

PN-ACM-252
110249

Introductory
DIGITAL
SYSTEMS for Engineering

Mahomed Rafi Bera



Juta & Co, Ltd

A

Acknowledgement:

The authors and publishers wish to thank the following persons and institutions for their invaluable contribution to the development of this publication:

- The United States Agency for International Development (USAID), for funding the project. This Materials Development Project formed part of the Tertiary Education Linkages Project (TELP) which focused on capacity building at Historically Disadvantaged Technikon through the establishment of linkages with universities in the United States of America.
- Contributors and moderators from the following South African institutions: Mangosuthu Technikon, ML Sultan Technikon, Peninsula Technikon, Technikon Eastern Cape, Technikon Northern Gauteng, Technikon Southern Africa.
- Contributors and moderators from the United States University Consortium comprising Howard University, Massachusetts Institute of Technology, Clark Atlanta University, North Carolina A & T University.
- The Motorola Corporation, Atlanta, USA, for permission to reproduce the data sheets in Appendix 2.

Note from the author

Special thanks go to Ahmed Moolla (Senior Lecturer, Department of Electronic Engineering, ML Sultan Technikon) for his support and assistance, and to Hemant Laloo (Lecturer, Department of Electronic Engineering, ML Sultan Technikon) for his willingness to test the preliminary manuscript in a real work situation and for checking the answers to exercises.

This publication is dedicated to my wife, without whose patience and support this project would not have been possible.

Mahomed Rafi Bera

First published 1999
ISBN 0 7021 4405 3

© Juta & Co, Ltd 1999
P.O. Box 14373, Kenwyn 7790

This book is copyright under the Berne Convention. In terms of the Copyright Act 98 of 1978, no part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the Publisher.

Editor: Ilse von Zeuner
Illustrators: Dennis Bagnall, Renato Balona, Hugh Lane
Book design and typesetting: JK Type & Graphic cc, Roodepoort
Cover design: Eugène Badenhorst
Printed and bound in South Africa by
The Rustica Press, Ndabeni, Western Cape

D7019

Table of contents

Preface

Unit 1	Introductory concepts	1
	Study objectives	1
1	What is in this book?	1
2	Overview	3
3	Analog or digital?	4
4	Representing digital information	7
5	Bits and bytes	8
6	Summary	9
	Self-evaluation	9
	Answers to activities	10
Unit 2	Basic logic functions	12
	Study objectives	12
1	Introduction	12
2	What are the basic logic functions?	12
3	The AND function	13
4	The OR function	21
5	The NOT function (INVERTER)	25
6	Summary	28
	Self-evaluation	28
	Answers to activities	30
Unit 3	Working with logic functions	33
	Study objectives	33
1	Introduction	33
2	How many rows in a truth table?	34
3	What happens when logic gates are combined?	35
4	Writing the Boolean expression for a logic circuit	37
5	Considering more complex circuits	38
6	Applying digital signals to these logic circuits	41
7	Practical implementation	43
8	Troubleshooting logic circuits	45
9	Summary	46
	Self-evaluation	46
	Answers to activities	49

Unit 4	The NAND and NOR logic gates	54
	Study objectives	54
	1 Introduction	54
	2 Combining the AND gate with the NOT gate.....	54
	3 Combining the OR gate with the NOT gate	55
	4 Applying digital signals	56
	5 Practical implementation	58
	6 Combining the various logic gates	59
	7 Drawing the logic circuit from the Boolean expression ...	61
	8 Troubleshooting logic circuits	64
	9 Simple applications of NAND and NOR gates	66
	10 Summary	69
	Self-evaluation	70
	Answers to activities	72
Unit 5	Simplification using Boolean algebra	79
	Study objectives	79
	1 Introduction	79
	2 Simple single-variable Boolean laws	80
	3 Two other useful Boolean laws	85
	4 The well-known DeMorgan's laws	87
	5 Generating Boolean expressions from truth tables	89
	6 Summary	93
	Self-evaluation	93
	Answers to activities	95
Unit 6	Simplification using Karnaugh maps	100
	Study objectives	100
	1 Introduction	100
	2 What are Karnaugh maps?.....	100
	3 Generating a Karnaugh map form a truth table.....	102
	4 Simplification using the Karnaugh map	107
	5 Summary	117
	Self-evaluation	117
	Answers to activities	120
Unit 7	Basic combinational logic circuits	128
	Study objectives	128
	1 Introduction	128
	2 Combinational logic circuits	129
	3 Design techniques	130
	4 The binary number system	135

d

5	The Exclusive-OR and the Exclusive-NOR gates	140
6	Comparing binary quantities	143
7	Comparator circuits	145
8	Summary	156
	Self-evaluation	156
	Answers to activities	158
Unit 8	Application of combinational logic	170
	Study objectives	170
1	Introduction	170
2	Number systems	170
3	Adders	178
4	Decoders and encoders	185
5	Multiplexers and demultiplexers	191
6	Summary	195
	Self-evaluation	195
	Answers to activities	197
Unit 9	Sequential logic circuits	202
	Study objectives	202
1	Introduction	202
2	Latches and flip-flops	203
3	Shift registers	216
4	Asynchronous counters	224
5	Summary	229
	Self-evaluation	230
	Answers to activities	231
Appendix 1	Pin outs of digital ICs	234
Appendix 2	Sample data sheets	236
Appendix 3	Glossary	249
Appendix 4	Answers to self-evaluation exercises	251
Appendix 5	Recommended reading	274

Preface

This book covers the basics of digital systems. It is designed as a one-semester course for technikon students who are studying electronic engineering.

The language in the text is simple, conversational English. Difficult concepts and technical terms are explained throughout the book. There is also a glossary at the end of the book in which terminology is explained.

HOW TO USE THIS BOOK

This book has 9 units. Students should start with Unit 1 and systematically work through the book until they reach the end of Unit 9. Students should not move on to a next unit until they understand the unit they are busy with. Each unit starts with a list of study objectives. These objectives set out what the student should be able to do at the end of the unit.

The text is set out in such a way that students should be able to work through the book by themselves. New concepts are explained and reinforced by giving students examples with solutions to work through. There are also many figures used throughout the text to aid understanding and clarify concepts. The mathematics in this course is thereby made clear and understandable.

Because this is a problem-solving course, there are also many activities for the students to work through. These activities allow the students to make sure that they have understood the work they have just covered. The answers to the activities appear at the end of each unit.

The summary at the end of each unit enables students to see at a glance what they should have learnt in the unit. The summary is followed by a section with self-evaluation exercises to enable students to assess their own progress. Answers to the self-evaluation exercises appear in Appendix 4 at the end of the book.

We have used three icons in this book to help guide the students through the text. These are as follows.



This is an **ACTIVITY** icon. When you see this icon you will know that it is time to **DO** something! The activities are active and enjoyable and they help you to understand the subject. Feel free to do them with a friend or a group of friends. The solutions to the activities are given at the end of each unit.



This is a **DEFINITION** icon. Read the definitions carefully because the details are important.



This **NOTE WELL** icon appears alongside all the extremely important information.

Introductory concepts

Study objectives

After studying this unit, you should be able to:

- ◇ outline the topics covered in this book
- ◇ relate digital systems to electrical engineering
- ◇ determine the difference between analog circuits and digital circuits
- ◇ represent digital information
- ◇ differentiate between bits and bytes.

1 What is in this book?

Before we look at some of the elementary concepts of digital systems and at where digital systems falls in the vast field of electrical engineering, we will look briefly at the topics covered in this book. Remember that this book is aimed at giving you a fairly good working knowledge of the first level of a digital systems course. It will also give you practical experience with some simple digital circuits. Here is an outline of the topics covered.

Unit 1 Introductory concepts

This unit provides an overview of digital systems with reference to electrical engineering and an explanation of some elementary terms and concepts.

Unit 2 Basic logic function

This unit introduces the three simplest circuit elements in digital systems – the AND gate, the OR gate and the INVERTER.

Unit 3 Working with logic functions

The basic gates which were discussed in Unit 2 are linked together in various combinations to form larger digital circuits. You will learn how to write digital expressions and draw up tables for these circuits, and how to apply digital signals to these circuits.

Unit 4 The NAND and NOR logic gates

The next two logic gates in digital systems (NAND and NOR) are presented. You will also be shown how to draw complex digital circuits – based on the logic gates studied so far – from digital expressions.

Unit 5 Simplification using Boolean algebra

A guiding principle in engineering is to keep things simple. This also applies to digital expressions and circuits. In this unit you will learn how to simplify digital expressions and circuits using a special mathematical technique called **Boolean algebra**.

Unit 6 Simplification using Karnaugh maps

In this unit you will learn to use another technique to simplify digital expressions and circuits. This graphical method uses diagrams called Karnaugh maps.

Unit 7 Basic combinational logic circuits

Digital circuits can generally be classified into one of two types: combinational or sequential. In this unit you will learn techniques involved in the design of combinational digital circuits. At this point it will be important to introduce the concept of binary numbers. Two other digital gates – the Exclusive-OR and the Exclusive-NOR – will also be introduced. You will also learn how to design circuits to compare binary numbers.

Unit 8 Applications of combinational logic

In this unit you will learn various numbering systems as used in digital electronics. You will use these numbering systems to do addition using integrated circuits (ICs) called **adders**. You will use decoders to convert one coding system to another, and multiplexers to do data selection.

Unit 9 Sequential logic circuits

In this unit you will learn about logic devices that have two stable states. These devices can maintain their output state after the input signals which have produced them, have been removed and, as such, form the basis on digital logic memories. You will learn about two types of bi-stable devices namely the latch and the flip-flop, and how to apply the flip-flop to simple sequential logic circuits, like shift registers and counters.

Appendix 1 Pin outs of digital ICs

To construct and test or troubleshoot a digital circuit it is necessary to know what each pin on a digital component or integrated circuit (IC) represents. This appendix contains diagrams showing some popular ICs and what each pin on the ICs represents.

Appendix 2 Sample data sheets

In many cases when designing, testing or troubleshooting electronic circuits, it is necessary to have a lot more information than just the component pin outs. The manufacturer of the component supplies comprehensive information on the component in the form of a data sheet. This section contains a few sample sheets of some of the ICs discussed in this book.

Appendix 3 Glossary

Appendix 4 Answers to self-evaluation exercises

Appendix 5 Recommended reading

2 Overview

Welcome to the world of digital systems. You are probably aware that the field of electrical engineering involves dealing with electronic circuits. Almost all electrical engineering equipment is controlled by some type of electronic circuit. These electronic circuits are made up of various types of devices (or electronic parts) linked together to perform a certain task or function. These parts are called **components**.

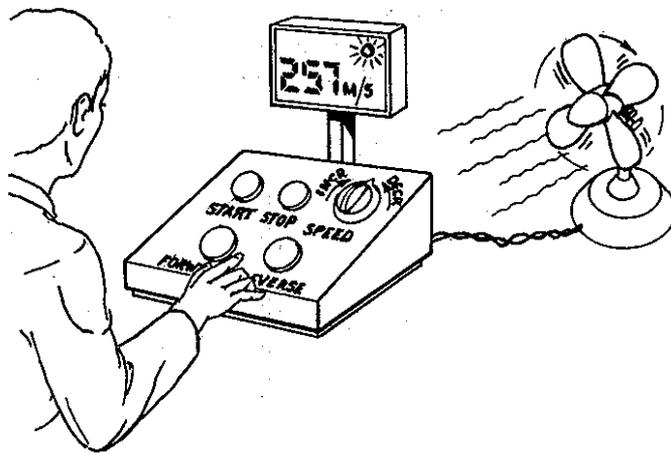


Fig 1.1 Diagram of a machine controlled by an electronic circuit.

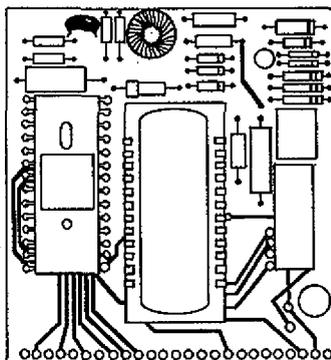


Fig 1.2 Diagram of an electronic circuit board.

Activity 1



Look around in your environment and make a list of items that have electronic circuits.

You will find the answers to the activities at the end of each unit. Use these to evaluate your answers. This will help you to monitor how well you have understood the work. By evaluating your progress, you can see which concepts you find difficult, and spend extra time revising them. Remember, you must evaluate your own progress.

3 Analog or digital?

Electronic circuits are usually divided into one of two types: **analog** or **digital**.

In this book we will be dealing only with digital devices and circuits. In other electrical engineering subjects you will learn more about analog systems and circuits. In practice there are systems which have both analog and digital circuits. These are referred to as **hybrid** systems.

ELECTRONIC CIRCUIT

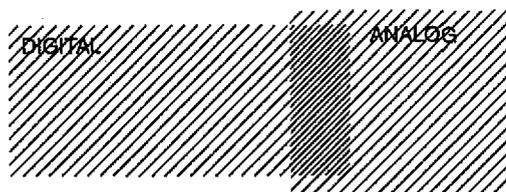


Fig 1.3

At the moment the trend is to use digital circuits, rather than analog, in most technical devices. The commonly-used PC (personal computer) has a large number of digital circuits.

You will often hear someone say that a certain item or device has been programmed to perform a certain task. For a device to be programmable it has to be designed with a great deal of digital circuitry.



To understand the difference between digital and analog circuits, we first have to understand the concept of electrical signals.

You have to keep two important points in mind:

- 1 Every electronic circuit needs electrical power to enable it to operate. For example, a portable radio needs batteries to supply the power.
- 2 Electronic signals may flow into a circuit, through the circuit and finally out of the circuit. For example, to make the radio sound louder, the volume knob must be turned. This sends a signal into the radio circuitry. The signal travels through the circuit and finally comes out as an output signal to the loudspeaker and we hear the sound louder.

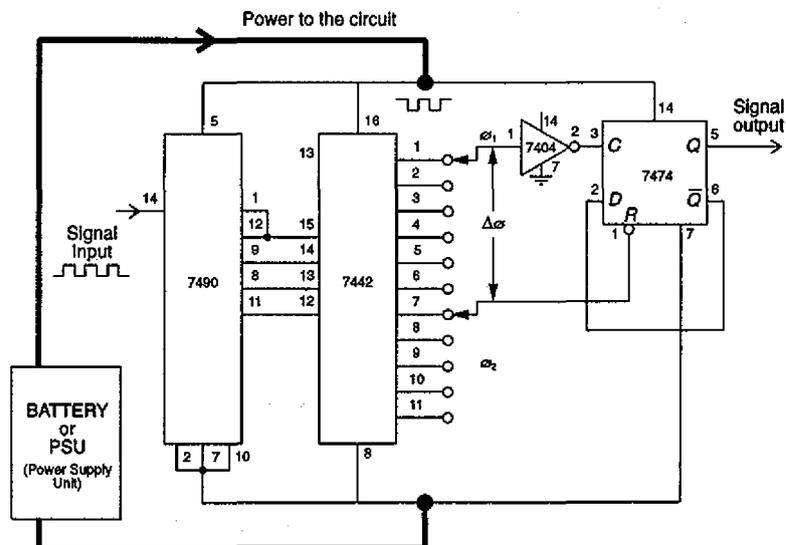


Fig 1.4

Electronic signals, sometimes referred to as **waveforms**, may generally be divided into two types: **analog** and **digital**.

Signals in an electronic circuit usually represent a voltage or a current. We will only look at the concept of voltage.

3.1 Analog signals

Let's first look at an analog voltage signal. The voltage in an analog voltage signal will have a certain value. As time goes by, the voltage may change (i.e. it may increase or decrease). Have a look at Fig 1.5. The line showing how the voltage changes with respect to time, is called a **voltage signal**. You will notice that the signal may take on any value (within limits, of course). Because there are no breaks in the signal, we say that the signal is continuous.

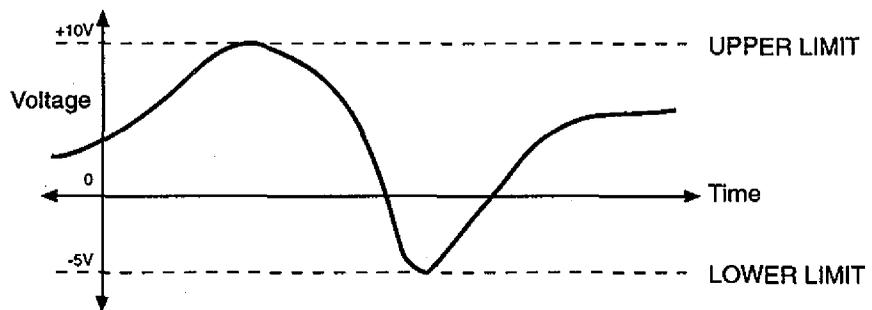


Fig 1.5 Analog voltage signal

3.2 Digital signals

Fig 1.6 shows a typical digital signal.

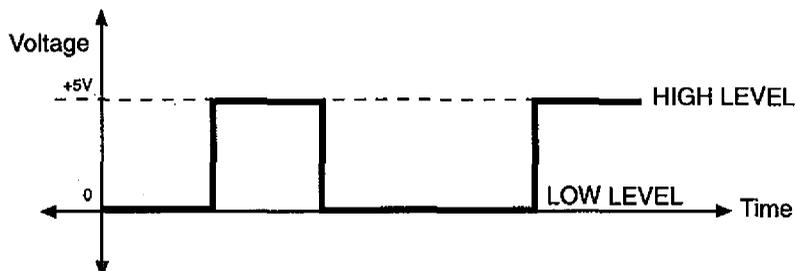


Fig 1.6 Digital voltage signal

Activity 2

What do you notice about the digital signal shown in Fig 1.6 when compared to the analog signal in Fig 1.5?

A digital signal can usually take only one of two values – either a high voltage or a low voltage. The signal remains at a specific level – either high or low – for a fixed time period. In the next fixed time period, the signal may change to a different specific level, again either high or low. This level is also called a **discrete level**, therefore we say that a digital signal is not continuous, but discrete.

Analog	Digital
continuous	discrete
varying level	fixed level

Table 1.1

Circuits that work with analog signals are called **analog circuits**, and circuits that work with digital signals are called **digital circuits**.

Devices that contain some sort of alphabetical or numerical display, for example, a digital wrist watch, usually have digital circuits.

Remember that many electronic systems may have both digital and analog circuits.

Activity 3

Make a list of devices which have digital circuits, analog circuits or a combination of both these circuits.

4 Representing digital information

When analog circuits are analysed or examined, the actual values of the voltage or current need to be specified, for example, 12V, 220V, 30A etc. However, in digital systems we are more interested in knowing whether the voltage signal is a high voltage or

a low voltage. For example, the actual high voltage may be 5V and the actual low voltage may be 0V. We will say that the signal is either **high** or **low**. Sometimes we will refer to the signal level as the **logic level** or **logic state**. Instead of using the terms high and low, the usual practice in digital systems is to use 1 and 0. These are read as 'one' and 'zero'. Fig 1.7 shows a simple digital signal and the various ways we use to specify it.

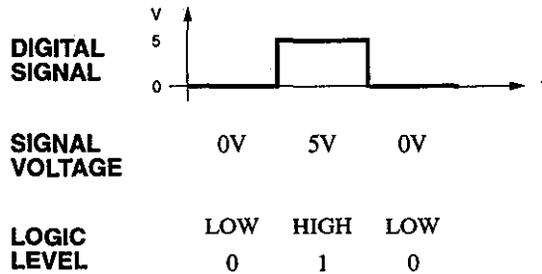


Fig 1.7

One of the main uses of digital circuits is to process and store information, for example, in computers. This information is often referred to as **data**. The data is actually in the form of digital signals. As mentioned above, the digital signals may be represented as zeros and ones. The data that is processed and stored by devices such as computers, can be examined and interpreted in the form of these numbers. A typical digital signal may then look like this: 0101000100101010011110.

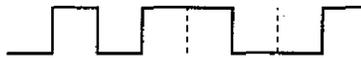
5 Bits and bytes

Each 0 or 1 of digital data is referred to as a bit of data, or simply as a **bit**. Bits are usually grouped together into sets of data. These sets are termed **bytes**. The standard practice is to refer to a group of 8 bits as a byte. Thus 16 bits of data make up 2 bytes, and so on.

In situations where bytes are used to represent numerical information, the position of the bit in the byte is very important. In Unit 7 you will learn about the concepts of the 'most significant bit' and the 'least significant bit'. These will be simply called the **MSB** and the **LSB**.

Activity 4

- 1 You may have heard people who use computers say that their computer has a 16MB memory or a 1 gigabyte hard drive. Find out what these terms mean.
- 2 Examine the data shown below and rewrite the data in the form of bytes: 101010000111101010101010111001
Is it easier to read the data when it is written as above or when it is written in the form of bytes?
- 3 Draw a digital waveform corresponding to the following bits: 10101100
- 4 Represent the following digital signal in the form of bits.

*Fig 1.8*

6 Summary

This unit began with an outline of the topics to be covered in this book. This unit served to introduce you to the subject of digital systems and showed where digital systems fit into the vast field of electrical engineering. Basic concepts on which the rest of this book is based were explained.

The next unit will begin with the concept of the three basic building blocks or components on which most of the circuits in digital systems are based.

Self-evaluation

Complete the following self-evaluation exercises **without** referring back to the unit.

- 1 Electronic circuits may be classified into two basic categories. Name them.
- 2 Many devices are made up of a combination of the two types named in number 1. True or false?
- 3 How would you differentiate between an analog and a digital signal?

- 4 Sketch an example of each of the following signals:
 - 4.1 a digital signal of which the high level is 12V and low level is 0V
 - 4.2 an analog signal which may vary between the limits of +5V and -5V.
- 5 How would you differentiate between a bit and a byte?
- 6 If a byte is made up of eight bits, how many bytes would 1024 bits represent?
- 7 A computer consists of 8MB of memory. How many bits is the memory made up of?
- 8 Represent the following in the form of bytes:
1010110010100011
- 9 Represent the following as a digital signal: 1000011010

You will find suggested answers to the self-evaluation exercises in Appendix 4 at the end of this book. Do not, however, consult these answers until you have completed all the exercises.

Answers to activities

Activity 1

Watch, clock, radio, television, computer, telephone, burglar alarm, camera, amplifier, etc.

Activity 2

In the digital signal:

- ◇ the level is only either high or low
- ◇ the signal is at a certain level for a certain minimum fixed time
- ◇ the signal is made up of distinct or discrete parts.

Activity 3

Without a knowledge of the actual signals involved in the circuitry, it is difficult to say exactly which circuits are analog or digital. As your knowledge of electronics, digital systems, and electrical engineering increases, you will be able to determine the extent of digital and analog circuitry in the various electronic devices.

Digital	Analog	Combination
digital watch	car speedometer	telephone
computer	radio	radio
	television	television
	camera	camera
	amplifier	amplifier

Activity 4

- 1 16MB memory: **M** stands for mega or 10^6 . **B** stands for bytes. The memory or data storage area consists of 16×10^6 bytes. (Remember that each byte is made up of 8 bits.)

1 gigabyte hard drive: Giga stands for 10^9 . The data storage capacity of the hard disk drive unit in the computer is 10^9 bytes.

- 2 10101000 01111010 10101010 10111001

It would seem easier to read the data when it is written in the form of bytes. If the bytes were coded into some simpler form it would be even easier to work with. You will learn more about this in Units 7 and 8.

3



Fig 1.9

- 4 01011001

Basic logic functions

Study objectives

After studying this unit, you should be able to:

- ◇ define the three basic logic functions, i.e. AND, OR and NOT
- ◇ identify the AND, OR and NOT logic symbols
- ◇ draw their truth tables
- ◇ write them as Boolean expressions
- ◇ draw output waveforms for the basic logic gates
- ◇ use data sheets to identify the component pin connections
- ◇ construct basic logic circuits
- ◇ test the logic circuits to verify their operation
- ◇ use the basic logic gates in simple applications
- ◇ apply digital signals to the basic logic gates
- ◇ identify basic logic gate fault conditions.

1 Introduction

In this unit you will be studying the basic logic functions, and the various ways in which these logic functions can be represented. You will then look at ways of practically testing the circuits that have been described.

2 What are the basic logic functions?

The three basic logic functions are the **AND**, the **OR** and the **NOT** functions. These will be explained in this unit, and you will also learn the various ways of representing these logic functions. Other logic functions, such as the **NAND**, the **NOR**, the **Exclusive-OR** and the **Exclusive-NOR**, which may be derived from the basic logic functions, will be discussed in later units.

3 The AND function

This logic function is probably the simplest to understand and use. Have a look at the example below, which illustrates the principle underlying the AND function.

Example 2.1 You want to go out with your friends, but to do so you need the permission of both your parents. Both your father and your mother must agree that you can go out. If either one or both do not give their permission, you cannot go out.

Let us summarise the situation in the form of a table.

Can you go out?

Father says	Mother says	So what do you do?
No	No	Stay at home
No	Yes	Stay at home
Yes	No	Stay at home
Yes	Yes	Go out, have fun!

This table represents the AND function, i.e. both your father **and** mother must say yes for you to be allowed to go out.

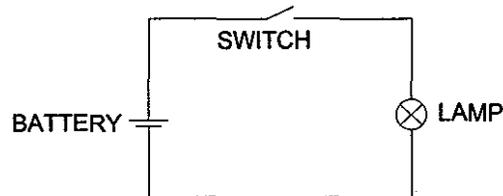


Fig 2.1

We can apply this principle to other situations. Have a look at the simple electrical circuit in Fig 2.1 above.

The lamp will turn **on** only when the switch is **closed**. When the switch is **open**, the lamp will remain **off**.

Now look at the electrical circuit in Fig 2.2.

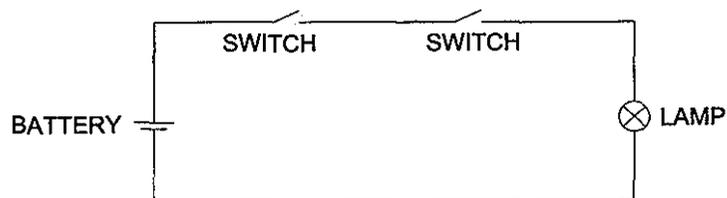


Fig 2.2

The lamp will turn **on** only when **both** switch A and switch B are **closed**. For all other cases the lamp will remain **off**. Now let's draw up a table to summarise how this circuit behaves.

Switch A	Switch B	Lamp
Open	Open	Off
Open	Closed	Off
Closed	Open	Off
Closed	Closed	On

Once again you see that the table represents the AND function, i.e. **both** switch A **and** switch B must be **closed** for the lamp to turn **on**.

As we explained in Unit 1, we usually represent information in digital systems in the form of a **low** and a **high** level, or by a **0** and a **1**. If we let the open switch be represented by a low level, and the closed switch by a high level; and the off lamp by a low level, and the on lamp by a high level, then the table can be rewritten as follows:

Switch A	Switch B	Lamp
Low	Low	Low
Low	High	Low
High	Low	Low
High	High	High

Alternatively, if we use 0 and 1 instead of low and high, we can rewrite the table in yet another way, as you can see below.



We call a table that shows all the possible cases of inputs and outputs for a digital circuit in terms of 0s and 1s a **truth table**.

Switch A	Switch B	Lamp
0	0	0
0	1	0
1	0	0
1	1	1



The **AND function** can be defined as a circuit that produces a 1 or a high output when all its inputs are at a 1 or high level.

In digital electronics there is a device that can perform the same logic functions as the AND function. This device is called the **AND gate**. It is usually represented by a special symbol called a **logic symbol**. The logic symbol for the AND gate is shown below.

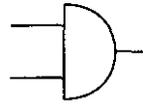


Fig 2.3

This AND gate has two inputs and one output. We can label these as follows:

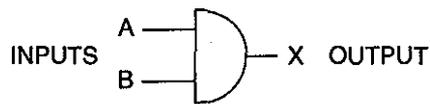


Fig 2.4

The truth table for the AND gate will be as follows:

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

The inputs are usually shown on the left-hand side of the table, with the outputs usually shown on the right-hand side.

In a digital circuit diagram the logic gate may be drawn in any direction.

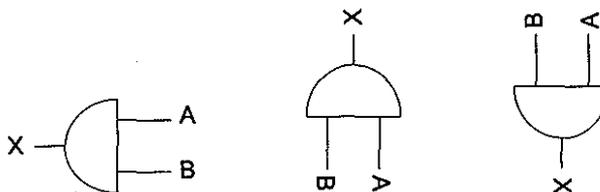


Fig 2.5

3.1 Representation of the AND function as a Boolean expression

Any logic function or circuit may be represented by a mathematical expression. We refer to the mathematics of logic circuits as **Boolean algebra**. The AND function can be written like this:

$$X = A \text{ AND } B$$

or more usually as: $X = A \cdot B$

Often the dot is left out and the expression becomes: $X = AB$.

The expression is still read as 'X equals A AND B'.

Activity 1



- 1 Define the basic AND function.
- 2 Define the term **truth table**.
- 3 Write down the Boolean expression and the truth table for the following device.

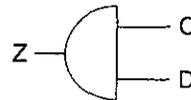


Fig 2.6

As you will remember, we discussed the concept of digital signals or digital waveforms in Unit 1.

Logic gates may process digital signals according to their particular logic function.

Take a look at the example below.

- Example 2.2** Determine the output signal, X, if the signals shown are fed into the inputs A and B.

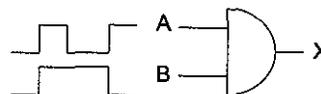


Fig 2.7

Solution

- 1 In the beginning both A and B are low or 0. According to the AND gate truth table in this case, the output must be low or 0. Draw the output as a low level under the corresponding input signal part.

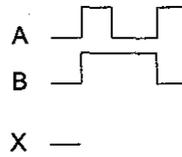


Fig 2.8

- 2 Next, A is at a high level and B is at a high level. According to the truth table, the output must be high. Draw a high level for signal X.

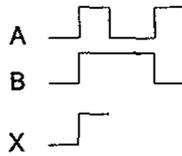


Fig 2.9

- 3 The rest of the signal can be determined in the same way.

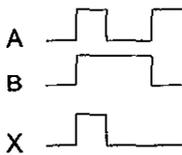


Figure 2.10

Activity 2



- 1 For the AND gate shown in Fig 2.7, determine the output signal X if the inputs A and B were fed with the following signals.

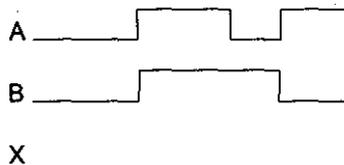


Fig 2.11

- 2 Determine the output signal or waveform X from the AND gate shown in Fig 2.7 given the signals for A and B as shown below.

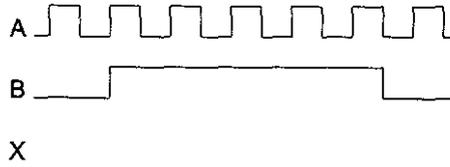


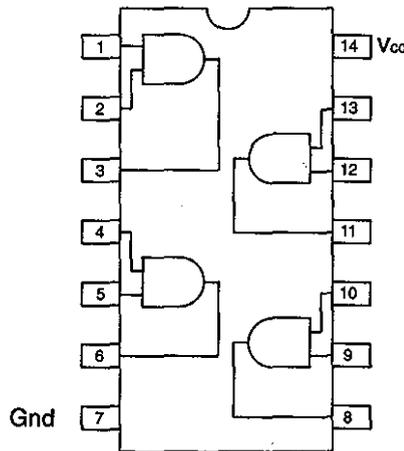
Fig 2.12

3.2 Construction and testing of the AND gate

To verify the operation of logic gates the circuit needs to be constructed with the actual digital components and tested with real electronic signals. The detailed description and usage of electronic (including digital) components are found in manufacturer's catalogues, called data books.

You will find sample pages taken from some digital component data books in Appendix 2 at the end of this book. Data books consist of a number of data sheets for different components, for example, logic gates and other digital devices.

The following diagram shows the internal arrangement and the pin connections for a digital component which contains four AND gates – each gate in this device has two inputs.



7408 quad 2-input
AND gate

Fig 2.13

The common term for most digital (and some analog) components connected as a circuit, is an **integrated circuit**, or **IC** for short. The above component can be called a **quad 2-input AND gate IC**. The word 'quad' tells you that there are four gates in the IC.

Two of the pins are usually connected to the **power supply** or battery to supply power to the IC. In this case these are pins 7 and 14.

Pins 1 and 2 are the two **inputs** of one of the AND gates and pin 3 is the **output**.

Activity 3



Before the actual AND gate IC can be tested, the circuit must first be constructed on a test board.

The steps listed below are very important. Make sure that you follow them carefully, as you will need to test more gates later in this unit.

Items required

- ◇ 1 quad 2-input AND gate IC (for example, 7408)
- ◇ 1 test board
- ◇ some pieces of connecting wire
- ◇ 1 LED and 1 K Ω resistor.

Method

Always double check the circuit wiring before switching the power on.

- 1 Place the IC numbered 7408 on the test board.
- 2 Connect the output of a gate (pin 3) to a lamp (LED).
- 3 Connect each of the inputs (pins 1 and 2) of the same gate to the 0V line.
- 4 Connect the positive power supply pin (Vcc) (pin 14) to the +5V line.
- 5 Connect the other power supply pin (Gnd) (pin 7) to the 0V line. **(NB Connecting a pin to the 0V line is called 'grounding' the pin.)**

Precaution: Never let a +5V line touch a 0V line!

- 6 Switch the power to the circuit ON.
Does the lamp remain off or turn on? Record the details in a table.



- 7 Switch the power to the circuit off.
Disconnect the 0V line from one of the inputs. Now connect this input to the +5V line. Does the lamp remain off or turn on? Continue to record the details in the table.
- 8 Switch the power to the circuit OFF.
Connect both the input pins to the +5V line. Does the lamp remain off or turn on? Continue to record the details in the table.
- 9 Finally, try the last combination of input switch conditions and record the results.
- 10 Compare the table obtained with that of the AND gate truth table.
- 11 If the theoretical truth table and the truth table you obtained experimentally do not correspond, you will have to troubleshoot the circuit.



Precaution: When removing the IC from the test board on completing the activity, be careful not to bend or break the IC pins.

Example 2.3

A printer will print only if both the printer cover is closed and the paper tray is full. A high voltage level is generated when the printer cover is closed. The paper tray also generates a high level when it is full. To start printing, the printer needs a high level. Determine what type of logic gate can be used to make the printer start printing.

Solution

At this stage we have only studied the AND gate. Will it work in the above application? Before deciding, let us draw a table of all the possibilities.

Printer cover	Paper tray	Printer
Open	Empty	Off
Open	Full	Off
Closed	Empty	Off
Closed	Full	On

If the printer cover is closed, it generates a high level. Therefore if the cover is open it will generate a low level. Also, if the paper tray is full, it generates a high level. Therefore if the tray is empty, it will generate a low level.

We have already seen that a high can be represented by a 1 and a low by a 0. If we redraw the above table with 0s and 1s, we get the following table:

Printer cover	Paper tray	Printer
0	0	0
0	1	0
1	0	0
1	1	1

Conclusion

This table is the same as that for the AND gate. Thus an AND gate would be suitable for this application. One input for the AND gate would come from the printer cover and the other input from the paper tray. The output would be used to start the printer.

We have now covered the basic theoretical and practical aspects of the AND function. Our next task is to study the OR function.

4 The OR function

This function can easily be demonstrated by using another switch-lamp circuit. Examine the parallel circuit shown below.

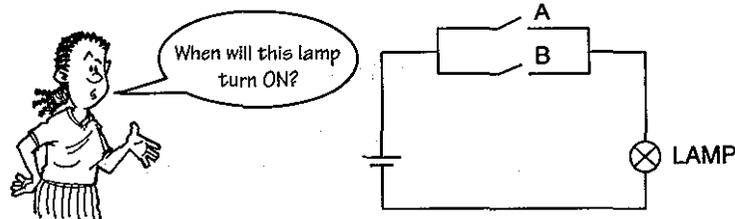


Fig 2.14

The lamp will turn on only when:

- ◇ switch A is closed **OR**
- ◇ switch B is closed **OR**
- ◇ both switches A and B are closed.

However, if **both** the switches are open, then the lamp will remain off.

Now let us summarise the above information in the form of a table.

Switch A	Switch B	Lamp
Open	Open	Off
Open	Closed	On
Closed	Open	On
Closed	Closed	On

Rewriting the table as a **truth table** using 0s and 1s, we get the following:

Switch A	Switch B	Lamp
0	0	0
0	1	1
1	0	1
1	1	1

This table defines the **OR** function.



The **OR function** can thus be defined as a circuit which produces a high output when any of its inputs are at a high level.

4.1 Representation of the OR function as a Boolean expression

The digital device that performs the OR function is called an **OR gate**. The logic symbol for the OR gate is:



Fig 2.15

In Boolean algebra the function for the OR gate is written as follows: $X = A \text{ OR } B$

or more usually as: $X = A + B$

The expression is still read as 'X equals A OR B'.

This OR sign (+) should **not** be confused with the normal arithmetic addition or 'plus' sign.

Activity 4



- 1 Write down the truth table and draw the logic symbol for the following expression: $K = S + T$
- 2 For the above logic gate, if the input S is kept low and the input T is fed with the following digital waveform, what will the output signal X look like?

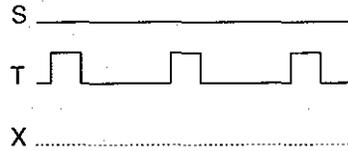


Fig 2.16

- 3 Determine the output waveform X for the same logic gate if the inputs are as shown below:

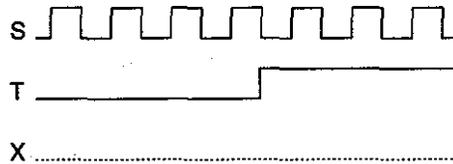
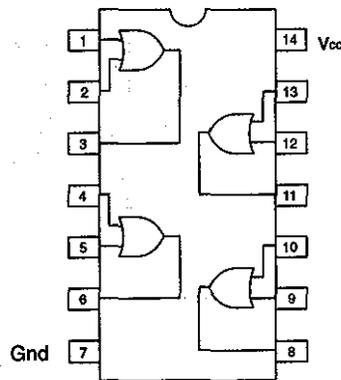


Fig 2.17

- 4 The following diagram shows the internal arrangement and the pin connections for a digital component which contains four OR gates – each gate in this device has two inputs.



7432 quad 2-input OR gate

Fig 2.18

- 4.1 What are the **functions** of pins numbered 7, 8, 12 and 14?
 - 4.2 The above component represents a quad 2-input OR gate IC. True or false? Give a reason for your answer.
-

4.2 Construction and testing of the OR gate

To verify the operation of an OR gate, the circuit needs to be constructed with the actual components and tested with real electronic signals as explained in Activity 3. The IC (No. 7432), which is shown in Fig 2.18 above, must be tested by following the same procedures as in the case of the AND gate in Activity 3.

Let's do our own construction and testing.

Activity 5



Items required

- ◇ 1 quad 2-input OR gate IC (for example, 7432)
- ◇ 1 test board
- ◇ some pieces of connecting wire
- ◇ 1 LED and 1 K Ω resistor.

Method

- 1 Wire up the circuit to test the first OR gate found in the IC 7432. The procedure should be similar to that described in Activity 3. The table obtained should also be similar to the truth table for the 2-input AND gate.
- 2 If the table differs considerably from the truth table for the 2-input OR gate, check the construction and testing procedure carefully by following the steps of Activity 3.

Application

In a factory a sensor is used to check the temperature of a liquid. If the temperature is above 40 degrees celsius an alarm must be sounded. The alarm must also be sounded if the tank containing the liquid is empty. What type of logic gate could we use to sound the alarm?

Let's first consider all the possibilities in the form of a table.

Temperature	Tank	Alarm
below 40 degrees	full	Off
below 40 degrees	empty	On
above 40 degrees	full	On
above 40 degrees	empty	On

We could represent the temperature below 40 degrees by a 0 and that above 40 degrees by a 1. Also, the full tank could be represented by a 0 and the empty tank by a 1. We could therefore rewrite the table of possibilities as:

Temperature	Tank	Alarm
0	0	0
0	1	1
1	0	1
1	1	1

This table looks exactly like the truth table for an OR gate. This shows that for this application an OR gate would be suitable.

We have now covered the basic theoretical and practical aspects of the OR function. Our next task is to study the last common logic function that we will be dealing with in this unit: the NOT function.

5 The NOT function (INVERTER)

This is the last of the three basic logic functions that we will be studying in this unit.



The **NOT function** simply produces a result that is always the **opposite** of the given or input condition.

If the **input** is high, the **output** is low – in other words, if the input is a 0, the output is a 1.

You will notice that this function has only **one** input and **one** output. A digital device that performs this function is called a **NOT gate**, or alternatively, an **INVERTER**.

5.1 Representation of the NOT function as a Boolean expression

The logic symbol for the NOT gate or INVERTER is:

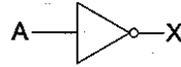


Fig 2.19

and the Boolean expression can be written as follows:

$$X = \text{NOT } A$$

$$\text{or more usually as: } X = \bar{A}$$

This is still read as 'X equals NOT A' (or sometimes as 'X equals A bar').

The **truth table** of the NOT gate will simply be:

A	X
0	1
1	0

Activity 6



- 1 Write down the Boolean expression and truth table for the following device in terms of highs and lows.

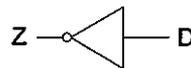


Fig 2.20

- 2 If two INVERTERS are connected as shown below, what will the output of the second gate be if the input of the first gate is a 0?

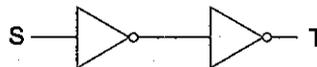
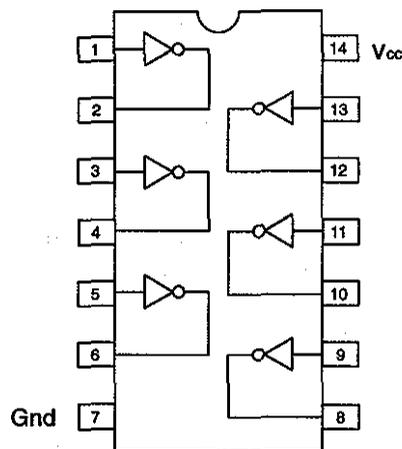


Fig 2.21

5.2 Construction and testing of the NOT gate

To verify the operation of an INVERTER (NOT gate), the IC (No. 7404) shown below may be used. This IC is termed a **hex INVERTER**. The circuit needs to be constructed with the actual components and tested with real electronic signals as explained in Activity 3.

Why is it a **hex** INVERTER and not a **quad** INVERTER? Easy! Hex means six, and we use six INVERTERS to construct this particular IC.



7404 hex INVERTER

Fig 2.22

Now let's do our own construction and testing.

Activity 7



Construction and testing

Construct the INVERTER test circuit in a similar way to the way you constructed the AND and OR gate circuits in Activities 3 and 5. However, remember that NOT gates have only **one** input each. After verifying that the wiring is correct, test the device as you did in Activity 3 and draw the NOT truth table.

6 Summary

This unit introduced you to the theory of the **three basic logic functions**, namely the **AND**, **OR** and **NOT** functions. Digital devices or gates to implement these functions were described, and you learnt that the logic gates are represented in three ways, namely as **logic symbols**, in **truth tables** and as **Boolean expressions**. You also learnt how the logic gates respond to input digital signals or waveforms to produce output waveforms and you used data sheets to identify the component pin connections. You constructed practical circuits with these logic gates and tested them with actual voltages. In addition, you were introduced to some applications for logic gates.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Determine the output of the gate shown below, given the inputs as follows.

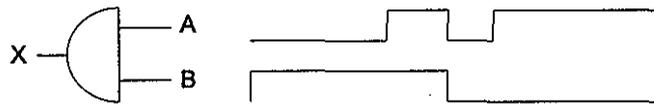


Fig 2.23

- 2 Draw the output waveform given the inputs shown below for a 2-input OR gate.

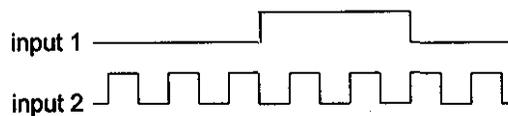


Fig 2.24

- 3 Determine the output of the following circuit if the input is low.

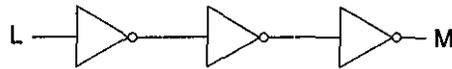


Fig 2.25

- 4 An automatic gate needs to be opened by any one of two keys. Each key is kept by a different security guard. What logic gate must be connected to the two keys to operate the door lock? Explain why you made this choice.
- 5 For an OR gate wired up for test purposes, the output pin was connected to the lamp (LED) as shown in Fig 2.26. When one of the inputs was made high the lamp remained ON no matter what we did to the other input. Is the OR gate functioning correctly? Explain.

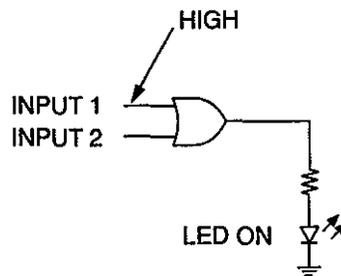


Fig 2.26

- 6 A certain basic 2-input logic gate was wired up for test purposes in a circuit similar to the one in exercise 5 above. When one of the inputs was grounded, the lamp remained off irrespective of what we did to the other input. Would the gate that you would be testing be an AND gate, an OR gate or an INVERTER (assuming that the gate was functioning correctly)? Explain.

Answers to activities

Activity 1

- 1 An AND function can be defined as a circuit that produces a 1 or a high output when all of its inputs are at a 1 or high level.
- 2 A truth table is a table that shows all the possible cases of inputs and outputs for a digital circuit in terms of 0s and 1s.
- 3 $Z = CD$

Activity 2

1

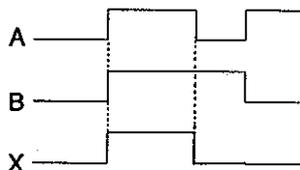


Fig 2.27

2

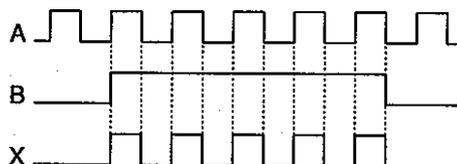


Fig 2.28

Activity 4

1 Truth table:

S	T	K
0	0	0
0	1	1
1	0	1
1	1	1

Logic symbol:

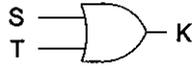


Fig 2.29

2

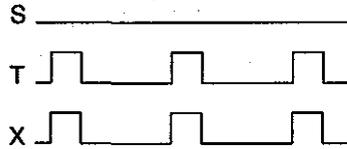


Fig 2.30

3

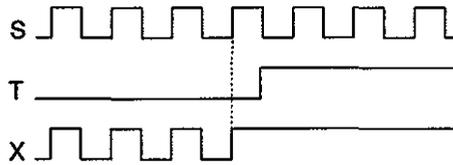


Fig 2.31

4.1

Pin no.	Function
7	0 volt power supply pin
8	output of an OR gate
12	input of another OR gate
14	+5 volt power supply pin

4.2 True. Quad means 'four'.

Activity 6

- 1 Make sure you identify which are inputs and which are outputs. In this example D is the input and Z the output.

D	Z
low	high
high	low

- 2 The output of the second gate will be 0, but the first **INVERTER** changes the 0 into a 1. Then the next **INVERTER** changes the 1 to a 0.

Working with logic functions

Study objectives

After studying this unit, you should be able to:

- ◇ determine how many possible cases a truth table must show
- ◇ connect logic gates to perform various tasks
- ◇ derive the Boolean expressions for these functions
- ◇ derive the truth tables for these functions
- ◇ apply digital signals to these logic circuits
- ◇ construct and test these logic circuits
- ◇ identify logic circuit fault conditions.

1 Introduction

Unit 2 introduced the three basic logic functions and the three simplest logic gates, namely the AND gate, the OR gate and the NOT or INVERTER. You learnt various ways of representing these logic functions: by means of logic symbols, truth tables and Boolean expressions, and analysed the response of these logic gates to digital inputs.

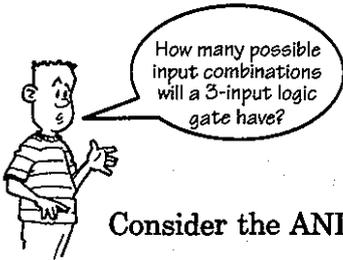
In this unit we examine how these basic logic gates can be combined to form more complex functions and logic circuits. We will determine the truth table and Boolean expression for each circuit, as well as the response of these logic circuits to digital signals. You will be guided through building and testing the corresponding practical digital circuits. Towards the end of the unit you will be introduced to troubleshooting faulty logic circuits.

However, the most logical way to start this unit is by discussing how to generate the truth table for circuits or gates that have more than two inputs.

2 How many rows in a truth table?

In Unit 2, when we drew up the truth table for a 2-input AND gate, the truth table showed **four** possible input cases, namely: 00, 01, 10 and 11. The truth table had **four rows**. For the sin-

gle input INVERTER or NOT gate, the truth table showed two possibilities for the input. In this case the truth table had two rows.



Consider the AND gate shown below.

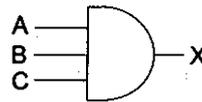
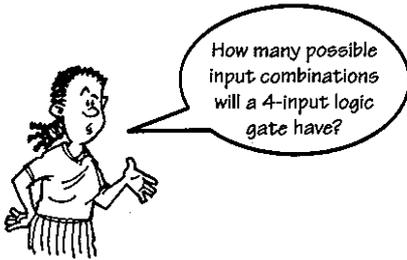


Fig 3.1

A, B and C can assume the values 000, 001, and so on. So if you said that there were **eight** possible cases, you were right! The full list of input combinations is shown below.



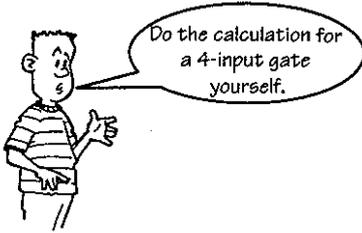
A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

A quick and easy way to determine how many possible combinations there are for a logic gate is to use the following formula:

$$\text{number of possible combinations} = 2^{\text{number of inputs}}$$

Let's apply the formula for a 1-input gate and a 2-input gate, which we learnt about in Unit 2, as well as a 3-input and 4-input gate.

$$\begin{aligned} \text{For a 1-input gate: number of possible combinations} &= 2^1 \\ &= 2 \end{aligned}$$



For a **2-input gate**:

$$\begin{aligned} \text{number of possible combinations} &= 2^2 \\ &= 4 \end{aligned}$$

For a **3-input gate**:

$$\begin{aligned} \text{number of possible combinations} &= 2^3 \\ &= 8 \end{aligned}$$

For a **4-input gate**:

$$\begin{aligned} \text{number of possible combinations} &= \dots \\ &= \dots \end{aligned}$$

3 What happens when logic gates are combined?

The calculations we have just done show that for any logic gate, the number of possible input combinations depends on the number of inputs. In fact, the same is true for any simple combinational logic circuit. (We will discuss combinational logic circuits later in this book.)

We now know that the operation of a logic circuit that has a total of **three inputs** can be represented by a truth table showing **eight possibilities** or rows.

Have a look at the following logic circuit:

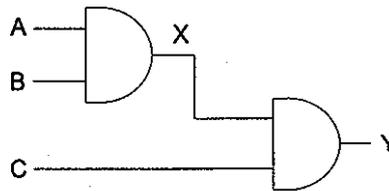
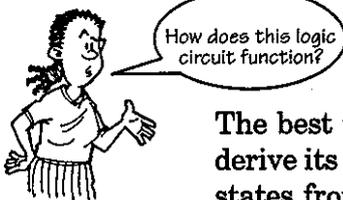


Fig 3.2

The output X of the first 2-input AND gate is connected to the input of the second 2-input AND gate, i.e. the signal that comes out of the first gate is fed into the second gate.



The best way of finding out how this logic circuit functions is to derive its truth table. The table will have eight rows showing logic states from 000 for A, B and C to 111. We will first write down the value for X for each combination, and then write down the value for Y.

Step 1

Write the values for **column X**.

To do this we have to consider the first logic gate. Using the values of A and B as inputs to an AND gate, we can determine the value for X for each row in the table.

For an AND gate, the output will only be a 1 if all the inputs are 1.

A	B	C	X	Y
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	
1	1	1	1	



Step 2

Write the values for **column Y**.

To do this we have to consider the second gate. This gate has two inputs, X and C. To determine Y we therefore have to look at X and C as two inputs into an AND gate.

A	B	C	X	Y
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1



Activity 1

- 1.1 Show how you would draw a 2-input, a 3-input, and a 4-input truth table. Indicate how the 0s and 1s are arranged in the columns.
- 1.2 Determine the truth table for the following circuit, which consists of two OR gates. Remember, an OR gate will have an output of 1 if any of the inputs is a 1.

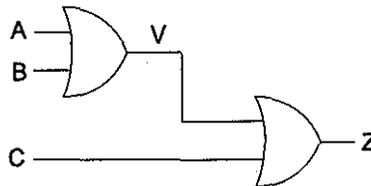


Fig 3.3

4 Writing the Boolean expression for a logic circuit

In Unit 2 we saw how a logic gate can be represented as a Boolean expression. You will also remember that **all** combinational logic circuits can be represented as Boolean expressions. The final Boolean expression is written by combining the individual Boolean expressions for the gates in the proper form.

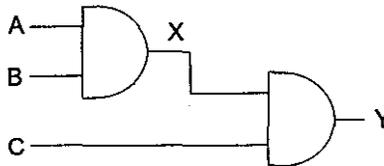


Fig 3.4

Look at the same AND gate circuit we studied earlier in this unit. To write down the final Boolean expression for this circuit we first have to write down the individual expressions starting at the input side. At the output of the **first** AND gate, the Boolean expression is:

$$X = AB \quad \text{[equation 1]}$$

The output of the **second** gate is:

$$Y = XC \quad \text{[equation 2]}$$

Combining (1) and (2), we arrive at: $Y = (AB)C$

Activity 2



Prove that the Boolean expression for the logic circuit shown in Fig 3.5 is $Z = (A + B) + C$.

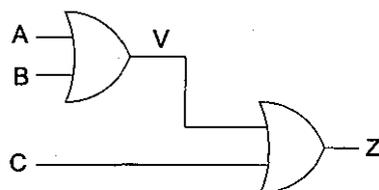


Fig 3.5

5 Considering more complex circuits

Let's determine the truth table and Boolean expression for the following logic circuit consisting of an AND gate, an OR gate and an INVERTER.

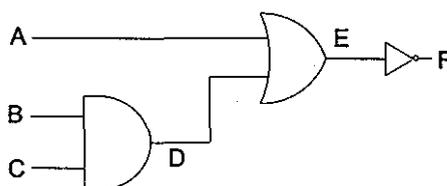


Fig 3.6

Make sure that you know the truth table for each of the gates in Fig 3.6 (see Unit 2).

- 1 The truth table will have **eight rows**, as the logic circuit has **three inputs**.
- 2 The input variables are A, B and C. The **intermediate** outputs are D and E. F is the final output.

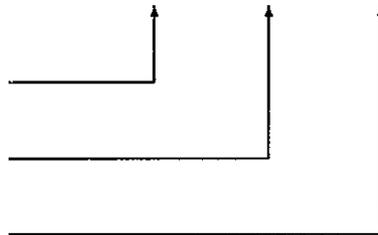
3 The truth table will therefore begin like this:

A	B	C	D	E	F
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

First determine column D using columns B and C.

Now determine column E using columns A and D.

Then determine column F using column E.



4 The truth table should then become:

A	B	C	D	E	F
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	0

It is not necessary to show the intermediate columns in the final answer.

Therefore the final truth table will be:

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

To determine the Boolean expression for the logic circuit in Fig 3.6, the procedure is the same as in section 4 of this unit. The circuit has been redrawn below.

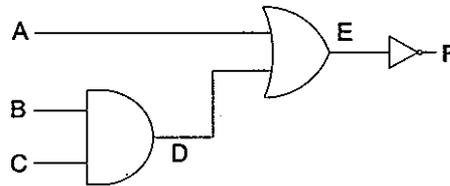


Fig 3.7

- From the AND gate: $D = BC$ [equation 1]
- From the OR gate: $E = A + D$ [equation 2]
- Using equation 1 and 2: $E = A + (BC)$ [equation 3]
- From the NOT gate: $F = \overline{E}$ [equation 4]
- Combining equation 3 and 4: $F = \overline{(A + (BC))}$

Activity 3



- 1 Determine the truth table for the following logic circuit consisting of a NOT gate, an OR gate and an AND gate.

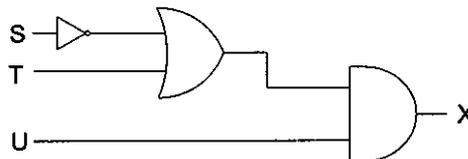


Fig 3.8

- 2 Write down the Boolean expression for the circuit in Fig 3.8 above.

6 Applying digital signals to these logic circuits

In Unit 2, we applied digital signals or waveforms to the **inputs** of a logic gate and determined the **output** waveform. To generate the output waveform, you had to know how that particular logic gate behaved, in other words you had to know the **truth table** for that specific gate. Digital signals may also be applied to the types of logic circuits described in this unit. To determine the final output waveform, you first have to work out the **intermediate** waveforms.

Again, you have to know the truth tables of the logic gates that make up the logic circuit.

Let's apply the following digital waveforms to the logic circuit given below.

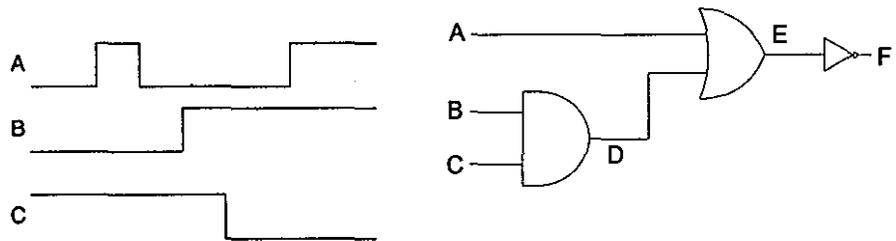


Fig 3.9

Step 1

Determine the **intermediate output** waveform for the gate to which the inputs are directly applied, i.e. the AND gate. Signals B and C are applied to this gate. The intermediate output is already labelled D. So, to determine D, the AND gate truth table will be used.

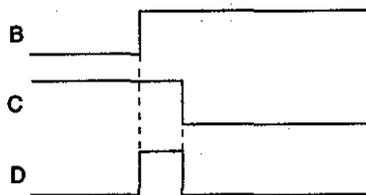


Fig 3.10

Step 2

Signals A and D are applied to the OR gate. The output is labelled E. To determine E, the OR gate truth table is used.

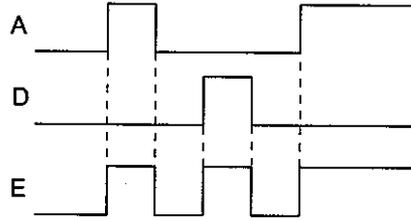


Fig 3.11

Step 3

The last step involves applying the E waveform to the INVERTER to generate the final output waveform F. In this case the INVERTER truth table is used.

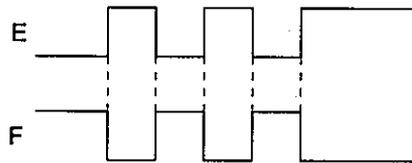


Fig 3.12

Activity 4



Apply the waveforms to the logic circuit in Fig 3.13 and determine the output waveform X. (Number the outputs as you did for Fig 3.8 in Activity 3.)

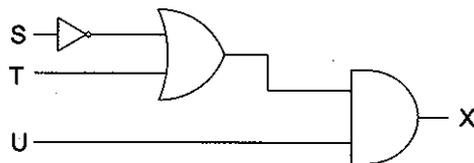
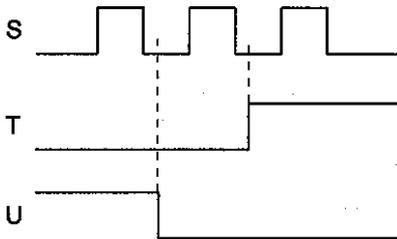
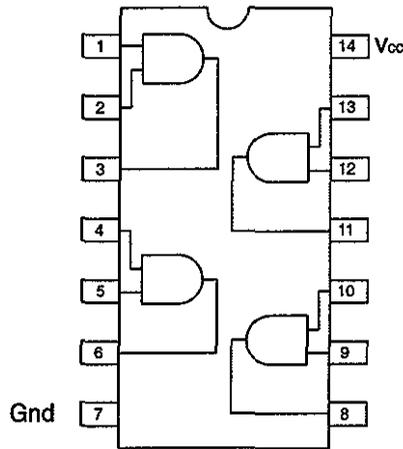


Fig 3.13

7 Practical implementation

Consider the circuit shown in Fig 3.14b. You will notice that the circuit is made up of two similar gates, i.e. two 2-input AND gates.

In Unit 2 the IC numbered **7408** (see Fig 2.13), which contained four 2-input AND gates, was used. We can use the same IC to test this circuit. The first step is to construct the logic circuit on the test board. Let's use the first two AND gates in the IC.



7408 quad 2-input
AND gate

Fig 3.14a

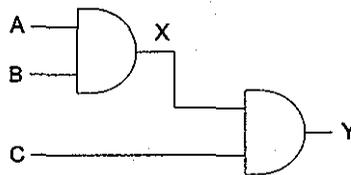


Fig 3.14b

Figure 3.15 on page 44 shows the practical logic circuit diagram as implemented for testing. Keep this circuit in mind and follow the next steps carefully to complete the correct testing procedures.

Insert the IC numbered 7408 into the test board. Connect the pins labelled Vcc and Gnd to the +5 V and 0V lines respectively, as we did in Unit 2. The **output** pin of the **first** AND gate (pin 3) must be connected with a piece of connecting wire to one of the inputs of the **second** AND gate (pin 4).

Connect the **output** pin of the **second** gate to the test (LED) lamp. Then test the circuit by connecting the three inputs of the logic circuit (pins 1, 2 and 5) to the +5 V line and the 0 V line according to the truth table. Record the results of the testing in a truth table.

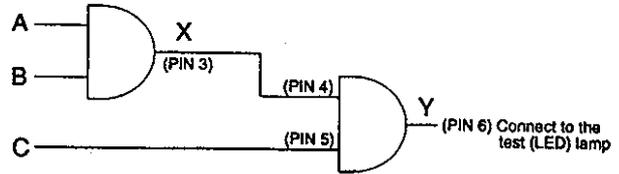


Fig 3.15

Now, let's see if you can do the following activity on your own!

Activity 5

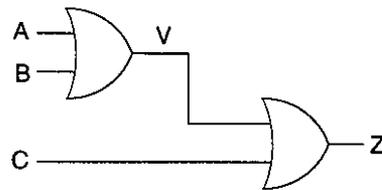


Fig 3.16

- 1 Construct and test the logic circuit shown in Fig 3.16 in the same way as we did with the one outlined in section 7 above.

Which IC is to be used, and why? Don't forget to re-draw the practical logic circuit diagram showing the pin numbers.

If you find it difficult to decide which IC to use, consult Appendix 1 on ICs at the end of this book.

- 2 Consider the logic circuit shown in Fig 3.17:

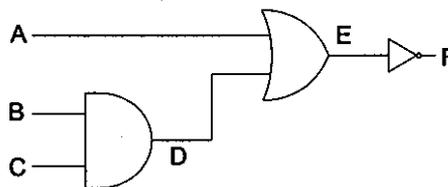


Fig 3.17

You will notice that in this circuit **three** different logic gates are used, i.e. an INVERTER, a 2-input AND gate and, a 2-input OR gate.

Which ICs should we use to test this circuit? Draw the practical logic circuit diagram and test the circuit shown in Fig 3.17.

8 Troubleshooting logic circuits

In this section we will learn how to check faulty logic circuits. We say that a logic circuit is faulty if it does not behave according to the truth table.

We use the term 'troubleshooting' to describe the process of identifying exactly which components are not functioning properly in any electronic circuit.



Troubleshooting involves the following:

- ◇ Checking **each** logic gate from the **input side**.
- ◇ Applying all the possible logic levels to each gate from the input side in an orderly manner until the gate that is not functioning correctly is found.
- ◇ Taking the faulty gate **out** of the circuit and testing it on its **own** to make certain that it is in fact the faulty component.
- ◇ **Replacing** the faulty gate with another similar gate and re-checking the circuit by repeating the troubleshooting techniques.

In some cases the gate suspected of being faulty is in fact not faulty at all. The fault may be caused by another gate to which it is connected.

Troubleshooting involves careful consideration and testing of circuits and gates. Although most faulty gates are easily detected, you will find that you have less and less trouble solving relatively difficult cases as you test more circuits and gain more practical experience.

In a later unit we will look at more examples of troubleshooting.

9 Summary

In this unit we looked at how the basic logic gates (the AND and OR gates and the INVERTER) could be **combined** to form various types of **logic circuits**. The logic circuits were represented as **Boolean expressions** and their behaviour or functioning was expressed in the form of **truth tables**. A truth table is usually drawn to show all the possible values that the inputs can assume, and the resulting outputs. In this unit we described how to draw a truth table for any logic circuit showing all the possible input conditions. We also applied **digital signals** or **waveforms** to the inputs of the logic circuits and determined the response of the circuit – in the form of an **output waveform** – with the aid of truth tables. The logic circuits were practically implemented with the aid of data sheets. This involved **constructing** and **testing** the logic circuits to verify their operation according to the theory. Finally, we discussed the important task of analysing faulty practical logic circuits and described the process of **troubleshooting** or identifying the faulty components.

In the next unit you will be introduced to two more logic gates, namely the NAND gate and the NOR gate.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Examine the logic circuit shown in Figure 3.18.

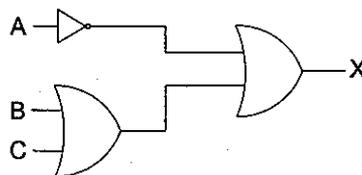


Fig 3.18

- 1.1 If C is high, determine the level at the output X.
- 1.2 If we draw a truth table for this logic circuit, how many rows will it have?
- 1.3 Draw a truth table for this logic circuit.
- 1.4 Write down the Boolean expression for this circuit.

- 1.5 If the following waveforms were applied to the inputs, determine the output signal.

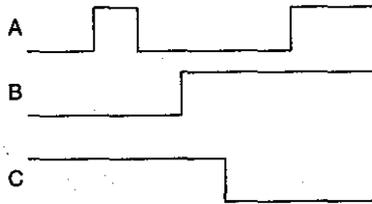


Fig 3.19

- 1.6 Which ICs must be used to test this circuit?
 2 Consider the logic gate shown in Fig 3.20.

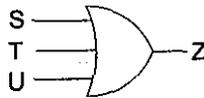


Fig 3.20

- 2.1 Identify the gate and write down the Boolean expression for this logic gate.
 2.2 Draw the truth table for this logic gate.
 2.3 Determine the output waveform if the following signals were applied to the inputs.

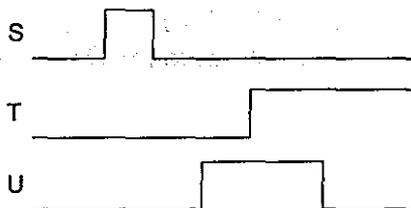


Fig 3.21

3 Consider the logic circuit shown in Fig 3.22.

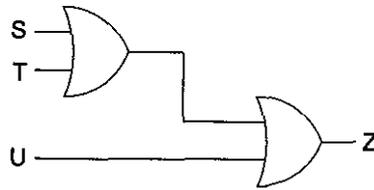


Fig 3.22

- 3.1 Write down the Boolean expression for this logic circuit.
- 3.2 Draw the truth table for this logic circuit.
- 3.3 Are the circuits shown in Fig 3.20 and 3.22 equivalent, i.e. do they have similar truth tables? Discuss.
- 4 Examine the circuits shown in Fig 3.1 and 3.2. Do they both behave in the same way? Explain your answer.
- 5 Consider the circuit shown in Fig 3.23.

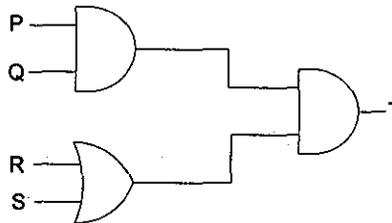


Fig 3.23

- 5.1 For the circuit shown in Fig 3.23, if a 0 is applied to input Q, can we determine the level of the output T? Explain your answer.
- 5.2 Write down the Boolean expression and draw the truth table for this circuit.

Answers to activities

Activity 1

1.1 2-input truth table

A	B	X
0	0	
0	1	
1	0	
1	1	

two 0s and two 1s alternating 0s and 1s

3-input truth table

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	

four 0s and four 1s two 0s and two 1s alternating 0s and 1s

4-input truth table

A	B	C	D	X
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

eight 0s and eight 1s alternating 0s and 1s
 four 0s and four 1s two 0s and two 1s

1.2 The truth table will have eight rows, as the circuit has three inputs. First obtain the outputs for column V. Then determine the outputs for column Z.

A	B	C	V	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Activity 2

From the first OR gate: $V = A + B$ [equation 1]

From the next OR gate: $Z = V + C$ [equation 2]

Combining equations 1 and 2: $Z = (A + B) + C$ [equation 3]

Activity 3

- 1 You will notice that the outputs from each of the logic gates have not been labelled. For the sake of convenience, we will label them as follows:

From the NOT: V

From the OR gate: W

Column V must be derived from column S. Then column W must be determined, using columns V and T. Finally column X is determined from columns W and U.

S	T	U	V	W	X
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	0	1	1

2 From the NOT gate: $V = \bar{S}$ [equation 1]

From the OR gate: $W = V + T$ [equation 2]

Using equations 1 and 2: $W = \bar{S} + T$ [equation 3]

From the AND gate: $X = WU$ [equation 4]

Using equations 3 and 4: $X = (\bar{S} + T)U$ [equation 5]

Activity 4

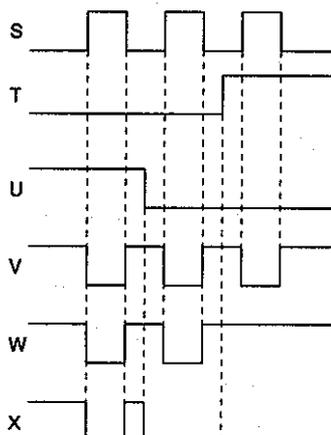
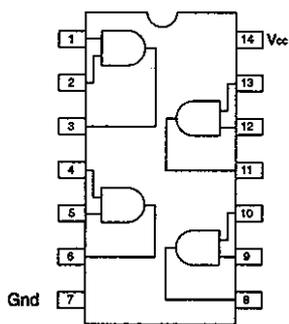


Fig 3.24

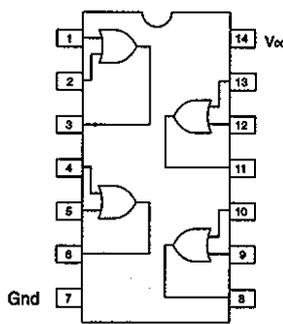
Activity 5

- 1 The 7432 quad 2-input OR gate IC will be used.
- 2 As each of the gates is different, **three** different ICs must be used. These are the **7408**, the **7432** and the **7404**, as shown below in Fig 3.25a–c). Once again the data sheets showing the IC pin connections must be consulted. The logic circuit showing the pin numbers and how the pins are wired up is shown below in Fig 25d.



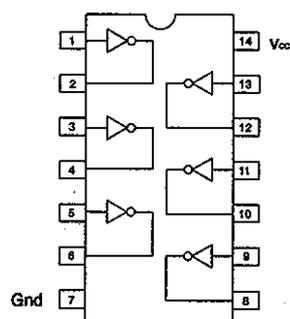
7408 quad 2-input
AND gate

Fig a



7432 quad 2-input
OR gate

Fig b



7404 hex INVERTER

Fig c

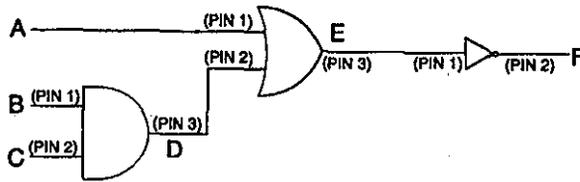


Fig d
Fig 3.25a-d

Thus we can see that pin 3 of the 7408 IC must be wired up or connected to pin 2 of the 7432 IC, and so on. Note that **each** of the ICs must be individually connected to the power supply points, i.e. the +5V line and the 0V or Gnd line.

The circuit should then be tested with the various level combinations of the inputs A, B and C. The results should be recorded in a truth table as shown below.

A	B	C	D	E	F
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	0

The final truth table should look like this:

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

The NAND and NOR logic gates

Study objectives

After studying this unit, you should be able to:

- ◇ derive the NAND and the NOR logic functions
- ◇ identify their logic symbols
- ◇ draw their truth tables
- ◇ write them as Boolean expressions
- ◇ apply digital signals to the NAND and NOR logic gates
- ◇ use data sheets to identify the component pin connections
- ◇ construct and test the logic circuits to verify their operation
- ◇ connect the various logic gates in different combinations
- ◇ draw logic circuits from Boolean expressions
- ◇ identify NAND and NOR gate fault conditions
- ◇ use the various logic gates in simple applications.

1 Introduction

So far you have been introduced to the three basic logic gates, namely, the AND gate, the OR gate, and the NOT gate or INVERTER. In this unit we will go one step further and look at two more simple logic functions or gates – the NAND and the NOR gates. The study objectives listed above will give you an indication of all the aspects that this unit will cover. You will also learn to draw a logic circuit from a given Boolean expression.

2 Combining the AND gate with the NOT gate

In the following logic circuit the AND gate is combined with the INVERTER or the NOT gate. The resulting truth table and the Boolean expression for the logic circuit are shown in Fig 4.1 on the next page.

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

$X = \text{NOT}(AB)$ which can be rewritten as $X = \overline{AB}$

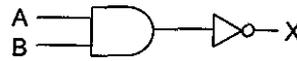


Fig 4.1

In digital systems there is a logic gate that can perform the function of the whole logic circuit shown above. The name of the logic function is derived from the two gates that are combined: NOT and AND, which is simplified to NAND. This NAND gate is the fourth logic function that we will study.

The logic symbol for the NAND gate, together with its truth table and Boolean expression, are shown in Fig 4.2 below.

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

$X = \overline{A B}$

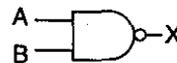


Fig 4.2



A **NAND gate** can be defined as a digital device that produces a 0 or a low output when all of its inputs are at a 1 or high level.

3 Combining the OR gate with the NOT gate

Just as we combined the AND function with the NOT function to form the NAND function, the OR and the NOT functions can also be combined. Fig 4.3 shows the OR gate connected to a NOT gate or an INVERTER.

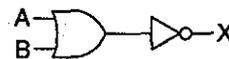


Fig 4.3

This circuit is represented by the following truth table and Boolean expression:

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

$$X = \text{NOT}(A + B)$$

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

In this case too there is a single logic gate that can perform the function of the above circuit. The name of the logic function is derived from the names of the two gates combined: NOT and OR, which is simplified to NOR. This NOR gate is the fifth logic gate that you have to know.

The logic symbol for the NOR gate, together with its truth table and Boolean expression, are shown below.

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

$$X = \overline{A + B}$$



Fig 4.4



A **NOR gate** can be defined as a device that produces a 0 or a low output when any of its inputs is at a 1 or high level.

4 Applying digital signals

We can therefore view the NAND and the NOR gates in the same way as the AND and the OR gates, which were described in Unit 2. To determine how these gates would respond to digital input signals, we simply need to remember the relevant truth table. Consider the following example.

Example 4.1 Signals A and B (in Fig 4.5) are applied to the inputs of the NAND gate shown in Fig 4.2. Using the NAND gate truth table shown below, the output X was obtained.

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

$$X = \overline{AB}$$

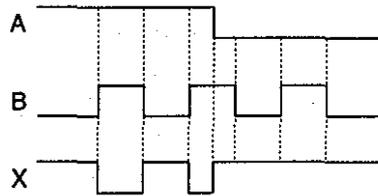


Fig 4.5

Activity 1



- 1 Define the basic NAND and NOR functions by describing and completing the truth table for each logic gate.
- 2 Identify the device illustrated in Fig 4.6, and write down the Boolean expression and the truth table for it.

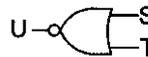


Fig 4.6

- 3 Fig 4.7 shows a 3-input NAND gate. How many rows will its truth table have? Write down the Boolean expression and truth table for this device.

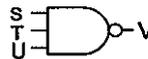


Fig 4.7

- 4 If the following signals were applied to the above gate, determine the output waveform.

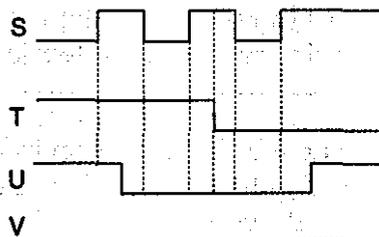


Fig 4.8

5 Practical implementation

The practical implementation and testing of the NAND and the NOR gates are very similar to the testing of the AND and OR gates, which we discussed in Unit 2. You will remember that we followed the procedure in Activity 3 in Unit 2. Please go back and make sure that you understand this testing procedure. Study it carefully, as you will be required to test other logic circuits as you work your way through the book. The following diagrams taken from data sheets show the pins of the 7400 and 7402 ICs. The 7400 is a quad 2-input NAND gate IC and the 7402 is a quad 2-input NOR gate IC. Refer to these diagrams when doing Activity 2.

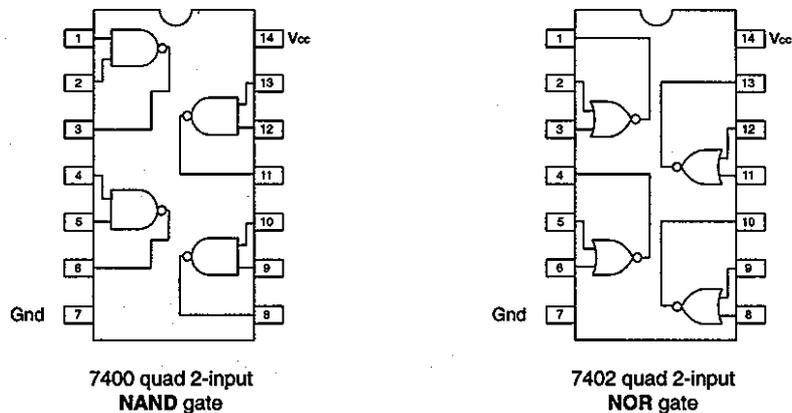


Fig 4.9

Activity 2



Following exactly the same procedure as in Activity 3 in Unit 2, construct the circuit and test the NAND and NOR gates, i.e. verify the functioning of the gates according to their truth tables. Always refer to the data sheet of the IC and use the correct pins. Again, pay careful attention to the precautions as indicated.

Activity 3



Examine the ICs shown in Fig 4.10 on the next page. Explain step by step how these ICs can be tested. (You do not actually have to test them.)

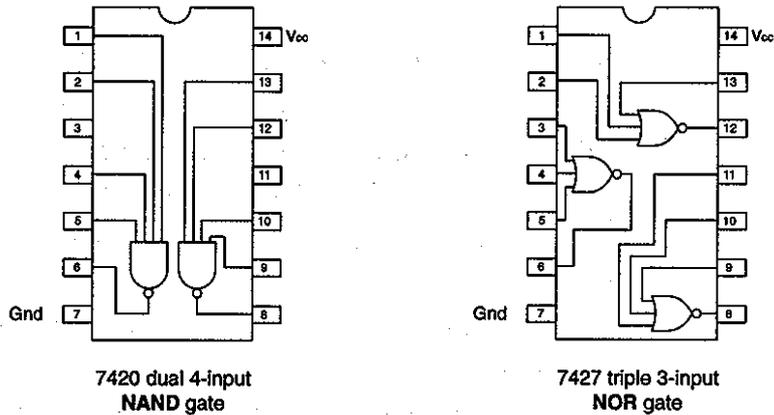


Fig 4.10

6 Combining the various logic gates

In Unit 3 we learnt how logic gates can be combined to form logic circuits.

The logic circuits we used were made up using the AND gate, the OR gate and the INVERTER, in various combinations. For each of these circuits we wrote down the Boolean expressions and truth tables. In this unit we will include the use of the NAND and the NOR gates in the same way.

Consider the following logic circuit:

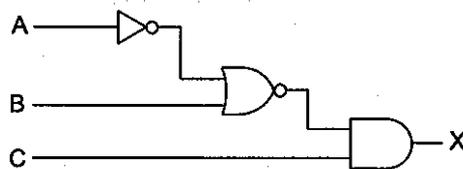


Fig 4.11

Let's write down the Boolean expression for this logic circuit, which consists of an INVERTER, a NOR gate and an AND gate.

First, from the INVERTER we get \bar{A} .

From the NOR gate the expression will be $\overline{\bar{A} + B}$.

Finally, from the AND gate we get $\overline{(\overline{A + B})} \cdot C$.

This can be rewritten without the dot: $X = \overline{(\overline{A + B})}C$.

Now let's write down the truth table for this logic circuit. This circuit has three inputs. Therefore, as you will have worked out, the truth table will have eight combinations or rows.

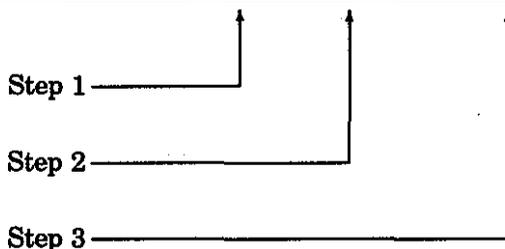
Let's start. Remember, the results will be clear from the truth table.

Step 1 In this example we will first have to determine the output of the INVERTER.

Step 2 Next, we will have to determine the output of the NOR gate.

Step 3 Finally we will have to calculate the output of the AND gate. To work out each output we have to know and use the truth table of **each gate** in the circuit as shown.

A	B	C	\overline{A}	$\overline{A + B}$	$\overline{(\overline{A + B})}C$
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	0	0	0



Activity 4



Examine the logic circuit shown in Fig 4.12. Determine the Boolean expression and the corresponding truth table for this circuit.

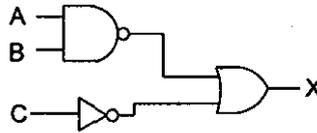


Fig 4.12

Activity 5



Using Fig 4.11 again, draw the output waveform X if signals A, B and C (shown in Fig 4.13) are applied to the logic circuit.

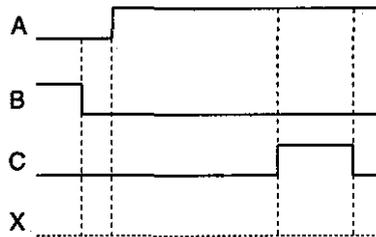


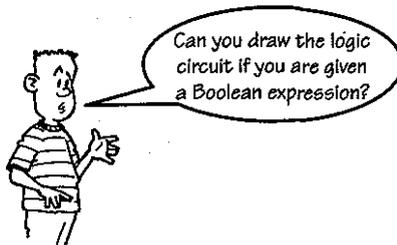
Fig 4.13

Activity 6



- 1 What ICs would have to be used if the logic circuit shown in Fig 4.11 is to be practically implemented and tested, and why would you use those specific ICs?
- 2 Implement and test the logic circuit shown in Fig 4.11. (Refer to Unit 3, section 7, if you are not sure how to do this.)

7 Drawing the logic circuit from the Boolean expression



We will try to do so by working through the following series of simple examples.

Example 4.2 $X = (A + B) + C$

This is read as (A OR B) OR C.

Solution

In the brackets the terms A and B are combined by the OR function. So, for this combination, we need a 2-input OR gate. The output of this gate is then combined with the next term, C, by another OR function. This means that we need another 2-input OR gate.

As you can see, for A and B we need a 2-input OR gate.



Fig 4.14a

The output from this OR gate and input C become the two inputs for another 2-input OR gate. Therefore the final circuit would look like this:

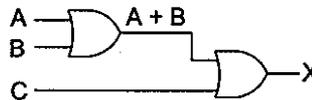


Fig 4.14b

Example 4.3 $Y = AB + C$

This is read as (A AND B) OR C.

Solution

The AND function is done before the OR function. A and B are fed into a 2-input AND gate (see Fig 4.15a). The output from the AND gate is then fed into a 2-input OR gate with C as shown in Fig 4.15b.



Fig a

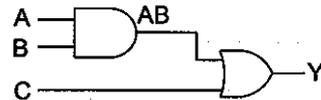


Fig b

Fig 4.15

Example 4.4 $Z = (A + B)(C + D)$

This is read as (A OR B) AND (C OR D).

Solution In this example, A and B are fed into a 2-input OR gate. The same applies to C and D. The outputs from these two gates are fed into an AND gate. The result is the circuit shown in Fig 4.16.

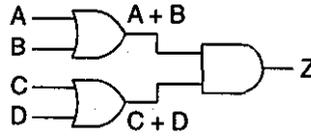


Fig 4.16

Example 4.5 $V = \bar{A} + (B + CD)$

Solution First of all, A is INVERTED so that it becomes \bar{A} .

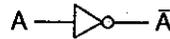


Fig 4.17a

C and D are combined with an AND gate.



Fig 4.17b

B and the output from the AND gate are fed into an OR gate.

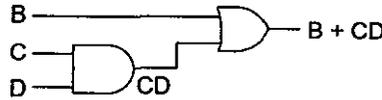


Fig 4.17c

The output from the INVERTER and the output from the OR gate are fed into another OR gate. The final circuit is shown in Fig 4.17d.

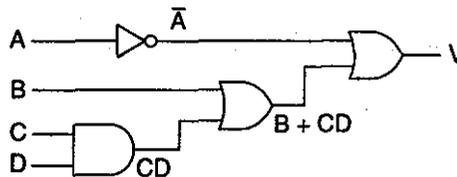


Fig 4.17d

Now try a few examples yourself!

Activity 7



Draw logic circuits for each of the following Boolean expressions.

- 1 $R = \bar{A} + \bar{B}$
- 2 $S = (A + \bar{B})C$
- 3 $T = (AB)(\bar{C}\bar{D})$
- 4 $U = (\bar{A}\bar{B})(C + \bar{D}\bar{E})$

8 Troubleshooting logic circuits

We introduced the process of identifying faulty components or troubleshooting a logic circuit in Unit 3 (section 8). In this unit we will look at typical examples featuring faulty components.

Example 4.6 A simple circuit is made up of just one 2-input NAND gate. The NAND gate IC is supplied with a +5 V power supply, i.e:

$$V_{cc} = +5 \text{ V.}$$

If the voltages at the inputs are at 0 V, and the output voltage is 0 V, is the gate functioning correctly?

Solution

NAND gate truth table:

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

$$X = \overline{AB}$$

According to the NAND gate truth table, the output is supposed to be a 1 or high. Therefore the measured voltage should be approximately +5 V. Therefore this gate seems to be faulty.

Example 4.7 The circuit shown in Fig 4.18 is to be used in an application.

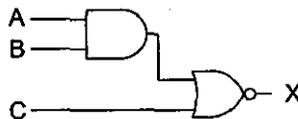


Fig 4.18

In the circuit, if the signal at point C is at +5V and the voltage at the output (point X) is measured to be +5V, what can we say about the circuit?

Solution

In a NOR gate, if any one of the inputs is high the output must be low. Therefore in this circuit the NOR gate must be faulty.

Note: It does not matter what the output from the AND gate is.

Example 4.8

For the logic circuit in Fig 4.19, if the INVERTER is not inverting, what will the truth table of the circuit look like?

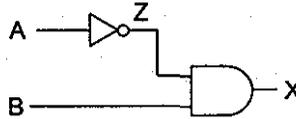


Fig 4.19

Solution

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

↑
↑

These columns are the same because the INVERTER is not inverting!

Activity 8



- 1 What would the truth table of Fig 4.19 have looked like if the INVERTER had been functioning correctly?
- 2 Consider the following logic circuit which is being tested:

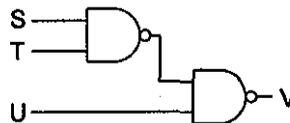


Fig 4.20

Inputs S, T and U are made low, i.e. 0 V. The output from the first gate is measured to be a low, and so is the output from the second gate. Are these gates functioning correctly? Explain.

- 3 A circuit is made up of four INVERTERS as shown below. When testing the circuit, it was found that if A is 0, then the level at Z is 1, and if the level at A is 1, then the level at Z is 0. The circuit is obviously faulty. How should one go about finding the faulty gate?

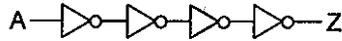


Fig 4.21

9 Simple applications of NAND and NOR gates

The important and interesting thing about NAND and NOR gates is that they may be used in situations where the basic logic gates are needed, but where the actual AND, OR or NOT gate ICs are for some reason unavailable. This means that NAND gates may be used in place of, say, the AND gate or the OR gate. The same can be done with NOR gates.

9.1 The NAND gate as an INVERTER

Imagine that an INVERTER is needed for a certain circuit, but there are none available. Can you use a NAND gate IC to replace the INVERTER?

Let us feed a signal – say A – into the two inputs of a NAND gate.

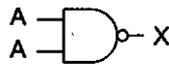


Fig 4.22a

We can redraw the gate as follows:

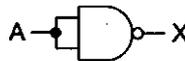


Fig 4.22b

Using the NAND gate truth table, we can determine that:

- ◇ if A is low, the level at X is high
- ◇ if A is high, the level at X is low.

Thus we can see that X is always the opposite or the inverse of A, i.e. $X = \bar{A}$. This shows us the NAND gate functioning as an INVERTER. In other words, if an INVERTER is unavailable we can use one NAND gate instead.

Activity 9



- 1 In a similar way as in 9.1 above, prove that a NOR gate can function as an INVERTER.
- 2 Can we use an AND gate as an INVERTER? Explain.

9.2 Replacing an AND gate with two NAND gates

Imagine that for a certain circuit we need a 2-input AND gate, but the AND gate IC is not available. However, we do have a NAND gate IC. Can we use the NAND gate IC instead?

Consider the following circuit:

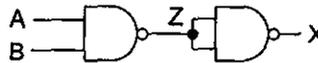


Fig 4.23

Let's draw the truth table for this logic circuit.

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

Table 4.1a

The table can be redrawn in a simplified version as:

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

Table 4.1b

Let's draw the truth table for an AND gate:

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

Table 4.1c

Compare tables 4.1b and 4.1c. What do you notice? They are exactly the same! This means that the circuit given in Fig 4.23 and the AND gate must function in the same way. Thus, two NAND gates connected as shown in Fig 4.23 can be used instead of an AND gate.

Activity 10



- 1 In a similar way as in 9.2 above, determine how two NOR gates can be used to replace one OR gate.
- 2 Prove that the following circuit made up of NAND gates can be used to replace a single OR gate.

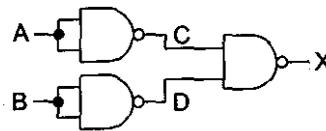


Fig 4.24



The NAND and the NOR gates are sometimes called **universal gates**. This is because they can be used to replace any of the other logic gates, i.e. a NAND gate may be used to replace an INVERTER, an AND gate, an OR gate, or a NOR gate. Similarly, a NOR gate can be used to replace an INVERTER, an AND gate, an OR gate, or a NAND gate.



Is $\overline{A \cdot B}$ equivalent to $\overline{A} \cdot \overline{B}$? Do you know how to read this? Easy! It reads 'Is NOT A AND NOT B equivalent to NOT (A AND B)?'

Let's look at their truth tables:

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

↑
↑

These two columns are not the same!

Draw the logic circuits for these two expressions and you will see that they are not the same either.

10 Summary

In this unit you were introduced to two new logic functions, namely the NAND and the NOR functions. We looked at representing these logic functions in various familiar ways, i.e. using logic gate symbols, truth tables and Boolean expressions. These gates – together with the basic AND, OR and NOT gates – were combined to form logic circuits. We analysed the circuits in terms of digital waveforms. In addition, we considered the practical implementation and testing of these logic circuits, and discussed the important aspect of drawing the logic circuit diagram from a given Boolean expression. We concluded the unit by troubleshooting circuits containing NAND and NOR gates, and showed how NAND and NOR gates can be used to replace other logic gates.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 For the circuit shown in Fig 4.25, write down the Boolean expression and draw the truth table.

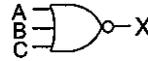


Fig 4.25

- 2 For the logic circuit shown in Fig 4.26a, draw the truth table and derive the Boolean expression. If the waveforms shown below (Fig 4.26b) are applied to the circuit, determine the output waveform.

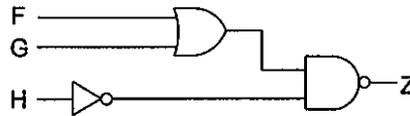


Fig 4.26a

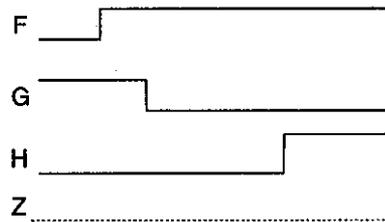


Fig 4.26b

- 3 Name the ICs that should be used if the circuits shown in Fig 4.25 and 4.26a are to be practically implemented and tested.
- 4 Draw the practical circuit diagram which is to be used to construct and test the circuit in Fig 4.27.

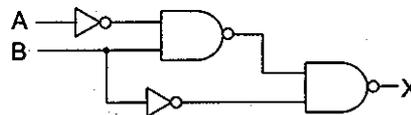


Fig 4.27

5 Are the following functions equivalent?

$$\overline{J + K + L} \text{ and } \overline{J} + \overline{K} + \overline{L}$$

Explain your answer using truth tables.

6 Replace each of the following gates with NOR gates only.

6.1 INVERTER

6.2 a two-input AND gate.

7 Determine the Boolean expression for each of the following logic circuits.

7.1

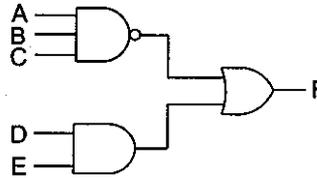


Fig 4.28

7.2

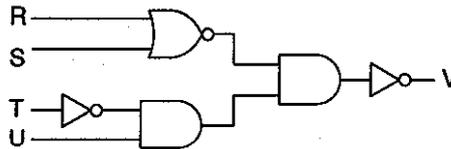


Fig 4.29

8 Draw logic circuits to represent the following Boolean expressions:

8.1 $A = (\overline{D} \overline{E} \overline{F}) + \overline{G} \overline{H}$

8.2 $B = \overline{(\overline{J} \overline{K}) + (\overline{J} \overline{M})}$

8.3 $C = (R + S)(\overline{T} + \overline{U})V$

Answers to activities

Activity 1

- 1 The NAND function is a logic function that produces a low or 0 output if all of the inputs are at a high or 1.

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

$$X = \overline{AB}$$

The NOR function is a logic function that produces a low or 0 output if any of the inputs is at a high or 1.

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

$$X = \overline{A + B}$$

- 2 The device is a 2-input NOR gate. Thus the Boolean expression is $U = \overline{S + T}$

S	T	U
0	0	1
0	1	0
1	0	0
1	1	0

- 3 Number of rows = $2^{\text{No. of inputs}} = 2^3 = 8$ rows

S	T	U	V
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$V = \overline{STU}$$

4

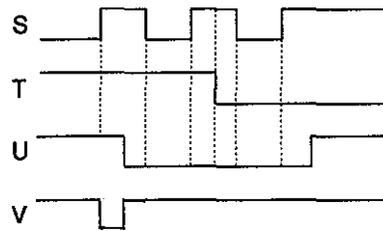


Fig 4.30

Activity 3

Fig a: This IC contains two NAND gates, each of which has four inputs.

$$X = \overline{ABCD}$$

Logic voltages (0V and +5V) must be applied to the inputs and the following truth table must be verified.

A	B	C	D	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Fig b: This IC contains three NOR gates, each of which has three inputs.

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

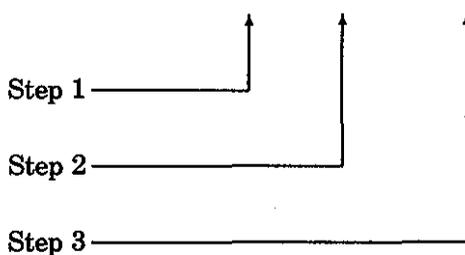
Logic voltages (0V and +5V) must be applied to the inputs and the following truth table must be verified.

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

Activity 4

From the NAND gate the output is \overline{AB} . The output from the INVERTER is \overline{C} . As we can see from the circuit, these two outputs become the two inputs for the 2-input OR gate. The output of the OR gate is: $\overline{AB} + \overline{C}$.

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



Activity 5

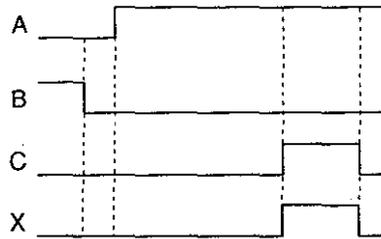


Fig 4.31

Activity 6

- The circuit consists of an INVERTER, a 2-input NOR gate, and a 2-input AND gate. If you consult data sheets, you will see that IC 7404 consists of six INVERTERS. (We used this IC in Unit 3.) We can use this IC here. The data sheets will show that IC 7402 consists of four 2-input NOR gates. Therefore we can also use this IC. The third IC that we can select for our circuit will have to be 7408 (the quad 2-input AND gate IC).

To summarise, the three ICs to be used are:

7404 (INVERTER IC)

7402 (NOR gate IC)

7408 (AND gate IC)

Activity 7

1

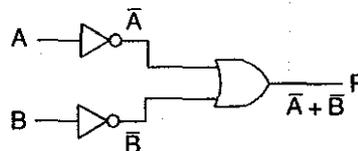


Fig 4.32

2

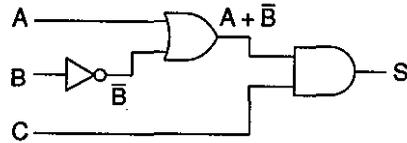


Fig 4.33

3

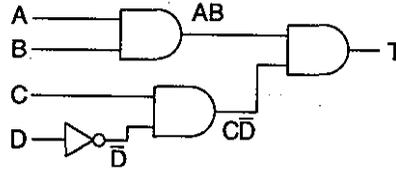


Fig 4.34

4

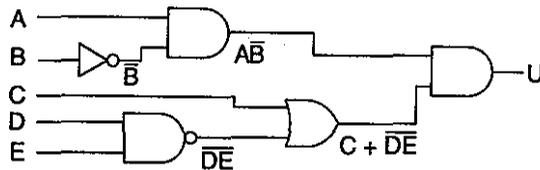


Fig 4.35

Activity 8

1

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

- 2 Gate 1, the first NAND gate, is faulty, as the output should be high. Gate 2, the second NAND gate, is also faulty: with both the inputs being at a low, the output should have been high.

- 3 Start testing the circuit from the first gate, i.e. try inputting a low and then a high. If the output from the first gate is correct, then check the next gate, and so on.

Activity 9

- 1 Here is a NOR gate with a signal A fed into both inputs:



Fig 4.36a

We can redraw this gate as:



Fig 4.36b

Using the NOR gate truth table:

- ◇ X is high if A is low
- ◇ X is low if A is high.

X is always the inverse of A; therefore the NOR gate functions as an INVERTER.

- 2 No! If we consider the AND gate as we did above:

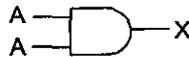


Fig 4.37

- ◇ X is low if A is low
- ◇ X is high if A is high.

Therefore we cannot use an AND gate as an INVERTER.

Activity 10

- 1 The circuit will be:

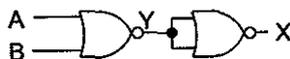


Fig 4.38

The truth table for this circuit is:

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

The truth table of an OR gate is the same as for the above circuit (column X). Therefore two NOR gates connected as shown can be used to replace one OR gate.

- 2 Again, draw the truth table for the circuit as shown below. If you compare the truth table you draw, to the truth table for an OR gate, you will see that the tables are the same. Therefore the circuit made up of NAND gates and connected as shown, can be used to replace one OR gate.

A	B	C	D	X
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

Simplification using Boolean algebra

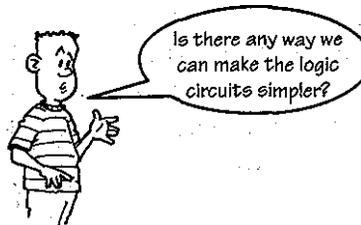
Study objectives

After studying this unit, you should be able to:

- ◇ recall the laws of Boolean algebra
- ◇ apply the laws of Boolean algebra to simplify logic expressions
- ◇ obtain logic expressions from truth tables.

1 Introduction

When you design digital electronic circuits, you may arrive at a solution that involves a number of logic circuits. These logic circuits can be represented by Boolean expressions. If a Boolean expression is too long or complex the logic circuit will consist of many logic gates or ICs. All these gates will have to be wired up or connected with many connections. A large number of gates or ICs will mean that the circuit will cost more to build. Later in the course you will also learn that if an electronic circuit has many connections, the circuit could be less reliable.



Yes! There are ways that logic circuits can be made simpler. In this unit you will learn how you can simplify logic circuits by simplifying their Boolean expressions. You will learn about **Boolean laws** and how these laws can help you simplify Boolean expressions.

In later units, when you are designing logic circuits for various applications, one of the main steps will involve deriving the Boolean expression from the truth table. Therefore this unit will end with a section that will show you how to derive or generate Boolean expressions from truth tables.

You can use Boolean laws to help you simplify Boolean expressions. Before you look at the Boolean laws, you should understand and remember the following common mathematical principles which also apply to Boolean algebra.

Commutation:

This means that $A \cdot B$ is the same as $B \cdot A$. And also that $A + B = B + A$.

Association:

In this case $A \cdot (B \cdot C) = (A \cdot B) \cdot C$. Also: $A + (B + C) = (A + B) + C$.

Distribution:

In this case $A(B + C) = AB + AC$.

Also: $(A + B)(C + D) = AC + AD + BC + BD$.

2 Simple single-variable Boolean laws

We can now turn our attention to the Boolean laws. The first set of Boolean laws consists of nine simple laws – these are called ‘single-variable Boolean laws’. By using your knowledge of logic functions and gates you will be able to work out these laws. However, the most important part of this section is to understand and memorise these laws so that you can use them to simplify Boolean expressions.

Law 1

Examine the simple logic gate shown in Fig 5.1. Can you name this gate? Good, write down the truth table for the AND gate on a piece of paper. We will need this AND truth table to prove the Boolean laws which follow.

$$X = A \cdot A$$

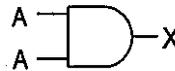


Fig 5.1

The signal A is fed into the two inputs of an AND gate. If A is at a logic 0, what do you think the value of X will be?

Correct! X will be 0. If A is at a logic 1, then X will be 1. Now we can say that X takes the value of A whether it is a 1 or a 0. If A is 0, X is 0. If A is 1, X is 1.

Therefore: $X = A \cdot A = A$.

Law 1: $A \cdot A = A$

Remember that $A \cdot A$ can also be written as AA .

Law 2

Look at Fig 5.2. It is also an AND gate; but have a look at the inputs.

$$X = A \cdot 1$$

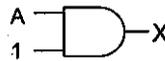


Fig 5.2

If A is 1, what will X be? X will be 1 AND 1 which will be 1. If A is 0, X will be 0 AND 1, which will be 0. We can see that X follows A, i.e.: If A is 0, X is 0, and if A is 1, X is 1. X is the same as A.

Therefore: $X = A \cdot 1 = A$.

Law 2: $A \cdot 1 = A$

Law 3

The following inputs to the AND gate (Fig 5.3) can be used to determine the third law.

$$X = A \cdot 0$$

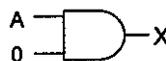


Fig 5.3

In this case no matter what logic value A takes (0 or 1) X will always be 0. (For an AND gate all inputs have to be 1 to get a 1 at the output.)

Therefore: $X = A \cdot 0 = 0$.

Law 3: $A \cdot 0 = 0$

Law 4

The following inputs to the AND gate can be used to determine the fourth law.

$$X = A \cdot \bar{A}$$

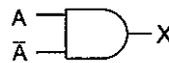


Fig 5.4

If A is 0, \bar{A} will be 1, and therefore X will be 0. If A is 1, \bar{A} will be 0, and therefore X will again be 0.

Therefore: $X = A \cdot \bar{A} = 0$.

Law 4: $A \cdot \bar{A} = 0$

What will the following four Boolean laws for OR gates be, if we analyse them in the same way as we did with the previous four laws? Keep in mind that we use the OR gate truth table for these proofs, but we replace the 1s and 0s with the A or other letters.

Law 5

$$X = A + A$$

You will see that if A is 0, X is 0; and if A is 1, X is 1. This means that X will always be the same as A.

Therefore, **Law 5: $A + A = A$**

Law 6

Now let's do the same with the Boolean expression: $X = A + 1$.

You will see that X is 1 in all cases, i.e. it does not matter whether A is a 1 or a 0, X is always 1.

Therefore, **Law 6: $A + 1 = 1$**

Law 7

Do the same with this Boolean expression: $X = A + 0$.



If A is 0, then X will be 0, and if A is 1, then X will be 1. In other words, X is the same or equal to A .

Therefore, **Law 7:** $A + 0 = A$

Law 8

What would this Boolean expression show: $X = A + \bar{A}$?

If A is 0, then \bar{A} will be 1, so $0 + 1 = 1$. If A is 1, then \bar{A} will be 0. Thus $1 + 0 = 1$. In both cases the expression is equal to 1.

Therefore, **Law 8:** $A + \bar{A} = 1$

Law 9

The last law in this series is very simple: $X = \bar{\bar{A}}$ [read as 'NOT (NOT A)'].

A is inverted twice. Therefore if A is 0, and is inverted the result is a 1. Inverting this result makes it 0 again. This means that if A is 0, X is 0; and if A is 1, then X is 1. In other words, X is the same as A .

Therefore, **Law 9:** $\bar{\bar{A}} = A$

You should memorise these laws. Later you will have to apply them to simplify Boolean expressions.

Here is a summary of the single-variable Boolean laws.

Law 1	$A \cdot A = A$
Law 2	$A \cdot 1 = A$
Law 3	$A \cdot 0 = 0$
Law 4	$A \cdot \bar{A} = 0$
Law 5	$A + A = A$
Law 6	$A + 1 = 1$
Law 7	$A + 0 = A$
Law 8	$A + \bar{A} = 1$
Law 9	$\bar{\bar{A}} = A$

Table 5.1

The Boolean laws can be written with any variable; not only A. For example: $P + \bar{P} = 1$ and $Y \cdot Y = Y$ and so on.

Now try and apply the Boolean laws in Activity 1.

Activity 1



1 Using the single-variable Boolean laws, simplify the following expressions.

1.1 $(A + A) + A = \dots$

1.2 $(B \cdot B) \cdot B = \dots$

1.3 $(C + C) + 1 = \dots$

1.4 $(D \cdot 1) + 0 = \dots$

1.5 $(E + \bar{E}) \cdot 1 = \dots$

1.6 $\bar{F} + (1 \cdot F) = \dots$

1.7 $(G \cdot 0) \cdot (G + H) = \dots$

1.8 $(J \cdot \bar{J}) \cdot K = \dots$

1.9 $\bar{\bar{L}} + 1 + 0 = \dots$

1.10 $M + 0 + M + \bar{M} = \dots$

2 Give two reasons why you need to simplify Boolean expressions.

3 Two other useful Boolean laws

If you now make use of the nine Boolean laws you have studied, you will be able to deduce two additional Boolean laws.

Law 10

Examine the following Boolean expression: $A + AB$. (Have your nine single-variable Boolean laws ready.)

$$A + AB = A \cdot 1 + AB \quad [\text{because } A \cdot 1 = A; \text{ law 2}]$$

This can be written simply as:

$$= A1 + AB$$

$$= A(1 + B) \quad [\text{using simple distribution}]$$

$$= A(1) \quad [\text{because } B + 1 = 1 + B = 1; \text{ law 6}]$$

$$= A \quad [\text{because } A \cdot 1 = A; \text{ law 2}]$$

This example can then be written in the form of a Boolean law.

$$\text{Law 10: } A + AB = A$$

Law 11

Consider the following Boolean expression. Using some of the Boolean laws that you have learnt so far, we can write:

$$A + \bar{A}B = (A + AB) + \bar{A}B \quad [\text{law 10}]$$

$$= (AA + AB) + \bar{A}B \quad [\text{law 1}]$$

$$= AA + AB + 0 + \bar{A}B \quad [\text{simply adding 0}]$$

$$= AA + AB + A\bar{A} + \bar{A}B \quad [\text{law 4}]$$

$$= A(A + B) + \bar{A}(A + B) \quad [\text{distribution}]$$

$$= (A + B)(A + \bar{A}) \quad [\text{distribution}]$$

$$= (A + B)(1) \quad [\text{law 8}]$$

$$= A + B \quad [\text{law 2}]$$

We can write this in the form of a Boolean law.

$$\text{Law 11: } A + \bar{A}B = A + B$$

Also memorise law 10 and law 11.

Study the following examples to see how the Boolean laws you have learnt so far can be used.

Example 5.1 Prove that $(A + B)(A + C) = A + BC$

There is no need to state which law is used in each step. However, for the ease of following the example, and to help you to recognise the laws, the laws are given in brackets.

Solution

$$\begin{aligned}
 (A + B)(A + C) &= AA + AC + AB + BC \\
 &= A + AC + AB + BC && \text{[law 1]} \\
 &= A(1 + C) + AB + BC && \text{[distribution]} \\
 &= A(1) + AB + BC && \text{[law 6]} \\
 &= A(1 + B) + BC && \text{[distribution]} \\
 &= A(1) + BC && \text{[law 6]} \\
 &= A + BC && \text{[Proved!]}
 \end{aligned}$$

Example 5.2 Prove that $AB + A\bar{B} + ABC = A$

Solution

$$\begin{aligned}
 AB + A\bar{B} + ABC &= AB(1) + ABC + A\bar{B} && \text{[law 2]} \\
 &= AB(1 + C) + A\bar{B} && \text{[distribution]} \\
 &= AB(1) + A\bar{B} && \text{[law 6]} \\
 &= AB + A\bar{B} && \text{[law 2]} \\
 &= A(B + \bar{B}) && \text{[distribution]} \\
 &= A(1) && \text{[law 8]} \\
 &= A && \text{[Proved!]}
 \end{aligned}$$

Example 5.3 Simplify the following Boolean expression:

$$A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C$$

Solution

$$\begin{aligned}
 &A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C \\
 &= A\bar{B}(\bar{C} + C) + \bar{A}C(B + \bar{B}) \text{ [distribution]} \\
 &= A\bar{B}(1) + \bar{A}C(1) && \text{[law 8]} \\
 &= A\bar{B} + \bar{A}C && \text{[law 2]}
 \end{aligned}$$

This expression cannot be simplified any further.

Activity 2

In this activity you do not need to state which Boolean law is used in each step of the exercises.

1 Prove each of the following:

1.1 $ABC + AB + A = A$

1.2 $A(\overline{ABC} + \overline{ABC}) = 0$

1.3 $(JK + J\overline{L})(K + J) + JK(\overline{K} + \overline{L}) = JK + J\overline{L}$

1.4 $(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$

2 Simplify the following Boolean expressions.

2.1 $AB + \overline{A}C + BC$

2.2 $X(Y + \overline{Z}) + Y(\overline{Y} + Z)$

4 The well-known DeMorgan's laws

To complete the list of Boolean laws that you need to know and use in the simplification of Boolean expressions, you have to memorise two more laws known as DeMorgan's laws. They are laws 12 and 13 below.

Law 12: $\overline{A \cdot B} = \overline{A} + \overline{B}$

This reads as 'NOT (A AND B) = (NOT A) OR (NOT B)'.

Law 13: $\overline{\overline{A} + \overline{B}} = \overline{\overline{A}} \cdot \overline{\overline{B}}$

This reads as 'NOT (A OR B) = (NOT A) AND (NOT B)'.

Each of these laws can be represented by sets of equivalent logic circuits. For example, Fig 5.5a and 5.5b illustrate the first of these two DeMorgan's laws, law 12.

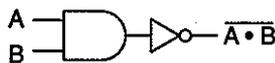


Fig 5.5a

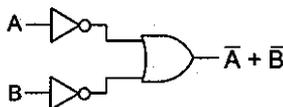


Fig 5.5b

Activity 3



Draw circuits which represent DeMorgan's second law, law 13.

These laws can also be extended to more than two variables. For example, with three variables we can write:

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

Here is a summary of the four new Boolean laws you have learnt.

Law 10	$A + AB = A$
Law 11	$A + \overline{A}B = A + B$
Law 12	$\overline{A \cdot B} = \overline{A} + \overline{B}$
Law 13	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Table 5.2

Activity 4



1 Simplify the following using DeMorgan's laws.

1.1 $\overline{C \cdot D} = \dots$

1.2 $\overline{D + E} = \dots$

1.3 $\overline{A + B + C} = \dots$

1.4 $\overline{C \cdot D \cdot E \cdot F} = \dots$

1.5 $\overline{X + Y} = \dots$

1.6 $A \cdot \overline{B \cdot C} = \dots$

1.7 $\overline{E + F} \cdot G = \dots$

1.8 $\overline{A + B + C} = \dots$

1.9 $\overline{J \cdot K \cdot L \cdot M} = \dots$



Before you study the rest of this unit, please make sure that you have memorised all thirteen Boolean laws. Refer to Tables 5.1 and 5.2 if necessary.

The next task involves the simplification of various types of Boolean expressions. The biggest problem is to recognise and identify which of the Boolean laws to use. You should also remember that there could be a number of ways to simplify a Boolean expression. You can use any of these ways, as long as you stick to

the laws. The only way that you will be good at the simplification of Boolean expression is by doing as many examples as possible.

Example 5.4 Prove that $\overline{AB} + \overline{A\overline{B}} = \overline{A}\overline{B} + \overline{A}B$

Solution

Choose the left-hand side of the equation to work with.

$$\begin{aligned}\overline{AB} + \overline{A\overline{B}} &= \overline{AB} \cdot \overline{A\overline{B}} && \text{[apply law 13 once]} \\ &= (\overline{A} + \overline{B})(\overline{\overline{A}} + \overline{\overline{B}}) && \text{[apply law 12 twice]} \\ &= (\overline{A} + \overline{B})(A + B) && \text{[apply law 9 twice]} \\ &= \overline{A}A + \overline{A}B + A\overline{B} + \overline{B}B \\ &= \overline{A}B + A\overline{B} && \text{[right-hand side of the equation]}\end{aligned}$$



This unit has a lot of examples. The best way to learn simplification, and of course digital systems, is to work out plenty of examples.

Activity 5



- 1 Prove that: $\overline{A}B\overline{C} + \overline{A}BC + \overline{A}\overline{B}C = \overline{A}B + \overline{A}C$
- 2 Prove that: $AB + BC + CA = (A + B)(B + C)(C + A)$
- 3 Simplify: $B = \overline{\overline{J}K(L + \overline{M})}$
- 4 Simplify: $D = \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{AB})}$

5 Generating Boolean expressions from truth tables

When you design a logic circuits for various applications, one of the main steps would be to derive the Boolean expression from the truth table. In this section you will learn how to derive or generate Boolean expressions from truth tables.

A Boolean expression is generated for each output term or variable. The truth table will show this as output columns. The expression will be written in terms of the inputs or input terms.

Let's look at an example.

Example 5.5 Examine the truth table illustrated below.

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

In this example the output column or variable is X. The inputs or input terms are A and B. The Boolean expression is derived as follows.

Examine the output column. Write down a term corresponding to each 1 in the output column. Each term is made up of all the input variables ANDed together. However, if there is a 0 in the input side, then the term is written with a bar.



A bar sign over a variable or term indicates that the term is inverted. For example, A inverted can be written as \bar{A} . This is read as 'NOT A' or as 'A bar'. Similarly, \bar{AB} is read as 'NOT (A AND B)' or as '(AB) bar'. In a logic circuit the bar can therefore be represented by means of an INVERTER.

The first term is therefore $\bar{A}B$. Note that A and B are ANDed together, but the bar is on the A.

To obtain the second term the same procedure is followed. Look for the next 1 in the output column. Then look at the corresponding input side and write down the input variables ANDed together. If there is a 0 in any of the input terms you have to invert that variable or write it with a bar.

In this case both the A and B columns have a 1: so there is no bar. The second term is written as AB.

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

→ $\bar{A}B$: first term

→ AB: second term

Next you will use the terms you have worked out to write down the expression by combining the terms with the OR symbol, which is the +.

This will result in the following expression:

$$X = \bar{A}B + AB.$$

This Boolean expression can sometimes be simplified using the techniques learnt earlier in this unit. Try simplifying this expression. You should get: $X = B$.

Let's look at another example to make sure you understand the procedure.

Example 5.6 Write down the Boolean expression for the following truth table.

J	K	L	X	
0	0	0	0	
0	0	1	1	← first term
0	1	0	1	← second term
0	1	1	0	
1	0	0	0	
1	0	1	1	← third term
1	1	0	0	
1	1	1	1	← fourth term

The output column X has four 1s. This means that the final expression will have four ANDed terms.

The first term, corresponding to J K L being 0 0 1, will be $\bar{J} \bar{K} L$.

The second term, corresponding to J K L being 0 1 0, will be $\bar{J} K \bar{L}$.

The third and fourth terms will be $J \bar{K} L$ and $J K L$.

To get the final expression, each of the ANDed terms must be separated by the OR sign, which is the +.

The final expression will therefore be:

$$X = \bar{J} \bar{K} L + \bar{J} K \bar{L} + J \bar{K} L + J K L.$$

Once again, note that this expression will not necessarily be in its simplest form. Simplification techniques can be used to simplify the expression.

Activity 6



- 1 Simplify the expression in example 5.6.
- 2 Write the Boolean expressions for each of these truth tables.

2.1

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

2.2

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

2.3

X	Y	Z	P
0	0	0	0
0	1	0	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

2.4

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

[This table has two outputs, therefore you have to work out two expressions.]

- 3 Simplify the Boolean expressions in 2.1 to 2.4 above, if it is possible to do so.

6 Summary

To design and implement digital circuits you will often have to generate the Boolean expression for the circuit. The circuit should be as simple as possible so that it will have fewer components or ICs. Fewer components would mean lower cost, quicker implementation and testing, and quicker troubleshooting.

Fewer components would also mean fewer connections or links between components. This in turn results in circuits being more reliable. Therefore the Boolean expressions need to be simplified as much as possible.

In this unit you were initially introduced to Boolean laws. These laws were then applied to simplify expressions. The unit was concluded with a section explaining how Boolean expressions can be generated from a truth table.

In the next unit you will study a different technique which is often used to simplify expressions.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Simplify the following Boolean expression using the laws of Boolean algebra.

$$1.1 \quad W = X\bar{Y} + X(Z + Y) + X\bar{Z}$$

$$1.2 \quad Z = (A + B)(A + C)(A + \bar{B})$$

$$1.3 \quad Q = AB + AB(\overline{C + D}) + \overline{ABC}$$

$$1.4 \quad R = AB + ABD + BCD + CD$$

$$1.5 \quad S = (\bar{A} + B)(A + C)(B + C + D)$$

$$1.6 \quad A = \overline{\overline{(X + \bar{Y}Z)}(\bar{X}YZ)}$$

- 2 Prove or disprove the following expressions.

$$2.1 \quad \overline{\bar{A}(C + \bar{D})} + \overline{\bar{C}(A + \bar{B})} = A + B + C + D$$

$$2.2 \quad ABC + AB(\overline{\bar{A}\bar{C}}) = A(C + \bar{B})$$

$$2.3 \quad X\bar{Y}Z + X\bar{Y}\bar{Z} + XYZ = X(Y + Z)$$

$$2.4 \quad \overline{\bar{A}\bar{B}} + \bar{A} + B = 1$$

3 Generate Boolean expressions for these truth tables.

3.1

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

3.2

J	K	L	Y
0	0	0	1
0	1	1	1
1	0	0	1
1	1	0	1
1	1	1	1

3.3

X	Y	L	M
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

This table has two outputs: L and M.

3.4

A	B	C	D	Z
0	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1

4 Simplify the expressions generated in question 3 if it is possible to do so.

Answers to activities

Activity 1

$$\begin{aligned} 1.1 \quad (A + A) + A &= A + A \\ &= A \end{aligned}$$

$$\begin{aligned} 1.2 \quad (B \cdot B) \cdot B &= B \cdot B \\ &= B \end{aligned}$$

$$\begin{aligned} 1.3 \quad (C + C) + 1 &= C + 1 \\ &= 1 \end{aligned}$$

$$\begin{aligned} 1.4 \quad (D \cdot 1) + 0 &= D + 0 \\ &= D \end{aligned}$$

$$\begin{aligned} 1.5 \quad (E + \bar{E}) \cdot 1 &= 1 \cdot 1 \\ &= 1 \end{aligned}$$

$$\begin{aligned} 1.6 \quad \bar{F} + (1 \cdot F) &= \bar{F} + F \\ &= 1 \end{aligned}$$

$$\begin{aligned} 1.7 \quad (G \cdot 0) \cdot (G + H) &= 0 \cdot (G + H) \\ &= 0 \end{aligned}$$

Remember that 0 AND any term always equals 0. (See the AND gate truth table.)

$$\begin{aligned} 1.8 \quad (J \cdot \bar{J}) \cdot K &= 0 \cdot K \\ &= 0 \end{aligned}$$

$$\begin{aligned} 1.9 \quad \bar{L} + 1 + 0 &= L + 1 + 0 \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

$$\begin{aligned} 1.10 \quad M + 0 + M + \bar{M} &= M + M + \bar{M} \\ &= M + \bar{M} \\ &= 1 \end{aligned}$$

2 To enable you to design simpler logic circuits using fewer components.

To enable you to make circuits with fewer connections between components.

To make circuits more reliable and easier to test and troubleshoot.

Activity 2

Although you were not asked to give the Boolean laws used in each step, the answers supplied here will show the actual laws used, for your benefit.

$$\begin{aligned}
 \mathbf{1.1} \quad & ABC + AB + A \\
 &= ABC + AB + A(1) \quad [\text{law 2}] \\
 &= A(BC + B + 1) \quad [\text{distribution}] \\
 &= A(BC + 1) \quad [\text{law 6}] \\
 &= A(1) \quad [\text{law 6}] \\
 &= A \quad [\text{law 2}] \\
 \\
 \mathbf{1.2} \quad & A(\overline{ABC} + \overline{ABC}) \\
 &= A\overline{ABC} + A\overline{ABC} \quad [\text{distribution}] \\
 &= 0 \cdot BC + 0 \cdot \overline{BC} \quad [\text{law 4}] \\
 &= 0 + 0 \quad [\text{law 3}] \\
 &= 0 \\
 \\
 \mathbf{1.3} \quad & (JK + J\overline{L})(K + J) + JK(\overline{K} + \overline{L}) \\
 &= JKK + JJK + JK\overline{L} + J\overline{L} \\
 &\quad + JK\overline{K} + JK\overline{L} \quad [\text{distribution}] \\
 &= JK + JK + JK\overline{L} + J\overline{L} \\
 &\quad + J(0) + JK\overline{L} \quad [\text{laws 1 and 4}] \\
 &= JK + JK\overline{L} + J\overline{L} + 0 \quad [\text{laws 3 and 5}] \\
 &= JK(1 + \overline{L}) + J\overline{L} \quad [\text{laws 2 and 7}] \\
 &= JK(1) + J\overline{L} \quad [\text{law 6}] \\
 &= JK + J\overline{L} \quad [\text{law 2}] \\
 \\
 \mathbf{1.4} \quad & (A + B)(\overline{A} + C)(B + C) \\
 &= (A\overline{A} + \overline{A}B + AC + BC)(B + C) \quad [\text{distribution}] \\
 &= (0 + \overline{A}B + AC + BC)(B + C) \quad [\text{law 4}] \\
 &= \overline{A}BB + \overline{A}BC + ABC + ACC \\
 &\quad + BBC + BCC \quad [\text{distribution}] \\
 &= \overline{A}B + \overline{A}BC + ABC + AC \\
 &\quad + BC + BC \quad [\text{law 1}] \\
 &= \overline{A}B + BC(\overline{A} + A) + AC \\
 &\quad + BC + BC \quad [\text{distribution}] \\
 &= \overline{A}B + BC + AC \quad [\text{laws 2, 5 and 8}]
 \end{aligned}$$

$$\begin{aligned}
 &= 0 + \overline{AB} + AC + BC \quad [\text{law 7}] \\
 &= A\overline{A} + \overline{AB} + AC + BC \quad [\text{law 4}] \\
 &= \overline{A}(A + B) + C(A + B) \quad [\text{distribution}] \\
 &= (A + B)(\overline{A} + C) \quad [\text{distribution}]
 \end{aligned}$$

$$2.1 \quad AB + \overline{AC} + BC$$

$$\begin{aligned}
 &= AB + \overline{AC} + BC(1) \quad [\text{law 2}] \\
 &= AB + \overline{AC} + BC(A + \overline{A}) \quad [\text{law 8}] \\
 &= AB + \overline{AC} + ABC + \overline{ABC} \quad [\text{distribution}] \\
 &= AB + ABC + \overline{AC} + \overline{ABC} \quad [\text{rearranging terms}] \\
 &= AB(1 + C) + \overline{AC}(1 + B) \quad [\text{law 2}] \\
 &= AB(1) + \overline{AC}(1) \quad [\text{law 6}] \\
 &= AB + \overline{AC} \quad [\text{law 2}]
 \end{aligned}$$

This expression cannot be simplified any further.

$$2.2 \quad X(Y + \overline{Z}) + Y(\overline{Y} + Z)$$

$$\begin{aligned}
 &= XY + X\overline{Z} + Y\overline{Y} + YZ \quad [\text{distribution}] \\
 &= XY + X\overline{Z} + 0 + YZ \quad [\text{law 4}] \\
 &= XY + X\overline{Z} + YZ \quad [\text{law 7}] \\
 &= XY(1) + X\overline{Z} + YZ \quad [\text{law 2}] \\
 &= XY(Z + \overline{Z}) + X\overline{Z} + YZ \quad [\text{law 8}] \\
 &= XYZ + XY\overline{Z} + X\overline{Z} + YZ \quad [\text{distribution}] \\
 &= YZ(X + 1) + X\overline{Z}(Y + 1) \quad [\text{law 2}] \\
 &= YZ(1) + X\overline{Z}(1) \quad [\text{law 6}] \\
 &= YZ + X\overline{Z} \quad [\text{law 2}]
 \end{aligned}$$

This expression cannot be simplified any further.

Activity 3

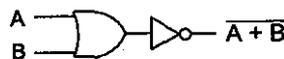


Fig 5.6

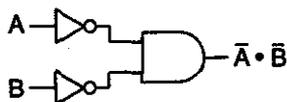


Fig 5.7

Activity 4

1.1 $\overline{C} + \overline{D}$

1.2 $\overline{D}\overline{E}$

1.3 $\overline{A}\overline{B}\overline{C}$

1.4 $\overline{C} + \overline{D} + \overline{E} + \overline{F}$

1.5 $\overline{X}Y$

1.6 $\overline{A}\overline{B} + \overline{A}\overline{C}$

1.7 $\overline{E}\overline{F}G$

1.8 $\overline{A}\overline{B}\overline{C}$

1.9 $(\overline{J} + \overline{K})(\overline{L} + \overline{M})$

Activity 5

1
$$\begin{aligned} & \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}C(B + \overline{B}) \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}C \\ &= \overline{A}(B\overline{C} + C) \\ &= \overline{A}(B + C) \\ &= \overline{A}B + \overline{A}C \end{aligned}$$

2 Work with the right-hand side of the equation.

$$\begin{aligned} & (A + B)(B + C)(C + A) \\ &= (AB + AC + BB + BC)(C + A) \\ &= ABC + ACC + BBC + BCC + AAB + AAC \\ & \quad + ABB + ABC \\ &= ABC + AC + BC + BC + AB + AC + AB + ABC \\ &= ABC + AC + BC + AB \\ &= AC(B + 1) + BC + AB \\ &= AC + BC + AB \\ &= AB + BC + CA \text{ [Same as left-hand side of the equation.]} \end{aligned}$$

$$\begin{aligned}
 3 \quad B &= \overline{\overline{JK}(\overline{L+M})} \\
 &= \overline{\overline{JK}} + \overline{\overline{L+M}} \\
 &= \overline{\overline{J}} + \overline{\overline{K}} + \overline{\overline{LM}} \\
 &= J + K + LM
 \end{aligned}$$

$$\begin{aligned}
 4 \quad D &= \overline{\overline{(\overline{A+B})(\overline{C+AB})}} \\
 &= \overline{\overline{(\overline{A+B})}} + \overline{\overline{(\overline{C+AB})}} \\
 &= \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{(\overline{C}AB)}} \\
 &= \overline{\overline{A}} + \overline{\overline{B}} + CAB \\
 &= \overline{\overline{A}} + \overline{\overline{B}} + ABC
 \end{aligned}$$

Activity 6

$$\begin{aligned}
 1 \quad X &= \overline{JKL} + \overline{JKL} + \overline{JKL} + \overline{JKL} \\
 &= \overline{JKL} + \overline{JKL} + JL(\overline{K} + K) \\
 &= \overline{JKL} + \overline{JKL} + JL \\
 &= L(\overline{JK} + J) + \overline{JKL} \\
 &= L(J + \overline{K}) + \overline{JKL} \\
 &= JL + L\overline{K} + \overline{JKL}
 \end{aligned}$$

$$2.1 \quad X = \overline{AB} + A\overline{B}$$

$$2.2 \quad X = \overline{AB}\overline{C} + \overline{AB}C + \overline{AB}\overline{C} + \overline{AB}C$$

$$2.3 \quad P = \overline{XYZ} + X\overline{YZ} + X\overline{YZ}$$

$$2.4 \quad X = \overline{AB} + A\overline{B} + AB;$$

$$Y = \overline{A}\overline{B} + \overline{A}B + A\overline{B} + AB$$

3.1 The expression in 2.1 cannot be simplified.

$$3.2 \quad X = \overline{A}\overline{B}(\overline{C} + C) + \overline{A}\overline{B}(\overline{C} + C)$$

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

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

$$= \overline{A}$$

$$3.3 \quad P = X\overline{Y} + \overline{X}YZ$$

$$3.4 \quad X = A + B;$$

$$Y = 1$$

Simplification using Karnaugh maps

Study objectives

After studying this unit, you should be able to:

- ◇ draw Karnaugh maps
- ◇ generate a Karnaugh map from a truth table
- ◇ use the Karnaugh map for simplification of logic circuits.

1 Introduction

One of the main features of a well-designed digital circuit is its simplicity. You will be able to design simple logic circuits if you know how to apply simplification techniques to complex expressions. In the previous unit you simplified expressions by using the laws of Boolean algebra.

In this unit you will be introduced to another simplification technique. In many cases this technique can be used to simplify logic expressions easily and relatively quickly. However, before we can proceed with the simplification, we will have to become familiar with a new concept: the **Karnaugh map**. (This is read as 'Car-no map'.)

2 What are Karnaugh maps?



A Karnaugh map represents a logic circuit in a graphical form. The map is drawn as a block consisting of a set of smaller blocks or cells. Each small block corresponds to a row in the associated truth table. Just as we used truth tables to represent logic circuits, we can use Karnaugh maps to represent logic circuits. However, the main function of the Karnaugh map is to enable us to simplify a logic circuit easily and quickly.

The diagrams on the next page show typical Karnaugh maps. As you can see, each map consists of small blocks or cells. In this book we will refer to the small blocks or cells simply as blocks. Each Karnaugh map is labelled in a particular way. Examine the diagrams closely and learn how to draw and label them.

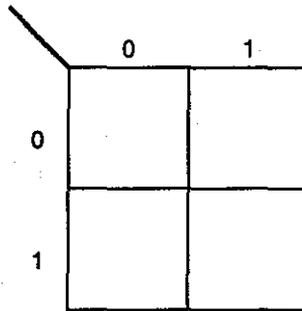


Fig 6.1a

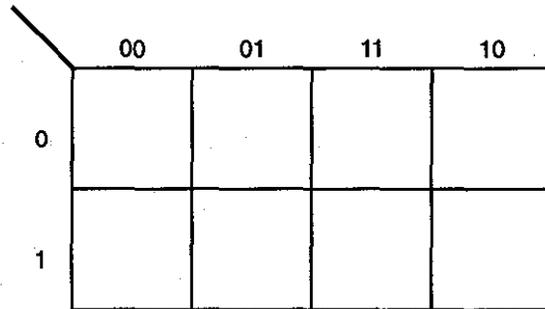


Fig 6.1b

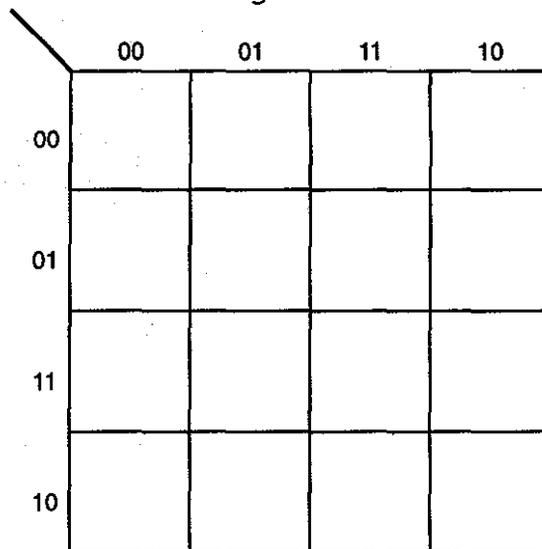


Fig 6.1c

The maps in Fig 6.1a-c are blank and therefore not fully labelled. Fig 6.2 shows an example of a complete Karnaugh map.

		BC			
		00	01	11	10
A	0	0	1	1	1
	1	0	0	0	1

Fig 6.2

3 Generating a Karnaugh map from a truth table

You will recall that for a logic circuit with 2-inputs, the truth table was made up of four rows. In a similar way the Karnaugh map to represent the same circuit will be made up of four blocks. Each block in the Karnaugh map corresponds to a particular row in the truth table.

Example 6.1

The table below is a simple truth table representing a 2-input logic circuit. The corresponding Karnaugh map is shown in Fig 6.3.

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

The first block represents the case where input A is 0 and input B is also 0. The output X as shown in the truth table is 0. This 0 is inserted into this block.

The second block represents the case where A is 0 and B is 1. The output X in this case is a 1. This 1 is inserted into this block.

		B	
		0	1
A	0	0	1
	1	1	0

Can you identify this block?

Yes! It represents the case where A is 1 and B is also 1. The value of the output X is 0.

What about this block?

Fig 6.3

Activity 1


- 1 What does a Karnaugh map represent?
- 2 Draw a blank Karnaugh map consisting of eight blocks.
- 3 Draw completed Karnaugh maps corresponding to the following truth tables.

3.1

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

3.2

J	K	Z
0	0	1
0	1	1
1	0	1
1	1	0

Example 6.2 The table below shows a 3-input truth table and Fig 6.4a is the corresponding Karnaugh map. Note that the same Karnaugh map may also be drawn as shown in Fig 6.4b.

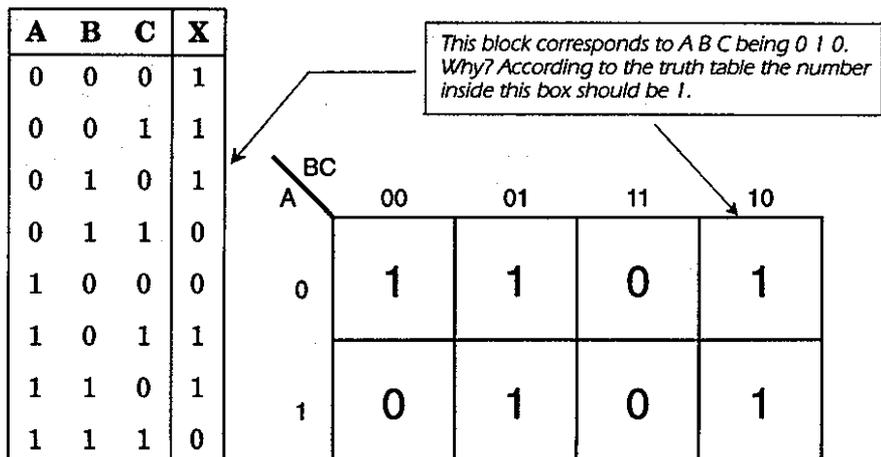


Fig 6.4a

A BC	0	1
00	1	0
01	1	1
11	0	0
10	1	1

Fig 6.4b

Try to work out:

- 1 How each of the other blocks in the map are labelled.
- 2 How each block corresponds to a specific row in the truth table.
- 3 How each block is filled with a value.

Example 6.3 A four-input or four-variable truth table has sixteen rows. Similarly a four-variable Karnaugh map has sixteen blocks – each block corresponding to a specific row in the truth table. The table on the next page shows a typical four-variable truth table and Fig 6.5 shows its corresponding Karnaugh map.

Once again, determine how the Karnaugh map is labelled and how the blocks are filled in.

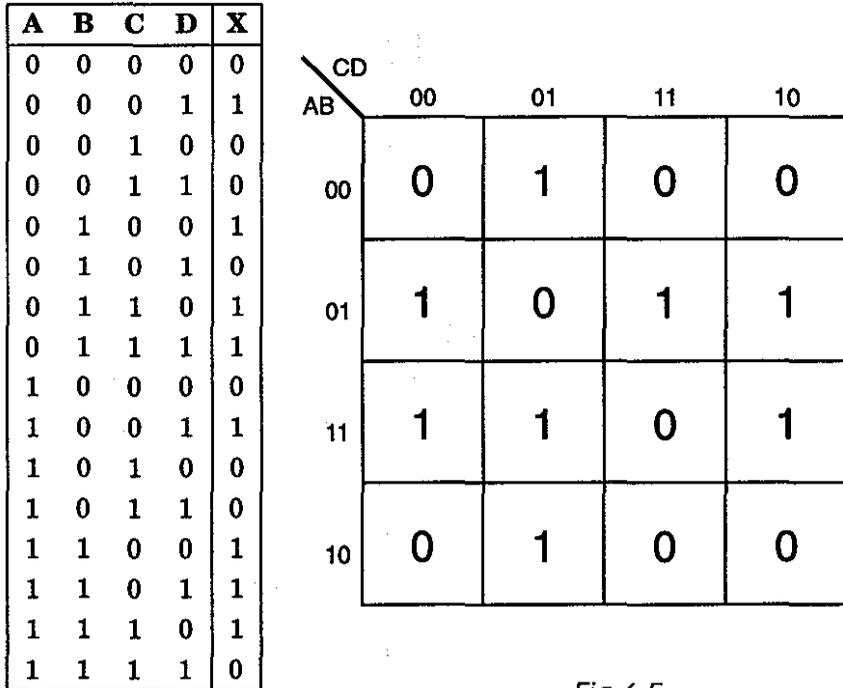


Fig 6.5

Activity 2



1 Draw completed Karnaugh maps for the following truth tables.

1.1

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

1.2

J	K	L	X
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	1
1	1	1	1

1.3

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

1.4

P	Q	R	S	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

4 Simplification using the Karnaugh map

So far you have been able to represent digital information in the form of a Karnaugh map. The process of simplifying logic circuits using the Karnaugh map, involves two steps:

- 1 finding the appropriate groups of 1s in the map
- 2 extracting the simplified Boolean expression from the identified groups.

4.1 Finding the appropriate groups

The main aim of this step is to find all the 1s that can be grouped. You have to find the biggest groups first.

How big are these groups?

In a Karnaugh map of 16 blocks:

- ◇ the biggest group consists of 16 1s
- ◇ the next group consists of 8 1s
- ◇ the next group consists of 4 1s
- ◇ the next group consists of 2 1s (in this order)

In a Karnaugh map of 8 blocks:

- ◇ the biggest group consists of 8 1s
- ◇ the next group consists of 4 1s
- ◇ the next group consists of 2 1s (in this order)

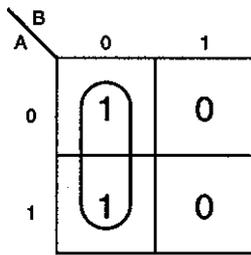
In a Karnaugh map of 4 blocks:

- ◇ the biggest group consists of 4 1s
- ◇ the next group consists of 2 1s (in this order)

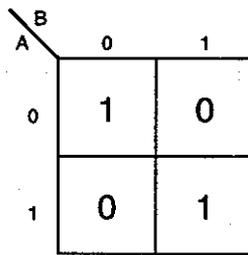
How are these groups found?

The groups must consist of 1s that are adjacent to one another. This means the ones must be alongside, above or below one another. A group cannot consist of ones that are diagonally across from one another. The groups should then be marked with a loop as shown in the figures on the next page. Groups may partly overlap. The easiest way to understand this grouping process is by looking at a few examples.

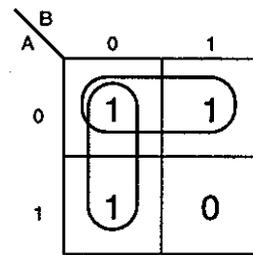
Example 6.4a Karnaugh maps with four blocks.



one group
Fig 6.6a

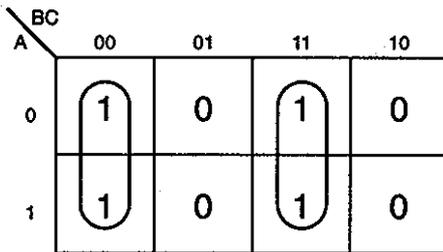


no groups
Fig 6.6b

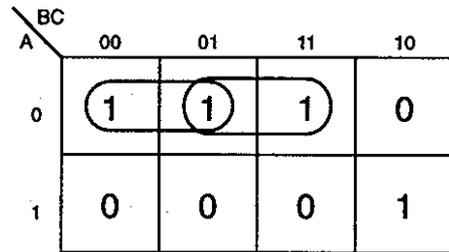


two overlapping groups
Fig 6.6c

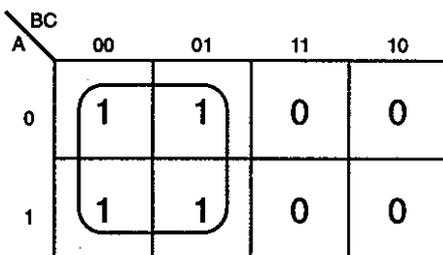
Example 6.4b Karnaugh maps with eight blocks.



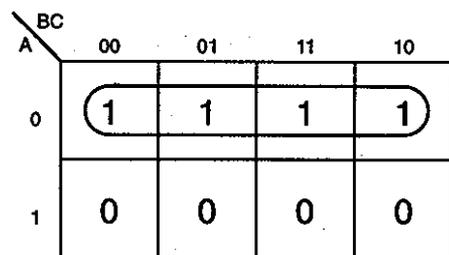
two groups of 2 1s each
Fig 6.7a



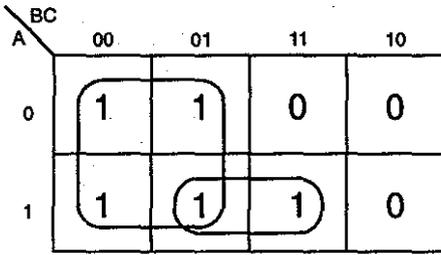
two overlapping groups of 2 1s
Fig 6.7b



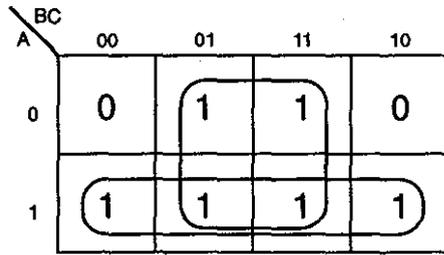
one group of 4 1s
Fig 6.7c



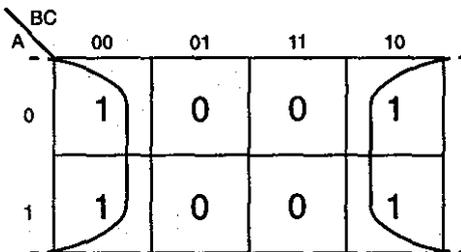
one group of 4 1s
Fig 6.7d



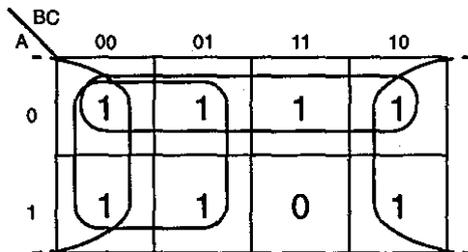
one group of 4 1s and
one group of 2 1s
Fig 6.7e



two groups of 4 1s
Fig 6.7f

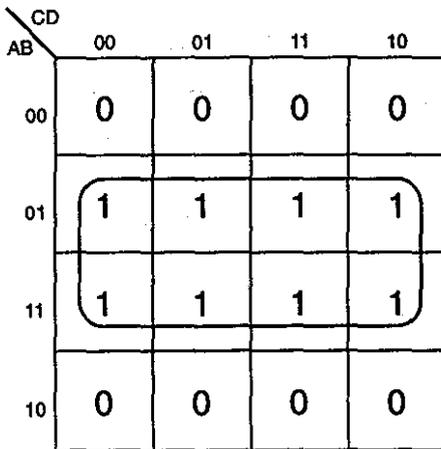


one group of 4 1s
The two sides of the Karnaugh
map are linked.
Fig 6.7g

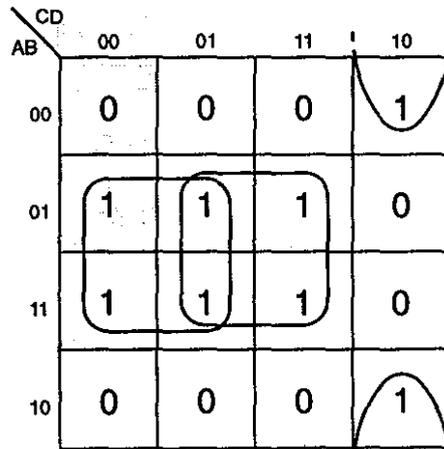


three group of 4 1s
Fig 6.7h

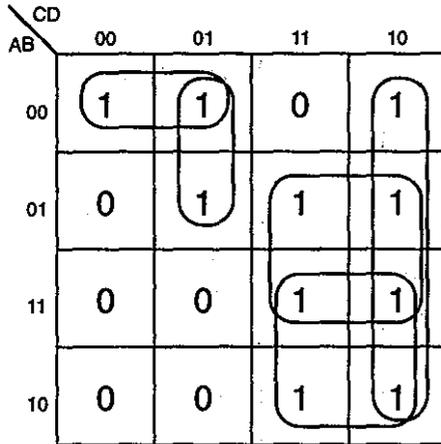
Example 6.4c Karnaugh maps with sixteen blocks



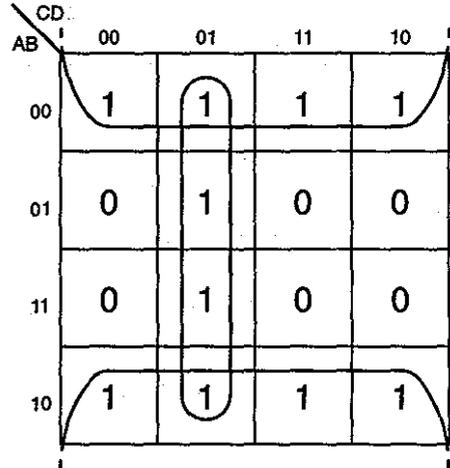
1 group of 8 1s
Fig 6.8a



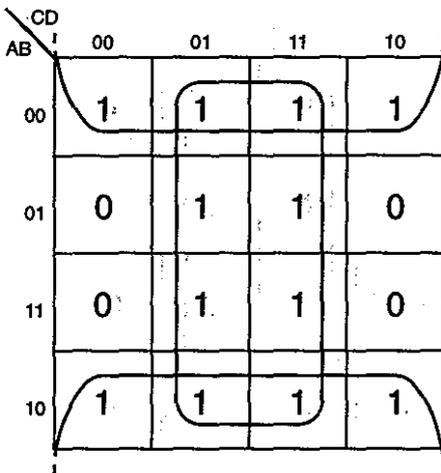
2 groups of 4 1s
1 group of 2 1s
Fig 6.8b



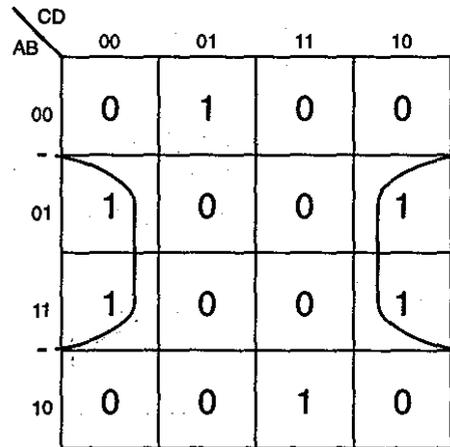
3 groups of 4 1s
2 groups of 2 1s
Fig 6.8c



1 group of 8 1s
1 group of 4 1s
Fig 6.8d



2 groups of 8 1s
Fig 6.8e



1 group of 4 1s
2 single 1s
Fig 6.8f

Activity 3



1 Group the 1s in the following Karnaugh maps as shown in the examples above.

1.1

		BC			
		00	01	11	10
A	0	1	1	1	0
	1	0	1	1	1

Fig 6.9a

1.2

		ST			
		00	01	11	10
R	0	1	0	1	0
	1	0	1	0	1

Fig 6.9b

1.3

		CD			
		00	01	11	10
AB	00	0	0	1	0
	01	1	1	1	0
	11	0	1	1	1
	10	0	1	0	0

Fig 6.9c

1.4

		TU			
RS		00	01	11	10
	00	1	0	0	1
	01	0	0	1	1
	11	0	0	1	1
	10	1	0	0	1

Fig 6.9d

- 2 Group the 1s in the Karnaugh maps which you drew for Activity 2.

4.2 Extracting the simplified Boolean expression

Each loop in the Karnaugh map can be uniquely labelled in a special way by a Boolean term. The final Boolean expression consists of all the Boolean terms combined together with the OR sign; which is the + sign. Again this procedure will best be understood by studying examples.

Example 6.5 Consider the Karnaugh map shown in Fig 6.10 below. The following points should be noted:

- ◇ the simplified Boolean expression may be written in terms of the two variables: A and B
- ◇ the map consists of **one loop**; therefore the simplified Boolean expression will be made up of **one term only**.

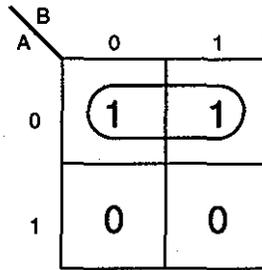


Fig 6.10

The loop is labelled as follows: Which variables are common to all the 1s in the loop? In this case $A = 0$ is common to both 1s in the loop. Thus we label the loop \bar{A} .

The simplified Boolean expression can then be written as: $X = \bar{A}$

Example 6.6 Consider the Karnaugh map shown in Fig 6.11.

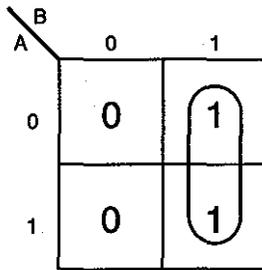


Fig 6.11

One loop means one term. $B = 1$ is common to all the 1s in the loop. Therefore the term is simply: B .

The Boolean expression can be written as: $X = B$

Example 6.7 Consider the Karnaugh map shown in Fig 6.12.

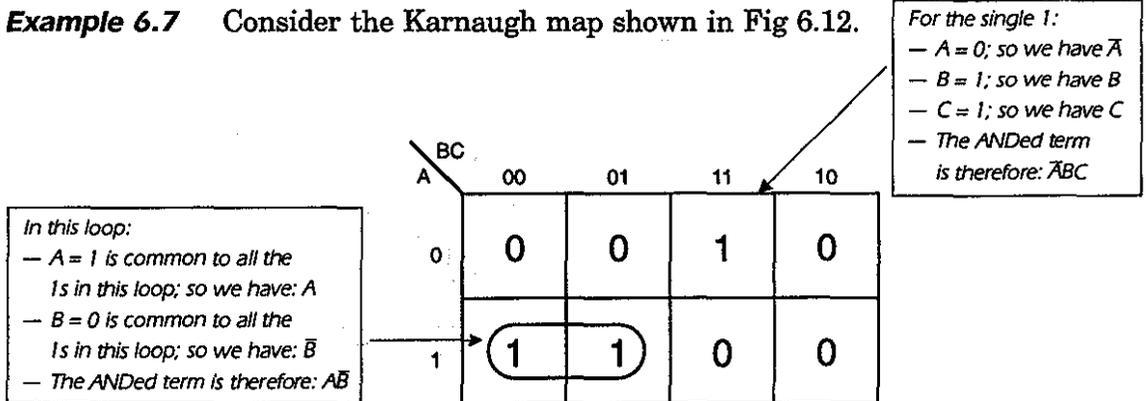


Fig 6.12

The simplified Boolean expression can be written as:

$$X = \overline{A}B + \overline{A}BC.$$

Example 6.8 In each of the following figures the term for each loop has been indicated, and the simplified Boolean expression appears below the figure.

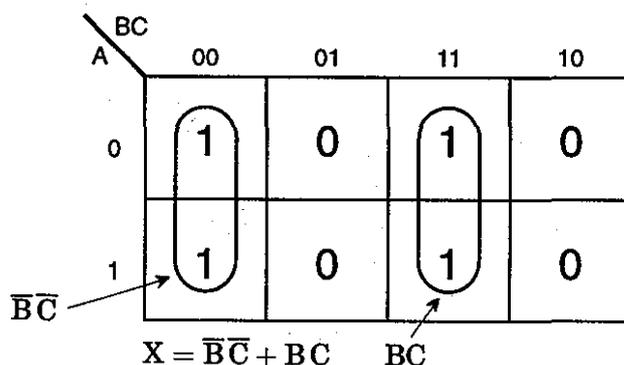


Fig 6.13

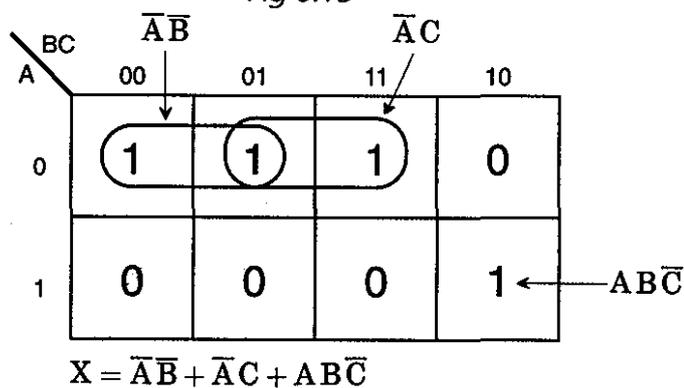


Fig 6.14

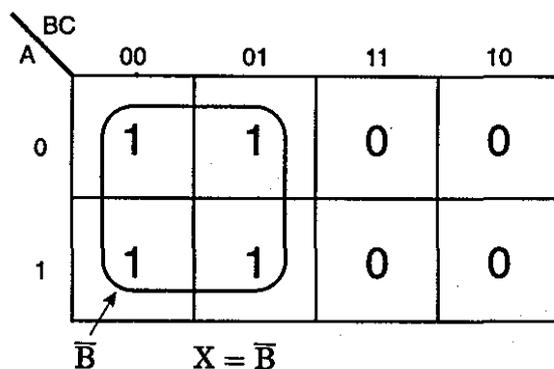
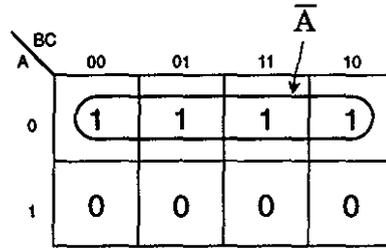
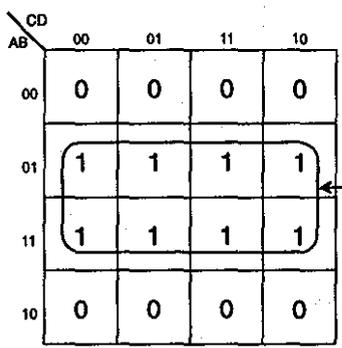


Fig 6.15



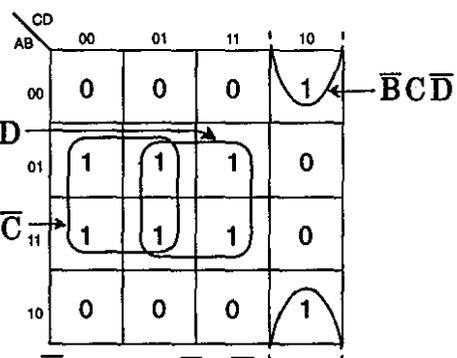
$X = \bar{A}$

Fig 6.16



$X = B$

Fig 6.17



$X = B\bar{C} + BD + \bar{B}\bar{C}\bar{D}$

Fig 6.18

Activity 4



- Group the 1s in each of the following Karnaugh maps. Then write down the simplified Boolean expression from each map.

1.1

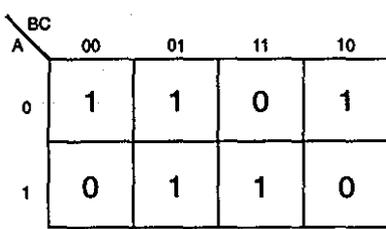


Fig 6.19a

1.2

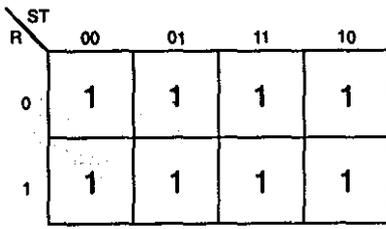


Fig 6.19b

1.3

		CD			
	AB	00	01	11	10
00		1	0	0	1
01		1	0	0	0
11		0	0	0	1
10		1	0	0	1

Fig 6.19c

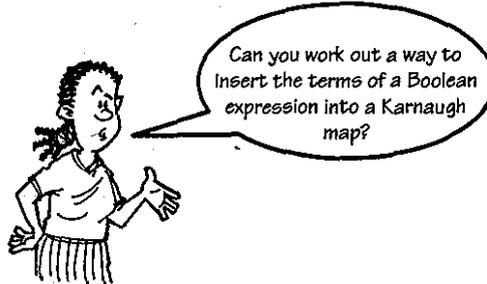
1.4

		CD			
	AB	00	01	11	10
00		1	0	0	0
01		0	1	0	0
11		0	0	1	0
10		0	0	0	1

Fig 6.19d

(For additional practice in simplification extract the Boolean expression for the examples in Activity 3.)

Extension task



Consider the Boolean expression of the type that we worked with in Unit 5. For example: $X = \overline{A}BC + \overline{A}B + ABC + \overline{B}$. The four terms of this expression can be inserted into a Karnaugh map in the form of 1s. The map can then be simplified in the usual way to extract the simplified Boolean expression.

The challenge is for you to work out how it is done!

5 Summary

This unit (together with Unit 5) completes the essential techniques necessary for the simplification of Boolean expressions. When using simplified Boolean expressions you will be able to build simpler logic circuits. These circuits will have fewer components, fewer connections between components, and therefore be cheaper and more reliable. The circuits would also be easier and quicker to troubleshoot.

At this stage you should be equipped with most of the basic logic circuit concepts and the tools necessary to tackle the problem of designing logic circuits for various applications. In the next few units we will look at the design of digital circuits for various applications.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Which two techniques can be used to simplify logic expressions?
- 2 List the advantages of having simple logic circuits.
- 3 Draw blank Karnaugh maps for the following:
 - 3.1 three variables: R S and T
 - 3.2 four variables: V W X and Y
- 4 Extract simplified logic expressions from the following Karnaugh maps.

4.1

		Y	
X		0	1
0	1	1	1
1	1	1	1

Fig 6.20a

4.2

		BC			
A		00	01	11	10
0	1	0	1	1	
1	0	1	0	1	

Fig 6.20b

4.3

		ST			
R		00	01	11	10
0	1	1	1	1	
1	1	0	0	1	

Fig 6.20c

4.4

		CD			
		00	01	11	10
AB	00	1	1	1	1
	01	1	0	0	1
	11	1	0	0	1
	10	1	1	1	1

Fig 6.20d

4.5

		CD			
		00	01	11	10
AB	00	1	1	0	1
	01	0	1	0	1
	11	1	0	1	0
	10	1	0	0	1

Fig 6.20e

4.6

		XY			
VW		00	01	11	10
00		0	0	1	1
01		1	1	0	0
11		1	1	0	1
10		0	0	1	1

Fig 6.20f

5 Simplify the following Boolean expressions using Karnaugh maps.

5.1 $X = A + B + AB$

5.2 $Y = A\bar{B} + ABC + \bar{A}BC + \bar{B}C + \bar{A}\bar{C}$.

Answers to activities

Activity 1

1 A Karnaugh map represents a logic circuit in a graphical form. The map is drawn as a block consisting of a set of smaller blocks or cells. Each small block corresponds to a row in the associated truth table.

2

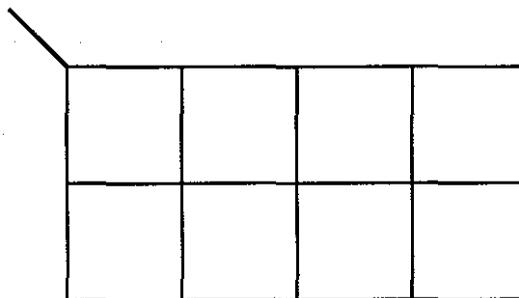


Fig 6.21

3.1 Karnaugh map for X (Fig 6.22a).

3.2 Karnaugh map for Z (Fig 6.22b).

		B	
		0	1
A	0	0	0
	1	1	1

Fig 6.22a

		K	
		0	1
J	0	1	1
	1	1	0

Fig 6.22b

Activity 2

1.1

		BC			
		00	01	11	10
A	0	1	1	0	1
	1	0	1	0	1

Fig 6.23a

1.2

		KL			
		00	01	11	10
J	0	0	0	0	
	1	1	1	1	

Fig 6.23b

1.3

AB \ CD	00	01	11	10
00	0	1	0	0
01	1	0	1	1
11	1	1	0	1
10	0	1	0	0

Fig 6.23c

1.4

PQ \ RS	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

Fig 6.23d

Activity 3

1.1

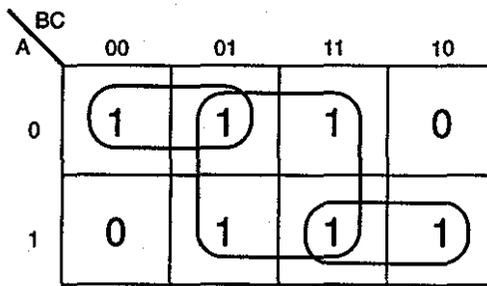


Fig 6.24a 1 group of 4 1s; 2 groups of 2 1s

1.2

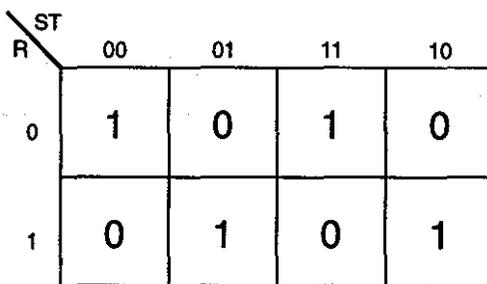


Fig 6.24b All single 1s

1.3 In this case we get four groups of 2 1s only. The middle group of 4 1s is excluded as all the 1s in this middle group are included in the other four groups.

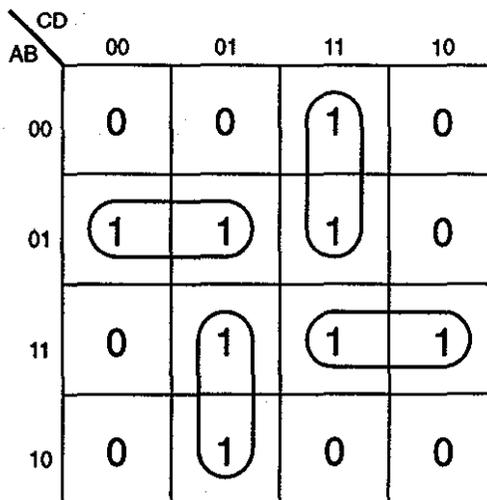


Fig 6.24c

1.4

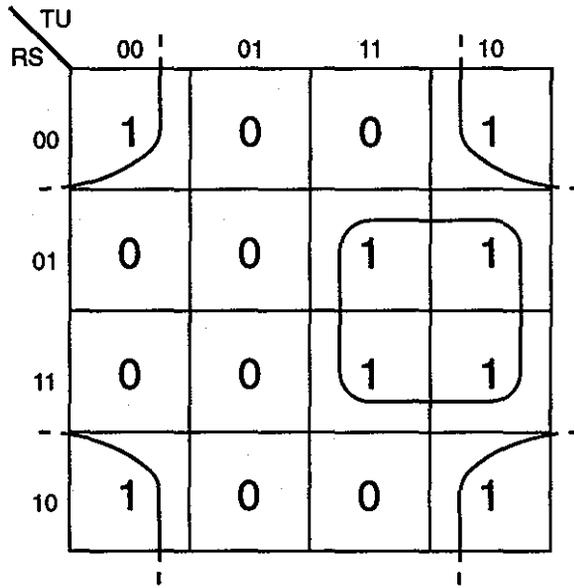


Fig 6.24d 2 groups of 4 1s

2.1

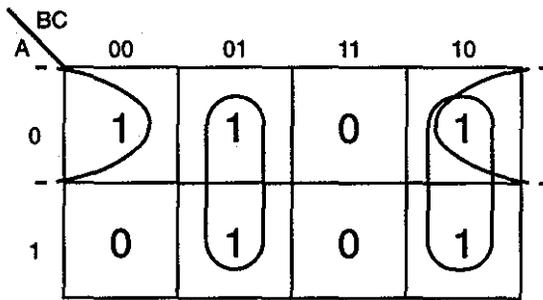


Fig 6.25a 3 groups of 2 1s

2.2

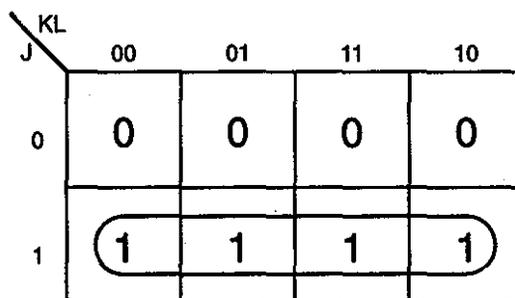


Fig 6.25b 1 group of 4 1s

2.3

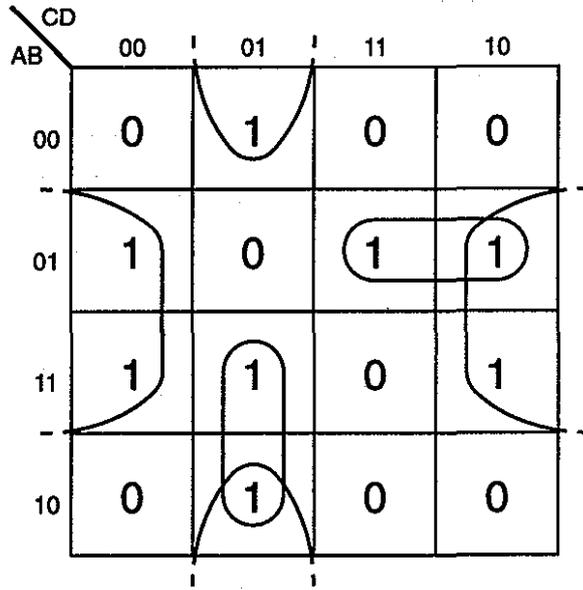


Fig 6.25c 1 group of 4 1s; 3 groups of 2 1s

2.4

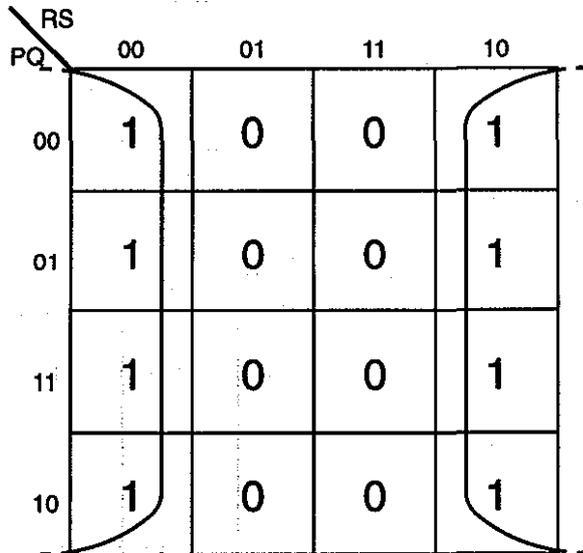
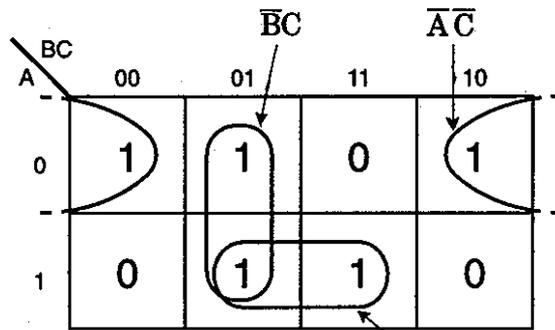


Fig 6.25d 1 group of 8 1s

Activity 4

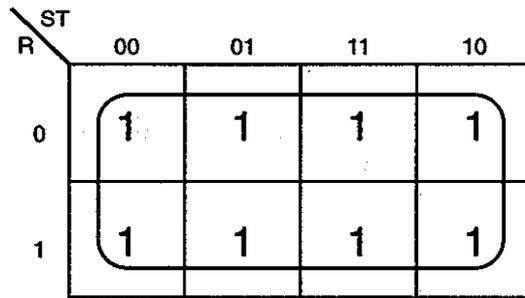
1.1



$$X = \overline{BC} + AC + \overline{AC}$$

Fig 6.26a

1.2

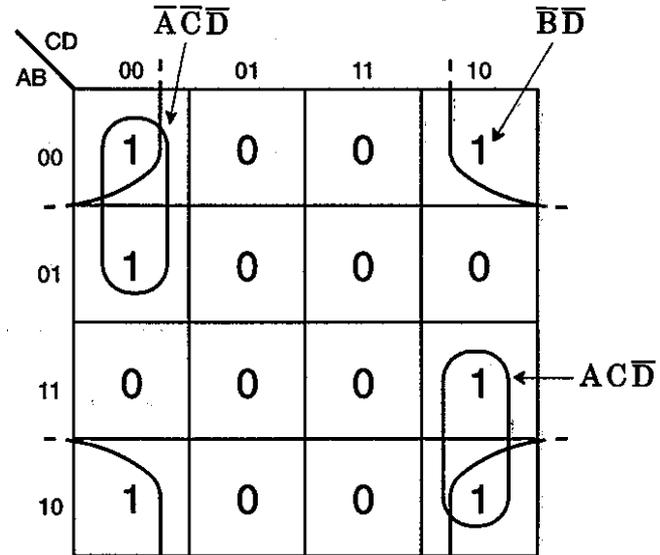


The output is always 1,

therefore: $X = 1$

Fig 6.26b

1.3



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

Fig 6.26c

1.4

AB \ CD		CD			
		00	01	11	10
AB	00	1	0	0	0
	01	0	1	0	0
	11	0	0	1	0
	10	0	0	0	1

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

Fig 6.26d

Basic combinational logic circuits

Study objectives

After studying this unit, you should be able to:

- ◇ describe a combinational logic circuit
- ◇ list the techniques involved in the design of combinational logic circuits
- ◇ design simple combinational logic circuits
- ◇ describe and use the binary number system
- ◇ describe the Exclusive-OR and the Exclusive-NOR gates
- ◇ understand the concept of comparing binary quantities
- ◇ design digital comparator circuits.

1 Introduction

In digital systems most circuits can be classified into one of two types: combinational or sequential circuits. In Units 7 and 8 you will be introduced to a number of combinational logic circuits. You will learn to describe, design and apply various types of combinational logic circuits. (In Unit 9 you will learn about sequential logic circuits.)

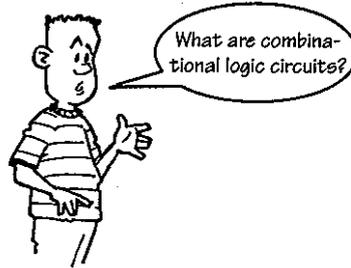
In this unit you will also learn about the binary number system. You need to know how to deal with and manipulate binary numbers in order to design logic circuits for all types of digital applications, including computer systems.

So far you have worked with logic circuits that were made up of the following logic gates: AND, OR, NOT, NAND and NOR. You also need to know two additional logic gates: the Exclusive-OR and the Exclusive-NOR gates. All these gates will then be used in future aspects of digital systems.

This unit will conclude with a specific type of combinational logic circuit: the comparator. We will look at how numbers are compared in digital systems, and at how logic circuits can be designed

to compare numbers. The concept of advanced digital ICs will be briefly introduced.

Let's first look at combinational logic circuits.



2 Combinational logic circuits

All the logic circuits that you have learnt about so far are combinational logic circuits. A combinational logic circuit is a logic circuit that has one or more outputs, and many inputs. The outputs are directly dependent on some logical combination of the inputs. Each output can therefore be described by a logic or Boolean expression. Fig 7.1 shows a diagram of a general combinational logic circuit.

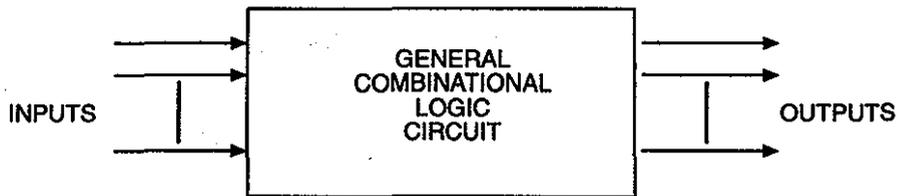


Fig 7.1

Fig 7.2 shows an example of a simple combinational logic circuit.

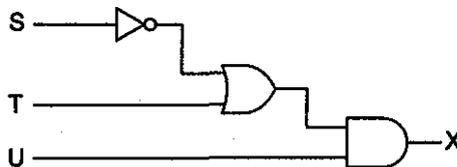


Fig 7.2

This circuit has three inputs and one output. The output X can be written in terms of the inputs. The Boolean expression for this combinational logic circuit is: $X = (\bar{S} + T) \cdot U$

Combinational logic circuits can be designed for various applications. In the next section we will look at how these logic circuits can be designed.

Activity 1



- 1 What are the two main types of digital circuits?
 - 2 Draw a simple combinational logic circuit that has one output and four inputs.
 - 3 Draw one combinational logic circuit with the following Boolean expressions:
 $X = A + BC$;
 $Y = ABC$
 - 4 How many inputs and outputs does the combinational logic circuit in question 3 have?
-

3 Design techniques

In most cases where a logic circuit is needed, the problem is set out in words. This problem needs to be analysed and then simplified into input and output terms. The design of a logic circuit can be done in one of two ways: (1) by following a simple method using a truth table (2) by logical reasoning.

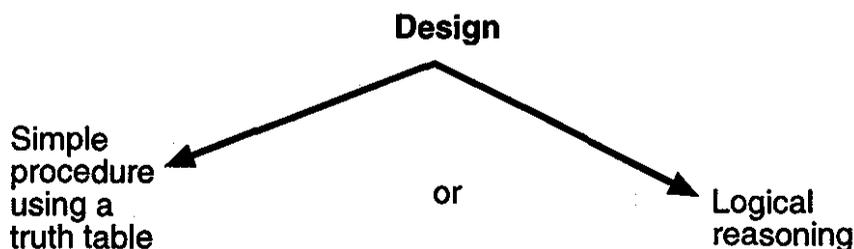


Fig 7.3

In most cases you will design combinational logic circuits by using a set method, or series of steps, including the use of a truth table. In a few cases you will use logical reasoning. You will learn how

to use the truth table method first. The method of logical reasoning will be discussed later, in section 7.2 of this unit.

3.1 Design method using a truth table

The diagram in Fig 7.4 summarises the steps you should follow when using this method.

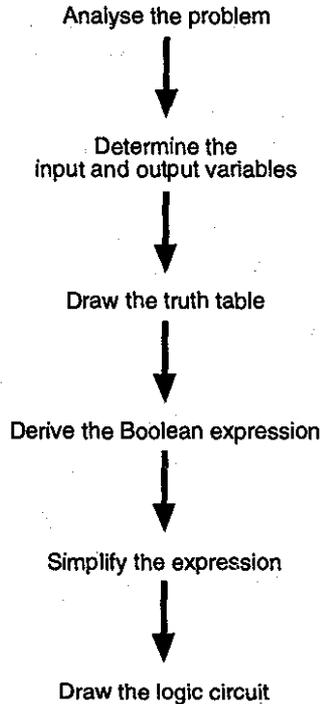


Fig 7.4

The most important step is to carefully and correctly analyse the problem. If this step is done properly the inputs and outputs will be determined correctly. The purpose of drawing the truth table is to change information from the problem statement into table form. The steps to derive the Boolean expression, simplify the expression, and draw the logic circuit, were covered in previous units. Simplification can be done by using either the Boolean laws or the Karnaugh maps.

The best way to understand how you can design combinational logic circuits using the set method involving truth tables, is by working through a few examples.

Example 7.1 A computer room has two windows. If one or both of the windows are broken the alarm must be sounded. Design a logic circuit to sound the alarm.

Solution **Analyse the problem:** The circuit should have one output to sound the alarm. Each of the windows can act as inputs.

Determine the variables: There is one output variable – to sound the alarm. Let's use X for the output variable. There are two input variables – window 1 and window 2. Let's use the symbols A and B for the two input variables.

Draw the truth table: Use the following logic states in the truth table: When X is 1 the alarm is on. When X is 0 the alarm is off.

When a window is broken, A or B is 1 and when a window is not broken, A or B is 0.

A	B	X	
0	0	0	(No windows broken, alarm off)
0	1	1	(Window 2 is broken, alarm on)
1	0	1	(Window 1 is broken, alarm on)
1	1	1	(Both windows are broken, alarm on)

Derive the Boolean expression: From the truth table we can write the following expression: $X = \bar{A}B + A\bar{B} + AB$. This expression could be simplified.

Simplify the expression: This could be done by using either the Boolean laws or a Karnaugh map.

We will use a Karnaugh map in this example. Drawing the Karnaugh map from the truth table we get:

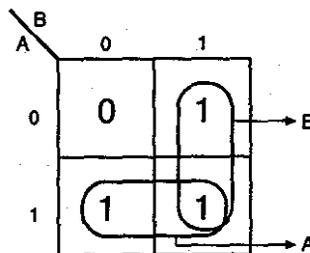


Fig 7.5

Using this Karnaugh map we get the simplified expression:

$X = A+B$. (You would have got the same answer if you had chosen to use the Boolean laws to simplify the expression.)

Draw the logic circuit: In this case only one simple logic gate is necessary – a 2-input OR gate.

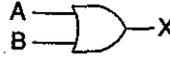


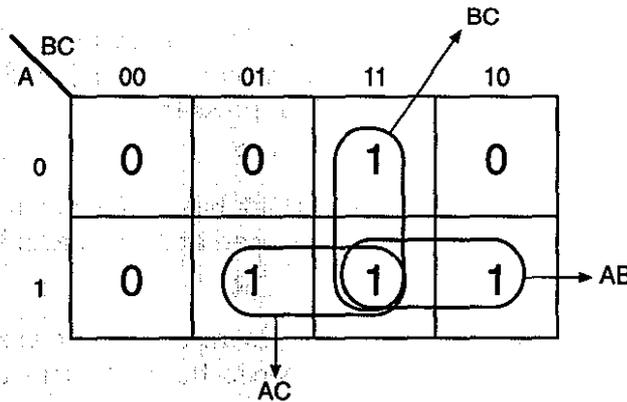
Fig 7.6

Example 7.2 In a factory a machine is run by three motors. If two or more of the motors develop a fault, the chief supervisor must be notified. Design a logic circuit that will monitor the functioning of the motors, and will turn on an indicator lamp for the supervisor should two or more of the motors develop a fault.

Solution Each of the motors can represent inputs to the logic circuit. The output will go to the indicator lamp. Let the motors be the input variables, A, B and C. Let the lamp be the output variable, X. If a motor is running smoothly, let the input be a 0. If the motor is faulty, the input must be a 1. If the output is 0, the lamp will stay off, but if the output is 1, the lamp will go on.

Truth table:

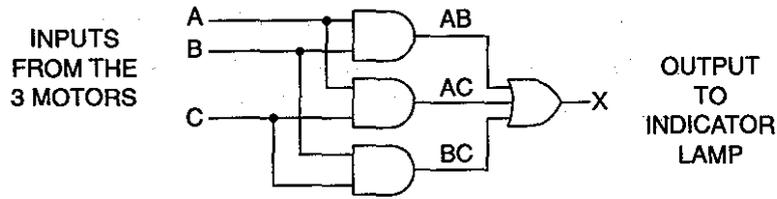
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Karnaugh map

Fig 7.7

Simplified Boolean expression: $X = AB + AC + BC$



Logic circuit
Fig 7.8

Activity 2



For each of the following exercises you do not need to show the steps in as great detail as shown in example 7.1 above.

- 1 An aeroplane has two wheels. Before the plane can land, a warning lamp must come on if the left wheel is down and the right wheel is up, or vice versa. The lamp must not turn on if both wheels are down. A separate alarm must sound if both wheels stay up. Design a logic circuit for the warning lamp and the alarm.
- 2 A lift has two doors and is driven by a drive motor. In the lift there is an alarm button that can be pressed by a person in the lift. The alarm bell of the lift must be activated under the following conditions:

Either

- the lift is stuck between floors (i.e. the drive motor is not running) and the alarm button has been pressed

OR

- the lift has arrived at a floor, but the doors do not open and the alarm bell is pressed

OR

- the doors of the lift are open when the lift is between floors. It does not matter whether the motor is running or not, or whether the alarm button is pressed or not.

Design a logic circuit for the alarm bell. The following input variables can be defined:

A: alarm button (pressed/not pressed)

- B: drive motor (not running/running)
- C: lift position (in-between floors/at floor)
- D: doors (open/closed)

4 The binary number system

At this point in the unit we are going to introduce a new concept, the **binary number system**. This concept may seem unrelated to logic circuit design, but it is in fact one of the main aspects of digital systems. Once you understand this concept, the rest of this course will be easier to understand.

In digital systems, information can only be expressed in terms of a high or low voltage level, or in numerical terms by a 1 or a 0. This 1 and 0 are the basis of the binary number system.

Before we look at the binary number system, let's look at a very familiar number system: the **decimal number system**. It is the number system you use every day.

The first number is 0. Then we count as follows: 1, 2, 3, 4, 5, 6, 7, 8, 9. You will notice that we use ten symbols. The next number is made up of the 1 and the 0. We call it ten, which is written as 10. The counting sequence then continues as follows: 11, 12, 13, 14, 15, 16, 17, 18, 19. To get the next number in the sequence we have to use the next symbol, 2, in front of the 0. This gives us 20, and the sequence continues as: 21, 22, 23, and so on until we reach 29. The next number will be 30, followed by 31, 32 and so on. This process continues until we reach 99. What is the next number? Of course, we start with a 1 and two 0s to get 100. This may seem very easy, but it is important to understand how the decimal system works because the same sort of counting process is applied in the binary number system.

In the **binary number system** there are only two symbols – the zero(0) and the one(1). The first number is a 0 and the second number is a 1. The third or next number is made up of the 1 and the 0, which is called 'one-zero' (not 'ten'), even though we write it as 10. The next number in the binary sequence is 11, or one-one. Can you work out the next number? You start with a 1 and two 0s, and get 100, which is read as 'one-zero-zero'. This is followed by 101, 110, 111, 1000, 1001, 1010 and so on.

The list below shows the sequence of binary numbers beginning at 0 and ending at 11111.

0 1 10 11 100 101 110 111 1000 1001 1010 1011
 1100 1101 1110 1111 10000 10001... 11111

4.1 Proper labelling in the different number systems

When you work with more than one number system, you could become confused when writing numbers. How will you know whether you are writing decimal or binary numbers (or even in another number system as you will discover in the Unit 8)? For example, is 10 a ten or a one-zero? You have to be able to differentiate between the different number systems. To do this we use subscripts. For the decimal system the subscript is 10 because the system is made up of ten numbers. The binary system has 2 as a subscript because the system is made up of two numbers. We say that the decimal system is a base ten system, while the binary system is a base two system.

The subscript is written immediately after the number, for example: 25_{10} 100_{10} 101_2 10_{10} 99_{10} 10_2 102_{10} 111_2

However, to make things easier we do not usually use the subscript (10) when we are writing decimal numbers. Therefore 101 will be one hundred and one, and not one-zero-one.

4.2 Relationship between the binary and the decimal number systems

Look at table 7.1. The table shows some decimal numbers and their binary equivalents. You will see that the bigger the decimal number, the more digits (or bits) are necessary to make up that binary number. (The word **bit** is a contraction of the words 'binary' and 'digit'. Each place in a binary number is called a 'bit'.)

Decimal	Binary	Decimal	Binary
0	0	11	1011
1	1	15	1111
2	10	16	10000
3	11	17	10001
4	100	20	10100
5	101	31	11111
6	110	32	100000
7	111	50	110010
8	1000	99	1100011
9	1001	100	1100100
10	1010	127	1111111

Table 7.1 Sample list of decimal and binary numbers

4.3 Converting decimal numbers into binary numbers

You need to follow a set of simple steps to convert a decimal number into its binary equivalent. These steps are:

- 1 Divide the number by 2. Make a note of the remainder, whether it is 1 or 0.
- 2 Divide the answer by 2. Again, make a note of the remainder.
- 3 Continue this division by 2 until you are left with $1 \div 2$ which is then equal to 0, with 1 as the remainder.
- 4 Write down the remainders from the last one to the first. This is the binary equivalent.

Here are a few examples to illustrate this conversion.

Example 7.3 Convert 13 to binary form.

Solution

$$\begin{array}{l}
 13 \div 2 = 6 \text{ remainder } 1 \text{ } \left. \begin{array}{l} \text{end} \\ \uparrow \\ \text{start} \end{array} \right| \\
 6 \div 2 = 3 \text{ remainder } 0 \\
 3 \div 2 = 1 \text{ remainder } 1 \\
 1 \div 2 = 0 \text{ remainder } 1
 \end{array}$$

Now write the remainders from the last one to the first one: 1101.

The binary equivalent of 13 is 1101_2 .

This series of steps could also be set out like this:

$$\begin{array}{r|l}
 2 & 13 \\
 \hline
 2 & 6 \text{ rem } 1 \\
 2 & 3 \text{ rem } 0 \\
 2 & 1 \text{ rem } 1 \\
 2 & 0 \text{ rem } 1 \\
 \hline
 & \uparrow
 \end{array}$$

Example 7.4 Convert 27 to binary form.

Solution

2	27	
2	13	rem 1
2	6	rem 1
2	3	rem 0
2	1	rem 1
	0	rem 1

Writing the remainders from the last one to the first one we get 11011.

The binary equivalent of 27 is 11011₂.

Example 7.5 Convert 100 to binary form.

Solution

2	100	
2	50	rem 0
2	25	rem 0
2	12	rem 1
2	6	rem 0
2	3	rem 0
2	1	rem 1
	0	rem 1

Write the remainders from the last one to the first. We get: 1100100.

The binary equivalent of 100 is 1100100₂.

4.4 Converting binary numbers into decimal numbers

Start on the extreme right of the binary number (i.e. at the least significant bit (LSB)) and allocate a power of 2 to each digit in the binary number. You will start with 2⁰, then 2¹, 2² and so on. Now calculate the power of 2 under each 1 in the binary number. To work out what the decimal number is, you will add up only the powers of 2 which were allocated to the 1s in the binary number. Do not add up the values allocated to the 0s.

If you look at a few examples, this method will be a lot easier to understand.

Example 7.6 Convert 111_2 to a decimal number.

Solution

1	1	1
↓	↓	↓
2^2	2^1	2^0
$4 + 2 + 1 = 7$		

Therefore $111_2 = 7$

Example 7.7 Convert 11001_2 to a decimal number.

Solution

1	1	0	0	1
↓	↓			↓
2^4	2^3			2^0
$16 + 8 + 1 = 25$				

Therefore $11001_2 = 25$

Example 7.8 Convert 1000000_2 to a decimal number.

Solution

1	0	0	0	0	0	0
↓						
2^6						

Therefore $1000000_2 = 64$

To check if your answer is correct you can convert the decimal number you work out back to the binary form – and vice versa.

Activity 3



- 1 Convert the following decimal numbers into binary form.
44 356 1010 3 151
(You can check that your answers are correct by converting the binary numbers back into decimal form.)
- 2 A decimal number is represented in binary by four bits A, B, C and D – with A being the most significant bit (MSB). Design a logic circuit of which the output will be high only for numbers between 9 and 13 inclusive (i.e. including 9 and 13).

Besides the decimal and binary number systems, there is another very important number system that is widely used in digital and computer systems: the hexadecimal number system. You will learn about this number system in Unit 8.

Extension task Fractional decimal numbers may also be represented in binary form.

For example: $14,125 = 1110,001_2$

Consult the recommended texts mentioned in Appendix 5 at the end of this book and work out how these fractional decimal numbers may be converted to binary form, and also how fractional binary numbers may be converted to decimal form.

5 The Exclusive-OR and the Exclusive-NOR gates

You have so far studied five logic gates, namely the AND, OR, NOT, NAND and NOR gates. We will now look at two additional logic gates, called the Exclusive-OR and the Exclusive-NOR gates.

5.1 The Exclusive-OR gate

Consider the following truth table:

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

From this table we can derive the following Boolean expression: $X = \bar{A}B + A\bar{B}$. This expression cannot be simplified.

The logic circuit for this truth table is shown in Fig 7.9 below.

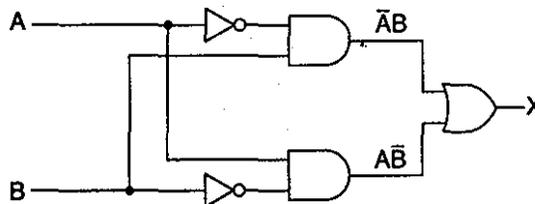


Fig 7.9

This logic circuit is made up of five logic gates. How many and which ICs do you think you would have to use to implement this logic circuit for testing? There is a single logic gate that can perform the function represented in the truth table shown above. This logic gate is known as the **Exclusive-OR gate**. The logic symbol for this gate is shown in Fig 7.10.

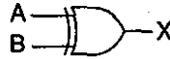


Fig 7.10

The Boolean expression for this gate is: $X = A \oplus B$. This is read as 'X equals A Exclusive-OR B'.

The truth table for this gate will be:

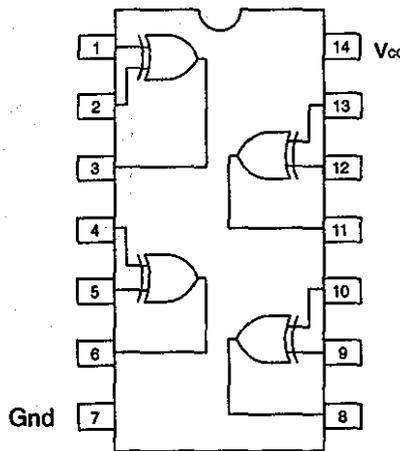
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



An easy way of describing the Exclusive-OR function is: The output is high only when the inputs are not the same. When the inputs are the same, i.e. 0, 0 or 1, 1 then the output is low.

The Exclusive-OR function, $A \oplus B$ is equivalent to the Boolean expression: $\overline{A}B + A\overline{B}$.

To implement this Exclusive-OR gate for testing purposes you should use the IC shown in Fig 7.12. This is a quad two-input Exclusive-OR package or IC and its number is 7486.



7486 quad 2-input
Exclusive-OR package

Fig 7.11

5.2 The Exclusive-NOR gate

This logic function or gate is the inverse of the Exclusive-OR gate. It is represented by this truth table:

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

The logic symbol for the Exclusive-NOR gate is:



Fig 7.12

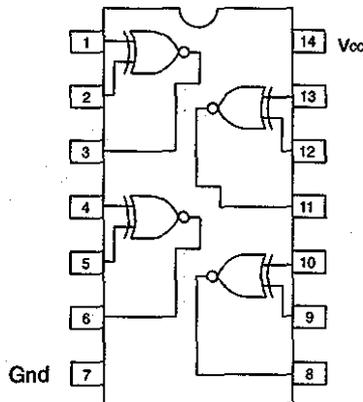
The Boolean expression is: $X = \overline{A \oplus B}$

The equivalent logic expression is: $X = \overline{A} \overline{B} + AB$



An easy way of describing the Exclusive-NOR function is: The output is high only when the inputs are the same. i.e. 0, 0 or 1, 1. When the inputs are not the same, then the output is low.

To implement this Exclusive-NOR gate for testing purposes you should use the IC shown in Fig 7.13. This is a quad two-input exclusive-NOR package or IC and its number is 74810.



74810 quad 2-input
Exclusive-NOR package

Fig 7.13

Activity 4

- 1 What is an Exclusive-OR gate?
- 2 What is an Exclusive-NOR gate?
- 3 Using the basic logic gates (i.e. AND, OR and NOT) draw the equivalent logic circuit for the following function:
 $Z = \overline{J} \oplus \overline{K}$
- 4 Which type of exclusive gate can be used for the warning lamp in the aeroplane example in Activity 2?
- 5 Write down the truth table for the logic circuit shown in Fig 7.14a.

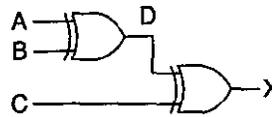


Fig 7.14a

- 6 The following waveforms are applied to the circuit in Fig 7.14a. Determine the output waveform.

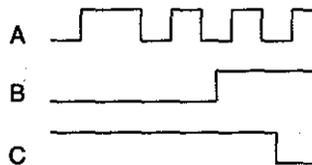


Fig 7.14b

Activity 5

Implement (i.e. wire up on the test board) and test the Exclusive-OR and the Exclusive-NOR gates. Use the 7486 and the 74810 ICs. If you cannot remember how to do this, please refresh your memory by referring to Unit 2, Activity 3.

6 Comparing binary quantities

Because you are very familiar with the decimal number system, it is very easy for you to look at two numbers and then work out which number is bigger or greater. However, when looking at two binary numbers it may not be so obvious which one is greater than the other. For example, which binary number is greater: 10110_2 or 11010_2 ? One way of finding out is to convert each number to its decimal form. It is then easy to see which number is greater.

$$10110_2 = 22 \quad 11010_2 = 34$$

34 is greater than 22; therefore 11010_2 is greater than 10110_2

However, it is also relatively easy to determine which number is greater by examining the binary numbers. This can be done by following these steps:

- 1 Rewrite both the numbers to be compared with the same number of bits.
- 2 Start on the side of the most significant bit (MSB), the left-hand side.
- 3 Compare the corresponding bits:
 - the number with a 1 is greater than the number with a 0
 - if both the bits are the same (i.e. a 1 or a 0) then examine the next bit in each of the two numbers
 - again, the number with a 1 is greater than the number with a 0.
- 4 This process of bit-by-bit comparison must be continued until you have determined which number is greater.

Let's study some examples where we can apply these steps.

Example 7.9 Compare 100_2 to 110_2 .

Solution Both binary numbers are already written with three bits.

100_2 The MSB is a 1.

110_2 The MSB is also a 1.

Examine the next bit in each of the two numbers.

100_2 The next bit is a 0.

110_2 The next bit is a 1.

Therefore, 110_2 is greater than 100_2 .

Example 7.10 Compare 10110_2 to 11010_2 .

Solution Both binary numbers are already written with five bits.

10110_2 The MSB is a 1.

11010_2 The MSB is also a 1.

Examine the next bit in each of the two numbers.

10110_2 The next bit is a 0.

11010_2 The next bit is a 1.

Therefore, 11010_2 is greater than 10110_2 .

Example 7.11 Compare 11110_2 to 1011_2 .

Solution 11110_2 is written with five bits and 1011_2 is written with four bits. You will have to rewrite both numbers with five bits: 1011_2 becomes 01011_2 .

Now compare 11110_2 to 01011_2 .

11110_2 The MSB is a 1.

01011_2 The MSB is a 0.

Therefore 11110_2 is greater than 01011_2 .

Example 7.12 Compare 01100_2 to 10000_2 .

Solution Both binary numbers are already written with five bits.

01100_2 The MSB is a 0.

10000_2 The MSB is a 1.

Therefore 10000_2 is greater than 01100_2 .

In all cases you might like to check your answer by converting each number to its decimal form.

Activity 6



1 Compare the following pairs of binary numbers:

1.1 1010_2 and 1001_2

1.2 010101_2 and 010100_2

1.3 1100_2 and 10100_2

2 Examine the following list of binary numbers:

10000_2 11011_2 01000_2 1010_2 0111_2 01010_2

2.1 Are any of the numbers equal?

2.2 Arrange the numbers in order from the smallest to the biggest.

7 Comparator circuits

An electronic circuit that compares two quantities is called a **comparator**. In digital systems a circuit can be designed to compare two binary quantities or numbers, and to indicate whether the quantities are equal or whether one is greater or less than the other.

We will begin by comparing just two binary bits, in other words: one single bit to another single bit. Later we will compare numbers which are made up of more than just one bit, for example, numbers with two or three bits. We will also discuss a special comparator IC.

7.1 Comparing two binary bits

We are going to try to design logic circuits that will compare two binary bits, and indicate whether the two bits are equal, or whether one is greater or less than the other.

First work out whether the two bits are equal.

The two bits will be inputs to a logic circuit. The output will indicate whether the two bits are equal or not. Let's call the two bits A and B. The output can be X. A and B can be a 0 or a 1. X can be a high or a 1 if $A = B$. All this information can be summarised in a truth table.

A	B	X	
0	0	1	(A and B are both 0. Therefore X must be a 1.)
0	1	0	(A is not the same as B. Therefore X must be a 0.)
1	0	0	(Again, A is not the same as B. So X must be a 0.)
1	1	1	(A and B are both 1. Therefore X must be a 1.)

The simplified Boolean expression will be: $X = \bar{A}\bar{B} + AB$.

The logic circuit will therefore be as shown in Fig 7.15.

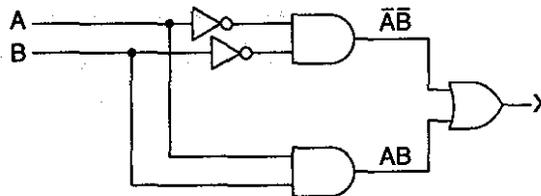


Fig 7.15

X will be high when A is equal to B.

You will recall that $X = \bar{A}\bar{B} + AB$ is equivalent to $\overline{A \oplus B}$. We can therefore replace the logic circuit shown in Fig 7.15 with a single Exclusive-NOR gate as shown in Fig 7.16.



Fig 7.16

Now work out whether one bit is greater than the other.

In this case the two bit will also be inputs to a logic circuit. The output of the logic circuit must indicate if the one bit is greater than the other bit. The input bits can again be A and B. However, we will call this output Y. Y must be high or a 1 if A is greater than B. In all other cases Y must be 0.

Summarising this information in a truth table.

A	B	Y	
0	0	0	(A $\not>$ B. Therefore Y must be 0.)
0	1	0	(Again, A $\not>$ B. Therefore Y must be 0.)
1	0	1	(A is 1 and B is 0. A > B. So Y must be 1.)
1	1	0	(Again, A $\not>$ B. Therefore Y must be 0.)

In this case we get a simple Boolean expression: $Y = A\bar{B}$.

The logic circuit is shown in Fig 7.17 below.

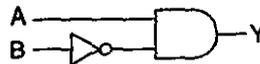


Fig 7.17

Y will be high when A is greater than B.

If we want to, we can combine the two circuits (Fig 7.16 and 7.17) which we have just designed, into one diagram (Fig 7.18).

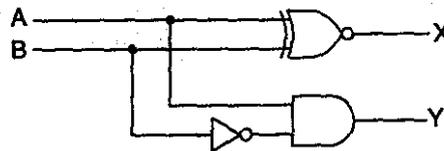


Fig 7.18

X will be high when A is equal to B. Y will be high when A is greater than B.

Activity 7



Design a logic circuit that will compare two binary bits A and B. The output Z must indicate that A is less than B.

7.2 Comparing binary numbers

In the previous section we designed logic circuits that compared single binary bits. In this section we will go one step further and design circuits that will compare binary numbers. We will start by looking at simple two-bit binary numbers. We will then look at circuits which can compare binary numbers which are made up of more than two bits.

We will first compare two-bit binary numbers. Consider a two-bit binary number A. Because it has two bits we will label the two bits A1 and A0. (A1 will be the most significant bit.)

We will call the second two-bit binary number B. We will label these bits B1 and B0. (B1 will be the most significant bit.)

Designing this comparator circuit is a two-step process:

- ◇ Step 1: Circuit for A is equal to B
- ◇ Step 2: Circuit for A is greater than B.

Step 1

You have to work out when the number A will be equal to the number B.

We can now use the logical reasoning technique mentioned in section 3 of this unit. This method simply means that you must think logically! Doing so, we can say that number A will be equal to number B when: A1 is equal to B1 AND A0 is equal to B0. Have you come across a circuit that can look at two bits and indicate whether the two bits are equal? Yes, you have! Look at Fig 7.16 again.

In Fig 7.19a we have redrawn the circuit with A1 and B1 as the inputs. This circuit compares A1 to B1.

Circuit 1

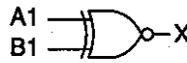


Fig 7.19a

X will be high when A1 is equal to B1.

In Fig 7.19b we have redrawn the circuit with A0 and B0 as inputs. This circuit compares A0 to B0.

Circuit 2



Fig 7.19b

Y will be high when A0 is equal to B0.

Circuit 1 indicates whether A1 is equal to B1. Circuit 2 indicates whether A0 is equal to B0.

For the two-bit binary numbers A1 and A0 to be equal to B1 and B0, we stated that: A1 must be equal to B1 AND A0 must be equal to B0. This means that the AND function or gate must be used to combine the output from circuit 1 to the output from circuit 2.

We can therefore draw the following logic circuit.

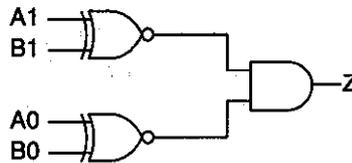


Fig 7.20

Z will be high when both A1 is equal to B1 AND A0 is equal to B0.

Before we continue with the design of other comparator logic circuits, let's prove that the design is correct by checking the circuit with some two-bit binary numbers.

Case 1: Compare 10_2 to 11_2 .

10_2 Let A1 be the 1 and A0 be the 0.

11_2 Therefore B1 will be the 1 and B0 will also be a 1.

When you apply the bits to the logic circuit shown in Fig 7.20 above:

- ◇ The output of the first Exclusive-NOR gate will be a 0. (Use the Exclusive-NOR truth table shown above Fig 7.12.)
- ◇ The output of the second Exclusive-NOR gate will be a 1.

Therefore, the output of the AND gate must be a 0. This 0 or low level indicates that 10_2 is not equal to 11_2 .

Case 2: Compare 01_2 to 01_2 .

01_2 A1 is 0 and A0 is 1.

01_2 Similarly, B1 is 0 and B0 is 1.

When you apply the bits to the logic circuit shown in Fig 7.20 above:

- ◇ The output of the first Exclusive-NOR gate will be a 1.
- ◇ The output of the second Exclusive-NOR gate will also be a 1.

Therefore, the output of the AND gate must be a 1. This 1 or high level indicates that 01_2 is equal to 01_2 .

Activity 8



Try designing a logic circuit that can compare two three-bit binary numbers and can indicate if the two numbers are equal.

Step 2

You have to work out when the number A will be greater than the number B.

Remember, we are still using the same symbols for the two-bit binary numbers, i.e. the first number is A1 and A0 and the second number is B1 and B0 (with A1 and B1 being the MSBs).

To work out when the number A will be greater than the number B, you will have to recall the techniques learnt in section 7 of this unit when we compared binary numbers.

From the technique it will appear that number A will be greater than number B if:

A1 is greater than B1

OR

A1 is equal to B1 AND A0 is greater than B0.

Now try to implement the above condition with logic circuits that you are familiar with. Have you come across a circuit that can look at two bits and indicate whether one bit is greater than the other bit? Yes! Look at Fig 7.17 again. The logic circuit is redrawn in Fig 7.21 with A1 and B1 as inputs.

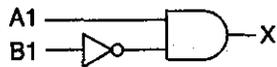


Fig 7.21

X will be high when A1 is greater than B1.

Now draw a circuit that will check if two bits (A1 and B1) are equal:



Fig 7.22

Y will be high when A1 is equal to B1.

Draw a circuit that will check if one bit, A0, is greater than the other bit, B0:

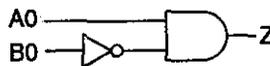


Fig 7.23

Z will be high when A0 is greater than B0.

You will recall that we said that number A will be greater than number B if: A1 is greater than B1 OR A1 is equal to B1 AND A0 is greater than B0.

We will therefore have to combine the three circuits from Fig 7.21, 7.22, and 7.23 into one circuit by using the AND and the OR gate, as in Fig 7.24.

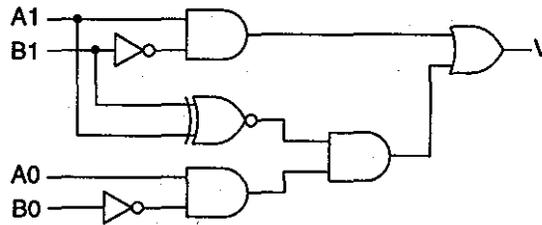


Fig 7.24

V will be high when:

- ◇ A1 is greater than B1 OR
- ◇ A1 is equal to B1 AND A0 is greater than B0.

Once again, let's prove that the design is correct by checking the circuit with some two-bit binary numbers.

Case 1: Compare 10_2 to 11_2 .

10_2 Let A1 be 1 and let A0 be 0.

11_2 Therefore B1 will be 1 and B0 will also be a 1.

Apply the bits to the logic circuit shown in Fig 7.24 above.

The circuit is redrawn (in Fig 7.25 below) showing the inputs and the logic levels at the output of each gate. As you can see the final output is a low or 0. This means that 10_2 is not greater than 11_2 .

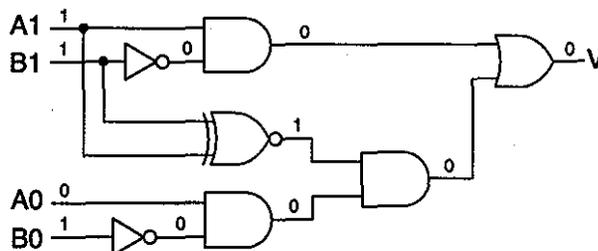


Fig 7.25

Case 2: Compare 10_2 to 10_2 .

10_2 Let A1 be 1 and let A0 be 0.

10_2 Therefore B1 will be 1 and B0 will be a 0.

Apply the bits to the logic circuit shown in Fig 7.24 above.

The circuit is redrawn (in Fig 7.26 below) showing the inputs and the logic levels at the output of each gate. As we can see the final output is a low or 0. This means that 10_2 is not greater than 10_2 .

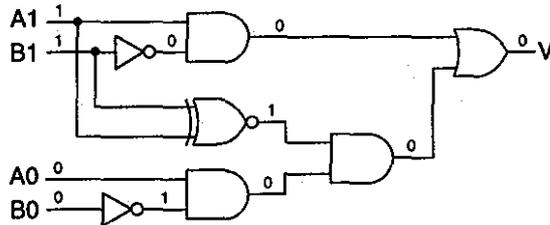


Fig 7.26

Activity 9



- 1 Design a comparator circuit that will indicate if A is less than B. (Remember that A and B are two-bit binary numbers.)
- 2 If A and B were three-bit binary numbers, design a circuit that will compare A and B and indicate whether A is greater than B.



[Note: In section 3 above we said that the design of a logic circuit may be done using truth tables or using logical reasoning. The truth table method was done in great detail in section 3.1. The process of logical reasoning was applied when comparator circuits were designed.]

Activity 10



- 1 What ICs would be required to test the comparator logic circuit shown in Fig 7.24 above?
- 2 If the Exclusive-NOR gate IC was not available, which ICs could be used instead? Explain your choice.

- 3 Implement (i.e. wire up on the test board) and test the comparator logic circuit shown in Fig 7.24 above. Remember to redraw the circuit with the IC pin numbers before wiring up the circuit.

7.3 Using a special comparator IC

Up to this point you have used logic gates (like the AND, NOT, Exclusive-OR etc.) to design logic circuits for various applications. However, it is important to note that in digital systems there are various advanced devices or ICs that you can use. These devices can perform the function of entire logic circuits in a single IC package. For example, the 7485 IC shown in Fig 7.27b below is a 4-bit comparator. This single IC can compare two binary numbers of up to four-bits each, and indicate whether one number is equal to, or greater than, or less than the other number.

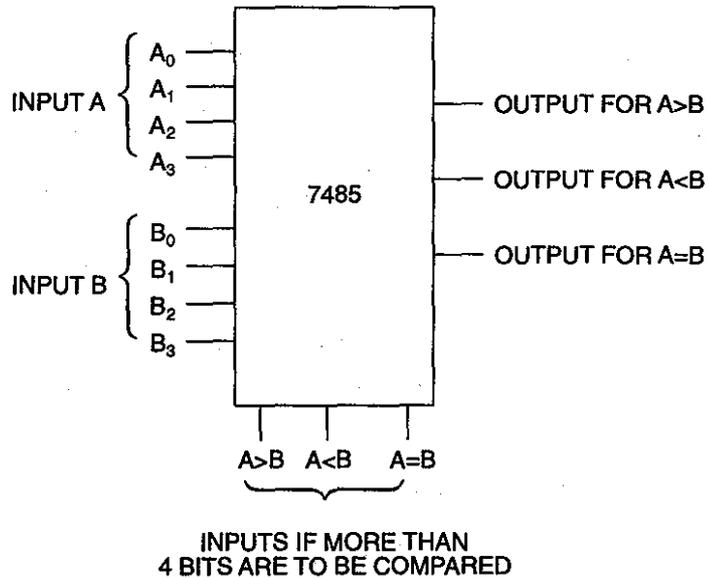


Fig 7.27a 4-bit comparator: logic symbol

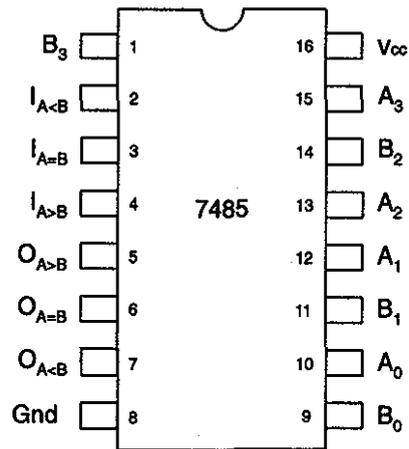


Fig 7.27b 4-bit comparator: 7485 IC pin out

The complete details of any electronic component, for example, a digital IC, are usually found in the device manufacturer's data book. A few pages taken from a digital IC data book are shown in Appendix 2 at the end of this book. Data sheets contain full details regarding the device, for example, the pin outs, the function of the device, how the device operates and the electrical characteristics of the device. The information contained in the data sheets is very useful in the designing, testing and troubleshooting of electronic circuits.

So instead of designing a 4-bit binary comparator circuit using a number of logic gates or ICs, the whole logic circuit could be implemented using a single IC – the 7485. Using this kind of IC has a number of advantages:

- ◇ less time is required for the design
- ◇ fewer components are necessary
- ◇ less space is used on the circuit board
- ◇ the circuit uses less power
- ◇ it is easier and quicker to test and troubleshoot the circuit.

In the next two units you will be introduced to many other types of digital ICs.

8 Summary

In this unit we described two techniques you can use to design combinational logic circuits, namely: designing with the aid of truth tables, and designing by using logical reasoning. The important concept of the binary number system was also introduced in this unit. To expand our knowledge of logic gates, two new gates – the Exclusive-OR and the Exclusive-NOR – were presented. There are many types of combinational logic circuits; in this unit the comparator circuits were explained and designed. The binary number system and the exclusive gates were applied in the design of comparator circuits. The unit ended with a brief look at the concept of using advanced digital ICs and component data sheets.

In the next unit you will learn about other number systems and about more combinational logic circuits.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Name the two categories of digital circuits.
- 2 Draw a combinational logic circuit with two inputs and two outputs.
- 3 In this unit two generally used digital design techniques were applied. Name the techniques.
- 4 List the steps used to design logic circuits when using truth table as an aid.
- 5 A Boolean expression may be obtained from the truth table when designing a logic circuit. What two methods may be used to simplify the Boolean expression?
- 6 After a logic circuit has been designed, how can you test whether the design will function correctly?
- 7 A certain application resulted in the following design:

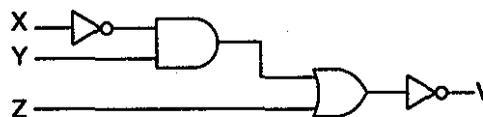


Fig 7.28

If only NAND gate ICs were available, how could you implement and test this circuit?

- 8 A security company has two safes: a large safe and a small safe. The company has four guards. Each guard has a different key for the safes. When all four keys are inserted into the locks only the large safe will open. For the small safe to open, at least any two keys must be inserted into the locks. Design a logic circuit to open the two safes.

Hint: This logic circuit must have two outputs – one for each safe.

- 9 A panel of four judges has been asked to test a series of new food products. Each judge may indicate whether he or she likes or dislikes the product. The final decision of all the judges together may therefore be: product is good, product is not good, or there may be a tie. This final decision can be indicated on a set of three lamps: good, not good, tie. Design a logic circuit connected to each of the three lamps.
- 10 Design a logic circuit in which the output will be high when any of the input 4-bit binary numbers are greater than or equal to 5 (i.e. 0101_2 in binary).
- 11 A large warehouse has two entrances – each at the opposite end of the warehouse. At each entrance there is a light switch. The lights in the warehouse can be switched on or off by using either of the switches. This means that a person could enter the warehouse from the one entrance, switch on the lights, walk through the warehouse to the other side, switch off the lights and leave by the other entrance. Design a logic circuit to control the warehouse lights.

- 12.1 Draw the truth table and write down the Boolean expression for the logic circuit shown in Fig 7.29.



Fig 7.29

12.2 Use this truth table to design a simplified logic circuit. This logic circuit will be equivalent to the logic circuit shown in Fig 7.29 above.

13.1 Arrange the following binary numbers in order of increasing magnitude.

1001_2 10011_2 01001_2 1000000_2 11111_2
 01011_2

13.2 Convert each of the numbers in the above list to a decimal number and check your answers.

13.3 Design a comparator circuit that will compare the first two numbers in the above list and indicate whether:

- the one number is equal to the other one
- the one number is greater than the other one.

14 Design a logic circuit that will compare two 4-bit binary numbers A and B and indicate whether A is less than B.

Answers to activities

Activity 1

1 combinational and sequential

2 You may draw any type of logic circuit featuring any of the logic gates you have encountered so far, namely, AND, OR, NOT, NAND, NOR. Here is a typical example:

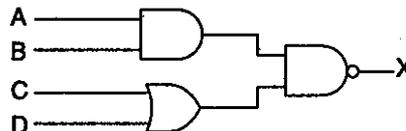


Fig 7.30

3

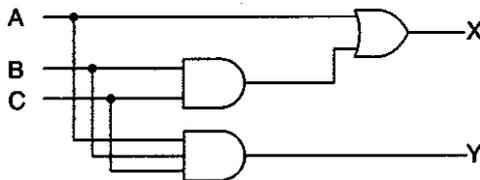


Fig 7.31

4 The circuit has 3 inputs: A, B, C, and 2 outputs: X, Y.

Activity 2

1 Input variables: Right wheel – A

Left wheel – B

Output variable: Warning lamp – X

Alarm –Y

Logic states

Wheel up: 0; wheel down: 1; lamp off: 0; lamp on: 1; alarm

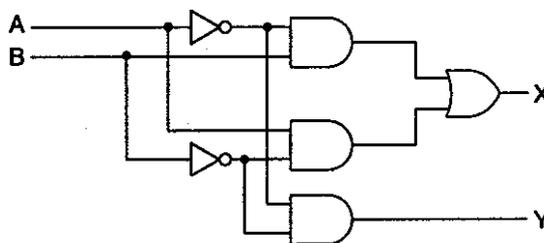
off: 0; alarm on: 1

Truth table:

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

Boolean expression: $X = \bar{A}B + A\bar{B}$ (cannot be simplified)

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



Logic circuit

Fig 7.32

- 2 Input variables: Alarm button – A
- Drive motor – B
- Lift position – C
- Doors – D
- Output variable: Alarm bell – X

We have assumed these logic states

Alarm button: pressed: 1; not pressed: 0

Drive motor: not running: 1; running: 0

Lift position: in-between floors: 1; at floor: 0

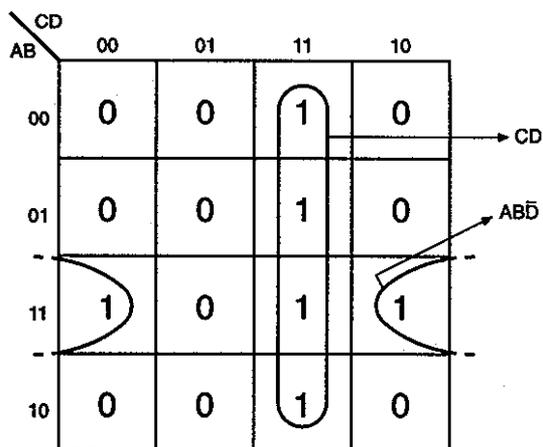
Doors: open: 1; closed: 0

Alarm bell: on: 1; off: 0

(You may have chosen different logic states. Either way, your final answer should be of the same form as this one.)

Truth table:

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



Karnaugh map
Fig 7.33

Simplified Boolean expression:

$$X = AB\bar{D} + CD$$

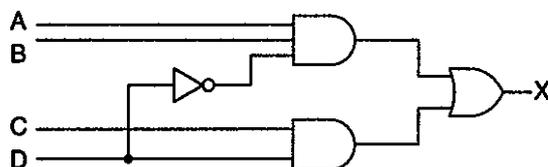


Fig 7.34 Logic circuit

Activity 3

1

2	44	
2	22	rem 0
2	11	rem 0
2	5	rem 1
2	2	rem 1
2	1	rem 0
2	0	rem 1

$44 = 101100_2$

2	356	
2	178	rem 0
2	89	rem 0
2	44	rem 1
2	22	rem 0
2	11	rem 0
2	5	rem 1
2	2	rem 1
2	1	rem 0
2	0	rem 1

$356 = 101100100_2$

2	1010	
2	505	rem 0
2	252	rem 1
2	126	rem 0
2	63	rem 0
2	31	rem 1
2	15	rem 1
2	7	rem 1
2	3	rem 1
2	1	rem 1
2	0	rem 1

$1010 = 1111110010_2$

$$\begin{array}{r|l}
 2 & 3 \\
 \hline
 2 & \underline{1} \text{ rem } 1 \\
 & 0 \text{ rem } 1
 \end{array}
 \quad \uparrow
 \quad 3 = 11_2$$

$$\begin{array}{r|l}
 2 & 151 \\
 \hline
 2 & \underline{75} \text{ rem } 1 \\
 2 & \underline{37} \text{ rem } 1 \\
 2 & \underline{18} \text{ rem } 1 \\
 2 & \underline{9} \text{ rem } 0 \\
 2 & \underline{4} \text{ rem } 1 \\
 2 & \underline{2} \text{ rem } 0 \\
 2 & \underline{1} \text{ rem } 0 \\
 & 0 \text{ rem } 1
 \end{array}
 \quad \uparrow
 \quad 151 = 10010111_2$$

2 Input variables:

A – MSB of the binary number

B – second bit of the binary number

C – third bit of the binary number

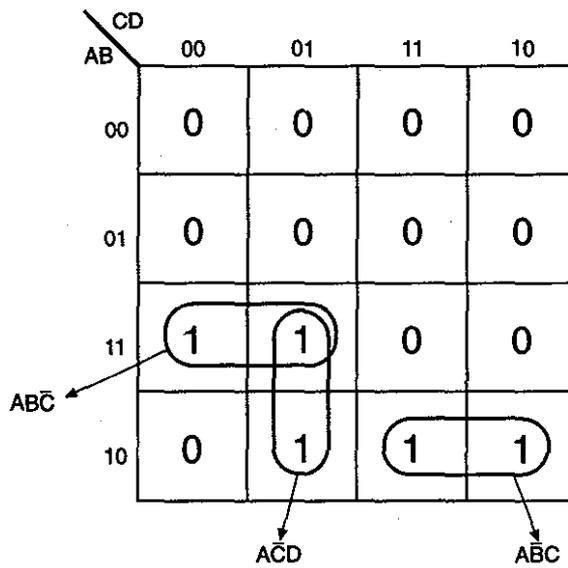
D – fourth bit of the binary number

Output variable: X

A, B, C and D will take on values from 0000, 0001... up to 1111. The truth table below shows when X will be high.

Decimal number	Binary equivalent				Output X
	A	B	C	D	
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

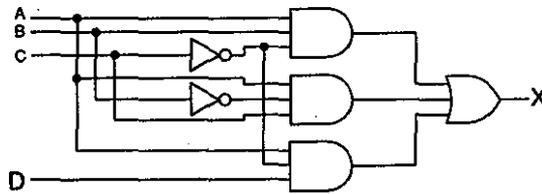
X will be high for the decimal numbers from 9 to 13.



Karnaugh map

Fig 7.35

Simplified Boolean expression: $X = ABC\bar{C} + A\bar{B}C + A\bar{C}D$



Logic circuit

Fig 7.36

Activity 4

- 1 **Exclusive-OR:** This is a logic gate in which the output is high when the inputs are not the same.
- 2 **Exclusive-NOR:** This is a logic gate in which the output is high when the inputs are the same.
- 3 $Z = J \oplus K$
 $= \bar{J}\bar{K} + JK$

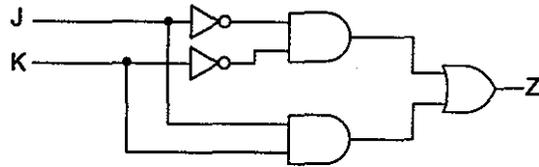


Fig 7.37

4 The Exclusive-OR gate

5

A	B	C	D	X
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

6

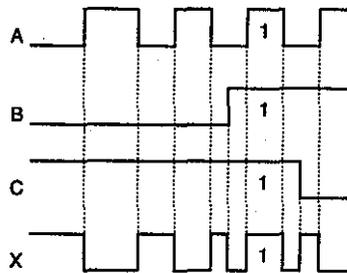


Fig 7.38

Activity 6

1.1 Both binary numbers are written with four bits.

1010₂ The MSB is a 1.

1001₂ The MSB is also a 1.

Therefore, examine the next bit.

1010₂ The next bit is a 0.

1001₂ The next bit is also a 0.

Therefore, examine the next bit.

1010₂ The next bit is a 1.

1001₂ The next bit is a 0.

Therefore, 1010₂ is greater than 1001₂.

1.2 Both binary numbers are written with six bits.010101₂ The MSB is a 0.010100₂ The MSB is also a 0.

Therefore, examine the next bit.

010101₂ The next bit is a 1.010100₂ The next bit is also a 1.

Therefore, examine the next bit.

010101₂ Next bit is a 0.010100₂ Next bit is also a 0.

Continue this comparison until you come to the last bit in each number.

010101₂ The last bit is a 1.010100₂ The last bit is a 0.Therefore, 010101₂ is greater than 010100₂.**1.3** We have to write both binary numbers with the same number of bits, namely five. The first number is written as 01100₂. The second number is 10100₂.01100₂ The MSB is a 0.10100₂ The MSB is a 1.Therefore, 10100₂ is greater than 1100₂.**2** The quick and easy way to do this exercise is to convert each binary number to its decimal equivalent.

$$10000_2 = 16$$

$$11011_2 = 16 + 8 + 2 + 1 = 27$$

$$01000_2 = 8$$

$$1010_2 = 8 + 2 = 10$$

$$0111_2 = 4 + 2 + 1 = 7$$

$$01010_2 = 8 + 2 = 10$$

2.1 1010₂ is equal to 01010₂

2.2	7	8	10
	or	0111 ₂	01000 ₂ 1010 ₂
		10	16 27
	or	01010 ₂	10000 ₂ 11011 ₂

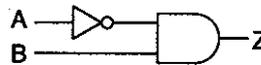
Activity 7

Truth table:

A	B	Z
0	0	0
0	1	1
1	0	0
1	1	0

←(A is less than B.
Thus Z is high.)

Boolean expression: $Z = \bar{A}B$



Logic circuit
Fig 7.39

Activity 8

Let the two binary numbers A and B be represented as follows: A₂, A₁, A₀ and B₂, B₁, B₀ – with A₂ and B₂ being the MSBs.

A will be equal to B when: A₂ is equal to B₂ AND A₁ is equal to B₁ AND A₀ is equal to B₀.

To check whether A₂ is equal to B₂ we can use the following circuit:

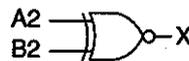


Fig 7.40a

We can do the same with A₁ and B₁, and A₀ and B₀.



Fig 7.40b



Fig 7.40c

These three gates must then be combined using the AND function or AND gate.

The final circuit will look like Fig 7.41.

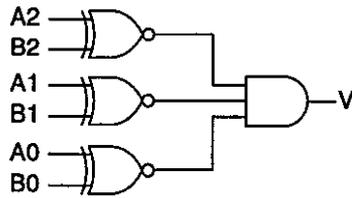


Fig 7.41

Activity 9

- Let the two binary numbers A and B be represented by A1 and A0, and B1 and B0 – with A1 and B1 being the MSBs.

A will be less than B when: A1 is less than B1 OR when A1 is equal to B1 AND A0 is less than B0.

To check whether A1 is less than B1 we can use the following circuit:

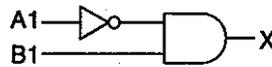


Fig 7.42a

To check whether A1 is equal to B1 we can use the following circuit:



Fig 7.42b

To check whether A0 is less than B0 we can use the circuit in Fig 7.42c.

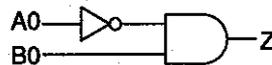


Fig 7.42c

These three circuits must then be combined into the following logic circuit:

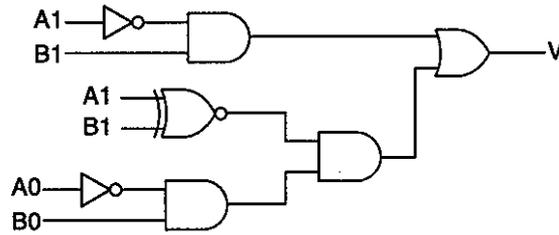


Fig 7.43

- 2 The logic circuit in Fig 7.44 can be designed using the same logical reasoning technique that was described in section 7.2 of this unit.

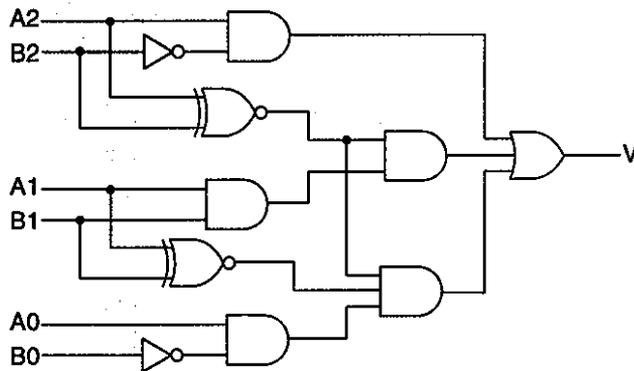


Fig 7.44

Activity 10

- 1 The circuit consists of:
- one 2-input OR gate
 - three 2-input AND gates
 - one 2-input Exclusive-NOR gate and
 - two NOT gates.

Therefore the following ICs can be used:

- 7432 (Quad 2-input OR gate)
- 7408 (Quad 2-input AND gates)
- 74810 (Quad 2-input Exclusive-NOR gate)
- 7404 (hex INVERTER).

2 Solution 1

Use the Exclusive-OR gate IC together with the INVERTER IC. (See Fig 7.45 below).

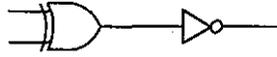


Fig 7.45

Solution 2

Replace the Exclusive-NOR gate with the combination of the basic logic gates - shown below in Fig 7.46.

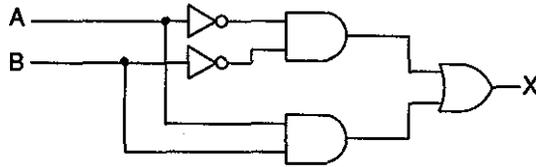


Fig 7.46

Application of combinational logic

Study objectives

After studying this unit, you should be able to:

- ◇ describe and use the hexadecimal and binary-coded decimal (BCD) number systems
- ◇ convert hexadecimal to binary and decimal and vice versa
- ◇ convert binary-coded decimal to binary and decimal and vice versa
- ◇ perform arithmetic addition using binary and hexadecimal numbers
- ◇ define, develop and use a half adder
- ◇ define, develop and use a full adder
- ◇ explain the operation and use of decoders and encoders
- ◇ explain the operation and use of multiplexers and demultiplexers.

1 Introduction

In this unit you will be studying other numbering systems as used in digital electronics. We will then look at how the adder, which is the basic arithmetic circuit, performs the addition of numbers. You will then look at circuits that perform number system conversions. The unit ends with a look at multiplexer and demultiplexer circuits.

2 Number systems

In Unit 7 you learnt about the decimal and binary number systems. As you know by now, digital electronic circuits make use of the binary number system. This system is used because it uses only two digits, 1 and 0 and can be easily represented with electrical voltage levels. In this section you will find out about a few other number systems.

2.1 Hexadecimal number system

The hexadecimal number system is used to represent binary numbers in a form which is easier to read. Most computers tend to use word lengths which are multiples of four bits, for example, 8, 16 and 32 bit microprocessors. This has led to the widespread use of the hexadecimal number system, or base sixteen system. The sixteen symbols used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Starting at zero we can count up to fifteen items before exhausting all the hexadecimal symbols. To represent the sixteenth item, a second digit is needed. We therefore reuse one of the elementary symbols.

Comparing decimal, binary and hexadecimal numbers, we have:

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
⋮	⋮	⋮
30	011110	1E
31	011111	1F
32	100000	20
33	100001	21
⋮	⋮	⋮
256	10000000	100

Example 8.1 Count from CF8 to D00 in hexadecimal.

Solution CF8
CF9
CFA (after 9 comes A in hexadecimal)
CFB
CFC
CFD
CFE
CFF (first 2 columns full: both change to 0, C changes to D)
D00

Activity 1 Write the hexadecimal numbers from EF_{16} to 103_{16} .



Hexadecimal to decimal conversion

Here we apply the same techniques as was used in Unit 7. Look at the following example.

Example 8.2 Convert CF8 hexadecimal to decimal.

Solution Hexadecimal CF8 = $(C \times 16^2) + (F \times 16^1) + (8 \times 16^0)$
= $(12 \times 256) + (15 \times 16) + (8 \times 1)$
= $3072 + 240 + 8$
= 3320 decimal

This is written as $CF8_{16} = 3320_{10}$, writing the base number as a subscript.

Decimal to hexadecimal conversion

Here we will apply the same technique we used in Unit 7 to convert decimal numbers to binary.

To convert to binary, we divided the decimal number by 2. To convert to hexadecimal we will divide the decimal number by 16.

Example 8.3 Convert 3320_{10} to hexadecimal.

Solution

16	3320	
16	207	rem 8
16	12	rem 15 (which is F)
	0	rem 12 (which is C)

↑

Therefore, $3320_{10} = CF8_{16}$

Activity 2

1 Convert the following hexadecimal numbers to decimal.

1.1 $A7_{16}$

1.2 $C3F_{16}$

2 Convert the following decimal numbers to hexadecimal.

2.1 49_{10}

2.2 2017_{10}

Binary to hexadecimal conversion

You will notice that four binary bits correspond perfectly with one hexadecimal digit. To represent binary numbers as hexadecimal numbers, mark off groups of four, starting at the LSB and moving to the left. Then convert each group into the corresponding hexadecimal digit.

Example 8.4 Convert 101101111_2 to a hexadecimal number.

Solution 0001 0110 1111 groups of 4, filling left group with zeroes
 1 6 F represented as hexadecimal

therefore $101101111_2 = 16F_{16}$

Activity 3

1 Convert the following binary numbers to hexadecimal.

1.1 10111_2

1.2 11101011_2

Hexadecimal to binary conversion

Convert hexadecimal to binary as follows: For each hexadecimal digit write the corresponding four binary bits.

Example 8.5 Convert $AF9_{16}$ to a binary number.

Solution

A	F	9
1010	1111	1001

therefore $AF9_{16} = 101011111001_2$

Activity 4



1 Convert the following hexadecimal to binary numbers.

1.1 AF_{16}

1.2 $3CD_{16}$

2.2 The binary-coded decimal (BCD) number system

The binary-coded decimal number system is a way of representing each of the decimal digits by using four binary bits, according to the 8, 4, 2, 1 (binary weights) binary system.

Decimal to BCD conversion

Convert decimal to BCD as follows: For each decimal digit write the corresponding binary bits.

Example 8.6 Convert 2048 decimal to BCD.

Solution

2	0	4	8
0010	0000	0100	1000

therefore $2048_{10} = 0010000001001000_{BCD}$

To ensure that each decimal digit is represented by four bits, we add zeros to the front of the binary bits.

Activity 5



1 Convert the following decimal numbers to BCD.

1.1 567_{10}

1.2 389_{10}

BCD to decimal conversion

It is just as easy to convert a BCD number to a decimal number. Start at the MSB and break up the code into groups of four bits. Convert each group of four bits into the corresponding decimal digit.

Example 8.7 Convert $0110100101110101_{\text{BCD}}$ to decimal.

Solution

0110	1001	0111	0101
6	9	7	5

therefore $011010001110101_{\text{BCD}} = 6975_{10}$

Activity 6

1 Convert the following BCD numbers to decimal.

1.1 01110000_{BCD}

1.2 $100101110011_{\text{BCD}}$

Hexadecimal to BCD

Example 8.8 shows you how to convert hexadecimal to BCD.

Example 8.8 Convert $C5_{16}$ to BCD.

Solution First convert $C5_{16}$ to binary.

C	5
1101	0101

$C5_{16} = 11010101_2$

Now convert 11010101_2 to decimal.

$$\begin{aligned} & 1 \times 2^7 + 1 \times 2^6 + 0 + 1 \times 2^4 + 0 + 1 \times 2^2 + 0 + 1 \times 2^0 \\ &= 128 + 64 + 16 + 4 + 1 \\ &= 253_{10} \end{aligned}$$

Next convert 253_{10} to BCD.

2	5	3
0010	1001	0011

therefore $C5_{16} = 001010010011_{\text{BCD}}$

Activity 7



1 Convert the following hexadecimal numbers to BCD.

1.1 $E2_{16}$

1.2 $1F2_{16}$

2.3 Addition using binary and hexadecimal numbers

Binary arithmetic is essential in all digital computers and many other digital electronic systems.

Binary addition

Binary addition is done in exactly the same way as decimal addition; except that the rules are simplified because there are fewer possible combinations of digits.

The four rules for binary addition are:

- 1** $0 + 0 = 0$ sum of 0 with carry of 0
- 2** $0 + 1 = 1$ sum of 1 with carry of 0
- 3** $1 + 0 = 1$ sum of 1 with carry of 0
- 4** $1 + 1 = 10$ sum of 0 with carry of 1

Example 8.9 Add the following using binary numbers:

- a** $14_{10} + 19_{10}$
- b** $5,25_{10} + 3,125_{10}$

Solution

	Decimal	Binary
a	$\begin{array}{r} 14 \\ +19 \\ \hline 33 \end{array}$	$\begin{array}{r} 01110 \\ +10011 \\ \hline 100001 \end{array}$
b	$\begin{array}{r} 5,250 \\ +3,125 \\ \hline 8,375 \end{array}$	$\begin{array}{r} 0101,010 \\ +0011,001 \\ \hline 1000,011 \end{array}$

Activity 8**1** Add the following binary numbers.

1.1 $1101001_2 + 11011_2$

1.2 $110111,101_2 + 1001,1111_2$

Hexadecimal addition

Addition with hexadecimal numbers can be done directly if you keep in mind that the decimal digits 10 to 15 are equivalent to the hexadecimal digits A to F. The rules are as follows:

- 1** If the sum of the two digits is 15_{10} or less, write down the corresponding hexadecimal number.
- 2** If the sum is greater than 15_{10} , subtract 16_{10} from the answer and write down the hexadecimal result; also carry 1 to the next column.

Example 8.10 Add the following hexadecimal numbers.

a $82_{16} + 15_{16}$

b $48_{16} + 32_{16}$

c $5C_{16} + F7_{16}$

Solution

$$\begin{array}{r} \mathbf{a} \quad 82_{16} \text{ right column } 2_{16} + 5_{16} = 2_{10} + 5_{10} = 7_{16} \\ + 15_{16} \text{ left column } 8_{16} + 1_{16} = 8_{10} + 1_{10} = 9_{16} \\ \hline 97_{16} \end{array}$$

$$\begin{array}{r} \mathbf{b} \quad 48_{16} \text{ right column } 8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16} \\ + 32_{16} \text{ left column } 4_{16} + 3_{16} = 4_{10} + 3_{10} = 7_{16} \\ \hline 7A_{16} \end{array}$$

$$\begin{array}{r} \mathbf{c} \quad 5C_{16} \text{ right column } C_{16} + 7_{16} = 12_{10} + 7_{10} = 19_{10} \\ + F7_{16} \text{ left column } 19_{10} - 16_{10} = 3_{10} = 3_{16} \text{ with a carry of 1.} \\ \hline 153_{16} \text{ left column } 5_{16} + F_{16} + 1_{16} \text{ (carry)} \\ = 5_{10} + 15_{10} + 1_{10} = 21_{10} \\ 21_{10} - 16_{10} = 5_{10} = 5_{16} \text{ with a carry of 1.} \end{array}$$

Activity 9**1** Add the following hexadecimal numbers.

1.1 $32_{16} + 28_{16}$

1.2 $128_{16} + 67_{16}$

3 Adders



A digital system uses only addition and subtraction to do arithmetic operations. All the other operations, such as multiplication and division, can be performed by repeated addition or subtraction. The simplest digital circuit that performs arithmetic operations is called the **adder**.

When we are doing addition, the quantities to be added are called the 'addend' and the 'augend'. When we combine these by addition, it produces a 'sum'. This is symbolically represented as:

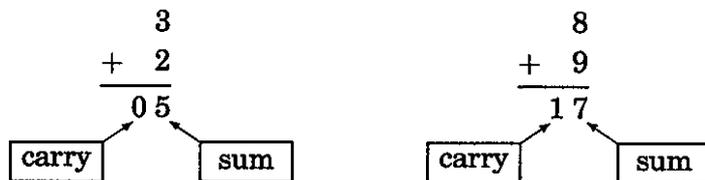
$$\begin{array}{r} X \text{ (augend)} \\ + Y \text{ (addend)} \\ \hline Z \text{ (sum)} \end{array}$$

where X and Y are the numerical quantities to be added and Z is their sum.

The sum can be subdivided into its:

- ◇ **least significant digit:** which is generically referred to as the 'sum'
- ◇ **most significant digit:** which is referred to as the 'carry'.

Example 8.11 Using the base ten number system (decimal) as an example:



3.1 The half adder

The same principles as outlined above, apply to the base two number system, the binary number system. All possible cases for the addition of two binary digits are illustrated below:

$$\begin{array}{r} 0 \\ + 0 \\ \hline 00 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 01 \end{array} \quad \begin{array}{r} 1 \\ + 0 \\ \hline 01 \end{array} \quad \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

If we are now asked to **summarise** (by means of a truth table) and **symbolise** (by means of a block diagram) the process of addition described above, we could do it in the following way.

Truth table for two bit binary adder:

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



The circuit which would implement the truth table above, is called a **half-adder**; often abbreviated to **HA**.

From the truth table we observe that our system should have two inputs and two outputs. The logic symbol for the half adder can therefore be modelled in block diagram form as:

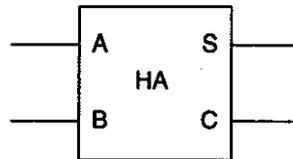


Fig 8.1

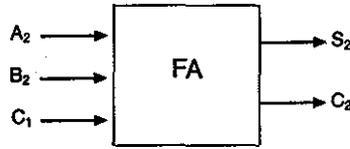
To determine the internal construction of this half adder we could separate the truth table into two parts, as shown below.

A	B	sum
0	0	0
0	1	1
1	0	1
1	1	0

A	B	carry
0	0	0
0	1	0
1	0	0
1	1	1

From the truth table on the left, we can conclude that the sum output can be implemented by means of an Exclusive-OR gate. From the truth table on the right, we can conclude that the carry output can be implemented by means of an AND gate. This is represented in Fig 8.2 which is the logic circuit of a half adder.

There are 3 inputs. If we look at the subscripts, we see that we are adding: the present augend to the present addend to the carry generated by the summation in the previous column.



There are 2 outputs. If we look at the subscripts, we see that we are generating: the present sum, and the present carry (which will be used when summing the next column).

Fig 8.3



The circuit which implements the logical function outlined above, is referred to as a **full adder**; often abbreviated to **FA**.

If we are now asked to **symbolise** (by means of a block diagram), and **summarise** (by means of a truth table) the process of addition for the full adder, we could do it as set out below. Fig 8.4 shows the logic symbol for the full adder.

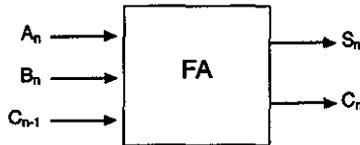


Fig 8.4

The truth table for the full adder is:

Augend bit input A_n	Addend bit input B_n	Carry bit input C_{n-1}	Sum bit output S_n	Carry bit output C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

For the sum output, we can now express this truth table as a Karnaugh map and then develop a minimal equation for the sum output, as shown in Fig 8.5.

		A_n					
		B_n	00	01	11	10	
C_{n-1}	0	0	1	0	1		
	1	1	0	1	0		

$$S_n = C_{n-1}\bar{A}_n\bar{B}_n + \bar{C}_{n-1}\bar{A}_nB_n + C_{n-1}A_nB_n + \bar{C}_{n-1}A_nB_n$$

$$= C_{n-1} \oplus A_n \oplus B_n \quad (\text{this can be proved!})$$

Fig 8.5

For the carry output, we can express this truth table as a Karnaugh map and develop a minimal equation for the carry output as shown in Fig 8.6.

		A_n					
		B_n	00	01	11	10	
C_{n-1}	0	0	0	1	0		
	1	0	1	1	1		

$$C_n = C_{n-1}A_n + C_{n-1}B_n + A_nB_n$$

Fig 8.6

Using the minimised equations developed above, we can now implement the complete logic circuitry for a full adder as shown in Fig 8.7.

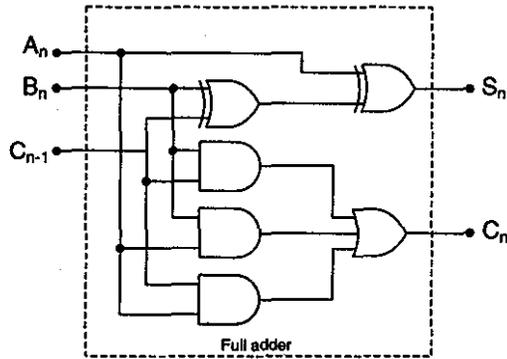


Fig 8.7

The full adder may also be constructed from two half-adders and an OR gate. Fig 8.8 shows the block diagram, and Fig 8.9 shows the complete logic diagram required to achieve this. The equations at the intermediate and final output points are also indicated on both figures.

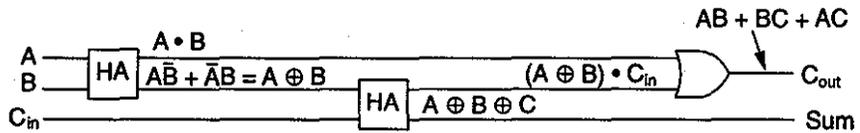


Fig 8.8

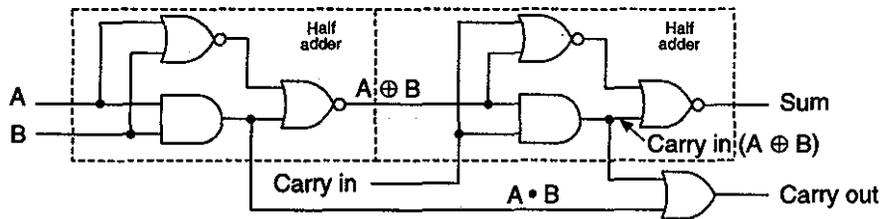


Fig 8.9

3.3 Parallel binary adders

At this stage we must take stock, and remember that the full adder is the solution to only one step of our original problem of adding two multi-digit numbers.

If we want to complete all the steps required to add two four-digit binary numbers, we would have to use the full adders in the way shown in Fig 8.10. Here we have a four-bit parallel binary adder implementation using full adders.

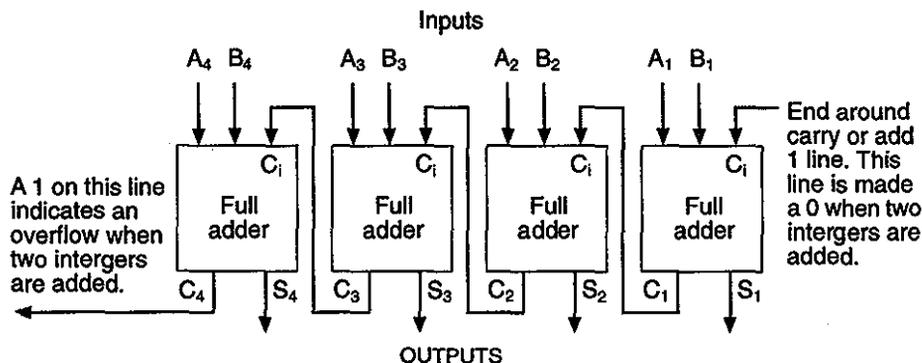


Fig 8.10

To implement the circuit above (a four-bit parallel binary adder) using discrete logic gates, would be a very labour intensive task, and because of this you could easily make mistakes. Luckily, many manufacturers have fabricated circuitry for this function onto an integrated circuit (IC).

Typical of this type of integrated circuit is the 7483 4-bit binary adder.

The logic symbol for the 7483 is shown in Fig 8.11.

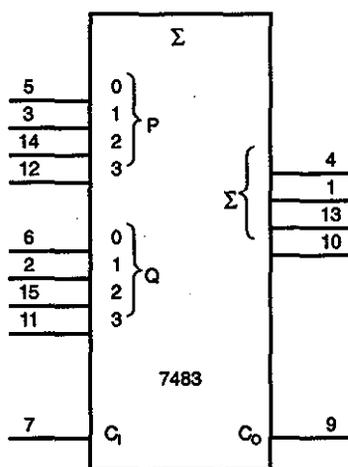


Fig 8.11

Activity 10

- 1 Use the 7482 two-bit binary full adder, and show how you would interconnect two of these integrated circuits to achieve the function of a four-bit binary full adder.
- 2 Show how two 7483 four-bit parallel adders can be connected to form an eight-bit parallel adder. Show outputs for the case when:

$$P_7 \ P_6 \ P_5 \ P_4 \ P_3 \ P_2 \ P_1 \ P_0 = 10011011$$

$$Q_7 \ Q_6 \ Q_5 \ Q_4 \ Q_3 \ Q_2 \ Q_1 \ Q_0 = 10101010$$

4 Decoders and encoders

4.1 Decoders



A decoder interprets an input binary number to produce a single output for each number. That is, one unique output will go **high** while the others will remain **low** for that particular number.

Two-bit decoder

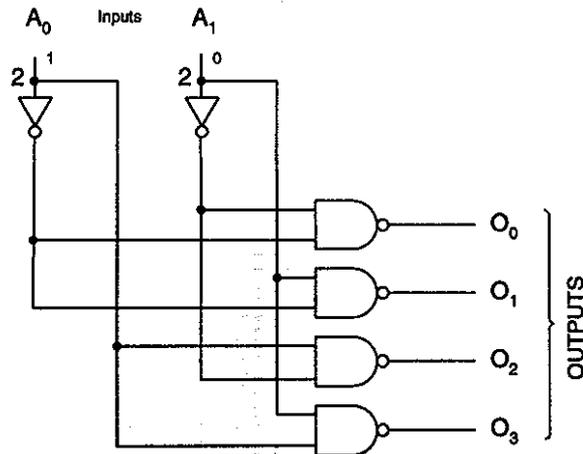


Fig 8.12

Truth table of a 2-bit decoder:

Inputs		Outputs			
A ₀	A ₁	O ₀	O ₁	O ₂	O ₃
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Three-bit decoder

The logic diagram of a 1-of-8 decoder is shown in Fig 8.13. If you were to increase the size of the binary number input to the decoder to 3 bits, the number of inputs would be 2^3 , or 8 if the decoder was to be a full decoder as shown in Fig 8.13. Notice that as the number of bits in the input increases, so does the number or inputs of AND gates used. If the binary number gets very large a full decoder can become very large. For example, a full 8-bit decoder would need 256 NAND gates, each with 8 inputs.

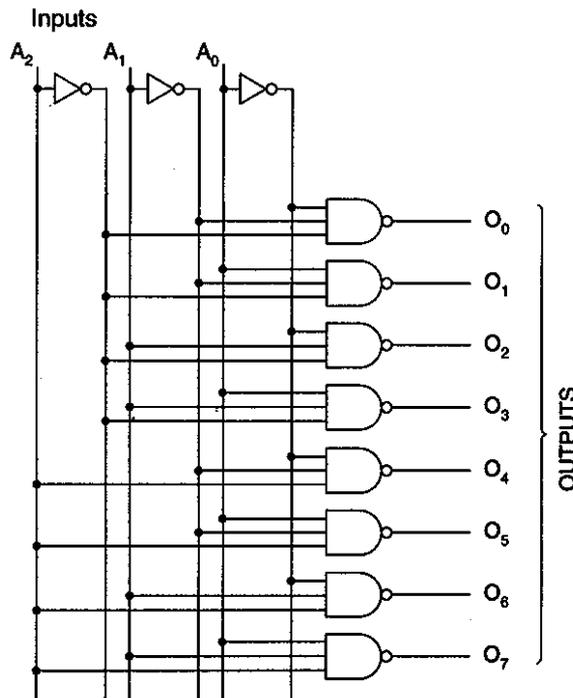


Fig 8.13

The logic symbol and logic diagram of a 3-bit decoder is given above in Fig 8.13. The decoder accepts three binary inputs (A_0, A_1, A_2) and it provides eight mutually exclusive active low outputs (O_0, \dots, O_7). The truth table will look like this:

Inputs			Outputs							
A_0	A_1	A_2	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Activity 11



The 7442 is an integrated circuit BCD-to-decimal decoder. The logic symbol is shown in Fig 8.14. Draw the truth table for the BCD-to-decimal decoder. If the input waveforms shown are applied to the inputs of the decoder, sketch the output waveforms.

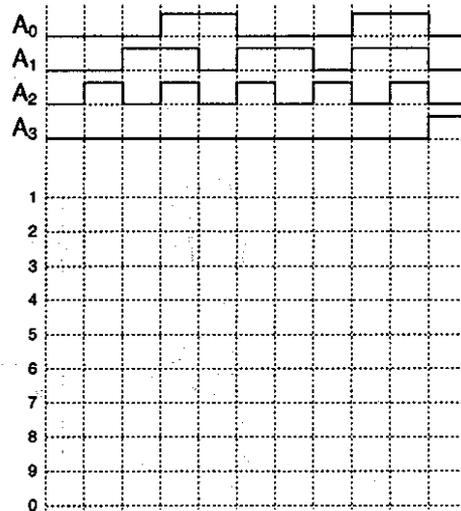
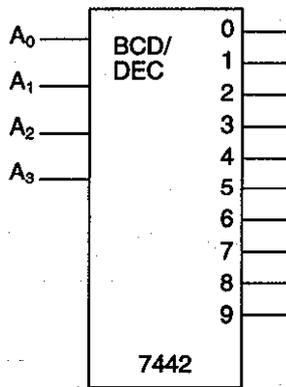


Fig 8.14

The BCD-to-seven-segment code decoder

To display numeric information, a device called a seven-segment display unit may be used. The device is shown in Fig 8.15a. It is made up of seven LEDs. The LED may turn on if it is activated. To display a certain number, the appropriate segments are activated. The segments are labelled A, B, C, D, E, F and G. For example, to display a 0, segments A, B, C, D, E and F must be activated. Segment G must not be activated. The truth table below shows the relationship between the BCD code and the seven-segment code.

Decimal	BCD				Seven-segment code						
	B ₃	B ₂	B ₁	B ₀	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

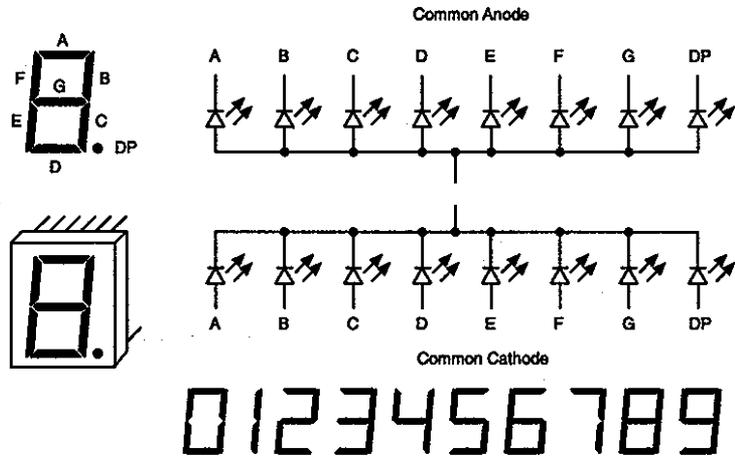


Fig 8.15a

Fig 8.15b shows the circuit diagram of an IC 7447 BCD-to-seven-segment code decoder, connected to a display unit.

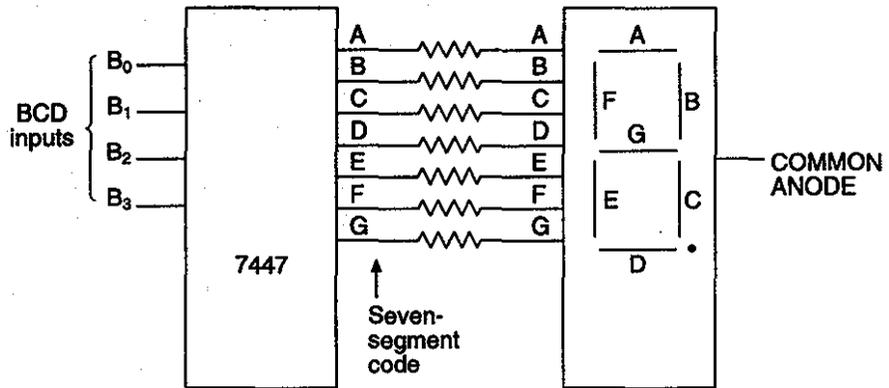


Fig 8.15b

4.2 Encoders



An encoder is a combinational logic circuit that essentially performs a 'reverse' decoder function. An encoder accepts an active level on one of the inputs representing a digit, such as a decimal or hexadecimal digit, and converts it to a coded output, such as BCD or binary. Encoders can also be devised to encode various symbols and alphabetic characters.

The decimal-to-BCD encoder

This type of decoder has ten inputs – one for each decimal digit – and four outputs corresponding to the BCD code, as shown in Fig 8.16.

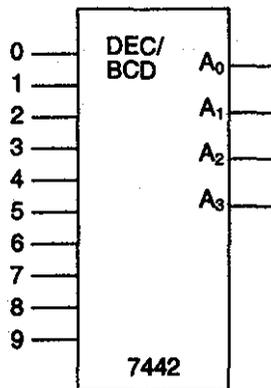


Fig 8.16

BCD code				
Decimal digit	A ₃	A ₂	A ₁	A ₀
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

The table above shows the BCD coded output for this device.

An encoder application

Fig 8.17 shows a simple keyboard encoder arrangement using a 74LS147 priority encoder. The keys are represented by ten push-button switches, each with a pull-up resistor. When a key is depressed, the line is connected to ground, and a low is applied to the corresponding encoder input. The zero key is not connected because the BCD output represents zero when none of the other keys are depressed.

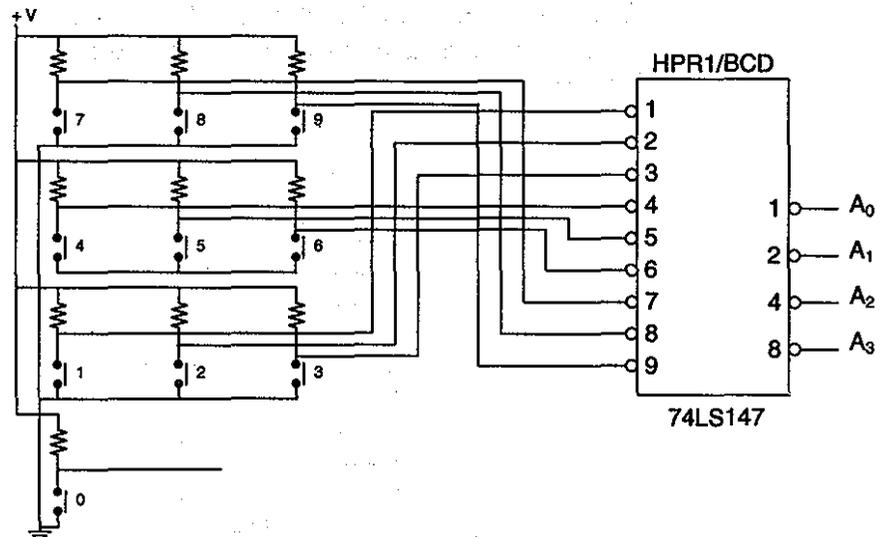


Fig 8.17

Activity 12

Fig 8.18 is the logic diagram of a decimal-to-BCD encoder. Draw the output with respect to waveforms given.

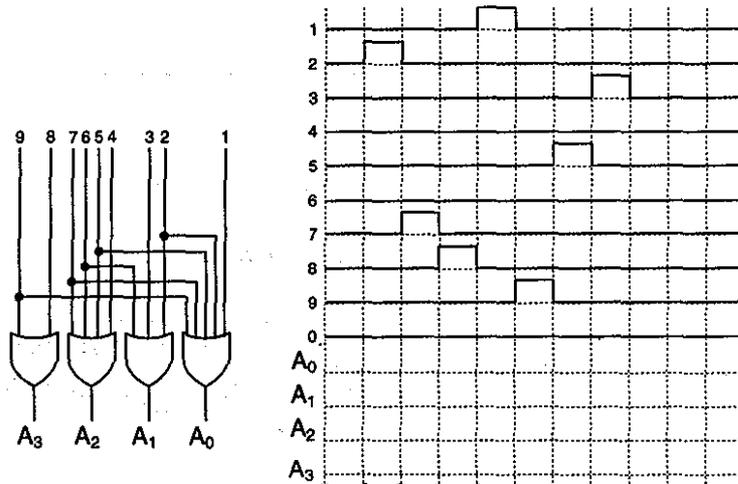


Fig 8.18

5 Multiplexers and demultiplexers

5.1 Multiplexers



A multiplexer is a device that allows data in binary form to be transferred by means of a single line. The multiplexer normally has several inputs, but a single output. The information can be transferred under the control of the data select lines. **Multiplexers** are also known as **data selectors**.

The basic operation of a multiplexer

A multiplexer selects one channel as an input and connects it to a signal output, as shown in the block diagram in Fig 8.19. The circuit consists of the input lines, select lines and output lines. The number of select lines is determined by the number of data lines, only one output line is normally necessary. In the block diagram in Fig 8.19, four data lines can be selected by two select lines.

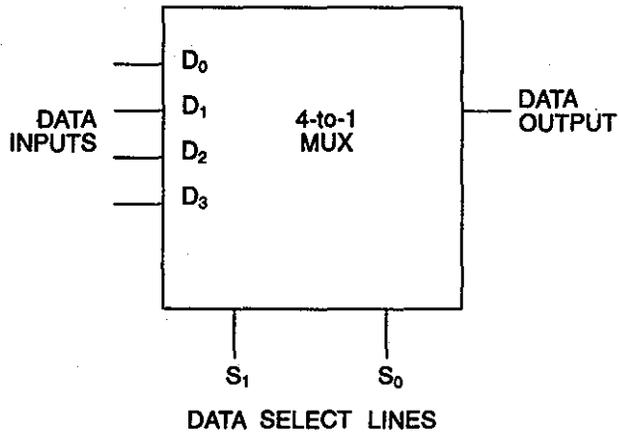


Fig 8.19

Data selection for a four-input		
Data selection line		Input selection
S ₁	S ₀	
0	0	D ₀
0	1	D ₁
1	0	D ₂
1	1	D ₃

A two-bit binary code on the data select lines will allow the data on the corresponding data input lines to pass through the multiplexer.

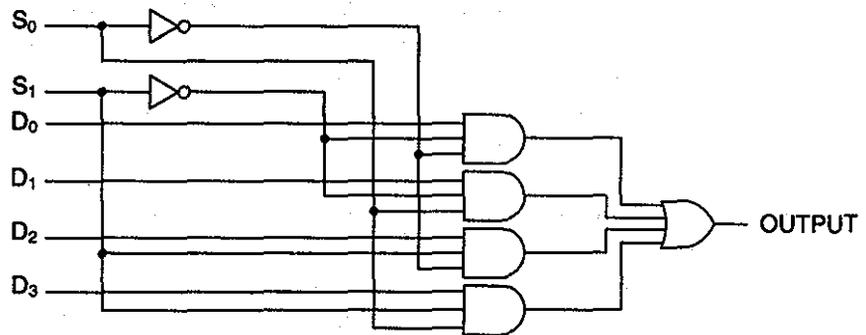


Fig 8.20

The AND gates will control the output, which is selected by the binary number input to the select inputs. The table on the previous page shows data selection for a four-input multiplexer.

An 8-input multiplexer

The 74LS151 is a high speed 8-input digital multiplexer. It provides in one package, the ability to select one line of data from up to eight sources. The logic diagram is shown in Fig 8.21.

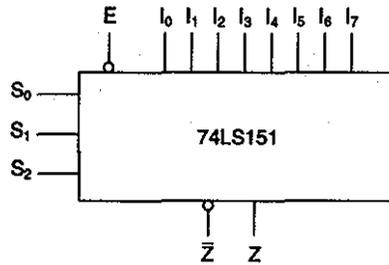


Fig 8.21

The 74157 quad 2-input multiplexer

The 74157 consists of four 2-input multiplexers. Each of the four multiplexers shares a common data-select line and a common enable. Because there are only two inputs to be selected in each multiplexer, a single data-select input is sufficient. Fig 8.22 shows the logic diagram and logic symbol of this multiplexer.

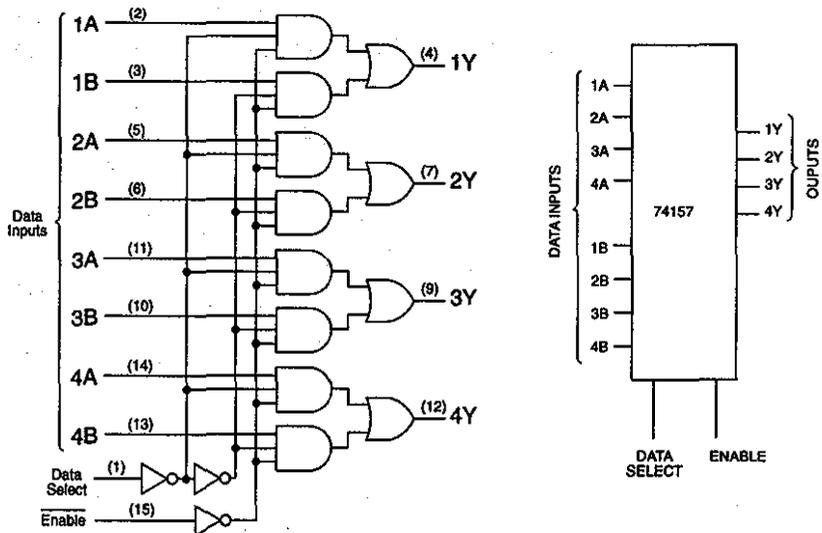


Fig 8.22

Activity 13



The data-input and data-select waveforms shown below are applied to the multiplexer in Fig 8.23. Determine the output waveform.

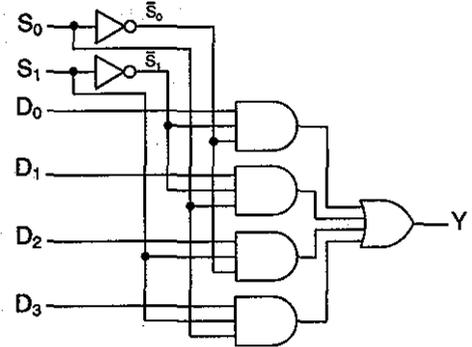
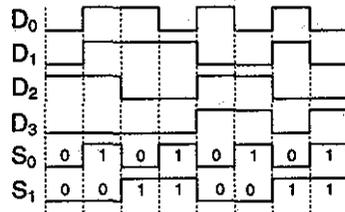


Fig 8.23

5.2 Demultiplexers



A demultiplexer is a digital switch which allows us to switch one input to one of many possible output lines. The line which we want the input to be connected to is determined by a binary number which is input to the demultiplexer. Fig 8.24 shows a 1-to-4 demultiplexer. A demultiplexer uses the input line as a data input. The data appears on the selected output line when the corresponding binary number is input to the select lines.

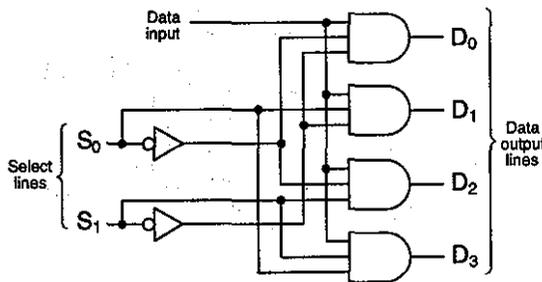


Fig 8.24

Example 8.12 The data-input waveform (data in) and data-select lines (S_0 and S_1) are shown in Fig 8.25. Determine the data-output waveforms on D_0 through to D_3 for the demultiplexer in Fig 8.24.

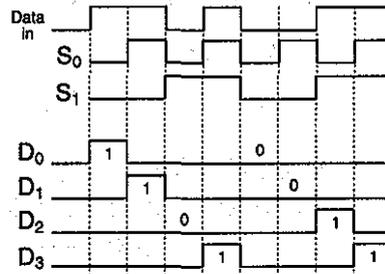


Fig 8.25

Activity 14 Draw the output waveforms for the demultiplexer in Fig 8.24 if the waveforms for S_0 in Fig 8.25 is inverted.



6 Summary

In this unit you looked at various applications of combinational logic. You began with an introduction to two new number systems, namely, the hexadecimal and BCD systems. Then you explored how circuits which can perform binary addition are designed. You were then shown how combinational logic is used in the design and implementation of decoders, encoders, multiplexers and demultiplexers.

In the next unit you will be introduced to other types of logic circuits, namely sequential logic circuits.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

1 Convert 1956_{10} to:

1.1 binary

1.2 BCD

1.3 hexadecimal

2 Convert $F23_{16}$ to:

2.1 decimal

2.2 BCD

3 Work out the sum of:

3.1 $2E_{16} + 4DC_{16}$

3.2 $1010_2 + 1111_2$

4 Use half adders and full adders to draw the diagram for an adder that will add two 3-bit binary numbers.

5 What is the difference between a decoder and an encoder?

6 What is a multiplexer used for?

Answers to activities

Activity 1

EF, F0, F1, F2, F3, F4, F5, F6, F7, F8, F9, FA, FB, FC, FD, FE, FF, 100, 101, 102, 103

Activity 2

1.1 $A \times 16^1 + 7 \times 16^0 = 10 \times 16 + 7 \times 1 = 167_{10}$

1.2 3135_{10}

2.1

16	49	
16	3	rem 1
16	0	rem 3

↑

$49_{10} = 31_{16}$

2.2

16	2017	
16	126	rem 1
16	7	rem D
16	0	rem 7

↑

$2017_{10} = 7D1_{16}$

Activity 3

- 1.1 0001 0111
 1 7
 therefore $10111_2 = 17_{16}$
- 1.2 EB_{16}

Activity 4

- 1.1 A F
 1010 1111
 therefore $AF_{16} = 10101111_2$
- 1.2 1111001101_2

Activity 5

- 1.1 5 6 7
 0101 0110 0111
 therefore $567_{10} = 010101100111_{BCD}$
- 1.2 001110001001_{BCD}

Activity 6

- 1.1 0111 0000
 7 0
 therefore $01110000_{BCD} = 70_{10}$
- 1.2 973_{10}

Activity 7

- 1.1 $E2_{16} = 226_{10} = 001000100110_{BCD}$
- 1.2 $1F2_{16} = 498_{10} = 010010011000_{BCD}$

Activity 8

- 1.1
$$\begin{array}{r} 1101001 \\ + 11011 \\ \hline 10000100_2 \end{array}$$
- 1.2 $1000001,1001_2$

Activity 9

1.1 $5A_{16}$

1.2 $18F_{16}$

Activity 10

- 1 Fig 8.26 shows two 7482s used as a four-bit adder. Additional bits can be added by connecting the CO of the most significant adder to the CI of the next adder, and so on. Note that the CI of the least significant adder is grounded (0) because there is no carry input to this stage.

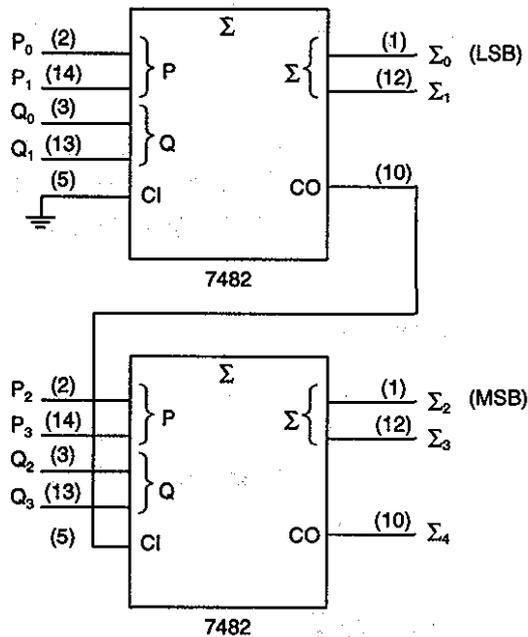


Fig 8.26

- 2 The only connection between the two 7483s is the carry output (CO) of the lower order adder to the carry input (CI) of the higher order adder as shown in Fig 8.27. The CI of the least significant 7483 is grounded. (no input carry)

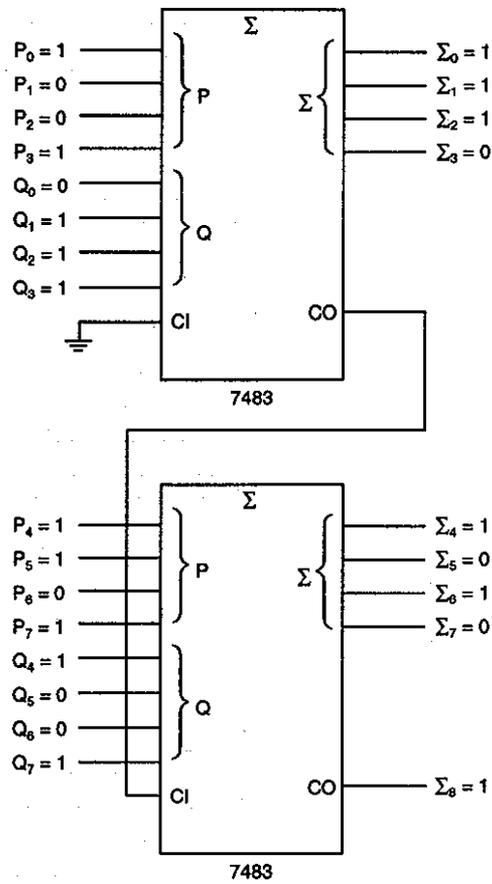


Fig 8.27

Activity 11

BCD inputs				Decimal outputs									
A ₀	A ₁	A ₂	A ₃	0 ₀	0 ₁	0 ₂	0 ₃	0 ₄	0 ₅	0 ₆	0 ₇	0 ₈	0 ₉
0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	0	1	1	1	1	1	1	1	1
0	1	0	0	1	1	0	1	1	1	1	1	1	1
1	1	0	0	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	1	1	0	1	1	1	1	1
1	0	1	0	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

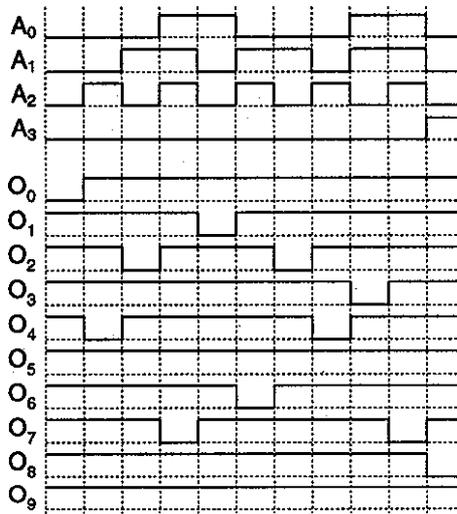


Fig 8.28

Activity 12

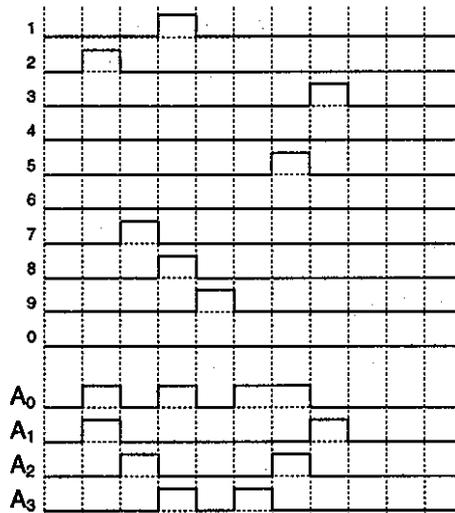


Fig 8.29

Activity 13

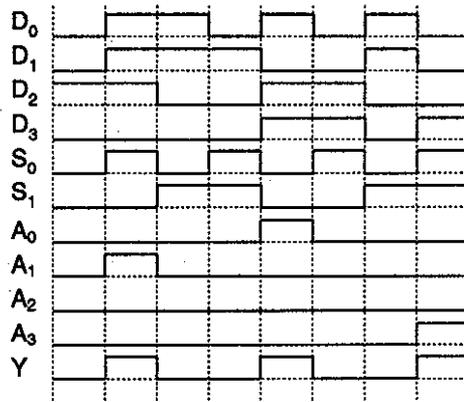


Fig 8.30

Activity 14

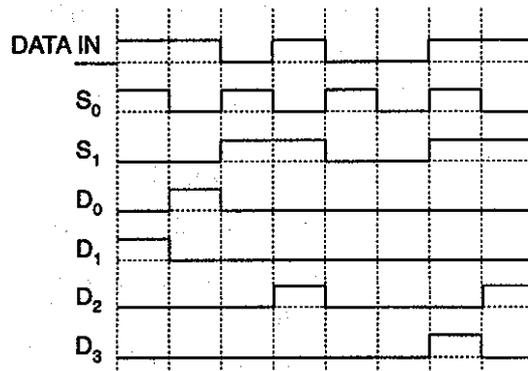


Fig 8.31

Sequential logic circuits

Study objectives

After studying this unit, you should be able to:

- ◇ use NAND and NOR logic gates to construct basic latches
- ◇ construct the timing diagram for a latch
- ◇ eliminate contact bounce using a latch
- ◇ draw the logic circuit for the gated S-R and D Latch as well as for the J-K flip-flop
- ◇ recognize the difference between latches and flip-flops
- ◇ apply flip-flops in basic applications
- ◇ define the basic operating modes of shift registers
- ◇ implement serial to parallel conversion using shift registers
- ◇ explain how a bi-directional shift register operates
- ◇ use shift register counters
- ◇ build a 3-bit up/down asynchronous counter using J-K flip-flops
- ◇ design a modulo-N asynchronous counter
- ◇ eliminate glitches in counter decoding.

1 Introduction

In Units 7 and 8 you were introduced to combinational logic circuits. You would have noticed that the outputs of those circuits at any particular time were determined by the state of the inputs at that instant. In this unit you will be introduced to devices with outputs that depend on present as well as past inputs. These are examples of memory devices. First you will cover two types of memory devices, latches and flip-flops. Then you will be shown how these devices can be connected together to form registers and counters. The remainder of the unit will be used to examine

various applications of flip-flops in registers and counters. Since such circuits operate by shifting data in a particular sequence from one device to another, they are called **sequential circuits**.

Can you think of possible applications of sequential circuits? The most obvious one is the digital watch.

2 Latches and flip-flops



In this section we cover the basic building blocks of sequential circuits, namely latches and flip-flops. These devices have two outputs which are always in opposite states. If the one output Q is high, the other output, \bar{Q} , is low. Alternatively, if \bar{Q} is high, then Q is low. As you can see, such devices have two stable states and are therefore called **bistable logic devices**.

Think of the flip-flop as a seesaw on a playground. A seesaw also has two stable states. If the one side is high, the other is low, and vice versa. The outputs of a flip-flop behave in a similar fashion.



2.1 The S-R latch

Fig 9.1 shows a simple NAND set-reset (S-R) latch, comprising two NAND gates connected in a crossed configuration. It has two inputs, \bar{S} and \bar{R} , two outputs, Q and \bar{Q} , and two stable states, the **set** state and the **reset** state. You will notice that the input signals are active low, as denoted by compliment bars over these signals. This means that the \bar{S} input must be 0 to set the Q output to 1. When the Q output is 1, the latch is **set** and when it is 0, the latch is **reset**.

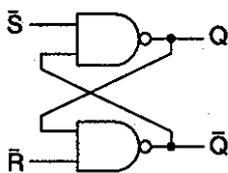


Fig 9.1

Fig 9.2 shows the latch in set mode. An active low signal is applied to the \bar{S} input and the \bar{R} input is held at 1. The latch goes to set condition, with the Q output at 1 and the \bar{Q} output at 0.

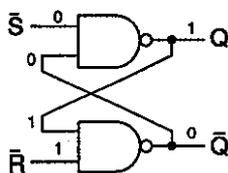


Fig 9.2

If the \bar{S} input goes to a 1 and the \bar{R} input remains at 1 as in Fig 9.3, the output does not change. This is because the outputs are fed back into the input of the opposite gate, which makes them retain their original output states. This is the memory mode of the latch.

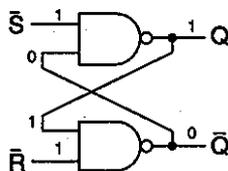


Fig 9.3

An active low signal is applied to the \bar{R} input, while keeping the \bar{S} input at 1, as in Fig 9.4. The latch then goes to the reset condition.

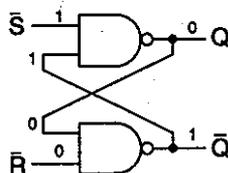


Fig 9.4

The only other possible state for the two inputs is to have them both 0 at the same time. This is an invalid condition since it results in both outputs being 1, as shown in Fig 9.5. In terms of what we said earlier, we never want a latch to have Q and \bar{Q} with the same value.

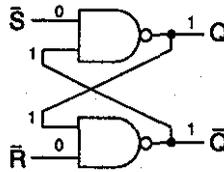


Fig 9.5

The operation or truth table for the NAND S-R latch can be summarised as shown below.

Inputs		Outputs		Comments
\bar{S}	\bar{R}	Q	\bar{Q}	
1	1	NC	NC	no change, latch remains in present state
0	1	1	0	latch set
1	0	0	1	latch reset
0	0	1	1	invalid condition

The S-R latch can also be constructed using NOR logic gates as shown in Fig 9.6. The inputs are now active-high and the Q and \bar{Q} outputs are reversed.

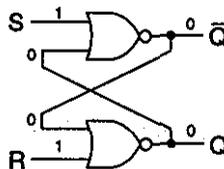


Fig 9.6

When the S input goes to 1, and the R input remains at 0, the latch is set since the Q output goes to a 1 state as in Fig 9.7.

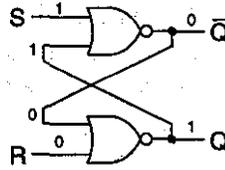


Fig 9.7

When the S input goes to a 0, the R input remaining at a 0, then the outputs Q and \bar{Q} do not change as shown in Fig 9.8. This is the memory mode of the latch.

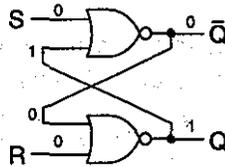


Fig 9.8

The R input is now made 1, while the S input remains at 0 as shown in Fig 9.9. This is the reset mode of the flip-flop since Q now becomes 0.

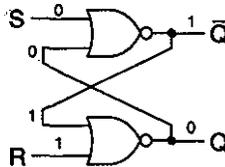


Fig 9.9

Fig 9.10 shows the invalid state for the NOR latch, where both inputs are 1, resulting in the outputs both being at 0.

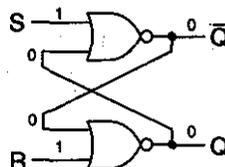


Fig 9.10

The operation or truth table for the NOR latch can be summarised as shown in the table below:

Inputs		Outputs		Comments
S	R	Q	\bar{Q}	
0	0	NC	NC	no change, latch remains in present state
1	0	1	0	latch set
0	1	0	1	latch reset
1	1	0	0	invalid condition

Timing diagram for a latch

Timing diagrams provide a useful means of analysing sequential circuits. A timing diagram for a NAND latch is shown in Fig 9.11. Here the \bar{S} and \bar{R} waveforms are applied to the latch with the truth table for the NAND S-R latch shown earlier. Assume that Q is initially at 0. Using the truth table you can predict the output Q at any time by simply considering the state of the inputs at that time.

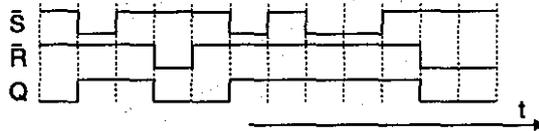


Fig 9.11

Activity 1

The waveforms in Fig 9.12 below are applied to the NAND latch. Sketch the resulting output waveform at Q, assuming that Q is initially at 0.

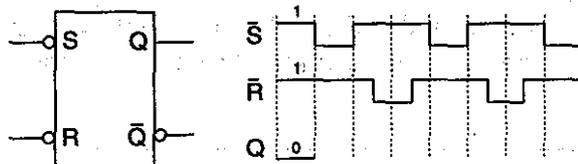


Fig 9.12

The latch as a contact bounce eliminator

When a normal switch opens or closes the contacts tend to bounce, as they do not make or break the circuit smoothly. This input and output waveform is shown in Fig 9.13.

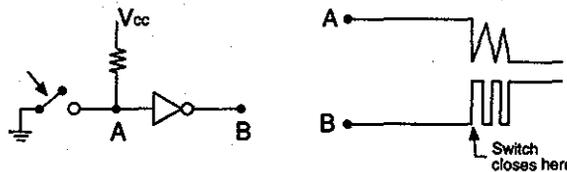


Fig 9.13

How can we use a latch to eliminate this problem? From what you have learnt so far, you will know that the output of a latch can remain unchanged even if the input condition changes. A possible solution would then be to include a latch in the circuit as shown in Fig 9.14.

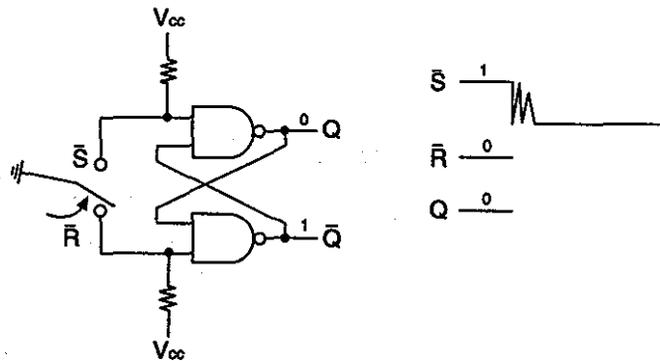


Fig 9.14

When the switch makes contact with the \bar{S} input, the output changes state and remains there even after the switch bounces.

Activity 2



Look at Fig 9.14. Assume that Q is initially at 0. Complete the timing diagram after the switch closes.

2.2 The gated S-R latch

In the latches we discussed so far, the state of the outputs (Q and \bar{Q}) changed in response to changes to the state of the inputs (S and R). This change in output occurred the moment the input state changed. A gated latch uses an additional input, the enable input or EN , to control when the output may change. The gated S-R latch is shown in Fig 9.15. You will notice that the output cannot change when EN is low irrespective of the state of the inputs. When EN is high, however (and as long as EN remains high), the output is controlled by the state of the S and R inputs.

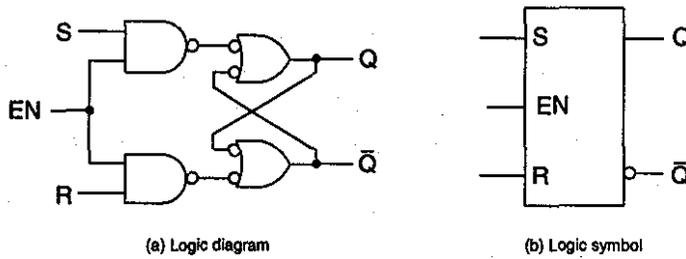


Fig 9.15

The timing diagram of the S-R Latch is shown in Fig 9.16. You will notice that the output only changes while EN is high.

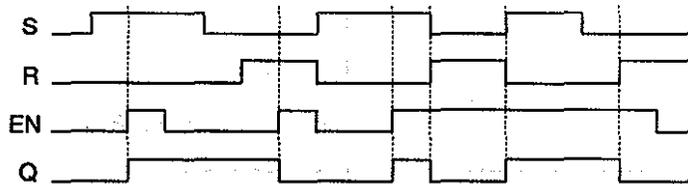


Fig 9.16

Activity 3



For the gated S-R latch shown in Fig 9.17, determine the Q output for the inputs shown, assuming Q is initially at 0.

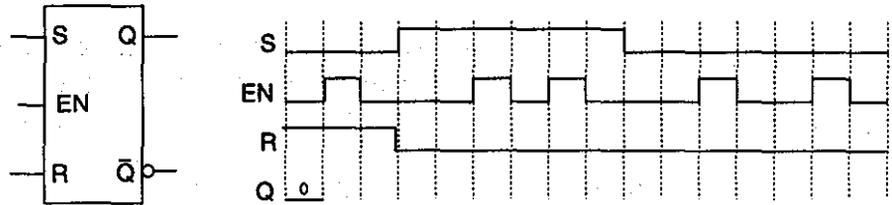


Fig 9.17

2.3 The gated D latch

The gated D latch differs from the gated S-R latch in that the S and the R inputs are connected by an INVERTER. Therefore the S and R inputs will always be in opposite states. In effect this latch only has one input, the D (or data) input. The logic diagram and logic symbol for the D latch are shown in Fig 9.18. You will notice that the output Q is the same as the input D whenever EN is high.

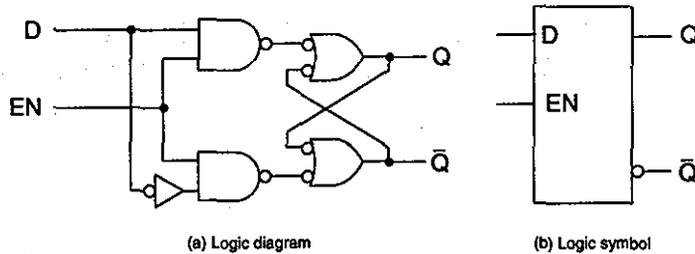


Fig 9.18

Activity 4



Determine the Q output for a gated D Latch and the waveforms shown in Fig 9.19.

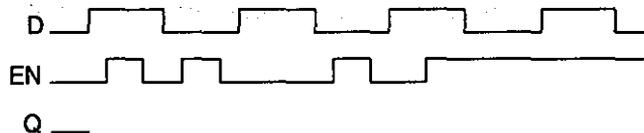


Fig 9.19

2.4 The edge triggered flip-flop

In the previous section we dealt with gated latches where the output responded to changes in the input only while an additional input, the enable input, remained high (or while the gate was 'open'). This may present a problem if you wanted the output to respond only at the exact instant the gate opened and not for the duration of time that it remained open.

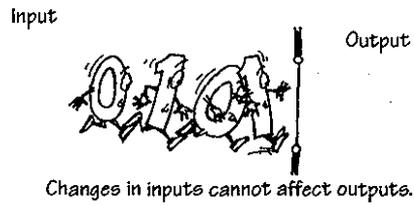
In this section we look at devices called flip-flops which allow you to overcome this problem. When you use a flip-flop the output changes either at the positive edge (rising edge) or the negative edge (falling edge) of a triggering input called **the clock** (CLK). This allows you more precise control over when the outputs change. The triggering input of the clock is depicted by an arrow, as follows:

rising edge-trigger: 

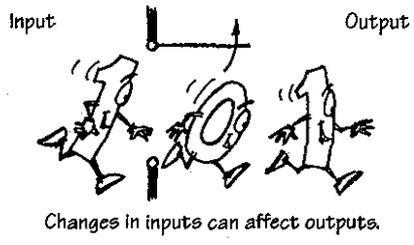
falling-edge trigger: 

It might help you distinguish between the gated latch and a flip-flop if you think of the gated latch as an ordinary gate, and the flip-flop as a turnstile.

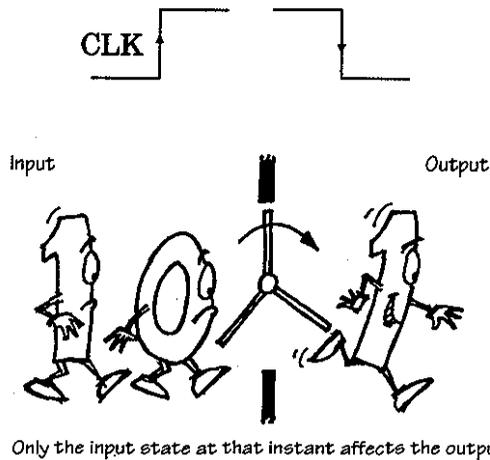
When the gate is closed (enable (EN) is low): $\overline{\text{EN}}$ 



When the gate is open (enable (EN) is high): EN 



When the turnstile turns (at the rising-edge or falling-edge of the clock CLK):



The flip-flop is a **synchronous bistable device**. We use the term **synchronous** here because changes to the output of such a device are synchronised to the triggering edge of the clock input. The logic symbols for three types of edge-triggered flip-flops are shown in Fig 9.20. Notice that the CLK input is indicated differently on the positive edge-triggered and negative edge-triggered flip-flops.

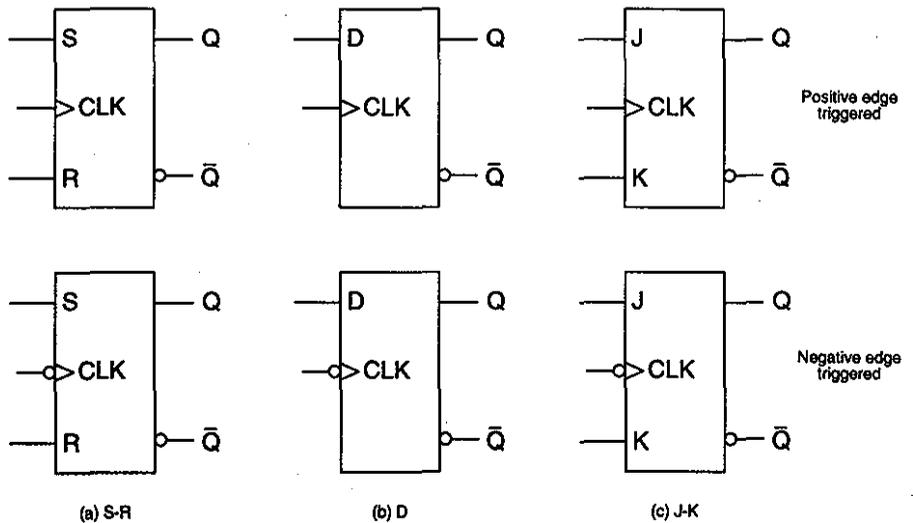


Fig 9.20

The truth tables for each of these flip-flops is shown below.

Truth table for a positive edge-triggered S-R flip-flop.

Inputs			Outputs		Comments
S	R	CLK	Q	\bar{Q}	
0	0	X	Q_0	\bar{Q}_0	no change
0	1	↑	0	1	reset
1	0	↑	1	0	set
1	1	↑	?	?	invalid

↑ = clock transition low to high

X = irrelevant ('don't care')

Q_0 = output level prior to clock transition

Truth table for a positive edge-triggered D flip-flop.

Inputs		Outputs		Comments
D	CLK	Q	\bar{Q}	
1	↑	1	0	set
0	↑	0	1	reset

↑ = clock transition low to high

Truth table for a positive edge-triggered J-K flip-flop.

Inputs			Outputs		Comments
J	K	CLK	Q	\bar{Q}	
0	0	↑	Q_0	\bar{Q}_0	no change
0	1	↑	0	1	reset
1	0	↑	1	0	set
1	1	↑	\bar{Q}_0	Q_0	toggle

↑ = clock transition low to high

Q_0 = output level prior to clock transition

The J-K flip-flop is the most widely used type of flip-flop. You will notice that the J-K flip-flop differs from the S-R flip-flop in that it has no invalid state (which occurs when both the S and R inputs are 1). Instead, when both the J and K inputs are 1, the outputs **toggle**, in other words both the Q and \bar{Q} outputs change state. Fig 9.21 shows a circuit diagram for the J-K flip-flop.

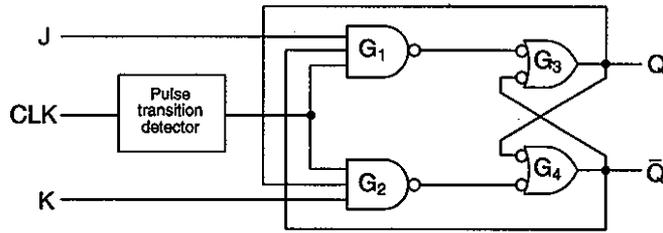


Fig 9.21

The timing diagram of a negative edge-triggered J-K flip-flop is shown in Fig 9.22.

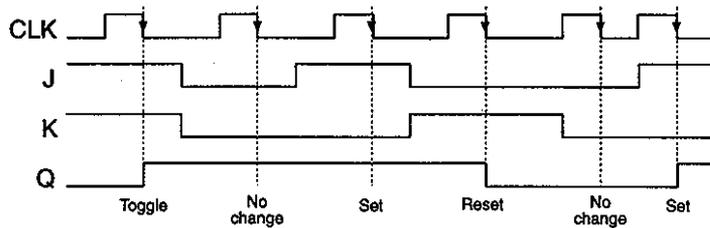


Fig 9.22

Activity 5



For the J-K flip-flop and waveforms shown in Fig 9.23 determine the Q output relative to the clock.

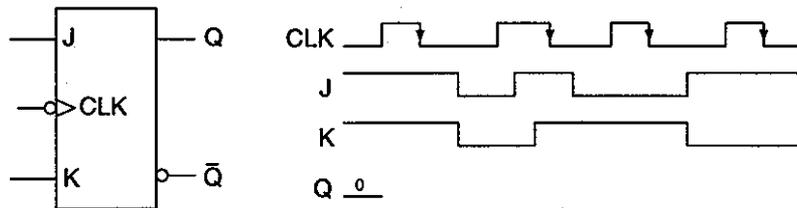


Fig 9.23

2.5 An application of flip-flops

Flip-flops can be used to divide the frequency of a periodic waveform. The J-K flip-flop in Fig 9.24 is connected in the toggle mode, with both the inputs high. The output Q only changes its state on the positive edge of the clock input. As can be seen from the timing diagram, the output changes at half the frequency of the input. This flip-flop can therefore be used as a divide-by-two device.

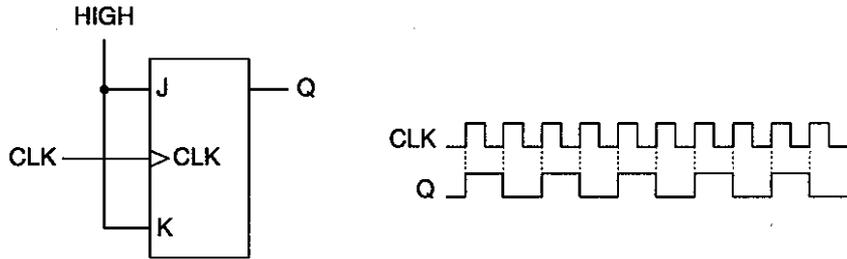


Fig 9.24

Activity 6



Assume that the output of the flip-flop in Fig 9.24 was used to provide the clock input to a similar flip-flop, like the one shown in Fig 9.25. Determine the timing diagram showing the output of the second flip-flop relative to the clock input.

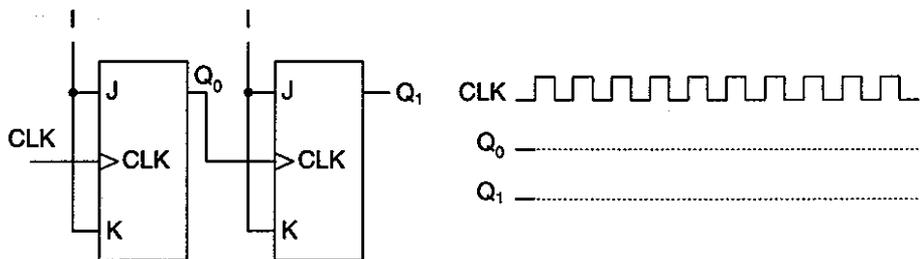


Fig 9.25

3 Shift registers



Shift registers consist of an arrangement of flip-flops, and are used primarily for the storage and transfer of digital data (1s and 0s). The shift register is therefore an important type of memory device.

3.1 Basic shift register functions

The storage capacity of a register is the number of bits (1s and 0s) of digital data that can be held in the register. Each flip-flop in the register represents one bit of storage capacity. The flip-flops are usually referred to as 'stages in the register'. A clock pulse is used to move data into and out of a register, or to shift data from stage to stage in a register. The four basic modes of shift register operation are shown in Fig 9.26.

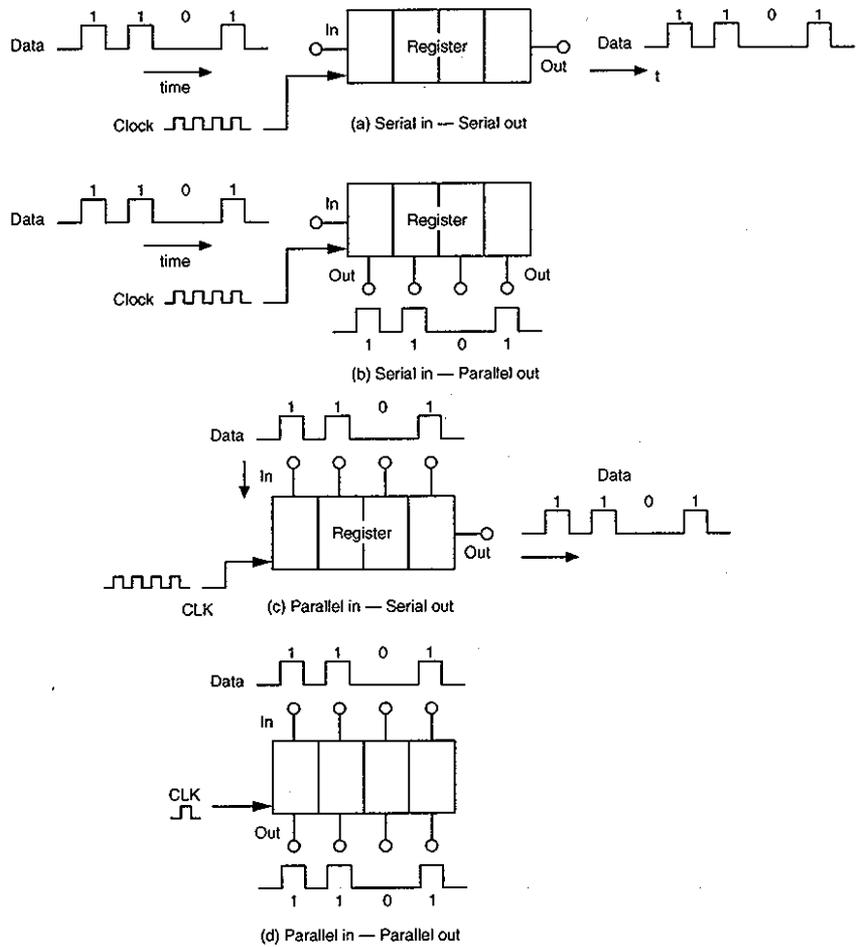


Fig 9.26

While the registers in Fig 9.26 only show serial data being shifted from left to right, serial data may in fact be shifted in both directions (left to right, and right to left).

3.2 Serial to parallel data conversion

Data bits are entered serially into this type of register. Once all the data has been stored, each bit appears on its respective output line. All bits on the parallel output lines are now available to be used at the same time.

Fig 9.27 shows a 4-bit serial in/parallel out shift register.

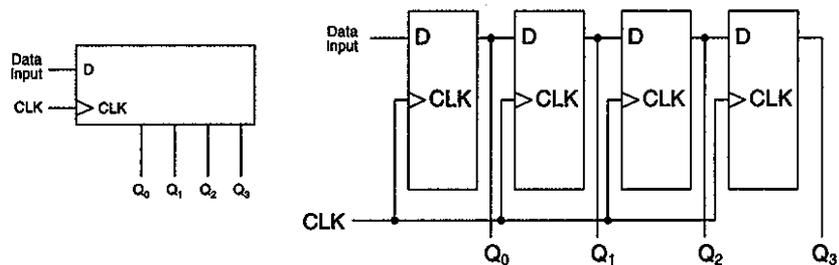


Fig 9.27

Serial entry of data into the shift register happens as is set out below, at the rising edge of each clock pulse. Let's assume that the register is initially clear and the 4-bit 1011 is to be entered into the register, beginning with the rightmost bit.

The 1 is put onto the data input line, making $D = 1$ for flip-flop 0 (FF0). When the first rising edge of the clock pulse is applied, flip-flop 0 is set, thus storing the 1. Next the second bit, which is also a 1, is applied to the data input, making $D = 1$ for FF0, and $D = 1$ for FF1, since the D input of FF1 is connected to the Q_0 output of FF0. When the rising edge of the next (second) clock pulse occurs, both FF0 and FF1 are now set. The third bit, a 0, is now put on the data line, and a clock pulse is applied. The 0 is entered into FF0, the one stored in FF0 is shifted to FF1 and the 1 stored in FF1 is shifted to FF2. Finally, the last bit, a 1, is now applied to the data input, and a clock pulse is applied. All bits are now shifted to the right to adjacent flip-flops. Serial entry of data is now complete and the register can store this information. Data is available at the 4 parallel outputs Q_0 to Q_3 for later retrieval. The state of the outputs after each clock pulse is shown in Fig 9.28.

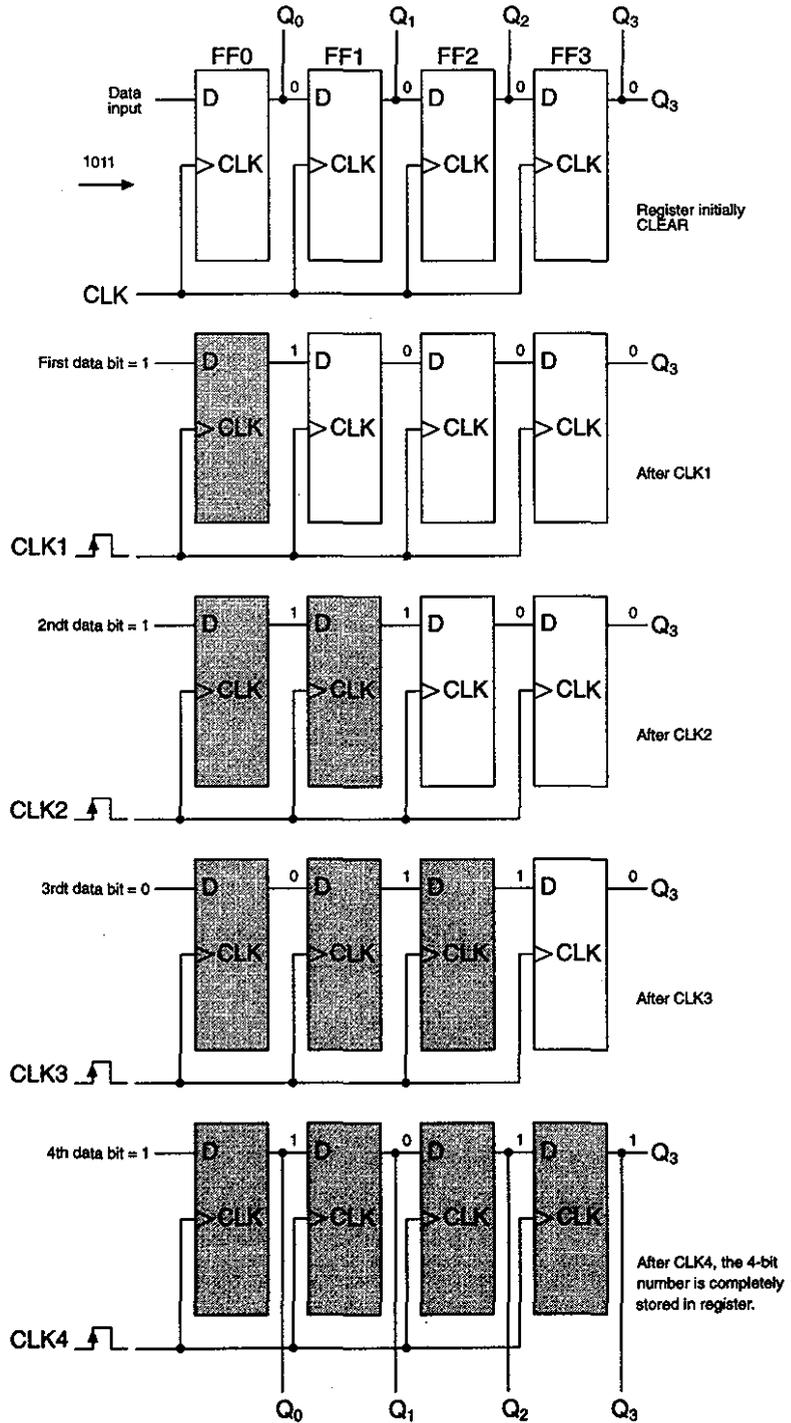


Fig 9.28

Activity 7



A 4-bit register has data inputs and a clock waveform as shown in Fig 9.29. Show the states of the register for each clock pulse. The register initially contains all 1s. Note the order in which the data is entered.

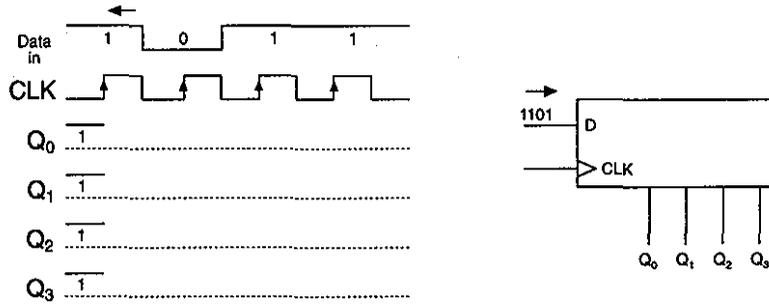


Fig 9.29

Activity 8



Show the states of the 3-bit register shown in Fig 9.30 for the given data input. Assume that the register is initially cleared.

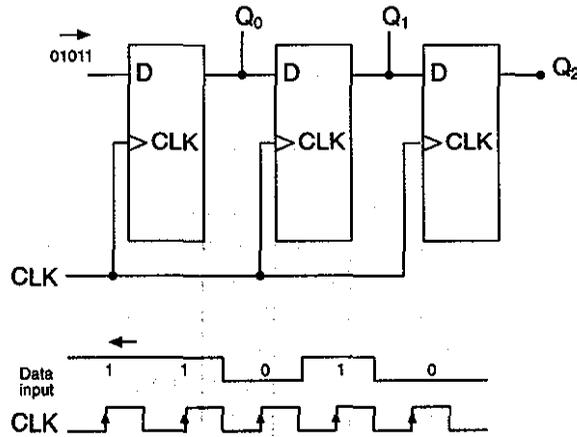


Fig 9.30

3.3 Bi-directional shift registers



A bi-directional shift register allows data to be shifted in both directions, either towards the left or towards the right. You will notice that the basic principle of shift register operation is that the output of one stage should be connected to the input of the next stage. To allow data to be shifted in both directions, we would have to introduce logic circuitry to connect a flip-flop to both the stage following it, as well as to the stage preceding it. The circuit in Fig 9.31 achieves this operation. You will notice that a high on the right/left control input enables gates G_1 to G_4 , effectively connecting the output of a particular stage to the input of the following stage. On the other hand, a low on the right/left input enables G_5 to G_8 , connecting the output of a particular stage to the input of the preceding stage.

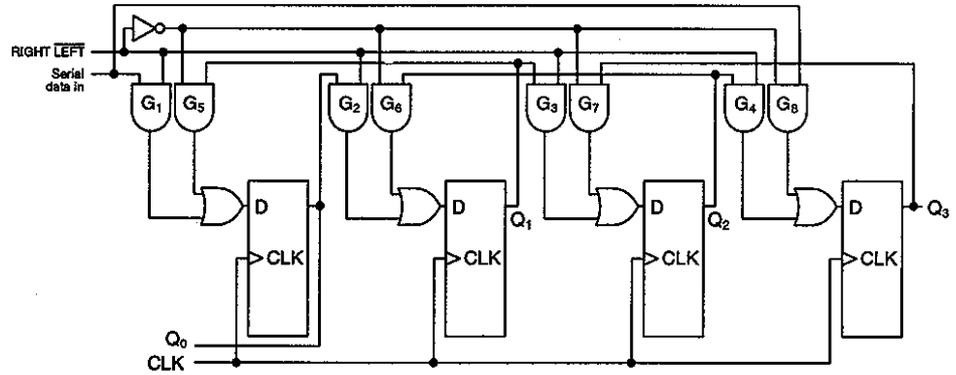


Fig 9.31

The 74LS194A bi-directional universal shift register

By now you may be concerned that the circuitry of these shift registers are becoming more and more complex. You may also be wondering whether you will ever be able to construct such devices from flip-flops and logic gates. Fortunately the 74LS194A has come to our rescue. This is a 4-bit universal shift register, which has both serial and parallel input and output capability and allows you to perform all the basic shift register operations we have discussed so far. The logic block symbol for this universal shift register is shown in Fig 9.32.

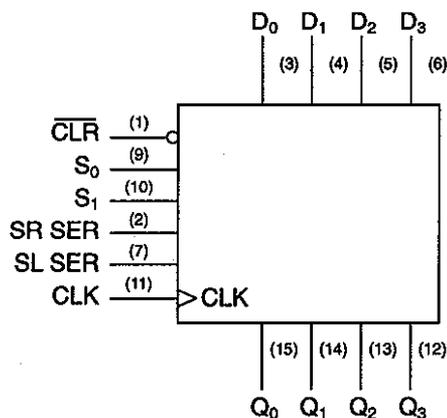


Fig 9.32

In Fig 9.32 D_0 , D_1 , D_2 and D_3 refer to parallel data inputs, while Q_0 , Q_1 , Q_2 and Q_3 refer to parallel data outputs. SR SER is the serial data input when shifting from right to left, while SL SER is the serial data input when shifting from left to right. (SL SER and SR SER will serve as the corresponding outputs.)

S_0 and S_1 refer to mode-control inputs and are activated to operate the shift register as follows:

S_1	S_0	operation
high	high	load parallel data
high	low	shift data left (input serial data)
low	high	shift data right (input serial data)
low	low	hold mode (existing data is retained)

The transfer of parallel and serial data

Parallel data can be transferred very rapidly, because all data bits are moved simultaneously. However, one data line is required for each data bit. **Serial data** requires only one data line for all bits because all the data is moved sequentially, one bit at a time. However, it takes longer to transfer the data. For example, it takes only one clock cycle to parallel transfer eight data bits. For serial transfer of eight data bits, it takes eight clock cycles, but only one data line.

It is more efficient to use combinations of both serial and parallel data transfers. For example, when data has to be transferred over a long distance, it could be more economical to convert parallel data to serial data, then transmit it in that format, and then convert it back to parallel data at the receiving end. Although there is a saving in data lines, the speed of transmission will be slower.

3.4 Shift register counters



In the previous section we used shift registers to shift data from one flip-flop to another. If the data is shifted in a particular sequence, the shift register also acts as a counter. **Shift register counters** are registers with the serial output connected back to the serial input, thereby producing a specific counting sequence. The two most common types of shift register counters are the **Johnson counter** and the **ring counter**.

The Johnson counter

A 4-bit Johnson counter is shown in Fig 9.33, where the complement of the output of the last flip-flop stage is fed back into the D input of the first flip-flop.

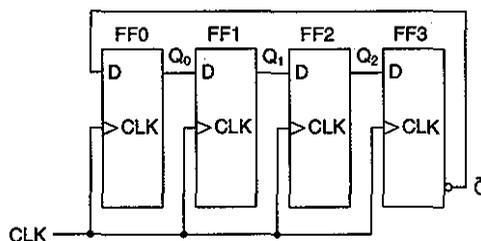


Fig 9.33

The 4-bit Johnson sequence will look like this:

Clock pulse	Q ₀	Q ₁	Q ₂	Q ₃
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

Notice that the 4-bit Johnson counter has eight states, or bit patterns. In general a Johnson counter will have $2N$ count states, where N is the number of flip-flop stages in the counter.

The timing sequence for a 4-bit Johnson counter is shown in Fig 9.34.

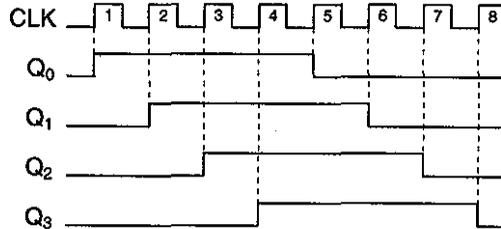


Fig 9.34

The ring counter

A ring counter is similar in inter-stage connections as a Johnson counter, except that Q rather than \bar{Q} is fed back from the last stage.

Activity 9



Derive the timing sequence of a 4-bit ring counter if the output of the first stage flip-flop is 1 and all the other stages are set at 0.

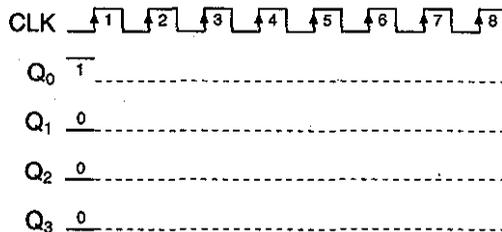


Fig 9.35

Shift register application

The serial in/serial out shift register can be used as a time-delay device. Data at the input of a shift register will be shifted from stage to stage on successive clock pulses until it appears at the output. The time delay between output and input will therefore

be equivalent to the number of stages, multiplied by the period of the clock signal. Fig 9.36 demonstrates how an four-stage shift register using a 1MHz-clock pulse can be used to introduce an $4\ \mu\text{s}$ time delay.

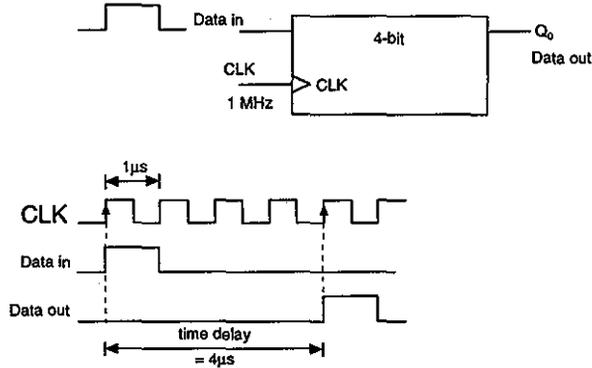


Fig 9.36

4 Asynchronous counters

In the previous section you saw how flip-flops in shift registers may be connected together to perform counting operations. In general, counters can be classified into two broad categories, **asynchronous** and **synchronous** counters, depending on the way they are clocked. In asynchronous counters, also called **ripple counters**, the first flip-flop is clocked by an external clock pulse. The output of this flip-flop then clocks the subsequent flip-flop and so on. That is why we use the term **ripple**, because the change of state in one flip-flop causes changes in the next flip-flop (a ripple effect). In synchronous counters the external clock signal is connected to all the flip-flop stages. It therefore clocks all stages at the same time. Synchronous counter analysis and design will be dealt with in a subsequent course.

4.1 A 3-bit asynchronous binary (ripple) counter

Fig 9.37 shows a 3-bit ripple counter and the waveform it generates. The counter has 8 states due to its 3 flip-flops. Notice that the counter progresses through a binary count of zero through seven and then starts the sequence again from zero. The circuit uses J-K flip-flops. Both the J-K inputs are tied high. The flip-flops are therefore in the toggle mode. This means that the flip-flop changes state or toggles when triggered by the clock signal.

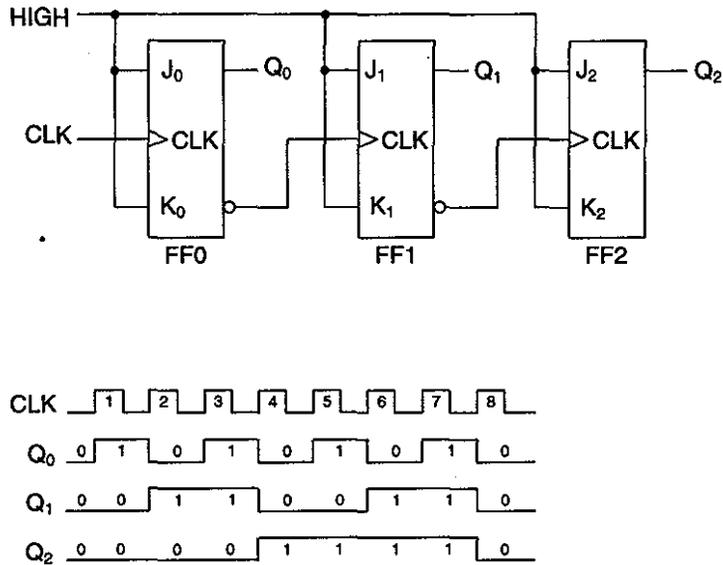


Fig 9.37

Notice also that the output frequency of each flip-flop is one-half the frequency of the previous flip-flop. The same is true for each succeeding flip-flop. In the 3-bit binary counter, the third flip-flop stage divides the input frequency by 2^3 or 8. We therefore call this counter a **divide-by-8 ripple counter**. Should you add another flip-flop stage, it would become a **divide-by-16 counter**, since $2^4 = 16$.

Time delays in ripple counters

The last flip-flop stage in a ripple counter must wait for the input signal to ripple through each preceding flip-flop stage before it can change. Consider the timing diagram of a 3-stage ripple counter in Fig 9.38. This cumulative delay of a ripple counter is a major disadvantage in counter applications because it limits the frequency or rate at which the counter can be clocked. It also creates counter decoding problems, giving rise to glitch conditions. Glitch conditions are unwanted logic levels that last for a very short duration. The maximum cumulative delay in a counter must be less than the period of the clock waveform.



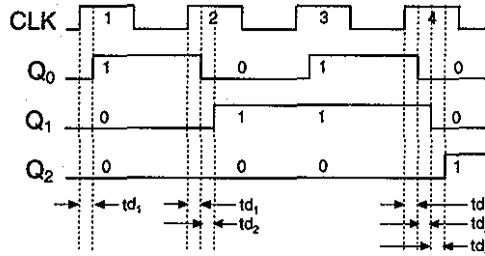


Fig 9.38

If we are assuming typical propagation delay times, $t_{PHL} = t_{PLH}$ is equal to 25 nanoseconds. It would therefore take 3×25 nanoseconds for the counter to change from 011 to 100 after the positive going edge of the clock pulse.

4.2 Divide-by-N ripple counter

The previous section dealt with ripple counters that count in binary to a number equal to $(2^N - 1)$, where N is the number of flip-flops – since the largest binary number that can be displayed for a given number of N flip-flops is equal to $(2^N - 1)$. For instance the 3-bit counter in Fig 9.37 could count up to 7 (111 in binary) since the next state, 000, would reset all the flip-flops, and the count would start once again. Likewise a 4-bit counter would count up to 15 (1111 in binary) before restarting, and so on.

How do we design a ripple counter that can count in binary to any particular number? Suppose we wish to design a counter that would count through five stages from 000 to 100. Such a counter is referred to as a **divide-by-5** or **modulo-5** ripple counter. Clearly we need three flip-flop stages in our design since we are working with a 3-bit number. The method we will use to design such a counter makes use of the **clear** input to the flip-flop. The clear input is an additional input that will reset the flip-flop when an appropriate signal is applied to it.

The count stages are 000, 001, 010, 011, and 100 after which the sequence starts from 000 again. Note that 000 is the first count state and 100 is the fifth state. We need to reset or clear the counter **after** we have identified or decoded the occurrence of the counter output state, 100. It is important that we do not reset or clear the counter when 100 occurs, since this state will then only last for the duration it takes to decode the state and clear

the flip-flops. This could take only nanoseconds, which effectively means that the final stage (100) will be missed. We therefore need to reset the counter at the next number in the sequence, namely 5 (or 101 in binary). We now have to identify the count state 101 using suitable combinational logic.

The count sequence is as follows:

Q ₂	Q ₁	Q ₀
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
*	*	*
*	*	*

We need a way to identify the state 101 uniquely from the previous count states, keeping in mind the reset signal for the flip-flop stages is active low.

A simple truth table can be drawn up for this problem and the solution then becomes apparent.

Q ₂	Q ₁	Q ₀	reset
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
*	*	*	X
*	*	*	X

The last two states are denoted 'don't care' (X) states as these will not occur since the counter is reset on 101. It is also assumed that the counter always starts up in state 000.

Inspection of the table above, reveals that $Q_0 = Q_2 = 1$ is unique for the state 101 and therefore can be used to distinguish it from the counter states 000 to 101. The truth table can be simplified as follows:

Q_2	Q_0	reset
0	0	1
0	1	1
1	0	1
1	1	0

Inspection of the table above shows that the decoding of the counter state 101 can be implemented by using a 2-input NAND logic gate. This circuit and the timing diagram for this counter is shown in Fig 9.39. This method of designing counters is also referred to as the **decode-and-clear** method, as it uses logic circuitry to decode the count and reset the flip-flops once the maximum count has been reached.

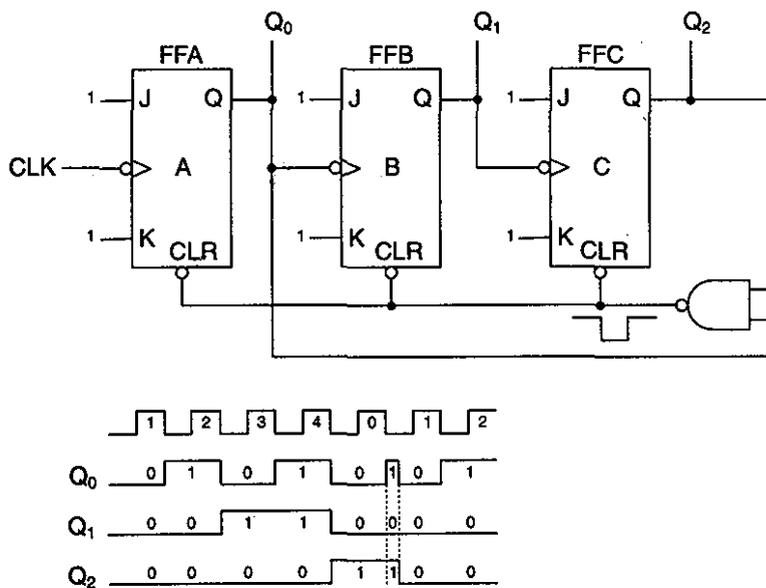


Fig 9.39

4.3 Decoding glitches

From the timing diagram for the counter in Fig 9.39 you will see that it produces a small spike at the 101 count on the FFA stage. This unwanted logic level of very short duration is called a **glitch**. The output of FFC is also a bit longer than it should be. Because they are counted as an extra pulse, the spikes can cause problems when they are used as inputs to other counters or decoders. One way to overcome the problem of glitches is to ensure that the counters or decoders into which these glitches feed, are only enabled after the glitches have disappeared. This method is known as **strobing**.

By way of example: If the counter in Fig 9.39 were connected to a decoder as in Fig 9.40, the effect of the glitches could be eliminated by using the low level of the clock to enable the decoder.

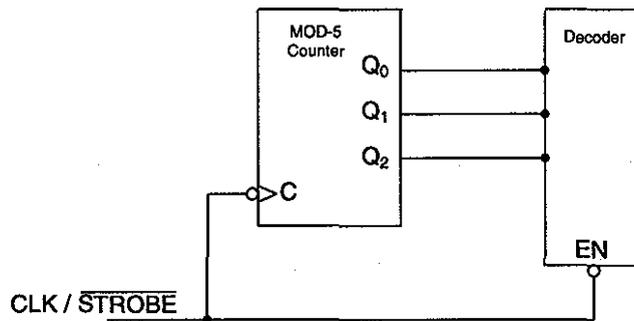


Fig 9.40

Activity 10



Design an asynchronous decade counter. Such a counter is also referred to as a **modulus 10** counter as it counts through ten states from 0000 (decimal 0) to 1001 (decimal 9). Show both the logic diagram and the timing diagram of this counter.

5 Summary

In this unit you learnt that:

- ◇ Latches and flip-flops are logic devices with two outputs, which are always in opposite states. They are also referred to as **one-bit memory elements**.

- ◇ A simple latch can be constructed by interconnecting two NAND gates in a cross coupled manner.
- ◇ A gated latch operates in a similar manner to a latch, except that an enable signal has to be applied in order for the outputs to respond to any changes in the input levels.
- ◇ Flip-flops such as the D and J-K flip-flop can change their state only at the rising or falling edge of a clock input. Such devices are synchronised to changes in the clock input.
- ◇ Flip-flops can be connected together (or cascaded) to form shift registers which are very useful in parallel-to-serial conversion, serial-to-parallel conversion, and as simple counters.
- ◇ Flip-flops, counters and shift registers are called **sequential circuits** as they sequence through a number of different states in response to a clock input.
- ◇ Unlike combinational logic circuits, sequential circuits can retain their state after the inputs which brought about the state, are removed.
- ◇ Sequential circuits are subject to propagation delays that should be taken into account in designing high-speed sequential circuits.

Self-evaluation

Complete the following self-evaluation exercises **without** referring to the unit.

- 1 Using two T flip-flops, design a frequency divider that generates two output signals for these frequencies:
 $f_1 = \frac{f}{2}$ and $f_2 = \frac{f}{4}$, where f is the input frequency.
- 2 Design a 2-bit register that will accumulate data from a previous operation and hold the output data for later use by logic circuits.
- 3 Design a 2-bit serial-to-parallel shift register using D flip-flops.
- 4 Design a modulo-8 synchronous counter.

Answers to activities

Activity 1

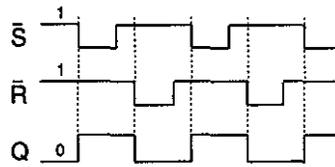


Fig 9.41

Activity 2

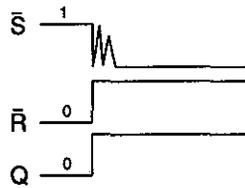


Fig 9.42

Activity 3

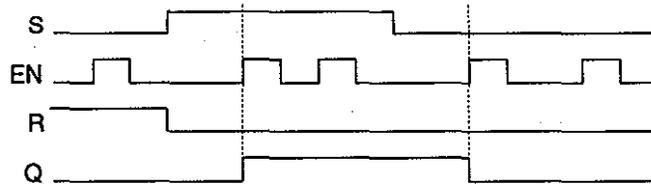


Fig 9.43

Activity 4

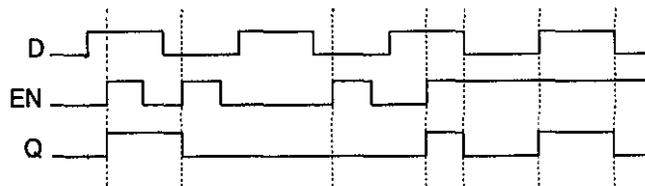


Fig 9.44

Activity 5

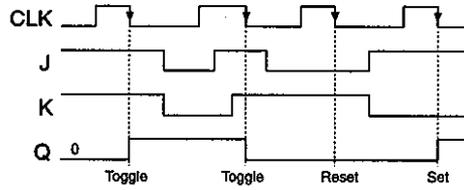


Fig 9.45

Activity 6

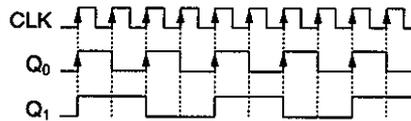


Fig 9.46

Activity 7

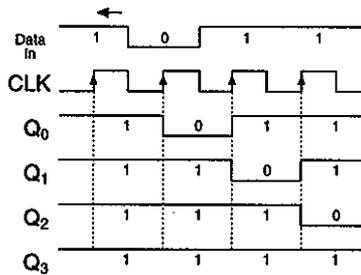


Fig 9.47

Activity 8

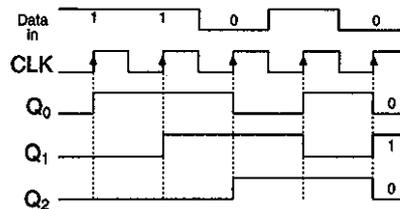


Fig 9.48

Activity 9

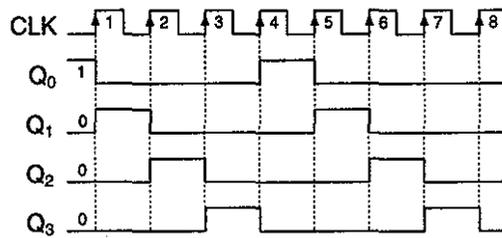


Fig 9.49

Activity 10

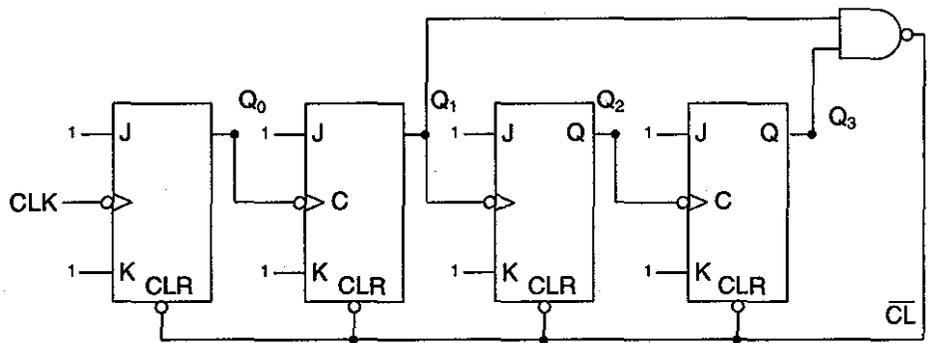


Fig 9.50

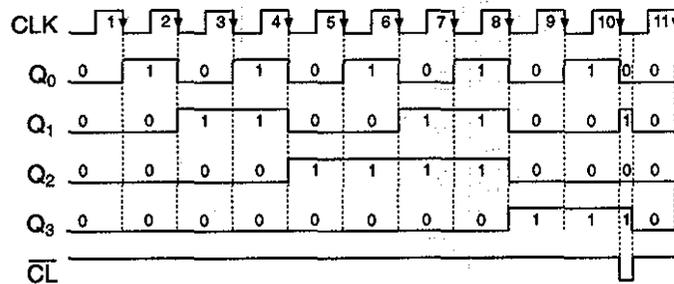
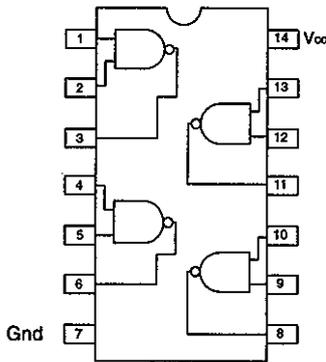


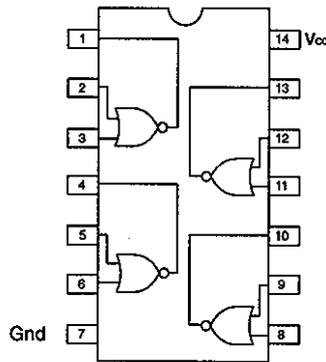
Fig 9.51

Pin outs of digital ICs

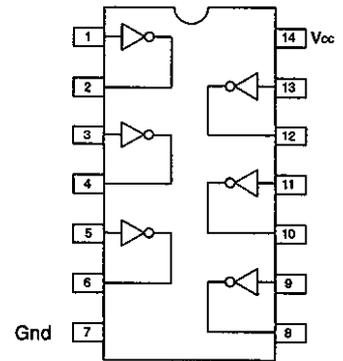
To construct and test or troubleshoot a digital circuit it is necessary to know what each pin on a digital component or integrated circuit (IC) represents. This appendix contains diagrams showing some popular ICs and what each pin on the ICs represents.



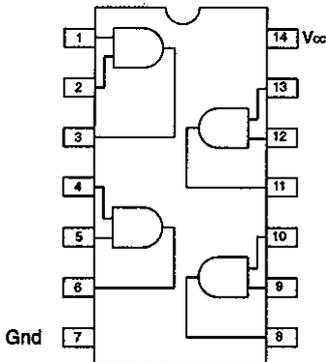
7400 quad 2-input
NAND gate



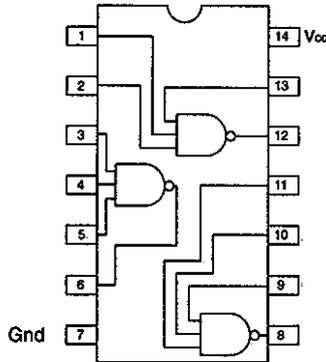
7402 quad 2-input
NOR gate



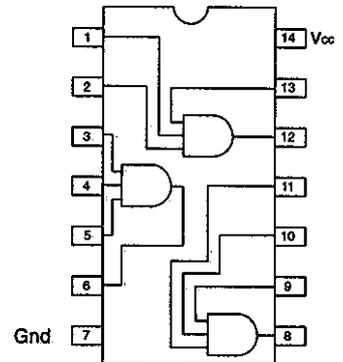
7404 hex INVERTER



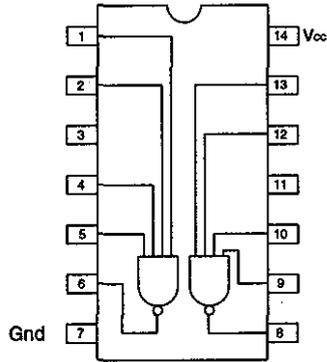
7408 quad 2-input
AND gate



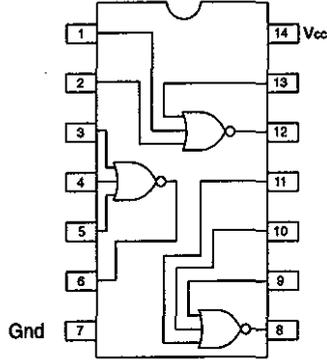
7410 triple 3-input
NAND gate



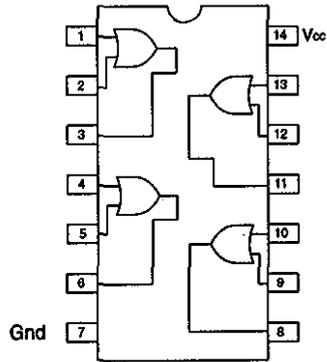
7411 triple 3-input
AND gate



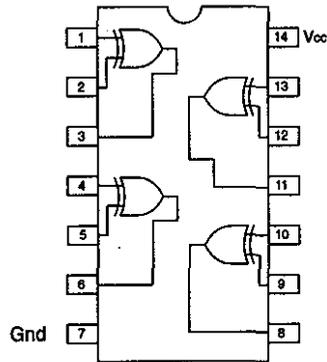
7420 dual 4-input
NAND gate



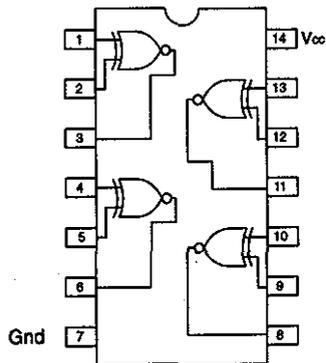
7427 triple 3-input
NOR gate



7432 quad 2-input
OR gate



7486 quad 2-input
Exclusive-OR gate



74810 quad 2-input
Exclusive-NOR
(coincidence) gate

Sample data sheets



DUAL JK FLIP-FLOP WITH SET AND CLEAR

The SN54/74LS76A offers individual J, K, Clock Pulse, Direct Set and Direct Clear inputs. These dual flip-flops are designed so that when the clock goes HIGH, the inputs are enabled and data will be accepted. The Logic Level of the J and K inputs will perform according to the Truth Table as long as minimum set-up times are observed. Input data is transferred to the outputs on the HIGH-to-LOW clock transitions.

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS				OUTPUTS	
	S _D	C _D	J	K	Q	\bar{Q}
Set	L	H	X	X	H	L
Reset (Clear)	H	L	X	X	L	H
*Undetermined	L	L	X	X	H	H
Toggle	H	H	h	h	q	q
Load "0" (Reset)	H	H	l	h	L	H
Load "1" (Set)	H	H	h	l	H	L
Hold	H	H	l	l	q	q

*Both outputs will be HIGH while both S_D and C_D are LOW, but the output states are unpredictable if S_D and C_D go HIGH simultaneously.

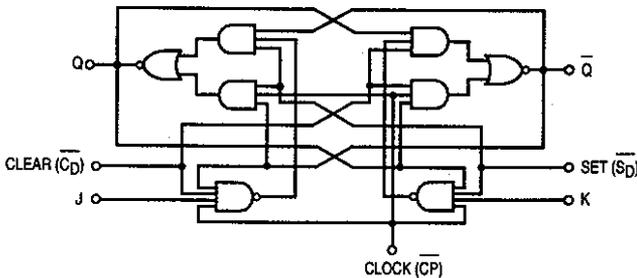
H, h = HIGH Voltage Level

L, l = LOW Voltage Level

X = Immaterial

f, h (q) = Lower case letters indicate the state of the referenced input (or output) one setup time prior to the HIGH-to-LOW clock transition

LOGIC DIAGRAM



SN54/74LS76A

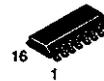
DUAL JK FLIP-FLOP WITH SET AND CLEAR
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

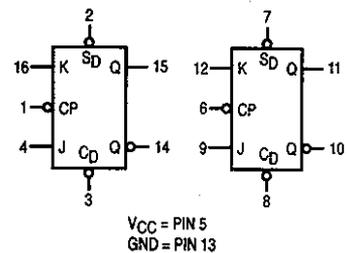


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

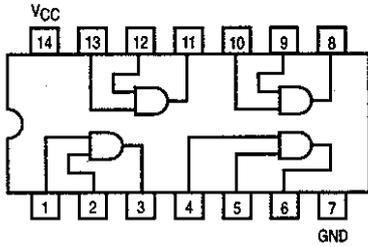
SN54LSXXJ Ceramic
SN74LSXXN Plastic
SN74LSXXD SOIC

LOGIC SYMBOL



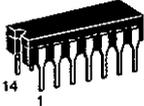


QUAD 2-INPUT AND GATE



SN54/74LS08

**QUAD 2-INPUT AND GATE
LOW POWER SCHOTTKY**



**J SUFFIX
CERAMIC
CASE 632-08**



**N SUFFIX
PLASTIC
CASE 646-06**



**D SUFFIX
SOIC
CASE 751A-02**

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V _{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T _A	Operating Ambient Temperature Range	54	-55	25	125	°C
		74	0	25	70	
I _{OH}	Output Current — High	54, 74			-0.4	mA
I _{OL}	Output Current — Low	54			4.0	mA
		74			8.0	



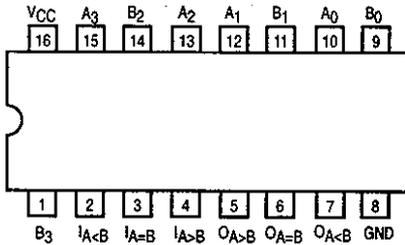
4-BIT MAGNITUDE COMPARATOR

The SN54/74LS85 is a 4-Bit Magnitude Comparator which compares two 4-bit words (A, B), each word having four Parallel Inputs (A_0-A_3, B_0-B_3); A_3, B_3 being the most significant inputs. Operation is not restricted to binary codes, the device will work with any monotonic code. Three Outputs are provided: "A greater than B" ($O_{A>B}$), "A less than B" ($O_{A<B}$), "A equal to B" ($O_{A=B}$). Three Expander Inputs, $I_{A>B}, I_{A<B}, I_{A=B}$, allow cascading without external gates. For proper compare operation, the Expander inputs to the least significant position must be connected as follows: $I_{A<B} = I_{A>B} = L, I_{A=B} = H$. For serial (ripple) expansion, the $O_{A>B}, O_{A<B}$ and $O_{A=B}$ Outputs are connected respectively to the $I_{A>B}, I_{A<B}$, and $I_{A=B}$ Inputs of the next most significant comparator, as shown in Figure 1. Refer to Applications section of data sheet for high speed method of comparing large words.

The Truth Table on the following page describes the operation of the SN54/74LS85 under all possible logic conditions. The upper 11 lines describe the normal operation under all conditions that will occur in a single device or in a series expansion scheme. The lower five lines describe the operation under abnormal conditions on the cascading inputs. These conditions occur when the parallel expansion technique is used.

- Easily Expandable
- Binary or BCD Comparison
- $O_{A>B}, O_{A<B}$, and $O_{A=B}$ Outputs Available

CONNECTION DIAGRAM DIP (TOP VIEW)



NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

PIN NAMES

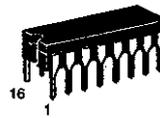
A_0-A_3, B_0-B_3	Parallel Inputs
$I_{A=B}$	A = B Expander Inputs
$I_{A<B}, I_{A>B}$	A < B, A > B, Expander Inputs
$O_{A>B}$	A Greater Than B Output (Note b)
$O_{A<B}$	B Greater Than A Output (Note b)
$O_{A=B}$	A Equal to B Output (Note b)

LOADING (Note a)	
HIGH	LOW
1.5 U.L.	0.75 U.L.
1.5 U.L.	0.75 U.L.
0.5 U.L.	0.25 U.L.
10 U.L.	5 (2.5) U.L.
10 U.L.	5 (2.5) U.L.
10 U.L.	5 (2.5) U.L.

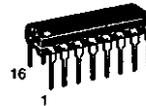
NOTES:
a) 1 TTL Unit Load (U.L.) = 40 μ A HIGH/1.6 mA LOW.
b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

SN54/74LS85

4-BIT MAGNITUDE COMPARATOR LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

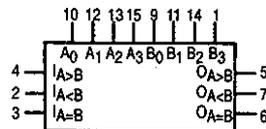


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

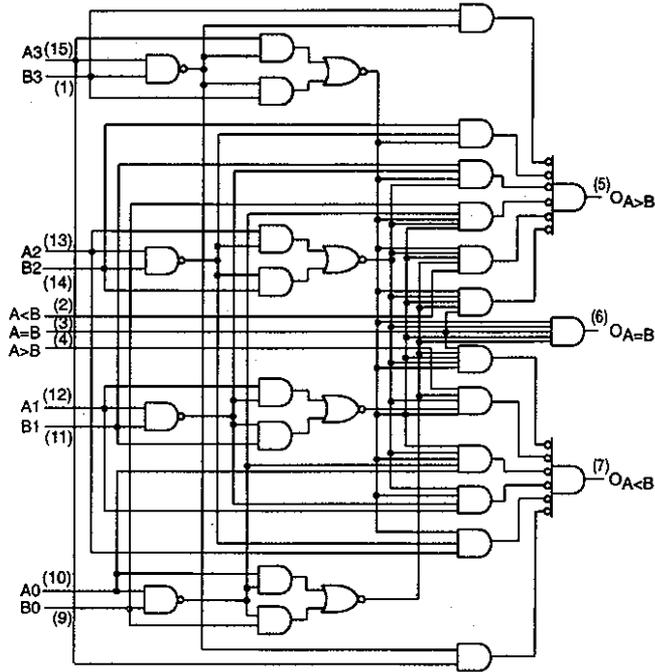
LOGIC SYMBOL



V_{CC} = PIN 16
 GND = PIN 8

SN54/74LS85

LOGIC DIAGRAM



TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A ₃ ,B ₃	A ₂ ,B ₂	A ₁ ,B ₁	A ₀ ,B ₀	I _{A>B}	I _{A<B}	I _{A=B}	O _{A>B}	O _{A<B}	O _{A=B}
A ₃ >B ₃	X	X	X	X	X	X	H	L	L
A ₃ <B ₃	X	X	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ >B ₂	X	X	X	X	X	H	L	L
A ₃ =B ₃	A ₂ <B ₂	X	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ >B ₁	X	X	X	X	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ <B ₁	X	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ >B ₀	X	X	X	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ <B ₀	X	X	X	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	H	L	L	H	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	L	H	L	L	H	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	X	X	H	L	L	H
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	H	H	L	L	L	L
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	L	L	L	H	H	L

H = HIGH Level
L = LOW Level
X = IMMATERIAL

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V _{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T _A	Operating Ambient Temperature Range	54	-55	25	125	°C
		74	0	25	70	
I _{OH}	Output Current — High	54, 74			-0.4	mA
I _{OL}	Output Current — Low	54			4.0	mA
		74			8.0	



BCD TO 7-SEGMENT DECODER/DRIVER

The SN54/74LS47 are Low Power Schottky BCD to 7-Segment Decoder/Drivers consisting of NAND gates, input buffers and seven AND-OR-INVERT gates. They offer active LOW, high sink current outputs for driving indicators directly. Seven NAND gates and one driver are connected in pairs to make BCD data and its complement available to the seven decoding AND-OR-INVERT gates. The remaining NAND gate and three input buffers provide lamp test, blanking input/ripple-blanking output and ripple-blanking input.

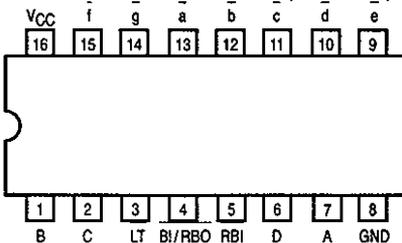
The circuits accept 4-bit binary-coded-decimal (BCD) and, depending on the state of the auxiliary inputs, decodes this data to drive a 7-segment display indicator. The relative positive-logic output levels, as well as conditions required at the auxiliary inputs, are shown in the truth tables. Output configurations of the SN54/74LS47 are designed to withstand the relatively high voltages required for 7-segment indicators.

These outputs will withstand 15 V with a maximum reverse current of 250 μ A. Indicator segments requiring up to 24 mA of current may be driven directly from the SN74LS47 high performance output transistors. Display patterns for BCD input counts above nine are unique symbols to authenticate input conditions.

The SN54/74LS47 incorporates automatic leading and/or trailing-edge zero-blanking control (RBI and RBO). Lamp test (LT) may be performed at any time which the BI/RBO node is a HIGH level. This device also contains an overriding blanking input (BI) which can be used to control the lamp intensity by varying the frequency and duty cycle of the BI input signal or to inhibit the outputs.

- Lamp Intensity Modulation Capability (BI/RBO)
- Open Collector Outputs
- Lamp Test Provision
- Leading/Trailing Zero Suppression
- Input Clamp Diodes Limit High-Speed Termination Effects

CONNECTION DIAGRAM DIP (TOP VIEW)



PIN NAMES

$\bar{A}, \bar{B}, \bar{C}, \bar{D}$	BCD Inputs
\bar{RBI}	Ripple-Blanking Input
\bar{LT}	Lamp-Test Input
\bar{BI}/\bar{RBO}	Blanking Input or Ripple-Blanking Output
—	Ripple-Blanking Output
a, to g	Outputs

LOADING (Note a)

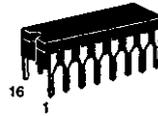
	HIGH	LOW
$\bar{A}, \bar{B}, \bar{C}, \bar{D}$	0.5 U.L.	0.25 U.L.
\bar{RBI}	0.5 U.L.	0.25 U.L.
\bar{LT}	0.5 U.L.	0.25 U.L.
\bar{BI}/\bar{RBO}	0.5 U.L.	0.75 U.L.
—	1.2 U.L.	2.0 U.L.
a, to g	Open-Collector	15 (7.5) U.L.

NOTES:

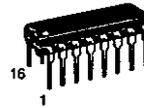
- a) 1 Unit Load (U.L.) = 40 μ A HIGH, 1.6 mA LOW.
 b) Output current measured at $V_{OUT} = 0.5$ V
 The Output LOW drive factor is 7.5 U.L. for Military (54) and 15 U.L. for Commercial (74) Temperature Ranges.

SN54/74LS47

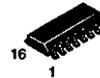
BCD TO 7-SEGMENT DECODER/DRIVER LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

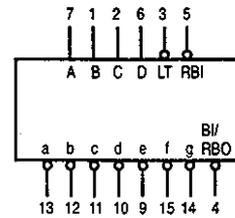


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

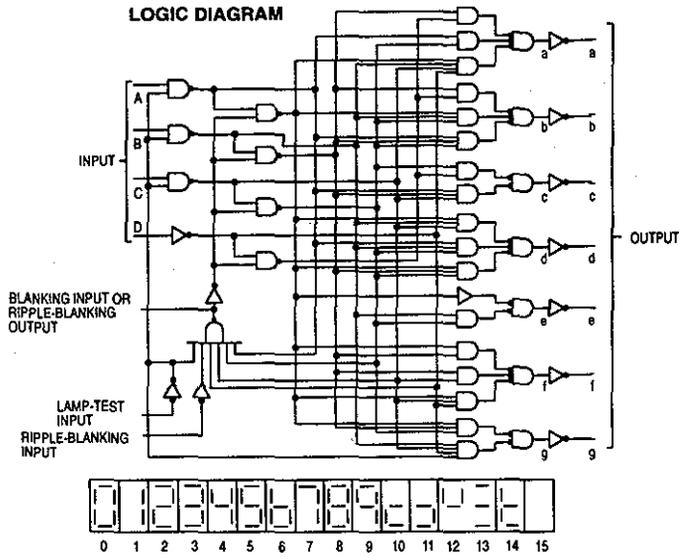
SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

SN54/74LS47



NUMERICAL DESIGNATIONS — RESULTANT DISPLAYS

TRUTH TABLE

DECIMAL OR FUNCTION	INPUTS							OUTPUTS							NOTE
	LT	RBI	D	C	B	A	B/RBO	a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	A
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	A
2	H	X	L	L	H	L	H	L	L	H	L	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	L
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	L
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	L
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	L
B _i	X	X	X	X	X	X	L	H	H	H	H	H	H	H	B
RBI	H	L	L	L	L	L	L	H	H	H	H	H	R	H	C
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	D

H = HIGH Voltage Level
 L = LOW Voltage Level
 X = Immaterial

NOTES:

- (A) B/RBO is wire-AND logic serving as blanking input (BI) and/or ripple-blanking output (RBO). The blanking out (BI) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input (RBI) must be open or at a HIGH level if blanking of a decimal 0 is not desired. X = input may be HIGH or LOW.
- (B) When a LOW level is applied to the blanking input (forced condition) all segment outputs go to a LOW level regardless of the state of any other input condition.
- (C) When ripple-blanking input (RBI) and inputs A, B, C, and D are at LOW level, with the lamp test input at HIGH level, all segment outputs go to a HIGH level and the ripple-blanking output (RBO) goes to a LOW level (response condition).
- (D) When the blanking input/ripple-blanking output (BI/RBO) is open or held at a HIGH level, and a LOW level is applied to lamp test input, all segment outputs go to a LOW level.

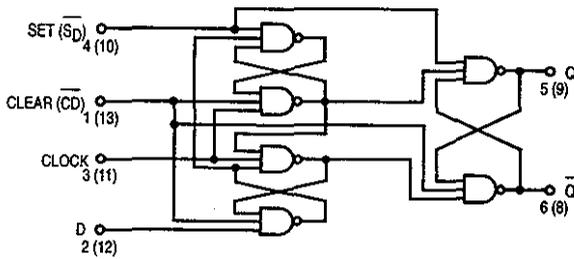


DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

The SN54/74LS74A dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and Q outputs.

Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.

LOGIC DIAGRAM (Each Flip-Flop)



MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	S _D	S _D	D	Q	Q
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

* Both outputs will be HIGH while both S_D and C_D are LOW, but the output states are unpredictable if S_D and C_D go HIGH simultaneously. If the levels at the set and clear are near V_{IL} maximum then we cannot guarantee to meet the minimum level for V_{OH}.

H, h = HIGH Voltage Level

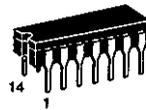
L, l = LOW Voltage Level

X = Don't Care

i, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

SN54/74LS74A

DUAL D-TYPE POSITIVE
EDGE-TRIGGERED FLIP-FLOP
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06

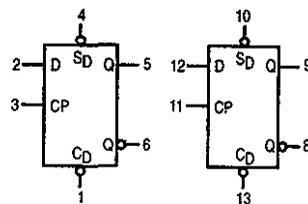


D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ Ceramic
SN74LSXXN Plastic
SN74LSXXD SOIC

LOGIC SYMBOL



V_{CC} = PIN 14
GND = PIN 7

SN54/74LS74A

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V _{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T _A	Operating Ambient Temperature Range	54	-55	25	125	°C
		74	0	25	70	
I _{OH}	Output Current — High	54, 74			-0.4	mA
I _{OL}	Output Current — Low	54			4.0	mA
		74			8.0	

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V _{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V _{IL}	Input LOW Voltage	54		0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74		0.8		
V _{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V	V _{CC} = MIN, I _{IN} = -18 mA
V _{OH}	Output HIGH Voltage	54	2.5	3.5	V	V _{CC} = MIN, I _{OH} = MAX, V _{IN} = V _{IH} or V _{IL} per Truth Table
		74	2.7	3.5	V	
V _{OL}	Output LOW Voltage	54, 74	0.25	0.4	V	I _{OL} = 4.0 mA
		74	0.35	0.5	V	I _{OL} = 8.0 mA
I _{IH}	Input High Current Data, Clock Set, Clear			20 40	μA	V _{CC} = MAX, V _{IN} = 2.7 V
	Data, Clock Set, Clear			0.1 0.2	mA	V _{CC} = MAX, V _{IN} = 7.0 V
I _{IL}	Input LOW Current Data, Clock Set, Clear			-0.4 -0.8	mA	V _{CC} = MAX, V _{IN} = 0.4 V
I _{OS}	Output Short Circuit Current (Note 1)	-20		-100	mA	V _{CC} = MAX
I _{CC}	Power Supply Current			8.0	mA	V _{CC} = MAX

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C, V_{CC} = 5.0 V)

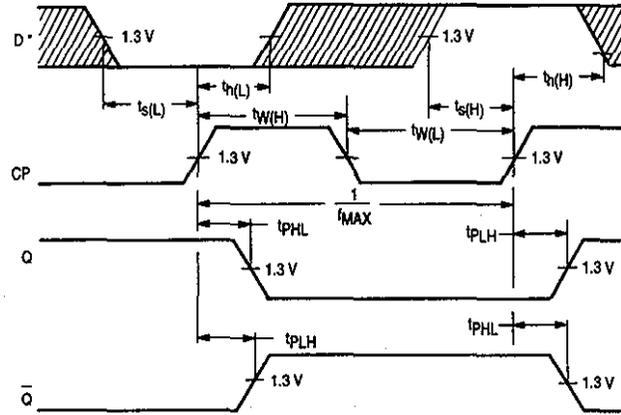
Symbol	Parameter	Limits			Unit	Test Conditions	
		Min	Typ	Max			
t _{MAX}	Maximum Clock Frequency	25	33		MHz	V _{CC} = 5.0 V C _L = 15 pF	
t _{PLH} t _{PHL}	Clock, Clear, Set to Output		13	25	ns		Figure 1
			25	40	ns		

AC SETUP REQUIREMENTS (T_A = 25°C)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t _{W(H)}	Clock	25			ns	V _{CC} = 5.0 V
t _{W(L)}	Clear, Set	25			ns	
t _s	Data Setup Time — HIGH LOW	20			ns	
		20			ns	
t _h	Hold Time	5.0			ns	

SN54/74LS74A

AC WAVEFORMS



*The shaded areas indicate when the input is permitted to change for predictable output performance.

Figure 1. Clock to Output Delays, Data Set-Up and Hold Times, Clock Pulse Width

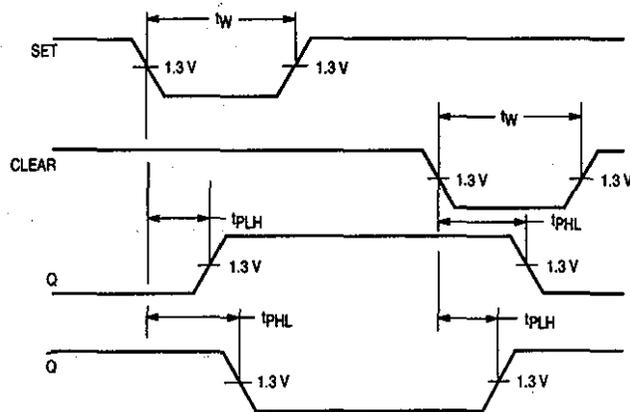


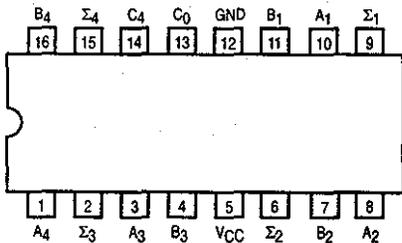
Figure 2. Set and Clear to Output Delays, Set and Clear Pulse Widths



4-BIT BINARY FULL ADDER WITH FAST CARRY

The SN54/74LS83A is a high-speed 4-Bit binary Full Adder with internal carry lookahead. It accepts two 4-bit binary words (A_1-A_4 , B_1-B_4) and a Carry Input (C_0). It generates the binary Sum outputs $\Sigma_1-\Sigma_4$ and the Carry Output (C_4) from the most significant bit. The LS83A operates with either active HIGH or active LOW operands (positive or negative logic). The SN54/74LS283 is recommended for new designs since it is identical in function with this device and features standard corner power pins.

CONNECTION DIAGRAM DIP (TOP VIEW)



NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

PIN NAMES

- A_1-A_4 Operand A Inputs
- B_1-B_4 Operand B Inputs
- C_0 Carry Input
- $\Sigma_1-\Sigma_4$ Sum Outputs (Note b)
- C_4 Carry Output (Note b)

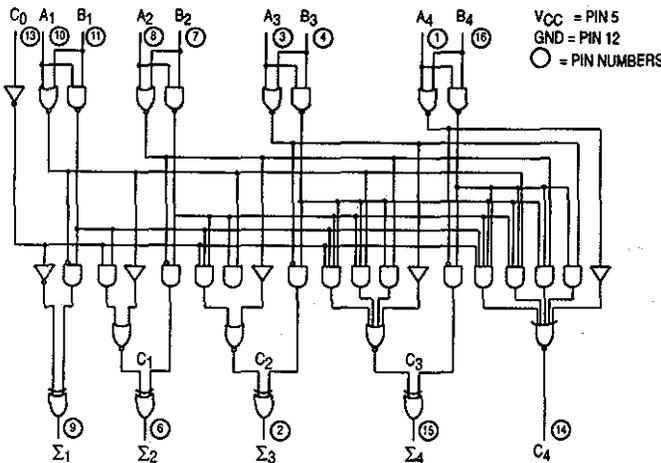
LOADING (Note a)

	HIGH	LOW
A_1-A_4	1.0 U.L.	0.5 U.L.
B_1-B_4	1.0 U.L.	0.5 U.L.
C_0	0.5 U.L.	0.25 U.L.
$\Sigma_1-\Sigma_4$	10 U.L.	5 (2.5) U.L.
C_4	10 U.L.	5 (2.5) U.L.

NOTES:

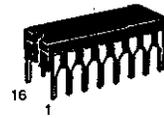
- a) 1 TTL Unit Load (U.L.) = 40 μ A HIGH/1.6 mA LOW.
- b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM

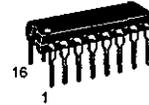


SN54/74LS83A

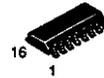
4-BIT BINARY FULL ADDER WITH FAST CARRY LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

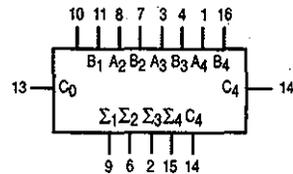


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

- SN54LSXXJ Ceramic
- SN74LSXXN Plastic
- SN74LSXXD SOIC

LOGIC SYMBOL



SN54/74LS83A

FUNCTIONAL DESCRIPTION

The LS83A adds two 4-bit binary words (A plus B) plus the incoming carry. The binary sum appears on the sum outputs ($\Sigma_1 - \Sigma_4$) and outgoing carry (C_4) outputs.

$$C_0 + (A_1+B_1)+2(A_2+B_2)+4(A_3+B_3)+8(A_4+B_4) = \Sigma_1+2\Sigma_2+4\Sigma_3+8\Sigma_4+16C_4$$

Where: (+) = plus

Due to the symmetry of the binary add function the LS83A can be used with either all inputs and outputs active HIGH (positive logic) or with all inputs and outputs active LOW (negative logic). Note that with active HIGH inputs, Carry Input can not be left open, but must be held LOW when no carry in is intended.

Example:

	C ₀	A ₁	A ₂	A ₃	A ₄	B ₁	B ₂	B ₃	B ₄	Σ ₁	Σ ₂	Σ ₃	Σ ₄	C ₄	
Logic Levels	L	L	H	L	H	H	L	L	H	H	H	L	L	H	
Active HIGH	0	0	1	0	1	1	0	0	1	1	1	0	0	1	(10+9 = 19)
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0	(carry+5+6 = 12)

Interchanging inputs of equal weight does not affect the operation, thus C₀, A₁, B₁, can be arbitrarily assigned to pins 10, 11, 13, etc.

FUNCTIONAL TRUTH TABLE

C (n-1)	A _n	B _n	Σ _n	C _n
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

C₁ — C₃ are generated internally
 C₀ — is an external input
 C₄ — is an output generated internally

GUARANTEED OPERATING RANGES

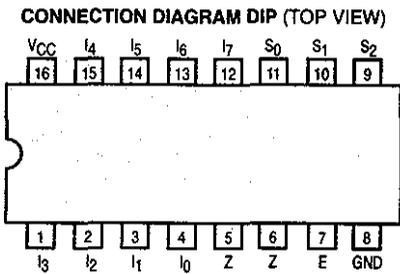
Symbol	Parameter		Min	Typ	Max	Unit
V _{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T _A	Operating Ambient Temperature Range	54	-55	25	125	°C
		74	0	25	70	
I _{OH}	Output Current — High	54, 74			-0.4	mA
I _{OL}	Output Current — Low	54			4.0	mA
		74			8.0	



8-INPUT MULTIPLEXER

The TTL/MSI SN54/74LS151 is a high speed 8-input Digital Multiplexer. It provides, in one package, the ability to select one bit of data from up to eight sources. The LS151 can be used as a universal function generator to generate any logic function of four variables. Both assertion and negation outputs are provided.

- Schottky Process for High Speed
- Multifunction Capability
- On-Chip Select Logic Decoding
- Fully Buffered Complementary Outputs
- Input Clamp Diodes Limit High Speed Termination Effects



PIN NAMES

S ₀ -S ₂	Select Inputs
E	Enable (Active LOW) Input
I ₀ -I ₇	Multiplexer Inputs
Z	Multiplexer Output (Note b)
Z	Complementary Multiplexer Output (Note b)

LOADING (Note a)

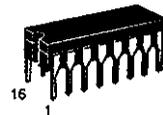
	HIGH	LOW
S ₀ -S ₂	0.5 U.L.	0.25 U.L.
E	0.5 U.L.	0.25 U.L.
I ₀ -I ₇	0.5 U.L.	0.25 U.L.
Z	10 U.L.	5 (2.5) U.L.
Z	10 U.L.	5 (2.5) U.L.

NOTES:

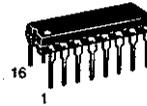
- a) 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
 b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

SN54/74LS151

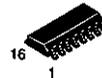
**8-INPUT MULTIPLEXER
LOW POWER SCHOTTKY**



**J SUFFIX
CERAMIC
CASE 620-09**



**N SUFFIX
PLASTIC
CASE 648-08**

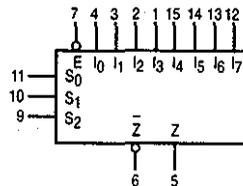


**D SUFFIX
SOIC
CASE 751B-03**

ORDERING INFORMATION

- SN54LSXXXJ Ceramic
- SN74LSXXXN Plastic
- SN74LSXXXD SOIC

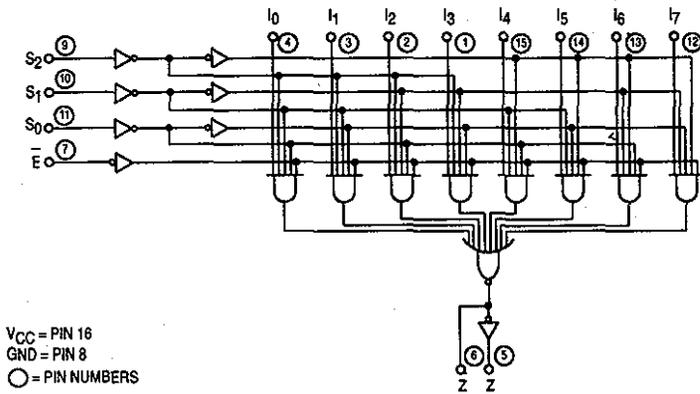
LOGIC SYMBOL



VCC = PIN 16
GND = PIN 8

SN54/74LS151

LOGIC DIAGRAM



FUNCTIONAL DESCRIPTION

The LS151 is a logical implementation of a single pole, 8-position switch with the switch position controlled by the state of three Select inputs, S₀, S₁, S₂. Both assertion and negation outputs are provided. The Enable input (E) is active LOW. When it is not activated, the negation output is HIGH and the assertion output is LOW regardless of all other inputs. The logic function provided at the output is:

$$Z = \bar{E} \cdot (I_0 \cdot \bar{S}_0 \cdot \bar{S}_1 \cdot \bar{S}_2 + I_1 \cdot \bar{S}_0 \cdot \bar{S}_1 \cdot S_2 + I_2 \cdot \bar{S}_0 \cdot S_1 \cdot \bar{S}_2 + I_3 \cdot \bar{S}_0 \cdot S_1 \cdot S_2 + I_4 \cdot S_0 \cdot \bar{S}_1 \cdot \bar{S}_2 + I_5 \cdot S_0 \cdot \bar{S}_1 \cdot S_2 + I_6 \cdot S_0 \cdot S_1 \cdot \bar{S}_2 + I_7 \cdot S_0 \cdot S_1 \cdot S_2)$$

The LS151 provides the ability, in one package, to select from eight sources of data or control information. By proper manipulation of the inputs, the LS151 can provide any logic function of four variables and its negation.

TRUTH TABLE

E	S ₂	S ₁	S ₀	I ₀	I ₁	I ₂	I ₃	I ₄	I ₅	I ₆	I ₇	Z	Z̄
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	H	X	X	X	X	L	X	X	X	L	H
L	H	L	H	X	X	X	X	H	X	X	X	H	L
L	H	H	L	X	X	X	X	X	H	X	X	L	H
L	H	H	H	X	X	X	X	X	X	X	X	H	L
L	H	H	H	X	X	X	X	X	X	X	X	L	H

H = HIGH Voltage Level
 L = LOW Voltage Level
 X = Don't Care

Glossary

Adder – A digital device that adds binary data.

Analog circuit – An electronic circuit that processes analog signals.

Analog signal – An electrical signal that is continuous and varies between certain limits.

AND gate – A logic circuit component that produces a 1 or a **high** output when all its inputs are at a 1 or **high** level.

Astable – Having no stable state.

Asynchronous – Having no fixed time relationships.

Binary number system – A number system in which only two symbols (1 and 0) are used to represent any quantity, for example, 1001, 100010.

Bistable – Having two stable states.

Bit – A single part of binary or digital data, such as, a 1 or a 0.

Boolean algebra – A mathematical notation in which digital circuits may be represented and manipulated.

Boolean expression – An equation in Boolean algebra representing a digital circuit.

Byte – A set of bits; usually eight bits make up one byte.

Combinational logic circuit – A logic circuit with many inputs and one or more outputs; the outputs are directly dependent on some logical combination of the inputs.

Comparator – A digital circuit that looks at different digital inputs and indicates whether one may be equal to, greater than, or less than the other.

Data – Information that is processed in a digital system or computer system.

Data sheet – Sheets of data from electronic component manufacturers. These sheets give detailed information on a particular electronic component. Data sheets may be compiled into data books or put onto CD-ROM.

Decoder – A digital device that converts coded information to another form.

Demultiplexer – A digital device that distributes digital data from one source or line to several lines.

Digital circuit – An electronic circuit that processes digital signals.

Digital signal – An electrical signal that has distinct levels for certain fixed time periods.

Encoder – A digital device that converts information into a coded form.

Exclusive-NOR gate – A logic circuit component that produces a 1 or a **high** output when the inputs are not at the same logic level.

Exclusive-OR gate – A logic circuit component that produces a 1 or a **high** output when the inputs are at the same logic level.

Feedback – A portion of the output is connected back to the input.

Flip-flop – A synchronous bistable device.

Integrated circuit (IC) – Electronic component made of semiconductor material. The device consists of a plastic or ceramic package with pins that are internally linked to the various input and output points of the circuitry found within the package.

INVERTER (or NOT gate) – A logic circuit component that produces the opposite binary level to the input level.

Karnaugh map – A chart representing the functioning of a combinational digital circuit. It is used for the simplification of logic circuits.

Latch – A bistable digital device.

Least Significant Bit (LSB) – The bit of a binary number that carries the least value.

Logic level (or logic state) – The state of a digital signal at a certain point in time – it may be **high** or **low**.

Logic symbol – Graphical sketch or diagram of a logic circuit component, for example, of an AND gate.

Monostable – Having only one stable state.

Most Significant Bit (MSB) – The bit of a binary number that carries the highest value.

Multiplexer – A digital device that can convert data from several lines into a single line.

NAND gate – A logic circuit component that produces a 0 or a **low** output when all its inputs are at a 1 or **high** level.

NOR gate – A logic circuit component that produces a 0 or a **low** output when any its inputs is at a 1 or **high** level.

OR gate – A logic circuit component that produces a 1 or a **high** output when any of its inputs is at a 1 or **high** level.

Parity bit – A bit that is attached to a group of information bits to make the total number of 1s even or odd.

Synchronous – Having a fixed time relationship.

Troubleshooting – The process of testing a faulty practical electronic circuit and locating the fault.

Truth table – A table representing the functioning of a digital circuit. The table shows the various inputs and corresponding output logic states of that particular digital circuit.

Universal gates – These logic gates may be used to replace any other logic gate in a digital circuit. These are the NAND and the NOR gates.

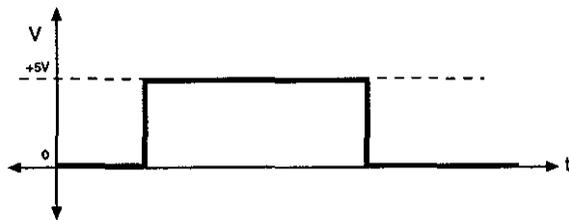
Waveform (or signal) – A representation of an electrical quantity as it varies over time, for example, voltage or current.

Answers to self-evaluation exercises

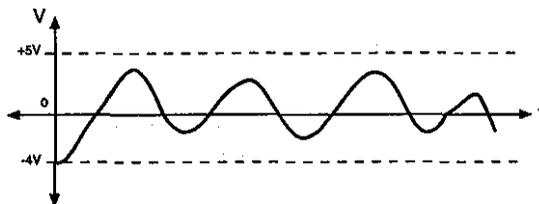
Unit 1

- 1 analog and digital
- 2 true
- 3 An analog signal is continuous and has a range of values, within limits. A digital signal is discrete and has fixed values.

4.1

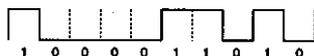


4.2



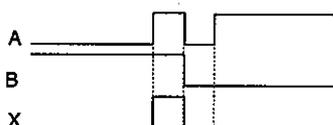
- 5 bit: a single digital logic level
byte: a set of eight bits
- 6 $1024 \div 8 = 128$ bytes
- 7 $8\text{MB} = 8 \times 10^6$ bytes
 $= (8 \times 10^6) \times 8$ bits
 $= 64 \times 10^6$ bits
- 8 Two bytes: 10101100 10100011

9

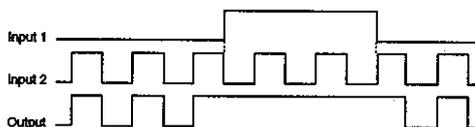


Unit 2

1



2



3 high

4 OR gate;

The door lock can be opened by key 1 OR key 2.

5 The OR gate is functioning correctly. To turn the LED on, a high level must be output from the OR gate. For the output to be high, any one of the inputs must be high.

6 This gate cannot be an INVERTER as an INVERTER has only one input.

OR gate truth table

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

AND gate truth table

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

With Input 1 at 0 level, the output is at a 0 level. Therefore the gate must be an AND gate.

Unit 3

1.1 X = high

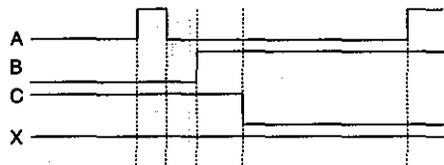
1.2 $2^3 = 8$ rows

1.3

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

1.4 $X = \bar{A} + B + C$

1.5



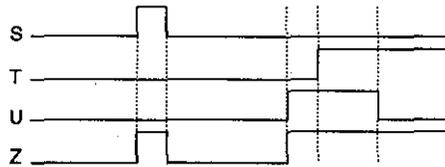
1.6 7404, 7432

2.1 3-input OR gate: $Z = S + T + U$

2.2

S	T	U	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2.3



3.1 $Z = (S + T) + U$

3.2

S	T	U	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- 3.3 Yes. Functionally they are equivalent as they have the same truth table. However, different ICs are used: (1) a 3-input OR gate (2) two 2-input OR gates.
- 4 Yes. They have the same truth table and are functionally equivalent.
- 5.1 Yes. If Q is 0, then the output of the first AND gate must be 0 (irrespective of what the other input is). This 0 is then the input for the second AND gate. The output of the second AND gate must also be 0 (again, irrespective of the other input to this AND gate).

5.2 $T = (P \cdot Q)(R + S)$

P	Q	R	S	T
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

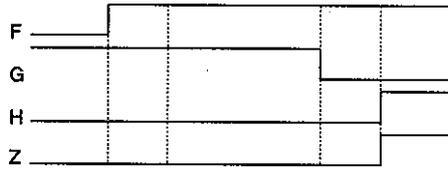
Unit 4

1 $X = \overline{A + B + C}$

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

2 $Z = \overline{(F + G)H}$

F	G	H	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



3 Fig 4.25: 3-input NOR gate IC no. 7427

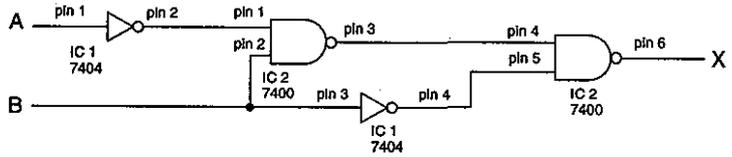
Fig 4.26a: 2-input OR gate IC no. 7432

2-input NAND gate IC no. 7400

INVERTER IC no. 7404

4 IC 1 pin 14: +5V; pin 7: 0V

IC 2 pin 14: +5V; pin 7: 0V

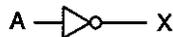


5 No.

J	K	L	$\overline{J + K + L}$	$\overline{J + \overline{K} + \overline{L}}$
0	0	0	1	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	0

These two columns are not the same!

6.1 INVERTER



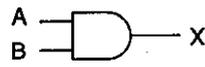
$$X = \overline{A}$$



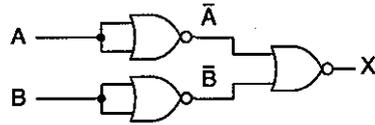
$$X = \overline{A + A}$$

$$= \overline{A}$$

6.2 AND gate



$X = AB$

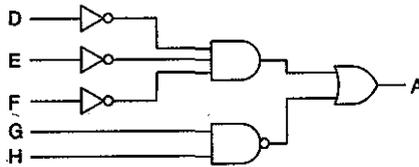


$X = \overline{A + B}$
 $= \overline{A} \cdot \overline{B}$
 $= AB$

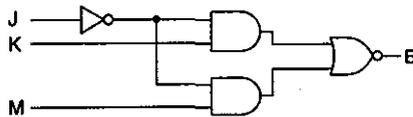
7.1 $F = \overline{ABC} + DE$

7.2 $V = \overline{(R + S)(TU)}$

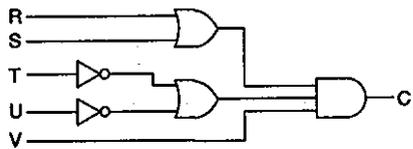
8.1



8.2



8.3



Unit 5

1.1 $W = X$

1.2 $Z = A$

1.3 $Q = I$

1.4 $R = AB + CD$

1.5 $S = BC + AB + \overline{AC}$

1.6 $A = \overline{\overline{XYZ}}$

2.1 left-hand side = $A + B + C + D$
right-hand side = $A + B + C + D$
Proved!

2.2 left-hand side = AB
right-hand side = $AC + \overline{AB}$
Disproved!

2.3 left-hand side = XZ
right-hand side = $XY + XZ$
Disproved!

2.4 left-hand side = 0
right-hand side = 1
Disproved!

3.1 $X = \overline{A}BC + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC$

3.2 $Y = \overline{J}K\overline{L} + \overline{J}KL + \overline{J}K\overline{L} + \overline{J}K\overline{L} + \overline{J}KL$

3.3 $L = \overline{X}Y + X\overline{Y};$
 $M = \overline{X}\overline{Y} + XY$

3.4 $Z = \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + A\overline{B}\overline{C}\overline{D}$

4.1 $X = \overline{A}C + B$

4.2 $Y = \overline{K}\overline{L} + KL + JK$

4.3 The expression in 3.3 cannot be simplified

4.4 $Z = \overline{C}\overline{D} + A\overline{B}CD$

Unit 6

- 1 Use of Boolean algebra, or use of Karnaugh maps.
- 2
 - fewer components used
 - fewer connections between components
 - cheaper
 - more reliable
 - easier to troubleshoot

3.1

	ST				
R		00	01	11	10
0					
1					

3.2

	XY				
VW		00	01	11	10
00					
01					
11					
10					

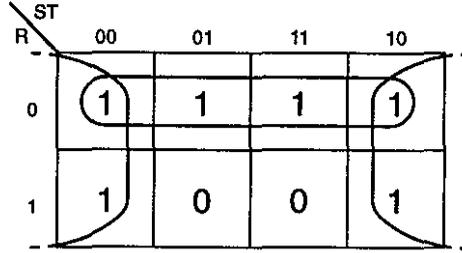
4.1 $Z = 1$

	Y		
X		0	1
0		1	1
1		1	1

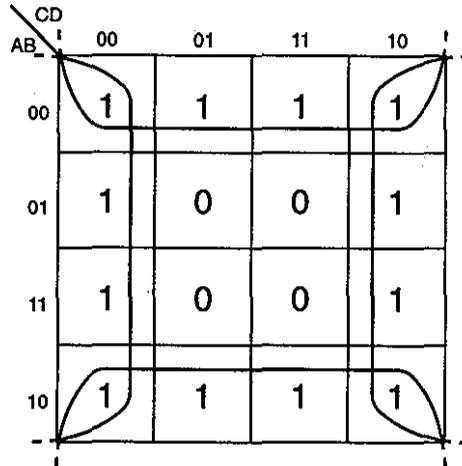
4.2 $X = \overline{A}\overline{C} + \overline{A}B + B\overline{C} + A\overline{B}C$

	BC				
A		00	01	11	10
0		1	0	1	1
1		0	1	0	1

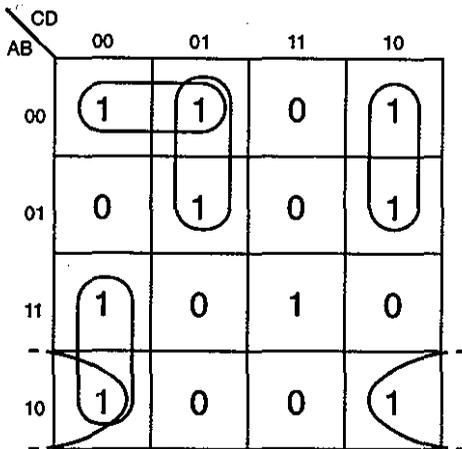
4.3 $X = \overline{R} + \overline{T}$



4.4 $X = \overline{B} + \overline{D}$



4.5 $X = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{C}\overline{D} + \overline{A}\overline{C}\overline{D} + \overline{A}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D}$
 There are also other possible answers.



4.6 $Z = W\bar{X} + \bar{W}X + VX\bar{Y}$

XY \ VW	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	1
10	0	0	1	1

5.1 $X = A + B$

B \ A	0	1
0	0	1
1	1	1

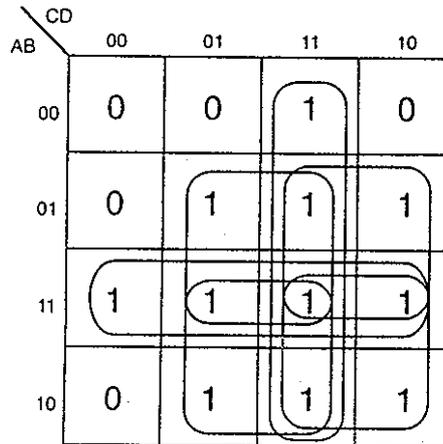
5.2 $Y = \bar{A} + \bar{B} + C$

BC \ A	00	01	11	10
0	1	1	1	1
1	1	1	1	0

- 8** Input variables:
 Guard 1 key – A
 Guard 2 key – B
 Guard 3 key – C
 Guard 4 key – D
 Key inserted: logic 1
 No key: logic 0
 Output variables:
 Small safe – X
 Large safe – Y
 Safe locked: logic 0
 Safe opened: logic 1
 Truth table:

A	B	C	D	X	Y
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

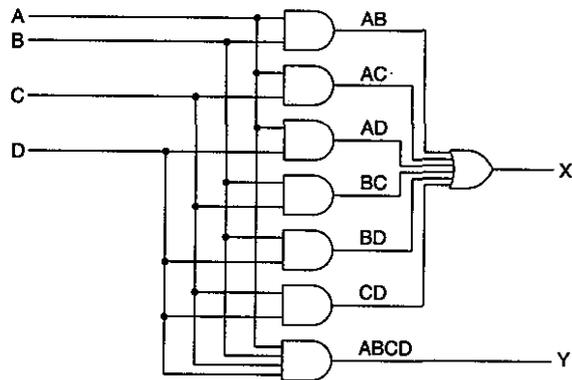
Karnaugh map for X



$$X = AB + AC + AD + BC + BD + CD$$

$$Y = ABCD$$

Logic circuit



9 Input variables:

Judge 1 - A

Judge 2 - B

Judge 3 - C

Judge 4 - D

Product liked: logic 1

Product disliked: logic 0

Output variables:

Lamp 1 good - X

Lamp 2 not good - Y

Lamp 3 tie - Z

Lamp on: logic 1

Lamp off: logic 0

Truth table:

A	B	C	D	X	Y	Z
0	0	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

Karnaugh map for X

		CD			
		00	01	11	10
AB	00	0	0	0	0
	01	0	0	1	0
	11	0	1	1	1
	10	0	0	1	0

$$X = BCD + ABD + ABC + ACD$$

Karnaugh map for Y

CD AB	00	01	11	10
00	1	1	0	1
01	1	0	0	0
11	0	0	0	0
10	1	0	0	0

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

Karnaugh map for Z

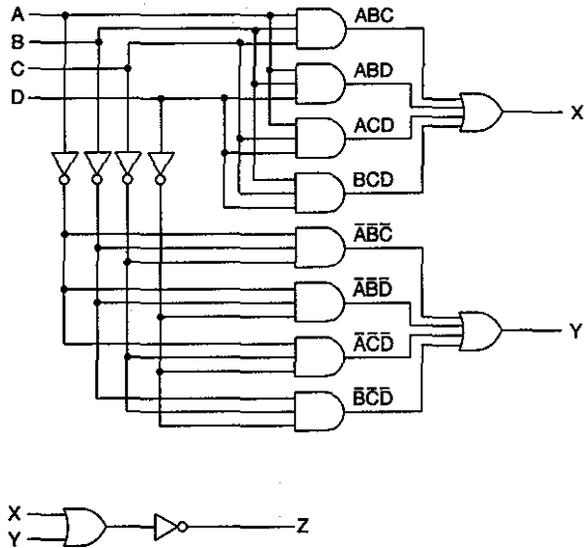
CD AB	00	01	11	10
00	0	0	1	0
01	0	1	0	1
11	1	0	0	0
10	0	1	0	1

No looping possible

Another solution for Z would be: The decision is a tie if it is: NOT [(good) OR (not good)], i.e. $Z = \text{NOT}(X \text{ OR } Y)$

Therefore, $Z = \overline{X + Y}$

Logic circuits

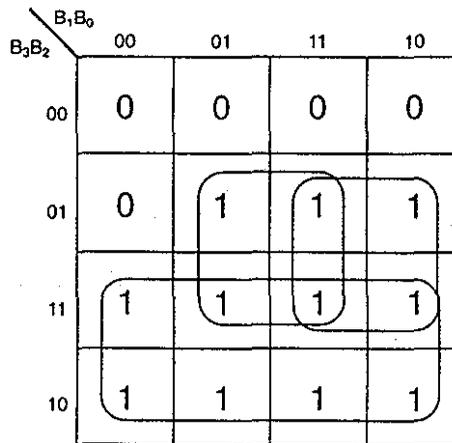


- 10** Input variables:
 LSB of binary number – B_0
 Next bit of binary number – B_1
 Third bit of binary number – B_2
 MSB of binary number – B_3

Output variables:
 X – high if binary number
 $B_3 B_2 B_1 B_0 \geq 0101_2$

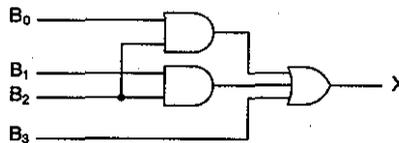
B_3	B_2	B_1	B_0	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh map for X



$$X = B_3 + B_2B_1 + B_2B_0$$

Logic circuit



11 Input variables:

Switch 1 - A

Switch 2 - B

Output variables:

Light - X

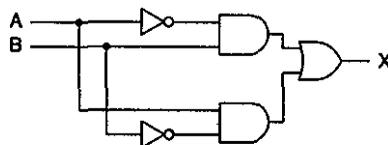
Truth table

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

light off
light on
light on
light off

$$X = \bar{A}B + A\bar{B}$$

Logic circuit

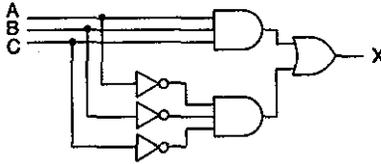


12.1

R	S	T	X
0	0	0	1
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	0

$$\begin{aligned}
 X &= \overline{R}\overline{S}\overline{T} + \overline{R}ST + R\overline{S}\overline{T} + RST \\
 &= \overline{R \oplus S \oplus T}
 \end{aligned}$$

12.2 $X = \overline{A}\overline{B}\overline{C} + ABC$



13.1 Rewrite all the binary numbers with seven bits:

- 0001001₂
- 0010011₂
- 0001001₂
- 1000000₂
- 0011111₂
- 0001011₂

Now arrange them in order of increasing magnitude:

- 0001001₂
- 0001001₂
- 0001011₂
- 0010011₂
- 0011111₂
- 1000000₂

- 13.2
- 0001001₂ = 9₁₀
 - 0010011₂ = 19₁₀
 - 0001001₂ = 9₁₀
 - 1000000₂ = 64₁₀
 - 0011111₂ = 31₁₀
 - 0001011₂ = 11₁₀

In order of increasing magnitude:

- 9₁₀ 9₁₀ 11₁₀ 19₁₀ 31₁₀ and 64₁₀

13.3 First number 1001_2 : A

Second number 10011_2 : B

To compare the numbers, use 5 bits each.

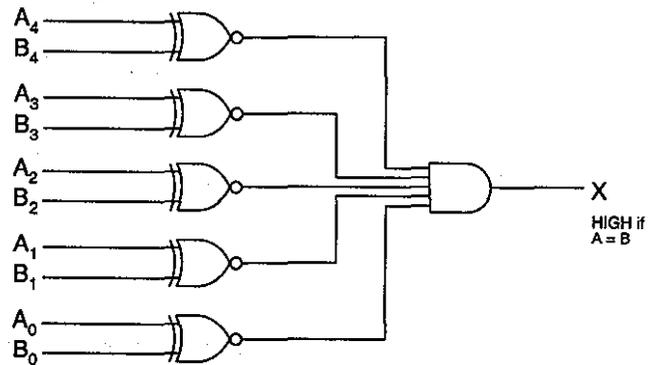
A written as $A_4 A_3 A_2 A_1 A_0$ with A_4 the MSB.

B written as $B_4 B_3 B_2 B_1 B_0$ with B_4 the MSB.

A = B if:

$A_4 = B_4$ and $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 = B_0$

Logic circuit



A > B if:

$(A_4 > B_4)$ or

$(A_4 = B_4)$ and $(A_3 > B_3)$ or

$(A_4 = B_4)$ and $(A_3 = B_3)$ and $(A_2 > B_2)$ or

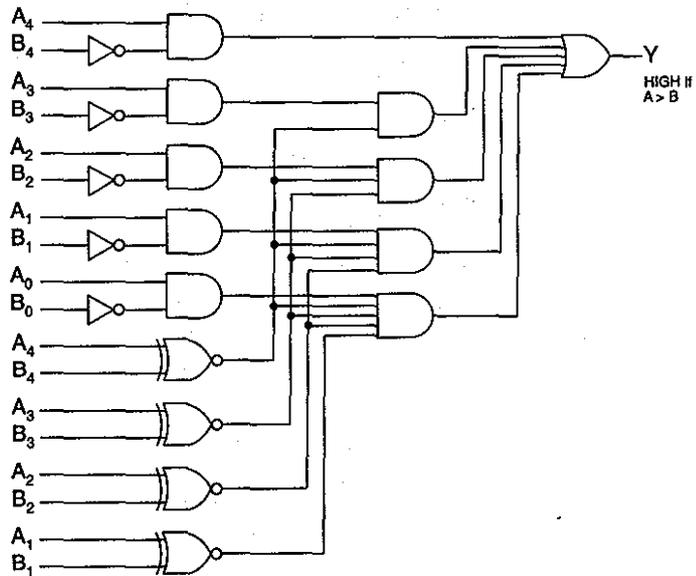
$(A_4 = B_4)$ and $(A_3 = B_3)$ and $(A_2 = B_2)$ and $(A_1 > B_1)$

or

$(A_4 = B_4)$ and $(A_3 = B_3)$ and $(A_2 = B_2)$ and $(A_1 = B_1)$

and $(A_0 > B_0)$.

Logic circuit



- 14 Number A written as A₃ A₂ A₁ A₀ with A₃ the MSB.
 Number B written as B₃ B₂ B₁ B₀ with B₃ the MSB.

A < B if:

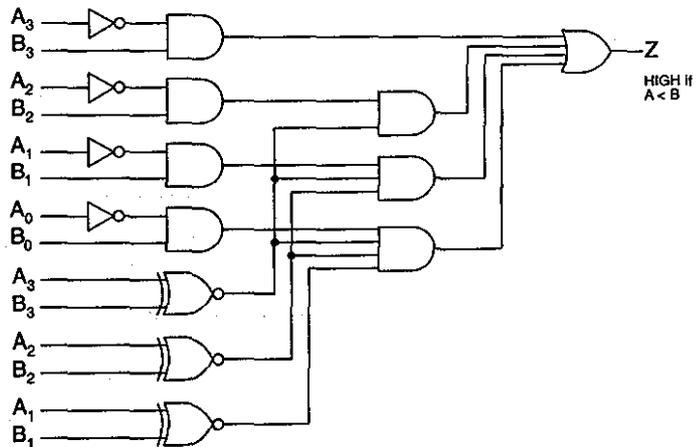
(A₃ < B₃) or

(A₃ = B₃) and (A₂ < B₂) or

(A₃ = B₃) and (A₂ = B₂) and (A₁ < B₁) or

(A₃ = B₃) and (A₂ = B₂) and (A₁ = B₁) and (A₀ < B₀)

Logic circuit



Unit 8

1.1 101100100_2

1.2 0001 1001 0101 0110 (BCD)

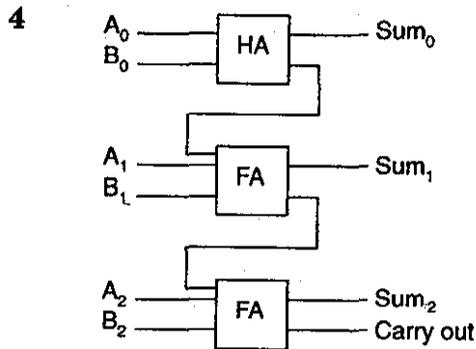
1.3 164_{16}

2.1 3775_{10}

2.2 0011 0111 0111 0101 (BCD)

3.1 $50A_{16}$

3.2 11001_2



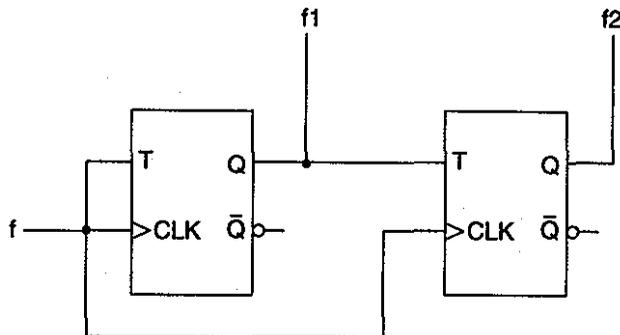
5 Decoder: Converts a code, for example, binary to a specific value like decimal.

Encoder: Converts a number like decimal to a code like binary.

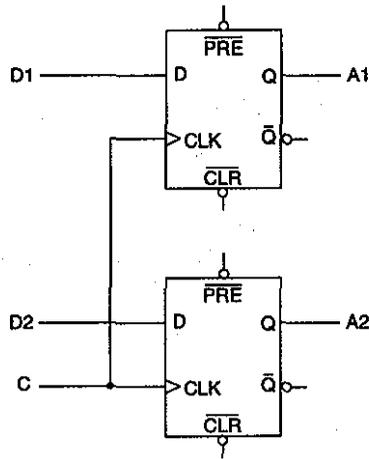
6 Data selection

Unit 9

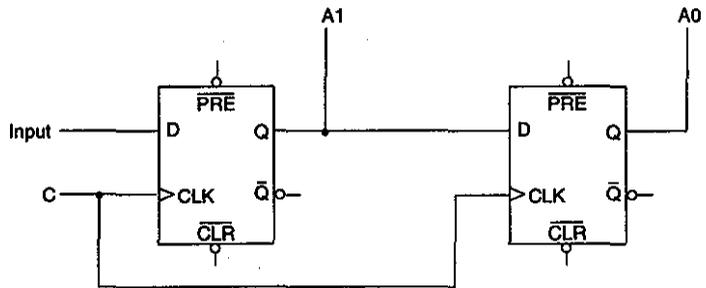
1



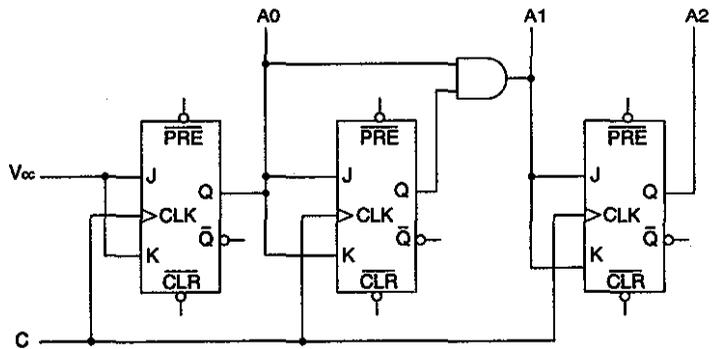
2 Tie preset and clear input to Vcc.



3 Tie preset and clear inputs to Vcc.



4 Tie preset and clear input to Vcc.



Recommended reading

- ◇ *Digital Fundamentals* by Thomas L. Floyd.
Macmillan College Publishing Company / Prentice Hall
International
ISBN 0-13-228677-7
- ◇ *Introduction to Digital Circuits* by Theodore F. Bogart, Jr.
Glencoe Macmillan / McGraw-Hill
ISBN 0-02-819941-3
- ◇ *Schaum's Outline of Theory and Problems of Digital
Principles* by Roger L. Tokheim
McGraw Hill
ISBN 0-07-065012-8