AU J.T. 13(3): 143-150 (Jan. 2010)

An Approach for Exploring Combinatorial Properties of *R*-path Omega Interconnection Networks

Gennady Veselovsky

Faculty of Engineering, Assumption University
Bangkok, Thailand
E-mail: <gveselovsky@au.edu>

Abstract

Multiple path multistage interconnection networks as compared to their parental one path cube-type networks provide better permutation capability and offer such important property as fault-tolerance. In this paper an approach for investigation permuting ability of R-path Omega networks is proposed. The analysis as concerned to arbitrary permutations is done with the help of number theory methods. As to BPC (bit-permute-complement) permutations admissibility check to the aforementioned type of networks, the modified window method is proposed. The aforementioned method reduces drastically the time complexity of admissibility check. The results of computational experiments with applying the technique to some permutations of BPC class are given.

Keywords: *Interconnection networks, R-path Omega network, BPC permutations, window method.*

Introduction

Multistage interconnection networks are of interest for use in large-scale parallel computer systems. The class of multistage, self-routing networks which in its basic form requires $\log_2 N$ stages of 2 x 2 switches to connect N inputs to N outputs with *NI2* switches are needed in each stage is known as a class of cube-type networks. This class includes Omega (Lawrie 1975), Generalized Cube (Siegel and Smith 1978), indirect binary *n*-cube (Pease 1978), delta (Patel 1981), to name but a few. The problem with this topology is that there is only one path from a given network input to a given network output so it is vulnerable to component faults. The fault-tolerance in the case of a multistage network requires multiple disjoint paths for each input-output pair. It can be achieved by either adding one or more stages in front of the original topology (Adams and Siegel 1982; Shen 1995) or by using basic switches with the number of inputs and outputs greater than 2. The authors of the latter approach (Padmanabhan and Lawrie 1983) introduce the concept of a general class of multistage networks that preserves the connection properties of the Omega network and at the same time provides significant tolerance to faults in the form of mlu1tiple disjoint paths for any input-output pair. Padmanabhan and Lawrie (1983) proved that non-faulty multiple path modified Omega network would pass any permutation that one path Omega network passed. In this paper the conditions for blocking occurrence when realizing arbitrary permutations on *R*-path Omega networks are stated. Those conditions are based on congruence notion from number theory. For checking admissibility of regular BPC (bit-permute-complement) permutations the modified window method based rather on symbolic transition sequence than on transition matrix analysis is proposed. The method reduces the time complexity of the check procedure drastically. All work is done in assumption of non-faulty condition of the networks under consideration. This paper is a generalization of research results represented earlier at two international conferences

(Veselovsky and Kupriyanova 1995; Veselovsky and Thaiyoo 2004).

Preliminaries and Definitions

A permutation in the context of a parallel computing system means simultaneous transferring of a data item from each node (processor) to another node, with all destination nodes (processors) being different. Permutations belong to the basic communication patterns in SIMD (single-instruction stream, multiple data stream) parallel computers, they can occur in MIMD (multiple instruction stream, multiple data stream) parallel computers as well when a communication is preceded by a barrier synchronization (Liszka *et al*.). Permutations occur in two main types: arbitrary and regular. By the statement that a permutation is regular it is meant that there is a common rule for producing a destination address from a source address for all input-output pairs in a given permutation, otherwise a permutation is arbitrary. Regular permutations play important role in parallel programming and there is a variety of classes of regular permutations, among which the class of BPC (bit-permute-complement) permutations is one of the most commonly used. A permutation is called a BPC permutation if the destination address can be obtained of the source address by permuting bits in the source address and/or complementing some or all of its bit positions. This permutation class meets most requirements imposed upon information exchanges of permutation type for a number of important applications. There are known most frequently used permutations which belong to BPC class (Grammatikakis *et al*.). Such permutations usually have names and they are referred to by their names.

A permutation is said to be admissible to an interconnection network if it doesn't cause conflicts (blockings) in process of realizing it under a chosen routing algorithm. The analytical approach proposed in this paper is based on the congruence notion from number theory. Lawrie pioneered in applying this formalism when exploring permuting ability of a one-path Omega network (Lawrie 1975), here similar approach is applied to *R*-path Omega networks if arbitrary permutations are considered. The Definition 1 that follows has been taken from Rosen (1999).

Definition 1: If a and b are integers and m is a positive integer, then a is congruent to b modulo m if m divides $a - b$. It means that in binary notation of the numbers a and b, $log_2 m$ lower-order bits are identical.

The notation $a \equiv {}_m b$ is used to indicate that a is congruent to b modulo m. If a and b are not congruent modulo m one can write that $a \not\equiv {}_m b$.

Definition 2: The notation $a \equiv {}^m_N b$ implies that the groups of bits (in other words, the "windows") that are shaped by cutting off $log_2 m$ lower-order bits from $log_2 N$ lower-order bits in the binary representation of a and b are identical.

For example, let $a = 1\mathit{101}0$ (binary), $b = 0\mathit{101}0$, and $c = 1\mathit{001}0$. Then $a \equiv {}_4 b \equiv {}_4 c$, $a \equiv {}^2_{16} b$ and $a \not\equiv {}^2_{16} c$ (since $\mathit{101} \neq \mathit{001}$).

The foregoing examples were taken from Lawrie (1975).

Here a permutation P_N is defined as a set of the integer pairs $P_N = \{(S_i, D_i), 0 \le i < N\}$ which represent a mapping of sources to destinations $S_0 \rightarrow D_0$, $S_1 \rightarrow D_1$, ..., $S_{N-1} \rightarrow D_{N-1}$. Its components will be considered as n-dimensional vectors whose elements are either 0 or 1: the vector $(s_{n-1} s_{n-2} \ldots s_0)$ is identified with the integer S_i, s_{n-1} being the most significant bit.

Suppose an N x N modified Omega network with B x B switches. In accordance with Padmanabhan and Lawrie (1983) if $N = 2^n$ and $B = i$ a modified Omega network consists of $K = \lceil \log_B N \rceil$ identical stages of BxB switching elements with each stages interconnected by the $B * NIB$ shuffle. Such a connection has the property of taking an input

at position i and moving it to position $\pi = (Bi + [i / NB]) \bmod N$, $0 \le i \le N$.

Under foregoing conditions the modified Omega network is an R-path Omega network where $R = B^{[n/b]}$.

In what follows, Omega networks shall be referred to as 1-path Omega networks as defined in Lawrie (1975). Modified Omega networks with a redundancy of R will be called R-path Omega network in accordance with the terminology used in Padmanabhan and Lawrie (1983).

A 4-path 16 x 16 Omega network is shown in Fig. 1. It is constructed using 8 x 8 cross-point switches and the 8 * 2 shuffle connection. To establish a connection in an R-path Omega network between source $S = s_0 s_1 ... s_{n-1} * ... * d_0 d_1 ... d_{n-1}$ and destination $D = d_0 d_1 ... d_{n-1}$ the transition sequence of the following form is used:

$$s_0 s_1 ... s_{n-1} * ... * d_0 d_1 ... d_{n-1}. \qquad (1)$$

In a transition sequence asterisks define bit positions necessary for selecting one route among the set of alternative routes for a given input-output pair with the number of these positions r being equal $\log_2 R$. The output terminal at stage i $(0 \le i \le \lceil \log_B N \rceil)$ is given by the n-bit window in a transition sequence starting at bit position bi.

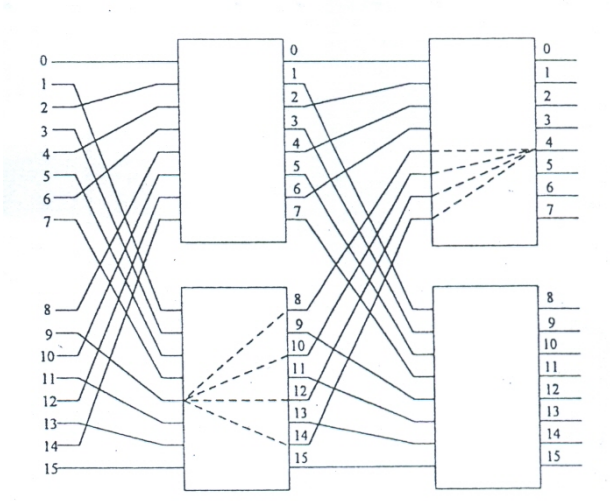

Fig. 1. A 4-path 16 x 16 modified Omega network.

Conditions of Blocking Occurrence for an Arbitrary Permutation

The set of transition sequences in a binary form for all input-output pairs of a given arbitrary permutation forms so-called transition matrix. For detecting a blocking at one or another stage of a network the comparing contents within a current window position is needed, in this case the coincidence of any two sequences means that two input-output pairs require the same terminal, i.e. a blocking occurs. The check can be done using an admissibility algorithm similar to that proposed in Shen (1995b) for base 2 multistage interconnection networks. The main distinctive feature in the case considered in this contribution is that a window at each step should be shifted by b positions but not by one. However the mathematical analysis is also possible. The conditions for blocking occurrence after the k-th stage in an R-path Omega network basing on the notion of congruence are stated, as follows:

$S_i \not\equiv_N S_j, S_i \equiv_v S_j, D_i \equiv^{v+R}_N D_j$, $E_i \equiv_R E_j$, with $v = 2^{n-bk}$, E_i and E_j being extra bit sequences in input-output pairs $(S_i D_i)$ and $(S_j D_j)$ respectively. The foregoing relations help to simplify the check for admissibility and can be also helpful for assigning the values of extra bits when routing an arbitrary permutation.

Admissibility Algorithm for BPC Permutations

The nature of BPC permutations when checking their admissibility allows to replace the analysis of a transition matrix (Shen 1995) with analysis only of a transition sequence given in a symbolic form. Such modification of the widely known window method reduces its time complexity drastically. In accordance with the definition given above the transition sequence for a BPC permutation π in a general case looks as follows:

$$s_0 s_1 ... s_{n-1} * ... * s_{\pi(0)} s_{\pi(1)} ... s_{\pi(n-1)} \qquad (2)$$

It should be noted that some or all components of the destination part of the foregoing sequence may be complemented but for the approach proposed here it makes no difference as it can be seen from the further discussion. Hereafter components of a transition sequence (2) given in above form will be called symbolic components in distinction to binary components like in transition sequence (1). It should be recalled that when routing each window defines terminal numbers in one of stages of the network. So to avoid blockings each window should allow forming different terminal numbers in a given stage for all input-output pairs but it is possible only if there are no symbolic components with the same subscripts, or like components, inside the window, otherwise the complete set of terminal numbers cannot be formed. Hence the window method provides a straightforward way to check the admissibility of any BPC permutation to an *R*-path Omega network of a given structure. It is quite evident that in this case only replica of components with the same subscripts within a window is of importance, it makes no difference whether such two components are mutual complements or not.

Generally the admissibility algorithm may be conceived as follows.

*St*ep 1: Define as above the transition sequence (2) for a given BPC permutation π for *R*-path Omega network with a specified configuration. Denote by i positions of bits in the transition sequence ($0 \le i \le 2n + r - 1$).

*St*ep 2: Fix the window of length n originally at $i = b$ position. (At the inputs and outputs of a network blockings are not possible on the definition of a permutation).

*St*ep 3: For the current window position compare each of symbolic components inside the window to the left of extra bits, with each of symbolic components inside the window but to the right of extra bites. At the first coincidence of a component to the left of extra bites with a component to the right of extra bits go to "Not admissible". If no like components to the left and to the right of extra bits in a window found then go to step 4.

*St*ep 4: Shift the window by b positions to the right, if the new position $i \le n - 1$ then go to Step 3 else go to "Admissible".

The problem being solved in this section is most closely resembles the problem of BPC permutation admissibility (PA) solved in Shen (1995). However, Shen's investigations are in concern with another class of interconnection networks, namely with *k*-extra stage multistage cube-type networks (*k*-EMCTN) where redundant disjoint paths for any input-output pair are provided by adding k more stages in front of an Omega network implemented of switches with size 2 x 2. Moreover, as it has been mentioned already, Shen's approach is based on complete transition matrix analysis. Nevertheless, the approach proposed here can be easily adjusted to *k*-EMCTN.

Results of Computational Experiments

Permutations which are known as the most frequently used and at the same time belong to the BPC class (Grammatikakis *et al*. 1998) are listed below, where each equation shows the mapping π of a source $S = s_0 s_1 s_2 ... s_{n-2} s_{n-1}$ to the destination.

Perfect shuffle:

$$\pi_{perfectshuffle} = s_1 s_2 ... s_{n-1} s_0$$

Unshuffle:

$$\pi_{\text{unshuffle}} = s_{n-1}\ s_0 \ldots\ s_{n-3}\ s_{n-2}$$

Bit shuffle:

$$\pi_{\text{bit shuffle}} = \begin{cases} s_0 s_2 \ldots s_{n-2} s_1 s_3 \ldots s_{n-1}, \\ if\ \ n = 2l \\ \\ s_0 s_2 \ldots s_{n-1} s_1 s_3 \ldots s_{n-2}, \\ if\ \ n = 2l + 1 \end{cases}$$

Vector reversal:

$$\pi_{\text{vector reversal}} = \bar{s}_0\ \bar{s}_1\ \ldots\ \bar{s}_{n-2}\ \bar{s}_{n-1}$$

Exchange:

$$\pi_{\text{exchange}} = s_0\ s_1 \ldots\ s_{i-1}\ \overline{s}_i\ s_{i+1}\ \ldots\ s_{n-2}\ s_{n-1}$$

Matrix transposition:

$$\pi_{\substack{\text{matrix}\\ \text{transposition}}} = \begin{cases} s_l\ s_{l+1} \ldots s_{2l-1}\ s_0\ s_1 \ldots\ s_{l-1}, \\ if\ \ n = 2l \\ \\ s_l\ s_{l+1} \ldots s_{2l}\ s_0\ s_1 \ldots\ s_{l-1}, \\ if\ \ n = 2l+1 \end{cases}$$

Shuffle row major:

$$\pi_{\substack{\text{shuffle}\\ \text{row major}}} = \begin{cases} s_0\ s_l\ s_1\ s_{l+1} \ldots s_{l-1}\ s_{2l-1} \\ if\ \ n = 2l \\ \\ s_0\ s_{l+1}\ s_1\ s_{l+2} \ldots s_{l-1}\ s_{2l}\ s_l \\ if\ \ n = 2l+1 \end{cases}$$

Bit reversal:

$$\pi_{bitreversal} = s_{n-1} s_{n-2} \ldots s_1 s_0$$

Butterfly:

$$\pi_{butterfly} = s_{n-1} s_1 \ldots s_{n-2} s_0$$

Of the above list one of especially important and commonly used in parallel programming is *perfect shuffle* permutation. Before proceeding further, an example of applying the proposed approach for testing *perfect shuffle* admissibility to 1-path and 2-path Omega networks is considered. In both cases, suppose $N = 32$ or $n = 5$.

It can be seen in Fig. 2 that symbolic numbers of terminals in windows have same components (in the window W_1 component S_1 appears twice, the same about component S_2 in the window W_2 etc.) thus in this case perfect shuffle is not admissible. For a 2-path Omega network of size $N = 32$, and with $B = 4$ or $b = 2$, only one extra bit is needed. The transition sequence with windows for perfect shuffle in this case looks as shown in Fig.3.

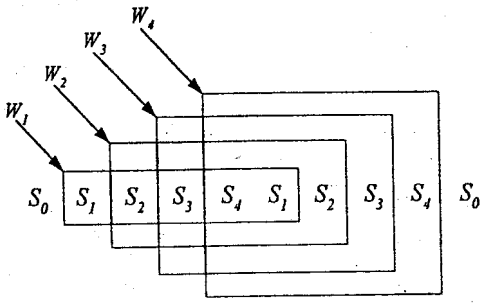

Fig. 2. Windows for the *perfect shuffle* permutation on 1-path 32x32 Omega network.

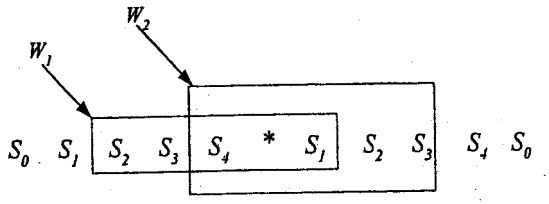

Fig. 3. Windows for the *perfect shuffle* permutation on a 2-path 32x32 Omega network.

Moving the window of length 5 bit, starting at each step in a $2i$ position, results in combinations of components inside two windows without the same subscripts, so the proper choice of extra bit values may provide the complete set of 5-bit binary numbers corresponding to the physical terminals and doing so avoid conflicts (blockings). Fig. 4 and Fig.5 demonstrate perfect shuffle permutation routing for two above examples, with conflicts in 1-path Omega network and conflict-free in 2- path Omega network, respectively.

A table with possible redundancy values R which can be obtained in an NxN modified Omega network and the sizes B (smallest) of the switches to be used for N up to 2,048 is given in Padmanabhan and Lawrie (1983).

Tables 1, 2 and 3 given below represent the results of testing admissibility of aforesaid frequently used BPC permutations to R-path Omega networks with N equal 128, 256 and 512, respectively, for all possible redundancies R and proper switch sizes.

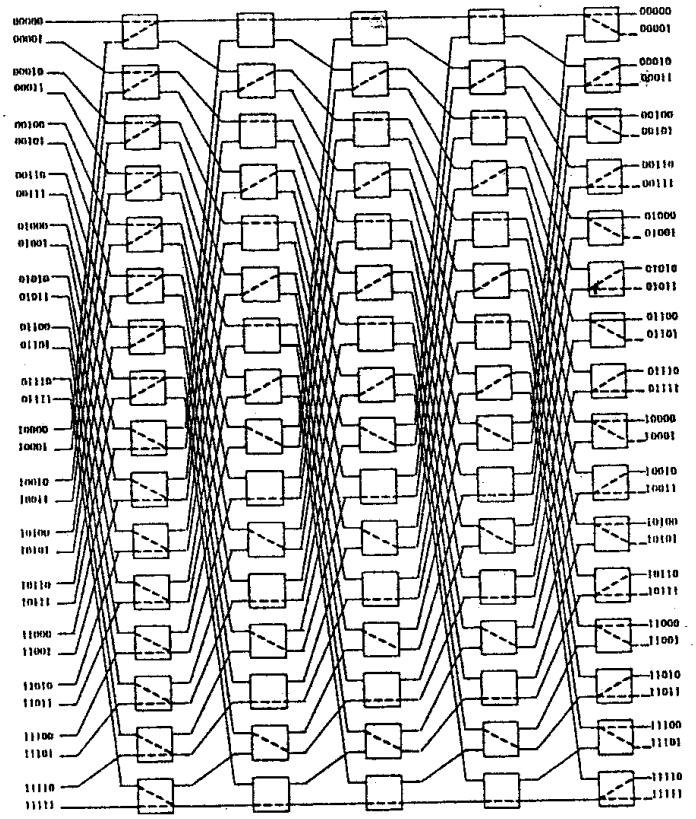

Fig. 4. A 1-path 32x32 Omega network: *perfect shuffle* permutation causes blockings.

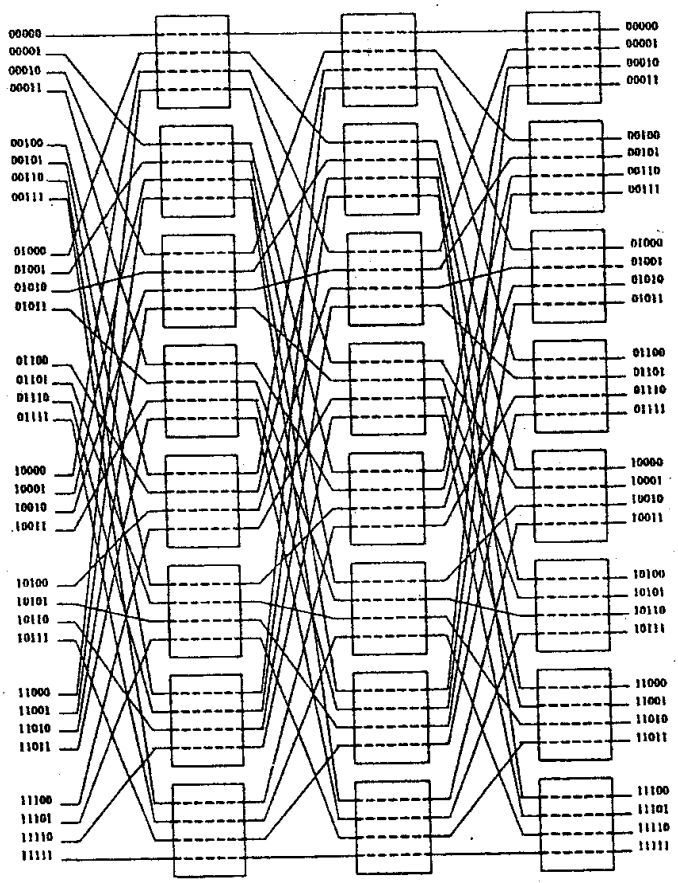

Fig. 5. A 2-path 32x32 Omega network with switches set up to realize *perfect shuffle* permutation.

Table 1. Admissibility of frequently used BPC permutations to *R*-path Omega networks for *N*=128.

Permutation Type	Redundancy (*R*)				
	1	2	4	8	32
Perfect Shuffle	No	Yes	Yes	Yes	Yes
Unshuffle	No	No	No	No	No
Bit Shuffle	No	No	Yes	Yes	Yes
Bit Reversal	No	No	No	No	No
Vector Reversal	Yes	Yes	Yes	Yes	Yes
Exchange	Yes	Yes	Yes	Yes	Yes
Butterfly	No	No	No	No	No
Matrix Transpo-sition	No	No	No	Yes	Yes
Shuffle Row Major	No	No	Yes	Yes	Yes

Table 2. Admissibility of frequently used BPC permutations to *R*-path Omega networks for *N*=256.

Permutation Type	Redundancy (*R*)				
	1	2	4	8	32
Perfect Shuffle	No	Yes	Yes	Yes	Yes
Unshuffle	No	No	No	No	No
Bit Shuffle	No	No	Yes	Yes	Yes
Bit Reversal	No	No	No	No	No
Vector Reversal	Yes	Yes	Yes	Yes	Yes
Exchange	Yes	Yes	Yes	Yes	Yes
Butterfly	No	No	No	No	No
Matrix Transpo-sition	No	No	No	Yes	Yes
Shuffle Row Major	No	No	Yes	Yes	Yes

"Yes" in a box means that the permutation given to the left with the redundancy given above is admissible, whereas "No" means on the contrary that it is not admissible.

From these tables it can be seen that of nine permutations under study1-path Omega network can realize for one pass only *vector reversal* and *exchange* permutations. Such useful and commonly used in parallel programming permutation as *perfect shuffle* is admissible if $R \geq 2$, other parameters make no difference.

On the contrary *unshuffie*, *butterfly*, and *bit reversal* are not admissible to *R*-path Omega networks in any case, whereas admissibility of *matrix transposition*, *bit shuffle*, and *shuffle row major* depends on redundancy (*R*) and network size (*N*).

For some of BPC permutations in question it is not difficult to ascertain if there exists an *R*-path Omega network configuration which allows to realize a given permutation for one pass or not.

The following reasoning is based on a fundamental property of *R*-path Omega networks which implies that the number of extra bits *r* in a transition sequence is always less than or equals to $n - 2$, i.e., $r \leq n - 2$. (It will be remembered that in the general case $r = \log_2 R$).

If the number of bits, including extra ones, between two like components in a transition sequence does not exceed $n - 2$ a window of length *n* can cover both of them and so a permutation is not admissible.

However, if the minimal number of bits between two like components in a transition sequence exceeds $n - 2$, a given permutation is admissible to a network.

Table 3. Admissibility of frequently used BPC permutations to *R*-path Omega networks for *N*=512.

Permutation Type	Redundancy (*R*)				
	1	2	4	8	32
Perfect Shuffle	No	Yes	Yes	Yes	Yes
Unshuffle	No	No	No	No	No
Bit Shuffle	No	No	No	Yes	Yes
Bit Reversal	No	No	No	No	No
Vector Reversal	Yes	Yes	Yes	Yes	Yes
Exchange	Yes	Yes	Yes	Yes	Yes
Butterfly	No	No	No	No	No
Matrix Transpo-sition	No	No	No	Yes	Yes
Shuffle Row Major	No	No	Yes	Yes	Yes

For the *unshuffle*, *butterfly* and *bit reversal* permutations the S_{n-1} like components in source and destination parts of a transition sequence respectively are separated by exactly *n - 2* bits in the best case, so there does not exist any configuration of an *R*-path Omega network to which aforesaid permutations are admissible.

Conclusion

Conditions for blocking occurrence in an *R*-path Omega network based on congruence notion from number theory are stated. The aforesaid conditions are true for a permutation of any type possible in parallel programming provided the routing algorithm is known. A modified window method based on transition vector analysis in contrast to transition matrix analysis (Shen 1995) for testing BPC permutations admissibility to *R*-path Omega networks is introduced. It is shown that the permutation capability of *R*-path Omega networks under a no-fault condition is much better than that of 1-path Omega networks. Of the nine most frequently used in parallel programming BPC permutations 1-path Omega network can realize for one pass only two, namely vector reversal and exchange , whereas *R*-path Omega network realizes up to six of them including those which admissibility depends on redundancy and size of a network. It should be mentioned that such important permutation as perfect shuffle whose extreme usefulness in parallel programming is widely known found to be admissible to *R*-path Omega network of any size if $R \geq 2$. The approach holds the promise to be efficient for permuting ability analysis of *k*-extra stage multistage cube-type networks (*k*-EMCTN) including optical ones.

References

Adams III, G.B.; and Siegel, H.J. 1982. The extra stage cube: a fault-tolerant interconnection network for supersystems. IEEE Trans. Computers 31(5): 443-54, May.

Grammatikakis, M.D.; Hsu, D.F.; Kraetzl, M.; and Sibeyn, J.F. 1998. Packet routing in fixed-connection networks: A survey. J. Parallel and Distributed Computing 54(2): 77-132, November.

Lawrie, D.H. 1975. Access and alignment of data in an array processor. IEEE Trans. Computers 24(2): 1145-55, December.

Liszka, K.J.; Antonio, J.K.; and Siegel, H.J. 1997. Is an alligator better than an armadillo? IEEE Concurrency 5(4): 18, 20-28, October-December.

Padmanabhan, K.; and Lawrie, D.H. 1983. A class of redundant path multistage interconnection networks. IEEE Trans. Computers 32(12): 1099-108, December.

Patel, J.H. 1981. Performance of processor-memory interconnections for multiprocessors. IEEE Trans. Computers 30(10): 771-80, October.

Pease III, M.C. 1977. The indirect binary *n*-cube microprocessor array. IEEE Trans. Computers 26(5): 458-73, May.

Rosen, K.H. 1999. Discrete mathematics and its applications. McGraw-Hill International Editions, New York, NY, USA.

Shen, X. 1995. Optimal realization of any BPC permutation on *k*-extra-stage Omega networks. IEEE Trans. Computers 44(5): 714-9, May.

Shen, X. 1995b. An optimal $O(N\lg N)$ algorithm for permutation admissibility to extra-stage cube-type networks. IEEE Trans. Computers 44(9): 1144-9, September.

Veselovsky, G.G.; and Kupriyanova, M.V. 1995. A study of combinatorial properties of multiple path dynamic interconnection networks. Proceedings of the Fifth International Workshop on Distributed Data Processing, Novosibirsk, Russia, 10-12 October 1995, pp. 28-32 (In Russian).

Veselovsky, G.; and Thaiyoo, P. On BPC permutations admissibility to *R*-path Omega networks. Proc. Int. Conf. on Parallel & Distributed Processing Techniques and Applications (PDPTA'04), Monte Carlo Resort, Las Vegas, NV, USA, 21-24 June 2004, vol. III, pp. 1107-13, CSREA Press.