Parallel Routing Algorithms for Nonblocking Electronic and Photonic Switching Networks

Enyue Lu, *Member*, *IEEE*, and S.Q. Zheng, *Senior Member*, *IEEE*

Abstract—We study the connection capacity of a class of rearrangeable nonblocking (RNB) and strictly nonblocking (SNB) networks with/without crosstalk-free constraint, model their routing problems as weak or strong edge-colorings of bipartite graphs, and propose efficient routing algorithms for these networks using parallel processing techniques. This class of networks includes networks constructed from Banyan networks by horizontal concatenation of extra stages and/or vertical stacking of multiple planes. We present a parallel algorithm that runs in $O(\lg^2 N)$ time for the RNB networks of complexities ranging from $O(N \lg N)$ to $O(N^{1.5} \lg N)$ crosspoints and parallel algorithms that run in $O(\min\{d^* \lg N, \sqrt{N}\})$ time for the SNB networks of $O(N^{1.5} \lg N)$ crosspoints, using a completely connected multiprocessor system of N processing elements. Our algorithms can be translated into algorithms with an $O(\lg N \lg \lg N)$ slowdown factor for the class of $N\text{-processor}$ hypercubic networks, whose structures are no more complex than a single plane in the RNB and SNB networks considered.

Index Terms—Banyan network, crosstalk, optical switching, rearrangeable nonblocking network, strictly nonblocking network, switch control, self-routing, graph coloring, parallel algorithm.

1 Introduction

To build a large IP router with capacity of 1 Tb/s and beyond, either electronic or optical switching can be used. The deployment of optical fibers as a transmission medium has prompted searching for the solution to the problem of speed mismatching between transmission and switching. Optical routers have better scalability than electronic routers in terms of switching capacity. However, the required optical technologies are immature for all-optical switching to happen any time soon. A hybrid approach in which optical signals are switched, but both switch control and routing decisions are carried out electronically, becomes more practical. Advances in electro-optic technologies provide a promising choice to meet the increasing demands for high channel bandwidth and low communication latency in optical communication. However, due to the nature of optical devices, optical switches hold their own challenges [26].

1.1 Crosstalk in Photonic Switching

A switching network usually comprises a number of switching elements (SEs), grouped into several stages interconnected by a set of links. Without loss of generality, we assume that an SE is of size 2×2, i.e., it has two inputs and two outputs. The two inputs (respectively, outputs) of an SE intending to be connected with the same output (respectively, input) causes *output link conflict*(respectively, *input link conflict*). If an I/O connection path does not have any link conflict with other connection paths, it is called a *conflict-free* path. Nonblocking switching networks have been favored in switching systems because they can be used to set up any conflict-free one-to-one I/O connection paths. There are three types of nonblocking networks: *strictly nonblocking (SNB)*, *wide-sense nonblocking (WSNB)*, and *rearrangeable nonblocking (RNB)* [3], [13]. In both SNB and WSNB networks, a connection can be established from any idle input to any idle output without disturbing existing connections. In SNB networks any of available conflict-free paths for a connection can be chosen and in WSNB networks, however, a rule must be followed to choose one. The high degree of connection capability in SNB and WSNB networks is at a high hardware cost. RNB networks, usually constructed with lower hardware cost, can establish a conflict-free path for the connection from any idle input to any idle output if the rearrangement of existing connections is allowed.

In an electrical switching network, links are wires and SEs are simple crossbar switches. In an optical switching network, links are implemented by optical waveguides and SEs can be implemented by electro-optical SEs such as common lithium-niobate ($LiNbO_3$) SEs (e.g., [11], [12], [28]). Each electro-optical SE is a directional coupler with two inputs and two outputs. Depending on the amount of voltage at the junction of two waveguides, optical signals carried on either of two inputs can be coupled to either of two outputs. An electronically controlled optical SE can have switching speed ranging from hundreds of picoseconds to tens of nanoseconds [27]. However, due to the nature of optical devices, optical switches introduce additional challenges. One problem is *path dependent loss*, the substantial signal loss is directly proportional to *connection diameter*, the number of SEs on the longest connection path.

- *E. Lu is with the Department of Mathematics and Computer Science, Richard A. Henson School of Science and Technology, Salisbury University, 1101 Camden Ave., Salisbury, MD 21801. Email: ealu@salisbury.edu.*
- *S.Q. Zheng is with the Department of Computer Science, Erik Jonsson School of Engineering and Computer Science, Box 830688, MS EC 31, University of Texas at Dallas, Richardson, TX 75083-0688. Email: sizheng@utdallas.edu.*

Manuscript received 16 Sept. 2003; revised 1 June 2004; accepted 30 Oct. 2004; published online 22 June 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0170-0903.

1045-9219/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

Another problem is *crosstalk*,[1] which is caused by undesired coupling between signals with the same wavelength carried in two waveguides so that two signal channels interfere with each other within an SE.

The crosstalk problem in photonic switching networks adds a new dimension of blocking, called *node conflict*, which happens when more than one connection with the same wavelength passes through the same SE at the same time. A technique called *space dilation* was introduced to avoid node conflict by increasing the number of SEs in a switching network (e.g., [15], [16], [24], [29], [30], [31], [33]).

1.2 Motivation and Main Results

In a switching network, when more than one input request to be connected with the same output, output contention occurs. Output contentions can be resolved by switch scheduling. For a set of connection requests without output contentions, the process of establishing conflict-free connection paths to satisfy these requests is called *switch routing*. A switch routing (or simply, routing) algorithm is needed to find these paths. Once a set of conflict-free paths is found, the SEs on these paths can be properly set up. Routing algorithms play a more fundamental role in WSNB and RNB networks since the nonblockingness depends on them. For SNB networks, routing algorithms tend to be overlooked since a conflict-free path is always guaranteed for the connection from any idle input to any idle output without rerouting the existing connections. An efficient routing algorithm, however, is still needed to find such a conflict-free path for each connection request. Any routing algorithm requiring more than linear time would be considered too slow. Thus, finding efficient algorithms to speed up routing process is crucial for high-speed switching networks.

Recently, a class of multistage nonblocking switching networks has been proposed. In this class, each network, denoted by $B(N, x, p, \alpha)$, has relatively low hardware cost and short connection diameter in terms of the number of SEs. A $B(N, x, p, \alpha)$, $\alpha \in \{0, 1\}$, is constructed by horizontally concatenating $x(\leq \lg N - 1)$ extra stages to an $N \times N$ Banyan-type network, and then vertically stacking p copies of the extended Banyan.[2] $B(N, x, p, 0)$ and $B(N, x, p, 1)$ are similar in structure, but the latter does not allow any two connections with the same wavelength passing through the same SE at the same time while the former does. $B(N, x, p, \alpha)$ contains $\frac{1}{2}p(x + \lg N)N = O(pN \lg N)$ SEs, and its diameter is $O(\lg N)$. $B(N, x, p, 0)$ and $B(N, x, p, 1)$ are suitable for electronic and optical implementation, respectively. It has been shown that $B(N, x, p, \alpha)$ can be SNB, WSNB, and RNB with certain values of x and p for given N and α [15], [16], [21], [30], [31].

The focus of this paper is studying the control aspect of the class $B(N, x, p, \alpha)$ networks in the context of being used as electrical and optical switching networks. In particular, our objective is to speed up routing process using parallel processing techniques. By examining the connection capacity of $B(N, x, p, \alpha)$, we reduce the routing problems for this class of networks to a problem of partitioning a bipartite graph into "disjoint" subgraphs. Three general approaches for solving this type of graph partition problems have been reported. They are matrix decomposition (e.g., [5], [17], [23], [25]), matching (e.g., [6], [7], [9]), and graph edge-coloring (e.g., [6], [7], [10], [19], [22], [32]). For routing, these approaches are essentially equivalent [13]. We model the routing problems for this class of networks as weak and strong edge-colorings of bipartite graphs, which unifies and extends previous models for RNB and SNB networks. Basing on our model, we propose fast routing algorithms for $B(N, x, p, \alpha)$ using parallel processing techniques. We show that the presented parallel routing algorithms can route K connections in $O(\lg N \lg K)$ time for an RNB $B(N, x, p, \alpha)$ and in $O(\min\{d^* \lg N, \sqrt{N}\})$ time for an SNB $B(N, 0, p^*, \alpha)$, where d^* is the degree of the I/O mapping graph of the new connections. Since $K = N$ and $d^* = O(\sqrt{N})$ in the worst case, the proposed algorithms can always route $O(N)$ connections in an RNB $B(N, x, p, \alpha)$ in $O(\lg^2 N)$ time and in an SNB $B(N, x, p, \alpha)$ in $O(\sqrt{N})$ time.

The remainder of this paper is organized as follows: In Section 2, we discuss the topology of $B(N, x, p, \alpha)$. In Section 3, we model routing in $B(N, x, p, \alpha)$ as two coloring problems of an I/O mapping graph $G(N, K, g)$. In Section 4, we propose a fast parallel routing algorithm for RNB $B(N, x, p, \alpha)$ based on a weak g-edge coloring of $G(N, K, g)$. In Section 5, we present parallel routing algorithms for SNB $B(N, x, p, \alpha)$ based on a strong $(2g - 1)$-edge coloring of $G(N, K, g)$. We conclude our paper in Section 6.

2 Nonblocking Networks Based on Banyan Networks

2.1 Banyan-Type Networks

A switching network is a *self-routing network* if any connection within which can be established only by the addresses of its source and destination regardless of other connections. Self-routing is an attractive feature in that no complicated control mechanism is needed for establishing connection. A class of multistage self-routing networks, *Banyan-type* networks, has received considerable attention. A network belonging to this class satisfies the following basic properties:

1. It has $N = 2^n$ inputs, $N = 2^n$ outputs, n-stages, and $N/2$ SEs in each stage.
2. There is a unique path between each input and each output.
3. Let u and v be two SEs in stage i, and let $S_j(u)$ and $S_j(v)$ be two sets of SEs to which u and v can reach in stage j, $0 < j = i + 1 \leq \lg N$, respectively. Then, $S_j(u) \cap S_j(v) = \emptyset$ or $S_j(u) = S_j(v)$ for any u and v.

Because of the above three properties (short connection diameter, unique connection path, uniform modularity, etc.), Banyan-type networks are very attractive for constructing switching networks. Several well-known networks, such as *Banyan*, *Omega*, and *Baseline*, belong to this class. It has been shown that these networks are topologically equivalent [1], [34]. In this paper, we use Baseline network as the representative of Banyan-type networks.

[1] In this paper, the crosstalk is referred to the first-order nonfilterable SE crosstalk [20], [21].

[2] In this paper, $N = 2^n$ ($n = \lg N$) and all logarithms are in base 2.

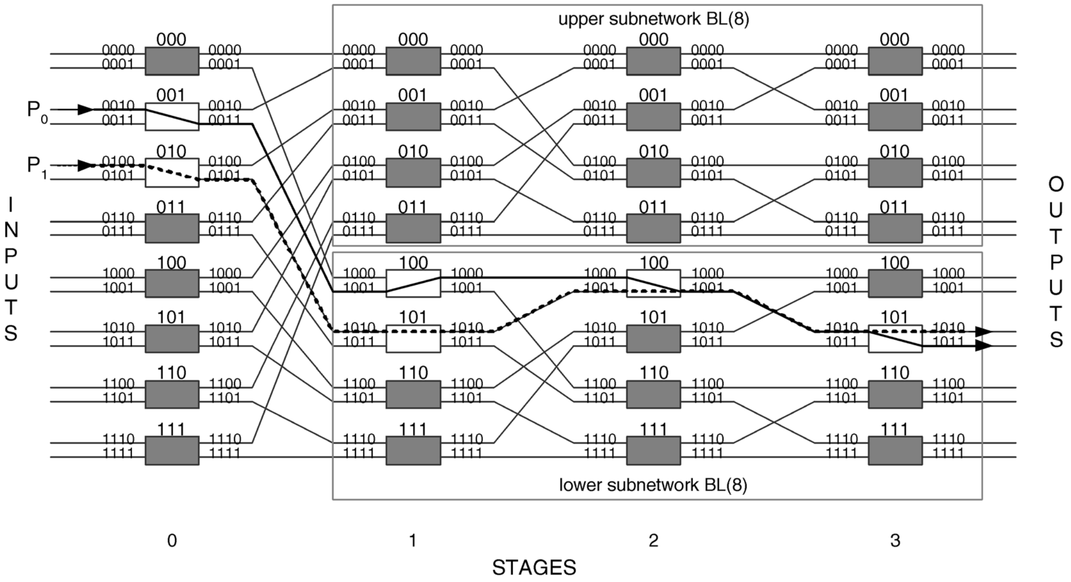


Fig. 1. Self-routing connection paths P_0 and P_1 in $BL(16)$ with link and node conflicts.

An $N \times N$ Baseline network, denoted by $BL(N)$, is constructed recursively. A $BL(2)$ is a 2×2 SE. A $BL(N)$ consists of a switching stage of $N/2$ SEs, and a shuffle connection, followed by a stack of two $BL(N/2)$s. Thus, a $BL(N)$ has $\lg N$ stages labeled by $0, \cdots, n-1$ from left to right, and each stage has $N/2$ SEs labeled by $0, \cdots, N/2-1$ from top to bottom. The upper and lower outputs of each SE in stage i are connected with two $BL(N/2^{i+1})$s, named *upper subnetwork* and *lower subnetwork*, respectively. The N links interconnecting two adjacent stages i and $i+1$ are called *output links* of stage i and *input links* of stage $i+1$. The input (respectively, output) links in the first (respectively, last) stage of $BL(N)$ are connected with N inputs (respectively, outputs) of $BL(N)$. To facilitate our discussions, the labels of stages, links, and SEs are represented by binary numbers. Let $a_l a_{l-1} \cdots a_1 a_0$ be the binary representation of a. We use $\bar{a}$ to denote the integer that has the binary representation $a_l a_{l-1} \cdots a_1 (1-a_0)$. An example is shown in Fig. 1.

The self-routing in $BL(N)$ is decided by the destination, $d_{n-1} d_{n-2} \cdots d_0$, of each connection. If $d_{n-i-1} = 0$, the input of the SE on the connection path in stage i is connected to the SE's upper output, and to the lower output otherwise (i.e., $d_{n-i-1} = 1$). As shown in Fig. 1, connection paths P_0 and P_1 are set up by self-routing in $BL(16)$. In general, the unique path for a connection from source $s_{n-1} \cdots s_0$ to destination $d_{n-1} \cdots d_0$ can be derived as follows: the path enters SE $d_{n-1} \cdots d_{n-i} s_{n-1} \cdots s_{i+1}$ in stage i via input link $d_{n-1} \cdots d_{n-i} s_{n-1} \cdots s_{i+1} s_i$ of the SE and leaving the SE using its output link $d_{n-1} \cdots d_{n-i} s_{n-1} \cdots s_{i+1} d_{n-i-1}$. By this self-routing property, the connection path for any input/output pairs of $BL(N)$ can be computed in $O(\lg N)$ time. Therefore, we have the following simple fact:

Lemma 1. *Given any $K(\leq N)$ one-to-one distinct input/output pairs, the connection paths in $BL(N)$ for these pairs can be computed in $O(\lg N)$ time using N processing elements (PEs) if each PE is assigned to $O(1)$ pairs.*

2.2 Horizontal Concatenation and Vertical Stacking

If Baseline network is used for photonic switching, it is a blocking network since two connections may pass through the same SE, which causes node conflict. Even if Baseline network is used for electronic switching, it is still blocking since two connections may try to pass through the same input (respectively, output) link, which causes input (respectively, output) link conflict. Fig. 1 shows two connection paths P_0 from 0010 to 1011 and P_1 from 0100 to 1010. P_0 and P_1 have output link conflict in stage 2 and input link conflict in stage 3. If each SE is an electro-optic SE in $BL(16)$, then they also have node conflict at SEs 4 and 5 in stages 2 and 3, respectively.

Although a Baseline network is blocking, a nonblocking network can be built by extending it in three ways: horizontal concatenation of extra stages to the back of a Baseline network, vertical stacking of multiple copies of a Baseline network, and the combination of both horizontal concatenation and vertical stacking [15], [16], [30], [31]. In the general approach, a network is constructed by concatenating the mirror image of the first $x(< n)$ stages of $BL(N)$ to the back of a $BL(N)$ to obtain $BL(N, x)$, then vertically making p copies of $BL(N, x)$, where each copy is called a *plane* and, finally, connecting the inputs (respectively, outputs) in the first (respectively, last) stage to N $1 \times p$ splitters (respectively, $p \times 1$ combiners). Specifically, the ith input (respectively, output) of the jth plane is connected with the jth output (respectively, input) of the ith $1 \times p$ splitter (respectively, $p \times 1$ combiner), which is connected with the ith input (respectively, output) of this network. We denote a network constructed in this way by $B(N, x, p, \alpha)$, where α is *crosstalk factor*: $\alpha = 0$ if the network has no crosstalk-free constraint (i.e., the network has only link conflict-free constraint) and $\alpha = 1$ if the network has crosstalk-free constraint (i.e., the network has node conflict-free constraint). Asymptotically, the cost of $B(N, x, p, \alpha)$ is $O(pN \lg N)$, measured either by the number of SEs or by the number of crosspoints [13]. Note that $B(N, x, p, \alpha)$ can be

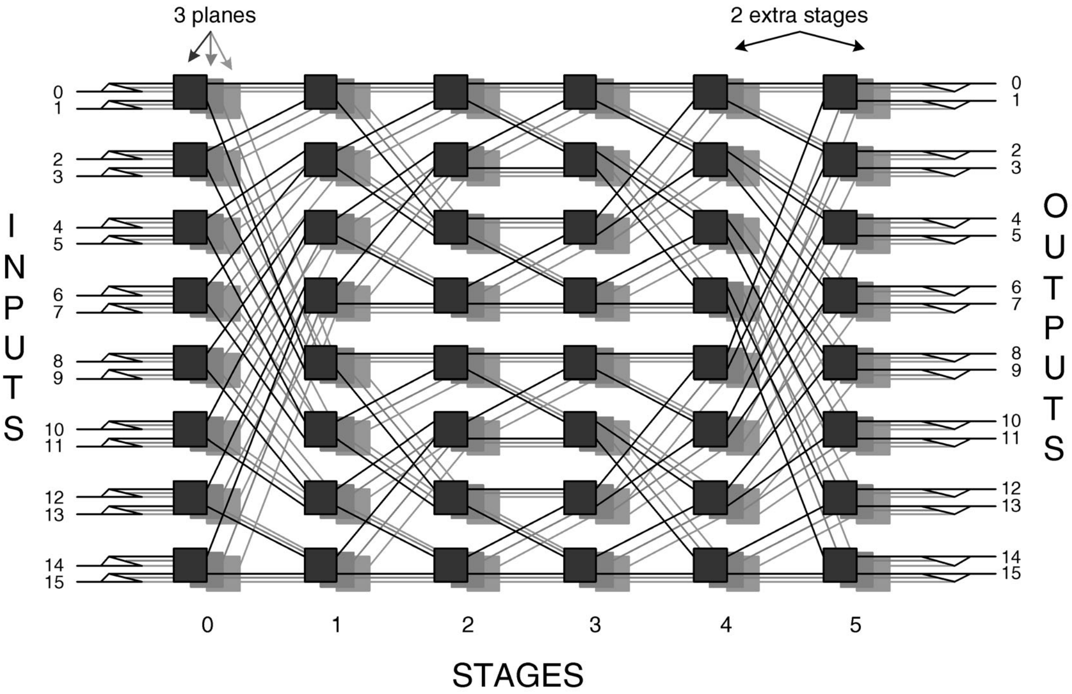


Fig. 2. A network $B(16, 2, 3, \alpha)$.

nonblocking for certain combinations of N, x, p, and α. The complexity of RNB networks considered in this paper have complexities ranging from $O(N \lg N)$ to from $O(N^{1.5} \lg N)$ and the SNB networks considered have complexity $O(N^{1.5} \lg N)$.

In $B(N, x, 1, \alpha)$, a *subnetwork*, denoted by $B(N, x, 1/2^l, \alpha)$ $(0 \leq l \leq n-1)$ is defined as a $B(N/2^l, \max\{x-l, 0\}, 1, \alpha)$ from stage l to stage $n + \max\{x-l, 0\} - 1$. Fig. 2 shows an example of $B(16, 2, 3, \alpha)$, which contains three planes of $B(16, 2, 1, \alpha)$, and each $B(16, 2, 1, \alpha)$ is constructed from $B(16, 0, 1, \alpha)$ by adding two extra stages. Each $B(16, 2, 1, \alpha)$ contains two $B(16, 2, 1/2, \alpha)$s, each being $B(8, 1, 1, \alpha)$, and four $B(16, 2, 1/4, \alpha)$s, each being $B(4, 0, 1, \alpha)$.

2.3 Designing Parallel Switch Routing Algorithms

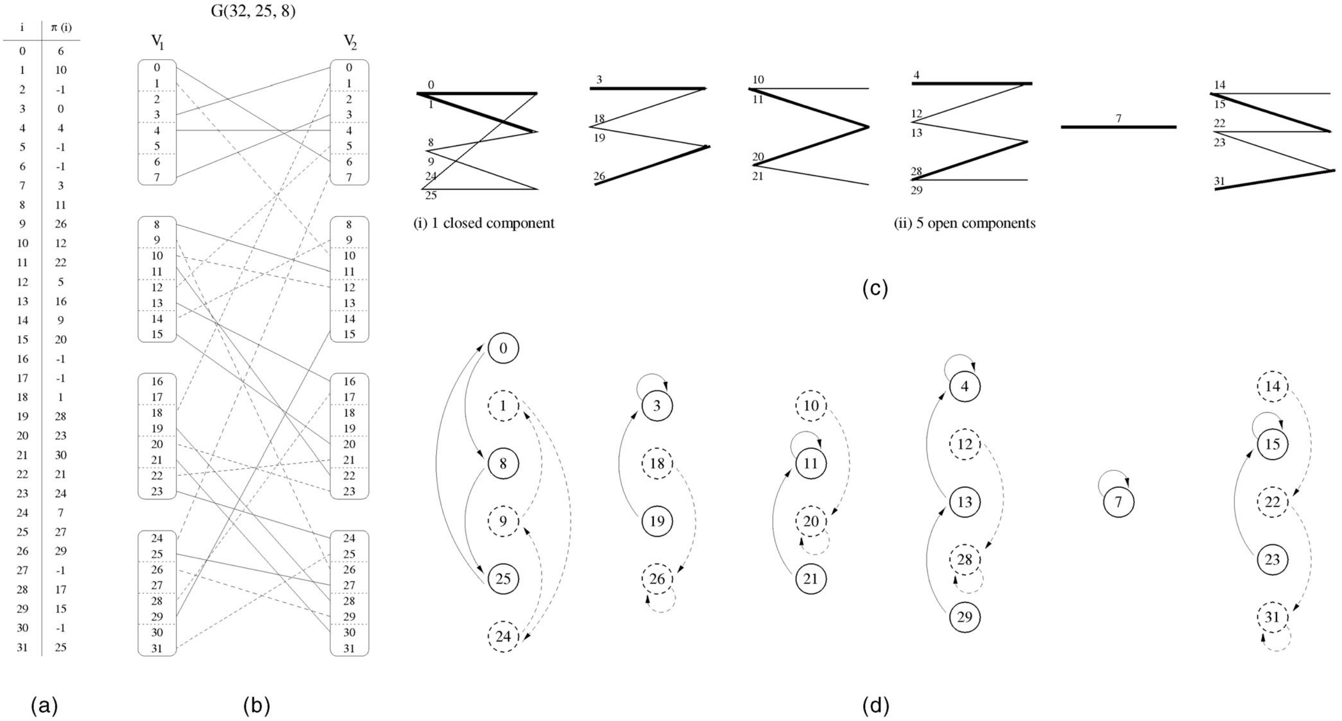
A trivial lower bound on the time for routing K $(0 \leq K \leq N)$ connections sequentially in $B(N, x, p, \alpha)$ is $\Omega(K \lg N)$. This lower bound is obtained by assuming that for any connection it takes $O(1)$ time to correctly guess which plane to use without conflict and $O(\lg N)$ time to compute the connection path in that plane. Clearly, correctly assigning connections to planes is not a simple task, when $x \neq 0$ and $p > 1$. When the number of connection requests is large, the routing time complexity is greater than $O(N)$. Parallel processing techniques should be used to meet the stringent real-time timing requirement [13]. To the best of our knowledge, except for some special cases such as Banyan network (i.e., $B(N, 0, 1, \alpha)$) and Benes network (i.e., $B(N, \lg N - 1, 1, \alpha)$), no effort of investigating faster routing for the whole class of these networks has been reported in the literature.

We choose to present our parallel algorithms for a completely connected multiprocessor system. A completely connected multiprocessor system of size N consists of N processing elements (PEs), PE_i, $0 \leq i \leq N-1$, connected in such a way that there is a connection between every pair of PEs. We assume that each PE can communicate with at most one PE during a communication step. The time complexity of an algorithm on such a multiprocessor system is measured in terms of the total number of parallel computation and communication steps required by the algorithm. Such a multiprocessor system is by no means to be practical, but used as a general abstract model to derive parallel algorithms. Efficient algorithms on more realistic models, such as the class of hypercubic parallel computers, whose architectural complexity is the same as that of a single plane of $B(N, x, p, \alpha)$, can be easily obtained from our algorithms.

3 Graph Model

3.1 I/O Mapping Graphs

For $B(N, x, p, \alpha)$, let I be a set of N inputs, $I_0, \cdots, I_{N-1}$, and O be a set of N outputs, $O_0, \cdots, O_{N-1}$. Let $g = 2^i$, $0 \leq i \leq n$. Then, the kth *modulo-g input group* comprises inputs $I_{(k-1)g}, I_{(k-1)g+1}, \cdots, I_{kg-1}$, and the kth *modulo-g output group* comprises outputs $O_{(k-1)g}, O_{(k-1)g+1}, \cdots, O_{kg-1}$, where $1 \leq k \leq N/g$. Let $\pi : I \longmapsto O$ be an *I/O mapping* that indicates connections from I to O. If there is a connection from I_i to O_j, then set $\pi(i) = j$ and $\pi^{-1}(j) = i$; otherwise, set $\pi(i) = -1$. If $j \neq \pi(i)$ for any I_i, then set $\pi^{-1}(j) = -1$. We say that an input (respectively, output, link, SE) is *active* if it is on a connection path, and *idle* otherwise. An I/O mapping from I to O is *one-to-one* if each I_i is mapped to at most one O_j and $\pi(i) \neq \pi(j)$ for any $i \neq j$. In this paper, all I/O mappings are one-to-one and all connections belong to a one-to-one I/O mapping. Our goal is to quickly route $K(\leq N)$ link (respectively, node) conflict-free paths for K connections of



i	π (i)
0	6
1	10
2	-1
3	0
4	4
5	-1
6	-1
7	3
8	11
9	26
10	12
11	22
12	5
13	16
14	9
15	20
16	-1
17	-1
18	1
19	28
20	23
21	30
22	21
23	24
24	7
25	27
26	29
27	-1
28	17
29	15
30	-1
31	25

Fig. 3. Finding a balanced 2-coloring: (a) An I/O mapping. (b) A balanced 2-coloring of an I/O mapping graph $G(32, 25, 8)$. (c) A set of components, where the Reps of each component are marked as dark lines and edges are labeled by their corresponding inputs. (d) Pointer initialization for pointer jumping.

any I/O mapping in $B(N, x, p, 0)$ (respectively, $B(N, x, p, 1)$). To achieve this goal, we decompose a set of connections into disjoint subsets, and route each subset in one plane of $B(N, x, p, \alpha)$ so that each subset is feasible for its assigned plane.

Given any I/O mapping with K connections for $B(N, x, p, \alpha)$, we construct a graph $G(N, K, g)$, named *I/O mapping graph*, as follows: The vertex set consists of two parts, V_1 and V_2. Each of them has N/g vertices labeled from 0 to $N/g - 1$. Each modulo-g input (respectively, output) group is represented by a vertex in V_1 (respectively, V_2). There is an edge between vertex $\lfloor i/g \rfloor$ in V_1 and vertex $\lfloor j/g \rfloor$ in V_2 if $j = \pi(i)$. Thus, $G(N, K, g)$ is a bipartite graph with N/g vertices in each of V_1 and V_2 and K edges, where at most g edges are incident at any vertex. Clearly, the *degree* of $G(N, K, g)$, the maximum number of edges incident at a vertex, is no larger than g. Since there may be more than one connection from a modulo-g input group to the same modulo-g output group, $G(N, K, g)$ may have *parallel edges*, the edges between the same two vertices, and it may be a multigraph. However, there is a one-to-one correspondence between active inputs/outputs in an I/O mapping and the edges in the I/O mapping graph and, thus, we can label each edge by its corresponding input.

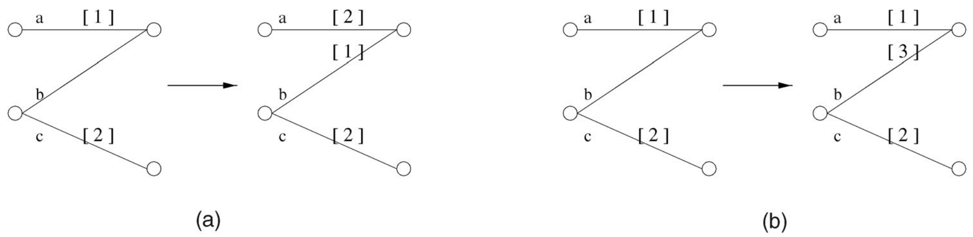
An edge e is called the *left edge* (respectively, *right edge*) of edge f if $e = \bar{f}$ (respectively, $\pi(e) = \overline{\pi(f)}$). Any edge has at most one left edge and at most one right edge in $G(N, K, g)$. Two edges e and f are called *neighboring edges* if e is the left or right edge of f. We define a *linear component* (or simply, a *component*) of $G(N, K, g)$ as follows: two edges e and f belong to the same component if and only if there is a sequence of edges $e = e_1, \cdots, e_j = f$ such that e_i and e_{i+1}, $1 \leq i \leq j - 1$, are neighboring edges. If every edge in a component has two neighboring edges, the component is called a *closed component*; otherwise, it is called an *open component*. By generalizing "neighboring edge" to an equivalent relation, each edge is in exactly one component and, thus, components are edge disjoint in $G(N, K, g)$. Fig. 3a shows an I/O mapping with 32 inputs, 25 of which are active. Fig. 3b shows the I/O mapping graph $G(32, 25, 8)$ of Fig. 3a, where V_1 (respectively, V_2) of $G(32, 25, 8)$ has four vertices and each vertex in V_1 (respectively, V_2) includes eight inputs (respectively, outputs) belonging to the same modulo-8 input (respectively, output) group. Fig. 3c shows all components of $G(32, 25, 8)$ in Fig. 3b.

3.2 Graph Coloring and Nonblockingness

Let us study the connection capability of $B(N, x, p, \alpha)$ first. We say that two connections *share* a modulo-g input (respectively, output) group if their sources (respectively, destinations) are in the same modulo-g input (respectively, output) group.

Lemma 2. *For any connection set C of $B(N, 0, 1, \alpha)$, if no two connections in C share any* modulo-g *input (respectively, output) group, then the connection paths for C satisfy the following conditions: 1) they are node conflict-free in the first (respectively, last)* $\lg g$ *stages, and 2) they are input link conflict-free in the first* $\lg g + 1$ *(respectively, last* $\lg g$*) stages and output link conflict-free in the first* $\lg g$ *(respectively, last* $\lg g + 1$*) stages.*

Lemma 3. *For any pair of input and output in $B(N, x, 1, \alpha)$, there are 2^x paths connecting them.*



(a)

(b)

Fig. 4. (a) A (weak) edge-coloring. (b) A strong edge-coloring.

It is easy to verify that Lemmas 2 and 3 are true according to the topology of $BL(N)$ (refer to [21] for formal proofs). We say that a set C of I/O connections is *feasible* for $B(N, x, p, 0)$ (respectively, $B(N, x, p, 1)$) if they can be routed without any link (respectively, node) conflict. Using the above two lemmas, the following claim can be easily derived from the results of [21].

Lemma 4. *Given a connection set C of $B(N, x, 1, \alpha)$, if any two connections in C do not share any* modulo-$2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$ *input group and also do not share any* modulo-$2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$ *output group, then C is feasible for $B(N, x, 1, \alpha)$.*

By Lemma 4, if we assign the connections of $B(N, x, p, \alpha)$ with sources (respectively, destinations) passing through the same modulo-g input (respectively, output) group to different planes, then we can route connections in $B(N, x, p, \alpha)$ without conflict. Thus, in order to route conflict-free connections in $B(N, x, p, \alpha)$, we first need to determine which plane to be used for each connection. By constructing an I/O mapping graph $G(N, K, g)$ with $g = 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$, we can reduce the problem of routing K connections in $B(N, x, p, \alpha)$ to the following two graph coloring problems:

Weak Edge Coloring Problem (WEC problem): Given an I/O mapping graph $G(N, K, g)$ with $K_0 (< K)$ colored edges, color K edges with a set of colors such that no two edges with the same color are incident at the same vertex of $G(N, K, g)$ with changing the colors of the K_0 colored edges allowed. If we can find a weak edge-coloring of $G(N, K, g)$ using at most c_1 different colors, we call this coloring *a (weak)*[3] c_1-edge *coloring* of $G(N, K, g)$.

Strong Edge Coloring Problem(SEC problem): Given an I/O mapping graph $G(N, K, g)$ with $K_0 (< K)$ colored edges, color $K - K_0$ uncolored edges with a set of colors such that no two edges with the same color are incident at the same vertex of $G(N, K, g)$ without changing the colors of the K_0 colored edges. If we can find a strong edge-coloring of $G(N, K, g)$ using at most c_2 different colors, we call this coloring *a strong* c_2-edge *coloring* of $G(N, K, g)$.

If we consider the colored (respectively, uncolored) edges in $G(N, K, g)$ as the existing (respectively, new) connections in $B(N, x, p, \alpha)$, a solution to the WEC problem is a plane assignment for routing in an RNB network since we can reroute existing connections, and a solution to the SEC problem is a plane assignment for routing in an SNB network since rerouting existing connections is prohibited. Clearly, for the same $G(N, K, g)$, $c_1 \leq c_2$. In Fig. 4, we show a simple example. There are three edges labeled a, b, c, respectively. Edges a and b have already been colored using colors 1 and 2, respectively. A WEC solution is given in Fig. 4a, and an SEC solution is given in Fig. 4b. Note that, in Fig. 4b, an additional color is needed for edge b because the colors of existing colored edges a and c cannot be changed. To our knowledge, no parallel algorithm for the SEC problem has been reported in the literature.

4 Routing in Rearrangeable Nonblocking Networks

4.1 Rearrangeable Nonblockingness of $B(N, x, p, \alpha)$

The following claim is implied by the results of [21].

Lemma 5. *If $p \geq 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$ for $0 \leq x \leq n-1$, then $B(N, x, p, \alpha)$ is rearrangeable nonblocking.*

It is important to note that the minimum value of p in Lemma 5 equals to the value of g in Lemma 4, where p is the number of $B(N, x, 1, \alpha)$ planes required for $B(N, x, p, \alpha)$ to be rearrangeable nonblocking. The number of crosspoints in such an RNB network is $O(N \lg N)$ for $x = n - 1$ and $O(N^{1.5} \lg N)$ for $x = 0$. By Lemmas 4 and 5, if we assign the connections (including existing and new connections) sharing the same modulo-g input/output group to different planes, the connections assigned to each plane are feasible for that plane. Then, the routing can be completed by finding conflict-free connection paths within each plane. The following known fact is useful.

Lemma 6. *Every bipartite multigraph G has a $\Delta(G)$-*edge *coloring, where $\Delta(G)$ is the degree of G.*

By Lemma 6 (see a proof in [4]), if we set $g = 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$ in $G(N, K, g)$, the plane assignments for a set of connections in RNB $B(N, x, p, \alpha)$ can be solved by finding a g-edge coloring of $G(N, K, g)$.

4.2 Algorithm for Balanced 2-Coloring of $G(N, K, g)$

In order to solve WEC problem efficiently, we present an algorithm for a related problem, named *balanced 2-coloring problem*: Given an I/O mapping graph $G(N, K, g)$, color its edges with two colors so that every vertex is adjacent to at most $g/2$ edges with one color and $g/2$ with the other.

Our algorithm is for a completely connected multiprocessor system of size N consists of N PEs. Initially, each PE_i reads $\pi(i)$ from input i and sets the value of π^{-1} in $\mathrm{PE}_{\pi(i)}$ as i. Then, the algorithm performs the following two steps.

[3] The definition of weak edge-coloring is the same as the definition of edge-coloring in graph theory. Thus, we omit "weak" in the rest of this paper.

Step 1. Divide the I/O mapping graph $G(N,K,g)$ into a set of components. This step can be done by each edge finding its left edge $\bar{i}$ and right edge $\pi^{-1}(\overline{\pi(i)})$.

Step 2. Color components with two colors, red and blue, so that neighboring edges in each component have different colors.

Each component has two specific representatives, simply referred to *Reps.* (There is an exception: for the component with length of 1, there is only one Rep, which is itself.). For closed and open components, the Reps are defined differently. For a closed component, we define two edges with the minimum labels as two Reps; for an open component, if an edge e has no left edge or e's left edge has no right edge, e is defined as one Rep. Fig. 3c shows the Reps of all possible types of components. Step 2 can be done by coloring edges with the Reps as references using the pointer jumping technique in [14]. At the beginning, each edge sets its pointer to point to the right edge of its left edge if it exists and to itself otherwise. By doing so, two disjoint directed cycles are formed for a closed component, and two disjoint directed paths are formed for an open component with more than one edge, each containing a Rep. For an open component, furthermore, the end pointer of every directed path is pointing to one of the Reps. For example, Fig. 3d shows that the directed cycles and paths formed from the components of Fig. 3c. Then, by performing $\lceil \lg K/2 \rceil$ times of parallel pointer jumping, each edge finds the Rep belonging to the same directed cycle or path. Finally, each edge can be colored by comparing the value of the Rep found by itself with that by its neighbor. That is, if the value of the Rep founded by an edge is no larger than its neighbor's, color the edge with red; otherwise, color it with blue. The detailed implementation of a balanced 2-coloring algorithm is referred to Algorithm 1[4] (see Fig. 5), and the correctness and time complexity of this algorithm are given in the following theorem.

Theorem 1. *A balanced 2-coloring of any $G(N,K,g)$ can be found in $O(\lg K)$ time using a completely connected multiprocessor system of N PEs.*

Proof. Given an I/O mapping graph $G(N,K,g)$, Step 1 can be done in $O(1)$ time using a completely connected multiprocessor system of N PEs. In Step 2, since the length of each directed cycle or path is at most $\lceil K/2 \rceil$, each edge can find a Rep by $\lceil \lg K/2 \rceil$ times of pointer jumping. Clearly, all edges in the same directed cycle or path are colored with the same color since they find the same Rep. The pointer initialization implies that each edge and its neighboring edge are in different directed cycle or path and, thus, they have different colors. By the definition of left/right edge, there are no more than $g/2$ pairs of neighboring edges incident at any vertex of $G(N,K,g)$. Thus, the coloring of all components compose a balanced 2-coloring of $G(N,K,g)$. Therefore, a balanced 2-coloring of any $G(N,K,g)$ can be found in $O(\lg K)$ time. □

4.3 Algorithm for g-Edge Coloring of $G(N,K,g)$

Based on the balanced 2-coloring algorithm, a WEC solution to any I/O mapping graph $G(N,K,g)$ with no more than g colors can be found as follows: Let d be the degree of $G(N,K,g)$. Let k be the smallest integer such that $d \le 2^k$. Clearly, $0 \le k \le \lg g$ since $d \le g$. First, remove colors of the K_0 colored edges. Then, perform at most $\lceil \lg d \rceil$ iterations as follows: In initial iteration (i.e., iteration 0), we find a balanced 2-coloring of $G(N,K,g)$ using colors 0 and 1 if $d > 1$, and let G_0 and G_1 be the graphs induced by the edges with colors 0 and 1, respectively. If $\Delta(G_0) > 1$ (respectively, $\Delta(G_1) > 1$), we execute iteration 1 to find a balanced 2-coloring for G_0 (respectively, G_1) using colors 00 and 01 (respectively, 10 and 11). This process recursively continues in a binary tree fashion until a solution to WEC is reached. More formally, in each recursive iteration i, $1 \le i \le \lceil \lg d \rceil - 1$, we find a balanced 2-coloring for each graph G_z using colors $z0$ and $z1$ (i.e., concatenate 0 or 1 with z) if $\Delta(G_z) > 1$, where z is a binary representation of an integer in $\{0, 1, \cdots, 2^i - 1\}$ denoting the color of edges in G_z in iteration $i-1$.

Theorem 2. *For any I/O mapping graph $G(N,K,g)$, a g-edge coloring can be found in $O(\lg d \cdot \lg K)$ time using a completely connected multiprocessor system of N PEs, where d is the degree of $G(N,K,g)$.*

Proof. Let $d' = 2^k$ such that k is the smallest integer satisfying $d \le 2^k$. We prove the theorem by induction on k. If $k = 1$, it is true since a balanced 2-coloring is a 2-edge coloring by Theorem 1. Assume that for any $k < m \le n$, the theorem holds. Now, we prove that the theorem holds for $k = m$. First, we find a balanced 2-coloring of $G(N,K,g)$, which can be done in $O(\lg K)$ time by Theorem 1. Let G_0 and G_1 be the graphs induced by the edges of two different colors from this balanced 2-coloring. By the definition of balanced 2-coloring, we know that $\Delta(G_0) \le d'/2$ and $\Delta(G_1) \le d'/2$. By the hypothesis, we can find a $(d'/2)$-edge coloring for each of G_0 and G_1 in $O((k-1) \cdot \lg K)$ time on a completely connected multiprocessor subsystem of $|E(G_0)|$ and $|E(G_1)|$ PEs, respectively. These two colorings can be carried out simultaneously since $E(G_0) \cap E(G_1) = \emptyset$. The $(d'/2)$-edge colorings of G_0 and G_1 compose a d'-edge coloring of $G(N,K,g)$, which takes total $O(k \cdot \lg K)$ time using a completely connected multiprocessor system of N PEs. Since $d'/2 < d \le d' \le g$, this theorem holds. □

4.4 Parallel Routing in a Plane

We have shown how to assign each connection to a plane in an RNB $B(N,x,p,\alpha)$. In this section, we show how connections are routed within each plane.

Lemma 7. *Let C be a set of feasible connections for $B(N,x,1,\alpha)$. If each connection in C is routed in the first and last x stages such that the output link in stage i and the input link in stage $\lg N - i$ on each connection are connected with the same subnetwork $B(N,x,1/2^{i+1},\alpha)$, $0 \le i \le x-1$, then C can be routed by self-routing in the middle $\lg N - x$ stages.*

Proof. By the topology of $B(N,x,1,\alpha)$, we know that each connection must pass through the same subnetwork $B(N,x,1/2^i,\alpha)$, $0 \le i \le \lg N - 1$. Since the middle $\lg N - x$ stages of $B(N,x,1,\alpha)$ consists of 2^x $BL(\frac{N}{2^x})$s, this lemma is true. □

Theorem 3. *Let C be a set of K feasible connections of $B(N,x,1,\alpha)$. Then, C can be correctly routed in $O(x \lg K + \lg N)$ time using a completely connected multiprocessor system of N PEs.*

[4] We use operator ":=" to denote an assignment local to a PE or to the control unit, and use operate "← " to denote an assignment requiring some interprocessor communication.

```
Input: $G(N, K, g)$
Output: a balanced 2-coloring of $G(N, K, g)$
  for all $PE_i$, $0 \leq i \leq N-1$, do
    $l(i) := r(i) := -1$;    /* $l(i)$, $r(i)$ are the left edge and right edge of edge $i$ respectively. */
    if $\pi(i) \neq -1$ then
      if $\pi(\bar{i}) \neq -1$ then
        $l(i) := \bar{i}$;
      end if
      if $\pi^{-1}(\overline{\pi(i)}) \neq -1$ then
        $r(i) \leftarrow \pi^{-1}(\overline{\pi(i)})$;
      end if
      if $l(i) \neq -1$ and $r(l(i)) \neq -1$ then
        $q(i) \leftarrow r(l(i))$;    /* $q(i)$ is a pointer */
        $p(i) := 0$;    /* $p(i)$ is used to find out if edge $i$ in a path or cycle */
      else
        $q(i) := i$;
        $p(i) := 1$;    /* the edge $i$ in a path */
      end if
      $m(i) := m'(i) := i$;    /* $m(i)$ (resp. $m'(i)$) is used to find the Rep found by edge $i$ (resp. $i$'s neighbor) */
      for $t := 1$ to $\lceil \lg K/2 \rceil$ do
        $m(i) \leftarrow \min\{m(i), m(q(i))\}$;
        $p(i) \leftarrow p(q(i))$;
        $q(i) \leftarrow q(q(i))$;
      end for
      if $p(i) = 1$ then
        $m(i) := q(i)$;    /* the Rep found by the edge $i$ is the label of PE to which $i$ is pointing */
      end if
      if $l(i) \neq -1$ then
        $m'(i) \leftarrow m(l(i))$;
      else if $r(i) \neq -1$ then
        $m'(i) \leftarrow m(r(i))$;
      end if
      if $m(i) \leq m'(i)$ then
        $c(i) := 0$;    /* color $i$ as red */
      else
        $c(i) := 1$;    /* color $i$ as blue */
      end if
    end if
  end for
```

Fig. 5. Algorithm 1: A balanced 2-coloring of an I/O mapping graph.

Proof. By Lemma 7, what we only need to do is to route C correctly in the first and last x stages for $x \geq 1$. By the topology of $B(N, x, 1, \alpha)$, we know that the output link in stage i and the input link in stage $\lg N - i$ on each connection are connected with the same subnetwork $B(N, x, 1/2^{i+1}, \alpha)$, $0 \leq i \leq x-1$. Thus, we need to decide which subnetwork is to be used for each connection since there are 2^i $B(N, x, 1/2^i, \alpha)$s. This can be reduced to a 2-edge coloring of a bipartite graph with degree of 2. For each subnetwork $B(N, x, 1/2^i, \alpha)$, $0 \le i \le x-1$, we construct an I/O mapping graph $G(N/2^i, K_i, 2)$, where K_i is the number of connections passing through it. We color the edges of $G(N/2^i, K_i, 2)$ with two different colors and assign the connections (edges) with the same color to the same subnetwork $B(N, x, 1/2^{i+1}, \alpha)$. Specifically, in each iteration i, $0 \le i \le x-1$, we run g-edge coloring algorithm for 2^i $G(N/2^i, K_i, 2)$s with $g = 2$. By Theorem 2, each iteration can be done in $O(\lg K)$ time. Thus, the time to route K feasible connections in the first and last x stages is $O(x \lg K)$. By Lemmas 1 and 7, we can route the connections in the middle $\lg N - x$ stages by self-routing, which takes $\lg N - x$ time. Therefore, the total time to route K feasible connections of $B(N, x, 1, \alpha)$ is $O(x \lg K + \lg N)$ using a completely connected multiprocessor system of N PEs. □

4.5 Overall Routing Performance

Theorem 4. *For any RNB $B(N, x, p, \alpha)$ such that $p \ge 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$, K connections (including existing and new connections) can be correctly routed in $O(\lg K \lg N)$ time using a completely connected multiprocessor system of N PEs.*

Proof. Let $g = 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$. By Theorem 2, we can find a g-edge coloring of the I/O mapping graph $G(N, K, g)$ in $O(\lg d \lg K)$ time, where d is the degree of $G(N, K, g)$. By Lemma 4, we assign the connections with the same color to the same plane. In each plane $B(N, x, 1, \alpha)$, by Theorem 3, we can route the connections in $O(x \lg K + \lg N)$ time. Since $x < \lg N$, $d \le g = 2^{\lfloor \frac{n-x+\alpha}{2} \rfloor}$, the total time is $O((x + \lg d) \lg K + \lg N) = O(\lg K \lg N)$. □

By Lemma 5, for special cases of an RNB $B(N, 0, p, \alpha)$ and an RNB $B(N, n-1, p, \alpha)$, the minimum number p of planes of Baseline network and Benes network, equals $2^{\lfloor \frac{n+\alpha}{2} \rfloor}$ and $2^{\lfloor \frac{1+\alpha}{2} \rfloor}$, respectively. Consequently, we can route N connections in $O(\lg^2 N)$ time for both $B(N, n-1, 1, \alpha)$ and $B(N, 0, \lfloor \frac{n+\alpha}{2} \rfloor, \alpha)$, which have $O(N \lg N)$ and $O(N^{1.5} \lg N)$ crosspoints, respectively. For the RNB $B(N, n-1, 1, 0)$, which is the electronic Benes network, this performance is the same as the best known results reported in [19], [22].

5 Routing in Strictly Nonblocking Networks

5.1 Strict Nonblockingness

The following lemma can be easily derived from the results of [31].

Lemma 8. *If*

$$p \ge \begin{cases} (1+\alpha)x + 2^{\frac{n-x}{2}}(\frac{3}{2} + \frac{1}{2}\alpha) - 1, & \text{for even } n-x \\ (1+\alpha)x + 2^{\frac{n-x+1}{2}}(1 + \frac{1}{2}\alpha) - 1, & \text{for odd } n-x, \end{cases}$$

then $B(N, x, p, \alpha)$ is strictly nonblocking.

For an SNB network, we can route new connections (as long as these connections form an I/O mapping from idle inputs to idle outputs) without disturbing the existing ones; however, this routing problem is harder than that in an RNB network when we need to route the new connections simultaneously. Based on the discussions in Section 3.2, we know that the routing problem for an SNB $B(N, x, p, \alpha)$ can be solved by finding a strong edge-coloring of the I/O mapping graph $G(N, K, g)$.

Lemma 9. *Any multigraph G has a strong $(2\Delta - 1)$-edge coloring, where Δ is the degree of G.*

Proof. Consider coloring edges in an arbitrary order. Since each edge in G is adjacent to at most $2\Delta - 2$ edges, any uncolored edge in G can always be assigned a color so that the total number of colors used is no larger than $2\Delta - 1$. □

We consider a subclass of SNB networks, $B(N, 0, p^*, \alpha)$ with $p^* = 2^{\lfloor \frac{n+\alpha}{2} \rfloor + 1} - 1$. By Lemma 8, we know that $B(N, 0, p^*, \alpha)$ is an SNB network. Since each plane of $B(N, 0, p^*, \alpha)$ is a Baseline network, the routing of connections in any plane can be done by self-routing. Thus, the problem of routing connections in $B(N, 0, p^*, \alpha)$ is reduced to finding a plane for each new connection so that all connections, including existing ones, are conflict-free. By Lemmas 4 and 9, this can be done by finding a strong $(2g - 1)$-edge coloring for $G(N, K, g)$ of $B(N, 0, p^*, \alpha)$ with K_0 existing connections and $K - K_0$ new connections, where $g = 2^{\lfloor \frac{n+\alpha}{2} \rfloor} = \frac{p^*+1}{2}$. In the next two sections, we present two parallel algorithms to find a strong $(2g - 1)$-edge coloring of $G(N, K, g)$ using different approaches.

Before presenting our algorithms, we give a couple of definitions. Let $G(N, K - K_0, g)$ and $G(N, K_0, g)$ denote the graphs obtained from $G(N, K, g)$ by removing the K_0 colored edges and only keeping K_0 colored edges, respectively. Since $G(N, K, g)$ is a bipartite multigraph, $G(N, K - K_0, g)$ is also a bipartite multigraph with two vertex set $V_1 = \{v'_1, v'_2, \cdots, v'_{N/g}\}$ and $V_2 = \{v''_1, v''_2, \cdots, v''_{N/g}\}$ such that v'_k and v''_k corresponds to the kth modulo-g input group and output group, respectively. We say color c is *free* at vertex v if none of edges adjacent to v has color c. If color c is free at two ends of edge e, then c is *free* for e. One edge e is *conflict* with another edge f if e and f are adjacent to each other and they have the same color.

5.2 First Algorithm for Strong Edge-Coloring of $G(N, K, g)$

The idea of the first algorithm is that we first partition the set of uncolored edges into edge-disjoint subsets, and then we color the subsets one by one. The edges in the same subset may be colored differently depending on the free colors for each edge. The edge-disjoint subsets can be found by finding a set of matchings of $G(N, K - K_0, g)$, where a *matching* of $G(N, K - K_0, g)$ is defined as a set M of edges in $G(N, K - K_0, g)$ such that no two edges in M are adjacent.

Let d^* is the degree of $G(N, K - K_0, g)$. Let $d' = 2^k$ such that k is the smallest integer satisfying $d^* \le 2^k$. Our first algorithm computes a strong $(2g - 1)$-edge coloring of $G(N, K, g)$ with $K_0 (< K)$ colored edges by performing the following two steps.

Step 1: Find a set of matchings $\{M_1, M_2, \cdots, M_{d'}\}$ of $G(N, K - K_0, g)$.

Step 2: For i from 1 to d', do the following: Color the edges in M_i without changing the colors of the edges in $G(N, K_0, g) \bigcup (\cup_{j<i} M_j)$.

Finding a set of d' matchings in a graph is equivalent to coloring the edges in the graph with d' different colors, because edges with the same color are not adjacent to each other. Thus, Step 1 can be done by finding a d'-edge coloring of $G(N, K - K_0, g)$ using the algorithm described in Section 4. This d'-edge coloring divides $K - K_0$ uncolored edges (corresponding to new connections) into d' matchings. By Theorem 2, Step 1 takes $O(\lg d' \cdot \lg(K - K_0)) = O(\lg d^* \cdot \lg(K - K_0))$ time using a completely connected multiprocessor system of N PEs.

In $G(N, K, g)$, each edge is adjacent to at most $2g - 2$ edges and, hence, there are at most $2g - 2$ colored edges adjacent to each edge in a matching M_i. Since edges with the same color cannot be adjacent, we can color every edge in a matching by one of the unused colors. This can be done by parallel searching for a free color among $2g - 1$ colors as follows: Associate a Boolean array $C[02g - 2]$ of $2g - 1$ elements with each vertex in $G(N, K, g)$, with $C[r] = 0$ if and only if an edge adjacent to the vertex has been colored with color r. Consider an edge e in M_i that connects vertices v' and v'' of $G(N, K, g)$, and let $C_{v'}$ and $C_{v''}$ be the C array associated with vertices v' and v'', respectively. Performing bit-wise AND operation on $C_{v'}$ and $C_{v''}$ and obtain a Boolean array $D_{v',v''}$ such that $D_{v',v''}[s] = C_{v'}[s] \wedge C_{v''}[s]$, $0 \le s \le 2g - 2$. Then, $D_{v',v''}[t] = 1$ if and only if color t is free for edge e. We can assign $g/2$ PEs to each vertex v of $G(N, K, g)$, and these PEs collectively maintain C_v. Then, using g PEs, $D_{v',v''}$ can be computed $O(1)$ time, and finding some t such that $D_{v',v''}[t] = 1$ by performing a parallel binary prefix sums operation on $D_{v',v''}$, which takes $O(\lg g)$ time. Since no two edges are adjacent in a matching, uncolored edges in the matching can be colored simultaneously by their assigned PEs in $O(\lg g)$ time, and Step 2 takes $O(d' \lg g)$ time. Since $d'/2 < d^* \le d'$, $O(d' \lg g) = O(d^* \lg g)$. Therefore, we have the following claim.

Theorem 5. *For any I/O mapping graph $G(N, K, g)$ with $K_0 (< K)$ colored edges, a strong $(2g - 1)$-edge coloring can be found in $O(\lg d^* \lg(K - K_0) + d^* \lg g)$ time using a completely connected multiprocessor system of N PEs, where d^* is the degree of $G(N, K - K_0, g)$.*

5.3 Second Algorithm for Strong Edge-Coloring of $G(N, K, g)$

Let $E_{i,j} = \{e_{i,j} | e_{i,j} = (v'_i, v''_j) \in G(N, K - K_0, g)\}$. Thus, $E_{i,j}$ contains all uncolored parallel edges between nodes v'_i and v''_j. Clearly, each uncolored edge in $G(N, K - K_0, g)$ is in exactly one of such $E_{i,j}$s.

Our second algorithm consists of $2g$ iterations. In each iteration, we try to color a set of nonparallel uncolored edges using one of colors in a set of $2g$ colors, $\{0, 1, \cdots, 2g - 1\}$, so that no two edges with the same color are adjacent to the same vertex. Then, for each edge e with color $2g - 1$, we recolor it by a free color in $\{0, 1, \cdots, 2g - 2\}$. The following is the outline of the algorithm:

```
for $l = 0$ to $2g - 1$ do
  for all $i, j \in \{1, 2, \cdots, N/g\}$ do
    $c_{i,j} := (i + j + l) \bmod 2g$;
    if there is an uncolored edge in $E_{i,j}$ and color $c_{i,j}$ is free at
    both $v'_i$ and $v''_j$ then
      assign color $c_{i,j}$ to this edge;
      update free colors at $v'_i$ and $v''_j$ and remove the colored
      edge from $E_{i,j}$;
    end if
  end for
end for
for all edges with color $2g - 1$ do
  color these edges with one of free colors in
  $\{0, 1, \cdots, 2g - 2\}$;
end for
```

The correctness of this algorithm can be derived from the following five simple facts:

1. *In iteration i, one uncolored edge, if any, in each $E_{i,j}$ is selected.* This is obvious. Note that such a selected edge may not be colored in the iteration.
2. *In iteration i, if two edges, one in $E_{i,j}$ and one in $E_{p,q}$, are assigned the same color, i.e., $c_{i,j} = c_{p,q}$, then $i \neq p$ and $j \neq q$.* Fact 2 can be proven by contradiction as follows: Assume that there are two pairs of (i, j) and (i, q) with $j \neq q$ and $c_{i,j} = c_{i,q}$. (For the case that there are two pairs of (i, j) and (p, j) with $i \neq p$ and $c_{i,j} = c_{p,j}$, the proof is similar.) Then, by the algorithm, $i + j + l \bmod 2g = i + q + l \bmod 2g$, which implies that $|j - q| = 2g \times y$, where y is a nonnegative iteger. Since $j, q \in \{1, 2, \cdots, N/g\}$ and $g = 2^{\lfloor \frac{n+\alpha}{2} \rfloor}$, we have $|j - q| < 2g$. Thus, $y = 0$ and $j = q$, which contradicts the assumption.
3. *For each uncolored edge, all $2g$ possible colors are tried before it is assigned a color in the worst case.* By the algorithm, this is obviously true.
4. *After $2g$ iterations, no two adjacent edges are assigned the same color.* By Fact 2, this is obviously true for any two nonparallel edges. For any two (parallel) edges in $E_{i,j}$, they are assigned different colors because of Fact 3 and the fact that their colors are computed using different l values in different iterations.
5. *The edges with the same color $2g$ can be recolored concurrently using the colors in $\{0, 1, \cdots, 2g - 2\}$ so that none of adjacent edges is assigned the same color.* By Fact 4 and Lemma 9, each edge with color $2g$ can be reassigned a color in $\{0, 1, \cdots, 2g - 2\}$ without resulting in any color conflict.

Now, we show that this algorithm can be implemented in $O(g) = O(\sqrt{N})$ time using a completely connected multiprocessor system of N PEs. This is equivalent to showing that each of the $2g$ iterations takes $O(1)$ time. We associate a $2g$-bit binary array $C_v[0..2g - 1]$ with each vertex v of $G(N, K, g)$ such that $C_v[c] = 1$ if and only if color c is available at vertex v, and assign $N/(2g)$ PEs to v. Then, the operations of finding if a given color c is available at v and updating $C_v[c]$ can be carried out in $O(1)$ time. We only need to make sure that the operation of finding an uncolored edge in $E_{i,j}$, $1 \le i, j \le N/g$, (if any) in each iteration can be done in $O(1)$ time. This can be achieved by a preprocessing step of sorting. For each vertex v'_i, we can sort all edges in each $E_i = \cup_{j=1}^{N/g} E_{i,j}$,

$1 \le i \le N/g$, of $G(N, K - K_0, g)$, using g PEs with $O(1)$ edges per PE, in nondecreasing order of j in $O(\lg^2 g)$ time. Then, we assign a set of $N/(2g) = O(g)$ PEs to each vertex of $G(N, K, g)$ in such a way that each $E_{i,j}$ is allocated $O(1)$ PE, which is used to find an uncolored edge in $E_{i,j}$. Based on the sorted edges, a PE associated with $E_{i,j}$ can find the starting locations of its assigned edges in $O(g)$ time. After this preprocessing, the operation of finding uncolored edges in each iteration can be done in $O(1)$ time. Finally, recoloring edges with color $2g$ can be done in $O(\lg g)$ time, since this operation is similar to one iteration of Step 2 of our first algorithm presented in the previous section. In summary, we have the following result.

Theorem 6. *For any I/O mapping graph $G(N, K, g)$ with $K_0 (< K)$ colored edges, a strong $(2g - 1)$-edge coloring can be found in $O(g)$ time using a completely connected multiprocessor system of N PEs.*

5.4 Performance Analysis

We summarize the overall performance of our routing algorithm for SNB network $B(N, 0, p^*, \alpha)$ by the following theorem.

Theorem 7. *For an SNB network $B(N, 0, p^*, \alpha)$ with $p^* = 2^{\lfloor \frac{n+\alpha}{2} \rfloor + 1} - 1$, connections from any $K - K_0$ idle inputs to any $K - K_0$ idle outputs, with K_0 existing connections, can be correctly routed in $O(\min\{d^* \lg N, \sqrt{N}\})$ time using a completely connected multiprocessor system of N PEs, where d^* is the degree of $G(N, K - K_0, g)$.*

Proof. By Theorems 5 and 6, we can find a strong $(2g - 1)$-edge coloring of $G(N, K, g)$ in $O(\lg d^* \lg(K - K_0) + d^* \lg g) = O(d^* \lg N)$ time and $O(g)$ time using our first and second algorithms, respectively. Using an algorithm for finding the maximum, d^* can be computed in $O(\lg N)$ time. If $d^* \le \frac{\sqrt{N}}{\lg N}$, we apply our first algorithm; otherwise, we apply our second algorithm. We assign each new connection with color i to the ith plane of $B(N, 0, p^*, \alpha)$. By Lemmas 1 and 4, these new connections can be routed by self-routing in $O(\lg N)$ time. Thus, the total time is $O(\min\{d^* \lg N, \sqrt{N}\})$. □

The two algorithms for strong $(2g - 1)$-edge coloring of $G(N, K, g)$ have time bounds $O(\lg d^* \lg(K - K_0) + d^* \lg g)$ and $O(\sqrt{N})$, where d^* is the degree of $G(N, K - K_0, g)$. In the worst case, $O(\lg d^* \lg(K - K_0) + d^* \lg g) = O(\sqrt{N} \lg N)$ and the first algorithm is slower than the second. But, when d^* is small, the first algorithm can be much faster.

By Lemma 8, we can derive the minimum number of planes, $p_{\min}$, for $B(N, 0, p, \alpha)$ to be SNB as follows: If there is no crosstalk-free constraint (i.e., $\alpha = 0$), then $p_{\min} = \frac{3}{2} 2^{\frac{n}{2}} - 1$ for even n and $p_{\min} = 2^{\frac{n+1}{2}} - 1$ for odd n. If there is a crosstalk-free constraint (i.e., $\alpha = 1$), then $p_{\min} = 2^{\frac{n}{2}+1} - 1$ for even n and $p_{\min} = \frac{3}{2} 2^{\frac{n+1}{2}} - 1$ for odd n. Compared with $B(N, 0, p_{\min}, \alpha)$, the hardware redundancy $p_{red} = p^* - p_{\min}$ of $B(N, 0, p^*, \alpha)$ is: $p_{red} = 0$ if $\alpha = 0$ and n is odd or $\alpha = 1$ and n is even, $p_{red} = \sqrt{N}/2$ if $\alpha = 0$ and n is even, and $p_{red} = \sqrt{2N}/2$ if $\alpha = 1$ and n is odd. The hardware cost of $B(N, 0, p^*, \alpha)$, in terms of the number of SEs, is higher than that of $B(N, 0, p_{\min}, \alpha)$ in half of the cases, but both have the same hardware complexity of $\Theta(N^{1.5} \lg N)$. The time for routing $O(N)$ connections, however, is improved from $\Omega(N \lg N)$ to sublinear $O(\sqrt{N})$ in the worst case.

6 Conclusion

The major contribution of this paper is the design and analysis of parallel routing algorithms for a class of nonblocking switching networks, $B(N, x, p, \alpha)$. Although the assumed parallel machine model is a completely connected multiprocessor system of N PEs, the proposed algorithms can be transformed to algorithms for more realistic parallel computing models. The pointer jumping technique and any one-to-one permutation communication step used in our proposed algorithms can be implemented by sorting on realistic parallel computing structures. Let $S(N)$ be the time for sorting N elements on a parallel machine M with N processors, then our algorithms can be implemented with a slow-down factor $S(N)$ on M. It is known that sorting N numbers on the class of hypercubic networks takes $O(\lg N \lg\lg N)$ time [8], [18]. This class of networks include hypercube, cube-connected-cycles, butterfly networks, baseline networks, reverse baseline networks, Omega networks, flip networks, de Bruijin graphs, shuffle-exchange networks, banyan networks, delta networks, bidelta networks, k-ary butterflies, and Benes networks [18]. Our algorithms can route connections in $B(N, x, p, \alpha)$ with a slow-down factor $O(\lg N \lg\lg N)$ on all these realistic parallel machine models, though some have topologies that are quite different from others, whose structural complexities are no larger than that of one plane in $B(N, x, p, \alpha)$. Compared with sequential algorithms, we consider that our algorithms on realistic parallel computers provide a significant speedup, making them potentially valid and useful for large switches.

The approach of applying edge-coloring techniques to investigate the capacity and routability of RNB switching networks has been widely used (refer to [6], [13], [19], [22]). We extended this approach to SNB networks by defining strong edge-coloring. For a class of RNB and SNB banyan-based switching networks obtained by horizontal expansion and vertical replication, we proposed a unified mathematical formulation, namely, WEC and SEC problems, for designing parallel routing algorithms using this approach. Our algorithms can find the solutions for WEC problem in polylogarithmic time and SEC problem in sublinear time. Finding faster parallel algorithms for WEC and SEC problems, especially for the SEC problem, however, remains to be very challenging.

The results of this paper have valuable architectural implications for the design and implementation of future large-scale electronic and optical switching networks. Scalable nonblocking switching networks tend to have no self-routing capability. For example, for a nonblocking switching network $B(N, x, p, \alpha)$, though self-routing capabilities exist in a portion of it, its routing is still computation intensive. Therefore, for the design of a switching network, in addition to its hardware cost in terms of the cost of SEs and interconnection links (and wavelengths), we must take the routing complexity into consideration. It remains a great challenge for finding low-cost high-speed nonblocking switching networks.

REFERENCES

[1] D.P. Agrawal, "Graph Theoretical Analysis and Design of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 32, no. 7, pp. 637-648, July 1983.

[2] V.E. Benes, "Permutation Groups, Complexes, and Rearrangeable Connecting Networks," *The Bell System Technical J.,* vol. 43, pp. 1619-1640, July 1964.

[3] V.E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic.* New York: Academic Press, 1965.

[4] J.A. Bondy and U.S. R. Murty, *Graph Theory with Applications.* Elsevier North-Holland, 1976.

[5] J. Carpinelli and A.Y. Oruc, "A Non-Blocking Matrix Decomposition Algorithm for Routing on Clos Networks," *IEEE Trans. Comm.,* vol. 39, pp. 1245-1251, 1993.

[6] J. Carpinelli and A.Y. Oruc, "Applications of Matching and Edge-Coloring Algorithms to Routing in Clos Networks," *Networks,* vol. 24, pp. 319-326, Sept. 1994.

[7] C.J. Chen and A.A. Frank, "On Programmable Parallel Data Routing Networks via Crossbar Switches for Multiple Element Computer Architectures," *Parallel Processing,* G. Goos and J. Harmanis, eds., New York: Springer-Verlag, 1975.

[8] R. Cypher and G. Plaxton, "Deterministic Sorting in Nearly Logarithmic Time on the Hypercube and Related Computers," *Proc. 22nd Ann. ACM Symp. Theory of Computing,* pp. 193-203, 1990.

[9] R. Cole and J. Hopcroft, "On Edge Coloring Bipartite Graphs," *SIAM J. Computing,* vol. 11, no. 1, pp. 540-546, 1982.

[10] O. Kariv and H. Gabow, "Algorithms for Edge Coloring Bipartite Graphs and Multigraphs," *SIAM J. Computing,* vol. 11, no. 1, pp. 117-129, 1982.

[11] H. Hinton, "A Non-Blocking Optical Interconnection Network Using Directional Couplers," *Proc. IEEE Global Telecomm. Conf.,* pp. 885-889, Nov. 1984.

[12] D.K. Hunter, P.J. Legg, and I. Andonovic, "Architecture for Large Dilated Optical TDM Switching Networks," *IEE Proc. Optoelectronics,* vol. 140, no. 5, pp. 337-343, Oct. 1993.

[13] F.K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks.* World Scientific, 1998.

[14] J. Jaja, *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.

[15] C.T. Lea, "Multi-log2N Networks and Their Applications in High-Speed Electronic and Photonic Switching Systems," *IEEE Trans. Comm.,* vol. 38, no. 10, pp. 1740-1749, Oct. 1990.

[16] C.T. Lea and D.J. Shyy, "Tradeoff of Horizontal Decomposition versus Vertical Stacking in Rearrangeable Nonblocking Networks," *IEEE Trans. Comm.,* pp. 899-904, vol. 39, no. 6, June 1991.

[17] H.Y. Lee, F.K. Hwang, and J. Carpinelli, "A New Decomposition Algorithm for Rearrangeable Clos Interconnection Networks," *IEEE Trans. Comm.,* vol. 44, pp. 1572-1578, 1997.

[18] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes.* Morgan Kaufmann Publishers, 1992.

[19] G.F. Lev, N. Pippenger, and L.G. Valiant, "A Fast Parallel Algorithm for Routing in Permutation Networks," *IEEE Trans. Computers,* vol. 30, no. 2, pp. 93-100, Feb. 1981.

[20] G. Maier, A. Pattavina, and S.G. Colombo, "Control of Non-Filterable Crosstalk in Optical-Cross-Connect Banyan Architectures," *Proc. IEEE Global Telecomm. Conf. GLOBECOM,* vol. 2, pp. 1228-1232, Nov.-Dec. 2000.

[21] G. Maier and A. Pattavina, "Design of Photonic Rearrangeable Networks with Zero First-Order Switching-Element-Crosstalk," *IEEE Trans. Comm.,* vol. 49, no. 7, pp. 1268-1279, July 2001.

[22] N. Nassimi and S. Sahni, "Parallel Algorithms to Set Up the Benes Permutation Network," *IEEE Trans. Computers,* vol. 31, no. 2, pp. 148-154, Feb. 1982.

[23] V.I. Neiman, "Structure et Command Optimals de Reseaux de Connxion Sans Blocage," *Annales des Telecomm.,* vol. 24, pp. 232-238, 1969.

[24] K. Padmanabhan and A. Netravali, "Dilated Network for Photonic Switching," *IEEE Trans. Comm.* , vol. 35, no. 12, pp. 1357-1365, Dec. 1987.

[25] D.C. Opferman and N.T. Tsao-Wu, "On a Class of Rearrangeable Switching Networks," *Bell System Technical J.,* vol. 50, no. 5, pp. 1579-1600, 1971.

[26] Y. Pan, C. Qiao, and Y. Yang, "Optical Multistage Interconnection Networks: New Challenges and Approaches," *IEEE Comm. Magazine,* vol. 37, no. 2, pp. 50-56, Feb. 1999.

[27] R. Ramaswami and K. Sivarajan, *Optical Networks: A Practical Perspective,* second ed. Morgan Kaufmann, 2001.

[28] G.H. Song and M. Goodman, "Asymmetrically-Dilated Cross-Connect Switches for Low-Crosstalk WDM Optical Networks," *Proc. IEEE Eighth Ann. Meeting Conf. Lasers and Electro-Optics Soc. Ann. Meeting,* vol. 1, pp. 212-213, Oct. 1995.

[29] F.M. Suliman, A.B. Mohammad, and K. Seman, "A Space Dilated Lightwave Network—A New Approach," *Proc. IEEE 10th Int'l Conf. Telecomm. (ICT 2003),* vol. 2, pp. 1675-1679, 2003.

[30] M. Vaez and C.T. Lea, "Wide-Sense Nonblocking Banyan-Type Switching Systems Based on Directional Couplers," *IEEE J. Selected Areas in Comm.,* vol. 16, no. 7, pp. 1327-1332, Sept. 1998.

[31] M. Vaez and C.T. Lea, "Strictly Nonblocking Directional-Coupler-Based Switching Networks under Crosstalk Constraint," *IEEE Trans. Comm.,* vol. 48, no. 2, pp. 316-323, Feb. 2000.

[32] V. Vizing, "On an Estimate of the Chromatic Class of a p-Graph," *Metody Diskret. Analiz,* pp. 25-30, 1964.

[33] J.E. Watson et al., "A Low-Voltage $8 \times 8\ \mathrm{Ti} : \mathrm{LiNbO_3}$ Switch with a Dilated Benes Architecture," *IEEE J. Lightwave Technology,* vol. 8, pp. 794-800, May 1990.

[34] C.L. Wu and T.Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 29, no. 8, pp. 694-702, Aug. 1980.

Enyue Lu received the PhD degree in computer science from the University of Texas at Dallas in 2004. Currently, she is an assistant professor in the Mathematics and Computer Science Department at Salisbury University, Maryland. Dr. Lu's main research interests include parallel processing and computing, computer and communication networks, algorithm design and analysis, computer architectures, and combinatorics and graph theory. She earned a Best Paper Award at the 14th IASTED International Conference on Parallel and Distributed Computing and Systems in 2002. She is a member of the IEEE.

S.Q. Zheng received the PhD degree from the University of California, Santa Barbara, in 1987. After being on the faculty of Louisiana State University for 11 years, he joined the University of Texas at Dallas in 1998, where he is currently a professor of computer science, computer engineering, and telecommunications engineering. Dr. Zheng's research interests include algorithms, computer architectures, networks, parallel and distributed processing, telecommunications, and VLSI design. He has published approximately 200 papers in these areas. He served as the program committee chairman of numerous international conferences and the editor of several professional journals. He is a senior member of the IEEE.