Decomposing BPC Permutations into Semi-Permutations for Crosstalk Avoidance in Multistage Optical Interconnection Networks

Gennady Veselovsky, Ritu Jain
Department of Computer and Network Engineering
Assumption University
Soi 24, Ramkhamhaeng Road, Huamark
Bangkok 10240, Thailand [1]

Abstract - *This paper introduces a simple O(N) algorithm that decomposes BPC (bit-permute-complement) permutations into semi-permutations for avoiding crosstalk when realizing them in N × N optical multistage interconnection networks (OMINs). Crosstalk means that two optical signals, sharing an optical switch, undergo a kind of undesired coupling. A semi-permutation is a partial permutation which meets the requirement for each switch in an input and output stages of the network to be used with only one optical signal at a time. It provides avoiding crosstalk in the first and the last stages of a network and creates the potential for crosstalk-free realization of a semi-permutation, and finally the whole permutation in question. The algorithm is based on employment the periodicity of appearing 1's and 0's in columns of transition matrices for BPC permutations.*

Keywords: *Multistage optical interconnection networks, BPC permutations, semi-permutations, crosstalk avoidance.*

1 Introduction

The vast number of processing elements in massively computers dictates heavy demands for performance of an interconnection network in use. Optical interconnection networks constitute a promising choice in the field because they offer gigabit transmission capacity, very big bandwidth, and low error probability. In what follows we consider hybrid optical multistage interconnection networks (OMINs) using guided wave technology and composed of electronically controlled directional couplers because other types of optical networks are still difficult to implement. In spite of the topological similarity of electronic and optical multistage interconnection networks, the latter cause some specific problems, and the major of them is optical crosstalk. It means that two optical signals, sharing an optical switch, undergo a kind of undesired coupling. The crosstalk reduces essentially signal to noise ratio and so limits the size of a network, whereas the number of processing nodes in modern supercomputers is increasing rapidly. To avoid crosstalk two optical signals should be sent at different time, if they use the same switch. It is called *time domain* approach in distinction from *space domain* approach when crosstalk avoidance is achieved with significant increase of hardware. Crosstalk avoiding problem is discussed in a large number of works, e.g. in [1], [2], [3], [4], [5], [6], to mention only a few. In what follows we have to refer often to the term *switch conflict*. It means using an optical switch (directional coupler) by two input signals at the same time resulting in above mentioned crosstalk. It is noteworthy also that for optical hybrid OMINs under consideration circuit switching rather than packet switching is usually preferred, since with packet switching the address information in each packet must be decoded in each stage that means conversion from optical signal to electronic and so can be very costly.

A permutation is one of the most common communication patterns in parallel computing systems. In the context of a parallel computing system, a permutation means simultaneous transferring of data items between the nodes, with all the destination nodes being different. The term *permutation* can be defined as a request for parallel connection of N sources to N destinations, where $N = 2^n$, with a distinct destination for each of the sources: $S_0 \to D_0, S_1 \to D_1, \ldots, S_{N-1} \to D_{N-1}$. Its components will be considered as n-dimensional vectors whose

The research has been funded by Assumption University of Thailand

elements are either 0 or 1: the vector $s_0 s_1 \ldots s_{n-2} s_{n-1}$ is identified with the integer S_i, s_0 being the most significant bit. The following sequence $s_0 s_1 \ldots s_{n-2} s_{n-1} d_0 d_1 \ldots d_{n-2} d_{n-1}$ is called a *transition sequence* for an input-output pair. The set of all transition sequences for a given permutation is called its *transition matrix* [7]. If a permutation does not possess any regularity, it is called an arbitrary permutation. On the contrary, if there is some general rule for producing a destination address from a source address, a permutation is called a regular one. There are known different classes of regular permutations. A permutation is called a bit-permute-complement (BPC) permutation if the destination address can be produced by permuting bits in the source address and/or complementing some or all of its bits positions. BPC permutations are widely used in parallel programming when solving various scientific problems (digital signal processing, large matrices processing etc). A transition matrix for a BPC permutation in a symbolic form can be represented as follows:

$$S_0 S_1 \ldots S_{n-2} S_{n-1} S_{\pi(0)} S_{\pi(1)} \ldots S_{\pi(n-2)} S_{\pi(n-1)} \text{[7].}$$

A semi-permutation is a partial permutation which meets the requirement for each switch in an input and output stages of the network to be used with only one optical signal at a time. It provides avoiding crosstalk in the first and the last stages and creates the potential for crosstalk-free realization of semi-permutations, and finally the whole permutation in question [1]. It is evident that for crosstalk-free realization of a semi-permutation crosstalk in intermediate switches also should be eliminated, and it is the next step in crosstalk avoiding. However, decomposing a permutation into semi-permutations can be considered as a specific problem. The problem is being solved for arbitrary permutations in [1], however, to our best knowledge decomposing BPC permutations to semi-permutations was considered before only in the work with participation of the first author of this paper but for some specific representatives of that class [8]. In this paper a fast algorithm for decomposing BPC permutations in general into semi-permutations is presented. Its time complexity is *O(N)*, where $N \times N$ is the size of a network. The algorithm presented in the paper is based on the regularity of BPC permutations, namely, on periodicity of appearing 1s and 0s in columns of their transition matrices. The algorithm is implemented in C language for two basic types of optical multistage interconnection networks (OMINs), namely with perfect shuffle interconnection pattern before the first stage and without it. In our work some computational experiments were carried out, including those in concern with BPC permutations known as worst cases from the viewpoint of routing. The total number of BPC permutations is $2^n n!$. There are some frequently used BPC permutations as given in [9] and in [10]. Such permutations are usually referred to by names, with each equation showing mapping a source $s_0 s_1 \ldots s_{n-2} s_{n-1}$ to the destination. The permutations which are referred to in our paper were taken from the list below:

1. *Perfect shuffle*

$$\pi_{PSH} = s_1 s_2 \ldots s_{n-1} s_0$$

2. *Unshuffle*

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

3. *Vector reversal*

$$\pi_{VR} = \overline{s_0}\,\overline{s_1} \ldots \overline{s_{n-2}}\,\overline{s_{n-1}}$$

4. *Butterfly*

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

5. *Exchange*

$$\pi_{EXCH} = s_0 s_1 \ldots s_{i-1} \overline{s_i} s_{i+1} \ldots s_{n-1} s_{n-2}$$

6. *Bit reversal*

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

7. *Matrix transpose*

$$\pi_{MT} = s_l s_{l+1} \ldots s_{2l-1} s_0 s_1 \ldots s_{l-1} \text{ if } n=2l$$

$$\pi_{MT} = s_l s_{l+1} \ldots s_{2l} s_0 s_1 \ldots s_{l-1} \text{ if } n=2l+1$$

8. *Bit shuffle*

$$\pi_{BSH} = s_0 s_2 \ldots s_{n-2} s_1 s_3 \ldots s_{n-1} \text{ if } n=2l$$

$$\pi_{BSH} = s_0 s_2 \ldots s_{n-1} s_1 s_3 \ldots s_{n-2} \text{ if } n=2l+1$$

9. *Shuffle row major*

$$\pi_{SHRM} = s_0 s_l s_1 s_{l+1} \ldots s_l s_{2l-1} \text{ if } n=2l$$

$$\pi_{SHRM} = s_0 s_{l+1} s_1 s_{l+2} \ldots s_{l-1} s_{2l-1} s_l \text{ if } n=2l+1$$

2 An algorithm for decomposing BPC permutations into semi-permutations

The essential part of the algorithm consists in determining so called *antagonists*, i.e. the pairs of inputs to an interconnection network which cause switch conflicts either in the first (input) stage or in the last (output) stage of the network. The regularity of BPC permutations provides possibility for finding crosstalk-free set of outputs for any crosstalk-free set of inputs. In its turn finding crosstalk-free set of inputs to a network is trivial. Here we consider two network topologies: (2n-1)-stage shuffle exchange network (SEN) and Benes network [11]. It is proved in [1] that any semi-permutation can be realized in a single pass in Benes network under the constraint of avoiding crosstalk. Benes network is known as a classic rearrangeable network, since (*2n*-1)-stage SEN is also always rearrangeable [12], we can here conjecture the same about it. SEN topology is considered here because it offers an identical interconnection pattern across all stages thus simplifying implementation especially in a photonic integration environment [13].

As it follows from the definition of a semi-permutation given in the Introduction, the basic requirements when decomposing a given BPC permutation into semi-permutations are the next:

- None of the source-destination pairs should share the same switch in the first stage of a network with another pair.
- None of the source-destination pairs should share the same switch in the output stage of a network with another pair.

For producing semi-permutations, two lists of antagonists are needed. The first one is of the inputs to a network sharing the inputs of the same switches in the first stage. In a SEN all connections between inputs to a network and inputs to the switches of the first stage are permanent and are implemented in accordance with perfect shuffle connection pattern :

$$Shuffle(a_{n-1}a_{n-2}...a_1a_0) = a_{n-2}...a_1a_0a_{n-1}$$ [14].

In other words the connection is based on cyclic shift to the left by one bit position. So the antagonists at inputs can be found using the following formula for any possible case:

$$N = 2^n, \quad Antagoinsts\ at\ inputs\colon\ i + \frac{N}{2}, \qquad 0 \le i \le \frac{N}{2}$$

As an example, for 16x16 SEN antagonistic pairs of inputs shown below:

0/8, 1/9, 2/10, 3/11, 4/12, 5/13, 6/14, 7/15.

The inputs shown together cannot be presented in the same semi-permutation. On the contrary for a Benes network there is no rearranging at inputs of the network, and antagonists at the inputs of 16x16 Benes network look as follows:

0/1, 2/3, 4/5, 6/7, 8/9, 10/11, 12/13, 14/15.

However, producing the list of input pairs – antagonists - causing switch conflicts in the output stage of the network is not so trivial, and it is based on periodicity of 0's and 1's in the rightmost column of the transition matrix of a given BPC permutation. That periodicity defines the periodicity of appearing identical $n-1$ combinations in the remainder part (without the rightmost bit) of the destinations half of the matrix. In other words it defines antagonistic rows in the binary version of a transition matrix. If resort to the transition matrix in its symbolic form, then with the rightmost component in the destination address being S_0, the period of changing 0's and 1's in the sequence of output numbers on the right side of a network diagram or in the rightmost column of a transition matrix is $N/2$, with S_1 that period is $N/4$, etc. with increasing degrees of 2 in denominators up to N ; so for S_{n-1} that period is 1. It is noteworthy that for all BPC permutations with the same rightmost component in their symbolic transition matrices the list of antagonists for outputs will be the same because permuting bits in the rest part of a destination address will not affect identity of the appropriate rows. Finally the formula for determining the afore said period R_i looks as follows: $R_i = 2^{(n-1)-I}$, $0 \le i \le n-1$, with *i* being the index of the rightmost component of the destination part in a symbolic transition matrix.

To summarize, decomposing a BPC permutation to semi-permutations includes the following steps:

1. Initialize N and n.
2. Initialize a transition matrix for a given BPC permutation in the symbolic form.

3. Produce the list of input pairs which are antagonistic at the network's first stage inputs basing on afore said reasoning.
4. Produce the list of input pairs causing switch conflicts in the output stage of a network (antagonists for the network's outputs) basing on the specific for a given permutation periodicity of 0's and 1's in the rightmost column of a binary version of a transition matrix in accordance with the formula for R_i above in the text.
5. Combine both lists of antagonists to one list as shown in the following examples, scan the lists of the inputs to a network with deleting antagonists to a current input. Each antagonist is encountered twice: either as an input or an output. Total number of pairs is $N/2$ in both cases, so exactly $N/2$ antagonists will be deleted (in other words, assigned to the second semi-permutation). Each antagonist is deleted once despite of encountering twice, so $N/2$ numbers of inputs are always left.

The above algorithm has time complexity *O (N).*

3 Examples of Decomposing

An example of applying our approach for decomposing Unshuffle permutation for realizing it in 8×8 shuffle-exchange and Benes networks is given below. Crosstalk-free routing in intermediate stages has been done by trial and error. In our case its transition matrix is $S_0S_1S_2S_2S_2S_0S_1$ and so the period *R=2*.

For *N=8* the original Unshuffle permutation looks as follows:
0→0, 1→4, 2→1, 3→5, 4→2, 5→6, 6→3, 7→7.

In accordance with afore said it can be easily decomposed into a pair of semi-permutations for 8×8 SEN:

0 →0, 1→4, 6→3, 7→7 : 1st S-P;
4→2, 5→6, 2→1, 3→5 : 2nd S-P.

Decomposing the same permutation for Benes network:

0 →0, 1→4, 6→3, 7→7: 1st S-P;
4→2, 5→6, 2→1, 3→5 : 2nd S-P.

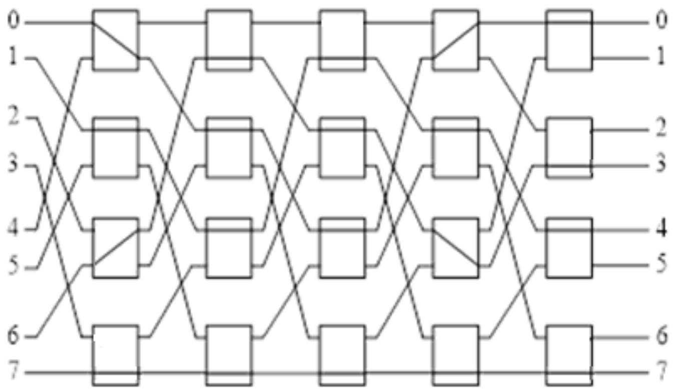

Figure 1. Routing the first semi-permutation of Unshuffle permutation on 8×8 5-stage shuffle-exchange network.

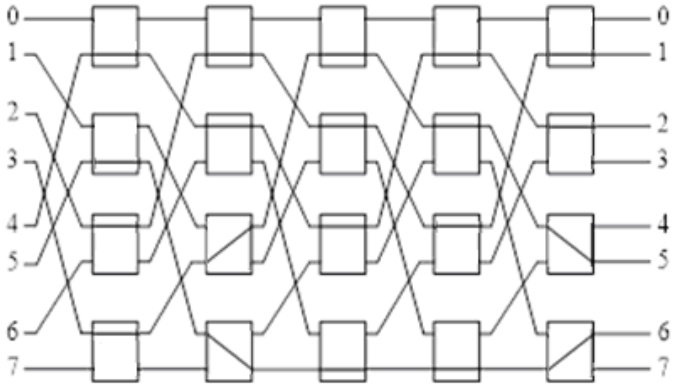

Figure 2. Routing the second semi-permutation of Unshuffle permutation on 8×8 5-stage shuffle-exchange network.

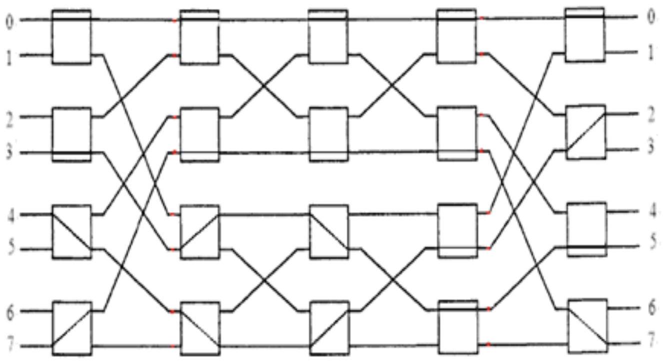

Figure 3. Routing the first semi-permutation of Unshuffle permutation on 8×8 5-stage Benes network.

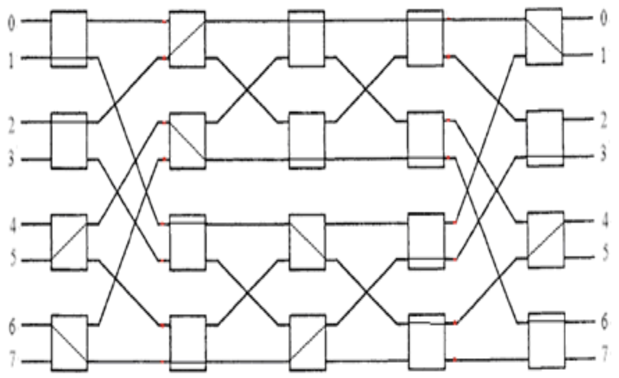

Figure 4. Routing the second permutation of Unshuffle permutation on 8×8 5-stage Benes network.

The second example is of larger size and concerns decomposing Butterfly permutation for 16×16 shuffle-exchange and Benes networks. In this case the transition matrix is $S_0S_1S_2S_3S_3S_1S_2S_0$, and the period *R=8.* The original permutation is:

0→0, 1→8, 2→2, 3→10, 4→4, 5→12, 6→6, 7→14, 8→1, 9→9, 10→3, 11→11, 12→5, 13→13, 14→7, 15→15.

More descriptive is its representing in the Table below.

Table 1
Butterfly permutation 16×16

Source Input $S_0S_1S_2S_3$	**Outputs** $S_3S_1S_2S_0$
0000 (0)	0000 (0)
0001 (1)	1000 (8)
0010 (2)	0010 (2)
0011 (3)	1010 (10)
0100 (4)	0100 (4)
0101 (5)	1100 (12)
0110 (6)	0110 (6)
0111 (7)	1110 (14)
1000 (8)	0001 (1)
1001 (9)	1001 (9)
1010 (10)	0011 (3)
1011 (11)	1011 (11)
1100 (12)	0101 (5)
1101 (13)	1101 (13)
1110 (14)	0111 (7)
1111 (15)	1111 (15)

It is easy to see from the Table 1 that periodicity of outputs which belongs to the same switch in the output stage, i.e. differ only in the rightmost bit, is really equals 8.

When decomposing afore said permutation for 16×16 SEN the steps are following.
Composing the list of antagonists at the input stage of the network (it is the same for any BPC permutation of a given size):

0/8, 1/9, 2/10, 3/11, 4/12, 5/13, 6/14, 7/15.

Composing the list of antagonists at the output stage is the next step. To be exact we find pair of inputs which evoke switch conflicts in the output stage for a given BPC permutation, e.g. simultaneous optical signals at inputs 0 and 8 would result in a switch conflict because corresponding outputs (0 and 1) belong to the same switch in the output stage of the network. All this can be easily seen from the Table 1. The rightmost component in the destination address is S_0, what means that we need to add 8 to the number of an input to find its antagonist from the viewpoint condition at the output stage:

0/8, 2/10, 4/12, 6/14, 1/9, 3/11, 5/13, 7/15.

The inputs shown together cannot be presented in the same semi-permutation; we delete the pair from the original permutation that uses the same switch either in an input or output stage.

We take a look at the first pair 0→0 from the original permutation.
From the lists of antagonist at inputs and outputs, 0 and 8 belong to the same switch.
Therefore, we delete 8→1 from the original permutation.

Next, we take a look at the pair 2→2.
From the lists of antagonist at inputs and outputs, 2 and 10 belong to the same switch.
Therefore, we delete 10→3 from the original permutation.

This is continued until we find the following semi-permutations:

0→0, 1→8, 2→2, 3→10, 4→4, 5→12, 6→6, 7→14 : 1st S-P
8→1, 9→9, 10→3, 11→11, 12→5, 13→13, 14→7, 15→15 : 2nd S-P

When decomposing the same permutation for a 16×16 Benes network, the list of antagonists for the first stage differs from that in the previous case and looks as follows:

0/1, 2/3, 4/5, 6/7, 8/9, 10/11, 12/13, 14/15.

However the list of antagonists concerning the output stage of the network is the same and produced in the same way by adding 8 to the number of an input:

0/8, 2/10, 4/12, 6/14, 1/9, 3/11, 5/13, 7/15.

The inputs shown together cannot be presented in the same semi-permutation; we delete the pair from the original permutation that belongs to the same switch in the same manner as in the previous case and as a result produce two following semi-permutations (the deleted pairs form another semi-permutation):

$0\to0, 2\to2, 4\to4, 6\to6, 9\to9, 11\to11, 13\to13, 15\to15$: 1st S-P
$1\to8, 3\to10, 5\to12, 7\to14, 8\to1, 10\to3, 12\to5, 14\to7$: 2nd S-P

4 Conclusion

In this paper an *O(N)* algorithm for decomposing BPC (bit-permute-complement) permutations into semi-permutations is introduced. The work contributes to solving the problem of crosstalk-free routing in hybrid optical multistage interconnection networks (OMINs). The algorithm is based on employment the property of periodicity 0's and 1's within columns of transition matrices for permutations belonging to the aforesaid class, it provides crosstalk-free conditions in the first and last stages of an OMIN. The algorithm is much simpler than known ones but for arbitrary permutations. The availability of semi-permutations creates the potential for crosstalk-free routing permutations for two passes through a network in an ideal case. Examples of decomposing some BPC permutations for (2n-1)-stage shuffle-exchange and Benes networks are given. The approach is applicable also to any type of banyan networks. The algorithm is implemented in C. A possible extension of the approach for providing crosstalk-free condition in intermediate stages of an OMIN is being studied.

5 References

[1] Y. Yang, J. Wang, Y. Pan, Permutation capability of optical multistage interconnection networks, *Journal of Parallel and Distributed Computing, 60,* 2000, 72-91.

[2] Y. Pan, C. Qiao, Y. Yang, Optical multistage interconnection networks: new challenges and approaches, *IEEE Communication , 2*, 1999, 50-56.

[3] S.Kaur, R. Vohra, S. Kaur, Analysis of various crosstalk avoidance techniques in optical multistage interconnection network, *International Journal of P2P Network Trends and Technology, 1(2)*, 2011, 26-30.

[4] X. Lin, Y. Zhao, and Y. Wu, An efficient crosstalk-free routing algorithm based on permutation decomposition for optical multi-$\log_2 N$ switching networks, *Proc. 10th IFIP International Conference, Guiyana, China, LNCS 8147,* 2013, 207-219.

[5] R. Bashirov and T. Karanfiller, On path dependent loss and switch crosstalk reduction in optical networks, *Information Sciences, (180)*, 2005, 1040-1050.

[6] S. Mishra, N. Chaudhary, K. Singh, Overview of optical interconnect technology, *International Journal of Scientific & Engineering Research*, 3(4), 2012, 1-7.

[7] X. Shen, Optimal realization of any BPC permutation on *k*-extra-stage Omega networks, *IEEE Trans. Computers, 44(5)*, 1995, 714-719.

[8] G. Veselovsky and A. Agrawal, On crosstalk-free BPC permutations routing in an optical variable-stage shuffle-exchange networks, *Proc. IASTED Intl. Conf. PDCN 2014, Innsbruck, Austria,* 2014, 232-238.

[9] J. Lenfant, A versatile mechanism to move data in an array processor, *IEEE Trans. Computers, C-34 (6)*, 1985, 506-522.

[10] M. D. Grammatikakis, D. Frank Hsu, M. Kraetzl, *Parrallel System Interconnections and Communication*, CRC Press, 2000.

[11] H. J. Siegel, *Interconnection networks for large-scale parallel processing. Theory and case studies*, 2nd ed. McGraw-Hill International Editions, 1990.

[12] A. Abdennadher and T. Y. Feng, On rearrangeability of Omega-Omega networks, *Proc. IEEE Int. Conf. Parallel Proc., I,* 1992, 159-165.

[13] A. Shacham, *Architectures of optical interconnection networks for high performance computing*, VDM Verlag Dr. Muller, 2007.

[14] H. Stone, Parallel processing with the perfect shuffle, *IEEE Trans. Computers, 20(2*), 1971, 153-161.