TPH 0171

Engineering Training Publication

Electronic Logic

Principles 1 - ETP 0072

Issue 1, 1978

Prepared by:
Training Group.

Authorised by:
Superintending Engineer,
Operations Planning and Programming Branch, HQ.

Distribution Code: 4L
Filing Category: K

Telecom Australia

Telecom Australia
ENGINEERING TRAINING
210 Kingsway South Melbourne 3205

Training
Publication
ETP 0072
1978

ELECTRONIC LOGIC PRINCIPLES 1

1. INTRODUCTION

1.1 Automatic Telephone Exchanges were one of the first machines in which decision making was left to the machine. Over the years telecommunications has lifted the "intelligence" of its automatic services by the application of more sophisticated decision making or logic techniques. For example, crossbar exchanges provide more facilities than step-by-step exchanges because more relay logic is used. From the complexity of a computer to the simplicity of a lamp control on a telephone, electronic logic circuits give faster, cheaper and more reliable operation in all fields of telecommunications.

1.2 It is important to realise that there is a fundamental difference between the logic circuits in this paper and the traditional relay or electronic circuit. Traditional circuits show individual component wiring and the function of the circuit is interpreted from the configuration of the components. On the other hand, logic circuits show the function, but do not indicate how hardware items achieve the function.

1.3 This publication explains the decision making aspect of electronic logic as a rational thought process of the mind. Emphasis is however placed on those topics which find application in actual hardware, and not on topics such as Boolean Algebra manipulation which is the province of the logic designer. Other publications in this series deal with the practical implementation of logic integrated micro circuits which perform the logic functions dealt with here.

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.

E.T.S. 18/0698

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 to electronic circuits using two defined voltage levels for the binary variable.

2.3 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
- TRUE
- SIGNIFICANT
- CONTACT CLOSED
- LIGHT ON

FIG. 1. LIGHT ON

:: LOGIC 0 to represent
- FALSE
- INSIGNIFICANT
- CONTACT OPEN
- LIGHT OFF

FIG. 2. LIGHT OFF

2.4 In an electronic logic circuit, the designations 1 and 0 are applied to different voltage levels. When a manual switch is used to provide an input to a logic circuit, the two levels can be obtained as shown in Fig. 3.

:: LOGIC 1 or + VCC appears at the output when the switch is OPEN (Fig. 3a).

:: LOGIC 0 or 0 VOLTS appears at the output when the switch is CLOSED (Fig. 3b).

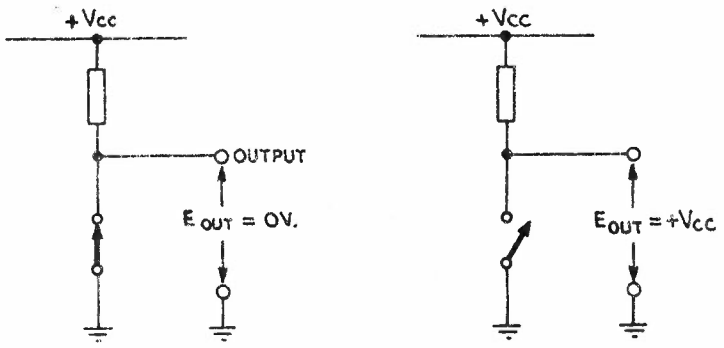


FIG. 3. DERIVATION OF VOLTAGE LEVELS

It is important to note that in the particular situation of Fig. 3, the significance of the switch contact conditions have reversed compared with that of para 2.3, thus

in Fig. 1, CONTACT CLOSED is LOGIC 1 and

in Fig. 3a, CONTACT CLOSED is LOGIC 0.

ALSO

in Fig. 2, CONTACT OPEN is LOGIC O and

in Fig. 3b, CONTACT OPEN is LOGIC 1.

This illustrates that 1 and 0, TRUE and FALSE are designations only, and they must be DEFINED at the outset before any study can be made of an actual circuit. Once the definition has been made it is usual for it to be maintained throughout a system.

2.5 POSITIVE AND NEGATIVE LOGIC. In Fig. 3 the definition given to logic 1 and logic 0 is called POSITIVE LOGIC.

:: LOGIC 1 - the MORE POSITIVE VOLTAGE.
:: LOGIC 0 - the LESS POSITIVE VOLTAGE.

It is, however, just as possible to define logic 1 and logic 0 using negative polarities and this is called NEGATIVE LOGIC.

:: LOGIC 1 - the MORE NEGATIVE VOLTAGE.
:: LOGIC 0 - the LESS NEGATIVE VOLTAGE.

Pure logic diagrams, such as the diagrams used in this paper, represent thought processes and do not show whether the positive logic or negative logic convention is being used. It is not necessary to know which logic convention is being used until a study is made of the schematic circuit of an actual system. This paper considers logic principles only and is not concerned with hardware implementation.

2.6 HIGH and LOW. A method which avoids the need to refer to either logic convention is to designate:

:: HIGH (H) as the more positive voltage.
:: LOW (L) as the less positive voltage.

3. AND GATES

3.1 The AND Function. An analogy of the AND function of an electronic logic gate can be illustrated by using manual switches as shown in Fig. 4.

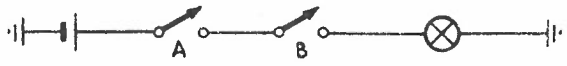


FIG. 4. SWITCHES CONNECTED TO PERFORM AND FUNCTION

If we decide that a SIGNIFICANT state occurs when the LIGHT is ON, this happens when we operate switch A AND switch B. By making the following designations:

:: TRUE (T) occurs when — CONTACT CLOSED
                         — LIGHT ON

:: FALSE (F) occurs when — CONTACT OPEN
                         — LIGHT OFF

Each switch can be in one of two positions and there are four different combinations of switch positions. A table of possibilities, or truth table shows all combinations of the switches and their resultant effect on the light. Table 6 shows the possibilities using the designations given to TRUE and FALSE above.

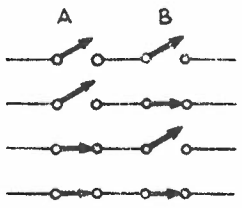


FIG. 5. CIRCUIT POSSIBILITIES

Switch A	Switch B	Light
F	F	F
F	T	F
T	F	F
T	T	T

TABLE 1. TABLE OF POSSIBILITIES FOR AND FUNCTION

There are three binary variables:

:: A and B, which are independent variables, under control of the person operating the switches.
:: The light, which is a dependent variable, under control of the switch positions.

We can say that the dependent variable the light, is TRUE, when and only when,

A AND B are TRUE

3.2 AND GATE. 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.

3.3 Fig. 6 shows the symbol used to represent a two input AND gate in logic diagrams in this publication. 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

3.4 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 at the output. The table uses the designations 1 and 0 instead of T and F and instead of being called a table of possibilities is called a truth table.

Inputs A	B	Output 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.

3.5 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.

3.6 From the rule and from the truth table, it can be seen that a significant change occurs in the output of the AND gate when all inputs become logic 1. On the other hand, if any input is logic 0, changes on other inputs have no effect on the output.

It can be said that an AND gate is:

:: ENABLED when all inputs are logic 1.
:: INHIBITED when any input is logic 0.

4. OR GATES

4.1 THE OR FUNCTION. 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.

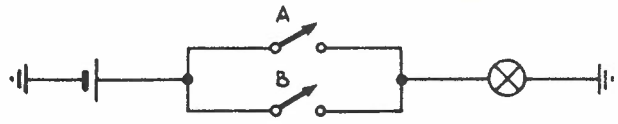


FIG. 7. SWITCHES CONNECTED TO PERFORM OR FUNCTION

If, as in para 3.1, we decide that a TRUE state occurs when the light is on, truth table 3 results from the four possible switch positions.

Switch A	Switch B	Light
F	F	F
F	T	T
T	F	T
T	T	T

TABLE 3. TRUTH TABLE FOR OR FUNCTION

We can say that it is TRUE that the light is on when:

A OR B is TRUE

4.2 OR GATE. 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.

4.3 The circuit symbol for an OR gate is shown in Fig. 8 and the truth table for all possible combinations of input levels is shown in Table 4.

FIG. 8. ELECTRONIC OR GATE SYMBOLS

Input A	Input B	Output C
0	0	0
0	1	1
1	0	1
1	1	1

TABLE 4. TRUTH TABLE FOR OR GATES

4.4 OR GATE RULE. The output of an OR gate is

:: Logic 1 when ANY input is logic 1.
:: Logic 0 when ALL inputs are logic 0.

4.5 A significant change occurs in the output of an OR gate when any input becomes logic 1.

The OR gate is:

:: ENABLED when any input is logic 1.
:: INHIBITED when all inputs are logic 0.

5. THE NOT FUNCTION

5.1 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 to 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 $\overline{A}$ or A', which is read as NOT A or A NOT. A and $\overline{A}$ are referred to as the complements of each other.

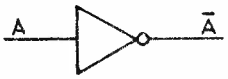


FIG. 9. INVERTER SYMBOL

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

TABLE 5. TRUTH TABLE FOR INVERTER

6. LOGIC CIRCUITS

6.1 Electronic logic circuits use combinations of AND, OR and NOT elements to achieve the decision making function required. When reading logic circuits it is necessary to be able to recognise various techniques used when gates are used in practice.

6.2 AND GATE INHIBIT. One input of an AND gate can be used to inhibit the operation of the gate by placing a logic 0 on any input. Changes on other inputs can have no effect on the ouput which stays at logic 0.

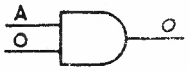


FIG. 10. INHIBITED AND GATE

6.3 AND GATE BUFFER. A permanent logic 1 connected to one input of a two input AND gate ensures that the output is the same as the variable input A. The same result is achieved when both inputs are commoned together. In this application the AND gate serves as a buffer or isolating device between one circuit and another without altering the logic variable.

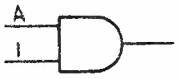


FIG. 11. AND GATE BUFFER

6.4 OR GATE BUFFER. Operation is similar to the AND gate buffer except that logic 0 is placed on the unused input.

FIG. 12. OR GATE BUFFER

6.5 ORDER OF GATE INPUTS. It makes no difference to the operation of the gate whether the inputs are connected in any particular order or sequence. As shown in Fig. 13, inputs labelled A, B, C could also be connected in the order C, A, B. They could also be connected in any other sequence.

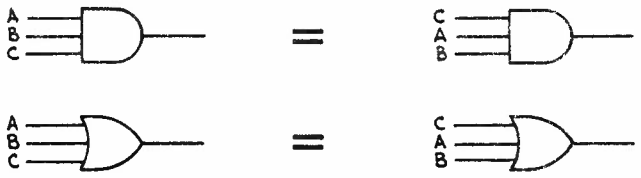


FIG. 13. ORDER OF GATE INPUTS

6.6 INCREASING THE NUMBER OF GATE INPUTS. A theoretical logic gate can have as many inputs as required for the function. Practical considerations limit the number and in integrated microcircuits it is usually eight. If a gate function requires more than the number of inputs available from a single gate, it is possible to connect gates together as shown in Fig. 14.

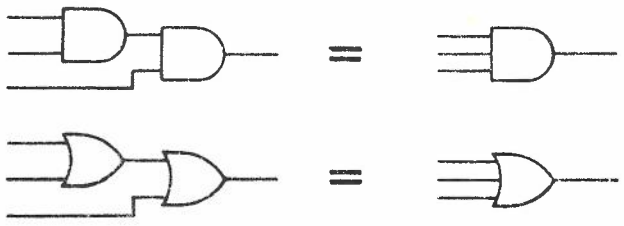


FIG. 14. INCREASING THE NUMBER OF GATE INPUTS

6.7 COMBINING GATE OUTPUTS. Most logic gate circuits suffer damage when their outputs are connected together. Fig. 15 shows how the outputs of two AND gates are connected when it is necessary that the final output is logic 1 when the output from either AND gate is logic 1.

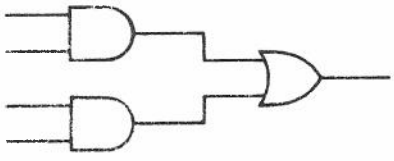


FIG. 15. COMBINING GATE OUTPUTS

6.8 WIRED AND. To avoid the necessity of using another gate to combine gate outputs, it is possible to use special gates in which the outputs can be directly connected. When this is done the effect on the operation is as if another AND gate was connected. To show that the AND function is performed, an AND symbol is placed over the junction of the outputs on the circuit. This configuration is called either WIRED AND or DOT AND. A resistor, external to the gate circuits, is usually required as shown in Fig. 16.

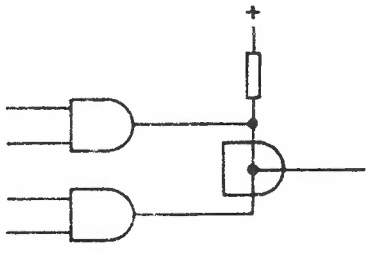


FIG. 16. WIRED AND (DOT AND) CONFIGURATION

7. TIMING DIAGRAMS

7.1 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. 17 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 (positive logic).

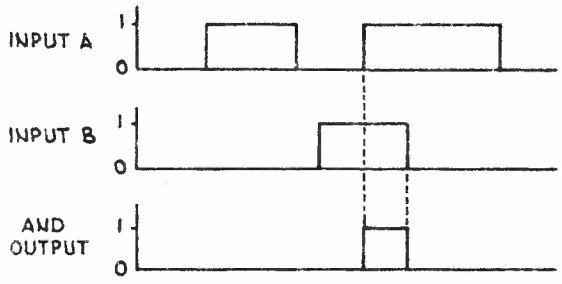


FIG. 17. AND GATE TIMING DIAGRAM

The graphs in Fig. 17 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. 18. In this case, the output is significant when either input A OR input B is significant.

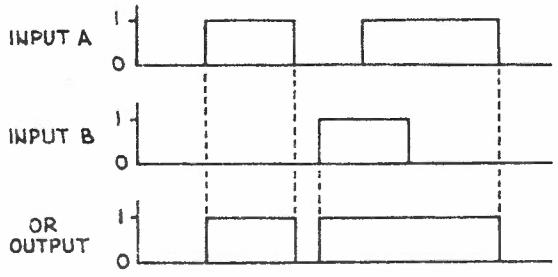


FIG. 18. OR GATE TIMING DIAGRAM

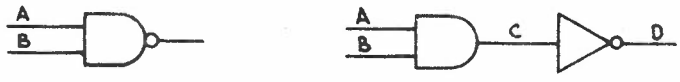
7.2 Electronic logic gates are considered to be static devices, in that the output is under direct control of the inputs. The time taken for a signal to propagate from input to output in electronic circuits is extremely short and in the diagram examples of Fig. 17 and 18, propagation time is not shown.

8. NAND AND NOR GATES

8.1 GENERAL. Electronic logic gates use active devices such as transistors to ensure that the voltage levels of the two logic states in the system are maintained when the output of a gate is used to drive the inputs of a number of other gates. The use of an amplifying device results in logic inversion. An electronic logic network which performs the AND function, together with a transistor amplifier which causes inversion, gives a gate in which the overall result is NOT AND or NAND. When the OR function is combined with inversion, the result is a NOT OR or NOR gate.

Modern NAND or NOR gates are built in the form of integrated circuits and are used in preference, for a number of reasons, to integrated circuit AND or OR gates. Both NAND and NOR gates can be arranged to perform the AND, OR or NOT functions and although the minimum number of elements is not necessarily achieved in a system build exclusively of NAND or NOR gates, economies result firstly in manufacture and secondly in fault repair through the use of a universal gate.

8.2 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. 19a is a symbol used to represent NAND gates in logic diagrams. A state indicator is added to an AND gate symbol to indicate an inversion following the AND function. Fig. 19(b) is an equivalent circuit of a NAND gate using basic AND and NOT elements.



(a) SYMBOL (b) LOGIC EQUIVALENT CIRCUIT

FIG. 19. ELECTRONIC NAND GATE

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

TABLE 6. NAND GATE TRUTH TABLE

8.3 NAND GATE RULE. Irrespective of the number of inputs, the output of a NAND gate is:

:: Logic 0 when ALL inputs are logic 1.
:: Logic 1 when ANY input is logic 0.

Note that when the NAND gate is:

:: ENABLED the output is logic 0.
:: INHIBITED the output is logic 1.

Logic 1 on all inputs gives logic 0 output

8.4 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 of the effective input, and is the inverse of it. In Fig. 20 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 7. Since the output in each case is the inverse of the input, the output is A NOT, ($\overline{A}$).

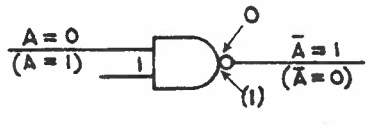


FIG. 20. NAND GATE USED AS INVERTER

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

TABLE 7. TRUTH TABLE FOR FIG. 20

8.5 NAND GATE COMBINATION TO PERFORM AND FUNCTION. In Fig. 21 two NAND gates are combined to perform 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 C produced in the first gate is inverted twice and appears in the output. Tables 8 and 9 are the truth tables for Fig. 21 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.

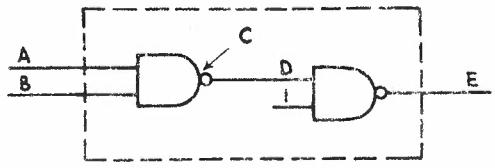


FIG. 21. NAND GATE COMBINATION WHICH PERFORMS THE AND FUNCTION

Inputs		Intermediate Points		Output
A	B	C	D	E
0	0	0	1	0
0	1	0	1	0
1	0	0	1	0
1	1	1	0	1

TABLE 8. TRUTH TABLE FOR FIG. 21

=

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

TABLE 9. AND GATE TRUTH TABLE

8.6 NAND GATE COMBINATION TO PERFORM THE OR FUNCTION. Fig. 22 shows how three NAND gates are combined to perform the OR function. Signal A is inverted to $\bar{A}$ signal B is inverted to $\bar{B}$, both inverted signals are "ANDed" together, and the output of the AND function is inverted. Note that the first two NAND gates are used as inverters only.

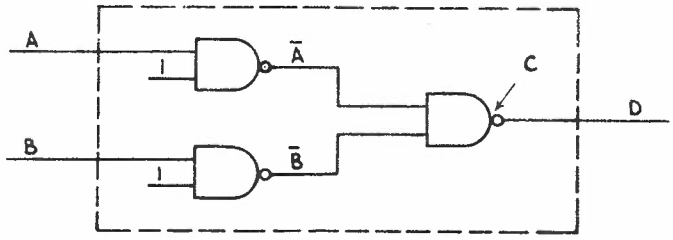


FIG. 22. NAND GATE COMBINATION WHICH PERFORMS THE OR FUNCTION

Tables 10 and 11 are the truth tables for Fig. 22 and an OR gate, respectively. Since the output results are the same in each case, the circuits are equivalent. Both perform the OR function.

Inputs		Intermediate Points			Output
A	B	$\bar{A}$	$\bar{B}$	C	D
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

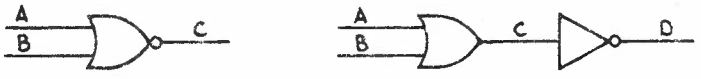
TABLE 10. TRUTH TABLE FOR FIG. 22

=

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

TABLE 11. OR GATE TRUTH TABLE

8.7 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. 23(a) shows a symbol used to represent NOR gates. Note that a state indicator is added to the OR gate symbol to indicate an inversion. Fig. 23(b) is an equivalent circuit of a NOR gate using basic OR and NOT logic elements.



(a) SYMBOL (b) EQUIVALENT CIRCUIT

FIG. 23. ELECTRONIC NOR GATE

Although the signal C is not usually available as an output, it often helps to include this intermediate step when considering NOR gates. Table 12 shows the function performed by a NOR gate.

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

TABLE 12. NOR GATE TRUTH TABLE

8.8 NOR GATE RULE. Irrespective of the number of inputs, the output of a NOR gate is:

:: Logic 0 when ANY input is logic 1.
:: Logic 1 when ALL inputs are logic 0.

Note that when the NOR gate is:

:: ENABLED the output is logic 0.
:: INHIBITED the output is logic 1.

Logic 1 on any input gives logic 0 output.

8.9 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. 24 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.

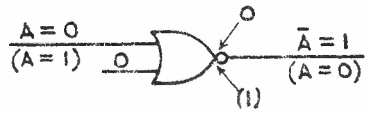


FIG. 24 NOR GATE USED AS INVERTER

8.10 NOR GATE COMBINATION TO PERFORM THE OR FUNCTION. Fig. 25 shows how two NOR Gates are combined to perform the OR function.

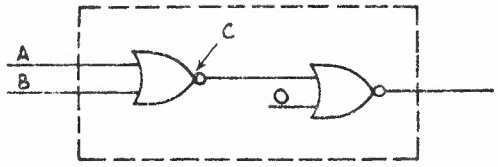


FIG. 25. NOR GATE COMBINATION WHICH PERFORMS OR FUNCTION

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

8.11 NOR GATE COMBINATION TO PERFORM THE AND FUNCTION. Fig. 26 shows how three NOR gates are combined to perform the AND function. The first two gates serve as inverters only, to produce the signals $\bar{A}$ and $\bar{B}$. These two signals are "ORed" together and the result is inverted.

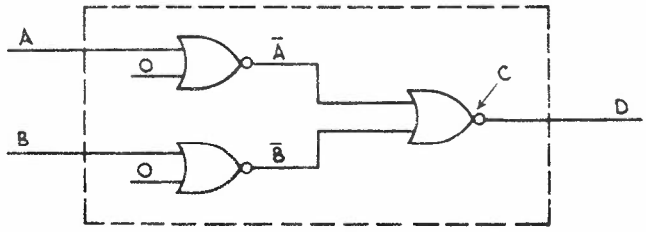


FIG. 26. NOR GATE COMBINATION WHICH PERFORMS AND FUNCTION

Tables 13 and 14 are the truth tables of Fig. 26 and an AND gate respectively. Since the input combinations and the output conditions in both tables are exactly the same, Fig. 26 must perform the AND function.

Inputs					Output
A	B	$\bar{A}$	$\bar{B}$	C	D
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 13. TRUTH TABLE FOR FIG. 26

=

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

TABLE 14. AND GATE TRUTH TABLE

9. EXCLUSIVE-OR AND COMPARATOR

9.1 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. The gate combination in Fig. 27 performs the Exclusive-OR function and Table 15 shows the output condition for each combination of inputs.

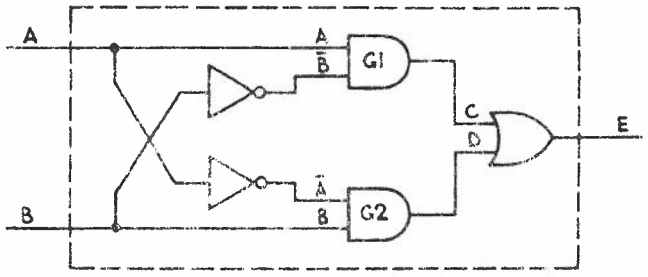


FIG. 27. EXCLUSIVE-OR GATE COMBINATION

Inputs						Output
A	B	$\bar{A}$	$\bar{B}$	C	D	E
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 15. EXCLUSIVE-OR TRUTH TABLE

Fig. 28 is the distinctive symbol for the Exclusive-Or gate.

FIG. 28. EXCLUSIVE-OR SYMBOL

9.2 In summary, the EXCLUSIVE-OR gate gives:

:: Logic 1 output when one, and only one, input is at logic 1.

9.3 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. Fig. 29 is a simple comparator circuit and Table 16 is the truth table for Fig. 29. 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 $\bar{A}$ and $\bar{B}$ are both logic 1, so the bottom AND gate provides a logic 1 output. The comparator circuit is said to test for equivalence.

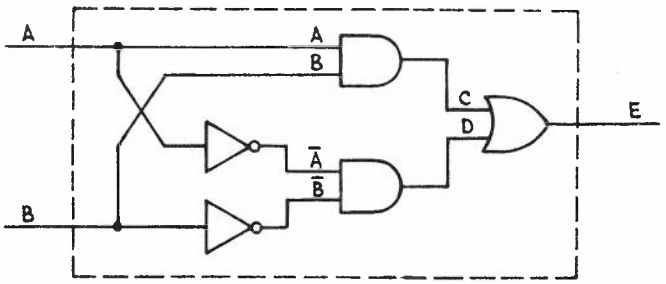


FIG. 29. SIMPLE COMPARATOR

Inputs						Output
A	B	$\bar{A}$	$\bar{B}$	C	D	E
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 16. TRUTH TABLE FOR FIG. 29

There is no distinctive symbol for a comparator. The Standards Association does however recommend a rectangular symbol which is shown in Section 12 of this publication.

9.4 Comparators may have more than two inputs and this leads to the general statement that a comparator gives:

:: Logic 1 output when ALL inputs are logic 1.
:: Logic 1 output when ALL inputs are logic 0.

Or, in summary, the comparator indicates with a logic 1 output that all inputs are at the same logic level.

9.5 The comparator function is also performed when the Exclusive-OR circuit is followed by an inverter, as shown in Fig. 30. Table 17 is the truth table for Fig. 30.

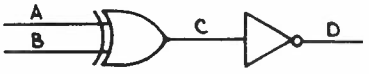


FIG. 30. COMPARATOR USING EXCLUSIVE-OR AND INVERTER

Inputs		O/P of Exclusive-OR	Output
A	B	C	D
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

TABLE 17. TRUTH TABLE FOR FIG. 3

10. READING LOGIC SYMBOLS

10.1 The functions of AND, OR and NOT are the functions that are basic to all logic gates, and all other gate functions are derived from these three basic elements. The symbols used so far comply with the Standards Association of Australia "distinctive symbols", and the "rectangular symbols" recommended by the Association are given in Section 12 of this publication.

10.2 SYMBOLS. Both positive and negative logic conventions assign a particular voltage level to the logic 1 state and another to the logic 0 state. When this is referred to the symbol, the 1-state is implied with the single lines connecting directly to the symbol on the inputs and outputs.

The AND symbol is read as follows:

:: Input lines connecting directly to the symbol indicate the 1-state.

:: The distinctive shape of the symbol indicates that the AND function is performed.

:: The single line directly from the output indicates that the 1-state is the result of the given inputs and the gate function.

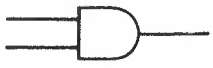


FIG. 31. AND SYMBOL

In summary, using the AND gate rule:

:: Logic 1 on all inputs results in a logic 1 output.

The OR symbol is read:

:: 1-state on inputs.

:: OR function from the distinctive shape.

:: 1-state on output from inputs and gate function.

FIG. 32. OR SYMBOL

In summary, using the OR gate rule:

:: Logic 1 on any input results in a logic 1 output.

10.3 State Indicator or negation indicator is a small circle used on the input or output connection line to the symbol. When the logic negation indicator is applied the state of the logic variable at an input or output is reversed.

In the example of an inverter:

:: 1-state on the input.

:: State indicator on the output means reversal of the logic 1 choice, i.e. the 0-state.

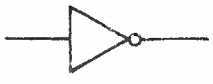


FIG. 33. INVERTER SYMBOL

In summary:

:: Logic 1 on the input results in a logic 0 on the output.

11. ALTERNATE SYMBOLS

11.1 DEFINITION OF LOGIC LEVELS. As stated in para. 2.4, it is very important to define logic levels and their significance. In a given logic system logic 0 and logic 1 are assigned particular voltage levels and it is usual to adhere to the nomenclature throughout the system. The symbols used in previous sections of this paper assume that the logic 1 level is the significant voltage level on inputs to logic elements. That is to say, logic 1 is considered to be the voltage level required to activate the device. In the case of NOT, NAND and NOR functions however, logic 1 input levels result in logic 0 output when the function is activated and on the symbol this is shown by means of a state indicator.

11.2 THE NEED FOR ALTERNATE SYMBOLS. The basis of the use of alternate symbols is to indicate which is the signficant logic state by using the drawing principles outlined in para 10. In summary:

:: The 1 state is significant when lines connect directly to the symbol.

:: The 0 state is significant when a state indicator is used on the symbol.

Logic 1 is not always the significant state in a logic circuit and there is a need to indicate this on circuit diagrams.

11.3 INVERTER ALTERNATE SYMBOL. Fig. 34 shows the alternate symbols for an inverter. Fig. 34(a) shows that when logic 1 is connected to the inverter input, logic 0 is the output. Fig. 34(b) shows that logic 0 on the input causes a logic 1 output.

(a) (b)

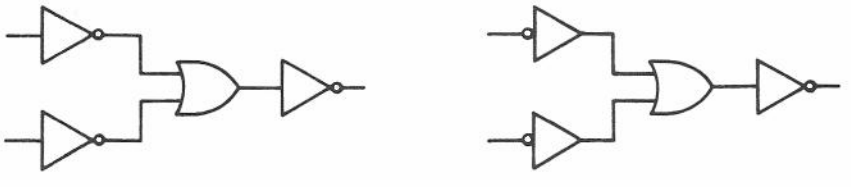
FIG. 34. INVERTER ALTERNATE SYMBOLS

11.4 ALTERNATE SYMBOLS. Fig. 35(a) shows the symbol for an AND gate in which logic 1 is significant and Fig. 35(b) is the symbol for the same gate when logic 0 is significant. Note that the AND distinctive shape has changed to OR.

(a) (b)

FIG. 35. AND ALTERNATE SYMBOLS

At first it does not seem possible that the same device could perform both AND and OR functions. Symbolically, proof can be obtained by referring to para. 8.11 in which a NOR gate combination performs the AND function. Fig. 36(a) is the same circuit as Fig. 26 with the exception that the input NOR gates which act as inverters are drawn simply as inverters and the NOR gate is drawn as its logic equivalent circuit of an OR gate and an inverter. Fig. 36(b) is again the same circuit but in which the input inverters are shown with their alternate symbol. Finally, Fig. 36(b) could be drawn the same as Fig. 35(b).



(a) (b)

FIG. 36. THE AND FUNCTION USING AN OR GATE

Another name for a state indicator is negation indicator. However when a negation indicator is used on a symbol it does not necessarily mean that a negator or inverter device is present. For example Fig. 35(b) and Fig. 36(b) perform the same function but in Fig. 35(b) there are no inverters.

Examination of the AND gate truth table also shows that the OR function is performed when logic 0 is considered to be the important input logic variable. Thus, as shown in Table 18, the output is logic 0 when ANY input is logic 0.

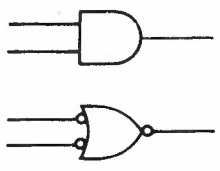


FIG. 37. AND GATE

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

TABLE 18.

Fig. 38 shows alternate symbols for the OR, NAND and NOR functions.

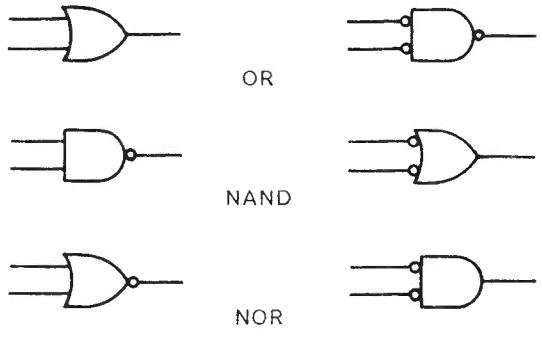


FIG. 38. ALTERNATE SYMBOLS

11.5 READING LOGIC SYMBOLS. When deducing the function of logic circuits, it is not necessary to know the actual hardware used if the logic symbol is correctly interpreted. Fig. 39 is a redrawn version of Fig. 27, a functional logic diagram of the EXCLUSIVE-OR function. Reading Fig. 39 the function of the circuit is that the output is logic 1 when A is logic 1 AND B is logic 0 OR when A is logic 0 AND B is logic 1.

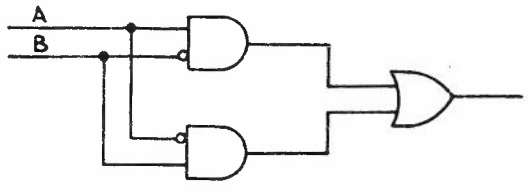


FIG. 39. EXCLUSIVE OR FUNCTION

12. AUSTRALIAN STANDARDS ASSOCIATION SYMBOLS

12.1 STANDARDS. The symbols used in this paper are in accordance with the Standards Association of Australia revision of AS1102, Part 9 - 1971. The Association has adopted a dual symbol system in which the distinctive symbols for gates used in AS1102, Part 9 - 1971 have been retained and gives as an alternative the symbols recommended by the International Electrotechnical Commission 117-15.

12.2 I.E.C. SYMBOLS. The IEC recommends that the general symbol for a gate consists of a rectangle, the length-width ratio of which is arbitrary and may vary according to the internal information required or the number of input or output lines that must be accommodated.

Fig. 40 shows the distinctive symbols used in earlier Sections of this publication together with their equivalent rectangular symbol. More comprehensive information on symbols may be obtained from the publication Revision of AS1102, Part 9 - 1971.

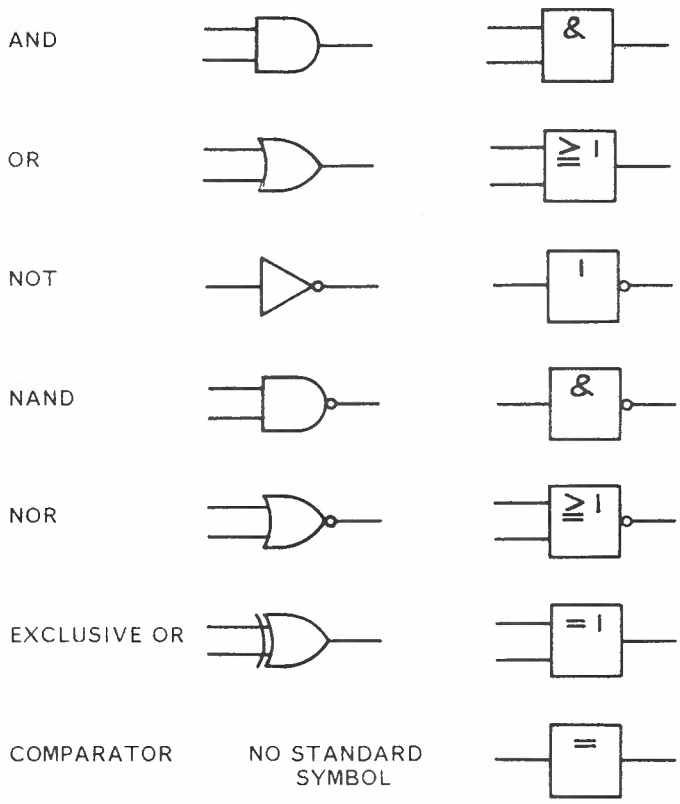
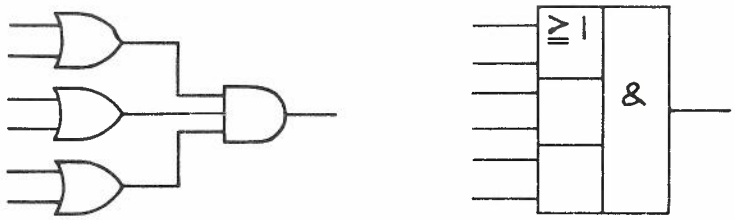


FIG. 40 SYMBOLS

The qualifying symbol inside the rectangle is a symbol generally indicating the number of inputs which must necessarily take on the defined logic 1 state to cause the output to take on its defined 1 state, provided the output is not negated. In the case of the OR gate the qualifying symbol ≧1 may be replaced by a 1 when there is no ambiguity about the function of the gate.

12.3 COMBINED SYMBOLS. The use of the rectangular shape allows a combination of symbols to be drawn in such a way as to save space. Fig. 41a shows three OR functions independent of each other but connected to an AND function. Note that in Fig. 41b there is no connection implied between the OR gates and that the qualifier (≧1) needs mention in only the first OR symbol since the rest are the same.



(a) (b)

FIG. 41. COMBINATION OF ELEMENTS

13. BOOLEAN EXPRESSIONS

13.1 DEVELOPMENT. In 1847, an English mathematician, George Boole, developed the basic laws and rules for a mathematics which could be applied to problems of deductive logic. In the late 1930s it was seen that Boolean mathematics could be adapted to the analysis of multi-contact networks in telephone exchanges. With the development of electronic logic in control circuits and computers, Boolean expression manipulation has been found to be very useful as a design aid.

13.2 VARIABLES. Boolean mathematics involves two valued variables and to philosophers in deductive logic they are usually referred to as true and false. As described in Section 2, the variables are given the values 0 and 1 in electronic logic circuits. To distinguish between different variables the letters of the alphabet are used, A, B, C and so on.

13.3 RELATIONSHIP BETWEEN VARIABLES. Standard arithmetical symbols are used for the relationships between variables. The arithmetical symbols used in Boolean expressions have the following meanings:

:: The symbol (.) represents the word AND

:: The symbol (+) represents the word OR

:: The symbols ($\bar{}$) or (') represent the word NOT The (') is termed a "Prime".

13.4 Word Statements are used when describing the operation of logic circuits and it is from the word statement that a Boolean expression is derived. For example the word statement for a two input AND gate would be:

The output C is logic 1 WHEN input A is logic 1 AND input B is logic 1.

13.5 Boolean expression for an AND gate would be:

C = A.B

:: A and B are independent variables.
:: C is the dependent variable.
:: The (.) replaces the AND of the word statement.
:: The (=) replaces the WHEN of the word statement.

13.6 BOOLEAN EXPRESSION OF GATE OPERATION. The following are Boolean expressions of logic gates in earlier sections of this paper.

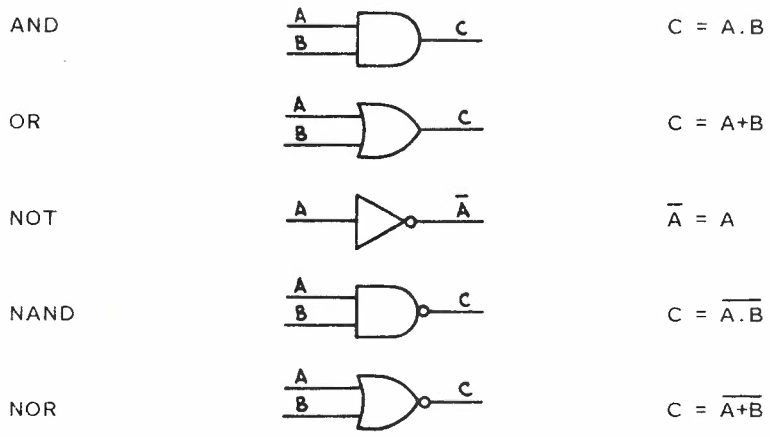


FIG. 42. BOOLEAN EXPRESSIONS

13.7 COMBINED LOGIC FUNCTIONS. It is possible to derive a Boolean expression for a logic circuit which has a mixture of gates. Alternatively a logic circuit can be drawn from a Boolean expression.

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. 43 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.

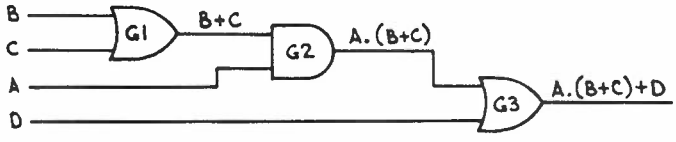


FIG. 43. LOGIC DIAGRAM REPRESENTING A.(B + C) + D.

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. 44.

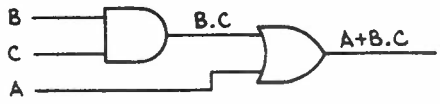


FIG. 44. LOGIC DIAGRAM REPRESENTING A + B.C.

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

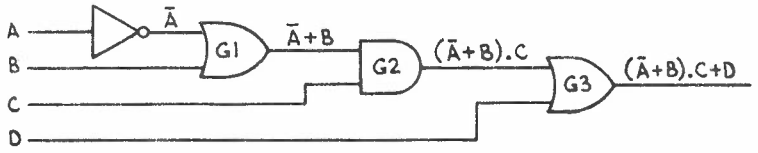
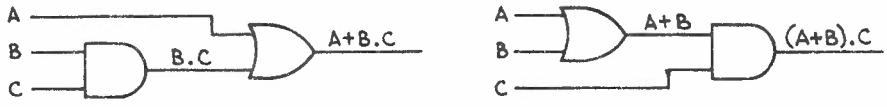


FIG. 45. $(\bar{A} + B).C + D$

13.8 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. 46(a) and 46(b) respectively.



(a) Logic Circuit of A + B.C (b) Logic Circuit of (A + B).C

FIG. 46. THE IMPORTANCE OF BRACKETS

13.9 BOOLEAN ALGEBRA MANIPULATION. When a logic circuit is being designed, it is usual to obtain a Boolean expression which describes the circuit requirements. The final logic circuit is derived when the rules of Boolean algebra are used to manipulate the expression for the minimum number of gates in the type of gate selected for the circuit (usually NAND or NOR). These techniques are beyond the scope of this publication and will not be discussed.

14. BINARY NOTATION

14.1 The use of the symbols 1 and 0 to represent the two states of a binary variable lead to the use of binary notation. This is a number system with a base of two.

14.2 In the use of the scale of ten (the decimal scale) counting is from 0 to 9. Each intermediate value has a specific symbol, 1, 2, 3, 4, 5, 6, 7, 8. On reaching ten, a one is put in the tens column thus, 10, and start again, 10, 11 ..... 98, 99. On reaching a hundred a one is put in the hundreds column thus, 100, and carry on, 101, 102.

In this system the number 5278 is really:

= 5 thousands, + 2 hundreds, + 7 tens, + 8 units
= $(5 \times 1000) + (2 \times 100) + (7 \times 10) + (8 \times 1)$
= $(5 \times 10^3) + (2 \times 10^2) + (7 \times 10^1) + (8 \times 10^0)$

14.3 The binary scale is obtained in exactly the same way, with the small difference that a number is expressed as powers of two rather than of ten. Counting proceeds from 0 to 1, in the units column or 1 zone. When 2 is reached, it is represented by 10, a 0 in 1 zone and a 1 in 2 zone. Each zone is designated by the decimal number equivalent to which power of two the zone represents.

Table 19 shows how decimal numbers can be expressed as powers of 2, how these powers can be represented by a 1 in the appropriate zone, and how the decimal number is written in its equivalent binary form.

Decimal Notation	Powers of Two	Numbers of Various Powers					Binary Notation
		2^4 16 zone	2^3 8 zone	2^2 4 zone	2^1 2 zone	2^o 1 zone	
0							0
1	2^0					1	1
2	2^1				1	0	10
3	$2^1 + 2^0$				1	1	11
4	2^2			1	0	0	100
5	$2^2 + 2^0$			1	0	1	101
6	$2^2 + 2^1$			1	1	0	110
7	$2^2 + 2^1 + 2^0$			1	1	1	111
8	2^3		1	0	0	0	1000
9	$2^2 + 2^0$		1	0	0	1	1001
10	$2^3 + 2^1$		1	0	1	0	1010
11	$2^3 + 2^1 + 2^0$		1	0	1	1	1011
12	$2^3 + 2^2$		1	1	0	0	1100
13	$2^3 + 2^2 + 2^0$		1	1	0	1	1101
14	$2^3 + 2^2 + 2^1$		1	1	1	0	1110
15	$2^3 + 2^2 + 2^1 + 2^0$		1	1	1	1	1111
16	2^4	1	0	0	0	0	10000

TABLE 19

14.4 Each digit in a binary number has an equivalent decimal number according to the "zone" it occupies. Each zone is a power of 2. The first place to the left of the binary point is called the "1 zone", as a digit in this place represents 1 in decimal notation (2^o).

The second place to the left of the binary point is called the "2 zone", as a digit in this place represents 2 in decimal notation (2^1).

Similarly the fifth place to the left of the binary point is called the "16 zone", as a digit in this place represents 16 in decimal notation (2^4).

To write any decimal number as a binary number:

Find the zone numbers whose sum is the number required.
Allot a 1 for each zone number used, and
an 0 for each unused zone.

Example 1. Express the number 43 in binary form:

The number 43 is the sum 32 + 8 + 2 + 1

16 zone and 4 zone are not used

Write the binary form from right to left as each zone is considered:

43 in binary form is 101011.

Consider the number "43" expressed in powers of two - it becomes:

$$32 + 0 + 8 + 0 + 2 + 1$$

$$= (\underline{1} \times 2^5) + (\underline{0} \times 2^4) + (\underline{1} \times 2^3) + (\underline{0} \times 2^2) + (\underline{1} \times 2^1) + (\underline{1} \times 2^0)$$

In the binary scale, 43 = 101011.

The number 101011 is not read as "one hundred and one thousand and eleven". As no separate nomenclature has been allotted in this notation, we can say in words only that 101011 is "one nought one nought one one".

14.5 There is another simple method of converting numbers in decimal notation to binary notation. Merely divide by 2 repeatedly until the quotient is zero, writing the remainders as they occur.

2	43
2	21 + 1
2	10 + 1
2	5 + 0
2	2 + 1
2	1 + 0
	0 + 1

The answer is read as the remainders in order from the last remainder found to the first. 101011 represents 43.

The number of digits in a number expressed in binary notation is often about three times as many as that in the decimal scale. This does not imply any difficulty in notation, as the method of use is adapted to the method of notation.

14.6 To convert a number in binary notation to decimal notation, merely allot the zone value to each binary 1 in the number to be converted. Add the zone values to find the decimal number.

Example 2. Convert the binary number 101110 to a decimal number.

$101110 = 32 + 0 + 8 + 4 + 2 + 0$

i.e. $(\underline{1} \times 2^5) + (\underline{0} \times 2^4) + (\underline{1} \times 2^3) + (\underline{1} \times 2^2) + (\underline{1} \times 2^1) + (\underline{0} \times 2^0)$

$= 46$

15. CONSTRUCTION OF TRUTH TABLES

15.1 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.

15.2 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 combintions is equal to 2^n, where n is the number of inputs. Table 20, which is the truth table for the circuit in Fig. 47 is an example.

(a) Head the columns inputs, intermediate points, and output, as shown in Table 20. (Intermediate points are not always shown in truth tables, but assist in determining the output condition).

(b) Designate a column for each input, the intermediate points, and the output.

(c) Determine the number of combinations. In this case the number of combinations is $2^3 = 8$ because there are three inputs in Fig. 47. The number of combinations determines the number of horizontal lines in the truth table.

(d) Commence all input columns with zeros.

(e) Complete the last input column (in this case C), by changing the condition of the binary variable on each line.

(f) Complete the second last input column (in this case B) by changing the condition of the binary variable after every two combinations.

(g) Complete the next input column (in this case A) by changing the condition of the binary variable after every four combinations.

(h) Record the conditions of the intermediate points. In this 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.

(i) 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.

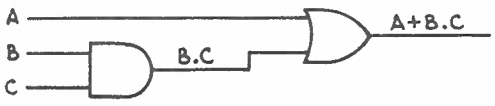
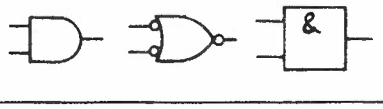
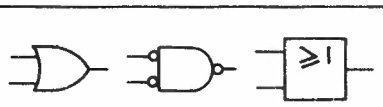
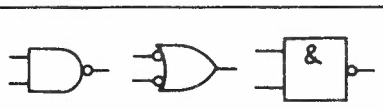
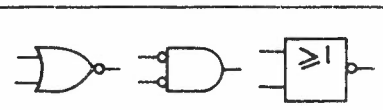
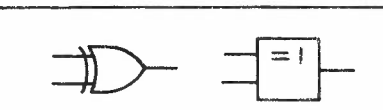
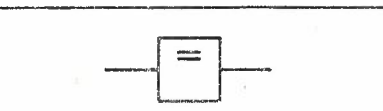
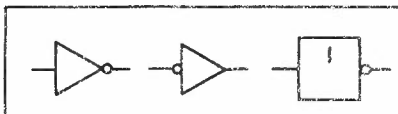


FIG. 47. LOGIC CIRCUIT

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 20. TRUTH TABLE

16. SUMMARY OF GATES

SYMBOLS	BOOLEAN EXPRESSION	TRUTH TABLE		
		INPUT A	INPUT B	OUTPUT C
AND				
	$C = A.B$	0 0 1 1	0 1 0 1	0 0 0 1
OR				
	$C = A+B$	0 0 1 1	0 1 0 1	0 1 1 1
NAND				
	$C = \overline{A.B}$	0 0 1 1	0 1 0 1	1 1 1 0
NOR				
	$C = \overline{A+B}$	0 0 1 1	0 1 0 1	1 0 0 0
EXCLUSIVE OR				
	$C = A.\overline{B}+\overline{A}.B$	0 0 1 1	0 1 0 1	0 1 1 0
COMPARATOR				
	$C = A.B+\overline{A}.\overline{B}$	0 0 1 1	0 1 0 1	1 0 0 1
NOT				
	$\overline{A} = A$	INPUT A 0 1		OUTPUT $\overline{A}$ 1 0