Prefix Computations on a Generalized Mesh-Connected Computer with Multiple Buses

Kuo-Liang Chung

Abstract— **The mesh-connected computer with multiple buses (MCCMB) is a well-known parallel organization, providing broadcast facilities in each row and each column. In this paper, we propose a 2-D generalized MCCMB (2-GMCCMB) for the purpose of increasing the efficiency of executing some important applications of prefix computations such as solving linear recurrences and tridiagonal systems, etc. A $k_1n_1 \times k_1n_2$ 2-GMCCMB is constructed from a $k_1n_1 \times k_1n_2$ mesh organization by enhancing the power of each disjoint $n_1 \times n_2$ submesh with multiple buses (sub-2-MCCMB). Given n data, a prefix computation can be performed in $O(n^{1/10})$ time on an $n^{3/5} \times n^{2/5}$ 2-GMCCMB, where each disjoint sub-2-MCCMB is of size $n^{1/2} \times n^{3/10}$. This time bound is faster than the previous time bound of $O(n^{1/8})$ for the same computation on an $n^{5/8} \times n^{3/8}$ 2-MCCMB. Furthermore, the time bound of our parallel prefix algorithm can be further reduced to $O(n^{1/11})$ if fewer processors are used. Our result can be extended to the d-dimensional GMCCMB, giving a time bound of $O(n^{1/(d2^d+d)})$ for any constant d; here, we omit the constant factors. This time bound is less than the previous time bound of $O(n^{1/(d2^d)})$ on the d-dimensional MCCMB.**

Index Terms— **Broadcasting, mesh-connected computers, mesh-connected computers with multiple buses, parallel algorithms, prefix computation, rectangular meshes.**

I. Introduction

The mesh-connected computer (MCC) has been of great interest to computer researchers. The main advantages of this organization are threefold: (1) it has a simple and modular connection pattern; (2) it corresponds to the data format of many applications in matrix computations, image processing, computational geometry, and graph algorithms; (3) it is suitable for VLSI implementation [17]. On a 2-D MCC (2-MCC), each processor is connected to its nearest neighbors by local links and can route data into one of its four nearest neighbors in unit time [22]. Based on this organization, parallel machines such as the Illiac IV [4], [14], [16] have been built. Also, some important parallel algorithms have been developed in [4], [30], [20], [21], [2], [18], [19].

The main drawback of MCC is its large diameter. On an $n^{1/2} \times n^{1/2}$ 2-MCC, for example, to route a data may take $\Omega(\sqrt{n})$ time in the worst case. To overcome the large diameter problem, it has been proposed to augment MCC's by adding the processors the ability to do broadcasting [28], [8], [1], [29], [24], [9], [10]. The broadcast mechanism can be implemented using a bus or a collection of buses. It is assumed that broadcasting takes one unit time and one processor is permitted to broadcast data on each row bus and each column bus at a time. Bokhari [8] showed how the 2-MCC with a global bus for broadcasting can be used to find the maximum in $O(n^{1/3})$ time.

Fig. 1 shows a 4×4 2-MCC with multiple buses (2-MCCMB) with a bus for each row and each column. Processor i, $1 \leq i \leq 16$, can broadcast data to the other processors in the same row (or column) via the row (or column) bus. Based on the 2-MCCMB organization, parallel machines such as the AMT DAP 500 [23] and the GRID array processor [25] have been built. Prefix computations [6], [7] are an important kernel of many typical algorithms. On a square 2-MCCMB with n processors, Prasanna Kumar and Raghavendra [24] and Stout [29] proposed $O(n^{1/6})$ parallel algorithms for prefix computations. Recently, the time bound is reduced from $O(n^{1/6})$ to $O(n^{1/8})$ for the same computations on a rectangular 2-MCCMB of size $n^{5/8} \times n^{3/8}$ [11], [3]. Also, some improved parallel algorithms for the selection problem [12], [5] are designed on this model. The main drawback of MCCMB's is the bus-contension problem for broadcasting data on buses.

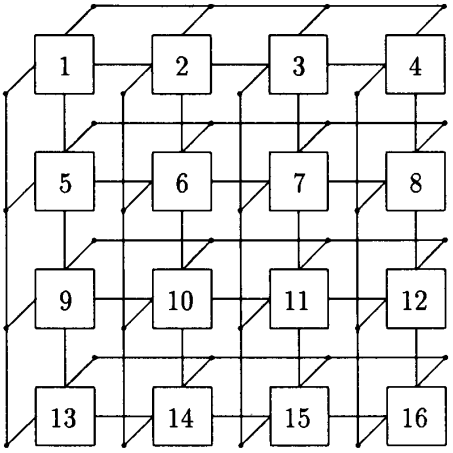

Fig. 1. A 4×4 2-MCCMB.

In this paper, we propose a generalized 2-MCCMB (2-GMCCMB) for the purpose of alleviating the bus-contension problem, keeping the good features of MCC's and MCCMB's, and increasing the efficiency of executing some important applications of prefix computatons. A $k_1n_1 \times k_1n_2$ 2-GMCCMB is constructed from a $k_1n_1 \times k_1n_2$ mesh organization by enhancing the power of each disjoint $n_1 \times n_2$ sub-2-MCC with multiple buses (sub-2-MCCMB). The more detailed description of 2-GMCCMB's will appear in Section II. Given n data, Section III shows that a prefix computation can be performed in $O(n^{1/10})$ time on an $n^{3/5} \times n^{2/5}$ 2-GMCCMB with n processors, where each disjoint sub-2-MCCMB is of size $n^{1/2} \times n^{3/10}$. This time bound is faster than the previous time bound of $O(n^{1/8})$ for the same computation on a rectangular 2-MCCMB of size $n^{5/8} \times n^{3/8}$ [11], [3]. Furthermore, the time bound of our parallel prefix algorithm can be further reduced to $O(n^{1/11})$ if fewer processors are used. Section IV shows that omitting the constant factors, a prefix computation can be performed on a d-dimensional GMCCMB in $O(n^{1/(d2^d+d)})$ time for any constant d. This time bound is less than the previous time bound of $O(n^{1/(d2^d)})$ [11], [3]. Some concluding remarks are addressed in Section V.

II. Generalized 2-D Mesh-Connected Computers with Multiple Buses

In this section, for simplicity we define only the 2-GMCCMB and its properties. As described in Section I, a $k_1n_1 \times k_1n_2$ 2-GMCCMB is constructed from a $k_1n_1 \times k_1n_2$ mesh by providing multiple buses in each $n_1 \times n_2$ sub-MCC. Fig. 2 shows an 8×8 2-GMCCMB with four 4×4 sub-2-MCCMB's, where wrap-around connections are not allowed. We designate the top-leftmost processor in each

Manuscript received February 12, 1993; revised August 11, 1993. The work was supported in part by the National Science Council of R.O.C. under Grant NSC82-0415-E011-180.

The author is with the Department of Information Management, National Taiwan Institute of Technology, Taipei, Taiwan 10672, R.O.C.

IEEE Log Number 9406343.

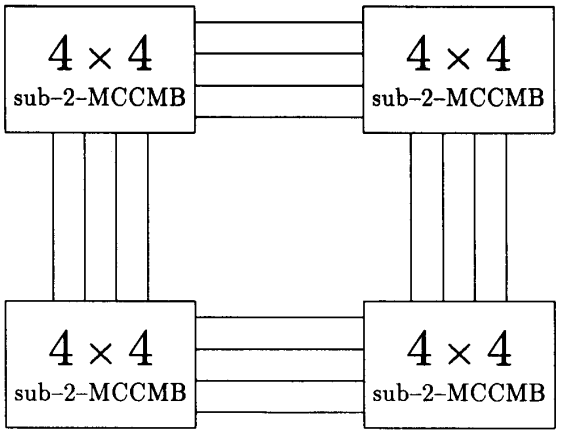

Fig. 2. An 8 × 8 2-GMCCMB with four 4 × 4 sub-2-MCCMB's.

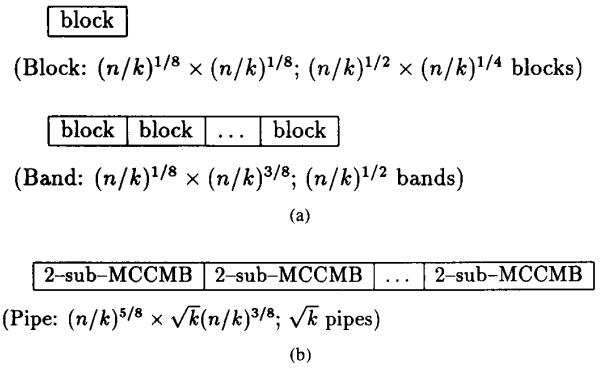

Fig. 3. (a)Blocks and bands of an $(n/k)^{5/8} \times (n/k)^{3/8}$ sub-2-MCCMB. (b) Pipes of an $\sqrt{k}(n/k)^{5/8} \times \sqrt{k}(n/k)^{3/8}$ 2-GMCCMB.

sub-2-MCCMB be the *head* processor. We call the 2-GMCCMB a generalized parallel organization because 2-GMCCMB becomes a $n_1 \times n_2$ 2-MCCMB when selecting $k_1 = 1$; becomes an $k_1 \times k_1$ 2-MCC when selecting $n_1 = n_2 = 1$. In other words, 2-MCC and 2-MCCMB are only two extreme cases of the 2-GMCCMB. In deed, many problems can be solved more efficiently on our generalized model if the values of n_1, k_1, and n_2 are determined properly.

Two types of data routing can be executed by the processors: route data to one of the four nearest neighbors; and broadcast data to a row of processors or a column of processors. At any time only one type of data routing is allowed. It is not difficult to check that the $k_1 n_1 \times k_1 n_2$ 2-GMCCMB has diameter $\Omega(k_1)$. For example, to construct a path of length at most from any processor s in one sub-2-MCCMB to any processor t in another sub-2-MCCMB, we first construct the path of length at most $O(k_1)$ time from s to processor u via row buses or/and horizontal local links, where the column-index of u is equal to the column-index of s. Then we construct the path of length at most $O(k_1)$ time from u to t via column buses or/and vertical local links. Totally, $O(k_1)$ time is required.

In the next section, we will take the prefix computation as a representative of many typical algorithms which utilize simple and identical operations in the sub-2-MCCMB's and use local links between sub-2-MCCMB's to perform complex global task with surprising speed.

III. Prefix Computations on 2-GMCCMB's

Given a set of data $x_1, x_2, \cdots, x_n$ and an associative operator, say $+$, find all partial sums $S_1 = x_1, S_2 = x_1 + x_2, \cdots, S_n = \sum_{i=1}^{n} x_i$. This problem is known as the prefix sum problem.

A. Segmented Prefix Computation on Each Sub-2-MCCMB

We first partition the 2-GMCCMB with n processors into k disjoint sub-2-MCCMB's, thus each sub-2-MCCMB contains n/k processors. Each sub-2-MCCMB will perform its own segmented prefix computation. For example, the first sub-2-MCCMB wants to compute the partial sums S_1, S_2,$\cdots$, and $S_{n/k}$. It has been shown in [11], [3] that using an $(n/k)^{5/8} \times (n/k)^{3/8}$ sub-2-MCCMB, a prefix computation for a data set of length n/k can be performed in $O((n/k)^{1/8})$ time. It first partitions the $(n/k)^{5/8} \times (n/k)^{3/8}$ sub-2-MCCMB into $(n/k)^{3/4}$ disjoint $(n/k)^{1/8} \times (n/k)^{1/8}$ submeshes, called *blocks*. A row of blocks in each sub-2-MCCMB is called a *band*. A row of sub-2-MCCMB's in the 2-GMCCMB is called a *pipe*. Fig. 3.(a) and (b) illustrate the blocks, bands, and pipes.

In the 2-GMCCMB, the pipes are numbered (starting from 1) in a row-major order. In each pipe, the sub-2-MCCMB's are numbered in a row-major order too. Similarly, in each sub-2-MCCMB, the bands and blocks are numbered in a row-major order, respectively. In each block, the processors are also numbered in a row-major order. The input data are initially distributed into the 2-GMCCMB in an increasing sequence of pipe numbers, sub-2-MCCMB numbers, band numbers, then block numbers, finally processor numbers. For example, x_2 is held in (pipe 1, sub-2-MCCMB 1, band 1, block 1, processor 2). Thus, if x_i is held in (pipe p, sub-2-MCCMB s, band a, block b, processor j), S_i depends on the data stored in the first $p-1$ pipes, the first $s-1$ sub-2-MCCMB's in pipe p, the first $a-1$ bands in sub-2-MCCMB s, the first $b-1$ blocks in band a, and the first j processors in block b.

Basically, the parallel segmented prefix algorithm for each sub-2-MCCMB consists of three phases. In the first phase, each $(n/k)^{1/8} \times (n/k)^{1/8}$ block performs its own prefix computation via only local links. There are $(n/k)^{1/8}$ rows in one block. Thus, it takes $O((n/k)^{1/8})$ time to perform the prefix computation for each row in parallel. The *active value* is defined as the sum of elements in each unit. For instance, the active value of a block means the sum of elements in a block. At this moment the rightmost processor in each row saves the sum of those corresponding $(n/k)^{1/8}$ values, called the active value of one row. Similarly, it takes $O((n/k)^{1/8})$ time to perform the prefix computation for those active values of rows in the processors of the rightmost column within one block. Then these $(n/k)^{1/8}$ new active values are shifted to the adjacent processors downward. Finally, associated with the current active value in the rightmost processor, each row takes $O((n/k)^{1/8})$ time to perform the updation to obtain the desired prefix values. It requires $O((n/k)^{1/8})$ to route the active value of each block to the top-leftmost processor of the block, *block-header*, via local links. Totally, $O((n/k)^{1/8})$ time is required to finish the prefix computation for each block in parallel.

In the second phase, each $(n/k)^{1/8} \times (n/k)^{3/8}$ band performs its own prefix computation via row buses and local links. There are $(n/k)^{1/4}$ blocks in one band. Naturally, the $(n/k)^{1/4}$ active values in the block-headers are partitioned into $(n/k)^{1/8}$ *groups* since there are $(n/k)^{1/8}$ row buses for each band. Therefore, each group of size $(n/k)^{1/8}$ is assigned a row bus for broadcasting data. We broadcast the $(n/k)^{1/8}$ active values of each group via the designated row

bus one by one to perform the prefix computation of each group. Simultaneously, the leftmost processors of each row sums up the broadcast values it receives.

From now on, there are $(n/k)^{1/8}$ intermediate values remaining in the leftmost column. Then it takes $O((n/k)^{1/8})$ time to perform the prefix computation for those intermediate values in the leftmost column within one band. These $(n/k)^{1/8}$ new active values are shifted to the adjacent processors downward. Finally, associated with the current active value in the leftmost processor, each row takes $O((n/k)^{1/8})$ time to perform the updation to obtain the desired prefix values. Totally, in this phase, it requires $O((n/k)^{1/8})$ time since each band is performed in parallel.

It requires extra $O(1)$ time to route the active value of each band to the top leftmost-processor of the band, *band-header*, via a row bus, a column bus, and a row bus. That is, first the active value of band i is routed to column i via a row bus. Second the active value is routed to the first row of band i via a column bus. Since all the indices of column buses used are different each other, the routing is collision-free. Then the active value is routed to the top-leftmost processor of band i via a row bus. There are still $(n/k)^{1/2}$ active values to be processed.

In the third phase, the $(n/k)^{1/2}$ active values in the band-headers are partitioned into $(n/k)^{3/8}$ groups since there are $(n/k)^{3/8}$ column buses in the sub-2-MCCMB. Thus each group of size $(n/k)^{1/8}$ is assigned a column bus for broadcasting data. Similar to the second phase, the prefix computation for these $(n/k)^{1/2}$ active values in the band-headers can be accomplished in $O((n/k)^{1/8})$ time. As a result, the segmented prefix computation for each sub-2-MCCMB can be performed in $O((n/k)^{1/8})$ time. It requires extra $O(1)$ time to route the active value of each sub-2-MCCMB to the head processor, via a row bus and a column bus.

B. The Algorithm

Our parallel prefix algorithm on the 2-GMCCMB consisting of three steps is described as follows.

Step 1 (Segmented prefix computation for each sub-2-MCCMB): Perform the prefix computation in parallel for each sub-2-MCCMB. This step consisting of three phases has been described in Section III-A and requires $O((n/k)^{1/8})$ time.

Step 2 (Prefix computation for the active values of all sub-2-MCCMB's): Associated with the active values in the head processors of sub-2-MCCMB's, perform the prefix computation for these active values via row and column buses and local links. This step is similar to the first phase of Step 1 and requires $O(\sqrt{k})$ time.

Step 3 (Updating each sub-2-MCCMB): Perform the updation in parallel for each sub-2-MCCMB via row buses and column buses. This step needs $O(1)$ time.

Based on the above three steps, we have the following theorem.

Theorem 3.1: Given n data, a prefix computation can be performed in $O((n/k)^{1/8} + \sqrt{k})$ time using a $\sqrt{k}(n/k)^{5/8} \times \sqrt{k}(n/k)^{3/8}$ 2-GMCCMB, where each sub-2-MCCMB is of dimensions $(n/k)^{5/8} \times (n/k)^{3/8}$.

We are going to decide the values of k in order to minimize the time bound required.

C. Analysis

By Theorem 3.1, the minimal time bound can be obtained when the equality, $(n/k)^{1/8} = \sqrt{k}$, holds for some k. Thus selecting $k = n^{1/5}$, we have the following theorem.

Theorem 3.2: A prefix computation can be performed in $O(n^{1/10})$ time using an $n^{3/5} \times n^{2/5}$ 2-GMCCMB, where each sub-2-MCCMB is of dimensions $n^{1/2} \times n^{3/10}$.

We consider a variant of the prefix computation above. Assuming that we assign each processor a local memory with size $O(n^{1/11})$. Every processor in the 2-GMCCMB of dimensions $n^{6/11} \times n^{4/11}$ first computes the prefix sums of its own data of size $O(n^{1/11})$ sequentially. Thisstep needs $O(n^{1/11})$ time. At the end of this step, the register of each processor holds its partial sum. For example, the register of the first processor holds the partial sum $S_{n^{1/11}}$ and the register of the last processor holds the partial sum $\sum_{i=n-n^{1/11}+1}^{n} x_i$. From now on, the algorithm is the same as the algorithm described in Section III-B. By Theorem 3.2, it is easy to see that this step also needs $O(n^{1/11})$. Finally, every processor takes $O(n^{1/11})$ time to update its own data of size $O(n^{1/11})$. Then we have the following theorem immediately.

Theorem 3.3: A prefix computation can be performed in $O(n^{1/11})$ time using an $n^{6/11} \times n^{4/11}$ 2-GMCCMB, where each processor has local memory of size $O(n^{1/11})$ and each sub-2-MCCMB is of dimensions $n^{5/11} \times n^{3/11}$.

IV. Extension to d-Dimensional GMCCMB's

We first consider the 3-D GMCCMB. It has been shown in [11], [3] that a prefix computation can be performed in $O(n^{1/24})$ time using an $n^{13/24} \times n^{7/24} \times n^{1/6}$ 3-MCCMB, where the 3-MCCMB is partitioned into blocks of size $n^{1/24} \times n^{1/24} \times n^{1/24}$. Furthermore, we show the result of 3-GMCCMB's as follows.

Theorem 4.1: A prefix computation can be performed in $O(n^{1/27})$ time using an $n^{14/27} \times n^{8/27} \times n^{5/27}$ 3-GMCCMB, where the 3-GMCCMB is partitioned into blocks of size $n^{1/27} \times n^{1/27} \times n^{1/27}$.

Proof: Assuming that the 3-GMCCMB is partitioned into k disjoint sub-3-MCCMB's, and each sub-3-MCCMB is of size n/k. The derivation of our parallel prefix algorithm is a straightforward extension of that described in Section III-B. By the result of 3-MCCMB's in [11], [3], the segmented prefix computation for each sub-3-MCCMB can be accomplished in $O((n/k)^{1/24})$ time using a $k^{1/3}(n/k)^{13/24} \times k^{1/3}(n/k)^{7/24} \times k^{1/3}(n/k)^{1/6}$ 3-GMCCMB, where the 3-GMCCMB is partitioned into blocks of size $(n/k)^{1/24} \times (n/k)^{1/24} \times (n/k)^{1/24}$.

By the similar arguments in Step 2 of the algorithm (see Section III-B), the prefix computation for the active values in the head processors of all sub-3-MCCMB's can be performed in $O(k^{1/3})$ time. It takes $O(1)$ time to update each sub-3-MCCMB via row buses and column buses to obtain the desired prefix values. Totally, it requires $O((n/k)^{1/24} + k^{1/3})$ time. If we select $k = n^{1/9}$, then the equality, $(n/k)^{1/24} = k^{1/3}$, holds. We complete the proof. Q.E.D.

When $d > 3$, our results are mostly of theoretical interest. In general, our result can be extended to the d-GMCCMB. Omitting the constant factors, it has been shown in [11], [3] that a prefix computation can be performed in $O(n^{1/(d2^d)})$ time for any constant d using an $n^{(2^{d-1}d+1)/(d2^d)} \times n^{(2^{d-2}d+1)/(d2^d)} \times \cdots \times n^{(d+1)/(d2^d)}$ d-MCCMB, where the d-MCCMB is partitioned into blocks of size $n^{1/(d2^d)} \times n^{1/(d2^d)} \times \cdots \times n^{1/(d2^d)}$. By the analysis of Theorem 4.1, we have the result of d-GMCCMB as follows.

Theorem 4.2: Using an

$$
\begin{aligned}
& n^{(d2^{d-1}+2)/(d2^d+d)} \\
& \quad \times n^{(d2^{d-2}+2)/(d2^d+d)} \times \cdots \times n^{(d+2)/(d2^d+d)} \\
& \quad (= n^{1/(d2^d+d)+2^d/(2^d+1)(2^{d-1}d+1)/(d2^d)} \\
& \quad \times n^{1/(d2^d+d)+2^d/(2^d+1)(2^{d-2}d+1)/(d2^d)} \times \cdots \\
& \quad \times n^{1/(d2^d+d)+2^d/(2^d+1)(d+1)/(d2^d)}) \ d\text{-GMCCMB}
\end{aligned}
$$

where the d-GMCCMB is partitioned into blocks of size $n^{1/(d2^d+d)} \times$

$n^{1/(d2^d+d)} \times \cdots \times n^{1/(d2^d+d)}$, a prefix computation can be performed in a time bound of $O(n^{1/(d2^d+d)})$ for any constant d; here, we omit the constant factors. This time bound is less than the time bound in [11], [3] on the d-MCCMB.

V. Conclusion

This paper has presented efficient parallel prefix algorithms on newly proposed d-GMCCMB's. Our parallel prefix algorithms are faster than the parallel algorithms for the same computations on d-MCCMB's or d-MCC's.

We now consider problems in solving linear recurrences and tridiagonal systems on our model. It was shown in [15] that the problem of solving linear recurrences can be transformed into the prefix-computation problem. In [26], [27], Stone showed that the problem of solving tridiagonal systems can be transformed into the prefix-computation problem too. Therefore, these two important problems can be solved in $O(n^{1/10})$ on our $n^{3/5} \times n^{2/5}$ 2-GMCCMB. The results are faster than the known algorithms for the same problems on 2-MCCMB's or 2-MCC's.

On $n^{3/8} \times n^{5/8}$ 2-MCCMB's, Chen *et al.* [12], designed a fast median-finding algorithm and its application to two-variable linear programming with time complexity $O(n^{1/8}\log n)$. On the same 2-MCCMB's, Bhagavathi *et al.* [5], designed a fast selection algorithm with the same time complexity. Recently, we have successfully designed a fast selection algorithm with time complexity $O(n^{1/10}\log n)$ on $n^{3/5} \times n^{2/5}$ 2-GMCCMB's [13].

In fact, our results can be applied to many important applications to achieve the better performance. These applications include as comparison between two numbers, data distribution, index computation, observer problem, connected components in image processing, maximum, semigroup computations, median row, lexical analysis, string matching, etc. [24], [16], [6], [7].

Acknowledgment

The author is indebted to the three reviewers for making some valuable suggestions and corrections that lead to the improved version of the paper.

References

[1] A. Aggarwal, "Optimal bounds for finding maximum on array of processors with k global buses," *IEEE Trans. Comput.*, vol. C-35, no. 1, pp. 62–64, Jan. 1986.

[2] M. J. Atallah and S. R. Kosaraju, "Graph problems on a mesh-connected processor array," *J. Assoc. Comput. Mach.*, vol. 31, no. 3, pp. 649–667, July 1984.

[3] A. Bar-Noy and D. Peleg, "Square meshes are not always optimal," *IEEE Trans. Comput.*, vol. C-40, no. 2, pp. 196–204, Feb. 1991.

[4] G. H. Barnes *et al.*, "The Illiac IV computer," *IEEE Trans. Comput.*, vol. C-17, pp. 746–757, 1968.

[5] D. Bhagavathi *et al.*, "Selection on rectangular meshes with multiple broadcasting," *BIT*, vol. 33, no. 1, pp. 7–14, 1993.

[6] G. E. Blelloch, "Scans as primitive operations," *IEEE Trans. Comput.*, vol. C-38, no. 11, pp. 1526–1538, Nov. 1989.

[7] ——, "Prefix sums and their applications," in *Synthesis of Parallel Algorithms*, J. H. Reif, Ed. San Mateo, CA: Morgan Kaufmann, 1993, Chapter 1, pp. 35–60.

[8] S. H. Bokhari, "MAX: An algorithm for finding maximum in an array processor with a global bus," *IEEE Trans. Comput.*, vol. C-33, no. 2, pp. 133–139, Feb. 1984.

[9] D. A. Carlson, "Modified mesh-connected parallel computers," *IEEE Trans. Comput.*, vol. C-37, no. 10, pp. 1315–1321, Oct. 1988.

[10] ——, "Solving linear recurrence systems on mesh-connected computers with multiple global buses," *J. Parallel and Distribut. Comput.*, vol. 8, pp. 89–95, 1990.

[11] Y. C. Chen *et al.*, "Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting," *IEEE Trans. Parallel and Distribut. Syst.*, vol. 1, no. 2, pp. 241–246, Apr. 1990.

[12] Y. C. Chen, W. T. Chen and G. H. Chen, "Efficient median finding and its application to two-variable linear programming on mesh-connected computers with multiple broadcasting," *J. Parallel and Distribut. Comput.*, vol. 15, pp. 79–84, 1992.

[13] K. L. Chung, "Fast selection algorithms on generalized mesh-connected computers with multiple buses," unpublished manuscript.

[14] K. Hwang and F. Briggs, *Computer Architecture and Parallel Processing*. New York: McGraw-Hill, 1984.

[15] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[16] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. Los Altos, CA: Morgan Kaufmann, 1992.

[17] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980.

[18] R. Miller and Q. F. Stout, "Geometric algorithms for digitized pictures on a mesh-connected computer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, pp. 216–228, Mar. 1985.

[19] ——, "Mesh computer algorithms for computational geometry," *IEEE Trans. Comput.*, vol. C-38, no. 3, pp. 321–340, Mar. 1989.

[20] D. Nassimi and S. Sahni, "Bitonic sort on a mesh-connected parallel computer," *IEEE Trans. Comput.*, vol. C-28, no. 1, pp. 2–7, Jan. 1979.

[21] ——, "Finding connected components and connected ones on a mesh-connected parallel computer," *SIAM J. Comput.*, vol. 9, pp. 744–757, Nov. 1980.

[22] ——, "Data broadcasting in SIMD computers," *IEEE Trans. Comput.*, vol. C-30, no. 2, pp. 101–107, Feb. 1981.

[23] D. Parkinson, D. J. Hunt and K. S. MacQueen, "The AMT DAP 500," in *Proc. 33rd IEEE Comput. Conf.*, San Francisco, CA, 1988, pp. 196–199.

[24] V. K. Prasanna Kumar and C. S. Raghavendra, "Array processor with multiple broadcasting," *J. Parallel and Distribut. Comput.*, vol. 2, pp. 173–190, 1987.

[25] I. N. Robinson and W. R. Moore, "A parallel processing array architecture and its implementation in silicon," in *Proc. IEEE Custom Integrated Circuits Conf.*, Rochester, NY, 1982.

[26] H. S. Stone, "An efficient algorithm for the solution of a tridiagonal linear system of equations," *J. Assoc. Comput. Mach.*, vol. 20, no. 1, pp. 27–38, 1973.

[27] ——, "Parallel tridiagonal equation solvers," *ACM Trans. Mathematical Software*, vol. 1, no. 4, pp. 289–307, 1975.

[28] Q. F. Stout, "Mesh-connected computers with broadcasting," *IEEE Trans. Comput.*, vol. C-32, no. 9, pp. 826–830, Sept. 1983.

[29] ——, "Meshes with multiple buses," in *Proc. 27th IEEE Symp. Foundations Comput. Sci.*, pp. 264–273, 1986.

[30] C. D. Thompson and H. T. Kung, "Sorting on a mesh-connected parallel computer," *Commun. Assoc. Comput. Mach.*, vol. 20, pp. 263–271, Apr. 1977.