Logic Design of Digital Systems

Third Edition

DONALD L. DIETMEYER

Logic Design of Digital Systems

Third Edition

DONALD L. DIETMEYER
University of Wisconsin—Madison

Allyn and Bacon, Inc.

Boston London Sydney Toronto

Library of Congress Cataloging-in-Publication Data

Dietmeyer, Donald Leo, 1932–
Logic design of digital systems/Donald L. Dietmeyer.—3rd ed.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-205-11294-3
1. Electronic digital computers—Design and construction.
2. Logic design. 3. Logic circuits. I. Title.
TK7888.3.D52 1988
621.395—dc19

87-35227
CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1 92 91 90 89 88

Preface

While the fundamentals of most subjects including logic design change little over a 10-year period, emphasis can and frequently does shift substantially within such an interval. The most difficult part of preparing this revision was determining the shifts that have taken place in the teaching and practice of logic design and attempting to predict those likely to take place in the next few years.

Major suggestions of reviewers and users of the previous editions were to:

(1) Reduce the size of the book
(2) Lower the level of the presentation
(3) Omit electronics
(4) Retain, cut back on, and eliminate the use of design language DDL
(5) Introduce test generation and fault detection

The largest change that will be noted by those familiar with prior editions is the removal of the advanced material on automated logic synthesis. These topics, probably only taught in very specialized graduate-level courses, are slowly becoming of interest in the industry and may reappear in a specialized book. For now they have been sacrificed to reduce the size of this book. Other topics were also removed, abbreviated, or relegated to appendixes in order to reduce this book to a size suitable for use in a one- or two-semester course, while providing instructors choice over the subject matter of their courses. These and other changes were also made to lower the level of the presentation without sacrificing accuracy and completeness. As

a result, this book may still not be as brief and straightforward as some would like. But, then, logic design is not simple and straightforward. For example different ways of expressing and manipulating switching functions are used by different designers and at different stages of system design; a variety of character and graphic symbols for logic operators are used in the industry and hence are presented here.

Reviewers made it clear that digital electronics should be left to texts on that subject. While sections devoted to electronics are no longer present, electronic considerations are no less important to logic design that they ever were. Therefore, a good deal of material has been added that presents those considerations without dwelling on the electronics behind them. Results of circuit simulation with the Spice program are presented to illustrate the definitions and (ill) effects of propagation delay, fan-in and fan-out. Design methods are motivated by these results.

Design language DDL has been removed from the text and replaced in the later chapters with more contemporary notation that has CONLAN [32] syntax and semantics for describing computer hardware. This notation is easy to read after only a small amount of explanation. Other changes, such as the inclusion of material on test generation and fault detection, are emphasized in the chapter review that follows.

Chapter 1 has changed the least. It again begins with a significant preview of the subject of logic design to motivate students and provide a more comprehensive base upon which to build, but less formal notation is used than in the previous edition. The remaining sections treat the more usual topics of codes, number systems, and arithmetic. With improved secondary education, it may be appropriate to skip some of these sections.

While Chapter 2 again presents fundamentals of combination logic, many changes have been made. Digital electronics is no longer reviewed. The development of Boolean algebra has been simplified by an expanded presentation and relegating examples of Boolean algebras to an appendix. The list of ways in which switching functions can be expressed has been expanded to include directed graphs. Simulation as an analysis technique has been expanded here and in other chapters. New sections at the end of Chapter 2 treat the reduction and minimization of switching functions because many feel that these subjects still belong in a first course on logic design. Their placement at the end of the chapter makes it easy for others to omit or partially treat these subjects. This is even more true for the more specialized topics presented in appendixes to the chapter.

Synthesis of combinational networks is the subject of Chapter 3. The new, standard graphic logic symbols are introduced and used at the point where they offer real advantage. Synthesis to programmable

array logic is now illustrated. A new section on electronic considerations includes algorithms for and examples of unit-delay simulations, and means of solving fan-in and fan-out problems. Then faults and testing methods and algorithms are introduced as a part of the design process.

Materials from a number of chapters of the second edition appear in the new Chapter 4 on sequential logic concepts and components. The components are latches and flip-flops, of course; their terminal properties and failure modes are examined, as well as their internal structure and operation. Synchronous finite-state machines are modeled, analyzed, and synthesized before the chapter ends with a limited treatment of asynchronous machine analysis and synthesis. Asynchronous logic is included, not because students are likely to do a great deal of asynchronous design, but to introduce them to the vocabulary and models so that they may better understand failures in digital systems. This chapter is also organized so that the last section(s) may be omitted.

Subsystems in Chapter 5 are registers, buses, multiplexers, adders through ALUs, control machines, and memories. CONLAN notation and ASM charts are useful descriptive tools at this level of organization. Decimal digit systems, error detecting and correcting subsystems, and formal fault detection in finite-state machines are available in appendixes, although elements of test generation and result compression are integrated throughout the chapter.

Chapter 6 utilizes all the previous chapters when designing an elementary digital computer (single-address, 14-instruction types) in great detail and then describing extensions up to a microprogrammed machine with register file and autonomous memory, CPU, and data channel.

All problems at the ends of chapters have been reviewed and many replaced. The set of answers supplied has been extended. More extensive (and interesting) homework problems may be assigned if students have software available to (1) minimize Boolean equations rapidly and (2) simulate gate networks. In addition, simulators for the digital computers designed in Chapter 6 are very helpful in the solution of programming problems and are likely to clarify the operation of these computers for students.

The content of a semester course must depend on the experience and goals of the instructor as well as the level of the students. A first course in switching theory and logic design for a broad spectrum of sophomores might well treat:

Ch. 1 All

Ch. 2
Sec. 2.1, 2.2 All

2.3 Emphasize equations, tables, maps
2.4 All
2.5 Through walkback
2.6 All
2.7 Map simplification

Ch. 3
Sec. 3.1 All
3.2 First part
3.3 All
3.4 Multiplexer and ROM synthesis
3.6 Concepts and definitions only

Ch. 4
Secs. 4.1 to 4.5 All

For a theoretical course, additional materials may be selected from Chapter 2; for a practical course, more time should be spent on Chapters 3 and 5. A second course might concentrate on Chapters 5 and 6. With more mature and/or motivated students, a number of additional sections can be taught at a reasonable pace in a first course. I prefer to conclude a first course with some discussion of an elementary, programmable machine, such as EDC, to put the theory and techniques of the course in perspective.

The author extends special thanks to the following reviewers whose contributions have enriched the text:

Professor Calvin Finn
University of Idaho

Professor Ronald G. Hoelzeman
University of Pittsburgh

Professor Charles Kime
University of Wisconsin—Madison

Professor Paul Stigall
University of Missouri at Rolla

Professor Robert Stuart
Northeastern University

D. L. D.

Contents

1

An Introduction to Digital Systems

Digital systems are assemblages of interacting parts that are capable of storing, communicating, and processing information expressed in discrete form. *Logic design* is the process of solving problems of subsystem and system organization so as to achieve desired information processing, communication, and storage. Other facets of digital system design such as the solution of mechanical, chemical, thermal, electrical, economic, and computer science problems affect logic design. The logic design of digital systems is really much easier than these general definitions suggest, as we can see by examining the words used in the definitions.

1.1 BASIC CONCEPTS OF DIGITAL SYSTEMS

Information—recorded or communicated facts or data—takes a variety of physical forms when being stored, communicated, or manipulated. Printed symbols store and convey information; behind the retina of the eye they take the form of chemical and electrical activity. In telephone and radio communications, information takes the form of voltages and currents that vary with time. Variations in air pressure encode information when people converse. The Native Americans used smoke. The list of examples can go on and on; for any parameter of nature that can be controlled by people can be, and probably has been, used to express information.

All forms that information may take can be classified as being either (1) discrete (composed of distinct parts) or (2) continuous. Numbers printed on paper record information in a discrete manner. A plot based on those numbers and printed on that same paper may record the same information in a continuous form. Usually the hands of a clock move continuously to display the time of day. But some clocks show a sequence of digits, which changes abruptly from time to time to give the same information, but in discrete form.

"Digit" originally meant a finger or toe. These were probably the first tools used by people to assist in counting. As a result, "digit" also refers to symbols commonly used to express numeric information. The symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 are familiar digits. In electronic computation and data processing, letters, punctuation marks, and mathematical symbols are stored, communicated, and processed in much the same fashion as these familiar digits. These other symbols are used on occasion to express numeric information. The definition of "digit" has been generalized as a result. *Digital* has become synonymous with "discrete"; *digit* has become synonymous with "symbol." A set of digits is called an *alphabet.*

Communicating information is moving it through space. In electronic networks, digital information is usually expressed by the value of electrical variables, voltage or current. A different value or range of values of such a parameter is used to express each digit of an alphabet. Wires then provide a means of moving information from one point to another Each of several wires may convey a digit of a digit sequence (parallel communication), or a single wire may convey the digits of a message sequentially in time (serial communication). Such movement does take time: fundamental laws of physics limit the speed with which information can be moved. Electromagnetic waves can travel no faster than 3×10^8 meters per second. It thus takes light and electronic signals at least 3.3×10^{-9} second (3.3 nanoseconds) to travel one meter. Mechanical and acoustic waves travel much more slowly.

Storing information is carrying that information through time. Since communicating takes time, communication mechanisms are sometimes used to provide memory. An electrical signal that is applied at one end of a circuit one meter long arrives at the other end approximately 3.3 nanoseconds later. The applied signal may have changed during that interval. The wire *delays* the supplied signal for a brief, but nonzero interval. *Dynamic* memory units carry information through long periods of time by repeatedly passing that information through a delay line. *Static* memory elements store by using special electronic circuits or by recording the information in a preservative fashion.

Processing information consists of forming new information by altering given information according to specified rules. Arithmetic operations are familiar ways of processing given information to generate new information; we know the rules. We commonly perform many other operations on information without specifying rules in detail. Selecting one of many units of information is an example. Just as electronic signals take time to propagate down lines, they also take time to pass through processing circuits.

Digital systems consist of processing and memory elements that are interconnected by communication mechanisms (see Fig. 1.1). Communicating and processing both take time; units that communicate and process require that the same information be continuously supplied for

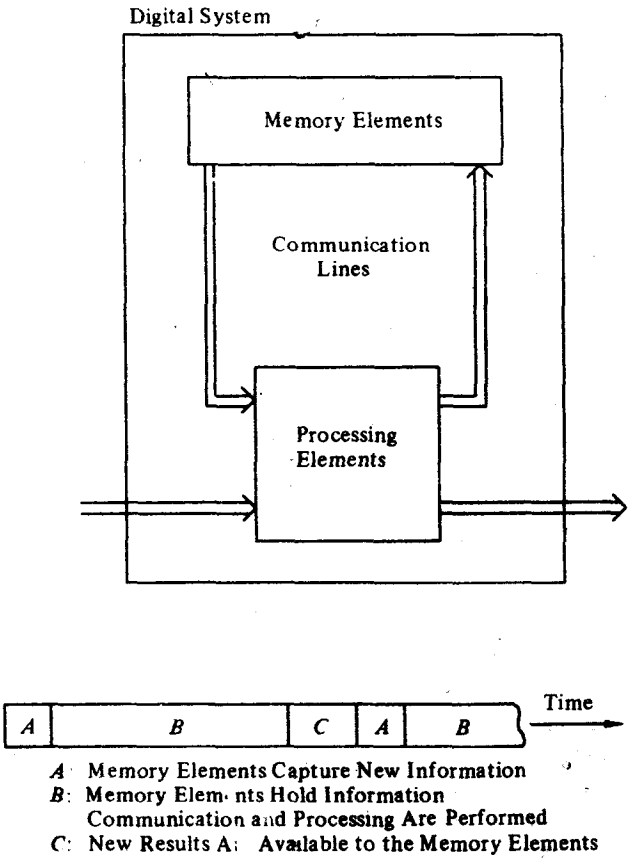

Figure 1.1 Structure and activity cycle of digital systems.

greater than some minimum interval of time. Memory elements hold and continuously supply information while it is carried to the processing elements and processed, and the results are conveyed back to the memory elements. Activity is cyclic: memory elements acquire new information; it is propagated to the processing elements and processed, which usually involves communication between elements; and the new information is propagated to the memory elements. Since useful systems must be able to accept information from their environment and supply information to it, the environment is also pictured in Fig. 1.1 as a source of information to be processed and a sink for results.

Coordination of activity is essential in digital systems. Processing elements must be told which set of rules to use; memory elements must be told when to "forget" old information by capturing new information. It is natural to distinguish *data*, that is, information to be processed to answers, from control information, which governs the processing steps taken to arrive at answers. Figure 1.1 does not distinguish between these two kinds of information; in a digital system both data and control information are stored, communicated, and processed by identical means. Thus, distinctions between types of information are external to the actual system. They may be difficult to make and are not always clear. When adding two numbers, their signs and magnitudes influence the steps taken as well as serve as data.

While the model of digital systems presented in Fig. 1.1 is very general, it is also very vague. The remainder of the book refines this model through detail and examples of specific systems. When considering the details, keep Fig. 1.1 in mind. Digital systems, like forests, consist of huge numbers of leaves, twigs, branches, and the like. By placing details in perspective, we can often avoid becoming lost or entwined in brambles.

1.2 BASIC COMPONENTS OF DIGITAL SYSTEMS

In practice, electronic circuits that can reliably determine what digit a wire conveys to it and in turn generate the electrical representation of the same or another digit deal only with two different digits. Reliability is the key. Digital systems today consist of tens of thousands to millions of amplifying circuits constructed of components with characteristics over which only limited control can be exercised. Transistor characteristics vary from unit to unit, and vary for a given unit with time and temperature, for example. While a few circuits might be made to deal with more than two digits by carefully selecting components and constantly "tuning" the circuits, large

assemblages of circuits cannot be economically manufactured and maintained by such means. Circuit operation must be made as nearly independent of component characteristics as possible. To date, the use of vacuum tubes, diodes, and transistors as switches (i.e., either turned fully on or fully off) has proved to be the most economical method of achieving such independence and hence the required reliability of circuit performance.

Thus, electronic digital information-processing systems deal with information expressed in terms of two digits, commonly symbolized by 0 and 1. The word "binary" expresses this "twoness." A digit from a binary alphabet is a "binary digit" or *bit*. Information theory defines the smallest unit of information as the *bit*. We will not distinguish between these two definitions, but will consider one binary digit to convey one unit of information.

When we wish to express more than one unit of information, more than one binary digit must be used. These digits are linked together (*concatenated*) to form a binary sequence. The small circle ∘ will be used to express the concatenation operation. Thus, linking a 1 on the right of a 0, $0 \circ 1$, gives one binary sequence. Linking a 0 on the right of a 1, $1 \circ 0$, gives a different sequence. Usually we omit the concatenation symbol and simply write the binary digits next to each other in the desired order. Thus, 01, 10, 101101, and so forth, are binary sequences.

Communicating Elements

In a machine, different digits take the form of different levels of some physical quantity such as voltage, current, pressure, or light intensity. A *terminal* is a communication mechanism that is capable of conveying binary expressed information. In electronic systems a terminal is often just a wire (a return path for current is assumed). When the rate with which information is to be conveyed is high, or the distance is long, the wire may take the form of a coaxial cable, perhaps with special driving and receiving amplifiers. Regardless of their physical construction or length, we will initially ignore the delay provided by terminals and think of communication as instantaneous. Even when delay is not ignored, we will assume that terminals have no ability to store information. We will take the delay as introducing a limit on the rate at which information may be presented to a terminal. We must not transmit a bit until the bit preceding it has propagated to the far end of the terminal.

A *signal* is a history of levels of the conveyed quantity (voltage or current) at a specific point in a circuit. The value of a signal changes with time as that level changes (i.e., as different digits are conveyed in

turn). Since terminals were defined to be conveyors of bits, the signals on terminals take values from the binary set $\{0, 1\}$.

It is very useful to assign names to terminals and signals. Confusion is greatly reduced if the same name is used for a terminal and the signal it bears (and later for the generator of that signal). A name to which any of a number of values may be assigned is known as a *variable*. The names assigned to signals are thus mathematical variables. If terminal X bears a binary signal, then signal X takes values from the set $\{0, 1\}$. At one instant variable X may be assigned value 0,

$$X = 0$$

at other instants it may be assigned the value 1,

$$X = 1$$

Variable X has the value of variable C,

$$X = C$$

when terminal X is *connected* to terminal C. Touching two bare wires connects them; cutting a wire disconnects the two halves. We will usually place electronic circuits between terminals so that connections may be made and broken at high speeds.

An ordered collection of n items is known as an *n-tuple*. An n-tuple is usually written as a list of the n items enclosed in parentheses, the items being separated by commas. Thus $(1, 2.5)$ is a 2-tuple and $(0, 1, 0, 0, 1)$ is a 5-tuple. If the items are numeric values, as in these examples, or variables representing values, the n-tuple is called a *vector*. Vectors are valuable to us when a terminal of many wires is used to achieve parallel communication. Terminal X might consist of five wires that can be individually identified with subscripts.

$$X = (X_1, X_2, X_3, X_4, X_5)$$

One way to convey the bits of a binary sequence to another location is to transmit them in turn down a single wire. For example, if binary sequence B consists of five bits, $b_1b_2b_3b_4b_5$, then b_1 is transmitted first. Then b_2 is transmitted, then b_3, and so on. If n bits are to be transmitted and the delay of the terminal is t seconds, then $n \times t$ seconds are required to transmit the entire sequence. While serial communication is time-consuming, it requires a minimum of communication hardware—one wire.

Parallel communication uses a terminal of n wires to convey a binary sequence of n bits. Each bit of the sequence is transmitted on a separate wire. Using terminal X defined above, bit b_1 is conveyed by wire X_1, b_2 is conveyed by X_2, and so on. Parallel communication is n times faster than serial communication, but requires n times as much hardware.

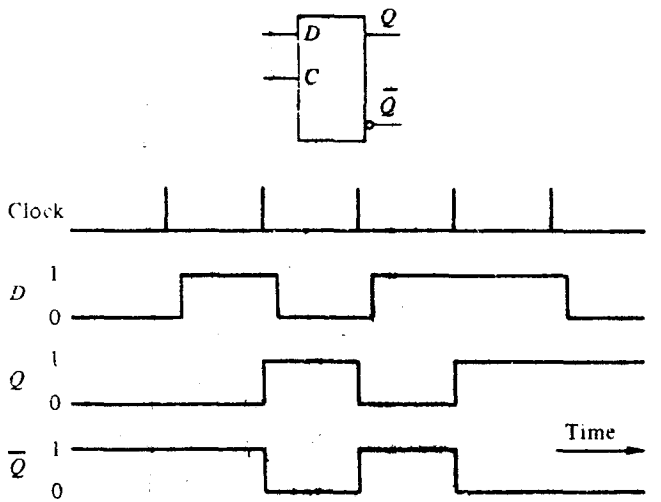

Figure 1.2 The value on the D input terminal just before the clock pulse is captured by the flip-flop and presented on its Q output terminal immediately after the pulse.

Memory Elements

A *memory element* is a system component that is able to store one unit of information until commanded to replace it with another. A variety of electronic circuits is able to hold one bit until commanded to replace it with another. Chapter 4 explores this variety; we will introduce an idealized version of one such circuit as a specific example here. The *D flip-flop* has two input terminals, D and C, and two output terminals as shown in Fig. 1.2. The signal on the named output terminal (Q in Fig. 1.2) indicates the bit being stored; when the flip-flop is storing a 1, variable Q has value 1. The output terminal with the small circle bears the *complementary* signal value. Hence it is labeled with a bar over the name. When Q has value 1, $\bar{Q}$ has value 0; when Q has value 0, $\bar{Q} = 1$. Table 1.1 summarizes this relation

TABLE 1.1
THE NOT RELATION BETWEEN BINARY VARIABLES

Q	$\bar{Q}$
0	1
1	0