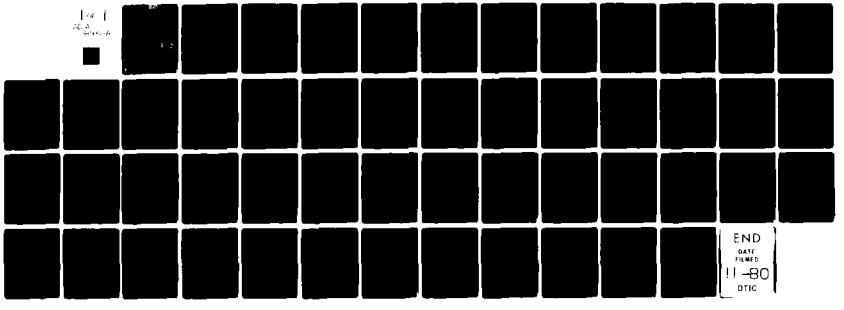
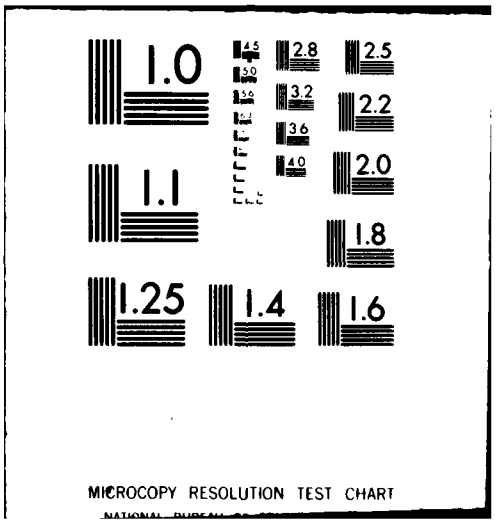
AD-A089 688 OAKLAND UNIV ROCHESTER MI SCHOOL OF ENGINEERING F/G 17/2
A UNIFORM REPRESENTATION OF SINGLE AND MULTI-STAGE INTERCONNECT--ETC(U)
JUN 80 D K PRADHAN, K L KODANDAPANI F49620-79-C-0119
UNCLASSIFIED AFOSR-TR-80-0950 NL



END
DATE FILMED
11-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

(6)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (19) AFOSR-TR-80-0950	2. GOVT ACCESSION NO. AD-A089688
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) A UNIFORM REPRESENTATION OF SINGLE AND MULTI-STAGE INTERCONNECTION NETWORKS USED IN SIMD MACHINES.	5. TYPE OF REPORT & PERIOD COVERED (9) Interim rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) D. K. Pradhan, K. L. Kodandapani (12) 32	8. CONTRACT OR GRANT NUMBER(s) (15) F49620-79-C-0119
9. PERFORMING ORGANIZATION NAME AND ADDRESS Oakland University, School of Engineering, Rochester, MI 48063	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 61102F (17) 2304/A6
11. CONTROLLING OFFICE NAME AND ADDRESS (11) Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC	12. REPORT DATE Jun[illegible]80
	13. NUMBER OF PAGES 49
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE SEP 30 1980 S D E

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A switching theoretic framework for the study of interconnection networks is developed. An equivalence relationship between networks id defined. Single stage and multi-stage netwcrks that are particularly useful for single instruc-tions multiple data stream (SIMD) machines are studied. It is shown that the networks form two distinct equivalence classes under this definition of equiva-lence relationship. It is shown that any multi-stage network can be easily modified to realize the permutations that are admissable by any other network which is equivalent to it.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 405252

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A089688

DDC FILE COPY

AFOSR-TR- 80-0950

To appear in IEEE Transactions on Computers (Fall 1980)

A UNIFORM REPRESENTATION OF SINGLE AND MULTI-STAGE INTERCONNECTION NETWORKS USED IN SIMD MACHINES

D. K. Pradhan and K. L. Kodandapani

ABSTRACT

A switching theoretic framework for the study of interconnection networks is developed. An equivalence relationship between networks is defined. Single stage and multi-stage networks that are particularly useful for single instruction multiple data stream (SIMD) machines are studied. It is shown that the networks form two distinct equivalence classes under this definition of equivalence relationship. It is shown that any multi-stage network can be easily modified to realize the permutations that are admissable by any other network which is equivalent to it.

D. K. Pradhan is with the School of Engineering, Oakland University, Rochester, Michigan 48063.
K. L. Kodandapani is with NCR Corporation, Wichita, Kansas.

Research reported supported by AFOSR Contract No. [redacted]
F49620-79-C-0119

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

I. INTRODUCTION

The study of permutations and permutation networks has been an important topic of research in parallel processing [4-12]. Permutations of the data as well as of the intermediate results are required in order to execute the algorithms that are used in parallel processing. Also, the ability to simultaneously access multiple data elements from memory is key to successful parallel processing. This simultaneous access is achieved by the use of multiple memory modules [1] where those data items that may be simultaneously needed are stored in different modules. Several permutation techniques [4-12] have been proposed for arranging the data items so that conflict-free access to the memory modules is achieved.

In the literature, certain permutations, such as uniform shift, unscrambling of t-ordered vectors, etc., have been identified as important in parallel processing (especially in SIMD-type machines). These findings have influenced the design of some of the permutation networks that have been developed for interconnection between memory modules and processors [4-12].

This paper introduces a switching theory framework for the study of interconnection networks. With this framework, techniques are developed that characterize both single-stage and multi-stage interconnection networks. In particular, a variety of interconnection networks - especially useful in single instruction multiple data stream (SIMD) machines - are able to be completely characterized by using this switching-theory formulation. The set of networks studied here includes uniform shift networks [4,7], scrambling/unscrambling networks [8], Omega networks [5], simplified Omega networks [6], and Flip networks [13]. Also, we characterize those networks obtained by reversing the original networks; i.e., the outputs are used as inputs and the inputs as outputs (see Fig. 1). The reverse networks, besides being of theoretical interest, have practical importance in that a network may be operated in a bidirectional mode, thus requiring the use of the reverse network.

Based on this characterization, it is shown that the interconnection networks satisfy certain equivalence relationships; these relationships result in two distinct equivalence classes of networks. The first class includes most of the single stage networks and certain of the multi-stage networks. The second class constitutes most of the multi-stage networks. Several related results regarding structure and number of permutations admissable by these two classes of networks are developed.

Section II develops the basic framework for the study of interconnection networks. Interconnection networks can be broadly classified as single-stage and multi-stage. A single-stage network uses one stage of switching and the designed permutation is performed by using single or multiple passes through the network. A useful example of such a network can be found in the Illiac IV computer [4]. The interconnection network used in Illiac IV can perform four different types of uniform shift permutations. Multi-stage networks use n is equal to $\log_2 N$ and N is the number of inputs. The desired permutation is usually realized by using a single pass through the network. A block diagram of a multi-stage network is shown in Fig. 1. Each stage typically performs some type of shuffle and switch permutations.

Next, in Section III, certain single-stage networks are characterized. We consider multi-stage networks in Section IV. We characterize many multi-stage networks that have implemented or proposed in literature. Both the forward and reverse versions of the networks are studied. Two equivalence classes of networks are defined and the networks classified accordingly. Specifically, it is shown that uniform shift, unscrambling/scrambling and simplified Omega belong to the first class, whereas most of the multi-stage networks belong to the second class. Also, results regarding the structure and number of permutations admissable by these classes of networks are presented.

II. BASIC FORMULATION

This section develops the basic framework for the study of permutations admitted by single and multistage networks.

A permutation p can be defined as a one to one and on to mapping from a set of integers into itself. The permutation p is usually represented as $\{(i,p(i)) \mid 0 \leq i \leq N-1\}$ where p(i) represents the mapping of i, $0 \leq i \leq N-1$.

In this paper, as in most of the previous work [6-8], we assume $N = 2^n$ for some n. In the following, we introduce P, an alternate representation of p.

Consider the binary numbers 0 to N-1. Let B^n represent these binary numbers. The set B^n consists of 2^n distinct binary n-tuples. Let $F : B^n \rightarrow B^n$ be a mapping, as defined below.

For each i, $0 \leq i \leq N-1$, let $\underline{i} = (i_n, i_{n-1}, \ldots, i_1)$ denote the binary number, i. Let $P(\underline{i}) = \underline{j}$ where $\underline{j} = (j_n, j_{n-1}, \ldots, j_k, \ldots, j_1)$ denote the binary number j, where $j = p(i)$. Thus, P is one-to-one and onto mapping of B^n into B^n and is the permutation p, expressed in terms of the binary numbers.

Now P in turn can be expressed as a collection of n switching functions $f_1, f_2, \ldots, f_k, \ldots, f_n$. Each of these functions, f_k, $1 \leq k \leq n$, is an n-variable functions, as defined below:

For $1 \leq k \leq n$, let:

$$Y_k = f_k(y_n, y_{n-1}, \ldots, y_1), \text{ and let}$$

$$j_k = f_k(y_n = i_n,\ y_{n-1} = i_{n-1}, \ldots, y_1 = i_1) \quad \text{where}$$

j_k is the k-th component of the binary number $\underline{j}$, given by $F(\underline{i}) = \underline{j}$.

Example 1:

Consider the following permutation:

		y_3	y_2	y_1	Y_3	Y_2	Y_1
i	j = p(i)	i_3	i_2	i_1	j_3	j_2	j_1
0	0	0	0	0	0	0	0
1	5	0	0	1	1	0	1
2	6	0	1	0	1	1	0
3	7	0	1	1	1	1	1
4	1	1	0	0	0	0	1
5	2	1	0	1	0	1	0
6	3	1	1	0	0	1	1
7	4	1	1	1	1	0	0

$$Y_3 = \bar{y}_3\bar{y}_2y_1 + \bar{y}_3y_2 + y_3y_2y_1$$

$$Y_2 = \bar{y}_3y_2 + y_3\bar{y}_2y_1 + y_3y_2\bar{y}_1$$

$$Y_1 = \bar{y}_3y_1 + y_3\bar{y}_1$$

Thus, we see that every permutation on 2^n elements can be represented in terms of n switching functions.

Let $(i,p(i))$ $0 \leq i \leq N-1$ be a permutation admitted by an interconnection network. Let the permutation, p, be represented by n-switching functions, $Y_n, Y_{n-1}, \ldots, Y_1$ of the variables, $y_n, y_{n-1}, \ldots, y_1$. It may be observed that in the truth table representations of these functions, each Y_j column contains exactly 2^{n-1} 1's; the rows describing Y_j's are all distinct. Thus, the Y_j function may possess certain regular structures and may satisfy certain relationships. This is precisely what we explore in the next two sections, for single and multistage networks, respectively.

The following develops a definition for an equivalence relationship between networks; and some related results that will be useful in the next two sections.

Definition: A bit-rearrangement (bit) permutation is a permutation for which the Y functions are of the type: $Y_k = y_j$; i.e., Y_k's are functions of single y - variables (that are positive).

The identity, perfect shuffle and bit reversal permutations are bit-rearrangement type permutations.

Definition: A network that performs a bit-rearrangement permutation will be referred to as a bit network.

Theorem 1 : There are n! bit permutations of $N = 2^n$ elements. This set of permutations forms a group.

Proof: Consider the set of n functions, $\{Y_1, Y_2, \ldots, Y_n\}$, where Y_k is of the type, $Y_k = y_j$. Since the n, Y_k's can be assigned n, y_j's and Y_k's have to be distinct in each set, there are n! such distinct assignments. Each of these assignments constitutes a bit-rearrangement permutation.

Clearly there permutations form a group, since (i) the identity permutation is a bit permutation ($Y_k = y_k$) (ii) the inverse of a bit permutation is also a bit permutation (iii) concatenation of two bit permutations is also a bit permutation

Q.E.D.

Let P and P^* denote the sets of admissable permutations for two networks, G and G^*, respectively. Let X-Y-Z denote the concatenation of networks X followed by Y followed by Z.

Let X^{-1} represent the reverse network of X.

Definition: The networks G and G^* are said to be equivalent if there exist some bit networks, Φ and Ψ, not necessarily distinct, such that $\Phi - G - \Psi$ admits P^* and $\Phi^{-1} - G^* - \Psi^{-1}$ admits P.

Note that (i) G is equivalent to G, itself, since and both can identify networks which are also bit networks. (ii) and characterize the equivalence relationship.

The above definition implies that two networks are equivalent if one can be realized by using the other, with the possible addition of fixed bit networks at the inputs and/or outputs to perform the required bit permutation. Thus, the above definition of equivalence is conceptually different from other equivalence relationships such as topological equivalence.

Corollary 1: Given a network, G, there are at most $(n!)^2-1$ other networks that are equivalent to G.

Proof: From Theorem 1, one has n! distinct bit permutation networks that can be placed at the inputs and/or outputs of G to obtain its equivalent networks. Thus, there are, at most, $(n!)^2$ networks that are equivalent to G. This includes G, itself. Hence, the theorem.

Q.E.D.

Corollary 2: If G and G^* are equivalent for each $p \in P$, there exists $p^* \in P^*$, such that:

$$p = \Phi^{-1} p^* \Psi^{-1}$$

$$p^* = \Phi p \Psi$$

Proof: Proof is an immediate consequence of the equivalence relationship.

Q.E.D.

Let (p, p^*) represent a pair of permutations that satisfy the relationship in Corollary 2. Let F_p and F_{p^*} represent the two sets of Y_k functions that describe P and P^*, respectively.

Corollary 3: By relabeling y and Y variables according to a fixed mapping defined by Φ and Ψ(Φ^{-1} and Ψ^{-1}), one can obtain F_p* from F_p (F_p from F_p*).

Proof: Proof is obvious.

Example: Let Φ, and Ψ, that define the equivalence relationship between G and G^*, both realize the bit reversal permutation. Thus, Φ^{-1} and Ψ^{-1} also realize bit reversal permutation.

Let N = 8 and let F_p represent the following three functions for some P:

$$Y_1 = y_1 \oplus y_2$$

$$Y_2 = y_2 \oplus y_3$$

$$Y_3 = y_1 \oplus y_3$$

The relabelling defined by bit reversal permutation is as follows:

$$y_1 \rightarrow y_3 \qquad Y_1 \rightarrow Y_3$$

$$y_2 \rightarrow y_2 \qquad Y_2 \rightarrow Y_2$$

$$y_3 \rightarrow y_3 \qquad Y_3 \rightarrow Y_3$$

Thus, F_p^* represents:

$$Y_3 = y_3 \oplus y_2$$

$$Y_2 = y_2 \oplus y_1$$

$$Y_1 = y_3 \oplus y_1$$

The permutation, p^*, described by F_p^*, is admissable by G^*.

III. SINGLE-STAGE NETWORKS

In this section, we consider various types of single-stage interconnection networks. Specifically, the permutations admissable by perfect shuffle networks, uniform shift networks and networks which perform unscrambling of t-ordered* vectors are each characterized. We restrict our discussion to permutations admissable by a single pass through the network. (The use of multiple passes through the network is equivalent to using single passes in a multi-stage network and is considered in section IV.)

First we introduce the following definition to be used in developing the results of this paper.

Definition: Let $g(x_n,x_{n-1},\ldots,x_i,\ldots,x_1)$ be an n-variable function. Then $\frac{dg}{dx_i} = g(x_n,x_{n-1},\ldots,x_i = 0,\ldots,x_1) \oplus g(x_n,x_{n-1},\ldots,x_i = 1,\ldots,x_1)$ where $\oplus$ is the exclusive-or operator. The function $\frac{dg}{dx_i}$ is said to be the Boolean difference [13,14] of g with respect to x_i.

Example 2:

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

For this function we can compute

$$\frac{dg}{dx_3} = x_1\bar{x}_2 .$$

It may be noted that given the ex-or sum of products expression for g, the

---

* In literature, this has been referred to as p-ordered vectors.

Boolean difference $\frac{dg}{dx_i}$ can be obtained by simply deleting all terms that do not involve x_i and also, all appearances of x_i.

(a) Perfect-Shuffle Network

The permutations admitted by perfect-shuffle networks are useful in parallel algorithms for polynomial evaluation, FFT, sorting, etc. [3]. Also, these networks are useful for constructing multistage networks.

Theorem 1: The Y_k functions that represent the permutations admitted by (i) the perfect-shuffle network, is given as:

$$Y_i = \begin{cases} y_n & i = 1 \\ y_{i-1} & 2 \leq i \leq n \end{cases}$$

(ii) the reverse perfect-shuffle network, is given as:

$$Y_i = \begin{cases} y_i & i = n \\ y_{i+1} & 1 \leq i \leq n-1 \end{cases}$$

Proof: Proof is obvious from the definition of perfection shuffle, given as:

$$p(i) = \begin{cases} 2i & 0 \leq i \leq 2^{n-1}-1 \\ 2i-N+1 & 2^{n-1} \leq i \leq 2^n \end{cases}$$ Q.E.D.

(b) Uniform Shift Network

These are simple networks and can perform many useful permutations in $O(\sqrt{N})$ units of time [7]. These networks have found use in Illiac IV computers and shift register memories.

The permutations admitted by these networks can be defined as:

$$p(i) = (i + d) \bmod N, \text{ where } d$$

represents the amount of shift which may be provided to the network through a control input.

It may be noted that the reverse of the uniform shift network is also

a uniform shift network.

Theorem 2: The Y_k functions that represent any permutation which is admitted by the uniform shift network can be expressed as:

$$Y_n = y_n \oplus g_n(y_{n-1}, y_{n-2}, \ldots, y_1) \quad \text{where}$$

the function, g_n, is defined by d, the amount of shift. The other functions, $Y_{n-1}, Y_{n-2}, \ldots, Y_1$ are related to Y_n through the following recursive rule:

$$\text{If } Y_k = y_k \oplus g_k(y_{k-1}, y_{k-2}, \ldots, y_1) \quad \text{then } Y_{k-1} = y_{k-1} \oplus \frac{dg_k}{dy_{k-1}} \ .$$

Proof: Proof provided in the Appendix.

Example 3: Consider the following permutation which corresponds to a uniform shift of 3 on N = 8.

i	p(i)	y_3	y_2	y_1	Y_3	Y_2	Y_1
0	3	0	0	0	0	1	1
1	4	0	0	1	1	0	0
2	5	0	1	0	1	0	1
3	6	0	1	1	1	1	0
4	7	1	0	0	1	1	1
5	0	1	0	1	0	0	0
6	1	1	1	0	0	0	1
7	2	1	1	1	0	1	0

$$Y_3 = y_3 \oplus y_2 \oplus y_1 \oplus y_1 y_2$$

$$Y_2 = y_2 \oplus y_1 \oplus 1$$

$$Y_1 = y_1 \oplus 1$$

$$g_3(y_3, y_2, y_1) = y_2 \oplus y_1 \oplus y_1 y_2$$

$$g_2(y_3,y_2,y_1) = y_1 \oplus 1$$

$$g_1(y_3,y_2,y_1) = 1$$

It may be seen that $g_2(y_3,y_2,y_1) = \frac{dg_3}{dy_2}$

$$g_1(y_3,y_2,y_1) = \frac{dg_2}{dy_1}$$

(c) PM2I Network [15]

These plus-minus 2^i networks form the basis for data manipulator and ADM multistage networks. These networks can be interpreted as uniform shift networks for $d = \pm 2^i$. Thus, Theorem 2 holds for these networks.

(d) Scrambling/Unscrambling

These networks are useful to scramble the data when stored in to parallel memories and unscramble the data when it is read from the memory. These are particularly useful in matrix manipulations as the permutations performed by these networks provide the required alignment of data to

(i) obtain conflict-free access of arrays from multiple memory modules and (ii) process the accessed data simultaneously using multiprocessors.

The basic permutation admitted by these networks is called unscrambling of t-ordered vectors defined as:

$$p(i) = ti \bmod N, \quad \text{where}$$

t and N are relatively prime and the value of t may be provided to the network by a control input.

It may be also noted here that the basic permutation admitted by the reverse network is also a permutation of the type:

$$p(i) = ti \bmod N.$$

Thus, we need to only characterize the forward network.

Theorem 3: The Y_k functions that represent the permutations admitted by the scrambling/unscrambling networks can be expressed as:

$$Y_n = y_n \oplus g_n(y_{n-1}, y_{n-2}, \ldots, y_1) \quad \text{where}$$

the function, g_n, is defined by t. The other functions, $Y_{n-1}, Y_{n-2}, \ldots, Y_1$, are related to Y_n through the following recurrsive rule:

If $\quad Y_k = y_k \oplus g_k(y_{k-1}, y_{k-2}, \ldots, y_1)$ then

$$Y_{k-1} = y_{k-1} \oplus \frac{dg_k^*}{dy_{k-1}} \quad \text{where}$$

$$\frac{dg_k^*}{dy_{k-1}} = \begin{cases} \dfrac{dg_k}{dy_{k-1}} & \text{if } (t-1)/2 \text{ is even} \\ 1 \oplus \dfrac{dg_k}{dy_{k-1}} & \text{if } (t-1)/2 \text{ is odd} \end{cases}$$

Proof: Proof is provided in the Appendix.

It may be seen that permutations admissable by networks to scrambling/unscrambling networks and uniform shift networks have certain similar characteristics. In the next section we will show that these permutations, as well as those realized by certain multistage networks, belong to the class of "symmetric permutations" (to be defined later).

IV. MULTI-STAGE NETWORKS

This section studies the multi-stage networks. These networks are more useful than single stage networks as they can admit a much larger class of permutations. Several different multi-stage networks are characterized in this section, and related results are derived.

Specifically, we characterize both forward and reverse versions of: a) Omega networks [5], b) simplified Omega networks [6], and c) Flip networks [13]. Certain equivalence relationships between these networks are then derived. The networks are shown to belong to the following two distinct classes:
Class I: Networks for which the Y_k function has the following general form:

$$Y_k = y_k \oplus f_k\ (y_{k-1}, y_{k-2}, \ldots, y_1)$$

The simplified Omega networks, reverse simplified Omega networks and certain single stage interconnection networks belong to this class, as shown.
Class II: Networks for which the Y_k functions have the following general form:

$$Y_k = y_k \oplus f_k\ (Y_n, Y_{n-1}, \ldots, Y_{k+1}, y_{k-1}, \ldots, y_1)$$

The Omega, reverse Omega, Flip and reverse Flip are shown to belong to this second class.

Further, it is shown that the fraction of Class II permutations that are admissable by Class I networks tends to 0 asymptotically. Also, we will see that Class I permutations are "symmetric" (to be defined later).

The techniques used to characterize the networks is based on the following two observations:

(i) Any permutation which is realized by an n-stage network can be expressed as a composition of a sequence of n permutations.

(ii) Each of these n permutations can, in turn, be expressed as a composition of certain easily characterizable elementary permutations.

The networks considered here are N-input, N-output networks where $N = 2^n$ and n is the number of stages

(a) Omega Networks

This network was introduced by Lawrie [5]. Each stage in this network is

composed of two subnetworks, which are denoted as S and E in Fig. 3. The first subnetwork, S, moves the contents of its input, i, to its output, j, where j = p(i), and p is the perfect shuffle map. The second subnetwork, E, moves the contents of its inputs, i, and (i+1) to its output, (i+1) and i, respectively, for certain selected i's, where i is an even number. For all other inputs, i, the contents are moved straight through to the outputs, i. Thus, in effect, E performs exchanges on the contents of certain selected pairs of adjacent inputs.

A set of control bits determines the subset of pairs which are selected for exchange. One bit per pass per data item is all that is needed for controlling the exchange operation. The data items carry with them these control bits, and thus, the control bits form an integral part of the contents of the inputs or outputs.

Each data item carries with it n-control bits. During the k-th pass, the k-th control bits are used for controlling the exchange operation. The contents of a certain pair of inputs are exchanged (not exchanged) during the k-th pass, if k-th control bits (which are contained in both of these inputs) are 1 (0).

Theorem 5: The Y_k functions that represent any permutation which is admitted by the Omega network can be expressed as:

$$Y_k = y_k \bullet f_k\,(Y_n, Y_{n-1}, \ldots, Y_{k+1}, y_{k-1}, y_{k-2}, \ldots, y_1)$$

for all k, $1 \leq k \leq n$ where the function, f_k, is defined by the control bits, C_i^k that are used during the k-th pass.

Proof: Proof given in the Appendix Q.E.D.

The above result was derived independently by Pease [9] and the authors [17] at about the same time. However, we have here provided a rigorous proof of the Theorem which was not provided in [9]. The techniques that are used for formulating the proof will be shown to be useful in deriving similar results for other networks.

In the following, we characterize the reverse Omega networks; these networks consist of n stages of exchange and inverse-shuffle networks. A

control algorithm for reverse Omega networks is also available [18].

Theorem 6: The Y_k functions that represent any permutations, which are admitted by the reverse Omega networks, can be expressed as:

$$Y_k = y_k \oplus f_k\,(y_n, y_{n-1}, \ldots, y_{k+1}, Y_{k-1}, Y_{k-2}, \ldots, Y_1)$$

for all k, $1 \leq k \leq n$, where the function, f_k, is defined by the control bits, C_i^k, that are used during the k-th pass.

Proof: Proof is similar to that given for Theorem 5 in the Appendix. The two differences here are that: the inverse shuffle performs one bit end around the right shift and is preceeded by the exchange permutation. The equations in the given proof can be modified in a straight-forward manner to obtain the Theorem. Q.E.D.

Theorem 7: The Omega and reverse Omega networks are equivalent networks.

Proof: It may be first seen that one can obtain the expressions given in Theorem 6 from those given in Theorem 7 and vice versa by relabeling both the input and output variables as described below:

y_k as y_{n-K+1} and Y_k as Y_{n-k+1} for all k, $1 \leq k \leq n$.

This implies that any permutation admissable by Omega (reverse Omega) networks is also admissable by a cascade of three networks, as shown in Fig. 4. The first and third networks here perform the bit reversal permutation. The second network is the reverse Omega (Omega) network. Hence the equivalence relationship. Q.E.D.

(b) Simplified Omega Networks

A simplified version of Omega networks was introduced by Lang-Stone [6].

The simplified Omega networks receive only one control bit per data item. These bits constitute the control bits that are used during the first pass. These control bits are used in the same way as in the SE network, where they control the exchange operation during the first pass. However, for every

successive pass after this first pass, a new control bit is computed for each data item, and these new bits now control the exchange operation during that pass. These new bits are computed from the control bits that have been used during the immediately preceding pass; this is described below:

During any pass other than the first, certain predefined Boolean operations are performed on every pair of control bits (which are contained in the i-th and (i+1)-th inputs of E, for all even i). The bits produced by these Boolean operations then replace the existing control bits (in their respective pairs of inputs) and these new bits are then used by E as control bits to perform the exchange operation for that current pass.

In the following, we discuss certain generalized, versions of the simplified Omega networks; this enables us to derive results with broader implications.

Let the control function that is used in any pass be a function of certain tag bits which are transmitted with the data items. Thus, one can select any arbitrary control function for any pass. (However, it may be noted that we restrict the control function to be the same for all data items, during a particular pass). Hence, in order to produce the desired permutation, the use of any combination of control functions for the n-different passes, is available.

Four tag bits are required in order to specify 1-out-of-16 possible different two-variable functions. Since there are n-passes, only 4n tag bits are required, altogether. (For example, given 256 data items, only 32 tag bits are required--a small number, when compared to the 256 control bits that are used.) As it will be seen later, the use of these additional bits can produce a much larger number of permutations. This is significant when compared to the number of permutations that are admitted by the network when the control functions are prespecified.

This version of the simplified Omega network, which allows for arbitrary control functions, will be hereafter referred to as Simplified Omega with Arbitrary Control (SOAC) networks.

Theorem 8: Any permutation that is admitted by the SOAC network can be represented by functions defined as below:

$$Y_n = y_n \oplus f_n (y_{n-1}, y_{n-2}, \ldots, y_1),$$

where the function, f_n, is defined by the control bits used during the first

pass. The functions, $Y_{n-1}, Y_{n-2}, \ldots, Y_1$, are defined by the following recursive relationship:

If $Y_k = y_k \oplus f_k(y_{k-1}, y_{k-2}, \ldots, y_1)$ then,

$Y_{k-1} = y_{k-1} \oplus f_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1)$, where

$$f_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1) = f_k(y_{k-1} = 0, y_{k-2}, \ldots, y_1) * f_k(y_{k-1} = 1, y_{k-2}, \ldots, y)$$

and * is that Boolean operation which is used during the (n-k+2)-th pass to compute the control bits.

Proof: Proof is given in the Appendix.

Now, it may be noted that if the control function that is used in the network is either an exclusive-or function, or an equivalence function (as it is proposed in [2], then the recursive relationship between Y_k and Y_{k-1} reduces to the following:

If $Y_k = y_k \oplus f_k(y_{k-1}, y_{k-2}, \ldots, y_1)$, then

$$Y_{k-1} = \begin{cases} y_{k-1} \oplus \dfrac{df_k}{dy_k}, & \text{if the control function is ex-or} \\ y_{k-1} \oplus \dfrac{df_k}{dy_k} \oplus 1, & \text{if the control function is equivance.} \end{cases}$$

It is also interesting to note that the above relationship is precisely the same relationship derived for certain single stage networks, in the last section.

In the following, we characterize the reverse SOAC network.

Theorem 9: Any permutation that is admitted by the reverse SOAC network can be represented by the functions given below:

$Y_1 = y_1 \oplus f_1(y_n, y_{n-1}, \ldots, y_2)$, where

the function, f_1, is defined by the control bits used during the first pass.

If $Y_k = y_k \oplus f_k\ (y_n, y_{n-1}, \ldots, y_{k+1})$, then

$Y_{k+1} = y_{k+1} \oplus f_{k+1}\ (y_n, y_{n-1}, \ldots, y_{k+2})$, where

$$f_{k+1}\ (y_n, y_{n-1}, \ldots, y_{k+2})$$

$$= f_k\ (y_n, y_{n-1}, \ldots, y_{k+2}, y_{k+1} = 0)\ *$$

$$f_k\ (y_n, y_{n-1}, \ldots, y_{k+2}, y_{k+1} = 1)$$

and * is that Boolean operation which is used during the k-th pass.

Proof: Proof is similar to that given for Theorem 8. Q.E.D.

If the control function, *, is an ex-or or an equivalence function, then the above recursive relationship can be expressed in terms of the Boolean difference, as shown above for SOAC networks.

Theorem 10: The SOAC networks and the reverse SOAC networks are equivalent.

Proof: The proof is similar to that given for the Omega network in Theorem 7. Q.E.D.

Thus, as it was seen in the case of Omega networks, with the use of fixed bit-reversal permutations at the inputs and outputs, one can realize reverse SOAC networks from SOAC network and vice versa.

(c) Flip Network

These networks were used in the STARAN computers [13]. These networks consist of two basic subnetworks: a flip network and a shift network.

The flip network performs the following permutation;

$j = i \oplus f$ where

f is a fixed control vector and ⊕ is a bit-by-bit ex-or function.

The shift network permutation is defined as:

$$j = i + 2^m \bmod 2^p, \text{ where } 0 \leq m \leq p \leq n \quad \text{and}$$

m, p represent control variables.

The Flip network has an equivalent multi-stage representation [16] that uses n stages of $\frac{N}{2}$ switches. Each stage performs basically an exchange permutation followed by a shuffle-type permutation. The $\frac{nN}{2}$ switches in the network may be controlled by control bits.

The following results can be derived by using the above equivalent representation, along with Theorem 5 and 6, or independently, by using the techniques similar to those used for characterizing Omega networks.

Theorem 11: The Y_k functions that represent any permutations which are admitted by the Flip networks can be expressed by:

$$Y_k = y_k \oplus f_k\,(y_n, y_{n-1}, \ldots, y_{k+1}, Y_{k-1}, Y_{k-2}, \ldots, Y_1)$$

for all k, $1 \leq k \leq n$, where the function, f_k, is defined by the control variables.

Theorem 12: The Y_k functions that represent any permutation which is admitted by the reverse Flip networks can be expressed by:

$$Y_k = y_k \oplus f_k\,(Y_n, Y_{n-1}, \ldots, Y_{k+1}, y_{k-1}, \ldots, y_1)$$

for all k, $1 \leq k \leq n$, where the function, f_k, is defined by the control variables.

It is readily seen that the Flip and Omega networks are equivalent. It can be seen that many multi-stage networks such as Banyan [14], Omega [5], Flip [13] and Indirect Binary n-cube [9] are indeed equivalent:

However, the important observation that may be made is that two distinct classes of networks do exist, as defined below:

(i) Class I network:

A network is said to belong to this class if the network is equivalent to

some network, G, which has the following form of characterizations:

$$Y_1 = y_1 \oplus f_1 (y_n, y_{n-1}, \ldots, y_2) \quad \text{and}$$

for all k, $2 \leq k \leq N$:

$$Y_k = y_k \oplus f_k (y_n, y_{n-1}, \ldots, y_{k+1}), \quad \text{where}$$

f_k is derived from f_{k-1}.

(ii) Class II Networks:

A network belongs to this class if the network is equivalent to some network, G, which has the following form of characterization:

$$Y_1 = y_1 \oplus f_1 (y_n, y_{n-1}, \ldots, y_2), \quad \text{and}$$

for all k, $2 \leq k \leq n$:

$$Y_k = y_k \oplus f_k (y_n, y_{n-1}, \ldots, y_{k+1}, Y_{k-1}, \ldots, Y_1)$$

where, f_k, can be any arbitrary function.

The chief difference between Class I and Class II networks is found in the way in which f_k is defined. For Class I networks, f_k is not a function of y_k and f_k is derived from f_{k-1}. On the other hand, for Class II networks, f_k can be any arbitrary function which can also be a function of y_k (since f_k is a function of Y_j, $1 \leq j \leq k-1$ which are inturn functions of y_k).

The simplified Omega, uniform shift, unscrambling/scrambling networks are all of the Class I type.

The Class II networks consist of Omega networks, Flip networks and the like.

Let P_1 be the set of all permutations that are admissable by the entire Class I networks.

Let P_2 be the set of all permutations that are admissable by the entire Class II networks.

Let $|X|$ denote the cardinality of set X.

Corollary 1: $P_1 \subseteq P_2$

Proof: Proof is obvious.

Corollary 2: $|P_1| \leq (n!)^2 \sqrt{2^{N + 8 \log N - 8}}$

Proof: First, it may be observed that the number of distinct Y_1 functions is equal to the number of distinct f_1 functions. Furthermore, every distinct multiple output function, $Y_2, Y_2, \ldots, Y_n$, represents a distinct permutation.

There are altogether $2^{2^{n-1}}$ functions of (n-1) variables. Thus, there are $2^{2^{n-1}} = \sqrt{2^N}$ distinct f_n (hence, Y_n) functions.

Using Theorem 6 and the fact that there are exactly 16 functions of two variables, one has, at most, 16 different f_k functions, given any f_{k-1} function.

From the above observations, it can be said that there are, at most $\sqrt{2^N \, 16^{n-1}}$ distinct multiple output functions, $Y_n, Y_{n-1}, \ldots, Y_1$. Hence, the number of distinct permutations is upper bounded by:

$$\sqrt{2^N \, 16^{n-1}} = \sqrt{2^{N + 8 \log n - 8}}$$

Proof is complete using the observation made in Theorem 1 and Corollary 1. Q.E.D.

Corollary 3: $\sqrt{2^{N \log N}} \leq |P_2| \leq (n!)^2 \sqrt{2^{N \log N}}$

Proof: The lower bound follows from the number of permutations that are admissable by the Omega network [5].

The upper bound is a direct consequence of Theorem 2 and the following observation:
There are exactly $2^{2^{n-1}}$ functions of (n-1) variables, and hence, there are $2^{2^{n-1}}$ distinct Y_j functions for each j. Q.E.D.

Corollary 4: The fraction of Class II permutations that are admitted by the Class I networks tends to 0 asymptotically with N.

Proof: $\dfrac{|P_1|}{|P_2|} \leq \dfrac{\text{upper bound on } |P_1|}{\text{lower bound on } |P_2|}$

≤ 0 as $N \to$ Q.E.D.

In the following, we provide a different characterization of P_2:

Let $p(i) = j$, and $p(i') = j'$, for some permutation, p, where $0 \leq i, j, i', j' \leq N-1$.

Let: $\underline{i} = (i_n, i_{n-1}, \ldots, i_k, \ldots, i_1)$,

$\underline{j} = (j_n, j_{n-1}, \ldots, j_k, \ldots, j_1)$,

$\underline{i}' = (i'_n, i'_{n-1}, \ldots, i'_k, \ldots, j_1)$ and,

$\underline{j}' = (j'_n, j'_{n-1}, \ldots, j'_k, \ldots, j_1)$ be the binary representation of i, j, i', and j', respectively.

Definition: A permutation, p, is said to be symmetric in the k-th bit of it satisfies the following: Given any i, i', for which $i_j = i'_j$ for all $j \neq k$, and i_k is the complement of i'_k ($i'_k = \bar{i}_k$), then the k-th bits of the resulting pair, j and j', are also the complement of each other; i.e.; $j'_k = \bar{j}_k$.

Definition: A permutation will be said to be symmetric if it is symmetric in all the n-bits.

Corollary 5: The permutations in P_1 are symmetric.

Proof: Let p represent any permutation in P_1.

Let $p(i) = j$, and $p(i') = j'$, for some i, i', j and j'.

Now consider the following equation for the k-th bit for class I networks:

$$j_k = i_k \oplus f_k (y_n = i_n, y_{n-1} = i_{n-1}, \ldots, y_{k+1} = i_{k+1}).$$

and $j_k' = i_k' \oplus f_k\ (y_n = i_n,\ y_{n-1} = i_{n-1}', \ldots, y_{k+1} = i_{k+1})$.

Let $i_j = i_j'$, $k+1 \leq j \leq n$, and $i_k = i_k'$.

Substituting this in the above equation.

for j_k and j_k', we get:

$$j_k = \overline{j_k}'$$ Q.E.D.

What is interesting about the above Corollary, is that it is also valid for multiple passes through the network. Therefore, this provides an explanation as to why such a permutation as bit reversal is <u>not</u> admissable by SOAC networks without using special techniques such as providing input queues/ buffering [19] to eliminate conflicts.

V. CONCLUSION

In this paper a switching theoretic formulation of SIMD interconnection networks is presented. It is shown that the networks can be classified into two distinct classes which are based on certain characteristics of the sets of admissable permutations. The first class of networks consists of certain single-stage networks and the simplified Omega networks. The other class of networks consists of most of the multi-stage networks, such as Omega, reverse Omega, Flip, etc. In Table III, we summarize the characteristics of these two classes of networks.

It is expected that the results of this paper should provide a new technique with which to analyze the interconnection networks. Further research in this area may be carried out by extending this work to multiple passes through the network, and hopefully we will be able to resolve some yet unanswered questions such as: whether two passes through a multi-stage network are sufficient to realize all permutations.

Network Type	Representative Networks	Characteristic Switching Functions	Structure of Admissable Permutation	Maximum Number of Permutations Realized
Class I	Uniform Shift, Unscrambling/Scrambling Simplified Omega	$Y_k = y_k \bullet f_k\ (y_n, y_{n-1}, \ldots, y_{k+1})$	Symmetric	$(n!)^2 \sqrt{2^{N+8 \log N - 8}}$
Class II	Omega, Flip, Banyan, etc.	$Y_k = y_k \bullet f_k\ (y_n, y_{n-1}), .., y_{k+1}, Y_{k-1}, \ldots, Y_.)$ f_k is not dependent on f_{k-1}	Not Restricted	$(n!)^2 \sqrt{2^{N \log N}}$

Table III: Summary of Characteristics

VI. REFERENCES

[1] J. P. Hayes, Computer Architecture and Organization, McGraw Hill, 1978.

[2] A. Mukhpadhayay, Recent Developments in Switching Theory, Academic Press, 1971.

[3] H. S. Stone, "Parallel Processing with Perfect Shuffle," IEEE Transactions on Computers, Vol. C-20, pp. 153-161, Feb. 1971.

[4] G. H. Barnes, et-al, "The Illiac IV Computer," IEEE Transactions on Computers, Vol. C-17, pp. 746-757, August 1968.

[5] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," IEEE Transactions on Computers, Vol. 4-21, No. 12, pp. 1145-1175, Dec. 1975.

[6] T. Lang and H. Stone, "A Shuffle-Exchange Network with Simplified Control," IEEE Transactions on Computers, Vol. C-25, No. 1, January 1976.

[7] S. E. Orcutt, "Efficient Data Routing Schemes for Illiac IV-Type Computers," Stanford University, DSL Report No. 70, April 1974 , Stanford University.

[8] R. C. Swanson, "Interconnection of Parallel Memories to Unscramble p-ordered Vectors," IEEE Transactions on Computers, Vol. C-23, pp. 1105-1116, Nov. 1974.

[9] M. C. Pease, "The Indirect Binary n-cube Microprocessor Array," IEEE Transactions on Computers, Vol. C-26, pp. 458-473, May 1977.

[10] _____, "An Adaption of the Fast Fourier Transform for Parallel Processing," TACM, Vol. 15, pp. 252-264, April 1968.

[11] T. Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations," IEEE Transactons on Computers, Vol. C-23, pp. 309-318, March 1974.

[12] S. Akers, "On a Theory of Boolean Functions," SIAM, J, Vol. 7, pp. 487-489.

[13] Batcher, K. E., The Flip Network in STARAN, Goodyear Aerospace Report, GER-16344, 1976.

[14] Goke, L. R. and Lipovoski, G. J., "Banyan Networks for Partitioning Multiprocessor Systems," in Proc. 1st Annual Symposium on Computer Architecture, pp. 21-28.

[15] Siegel, H. J., "Analysis Techniques for SIMD Machine Interconnection Networks and the Effects of Processor Address Masks," IEEE Trans. Computer, Vol. C-26, No. 2, Feb. 1977, pp. 153-161.

[16] C. Wu and T. Feng, "Routing Techniques for a Class of Multi-stage Interconnection Networks," Proc of the 1978 International Conference on Parallel Processing, pp. 197-205, August 1978.

[17] D. K. Pradhan and K. L. Kodandapani, "A Generalization of Shuffle-Exchange Networks," Technical Report, University of Regina, Regina, Canada, April 1976.

[18] J. Lenfant and S. Tahe, "Permuting Data With the Omega Network, RADC Final Report, Nov. 1978.

[19] T. Lang, "Interconnections between Professors and Memory Modules Using the Shuffle-Exchange Network," IEEE Transactions on Computers, May 1976, Vol. C-25, Number 5, pp. 496-502.

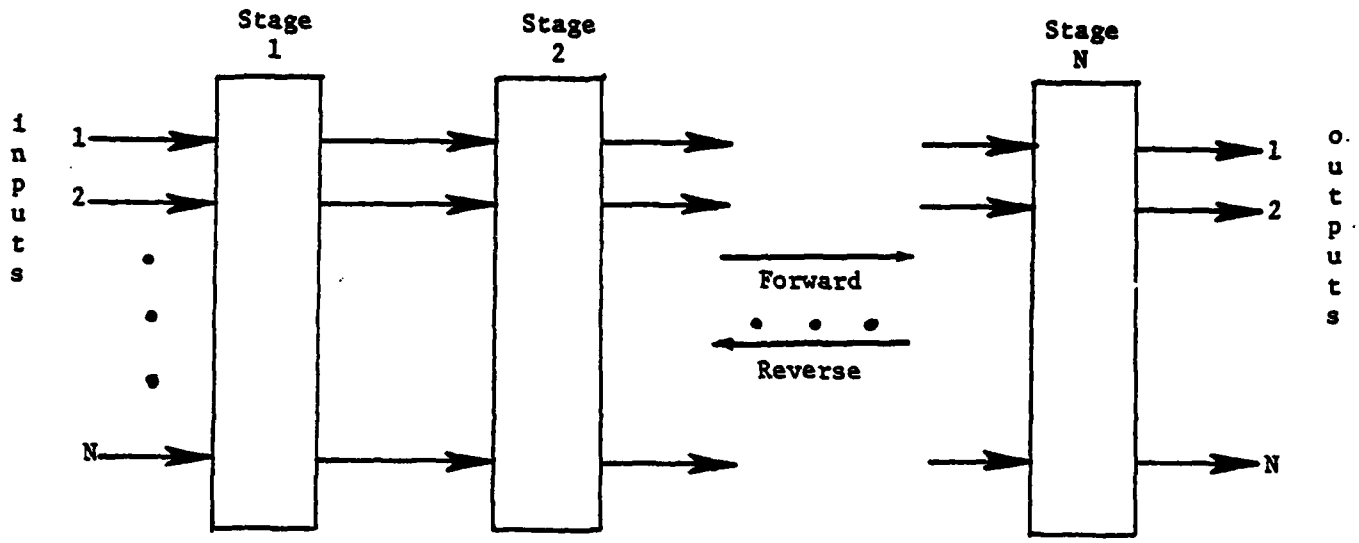


Fig. 1 Multi-stage Networks

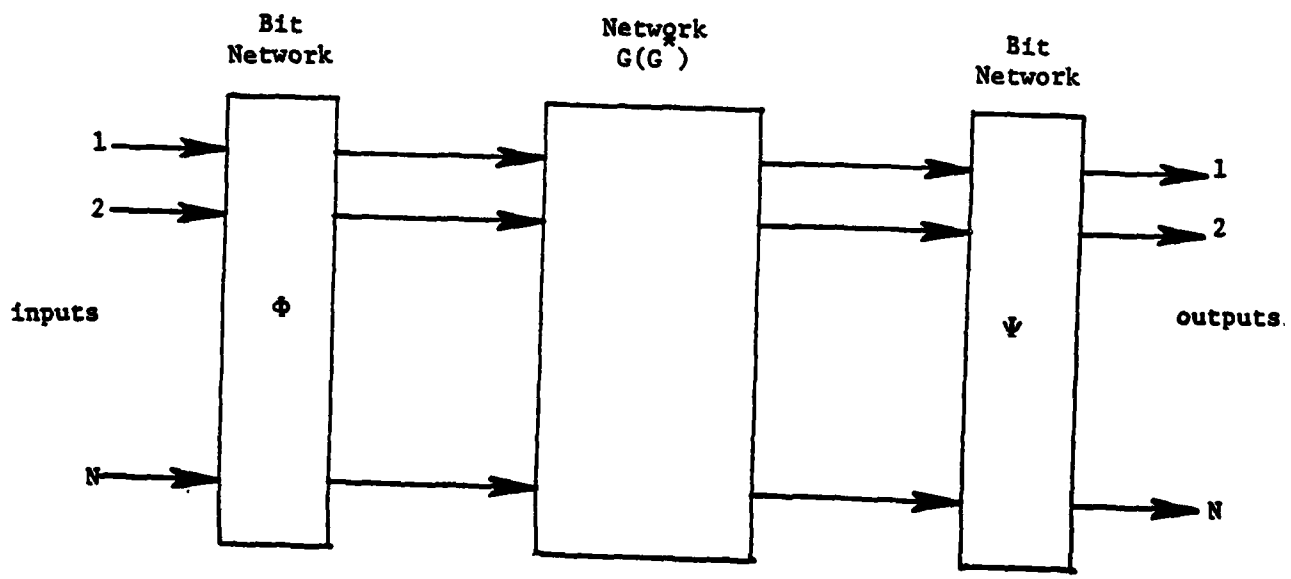


Fig. 2 Equivalence relationship between G and G^*

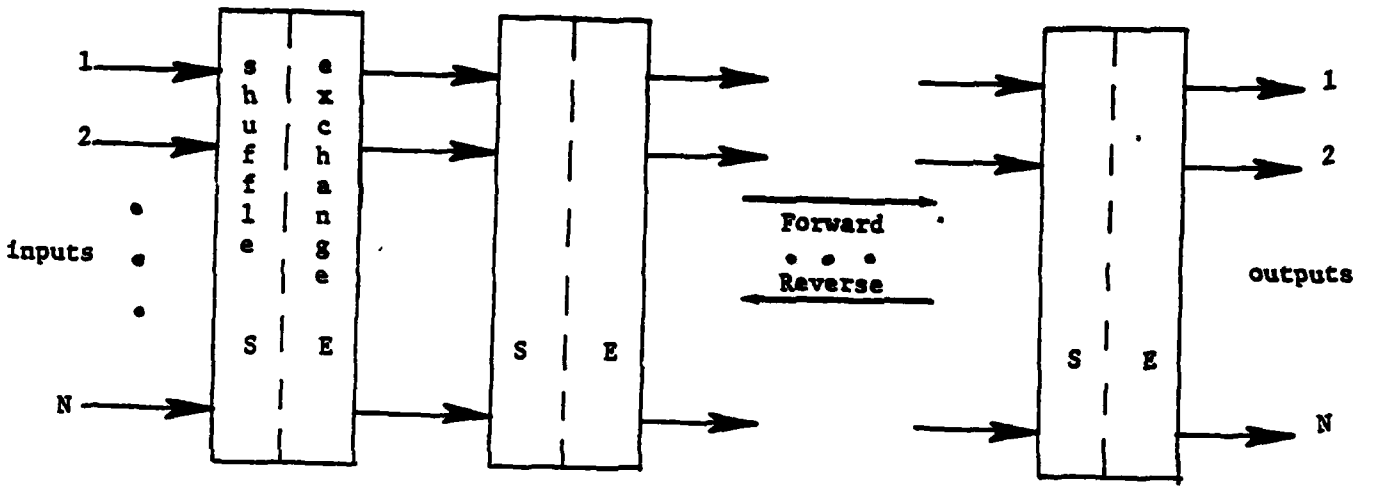


Fig. 3 Omega Network

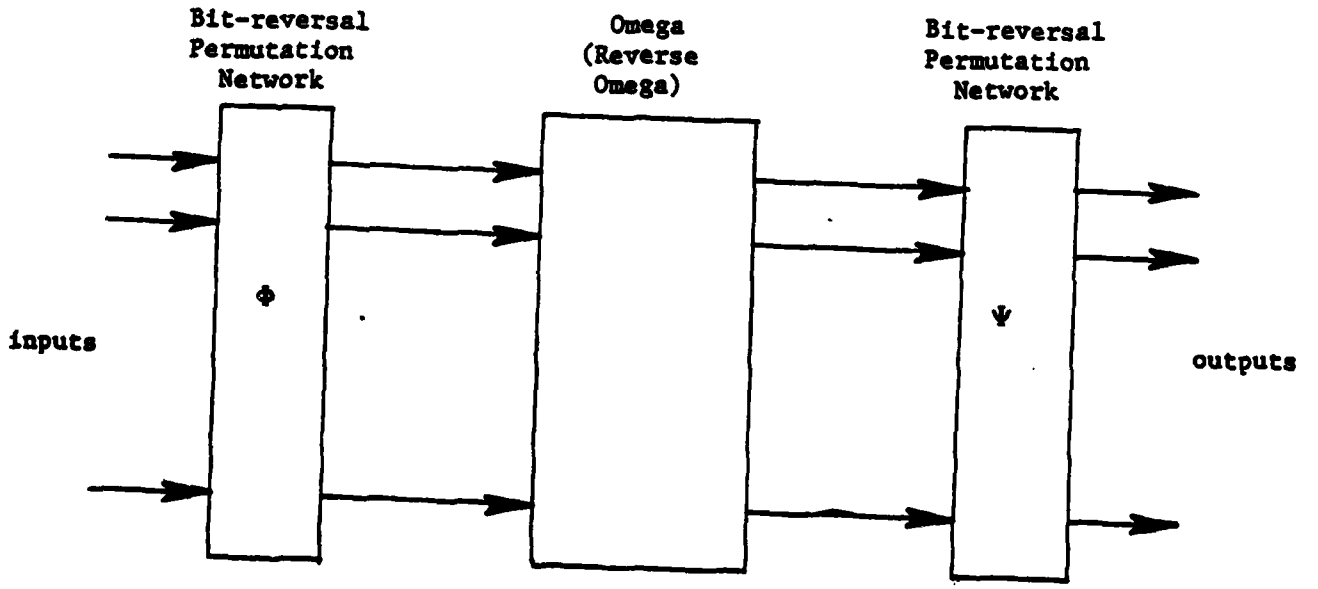


Fig. 4 Equivalence between Omega and Reverse Omega

APPENDIX

Proof of Theorem 3.

Proof: Let $(d_n, d_{n-1}, \ldots, d_1)$ be the binary-representation of the number i, the amount of shift. The permutation $p(i) = (i+d) \bmod N$ corresponds to addition of the fixed number d to each i. Since the addition is performed modulo $N = 2^n$, any carry from the n-th position is discarded.

The variables $Y_n, Y_{n-1}, \ldots, Y_1$ represent the sum bits produced by adding d to the number represented by the variables $y_n, y_{n-1}, \ldots, y_1$.

Thus, the function Y_n can be expressed as

$Y_n = y_n \oplus d_n \oplus C_n(y_{n-1}, y_{n-2}, \ldots, y_1)$, where C_n is the function representing the carry bit in to the n-th position. This carry bit is a function of the bits to the right of the n-th position. However, it may be noted that since d is a fixed integer for all i, C_n is expressed as a function of only the variables $y_{n-1}, y_{n-2}, \ldots, y_1$.

Let $g_n(y_{n-1}, y_{n-2}, \ldots, y_1) = d_n \oplus C_n(y_{n-1}, y_{n-2}, \ldots, y_1)$. Thus,

$Y_n = y_n \oplus g_n(y_{n-1}, y_{n-2}, \ldots, y_1)$.

Similarly, for any k, we can express $Y_k = y_k \oplus g_k(y_{k-1}, y_{k-2}, \ldots, y_1)$ where

$$g_k(y_{k-1}, y_{k-2}, \ldots, y_1) = d_k \oplus C_k(y_{k-1}, y_{k-2}, \ldots, y_1). \quad (1)$$

The function $C_k(y_{k-1}, y_{k-2}, \ldots, y_1)$ represents the carry into the k-th position.

Now, one can express

$$C_k(y_{k-1}, y_{k-2}, \ldots, y_1)$$

$$= d_{k-1}y_{k-1} + d_{k-1}C_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1)$$

$$+ y_{k-1}C_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1) \quad (2)$$

$$= d_{k-1}y_{k-1} \oplus d_{k-1}C_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1)$$

$$\oplus y_{k-1}C_{k-1}(y_{k-2}, y_{k-3}, \ldots, y_1) \quad (3)$$

where $C_{k-1}(y_{k-2},y_{k-3},\ldots,y_1)$ is the carry function into the $(k-1)$st position. Substituting (3) in (1), one has

$$g_k(y_{k-1},y_{k-2},\ldots,y_1)$$

$$= d_k \oplus d_{k-1}y_{k-1} \oplus d_{k-1}C_{k-1}(y_{k-2},\ldots,y_1)$$

$$\oplus y_{k-1}C_{k-1}(y_{k-2},\ldots,y_1) \qquad (4)$$

Now

$$g_k(y_{k-1}=0,y_{k-2},\ldots,y_1) = d_k \oplus d_{k-1}C_{k-1}(y_{k-2},y_{k-3},\ldots,y_1) \qquad (5)$$

and

$$g_k(y_{k-1}=1,y_{k-2},\ldots,y_1) = d_k \oplus d_{k-1} \oplus d_{k-1}C_{k-1}(y_{k-2},y_{k-3},\ldots,y_1)$$

$$C_{k-1}(y_{k-2},\ldots,y_1) \qquad (6)$$

From (5) and (6), one has

$$\frac{dg_k}{dy_{k-1}} = d_{k-1} \oplus C_{k-1}(y_{k-2},y_{k-3},\ldots,y_1) \qquad (7)$$

But,

$$Y_{k-1} = y_{k-1} \oplus d_{k-1} \oplus C_{k-1}(y_{k-2},y_{k-3},\ldots,y_1) \qquad (8)$$

Substituting (7) in (8), we get

$$Y_{k-1} = y_{k-1} \oplus \frac{dg_k}{dy_{k-1}}$$

Q.E.D.

Proof of Theorem 4.

Proof: Consider the following table which illustrates the various

			y_n	y_{n-1}	...	y_k	y_{k-1}	...		y_1
			t_n	t_{n-1}	...	t_k	t_{k-1}	...		t_1
			t_1y_n	t_1y_{k-1}	...	t_1y_3	t_1y_2	...		t_1y_1
		t_2y_n	t_2y_{n-1}	t_2y_{k-2}	...	t_2y_2	t_2y_1			
						t_3y_1				
						c_3				
	$t_{k-1}y_n$	$t_{k-1}y$	$t_{k-1}y_{n-k+2}$	$t_{k-1}y_1$						
n		t_ky_{n-k}	t_ky_{n-k+1}	c_{k-1}						
				c_k						
		t_ny_2	t_ny_1							
			c_n							
			Y_n	...	Y_k	Y_{k-1}	...	Y_3	Y_2	Y_1

Table 1. Multiplication Table

terms in the multiplication of $(t_n, t_{n-1}, \ldots, t_1)$ with the variables $(y_n, y_{n-1}, \ldots, y_1)$.

Since the multiplication is performed modulo $N=2^n$, we can discard all the partial product bits to the left of the n-th bit position, as shown by the dotted line.

First, it may be observed that in each column, there may be one or more carry bits in addition to the partial product bits. These carry bits are generated during the summation of the columns to the right of this column, which are then propagated to this position. Let C_k, $2 \leq k \leq n$, represent the ex-or sum or these carry bits in each position. Note that $C_1 = C_2 = 0$.

Since t is a fixed number, C_k is a function of $y_{k-1}, y_{k-2}, \ldots, y_1$ only. Thus, $Y_n = t_1 y_n \oplus t_2 y_{n-1} \oplus \ldots \oplus t_n y_1 \oplus C_{n-1}(y_{n-1}, y_{n-2}, \ldots, y_1)$. Let $g_n(y_{n-1}, y_{n-2}, \ldots, y_1) = t_2 y_{n-1} \oplus \ldots \oplus t_n y_1 \oplus C_{n-1}(y_{n-1}, y_{n-2}, \ldots, y_1)$. Since $t_1 = 1$, one has $Y_n = y_n \oplus g_n(y_{n-1}, y_{n-2}, \ldots, y_1)$. Similarly, Y_k can be expressed as $Y_k = y_k \oplus g_k(y_{k-1}, y_{k-2}, \ldots, y_1)$, where

$$g_k(y_{k-1}, y_{k-2}, \ldots, y_1) = t_2 y_{k-1} \oplus t_3 y_{k-2} \oplus \ldots \oplus t_k y_1 \oplus C_k(y_{k-1}, y_{k-2}, \ldots, y_1) .$$

Now, $g_k(y_{k-1}=0, y_{k-2}, \ldots, y_1) = t_3 y_{k-2} \oplus \ldots \oplus t_k y_1 \oplus C_k(Y_{k-1}=0, y_{k-2}, \ldots, y_1)$ and $g_k(y_k=1, y_{k-2}, \ldots, y_1) = t_2 \oplus t_3 y_{k-2} \oplus \ldots \oplus t_k y_1 \oplus C_k(y_{k-1}=1, y_{k-2}, \ldots, y_1)$.

Thus

$$\frac{dg_k}{dy_{k-1}} = t_2 \oplus C_k(y_{k-1}=0, y_{k-2}, \ldots, y_1) \oplus C_k(y_{k-1}=1, y_{k-2}, \ldots, y_1)$$

$$= t_2 \oplus \frac{dC_k}{dy_{k-1}} \quad . \tag{9}$$

We can express $C_k(y_{k-1}, y_{k-2}, \ldots, y_1)$, which is the ex-or sum of all the carry bits into the k-th position, as:

$$C_k(y_{k-1}, y_{k-2}, \ldots, y_1) = C_k^1(y_{k-1}, y_{k-2}, \ldots, y_1) \oplus C_k^2(y_{k-2}, y_{k-3}, \ldots, y_1).$$

In this, $C_k^1(y_{k-1}, y_{k-2}, \ldots, y_1)$ represents the function for the carry bit produced in the addition of (k-1)st column only. Where as $C_k^2(y_{k-2}, y_{k-3}, \ldots, y_1)$ represents the function for ex-or sum of the carry bits produced in the addition of j-th columns $2 \leq j \leq (k-2)$, and which are propagated into the k-th column. It may be noted that y_{k-1} does not appear anywhere in the 1st through (k-2)nd column. The first time y_{k-1} appears is in the (k-1)-st column. Therefore, the function C_{k-1}^2 is independent of y_{k-1}.

Thus,

$$\frac{dC_k}{dy_{k-1}} = C_k^1(y_{k-1}=0, y_{k-2}, \ldots, y_1) \oplus C_k^1(y_{k-1}=1, y_{k-2}, \ldots, y_1)$$

$$\oplus C_k^2(y_{k-2}, y_{k-3}, \ldots, y_1) \oplus C_k^2(y_{k-2}, y_{k-3}, \ldots, y_1)$$

$$= \frac{dC_k^1}{dy_{k-1}} \tag{10}$$

Consider the function $C_k^1(y_{k-1}, y_{k-2}, \ldots, y_1)$. This, by definition, is the carry bit into k-th column, due to the addition of the terms y_{k-1}, $t_2 y_{k-2}$, $t_3 y_{k-3}$, $\ldots$, $t_{k-1} y_1$, C_{k-1} which appear in (k-1)-st column. This function is represented in the following table.

	y_{k-1}	t_2y_{k-2}	t_3y_{k-3}	...	$t_{k-1}y_1$	C_{k-1}	C_k^1	
	0	0	0	...	0	0	0	
	0	0	0		0	1	0	
	0	0	0		1	0	0	
B_0	0	0	0		1	1	1	← i
	⋮							
	0	1	1	...	1	1		
	1	0	0	...	0	0	1	
	1	0	0		0	1	1	
	1	0	0		1	0	1	
B_1	⋮							
	1	0	0		1	1	1	← i
	⋮							
	1	1	1	...	1	1		

Table 2. The Truth Table for C_k^1.

The function $C_k^1 = 1$ represents all the rows which have 2i,1's for i = odd integers. This follows from the observation that if the number of 1's in the (k-1)-st column is equal to 2, 6, 10, 14, etc., then we get a carry in to the k-th column.

Now, consider the function $\frac{dC_k^1}{dy_{k-1}} = C_k^1(y_{k-1}=0, y_{k-2}, \ldots, y_1) \oplus C_k^1(y_{k-1}=1, y_{k-2}, \ldots, y_1)$. The truth table for this function can be computed in the following way. First, note that $\frac{dC_k^1}{dy_{k-1}}$ is independent of y_{k-1} and hence, the truth table will have half as many rows as the above one for C_k^1. The value of the function $\frac{dC_k^1}{dy_{k-1}}$ for the i-th row can be computed as follows: Take the value of the function C_k^1 for the i-th row in the upper half B_0, and then compute the ex-or sum of this with the value of the function for the i-th row in the lower half B_1 as shown in Table II. The i-th rows in the B_0 and B_1 are identical in the t_2y_{k-1}, t_3y_{k-1}, ..., C_{k-1} positions. Further, it may be seen that if the i-th rows have an even number of 1's in these positions, then the two values of C_k^1 are identical. On the other hand, if they have an odd number of 1's, then C_k^1 has complementary values for these two rows. This, therefore, implies that $\frac{dC_k^1}{dy_{k-1}}$ is equal to 1 for the i-th row only if it has an odd number of 1's in the t_2y_{k-1}, t_3y_{k-1},....,C_{k-1} positions, and is equal to 0 otherwise. From this it can now be deduced that

$$\frac{dC_k^1}{dy_{k-1}} = t_2y_{k-1} \ominus t_3y_{k-2} \oplus \ldots \oplus t_{k-1}y_1 \oplus C_{k-1}.$$ Substituting this first in (10) and then in (9) one has:

$$\frac{dg_k}{dy_{k-1}} = t_2 \oplus t_2 y_{k-1} \oplus t_3 y_{k-2} \oplus \cdots \oplus t_{k-1} y_1 \oplus C_{k-1} \tag{11}$$

But

$$Y_{k-1} = y_{k-1} \oplus t_2 y_{k-1} \oplus t_3 y_{k-2} \oplus \cdots \oplus t_{k-1} y_1 \oplus C_{k-1} . \tag{12}$$

From (11)and (12)

$$Y_{k-1} = y_{k-1} \oplus t_2 \oplus \frac{dg_k}{dy_{k-1}} .$$

Case I: $(t-1)/2$ = even number

In this case, $t_2 = 0$

$$Y_{k-1} = y_{k-1} \oplus \frac{dg_k}{dy_{k-1}}$$

Case II: $(t-1)/2$ = odd number

In this case, $t_2 = 1$

$$Y_{k-1} = y_{k-1} \oplus \frac{dg_k}{dy_{k-1}} \oplus 1$$

Q.E.D.

Proof of Theorem 5

First, we introduce additional notations which will be useful in deriving the results. Let $\{(i, p_k, (i)) \mid 0 \leq i \leq N-1\}$ be any permutation which is admitted by the network in k stages. Let the permutation, p_k, be represented by n-switching functions, Y_n^k, Y_{n-1}^k, ..., Y_1^k, of the variables y_n, y_{n-1}, ..., y_1.

Next, let $\{(i, p_{ks}(i)) \mid 0 \leq i \leq N-1\}$ be that intermediate permutation which is realized at the output of the shuffle network, during the k-th stage, which results, at the outputs of the k-th stage in the permutation, p_k, at the network output. Let this permutation, p_{ks}, be represented by n-switching function, X_n^k, X_{n-1}^k, ..., X_1^k of the variables, y_n, y_{n-1}, ..., y_1.

Finally, let C_i^k represent that control bit used in E_i for the contents of the j-th input during the k-th pass.

It follows from the above notations that for all j, $1 \leq j \leq n$:

$$Y_j^k = \begin{cases} y_j, & \text{for } k=0, \text{ and} \\ Y_j, & \text{for } k=n. \end{cases} \tag{13}$$

Lemma 1:

$$X_j^k = \begin{cases} Y_{j-1}^{k-1}, & \text{for } 2 \leq j \leq n, \text{ and} \\ Y_n^{k-1}, & \text{for } j = 1 \end{cases}$$

Proof: Proof is a direct consequence of Theorem 1, in conjunction with the fact that the outputs of the (k-1)-th pass are fed back to form the inputs to the k-th pass.

Q.E.D.

The following is the proof of Theorem 5.

Proof: For the sake of simplicity, this proof will be developed in two parts:

First, we will show that:

$$Y_n = Y_n \oplus f_n(y_{n-1}, y_{n-2}, \cdots, y_1).$$

Then, we prove, in general that:

$$Y_k = Y_k \oplus f_k (Y_n, Y_{n-1}, \cdots, Y_{k+1}, y_{k-1}, y_{k-2}, \cdots, y_1)$$

Using both Lemma 1 (for k=1) and the relationship (13), one has:

$$X_j^1 = \begin{cases} Y_{j-1} & 2 \leq j \leq n \\ Y_n & j = 1 \end{cases} \tag{14}$$

The following table describes the relationship between X_n^1, X_{n-1}^1, ...,X_1^1 and Y_n^1, Y_{n-1}^1, ..., Y_1^1 .

	X_n^1 X_{n-1}^1 . . .	X_2^1 X_1^1	Y_n^1 Y_{n-1}^1	Y_2^1 Y_1^1
	0 0	0 0	0 0	0 C_0^1
	0 0	0 1	0 0	0 $\bar{C}_1^1$
	0 0	1 0	0 0	1 C_2^1
i	- - - - - - - - - - - -	0	- - - - - - - - - - - -	$-C_i^1$
i+1	- - - - - - - - - - - -	1	- - - - - - - - - - - -	$-\bar{C}_{i+1}^1$
	1 1	1 0	1 1	1 $C_{2^n-2}^1$
	1 1	1 1	1 1	1 $\bar{C}_{2^n-1}^1$

TABLE III: Y and X Variables during the first pass.

The above table is derived by using the following observations regarding the mapping of input addresses to output addresses, as performed by E:

(a) The output pair is identical to the input pair when the input pair is not exchanged.

(b) On the other hand, when an input pair <u>is</u> exchanged, the resulting output pair has the following characteristic:

The binary numbers that represent the output pair are identical to the binary numbers that represent the input pair, in all the positions except the least significant position. The bit in the least significant position of the output pair is exactly the complement of the bit in the least significant position of the input pair.

(c) The least significant bit of any input pair, i, and (i+1) is 0 and 1, respectively, because i is even.

(d) The input pair, i, and (i+1), is exchanged if $C_i^1 = C_{i+1}^1 = 1$; and the pair is not exchanged if $C_i^1 = C_{i+1}^1 = 0$.

As a direct consequence of the above observations, it is evident that all the Y_j's are identical to all of the X_j's, except for j=1. The column, Y_j^1, can be represented as C_1^i and $\bar{C}_{i+1}^1$, in the i-th and (i+1)-th rows, respectively for all i.

From this, it can be derived that:

(1) $Y_j^1 = X_j^1$, $2 \leq j \leq n$,

(2) Y_1^1 is a function of X_n^1 , X_{n-1}^1, ..., X_1^1 .

The function, Y_1^1 , can be expressed, as shown below:

$$Y_1^1 = C_0^1 \bar{X}_n^1 \bar{X}_{n-1}^1 \cdots \bar{X}_2^1 \bar{X}_1^1 \oplus \bar{C}_1^1 \bar{X}_n \bar{X}_{n-1} \cdots \bar{X}_2 X_1$$

$$\oplus \cdots \oplus C_{2^n-2}^1 X_n^1 X_{n-1}^1 \cdots X_2^1 \bar{X}_1^1 \oplus C_{2^n-1}^1 X_n^1 X_{n-1}^1 \cdots X_2^1 X_1^1$$

$$= C_0^1 \bar{X}_n^1 \bar{X}_{n-1}^1 \cdots \bar{X}_2^1 \bar{X}_1^1 \oplus \bar{C}_0^1 \bar{X}_n \bar{X}_{n-1} \cdots \bar{X}_2^1 X_1^1$$

$$\oplus \cdots \oplus \bar{C}_{2^n-2}^1 X_n^1 X_{n-1}^1 \cdots X_2^1 \bar{X}_1^1 + C_{2^n-2} X_n^1 X_{n-i}^1 \cdots X_1^1 \qquad (15)$$

since $C^1 = C_{i+1}^1$, for all even i .

Consider the following well-known identities in Boolean algebra:

(a) $P \oplus Q = P \quad Q$, if $PQ = 0$. (16)

(b) $\bar{P} = 1 \oplus P$ (17)

(c) $P \oplus \bar{P} = 1$ (18)

Using these identities, one can express (15) as:

$$Y_1^1 = X_1^1 \oplus C_0^1 \bar{X}_n^1 \bar{X}_{n-1}^1 \cdots \bar{X}_2^1 \oplus C_2^1 \bar{X}_n^1 \bar{X}_{n-1}^1 \cdots \bar{X}_3^1 X_2^1$$

$$\oplus \cdots \oplus C_{2^n-4}^1 X_n^1 X_{n-1}^1 \cdots X_3^1 \bar{X}_2^1 \oplus C_{2^n-2}^1 X_n^1 X_{n-1}^1 \cdots X_3^1 X_2^1$$

$$= Y_n \oplus C_0^1 \bar{y}_{n-1}^1 \bar{y}_{n-2} \cdots \bar{y}_2 \bar{y}_1 \oplus C_2^1 \bar{y}_{n-1} \bar{y}_{n-2} \cdots \bar{y}_2 y_1$$

$$\oplus \cdots \oplus C_{2^n-4}^1 y_{n-1} y_{n-2} \cdots y_2 \bar{y}_1 \oplus C_{2^n-2}^1 y_{n-1} \bar{y}_{n-2} \cdots y_2 y_1$$

using (13) and Lemma 1.

Let $f_n(y_{n-1}, y_{n-2}, \cdots, y_1)$

$$= C_0^1 \bar{y}_{n-1} \bar{y}_{n-2} \cdots \bar{y}_2 \bar{y}_1 \oplus C_2^1 \bar{y}_{n-1} \bar{y}_{n-1} \bar{y}_{n-2} \cdots \bar{y}_2 y_1$$

$$\oplus \cdots \oplus C_{2^n-4}^1 y_{n-1} y_{n-2} \cdots y_2 \bar{y}_1 \oplus C_{2^n-2}^1 y_{n-1} y_{n-2} \cdots y_2 y_1$$

Thus,

$$Y_1^1 = y_n \oplus f_n (y_{n-1}, y_{n-2}, \cdots, y_1). \tag{19}$$

Using Lemma 1, and the above observation regarding the least significant Y-bit, one can state that, in general:

(a) $Y_j^k = y_{y-1}^{k-1}$, $2 \leq j \leq n$, and

(b) Y_1^k is a function of the variables, y_n^{k-1}, y_{n-1}^{k-1}, ..., Y_1^{k-1}. This, in turn, implies the following, in general, for any k:

$$Y_1^{n-k+1} = Y_k^n = Y_k \quad , \quad 1 \leq j \leq n \ , \tag{20}$$

$$Y_k^j = Y_{n-k+j}^n = Y_{n-k+j} \ , 1 \leq j \leq k \ , \tag{21}$$

$$Y_j^k = Y_{j-k}^0 = Y_{j-k} \ , (k+1) \leq j \leq n \ , \tag{22}$$

Substituting k=n in (20), one has:

$$Y_1^1 = Y_n.$$

Thus, eq. (19) now becomes:

$$Y_n = y_n \oplus f_n (y_{n-1}, y_{n-2}, \cdots, y_1) \ .$$

Now, to prove the theorem, in general, for any Y_k, consider the (n-k+1)-th stage of the network. For the sake of convenience, let n-k+1 be denoted as m.

One can derive the following equation for $Y_1^m = Y_1^{n-k+1}$, by using techniques similar to those used for deriving Y_1^1.

$$Y_1^m = X_1^m \oplus C_0^m \bar{X}_n^m \bar{X}_{n-1}^m \cdots \bar{X}_3^m \bar{X}_2^m \oplus C_2^m \bar{X}_n^m \bar{X}_{n-1}^m \cdots \bar{X}_3^m X_2^m$$

$$\oplus \cdots \oplus C_{2^n-2}^m X_n^m X_{n-1}^m \cdots X_3^m X_2^m \quad . \tag{23}$$

Using Lemma 1 in conjunction with equations (20) - (22), one can deduce the following:

$$X_1^m = X_1^{n-k+1} = Y_k \tag{24}$$

$$X_j^m = X_j^{n-k+1} = Y_{k+j-1} \qquad 2 \le j \le m, \tag{25}$$

$$X_j^m = X_j^{n-k+1} = Y_{j-m}, \qquad m+1 \le j \le n. \tag{26}$$

Substituting (24) - (26) in (23), one has:

$$Y_k = y_k \oplus C_0^m \bar{y}_{k-1} \bar{y}_{k-2} \cdots \bar{y}_1 \bar{Y}_n \bar{Y}_{n-1} \cdots \bar{Y}_{k+2} \bar{Y}_{k+1}$$

$$\oplus C_2^m \bar{y}_{k-1} \bar{y}_{k-2} \cdots \bar{y}_1 \bar{Y}_n \bar{Y}_{n-1} \cdots \bar{Y}_{k+2} Y_{k+1} \oplus \cdots$$

$$\cdots \oplus C_{2^n-2}^m y_{k-1} y_{k-2} \cdots y_1 Y_n Y_{n-1} \cdots Y_{k+2} Y_{k+1}$$

$$= y_k \oplus f_k (Y_n, Y_{n-1}, \ldots, Y_{k+1}, y_{k-1}, y_{k-2}, \ldots, y_1) ,$$

where $f_k (Y_n, Y_{n-1}, \ldots, Y_{k+1}, y_{k-1}, y_{k-2}, \ldots, y_1)$

$$\cdots c_0^m \bar{y}_{k-1} \bar{y}_{k-2} \cdots \bar{y}_1 \bar{Y}_n \bar{Y}_{n-1} \cdots \bar{Y}_{k+2} \bar{Y}_{k+1}$$

$$\oplus\ c_2^m \bar{y}_{k-1} \bar{y}_{k-2} \cdots \bar{y}_1 \bar{Y}_n \bar{Y}_{n-1} \cdots \bar{Y}_{k+2} \bar{Y}_{k+1} \oplus \cdots$$

$$\cdots \oplus c_{2^n-2}^m y_{k-1} y_{k-2} \cdots y_1 Y_n Y_{n-1} \cdots Y_{k+2} Y_{k+1} \quad . \qquad \text{Q.E.D.}$$

Proof of Theorem 7.

Lemma 2:

$$(a_0P_0 \oplus a_1P_1 \oplus \cdots \oplus a_tP_t) * (b_0P_0 \oplus b_1P_1 \oplus \cdots \oplus b_tP_t)$$

$$= (a_0 * b_0)P_0 \oplus (a_1 * b_1)\ P_1 \oplus \cdots \oplus (a_t * b_t)\ P_t \quad ,$$

where (1) a_i's and b_i's are constants

and are equal to 0 or 1.

(2) P_i's are product terms over some variables.

(3) $*$ is any Boolean operation.

(4) $P_i\ P_j = 0$, for all i, j and $i \neq j$

(5) $P_0 \oplus P_1 \oplus \cdots \oplus P_t = 1$.

Proof: Proof is based on the principle of induction, and also on the observation that:

$x * y = k_0 \oplus k_1 x \oplus k_2 y \oplus k_3 xy$, where k_0, k_1, k_2 and k_3 are binary constants.

Q. E. D.

Lemma 3: The control bits that are used in the SSEAC network satisfy the following relationship:

$C_{2i}^{m+1} = C_{2i+1}^{m+1} = C_i^m * C_{i+2^{n-1}}^m$, where $0 \leq i \leq 2^n - 1$, where $*$ is that Boolean operation used during the (m+1)-th pass, to produce the new control bits.

Proof: After the completion of the m-th pass and during the (m+1)-th pass, the contents of the i-th output of the SSEAC network are transmitted through

the S network to the j-th input of the E network.

The following relationship between i and j is derived from the fact that S performs perfect shuffle permutation:

$$i = 2j, \qquad 0 \leq j \leq 2^{n-1} - 1 \tag{27}$$

$$i = 2j - 2^{n-1}, \qquad 2^{n-1} \leq j \leq 2^n - 1 \tag{28}$$

Consider 2i-th and (2i+1)-th inputs of the E network, for some i, $0 \leq i \leq 2^{n-1} - 1$. These are two adjacent inputs. The Boolean operation, $*$, is performed during the (m+1)-th stage on the two control bits that are contained in these inputs. This produces the new control bits for these inputs.

Using (27) and (28), one can see that the contents of these 2i-th and (2i+1)-th inputs to the E network correspond to the contents of the i-th and (i+n-1)-th outputs of the SOAC network, respectively, at the outputs of the m-th stage. Thus, C_i^m and $C_{i+2^{n-1}}^m$ are the two control bits that are contained in 2i-th and (2i+1)-th inputs, respectively, and these are used for generating C_{2i}^{m+1} in the (m+1)-th stage. Hence the Lemma.

Q. E. D.

The following is the proof of Theorem 7.

Proof: During the first stage, the operations of a SOAC network and a Omega network are identical. Consequently, the following equation is also true for SSE networks:

$$Y_n = y_n \oplus f_n (y_{n-2}, \ldots, y_1).$$

Now, in order to prove the relationship between Y_k and Y_{k-1}, the following known fact [6] regarding the control bits may be observed:

During [illegible] m-th pass [illegible] blocks of 2^m [illegible], where the bits within any block are identical [illegible] be expressed in terms of the following equation:

$$Y_i^m = C_{[illegible]}^m \qquad (29)$$

where (a) [illegible] mod 2^m,

(b) $0 \le [illegible] \le 2^m - 1$ and

(c) [illegible] = [illegible] 2^m

Now, consider the (n-k+1)-th **stage of** the network. Let m=n-k+1. The following table describes the relationship between the X and Y variables during the m-th pass. (In deriving the output, Y_1^m, we have used the observations made in the proof of Theorem 5, regarding the least significant bit, as well as using equation (29).

One can derive the following simplified expression for Y_1^m by reducing the sum-of-products expression, using identities of Boolean algebra, such as $P + P = P$, and $P + \bar{P} = 1$.

$$Y_1^m = C_0^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m \bar{X}_m^m \bar{X}_1^m \oplus C_0^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m \bar{X}_m^m X_1^m$$

$$\oplus C_{2^m}^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m X_m^m \bar{X}_1^m \oplus \bar{C}_{2^m}^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m X_m^m X_1^m$$

$$\oplus C_{2^n-2^m}^m X_n^m \cdots X_{m+1}^m X_m^m \bar{X}_1^m \oplus \bar{C}_{2^n-2^m}^m X_n^m \cdots X_{m+1}^m X_m^m X_1^m .$$

Using the identities of Boolean algebra given earlier (16) - (13), one can convert the above expression to the following form:

$$Y_1^m = X_1^m \oplus C_0^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m \bar{X}_m^m$$

$$\oplus C_{2^m}^m \bar{X}_n^m \cdots \bar{X}_{m+1}^m X_m^m \oplus \cdots \oplus C_{2^n-2^m}^m X_n^m \cdots X_{m+1}^m X_m^m \qquad (30)$$

X_n^m	X_{n-1}^m	...	X_{m+1}^m	X_m^m	...	X_2^m	X_1^m	Y_n^m	Y_{n-1}^m	...	Y_{m+1}^m	Y_m^m	...	Y_2^m	Y_1^m
0	0	...	0	0	...	0	0	0	0	...	0	0	...	0	C_0^m
0	0	...	0	0	...	0	1	0	0	...	0	0	...	0	$\bar{C}_0^m$
0	0	...	0	0	...	1	0	0	0	...	0	0	...	1	C_0^m
.															
.															
.															
0	0	...	0	1	...	1	1	0	0	...	0	1	...	1	$\bar{C}_0^m$
0	0	...	1	0	...	0	0	0	0	...	0	0	...	0	$C_{2^m}^m$
0	0	...	1	0	...	0	1	0	0	...	1	0	...	0	$\bar{C}_{2^m}^m$
.															
.															
.															
1	1	...	1	0	...	0	0	1	1	...	1	0	...	0	$C_{2^n-2^m}^m$
1	1	...	1	0	...	0	1	1	1	...	1	0	...	0	$\bar{C}_{2^n-2^m}^m$
.															
.															
.															
1	1	...	1	1	...	1	1	1	1	...	1	1	...	1	$\bar{C}_{2^n-2^m}^m$

Now, consider the next stage (the (n-k+2)-th stage) through the network. Using techniques similar to those used for deriving Y_k, one can derive the following:

$$Y_{k-1} = Y_{k-1} \oplus f_{k-1}(y_{k-2}, \cdots y_2, y_1)$$

where

$$f_{k-1}(y_{k-2}, \cdots, y_2, y_1)$$

$$= C_0^{m+1} \bar{y}_{k-2} \cdots \bar{y}_2 \bar{y}_1 \oplus C_{2^m}^{m} \bar{y}_{k-2} \cdots \bar{y}_2 y_1$$

$$\oplus \cdots \oplus C_{2^{n-1}-2^m}^{m} y_{k-2} \cdots y_2 y_1, \qquad (34)$$

From (35), one has:

$$f_k(y_{k-1} = 0, y_{k-2}, \ldots, y_1)$$

$$= C_0^m \bar{y}_{k-2}, \ldots \bar{y}_2 \bar{y}_1 \oplus C^m \bar{y}_{k-2}, \ldots \bar{y}_2 y_1 \oplus \cdots \oplus C_{2^{n-1}-2^m}^m y_{k-2}, \ldots y_2 y_1,$$

and

$$f_k(y_{k-1} = 1, y_{k-2} \ldots, y_1) \qquad (35)$$

$$= C_{2^{n-1}}^m \bar{y}_{k-2}, \ldots \bar{y}_2 \bar{y}_1 \oplus C_{2^{n-1}+2^m}^m \bar{y}_{k-2}, \ldots \bar{y}_2 y_1 \cdots + C_{2^n-2^m}^m y_{k-2}, \ldots y_2 y_1$$

Thus,

$$f_k(y_{k-1} = 0, y_{k-2}, \ldots, y_1) * f_k(y_{k-1} = 1, y_{k-2}, \ldots, y_1)$$

$$= (C_0^m \bar{y}_{k-2} \cdots \bar{y}_2 \bar{y}_1 \oplus \cdots \oplus C_{2^{n-1}-2^m}^m y_{k-2} \cdots y_2 y_1)$$

$$* (C_{2^{n-1}}^m \bar{y}_{k-2} \cdots \bar{y}_2 \bar{y}_1 \oplus \cdots \oplus C_{2^n-2^m} y_{k-2} \cdots y_2 y_1)$$

$$= (C_0^m * C_{2^n-1}) \bar{y}_{k-2} \cdots \bar{y}_2 \bar{y}_1 \oplus (C_{2^m}^m * C_{2^m+2^{n-1}}^m) \bar{y}_{k-2} \cdots \bar{y}_2 y_1$$

$$\oplus \cdots \oplus (C_{2^{n-1}+2^m}^m * C_{2^n+2^m}^m) y_{k-2} \cdots y_2 y_1 \qquad (36),$$

using Lemma 3.

Using Lemma 3, one has:

$$C_0^{m+1} = C_0^m * C_{2^{n-1}}^m$$

$$C_{2^m+1}^{m+1} = C_{2^m}^m * C_{2^{n-1}+2^m}^m$$

.

.

.

$$C_{2^n-2^{m+1}}^{m+1} = C_{2^{n-1}-2^m}^m * C_{2^n-2^m}^m .$$

Substituting these in (35), one has the expression given in (34). Hence, the proof.

Q. E. D.