11 BOOLEAN ALGEBRA

Objectives

After studying this chapter you should

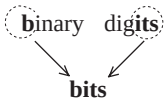
- be able to use AND, NOT, OR and NAND gates;
- be able to use combinatorial and switching circuits;
- understand equivalent circuits;
- understand the laws of Boolean algebra;
- be able to simplify Boolean expressions;
- understand Boolean functions;
- be able to minimise circuits;
- understand the significance of half and full adder circuits.

11.0 Introduction

When *George Boole* (1815-186) developed an algebra for logic, little did he realise that he was forming an algebra that has become ideal for the analysis and design of circuits used in computers, calculators and a host of devices controlled by microelectronics. Boole's algebra is physically manifested in electronic circuits and this chapter sets out to describe the building blocks used in such circuits and the algebra used to describe the logic of the circuits.

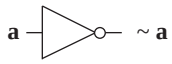
11.1 Combinatorial circuits

The circuits and switching arrangements used in electronics are very complex but, although this chapter only deals with simple circuits, the functioning of all microchip circuits is based on the ideas in this chapter. The flow of electrical pulses which represent the **binary digits** 0 and 1 (known as **bits**) is controlled by combinations of electronic devices. These **logic gates** act as switches for the electrical pulses. Special symbols are used to represent each type of logic gate.



NOT gate

The NOT gate is capable of reversing the input pulse. The truth table for a NOT gate is as follows:



This is a **NOT** *gate*

Input **a**	Output ~**a**
0	1
1	0

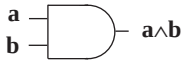
The NOT gate receives an input, either a pulse (1) or no pulse (0) and produces an output as follows :

If input **a** is 1, output is 0;

and if input **a** is 0, output is 1.

AND gate

The AND gate receives two inputs **a** and **b**, and produces an output denoted by $a \wedge b$. The truth table for an AND gate is as follows :



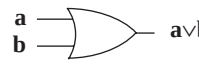
This is an **AND** *gate*

Input **a**	**b**	Output $a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

The only way that the output can be 1 is when **a** AND **b** are both 1. In other words there needs to be an electrical pulse at **a** AND **b** before the AND gate will output an electrical pulse.

OR gate

The OR gate receives two inputs **a** and **b**, and produces an output denoted by $a \vee b$. The truth table for an OR gate is as follows:



This is an **OR** *gate*

Input **a**	**b**	Output $a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

The output will be 1 when **a** or **b** or both are 1.

These three gates, NOT, AND and OR, can be joined together to form **combinatorial circuits** to represent Boolean expressions, as explained in the previous chapter.

Example

Use logic gates to represent

(a) $\sim p \vee q$

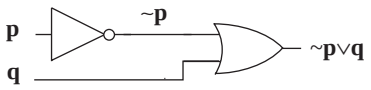
(b) $(x \vee y) \wedge \sim x$

Draw up the truth table for each circuit

Solution

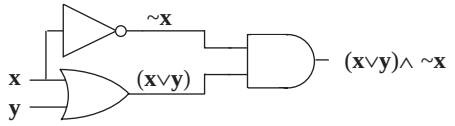
(a)

p	**q**	**~p**	**~p ∨ q**
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1



(b)

x	**y**	**x∨y**	**~x**	**(x∨y)∧~x**
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	0	0



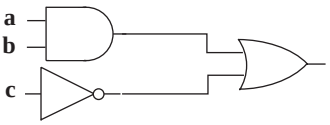
Exercise 11A

Use logic gates to represent these expressions and draw up the corresponding truth tables.

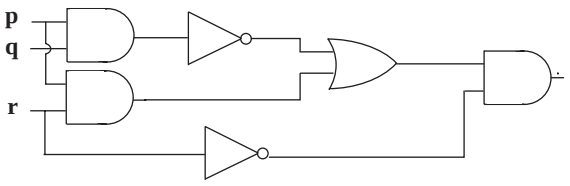
1. $x \wedge (\sim y \vee x)$
2. $a \vee (\sim b \wedge c)$
3. $[a \vee (\sim b \vee c)] \wedge \sim b$

Write down the Boolean expression for each of the circuits below.

4.



5.

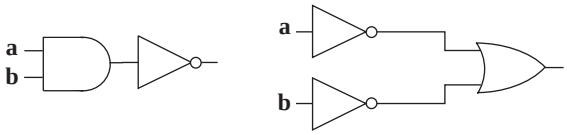


11.2 When are circuits equivalent?

Two circuits are said to be **equivalent** if each produce the same outputs when they receive the same inputs.

Example

Are these two combinatorial circuits equivalent?

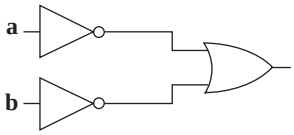


Solution

The truth tables for both circuits will show if they are equivalent :

a	**b**	$\sim(a \wedge b)$
0	0	1
0	1	1
1	0	1
1	1	0

a	**b**	$\sim a \vee \sim b$
0	0	1
0	1	1
1	0	1
1	1	0

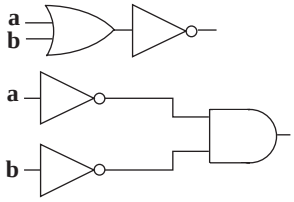


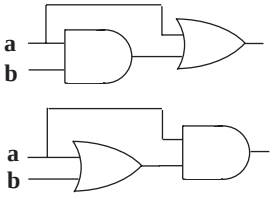
Work through the values in the truth tables for yourself. Since both tables give the same results the two circuits are equivalent. Indeed the two Boolean expressions are equivalent and can be put equal;

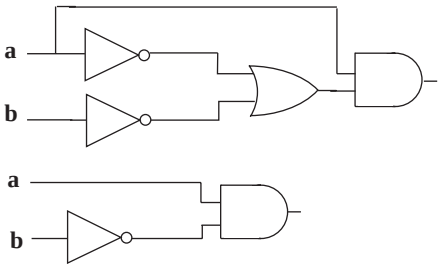
i.e. $\sim(a \wedge b) = \sim a \vee \sim b$

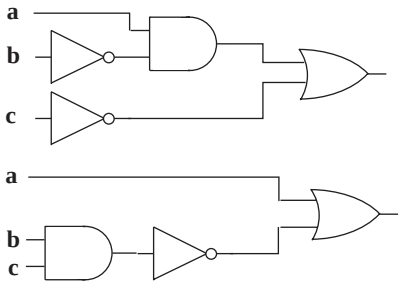
Exercise 11B

Show if these combinatorial circuits are equivalent by working out the Boolean expression and the truth table for each circuit.

1. 

2. 

3. 

4. 

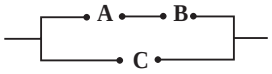
11.3 Switching circuits

A network of switches can be used to represent a Boolean expression and an associated truth table.

Generally the switches are used to control the flow of an electrical current but you might find it easier to consider a switching circuit as a series of water pipes with taps or valves at certain points.

One of the reasons for using switching circuits rather than logic gates is that designers need to move from a combinatorial circuit (used for working out the logic) towards a design which the manufacturer can use for the construction of the electronic circuits.

This diagram shows switches **A**, **B** and **C** which can be **open** or **closed**. If a switch is closed it is shown as a 1 in the following table whilst 0 shows that the switch is open. The **switching table** for this circuit is as follows :



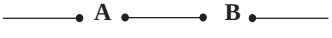
A	B	C	Circuit output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The table shows that there will be an output (i.e. 1) when **A** AND **B** are 1 OR **C** is 1. This circuit can therefore be represented as

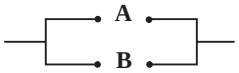
(**A** AND **B**) OR **C**

ie. $(\mathbf{A} \wedge \mathbf{B}) \vee \mathbf{C}$

The circuit just considered is built up of two fundamental circuits:

- a **series circuit**, often called an AND gate, $\mathbf{A} \wedge \mathbf{B}$ 

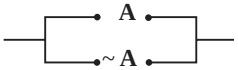
- a **parallel circuit**, often called an OR gate, $\mathbf{A} \vee \mathbf{B}$.



The next step is to devise a way of representing negation. The negation of the truth value 1 is 0 and vice versa, and in switching circuits the negation of a 'closed' path is an 'open' path.

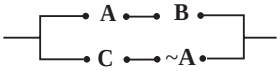
This circuit will always be 'open' whatever the state of **A**. In other words the output will always be 0, irrespective of whether **A** is 1 or 0.

This circuit will always be 'closed' whatever the state of **A**. The output will always be 1 irrespective of whether **A** is 1 or 0.



Example

Represent the circuit shown opposite symbolically and give the switching table.



Solution

The symbolic representation can be built up by considering

the top line of the circuit $(\mathbf{A} \wedge \mathbf{B})$

the top bottom of the circuit $(\mathbf{C} \wedge \sim\mathbf{A})$.

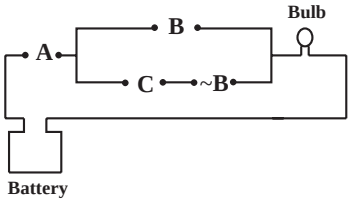
Combining these gives the result $(\mathbf{A} \wedge \mathbf{B}) \vee (\mathbf{C} \wedge \sim\mathbf{A})$

The table is as follows.

A	B	C	~A	$A \wedge B$	$C \wedge \sim A$	$(A \wedge B) \vee (C \wedge \sim A)$
0	0	0	1	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	0	1	0	1

Activity 1 Make your own circuit

Using a battery, some wire, a bulb and some switches, construct the following circuit.



A simple switch can be made using two drawing pins and a paper clip which can swivel to close the switch. The pins can be pushed into a piece of corrugated cardboard or polystyrene.

Using the usual notation of 1 representing a closed switch and 0 representing an open switch, you can set the switches to represent each line of this table:

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

Remember that the '**~B**' switch is always in the opposite state to the '**B**' switch.

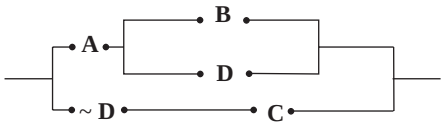
Record the output using 1 if the bulb lights up (i.e. circuit is closed) and 0 if the bulb fails to light (i.e. circuit is open)

Represent the circuit symbolically and draw up another table to see if you have the same output.

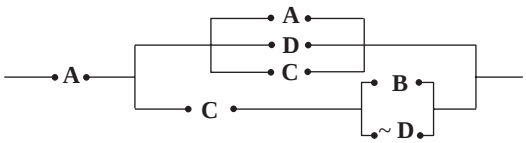
Exercise 11C

Represent the following circuits by Boolean expressions:

1.



2.



Draw switching circuits for these Boolean expressions:

3. $A \vee (\sim B \wedge C)$

4. $A \wedge ((\sim B \wedge C) \vee (B \wedge \sim C))$

11.4 Boolean algebra

A variety of Boolean expressions have been used but George Boole was responsible for the development of a complete algebra. In other words, the expressions follow laws similar to those of the algebra of numbers.

The operators $\wedge$ and $\vee$ have certain properties similar to those of the arithmetic operators such as $+$, $-$, $\times$ and $\div$.

(a) **Associative laws**

$$(a \vee b) \vee c = a \vee (b \vee c)$$

and $(a \wedge b) \wedge c = a \wedge (b \wedge c)$

(b) **Commutative laws**

$$a \vee b = b \vee a$$

and $a \wedge b = b \wedge a$

(c) **Distributive laws**

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

These laws enable Boolean expressions to be simplified and another law developed by an Englishman, *Augustus de Morgan* (1806-1871), is useful. He was a contemporary of Boole and worked in the field of logic and is now known for one important result bearing his name:-

(d) **de Morgan's laws**

$$\sim (a \vee b) = \sim a \wedge \sim b$$

and $\sim (a \wedge b) = \sim a \vee \sim b$

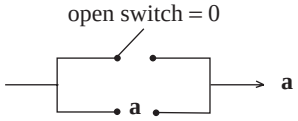
Note: You have to remember to change the connection,

$\wedge$ changes to $\vee$, $\vee$ changes to $\wedge$.

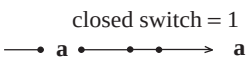
Two more laws complete the range of laws which are included in the Boolean algebra.

(e) **Identity laws**

$$a \vee 0 = a$$

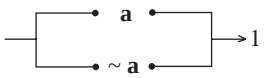


and $a \wedge 1 = a$



(f) **Complement laws**

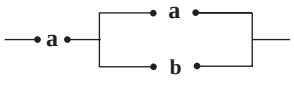
$a \vee \sim a = 1$



and $a \wedge \sim a = 0$

The commutative law can be developed to give a further result which is useful for the simplification of circuits.

Consider the expressions $a \wedge (a \vee b)$ and the corresponding circuit.



If switch a is open (a = 0) what can you say about the whole circuit? What happens when switch a is closed (a = 1)? Does the switch b have any effect on your answers?

The truth table for the circuit above shows that $a \wedge (a \vee b) = a$.

a	b	$a \wedge (a \vee b)$
0	0	0
0	1	0
1	0	1
1	1	1

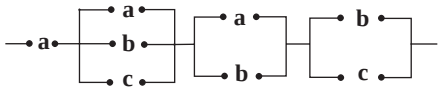
This result can be extended to more switches. For example

if $a \wedge (a \vee b) = a$

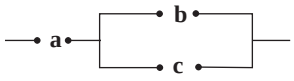
then $a \wedge (a \vee b \vee c) = a$

and $a \wedge (a \vee b \vee c) \wedge (a \vee b) \wedge (b \vee c) = a \wedge (b \vee c)$.

The last of these expressions is represented by this circuit:

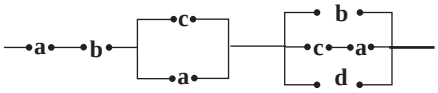


which can be replaced by the simplified circuit:



Example

Write down a Boolean expression for this circuit. Simplify the expression and draw the corresponding circuit.



Solution

$$a \wedge b \wedge (a \vee c) \wedge \big(b \vee (c \wedge a) \vee d\big)$$
$$a \wedge (a \vee c) \wedge b \wedge \big(b \vee (c \wedge a) \vee d\big)$$

Since $a \wedge (a \vee c) = a$

and $b \wedge \big(b \vee (c \wedge a) \vee d\big) = b,$

an equivalent expression is $a \wedge b$

and the circuit simplifies to 

Activity 2 Checking with truth tables

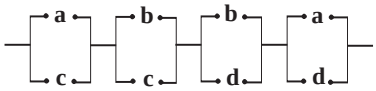
Draw truth tables for the example above to check that

$$a \wedge b \wedge (a \vee c) \wedge \big(b \vee (c \wedge a) \vee d\big) = a \wedge b$$

Exercise 11D

Simplify the following and check your answers by drawing up truth tables:

1. $a \vee (\sim a \wedge b)$
2. $a \wedge \big[b \vee (a \wedge b)\big] \wedge \big[a \vee (\sim a \wedge b)\big]$
3. Simplify the following circuit:



11.5 Boolean functions

In the same way as algebraic functions describe the relationship between the domain, (a set of inputs) and the range (a set of outputs), a **Boolean function** can be described by a **Boolean expression**. For example, if

$$f\,(x_1, x_2, x_3) = x_1 \wedge (\sim x_2 \vee x_3)$$

then f is the Boolean function and

$$x_1 \wedge (\sim x_2 \vee x_3)$$

is the Boolean expression.

Example

Draw the truth table for the Boolean function defined as

$$f(x_1, x_2, x_3) = x_1 \wedge (\sim x_2 \vee x_3)$$

Solution

The inputs and outputs of this Boolean function are shown in the following table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

It is sometimes necessary to form a function from a given truth table. The method of achieving this is described in the following example.

Example

For the given truth table, form a Boolean function

a	**b**	**c**	$f(a, b, c)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solution

The first stage is to look for the places where f (**a**,**b**,**c**) is 1 and then link them all together with 'OR's. For example, in the last row f (**a**, **b**, **c**) = 1 and this is the row in which **a**, **b** and **c** are all true; i.e. when $a \wedge b \wedge c = 1$.

The output is also 1, (i.e. f $(a, b, c) = 1$) in the 7th row which leads to the combination

$$a \wedge b \wedge \sim c = 1$$

Similarly, for the 5th row

$$a \wedge \sim b \wedge \sim c = 1$$

and for the 2nd row

$$\sim a \wedge \sim b \wedge c = 1$$

and for the 1st row

$$\sim a \wedge \sim b \wedge \sim c = 1$$

All these combinations are joined using the connective $\vee$ to give the Boolean expression

$$(a \wedge b \wedge c) \vee (a \wedge b \wedge \sim c) \vee (a \wedge \sim b \wedge \sim c) \vee (\sim a \wedge \sim b \wedge c) \vee (\sim a \wedge \sim b \wedge \sim c)$$

If the values of **a**, **b** and **c** are as shown in the 1st, 2nd, 5th, 7th and 8th row then the value of f $(a, b, c) = 1$ in each case, and the expression above has a value of 1. Similarly if **a, b** and **c** are as shown in the table for which f $(a, b, c) = 0$ then the expression above has the value of 0.

The Boolean function for the truth table is therefore given by

$$f(a, b, c) = (a \wedge b \wedge c) \vee (a \wedge b \wedge \sim c) \vee (a \wedge \sim b \wedge \sim c) \vee (\sim a \wedge \sim b \wedge c) \vee (\sim a \wedge \sim b \wedge \sim c)$$

This is called the **disjunctive normal form** of the function f; the combinations formed by considering the rows with an output value of 1 are joined by the disjunctive connective, OR.

Exercise 11E

Find the disjunctive normal form of the Boolean function for these truth tables:

1.

a	**b**	f **(a, b)**
0	0	1
0	1	0
1	0	1
1	1	0

2.

a	**b**	f **(a, b)**
0	0	1
0	1	1
1	0	0
1	1	1

3.

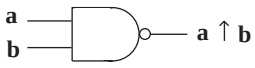
x	**y**	**z**	f **(x, y, z)**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

11.6 Minimisation with NAND gates

When designing combinatorial circuits, efficiency is sought by minimising the number of gates (or switches) in a circuit. Many computer circuits make use of another gate called a NAND gate which is used to replace NOT AND, thereby reducing the number of gates.

The NAND gate receives inputs **a** and **b** and the output is denoted by $a \uparrow b$.

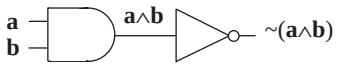
The symbol used is



The truth table for this is

a	**b**	$a \uparrow b$
0	0	1
0	1	1
1	0	1
1	1	0

The NAND gate is equivalent to



Note that, by de Morgan's law, $\sim(a \wedge b) = \sim a \vee \sim b$.

Example

Use NAND gates alone to represent the function

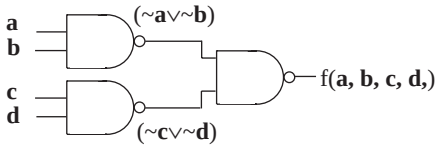
$$f(a, b, c, d) = (a \wedge b) \vee (c \wedge d)$$

Solution

The use of NAND gates implies that there must be negation so the function is rewritten using de Morgan's Laws:

$$(a \wedge b) \vee (c \wedge d) = \sim[(\sim a \vee \sim b) \wedge (\sim c \vee \sim d)]$$

The circuit consisting of NAND gates is therefore as follows:



Example

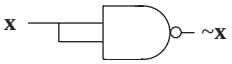
Design combinatorial circuits to represent (a) the negation function $f(x) = \sim x$ and (b) the OR function $f(x, y) = x \vee y$.

Solution

(a) $\sim x = \sim (x \vee x)$

$= \sim x \wedge \sim x$

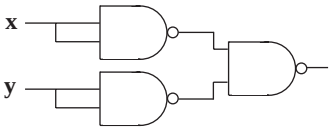
$= x \uparrow x$



(b) $x \vee y = \sim (\sim x \wedge \sim y)$

$= \sim x \uparrow \sim y$

$= (x \uparrow x) \uparrow (y \uparrow y)$



Exercise 11F

Design circuits for each of the following using only NAND gates.

1. $a \wedge b$
2. $a \wedge \sim b$
3. $(\sim a \wedge \sim b) \vee \sim b$

11.7 Full and half adders

Computers turn all forms of data into **b**inary dig**its** , (0 s and 1 s), called bits, which are manipulated mathematically. For example the number 7 is represented by the binary code 00000111 (8 bits are used because many computers use binary digits in groups of 8, for example, ASCII code). This section describes how binary digits can be added using a series of logic gates. The basic mathematical operation is **addition** since

subtraction is the addition of negative numbers,

multiplication is repeated addition,

division is repeated subtraction.

When you are adding two numbers there are two results to note for each column; the entry in the answer and the carrying figure.

254
178

432
11

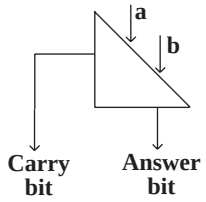
When 4 is added to 8 the result is 12, 2 is noted in the answer and the digit 1 is carried on to the next column.

When adding the second column the carry digit from the first column is included, i.e. $5+7+1$, giving yet another digit to carry on to the next column.

Half adder

The half adder is capable of dealing with two inputs, i.e. it can only add two bits, each bit being either 1 or 0.

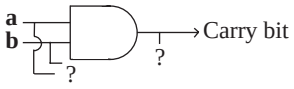
a	b	Carry bit	Answer bit
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Activity 3 Designing the half adder circuit

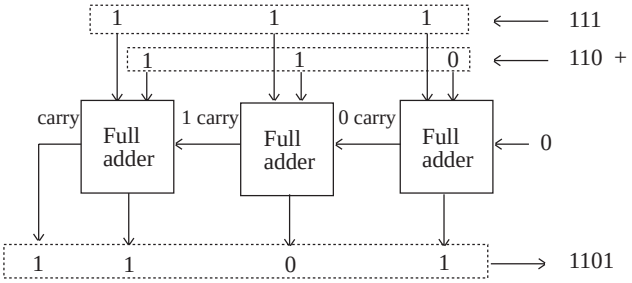
The next stage is to design a circuit which will give the results shown in the table above.

The first part of the circuit is shown opposite; complete the rest of the circuit which can be done with a NOT gate, an OR gate and an AND gate to give the answer bit.



Full Adder

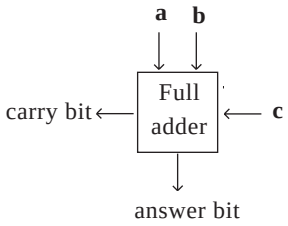
A half adder can only add two bits; a full adder circuit is capable of including the carry bit in the addition and therefore has three inputs.



Activity 4 Full adder truth table

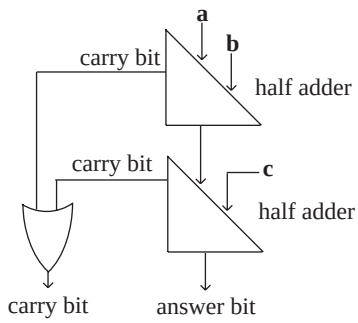
Complete this truth table for the full adder.

Inputs				
a	**b**	**c**	**Carry bit**	**Answer bit**
0	0	0	0	0
0	0	1	0	1
0	1	0		
	etc ↓			
1	1	1		

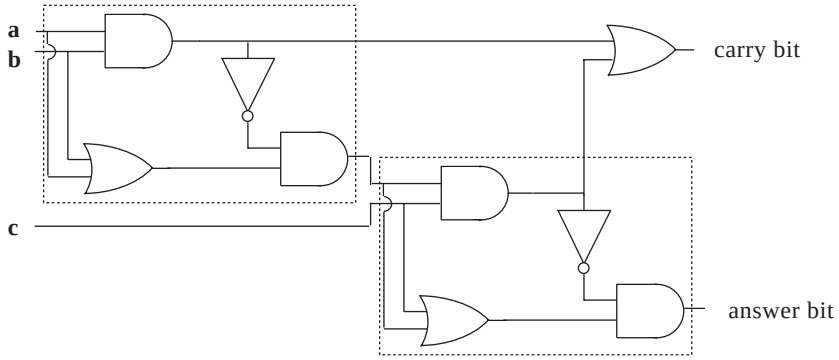


The circuit for a full adder is, in effect, a combination of two half adders.

If you think about it, the carry bit of the full adder must be 1 if either of the two half adders shown gives a carry bit of 1 (and in fact it is impossible for both those half adders to give a carry bit of 1 at the same time). Therefore the two carry bits from the half adders are fed into an OR gate to give an output equal to the carry bit of the full adder.



The circuit for a full adder consists, therefore, of two half adders with the carry bits feeding into an OR gate as follows:



The dotted lines enclose the two half adders with the whole circuit representing a full adder.

Activity 5 NAND half adder

Draw up a circuit to represent a half adder using only NAND gates.

11.8 Miscellaneous Exercises

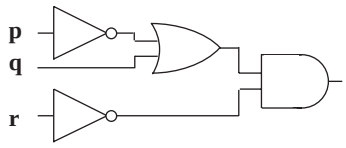
1. Use logic gates to represent these expressions and draw up the corresponding truth tables:

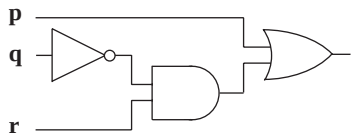
   (a) $\sim[(a \wedge b) \vee c]$

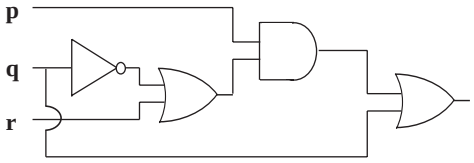
   (b) $(a \wedge b) \vee \sim c$

   (c) $\sim c \wedge [(a \wedge b) \vee \sim (a \wedge c)]$

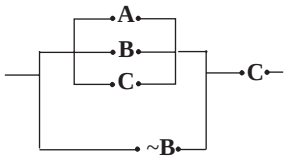
2. Write down the Boolean expression for each of these circuits:

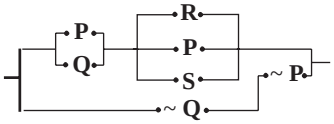
   (a)
   

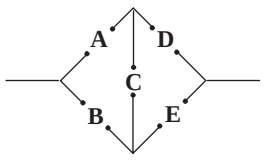
   (b)
   

   (c)
   

3. Write down the Boolean expressions for these circuits :

   (a)
   

   (b)
   

   (c)
   

4. Draw switching circuits for these Boolean expressions:

   (a) $(B \wedge C) \vee (C \wedge A) \vee (A \wedge B)$

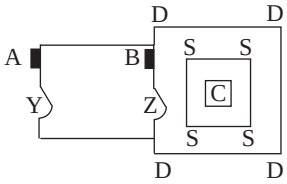
   (b) $\left[A \wedge \left((B \wedge \sim C) \vee (\sim B \wedge C)\right)\right] \vee (\sim A \wedge B \wedge C)$

5. Draw the simplest switching circuit represented by this table:

P	Q	R	S	Output
1	1	1	1	1
0	1	1	1	1
1	1	0	1	1
0	1	0	1	1

6. A burglar alarm for a house is controlled by a switch. When the switch is on, the alarm sounds if either the front or back doors or both doors are opened. The alarm will not work if the switch is off. Design a circuit of logic gates for the alarm and draw up the corresponding truth table.

*7. A gallery displaying a famous diamond uses a special Security Unit to protect access to the Display Room (D). The diagram below shows the layout of the system.



The display cabinet (C) is surrounded by a screen of electronic eyes (S).

Access to the display room is through doors (Y), (Z). Boxes (A), (B) are used in the system. The following persons are involved in the system :

Manager,
Deputy Manager,
Chief Security Officer.

The Display Room is opened as follows :

The Unit must be activated at box A.

Door (Y) is opened by any two of the above persons at box A.

Box B is activated by the Manager and Deputy Manager together.

The screen (S) is activated by the Chief Security Officer alone at box B only.

Door (Z) can only be opened once the screen (S) is activated.

Draw a circuit of logic gates required inside the Unit to operate it. Ensure your diagram is documented.

(AEB)

8. Simplify the following expressions and check your answer by drawing up truth tables.

   (a) $(a \wedge b \wedge c) \vee (\sim a \wedge b \wedge c)$

   (b) $a \vee (\sim a \wedge b \wedge c) \vee (\sim a \wedge b \wedge \sim c)$

   (c) $(p \wedge q) \vee (\sim p \vee \sim q) \wedge (r \vee s)$

9. Find the disjunctive normal form of this function; simplify and draw the combinatorial circuit.

a	**b**	**c**	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

10. Design a circuit representing $\sim a \vee b$ using NAND gates.

*11. Write a computer program or use a spreadsheet that outputs a truth table for a given Boolean expression.

12. (a) Establish a truth table for the Boolean function $f(x_1, x_2, x_3) = (\sim x_1 \vee x_2) \wedge (\sim x_3 \vee x_2)$.

    (b) Design a circuit using as few AND, OR and NOT gates as possible to model the function in (a).

13. (a) Show, by constructing truth tables or otherwise, that the following statements are equivalent.

    $p \Rightarrow q$ and $\sim (\sim (p \wedge q) \wedge p)$.

    (b) With the aid of (a), or otherwise, construct combinatorial circuits consisting only of NAND gates to represent the functions

    $f(x,y) = x \Rightarrow y$ and $g(x,y) = \sim (x \Leftrightarrow y)$.