Logic Design and Switching Theory

SABURO MUROGA

Professor of Computer Science
and Electrical Engineering
University of Illinois

A WILEY-INTERSCIENCE PUBLICATION

JOHN WILEY & SONS

New York / Chichester
Brisbane / Toronto

Library of Congress Cataloging in Publication Data

Muroga, Saburo.
Logic design and switching theory.

"A Wiley-Interscience publication."
Includes bibliographical references and index.
1. Logic circuits. 2. Logic design. 3. Switching theory. 4. Integrated circuits—Large scale integration. 5. Electronic circuit design—Data processing.
I. Title.

TK7868.L6M87 621.3815'37 78-12407
ISBN 0-471-04418-0

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Logic Design and Switching Theory

Preface

The enormous progress in integrated circuit technology in the last decade has brought about dramatic changes in logic design, and the pace of change continues to accelerate. Progress in large-scale integration (LSI) has forced many manufacturers having no previous connection with electronics—such as watch, toy, or sewing machine manufacturers—into LSI design and even in-house LSI production. Consequently, logic design is now important for a much wider circle of people, and this circle will continue to grow. Although logic design has been facilitated by off-the-shelf packages, the opportunities for custom LSI design are increasing.

Logic design has also evolved into a more complex discipline, and the logic designer must be familiar with its many aspects. As the number of different integrated circuit (IC) logic families—e.g., ECL, TTL, NMOS, CMOS, VMOS, and IIL—continues to grow, the designer must choose the appropriate family to attain different design objectives, and use different logic design procedures for different families. Compact and inexpensive semiconductor memories, particularly ROMs and PLAs, are now often used as important parts of logic networks. Hence the designer must carefully consider which parts of a network should be realized with gates or memories. Also, as IC costs decline, many complex job functions—such as the computation of trigonometric functions—are being designed in logic networks instead of in software, and the designer must find appropriate algorithms to realize such job functions. As a result of these and many other changes, logic designers must now be familiar with not only switching theory but also architecture, different IC logic families, memories, algorithms for hardware implementation of complex job functions, programming for firmware, diagnosis methods, electronics, and possibly many other related concerns.

The coverage of all these topics in this book is obviously impossible, and instead emphasis is on minimization procedures, a classic topic that retains its importance even in logic design with integrated circuits. Of course, the minimization of an entire system depends on other factors, including the choice of appropriate architectures, algorithms, and electronic circuit designs, but the minimization of logic networks (in particular, key networks

such as adders) is still crucial. Unlike conventional minimization with discrete gates, minimization in LSI design does not result directly in cost reduction. Instead, it reduces the size and power consumption of a chip, which in turn reduces the cost of the chip and increases its speed. As designers attempt to pack more and more elements into a single chip, even partial minimization will often dramatically reduce the chip size.

This book emphasizes algebraic minimization procedures for three reasons: (1) Algebraic procedures sometimes can reduce design time drastically. For example, the algebraic procedure for deriving a minimal sum in Section 4.6 is nearly 100 times faster than the conventional tabular method treated in textbooks.

(2) Development of programs for computer-aided design (CAD) has been extensively carried on in industry in regard to every aspect of design and manufacture (including logic design), in order to reduce time, cost, and error. This trend will be even stronger in the future. Algebraic concepts are useful in implementing many CAD programs (also data base management systems, as mentioned in Section 3.7). Like many computer programs in other areas (such as business applications), many CAD programs need continuous maintenance or changes, as technology or usage changes. The computational efficiences of CAD programs are highly dependent on the details of the procedures. Also, many variations of the procedures may be needed for different CAD programs. Thus the computational efficiencies and some variations are discussed in this book so that students can handle different problems efficiently.

(3) When a large number of transistors (soon a quarter million) are to be packed into a single LSI chip, we need to rely more on CAD programs implementing algebraic minimization procedures in order to reduce design time and error. In this case, appropriately specifying or modifying the details of the procedures will result in great differences in the size or speed of the designed networks. This is particularly true when heuristic procedures need to be applied in designing very large networks, giving up on absolute minimization.

In conclusion, it is believed that slightly advanced knowledge of algebraic procedures will greatly enhance students' skill in logic design and also in CAD programming.

This book is intended for a course that follows an introductory course on digital computers (discussion of number representations, elementary switching theory, standard networks, and control logic, typically based on such textbooks as *Computer Logic Design* by M. M. Mano, and *Design of Digital Computers* by H. Gschwind and E. J. McCluskey. To make this book self-contained, some elementary concepts (such as Karnaugh maps) are

included which may overlap with topics in the introductory course on digital computers and hence require only a review.

After the introduction in Chapter 1, this book discusses hardware implementation of logic operations. Analysis of MOS networks is discussed in detail, considering the increasing importance of the MOS in LSI.

Chapters 3 through 5 discuss switching algebra and algebraic procedures. To strengthen students' intuitive grasp, mathematical concepts are pictorially illustrated with Karnaugh maps. Excercises are indispensable for understanding. Students who may be overwhelmed initially by algebraic terminology will understand algebraic concepts as they advance through succeeding chapters. It is believed that Tison's methods in Procedures 3.4.2, 5.1.1, and 5.1.2 are presented here for the first time in a textbook. The author's Procedures 4.6.1, 5.2.1, and 5.2.2 are not available elsewhere in book or paper form. (Procedure 4.6.1, in particular, is highly efficient for hand computation, and useful when a minimal sum cannot be conveniently handled by the Karnaugh map method. Also, Procedure 5.1.1 or Procedure 5.1.2 can yield a minimal sum for a function of many variables very quickly if the given function is not too complex. Where an engineer in industry, as it has come to light, tried to find a minimal sum by working on 16 maps simultaneously, these procedures would have provided a powerful improvement.) By studying these chapters, students can learn techniques to speed up calculation in switching algebra.

Logic design of NAND (or NOR) gate networks is usually not discussed in detail in textbooks despite its vital importance in current IC technology. A major part of Chapter 6 is thus devoted to this subject. To treat this problem, the map-factoring method of G. A. Maley and J. Earle is discussed, with minor modification. Also, the author's extensions (not available elsewhere) are presented as Procedures 6.4.1 and 6.5.1. The rest of Chapter 6 is devoted to problems that are important in design practice; in particular, Section 6.7 discusses the response time of combinational networks.

Chapters 7 and 8 are devoted to sequential networks, the design of which is inherently complex and consequently difficult to teach. If we emphasize easy understanding, as has been done in the past, the complex situation is oversimplified, giving students an unrealistic picture. On the other hand, if we emphasize the realistic picture of sequential network design, this may be harder for students to understand. Facing this dilemma, the author concluded that it may be time to give the realistic picture, since logic design is becoming a more important profession and this book is for an advanced course. On the basis of this decision (this stance is maintained throughout the book—procedures would not be useful if calculation techniques were not detailed), asynchronous sequential networks, considerably more

complex than synchronous sequential networks, are first discussed in Chapter 7—because sequential networks in fundamental mode are basic in both asynchronous and synchronous sequential networks (see the next paragraph); because unless the malfunctioning of asynchronous networks due to hazards or races is understood, the advantages of synchronous networks with raceless flip-flops may not be appreciated; and because asynchronous sequential networks are useful for high-speed computers.

Another problem in the discussion of sequential networks is pulse mode, which is widely presented in literature. Since the concept of pulse mode is only vaguely defined or inappropriate in electronic implementation (see the footnote in Section 8.3), skew mode is introduced in conjunction with raceless flips-flops, with the view that the concept is simply a convenient means to analyze or synthesize sequential networks with raceless flip-flops, which can also be analyzed or synthesized by fundamental mode. Also, there are sequential networks that are not in skew mode but in fundamental mode.

No attempt is made to present elaborate mathematical results on sequential networks, which are abundant in literature, because space is limited and such results, though conceptually important and exceedingly interesting academically, are time-consuming in nature and hence usually difficult to apply in design practice.

Chapter 9 tries to bridge the developments in the preceding chapters with logic design practice, discussing computer design practice, various design motivations, design time, important standard networks, logic networks with a mixture of gates and memories, PLAs (programmable logic arrays), flow charts for logic design, and diagnosis methods. Some of these, such as PLAs, are important topics for CAD programs. Since logic design practice is becoming enormously complex, as mentioned in the beginning, this book does not attempt to discuss the entire picture of design practice or all aspects of computer system design. The discussion in Chapter 9 is limited to the aspects related to the preceding chapters. The design procedures in the earlier chapters are useful in LSI design—for example, PLAs and gate arrays, widely used as inexpensive LSI implementations. The author's procedure in Section 9.3(b) is not available elsewhere.

The remarks scattered throughout this book provide pertinent information for advanced readers, with the intention of making this book serve as a reference book. In college courses, it is appropriate that students skip these remarks.

This book is the outcome of lecture notes used for the last several years at the University of Illinois at Urbana. Many people have helped on numerous occasions, for which the author is grateful. In particular, Professors L. L. Dornhoff, the late F. E. Hohn, G. A. Metze, and D. E. Muller, and Dr. J. E. Forbes, used the classnotes in their teaching; and Messrs. R. B. Cutler,

W. T. Dumas, R. Fujimoto, S. Isoda, S. Murai, and T. Ogino, A. Sakurai, Professor D. S. Watanabe, Mrs. M. H. Young, and Dr. R. O. Winder, made valuable discussions and suggestions. Also, especially, Professor L. L. Dornhoff read through the entire manuscript repeatedly and made many corrections and improvements, and Mr. R. B. Cutler's discussions on the procedures in Sections 5.1 and 5.2 were invaluable. The Department of Computer Science at the University of Illinois provided generous support for the preparation of the classnotes, and grants by the National Science Foundation (GJ-503-1, GJ-40221, DCR73-03421 A01, and MSC77-09744) produced research results used throughout this book. Finally, thanks to Mrs. Z. Arbatsky and Mrs. R. Taylor for their excellent and patient typing of many revisions of the classnotes.

Urbana, Illinois
January 1979

SABURO MUROGA

How To Use This Book

On the Exercises

A, B, C, . . . denote exercises of the same nature, so only one of them should be assigned.

(R) means that an exercise serves to review what is discussed in text.

(E) means that an exercise is easy.

(M) means that an exercise is of medium difficulty. Among those marked with M, exercises easier than the average and those more difficult are designated as M− and M+, respectively, the difficulty increasing in the order of M−, M, and M+.

(D) means that an exercise is difficult.

(T) means that an exercise is time-consuming.

On the Sections Labeled with ▲

Sections or topics labeled with ▲ are optional, and skipping them will not impede the understanding of later material. Exercises corresponding to these sections or topics are labeled with ▲.

On the Remarks

Remarks present advanced information, and skipping them will not impede the understanding of later material. It is advised that in college teaching, the remarks be completely skipped.

Contents

CHAPTER 6

CHAPTER 7

CHAPTER 1

Introduction

Digital systems such as electronic digital computers or electronic telephone exchanges contain digital networks that perform logic operations on digital input signals. Each digital network consists of gates that perform basic logic operations. This book discusses the design of the logic operation aspect of digital systems (in contrast to electronic design or others) as well as a related theory—in other words, logic design of networks with gates and switching theory.

Chapter 1 discusses motivations, history, and some basic concepts of logic design and switching theory.

1.1 Motivations for Study of Logic Design and Switching Theory

Logic design is the design of digital networks with gates so that the networks perform specified logic operations on input signals, in contrast to electronic design of digital networks, which derives electronic circuits with the desired electronic performance, such as the desired speed, power consumption, or waveforms. In a broader sense, "logic design" includes the design of structural aspects of the entire computer systems, that is, the architectural design of computer systems. The distinction between these meanings of "logic design" is often not very clear. This book deals mainly with logic design in the first sense. The design theory of digital networks is called **switching theory**, since it is a theory not only for networks of gates but also for those of switching devices such as relays or electrical switches that are older digital networks. A special case of Boolean algebra that serves as a basic mathematical tool in switching theory is called **switching algebra**.*

The integrated circuit (IC) is a recently developed and still developing approach to implementing gates, connections, or entire networks on a tiny semiconductor chip in an integrated way. The development of IC technology

* Very often, "Boolean algebra" is used by authors as a synonym of "switching algebra" because the difference is subtle. Strictly speaking, however, switching algebra is a special case of Boolean algebra, as will be explained in Section 3.6.

has brought about many changes. The enormous reduction in the cost and size of digital networks due to IC technology is expanding the applications of digital systems from the traditional ones, such as digital computers and electronic telephone exchanges, into many new areas. Examples of such new applications are sewing machines (e.g., Singer's Athena 2000), cameras (e.g., Polaroid SX-70), digital watches, video games, nonvideo games (e.g., Parker Brother's Sector), automobiles, and home computers [Weisbecker 1974]; many others are yet to come. Thus logic design not only is indispensable for computers but also is becoming very important for all products that incorporate digital systems (no matter how small or large these systems may be). As IC technology is still making great strides, it is very difficult at the current stage to foresee what great technological or social impacts it will have. In the following paragraphs we will review the changes in logic design that IC technology is making.

If a designer is not attempting to attain in his or her networks the best electronic performance (e.g., speed) or economy possible with current IC technology, logic design is now much simpler than before, since a large number of ready-made IC packages (i.e., **off-the-shelf packages**), which contain **standard networks**, that is, frequently used networks such as adders or multipliers, are commercially available. What the designer must do is simply to choose appropriate off-the-shelf packages and then interconnect them. In particular, fairly complex digital systems can be quickly and easily implemented by using off-the-shelf microcomputers. Consequently a designer need not design individual networks from scratch, though it may occasionally be necessary to so design some small interface networks which interconnect the off-the-shelf packages. Architectural considerations or clever assembling of IC packages is important.

Custom design of large scale integrated circuit packages, that is, **LSI packages**, including custom design of microcomputer packages such as those for automobiles [Puckett, Marley, and Gragg 1977], is becoming popular, as will be explained in Section 2.4. If a designer does not attempt to exert time-consuming efforts in laying out networks on a chip inside an LSI package, packing as many gates as possible, he or she can take an approach similar to the assembling of off-the-shelf packages explained above. In other words, for each custom LSI design, the designer simply assembles standard layouts, which are previously prepared for frequently used networks, on an LSI chip. With this approach an LSI chip can be easily and quickly custom-designed, though best performance is seldom attained. The short design time reduces design cost. On the other hand, the manufacturing cost of the designed LSI is usually high. This can be justified, however, when the LSI chip is part of much more expensive products such as measurement instruments or intelligent terminals.

If a designer wants to attain the best economy or electronic performance achievable with current IC technology, however, logic design is far more difficult and challenging for the reasons presented below. Excellent LSI designers will become the "superstars" of the profession [Petritz 1977].

First, unlike the days when only two logic families, TTL (transistor–transistor logic) for low cost and ECL (emitter coupled logic) for high speed, were available, there now exist a large number of IC **logic families**, that is, different types of electronic circuits to implement gates or logic operations. As these logic families have different features in regard to economy, performance, and size, the designer must single out the most appropriate family. Design procedures for each logic family are different.

Second, networks can be implemented with discrete components or with IC packages of different integration sizes, that is, SSI (small scale integration), MSI (medium scale integration), LSI, or VLSI (very large scale integration). Also, LSI or VLSI packages can be either custom-designed or off-the-shelf packages. The designer must choose the most appropriate ones (possibly a mixture) by considering production volume, completion time, design change possibilities, cost, size, and performance. If frequent design changes are expected, discrete components are still the common choice. When production volume is high and no design change is expected, custom LSI or VLSI design is becoming popular. In this case, if the designer is trying to attain the best electronic performance or economy, custom design is a highly challenging problem, since it is necessary to pack gates as tightly as possible and to find the most appropriate layout under complex layout rules. Whenever new IC processing methods or layout rules are developed, designers usually must redesign even standard networks from scratch.

Third, IC technology has made possible the replacement of gates by memories (mainly, read-only memories), since memories are becoming compact, inexpensive, and fast, and consequently logic networks are often designed with a mixture of gates and memories, in order to reduce cost. (In this case, memories are not used to store computer programs, data, and intermediate computational results, as is the conventional practice in computers.) Finding the best approaches to using memories is not an easy task for designers.

Fourth, enormously large networks are being designed compactly and economically because of the ever-decreasing size and cost of gates, but it is becoming vitally important to design such very large networks without errors, as well as to determine whether or not designed or manufactured IC packages or digital systems are faulty. Logic design must take care of this problem.

In addition, there are many other problems such as reliability of systems or power supplies.