An Improved Generalization of Mesh-Connected Computers with Multiple Buses

Yi Pan, *Senior Member*, *IEEE*, S.Q. Zheng,
Keqin Li, *Senior Member*, *IEEE*, and Hong Shen, *Member*, *IEEE*

Abstract—Mesh-connected computers (MCCs) are a class of important parallel architectures due to their simple and regular interconnections. However, their performances are restricted by their large diameters. Various augmenting mechanisms have been proposed to enhance the communication efficiency of MCCs. One major approach is to add nonconfigurable buses for improved broadcasting. A typical example is the mesh-connected computer with multiple buses (MMB). We propose a new class of generalized MMBs, the improved generalized MMBs (IMMBs). We compare IMMBs with MMBs and a class of previously proposed generalized MMBs (GMMBs). We show the power of IMMBs by considering semigroup and prefix computations. Specifically, as our main result we show that for any constant $0 < \epsilon < 1$, one can construct an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ 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. Compared with the previous best complexities $O(N^{\frac{1}{8}})$ and $O(N^{\frac{1}{16}})$ achieved on a rectangular MMB and GMMB, respectively, for the same computations, our results show that IMMBs are more powerful than MMBs and GMMBs.

Index Terms—Bus, mesh-connected computer, mesh-connected computer with multiple buses, parallel algorithm, parallel architecture, parallel computing, processor array.

1 Introduction

Among various parallel architectures, mesh-connected computers (MCCs) have received considerable attention. The processors in an MCC are arranged as a processor array, and each processor is connected to its nearest neighbors. Due to its simple and regular interconnection pattern, an 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 MCCs. One major approach is to add buses for improved broadcasting [1], [7], [8], [9], [13], [15], [19], [21], [22], [23], [24]. A typical example is the mesh-connected computer with multiple broadcasting (MMB) [21]. A two-dimensional MMB is a two-dimensional (2D) MCC with a bus for each row and each column. Fig. 1 shows a 4×4 2D MMB.

• Y. Pan is with the Department of Computer Science, Georgia State University, Altanta, GA 30303. E-mail: pan@cs.gsu.edu.
• S.Q. Zheng is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688.
• K. Li is with the Department of Mathematics and Compute Science, State University of New York, New Paltz, NY 12561-2499.
• H. Shen is with the School of Computing and Information Technology, Griffith University, Nathan, Queensland 4111, Australia.

Manuscript received 26 Jan. 1998; revised 22 Nov. 1999; accepted 18 Jan. 2000.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 106206.

In this paper, we propose a class of improved generalized mesh-connected computers with multiple buses (IMMB). We compare the performances of IMMBs and MMBs by considering parallel semigroup and prefix computations. Semigroup computations are an important class of computation problems. Examples include computing sum, product, minimum/maximum, Boolean parity, AND, and OR. Prefix computations are related to semigroup computations; they have a wide range of applications such as processor allocation, data distribution and alignment, data compaction, job scheduling, sorting, packet routing, string matching, lexical analysis, matrix computation, linear recurrence, polynomial evaluation, graph algorithms, general Horner expressions and general arithmetic formulae. Refer to [2], [6], [16], [17] for references of these applications. Efficient semigroup and prefix algorithms serve as important primitives for parallel computing.

Various algorithms for semigroup and prefix computations on different machine models have been proposed in the literature. Kumar and Raghavendra [21] showed that semigroup computations on N operands can be performed using an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square MMB in $O(N^{\frac{1}{6}})$ time. Chen et al. [11] later showed that prefix computations on N operands can be performed using on an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ square MMB in the same amount of time. They showed that if the MMB has a rectangular shape, i.e., the sizes of two dimensions are not the same, better complexity can be achieved. In particular, they showed that semigroup and prefix computations on

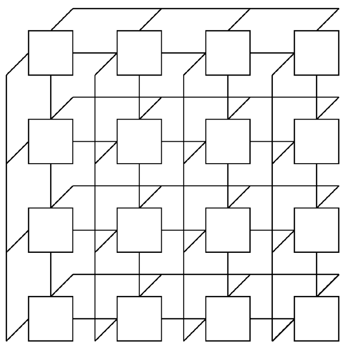


Fig. 1. A 4×4 MMB.

N operands can be performed using an $N^{\frac{5}{8}} \times N^{\frac{1}{8}}$ MMB in $O(N^{\frac{1}{8}})$ time. Chung [13] proposed a generalized MMB (GMMB) architecture in which k^2 $n_1 \times n_2$ MMBs, arranged as a $k \times k$ array, are connected by local links (defined in the next section). By selecting $k = N^{\frac{1}{10}}$, $n_1 = N^{\frac{1}{2}}$, and $n_2 = N^{\frac{3}{10}}$, he showed that semigroup and prefix computations can be performed using an $N^{\frac{3}{5}} \times N^{\frac{2}{5}}$ GMMB in $O(N^{\frac{1}{10}})$ time, which is the best possible because the diameter of this generalized MMB is $O(N^{\frac{1}{10}})$. The major drawback of this GMMB is that broadcasting performance is sacrificed by a factor of $O(N^{1/c})$, where $c \leq 4$, for improved semigroup and prefix computation performances with increased number of buses, compared with MMB. Semigroup and prefix computations using d-dimensional MMBs and GMMBs have also been considered. In [3], [11], 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 [13], 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. Define the aspect ratio of a $n_1 \times n_2 \times \cdots \times n_d$ d-dimensional mesh as $\frac{\max\{n_1,n_2,\cdots,n_d\}}{\min\{n_1,n_2,\cdots,n_d\}}$. We call a d-dimensional mesh a square mesh if its aspect ratio is 1. The performances claimed in [3], [11], [13] are only valid for meshes with very large aspect ratios. Dighe, Vaidyanathan and Zheng proposed a multiple-bus architecture called bus-connected ringed tree (BRT) in [15]. In a BRT, each processor is connected to two buses, and all the buses have the same size, which is the number of processors connected to a bus. They showed that a 2D BRT, which is a 2D processor array with multiple buses, can simulate a mesh-of-trees efficiently. Based on the prefix sum algorithm of [17] for a tree, a 2D BRT can carry out a prefix computation in $O(\log N^2)$ time.

Better performances are possible if switches are added to make buses reconfigurable. For example, prefix and semigroup computations can be easily carried out in $O(\log N)$ time using an MBB whose buses can be partitioned into segments by switches ([22]) or a reconfigurable mesh ([19]). As a special case, Olariu et al. [20] showed that the prefix sum operation on N integer operands in the range $[0, N-1]$ can be performed on an $N \times N$ reconfigurable mesh in $O(1)$ time. In this algorithm, the number of processors used is much larger than the number of operands, and dynamically reconfigured paths are used as an integral part of computation. This technique cannot be generalized to solve general semigroup and prefix problems with the same time complexity.

Like the MMBs of [21] and GMMBs of [13], buses of the IMMBs 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 IMMBs. 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 [13] with a constant factor of slowdown, 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 multilevel $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ 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. The generalization of these results to d-dimensional IMMBs is discussed in Section 5. The major results presented in this section is that, one can perform semigroup and prefix computations on N operands in an N-processor (d,l)-IMMB in $O(N^{\frac{1}{ld2^d}})$ time. When selecting $l = d$, one can always obtain a d-dimensional square IMMB. We conclude the paper in Section 6 by discussing the implications of our results.

2 Two-Dimensional IMMBs

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 2D 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

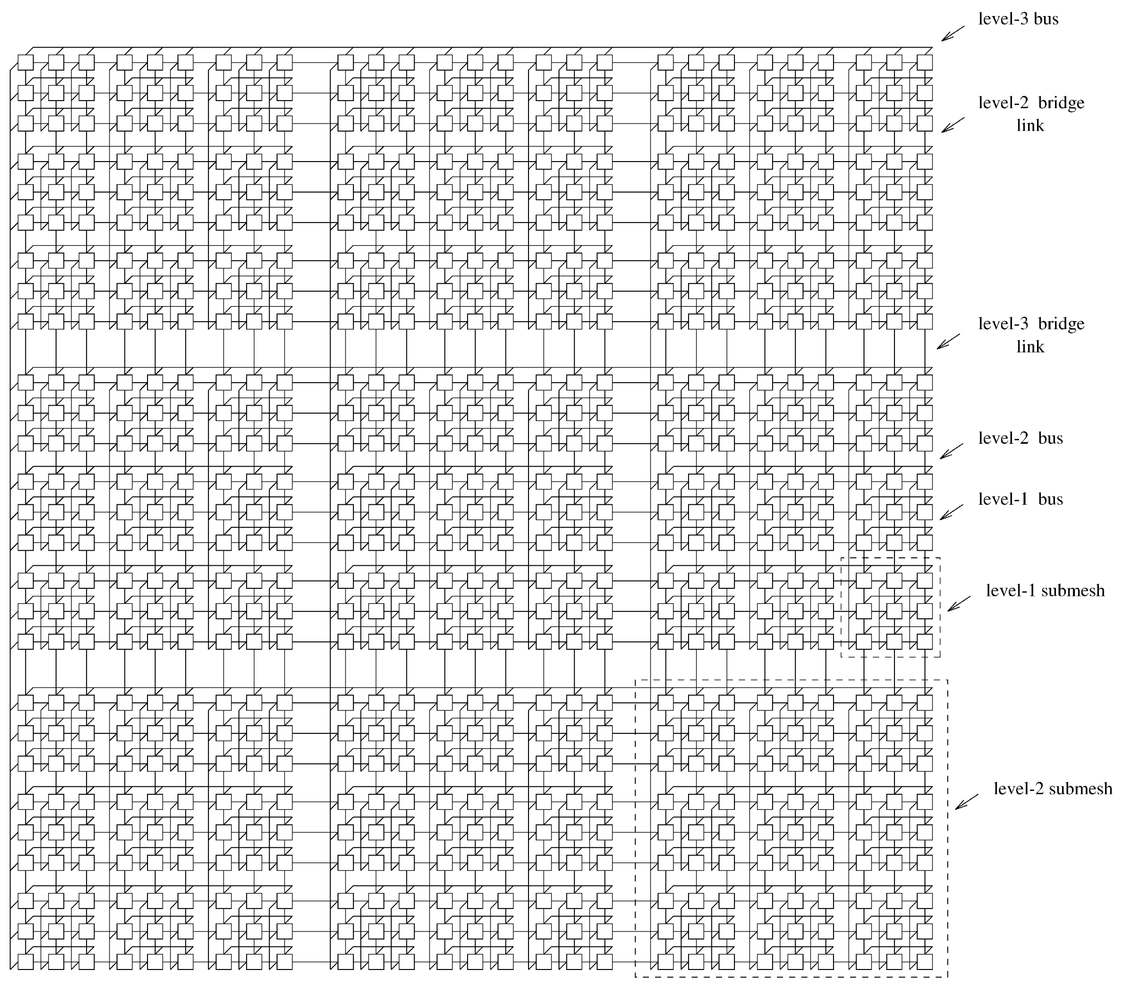


Fig. 2. A 2D 3-level 27×27 $I(3, (3,3), (3,3), (3,3))$, which consists of a 3×3 array of level-2 submeshes. Each level-2 submesh consists of a 3×3 array of level-1 submeshes.

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 (respectively, column) of the $n_{l,1} \times n_{l,2}$ array of level-$(l-1)$ submeshes, we do the following: Merge the topmost (respectively, left most) row (column) buses, which were level-$(l-1)$ buses, of these IMMBs 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, $1 \leq k \leq l-1$, 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 2D IMMB is also referred to as a $(2, l)$-IMMB. To avoid degeneracy, we assume that $n_{i,1} \geq 3$ and $n_{i,2} \geq 3$ for $1 \leq i \leq l$. Define $b_1 = n_{1,1}n_{1,2}$ and

$$\begin{aligned} b_i &= n_{i,1}n_{i,2}b_{i-1} - n_{i,1}(n_{i,2}-1) - n_{i,2}(n_{i,1}-1) \\ &= n_{i,1}n_{i,2}(b_{i-1}-2) + n_{i,1} + n_{i,2} \end{aligned}$$

for $1 < i \leq l$. Clearly, b_l is the number of buses in $I(l, (n_{1,1}, n_{1,2}),\ (n_{2,1},\ n_{2,2}), \cdots,\ (n_{l,1},\ n_{l,2}))$. Fig. 2 shows the structure of $I(3, (3,3),\ (3,3),\ (3,3))$, a $(2,3)$-IMMB.

An IMMB is represented by a hypergraph G, whose vertices and hyperedges correspond to the processors and connections (buses and local links), respectively. Using the hypergraph theory ([4]), one can derive many topological properties of IMMB. We refer readers to [26] for the analysis of hypergraph-based interconnection structures.

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 $1 \leq i \leq k-1$, 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.

Throughout this paper, we adopt the same assumptions that have been used in all previous work on MMBs and their variants. We assume that it takes constant time to

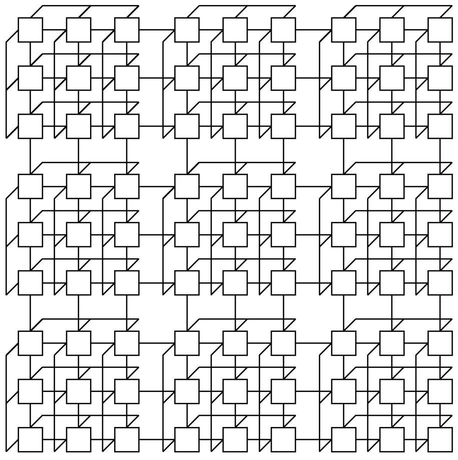


Fig. 3. A 9×9 GMMB, which is constructed using nine 3×3 MMBs.

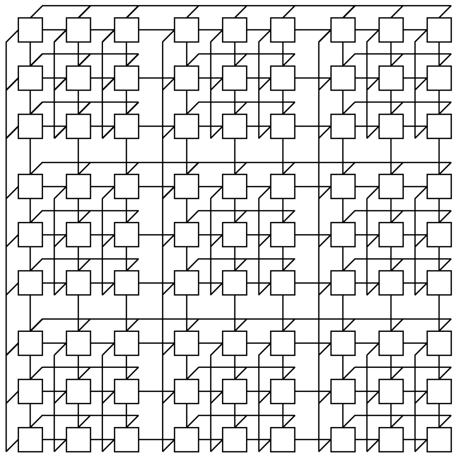


Fig. 4. A 9×9 IMMB corresponding to the GMMB of Fig. 3.

broadcast a message on a bus as in [1], [7], [8], [9], [13], [15], [19], [21], [22], [23], [24]. To ensure conflict-free accesses of buses, each processor is equipped with an off-line circuitry so that bus allocations, although operated dynamically, are predetermined by an off-line scheduling algorithm which is known at compilation time. The bus accesses are compiled in advance so that no two processors attempt to use the same bus at the same time. With these assumptions, bus arbitration overheads are ignored, algorithm analysis is simplified, and comparing different algorithms becomes easier. The complexity of an algorithm is measured by the total number of parallel computation steps and parallel communication steps. These assumptions are adopted in all previous work on processor arrays with synchronous buses.

Let $m = \prod_{i=1}^{l} n_{i,1}$ and $n = \prod_{i=1}^{l} n_{i,2}$. We use $P_{i,j}$, $1 \leq i \leq m$ and $1 \leq j \leq n$, to denote the processor in row i and column j. Consider the diameter of $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$. Between any two level-$(l-1)$ submeshes in an $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$, there is a path of length at most 2 using level-l buses. Routing a message from a processor $P_{i',j'}$ to a processor $P_{i'',j''}$ using buses can be done as follows: If $P_{i',j'}$ and $P_{i'',j''}$ are in the same level-$(l-1)$ submesh, then we only need to consider routing within this submesh. If they are in two different level-$(l-1)$ submeshes, then we find a path from the level-$(l-1)$ submesh M' that contains $P_{i',j'}$ to the level-$(l-1)$ submesh M'' that contains $P_{i'',j''}$. This path consists of at most two buses, say L' and L''. Let $P_{a',b'}$ and $P_{a'',b''}$ be the two processors in M' and M'' that is connected to L' and L'', respectively. Then, to complete the path from $P_{i',j'}$ to $P_{i'',j''}$, we need to find a path from $P_{i',j'}$ to $P_{a',b'}$ in M' and a path from $P_{i'',j''}$ to $P_{a'',b''}$ in M'' recursively. By induction, there exists a path from any $P_{i',j'}$ to any $P_{i'',j''}$ in an $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$ of length no more than $2l$. It is not difficult to verify that the distance between $P_{1,1}$ and $P_{m,n}$ in an $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$ is $2l$. Therefore, the diameter of an $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$ is $2l$. Clearly, broadcasting a message to all processors in an $I(l, (n_{1,1}, n_{1,2}), (n_{2,1}, n_{2,2}), \cdots, (n_{l,1}, n_{l,2}))$ takes $O(l)$ time.

Our two-level IMMBs closely resemble the GMMBs proposed in [13], which are constructed using multiple copies of MMB by only introducing additional local links. A 9×9 two-dimensional GMMB is shown in Fig. 3, and its corresponding IMMB is shown in Fig. 4. 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 2D GMMB of [13] as follows. In each level-1 submesh, we use the level-1 bus that connects the second processor row (respectively, column) to simulate a level-1 bus that connects all processors in the first row (respectively, column). For example, suppose that processor $P_{i,j}$ in the first row (respectively, column) of a level-1 submesh wants to broadcast a message to all processors in the same row (respectively, column) of the same submesh. It can first send the message to processor $P_{i+1,j}$ (respectively, $P_{i,j+1}$) in the second row (respectively, column) using a local link, and then broadcast it to all processors in that row (respectively, column) using the level-1 bus that connects them. After this, each processor in the second row, (respectively, 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 MMBs defined on the level-1 submeshes, and each $n_{1,1} \times n_{1,2}$ MMB substructure of the 2D GMMB can be simulated by a logical MMB. This leads to the following claim.

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

Obviously, the converse of this theorem is not true. This is because the diameter of the 2D 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 IMMBs are more powerful than MMBs and GMMBs 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 number of 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 2D 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 $1 \le i \le N$. 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 [13] on GMMBs, 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 IMMBs, 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 $1 \le i \le n_2$ and $1 \le j \le n_1$, and they form an $n_2 \times n_1$ array. It is simple to see that the leaders P_{in_1, kn_2} and $P_{in_1, (k+1)n_2}$ (respectively, 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:

$$P_{in_1,kn_2} \overset{bridge\ link}{\longrightarrow} P_{in_1,kn_2+1} \overset{level-1\ bus}{\longrightarrow} P_{in_1,(k+1)n_2}$$
$$(respectively, P_{kn_1,jn_2} \overset{bridge\ link}{\longrightarrow} P_{kn_1+1,jn_2} \overset{level-1\ bus}{\longrightarrow} P_{(k+1)n_1,jn_2}).$$

Furthermore, the kth level-2 row (respectively, column) bus can be used to simulate a bus that connects kth leader row (column). For example, suppose that a submesh leader wants to broadcast a message to all leader processors in its row (respectively, column). It can send the message to the processor in the first row (respectively, column) of its submesh via the level-1 column (respectively, 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 (respectively, column) send the received messages to their corresponding processors (leaders) in the last row (respectively, column) of the submesh via level-1 column (respectively, 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

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

is the function used by algorithm $\mathcal{A}$ to map a_is and s_is 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, $1 \le i \le n$, 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 $1 \le k \le n_2$ and $1 \le j \le n_1$. Submesh $M_{k,j}$ consists of processors $P_{a,b}$, $(k-1)n_1 + 1 \le a \le kn_1$ and $(j-1)n_2 + 1 \le j \le jn_2$. Define

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

such that $g(i) = (k,j)$ if $f(i) = (j,k)$. We use $g(i)$ to map A_is 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 $1 \le q \le n$, 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 $1 \le i, i' \le n$. 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.

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 $n_1 \times n_2$ MMBs with only a constant-factor slowdown. Hence, these operations take $O(t(n))$ time.
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.
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 MMBs.

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 (respectively, 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 [11], we have the following corollary of Theorem 2.

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 $n^2 t(n)$ 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 $O(n^2 t(n))$, 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 $n^2 t(n)$ 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 3 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 [11] 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*)-IMMBs

The algorithms presented in the previous section can be extended to run on (2, *l*)-IMMBs, the 2D *l*-level IMMBs, 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.

The simulation of a local link connecting the leaders of $M^{i,j,k+1}_{i',j'}$ and $M^{i,j,k+1}_{i',j'+1}$ in the same level-(k + 1) submesh $M^{i,j,k+1}$ is by the following path:

$$\begin{aligned} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,j'n^k-n^{k-1}+1} &\xrightarrow{level-k\ bus} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,j'n^k} \\ &\xrightarrow{level-(k+1)\ bridge\ link} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,j'n^k+1} \\ &\xrightarrow{level-k\ bus} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,(j'+1)n^k-n^{k-1}+1}. \end{aligned}$$

Similarly, the simulation of a local link connecting the leaders of $M^{i,j,k+1}_{i',j'}$ and $M^{i,j,k+1}_{i'+1,j'}$ in the same level-(k + 1)submesh $M^{i,j,k+1}$ is by the following path:

$$\begin{aligned} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,j'n^k-n^{k-1}+1} &\xrightarrow{level-k\ bus} P^{i,j,k+1}_{i'n^k,j'n^k-n^{k-1}+1} \\ &\xrightarrow{level-(k+1)\ bridge\ link} P^{i,j,k+1}_{i'n^k+1,j'n^k-n^{k-1}+1} \\ &\xrightarrow{level-k\ bus} P^{i,j,k+1}_{(i'+1)n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}. \end{aligned}$$

It is easy to verify that each level-*k* bus is used in at most two of these paths, and each level-(k + 1) bus is used in at most one of these paths. Using these paths, the leaders of level-*k* submeshes can simulate an $n \times n$ MCC. There are $n-1$ horizontal (respectively, vertical) level-(k + 1) buses in a level-(k + 1) submesh $M^{i,j,k+1}$ for $1 \le k < l$. In this case, we label them with $B^{i,j,k+1}_{h_1}$ to $B^{i,j,k+1}_{h_{n-1}}$ (respectively, $B^{i,j,k+1}_{v_1}$ to $B^{i,j,k+1}_{v_{n-1}}$) from top to bottom (respectively, from left to right). The level-(k + 1) bus $B^{i,j,k+1}_{h_r}$ (respectively, $B^{i,j,k+1}_{v_r}$)

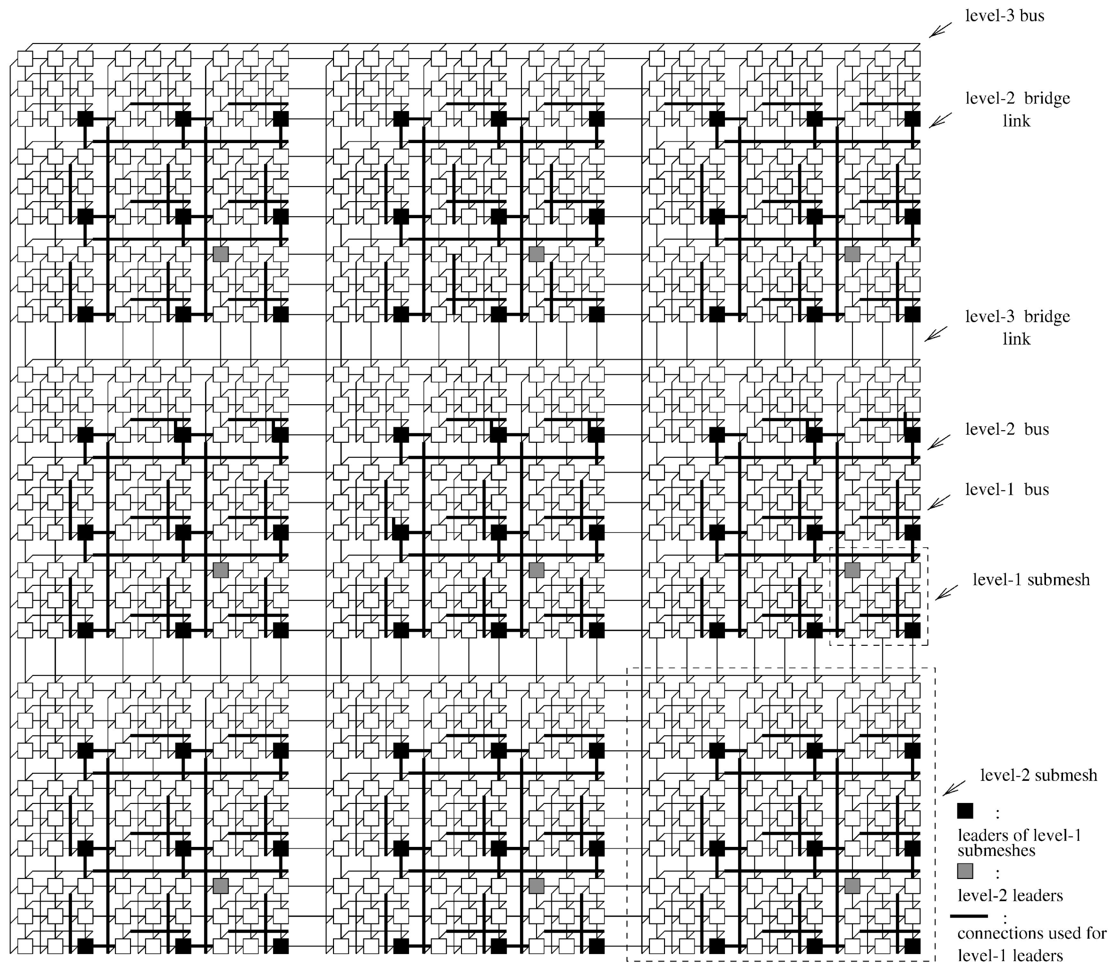


Fig. 5. Connections in $(2,3)$-IMMB(3) used to simulate nine disjoint 3×3 logical MMBs of level-1 submesh leaders.

connects processors $P^{i,j,k+1}_{rn^k+1,s}$ (respectively, $P^{i,j,k+1}_{s,rn^k+1}$) where $1 \leq s \leq n^{k+1}$. In a level-(k + 1) submesh $M^{i,j,k+1}$, the level-k leader $P^{i,j,k+1}_{(r+1)n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}$ (respectively, $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,(r+1)n^k-n^{k-1}+1}$), $1 \leq r \leq n-1$, is connected to processor $P^{i,j,k+1}_{rn^k+1,j'n^k-n^{k-1}+1}$ (respectively, $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,rn^k+1}$) by a unique level-$k$ bus. Hence, we can use processors $P^{i,j,k+1}_{rn^k+1,j'n^k-n^{k-1}+1}$ (respectively, $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,rn^k+1}$) to simulate level-$k$ leaders $P^{i,j,k+1}_{(r+1)n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}$ (respectively, $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,(r+1)n^k-n^{k-1}+1}$), $1 \leq j' \leq n$, and use level-(k + 1) bus $B^{i,j,k+1}_{h_r}$ (respectively, $B^{i,j,k+1}_{v_r}$) to simulate a level-(k + 1) bus that connects these level-k leaders. These simulations cause a constant-factor slowdown. We also note that the level-k leader processor $P^{i,j,k+1}_{n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}$ is connected to processor $P^{i,j,k+1}_{n^k+1,j'n^k-n^{k-1}+1}$ by a path

$$P^{i,j,k+1}_{n^k-n^{k-1}+1,j'n^k-n^{k-1}+1} \overset{level-k\ bus}{\longrightarrow} P^{i,j,k+1}_{n^k,j'n^k-n^{k-1}+1} \overset{level-(k+1)\ bridge\ link}{\longrightarrow} P^{i,j,k+1}_{n^k+1,j'n^k-n^{k-1}+1}.$$

and the level-k leader processor $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k-n^{k-1}+1}$ is connected to processor $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k+1}$ by a path

$$P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k-n^{k-1}+1} \overset{level-k\ bus}{\longrightarrow} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k} \overset{level-(k+1)\ bridge\ link}{\longrightarrow} P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k+1}.$$

Thus, we are able to use the level-(k + 1) bus $B^{i,j,k+1}_{h_1}$ (respectively, $B^{i,j,k+1}_{v_1}$) to simulate a level-(k + 1) bus that connects level-k leaders $P^{i,j,k+1}_{n^k-n^{k-1}+1,j'n^k-n^{k-1}+1}$ (respectively, $P^{i,j,k+1}_{i'n^k-n^{k-1}+1,n^k-n^{k-1}+1}$), $1 \leq j' \leq n$, with a constant-factor slowdown. For $k = l - 1$, there are n horizontal level-(k + 1) buses and n vertical level-(k + 1) buses, and all of them can be used to simulate the buses connecting level-$(l-1)$ leaders. For example, for the $(2,3)$-IMMB(3) shown in Fig. 2, the connections used to simulate 3×3 logical MMBs of level-1 leaders are shown in Fig. 5 and the connections used to simulate the 3×3 logical MMB of level-2 leaders are shown in Fig. 6. The shaded processor in Fig. 6 is the leader of a level-3 submesh. This IMMB is used to construct a $(2,4)$-IMMB(3). Summarizing the above discussions, we have the following lemma.

Lemma 1. *For any level-(k + 1) submesh $M^{i,j,k+1}$, where $1 < k < l$ and $1 \leq i,j \leq n^{l-k-1}$, 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.*

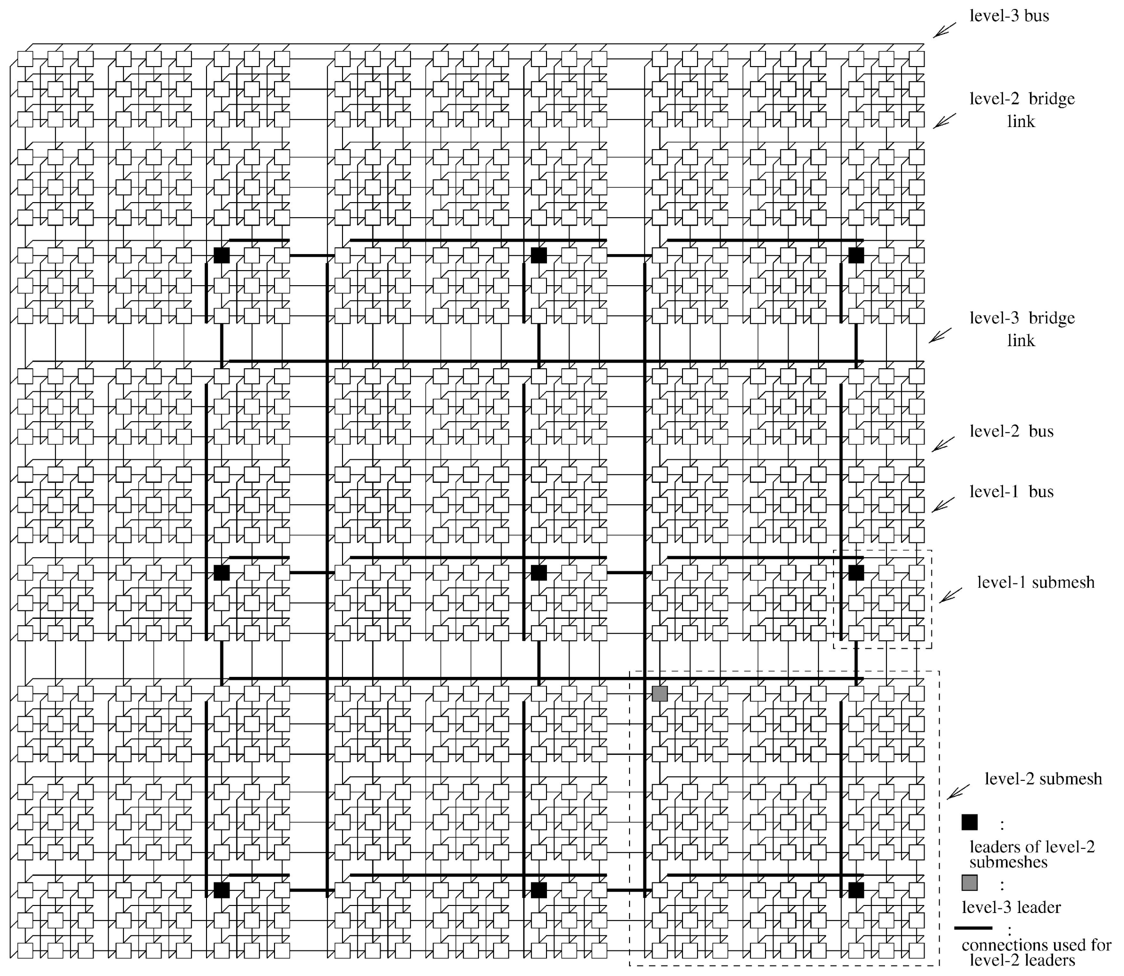


Fig. 6. Connections in $(2,3)$-IMMB(3) used to simulate the 3×3 logical MMB of level-2 submesh leaders.

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 $1 \leq i, j \leq n^{l-k-1}$, 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

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

is the function used by algorithm $\mathcal{A}$ to map a_is and s_is 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, $1 \leq i \leq n^2$, 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 $A_{i_{l-1},i_{l-2},\cdots,i_{k+1},i_k}$, $1 \leq i_k \leq n^2$, 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 $1 \leq k \leq l-1$, 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 have $l-1$ iterations. In the first iteration, Algorithm $\mathcal{A}$ is performed on all logical $n \times n$ MMBs defined on level-1 submeshes, and the active values of these submeshes are stored in their leaders. In the kth iteration, $1 < k \leq l-1$, algorithm $\mathcal{A}$ is performed on all logical $n \times n$ MMBs 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 $O(l\,t(n^2))$.

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 1) sending its active value to one of its level-$(k-1)$ leaders, 2) 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 3) 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 jth step, level-$(l-2i-j)$ leaders broadcast to the level-$(l-2i-j-1)$ leaders in their level-$(l-2i-j)$ submesh, where $0 \le i \le \frac{l-j}{2}$ using logical $n \times n$ MMBs 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 $O(l\,t(n^2))$ time. The first phase of this algorithm can be used to perform a semigroup operation. By the results of [11], [21], 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 $l \ge \frac{1}{6\epsilon}$ 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 $l \ge \frac{1-\epsilon}{6\epsilon}$ 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 IMMBs

Our definition of l-level 2D IMMBs can be extended to define d-dimensional IMMBs. 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 $1 \le i_j \le n_j$ and $1 \le j \le d$. In a way similar to the definition of 2D IMMBs, we can formally define (d, l)-IMMBs 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 buses, 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 $(d, l-1)$-IMMBs, 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. We illustrate the construction of a $(3, l)$-IMMB from $(3, l-1)$-IMMBs in Fig. 7. In Fig. 7a, we show a $(3, l-1)$-IMMB with thick line segments designating its three representative level-$(l-1)$ buses. In Fig. 7b, we illustrate the construction of a (3,l)-IMMB from nine copies of the $(3, l-1)$-IMMB shown in Fig. 7a. In this case, $n_{l,1} = n_{l,2} = n_{l,3} = 3$. The representative level-$(l-1)$ buses of level-$(l-1)$ 3D submeshes are merged into level-l buses, as shown in Fig. 7b, and the three thicker line segments designate the representative level-l buses of the constructed (3, l)-IMMB. If the bus merge operation is not performed in the construction of a 2-level d-dimensional IMMB, we obtain a d-dimensional GMMB. By Theorem 1, it is easy to

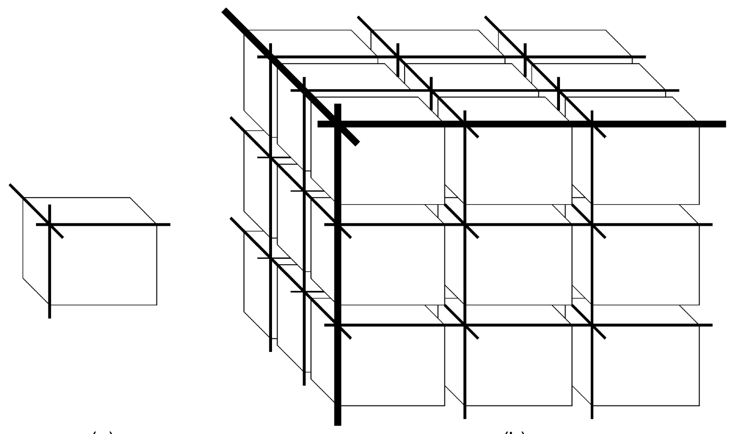


Fig. 7. The construction of a 3D l-level IMMB. (a) A $(3, l-1)$-IMMB, with its three representative level-$(l-1)$ level-$(l-1)$ buses designated by thick line segments. (b) A $(3, l)$-IMMB constructed using 9 $(3, l-1)$-IMMBs of (a).

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.

It was shown in [3], [11] that a prefix computation on N operands can be performed in $O(N^{\frac{1}{24}})$ time using an $N^{\frac{13}{24}} \times N^{\frac{7}{24}} \times N^{\frac{1}{6}}$ 3D MMB. The aspect ratio of this MMB is $N^{\frac{3}{8}}$. It was shown in [13] that the same computation can be performed in $O(N^{\frac{1}{27}})$ time using an $N^{\frac{14}{27}} \times N^{\frac{8}{27}} \times N^{\frac{5}{27}}$ 3D GMMB. The aspect ratio of this GMMB is $N^{\frac{1}{3}}$. Since a 3D GMMB of [13] can be simulated by a 2-level 3D IMMB obtained by merging a subset of its buses with a constant-factor slowdown, semigroup and prefix computations on N operands can be performed in $O(N^{\frac{1}{27}})$ time using a 2-level 3D $N^{\frac{14}{27}} \times N^{\frac{8}{27}} \times N^{\frac{5}{27}}$ 3D IMMB. Actually, we can do much better.

Let $n_1 = n^{\frac{13}{24}}$, $n_2 = n^{\frac{7}{24}}$, $n_3 = n^{\frac{1}{6}}$, and $n = n_1 n_2 n_3$. We construct a (3,2)-IMMB by arranging n copies of $n^{\frac{13}{24}} \times n^{\frac{7}{24}} \times n^{\frac{1}{6}}$ MCC, each being referred to as a level-1 3D submesh, as an $n^{\frac{1}{6}} \times n^{\frac{7}{24}} \times n^{\frac{13}{24}}$ 3D array. Each processor is referred by $P_{x,y,z}$, where $1 \leq x \leq n^{\frac{17}{48}}$, $1 \leq y \leq n^{\frac{14}{48}}$, and $1 \leq z \leq n^{\frac{17}{48}}$. It is not difficult to see that all level-1 buses and a subset of links can be used to simulate n disjoint $n^{\frac{13}{24}} \times n^{\frac{7}{24}} \times n^{\frac{1}{6}}$ 3D MMBs. We call the processor in a level-1 3D submesh that has the largest index according to lexicographical order the *leader of the submesh*. The reader will not fail to see that the buses and links in this $(3,2)$-IMMB define an $n^{\frac{1}{6}} \times n^{\frac{7}{24}} \times n^{\frac{13}{24}}$ logical MMB on the leaders of level-1 3D submeshes. Hence, the algorithm of [11] can be run on all level-1 submeshes concurrently, and the same algorithm can be run on the logical $n^{\frac{1}{6}} \times n^{\frac{7}{24}} \times n^{\frac{13}{24}}$ MMB defined on level-1 leaders. By letting $N = n^2$, we have the following theorem:

Theorem 6. *Semigroup and prefix computations on N operands can be performed using an* $N^{\frac{17}{48}} \times N^{\frac{14}{48}} \times N^{\frac{17}{48}}$ $(3,2)$*-IMMB in* $O(N^{\frac{1}{48}})$ *time.*

This theorem is a generalization of Theorem 2. Note that a (3,2)-IMMB can be considered as constructed from a three-dimensional GMMB by merging some of its buses. This result is an improvement over the best known $O(N^{\frac{1}{27}})$ time complexity achieved by a GMMB. The aspect ratio of the $N^{\frac{17}{48}} \times N^{\frac{14}{48}} \times N^{\frac{17}{48}}$ $(3,2)$-IMMB is $N^{\frac{1}{16}}$ and is also better. We might have constructed an $N^{\frac{26}{48}} \times N^{\frac{14}{48}} \times N^{\frac{8}{48}}$ $(3,2)$-IMMB with an $n^{\frac{13}{24}} \times n^{\frac{7}{24}} \times n^{\frac{1}{6}}$ array of $n^{\frac{13}{24}} \times n^{\frac{7}{24}} \times n^{\frac{1}{6}}$ level-1 submeshes. While having the same $O(N^{\frac{1}{48}})$ time complexity for semigroup and prefix computations, this $(3,2)$-IMMB has a much larger aspect ratio $N^{\frac{3}{8}}$. In fact, without lossing time efficiency, several $(3,2)$-IMMBs with different aspect ratios can be constructed by choosing different permutations of n_1, n_2, and n_3.

A 3-level $N^{\frac{1}{3}} \times N^{\frac{1}{3}} \times N^{\frac{1}{3}}$ 3D IMMB can be recursively constructed from an arbitrary $n_1 \times n_2 \times n_3$ MMB. First, we construct a 2-level $n_2n_1 \times n_3n_2 \times n_1n_3$ IMMB by arranging $n = n_1n_2n_3$ copies of an $n_1 \times n_2 \times n_3$ MMB, which are level-1 3D submeshes, as an $n_2 \times n_3 \times n_1$ 3D array, and properly adding bridge links and merging some level-1 buses into level-2 buses. Then, we construct a 3-level $n_3n_2n_1 \times n_1n_3n_2 \times n_2n_1n_3$ IMMB by arranging n copies of this $n_2n_1 \times n_3n_2 \times n_1n_3$ IMMB, which are level-2 3D submeshes, as an $n_3 \times n_1 \times n_2$ 3D array, and properly adding bridge links and merging some level-2 buses into level-3 buses. The resulting structure is a 3-level $n \times n \times n$ IMMB. Let $n_1 = n^{\frac{13}{24}}$, $n_2 = n^{\frac{7}{24}}$, $n_3 = n^{\frac{1}{6}}$. By properly selecting leaders of 3D submeshes at every level, we can identify disjoint logical MMBs and run the prefix algorithm of [3], [11] on these MMBs in a way similar to that described in the previous section. It is simple to verify that semigroup and prefix computations on n^3 operands on this IMMB takes $O(n^{\frac{1}{24}})$ time. Letting $N = n^3$, we have the following theorem.

Theorem 7. *Semigroup and prefix computations on N operands can be performed using an* $N^{\frac{1}{3}} \times N^{\frac{1}{3}} \times N^{\frac{1}{3}}$ $(3,3)$*-IMMB in* $O(N^{\frac{1}{72}})$ *time.*

In [3], [11], 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 [13], 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 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 to what we did for the 2D 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 [3], [11] on logical MMBs defined on these levels. It is not difficult, though tedious, to prove the following generalization of Theorems 6 and 7.

Theorem 8. *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 8 can be achieved by a set of (d,l)-IMMBs 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

$$n_{1+[(1+l-2)\ mod\ d]} \times n_{1+[(2+l-2)\ mod\ d]} \times \cdots \\ \times n_{1+[(i+l-2)\ mod\ d]} \times \cdots \times n_{1+[(d+l-2)\ mod\ d]}.$$

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_is such that $n_i = N^{\frac{d2^{d-i}+1}{d2^d}}$, we have the following extension of Theorem 8.

Theorem 9. *Semigroup and prefix computations on N operands can be performed using a d-level* $\overbrace{N^{\frac{1}{d}} \times N^{\frac{1}{d}} \times \cdots \times N^{\frac{1}{d}}}^{d}$ *d-dimensional IMMB in* $O(N^{\frac{1}{d^2 d^d}})$ *time.*

6 Concluding Remarks

In this paper, we proposed a generalization of mesh-connected computers with multiple buses, the IMMBs. 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. Like an MMB and a GMMB, each processor in an IMMB is connected to exactly two buses, and consequently, each processor has a constant number of ports. We investigated the computation power of IMMBs by comparing their semigroup and prefix computation performances with that of MMBs and GMMBs. Specifically, we showed that for any constant $0 < \epsilon < 1$, one can construct an $N^{\frac{1}{2}} \times N^{\frac{1}{2}}$ 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. This significantly improves the best time complexities achieved on MMBs and IMMBs for the same computations.

Compared with the previous best complexities $O(N^{\frac{1}{8}})$ and $O(N^{\frac{1}{16}})$ achieved on a rectangular MMB and GMMB, respectively, for the same computations, our results show that IMMBs are more powerful than both MMBs and GMMBs. We also provided improved results for higher dimensional processing array We showed that 2-level d-dimensional IMMBs are more powerful than their corresponding d-dimensional GMMBs, since 2-level d-dimensional IMMBs can simulate their GMMB counterparts efficiently, while having fewer number of buses and relatively smaller diameters. This statement is not restricted to semigroup and prefix computations; it holds for solving any problem.

The IMMBs provide more design alternatives, since in addition to d, the number of dimensions, one can also select l, the number of bus levels. For a practical N value, the number of processors, both l and d can be selected as small constants no greater than 3, though theoretically it was shown that larger l and d values lead to better performances for sufficiently large N. In our examples, l is fixed for all dimensions. It is possible to select different l values for different dimensions. The best semigroup and prefix time complexities achieved on MMBs and GMMBs so far were developed for MMBs and GMMBs with large aspect ratios. We showed that l can be used to control aspect ratios, in addition to its uses for better performances. In particular, we showed how to construct IMMBs with aspect ratios equal to 1 that have very good semigroup and prefix computation performances. This is not possible for both MMBs and GMMBs.

In a real implementation of an IMMB, buses at higher levels may be implemented with larger bandwidths by either using more wires or more expensive technology to alleviate the increased congestions. This approach is similar to the fat trees proposed in [18] and implemented in the CM-5 machine [25]. Fiber optics may be used to implement buses at the highest level. Three dimensions may be preferred to two dimensions for reducing the number of processors attached to a bus.

Solving other problems using MMBs and GMMBs has been considered [5], [9], [10], [12], [14], [21]. For example, Chen et al. [10] designed an $O(N^{\frac{1}{8}}\log N)$-time median-finding algorithm for an $N^{\frac{3}{8}} \times N^{\frac{5}{8}}$ MMB, and Bhagavathi et al. [5] designed an selection algorithm with the same complexity. Chung [14] designed an $O(N^{\frac{1}{10}}\log N)$-time selection algorithm for an $N^{\frac{3}{5}} \times N^{\frac{2}{5}}$ GMMB. It is not difficult to show that the time complexities of selection algorithms using IMMBs are within $(\log N)$ times the time complexities of our semigroup and prefix algorithms using IMMBs.

Acknowledgments

The authors would like to acknowledge the following support received during this research project: US National Science Foundation (NSF) grants CCR-9211621 and CCR-9503882, the US Air Force Avionics Laboratory of Wright Laboratory grant F33615-C-2218, an Ohio Board of Regents Research Challenge grant, an Ohio Board of Regents Investment Fund Competition grant, and the AFOSR Summer Faculty Research Program (for Yi Pan); NSF grant ECS-9626215 and Louisiana grant LEQSF (1996-99)-RD-A-16 (for S.Q. Zheng); US National Aeronautics and Space Administration (NASA) and the State University of New York through the NASA/University Joint Venture (JOVE) Program, and the 1996 NASA/ASEE Summer Faculty Fellowship Program (for Keqin Li); and Australian Research Council Grant A849602031 (for Hong Shen).

References

[1] A. Aggarwal, "Optimal Bounds for Finding Maximum on Array of Processors with *k* Global Buses," *IEEE Trans. Computers,* vol. 35, no. 1, pp. 62-64, Jan. 1986

[2] S.G. Akl, *Parallel Computation: Models and Methods.* Upper Saddle River, N.J.: Prentice Hall, 1997.

[3] A. Bar-Noy and D. Peleg, "Square Meshes Are Not Always Optimal," *IEEE Trans. Computers,* vol. 40, no. 2, pp. 196-204, Feb. 1991.

[4] C. Berge, *Hypergraphs.* North-Holland, 1989.

[5] D. Bhagavathi, P.J. Looges, S. Olariu, J.L. Schwing, and J. Zhang, "Selection on Rectangular Meshes with Multiple Broadcasting," *BIT,* vol. 33, pp. 7-14, 1993.

[6] G.E. Blelloch, "Prefix Sums and Their Applications," *Synthesis of Parallel Algorithms,* J.H. Reif, ed., pp. 35-60. Morgan Kaufmann, 1993.

[7] S.H. Bokhari, "Finding Maximum on an Array Processor with a Global Bus," *IEEE Trans. Computers,* vol. 32, no. 2, pp. 133-139, Feb. 1984.

[8] D.A. Carlson, "Modified Mesh-Connected Parallel Computers," *IEEE Trans. Computers,* vol. 37, no. 10, pp. 1315-1321, Oct. 1988.

[9] D.A. Carlson, "Solving Linear Recurrence Systems on Mesh-Connected Computers with Multiple Global Buses," *J. Parallel and Distributed Computing,* vol. 8, pp. 89-95, 1990.

[10] 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 Distributed Computing,* vol. 15, pp. 79-84, 1992.

[11] 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. Parallel and Distributed Systems,* vol. 1, no. 2, pp. 241,-245, Apr. 1990.

[12] Y.C. Chen, W.T. Chen, and G.H. Chen, "Two-Variable Linear Programming on Mesh-Connected Computers with Multiple Broadcasting," *Proc. Int'l Conf. Parallel Processing,* vol. 3, pp. 270-273, 1990.

[13] K.L. Chung, "Prefix Computations on a Generalized Mesh-Connected Computer with Multiple Buses," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 2, pp. 196-199, Feb. 1995.

[14] K.L. Chung, "Fast Selection Algorithms on Generalized Mesh-Connected Computers with Multiple Buses," unpublished manuscript.

[15] O.M. Dighe, R. Vaidyanathan, and S.Q. Zheng, "The Bus-Connected Ringed Tree: A Versatile Interconnection Network," *J. Parallel and Distributed Computing,* vol. 33, pp. 189-196, 1996.

[16] S. Lakshmivarahan and S.K. Dhall, *Parallel Computing Using the Prefix Problem.* Oxford Univ. Press, 1994.

[17] F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercube.* Morgan Kaufmann, 1992.

[18] C.E. Leiserson, "Fat Trees: Universal Networks for Hardware Efficient Supercomputing," *IEEE Trans. Computers,* vol. 34, pp. 892-901, 1985.

[19] R. Miller, V.K. Prasanna-Kumar, D. Reisis, and Q.F. Stout, "Meshes with Reconfigurable Buses," *Proc. MIT Conf. Advanced Research in VLSI,* pp. 163-178, 1988.

[20] S. Olariu, J.L. Schwing, and J. Zhang, "Fundamental Algorithms on Reconfigurable Meshes," *Proc. 29th Ann. Allerton Conf. Comm., Control, and Computing,* pp. 811-820, Oct. 1991.

[21] V.P. Kumar and C.S. Raghavendra, "Array Processor with Multiple Broadcasting," *J. Parallel and Distributed Computing,* vol. 4, no. 2, pp. 173-190, Apr. 1987.

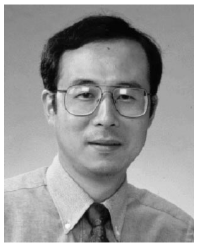
[22] M.J. Serrano and B. Parhami, "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 10, pp. 1073-1080, Oct. 1993.

[23] Q.F. Stout, "Mesh Connected Computers with Broadcasting," *IEEE Trans. Computers,* vol. 32, no. 9, pp. 826-830, Sept. 1983.

[24] Q.F. Stout, "Mesh with Multiple Buses," *Proc. 27th IEEE Symp. Foundations of Computer Science,* pp. 264-273, 1986.

[25] "TMCC, The CM-5 Technical Summary," Thinking Machines Corporation, Cambridge, Mass., 1991.

[26] S.Q. Zheng, "An Abstract Model for Optical Interconnection Networks," *Parallel Computing Using Optical Interconnection Networks,* K. Li, Y. Pan, and S.Q. Zheng, eds., pp. 139-162. Kluwer Academic, 1998.



Yi Pan received the BEng degree in computer engineering from Tsinghua University, China, in 1982, and the PhD degree in computer science from the University of Pittsburgh in 1991. Currently, he is an associate professor in the Department of Computer Science at Georgia State University. His research interests include parallel algorithms and architectures, optical communication and computing, high performance data mining, distributed computing, task scheduling, and networking. He has published more than 110 research papers, including more than 46 papers in international journals, such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Reliability*, *IEEE Communications*, *IEEE Computing in Science and Engineering*, *Journal of Parallel and Distributed Computing*, *Optical Engineering*, and *Journal of Supercomputing*. He has received many awards, including the Outstanding Scholarship Award of the College of Arts and Sciences at the University of Dayton (1999), a Japanese Society Award for the Promotion of Science Fellowship (1998), an AFOSR Summer Faculty Fellowship (1997), and two US National Science Foundation Research Opportunity Awards (1994 and 1996).

Dr. Pan is currently an area editor-in-chief of the journal of *Information*, an associate editor of the *IEEE Transactions on Systems, Man, and Cybernetics*, an editor of the journal of *Parallel and Distributed Computing Practices*, an associate editor of the *International Journal of Parallel and Distributed Systems and Networks*, and on the editorial board of *The Journal of Supercomputing*. Dr. Pan is an IEEE Computer Society Distinguished Visitor, a senior member of the IEEE, and a member of the IEEE Computer Society.

Si Qing Zheng received the PhD degree in computer science from the University of California, Santa Barbara, in 1987. After having been on the faculty of Louisiana State University for 11 years (since 1987), he joined the University of Texas at Dallas, where he is currently a professor of computer science. Dr. Zheng's research interests include algorithms, computer architectures, networks, parallel and distributed processing, telecommunication, and VLSI design. He has published extensively in these areas. He served as the program committee chairman of numerous international conferences and the editor of several professional journals.

Keqin Li received the BS degree in computer science from Tsinghua University, China, in 1985, and the PhD degree in computer science from the University of Houston in 1990. He is currently a full professor of computer science in State University of New York at New Paltz. Dr. Li's research interests are mainly in design and analysis of algorithms and parallel and distributed computing, with particular interests in approximation algorithms, parallel algorithms, job scheduling, task dispatching, load balancing, performance evaluation, dynamic tree embedding, scalability analysis, and parallel computing using optical interconnects. His pioneering work on processor allocation and job scheduling on partitionable meshes has inspired extensive subsequent work by numerous researchers and created a very active and productive research field. He has published more than 140 journal and refereed conference papers. He has also coedited six international conference proceedings and a book entitled *Parallel Computing Using Optical Interconnections* (Kluwer Academic, 1998).

Dr. Li received best paper awards at the 1996 International Conference on Parallel and Distributed Processing Techniques and Applications, 1997 IEEE National Aerospace and Electronics Conference, and 2000 IEEE International Parallel and Distributed Processing Symposium. He received a recognition award from the International Association of Science and Technology for Development in 1998. Dr. Li is a senior member of the IEEE and a member of the IEEE Computer Society and the ACM.

Dr. Hong Shen is a full professor at the School of Computing and Information Science, Griffith University, Australia. He is also the research director of Parallel Computing Unit. He has published more than 120 technical papers on algorithms, parallel and distributed computing, interconnection networks, parallel databases and data mining, multimedia systems, and networking. He has served as an editor of the *Journal of Parallel and Distributed Computing Practices*, associate editor of the *International Journal of Parallel and Distributed Systems and Networks*, and an editorial board member of the *Journal of Parallel Algorithms and Applications* and *International Journal of Computer Mathematics*. He is a member of the IEEE.