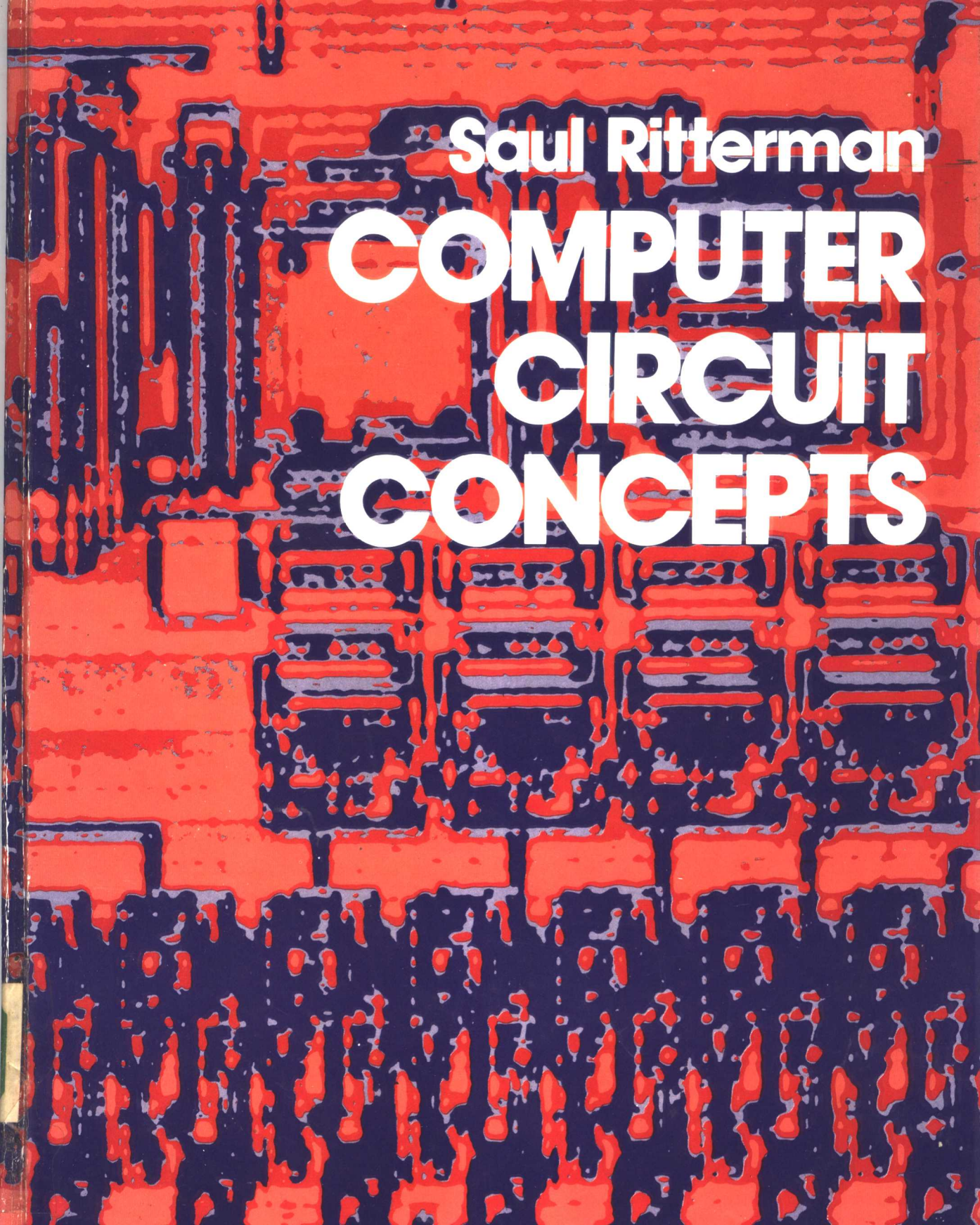
Saul Ritterman
COMPUTER
CIRCUIT
CONCEPTS

COMPUTER CIRCUIT CONCEPTS

Saul Ritterman

Bronx Community College
City University of New York

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá
Hamburg Johannesburg London Madrid Mexico Montreal
New Delhi Panama Paris São Paulo Singapore Sydney
Tokyo Toronto

COMPUTER CIRCUIT CONCEPTS

1234567890DOCDOC898765

ISBN 0-07-052952-3

This book was set in Times Roman.
The editors were Sanjeev Rao and J. W. Maisel;
the designer was Elliot Epstein;
the production supervisor was Phil Galea.
The drawings were done by J & R Services, Inc.
R. R. Donnelley & Sons Company was printer and binder.

Cover photograph courtesy of Advanced
Micro Devices, Inc., Sunnyvale, California.

Library of Congress Cataloging in Publication Data

Ritterman, Saul A.
Computer circuit concepts.

(McGraw-Hill series in electrical engineering. Computer engineering)
Includes index.
1. Computers—Circuits. I. Title. II. Series.
TK7888.4.R58 1986 621.3819'5835 85-6687
ISBN 0-07-052952-3

PREFACE

The first industrial revolution began in the 1750s. Human labor, in many cases very skilled, was gradually replaced by machines. Changes caused by the first industrial revolution affected more than manufacturing processes. The entire social and economic fabric of civilization changed permanently. The second industrial revolution began some 200 years later, when computers began taking over operations performed by humans. This revolution began in the 1960s with the introduction of integrated circuits (ICs), which shrink the size and power requirements of electronic circuits. Results have been astounding.

Until quite recently diverse activities such as home entertainment, marketing, business office operations, and manufacturing were conducted without benefit of computers. This is no longer the case. Stereo tuners are digital and TV sets are remote control. Talking cash registers read package codes, present itemized bills, and announce correct change. Typewriters have become word processors, and business decisions are based on computer analysis of data. In manufacturing, computer-controlled robots are becoming the work force. Clearly, computers have and will continue to have profound effects on our lives.

Proliferation of computer texts is a by-product of the computer revolution. On the surface these texts plow similar ground with equal attention to equally important topics. However, equal attention is an illusion. Chapter lengths as well as numbers of sections per chapter vary considerably. As a result, equally important topics cannot receive equal treatment. One text stresses systems while another stresses circuits. Some texts emphasize computer mathematics while others are in effect catalogs for a specific family of chips. Still other texts delve into minute detail about the author's favorite programming language. These topics are important, but not equally important.

Unbalanced treatment results in an unbalanced understanding of computer operation. By selecting a specific text, one becomes proficient in machine language programming but has only a hazy understanding of how a

computer performs calculations. Similarly, another text stresses Boolean algebra but fails to demonstrate that the utility of microcomputers lies in ease of interfacing with the outside world. A sense of proportion is needed.

This book presents a balanced approach. There are nine chapters of equal importance and approximately equal length. Moreover, each chapter contains eight sections. Equal treatment of equally important topics is logical and also simplifies studying. Furthermore, topics are arranged in sequential order.

Each chapter builds on the previous:

Chapter 1: Introduces Boolean algebra, the basis of computer operation.

Chapter 2: Extends Boolean algebra to binary numbers because digital computers are really binary computers.

Chapter 3: Describes gates, the basic circuits for implementing Boolean and binary operations.

Chapter 4: Introduces flip-flops, groups of gates connected for storage and transfer functions.

Chapter 5: Introduces counters and registers. These circuits use flip-flops and are the lowest level of complex ICs used in modern computers.

Chapter 6: Describes arithmetic operations. Arithmetic circuits are extensions of counters and registers.

Chapter 7: Describes the memory circuits. Storing arithmetic results and instructions is vital to computer operation.

Chapter 8: Discusses peripheral devices. Previously described techniques are used to connect computers to the outside world.

Chapter 9: Combines material developed in all previous chapters to obtain a complete computer.

Each chapter begins by stating an objective and ends with summary; this approach emphasizes the thrust of each chapter. As a further study aid, answers to odd-numbered problems are presented along with the problems. Typically, answers are in the rear of a book, but there is no educational benefit to flipping pages back and forth between problems and answers.

In conclusion, this book presents a chain of equally important computer topics. The reader comes away with a balanced understanding of what a computer does and how it does it.

ACKNOWLEDGMENTS

In a real sense my students wrote this book. The subject matter and treatment are in response to their needs. I would like to express my thanks for the many

useful comments and suggestions provided by colleagues who reviewed this text during the course of its development, especially to R. Dale Anderson, Iowa State University; R. H. Berube, Community College of Rhode Island; David Beyer, Middlesex County College; A. O. Brown; Denton Dailey; S. W. Director, Carnegie-Mellon University; Frank Duda, Grove City College; Antulio Gomez, El Camino Community College; Bernard Harris, Manhattan College; Martin Kaliski, Northeastern University; W. H. McDonald, U.S. Merchant Marine Academy; Michael Miller, DeVry Institute of Technology; John L. Morgan, Neury Institute of Technology; John Pavlat, Iowa State University; Lee Rosenthal, Farleigh Dickinson University; and M. Silevitch, Northeastern University.

Saul Ritterman

CONTENTS

1
LOGIC

1.0 INTRODUCTION

Electronic devices are useful in many applications. In particular, electronic computers perform all sorts of computations rapidly.

Modern computer circuits are based on a single characteristic of semiconductor devices, the ability to switch between two states in extremely short times. Every computer circuit only has two possible states, on or off. A special set of rules exists for working with two-state devices. These rules, developed by George Boole in the 1840s, are known as *Boolean algebra*. Originally Boolean algebra was developed for analyzing the validity of philosophical arguments, but in the 1930s Claude Shannon adapted Boolean algebra to designing electronic switching circuits. Shannon's work can be considered as the beginning of the modern electronic computer.

Since Boolean algebra is the basis of computer operation, this book begins with a discussion of Boolean algebra. In effect, computer circuits are the electronic version of Boolean equations. Since simpler equations result in simpler circuits, techniques for reducing Boolean equations are investigated; both algebraic and graphical reduction methods are practical. This chapter discusses constructing circuits to perform Boolean operations. The methods presented here are used throughout the entire book.

1.1 BOOLEAN ALGEBRA

Boolean algebra only considers two possibilities. A quantity either exists or it does not exist. Existence is represented by the number 1 and nonexistence by the number 0. For example, if A represents airplanes then

$$A = 1 \tag{1.1a}$$

means there are airplanes, while

$$A = 0 \tag{1.1b}$$

means there are no airplanes.

TABLE 1.1
Developing a Truth Table

A	A	$\bar{A}$	A	$\bar{A}$	$\bar{\bar{A}}$
0	0	1	0	1	0
1	1	0	1	0	1

(*a*) (*b*) (*c*)

Boolean algebra relies on precise definitions of the three words NOT, OR, and AND, which are capitalized to emphasize Boolean rather than ordinary meaning.

In Boolean algebra NOT means *opposite*. Boolean equations indicate NOT as a bar over the quantity. If

$$A = 1 \quad \text{then} \quad \bar{A} = 0 \tag{1.2a}$$

Similarly if $A = 0$ then $\bar{A} = 1$ (1.2*b*)

These Boolean equations apply to airplanes or any other quantity represented by the symbol A. If a quantity exists, then its opposite does NOT exist, and vice versa.

A NOT statement is sometimes called an *inverse*. Double inverts are possible. For example,

$$\bar{\bar{A}} = A \tag{1.3a}$$

This equation says that the inverse of an inverse is the original quantity. Again, if A represents airplanes, then $\bar{A}$ represents no airplanes. Similarly, $\bar{\bar{A}}$ represents the inverse of no airplanes, which is airplanes. Equation (1.3*a*) can be extended to more than two inverts:

$$\bar{\bar{\bar{A}}} = \bar{A} \tag{1.3b}$$

Boolean equations can be displayed as *truth tables*. A truth table is a tabulated presentation of all possible values of the quantity. A truth table for Eq. (1.3*a*) begins with A as the first column; as shown in Table 1.1*a*, the only possible values of A are 0 and 1. Table 1.1*b* shows the corresponding values of $\bar{A}$, which are given by Eqs. (1.2). Table 1.1*c*, the completed truth table, shows the corresponding $\bar{\bar{A}}$ values, which are also determined from Eq. (1.2). Since the A and $\bar{\bar{A}}$ columns are identical, the truth table verifies Eq. (1.3*a*).

OR is the second Boolean term. The meaning of OR is that *at least one member* of the class *exists*. The Boolean symbol for OR is a plus sign. For example

$$A + \bar{A} = 1 \tag{1.4a}$$

means that either A OR its inverse exists.

An OR truth table can be used to verify Eq. (1.4*a*). The first column of Table 1.2 contains all possible values of A, and the second column contains the corresponding values of $\bar{A}$. In the last column, the results of ORing A with $\bar{A}$ are shown. Since OR means the existence of at least one member, if a single

TABLE 1.2
$A + \bar{A}$ Truth Table

A	$\bar{A}$	$A + \bar{A}$
0	1	1
1	0	1

1 exists in the classes being ORed, the result of ORing is 1. Since Table 1.2 shows all 1s in the $A + \bar{A}$ column, Eq. (1.4*a*) is valid.

Another Boolean relationship is ORing A with 0.

$$A + 0 = A \tag{1.5a}$$

This equation may seem trivial, but it is important. The validity of Eq. (1.5*a*) can also be demonstrated with a truth table.

Any Boolean quantity can be ORed with itself

$$A + A = A \tag{1.6a}$$

Table 1.3 is the truth table for this equation. Since A is ORed with A, the first and second columns are identical. The third column shows the result of ORing A with itself. The first and third columns are identical, and thus Eq. (1.6*a*) is valid.

$A + \bar{A}$ can be considered as adding $\bar{A}$ to A. In fact ORing is sometimes called *logical addition*. However, in complicated Boolean expressions the similarity to addition breaks down. This truth table also shows why the OR symbol should be read as OR and not a plus sign: 1 + 1 does not equal 1 in ordinary arithmetic, but 1 OR 1 does equal 1 in Boolean algebra.

AND is the last Boolean word. AND means *all* members of the class. Thus AND has a more restricted meaning than OR. While OR can be used in the sense of one, several, or all, AND requires the inclusion of each and every quantity. The symbol for AND is a dot between the quantities being ANDed

$$A \cdot A = A \tag{1.6b}$$

AND statements can also be verified with truth tables. The result of ANDing can only be 1 if all quantities are 1. Table 1.4 shows the truth table for this equation.

Table 1.4 resembles ordinary multiplication, and ANDing is sometimes referred to as *logical multiplication*. As with ORing, there is only partial agreement. It is safer to read a dot in Boolean algebra as AND instead of a times sign.

TABLE 1.3
$A + A$ Truth Table

A	A	$A + A$
0	0	0
1	1	1

TABLE 1.4
***A*·*A* Truth Table**

A	*A*	*A*·*A*
0	0	0
1	1	1

It is convenient to group Eqs. (1.6*a*) and (1.6*b*) together because in both cases a quantity is combined with itself. Similarly, the AND relationships below can be grouped with Eqs. (1.4*a*) and (1.5*a*):

$$A \cdot \bar{A} = 0 \tag{1.4b}$$

$$A \cdot 0 = 0 \tag{1.5b}$$

These results differ from the corresponding OR equations because OR and AND have different meanings. If necessary, Eq. (1.4*b*) and (1.5*b*) can be verified with truth tables.

There is another pair of equations which involve a single quantity. One equation ORs the quantity with 1, and the other ANDs it with 1.

$$A + 1 = 1 \tag{1.7a}$$

$$A \cdot 1 = A \tag{1.7b}$$

Since A and $\bar{A}$ are opposite members of the same class, Eqs. (1.1) through (1.7) contain only one variable. When Boolean statements contain more variables, truth tables contain more conditions. If two variables exist, there are four possible combinations. Table 1.5 lists the possibilities for variables called A and B.

Actually Table 1.5 is the beginning of counting in the binary number system, a topic discussed in the next chapter. For now it is sufficient to present a pattern which guarantees all possible combinations of variables. In Table 1.5 the top row column begins with 0 and alternates 0s and 1s. The bottom row column also begins with 0 but alternates pairs of 0s and 1s.

Equations (1.8*a*) and (1.8*b*) are the Boolean *commutative* OR and AND equations.

$$A + B = B + A \tag{1.8a}$$

$$A \cdot B = B \cdot A \tag{1.8b}$$

It happens that conventional algebra is also commutative. But this is not a proof that Boolean ORing and ANDing can be performed in any order, since differences between Boolean and conventional algebra have been observed. A truth table verifies a Boolean equation; the table for Eq. (1.8*a*) is shown In

TABLE 1.5
Two-variable Combinations

A	0	0	1	1
B	0	1	0	1

TABLE 1.6
Commutative OR Table

A	B	$A + B$	$B + A$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Table 1.6. The first two colums list all possible combinations of two variables, the third column ORs A with B, and the fourth column ORs B with A. If at least one quantity is 1, the result of ORing is 1. Since the third and fourth elements of Table 1.6 are identical, Eq. (1.8*a*) is valid. Equation (1.8*b*) can be verified in the same manner.

Having admitted that two variables are possible, we can also consider three variables; Eqs. (1.9*a*) and (1.9*b*) are the three-variable *associative* equations.

$$A + (B + C) = (A + B) + C \tag{1.9a}$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C \tag{1.9b}$$

Similarly, Eqs. (1.10*a*) and (1.10*b*) are the three-variable *distributive* Boolean equations:

$$A \cdot (B + C) = A \cdot B + A \cdot C \tag{1.10a}$$

$$(A + B) \cdot (A + C) = A + B \cdot C \tag{1.10b}$$

When three variables occur, there are, as shown in Table 1.7, eight possible combinations. Listing all combinations of three variables is an extension of the two-variable method.

- C alternates between 0 and 1 each time
- B alternates in pairs
- A alternates in fours

TABLE 1.7
Three-Variable Combinations

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

TABLE 1.8
Three-Term Truth Table

A	B	C	$(A + B)$	$(A + C)$	$(A + B) \cdot (A + C)$	$B \cdot C$	$A + B \cdot C$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

This table begins as any truth table for a three-variable Boolean equation. For example, to test the validity of Eq. (1.10*b*), begin with all possible combinations of three variables. Then combine variables from each side of the equation. Finally, compare both sides of the equation. Table 1.8 demonstrates this situation. Similarly, combining terms of other three-variable equations such as Eqs. (1.9*a*), (1.9*b*), and (1.10*a*), tests their validity.

The equations in this section are the theorems of Boolean algebra and are shown in Table 1.9. Boolean theorems are used to write Boolean equations. Such equations are the basis of computer circuit design.

1.2 WRITING BOOLEAN EQUATIONS

A systematic procedure exists for designing computer circuits. The first step is problem definition. In other words, the start is a description of circuit requirements: The circuit must . . ., but the circuit must not . . . . After circuit requirements are specified, the specifications are translated into a truth table, and the

TABLE 1.9
Boolean Theorems

	(*a*)	(*b*)
(1.1)	$A = 1$	$A = 0$
(1.2)	$A = 1 \rightarrow \overline{A} = 0$	$A = 0 \rightarrow \overline{A} = 1$
(1.3)	$\overline{\overline{A}} = A$	$\overline{\overline{A}} = \overline{A}$
(1.4)	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
(1.5)	$A + 0 = A$	$A \cdot 0 = 0$
(1.6)	$A + A = A$	$A \cdot A = A$
(1.7)	$A + 1 = 1$	$A \cdot 1 = A$
(1.8)	$A + B = B + A$	$A \cdot B = B \cdot A$
(1.9)	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
(1.10)	$A \cdot (B + C) = A \cdot B + A \cdot C$	$(A + B) \cdot (A + C) = A + B \cdot C$

TABLE 1.10
Darkroom Truth Table

L	A
0	0
1	1

truth table is converted into a Boolean equation. Finally, the circuit which performs the Boolean equation is constructed.

We are not yet in a position to design computer circuits, but we can relate Boolean algebra to everyday occurrences. Writing Boolean equations for ordinary situations simplifies writing Boolean equations for computer circuits.

Consider a photographic darkroom. There is a warning lamp above the darkroom door. When the lamp is on the darkroom is in use, and the door must remain closed. But when the lamp is off, the darkroom is not in use, and it is safe to open the darkroom door. These conditions define the problem. Only two conditions are possible: the lamp is either on or off. Boolean existence and nonexistence represent the lamp conditions. The number 1 means the lamp (L) is on, and the number 0 means the lamp is off. Table 1.10 shows the truth table for the alarm (A) that indicates the lamp condition.

Next the Boolean equations represented by this truth table are written by combining all conditions which result in a 1. In this case there is only one condition

$$L = A$$

Similarly, the inverse Boolean equation combines all conditions which result in a 0. In this case

$$\bar{L} = \bar{A}$$

Boolean equations with one variable are important, but applications are limited. Much more is accomplished when equations contain two or more variables.

Example 1.1 Andy A and Bruce B both like the movies, but they don't like each other. As a result Andy will not go to the movies M if Bruce goes. Feelings are mutual, and Bruce will not go if Andy goes. Show Boolean equations for: (a) somebody going to the movies; and (b) nobody going to the movies.

Solution The two variables are Andy and Bruce. There are four possible combinations:

- Neither Andy nor Bruce goes
- Andy does not go but Bruce does
- Andy goes but Bruce does not
- Both Andy and Bruce go

TABLE 1.11
Movies Truth Table

A	B	M
0	0	0
0	1	1
1	0	1
1	1	0

We represent going to the movies by a 1 and not going by a 0. Of the four possibilities, only two result in a moviegoer. The truth table for this Andy-Bruce-movie situation is shown in Table 1.11.

(*a*) The possibilities for somebody going to the movies have a 1 in the *M* column. These are

NOT Andy AND Bruce

which in Boolean algebra is $\bar{A} \cdot B$

OR Andy AND NOT Bruce

which is $+ A \cdot \bar{B}$.

The Boolean equation for somebody going to the movies is the ORing of conditions for which someone goes to the movies

$$M = \bar{A} \cdot B + A \cdot \bar{B}$$

(*b*) The possibilities for nobody going to the movies have a 0 in the *M* column. These are

NOT Andy AND NOT Bruce

which is $\bar{A} \cdot \bar{B}$.

OR Andy AND Bruce

which is $+ A \cdot B$.

The Boolean equation for nobody going is

$$\bar{M} = \bar{A} \cdot \bar{B} + A \cdot B$$

This example is a trivial case of two variables. Whether or not anyone goes to the movies is unimportant. However, a circuit with the same truth table is the basic addition circuit in real computers.

Situations involving Andy and Bruce can be more involved. A three-variable problem delves further into Boolean equations.

Example 1.2 Andy and Bruce both want to take Cindy to the movies. Andy is shy and will only ask Cindy if Bruce is not present. However, Bruce is not shy; he will ask