Copyright © 1971 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 50, No. 5, May-June, 1971
Printed in U.S.A.

On a Class of Rearrangeable Switching Networks

Part I: Control Algorithm

By D. C. OPFERMAN and N. T. TSAO–WU

(Manuscript received December 1, 1970)

An algorithm is developed to control a class of rearrangeable switching networks, particularly with the base-2 structure. Various methods of implementing this algorithm are also described. System organization and processing time for rearranging the network are studied and are shown to be practical.

I. INTRODUCTION

One type of a switching network which has drawn considerable interest lately is the class of rearrangeable switching networks (RSN). With these networks, any idle input terminal of the network can always be connected to any idle output terminal by rerouting the existing connections if necessary. These networks can be used where one-to-one full access and nonblocking features are required, and rerouting is feasible, e.g., main distribution frames[1] and facility switches[2] in telephone systems and data transfer networks in a multiprocessor computer system.[3]

Most of the earlier efforts, notably by C. Clos,[4] V. E. Beneš,[5] and A. E. Joel, Jr.,[6] have been made in the context of telephone switching networks. Their emphasis has been on the network structure, on its combinatorial properties and on bounds on the number of connections that require rerouting. Recently, this type of network has been of interest in such computer areas as data-sorting systems[7] and self-repairing multiprocessors.[3] The network structure is also applicable for cellular arrays.[8] However, very few reports[9,10] have been made on the control aspect of these networks.

This paper will begin with a brief discussion of the general structure of RSN's, followed by the development of a method for the control of these networks and its practical implementation. The relationship between the network structure and the ease (or difficulty) with which it can be controlled will also be discussed.

II. THE NETWORK STRUCTURE

Discussion will be limited to a class of rearrangeable switching networks connecting N input terminals and N output terminals [abbreviated as $(N \times N)$ networks]. Extension to the more general case for $(N \times M)$ networks, $N \neq M$, can be readily made.

Let $N = dq$, where d and q are integer factors of N. A $(N \times N)$ network can be decomposed into an input stage and an output stage having altogether $2N/d$ $(d \times d)$ networks (one of which can be eliminated) and a middle stage having d $(N/d \times N/d)$ networks, as shown in Fig. 1. This network is said to have a base-d structure, and the smaller networks are called subnetworks. This type of network structure falls into the general class considered by Clos and Beneš.

The network with base-2 structure is of great importance, for two primary reasons. First, it yields the most efficient network (or the least number of two-state switching elements) and, secondly, its control is relatively simple. It consists of (2×2) networks in the input and output stages, altogether $(N - 1)$ in number where N is even or odd. Two $(N/2 \times N/2)$ networks are in the middle stage if N is even, or one $[(N-1)/2 \times (N-1)/2]$ network and one $[(N+1)/2 \times (N+1)/2]$ network are in the middle stage if N is odd, as shown in Fig. 2a and b. Clearly, further decomposition of the networks in the middle stage is possible, and if the base-2 structure is carried throughout, one may show by an iterative process that the total number of basic switch-

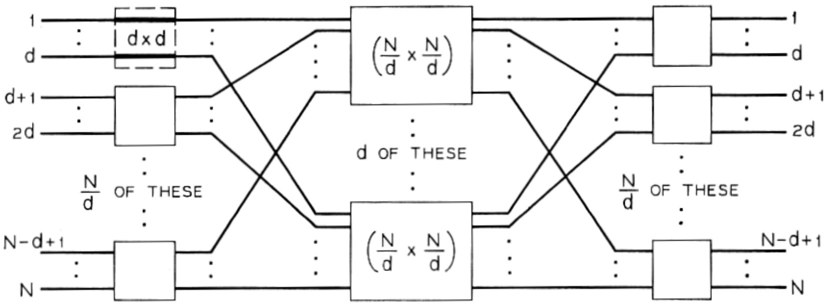

Fig. 1—Rearrangeable $(N \times N)$ network of a general base-d structure.

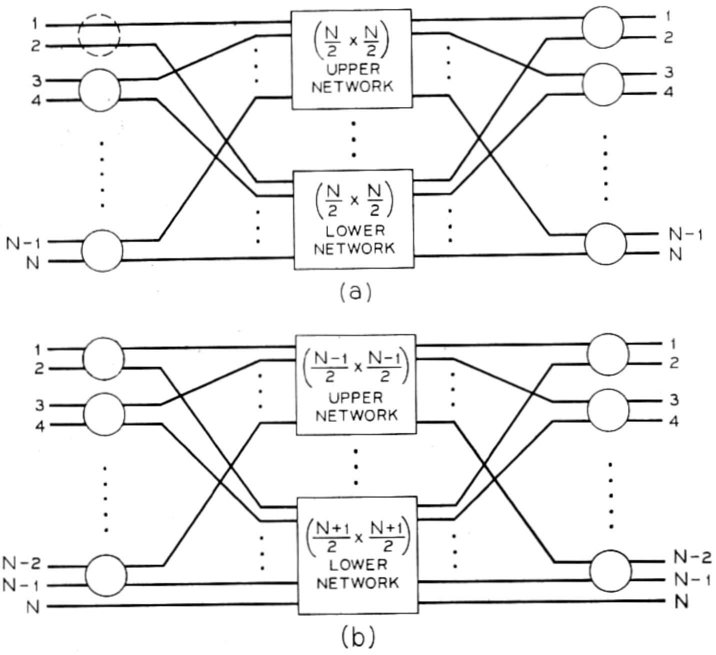
1
2
3
4
N-1
N
$\left(\frac{N}{2} \times \frac{N}{2}\right)$
UPPER NETWORK
$\left(\frac{N}{2} \times \frac{N}{2}\right)$
LOWER NETWORK
(a)
N-2
$\left(\frac{N-1}{2} \times \frac{N-1}{2}\right)$
$\left(\frac{N+1}{2} \times \frac{N+1}{2}\right)$
(b)

Fig. 2—An $(N \times N)$ network with base-2 structure for: (a) N even; (b) N odd.

ing elements or (2×2) networks, named β-elements by Joel,[6] is given by*

$$N\langle \log_2 N \rangle - 2^{\langle \log_2 N \rangle} + 1.$$

It can easily be seen that the number of these β-elements is bounded by $\langle \log_2 N! \rangle$. A (11×11) network consisting entirely of β-elements is shown in Fig. 3. It is a very efficient network, since there are 29 such elements and one must have 26 $(> \log_2(11!) > 25)$ two-state devices to accommodate all possible permutations. Some of the enumeration studies given in Part II will account for the additional states.

III. THE NETWORK CONTROL

The control algorithm is first developed for the general $(N \times N)$ network, having a base-d structure, as it is shown in Fig. 1. The special case where $d = 2$ will then be considered, and practical implementations of the control algorithm will be given in Section IV. First, some definitions are needed.

* $\langle z \rangle$ is the smallest integer greater than z.

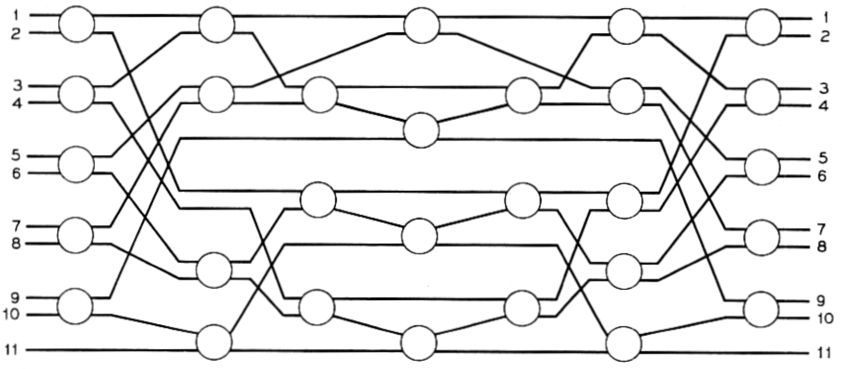

Fig. 3—An (11 × 11) network with base-2 structure.

3.1 *Definitions*

(*i*) An input-output pair $\binom{x}{\pi(x)}$ defines a connection between input terminal x and output terminal $\pi(x)$ of a $(N \times N)$ network, where $1 \leqq x \leqq N$, $1 \leqq \pi(x) \leqq N$

(*ii*) A connection set C is a collection of m such input-output pairs, $m \leqq N$, expressed as follows:

$$C = \begin{pmatrix} x_1 & x_2 & \cdots & x_m \\ \pi(x_1) & \pi(x_2) & \cdots & \pi(x_m) \end{pmatrix}.$$

If $m = N$, C is then in the form of the familiar permutation. This may be denoted by P, with $C \subseteq P$, describing all the connections (or the traffic pattern) through the $(N \times N)$ network.

(*iii*) Let $I = \{1, 2, \cdots, N\}$ be a set of positive distinct integers; a subset $J(l, d)$ is defined by

$$J(l, d) = \left\{a \mid \left[\frac{a + d - 1}{d}\right]^* = l\right\}, \quad \text{and} \quad a \in I$$

where d divides N, and l is some constant integer, $1 \leqq l \leqq N/d$. $J(l, d)$ is called an integer set of order d and of characteristic l, and any element belonging to it is said to have characteristic l relative to the base-d. This is merely a formal way of grouping all those terminals associated with the same subnetwork in the input (or output) stage.

(*iv*) A connection set C having m input-output pairs is said to be

* $[w]$ denotes the integral value of w.

reducible if and only if on replacing every integer by its characteristic, C becomes a permutation on m distinct integers.

3.2 *The Generalized Control Algorithm*

Any given set of connections through the network can be described as:

$$P = \begin{pmatrix} x_1 & x_2 & \cdots & x_N \\ \pi(x_1) & \pi(x_2) & \cdots & \pi(x_N) \end{pmatrix}.$$

The objective of the control algorithm is to derive from P permutations to be realized by each of the subnetworks. This is accomplished by first decomposing P into reducible connection sets $C_1, C_2, \cdots, C_d$. Permutations for the middle subnetworks are defined by using the characteristics of the elements in these sets, and permutations for the input and output stages are determined directly from these elements.

3.2.1 *To Decompose a Permutation into Reducible Connection Sets*

Let the output terminals be partitioned into sets denoted by S_l where

$$S_l = \{\pi(x_i) \mid x_i \in J(l, d)\}, \qquad 1 \leqq l \leqq \frac{N}{d}.$$

The reducible connection sets C_i, $1 \leqq i \leqq d$, are constructed by grouping N/d input-output pairs of which the output integers are selected, one from each S_l, such that no two output integers have the same characteristic.

Let C_i be expressed as

$$C_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,N/d} \\ \pi(x_{i,1}) & \pi(x_{i,2}) & \cdots & \pi(x_{i,N/d}) \end{bmatrix}, \qquad 1 \leqq i \leqq d;$$

when it is reduced, one has the permutation

$$P_i = \begin{pmatrix} p_{i,1} & p_{i,2} & \cdots & p_{i,N/d} \\ \pi(p_{i,1}) & \pi(p_{i,2}) & & \pi(p_{i,N/d}) \end{pmatrix}$$

where $p_{i,j}$ and $\pi(p_{i,j})$ are the characteristics of the integers $x_{i,j}$ and $\pi(x_{i,j})$, respectively, $1 \leqq i \leqq d$, $1 \leqq j \leqq N/d$. Each of these d permutations P_i can be realized by any of the d $(N/d \times N/d)$ subnetworks. However, if each C_i is assigned to a particular $(N/d \times N/d)$ subnetwork, one of the $(d \times d)$ subnetworks, say the one at the upper-left corner of the input stage (Fig. 1), can be eliminated. This elimination implies that the connection set C_i, and hence P_i, is assigned to the first $(N/d \times N/d)$ subnetwork in the middle stage if and only if $x_{i,j} = 1$ for some j, $1 \leqq j \leqq N/d$. In general, C_i, and hence P_i, is assigned to the k^{th} $(N/d \times N/d)$ subnetwork in the middle stage, $1 \leqq k \leqq d$, (counting from the top) if and only if $x_{i,j} = k$ for some j, $1 \leqq j \leqq N/d$. One may then reorder the indices i such that C_i contains the input-output pair $\begin{bmatrix} x_{i,j=i} \\ \pi(x_{i,j}) \end{bmatrix}$, and it follows that the permutations $P_1, P_2, \cdots, P_d$ are similarly ordered for the d $(N/d \times N/d)$ subnetworks in the middle stage.

3.2.2 *To Obtain Permutations for Subnetworks in the Input and Output Stages*

Let $P_{I,1}, P_{I,2}, \cdots, P_{I,N/d}$ and $P_{0,1}, P_{0,2}, \cdots, P_{0,N/d}$ be the sets of permutations to be realized by the 1st, 2nd, $\cdots$, $(N/d)^{\text{th}}$ $(d \times d)$ subnetworks in the input and output stage, respectively. Moreover, let the reducible connection set C_i be written as

$$C_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,N/d} \\ \pi(x_{i,1}) & \pi(x_{i,2}) & \cdots & \pi(x_{i,N/d}) \end{bmatrix}, \qquad 1 \leqq i \leqq d$$

such that $x_{i,1} < x_{i,2} \cdots < \cdots < x_{i,N/d}$.

Then, from the network structure, one can see simply that the permutations

$$P_{I,j} = \begin{pmatrix} \pi_j^{-1}(1) & \pi_j^{-1}(2) & \cdots & \pi_j^{-1}(d) \\ 1 & 2 & \cdots & d \end{pmatrix}$$

and

$$P_{0,k} = \begin{pmatrix} 1 & 2 & \cdots & d \\ \pi_k(1) & \pi_k(2) & \cdots & \pi_k(d) \end{pmatrix}$$

can be obtained from C_i with

$$\pi_j^{-1}(a) = x_{a,j} - (j-1)d, \quad 1 \leqq j \leqq N/d, \quad 1 \leqq a \leqq d$$

where $\pi^{-1}(a)$ denotes the input terminal to be connected to the output terminal a, and $\pi_k(a) = \pi(x_{a,t}) - (k-1)\,d$, $1 \leqq k \leqq N/d$, for some t such that $\pi(x_{a,t}) \in J(k, d)$.

3.2.3 *An Example on Decomposing a Permutation*

Consider a (15×15) network, having a base $d = 5$ structure. Such a network is shown in Fig. 4, with two (5×5) subnetworks in the input stage, three (5×5) subnetworks in the output stage, and five

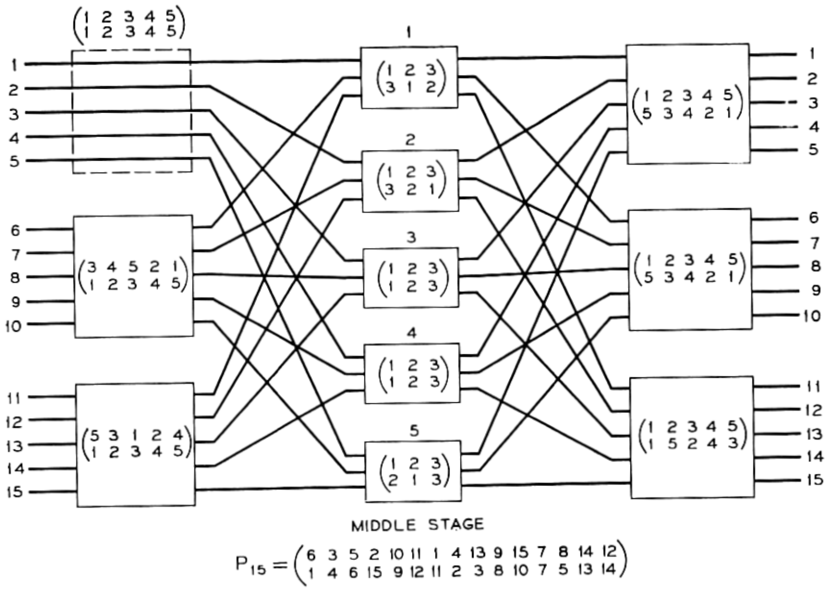
$\begin{pmatrix}1&2&3&4&5\\1&2&3&4&5\end{pmatrix}$
$\begin{pmatrix}3&4&5&2&1\\1&2&3&4&5\end{pmatrix}$
$\begin{pmatrix}5&3&1&2&4\\1&2&3&4&5\end{pmatrix}$
1 $\begin{pmatrix}1&2&3\\3&1&2\end{pmatrix}$
2 $\begin{pmatrix}1&2&3\\3&2&1\end{pmatrix}$
3 $\begin{pmatrix}1&2&3\\1&2&3\end{pmatrix}$
4 $\begin{pmatrix}1&2&3\\1&2&3\end{pmatrix}$
5 $\begin{pmatrix}1&2&3\\2&1&3\end{pmatrix}$
$\begin{pmatrix}1&2&3&4&5\\5&3&4&2&1\end{pmatrix}$
$\begin{pmatrix}1&2&3&4&5\\5&3&4&2&1\end{pmatrix}$
$\begin{pmatrix}1&2&3&4&5\\1&5&2&4&3\end{pmatrix}$
MIDDLE STAGE
$P_{15}=\begin{pmatrix}6&3&5&2&10&11&1&4&13&9&15&7&8&14&12\\1&4&6&15&9&12&11&2&3&8&10&7&5&13&14\end{pmatrix}$

Fig. 4—A (15×15) network with permutations assignment.

(3×3) subnetworks in the middle stage. Let the connections through the network be described by the following permutation:

$$P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 11 & 15 & 4 & 2 & 6 & 1 & 7 & 5 & 8 & 9 & 12 & 14 & 3 & 13 & 10 \end{pmatrix}.$$

The output integers are partitioned as follows:

$$S_1 = \{11, 15, 4, 2, 6\},$$
$$S_2 = \{1, 7, 5, 8, 9\},$$
$$S_3 = \{12, 14, 3, 13, 10\}.$$

From these, the reducible connection sets C_1, C_2, C_3, C_4, and C_5 are constructed as shown in Table I and, in general, they are not unique. The corresponding permutations (P_i), which are also shown in Table I, are obtained from C_i by replacing each element with its characteristic. Note that connection sets are ordered according to the input integers (1, 2, 3, 4, 5) and that $x_{i,1} < x_{i,2} < x_{i,3}$ for each i. Table II shows the permutations derived from C_i for subnetworks in the input and output stages using the relations for $\pi_j^{-1}(a)$ and $\pi_k(a)$ as given in Section 3.2.2.

TABLE I—REDUCED CONNECTION SETS AND THEIR CORRESPONDING PERMUTATIONS

i	C_i	P_i
1	$\begin{bmatrix} 1 & 8 & 15 \\ 11 & 5 & 10 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$
2	$\begin{bmatrix} 2 & 9 & 13 \\ 15 & 8 & 3 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$
3	$\begin{bmatrix} 3 & 10 & 11 \\ 4 & 9 & 12 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$
4	$\begin{bmatrix} 4 & 7 & 12 \\ 2 & 7 & 14 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$
5	$\begin{bmatrix} 5 & 6 & 14 \\ 6 & 1 & 13 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$

TABLE II—PERMUTATIONS FOR SUBNETWORKS IN THE INPUT AND OUTPUT STAGES

j	$P_{I,j}$	$P_{O,j}$
1	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 4 & 2 & 1 \end{pmatrix}$
2	$\begin{bmatrix} 3 & 4 & 5 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 4 & 2 & 1 \end{pmatrix}$
3	$\begin{bmatrix} 5 & 3 & 1 & 2 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 2 & 4 & 3 \end{pmatrix}$

IV. THE CONTROL ALGORITHM FOR THE BASE-2 NETWORK

The selection of output integers from the set S_l, $1 \leqq l \leqq N/d$, to form the connection sets C_i, $1 \leqq i \leqq d$, is by no means simple. One procedure has been reported by V. I. Neiman.[10] By modifying the previous example, it can be shown that the selection can *not* be made on a strictly sequential basis. Let the given permutation be modified such that the sets S_1, S_2, and S_3 are as follows:

$$S_1 = \{11, 10, 4, 2, 6\},$$
$$S_2 = \{1, 7, 5, 8, 9\},$$
$$S_3 = \{12, 14, 3, 13, 15\}.$$

If one had chosen $\binom{1}{11}$ and $\binom{6}{1}$ input-output pairs to form C_1, where the output integers 11 and 1 are from S_1 and S_2 respectively, there is no output integer in S_3 which would have a characteristic different from that of the integer 11 or 1. Thus, in general, simultaneous selections must be made in the construction of the connection sets. For networks with base-2 structure, however, the selection is reduced to a mere binary choice, resulting in a much simpler algorithm. For the other extreme case,[11] where $d = N/2$, the difficulty described above will not arise because there are only two sets S_1 and S_2, each containing exactly d integers.

For a $(N \times N)$ network with a base-2 structure, the control algorithm for setting the β-elements to realize a given permutation P consists of three parts: (*i*) decomposing of P into reducible connection sets C_1 and C_2; (*ii*) reducing C_1 and C_2 to P_1 and P_2 respectively; and (*iii*) setting the β-elements in the input and output stages. Since the network has an iterative structure, the same procedure is applied to each of the $(N/2 \times N/2)$, $(N/4 \times N/4)$, $\cdots$, (2×2) subnetworks. There are $\log_2 N$ levels of an $(N \times N)$ network with the last level of (2×2) subnetworks being a trivial case, assuming N is a power of 2.

A coding scheme for the input and output integers which facilitates the required operations and two methods for decomposing P are described in the following sections.

4.1 *Coding Scheme*

It is clear from what has been discussed so far that the control algorithm essentially accepts the connection requirements as input data and, after processing, generates a set of output data which are used to rearrange the network. It is necessary, therefore, to have an input/output (I/O) memory, which stores the output terminal $\pi(x_i)$ at the address determined by the numerical value of the input terminal x_i. A simple coding scheme for these integers proves to simplify the implementation of the algorithm.

Referring again to the network, one can, of course, use the set of integers $(0, 1, 2, \cdots, N - 1)$ to number the input and output terminals, without loss of generality in all the previous discussions.* Then, the familiar binary code can be used directly, both for the input x_i as address and for the output $\pi(x_i)$ as the contents at x_i. We shall now show how this code can be used at all $\log_2 N$ levels. Let the binary representation be

$$b_{n-1}b_{n-2} \cdots b_1 b_0 ,$$

where $n = \log_2 N$, and assuming N a power of 2. Beginning at the first level of the network, the least significant bit of the address is used to set the input β-element defined by the remaining $n - 1$ bits, and that of the contents to set the output β-element defined by the remaining $n - 1$ bits of the contents. Moreover, the conversion from C_1 (or C_2) to P_1 (or P_2) is accomplished by merely eliminating b_0 for each coded output integer. Finally, for an output integer $\pi(x_i)$, the bit b_0 is set to identify whether the particular $\pi(x_i)$ belongs to P_1 or P_2. However, for an input integer, the most significant bit of the address, b_{n-1}, indicates whether x_i belongs to P_1 or P_2.

The same coding procedure is applied at each subnetwork, and the I/O memory is partitioned (part of algorithm) in the appropriate manner. In general, at the i^{th} level of the network, $i < \log_2 N$, the $(i-1)$ least significant bits of a word in memory define the particular subnetwork of size $N(2^{1-i})$, and the $\log_2 N - (i - 1)$ most significant bits define the output integer. The $(i-1)$ most significant bits, however, of the address designate the subnetwork, and the remaining bits define the input integer. An example will be given in detail to illustrate this in Section 4.2.1.

4.2 *Decomposition by Looping*

With $d = 2$, an integer set is reduced to an integer pair, consisting of only two elements, and one is said to be the dual of the other. If one continues to use the integers $(0, 1, \cdots, N - 1)$ to number the input and output terminals of an $(N \times N)$ network, then the integers a and b constitute an integer pair if

$$\left\lfloor \frac{a}{2} \right\rfloor = \left\lfloor \frac{b}{2} \right\rfloor = l$$

for some integer l. The dual of a (or b) is denoted by $\hat{a}$ (or $\hat{b}$), and,

* Except that the definition of the integer set $J(l, d)$ needs to be slightly modified, as follows:

$$J(l, d) = \{a | [a/d] = l\}, \qquad 0 \leqq l \leqq (N/d) - 1$$

therefore,

$$\hat{a} = b \quad \text{and} \quad \hat{b} = a.$$

Moreover, any permutation is decomposed into only two reducible connection sets C_1 and C_2, and if $\left[\begin{smallmatrix} x \\ \pi(x) \end{smallmatrix}\right] \in C_1$, then $\left[\begin{smallmatrix} \hat{x} \\ \pi(\hat{x}) \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} x_i \\ \pi(x_i) \end{smallmatrix}\right] \in C_2$ for some x_i, where $\pi(x_i)$ is the dual of $\pi(x)$. In coded form, the dual is obtained merely by complementing the least significant bit.

One method, called the looping procedure, of constructing C_1 and C_2 from P is to search the I/O memory for the required outputs. The sequence starts by selecting the output $\pi(0)$ in the first location of the memory (or $\left[\begin{smallmatrix} 0 \\ \pi(0) \end{smallmatrix}\right]$). The first $(n - 1)$ bits $(a_{n-1}a_{n-2} \cdots a_1)$ of the address which are all zeros define the first β-element (β_{i1}) in the input stage, and the last bit (a_0) which is also zero defines β_{i1} to be set to the "straight through" state (see Fig. 5). The bits $m_{n-1}m_{n-2} \cdots m_1$ and m_0 at this address define one of the β-elements (β_{oi}) in the output stage and its setting respectively. Bit m_0 is now reset to zero to designate that this particular output $(m_{n-1}m_{n-2} \cdots m_1)$ is for P_1. There is also another bit m_n which is set to one when that particular output has been placed in P_1 or P_2 so that an unused output can be selected when it is necessary. The example in Section 4.2.1 will clarify this.

The memory is next scanned for the dual of $\pi(0)$; this output and its address x_k define $\left[\begin{smallmatrix} x_k \\ \pi(x_k) \end{smallmatrix}\right]$ for C_2. For the input-output pairs in C_2

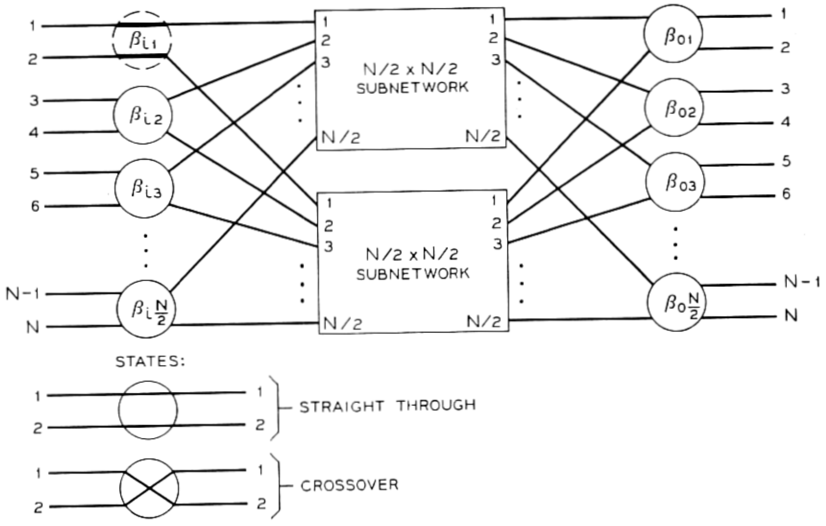

Fig. 5—Base-2 $(N \times N)$ rearrangeable switching network.

it is only necessary to set m_n and m_0 of the memory word to '1' to signify that this word (output) has been used and that it is for P_2. The output $\pi(\hat{x}_k)$ at the address $\hat{x}_k$ defines $\binom{\hat{x}_k}{\pi(\hat{x}_k)}$ and is designated for C_1. The same looping procedure is continued until all $\pi(x_i)$, $1 \leqq i \leqq N$ are assigned to either C_1 or C_2. In most cases, however, the looping will end before all output integers are used. The procedure is started again by arbitrarily selecting an unassigned output for P_1, by examining m_n. Because of this arbitrariness, C_1 is, in general, not unique. This fact is used to advantage in the other procedure to be described in Section 4.3.

After the looping procedure is completed, the memory is reorganized to have P_1 and P_2 in locations 0 to $N/2 - 1$ and $N/2$ to $N - 1$, respectively. (A small scratch pad memory may be necessary.) Then the same procedure is applied to each of these $(N/2 \times N/2)$ subnetworks, and it is continued until all of the β-elements in the $(N \times N)$ network are set.

The searching can be eliminated by employing two memories, one of which is the I/O memory. The additional one is an output/input (O/I) memory that stores the input x_i at the address corresponding to the numerical value of the output $\pi(x_i)$. The decomposition of P into P_1 and P_2 is achieved by crisscrossing between the two memories. For example, output $\pi(x_i)$ and its corresponding address x_i in the I/O memory define $\binom{x_i}{\pi(x_i)}$ for C_1. Now the dual of $\pi(x_i)$, say $\pi(x_j)$, is the address for the O/I memory and the corresponding word x_j defines $\binom{x_j}{\pi(x_j)}$ for C_2. Then $\hat{x}_j$ is used for the address in the I/O memory. If the I/O memory is a content addressable memory (CAM),[12] the required $\pi(x_i)$'s for C_1 and C_2 are determined directly in the content addressable mode, without the use of a second memory.

4.2.1 *An Example of the Looping Procedure*

In order to illustrate in a meaningful way the looping procedure using the coding scheme at various levels, a permutation for a (32×32) network, as given in Table III, will be utilized.

The looping procedure begins with $\pi(0) = 14$ and continues to select input-output pairs for the connection set C_1. (The sequence of this selection is indicated by the number in the "looping sequence" column.) As each output integer is selected for C_1, the last bit m_0 is used to set the output β-element designated by $(m_4m_3m_2m_1)$ (see Fig. 6). Then the bit m_0 is set to '0' to indicate that it is for C_1, and the bit m_5 is set to '1' to indicate that it has been used. The bits m_0 and m_5 of the output integers for C_2 are merely set to '1'. In this particular

TABLE III—MEMORY CONTENTS ON THE INTERCONNECTIONS

Input Terminals	Outputs Terminals	Coded Output Integers m_5	m_4	m_3	m_2	m_1	m_0	Looping Sequence
0	14	0	0	1	1	1	0	1
1	23	0	1	0	1	1	1	
2	3	0	0	0	0	1	1	7
3	20	0	1	0	1	0	0	
4	28	0	1	1	1	0	0	8
5	2	0	0	0	0	1	0	
6	16	0	1	0	0	0	0	
7	11	0	0	1	0	1	1	
8	31	0	1	1	1	1	1	9
9	29	0	1	1	1	0	1	
10	1	0	0	0	0	0	1	
11	27	0	1	1	0	1	1	5
12	12	0	0	1	1	0	0	3
13	18	0	1	0	0	1	0	
14	5	0	0	0	1	0	1	
15	24	0	1	1	0	0	0	11
16	26	0	1	1	0	1	0	
17	21	0	1	0	1	0	1	6
18	0	0	0	0	0	0	0	4
19	13	0	0	1	1	0	1	
20	8	0	0	1	0	0	0	
21	6	0	0	0	1	1	0	13
22	19	0	1	0	0	1	1	2
23	15	0	0	1	1	1	1	
24	4	0	0	0	1	0	0	10
25	30	0	1	1	1	1	0	
26	7	0	0	0	1	1	1	
27	22	0	1	0	1	1	0	14
28	17	0	1	0	0	0	1	
29	10	0	0	1	0	1	0	
30	25	0	1	1	0	0	1	
31	9	0	0	1	0	0	1	12

example, the looping procedure ends prematurely and leaves the connection set

$$\begin{pmatrix} 6 & 7 & 28 & 29 \\ 16 & 11 & 17 & 10 \end{pmatrix}$$

as indicated on Table IV.

One then repeats the looping procedure by starting arbitrarily at some remaining output integers (as indicated by $m_5 = 0$). After every output integer has been assigned to C_1 (or C_2), one rearranges the memory such that all the integers with $m_0 = 0$ occupy the upper half of the memory, corresponding to the connections through the upper network, with the remaining output integers for the lower network. This is shown in Table V.

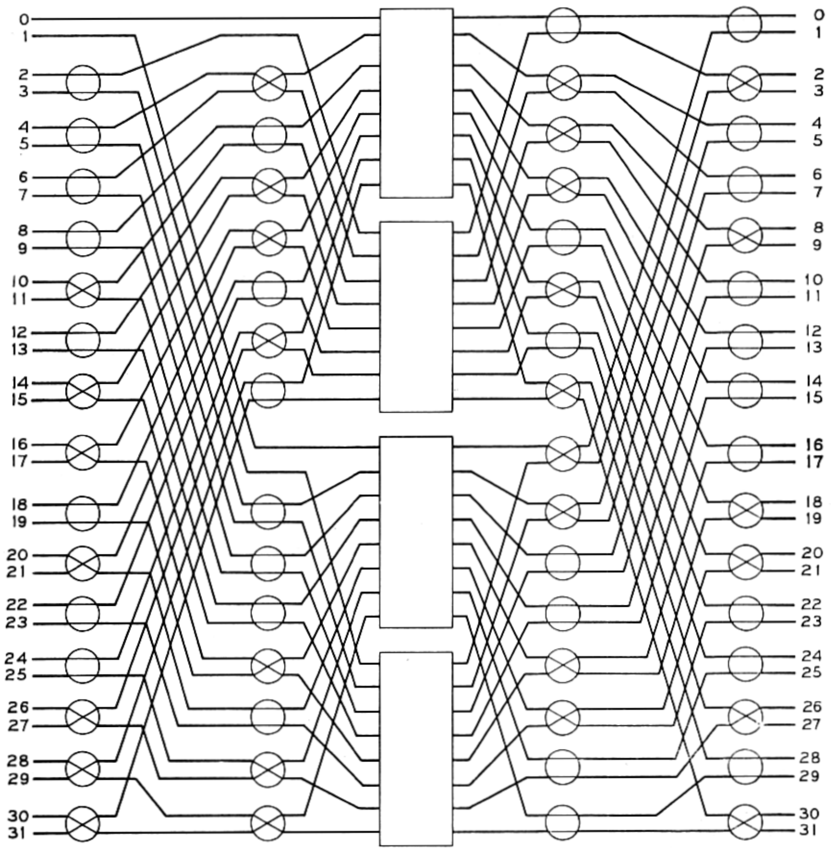

Fig. 6—A (32 $\times$ 32) rearrangeable network with partial setting of β-elements.

Table VI shows the memory contents after another looping procedure is applied to the output integers (for both the upper and lower networks) and subsequent rearranging. At this level, $m_1 m_0$ denotes the (8 $\times$ 8) subnetwork, and if one reverses the order to $m_0 m_1$, it is just the binary representation of natural ascending numbers, 0 being the upmost (8 $\times$ 8) subnetwork and 3 being the lowest (8 $\times$ 8) subnetwork. One could also obtain the same information from the addresses, since the contents are rearranged into this order.

4.3 *Decomposition with a Trial-Partition Procedure*

A second method for decomposing P which incorporates a trial-and-error procedure is presented. Although this method is practical only for small networks, it is very important because of the simple implementation, and intelligence can be included to reduce the average processing time by taking advantage of the fact that the decomposition may not be unique.

From the derivation of the reducible connection sets C_1 and C_2, it is seen that they must contain one and only one $\pi(x_i)$ from each S_l, $0 \leqq l \leqq (N/2) - 1$, and the corresponding input x_i. Since $\begin{bmatrix} 0 \\ \pi(0) \end{bmatrix}$ and $\begin{bmatrix} 1 \\ \pi(1) \end{bmatrix}$ are defined to be in C_1 and C_2, respectively, only $(N/2 - 1)$ additional input-output pairs must be selected for C_1; the remaining pairs are for C_2. After C_1 and C_2 are determined, P_1 and P_2 and the rearranging of the I/O memory are derived in the same manner as in the looping procedure. Let Ψ be the set of connection sets; each includes $(N/2 - 1)$ input-output pairs formed by having one $\pi(x_i)$ from

TABLE IV—MEMORY CONTENTS AFTER SEQUENCING THROUGH ONE LOOP

m_5	m_4	m_3	m_2	m_1	m_0	
1	0	1	1	1	0	
1	1	0	1	1	1	
1	0	0	0	1	0	
1	1	0	1	0	1	these belong
1	1	1	1	0	0	to another loop
1	0	0	0	1	1	
0	1	0	0	0	0	→ 1 1 0 0 0 0
0	0	1	0	1	1	→ 1 0 1 0 1 1
1	1	1	1	1	0	
1	1	1	1	0	1	
1	0	0	0	0	1	
1	1	1	0	1	0	
1	0	1	1	0	0	
1	1	0	0	1	1	
1	0	0	1	0	1	
1	1	1	0	0	0	
1	1	1	0	1	1	
1	1	0	1	0	0	
1	0	0	0	0	0	
1	0	1	1	0	1	
1	0	1	0	0	1	
1	0	0	1	1	0	
1	1	0	0	1	0	
1	0	1	1	1	1	
1	0	0	1	0	0	
1	1	1	1	1	1	
1	0	0	1	1	1	
1	1	0	1	1	0	
0	1	0	0	0	1	→ 1 1 0 0 0 1
0	0	1	0	1	0	→ 1 0 1 0 1 0
1	1	1	0	0	1	
1	0	1	0	0	0	

TABLE V—MEMORY CONTENTS AFTER REARRANGING

m_5	m_4	m_3	m_2	m_1	m_0	
0	0	1	1	1	0	
0	0	0	0	1	0	
0	1	1	1	0	0	
0	1	0	0	0	0	
0	1	1	1	1	0	
0	1	1	0	1	0	
0	0	1	1	0	0	upper
0	1	1	0	0	0	subnetwork
0	1	0	1	0	0	
0	0	0	0	0	0	
0	0	0	1	1	0	
0	1	0	0	1	0	
0	0	0	1	0	0	
0	1	0	1	1	0	
0	0	1	0	1	0	
0	0	1	0	0	0	
0	1	0	1	1	1	
0	1	0	1	0	1	
0	0	0	0	1	1	
0	0	1	0	1	1	
0	1	1	1	0	1	
0	0	0	0	0	1	lower
0	1	0	0	1	1	subnetwork
0	0	0	1	0	1	
0	1	1	0	1	1	
0	0	1	1	0	1	
0	0	1	0	0	1	
0	0	1	1	1	1	
0	1	1	1	1	1	
0	0	0	1	1	1	
0	1	0	0	0	1	
0	1	1	0	0	1	

each S_l, $l > 0$. Since there are two elements in each S_l, Ψ consists of $2^{(N/2)-1}$ connection sets. For any arbitrarily selected $C \in \Psi$, the test will be only on the output integers. Therefore, C defines C_1 if and only if for any $\pi(x_i) \in C$, its dual is not in C.

These ideas lead to the application of a finite state machine with $2^{(N/2)-1}$ states which are used to generate Ψ. This machine, called the Trial-Partition Machine (TPM), is composed of $(N/2 - 1)$ two-state storage devices (flip-flops), each of which represent an input integer pair. The "0" or "1" state of the flip-flop designates that the odd or even input, respectively, of the input integer pair and the corresponding output is in C. If σ_i is a state of the TPM, then it defines C, and the $\pi(x_i) \in C$ are tested for an integer pair (two outputs with the same characteristic). If C contains no pairs, then it is reducible and can be used as C_1. However, if there is at least one output integer pair in C, the TPM is advanced to σ_{j+1}. Since the outputs are serially stored in the I/O memory, the memory must be sequenced in order to perform the test for each σ_j.

This type of a TPM can be easily implemented with a $(N/2 - 1)$ bit binary counter. However, it may be desired to have a more intelligent machine so that C_1 can be determined in less time (with fewer trials). Of course, the complexity and, consequently, the cost of the TPM will increase as the intelligence of the machine grows. One way to

TABLE VI—MEMORY CONTENTS AT THE THIRD LEVEL (AFTER TWO LOOPING PROCEDURES)

m_5	m_4	m_3	m_2	m_1	m_0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	1	0	0
0	1	1	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	1	0	1	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	1	1	1	1	0
0	1	1	0	1	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	1	0	1	0	1
0	0	0	0	0	1
0	1	1	1	0	1
0	1	0	0	0	1
0	0	1	1	0	1
0	0	1	0	0	1
0	0	0	1	0	1
0	1	1	0	0	1
0	1	0	1	1	1
0	0	1	0	1	1
0	0	0	0	1	1
0	0	0	1	1	1
0	1	1	0	1	1
0	0	1	1	1	1
0	1	1	1	1	1
0	1	0	0	1	1

enhance the intelligence is to count the number of output integer pairs in C. This number (Z) specifies how many $\begin{bmatrix} x_i \\ \pi(x_i) \end{bmatrix}$ must be changed in order to have all $\pi(x_i)$ from different output integer pairs. Then the next set (C') is selected from those sets which contain at least Z different pairs and have not been previously tested. Another method is to observe the input-output pairs $\begin{bmatrix} x_i \\ \pi(x_i) \end{bmatrix}$ and $\begin{bmatrix} x_j \\ \pi(x_j) \end{bmatrix}$ that form an output integer pair, and select a C' that contains either $\hat{x}_i$ or $\hat{x}_j$. For an example of a TPM, see Ref. 11.

The TPM is applicable only for small networks because the number of tests becomes prohibitive as N increases. There are $2^{(N/2)-1}$ connection sets in Ψ and, on the average, half of these must be tried. A TPM with intelligence, however, will reduce the number of sets to be tested, but for networks larger than (64×64) this will still be too time consuming.

4.4 *Using Combinational Logic*

The control algorithm for the (4×4) network can be implemented with combinational logic. This is achieved by assigning the states of the five β-elements for each of the 24 possible input-output permutations. The combinational logic, which consists of 13 NAND gates, determines the correct setting from the outputs which are coded in the binary code. This method is practical only for very small networks because the number of permutations grows very rapidly. The (4×4) network is important, however, because it could be used as a building block to construct larger networks, and the combinational logic and the β-elements could be on the same semiconductor chip. This same idea also applies to an (8×8) or (16×16) network using the TPM.

4.5 *System Description*

The block diagram of a rearrangeable switching system with only one memory (I/O Memory) is shown in Fig. 7. The Network Control realizes the control algorithm by employing one or more of the above methods. For example, it may be advantageous to use the combinational logic and TPM for the small networks (or subnetworks) because they are relatively inexpensive to implement. However, as the size of the network grows, the processing time becomes critical. Then the looping procedure with one and two memories (or a content addressable memory) may be used for less than 1000 terminals and more than 1000 terminals, respectively. The timing considerations for these methods are discussed in the next section.

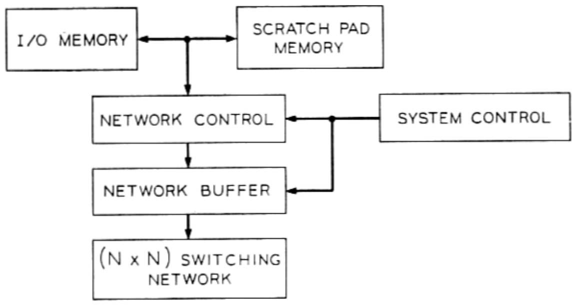

Fig. 7—System block diagram.

The Scratch Pad Memory temporarily stores the outputs during decomposition of P and also while the I/O Memory is being partitioned. The System Control generates the timing and control sequences for all of the operations. This unit, as well as the Network Control, could be implemented with stored program techniques if it is economical and if there is sufficient real time.

The other important unit, called the Network Buffer, isolates the network from the Network Control so that the existing traffic will not be affected during the processing time of the control algorithm. As the settings of the β-elements are determined, they are stored in the Network Buffer. After the control algorithm is completed, the states of the β-elements are set within the time required to insure the quality of the transmission. This time is dictated by the switching transients of the β-elements, network terminations, and the application. Serial shift-registers can provide economical buffers, if the β-elements can be set in a stage-by-stage sequence.

V. TIMING CONSIDERATIONS

In this section, the processing time and the necessary equipment for the various methods of implementing the control algorithm are discussed. The processing time for the algorithm must be sufficiently short to accommodate the traffic changes. The combinational logic method is the fastest; however, it is applicable only for very small networks. The TPM has the next order of complexity, and it is applicable for (64 $\times$ 64) networks or smaller. The processing time, which increases exponentially, is derived as follows: For an ($N \times N$) network, there are $2^{(N/2)-1}$ states of the TPM. If the TPM is a binary counter (no intelligence), on the average about half of the states or $2^{(N/2)-2}$ must be tested before the given P is decomposed to C_1 and C_2. Also, on the average, one-half of the I/O memory is scanned before an output integer pair is detected. Therefore, the number (A_T) of times that the I/O memory is accessed for $\log_2 N$ levels [the $\log_2 N$th level is the trivial (2×2) network, and the number of times the memory is accessed is simply $N/2$] is:

$$A_T = \frac{N}{8} \sum_{i=1}^{\log_2 N - 1} 2^{N/2^i} + \frac{N}{2}. \tag{1}$$

The processing time can be greatly reduced by using the looping procedure. If a random-access I/O memory is employed, it is necessary to search for one-half of the outputs (the remaining outputs are obtained directly from memory) on each level of the network. For an $(N \times N)$ network, $N/2$ outputs are determined by accessing the I/O memory, on the average, $N/2$ times. Then

$$A_T = N^2 \sum_{i=1}^{\log_2 N} \left(\frac{1}{2}\right)^{i+1} = \tfrac{1}{2}N(N-1). \tag{2}$$

In addition, the access time required to partition the memory is $2N \log_2 N$ and should be included in equations (1) and (2) to give the total processing time.

If two memories (in the crisscross manner) or a CAM is utilized, no searching is necessary, and access times for the decomposition of P and partitioning of memories are:

$$A_T = 4N \log_2 N$$

and (3)

$$A_T = 3N \log_2 N \quad \text{respectively.}$$

For $N = 16{,}384$ and a CAM with 1-μsec access time, the control algorithm, implemented with wired logic and a 10-MHz basic clock, can be accomplished in approximately *750 msec.* If two memories (random access) with 1-μsec cycle time are used, a processor with an instruction execution time of 3 μsec can implement the control algorithm in approximately 50 seconds.

VI. CONCLUSION

In this paper, the network structure and control algorithm for certain $(N \times N)$ rearrangeable switching networks are described. The algorithm consists of decomposing a given permutation into d (where

d is the base of the network) permutations for the d $(N/d \times N/d)$ subnetworks, and determining the connections for the N/d $(d \times d)$ networks in the input and output stages. The same procedure is applied to the subnetworks in an iterative manner until all of the connections in the $(N \times N)$ network are defined.

Although the network can be constructed with various building blocks (bases), the base-2 structure is the most important because it requires the least number of two-state devices (β-elements) and the control algorithm is relatively simple. The algorithm is implemented by performing the decomposition either by the looping procedure or by a Trial-Partition Machine. Also, an efficient coding scheme is defined to facilitate the decomposition of the permutation and the partitioning of the memory. With the base-2 structure, the control algorithm and the coding scheme can be used in a consistent manner at each level of the network. There are other classes of network structures with the same number of β-elements, such as the nested-tree networks,[6] that do not have this property.

The processing time and equipment complexity vary with the methods of implementation. The combinational logic is the fastest and least expensive; however, it is only applicable for very small networks. The Trial-Partition Machine is economical, but it is too slow for large networks; however, intelligence can be designed into the machine taking advantage of the fact that for a large number of permutations there are more ways than one of setting the network. Consequently, the processing time is dependent on the permutations given, as well as the amount of intelligence built in. The most suitable method for networks larger than (64×64) is the looping procedure with a content addressable memory to store the outputs. The processing time is independent of the permutations given. With this method, it is possible to determine the setting of all the β-elements for a $(16{,}384 \times 16{,}384)$ network in less than one second for *any* number of new connections or terminations. During the processing time, the new settings for β-elements are stored in a buffer; then their states are changed. The memory required for this system is about 300k bits or approximately 20 bits per input terminal.

This paper has described a control algorithm for a rearrangeable switching network that is practical from both the system and processing time viewpoints. The application of this network should be considered where full access and nonblocking is required, and rerouting is possible.

REFERENCES

1. Joel, A. E., Jr., unpublished work.
2. Case, C. C., unpublished work.
3. Levitt, K. N., et al., "A Study of the Data Communication Problems in a Self-Repairable Multiprocessor," Spring 1968 Joint Computer Conf., AFIPS Proc., *32,* pp. 515–527.
4. Clos, C., "A Study of Non-Blocking Switching Networks," B.S.T.J., *32,* No. 2 (March 1953), pp. 406–424.
5. Beneš, V. E., "Optimal Rearrangeable Multistage Connecting Networks," B.S.T.J., *43,* No. 4, Part 2 (July 1964), pp. 1641–1656.
6. Joel, A. E., Jr., "On Permutation Switching Networks," B.S.T.J., *47,* No. 5 (June 1968), pp. 813–822.
7. Batcher, K. E., "Sorting Networks and Their Applications," Spring 1968 Joint Computer Conf., AFIPS Proc., *32,* pp. 307–314.
8. Kautz, W. H., et al., "Cellular Interconnection Arrays," IEEE Trans. Computers, *EC-17* (May 1968), pp. 443–451.
9. Waksman, A., "A Permutation Network," JACM, *15,* No. 1 (January 1968), pp. 159–163.
10. Neiman, V. I., "Structure et Commande Optimales de Réseaux de Connexion Sans Blocage," Annales des Tèlècommunications, July/August 1969, pp. 232–238.
11. Tsao-Wu, N. T., and Opferman, D. C., "On Permutation Algorithms for Rearrangeable Switching Networks," IEEE Int. Conf. Commun., 1969, Conf. Record, pp. 10–29—10–34.
12. Koo, J., "Integrated Circuit Content Addressable Memories," IEEE Int. Solid-State Circuit Conf., 1970, Digest of Technical Papers, pp. 72–73.