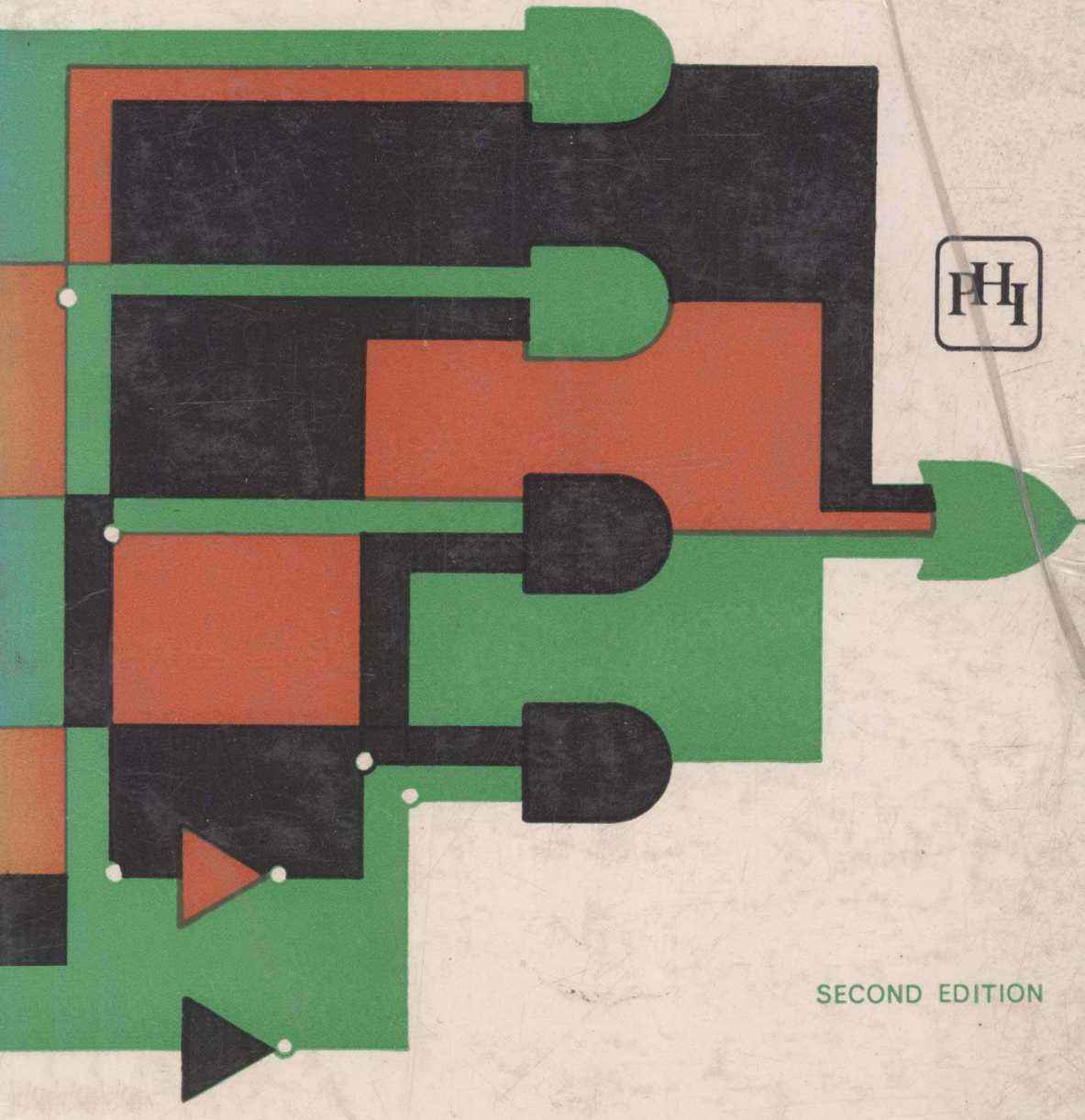
V. RAJARAMAN ■ T. RADHAKRISHNAN

Eastern Economy Edition

An Introduction to Digital Computer Design



PHI

SECOND EDITION

AN INTRODUCTION TO DIGITAL COMPUTER DESIGN

SECOND EDITION

V. RAJARAMAN, Ph.D.
Professor of Electrical Engineering and
Computer Sciences
Indian Institute of Technology
Kanpur

and

T. RADHAKRISHNAN, Ph.D.
Assistant Professor of Computer Sciences,
Concordia University
Montreal, Canada

Prentice-Hall of India Private Limited
New Delhi-110001
1982

Rs. 25.00

AN INTRODUCTION TO DIGITAL COMPUTER DESIGN,
2nd Ed.
by V. Rajaraman and T. Radhakrishnan

PRENTICE-HALL INTERNATIONAL, INC., Englewood Cliffs.
PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi.
PRENTICE-HALL INTERNATIONAL, INC., London.
PRENTICE-HALL OF AUSTRALIA, PTY. LTD., Sydney.
PRENTICE-HALL OF CANADA LTD., Toronto.
PRENTICE-HALL OF JAPAN, INC., Tokyo.
PRENTICE-HALL OF SOUTHEAST ASIA (PTE.) LTD., Singapore.
WHITEHALL BOOKS LIMITED, Wellington, New Zealand.

ISBN-0-87692-021-0

First Printing	(First Edition)	...	...	April, 1976
Second Printing	(Second Edition)	...	...	August, 1978
Third Printing	(„ „)	...	...	March, 1982

Printed by G. D. Makhija at India Offset Press, New Delhi-110064 and Published by Prentice-Hall of India Private Limited, M-97, Connaught Circus New Delhi-110001.

PREFACE

This book is an elementary introduction to digital computer design. It does not assume an extensive knowledge of electronics or mathematics. A student in the final year B.Sc. or B. Tech. would be able to follow the book. A knowledge of programming in FORTRAN would, however, be useful to understand some of the simulation programs given in the book and would also give the proper perspective to appreciate the development of the subject.

After a brief introduction, Chapter II deals with the representation of numbers suitable for manipulation on digital computers. Binary numbers, conversion between number bases, and arithmetic operations are discussed in this chapter. Algorithms (written in English) corresponding to the different procedures are presented wherever suitable. Boolean Algebra is introduced in Chapter III. Karnaugh map and McCluskey chart method for simplifying Boolean expressions are discussed in this chapter. We have found that most students are able to use mechanically the algorithms for simplifying Boolean expressions but they have difficulty in formulating truth tables from word statements. We have thus devoted most of Chapter IV to illustrate this procedure through examples. Synthesis of combinational circuits with NAND/NOR gates is also discussed in this chapter. Chapter V is on flip-flops, registers and counters. We have discussed these keeping in view the advent of integrated circuits. We have thus avoided detailed presentation of sequential circuit synthesis. We have, however, given synthesis procedures for synchronous counters and the design of counter decoders to generate timing signals. With the advent of medium and large scale integrated circuits, the use of microprogramming languages to describe digital systems has become important. We have thus devoted Chapter VI to define a simple microprogramming language. This language is used in the chapter to design some small digital subsystems. It is used again in Chapter X to design a small computer. A course in digital techniques given to undergraduates in Electronics may use only these six chapters as they are self-contained.

From Chapter VII the discussion shifts to the design of general purpose digital computers. In Chapter VII it is shown that computation begins with an algorithm which is expressed in a suitable language. This language is translated by a computer program to a set of its own instructions which are executed by the computer's electronic circuits. This discussion lays the ground work for the later discussion of "software-hardware trade offs" in computer design. Chapter VIII deals with the design of computer memories using magnetic cores and semiconductors. Recent developments in memories, namely, Read Only Memories (ROM) and content addressable memories are also discussed in this Chapter. In Chapter IX the organization of a small computer is presented. A simulator program in FORTRAN is given for this computer to enable students to write machine language programs for this hypothetical machine and test them. The need for having extra instructions in the machine to conveniently solve new problems is brought out. These new instructions are introduced in the machine and by the end of the chapter the machine becomes fairly sophisticated. This evolutionary development of the computer enables the students to appreciate "hardware-software trade off" in computer design. Chapter X presents the design of a small binary computer. The idea of microprogrammed control unit is also given in this chapter. Chapter XI discusses various types of data structures such as strings, lists, arrays, queues etc., which are stored and processed in general purpose digital computers. Chapter XII introduced peripheral devices used by computers. This is followed by a discussion of Input/Output architecture in general purpose computers. The last chapter is devoted to a discussion of microprocessors and microcomputers. Both hardware and software aspects of microcomputers is discussed in this chapter.

In 1971 the authors started writing notes from which this book evolved. The cyclostyled version of the book has been thoroughly class room tested and has undergone four revisions. It has been used for the first course in digital computer design taught to post graduate students in computer science at Indian Institute of Technology, Kanpur, who come with no previous background in computing. It has also been used in an optional course on digital computer design given to final year undergraduate students in Electrical Engineering. The notes have been used several times for short intensive courses on computer logic design taught to practicing engineers.

The first edition of this book was published in 1976. The edition was warmly received by students and was sold out in the first year. The authors felt the need to extensively revise the book to fill gaps in the previous edition and include some recent developments. Accordingly in this edition all chapters were thoroughly reviewed. Chapter VI on microprogramming language and Chapter X on logic design of a small computer have been rewritten. Chapter XII of the first edition has been replaced by a chapter on Input/Output devices and architecture. A new chapter, Chapter XIII, has been written on microprocessors and microcomputers.

We would like to thank all our students and colleagues for the many constructive comments given by them over the years. Our thanks are due to our students, Mr. V. Srinivasan and Mr. Ramesh Chandra, for proof reading the last two versions of the book and to Mr. V.M. Malhotra who thoroughly reviewed Chapters VI and X. We would like to thank Mr. H.K. Nathani for his excellent typing and Mr. A.K. Bhargava for advice on line drawings.

We would like to thank Dr. A. Bhattacharyya, Director, Indian Institute of Technology, Kanpur for providing the facilities to write this book. Thanks are due to the Educational Development Centre at IIT Kanpur which provided funds to prepare the manuscript of the first edition of this book.

Finally, the first author would like to thank his wife, Dharma, for drawing all the figures in the book, for proof reading and indexing the book and her dedication support in all his endeavours.

IIT-Kanpur - *V. Rajaraman*
May 1978 - *T. Radhakrishnan*

CONTENTS

CHAPTER I

INTRODUCTION

We may define a digital computer as a machine which accepts a stream of symbols, manipulates them according to precise rules, and produces a stream of symbols at its output. At the simplest level a digital processor may accept a single symbol at its input, perform an operation on it, and produce another symbol at its output. For example, a processor to find the square root of a one digit number would fall in this category. At a more complex level a large number of symbols may be processed using extensive rules. A digital system to automatically print a book would fall in the second category. Such a system should accept a string of symbols, namely, the typewritten material. Given the number of letters which could be accomodated on a line and the rules for breaking a word it should determine the space to be left between words on a line so that all lines are aligned on both the left and right hand sides of a page. The processor should also arrange lines into paragraphs and pages. Decisions to leave space for figures should be made. A multitude of such decisions are to be taken before a well laid out book is obtained. Such complex processing would require extensive special facilities such as a large amount of storage, electronic circuits to count and manipulate characters, and a printer which has a complete assortment of various sizes and styles of letters. Regardless of the complexity of processing, there are some basic features which are common to all digital processing of information which enables us to treat the subject in a unified manner. These are:

(1) All strings of input symbols to a digital system are transformed to an equivalent string which has only two distinct symbols. These symbols are 0 and 1 and the transformed string is known as a string of binary digits or bits.

(2) Rules for manipulating the symbols are to be precisely specified as a sequence of instructions. The instructions are also coded and stored as a string of binary digits.

(3) Manipulation or processing of binary symbols is realized by electronic circuits.

> One may design a variety of processing elements (i.e. electronic circuits) which accept binary symbols as inputs and transform them to a preassigned binary output symbol or symbols. In theory any manipulation or storage of binary symbols could be performed by a single type of "universal processing element". In other words a processor may require a large number of processing elements of the same kind. In actual digital processing systems, due to economic reasons, one may not use only "universal elements" but one would definitely minimise the variety of processing elements. The fact that it is possible to design complex digital systems with a small variety of processing elements is an important concept, which should be emphasised.

The logic design of digital computers and systems consists of implementing the three basic steps enumerated above keeping in view the engineering constraints such as the availability of processing elements, their cost, reliability, maintainability and ease of fabrication.

At this stage, we should distinguish between the design of a general purpose digital computer and that of a specialised digital subsystem. Even though the three basic steps in design are common to both, the constraints which are peculiar to each of these leads to a difference in the philosophy of design.

The general purpose machine is designed to perform a variety of tasks. Each task would require the execution of a different sequence of processing rules. The processing rules to be followed may vary widely. At the outset one would not be able to predict all the tasks one may like to do with a machine. A flexible design is thus required. This flexibility is achieved by carefully selecting a small set of *elementary processing rules* or operations and designing electronic circuits to implement these rules. These electronic circuits are together called *hardware*. One may realise a complex operation by using various sequences of elementary operations. For example, one may realise a multiplication operation by repeated use of addition operation. A sequence of elementary processing rules is called a *program* and may be

thought of as a *macro* operation. A set of macros could be used to perform more complex tasks. We can thus build up a hierarchy of programs, all stored in the computer's memory, which can be invoked by any user to perform a very complex task. A user need not work only with the elementary processing rules available as hardware functions. He can use the hierarchy of programs which constitute the *software* of a computer and which is an integral part of a general purpose digital computer.

It should be observed that it is possible to perform macro operations entirely by specially designed electronic circuits rather than by using macros. Thus software can be replaced by hardware and vice versa. What basic tasks are to be performed by hardware and what is to be done by combined software and hardware is an engineering design decision which depends on costs and other constraints prevailing at a given time. One of the purposes of this book is to bring out the hardware - software trade off which is important in the design of general purpose computers.

Specialised digital subsystems do not normally require a detailed study of software - hardware trade off. They are usually designed to perform a specific processing function using available processing elements. The first part of the book considers this design problem. The second part of the book is concerned with the more difficult problem of designing general purpose digital computers.

CHAPTER II

BINARY NUMBERS, CODES AND ARITHMETIC

In designing digital systems it is necessary to count and perform arithmetic operations such as addition, subtraction, multiplication and division. In this chapter, we discuss how these operations could be performed reliably and economically.

2.1 Numbering Systems

The most widely used number system is the positional system. In this system the position of a digit indicates the significance to be attached to that digit. For example, the number 8972.443 is taken to mean

$$\underset{\substack{1000\text{th}\\\text{position}}}{8 \times 10^3} + \underset{\substack{100\text{th}\\\text{position}}}{9 \times 10^2} + \underset{\substack{\text{Tenth}\\\text{position}}}{7 \times 10^1} + \underset{\substack{\text{Unit}\\\text{position}}}{2 \times 10^0} + \underset{\substack{1/10\text{th}\\\text{position}}}{4 \times 10^{-1}}$$

$$+ \underset{\substack{1/100\text{th}\\\text{position}}}{4 \times 10^{-2}} + \underset{\substack{1/1000\text{th}\\\text{position}}}{3 \times 10^{-3}}$$

In this notation 8003 cannot be written as 83 as the zeros in the middle give a different weight to 8.

An example of a non-positional number system is the Roman numeral system. This number system is difficult to use due to the absence of a symbol for zero.

Positional number systems have a radix or a base. In the decimal system the radix is 10. A number system with radix r will have r symbols. A number in a radix r system would be written in general as:

$$a_n\ a_{n-1}\ a_{n-2} \cdots a_0 \cdot a_{-1}\ a_{-2} \cdots a_{-m}$$

and would be interpreted to mean:

$$a_n r^n + a_{n-1} r^{n-1} + \cdots a_o r^o + a_{-1} r^{-1} + a_{-2} r^{-2} + \cdots$$

$$a_{-m} r^{-m}$$

The symbols a_n, a_{n-1},a_{-m} used in the above representation should be one of the r symbols allowed in the system. In the above representation a_n is called the *most significant digit* of the number and a_{-m} (the last digit) is called the *least significant digit.*

In digital instruments and computers, the number system used has a radix 2 and is called the binary system. In this system only two symbols, namely 0 and 1 are used. The symbol is called a *bit* , a shortened form for binary digit. Computers use the binary system as many physical elements used for storing and operating on numbers in digital systems are by their nature binary or two state devices. For example, a hole in the card is either punched or not punched, a switch is either "ON" or "OFF". In each one of the above cases no other state exists. Many other elements like transistor circuits are not intrinsically binary but operate with maximum reliability when used in the binary mode.

A number in the binary system will be written as a sequence of 1's and 0's. Thus, for example, 1011.101 is a binary number and would be taken to mean:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

The equivalent number in decimal is thus:

$$8 + 0 + 2 + 1 + 1/2 + 0 + 1/8 = 11.625$$

The following table gives the decimal numbers from zero to seventeen and their binary equivalents.

Decimal	Binary	Decimal	Binary	Decimal	Binary
0	0	6	110	12	1100
1	1	7	111	13	1101
2	10	8	1000	14	1110
3	11	9	1001	15	1111
4	100	10	1010	16	10000
5	101	11	1011	17	10001

Table 2.1 Binary Equivalents of Decimal Numbers

It is seen that the length of binary numbers can become quite long and cumbersome to use. Octal system (base 8) is thus often used to convert binary to a form

requiring lesser number of digits. The octal system uses the eight symbols 0, 1, 2, ... 7. As its radix 8 is a power of 2, namely 2^3, it is fairly simple to convert binary to octal and vice versa. (One must contrast this with conversion of binary to decimal)

Example 2.1

Binary number:	001	100	111	001
Octal equivalent:	1	4	7	1

Decimal equivalent: $1x8^3 + 4x8^2 + 7x8^1 + 1x8^0$

$= 512 + 256 + 56 + 1$

$= 825$

As illustrated in Example 2.1, one may convert a binary number to octal by grouping together successive three bits of the binary number starting with its least significant bit. These three bit groups are then replaced by their octal equivalents. This works because all the digits in the octal system, namely 0, 1, ..., 7, may be represented by three bit groups. Table 2.2 illustrates this:

Octal	Binary	Octal	Binary	Octal	Binary
0	000	3	011	6	110
1	001	4	100	7	111
2	010	5	101		

Table 2.2 Octal Numbers and Their Binary Equivalents.

Because of the simplicity of binary to octal conversion it is often faster, when converting from binary to decimal, to first convert from binary to octal and then convert the octal to decimal.

Another base which is often used in digital systems is known as *hexadecimal*. In hexadecimal base there are sixteen symbols. Thus each hexadecimal symbol may be represented by a 4 bit equivalent. Hexadecimal representation of 4 bit binary numbers is given in Table 2.3.

Binary	Hexadecimal	Binary	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Table 2.3 Hexadecimal Numbers and their Binary Equivalents.

Example 2.2

Convert the following binary number to hexadecimal

Binary number:	10	1010	1011	0111
Hexadecimal:	2	A	B	7

2.2 Decimal to Binary Conversion

In addition to knowing how to convert binary numbers to decimal it is also necessary to know the technique of converting a decimal number to a binary number. The method is based on the fact that a decimal number may be represented by:

$$d = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 2^1 + a_0 2^0 \qquad (2.1)$$

If we divide d by 2, we obtain:

$$\text{Quotient } q = d/2 = a_n 2^{n-1} + a_{n-1} 2^{n-2} + \ldots + a_1 2^0 \qquad (2.2)$$

and Remainder $r = a_0$

Observe that a_0 is the least significant bit of the binary equivalent of d. Dividing the quotient by 2 we obtain:

$$q/2 = d/(2x2) = a_n 2^{n-2} + a_{n-1} 2^{n-3} \ldots + a_2 2^0 \qquad (2.3)$$

and Remainder equals a_1.