Synopsis: A sequential network consists of an interconnection of logical elements, such as "and"-gates, "or"-gates, inverters, etc., and storage elements, such as flip-flops and delay lines. These networks process signals, usually binary signals, in the sense that they convert input sequences of 0's and 1's into output sequences functionally related to the input sequences. Typically, these networks occur as parts of digital computers and control systems, and perform operations such as counting, code conversion, program control, addition, comparison, sequence generation, etc. Such networks are presently designed by unsystematic cut-and-try methods. Consequently they are unnecessarily costly, and are often more difficult to test and maintain because of the lack of patterned structure in their realization.

Probably the knowledge most lacking for development of good synthesis techniques for sequential networks is a better understanding of the relationship between the internal logic and the state-sequential behavior of such networks. This paper explores this relationship through the mechanism of the C-matrix. In particular, it derives some conditions for the matrix to be non-singular; i.e., to correspond to a state diagram which is deterministic even in reversed time, and show some consequences of the nonsingularity condition. Also, the effects on the state graph of several kinds of constraints imposed on the logic are determined. Several special classes of sequential nets are analyzed with the C-matrix: the nonlinear-feedback shift register, the safe asynchronous net, the fully self-independent net, and the net with cyclically permuted logic. Finally, the realization of C-matrices is discussed using various types of binary storage elements, such as set-reset flip-flops, trigger flip-flops, and relays.

Model of a Sequential Net

A sequential network consists of an interconnection of logical elements, such as "and" gates, "or" gates, inverters, etc., and storage elements, such as flip-flops and delay lines. These networks process signals, usually binary signals, in the sense that they convert input sequences of 0's and 1's into output sequences functionally related to the input sequences.

\[
\begin{align*}
Y_1(Y_0, X_1, X_2, ... X_n, Y_1, ... Y_p) \\
Y_2(Y_0, X_1, X_2, ... X_n, Y_1, ... Y_p) \\
... \\
Y_p(Y_0, X_1, X_2, ... X_n, Y_1, ... Y_p)
\end{align*}
\]

Such a network may be depicted with no loss of generality as shown in Fig. 1. In this model the binary storage elements are given the form of delay elements, and are separated for purposes of analysis from the purely logical elements, which make up the combinational network. The general property of the delay element will be apparent in a later section. This network, being purely combinational, contains no memory of any kind. As an abstract model of a physical network, it may be assumed to respond immediately to changes in its inputs. The network is assumed for the time being to operate synchronously; that is, all changes in the internal variables occur simultaneously, as if under the control of an "advance clock."

For a sequential net with \(n\) delay elements, \(p\) inputs, and \(q\) outputs, the combinational network has \(n+p\) input signals and \(n+q\) output signals. The terminal behavior may, therefore, be described completely in terms of the \(n+q\) switching functions of \(n+p\) variables:

\[
x_i'(X_1, X_2, ... X_n, Y_1, ... Y_p), \quad k = 1, 2, ... n \\
x_j(X_1, X_2, ... X_n, Y_1, ... Y_p), \quad j = 1, 2, ... q
\]

These equations state that the next state and the output of the network are completely determined by the present state and the input.

Techniques for the analysis and synthesis of combinational nets are well known, although truly economical synthesis is a goal not yet achieved, particularly for multioutput networks such as that of Fig. 1.

The content of the equations may be expressed in tabular form in a "table of combinations" (see following page).
The rows of the left section of this table are filled in with all possible \((n+p)\)-digit binary numbers, usually in their natural binary order. Thus, there are \(2^{n+p}\) rows. Each row represents the situation of one of the possible inputs in conjunction with one possible internal state. The rows of the right section of the table, when completely filled in, define the sequential network, in that the output and the next state are then specified. Clearly, any manner of completely filling in the right section of the table with '0's and '1's yields the table of a realizable sequential network. The number of such networks is equal to the number of ways in which this part of the table may be filled in: \(2^{(n+p)2^{n+p}}\).

The C-matrix is now defined as the array of entries in the right-hand section of the table, assuming that the rows of the left-hand section are in their natural binary order. For convenience turn the pattern on its side:

\[
\begin{bmatrix}
  & y_1, y_2, \ldots, y_p & x_1, x_2, \ldots, x_n & x_1', x_2', \ldots, x_n' & z_1, z_2, \ldots, z_q \\
0000 & 0000 & 0 & 0 & 0 \\
0001 & 0001 & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0100 & 0100 & 1 & 0 & 0 \\
0101 & 0101 & 1 & 0 & 1 \\
1010 & 1010 & 1 & 1 & 0 \\
1011 & 1011 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Thus, when the input \(y_1=0\), the counter advances with state 01 following state 00 (column 0), state 10 following state 01 (column 1), state 11 following state 10 (column 2), and state 00 following state 11 (column 3). When \(y_1=1\), state 11 follows state 00 (column 4), state 00 follows state 01 (column 5), state 01 follows 10 (column 6), and state 10 follows state 11 (column 7). \(z_1=1\) only when the present state is 11.

The expression of a switching function of \(n\) variables as a \(2^n\)-digit binary number, this number being a column of the table of combinations (a row of the C-matrix), is sometimes called the designation number or characteristic number of the function. Thus, the characteristic number of \(x_1'\) in the matrix is

\[
x_1' = [1010 \ 1010]
\]

The 1's and 0's in this representation indicate which items in the "disjunctive normal form" or canonical expansion of the function \(x_1'\) are present and absent, respectively. Here

\[
x_1' = 1\cdot y_1\cdot y_2\cdot x_3 + 0\cdot y_1\cdot y_2\cdot x_3 + 1\cdot y_1\cdot y_2\cdot x_3 + 0\cdot y_1\cdot x_2\cdot x_3 + 1\cdot y_1\cdot y_2\cdot x_3 + 0\cdot y_1\cdot x_2\cdot x_3 + 1\cdot y_1\cdot y_2\cdot x_3 + 0\cdot y_1\cdot x_2\cdot x_3
\]

The Coding Problem

The major objective of sequential synthesis studies is the development of methods of deriving the set of combinational switching functions \(x_i'\) \((k=1, 2, \ldots, n)\) and \(z_i\) \((i=1, 2, \ldots, q)\) from a terminal description of the sequential network as a whole. This problem mainly involves obtaining answers to the following two important questions:

1. Precisely what constitutes an adequate terminal description of a sequential network?
2. What is the essence of the desired relationship between this terminal description and the equations of the combinational network?

From the synthesis viewpoint, question 1 amounts to asking for a mathematical model and language in which to express the specifications of the network to be synthesized. Question 2 gets at the heart of the synthesis process itself, and it is with this question that this paper is concerned.

Virtually all of the past effort on the sequential network problem has been directed towards the network characterization problem \(1, 2, 3\) and the linear transducer.\(^7\)

Two simplifications are in order. First, since question 1 has not yet been fully answered, the 'state graph,' which is the most suitable terminal description of a sequential net presently available will be used. The state graph, or an equivalent tabular presentation, represents each of the \(2^n\) possible delay-element output signals, or states, \((x_1, x_2, \ldots, x_n)\) as a node in a graph, with one interconnecting directed branch for each allowed state-to-state transition. Fig. 2 shows a simple but typical state graph corresponding to the reversible counter previously described. Each transition is conditional on the appropriate input (the number preceding the slant) and gives rise to an output (the number following the slant). For example, if the net is in state 01 with input 0, the next state will be 10, and output 0 will occur.

The second simplification is that this paper will consider only the case when the network is autonomous, that is, the case when there are no inputs or outputs.
The switching functions may then be formed by inspection, and these expressed sequentially network starts with the netlogic. A graph can then be derived from the work arranged in the model of Fig. 1. Branches of the state graph (transition), which need carry no economy in synthesis, the output functions may be derived appropriate to the combinational network is linear; i.e., when it consists of exclusive "or" gates only. There are good indications that it is true in general.

To eliminate the outputs as well as the inputs is to focus attention on the next-state functions \( x_k' \) as the appropriate indications of the internal behavior of the net. Except from the standpoint of economy in synthesis, the output functions \( z_j \) may be viewed merely as appendages to the internal state-controlling logic.

With these simplifications, the digits both preceding and following the slant on the branches of the state graph of an autonomous net may be eliminated. Each node (state) then has only one exit branch (transition), which need carry no label.

The analysis of an autonomous sequential network starts with the network arranged in the model of Fig. 1. The switching functions may then be formed by inspection, and these expressed as the rows of the C-matrix (columns of the table of combinations). The state graph can then be derived from the columns of the C-matrix (rows of the table of combinations). These steps are shown in Figs. 3 and 4 for a simple example.

It is apparent that the C-matrix contains implicitly a relationship between the terminal or state-behavior and the internal logic of a sequential network. The state-behavior is contained in the columnar view of the matrix, while its rows express the logic. The remainder of this paper investigates more fully this relationship as expressed in the C-matrix, for the case of autonomous networks.

The synthesis of an autonomous sequential network may be taken to start with a state graph which has no binary labels on the nodes. The first step of the synthesis is the assignment of a unique \( n \)-digit binary number to each of the nodes, a process referred to as the coding of the state graph. The C-matrix may then be formed, column by column, by noting the successor states to states \( 00 \ldots 00, 00 \ldots 01, 00 \ldots 10, \ldots , 11 \). The derivation of the next-state functions \( x_k' \) follows from the rows, and from these functions equivalent expressions may be derived appropriate to the binary storage elements which are being employed.

The network and the C-matrix of an autonomous state graph which remains autonomous if the arrows on the branches are reversed are said to be nonsingular. (Huffman terms such a network lossless.) Each state then has a unique preceding state as well as a unique succeeding state. This definition is clearly equivalent to the property of possessing a deterministic time-inverse. Nonsingular networks are worthy of special consideration for two reasons. First, the steady-state behavior of autonomous networks clearly depends only upon the cycles present in its state graph, and not on the "tails" of branches which lead into the cycles. Second, as will be shown in the next section, the most general transformation which carries one coded state graph into another is most naturally represented as an arbitrary nonsingular C-matrix.

It follows directly that the state graph of an autonomous nonsingular net consists entirely of closed loops or cycles. A complete description of the structure of such a state graph can then be given by its cycle set, which is a listing of the lengths \( \ell_i \) of its cycles (the number of states in the cycles), each with its multiplicity \( r_i \). To describe the cycle set, the notation, \( c.s. = (\ell_1 r_1, \ell_2 r_2, \ldots , (\ell_n r_n) \) is used. E.g., for the state graph \( S \) of Fig. 5, the expression \( c.s.(S) = (1_2, 2_3, 3_4) \) indicates that \( S \) has two 1-cycles, one 2-cycle, two 3-cycles, and one 6-cycle.

Since the total number of states is \( 2^n \), the numbers \( \ell_i \) and \( r_i \) must obviously satisfy the condition:

\[
\sum_{i=1}^{n} \ell_i r_i = 2^n
\]

This sum condition is clearly necessary for a cycle set to be realizable. It should also be clear that any coding whatsoever of the state graph will lead to a realizable C-matrix so long as: 1. no two nodes are given the same state number, and 2. the state graph has \( 2^n \) nodes. Thus, the sum condition is sufficient as well as necessary for a cycle set to be realizable.

For example, realization of a cycle set \((1, 7)\) requires that the state graph of
binary storage elements and logical elements, the economical realization of the multioutput logic. This has led to much more intricate circuit network realizing the given cycle set is that displayed in Fig. 8, drawn in two ways. Almost any other coding would correspond to the successive application of the two transformations, viewed individually as state-to-state transformations and as a set of transforming equations. Also, the inverse $C^{-1}$ of a nonsingular C-matrix is that matrix which satisfies the equation $C^{-1}C = CC^{-1} = I$ so that if $x' = Cx$, then $x = C^{-1}x'$. From a state-sequential viewpoint, the inverse C-matrix can be written down immediately from the state graph by imagining all the arrows on the branches to be reversed in direction. From a logic-equation viewpoint, $C^{-1}$ is an expression of the solution of the equations $x_1'(x_1, \ldots, x_n)$ for the unprimed variables.

Powers $C^m$ of a C-matrix correspond to simple iteration. $C^m$ transforms each state $x$ into the state $m$ time intervals in the future. The state cycle structure of the state graph can be studied through the periodicity properties of the powers of a C-matrix. E.g., it is not hard to show that if a power $M$ can be found such that $C^M = I$ then all cycle lengths in the state graph for $C$ divide $M$.

An interesting example is afforded by the conventional binary counter, which carries each binary state number into the next number in the natural binary counting sequence. The C-matrix of such a counter is identical to the identity C-matrix $I$, except that all of the columns are shifted one place to the left; e.g., for $n = 3$,

$$C_3 = \begin{bmatrix} 1010 & 1010 \\ 0110 & 0110 \\ 0001 & 1110 \end{bmatrix}$$

(The first column is shifted around to the last position.) The logic can be read directly from the rows. For a binary counter which counts $m$ at a time instead of $I$ at a time, it is only necessary to raise $C_3$ to the $m$th power. This amounts to shifting the columns of the identity matrix $m$ instead of $I$ place to the left. As long as $m$ and $2^n$ are relatively prime numbers (i.e., as long as $m$ is odd), the state graph will consist of a single $2^n$-cycle.

In view of the importance of the coding problem, it is worthwhile to inquire into the nature of the transformation which carries one coding of a state graph into another. Clearly, any such recoding amounts merely to a permutation of the $2^n$ possible state numbers. The most general such transformation is therefore of the nature of a nonsingular C-matrix itself.

If the original state graph is labelled with states $x$ and the recoded state graph with states $y$, and call the transformation matrix $Q$, then $y = Qx$. Similarly, $y' = Qx'$, and from $x' = Cx$, the result is $Q^{-1}y' = Q^{-1}Qx' = x'$. or, since $Q$ is nonsingular, $y' = Q^{-1}Qx'$. The new C-matrix for the $y$-states, $C'$,

**Manipulation of C-Matrices**

If the state numbers are defined as column matrices,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

then the formal equation $x' = Cx$ expresses the role of the C-matrix as a transformation from a present state to a next state. The multiplication indicated is clearly not matrix multiplication in the usual sense, but is an expression of the fact that the independent and dependent variables in the row-functions of the C-matrix are the elements of the column matrices $x$ and $x'$, respectively. The identity C-matrix $I$ (Campeau's A-matrix) is therefore the C-matrix for which $x' = Ix = x$.

That is, it is the matrix which expresses the set of equations $x_i' = x_i$:

$$I_1 = [01], \quad I_2 = \begin{bmatrix} 0101 \\ 0011 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 0101 & 0101 \\ 0011 & 0011 \\ 0000 & 1111 \end{bmatrix}$$

The corresponding state graph then consists of $2^n$ 1-cycles, since each state is followed by the same state. Similarly, multiplication of two C-matrices, $A$ and $B$, also has both a state sequential and a logical interpretation, corresponding to the successive application of the two transformations, viewed individually as state-to-state transformations and as a set of transforming equations.
Realization of Storage Elements

Types of binary storage elements other than the delay element may be related to the delay element by exhibiting the extent of self-dependence in the individual switching functions (individual rows of the C-matrix)

\[ x'_k = f_k(x_1, x_2, \ldots, x_n) \]

Expanding \( x'_k \) about \( x_k \) results in:

\[ x'_k = g_k(x_1, x_2, \ldots, x_n) + x_k h_k(x_1, x_2, \ldots, x_n) \]

where \( g_k \) and \( h_k \) are independent of \( x_k \). In fact,

\[ g_k = f_k \quad x_k = 0 \]
\[ h_k = f_k \quad x_k = 1 \]

A further development gives

\[ x'_k = x_k h_k + x_k g_k + x_k \]

in which the functions \( l_k, q_k, s_k \), and another function \( r_k \), are also independent of \( x_k \), and in addition

\[ g_k + r_k + s_k + l_k = 1 \]
\[ q_k + r_k = g_k + h_k + r_k + s_k = x_k h_k = 0 \]

Thus, the \( q\)-\( r\)-\( s\)-\( t\)-functions are disjunctive; that is, one and only one of them has the value "1" for each combination of values of the \( n-1 \) variables \( x_1, x_2, x_3, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n \). The \( q\)-\( r\)-\( s\)-\( t\)-functions, and \( g\) and \( h\)-functions are simply related by the expressions

\[ q_k = g_k h_k \]
\[ r_k = h_k g_k + l_k \]
\[ s_k = g_k h_k + l_k \]
\[ g_k = l_k + x_k \]
\[ h_k = q_k + s_k \]

The \( q\)-\( r\)-\( s\)-\( t\)-functions may be interpreted in terms of the signals applied to a conventional flip-flop as follows:

- \( r_k = 1 \), so \( x'_k = 0 \): reset (i.e., switch the flip-flop to the "0" state)
- \( l_k = 1 \), so \( x'_k = x_k \): complement (trigger)
- \( q_k = 1 \), so \( x'_k = x_k \): hold (i.e., leave the flip-flop as it is) (quiescent condition)
- \( s_k = 1 \), so \( x'_k = 1 \): set (i.e., switch the flip-flop to the "1" state)

From a given function \( x'_k \) expanded in terms of the \( q\)-\( r\)-\( s\)-\( t\)-functions, therefore, one may write down immediately the switching functions for the set, reset, and complementing signals to be applied to a flip-flop realizing the given function \( x'_k \):

Case I
- set = \( S_k = s_k \)
- reset = \( R_k = r_k \)
- complement = \( T_k = l_k \)

Case II
- The flip-flop has only two terminals, set and reset. Complementing is accomplished by applying a "1" to both set and reset terminals at the same time. In this case:
  \( S_k = s_k + l_k = h_k \)
  \( R_k = r_k + l_k = g_k \)

Case III
- No complementing terminal is available and simultaneous signals on the set and reset terminals are not allowed. Set and reset signals to flip-flop no. \( k \) are then generally dependent on the outputs of flip-flop no. \( k \) itself:
  \( S_k = s_k + t_k \)
  \( R_k = r_k + t_k \)
  \( g_k = g_k + t_k \)

Actually, in this case, application of the "hold" signal to the set or reset terminal can do no harm if the flip-flop is already "on" or "off", respectively. So the preceding equations may be replaced by the condition:

\[ S_k = s_k + l_k = h_k \]
\[ R_k = r_k + l_k = g_k \]

where the bracket signifies a "don't care" condition, indicating that any part or all of the bracketed function may be combined with the rest of the expression to effect a possible simplification. (Since the \( q\)-\( r\)-\( s\)-\( t\)-functions constitute a disjunctive set, however, combinations will be possibly only with the more unusual types of logic elements in the network.)
Neutral Switching Functions

If the number of 1's in the characteristic number of a function \( f \) is equal to the number of 0's, the function is said to be neutral. It will be shown now that every switching function in a nonsingular C-matrix is neutral.

From the purely cyclic character of the state graph, it follows that each of the \( 2^n \) states has a unique subsequent state. Thus, each of the \( 2^n \) binary state numbers must appear once and only once as columns in the C-matrix. The columns of a nonsingular C-matrix are, therefore, merely a permutation of the columns of the identity C-matrix of the same order, for which the switching functions are \( x_k' = x_k (k = 1, 2, \ldots, n) \). Since the functions \( x_k \) are all neutral, the functions resulting from any column permutation of the identity C-matrix are also neutral.

The only neutral functions of two variables, for example, \( x \) and \( y \), are of symmetry type

\[
x, x \oplus y
\]

and for three variables, \( x, y, z \):

\[
x, x \oplus y, x \oplus y \oplus z
\]

The operation \( + \) may be substituted for \( \oplus \) in the last two expressions. There are 74 neutral symmetry types of 4 variables. Forty-two of these, along with all neutral symmetry types of 2 and 3 variables, are the same as their complements; the complements of the other 32 are not the same, but form 16 mutually complementary pairs.

If \( g \) and \( h \) are any neutral functions independent of a variable \( x \), then it may easily be shown that the function \( \overline{g}x + x\overline{h} \) is also neutral. Further, if \( f_0 \) is any function independent of \( x \), then the function \( n\overline{f}_0 \) is neutral.

Nonsingularity of C-Matrices

Although each of the \( n \) switching functions \( x_k' \) in a nonsingular C-matrix is neutral, most sets of \( n \) neutral functions represent singular C-matrices. It will be shown that a C-matrix is nonsingular if and only if all linear sums

\[\sum x_k'\]

are neutral. (Note: On this page the symbol \( \Sigma \) means the sum of modulo-2.)

**Proof:**

The characteristic number of each of the \( 2^n - 1 \) sums \( \Sigma x_k' \) can be formed by adding (modulo-2) the characteristic numbers of each possible subset of rows of the C-matrix. If the C-matrix is nonsingular, its columns are merely a permutation of the columns of the identity C-matrix. Therefore, each sum \( \Sigma x_k' \) will be neutral only according as the sum of the corresponding rows of the identity C-matrix are neutral. Since the individual rows of the identity C-matrix represent the functions \( x_1, x_2, \ldots, x_n \), any sum of these rows has the form \( \Sigma x_i \), and by the last statement of the previous section is certainly neutral. Therefore, all sums \( \Sigma x_k' \) are neutral for a nonsingular C-matrix.

To establish the converse, the measure \( \mu(f) \) of a logic function \( f \) is defined to be the number of 1's in the characteristic number of \( f \). Thus the measure of a neutral function is \( 2^n - 1 \); in fact it is true that:

\[\mu(f) = \mu(f) = 2^n - 1\]

By adding their characteristic numbers, it is easily seen that for any two functions \( f_1 \) and \( f_2 \) of \( n \) variables

\[\mu(f_1 \oplus f_2) = \mu(f_1) + \mu(f_2) - 2\mu(f_1 f_2)\]

\[\mu(f_1 f_2) = \frac{1}{2} [\mu(f_1) + \mu(f_2) - \mu(f_1 \overline{f}_2)]\]

Similarly,

\[\mu(f_1 f_2) = \frac{1}{4} [\mu(f_1) + \mu(f_2) - \mu(f_1 \overline{f}_2) - \mu(f_1 f_2)]\]

etc., for more than three functions. If all of these functions and all of their \( \oplus \)-sums are neutral, each \( \mu(\ ) \) expression equals \( 2^n - 1 \); and it follows, therefore, that

\[\mu(f_1, f_2, \ldots, f_n) = 2^n - 1\]

In particular, \( \mu(f_1, f_2, \ldots, f_n) = 1 \), which is to say that the C-matrix formed from these \( n \) functions has only a single column containing all 1's.

If one or more of the functions \( f_i \) is complemented, this last equation still holds, since \( \mu(f) = \mu(\overline{f}) \) for a neutral function. So for all \( 2^n \) products,

\[\mu(f_1, f_2, \ldots, f_n) = 1\]

where \( f_i \) denotes \( \overline{f}_i \) or \( f_i \). Thus, the C-matrix has one and only one column in each of the \( 2^n \) possible patterns of 0's and 1's. The matrix is therefore a column permutation of the identity C-matrix, and is nonsingular.

It follows directly from this theorem that if a C-matrix is nonsingular, any set of permutations or complementations of the row-functions will preserve the nonsingularity property, although in general its cycle set will be modified drastically. Further, any linearly independent set of \( n \) functions selected from the \( 2^n - 1 \) functions \( \Sigma x_k' \) may be assembled to form a new C-matrix, which is then guaranteed to be nonsingular. (With the function "0," these \( 2^n - 1 \) neutral functions form an abstract group under the \( \oplus \) operation. The \( n \) functions selected for the C-matrix then constitute a set of generators for the group.)

The set of C-matrices generated in this way from the identity C-matrix constitutes the set of all linear C-matrices; that is, each function \( x_k' \) is a linear (modulo-2) sum of the unprimed variables. The linear case has been extensively studied by Huffman and by Elspas.

Huffman has shown in an unpublished memorandum an alternate condition for

\[Kaufe-State-Logic Relations\]
A further consequence of the stated condition follows from the form of the bottom row of the C-matrix: \( x_n' \) is fully dependent on \( x_n \), i.e.,
\[
x_n' = g_n(x_n)
\]
where \( g_n \) is a function which is independent of \( x_n \). Its characteristic number is given by the left half of the bottom row of the C-matrix.

Each of the three necessary conditions may be interpreted to be sufficient, under some suitable partial encoding of the state graph. Relation 1 for example, may be inverted to read: if all cycles are of even length, then there exists at least one coding of the state graph which makes \( x_n' = y_n \). The relations 4 through 16 which follow may be similarly inverted.

With the exception of relation 16, the proofs of sufficiency of all of these conditions represent merely a reverse of the necessity argument.

Combinations of these conditions lead to further relations. The second function which is restricted may be taken without loss of generality to be \( x_n' \), whose characteristic number is the last row of the C-matrix.

1. If \( x_n' = x_n \), then all cycles are even; i.e., of even length.
   
   Proof: \( x_n' = (1 \ldots 1 \ 0 \ldots 0) \) so that each state whose number is \( < 2^{n-1} \) is followed by a state \( \geq 2^{n-1} \), and each state \( \geq 2^{n-1} \) is followed by a state \( < 2^{n-1} \). Thus, on each cycle the state numbers alternate between these two nonintersecting sets \( (\geq 2^{n-1}) \) and \( (\leq 2^{n-1}) \). If, further, each cycle is made up of two subsets, each with half of the total number of states. Then, \( x_n' = x_n \) is satisfied.

2. If \( x_n' = x_n \), then the cycle-set can be halved; that is, the list of cycle lengths may be separated into two distinct subsets, each with half of the total number of states.
   
   Proof: Here \( x_n' = (0 \ldots 0 \ 1 \ldots 1) \), so that each state whose number is \( \leq 2^{n-1} \) is followed by a state \( \geq 2^{n-1} \), and each state \( \geq 2^{n-1} \) is followed by a state \( \leq 2^{n-1} \). The stated condition follows directly.

A further consequence of this condition is that the longest cycle length must be \( \leq 2^{n-1} \).

3. If all state variables \( x_n \) are independent of \( x_n \), then all odd cycles occur with even multiplicity.
   
   Proof: A function \( x_n' \) is independent of \( x_n \) if and only if the left and right halves of the characteristic number of \( x_n \) are identical. Under the present hypothesis, therefore, the left and right halves of the entire C-matrix, exclusive of the bottom row, are identical. The left and right halves of the bottom row must be complementary, since every state (column) occurs just once. Thus, every state \( \leq 2^{n-1} \) is matched by an "image" state \( \geq 2^{n-1} \). Their successors are also images, since the next-state numbers also differ by \( 2^{n-1} \). Thus the state graph must have complete mirror symmetry, and any odd cycle must be matched by an identical image. Statement 3 directly follows. (See Fig. 9.)

Finally, one additional condition is presented:

16. If \( x_n' \) depends on \( x_n \), according to the equation
   \[
x_n' = g_n x_n + h_n x_n,
\]
   where \( g_n \) and \( h_n \) are independent of \( x_n \) and individually neutral (actually, the neutrality of either \( g_n \) or \( h_n \) follows from the neutrality of the other and the nonsingularity of the C-matrix), then the number of odd cycles is no greater than \( 2^{n-1} \).

Proof: The states may be divided into four disjoint sets, as follows:
   
   \begin{align*}
   A: & \ x_n = 0, x_n' = 0 \\
   B: & \ x_n = 0, x_n' = 1 \\
   C: & \ x_n = 1, x_n' = 0 \\
   D: & \ x_n = 1, x_n' = 1
   \end{align*}

The neutrality of \( g_n \) and \( h_n \) ensures that the four sets are of equal size; i.e., each contains one-fourth of the total number of states. By definition of the sets, each member state of each set has a successor state in only certain of the other sets. This transition restriction is expressed by the branches on the graph of Fig. 10; e.g., each state in set A or set C \( (x_n = 0) \) is followed by a state in either set A or set B \( (x_n = 0) \). Clearly, subsets of states only in sets B or C cannot be involved in odd-cycles, but each state in sets A and D can be made part of an odd cycle; so the number of odd-cycles is bounded above by \( 2^{n-1} \), the total number of states in sets A and D.

The sufficiency of this condition is tedious to demonstrate. Its proof provides little insight, and is not given.

**Special Cases of Interest**

The C-matrix will now be formulated for several interesting special network configurations, and some preliminary observations will be made regarding the conditions of nonsingularity. Each of these cases amounts to a restriction on the logic of the network, and the effect of this restriction on the state behavior will be described in part.
The Feedback Shift Register

For this case, all of the delay elements are concatenated in a single chain, as shown in Fig. 11. The first element is driven by a single-output combinational network, whose inputs are the other state variables. The next-state equations are therefore

\[ x_n' = x_{n-1} \]
\[ x_{n-1}' = x_{n-2} \]

and all but the top row of the C-matrix may be filled in immediately. E.g., for \( n = 4 \):

\[
C = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1
\end{bmatrix}
\]

The top row is arbitrary unless the condition of nonsingularity is imposed. In this case, it follows from theorem 3 that

\[ x_1' = s_0 x_0 + s_1 x_1 \]
\[ x_1' = s_2 x_2 \]

where \( g_1 \) is independent of \( x_0 \). The characteristic number of \( g_1 \) is given by the left half of the characteristic number of \( x_1 \), and carries no further restriction. A necessary and sufficient condition for a feedback shift register to be nonsingular is therefore that \( x_1' \) depend on \( x_0 \) in accordance with this relation. The network then takes the form of Fig. 12.

If the combinational logic is entirely linear, the register is then a linear feedback shift register, which is known to be a canonical form for all linear nets. It is known that the nonlinear feedback shift register is not a canonical form for nonlinear nets, but there are good indications that it is a very versatile network, capable of producing a wide variety of different cycle-sets.

From the columns of the C-matrix, it is apparent that there are only two choices for the successor of each state. In fact, since the shifting action of the register amounts to a doubling of the binary number which it contains, then state no. \( s \) is followed by either \( 2s \) (modulo-2\(^n\)) if the digit of the top row is a ‘0’, or state no. \( 2s+1 \) (modulo-2\(^n\)) if the digit in the top row is ‘1’. These constraints on the state behavior may be condensed in a “generalized” state graph, shown in Fig. 13 for \( n = 3 \). Here the solid line transitions result from \( g = 0 \), or \( x_1' = x_n \), which is the pertinent equation for a simple loop of \( n \) delay elements, and the dashed-line transitions result from \( g = 1 \), or \( x_1' = x_n \), which is the pertinent equation for a loop of \( n \) delay elements and a single inverter.

Many interesting constraints on the possible cycle-sets of the feedback-shift register can be derived from this view of the state graph as a superposition of two simpler state graphs. These results constitute a separate study and are not reported here.

The Safe-Asynchronous Network

From the definition of a safe-asynchronous network in a previous section, it follows immediately that every column of the C-matrix of a safe-asynchronous network differs in no more than one binary digit from the corresponding column of the identity C-matrix. E.g., the C-matrix

\[
X = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1
\end{bmatrix}
\]

satisfies this condition, with the differing digets underlined. Thus, the binary number of each state agrees in all but one binary digit with the binary number of the successor of that state. The generalized state graph for safe-asynchronous networks is merely the binary \( n \)-dimensional cube, in which vertices (nodes) represent states, and the edges (branches) the permitted transitions. These graphs are shown for \( n = 2, 3, \) and 4 in Fig. 14. (The branches carry no arrows, since transitions in either or both directions are allowed.)

The condition for the realizability of a given state graph in a safe-asynchronous network is now simply stated. If the state graph is otherwise realizable; i.e., if it has \( 2^n \) nodes, and has only one branch emanating from each node, then it is realizable in a safe-asynchronous network if and only if it is embeddable in the generalized graph of the \( n \)-cube. Many necessary conditions for embeddability can be stated in terms of more obvious features of graphs, but sufficient conditions are not known.

If the condition of nonsingularity is imposed, embeddability requires that the resultant cycle set contain even cycles only. This condition is also sufficient for realizability but the proof will not be given here.

The Fully-Self-Independent Network

If each switching function \( x_k' \) is independent of \( x_k \), then the corresponding network will be called fully-self-independent. For the nonsingular case, this condition requires that \( g_k = h_k \) in condition 16 of a preceding section so that

\[ x_k' = g_k \]

where \( g_k \) is independent of \( x_k \) and the result is that the number of odd cycles is bounded by \( 2^{n-1} \). Actually, much stronger conditions of this type follow from extensions of condition 16, and place tight bounds on the multiplicities of short cycles. For \( n = 2 \), the only cycle sets possible are \( (1, 2, 3) \) and \( (4) \), and for \( n = 3 \), only \( (1, 7), (2, 6), \) and \( (1, 3, 9) \).

The Network with Cyclically Permutied Logic

Consider now the network whose next-state switching functions \( x_k' \) are all expressible in terms of a single function \( \phi \):

\[
x_k' = \phi(x_1, x_2, \ldots x_n)
\]

\[
x_k' = \phi(x_1, x_2, \ldots x_n)
\]

Thus, the function remains the same, but its arguments are permuted cyclically with the advance of subscript \( k \) on \( x_k' \). Specification of one row of the C-matrix is now tantamount to specification of the entire matrix. For \( n = 3 \), for example, we may designate the bottom row with the characteristic number \([e_0, e_1, e_2, \ldots e_3]\) and then derive the rest of the matrix from the previous equations:

\[
C = \begin{bmatrix}
e_0 & e_1 & e_2 & e_3 \\
e_0 & e_1 & e_2 & e_3 \\
e_0 & e_1 & e_2 & e_3 \\
e_0 & e_1 & e_2 & e_3
\end{bmatrix}
\]

It is apparent that the \( e \)'s fall into distinct groups by columns: all of the \( e_0 \)'s are in column \( 0 \), \( e_1, e_2, e_3 \) and \( e_4 \) occupy columns 1, 2, and 4; \( e_5 \) occupies columns 3, 6, and 5; and \( e_7 \) is in column 7. These groups are those defined by the cycles of the simple circulating shift register made up of a loop of \( 3 \)-delay elements. This register yields the state graph consisting of the nodes and solid line transitions of Fig. 13, with cycle set \( (1, 3, 9) \) and state-rows \( (0), (7), (1, 2, 4), \) and \( (3, 6, 5) \). That this property is generally true may be easily proved. Indeed, it is not surprising in view of the fact that the simple circulating register itself has cyclically permuted logic: \( x_k' = x_{k-1} \) (\( k = 2, 3, \ldots n \), \( x_1' = x_n \)).

Conditions for nonsingularity and for realizability of arbitrary cycle sets may be derived, based on this approach.

Conclusions

Campeau's C-matrix and the concepts behind it have been applied to the derivation of specific relationships between the
System Evaluation and Instrumentation of a Special-Purpose Data Processing System Using Simulation Equipment

A. J. STRASSMAN  L. H. KURKJIAN

TESTING and instrumentation are essential prerequisites for the completion and operation of any new system. A system can be defined as a number of components that are amalgamated or integrated together to perform a desired operation. Throughout this paper a "component" is considered to be a complete functional part of a data-processing system such as an arithmetic unit or a buffer. To ascertain if a component in the system is going to perform its specific function, it is sometimes necessary for the implementation of tests to be more complex than the component undergoing the testing. This becomes apparent when the component is a part of a large system and has many inputs and outputs.

To prove the system feasibility or operation of the components it is necessary to do either of two things: 1. duplicate and maintain an entire system and use it as one master-test fixture to evaluate each functional component; or, 2. provide individual test facilities for the component is a part of a large system as one master-test fixture to evaluate each functional component. The second approach requires the design of simulation equipment to provide the necessary inputs (control signals and data) to check out completely the operation of each individual component. It is believed that this approach offers the greatest advantages for large special-purpose data processing systems.

It is necessary to provide the proper work organization for the evaluation of these computer systems. A differentiation can be made between small and large systems and the work organization can be adjusted accordingly. Although the basic philosophy of test remains the same, the details evolved for the testing or evaluation of a small system will be different from that evolved for a large system. For the purpose of this paper in which the evaluation and instrumentation of a large system will be described, a "large" system will be defined arbitrarily as one that contains more than 500 flip-flops. Since the flip-flop is a basic part of any digital computer, the number of flip-flops can be used as an indication of the size and complexity of the system.

The philosophies contained within this paper led to the basic planning considerations for the test and evaluation of a special-purpose data processing system; parts of which will be described in later paragraphs. This data processing system contains approximately 1,500 tubes, 2,500 transistors, 40,000 logical gating diodes and 3,500 flip-flops. Each flip-flop in the system has four transistors making a total of 16,500 transistors in the entire system. This qualifies the described system to be classified as a large system.

In the case of a small system, all the

Fig. 1. Standard digital elements and element tester

Fig. 2. Matrix assembly and matrix tester

Strassman, Kurkjian—A Special-Purpose Data Processing System

127