introduction to SWITCHING THEORY AND LOGICAL DESIGN

SECOND EDITION

Fredrick J. Hill

Professor of Electrical Engineering
University of Arizona

Gerald R. Peterson

Professor of Electrical Engineering
University of Arizona

JOHN WILEY & SONS

New York *London* *Sydney* *Toronto*

Library of Congress Cataloging in Publication Data:

Hill, Frederick J.
Introduction to switching theory and logical design.

Includes bibliographies.
1. Switching theory. 2. Digital electronics.
I. Peterson, Gerald R., joint author. II. Title. III.
Title: Switching theory and logical design.

TK7868.S9H5 1974 621.3819'58'2 72-21783
ISBN 0-471-39882-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Preface

In the field of digital computers, there is an area somewhere between circuit design and system formulation which is usually identified as switching theory. Although this topic must be mastered by the digital computer engineer, it does not answer all the questions concerning layout of an efficient digital computing system. The remaining questions, which are less susceptible to formalization, may be grouped under the heading "Digital System Theory."

Our approach is to follow the basic framework of switching theory. We hope, however, to motivate the student by presenting examples of the many problems which appear repeatedly in the design of digital systems. It is not our intent to provide the student with a detailed knowledge of such specialized system topics as computer arithmetic. Instead, we attempt to provide the framework through which the student might develop a sound design philosophy applicable to any digital design problem. Thus, we do not feel that the use of the term "logical design" in the title is unjustified.

The usefulness of switching theory is not restricted to engineers actively engaged in computer design. In fact, it is our contention that almost every design engineer in the field of electronics will have some opportunity to draw on this subject. Applications occur whenever information in communication, control, or instrumentation systems, for example, is handled in other than analog form.

The above three paragraphs, which began the preface of the First Edition of this book, seem to us to be even more accurate today than they were six years ago. In these years, digital techniques have become virtually standard in many areas once considered the exclusive preserve of analog techniques. This "digitalization" of electrical engineering has become so pervasive that it is a rare electrical engineer who will not have some contact with digital design problems.

We undertook preparation of this second edition in order to strengthen some of the pedagogical weak spots of the first edition as well as to include new material made necessary by recent developments, particularly in the area of integrated circuits. As an example of the latter, we have rewritten

Chapter 3 Truth Functions

3.1 Introduction

The power of many digital systems, particularly computers, is largely dependent on the capability of their components to (1) make decisions and (2) store information, including the results of decisions. We shall defer consideration of the storage capability until Chapter 9 and first explore the problem of physical implementation of logical decision processes.

Digital computers are not the only physical systems which utilize electronic components to make logical decisions, although in digital computers this practice has reached its highest degree of sophistication. In order to explore the relation of electrical systems to logical decisions, let us consider a relatively simple design problem.

A logic network is to be designed to implement the seat belt alarm which is required on all new cars. A set of sensor switches is available to supply the inputs to the network. One switch will be turned on if the gear shift is engaged (not in neutral). A switch is placed under each front seat and each will turn on if someone sits in the corresponding seat. Finally, a switch is attached to each front seat which will turn on if and only if that seat belt is fastened. *An alarm buzzer is to sound when the ignition is turned on and the gear shift is engaged provided that either of the front seats is occupied and the corresponding seat belt is not fastened.*

These conditions are sufficiently simple that the reader could most likely work out a circuit design by trial and error. We shall take another approach

requiring a formal analysis of the statements in the previous paragraph. In this way we shall lead up to formal procedures which can be applied to much more complicated systems.

The statements in the design specification may be listed as follows:

The alarm will sound:	A
The Ignition is on:	I
The Gearshift is engaged:	G
The Left front seat is occupied:	L
The Right front seat is occupied:	R
The Left seat Belt is fastened:	B_L
The Right seat Belt is fastened:	B_R

To each statement, we have assigned a *statement variable*, which will represent the statement and which may take on a *truth value*, **T** or **F**, according to whether the corresponding statement is true or false. In general, we define a statement as any declarative sentence which may be classified as true or false.

The mathematics of manipulating statement variables and assigning truth values is known as *truth-functional calculus*. The application of truth-functional calculus is not restricted to simple positive statements such as those above. First, for any positive statement, there is a corresponding negative statement. For example,

"The left seat belt is *not* fastened": $\bar{B}_L$

Since the statement $\bar{B}_L$ is true whenever B_L is false and is false whenever B_L is true, $\bar{B}_L$ is called the *negation* of B_L. (Other commonly used symbols for the negation of a statement A are $\sim A$ and A'.)

Consider next this statement:

"The left seat belt is not fastened
and the left front seat is occupied": $\bar{B}_L \wedge L$

This is a *truth-functional compound*, a compound statement, the truth value of which may be determined from the truth values of the component statements. The exact relationship between the truth of the component statements and the truth of the compound statements is determined by the *connective* relating the components. In this case, the connective is AND, indicating that the statement $\bar{B}_L \wedge L$ is true if and only if both the component statements, $\bar{B}_L$ AND L, are true. Since AND is a common relationship between statements, we shall assign it a standard symbol, $\wedge$, so that a compound of two statements, A and B, related by AND may be represented by $A \wedge B$.

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
F	F	F	F	F
F	T	F	T	T
T	F	F	T	T
T	T	T	T	F

FIGURE 3.1

There are only four possible combinations of truth values of the two component statements A and B. We may, therefore, completely define $A \wedge B$ by listing its truth values in a four-row table such as Fig. 3.1. A table of this form is called a *truth table*.

This table also defines $A \vee B$, which symbolizes a statement which is true if and only if either statement A is true, statement B is true, or both of these statements are true. Common usage of the word "or" is not always in agreement with the definition of $A \vee B$. Consider the sentence

"Joe will be driving either a blue sedan or a green convertible."

Clearly, it does not mean that Joe might be driving both cars. The latter usage of "or" is called the *exclusive-or* and is symbolized by $A \oplus B$. The truth values of $A \oplus B$ are also tabulated in Fig. 3.1. The connective $\vee$ is known by contrast as *inclusive-or*. Since this connective turns out to be more common in logical design problems than the exclusive-or, it is standard to refer to it simply as OR, leaving the "inclusive" understood.

One more connective, $\equiv$, must be defined before truth-functional calculus can be applied to the seat belt problem. By $A \equiv B$ it is meant that A and B always have the same truth value, i.e., A is true if and only if B is true. This connective is tabulated in Fig. 3.2.

Let us now attempt to represent the information contained in the verbal specification of the alarm system by a truth-functional equation. Rephrasing the specifications in terms of the simple statements, we see that *the alarm will sound* (A) if and only if *the ignition is on* (I) and *the gearshift is engaged* (G) and *the left front seat is occupied* (L) and *the left seat belt is not fastened*

A	B	$A \equiv B$
F	F	T
F	T	F
T	F	F
T	T	T

FIGURE 3.2

($\bar{B}_L$), or *the right front seat is occupied* (R) and *the right seat belt is not fastened* ($\bar{B}_R$). As a truth functional equation, all this is simply written as

$$A \equiv I \wedge [G \wedge ((L \wedge \bar{B}_L) \vee (R \wedge \bar{B}_R))]. \tag{3.1}$$

The design process is certainly not complete, but we have seen that the truth-functional calculus enables us to put the specifications in a form that is compact, unambiguous, and suitable for mathematical manipulation. It should be noted that the designer must be able to express the specifications in the form of statements related by clearly defined connectives. Not all declarative sentences are subject to such expression. Consider the sentence

"The alarm sounded because the right seat belt was not fastened."

This is not a truth-functional compound, since the truth of the statement cannot be determined solely from the truth of the component statements. For example, even if both components are true, the compound statement may not be true, since the right seat may have been unoccupied, in which case the latter half of the statement is true, but the alarm was set off by the driver not fastening his seat belt. Such sentences cannot be handled by truth-functional calculus. In the next four sections, it will become clear that the basic usefulness of electrical circuits in making logical decisions is limited to situations in which truth-functional calculus is applicable. Thus, the designer must avoid formulating a design statement in such a way that it does not constitute a truth-functional compound.

3.2 Binary Connectives

The connectives defined by the truth tables of Figs. 3.1 and 3.2 are *binary*, since they relate only two component statements. Each binary connective corresponds to a unique assignment of truth values to the four rows of the truth table, corresponding to the four possible combinations of truth values of two component statements. There are $2^4 = 16$ ways of arranging T's and F's on four rows, so there are sixteen possible binary connectives, as tabulated in Fig. 3.3. Each connective is numbered, and symbols are indicated for those already defined and for others of particular interest.

Note particularly connective 13, which may be written

$$A \supset B$$

and which represents the compound sentence:

"If A is true, then B is true."

A	B	0	1 ($\wedge$)	2	3 (A)	4	5 (B)	6 ($\oplus$)	7 ($\vee$)	8 ($\downarrow$)	9 ($\equiv$)	10 ($\bar{B}$)	11	12 ($\bar{A}$)	13 ($\supset$)	14 ($\uparrow$)	15
F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
F	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
T	F	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
T	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

FIGURE 3.3

This is similar to the *if and only if* statement discussed previously, except that B is not necessarily false if A is false. For example, consider the statement

$$(2 > X > 1) \supset (X > 0) \tag{3.2}$$

If $X = .5$, for example, the statement $X > 0$ is true; but the statement $2 > X > 1$ is false. There is no contradiction to statement 3.2, and it is true as defined in Fig. 3.3.

The truth values for the *if-then* statement may seem unnatural at first, but this concept is necessary in many mathematical arguments. The distinction between $\supset$ and $\equiv$ is very important. One is required to prove both

$$A \supset B \quad \text{and} \quad B \supset A$$

in order to assert that

$$B \equiv A \tag{3.3}$$

We shall leave it as a problem for the reader to prove that $(A \supset B) \wedge (B \supset A)$ means the same as $B \equiv A$. That is,

$$(B \equiv A) \equiv [(A \supset B) \wedge (B \supset A)] \tag{3.4}$$

Since the logic designer is often concerned with converting natural language specifications into precise mathematical form, he should keep in mind that, in conventional usage, *if-then* may be thoughtlessly used when *if and only if* is intended. For example, a carelessly written specification might say:

"If the switch is closed, then the circuit output will be 5 volts,"

although what is almost certainly meant is:

"The circuit output will be 5 volts if and only if the switch is closed."

The designer thus has a dual responsibility—to be precise in his own writing of specifications and to be careful in interpreting specifications written by others.