Design of
TRANSISTORIZED CIRCUITS
for
DIGITAL COMPUTERS

Abraham I. Pressman

Design of
TRANSISTORIZED CIRCUITS
for
DIGITAL COMPUTERS

ABRAHAM I. PRESSMAN, B.S., M.S.

Consulting Engineer

JOHN F. RIDER PUBLISHER, INC., NEW YORK

LIBRARY OF CONGRESS CATALOG CARD NUMBER 58-13426

Printed in the United States of America

PREFACE

This book is concerned with the design of transistorized digital computer, and digital type circuits. Written from the viewpoint of the circuits designer, it is intended both for those who have no prior knowledge of either transistor or computer type circuits, and those with a knowledge of computer type circuitry but who have not yet converted to transistors.

Present day digital computer designers rarely think of circuitizing their block diagrams with vacuum tubes. The choice now lies between transistors and magnetics. If transistors are chosen, a major decision as to how to mechanize the basic logical elements — the AND and OR gates — must be made, because how these elements are circuitized in a large measure fixes power dissipation, d-c voltage levels, power supply requirements, cost, and reliability and operating speed of the entire computer.

This book discusses detailed design calculations of nearly all the main methods of circuitizing these computer logical elements and other building blocks, with transistors. Emphasis has been on worst-case design techniques so that the basic circuits may operate with resistors, voltages, transistor parameters, diode forward drops, and reverse leakages, all simultaneously at those extremes of their tolerance limits that would most tend to make the circuits inoperative.

Methods of calculating transistor switching speeds, and signal rise and fall times, have been shown, with numerous examples. Factors that influence important decisions such as range of operating currents and d-c voltage levels, have been discussed.

Calculation and analysis of the various circuits has been done without requiring use of the four-terminal network equivalent circuits primarily helpful in dealing with small-signal linear amplifiers. Circuits of the nature discussed herein are mainly large-signal, on-off current switches, and it is shown how they may be designed from the static volt-ampere curves of the transistors, and some knowledge of their transient response characteristics. The behavior of the circuits can be predicted from relatively simple arithmetic calculations.

The first two chapters of the book deal with computer logic, logical building

blocks, and Boolean algebra, in sufficient detail that those without previous experience in computers may gain an appreciation of how the circuits designed in the latter part of the volume fit into the overall picture of a computer. Chapter 3 presents a brief discussion of transistor physics and transistor fundamentals.

It is hoped that such a book, emphasizing worst-case design techniques, switching circuits, and switching time calculations, will be found useful in this increasingly important field.

For the technical review of the manuscript, and their many helpful suggestions, my appreciation is particularly due to Dr. W. T. Chow, Mr. J. Tellerman and Mr. E. Keonjian of the Arma Corporation, New York.

ABRAHAM I. PRESSMAN, M. S.

Philadelphia, Pa.,
March 1959.

CONTENTS

Chapter 1

BASIC BUILDING BLOCKS IN DIGITAL COMPUTERS

Digital computer circuits are built by the interconnection of a relatively small number of basic building blocks. However complex a digital computer or control system may be, except for some small specialized portions, it can always be designed as a unique assembly of these fundamental blocks.

The actual fabrication of a digital system usually occurs in one of the following ways. A basic building block or multiples of these blocks in various combinations may be packaged on a single plug-in chassis. All the input and output points of the block are brought out to standard pins on the plug-in chassis. By varying the interconnections between input and output points any digital circuit may be realized. This has many advantages. Only standard packages are built, which simplifies the production problem. It also simplifies the trouble-shooting problem as suspected failures can be verified by merely replacing one standard package with another. Also, the same set of packages can easily be used to mechanize any number of completely different digital circuits.

Still another scheme uses a building block concept but does not package the block or combinations thereof in standard packages. It generally employs a standard plug-in chassis, but does not use a standard array of circuits. With the building block designed and its drive requirements and capabilities known, a nonstandard array of blocks is assembled to fill each plug-in chassis. Thus no two chassis are alike. This arrangement has advantages in that there are fewer redundant components, and circuits electrically related are usually close to one another, a desirable condition that eases the ever present computer problem of driving-wiring capacity between building blocks. Such a scheme is usually more effective in smaller machines.

Whatever the plan used, however, the circuit designer's problem is the same: to design building blocks that perform the required logical operations with the

necessary speed, and calculate their input drive requirements and output load driving capability. In this chapter, the properties of such building blocks and their symbolic representation are discussed. They are treated here only as "black-boxes" that have certain operations to perform, without any consideration as to their internal construction. Later chapters go into the matter of detailed design and perform worst-case calculations to enable them to perform their operations when all components and voltages are at the extremes of their tolerance limits.

1.1 Digital Signals and Their Significance. Digital computer or control circuits are basically information processing devices. The information they pro-

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Fig. 1-1. Eight possible combinations of a 3-bit code.

cess is in the form of a time or space sequence of signals. These signals may have either of two values having the significance of a "one" or a "zero." In writing a one or zero, two different marks of any sort, such as 1 or 0 may be used. These one or zero signals may be represented physically in various ways. Thus a one may be a positive voltage level, and a zero a negative voltage level. The levels need not be positive or negative with respect to a common ground point, only with respect to one another. A one may be represented by the presence of a voltage pulse, zero by the absence of a pulse at a specific time. A vacuum tube or thyratron may be in the "on" or "off" state, a relay may be energized or de-energized to represent the two valued signal. The presence or absence of a hole punched in card or paper tape may represent one or zero. A magnetic toroid may be magnetized in a clockwise or counterclockwise direction; the surface of a drum or tape coated with magnetic material may be magnetized in one or the other direction for ones or zeros, or ones may be represented by magnetic marks and zeros by the absence of magnetic marks.

Whatever the physical process involved, each such one or zero signal is commonly referred to as a "bit." A coded time or space sequence of such bits can be used to represent alphabetic or numerical characters. Because each bit can have the value of either one or zero, a code with n bits can represent 2^n different characters. A code where each character is represented by the position of bits

in a three-column array is shown in Fig. 1-1. There are only eight possible combinations in such a 3-bit code.

To represent the 10 unique decimal digits, 1 through 9 and 0, would require a 4-bit code. To represent the decimal digits, alphabetic characters and various punctuation marks available on a standard typewriter keyboard — 56 characters — would require a 6-bit code. The number system in everyday use is based on the number 10. In this system there are 10 discrete possible marks, 1 through 9 and 0. Any number is represented by the sum of powers of 10, each power of 10 being preceded by one of these marks signifying the number of times that power of 10 is to be taken into the sum. Thus the number 1234 has the meaning $(1 \times 10^{+3}) + (2 \times 10^{+2}) + (3 \times 10^{+1}) + (4 \times 10^{0})$.

To design machines performing high-speed numerical computation directly in the decimal system would be extremely difficult in that this would require a device capable of changing rapidly to any of 10 different discrete states representing the 10 basic numbers. There are however, as already mentioned, any number of ways in which a device can change rapidly to either of two possible physical states. Consequently a number system based on powers of two rather than 10 is used in high-speed numerical machine computation.

In such a binary system there are only two permissible marks, 1 or 0. Numbers are represented by the sum of powers of two, each power of two being preceded by a 1 or a 0 coefficient signifying the number of times that power is to be taken into the sum. Numbers are written in the binary system merely as these coefficients set side by side, with the extreme right-hand coefficient usually representing the lowest power of 2, 2^0 or 1. Thus the number 13 would be written in the binary system as 1101 signifying $(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$.

The operations of addition, multiplication, subtraction and division may be performed directly in the binary system. Some of these operations will be considered later.

In addition to representing an alphabetic or numerical character, a binary signal may represent a control condition or command. Thus a voltage level being in either its one or zero state may have the significance of "add" or "do not add", "read" or "do not read", "commence an operation" or "do not commence an operation", a "given condition is true" or a "given condition is not true". Any one of an infinite variety of conditions that can be sensed for or signalled by an electrical or mechanical device may be represented by such a one or zero signal.

The manipulation, routing and general processing of such binary data representing alphabetic or numerical characters under the control of binary signal commands or conditions, is performed essentially by six basic logical elements. These are the flip-flop, binary counter, AND gate, OR gate, time delay unit, and signal inverter. In addition, there are units — such as power amplifiers and transducers of various kinds to convert mechanical into electrical signals — that are essential, but which do not perform logical operations. There are also memory devices that have the capacity to store large numbers of binary bits, but they will not be discussed in this book.

1.2. Flip-Flops. The flip-flop performs the logical operation of remembering. It has two possible states, the one and zero state. It can be flipped or flopped from one state to the other by short duration impulses, and remembers indefinitely the last state into which it has been thrown. The usual symbol for the flip-flop is shown in Fig. 1-2.

Input signals are usually short duration impulses shown in the symbol by arrows directed *into* the side of the flip-flop. Output signals are indicated in the symbol by arrows coming *out of* the top of the rectangle from the same side as the corresponding input signal. Inherent in the logic symbol is the logical sense

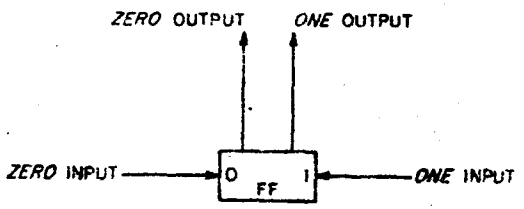

Fig. 1-2. Logical symbols for a flip-flop.

of "enable" or "inhibit", "true" or "false." That is, a signal last delivered to the zero input side will set the flip-flop so that the zero side generates an enable or true signal, and the one side generates an inhibit or false signal. This state is maintained indefinitely and is referred to as the zero state of the flip-flop.

A second impulse of the same nature delivered to the zero side of the flip-flop does not change its state, but an impulse delivered to the one input side reverses

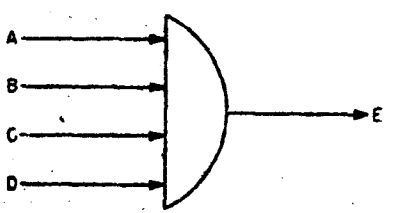

Fig. 1-3. Logical symbol for an AND gate.

the state of the flip-flop so that its one output terminal generates an enable or true signal, and its zero side an inhibit or false signal. The flip-flop is now in the one state and remains there until an impulse is next delivered to the zero input terminal. Enable and inhibit signals have meaning only with respect to the AND and OR gates driven by the flip-flop. These signals are usually voltages of either of two levels. The gates driven from the flip-flop can be designed so that either the upper or lower voltage level constitutes a true signal.

The usual flip-flop — that dealt with here — contains two output signals at either of two d-c voltage levels. But in the logical sense a flip-flop is any device

that can, upon receipt of a triggering pulse, assume either of two stable states, and will remain in that state until triggered to the opposite state. A flip-flop can be a thyratron with only one output terminal, its two states being "fired" and "not fired." Or it may be a square hysteresis-loop magnetic core magnetized to the state of either positive or negative remanence. In such a case the state in

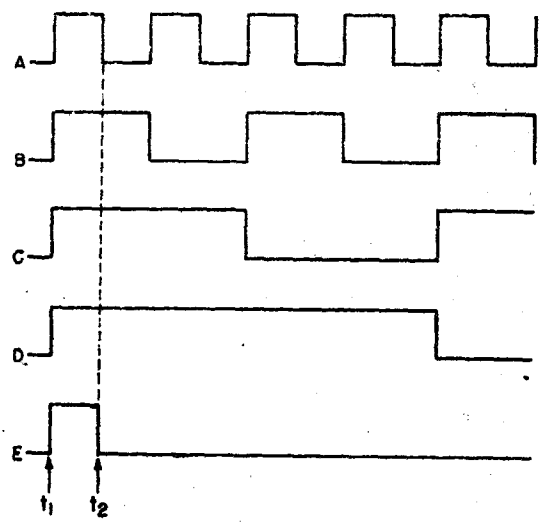

Fig. 1-4. *A possible sequence of signals at AND gate inputs (A to D), and the resultant output at E.*

which it resides is only known by attempting to set it back to a standard state.

The flip-flop discussed in this book contains two transistors of which one is turned off, generating one voltage level, and the other is switched on, generating the other voltage level.

1.3. AND Gates. A logical AND gate is a device with a multiplicity of input terminals and a single output terminal. The signals on its input terminals are the true or false, enable or inhibit, one or zero binary signals discussed above. The AND gate performs the logical operation of emitting a true output signal only when all its input signals are simultaneously true. A false or inhibit signal on one or more of its input terminals produces a false or inhibit signal at its output. The logical symbol for the AND gate is shown in Fig. 1-3. Inputs are shown as arrows directed into the semicircle, and the output as the arrow directed out of it.

Figure 1-4 illustrates a typical sequence of signals at an AND gate for true upper-level and false lower-level signals. Only from t_1 to t_2 are all four inputs at the enabling level giving a true out for that duration at *E*.

Sometimes a logical AND operation must be inhibited under a given condition. For example, an AND gate output may be desired if conditions A and B are true, but if C is simultaneously true, a true output should be inhibited. This is

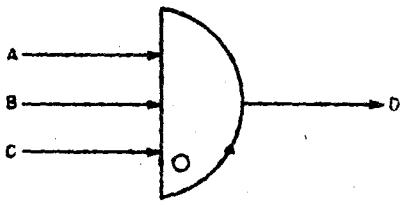

Fig. 1-5. Logical symbol for an AND gate with an inhibiting input at C.

shown symbolically by Fig. 1-5. Here the circle inside the AND gate indicates that a true signal at C is to inhibit or prevent an output at D, whatever the conditions at A and B. A typical set of waveforms in such a gate would be as in Fig. 1-6 for true upper-level, and false lower-level waveforms.

The lack of an inhibiting signal from t_1 to t_2 permits the gating of A and B to give ouputs for every enabling input at A because B is continuously at the

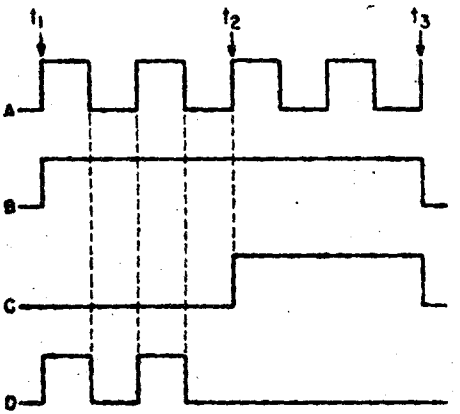

Fig. 1-6. Possible input and resulting output signals for the AND gate of Fig. 1-5.

enable-level for this interval. From t_2 to t_3 the presence of a signal at C inhibits the gate and prevents any true output signal, even though A and B are at times both true in this interval.

1.4. OR Gates. The OR gate is a device with a multiplicity of inputs and one output. The inputs can each assume either of two possible values, one designated true, the other false. The gate performs the logical operations of producing a true output for the time duration that one or more of its inputs is true, and a false output for the time durations in which none of its inputs are true.

Its logical symbol is shown in Fig. 1-7 where the inputs are shown directed into the triangle, and the resultant signal at D is directed outwards. A typical example for true upper-level and false lower-level signals is shown in Fig. 1-8.

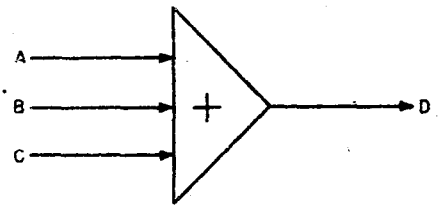

Fig. 1-7. Logical symbol for an OR gate.

1.5. Binary Counter. The binary counter, like the flip-flop is a device with two stable states and having the property of remaining indefinitely in the state into which it was last thrown. By the repeated application of a short-duration input pulse to one of its input terminals it is alternately thrown from one state to the other. Its logical symbol is shown in Fig. 1-9.

In operation too, it resembles the flip-flop, with the added facility of being able to reverse its state, whether one or zero, by the application of a pulse at

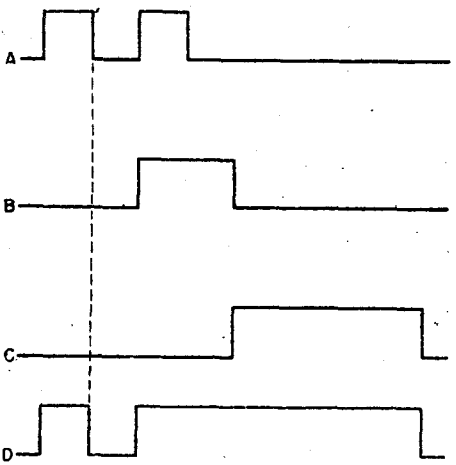

Fig. 1-8. Possible input and resulting output signal for the OR gate of Fig. 1-7.

the binary counter input terminal. As well as its binary count input, it generally has the normal flip-flop terminals that permit it to be set to the one or zero state by pulses at these points. And like the flip-flop, if it is in the zero state, a pulse at the one input terminal will set it to the one state, but if it is already in the one state a pulse at that terminal will have no effect.

A simple example of the way in which building blocks are combined is given in Fig. 1-10, which shows how a binary counter may be built from a flip-flop.

In this illustration a standard flip-flop is feeding two AND gates for steering purposes. One of the input terminals of G1 is fed from the zero output side of the flip-flop, and the output of AND gate G1 feeds the one input side. Similarly, one of the input terminals of G2 is fed from the one output side, and the output of the AND gate G2 feeds the zero input terminal. "Count" pulses are

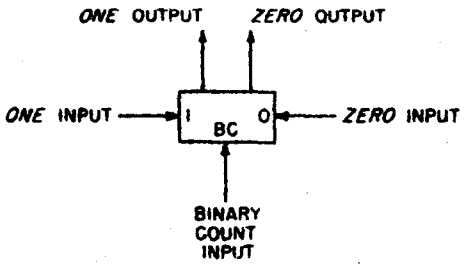

Fig. 1-9. Logical symbol for a binary counter.

simultaneously applied to the second input terminals of gates G1 and G2. This connection automatically steers the count pulse into that input terminal that will change its state. A change of state occurs at each pulse applied to the count terminal, because if the flip-flop is initially in the zero state, G1 will be enabled and G2 will be inhibited. The first count pulse coming along cannot pass

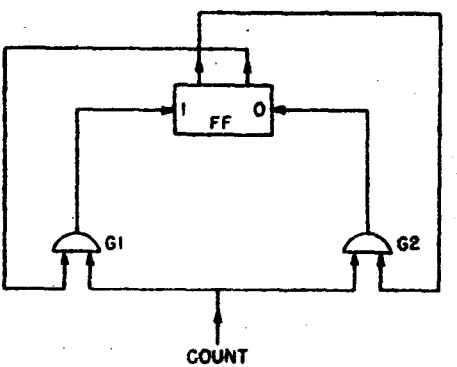

Fig. 1-10. Logical interconnections for building a binary counter from a flip-flop and two AND gates.

through G2 but is routed through G1 to set the flip-flop to the one state. Now G2 is enabled and G1 is inhibited. The next count pulse is therefore routed through G2 and sets the flip-flop back to the zero state.

Binary counters are frequently made up in this manner with the AND gates recognizable as the normal AND gates in use wherever an AND operation is required. In some counters however, such steering AND gates are not immediately recognizable as such. They may be built-in with resistors and capacitors

or with a pair of diodes to achieve the pulse steering properties in the simplest possible way.

1.6. Signal Inverters. The signal inverter as its name implies, merely reverses the sense of a signal. If its input is a one or true signal, its output is a zero or false signal, and vice versa. Normally built with an active element

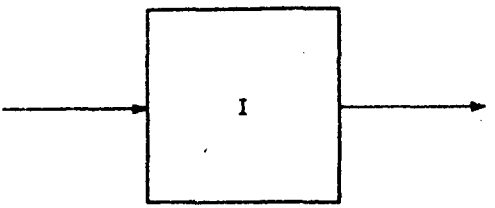

Fig. 1-11. Symbol for a logical signal inverter.

such as a tube or transistor, it gives power gain in addition to signal reversal. Its logical symbol is shown in Fig. 1-11.

1.7. Time-Delay Element. Time delays in computer circuits are produced either by a passive network of inductances and capacitors for small delay times, or by an active element such as a tube or transistor in combination with

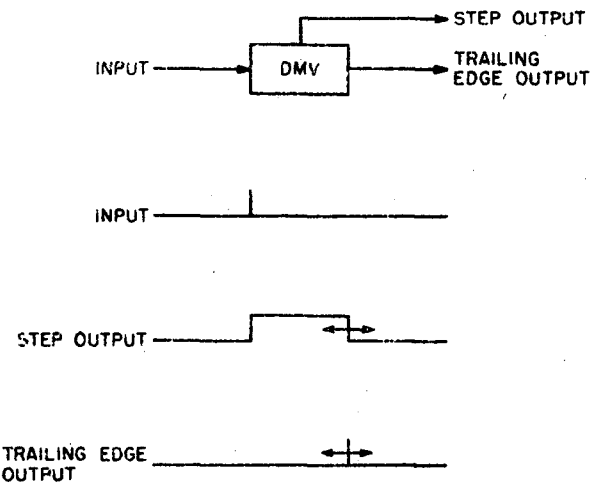

Fig. 1-12. Logical symbol for a delay multivibrator.

an R-C network for longer delays. Passive-network delay lines are designed as low-pass filters[1], and are usually suitable for relatively short delays up to about 5 microseconds.

[1] *Pulse Techniques*, Moskowitz, S., and Racker, J., pp. 100-119. New York: Prentice-Hall, Inc., 1951.
Pulse and Digital Circuits, Millman, J., and Taub, H., pp. 286-322. New York: Mc Graw-Hill Book Co., Inc., 1956.