15 Digital Logic Circuits

Contents

15.2

Introduction

Logic circuits are at the heart of every digital device, ranging from supercomputers to miniature IoT devices. Desktop computers and mobile phones are in the middle of this range, and are well-known by consumers.

All logic circuits rely on just a handful of basic *logic gates*. These gates perform what is known as Boolean algebra – an algebra put forward by George Boole in 1854 in order to systematise logic.

15.1 Number Systems

The most common numbering system in use today is the decimal system which utilises ten digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The number of digits used in the system is known as its base or radix; the base of the decimal system is ten.

When digits are strung together a number is formed and each digit within the number is associated with the radix raised to a power according to its position in the number.

EXAMPLE 15.1 Base 10 Number

decimal 853.4 = $853.4_{10} = (8\times10^2) + (5\times10^1) + 3\times10^0) + 4\times10^{-1}$

Note: subscript represents the base of the number.

Another common numbering system used is the binary system which utilises two digits 0, 1 and thus has a base of two. Binary digits are often referred to as "bits" (binary digits), an eight-bit number (byte) will have 8 binary digits.

A binary number can be converted to its equivalent decimal number by expanding the number as a polynomial in powers of the base and the adding the terms.

EXAMPLE 15.2 Base 2 (Binary) Number

$$\begin{aligned}1101_2 &= (1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)\\ &=8+4+0+1\\ &=13_{10}\end{aligned}$$

Any base may be converted to base ten by this method.

15.2 Decimal To Any Base Conversion

A decimal number may be converted to its equivalent in another base by the repeated division by the base, the successive remainders giving the required number.

EXAMPLE 15.3 Base Conversion

eg. 20_{10} to binary

$$\begin{array}{r l}
2\,\big)\underline{\;20\;} & \\
2\,\big)\underline{\;10\;} & \text{remainder 0 (LSB)} \\
2\,\big)\underline{\;\;5\;} & \text{remainder 0} \\
2\,\big)\underline{\;\;2\;} & \text{remainder 1} \\
2\,\big)\underline{\;\;1\;} & \text{remainder 0} \\
0\; & \text{remainder 1 (MSB)}
\end{array}$$

thus $20_{10} = 10100_2$

Note: LSB = Least Significant Bit

MSB = Most Significant Bit

Fractions can be converted by repeated multiplication by the base with the successive integers giving the required number.

EXAMPLE 15.4 Base Conversion

0.6_{10} to binary:

$$\begin{aligned}
0.6 \times 2 &= 1 + 0.2 \\
0.2 \times 2 &= 0 + 0.4 \\
0.4 \times 2 &= 0 + 0.8 \\
0.8 \times 2 &= 1 + 0.6 \\
0.6 \times 2 &= 1 + 0.2
\end{aligned}$$

etc. Thus $0.6_{10} = 0.10011001\ \ldots$

Note: When converting fractions there is not always an exact conversion.

15.3 Binary Coded Decimal (BCD)

As can be seen from the previous examples, it can take a large number of bits to represent a large decimal number and conversion of a large number of bits back to decimal will be long and tedious. The conversion can be overcome if the binary number is coded such that each four bits represent one decimal digit.

EXAMPLE 15.5 Base Conversion

$$
\begin{array}{lccc}
534_{10} = & 0101 & 0011 & 0100_{BCD} \\
\quad = & 5 & 3 & 4
\end{array}
$$

Note: the leading zero is required.

15.4 Boolean Quantities and Variables

A Boolean quantity is a quantity that has only two possible states.

EXAMPLE 15.6 Boolean Quantities

The state of a switch	-	ON/OFF
Is it raining?	-	YES/NO
Flipping a coin	-	HEADS/TAILS

A Boolean variable is a variable that has been assigned to a Boolean quantity.

For example, for the state of a switch the variable S = switch ON and NOT S or $\bar{S}$ represents the switch being OFF.

S is the Boolean variable assigned to the state of the switch. If the switch is on, S is true and has the binary value of "1", if the switch is off, $\bar{S}$ is true and S is false. Here S has the binary value of "0" and $\bar{S}$ has the binary value of "1".

15.5 Boolean Functions

15.5.1 The AND Function

Consider the circuit below:

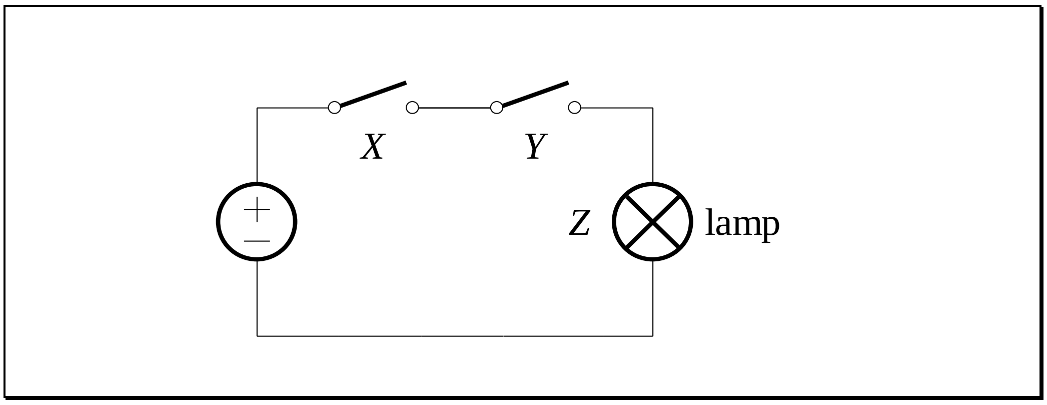


Figure 15.1

There are three Boolean variables *X*, *Y* and *Z*. The two switches have been assigned the variable *X* and *Y* and have the value "1" when the switches are operated. The variable *Z* has been assigned to the lamp and has the value "1" when the lamp is on.

From the circuit it can be seen that the lamp will be on only when both switches, *X* and *Y*, are operated and thus it can be written:

$$\boxed{Z = X \text{ AND } Y \quad \text{or} \quad Z = X.Y} \qquad (15.1)$$

The "." represents the **AND** function.

The function Z = X.Y can be tabulated in a *truth table* as shown in Table 15.1. The truth table tabulates the outputs of a function against all possible variations of input. For *N* input variables there are 2^N possible combinations.

Input		Output
X	*Y*	*Z*
0	0	0
0	1	0
1	0	0
1	1	1

Table 15.1 – AND gate 'truth' table

The symbol for the **AND** gate is shown below:

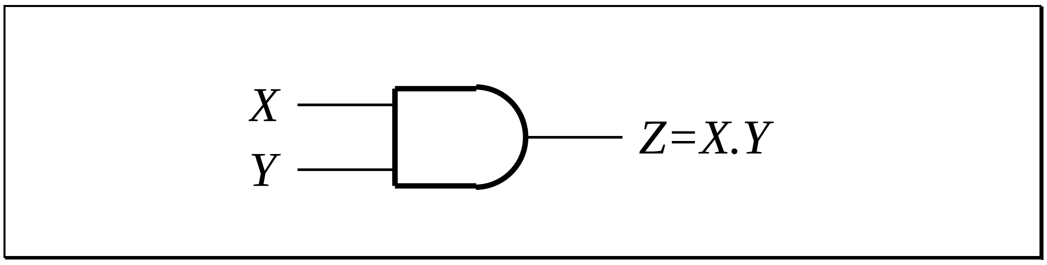


Figure 15.2

In practice the inputs *X* and *Y* are voltage levels

e.g.	logic 1	=	+ 3.3 volts
	logic 0	=	0 volts

and the output will also be a corresponding voltage level.

15.5.2 The OR Function

Consider the circuit below:

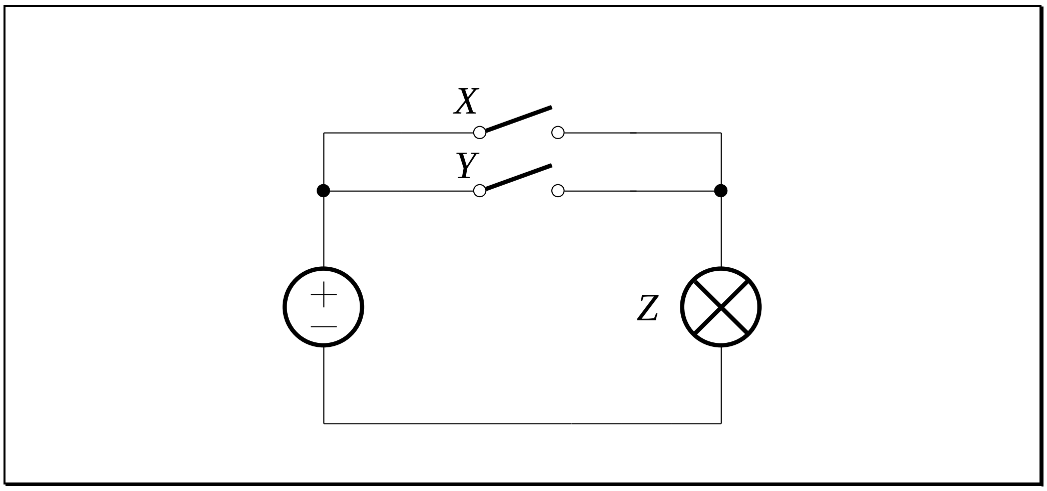


Figure 15.3

From the circuit shown in Figure 15.3, it can be seen that the **lamp** is on when either switch *X* or *Y* is operated and the output, *Z* can be written as:

$$\boxed{Z = X \text{ OR } Y \quad \text{or} \quad Z = X + Y} \tag{15.2}$$

where "+" represents the **OR** function. The truth table is shown below:

Input		Output
X	*Y*	*Z*
0	0	0
0	1	1
1	0	1
1	1	1

Table 15.2 – OR gate 'truth' table

The symbol for the **OR** gate is shown below:

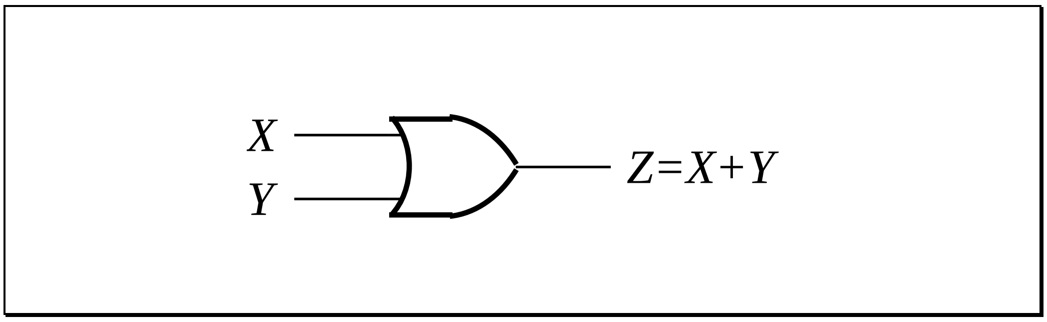


Figure 15.4

15.5.3 The NOT Function

Consider the circuit below:

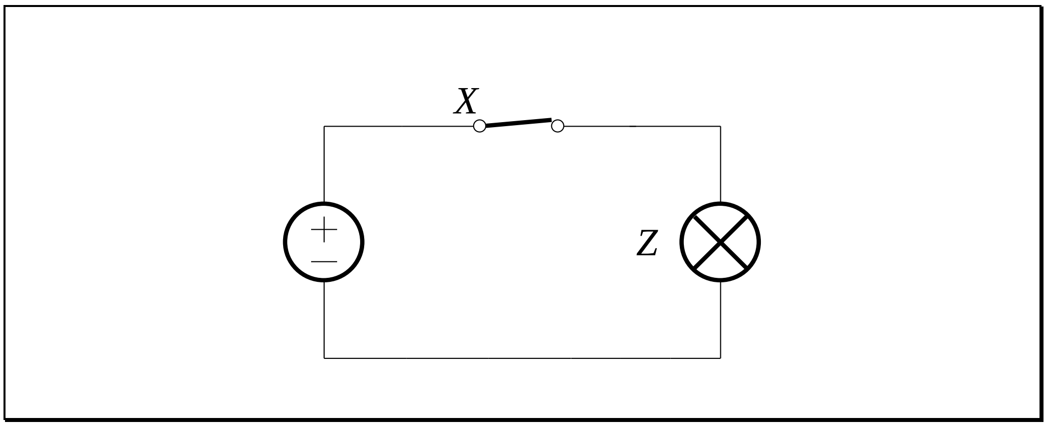


Figure 15.5

From the circuit shown in Figure 15.5, when the switch X is operated (X=1) the contacts are open and thus the lamp goes out and thus the expression can be written as:

$$\boxed{\text{Z = NOT X} \qquad \text{or} \quad Z = \overline{X}} \qquad (15.3)$$

The **NOT** function is more commonly called an inverter. The truth table is shown below:

Input	Output
X	Z
0	1
1	0

Table 15.3 – NOT gate 'truth' table

The symbol for the inverter is shown below:

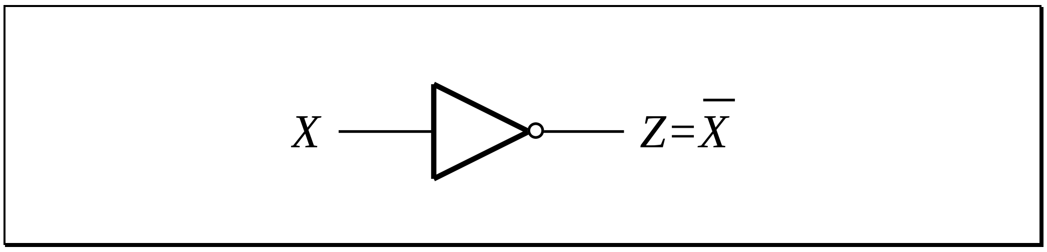


Figure 15.6

Note that it is the logic level that is inverted, not the actual voltage representing the logic level, i.e. If you NOT +3.3 V you get 0 V, not -3.3 V.

15.5.4 The NAND Function

The combination of an AND gate followed by an inverter operates as a gate known as the NAND gate i.e. the NOT AND gate, as shown below:

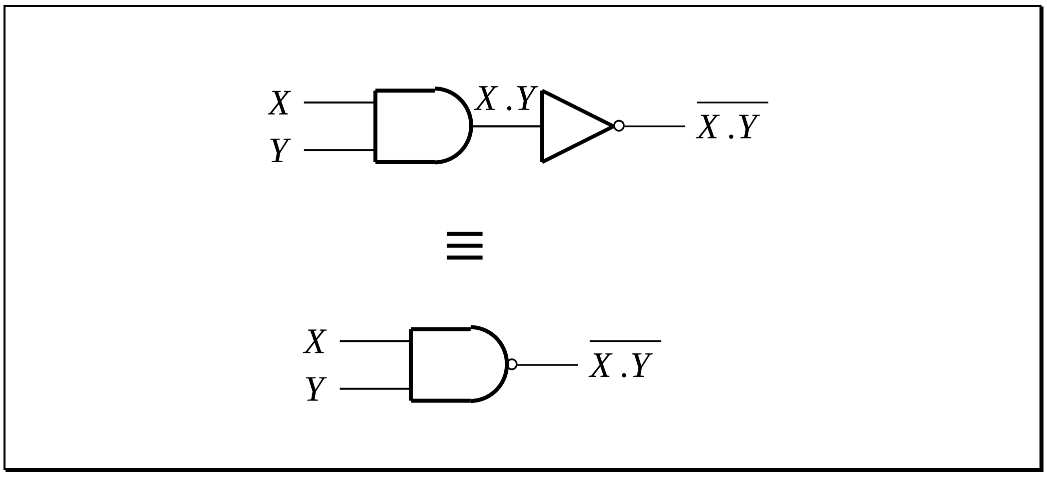


Figure 15.7

The truth table of a NAND gate is shown in Table 15.4.

Input		Output
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Table 15.4 – NAND gate ‘truth’ table

From this table the output is true when $\overline{X}$ or $\overline{Y}$ are true, i.e. either X or Y are false, thus the NAND function can be represented by the circuit shown below:

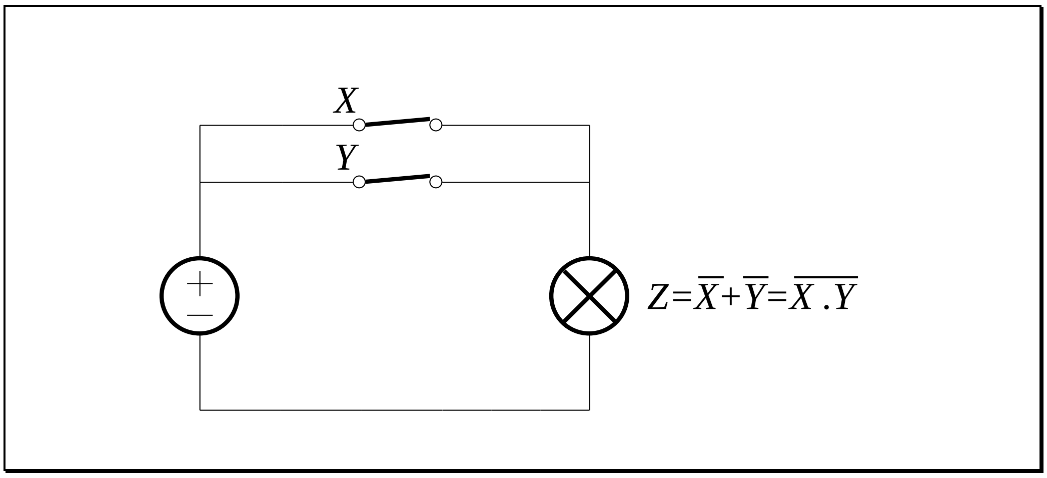


Figure 15.8

15.5.5 The NOR Function

The combination of an OR gate followed by an inverter operates as a gate known as the NOR gate i.e. the NOT OR gate, as shown below:

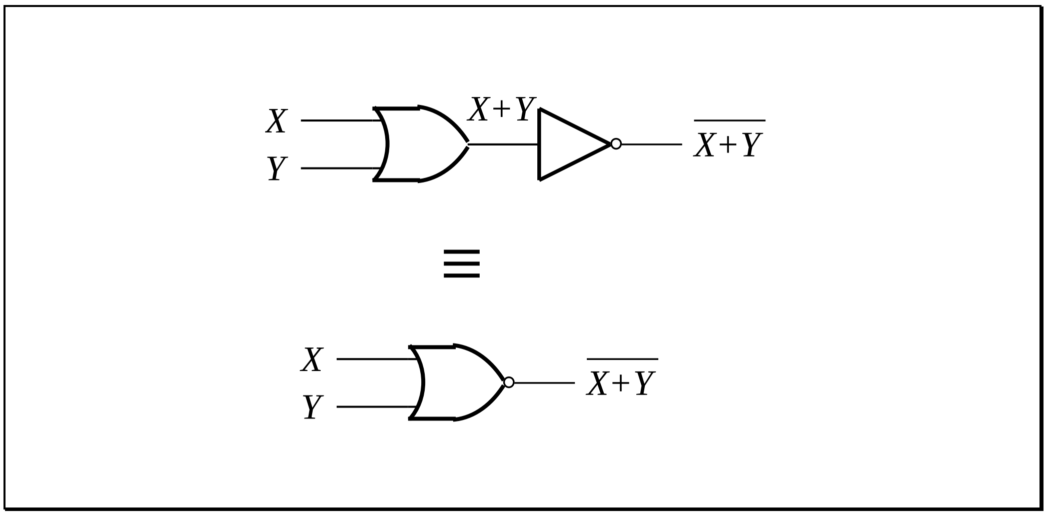


Figure 15.9

The truth table of a NOR gate is shown in Table 15.5.

Input		Output
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Table 15.5 – NOR gate 'truth' table

From this table the output is true when $\overline{X}$ and $\overline{Y}$ are true, i.e. X and Y are both false, thus the NOR function can be represented by the circuit shown below:

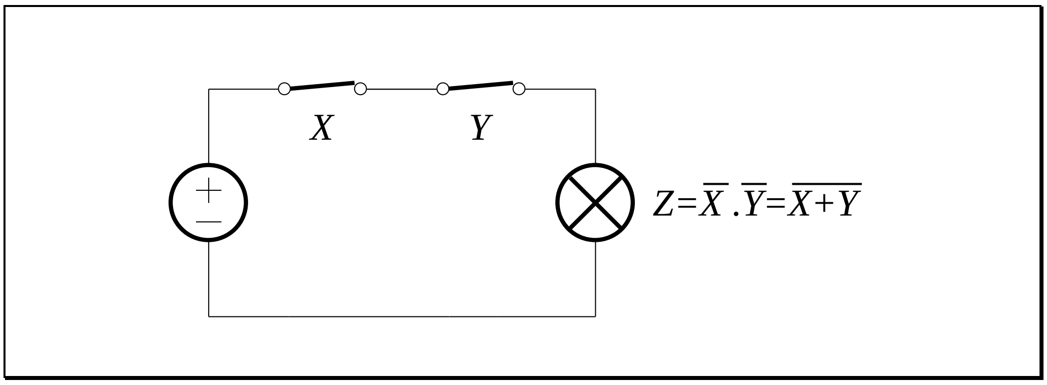


Figure 15.10

15.6 Electronic Logic Families

So far we have considered logic gates as a certain combination of switches. In practice, switches are not a suitable form of input and also cannot be operated fast enough. Generally logic systems are required to operate off an input voltage in a time period which ranges from days to nanoseconds.

The predominant logic family today is made from pairs of transistors known as Complimentary Metal Oxide Semiconductor (CMOS) transistors. They can operate at frequencies in the GHz range and have an extremely low power consumption, making them almost the "ideal" logic family.

15.7 Packing Density

With integrated circuits (ICs) there are several terms associated with the number of transistors that can be placed on a single chip.

- Small Scale Integration (SSI) – up to 100 transistors
- Medium Scale Integration (MSI) – up to 1 000 transistors
- Large Scale Integration (LSI) – up to 100 000 transistors
- Very Large Scale Integration (VLSI) – millions of transistors

15.8 Boolean Algebra

As with normal algebra, the associative, commutative and distributive laws also apply in Boolean algebra. The associative laws are shown in Figure 15.11 and Figure 15.12 for the OR and AND functions – the equivalent logic diagram is also shown.

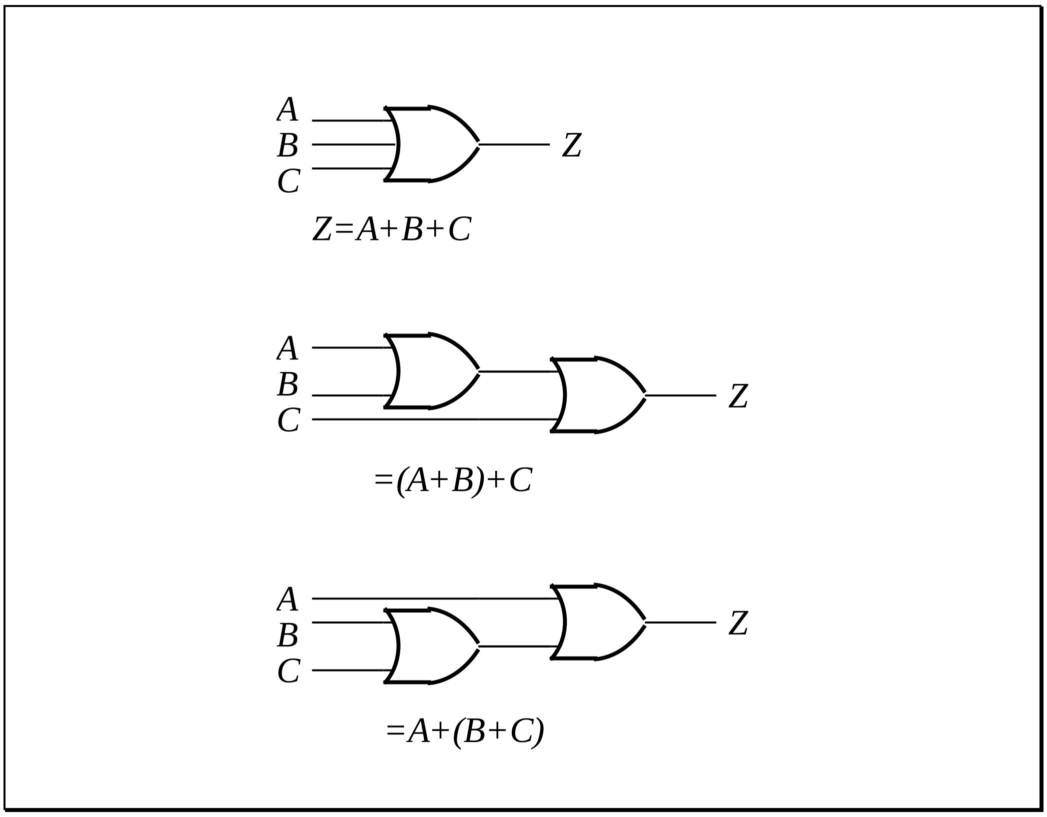


Figure 15.11

15.14

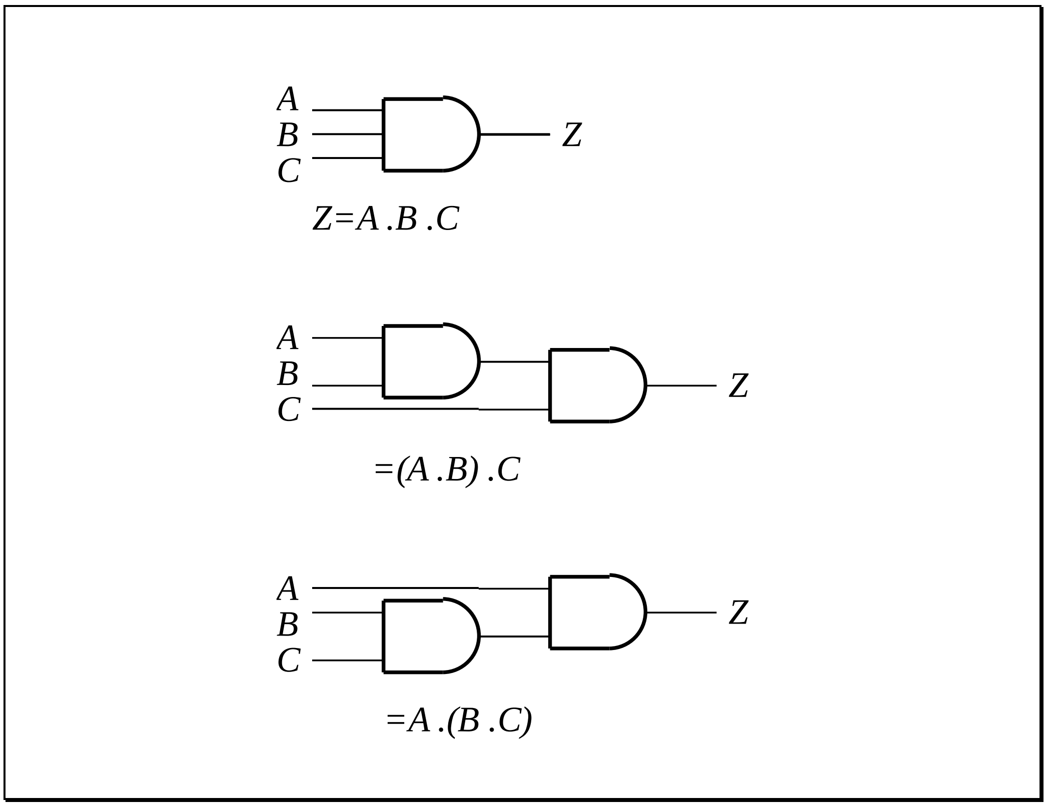


Figure 15.12

The commutative law states:

$A.B = B.A$ and $A + B = B + A$

The distributive law is shown below:

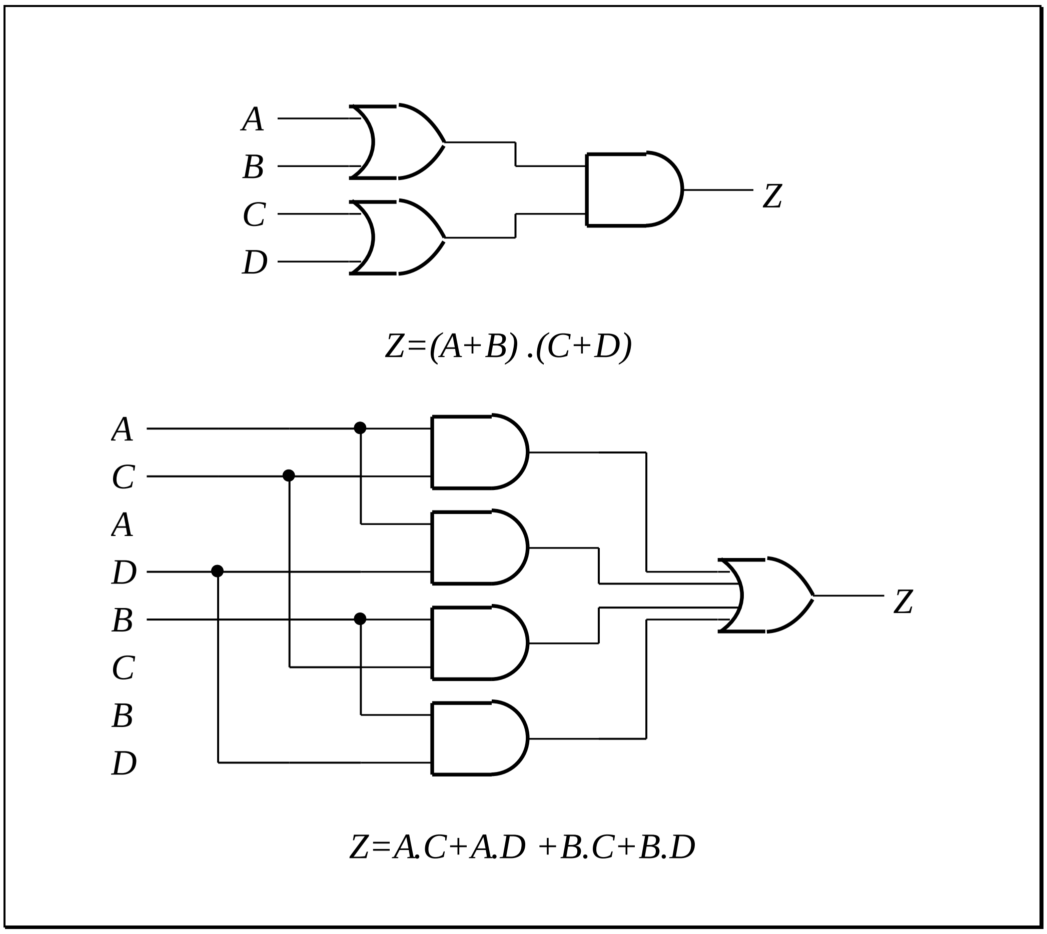


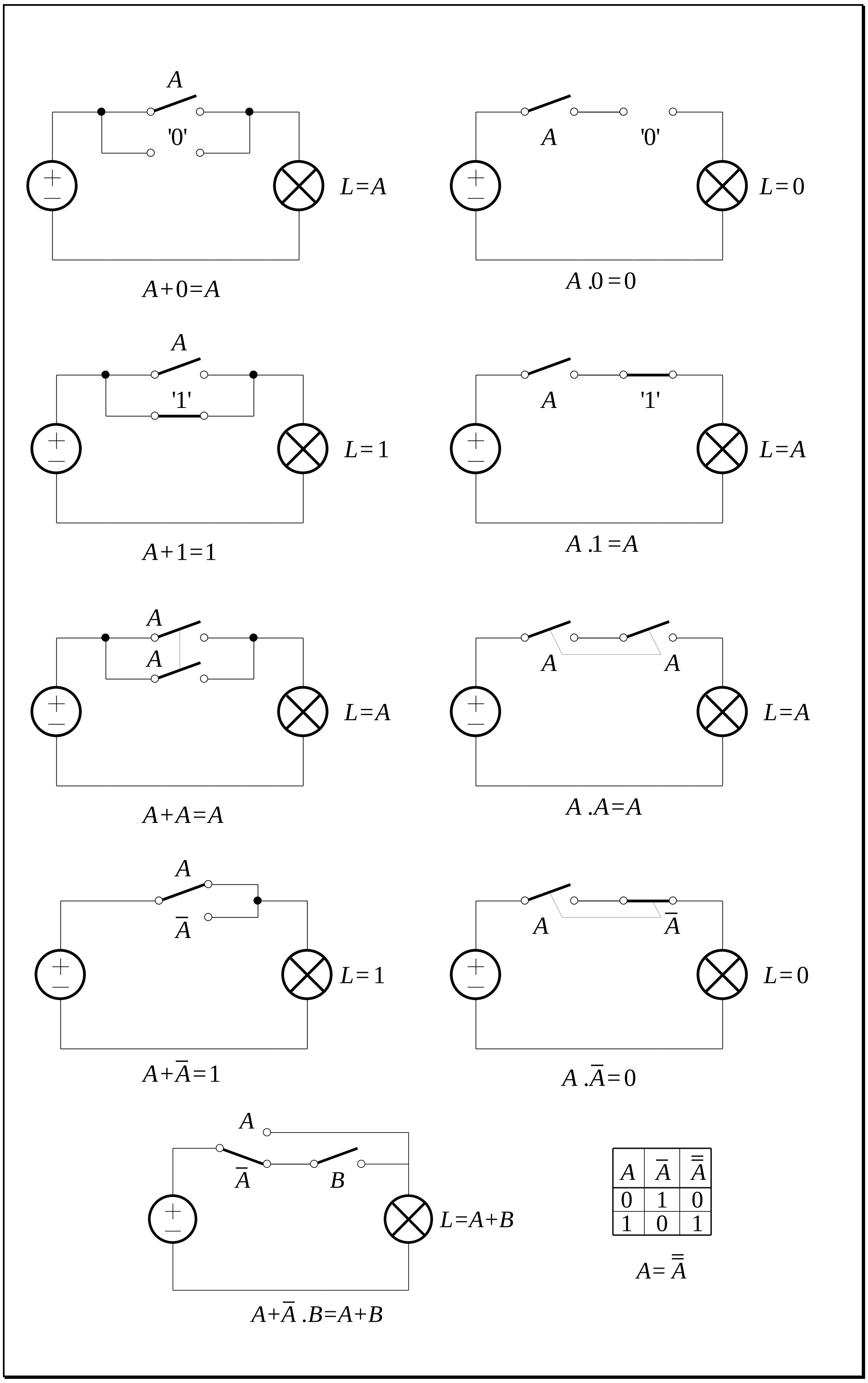
Figure 15.13

15.9 Boolean Identities

There are several Boolean identities. These are summarised below:

$$\begin{array}{cc} A+0=A & A.0=0 \\ A+1=1 & A.1=A \\ A+A=A & A.A=A \\ A+\overline{A}=1 & A.\overline{A}=0 \\ A+\overline{A}B=A+B & \overline{\overline{A}}=A \end{array} \qquad (15.4)$$

A switch analogy of the identities is shown in Figure 15.14.



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

Figure 15.14

15.10 De Morgan's Theorem

De Morgan's Theorem states:

$$\overline{A.B} = \overline{A} + \overline{B}$$
$$\overline{A+B} = \overline{A}.\overline{B} \qquad (15.5)$$

or equivalently:

$$A.B = \overline{\overline{A} + \overline{B}}$$
$$A+B = \overline{\overline{A}.\overline{B}} \qquad (15.6)$$

The theorem can be proved by truth table. This proof is shown below:

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

General rules for De Morgan's Theorem are:

a) change AND to OR and OR to AND

b) negate each variable (we now have the *dual* of the original equation)

c) negate the function

De Morgan's Theorem is used to convert a function containing AND and OR operators to a function containing only one type of operator. The function can then be realised by using only NAND or NOR gates.

EXAMPLE 15.7 De Morgan's Theorem

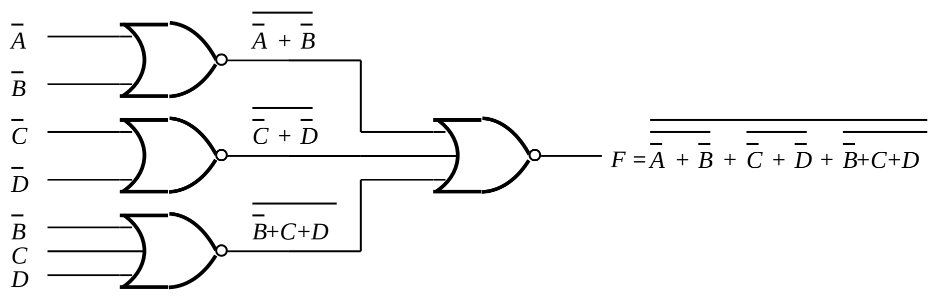
Given:

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

we apply De Morgan's Theorem to get:

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

This function can now be realised by using only NOR gates and is shown below:


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

15.11 Realising Logic

An algebraic expression can be realised from a set of logical statements.

EXAMPLE 15.8 Realising Logic

A punch press is controlled by an electronic logic circuit such that it will operate only under the following conditions:

a) Guard grille is closed, work is in position, next job is waiting.

b) Guard grill is closed, no work is in position, next job is waiting.

c) Guard grille is closed, work is in position, no job is waiting.

To start, Boolean variables must be assigned to each variable:

P = press operated

G = guard grille closed

W = work in position

N = next job waiting

The expression above can now be evaluated

a) $= G.W.N$

b) $= G.\overline{W}.N$

c) $= G.W.\overline{N}$

and thus:

$$P = G.W.N + G.\overline{W}.N + G.W.\overline{N}$$

The Boolean identities can be used to reduce this expression.

$P = G.W.N + G.\overline{W}.N + G.W.\overline{N}$

$P = G.W.N + G.W.N + G.\overline{W}.N + G.W.\overline{N}$ using $A + A = A$

$P = G.W.N + G.\overline{W}.N + G.W.N + G.W.\overline{N}$ using $A + B = B + A$

$P = G.W.\left(N + \overline{N}\right) + G.N.\left(W + \overline{W}\right)$ using commutative law

$P = G.W + G.N$ using $A + \overline{A} = 1$ and $A.1 = A$

$P = G.\left(W + N\right)$ using commutative law

Thus $P = G.W.N + G.\overline{W}.N + G.W.\overline{N} = G.\left(W + N\right) = G.W + G.N$.

This reduction can be verified by a truth table, shown below:

G	W	N		P	$G.W$	$G.N$	$G.W+G.N$
0	0	0	$\overline{G}.\overline{W}.\overline{N}$	0	0	0	0
0	0	1	$\overline{G}.\overline{W}.N$	0	0	0	0
0	1	0	$\overline{G}.W.\overline{N}$	0	0	0	0
0	1	1	$\overline{G}.W.N$	0	0	0	0
1	0	0	$G.\overline{W}.\overline{N}$	0	0	0	0
1	0	1	$G.\overline{W}.N$	1	0	1	1
1	1	0	$G.W.\overline{N}$	1	1	0	1
1	1	1	$G.W.N$	1	1	1	1

When the two variable function $G.W$ is entered into the three variable truth table a "1" appears in the $G.W.N$ and $G.W.\overline{N}$ positions. This is done because we don't care what the value of N is. The same applies for $G.N$.

The figure on the next page shows the various circuit configurations for operating the press.

The circuits shown are:

a) The original equation for press equation

$$P = G.W.N + G.\overline{W}.N + G.W.\overline{N}$$

b) The reduced equation

$$P = G.(W + N)$$

c) NOR gate representation of the reduced equation

$$P = G.(W + N)$$

$$P = \overline{\overline{G} + \overline{W + N}}$$

d) NAND gate representation of the reduced equation

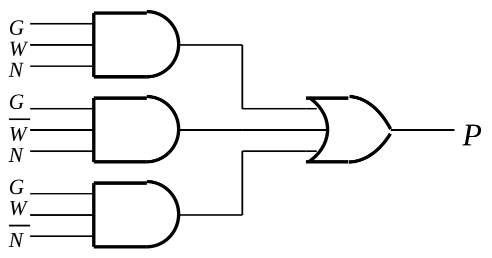
$$P = G.\overline{\overline{W}.\overline{N}} = \overline{\overline{G.\overline{\overline{W}.\overline{N}}}}$$

The final NAND gate is used as an inverter to convert the previous NAND gate to an AND gate.

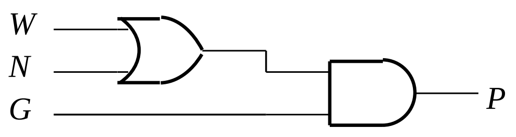
e) NAND gate representation of

$$P = G.W + G.N$$

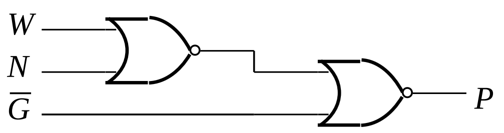
$$= \overline{\overline{G.W}.\overline{G.N}}$$



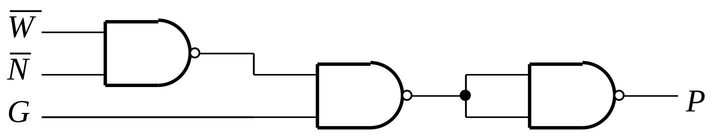
(a) $P=G.W.N+G.\overline{W}.N+G.W.\overline{N}$



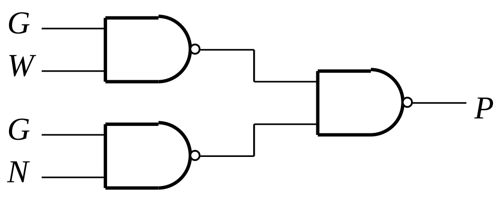
(b) $P=G.(W+N)$



(c) $P=\overline{\overline{G}+\overline{(W+N)}}$



(d) $P=\overline{\overline{G.\overline{(\overline{W}.\overline{N})}}}$



(e) $P=\overline{\overline{G.W}.\overline{G.N}}$

From the circuits, the NOR gate representation in figure (c) is the simplest representation. Figure (b) requires two different gates. We normally try to reduce our logic to that which requires just one type of gate, since different gates are not available in one package. Four two-input NOR gates are available in one package (e.g. 74AC32) and four two-input NAND gates are available in one package (e.g. 74AC00), so both the all NOR and all NAND implementations will use just one IC.

15.12 Summary

- The binary system, which utilises two digits 0, 1 and thus has a base of two, is used in digital logic circuits and computing in general. Binary digits are often referred to as "bits" (<u>b</u>inary dig<u>its</u>). An eight-bit number is called a *byte*.

- A Boolean quantity is a quantity that has only two possible states – 0 or 1. A Boolean variable is a variable that has been assigned to a Boolean quantity.

- Boolean algebra is a new form of algebra, designed to operate with Boolean variables. Some laws are intuitive, whilst some are not. Most rules of the algebra can be derived using truth tables.

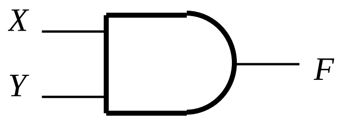
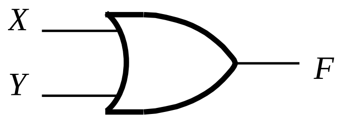
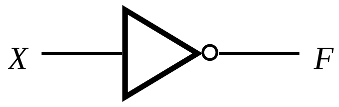
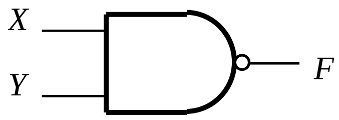
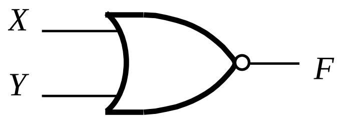
- Some important Boolean identities are:

$$\begin{array}{cc}
A+0=A & A.0=0 \\
A+1=1 & A.1=A \\
A+A=A & A.A=A \\
A+\overline{A}=1 & A.\overline{A}=0 \\
A+\overline{A}B=A+B & \overline{\overline{A}}=A
\end{array}$$

- De Morgan’s Theorem can be used to eliminate either the OR operation or the AND operation:

$$\begin{aligned}
A.B &= \overline{\overline{A}+\overline{B}} \\
A+B &= \overline{\overline{A}.\overline{B}}
\end{aligned}$$

- A summary of the fundamental digital logic gates is given below:

Name	**Graphic symbol**	**Algebraic function**	**Truth table**
AND		$F = X.Y$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{X}$	X | F 0 | 1 1 | 0
NAND		$F = \overline{X.Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

15.13 References

Mano, M. Morris: *Digital Logic and Computer Design*, Prentice-Hall, Sydney, 1979.

Exercises

1.

Write out the first 20 decimal digits in:

a) Base 16 (hex)

b) Base 8 (octal)

c) Base 2 (binary)

2.

Convert the decimal number 250.5 to:

a) Base 8

b) Base 16

c) Base 2

3.

Convert the following decimal numbers to binary:

a) 12.0625

b) 10^4

c) 673.23

d) 1998

4.

Convert the following binary numbers to decimal:

a) 10.10001

b) 101110.0101

c) 1110101.110

d) 1101101.111

5.

Convert:

a) binary 11010111.110 to decimal, octal hexadecimal

b) octal 623.77 to decimal, binary, hexadecimal

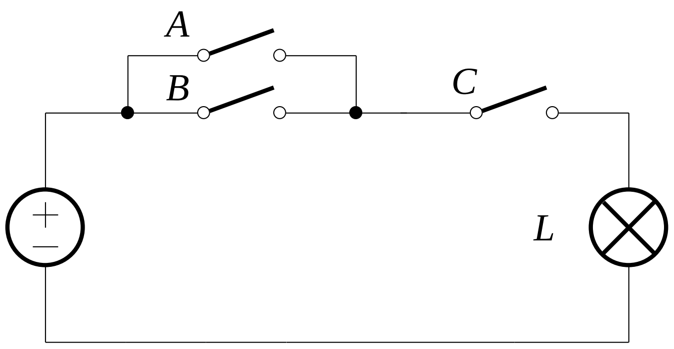
c) hex 2AC5.D to decimal, octal, binary

6.

Write down in BCD the decimal number 8620 and compare this with its binary equivalent.

7.

Express the following switching circuit in binary logic notation:

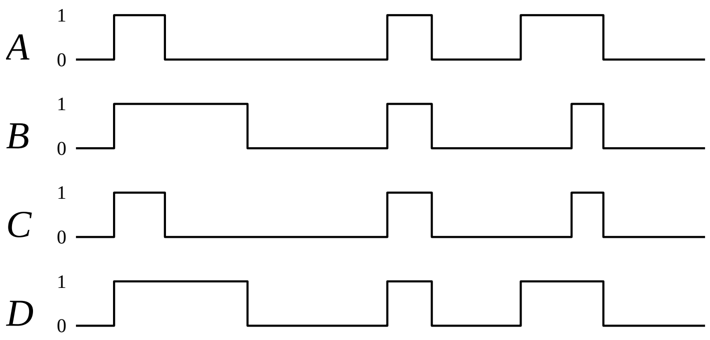


8.

A and *B* are inputs applied to two logic gates. *C* and *D* are the respective outputs of each gate. Name the type of logic gate used to give:

a) output *C*

b) output *D*



9.

Simplify the following Boolean functions to a minimum number of literals:

a) $xy + x\overline{y}$

b) $(x + y)(x + \overline{y})$

c) $xyz + \overline{x}y + xy\overline{z}$

d) $zx + z\overline{x}y$

e) $\overline{(A + B)} \cdot \overline{(\overline{A} + \overline{B})}$

f) $y(w\overline{z} + wz) + xy$

10.

Reduce the following Boolean functions to the required number of literals:

a) $ABC + \overline{A}\overline{B}C + \overline{A}BC + AB\overline{C} + \overline{A}\overline{B}\overline{C}$ to five literals

b) $BC + A\overline{C} + AB + BCD$ to four literals

c) $\overline{(\overline{CD} + A)} + A + CD + AB$ to three literals

d) $(A + C + D)(A + C + \overline{D})(A + \overline{C} + D)(A + \overline{B})$ to four literals

11.

Find the complement of the following Boolean functions and reduce them to a minimum number of literals:

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

b) $\overline{B}D + \overline{A}B\overline{C} + ACD + \overline{A}BC$

c) $(\overline{(AB)}A)(\overline{(AB)}B)$

d) $A\overline{B} + \overline{C}\overline{D}$

12.

Obtain the truth table of the function:

$$F = xy + x\overline{y} + \overline{y}z$$

13.

Implement the simplified Boolean functions from Q10 with logic gates.

14.

Given the Boolean function:

$$F = xy + \bar{x}\,\bar{y} + \bar{y}z$$

a) Implement with AND, OR and NOT gates.

b) Implement with only OR and NOT gates.

c) Implement with only AND and NOT gates.

d) Implement with only NAND gates.

15.

Simplify the functions T_1 and T_2 to a minimum number of literals, and realise the circuit with logic gates.

A	B	C	T_1	T_2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1