THE ESSENCE OF LOGIC CIRCUITS

STEPHEN H. UNGER

1-112

THE ESSENCE OF LOGIC CIRCUITS

STEPHEN H. UNGER

Columbia University

PRENTICE HALL, Englewood Cliffs, New Jersey, 07632

Preface

As suggested by its title, this book is about what I consider to be the essentials of logic circuits. I have tried to present the material in a clear, concise form, striking a balance between theory and practice. Too much concern with theory makes a subject such as this unnecessarily difficult to learn and detracts from its usefulness to those who are going to put it to work. At the other extreme, presenting "useful" procedures without justifying them, and concentrating on details of current practice would also make for dull reading, and, what is worse, would leave readers stranded when advances in technology make the material presented obsolete. I have relied heavily on the use of examples to clarify the general theory, and have tried to make these examples do double duty by using them to introduce useful devices.

Logic circuit design has gone through some interesting evolutions in the past few decades. Initially, logic gates were constructed from discrete components, such as vacuum tubes, diodes, transistors, resistors, etc. Each gate and the associated wiring entailed a cost in components, labor, bulk, and power requirements. A major design objective was to use the individual gates most efficiently. Next came the era of small and medium scale integration (SSI and MSI), in which the building blocks became sets of gates on a chip, or small functional units such as decoders. Now it became important to learn how to use these units efficiently. There was less value attached to minimizing logic expressions. This was also true when such devices as ROM's and PLA's became available off the shelf. Either a logic function could be realized on a certain size PLA or it couldn't. Except in marginal cases, there was not much point working to minimize logic. But with the advent of large and very large scale integration (LSI and VLSI) the situation has, to a surprising extent reverted to the first stage. Now substantial systems and subsystems are realized on a single chip. Reductions in the number of elements required to realize each function may make it possible to use fewer chips or to do more on a given chip. In high performance systems, eliminating redundant gates not only frees up chip area, it reduces power dissipation, often the limiting factor. The size of a PLA may be specified by the designer, and there are even special tricks for reducing the size of a personalized PLA. There are of course factors that were not present in the days of discrete logic elements, such as the value of regularity in the arrangement of elements on a chip. But the need for powerful methods for generating efficient logic circuits has indeed returned. A major consequence of the larger scale of integration is the enormous size and complexity of our systems, and the great importance of testing. An effort has been made in this book to focus on those problems that seem to be of greater importance for the technologies that are here now and coming in the foreseeable future. While there remains a place for SSI in patching together larger chips, it must be recognized that the emphasis is now on larger scale integration.

The book is a suitable text for an introductory course in logic circuits for undergraduate students in computer science, computer engineering or electrical engineering. It should also be of interest to practicing engineers and logic designers interested in learning about the subject or updating their knowledge. There are no prerequisites except perhaps for some sort of beginning course in programming.

Although there are only a few topics covered here for which some knowledge of electrical circuits and electronics is needed, I believe that it is helpful in acquiring a deeper appreciation of some important issues, such as wired logic, tri-state devices, and tradeoffs between power and speed. Since most computer science students lack this background, a simple presentation of the essentials has been added in an appendix.

Another relevant area, sometimes covered in other courses, pertains to manipulations of numbers in different number bases and some related matters. Since these are not central to logic circuits, they have also been relegated to an appendix.

Pointers to sources have been confined to a section at the end of each chapter, so as to avoid cluttering up the text. These all refer to a unified list of references at the end of the book. There is a summary at the end of each chapter, and a substantial number of problems dealing with virtually every topic.

Although it is possible to cover all of the topics in the book in a 1-semester course, most instructors would probably prefer to omit, or just skim lightly over, some of the more advanced topics. Since it was not possible to include detailed treatments of all of the many interesting aspects of this field, I chose to do a rather thorough job of explaining the fundamentals, particularly the subject of combinational logic, treating other topics in varying varying degrees of detail. In some cases, e.g. asynchronous sequential circuits, I included complete treatments of the basics, selectively omitting topics deemed less important relative to their complexity. In other cases, I presented some key points in detail, giving only overviews of the topics as a whole. This was done, for example, with respect to the vast subject of fault detection.

A number of topics included here generally receive little or no attention in introductory texts. These include iterative circuits, particularly the discussion of iterative circuits in tree form (culminating in a derivation of the full carry lookahead adder). The material on clocking schemes is new. Metastability has become an increasingly important problem in recent years; it is important that it be brought to the attention of logic designers at an early stage. The use of computers to assist logic designers, both in the design and simulation stages, is another subject introduced here that is not usually presented to beginning students. While many texts include some mention of duality, it is not usually given the attention it merits as a valuable conceptual tool that can sometimes be a real worksaver.

In the introductory chapter, the basic elements of digital logic are introduced with discussions of switching circuits, gate circuits and pass networks, and how they are related. The second chapter is about Boolean algebra and its correspondence to logic circuits. The simplification of expressions with the aid of the theorems is illustrated. Duality is explained and applied to the relations between positive and negative logic, relations between NAND- and NOR-gates, and to circuits with XOR-gates.

In Chapter 3, Karnaugh maps are introduced both as n-dimensional cubes and as Venn diagrams, and a rigorous procedure for using them to find minimal sum-of -products expressions is explained and illustrated. These ideas are extended to encompass variable-entered K-maps, a very useful concept. Then the Quine-McCluskey procedure for handling the same problem is presented. Next, these methods are extended to multi-output circuits. Some approaches to multi-stage logic are outlined. The special properties of unate (monotone) Boolean functions are pointed out. The treatment of the very important problem of finding diagnostic tests for combinational logic circuits is based on the idea of path sensitization.

In the first part of Chapter 4, the most useful combinational logic macros, decoders, MUX's, and encoders, are presented. This is followed by a section on how to realize irregular logic expressions with quasi-regular arrays; i.e. ROM's, PAL's, PLA's, and gate arrays. The incorporation of PLA's on LSI and VLSI chips is stressed; material on the use of decoders and folding is included. The handling of functions with inherent regularity is treated in Chapter 5. The flow table is introduced to describe iterative functions, and the analogy between spatial and temporal sequences in pointed out. After showing how to realize symmetric and iterative functions with linear iterative circuits, it is demonstrated that such functions can also be implemented with regular circuits in tree form, where complexity grows linearly with the number of inputs, while delay increases only logarithmically. Important examples are parity circuits, comparators, and the binary adder.

Sequential circuits are treated in Chapter 6. Since concepts involving synchronous and asynchronous circuits are intertwined, the discussion in the chapter alternates between them, beginning with an introduction to simple clocked circuits. Flow tables and state diagrams are presented along with procedures for merging equivalent states, and going from a flow table to a circuit. This is followed by a discussion of the basic elements of asynchronous sequential circuit theory including types of functions, types of delays, flow table forms and manipulations, combinational hazards, sequential hazards, analysis, and the state assignment problem. No attempt is made to cover the last of these topics thoroughly. Rather, the problem of critical races is explained, state assignments of various types are illustrated with examples, and one simple general solution , the 1-hot code, is presented. The examples in this section generally involve useful devices such as the SR-FF, the latch, and the edge-triggered D-FF. In the following section, it is shown how the duality concept can be extended to sequential circuits. Next some examples are given showing how informal design procedures are often adequate in important practical cases, such as registers with various capabilities. Formal design procedures for synchronous systems are then described. These entail problem specifications in various forms ranging from informal verbal descriptions, to register transfer sequences in structured form, state graphs, and ASM charts. Examples include a multiplier controller and a controller for a cache interface unit. The generation of logic from these descriptions is illustrated, one approach being via a state map that leads to variable entry K-maps. Next, it is shown how to choose the parameters of clocking systems so as to maximize operating speed while ensuring proper operation. Where signals that may change at arbitrary times relative to one another must interact, there is a danger of indeterminate behavior arising through the mechanisms of meta-stability or runt pulses. A general method for identifying such situations is presented next, along with appropriate countermeasures. In the course of this discussion, arbiters and synchronizers are introduced. Chapter 6 continues with an introduction to speed independent circuits, beginning with the C-element and concluding with a design for a speed independent buffer that can be used independently or as a building block to construct speed independent networks. The last section deals with the problem of testing sequential systems. Scanning (e.g. LSSD) is described, and the elements of built in testing, using the signature concept are outlined.

Chapter 7 concerns computer aids for digital system designers. Several different approaches to using computers to generate minimal combinational logic are discussed, and illustrated, including the Logic Synthesis System used at IBM. The less developed field of aids to sequential logic design is only briefly scanned. Next, the very important area of simulation is introduced, starting with a brief discussions of high level and logic level simulations. The value of switch level simulators for MOS based technologies is illustrated with some examples motivating key features of the MOSSIM II simulator. The role of circuit level simulators is then outlined. In the last section, a computer algorithm is described for timing analysis, i.e. finding bounds on propagation delays through a large logic network.

The first part of the appendix is concerned with arithmetic operations in binary, and number conversions of both integers and fractions. Other topics are octal, hexadecimal, BCD, and ASCII. Gray codes are also introduced. On a higher level, is the discussion of error correcting codes and how to implement them with logic circuits. (Reading this should be postponed until after XOR gates are understoood.) Section 2 of the appendix is a short introduction to simple electrical circuits and electronics, beginning with a treatment of circuits with resistors, voltage sources, and switches. The voltage divider, a key configuration in digital elements, is given particular attention. Next comes a simple explanation of how an MOS transistor works. The section concludes with a discussion of capacitors and simple RC circuits. This is then related to digital circuitry, to show how rough estimates of propagation delays can be made. Section 3 outlines the characteristics of the principal logic families, nMOS, CMOS, TTL, ECL, and I^2L.

The book concludes with a brief postscript outlining what I feel are some important aspects of the applied science and engineering professions that all too often "fall between the cracks."

Contents

PREFACE xi

1 INTRODUCTION 1

1.1 What Is Logic Design? 1

1.2 Building Blocks for Logic Circuits 2

1.2.1 Switching Circuits, 2
1.2.2 Logic Gates, 4

1.3 Direct Connections of Gate Outputs 6

1.3.1 Wired (or Implied) Logic Gates, 6
1.3.2 Tri-State Outputs, 8

1.4 Examples of Logic Circuits 9

1.5 Overview and Summary 11

Sources 11

Problems 12

2 BOOLEAN ALGEBRA APPLIED TO LOGIC CIRCUITS 14

2.1 Boolean Algebra — Definitions and Postulates 15

2.2 Boolean Expressions 16

2.3 Truth Tables and Theorems 16

2.4 Relating Logic Expressions to Logic Circuits 24

2.4.1 Gate-Type Circuits, 24
2.4.2 Switching Circuits, 26

2.5 Using Boolean Algebra to Manipulate Logic Circuits 29

2.5.1 Truth Tables and Minsums, 29
2.5.2 Goals of Logic Design, 30
2.5.3 Using Algebra to Simplify Logic, 31

2.6 NAND- and NOR-Gates 32

2.7 Duality of Functions 39

2.8 Positive and Negative Logic 40

2.9 Exclusive-OR-Gates 41

2.10 Multiple-Valued Logic 44

2.11 Overview and Summary 44

Sources 45

Problems 45

3 DESIGNING COMBINATIONAL LOGIC CIRCUITS ***49***

3.1 General Concepts 49

3.2 Prime Implicants 50

3.3 Karnaugh Maps and Their Use in Finding Minimal-Sum Expressions 52

3.3.1 Cubes in n-Dimensions, 53
3.3.2 Structure of Karnaugh Maps, 55
3.3.3 Identifying Subcubes on K-Maps, 59
3.3.4 Using K-Maps to Find Minimal SOP Expressions, 61
3.3.5 Variable Entered K-Maps, 72

3.4 Quine-McCluskey Method for Finding Minimal SOP Expressions 75

3.4.1 Finding Prime Implicants, 75
3.4.2 Finding Minimal Covers, 78

3.5 Simplification of Multi-Output Logic 81

3.5.1 Using K-Maps, 82
3.5.2 Using the Quine-McCluskey Method, 85

3.6 Unate Functions: An Interesting Special Case 88

3.7 Multi-Stage Logic, Fanin, Fanout 91

3.8 Fault Detection in Combinational Logic 93

3.9 Overview and Summary 98

Sources 99

Problems 99

4 *COMBINATIONAL LOGIC CIRCUITS IN REGULAR FORMS* ***108***

4.1 Important Logic Macros 108

4.1.1 Decoders, 109
4.1.2 Multiplexers, 112
4.1.3 Encoders, 114

4.2 Implementing Irregular Logic with Regular Arrays 115

4.2.1 ROM, 116
4.2.2 PAL, 117
4.2.3 PLA, 119
4.2.4 Gate Arrays, 126

4.3 Overview and Summary 127

Sources 127

Problems 128

5 *SYMMETRIC AND ITERATIVE CIRCUITS* ***130***

5.1 Symmetric Functions 130

5.1.1 Definition, Notation, and Examples, 130
5.1.2 Operations on Symmetric Functions, 131
5.1.3 Implementing Symmetric Functions with the Tally Circuit, 132

5.2 Iterative Functions 133

5.2.1 Initial Definition and Some Examples, 133
5.2.2 Flow Table Representations, 135
5.2.3 Synthesis of Linear Iterative Circuits, 140

5.3 Realizations in Tree Form 144

5.3.1 Introduction to the Tree Form, 144
5.3.2 State Mappings and Their Products, 145
5.3.3 Specifications of Tree Circuit Cells, 147
5.3.4 How the Cells are Interconnected in the Tree Circuit, 148
5.3.5 Logic Design of the Tree Cells, 149
5.3.6 Some Refinements, Derivation of the Full-Carry Look-ahead Adder, 152

5.4 Overview and Summary 153

Sources 154

Problems 155

6 **SEQUENTIAL LOGIC CIRCUITS** 157

6.1 Synchronous Circuits with Ideal Unit Delays 157

6.1.1 Flow Table and State Diagram Descriptions of Sequential Functions, 158
6.1.2 Reducing Flow Tables, 160
6.1.3 Generating Logic Expressions from Flow Tables, 164

6.2 Asynchronous Circuits 166

6.2.1 Some Basic Concepts, 166
6 2 2 Primitive Flow Tables and Flow Table Reduction for Asynchronous Functions, 168
6.2.3 Types of Delays, 169
6.2.4 Generating Logic Expressions for Asynchronous Functions, 170
6.2.5 Critical Races and How to Avoid Them: The State Assignment Problem, 172
6.2.6 Combinational Hazards and What to Do about Them, 177
6.2.7 Essential Hazards and the Role of Delay Elements, 183
6.2.8 Analysis of Asynchronous Sequential Circuits, 186

6.3 Duality for Sequential Circuits 188

6.4 Designing Sequential Circuits: An Informal Approach 190

6.4.1 The SR-FF and a Debouncing Circuit, 190
6.4.2 Using Latches to Design an Edge-Triggered FF, 191
6.4.3 Designing Registers, 193

6.5 Formal Design of Synchronous Circuits 197

6.5.1 Problem Specification with Structured Statement Sequences, ASM Charts and Flow Tables, 198
6.5.2 Generating the Logic, 206

6.6 Clocking Schemes 211

6.6.1 One -Phase Clocking with Edge-Triggered D-FFs, 212
6.6.2 One -Phase Clocking with Latches, 215
6.6.3 Two-Phase Clocking with Latches, 218

6.7 The Synchronization Problem 222

6.7.1 Runt Pulses, 222
6.7.2 The Unrealizability of Ideal Inertial Delays, 224
6.7.3 Metastable States, 224
6.7.4 The Arbiter, 226
6.7.5 The Synchronizer, 227

6.8 **Speed Independent Circuits** 229

6.8.1 The Concurrency- or C-Element, 230
6.8.2 A Speed Independent Buffer, 230

6.9 Fault Detection in Sequential Logic Circuits 234

6.9.1 The Scan Method, 235
6.9.2 Built in Test, 237

6.10 Overview and Summary 240

Sources 242

Problems 243

7 SOFTWARE TOOLS **249**

7.1 Logic Design 249

7.1.1 Combinational Logic, 250
7.1.2 Sequential Logic, 253

7.2 Simulation 254

7.2.1 High Level, 254
7.2.2 Logic Level, 255
7.2.3 Switch Level, 256
7.2.4 Circuit Level, 258

7.3 Timing Analysis 259

7.4 Overview and Summary 261

Sources 262

Problems 262

APPENDIX **264**

A.1 Number Systems and Codes 264

A.1.1 Number Systems, 264
A.1.2 Arithmetic in Binary, 265
A.1.3 Integer Conversions to Base-10, 266
A.1.4 Conversions from Base-10, 266
A.1.5 Fractions, 267
A.1.6 Conversions between Base-b and Base-bk , 268
A.1.7 Binary, Octal, and Hexadecimal, 268
A.1.8 Binary Coded Decimal, 269
A.1.9 ASCII Code, 269
A.1.10 Gray Code, 270
A.1.11 Error Correcting Codes, 270

A.2 Simple Electrical Circuits 273

A.2.1 Resistors and Voltage Sources, 273

A.2.2 How an MOS Transistor Works, 277
A.2.3 Capacitors and Time Constants, 280

A.3 Principal Logic Families 284

A.3.1 MOS Based Technologies, 284
A.3.2 Bipolar Families, 285

A.4 Summary 286

Sources 286

Problems 287

A POSTSCRIPT ON PROFESSIONALISM ***290***

REFERENCES ***292***

INDEX ***297***

Chapter 1

Introduction

1.1 WHAT IS LOGIC DESIGN?

Logic circuits are what make things happen in digital computers. They govern the input of information, the fetching and execution of instructions, the conversion of data from one form to another, and the output of results. Logic design is about interconnecting large numbers of simple elements to make them perform intricate operations. The elements themselves are an amazing culmination of a vast array of engineering and scientific knowledge. But, using them entails a rather different kind of knowledge.

Logic designers do not employ the classical techniques of electrical engineering or physics or chemistry. Nor are they much concerned with the kind of mathematics, largely based on differential equations, that underpins much of these disciplines. Corresponding to the simple physical elements used in logic design is a mathematics based on 1s and 0s rather than on real numbers. While some of this mathematics, such as Boolean algebra, is well developed, much of it is highly irregular, with bits and pieces invented as needed. Those who enjoy solving puzzles are likely to feel at home in this field.

Logic circuits and many sophisticated techniques for designing them antedate the digital computer by many years. The initial applications were to the design of telephone central office equipment. The key concept, which transformed the design process from an art to a science, was the idea of describing both the functions performed and the circuits themselves in terms of Boolean algebra. This was embodied in Claude Shannon's master's thesis in electrical engineering at MIT, one of the most brilliant master's degree theses ever written. It was published in 1938. A substantial body of knowledge has been generated since then, making logic design perhaps the most highly developed area of computer science. Yet many important and interesting problems remain to be solved.

There are two basic classes of logic circuits. *Combinational* logic circuits are those whose present outputs are functions only of the present inputs. *Sequential* logic circuits are those whose outputs depend on *past* as well as present inputs. The examples in this chapter all illustrate combinational logic; not until Chapter 6 is the more complex subject of sequential logic introduced.

A major use of logic circuits is to perform arithmetic operations on numbers represented in terms of two-valued, or binary, signals. Circuits that count, add, and multiply in binary are basic applications of logic circuit techniques and are used in subsequent chapters to illustrate these techniques. Hence, logic designers should have a good grasp of how numbers are represented and manipulated in binary. They should also understand how nonnumeric information can be represented in terms of two-valued signals, as in ASCII. An introduction to this subject is in Appendix A.1.

An understanding of simple electrical circuits with resistors and batteries is sufficient to master the material in this book pertaining to logic circuit components. A little further acquaintance with circuits and electronics is very helpful in enhancing one's appreciation of certain important issues in logic design. Strangers to these subjects are cordially invited to "break the ice" by perusing the introductory material in Appendix A.2.

The basic elements from which logic circuits are constructed are introduced in the following section. Some simple, but nontrivial, examples are then used to illustrate how these devices can be put to work.

1.2 BUILDING BLOCKS FOR LOGIC CIRCUITS

There are two principal types of logic circuits: those made of switches and those made of gates. These are related in that one can think of gates as being constructed from switching circuits. In some systems, both types are used.

1.2.1 Switching Circuits

It will be assumed here that when a switch is open, no current can pass through it (it has infinite resistance), and that when it is closed, current can pass through it without any voltage being developed across it (it has 0 resistance). Although real switches can only approximate such behavior, the approximations are adequate in most situations. Associated with each switch is a binary valued control signal, usually represented by a voltage value. The switch is open or closed depending on whether the value of this signal is 0 or 1, respectively.

Such switches might be implemented as contacts on an electromechanical relay, in which case the control signal corresponds to the voltage across the coil or winding (the value of the control signal being 1 or 0, respectively, depending on whether this voltage is or is not sufficient to activate the relay). A modern implementation would be the region between source and drain of a MOSFET transistor, with the control signal corresponding to the voltage on the gate. While a MOSFET switch acts very much like an open circuit when the gate voltage is low, its resistance when the gate voltage is high is far from 0. This is a factor that must be taken seriously in electrical aspects of the design process. Those not acquainted with the terms and concepts just introduced can find more information in Appendix A.2 (or in some other source), or else they can accept on faith the idea that devices behaving as switches really do exist.

Our concern here is with two-terminal circuits as shown in Figure 1.1a. Between the two terminals is a circuit composed of switches. Some simple examples are shown in parts b, c, and d of Figure 1.1. The issue of interest concerning a switching circuit is whether or not there is a path through it. This, of course, depends on the structure of the circuit *and* on the states of the variables controlling the switches in it. T, the *transmission* of a switching circuit is defined as a variable with value 1 if there is a path and 0 if there is no path. In the very simple case of a single switch (Figure 1.1b), T = 0 if A = 0 and T = 1 if A = 1, i.e., T = A. For circuit (c), T = 1 if and only if A and B are both 1. (We often refer to a switch control variable or a circuit transmission that is equal to 1 or 0 as being *on* or *off*, respectively.) For circuit (d), T = 1 if A or B is on (or both— the word *or*, here, is being used in the *inclusive* sense). For circuit (e) T = 1 if C is on, or if both A and B are on.

Transmission variables are often converted to voltages, which can then be used as switch control variables. Two ways to do this are shown in Figure 1.2. In Figure 1.2a, if T = 0 for switching circuit S, the output voltage Z will also be 0 (we always assume that the voltage level of the ground connection is 0). Otherwise, if T = 1, Z will be pulled *up* to the value of the supply voltage V (which we assume is high enough to qualify for on status), and is therefore 1. We refer to a switching circuit in such a position (connecting the output to the supply voltage) as a *pullup* circuit. The Figure 1.2b circuit works in the opposite direction, with S connecting the output to the ground connection thus pulling *down* the output when T = 1 or allowing the output to rise when T = 0. Hence, this is referred to as a *pulldown* circuit.

Switching networks can also be used to route or *pass* logic signals from one terminal to another. Several such configurations are illustrated in Figure 1.3. In Figure 1.3a, the signal A is transmitted to Z if the transmission of the switching circuit S is 1; otherwise the Z-signal is pulled down to 0 through the resistor connected to ground. The Figure 1.3b configuration differs in that, if S is off, the value of Z is pulled up to 1 by the resistor connected to the supply voltage.

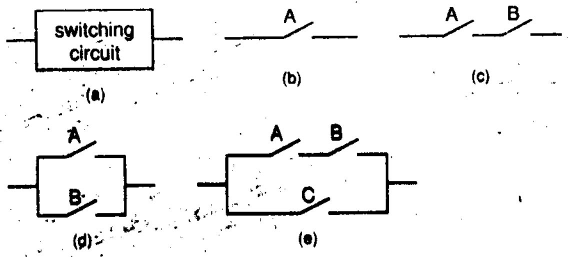

Figure 1.1 Block diagram of switching circuit and some examples.

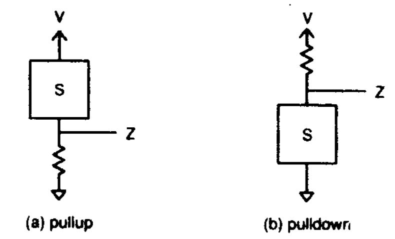

Figure 1.2 Switching circuits as pullups or pulldowns.

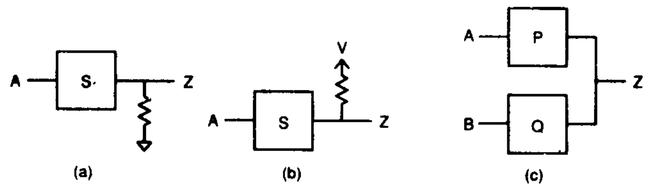

Figure 1.3 Switching circuits as pass networks.

A somewhat more complex situation is depicted in Figure 1.3c. Here, if only circuit P is on, then Z takes on the value of A. If only circuit Q is on, then the value of B is passed to Z. It is generally assumed for such arrangements that the signals controlling the transmissions of P and Q are so related that there will always be *exactly* one *signal* passed to Z. It is forbidden to have *both* P and Q on or *both* P and Q off, since that might leave Z undetermined. Clearly, this circuit could be generalized to include more than two switching circuits. Another variation would be to permit both P and Q to be simultaneously off, but to add either a grounded resistor (as in Figure 1.3a) or a resistor connected to the supply (as in Figure 1.3b) to determine the value of Z when neither switching circuit is on.

1.2.2 Logic Gates

The most common type of building block used in the design and analysis of logic circuits, and the one primarily used in this book, is the logic *gate*. These are devices with one or more inputs and one output. Both inputs and outputs are binary valued logic signals. A basic set of logic gates is shown in Figure 1.4.

Part a shows a gate, called an *inverter*, with one input. The output Z takes on the value opposite that of the input. Inverters are usually realized by circuits in the form of Figure 1.2b, in which the pulldown circuit consists of a single switch. controlled by the input to the inverter. The gate shown in part (b), called an *OR-gate*, has two inputs, and its output is 1 if A *or* B or both are equal to 1. (The word *or* is used here in what is called the *inclusive* sense , as opposed to the *exclusive* sense which rejects the case where *both* variables involved are 1.) An OR-gate could be (but seldom is) realized by a circuit in the form of Fig, 1.2a, in which the pullup circuit consists of two switches in parallel (as in Figure 1.1d), one controlled by each input. The third type