complete the next input column (in this case A) by changing the condition of the binary variable after every four combinations.
8. Record the conditions of the intermediate points. In the example, intermediate point B.C is logic 1 only when B is logic 1 AND C is logic 1. If either B or C are at logic 0 the intermediate point B.C is at logic 0.
9. Record the output condition for each combination of input conditions. In the example, D is logic 1 when A is logic 1 OR when the intermediate point B.C is at logic 1 (that is, D = A + B.C).

When more than three inputs are involved, the same principle applies. Each input column, from last to first has its condition changed according to the binary number pattern. For example, the fourth last input column of a truth table would have the condition of the binary variable changed every eight combinations, and the fifth last column every 16 combinations, and so on.

Question 5 on page 37 is an exercise on constructing truth tables.

11. NAND AND NOR GATES.

11.1 GENERAL. Electronic AND and OR gates are usually passive networks containing diode and resistors only and, therefore, have an output power which is less than the input power. A desirable feature of logic gates is that their output signal should be capable of operating a number of other gates. The limited output power from passive AND and OR gates prevent them from doing this without some sort of amplification. For this reason, gate circuits have been developed which provide a transistor amplifier in conjunction with the basic AND and OR gates. Because of the transistor amplifier connected in the output, the gate provides a logic inversion. An electronic gate which contains an AND gate with an inverter (transistor amplifier) is called a NOT AND or NAND gate. An electronic gate which contains an OR gate with an inverter is called a NOT OR or NOR gate.

Both NAND and NOR gates can be arranged to perform the AND, OR or NOT functions, and in some cases, complete systems are built up using one type of NAND or NOR gate. Modern NAND or NOR gates are built in the form of integrated circuits, which can be manufactured more cheaply than AND or OR gates built from discrete (individual) components. Although the minimum number of elements is not necessarily achieved in a system built up from modern NAND or NOR gates, these systems have the advantage that they can be manufactured more economically than a system using discrete components, and faults are more easily rectified by substituting spare universal NAND or NOR elements.

11.2 NAND GATES. A NAND gate is defined as an electronic circuit which provides a logic 0 voltage level on its output when all inputs are at the logic 1 voltage level. Fig. 18a is a symbol used to represent NAND gates in logic diagrams; other NAND gate symbols are shown on page 32. A small circle, called a state indicator, is added to an AND gate symbol to indicate an inversion following the AND function. Fig. 18b is an equivalent circuit of a NAND gate using basic AND and NOT elements.

Since the output of a NAND gate is the inverted output of the AND function A.B, it is written $\overline{A.B}$, which is stated as (A AND B) NOT, or NOT (A AND B). The expression $\overline{A.B}$ indicates, therefore, that the two input signals A and B are "ANDed" together and the result is inverted. If the result of A.B is logic 1, $\overline{A.B}$ is logic 0 (that is, NOT logic 1). If the result of A.B is logic 0, $\overline{A.B}$ is logic 1 (that is, NOT logic 0). This is verified in table 7 which is the truth table for a NAND gate. Although signal $\overline{A.B}$ in Fig. 18b is not always available as an output, it often helps to include this intermediate point when considering NAND gates.

(a) Symbol (b) Logic Equivalent Circuit.

FIG. 18. ELECTRONIC NAND GATE.

Inputs			Output
A	B	A.B	$\overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Table 7. NAND GATE TRUTH TABLE.

11.3 NAND GATES USED TO PERFORM THE NOT FUNCTION. When only one input to a NAND gate is used, and the others have a permanent logic 1 connected to them, the NOT function is performed. The output is dependent on the condition on the effective input, and is the inverse of it. In Fig. 19 the second input is permanently tied to logic 1, and the conditions which exist when A is logic 0 are shown above the line. Similarly, the conditions which exist when A is logic 1 are shown below the line. These conditions are recorded in Table 8. Since the output in each case is the inverse of the input, the output is A NOT, ($\bar{A}$).

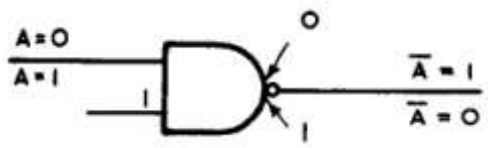


FIG. 19. NAND GATE USED AS INVERTER.

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

TABLE 8. TRUTH TABLE FOR FIG. 19.

11.4 When NAND or NOR gates are used to perform a logic function, a Boolean expression is obtained at the output which describes the elements involved, but does not necessarily describe the function in its simplest form. For example, Fig. 21 shows how three NAND gates are connected to perform the OR function. The Boolean expression $\overline{\overline{A}.\overline{B}}$ at the output of this circuit indicates the elements used, but does not describe the basic function of the circuit, which is A + B. Two methods which can be used to prove the equivalence of two Boolean expressions, (that is, to prove that $\overline{\overline{A}.\overline{B}}$ = A + B) are:-

- Comparison of the truth tables of each expression. If all input combinations and the resulting output conditions in each truth table are the same, the expressions are equivalent.
- Use of Boolean manipulation techniques. An introduction to basic Boolean manipulation techniques is given in Section 14 of this paper.

11.5 NAND GATE COMBINATION TO PERFORM THE AND FUNCTION. In Fig. 20 two NAND gates are combined to perform the AND function.

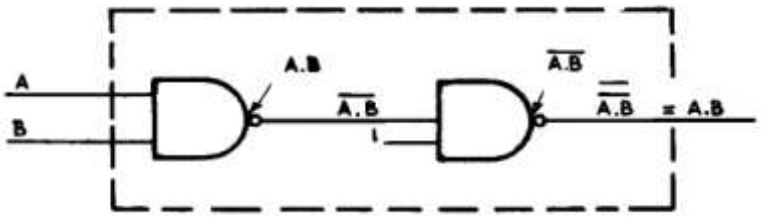


FIG. 20. NAND GATE COMBINATION WHICH PERFORMS THE AND FUNCTION.

Since only one input to the second NAND gate is used, and the other is permanently connected to logic 1, this gate performs the NOT function only. Therefore, the signal A.B produced in the first gate is inverted twice and appears in the output. Tables 9 and 10 are the truth tables for Fig. 20 and an AND gate, respectively. Since the input combinations in the tables are arranged in the same order, and the output results are the same, both circuits perform the AND function. Also, because Fig. 15 and the AND gate both perform the same function, $\overline{\overline{A}.\overline{B}}$ is equivalent to A.B, that is, $\overline{\overline{A}.\overline{B}}$ = A.B. From this it can be seen that a double inversion restores an expression back to its original form.

Inputs		Intermediate Point		Output
A	B	A.B	$\overline{A.B}$	$\overline{\overline{A.B}}$
0	0	0	1	0
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

Table 9. TRUTH TABLE FOR FIG. 13.

=

Inputs		Output
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 10. AND GATE TRUTH TABLE.

11.6 NAND GATE COMBINATION TO PERFORM THE OR FUNCTION. Fig. 21 shows how three NAND gates are combined to perform the OR function. The expression $\overline{\bar{A}.\bar{B}}$ on the output indicates that the following functions have occurred in the circuit: Signal A is inverted to $\bar{A}$, signal B is inverted to $\bar{B}$, both inverted signals are "ANDed" together ($\bar{A}.\bar{B}$), and the output of the AND function is inverted to $\overline{\bar{A}.\bar{B}}$. Each of these functions is shown in the circuit. Note that the first two NAND gates are used as inverters only.

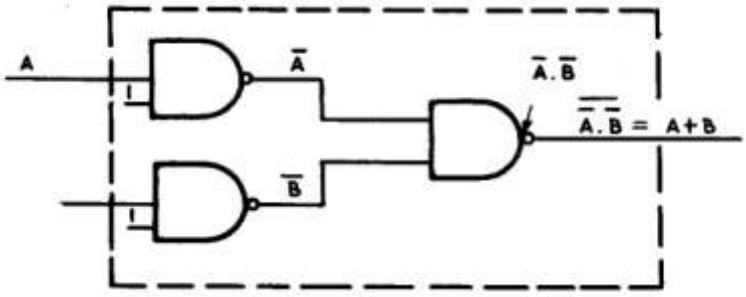


FIG. 21. NAND GATE COMBINATION WHICH PERFORMS THE OR FUNCTION.

Tables 11 and 12 are the truth tables for Fig. 21 and an OR gate, respectively. Since the output results are the same in each case, the circuits are equivalent. Both perform the OR function. Also, because Fig. 21 and the OR gate both perform the same function, $\overline{\bar{A}.\bar{B}} = A + B$.

Inputs		Intermediate Point			Output
A	B	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$	$\overline{\bar{A}.\bar{B}}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Table 11. TRUTH TABLE FOR FIG. 21.

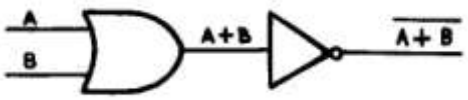
=

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

TABLE 12. OR GATE TRUTH TABLE.

11.7 NOR GATES. A NOR gate is defined as an electronic circuit which provides a logic 0 voltage level on its output when any input is at logic 1. Fig. 22a shows a symbol used to represent NOR gates. Other NOR gate symbols are shown on page 32. Note that a state indicator is added to the OR gate symbol to indicate an inversion. Fig. 22b is an equivalent circuit of a NOR gate using basic OR and NOT logic elements. The expression $\overline{A+B}$ means that signals A and B are "ORed" together and the result inverted.

(a) Symbol.



(b) Equivalent circuit.

FIG. 22. ELECTRONIC NOR GATE.

Although the signal A + B is not usually available as an output, it often helps to include this intermediate step when considering NOR gates. Table 13 shows the function performed by a NOR gate.

Inputs			Output
A	B	A + B	$\overline{A.B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Table 13. NOR GATE TRUTH TABLE.

11.8 NOR GATES USED TO PERFORM THE NOT FUNCTION. The NOT function is performed when one input to a NOR gate is used and the others are tied to logic 0. In Fig. 23, the conditions which exist when A is logic 0 are shown above the line. Similarly, the conditions which exist when A is logic 1 are shown below the line. Since the output in each case is the inverse of the input, the NOT function is performed.

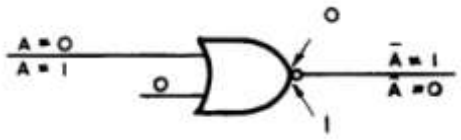


FIG. 23. NOR GATE USED AS INVERTER.

11.9 NOR GATE COMBINATION TO PERPERFORM THE OR FUNCTION. Fig. 24 shows how two NOR gates are combined to perform the OR function.

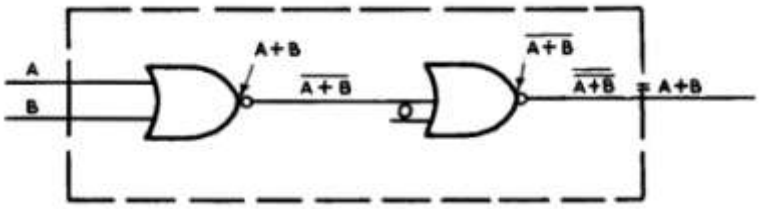


FIG. 24. NOR GATE COMBINATION WHICH PERFORMS OR FUNCTION.

The second NOR gate acts as an inverter, so that the signal A+B produced in the first gate is inverted twice and becomes the output.

11.10 NOR GATE COMBINATION TO PERFORM THE AND FUNCTION. Fig. 25 shows how three NOR gates are combined to perform the AND function. The first tow gates serve as inverters only, to produce the signals $\overline{A}$ and $\overline{B}$. These two signals are "ORed" together and the result is inverted to signal $\overline{\overline{A}+\overline{B}}$.

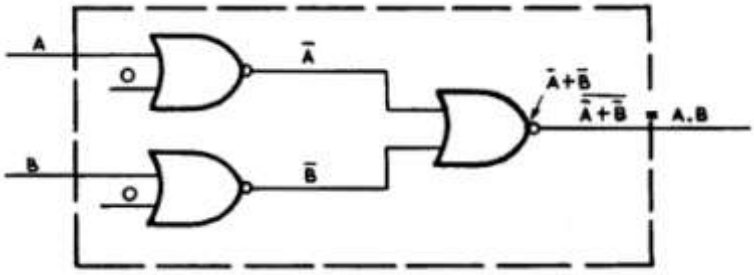


FIG. 25. NOR GATE COMBINATION WHICH PERFORMS AND FUNCTION.

Tables 14 and 15 are the truth tables of Fig, 25 and an AND gate, respectively. Since the input combinations and the output conditions in both tables are exactly the same, Fig. 25 must perform the AND function. Also, this means that the Boolean expressions $\overline{\overline{A}+\overline{B}}$ and A.B are equivalent, that is, $\overline{\overline{A}+\overline{B}}$ = A.B.

Inputs		Intermediate Point			Output
A	B	$\bar{A}$	$\overline{B}$	$\overline{A}+\overline{B}$	$\overline{\overline{A}.\overline{B}}$
0	0	1	1	1	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	0	1

Table 14. TRUTH TABLE FOR FIG. 21.

=

Inputs		Output
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 15. OR GATE TRUTH TABLE.

11.11 **COMPLEX FUNCTIONS USING NAND OR NOR GATES.** The development of the basic functions from NAND and NOR gates tends to indicate that the use of these universal gates is uneconomical. However, careful design can often lead to complex circuits being made from a similar number of NAND and NOR gates, to the number required if AND, OR and NOT elements were used.

Fig. 26 shows how the complex function considered in section 9 is performed using NAND gates. This circuit still performs the function D = A + B.C. The operation of the circuit can be readily understood if the NAND gate truth table is used in conjunction with the following description.

Whilst all light beams are unbroken, A, B and C are at logic 0, and the output D is at logic 0. If light beam A is broken, logic 1 is applied to G1, the output of G1 goes to logic 0, and the output of G3 goes to logic 1. If either B or C go to logic 1, the output of G2 will remain at logic 1, and the output of G3 remains at logic 0. If both B and C go to logic 1, the output of G2 goes to logic 0, and the output of G3 goes to Logic 1.

Question 6 on page 37 is an exercise in reading NAND and NOR gate logic diagrams.

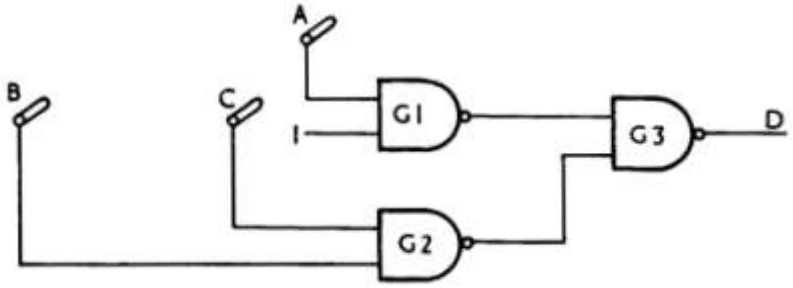


FIG. 26. NAND GATE COMBINATION WHICH PERFORMS COMPLEX FUNCTION.

12. EXCLUSIVE-OR, COMPARATOR AND ADDER.

12.1 **EXCLUSIVE-OR FUNCTION.** The function performed by an OR gate is sometimes referred to as the Inclusive-OR function, because it includes in the conditions which produce a logic 1 output, the case where both inputs are logic 1. Another type of OR function, known as the Exclusive-OR function, provides a logic 1 output when only one input is logic 1 and the other logic 0, and it excludes the condition where both inputs are logic 1. This is expressed as $A.\overline{B} + \overline{A}.B$. The gate combination in Fig. 27 performs the Exclusive-OR function, and Table 16 shows the output condition for each combination of inputs. The expression $A.\overline{B} + \overline{A}.B$ means that the output is logic 1 when A is logic 1 AND B NOT is logic 1 (that is, B is logic 0), OR when A NOT is logic 1 (that is, A is logic 0) AND B is logic 1.

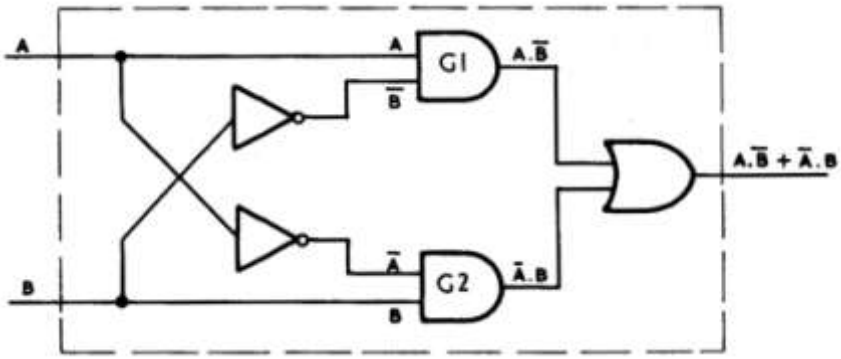


FIG. 27. ECLUSIVE-OR GATE COMBINATION.

Inputs						Output
A	B	$\overline{A}$	B	$A.\overline{B}$	$\overline{A}.B$	$A.\overline{B} + \overline{A}.B$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

Table 16. EXCLUSIVE-OR TRUTH TABLE.

12.2 Another circuit which performs the Exclusive-OR function is shown in Fig. 28. The expression $(A + B).\overline{A.B}$ is another way of expressing the Exclusive-OR function. Expressed in words it states that the output is logic 1 when A OR B are logic 1 AND NOT when A AND B are logic 1.

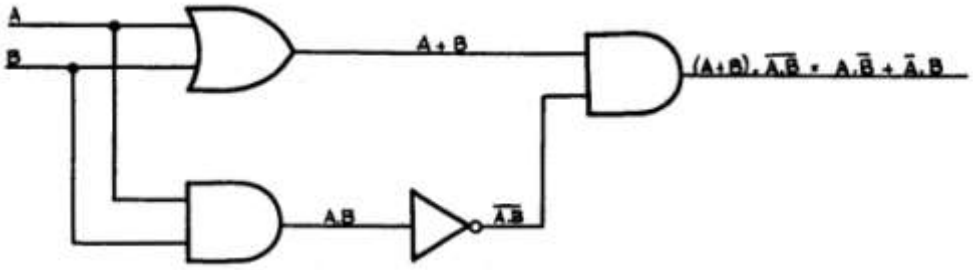


FIG. 28. ALTERNATIVE EXCLUSIVE-OR GATE COMBINATION.

Fig. 29 is a logic symbol which is sometimes used to indicate an Exclusive-OR circuit. The circuit represented by the symbol may contain gate combinations such as those shown on Figs. 27 and 28.

FIG. 29. EXCLUSIVE-OR SYMBOL.

12.3 COMPARATORS. Comparators are special gate combinations which provide a logic 1 output when their inputs have the same logic condition, that is, when both inputs are at logic 1, or when both inputs are at logic 0. This function is expressed as $A.B + \overline{A}.\overline{B}$. Fig. 30 is a simple comparator circuit and Table 17 is the truth table for Fig. 30. When A and B are both logic 1, the top AND gate provides a logic 1 output. When A and B are both logic 0, the signals $\overline{A}$ and $\overline{B}$ are both logic 1, so the bottom AND gate provides a logic 1 output. The comparator circuit is said to test for equivalence.

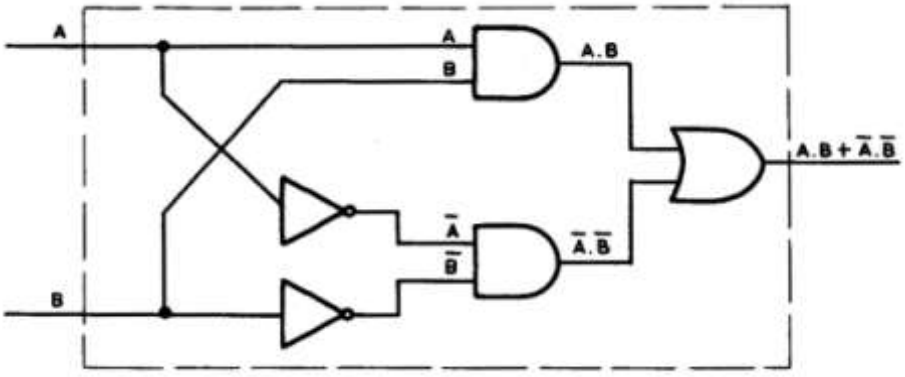


FIG. 30. SIMPLE COMPARATOR.

Inputs						Output
A	B	$\overline{A}$	$\overline{B}$	A.B	$\overline{A}.\overline{B}$	$(A + B).+\overline{A}.\overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

Table 17. TRUTH TABLE FOR FIG. 30.

12.4 The comparator function is also performed when the Exclusive-OR circuit is followed by an inverter, as shown in Fig. 31. Table 18 is the truth table for Fig. 31, and shows that $(A + B).+ \overline{A}.\overline{B}$ is obtained at the output.

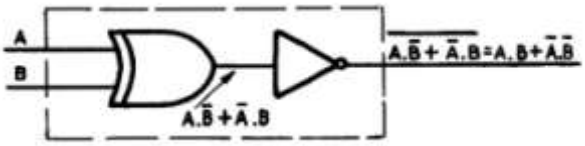


FIG. 31. COMPARATOR USING EXCLUSIVE-OR AND INVERTER.

Inputs			Output
A	B	O/P of Exclusive-OR	$(A + B).+ \overline{A}.\overline{B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Table 18. TRUTH TABLE FOR FIG. 31.

12.5 COMPARATOR USING AND-OR-INVERT COMBINATION. When the signals to be compared and their complements are available, as is often the case, the AND-OR-INVERT combination shown in Fig. 32 can be used as a comparator. Table 19 is the truth table for Fig. 32 and shows that the output of the AND-OR-INVERT combination is exactly the same as that shown in Tables 17 and 18, therefore, the expression $\overline{A.\overline{B} + \overline{A}B}$ is equivalent to $A.B + \overline{A}.\overline{B}$.

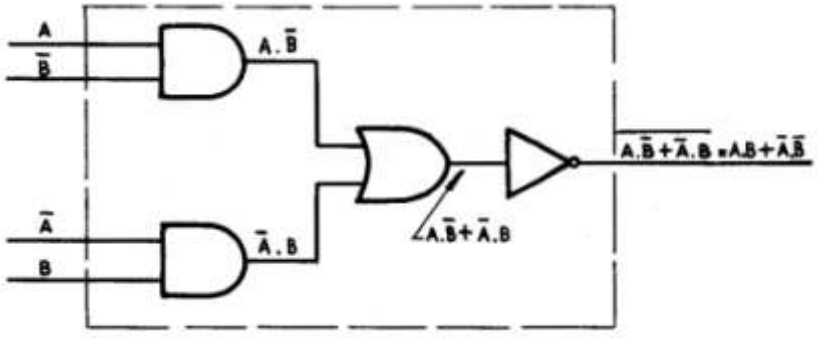


FIG. 32. COMPARATOR USING AND-OR-INVERT COMBINATION.

Inputs							Output
A	B	$\overline{A}$	$\overline{B}$	$A.\overline{B}$	$\overline{A}B$	$(A + B).+ \overline{A}.\overline{B}$	$(A + B).+ \overline{A}.\overline{B}$
0	0	1	1	0	0	0	1
0	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0
1	1	0	0	0	0	0	1

Table 17. TRUTH TABLE FOR FIG. 32.

12.6 HALF ADDER. Adders are special gate combinations which add binary numbers together. When two binary digits are added, the possible answers are:-

0	1	0	1
0	0	1	1
00	01	01	10

When adding binary numbers, the answer is split into two components, the sum, which goes in the right-hand column, and the carry, which is carried into the next most significant column. Notice that the sum is 1 when one digit is 1 and the other is 0, and that the only time a carry is produced is when both digits are 1. A circuit which determines the sum and carry when two digits are added is called a half-adder. Fig. 33 is the basic logic diagram of a half-adder, and Table 20 shows how its function is tabulated in a truth table. The Exclusive-OR gate combination provides a logic 1 output only when one of the inputs is at logic 1, and is used to determine the sum. Similarly an AND gate, which needs logic 1 on both inputs to give a logic 1 output, is used to determine the value of the carry.

For example, if one input is at logic 1 and the other at logic 0, the output of the Exclusive-OR gate (the sum) is logic 1, and the output of the AND gate is logic 0 (the carry). If both inputs are at logic 1, the output of the Exclusive-OR gate is logic 0 (the sum) and the output of the AND gate is logic 1 (the carry).

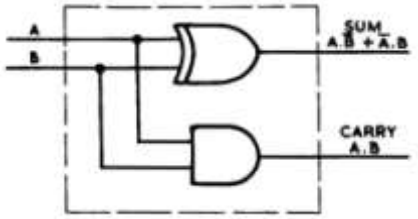


FIG. 33. HALF-ADDER.

Inputs A	Inputs B	Outputs C CARRY	Outputs S SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

TABLE 20. TRUTH TABLE FOR HALF-ADDER.

12.7 FULL ADDER. A full-adder is able to add the two digits on the inputs plus the carry from another adder. Fig. 34 shows that a full-adder contains two half-adders and an OR gate. Table 21 is the truth table for a full adder. Note that the binary sum of the two inputs and the carry-in, in the first three columns, produces the result in the last two columns.

An understanding of the operation of the full adder can be gained by considering each input combination, in turn, through the circuit, and checking that the correct sum and carry-out is obtained, according to the truth table.

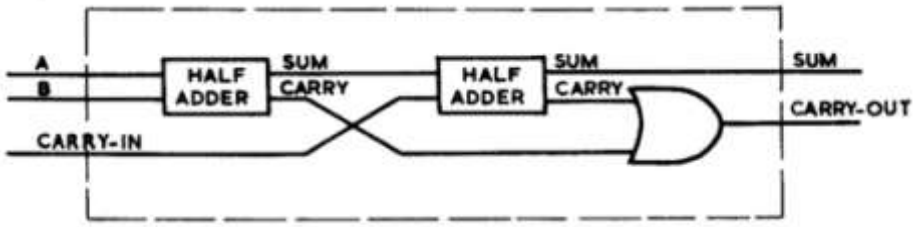


FIG. 34. FULL-ADDER.

Inputs A	Inputs B	Inputs Carry-in	Outputs Carry-out	Outputs Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

TABLE 21. TRUTH TABLE FOR FULL ADDERS.

Question 7 on page 38 is an exercise on comparators and exclusive ORs.

13. ALTERNATIVE WORD STATEMENTS AND LOGIC SYMBOLS.

13.1 GENERAL. In the preceding sections, the word statements used to explain the operation of logic elements are based on the logic 1 conditions required to perform a designated function. Also, the symbols used to represent the elements are derived from the functions performed when the logic 1 conditions are considered. In this section, the truth tables for these elements are reviewed to show that an alternative word statement may be derived when the logic 0 input conditions are considered, and that alternative symbols may be used to represent the alternative word statements. The alternative word statements and logic symbols are used in some applications, because they show more clearly the relationship between a gate and the elements around it.

It should be noted that alternative symbols are not always used in logic diagrams to represent alternative functions. Some manufacturers use the normal symbols to represent elements performing alternative functions, and leave it to the person reading the circuit to apply their knowledge of truth tables to establish the alternative functions of the elements.

13.2 AND GATE. Table 22 is the truth table for an AND gate.

Inputs A	Inputs B	Output C
0	0	0
0	1	0
1	0	0
1	1	1

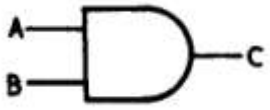
TABLE 22. AND GATE TRUTH TABLE.

The word statements below are derived by examining the AND gate truth table. The normal word statement describes the input requirements to produce a logic 1 on the output; the alternative word statement describes the input requirements to produce a logic 0 on the output. When the circuit connected to the output of the gate is activated by a logic 0, the alternative word statement applies.

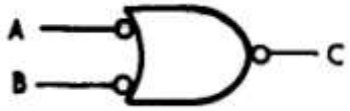
NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
C is logic 1 when	C is logic 0 when
A is logic 1	A is logic 0
AND	OR
B is logic 1	B is logic 0

Note that the normal word statement shows that an AND gate performs the AND function when logic 1 voltage levels are considered, but the alternative statement shows that an AND gate performs the OR function when logic 0 voltage levels are considered. Since it is usual for logic 1 to have the greater significance, the gate derives its name (AND) from the function it performs when considering the logic 1 conditions. Nevertheless it does perform the function:- "C is logic 0 when A is logic 0 OR B is logic 0."

The symbol shown in Fig. 35b is an alternative symbol which may be used in logic diagrams to represent an AND gate which performs the OR function with logic 0 voltage levels. The shape of the symbol indicates the OR function and the state indicators on the inputs and outputs show that this function is performed when logic 0 voltage levels are considered. The normal symbol (Fig. 35a) indicates the AND function, and the absence of state indicators shows that the function is performed when logic 1's are considered.



(a) NORMAL SYMBOL – shows operation for logic 1 conditions.



(b) ALTERNATIVE SYMBOL – shows operation for logic 0 conditions.

FIG. 35. AND GATE – NORMAL AND ALTERNATIVE SYMBOLS.

13.3 OR GATE. Table 23 is the truth table for an OR gate.

Inputs A	Inputs B	Output C
0	0	0
0	1	1
1	0	1
1	1	1

TABLE 23. OR GATE TRUTH TABLE.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
C is logic 1 when	C is logic 0 when
A is logic 1	A is logic 0
OR	AND
B is logic 1	B is logic 0

The normal word statement shows that an OR gate performs the OR function when logic 1 voltage levels are considered, while the alternative statement shows that the same gate performs the AND function when logic 0 voltage levels are considered. It is called an OR gate because the logic 1 condition is generally more significant; however, it does perform the function "C is logic 0 when A is logic 0 AND B is logic 0."

The symbol in Fig. 36b is an alternative symbol which may be used in logic diagrams to represent an OR gate when the logic 0 conditions show the overall operation of a logic circuit more clearly. The shape of the symbol indicates the AND function and the state indicators show that this function is performed when the logic 0 voltage levels are considered. The shape of the normal symbol (Fig. 36a) indicates the OR function, and the absence of state indicators shows that the function is performed when logic 1's are considered.

(a) NORMAL SYMBOL – shows operation for logic 1 conditions.

(b) ALTERNATIVE SYMBOL – shows operation for logic 0 conditions.

FIG. 36. OR GATE – NORMAL AND ALTERNATIVE SYMBOLS.

13.4 INVERTER. Table 24 is the truth table for an inverter.

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

TABLE 24. INVERTER TRUTH TABLE.

The word statements derived from an inverter truth table are shown below.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
$\bar{A}$ is logic 0 when	$\bar{A}$ is logic 1 when
A is logic 1	A is logic 0

A choice is made between inverter symbols in Fig. 37 to represent inverters in logic diagrams.

(a) NORMAL SYMBOL – shows operation for logic 0 output.

(b) ALTERNATIVE SYMBOL – shows operation for logic 1 output.

FIG. 37. INVERTER – NORMAL AND ALTERNATIVE SYMBOLS.

13.5 NAND GATE. Table 25 is the truth table for a NAND gate.

Inputs A	Inputs B	Output C
0	0	1
0	1	1
1	0	1
1	1	0

TABLE 25. NAND GATE TRUTH TABLE.

The word statements below are derived by examining the NAND gate truth table.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
C is logic 0 when	C is logic 1 when
A is logic 1	A is logic 0
AND	OR
B is logic 1	B is logic 0

The normal word statement shows that a NAND gate performs the AND function when logic 1 voltage levels are connected to its input, and the resultant output voltage is logic 0. This operating condition is shown by the normal NAND gate symbol in Fig. 38a. The inputs have no state indicators and this shows that the AND function is performed when the inputs are at the logic 1 voltage level. The state indicator on the output shows that a logic 0 voltage level is produced when the AND function is performed.

(a) NORMAL SYMBOL – shows operation for logic 0 output.

(b) ALTERNATIVE SYMBOL – shows operation for logic 1 output.

FIG. 38. NAND GATE – NORMAL AND ALTERNATIVE SYMBOLS.

The alternative word statement shows that a NAND gate performs the OR function when logic 0 voltage levels are considered on the inputs, and the resultant output voltage is logic 1. This operating condition is shown by the symbol in Fig. 38b. The shape of the symbol indicates that the OR function is performed, and the state indicators show that logic 0 voltage levels are required on the inputs to perform this function. The absence of a state indicator on the output shows that a logic 1 voltage level is developed on the output when one input is at logic 0.

13.6 NOR GATE. The truth table for a NOR gate is shown in Table 26.

Inputs A	Inputs B	Output C
0	0	1
0	1	0
1	0	0
1	1	0

TABLE 26. NOR GATE TRUTH TABLE.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
C is logic 0 when	C is logic 1 when
A is logic 1	A is logic 0
OR	AND
B is logic 1	B is logic 0

The normal word statement shows a NOR gate performs the OR function to produce a logic 0 on its output. This operating condition is shown by the normal NOR gate symbol in Fig. 39a.

(a) NORMAL SYMBOL – shows operation for logic 0 output.

(b) ALTERNATIVE SYMBOL – shows operation for logic 1 output.

FIG. 39. NOR GATE – NORMAL AND ALTERNATIVE SYMBOLS.

The alternative word statement shows that a NOR gate requires logic 0 input conditions to perform the AND function and produce a logic 1 on its output. This operating condition is shown by the alternative NOR gate symbol in Fig. 39b. The symbol shape indicates the AND function, the state indicators on the inputs show that this function is performed when the inputs are at logic 0, and the absence of a state indicator on the output shows that the AND condition is satisfied when the output is at logic 1.

13.7 SPECIAL GATES. Sometimes special logic circuits are encountered which cannot be classified as AND, OR, NAND or NOR gates. They are represented in logic diagrams by using the AND or OR shape symbols, which show the function performed, and state indicators to show whether logic 1's or logic 0's are required on the inputs and outputs to perform the function indicated. Table 27 is the truth table for one of these special gates, and Fig. 40 shows the logic symbols.

Inputs A	Inputs B	Output C
0	0	0
0	1	0
1	0	1
1	1	0

TABLE 27. SPECIAL GATE TRUTH TABLE.

The following word statements are derived from Table 27 and explain the input conditions required to make the output logic 1 and logic 0 respectively.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
C is logic 1 when	C is logic 0 when
A is logic 1	A is logic 0
AND	OR
B is logic 0	B is logic 1

The logic symbols shown in Fig. 40 are developed from these statements.

(a) NORMAL SYMBOL – shows operation for logic 1 output.

(b) ALTERNATIVE SYMBOL – shows operation for logic 0 output.

FIG. 40. SPECIAL GATE SYMBOLS.

When the gate is used to perform the function represented by the symbol in Fig. 40a it is called an "Inhibit Gate", because a logic 1 on input B (the inhibiting input) prevents (or "inhibits") the output from becoming logic 1.

13.8 GATE COMBINATIONS. The operation of any gate combination can be expressed by two word statements. For example the operation of the circuit in Fig. 41a is explained by either of the two word statements below.

NORMAL WORD STATEMENT	ALTERNATIVE WORD STATEMENT
F is logic 1 when	F is logic 0 when
A, AND B AND C are logic 1	A, OR B OR C is logic 0
ORD	AND
when D AND E are logic 1	when D OR E is logic 0

(a) NORMAL SYMBOL – shows operation for logic 1 output.

(b) ALTERNATIVE SYMBOL – shows operation for logic 0 output.

FIG. 41. GATE COMBINATION.

When the circuit connected to the output of Fig. 41a is activated by a logic 0, the second word statement is more useful. Also, the use of the alternative symbols for the AND and OR gates, as shown in Fig. 41b, helps to show more clearly the input conditions required to produce a logic 0 at the output.

Question 8 on page 39 is an exercise in reading logic diagrams which use the alternative logic symbols.

14. INTRODUCTION TO BOOLEAN MANIPULATION TECHNIQUES.

14.1 COMPLEX LOGIC CIRCUITS. Logic circuits often contain more elements than would appear necessary to perform the required logic function. For example, Fig. 21 shows how three NAND gates are used to perform the OR function. In this example, a Boolean expression $\overline{\bar{A}.\bar{B}}$ is developed which describes the actual NAND gate combination used, and the equivalence of $\overline{\bar{A}.\bar{B}}$ to to A+B was proven with truth tables.

In practice, however, the Boolean expression describing the actual gate combination in a complex circuit may not readily indicate the basic function of the circuit, and the basic function may not be known. For example, consider the circuit shown in Fig. 42. The Boolean expression $\overline{A + \bar{B}.\bar{C}}$ developed at the output of this circuit describes the actual circuit used, but the input conditions required to make the output significant are not immediately obvious. The operation of this complex circuit can be more readily understood if the basic function of the circuit is available.

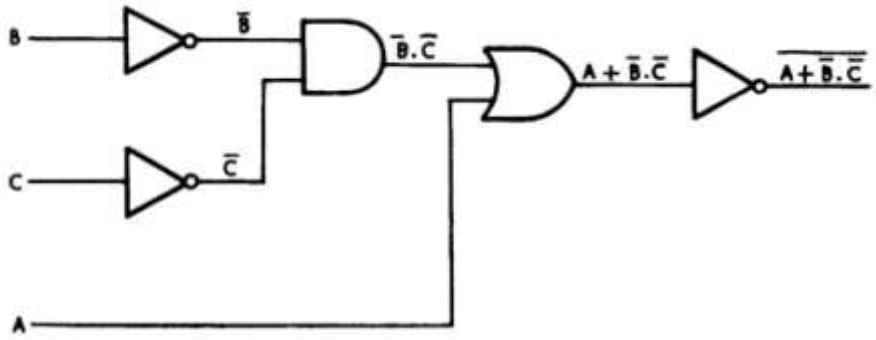


FIG. 42. CIRCUIT OF $\overline{A + \bar{B}.\bar{C}}$.

14.2 BASIC FUNCTION OF COMPLEX LOGIC CIRCUIT. One method of obtaining the basic function of a complex Boolean expression is to apply known Boolean identities and laws to simplify the expression. For example, the application of Boolean identities and laws to $\overline{A + \bar{B}.\bar{C}}$ enables it to be simplified to $\bar{A}.(B + C)$. This expression reveals that a significant (logic 1) output is obtained in Fig. 42 when $\bar{A}$ is logic 1 AND when either B OR C is logic 1. Note that $\bar{A}$ is logic 1 when A is logic 0.

14.3 FUNCTIONAL LOGIC DIAGRAM. A functional logic diagram can be drawn to represent a simplified Boolean expression. A functional logic diagram is a simplified diagram having the same truth table as the actual logic circuit from which it was derived. For example, Fig. 43 is a functional logic diagram of Fig. 42 as it represents the Boolean expression $\bar{A}.(B + C)$, which is the simplified expression of $\overline{A + \bar{B}.\bar{C}}$.

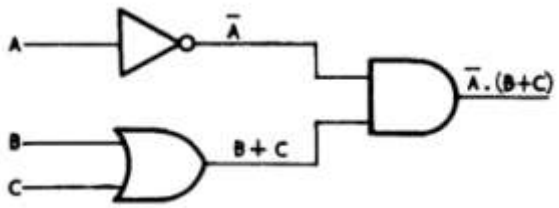


FIG. 43. FUNCTIOAL LOGIC DIAGRAM OF FIG> 42.

It should be realised that some logic circuits may already be in their simplest form. In these cases the function of the circuit is obtained directly from the Boolean expression describing the circuit.

14.4 BOOLEAN IDENTITIES. In this section of the paper it is intended to introduce some of the basic identities commonly used in converting complex Boolean expressions to their simplest function. A Boolean identity equates two expressions which are equal for all possible combinations of their variables. The equivalence of the two expressions in a Boolean identity can always be proven with the aid of truth tables.

14.5 AND FUNCTION IDENTITIES. Consider a two input AND gate with one input permanently tied to logic 1 as shown in Fig. 44.

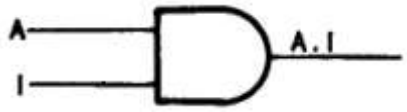


FIG. 44.

The identity derived from this expression is:-

$$A.1 = A \quad .. \quad .. \quad .. \quad .. \quad (1)$$

This means that the output of the gate is always dependent on the logic condition at input A. This can be proven as follows. The variable A has two states, logic 1 and logic 0. Substituting these in the identity A.1 = A we get:-

1.1 = 1, when A is logic 1, and
0.1 = 0, when A is logic 0.

This identity is true because the output is equal to A for both states.

Now consider an AND gate with one input permanently at logic 0, as shown in Fig. 45. The output is always at logic 0 regardless of the condition of A. The identity derived from this circuit is:-

$$A.0 = 0 \quad .. \quad .. \quad .. \quad .. \quad (2)$$

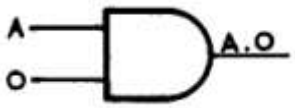


Fig. 45.

When both inputs of an AND gate have the same signal applied, as shown in Fig. 46, the identity applying to the circuit is:-

$$A.A = A \quad .. \quad .. \quad .. \quad .. \quad (3)$$

When A is logic 1 the output is logic 1, and when A is logic 0 the output is logic 0, thus the output is always equal to the condition of A.

Fig. 46.

Fig. 47 shows a signal $\bar{A}$, and its complement A applied to an AND gate. Since it is impossible for both A and $\bar{A}$ to be at logic 1 at the same time, the output of the gate must always be at logic 0. Therefore, the following identity applies.

$$A.\bar{A} = 0 \quad .. \quad .. \quad .. \quad .. \quad (4)$$

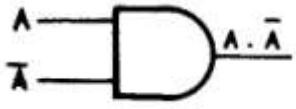


Fig. 47.

14.6 OR FUNCTION IDENTITIES. Consider an OR gate with one input permanently tied to logic 1, as shown in Fig. 48. An OR gate requires logic 1 at one input only to obtain a logic 1 output, therefore, the following identity applies:-

$$A + 1 = 1 \quad .. \quad .. \quad .. \quad .. \quad (5)$$

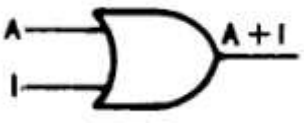


Fig. 48

The condition of the OR gate output in Fig. 49 depends on the logic condition of input A, because the other input is permanently tied to logic 0. If A is at logic 1 the output is at logic 1, and if A is at logic 0 the output is at logic 0, therefore the following identity applies.

$$A + 0 = A \quad .. \quad .. \quad .. \quad .. \quad (6)$$

Fig. 49.

The OR gate in Fig. 50 has the same signal applied to both inputs. Therefore, when A is logic 1 the output is logic 1, and when A is logic 0 the output is logic 0. Thus the output is equal to the logic condition of A, and the following identity applies:-

$$A + A = A \quad .. \quad .. \quad .. \quad .. \quad (7)$$

Fig. 50.

Fig. 51 shows an OR gate with signal A applied to one input, and its complement $\bar{A}$ applied to the other input. Since one of these signals must always be at logic 1, the output of the gate must always be at logic 1. Therefore, the following identity applies:-

$$A + \bar{A} = 1 \quad .. \quad .. \quad .. \quad .. \quad (8)$$

Fig. 51.

14.7 APPLICATION OF AND AND OR IDENTITIES. The following is a summary of identities applying to the AND and OR functions.

AND	OR
$A.1 = A \quad .. \quad .. \quad .. \quad .. \quad (1)$	$A + 1 = 1 \quad .. \quad .. \quad .. \quad .. \quad (5)$
$A.0 = 0 \quad .. \quad .. \quad .. \quad .. \quad (2)$	$A + 0 = A \quad .. \quad .. \quad .. \quad .. \quad (6)$
$A.A = A \quad .. \quad .. \quad .. \quad .. \quad (3)$	$A + A = A \quad .. \quad .. \quad .. \quad .. \quad (7)$
$A.\bar{A} = 0 \quad .. \quad .. \quad .. \quad .. \quad (4)$	$A + \bar{A} = 1 \quad .. \quad .. \quad .. \quad .. \quad (8)$

Some examples of the application of these identities are as follows:-

Example 1. Simplify A.1 + A.A + B.

Substitute A for A.1 (identity 1).

= A + A.A + B.

Substitute A for A.A (identity 3).

= A + A + B.

Substitute A for A + A (identity 7).

= A + B (answer).

Example 2. Simplify (B + 0).(B + 1).C.

Substitute B for B + 0 (identity 6) and 1 for B + 1 (identity 5).

= B.1.C

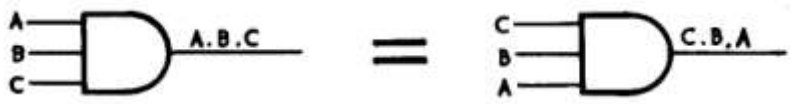
Substitute B for B.1 (identity 1).

= B.C (answer).

Further examples of simplifying expression with AND and OR identities are contained in Question 9 on page 39.

14.8 COMMUTATIVE LAW. This law states that the inputs to a logic gate may be listed in any order without affecting the logical operation. For example, consider the three input AND gates in figs. 52a and 52b. Since both these gates have the same signals applied, they perform the same logical operation. Therefore it is true to say that:-

$$A.B.C = C.B.A \quad .. \quad .. \quad .. \quad .. \quad (9)$$

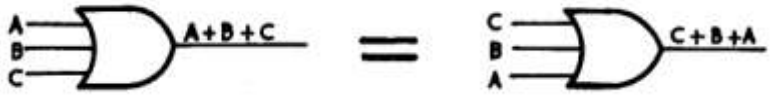


(a) (b)

FIG. 52. COMMUTATIVE LAW APPLIED TO AND FUNCTION.

The commutative law also applies to ORed variables, as shown in Fig. 53. The logic function remains the same regardless of the order in which the input variables are listed. This can be expressed in the following identity.

$$A + B + C = C + B + A \quad .. \quad .. \quad .. \quad .. \quad (10)$$



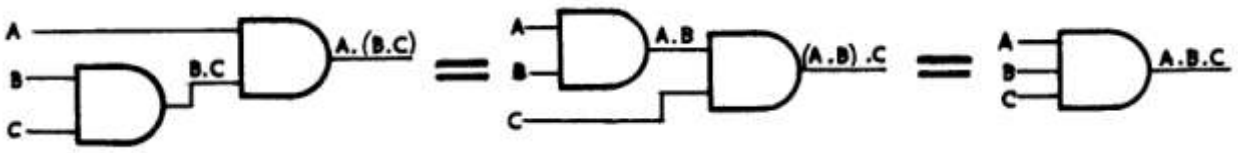
(a) (b)

FIG. 53. COMMUTATIVE LAW APPLIED TO OR FUNCTION.

14.9 ASSOCIATIVE LAW. This law states that ANDed variables or ORed variables can be grouped together in any order in a Boolean expression. This can be expressed in logical form for three ANDed variables as follows:-

$$A.(B.C) = (A.B).C = A.B.C \quad .. \quad .. \quad .. \quad .. \quad (11)$$

Each of the expressions has a different logic circuit, as shown in Fig. 54. However the basic logic function of Figs. 54a and 54b is exactly the same as Fig. 54c.



(a) (b) (c)

FIG. 54. ASSOCIATIVE LAW APPLIED TO AND FUNCTION.

The associative law can be applied to ORed variables as follows:-

$$A + (B + C) = (A + B) + C = A + B + C \quad .. \quad .. \quad .. \quad .. \quad (12)$$

The logic diagrams representing each of the expressions in identity 12 are shown in Fig. 55. Each of these circuits has exactly the same basic function which is A + B + C, as shown in Fig. 55c.

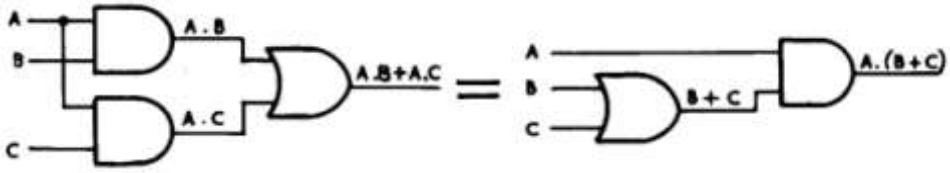
(a) (b) (c)

FIG. 55. ASSOCIATIVE LAW APPLIED TO OR FUNCTION.

14.10 DISTRIBUTIVE LAW. The distributive law is expressed as follows:-

$$A.B + A.C = A.(B + C) \quad .. \quad .. \quad .. \quad .. \quad (13)$$

Figs. 56a and 56b show the logic circuits used to represent each expression in the identity. Although each circuit is different, they both have the same function. The distributive law shows that normal algebraic factorising techniques can be applied to a Boolean expression.



(a) (b)

FIG. 56. DISTRIBUTIVE LAW.

14.11 The following are further examples of simplification using some of the previously stated identities.

Example 1. Simplify A.(A + B)

Substitute A.A + A.B for A.(A + B) (identity 13).

= A.A + A + B

Substitute A. for A.A (identity 3).

= A + A.B

Factorise the expression (identity 13).

= A.(1 + B)

Substitute 1 for (1 + B) (identity 5).

= A.1

Substitute A for A.1 (identity 1).

= A (answer).

Example 2. Simplify $\bar{A}.B + \bar{A}.\bar{B}$

Factorise the expression (identity 13).

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

Substitute 1 for $B + \bar{B}$ (identity 8).

= $\bar{A}.1$

Substitute $\bar{A}$ for $\bar{A}.1$ (identity 1).

= $\bar{A}$ (answer).

Question 10 on page 39 contains further examples on Boolean simplification.

14.12 De MORGAN'S LAWS. Generally, the basic function of a logic circuit can not be readily interpreted from a negated expression situated at an intermediate point, or at an output. For example, the basic function of the negated expression $\overline{(A + B).C.D}$ is not immediately obvious. De Morgan's theorem, which is usually expressed in terms of two laws, provides a method by which a complex negated expression can be replaced by an alternative simplified expression. De Morgan's theorem implies that when an expression is negated, it can be replaced by another expression in which each variable is negated, each AND is changed to OR, and each OR is changed to AND. The two laws derived from this theorem are as follows:-

$$\overline{A.B} = \overline{A} + \overline{B} \quad .. \quad .. \quad .. \quad .. \quad (14)$$

$$\overline{A+B} = \overline{A}.\overline{B} \quad .. \quad .. \quad .. \quad .. \quad (15)$$

De Morgan's Laws are frequently used identities in Boolean manipulation.

The validity of the first of De Morgan's laws ($\overline{A.B} = \overline{A} + \overline{B}$) can be shown with the aid of a two input NAND gate (Fig. 57) and truth tables. The normal expression on the output of this gate is $\overline{A.B}$, but this expression does not readily reveal the input conditions required to make the output significant. However, and equivalent expression for $\overline{A.B}$ can be obtained by applying De Morgan's theorem as follows: A is changed to $\bar{A}$, B is changed to $\overline{B}$, and the AND is changed to an OR. The new expression on the output of the NAND gate becomes $\overline{A} + \overline{B}$, which indicates that the output is significant when $\bar{A}$ is logic 1 OR $\overline{B}$ is logic 1, that is, when A is logic 0 OR B is logic 0. The equivalence of the two expressions $\overline{A.B}$ and $\overline{A} + \overline{B}$ is proven in truth tables 28 and 29.

FIG. 57.

A	B	A.B	$\overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Table 28. TRUTH TABLE FOR $\overline{A.B}$.

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

TABLE 29. TRUTH TABLE $\overline{A} + \overline{B}$.

The validity of the second of De Morgan's Laws ($\overline{A + B} = \overline{A}.\overline{B}$) can be shown with the aid of a NOR gate (Fig. 58) and truth tables. The input conditions required to make the output significant are not obvious from the output expression $\overline{A+B}$. Another expression can be obtained for $\overline{A+B}$ by applying De Morgan's theorem as follows: A is inverted to $\bar{A}$, B is inverted to $\overline{B}$, and the OR is changed to an AND. The new expression on the output of the gate becomes $\overline{A}.\overline{B}$, which indicates that the output is significant when $\overline{A}$ is logic 1 AND $\overline{B}$ is logic 1, that is, when A is logic 0 and B is logic 0, The equivalence of $\overline{A+B}$ and $\overline{A}.\overline{B}$ is shown in truth tables 30 and 31.

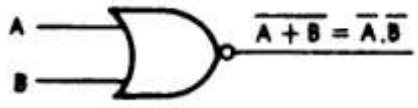


FIG. 57.

A	B	A + B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Table 30. TRUTH TABLE FOR $\overline{A+B}$.

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

TABLE 31. TRUTH TABLE $\overline{A}.\overline{B}$.

It should be realised that De Morgan's Laws (identities 14 and 15) can be written as follows, without altering their meaning.

$$\overline{A.B.C.D\,etc} = \overline{A} + \overline{B} + \overline{B} + \overline{C} + \overline{etc} \quad \text{(identity 14)}$$

$$\overline{A+B+C+D\ etc} = \overline{A}.\overline{B}.\overline{B}.\overline{C}.\ \overline{etc} \quad \text{(identity 15)}$$

14.13 Some examples of the applications of De Morgan's Laws are as follows.

Example 1. Simplify $\overline{D+E+F}$.

Applying identity 15 ($\overline{A+B} = \overline{A}.\overline{B}$)

$= \overline{D}.\overline{E}.\overline{F}.$ (answer).

When applying De Morgan's Laws to complex expressions, it is essential that parts of the expressions which are bracketed, or would normally be considered as being bracketed, are considered as complete units until all "De Morganising" has been completed, then the units can be simplified. This is demonstrated in example 2.

Example 2. Simplify $\overline{A.B+C.D}$.

Since an AND function takes precedence over an OR function in a Boolean expression, A.B and C.D are considered as two separate units for the application of De Morgan's Laws, as follows:-

Let x = A.B and,

y = C.D.

Substitute x and y into the expression

$= \overline{x+y}$

Apply identity 15 ($\overline{A+B} = \overline{A}.\overline{B}$) by substituting x for A, and y for B.

$= \overline{x}.\overline{y}$

Re-substitute A.B for x and C.D for y

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

Substitute $\overline{A} + \overline{B}$ for $\overline{A.B}$; and $\overline{C} + \overline{D}$ for $\overline{C.D}$ (identity 14)

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

Generally, the procedure shown in example 2 can be simplified by mentally recognising the units in an expression, and then applying De Morgan's Laws direct to the expression, and shown in examples 3 and 4.

Example 3. Simplify $\overline{(C + D).(E + F)}$.

As (C + D) and (E + F) are bracketed, they are considered as complete units.

Apply identity 14 ($\overline{A.B} = \overline{A} + \overline{B}$) by substituting (C + D) for A, and (E + F) for B.

$= \overline{C+D} + \overline{E+F}.$ Substitute $\overline{C}.\overline{D}$ for $\overline{C+D}$; and $\overline{E}.\overline{F}$ for $\overline{E+F}$ (identity 15)

$= \overline{C}.\overline{D} + \overline{E}.\overline{F}$ (Answer)

Example 4. Simplify $\overline{(A + B + C).D}$

(A + B + C) and (D) are considered as separate units.

Apply identity 14 ($\overline{A.B} = \overline{A} + \overline{B}$) by substituting (A + B + C) for A and (D) for B.

$= \overline{A+B+C} + \overline{D}$

Substitute $\overline{A}.\overline{B}.\overline{C}$ for $\overline{A+B+C}$ (identity 14)

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

Sometimes further simplification can be applied after De Morgan's Laws have been used on an expression, as shown in example 5.

Example 5. Simplify $\overline{\overline{A.B}.\overline{A.C}}$

In this case $\overline{A.B}$ and $\overline{A.C}$ are considered as complete units.

Apply identity 14 ($\overline{A.B} = \overline{A} + \overline{B}$) by substituting $\overline{A.B}$ for A, and $\overline{A.C}$ for B

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

= A.B + A.C

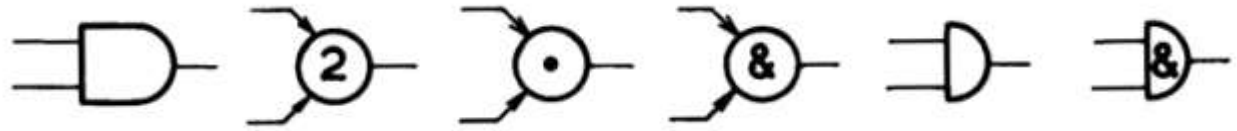
Simplify this by applying identity 13.

= A.(B + C) (Answer).

Further examples on De Morgan's Laws are contained in Question 11 on page 40.

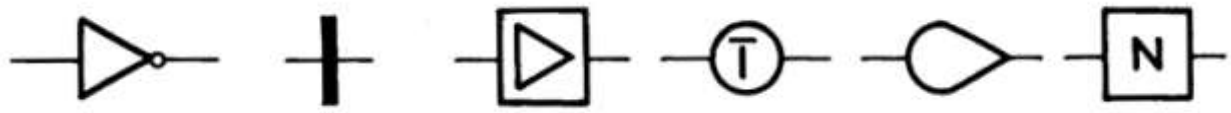
15. LOGIC SYMBOLS.

15.1 A variety of logic symbols are used in the documentation provided with logic equipment supplied to the department. Some of these symbols are shown below. The figures shown in some of the symbols indicate the number of inputs which must be at logic 1 to activate the gate.

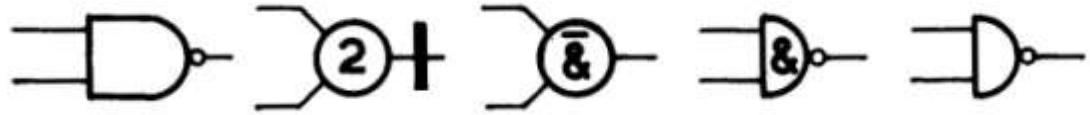


AND GATE SYMBOLS.

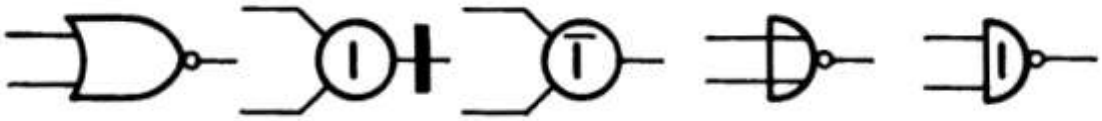
OR GATE SYMBOLS.



INVERTER SYMBOLS.



NAND GATE SYMBOLS.

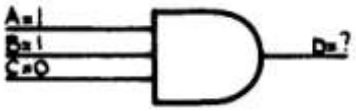


NOR GATE SYMBOLS.

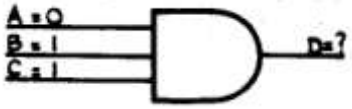
FIG. 59. LOGIC SYMBOLS.

16. TEST QUESTIONS.

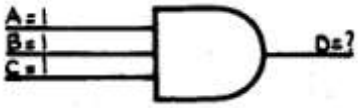
Q.1 Determine whether the outputs of the circuits in Fig. 60 are logic 1 or logic 0. (ANS on page 40).



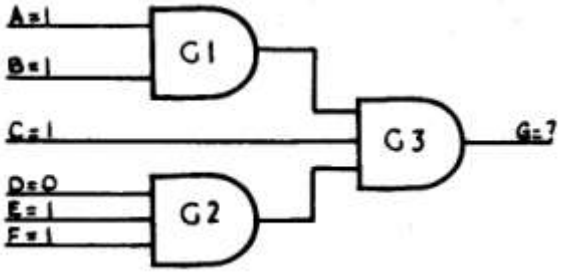
(a)



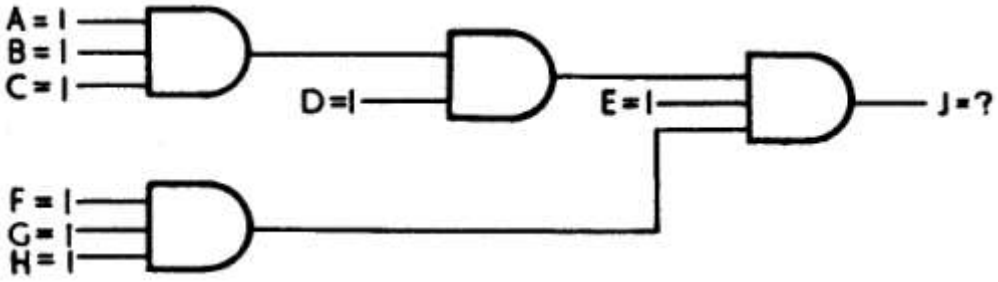
(b)



(c)



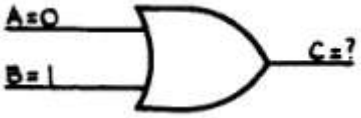
(d)



(e)

FIG. 60.

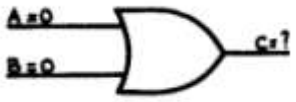
Q.2 Determine whether the outputs of the circuits in Fig. 61 are logic 1 or logic 0. (ANS on page 40).



(a)

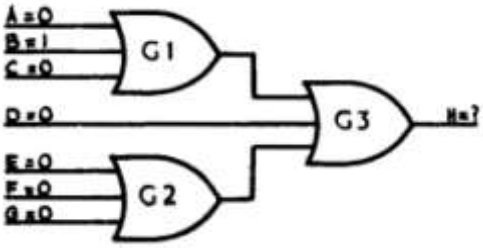
FIG. 61.

Q.2 (contd.)



(b)

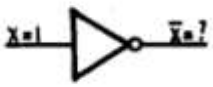
(c)



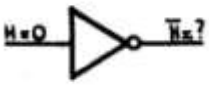
(b)

FIG. 61. (contd.)

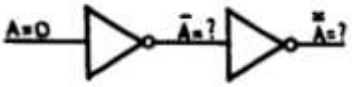
Q.3 Determine whether the outputs of the circuits in Fig. 62 are logic 1 or logic 0. What relationship exists between A and $\overline{\overline{A}}$? (ANS on page 40).



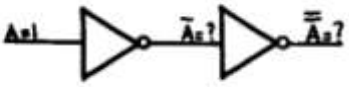
(a)



(b)



(a)



(b)

FIG. 62.

Q.4 (a) Draw the logic diagram described by the following expressions. (ANS on page 40).

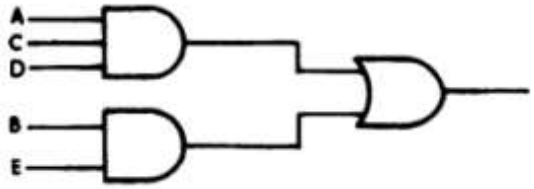
(i) A.B + C

(ii) A.C + A.B

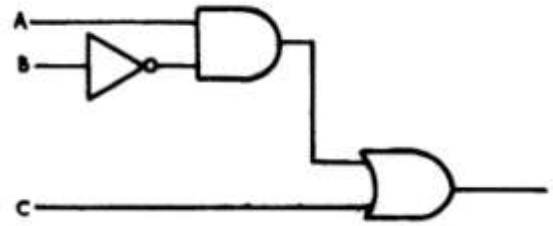
(iii) B.C + A.$\overline{C}$.D

Q.4 (b) Write Boolean expressions which describe the logic circuits shown in Fig. 63 (ANS on page 41).

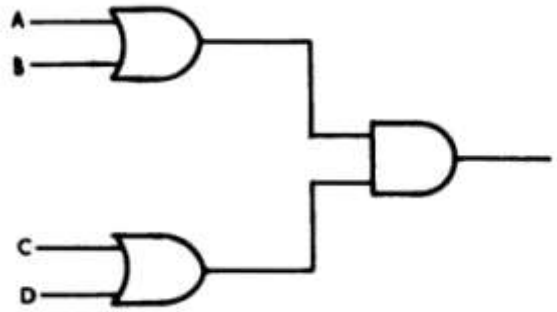
(i)



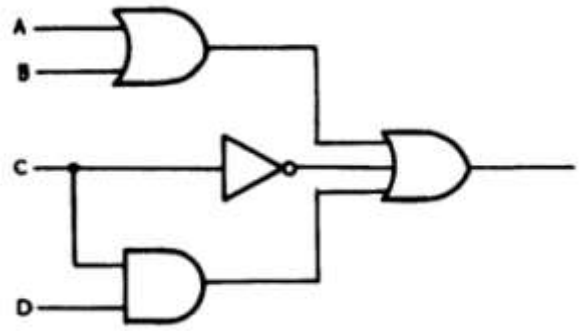
(ii)



(iii)



(iv)



(v)

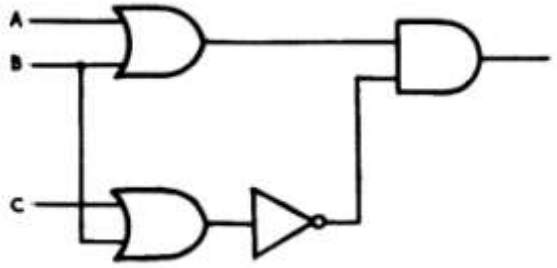


FIG. 63.

Q.4 (c) Referring to Fig. 64 cross pout the incorrect answers in the following statements for the output condition on D. (ANS on page 42).

(i) If A = logic 0, D = logic 0 / depends on other inputs / logic 1

(ii) If B = logic 0, D = Logic 0 / depends on other inputs / logic 1

(iii) If C = logic 0, D = Logic 0 / depends on other inputs / logic 1

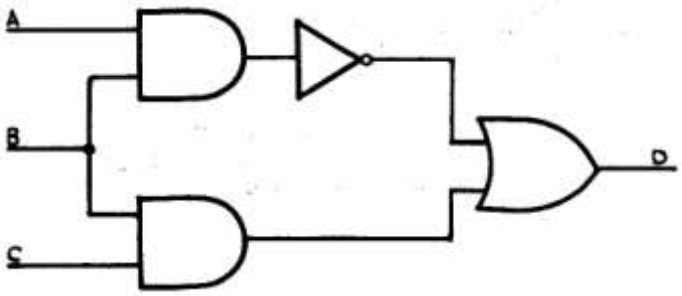


FIG. 64.

Q.4 (d) Referring to Fig. 65 cross pout the incorrect answers in the following statements for the output condition on F. (ANS on page 42).

(i) If B = logic 0, F = logic 0 / depends on other inputs / logic 1

(ii) If C = logic 0, F = logic 0 / depends on other inputs / logic 1

(iii) If A,B and C are all logic 1, F = logic 0 / depends on other inputs / logic 1

(iv) If D = logic 1, and E = logic 1, F = logic 0 / depends on other inputs / logic 1

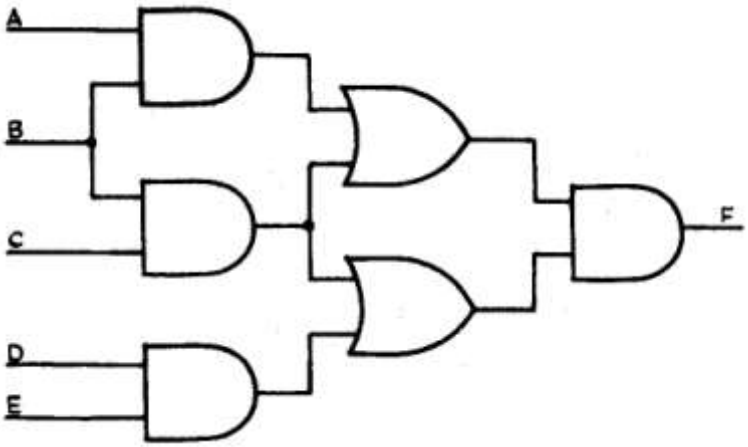
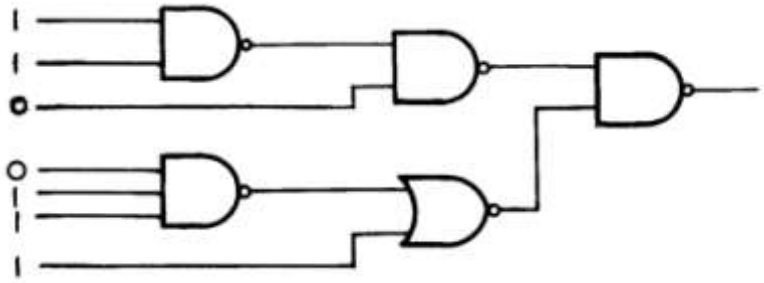


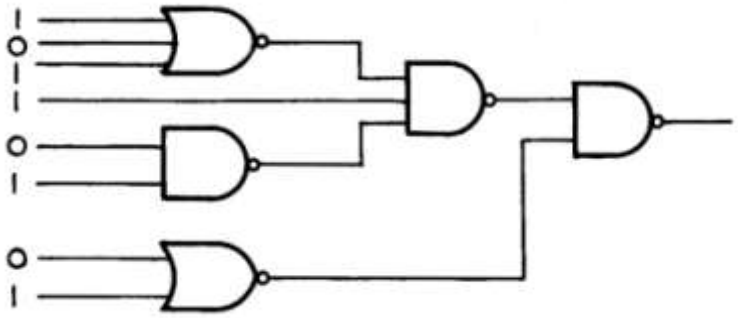
FIG. 65.

Q.5 Construct a truth table for the circuit in Fig. 63 (ii). (ANS on page 42).

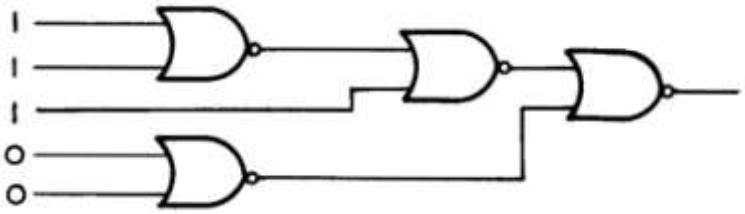
Q.6 Determine the logic condition on the outputs of the circuits in Fig. 66. Show the logic condition of the intermediate pints in each circuit. (ANS on page 42).



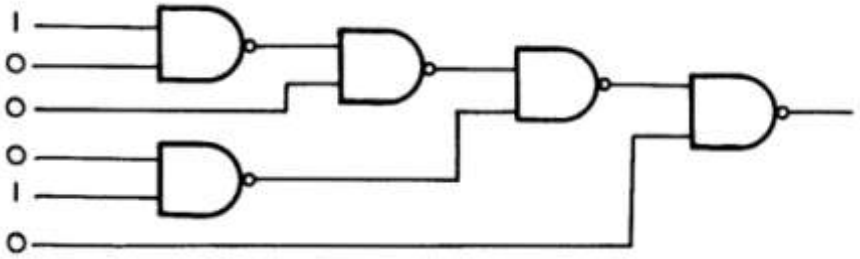
(a).



(b).



(c).



(d).

FIG. 66.

Q.7 (a) What function is performed by the switch contacts in Fig. 67? (ANS on page 43).

(a) (b)

FIG. 67.

(b) What function is performed by the logic circuit in Fig. 68? (A Truth Table should be used as an aid to determine the function).

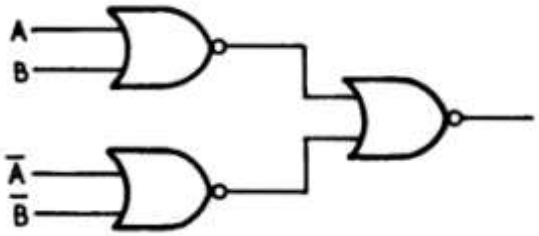
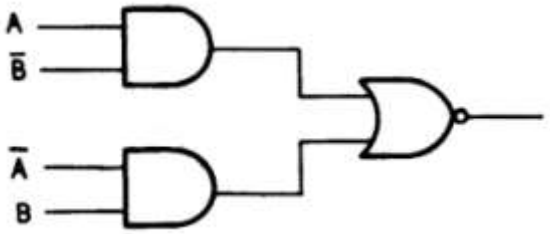
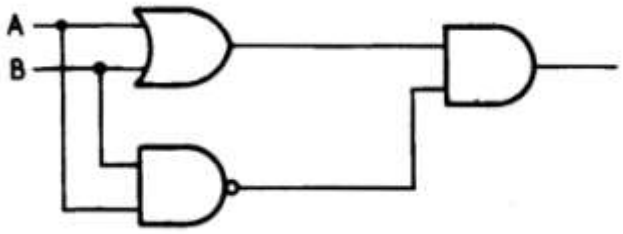


FIG. 68.

(c) What functions are performed by the logic circuits in Fig. 69? (Truth Tables should be used as an aid to determine the functions).



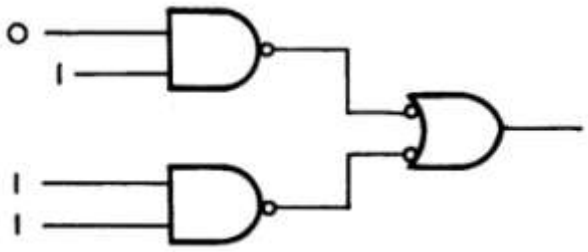
(i).



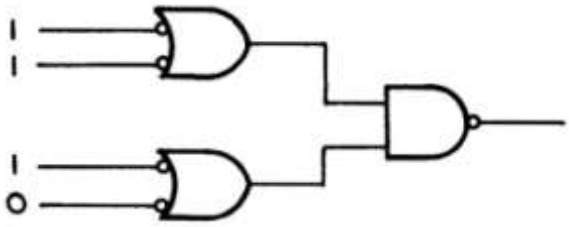
(ii).

FIG. 69.

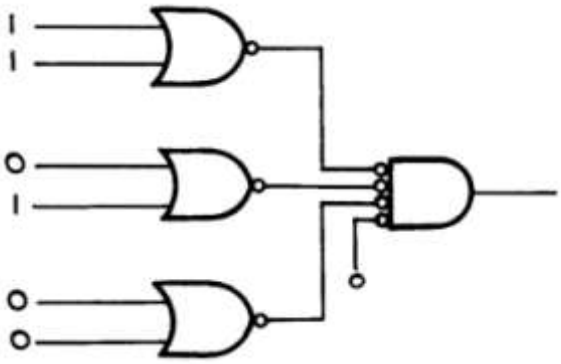
Q.8 Determine the logic condition at the output of the circuits in Fig. 70. Show the logic condition of the intermediate points in each circuit. (ANS on page 43).



(a).



(b).



(c).

FIG. 70.

Q.9 Simplify the following expressions (ANS on page 44).

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

(b) $A + B + \bar{A} + B$

(c) $B.(C + 1).C.B$

Q.10 Simplify the following expressions (ANS on page 44).

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

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

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

Q.11 Simplify the following expressions (ANS on page 44).

(a) $\overline{\overline{A} + \overline{B}}$

(b) $\overline{\overline{A}.\overline{B}.\overline{C}.\overline{D}}$

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

17. ANSWERS TO TEST QUESTIONS

ANS.1 (a) D = 0 **(b) D = 0** **(c) D = 1**

(d) G = 0 (The output of G2 is logic 0, therefore one of the inputs to G3 is logic 0. All the inputs of G3 would have to be at logic 1 to obtain a logic 1 output).

(c) J = 1

ANS.2 (a) C = 1 **(b) C = 0** **(c) D = 1**

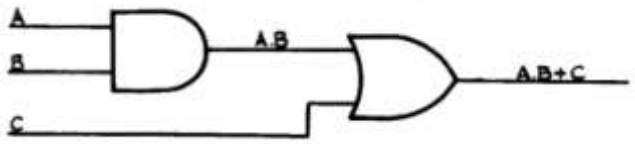
(b) H = 1 (The output of G1 is logic 1, therefore, one of the inputs of G3 is logic 1. G3 only needs one of its inputs at logic 1 to produce a logic 1 output).

ANS.3 (a) $\overline{X} = 0$ **(b)** $\overline{H} = 0$

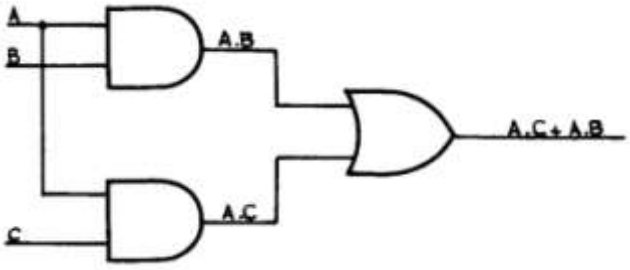
(c) $\bar{A} = 1$, $\overline{\overline{A}} = 0$, $\overline{\overline{A}} = A$ **A has been inverted twice and a double inversion restores an expression back to its original form.**

(d) $\bar{A} = 0$, $\overline{\overline{A}} = 1$

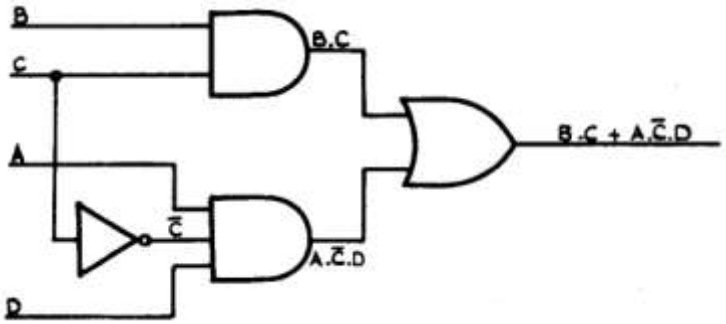
ANS.4 (a) (i) The expression A.B + C means that A and B are ANDed together and the output (A.B) of the AND gate is ORed with C.



(ii) The expression A.C + A.B means that A and B are ANDed together, A and C are ANDed together, and the outputs of both AND gates A.C and A.B are ORed together.

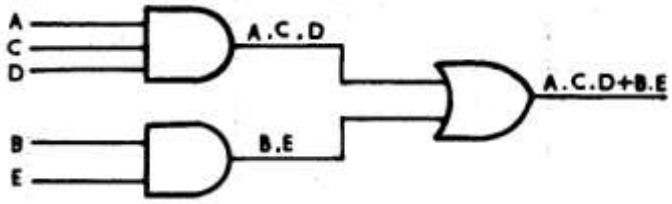


ANS.4 (a) (iii)

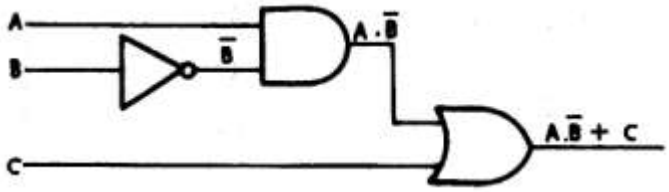


ANS.4 (b) Each of the figures in Q4(b) has been reproduced to show how the final expression is solved by progressively considering the intermediate points.

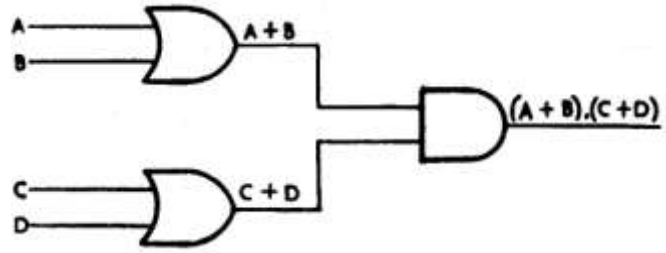
(i)



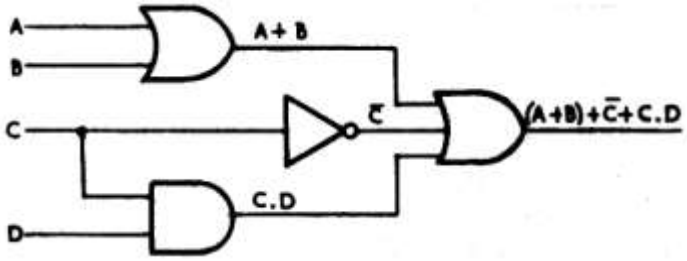
(ii)



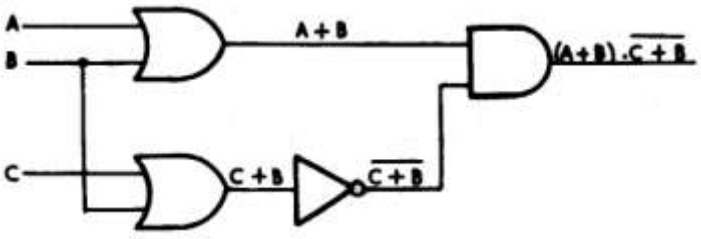
(iii)



(iv)



(v)



ANS.4 (c) Referring to Fig. 64.

(i) If A = 0, D = 1

(ii) If B = 0, D = 1

(iii) If C = 0, D depends on other inputs.

ANS.4 (d) Referring to Fig. 65.

(i) If B = 0, F = 0

(ii) If C = 0, F depends on other inputs.

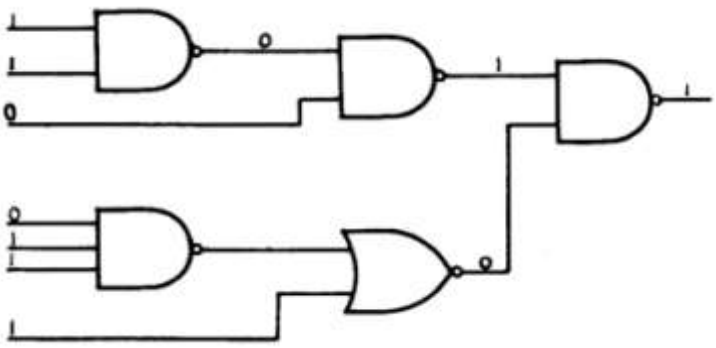
(iii) If A, B and C are all 1, F = 1.

(iV) If D = 1, and E = 1, F depends on other inputs.

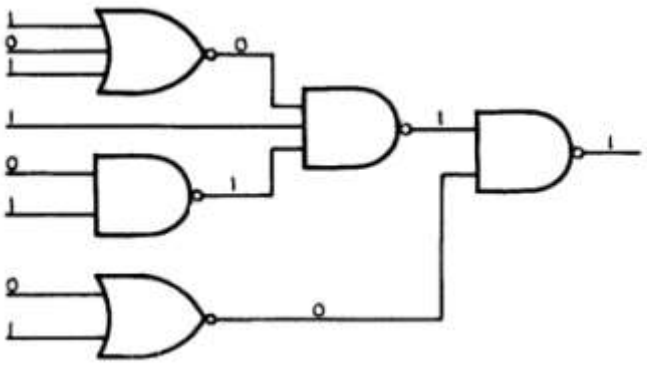
ANS.5

Inputs			Intermediate Points		Outputs
A	B	C	$\overline{B}$	$A.\overline{B}$	$D = A.\overline{B} + C$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

ANS.6 (a)

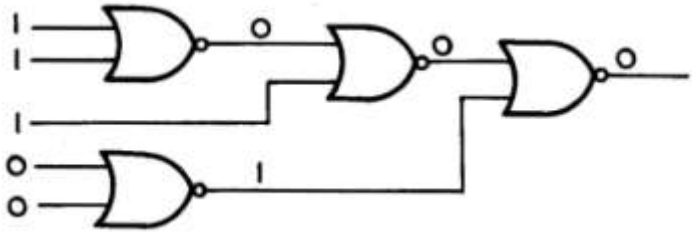


(b)

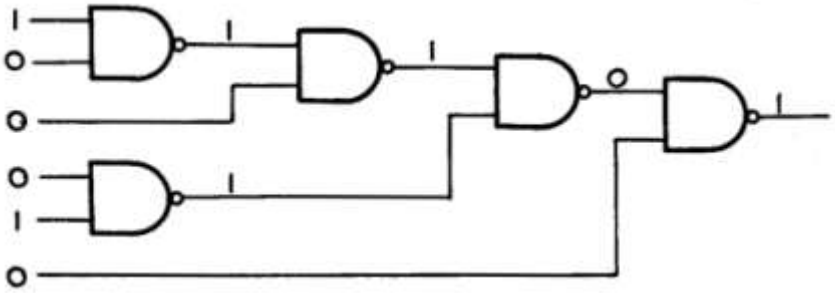


ANS.6 (contd.)

(c)



(d)

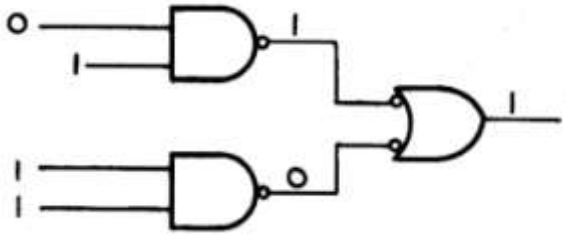


ANS.7 (a) The contacts in Fig. 67 (i) perform the comparator function, and those in Fig. 67 (ii) the Exclusive-OR function.

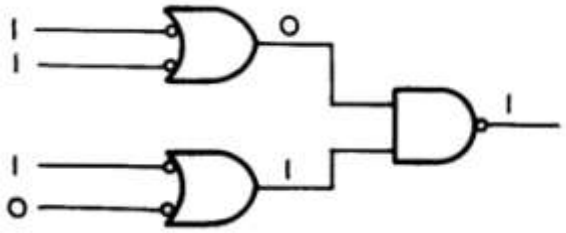
(b) The circuit in Fig. 68 performs the Exclusive-OR function.

(c) The circuit in Fig. 69 (i) performs the comparator function. The circuit in Fig. 69 (ii) performs the Exclusive-OR function.

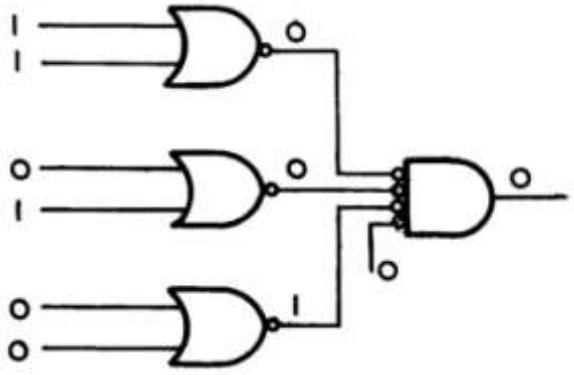
ANS.8 (a)



(b)



(c)



ANS.9 **(a)** $A.B.\bar{A} + C.B.C$ .. .. (Substitute $A.\bar{A} = 0$; $C.C = C$)

$= B.0 + B.C$ .. .. (" ($B.0 = 0$)

$= 0 + B.C$ .. .. (" ($0 + B.C = B.C$)

$= B.C$

(b) $A + B + \bar{A} + B$ .. .. (" ($A + \bar{A} = 1$; $B + B = B$)

$= 1 + B$ .. .. (" ($B + 1 = 1$)

$= 1$

(c) $B.(C + 1).C.B$ .. .. (" ($C + 1 = 1$)

$= B.1.C.B$ .. .. (" ($B.B = B$; $C.1 = c$)

$= B.C$

ANS.10 **(a)** $A.B + B.C + \bar{C}.B$ .. .. (" ($C + \bar{C} = 1$)

$= B.(A + 1)$ .. .. (" ($A + 1 = 1$)

$= B.1$ .. .. (" ($B.1 = B$)

$= B$

(b) $A.(B.C + B.\bar{C})$ .. .. (" ($C + \bar{C} = 1$)

$= A.B.1$ .. .. (" ($B.1 = B$)

$= A.B$

(c) $A.\bar{A} + A.B + \bar{A}.B + B.B + C.A$ (" ($A.\bar{A} = 0$; $B.B = b$)

$= 0 + A.B + \bar{A}.B + B + C.A$

$= B.(A + \bar{A} + 1) + C.A$ (" ($A + \bar{A} = 1$)

$= B(.(1 + 1) + C.A$ .. (" ($1 + 1 = 1$)

$= B + C.A$

ANS.11 **(a)** $\overline{\bar{A} + \bar{B}}$

Apply identity 15.

$= \bar{\bar{A}}.\bar{\bar{B}}$

$= A.B$ (Answer)

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

Apply identity 14.

$= \bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}} + \bar{\bar{D}}$

$= A + B + C + D$ (Answer)

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

Apply identity 14 ($\overline{A.B.C} = \bar{A} + \bar{B} + \bar{C}$) by substituting ($\bar{A} + \bar{B}$) for A, ($\bar{A} + \bar{C}$) for B and ($\bar{A} + \bar{D}$) for C.

$= \overline{\bar{A} + \bar{B}} + \overline{\bar{A} + \bar{C}} + \overline{\bar{A} + \bar{D}}$

Apply identity 15 to each unit.

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

$= A.B + A.C + A.D$

Apply identity 13.

$= A(B + C + D)$ (Answer).