# PRACTICAL DIGITAL DESIGN USING ICs

*SECOND EDITION*

**Joseph D. Greenfield**

# PRACTICAL DIGITAL DESIGN USING ICs

*SECOND EDITION*

**Joseph D. Greenfield**
ROCHESTER INSTITUTE OF TECHNOLOGY

**JOHN WILEY & SONS**
*NEW YORK CHICHESTER BRISBANE TORONTO SINGAPORE*

**MANUSCRIPT EDITOR: Dex Ott**
**PRODUCTION SUPERVISOR: Deborah Pokos**
**COVER DESIGN: Jules Perlmutter**



***Library of Congress Cataloging in Publication Data:***

Greenfield, Joseph D., 1930–
Practical digital design using ICs

(Electronic technology series, ISSN 0422-910X)
Includes bibliographical references and index
1. Digital integrated circuits 2. Digital electronics
3. Logic circuits I. Title II. Title Practical
digital design using I C s III. Series
TK7874.G72 1983 621 3815'3 82-10931
ISBN 0-471-05791-6

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

# PREFACE

The purpose of the second edition of *Practical Digital Design Using ICs* is to update the material in the first edition. The general thrust of the first edition—explaining the **7400** series of ICs—has been retained because it was well received, and because the **7400** is still the most popular and widely used IC series.

The major event in the digital field in the past five years has been the proliferation of microprocessors. I have accommodated this event in the second edition by providing a chapter on microprocessor interfacing and by emphasizing 3-state gates and line drivers and transceivers. Many new and more sophisticated ICs that work with microprocessors, such as UARTs and PIAs, have also been produced, and these are covered in as much detail as space and their importance allow.

Though the microprocessor seems to be ubiquitous, the use of small and medium scale ICs is even more widespread. Microprocessors will not soon replace ICs. Instead, they will work in conjunction with ICs to provide more sophisticated and powerful systems. As I point out in Chapter 1, the Apple computer uses about 50 ICs, most of which are discussed in this book, in addition to its **6502** microprocessor. Thus the study of ICs remains a necessary part of the curriculum for digital engineers and technologists.

Chapter 1 is an introduction to binary arithmetic. 2s complement and hexadecimal arithmetic are introduced in Chapter 14, where they can be presented in conjunction with the arithmetic circuits that actually use them.

An introduction to the basic IC gates has been moved into Chapter 2 to facilitate laboratory work with the Boolean algebra presented in the chapter. I have added a section on multiple outputs to Chapter 3 on reduction and Karnaugh maps. In Chapter 5 the material on 3-state gates has been expanded because of their widespread use.

Chapters 6, 7, 8, and 9 on flip-flops, one-shots, counters, and shift registers all have been modernized and some new material is introduced. Greater use of oscilloscope traces has been made where they can best illustrate a point.

In Chapter 10 on IC construction and testing, I have added a section on logic analyzers. The chapter on multiplexers has been moved to Chapter 11, so it can precede the chapters on multiplexing 7-segment displays and arithmetic circuits.

Chapter 15 has been updated to include more commonly used memories. The sections on dynamic RAMs and EPROMs have been expanded.

Chapter 16 is an entirely new chapter explaining how a digital computer works and how it can be built using the ICs introduced in the previous chapters. A very small computer, built by RIT students, is also introduced and explained. Making students build their own computer is, I feel, the best way to be sure they understand how computers work.

Chapter 17 is on minicomputer interfacing, and Chapter 19, which is new, examines microprocessor interfacing. In Chapter 17, the material on A/D converters has been expanded.

Chapter 18 discusses RS-232 and 20-mA current loop conversion. A detailed discussion of a commonly used UART has been included.

Chapter 20 is on the CRT Display Generator. It has been updated to conform to modern practice. I have added appendixes on the ASCII code and answers to selected problems. The latter has been requested by many people.

I would like to thank all the people who helped me with the second edition. I must mention my colleagues in the Department of Electronic Technology at Rochester Institute of Technology, and my students there, whose input kept my courses up to date. I must also thank the people at John Wiley, who kept pushing and prodding to get the manuscript done, and the pre-publication reviewers, who were very kind and enthusiastic, and provided many excellent suggestions for improvement.

I especially thank my wife Gladys. Without her love, encouragement, cooperation, and typing, this book would never have been written.

JOSEPH D. GREENFIELD
*Rochester, New York*

# CONTENTS

CHAPTER

# 1

# BINARY NUMBERS

## 1-1 INTRODUCTION TO DIGITAL ELECTRONICS

This book introduces the reader to digital electronics and enables him or her to understand, design, and construct digital circuits. The book is primarily concerned with the functions and uses of digital integrated circuits (ICs) and emphasizes the 7400 TTL (transistor transistor logic) series. These ICs have experienced a phenomenal growth in the past several years and millions of them have been manufactured. They can be purchased from manufacturers, electronics distributors, electronics stores such as Radio Shack, many computer stores, and by mail or phone from discount houses.[1] Their low price and wide availability have caused them to be designed into almost all digital circuits produced in the last 10 years.

Microprocessors ($\mu$Ps) have also enjoyed phenomenal growth recently. Some engineers believed that $\mu$Ps would replace ICs, but this has not happened. Rather, $\mu$Ps work in conjunction with digital ICs to provide a high level of intelligence and sophistication to digital hardware. The Apple computer, for example, uses a 6502 $\mu$P as its main computing element. It also uses about 50 TTL ICs (mostly types described in this book), plus ROMs and RAMs. Thus even with a $\mu$P in the system, many small and medium scale ICs are required to support it.

[1]These discount houses often advertise in magazines such as *Popular Electronics* or *Byte*.

Both standard and special purpose ICs are also needed to enable a $\mu$P to interface or communicate with other devices such as disks and printers. Some of these special purpose ICs and interaction between $\mu$Ps and ICs is covered in the later chapters of this book.

Both ICs and $\mu$Ps are the results of advances in *digital electronics*. Modern digital electronics is the product of a marriage between switching theory (a discipline in its infancy in the 1950s) and integrated circuits (a family of devices that was not developed until the 1960s). However, digital circuits have proven to be the best way to construct many useful devices, including the digital computer; consequently, its growth has been rapid, and a significant percentage of the jobs currently available in electrical engineering require digital training.

The output of most electronic circuits is an *analog quantity*, typically a voltage. An analog quantity is a quantity that may assume any numerical value within the range of possible outputs. Therefore, an electronic circuit is capable of producing many outputs in response to different inputs. For example, 5.12 volts (V) might be one output, 3.76 V another; any voltage within the precision of the voltmeter is a legitimate output. *Digital circuits* depart from analog circuits by providing only two values as an output: an output can either be a one (1) or a zero (0), *and nothing else*. Of course, digital engineers are still dealing with electronic circuits whose outputs are voltages. What they have done is to *define* a certain range of voltages as a logic 1 and another range of voltages as a logic 0. Typically, the 1 and 0 ranges are separated by a *forbidden* range of voltages. TTL integrated circuits define a 1 output as any voltage between 2 and 5 V and a 0 as any voltage between ground (zero) and 0.8 V. But what if the actual output of the circuit is in a forbidden or undefined range, 1 V, for instance? "Then," says the digital engineer succinctly, "the circuit is malfunctioning. Fix it." Methods of diagnosing and repairing malfunctions in digital circuits are discussed in Chapter 10.

Advances in digital technology have been more spectacular than advances in switching theory. The first computers were built in the early 1950s using large vacuum tubes that had low reliability and consumed a great deal of power. Tubes were replaced by transistors as soon as the latter became available and reliable. Engineers quickly found ways to build small, efficient, and inexpensive digital circuits using a single transistor, a couple of diodes, and a few resistors. Now, however, these discrete components (the resistors, transistors, and diodes) have all disappeared inside the IC. Recently ICs have become slightly larger and far more complex. The microprocessor itself is an example of a very large scale (VLSI) integrated circuit containing several thousand gates.

The first three chapters of this book develop the required switching theory. The remaining chapters describe the behavior of ICs, predominantly the 7400 TTL series, and ways in which to design useful circuits with them. These circuits can then be used to build computers, or *controllers* and *interfaces* that allow other devices to communicate with a computer. Examples of computer interfaces and controller design are presented in the later chapters of this book.

## 1-2 INSTRUCTIONAL OBJECTIVES

This first chapter acquaints the student with the binary number system and gives some facility in handling binary numbers. After reading this chapter, the student should be able to:

1. Convert binary numbers to decimal numbers.
2. Convert decimal numbers to binary numbers.
3. Find the sum and difference of two binary numbers.

## 1-3 SELF-EVALUATION QUESTIONS

As the student reads the chapter, he or she should be able to answer the following questions:

1. What is the difference between digital and analog circuits?
2. How are the outputs of a circuit defined to make the output digital?
3. What are the advantages of digital circuits?
4. What is the difference between a *bit* and a *decimal digit?* In what respects are they similar?
5. In a *flow chart*, what is the function of the rectangular and diamond-shaped boxes?
6. How are binary addition and subtraction different from decimal addition and subtraction? How are they similar?

## 1-4 USES OF BINARY BITS

The output of a digital circuit is a single binary digit (a 1 or a 0) commonly called a *bit*. There are two advantages gained by restricting the output of an electronic circuit to one of two possible values. First, it is rarely necessary to make fine distinctions. Whether an output is 3.67 or 3.68 V no longer matters; in both cases it is a logic 1. Since well-designed logic circuits produce voltages near the middle of the range defined for 1 or 0, there is no difficulty in distinguishing between them. In addition, a digital circuit is very tolerant of any drift in the output caused by component aging or changes. A change in a component would almost have to be catastrophic to cause the output voltage to drift from a 1 to a 0 or an undefined value. The second advantage is that it is far easier for electronic circuits to remember a 1 or a 0 than to remember an analog quantity like 3.67 V. Since all but the simplest digital circuits require the ability to remember the value of a voltage *after the conditions that caused that voltage have disappeared,* this is a very important consideration.

The output of a single digital circuit, a single bit, is enough to answer any question that has *only two* possible answers. For example, a typical job application might ask "What is your sex?" A 1 could arbitrarily be assigned to a male and a 0 to a female, so a single bit is enough to describe the answer to this question. A single bit is all the space a programmer needs to reserve in his computer for this answer.

However, another question on the job application might be "What is the color of your hair?" If the possible answers are black, brown, blonde, and red, a single bit cannot possibly describe them all. Now *several* bits are needed to describe all possible answers. We could assign one bit to each answer (i.e., brown = 0001, black = 0010, blonde = 0100, red = 1000), but if there are many possible answers to the given question many bits are required. The coding scheme presented above is not optimum; it requires more bits than are really necessary to answer the question.

It is most economical to use *as few bits as possible* to express the answer to a question, or a number, or a choice. The crucial question is:

"*What is the minimum number of bits required to distinguish between n different things?*"

Whether these *n* things are objects, possible answers, or *n* numbers is immaterial. To answer this question we realize that each bit has two possible values. Therefore, *k* bits would have $2^k$ possible values. This fact is the basis for Theorem 1.

## Theorem 1

*The minimum number of bits required to express n different things is k, where k is the smallest number such that* $2^k \geq n$.

A few examples should make this clear.

---

**EXAMPLE 1-1**

What is the minimum number of bits required to answer the hair color question, and how could they be coded to give distinct answers?