Sublogarithmic Deterministic Selection on Arrays with a Reconfigurable Optical Bus

Yijie Han, Yi Pan, *Senior Member*, *IEEE*, and Hong Shen

Abstract—The Linear Array with a Reconfigurable Pipelined Bus System (LARPBS) is a newly introduced parallel computational model, where processors are connected by a reconfigurable optical bus. In this paper, we show that the selection problem can be solved on the LARPBS model deterministically in $O((\log\log N)^2/\log\log\log N)$ time. To our best knowledge, this is the best deterministic selection algorithm on any model with a reconfigurable optical bus.

Index Terms—Analysis of algorithms, massive parallelism, optical bus, parallel algorithms, selection.

1 Introduction

Fiber optic communication offers the advantage of high bandwidth, low error probability, and gigabit transmission capacity. Therefore, its use in interconnecting multiprocessors of a parallel computer has been proposed by many researchers [4], [15], [16], [31]. Among them, the distributed-memory SIMD (Single Instruction Multiple Data) computer with pipelined optical buses has received much attention [6], [10], [16], [17], [18], [27], [28] due to its simplicity and low cost. On such systems, messages are transmitted concurrently on the optical bus which connects all processors. One cycle of such a parallel system is merely the delay of the light between the furthest processors over a waveguided bus. This design integrates the advantages of both optical transmission and electronic computation. Several slightly different versions of such a system have been proposed in the literature, including the *array processors with pipelined buses* (APPB) [7], [16], the *array processors with pipelined buses using switches* (APPBS) [8], the *array with synchronous optical switches* (ASOS) [28], the *reconfigurable array with spanning optical buses* (ROSOB) [29].

Based on the research in reconfigurable meshes and pipelined optical buses, the model of arrays with reconfigurable optical buses has been proposed independently by Pavel and Akl [23], Pavel [24], and by Pan and Hamdi [19], and Pan and Li [20]. The model proposed by Pavel and Akl [23] and Pavel [24] is the *arrays with reconfigurable optical buses* (AROB), and the model proposed by Pan and Hamdi [19] and Pan and Li [20] is the *linear arrays with a reconfigurable pipelined bus system* (LARPBS). Many algorithms have been designed on these models, including inversion number computation [9], neural network computation [3], various matrix operations [11], [12], [25], selection [18], [30], sorting [19], [20], [26], [30]. The two models differ in some aspects. For example, counting is not allowed during a bus cycle on the LARPBS model, while it is permitted on the AROB model.

In this paper, we study the selection problem on the LARPBS model. The selection problem, when given s, is to select the sth smallest data item among the N ordered (but not sorted) data items. A straightforward way of solving the selection problem is to sort the input data items and then pick the sth smallest item. This approach, however, is usually not efficient due the relatively high cost of performing the sorting. Currently, sorting requires $O(\log N)$ expected time or $O(\log^2 N)$ deterministic time for sorting N data items on N processor LARPBS [19], [21]. Solving the selection problem without sorting first has been considered by researchers. Pan has studied the selection problem on the LARPBS model and gave a selection algorithm which runs in $O(N\log N/p)$ expected time using p processors [18]. Li et al. described a deterministic algorithm which selects in $O(\log N)$ deterministic time for N data items on N-processor LARPBS. Selection problem has also been considered on other related model. For example, Rajasekaran and Sahni solved the selection problem on the 2D AROB model. Their randomized algorithm runs in $O(1)$ time with high probability [30].

We note that the selection problem on the PRAM model has been studied by Cole [2] and Chaudhuri et al. [1]. The original idea is from Cole's paper [2]. Chaudhuri et al. improved Cole's algorithm on the CRCW model by using approximate counting. Currently, the PRAM algorithms on the EREW model [2] runs in $O(\log N\log^* N)$ time with optimal processor speedup. On the CRCW model, the algorithm by Chaudhuri et al. [1] runs in $O(\log N/\log\log N)$ time with optimal processor speedup. Selection algorithms have also been studied on hypercubes [32].

In this paper, we exploit the features of the LARPBS model. We also make use of the ideas in Cole's paper [2]. By using fast sorting algorithm to sort a smaller set of data on the LARPBS model, we are able to exhibit a selection

- *Y. Han is with Electronic Data Systems, Inc., 750 Tower Dr., CPS, Mail Stop 7121, Troy, MI 48098. E-mail: han@cstp.umkc.edu.*
- *Y. Pan is with the Department of Computer Science, Georgia State University, Atlanta, GA 30303. E-mail: pan@cs.gsu.edu.*
- *H. Shen is with the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan. E-mail: shen@jaist.ac.jp.*

Manuscript received 2 July 1999; revised, 30 Apr. 2001; accepted 22 May 2001.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 110175.

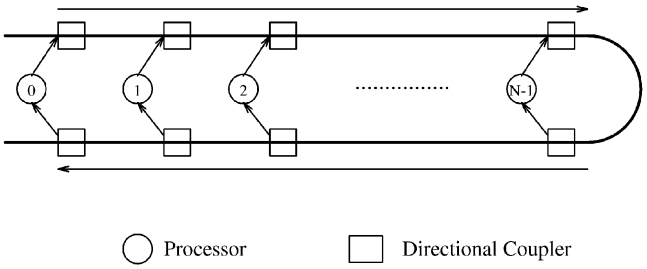


Fig. 1. A linear optical bus system of N processors.

algorithm for the LARPBS model which runs in $O((\log\log N)^2/\log\log\log N)$ time. Note that our algorithm is not simply a simulation of Cole's PRAM algorithm. Novel features and efficient basic data movement operations on the LARPBS model are exploited to speedup the algorithm, thus achieving a better time complexity than those on the PRAM. Many research groups are working in this area [9], [3], [11], [12], [25], [30]. To our best knowledge, this is the first deterministic selection algorithm with sublogarithmic time complexity on the LARPBS model.

2 The LARPBS Model

The LARPBS model uses an optical bus to connect its processors. A pipelined optical bus system uses optical waveguides instead of electrical buses to transfer messages among electronic processors. The advantages of using waveguides can be seen as follows: Besides the high propagation speed of light, there are two important properties of optical signal (pulse) transmission on an optical bus: unidirectional propagation and predictable propagation delay per unit length. These two properties enable synchronized concurrent access of an optical bus in a pipelined fashion [6], [16], [27], [28]. This, combined with the abilities of a bus structure to do efficient broadcasting or multicasting, makes the architecture suitable for many applications that involve intensive communication operations.

Fig. 1 shows a linear array in which electronic processors are connected with an optical bus. Each processor is connected to the bus with two-directional couplers, one for transmitting on the upper segment and the other for receiving from the lower segment of the bus [6], [16], [27], [28]. Messages are organized as fixed-length $messageframes$. Note that optical signals propagate unidirectionally from left to right on the upper segment and from right to left on the lower segment. This bus system is also referred to as the folded-bus connection in [6].

A linear array with a reconfigurable pipelined bus system (LARPBS) consists of N processors $P_1, P_2, \ldots, P_N$ connected by an optical bus. In addition to the tremendous communication capabilities, a LARPBS can also be partitioned into $k \geq 2$ independent subarrays

$$\mathrm{LARPBS}_1, \mathrm{LARPBS}_2, \ldots, \mathrm{LARPBS}_k,$$

such that LARPBS_j contains processors

$$P_{i_{j-1}+1}, P_{i_{j-1}+2}, \ldots, P_{i_j},$$

where $0 = i_0 < i_1 < i_2 \cdots < i_k = N$. The subarrays can operate as regular linear arrays with pipelined optical bus systems and all subarrays can be used independently for different computations without interference [19], [21]. Fig. 2 shows the LARPBS model with six processors. The array is split into two subarrays, with the first one having four processors and the second one having two processors. As in many other synchronous parallel computing systems, a LARPBS computation is a sequence of alternate global communication and local computation steps. The time complexity of an algorithm is measured in terms of the total number of bus cycles in all the communication steps, as long as the time of the local computation steps between successive communication steps is bounded by a constant and independent of the problem size. This complexity measure implies that a bus cycle takes constant time and this assumption has been adopted widely in the literature [3], [6], [5], [9], [11], [12], [13], [23], [24], [25], [26], [27], [28], [29], [30], [33]. (Remark: To avoid controversy, let us emphasize that in this paper, by "$O(f(p))$ time," we mean $O(f(p))$ bus cycles for global communication plus $O(f(p))$ number of local arithmetic/logic operations.)

3 Basic Operations

For ease of algorithm development and specification, a number of basic communication, data movement, and global operations on the LARPBS model implemented

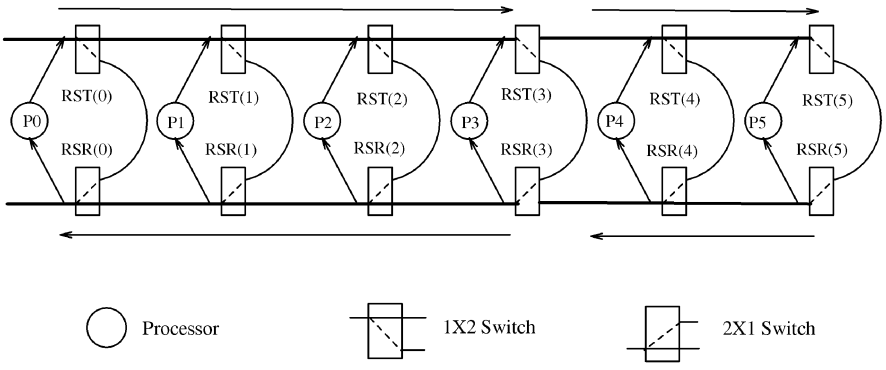


Fig. 2. The LARPBS model of size 6 with two subarrays.

using the coincident pulse processor addressing technique have been developed ([11], [19], [20], [21]). Each of these primitive operations can be performed in a constant number of bus cycles. These powerful primitives that support massive parallel communications, plus the reconfigurability of the LARPBS model, make the LARPBS very attractive in solving problems. Optical buses are not only communication channels among the processors, but also active components and agents of certain computations, e.g., binary prefix sum. The following primitive operations on LARPBS are used in this paper and our selection algorithm is developed using these operations as building blocks.

3.1 One-to-One Communication

In this operation, each processor sends one data item to another processor. This operation can be done easily in one bus cycle (see detailed discussion in [19], [21]).

3.2 Broadcast

In a broadcast operation, we have a source processor who sends a value in its local register to all the N processors. The details of this operation are described in [19], [21] and can be accomplished in one bus cycle.

3.3 Multicast

Multicast is a one-to-many communication operation. Each processor may send a message to a group of processors in the system. Each processor receives only one message from a source processor during a bus cycle. This is a special case of the h-relation, where $h = O(1)$, defined in [33], can be done in $O(1)$ bus cycles.

3.4 Compression

Assume an array of N data elements with each processor having one data element. Also, assume that the number of active data elements in the array is s. Active elements are labeled based upon certain values of their local variables. A processor with an active element is referred to as an active processor. The compression algorithm moves these active data elements to processors $N-s-1, N-s, \ldots, N-1$. In other words, the compression algorithm moves all active data items to the right side of the array. This operation can be done in $O(1)$ bus cycle on a LARPBS [20], [21].

3.5 Binary Prefix Sum

Consider a LARPBS with N processors and N binary values v_i, $0 \le i \le N-1$. The binary prefix sum requires the computation of

$$psum_i = v_0 + v_1 + \ldots + v_{i-1},$$

for all $0 \le i \le N-1$. It is shown that the binary prefix sum can be done in $O(1)$ bus cycle on a LARPBS [20], [21].

4 Fast Selection on the LARPBS Model

We apply the ideas in [2] to speed up our algorithm. In order to do so, we need a fast sorting algorithm. We first show that N data items can be sorted in constant time with N^2 processors on the LARPBS model and that M data items can be sorted in $O(\log\log N)$ time with Mk processors on the LARPBS model, where $M = k^{\sqrt{\log\log N}}$. These sorting algorithms will be used to speed up the selection process.

To sort N data items $a_0, a_1, \ldots, a_{N-1}$ with N^2 processors, we first broadcast a_i to processors iN through $(i+1)N-1$. Then, processor $iN+j$ compare a_i with a_j at processor $jN+i$. If $a_j < a_i$ or $a_j = a_i$ and $j < i$ processor, $iN+j$ will record a 1 otherwise it will record 0. Then, we compute the sum S_i of binary bits at processors iN through $(i+1)N-1$. S_i is the rank of a_i. After computing S_i, a_i is moved to the S_ith processor. This sorts the input data items and the time consumed is a constant. See more details of the algorithm in [22].

We use $f(M)$ to denote the time needed to sort M data items with Mk processors. To sort $M = k^{\sqrt{\log\log N}}$ data items with Mk processors, we first divide M data items into $\sqrt{k}$ sets with each set containing $M/\sqrt{k}$ data items. We sort each set in parallel and by recursion. This recursion takes $f(M/\sqrt{k})$ time. After we return from recursion, we have $\sqrt{k}$ sorted sets $S_0, S_1, \ldots, S_{\sqrt{k}-1}$, each containing $M/\sqrt{k}$ data items. Now, we show how to merge them into one sorted set. We use $g(a,b)$ to denote the time needed for merging b sorted sets each containing a data items with k processors allowed for each data item. Thus, our merging problem takes $g(M/\sqrt{k}, \sqrt{k})$ time. For each set, we pick every M/kth data item from the sorted data items. Thus, $\sqrt{k}$ data items are picked from each set. The total number of data items picked is k. We sort picked data items in constant time using the algorithm outlined in the previous paragraph. Let the k data items in sorted order be $b_0, b_1, \ldots, b_{k-1}$. These k picked data items, after being sorted, defines $k+1$ intervals. The jth interval is $> b_{j-1}$ but $\le b_j$ (b_{-1} is $-\infty$ and b_k is ∞). Each input data item is in one interval. Because the way these k data items are picked, there are no more than M/k data items from any one of S_i, $0 \le i < \sqrt{k}$, falls within any interval. Each data item uses k processors to compare itself with the k picked data items to determine into which interval it falls. Thus, each set S_i, $0 \le i < \sqrt{k}$, is divided into $k+1$ segments with each segment containing no more than M/k data items. We first compress the data items from S_i, $0 \le i < \sqrt{k}$, in the 0th interval, in constant time, using the compressing algorithm in the previous section. We then move data item a in processor i, $0 \le i < M$, to processor $M(j-1)+i$ if a falls into the jth interval for $j > 0$. We compress these data items. We then arrange data items so that data items that fall into the 0th interval are before data items in other intervals. What we have done so far is to have data items in the $j+1$th interval follow data items in the jth interval. The data items in the jth interval are coming from S_i, $0 \le i < M/\sqrt{k}$. Therefore, what we need to do next is to merge the $M/\sqrt{k}$ sets, $T_0, T_1, \ldots, T_{\sqrt{k}-1}$, of sorted data items in each interval into one set. As we have explained before, each T_i, $0 \le i < \sqrt{k}$, has no more than M/k data items. Thus, the merging which needs to be done takes $g(M/k, \sqrt{k})$ time.

From the above paragraph, we arrive at two equations:

$$f(M) = f(M/\sqrt{k}) + g(M/\sqrt{k}, \sqrt{k}),$$
$$g(M/\sqrt{k}, \sqrt{k}) = g(M/k, \sqrt{k}) + c_1,$$

where c_1 is a constant. The first equation says that to sort M data items we need to only sort $M/\sqrt{k}$ data items (in parallel) plus merge $\sqrt{k}$ sorted sets with each set containing no more than $M/\sqrt{k}$ data items. The second equation says that we can expend constant time to reduce the problem of merging $\sqrt{k}$ sorted sets with each set containing no more than $M/\sqrt{k}$ data items to the problem of merging $\sqrt{k}$ sorted sets with each set containing no more than M/k data items. We can also add that $f(k) = c_3$ and $g(\sqrt{k}, \sqrt{k}) = c_2$ because k data items can be sorted in constant time with k^2 processors using the sorting algorithm in previous paragraph.

Solving $g(M/\sqrt{k}, \sqrt{k}) = g(M/k, \sqrt{k}) + c_1$, we obtain that $g(M, \sqrt{k}) = c_3 \log M / \log k$, where c_3 is a constant.

We now have that

$$f(M) = f(M/\sqrt{k}) + c_3 \log M / \log k$$
$$= \frac{c_3}{\log k} \log\left(M \cdot \frac{M}{\sqrt{k}} \cdot \frac{M}{k} \cdot \cdot\cdot\right).$$

Because $M = k^{\sqrt{\log\log N}}$, we have that

$$f(M) = \frac{c_3}{\log k} \log \frac{M^{2\sqrt{\log\log N}+1}}{k^{\sqrt{\log\log N}\left(2\sqrt{\log\log N}+1\right)/2}}$$
$$= \frac{c_3}{\log k} \log k^{\sqrt{\log\log N}\left(2\sqrt{\log\log N}+1\right)/2}$$
$$= O(\log\log N).$$

Using the ideas in [2], we now show how to speed up the selection process. Suppose that we have L data items and $N = Lk$ processors and we are going to select the sth item. We show how to find a data item p such that the rank of p is between $s - L/\left(2k^{\sqrt{\log\log N}/4}\right)$ and s. The same method also allows us to find another data item q such that the rank of q is between s and $s + L/\left(2k^{\sqrt{\log\log N}/4}\right)$. After we find p and q, we can then eliminate $L - L/k^{\sqrt{\log\log N}/4}$ data items. Therefore, we reduce the selection problem of selecting from L data items to the problem of selecting from $L/k^{\sqrt{\log\log N}/4}$ data items. As will be seen, the whole reduction takes $O(\log\log N)$ time.

We use S to denote the set of L data items. We have Lk processors and we are going to select the sth data item. We use two stages. In the first stage, we first divide L data items in S into $L/k^{\sqrt{\log\log N}}$ sets with each set containing $k^{\sqrt{\log\log N}}$ data items. We sort each set. Because k processors can be allocated to each data item, the sorting can be done in $O(\log\log N)$ time by our sorting algorithm given in previous paragraphs. We pick every $k^{\sqrt{\log\log N}/2}$th item from each sorted set and form set S_1. The item ranked s_1th in set S_1 is ranked at least $s_1 k^{\sqrt{\log\log N}/2}$th and at most

$$\left(s_1 k^{\sqrt{\log\log N}/2} + L/k^{\sqrt{\log\log N}/2}\right)\text{th}$$

in S. Let

$$s_1 = \left\lfloor s/k^{\sqrt{\log\log N}/2} - L/k^{\sqrt{\log\log N}} \right\rfloor.$$

We now try to find the s_1th items in S_1. Note that S_1 has only $L/k^{\sqrt{\log\log N}/2}$ data items.

In the second stage, we execute a loop until we find p. Suppose we have a set S_i of L_i data items and $L_i k_i$ processors at the beginning of the ith iteration of the loop and we are looking for the s_ith item. If $k_i \geq L_i$, we use our constant time sorting algorithm to sort the L_i data items and then return the s_ith data item. Otherwise, we divide L_i data items into L_i/k_i sets with each set containing k_i items. We then sort each set in constant time by using our sorting algorithm. After sorting, we pick every $\sqrt{k_i}$th item from each set and form set S_{i+1}. The s_{i+1}th item in S_{i+1} is ranked at least $s_{i+1}\sqrt{k_i}$ and at most $s_{i+1}\sqrt{k_i} + L_i/\sqrt{k_i}$ in S_i. Let $s_{i+1} = \left\lfloor s_i/\sqrt{k_i} - L_i/k_i \right\rfloor$. And, we have finished the ith iteration of the loop.

Suppose that the loop in the second stage executed j iterations and p is the final item returned from the jth iteration which ranked s_j in S_j. By the way, we fix s_i, $2 \leq i \leq j$, the rank of p in S_1 is at most s_1. Also, by the way, we fix s_i, $2 \leq i \leq j$, p is ranked at least $s_{j-1} - L_{j-1}/\sqrt{k_{j-1}}$ in S_{j-1}, at least

$$s_{j-2} - L_{j-2}/\sqrt{k_{j-2}} - L_{j-1}/\sqrt{k_{j-1}/k_{j-2}}$$

in $S_{j-2}, \ldots,$ at least

$$s_1 - L_1/\sqrt{k_1} - L_2/\sqrt{k_2/k_1} - \cdots$$
$$- L_{j-1}/\sqrt{k_{j-1}/(k_1 k_2 \cdots k_{j-2})} = s_1 - \sum_{i=1}^{j} L_i/\sqrt{k_i/\prod_{m=1}^{i-1} k_m}$$

in S_1. Because $L_{i+1} = L_i/\sqrt{k_i}$, we have

$$k_{i+1} = N/L_{i+1} = N\sqrt{k_i}/L_i = k_i^{3/2}.$$

Thus, p is ranked at least $s_1 - 2L_1/\sqrt{k_1}$ in S_1. Also, by the way, we fix s_1, p is ranked at most s in S and at least

$$\left(s_1 - 2L_1/\sqrt{k_1}\right)k^{\sqrt{\log\log N}/2}$$
$$\geq \left(s/k^{\sqrt{\log\log N}/2} - L/k^{\sqrt{\log\log N}} - 2L_1/\sqrt{k_1}\right)k^{\sqrt{\log\log N}/2}$$
$$= s - L/k^{\sqrt{\log\log N}/2} - 2L_1 k^{\sqrt{\log\log N}/2}/\sqrt{k_1}$$

in S. Because $L_1 = L/k^{\sqrt{\log\log N}/2}$, we have

$$k_1 = k^{1+\sqrt{\log\log N}/2}.$$

Thus p is ranked at least $s - L/\left(2k^{\sqrt{\log\log N}/4}\right)$.

After we find q (which is symmetrical to p) which is ranked at least s and at most $s + L/\left(2k^{\sqrt{\log\log N}/4}\right)$, we can use p and q to eliminate most of the data items in S so that there will be at most $L/k^{\sqrt{\log\log N}/4}$ data items left from which we form set T. Suppose the rank of p in S is s'. By setting a 1 for data items less than p and setting a 0 for other data items, and by using summation on binary bits in

constant time, we can compute s'. We have now reduced the selection problem of selecting the sth data item in S to the problem of selecting the $(s-s')$th data item in T. This reduction takes $O(\log\log N)$ time.

Thus, we are able to reduce a selection problem of size L with Lk processors to the selection problem of size $L/k^{\sqrt{\log\log N}/4}$ in $O(\log\log N)$ time. We call such a reduction a phase of our algorithm. Initially, the processor advantage is a constant c, i.e., we could use one processor to simulate c processors. After i phases of reduction, the number of data items left will be no more than $N/c^{\left(\sqrt{\log\log N}/4\right)^i}$. Thus, the total number of phases needed to reduce the number of data items to constant is $O(\log\log N/\log\log\log N)$. Because we use $O(\log\log N)$ time for each phase, the time complexity of our algorithm is $O((\log\log N)^2/\log\log\log N)$.

Theorem 1. *The selection problem for N data items can be solved on an N processor LARPBS in $O((\log\log N)^2/\log\log\log N)$ time.*

5 Conclusions

We have demonstrated a fast deterministic selection algorithm on the LARPBS model. Because certain operations such as compression and sorting a smaller set of data can be done in constant time on the LARPBS model, we are able to take advantage of them and to make our selection algorithm faster than any existing PRAM selection algorithm in the literature. We believe many other algorithms can also take advantage of the high communication bandwidth on the LARPBS model. Our selection algorithm may also implicitly improve the results for many other algorithms on the LARPBS and related models. We expect to see more results in this area published in the future.

Acknowledgments

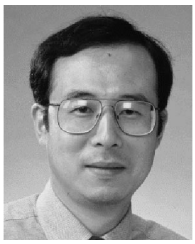
This research was supported in part by the US National Science Foundation under Grants CCR-9211621, OSR-9350540, and CCR-9503882 and the Australian Research Council under its Large Grants Scheme (1996-98) A849602031 and Small Grants Scheme (1998).

References

[1] S. Chaudhuri, T. Hagerup, and R. Raman, *Computer Science,* Springer-Verlag, pp. 352-361, 1993.
[2] R.J. Cole, "An Optimally Efficient Selection Algorithm," *Information Processing Letters 26,* pp. 295-299, 1987/1988.
[3] B. Cong, "Mapping of ANNs on Linear Array with a Reconfigurable Pipelined Bus System," *Proc. 1997 Int'l Conf. Parallel and Distributed Processing Techniques and Applications,* pp. 522-529, 1997.
[4] P.W. Dowd, "Wavelength Division Multiple Access Channel Hypercube Processor Interconnection," *IEEE Trans. Computers,* vol. 41, no. 10, pp. 1223-1241, Oct. 1992.
[5] M. Hamdi, C. Qiao, Y. Pan, and J. Tong, "Communication-Efficient Sorting Algorithms on Reconfigurable Array of Processors with Slotted Optical Buses," *J. Parallel and Distributed Computing,* vol. 57, no. 2, pp. 166-187, May 1999.
[6] Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan, "Pipelined Communication in Optically Interconnected Arrays," *J. Parallel and Distributed Computing,* vol. 12, no. 3, pp. 269-282, 1991.
[7] Z. Guo, "Sorting on Array Processors with Pipelined Buses," *Proc. 1992 Int'l Conf. Parallel Processing,* pp. 289-292, 1992.
[8] Z. Guo, "Optically Interconnected Processor Arrays with Switching Capacity," *J. Parallel and Distributed Computing,* vol. 23, pp. 314-329, 1994.
[9] H. Kimm, "Inversion Number Algorithm on a Linear Array with Reconfigurable Pipelined Bus System," *Proc. 1996 Int'l Conf. Parallel and Distributed Processing Techniques and Applications,* pp. 1398-1408, 1996.
[10] S. Levitan, D. Chiarulli, and R. Melhem, "Coincident Pulse Techniques for Multiprocessor Interconnection Structures," *Applied Optics,* vol. 29, no. 14, pp. 2024-2039, 1990.
[11] K. Li, Y. Pan, and S.-Q. Zheng, "Fast and Processor Efficient Parallel Matrix Multiplication Algorithms on a Linear Array with Reconfigurable Pipelined Bus System," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 8, pp. 705-720, Aug. 1998.
[12] K. Li, Y. Pan, and S.-Q. Zheng, "Parallel Matrix Computations Using a Reconfigurable Pipelined Optical Bus," *J. Parallel and Distributed Computing,* vol. 59, no. 1, pp. 13-30, Oct. 1999.
[13] K. Li, Y. Pan, and S.-Q. Zheng, "Efficient Deterministic and Probabilistic Simulations of PRAMs on Linear Arrays with Reconfigurable Pipelined Bus Systems," *The J. Supercomputing,* vol. 15, no. 2, pp. 163-181, Feb. 2000.
[14] Y. Li, Y. Pan, and S.-Q. Zheng, "Pipelined Time-Division Multiplexing Optical Bus with Conditional Delays," *Optical Eng.,* vol. 36, no. 9, pp. 2417-2424, Sept. 1997.
[15] A. Louri, "Three-Dimensional Optical Architecture and Data-Parallel Algorithms for Massively Parallel Computing," *IEEE Micro,* vol. 11, no. 2, Apr. 1991.
[16] R. Melhem, D. Chiarulli, and S. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems," *The Computer J.,* vol. 32, no. 4, pp. 362-369, 1989.
[17] Y. Pan, "Hough Transform on Arrays with an Optical Bus," *Proc. Fifth ISMM Int'l Conf. Parallel and Distributed Computing and Systems,* pp. 161-166, 1992.
[18] Y. Pan, "Order Statistics on Optically Interconnected Multiprocessor Systems," *Proc. First Int'l Workshop Massively Parallel Processing Using Optical Interconnections,* pp. 162-169, 1994.
[19] Y. Pan and M. Hamdi, "Quicksort on a Linear Array with a Reconfigurable Pipelined Bus System," *Proc. IEEE Int'l Symp. Parallel Architectures, Algorithms, and Networks,* pp. 313-319, 1996.
[20] Y. Pan and K. Li, "Linear Array with a Reconfigurable Pipelined Bus System—Concepts and Applications," *Information Sciences,* vol. 106, no. 3/4, pp. 237-258, May 1998.
[21] Y. Pan, "Basic Data Movement Operations on the LARPBS Model," *Parallel Computing Using Optical Interconnections,* K. Li, Y. Pan, and S.Q. Zheng, eds., Boston: Kluwer Academic Publishers, 1998.
[22] Y. Pan, K. Li, and S.-Q. Zheng, "Fast Nearest Neighbor Algorithms on a Linear Array with a Reconfigurable Pipelined Bus System," *Parallel Algorithms and Applications,* vol. 13, pp. 1-25, 1998.
[23] S. Pavel and S.G. Akl, "On the Power of Arrays with Optical Pipelined Buses," *Proc. 1996 Int'l Conf. Parallel and Distributed Processing Techniques and Applications,* pp. 1443-1454, 1996.
[24] S. Pavel, "Computation and Communication Aspects of Arrays with Optical Pipelined Buses," PhD Dissertation, Dept. of Computing and Information Science, Queen's Univ., Canada, Oct. 1996.
[25] S. Pavel and S.G. Akl, "Matrix Operations Using Arrays with Reconfigurable Optical Buses, *Parallel Algorithms and Applications,* vol. 11, pp. 223-242, 1996.
[26] S. Pavel and S.G. Akl, "Integer Sorting and Routing in Arrays with Reconfigurable Optical Bus," *Proc. 1996 Int'l Conf. Parallel Processing,* vol. III, pp. 90-94, Aug. 1996.
[27] C. Qiao, R. Melhem, D. Chiarulli, and S. Levitan, "Optical Multicasting in Linear Arrays," *Int'l J. Optical Computing,* vol. 2, no. 1, pp. 31-48, 1991.
[28] C. Qiao and R. Melhem, "Time-Division Optical Communications in Multiprocessor Arrays," *IEEE Trans. Computers,* vol. 42, no. 5, pp. 577-590, May 1993.
[29] C. Qiao, "Efficient Matrix Operations in a Reconfigurable Array with Spanning Optical Buses," *Proc. Fifth IEEE Symp. Frontiers of Massively Parallel Computations,* pp. 273-280, 1995.

[30] S. Rajasekaran and S. Sahni, "Sorting, Selection and Routing on the Arrays with Reconfigurable Optical Buses," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 11, pp. 1123-1132 Nov. 1997.
[31] S. Sahni, "Models and Algorithms for Optical and Optoelectronic Parallel Computers," *Proc. Fourth IEEE Int'l Symp. Parallel Architectures, Algorithms, and Networks,* pp. 2-7, 1999.
[32] H. Shen, "Improved Universal k-Selection in Hypercubes," *Parallel Computing,* vol. 18, no. 2, pp. 177-184, 1992.
[33] J.L. Trahan, A.G. Bourgeois, Y. Pan, and R. Vaidyanathan, "An Optimal and Scalable Algorithm for Permutation Routing on Reconfigurable Linear Arrays with Optically Pipelined Buses," *J. Parallel and Distributed Computing,* vol. 60, no. 9, pp. 1125-1136, Sept. 2000.

Yijie Han received the BS degree from the University of Science and Technology of China, Hefei, Anhui, China, and the MA and the PhD degrees from Duke University, Durham, North Carolina, respectively. He was an assistant professor at the University of Kentucky, a lecturer at the Hong Kong University, and an information specialist at Electronic Data Systems, Inc. Currently, he is at the University of Missouri, Kansas City. His main research interest is algorithm design. Parallel algorithms invented by him together with his collegues on several important problems including linked list coloring, graph coloring, all-pairs shortest path, integer sorting, minimum spanning tree and connected components (on the EREW PRAM model), maximal independent set, maximal matching, etc. are the best algorithms to date.



Yi Pan received the BEng degree in computer engineering from Tsinghua University, China, in 1982, and the PhD degree in computer science from the University of Pittsburgh in 1991. Currently, he is an associate professor in the Department of Computer Science at Georgia State University. Previously, he was a faculty member in the Department of Computer Science at the University of Dayton, Ohio. His research interests include parallel algorithms and architectures, optical communication and computing, wireless networks, high-performance data mining, distributed computing, task scheduling, and networking. He has published more than 120 research papers including over 50 papers in international journals such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Reliability*, *IEEE Communications Magazine*, *IEEE Intelligent Systems*, *IEEE Computing in Science and Engineering*, *Journal of Parallel and Distributed Computing*, *Optical Engineering*, and *The Journal of Supercomputing*. He has received many awards including the Outstanding Scholarship Award of the College of Arts and Sciences at University of Dayton (1999), a Japanese Society for the Promotion of Science Fellowship (1998), an AFOSR Summer Faculty Fellowship (1997), US National Science Foundation (NSF) Research Opportunity Awards (1994 and 1996), and the best paper award from PDPTA '96 (1996). His research has been supported by the NSF, the AFOSR, the US Air Force, and the state of Ohio. Dr. Pan is currently an associate editor of the *IEEE Transactions on Systems, Man, and Cybernetics*, area editor-in-chief of the journal of *Information*, editor of the journal of *Parallel and Distributed Computing Practices*, associate editor of the *International Journal of Parallel and Distributed Systems and Networks*, and serves on the editorial board of *The Journal of Supercomputing*. He has served as a guest editor of special issues for several journals, and as general chair, program chair, vice program chair, publicity chair, session chair, and as a member for steering, advisory, and program committees for numerous international conferences and workshops. He is an IEEE Computer Society Distinguished Visitor, a senior member of the IEEE and a member of the IEEE Computer Society. He is listed in *Men of Achievement*, *Marquis Who's Who in America*, and *Marquis Who's Who in Midwest*.

Hong Shen received the BEng degree from the Beijing University of Science and Technology, the MEng degree from the University of Science and Technology of China, and the PhLic and PhD degrees from Abo Akademi University, Finland. Currently, he is a professor in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology. Professor Shen has published more than 130 technical papers on algorithms, parallel and distributed computing, networking, parallel databases, data mining, and multimedia systems. He has served as an editor, associate editor, and editorial-board member of five international journals and chaired several international conferences.