Chapter 4
Logic gates and logic circuits

Learning objectives

By the end of this chapter you should be able to:

- use logic gate symbols for NOT, AND, OR, NAND, NOR and XOR
- understand and define the functions of the NOT, AND, OR, NAND, NOR and XOR (EOR) gates
- construct the truth table for each of the above logic gates
- construct a logic circuit from:
  - a problem statement
  - a logic expression
  - a truth table
- construct a truth table from:
  - a problem statement
  - a logic circuit
  - a logic expression
- construct a logic expression from:
  - a problem statement
  - a logic circuit
  - a truth table.

4.01 Boolean logic and problem statements

Consider the following question:

Is Colombo further north than Singapore?

In everyday language the answer will be either yes or no. (‘Yes’, in fact.) However, the question could be rephrased to make use of the language of Boolean logic:

Colombo is further north than Singapore: TRUE or FALSE?

More formally, the statement:

Colombo is further north than Singapore.

can be described as an example of a logic assertion or a **logic proposition** that can have only one of the two alternative Boolean logic values: TRUE or FALSE.

Now consider the following two individual statements.

- You should take an umbrella if *it is raining* or if *the weather forecast is for rain later*.
- The air-conditioning system is set to come on in an office only *during working hours* but also only *if the temperature rises to above 25°C*.

Each of these statements contains two logic propositions which are highlighted. In each statement these logic propositions are combined in some way. Finally, each statement has the addition of an outcome which is dependent on the combination of the two propositions. Each of these is, therefore, an individual example of a **problem statement**.

4.02 Boolean operators

The problem statements identified above can be more formally expressed in a form that is suitable for handling with Boolean logic. To do this it is necessary to use Boolean operators. The three basic Boolean operators are AND, OR and NOT.

The definition for AND, OR and NOT can be expressed as:

- A AND B is TRUE if A is TRUE and B is TRUE
- A OR B is TRUE if A is TRUE or B is TRUE
- NOT A is TRUE if A is FALSE.

Here, both A and B represent any logic proposition or assertion that has a value TRUE or FALSE.

The two problem statements above can now be rephrased as follows:

- Take_umbrella = TRUE IF (raining = TRUE) OR (rain_forecast = TRUE)
- System_on = TRUE IF (office hours = TRUE) AND (temperature > 25°C).

Each original problem statement has now been rephrased to include a form of **logic expression**. The format of each expression here does not follow any formally defined convention but the structure does allow the underlying logic to be understood. In general, a logic expression consists of logic propositions combined using Boolean operators. The expression may be included in an equation with a defined output.

Any logic expression can be constructed using only the Boolean operators AND, OR and NOT but it is often convenient to use other operators. Here are the definitions for the three other operators that you should be familiar with:

- A NAND B is TRUE if A is FALSE or B is FALSE
- A NOR B is TRUE if A is FALSE and B is FALSE
- A XOR B is TRUE if A is TRUE or B is true but not both of them.

WORKED EXAMPLE 4.01

Constructing a logic expression from a problem statement

Consider the following problem statement.

A shopkeeper orders a delivery of goods at the end of each month. However, if the stock of a particular item falls to the re-order level before the end of the month, a delivery is ordered immediately. Also, if a regular customer orders a large amount of goods, a delivery is ordered immediately.

We need to identify the conditions in the statement that can have true or false values. These can be underlined:

A shopkeeper orders a delivery of goods at the end of each month. However, if the stock of a particular item falls to the re-order level before the end of the month a delivery is ordered immediately. Also, if a regular customer orders a large amount of goods a delivery is ordered immediately.

The conditions can now be collected together in one logic expression:

End_of_month OR re-order_level_reached OR (regular_customer AND large_amount)

To simplify this we change each condition into a symbol.

- Let A represent End_of_month.
- Let B represent re-order_level_reached.

- Let C represent regular customer.
- Let D represent large_amount.

The logic expression can now be written in an equation using X to represent ‘a delivery is ordered’:

X = A OR B OR (C AND D)

TASK 4.01

Convert the conditions in the following problem statement into a simple logic expression:

A document can only be copied if it is not covered by copyright or if there is copyright and permission has been obtained.

4.03 Truth tables

The truth table is a simple but powerful technique for representing any logic expression or for describing the possible outputs from a logic circuit.

A truth table is presented by making use of the convention that TRUE can be represented as 1 and FALSE can be represented as 0. The simplest use of a truth table is to represent the logic associated with a Boolean operator.

As an example, let us consider the AND operator. The labelling of the truth table follows the convention that the initially defined values are represented by A and B and the value obtained from the simple expression using the AND operator is represented by X. In other words, we write the truth table for X = A AND B. Remembering that AND only returns TRUE if both A and B are TRUE we expect a truth table with only one instance of X having the value 1. The truth table is shown in Table 4.01.

The truth table has four rows corresponding to the four combinations of the truth values for A and B. Three of these lead to a 0 in the X column as expected.

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Table 4.01 The truth table for the AND operator

TIP

When constructing a truth table make sure that the left-hand columns for the input values are written as though they were increasing binary values.

TASK 4.02

Without looking further on in the chapter, construct the truth table for the OR operator.

4.04 Logic circuits and logic gates

The digital circuits that constitute the inner workings of a computer system operate as logic circuits where each individual part of the circuit is either in an ‘on’ state, corresponding to a 1, or in an off state, corresponding to a 0. A logic circuit comprises component parts called logic gates. Each different **logic gate** has an operation that matches a Boolean operator.

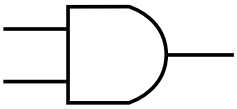


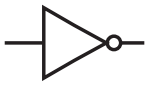
Figure 4.01 The symbol for the AND logic gate

When drawing a circuit, standard symbols are used for the logic gates. As an example, the symbol shown in Figure 4.01 represents an AND gate.

The first point to note here is that the shape of the symbol tells us the type of gate. The second point is that the inputs are shown on the left-hand side and the output is shown on the right-hand side. In general, the number of inputs is not limited to two. We will only consider circuits where the number of inputs is two or fewer.

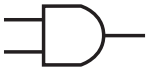
Figure 4.02 shows the logic gate symbols and the associated truth tables for each of the six Boolean operators introduced in Section 4.02.

NOT



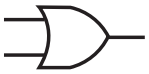
A	X
0	1
1	0

AND



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

OR

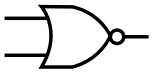


A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

NAND

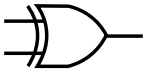
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

XOR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Figure 4.02 Logic gate symbols and their associated truth tables

There are two other points to note here. The NOT gate is a special case, having only one input. The second point is that a NAND gate is a combination of a AND gate followed by a NOT gate, and a NOR gate is a combination of an OR gate followed by a NOT gate. NAND and NOR gates produce a complementary output to the AND and OR gates.

TASK 4.03

Draw a circuit where A and B are input to an AND gate, from which the output is carried to a NOT gate, from which there is an output X. Show that this has the same outcome as having one NAND gate.

Extension Question 4.01

Could the same outcome be produced by positioning a NOT gate before the AND gate?

TIP

You need to remember the symbol for each of these gates. A good start here is to remember that AN**D** has the proper **D** symbol and OR has the curvy one.

You also need to remember the definitions for the gates so that you can construct the corresponding truth table for each gate.

Question 4.01

Can you recall from memory the symbols and definitions of the six logic gates introduced in this chapter?

WORKED EXAMPLE 4.02

Constructing a logic circuit from a problem statement or logic expression

You need to be able to construct a logic circuit from either a problem statement or from a logic expression. If you are given a problem statement, the best approach is to first convert it to a logic expression.

Consider the following problem statement: A bank offers a special lending rate to customers subject to certain conditions. To qualify, a customer must satisfy certain criteria.

- The customer has been with the bank for two years.
- Two of the following conditions must also apply:
  - the customer is married
  - the customer is aged 25 years or older
  - the customer's parents are customers of the bank.

To convert this statement to a logic expression using symbols we can choose:

- let A represent an account held for two years
- let B represent that the customer is married
- let C represent that the customer is aged 25 years or older
- let D represent that the customer's parents have an account.

The logic expression can then be written as:

A AND (((B AND C) OR (B AND D)) OR (C AND D))

This could alternatively be presented with an outcome:

Special_rate IF A AND (((B AND C) OR (B AND D)) OR (C AND D))

alternatively as

X = A AND (((B AND C) OR (B AND D)) OR (C AND D))

Note the use of brackets to ensure that the meaning is clear. You may think that not all of the

brackets are needed. In this example, an extra pair has been included to guide the construction of the circuit where only two inputs are allowed for any of the gates.

From this, we can see that the logic circuit corresponding to this logic expression derived from the original problem statement could be constructed using four AND gates and two OR gates as shown in Figure 4.03.

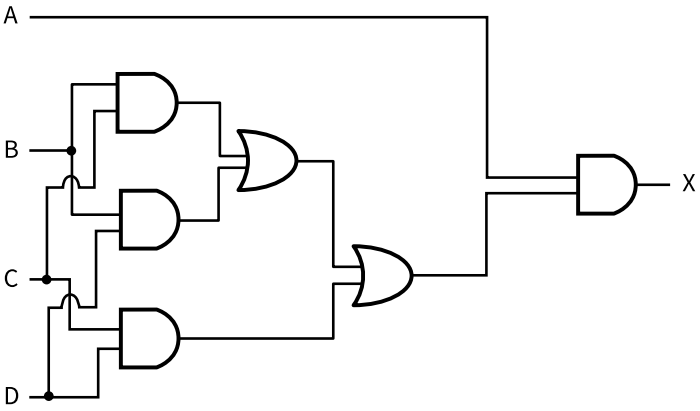


Figure 4.03 A logic circuit constructed from a problem statement

WORKED EXAMPLE 4.03

Constructing a truth table from a logic expression or logic circuit

You also need to be able to construct a truth table from either a logic expression or a logic circuit. We might have continued with the problem in Worked Example 4.02 but four inputs will lead to 16 rows in the truth table. Instead, we consider a slightly simpler problem with only three inputs and therefore only eight rows in the truth table. We will start with the circuit shown in Figure 4.04.

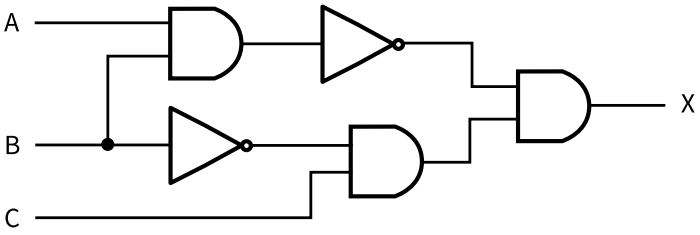


Figure 4.04 A circuit with three inputs for conversion to a truth table

Table 4.02 shows how the truth table needs to be set up initially. There are two points to note here. The first is that you must take care to include all of the eight different possible combinations of the input values. The second point is that for such a circuit it is not sensible to try to work out the outputs directly from the input values. Instead a systematic approach should be used. This involves identifying intermediate points in the circuit and recording the values at each of them in the columns headed ‘Workspace’ in Table 4.02.

Inputs			**Workspace**				**Output**
A	**B**	**C**					**X**
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					

1	1	1					

Table 4.02 The initial empty truth table

Figure 4.05 shows the same circuit but with four intermediate points labelled M, N, P and Q identified. Each one has been inserted on the output side of a logic gate.

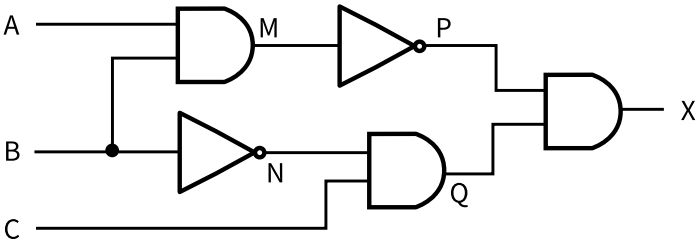


Figure 4.05 The circuit in Figure 4.04 with intermediate points identified

Now you need to work systematically through the intermediate points. You start by filling in the columns for M and N. Then you fill in the columns for P and Q which feed into the final AND gate. The final truth table is shown as Table 4.03. The circuit has two combinations of inputs that lead to a TRUE output from the circuit.

The columns containing the intermediate values (the workspace) could be deleted at this stage.

Inputs			**Workspace**				**Output**
A	**B**	**C**	**M**	**N**	**P**	**Q**	**X**
0	0	0	0	1	1	0	0
0	0	1	0	1	1	1	1
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	1	1	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	0	0

Table 4.03 The truth table for the circuit shown in Figure 4.05

One final point to make here is that you may be able to check part of your final solution by looking at just part of the circuit. For this example, if you look at the circuit you will see that the path from input C to the output passes through two AND gates. It follows, therefore, that for all combinations with C having value 0 the output must be 0. Therefore, in order to check your final solution you only need to examine the other four combinations of input values where C has value 1.

If a logic circuit is to be constructed from a truth table, the first stage is to create a logic expression. To do this only the rows producing a 1 output are used. Consider the truth table shown in Table 4.04. There are three rows producing a 1 output. Each of these produces a logic expression with AND operators. These three logic expressions are then combined with OR operators.

Inputs			**Output**
A	**B**	**C**	**X**
0	0	0	0
0	0	1	1
0	1	0	0

0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 4.04 A truth table to be converted to a logic circuit

The three rows that produce a 1 output have the following values for the inputs:

A = 0, B = 0 and C = 1

A = 0, B = 1 and C = 1

A = 1, B = 0 and C = 0

Each one can be converted to a logical expression:

NOT A AND NOT B AND C

NOT A AND B AND C

A AND NOT B AND NOT C

The combination of the three individual expressions produces the following:

NOT A AND NOT B AND C

OR

NOT A AND B AND C

OR

A AND NOT B AND NOT C

This could be used to create a logic circuit, but the circuit would be quite complex. In Chapter 19 methods will be discussed that allow the simplest possible circuit to be constructed for a given logic problem.

If a logic expression is to be constructed from a logic circuit the first step is to construct a truth table from the circuit. Then the above method can be applied to this truth table.

TASK 4.04

An oven has a number of components that should all be working properly. For each component there is a signalling mechanism that informs a management system either if all is well, or if there is a problem. Table 4.05 summarises the signal values that record the status for each component.

Signal	Value	Component condition
A	0	Fan not working
	1	Fan working properly
B	0	Internal light not working
	1	Internal light working properly
C	0	Thermometer reading too high
	1	Thermometer reading in range

Table 4.05 Signals from the oven components

If the thermometer reading is in range but either or both the fan and light are not working, the management system has to output a signal to activate a warning light on the control panel. Draw a logic circuit for this fault condition.

Reflection Point:

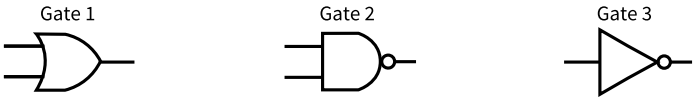
Looking back over the chapter content, what would you say is the central concept in the subject matter?

Summary

- A logic scenario can be described by a problem statement or a logic expression.
- A logic expression comprises logic propositions and Boolean operators.
- Logic circuits are constructed from logic gates.
- The operation of a logic gate matches that of a Boolean operator.
- The outcome of a logic expression or a logic circuit can be expressed as a truth table.
- A logic expression can be created from a truth table using the rows that provide a 1 output.

Exam-style Questions

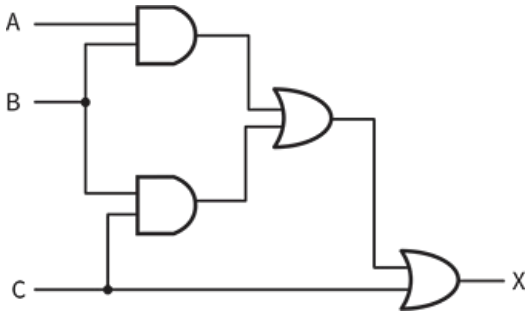
1 **a** The following are the symbols for three different logic gates.



i Identify each of the logic gates. [3]

ii Sketch the truth table for either Gate 1 or Gate 2. [2]

b Consider the following circuit:



i Complete the truth table for the circuit using the following template:

Inputs			**Workspace**			**Output**
A	**B**	**C**				**X**
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

[8]

ii There is an element of redundancy in this diagram. Explain what the problem is. [2]

2 **a** The definition of the NAND gate can be expressed as:

A NAND B is TRUE if A is FALSE or B is FALSE

Sketch the truth table for a NAND gate. [2]

b Consider the following statement:

In a competition, two teams play two matches against each other. One of the teams is declared the winner if one of the following results occurs:

- The team wins both matches.
- The team wins one match and loses the other but has the highest total score.

i Identify the **three** logic propositions in this statement. [3]

ii By assigning the symbols A, B and C to these three propositions give the outcome of the competition as a logic expression. [3]

iii Sketch a logic circuit to match this logic expression. [4]

3 A domestic heating system has a hot water tank and a number of radiators. There is a computerised management system which receives signals. These signals indicate whether or not the conditions for

components are as they should be. The following table summarises the signals received:

Signal	Value	Component condition
A	0	Water flow in the radiators is too low
	1	Water flow in the radiators is within limits
B	0	Hot water tank temperature too high
	1	Hot water tank temperature within limits
C	0	Water level in hot water tank too low
	1	Water level in hot water tank within limits

a Consider the following fault condition. The water level in the hot water tank is too low and the temperature in the hot water tank is too high. The management system must output a signal to switch off the system.

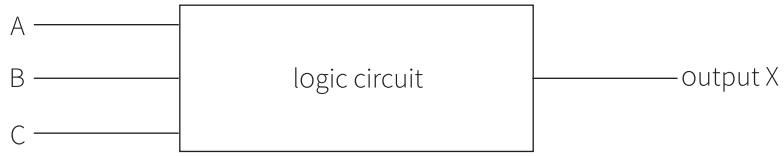
i Sketch a truth table for this fault condition including the A, B and C signals. [4]

ii Sketch the circuit diagram for this fault condition to match this truth table. [5]

b Consider the fault condition where the hot water tank temperature is within limits but the water flow in the radiators is too low and the water level in the hot water tank is too low. Sketch the circuit diagram for this fault condition which requires the management system to output a signal to increase water pressure. [5]

4 **a** Three digital sensors A, B and C are used to monitor a process. The outputs from the sensors are used as the inputs to a logic circuit.

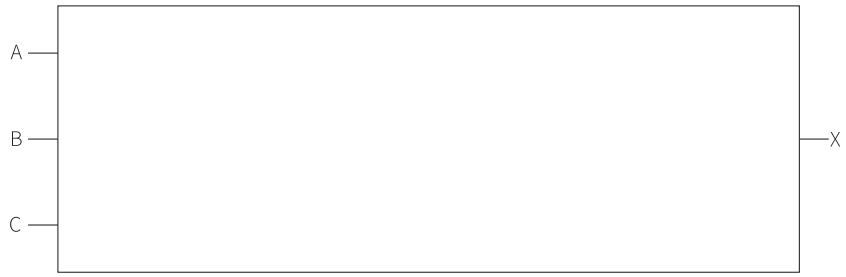
A signal, X, is output from the logic circuit:



Output, X, has a value of 1 if either of the following two conditions occur:

- sensor A outputs the value 1 OR sensor B outputs the value 0
- sensor B outputs the value 1 AND sensor C outputs the value 0

Draw a logic circuit to represent these conditions.



[5]

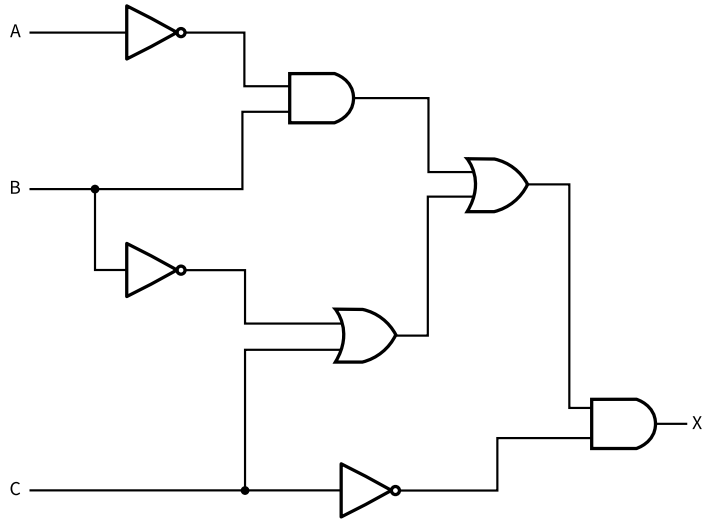
b Complete the truth table for the logic circuit described in **part (a)**.

A	B	C	Workspace	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		

1	0	1		
1	1	0		
1	1	1		

[4]

c Write a logic statement that describes the following logic circuit.



[3]

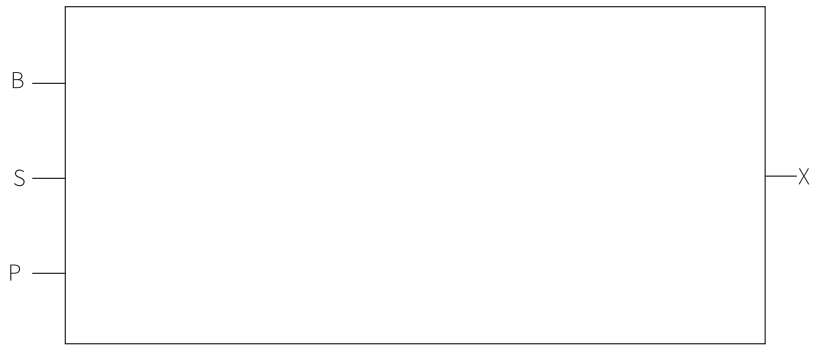
Cambridge International AS & A level Computer Science 9608 paper 13 Q6 June 2015

5 a A student writes the following logic expression:

X is 1 IF (B is NOT 1 AND S is NOT 1) OR (P is NOT 1 AND S is 1)

Draw a logic circuit to represent this logic expression.

Do **not** attempt to simplify the logic expression.



[6]

b Complete the truth table for the logic expression given in **part (a)**.

B	S	P	Workspace	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[4]

Cambridge international AS & A Level Computer Science 9608 paper 12 Q1 November 2016