UNIVERSITY OF CALIFORNIA

DIVISION OF ELECTRICAL ENGINEERING

ELECTRONICS RESEARCH LABORATORY

DIGITAL COMPUTER GROUP

THE TRANSLATION OF ARITHMETIC OPERATIONS INTO SWITCHING OPERATIONS IN DIGITAL COMPUTER DESIGN

by

Torben H. Meisling

Office of Naval Research
Contract N7-CNR-295-XIV

Director: Paul L. Morton

Abstract

This report presents some new methods and concepts useful in the analysis and synthesis of arithmetic elements in digital computers.

The development of comprehensive and systematic design methods is particularly important in general-purpose digital computers because of the functional complexity of the switching circuits needed. The use of the broadest viewpoints is rewarded not only by savings in equipment but frequently also by the attainment of greater functional simplicity.

Digital computers have for some time been recognized as devices capable of storing, moving, interpreting, and modifying information. In this report it is shown how the information concept of the mathematical theory of communication can be applied both to arithmetic operations on numbers and to switching operations. The essential characteristic of an arithmetic operation is a distribution of output symbols with respect to input symbols, and any physical process which provides the proper distribution can be used as an arithmetic element for that operation. Switching circuits provide a particularly flexible means of realizing needed distributions, and they can be studied most efficiently in terms of Boolean algebra.

Examples of the application of the ideas and theorems discussed in the design of practical computing circuits are given in the report.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

ERRATA

Page 9, after equation 2.2: That y=1... read That h=1...

Page 9, after equation 2.4: That y=1... read That h=1...

Page 10, after equations 2.8: ... with its complement,...
read ... with its inverse,...

Page 10, last line: opposite that .. read opposite of
that ...

Page 19, first equation: $H=4\cdot\frac{1}{4}\log_2\frac{1}{4}=$ read
$H=-4\cdot\frac{1}{4}\log_2\frac{1}{4}=$

Page 50, figure 54. the pulse gate control signals are incorrectly labelled; for each $(h^1)'$ read $(h^0)'$ and for each $(h^0)'$ read $(h^1)'$: for example, for $(h_1^1)'$ read $(h_1^0)'$.

Page 100, figure 8.9. The horizontal channel labelled $x_1' + x_8' + x_4'$ should be labelled $x_1' + x_2' + x_4'$

Page 104, line 8 from bottom: ... and B is... read and D is ...

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

CHAPTER I

INTRODUCTION

Any system performing digital computations can be broken down into the following functional parts: A memory, an element capable of carrying out arithmetic operations; a coordinating element (the control), and the input and output channels. This is equally true for the most primitive counting device, for a desk calculator, and for a modern automatically sequenced electronic digital computer. The present paper concerns itself with the arithmetic element in the latter class of computing machines; they will in the following, for short, be termed digital computers.

I. OBJECTIVES

The most important advantage of digital computers over earlier means for computation is their speed of operation. This speed, for example the multiplication of two ten digit decimal numbers in one thousandth of a second, depends on the use of components originally developed in communication engineering; in particular vacuum tubes and germanium diodes. In digital computer circuits such components are used to their greatest advantage as binary elements. A vacuum tube may be either conducting or non-conducting; in an Eccles-Jordan trigger circuit one of the two vacuum tubes is conducting, but not both; a piece of magnetic material is magnetized in one direction or in the opposite direction.

Because of the availability of binary elements many digital computers have been built to operate in the binary number system.

In other cases the binary elements are used to represent decimal digits with all computations being carried out in the decimal system. The discussion in this paper will primarily have a bearing on decimal machines.

A digital computer does not deal solely with numbers. It must also accept instructions which are coded into binary variables, just as the numbers. Both of these items may be included in the term "information". The concept of information has recently been subject to investigation in the mathematical theory of communication. One of the aims of this paper is to clarify this concept as it applies to digital computers.

The main subject of investigation is the theory and practice of translation from non-binary arithmetic operations to operations on binary variables. A typical problem would be the following: Given any two decimal digits coded into four binary variables each in an arbitrarily chosen code, find the circuit which will produce the product of the two digits, also coded into a set of binary variables. A systematic solution of such problems will be presented.

Another problem might consist in, first, finding the sum of two decimal digits, the digits and the sum being coded into binary variables without commitment to a particular code, and, second, determining the code which results in the simplest circuit for producing the sum. The first part of this problem will be formulated. The second part still awaits further development of the theory for its solution.

The paper is chiefly concerned with the logical aspects of digital computer synthesis. A set of techniques for the realization of logical functions will, however, be described so that some circuit

features important to the logical design may be taken into account.

The methods will mainly be developed by means of examples in the course of which counters, converting circuits, and (in the Appendix) a binary coded decimal adder will be subject to detailed discussion.

II. EARLIER WORK

The post-war activity in the field of digital computers has been both extensive and intensive. The remarkable lack of literature adequately representing the subject is characteristic of a fast moving development. Most material is to be found in project reports issued in the course of construction of the various computers. The major part of the development of digital computers in general must be said to be due to the numerous members of these computer projects.

The development in this paper is based first of all on C. E. Shannon's remarkable contribution in introducing symbolic logic as a descriptive algebra for switching circuits in 1938. Shannon did not think in terms of computer circuits since digital computers had not reached any high degree of development at that time.

The next step in the symbolic approach is unquestionably due to the group at the Harvard Computation Laboratory. Their extensive tabulation of switching functions, that is, Boolean Algebra expressions applied to switching phenomena, and their derivation of vacuum tube operators has had decisive influence on the development. The Harvard group has primarily been interested in establishing the equivalence between operations in Boolean Algebra and vacuum tube circuits and in developing methods for finding the simplest solution.

The translation from arithmetic requirements to logical opera-

tions of binary variables, the subject of this paper, has not, so far, been systematically investigated.

The development of broader and improved notions of the nature of information is due to C. E. Shannon and to N. Wiener. Its use in this paper is almost entirely qualitative. The future development may, however, very well be determined by the introduction of such notions.

III. ORGANIZATION OF CHAPTERS

Chapter II deals with the fundamental concepts of the logical design and reviews concepts and formulas of Boolean Algebra as they apply to the fundamental logical operations. In the second section of Chapter II the operations of Boolean Algebra, representing the logical operations, are considered from the viewpoint of information flow. The quantitative concept of information is applied to the logical functions.

In Chapter III arithmetic operations on numbers are discussed in terms of the concept of information.

Chapter IV deals with a set of techniques for the realization of the logical operations discussed in Chapter II. It brings in the most important circuit features and the physical nature of the binary signals introduced in Chapter II.

Chapter V uses counters and related circuits as examples of a method of design centered around what will be termed a "transition chart". This chapter at the same time discusses the logical behaviour of counters.

In Chapter VI the equations governing decimal addition and

other arithmetic operations on binary variables, representing decimal digits in arbitrary codes, are formulated. The chapter also covers some of the ways in which an arithmetic operation can be divided into several steps in order to simplify the circuitry.

Chapter VII contains a summary of the results and some recommendations for future investigations.

In the Appendix an example of a binary coded decimal adder is described and the process of adding and subtracting multi-digit numbers with arbitrary signs is outlined.

The formulas in the various chapters are identified by a decimal number; the units in this number refer to the chapter and the number after the decimal point is the number of the formula in the chapter; 5.17, for example, refers to formula 17 in Chapter V. The figures and tables are numbered separately in the same manner. The Appendix is given the number VIII in continuation of the chapter numbers.

A numerical superscript in the text,[7], refers to the bibliography in back. One or more asterisks, *, refer to a footnote on the same page.

CHAPTER II

THE MATHEMATICAL TOOLS USED IN THE ANALYSIS

Boolean Algebra has been used quite extensively in the design of digital computers since its first engineering application by C. E. Shannon in 1938[2]. It is an algebra of binary variables whose operations are exactly analogous to the fundamental operations employed in digital computers.

The Concept of Information[5,6] has not, so far, been applied in the analysis of digital computer operations.

I. BOOLEAN ALGEBRA

After having been applied by Shannon in the analysis of relay circuits, Boolean Algebra was developed into an elaborate system of "Vacuum Tube Operators" especially suited for design problems in digital computers by the staff of the Harvard Computation Laboratory[3].

There is a slight difference between the algebra originally introduced by Shannon and that used at Harvard. The latter is simply the special case of ordinary algebra for which all variables are restricted to the values zero and one. In Shannon's system one departure from ordinary algebra is made. It will be pointed out later in this section (see equations 2.7 and 2.11).

Shannon's system lends itself well to the representation of diode circuits; also, it is simpler to manipulate than the Harvard system. Since diode circuits will be introduced later (in Chapter IV) to illustrate the design principles, however,

Shannon's system will be used in what follows. This choice is of no deeper consequence, however.

It is not the purpose here to justify formally the application of Boolean Algebra to the logical problems encountered in digital computer design. For a clarification of this point the reader is referred to Shannon's article[2]. Since, however, some differences in viewpoint and terminology exist, a review will be made of the fundamental concepts and theorems as they will be used here. This collection of formulas will at the same time serve as a reference table.

The binary variables, which carry the information in a digital computer, are stored in bistable circuits of which an Eccles-Jordan trigger circuit[7] may serve as an example. The symbol used to represent the logical unit, which will be called a toggle, is shown in Figure 2.1.

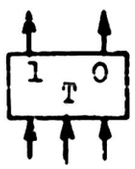


Figure 2.1 - Symbol representing a toggle

A toggle has two stable states, called the 1-state and the 0-state. It has two output terminals, marked "1" and "0" and three input terminals, one (opposite the 1-output terminal) which upon reception of a pulse will transfer the toggle to the 1-state (if it is not already in the 1-state); one which upon reception of a pulse will transfer it to the 0-state; and one (in the middle)

which will reverse the state of the toggle, that is, transfer it to the opposite state irrespective of the initial state, upon reception of a pulse.

The output terminals carry signals, voltage levels, for example, indicating the state of the toggle. There are only two possible signals, one will be called a 1-signal and the other a 0-signal. When the toggle is in its 1-state, the 1-output terminal will carry a 1-signal and the 0-output terminal a 0-signal. When the toggle is in its 0-state, the output signals will be reversed.

The state of a toggle will be represented by a variable, x, assigned to it. When the toggle is in its 1-state, $x=1$, when it is in its 0-state, $x=0$. From this it follows that x also represents the signal on the 1-output terminal. The signal on the 0-output terminal will be represented by x' ("x prime"),

$$x' = 1 - x,$$

where the minus sign has the same meaning as in ordinary algebra.

More will be said about the input terminals in Chapter IV.

Binary signals depending on the state of a single toggle each, may be combined into a binary signal depending on the states of a number of toggles by means of two fundamental operations, gating and mixing. A gate will be symbolized by a rectangle with the letter G, a mixer by a triangle with the letter M (see Figure 2.2).

Gates and mixers may have any number of input terminals, but only one output terminal. A gate has a 1-signal on its output term-

inal if and only if all input terminals receive 1-signals.

Figure 2.2 - Symbols representing a gate and a mixer

Let the input signals be represented by $x_1, x_2, x_3, \ldots$, and the output by $g(x_1, x_2, x_3, \ldots)$, then the equation

$$g(x_1, x_2, x_3, \ldots) = x_1 \cdot x_2 \cdot x_3, \ldots, \qquad 2.1$$

in which the multiplication sign has the same meaning as in ordinary algebra, represents a gate.

A mixer will have a 1-signal on its output terminal if at least one 1-signal is present on its input terminals. Again, let the input signals be represented by $x_1, x_2, x_3, \ldots$; and the output by $h(x_1, x_2, x_3, \ldots)$; then the equation

$$h(x_1, x_2, x_3, \ldots) = x_1 + x_2 + x_3 + \ldots \qquad 2.2$$

represents a mixer. That y=1 when only one of the input signals is 1 follows from the rules for addition in Boolean Algebra

$$0 + 0 = 0 \qquad 2.3$$

$$0 + 1 = 1 + 0 = 1 \qquad 2.4$$

That y=1 when more than one input signal is 1 follows from the postulate used by Shannon

$$1 + 1 = 1 \qquad 2.5$$

A mixer could also be characterized the following way, its output signal is 0 if and only if all input signals are 0, meaning that the equation

$$h'(x_1, x_2, x_3, \ldots) = h(x_1, x_2, x_3, \ldots)' = x_1' \cdot x_2' \cdot x_3' \cdot \ldots \qquad 2.6$$

is valid.

The transition from equation 2.2 to equation 2.6 expresses de Morgan's Theorem which explicitly states that if

$$y = x_1 + x_2 + x_3 + \cdots \qquad 2.7$$

then

$$y' = x_1' \cdot x_2' \cdot x_3' \cdot \cdots ,$$

and if

$$z = x_1 \cdot x_2 \cdot x_3 \cdot \cdots \qquad 2.8$$

then

$$z' = x_1' + x_2' + x_3' + \cdots$$

One other operation deserves attention. The operation of inversion replaces a signal with its complement, a 1-signal is converted to a 0-signal and a 0-signal to a 1-signal. Although inversion as an isolated operation will not be considered in the engineering design examples, a symbol as shown in Figure 2.3 will be introduced for its representation.

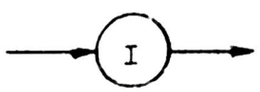


Figure 2.3 - Symbol representing an inverter

An inverter has one input, x, and one output, y, and can be represented by the equation

$$y = x' \equiv 1 - x \qquad 2.9$$

From this it is clear that inversion is inherently present in a toggle since the output signal at one terminal is always the opposite that at the other.

In the above the equivalence between the operations encountered in digital computer circuits and those encountered in Boolean Algebra has been stated. Below are listed the postulates and some additional theorems which will be used in the following.

$$x \cdot x \cdot x \cdot \ldots = x \tag{2.10}$$

$$x + x + x + \ldots = x \tag{2.11}$$

$$x_1 \cdot x_2 = x_2 \cdot x_1 \tag{2.12}$$

$$x_1 + x_2 = x_2 + x_1 \tag{2.13}$$

$$x + 1 = 1 \tag{2.14}$$

$$x + 0 = x \tag{2.15}$$

$$x \cdot 1 = x \tag{2.16}$$

$$x \cdot 0 = 0 \tag{2.17}$$

$$x' \cdot x = 0 \tag{2.18}$$

$$x + x' = 1 \tag{2.19}$$

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 = x_1 + x_2 + x_3 \tag{2.20}$$

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) x_3 = x_1 \cdot x_2 \cdot x_3 \tag{2.21}$$

$$x_1 (x_2 + x_3) = x_1 x_2 + x_1 x_3 \tag{2.22}$$

$$x_1 + x_2 \cdot x_3 = (x_1 + x_2)(x_1 + x_3) \tag{2.23}$$

$$x_1 + x_1' \cdot f(x_2, x_3, x_4, \ldots) = x_1 + f(x_2, x_3, x_4, \ldots) \tag{2.24}$$

$$x_1 \cdot f'(x_1, x_2, x_3, \ldots) = x_1 \cdot f'(1, x_2, x_3, \ldots) \tag{2.25}$$

As an example of a proof, take equation 2.23. When the right side of the equation is multiplied out by use of equations 2.10 and 2.22 the following expression is obtained

$$x_1 + x_2 x_1 + x_3 x_1 + x_3 x_2$$

When equation 2.22 is used, this becomes

$$x_1 (1 + x_2 + x_3) + x_3 x_2$$

According to equation 2.14, however,

$$1 + x_2 * x_3 = 1$$

Then the right side expression becomes

$$x_1 * x_3 x_2$$

which is the left side of the equation 2.23.

In Figure 2.4 is shown a simple block diagram representing the formation of a signal depending on the states of 3 toggles, the formation of a function of 3 variables.

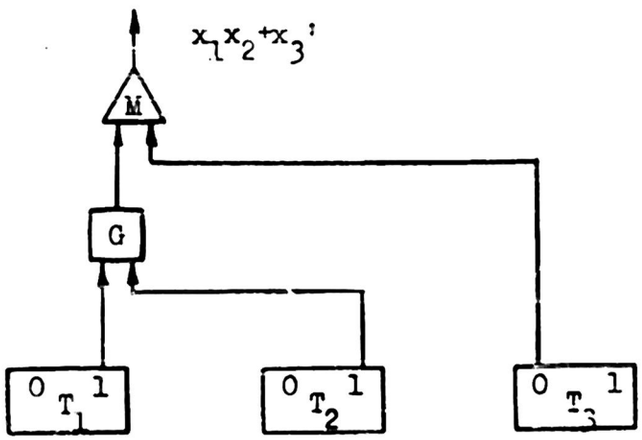


Figure 2.4 - Formation of a switching function of three variables

A function may be described by means of a truth table, giving its value for all value combinations of the variables upon which it depends. The truth table for the function in Figure 2.4 is shown below

State Number	x_1	x_2	x_3	f(x)
0	0	0	0	1
1	1	0	0	1
2	0	1	0	1
3	1	1	0	1

(continued on the following page)

State Number	x_1	x_2	x_3	f(x) (continued)
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	1

A function of binary variables, such as the one shown in Figure 2.4 is called a switching function.

Any switching function can be expanded in a sum of products, each product containing as many factors as there are independent variables.

As an example f(x) of Figure 2.4 will be expanded. Each term is multiplied by unity,

$$f(x) = x_1x_2+x_3' = x_1x_2(x_3+x_3') + x_3'(x_1+x_1')(x_2+x_2')$$

$$f(x) = x_1x_2x_3+x_1x_2x_3'+x_1x_2x_3'+x_1x_2'x_3'+x_1'x_2x_3'+x_1'x_2'x_3'$$

$$f(x) = x_1x_2x_3+x_1x_2x_3'+x_1x_2'x_3'+x_1'x_2x_3'+x_1'x_2'x_3'$$

It is important to note that each product in the expansion becomes 1 for one and only one value combination of the independent variables. The number of 1's occurring in the truth table must, therefore, be the same as the number of terms in the expansion.

A constant 1-signal can be considered a switching function of any desired variables. Since it is never 0, it must contain all the possible products in its expansion. If it is considered a function of three variables, x_1, x_2, x_3, the following expansion is valid

$$1 = x_1x_2x_3+x_1'x_2x_3+x_1x_2'x_3+x_1x_2'x_3+x_1x_2x_3'+x_1'x_2x_3'+x_1x_2'x_3'+$$

$$+x_1'x_2'x_3' \qquad 2.2c$$

Note that the right side can be found by multiplying out the following expression, which according to equation 2.19 has the value 1.

$$(x_1+x_1')(x_2+x_2')(x_3+x_3') = 1 \qquad 2.22$$

II. THE CONCEPT OF INFORMATION

A digital computer derives its utility from its capability to transfer, interpret and modify information. The transfer presents few logical problems. As will be seen presently, the modification of information, for example, arithmetic operations on numbers, and the interpretation of information, interpretation of instructions, for example, are logically similar. Interpretation may be said to be the static and modification the dynamic form of a filtering process. Before going further, it will be necessary to clarify what will be meant by information.

Consider three binary variables, x_1, x_2, x_3. Assume that the value combination which occurs at a given time is determined, wholly or in part, by an external source of which no knowledge is available otherwise. Under these conditions the three variables are said to be carriers of information relative to the source. In a digital computer, the source may be the problem data and the instructions pertaining to a particular problem of which no knowledge is had at the time of the design of the machine. A counter or any other logical circuitry which solely depends on the machine cycle, that is, whose state is predictable at all times, independently of the input data, has no information content relative to the input data in this sense.

Let it, for the moment, be assumed that the variables, x_1, x_2, x_3, depend entirely on the input data, that they are subject to no restrictions due to the machine structure. It is natural to suppose that the input data is distributed evenly over the possible combinations if a large variety of problem types is taken into consideration. This means that each variable will have a probability of one-half of taking on the value 1 and a probability of one-half of taking on the value 0 independently of the value of the other variables. Then the eight possible value combinations listed below will each occur with the probability of one-eighth.

x_1	x_2	x_3
0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

Each value combination corresponds to a symbol in a discrete noiseless system as discussed by Shannon*. He has introduced a measure for the information per symbol**.

$$H = -K \sum_{i=1}^{n} p_i \cdot \log p_i \qquad 2.20$$

---

*See The Mathematical Theory of Communication[5], page 7.

**See The Mathematical Theory of Communication[5], page 19.

where p_i is the probability for the occurrence of the i'th symbol of n possible symbols and K is a constant determining the unit of measure. Let K=1 and let the logarithm with base 2 be used; then, in the case presented above,

$$H = -8 \cdot \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits}$$

Frequently the machine structure imposes restrictions upon the information carrying variables. Such a restriction may be written in terms of a logical equation, for example,

$$x_2 \cdot x_3 = 0$$

excludes the two last value combinations in the table given above. Again assuming an even distribution of combinations, the amount of information will be

$$H = -6 \frac{1}{6} \log_2 \frac{1}{6} = 2.58 \text{ bits}$$

The restriction thus decreases the amount of information per symbol.

Next consider a function, f(x), of the three independent variables, x_1, x_2, x_3. Let the function be the one shown in Figure 2.4 page 12,

$$f(x) = x_1 \cdot x_2 + x_3 \qquad 2.29$$

Suppose now that a certain combination of values, x_1, x_2, x_3, occurred at a given time; then, even if it is not known which one actually did occur, a clue is to be had from the knowledge of f(x). Referring to the table on pages 12 and 13, if f(x)=0 then one of the combinations, or symbols, number 4, 5, or 6, must have occurred; which one cannot be determined without further information. A switching function, therefore, conveys some of the information present in the variables of which it is a function.

Since a single switching function only represents two possible symbols, 0 and 1, one of which must occur, equation 2.28, for a single switching function becomes

$$H = -[p \log_2 p + (1-p) \log_2 (1-p)] \text{ bits} \qquad 2.30$$

The maximum value of this expression, 1, occurs for $p=\frac{1}{2}$*. A single switching function will, therefore, never convey more than one bit of information about its variables. If more information is desired several switching functions must be employed.

If x_1, x_2, and x_3 are unrestricted the amount of information conveyed by $f(x) = x_1 \cdot x_2 + x_3$' can be found by counting the 1's and 0's in the table on pages 12 and 13.

$$H = -\left[\frac{5}{8} \log_2 \frac{5}{8} + \frac{3}{8} \log_2 \frac{3}{8}\right] \text{ bits} = 0.96 \text{ bits}$$

If several switching functions are considered the total amount of information conveyed by the set is always less than or equal to the sum of the amounts of information carried by the individual functions. If it is less, the functions are said to be dependent.

In analyzing the information conveyed by a set of switching functions, two functions, f and g may serve as an example. The two functions represent the equivalent of four different symbols, one of which must occur.

By counting the number of x-combinations, the combination of $x_1=1$, $x_2=0, x_3=1$, for example, for which each symbol occurs and dividing the numbers by the total number of x-combinations a probability for each symbol is derived. Let $p[h(x_1, x_2, x_3, \ldots)]$ be the ratio of

---

*See The Mathematical Theory of Communication, page 20.

the number of x-combinations for which $h(x_1, x_2, x_3, \ldots) = 1$ to the total number of x-combinations; then these probabilities are easily seen to be

$$p[f \cdot g], \quad p[f' \cdot g], \quad p[f \cdot g'], \text{ and } p[f' \cdot g']$$

By use of these probabilities in equation 2.28, the amount of information may be computed.

The discussion above establishes the means by which the logical elements of a digital computer modify and interpret binary coded information. It shows, by adoption of the quantitative information concept, how this information can be measured, and, by virtue of equation 2.28, it points to the importance of distributions in the mechanism. A set of switching functions will be said to represent an information filter.

As a simple example of filter consider

$$f(x_1, x_2) = x_1 \qquad 2.31$$

$$g(x_1, x_2) = x_1x_2 + x_1'x_2' \qquad 2.32$$

Note that

$$g' = (x_1' + x_2')(x_1 + x_2) = x_1x_2' + x_1'x_2$$

Derive

$$f \cdot g = x_1x_2$$

$$f' g = x_1'x_2'$$

$$f\, g' = x_1x_2'$$

$$f' g' = x_1'x_2 ,$$

then

$$p[f \cdot g] = p[f' \cdot g] = p[f \cdot g'] = p[f' \cdot g'] = \frac{1}{4}$$

and

$$H = 4 \cdot \frac{1}{4} \log_2 \frac{1}{4} = 2 \text{ bits},$$

meaning that there is no loss of information in the filter. As a consequence it must be possible to solve for x_1 and x_2. Put into the equation 2.32 the value $x_1 = f$ from equation 2.31,

$$g = f \cdot x_2 + f' x_2' \ ;$$

derive $$g' = f x_2' + f' x_2$$

Multiply the first of these two equations by f and the second by f' and add them,

$$g \cdot f + g' \cdot f' = f x_2 + f' x_2 = x_2$$

The solved equations are, therefore,

$$x_1(f,g) = f$$

$$x_2(f,g) = f \cdot g + f' \cdot g'$$

CHAPTER III

THE ARITHMETIC OPERATION ON NUMBERS

A digital computer interprets and modifies the information present in its input data. The interpretation serves to translate the instructions into machine language and the modification consists in arithmetic operations to be carried out upon the input data in accordance with the instructions. The derivation of a square root to three significant digits, the summation of a set of numbers, and the calculation of a determinant are all examples of arithmetic operations.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Table 3.1 - Multiplication of two decimal digits

Let one particular arithmetic operation be considered separate-

ly, that is, let the information content inherent in the choice of one arithmetic operation out of several be ignored for the moment; then a filtering process on numerical data is at hand. At the time of the design of the machine, the numerical data can, of course, not be predicted. The arithmetic element must then be capable of handling a finite set of numbers, or, in the language of the Theory of Communication, a finite set of symbols.

Consider, as an example, the multiplication of two decimal digits (see Table 3.1).

Symbol	Occurrences	Symbol	Occurrences	Symbol	Occurrences
0	19	15	2	40	2
1	1	16	3	42	2
2	2	18	4	45	2
3	2	20	2	48	2
4	3	21	2	49	1
5	2	24	4	54	2
6	4	25	1	56	2
7	2	27	2	63	2
8	4	28	2	64	1
9	3	30	2	72	2
10	2	32	2	81	1
12	4	35	2		
14	2	36	3		

Table 3.1 - Frequency of output symbols in multiplication of two decimal digits

From Table 3.1 it appears that the operation of multiplication upon two decimal digits has one hundred possible input symbols, any combination of two decimal digits, and 37 possible output symbols, occuring with the frequencies indicated in Table 3.2.

Assuming the input evenly distributed over all symbols, that is, that all input symbols occur with the same probability, the information content is

$$H_i = -100 \frac{1}{100} \log_2 \frac{1}{100} = 6.64 \text{ bits}$$

The information content in the product, the output, is

$$H_0 = -[0.19 \cdot \log_2 0.19 + 5 \cdot 0.01 \log_2 0.01 + 22 \cdot 0.02 \cdot \log_2 0.02 + 4 \cdot 0.03 \log_2 0.03 + 5 \cdot 0.04 \log_2 0.04] \text{ bits} = 4.80 \text{ bits}$$

As is understandable, since there are only 37 output symbols compared to one hundred equally probable input symbols, a loss of information has taken place.

From the above it is clear that any discrete filter which brings about the correct distribution may serve as an arithmetic element for that operation. The amount of information in the output must at least be that prescribed by the operation, H_0 in the example; if it is more, further filtering may be necessary.

When formulated this way, the problem of design becomes one of realizing a distribution first and then determining the number code, that is, the interpretation of individual symbols in terms of individual numbers.

The choice of operation represents another source of information which could be considered along with the numerical data. If considered, it would add to the information in the result.

Arithmetic operations do not represent a unique group of

discrete filters. The term merely serves to define those filters which are of primary importance in digital computer design.

The fundamental operations on binary variables and some of their properties as information carriers have been discussed in the previous chapter. The next step will be to bridge the gap between arithmetic operations on numbers and logical operations on binary variables, represented by switching functions. Before this step will be taken, however, the realization of switching functions by means of diode networks will be discussed in order that some engineering design aspects, which affect the logical design, may be taken into account.

CHAPTER IV

THE REALIZATION OF SWITCHING FUNCTIONS BY MEANS OF DIODE NETWORKS

The discussion in Chapter II of the characteristics of the basic logical elements, toggles, gates and mixers, did not in any way bring in the physical nature of these elements. In this chapter a set of techniques for their realization will be briefly described.

The Eccles-Jordan trigger circuit is undoubtedly the most commonly used toggle circuit. The diagram for a self-biased trigger circuit is shown in Figure 4.1.

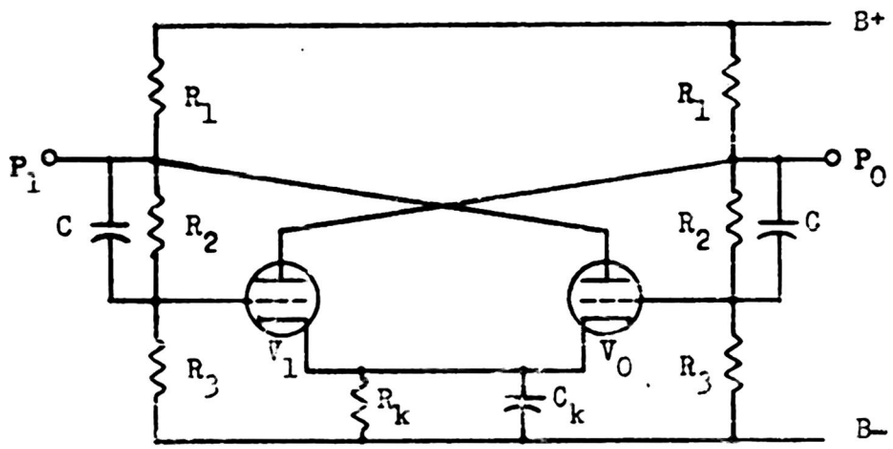


Figure 4.1 - Self-biased Eccles-Jordan trigger circuit

The resistors R_1, R_2, R_3, and R_k are chosen so that only one of the tubes, V_1 or V_0, is conducting in the stable state. When V_1 is conducting P_1 assumes the higher of its two potentials and P_0 the lower of its potentials; when V_0 is conducting, the situation

is reversed*.

Let the 1-state correspond to conduction of V_1, the 0-state to conduction of V_0. Furthermore, let P_0 be the 0-output terminal and P_1 the 1-output terminal. Then a 1-signal corresponds to the higher plate voltage and a 0-signal to the lower plate voltage. These signals will be referred to as level signals (compare Chapter II, page 7).

In order to change the state of the trigger circuit, a negative pulse is applied to the grid of the conducting tube. If negative pulses of approximately equal amplitude are supplied to both grids simultaneously, the state of the trigger circuit will be reversed.

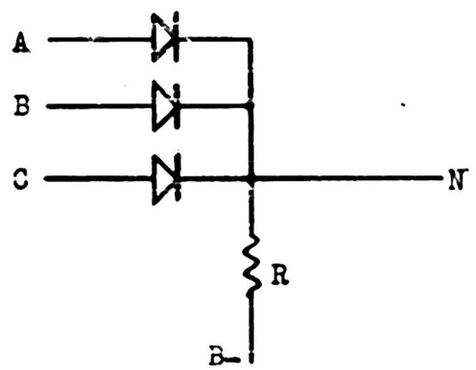


Figure 4.2 - Diode mixer

*For a more detailed account of the operation of this circuit, see R.F.K.Seal, Cathode Coupled Half Shot Multivibrator, Electronics, volume 20, (September 1947), pages 150-158

The output signals from two or more trigger circuits may be combined in a diode mixer as shown in Figure 4.2, (the arrow, formed by the triangle in the diode symbol, indicates the direction of the path of low resistance for a positive current).

Let all trigger circuits have the same higher and lower plate potentials. Their plates are connected, directly or through cathode followers, to the points A, B, and C. The potential B- is considerably below the lower of the two possible plate potentials. The resistance of R is large compared to the forward resistance of the diodes. It is easily seen that N will at all times be at a potential approximately equal to the highest of the potentials at A, B, and C. In other words, N will assume its higher potential, that is, hold a 1-signal, only if at least one of the potentials A, B, and C, assumes its higher value, that is, only if at least one 1-signal is present on the input (compare Chapter II, page 9).

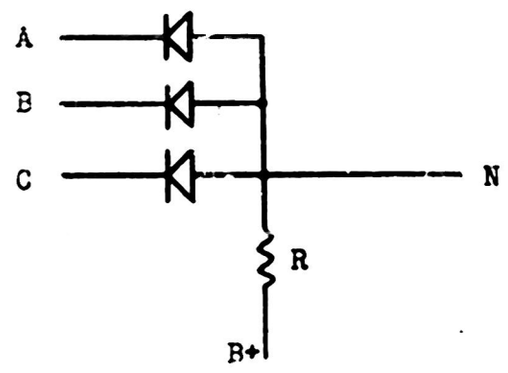


Figure 4.3 - Diode gate

A diode gate is shown in Figure 4.3

B+ is a voltage considerably above the higher trigger circuit plate voltage; R_1 again, must be large compared to the forward resistance of the diodes. Under these conditions N will be approximately equal to the lowest of the potentials at A, B, and C, or N will assume its higher potential, that is, hold a 1-signal, if and only if, all the potentials at A, B, and C assume their higher value, that is, if and only if, all input signals are 1-signals.

The trigger circuit and the two diode circuits described above form a system by which any switching function can be realized in terms of high and low voltage levels. This system will suffice for what has been termed interpretation or decoding; it does not bring in any modification of the information stored in the toggles.

In the majority of cases the switching functions are, however, used to control the changes of toggles in other parts of the computer, or, as is the case in most arithmetic elements, to control changes in the very toggles from which they are derived.

The input signals to the trigger circuits are usually in the form of negative pulses. It is, therefore, necessary to introduce a second type of binary signals, pulse signals, at least at the last stage preceding the input terminals to the trigger circuits.

The question now arises whether to perform the logical operations on the pulse signals, that is, to introduce the pulses at an early stage, or to perform them on the level signals, that is, to introduce the pulses at a late stage in the logical circuitry.

Both methods have been used successfully.

Pulse operation requires gates and mixers which have rise and fall times short enough to permit a faithful transmission of the pulses. Voltage level operation requires rise and fall times comparable to the interval between two consecutive pulses. In systems where the shortest pulse interval is five or more times as large as the pulse width and where diode gates and mixers are used, it will generally be an advantage to use the level signals in the logical circuitry. Figure 4.4 illustrates the principle of level coded operation as it will be used in the design examples in the later chapters.

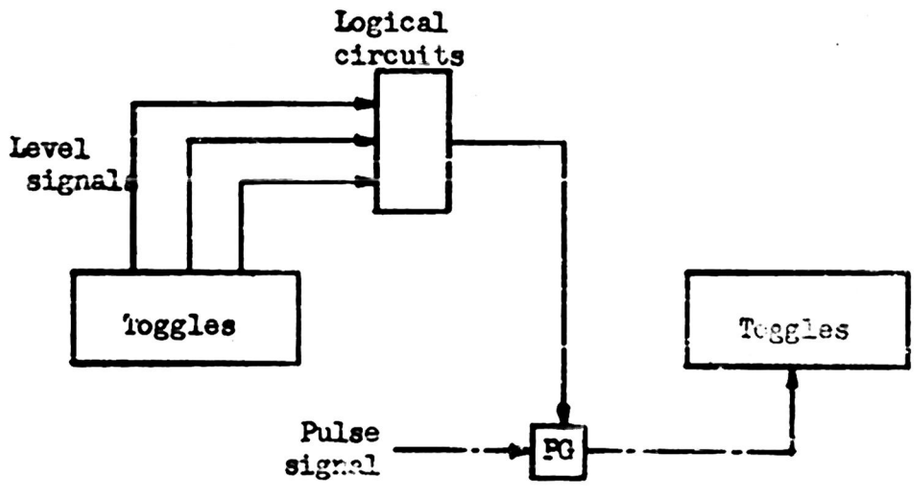


Figure 4.4 - Principle of level coded operation

A pulse gate receives on one terminal a level coded signal and on the other a pulse coded signal. The output signal is pulse coded. A circuit for such a gate is shown in Figure 4.5 together

with the trigger circuit to which it connects*.

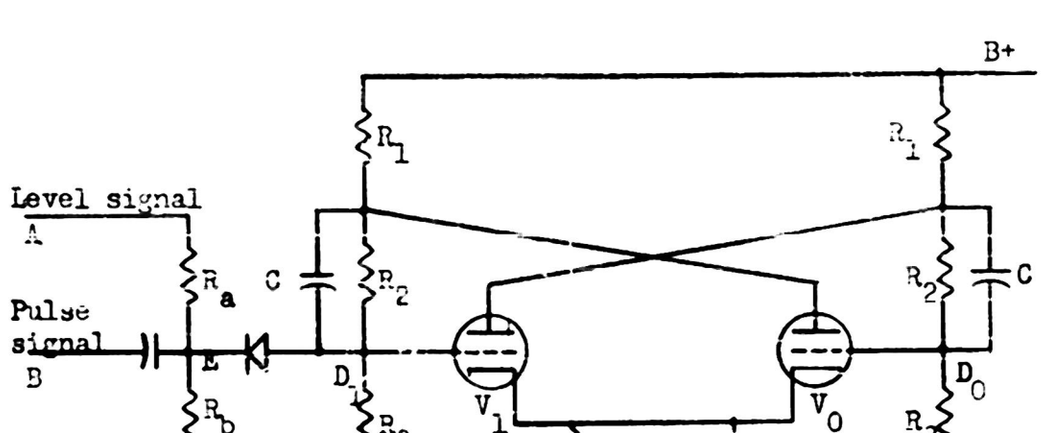


Figure 4.5 - Pulse gate and trigger circuit

The signal at A consists of two possible levels. All trigger circuits being supplied by the same two supply levels B+ and B-, the levels at A will be approximately equal to the two plate potentials of the trigger circuit to which the gate connects. Let, for example, these two levels be 160 and 110 volts above B-**. The point D_1 of the trigger circuit also has two possible potential levels; let, in the example, these two levels be 60 and 40 volts above B-. Furthermore, let $R_a = R_b$. If no pulse is present, E will

---

* This pulse gate was introduced by James Knapton and Louis Stevens, both at that time working on the Computer project at the University of California in Berkeley.

** This example is taken from the circuits in the California Digital Computer.

tend to assume one of the potentials 80 or 55 volts. Now for point D_1 to be sensitive to negative pulses at all, V_1 must be conducting, that is, D_1 must be at the higher of its two potentials, 60 volts. If A is at 110 volts, then E tends to be at 55 volts and the diode will be driven into its conducting region, meaning that a negative pulse of adequate magnitude applied at B will reach the grid at D_1 and cause the trigger circuit to be switched. If A is at 160 volts, then E tends to be at 80 volts, meaning that if the negative pulse applied to E is less than 20 volts, it will be subject to the attenuation of the back resistance of the diode and not cause the trigger circuit to change state.

In order to arrive at a pulse gate for a reversing input connection, another diode may be inserted between D_0 and E. It will be noted that, due to the two potentials at D_1 and D_0, a pulse will only be applied to one of the grids, the grid of the conducting tube, at any time.

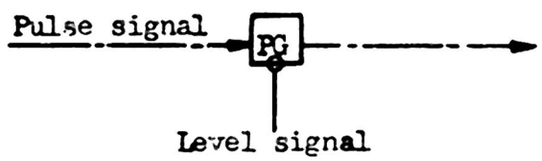


Figure 4.6 - Symbol representing low-pass pulse gate

A pulse gate of this type which passes the pulse for a low control voltage and stops it for a high control voltage will be called a low-pass pulse gate. It will be represented by a combination

of two symbols, an inverter symbol and a high-pass pulse gate symbol. This combination is shown in Figure 4.6. A broken line, as shown, will be used to indicate a pulse channel.

For trigger circuits with other plate and grid potentials than the ones used in the example, R_a and R_1 must, of course, be chosen differently. The example serves to demonstrate the principle rather than to give engineering design details.

If pulses are supplied to the two grids of a trigger circuit simultaneously, they must, as mentioned, be approximately of equal amplitude. This condition is easily fulfilled in the reversing input connection described above, where the pulses originate in the same pulse gate.

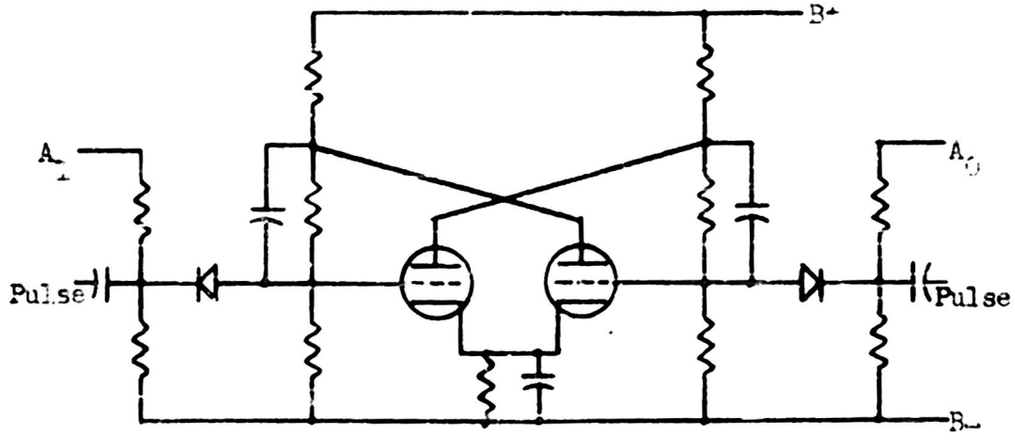


Figure 4.7 – Trigger circuit with two pulse gates

If, however, the two pulses come from separate logical circuits, see Figure 4.7, slight variations in levels at A_1 and A_0 are likely to result in unequal pulse amplitudes at the grids causing erratic

operation with this type of circuitry. This condition must be taken into account in the logical design where the following rule will be adopted. No toggle may at any time receive simultaneous 0-input and 1-input signals from separate pulse gates.

CHAPTER V

COUNTERS AND CONVERTING CIRCUITS

The two operations to be discussed in this chapter modify the information stored in a set of toggles. A counter is an arithmetic element of a particularly simple type. By a converting circuit will be understood a circuit which, upon application of a pulse, changes the content of a set of toggles from one value to another. Although conversion would not always be thought of as an arithmetic operation, there is no significant difference between the two operations, conversion and counting. One method of treatment will be used in both cases.

I. MODE OF OPERATION

An array of toggles storing binary variables belonging to the same logical unit, for example, a register or the toggles in a counter, will be called a toggle group. The total number of distinct states in a toggle group is

$$2^n,$$

where n is the number of toggles in the group. The output signal from the group consists of 2^n different symbols, one for each state. These output symbols are evident, only, by simultaneous observation of the state of all toggles. The individual toggle output signals can be subjected to logical operations as described in Chapter II. In other words, the information in the 2^n output symbols may be subjected to filtering by means of these logical operations.

For the type of operation which will be discussed here the filtered output information will be used to control the change of state which will take place in the group when a pulse is applied. This operation is shown schematically in Figure 5.1.

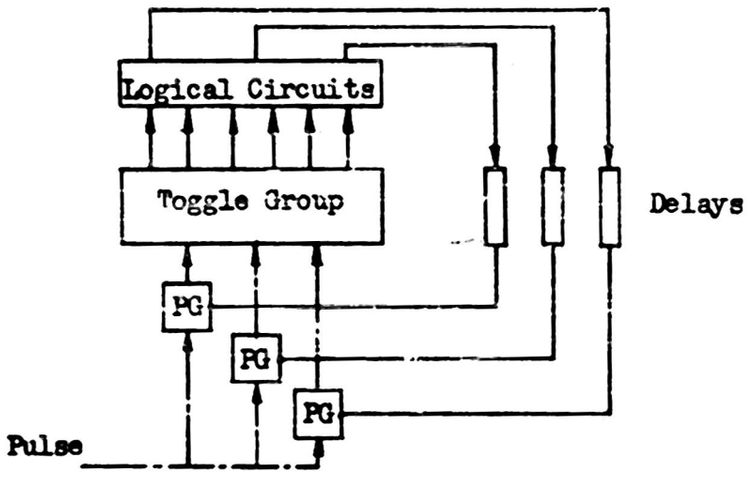


Figure 5.1 - General block diagram for converters and counters

For each state the logical filter specifies the next state into which the group is to be transferred by application of a pulse. In order that the pulse gates may remain unchanged during the pulse, a delay larger than the pulse width but less than the minimum pulse interval, is inserted between the logical filter and the pulse gates. Such a delay is always assumed present. It will not be shown in the block diagrams*.

---

*When constructed the toggle filter combination usually provides this delay automatically by virtue of the finite rise and fall times present in the circuit.

An operation of this type is entirely specified when the transitions following each of the 2^n possible states are given. The following sections will show how this specification can be made in terms of the fundamental binary variables and how the logical design follows immediately from this specification.

II. THE TRANSITION CHART

Consider a group of toggles, T_0, T_1, T_2, T_3, . . . T_{n-1}, and let the state of each toggle be represented by the binary variables, x_0, x_1, x_2, . . . x_{n-1}. Each of the 2^n possible states of the group will be designated by the decimal equivalent of the binary number represented by the succession of binary digits formed by the variables x_{n-1}, x_{n-2}, . . ., x_1, x_0. The arithmetic expression for this designation, d, will be

$$d = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + \ldots + x_{n-1} \cdot 2^{n-1} \qquad 5.1$$

Each designation represents one symbol of the 2^n possible. No identification of the states with the number represented by the designation is in any way implied.

For purposes of explanation a specific problem will be formulated. Let it be desired to design a circuit for which the following transitions between the states of a toggle group containing four toggles T_0, T_1, T_2, and T_3 are required

1→2→3→5→4→6→7→8→9→13→12→11→10→1 ;

0→0 ; 14→0; 15→0 *

---

* This example is chosen on basis of its merits as a demonstration model. It is probably not of any practical use.

It will be necessary to distinguish between the values of the variables before a transition and those after a transition. Let the values before the transition be x_0, x_1, x_2, x_3 and those after, y_0, y_1, y_2, y_3. A y, of course, becomes an x for the next transition.

Next it is observed that a state of the group can be represented by an equation, state 9, for example, can be described by

$$x_3x_2'x_1x_0 = 1 \qquad 5.2$$

since this equation is only fulfilled if

$$x_3 = 1\ ;\ x_2 = 0\ ;\ x_1 = 0\ ;\ x_0 = 1$$

corresponding to

$$d = 1\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 9$$

The states after a transition can be represented by similar equations in y.

Now, if state 2 is to occur after state 1 and only after state 1, then this means that

$$y_3'y_2y_1'y_0' = 1$$

only when

$$x_3'x_2x_1'x_0 = 1$$

that is, when

$$y_3'y_2y_1'y_0' = x_3'x_2x_1'x_0 \qquad 5.3$$

If state 0 is to occur only after states 11, 15 and 0, then

$$y_3'y_2'y_1'y_0' = 1$$

only if one of the equations

$$x_3x_2x_1x_0' = 1\ ,$$

$$x_3x_2x_1'x_0 = 1\ ,$$

$$x_3'x_2'x_1'x_0' = 1$$

is fulfilled, that is, when

$$y_3'y_2'y_1'y_0' = x_3x_2x_1x_0' + x_3x_2x_1x_0 + x_3'x_2'x_1'x_0' \qquad 5.4$$

is valid. There will be an equation of this type for each y-state. In order to save writing, let

$$s_d = 1$$

be the equation for the x-state corresponding to the designation d; each s is a product, for example,

$$s_{14} = x_3x_2x_1x_0';$$

similarly let t_d represent the y-products

Explicitly

$$\begin{aligned}
s_0 &= x_3'x_2'x_1'x_0' & t_0 &= y_3'y_2'y_1'y_0' \\
s_1 &= x_3'x_2'x_1'x_0 & t_1 &= y_3'y_2'y_1'y_0 \\
s_2 &= x_3'x_2'x_1x_0' & t_2 &= y_3'y_2'y_1y_0' \\
s_3 &= x_3'x_2'x_1x_0 & t_3 &= y_3'y_2'y_1y_0 \\
s_4 &= x_3'x_2x_1'x_0' & t_4 &= y_3'y_2y_1'y_0' \\
s_5 &= x_3'x_2x_1'x_0 & t_5 &= y_3'y_2y_1'y_0 \\
s_6 &= x_3'x_2x_1x_0' & t_6 &= y_3'y_2y_1y_0' \\
s_7 &= x_3'x_2x_1x_0 & t_7 &= y_3'y_2y_1y_0 \\
s_8 &= x_3x_2'x_1'x_0' & t_8 &= y_3y_2'y_1'y_0' \\
s_9 &= x_3x_2'x_1'x_0 & t_9 &= y_3y_2'y_1'y_0 \\
s_{10} &= x_3x_2'x_1x_0' & t_{10} &= y_3y_2'y_1y_0' \\
s_{11} &= x_3x_2'x_1x_0 & t_{11} &= y_3y_2'y_1y_0 \\
s_{12} &= x_3x_2x_1'x_0' & t_{12} &= y_3y_2y_1'y_0' \\
s_{13} &= x_3x_2x_1'x_0 & t_{13} &= y_3y_2y_1'y_0 \\
s_{14} &= x_3x_2x_1x_0' & t_{14} &= y_3y_2y_1y_0' \\
s_{15} &= x_3x_2x_1x_0 & t_{15} &= y_3y_2y_1y_0
\end{aligned} \qquad 5.5$$

The notation introduced in equations 5.5 applies to any system of four binary variables, of course. It can easily be extended to problems involving more than four variables.

The complete system of equations, of which equations 5.3 and 5.4 are examples, can be written in matrix form

$$\begin{Bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \\ t_{11} \\ t_{12} \\ t_{13} \\ t_{14} \\ t_{15} \end{Bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{Bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \\ s_{15} \end{Bmatrix} \qquad 5.6$$

There will be one and only one 1 in each column. If in one column there were only 0's, it would mean that neither a transition to any other state nor lack of change would be permissible for the state corresponding to that column. If there were more than one 1 in a column, it would mean that simultaneous transition to two or more states would be required from the state corresponding to that column. Both of these requirements would, of course, be impossible to fulfil.

When the products t and s are always written in the same order in equation 5.6, it is only necessary to write down the matrix in order to specify the transitions. This matrix, with the 0's left out, will be called a transition chart.

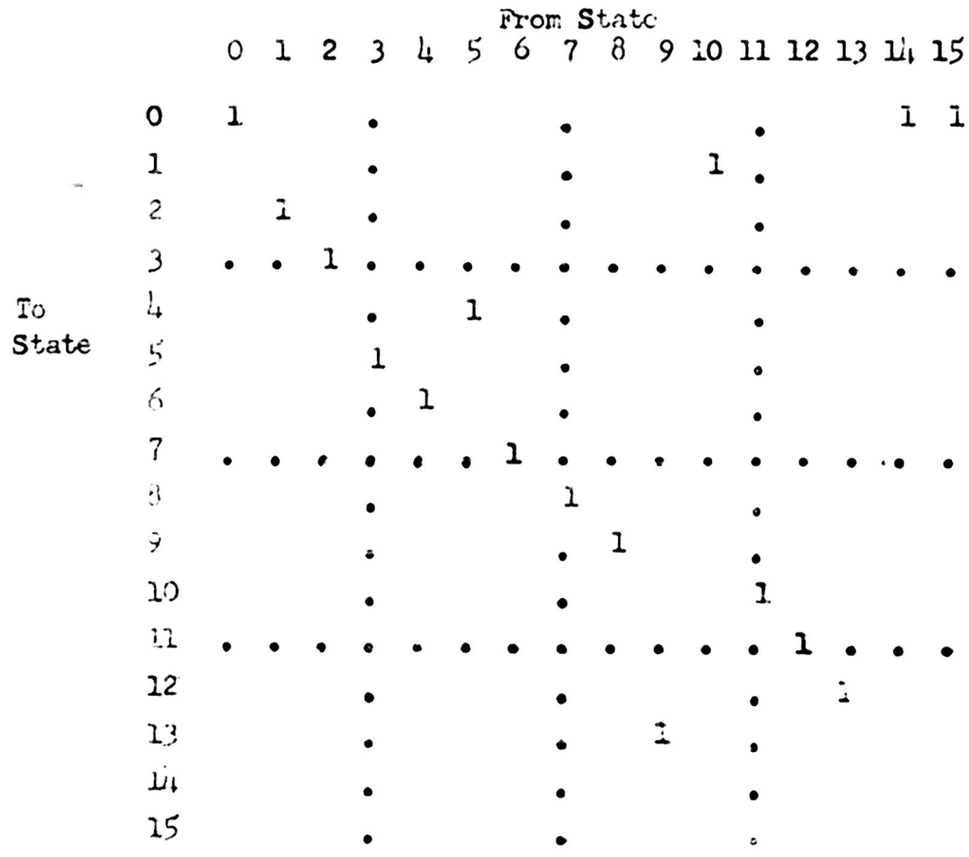


Figure 5.2 – Transition chart for thirteen-step counter

The logical structure specified in the example is a counter which proceeds cyclically through thirteen states

1-2-3-5-4-6-7-8-9-13-12-11-10-1-etc.

It will be noted that a counter, unless it counts to a whole power of two, has a number of states outside the regular counting cycle; these states will be referred to as "forbidden" states.

In the example, these states are 0, 14 and 15. If only the regular counting cycle is to be considered the transitions from the forbidden states are arbitrary and the ambiguity can be used to simplify the design. In many cases it may, however, be desirable to specify these transitions in the manner shown in the transition chart, so that an accidental transition to a forbidden state may be arrested, leaving the counter unchanged in the 0-state after the next counting pulse.

III. THE TRANSITION EQUATIONS

As the next step the variables y will be expressed explicitly as functions of the variables x. Application of equations similar to equation 2.26 to equations 5.5 will show that

$$y_0 = t_1 + t_3 + t_5 + t_7 + t_9 + t_{11} + t_{13} + t_{15}$$

$$y_1 = t_2 + t_3 + t_6 + t_7 + t_{10} + t_{11} + t_{14} + t_{15} \qquad 5.7$$

$$y_2 = t_4 + t_5 + t_6 + t_7 + t_{12} + t_{13} + t_{14} + t_{15}$$

$$y_3 = t_8 + t_9 + t_{10} + t_{11} + t_{12} + t_{13} + t_{14} + t_{15}$$

Equations 5.7 are perfectly general; they are true for any system of four binary variables for which the notation indicated in equations 5.5 has been introduced.

By the matrix equation, equation 5.6, however, each t is a function of $s_0, s_1, \ldots s_{15}$, these in turn being functions of x_0, x_1, x_2, x_3 by virtue of equation 5.5. In the example

$$\begin{aligned} y_0 &= s_{10}+s_2+s_3+s_6+s_8+s_{12}+s_9 \\ y_1 &= s_1+s_2+s_4+s_6+s_{11}+s_{12} \\ y_2 &= s_5+s_3+s_4+s_6+s_{13}+s_9 \\ y_3 &= s_7+s_8+s_{11}+s_{12}+s_{13}+s_9 \end{aligned} \qquad 5.8$$

By introduction of the variables x (see equations 5.5) equations 5.8, which are valid for the particular example under consideration, become

$$\begin{aligned} y_0 &= x_3x_2'x_1x_0'+x_3'x_2'x_1x_0'+x_3'x_2'x_1x_0+x_3'x_2x_1x_0'+x_3x_2'x_1'x_0'+ \\ &\quad x_3x_2x_1'x_0'+x_3x_2'x_1'x_0 \\ y_1 &= x_3'x_2'x_1'x_0+x_3'x_2'x_1x_0'+x_3'x_2x_1'x_0'+x_3'x_2x_1x_0+x_3x_2'x_1x_0+ \\ &\quad x_3x_2x_1'x_0' \\ y_2 &= x_3'x_2x_1'x_0+x_3'x_2'x_1x_0+x_3'x_2x_1'x_0'+x_3'x_2x_1x_0'+x_3x_2'x_1'x_0+ \\ &\quad x_3x_2'x_1'x_0 \\ y_3 &= x_3'x_2x_1x_0+x_3x_2'x_1'x_0'+x_3x_2'x_1x_0+x_3x_2x_1'x_0'+x_3x_2x_1'x_0+ \\ &\quad x_3x_2'x_1'x_0 \end{aligned} \qquad 5.9$$

A simplification of the equations will lead to the expenditure of fewer diodes in the final circuit as will be seen presently. The simplified equations are

$$\begin{aligned} y_0 &= x_3x_1'(x_0'+x_2')+x_2'x_1(x_0'+x_3')+x_0'x_1x_3' \\ y_1 &= x_0'(x_1x_3'+x_1'x_2)+x_2'x_0(x_1x_3+x_1'x_3') \\ y_2 &= x_3'(x_2(x_1'+x_0')+x_1x_2'x_0)+x_3x_1'x_0 \\ y_3 &= x_3(x_1'+x_2'x_0)+x_3'x_0x_1x_2 \end{aligned} \qquad 5.10$$

Equations of this type, giving the binary variables after the transition, as functions of the binary variables before the transition, will be called transition equations.

In the next section it will be shown how the logical design of the circuit performing these transitions can be derived from the transition equations.

IV. DERIVATION OF DESIGN EQUATIONS

Let it be assumed that separate 1-set and 0-set input connections to the toggles are to be used. This type of operation will be called split input operation. As will be seen presently the functions corresponding to reversing input connections can be derived easily from the split input functions.

Consider the variables x and y indicating the state of any one toggle in the group before and after a transition. When the toggle changes from the 0-state to the 1-state,

$$x = 0 \qquad \text{and} \qquad y = 1$$

When it changes from the 1-state to the 0-state,

$$x = 1 \qquad \text{and} \qquad y = 0$$

Then the function

$$h^1 = y(x, \ldots)x' = y(0, \ldots)x' \tag{5.11}$$

assumes the value I when the toggle changes from the 0-state to the 1-state and the function

$$h^0 = y'(x, \ldots)x = y'(1, \ldots)x \tag{5.12}$$

becomes I when the toggle changes from the 1-state to the 0-state. The functions h^0 and h^1 are then two usable input control functions since each assumes the value 1 when the input terminal associated

with it is to receive a pulse. The functions y and y' might also serve as input control functions; y and y' would provide input pulses both when the toggle is required to change state and when it is required to remain unchanged, a 1-input signal, for example, would be provided when the toggle is to remain unchanged in the 1-state. The functions h^1 and h^0 are usually simpler to form than y and y'; however, in a very few cases where

$$y(0, \ldots) \cdot y'(1, \ldots) = 0 \qquad 5.13$$

the factors x' and x may be left out in the control functions

$$h^1 = y(0, \ldots)$$
$$h^0 = y'(1, \ldots) \qquad 5.14$$

Equations 5.13 and 5.14 are in accordance with the design rule introduced in the last paragraph of Chapter IV, page 32.

When equations 5.11 and 5.12 are applied to the transition equations, 5.10, 0-set and 1-set control functions are derived. These functions will either pass (for the value 1) or stop (for the value 0) the counting pulse on its way to the input terminal of the toggle.

Let h_n^k be the input control functions, the subscript n indicating the toggle, n=0,1,2,3, and k indicating whether it controls the input to the 1-terminal or the 0-terminal, k=1 or k=0.

Equations 5.10, 5.11, and 5.12 then give for the transition sequence given on page 35

$$h_0^1 = y_0 \cdot x_0' + [x_3 x_1' \cdot (1 + x_2') + x_2' x_1 (1 + x_3') + 1 \cdot x_1 x_3'] \cdot x_0'$$
$$h_0^1 = [x_3 x_1' + x_2' x_1 + x_1 x_3'] \cdot x_0'$$
$$h_0^1 = [x_3 x_1' + x_1 (x_2' + x_3')] \cdot x_0' \qquad 5.15a$$

and similarly

$$h_1^1 = [x_0'x_2 + x_2'x_0x_3'] \cdot x_1' \qquad 5.15b$$

$$h_2^1 = [x_0(x_3'x_1 + x_3x_1')] \cdot x_2' \qquad 5.15c$$

$$h_3^1 = [x_0x_1x_2] \cdot x_3' \qquad 5.15d$$

Then the 0-input control functions are

$$h_0^0 = y_0'x_0 = [x_3x_1'(0 + x_2') + x_2'x_1(0 + x_3) + 0 \cdot x_1 \cdot x_3']'x_0$$

$$h_0^0 = [x_3x_1'x_2' + x_2'x_1x_3']'x_0$$

$$h_0^0 = [(x_3x_1' + x_1x_3')\, x_2']'x_0$$

$$h_0^0 = [(x_3 + x_1)(x_1' + x_3') + x_2] \cdot x_0$$

$$h_0^0 = [x_3'x_1' + x_3x_1 + x_2] \cdot x_0 \qquad 5.16a$$

and similarly

$$h_1^0 = [x_0(x_2 + x_3') + x_3(x_2 + x_0')] \cdot x_1 \qquad 5.16b$$

$$h_2^0 = [x_1(x_3 + x_0) + x_3x_0'] \cdot x_2 \qquad 5.16c$$

$$h_3^0 = [x_1(x_2 + x_0')] \cdot x_3 \qquad 5.16d$$

As mentioned, equations 5.13 and 5.14 indicate that there is a possibility for leaving out the x'-factors in 1-input control functions and the x-factors in the 0-input control functions appearing outside the brackets. The condition for dropping these factors is that the product of the brackets appearing in h^1 and h^0 for one toggle is identically zero (see equation 5.13). To check this possibility, consider each product.

$$[x_3x_1' + x_1 \cdot (x_1' + x_3')][x_3'x_1' + x_3x_1 + x_2] =$$

$$x_3x_1x_2' + x_3x_2x_1' + x_1x_2x_3' \not\equiv 0$$

$$[x_0'x_2 + x_2'x_0x_3'][x_0(x_2 + x_3') + x_3(x_2 + x_0')] = x_0x_3'x_2' + x_3x_2x_0' \not\equiv 0$$

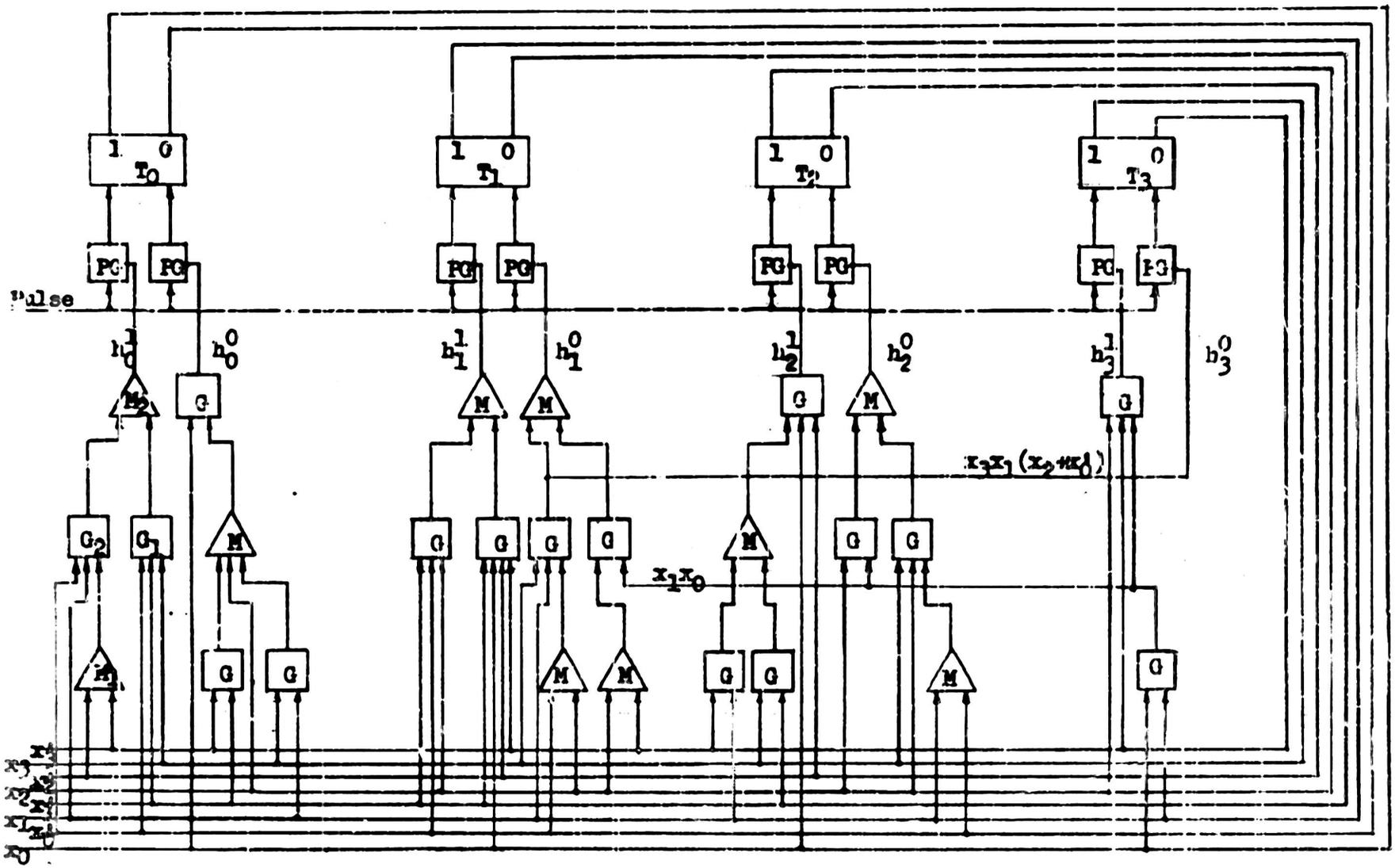


Figure 5.3 - Block diagram for thirteen-step counter using high-pass pulse gates

The output from G_2 is

$$f_3 = x_0'x_1 \cdot f_2 \qquad \text{(3 diodes)}$$

And the output from H_2

$$f_4 = f_1 + f_3 \qquad \text{(2 diodes)}$$

$$f_4 = x_0'x_1'x_3 + x_0'x_1(x_2' + x_3') = h_0^1$$

The total number of diodes needed in the formation of h_0' is

$$3 + 2 + 3 + 2 = 10 \text{ diodes}$$

Next, the simplification possible by leaving the transitions from the forbidden states arbitrary will be demonstrated.

V. THE TRANSITIONS FROM FORBIDDEN STATES

The forbidden states 0, 14, and 15, before the transition, are represented by the equations:

$$x_3'x_2'x_1'x_0' = 1$$

$$x_3x_2x_1x_0' = 1$$

$$x_3x_2x_1x_0 = 1$$

If these states are never present, then the equations

$$x_3'x_2'x_1'x_0' = 0$$

$$x_3x_2x_1x_0' = 0$$

$$x_3x_2x_1x_0 = 0$$

are always simultaneously fulfilled, or

$$x_3'x_2'x_1'x_0' + x_3x_2x_1x_0' + x_3x_2x_1x_0 = 0$$

which, after reduction, becomes

$$x_3'x_2'x_1'x_0' = 0 \qquad 5.17$$

and

$$x_3x_2x_1 = 0. \qquad 5.18$$

From this follows in particular that

$$[x_0(x_3'x_1+x_3x_1')][x_1(x_3+x_0)+x_3x_0'] = x_1x_0x_3' \neq 0$$

$$[x_0x_1x_2][x_1(x_2+x_0')] = x_1x_2x_0 \neq 0$$

In the case considered, no further simplification by this means is possible.

The block diagram for the counter, as represented by equations 5.15 and 5.16, is shown in Figure 5.3. The diagram assumes a pulse gate which passes the pulse when the control signal is 1.

The total number of diodes required for the formation of a set of switching functions can be found by counting one diode for each input to the mixers and the gates. This is the equivalent of counting one diode for each product term and one for each sum term in the switching functions (compare Chapter IV). When a logical expression appears in several places it may be advantageous to form it in only one place in order to reduce the total number of diodes needed. Simplifications of the logical equations aim at reducing the number of diodes.

In the diagram, Figure 5.3, the function $x_0 \cdot x_1$ is needed in three places, for the formation of h_3^1, h_1^0, and h_2^0. The output of one gate, located below T_3, supplies the function for all three circuits.

Consider the formation of h_0^1 in the far left of the diagram. The output function from G_1 is

$$f_1 = x_0'x_1'x_3 \qquad \text{(3 diodes)}$$

The output from M_1 is

$$f_2 = x_2' + x_3' \qquad \text{(2 diodes)}$$

$$x_3'x_2x_1 = x_2x_1$$

This simplifies h_3^1, h_1^0, h_2^0, and h_3^0 somewhat

$$h_3^1 = x_0x_1x_2x_3' = x_0x_1x_2$$

$$h_1^0 = [x_0(x_2+x_3') + x_3(x_2+x_0')]x_1$$

$$= [x_0(x_2+x_3')+x_3+x_0']x_1$$

$$h_2^0 = [x_1(x_3+x_0) + x_3x_0']x_2$$

$$= [x_1x_0+x_3x_0']x_2$$

$$h_3^0 = [x_1(x_2+x_0')]x_3 = x_1x_0'x_3$$

For unspecified transitions from states 0, 14, and 15, the input control functions will then be

$$h_0^1 = [x_3x_1'+x_1(x_2'+x_3')]x_0'$$

$$h_1^1 = [x_0'x_2+x_2'x_0x_3']x_1'$$

$$h_2^1 = [x_0(x_3'x_1+x_3x_1')]x_2' \qquad 5.19$$

$$h_3^1 = x_0x_1x_2$$

$$h_0^0 = [x_3'x_1'+x_3x_1+x_2]x_0$$

$$h_1^0 = [x_0(x_2+x_3')+x_3x_0']x_1$$

$$h_2^0 = [x_1x_0+x_3x_0']x_2 \qquad 5.20$$

$$h_3^0 = x_1x_0'x_3$$

Each of the functions 5.19 and 5.20 take on the value 1 when a pulse is to be passed by the pulse gate which it controls. For the pulse gate discussed in Chapter IV it was, however, required that the control voltage be at the lower value when the pulse was

to be passed (see page 29); this will correspond to the value 0 for the functions 5.19 and 5.20. In order to derive the control functions for this type of a pulse gate it is, therefore, necessary to derive the inverted functions of 5.19 and 5.20.

$$(h_0^1)' = (x_3'+x_1)(x_1'+x_2x_3) + x_0 = x_3'x_1' + x_1x_2x_3 + x_0$$
$$= x_3'x_1' + x_0$$

Use has been made of equation 5.18.

$$(h_1^1)' = (x_0+x_2')(x_2+x_0'+x_3)+x_1 = x_0x_2+x_0x_3+x_2'x_0'+x_2'x_3+x_1$$
$$= x_0(x_2+x_3)+x_2'(x_0'+x_3)+x_1$$

$$(h_2^1)' = x_0'+(x_3+x_1')(x_3'+x_1)+x_2 = x_0'+x_3x_1+x_3'x_1'+x_2$$

$$(h_3^1)' = x_0'+x_1'+x_2' \qquad 5.21$$

$$(h_0^0)' = (x_3+x_1)(x_3'+x_1')x_2'+x_0' = x_2'x_3x_1'+x_1x_3'x_2'+x_0'$$

$$(h_1^0)' = (x_0'+x_2'x_3)(x_3'+x_0)+x_1' = x_0'x_3'+x_2'x_3x_0+x_1'$$

$$(h_2^0)' = (x_1'+x_0')(x_3'+x_0)+x_2' = x_1'x_3'+x_3'x_0'+x_2'x_0'+x_1'$$
$$= x_3'(x_1'+x_0')+x_2'+x_0x_1'$$

$$(h_3^0)' = x_1'+x_0+x_3'$$

The block diagram is shown in Figure 5.4.

The realization of the functions by the minimum number of diodes, not the subject for investigation here, provides a rather distinct mathematical problem. As mentioned earlier, an approach to this problem for a number of simple but important cases has been made at the Harvard Computation Laboratory*. It should, however,

---

* See Synthesis of Electronic Computing and Control Circuits[3] pages 50-67

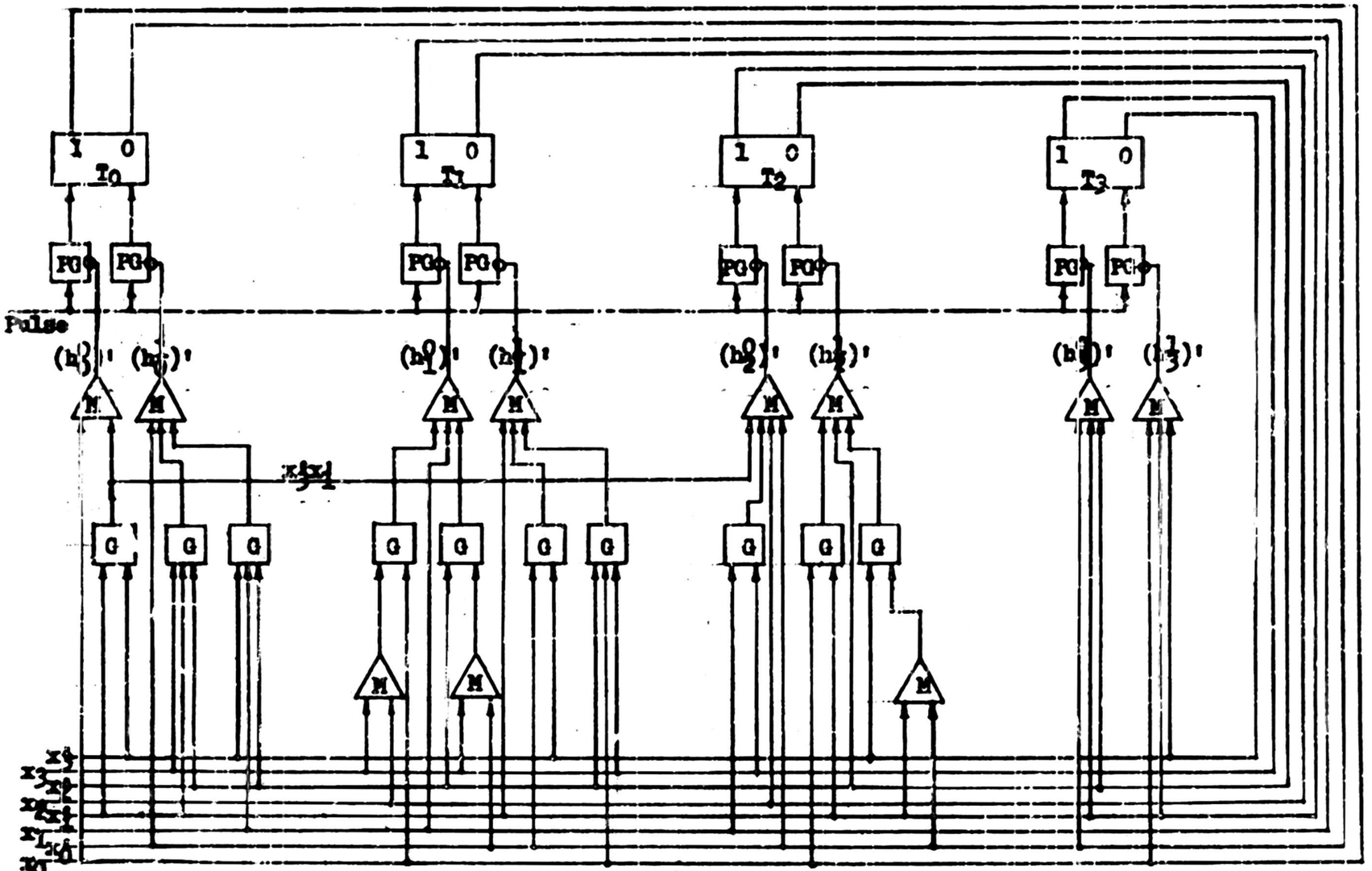


Figure 5.4 - Block diagram for simplified thirteen-step counter using low-pass pulse gates

be kept in mind that although a mathematical problem can be defined, a number of factors which are difficult to express analytically enter into the question. As is the case with equations 5.21, a solution reasonably close to the minimum can usually be arrived at by mere inspection of the equations. Factors such as the inclusion of a left-over tube-half, not taken into account in the first design approach, or the availability of increased driving power from some toggles, due to circuit features not included in the problem at hand, may make a mathematical minimum meaningless. They can, however, well be taken into account when the equations are merely inspected.

The translation from the block diagram in Figure 5.4 to the actual circuit does not present any logical problems.

It remains to discuss the use of reversing input signals. A control function for the reversing input pulses may be derived by observing that the function

$$h = y'x + yx' \qquad 5.22$$

assumes the value 1 only when the toggle changes during the transition. In order to obtain the control functions for the described diode pulse gates, h' is derived.

$$h' = yx + y'x' \qquad 5.23$$

After the derivation of reversing control functions from the transition equations, equations 5.10 in the example, the simplifications and the design can be arrived at in exactly the same manner as for split input connections.

It will be noted, by comparison with equations 5.11 and 5.12

that

$$h = h^1 \cdot h^0 \qquad 5.24$$

The example above was intended to demonstrate the method rather than provide a useful circuit. Another example of more practical nature will be given in the next section of this chapter.

VI. A CONVERTING CIRCUIT

As the next example let the problem be to design a circuit which will convert the content of four toggles between the 1-2-4-8 system and the 1-2-4-$\bar{2}$ system. Let the notation be the same as in the previous example; the variables before the transition are x_0, x_1, x_2, x_3, and after the transition y_0, y_1, y_2, y_3. The products in x are $s_1, \ldots, s_{15}$, and the products in y, $t_1, \ldots, t_{15}$.

The two number codes refer to two ways of representing a decimal digit in terms of four binary variables. The number in the code designations, for example, 1-2-4-$\bar{2}$, are the weights associated with each binary variable. The formula for the numerical value, v, corresponding to a symbol in the 1-2-4-8 system is

$$v = x_3 \cdot 8 + x_2 \cdot 4 + x_1 \cdot 2 + x_0 \cdot 1; \qquad 5.25$$

the value is that of the designation introduced earlier (see equation 5.1) for four variables. In the 1-2-4-$\bar{2}$ system, the value is

$$u = x_3 \cdot 2 + x_2 \cdot 4 + x_1 \cdot 2 + x_0 \cdot 1 \qquad 5.26$$

Only in some of the number codes is it possible to calculate the value by means of a simple formula like 5.25 and 5.26: in general it is not possible to associate each binary digit with a constant numerical weight. A table of values and the correspond-

ing combinations will, of course, describe an arbitrary number code. The tables for the two systems, 1-2-4-8 and 1-2-4-$\bar{2}$, are shown in Table 5.1.

State Designation	x_0	x_1	x_2	x_3	Numerical Value 1-2-4-8	Numerical Value 1-2-4-$\bar{2}$
0	0	0	0	0	0	0
1	1	0	0	0	1	1
2	0	1	0	0	2	2
3	1	1	0	0	3	3
4	0	0	1	0	4	4
5	1	0	1	0	5	
6	0	1	1	0	6	
7	1	1	1	0	7	
8	0	0	0	1	8	
9	1	0	0	1	9	
10	0	1	0	1		
11	1	1	0	1		5
12	0	0	1	1		6
13	1	0	1	1		7
14	0	1	1	1		8
15	1	1	1	1		9

Table 5.1 - The 1-2-4-8 and the 1-2-4-$\bar{2}$ number systems.

Now let an arbitrary number in the 1-2-4-8 system be stored in the four toggles. By application of a pulse to the circuit this combination is to be converted to the combination representing the same number in the 1-2-4-$\bar{2}$ system.

If a number in the 1-2-4-$\bar{2}$ system is stored in the toggles, application of a pulse to the same point must cause the combination to change to the equivalent 1-2-4-8 combination. Application of two pulses leaves the toggles unchanged. These requirements specify a transition from each state with the exception of state Number 10. Let the transition from this state be left arbitrary, that is, let s_{10} be added or omitted in the equations, whichever leaves them in the simpler form.

The transition chart for the conversion is shown in Figure 5.5.

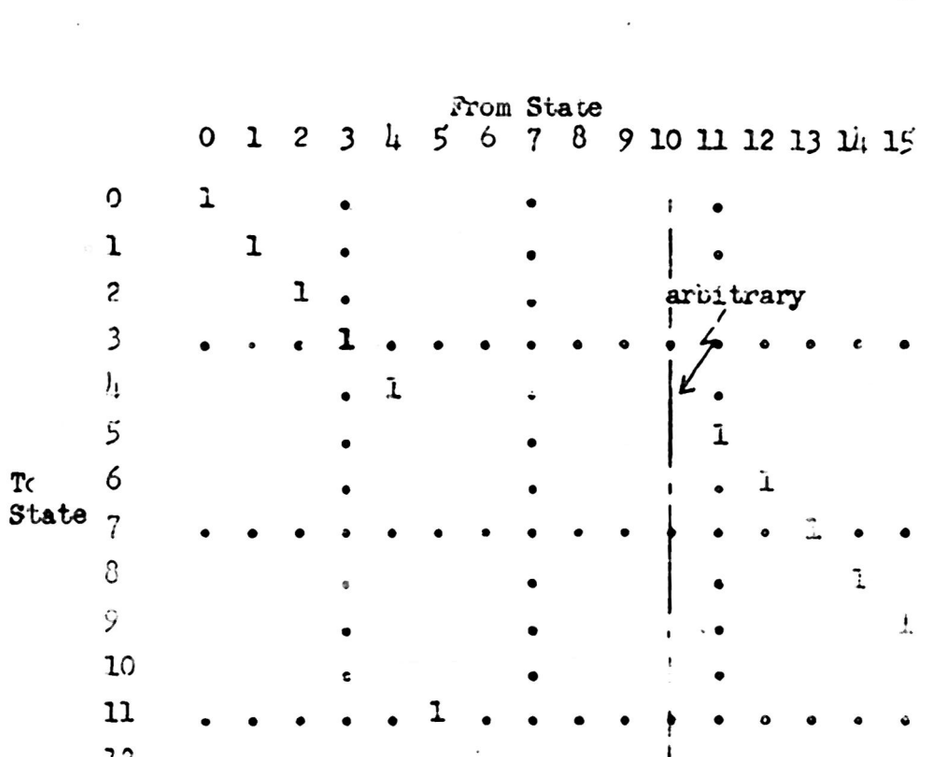


Figure 5.5 – Transition chart for conversion between 1-2-4-8 system and 1-2-4-$\bar{2}$ system

The procedure is now the same as in the previous example. The transition equations are obtained from the transition chart by means of equations 5.7, repeated here,

$$y_0 = t_1 + t_3 + t_5 + t_7 + t_9 + t_{11} + t_{13} + t_{15}$$
$$y_1 = t_2 + t_3 + t_5 + t_7 + t_{10} + t_{11} + t_{14} + t_{15}$$
$$y_2 = t_4 + t_5 + t_6 + t_7 + t_{12} + t_{13} + t_{14} + t_{15}$$
$$y_8 = t_8 + t_9 + t_{10} + t_{11} + t_{12} + t_{13} + t_{14} + t_{15};$$
$$y_0 = s_1 + s_2 + s_{11} + s_{13} + s_{15} + s_5 + s_7 + s_9$$
$$y_1 = s_2 + s_3 + s_{12} + s_{13} + [s_{10}] + s_5 + s_8 + s_9 \qquad 5.27$$
$$y_2 = s_4 + s_{11} + s_{12} + s_{13} + s_6 + s_7 + s_8 + s_9$$
$$y_3 = s_{14} + s_{15} + [s_{10}] + s_5 + s_6 + s_7 + s_8 + s_9$$

It follows that

$$y_0 = x_0[x_3'x_2'x_1' + x_3'x_2'x_1 + x_3'x_2x_1' + x_3'x_2x_1 + x_3x_2'x_1' + x_3x_2'x_1 + x_3x_2x_1' + x_3x_2x_1] = x_0$$

$$y_1 = x_3'x_2'x_1(x_0+x_0') + x_3x_2x_1'(x_0+x_0') + x_3'x_2x_1'x_0 + x_3x_2'x_1'(x_0'+x_0),$$

where s_{10} has been left out.

$$y_1 = x_3'x_2'x_1 + x_3x_1'(x_2+x_2') + x_3'x_2x_1'x_0$$
$$y_1 = x_3'x_2'x_1 + x_1'(x_3+x_3'x_2x_0)$$
$$y_1 = x_3'x_2'x_1 + x_1'(x_3+x_2x_0)$$

where equation 2.24 has been applied

$$y_2 = x_3'x_2x_1'x_0' + x_3x_2'x_1x_0 + x_3x_2x_1'(x_0+x_0') + x_3'x_2x_1(x_0+x_0') + x_3x_2'x_1'(x_0+x_0')$$
$$y_2 = x_3x_1'(x_2+x_2') + x_3'x_2x_1 + x_3'x_2x_1'x_0 + x_3x_2'x_1x_0$$
$$y_2 = x_3(x_1'+x_1x_2'x_0) + x_3'x_2(x_1+x_1'x_0')$$
$$y_2 = x_3(x_1'+x_2'x_0) + x_3'x_2(x_1+x_0')$$

$$y_2 = x_3x_1' + x_3x_2'x_0 + x_3'x_2(x_1 + x_0') + x_3x_2'x_1x_0'$$

where s_{10} has been added;

but

$$x_3x_1' + x_3x_2'x_1x_0' = x_3(x_1' + x_1x_2'x_0') = x_3x_1' + x_3x_2'x_0',$$

giving

$$y_2 = x_3x_1' + x_3x_2'(x_0 + x_0') + x_3'x_2(x_1 + x_0')$$

$$y_2 = x_3(x_1' + x_2') + x_3'x_2(x_1 + x_0')$$

$$y_3 = x_3x_2x_1(x_0 + x_0') + x_3'x_2x_1'x_0 + x_3'x_2x_1(x_0 + x_0') + x_3x_2'x_1'(x_0 + x_0'),$$

where s_{10} has been left out;

$$y_3 = x_2x_1(x_3 + x_3') + x_3x_2'x_1' + x_3'x_2x_1'x_0$$

$$y_3 = x_2(x_1 + x_1'x_3'x_0) + x_3x_2'x_1'$$

$$y_3 = x_2(x_1 + x_3'x_0) + x_3x_2'x_1'$$

Written together the transition equations are

$$\begin{aligned} y_0 &= x_0 \\ y_1 &= x_3'x_2'x_1 + x_1'(x_3 + x_2x_0) \\ y_2 &= x_3(x_1' + x_2') + x_3'x_2(x_1 + x_0') \\ y_3 &= x_2(x_1 + x_3'x_0) + x_3x_2'x_1' \end{aligned} \qquad 5.28$$

From equations 5.28 the split input control functions are derived in accordance with equations 5.11 and 5.12. The 1-input control functions, assuming the value 1 when a pulse is to be passed, are h_0^1, h_1^1, h_2^1, and h_3^1;

$$\begin{aligned} h_0^1 &= 0 \cdot x_0' = 0 \\ h_1^1 &= [x_3'x_2' \cdot 0 + 1 \cdot (x_3 + x_2x_0)]\, x_1' = [x_3 + x_2x_0]x_1' \\ h_2^1 &= [x_3(x_1' + 1) + x_3' \cdot 0 \cdot (x_1 + x_0')]\, x_2' = [x_3]x_2' \\ h_3^1 &= [x_2(x_1 + 1 \cdot x_0) + 0 \cdot x_2' \cdot x_1']\, x_3' = [x_2(x_1 + x_0)]x_3' \end{aligned} \qquad 5.29$$

The 0-input control functions, assuming the value 1 when a pulse is to be passed, are

$$h_0^0 = [1]'x_0 = 0$$
$$h_1^0 = [x_3'x_2' + 0\cdot(x_3+x_2x_0)]'x_1 = [x_3+x_2]x_1$$
$$h_2^0 = [x_3(x_1'+0)+x_3'\cdot 1\cdot(x_1+x_0')]'x_2 = \qquad 5.30$$
$$[(x_3'+x_1)(x_3+x_1'x_0)]\,x_2 = [x_3x_1+x_0x_3'x_1']x_2$$
$$h_3^0 = [x_2(x_1+0\cdot x_0)+1\cdot x_2'x_1']'x_3 = [(x_2'+x_1')(x_2+x_1)]x_3 =$$
$$[x_1'x_2+x_2'x_1]\,x_3$$

Application of equations 5.13 and 5.14 does not simplify these equations further.

The following reversing input control functions h_0, h_1, h_2 and h_3 are derived in accordance with equation 5.24.

$$h_0 = 0+0 = 0$$
$$h_1 = (x_3+x_2x_0)x_1'+(x_3+x_2)x_1 = x_3+x_2(x_0x_1'+x_1) = x_3+x_2(x_0+x_1)$$
$$h_2 = x_3x_2'+(x_3x_1+x_0x_3'x_1')x_2 = x_3(x_2'+x_2x_1)+x_0x_3'x_1'x_2 = \qquad 5.31$$
$$x_3(x_2'+x_1)+x_0x_3'x_1'x_2$$
$$h_3 = x_2(x_1+x_0)x_3'+(x_1'x_2+x_2'x_1)x_3$$

From equations 5.29 and 5.30 a circuit employing split input connections can be drawn immediately. If reversing input connections are to be used, a circuit can be drawn from equations 5.31.

If the pulse gate described earlier is to be used, each input control function must be inverted in order to provide a low voltage for the pass position of the gate.

If only one way conversion is to be performed, strong simplifications, due to increased redundancy, are possible. Let the circuit be limited to conversion from the 1-2-4-8 system to the 1-2-4-7 system.

In that case the following states are forbidden before the transition

10, 11, 12, 13, 14, 15

meaning that the following equations in x are always true

$$x_3x_2'x_1x_0' = 0 \qquad x_3x_2x_1'x_0 = 0$$
$$x_3x_2'x_1x_0 = 0 \qquad x_3x_2x_1x_0' = 0 \qquad 5.32$$
$$x_3x_2x_1'x_0' = 0 \qquad x_3x_2x_1x_0 = 0,$$

or

$$x_3x_2'x_1(x_0'+x_0)+x_3x_2x_1'(x_0'+x_0)+x_3x_2x_1(x_0'+x_0) = 0$$
$$x_3x_2'x_1+x_3x_2(x_1+x_1') = 0$$
$$x_3(x_1x_2'+x_2) = 0$$
$$x_3(x_1+x_2) = 0 \qquad 5.33$$

or

$$x_3x_1 = 0 \qquad \text{and} \qquad x_3x_2 = 0$$

Equation 5.33 is the redundancy equation characteristic of the 1-2-4-8 system. The following relations are useful when the redundancy equation is applied

$$x_3'x_1 = x_3'x_1+x_3x_1 = x_1 \qquad 5.34$$
$$x_3'x_2 = x_2 \qquad 5.35$$
$$x_3x_2' = x_3 \qquad 5.36$$
$$x_3x_1' = x_3 \qquad 5.37$$

Under these conditions, equations 5.29, 5.30, and 5.31 are simplified considerably.

$$h_0^1 = 0 \qquad h_0^0 = 0$$
$$h_1^1 = x_3+x_2x_0x_1' \qquad h_1^0 = x_2x_1$$
$$h_2^1 = x_3 \qquad h_2^0 = x_0x_1'x_2 \qquad 5.38$$
$$h_3^1 = x_2(x_1+x_0) \qquad h_3^0 = 0$$

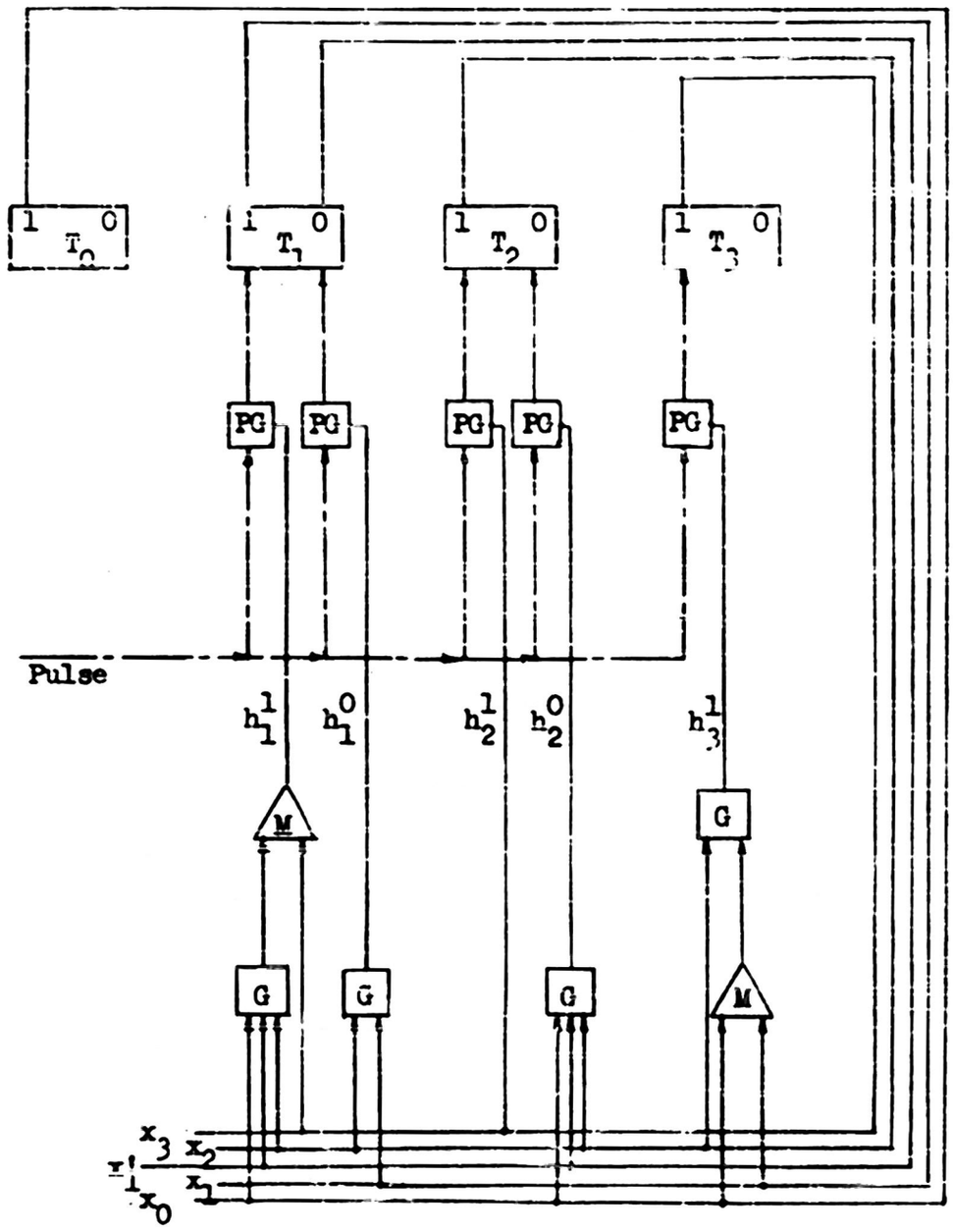


Figure 5.6 - Block diagram for 1-2-4-8 to 1-2-4-$\bar{2}$ system converter using split input connections

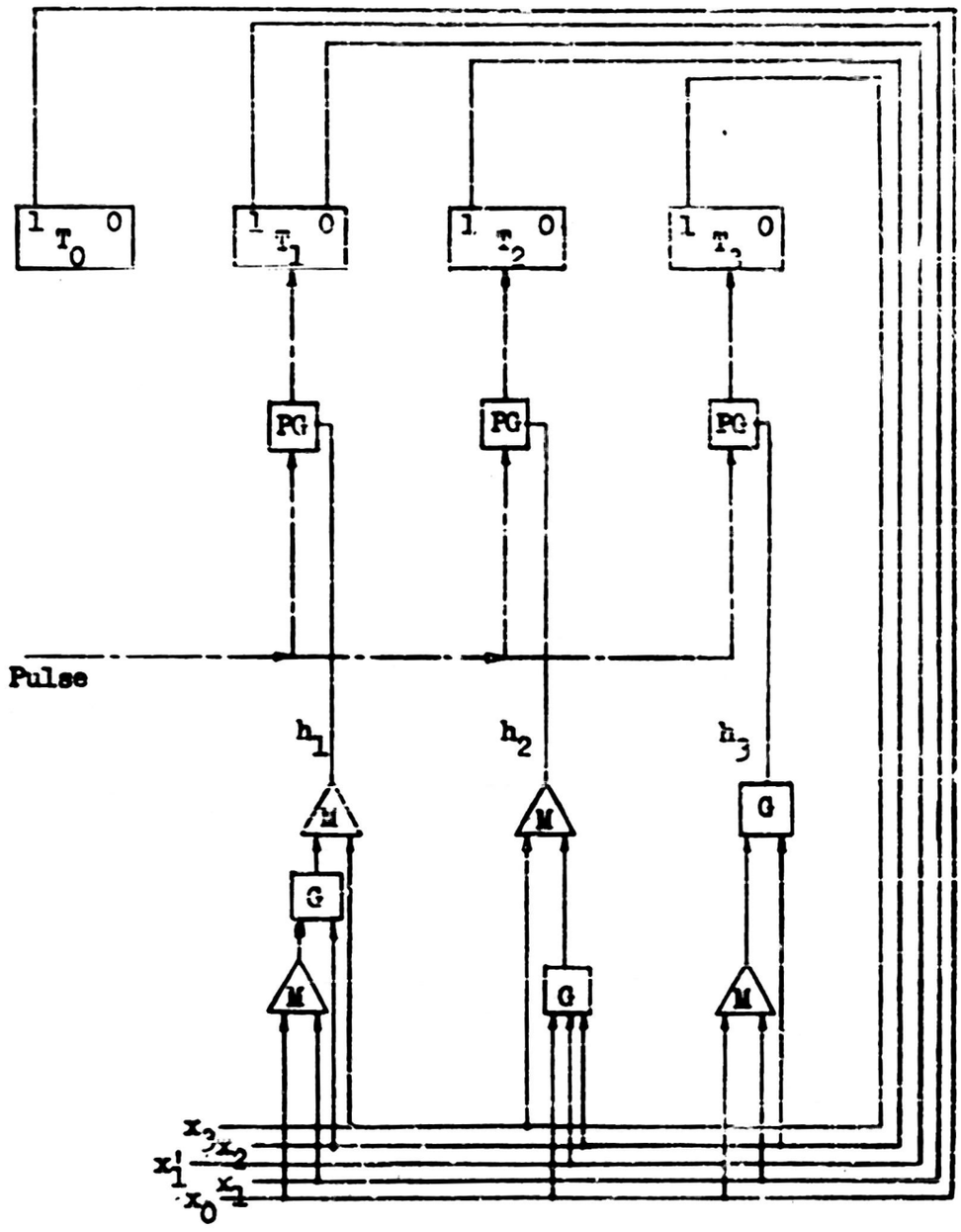


Figure 5.7 - Block diagram for 1-2-4-8 to 1-2-4-$\bar{8}$ system converter using reversing input connections

$$h_0 = 0$$

$$h_1 = x_3 + x_2(x_0 + x_1) \qquad 5.39$$

$$h_2 = x_3 + x_0 x_1 + x_2$$

$$h_3 = x_2(x_1 + x_0)$$

The block diagrams corresponding to equations 5.36 and 5.39 are shown in Figures 5.6 and 5.7

SUMMARY OF THE CHAPTER.

It is characteristic of the circuits discussed that application of a pulse causes a transition from one state to another to take place in a toggle group.

In the case of a counter, a sequence of states is repeated cyclically by continuous application of pulses; certain states, the forbidden states, never occur in the regular counting period. In the case of a converting circuit, the content of the toggle group is changed from one value to another.

In all cases the transitions can be specified by means of a transition chart, which offers a complete description of the logical response to a pulse input. If some transitions are arbitrary, this condition can be introduced by means of a set of redundancy equations.

From the transition chart, a set of transition equations can be derived. The block diagram follows by some simple algebraic operations on the transition equations, whether split input connections or complementing input connections to the toggles are to be used. Possible redundancy equations can be used in the course of the synthesis for purposes of simplification. The block diagram can be interpreted directly in terms of actual circuitry by means of the techniques described in Chapter IV.

The connection between the logical design equations and the circuit design is so close that it is usually not necessary to draw a circuit diagram or even a block diagram in order to compare

different logical possibilities.

The stress is on the logical design. For this reason, no actual circuits are drawn. It is evident, however, from Chapter IV that all logical operations are realizable by a term-by-term interpretation of the equations.

CHAPTER VI

THE TRANSLATION BETWEEN LOGICAL AND ARITHMETIC OPERATIONS

When two decimal digits, each represented by four binary variables, are to be added in a digital computer, it is necessary to translate the operation of addition of the two decimal digits into a logical operation on the binary variables which constitute the two digits. It has been mentioned earlier that, over and above the numerical interpretation, an arithmetic operation, as well as any other filtering operation, can be characterised by the distribution of its output symbols over all possible input symbols. As an example, consider the addition of two decimal digits. The addition table is shown in Table 6.1.

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Table 6.1 - Addition of two decimal digits

It is usually required that a special symbol, a binary variable, is set aside for the carry in order to make it available, without further filtering, at the addition of the next two decimal digits. When the carry is left out in the addition table, it appears that there are ten possible output symbols, the digits 0 through 9, and one hundred input symbols, all combinations of two decimal digits. Each output symbol occurs for any of a group of ten input symbols; the output symbols have a perfectly even distribution over the input symbols.

If a carry from the preceding sum in the addition of two multi-digit numbers is taken into account, there will be two hundred possible input symbols and, again considering the carry to the following addition separately, ten possible output symbols. Each output symbol now occurs for any of a group of twenty input symbols.

In all cases it is true that any logical circuitry which realizes the above distributions of output symbols over the input symbols is capable of adding two decimal digits. The code comes about by interpretation of each symbol as a numerical quantity, a decimal digit, in accordance with the addition table.

In the following, the addition of two decimal digits without a carry from the preceding addition will be considered. The symbols to be dealt with are associated with groups of binary variables.

The input is naturally thought of as two channels each having ten possible symbols and the output as a single channel also having

ten possible symbols.

The binary variables corresponding to each of the channels must be at least four in number since otherwise an insufficient number of symbols would be available. Let the two input channels have the binary variables x_0, x_1, x_2, x_3, and y_0, y_1, y_2, y_3, and the output channel the binary variables z_0, z_1, z_2, z_3.

As was discussed in Chapter V, the occurrence of a particular symbol corresponds to equating a product, for example, $x_0 \cdot x_1' \cdot x_2 \cdot x_3$, with 1.

Let the products in x be termed s; the products in y, t; and the products in z, r; in the manner indicated in Chapter V, page 37. Furthermore, let the superscript on s, t, and r, indicate the digit, the numerical value, with which a particular product is associated.

If

$$r^9 = z_0 \cdot z_1 \cdot z_2' z_3$$

then the value combination

$$r^9 = z_0 \cdot z_1 z_2' z_3 = 1$$

or $$z_0 = 1;\ z_1 = 1;\ z_2 = 0;\ z_3 = 1$$

has been associated with the digit 9.

Now if

$$s^3 = 1 \qquad \text{and} \qquad t^4 = 1$$

then

$$r^7 = 1$$

since

$$3+4 = 7$$

this can be expressed

$$r^7 = s^3 t^4 + \cdots$$

where the presence of other terms is indicated because the sum, 7, occurs for other input combinations. When equations of this type are applied to all permitted input and output signals considered, the following equations result:

$$\begin{aligned}
r^0 &= s^0t^0+s^1t^9+s^2t^8+s^3t^7+s^4t^6+s^5t^5+s^6t^4+s^7t^3+s^8t^2+s^9t^1\\
r^1 &= s^0t^1+s^1t^0+s^2t^9+s^3t^8+s^4t^7+s^5t^6+s^6t^5+s^7t^4+s^8t^3+s^9t^2\\
r^2 &= s^0t^2+s^1t^1+s^2t^0+s^3t^9+s^4t^8+s^5t^7+s^6t^6+s^7t^5+s^8t^4+s^9t^3\\
r^3 &= s^0t^3+s^1t^2+s^2t^1+s^3t^0+s^4t^9+s^5t^8+s^6t^7+s^7t^6+s^8t^5+s^9t^4\\
r^4 &= s^0t^4+s^1t^3+s^2t^2+s^3t^1+s^4t^0+s^5t^9+s^6t^8+s^7t^7+s^8t^6+s^9t^5\\
r^5 &= s^0t^5+s^1t^4+s^2t^3+s^3t^2+s^4t^1+s^5t^0+s^6t^9+s^7t^8+s^8t^7+s^9t^6\\
r^6 &= s^0t^6+s^1t^5+s^2t^4+s^3t^3+s^4t^2+s^5t^1+s^6t^0+s^7t^9+s^8t^8+s^9t^7 \qquad 6.1\\
r^7 &= s^0t^7+s^1t^6+s^2t^5+s^3t^4+s^4t^3+s^5t^2+s^6t^1+s^7t^0+s^8t^9+s^9t^8\\
r^8 &= s^0t^8+s^1t^7+s^2t^6+s^3t^5+s^4t^4+s^5t^3+s^6t^2+s^7t^1+s^8t^0+s^9t^9\\
r^9 &= s^0t^9+s^1t^8+s^2t^7+s^3t^6+s^4t^5+s^5t^4+s^6t^3+s^7t^2+s^8t^1+s^9t^0
\end{aligned}$$

The requirements for addition do not, of course, specify when the six surplus symbols, r^{10}, r^{11}, r^{12}, r^{13}, r^{14}, and r^{15}, are to occur; it only specifies that they are not to occur when both t and s represent legitimate, non-forbidden symbols. One of the forbidden output symbols may, for example, be

$$r^{11} = s^9t^{13}+s^{11}t^0+s^{13}t^{15}$$

because

$$r^{11} = s^9 \cdot 0 + 0 \cdot t^0 + 0 \cdot 0 = 0$$

under normal operating conditions. This means that the symbol corresponding to $r^{11} = 1$ never occurs in normal operation. In

other words, a forbidden output symbol may only occur when either s or t are in the forbidden range. When

$$s^{10}+s^{11}+s^{12}+s^{13}+s^{14}+s^{15} = 0 \qquad 6.2$$

and

$$t^{10}+t^{11}+t^{12}+t^{13}+t^{14}+t^{15} = 0 \qquad 6.3$$

then

$$r^{10}+r^{11}+r^{12}+r^{13}+r^{14}+r^{15} = 0 \qquad 6.4$$

Equations 6.2, 6.3, and 6.4 are the redundancy equations for the system. If the result of input symbols in the forbidden range is to be left arbitrary, these equations may be used to simplify the logical equations in the manner demonstrated in Chapter V.

Equations 6.1 can be written more compactly as

$$r^k = \sum_{i=0}^{9} s^i t^{k-i} \qquad 6.5$$

where

$$t^{k-i} = t^{10+k-i} \qquad \text{if} \qquad k-i<0$$

and

$$0 \leq k \leq 9$$

Equation 6.5 has a simple arithmetic content; it states that the output symbol associated with the digit value k occurs, that is r^k takes on the value 1, only when two input symbols associated with digits having the sum k or 10+k occur, that is, when $s^i=1$ and $t^{k-i}=1$. It is the fact that equation 6.5 also gives the logical relationship between the binary variables representing these symbols, which makes it significant. In other words, equation 6.5 expresses both the arithmetic operation on numbers and its execution as a logical operation on binary variables.

In order to arrive at design equations, it is necessary to derive z_0, z_1, z_2, and z_3 as functions of x_0, x_1, x_2, x_3, and y_0, y_1, y_2, y_3. When a code is chosen, s and t are readily expressed as products in x and y. The dependent variables, z, can be arrived at by equations similar to equations 5.7. When z_0, z_1, z_2, and z_3 are derived, the rest of the design may follow the principles discussed in Chapter V.

Equation 6.5 can easily be generalized to cover other operations than addition. The products r, s, and t, can include any number of variables sufficient to specify the total number of output and input symbols in the operation considered. In the case of multiplication (see Chapter III), for example, where 37 output symbols are necessary, at least six variables, z_0, z_1, z_2, z_3, z_4, z_5, would be necessary in r. In all cases the group of input symbols which yield the same output symbol appear in the expression for that output symbol; or

$$r^k = \sum_{f(i,p)=k} s^i t^p \qquad 6.6$$

where the operation in question is indicated by

$$f(i,p) = k$$

In the case of multiplication, equation 6.6 becomes

$$r^k = \sum_{k=i\cdot p} s^i t^p$$

For example

$$r^{16} = s^1 \cdot t^{16} + s^2 t^8 + s^4 t^4 + s^8 \cdot t^2 + s^{16} \cdot t^1$$

In the equations the code has been left arbitrary. The choice

of code can, at the present, only in principle be used as an optimizing parameter.

The codes for r, s, and t are, therefore, usually chosen as one of the conventional codes, 1-2-4-8 or 1-2-4-2, for example.

When two multi-digit numbers are to be added, the operation consists in more than a digit by digit and carry addition. If the digits appear serially, they must be shifted into the adder one by one and the result must be shifted out; provision must be made for handling signs, in some systems of operation the nines complement of each decimal digit must be derived. Although it is conceivable to perform all these operations in one step, it will frequently lead to needless complexity with a small overall gain in speed.

In digital computers employing a memory with long access time, for example a magnetic drum memory, it will usually pay to break even the addition of the individual decimal digits down into several steps.

Figure 6.1 - Add-correct operation

In an add-correct operation, there are at least two steps. First, the sum is derived in an arbitrary code, whichever leads to simple circuitry; second, the code is changed to the specified

code. This is indicated in Figure 6.1.

Step Number 1 may be broken down into further steps. As an example, consider addition of two decimal digits in the 1-2-4-8 system. Let x_0 and y_0, x_1 and y_1, x_2 and y_2, and x_3 and y_3, be added in separate binary adders producing a sum and a binary carry for each stage, that is, for each binary pair, (index 0, 1, 2 and 3). Let the carries be delayed in their passage from stage to stage. The process of addition will now be the following

1. The individual pairs, x, y, are added, producing sums and binary carries.
2. The carries from (1) are added to the sums from (1) giving rise to new binary sums and possibly new binary carries from stages 1, 2, and 3.
3. The carries from (2) are added to the sums from (2) giving rise to new sums and possibly new carries from stages 2 and 3.
4. The carries from (3) are added to the sums from (3) giving rise to new sums and possibly a carry from stage 3.

As before, the decimal carry will not be taken into account. So far, the operation has produced the binary sum of two binary numbers. This sum is correct if it is nine or less. If it is above nine, the sum must be corrected. The last step, therefore, is

5. If the sum is more than 9, add 6.

If, for example, the result after step Number 4 is 13, the

binary variables z_0, z_1, z_2 and z_3, will have the values

$$z_3 = 1; \qquad z_2 = 1; \qquad z_1 = 0; \qquad z_0 = 1$$

The correction step Number 5 adds 6, making the sum 19, producing a carry, and the number 19-16=3, which is the correct last digit in the sum 13.

The above type of operation is a well-known example of step-by-step addition. In the Appendix, where the discussed design methods will be tied together in a single example, another way of breaking down the addition process will be demonstrated.

The most important example of step-by-step operation is, perhaps, digit-by-digit addition of multi-digit numbers and serial operation in general. In all cases, the purpose of step-by-step operation is to trade speed for simplicity.

CHAPTER VII

SUMMARY AND RECOMMENDATIONS

The paper is exclusively concerned with the theory and practice of translation of arithmetic operations on numbers into logical operations on binary variables.

Digital computers have, for some time, been recognized as devices capable of storing, moving, interpreting, and modifying information. This point of view forms the basis for the paper.

A suitable mathematical information concept is found in the mathematical theory of communication. It is shown how this concept can be applied both to the arithmetic operations on numbers, which the arithmetic unit of a digital computer is required to perform, and to the logical operations which digital computer elements are capable of performing.

From the analysis it appears that when an arithmetic operation is viewed in the light of the mathematical concept of information, its numerical content becomes a matter of secondary importance. Of primary importance is the distribution of symbols in the operation. Any physical process capable of bringing about a particular distribution of output symbols over input symbols may be used as an arithmetic element for that particular operation.

When the information concept is applied to the logical operations available in digital computers use is made of the symbolic-logic representation of these operations. Through the work of the staff of the Harvard Computation Laboratory, the use of Boolean

Algebra has reached a high degree of development and, although some problems still remain unsolved, it is adequate for the usual practical design problems. A further step in the development is taken, in this paper, by introduction of the information concept in the application of Boolean Algebra. It is shown how the operations of Boolean Algebra represent filtering processes, that is processes that select certain information just as a band-pass filter in classical communication theory selects certain frequencies. The concept of a discrete information filter is due to Shannon. The fact that the functions of Boolean Algebra, and of equivalent symbolic-logic systems, represent the mechanics of such filters, is new.

The theory of communications deals with symbols, but is not concerned with the meaning of these symbols; it speaks about the probability of their occurrence and it measures their capacity to convey information without saying what information.

The symbols dealt with in Boolean Algebra are combinations of values of binary variables. The design method developed in this paper, although not vastly superior to earlier "truth table" methods, has the advantage that at the stage of formulation of a problem it deals directly with these Boolean symbols. The application of the concept of information is not considered an end in itself but rather a first step toward broader notions of computation and computing machines. It allows a formulation of the arithmetic requirements which is directly interpretable in terms of the logical symbols.

The paper also gives an account of some logical design

practices used in connection with work on the California Digital Computer. Such design practices cannot always follow a system of rigid rules; they can best be described by means of examples. They represent the methods which the writer has developed to solve practical design problems. These design practices, which are the result of independent work, differ sufficiently from methods developed elsewhere, at Harvard for example, to justify their description here.

From the results presented of the application of the concept of information to Boolean Algebra, two future directions of development seem to hold some promise.

1. The development of a theory of information centered around two-valued logic.

Such a theory would deal with information in its most fundamental terms such as have been recognized by pure logic. It might prove of value to both communication engineering and computer engineering.

2. The application of digital computer techniques to problems of communication over discrete channels.

In digital computers the filtering of information attains a highly developed form. Communication engineering could well take advantage of its advanced techniques.

Application of more powerful algebraic minimizing techniques, at present not available, might use the general equations for addition (Chapter VI) or multiplication as a point of departure for a determination of the optimum number code. It is felt, however,

that a solution of such a problem, although interesting from the academic point of view, would have few practical consequences.

As a whole it is thought that although many problems in digital computer design can be formalized and successfully solved by means of Boolean Algebra, the exaggerated dependence upon a symbolic procedure in design may be misleading. The use of switching functions with six or more variables becomes so complicated a matter that it is usually necessary to break a problem of this size into pieces, each containing fewer variables. If a minimum is arrived at for such a part of a larger problem, it does not necessarily imply that a solution even close to the minimum for the whole has been found. Solutions arrived at by formal means should, therefore, always be evaluated critically. It usually pays to provide alternative solutions for a given problem and make a comparison on the basis of expenditure of equipment, simplicity of operation, reliability, etc.

If used subordinately in the course of design, however, Boolean Algebra provides an efficient tool and a compact notation of great value to the designer. For this purpose, the application of symbolic logic to computer circuits has reached an adequate degree of development.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. George Boole, The Laws of Thought, Open Court Publishing Company, 1916, 1940.

2. C.E.Shannon, A Symbolic Analysis of Relay and Switching Circuits, AIEE Transactions, volume 57, pages 713-723, 1938.

3. The Staff of the Harvard Computation Laboratory, Synthesis of Electronic Computing and Control Circuits, Harvard University Press, 1951.

4. C.E.Shannon, The Synthesis of 2-terminal Switching Circuits, Bell System Technical Journal, volume 28, pages 59-98, (Jan.1949).

5. C.E.Shannon, The Mathematical Theory of Communication, The University of Illinois Press, 1949.

6. N. Wiener, Cybernetics, John Wiley and Sons, 1949.
Chapter III, Time Series, Information, and Communication.

7. W.H.Eccles and F.W.Jordan, A Trigger Relay Utilising Three Electrode Vacuum Tubes, Radio Review, volume 1, pages 143-146, (December 1919).

APPENDIX

APPENDIX

A BINARY CODED DECIMAL ADDER

The adder to be described operates in the 1-2-4-8 system. It is part of a computer* which is serial in the decimal digits but in which the binary digits making up each decimal digit appear in parallel in four separate channels. The add-circuit and a circuit for deriving the nines complement will be described. It will be shown how these operations, shifting, and carry addition are coordinated to form a system which will add or subtract two multi-digit numbers with arbitrary signs.

I. THE OVER-ALL OPERATION

At the beginning of the operation the two ten digit decimal numbers are stored in two registers, called the A-register and the D-register (see Figure 8.1). After the operation, the sum, with the correct sign, will occupy the A-register.

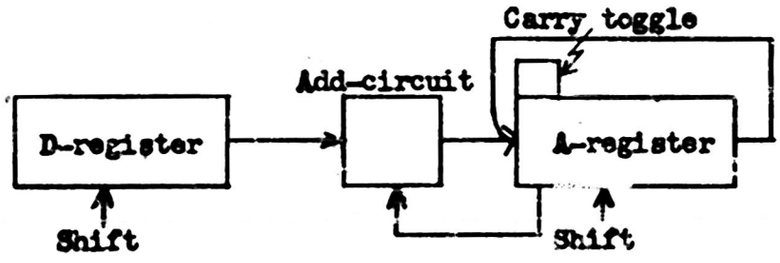


Figure 8.1 - Over-all operation of binary coded decimal adder

*The California Digital Computer, "THE CALDIC", under construction at University of California in Berkeley

Each decimal digit occupies four toggles, T_1, T_2, T_4, T_8, the index now indicating the weight associated with each toggle. In each register there is a column or group of four toggles for each decimal digit and a column of four for the sign. The registers are capable of shifting; every time a pulse is received on the shift terminal each decimal digit is shifted one place to the right; the far right digit, in the 10 column, may be shifted into another register; the far left column, the sign column may either receive a decimal digit from the outside, possibly from the 10-column, or it may be set to 0 in the process of shifting. The sign column, thus, not only stores the sign (a single toggle would suffice for that), but it also allows the register to receive information, serving as a transit station for the decimal digits as they are shifted into the register from the left. The column to the right of the sign column is the 1-column, the next one is the 2-column, the 3-column, etc., ending in the 10-column in the far right end of the register. The number is oriented in such a way that the most significant digit occupies the 1-column, the least significant digit the 10-column. The 10-column serves as a transit station for outgoing information. A "1" in the sign column indicates a minus, a "0" indicates a plus.

The addition process will be the following: The digit in the 10-column of the A-register is shifted into the sign column of the A-register, the rest of the register being shifted one step to the right in the regular manner at the same time; this operation is known as "circulation". The least significant digit of the number in the

A-register now occupies the sign-column of that register. In the next step the information in the 10-column of the D-register and in the sign-column of the A-register is combined in the add-circuit changing the content of the A-register sign-column, also known as the adder-column, to the sum of the two digits and possibly producing a decimal carry in a separate carry toggle in that column. At the same time the number in the D-register is shifted one place to the right. Next, the A-register is circulated one place putting the next to the least significant digit into the sign-column and shifting the result of the last digital addition into the 1-column. The two registers are now ready for another digital addition which this time must incorporate addition of a decimal carry. By repetition of this process the sum of the two numbers will eventually occupy the correct columns of the A-register. The process includes addition of the sign digits as will be discussed later.

When two numbers with the same sign are to be subtracted or when two numbers with opposite signs are to be added, the nines complement of each digit as it appears in the A-register 10-column is formed and a carry is stored in the carry toggle before the addition begins. If the decimal points are thought of as being immediately to the left of the 1-columns before the addition starts, subtraction is in effect performed by replacing the number in the A-register, A, by 1-A. Depending on the relative magnitudes and signs, the result may have to be decomplemented after subtraction.

Next, an add-circuit and a circuit for derivation of the nines complement will be designed to suit the approximate require-

ments given above.

II. DERIVATION OF THE NINES COMPLEMENT IN THE 1-2-4-8 SYSTEM

The digit to be complemented with respect to 9 is stored in a toggle group T_1, T_2, T_4, T_8. The circuit is like the converting circuit discussed in Chapter V. By application of a pulse the content of T_1, T_2, T_4, T_8, must be changed to the combination representing the nines complement of whatever digit was stored before application of the pulse.

The transition chart shown in Figure 8.2 applies.

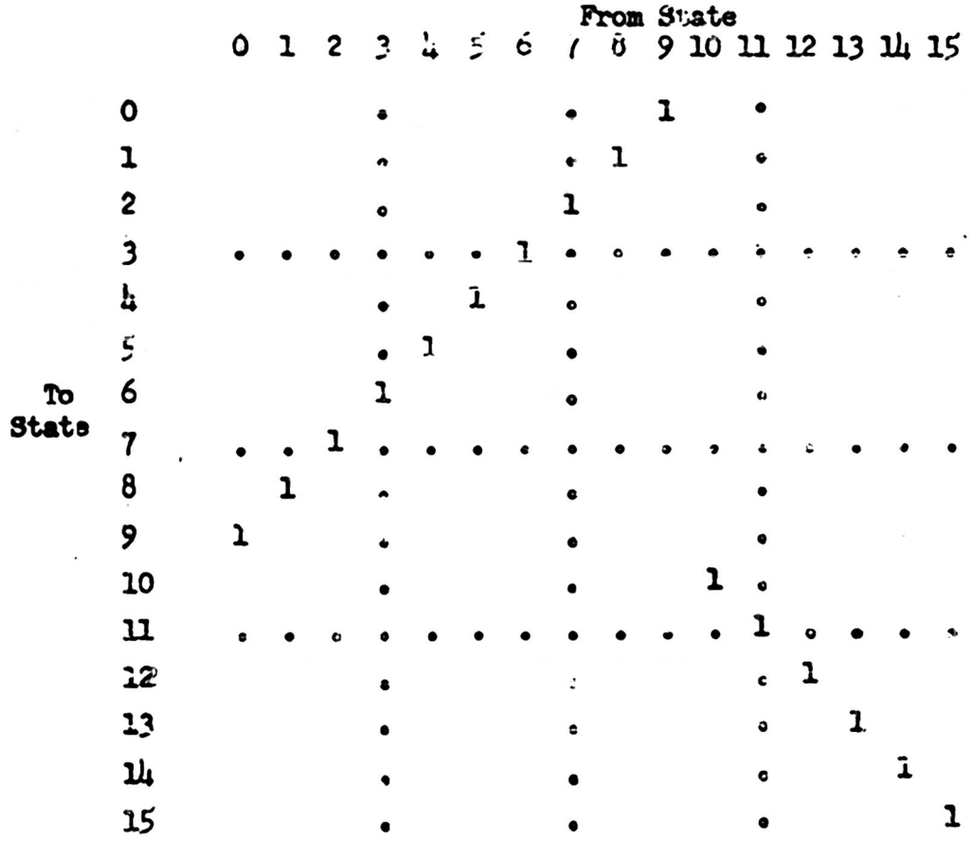


Figure 8.2 - Transition chart for derivation of nines complement in 1-2-4-8 system

The 1's in columns 10, 11, 12, 13, 14, and 15, require the toggles to be unchanged upon application of a pulse when a forbidden combination occurs. As earlier, let the variables before the transition be x_1, x_2, x_4, and x_8, and after y_1, y_2, y_4, and y_8. Let the products in x be s and the products in y, t; the subscript now indicates both designation and numerical value.

As usual,

$$y_1 = t_1+t_3+t_5+t_7+t_9+t_{11}+t_{13}+t_{15}$$
$$y_2 = t_2+t_3+t_6+t_7+t_{10}+t_{11}+t_{14}+t_{15}$$
$$y_4 = t_4+t_5+t_6+t_7+t_{12}+t_{13}+t_{14}+t_{15}$$
$$y_8 = t_8+t_9+t_{10}+t_{11}+t_{12}+t_{13}+t_{14}+t_{15}$$

In comparison with equations 5.7, note that the subscripts on y have been changed to indicate the weights in the 1-2-4-8 system.

Then, from the transition chart,

$$y_1 = s_8+s_6+s_4+s_2+s_0+[s_{11}+s_{13}+s_{15}]$$
$$y_2 = s_7+s_6+s_3+s_2+[s_{10}+s_{11}+s_{14}+s_{15}]$$
$$y_4 = s_5+s_4+s_3+s_2+[s_{12}+s_{13}+s_{14}+s_{15}]$$
$$y_8 = s_1+s_0+[s_{10}+s_{11}+s_{12}+s_{13}+s_{14}+s_{15}]$$

If the transitions from the forbidden states, columns 10, 11, 12, 13, 14, and 15, of the transition chart are left arbitrary, the expressions in the brackets can be omitted or replaced by other forbidden combinations in s. Omitting them amounts to ignoring the 1's in columns 10, 11, 12, 13, 14, and 15 in the chart. If, as here, the redundancy equations are to be applied for purposes of simplification later anyway, the 1's in the forbidden columns

can be ignored.

Then

$$y_1 = s_8+s_6+s_4+s_2+s_0$$
$$y_2 = s_7+s_6+s_3+s_2$$
$$y_4 = s_5+s_4+s_3+s_2$$
$$y_8 = s_1+s_0$$

$$\begin{aligned}
y_1 &= x_8x_4'x_2'x_1'+x_8'x_4x_2'x_1'+x_8'x_4'x_2x_1'+x_8'x_4'x_2'x_1+x_8'x_4'x_2'x_1' \\
y_2 &= x_8'x_4x_2x_1'+x_8'x_4'x_2x_1'+x_8'x_4'x_2'x_1+x_8'x_4'x_2x_1' \\
y_4 &= x_8'x_4x_2'x_1'+x_8'x_4'x_2x_1'+x_8'x_4'x_2'x_1+x_8'x_4'x_2x_1' \\
y_8 &= x_8'x_4'x_2'x_1'+x_8'x_4'x_2'x_1'
\end{aligned} \qquad 8.1$$

The redundancy equation for the 1-2-4-8 system has been derived earlier (see equation 5.33); in the notation used here it is

$$x_8(x_4+x_2) = 0; \qquad 8.2$$

then

$$\begin{aligned}
&x_2x_8 = 0 \\
&x_4x_8 = 0 \\
&x_2x_8' = x_2x_8'+x_2x_8 = x_2; \\
&x_2'+x_8 = x_2' \\
&x_4x_8' = x_4; \\
&x_4'+x_8 = x_4' \\
&x_2'x_8 = x_8 \\
&x_2+x_8' = x_8' \\
&x_4'x_8 = x_8 \\
&x_4+x_8' = x_8'
\end{aligned} \qquad 8.3$$

The following simplifications of equations 8.1 can be made by means of these relations

$$y_1 = x_8x_1' + x_4x_2x_1' + x_4x_2'x_1' + x_4'x_2x_1' + x_8'x_4'x_2'x_1'$$

$$y_1 = x_1'(x_8 + x_4x_2 + x_4x_2' + x_4'x_2 + x_8'x_4'x_2')$$

$$y_1 = x_1'(x_8 + x_4 + x_2 + x_4'x_2')$$

$$y_1 = x_1'(x_8 + 1 + x_2) = x_1'$$

$$y_2 = x_4x_2x_1 + x_4x_2x_1' + x_4'x_2x_1 + x_4'x_2x_1'$$

$$y_2 = x_4x_2 + x_4'x_2 = x_2$$

$$y_4 = x_4x_2'x_1 + x_4x_2'x_1' + x_4'x_2x_1 + x_4'x_2x_1'$$

$$y_4 = x_4x_2' + x_4'x_2$$

$$y_8 = x_8'x_4'x_2'$$

Let reversing input connections be used for the realization of the circuit and let the control functions be h_1, h_2, h_4, and h_8, then (see equation 5.22),

$$h_1 = y_1x_1' + y_1'x_1 = x_1' + x_1 = 1$$

$$h_2 = y_2x_2' + y_2'x_2 = x_2 \cdot x_2' + x_2'x_2 = 0$$

$$h_4 = y_4x_4' + y_4'x_4 = (x_4x_2' + x_4'x_2)x_4' + (x_4' + x_2)(x_4 + x_2')x_4$$

$$= x_4'x_2 + x_4x_2 = x_2$$

$$h_8 = y_8x_8' + y_8'x_8 = x_8'x_4'x_2' + (x_8 + x_4 + x_2)x_8 = x_8'x_4'x_2' + (1 + x_4 + x_2)x_8$$

$$= x_8'x_4'x_2' + x_8 = x_8 + x_4'x_2' = (x_8 + x_4')(x_8 + x_2') = x_4' \cdot x_2'$$

(see equation 2.23)

If a gate type which passes the pulse for low control voltage is used, the control functions should be inverted.

$(h_1)' = 0$

$(h_2)' = 1$

$(h_4)' = x_2'$

$(h_8)' = x_2 + x_4$

The block diagram for the circuit is shown in Figure 8.3.

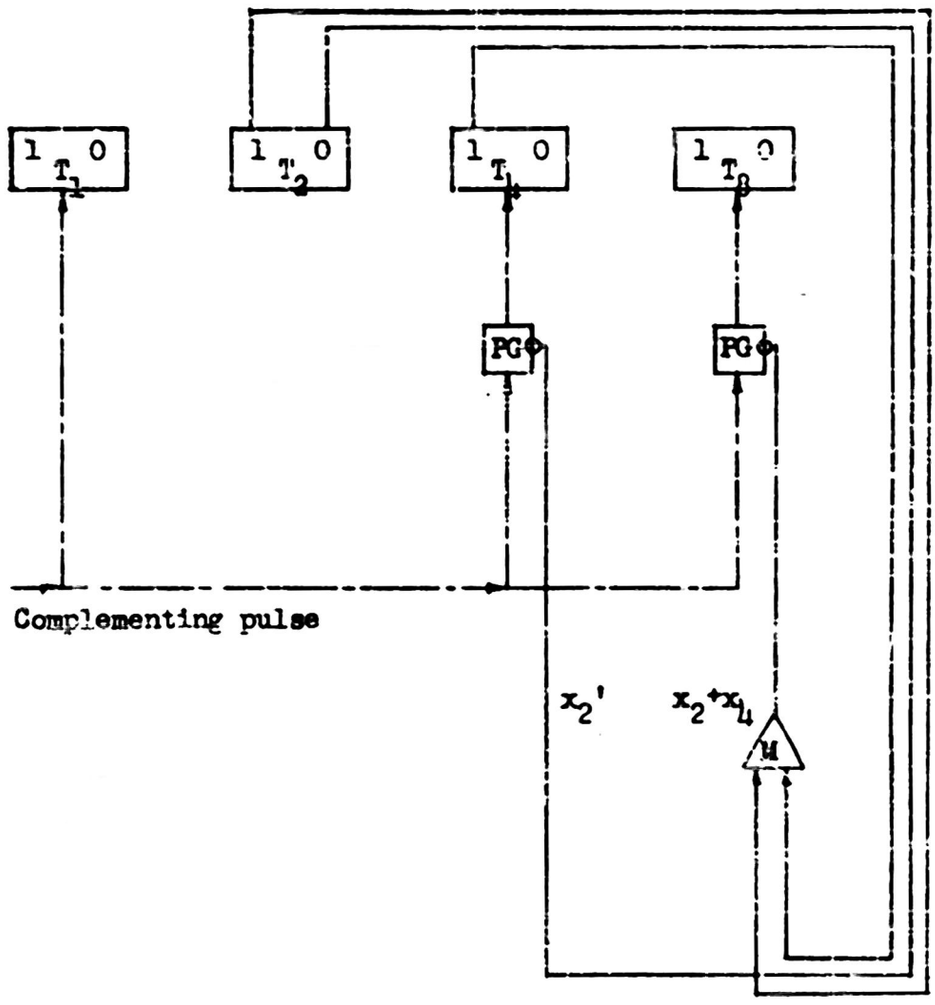


Figure 8.3 - Block diagram for circuit deriving nines complement in 1-2-4-8 system

III. THE ADD-CIRCUIT

In the circuit to be described, the addition of two decimal digits takes place in several steps. For purposes of addition, the binary digits of a decimal digit appear serially. The addition of the individual binary digits is carried out in a single step. In other words, a circuit which will add 1, 2, 4, and 8, separately, in one step each, is used; by letting the binary signals appear in time sequence, in the form of pulses from the D-register, the content of the adder-column is changed to the sum in steps. This operation is shown schematically in Figure 8.4

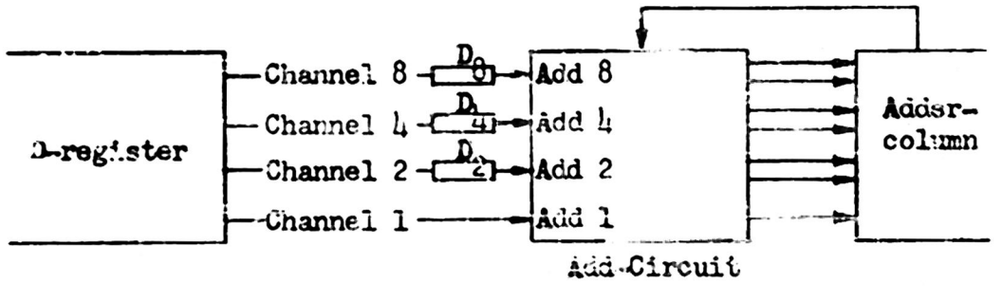


Figure 8.4 - Block diagram for step-by-step addition of a decimal digit

In the diagram each of the channels 1, 2, 4, and 8, from the D-register will at the time of addition either carry a pulse or no pulse; a pulse is carried when a "1" is present, no pulse when a "0" is present in the toggle corresponding to the channel in the D-register 1C-column. These pulses are made to appear in time sequence by insertion of the delays D_2, D_4, and D_8, in the connection between the channels and the add circuit. When a pulse

arrives at the add-1 terminal of the add circuit, 1 is added to the content of the adder column; similarly 2, 4, and 8, are added by application of pulses, at different times, to the add-2, the add-4, and the add-8 terminal.

Due to the redundancy, characteristic of the 1-2-4-8 system, however, two pulses will never appear simultaneously in channel 4 and channel 8, or in channel 2 and channel 8.

This follows from the redundancy equation, equation 8.2.

If

$$x_8 = 1$$

then

$$x_4 = 0;$$

and if

$$x_4 = 1$$

then

$$x_8 = 0 .$$

Consequently the delays D_4 and D_8, or D_2 and D_8, can be made the same. The addition takes place in three steps.

1. Add 1
2. Add 2
3. Add 4 or add 8.

The problem is then to design a circuit which will add 1, 2, 4, and 8, separately and without need for correction, in the 1-2-4-8 system.

Add 1. The transition chart is shown in Figure 8.5.

Here and in the following the columns and rows representing

forbidden combinations have been left out since the transitions from these states will be assumed arbitrary (see Section I of this Chapter).

To State \ From State	0	1	2	3	4	5	6	7	8	9
0				.			.			1
1	1			.			.			
2		1		.			.			
3	.	.	1	.	.	.	.	.	.	.
4				1			.			
5				.	1		.			
6	.	.	.	.	.	1	.	.	.	.
7				.			1			
8				.			.	1		
9				.			.		1	

Figure 8.5 - Transition chart for add-1 operation

Then, with the notation introduced earlier

$$y_1 = s_0 + s_2 + s_4 + s_6 + s_8$$

$$y_2 = s_1 + s_2 + s_5 + s_6$$

$$y_4 = s_3 + s_4 + s_5 + s_6$$

$$y_8 = s_7 + s_8$$

giving

$$y_1 = x_8'x_4'x_2'x_1' + x_8'x_4'x_2x_1' + x_8'x_4x_2'x_1' + x_8'x_4x_2x_1' + x_8x_4'x_2'x_1'$$

$$y_2 = x_8'x_4'x_2'x_1 + x_8'x_4'x_2x_1' + x_8'x_4x_2'x_1 + x_8'x_4x_2x_1'$$

$$y_4 = x_8'x_4'x_2x_1 + x_8'x_4x_2'x_1' + x_8'x_4x_2'x_1 + x_8'x_4x_2x_1'$$

$$y_8 = x_8'x_4'x_2x_1 + x_8x_4'x_2'x_1'$$

and using the redundancy equation, equation 8.2, and equations 8.3

$$y_1 = x_8'x_4'x_1' + x_4x_1' + x_8x_1' = x_1'(x_8 + x_8'x_4' + x_4) = x_1'(x_8 + x_4' + x_4) = x_1'$$

$$y_2 = x_8'x_4'x_2'x_1 + x_4'x_2'x_1 + x_4x_2x_1 + x_4x_2'x_1' = x_1x_2'(x_4' + x_4'x_8') + x_2x_1'$$

$$= x_1x_2'(x_4' + x_8') + x_2x_1' = x_1x_2'x_8' + x_2x_1'$$

$$y_4 = x_4'x_2x_1 + x_4x_2'x_1' + x_4x_2'x_1 + x_4x_2x_1' = x_4'x_2x_1 + x_4(x_2'x_1' + x_2'x_1 + x_2x_1')$$

$$= x_4'x_2x_1 + x_4(x_1' + x_2')$$

$$y_8 = x_4x_2x_1 + x_8x_1'$$

Let split input connections be used, then for the add-1 circuit, the following control functions, h (upper index indicates 0 or 1-side, the lower the toggle) are derived:

$$h_1^1 = y_1x_1' = x_1'$$

$$h_2^1 = y_2x_2' = x_1x_2'x_8'$$

$$h_4^1 = y_4x_4' = x_4'x_2x_1$$

$$h_8^1 = y_8x_8' = x_4x_2x_1x_8' = x_1x_2x_4,$$

and applying equation 2.25

$$h_1^0 = y_1'x_1 = [0]'x_1 = x_1$$

$$h_2^0 = y_2'x_2 = [x_1']'x_2 = x_1x_2$$

$$h_4^0 = y_4'x_4 = [x_1' + x_2']'x_4 = x_1 \cdot x_2 \cdot x_4$$

$$h_8^0 = y_8'x_8 = [x_4x_2x_1 + x_1']'x_8 = (x_4' + x_2')x_1x_8 = x_1x_8$$

The input to the 1-toggle is obviously most easily realised by means of a complementing input connection

$$h_1 = h_1^1 + h_1^0 = x_1' + x_1 = 1$$

Application of equation 5.13 does not lead to further simplifications.

If a pulse gate passing for a low control voltage is used,

the inverted control functions must be used.

$$(h_1)' = 0$$

$$(h_2^1)' = x_1' + x_2 + x_8 \qquad (h_2^0)' = x_1' + x_2'$$

$$(h_4^1)' = x_4 + x_2' + x_1' \qquad (h_4^0)' = x_1' + x_2' + x_4'$$

$$(h_8^1)' = x_1' + x_2' + x_4' \qquad (h_8^0)' = x_1' + x_8'$$

Add 2.

The procedure is exactly the same as for the add-1 circuit. The transition chart is shown in Figure 8.6.

To State \ From State	0	1	2	3	4	5	6	7	8	9
0				.			.		1	
1				.			.			1
2	1			.			.			
3	.	1	.	.	.	.	.	.	.	.
4			1	.			.			
5				1			.			
6	.	.	.	.	1	.	.	.	.	.
7				.		1	.			
8				.			1			
9				.			.	1		

Figure 8.6 - Transition chart for add-2 operation

$$y_1 = s_9 + s_1 + s_3 + s_5 + s_7$$

$$y_2 = s_0 s_1 + s_4 + s_5$$

$$y_4 = s_2 + s_3 + s_4 + s_5$$

$$y_8 = s_6 + s_7$$

$$y_1 = x_8x_4'x_2'x_1 + x_8'x_4'x_2x_1 + x_8'x_4x_2'x_1 + x_8'x_4x_2x_1 + x_8'x_4x_2x_1$$

$$y_2 = x_8'x_4'x_2'x_1 + x_8'x_4'x_2x_1 + x_8'x_4x_2'x_1 + x_8'x_4x_2'x_1$$

$$y_4 = x_8'x_4'x_2x_1 + x_8'x_4'x_2x_1 + x_8'x_4x_2'x_1 + x_8'x_4x_2'x_1$$

$$y_8 = x_8'x_4x_2x_1 + x_8'x_4x_2x_1;$$

$$y_1 = x_8x_1 + x_8'x_4'x_2'x_1 + x_4'x_2x_1 + x_4x_2'x_1 + x_4x_2x_1$$

$$= x_1(x_8 + x_8'x_4'x_2' + x_2x_4' + x_4x_2' + x_4x_2)$$

$$= x_1(x_8 + x_4'x_2' + x_2x_4' + x_4)$$

$$= x_1(x_8 + x_2' + x_4 + x_2)$$

$$y_1 = x_1;$$

$$y_2 = x_8'x_4'x_2' + x_4x_2'$$

$$= x_2'(x_4 + x_4'x_8')$$

$$y_2 = x_2'(x_4 + x_8') = x_2'x_8'$$

$$y_4 = x_4'x_2(x_1' + x_1) + x_4x_2'(x_1' + x_1)$$

$$y_4 = x_4'x_2 + x_4x_2'$$

$$y_8 = x_4x_2(x_1' + x_1) = x_4x_2 .$$

And

$$h_1^1 = y_1x_1' = 0$$

$$h_2^1 = y_2x_2' = x_2'x_8'$$

$$h_4^1 = y_4x_4' = x_4'x_2$$

$$h_8^1 = y_8x_8' = x_4x_2x_8' = x_4x_2$$

$$h_1^0 = y_1'x_1 = 0$$

$$h_2^0 = y_2'x_2 = [0]'x_2 = x_2$$

$$h_4^0 = y_4'x_4 = [x_2']'x_4 = x_4x_2$$

$$h_8^0 = y_8'x_8 = [x_2 + x_4]'x_8 = (x_2'x_4')x_8 = x_8 .$$

And the inverted functions

$(h_1^1)' = 1$ $\qquad$ $(h_1^0)' = 1$

$(h_2^1)' = x_2 + x_8$ $\qquad$ $(h_2^0)' = x_2'$

$(h_4^1)' = x_4 + x_2'$ $\qquad$ $(h_4^0)' = x_4' + x_2'$

$(h_8^1)' = x_4 + x_2$ $\qquad$ $(h_8^0)' = x_8'$

As before, no simplifications are possible by means of equation 5.13.

Add 4. The transition chart is shown in Figure 8.7.

To State \ From State	0	1	2	3	4	5	6	7	8	9
0				.			1			
1				.			.	1		
2				.			.		1	
3	.	.	.	.	.	.	.	.	.	1
4	1			.			.			
5		1		.			.			
6	.	.	1	.	.	.	.	.	.	.
7				1			.			
8				.	1		.			
9				.		1	.			

Figure 8.7 - Transition chart for add-4 operation

$y_1 = s_7 + s_9 + s_1 + s_3 + s_5$

$y_2 = s_8 + s_9 + s_2 + s_3$

$y_4 = s_0 + s_1 + s_2 + s_3$

$y_8 = s_4 + s_5$;

$y_1 = x_1$

The expression for y_1 is the same as in the add 2 case,

$$y_2 = x_8x_4'x_2'x_1' + x_8x_4'x_2'x_1 + x_8'x_4'x_2x_1' + x_8'x_4'x_2x_1$$

$$y_4 = x_8'x_4'x_2'x_1' + x_8'x_4'x_2'x_1 + x_8'x_4'x_2x_1' + x_8'x_4'x_2x_1$$

$$y_8 = x_8'x_4x_2'x_1' + x_8'x_4x_2'x_1;$$

$$y_1 = x_1$$

$$y_2 = x_8(x_1' + x_1) + x_2x_4'(x_1' + x_1)$$

$$= x_8 + x_2x_4'$$

$$y_4 = x_8'x_4'x_2'(x_1' + x_1) + x_4'x_2(x_1' + x_1)$$

$$= x_4'(x_8'x_2' + x_2) = x_4'(x_8' + x_2) = x_4'x_8'$$

$$y_8 = x_4x_2'(x_1' + x_1) = x_4x_2' .$$

And

$$h_1^1 = y_1x_1' = 0$$

$$h_2^1 = y_2x_2' = x_8x_2' = x_8$$

$$h_4^1 = y_4x_4' = x_4'x_8'$$

$$h_8^1 = y_8x_8' = x_4x_2'$$

$$h_1^0 = y_1'x_1 = 0$$

$$h_2^0 = y_2'x_2 = [x_8 + x_4']'x_2 = x_8' \cdot x_4 \cdot x_2 = x_4x_2$$

$$h_4^0 = y_4'x_4 = [0]'x_4 = x_4$$

$$h_8^0 = y_8'x_8 = [x_4x_2']'x_8 = (x_4' + x_2)x_8 = x_4'x_8 = x_8$$

Here, equation 5.13 is fulfilled in one case

$$h_2^1 = x_8 = x_8 \cdot x_2' \; ; \quad h_2^0 = x_4 \cdot x_2 \; ;$$

but

$$x_8 \cdot x_4 = 0 \, ,$$

then the factors x_2 and x_2' can be omitted

$$h_2^1 = x_8 \qquad\qquad h_2^0 = x_4$$

The inverted functions are

$(h_1^1)' = 1$ $\qquad$ $(h_1^0)' = 1$

$(h_2^1)' = x_8'$ $\qquad$ $(h_2^0)' = x_4'$

$(h_4^1)' = x_4 + x_8$ $\qquad$ $(h_4^0)' = x_4'$

$(h_8^1)' = x_4' + x_2$ $\qquad$ $(h_8^0)' = x_8'$

Add 8.

The transition chart is shown in Figure 8.8.

		From State									
		0	1	2	3	4	5	6	7	8	9
	0			1	•			•			
	1				1			•			
	2				•	1		•			
To State	3	•	•	•	•	•	1	•	•	•	•
	4				•			1			
	5				•			•	1		
	6	•	•	•	•	•	•	•	•	1	•
	7				•			•			1
	8	1			•			•			
	9		1		•			•			

Figure 8.8 - Transition chart for add-8 operation

$y_1 = s_3 + s_5 + s_7 + s_9 + s_1$

$y_2 = s_4 + s_5 + s_6 + s_9$

$y_4 = s_6 + s_7 + s_8 + s_9$

$y_8 = s_0 + s_1$

$y_1 = x_1$, as before.

$$y_2 = x_8'x_4x_2'x_1' + x_8'x_4x_2'x_1 + x_8x_4'x_2'x_1' + x_8x_4'x_2'x_1$$

$$y_4 = x_8'x_4x_2x_1' + x_8'x_4x_2x_1 + x_8x_4'x_2'x_1' + x_8x_4'x_2'x_1$$

$$y_8 = x_8'x_4'x_2'x_1' + x_8'x_4'x_2'x_1;$$

$$y_1 = x_1$$

$$y_2 = x_4x_2'(x_1' + x_1) + x_8(x_1' + x_1) = x_4x_2' + x_8$$

$$y_4 = x_4x_2(x_1' + x_1) + x_8(x_1' + x_1) = x_4x_2 + x_8$$

$$y_8 = x_8'x_4'x_2'.$$

And

$$h_1^1 = y_1x_1' = 0$$

$$h_2^1 = y_2x_2' = (x_4 + x_8)x_2' = x_4x_2' + x_8$$

$$h_4^1 = y_4x_4' = x_8x_4' = x_8$$

$$h_8^1 = y_8x_8' = x_8'x_4'x_2'$$

$$h_1^0 = y_1'x_1 = 0$$

$$h_2^0 = y_2'x_2 = x_8'x_2 = x_2$$

$$h_4^0 = y_4'x_4 = [x_2 + x_8]'x_4 = x_2'x_8'x_4 = x_2'x_4$$

$$h_8^0 = y_8'x_8 = [0]'x_8 = x_8.$$

The inverted functions are

$$(h_1^1)' = 1 \qquad (h_1^0)' = 1$$

$$(h_2^1)' = (x_4' + x_2)x_8' \qquad (h_2^0)' = x_2'$$

$$(h_4^1)' = x_8' \qquad (h_4^0)' = x_2 + x_4'$$

$$(h_8^1)' = x_8 + x_4 + x_2 \qquad (h_8^0)' = x_8'$$

Carry Toggle

The carry toggle is set to the 0-state before the addition of each decimal digit. It has to receive a 1-input signal every time a carry is produced by a decimal addition. Let the function which

controls the 1-input terminal to the carry toggle be h_c. Then, in the case of add-1, clearly

$$\text{for add-1} \qquad h_c = s_9 ,$$

only when the adder-column contains the digit 9 will a carry be produced by addition of 1.

Similarly for

$$\text{add-2} \qquad h_c = s_8+s_9,$$

$$\text{add-4} \qquad h_c = s_6+s_7+s_8+s_9,$$

$$\text{add-8} \qquad h_c = s_2+s_3+s_4+s_5+s_6+s_7+s_8+s_9 .$$

Then for

$$\text{add-1} \qquad h_c = x_8x_4'x_2'x_1 = x_8x_1$$

$$\text{add-2} \qquad h_c = x_8x_4'x_2'x_1'+x_8x_4'x_2'x_1' = x_8$$

$$\text{add-4} \qquad h_c = x_8'x_4x_2x_1'+x_8'x_4x_2x_1+x_8x_4'x_2'x_1'+ x_8x_4'x_2'x_1 = x_4x_2+x_8$$

$$\text{add-8} \qquad h_c = x_8'x_4'x_2x_1'+x_8'x_4'x_2x_1+x_8'x_4x_2'x_1'+ x_8'x_4x_2'x_1+x_8'x_4x_2x_1'+x_8'x_4x_2x_1+ x_8x_4'x_2'x_1'+x_8x_4'x_2'x_1 = x_4'x_2+x_4x_2'+x_4x_2+x_8 = x_2+x_4+x_8$$

In order to suit a pulse gate type which passes for a low control voltage, these functions will be inverted;

$$\text{add-1} \qquad (h_c)' = x_8'+x_1'$$

$$\text{add-2} \qquad (h_c)' = x_8'$$

$$\text{add-4} \qquad (h_c)' = (x_4'+x_2')x_8'$$

$$\text{add-8} \qquad (h_c)' = x_2'x_4'x_8'$$

The complete set of input control functions rearranged accord-

ing to toggles and for use with a pulse gate passing for low control voltage is:

		0-input	1-input
T_1:	add-1	0 (reversing)	
	add-2	1	1
	add-4	1	1
	add-8	1	1
T_2:	add-1	$(x_1'+x_2')_1$	$(x_1'+(x_8+x_2)_5)_7$
	add-2	x_2'	$(x_8+x_2)_5$
	add-4	x_4'	x_8'
	add-8	x_2'	$((x_4'+x_2)_3x_8')_{12}$
T_4:	add-1	$((x_1'+x_2')_1+x_4')_6$	$((x_1'+x_2')_1+x_4')_8$
	add-2	$(x_2'+x_4')_4$	$(x_2'+x_4)_9$
	add-4	x_4'	$(x_4+x_8)_{10}$
	add-8	$(x_2+x_4')_3$	x_8'
T_8:	add-1	$(x_1'+x_8')_2$	$((x_1'+x_2')_1+x_4')_6$
	add-2	x_8'	$(x_2'+x_4')_4$
	add-4	x_8'	$(x_2+x_4')_3$
	add-8	x_8'	$((x_2+x_8)_5+x_4)_{13}$
T_c(Carry):	add-1	1	$(x_1'+x_8')_2$
	add-2	1	x_8'
	add-4	1	$((x_2'+x_4')_4x_8')_{11}$
	add-8	1	$(x_2'\cdot x_4'\cdot x_8')_{14}$

The indexed parentheses indicate groups of idential logical ex-

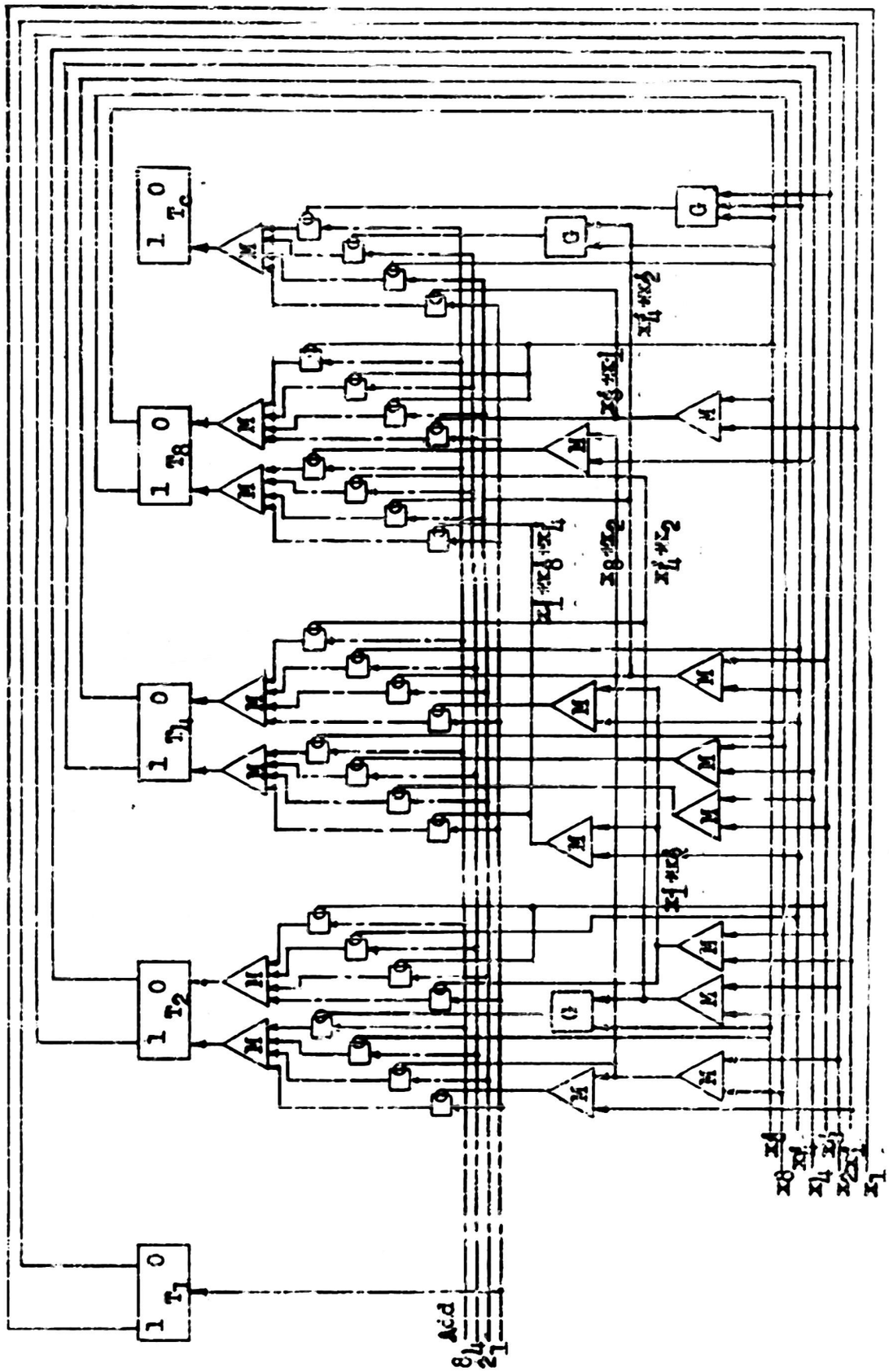


Figure 8.9 - Block diagram for add-circuit

pressions. It is necessary to form functions which appear in several places once only, $f_1 = (x_1' + x_2')_1$, for example, only has to be formed once although it appears in four different places.

The block diagram for the formation of these control functions is shown in Figure 8.9.

IV. COORDINATION OF ADD-SUBTRACT OPERATIONS

The operations described in the preceding sections are coordinated by means of separate control circuits which will not be described here. The steps in an add-subtract process will, however, be discussed.

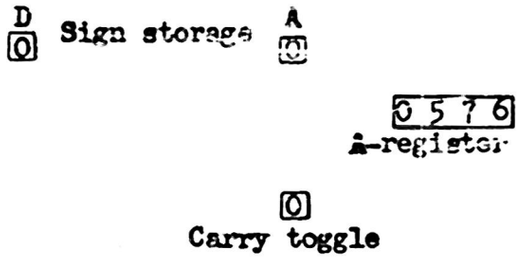
In the course of the arithmetic operation, which includes addition of the sign digits, the individual sign digits are lost. In the beginning of the process, therefore, the two signs are stored in two toggles in the control circuits. In case of subtraction, the sign of the subtrahend is reversed simultaneously. The final signs in the sign storage are then always associated with addition. Thus only addition, with four possible sign combinations in the sign storage, has to be considered. At the beginning, when the signs are stored, the two sign-columns are cleared. During the operation, the content of the sign-columns is treated the same way as the content of the digit columns, that is, it is subject to complementation, carry transfer, and addition.

When all digits and the signs have been operated on, it remains to extract the result. This involves determination of the sign of the result and possibly decomplementing; the latter will be performed by another complete arithmetic cycle.

For the sake of explanation let it be assumed that the registers, A-register and D-register, have four columns, one sign and three decimal digits. The decimal point is immediately to the left of the 1-column. When the sign column holds a "1", the number is negative; when it holds a "0", the number is positive. The number in the D-register will be denoted D; the number in the A-register, A.

Consider addition of two numbers.

D = 0.234; A = 0.576; A+D = 0.810



0. The sign columns are cleared and final signs, associated with addition, are stored in the sign storage. If the signs (in the sign storage) are opposite, a 1 is stored in the carry toggle.

Example: No change

1.1 If the signs are opposite the number in the last column of the A-register is replaced by the nines complement.

No change

1.2 The A-register is circulated one step

1.3 The digit in the last column of the D-register and the carry

are added to the digit in the adder-column (the far left column in the A-register). The D-register is shifted one place.

2.1 1.1 is repeated

No change

2.2 1.2 is repeated

2.3 1.3 is repeated

3.1 1.1 is repeated

No change

3.2 1.2 is repeated

3.3 1.3 is repeated

4.1 1.1 is repeated

No change

4.2 1.2 is repeated

4.3 1.3 is repeated

a. The signs are the same

a_1 If there is a 1 in the adder-column an overflow alarm is sounded.

a_2 If there is a 0 in the adder-column and the signs are plus, the operation is complete.

a_3 If there is a 0 in the adder-column and the signs are minus, the adder-column is set to "1" and the operation is complete.

b. A is negative and B is positive

b_1 If the adder-column contains a 0 or a 1, the carry toggle is set to 0. The operation is complete.

b_2 If the adder-column contains a 9, it is changed to an 8, the carry toggle is set to 1, and the operation is continued through another complete (decomplementing) cycle.

c. A is positive and B is negative

c_1 If the adder-column contains a 0, it is changed to a 1, and

the carry toggle is set to 0. The operation is complete.

c_2 If the adder-column contains a 9, the signs in the sign storage are reversed, the carry toggle is set to 1, and the operation is continued through another (decomplementing) cycle.

The example comes under a_2; the operation is complete.

The four cases of addition with opposite signs, b_1, b_2, c_1, c_2, will be considered briefly by examples.

D = 0.300; A = -0.100; A+D = 0.20

0. [0] [1]

[0 3 0 0] [1] [0 1 0 0]

(The sign storage is not repeated in the following steps)

1.1 [0 3 0 0] [0 1 0 9]

[1]

1.2 [0 2 0 0] [9 0 1 0]

[1]

1.3 [0 0 3 0] [0 0 1 0]

[1]

2.1. . .2.3

[0 0 0 3] [0 0 0 1]

[1]

3.1. . .3.3

[0 0 0 0] [2 0 0 0]

[1]

4.1. . .4.3

[0 0 0 0] [0 2 0 0]

5. case b_1

Operation complete

D = -0.300; A = 0.100; A+D = -0.200

0. 1 0

1.1. . .4.3

5. case c_1

Operation complete

D = -0.100; A = 0.300; A+D = 0.200

0. 1 0

1.1. . .4.3

5. case a_2

0	1
0000	9800
	1

New cycle

1.1. . .4.3

0	1
5000	0200
	0

5. Now case b_1. Operation complete

D = 0.100; A = −0.300; A+D = −0.20

0.	0	1
	0100	0300
		1

1.1. . .4.3.

0	1
0000	9800
	0

5. case b_2

0	1
0000	8800
	1

1.1. . .4.3

0000	1200
	0

5. Now case b_1. The operation is complete.

In the operation described, step 1.3 and 2.1 can be performed simultaneously. This is also true for step 2.3 and 3.1 and step 3.3 and 4.1, of course.

The decimal carry may either be taken care of by special circuits or be applied to the add-1 terminal of the add circuit.