2009 Eighth International Conference on Networks

All-to-All Personalized Exchange Algorithms in Generalized Shuffle-exchange Networks

Well Y. Chou, Richard B. Chen, and Chiuyuan Chen
Department of Applied Mathematics
National Chiao Tung University
Hsinchu 300, Taiwan
Email of corresponding author: cychen@mail.nctu.edu.tw

Abstract

All-to-all personalized exchange (ATAPE) occurs in many parallel applications. Previous ATAPE algorithms were mainly developed for hypercube, mesh, and torus networks. Recently, Yang and Wang [8] and also Massini [4] proposed an alternative approach to ATAPE by using multistage interconnection networks (MINs); they proposed new ATAPE algorithms for a class of unique-path, self-routable MINs (for example, baseline, shuffle-exchange (or omega), banyan network, and the reverse networks of these networks). However, the algorithms in [4] and [8] require that the given MIN must have unique-path property and satisfy $N = 2^n$, in which N is the number of inputs (outputs) and n is the number of stages in the MIN. In [5], Padmanabhan proposed the generalized shuffle-exchange network (GSEN), which allows N to be any even number. Since the GSEN is not a unique-path MIN, the algorithms in [4] and [8] do not work on it. The purpose of this paper is to consider ATAPE in MINs without unique-path property. To our knowledge, no one has studied ATAPE in this type of MINs. We prove that under stage control technique, ATAPE algorithms for GSENs require at least 2^n rounds. We propose an algorithm which uses a variation of stage control and works for all $N \equiv 2 \pmod 4$. We will prove that our algorithm takes N rounds and therefore is optimal.

1. Introduction

Processors in parallel/distributed computing system often need to communicate with other processors. The communication among these processors could be *one-to-one*, *one-to-many*, or *all-to-all*. All-to-all communication can be further classified into *all-to-all broadcast* (ATABR) and *all-to-all personalized exchange* (ATAPE). In ATABR, each processor sends the same message to all other processors; while in ATAPE, each processor sends a specific message to every other processor. For convenience, in the remaining part of this paper, ATA means all-to-all communication and it can be either ATABR or ATAPE.

ATAPE occurs in many important applications (for example, matrix transposition and fast Fourier transform) in parallel and distributed computing. The ATAPE problem has been extensively studied for hypercubes, meshes, and tori. As was mentioned in [8], although the algorithm for a hypercube achieves optimal time complexity, a hypercube suffers from unbounded node degrees and therefore has poor scalability; on the other hand, although a mesh or torus has a constant node degree and better scalability, its algorithm has a higher time complexity. In [8], Yang and Wang proposed an alternative approach for ATAPE by using MINs and showed that an MIN is a better choice due to its shorter communication delay and better scalability.

Given N processors $P_0, P_1, \ldots, P_{N-1}$, an $N \times N$ MIN can be used for communication among these processors as shown in Fig. 1 and Fig. 2, where $N \times N$ means N inputs and N outputs. A column in an MIN is called a *stage* and the nodes in an MIN are called *switches*. Throughout this paper, N denotes the number of processors and n denotes the number of stages. Also, all the switches are assumed to be of size 2×2 since switches of size 2×2 are the most commonly used ones; see also [1], [3]. It is well known that a 2×2 switch has only two possible states: *straight* and *cross*, as shown in Fig. 3.

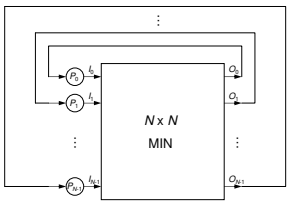

Figure 1. Communications among processors using an MIN.

An MIN is *unique-path* if there is only one path between each pair of input and output. An MIN is *self-routable* if the routing decision at a switch depends only on the addresses of the source processor and the destination processor. In [8], Yang and Wang proposed an ATAPE algorithm for a class of unique-path, self-routable MINs; for example, baseline, omega, banyan networks, and the reverse networks of these

978-0-7695-3552-4/09 $25.00 © 2009 IEEE
DOI 10.1109/ICN.2009.58

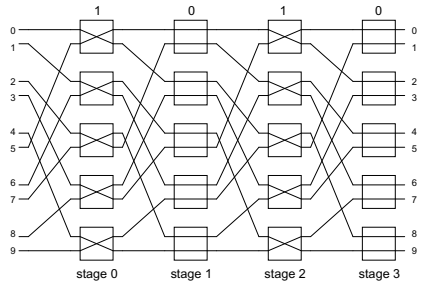

Figure 2. A 10×10 GSEN with network configuration $(1010)_2$.

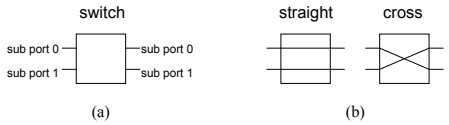

Figure 3. (a) A 2×2 switch and its sub ports. (b) Two states.

networks. Yang and Wang's algorithm [8] uses stage control, which is a commonly used technique to reduce the cost of the network setting for ATA. *Stage control* means that the states of all the switches of a stage must be identical. With stage control a single control bit (0 for straight and 1 for cross), or in other words, a single electronic driver circuit, can be used to control all the switches of a stage so that the network const can be reduced.

Recently, Massini [4] observed that Yang and Wang's algorithm depends on the network topologies and requires pre-computation and memory allocation for a Latin square. Massini proposed a new ATAPE algorithm, which is independent of the network topologies and does not require pre-computation or memory allocation for a Latin square.

The shuffle-exchange network has been proposed as a popular architecture for MINs. ATAPE of this network can be realized using algorithms in [4] or [8]. The generalized shuffle-exchange network (GSEN) is a generalization of shuffle-exchange network and was proposed by Padmanabhan in [5]. More precisely, let N be an even number, $n = \lceil \log_2 N \rceil$ and $2^{n-1} < N \leq 2^n$. An $N \times N$ GSEN is an $N \times N$ MIN with exactly n stages such that each stage consists of the perfect shuffle on N terminals followed by $N/2$ switches. The N terminals in an $N \times N$ GSEN are numbered $0, 1, \ldots, N-1$ and the *perfect shuffle operation* on the N terminals is the permutation π defined by $\pi(i) = (2i + \lfloor \frac{2i}{N} \rfloor) \bmod N$, $0 \leq i < N$. See Fig. 2.

Do notice that the algorithms in [4] and [8] require the given MIN must have unique-path property and satisfy $N = 2^n$. A GSEN may not satisfy $N = 2^n$ and may not have unique-path property. Consequently, the ATAPE algorithms in [4] and [8] do not work on a GSEN. The purpose of this paper is to consider ATAPE in MINs without unique-path property. We will prove that under stage control technique, ATAPE algorithms for GSENs need at least 2^n rounds. We will propose an algorithm using a variation of stage control and works for all $N \equiv 2 \pmod 4$. We will prove that our algorithm takes only N rounds and therefore is optimal.

This paper is organized as follows: Section 2 gives preliminaries. Section 3 is a lower bound for the ATAPE problem when the stage control technique is assumed. Section 4 is our ATAPE algorithm for GSENs with $N \equiv 2 \pmod 4$. Concluding remarks are given in the final section.

2. Preliminaries

In the paper, a MIN means an $N \times N$ MIN and a GSEN means an $N \times N$ GSEN. In a GSEN, the switches are aligned in n stages: stage 0, stage 1, ..., stage $n-1$, with each stage consists of $N/2$ switches. The *network configuration* of a MIN is defined by the states of its switches. Since a GSEN has $(N/2) \times n$ switches, its network configuration can be represented by an $(N/2) \times n$ matrix in which each entry is defined by the state of its corresponding switch.

When stage control technique is assumed, the network configuration of a GSEN can be represented by a number as follows. Let $c_\ell = 0$ ($c_\ell = 1$) if the state of all switches at stage $n-1-\ell$ is 0 (1). The network configuration can be represented by the number $C = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + \cdots + c_1 2^1 + c_0 2^0$ or by binary number $(c_{n-1}\ c_{n-2}\ \ldots\ c_1\ c_0)_2$. See Fig. 2 for an example. Clearly, $0 \leq C < 2^n$.

A *permutation* of a MIN is one-to-one mapping between inputs and outputs of the MIN. If there is a permutation that maps input i to output $p(i)$, where $p(i) \in \{0, 1, \ldots, N-1\}$ for $i = 0, 1, \ldots, N-1$, then we simply use $p(0)\ p(1)\ \ldots\ p(N{-}1)$ to denote the permutation. Given the network configuration of a MIN, a permutation can be obtained. For example, the network configuration shown in Fig. 2 obtains permutation 1 4 6 0 7 2 9 3 5 8. It is obvious that there are $N!$ possible permutations. However, not all of the $N!$ permutations are realizable. Permutations realizable by a MIN are called *admissible permutations* of that MIN.

In the paper, terminal i (j) is assumed on the left-hand (right-hand) side of the network and therefore is an input (output) processor. Consider a message sending from i to j; see Fig. 4. The path can be described by a sequence of labels that label the successive links on this path. Such a sequence is called a *forward control tag* or *tag*. The control tag may be used as a header for routing a message.

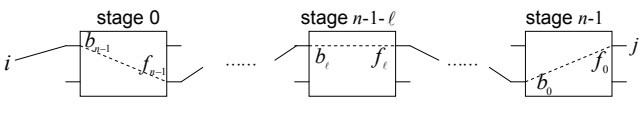

Figure 4. The sub ports on the path P between i and j.

When a message is sent from i to j along a path P, the message enters a switch at stage $n-1-\ell$ via sub port b_ℓ and leaves the switch via sub port f_ℓ. Conversely, when a

message is sent from j to i along P, the message enters a switch at stage $n-1-\ell$ via sub port f_ℓ and leaves the switch via sub port b_ℓ. We use $F = f_{n-1}2^{n-1}+f_{n-2}2^{n-2}+\cdots+f_1 2^1+f_0 2^0$ to denote a forward control tag. Let $B = b_{n-1}2^{n-1}+b_{n-2}2^{n-2}+\cdots+b_1 2^1+b_0 2^0$. B is called the *backward control tag*. For example, in Fig. 2, $i = 2$ can get to $j = 6$ by using $F = 14 = (1110)_2$; conversely, $j = 6$ can get to $i = 2$ by using the backward control tag $B = 4 = (0100)_2$. Note that $0 \le F < 2^n$ and $0 \le B < 2^n$.

Let $P(i, F)$ denote the path started from i by using the forward control tag F; note that the destination processor j of $P(i, F)$ can be determined by Lemma 3 in the next section. Let $B(i, F)$ denote the backward control tag of $P(i, F)$ and let $\mathcal{B}_F = \{B(i, F) \mid i = 0, 1, \ldots, N-1\}$.

In this paper, $\oplus$ denotes the bitwise XOR operation. As a reference, $0\oplus 0 = 0$, $0\oplus 1 = 1$, $1\oplus 0 = 1$, $1\oplus 1 = 0$. If $U = (u_{n-1}\ u_{n-2}\ \ldots\ u_0)_2$ and $V = (v_{n-1}\ v_{n-2}\ \ldots\ v_0)_2$, then we define $U\oplus V = (u_{n-1}\oplus v_{n-1}\ \ u_{n-2}\oplus v_{n-2}\ \ \ldots\ \ u_0\oplus v_0)_2$.

Let $\mathcal{R}(N)$ denote the minimum number of network configurations required to realize ATA in an $N \times N$ GSEN. Also, let $\mathcal{R}_{sc}(N)$ denote the minimum number of network configurations required to realize ATA in an $N \times N$ GSEN when stage control technique is assumed. A *round* is the process of transmitting all the messages from the input stage to the output stage.

3. A lower bound when stage control technique is assumed

The main result of this section is to prove that $\mathcal{R}_{sc}(N)$ has a lower bound 2^n. We first prove a theorem.

Theorem 1: In a GSEN,

$$N \le \mathcal{R}(N) \le \mathcal{R}_{sc}(N) \le 2^n.$$

Proof: Given a network configuration, at most N messages can be sent simultaneously. $N \le \mathcal{R}(N)$ thus follows from that fact that N^2 messages have to be sent to fulfill ATA and each network configuration can send only N of them. $\mathcal{R}(N) \le \mathcal{R}_{sc}(N)$ is obvious. $\mathcal{R}_{sc}(N) \le 2^n$ follows from the fact that a GSEN has at most 2^n network configurations when stage control technique is assumed. ∎

In [2], Lan et al. considered a GSEN with switches of size $k \times k$. By setting $k = 2$, we have the following results.

Theorem 2: [2] In a GSEN, the four parameter i, j, F, B of a path satisfy $2^n i + F = BN + j$.

Lemma 3: [2] Given i and F in a GSEN, the destination processor j is determined by $j = (i \cdot 2^n + F) \bmod N$.

Lemma 4: [2] In a GSEN, the backward control tag B of the path $P(i, F)$ is given by $B = \left\lfloor \frac{i\cdot 2^n + F}{N} \right\rfloor$.

See Fig. 2 for an illustration. Suppose $i = 2$ sends a message by using the forward control tag $F = 14$. By Lemma 3, the destination is $j = (2 \cdot 16 + 14) \bmod 10 = 6$; by Lemma 4, the corresponding B is $B = \left\lfloor \frac{2\cdot 16+14}{10} \right\rfloor = 4$.

The reason of introducing the forward control tag F and the backward control tag B of a given path is to carry out the configuration C containing that path by bitwise XOR operation.

Lemma 5: When the stage control technique is assumed, F and B together uniquely determine the network configuration C and $C = B \oplus F$.

Proof: Consider stage $n-1-\ell$. Since the stage control technique is assumed, all switches in stage $n-1-\ell$ are of the same state. Let $C = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + \cdots + c_1 2^1 + c_0 2^0$ be the network configuration and see Fig. 4. At stage $n-1-\ell$, a message enters sub port b_ℓ and leaves sub port f_ℓ. If $b_\ell = f_\ell$, then the state of the switch is straight; hence $c_\ell = 0 = b_\ell \oplus f_\ell$. If b_ℓ differs from f_ℓ (in this case, (b_ℓ, f_ℓ) is $(0, 1)$ or $(1, 0)$), then the state of the switch is cross; hence $c_\ell = 1 = b_\ell \oplus f_\ell$. So $C = B \oplus F$. ∎

Before bring out the main result of this section, we need a little more information about the unique-path: how can we tell if a path is unique or not from F directly?

Lemma 6: In a GSEN, a path $P(i, F)$ is a unique-path if and only if $2^n - N \le F < N$. (See Fig. 5 for illustration.)

Proof: Suppose there are two different paths $P(i, F_1)$, $P(i, F_2)$ joining i to j. By Lemma 3, the difference between F_1 and F_2 is N. Without loss of generality, let $F_2 - F_1 = N$. Since $F_1 \ge 0$, we have $F_2 \ge N$. Since $F_2 < 2^n$, $F_1 < 2^n - N$. When $2^n - N \le F < N$, F is neither F_1 nor F_2; thus $P(i, F)$ must be a unique-path. ∎

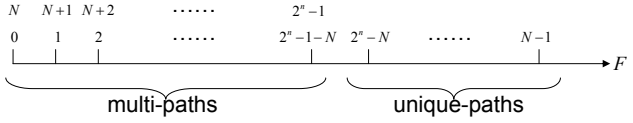

Figure 5. Unique paths and multiple paths joining i to j.

Lemma 7: $P(i, 2^{n-1})$ and $P(i, 2^{n-1}+1)$ are unique paths.

Proof: Follows from Lemma 6. ∎

Lemma 8: In a GSEN, $\mathcal{B}_{2^{n-1}} = \mathcal{B}_{2^{n-1}+1}$.

Proof: The binary representations of 2^{n-1} and $2^{n-1}+1$ differ only at their rightmost bits. Thus for $i = 0, 1, \ldots, N-1$, paths $P(i, 2^{n-1})$ and $P(i, 2^{n-1}+1)$ differ only at their destination processors; so $B(i, 2^{n-1}) = B(i, 2^{n-1}+1)$. Consequently, $\mathcal{B}_{2^{n-1}} = \mathcal{B}_{2^{n-1}+1}$. ∎

For convenience, if a number is in $\{0, 1, 2, \ldots, 2^n - 1\}$ but is not in $\mathcal{B}_F$, then we call it a *hole* of $\mathcal{B}_F$. The following lemma shows that the elements of $\mathcal{B}_F$ are spread very *evenly* on the set $\{0, 1, 2, \ldots, 2^n - 1\}$.

Lemma 9: For any $F \in \{0, 1, 2, \ldots, 2^n - 1\}$, $\mathcal{B}_F$ has no two consecutive holes.

Proof: We prove this lemma by showing $B(0, F) \le 1$, $B(i-1, F)+1 \le B(i, F) \le B(i-1, F)+2$ for $i = 1, 2, \ldots, N-1$, and $B(N-1, F) \ge 2^n - 2$. By Lemma 4, $B(0, F) = \left\lfloor \frac{F}{N} \right\rfloor \le 1$. Also, $B(N-1, F) = \left\lfloor \frac{(N-1)\cdot 2^n + F}{N} \right\rfloor \ge \left\lfloor \frac{(N-1)\cdot 2^n}{N} \right\rfloor \ge 2^n - 2$. Finally, consider $i = 1, 2, \ldots, N-1$. By Lemma 4,

$B(i-1,F)+1 = \left\lfloor \frac{(i-1)\cdot 2^n+F}{N} \right\rfloor + 1 = \left\lfloor \frac{i\cdot 2^n+F}{N} - \frac{2^n}{N} \right\rfloor + 1 \leq \left\lfloor \frac{i\cdot 2^n+F}{N} \right\rfloor = B(i,F) = \left\lfloor \frac{(i-1)\cdot 2^n+F}{N} + \frac{2^n}{N} \right\rfloor \leq \left\lfloor \frac{(i-1)\cdot 2^n+F}{N} \right\rfloor + 2 = B(i-1,F)+2.$ ∎

Now we are ready to give the main result of this section.

Theorem 10:

$$\mathcal{R}_{sc}(N) = 2^n.$$

Proof: It suffices to prove that when stage control technique is assumed, each of 2^n network configurations is required for every processor to receive N messages. By Theorem 1, $\mathcal{R}_{sc}(N) \leq 2^n$. It remains to prove $\mathcal{R}_{sc}(N) \geq 2^n$.

When stage control technique is assumed, the network configuration C can be determined by an arbitrary path P set up by C. In particular, if F is the control tag used by P, and B is the the backward control tag of P (see Fig. 4), then by Lemma 5, $C = B \oplus F$. If P is a unique path, then C must be used in ATA. Recall that $0 \leq C < 2^n$. Our idea used in proving $\mathcal{R}_{sc}(N) \geq 2^n$ is to prove that for each C in $\{0, 1, \ldots, 2^n - 1\}$, at least one of the paths set up by C is a unique path and hence C must be used in ATA.

Suppose to the contrary there is a $\hat{C}$ in $\{0, 1, \ldots, 2^n - 1\}$ such that none of the paths set up by $\hat{C}$ is a unique path. Then consider $2^{n-1} \oplus \hat{C}$ and let $\hat{B} = 2^{n-1} \oplus \hat{C}$; consider $(2^{n-1}+1) \oplus \hat{C}$ and let $\hat{B}' = (2^{n-1}+1) \oplus \hat{C}$. Since none of the paths set up by $\hat{C}$ is a unique path, we claim that $\hat{B} \notin \mathcal{B}_{2^{n-1}}$ and $\hat{B}' \notin \mathcal{B}_{2^{n-1}+1}$. Suppose this clam is not true. Then $\hat{B} \in \mathcal{B}_{2^{n-1}}$ or $\hat{B}' \in \mathcal{B}_{2^{n-1}+1}$. Without loss of generality, suppose $\hat{B} \in \mathcal{B}_{2^{n-1}}$ holds. Then $\hat{C} = \hat{B} \oplus 2^{n-1}$. By Lemma 7, $\hat{C}$ conducts a unique path, a contradiction.

By Lemma 8, $\mathcal{B}_{2^{n-1}} = \mathcal{B}_{2^{n-1}+1}$. Thus $\hat{B} \notin \mathcal{B}_{2^{n-1}}$ and $\hat{B}' \notin \mathcal{B}_{2^{n-1}}$. Since $\hat{B}$ and $\hat{B}'$ differ by 1, they are two consecutive holes in $\mathcal{B}_{2^{n-1}}$; this contradicts with Lemma 9. Thus for each network configuration C in $\{0, 1, \ldots, 2^n - 1\}$, at least one of the paths set up by C is a unique path; hence C must be used in ATA. So $\mathcal{R}_{sc}(N) \geq 2^n$. ∎

4. ATAPE of GSENs with $N \equiv 2 \pmod 4$

We now introduce a variation of stage control technique, called *alternating stage control* (ASC), meaning that the states of the switches of a stage *alternate between straight and cross*. See Fig. 6 for an illustration.

When alternating stage control is used, the network configuration of a GSEN can be represented by a number as follows. Let a_ℓ denotes the states of the switches at stage $n-1-\ell$ such that

- $a_\ell = 0$ means the states are $0, 1, 0, \ldots$.
- $a_\ell = 1$ means the states are $1, 0, 1, \ldots$.

The network configuration of the GSEN can be represented by the number $A = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots + a_1 2^1 + a_0 2^0$ or $(a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_2$ in the binary form; see Fig. 6. Clearly, $0 \leq A < 2^n$. We will call A an *alternating configuration*. When $N \equiv 2 \pmod 4$ and alternating stage control is used, the N input terminals and N output terminals of stage $n-1-\ell$ have the following property.

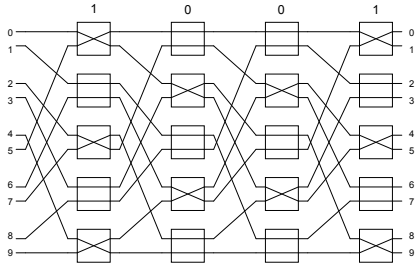

Figure 6. Applying alternating stage control $A = 9 = (1001)_2$ on a 10×10 GSEN.

Property ($*$): (see Fig. 7 for an illustration)

1) If $a_\ell = 0$, then $Even \xrightarrow{0} Even$, $Odd \xrightarrow{1} Odd$. That is, every even-numbered input terminal is connected to an even-numbered output terminal via sub port 0, and every odd-numbered input terminal is connected to an odd-numbered output terminal via sub port 1.
2) If $a_\ell = 1$, then $Even \xrightarrow{1} Odd$, $Odd \xrightarrow{0} Even$.

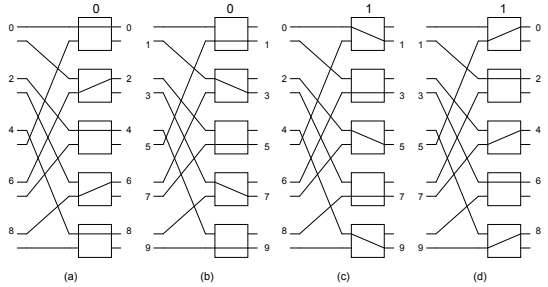

Figure 7. A stage in a 10×10 GSEN. (a) and (b) are for $a_\ell = 0$; (c) and (d) are for $a_\ell = 1$.

Notice that if $N \not\equiv 2 \pmod 4$, Property $(*)$ does not hold. We now give other properties of alternating stage control.

Lemma 11: Suppose $N \equiv 2 \pmod 4$, alternating stage control is used, and $A = (a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_2$ is the network configuration. Then: (1) the forward control tags of even-numbered inputs are identical, and (2) the forward control tags of odd-numbered inputs are identical.

Proof: By Property $(*)$, messages from even-numbered inputs are via the same sub port at every stage $n-1-\ell$, $(\ell = n-1, n-2, \ldots, 0)$. Since the control tag is the sub ports passed by a message, (1) holds. Similarly, (2) also holds. ∎

Theorem 12: Suppose $N \equiv 2 \pmod 4$, alternating stage control is used, and A is the network configuration. Let F and $\overline{F}$ denote the forward control tags of even-numbered inputs and odd-numbered inputs, respectively. Then:

(1) $F \oplus \overline{F} = (11\ldots11)_2$;
(2) $A = F \oplus \left\lfloor \frac{F}{2} \right\rfloor$;
(3) $F = A \oplus \left\lfloor \frac{A}{2} \right\rfloor \oplus \left\lfloor \frac{A}{2^2} \right\rfloor \oplus \cdots \oplus \left\lfloor \frac{A}{2^{n-1}} \right\rfloor$.

Proof: (1) Let $F = (f_{n-1}\ f_{n-2}\ \ldots\ f_1\ f_0)_2$ and $A = (a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_2$. By Property $(*)$, if messages from even-numbered inputs are via sub port f_ℓ at stage $n-1-\ell$, then messages from odd-numbered inputs are via sub port $1-f_\ell$ at stage $n-1-\ell$, $(\ell = n-1, n-2, \ldots, 0)$. Thus $F \oplus \overline{F} = (11\ldots11)_2$.

(2) Clearly, $a_{n-1} = f_{n-1}$. For $\ell = n-2, n-3, \ldots, 0$, by Property $(*)$, we have:

- If $a_\ell = 0$, then $f_\ell = 0$ whenever $f_{\ell+1} = 0$ and $f_\ell = 1$ whenever $f_{\ell+1} = 1$.
- If $a_\ell = 1$, then $f_\ell = 0$ whenever $f_{\ell+1} = 1$ and $f_\ell = 1$ whenever $f_{\ell+1} = 0$.

Thus, $a_\ell = f_\ell \oplus f_{\ell+1}$, $(\ell = n-2, n-3, \ldots, 0)$. Therefore,

$$\begin{aligned} A &= (a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_2 \\ &= (f_{n-1}\ \ f_{n-2}\oplus f_{n-1}\ \ f_{n-3}\oplus f_{n-2}\ \ \ldots\ \ f_0\oplus f_1)_2 \\ &= (f_{n-1}\oplus 0\ \ f_{n-2}\oplus f_{n-1}\ \ f_{n-3}\oplus f_{n-2}\ \ \ldots\ \ f_0\oplus f_1)_2 \\ &= (f_{n-1}\ f_{n-2}\ f_{n-3}\ \ldots\ f_0)_2 \oplus (0\ f_{n-1}\ f_{n-2}\ \ldots\ f_1)_2 \\ &= F \oplus \left\lfloor \tfrac{F}{2} \right\rfloor. \end{aligned}$$

(3) $f_\ell = a_\ell \oplus a_{\ell+1} \oplus \cdots \oplus a_{n-1},\ (\ell = n-2, n-3, \ldots, 0)$. Thus, $F = A \oplus \left\lfloor \frac{A}{2} \right\rfloor \oplus \left\lfloor \frac{A}{2^2} \right\rfloor \oplus \cdots \oplus \left\lfloor \frac{A}{2^{n-1}} \right\rfloor$. ∎

The above theorem gives a one-to-one correspondence between A and F. Let A_F be the corresponding alternating configuration of forward control tag F. By (2) of Theorem 12, $A_k = k \oplus \left\lfloor \frac{k}{2} \right\rfloor$.

Lemma 13: When $N \equiv 2 \pmod 4$ and the GSEN is set by the alternating configuration A_k, the forward control tag of even-numbered inputs is k and the forward control tag of the odd-numbered inputs is $2^n - 1 - k$.

Proof: By definition of A_k and (1) of Theorem 12. ∎

Let $\mathcal{A}$ denote a set of alternating configurations which can be used to fulfill ATA. We now prove a theorem, which is the foundation of our optimal ATA algorithm.

Theorem 14: When $N \equiv 2 \pmod 4$, the set of alternating configurations $\mathcal{A} = \{A_0, A_1, \ldots, A_{N-1}\}$ can fulfill ATA, where $A_k = k \oplus \left\lfloor \frac{k}{2} \right\rfloor$.

Proof: Let i be an arbitrary input. To prove this theorem, it suffices to prove that when $A_0, A_1, \ldots, A_{N-1}$ are used, i can get to every output. Let j_k be the destination processor when the network configuration is A_k. First consider the case that i is even. By Lemmas 3 and 13, $j = (i \cdot 2^n + k) \mod N$; since k varies from 0 to $N-1$, i can get to every output. Now consider the case that i is odd. By Lemmas 3 and 13, $j = (i \cdot 2^n + 2^n - 1 - k) \mod N$; again, since k varies from 0 to $N-1$, i can get to every output. ∎

As an illustration, for a 10×10 GSEN, $\mathcal{A} = \{0, 1, 3, 2, 6, 7, 5, 4, 12, 13\}$ can fulfill ATA. We now obtain $\mathcal{R}(N)$ for $N \equiv 2 \pmod 4$.

Theorem 15:

$$\text{For GSENs with } N \equiv 2 \pmod 4,\ \mathcal{R}(N) = N.$$

Proof: Follows from Theorems 1 and 14. ∎

Note that $A_0, A_1, \ldots, A_{N-1}$ are not the only way to fulfill ATA. In fact, any consecutive N integers in $0, 1, \ldots, N-1$ can fulfill ATA. Now we are ready to propose our ATA algorithms for GSENs with $N \equiv 2 \pmod 4$. The first algorithm (Algorithm 1) generates a set of configurations that can fulfill ATA. The second algorithm (Algorithm 2) uses the output of the first algorithm to fulfill ATABR.

Algorithm 1 Generate-Configurations-using-ASC

Input: A number N such that $N \equiv 2 \pmod 4$.
Output: A set $\mathcal{A}$ of alternating configurations that can fulfill ATA in an $N \times N$ GSEN.

```
A ← ∅
for k = 0 to N − 1 do
    A_k ← k ⊕ ⌊k/2⌋
    A ← A ∪ {A_k}
end for
return A
```

Algorithm 2 ATABR-using-ASC

Input: The set $\mathcal{A}$ of alternating configurations.
Output: Fulfilling ATABR in a GSEN.

```
Each processor i (0 ≤ i < N) prepare a broadcast message
for each round k = 0 to N − 1 do
    Set the configuration of the GSEN to be A_k
    Transmit messages
end for
```

In ATAPE, messages sent to different destinations are different. By constructing a matrix $D = (d_{i,k})$ where $d_{i,k} = j$ means processor j (the destination) will receive a personalized message from processor i at round k, we can fulfill ATAPE. Algorithm 3 prepares such a matrix.

Algorithm 3 Generate-Destination-Matrix-using-ASC

Input: A number N such that $N \equiv 2 \pmod 4$.
Output: Constructing the destination matrix $D = (d_{i,k})$.

```
n ← ⌈log_2 N⌉
for each i = 0 to N − 1 do
    if i is even then
        m_i = (i · 2^n) mod N
    else
        m_i = ((i + 1)2^n − 1) mod N
    end if
    for each k = 0 to N − 1 do
        if i is even then
            d_{i,k} = (m_i + k) mod N
        else
            d_{i,k} = (m_i − k) mod N
        end if
    end for
end for
```

For example, the matrix D of a 10×10 GSEN is below.

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 0 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 \\ 3 & 2 & 1 & 0 & 9 & 8 & 7 & 6 & 5 & 4 \\ 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 \\ 5 & 4 & 3 & 2 & 1 & 0 & 9 & 8 & 7 & 6 \\ 6 & 7 & 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 9 & 8 \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Note that D needs to be constructed only once. It can be pre-computed and can be used again and again. The last algorithm uses the outputs (set $\mathcal{A}$ and matrix D) of Algorithms 1 and 3 to fulfill ATAPE.

Algorithm 4 ATAPE-using-ASC

Input: The set $\mathcal{A}$ of alternating configurations and the destination matrix D.

Output: Fulfilling ATAPE in a GSEN.

1: **for** each round k = 0 to $N-1$ **do**
2: Each processor i $(0 \leq i < N)$ prepare a personalized message for $d_{i,k}$
3: Set the configuration of the GSEN to be A_k
4: Transmit messages
5: **end for**

An example of Algorithm Algorithm 4 is shown in Fig. 8. By Theorem14, Algorithms 1 and 2 are correct; each of them takes N rounds and takes $O(N)$ time. Algorithm 3 is correct if we can show that at round k, the message sent by processor i will reach processor $(m_i + k) \bmod N$ if i is even and reach $(m_i - k) \bmod N$ if i is odd. We only prove the case that i is odd. By Lemma 13, at round k, the messages sent by processor i uses forward control tag $2^n - 1 - k$. By Lemma 3, the destination processor is be $j = (i \cdot 2^n + 2^n - 1 - k) \mod N = ((i+1) \cdot 2^n - 1 - k) \mod N = (m_i - k) \mod N$. Thus Algorithm 3 is correct. It is not difficult to see that the algorithm takes $O(N^2)$ time. The correctness of Algorithm 4 follows from that of Algorithm 3; the algorithm takes N rounds and takes $O(N)$ time.

5. Concluding remarks

In this paper, we consider the generalized shuffle-exchange network (GSEN), which is not necessarily a unique-path MIN. We have proposed an optimal ATAPE algorithm for GSENs. Unlike the algorithms in [4] and [8], we abandon the requirement on the unique-path property. Our algorithm use alternating stage control and works for all $N \equiv 2 \pmod 4$. By Theorems 10 and 15, for GSENs with $N \equiv 2 \pmod 4$, we have $N = \mathcal{R}(N) < \mathcal{R}(N) = 2^n$. It remains open to determine $\mathcal{R}(N)$ for $N \equiv 0 \pmod 4$.

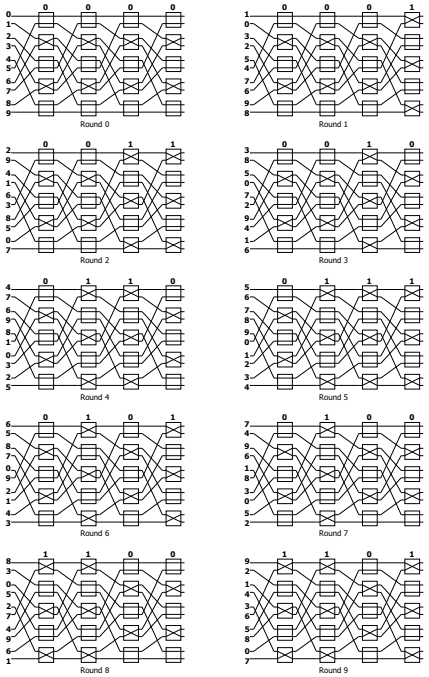

Figure 8. An example of Algorithm 4.

References

[1] C. Chen and J. K. Lou, "An efficient tag-based routing algorithm for the backward network of a bidirectional general shuffle-exchange network," *IEEE Commun. Lett.*, vol. 10, no. 4, pp. 296-298, 2006.

[2] J. K. Lan, W. Y. Chou, and C. Chen, "Efficient routing algorithms for the bidirectional general shuffle-exchange network," submitted for possible publication.

[3] V. W. Liu, C. Chen, and R. B. Chen, "Optimal all-to-all personalized exchange in d-nary banyan multistage interconnection networks," *J. Comb. Optim.*, vol. 14, pp. 131-142, 2007.

[4] A. Massini, "All-to-all personalized communication on multistage interconnection networks," *Discrete Appl. Math.*, vol. 128, no. 2, pp. 435-446, 2003.

[5] K. Padmanabham, "Design and analysis of even-sized binary shuffle-exchange networks for multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 2, no. 4, pp. 385-397, Oct. 1991.

[6] Y. Yang, J. Wang, "All-to-all personalized exchange in banyan networks," *Proc. Parallel and Distributed Computing and Sysetems (PDCS'99)*, Cambridge, MA, pp. 78-86, 1999.

[7] Y. Yang, J. Wang, "Optimal all-to-all personalized exchange in multistage networks," *Proc. Seventh International Conference on Parallel and Distributed Systems (ICPADS'00)*, Iwale, Japan, 2000.

[8] Y. Yang, J. Wang, "Optimal all-to-all personalized exchange in self-routable multistage networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 3, pp. 261-274, 2000.