Parallel Processing Letters Vol. 7 No. 3 (1997) 219–224

DETERMINISTIC ROUTING ON THE ARRAY WITH RECONFIGURABLE OPTICAL BUSES

SANGUTHEVAR RAJASEKARAN* and SARTAJ SAHNI†
Department of CISE, University of Florida,
Gainesville, FL 32611, U.S.A.

Received October 1996
Revised March 1997
Accepted by R. Miller

ABSTRACT

In this paper we present efficient deterministic algorithms for various classes of routing problems on the array with reconfigurable optical buses (AROB).

Keywords: Reconfigurable Networks, Packet Routing

1. Introduction

In any fixed connection network, a single step of interprocessor communication can be thought of as a packet routing task. The problem of routing can be stated as follows: There is a packet of information at each node that is destined for some other node. Send all the packets to their correct destinations as quickly as possible making sure that at most one packet crosses any edge at any time. Packet routing is equivalent to the random access write operation first defined by Nassimi and Sahni [6]. The *run time* of any packet routing algorithm is defined to be the time taken by the last packet to reach its destination. The *queue size* is the maximum number of packets that any processor will have to store during the algorithm.

The problem of *partial permutation routing* is the task of routing where at most one packet originates from any node and at most one packet is destined for any node. Any routing problem where at most h packets originate from any node and at most h packets are destined for any node will be called $h-h$ *routing* or h*-relations* [10]. In this paper we study routing problems on the AROB.

An AROB [8, 9] is essentially an $m \times n$ reconfigurable mesh in which the buses are implemented using optical technology. This model has attracted the attention

*Research of this author was supported, in part, by an NSF Grant CCR-9596065
†Research of this author was supported, in part, by the Army Research Office under grant DAA H04-95-1-0111

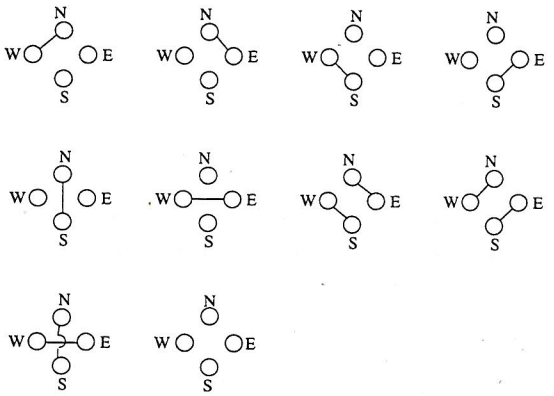

Figure 1: Possible Switch Connections

of many researchers in the recent past owing to its promise in superior practical performance. The allowable switch settings of the processors are the same as those in the RN model of [2]. These are shown in Figure . A bus link connects two adjacent processors x and y and has two associated wave guides. One of the wave guides permits an optical signal to travel from x to y and the other permits signal movement from y to x. By setting processor switches, bus links are connected together to form disjoint buses. On each bus, we need to specify which orientation of the waveguide on each link of the bus is to be used. The time needed to transmit a message on a bus is referred to as one cycle. A cycle is divided into slots of duration τ and each slot can carry a different optical signal. τ is the time needed for an optical pulse to move down one bus link. Pavel and Akl [8] have argued that for reasonable size meshes (say up to 1000×1000), the number of slots in a cycle may be assumed to be n for an $n \times n$ mesh. Further, the duration of a cycle may be assumed constant and comparable to the time for a CPU operation. A linear AROB (LAROB) is a $1 \times n$ AROB [8]. More details of the model can be found in [8, 9].

1.1. Known Results

In [8], the AROB model has been defined. Similar models have been employed before as well (see e.g., [7]). A related model known as the *Optical Communication Parallel Computer (OCPC)* has also been defined in the literature (see e.g., [1], [3], [10], [4]).

In an OCPC any processor can communicate with any other processor in one unit of time, provided there are no conflicts. If more than one processors try to send a message to the same processor, no message reaches the intended destination. [7] considers the problem of selection on a mesh with optical buses.

In [8], algorithms for such problems as prefix computation, routing on a linear array, matrix multiplication, etc. have been given for the AROB. On the other hand, [7] considers the problem of selection on a mesh with optical buses.

The proof of the following Lemma can be found in [9]:

Lemma 1 *Let $\mathcal{L}$ be a LAROB of size n. Consider a routing problem where $O(1)$ packets originate from any node and $O(1)$ packets are destined for any node. This problem can also be solved in $O(1)$ cycles.*

The following result is due to [8]:

Lemma 2 *Say there are k elements arbitrarily distributed (at most one per processor) in a two dimensional AROB of size $\sqrt{n} \times \sqrt{n}$. We would like to 'compact' them in the first $\lceil \frac{k}{\sqrt{n}} \rceil$ rows. This problem can be solved in $O(1)$ cycles.*

Several packet routing algorithms for the OCPC model can be found in the literature. Anderson and Miller have shown that a special case of $\log n$-relations on an n-node OCPC can be routed in $\widetilde{O}(\log n)$ time [1]. Also, [10] and [3] have presented efficient algorithms for h-relations. An algorithm for arbitrary h-relations with a run time of $\widetilde{O}(h + \log\log n)$ has been given by Goldberg, Jerrum, Leighton, and Rao [4]. In [9], Rajasekaran and Sahni show that any h-relation can be routed in $\widetilde{O}(h)$ time on a LAROB. In contrast, the best known algorithm for the OCPC model has a run time of $\widetilde{O}(h + \log\log n)$ [4]. They also present an algorithm for h-relation routing on a $\sqrt{n} \times \sqrt{n}$ AROB with a run time of $\widetilde{O}(h + \log\log n)$. The best known algorithm for the same problem on the OCPC model has a run time of $\widetilde{O}(h + \frac{\log n}{\log\log n})$ [3]. Their deterministic algorithm for h-relations runs in time $O(h \log n)$ on a $\sqrt{n} \times \sqrt{n}$ AROB as well as on an n-node LAROB.

Definition [BPC Permutations][5]: In a network $\mathcal{N}$ with N nodes, $N!$ permutations are possible. An important subset of these permutations is called *Bit Permute and Complement (BPC)*. Assume that N is an integral power of 2 (i.e., $N = 2^p$ for some integer p). Each node of $\mathcal{N}$ can be labeled with a p-bit sequence $a_{p-1}, a_{p-2}, \cdots, a_1, a_0$. Any BPC permutation, π can be described with a vector $(\pi_{p-1}, \pi_{p-2}, \cdots, \pi_1, \pi_0)$, where $\pi_i \in \{\pm 0, \pm 1, \cdots, \pm(p-1)\}$ and $(|\pi_{p-1}|, |\pi_{p-2}|, \cdots, |\pi_1|, |\pi_0|)$ is a permutation of $(0, 1, 2, \cdots, p-1)$. Under this permutation, if the origin of a packet is $a_{p-1}, a_{p-2}, \cdots, a_1, a_0$ then its destination will be $d_{p-1}, d_{p-2}, \cdots, d_1, d_0$, where $d_{|\pi_i|} = a_i$ if π_i is non negative and $d_{|\pi_i|} = \bar{a_i}$ otherwise.

Consider a network $\mathcal{N}$ with 8 nodes, for example. Let the permutation under concern be $\pi = (-2, 0, 1)$. Under this permutation a packet originating from the node (a_2, a_1, a_0) will be destined for $(\bar{a_2}, a_0, a_1)$. For instance a packet from node $(1, 1, 0)$ will have $(0, 0, 1)$ as its destination.

Many important permutations such as matrix transpose, bit reversal, prefect shuffle, etc. belong to the class of BPC permutations. For example the perfect shuffle can be characterized with the vector $(0, p-1, p-2, \cdots, 2, 1)$. The vector $(p/2-1, p/2-2, \cdots, 1, 0, p-1, p-2, \cdots, p/2)$ corresponds to matrix transpose. Bit reversal is described by $(0, 1, 2, \cdots, p-1)$, and so on. The number of permutations in the BPC class is $2^p p!$ where the network size is $N = 2^p$.

2. Matrix Transpose Routing

In this section we show how to perform matrix transpose routing in $O(1)$ cycles

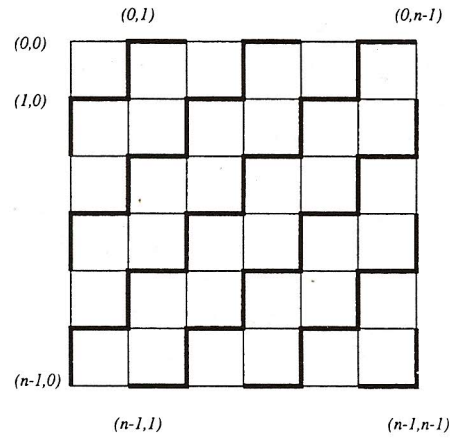

Figure 2: Matrix transpose

on an $n \times n$ AROB. Recall that the matrix transpose permutation is a BPC permutation characterized by the vector $(p/2-1, p/2-2, \cdots, 1, 0, p, p-1, \cdots, p/2)$. If a packet originates at the node (i,j) $(1 \leq i,j \leq n)$ in the 2D mesh, then its destination is (j,i).

To perform this routing in $O(1)$ cycles, we connect the switches of nodes as shown in Figure . The buses formed are along the diagonals. The switch connection for any processor (i,j) is SE if $i+j$ is odd and it is NW if $i+j$ is even. Note that every node of the mesh is on exactly one bus. Also, if the node (i,j) is on some bus, then (j,i) will also be on the same bus. Perform routing along each bus thus formed. All the packets will reach their correct destinations at the end of this routing.

Since routing along a LAROB can be performed in $O(1)$ cycles (c.f. Lemma 1), matrix transpose can also be completed in $O(1)$ cycles. Thus we get the following Lemma:

Lemma 3 *Matrix transpose can be performed on an $n \times n$ AROB in $O(1)$ cycles.*
□

The above Lemma will prove helpful in devising routing algorithms for general BPC permutations.

3. BPC Permutation Routing

In this section we present our $O(1)$ cycles algorithm for general BPC permutations. The basic idea is to decompose any BPC permutation into a sequence of five permutations. The first and the fourth permutations in the decomposition are such that they correspond to data movements only along the columns. The second and the fifth permutations are along the rows. Using Lemma 1, these four permutations can be performed in $O(1)$ cycles. On the other hand, the third permutation is a matrix transpose which can also be realized in $O(1)$ cycles (c.f. Lemma 3). More details follow. Assume w.l.o.g. that p is even.

Let $\pi = (\pi_{p-1}, \pi_{p-2}, \cdots, \pi_{p/2}, \pi_{p/2-1}, \cdots, \pi_1, \pi_0)$ be the BPC permutation

to be routed. Denote the first (second) half of the vector as P_1 (resp. P_0). In other words, $P_1 = \pi_{p-1}, \pi_{p-2}, \cdots, \pi_{p/2}$ and $P_0 = \pi_{p/2-1}, \cdots, \pi_1, \pi_0$. Let k be the number of symbols in the sequence P_1 that belong to $\{0, 1, 2, \cdots, p/2-1\}$ and let $\pi_{i_1}, \pi_{i_2}, \cdots, \pi_{i_k}$ be these symbols. Clearly, the number of symbols in P_0 that belong to $\{p/2, p/2+1, \cdots, p-1\}$ will also be k. Let these symbols be $\pi_{j_1}, \pi_{j_2}, \cdots, \pi_{j_k}$.

Let $A = (a_{p-1}, a_{p-2}, \cdots, a_1, a_0)$ be any node and q be the packet originating from A. The five phases in the algorithm are described below:

- **Phase I.** Let $A = A_1, A_0$ where A_1 is the first half of A and A_0 is the second half of A. Also let $A'_1 = a_{i_1}, a_{i_2}, \cdots, a_{i_k}$ and $A'_0 = a_{j_1}, a_{j_2}, \cdots, a_{j_k}$. Route the packet from A to $A' = A_1 - A'_1, A'_1, A_0$. Here $A_1 - A'_1$ is the subsequence that results from A_1 after eliminating symbols of A'_1. This task involves routing along the columns. As a part of this phase take care of complements in A_1, if any. I.e., if $a_i \in A_1$ and π_i is negative, then a_i will appear in A' as $\bar{a_i}$ (in an appropriate place).

- **Phase II.** Now route the packet q from its current location to $A_1 - A'_1, A'_1, A_0 - A'_0, A'_0$. This task involves routing along rows. Any complements in A_0 can also be taken care of.

- **Phase III.** Partition the mesh into submeshes of size $2^k \times 2^k$ and employ matrix transpose within the submeshes. As a result q will reach the node $A_1 - A'_1, A'_0, A_0 - A'_0, A'_1$.

- **Phase IV.** This phase is analogous to phase I and here routing is done along the columns. At the end, q will be in a node the first half of whose label is the same as the first half of the final destination of q. In other words, q will be in its destination row.

- **Phase V.** Finally, a routing step along the rows takes q to its desired destination.

Example. Let $\mathcal{N}$ be a network with $N = 2^8$ nodes. Let the BPC permutation under concern be $(6, -3, -4, 1, 0, -2, 5, 7)$. If the packet q originates from $(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$, $A_1 = a_7, a_6, a_5, a_4$; $A_0 = a_3, a_2, a_1, a_0$; $A'_1 = a_6, a_4$; and $A'_0 = a_1, a_0$.

In the first phase, q goes to $a_7, \bar{a_5}, \bar{a_6}, a_4, a_3, a_2, a_1, a_0$. In the second phase it goes to $a_7, \bar{a_5}, \bar{a_6}, a_4, a_3, \bar{a_2}, a_1, a_0$. After the matrix transpose in phase III, q will reach $a_7, \bar{a_5}, a_1, a_0, a_3, \bar{a_2}, \bar{a_6}, a_4$.

Phase IV takes q to $a_0, a_7, a_1, \bar{a_5}, a_3, \bar{a_2}, \bar{a_6}, a_4$. And finally at the end of phase V, q will reach $a_0, a_7, a_1, \bar{a_5}, \bar{a_6}, \bar{a_2}, a_4, a_3$. □

Phases I, II, IV, and V take $O(1)$ cycles each (c.f. Lemma 1). Phase III takes $O(1)$ cycles as well in accordance with Lemma 3. Thus the whole algorithm takes $O(1)$ cycles. Note that the routing we perform in each phase is a permutation, i.e., there are no conflicts among the packets. We get the following theorem:

Theorem 1 *Any BPC permutation can be completed in $O(1)$ cycles on a 2D AROB.*
□

References

1. R.J. Anderson and G.L. Miller, Optical Communication for Pointer Based Algorithms, *Technical Report CRI-88-14*, Computer Science Department, University of Southern California, 1988.
2. Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, The Power of Reconfiguration, *Journal of Parallel and Distributed Computing*, 1991, pp. 139-153.
3. M. Geréb-Graus and T. Tsantilas, Efficient Optical Communication in Parallel Computers, in *Proc. Symposium on Parallel Algorithms and Architectures*, 1992, pp. 41-48.
4. L. Goldberg, M. Jerrum, T. Leighton, and S. Rao, A Doubly-Logarithmic Communication Algorithm for the Completely Connected Optical Communication Parallel Computer, in *Proc. Symposium on Parallel Algorithms and Architectures*, 1993.
5. D. Nassimi and S. Sahni, An Optimal Routing Algorithm for Mesh-Connected Parallel Computers, *Journal of the ACM*, 27(1), 1980, pp. 6-29.
6. D. Nassimi and S. Sahni, A Self-Routing Benes Network and Parallel Permutation Algorithms, *IEEE Transactions on Computers*, C-30(5), 1981, pp. 332-340.
7. Y. Pan, Order Statistics on Optically Interconnected Multiprocessor Systems, in *Proc. First International Workshop on Massively Parallel Processing Using Optical Interconnections*, 1994, pp. 162-169.
8. S. Pavel and S.G. Akl, Matrix Operations using Arrays with Reconfigurable Optical Buses, manuscript, 1995.
9. S. Rajasekaran and S. Sahni, Sorting, Selection and Routing on the Array with Reconfigurable Optical Buses, to appear in *IEEE Transactions on Parallel and Distributed Systems.*
10. L.G. Valiant, General Purpose Parallel Architectures, in *Handbook of Theoretical Computer Science: Vol. A* (J. van Leeuwen, ed.), North Holland, 1990.