Semigroup and Prefix Computations on Improved Generalized Mesh-Connected Computers with Multiple Buses

Yi Pan *, S.Q. Zheng †, Keqin Li ‡, and Hong Shen §

* Dept. of Computer Science, University of Dayton, Dayton, OH 45469-2160
Phone: (937) 229-3807, Fax: (937) 229-4000, Email: pan@cps.udayton.edu
† Dept. of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688
‡ Dept. of Mathematics & Computer Science, State University of New York, New Paltz, NY 12561-2499
§ School of Computing and Information Technology, Griffith University, Nathan, QLD 4111, Australia

Abstract

Various augmenting mechanisms have been proposed to enhance the communication efficiency of mesh-connected computers (MCC's). One major approach is to add non-configurable buses for improved broadcasting. A typical example is the mesh-connected computer with multiple buses (MMB). In this paper, we propose a new class of generalized MMB's, the improved generalized MMB's (IMMB's). Each processor in an IMMB is connected to exactly two buses. We show the power of IMMB's by considering semigroup and prefix computations. Specifically, we show that semigroup and prefix computations on N operands, and data broadcasting all take $O(\log N)$ time on IMMB's. This is the first $O(\log N)$ time algorithm for these problems on arrays with fixed broadcasting buses.

Keywords: bus, mesh-connected computer, mesh-connected computer with multiple buses, parallel algorithm, parallel architecture, parallel computing, processor array.

1 Introduction

Due to its simple and regular interconnection pattern, a mesh-connected computer (MCC) is feasible for hardware implementation and suitable for solving many problems such as matrix manipulation and image processing. However, the relatively large diameter of an MCC causes a long communication delay between processors that are far apart. The time complexities of algorithms running on an MCC are lower bounded by its diameter. To overcome this problem, various augmenting mechanisms have been proposed to enhance the communication efficiency of MCC's. One major approach is to add buses for improved broadcasting [2, 3, 6, 7, 9, 10, 11]. A typical example is the mesh-connected computer with multiple broadcasting (MMB)[9]. A two-dimensional MMB is a two-dimensional (2-D) MCC with a bus for each row and each column.

In this paper, we propose a class of improved generalized mesh-connected computers with multiple buses (IMMB). We compare the performances of IMMB's and MMB's by considering parallel semigroup and prefix computations on these architectures. Like MMB's of [9] and GMMB's of [6], buses of the IMMB's proposed in this paper do not have switches on them. The major difference between our IMMB architectures and existing MMB-like architectures is that the buses in our architectures are partitioned into levels, while maintaining that each processor is connected to exactly two orthogonal buses. In an l-level IMMB, buses are partitioned into l levels of different spans. The diameter of a d-dimensional l-level IMMB (called (d,l)-IMMB) is dl. In Section 2, we define the two-dimensional IMMB's. In Section 3, we show that semigroup and prefix computations on N operands can be performed using an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square $(2,2)$-IMMB in $O(N^{\frac{1}{16}})$ time. We would like to point out that a $(2,2)$-IMMB can simulate its corresponding GMMB proposed in [6] with a constant factor of slowndown, while having fewer buses. In terms of number and size of buses, an IMMB is a trade-off of an MMB and a GMMB. The performance of of an IMMB is better than that of an MMB and GMMB. Further performance improvement can be achieved by increasing the number of levels. In Section 4, we show that for any constant $0 < \epsilon < 1$, there exists a multi-level $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square IMMB using which semigroup and prefix computations on N operands can be carried out in $O(N^{\epsilon})$ time, while maintaining $O(1)$ broadcasting time. We also show how to construct an l-level $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square IMMB, where $l = O(\log N)$, on which semigroup and prefix computations on N operands, and data broadcasting all take $O(\log N)$ time.

2 Two-Dimensional IMMB's

A two-dimensional IMMB is a two-dimensional mesh-connected computer (MCC) augmented with buses. We call the links for the mesh connections *local links*. The added buses are divided into l levels, which form a hierarchical structure. A 2-D IMMB is formally defined as follows.

An $I(1,(n_{1,1},n_{1,2}))$, a one-level IMMB, is an $n_{1,1}\times n_{1,2}$ MMB. Processors that are in the boundary rows and columns are called *boundary processors*. An $I(l,(n_{1,1}, n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l,1},\ n_{l,2}))$, where $l>1$, is an l-level IMMB that is constructed by arranging $n_{l,1}n_{l,2}$ copies of $I(l-1,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l-1,1},\ n_{l-1,2}))$ as an $n_{l,1}\times n_{l,2}$ array. The processors in each copy of $I(l-1,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l-1,1},\ n_{l-1,2}))$ are collectively called a *level-$(l-1)$ submesh*. Local links are added to connect boundary processors of level-$(l-1)$ submeshes to enforce nearest-neighbor mesh connections. For easy references, these local links are referred to as *level-l bridge local links*. For each row (resp. column) of the $n_{l,1}\times n_{l,2}$ array of level-$(l-1)$ submeshes, we do the following: Merge the top most (resp. left most) row (column) buses, which were level-$(l-1)$ buses, of these IMMB's into one bus. The buses obtained by these merging operations are called the level-l buses of $I(l,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l,1},\ n_{l,2}))$, and they are no longer level-$(l-1)$ buses. The remaining level-k buses, $1\le k\le l-1$, of the $n_{l,1}\times n_{l,2}$ component level-$(l-1)$ submeshes are called the level-k buses of $I(l,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l,1},\ n_{l,2}))$. An l-level two-dimensional IMMB is also referred to as a $(2,l)$-IMMB. To avoid degeneracy, we assume that $n_{i,1}\ge 3$ and $n_{i,2}\ge 3$ for $1\le i\le l$. Define $b_1=n_{1,1}n_{1,2}$, and $b_i=n_{i,1}n_{i,2}b_{i-1}-n_{i,1}(n_{i,2}-1)-n_{i,2}(n_{i,1}-1)=n_{i,1}n_{i,2}(b_{i-1}-2)+n_{i,1}+n_{i,2}$ for $1<i\le l$. Clearly, b_l is the number of buses in $I(l,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l,1},\ n_{l,2}))$.

In an $I(l,(n_{1,1},n_{1,2}),\ (n_{2,1},\ n_{2,2}),\cdots,\ (n_{l,1},\ n_{l,2}))$, there are $N=\prod_{i=1}^{l}(n_{i,1}n_{i,2})$ processors arranged as a $(\prod_{i=1}^{l}n_{i,1})\times(\prod_{i=1}^{l}n_{i,2})$ processor array. It contains a $(\prod_{i=1}^{l}n_{i,1})\times(\prod_{i=1}^{l}n_{i,2})$ MCC (connected by local links) as a substructure; i.e. after removing buses, we obtain an MCC. In addition to local links, each processor is connected to exactly two buses. Each bus belongs to a unique level. It is important to note that a level-k bus, $k>1$, is shared by processors from several level-$(k-i)$ submeshes, where $1\le i\le k-1$, so it cannot support concurrent data transmissions among the level-$(k-i)$ submeshes connected by it. In some situations, as will be shown shortly, concurrent data transmissions on such a bus can be simulated using other buses with only a small constant slowdown factor.

Our two-level IMMB's closely resemble the GMMB's proposed in [6], which are constructed using multiple copies of MMB by only introducing additional local links. Define an *$r\times s$ logical MMB* on a subset V of rs processors of an IMMB as a substructure of the IMMB that can simulate each parallel step of an $r\times s$ MMB, with its processors being in V, in constant time. We can use a $(2,2)$-IMMB, $I(2,(n_{1,1},n_{1,2}),(n_{2,1},n_{2,2}))$, to simulate its corresponding 2-D GMMB of [6] as follows. In each level-1 submesh, we use the level-1 bus that connects the second processor row (resp. column) to simulate a level-1 bus that connects all processors in the first row (resp. column). For example, suppose that processor $P_{i,j}$ in the first row (resp. column) of a level-1 submesh wants to broadcast a message to all processors in the same row (resp. column) of the same submesh. It can first send the message to processor $P_{i+1,j}$ (resp. $P_{i,j+1}$) in the second row (resp. column) using a local link, and then broadcast it to all processors in that row (resp. column) using the level-1 bus that connects them. After this, each processor in the second row, (resp. column) sends the received message to the corresponding processor in the first row via a local link. This scheme implies that there exists $n_{2,1}n_{2,2}$ disjoint $n_{1,1}\times n_{1,2}$ logical MMB's defined on the level-1 submeshes, and each $n_{1,1}\times n_{1,2}$ MMB substructure of the 2-D GMMB can be simulated by a logical MMB. This leads to the following claim.

Theorem 1 *A $(2,2)$-IMMB can simulate its corresponding 2-D GMMB with a constant-factor slowdown.*

Obviously, the converse of this theorem is not true. This is because that the diameter of the 2-D GMMB corresponding to $I(2,(n_{1,1},n_{1,2}),(n_{2,1},n_{2,2}))$ is $2(n_{2,1}+n_{2,2}-1)$, whereas the diameter of $I(2,(n_{1,1},n_{1,2}),(n_{2,1},n_{2,2}))$ is 4. We wlll show that IMMB's are more powerful than MMB's and GMMB's by considering semigroup and prefix computations.

Let us compare the structures of two dimensional MMB, GMMB and IMMB of the same size. A common feature of MMB, GMMB and IMMB is that all of them contain an MCC as a substructure, and each processor is connected to exactly two buses. Define the size of a bus as the number of processors connected by the bus. All buses in an MMB and a GMMB are of the same size, whereas the buses in an IMMB have variable sizes. The largest bus size of a GMMB is the same as the size of buses in an MMB. In terms of semigroup and prefix computations, a GMMB improves the performance of an MMB by adding more buses, and IMMB improves the performance of a GMMB by allowing variable bus sizes. It is important to note that the improved performance of an IMMB over a GMMB can even be achieved by using a smaller buses as in the case of two-level IMMBs.

3 Semigroup and Prefix Computations on a $(2,2)$-IMMB

In this section, we consider semigroup and prefix computations using a $(2,2)$-IMMB, a 2-D two-level IMMB. A semigroup computation is formally defined by a tuple $(\oplus,S)$, where $\oplus$ is an associative operator, and $S=$

$\{a_1, a_2, \cdots, a_N\}$ is a set of operands. This tuple specifies computation $a_1 \oplus a_2 \oplus \cdots \oplus a_N$. A prefix computation is also defined by a tuple $(\oplus, S)$, where $\oplus$ is an associative operator, and $S = (a_1, a_2, \cdots, a_N)$ is a sequence of operands. This tuple specifies computations $s_i = a_1 \oplus a_2 \oplus \cdots \oplus a_i$ for $1 \leq i \leq N$. We assume that the operation $\oplus$ performed on two operands takes constant time. Since the result s_N of a prefix computation is a result of a semigroup computation, any algorithm for prefix computations can be used for a semigroup computation with the same complexity. Thus, we only need to discuss prefix computations.

By Theorem 1 and the result of [6] on GMMB's, semigroup and prefix computations on N operands can be done using $I(2, (N^{\frac{1}{2}}, N^{\frac{3}{10}}), (N^{\frac{1}{10}}, N^{\frac{1}{10}}))$ in $O(N^{\frac{1}{10}})$ time. By further exploiting the power of IMMB's, we obtain better results.

Consider $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ such that $n_{1,1} = n_{2,2} = n_1$ and $n_{1,2} = n_{2,1} = n_2$. Clearly, $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ consists of $N = n^2$ processors arranged as an $n \times n$ square array, where $n = n_1 n_2$. In another view, $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ is an $n_2 \times n_1$ array of $n_1 \times n_2$ level-1 submeshes. We call the processor in such a submesh that has the largest index according to lexicographical order the *leader of the submesh*. We observe that the leaders of level-1 submeshes are processors P_{in_1, jn_2}, where $1 \leq i \leq n_2$ and $1 \leq j \leq n_1$, and they form an $n_2 \times n_1$ array. It is simple to see that the leaders P_{in_1,kn_2} and $P_{in_1,(k+1)n_2}$ (resp. P_{kn_1,jn_2} and $P_{(k+1)n_1,jn_2}$) of two adjacent level-1 submeshes are connected by the following path:

$$P_{in_1,kn_2} \overset{bridge\ link}{\longrightarrow} P_{in_1,kn_2+1} \overset{level-1\ bus}{\longrightarrow} P_{in_1,(k+1)n_2}$$

$$(P_{kn_1,jn_2} \overset{bridge\ link}{\longrightarrow} P_{kn_1+1,jn_2} \overset{level-1\ bus}{\longrightarrow} P_{(k+1)n_1,jn_2})$$

Furthermore, the k-th level-2 row (resp. column) bus can be used to simulate a bus that connects k-th leader row (column). For example, suppose that a submesh leader wants to broadcast a message to all leader processors in its row (resp. column). It can send the message to the processor in the first row (resp. column) of its submesh via the level-1 column (resp. row) bus it is connected to, and use the level-2 bus to broadcast the message to all processors in this row. Then, processors in this row (resp. column) send the received messages to their corresponding processors (leaders) in the last row (resp. column) of the submesh via level-1 column (resp. row) buses. Thus, $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ contains a logical $n_2 \times n_1$ MMB defined on the level-1 leaders.

Let $(a_1, a_2, \cdots, a_n)$ be a sequence of $n = n_1 \times n_2$ operands for a prefix computation, and $\mathcal{A}$ be a prefix algorithm that runs in $O(t(n))$ time on an $n_1 \times n_2$ MMB with each processor holding one operand. Suppose that

$$f : \{i | 1 \leq i \leq n\} \rightarrow \{(j,k) | 1 \leq j \leq n_1, 1 \leq k \leq n_2\}$$

is the function used by algorithm $\mathcal{A}$ to map a_i's and s_i's to processors; i.e. if $f(i) = (j,k)$, input a_i and result $s_i = a_1 \oplus a_2 \oplus \cdots \oplus a_i$ are in $P_{j,k}$ before and after the computation, respectively. We want to perform prefix computation on $(a_1, a_2, \cdots, a_{n^2})$ using $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$. We partition $(a_1, a_2, \cdots, a_{n^2})$ into n subsequences A_i, $1 \leq i \leq n$, each having n operands, such that $A_i = (a_{(i-1)n+1} a_{(i-1)n+2} \cdots, a_{in})$. Recall that $I(2, (n_{2,1}, n_{2,2}), (n_{1,1}, n_{1,2}))$ consists of an $n_2 \times n_1$ array of level-1 submeshes, each being an $n_1 \times n_2$ processor array. Denote these submeshes as $M_{k,j}$'s, where $1 \leq k \leq n_2$ and $1 \leq j \leq n_1$. Submesh $M_{k,j}$ consists of processors $P_{a,b}$, $(k-1)n_1 + 1 \leq a \leq kn_1$ and $(j-1)n_2 + 1 \leq j \leq jn_2$. Define

$$g : \{i | 1 \leq i \leq n\} \rightarrow \{(k,j) | 1 \leq j \leq n_1, 1 \leq k \leq n_2\}$$

such that $g(i) = (k,j)$ if $f(i) = (j,k)$. We use $g(i)$ to map A_i's to $M_{k,j}$'s. For each A_i, we map its $a_{(i-1)n+q}$ to processor P_{j_q,k_q}, where $1 \leq q \leq n$, and $(j_q, k_q) = f(q)$. In other words, initially processor $P_{(k-1)n_1+j', (j-1)n_2+k'}$, which is in submesh $M_{k,j}$, stores $a_{(i-1)n+i'}$, where $(k,j) = f(i)$ and $(j',k') = g(i')$ and $1 \leq i, i' \leq n$. With this data distribution, the prefix computation using an $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ can be carried out in the following four steps.

Step 1: Execute algorithm $\mathcal{A}$ on n level-1 submeshes concurrently to compute local prefixes so that processor $P_{(k-1)n_1+j', (j-1)n_2+k'}$ in submesh $M_{k,j}$ obtains $a_{(i-1)n+1} \oplus a_{(i-1)n+2} \oplus \cdots a_{(i-1)n+i'}$. Let $S_i = a_{(i-1)n+1} \oplus a_{(i-1)n+2} \oplus \cdots \oplus a_{in}$. For each submesh $M_{k,j}$, store its computed S_i in its leader processor. By Theorem 1, $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ can simulate n disjoint $n_1 \times n_2$ MMB's with only a constant-factor slowdown. Hence, these operations take $O(t(n))$ time.

Step 2: Execute algorithm $\mathcal{A}$ on a logical $n_2 \times n_1$ MMB with the leaders of level-1 submeshes as its processors. The computed result stored in the leader of $M_{k,j}$ is $T_i = S_1 \oplus S_2 \oplus \cdots S_i$, where $(k,j) = g(i)$. This step takes $O(t(n))$ time.

Step 3: The leader of each submesh $M_{k,j}$, where $(k,j) = g(i)$, broadcasts the value S_{i-1}, which can be computed from T_i and S_i, to all processors in $M_{k,j}$. Operating in parallel, this can be done in $O(1)$ time since $I(2, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}))$ can simulate n independent MMB's.

Step 4: Each processor performs operation $\oplus$ on the local prefix computed in Step 1 and the value it received in Step 3. This takes $O(1)$ time.

In summary, we have the following result.

Theorem 2 *If a prefix (resp. semigroup) operation on n operands can be carried out in $O(t(n))$ time using an n-processor MMB, then the same prefix (resp. semigroup) operation on n^2 operands can be carried out in $O(t(n))$ time using an n^2-processor square $(2,2)$-IMMB.*

Since the prefix and semigroup computations on n operands can be performed using an $n^{\frac{3}{8}} \times n^{\frac{5}{8}}$ MMB in $O(n^{\frac{1}{8}})$ time [5], we have the following corollary of Theorem 3.

Corollary 1 *Prefix and semigroup computations on N operands can be carried out in $O(N^{\frac{1}{16}})$ time using an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square $(2,2)$-IMMB.*

If we let each processor hold more than one operand, semigroup and prefix computations may be performed more efficiently. To see this, let us distribute $n^2t(n)$ operands to n^2 processors such that each processor holds $t(n)$ operands. In parallel, each processor performs prefix (or semigroup) computation on its own $t(n)$ operands sequentially. The total parallel time for this process is $O(t(n))$. Then, the parallel operations described above are performed on the partial results. Since the product of time and the number of processors is $O(n^2t(n))$, this computation is cost optimal.

Theorem 3 *If semigroup and prefix computations on n operands can be carried out in $O(t(n))$ time using an n-processor MMB, then the same computations on $n^2t(n)$ operands can be carried out using an n^2-processor square IMMB in $O(t(n))$ time, which is cost-optimal.*

The following corollary of Theorem 4 is obtained by letting $n = N^{\frac{8}{17}}$ and $t(n) = n^{\frac{1}{8}}$ for the $O(n^{\frac{1}{8}})$-time algorithms of [5] that use an $n^{\frac{3}{8}} \times n^{\frac{5}{8}}$ MMB.

Corollary 2 *Semigroup and prefix computations on N operands can be carried out using an $N^{\frac{8}{17}} \times N^{\frac{8}{17}}$ square $(2,2)$-IMMB in $O(N^{\frac{1}{17}})$ time, which is cost optimal.*

4 Semigroup and Prefix Computations on $(2,l)$-IMMB's

The algorithms presented in the previous section can be extended to run on $(2,l)$-IMMB's, the two-dimensional l-level IMMB's, where $l > 2$. Without loss of generality, we only consider $I(l,(n_{1,1},n_{1,2}),(n_{2,1},n_{2,2}),\cdots,(n_{l,1},n_{l,2}))$ such that $n_{l,1} = n_{l,2} = n_{l-1,1} = n_{l-1,2} = \cdots = n_{1,1} = n_{1,2} = n$. For easy reference, we denote this special IMMB by $(2,l)$-IMMB(n). Clearly, there are n^{2l} processors in $(2,l)$-IMMB(n), and these processors form an $n^l \times n^l$ processor array. This processor array can be viewed as an $n^{l-k-1} \times n^{l-k-1}$ array of $n^{k+1} \times n^{k+1}$ arrays, each being a level-$(k+1)$ submesh. A level-$(k+1)$ submesh is in turn considered as an $n \times n$ array of level-k submeshes. Let $M^{i,j,k+1}$, where $1 \le i,j \le n^{l-k-1}$, denote a level-$(k+1)$ submesh. We use $M^{i,j,k+1}_{i'j'}$, $1 \le i',j' \le n$, to denote a level-k submesh of $M^{i,j,k+1}$, and use $P^{i,j,k+1}_{a,b}$ to denote the processor in the a-th row and b-th column of processors in $M^{i,j,k+1}$, where $1 \le a,b \le n^{k+1}$. Define processor $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}$ as the leader of level-k submesh $M^{i,j,k+1}_{i'j'}$ for $1 < k < l$. As before, we call the processor in a level-1 submesh that has the largest index according to lexicographical order the leader of the level-1 submesh. The leaders of level-k submeshes are called level-k leaders for short. The level-k leaders in a level-$(k+1)$ submesh form an $n \times n$ processor array. We show that for each level-$(k+1)$ submesh, its level-$(k+1)$ buses, a subset of its level-k buses and and a subset of its level-k bridge local links (bridge links for short) form an $n \times n$ logical MMB defined on its level-k leaders.

Through vigorous analysis, we obtain the following lemma.

Lemma 1 *For any level-$(k+1)$ submesh $M^{i,j,k+1}$, where $1 < k < l$ and $1 \le i,j \le n^{l-k-1}$, in $(2,l)$-IMMB(n), there is a logical $n \times n$ MMB on its level-k submesh leaders using all its level-$(k+1)$ buses, a subset of its level-k buses, and a subset of its level-$(k+1)$ bridge links.*

The following fact is also useful in our prefix algorithm. For brevity, we omit its proof.

Lemma 2 *For any level-$(k+1)$ submesh $M^{i,j,k+1}$, where $1 < k < l$ and $1 \le i,j \le n^{l-k-1}$, in $(2,l)$-IMMB(n), there is a path that consists of a level-k bus and a level-$(k+1)$ bus from its leader to any of its level-k leader.*

Let $(a_1,a_2,\cdots,a_{n^2})$ be a sequence of n^2 operands for the prefix computation, and $\mathcal{A}$ be a prefix algorithm that runs in $O(t(n^2))$ time on an $n \times n$ MMB with each processor holding one operand. Suppose that

$$f : \{i | 1 \le i \le n^2\} \to \{(j,k) | 1 \le j,k \le n\}$$

is the function used by algorithm $\mathcal{A}$ to map a_i's and s_i's to processors in an $n \times n$ MMB; i.e. if $f(i) = (r,c)$, input a_i and result $s_i = a_1 \oplus a_2 \oplus \cdots \oplus a_i$ are in $P_{r,c}$ before and after the computation, respectively. We want to perform prefix computation on a sequence $A = (a_1,a_2,\cdots,a_{n^{2l}})$ using $(2,l)$-IMMB(n). We partition A into n^2 subsequences A_i, $1 \le i \le n^2$, each having $n^{2(l-1)}$ operands, such that $A_i = (a_{(i-1)n+1}a_{(i-1)n+2}\cdots,a_{in^{2(l-1)}})$. We use $f(i)$ to map A_i to the level-$(l-1)$ submesh $M^{r,c,l}$. Let $S_i = a_{(i-1)n+1} \oplus a_{(i-1)n+2} \oplus \cdots \oplus a_{in^{2(l-1)}}$. We want to use each level-$(l-1)$ submesh $M^{r,c,l}$ to compute S_i, where $(r,c) = f(i)$. To do so, we partition each A_i into n^2 subsequences $A_{i,1},A_{i,2},\cdots,A_{i,n^2}$, each having $n^{2(l-2)}$ operands. We assign $A_{i,j}$ to the level-$(l-2)$ submesh $M^{r,c,l}_{r',c'}$, where $(r,c) = f(i)$ and $(r',c') = f(j)$. We recursively partition each subsequence $A_{i_{l-1},i_{l-2},\cdots,i_{k+1}}$ into n^2 subsequences $A_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k}$, $1 \le i_k \le n^2$, and assign each of these subsequences to a level-k submesh of the level-$(k+1)$ submesh for $A_{i_{l-1},i_{l-2},\cdots,i_{k+1}}$ using function f until each $A_{i_{l-1},i_{l-2},\cdots,i_0}$ contains exactly one operand, which is assigned to a unique processor. For $1 \le k \le l-1$, let

$$S_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k} = \oplus_{a_j \in A_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k}} a_j.$$

We call $S_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k}$ the *active value* of unit $A_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k}$. Our algorithm consists of three phases. The first phases has $l-1$ iterations. In the first iteration, algorithm $\mathcal{A}$ is performed on all logical $n \times n$ MMB's defined on level-1 submeshes, and the active values of these submeshes are stored in their leaders. In the k-th iteration, $1 < k \leq l-1$, algorithm $\mathcal{A}$ is performed on all logical $n \times n$ MMB's defined on the leaders of level-$(k-1)$ submeshes, and the active values of all level-$(k+1)$ submeshes are routed to their respective leaders. By Lemma 1 and Lemma 2, each iteration takes $O(t(n^2))$ time, which is the time required for prefix computation on n operands using an $n \times n$ MMB. The total time for the first phase is $O(lt(n^2))$.

In the second phase, the leader of a level-k submesh broadcasts its active value to its level-1 leaders in the submesh. This broadcasting operation involves (i) sending its active value to one of its level-$(k-1)$ leaders, (ii) broadcasting the item to all its level-$(k-1)$ leaders using the logical $n \times n$ MMB defined on these level-$(k-1)$ leaders, and (iii) recursively broadcasting to leaders of lower levels. A bus conflict problem arises: broadcasting from the leader of a level-k submesh to all its level-$(k-1)$ leaders and broadcasting from the leader of a level-$(k-1)$ submesh to all its level-$(k-2)$ leaders may need to use the same level-k buses. To avoid such conflicts, we use a "pipelining" strategy, which consists of $l-2$ steps, Step 1 through Step $l-2$. In the j-th step, level-$(l-2i-j)$ leaders broadcast to the level-$(l-2i-j-1)$ leaders in their level-$(l-2i-j)$ submesh, where $0 \leq i \leq \frac{l-j}{2}$ using logical $n \times n$ MMB's defined on the level-$(l-2i-j-1)$ leaders. By Lemma 1 and Lemma 2, there is no conflict in the use of buses in this process. It is easy to verify that after $(l-2)$ steps, all data at higher level leaders are broadcast to level-1 leaders. Then, each level-1 leader broadcasts all data it has received to the processors in its level-1 submesh. The overall running time for the second phase is $O(l)$.

The task of the third phase is for each processor to update its prefix value using the data it received in the second phase. The time for this phase is obviously $O(l)$. In summary, the total time for this three-phase prefix algorithm is $O(lt(n^2))$, assuming that algorithm $\mathcal{A}$ takes $O(lt(n^2))$ time. The first phase of this algorithm can be used to perform a semigroup operation. By the results of [5, 9], semigroup and prefix computations on n^2 operands can be carried out using an $n \times n$ MMB $O(n^{\frac{1}{3}})$ time, i.e. $t(n^2) = O(n^{\frac{1}{3}})$. Let $N = n^{2l}$. Then, semigroup and prefix computations on N operands can be carried out using an l-level square $(2,l)$-IMMB(n) in $O(lN^{\frac{1}{6l}})$ time. For any given constant $0 < \epsilon < 1$, selecting l such that $l \geq \frac{1}{6\epsilon}$ leads to the following theorem.

Theorem 4 *For any constant $0 < \epsilon < 1$, there exists an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square $(2,l)$-IMMB(n) using which semigroup and prefix operations on N operands can be carried out in $O(N^{\epsilon})$ time.*

For any constant $0 < \epsilon < 1$, let $N^{1-\epsilon} = n^{2l}$. We select l such that $l \geq \frac{1-\epsilon}{6\epsilon}$ to construct an $N^{\frac{1-\epsilon}{2}} \times N^{\frac{1-\epsilon}{2}}$ $(2,l)$-IMMB(n). If we distribute N operands to $N^{1-\epsilon}$ processors of this $(2,l)$-IMMB(n), semigroup and prefix operations on these N operands can be carried out in $O(N^{\epsilon})$ time. Hence, we have the following claim.

Corollary 3 *For any constant $0 < \epsilon < 1$, there exists an $N^{\frac{1-\epsilon}{2}} \times N^{\frac{1-\epsilon}{2}}$ square $(2,l)$-IMMB(n) using which semigroup and prefix operations on N operands can be carried out in $O(N^{\epsilon})$ time, which is cost optimal.*

If we let n be a constant, say $n = 3$, then semigroup and prefix computations on N operands can be performed on a square $(2,l)$-I(3) in $O(l)$ time, which leads to the following claim.

Theorem 5 *Semigroup and prefix computations on N operands can be performed using an $O(\log N)$-level $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square IMMB in $O(\log N)$ time.*

Let each processor hold $\log N$ operands, the following corollary is straightforward.

Corollary 4 *Semigroup and prefix computations on N operands can be performed on an $O(\log N)$-level $\frac{N^{\frac{1}{2}}}{\sqrt{\log N}} \times \frac{N^{\frac{1}{2}}}{\sqrt{\log N}}$ square IMMB in $O(\log N)$ time, which is cost optimal.*

There is no contradiction between Theorem 4 and Theorem 5. The better performance claimed in Theorem 5 is achieved by using more buses.

5 Extension to d-Dimensional IMMB's

Our definition of l-level 2-D IMMB's can be extended to define d-dimensional IMMB's. A d-dimensional l-level IMMB is denoted by (d,l)-IMMB. The processors in an $n_1 \times n_2 \times \cdots \times n_d$ d-dimensional IMMB is denoted by $P_{i_1,i_2,\cdots,i_d}$, where $1 \leq i_j \leq n_j$ and $1 \leq j \leq d$. In a way similar to the definition of 2-D IMMB's, we can formally define (d,l)-IMMB's in a recursive fashion, assuming that a d-dimensional MMB is a 1-level d-dimensional IMMB, and its buses are level-1 buses. For a $(d,l-1)$-IMMB, we call its level-$(l-1)$ buses that connect the processor with the smallest index according to lexicographical order (i.e. $P_{1,1,\cdots,1}$) its *representative level-$(l-1)$ buses*. Clearly, there are exactly d representative level-k buss, one for each dimension. We use $n = \prod_{i=1}^{d} n_{l,i}$ copies of a $(d,l-1)$-IMMB to construct a (d,l)-IMMB as follows. Arrange these $(d,l-1)$-IMMB's, which are referred to as level-$(l-1)$ submeshes, as an $n_{l,1} \times n_{l,2} \times \cdots \times n_{l,d}$ array. Boundary processors are connected by bridge local links, and representative level-$(l-1)$ buses are merged into level-l buses. For brevity, we omit the detailed formal definition. By Theorem 1, it is easy to verify that the diameter of

a (d,l)-IMMB is dl. It is also straightforward that any d-dimensional GMMB can be simulated by its corresponding 2-level d-dimensional IMMB (which has fewer buses) with a constant-factor slowdown.

In [1, 5], it was shown that semigroup and prefix computations can be performed on N operands using an $N^{\frac{d2^{d-1}+1}{d2^d}} \times N^{\frac{d2^{d-2}+1}{d2^d}} \times \cdots \times N^{\frac{d+1}{d2^d}}$ d-dimensional MMB in $O(N^{\frac{1}{d2^d}})$ time. In [6], it is shown that semigroup and prefix computations can be performed on N operands using an $N^{\frac{d2^{d-1}+2}{d2^d+d}} \times N^{\frac{d2^{d-2}+2}{d2^d+d}} \times \cdots \times N^{\frac{d+2}{d2^d+d}}$ d-dimensional GMMB in $O(N^{\frac{1}{d2^d+d}})$ time. Using the $n^{\frac{d2^{d-1}+1}{d2^d}} \times n^{\frac{d2^{d-2}+1}{d2^d}} \times \cdots \times n^{\frac{d+1}{d2^d}}$ d-dimensional MMB as a level-1 IMMB, we can construct a (d,l)-IMMB recursively, with a (d,k)-IMMB being constructed using n copies of $n^{\frac{d2^{d-1}+1}{d2^d}} \times n^{\frac{d2^{d-2}+1}{d2^d}} \times \cdots \times n^{\frac{d+1}{d2^d}}$ level-$(k-1)$ submeshes. By properly selecting processors as leaders at different levels in a similar way as what we did for the 2-D case in the previous section, we can run the algorithm for the $n^{\frac{d2^{d-1}+1}{d2^d}} \times n^{\frac{d2^{d-2}+1}{d2^d}} \times \cdots \times n^{\frac{d+1}{d2^d}}$ d-dimensional MMB given in [1, 5] on logical MMB's defined on these levels. It is not difficult, though tedious, to prove the following theorem.

Theorem 6 *Semigroup and prefix computations on N operands can be performed using an N-processor (d,l)-IMMB in $O(N^{\frac{1}{ld2^d}})$ time.*

Since a $(d,2)$-IMMB can be considered as constructed from a d-dimensional GMMB by merging some of its buses, our time complexity $O(N^{\frac{1}{ld2^d}})$ is a significant improvement over the time complexity $O(N^{\frac{1}{d2^d+d}})$ achieved by a d-dimensional GMMB. Also, the time complexity claimed in Theorem 10 can be achieved by a set of (d,l)-IMMB's with a wide range of different aspect ratios.

Consider constructing a (d,d)-IMMB recursively as follows. The (d,l)-IMMB, $1 \leq l \leq d$, is constructed by an $n_{1+[(1+l-2) \bmod d]} \times n_{1+[(2+l-2) \bmod d]} \times \cdots \times n_{1+[(i+l-2) \bmod d]} \times \cdots \times n_{1+[(d+l-2) \bmod d]}$. The resulting (d,d)-IMMB is an $\overbrace{n \times n \times \cdots \times n}^{d}$ (d,d)-IMMB, where $n = \prod_i^d n_i$. Choosing n_i's such that $n_i = N^{\frac{d2^{d-1}+1}{d2^d}}$, we have the following extension of Theorem 6.

Theorem 7 *Semigroup and prefix computations on N operands can be performed using an d-level $\overbrace{N^{\frac{1}{d}} \times \cdots \times N^{\frac{1}{d}}}^{d}$ d-dimensional IMMB in $O(N^{\frac{1}{d^2 2^d}})$ time.*

6 Conclusion

In this paper, we proposed a generalization of mesh-connected computers with multiple buses, the IMMB's. Processors in an IMMB form a hierarchy of clusters (submeshes) of different sizes, and buses are partitioned into levels for fast data movement among processor clusters at different levels. We investigated the computation power of IMMB's by comparing their semigroup and prefix computation performances with that of MMB's and GMMB's. Our results significantly improve the best time complexities achieved on MMB's and IMMB's for the same computations.

References

[1] A. Bar-Noy and D. Peleg, "Square meshes are not always optimal," *IEEE Trans. on Computers*, Vol. 40, No. 2, pp. 196-204, Feb. 1991.

[2] S. H. Bokhari, "Finding maximum on an array processor with a global bus," *IEEE Trans. on Computers*, Vol. C-32, No. 2, pp. 133-139, 1984.

[3] D. A. Carlson, "Modified mesh-connected parallel computers", *IEEE Trans. on Computers*, Vol. C-37, No. 10, pp. 1315-1321, Oct. 1988.

[4] 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", *Journal of Parallel and Distributed Computing*, Vol. 15, pp. 79-84, 1992.

[5] Y.C Chen, W.T. Chen, G.H. Chen, and J.P. Sheu, "Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting," *IEEE Trans. on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 241- 245, April 1990.

[6] K. L. Chung, "Prefix computations on a generalized mesh-connected computer with multiple buses," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 2, pp. 196-199, February 1995.

[7] R. Miller, V. K. Prasanna-Kumar, D. Reisis, and Q.F. Stout, "Meshes with reconfigurable buses," *MIT Conference on Advanced Research in VLSI*, pp. 163-178, 1988.

[8] S. Olariu, J. L. Schwing and J. Zhang, "Fundamental algorithms on reconfigurable meshes", *Proc. 29-th Annual Allerton Conference on Communication, Control, and Computing*, pp. 811-820, Oct. 1991.

[9] V. Prasanna Kumar and C. S. Raghavendra, "Array processor with multiple broadcasting," *Journal of Parallel and Distributed Computing*, Vol. 4, No. 2, pp. 173-190, Apr. 1987.

[10] M. J. Serrano and B. Parhami, "Optimal architectures and algorithms for mesh-connected parallel computers with separable row/column buses", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 10, pp. 1073-1080, 1993.

[11] Q. F. Stout, "Mesh connected computers with broadcasting," *IEEE Transactions on Computers*, Vol. C-32, No. 9, pp. 826-830, Sept. 1983.