

Short Paper

Optimal Parallel Prefix on the Postal Model*

YEN-CHUN LIN AND CHING-SUNG YEH[†]

Department of Computer Science and Information Engineering

[†]*Department of Electronic Engineering*

National Taiwan University of Science and Technology

Taipei, 106 Taiwan

E-mail: yclin@computer.org

This paper explores the prefix operation on a message-passing fully connected multicomputer with multiport postal communication. We present an exact communication lower bound for the prefix operation on the model. Two efficient parallel prefix algorithms are also presented; they are optimal in terms of the number of communication steps. For an input of size n , one of the algorithms using n processors is also time-optimal; the other algorithm using $p < n$ processors can be cost-optimal and can achieve linear speedup.

Keywords: exact communication lower bound, message-passing multicomputer, parallel algorithms, postal model, prefix operation

1. INTRODUCTION

The prefix operation is defined as follows: Given n values $v(0), v(1), \dots, v(n-1)$ and an associative binary operation \oplus , compute $v(0) \oplus v(1) \oplus \dots \oplus v(i)$, for $0 \leq i \leq n-1$. This operation has many applications [1, 6, 9, 10], and it has been proposed as a primitive operation [4]. Thus, many parallel prefix algorithms on various models have been presented [7-16, 18]. This paper explores the prefix operation on a message-passing fully connected multicomputer with multiport postal communication. On this model, each processing element (PE), in a communication step, can send k messages to k other PEs and receive k messages from k PEs, for some $k \geq 1$. In addition, to model the communication latency, a message sent in step j is received in step $j + \lambda - 1$, and the received message can then be forwarded in step $j + \lambda$, where $\lambda \geq 1$. This model, called the k -port postal model, or simply the postal model, in this paper, has been recognized as a trend in multicomputers and in programming environments [2, 3, 5].

In a multicomputer, the time taken to transfer a message between two PEs is significant, and it is desirable to have communication-efficient algorithms. This paper contributes to solving the prefix problem in as few communication steps as possible. We present an exact communication lower bound on the k -port postal model, and pro-

Received December 19, 2000; revised April 11 & October 24, 2001; accepted November 13, 2001.
Communicated by Chu-Sing Yang.

* A preliminary version of the paper was presented at the 1999 National Computer Symposium, Taipei, Dec. 1999.

pose two efficient parallel prefix algorithms that are optimal in terms of the number of communication steps. The first algorithm using n processors is also time-optimal. The second algorithm using $p < n$ processors can be cost-optimal and can achieve linear speedup. It should be noted that when $\lambda = 1$, meaning that a message can be received in the same step it is sent, the model is reduced to the k -port model. The prefix problem already has efficient solutions on the simpler k -port model [16].

Section 2 presents the exact communication lower bound for the prefix operation on the postal model. Section 3 gives an efficient prefix algorithm using n PEs, which is not only optimal in terms of the number of communication steps, but is also time-optimal. Section 4 presents a parallel prefix algorithm on a system of $p < n$ PEs; the algorithm is also optimal in terms of the number of communication steps, and may be cost-optimal and achieve linear speedup. Section 5 concludes this paper.

2. COMMUNICATION LOWER BOUND

This section presents a communication lower bound for the prefix operation on the k -port postal model with n PEs. Initially, PE i has $v(i)$, $0 \leq i \leq n - 1$, and PE 0 can send $v(0)$ to k PEs in communication step 1. To complete the prefix operation, every PE must have either $v(0)$ or a value contributed to by $v(0)$; for simplicity, we say that every PE must contain $v(0)$. Let $G(j)$ denote the maximum number of PEs that can contain $v(0)$ after step j in all possible ways, where $j \geq 1$. Since no PEs can receive any message in step j , where $1 \leq j < \lambda$, only PE 0 has $v(0)$ after step j . We also let $G(0) = 1$, indicating that initially, only one PE, i.e., PE 0, contains $v(0)$. Therefore, $G(j) = 1$, for $0 \leq j < \lambda$.

By definition, when $j \geq \lambda$, after step $j - \lambda$, $G(j - \lambda)$ PEs can contain $v(0)$. Each of these $G(j - \lambda)$ PEs can send $v(0)$ to k PEs in step $j - \lambda + 1$; thus, in step j , $kG(j - \lambda)$ PEs can receive $v(0)$. Also by definition, after step $j - 1$, $G(j - 1)$ PEs can contain $v(0)$. Hence, after step j , $G(j - 1) + kG(j - \lambda)$ PEs can contain $v(0)$. That is, $G(j) = G(j - 1) + kG(j - \lambda)$, for $j \geq \lambda$. Suppose that at least i communication steps are needed to solve the prefix problem; then $G(i) \geq n$. Thus, a prefix algorithm needs at least $\min\{j \mid G(j) \geq n\}$ communication steps. Therefore, we have the following theorem.

Theorem 1. On the k -port postal model with n PEs, a prefix algorithm needs at least $\min\{j \mid G(j) \geq n\}$ communication steps, where

$$\begin{aligned} G(j) &= 1, & 0 \leq j < \lambda; \\ G(j) &= G(j - 1) + kG(j - \lambda), & j \geq \lambda. \end{aligned}$$

3. ALGORITHM A

This section presents a prefix algorithm on the k -port postal model with n PEs. To design the algorithm, it helps to consider how $v(0)$ can be sent from PE 0 to other PEs. Since $G(j - 1)$ PEs can contain $v(0)$ after step $j - 1$, where $j \geq 1$, assume that these PEs are numbered $0, 1, \dots, G(j - 1) - 1$. Each of these $G(j - 1)$ PEs can send $v(0)$ to k PEs in step j ; thus, $kG(j - 1)$ PEs can receive $v(0)$ in step $j + \lambda - 1$. To specify which PEs are to receive $v(0)$, we assume that the $G(j + \lambda - 2)$ PEs containing $v(0)$ after step $j + \lambda - 2$ are numbered 0 through $G(j + \lambda - 2) - 1$, and we want the $G(j + \lambda - 1)$ PEs that contain

$v(0)$ after step $j + \lambda - 1$ to be numbered 0 through $G(j + \lambda - 1) - 1$. To achieve this, in step j , a way of passing $v(0)$ to PEs $G(j + \lambda - 2)$ through $G(j + \lambda - 1) - 1$ is as follows. For all $t \in \{0, 1, \dots, k - 1\}$,

PE 0 sends $v(0)$ to PE $G(j + \lambda - 2) + tG(j - 1)$,
 PE 1 sends $v(0)$ to PE $1 + G(j + \lambda - 2) + tG(j - 1)$,

 PE $G(j - 1) - 1$ sends $v(0)$ to PE $G(j - 1) - 1 + G(j + \lambda - 2) + tG(j - 1)$.

That is, for all $t \in \{0, 1, \dots, k - 1\}$,

PE i sends $v(0)$ to PE $i + G(j + \lambda - 2) + tG(j - 1)$, $0 \leq i \leq G(j - 1) - 1$.

Thus, in step $j + \lambda - 1$, $kG(j - 1)$ PEs receive $v(0)$, and consequently, $G(j + \lambda - 2) + kG(j - 1) = G(j + \lambda - 1)$ PEs can contain $v(0)$.

Now we will describe the exact communication pattern among PEs achieved through the construction of n spanning trees rooted at nodes labeled from 0 to $n - 1$, respectively. Let T_i denote the tree rooted at node i , $0 \leq i \leq n - 1$; T_i will finally contain $n - i$ nodes labeled i through $n - 1$, and these nodes represent PE i through PE $n - 1$. The construction process for T_i corresponds to how and when a message containing $v(i)$ is sent from PE i to PEs $i + 1$ through $n - 1$.

The tree T_i is constructed step by step as follows. Initially, T_i consists of only node i , indicating that PE i initially contains $v(i)$. For any node x in the partial tree before step j , the set of child nodes of node x added in step j is defined by

$$S_{x,j} = \{y \mid y = x + G(j + \lambda - 2) + tG(j - 1) < n, 0 \leq t \leq k - 1, t \text{ is an integer}\}.$$

That is, edge (x, y) is added to T_i in step j for each $y \in S_{x,j}$. This means that PE x sends a message containing $v(i)$ to PE y in step j . On the postal model, a message sent in step j is received in step $j + \lambda - 1$, and the received message can then be forwarded in step $j + \lambda$. Thus, if $y \in S_{x,j}$, then node y cannot have a child node before step $j + \lambda$.

For example, consider constructing T_0 (see Fig. 1) with $n = 10$, $k = 2$, $\lambda = 3$. Clearly, $G(0) = 1$, $G(1) = 1$, $G(2) = 1$, $G(3) = 3$, $G(4) = 5$, $G(5) = 7$, and $G(6) = 13$. The construction steps are as follows:

Step 1 ($j = 1$): $S_{0,1} = \{1, 2\}$; add edges $(0, 1)$, $(0, 2)$ to T_0 .
 Step 2 ($j = 2$): $S_{0,2} = \{3, 4\}$; add edges $(0, 3)$, $(0, 4)$ to T_0 .
 Step 3 ($j = 3$): $S_{0,3} = \{5, 6\}$; add edges $(0, 5)$, $(0, 6)$ to T_0 .
 Step 4 ($j = 4$): $S_{0,4} = \{7\}$; add edge $(0, 7)$ to T_0 .
 $S_{1,4} = \{8\}$; add edge $(1, 8)$ to T_0 .
 $S_{2,4} = \{9\}$; add edge $(2, 9)$ to T_0 .

When $j \geq 5$, $S_{x,i} = \emptyset$ for each node x in T_0 . That is, T_0 is completed in four steps. Note that as shown in Fig. 1, a newly added leaf node is white, and it turns black when it has at least one child. Fig. 2 shows the last step of constructing T_1 and T_2 .

By definition, if $y \in S_{x,j-\lambda+1}^*$, then $y = x + G(j - 1) + tG(j - \lambda) < n$, where $0 \leq t \leq k - 1$; thus, $x = y - G(j - 1) - tG(j - \lambda)$. Since a value sent out in step $j - \lambda + 1$ will be received in step j , the set of PEs from which a PE y receives a value in step j , $j \geq \lambda$, is

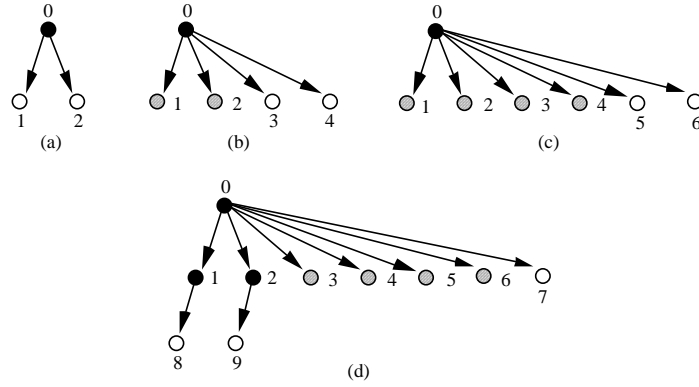


Fig. 1. Construction of T_0 with $n = 10$, $k = 2$, $\lambda = 3$: (a) step 1, (b) step 2, (c) step 3, (d) step 4.

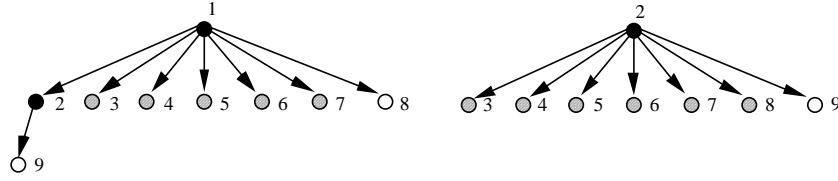


Fig. 2. T_1 and T_2 with $n = 10$, $k = 2$, $\lambda = 3$.

$$R_{y,j} = \{ x \mid x = y - G(j-1) - tG(j-\lambda) \geq 0, 0 \leq t \leq k-1, t \text{ is an integer} \}.$$

Now we can present the parallel prefix algorithm shown in Fig. 3. The algorithm consists of $\min\{j \mid G(j) \geq n\}$ time steps, each being an iteration of the **for** loop. Note that we use $i:j$ to represent the result of $v(i) \oplus v(i+1) \oplus \dots \oplus v(j)$, where $i \leq j$.

Algorithm A. /* Prefix algorithm executed by PE x , $0 \leq x < n$, on the k -port postal model. A memory location in PE x , $c(x)$, initially stores $v(x)$, then a partial result, and finally $0:x$. The cardinality of $R_{x,j}$ is $|R_{x,j}| = h$, and e_r denotes the r th smallest element in $R_{x,j}$ */

```

m := min{ j | G(j) ≥ n }
for j = 1 to m do          /* min{ j | G(j) ≥ n } time steps */
  if 1 ≤ j ≤ m - λ + 1 then
    for all y ∈ Sx,j, send c(x) to PE y      /* 1 ≤ |Sx,j| ≤ k */
  end if
  if λ ≤ j ≤ m then
    for all r ∈ {1, 2, ..., h}, receive c(er) from PE er      /* 1 ≤ h ≤ k */
    c(x) := c(e1) ⊕ c(e2) ⊕ ... ⊕ c(eh) ⊕ c(x)
  end if
end for

```

Fig. 3. Parallel prefix algorithm using n PEs on the k -port postal model.

To illustrate the execution of Algorithm A, consider the case where $n = 10$, $k = 2$, $\lambda = 3$. Because $G(5) = 7$ and $G(6) = 13$, we have $m = 6$. We will use $C_j = \langle c_j(0), c_j(1), \dots, c_j(9) \rangle$ to represent the values of $c(0), c(1), \dots, c(9)$ in PE 0, PE 1, ..., PE 9, respectively, after time step j . In particular, C_0 represents initial values of $c(i)$, and we let $c(i) = i$, $0 \leq i \leq 9$; that is,

$$C_0 = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle.$$

In time step 1, $S_{i,1} = \{i + 1, i + 2\}$ for $0 \leq i \leq 7$, and $S_{8,1} = \{9\}$. The values sent out are taken from C_0 . Since none are received in time step 1, after time step 1 we have

$$C_1 = C_0 = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle.$$

In time step 2, $S_{i,2} = \{i + 3, i + 4\}$ for $0 \leq i \leq 5$, and $S_{6,2} = \{9\}$. The values sent out are taken from C_1 . Since none are received in time step 2, we have

$$C_2 = C_1 = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle.$$

In time step 3, $S_{i,3} = \{i + 5, i + 6\}$ for $0 \leq i \leq 3$, and $S_{4,3} = \{9\}$. The values sent out are taken from C_2 . Now, $R_{0,3} = \phi$, $R_{1,3} = \{0\}$, and $R_{i,3} = \{i - 2, i - 1\}$ for $2 \leq i \leq 9$. The values received come from C_0 ; thus,

$$C_3 = \langle 0, 0:1, 0:2, 1:3, 2:4, 3:5, 4:6, 5:7, 6:8, 7:9 \rangle.$$

In time step 4, $S_{i,4} = \{i + 7\}$ for $0 \leq i \leq 2$. The values sent out are taken from C_3 . Now, $R_{i,4} = \phi$ for $0 \leq i \leq 2$, $R_{3,4} = \{0\}$, and $R_{i,4} = \{i - 4, i - 3\}$ for $4 \leq i \leq 9$. The values received come from C_1 ; thus,

$$C_4 = \langle 0, 0:1, 0:2, 0:3, 0:4, 1:5, 2:6, 3:7, 4:8, 5:9 \rangle.$$

In time step 5, no data are sent out. Now, $R_{i,5} = \phi$ for $0 \leq i \leq 4$, $R_{5,5} = \{0\}$, and $R_{i,5} = \{i - 6, i - 5\}$ for $6 \leq i \leq 9$. The values received come from C_2 ; thus,

$$C_5 = \langle 0, 0:1, 0:2, 0:3, 0:4, 0:5, 0:6, 1:7, 2:8, 3:9 \rangle.$$

In time step 6, no data are sent out. Now, $R_{i,6} = \phi$ for $0 \leq i \leq 6$, and $R_{i,6} = \{i - 7\}$ for $7 \leq i \leq 9$. The values received come from C_3 ; thus,

$$C_6 = \langle 0, 0:1, 0:2, 0:3, 0:4, 0:5, 0:6, 0:7, 0:8, 0:9 \rangle.$$

Therefore, it takes 6 time steps to complete the prefix operation.

Fig. 4 shows the communications of the above example. Every vertical line represents a PE whose label is on top of the line. Each horizontal line represents a time step. Every oblique line is assumed to be directed downward and ends with a dot on a vertical line. An oblique line that starts at vertical line i at time step t and ends at vertical line j at time step $t + 2$ denotes that a message is sent by PE i at time step t and received by PE

j at time step $t + 2$. Furthermore, a dot on a vertical line j denotes that PE j performs \oplus operations on the received values; that is, $c(e_1) \oplus c(e_2) \oplus \dots \oplus c(e_h) \oplus c(j)$. For ease of understanding, oblique lines that start at even-numbered time steps are dashed.

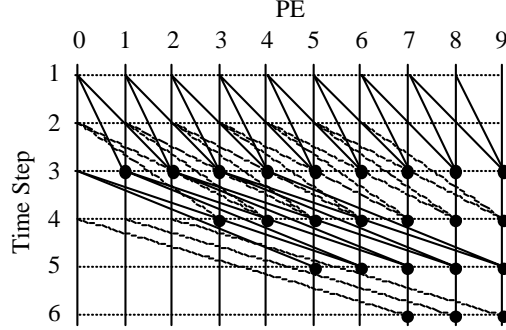


Fig. 4. Communication graph of Algorithm A for $n = 10$, $k = 2$, $\lambda = 3$.

Lemma 2. In time step $j (\geq \lambda)$ of Algorithm A:

- Case 1. If $0 \leq i < G(j - 1)$, then PE i receives no data.
- Case 2. If $G(j - 1) + sG(j - \lambda) \leq i < G(j - 1) + (s + 1)G(j - \lambda)$, $0 \leq s \leq k - 1$ and s is an integer, then for $t \in \{0, 1, \dots, s\}$, PE i receives $c(i - G(j - 1) - tG(j - \lambda))$ from PE $(i - G(j - 1) - tG(j - \lambda))$.
- Case 3. If $G(j - 1) + kG(j - \lambda) \leq i < n$, then for $t \in \{0, 1, \dots, k - 1\}$, PE i receives $c(i - G(j - 1) - tG(j - \lambda))$ from PE $(i - G(j - 1) - tG(j - \lambda))$.

Proof: The proof is straightforward, and can be found in [17].

Lemma 3. After time step j , $1 \leq j \leq m$, of Algorithm A, PE i , $0 \leq i < n$, contains

$$\begin{aligned} c(i) &= 0:i, & \text{if } 0 \leq i < G(j); \\ c(i) &= (i - G(j) + 1):i, & \text{if } G(j) \leq i < n. \end{aligned}$$

Proof: The proof is by induction on j and can be found in [17].

Theorem 4. Algorithm A performs the prefix operation in $\Theta(m)$ time, where $m = \min\{j \mid G(j) \geq n\}$. It is optimal in terms of the number of communication steps and is time-optimal.

Proof: By Lemma 3, after time step j , PE i contains $c(i) = 0:i$ if $0 \leq i < G(j)$. Thus, after time step m , PE i has the final result $c(i) = 0:i$, $0 \leq i < n \leq G(m)$. The number of communication steps taken by Algorithm A, m , equals the lower bound given in Theorem 1.

The computing time of Algorithm A is $O(k(m - \lambda))$. Since k and λ are constants for any particular computer, $O(k(m - \lambda)) = O(m)$. The computing time plus the communication time is $O(m) + \Theta(m) = \Theta(m)$. Because Theorem 1 implies that the prefix problem requires $\Omega(m)$ time, Algorithm A is time-optimal. Q.E.D.

4. ALGORITHM B

In this section, we will present a parallel prefix algorithm that uses only $p < n$ PEs. The algorithm is derived from Algorithm A. Using the same number of PEs on the postal model, the algorithms have the same communication pattern. Let $q = n/p$. Assume that initially, PE 0 through PE $(n - p \times \lfloor q \rfloor - 1)$ each contain $\lceil q \rceil$ initial values, and that the other PEs each contain $\lfloor q \rfloor$ values. For ease of presentation, we further assume that q is an integer, and that PE i initially contains $v(iq), v(iq+1), \dots, v((i+1)q-1)$, for $0 \leq i < p$. PE i uses the following memory locations: $c(i)$, $d(i)$, and $temp$ for saving partial results; and an array of q elements $u[iq..(i+1)q-1]$ for storing input data $v(iq), v(iq+1), \dots, v((i+1)q-1)$ and final results $0:iq, 0:iq+1, \dots, 0:(i+1)q-1$. Fig. 5 gives the algorithm.

Algorithm B. /* Prefix algorithm executed by PE x , $0 \leq x < p < n$, on the k -port postal model. Assume that $q = n/p$ is an integer. The cardinality of R_{xj} is $|R_{xj}| = h$, and e_r denotes the r th smallest element in R_{xj} . */

```

 $c(x) := u[iq] \oplus u[iq + 1] \oplus \dots \oplus u[(i + 1)q - 1]$ 
 $d(x) := u[iq]$ 
 $s := \min\{j \mid G(j) \geq p\}$ 
for  $j = 1$  to  $s$  do
  if  $1 \leq j \leq s - \lambda + 1$  then
    for all  $y \in S_{xj}$ , send  $c(x)$  to PE  $y$       /*  $1 \leq |S_{xj}| \leq k$  */
  end if
  if  $\lambda \leq j \leq s$  then
    for all  $r \in \{1, 2, \dots, h\}$ , receive  $c(e_r)$  from PE  $e_r$       /*  $1 \leq h \leq k$  */
     $temp := c(e_1) \oplus c(e_2) \oplus \dots \oplus c(e_h)$ 
     $c(x) := temp \oplus c(x)$ 
     $d(x) := temp \oplus d(x)$ 
  end if
end for
 $u[xq] := d(x)$ 
for  $j = 1$  to  $q - 1$  do
   $u[xq + j] := u[xq + j - 1] \oplus u[xq + j]$ 
end for

```

Fig. 5. Prefix algorithm using $p < n$ PEs on the k -port postal model.

Theorem 5. Algorithm B solves the prefix problem in $\Theta(n/p + s)$ time, where $s = \min\{j \mid G(j) \geq p\}$. It is optimal in terms of the number of communication steps. When $n = \Omega(ps)$, it is cost-optimal and achieves linear speedup.

Proof: Algorithm B takes only s communication steps; by Theorem 1, this is optimal. The communication time is thus $\Theta(s)$. In addition, the algorithm requires $\Theta(n/p) + O(ks)$ computing time. Therefore, the total execution time is $\Theta(n/p + s) + O(ks)$. Since k is a constant for any particular computer, the execution time is $\Theta(n/p + s)$. If $n = \Omega(ps)$,

then $n/p = \Omega(s)$, and $\Theta(n/p + s) = \Theta(n/p)$. The prefix operation requires $\Theta(n)$ time on a sequential computer; thus, the algorithm is cost-optimal and achieves linear speedup.

Q.E.D.

5. CONCLUDING REMARKS

We have presented an exact communication lower bound for the prefix problem on the postal model. Two efficient prefix algorithms have also been introduced. The first algorithm uses n PEs; it is not only optimal in terms of the number of communication steps, but is also time-optimal. The second one solves the prefix problem by using only p PEs, where $p < n$. It is also optimal in terms of the number of communication steps. Furthermore, it may be cost-optimal and may achieve linear speedup.

ACKNOWLEDGMENT

This research was supported in part by the National Science Council of the R.O.C. under contract NSC 87-2213-E-011-005.

REFERENCES

1. S. G. Akl, *Parallel Computation: Models and Methods*, Prentice-Hall, Upper Saddle River, NJ, 1997.
2. A. Bar-Noy, J. Bruck, C. T. Ho, S. Kipnis, and B. Schieber, "Computing global combine operations in the multiport postal model," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, 1995, pp. 896-900.
3. A. Bar-Noy and S. Kipnis, "Multiple message broadcasting in the postal model," *Networks*, Vol. 29, 1997, pp. 1-10.
4. G. E. Blelloch, "Scans as primitive operations," *IEEE Transactions on Computers*, Vol. 38, 1989, pp. 1526-1538.
5. J. Bruck and C. T. Ho, "On the design and implementation of broadcast and global combine operations using the postal model," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, 1996, pp. 256-265.
6. A. L. Fisher and A. M. Ghuloum, "Parallelizing complex scans and reductions," in *Proceedings of ACM SIGPLAN '94 Conference on Programming Language Design and Implementation*, 1994, pp. 135-146.
7. C. P. Kruskal, T. Madej, and L. Rudolph, "Parallel prefix on fully connected direct connection machines," in *Proceedings of International Conference on Parallel Processing*, 1986, pp. 278-284.
8. R. E. Ladner and M. J. Fischer, "Parallel prefix computation," *Journal of the Association for Computing Machinery*, Vol. 27, 1980, pp. 831-838.
9. S. Lakshmirarahan and S. K. Dhall, *Parallel Computing Using the Prefix Problem*, Oxford University Press, Oxford, UK, 1994.
10. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.
11. Y.-C. Lin, "Optimal parallel prefix circuits with fan-out 2 and corresponding parallel

- algorithms,” *Neural, Parallel & Scientific Computations*, Vol. 7, 1999, pp. 33-42.
12. Y.-C. Lin, Y. H. Hsu, and C. K. Liu, “Depth-size optimal parallel prefix circuits with fan-out 4 and small depth,” in *Proceedings of International Conference on Parallel and Distributed Computing, Applications, and Technologies*, 2001, pp. 74-82.
 13. Y.-C. Lin and C. M. Lin, “Efficient parallel prefix algorithms on multicomputers,” *Journal of Information Science and Engineering*, Vol. 16, 2000, pp. 41-64.
 14. Y.-C. Lin and C. C. Shih, “Optimal parallel prefix circuits with fan-out at most 4,” in *Proceedings of 2nd IASTED International Conference on Parallel and Distributed Computing and Networks*, 1998, pp. 312-317.
 15. Y.-C. Lin and C. C. Shih, “A new class of depth-size optimal parallel prefix circuits,” *The Journal of Supercomputing*, Vol. 14, 1999, pp. 39-52.
 16. Y.-C. Lin and C. S. Yeh, “Efficient parallel prefix algorithms on multiport message-passing systems,” *Information Processing Letters*, Vol. 71, 1999, pp. 91-95.
 17. Y.-C. Lin and C. S. Yeh, “Optimal parallel prefix on the postal model,” Technical Report, NTUST-ET-TR01002, Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, Apr. 2001.
 18. M. Snir, “Depth-size trade-offs for parallel prefix computation,” *Journal of Algorithms*, Vol. 7, 1986, pp. 185-201.

Yen-Chun Lin (林彥君) received his Ph.D. degree in electrical engineering from National Taiwan University in 1988. Since 1988 he has been on the faculty at National Taiwan Institute of Technology, which was renamed as National Taiwan University of Science and Technology in 1997. He was a professor in Department of Electronic Engineering from February 1993 to July 2001. Dr. Lin transferred to Department of Computer Science and Information Engineering in August 2001, and was director of Internet Technology Center. He served as program chair of the 2001 International Conference on Parallel and Distributed Computing, Applications, and Technologies. He was visiting scientist at the IBM Almaden Research Center, San Jose, California, from 19993 to 1994. He was a senior software engineer at Wang Computer (Taiwan) Ltd. From 1984 to 1985 and was an associate development engineer at IBM Taiwan Corporation from 1985 to 1988. His research interests include parallel computing and multimedia systems. Dr. Lin is a member of the IEEE and the ACM.

Ching-Sung Yeh (葉青松) received his bachelor degree in computer science and information engineering from Tamkang University, Tamsui, Taiwan in 1996 and received his M.S. degree in electronic engineering from National Taiwan University of Science and Technology, Taipei, Taiwan in 1998. His research interests include parallel computing.