Logical Design Using Integrated Circuits

WILLIAM D. BECHER

Logical Design Using Integrated Circuits

WILLIAM D. BECHER

Professor of Electrical Engineering
The University of Michigan—Dearborn

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

To my parents, who started it all, and my family, Helen, Eric, and Patricia, who make it worthwhile

Library of Congress Cataloging in Publication Data

Becher, William D 1929-
Logical design using integrated circuits.

Includes bibliographies and index.
1. Logic circuits. 2. Logic design. 3. Digital electronics. 4. Integrated circuits. I. Title.
TK7868.L6B4 621.3819'58'2 77-23254
ISBN 0-8104-5858-

Printed in the United States of America

1 2 3 4 5 6 7 8 9 PRINTING

77 78 79 80 81 82 83 84 85 YEAR

Preface

Digital logic and switching theory have found many uses in modern technology. Telephone switching was one of the early applications of these principles, and they contributed significantly to its rapid growth and utility. Other early contributors to this field were radar systems, digital instrumentation systems, and data communication systems which lead to the formalization of pulse circuit design and application. Closely following the development of these systems was the development of data handling and data processing systems.

Although the development of these systems was a major impetus to the development of digital technology, the explosive growth would not have occurred without the rapid improvement of digital components. In earlier periods the electromagnetic relay served as the basic digital element. Digital technology made a significant leap forward with the application of the vacuum tube to digital circuits. The transistor provided even greater improvements, but it was the emergence of integrated circuit components which provided the spectacular growth of recent times. Not only has integrated circuit technology provided improvements in the performance characteristics of digital systems but it has also provided economical savings which promise even more dramatic growth in the future.

No one can expect to escape the influence of digital logic systems today. They have become an important component of many systems—from the automated traffic lights to process control and automation.

It is the purpose of this book to provide an introduction to logical analysis and design using integrated circuit components. Although it is sufficiently theoretical to provide insight and understanding of modern formal design methods, emphasis has been placed on the practical aspects of logical design. Each topic included has been selected to bring the reader, as rapidly as possible, to the level of competence required to understand and design complex digital systems successfully. Readers who complete this book will also have the technical background to pursue more advanced topics in switching theory and logical design.

The book is intended for anyone with an interest in logical design. It assumes no technical background beyond a familiarity with ordinary algebra, which will provide the mathematical maturity required to follow the development of the Boolean algebraic techniques. All analytical tools required are developed as needed. Although some electrical circuit details are included for completeness of material, the reader who finds electrical fundamentals difficult or boring will not be seriously handicapped. The few basic ideas required can be easily mastered and the use of integrated circuits effectively made.

The material in this book has been used in a wide variety of courses and presented to students with diverse backgrounds. It originated as an elective undergraduate/graduate electrical engineering course at the University of Michigan-

Dearborn. Most of the material is now an undergraduate requirement in electrical engineering. Throughout the long history of the course many non-electrical engineering students, including numerous students from the College of Arts, Sciences, and Letters, have successfully completed this material. Student levels have ranged from freshman to first-year graduate students.

A number of the students who graduated from this course have worked with the author in the laboratory and have found the topics covered sufficiently broad to design many useful digital systems; others have reported successful application of the material in industrial organizations. This success in practical application led to the offering of the material in a one-week continuing engineering education course at the University of Michigan in Ann Arbor. The course has been offered each year since 1971 to practicing scientists, mathematicians, engineers, and technicians. It was largely from the notes developed for this course and the encouragement of the participants that this book emerged.

The organization of the book is designed to develop the skill and understanding of the reader systematically. The book begins with a study of the characteristics of number systems and manipulative techniques required for logical design. Since most digital elements are binary in nature and most digital systems are coded in number systems other than binary, it is necessary to understand the relations between number systems. Chapter 2 introduces the reader to coding systems and their conversion to other coding systems.

Formulation and manipulation of switching systems and logic circuits is facilitated with a knowledge of Boolean algebra. Chapter 3 develops the rules of Boolean algebra—first from the more intuitive set theory and then more formally by an axiomatic approach. Because of the ease of understanding its operation, relay logic is introduced in Chapter 4 to provide the first exposure of the application of Boolean algebra to design and analysis of switching circuits. Chapter 5 extends the application to the integrated circuit element, electrically more complicated, but more useful and relevant to modern digital and computer systems. A description is given of basic logic elements and the effects of the element parameters, and a discussion is presented of the types of logic families commercially available. It is the intent of the chapter to provide a survey of practical and useful integrated circuit logic elements.

Although formal Boolean algebra techniques and manipulative skills are basic to a firm understanding of logical design, tools are available which greatly enhance these more formal analytic methods. These are the Karnaugh map and Quine-McCluskey minimization techniques described in Chapters 6 and 7, respectively. Most algebraic expressions arising from algebraic design techniques are in a form which are not entirely compatible with integrated circuit logic elements. Chapter 8 provides several methods of altering the equations to permit the direct application of these elements.

Chapter 9 summarizes the techniques studied in the preceding chapters. Its purpose is twofold—it provides several design examples and describes a number of MSI integrated circuit elements commercially available, including code converters, code generators, and arithmetic units.

A new logical element, the flip-flop, is introduced in Chapter 10. Many types of commercial flip-flops are illustrated and analyzed. Chapter 11 discusses the formal procedures for designing logic circuits which incorporate flip-flop elements. These circuits are called clocked and pulsed sequential circuits and

have special characteristics which make them particularly easy to design. Chapter 12 extends the sequential design techniques to the level-mode sequential machines which require more elaborate methods.

Additional examples of sequential designs are given in Chapter 13 not only to illustrate further the procedures but also to introduce some of the common MSI integrated circuit sequential elements presently available. Also included is a description of circuits used for digital system timing and control. Chapter 14 describes several design examples of digital systems to illustrate the ease with which complex systems can be designed by using the techniques and components introduced in the book.

Problems are provided at the end of each chapter. These were chosen to illustrate the main topics of the book and, in some cases, to extend the material. Answers are given to selected problems to verify the solutions. The reader who solves all the problems will find that he has mastered the text material. Many references have been provided for those who wish to study certain topics more thoroughly.

The book can easily be covered in a three semester hour course. Other arrangements are possible. For example, the material in Chapters 1 through 10 and part of Chapter 13 has been successfully used in a two semester hour course. The entire book has also been presented in a one-week intensive course. Others have reported that the material is effective as a self-study text.

As is the case with all books, there are many who contributed to its preparation. Particular thanks, however, must be given to a select few. First of all, thanks is due to the students who have suffered through the many editions of the manuscript and who made many valuable criticisms and suggestions. Thanks also go to Dean J. Robert Cairns for providing the opportunity to teach the digital logic course repeatedly, and to my Department of Electrical Engineering colleagues who helped select the material most appropriate for the electrical engineering curriculum.

Special thanks also go to Mr. Raymond E. Carroll, who supported the development of the one-week continuing engineering education course and to the instructors who worked with me on the course. These included Professors David J. Anderson, Keki B. Irani, Murray H. Miller, Norman R. Scott, and John F. Riordan, and Messrs. John O. Brown, Mitchell A. Goodkind, Raymond R. Jonassen, Thomas A. King, Charles W. Kleekamp, and George W. McClure. Ms. Jane A. Strom directed the preparation of the course notes.

Thanks must be given to my secretary, Ms. Mary L. Wilkie, who helped arrange my time and activities so that it was possible to complete the manuscript. My son and daughter, Eric and Patricia, also deserve special mention for their patience and understanding during the lengthy period of manuscript preparation. And a special thanks to Cookie, who kept me company during the long sessions in my Study.

And finally, loving appreciation to my wife, Helen, who not only encouraged the writing of this book, but also was an important contributor to the effort by typing, retyping, and proofreading the entire manuscript. Without her the book would not have been written.

W.D.B.

Contents

1
Number Systems

Introduction

The earliest systematic numbering system is believed to be the one developed by the Egyptians approximately 5,000 years ago. The system, known as hieroglyphics, has been well documented from numerous archeological records and is basically a tally or unitary system. One can speculate that it arose from the need to tally the quantity of products traded between farmers. Perhaps during a trade of cattle for wheat, it occurred to the traders simply to tally the number of cows and bins of wheat traded for later comparison rather than to pair a cow with eacn bin. Marks scratched in the sand may have been used for record keeping during the transaction.

As the number of units involved in the trade grew, interpretation of repetitive marks representing large quantities undoubtedly became difficult, and it is quite natural that additional symbols were introduced to replace various collections of marks. Specifically the hieroglyphic system used the vertical stroke | to represent the basic tally unit, the croquet wicket ∩ to represent ten vertical strokes, the snare 𓍢 to represent ten croquet wickets, etc. Thus, for example, the quantity 354 was represented as 𓍢𓍢𓍢∩∩∩∩∩|||. Note that each new symbol denotes a tally ten times larger than the symbol it replaces—a forerunner of the present-day decimal (base-10) system. Perhaps this use of ten resulted from counting with the ten fingers—or toes.

Approximately 1,000 years later (records are available from around 1850 B.C.) the hieroglyphic system had evolved into the more enumerative hieratic system. In this system groups of hieroglyphic symbols were replaced by simpler marks in the same manner as our modern system, which uses the symbols 1, 2, 3, 4, etc. The hieratic representation of the quantity 354, for example, was 𓏺𓎆𓍢 where the first symbol denotes 4, the second 50, and the third 300.

By the year 1650 B.C., the Babylonians had developed a system similar to that of the Egyptians except that it was a sexagesimal system (base-60) with a decimal (base-10) substratum. The reason for the use of base-60 instead of base-10 is obscure—they surely didn't have 60 fingers—but possibly it was influenced by the large number of exact divisors found in 60. A major numerical contribution of the Babylonian system was the use of the principle of position, i.e., the concept that the relative position of the symbol within the number in-

fluences its value. For example, since 𒁹 was the symbol for unity and 𒌋 the symbol for 10, 𒁹𒌋 represented the quantity $1(60)^1 + 10(60)^0 = 70$ whereas 𒌋𒁹 represented $10(60)^1 + 1(60)^0 = 601$. Unfortunately the Babylonians had no symbol for missing positions, a major deterrent to numerical maturity.

The most well-known ancient method of numeration is the Roman numeral system developed around 700 B.C. This system is basically a base-10 system with the symbols subdivided in groups of five, i.e., I = I, 5 = V, 10 = X, 50 = L, etc. Positional ideas were used in this system, too (e.g., 4 = IV and 6 = VI). Anyone who has tried to perform arithmetic operations or represent large numbers with the Roman system must marvel that Roman numerals are still in use. This is surely a result of the strong influence of the Roman Empire on modern civilization and not because of the utility or practicality of the Roman system.

The basis for our modern number system was not developed until around A.D. 700 when the Hindus used decimal, ciphered, and positional notation. This system, which introduced the extremely important concept of the cipher 0, permitted large quantities to be represented with only ten symbols. It wasn't until the late tenth century that modern numerals (of Hindu-Arabic origin) were introduced into Western Europe. As surprising as it might seem, the system was not extended to fractions until A.D. 1585, and the decimal point was not introduced until the early seventeenth century! One has only to speculate about the major numerical concepts awaiting discovery.

The Decimal System

The modern decimal number system is based on the use of ten numerals or symbols (0, 1, 2, . . . , 9) which obtain their numerical value from their relative position in the numerical representation. In this system, the right-most position (or digit) represents units, the next adjacent digit represents tens, the next hundreds, etc. In other words, each digit is multiplied by a power of ten which is ten greater than the multiplier of the digit on its immediate right. Ten is called the radix, or base, of the system and the power corresponds to the order. The four-digit number (or quantity) 1534 will serve as an example. It may be trivially decomposed into the value of each digit as

$$
\begin{array}{cccc}
1 & 5 & 3 & 4 \\
| & | & | & \llcorner\!\!\rightarrow 4 \times 1 = 4 \times 10^0 = 4 \\
| & | & \llcorner\!\!\rightarrow & 3 \times 10 = 3 \times 10^1 = 30 \\
| & \llcorner\!\!\rightarrow & & 5 \times 100 = 5 \times 10^2 = 500 \\
\llcorner\!\!\rightarrow & & & 1 \times 1000 = 1 \times 10^3 = \underline{1000} \\
 & & & \qquad\qquad\qquad\qquad\quad 1534
\end{array}
$$

In this example, 4 is referred to as the zero- or low-order digit, 3 as the first-order digit, etc. The low- and high-order digits are also sometimes called the least and most significant digits, respectively.

The Octal System

The counting procedure in the decimal system uses nine numerals to represent the numbers from 1 to 9. Once the count of 9 is reached there are no additional nonzero numerals available and a 1 is placed in the first-order (tens) position

and a 0 (zero has no numerical value except to influence the positional location of the other digits) is placed in the low-order (units) position. Counting then proceeds in the same manner with the numerals 1 to 9 placed in the units position. When the numerals run out again, the numeral in the first-order position is incremented by 1 and the process is repeated, e.g., 1, 2, . . . , 9, 10, 11, 12, . . . , 19, 20, 21, 22, ... When all nine numerals in the first order have been used, the process is repeated, incrementing the second-order numerals as required. This, of course, is a procedure well known to kindergarteners.

Had our predecessors had only eight fingers, our numbering system would undoubtedly have evolved differently. Instead of a radix-10 (decimal) system, the radix-8 (octal) system might have been used. In this system there would have been eight numerals including zero. Thus, using the first eight numerals of the decimal system as the eight numerals of the octal system, the counting sequence becomes

$$1, 2, \ldots, 7, 10, 11, 12, \ldots, 17, 20, 21, 22, \ldots$$

Therefore, to represent the number 8 (decimal), it is necessary to use both the zero and first-order digits of the octal system. It follows that[1]

$$(10)_8 = (8)_{10}$$
$$(20)_8 = (16)_{10}$$

By following the method of decomposition used to evaluate decimal numbers, the decimal equivalent of the octal number $(1534)_8$ is obtained as

$$\begin{aligned}(1\ 5\ 3\ 4)_8 \\ 4 \times 8^0 = 4 \times 1 &= 4 \\ 3 \times 8^1 = 3 \times 8 &= 24 \\ 5 \times 8^2 = 5 \times 64 &= 320 \\ 1 \times 8^3 = 1 \times 512 &= \underline{512} \\ &(860)_{10}\end{aligned}$$

where powers of 8, instead of 10, are used to weight the digit positions.

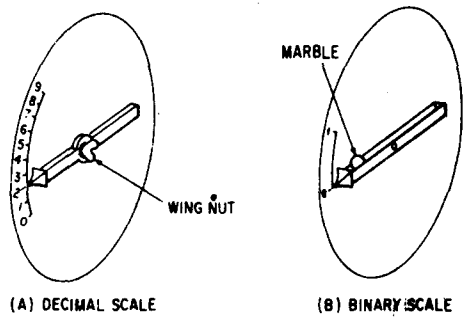
It is interesting to note that the number of digits required to represent a number in the decimal system is less than the number required for the octal system. What would be the advantage of the sexagesimal system? Why do you suppose the Babylonians used a decimal substratum?

The Binary System

Suppose it is necessary to design a mechanically adjustable numerical indicator which will be subjected to severe vibration. Two approaches might be taken. A ten-valued indicator corresponding to the popular decimal system might be used for each digit. A typical digit is depicted in Fig. 1–1A. Once set, the

[1] Whenever mixed radix systems are discussed it will be necessary to distinguish between systems. Thus, if it is not clear from the accompanying text which radix system is implied, subscripts denoting the radix will be appended to each number. Perhaps a better method would be to use entirely new symbols for the numerals of each system, but unfortunately, only a limited quantity of symbols is available to the typesetter.

Fig. 1–1. Mechanical digit indicators.



pointer will remain stationary only if the pivot arm is forcibly constrained by some mechanical device such as the wing nut indicated. On the other hand, the two-valued (binary) system (Fig. 1–1B), if properly designed, will remain in either of the two positions with no additional mechanical restrictions on the pivot arm. The system thus possesses greater positional stability than the decimal system. Similar stability advantages can be obtained in two-valued electrical and electronic circuits; e.g., an electrical switch is usually designed to be either open or closed and indicator lamps are either dark or lit. For these reasons it should be useful to develop a counting system based on a radix of two, i.e., a radix-2, or binary, system.

By following the same counting procedure used in the decimal and octal systems, the nonzero binary numerals are first placed in the low-order binary position (*binary digit*, or *bit*) until all have been used. Since there is only one nonzero binary numeral, the next higher order binary position must be used immediately after the first count, etc. Thus, in binary, counting proceeds

1, 10, 11, 100, 101, 110, 111, . . .

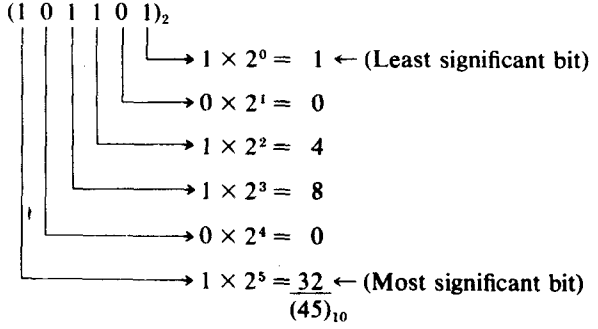
Table 1–1. Number Systems

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

Table 1–1 shows the relationship between the decimal, binary, octal, and hexadecimal[2] systems. As in the decimal system, leading zeros are optional in any system and have been suppressed in some cases. The reader will find it worthwhile to memorize the numerical equivalences between the different systems in the first $(16)_{10}$ rows.

Binary-to-Decimal Conversion

Several methods are available for the conversion of binary numbers to decimal numbers. The first method parallels the method used for octal-to-decimal conversion previously described. It is presented without further discussion.



Thus $(101101)_2 = (45)_{10}$. Notice how much more compact the decimal system is (two digits) compared to the binary system (six digits).

Table 1–2 is provided to assist in the conversion to the decimal system from the binary, octal, and hexadecimal systems. To illustrate its use, observe that, from the fifth line of the table, $4096 = 2^{12} = 8^4 = 16^3$ where l, m, and n equal 12, 4, and 3, respectively.

The second method is known as the double-dabble method. It is based on the following rule:

> Starting with the most significant nonzero digit, add 1 (dabble), and proceed to the adjacent lower-order bit, multiplying by 2 (double) on the way. Continue dabbling and doubling for each successive bit in descending order, dabbling only if the bit is nonzero but always doubling regardless of the value of the bit. The process stops with the dabble (or nondabble) at the least significant bit.

Application of the double-dabble rule to the example above is illustrated in Fig. 1–2. In step a, the most significant nonzero bit (order five) is dabbled, i.e., the running sum N is set to 1. In step b, the result is doubled to produce $N = (2)_{10}$. No dabbling occurs in step c, since the fourth-order bit is zero and N remains equal to $(2)_{10}$. In step d, the result is again doubled to make $N = (4)_{10}$. The process repeats until, finally, $N = (45)_{10}$, which agrees with the decimal number obtained by the first method.

Reflection on the double-dabble procedure reveals that all nonzero bits are doubled according to the order (position) of the bit; i.e., the process can be written

---

[2] Hexadecimal (radix-16) systems will be considered in Chapter 2.

Table 1–2. Decimal Equivalents for Integer Powers of 2, 8, and 16

Decimal	Binary (2^l) l	Octal (8^m) m	Hexadecimal (16^n) n
65 536.	16		4
32 768.	15	5	
16 384.	14		
8 192.	13		
4 096.	12	4	3
2 048.	11		
1 024.	10		
512.	9	3	
256.	8		2
128.	7		
64.	6	2	
32.	5		
16.	4		1
8.	3	1	
4.	2		
2.	1		
1.	0	0	0
0.5	−1		
0.25	−2		
0.125	−3	−1	
0.062 5	−4		−1
0.031 25	−5		
0.015 625	−6	−2	
0.007 812 5	−7		
0.003 906 25	−8		−2
0.001 953 125	−9	−3	
0.000 976 562 5	−10		
0.000 488 281 25	−11		
0.000 244 140 625	−12	−4	−3
0.000 122 070 312 5	−13		
0.000 061 035 156 25	−14		
0.000 030 517 578 125	−15	−5	
0.000 015 258 789 062 5	−16		−4

$$\begin{array}{c} a\ b\ \ c\ d\ \ e\ f\ \ g\ h\ \ i\ j\ \ k \\ \downarrow\ \downarrow\ \ \downarrow\ \downarrow\ \ \downarrow\ \downarrow\ \ \downarrow\ \downarrow\ \ \downarrow\ \downarrow\ \ \downarrow \\ (((((1)\cdot 2+0)\cdot 2+1)\cdot 2+1)\cdot 2+0)\cdot 2+1=45 \end{array}$$

where the steps in Fig. 1–2 have been identified to clarify the correspondence with the double-dabble method. In this expression, each 1 corresponds to a dabble, each 0 to no-dabble, and each 2 to a double. Collecting all 2s, this expression becomes

$$1\times 2^5+0\times 2^4+1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0=45$$

which corresponds exactly to the first method.

A major advantage of the double-dabble method is the ease with which binary numbers can be mentally converted to their decimal equivalents. A few practice conversions should soon convince the reader of this.