

THE TRANSMISSION OF INFORMATION

ROBERT M. FANO

TECHNICAL REPORT NO. 65

MARCH 17, 1949

RESEARCH LABORATORY OF ELECTRONICS
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

The research reported in this document was made possible through support extended the Massachusetts Institute of Technology, Research Laboratory of Electronics, jointly by the Army Signal Corps, the Navy Department (Office of Naval Research) and the Air Force (Air Materiel Command), under Signal Corps Contract No. W36-039-sc-32037, Project No. 102B; Department of the Army Project No. 3-99-10-022.

THE TRANSMISSION OF INFORMATION

Robert M. Fano

Abstract

This report presents a theoretical study of the transmission of information in the case of discrete messages and noiseless systems. The study begins with the definition of a unit of information (a selection between two choices equally likely to be selected), and this is then used to determine the amount of information conveyed by the selection of one of an arbitrary number of choices equally likely to be selected. Next, the average amount of information per selection is computed in the case of messages consisting of sequences of independent selections from an arbitrary number of choices with arbitrary probabilities of their being selected. A recoding procedure is also presented for improving the efficiency of transmission by reducing, on the average, the number of selections (digits or pulses) required to transmit a message of given length and given statistical character. The results obtained in the case of sequences of independent selections are extended later to the general case of non-independent selections. Finally, the optimum condition is determined for the transmission of information by means of quantized pulses when the average power is fixed.

Introduction

It is the opinion of many workers in the field of electrical communications that the communication art is today at a major turning point of its development. The objective of almost all electrical communication systems has been, up to now, to eliminate distance in some form of human activity or relationships between men. Telegraph, telephone and television are typical examples of such communication systems. We may add to these teletype, tele-control and telemetering. It is interesting to note that the names of all these communication systems involve the prefix tele, meaning "at a distance".

Although, for obvious reasons, forms of communication over distances much greater than the ranges of human senses and reach were first to receive attention, the magnitude of the distance involved is not of primary importance from a logical point of view in the concept of communication. Communication is basically any form of transmission of information, regardless of the distance between the transmitter and the receiver. In a broader sense, the field of communication includes any handling, combining, comparing or employing of information, since such processes involve and are intimately connected with the transmission of such information.

It is clear, then, that most human activities involve communication in a broad sense, and, in particular, those activities which are considered of higher intellectual type because they depend to a high degree on the process of "thinking". Thinking itself, in fact, involves a natural communication system of a complexity far beyond that conceivable for any man-made system.

The above considerations point clearly to a very wide field of useful applications of the communication art which has hardly been touched as yet. It is to be expected that each application should present problems of a higher order of complexity than those encountered in the past. Consequently, it is also to be expected that the solution of these problems should necessitate the use of more powerful analytical tools and, particularly, should require a more fundamental study of the process of transmission of information. As a matter of fact, the first and most significant step in the direction of such a study was made by Norbert Wiener (1) in connection with the development of predictors for antiaircraft fire control. The statistical nature of this problem led him to the realization that all communication problems are fundamentally of a statistical nature, and must be handled accordingly. He argued that the signal to be transmitted in a communication system can never be considered as a known function of time, because if it were a priori known it could not convey any new information and therefore would not need to be transmitted. On the other hand, what can be known

a priori about a signal to be transmitted is its statistical character — that is, for instance, the probability distribution of its amplitude. In addition, it is equally clear, that noise, which plays such an important part in communication problems, can be described only in statistical terms. It follows that all communication problems are inherently statistical in nature, and that disregarding this fact may lead to unexplainable inconsistencies in addition to precluding a deeper understanding of such problems.

The statistical theory of optimum prediction and filtering developed by Wiener led further to the realization of the need for a basic and general criterion for judging the quality of communication systems. In fact, the mean-square error criterion used by Wiener in this part of his work is dictated by mathematical convenience rather than by physical considerations; consequently it may not be useful in certain practical problems. The search for a more appropriate criterion leads naturally to the question of what is the operation that a communication system must perform. If we take as an example a telegraph system, it might seem at first obvious that such a system must reproduce at the output each and every letter of the input message in the proper order. We may observe, however, that if one letter is received incorrectly, the word containing it is still perfectly understandable in most cases, and so, of course, is the whole message. Moreover, the message would still be comprehensible if, for instance, all the vowels were eliminated (which is what is done in written Hebrew). On the other hand, the incorrect transmission of a digit in a number would make the received message incorrect.

It appears therefore that the transmission of the information conveyed by a written message is what we wish to obtain and that this is not necessarily equivalent to the transmission of all the letters contained in the written message. More precisely, it appears that the different symbols, letters or figures contained in a written message do not contribute equally to the transmission of information — so much so, that some of them may be completely unnecessary. Similar conclusions are reached by considering other types of communication systems. In particular, the recent work on the Vocoder (2) and the clipping of speech waves (3) has provided considerable evidence in the same general direction.

The above considerations are relevant to another problem with which communication engineers are becoming more and more concerned, namely, that of bandwidth reduction. As a matter of fact, the Vocoder was developed primarily for the purpose of reducing the bandwidth required for speech transmission. It is clear that if different parts of a message are not equally important, some saving in bandwidth might be possible by providing transmission facilities which are proportional to the importance of these

different parts. The bandwidth problem, in turn, is intimately connected with the noise-reduction problem. In fact, all the different types of modulation developed for the purpose of noise and interference reduction require a bandwidth wider than that required by amplitude modulation. This method of paying for an improved signal-to-noise ratio with an increased bandwidth appears to be the result of some fundamental limitation which, however, the conventional approach to communication problems has failed to clarify.

The above discussion of some of the problems confronting or likely to confront the communication engineer indicates clearly the necessity of providing a measure for the "thing" which is to be transmitted and which has been vaguely called "information". Such a measure will then permit a quantitative and more fundamental study of the process involved in the transmission of information which, in turn, will lead eventually to the design of better and more efficient communication devices. A considerable amount of work in this direction has already been done independently by Norbert Wiener (4) and Claude Shannon (5). The work of Wiener is particularly outstanding because of its philosophical profoundness and its importance in many branches of science other than communication engineering. Mention should be made also of the pioneering work of Hartley (6) and of the more recent work of Tuller (7).

This paper presents the work done by the author in the past year on the transmission of discrete signals through a noiseless channel. Although most of the results obtained have already been published by Wiener and Shannon, it is felt that the method of approach used here is sufficiently different to justify this redundant presentation.

I. Definition of the Unit of Information

In order to define, in an appropriate and useful manner, a unit of information, we must first consider in some detail the nature of those processes in our experience which are generally recognized as conveying information. A very simple example of such processes is a yes-or-no answer to some specific question. A slightly more involved process is the indication of one object in a group of N objects, and, in general, the selection of one choice from a group of N specific choices. The word "specific" is underlined because such a qualification appears to be essential to these information-conveying processes. It means that the receiver is conscious of all possible choices, as is, of course, the transmitter (that is, the individual or the machine which is supplying the information). For instance, saying "yes" or "no" to a person who has not asked a question obviously does not convey any information. Similarly, the reception of a code number which

is supposed to represent a particular message does not convey any information unless there is available a code book containing all the messages with the corresponding code numbers.

Considering next more complex processes, such as writing or speaking, we observe that these processes consist of orderly sequences of selections from a number of specific choices, namely, the letters of the alphabet or the corresponding sounds. Furthermore, there are indications that the signals transmitted by the nervous system are of a discrete rather than of a continuous nature, and might also be considered as sequences of selections. If this were the case, all information received through the senses could be analyzed in terms of selections. The above discussion indicates that the operation of selection forms the basis of a number of processes recognized as conveying information, and that it is likely to be of fundamental importance in all such processes. We may expect, therefore, that a unit of information, defined in terms of a selection, will provide a useful basis for a quantitative study of communication systems.

Considering more closely this operation of selection, we observe that different informational value is naturally attached to the selection of the same choice, depending on how likely the receiver considered the selection of that particular choice to be. For example, we would say that little information is given by the selection of a choice which the receiver was almost sure would be selected. It seems appropriate, therefore, in order to avoid difficulty at this early stage, to use in our definition the particular case of equally likely choices - that is, the case in which the receiver has no reason to expect that one choice will be selected rather than any other. In addition, our natural concept of information indicates that the information conveyed by a selection increases with the number of choices from which the selection is made, although the exact functional relation between these two quantities is not immediately clear.

On the basis of the above considerations, it seems reasonable to define as the unit of information the simplest possible selection, namely, the selection between two equally likely choices, called, hereafter, the "elementary selection". For completeness, we must add to this definition the postulate, consistent with our intuition, that N independent selections of this type constitute N units of information. By independent selections we mean, of course, selections which do not affect one another. We shall adopt for this unit the convenient name of "bit" (from "binary digit"), suggested by Shannon. We shall also refer to a selection between two choices (not necessarily equally likely) as a "binary selection", and to a selection from N choices, as an N -order selection. When the choices are, *a priori*, equally likely, we shall refer to the selection as an "equally likely selection".

We can now proceed to develop ways of measuring the information content of discrete messages in terms of the unit just defined. Most of this paper will be devoted to the solution of this problem.

II. Selection from N Equally Likely Choices

Consider now the selection of one among a number, N , of equally likely choices. In order to determine the amount of information corresponding to such a selection, we must reduce this more complex operation to a series of independent elementary selections. The required number of these elementary selections will be, by definition, the measure in bits of the information given by such an N -order selection.

Let us assume for the moment that N is a power of two. In addition (just to make the operation of selection more physical), let us think of the N choices as N objects arranged in a row, as indicated in Figure 1.

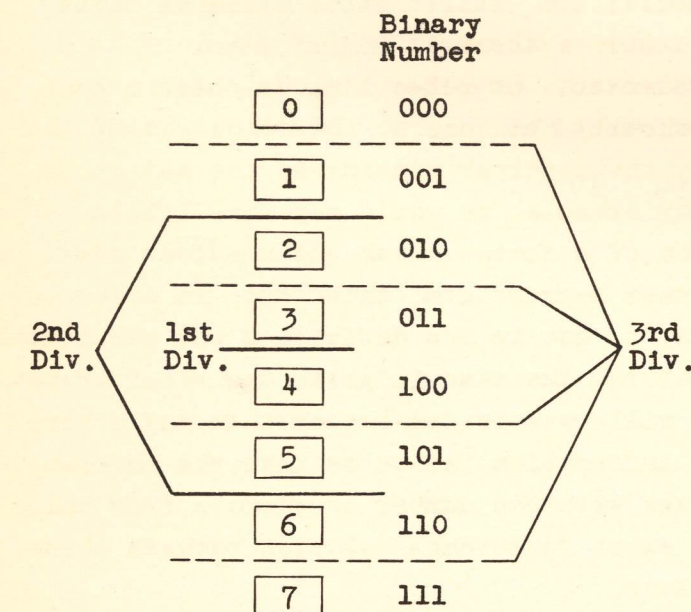


Fig. 1 Selection procedure for equally likely choices.

These N objects are first divided in two equal groups, so that the object to be selected is just as likely to be in one group as in the other. Then the indication of the group containing the desired object is equivalent to one elementary selection, and, therefore, to one bit. The next step consists of dividing each group into two equal subgroups, so that the object to be selected is again just as likely to be in either subgroup. Then one additional elementary selection, that is a total of two elementary selections, will suffice to indicate the desired subgroup (of the possible four subgroups). This process of successive subdivisions and corresponding elementary selections is carried out until the desired object is isolated from

the others. Two subdivisions are required for $N = 4$, three for $N = 8$, and, in general, a number of subdivisions equal to $\log_2 N$, in the case of an N -order selection.

The same process can be carried out in a purely mathematical form by assigning order numbers from 0 to $N-1$ to the N choices. The numbers are then expressed in the binary system, as shown in Figure 1, the number of binary digits (0 or 1) required being equal to $\log_2 N$. These digits represent an equal number of elementary selections and, moreover, correspond in order to the successive divisions mentioned above. In conclusion, an N -order, equally likely selection conveys an amount of information

$$H_N = \log_2 N \quad (1)$$

The above result is strictly correct only if N is a power of two, in which case H_N is an integer. If N is not a power of two, then the number of elementary selections required to specify the desired choice will be equal to the logarithm of either the next lower or the next higher power of two, depending on the particular choice selected. Consider, for instance, the case of $N = 3$. The three choices, expressed as binary numbers, are then

$$00 ; 01 ; 10 \quad .$$

If the binary digits are read in order from left to right, it is clear that the first two numbers require two binary selections — that is, two digits, while the third number requires only the first digit, 1, in order to be distinguished from the other two. In other words, the number of elementary selections required when N is not a power of two is equal to either one of the two integers closest to $\log_2 N$. It follows that the corresponding amount of information must lie between these two limits, although the significance of a non-integral value of H is not clear at this point. It will be shown in the next section that Eq.(1) is still correct when N is not a power of two, provided H_N is considered as an average value over a large number of selections.

III. Messages and Average Amount of Information

We have determined in the preceding section the amount of information conveyed by a single selection from N equally likely choices. In general, however, we have to deal with not one but long series of such selections, which we call messages. This is the case, for instance, in the transmission of written intelligence. Another example is provided by the communication system known as pulse-code modulation, in which audio waves are sampled at equal time intervals and then each sample is quantized, that is approximated by the closest of a number N of amplitude levels.

Let us consider, then, a message consisting of a sequence of n successive N -order selections. We shall assume, at first, that these selections are independent and equally likely. In this simpler case, all the different sequences which can be formed equal in number to

$$S = N^n \quad (2)$$

are equally likely to occur. For instance, in the case of $N = 2$ (the two choices being represented by the numbers 0 and 1) and $n = 3$, the possible sequences would be 000, 001, 010, 100, 011, 101, 110, 111. The total number of these sequences is $S = 8$ and the probability of each sequence is $1/8$. In general, therefore, the ensemble of the possible sequences may be considered as forming a set of S equally likely choices, with the result that the selection of any particular sequence yields an amount of information

$$H_S = \log_2 S = n \log_2 N \quad (3)$$

In words, n independent equally likely selections give n times as much information as a single selection of the same type. This result is certainly not surprising, since it is just a generalization of the postulate, stated in Section II, which forms an integral part of the definition of information.

It is often more convenient, in dealing with long messages, to use a quantity representing the average amount of information per N -order selection, rather than the total information corresponding to the whole message. We define this quantity in the most general case as the total information conveyed by a very long message divided by the number of selections in the message, and we shall indicate it with the symbol H_N , where N is the order of each selection. It is clear that when all the selections in the message are equally likely and independent and, in addition, N is a power of two, the quantity H_N is just equal to the information actually given by each selection, that is

$$H_N = \frac{1}{n} \log_2 S = \log_2 N \quad (4)$$

We shall show now that this equation is correct also when N is not a power of two, in which case H_N has to be actually an average value taken over a sufficiently long sequence of selections.*

The number S of different and equally likely sequences which can be formed with n independent and equally likely selections is still given by Eq.(2), even when N is not a power of two. On the contrary, the number of elementary selections required to specify any one particular sequence must

* The author is indebted to Mr. T. P. Cheatham, Jr. (of this Laboratory) for the original idea on which is based both this proof and the corresponding recoding procedure (see Section IV).

be written now in the form

$$B_S = \log_2 S + d, \quad (5)$$

where d is a number, smaller in magnitude than unity, which makes B_S an integer and which depends on the particular sequence selected. The average amount of information per N -order selection is then, by definition,

$$H_N = \lim_{n \rightarrow \infty} \frac{1}{n} (\log_2 S + d). \quad (6)$$

Since N is a constant and since the magnitude of d is smaller than unity while n approaches infinity, this equation together with Eq.(2) yields

$$H_N = \log_2 N. \quad (7)$$

We shall consider now the more complex case in which the selections, although still independent, are not equally likely. In this case, too, we wish to compute the average amount of information per selection. For this purpose, we consider again the ensemble of all the messages consisting of n independent selections and we look for a way of indicating any one particular message by means of elementary selections. If we were to proceed as before, and divide the ensemble of messages in two equal groups, the selection of the group containing the desired message would no longer be a selection between equally likely choices, since the sequences themselves are not equally likely. The proper procedure is now, of course, to make equal for each group not the number of messages in it but the probability of its containing the desired message. Then the selection of the desired group will be a selection between equally likely choices. This procedure of division and selection is repeated over and over again until the desired message has been separated from the others. The successive selections of groups and subgroups will then form a sequence of independent elementary selections.

One may observe, however, that it will not generally be possible to form groups equally likely to contain the desired message, because shifting any one of the messages from one group to the other will change, by finite amounts, the probabilities corresponding to the two groups. On the other hand, if the length of the messages is increased indefinitely, the accuracy with which the probabilities of the two groups can be made equal becomes better and better since the probability of each individual message approaches zero. Even so, when the resulting subgroups include only a few messages after a large number of divisions, it may become impossible to keep the probabilities of such subgroups as closely equal as desired unless we proceed from the beginning in an appropriate manner as indicated below. The

messages are first arranged in order of their probabilities, which can be easily computed if the probabilities of the choices are known. The divisions in groups and subgroups are then made successively without changing the order of the messages, as illustrated in Figure 2. In this manner, the smaller subgroups will contain messages with equal or almost equal probabilities, so that further subdivisions can be performed satisfactorily.

It is clear that when the above procedure is followed, the number of binary selections required to separate any message from the others varies

Probabilities of Groups Obtained by Successive Divisions						Message	P(1)	Recorded Message	P(1) $B_g(1)$	
I Div.	II Div.	III Div.	IV Div.	V Div.	VI Div.					
0.49						00	0.49	0	0.49	
0.51	0.28	0.14				01	0.14	100	0.42	
		0.14				10	0.14	101	0.42	
	0.23	0.14	0.07				02	0.07	1100	0.28
			0.07				20	0.07	1101	0.28
		0.09	0.04				11	0.04	1110	0.16
			0.05				12	0.02	11110	0.10
	0.03	0.02	0.02				21	0.02	111110	0.12
				0.01				22	0.01	111111
										$(B_g)_{av.} = 2.33$

Fig. 2 Recoding of messages consisting of 2 third-order selections, for choice probabilities $p(0) = 0.7$, $p(1) = 0.2$, $p(2) = 0.1$, $H_3 = - [0.7 \log_2 0.7 + 0.2 \log_2 0.2 + 0.1 \log_2 0.1] = 1.157$.

For original code $\eta = \frac{H_3}{\log_2 3} = 0.73$;

For new code $\eta = \frac{2H_3}{(B_g)_{av.}} = 0.993$.

from message to message. Messages with a high probability of being selected require less binary selections than those with lower probabilities. This fact is in agreement with the intuitive notion that the selection of a little-probable message conveys more information than the selection of a more-probable one. Certainly, the occurrence of an event which we know a priori to have a 99 per cent probability is hardly surprising or, in our terminology, yields very little information, while the occurrence of an event which has a probability of only 1 per cent yields considerably more information. More precisely, as shown below, if $P(1)$ is the probability of the 1^{th} message, the number of binary selections required to indicate this message will be an integer $B_S(1)$ close to $-\log_2 P(1)$. In fact, $P(1)$ is just the probability of the last subgroup obtained by successively halving (approximately) the probability of the whole ensemble of messages (which is unity) a number of times equal to $B_S(1)$, so that $P(1) \approx 2^{-B_S(1)}$. By making the messages sufficiently long - that is, the number n of N -order selections sufficiently large - the integer $B_S(1)$ can be made to differ in percentage from $-\log_2 P(1)$ by less than any desired amount. Hence, in this limiting case, we can write

$$B_S(1) = -\log_2 P(1) \quad (8)$$

Let us consider now a sequence of M selections of messages, each message consisting of n N -order selections (forming a sequence of nM selections). By making the number M sufficiently large, we can be practically sure that the 1^{th} message will appear in the sequence with a frequency as close to $P(1)$ as desired. Therefore the number of binary selections required on the average to select one message, that is, "the mathematical expectation of B_S ", will be

$$E(B_S) = \sum_{i=0}^{S-1} P(1) B_S(1) \quad (9)$$

The average amount of information per N -order selection is then, from Eqs. (8) and (9),

$$H_N = \lim_{n \rightarrow \infty} \frac{E(B_S)}{n} = \lim_{n \rightarrow \infty} -\left(\frac{1}{n}\right) \sum_{i=0}^{S-1} P(1) \log_2 P(1) \quad (10)$$

that is, the limit of the ratio of the number of binary selection required, on the average, to select one message to the number of N -order selections in the message.

Now let $p(k)$ be the probability of the k^{th} choice (of the N), and n_k

be the number of times the k^{th} choice is selected in the 1^{th} message (sequence of n selections). The probability of the 1^{th} message is

$$P(1) = \prod_{k=0}^{N-1} [p(k)]^{n_k(1)} \quad (11)$$

The number of binary selections required to indicate this message can be written as

$$B_S(1) = -\log_2 \left[\prod_{k=0}^{N-1} [p(k)]^{n_k(1)} \right] = -\sum_{k=0}^{N-1} n_k(1) \log_2 p(k) \quad (12)$$

with any degree of accuracy desired. In the limit when n approaches infinity these binary selections become elementary selections, that is, binary selections between equally likely choices. We must now compute $E(B_S)$ according to Eq.(9). The number of sequences of selections, that is, messages, to which correspond the same values of $P(1)$ and $B_S(1)$, is equal to the number of different permutations of the choices selected in the 1^{th} sequence; that is, to

$$\frac{n!}{\prod_{k=0}^{N-1} n_k(1)!}$$

It follows that the average value of $B_S(1)$ is given by

$$E(B_S) = -\sum \left\{ \left[\frac{n!}{\prod_{k=0}^{N-1} n_k!} \right] \left[\prod_{k=0}^{N-1} [p(k)]^{n_k} \right] \times \left[\sum_{k=0}^{N-1} n_k \log_2 p(k) \right] \right\} \quad (13)$$

where the n_k and $p(k)$ are always positive and subject to the conditions

$$\sum_{k=0}^{N-1} n_k = n \quad (14)$$

$$\sum_{k=0}^{N-1} p(k) = 1 \quad (15)$$

The overall summation in Eq.(13) is made over all possible combinations of integral positive values of the n_k which satisfy Eq.(14).

In order to compute the values of $E(B_S)$ we begin by expressing the factorials in Eq.(13) by means of Stirling's formula (8)(9).

$$n! = \sqrt{2\pi n} n^n e^{-n} \quad (16)$$

valid for large values of n . We obtain then

$$\frac{n!}{\prod_{k=0}^{N-1} n_k!} \prod_{k=0}^{N-1} [p(k)]^{n_k} = \frac{\sqrt{2\pi n} n^n e^{-n}}{\prod_{k=0}^{N-1} \sqrt{2\pi n_k}} \prod_{k=0}^{N-1} \left\{ \left[\frac{p(k)}{n_k} \right]^{n_k} e^{n_k} \right\} = n^{-(N-1)} f(x) \quad (17)$$

where

$$f(x) = \left(\frac{n}{2\pi} \right)^{(N-1)/2} \left\{ \prod_{k=0}^{N-1} \left[\frac{p(k)}{x_k} \right]^{x_k} \right\}^n \left\{ \prod_{k=0}^{N-1} (x_k)^{-1/2} \right\} \quad (18)$$

The variables $x_k = n_k/n$ are always positive, smaller than unity and subject to the constraint

$$\sum_{k=0}^{N-1} x_k = 1 \quad (19)$$

It is convenient, at this point, to consider the function $f(x)$ as a continuous, rather than a discontinuous, function of the x_k and to transform the summation of Eq.(13) into an integral. We observe, in this regard, that when n_k varies from zero to n , x_k varies from zero to one. It follows that to a unit increment of n_k (n_k takes only integral values) corresponds an increment of x_k equal to $1/n$. Therefore, when n approaches infinity, to the unit increments of the n_k correspond the differentials $dx_k = 1/n$. In conclusion, the summation of Eq.(13) can be transformed (10) into an integral and Eq.(10) then becomes

$$H_N = - \lim_{n \rightarrow \infty} \int dx_1 \int dx_2 \dots \int dx_{N-1} \left[f(x) \sum_{k=0}^{N-1} x_k \log_2 p(k) \right] \quad (20)$$

The integration is extended over the region of the hyperplane defined by

Eq.(19), in which all the x_k are positive and smaller than one. It will be noted that in Eq.(20) x_0 is considered as a function of all the other x_k .

$$x_0 = 1 - \sum_{k=1}^{N-1} x_k \quad (21)$$

so as to limit the integration to the above-mentioned hyperplane.

To compute the integral appearing in Eq.(20), we observe first that the integral of $f(x)$ alone over the same region represents the summation of the probabilities of all possible messages consisting of n selections, provided, of course, that n is sufficiently large. Therefore, the integral of $f(x)$ must be equal to unity for all large values of n . On the other hand, as shown in Appendix I, $f(x)$ has a peak at a point which approaches $x_k = p(k)$ when n approaches infinity. The height of this peak is proportional to $(N-1)/n^2$. It follows that when n approaches infinity, $f(x)$ becomes a delta-function, or unit impulse, located at $x_k = p(k)$. The integral of Eq.(20) is, therefore, equal to the value for $x_k = p(k)$ of the rest of the integrand, that is, of the summation. Eq.(20) yields finally

$$H_N = - \sum_{k=0}^{N-1} p(k) \log_2 p(k) \quad (22)$$

which is then the average amount of information per N -order selection.

The conclusions which can be reached from the evaluation of the integral in Eq.(20) extend far beyond Eq.(22). It is easy to see that if the function

$$\sum_{k=0}^{N-1} x_k \log_2 p(k)$$

were any other finite function of the x_k , the limiting value of the integral would still be equal to the value of the function for $x_k = p(k)$. In other words, the expectation (or average value) of any function of the x_k is equal to the value of the function itself for $x_k = p(k)$. From a physical point of view, we can say that the ensemble of possible sequences of selections can be divided in two groups. The first group consists of sequences for which the frequencies x_k of occurrence of the different choices differ from the probabilities $p(k)$ of the choices by less than amounts which approach zero as $1/\sqrt{n}$ when n approaches infinity. The total probability of the sequences in this group approaches unity when n increases indefinitely, and therefore the number of sequences in this group approaches

$$M = \prod_{k=0}^{N-1} [p(k)]^{-np(k)} = 2^{nH_N} \quad (23)$$

The second group consists of all other sequences, and its total probability approaches zero when n approaches infinity.

The sequences of the first group are all equally probable and, therefore, the selection of one of them out of the group requires a number of binary selections equal to

$$\log_2 M = nH_N \quad (24)$$

In other words, the sequences of the first group can be represented by means of sequences of nH_N binary digits, that is H_N digits per N -order selection. All the other sequences together, regardless of the way in which they are represented, cannot increase by any finite amount, beyond H_N , the number of binary digits required on the average per N -order selection.

The expression for H_N obtained above indicates that H_N can be considered as the expectation of $\log_2 [1/p(k)]$. In other words, we may say that the selection of a particular choice k conveys an amount of information equal to the logarithm-base-two of the reciprocal of its probability. This interpretation is fundamental. It will be shown later to apply also to the general case of non-independent selections, in which case $p(k)$ will be substituted by the conditional probability that the k^{th} choice will be selected, based on the knowledge of all preceding selections.

It is easy to see from Eq.(22) that H_N vanishes only when all but one of the $p(k)$ are equal to zero, in which case the one different from zero must be equal to unity. In other words, H_N vanishes only when the choice which will be selected is known a priori with unity probability. In this instance, it is intuitively clear that no information is being transmitted. On the other hand, H_N is a maximum (as shown in Appendix I), when all the $p(k)$ are equal, that is, when there is no a priori knowledge at all about the selections. Under these circumstances, Eq.(22) reduces to Eq.(7), since $p(k) = 1/n$. The manner in which H_N varies with the probabilities of the choices is illustrated in Figure 3, for the particular case of $N = 2$.

The amount of information conveyed by a message of given length was defined above as the number of independent elementary (binary, equally likely) selections required, on the average, to specify such a message. The notion of a minimum number of binary selections required did not enter the definition. It should be intuitively clear, however, that the minimum number of binary selections required, on the average, to specify a message is equal to the average information conveyed, or, in other words, the number of

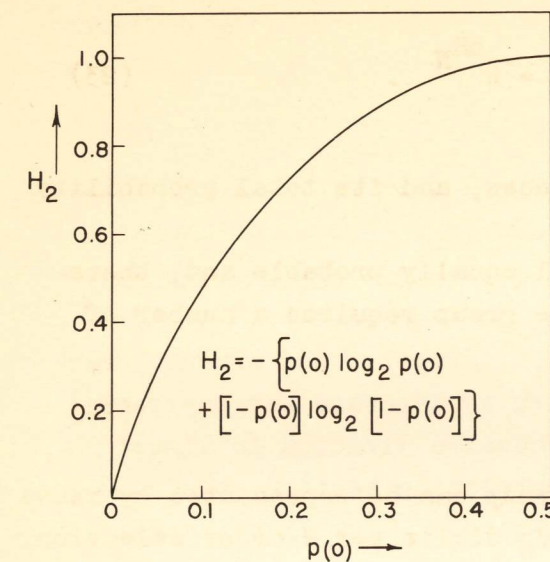


Fig. 3 The amount of information per binary selection as a function of the probability of either choice.

binary selections becomes a minimum when the selections are equally likely and independent. To prove this identity, we observe that the amount of information conveyed by a sequence of independent binary selections is a maximum when the selections are equally likely. Conversely, therefore, it is always possible to represent any sequence of m binary, not equally likely selections with a number of elementary selections smaller, on the average, than m . It follows that no binary representation of a message can be obtained with a number of selections smaller than the amount of information conveyed. It is clear, of course, that all message representations, which employ independent equally likely selections, require, on the average, the same number of selections. It will be shown later that a larger number of selections is required whenever non-independent selections are used.

It is appropriate to point out here that the mathematical form of Eq.(22) suggests a very interesting analogy between information and entropy, as expressed in statistical mechanics. In fact, H_N appears formally as the entropy of a system whose possible states have probabilities $p(k)$. For a physical interpretation of this analogy, the reader is referred to the work of Norbert Wiener (Ref. 1).

IV. Codes and Code Efficiency

The preceding sections have been devoted to the definition of the unit of information and to the computation of the average amount of information per selection in the case of messages consisting of sequences of independent N -order selections. It was pointed out in Section III that H_N represents the minimum number of binary selections required, on the average, to perform an N -order selection with given choice probabilities. Therefore, if we take

the number of binary selections employed as a basis for comparing different methods of conveying the same information, H_N represents a theoretical limit corresponding to maximum efficiency.

The knowledge of such a theoretical limit is extremely important, but perhaps even more important is the ability to approach this limit in practice. In our case, fortunately, the procedure followed in computing H_N (that is, the theoretical limit) indicates a convenient method for approaching this limit in practice. Let us consider again all the sequences of n N -order selections (in which, however, n may be a small integer), and arrange them in order of increasing probability. If we wish to separate any one particular sequence from the others by means of successive division in almost equally probable groups, as discussed in the preceding section, the number of divisions required, on the average, that is, $E(B_S)$, will be larger than nH_N . However, if we increase n , that is, the length of the sequences, we find that $E(B_S)/n$ keeps decreasing and approaches H_N when n approaches infinity. It must be kept in mind, in this regard, that $E(B_S)/n$ does not decrease necessarily in a monotonic manner, but may have an oscillatory behavior as a function of n .^{*} It follows that an increase of n may actually produce an increase of $E(B_S)/n$. For instance (as shown in Figure 4), in the case of $N = 2$, $p(0) = 0.7$, $p(1) = 0.3$, the value of $E(B_S)/n$ is 0.905 for $n = 2$, 0.909 for $n = 3$, and 0.895 for $n = 4$, the limiting value being $H_2 = 0.882$.

The above discussion indicates that, in transmitting a message consisting of a large number of selections, we should transmit the selections not individually, but in sequences of n as units, the number n being as large as permitted by practical considerations. The transmission of each of these units is then performed by means of sequences of binary selections corresponding in order to the successive divisions of the ensemble of all possible sequences of n N -order selections, as indicated in Figures 2, 4, and 5. It will be noted that, although the sequences of binary selections are not equal in length, it is always possible to identify the end of any of them in a long message. In fact, the first m selections of any sequence of length larger than m are always different from any of the sequences consisting of exactly m selections.

If it is desired to perform the transmission by means of N' -order selections (N' being any integer), we can proceed in the same manner as in the case of binary selections, the only difference being that we must divide successively the ensemble of all possible sequences in N' groups instead of just two. After each division, the groups containing the desired sequence

^{*} This fact was first pointed out to me by L. G. Kraft of this Laboratory.

Original Message	P(1)	Recoded Message	P(1) $B_S(1)$	Original Message	P(1)	Recoded Message	P(1) $B_S(1)$
00	0.49	0	0.49	0000	0.2400	00	0.480
01	0.21	10	0.42	0001	0.1030	010	0.309
10	0.21	110	0.63	0010	0.1030	011	0.309
11	0.09	111	0.27	0100	0.1030	100	0.309
$E(B_S) = 1.81$ $E(B_S)/2 = 0.905$ $\eta = 0.975$				1000	0.1030	1010	0.412
$E(B_S) = 2.726$ $E(B_S)/3 = 0.909$ $\eta = 0.972$				0011	0.0441	1011	0.1764
000	0.343	00	0.686	0110	0.0441	11000	0.2205
001	0.147	01	0.294	1100	0.0441	11001	0.2205
010	0.147	100	0.441	0101	0.0441	11010	0.2205
100	0.147	101	0.441	1001	0.0441	11011	0.2205
011	0.063	1100	0.252	1010	0.0441	11100	0.2205
101	0.063	1101	0.252	0111	0.0189	11101	0.0945
110	0.063	1110	0.252	1011	0.0189	111100	0.1134
111	0.027	1111	0.108	1101	0.0189	111101	0.1134
$E(B_S) = 3.5812$ $E(B_S)/4 = 0.895$ $\eta = 0.985$				1110	0.0189	111110	0.1134
$E(B_S) = 1.29$ $\eta = 0.725$				1111	0.0081	111111	0.0486

Fig. 4 Recoding of binary messages for $n = 2, 3, 4$; $p(0) = 0.7$, $p(1) = 0.3$, $H_2 = 0.882$.

Original Message	P(1)	Recoded Message	P(1) $B_S(1)$	Original Message	P(1)	Recoded Message	P(1) $B_S(1)$
00	0.81	0	0.81	0000	0.0550	0	0.6550
01	0.09	10	0.18	0001	0.0729	100	0.2187
10	0.09	110	0.27	0010	0.0729	101	0.2187
11	0.01	111	0.03	0100	0.0729	110	0.2187
$E(B_S) = 1.29$ $\eta = 0.725$				1000	0.0729	1110	0.2916
$E(B_S) = 1.594$ $\eta = 0.882$				0011	0.0081	111100	0.0486
000	0.729	0	0.729	0110	0.0081	1111010	0.0567
001	0.081	100	0.243	1100	0.0081	1111011	0.0567
010	0.081	101	0.243	0101	0.0081	1111100	0.0567
100	0.081	111	0.243	1010	0.0081	1111101	0.0567
011	0.009	11100	0.045	1001	0.0081	1111110	0.0567
101	0.009	11101	0.045	0111	0.0009	111111100	0.0081
110	0.009	11110	0.045	1011	0.0009	111111101	0.0081
111	0.001	11111	0.001	1101	0.0009	111111110	0.0081
$E(B_S) = 1.9691$ $\eta = 0.95$				1110	0.0009	1111111110	0.0090
$E(B_S) = 1.594$ $\eta = 0.882$				1111	0.0001	1111111111	0.0010

Fig. 5 Recoding of binary messages for $n = 2, 3, 4$; $p(0) = 0.9$, $p(1) = 0.1$, $H_2 = 0.468$.

will then be indicated by means of an N' -order selection.

The operation described above is, effectively, a change of code, that is, we may say, of the conventional language in which the message is written. Therefore this operation will be referred to as "message recoding". The advantage resulting from this recoding is conveniently expressed in terms of the code efficiency

$$\eta = \frac{H_N}{\log_2 N} \quad , \quad (25)$$

that is, the ratio of the information transmitted on the average per selection, to the information which could be transmitted with an equally likely selection of the same order. The efficiency of a binary code resulting from the recoding of sequences of N -order selections can be computed most conveniently in the form

$$\eta = \frac{nH_N}{E(B_S)} \quad , \quad (26)$$

where n is the number of N -order selections used in the recoding operation. Note that nH_N is the average amount of information per sequence of n N -order selections and $E(B_S)$ represents the amount of information which could be transmitted, on the average, by one of the sequences of binary selections in which the original sequences are recoded, if these binary selections were equally likely. If the new code is of N' order, we must substitute for $E(B_S)$ the product of $\log_2 N'$ by the number of N' -order selections required, on the average, to specify a sequence of n N -order selections.

A final remark must be made regarding the recoding operation. Since the process of successive divisions of an ensemble of sequences into equally probable groups cannot be carried out exactly, it is not clear at times whether one sequence should be included in one group or in another. Of course, we wish to perform all divisions in such a way as to obtain at the end the most efficient code. Unfortunately, no general rule could be found for determining at once how the divisions should be made in doubtful cases in order to obtain maximum code efficiency. However, so long as the divisions are made in a reasonable manner the resulting code efficiency will not differ appreciably from its maximum value.

We have implicitly assumed in the foregoing discussion that we know a priori the probabilities $p(k)$ of the choices for a message still to be transmitted. It seems appropriate at this point to discuss in some detail this assumption, since the practical value of the results obtained above depends entirely on its validity. When we state that the probability of a particular choice has a value $p(k)$ we mean that the frequency of occurrence of that choice in a message originating from a given source is expected to be close to $p(k)$. The longer is the message, the closer we expect the

frequency to approach $p(k)$. It must be clear, however, that we have no assurance that the frequency of occurrence will not differ considerably from the probability even in the case of a very long message, although such a situation is very unlikely to arise.

In practice, $p(k)$ must be estimated experimentally following the reverse process, that is, by inference from the measurement of the frequency in a number of sample messages. If the frequencies in the sample messages are reasonably alike, or, more precisely, if their values are scattered in the manner which might be expected on the basis of the length of the messages used, we may feel relatively safe in taking their average value as a good estimate of the probability. In other words, we may expect that the frequency in any other message originating from the same source will be reasonably close to the average value obtained. If this is the case, the source of such messages is said to have a stationary statistical character. We can conceive the case, however, in which the frequencies in the sample messages available are so widely scattered that hardly any significance can be attributed to their average value. Such a result may mean that the source has not a stationary statistical character, at least for practical purposes, in which case the concept of probability loses any physical significance. Fortunately, however, the sources of interest appear to have a stationary character for any practical purpose. In addition, the estimates of the probabilities of the choices do not need to be too close. It should be clear, in this respect, that the fact that a code has been designed for a particular set of choice probabilities does not mean that only messages with the same statistical character can be transmitted. It means only that such a code will transmit most efficiently, that is, with the smallest number of selections - messages with the choice frequencies equal to the assumed probabilities. Moreover, we can expect that the efficiency of transmission will not depend in a critical manner on the actual frequencies of the messages to be transmitted. A proof that this is actually the case is given below.

Suppose that a code which is optimum for a set of choice probabilities $p'(k)$ is used to transmit messages with choice probabilities $p(k)$. If we consider again all possible sequences of n selections, the expression for the number of binary selections required, on the average, to indicate one particular sequence, $E(B'_S)$, is still given by Eq.(13), where, however, the $p(k)$ which appear in the form $\log_2 p(k)$ should be changed into $p'(k)$. It follows that, in the limit when n approaches infinity, the number of binary selections per N -order selection will approach, according to Eq.(22), the value

$$H'_N = - \sum_{k=0}^{N-1} p(k) \log_2 p'(k) \quad . \quad (27)$$

It is clear from this equation that H_N^i varies rather slowly with any one of the $p'(k)$, unless the corresponding $p(k)$ is close to zero or unity. H_N^i is, of course, a minimum when $p'(k) = p(k)$. The case of $N = 2$ is illustrated in Figure 6 for $p(0) = 0.5$ and $p(0) = 0.7$. We may conclude, therefore, that the statistical characteristics assumed a priori can be rather different from those of the messages actually transmitted, without the efficiency being lowered too much.

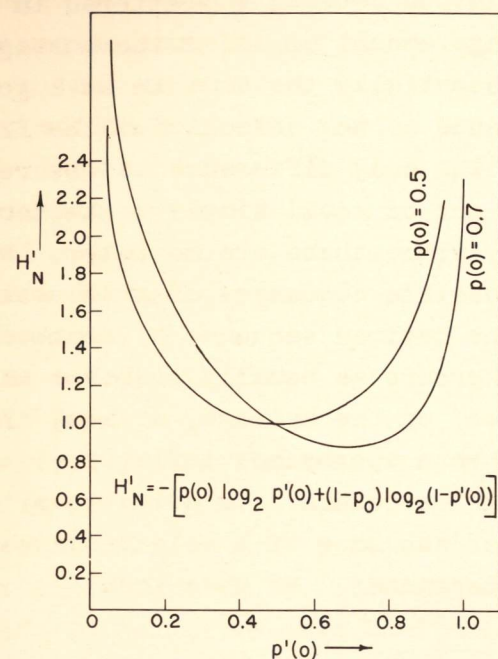


Fig. 6 Behavior of H_N^i as a function of $p'(0)$ for binary messages.

V. The Case of Non-Independent Selections

Thus far we have been considering only messages of a particularly simple type, namely, messages consisting of sequences of independent selections. Obviously, the statistical character of most practical messages is much more complex. Any particular selection depends generally on a number of preceding selections. For instance, in a written message the probability that a certain letter will be an "h" is highest when the preceding letter is a "t". In a television signal the light intensity of a certain element of a scanning line depends very strongly on the light intensities of the corresponding elements in the preceding lines and in the preceding frames. In fact, the light intensity is very likely to be almost uniform over wide regions of the picture and to remain unchanged for several successive frames.

The simplifying assumption that any one selection is independent of the preceding selections, although quite unrealistic, does not invalidate completely the results obtained in the preceding sections, but merely reduces their significance to that of first approximations. Intuitively, the average

amount of information conveyed by a sequence of given length is decreased by the a priori knowledge of any correlation existing between successive selections. Therefore, the value given by Eq.(22) will always be larger than the correct value for the average amount of information per selection, and the same is true of the code efficiency given by Eq.(25). Similarly, any recoding operation performed in the manner discussed in Section IV will result in a higher efficiency of transmission, but not so high as could be obtained by taking into account the correlation between successive selections.

The procedure for computing the average amount of information per selection and for recoding messages is still essentially the same as that used in Sections III and IV, even when the dependence of any selection on the preceding selections is taken into account. The only difference is that the probability of a particular sequence will not be equal simply to the product of the probabilities of the choices in it, since these are no longer independent. We must still arrange all the possible sequences of given length n in order of probability, and separate the desired sequence by successive divisions of the ensemble of sequences in groups as equally probable as possible. The number of divisions required, on the average, divided by the number n of selections will approach H_N when n approaches infinity.

Let $P_n(i)$ be the probability of the i^{th} sequence of n selections, and $H_S(n)$ the average amount of information per sequence of n selections when successive sequences are assumed to be independent. We have then

$$H_S(n) = - \sum_{i=0}^{N^n-1} P_n(i) \log_2 P_n(i) \quad (28)$$

Let us consider next a sequence of $n+1$ selections and let $P_{n+1}(i;k)$ be the conditional probability that the i^{th} sequence (of the $S = N^n$ sequences of n selections) is followed by the k^{th} choice (of the N). We have then

$$H_S(n+1) = - \sum_{k=0}^{N-1} \sum_{i=0}^{N^n-1} P_n(i) P_{n+1}(i;k) \log_2 P_n(i) P_{n+1}(i;k) \quad (29)$$

which, since

$$\sum_{k=0}^{N-1} P_{n+1}(i;k) = 1 \quad (30)$$

becomes

$$H_S(n+1) = H_S(n) - \sum_{k=0}^{N-1} \sum_{i=0}^{N^n-1} P_n(i) P_{n+1}(i;k) \log_2 P_{n+1}(i;k) \quad (31)$$

The increment of information resulting from the $(n+1)^{\text{th}}$ selection is then, on the average,

$$H_N(n+1) = - \sum_{k=0}^{N-1} \sum_{i=0}^{N^n-1} P_n(i) P_{n+1}(i;k) \log_2 P_{n+1}(i;k) . \quad (32)$$

Expressing now $H_S(n)$ in terms of the successive increments, we obtain

$$H_S(n) = \sum_{m=1}^n H_N(m) . \quad (33)$$

The final correct value of the average amount of information per selection can then be written in the form

$$H_N = \lim_{n \rightarrow \infty} (1/n) \sum_{m=1}^n H_N(m) . \quad (34)$$

To proceed further in our analysis, we must distinguish between two types of statistical character of practical importance. We shall say that the output of a certain source is statistically uniform if each and any selection depends in the same manner on the m^{th} preceding selection, as seems to be the case in a written message. We shall say that the output is periodically discontinuous if it is possible to divide any output sequence in sub-sequences of fixed and equal length, so that each and any selection depends in the same manner on the m^{th} preceding selection of the same sub-sequence but is independent of all selections of the preceding sub-sequences. This is the case when messages transmitted in succession are similar in character and equal in length but entirely unrelated to one another, as, for example, in facsimile transmission. The above differentiation of statistical character is not an exhaustive classification but only a characterization of two special cases of practical interest in which different results are obtained.

Considering now in more detail the increments of information $H_N(n+1)$, our intuition indicates that the average amount of information conveyed by any additional selection can be, at most, equal to the value obtained when the selection is independent of all preceding selections. Mathematically, it must be

$$H_N(n+1) \leq H_N(1) = - \sum_{k=0}^{N-1} p(k) \log_2 p(k) . \quad (35)$$

A proof of this inequality is given in Appendix II. In addition, it is

intuitively clear also that, in the case of uniform statistical character, the average amount of information conveyed by the $(n+1)^{\text{th}}$ selection of a sequence can be, at most, equal to the amount of information conveyed by the n^{th} selection, since the latter has less preceding selections on which to depend. Mathematically, we expect that, for statistically uniform sequences,

$$H_N(n+1) \leq H_N(n) . \quad (36)$$

A proof of this inequality is given also in Appendix II. Eq.(36) is satisfied with the equal sign when the $(n+1)^{\text{th}}$ selection, and therefore any following selection, depends only on the $n-1$ preceding selections.

Eq. (36) shows that the limit in Eq.(34) is approached in a monotonic manner. In addition, we expect $H_N(m)$ to approach monotonically a limit with increasing m , since the dependence of any selection on the preceding selections cannot extend, in practice, over an indefinitely large number of selections. Suppose, for instance, that this dependence extends only over the n_0-1 preceding selection. Then $H_N(m)$ becomes constant and equal to $H_N(n_0)$ when m is larger than n_0 , and Eq.(34) yields

$$H_N = H_N(n_0) . \quad (37)$$

This result is correct, of course, only in the case of statistically uniform sequences.

In the case of a periodically discontinuous statistical character, Eq.(36) is valid only when the n^{th} and the $(n+1)^{\text{th}}$ selections belong to the same sub-sequence. If this is not the case, the $(n+1)^{\text{th}}$ selection must be the first selection of a sub-sequence, and therefore is independent of all preceding selections. It follows that $H_N(m)$ is a periodic function of m with period equal to the length n'_0 of the sub-sequences, and that the limit of Eq.(34) is approached in an oscillatory manner. If we compute this limit by increasing n in steps equal to n'_0 , it is easily seen that Eq.(34) yields

$$H_N = (1/n'_0) \sum_{m=1}^{n'_0} H_N(m) , \quad (38)$$

a value larger than that given by Eq.(37), as was expected.

The recoding procedure in the case of messages consisting of non-independent selections is still the same as in the case of independent selections. The efficiency of transmission, still given by Eq.(25), increases (although not necessarily monotonically), with the number of selections used as units in the recoding process, and approaches unity when the number increases indefinitely. It is worth emphasizing that in the recoding process any sequence, even if statistically uniform, is considered as periodically discontinuous. In fact,

the groups of selections recoded as units are effectively sub-sequences which are treated as though they were totally unrelated. It follows that, if the recoding operation of a statistically uniform sequence is performed on groups of n_0 selections, the efficiency of transmission after recoding can be at most equal to

$$\eta(n_0) = \frac{n_0 H_N(n_0)}{\sum_{m=1}^{n_0} H_N(m)} \quad (39)$$

In the case of statistically discontinuous sequences, it would seem reasonable to make the number of selections in the recoding groups an integral fraction or multiple of the length of the sub-sequences.

A final remark is in order regarding the fitting of the recoding procedure to the statistical character of the messages to be transmitted. It may happen, as it does in the case of television signals, that the dependence of any one selection on the m^{th} preceding selection does not decrease monotonically when m increases, but behaves in an oscillatory manner. In this case, one should first reorder the selections before recoding, in such a manner that selections which are closely related take positions close to one another in the sequence. This idea of reordering the selections in the sequence can be generalized as follows. Any type of transmission of information can be considered as the transmission, in succession, of patterns in a two-dimensional or multi-dimensional space, time being one of the dimensions. Then the problem of ordering selections in an appropriate manner can be generalized to the problem of how best to scan these patterns. It is clear, on the other hand, that such a scanning problem is also at the root of the problem of reducing the bandwidth required by television signals. The generalized scanning problem seems to be, therefore, of fundamental practical, as well as theoretical, importance. However, no work can yet be reported on this subject.

VI. Practical Considerations

The main purpose of this paper was to provide a logical basis for the measurement of the rate of transmission of information. It has been shown that an appropriate measure for the rate of transmission in the case of sequences of selections can be provided by the minimum number of binary selections required, on the average, to indicate one of the original selections. We were then led naturally to consider the problem of actually performing the transmission of the original sequences by means of as few binary or higher-order selections as possible. We did not consider, however, the physical process corresponding to such selections — that is, their transmission by electrical means.

A convenient way of transmitting binary selections in a practical communication system is by means of pulses with two possible levels, one and zero. This is just the technique employed in pulse-code modulation. The maximum rate at which information can be transmitted in this case is simply equal to the number of pulses per second which can be handled by the electrical system — which we know to be proportional to the frequency band available. However, as soon as we start dealing with electrical pulses rather than logical operations like selections, an additional item must be considered in the problem, namely, the power required for the transmission. In the case of two-level pulses, the average power corresponding to the maximum rate of transmission of information is equal to one-half the pulse power, since the zero and one levels are equally probable.

If pulses with N rather than two levels equally spaced in voltage are used, the maximum rate of transmission is equal to $\log_2 N$ times the number of pulses per second which can be handled by the system. The average power required becomes, in this case,

$$W = \left(\frac{W_0}{N} \right) \sum_{k=0}^{N-1} k^2, \quad (40)$$

where W_0 is the power corresponding to the lowest (non-zero) voltage level.

The theoretical limit stated above for the rate of transmission of information certainly has practical significance when the limiting factors in the physical problem are the frequency band available and the number of pulse levels permitted by technical and economical considerations. It is to be noted, in this regard, that the effect of noise is here taken into account, to a first approximation, by setting a lower limit to the voltage difference between pulse levels, and therefore to W_0 . For a detailed discussion of the effect of noise, the reader is referred to the work of Shannon (5).

Eq.(40) shows, on the other hand, that the average power increases approximately as N^2 , while the rate of transmission is proportional only to $\log_2 N$. It follows that, if no limitation is placed on the frequency band employed, the smallest value of N should be used — that is, two. This value has, in addition, the very important practical advantage that the receiver is not required to measure a pulse, but only to detect the existence or the lack of a pulse. It might happen, on the other hand, that the frequency band and the average power are the limiting factors, while any reasonable number of pulse levels can be allowed. This case represents quite a different problem from those considered above, and the maximum rate of transmission of information is no longer obtained by making the pulse levels

(that is, the choices) equally probable, as one might think at first. For example, more than one unit of information per pulse can be transmitted with an average power $W = W_0/2$, by using pulses with three levels not equally probable. It seems worth while, therefore, to determine the maximum amount of information which can be transmitted per pulse, for a given average power W , a minimum level power W_0 , and an unlimited number of pulse levels equally spaced in voltage.

Let, therefore, $p(0)$ be the probability of occurrence of the zero level (no pulse), and $p(k)$ the probability of the k^{th} level. The amount of information per pulse is given by

$$H = - \sum_{k=0}^{\infty} p(k) \log_2 p(k) \quad , \quad (41)$$

and the average power by

$$W = W_0 \sum_{k=0}^{\infty} p(k) k^2 \quad . \quad (42)$$

We wish to maximize H with respect to the $p(k)$, subject to the condition expressed by Eq.(42) and, of course, the usual condition

$$\sum_{k=0}^{\infty} p(k) = 1 \quad . \quad (43)$$

The maximization procedure is carried out in Appendix III, and yields

$$H_{\text{max.}} = - \frac{W}{W_0} \left[\log_2 \left(\frac{p(1)}{p(0)} \right) + \log_2 p(0) \right] \quad ; \quad (44)$$

$$\frac{p(k)}{p(0)} = \left(\frac{p(1)}{p(0)} \right)^{k^2} \quad . \quad (45)$$

The values of $p(1)/p(0)$ and $p(0)$ are plotted in Figure 7 as functions of W/W_0 . The value of $H_{\text{max.}}$ is plotted as a function of the same variable in Figure 8. The latter curve shows, for instance, that the maximum amount of information per pulse for $W=W_0/2$ is 1.14, that is, 14 per cent higher than the value obtained by using two equally probable levels.

The procedure for approaching in practice the theoretical limit obtained above by appropriate recoding of the messages is very similar to that discussed in Section IV. It differs only in that the ensemble of all sequences of given length must now be divided in groups with probabilities $p(0)$, $p(1)$... $p(k)$..., instead of in equally probable groups. The number of pulse levels to be used in practice (it should be infinite in theory) must be selected

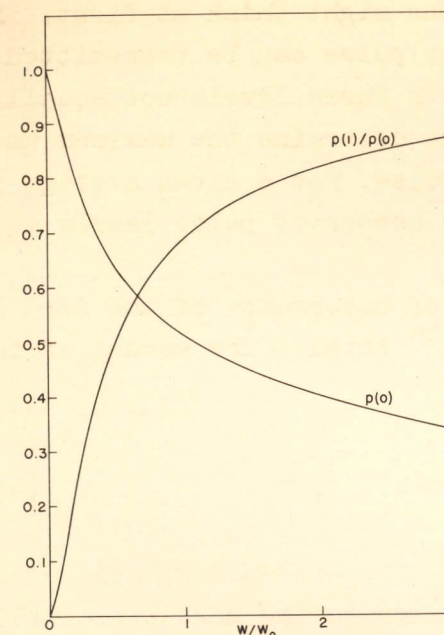


Fig. 7 Behavior of $p(1)/p(0)$ and $p(0)$ as functions of W/W_0 .

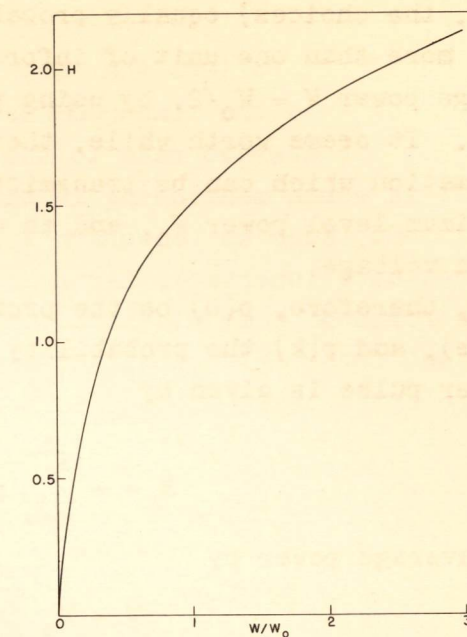


Fig. 8 Maximum information per pulse, $H_{\text{max.}}$, as a function of W/W_0 .

on a compromise basis, and the values of the $p(k)$ must then be readjusted, accordingly to make

$$\sum_{k=0}^{N-1} p(k) = 1 \quad .$$

In addition to the effect of limitations on the average power, another important practical consideration has been neglected in the preceding sections. All the types of recoding procedures suggested, for approaching in practice the theoretical limits derived above, require the use of devices capable of storing the information for a certain length of time in both the transmitter and the receiver. Such storage devices are needed to stretch or compress the time scale according to the probability of the group of original selections being recoded for transmission.

Satisfactory storage units are not yet available. In addition, even were they available, their use would undoubtedly add considerably to the complexity of communication systems. On the other hand, any substantial increase of transmission efficiency is fundamentally based on time stretching. In fact, since the logarithm of the probability of the choice or sequence of choices selected is a measure of the information conveyed by the selection (see p. 14), the time rate at which information is conveyed in actual signals may vary considerably with time. Even so, a communication

system must be able to handle at any time the peak rate which may be present in the signal. It follows that any system not employing storage devices to stretch or compress the time scale is bound to have an efficiency lower than the ratio of the average rate to the peak rate at which information is fed to it. It is worth mentioning in this connection that in certain types of communications, such as telegraph and television, the input and output signals do not have inherently fixed time scales. This is the same as saying that such forms of communication inherently incorporate storage devices. In the case of the telegraph, the written messages at the input and at the output are effectively storage devices. In the case of television, the image to be televised and the cathode-ray tube perform the same function.

Although no reduction of frequency band for a given noise level can be obtained without storage devices, appropriate coding may lead to some reduction of average power. This reduction can be obtained by assigning sequences of pulses requiring the smallest energy to the most probable messages, and vice versa. In the particular case of pulse-code modulation, for instance, this can be done as follows. We arrange all digit combinations in order of increasing amount of energy required and the sampling levels in order of decreasing probability. We assign then the digit combinations to the sampling levels in the resulting order. Such a coding method requires, however, more flexible coding and decoding units than those used in present-day systems.

Before concluding this section, it should be made clear that the improvement of transmission efficiency discussed above and the resulting possible reduction of bandwidth requirements for a given signal power have little to do with the bandwidth reduction obtained by means of the Vocoder or other similar schemes. The Vocoder (2), for instance, does not improve the efficiency of transmission, but achieves a reduction in bandwidth by eliminating that part of the speech signal which is not strictly necessary for the mere understanding of the words spoken. Obviously, the recoding of messages according to their statistical character and the elimination of unnecessary information represent fundamentally different but equally important contributions to the solution of the bandwidth-reduction problem.

Appendix I

Maximization of $f(x)$

In determining the values of the x_k for which $f(x)$, as given in Eq.(18), is a maximum, it is more convenient to operate on the function

$$\varphi(x) = \ln f(x) \quad (I-1)$$

whose maxima and minima at non-singular points coincide with those of $f(x)$. The x_k are the variables in the maximization process, but are subject to the constraint

$$\sum_{k=0}^{N-1} x_k = 1 \quad (I-2)$$

Using Lagrange's method, we equate to zero the partial derivatives, with respect to the x_k of the function

$$\varphi(x) + \lambda \sum_{k=0}^{N-1} x_k, \quad (I-3)$$

where λ is a constant to be determined later. We obtain then N equations of the form

$$n[\ln p_k - (1 + \ln x_k)] - \frac{1}{2x_k} + \lambda = 0 \quad (I-4)$$

It is clear that when n approaches infinity these equations can be satisfied simultaneously only when $x_k = p_k$, in which case Eq.(I-2) is also satisfied. In addition, the function $f(x)$ is neither discontinuous nor a minimum at the point $x_k = p_k$, so that the existence of a maximum at this point does not require any further mathematical proof.

Maximization of H_N

The function H_N given by Eq.(22) must be maximized with respect to the $p(k)$ which are, of course, subject to the constraint

$$\sum_{k=0}^{N-1} p(k) = 1 \quad (I-5)$$

Following the same method as above, we obtain N equations of the form

$$\frac{\partial}{\partial p(k)} \left[H_N + \lambda \sum_{k=0}^{N-1} p(k) \right] = -\frac{1}{\ln 2} [1 + \ln p(k)] + \lambda = 0. \quad (I-6)$$

This set of equations can be satisfied only if all the $p(k)$ are equal. Again it is clear that H_N is neither discontinuous nor a minimum when all the $p(k)$ are equal, and therefore it must be a maximum.

Proof That $H_N(n+1) \leq H_N(1)$

We wish to show, first, that the increment of the amount of information

$$H_N(n+1) = - \sum_{k=0}^{N-1} \sum_{i=0}^{N^n-1} P_n(i) P_{n+1}(i;k) \log_2 P_{n+1}(i;k) \quad (\text{II-1})$$

is a maximum when $P_{n+1}(i;k) = p(k)$, the probability of the k^{th} choice, that is, when the additional selection is independent of all preceding selections. Mathematically, we must maximize the function $H_N(n+1)$ with respect to the N^{n+1} variables $P_{n+1}(i;k)$, subject to the conditions

$$\sum_{i=0}^{N^n-1} P_n(i) P_{n+1}(i;k) = p(k), \quad (\text{II-2})$$

and

$$\sum_{k=0}^{N-1} P_{n+1}(i;k) = 1 \quad (\text{II-3})$$

Following Lagrange's method, we equate to zero the derivatives with respect to the $P_{n+1}(i;k)$ of the function

$$H_N(n+1) + \sum_{i=0}^{N^n-1} \sum_{k=0}^{N-1} \lambda_i P_{n+1}(i;k) + \mu_k P_n(i) P_{n+1}(i;k) \quad (\text{II-4})$$

where the λ_i and μ_k are constants to be determined later. We obtain then, for each pair of values of i and k , an equation of the form

$$P_n(i) [1 + \ln P_{n+1}(i;k)] - \lambda_i - \mu_k P_n(i) = 0 \quad (\text{II-5})$$

The solution of the N^{n+1} equations of this type, together with Eqs. (II-2) and (II-3), is clearly

$$\lambda_i = P_n(i) \quad (\text{II-6})$$

$$\mu_k = \ln P_{n+1}(i;k) = \ln p(k) \quad (\text{II-7})$$

Therefore, the increment of information H_N is a maximum for $P_{n+1}(i;k) = p(k)$, since this is the only point at which a maximum can exist and a maximum must exist at some point. This result can also be stated in the form

$$H_N(n+1) \leq H_N(1) \quad (\text{II-8})$$

where

$$H_N(1) = - \sum_{k=0}^{N-1} p(k) \log_2 p(k) \quad (\text{II-9})$$

is the average amount of information per selection, that is, the average increment of information, when each selection is independent of all preceding selections.

Proof That $H_N(n+1) \leq H_N(n)$

Let us consider a sequence of n selections as consisting of a first selection followed by a sequence of $n-1$ selections. Let $P_n(h;j)$ be the conditional probability that the selection of the h^{th} choice is followed by the selection of the j^{th} sequence from the N^{n-1} possible sequences of $n-1$ selections. Let also $P_{n+1}(h,j;k)$ be the conditional probability that the k^{th} choice is selected after the h^{th} choice and the j^{th} sequence. We shall still indicate with $p(k)$ the probability of the k^{th} choice and, similarly, with $p(h)$ the probability of the h^{th} choice. Using these new symbols, Eq. (II-1) becomes

$$H_N(n+1) = - \sum_{h=0}^{N-1} \sum_{j=0}^{N^{n-1}-1} \sum_{k=0}^{N-1} p(h) P_n(h;j) P_{n+1}(h,j;k) \log_2 P_{n+1}(h,j;k) \quad (\text{II-10})$$

We wish to show that, for a statistically uniform sequence, $H_N(n+1)$ is a maximum when $P_{n+1}(h,j;k)$ is independent of h . Mathematically, we must again maximize the function $H_N(n+1)$ with respect to the N^{n+1} variables $P_{n+1}(h,j;k)$, subject to the conditions

$$\sum_{k=0}^{N-1} P_{n+1}(h,j;k) = 1 \quad (\text{II-11})$$

and

$$\sum_{h=0}^{N-1} p(h) P_n(h;j) P_{n+1}(h,j;k) = P_{n-1}(j) P_n(j;k) \quad (\text{II-12})$$

where $P_{n-1}(j)$ is the probability of the j^{th} sequence of $n-1$ selections, and $P_n(j;k)$ is the conditional probability that the k^{th} choice will be selected after the j^{th} sequence. These two probabilities must, in turn, satisfy the condition

$$\sum_{j=0}^{N^{n-1}-1} P_{n-1}(j) P_n(j;k) = p(k) \quad , \quad (\text{II-13})$$

which, however, does not concern us, since it does not involve directly the $P_{n+1}(h,j;k)$. It must be clear, on the other hand, that the $P_n(j;k)$ are kept constant in the maximization process. In other words, the dependence of the $(n+1)^{\text{th}}$ selection on the $n-1$ preceding selection is fixed in this case, while in the case discussed previously it was allowed to vary. In addition, since we are dealing with a statistically uniform sequence, the $(n+1)^{\text{th}}$ selection depends on the $n-1$ preceding selections in the same manner as the n^{th} selection depends on its $n-1$ preceding, that is, on all the preceding selections.

Proceeding in the same manner as in the proof that $H_N(n+1) \leq H_N(1)$, we find that, for given $P_n(j;k)$, the $P_{n+1}(h,j;k)$ make $H_N(n+1)$ a maximum when they are independent of h , that is, of the first selection of the sequence. Mathematically speaking, the maximum occurs when $P_{n+1}(h,j;k) = P_n(j;k)$. It follows that Eq.(II-10) yields, with the help of Eq.(II-11),

$$H_N(n+1)_{\text{max.}} = - \sum_{j=0}^{N^{n-1}-1} \sum_{k=0}^{N-1} P_{n-1}(j) P_n(j;k) \log_2 P_n(j;k) = H_N(n) \quad (\text{II-14})$$

This result can also be stated in the form

$$H_N(n+1) \leq H_N(n) \quad . \quad (\text{II-15})$$

It must be clear that, in the case of non-statistically uniform sequences, $P_n(j;k)$ may be an entirely different function than that representing the dependence of the n^{th} selection on the first $n-1$ selections of the sequence, since, for instance, the $(n+1)^{\text{th}}$ selection can be entirely independent of the preceding selections while the n^{th} selection is not. It follows, in this latter case, that Eq.(II-14) is not valid, and $H_N(n+1)$ can be as large as $H_N(1)$.

Appendix III

We wish to maximize the average amount of information per pulse, H , for a given average power and an unlimited number of pulse levels equally spaced in voltage. Mathematically, this amounts to maximizing the function given by Eq.(41), subject to the conditions imposed by Eqs.(42) and (43). Following Lagrange's method, as in Appendices I and II, we obtain an infinite set of equations of the form

$$1 + \ln p(k) = \lambda + k^2 \mu \quad , \quad (\text{III-1})$$

where λ and μ are indeterminate constants. The first of these constants, λ , can be eliminated by subtracting the equation with $k=0$ from all the other equations of the set, which take then the form

$$\ln p(k) - \ln p(0) = k^2 \mu \quad . \quad (\text{III-2})$$

The remaining constant, μ , is then eliminated by subtracting k^2 times Eq.(III-2) - with $k=0$ - from the other equations of the same set. We obtain in this manner a set of equations of the form

$$[\ln p(k) - \ln p(0)] - k^2 [\ln p(1) - \ln p(0)] = 0 \quad (\text{III-3})$$

It follows that

$$\frac{p(k)}{p(0)} = \left[\frac{p(1)}{p(0)} \right]^{k^2} \quad . \quad (\text{III-4})$$

Eqs.(42) and (43) can now be written in the forms

$$p(0) \sum_{k=1}^{\infty} k^2 \left[\frac{p(1)}{p(0)} \right]^{k^2} = \frac{W}{W_0} \quad , \quad (\text{III-5})$$

and

$$p(0) \sum_{k=1}^{\infty} \left[\frac{p(1)}{p(0)} \right]^{k^2} = 1 \quad . \quad (\text{III-6})$$

The values of $p(1)/p(0)$ are plotted in Figure 7 as functions of W/W_0 . From these values, the $p(k)$ are immediately obtained by means of Eq.(III-4).

The maximum value of the average amount of information H can now be obtained without difficulty by substituting for the $p(k)$ in Eq.(41) the values determined above. We have then, after appropriate manipulation of the equation,

$$\begin{aligned} H_{\text{max.}} &= - p(0) \left\{ \sum_{k=0}^{\infty} \left[\frac{p(k)}{p(0)} \right] \left[\log_2 \frac{p(k)}{p(0)} + \log_2 p(0) \right] \right\} \\ &= - \log_2 p(0) - p(0) \sum_{k=1}^{\infty} \left[\frac{p(1)}{p(0)} \right]^{k^2} \log_2 \left[\frac{p(1)}{p(0)} \right]^{k^2} \\ &= - \log_2 p(0) - p(0) \left\{ \sum_{k=1}^{\infty} k^2 \left[\frac{p(1)}{p(0)} \right]^{k^2} \right\} \log_2 \frac{p(1)}{p(0)} \quad . \end{aligned} \quad (\text{III-7})$$

Using now Eq.(III-5), we obtain finally

$$H_{\max.} = - \left[\frac{W}{W_0} \log_2 \frac{p(1)}{p(0)} + \log_2 p(0) \right] \quad . \quad (\text{III-8})$$

The value of $H_{\max.}$ is plotted in Figure 8 as a function of W/W_0 , using the values of $p(1)/p(0)$ and $p(0)$ given in Figure 7.

Acknowledgment

The author wishes to express his gratitude to Dr. Wiener and Dr. Shannon for discussing with him their work and allowing him to read their manuscripts before publication. In addition, he wishes to thank his colleagues in the Communications Group of the Research Laboratory of Electronics for their constructive criticisms as well as for their very helpful suggestions.

References

- (1) N. Wiener, "The Extrapolation, Interpolation, and Smoothing of Stationary Time Series", N.D.R.C. Report, M.I.T. (Feb. 1, 1942) (being reprinted by the Technology Press).
- (2) H. Dudley, J. Acous. Soc. Am., 11, 169(1939).
- (3) J. C. R. Licklider and I. Pollack, J. Acous. Soc. Am., 20, 42 (1948).
- (4) N. Wiener, "Cybernetics", New York, The Technology Press, John Wiley and Sons (1948).
- (5) C. E. Shannon, "A Mathematical Theory of Communication", B.S.T.J. 27 (July and Oct. 1948).
- (6) R. V. L. Hartley, "Transmission of Information", B.S.T.J. (July 1928).
- (7) W. Tuller, "Theoretical Limitations on the Rate of Transmission of Information", Sc. D. Thesis in E.E. Dept., M.I.T. (June 1948).
- (8) J. V. Uspensky, "Introduction to Mathematical Probability", McGraw-Hill, New York (1937) App. I, Sec. 2.
- (9) T. C. Fry, "Probability and Its Engineering Uses" (Van Nostrand, New York, 1928), p. 103.
- (10) Uspensky, op. cit., App. I, Sec. 1.

MONOSTROPHIC CODES

A Thesis
Presented to
the Faculty of the Department of Electrical Engineering
Northeastern University

by
DALE SHERWOOD COCKLE

June, 1964

MONOSTROPHIC CODES

A Thesis

Presented to

the Faculty of the Department of Electrical Engineering
Northeastern University

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science

by

Dale Sherwood Cockle

June, 1964

NORTHEASTERN UNIVERSITY

Graduate School

Thesis Title: Monostrophic Codes

Author: Dale Sherwood Cockle

Department: Electrical Engineering

Approved for Thesis Requirements of the Master of Science
Degree in Electrical Engineering

Marcello J. Carrabes
Thesis Advisor

Arthur E. Fitzgerald
Chairman of the Department

Graduate School Notified of Acceptance

George W. Hankinson
Program Director

ACKNOWLEDGEMENTS

This thesis contains the contributions of many organizations and individuals, whose material, advice and encouragement made this endeavor possible. I have given bibliographical reference to those sources which I consider reasonably accessible to the reader. However, I wish to express my personal gratitude here to the following organizations for their kind contributions.

C. P. Clare and Co.	Chicago, Ill.
Computer Control Co., Inc.	Framingham, Mass.
Datex Corp.	Monrovia, Calif.
Digital Equipment Corp.	Maynard, Mass.
Engineered Electronics Co.	Santa Ana, Calif.
General Precision, Inc. Commercial Computer Div.	Burbank, Calif.
Kearfott Div.	Little Falls, N. J.
Northern Precision Laboratories, Inc.	Franklin Lakes, N. J.
Perkin-Elmer Corp.	Norwalk, Conn.
Raytheon Co.	Lexington, Mass.
Theta Instrument Corp.	Saddlebrook, N. J.
W. & L. E. Gurley	Troy, N. Y.
Wang Laboratories, Inc.	Tewksbury, Mass.
Wayne-George Corp.	Newton, Mass.

My most grateful thanks go to the wonderful people at Digital Equipment Corporation, in particular, my dear friends Norton Ruderman and John O'Connell (Digital

Module Application Engineers) under whose patient tutelage I learned the practical side of digital logic implementation during my co-operative work periods, and to the personnel of the Advertising Department whose efforts resulted in the expert reproduction of this thesis.

To William E. Walker, New England District Manager of Engineered Electronics Company, I offer my thanks for his personal contributions.

I wish to express my appreciation to P. J. Lawrence, Publications Manager of the Datex Corporation, for his kind and prompt assistance.

I am greatly indebted to my thesis advisor, Marcello J. Carrabes, for his many hours of expert instruction in digital logic and techniques, and for the inspiration he gave which sustained me through the many long hours of work preparing this thesis.

To my wife, Joanna, whose understanding, encouragement, and long hours of typing contributed materially to the timely completion of this thesis, goes my most sincere and humble thanks.

D. S. C.

TABLE OF CONTENTS

Chapter I	Introduction.....	1
Chapter II	General.....	4
Chapter III	Monostrophic Code Synthesis.....	10
Chapter IV	Monostrophic-Polystrophic Code Conversion.....	20
Chapter V	Generation of Monostrophic Codes.....	42
Chapter VI	Conclusion.....	62
Appendix I	Four Bit Gray and Pseudo-Gray Codes	
Appendix II	Legend of symbols Used in this Paper	
Appendix III	Exclusive Or/Ring Sum	
Bibliography		

CHAPTER I. INTRODUCTION

Monostrophic codes, binary codes in which only one bit changes from count to count, are commonly found in applications where elimination of ambiguities during transition from one count to the next is desired.

The most common monostrophic code is the Gray code. Figure I.1 is an example of a 4 bit Gray code with the normally employed binomially weighted (2^{n-1} , - - -, 8,4,2,1) binary and decimal equivalents.

DECIMAL	BINARY	
	BINOMIAL 8 4 2 1	GRAY
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Fig. I.1

Digital logic engineers undoubtedly have unknowingly used the Gray code or other monostrophic codes as a tool for logic simplification and/or sequential switching

synthesis. The Karnaugh Map, which will be used as a tool in this paper, is such an application. Figure I.2 is an example of a 4 variable Karnaugh Map in which there is a change in only one variable between adjacent compartments.

	AB				
CD		00	01	11	10
00					
01					
11					
10					

4 VARIABLE KARNAUGH MAP

Fig. I.2

Other common applications of monostrophic codes are found in analog - digital and digital - analog conversion devices.

Although the Gray code is nothing new it is difficult to find references which comprehensively discuss monostrophic codes, particularly those which are not Gray. Therefore, by means of this paper, this author is attempting to present a compilation of information pertaining to monostrophic codes, the sources being found in texts, trade publications, manufacturers' product bulletins and applications notes, and that information which, because of the meager quantities found in the aforementioned sources,

has been self-generated. The succeeding chapters will discuss, with respect to monostrophic codes, the following topics:

- (1) Synthesis of monostrophic codes.
- (2) Conversion between monostrophic and polystrophic codes.
- (3) Generating monostrophic codes using electronic and switching logic elements (counters).

Preceding the above mentioned topics terms used in this paper will be defined.

CHAPTER II, GENERAL

In order to be precise, this author has attempted to employ in this paper terms that have commonly accepted definitions in the digital field. In so doing many small disparities have arisen that might lead to confusion if used in this paper without defining the terms herein. Therefore, I will commit the common sin of "defining definitions" before proceeding with the substance of this paper.

The Gray code many times is considered to be any monostrophic code. On the other hand, the terms Gray code and reflected code are often used synonymously. For the purpose of further discussion let us clear up this disparity with the following definitions because, clearly, not all monostrophic codes are reflected codes.

GRAY CODE: Normally the Gray code is considered to be a specific n bit counting sequence of 2^n counts having the characteristics of being non-weighted, monostrophic and reflected (see definition of reflected codes below), and represents a specific ordered numbering system of 2^n counts. In Chapter I an example of a 4 bit Gray code was given. The specific counting sequence may be explained as follows: the counting sequence for the least significant bit is 0110 repeated $2^{n-1}/4$ times, with the more significant bits going through the same sequence at half the rate as

the next less significant bit, and the most significant bit going through only half the 0110 sequences as shown in Fig. II.1.

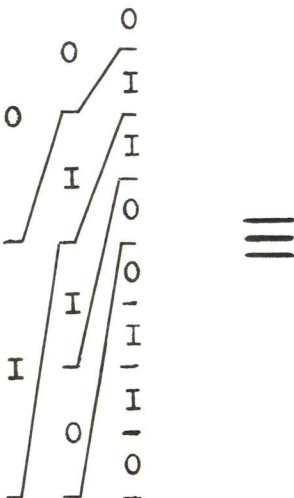
$n = 3$	GRAY	BINOMIAL BINARY	DECIMAL
	0 0 0	0 0 0	0
	0 0 1	0 0 1	1
	0 1 1	0 1 0	2
	0 1 0	0 1 1	3
	1 1 0	1 0 0	4
	1 1 1	1 0 1	5
	1 0 1	1 1 0	6
	1 0 0	1 1 1	7

Fig. II.1

PSEUDO-GRAY CODE AND NORMALIZING: By complementing and/or permuting bits of a Gray code, other monostrophic codes can be formed which satisfy all the required conditions for being a Gray code except the binomial binary equivalent and, part of the time, the counting sequence. In the example of Fig. II.2, ABCD is a Gray code with its binomial binary equivalent of WXYZ. EFGH is not Gray, but by complementing E and permuting E and F we have converted it to a Gray code. This conversion process will hereafter be called normalizing, and a monostrophic code that can be normalized to a Gray code will be called a pseudo-Gray code.

n = 4	GRAY	ANORMAL GRAY	BINOMIAL BINARY	DECIMAL
	<u>A B C D</u>	<u>E F G H</u>	<u>W X Y Z</u>	
	0 0 0 0	1 0 0 0	0 0 0 0	0
	0 0 0 1	1 0 0 1	0 0 0 1	1
	0 0 1 1	1 0 1 1	0 0 1 0	2
	0 0 1 0	1 0 1 0	0 0 1 1	3
	0 1 1 0	0 0 1 0	0 1 0 0	4
	0 1 1 1	0 0 1 1	0 1 0 1	5
	0 1 0 1	0 0 0 1	0 1 1 0	6
	0 1 0 0	0 0 0 0	0 1 1 1	7
	1 1 0 0	0 1 0 0	1 0 0 0	8
	1 1 0 1	0 1 0 1	1 0 0 1	9
	1 1 1 1	0 1 1 1	1 0 1 0	10
	1 1 1 0	0 1 1 0	1 0 1 1	11
	1 0 1 0	1 1 1 0	1 1 0 0	12
	1 0 1 1	1 1 1 1	1 1 0 1	13
	1 0 0 1	1 1 0 1	1 1 1 0	14
	1 0 0 0	1 1 0 0	1 1 1 1	15

$$A = F, B = \bar{E}, C = G, D = H$$

Fig. II.2

Appendix I contains an exhibit of all combinations of complements and all permutations for a 4 bit Gray code with a table of comparisons of counting sequences. A 4 bit Gray code was used in Appendix I because of its ability to be used directly most often and, at the same time, make the point clear.

REFLECTIVE CODES: For a code to be reflected it must represent a numbering system of a radix r , and by complementing a certain bit (same bit for all counts and usually the most significant bit) will yield the $r-1$'s complement of the original count. In Figure II.3a a 3 bit reflective code is shown representing a Radix 8 numbering system. Notice that the reflected code is not

POLYSTROPHIC	#	7's COMP
0 0 0	0	7
0 0 I	1	6
0 I 0	2	5
0 I I	3	4
I I I	4	3
I I 0	5	2
I 0 I	6	1
I 0 0	7	0

a.

GRAY	#	7's COMP
0 0 0	0	7
0 0 I	1	6
0 I I	2	5
0 I 0	3	4
I I 0	4	3
I I I	5	2
I 0 I	6	1
I 0 0	7	0

b.

Fig. II.3

monostrophic. In Fig. II.3b the code is monostrophic and also Gray. By complementing the most significant bit in either reflected code of Fig. II.3, the $r-1$'s complement ($r=8$) is the result. The monostrophic and polystrophic reflected codes were shown to clearly point out that Gray and reflected are not synonymous, but simply that the Gray code (but not all monostrophic codes) is a reflected code.

CYCLIC: The Gray code is also called a cyclic code.

The Modern Dictionary of Electronics put out by Howard Sams defines a cyclic code as "any binary code that changes only one bit when going from one number to the number immediately following". This, in effect, is synonymous with monostrophic. This author has found that usage of the word cyclic in the digital industry means different things to different people. Some people interpret cyclic as returning upon itself by the same path or synonymously with reflective rather than as a monostrophic characteristic. Therefore, for the sake of being precise, I will avoid the use of the term cyclic.

MONOSTROPHIC CLOSURE: If the transition between the first and last counts of a counting sequence requires only one bit to change we have a monostrophic closure. The Gray code and pseudo-Gray codes possess this characteristic, but possession of this characteristic is not required for a code to be monostrophic.

FULL COUNT: A full count, as used in this paper, signifies a counting n bit sequence to which all of the 2^n counts have an assigned numerical value. We can see that a full count usually goes hand-in-hand with a numbering system having a radix of 2^n . A full counting sequence may be made monostrophic and/or reflected with or without monostrophic closure.

FORESHORTENED COUNT: When all 2^n combinations are not used to form a counting sequence we have a foreshortened

count. A foreshortened counting sequence may be made monostrophic, but to be reflective and/or close monostrophically the counting sequence must contain an even number of counts.

CHAPTER III. MONOSTROPHIC CODE SYNTHESIS

This chapter will discuss the synthesis of Gray, pseudo-Gray and other full and foreshortened count monostrophic codes, and will demonstrate the use of Karnaugh Maps and path diagrams as synthesis tools.

GRAY CODE: Because the Karnaugh Maps and path diagrams utilize the Gray code, the synthesis of the Gray code must be described independently. In effect, the definition of the Gray code given in Chapter II, describes the synthesis of the Gray code. For an n bit code starting with the least significant bit, form the sequence 0110 $2^{n/4}$ times. The next significant bit's sequence is the same but at half the rate, etc., until we get to the most significant bit which goes through only half the 0110 sequence, the first half of the sequence being 0's and the last half being 1's. This synthesis process, shown for a 4 bit code in Fig.III.1, yields a monostrophic code which monostrophically closes. If the following additional characteristics are present we have synthesized a reflected code which is Gray:

1. All 2^n counts represent a count of a numbering system.
2. All bits non-asserted (all 0's) represents zero.
3. Represented numbering system counts progressively from 0 to 2^n-1 .

KARNAUGH MAPS: Any Karnaugh Map is a table of all possible combinations of an n bit binary word so arranged

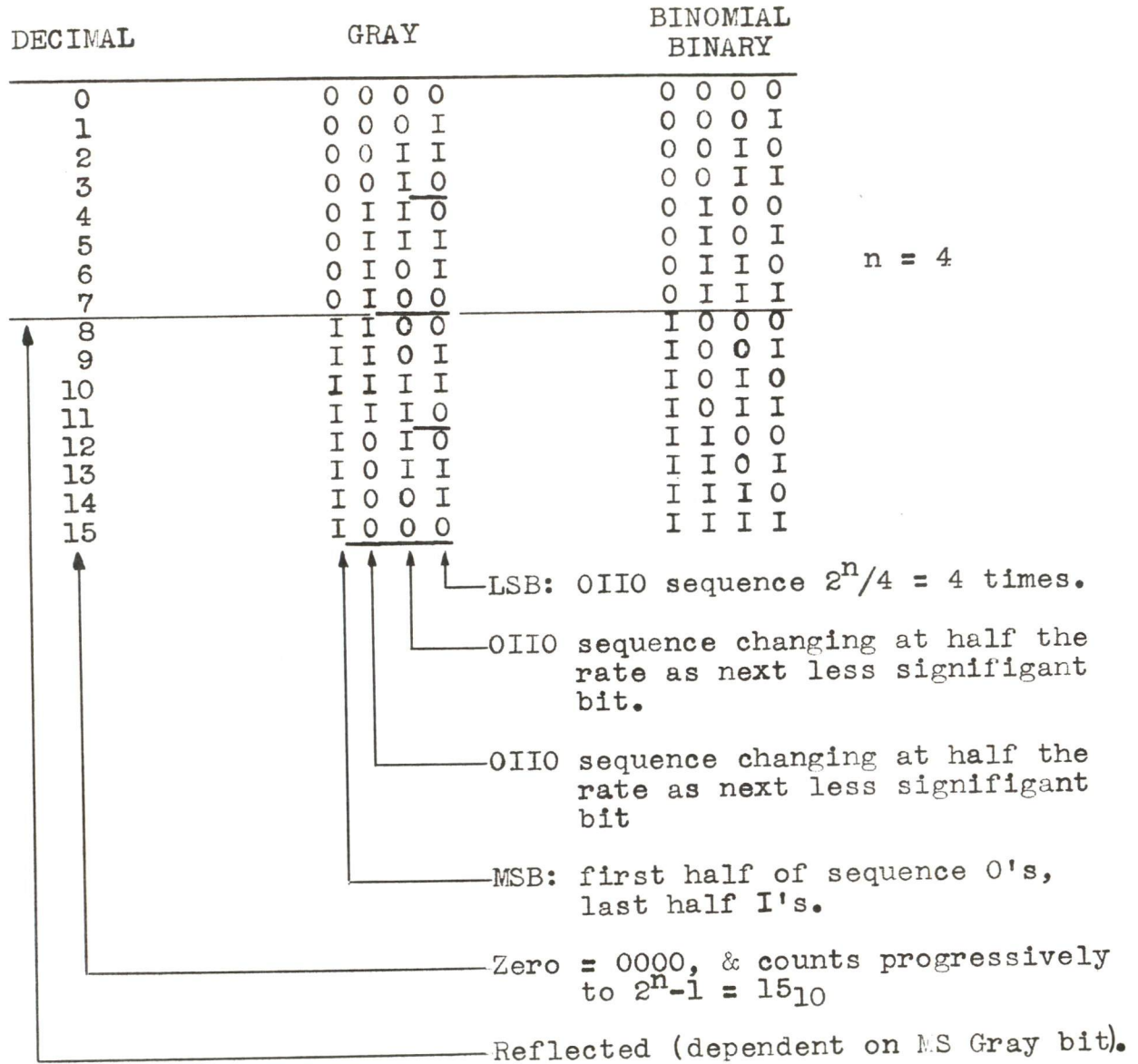


Fig. III.1

that there is only one bit change when going from any compartment to an adjacent or "mirrored" compartment. For maximum utility the map is arranged so that approximately half the variables (usually the more significant bits) are on the horizontal margin and the remaining variables (usually the lesser significant bits) are on the

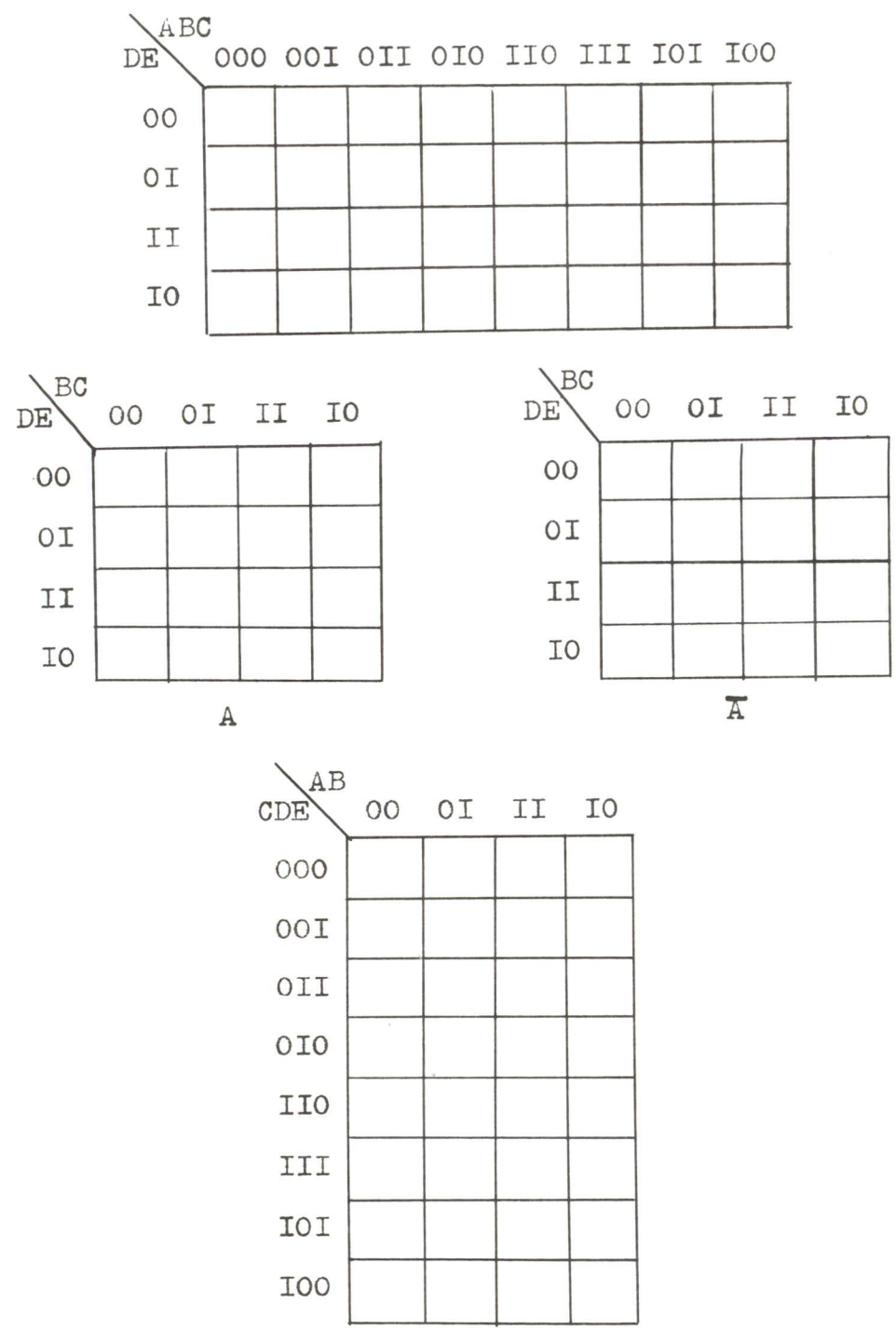
vertical margin, e.g., a $2^2 \times 2^3$ or $2^3 \times 2^2$ configuration represents the 32 combinations of a 5 variable (bit) word.

In order that there be only one bit change between adjacent and "mirrored" compartments a reflected monostrophic code is used on the margins. In fact, the counting sequences of a Gray code are used on the margins. FigIII.2 shows 3 variations of a 5 variable Karnaugh Map.

Any path through a Karnaugh Map describes a monostrophic code.

PATH DIAGRAM: All the possible paths through a Karnaugh Map are difficult to see because one may jump to "mirrored" compartments, particularly if there are more than 4 variables (bits). To solve this problem the path diagram may be employed. The path diagram's use also simplifies the incorporation of code requirements during synthesis. This point will become evident in subsequent chapters.

Basically, the path diagram for an n bit code is formed by equally spacing n points in a circular pattern. The points are then numbered, starting with the n bit Gray coded zero, and progressing around the circle with the Gray code counting sequence. Then all those points are connected which differ by only one variable. For an n bit code each point will radiate n lines. The pattern generated is symmetric aiding in drawing a path diagram.

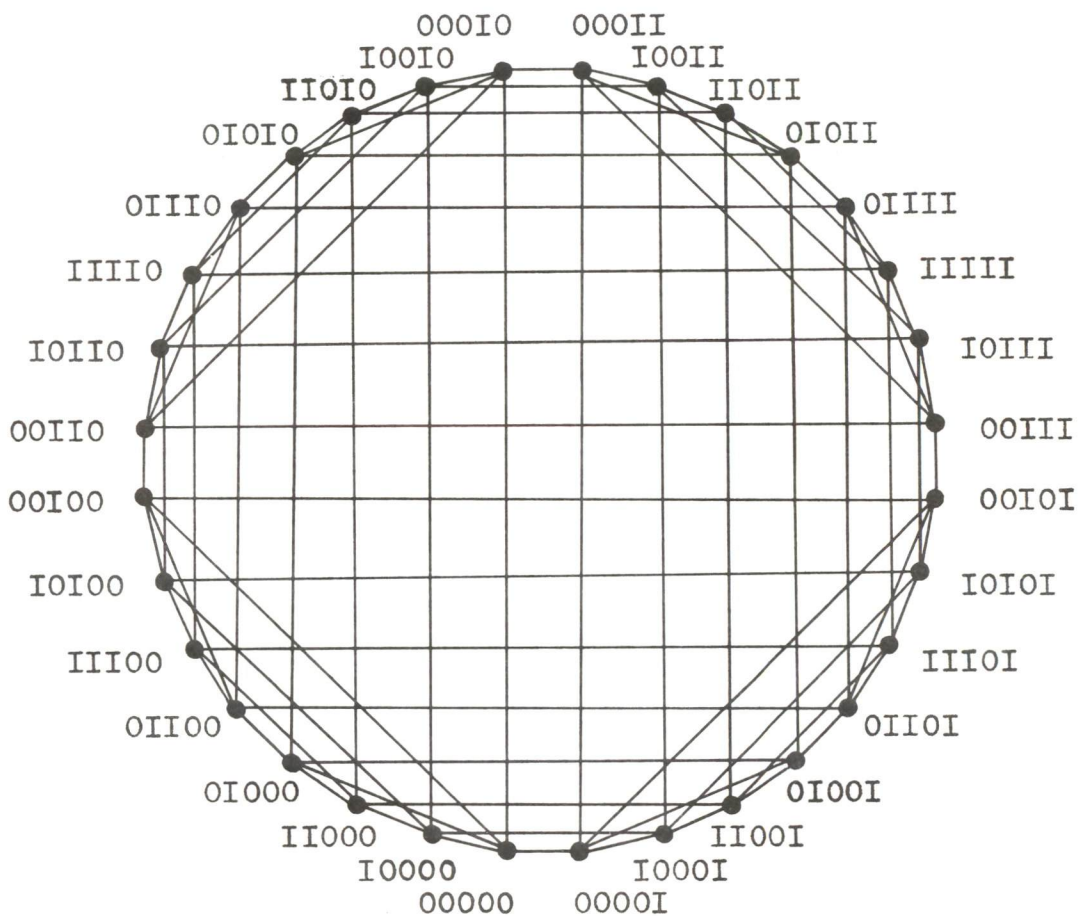


3 VARIATIONS OF A 5 VARIABLE KARNAUGH MAP

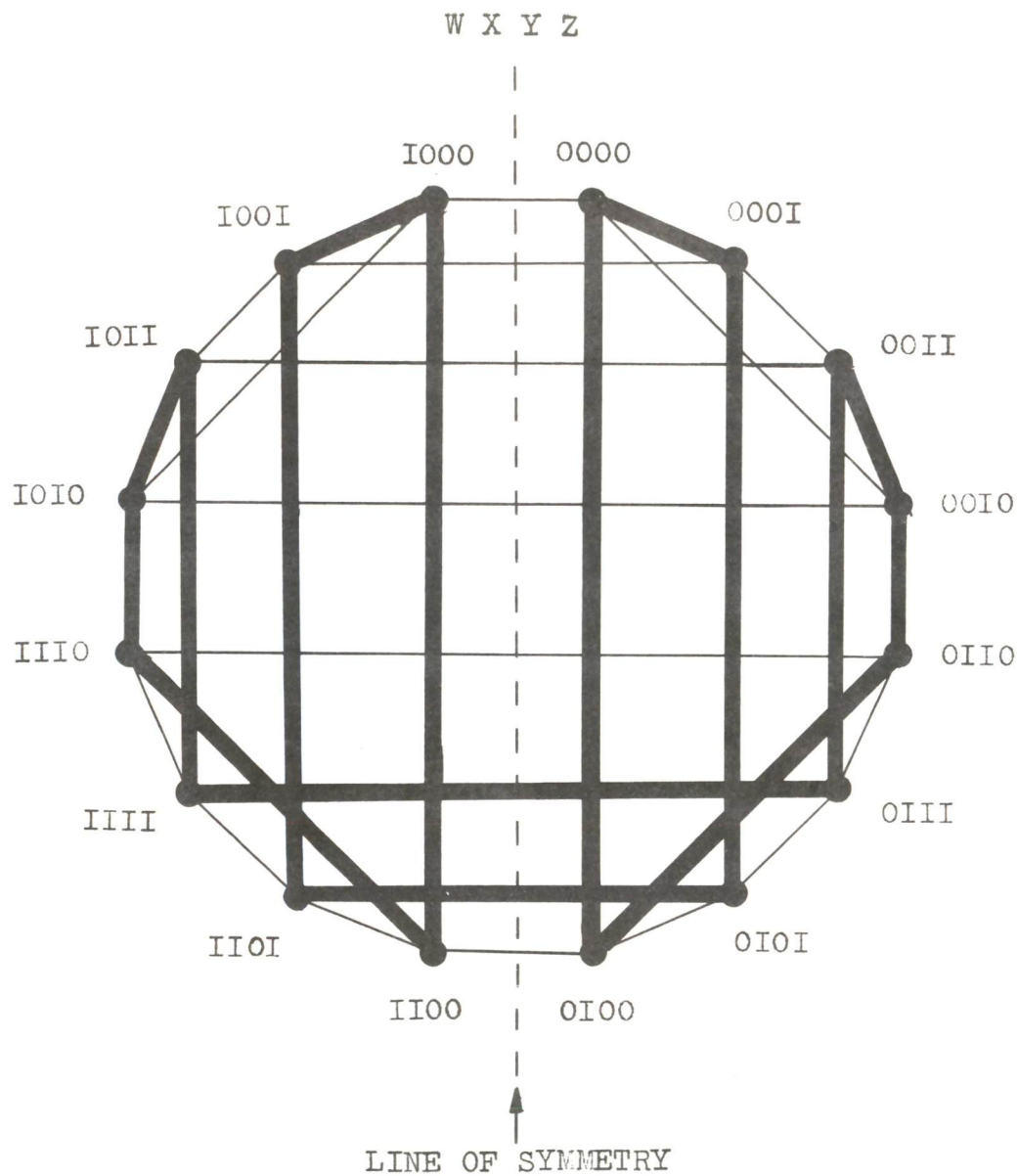
Fig. III.2

code is a pseudo-Gray code becomes important when one Gray code. The ability to recognize that a monostrophic combination of complements and permutations of an n bit code is trivial since, by definition it is only a particular PSEUDO-GRAY CODES: The synthesis of a pseudo-Gray

Fig. III.3



of a 5 bit path diagram. Any path through the path diagram which does not retrace itself or does not go through the same point twice is a distinct monostrophic code. A path closure, naturally, indicates a monostrophic closure. Fig. III.3 is an example



PSEUDO- GRAY	Z	I I 0 0 0 0 I I I I 0 0 0 0 I I
	Y	0 0 0 0 I I I I I I I I 0 0 0 0
	X	I 0 0 I I 0 0 I I 0 0 I I 0 0 I
	W	0 0 0 0 0 0 0 0 I I I I I I I I
NORMALIZED GRAY	X	0 I I 0 0 I I 0 0 I I 0 0 I I 0
	Z	0 0 I I I I 0 0 0 0 I I I I 0 0
	Y	0 0 0 0 I I I I I I I I 0 0 0 0
	W	0 0 0 0 0 0 0 0 I I I I I I I I

Fig. III.4

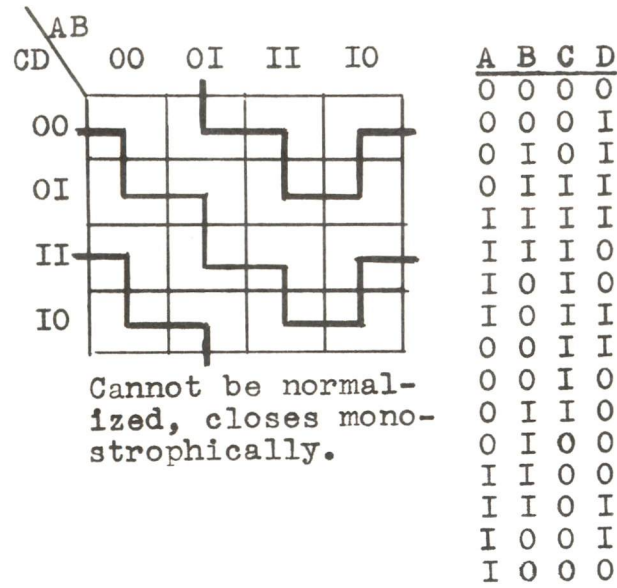
considers counting and conversions which are to be discussed in following chapters.

Nevertheless, a look at the Karnaugh Maps in Appendix I will show that, for a 4 bit code which is Gray or pseudo-Gray there exists a symmetry which may be described as the presence of mirror images (ignoring direction arrows) in all maps about either the vertical or horizontal center lines of the maps. A similar symmetry will be present in a Gray code of any number of bits.

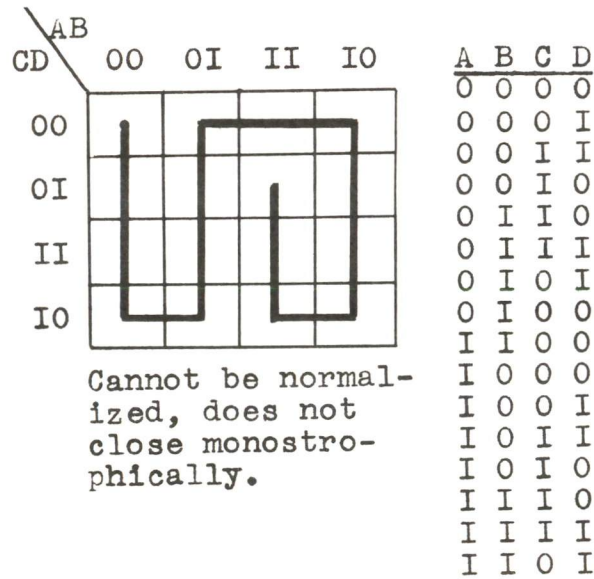
On a path diagram a Gray or pseudo-Gray code will result in a symmetrical pattern about a line dividing the circle in half. Fig.III.4 demonstrates this for a 4 bit pseudo-Gray code.

OTHER FULL COUNT MONOSTROPHIC CODES: There are many paths through a Karnaugh Map or path diagram which use all 2^n combinations available in an n bit code that cannot be normalized to a Gray code or do not close monostrophically. Fig.III.5 shows examples of such 4 bit codes. These codes have no general direct application, but are mentioned only to insure knowledge of their presence. A possible employment of such a code is in cryptographic systems. Also, they may be encountered when one considers the sequence of only certain bits in a code as we will do further along.

FORESHORTENED COUNT MONOSTROPHIC CODES: The most common use of n bit foreshortened counting sequences, both monostrophic and polystrophic, is to represent numbering



a.



b.

Fig. III.5

systems of radices of less than 2^n . For example, it takes ten of the sixteen possible combinations of a 4 bit code to represent the ten digits of a decimal numbering system.

As previously mentioned, one may take advantage of the unused combinations to simplify decoding and conversion, or they may be used as error detection and/or correction. Other uses of the unused combinations are possible such as reduction of power supply current and/or regulation requirements. Discussion in subsequent chapters will make more clear advantageous use of unused combinations in foreshortened counts.

Synthesis of foreshortened count monostrophic codes is easily accomplished by the use of Karnaugh Maps and path diagrams just as in all full count monostrophic codes. The observations made for full count codes hold true for foreshortened counts with the following exceptions:

- a) All 2^n combinations are not used.
- b) Monostrophic closure is not possible if the number of counts is odd.

REFLECTED MONOSTROPHIC CODES: In general, a reflected full or foreshortened monostrophic code must be synthesized as previously discussed, observing the symmetry rule on Karnaugh Maps or path diagrams if used.

If the bit upon which reflectivity depends (usually the most significant bit) will not change except between

the lower and upper halves of the counting sequence and upon closure (0 for the first half and 1 for the last half of counting sequence, or vice versa), synthesis of the reflected monostrophic code can be simplified. Only the bits exclusive of the one upon which reflectivity depends must be considered. The counting sequence of these bits must have one-half the counts of the total counting sequence, be monostrophic, but does not have to close monostrophically. In effect, the counting sequence of these bits counts up during the first half of the sequence and down during the last half of the sequence, retracing its path to "zero".

CHAPTER IV. MONOSTROPHIC-POLYSTROPHIC CODE CONVERSION

Monostrophic codes are unweighted codes and therefore, are difficult to manipulate arithmetically. Consequently, conversions between monostrophic and polystrophic codes, the subject of this chapter, are often required.

The use of hardware-oriented examples interspersed with academic discussion will be the general approach of this chapter. The examples will show electronic logic and switching logic, both employing commonly accepted symbols which are explained in Appendix II.

Ring sum or Exclusive Or functions appear repeatedly in this chapter. Appendix III contains a discussion of this function.

PARALLEL CONVERSION OF GRAY TO BINOMIAL BINARY: The relationship between these two codes may best be described as follows: The most significant bits of both codes are equal, and the lesser significant bits of the binomial binary code are equal to the ring sum (Exclusive Or) of the corresponding Gray code bit and all the more significant Gray code bits. Fig. IV.1 diagrammatically shows this.

An unvigorous method of proving the relationship shown in Fig. IV.1 is to solve the relationship for a 4 bit Gray to a 4 bit binomial binary code conversion. This is done in Fig. IV.2. Once this is accomplished it can be

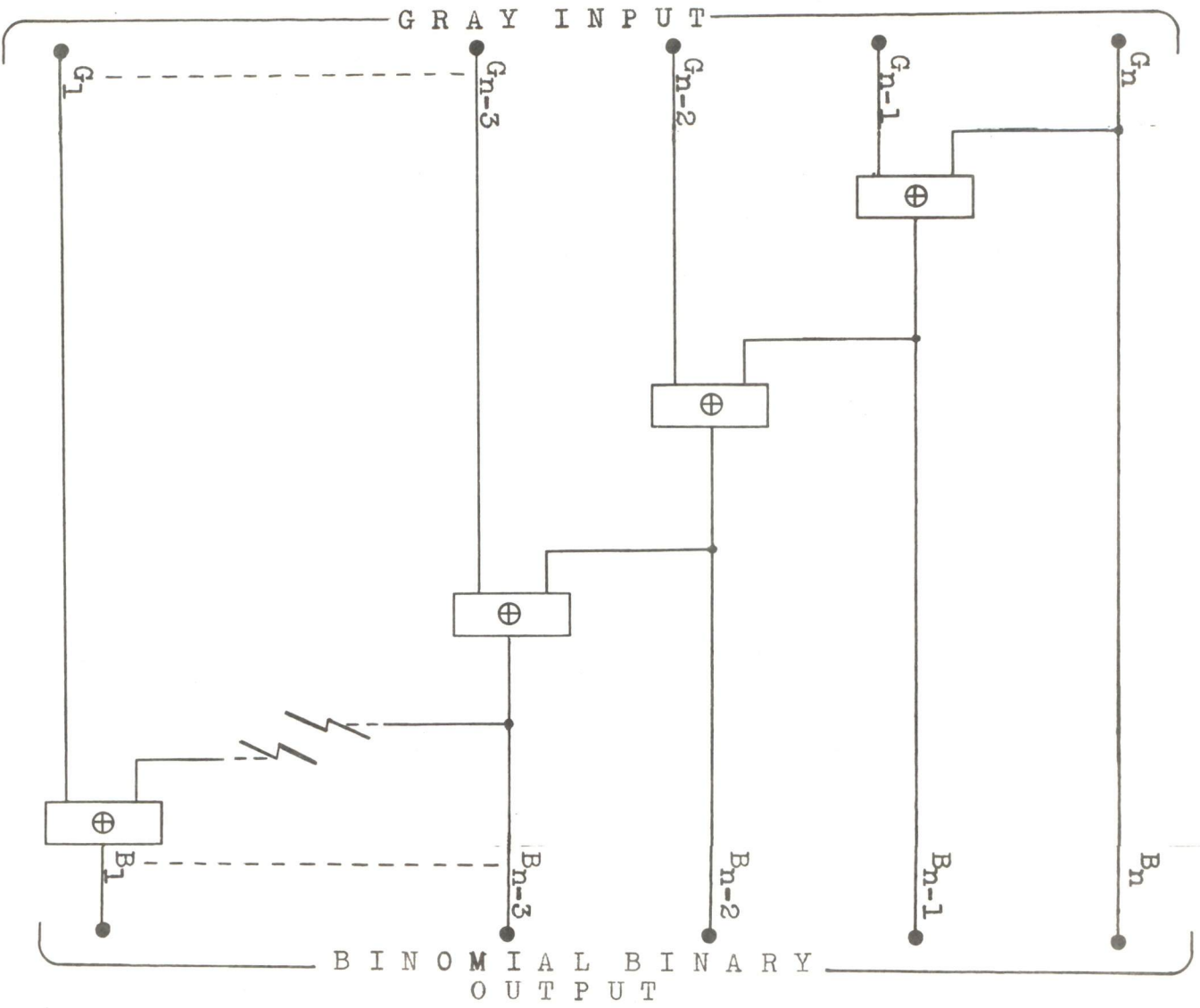


FIG. IV.1

GRAY				BINOMIAL BINARY			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

W=A, by inspection

		AB			
CD		00	01	11	10
00			I		I
01			I		I
11			I		I
10			I		I

$$X = \bar{A}\bar{B} + \bar{A}B$$

$$= A \oplus B$$

		AB			
CD		00	01	11	10
00			I		I
01			I		I
11		I		I	
10		I		I	

$$Y = (\bar{A}\bar{B} + AB)C + (\bar{A}\bar{B} + \bar{A}B)\bar{C}$$

$$= A \oplus B \oplus C$$

		AB			
CD		00	01	11	10
00			I		I
01		I		I	
11			I		I
10		I		I	

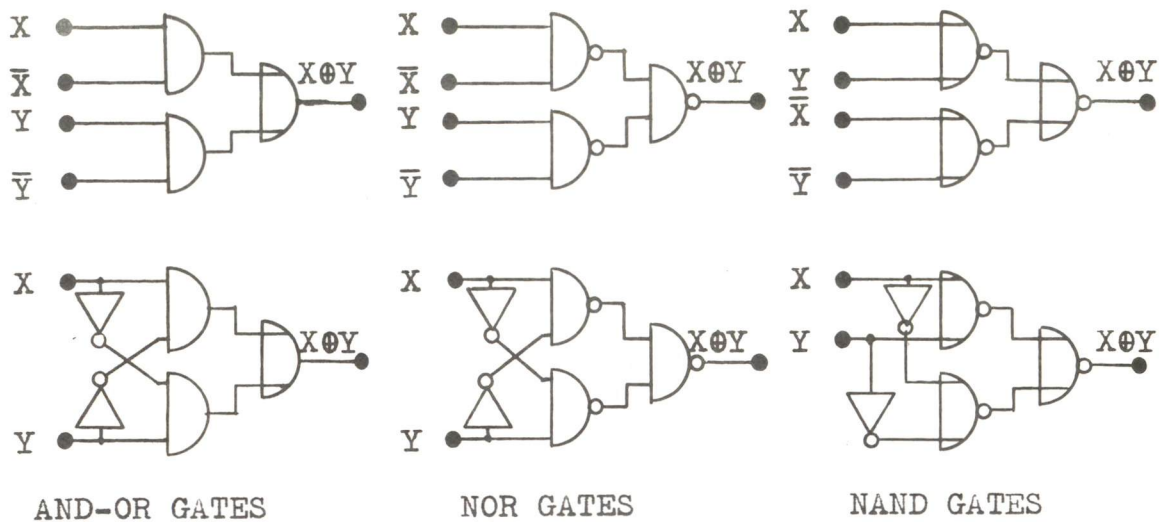
$$Z = S_{1,3}(A, B, C, D)$$

$$= A \oplus B \oplus C \oplus D$$

Fig. IV.2

intuitively seen that the relationship holds for n bits.

To implement Fig. IV.1 with electronic logic each ring sum symbol can be replaced by an Exclusive-Or module, or a combination of And-Or, Nand or Nor gates as shown in Fig. IV.3.

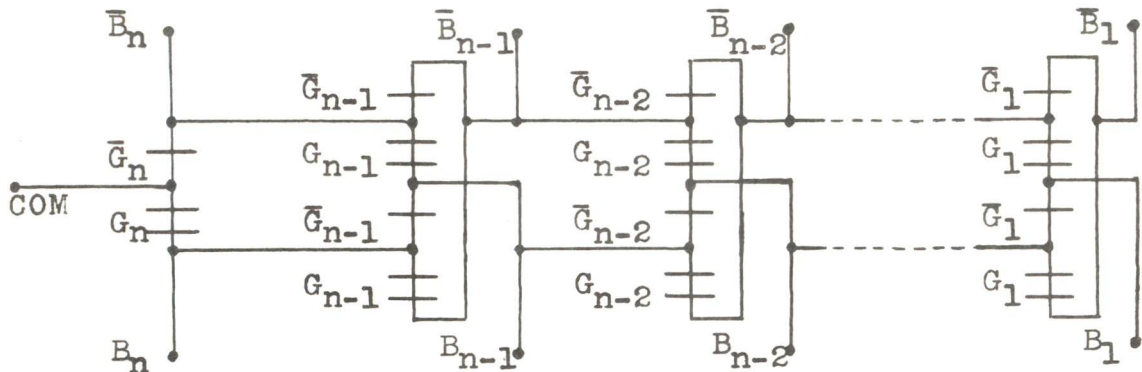


EXCLUSIVE OR LOGIC

Fig. IV.3

Switching logic can be used to convert Gray to binomial binary with one transfer set associated with the most significant Gray bit and two transfer sets for each lesser significant Gray bit. Fig. IV.4 demonstrates this conversion.

Remembering that the ring sum of a number of variables is the symmetric function S_{odd} of the variables, and the complement of the ring sum of a number of variables is the symmetric function S_{even} of the variables, it would seem that a symmetric switching circuit with appropriate pick-off points might furnish all the logic required for a Gray to binomial binary conversion. Fig. IV.5 shows a folded symmetric switching circuit which functions as a Gray to binomial binary converter, and is the same circuit shown in Fig. IV.4, laid out differently to clearly demonstrate the symmetric switching circuit relationship.



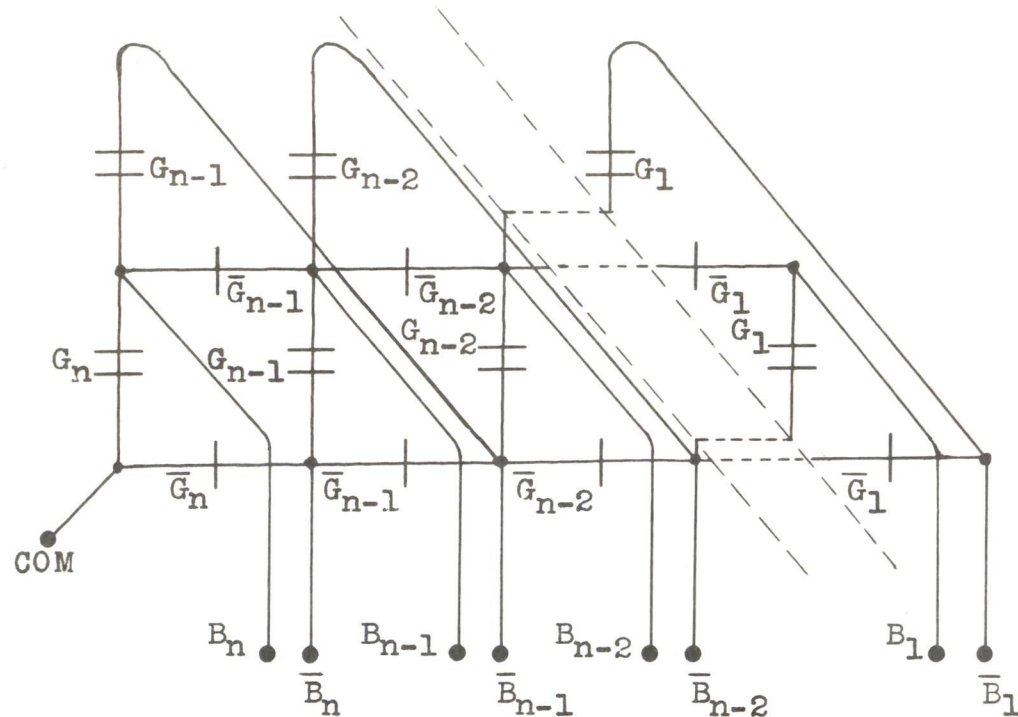
GRAY TO BINOMIAL BINARY

Fig. IV.4

PARALLEL CONVERSION OF PSEUDO-GRAY TO BINARY: By use of the methods of conversion described for Gray to binomial binary codes and by effecting normalization of the input variables, pseudo Gray codes may easily be converted to binomial binary.

PARALLEL CONVERSION OF OTHER FULL COUNT MONOSTROPHIC CODES: Generally, requirements for parallel conversion of other than Gray or pseudo-Gray codes to weighted codes are not encountered. If the requirement does arise, "brute force" techniques can be employed to accomplish such conversion, but the complexity of such conversions will be much more than for the Gray and pseudo Gray to binomial binary.

PARALLEL CONVERSION OF FORESHORTENED MONOSTROPHIC CODES: There are many reasons for the use of foreshortened codes, the most obvious of which is the encoding of a numbering system whose radix is not an integer power of two. Other



GRAY TO BINOMIAL BINARY
FOLDED SYMMETRIC CIRCUIT

Fig. IV.5

reasons are ability for error detection and/or correction, ease of conversion to a weighted code, etc..

In selecting a foreshortened monostrophic code more than the minimum number of bits may be used to encode a numbering system in order to satisfy all the requirements imposed on the coding system. In general, the higher the number of bits used to encode a numbering system (above the minimum required), the easier it is to convert (or decode) and more difficult to transmit.

The monostrophic code used to represent a numbering system which must be converted to another binary code is

usually selected so that conversion is simplified. For example, a monostrophic coding of a decimal numbering system that must be converted to the self-complementing excess-3 code (XS3) would be simplest to convert if counts 3 through 12 of the Gray code were used to represent 0 through 9 of the excess-3 decimal code. Such a selection would result in a code being reflective, that closes monostrophically, and following the Gray to binomial binary conversion rules. Fig. IV.6 shows this. The same philosophy applied to the NBC Decimal system would result

DECIMAL	MONO- STROPHIC	XS3
0	0010	0011
1	0110	0100
2	0111	0101
3	0101	0110
4	0100	0111
5	1100	1000
6	1101	1001
7	1111	1010
8	1110	1011
9	1010	1100

Conversion to XS3 follows same rules as Gray to binomial binary.

Fig. IV.6

in counts 0-9 of the Gray code to represent 0-9 of the NBC Decimal system simplifying conversion, but the monostrophic code lacks reflectivity and monostrophic closure, thereby being a poor choice.

Fig. IV.7 shows a monostrophic coding of a 2421 binary coded decimal numbering system. Again, the monostrophic code is reflective and closes monostrophically, and follows the simple Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	BINARY 2421	
0	0000	0000	
1	0001	0001	Conversion to 2421
2	0011	0010	BC Decimal follows
3	0010	0011	same rules as Gray
4	0110	0100	to binomial binary.
5	1110	1011	
6	1010	1100	
7	1011	1101	
8	1001	1110	
9	1000	1111	

Fig. IV.7

Occasionally the duo-decimal (Radix¹²) numbering system rears its ugly head in such applications of distance measuring systems and monetary systems. A self-complementing 4 bit code that could be used to represent such a code is the excess-2 (XS2) code. Fig. IV.8 shows the XS2 code and a monostrophic equivalent which is reflective, closes monostrophically and follows the Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	XS2	
0	0011	0010	
1	0010	0011	
2	0110	0100	
3	0111	0101	Conversion to XS2
4	0101	0110	follows same rules
5	0100	0111	as Gray to binomial
6	1100	1000	binary.
7	1101	1001	
8	1111	1010	
9	1110	1011	
10	1010	1100	
11	1011	1101	

Fig. IV.8

A weighted 4 bit code for a binary coded duo-decimal numbering system is the 4421 code. Fig. IV.9 shows this

code with a monostrophic equivalent which is reflective, closes monostrophically, and follows the Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	4421	
0	0000	0000	
1	0001	0001	
2	0011	0010	
3	0010	0011	
4	0110	0100	Conversion to 4421
5	0111	0101	binary coded duo-
6	1111	1010	decimal follows same
7	1110	1011	rules as Gray to
8	1010	1000	binomial binary.
9	1011	1001	
10	1001	1110	
11	1000	1111	

Fig. IV.9

DATEX CODE:* A foreshortened monostrophic code that is frequently encountered is the Datex code shown in Fig. IV.10. It is a 10 count, 4 bit monostrophic code which is reflective and closes monostrophically.

DECIMAL	DATEX
0	0001
1	0011
2	0010
3	0110
4	0100
5	1100
6	1110
7	1010
8	1011
9	1001

Fig. IV.10

Notice that the 0000 and 1111 combinations are not used enabling partial error detection and reducing power supply regulation and power requirements.

* Datex Corp., 1307 S. Myrtle Ave., Monrovia, Calif.

If more than one decade is employed in the counting sequence, the decimal counting sequence is also monostrophic in nature, i.e., only one decimal digit changes at a time, resulting in a decimal counting sequence as follows: 0 through 9, 19 through 10, 20 through 29, 39 through 30, etc. Therefore when converting to any decimal code, starting with the next to the most significant digit and working to the least significant digit the nine's complement must be sensed if the next more significant digit is odd. The most significant digit is always assumed to have a zero (which is even) preceding it so it never needs to be nine's complemented.

In order to convert to a decimal code we can, in effect, say that the Datex code is a 5 bit code as shown in Fig. IV.11, the E_{in} bit indicating whether the next more significant digit is odd or even (1 = odd, 0 = even).

DATEX ABCD	DECIMAL	
	$E_{in}=0$	$E_{in}=1$
0001	0	9
0011	1	8
0010	2	7
0110	3	6
0100	4	5
1100	5	4
1110	6	3
1010	7	2
1011	8	1
1001	9	0

Fig.IV.11

The easiest method of conversion (or decoding) to another binary coded decimal system is to use conversion

(or decoding) logic for Fig. IV.10, and use the E_{in} bit to reverse the sense of the A bit by the conversion (or decoding) logic as shown in Fig. IV.12. This technique can be used for conversion (or decoding) any multi-decade reflective coding system whose coded numbering system is similar in nature to the Datex code. The

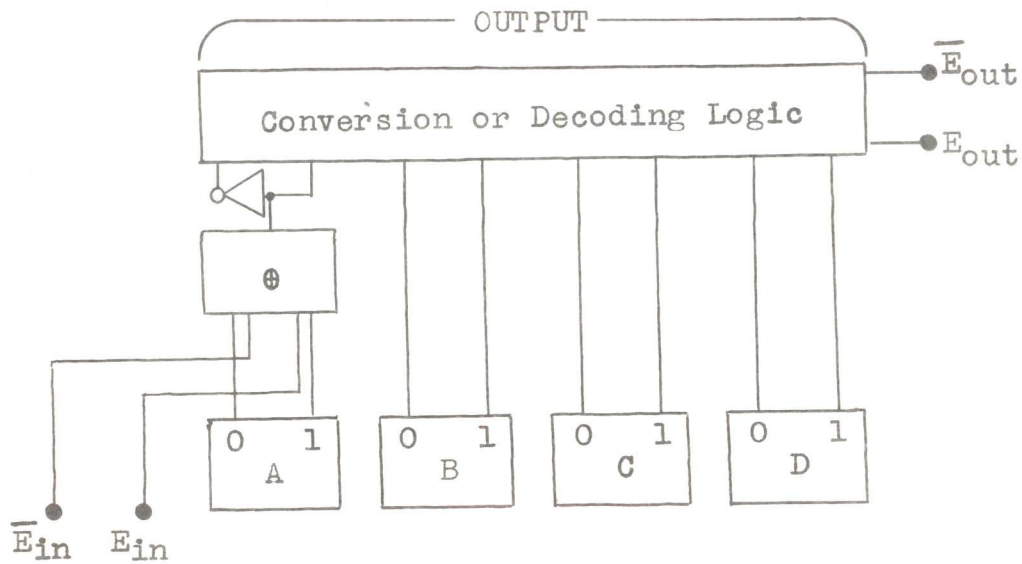


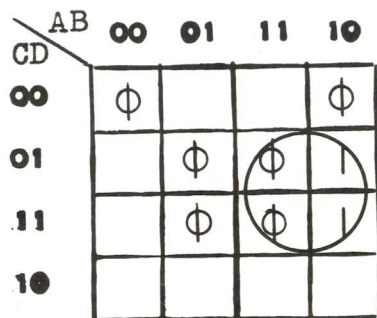
Fig. IV.12

logic required for the conversion technique in Fig. IV.12 is shown in Fig. IV.13 for the 8421, 2421 and XS3 binary coded decimal systems.

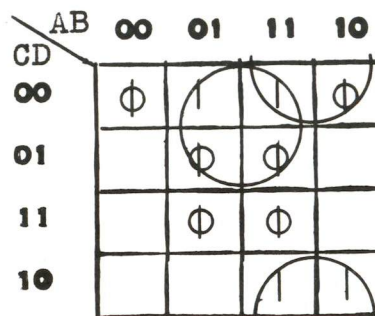
SWITCH-TAIL RING COUNTER CODE: Another foreshortened monostrophic code often employed is a code easily generated in a switch-tail ring counter (shift register) described in the next chapter. The number of counts in such an n bit code is 2^n counts, and decoding each of the

DECIMAL	DATEX				8 4 2 1				2 4 2 1				X S 3			
	A	B	C	D	H	I	J	K	L	M	N	O	P	Q	R	S
0	0	0	0	I	0	0	0	0	0	0	0	0	0	0	I	I
1	0	0	I	I	0	0	0	I	0	0	0	I	0	I	0	0
2	0	0	I	0	0	0	I	0	0	0	I	0	0	I	0	I
3	0	I	I	0	0	0	I	I	0	0	I	I	0	I	I	0
4	0	I	0	0	0	I	0	0	0	I	0	0	0	I	I	I
5	I	I	0	0	0	I	0	I	I	0	I	I	I	0	0	0
6	I	I	I	0	0	I	I	0	I	I	0	0	I	0	0	I
7	I	0	I	0	0	I	I	I	I	I	0	I	I	0	I	0
8	I	0	I	I	I	0	0	0	I	I	I	0	I	0	I	I
9	I	0	0	I	I	0	0	I	I	I	I	I	I	0	0	0

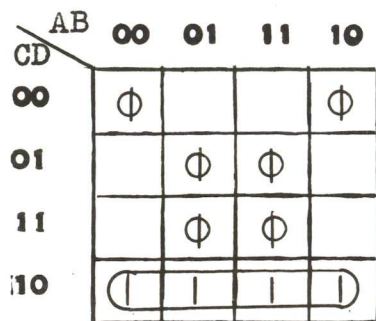
8 4 2 1



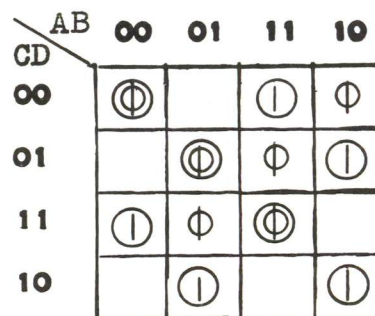
$H=AD$



$I=A\bar{D}+B\bar{C}$



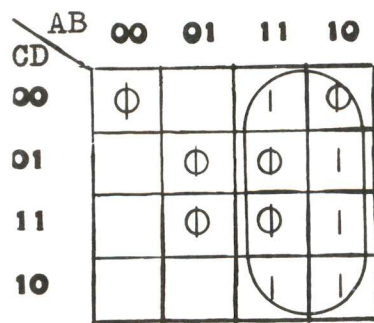
$J=C\bar{D}$



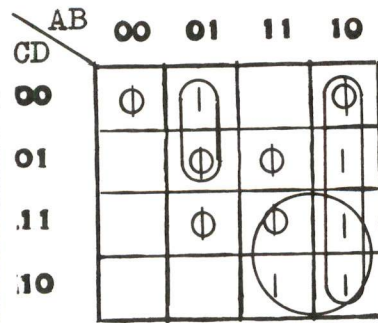
$K=E_{out}=(A\oplus B\oplus C\oplus D)$

Fig. IV.13
(Cont'd next page)

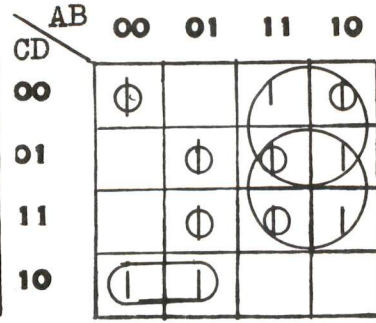
2 4 2 1



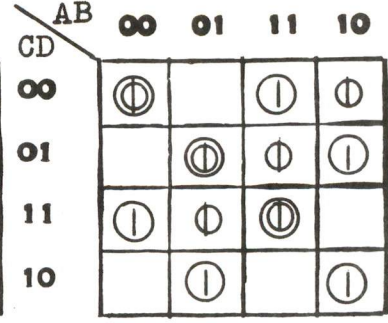
L=A



M=A(B̄+C)+ĀBC̄

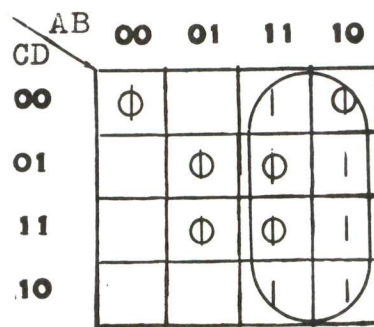


N=A(C̄+D)+ĀC̄D̄

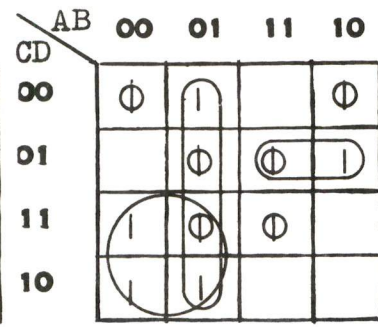


O=E_{out}=A⊕B⊕C⊕D

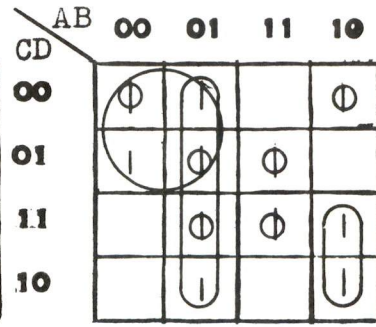
X S 3



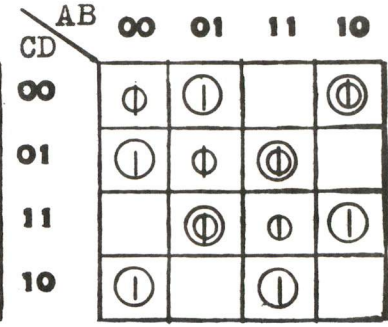
P=A



Q=Ā(B+C)



R=Ā(B+C̄)



S=E_{out}=A⊕B⊕C⊕D

Fig. IV.13
 (Cont'd from previous page)

2^n combinations to one of 2^n lines requires sensing of only two of the n bits. Fig. IV.14 demonstrates the switch-tail code for a 5 bit unweighted decimal code.

SWITCH-TAIL					DECIMAL	REQ'D LOGIC
A	B	C	D	E		
0	0	0	0	0	0	$\overline{A}\overline{E}$
0	0	0	0	I	1	$\overline{D}\overline{E}$
0	0	0	I	I	2	$\overline{C}\overline{D}$
0	0	I	I	I	3	$\overline{B}\overline{C}$
0	I	I	I	I	4	$\overline{A}\overline{B}$
I	I	I	I	I	5	$\overline{A}\overline{E}$
I	I	I	I	0	6	$\overline{D}\overline{E}$
I	I	I	0	0	7	$\overline{C}\overline{D}$
I	I	0	0	0	8	$\overline{B}\overline{C}$
I	0	0	0	0	9	$\overline{A}\overline{B}$

Fig. IV.14

Notice that the bits sensed for each count are the two adjacent bits that differ except for all 0's and all I's in which case the two end bits are sensed. This holds true for any number of bits.

IN-REGISTER PARALLEL CONVERSION: Quite often it is desired to convert in parallel the contents of a register containing a monostrophic code to a polystrophic code, with the result of the conversion placed in the same register. This can be done by loading the register, through the necessary conversion logic, into the same register.

The logic required can be the same conversion logic as described earlier in this chapter, but in most cases it may be simplified. This is true because the state of

W	X	Y	Z	A ₁	B ₁	C ₁	D ₁
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

W, X, Y & Z are the contents of flipflops A, B, C & D prior to conversion.

A₁, B₁, C₁ & D₁ are contents of flipflops A, B, C & D respectively after conversion.

Logic terms required to complement "incorrect" bits:

A₁=W FFA requires no correction.

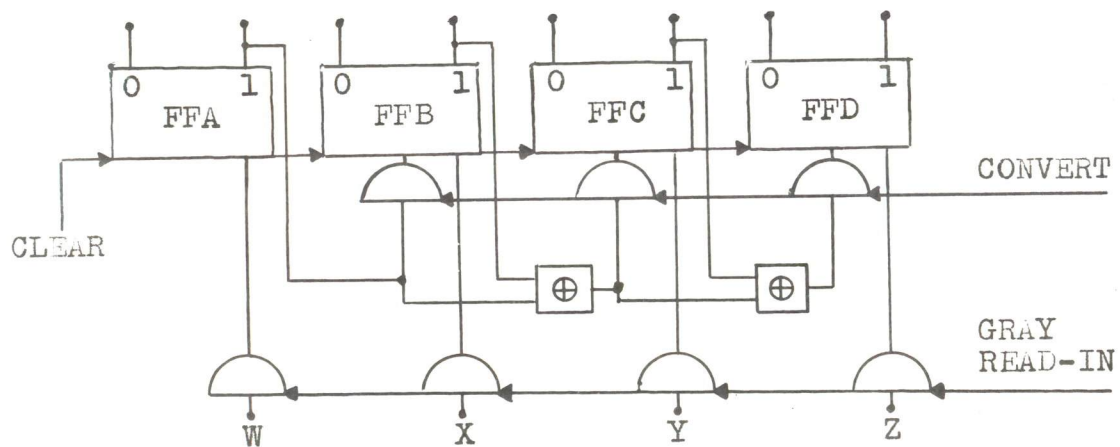
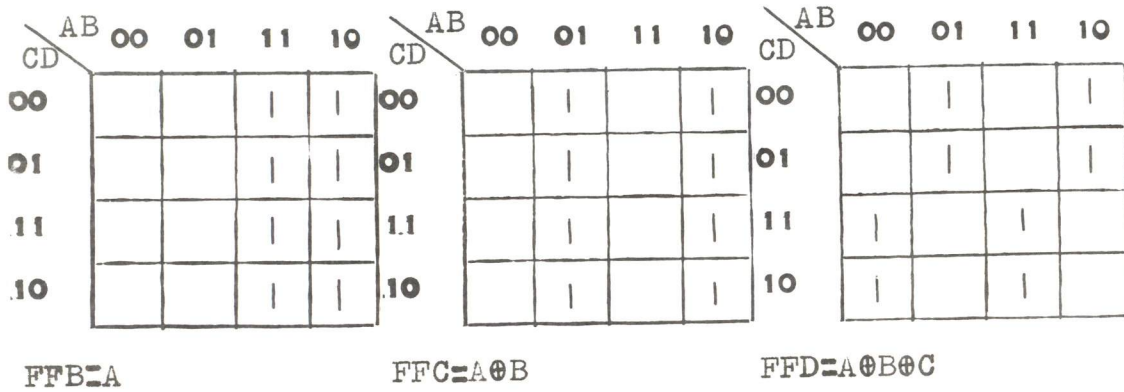


Fig. IV.15

each bit only needs correction, not generation. Fig. IV.15 demonstrates the correction approach to a 4 bit Gray to binomial binary in-register parallel conversion by complementing the "incorrect" bits. To expand to n bits, additional similar stages can be added on the least significant end of the 4 bit register shown in Fig. IV.15.

If the flipflops of the register have no complement inputs (or if pulsing both inputs simultaneously does not complement the flipflops) they must be appropriately set or reset to correct the "incorrect" bits. Fig. IV.16 demonstrates this approach for the same code conversion as shown in Fig. IV.15. Again FFA requires no correction as $A_1=W$.

SERIAL CONVERSION OF GRAY TO BINOMIAL BINARY: If a Gray coded number is being transmitted serially (bit by bit) with the most significant bit leading, serial conversion to binomial binary is possible upon receipt of each bit. The conversion is accomplished using the two rules:

- 1) The most significant Gray bit is identical to the most significant binomial binary bit.

- 2) If a bit is a 1 after conversion the bit following it is complemented.

Fig. IV.17 demonstrates the conversion.

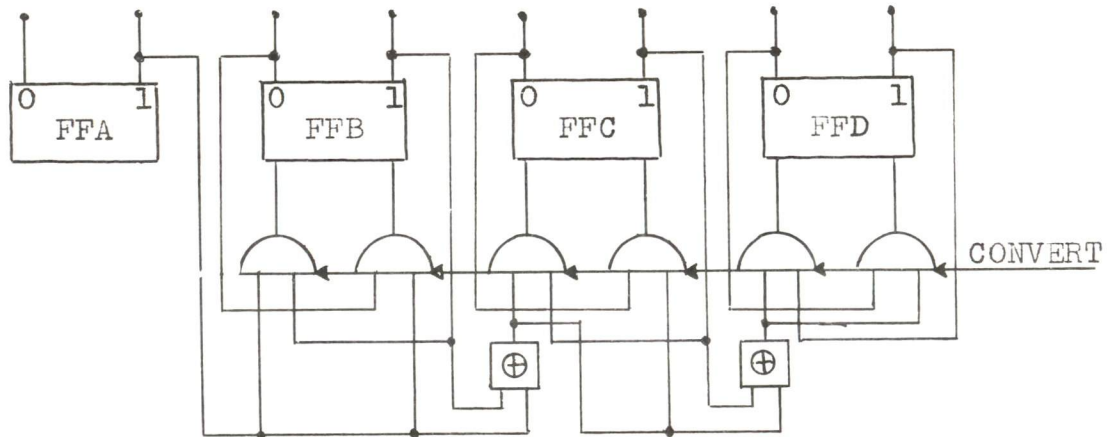
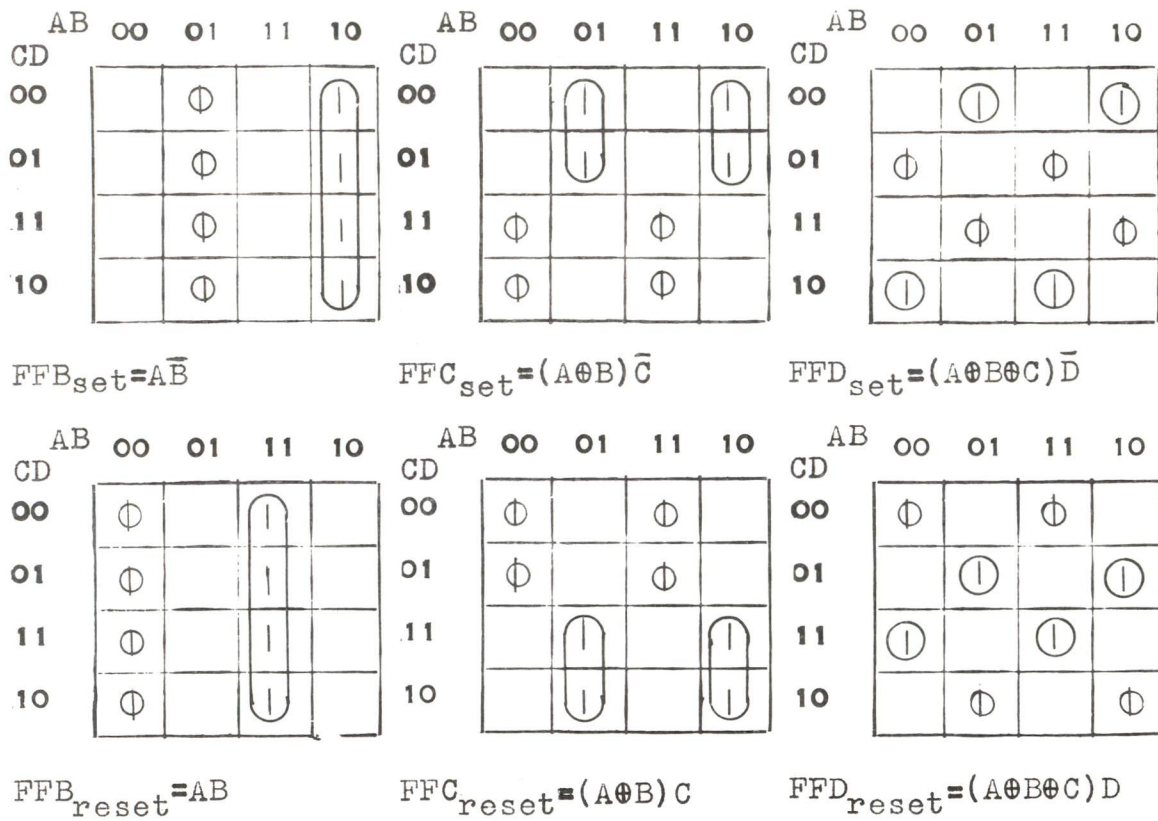
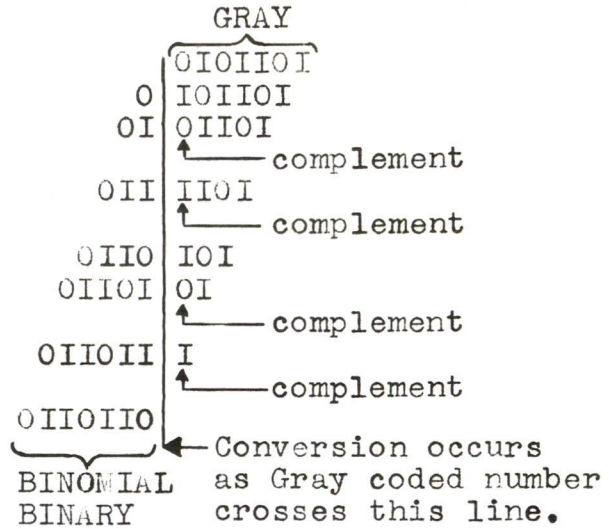


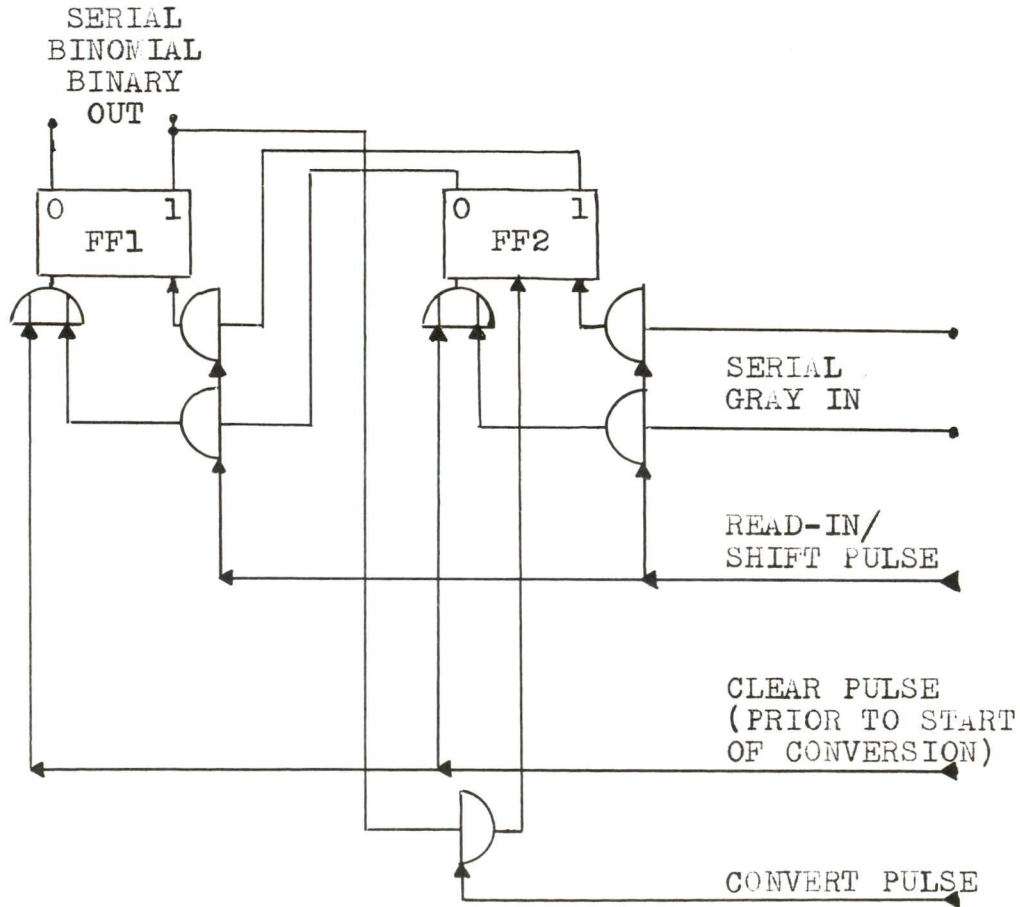
Fig. IV.16

Direction of shift is to the left with the most significant bit leading. When bit just to left of vertical line is 1 the bit just to the right of the vertical line is complemented.



SERIAL GRAY TO BINOMIAL BINARY CONVERSION

Fig. IV.17



SERIAL GRAY TO BINOMIAL BINARY CONVERTER

Fig. IV.18

the input but before the convert pulse. (The convert pulse can be delayed until slightly before the next input bit to allow more sampling time if output is taken from the output of the Exclusive Or gate.)

SERIAL CONVERSION OF OTHER MONOSTROPHIC CODES:

Serial, bit by bit, conversion of codes other than those that follow the rules for Gray to binomial binary conversion is usually not possible in the true sense of the word, serial. If such codes are received for conversion in serial form, they are usually shifted into a register and converted in parallel when the whole binary coded number is in the register using parallel conversion techniques.

POLYSTROPHIC TO MONOSTROPHIC CODE CONVERSION: In order for the discussion of conversion to be considered nearly complete, conversion from polystrophic to monostrophic codes must be considered. If the reverse process of each conversion previously mentioned was discussed in as much detail, considerable redundancy of thought would be involved. Therefore, the discussion of this topic will be kept brief.

Any conversion which, when going from a monostrophic to a polystrophic code, follows the Gray to binomial binary conversion rules, can be accomplished in the reverse direction by the following rule: The monostrophic

code equivalent of a binary coded number is the bit-by-bit Exclusive Or of the number with itself shifted one bit (to the right or left) ignoring the least significant bit of the result (See Fig. IV.20). It is obvious

```

1001101 ←# to be converted to monostrophic equiv.
 1001101 ←# shifted one bit
1101011 ← Bit by bit Exclusive Or
           ← LSB, ignore
1101011 ← Result, monostrophic equiv.

```

Fig. IV.20

that the most significant bit is the same in either code, and the lesser significant bits of the monostrophic equivalent are the Exclusive Ors of adjacent bits of the binary coded number. Parallel conversion, therefore, can be accomplished as shown in Fig. IV.21.

Serial conversion, following the same rules requires the number to arrive for conversion in serial form with the most significant bit leading. The previous bit (unconverted) must be stored and Exclusive Or'd with the arriving bit (a zero is assumed to be stored in the one bit memory prior to arrival of the most significant bit). The output of the Exclusive Or is the serial converted output. Fig. IV.22 shows a serial Polystrophic-Monostrophic converter with a flip-flop to store the output (could be entrance bit to a shift register). The storage flip-flop must have a clear input shown, and it must be cleared (set to 0) prior to start of conversion.

Conversion between codes that do not have the

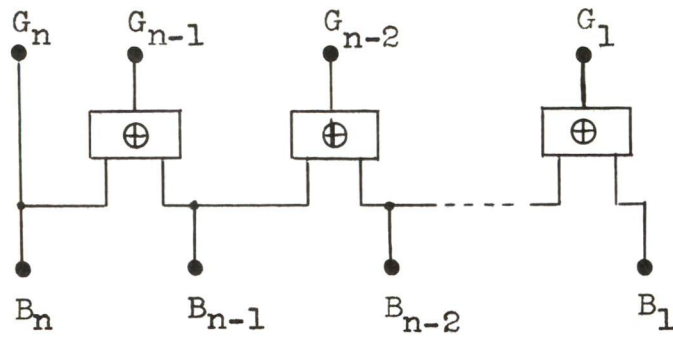


Fig. IV.21

Gray-binomial binary relationship is usually not required but, if encountered, must be accomplished by brute force parallel conversion techniques.

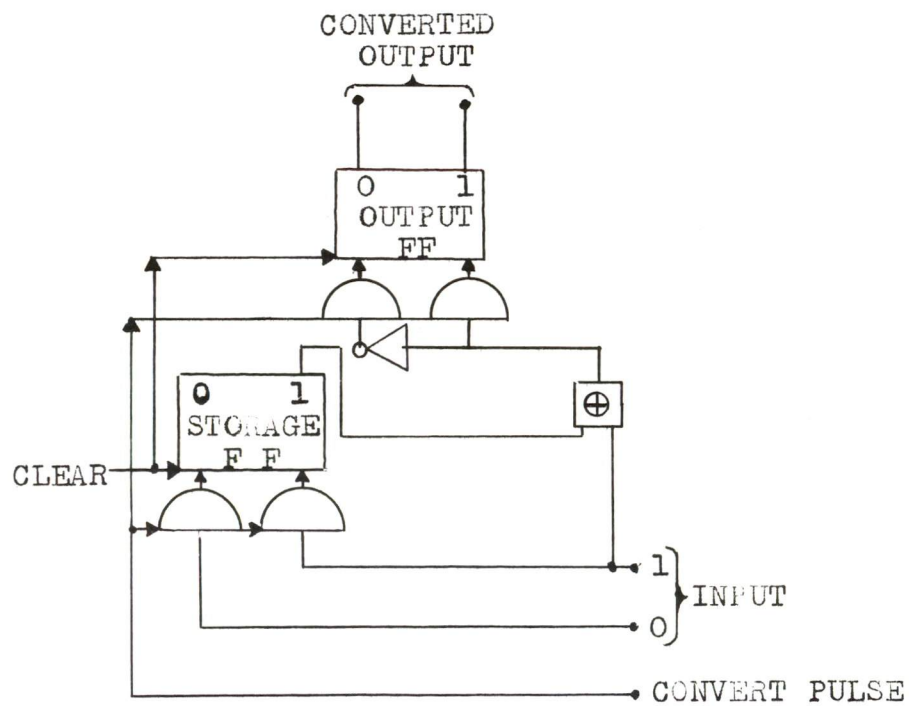


Fig. IV.22

CHAPTER V. GENERATION OF MONOSTROPHIC CODES

The most developed method of monostrophic code generation is the use of electro-mechanical and electro-optical-mechanical sensing of segmented, multiple track, rotatable discs in shaft angle encoding systems. The advantage of monostrophic codes in such applications is the ability to sample the readout "on the fly" without fear of ambiguity. Since this method is well known and already well documented this author will concentrate on generation of monostrophic codes by other than the disc technique, i.e., monostrophic code counters by electronic logic and sequential switching techniques.

PHILOSOPHY OF PULSE COUNTING DIRECTLY IN A MONOSTROPHIC CODE: Because only one bit changes between adjacent counts in any monostrophic code, one cannot take advantage of carries as in, for instance, an electronic binomial binary counter when counting in a monostrophic code. Nor can the state of the next less significant bit be depended upon to furnish the proper levels to the complement input gate of a bit as in an electronic binomial binary counter. Therefore, the state of the complete binary number must be employed to furnish, through logic, gating levels to the complement input pulse gate (or set and reset pulse gates) of the storage element of each bit (bistable flip-flops). This approach is analogous to the

anticipated carry approach used with other binary counters to enable faster counting. Also, it is the approach that must usually be taken for any type of sequential switching (relay) counters. Fig. V.1 demonstrates this approach for a 3 bit Gray code counter.

N BIT GRAY CODE COUNTER: The previous section described a 3 bit Gray code counter. In this section an n bit Gray code counter and required counting logic will be demonstrated. A method for easily reversing the counting sequence will also be shown. The importance of the Exclusive Or function will again appear.

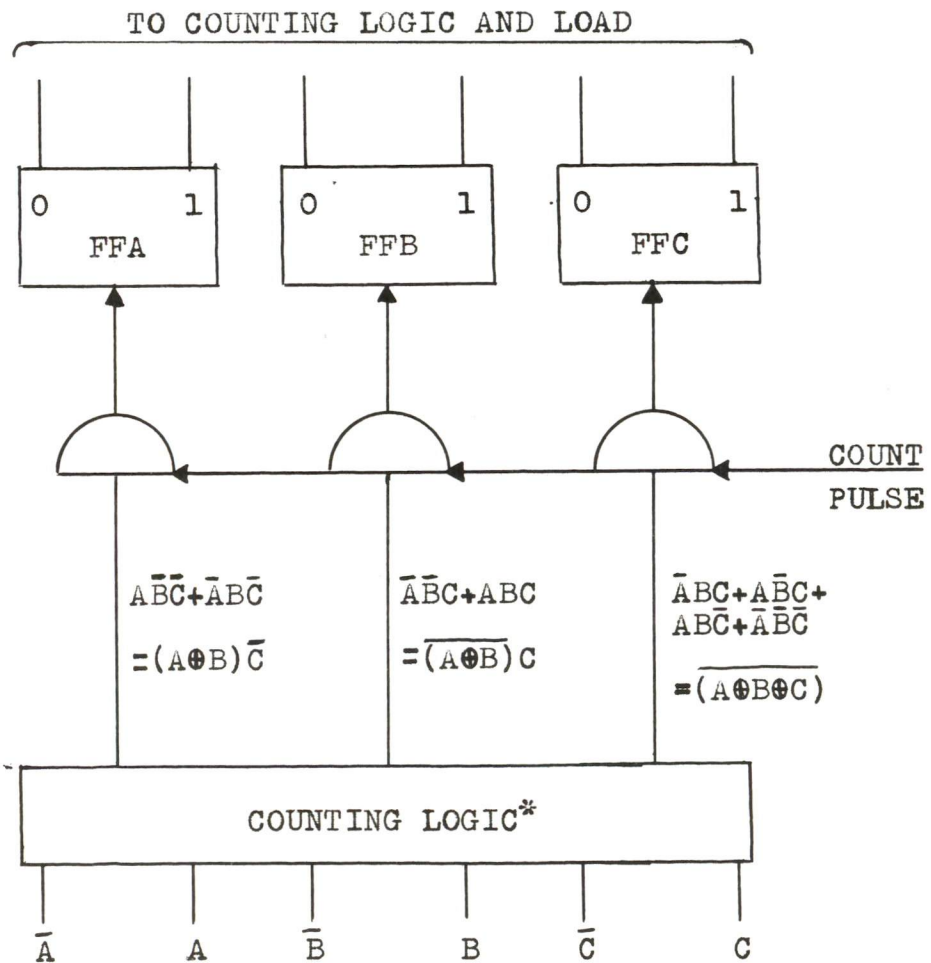
If the method for determining the counting logic described in the previous section is used to determine the counting logic for a 2, 3, 4 and 5 bit Gray code counter, the information shown in Fig. V.2 is obtained after simplification. Notice that the state of every bit influences the logic levels associated with the complement pulse gate of each bit as previously stated at the beginning of this chapter.

Another set of relationships evident in Fig. V.2 and explained below can be used to simplify the determination of the logical expressions required for each bit of an n bit Gray code counter.

1. The least significant bit's term for an up counter is the complement of the ring sum of all bits.

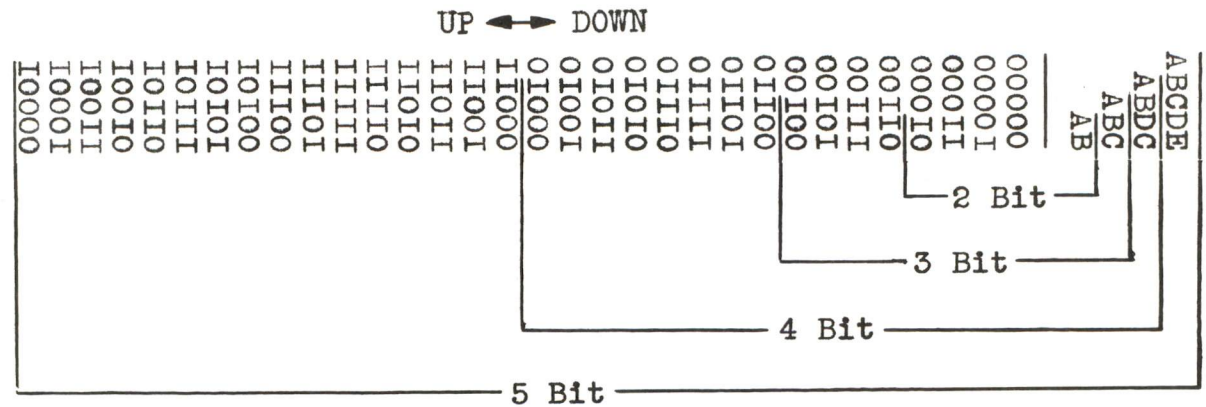
COUNT SEQ A B C	PREVIOUS COUNT						BIT OF PREVIOUS COUNT COMPLEMENTED	
	UP			DOWN			UP	DOWN
A B C	A B C	A B C	A B C	A B C	A B C			
0 0 0	1 0 0	0 0 1	0 0 1	0 0 1	0 0 1	A	C	
0 0 1	0 0 0	0 1 1	0 1 1	0 1 1	0 1 1	C	B	
0 1 1	0 0 1	0 1 0	0 1 0	0 1 0	0 1 0	B	C	
0 1 0	0 1 1	1 1 0	1 1 0	1 1 0	1 1 0	C	A	
1 1 0	0 1 0	1 1 1	1 1 1	1 1 1	1 1 1	A	C	
1 1 1	1 1 0	1 0 1	1 0 1	1 0 1	1 0 1	C	B	
1 0 1	1 1 1	1 0 0	1 0 0	1 0 0	1 0 0	B	C	
1 0 0	1 0 1	0 0 0	0 0 0	0 0 0	0 0 0	C	A	

DOWN
↑
↓
UP



*Output of counting logic shown for Up Counter

Fig. V.1



	DIRECTION OF COUNT	A	B	C	D	E
5 Bit	UP	$(A \oplus B) \bar{C} \bar{D} \bar{E}$	$(\bar{A} \oplus \bar{B}) C \bar{D} \bar{E}$	$(\bar{A} \oplus B \oplus C) D \bar{E}$	$(\bar{A} \oplus B \oplus C \oplus D) E$	$(\bar{A} \oplus B \oplus C \oplus D \oplus E)$
	DOWN	$(\bar{A} \oplus \bar{B}) \bar{C} \bar{D} \bar{E}$	$(A \oplus B) C \bar{D} \bar{E}$	$(A \oplus B \oplus C) D \bar{E}$	$(A \oplus B \oplus C \oplus D) E$	$(A \oplus B \oplus C \oplus D \oplus E)$
4 Bit	UP	$(A \oplus B) \bar{C} \bar{D}$	$(\bar{A} \oplus \bar{B}) C \bar{D}$	$(\bar{A} \oplus B \oplus C) D$	$(\bar{A} \oplus B \oplus C \oplus D)$	
	DOWN	$(\bar{A} \oplus \bar{B}) \bar{C} \bar{D}$	$(A \oplus B) C \bar{D}$	$(A \oplus B \oplus C) D$	$(A \oplus B \oplus C \oplus D)$	
3 Bit	UP	$(A \oplus B) \bar{C}$	$(\bar{A} \oplus \bar{B}) C$	$(\bar{A} \oplus B \oplus C)$		
	DOWN	$(\bar{A} \oplus \bar{B}) \bar{C}$	$(A \oplus B) C$	$(A \oplus B \oplus C)$		
2 Bit	UP	$(A \oplus B)$	$(\bar{A} \oplus \bar{B})$			
	DOWN	$(\bar{A} \oplus \bar{B})$	$(A \oplus B)$			

Fig. V.2

2. The term for any bit of an up counter other than the most significant bit is the complement of the ring sum of all the terms excluding the lesser significant bits Anded with the next less significant bit and the And of the complement of all lesser significant bits.

3. The term for the most significant bit is ring sum of the most and next most significant bits Anded with the And of the complement of all lesser significant bits.

4. For an n bit down counter the logic terms are the same except all of the ring sum portion of the up counter logic terms are complemented.

From the above relationships a set of rules are self-evident for determining the complement pulse gate logic terms for all bits of an n bit Gray code counter. As an example, for a seven bit Gray code up counter in which the bits are labelled A,B,C,D,E,F, and G, A being the most significant bit and G the least significant bit, the seven terms are as follows:

$$\begin{aligned}
 A; & (A \oplus B) \bar{C} \bar{D} \bar{E} \bar{F} \bar{G} \\
 B; & \overline{(A \oplus B)} C \bar{D} \bar{E} \bar{F} \bar{G} \\
 C; & \overline{(A \oplus B \oplus C)} \bar{D} \bar{E} \bar{F} \bar{G} \\
 D; & \overline{(A \oplus B \oplus C \oplus D)} E \bar{F} \bar{G} \\
 E; & \overline{(A \oplus B \oplus C \oplus D \oplus E)} F \bar{G} \\
 F; & \overline{(A \oplus B \oplus C \oplus D \oplus E \oplus F)} G \\
 G; & \overline{(A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G)}
 \end{aligned}$$

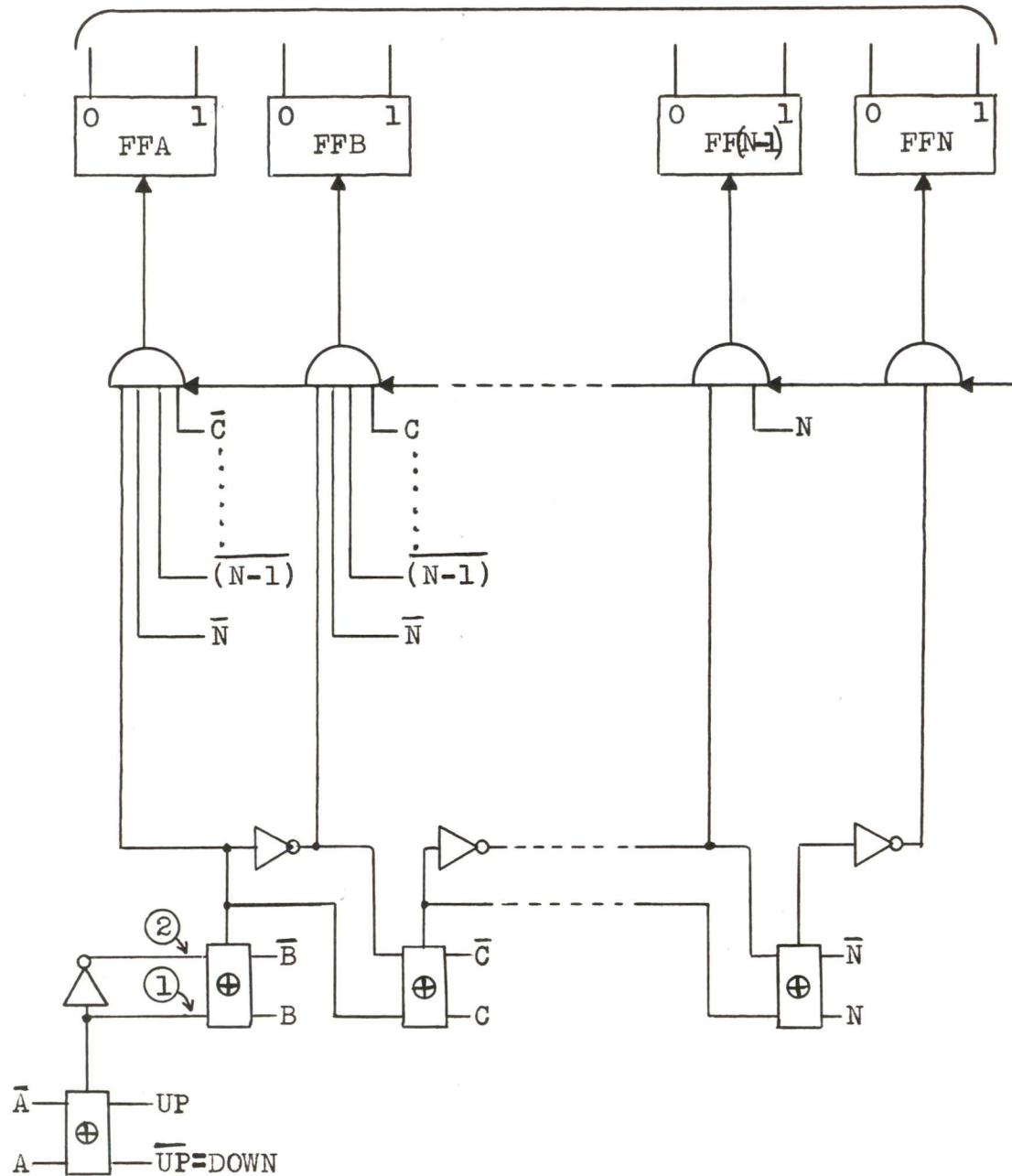
Complementing the ring sum portion of each term yields the logic terms for a down counter.

By complementing any bit in a ring sum term the ring sum term is complemented. This leads us to a simple method of reversing the count in an n bit Gray code counter resulting in an up-down counter. Notice that the most and next most significant bits appear in all ring sum portions of all terms. Simply by providing a means of reversing the sense of either (not both) of these bits at will, a method of reversing the direction of count is provided. Fig. V.3 shows an n bit up-down Gray code counter embodying all thoughts of this section.

N BIT PSEUDO-GRAY CODE COUNTER: Remembering that a pseudo-Gray code is only a combination of complemented and/or permuted bits it is easily seen that the Gray code counter discussion of the previous section applies if, in addition, the counting logic inputs are properly complemented and permuted (normalized) so that the proper counting sequence occurs. Therefore no more discussion of pseudo-Gray code counters will be pursued.

OTHER MONOSTROPHIC CODE COUNTERS: The method shown in Fig. V.1 for determining the logic terms required for the complement pulse gate enabling levels can be used for any full or foreshortened monostrophic (polystrophic for that matter) code counting sequence. For example, the 4 bit (4 bits per decade) Datex described in the last chapter is considered. See Fig. V.4.

TO LOAD AND COUNTING LOGIC

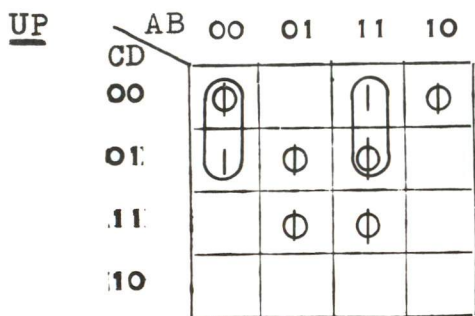


To count in only one direction eliminate \oplus circuit to which A, A, UP and UP are connected and connect A and A as follows: A to ① and A to ② for up counter, A to ② and A to ① for down counter.

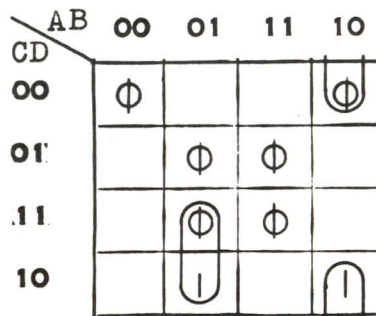
N BIT GRAY CODE COUNTER

Fig. V.3

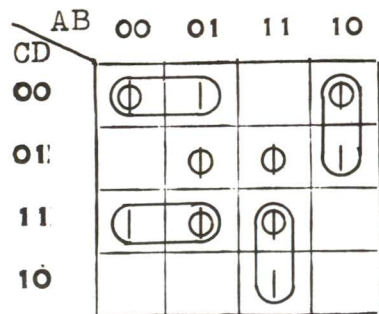
DATEX SEQ				PREVIOUS COUNT				BIT OF PREVIOUS COUNT COMPLEMENTED			
A	B	C	D	UP		DOWN		UP		DOWN	
A	B	C	D	A	B	C	D	A	B	C	D
0	0	0	1	1	0	0	1	0	0	1	1
0	0	1	1	0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1	0	1	1	0
0	1	1	0	0	0	1	0	0	1	0	0
0	1	0	0	0	1	1	0	1	1	0	0
1	1	0	0	0	1	0	0	1	1	1	0
1	1	1	0	1	1	0	0	1	0	1	0
1	0	1	0	1	1	1	0	1	0	1	1
1	0	1	1	1	0	1	0	1	0	0	1
1	0	0	1	1	0	1	1	0	0	0	1



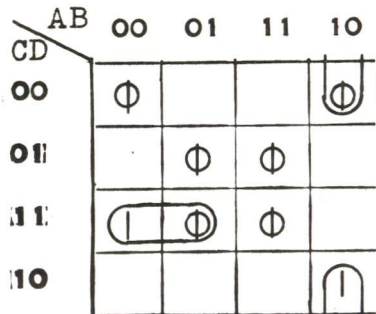
$$FFA = \overline{(A \oplus B)} \overline{C}$$



$$FFB = \overline{A}BC + A\overline{B}D$$



$$FFC = \overline{(C \oplus D)} \overline{A} + (B \oplus C) A$$



$$FFD = \overline{A}CD + A\overline{B}D$$

DOWN Datex is a reflected code with reflectivity dependent upon the A bit. Therefore, reversing sense of the A bit by counting logic reverses direction of count.

COUNTING LOGIC FOR ONE-DECADE DATEX COUNTER

Fig. V.4

Usually when the Datex code is used there is more than one decade in which case the Datex counting sequence of each decade other than the most significant decade does not pass directly from 0 to (2^n-1) and (2^n-1) to 0. Instead, each decade counts to the end of a sequence (zero or nine), the next more significant decade steps one count, then the first decade counts to the other end of its sequence (nine or zero). Any decade counts up when the next more significant decade is even, and down when the next more significant decade is odd. There is a similarity in this counting sequence with that of a Gray code, and the techniques employed to make use of this characteristic in Gray code counters (described in next section) can be used to control the direction of count in each decade of a Datex code counter.

CASCADING GRAY CODE COUNTERS: An examination of an n bit Gray code reveals that if the n bits are divided into two groups, say l bits and m bits where $l+m = n$, an interesting characteristic emerges. Look at the 4 bit code in Fig. V.5 that has been divided into two 2 bit groups. The group including C and D counts up, then down, then up and then down. The direction of count is dependent upon AB being even or odd; up when AB is even, down when AB is odd. This relationship holds for any n bit Gray code counter regardless where the division is. Also, if the

	A	B	C	D	
even	0	0	0	0	
	0	0	0	1	↓up
	0	0	1	1	
	0	0	1	0	
odd	0	1	1	0	
	0	1	1	1	↑down
	0	1	0	1	
	0	1	0	0	
even	1	1	0	0	
	1	1	0	1	↓up
	1	1	1	1	
	1	1	1	0	
odd	1	0	1	0	
	1	0	1	1	↑down
	1	0	0	1	
	1	0	0	0	

Fig. V.5

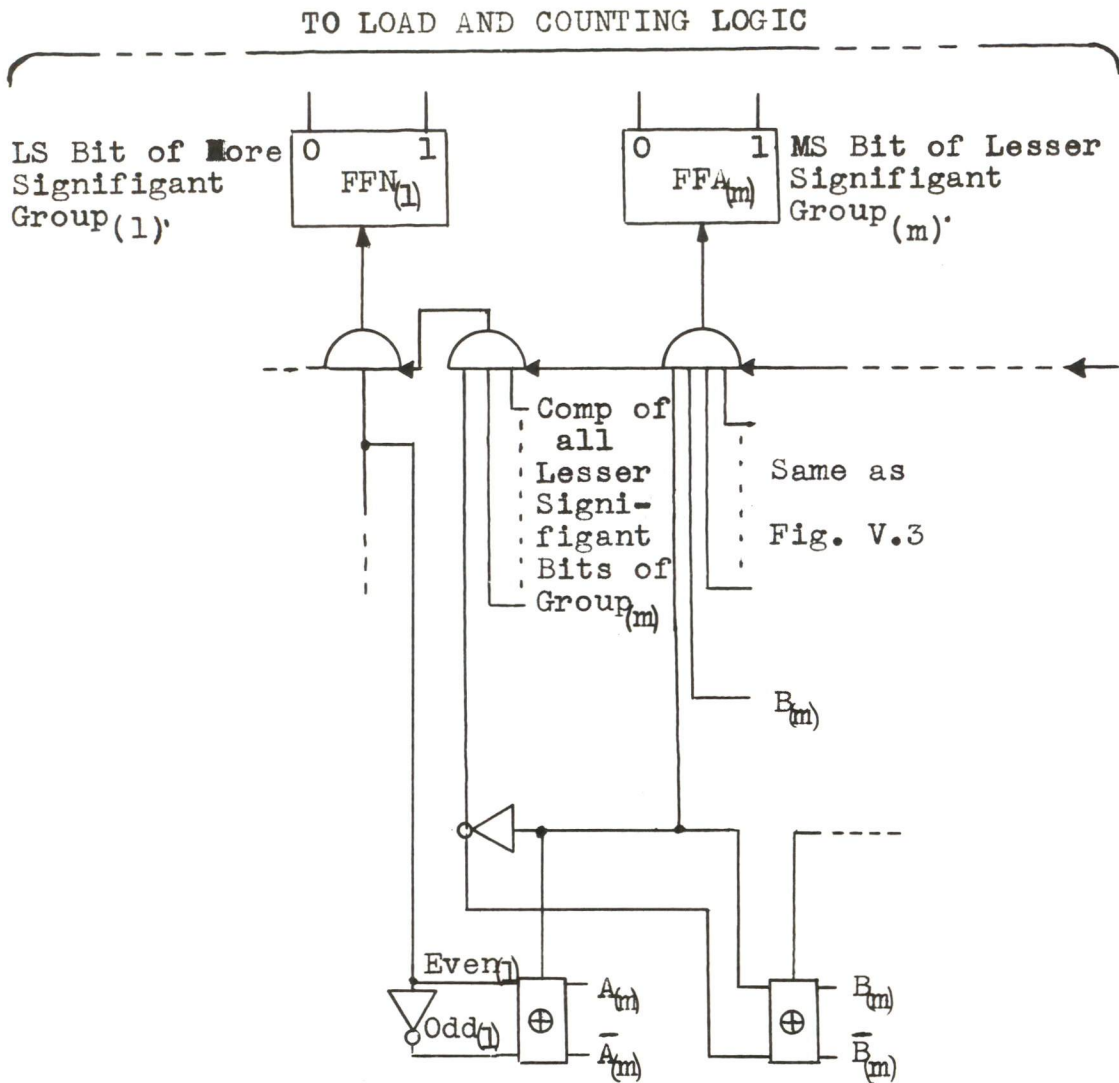
code is divided into more than two groups the relationship holds for any two adjacent groups. This characteristic of Gray codes allows one to cascade many smaller up-down Gray code counters with slightly modified counting logic to make one large Gray code counter (and to cascade decades of the Datex decade counters mentioned in the previous

section).

To go directly from the first count to the 2^m th count or vice versa of a m bit counter the most significant bit is the one that is complemented. If the logic term for the most significant bit is modified so that this change is not possible, and a $(m+1)$ th logic term is developed in the counting logic to gate the counting pulse to the next more significant group when the first group has reached the "end" we have solved part of the problem. The other part of the problem is already solved in that the logic term of the next more significant group's least significant bit indicates whether that group is odd or even (the ring sum of any number of bits is $S_{(odd)}$ of those bits and the complement of the ring sum is $S_{(even)}$). Therefore, the logic term of the next more significant group's least significant bit can also be used to tell the next

lesser significant group which direction to count (an input to the up-down control logic described earlier in this chapter). For the Datex decade this is not true. Extra logic is therefore required to determine if a decade is even or odd. Hence the desired action is as follows: When a group is counting up (next more significant group is even), the counting logic prevents the transition from the highest count to zero (the first count) but instead steps the next more significant group by one, changing it from even to odd which reverses the direction of count in the first group. Fig. V.6 demonstrates this method of cascading Gray code counters to make one larger one.

SWITCH TAIL RING COUNTER: The switch tail ring counter mentioned in the last chapter is simply a shift register whose most significant bit's transposed output is fed to the input of the least significant bit when shifting left (counting up), and the transposed output of the least significant bit is fed to the input of the most significant bit when shifting right (counting down). Hence, when counting up the complement of the most significant bit is shifted into the least significant bit and, when counting down the complement of the least significant bit is shifted into the most significant bit. For an n bit counter, a foreshortened monostrophic counting sequence of $2n$ counts is generated. Fig. V.7 shows the counting sequence for a 5 bit switch tail counting sequence.



Method of cascading Gray code counters into one larger counter. Only differences and additions to Gray code counter shown in Fig. V.3 are shown. The differences and additions affect only the two bits adjacent to the division (point of cascading) of groups (1 & m).

Fig. V.6

Notice that for four counts only 2 bits are required, the same as for a Gray code, and is in fact a 2 bit Gray code. For eight counts, four bits are required, only one more than for a Gray code with the same number of counts. For ten counts, five bits are required, only one more than for a foreshortened monostrophic code that follows the Gray to binomial binary conversion rules. Above ten counts many more bits are required than for more compact monostrophic codes. Since only 2 bits must be sampled per each of the 2^n combinations as discussed in the previous chapter, a switch tail ring counter may be preferable from an economy standpoint if decoding (rather than conversion) is required and the total count is not more than about ten.

	A	B	C	D	E
	0	0	0	0	0
	0	0	0	0	1
down	0	0	0	1	1
↑	0	0	1	1	1
↓	0	1	1	1	1
up	1	1	1	1	1
	1	1	1	1	0
	1	1	1	0	0
	1	1	0	0	0
	1	0	0	0	0
	1	0	0	0	0

Fig. V.7

SIMULTANEOUS GENERATION OF MONOSTROPHIC & POLY-

STROPHIC CODES: Many times when generating a monostrophic code count, it is desirable to have a polystrophic binary code count generated simultaneously with the monostrophic code.

Immediately, one may feel that the easiest and cheapest way to accomplish this is to count in one code with conversion logic for the other code connected to the output of the counter yielding both codes. Because methods of counting in binomial binary and conversion from binomial binary to Gray codes are so well known and simple, this might be the tendency. Where "on the fly" sampling of the monostrophic code is required this approach is not satisfactory because the ambiguities during transition from one count to the next in the binomial binary counter's output are transferred through the conversion logic resulting in ambiguities during transition of the monostrophic code. Therefore, if conversion logic on the output of the counter is to be used to generate both codes in parallel the counter must count monostrophically, with conversion to the binomial binary code. This approach yields a monostrophic code with no ambiguities, with the simultaneous generation of the binomial binary code.

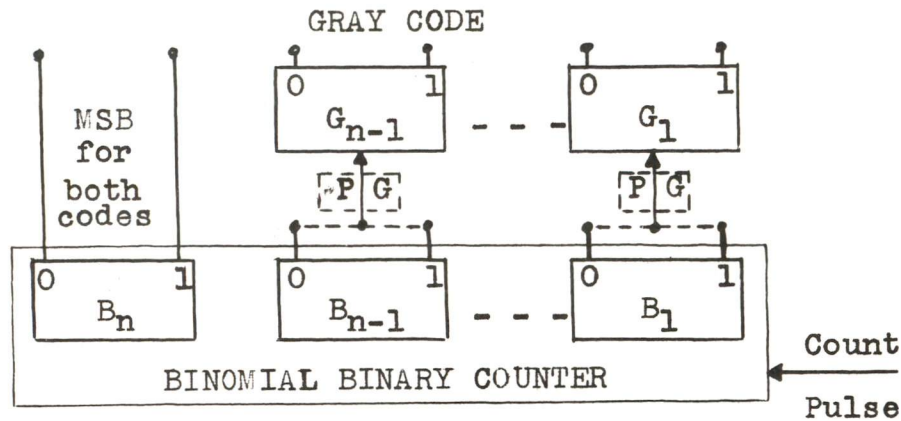
CONTINUOUS CONVERSION OF BINOMIAL BINARY TO GRAY: If the monostrophic code is Gray or pseudo-Gray and the polystrophic code is binomial binary the continuous conversion method shown in Fig. V.8 and described in the following discussion is applicable.

A relationship between corresponding bits (except most significant bits which are always equal) of binomial binary and Gray codes is that, in the up counting sequence of both

codes, when a bit changes from 0 to 1 in the binomial binary code the corresponding bit of the Gray code is complemented. A 1 to 0 transition of any binomial binary bit does not affect the Gray code. The opposite is true if the counting sequence of both codes is down. This relationship can be used to enable the transitions of bits in an n bit binomial binary counter to modify the corresponding bits of an (n-1) bit register which contains the lesser significant bits of the Gray code (or pseudo-Gray code if the output sensing pattern is complemented and/or permuted) count. Fig.V.8 demonstrates this approach. Using some manufacturers electronic logic modules this approach is the cheapest way to generate Gray (or pseudo-Gray) codes.

In sequential switching schemes using relay flip-flops to generate Gray (or pseudo-Gray) code counting sequences this approach is a good one because of the reduction of relay contacts (paid for by doubling the number of relays). Fig. V.9 shows an all relay n bit binomial binary counter with continuous conversion to an n bit Gray code.

N BIT DECODERS FOR COUNTING LOGIC: Most electronic logic manufacturers include in their product lines 2, 3 and occasionally 4 bit decoders. These decoders usually have as inputs two lines per bit (the logical value of the bit and the complement) and 2^n output lines, of which only



Connect complement input of each Gray bit (less MSB) to the 1 output of the corresponding binary bit for up counting sequence and the 0 output for down counting sequence. If flip-flops require a pulse instead of 0-1 level change pulse generators must be added as shown.

CONTINUOUS CONVERTER BINOMIAL BINARY TO GRAY

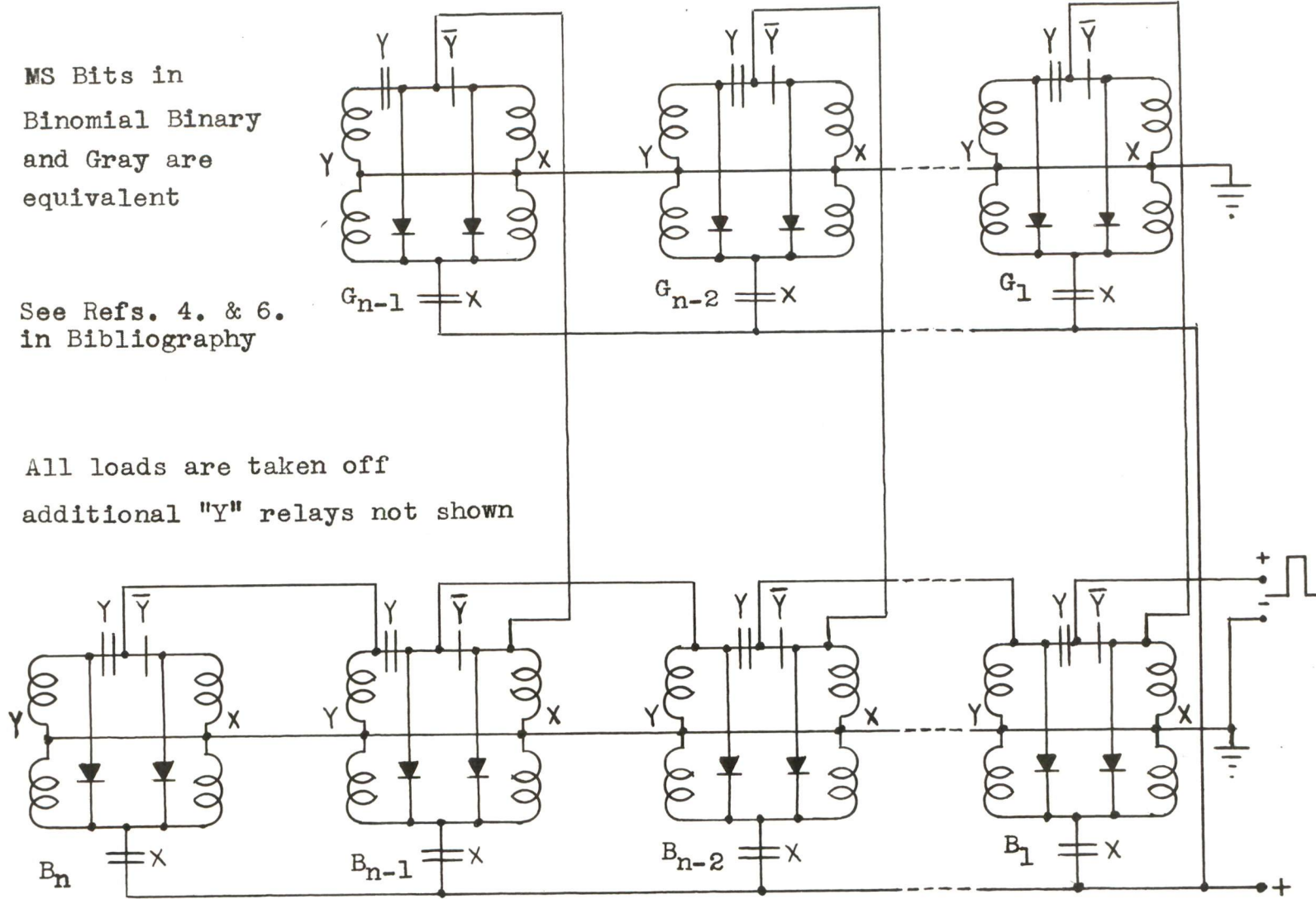
Fig. V.8

one may be selected at one time. Each output line is usually labelled with the decimal equivalent of the n bit binomial binary number which, as an input, selects that line. If an n bit decoder is connected to the output of an n bit flip-flop register, the proper output lines can be Or'd into n pulse gates, each one associated with the complement input of each flip-flop in the register. By pulsing the pulse gates the contents of the register goes through a counting sequence dependent upon the pattern of Or'ing the decoder output lines. For a monostrophic counting sequence, each decoder output line will be associated with

MS Bits in
Binomial Binary
and Gray are
equivalent

See Refs. 4. & 6.
in Bibliography

All loads are taken off
additional "Y" relays not shown



CONTINUOUS BINOMIAL BINARY TO GRAY CONVERTER

Fig. V.9

only one pulse gate. This is not true for a polystrophic counting sequence.

This approach is, in effect, the method used in binary counters that operate on the principle of "anticipated carry" with the accompanying advantages of equal settling time of the counting logic for all counts and faster propagation of logic levels from the output of the register to the pulse gates enabling faster counting. (Note that the counting logic shown in Fig. V.3 requires propagation of a level change through the cascaded ring sum logic of all n bits when either of the two most significant bits or the direction of count is complemented.)

In an up-down Gray counter, or when many small counters are cascaded into one large counter (requiring the control of direction of count in each small counter), the carry to the next most significant group and the inhibition of the "return to zero" in each group is more easily implemented using the decoder counting logic technique under discussion in this section. Fig. V.10 shows a 6 bit Gray counter comprising two 3 bit counters, each using a 3 bit decoder as part of the counting logic. A comparison with Fig. V.3 and Fig. V.6 shows that this approach for an up-down counter made up of smaller counters is simpler.

To determine the Oring pattern of the decoder outputs, write out the counting sequence and, beside each count, write the decimal equivalent of the binomial binary

interpretation of the count. As previously explained in this chapter, the contents of the counter determines which bit is complemented on the next count. The decimal numbers associated with the counts preceding the complementing of a bit in a counting sequence correspond to the lines that must be Ored into that bit's complement pulse gate.

If the counting sequence is reflective the counting sequence may be reversed by reversing the sense by the decoder of the most significant bit. When groups of small counters are cascaded into one larger counter the decimal equivalent of the binomial binary interpretation of the last count in the sequence is the one that enables the carry (and inhibits the "return to zero" by not being associated with any complement pulse gate of that group).

When cascading x groups of m bit counters into one xm bit counter ($xm=n$) rather than using an n bit counter like that in Fig. V.3, the propagation time of the logic is dependent upon x rather than n , hence the possibility of a faster counter.

CHAPTER VI. CONCLUSION

Most likely, while studying the foregoing, the imaginative reader has already thought of varied applications for monostrophic codes. Generally monostrophic codes have applications not only where asynchronous "on-the-fly" sampling is required, but also where continuous monitoring of the continually changing code to detect a value (or values) is required. In both cases, if a polystrophic code were used, precautions against ambiguities would need to be taken which adds complexity to the system.

Gray or pseudo-Gray codes are the easiest of the full count monostrophic codes to generate and/or convert. Of the foreshortened codes, those which follow the Gray to binomial binary conversion rules are the easiest to handle.

Where conversion between a monostrophic code and another specified binary coded numbering system is required, judicious selection (if both codes are not already specified) of the monostrophic code will result in simpler conversion logic, particularly if Gray to binomial binary conversion rules apply. This was shown in the examples of Chapter IV.

The pulse counting techniques shown in Chapter V (except for the continuous converter) have a characteristic which is highly desirable in many applications; equal time delay between any count and the new count after the count pulse (assuming equal logical delays in all flip-flops).

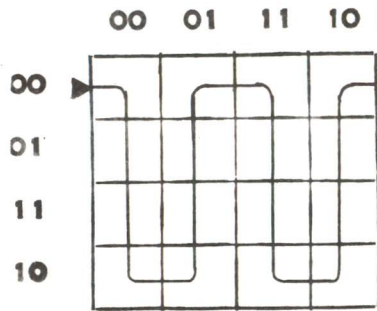
This is not so with polystrophic code counters that depend upon carry propagation. Also, the use of parallel counting logic, e. g., n bit decoders, to increase the maximum counting rate is simpler for a monostrophic code counter than with a polystrophic code counter because each output line is associated with only one complement (or set and reset) gate of the monostrophic code counter.

Besides the usual mechanical (and sometimes electronic) analog to digital conversion processes using monostrophic codes, there are many other potential applications of these codes which cannot be appreciated unless one has a "feel" for them. The overall objective of this paper is to present to the reader a better insight of monostrophic codes based on my study of them to enable him to better evaluate possible new applications employing them.

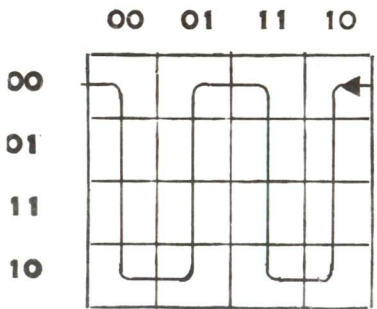
APPENDIX I
4BIT PSEUDO-GRAY CODES

The following pages contain all combinations of complemented bits and permutations of the two most significant bits of the 4 bit Gray Code. They are grouped so that each group has the same counting sequence as shown on the accompanying Karnaugh Maps. The top group of each page has the reverse counting sequence as the bottom group on the page.

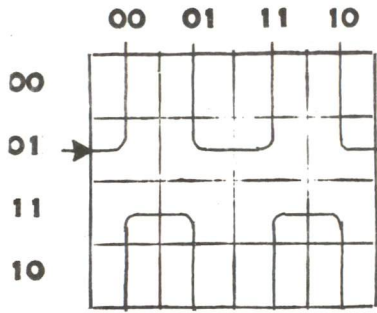
The least significant two bits' permutations are not included because, in analyzing or normalizing a pseudo-Gray code, the bits are arranged so that the least significant bit is considered the bit that is complemented most often in the counting sequence, the bit with the next fastest rate of change is considered the next to the least significant bit, etc. The two most significant bits are both complemented the same number of times in a counting sequence, hence their arrangement is arbitrary, and only their permutations are included in this appendix.



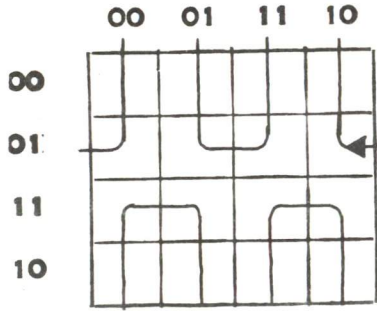
A	B	C	D	\bar{A}	\bar{B}	\bar{C}	\bar{D}	B	\bar{A}	\bar{C}	D	\bar{B}	A	\bar{C}	D
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	0	1	0	0	0	1	0	0	1	1	0	1	1	1
1	1	1	1	0	0	0	1	1	0	0	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0



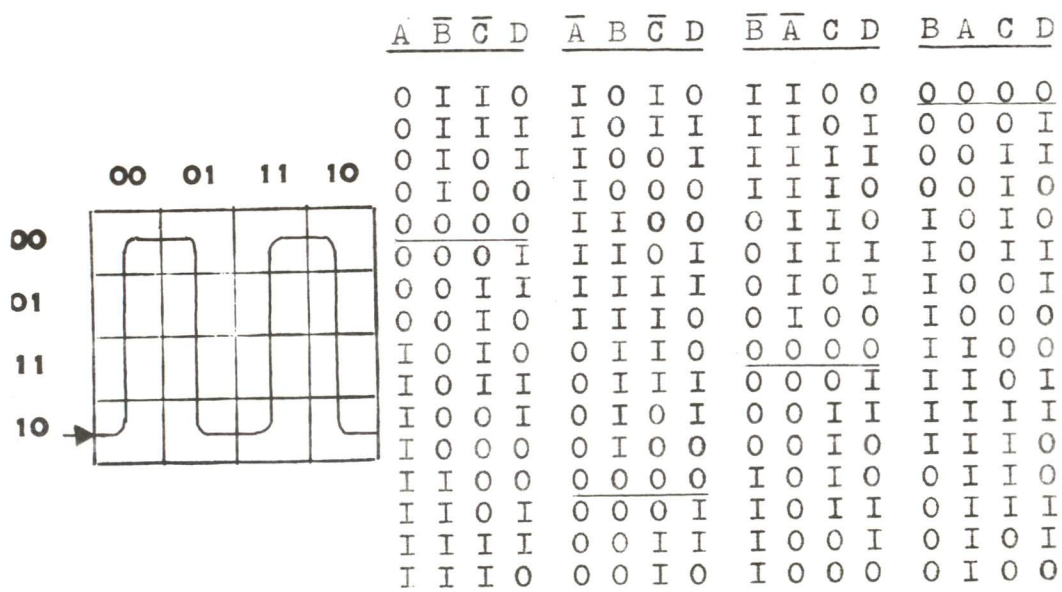
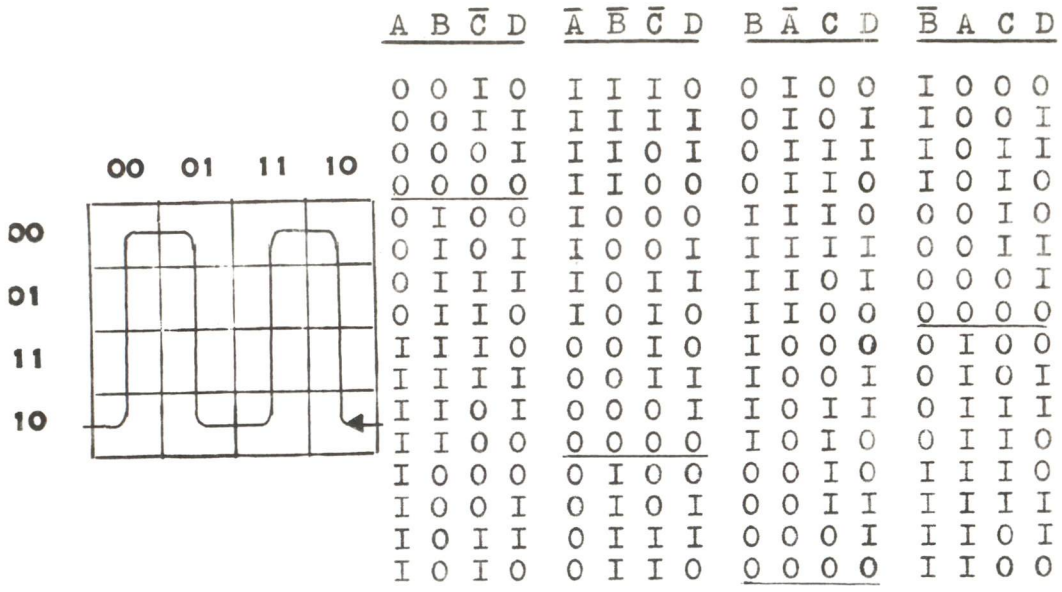
A	\bar{B}	C	D	\bar{A}	B	C	D	\bar{B}	\bar{A}	\bar{C}	D	B	A	\bar{C}	D
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	1
1	1	0	1	0	0	0	0	1	1	0	1	1	1	1	1
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0

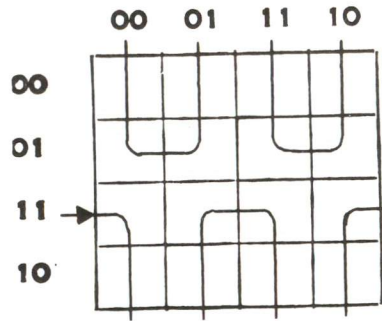


A	B	C	\bar{D}	\bar{A}	\bar{B}	C	\bar{D}	B	\bar{A}	\bar{C}	\bar{D}	\bar{B}	A	\bar{C}	\bar{D}
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	1	0	0	0	0	0
0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	1
0	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1
1	1	1	0	0	0	0	0	1	0	1	0	0	1	1	1
1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0
1	1	1	1	0	0	1	1	1	0	0	0	1	0	1	1
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1

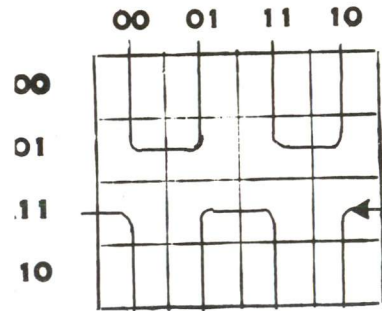


A	\bar{B}	C	\bar{D}	\bar{A}	B	C	\bar{D}	\bar{B}	\bar{A}	\bar{C}	\bar{D}	B	A	\bar{C}	\bar{D}
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	1	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1





A	B	\bar{C}	\bar{D}	\bar{A}	\bar{B}	\bar{C}	\bar{D}	B	\bar{A}	C	\bar{D}	\bar{B}	A	C	\bar{D}
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	1	1	0	0	0	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1

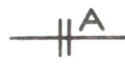






A	\bar{B}	\bar{C}	\bar{D}	\bar{A}	B	\bar{C}	\bar{D}	\bar{B}	\bar{A}	C	\bar{D}	B	A	C	\bar{D}
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	0	0	0	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	1	1	1	0	0	0	1	1	0	1	0	1	0
0	0	1	0	1	1	1	0	0	1	0	1	1	0	0	1
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	0	1	1	1
1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	1

APPENDIX II

LEGEND OF SYMBOLS USED IN THIS PAPER

	<p>Exclusive Or; Shown with two variable input, and with two variable with complements input.</p>		
	<p>Flip-flop; Assumed to have logical delay enabling simultaneous read-in and sampling of output. Shown with Set, Reset and Complement inputs. An abbreviated method of showing a Reset input (Clear input) is an arrow at lower left-hand corner of flip-flop.</p>		
	<p>And Gate</p>		<p>Or gate</p>
	<p>Nor Gate</p>		<p>Nand gate</p>
	<p>Inverter</p>		<p>Diode</p>

	<p>Normally open switch or relay contacts associated with switch or relay A.</p>
	<p>Normally closed switch or relay contacts associated with switch or relay A.</p>
<p> ∅</p>	<p>Assertion Optional Term Non-Assertion left blank</p> <p>} On Karnaugh Maps</p>
	<p>Relay Winding</p>
	<p>Logic Level Line</p>
	<p>Pulsed Line</p>

APPENDIX III
 EXCLUSIVE OR/RING SUM

The Exclusive Or function, also called the ring sum and commonly signified by the sign \oplus (and occasionally by Ψ), is an important logic function when working with generation and conversion of monostrophic codes. This function also appears in binary adders. In fact, the ring sum of two variables is often called the sum modulo two, because the ring sum of two variables satisfies the logic requirements of a half adder as shown.

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

RING SUM
 TRUTH
 TABLE

Y \ X	0	1
0	0	1
1	1	0

HALF ADDER

The following postulates and theorems will enable one to relate ring sum functions to binary logic operations.*

POSTULATES

- 1) $0 \oplus 0 = 0$
- 2) $1 \oplus 1 = 0$
- 3) $0 \oplus 1 = 1 \oplus 0 = 1$

THEOREMS

- 1) $X \oplus 1 = \bar{X}$
- 2) $X \oplus \bar{X} = 1$

*Samuel H. Caldwell, Switching Circuits and Logical Design, p. 667.

$$3) X \oplus X = 0$$

$$4) X \oplus \bar{X} = 1$$

$$5) X \oplus X \oplus \dots \oplus X = \begin{cases} X & \text{for odd number of terms} \\ 0 & \text{for even number of terms} \end{cases}$$

$$6) X \oplus Y = Y \oplus X = \bar{X}\bar{Y} \oplus XY = (X+Y)(\bar{X}+\bar{Y})$$

$$7) (X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

$$8) XY \oplus XZ = X(Y \oplus Z)$$

$$9) \overline{(X \oplus Y)} = \overline{(Y \oplus X)} = \bar{X}Y + X\bar{Y} = (\bar{X}+Y)(X+\bar{Y}) = \overline{(\bar{X}Y + X\bar{Y})}$$

Theorems 6) and 7) give the clue to the manipulation of more than two variables. For example, let us expand the ring sum of four variables, $A \oplus B \oplus C \oplus D$.

by theorem 7

$$A \oplus B \oplus C \oplus D = (A \oplus B \oplus C) \oplus D$$

by theorem 6

$$= (A \oplus B \oplus C) \bar{D} \oplus \overline{(A \oplus B \oplus C)} D$$

by theorem 7

$$= [(A \oplus B) \oplus C] \bar{D} \oplus \overline{[(A \oplus B) \oplus C]} D$$

by theorem 6

$$= [(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C] \bar{D} \oplus \overline{[(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C]} D$$

by theorems 6 and 9

$$\begin{aligned} &= [(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C] \bar{D} \oplus [(AB + \bar{A}\bar{B}) \bar{C} + (A\bar{B} + \bar{A}B) C] D \\ &= A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + ABC\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}D + \bar{A}\bar{B}CD \end{aligned}$$

Notice that the expanded term is the symmetric function $S_{(1,3)}(A,B,C,D)$. It can be shown that the ring sum of n variables is the symmetric function $S_{\text{odd}}(A,B,---,N)$

and the complement of the ring sum of n variables is $S_{\text{even}}(A,B,---,N)$. Because the ring sum of more than two variables is difficult to recognize on a Karnaugh Map, the relationship of ring sum functions to symmetric functions is probably the easiest way to recognize ring sum functions of more than two variables.

BIBLIOGRAPHY

1. Barr, Paul "Analog to Digital Converters," Electromechanical Design, 8:165-176, January, 1964.
2. Building Blocks Technical Manual (Norwood, Mass.: Raytheon Co., Communications and Data Processing Operation, n. d.).
3. Caldwell, S. H. Switching Circuits and Logical Design (New York: John Wiley and Sons, Inc., 1958).
4. Clareed Control Modules, Manual 400 (Chicago: C. P. Clare and Co., 1962).
5. Couleur, John F. "BIDEC--A Binary-to-Decimal or Decimal-to-Binary Converter," IRE Transactions on Electronic Computers, 313-316, December, 1958.
6. Counting With Clareed Control Modules, Application Manual 401 (Chicago: C. P. Clare and Co., 1962).
7. Datex Code, Bulletin 001 (Monrovia, Calif.: Datex Corp., 1963).
8. Digital Modules, Catalog A-705A (Maynard, Mass.: Digital Equipment Corp., 1962).
9. Evans, D. S. Digital Data (London: Hilger and Watts, Ltd., 1961).
10. Grabbe, Ramo and Wooldridge (Editors) Handbook of Automation, Computation and Control (New York: John Wiley and Sons, Inc., 1959). 3 Vols.
11. Graf, R. F. Dictionary of Electronics, 2nd Edition (Indianapolis: Howard W. Sams and Co., Inc., 1963).
12. Holcombe, Waldo "Relays That Challenge Semiconductors," Electronics, 37: 56-60, March 23, 1964.
13. Humphrey, W. S. Switching Circuits with Computer Applications (New York: McGraw-Hill, 1958).
14. Laboratory Module Handbook (Maynard, Mass.: Digital Equipment Corp., 1963).
15. Maley, Gerald A., and Earle, John The Logic Design of Transistor Digital Computers (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1963).

16. Marcus, M. P. Switching Circuits for Engineers (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1962).
17. Meyer, B. W. "Arbitrary-Length Binary Counters," Electronics, 36: 34-36, December 27, 1963.
18. Robbins, Lionel "One-Brush Encoder," Instruments and Control Systems, 36: February, 1963.
19. "Shaft-Angle Encoders" Computer Design, 2: 16-41, December, 1963.
20. Shaft Position Encoders, Bulletin 312-B (Monrovia, Calif.: Datex Corp., 1963).
21. Wilson, M. Carr "Gray to Binary Converter," Instruments and Control Systems, 35: June, 1962.

MONOSTROPHIC CODES

A Thesis
Presented to
the Faculty of the Department of Electrical Engineering
Northeastern University

by
DALE SHERWOOD COCKLE

June, 1964

MONOSTROPHIC CODES

A Thesis
Presented to
the Faculty of the Department of Electrical Engineering
Northeastern University

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science

by
Dale Sherwood Cockle
June, 1964

NORTHEASTERN UNIVERSITY

Graduate School

Thesis Title: Monostrophic Codes

Author: Dale Sherwood Cockle

Department: Electrical Engineering

Approved for Thesis Requirements of the Master of Science
Degree in Electrical Engineering

Marcello J. Carrabes
Thesis Advisor

Arthur E. Fitzgerald
Chairman of the Department

Graduate School Notified of Acceptance

George W. Hankinson
Program Director

ACKNOWLEDGEMENTS

This thesis contains the contributions of many organizations and individuals, whose material, advice and encouragement made this endeavor possible. I have given bibliographical reference to those sources which I consider reasonably accessible to the reader. However, I wish to express my personal gratitude here to the following organizations for their kind contributions.

C. P. Clare and Co.	Chicago, Ill.
Computer Control Co., Inc.	Framingham, Mass.
Datex Corp.	Monrovia, Calif.
Digital Equipment Corp.	Maynard, Mass.
Engineered Electronics Co.	Santa Ana, Calif.
General Precision, Inc. Commercial Computer Div.	Burbank, Calif.
Kearfott Div.	Little Falls, N. J.
Northern Precision Laboratories, Inc.	Franklin Lakes, N. J.
Perkin-Elmer Corp.	Norwalk, Conn.
Raytheon Co.	Lexington, Mass.
Theta Instrument Corp.	Saddlebrook, N. J.
W. & L. E. Gurley	Troy, N. Y.
Wang Laboratories, Inc.	Tewksbury, Mass.
Wayne-George Corp.	Newton, Mass.

My most grateful thanks go to the wonderful people at Digital Equipment Corporation, in particular, my dear friends Morton Ruderman and John O'Connell (Digital

Module Application Engineers) under whose patient tutelage I learned the practical side of digital logic implementation during my co-operative work periods, and to the personnel of the Advertising Department whose efforts resulted in the expert reproduction of this thesis.

To William E. Walker, New England District Manager of Engineered Electronics Company, I offer my thanks for his personal contributions.

I wish to express my appreciation to P. J. Lawrence, Publications Manager of the Datex Corporation, for his kind and prompt assistance.

I am greatly indebted to my thesis advisor, Marcello J. Carrabes, for his many hours of expert instruction in digital logic and techniques, and for the inspiration he gave which sustained me through the many long hours of work preparing this thesis.

To my wife, Joanna, whose understanding, encouragement, and long hours of typing contributed materially to the timely completion of this thesis, goes my most sincere and humble thanks.

D. S. C.

TABLE OF CONTENTS

Chapter I	Introduction.....	1
Chapter II	General.....	4
Chapter III	Monostrophic Code Synthesis.....	10
Chapter IV	Monostrophic-Polystrophic Code Conversion.....	20
Chapter V	Generation of Monostrophic Codes.....	42
Chapter VI	Conclusion.....	62
Appendix I	Four Bit Gray and Pseudo-Gray Codes	
Appendix II	Legend of symbols Used in this Paper	
Appendix III	Exclusive Or/Ring Sum	
Bibliography		

CHAPTER I. INTRODUCTION

Monostrophic codes, binary codes in which only one bit changes from count to count, are commonly found in applications where elimination of ambiguities during transition from one count to the next is desired.

The most common monostrophic code is the Gray code. Figure I.1 is an example of a 4 bit Gray code with the normally employed binomially weighted (2^{n-1} , - - -, 8,4,2,1) binary and decimal equivalents.

DECIMAL	BINARY	
	BINOMIAL 8 4 2 1	GRAY
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Fig. I.1

Digital logic engineers undoubtedly have unknowingly used the Gray code or other monostrophic codes as a tool for logic simplification and/or sequential switching

synthesis. The Karnaugh Map, which will be used as a tool in this paper, is such an application. Figure I.2 is an example of a 4 variable Karnaugh Map in which there is a change in only one variable between adjacent compartments.

	AB				
CD		00	01	11	10
00					
01					
11					
10					

4 VARIABLE KARNAUGH MAP

Fig. I.2

Other common applications of monostrophic codes are found in analog - digital and digital - analog conversion devices.

Although the Gray code is nothing new it is difficult to find references which comprehensively discuss monostrophic codes, particularly those which are not Gray. Therefore, by means of this paper, this author is attempting to present a compilation of information pertaining to monostrophic codes, the sources being found in texts, trade publications, manufacturers' product bulletins and applications notes, and that information which, because of the meager quantities found in the aforementioned sources,

has been self-generated. The succeeding chapters will discuss, with respect to monostrophic codes, the following topics:

- (1) Synthesis of monostrophic codes.
- (2) Conversion between monostrophic and polystrophic codes.
- (3) Generating monostrophic codes using electronic and switching logic elements (counters).

Preceding the above mentioned topics terms used in this paper will be defined.

CHAPTER II, GENERAL

In order to be precise, this author has attempted to employ in this paper terms that have commonly accepted definitions in the digital field. In so doing many small disparities have arisen that might lead to confusion if used in this paper without defining the terms herein. Therefore, I will commit the common sin of "defining definitions" before proceeding with the substance of this paper.

The Gray code many times is considered to be any monostrophic code. On the other hand, the terms Gray code and reflected code are often used synonymously. For the purpose of further discussion let us clear up this disparity with the following definitions because, clearly, not all monostrophic codes are reflected codes.

GRAY CODE: Normally the Gray code is considered to be a specific n bit counting sequence of 2^n counts having the characteristics of being non-weighted, monostrophic and reflected (see definition of reflected codes below), and represents a specific ordered numbering system of 2^n counts. In Chapter I an example of a 4 bit Gray code was given. The specific counting sequence may be explained as follows: the counting sequence for the least significant bit is 0110 repeated 2^{n-1} times, with the more significant bits going through the same sequence at half the rate as

numbering system. Notice that the reflected code is not

bit reflective code is shown representing a Radix 8 complement of the original count. In Figure II.3a a 3 usually the most significant bit) will yield the r-1's complementing a certain bit (same bit for all counts and must represent a numbering system of a radix r, and by

REFLECTIVE CODES: For a code to be reflected it

time, make the point clear. ability to be used directly most often and, at the same A 4 bit Gray code was used in appendix I because of its code with a table of comparisons of counting sequences. of complements and all permutations for a 4 bit Gray Appendix I contains an exhibit of all combinations

Fig. II.2

$A = F, B = E, C = G, D = H$

		$n = 4$			
	DECIMAL	BINOMIAL BINARY W X Y Z	ANORMAL GRAY E F G H	GRAY A B C D	
0	0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0
1	1	0 0 0 1	1 0 0 1	0 0 1 1	0 0 1 1
2	2	0 0 1 0	1 0 1 0	0 0 1 0	0 0 1 0
3	3	0 0 1 1	1 0 1 1	0 0 1 0	0 0 1 1
4	4	0 1 0 0	0 0 1 0	0 0 1 1	0 1 1 0
5	5	0 1 0 1	0 0 1 1	0 0 1 1	0 1 1 1
6	6	0 1 1 0	0 0 1 1	0 0 0 1	0 1 0 1
7	7	0 1 1 1	0 0 0 0	0 0 0 0	0 1 0 0
8	8	1 0 0 0	0 1 0 0	0 1 0 0	1 1 0 0
9	9	1 0 0 1	0 1 0 1	0 1 0 1	1 1 0 1
10	10	1 0 1 0	0 1 1 1	0 1 1 1	1 1 1 1
11	11	1 0 1 1	0 1 1 0	0 1 1 0	1 1 1 0
12	12	1 1 0 0	1 1 1 0	1 1 1 0	1 0 1 0
13	13	1 1 0 1	1 1 1 1	1 1 1 1	1 0 1 1
14	14	1 1 1 0	1 1 0 1	1 1 0 1	1 0 0 1
15	15	1 1 1 1	1 1 0 0	1 1 0 0	1 0 0 0

POLYSTROPHIC	#	7's COMP
0 0 0	0	7
0 0 I	1	6
0 I 0	2	5
0 I I	3	4
I I I	4	3
I I 0	5	2
I 0 I	6	1
I 0 0	7	0

a.

GRAY	#	7's COMP
0 0 0	0	7
0 0 I	1	6
0 I I	2	5
0 I 0	3	4
I I 0	4	3
I I I	5	2
I 0 I	6	1
I 0 0	7	0

b.

Fig. II.3

monostrophic. In Fig. II.3b the code is monostrophic and also Gray. By complementing the most significant bit in either reflected code of Fig. II.3, the $r-1$'s complement ($r=8$) is the result. The monostrophic and polystrophic reflected codes were shown to clearly point out that Gray and reflected are not synonymous, but simply that the Gray code (but not all monostrophic codes) is a reflected code.

CYCLIC: The Gray code is also called a cyclic code.

The Modern Dictionary of Electronics put out by Howard Sams defines a cyclic code as "any binary code that changes only one bit when going from one number to the number immediately following". This, in effect, is synonymous with monostrophic. This author has found that usage of the word cyclic in the digital industry means different things to different people. Some people interpret cyclic as returning upon itself by the same path or synonymously with reflective rather than as a monostrophic characteristic. Therefore, for the sake of being precise, I will avoid the use of the term cyclic.

MONOSTROPHIC CLOSURE: If the transition between the first and last counts of a counting sequence requires only one bit to change we have a monostrophic closure. The Gray code and pseudo-Gray codes possess this characteristic, but possession of this characteristic is not required for a code to be monostrophic.

FULL COUNT: A full count, as used in this paper, signifies a counting n bit sequence to which all of the 2^n counts have an assigned numerical value. We can see that a full count usually goes hand-in-hand with a numbering system having a radix of 2^n . A full counting sequence may be made monostrophic and/or reflected with or without monostrophic closure.

FORESHORTENED COUNT: When all 2^n combinations are not used to form a counting sequence we have a foreshortened

count. A foreshortened counting sequence may be made monostrophic, but to be reflective and/or close monostrophically the counting sequence must contain an even number of counts.

CHAPTER III. MONOSTROPHIC CODE SYNTHESIS

This chapter will discuss the synthesis of Gray, pseudo-Gray and other full and foreshortened count monostrophic codes, and will demonstrate the use of Karnaugh Maps and path diagrams as synthesis tools.

GRAY CODE: Because the Karnaugh Maps and path diagrams utilize the Gray code, the synthesis of the Gray code must be described independently. In effect, the definition of the Gray code given in Chapter II, describes the synthesis of the Gray code. For an n bit code starting with the least significant bit, form the sequence 0110 $2^{n-1}/4$ times. The next significant bit's sequence is the same but at half the rate, etc., until we get to the most significant bit which goes through only half the 0110 sequence, the first half of the sequence being 0's and the last half being 1's. This synthesis process, shown for a 4 bit code in Fig.III.1, yields a monostrophic code which monostrophically closes. If the following additional characteristics are present we have synthesized a reflected code which is Gray:

1. All 2^n counts represent a count of a numbering system.
2. All bits non-asserted (all 0's) represents zero.
3. Represented numbering system counts progressively from 0 to 2^n-1 .

KARNAUGH MAPS: Any Karnaugh Map is a table of all possible combinations of an n bit binary word so arranged

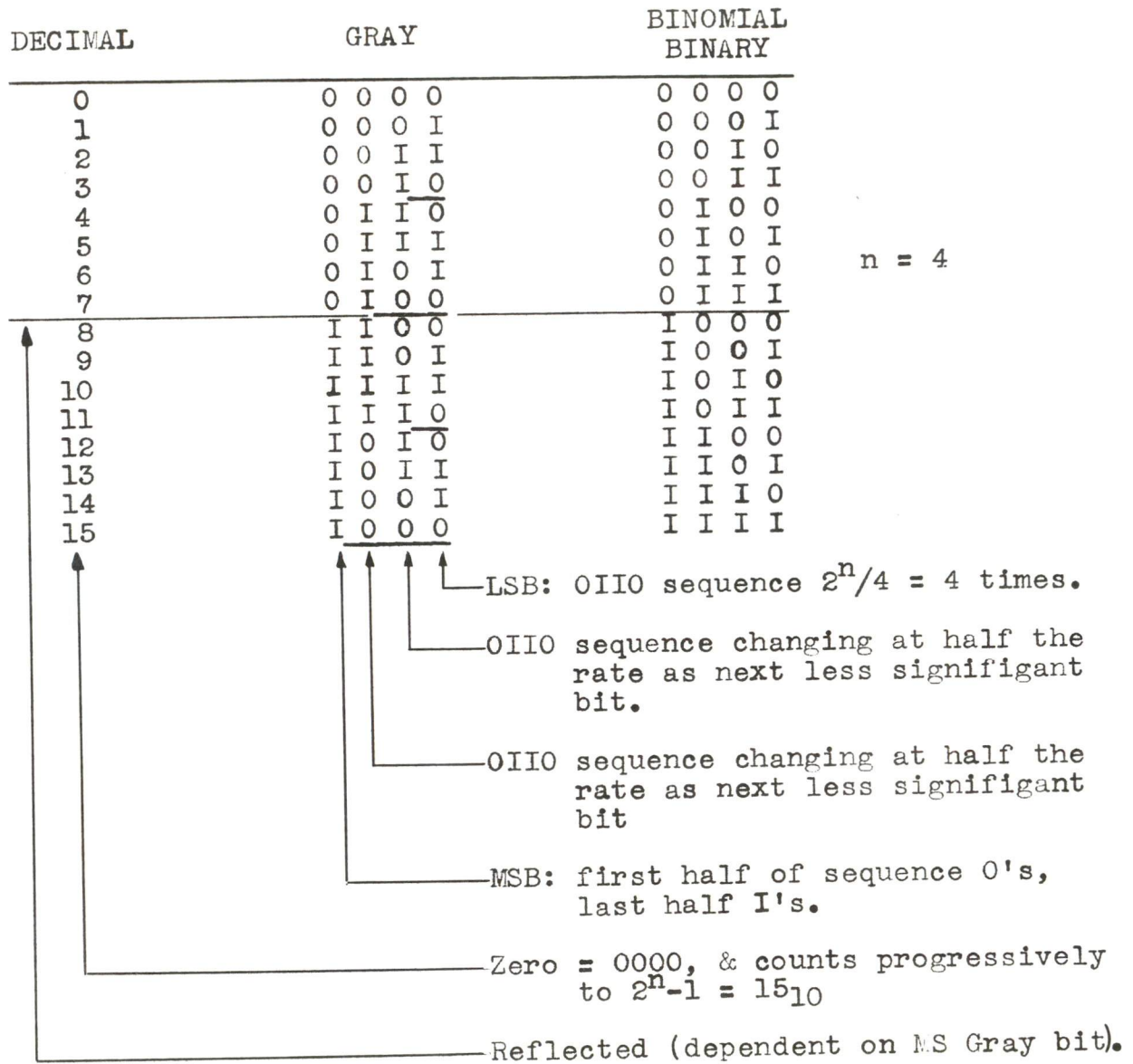


Fig. III.1

that there is only one bit change when going from any compartment to an adjacent or "mirrored" compartment. For maximum utility the map is arranged so that approximately half the variables (usually the more significant bits) are on the horizontal margin and the remaining variables (usually the lesser significant bits) are on the

vertical margin, e.g., a $2^2 \times 2^3$ or $2^3 \times 2^2$ configuration represents the 32 combinations of a 5 variable (bit) word.

In order that there be only one bit change between adjacent and "mirrored" compartments a reflected monostrophic code is used on the margins. In fact, the counting sequences of a Gray code are used on the margins. FigIII.2 shows 3 variations of a 5 variable Karnaugh Map.

Any path through a Karnaugh Map describes a monostrophic code.

PATH DIAGRAM: All the possible paths through a Karnaugh Map are difficult to see because one may jump to "mirrored" compartments, particularly if there are more than 4 variables (bits). To solve this problem the path diagram may be employed. The path diagram's use also simplifies the incorporation of code requirements during synthesis. This point will become evident in subsequent chapters.

Basically, the path diagram for an n bit code is formed by equally spacing n points in a circular pattern. The points are then numbered, starting with the n bit Gray coded zero, and progressing around the circle with the Gray code counting sequence. Then all those points are connected which differ by only one variable. For an n bit code each point will radiate n lines. The pattern generated is symmetric aiding in drawing a path diagram.

		ABC							
		DE	000	001	011	010	110	111	101
DE	00								
	01								
	11								
	10								

		BC			
		DE	00	01	11
DE	00				
	01				
	11				
	10				

A

		BC			
		DE	00	01	11
DE	00				
	01				
	11				
	10				

\bar{A}

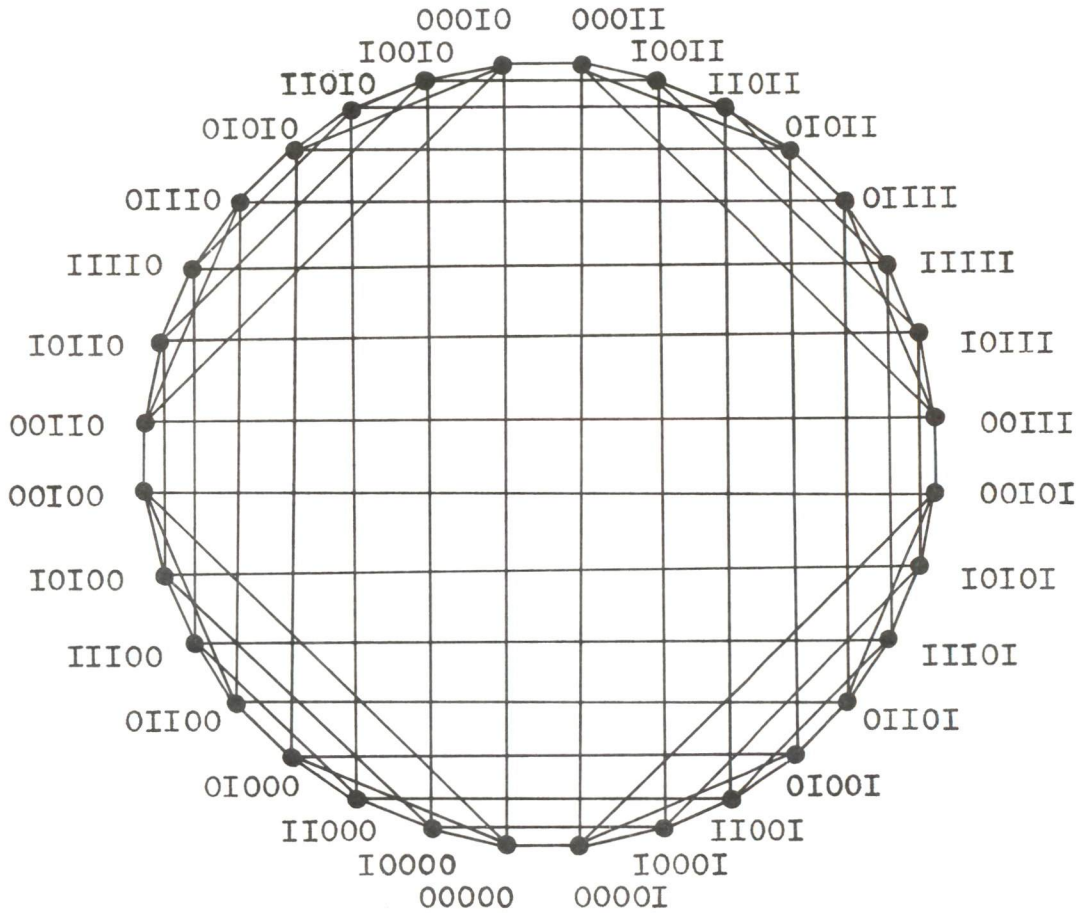
		AB			
		CDE	00	01	11
CDE	000				
	001				
	011				
	010				
	110				
	111				
	101				
	100				

3 VARIATIONS OF A 5 VARIABLE KARNAUGH MAP

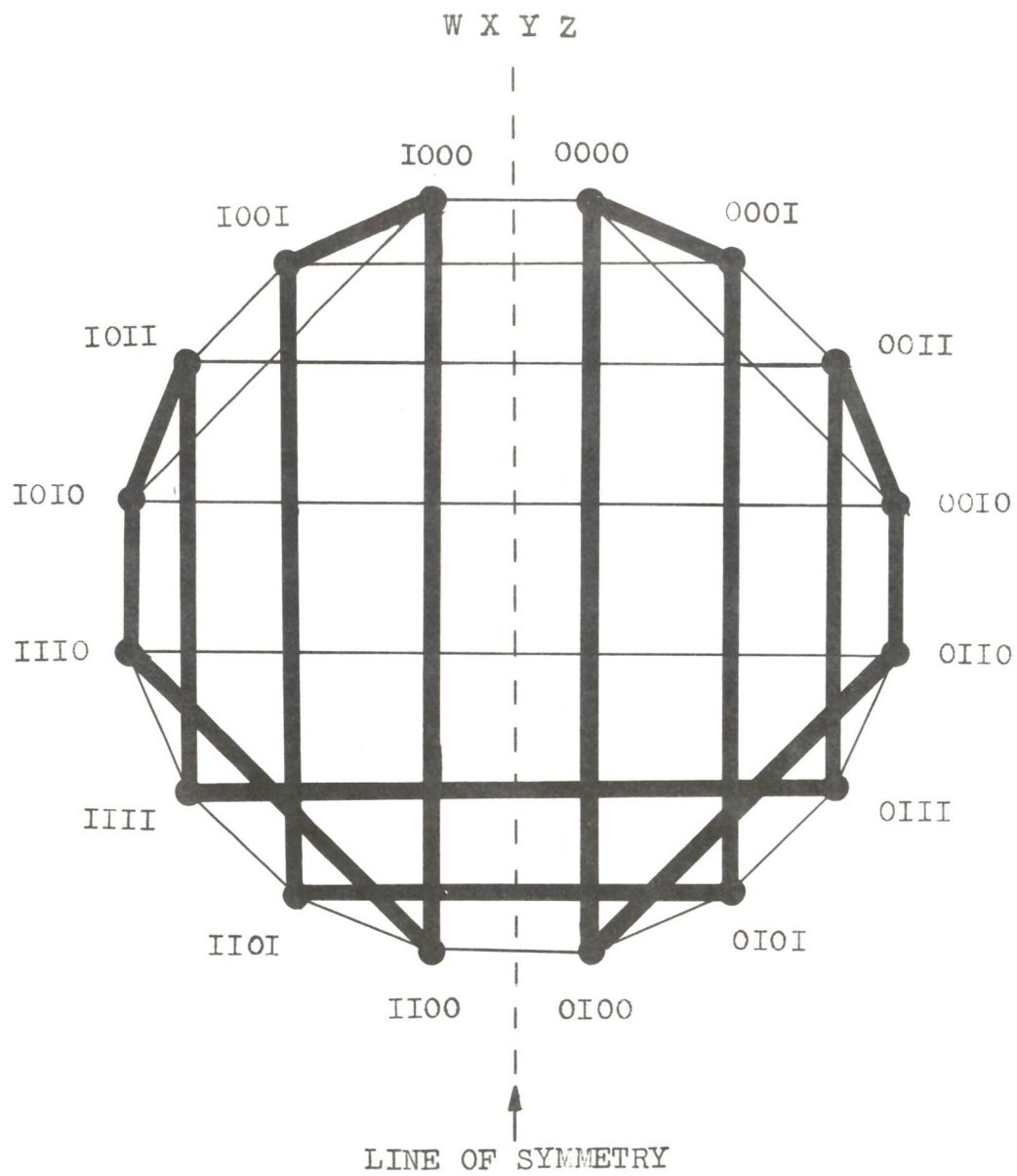
Fig. III.2

PSEUDO-GRAY CODES: The synthesis of a pseudo-gray code is trivial since, by definition it is only a particular combination of complements and permutations of an n bit Gray code. The ability to recognize that a monostrophic code is a pseudo-gray code becomes important when one

Fig. III.3



Any path through the path diagram which does not retrace itself or does not go through the same point twice is a distinct monostrophic code. A path closure, naturally, indicates a monostrophic closure. Fig. III.3 is an example of a 5 bit path diagram.



PSEUDO- GRAY	Z	I I 0 0 0 0 I I I I 0 0 0 0 I I
	Y	0 0 0 0 I I I I I I I I 0 0 0 0
	X	I 0 0 I I 0 0 I I 0 0 I I 0 0 I
	W	0 0 0 0 0 0 0 0 I I I I I I I I
NORMALIZED GRAY	X	0 I I 0 0 I I 0 0 I I 0 0 I I 0
	Z	0 0 I I I I 0 0 0 0 I I I I 0 0
	Y	0 0 0 0 I I I I I I I I 0 0 0 0
	W	0 0 0 0 0 0 0 0 I I I I I I I I

Fig. III.4

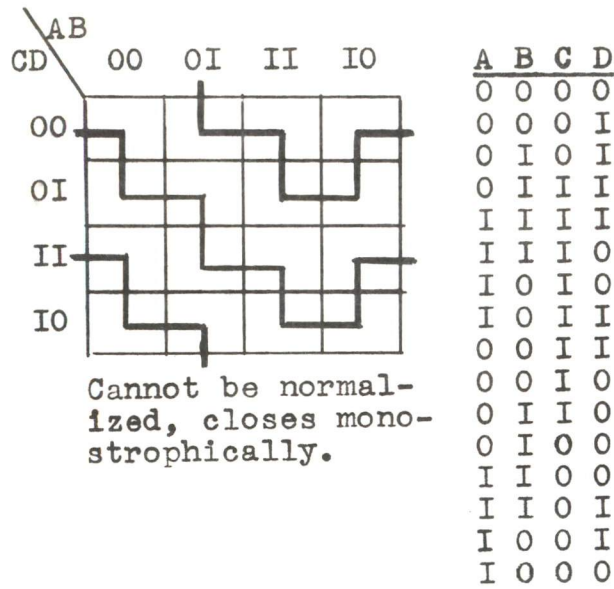
considers counting and conversions which are to be discussed in following chapters.

Nevertheless, a look at the Karnaugh Maps in Appendix I will show that, for a 4 bit code which is Gray or pseudo-Gray there exists a symmetry which may be described as the presence of mirror images (ignoring direction arrows) in all maps about either the vertical or horizontal center lines of the maps. A similar symmetry will be present in a Gray code of any number of bits.

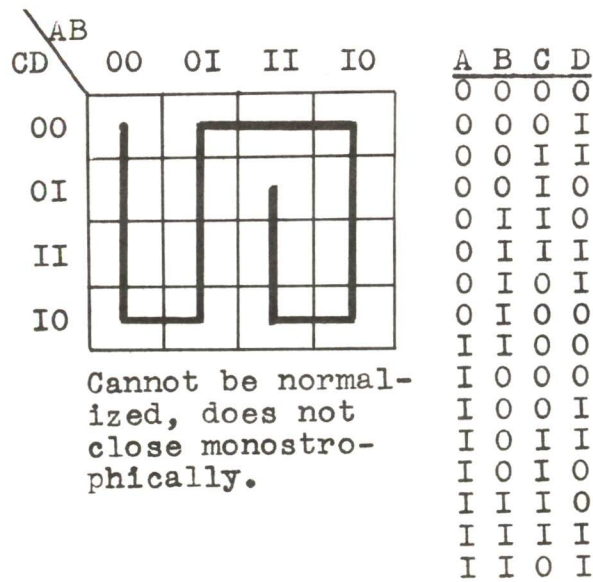
On a path diagram a Gray or pseudo-Gray code will result in a symmetrical pattern about a line dividing the circle in half. Fig.III.4 demonstrates this for a 4 bit pseudo-Gray code.

OTHER FULL COUNT MONOSTROPHIC CODES: There are many paths through a Karnaugh Map or path diagram which use all 2^n combinations available in an n bit code that cannot be normalized to a Gray code or do not close monostrophically. Fig.III.5 shows examples of such 4 bit codes. These codes have no general direct application, but are mentioned only to insure knowledge of their presence. A possible employment of such a code is in cryptographic systems. Also, they may be encountered when one considers the sequence of only certain bits in a code as we will do further along.

FORESHORTENED COUNT MONOSTROPHIC CODES: The most common use of n bit foreshortened counting sequences, both monostrophic and polystrophic, is to represent numbering



a.



b.

Fig. III.5

systems of radices of less than 2^n . For example, it takes ten of the sixteen possible combinations of a 4 bit code to represent the ten digits of a decimal numbering system.

As previously mentioned, one may take advantage of the unused combinations to simplify decoding and conversion, or they may be used as error detection and/or correction. Other uses of the unused combinations are possible such as reduction of power supply current and/or regulation requirements. Discussion in subsequent chapters will make more clear advantageous use of unused combinations in foreshortened counts.

Synthesis of foreshortened count monostrophic codes is easily accomplished by the use of Karnaugh Maps and path diagrams just as in all full count monostrophic codes. The observations made for full count codes hold true for foreshortened counts with the following exceptions:

- a) All 2^n combinations are not used.
- b) Monostrophic closure is not possible if the number of counts is odd.

REFLECTED MONOSTROPHIC CODES: In general, a reflected full or foreshortened monostrophic code must be synthesized as previously discussed, observing the symmetry rule on Karnaugh Maps or path diagrams if used.

If the bit upon which reflectivity depends (usually the most significant bit) will not change except between

the lower and upper halves of the counting sequence and upon closure (0 for the first half and 1 for the last half of counting sequence, or vice versa), synthesis of the reflected monostrophic code can be simplified. Only the bits exclusive of the one upon which reflectivity depends must be considered. The counting sequence of these bits must have one-half the counts of the total counting sequence, be monostrophic, but does not have to close monostrophically. In effect, the counting sequence of these bits counts up during the first half of the sequence and down during the last half of the sequence, retracing its path to "zero".

CHAPTER IV. MONOSTROPHIC-POLYSTROPHIC CODE CONVERSION

Monostrophic codes are unweighted codes and therefore, are difficult to manipulate arithmetically. Consequently, conversions between monostrophic and polystrophic codes, the subject of this chapter, are often required.

The use of hardware-oriented examples interspersed with academic discussion will be the general approach of this chapter. The examples will show electronic logic and switching logic, both employing commonly accepted symbols which are explained in Appendix II.

Ring sum or Exclusive Or functions appear repeatedly in this chapter. Appendix III contains a discussion of this function.

PARALLEL CONVERSION OF GRAY TO BINOMIAL BINARY: The relationship between these two codes may best be described as follows: The most significant bits of both codes are equal, and the lesser significant bits of the binomial binary code are equal to the ring sum (Exclusive Or) of the corresponding Gray code bit and all the more significant Gray code bits. Fig. IV.1 diagrammatically shows this.

An unvigorous method of proving the relationship shown in Fig. IV.1 is to solve the relationship for a 4 bit Gray to a 4 bit binomial binary code conversion. This is done in Fig. IV.2. Once this is accomplished it can be

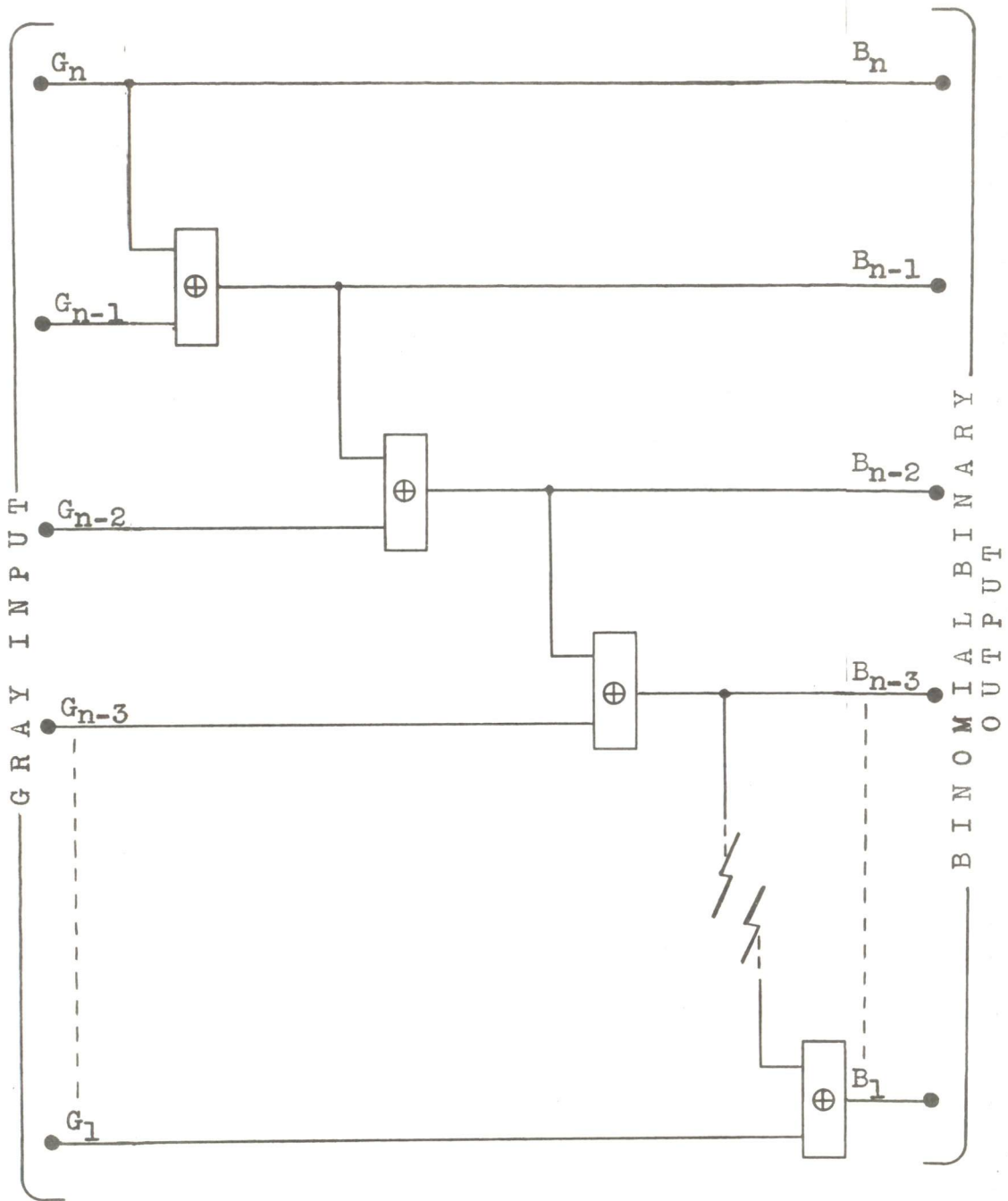


Fig. IV.1

GRAY				BINOMIAL BINARY			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

W=A, by inspection

		AB			
CD		00	01	11	10
00			1		1
01			1		1
11			1		1
10			1		1

$$X = A\bar{B} + \bar{A}B$$

$$= A \oplus B$$

		AB			
CD		00	01	11	10
00			1		1
01			1		1
11		1		1	
10		1		1	

$$Y = (\bar{A}\bar{B} + AB)C + (A\bar{B} + \bar{A}B)\bar{C}$$

$$= A \oplus B \oplus C$$

		AB			
CD		00	01	11	10
00			1		1
01		1		1	
11			1		1
10		1		1	

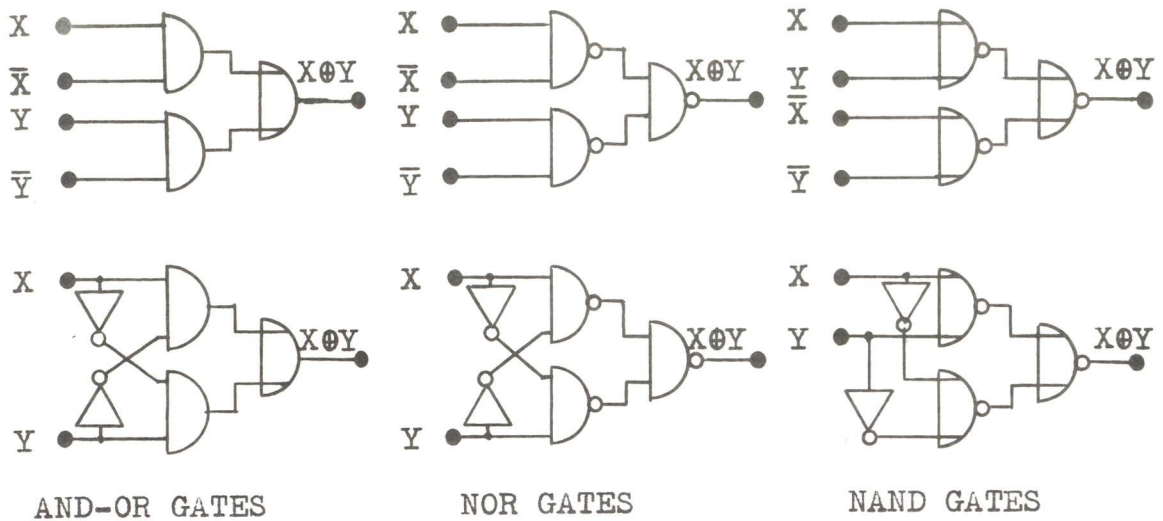
$$Z = S_{1,3}(A, B, C, D)$$

$$= A \oplus B \oplus C \oplus D$$

Fig. IV.2

intuitively seen that the relationship holds for n bits.

To implement Fig. IV.1 with electronic logic each ring sum symbol can be replaced by an Exclusive-Or module, or a combination of And-Or, Nand or Nor gates as shown in Fig. IV.3.

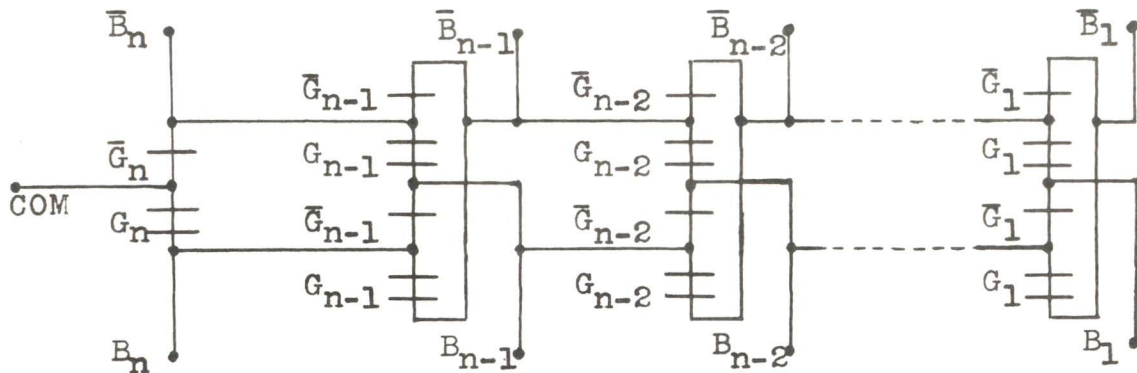


EXCLUSIVE OR LOGIC

Fig. IV.3

Switching logic can be used to convert Gray to binomial binary with one transfer set associated with the most significant Gray bit and two transfer sets for each lesser significant Gray bit. Fig. IV.4 demonstrates this conversion.

Remembering that the ring sum of a number of variables is the symmetric function S_{odd} of the variables, and the complement of the ring sum of a number of variables is the symmetric function S_{even} of the variables, it would seem that a symmetric switching circuit with appropriate pick-off points might furnish all the logic required for a Gray to binomial binary conversion. Fig. IV.5 shows a folded symmetric switching circuit which functions as a Gray to binomial binary converter, and is the same circuit shown in Fig. IV.4, laid out differently to clearly demonstrate the symmetric switching circuit relationship.



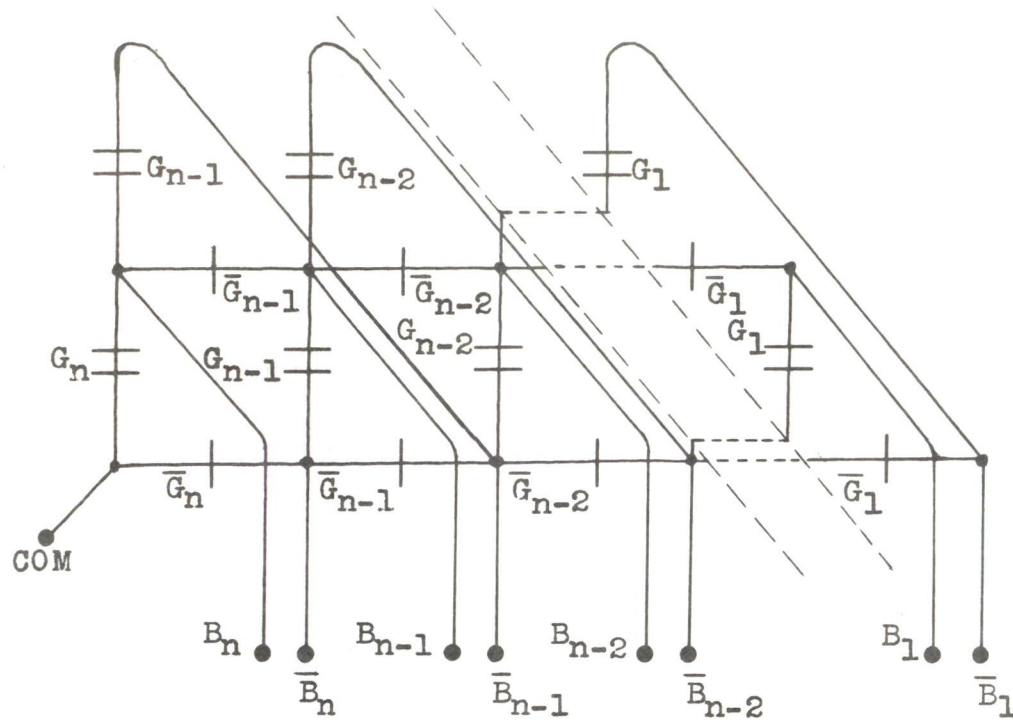
GRAY TO BINOMIAL BINARY

Fig. IV.4

PARALLEL CONVERSION OF PSEUDO-GRAY TO BINARY: By use of the methods of conversion described for Gray to binomial binary codes and by effecting normalization of the input variables, pseudo Gray codes may easily be converted to binomial binary.

PARALLEL CONVERSION OF OTHER FULL COUNT MONOSTROPHIC CODES: Generally, requirements for parallel conversion of other than Gray or pseudo-Gray codes to weighted codes are not encountered. If the requirement does arise, "brute force" techniques can be employed to accomplish such conversion, but the complexity of such conversions will be much more than for the Gray and pseudo Gray to binomial binary.

PARALLEL CONVERSION OF FORESHORTENED MONOSTROPHIC CODES: There are many reasons for the use of foreshortened codes, the most obvious of which is the encoding of a numbering system whose radix is not an integer power of two. Other



GRAY TO BINOMIAL BINARY
FOLDED SYMMETRIC CIRCUIT

Fig. IV.5

reasons are ability for error detection and/or correction, ease of conversion to a weighted code, etc..

In selecting a foreshortened monostrophic code more than the minimum number of bits may be used to encode a numbering system in order to satisfy all the requirements imposed on the coding system. In general, the higher the number of bits used to encode a numbering system (above the minimum required), the easier it is to convert (or decode) and more difficult to transmit.

The monostrophic code used to represent a numbering system which must be converted to another binary code is

usually selected so that conversion is simplified. For example, a monostrophic coding of a decimal numbering system that must be converted to the self-complementing excess-3 code (XS3) would be simplest to convert if counts 3 through 12 of the Gray code were used to represent 0 through 9 of the excess-3 decimal code. Such a selection would result in a code being reflective, that closes monostrophically, and following the Gray to binomial binary conversion rules. Fig. IV.6 shows this. The same philosophy applied to the NBC Decimal system would result

DECIMAL	MONO- STROPHIC	XS3
0	0010	0011
1	0110	0100
2	0111	0101
3	0101	0110
4	0100	0111
5	1100	1000
6	1101	1001
7	1111	1010
8	1110	1011
9	1010	1100

Conversion to XS3 follows same rules as Gray to binomial binary.

Fig. IV.6

in counts 0-9 of the Gray code to represent 0-9 of the NBC Decimal system simplifying conversion, but the monostrophic code lacks reflectivity and monostrophic closure, thereby being a poor choice.

Fig. IV.7 shows a monostrophic coding of a 2421 binary coded decimal numbering system. Again, the monostrophic code is reflective and closes monostrophically, and follows the simple Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	BINARY 2421	
0	0000	0000	
1	0001	0001	Conversion to 2421
2	0011	0010	BC Decimal follows
3	0010	0011	same rules as Gray
4	0110	0100	to binomial binary.
5	1110	1011	
6	1010	1100	
7	1011	1101	
8	1001	1110	
9	1000	1111	

Fig. IV.7

Occasionally the duo-decimal (Radix¹²) numbering system rears its ugly head in such applications of distance measuring systems and monetary systems. A self-complementing 4 bit code that could be used to represent such a code is the excess-2 (XS2) code. Fig. IV.8 shows the XS2 code and a monostrophic equivalent which is reflective, closes monostrophically and follows the Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	XS2	
0	0011	0010	
1	0010	0011	
2	0110	0100	
3	0111	0101	Conversion to XS2
4	0101	0110	follows same rules
5	0100	0111	as Gray to binomial
6	1100	1000	binary.
7	1101	1001	
8	1111	1010	
9	1110	1011	
10	1010	1100	
11	1011	1101	

Fig. IV.8

A weighted 4 bit code for a binary coded duo-decimal numbering system is the 4421 code. Fig. IV.9 shows this

code with a monostrophic equivalent which is reflective, closes monostrophically, and follows the Gray to binomial binary conversion rules.

DECIMAL	MONO-STROPHIC	4421	
0	0000	0000	
1	0001	0001	
2	0011	0010	
3	0010	0011	
4	0110	0100	Conversion to 4421
5	0111	0101	binary coded duo-
6	1111	1010	decimal follows same
7	1110	1011	rules as Gray to
8	1010	1000	binomial binary.
9	1011	1001	
10	1001	1110	
11	1000	1111	

Fig. IV.9

DATEX CODE:* A foreshortened monostrophic code that is frequently encountered is the Datex code shown in Fig. IV.10. It is a 10 count, 4 bit monostrophic code which is reflective and closes monostrophically.

DECIMAL	DATEX
0	0001
1	0011
2	0010
3	0110
4	0100
5	1100
6	1110
7	1010
8	1011
9	1001

Fig. IV.10

Notice that the 0000 and 1111 combinations are not used enabling partial error detection and reducing power supply regulation and power requirements.

* Datex Corp., 1307 S. Myrtle Ave., Monrovia, Calif.

If more than one decade is employed in the counting sequence, the decimal counting sequence is also monostrophic in nature, i.e., only one decimal digit changes at a time, resulting in a decimal counting sequence as follows: 0 through 9, 19 through 10, 20 through 29, 39 through 30, etc. Therefore when converting to any decimal code, starting with the next to the most significant digit and working to the least significant digit the nine's complement must be sensed if the next more significant digit is odd. The most significant digit is always assumed to have a zero (which is even) preceding it so it never needs to be nine's complemented.

In order to convert to a decimal code we can, in effect, say that the Datex code is a 5 bit code as shown in Fig. IV.11, the E_{in} bit indicating whether the next more significant digit is odd or even (1 = odd, 0 = even).

DATEX ABCD	DECIMAL	
	$E_{in}=0$	$E_{in}=1$
0001	0	9
0011	1	8
0010	2	7
0110	3	6
0100	4	5
1100	5	4
1110	6	3
1010	7	2
1011	8	1
1001	9	0

Fig.IV.11

The easiest method of conversion (or decoding) to another binary coded decimal system is to use conversion

(or decoding) logic for Fig. IV.10, and use the E_{in} bit to reverse the sense of the A bit by the conversion (or decoding) logic as shown in Fig. IV.12. This technique can be used for conversion (or decoding) any multi-decade reflective coding system whose coded numbering system is similar in nature to the Datex code. The

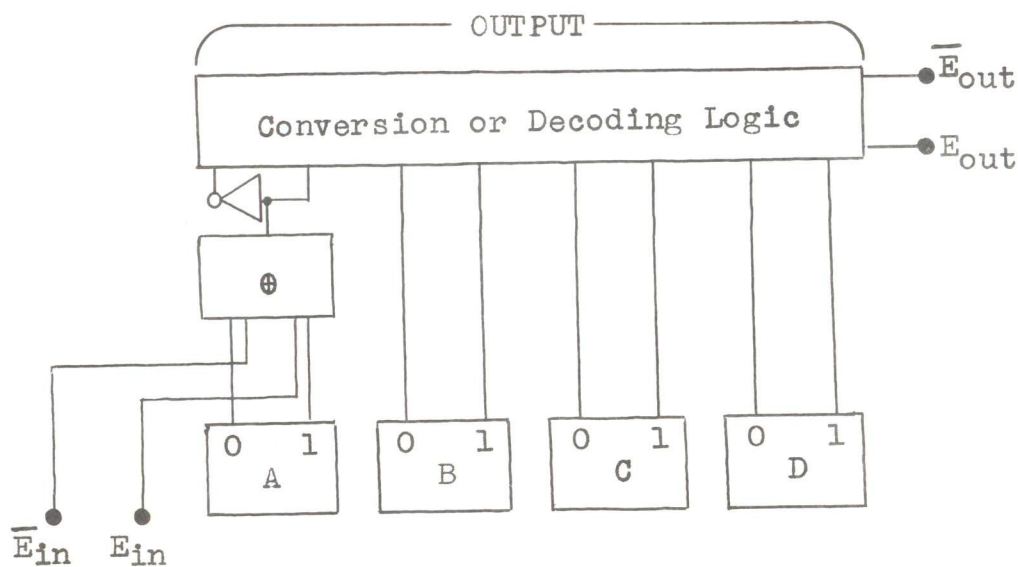


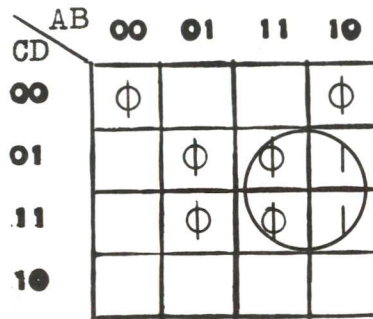
Fig. IV.12

logic required for the conversion technique in Fig. IV.12 is shown in Fig. IV.13 for the 8421, 2421 and XS3 binary coded decimal systems.

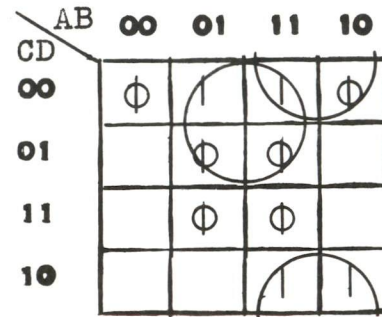
SWITCH-TAIL RING COUNTER CODE: Another foreshortened monostrophic code often employed is a code easily generated in a switch-tail ring counter (shift register) described in the next chapter. The number of counts in such an n bit code is 2^n counts, and decoding each of the

DECIMAL	DATEX				8 4 2 1				2 4 2 1				X S 3			
	A	B	C	D	H	I	J	K	L	M	N	O	P	Q	R	S
0	0	0	0	I	0	0	0	0	0	0	0	0	0	0	I	I
1	0	0	I	I	0	0	0	I	0	0	0	I	0	I	0	0
2	0	0	I	0	0	0	I	0	0	0	I	0	0	I	0	I
3	0	I	I	0	0	0	I	I	0	0	I	I	0	I	I	0
4	0	I	0	0	0	I	0	0	0	I	0	0	0	I	I	I
5	I	I	0	0	0	I	0	I	I	I	0	I	I	0	0	0
6	I	I	I	0	0	I	I	0	I	I	0	0	I	0	0	I
7	I	0	I	0	0	I	I	I	I	I	0	I	I	0	I	0
8	I	0	I	I	I	0	0	0	I	I	I	0	I	0	I	I
9	I	0	0	I	I	0	0	I	I	I	I	I	I	I	0	0

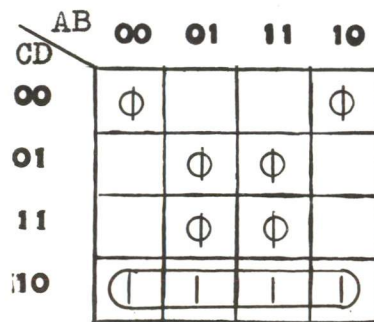
8 4 2 1



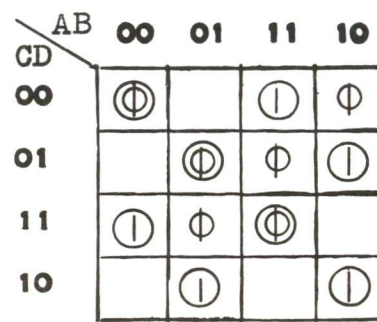
$H=AD$



$I=A\bar{D}+B\bar{C}$



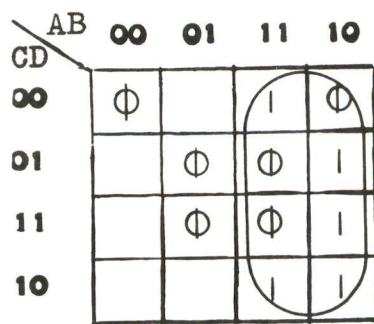
$J=C\bar{D}$



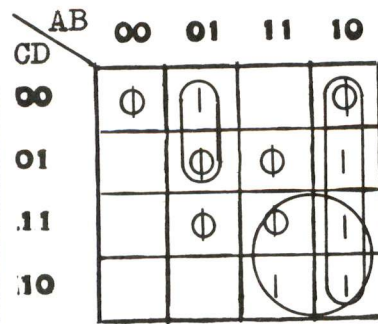
$K=E_{out}=(A\oplus B\oplus C\oplus D)$

Fig. IV.13
(Cont'd next page)

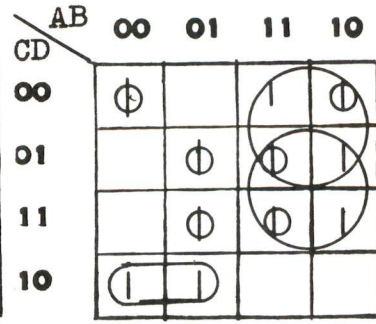
2 4 2 1



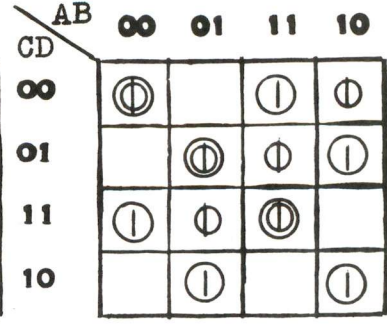
$L=A$



$M=A(\bar{B}+C)+\bar{A}B\bar{C}$

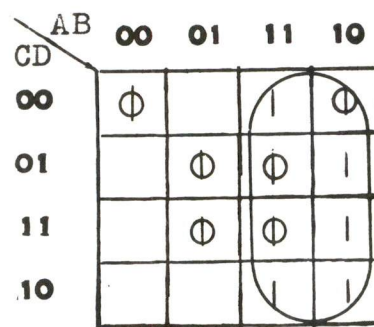


$N=A(\bar{C}+D)+\bar{A}C\bar{D}$

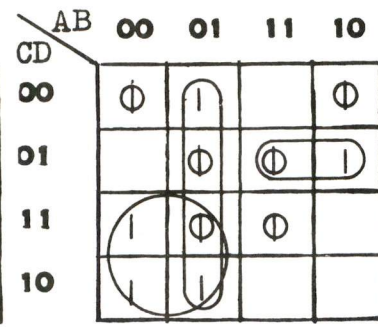


$O=\bar{E}_{out}=(A\oplus B\oplus C\oplus D)$

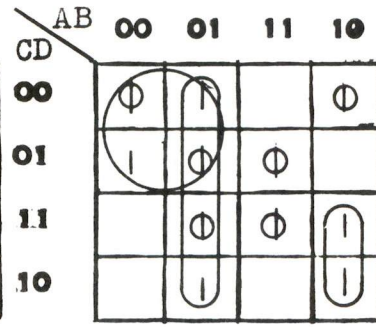
X S 3



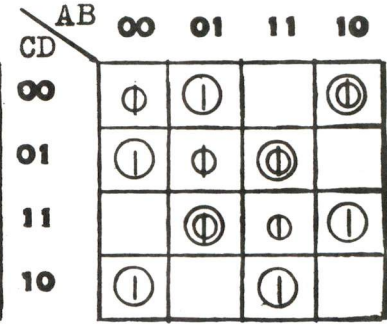
$P=A$



$Q=\bar{A}(B+C)$



$R=\bar{A}(B+\bar{C})$



$S=\bar{E}_{out}=(A\oplus B\oplus C\oplus D)$

Fig. IV.13
(Cont'd from previous page)

2^n combinations to one of 2^n lines requires sensing of only two of the n bits. Fig. IV.14 demonstrates the switch-tail code for a 5 bit unweighted decimal code.

SWITCH-TAIL					DECIMAL	REQ'D LOGIC
A	B	C	D	E		
0	0	0	0	0	0	$\overline{A}E$
0	0	0	0	I	1	$\overline{D}E$
0	0	0	I	I	2	$\overline{C}D$
0	0	I	I	I	3	$\overline{B}C$
0	I	I	I	I	4	$\overline{A}B$
I	I	I	I	I	5	$A\overline{E}$
I	I	I	I	0	6	$D\overline{E}$
I	I	I	0	0	7	$\overline{C}D$
I	I	0	0	0	8	$\overline{B}C$
I	0	0	0	0	9	$\overline{A}B$

Fig. IV.14

Notice that the bits sensed for each count are the two adjacent bits that differ except for all 0's and all I's in which case the two end bits are sensed. This holds true for any number of bits.

IN-REGISTER PARALLEL CONVERSION: Quite often it is desired to convert in parallel the contents of a register containing a monostrophic code to a polystrophic code, with the result of the conversion placed in the same register. This can be done by loading the register, through the necessary conversion logic, into the same register.

The logic required can be the same conversion logic as described earlier in this chapter, but in most cases it may be simplified. This is true because the state of

each bit only needs correction, not generation. Fig. IV.15 demonstrates the correction approach to a 4 bit Gray to binomial binary in-register parallel conversion by complementing the "incorrect" bits. To expand to n bits, additional similar stages can be added on the least significant end of the 4 bit register shown in Fig. IV.15.

If the flipflops of the register have no complement inputs (or if pulsing both inputs simultaneously does not complement the flipflops) they must be appropriately set or reset to correct the "incorrect" bits. Fig. IV.16 demonstrates this approach for the same code conversion as shown in Fig. IV.15. Again FFA requires no correction as $A_1=W$.

SERIAL CONVERSION OF GRAY TO BINOMIAL BINARY: If a Gray coded number is being transmitted serially (bit by bit) with the most significant bit leading, serial conversion to binomial binary is possible upon receipt of each bit. The conversion is accomplished using the two rules:

1) The most significant Gray bit is identical to the most significant binomial binary bit.

2) If a bit is a 1 after conversion the bit following it is complemented.

Fig. IV.17 demonstrates the conversion.

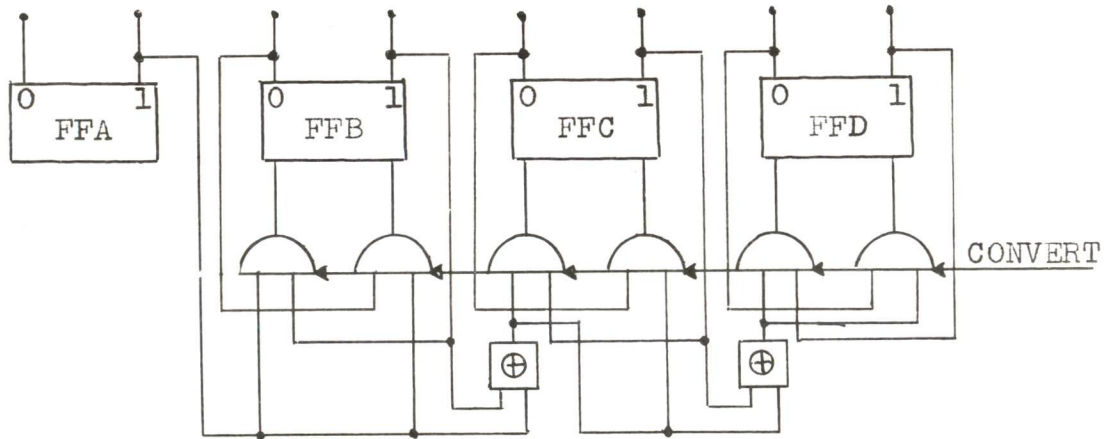
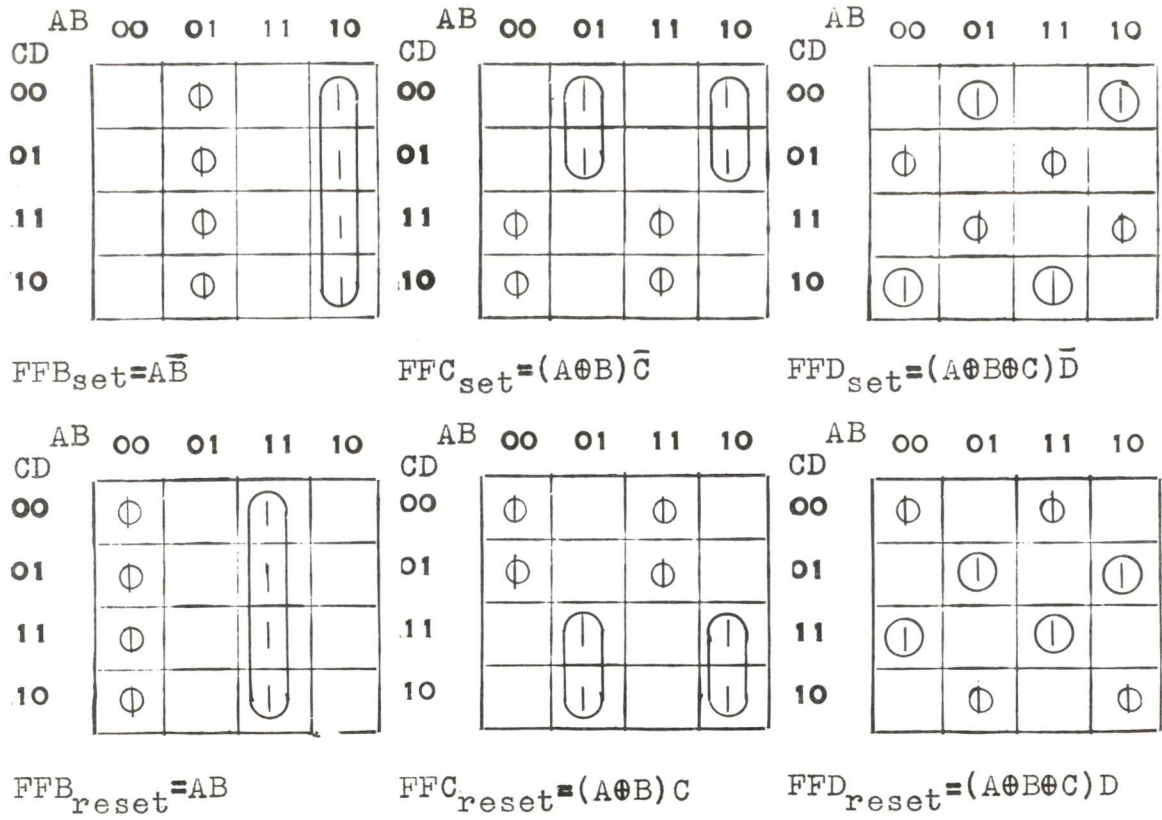
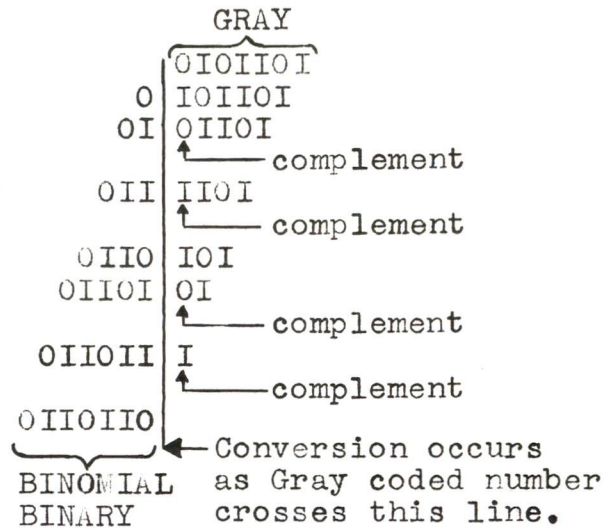


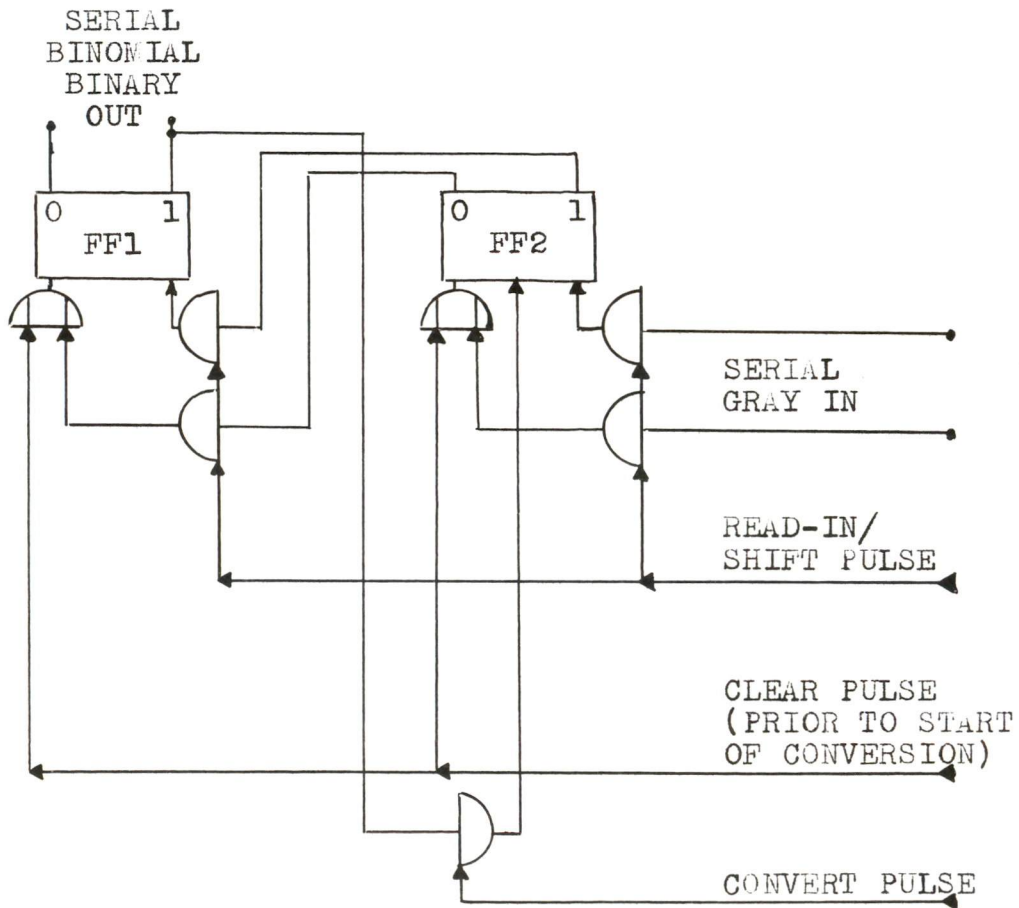
Fig. IV.16

Direction of shift is to the left with the most significant bit leading. When bit just to left of vertical line is 1 the bit just to the right of the vertical line is complemented.



SERIAL GRAY TO BINOMIAL BINARY CONVERSION

Fig. IV.17



SERIAL GRAY TO BINOMIAL BINARY CONVERTER

Fig. IV.18

the input but before the convert pulse. (The convert pulse can be delayed until slightly before the next input bit to allow more sampling time if output is taken from the output of the Exclusive Or gate.)

SERIAL CONVERSION OF OTHER MONOSTROPHIC CODES:

Serial, bit by bit, conversion of codes other than those that follow the rules for Gray to binomial binary conversion is usually not possible in the true sense of the word, serial. If such codes are received for conversion in serial form, they are usually shifted into a register and converted in parallel when the whole binary coded number is in the register using parallel conversion techniques.

POLYSTROPHIC TO MONOSTROPHIC CODE CONVERSION: In

order for the discussion of conversion to be considered nearly complete, conversion from polystrophic to monostrophic codes must be considered. If the reverse process of each conversion previously mentioned was discussed in as much detail, considerable redundancy of thought would be involved. Therefore, the discussion of this topic will be kept brief.

Any conversion which, when going from a monostrophic to a polystrophic code, follows the Gray to binomial binary conversion rules, can be accomplished in the reverse direction by the following rule: The monostrophic

code equivalent of a binary coded number is the bit-by-bit Exclusive Or of the number with itself shifted one bit (to the right or left) ignoring the least significant bit of the result (See Fig. IV.20). It is obvious

```

1001101 ←# to be converted to monostrophic equiv.
 1001101 ←# shifted one bit
1101011 ← Bit by bit Exclusive Or
           ← LSB, ignore
1101011 ← Result, monostrophic equiv.

```

Fig. IV.20

that the most significant bit is the same in either code, and the lesser significant bits of the monostrophic equivalent are the Exclusive Ors of adjacent bits of the binary coded number. Parallel conversion, therefore, can be accomplished as shown in Fig. IV.21.

Serial conversion, following the same rules requires the number to arrive for conversion in serial form with the most significant bit leading. The previous bit (unconverted) must be stored and Exclusive Or'd with the arriving bit (a zero is assumed to be stored in the one bit memory prior to arrival of the most significant bit). The output of the Exclusive Or is the serial converted output. Fig. IV.22 shows a serial Polystrophic-Monostrophic converter with a flip-flop to store the output (could be entrance bit to a shift register). The storage flip-flop must have a clear input shown, and it must be cleared (set to 0) prior to start of conversion.

Conversion between codes that do not have the

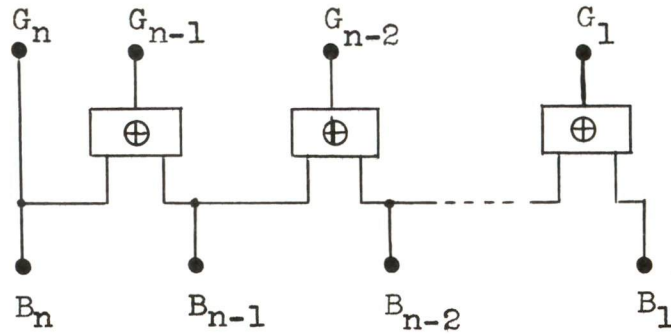


Fig. IV.21

Gray-binomial binary relationship is usually not required but, if encountered, must be accomplished by brute force parallel conversion techniques.

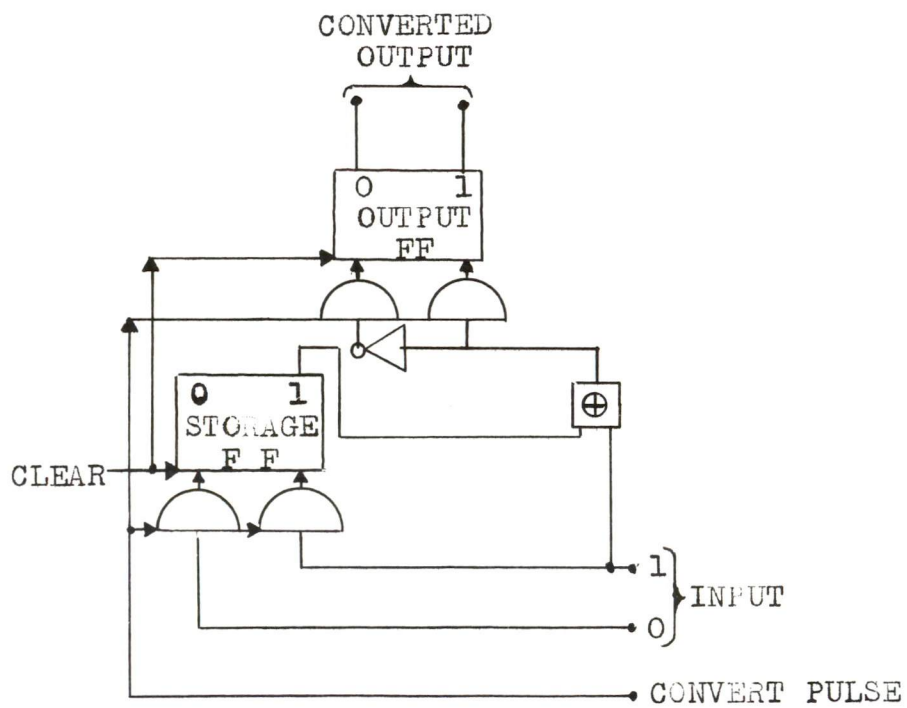


Fig. IV.22

CHAPTER V. GENERATION OF MONOSTROPHIC CODES

The most developed method of monostrophic code generation is the use of electro-mechanical and electro-optical-mechanical sensing of segmented, multiple track, rotatable discs in shaft angle encoding systems. The advantage of monostrophic codes in such applications is the ability to sample the readout "on the fly" without fear of ambiguity. Since this method is well known and already well documented this author will concentrate on generation of monostrophic codes by other than the disc technique, i.e., monostrophic code counters by electronic logic and sequential switching techniques.

PHILOSOPHY OF PULSE COUNTING DIRECTLY IN A MONOSTROPHIC CODE: Because only one bit changes between adjacent counts in any monostrophic code, one cannot take advantage of carries as in, for instance, an electronic binomial binary counter when counting in a monostrophic code. Nor can the state of the next less significant bit be depended upon to furnish the proper levels to the complement input gate of a bit as in an electronic binomial binary counter. Therefore, the state of the complete binary number must be employed to furnish, through logic, gating levels to the complement input pulse gate (or set and reset pulse gates) of the storage element of each bit (bistable flip-flops). This approach is analogous to the

anticipated carry approach used with other binary counters to enable faster counting. Also, it is the approach that must usually be taken for any type of sequential switching (relay) counters. Fig. V.1 demonstrates this approach for a 3 bit Gray code counter.

N BIT GRAY CODE COUNTER: The previous section described a 3 bit Gray code counter. In this section an n bit Gray code counter and required counting logic will be demonstrated. A method for easily reversing the counting sequence will also be shown. The importance of the Exclusive Or function will again appear.

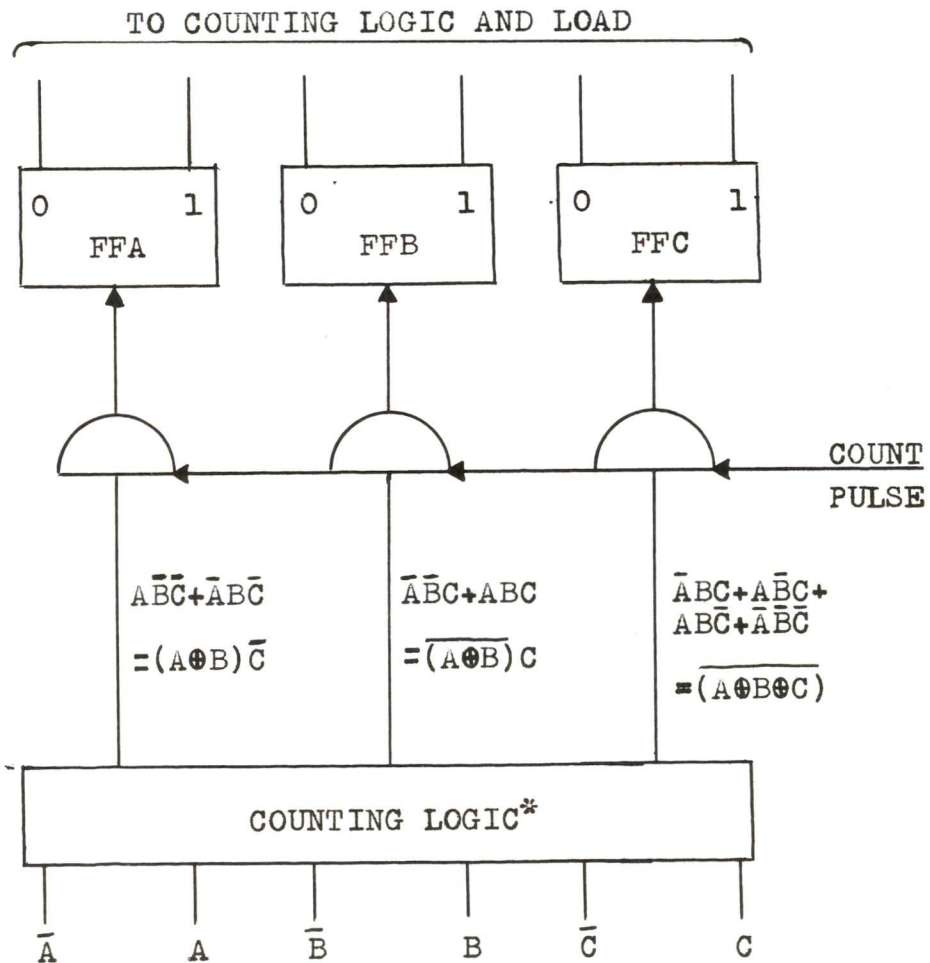
If the method for determining the counting logic described in the previous section is used to determine the counting logic for a 2, 3, 4 and 5 bit Gray code counter, the information shown in Fig. V.2 is obtained after simplification. Notice that the state of every bit influences the logic levels associated with the complement pulse gate of each bit as previously stated at the beginning of this chapter.

Another set of relationships evident in Fig. V.2 and explained below can be used to simplify the determination of the logical expressions required for each bit of an n bit Gray code counter.

1. The least significant bit's term for an up counter is the complement of the ring sum of all bits.

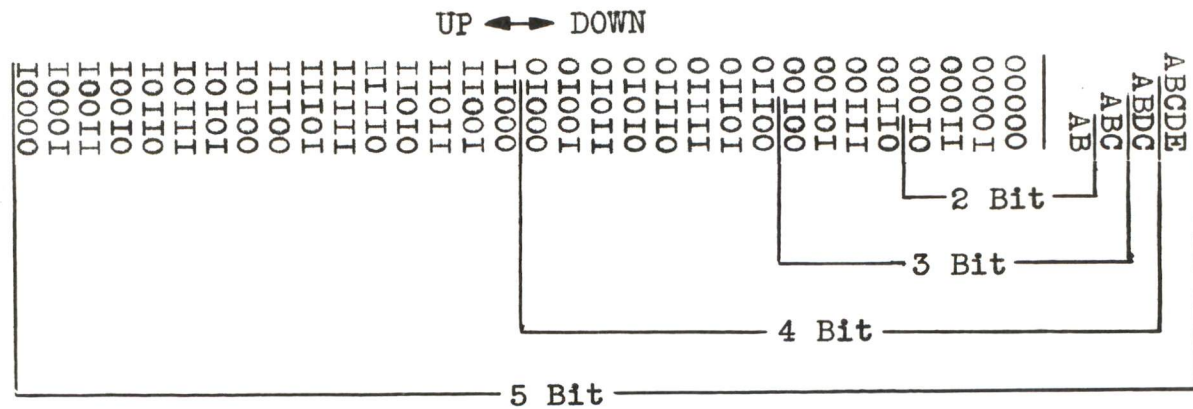
COUNT SEQ A B C	PREVIOUS COUNT						BIT OF PREVIOUS COUNT COMPLEMENTED	
	UP			DOWN			UP	DOWN
A B C	A B C	A B C	A B C	A B C	A B C			
0 0 0	1 0 0	0 0 1	0 0 1	0 0 1	0 0 1	A	C	
0 0 1	0 0 0	0 1 1	0 1 1	0 1 1	0 1 1	C	B	
0 1 1	0 0 1	0 1 0	0 1 0	0 1 0	0 1 0	B	C	
0 1 0	0 1 1	1 1 0	1 1 0	1 1 0	1 1 0	C	A	
1 1 0	0 1 0	1 1 1	1 1 1	1 1 1	1 1 1	A	C	
1 1 1	1 1 0	1 0 1	1 0 1	1 0 1	1 0 1	C	B	
1 0 1	1 1 1	1 0 0	1 0 0	1 0 0	1 0 0	B	C	
1 0 0	1 0 1	0 0 0	0 0 0	0 0 0	0 0 0	C	A	

DOWN
↑
↓
UP



*Output of counting logic shown for Up Counter

Fig. V.1



	DIRECTION OF COUNT	A	B	C	D	E
5 Bit	UP	$(A \oplus B) \bar{C} \bar{D} \bar{E}$	$(\bar{A} \oplus \bar{B}) \bar{C} \bar{D} \bar{E}$	$(\bar{A} \oplus B \oplus C) \bar{D} \bar{E}$	$(\bar{A} \oplus B \oplus C \oplus D) \bar{E}$	$(\bar{A} \oplus B \oplus C \oplus D \oplus E)$
	DOWN	$(\bar{A} \oplus B) \bar{C} \bar{D} \bar{E}$	$(A \oplus B) \bar{C} \bar{D} \bar{E}$	$(A \oplus B \oplus C) \bar{D} \bar{E}$	$(A \oplus B \oplus C \oplus D) \bar{E}$	$(A \oplus B \oplus C \oplus D \oplus E)$
4 Bit	UP	$(A \oplus B) \bar{C} \bar{D}$	$(\bar{A} \oplus \bar{B}) \bar{C} \bar{D}$	$(\bar{A} \oplus B \oplus C) \bar{D}$	$(\bar{A} \oplus B \oplus C \oplus D)$	
	DOWN	$(\bar{A} \oplus B) \bar{C} \bar{D}$	$(A \oplus B) \bar{C} \bar{D}$	$(A \oplus B \oplus C) \bar{D}$	$(A \oplus B \oplus C \oplus D)$	
3 Bit	UP	$(A \oplus B) \bar{C}$	$(\bar{A} \oplus \bar{B}) C$	$(\bar{A} \oplus B \oplus C)$		
	DOWN	$(\bar{A} \oplus B) \bar{C}$	$(A \oplus B) C$	$(A \oplus B \oplus C)$		
2 Bit	UP	$(A \oplus B)$	$(\bar{A} \oplus \bar{B})$			
	DOWN	$(\bar{A} \oplus B)$	$(A \oplus B)$			

Fig. V.2

2. The term for any bit of an up counter other than the most significant bit is the complement of the ring sum of all the terms excluding the lesser significant bits Anded with the next less significant bit and the And of the complement of all lesser significant bits.

3. The term for the most significant bit is ring sum of the most and next most significant bits Anded with the And of the complement of all lesser significant bits.

4. For an n bit down counter the logic terms are the same except all of the ring sum portion of the up counter logic terms are complemented.

From the above relationships a set of rules are self-evident for determining the complement pulse gate logic terms for all bits of an n bit Gray code counter. As an example, for a seven bit Gray code up counter in which the bits are labelled A,B,C,D,E,F, and G, A being the most significant bit and G the least significant bit, the seven terms are as follows:

$$\begin{aligned}
 A; & \quad (A \oplus B) \bar{C} \bar{D} \bar{E} \bar{F} \bar{G} \\
 B; & \quad \overline{(A \oplus B)} C \bar{D} \bar{E} \bar{F} \bar{G} \\
 C; & \quad \overline{(A \oplus B \oplus C)} \bar{D} \bar{E} \bar{F} \bar{G} \\
 D; & \quad \overline{(A \oplus B \oplus C \oplus D)} \bar{E} \bar{F} \bar{G} \\
 E; & \quad \overline{(A \oplus B \oplus C \oplus D \oplus E)} \bar{F} \bar{G} \\
 F; & \quad \overline{(A \oplus B \oplus C \oplus D \oplus E \oplus F)} G \\
 G; & \quad \overline{(A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G)}
 \end{aligned}$$

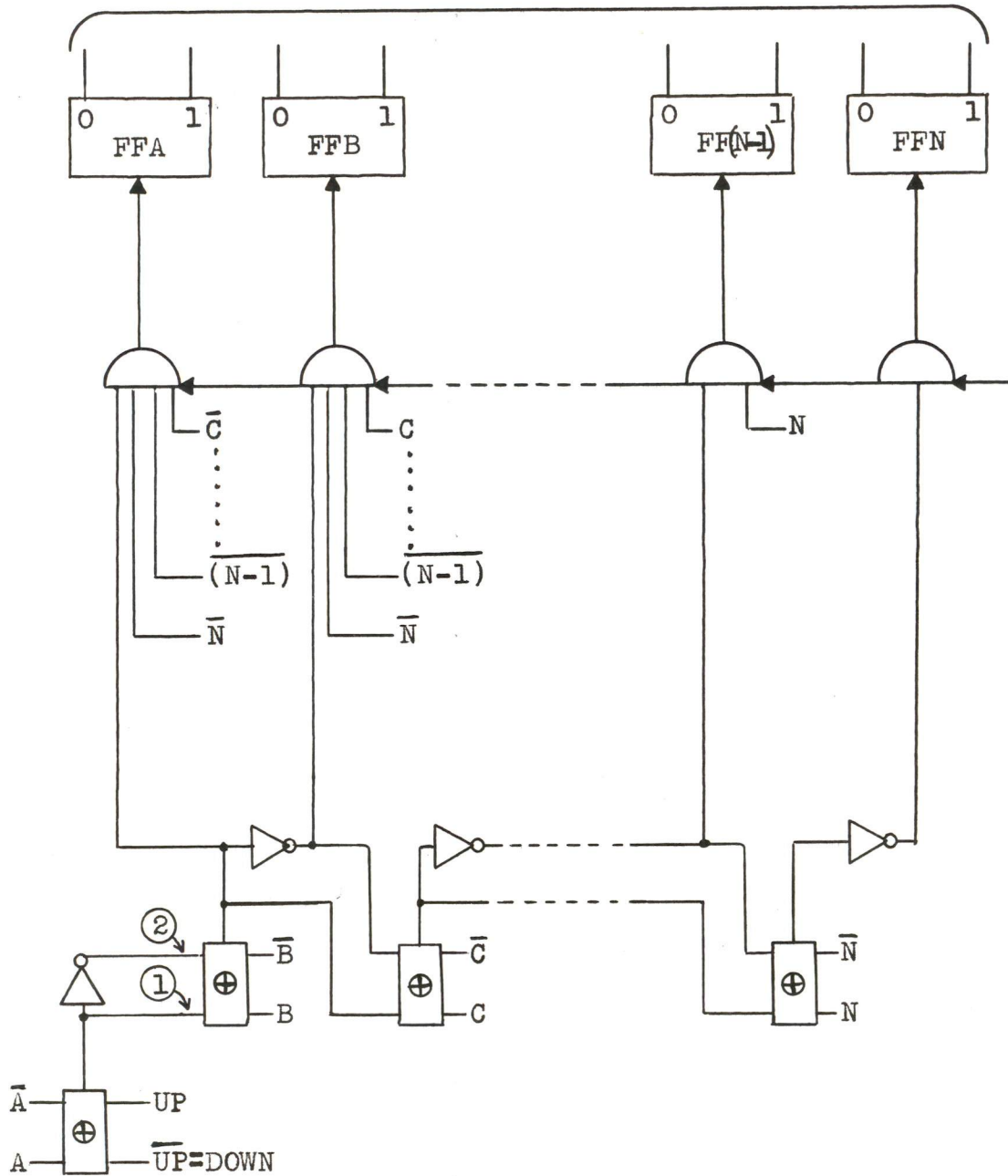
Complementing the ring sum portion of each term yields the logic terms for a down counter.

By complementing any bit in a ring sum term the ring sum term is complemented. This leads us to a simple method of reversing the count in an n bit Gray code counter resulting in an up-down counter. Notice that the most and next most significant bits appear in all ring sum portions of all terms. Simply by providing a means of reversing the sense of either (not both) of these bits at will, a method of reversing the direction of count is provided. Fig. V.3 shows an n bit up-down Gray code counter embodying all thoughts of this section.

N BIT PSEUDO-GRAY CODE COUNTER: Remembering that a pseudo-Gray code is only a combination of complemented and/or permuted bits it is easily seen that the Gray code counter discussion of the previous section applies if, in addition, the counting logic inputs are properly complemented and permuted (normalized) so that the proper counting sequence occurs. Therefore no more discussion of pseudo-Gray code counters will be pursued.

OTHER MONOSTROPHIC CODE COUNTERS: The method shown in Fig. V.1 for determining the logic terms required for the complement pulse gate enabling levels can be used for any full or foreshortened monostrophic (polystrophic for that matter) code counting sequence. For example, the 4 bit (4 bits per decade) Datex described in the last chapter is considered. See Fig. V.4.

TO LOAD AND COUNTING LOGIC

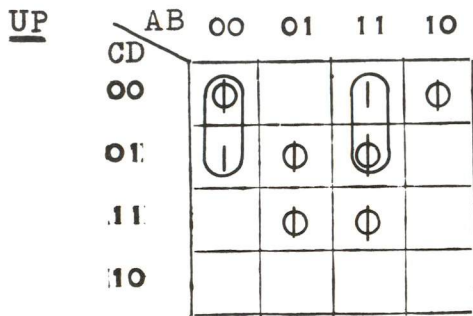


To count in only one direction eliminate \oplus circuit to which \bar{A} , A, UP and $\overline{UP=DOWN}$ are connected and connect A and \bar{A} as follows: A to (1) and \bar{A} to (2) for up counter, \bar{A} to (2) and A to (1) for down counter.

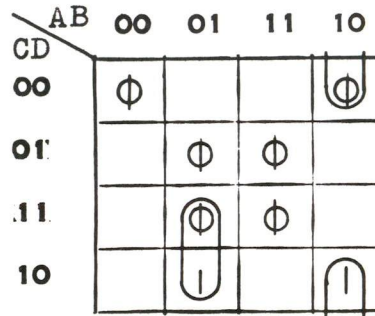
N BIT GRAY CODE COUNTER

Fig. V.3

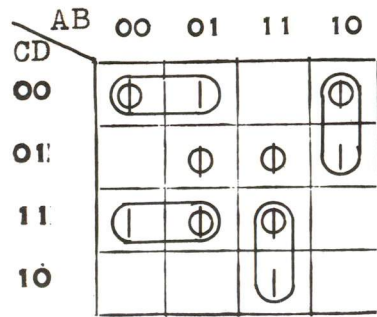
DATEX SEQ				PREVIOUS COUNT				BIT OF PREVIOUS COUNT COMPLEMENTED			
A	B	C	D	UP		DOWN		UP		DOWN	
A	B	C	D	A	B	C	D	A	B	C	D
0	0	0	1	1	0	0	1	0	0	1	1
0	0	1	1	0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1	0	1	1	0
0	1	1	0	0	0	1	0	0	1	0	0
0	1	0	0	0	1	1	0	1	1	0	0
1	1	0	0	0	1	0	0	1	1	1	0
1	1	1	0	1	1	0	0	1	0	1	0
1	0	1	0	1	1	1	0	1	0	1	1
1	0	1	1	1	0	1	0	1	0	0	1
1	0	0	1	1	0	1	1	0	0	0	1



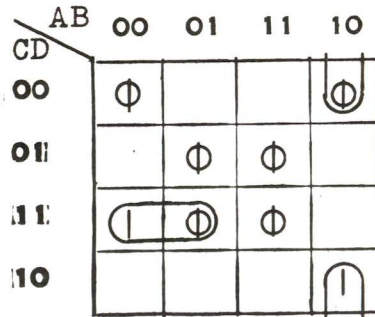
$$FFA = \overline{(A \oplus B)} \overline{C}$$



$$FFB = \overline{A}BC + A\overline{B}D$$



$$FFC = \overline{(C \oplus D)} \overline{A} + (B \oplus C) A$$



$$FFD = \overline{A}CD + A\overline{B}D$$

DOWN DateX is a reflected code with reflectivity dependent upon the A bit. Therefore, reversing sense of the A bit by counting logic reverses direction of count.

COUNTING LOGIC FOR ONE-DECADE DATEX COUNTER

Fig. V.4

Usually when the Datex code is used there is more than one decade in which case the Datex counting sequence of each decade other than the most significant decade does not pass directly from 0 to (2^n-1) and (2^n-1) to 0. Instead, each decade counts to the end of a sequence (zero or nine), the next more significant decade steps one count, then the first decade counts to the other end of its sequence (nine or zero). Any decade counts up when the next more significant decade is even, and down when the next more significant decade is odd. There is a similarity in this counting sequence with that of a Gray code, and the techniques employed to make use of this characteristic in Gray code counters (described in next section) can be used to control the direction of count in each decade of a Datex code counter.

CASCADING GRAY CODE COUNTERS: An examination of an n bit Gray code reveals that if the n bits are divided into two groups, say l bits and m bits where $l+m = n$, an interesting characteristic emerges. Look at the 4 bit code in Fig. V.5 that has been divided into two 2 bit groups. The group including C and D counts up, then down, then up and then down. The direction of count is dependent upon AB being even or odd; up when AB is even, down when AB is odd. This relationship holds for any n bit Gray code counter regardless where the division is. Also, if the

	A	B	C	D	
even	0	0	0	0	↓up
	0	0	0	1	
	0	0	1	1	
	0	0	1	0	
odd	0	1	1	0	↑down
	0	1	1	1	
	0	1	0	1	
	0	1	0	0	
even	1	1	0	0	↓up
	1	1	0	1	
	1	1	1	1	
	1	1	1	0	
odd	1	0	1	0	↑down
	1	0	1	1	
	1	0	0	1	
	1	0	0	0	

Fig. V.5

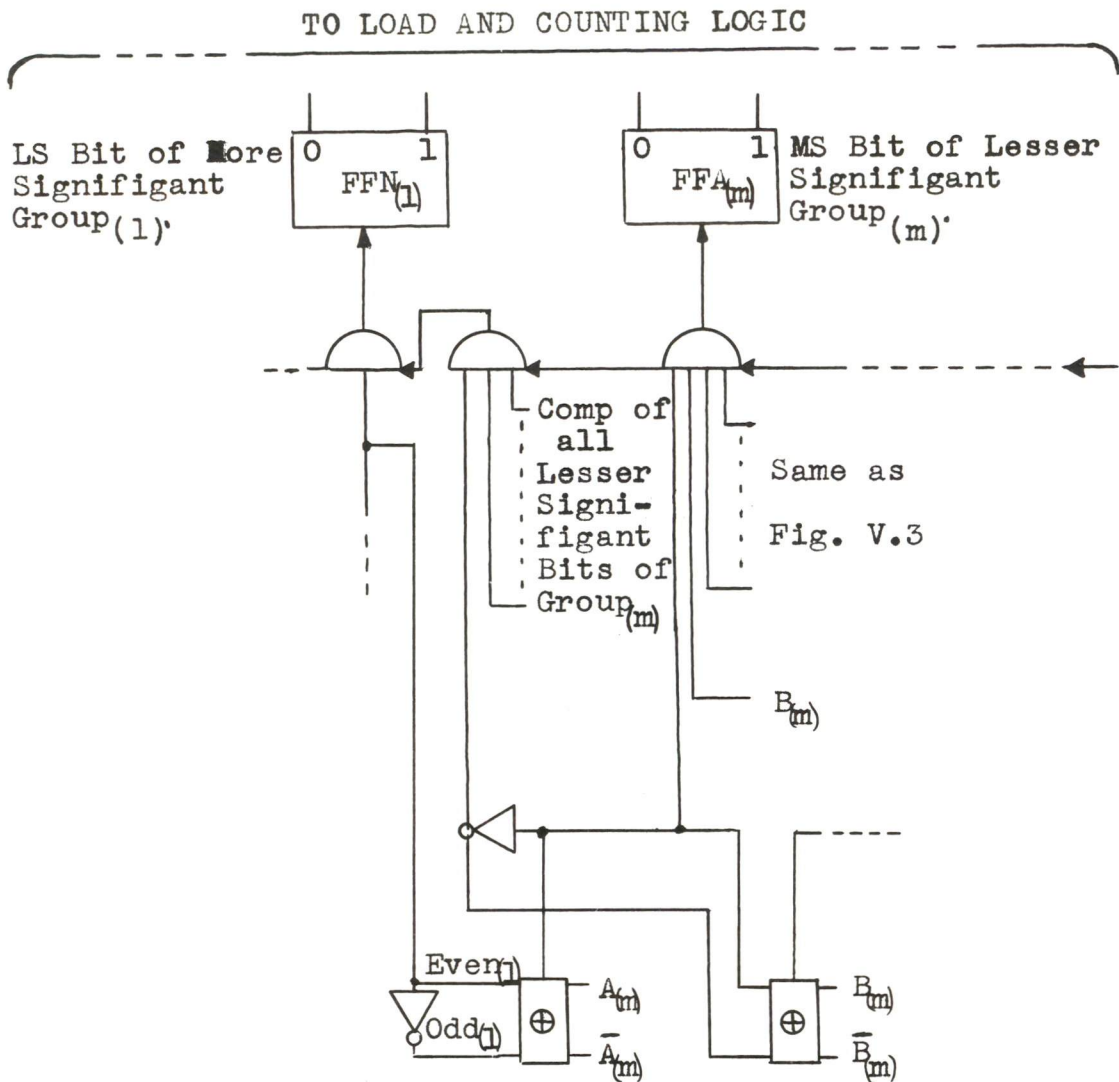
code is divided into more than two groups the relationship holds for any two adjacent groups. This characteristic of Gray codes allows one to cascade many smaller up-down Gray code counters with slightly modified counting logic to make one large Gray code counter (and to cascade decades of the Datex decade counters mentioned in the previous

section).

To go directly from the first count to the 2^m th count or vice versa of a m bit counter the most significant bit is the one that is complemented. If the logic term for the most significant bit is modified so that this change is not possible, and a $(m+1)$ th logic term is developed in the counting logic to gate the counting pulse to the next more significant group when the first group has reached the "end" we have solved part of the problem. The other part of the problem is already solved in that the logic term of the next more significant group's least significant bit indicates whether that group is odd or even (the ring sum of any number of bits is $S_{(odd)}$ of those bits and the complement of the ring sum is $S_{(even)}$). Therefore, the logic term of the next more significant group's least significant bit can also be used to tell the next

lesser significant group which direction to count (an input to the up-down control logic described earlier in this chapter). For the Datex decade this is not true. Extra logic is therefore required to determine if a decade is even or odd. Hence the desired action is as follows: When a group is counting up (next more significant group is even), the counting logic prevents the transition from the highest count to zero (the first count) but instead steps the next more significant group by one, changing it from even to odd which reverses the direction of count in the first group. Fig. V.6 demonstrates this method of cascading Gray code counters to make one larger one.

SWITCH TAIL RING COUNTER: The switch tail ring counter mentioned in the last chapter is simply a shift register whose most significant bit's transposed output is fed to the input of the least significant bit when shifting left (counting up), and the transposed output of the least significant bit is fed to the input of the most significant bit when shifting right (counting down). Hence, when counting up the complement of the most significant bit is shifted into the least significant bit and, when counting down the complement of the least significant bit is shifted into the most significant bit. For an n bit counter, a foreshortened monostrophic counting sequence of $2n$ counts is generated. Fig. V.7 shows the counting sequence for a 5 bit switch tail counting sequence.



Method of cascading Gray code counters into one larger counter. Only differences and additions to Gray code counter shown in Fig. V.3 are shown. The differences and additions affect only the two bits adjacent to the division (point of cascading) of groups (1 & m).

Fig. V.6

Notice that for four counts only 2 bits are required, the same as for a Gray code, and is in fact a 2 bit Gray code. For eight counts, four bits are required, only one more than for a Gray code with the same number of counts. For ten counts, five bits are required, only one more than for a foreshortened monostrophic code that follows the Gray to binomial binary conversion rules. Above ten counts many more bits are required than for more compact monostrophic codes. Since only 2 bits must be sampled per each of the 2^n combinations as discussed in the previous chapter, a switch tail ring counter may be preferable from an economy standpoint if decoding (rather than conversion) is required and the total count is not more than about ten.

	A	B	C	D	E
	0	0	0	0	0
	0	0	0	0	1
down	0	0	0	1	1
↑	0	0	1	1	1
↓	0	1	1	1	1
up	1	1	1	1	1
	1	1	1	1	0
	1	1	1	0	0
	1	1	0	0	0
	1	0	0	0	0

Fig. V.7

SIMULTANEOUS GENERATION OF MONOSTROPHIC & POLY-

STROPHIC CODES: Many times when generating a monostrophic code count, it is desirable to have a polystrophic binary code count generated simultaneously with the monostrophic code.

Immediately, one may feel that the easiest and cheapest way to accomplish this is to count in one code with conversion logic for the other code connected to the output of the counter yielding both codes. Because methods of counting in binomial binary and conversion from binomial binary to Gray codes are so well known and simple, this might be the tendency. Where "on the fly" sampling of the monostrophic code is required this approach is not satisfactory because the ambiguities during transition from one count to the next in the binomial binary counter's output are transferred through the conversion logic resulting in ambiguities during transition of the monostrophic code. Therefore, if conversion logic on the output of the counter is to be used to generate both codes in parallel the counter must count monostrophically, with conversion to the binomial binary code. This approach yields a monostrophic code with no ambiguities, with the simultaneous generation of the binomial binary code.

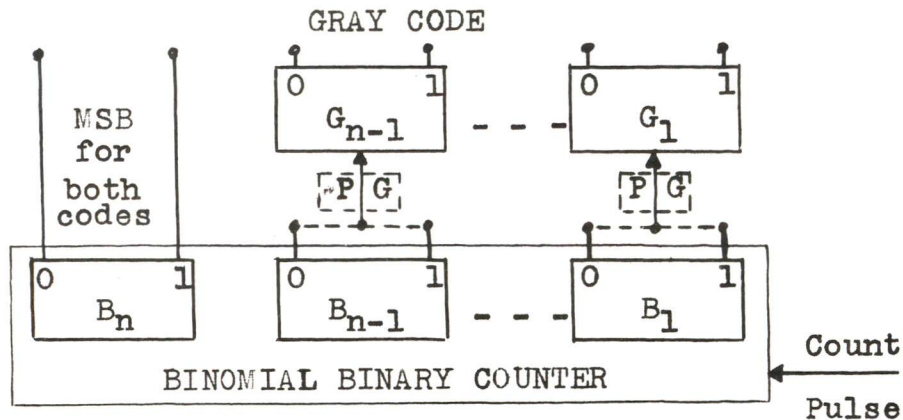
CONTINUOUS CONVERSION OF BINOMIAL BINARY TO GRAY: If the monostrophic code is Gray or pseudo-Gray and the polystrophic code is binomial binary the continuous conversion method shown in Fig. V.8 and described in the following discussion is applicable.

A relationship between corresponding bits (except most significant bits which are always equal) of binomial binary and Gray codes is that, in the up counting sequence of both

codes, when a bit changes from 0 to 1 in the binomial binary code the corresponding bit of the Gray code is complemented. A 1 to 0 transition of any binomial binary bit does not affect the Gray code. The opposite is true if the counting sequence of both codes is down. This relationship can be used to enable the transitions of bits in an n bit binomial binary counter to modify the corresponding bits of an (n-1) bit register which contains the lesser significant bits of the Gray code (or pseudo-Gray code if the output sensing pattern is complemented and/or permuted) count. Fig.V.8 demonstrates this approach. Using some manufacturers electronic logic modules this approach is the cheapest way to generate Gray (or pseudo-Gray) codes.

In sequential switching schemes using relay flip-flops to generate Gray (or pseudo-Gray) code counting sequences this approach is a good one because of the reduction of relay contacts (paid for by doubling the number of relays). Fig. V.9 shows an all relay n bit binomial binary counter with continuous conversion to an n bit Gray code.

N BIT DECODERS FOR COUNTING LOGIC: Most electronic logic manufacturers include in their product lines 2, 3 and occasionally 4 bit decoders. These decoders usually have as inputs two lines per bit (the logical value of the bit and the complement) and 2^n output lines, of which only



Connect complement input of each Gray bit (less MSB) to the 1 output of the corresponding binary bit for up counting sequence and the 0 output for down counting sequence. If flip-flops require a pulse instead of 0-1 level change pulse generators must be added as shown.

CONTINUOUS CONVERTER BINOMIAL BINARY TO GRAY

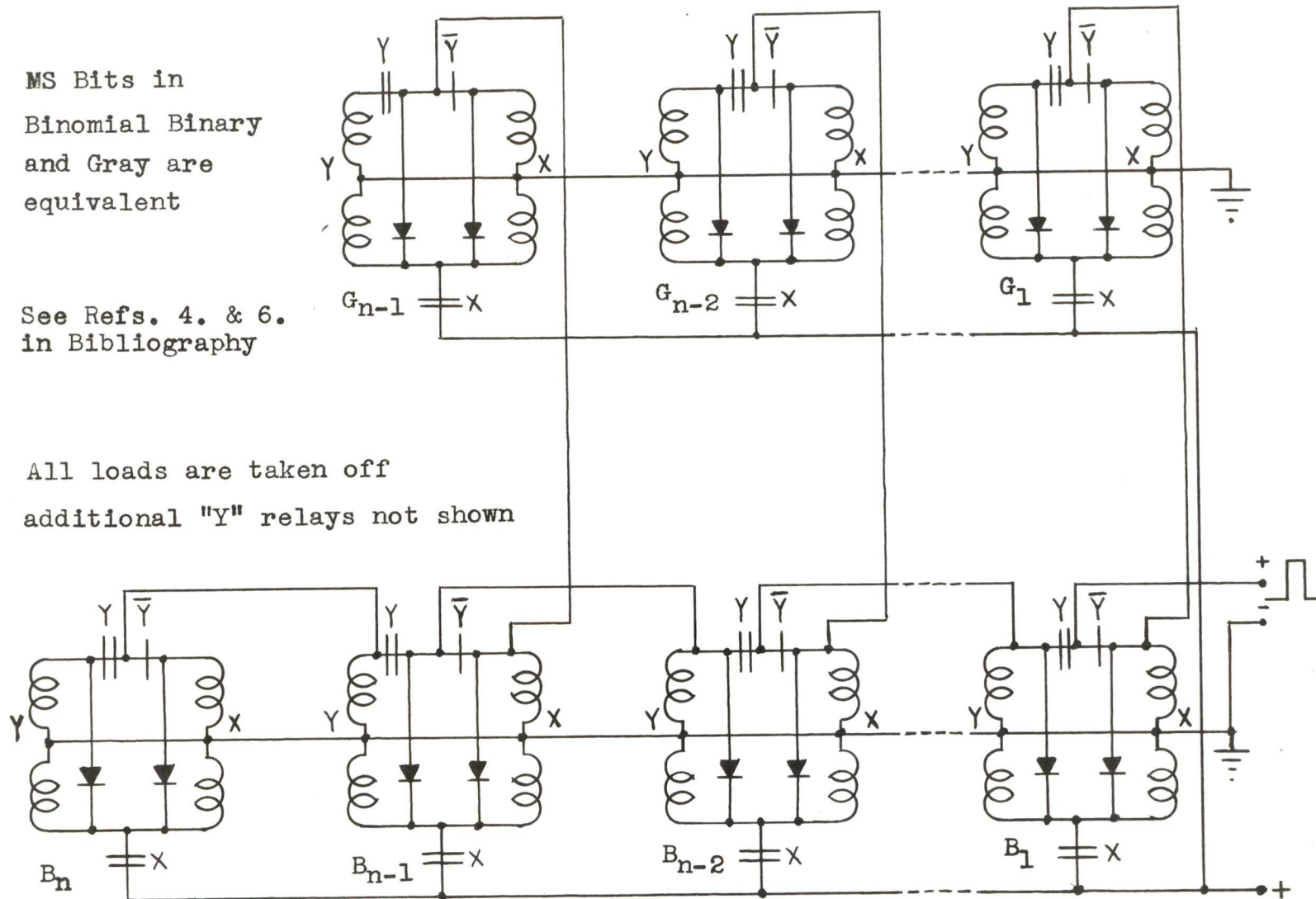
Fig. V.8

one may be selected at one time. Each output line is usually labelled with the decimal equivalent of the n bit binomial binary number which, as an input, selects that line. If an n bit decoder is connected to the output of an n bit flip-flop register, the proper output lines can be Ored into n pulse gates, each one associated with the complement input of each flip-flop in the register. By pulsing the pulse gates the contents of the register goes through a counting sequence dependent upon the pattern of Oring the decoder output lines. For a monostrophic counting sequence, each decoder output line will be associated with

MS Bits in
Binomial Binary
and Gray are
equivalent

See Refs. 4. & 6.
in Bibliography

All loads are taken off
additional "Y" relays not shown



CONTINUOUS BINOMIAL BINARY TO GRAY CONVERTER

only one pulse gate. This is not true for a polystrophic counting sequence.

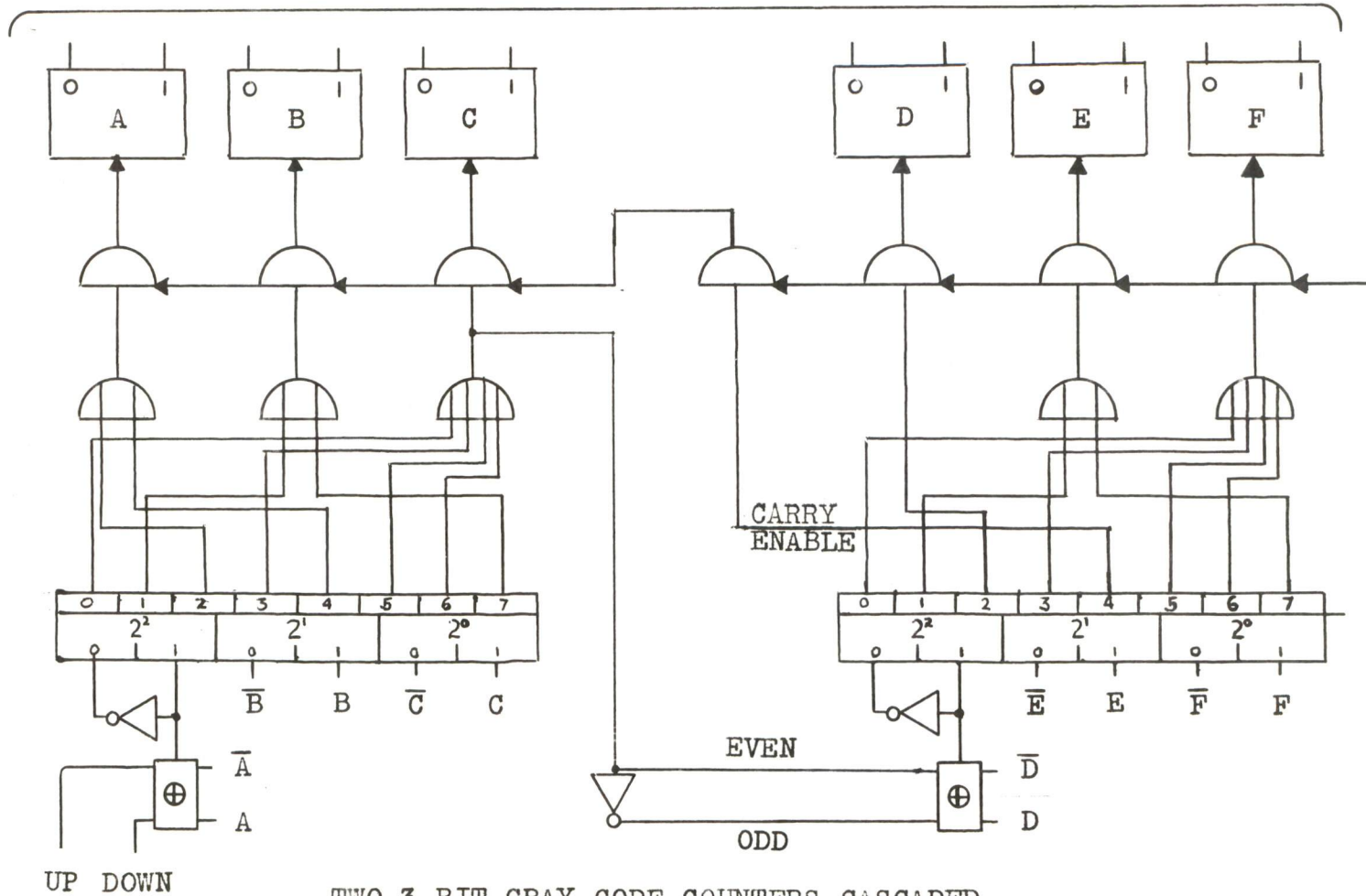
This approach is, in effect, the method used in binary counters that operate on the principle of "anticipated carry" with the accompanying advantages of equal settling time of the counting logic for all counts and faster propagation of logic levels from the output of the register to the pulse gates enabling faster counting. (Note that the counting logic shown in Fig. V.3 requires propagation of a level change through the cascaded ring sum logic of all n bits when either of the two most significant bits or the direction of count is complemented.)

In an up-down Gray counter, or when many small counters are cascaded into one large counter (requiring the control of direction of count in each small counter), the carry to the next most significant group and the inhibition of the "return to zero" in each group is more easily implemented using the decoder counting logic technique under discussion in this section. Fig. V.10 shows a 6 bit Gray counter comprising two 3 bit counters, each using a 3 bit decoder as part of the counting logic. A comparison with Fig. V.3 and Fig. V.6 shows that this approach for an up-down counter made up of smaller counters is simpler.

To determine the Oring pattern of the decoder outputs, write out the counting sequence and, beside each count, write the decimal equivalent of the binomial binary

TO COUNTING LOGIC AND LOAD

FIG. V.10



TWO 3 BIT GRAY CODE COUNTERS CASCADED
USING 3 BIT DECODERS AS COUNTING LOGIC

interpretation of the count. As previously explained in this chapter, the contents of the counter determines which bit is complemented on the next count. The decimal numbers associated with the counts preceding the complementing of a bit in a counting sequence correspond to the lines that must be Ored into that bit's complement pulse gate.

If the counting sequence is reflective the counting sequence may be reversed by reversing the sense by the decoder of the most significant bit. When groups of small counters are cascaded into one larger counter the decimal equivalent of the binomial binary interpretation of the last count in the sequence is the one that enables the carry (and inhibits the "return to zero" by not being associated with any complement pulse gate of that group).

When cascading x groups of m bit counters into one xm bit counter ($xm=n$) rather than using an n bit counter like that in Fig. V.3, the propagation time of the logic is dependent upon x rather than n , hence the possibility of a faster counter.

CHAPTER VI. CONCLUSION

Most likely, while studying the foregoing, the imaginative reader has already thought of varied applications for monostrophic codes. Generally monostrophic codes have applications not only where asynchronous "on-the-fly" sampling is required, but also where continuous monitoring of the continually changing code to detect a value (or values) is required. In both cases, if a polystrophic code were used, precautions against ambiguities would need to be taken which adds complexity to the system.

Gray or pseudo-Gray codes are the easiest of the full count monostrophic codes to generate and/or convert. Of the foreshortened codes, those which follow the Gray to binomial binary conversion rules are the easiest to handle.

Where conversion between a monostrophic code and another specified binary coded numbering system is required, judicious selection (if both codes are not already specified) of the monostrophic code will result in simpler conversion logic, particularly if Gray to binomial binary conversion rules apply. This was shown in the examples of Chapter IV.

The pulse counting techniques shown in Chapter V (except for the continuous converter) have a characteristic which is highly desirable in many applications; equal time delay between any count and the new count after the count pulse (assuming equal logical delays in all flip-flops).

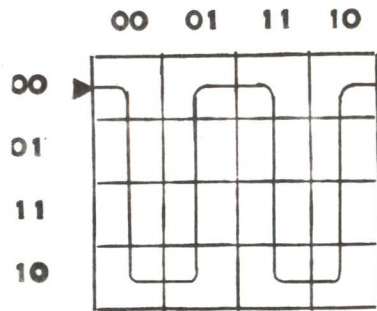
This is not so with polystrophic code counters that depend upon carry propagation. Also, the use of parallel counting logic, e. g., n bit decoders, to increase the maximum counting rate is simpler for a monostrophic code counter than with a polystrophic code counter because each output line is associated with only one complement (or set and reset) gate of the monostrophic code counter.

Besides the usual mechanical (and sometimes electronic) analog to digital conversion processes using monostrophic codes, there are many other potential applications of these codes which cannot be appreciated unless one has a "feel" for them. The overall objective of this paper is to present to the reader a better insight of monostrophic codes based on my study of them to enable him to better evaluate possible new applications employing them.

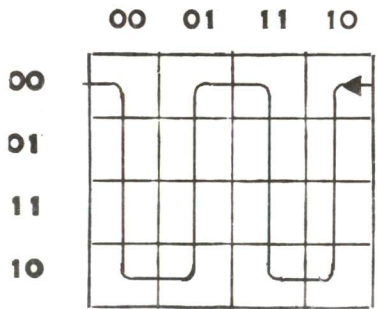
APPENDIX I
4BIT PSEUDO-GRAY CODES

The following pages contain all combinations of complemented bits and permutations of the two most significant bits of the 4 bit Gray Code. They are grouped so that each group has the same counting sequence as shown on the accompanying Karnaugh Maps. The top group of each page has the reverse counting sequence as the bottom group on the page.

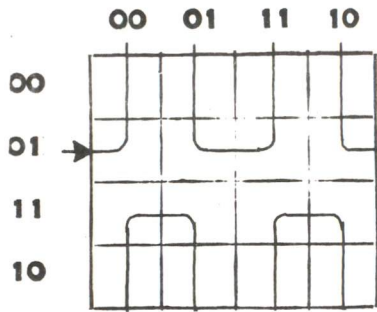
The least significant two bits' permutations are not included because, in analyzing or normalizing a pseudo-Gray code, the bits are arranged so that the least significant bit is considered the bit that is complemented most often in the counting sequence, the bit with the next fastest rate of change is considered the next to the least significant bit, etc. The two most significant bits are both complemented the same number of times in a counting sequence, hence their arrangement is arbitrary, and only their permutations are included in this appendix.



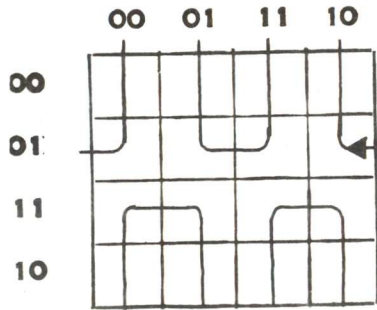
A	B	C	D	\bar{A}	\bar{B}	\bar{C}	\bar{D}	B	\bar{A}	\bar{C}	D	\bar{B}	A	\bar{C}	D
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	0	1	0	0	0	1	0	0	1	1	0	1	1	1
1	1	1	1	0	0	0	1	1	0	0	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0



A	\bar{B}	C	D	\bar{A}	B	C	D	\bar{B}	\bar{A}	\bar{C}	D	B	A	\bar{C}	D
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	1
1	1	0	1	0	0	0	0	1	1	0	1	1	1	1	1
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0



A	B	C	\bar{D}	\bar{A}	\bar{B}	C	\bar{D}	B	\bar{A}	\bar{C}	\bar{D}	\bar{B}	A	\bar{C}	\bar{D}
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	1	0	0	0	0	0
0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
1	1	0	0	0	0	0	0	1	1	0	1	0	1	1	0
1	1	0	1	0	0	1	0	1	0	1	0	0	1	0	0
1	1	1	0	0	0	1	1	1	0	0	0	0	1	0	1
1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	1
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1



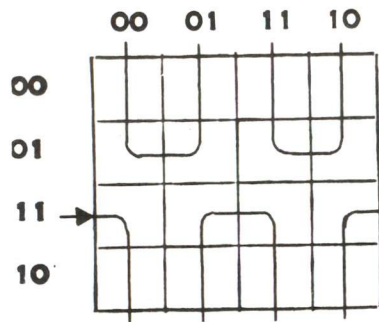
A	\bar{B}	C	\bar{D}	\bar{A}	B	C	\bar{D}	\bar{B}	\bar{A}	\bar{C}	\bar{D}	B	A	\bar{C}	\bar{D}
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	1	0	1
1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1

	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

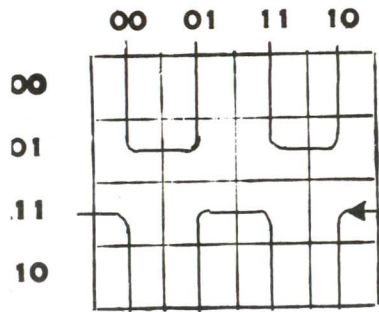
	<u>A B \bar{C} D</u>	<u>$\bar{A} \bar{B} \bar{C} D$</u>	<u>B \bar{A} C D</u>	<u>$\bar{B} A C D$</u>
00010	1	1	1	0
00011	1	1	1	1
00001	1	1	0	1
00000	1	1	0	0
01000	1	0	0	0
01001	1	0	0	1
01011	1	0	1	1
01010	1	0	1	0
01100	1	0	0	0
01101	1	0	0	1
01111	1	0	1	1
01110	1	0	1	0
10000	0	0	0	0
10001	0	0	0	1
10011	0	0	1	1
10010	0	0	1	0
10100	0	0	0	0
10101	0	0	0	1
10111	0	0	1	1
10110	0	0	1	0

	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

	<u>A \bar{B} \bar{C} D</u>	<u>$\bar{A} B \bar{C} D$</u>	<u>$\bar{B} \bar{A} C D$</u>	<u>B A C D</u>
01110	1	0	1	0
01111	1	0	1	1
01011	1	0	0	1
01010	1	0	0	0
00100	1	1	0	0
00101	1	1	0	1
00111	1	1	1	1
00110	1	1	1	0
00011	1	0	1	1
00010	1	0	1	0
00001	1	0	0	1
00000	1	0	0	0
10100	0	1	0	0
10101	0	1	0	1
10111	0	1	1	1
10110	0	1	1	0
10011	0	0	1	1
10010	0	0	1	0
10001	0	0	0	1
10000	0	0	0	0



A	B	\bar{C}	\bar{D}	\bar{A}	\bar{B}	\bar{C}	\bar{D}	B	\bar{A}	C	\bar{D}	\bar{B}	A	C	\bar{D}
0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	1	0	1	1
0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
1	1	1	1	0	0	1	1	1	0	0	0	0	1	0	1
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0	1	1	0	1	1	1
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	1	1	0	1	1	1	0	0	0	1	1	0	1	1








A	\bar{B}	\bar{C}	\bar{D}	\bar{A}	B	\bar{C}	\bar{D}	\bar{B}	\bar{A}	C	\bar{D}	B	A	C	\bar{D}
0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1
0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	1	1	1	0	0	0	1	0
0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
0	0	0	1	1	1	0	1	0	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	0
0	0	1	0	1	1	1	0	0	1	0	0	1	0	0	1
0	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1
1	0	1	1	0	1	1	1	0	0	0	0	1	1	0	1
1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0
1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1
1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0
1	1	1	0	0	0	1	0	1	0	0	0	0	1	0	0
1	1	1	1	0	0	1	1	1	0	0	0	0	1	0	1

APPENDIX II

LEGEND OF SYMBOLS USED IN THIS PAPER

	<p>Exclusive Or; Shown with two variable input, and with two variable with complements input.</p>		
	<p>Flip-flop; Assumed to have logical delay enabling simultaneous read-in and sampling of output. Shown with Set, Reset and Complement inputs. An abbreviated method of showing a Reset input (Clear input) is an arrow at lower left-hand corner of flip-flop.</p>		
	<p>And Gate</p>		<p>Or gate</p>
	<p>Nor Gate</p>		<p>Nand gate</p>
	<p>Inverter</p>		<p>Diode</p>

	<p>Normally open switch or relay contacts associated with switch or relay A.</p>
	<p>Normally closed switch or relay contacts associated with switch or relay A.</p>
<p> ∅</p>	<p>Assertion Optional Term Non-Assertion left blank</p> <p>} On Karnaugh Maps</p>
	<p>Relay Winding</p>
	<p>Logic Level Line</p>
	<p>Pulsed Line</p>

APPENDIX III
 EXCLUSIVE OR/RING SUM

The Exclusive Or function, also called the ring sum and commonly signified by the sign \oplus (and occasionally by Ψ), is an important logic function when working with generation and conversion of monostrophic codes. This function also appears in binary adders. In fact, the ring sum of two variables is often called the sum modulo two, because the ring sum of two variables satisfies the logic requirements of a half adder as shown.

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

RING SUM
 TRUTH
 TABLE

Y \ X	0	1
0	0	1
1	1	0

HALF ADDER

The following postulates and theorems will enable one to relate ring sum functions to binary logic operations.*

POSTULATES

- 1) $0 \oplus 0 = 0$
- 2) $1 \oplus 1 = 0$
- 3) $0 \oplus 1 = 1 \oplus 0 = 1$

THEOREMS

- 1) $X \oplus 1 = \bar{X}$
- 2) $X \oplus \bar{X} = 1$

*Samuel H. Caldwell, Switching Circuits and Logical Design, p. 667.

3) $X \oplus X = 0$

4) $X \oplus \bar{X} = 1$

5) $X \oplus X \oplus \dots \oplus X = \begin{cases} X & \text{for odd number of terms} \\ 0 & \text{for even number of terms} \end{cases}$

6) $X \oplus Y = Y \oplus X = \bar{X}\bar{Y} \oplus XY = (X+Y)(\bar{X}+\bar{Y})$

7) $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$

8) $XY \oplus XZ = X(Y \oplus Z)$

9) $\overline{(X \oplus Y)} = \overline{(Y \oplus X)} = \bar{X}Y + X\bar{Y} = (\bar{X}+Y)(X+\bar{Y}) = \overline{(\bar{X}Y + X\bar{Y})}$

Theorems 6) and 7) give the clue to the manipulation of more than two variables. For example, let us expand the ring sum of four variables, $A \oplus B \oplus C \oplus D$.

by theorem 7

$$A \oplus B \oplus C \oplus D = (A \oplus B \oplus C) \oplus D$$

by theorem 6

$$= (A \oplus B \oplus C) \bar{D} \oplus (A \oplus B \oplus C) D$$

by theorem 7

$$= [(A \oplus B) \oplus C] \bar{D} \oplus [(A \oplus B) \oplus C] D$$

by theorem 6

$$= [(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C] \bar{D} \oplus [(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C] D$$

by theorems 6 and 9

$$= [(A\bar{B} + \bar{A}B) \bar{C} + (AB + \bar{A}\bar{B}) C] \bar{D} \oplus [(AB + \bar{A}\bar{B}) \bar{C} + (A\bar{B} + \bar{A}B) C] D \\ = A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}D + \bar{A}\bar{B}CD$$

Notice that the expanded term is the symmetric function $S_{(1,3)}(A,B,C,D)$. It can be shown that the ring sum of n variables is the symmetric function $S_{\text{odd}}(A,B,---,N)$

and the complement of the ring sum of n variables is $S_{\text{even}}(A,B,---,N)$. Because the ring sum of more than two variables is difficult to recognize on a Karnaugh Map, the relationship of ring sum functions to symmetric functions is probably the easiest way to recognize ring sum functions of more than two variables.

BIBLIOGRAPHY

1. Barr, Paul "Analog to Digital Converters," Electro-mechanical Design, 8:165-176, January, 1964.
2. Building Blocks Technical Manual (Norwood, Mass.: Raytheon Co., Communications and Data Processing Operation, n. d.).
3. Caldwell, S. H. Switching Circuits and Logical Design (New York: John Wiley and Sons, Inc., 1958).
4. Clareed Control Modules, Manual 400 (Chicago: C. P. Clare and Co., 1962).
5. Couleur, John F. "BIDEC--A Binary-to-Decimal or Decimal-to-Binary Converter," IRE Transactions on Electronic Computers, 313-316, December, 1958.
6. Counting With Clareed Control Modules, Application Manual 401 (Chicago: C. P. Clare and Co., 1962).
7. Datex Code, Bulletin 001 (Monrovia, Calif.: Datex Corp., 1963).
8. Digital Modules, Catalog A-705A (Maynard, Mass.: Digital Equipment Corp., 1962).
9. Evans, D. S. Digital Data (London: Hilger and Watts, Ltd., 1961).
10. Grabbe, Ramo and Wooldridge (Editors) Handbook of Automation, Computation and Control (New York: John Wiley and Sons, Inc., 1959). 3 Vols.
11. Graf, R. F. Dictionary of Electronics, 2nd Edition (Indianapolis: Howard W. Sams and Co., Inc., 1963).
12. Holcombe, Waldo "Relays That Challenge Semiconductors," Electronics, 37: 56-60, March 23, 1964.
13. Humphrey, W. S. Switching Circuits with Computer Applications (New York: McGraw-Hill, 1958).
14. Laboratory Module Handbook (Maynard, Mass.: Digital Equipment Corp., 1963).
15. Maley, Gerald A., and Earle, John The Logic Design of Transistor Digital Computers (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1963).

16. Marcus, M. P. Switching Circuits for Engineers (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1962).
17. Meyer, B. W. "Arbitrary-Length Binary Counters," Electronics, 36: 34-36, December 27, 1963.
18. Robbins, Lionel "One-Brush Encoder," Instruments and Control Systems, 36: February, 1963.
19. "Shaft-Angle Encoders" Computer Design, 2: 16-41, December, 1963.
20. Shaft Position Encoders, Bulletin 312-B (Monrovia, Calif.: Datex Corp., 1963).
21. Wilson, M. Carr "Gray to Binary Converter," Instruments and Control Systems, 35: June, 1962.