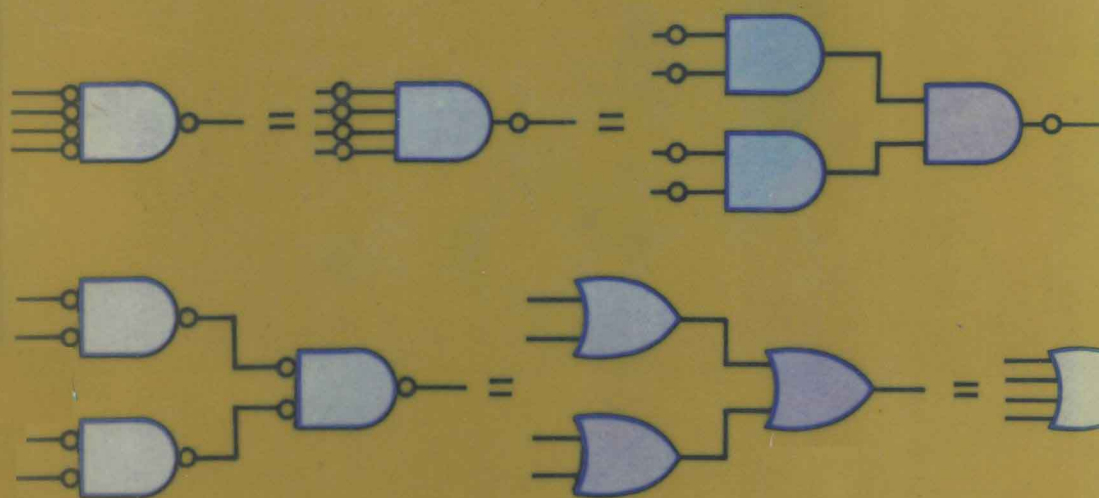


Analysis and Design of Sequential Digital Systems

L. F. Lind and J. C. C. Nelson

7857214



Analysis and Design of Sequential Digital Systems

L. F. Lind

*Department of Electrical Engineering Science,
University of Essex*

J. C. C. Nelson

*Department of Electrical and Electronic Engineering,
University of Leeds*

M

© L. F. Lind and J. C. C. Nelson 1977

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission

First edition 1977

Reprinted 1979

Published by
THE MACMILLAN PRESS LTD
London and Basingstoke
Associated companies in Delhi Dublin
Hong Kong Johannesburg Lagos Melbourne
New York Singapore and Tokyo

ISBN 0 333 19266 4 (hard cover)

0 333 19267 2 (paper cover)

Set in Monophoto Times by
Doyle Photosetting Ltd, Tullamore, Ireland
Printed by Unwin Brothers Limited,
The Gresham Press, Old Woking, Surrey.

This book is sold subject to the standard conditions of the Net Book Agreement.

The paperback edition of this book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser.

Analysis and Design of Sequential Digital Systems

Other titles in Electrical and Electronic Engineering

G. B. Clayton: EXPERIMENTS WITH OPERATIONAL
AMPLIFIERS

G. B. Clayton: LINEAR INTEGRATED CIRCUIT APPLICATIONS

J. C. Cluley: ELECTRONIC EQUIPMENT RELIABILITY

R. F. W. Coates: MODERN COMMUNICATION SYSTEMS

A. R. Daniels: INTRODUCTION TO ELECTRICAL MACHINES

C. W. Davidson: TRANSMISSION LINES FOR COMMUNICATIONS

W. Gosling: A FIRST COURSE IN APPLIED ELECTRONICS

B. A. Gregory: AN INTRODUCTION TO ELECTRICAL
INSTRUMENTATION

Paul A. Lynn: AN INTRODUCTION TO THE ANALYSIS AND
PROCESSING OF SIGNALS

A. G. Martin and F. W. Stephenson: LINEAR MICROELECTRONIC
SYSTEMS

R. G. Meadows: ELECTRICAL COMMUNICATIONS THEORY,
WORKED EXAMPLES AND PROBLEMS

J. E. Parton and S. J. T. Owen: APPLIED ELECTROMAGNETICS

A. Potton: AN INTRODUCTION TO DIGITAL LOGIC

J. T. Wallmark and L. G. Carlstedt: FIELD-EFFECT TRANSISTORS
IN INTEGRATED CIRCUITS

G. Williams: AN INTRODUCTION TO ELECTRICAL CIRCUIT
THEORY

Preface

This text presents the basic information required for successful design of modern logic systems. The information is presented in a form that should be immediately applicable by practising engineers but that also forms the basis of a comprehensive final-year university or polytechnic course. It will also prove useful to postgraduates in disciplines relying on the application of digital methods, for example, computer science, control engineering and instrumentation.

Certain mathematical theorems relating to Boolean variables are excluded for over-all clarity. The use of these theorems to simplify Boolean expressions has, in large part, been superseded by mapping and tabular techniques. Excessive treatment of the mathematical basis of some of the procedures would tend to obscure the engineering concepts involved.

The theoretical material is illustrated with practical examples wherever possible. The examples can be used for extending the reader's knowledge from the particular to the general, which, in many cases, appears to be a more natural approach than the reverse. Wide use of references is made where this is felt desirable, to free the text from detail not immediately relevant and to provide a basis for further reading.

An important feature of the book is the consideration of logic design from a human point of view. An attempt is made to relate the more formal design techniques to the widely used intuitive or 'cut-and-try' approach. For example, the problem of state assignment is simplified by considering only those possibilities that fit in with a natural (or human) approach to the problem. This consideration has repercussions in all phases of the design procedure. One of the most important results lies in the increased understanding of the operations of the circuit by service personnel. Also, by requiring the solution to look 'natural', the number of possibilities is greatly reduced (sometimes to one), thereby reducing the amount of design time required.

The partitioning of one complex circuit into a number of smaller circuits is also carefully considered. To date there is no automatic procedure for doing this; instead, *ad hoc* solutions are generally used. Guidelines are laid down for accomplishing this partitioning in this book. Again, 'naturalness' is the keynote of the partitioning procedure. A well-partitioned circuit should be relatively simple to test and troubleshoot, which leads to enormous practical advantages in the post-design career of the circuit.

Particular attention is devoted to hazards, races and other phenomena that, although not apparent from a superficial appraisal of a proposed design, can lead to mal-operation. Careful attention to these points, together with the information provided in the final chapter on practical implementation, should lead to the production of reliable designs every time.

The authors would like to thank R. Coleman, Technical Director of Trend Communications Ltd, for his assistance with chapter 6. They would also like to thank Mrs S. Nelson for her accuracy and patience in typing the manuscript.

Contents

<i>Preface</i>	vii
1. Introduction	1
1.1 What are Digital Systems	1
1.2 How are Digital Systems Realised?	2
1.3 Binary Representation of Quantities	3
1.4 Serial and Parallel Data	4
1.5 Comparison of Analogue and Digital Systems	4
2. Review of Combinational-logic Techniques	5
2.1 Logic Levels	5
2.2 Gates	6
2.3 The Karnaugh Map	12
2.4 Partitioning	20
2.5 Iterative Circuits	22
2.6 Multiple Outputs	23
2.7 Concluding Remarks	24
2.8 Examples	24
References	28
3. Introduction to Sequential Systems	29
3.1 Fundamental Concepts	29
3.2 Storage Devices	32
3.3 Sequential Sub-systems	38
3.4 Intuitive Design of Sequential Systems	47
3.5 Examples	49
References	53
4. Asynchronous Sequential Systems	54
4.1 Basic Concepts	54
4.2 Analysis Techniques	57
4.3 Races and Hazards	63
4.4 System Design	68
4.5 Examples	87
References	102

5. Synchronous Sequential Systems	104
5.1 Advantages and Disadvantages of Synchronous Systems	104
5.2 Preliminary Design	106
5.3 Flow-chart Method of Design	109
5.4 Pictorial Aids	119
5.5 State Assignment	124
5.6 Examples	126
References	131
 6. Practical Design Considerations	 132
6.1 Initial Specification	132
6.2 Detailed Design	132
6.3 Prototype Development	134
6.4 Printed-circuit Boards	134
6.5 Testing and Documentation	136
6.6 Conclusion	136
References	137
 <i>Appendix</i>	 139
 <i>Index</i>	 141

Introduction

Digital systems are not new. The simple ‘on–off’ signalling techniques used by nineteenth-century telegraphers are one example of an early application of digital techniques. However, for many years development of the various applications of digital principles continued virtually independently; construction of electromechanical telephone exchanges, for example, progressed quite independently of the development of digital computing systems.

Two factors have influenced the present importance of digital-system design as a discipline in its own right. One is the development of design procedures—many of which are discussed in this book—that apply to all digital systems regardless of their particular form of realisation or application. The other is the availability of electronic logic devices in integrated-circuit form, at a price that, a few years ago, would have been regarded as impossibly low. This low price has enabled digital techniques to be used in systems where previously such methods would have been regarded as quite uneconomic.

1.1 What are digital systems?

All methods of specifying quantities are either continuous or discrete. In the former case, for example, the height of a column of liquid in a tube, all heights are possible between zero and some maximum value limited by the length of the tube. A scale might be used to measure the height of the liquid but the liquid itself will move smoothly between the divisions on the scale and, if measurement could be made sufficiently accurate, an infinity of possible heights would be available. Discrete systems, for example, the mileometer (odometer) used on motor cars, have a limited (although perhaps large) number of possible readings or output values. It is normally impossible to specify the distance travelled by the car to better than the nearest whole mile;

if a 'tenths' digit is provided the resolution of the system is improved but the distance can still be specified only to the nearest tenth of a mile.

In general all systems that fall into the discrete category can be regarded as digital, although certain classes of digital system have a more restricted definition. In electrical terminology continuous systems, as opposed to digital ones, are often referred to as *analogue*. This is because most, if not all, physical variables (for example, temperature, pressure, voltage and current) are continuous quantities and it is often convenient to process a voltage or current that is the analogue of some non-electrical quantity. This technique, analogue computing, has been highly developed and is widely used in specialised applications such as control-system simulation. Computers operating in a discrete mode—digital computers—are used in a much wider range of applications.

1.2 How are digital systems realised?

Any system that can be constrained to have a finite number of levels can be regarded as digital. For example, the discs carrying the figures in the mileometer are constrained by mechanical means to have only ten allowed angular positions, as opposed to complete freedom of rotation. The use of ten levels is particularly convenient for systems requiring human involvement. In engineering practice, however, the use of only two levels is particularly attractive. The levels can be widely separated; this allows each level to be severely degraded by deterioration or inadequacy of system components before there is any danger of confusion between the two. For this reason most digital systems operate in a two-level or 'binary' mode, with translation to (or from) ten levels where human intervention is required.

The levels chosen are, very often, merely the presence, or absence, of some physical quantity. In electrical terms this naturally suggests the presence or absence of a voltage (or current); alternatively a circuit, represented by the contacts of a switch or relay, might be 'open' or 'closed'. Hydraulic and pneumatic digital systems are widely used in the control of machinery; here the two levels are represented by the presence or absence of the appropriate pressurised fluid.

Each variable that is permitted to have one of only two possible values is a binary digit (or *bit*). Digital-system design consists essentially of devising an interconnection of processing elements that produces the desired relationship between an input pattern of bits and the required output pattern. In sequential systems, introduced in chapter 3, the output pattern depends not only on the present input pattern but also on previous ones. In view of the use of only two levels, the number of possibilities when acting on a given number of digits can be unambiguously defined. Details are discussed in chapter 2. The basic principles of design discussed in this book apply regardless of the physical form that the binary variables take. However, purely electronic realisation is

by far the most common and this book is written primarily from this point of view. Electronic logic circuits are now normally, but not always, realised in integrated-circuit form. Integrated circuits consist of the components required for realisation of a series of digital functions produced on a single 'chip' of silicon that is only a few millimetres square.

1.3 Binary representation of quantities

An appropriate code must be used if quantities are to be satisfactorily represented over a wide range using binary digits. Although by no means the only possibility, pure binary code is very widely used. In this code adjacent digits are given values (or *weights*) that are related by a factor of two, starting with $2^0 (= 1)$, just as adjacent digits in decimal numbers are related by a factor of ten. In the decimal system the magnitude of each digit must be specified between zero and nine; in the binary system each digit can be only present (*one*) or absent (*zero*). For example, 1101 represents

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13$$

By a similar process decimal fractions can be specified as a sum of binary fractions. Note that, because it normally takes a finite time to convert a physical quantity into binary digits, and these digits can change only at a finite rate, digital quantities are discretised not only with respect to magnitude but also with respect to time (that is, they are *sampled*).

Quantities that can be either positive or negative require an additional digit to specify the sign. Although 'sign-and-magnitude' notation is occasionally used, two's complement notation is more common. In this, the magnitude of a negative number is subtracted from 2^n to form the complement, where n is the number of digits in the magnitude part of the number. This approach has the advantage that subtraction may be performed merely by adding the complement of the number to be subtracted.

Although convenient from a processing point of view, pure binary code leads to hardware complexities when a decimal input or display is required. For this reason, systems in which decimal input or output facilities are a significant part of the total system (for example, digital-voltmeters and pocket calculators) often work in a binary-coded decimal (BCD) mode. In this, numbers are represented in decades, but within the decades the ten levels are specified by means of binary code. For example, 0001 0011 1001 represents 139. The price of easy conversion to and from decimal is that more digits are required and some operations, particularly arithmetic, become more complex. Codes having other weightings (for example, 1, 1, 2 and 5) have been used to represent the ten levels within each decade. These are to be avoided since true binary-coded decimal is widely used and any deviation from this can result in incompatible pieces of equipment.

1.4 Serial and parallel data

It is evident from the previous section that transmission of non-trivial numerical information requires several digits. These digits can be transmitted simultaneously (*in parallel*) or sequentially (*serially*). The former has the advantage of speed but requires one transmission path for each digit. Serial transmission requires only one path regardless of the number of digits but restricts the speed of transmission by an amount that, for a given transmission system, is proportional to the number of digits. Parallel transmission would normally be used over short distances (perhaps within a system). For economy over longer distances a serial method would normally be used. Parallel-to-serial and serial-to-parallel conversion is readily achieved (see chapter 3).

1.5 Comparison of analogue and digital systems

It could be inferred easily from section 1.1 that analogue systems are potentially more accurate than digital ones, which have a finite resolution. In practice this is not the case; in all natural processes there is an inevitable randomness, generally described as *noise*, which means that the infinitesimal resolution that is theoretically possible can never be realised as a reliable measurement. Quantities represented in digital form, however, can often be interpreted with no loss of accuracy even in the presence of considerable noise; but when the noise exceeds a certain threshold, errors can be very large.

These concepts can be illustrated by means of the elementary examples of section 1.1. Unfavourable conditions such as poor visibility, vibration and an unstable location for the observer would decrease the accuracy with which the height of liquid in the tube could be measured. On the other hand the figures on a series of dials, such as a mileometer, could be read without loss of accuracy under such conditions. However, if the conditions were to become particularly poor, a 6 (for example) could be mistaken for an 8 or a 9 and relatively large errors would occur.

Digital systems operating properly below the critical level of noise will always provide the same results for a given series of inputs. This can give rise to a false sense of confidence since errors due to sampling and discretisation will always be present on the input data that the system processes.

In general, therefore, analogue systems are less precise than digital ones but are often faster since they can operate at their maximum rate without the need for sampling. Historically they have also been regarded as simpler and cheaper. However, the development of complex digital integrated circuits has swung the balance in favour of digital realisation for many applications.

2

Review of Combinational-logic Techniques

2.1 Logic levels

Any logic device (or indeed, system) can be viewed as a processing unit. Certain inputs are applied, and certain outputs then appear. These inputs and outputs can take many different forms. For example, they can be voltages, currents, pressure, light, etc., or a mixture of these variables. In the majority of logic devices currently available, the inputs and outputs are voltages. For definiteness, all inputs and outputs in the ensuing discussion will be assumed to be voltages unless stated otherwise.

Having agreed to select voltages to represent inputs and outputs, the next step is to specify how voltage waveforms should be used to convey information. The information of concern is assumed to be a binary digit (bit), which can have only one of two possible values. These values can have various meanings. For example, true, false; on, off; *one*, *zero*; 1, 0 are some of the meanings that have been attached to these two values. The voltage waveforms necessary to distinguish between these values could, in principle, be chosen in a great variety of ways. For example, two different sinusoidal frequencies could be used, or two different phases of one frequency (with respect to some reference phase) could be employed. Alternatively, a square wave and a triangular wave could be used, or two rectangular waves with different mark-to-space ratios. The use of two different d.c. levels could also be considered.

The last possibility is very attractive from the design point of view, since it will probably result in a less complicated circuit than the other ideas mentioned. The use of d.c. levels can be related to the well-known switching properties of transistors (which can be manufactured in integrated-circuit form). It is for this reason that the two values of a bit are usually associated with *d.c.* levels, and are referred to as the *logic* levels of the bit.

Strictly speaking, the association of a logic level with a d.c. level is not

correct. The logic level in practice must cover a *range* of d.c. levels. It is necessary to have a range in order to cope with the problems of noise pick-up, varying power supply, ageing of components, etc. In transistor–transistor logic, for example, inputs are allowed in the ranges -1.5 to 0.8 V (logic level *zero*) and 2.0 to 5.5 V (logic level *one*). The corresponding output ranges are 0 to 0.4 V and 2.4 to 5.0 V. It is seen that both input ranges are somewhat larger than the corresponding output ranges. The minimum difference in ranges is defined to be the *worst-case* noise margin for each logic level. For example, logic level *one* has a worst-case noise margin $2.4 - 2.0$ V = 400 mV. In this case the same noise margin exists for logic level *zero*.

For both input and output ranges there exists a forbidden band of values. If an input exists in this forbidden band, the output might be anywhere from 0 to 5 V (which includes its forbidden band). Whatever voltage the output assumes, this value could change dramatically if the original device were replaced by another of the same type. For this reason, operation in the forbidden bands is avoided in practice.

There are many textbooks that give detailed design information on logic levels and noise immunity for various logic families. The reader is referred to Motorola (1968, 1973) for more details.

2.2 Gates

The majority of the logic gates that are encountered in practice are of the multiple-input/single-output type. Thus the gate can be considered as an *information-combining* unit. It accepts several input bits of information and then processes these bits to produce only one output bit. In a sense, information is being destroyed (or entropy increased) in the gate. This entropy point of view has been developed by Matheson (1971a, 1971b).

An important feature of electronic gates is that of unidirectional operation. No matter what signals are impressed on a gate output, the input levels normally remain unaffected. This situation is not true, for example, with relay circuits. With these bilateral circuits, backward paths can occur, which might give rise to incorrect operation.

The simplest way to describe the action of a gate is to prepare an input/output table for the gate. Such a table is conventionally referred to as a *truth table*. This table should show the resulting output for all possible input combinations. Inputs will be denoted by x_1, x_2, \dots, x_n , and the output by z . All these variables are binary, and so can assume only the logic levels *zero* or *one* as discussed previously.

x_1	x_2	z
0	0	0
0	1	0
1	0	0
1	1	1

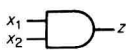


Figure 2.1 Truth table and symbol for a two-input AND gate.

As a first example the truth table of a two-input AND gate will be considered. This truth table is shown in figure 2.1, where logic level *one* is indicated by 1, and logic level *zero* is indicated by 0. Note that all possible input combinations are listed in the table. The symbol for this gate is also shown in figure 2.1. The gate symbols used in this book are in accordance with MIL STD 806 B. It is common practice for manufacturers to use this symbol set. In appendix 1 a list of equivalences is given between these symbols and other widely used equivalents.

The action of this gate can also be described algebraically, as follows

$$z = x_1 x_2$$

In this notation 'multiplication' indicates the AND operation. Letting x_1 and x_2 assume definite values and referring to the truth table of figure 2.1, it is seen that

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

which explains why the AND operator is represented by multiplication. It is obvious that $z = x_1 x_2 = x_2 x_1$, which shows that the AND operation is commutative.

The action of the AND gate can be extended to the n inputs x_1, x_2, \dots, x_n . Then $z = x_1 x_2 \dots x_n$, which indicates that $z = 1$ only when all $x_i = 1$.

x_1	x_2	z
0	0	0
0	1	0
1	0	0
1	1	1

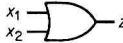


Figure 2.2 Truth table and symbol for a two-input OR gate.

The truth table and symbol for the two-input inclusive OR gate are shown in figure 2.2. Algebraically this truth table is represented as

$$z = x_1 + x_2$$

where the plus sign indicates the OR operation. Again, letting x_1 and x_2 assume definite values, the table shows that

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

which agrees with our notion of addition (except for the last line). This gate is called inclusive OR because the last line gives $z = 1$ for both $x_1 = 1$ and $x_2 = 1$. The operation is again commutative, for $z = x_1 + x_2 = x_2 + x_1$. There is

another gate, the exclusive OR gate, which has the property $z = 1$ when either but not both inputs are at *one*. This gate excludes or prevents the output becoming *one* when the two inputs are simultaneously at *one*. The exclusive OR operation is also commutative.

In the inclusive OR gate, let n inputs x_1, x_2, \dots, x_n be applied. Then $z = x_1 + x_2 + \dots + x_n$, with the result that z is *one* whenever one or more of the inputs are at *one*.

Some simple rules for the ‘addition’ and ‘multiplication’ of logic expressions are now given. Recalling that 1 (or *one*) stands for ‘on’ and that 0 (or *zero*) stands for ‘off’, it is readily seen that for some logic expression g , $1 \cdot g = g$, $0 \cdot g = 0$, $1 + g = 1$, $0 + g = g$. It also turns out that $(e + f)(g + h) = eg + eh + fg + fh$, as is the case for algebraic multiplication. With a combination of these rules, various results can be obtained. For example, $(x + g)(x + h) = xx + xh + xg + gh = x + x(g + h) + gh = x(1 + g + h) + gh = x \cdot 1 + gh = x + gh$. Rather than develop a long list of such specialised formulae, it is quite often easier to work directly with Karnaugh maps, as will be seen subsequently.

Returning to the discussion of gates, the simplest gate of all is the inverter of figure 2.3. This gate can be represented by a circle or a triangle followed by

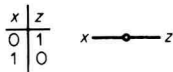


Figure 2.3 Truth table and symbol for an inverter gate.

a circle, and has the property of complementing (or loosely, ‘inverting’) the input. The output is often indicated by \bar{x} , where the bar indicates the complement (or logical inverse) of x . The definition of a logical inverse is that $x + \bar{x} = 1$. From the truth table, it is readily seen that $x\bar{x} = 0$ and $(\bar{\bar{x}}) = x$.

The first equation will be central to the Karnaugh map, which is discussed in section 2.3. The last equation will prove to be extremely useful for logic-circuit transformation. In words, this equation states that two (or any even number, for that matter) inverters in series cancel each other out. Conversely, it is possible to ‘grow’ two inverters on a connection wire without disturbing the operation of a logic circuit.

The inverter can be combined with the AND- and OR-gate outputs to create the NAND and NOR gates, respectively. The N indicates that the output is negated (that is, inverted). The symbols and truth tables for these gates are shown in figure 2.4. In this figure it is seen that if $x_1 = x_2$ both of these gates reduce to inverters.

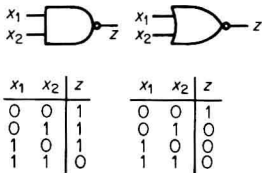


Figure 2.4 Symbols and truth table for two-input NAND and NOR gates.