Designing Efficient Parallel Algorithms on Mesh-Connected Computers with Multiple Broadcasting

YEN-CHENG CHEN, WEN-TSUEN CHEN, SENIOR MEMBER, IEEE, GEN-HUEY CHEN, MEMBER, IEEE, AND JANG-PING SHEU

Abstract—**Semigroup and prefix computations on *two-dimensional mesh-connected computers with multiple broadcasting* (2-MCCMB's) are studied in this paper. Previously, only square 2-MCCMB's with N processing elements were considered for semigroup computations of N data items, and $O(N^{1/6})$ time was required. It is found that square machines are not the best form for semigroup computations, and an $O(N^{1/8})$ time algorithm is thus derived on an $N^{5/8} \times N^{3/8}$ rectangular 2-MCCMB. This time complexity can be further reduced to $O(N^{1/9})$ if fewer PE's are used. Following the same way, parallel algorithms for prefix computations are also derived with the same time complexities.**

Index Terms—**Mesh-connected computers, mesh-connected computers with multiple broadcasting, parallel algorithms, prefix computation, rectangular meshes, semigroup computation.**

I. INTRODUCTION

RECENTLY, great advances in hardware technology have made it possible to design various computer architectures. As a result, some multiprocessor architectures were proposed for parallel processing [21], [10], [11]. Among these, the mesh-connected computers (MCC's) have been widely used because their regular structure and simple interconnection are quite suitable for VLSI implementation. A *k-dimensional mesh-connected computer* (*k-MCC*) of size N contains N processing elements (PE's) arranged in a k-dimensional grid [16]. That is, the PE's may be thought of as logically arranged as in a k-dimensional array $A(n_{k-1}, n_{k-2}, \cdots, n_0)$, where n_i is the size of the ith dimension and $N = n_{k-1} \times n_{k-2} \times \cdots \times n_0$. Very often, $n_{k-1} = n_{k-2} = \cdots = n_0 = N^{1/k}$ [2], [14], [15], [20]. Let $\mathrm{PE}(i_{k-1}, i_{k-2}, \cdots, i_0)$ denote the PE at location $A(i_{k-1}, i_{k-2}, \cdots, i_0)$. $\mathrm{PE}(i_{k-1}, i_{k-2}, \cdots, i_0)$ is connected to $\mathrm{PE}(i_{k-1}, \cdots, i_j \pm 1, \cdots, i_0)$, $0 \le j < k$, provided $\mathrm{PE}(i_{k-1}, \cdots, i_j \pm 1, \cdots, i_0)$ exists. In other words, each PE is connected to its $2k$ nearest neighbors (the PE's along the boundary have fewer neighbors). Fig. 1 shows examples of a 1-MCC and a 2-MCC. As in [2] and [16], it is assumed that a PE may transmit data to any of its nearest neighbors in unit time. Besides, each PE has some local memory.

Manuscript received April 21, 1989; revised November 6, 1989. This work was supported by the National Science Council, Taiwan, R.O.C., under Contract NSC78-0408-E007-11.

Y.-C. Chen and W.-T. Chen are with the Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China.

G.-H. Chen is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, Republic of China.

J.-P. Sheu is with the Department of Electrical Engineering, National Central University, Chung-Li, Taiwan, Republic of China.

IEEE Log Number 8934123.

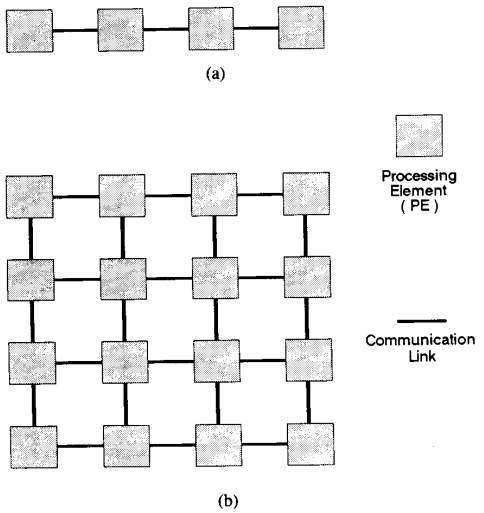


Fig. 1. Mesh-connected computers (MCC's). (a) A 1-MCC of size 4. (b) A 4×4 2-MCC of size 16.

The regular pattern of the mesh-connected network makes MCC's suitable for solving many problems from matrix manipulation and image processing [5], [14]. However, its local connectivity nature will result in a long communication delay when data have to be moved over a long distance. In an $N^{1/2} \times N^{1/2}$ 2-MCC, for example, to move data from one PE to another may take as much as $N^{1/2}$ time in the worst case. Thus, a large diameter is a main shortcoming of the MCC's. Therefore, the execution time of the parallel algorithms that need long distance communications is often dominated by the long communication time. To overcome the communication inefficiency, several authors have augmented the MCC's with various faster mechanisms [1], [3], [4], [6], [12], [17]–[19]. In some of them, the capability of long distance communications is enhanced through the use of one or more broadcast buses. Several variants of MCC's with broadcasting features were thus proposed [1], [4], [12], [18], [19]. Indeed, many problems can be solved more efficiently on these modified MCC's.

This paper considers semigroup and prefix computations on *two-dimensional mesh-connected computers with multiple broadcasting (2-MCCMB's)* [12]. A 2-MCCMB is a two-dimensional mesh-connected computer (2-MCC) with a bus for each row and each column. A detailed description of 2-

1045-9219/90/0400-0241$01.00

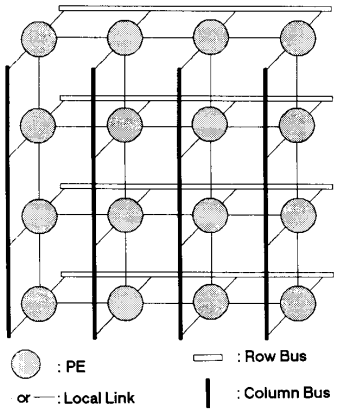


Fig. 2. A 4 × 4 mesh-connected computer with multiple broadcasting (MCCMB).

MCCMB's will appear in the next section. Semigroup computations are an important class of computational problems. Some typical examples are to compute sum, product, maximum/minimum, and Boolean parity. In general, a semigroup computation can be formally described by a tuple $(\oplus, S)$, where $\oplus$ is an associative operator and $S = \{a_0, a_1, \cdots, a_{N-1}\}$ is a set of data items. The problem is to compute $a_0 \oplus a_1 \oplus \cdots \oplus a_{N-1}$. Using an $N^{1/2} \times N^{1/2}$ square 2-MCCMB, Kumar and Raghavendra [12] have proposed an $O(N^{1/6})$ time algorithm for a semigroup computation of N data items, which was declared to be optimal with respect to the square machine. However, we find in Section III that to obtain better performance, the 2-MCCMB's used for semigroup computations are no longer square ones. Instead, an $N^{5/8} \times N^{3/8}$ rectangular 2-MCCMB is used, on which an $O(N^{1/8})$ time algorithm is developed. This time complexity can be further reduced to $O(N^{1/9})$ if fewer PE's are used. It is shown in Section IV that given an arbitrary $X \times Y$ rectangular 2-MCCMB with $X \geq Y$ and $XY = N$, a semigroup computation of N data items can be performed in $O(\max\{Y^{1/3}, N^{1/2}/Y\})$ time. In addition, the generalization of the algorithm to k-dimensional MCCMB's is also discussed; an $O(k^2 N^{1/(k2^k)})$ time algorithm is obtained by using an $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \cdots \times N^{(k+1)/(k2^k)}$ k-MCCMB.

Another problem related to semigroup computation is prefix computation. Given N numbers $a_0, a_1, \cdots, a_{N-1}$, a prefix computation is to compute $a_0 \oplus a_1 \oplus \cdots \oplus a_i$ for $i = 0, 1, \cdots, N-1$, where $\oplus$ is an associative operator. In Section V, a parallel algorithm for prefix computation is developed. Since prefix computation can be considered to be a number of interrelated semigroup computations, the algorithms for prefix computation can be developed by augmenting the ones for semigroup computation with additional computations.

II. Mesh-Connected Computers with Multiple Broadcasting

A *two-dimensional mesh-connected computer with multiple broadcasting (2-MCCMB)* [12] is a two-dimensional mesh-connected computer (2-MCC) with a bus for each row and each column. That is, the PE's in the same row or column are connected to a bus in addition to the local links, as shown in Fig. 2. Hence, 2-MCCMB's have broadcasting capability in each row and each column. These broadcasting features allow parallel data transfers within each row and each column. On a 2-MCCMB, there are two types of data transfers executed by each PE: routing data into one of its four nearest neighbors via a local link and broadcasting data to the other PE's in the same row (or column) via the bus on this row (or column). As in [12] and [19], it is assumed that broadcasting takes constant time and only one PE is permitted to broadcast data on each row bus and each column bus at a time. However, to avoid the overhead caused by conflicts and to handle broadcasting more easily, all algorithms on MCCMB's are required to be conflict-free [12], [19]. In other words, collision resolution schemes are not necessary and algorithms on MCCMB's are developed so that no two PE's "attempt" to use the same bus at the same time. By a straightforward extension, the structure of MCCMB's can be easily extended from two dimensions to higher dimensions [12].

Row and column broadcasting has been shown to be a powerful communication mechanism. Kumar and Raghavendra [12] showed that a binary tree with n leaf nodes can be logically simulated by an $n \times n$ 2-MCCMB through the use of row and column buses. First, these n leaf nodes are arranged in one row (or column) of the 2-MCCMB. For example, suppose that each of the PE's in row 0 (we number rows, columns, row buses, and column buses, all starting from 0) owns a data item initially and the maximum of the n data items is to be computed. The problem can be solved with $\lceil \log n \rceil$ steps by applying a technique called the *simulation tree* technique. That is, in the ith step, $1 \leq i \leq \lceil \log n \rceil$, the intermediate value in PE$(0, (2k+1)2^{i-1})$, $0 \leq k \leq n/2^i - 1$, is sent to PE$(0, 2k2^{i-1})$ via column buses $2k2^{i-1}$, $(2k+1)2^{i-1}$ and row bus k, and then the maximum of two is found and stored in PE$(0, 2k2^{i-1})$. After $\lceil \log n \rceil$ steps, the final result can be obtained in PE$(0, 0)$. Thus, a semigroup computation of n data items can be performed in $O(\log n)$ time on an $n \times n$ 2-MCCMB if these data items are placed in the same row or column. However, this scheme needs too many processing elements and buses, compared to the number of data items. Kumar and Raghavendra [12] also derived another algorithm for a semigroup computation of N data items on an $N^{1/2} \times N^{1/2}$ 2-MCCMB. Initially, each PE holds one data item. The achieved time complexity is $O(N^{1/6})$, which was declared to be optimal. This algorithm can be regarded as a two-phase one. That is, it first partitions the $N^{1/2} \times N^{1/2}$ 2-MCCMB into $N^{2/3}$ disjoint $N^{1/6} \times N^{1/6}$ submeshes, called blocks. In the first phase, the computation result for each block is obtained via local communications. The algorithm then switches over to the second phase, in which the $N^{2/3}$ intermediate values are computed with the aid of row and column broadcasting. For a more detailed description, the interested reader may consult [12]. In addition, when extended to k-dimensional MCCMB's (k-MCCMB's), a semigroup computation of N data items can be performed in $O(N^{1/(k(k+1))})$ time using an $N^{1/k} \times \cdots \times N^{1/k}$ k-MCCMB [12].

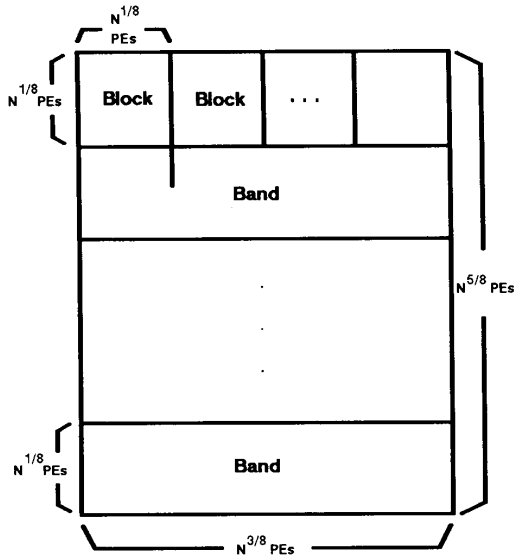


Fig. 3. Partitioning of an $N^{5/8} \times N^{3/8}$ 2-MCCMB.

III. Semigroup Computations on Rectangular 2-MCCMB's

Square 2-MCC's are frequently used in applications. Data are easily mapped onto a square, especially in graph problems [2], image processing [14], and matrix computations [5]. Also, a square 2-MCC has minimal diameter, so that better performance can be obtained for some problems, e.g., semigroup computations. Hence, it is very natural to consider only square 2-MCCMB's in designing parallel algorithms. However, the number of buses in a 2-MCCMB depends on both the number of PE's and their arrangement. Clearly, for the same number of PE's, a rectangular 2-MCCMB has more buses than a square one. This fact suggests the possiblity of improving performance by using rectangular 2-MCCMB's. In this section, we derived a parallel algorithm for semigroup computations on rectangular 2-MCCMB's, which is superior in performance to the best one obtained so far on square 2-MCCMB's. Therefore, to achieve high performance on 2-MCCMB's, square machines are not necessarily the right choice.

It has been shown in [12] that on an $N^{1/2} \times N^{1/2}$ 2-MCCMB, a semigroup computation can be performed in $O(N^{1/6})$ time. However, the time complexity can be further reduced to $O(N^{1/8})$ by using an $N^{5/8} \times N^{3/8}$ 2-MCCMB. The $N^{5/8} \times N^{3/8}$ 2-MCCMB is first partitioned into $N^{3/4}$ disjoint $N^{1/8} \times N^{1/8}$ submeshes, called *blocks*. A row of blocks is called a *band*. Refer to Fig. 3. Initially each PE owns one data item. The algorithm is as follows.

Step 1: Perform the semigroup computation in parallel for each block. Only local links are used and $O(N^{1/8})$ time is required.

The result of each block is held in the upper left PE of the block, called the *block leader*. In each band, there are now $N^{1/4}$ intermediate values. However, the distance between any two is at least $N^{1/8}$. Thus, broadcasting must be exploited to achieve better performance. To use all the row buses in parallel, the intermediate values are duplicated.

Step 2: Copy the intermediate value in each block leader to all the PE's in the same column within the same block via local links. $O(N^{1/8})$ time is required.

Since the number of row buses in each band is $N^{1/8}$, the $N^{1/4}$ intermediate values within a band are partitioned into $N^{1/8}$ groups, each containing $N^{1/8}$ values, so that each group can be assigned a row bus for broadcast communications.

Step 3: Broadcast the $N^{1/8}$ intermediate values of each group in turn via the assigned row bus. In the meantime, the leftmost PE of each row performs the semigroup computation of the broadcast values it receives. Since the broadcasting is performed in parallel for each row bus, $O(N^{1/8})$ time is required.

At the end of step 3, there are still $N^{5/8}$ intermediate values remaining in the leftmost column, one for each PE. Note that each band contains $N^{1/8}$ values of them.

Step 4: Perform the semigroup computation of $N^{1/8}$ intermediate values in parallel for each band via local links. This step takes $O(N^{1/8})$ time.

The computation result of each band is held in the upper left PE of the band, called the *band leader*. Totally, $N^{1/2}$ intermediate values remain. In order to use all the column buses in parallel, duplicating values via row broadcasting is required.

Step 5: Copy the intermediate value in each band leader to all the PE's in the same row via broadcasting. This takes constant time.

The remaining $N^{1/2}$ intermediate values are partitioned into $N^{3/8}$ groups each containing $N^{1/8}$ intermediate values so that each group can share a column bus.

Step 6: Perform the semigroup computation in parallel for each group via column broadcasting. $O(N^{1/8})$ time is required.

At the end of the step, $N^{3/8}$ intermediate values remain. The computation results are held in the uppermost row, one for each PE.

Step 7: Perform the semigroup computation of the remaining $N^{3/8}$ intermediate values by using the simulation tree technique. Therefore, the final result can be found in the upper left PE of the 2-MCCMB. This step takes O (log N) time.

According to the above description, we have the following theorem.

Theorem 3.1: A semigroup computation of N data items can be performed in $O(N^{1/8})$ time using an $N^{5/8} \times N^{3/8}$ 2-MCCMB.

One way to further reduce the time complexity of the algorithm is to use fewer PE's. This can be described in the following theorem.

Theorem 3.2: A semigroup computation of N data items can be performed in $O(N^{1/9})$ time using an $N^{5/9} \times N^{1/3}$ 2-MCCMB with each PE containing $N^{1/9}$ data items initially.

The proof of the theorem is omitted for sake of brevity. The interested reader may consult [8].

IV. Discussion

A. Extension to Arbitrary Rectangular 2-MCCMB's

The algorithm proposed in Section III is a two-phase algorithm. In the first phase, data communications are via only

local links. When the distance between any two intermediate values becomes greater than or equal to $N^{1/8}$, the algorithm enters the second phase and starts to use row and column broadcasting for efficient communications. The time complexity achieved in Theorem 3.1 is due to the selection of the switchover point. That is, $N^{1/8} \times N^{1/8}$ is the optimal block size for the two-phase algorithm on the $N^{5/8} \times N^{3/8}$ 2-MCCMB. It will be seen that the optimal block size depends on the form of the rectangular 2-MCCMB. In the following, we show how to determine the block size for an arbitrary rectangular 2-MCCMB. It can be also seen that the achieved time complexity depends on the form of the rectangular 2-MCCMB. Indeed, $N^{5/8} \times N^{3/8}$ is the optimal form of rectangular 2-MCCMB's for semigroup computations.

Suppose that an $X \times Y$ 2-MCCMB with $X \geq Y$ and $XY = N$ is used for a semigroup computation of N data items. Let the block size be $m \times m$. The previous algorithm can be gerneralized as follows.

Step 1: Perform the semigroup computation in parallel for each block by using local links. This takes $O(m)$ time.

In each band, there are now Y/m intermediate values.

Step 2: Copy the intermediate value in each block leader to all the PE's in the same column within the same block via local links. *O(m)* time is required.

Since the number of row buses in each band is m, the Y/m intermediate values within a band are partitioned into m groups each containing Y/m^2 values.

Step 3: Broadcast the Y/m^2 intermediate values of each group in turn via the assigned row bus. In the meantime, the leftmost PE of each row performs the semigroup computation of the broadcast values it receives. This step requires $O(Y/m^2)$ time.

There are now X values remaining in the leftmost column; each band contains m values in its leftmost block.

Step 4: Perform the semigroup computation of m intermediate values for each band via local links. This step takes $O(m)$ time.

Now, X/m values remain; each band leader contains one of them.

Step 5: Copy the intermediate value in each band leader to all the PE's in the same row via row broadcasting. This takes constant time.

The remaining X/m values are partitioned into Y groups each containing $X/(Ym)$ values.

Step 6: Perform the semigroup computation in parallel for each group via column broadcasting. $O(X/(Ym)) = O(N/(Y^2m))$ time is required.

After step 6, each PE in the uppermost row owns an intermediate value.

Step 7: Perform the semigroup computation of the remaining Y values by using the simulation tree technique. This step takes O (log Y) time.

The total time complexity is O (max $\{m, Y/m^2, N/(Y^2m),$ log $Y\}$), which is a function of m. The minimum is O (max $\{Y^{1/3}, N^{1/2}/Y\}$) when $m =$ max $\{Y^{1/3}, N^{1/2}/Y\}$. Thus, we have the following theorem.

Theorem 4.1: Given an $X \times Y$ 2-MCCMB with $X \geq Y$ and $XY = N$, a semigroup computation of N data items can be performed in O (max $\{Y^{1/3}, N^{1/2}/Y\}$) time.

Since max $\{Y^{1/3}, N^{1/2}/Y\}$ has the minimum value $N^{1/8}$ when $Y = N^{3/8}$, we have the following theorem.

Theorem 4.2: The optimal form of 2-MCCMB's with size N is $N^{5/8} \times N^{3/8}$ when the two-phase algorithm is used to perform a semigroup computation of N data items. The time complexity is $O(N^{1/8})$.

B. Extension to k-Dimensional MCCMB's

The results obtained in Section III and Section IV-A can be further extended to k-dimensional MCCMB's (k-MCCMB's). For example, a semigroup 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 first partitioned into blocks of size $N^{1/24} \times N^{1/24} \times N^{1/24}$ [7]. It is not difficult to extend the result to higher dimensions. We describe the generalized result on k-MCCMB's as follows.

Theorem 4.3: A semigroup computation of N data items can be performed in $O(k^2N^{1/(k2^k)})$ time using an $N^{(2^{k-1}k+1)/(k2^k)} \times N^{(2^{k-2}k+1)/(k2^k)} \times \cdots \times N^{(k+1)/(k2^k)}$ k-MCCMB, $k \geq 1$. The block size is

$$\overbrace{N^{1/(k2^k)} \times N^{1/(k2^k)} \times \cdots \times N^{1/(k2^k)}}^{k}.$$

Following the same idea, the algorithm performing the semigroup computation on the k-MCCMB can be derived with some effort [7]. Since the detailed description about the algorithm is very tedious, we omit it here.

For an arbitrary given k-MCCMB, the time required for a semigroup computation can also be determined. For example, $O(N^{1/4})$ time is required for an $N^{3/4} \times N^{1/4}$ 2-MCCMB, and $O(N^{1/18})$ time is required for an $N^{4/9} \times N^{3/9} \times N^{2/9}$ 3-MCCMB. This can be stated formally as follows.

Theorem 4.4: Consider an $A_{k-1} \times A_{k-2} \times \cdots \times A_0$ k-MCCMB, $k \geq 1$, with $A_{k-1} \geq A_{k-2} \geq \cdots \geq A_0$ and $A_{k-1} \times A_{k-2} \times \cdots \times A_0 = N$. Initially, each PE contains one data item. The time complexity achieved by the two-phase algorithm is $O(\max\{A_0^{1/(k+1)}, (A_1/A_0)^{1/k}, (A_2/(A_1A_0))^{1/(k-1)}, \cdots, (A_{k-1}/(A_{k-2} \times A_{k-3} \times \cdots \times A_0))^{1/2}\})$.

The derivation of the two-phase algorithm is a straightforward extension of that described in Section IV-A. Note that the minimal time complexity occurs when

$$\begin{aligned} &O(A_0^{1/(k+1)}) \\ &= O((A_1/A_0)^{1/k}) \\ &= O((A_2/(A_1A_0))^{1/(k-1)}) \\ &\quad \vdots \\ &= O((A_{k-1}/(A_{k-2} \times A_{k-3} \times \cdots \times A_0))^{1/2}). \end{aligned}$$

Therefore, the time complexity stated in Theorem 4.3 is the minimal one for the two-phase algorithm on arbitrary k-MCCMB's.

V. Algorithm for Prefix Computation

Let $S_i = a_0 \oplus a_1 \oplus \cdots \oplus a_i$. A prefix computation is to compute all S_i's, $0 \leq i \leq N-1$. Also let $S_{i,j} = a_i \oplus a_{i+1} \oplus \cdots \oplus a_j$, $i \leq j$, which is called *local prefix from a_i to a_j*. Clearly, $S_j = S_{0,j} = S_{0,i_1} \oplus S_{i_1+1,i_2} \oplus \cdots \oplus S_{i_M+1,j}$, where 0

$< i_1 < i_2 < \cdots < i_M < j$. In our algorithms each S_j is obtained through continuous merging of adjacent local prefixes. As discussed in previous sections, the PE's in a 2-MCCMB are grouped into *blocks, groups,* and *bands.* Local prefix computations are performed simultaneously within each block, each group, and each band. The blocks are numbered (starting from 0) in a row-major order. Also, within each block, the PE's are numbered in a row-major order. The input data items $a_0, a_1, \cdots, a_{N-1}$ are distributed into the 2-MCCMB in an increasing sequence of block numbers and in an increasing sequence of PE numbers when they are within the same block. Thus, if a_i is held in PE j of block k, S_i depends on the data items stored in the first k blocks and the first $j + 1$ PE's of block k. The algorithms are to generate S_i in the PE holding a_i. According to the data distribution, the algorithm consists of the following four major steps.

Step 1: Compute $S_{i_b,j}$ in parallel for each block, where a_j is held in the block and $i_b = \min \{j | a_j$ is held in the block$\}$.

Step 2: Compute $S_{i_g,j}$ in parallel for each group, where a_j is held in the group and $i_g = \min \{j | a_j$ is held in the group$\}$.

Step 3: Compute $S_{i_{bd},j}$ in parallel for each band, where a_j is held in the band and $i_{bd} = \min \{j | a_j$ is held in the band$\}$.

Step 4: Compute S_j, $0 \le j \le N - 1$.

It is not difficult to derive an $O(N^{1/6})$ time algorithm performing a prefix computation of N data items on an $N^{1/2} \times N^{1/2}$ 2-MCCMB, with block size $N^{1/6} \times N^{1/6}$, group size $N^{1/6} \times N^{1/3}$, and band size $N^{1/6} \times N^{1/2}$. Considering rectangular 2-MCCMB's, we use an $N^{5/8} \times N^{3/8}$ 2-MCCMB for prefix computations. The block size is $N^{1/8} \times N^{1/8}$. $N^{1/8}$ continguous blocks form a group and a row of blocks form a band. In addition, we define a *band-group* to be $N^{1/8}$ contiguous bands. The algorithm first computes local prefixes for each block (i.e., Step 1). This step takes $O(N^{1/8})$ time. At Step 2, in $O(N^{1/8})$ time, each block in a group can obtain, by the aid of a series of data broadcasting, the result of the semigroup computation of all the data items that are contained in its preceding blocks of the same group. Furthermore, in additional $O(N^{1/8})$ time, the obtained result is propagated to each PE of the same block. Then, the local prefixes for each group are computed in constant time. Like Step 2, Step 3 can be performed in $O(N^{1/8})$ time. At the end of Step 3, local prefixes for each band are obtained. Then, each band-group is assigned a column bus and local prefixes for each band-group are determined in $O(N^{1/8})$ time. Finally, by the use of a simulation tree, all S_j's can be computed in O (log N) time. Since each of the above four steps takes $O(N^{1/8})$ time, we have the following theorem.

Theorem 5.1: A prefix computation of N data items can be performed in $O(N^{1/8})$ time on an $N^{5/8} \times N^{3/8}$ 2-MCCMB.

As in Theorem 3.2, if fewer PE's are used for prefix computation, an $O(N^{1/9})$ time algorithm can be obtained.

Theorem 5.2: A prefix computation of N data items can be performed in $O(N^{1/9})$ time on an $N^{5/9} \times N^{1/3}$ 2-MCCMB with each PE containing $N^{1/9}$ data items initially.

VI. Concluding Remarks

In this paper, it is shown that square 2-MCCMB's are not favorable for semigroup computations and prefix computations. We have discussed semigroup computations and prefix computations on rectangular 2-MCCMB's, and were able to achieve better preformance than on square 2-MCCMB's. Algorithms with time complexity $O(N^{1/8})$ were developed on an $N^{5/8} \times N^{3/8}$ 2-MCCMB, versus $O(N^{1/6})$ on an $N^{1/2} \times N^{1/2}$ 2-MCCMB.

On $N^{1/2} \times N^{1/2}$ 2-MCCMB's, Kumar and Raghavendra [12] have developed algorithms for the median row problem and the median problem with time complexity $O(N^{1/6})$ and $O(N^{1/6} \times \log^{2/3} N)$, respectively. Both semigroup computations and prefix computations can be regarded as basic operations for solving these two problems. We have successfully developed algorithms for the median row problem (and the median problem) with time complexity $O(N^{1/8})$ ($O(N^{1/8} \times \log N)$) also on $N^{5/8} \times N^{3/8}$ 2-MCCMB's [9].

References

[1] A. Aggarwal, "Optimal bounds for finding maximum on array of processors with k global buses," *IEEE Trans. Comput.,* vol. C-35, 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] S. H. Bokhari, "MAX: An algorithm for finding maximum in an array processor with a global bus," in *Proc. Int. Conf. Parallel Processing,* Aug. 1981, pp. 302–303.

[4] ——, "Finding maximum on an array processor with a global bus," *IEEE Trans. Comput.,* vol. C-33, pp. 133–139, Feb. 1984.

[5] P. R. Cappello, "A mesh automaton for solving dense linear systems," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 418–425.

[6] D. A. Carlson, "Performing tree and prefix computations on modified mesh-connected parallel computers," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 715–718.

[7] Y. C. Chen and G. H. Chen, "Parallel algorithms on mesh-connected computers with multiple broadcasting," unpublished manuscript.

[8] Y. C. Chen, W. T. Chen, G. H. Chen, and J. P. Sheu, "Reducing time complexties of semigroup computations on mesh-connected computers with multiple broadcasting," in *Proc. Int. Conf. Parallel Processing,* Vol. III, Aug. 1989, pp. 234–241.

[9] Y. C. Chen, W. T. Chen, and G. H. Chen, "Finding median and median row on mesh-connected computers with multiple broadcasting," to be published.

[10] T. Y. Feng, "A survey of interconnection networks," *IEEE Comput. Mag.,* pp. 12–27, Dec. 1981.

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

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

[13] S. P. Levitan, "Algorithms for a broadcast protocol multiprocessor," in *Proc. 3rd Int. Conf. Distribut. Comput. Syst.,* 1982, pp. 666–671.

[14] 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, Mar. 1985.

[15] ——, "Varying diameter and problem size in mesh-connected computers," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 697–699.

[16] D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Trans. Comput.,* vol. C-30, p. 101–107, Feb. 1981.

[17] Q. F. Stout, "Broadcasting in mesh-connected computers," in *Proc. 1982 Conf. Info. Sci. Syst.,* Princeton Univ., Princeton, NJ, 1982, pp. 85-90.

[18] ——, "Mesh-connected computers with broadcasting," *IEEE Trans. Comput.,* vol. C-32, pp. 826–830, Sept. 1983.

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

[20] C. D. Thompson and H. T. Kung, "Sorting on a mesh-connected computer," *Commun. Assoc. Comput. Mach.,* vol. 20, no. 4, pp. 263-271, Apr. 1977.

[21] L. D. Wittie, "Communication structures for large networks of microcomputers," *IEEE Trans. Comput.,* vol. C-30, no. 4, pp. 264–273, Apr. 1981.

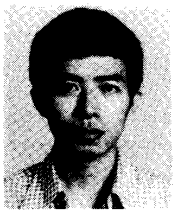
Yen-Cheng Chen received the B.S. degree in electrical engineering from National Cheng Kung University in 1986 and the M.S. degree in information engineering from Tatung Institute of Technology in 1988.

Since September 1988, he has been working towards the Ph.D. degree in the Institute of Computer Science of the National Tsing Hua University, Taiwan, Republic of China. His research interests are parallel algorithms and multiprocessor systems.

Wen-Tsuen Chen (M'87–SM'89) was born in Taiwan, Republic of China, on May 27, 1948. He received the B.S. degree in nuclear engineering from National Tsing-Hua University, Taiwan, Republic of China, and the M.S. and Ph.D. degrees both from University of California, Berkeley, in 1970, 1973, and 1976, respectively.

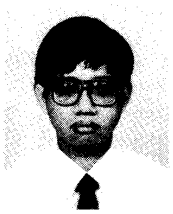
He joined the faculty of the Institute of Computer Science, National Tsing-Hua University in March 1976 as an Associate Professor. Since 1979, he has been a Professor and from 1983 to 1988 he served as the Director of the Institute. In 1980, he was a visiting professor in the EECS Department of the University of California, Berkeley. From 1984 to 1985, he was elected as an IEEE Distinguished Visitor in region 10. Since 1988, he has been a member of the Science and Technology Advisory Board of the Ministry of Education, Taiwan, Republic of China. His current research interests include computer networks, broadband ISDN, multiprocessing systems, parallel algorithms.

Dr. Chen is a member of the Association for Computing Machinery.



Gen-Huey Chen (M'88) was born in Taiwan, on October 10, 1959. He received the B.S. degree in computer science from National Taiwan University, Taiwan, in June, 1981 and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University in June 1983 and January 1987, respectively.

In February 1987, he joined the faculty of National Taiwan University and he now is an Associate Professor of the Department of Computer Science and Information Engineering. His current research interests include design and analysis of algorithms, distributed algorithms, parallel computation, and parallel computer architectures.



Jang-Ping Sheu (S'85–M'86) was born in Taiwan, Republic of China, on April 28, 1959. He received the B.S. degree in computer science from Tamkang University, Taiwan, in 1981 and the M.S. and Ph.D. degrees in computer science from the National Tsing Hua University, Taiwan, in 1983 and 1987, respectively.

He joined the faculty in the Department of Electrical Engineering at the National Central University, Taiwan, as an Associate Professor in 1987. His current research interests include distributed computing systems and parallel processing.