Efficient Simulation Algorithms among Processor Arrays with Broadcasting Buses

Susumu Matsumae

January 2000

Efficient Simulation Algorithms among Processor Arrays with Broadcasting Buses

Susumu Matsumae

January 2000

Dissertation submitted to
the Graduate School of Engineering Science of Osaka University
for the degree of Doctor of Engineering

Abstract

The mesh architecture has been studied as one of promising models for parallel computation. Its structure is natural for solving problems in matrix computations and image processing, and is suitable for VLSI implementation. However, since each processor can communicate with only adjacent processors in a single time step, in many cases the time complexity of an algorithm on the mesh is lower-bounded by its large diameter. This is a crucial drawback for a parallel computational model, and as a result, the mesh has been enhanced by the addition of various types of broadcasting capability. In this dissertation, we study in inter-model simulations among several of these enhanced mesh models.

Here, we consider the step-by-step simulations; we say that a mesh $\mathcal{M}$ can be simulated in T steps on a mesh $\mathcal{M}'$ if there exists an algorithm on $\mathcal{M}'$ that computes the result of an arbitrary step of $\mathcal{M}$ in T steps. From a theoretical point of view, a simulation result between two models is useful to relate the time-complexity classes of computational problems of each models. Also, it may provide a lower-bound or upper-bound for a problem on one model if the problem is well studied on the other model. From a practical point of view, a simulation algorithm provides the simulated model as a higher level programming platform for the simulating model.

First, we deal with the *reconfigurable mesh* (RM) and *mesh with multiple broadcasting* (MWMB). The RM is a processor array that consists of processors arranged to a two-dimensional grid with a reconfigurable bus system. The bus system can be used to dynamically obtain

various interconnection patterns among the processors during the execution of programs. A *horizontal-vertical RM* (HV-RM) is obtained from the general RM model, by restricting the network topology it can take to the ones that each bus segment must be along row or column. The MWMB is an enhanced mesh, which has additional broadcasting buses endowed to every row and column. We present two algorithms: 1) an algorithm that simulates the HV-RM of size $n \times n$ in $\Theta(\sqrt{n})$ steps on the MWMB of size $n \times n$, and 2) an algorithm that simulates the RM of size $n \times n$ in $O(\log^2 n)$ steps on the HV-RM of size $n \times n$. The time cost of the former algorithm is shown to be optimal in the worst case under the assumption on the processor mapping in this dissertation. Furthermore, as the application of the latter algorithm, we show that the RM of size $n \times n$ can be simulated in $O((\frac{n}{m})^2 \log n \log m)$ steps on the HV-RM of size $m \times m$ $(m < n)$. Previously, it was shown that the RM of size $n \times n$ can be simulated in $O((\frac{n}{m})^2 \log \frac{n}{m} \log m)$ steps on the RM of size $m \times m$. In terms of scaling simulation for the RM model, our algorithm is less efficient than the previously known best result. However, our algorithm only needs the power of HV-RM, which is much simpler and weaker model than the general RM model.

Next, we investigate the problem of simulating the *mesh with separable buses* (MSB) by the *mesh with partitioned buses* (MPB) and *mesh with restricted separable buses* (MRSB). The MSB and MPB are the two-dimensional mesh-connected computers which have additional broadcasting buses along every row and column. The broadcasting buses of the MSB can be dynamically sectioned into smaller bus segments of various lengths by the program control, while those of the MPB are statically partitioned in advance by a fixed length ℓ. The

MSB is equal in computational power to the HV-RM. The MRSB is a restricted model of the MSB, in which the broadcasting buses are placed only on every ℓ rows and ℓ columns, and only those processors located at the crossing point of the broadcasting buses can gain access to the buses. We show that the MSB of size $n \times n$ can be simulated in $\Theta(n^{1/3})$ steps by the MPB of size $n \times n$ when $\ell = \Theta(n^{2/3})$, and in $\Theta(\ell)$ steps by the MRSB of size $n \times n$. These time costs are shown to be optimal in the worst case under the assumption on the processor mapping.

In this dissertation we propose simulation algorithms among several enhanced mesh models. Among them, the RM is the most powerful model. In fact, it has been argued that the RM can be used as a universal chip capable of simulating any equivalent-area architecture (e.g. the pyramid, mesh of trees, hypercube, etc.) without loss in time. By combining our simulation algorithms properly, we can obtain the time costs for simulating the RM on the other models, respectively. This means that our results give the upper bounds for the HV-RM, MWMB, MSB, MPB, and MRSB to simulate other equivalent-area architectures.

List of Publications

1. R. Sasada, S. Matsumae, and N. Tokura: A Sorting Algorithm on a Fixed Size Reconfigurable Mesh, *IEICE Technical Report*, COMP97-76, pp.25–32, in Japanese (Dec. 1997).

2. S. Matsumae, R. Sasada, and N. Tokura: A Sorting Algorithm on a Small Size Reconfigurable Mesh, *IEICE Technical Report*, COMP97-99, pp.97–104, in Japanese (Mar. 1998).

3. R. Sasada, S. Matsumae, and N. Tokura: A Convex Hull Algorithm on a Fixed Size Reconfigurable Mesh, *IEICE Technical Report*, COMP97-100, pp.105–112, in Japanese (Mar. 1998).

4. S. Matsumae and N. Tokura: Simulation Algorithms among Enhanced Mesh Models, *Technical Report 99-ICS-1*, Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University (Mar. 1999).

5. S. Matsumae and N. Tokura: Simulating a Mesh with Separable Buses by a Mesh with Partitioned Buses, *Proc. of 1999 International Symposium on Parallel Architectures, Algorithms and Networks* (*I-SPAN'99*), pp.198–203, IEEE CS press (June 1999).

6. S. Matsumae and N. Tokura: Simulation Algorithms among Enhanced Mesh Models, *IEICE Transactions on Information and Systems*, Vol.E82–D, No.10, pp.1324–1337 (Oct. 1999).

7. S. Matsumae and N. Tokura: Simulating a Mesh with Separable Buses, *Transactions of Information Processing Society of Japan*, Vol.40, No.10, pp.3706–3714 (Oct. 1999).

Acknowledgments

The author has been fortunate to receive assistance from many people. He would especially like to express his gratitude to his supervisor Professor Nobuki Tokura for his continuous guidance and encouragement. The author has also received precious advice from Professors of the Graduate School of Engineering Science, Osaka University. Among them, he would like to extend his gratitude to Professor Toshinobu Kashiwabara and Professor Ken'ichi Hagihara for their valuable suggestions and criticisms of this dissertation.

The author would also like to thank Professor Yoshihiro Tsujino of Kyoto Institute of Technology and Mr. Ryoji Sasada of Fuji Photo Film Co., Ltd. for their discerning comments and worthy discussions.

The author would like to thank Mr. Chan Wei Siang, my friend, for his careful reading of the draft of this dissertation. His valuable advice helped a lot to complete this dissertation.

The author would also like to express hearty thanks to Mrs. Masuyo Miyashita, Mrs. Yukiko Tanobe, and Miss Tomoko Sumiyoshi for their kind support. He is also grateful to the staffs and students of Laboratory of Computer Languages, Graduate School of Engineer Science, Osaka University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The mesh architecture has been studied as one of promising models for parallel computation. The two-dimensional mesh-connected computer (mesh for short) is a processor array that consists of processors arranged in a two-dimensional grid. Each processor is connected via bi-directional unit-time communication links to four of its adjacent processors. Such structure is natural for solving problems in matrix computations and image processing, and many algorithms have been designed on this model. Also, the mesh structure is suitable for VLSI implementation and allows a high degree of integration.

The main drawback of the mesh is its large communication diameter. That is, on the mesh of size $n \times n$, it requires $\Omega(n)$ steps to move data from one end to the other, since each processor can communicate with only adjacent processors in a single time step. This leads many cases whereby the time complexity of an algorithm on the mesh is lower-bounded by its diameter. To overcome this problem, the mesh model has been enhanced by the addition of various types of broadcasting buses. In this dissertation, we study in inter-model simulations

among several of these enhanced mesh models.

First, we deal with the *reconfigurable mesh* (RM) and *mesh with multiple broadcasting* (MWMB). The RM is a mesh enhanced with a reconfigurable bus system [29, 23, 3]. The bus system can be used to dynamically obtain various interconnection patterns among the processors during the execution of programs. The processors in the same interconnected fragment can be seen as the ones which own a unit-time broadcasting bus over the processors. According to the restrictions put on the interconnection patterns, the RM is classified into several submodels, such as *horizontal-vertical RM* (HV-RM, in which each bus segment must be along row or column) and *linear RM* (LRM, in which each bus segment must be "linear", i.e., no bus-branching is allowed). The MWMB is an enhanced mesh, which has additional broadcasting buses along with every row and column [27, 24, 5, 6, 10, 7]. Unlike the RM, the broadcasting buses in the MWMB model are static, i.e., their connection pattern cannot be dynamically changed. Previously, many researchers compared these models with other models (e.g. the PRAM, pyramid, mesh of trees, hypercube, reconfigurable network, branching program, and so forth) by giving inter-model simulation algorithms between them [19, 4, 2, 12, 28, 20, 26]. But there had been no result among the RM, HV-RM, and MWMB.

In this dissertation, we show the following two algorithms: 1) an algorithm that simulates the HV-RM of size $n \times n$ in $\Theta(\sqrt{n})$ steps on the MWMB of size $n \times n$, and 2) an algorithm that simulates the RM of size $n \times n$ in $O(\log^2 n)$ steps on the HV-RM of size $n \times n$. As for the former algorithm, we show that its time-complexity is optimal in the worst case under the assumption on the processor mapping in this

dissertation. The advantage of the former algorithm is as follows. On the MWMB model, it is not easy to design an algorithm using the popular *divide-and-conquer strategy*, for writing such an algorithm on this model involves ensuring a conflict free allocation of broadcasting buses to submeshes. On the other hand, on the HV-RM model where row and column buses can be freely separated into sub-buses, it is not necessary to confuse the one who writes the algorithm with such a bus allocation problem. Therefore, it is practical and useful to have an algorithm that simulates an HV-RM program on the MWMB model. In other words, we can view the HV-RM model as a higher level programming platform for the MWMB. As for the latter algorithm, since the RM is a very powerful computational model and there are many efficient algorithms on it (the interested reader should refer to [22]), we can also obtain many efficient algorithms for the HV-RM, by just simulating those RM algorithms. To apply this simulation algorithm, we next consider the problem of simulating an RM using a smaller RM. This sort of problem is called *scaling simulation* or *self-simulation* [16, 3, 17, 21, 9]. Here, we show that the RM of size $n \times n$ can be simulated in $O(\,(\frac{n}{m})^2 \log n \log m\,)$ steps on the HV-RM of size $m \times m$ ($m < n$). Although this time cost is less efficient than $O(\,(\frac{n}{m})^2 \log \frac{n}{m} \log m\,)$ steps obtained by Fernández-Zepeda et al. in [9], our algorithm only needs the power of HV-RM, which is simpler and weaker model than that used in [9].

Next, we consider simulation problems among the *mesh with separable buses* (MSB), *mesh with restricted separable buses* (MRSB), and *mesh with partitioned buses* (MPB). The MSB is equal in computational power to the HV-RM, and is more powerful than the MRSB and MPB. The MSB is an enhanced mesh model proposed by Maeba et

al. in [15]. They considered sectioning the broadcasting buses of the MWMB by inserting the processor-controlled switches into the buses. The broadcasting buses of the MSB, called *separable buses*, can be dynamically divided into smaller bus segments by the program control. The MRSB [25] is a restricted model of the MSB in which the separable buses are placed only on every ℓ rows and ℓ columns, and only those processors located at the crossing point of the broadcasting buses can gain access to the buses. The MPB was proposed in [8, 14]. Like the MWMB and MSB, the MPB broadcasts along every row and column. The broadcasting buses of the MPB have no sectioning switch inserted, but are partitioned in advance by a fixed length ℓ.

In this dissertation, we show that the MSB of size $n \times n$ can be simulated in $\Theta(n^{1/3})$ steps by the MPB of size $n \times n$ when $\ell = \Theta(n^{2/3})$, and in $\Theta(\ell)$ steps by the MRSB of size $n \times n$. Furthermore, these time costs are shown to be optimal in the worst case under the assumption on the processor mapping. Theoretically, since we have shown that the MSB (= the HV-RM in computational power) of size $n \times n$ can simulate the RM of size $n \times n$ in $O(\log^2 n)$ steps, we can say that any problem that can be solved in time T by the RM of size $n \times n$ can be solved in $O(TT' \log^2 n)$ steps by the MPB or MRSB of size $n \times n$, where T' is the time cost of simulating the MSB of size $n \times n$. Practically, compared to the MSB, the number of broadcasting buses or that of switch elements inserted to the buses can be rather small on these simulating models, and thus it is expected that the scalability of them is better than that of the MSB. Also, considering the propagation delay of a broadcasting bus introduced by the length of the bus (i.e., signal propagation delay) and those switch elements inserted to the bus (i.e.,

device propagation delay), the propagation delay of the buses in the MPB and MRSB can be small in practice, and hence we consider that our simulation algorithms are useful when the mesh size becomes so large that we cannot neglect the delay.

Figure 1.1 summarizes the inter-model simulation results obtained in this dissertation.

This dissertation is organized as follows. In Chapter 2 we discuss the simulation problems of the RM, HV-RM, and MWMB. In Chapter 3 we present simulation algorithms for the MSB, MPB, and MRSB. And in Chapter 4 we summarize our results.

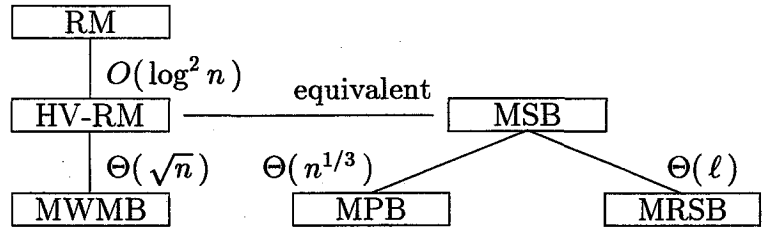


Figure 1.1: Enhanced mesh models and simulation costs (the mesh size is of $n \times n$).

Chapter 2

Simulation of Reconfigurable Mesh

In this chapter, we deal with the *reconfigurable mesh* (RM), *horizontal-vertical reconfigurable mesh* (HV-RM), and *mesh with multiple broadcasting* (MWMB).

The simulation problems related to these models have been studied well. Previously, Ben-Asher et al. [4] investigated the relation among the RN (reconfigurable network, a more flexible version of the RM, whose underling network topology need not to be the mesh), PRAM, and BP (branching program). In [2], they also presented several inter-model simulations among the RM and its various submodels. Lin et al. [12] proposed simulations of the MWMB and B-RM (basic RM, a variant of the HV-RM) by the PRAM. Trahan et al. [28] established a hierarchy of powers of the RMBM (reconfigurable multiple bus machine), RN, and PRAM, focusing on the ability to "segment" and/or "fuse" buses and various bus-access rules. Miller et al. [20] compared the RMESH (a variant of the RM, in which no *cross-over* of buses is allowed) with the ordinary mesh, pyramid, mesh of trees, hypercube,

and PRAM. Shi et al. [26] demonstrated the equivalency of the RMESH with 8-connectivity and the RM by giving simulations between them. But there had been no result among the RM, HV-RM, and MWMB. [1]

First, we show the following two algorithms:

1. An algorithm that simulates the HV-RM of size $n \times n$ in $\Theta(\sqrt{n})$ steps on the MWMB of size $n \times n$,
2. An algorithm that simulates the RM of size $n \times n$ in $O(\log^2 n)$ steps on the HV-RM of size $n \times n$.

As for the former algorithm, we show that the time-complexity is optimal in the worst case under the assumption on the processor mapping we take here.

As the applications of the above two simulation algorithms, we also consider simulating an RM on a smaller RM. This sort of problem is called *scaling simulation* or *self-simulation.* Clearly, it takes $\Omega((\frac{n}{m})^2)$ steps to simulate $n \times n$ processors using $m \times m$ processors $(m < n)$. Previously, Maresca [16] showed that the PPA (Polymorphic Processor Arrays, a variant of the HV-RM) has an optimal self-simulation. Ben-Asher et al. [3] proposed an $O((\frac{n}{m})^2 \log n \log \frac{n}{m})$ step self-simulation algorithm for the general RM, and optimal self-simulation algorithms for the HV-RM and LRM, respectively. Matias and Schuster [17] presented the randomized algorithm for the RM, which runs in $O((\frac{n}{m})^2 + \log n \log\log m)$ steps with high probability on the LRM and uses the "ARBITRARY" concurrent write rule. Murshed and Brent [21] considered certain global restriction for the RM,

[1] Unlike the result for the RMESH in [20], the component labeling algorithm for a binary image does not imply the straightforward solution for the RM here. Note that the RM allows *cross-over* among buses, while the RMESH does not.

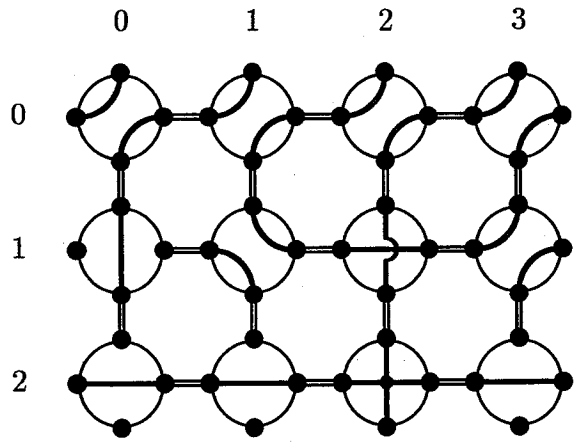
and showed that the RM with that restriction enjoys optimal self-simulation. Recently, Fernández-Zepeda et al. [9] improved the result given in [3], and presented an $O((\frac{n}{m})^2 \log \frac{n}{m} \log m)$ step algorithm for the RM self-simulation.

In this chapter we show that the RM of size $n \times n$ can be simulated in $O((\frac{n}{m})^2 \log n \log m)$ steps on the HV-RM of size $m \times m$, and in $O((\frac{n}{m})^2 \sqrt{m} \log n \log m)$ steps on the MWMB of size $m \times m$ $(m < n)$. Although the time cost $O((\frac{n}{m})^2 \log n \log m)$ of our HV-RM algorithm is less efficient than that obtained by Fernández-Zepeda et al. in [9], our algorithm only needs the power of HV-RM, which is simpler and weaker model than that used in [9].

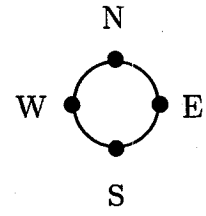
This chapter is organized as follows. Section 2.1 describes the RM, HV-RM, and MWMB models. Section 2.2 and 2.3 discuss the simulation and related problems. Section 2.4 presents an algorithm that simulates the HV-RM time-optimally on the MWMB. Section 2.5 gives preliminary lemmas for Section 2.6, and Section 2.6 presents an algorithm that simulates the RM on the HV-RM. Section 2.7 discusses the applications of our algorithms, including the simulation of an RM by a smaller enhanced mesh.

2.1 Models

A *reconfigurable mesh* (RM) is the mesh enhanced with a reconfigurable bus system. An $m \times n$ RM is the RM which consists of mn identical SIMD processors with m rows and n columns (Figure 2.1). The processor located in row i and column j is denoted as PE$[i, j]$, and is assumed to know its coordinates i and j $(0 \leq i < m,\ 0 \leq j < n)$. We do not



A 3 × 4 RM



A single processor and its 4 ports

Figure 2.1: A 3 × 4 RM and a single processor.

assume the existence of shared memory, and only assume that every processor has local memory.

Every processor has four ports, N (north), S (south), E (east), and W (west). The port S of PE$[i,j]$ is connected via a static external bus to the port N of PE$[i+1,j]$ $(0 \leq i < m-1, 0 \leq j < n)$, and the port E of PE$[i,j]$ is to the port W of PE$[i,j+1]$ $(0 \leq i < m, 0 \leq j < n-1)$. In each processor, ports can be connected in pairs by internal buses, and the connection pattern can be dynamically changed during the execution of the programs. The connection pattern of local ports in a processor is represented as partition of the ports of the processor. For example, when only port N and port S are connected, it is written as {NS,E,W}. There are altogether 15 possible partitions: {NSEW}, {NEW,S}, {NSE,W}, {N,SEW}, {NSW,E}, {NW,SE}, {NE,SW}, {NW,S,E}, {N,SW,E}, {NE,S,W}, {N,SE,W},

{NS,EW}, {N,S,EW}, {NS,E,W}, and {N,S,E,W}. Each interconnected bus segment forms a broadcasting bus whose communication latency is assumed to be a constant time.

A single time step of the RM is composed of the following 4 substeps:

1. **BUS substep**: Every processor changes the connection pattern of its local ports by local decision.

2. **WRITE substep**: Along each bus, one or more processors on the bus send data to the bus. These processors are called the *speakers* of the bus.

3. **READ substep**: Several of the processors connected to a bus read the data transmitted on the bus by the speaker(s). These processors are called the *listeners* of the bus.

4. **COMPUTE substep**: Some local computation is performed by every processor.

We assume each of the 4 substeps is executed in a constant time. The data transmitted on a bus in a single time step is assumed to be a k-bit binary data for some constant k, where $k \geq \log mn$.

We assume the CRCW bus model for the RM here. Concurrent writes are resolved by the COMMON-COLLISION rule. If there is no speaker on a bus, the listeners of the bus receive a special value ϕ. If there is one or more speakers on a bus, all sending the same data, then the listeners of the bus receive the data. If there is more than one speaker on a bus, two or more speakers sending different data, then the listeners of the bus receive *error*.

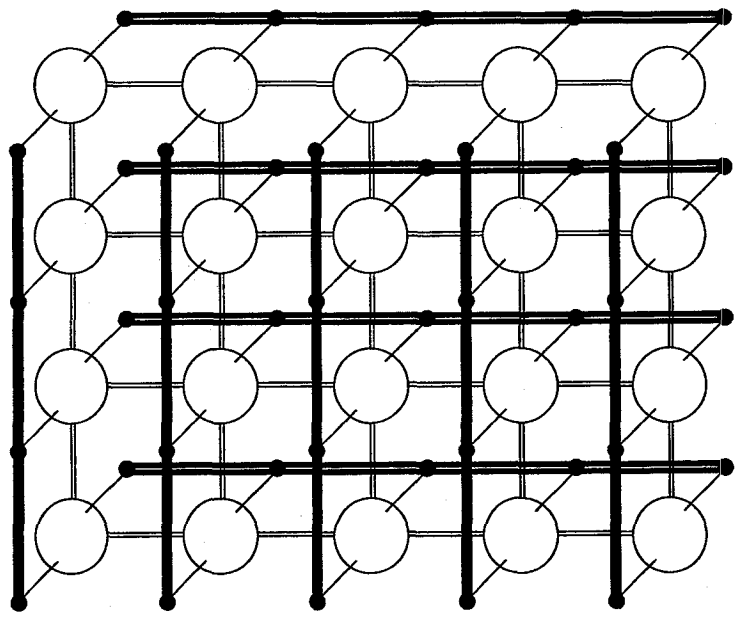


Figure 2.2: A 4 × 5 MWMB.

A *horizontal-vertical RM* (HV-RM) is a restricted model of the RM, in which only {N,S,E,W}, {N,S,EW}, {NS,E,W}, and {NS,EW} are allowed as the connection pattern of local ports in each processor.

A *mesh with multiple broadcasting* (MWMB) is the enhanced mesh which has broadcasting buses along with every row and column. An $m \times n$ MWMB is the MWMB which consists of mn identical SIMD processors with m rows and n columns. Except for the broadcasting capability in each row and column, an MWMB is the same as the ordinary mesh which has local buses connecting adjacent processors. See Figure 2.2. As in the RM, the processor located in row i and column j is denoted as PE$[i, j]$, and is assumed to know its coordinates i and j ($0 \leq i < m$, $0 \leq j < n$). We assume that every processor has

local memory, but without the existence of shared memory.

A single time step of the MWMB is composed of the following 4 substeps: 1) LOCAL COMM, 2) WRITE, 3) READ, and 4) COMPUTE substeps. In the LOCAL COMM substep, every processor communicates with its adjacent processors via local buses. The other substeps are the same as those in the RM model. We assume that each of the 4 substeps is executed in a constant time. The data transmitted on a bus in a single time step is assumed to be a k-bit binary data for some constant k, where $k \geq \log mn$. In this model, we assume there is no write conflict on a bus, i.e., the bus accessing capability is of the CREW. If there is no speaker on a bus, the listeners of the bus receive the special value ϕ.

The MWMB can be seen as a weaker model of the HV-RM. The local buses (resp. broadcasting buses) of the MWMB can be mimicked on the HV-RM, by letting every processor of the HV-RM statically take {N,S,E,W} (resp. {NS,EW}) as the bus configuration. Thus, any step of an MWMB can be simulated in a constant time on an HV-RM of the same size.

2.2 Problem Definitions

We define *simulation* as follows.

Definition 1 (Simulation) *Let $\mathcal{M}$ and $\mathcal{M}'$ be two enhanced meshes. We say that $\mathcal{M}'$* simulates *a step of $\mathcal{M}$ with slowdown C if there is an algorithm running in $O(C)$ steps on $\mathcal{M}'$ that achieves the effect of an arbitrary step of $\mathcal{M}$.* ■

The effect of a step of $\mathcal{M}$ is characterized by the contents of registers and memory cells of each processor of $\mathcal{M}$ before and after the execution of the step.

In this chapter, we assume the following.[2] Here, $\mathcal{M}$ denotes a simulated mesh, and $\mathcal{M}'$ a simulating mesh.

1. The size of $\mathcal{M}$ and that of $\mathcal{M}'$ are the same.

2. Each PE$[i,j]$ of $\mathcal{M}$ is mapped to PE$[i,j]$ of $\mathcal{M}'$. An injective mapping from registers and memory cells of a processor of $\mathcal{M}$ into those corresponding processor of $\mathcal{M}'$ is defined appropriately.

3. The computing power of processors and the bus bandwidth are equivalent in both $\mathcal{M}$ and $\mathcal{M}'$, except that every processor of $\mathcal{M}'$ has some amount of extra registers and memory cells for the simulation.

With these assumptions, the main difference between a simulated mesh and a simulating one becomes only their broadcasting capabilities. Hence, to devise a simulation algorithm, we only need to consider how to achieve the bus structure of a simulated mesh on a simulating one.

Next, we define a graph called *port connectivity graph*, and after that, consider labeling the connected components of the graph. To avoid confusing the reader, from now on we write PE$_{\mathcal{M}}[i,j]$ for denoting PE$[i,j]$ of $\mathcal{M}$ where $\mathcal{M}$ is an enhanced mesh.

[2] The assumption 1 and 2 are not applied to Lemma 14, Theorem 4, and Corollary 3 in Section 2.7.

Definition 2 (Port Connectivity Graph) *Let $\mathcal{M}$ be the RM of size $m \times n$, and let $\mathcal{S}$ a step of $\mathcal{M}$. Then, the port connectivity graph (pc-graph) of $(\mathcal{M},\mathcal{S})$ is defined as an undirected graph $G = (V, E)$ such that*

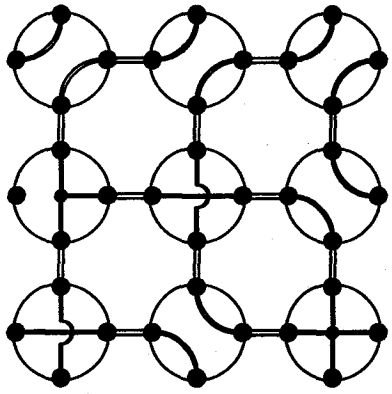
$$V = \{\,(i,j,X) \mid (i,j) \in ID_{\mathcal{M}},\ X \in \{N,S,E,W\}\,\}$$
$$\textit{where } ID_{\mathcal{M}} = \{\,(i,j) \mid 0 \le i < m,\ 0 \le j < n\,\},$$

$$E = E^{\text{ext}}_{\mathcal{M}} \cup E^{\text{int}}_{\mathcal{M},\mathcal{S}}$$
where
$$E^{\text{ext}}_{\mathcal{M}} = \{\,\{(i,j,S),\ (i+1,j,N)\} \mid 0 \le i < m-1,\ 0 \le j < n\,\}$$
$$\cup$$
$$\{\,\{(i,j,E),\ (i,j+1,W)\} \mid 0 \le i < m,\ 0 \le j < n-1\,\},$$
$$E^{\text{int}}_{\mathcal{M},\mathcal{S}} = \{\,\{(i,j,X),\ (i,j,Y)\}$$
$$\mid \textit{port } X \textit{ and } Y \textit{ of } PE_{\mathcal{M}}[i,j] \textit{ is connected at the BUS substep of } \mathcal{S}\,\}.$$

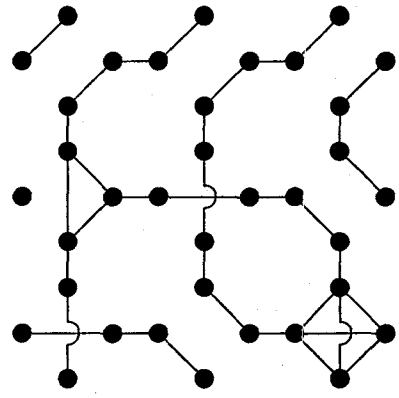
The pc-graph of $(\mathcal{M},\mathcal{S})$ is written as $PCG\,(\mathcal{M},\mathcal{S})$. ■

Intuitively, the graph $PCG\,(\mathcal{M},\mathcal{S})$ represents the port-to-port connection among ports of processors of $\mathcal{M}$ established at the BUS substep of $\mathcal{S}$ (Figure 2.3). Each vertex $(i,j,X) \in V$ corresponds to port X of $\text{PE}_{\mathcal{M}}[i,j]$. Edges in $E^{\text{ext}}_{\mathcal{M}}$ stand for the port-to-port connections via external buses, and edges in $E^{\text{int}}_{\mathcal{M},\mathcal{S}}$ for the connections by internal buses. Note that pc-graphs are defined on the RM, not on the MWMB model.

In the literature, connected component labeling is usually done by labeling vertices in each component C with the smallest index number of all the vertices in C. In this chapter, we use the labels defined as follows.



The bus configuration of $\mathcal{M}$ at the BUS substep of $\mathcal{S}$



$PCG\,(\mathcal{M},\mathcal{S})$

Figure 2.3: A pc-graph example ($\mathcal{M}$ is a 3×3 RM).

Definition 3 (Initial Label) *Let $G = (V, E)$ be $PCG\,(\mathcal{M},\mathcal{S})$. For each $(i, j, X) \in V$, we associate a label, denoted as $ini_{\mathcal{M},\mathcal{S}}(i, j, X)$, in such a way that*

$$ini_{\mathcal{M},\mathcal{S}}(i,j,X) = \begin{cases} x & \text{if } PE_{\mathcal{M}}[i,j] \text{ is a speaker sending a value } x \text{ through} \\ & \text{port } X \text{ at the WRITE substep of } \mathcal{S}, \\ \top & \text{otherwise.} \end{cases}$$

These labels are called initial labels. Here, $\top$ is assumed to be a special value distinct from any other labels. ■

A total order $\preceq$ on the initial labels is defined as a natural extension of the total order on binary numbers, by regarding $\top$ as the greatest element. The binary operator which selects the smaller value (w.r.t. $\preceq$) of the given two arguments is denoted as $\downarrow$. For a label set T, $\downarrow T$ denotes the minimum (w.r.t. $\preceq$) among the elements in T.

Let $v \overset{G}{\leftrightarrow} v'$ denote that vertices v and v' are connected in graph G. Then, we can now define the connected component labeling problem, as follows.

Definition 4 (LCC-PCG Problem) *The problem of labeling connected component of pc-graph (LCC-PCG problem) is defined as follows:*

Input: A pc-graph $G = (V, E)$ is given. Every vertex in V is labeled with its initial label.

Output: For each $v \in V$, v is labeled by $\downarrow\{$ initial label of $v' \mid v \overset{G}{\leftrightarrow} v' \}$.

The labels obtained as output are called component labels. Let $G' = (V', E')$ be a reduced graph of G obtained by restricting the vertex set into $V' \subseteq V$. For each $v \in V'$, the value $\downarrow\{$initial label of $v' \mid v \overset{G'}{\leftrightarrow} v' \}$ is called its local component label within G'. ■

We solve the LCC-PCG problem on enhanced meshes. The input and output conditions for such LCC-PCG algorithms in this chapter is fixed as follows.

Definition 5 (LCC-PCG Problem on Mesh) *Let $G = (V, E)$ be PCG $(\mathcal{M}, \mathcal{S})$, and $\mathcal{M}'$ be a mesh whose size is identical to that of $\mathcal{M}$. Then, the LCC-PCG problem of G on $\mathcal{M}'$ is defined as follows:*

Input: For each $(i, j, X) \in V$, the edges connected to (i, j, X) and the initial label of (i, j, X) are stored in $PE_{\mathcal{M}'}[i, j]$.

Output: For each $(i, j, X) \in V$, the component label of (i, j, X) is obtained at $PE_{\mathcal{M}'}[i, j]$.

■

2.3 Simulation by LCC-PCG Algorithm

In this section, we show that the simulation problem can be reduced to the LCC-PCG problem.

First, we consider a virtual model of the RM, called MIN RM. A MIN RM is the RM in which the bus accessing capability is of MIN bus model. In the MIN bus model, if there is one or more speakers on a bus, the listeners of the bus receive the minimum among the values sent by the speakers of the bus, otherwise the listeners of the bus receive a special value ϕ. Similarly, the MIN HV-RM is defined as a variant of the HV-RM model, whose bus model is of MIN. To avoid confusion, we write COMMON-COLLISION RM (resp. COMMON-COLLISION HV-RM) for the RM (resp. HV-RM) defined in Section 2.1. Since the bus accessing capability has no influence on the arguments in Section 2.2, the simulation problem, pc-graph, and LCC-PCG problem can also be defined on the MIN RM model in the same way.

Then, we can state the following lemma.

Lemma 1 *Let $\mathcal{M}$ be a* MIN *RM (HV-RM) of size $m \times n$. Then, the mesh of size $m \times n$ can simulate a step of $\mathcal{M}$ with slowdown $O(C)$ if there exists an algorithm $\mathcal{A}$ that solves LCC-PCG problem of PCG $(\mathcal{M}, \mathcal{S})$ for any $\mathcal{S}$ in C steps on the mesh.*

Proof: Assume there exists such an algorithm $\mathcal{A}$, and let $\mathcal{M}'$ denote the mesh on which $\mathcal{A}$ runs. Then, given $\mathcal{S}$, by solving the LCC-PCG problem of $PCG(\mathcal{M}, \mathcal{S})$, $\mathcal{M}'$ can simulate the effect of BUS, WRITE, and READ substeps of $\mathcal{S}$. Note that $\mathcal{M}'$ can generate necessary data for the input condition of the LCC-PCG problem in a constant time.

As for the COMPUTE substep, $\mathcal{M}'$ has only to perform corresponding computations in each processor. As a whole, this simulation takes $O(C)$ steps, and thus the conclusion follows. ■

The MIN RM (resp. MIN HV-RM) can simulate any step of the COMMON-COLLISION RM (resp. COMMON-COLLISION HV-RM) in a constant number of steps (see [18] for the details, the proof is similar to the simulation of the COLLISION bus model by the PRIORITY one). This fact, coupled with Lemma 1, implies the following corollary.

Corollary 1 *Let $\mathcal{M}$ be the* COMMON-COLLISION *RM (HV-RM) of size $m \times n$. Then, the mesh of size $m \times n$ can simulate a step of $\mathcal{M}$ with slowdown $O(C)$ if there exists an algorithm $\mathcal{A}$ that solves LCC-PCG problem of PCG $(\mathcal{M}, \mathcal{S})$ for any $\mathcal{S}$ in C steps on the mesh.* ■

In the following, when just written as RM or HV-RM, we do not distinguish whether it is of MIN or of COMMON-COLLISION bus model. In such a case, the proposed assertion holds for both bus models.

2.4 Simulation of HV-RM by MWMB

In this section, we show that the MWMB of size $n \times n$ can simulate a step of the HV-RM of size $n \times n$ with slowdown $\Theta(\sqrt{n})$. This slowdown is time-optimal in the worst case.

First, we show the lower bound for the problem. The problem of simulating the HV-RM on the MWMB can be reduced to that of computing local minima among data items dispersed one item per processor on the MWMB. In the case of just computing the minimum of all the

data items, the task can be completed in $O(n^{1/3})$ steps on the MWMB of size $n \times n$ [24]. But in the case of computing local minima, the task needs $\Omega(\sqrt{n})$ steps, which can be shown as a corollary of the following lemma.

Lemma 2 *Any algorithm that correctly simulates a step of the HV-RM of size $n \times n$ on the MWMB of size $n \times n$ must take $\Omega(\sqrt{n})$ steps in the worst case.*

Proof: Let $\mathcal{M}$ be the HV-RM of size $n \times n$ (simulated mesh), and let $\mathcal{M}'$ the MWMB of the same size (simulating mesh). Consider the step $\mathcal{S}$ of $\mathcal{M}$ which consists of the following substeps:

1. *BUS substep*: $\mathrm{PE}_{\mathcal{M}}[i,j]$ takes {N,S,EW} as its internal bus configuration if $(j \bmod \sqrt{n}) \neq 0$, otherwise takes {N,S,E,W} $(0 \leq i,j < n)$.

2. *WRITE substep*: $\mathrm{PE}_{\mathcal{M}}[i,j]$ sends the content of variable a to port E if $(j \bmod \sqrt{n}) = 0$ $(0 \leq i,j < n)$.

3. *READ substep*: $\mathrm{PE}_{\mathcal{M}}[i,j]$ receives data through port W and stores the data in variable b if $(j \bmod \sqrt{n}) \neq 0$ $(0 \leq i,j < n)$.

Here, we do not take into account the COMPUTE substep of $\mathcal{S}$. At the BUS substep of $\mathcal{S}$, $n\sqrt{n}$ horizontal broadcasting buses are formed over $\mathcal{M}$. Along each bus, the leftmost processor of the bus is a single speaker of the bus sending the content of its local variable a to its $\sqrt{n}-1$ listeners. We call those values to be transmitted as *a-values*. Note that there are possibly $n\sqrt{n}$ different a-values. In the following, we show

that it must take $\Omega(\sqrt{n})$ steps to simulate $\mathcal{S}$ on $\mathcal{M}'$, no matter what algorithm is used.

Let $\mathcal{A}$ be any algorithm that correctly simulates $\mathcal{S}$ on $\mathcal{M}'$. We count how many steps it needs as follows. Since the simulation is performed on the MWMB model, the simulating processors communicate via local and broadcasting buses. Then, since every processor that initially holds an a-value is different from those processors which will receive the value, every a-value must be transmitted on those buses before $\mathcal{A}$ terminates. With these observations, we count the necessary steps for $\mathcal{A}$, by considering the following two cases:

Case 1: There exists an a-value transmitted only through local buses during the simulation.

Case 2: There is no a-value transmitted only through local buses during the simulation.

In Case 1, since the distance between the processor initially holding the a-value and the most distant processor which will receive it is $\sqrt{n}-1$, $\mathcal{A}$ must take at least $\sqrt{n}-1$ steps before it terminates. On the other hand, in Case 2, every a-value must be transmitted via broadcasting buses at least once during the simulation. Since the total number of a-value is $n\sqrt{n}$ and the number of data which can be transmitted through broadcasting buses in a single step is at most $2n$ (the total number of broadcasting buses of $\mathcal{M}'$), $\mathcal{A}$ must take at least $\sqrt{n}/2$ steps before it terminates. In either case, $\mathcal{A}$ needs $\Omega(\sqrt{n})$ steps. Thus, the conclusion follows. ■

Next, we show an algorithm on the MWMB of size $1 \times n$.

Lemma 3 *Let $G = (V, E)$ be PCG $(\mathcal{M}, \mathcal{S})$ where $\mathcal{M}$ is the HV-RM of size $1 \times n$ and $\mathcal{S}$ is a step of $\mathcal{M}$. Then, the LCC-PCG problem of G can be solved in $O(\sqrt{n})$ steps on the MWMB of size $1 \times n$.*

Proof: Let $\mathcal{M}'$ be the MWMB of size $1 \times n$, on which the LCC-PCG problem of G is solved. First, divide $\mathcal{M}'$ into $\sqrt{n}$ disjoint submeshes $\mathcal{M}'_p$ of size $1 \times \sqrt{n}$ $(0 \leq p < \sqrt{n})$, and for each submesh $\mathcal{M}'_p$, we associate a graph $G_p = (V_p, E_p)$ such that G_p is a reduced graph of G obtained by restricting the vertex set into $\{(0, j, X) \in V \mid \mathrm{PE}_{\mathcal{M}'}[0, j] \text{ is in } \mathcal{M}'_p\}$. For each G_p, let v_p^{L} (resp. v_p^{R}) denote the leftmost vertex (resp. rightmost vertex) of G_p, i.e., $(0, p\sqrt{n}, \mathrm{W}) \in V_p$ (resp. $(0, (p+1)\sqrt{n} - 1, \mathrm{E}) \in V_p$). Then, we can compute the component labels for every vertex in V, in the following three phases:

Phase 1: Each $\mathcal{M}'_p$ computes the local component label within G_p for every vertex in V_p, and determines whether $v_p^{\mathrm{L}} \overset{G_p}{\leftrightarrow} v_p^{\mathrm{R}}$ holds.

Phase 2: Compute component labels of every v_p^{L} and v_p^{R} $(0 \leq p < \sqrt{n})$.

Phase 3: Each $\mathcal{M}'_p$ computes the component labels of those vertices in V_p connected to v_p^{L} and/or v_p^{R}.

Phase 1 can be performed in $O(\sqrt{n})$ steps, by sequentially scanning labels from left to right and then from right to left in each $\mathcal{M}'_p$ using regular mesh connections. As for Phase 2, by sequentially broadcasting the local component labels and connectivity information obtained at

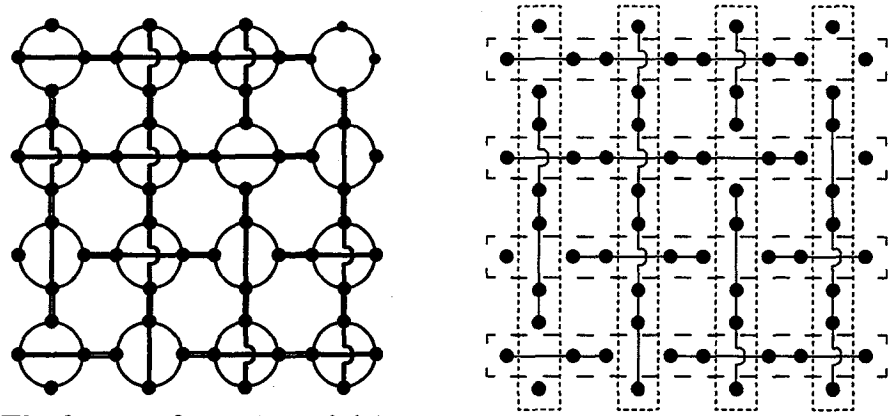


Figure 2.4: A pc-graph example ($\mathcal{M}$ is a 4×4 HV-RM).

Phase 1 of those v_p^{L} and v_p^{R} $(0 \leq p < \sqrt{n})$, the labels of those vertices can be updated to the component labels in $O(\sqrt{n})$ time. And finally, Phase 3 can be done in $O(\sqrt{n})$ steps similarly to Phase 1. Thus, the conclusion follows. (See [18] for the details[3] .) ■

Here, we deal with the LCC-PCG problem of $PCG\,(\mathcal{M},\mathcal{S})$ where $\mathcal{M}$ is of HV-RM model. Hence, for each vertex (i,j,X), if X is N or S, then its component label can be determined by checking the reduced graph composed of those vertices (i',j',X') satisfying $j' = j$. Similar argument holds for vertex (i,j,X) such that X is E or W. See Figure 2.4. In Figure 2.4, the connected components of $PCG\,(\mathcal{M},\mathcal{S})$ can be determined by checking each of the subgraphs (graphs enclosed by

[3] The algorithm in [18], however, does not follow this strategy as it is, although its concept is essentially the same.

dashed boxes) independently.

Hence, the following lemma holds.

Lemma 4 *Let $G = (V, E)$ be PCG $(\mathcal{M}, \mathcal{S})$ where $\mathcal{M}$ is the HV-RM of size $n \times n$ and $\mathcal{S}$ is a step of $\mathcal{M}$. Then, LCC-PCG problem of G can be solved in $O(\sqrt{n})$ steps on the MWMB of size $n \times n$.*

Proof: The algorithm proceeds in two phases. In the first phase, compute the component labels for each $(i, j, X) \in V$ such that X is E or W in each row in parallel. And in the second phase, compute the component labels for each $(i, j, X) \in V$ such that X is N or S in each column in parallel similarly. Each phase can be performed in $O(\sqrt{n})$ steps from Lemma 3, and thus the conclusion follows. ■

We can now state the main theorem of this section.

Theorem 1 *The MWMB of size $n \times n$ can simulate a step of the HV-RM of size $n \times n$ with slowdown $\Theta(\sqrt{n})$. This slowdown is time-optimal in the worst case.*

Proof: By Lemma 1, 2, 4, and Corollary 1. ■

2.5 Properties of PC-Graph

In this section, we show some properties of the pc-graph, and prove some lemmas using them. Our main goal here is to obtain Corollary 2, which plays an important role in the algorithm in Section 2.6.

Some of the lemmas proved here rely on the algorithms by Miller et al. proving the following results.

Lemma 5 (Theorem 1 in [20]) *Given the adjacency matrix A of an undirected graph with n vertices distributed so that element $A_{i,j}$ is stored in $PE[i,j]$ of the HV-RM of size $n \times n$, the connected components of the graph can be determined in $O(\log n)$ steps.* ■

Lemma 6 (Lemma 1 in [20]) *Given a set $S = \{a_i\}$ of n values, distributed one per processor on the HV-RM of size $1 \times n$ so that $PE[0,j]$ contains a_j, $0 \leq j < n$, and a unit-time binary associative operation $\otimes$, in $O(\log n)$ steps the parallel prefix problem can be solved so that each processor $PE[0,j]$ knows $a_0 \otimes a_1 \otimes \cdots \otimes a_j$.* ■

Although these algorithms have been presented for the RM model, they can be executed on the HV-RM as well.[4]

Throughout this section, we assume that every initial label is distinct from any other labels[5] , which enables us to know whether two vertices are connected or not by just checking their (local) component labels. Furthermore, to avoid repetition, we fix some notations as follows. $G = (V, E)$ denotes $PCG\,(\mathcal{M}, \mathcal{S})$ where $\mathcal{M}$ is the RM of size $m \times n$ ($m \leq n$, n is even) and $\mathcal{S}$ is a step of $\mathcal{M}$. We divide V into two disjoint sets V^{L} and V^{R} such that

$$\begin{aligned} V^{\mathrm{L}} &= \{(i,j,X) \in V \mid 0 \leq j < \tfrac{n}{2}\}, \\ V^{\mathrm{R}} &= \{(i,j,X) \in V \mid \tfrac{n}{2} \leq j < n\}, \end{aligned}$$

[4] As for Lemma 5, refer to Section 5.1 in [3].

[5] This assumption does not hold in general. To make use of the results in this section, we shall extend the definition of initial labels in Section 2.6.

and let G_{L} (resp. G_{R}) denote a reduced graph of G obtained by restricting the vertex set into V^{L} (resp. V^{R}). See Figure 2.5 for an example. Also, we let $\mathcal{M}'$ denote the COMMON-COLLISION HV-RM of size $m \times n$, in which problems are solved here.

Let $V^{\mathrm{L}}_{\mathrm{con}}$ and $V^{\mathrm{R}}_{\mathrm{con}}$ be the sets:

$$\begin{aligned} V^{\mathrm{L}}_{\mathrm{con}} &= \{(i, \tfrac{n}{2}-1, \mathrm{E}) \in V^{\mathrm{L}} \mid 0 \le i < m\}, \\ V^{\mathrm{R}}_{\mathrm{con}} &= \{(i, \tfrac{n}{2}, \mathrm{W}) \in V^{\mathrm{R}} \mid 0 \le i < m\}, \end{aligned}$$

and let V^{*}_{con} denotes $V^{\mathrm{L}}_{\mathrm{con}} \cup V^{\mathrm{R}}_{\mathrm{con}}$. The vertices in V^{*}_{con} are called *contacting vertices* (in Figure 2.5, contacting vertices are marked with †). Note that the vertices in $V^{\mathrm{L}}_{\mathrm{con}}$ (resp. $V^{\mathrm{R}}_{\mathrm{con}}$) are the only vertices in V^{L} (resp. V^{R}) connected to vertices in V^{R} (resp. V^{L}) in G. Then, we define a graph called *bridge graph* as follows. Here, π_1 denotes a projection function such that $\pi_1((x, y, Z)) = x$.

Definition 6 (Bridge Graph of G_{L} and G_{R}) *The bridge graph of G_{L} and G_{R} is defined as an undirected graph $G_{\mathrm{bri}} = (V_{\mathrm{bri}}, E_{\mathrm{bri}})$ such that*

$$V_{\mathrm{bri}} = \{0, \ldots, m-1\} \ (= \{\pi_1(v) \mid v \in V^{*}_{\mathrm{con}}\}),$$

$$E_{\mathrm{bri}} = E^{\mathrm{L}}_{\mathrm{bri}} \cup E^{\mathrm{R}}_{\mathrm{bri}}$$

where

$$E^{\mathrm{L}}_{\mathrm{bri}} = \{\ \{\pi_1(v), \pi_1(v')\} \mid v, v' \in V^{\mathrm{L}}_{\mathrm{con}}, v \overset{G_{\mathrm{L}}}{\leftrightarrow} v'\ \},$$

$$E^{\mathrm{R}}_{\mathrm{bri}} = \{\ \{\pi_1(v), \pi_1(v')\} \mid v, v' \in V^{\mathrm{R}}_{\mathrm{con}}, v \overset{G_{\mathrm{R}}}{\leftrightarrow} v'\ \}.$$

■

For example, when G_{L} and G_{R} are the same graphs shown in Figure 2.5, we have V_{bri} ={ 0, 1, 2, 3 } and E_{bri} ={ {0,0}, {1,1}, {2,2}, {3,3}, {0,2}, {1,3}, {2,3} }.

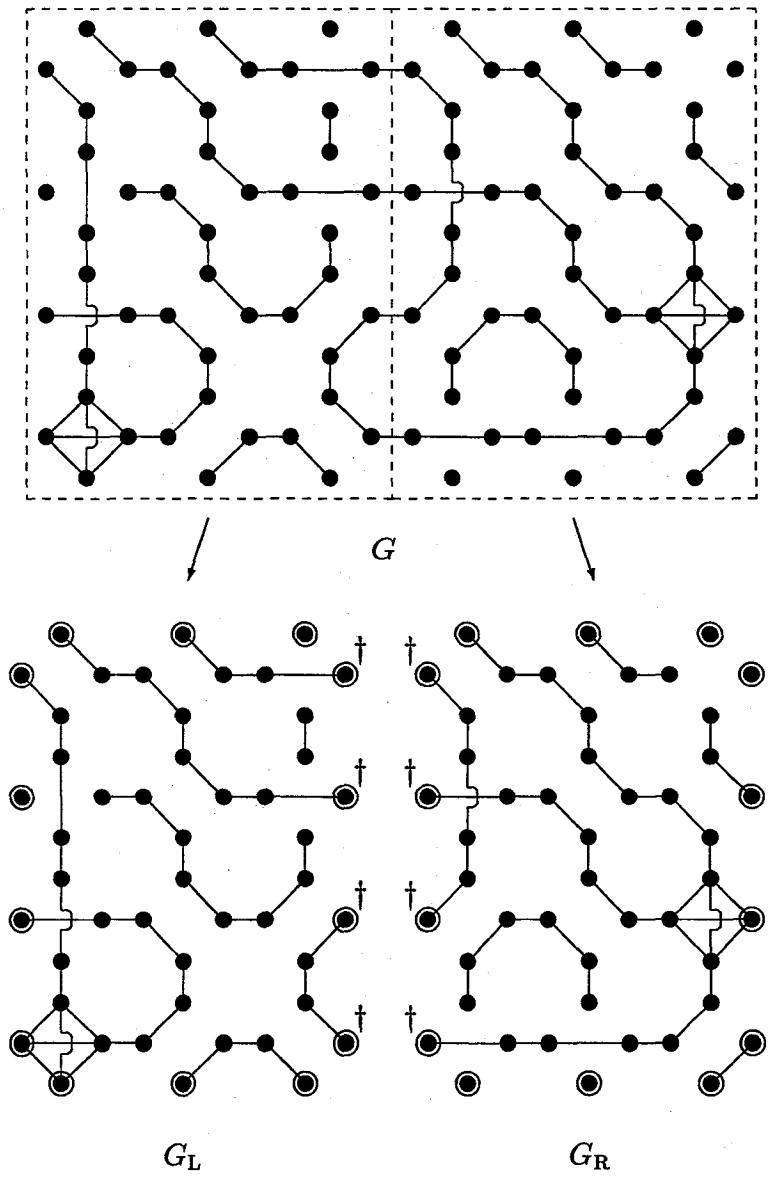


Figure 2.5: A pc-graph G and its subgraphs G_L and G_R.

The bridge graph enjoys the following property.

Property 1 (Relation between $\overset{G}{\leftrightarrow}$ and $\overset{G_{\mathrm{bri}}}{\leftrightarrow}$)

$$\forall v, v' \in V^*_{\mathrm{con}},\ [\ v \overset{G}{\leftrightarrow} v' \ \Leftrightarrow\ \pi_1(v) \overset{G_{\mathrm{bri}}}{\leftrightarrow} \pi_1(v')\ ]$$

Proof: Take any $v, v' \in V^*_{\mathrm{con}}$. Assume $v \overset{G}{\leftrightarrow} v'$ holds. Consider a path between v and v' in G, and enumerate every contacting vertex that appears in the path from v. Then, for every two consecutive v'' and v''' of these vertices, $\pi_1(v'') \overset{G_{\mathrm{bri}}}{\leftrightarrow} \pi_1(v''')$ must hold, which implies $\pi_1(v) \overset{G_{\mathrm{bri}}}{\leftrightarrow} \pi_1(v')$. On the other hand, if $\pi_1(v) \overset{G_{\mathrm{bri}}}{\leftrightarrow} \pi_1(v')$ holds, then, from the construction of G_{bri}, obviously $v \overset{G}{\leftrightarrow} v'$ follows. ■

Next, we consider how to compute the connected components of G_{bri}. Let c, c'_{L}, and c'_{R} be the functions which return the component labels, local component labels within G_{L}, and local component labels within G_{R}, respectively. And for each $v \in V$, let $c'(v)$ denote $c'_{\mathrm{L}}(v)$ if $v \in V^L$ otherwise $c'_{\mathrm{R}}(v)$. Then, the following lemma holds.

Lemma 7 *The following operation can be performed in $O(\log m)$ steps on $\mathcal{M}'$.*

Input: For each $(i,j,X) \in V^*_{\mathrm{con}}$, $c'((i,j,X))$ is given to $PE_{\mathcal{M}'}[i,j]$.

Output: The connected components of G_{bri} are obtained in such a way that all processors in row i and row j know the same component number $k \in \{0,\ldots,m-1\}$ iff $i \overset{G_{\mathrm{bri}}}{\leftrightarrow} j$ holds $(0 \le i,j < m)$.

Proof: This operation can be done in two phases. In the first phase, the adjacency matrix A of G_{bri} is generated so that each element $A_{i,j}$ is stored in $\mathrm{PE}_{\mathcal{M}'}[i,j]$. This is achieved by first broadcasting $c'_{\mathrm{L}}((i,\frac{n}{2}-1,\mathrm{E}))$ and $c'_{\mathrm{R}}((i,\frac{n}{2},\mathrm{W}))$ in each row i $(0 \le i < m)$, and next broadcasting $c'_{\mathrm{L}}((j,\frac{n}{2}-1,\mathrm{E}))$ and $c'_{\mathrm{R}}((j,\frac{n}{2},\mathrm{W}))$ from $\mathrm{PE}_{\mathcal{M}'}[j,j]$ in each column j $(0 \le j < m)$. Now, each $\mathrm{PE}_{\mathcal{M}'}[i,j]$ has sufficient information to determine $A_{i,j}$, and does $A_{i,j} \leftarrow 1$ if $c'_{\mathrm{L}}((i,\frac{n}{2}-1,\mathrm{E})) = c'_{\mathrm{L}}((j,\frac{n}{2}-1,\mathrm{E}))$ or $c'_{\mathrm{R}}((i,\frac{n}{2},\mathrm{W})) = c'_{\mathrm{R}}((j,\frac{n}{2},\mathrm{W}))$ holds, otherwise does $A_{i,j} \leftarrow 0$ $(0 \le i,j < m)$. Then, in the second phase, the connected components of G_{bri} is computed. The first phase needs a constant time, and the second phase takes $O(\log m)$ steps from Lemma 5. Hence, the conclusion follows. ■

Next, we consider how to compute the component labels of contacting vertices. These labels can be obtained from the information on their local component labels and the bridge graph, as shown below.

Property 2 (Component Label of Contacting Vertex)

$$\forall v \in V^*_{\mathrm{con}},\ c(v) = \downarrow\{c'(v') \mid \pi_1(v) \overset{G_{\mathrm{bri}}}{\leftrightarrow} \pi_1(v'), v' \in V^*_{\mathrm{con}}\}$$

Proof: From Property 1, the claim is equivalent to

$$\forall v \in V^*_{\mathrm{con}},\ c(v) = \downarrow\{c'(v') \mid v \overset{G}{\leftrightarrow} v', v' \in V^*_{\mathrm{con}}\}. \tag{2.1}$$

We prove (2.1) as follows. Take any $v \in V^*_{\mathrm{con}}$. Since $c(v') \le c'(v')$ holds for all $v' \in V$, we have

$$\downarrow\{c'(v') | v \overset{G}{\leftrightarrow} v', v' \in V^*_{\mathrm{con}}\} \ge \downarrow\{c(v') | v \overset{G}{\leftrightarrow} v', v' \in V^*_{\mathrm{con}}\} = c(v). \tag{2.2}$$

On the other hand, consider the vertex $v'' \in V$ whose initial label is $c(v)$, and a path $v = v_0, v_1, \ldots, v_l = v''$ in G. And let $e \in \{0, \ldots, l\}$ be the largest index satisfying $v_e \in V^*_{\mathrm{con}}$. Then, $c'(v_e) =$ (initial label of v'') $= c(v)$ holds, and thus

$$c(v) \geq \downarrow\{c'(v') \mid v \overset{G}{\leftrightarrow} v', v' \in V^*_{\mathrm{con}}\} \tag{2.3}$$

follows. The two inequalities (2.2) and (2.3) imply (2.1), and the conclusion is obtained. ■

Then, we can prove the following lemma.

Lemma 8 *The following operation can be performed in $O(\log m)$ steps on $\mathcal{M}'$.*

Input: For each $(i,j,X) \in V^*_{\mathrm{con}}$, $c'((i,j,X))$ is given to $PE_{\mathcal{M}'}[i,j]$. The connected components of G_{bri} is given in such a way that all processors in row i and row j know the same component number $k \in \{0, \ldots, m-1\}$ iff $i \overset{G_{\mathrm{bri}}}{\leftrightarrow} j$ holds $(0 \leq i, j < m)$.

Output: For each $(i,j,X) \in V^*_{\mathrm{con}}$, $c((i,j,X))$ is obtained at every processor in row i.

Proof: This operation can be performed in two phases. In the first phase, $\mathrm{PE}_{\mathcal{M}'}[i,j]$ does $\mathtt{t} \leftarrow c'_{\mathrm{L}}((j, \frac{n}{2}-1, \mathrm{E})) \downarrow c'_{\mathrm{R}}((j, \frac{n}{2}, \mathrm{W}))$ if $i \overset{G_{\mathrm{bri}}}{\leftrightarrow} j$ holds, otherwise does $\mathtt{t} \leftarrow \top$ $(0 \leq i, j < m)$. The necessary information for this task can be delivered to each processor by using row and column broadcasts similarly to the algorithm used for Lemma 7. Then, in the second phase, by computing the minimum (w.r.t. $\preceq$) among the

values stored in the variable t of the left m processors in each row, the output condition is satisfied (Property 2). The first phase needs a constant time, and the second phase takes $O(\log m)$ steps from Lemma 6. Thus, the conclusion follows. ■

Finally, we consider how to compute the component labels of *boundary vertices*, from the information on their local component labels. Boundary vertices are defined as follows. Let $V^{\mathrm{L}}_{\mathrm{bd}}$ and $V^{\mathrm{R}}_{\mathrm{bd}}$ be the sets:

$$\begin{aligned}
V^{\mathrm{L}}_{\mathrm{bd}} &= \{(0,j,\mathrm{N}) \in V^{\mathrm{L}} \mid 0 \le j < \tfrac{n}{2}\} \\
&\quad \cup \{(m-1,j,\mathrm{S}) \in V^{\mathrm{L}} \mid 0 \le j < \tfrac{n}{2}\} \\
&\quad \cup \{(i,\tfrac{n}{2}-1,\mathrm{E}) \in V^{\mathrm{L}} \mid 0 \le i < m\} \\
&\quad \cup \{(i,0,\mathrm{W}) \in V^{\mathrm{L}} \mid 0 \le i < m\}, \\
V^{\mathrm{R}}_{\mathrm{bd}} &= \{(0,j,\mathrm{N}) \in V^{\mathrm{R}} \mid \tfrac{n}{2} \le j < n\} \\
&\quad \cup \{(m-1,j,\mathrm{S}) \in V^{\mathrm{R}} \mid \tfrac{n}{2} \le j < n\} \\
&\quad \cup \{(i,n-1,\mathrm{E}) \in V^{\mathrm{R}} \mid 0 \le i < m\} \\
&\quad \cup \{(i,\tfrac{n}{2},\mathrm{W}) \in V^{\mathrm{R}} \mid 0 \le i < m\},
\end{aligned}$$

and let $V^{*}_{\mathrm{bd}} = V^{\mathrm{L}}_{\mathrm{bd}} \cup V^{\mathrm{R}}_{\mathrm{bd}}$. The vertices in V^{*}_{bd} are called boundary vertices of G_{L} and G_{R}, as they are located at the boundary of each graph (in Figure 2.5, those circled vertices in G_{L} and G_{R} denote the boundary vertices). W.r.t. the boundary and contacting vertices, the following property holds.

Property 3 (Component Label of Boundary Vertex)

1. *For each* $v \in V^{\mathrm{L}}_{\mathrm{bd}}$, *if there exists a vertex* $v' \in V^{\mathrm{L}}_{\mathrm{con}}$ *such that* $c'_{\mathrm{L}}(v') = c'_{\mathrm{L}}(v)$, *then* $c(v) = c(v')$ *holds for such* v', *otherwise* $c(v) = c'_{\mathrm{L}}(v)$.

2. *For each* $v \in V^{\mathrm{R}}_{\mathrm{bd}}$, *if there exists a vertex* $v' \in V^{\mathrm{R}}_{\mathrm{con}}$ *such that*

$c'_{\mathrm{R}}(v') = c'_{\mathrm{R}}(v)$, *then* $c(v) = c(v')$ *holds for such* v', *otherwise* $c(v) = c'_{\mathrm{R}}(v)$.

Proof: Obvious from the assumption on the initial labels and the construction of G. ■

Then, the following Lemma holds.

Lemma 9 *The following operation can be performed in a constant time on* $\mathcal{M}'$.

Input: For each $(i,j,X) \in V^*_{\mathrm{bd}}$, $c'((i,j,X))$ *is given to* $PE_{\mathcal{M}'}[i,j]$. *For each* $(i,j,X) \in V^*_{\mathrm{con}}$, $c((i,j,X))$ *is given to* $PE_{\mathcal{M}'}[i,j]$.

Output: For each $(i,j,X) \in V^*_{\mathrm{bd}}$, $c((i,j,X))$ *is obtained at* $PE_{\mathcal{M}'}[i,j]$.

Proof: Let us consider the vertices in $V^{\mathrm{L}}_{\mathrm{bd}}$ only, the details for the vertices in $V^{\mathrm{R}}_{\mathrm{bd}}$ are similar. We divide $V^{\mathrm{L}}_{\mathrm{bd}}$ into the following 4 disjoint sets:

$$\begin{aligned} V_{\mathrm{Nbd}} &= \{(0,j,\mathrm{N}) \in V^{\mathrm{L}} \mid 0 \le j < \tfrac{n}{2}\}, \\ V_{\mathrm{Sbd}} &= \{(m-1,j,\mathrm{S}) \in V^{\mathrm{L}} \mid 0 \le j < \tfrac{n}{2}\}, \\ V_{\mathrm{Ebd}} &= \{(i,\tfrac{n}{2}-1,\mathrm{E}) \in V^{\mathrm{L}} \mid 0 \le i < m\}, \\ V_{\mathrm{Wbd}} &= \{(i,0,\mathrm{W}) \in V^{\mathrm{L}} \mid 0 \le i < m\}. \end{aligned}$$

Then, the component labels can be computed in the following way.

- Vertices in V_{Nbd}: From Property 3, we has only to update the label of each $v \in V_{\mathrm{Nbd}}$ properly if there exists $v' \in V^{\mathrm{L}}_{\mathrm{con}}$ such that $c'(v') = c'(v)$, otherwise let $c(v) = c'(v)$. First, using row and column broadcasts, let $\mathrm{PE}_{\mathcal{M}'}[i,j]$ know $c'(i,\frac{n}{2}-1,\mathrm{E})$, $c(i,\frac{n}{2}-$

1, E), and $c'(0, j, \mathrm{N})$ $(0 \leq i < m,\ 0 \leq j < \frac{n}{2})$. Then, in each column j $(0 \leq j < \frac{n}{2})$, by letting every processor $\mathrm{PE}_{\mathcal{M}'}[i, j]$ with $c'(i, \frac{n}{2} - 1, \mathrm{E}) = c'(0, j, \mathrm{N})$ (if exists) broadcast $c(i, \frac{n}{2} - 1, \mathrm{E})$, $\mathrm{PE}_{\mathcal{M}'}[0, j]$ can know the component label of $(0, j, \mathrm{N}) \in V_{\mathrm{Nbd}}$. Note that $c'(i, \frac{n}{2} - 1, \mathrm{E}) = c'(i', \frac{n}{2} - 1, \mathrm{E})$ implies $c(i, \frac{n}{2} - 1, \mathrm{E}) = c(i', \frac{n}{2} - 1, \mathrm{E})$ $(0 \leq i, i' < m)$, and thus no collision occurs (i.e., no *error* is detected) on this broadcasting.

- Vertices in V_{Sbd}: Similarly to vertices in V_{Nbd}.
- Vertices in V_{Ebd}: Already obtained by the input condition.
- Vertices in V_{Wbd}: First move each $c(i, \frac{n}{2} - 1, \mathrm{E})$ and $c'(i, \frac{n}{2} - 1, \mathrm{E})$ to $\mathrm{PE}_{\mathcal{M}'}[0, i]$ by using row and column broadcasts $(0 \leq i < m)$, and follow the strategy used for V_{Nbd}.

These operations can be done in a constant time, and thus the conclusion follows. ■

Now, we can state the following.

Corollary 2 *The following operation can be performed in* $O(\log m)$ *steps on* $\mathcal{M}'$.

Input: For each $(i, j, X) \in V^*_{\mathrm{bd}}$, $c'((i, j, X))$ *is given to* $PE_{\mathcal{M}'}[i, j]$.

Output: For each $(i, j, X) \in V^*_{\mathrm{bd}}$, $c((i, j, X))$ *is obtained at* $PE_{\mathcal{M}'}[i, j]$.

Proof: From Lemma 7, 8, and 9. ■

2.6 Simulation of RM by HV-RM

In this section we show that the COMMON-COLLISION HV-RM of size $n \times n$ can simulate a step of the RM of the same size with slowdown $O(\log^2 n)$. Our algorithm is based on the component labeling algorithm for a binary image proposed by Maresca et al. in [1, 11]. We apply their divide-and-conquer algorithm to solve the LCC-PCG problem, and improve the memory usage of it.

The algorithm presented in this section makes use of the results in Section 2.5. However, in Section 2.5, every initial label is assumed to be distinct, which does not hold in general. Hence, we need to extend the definition of initial labels. The extension is done by adding the vertex information to the labels.

Definition 7 (Extension of Initial Label) *Let $G = (V, E)$ denote PCG $(\mathcal{M}, \mathcal{S})$. For each $(i, j, X) \in V$, we associate a label, denoted as $ini^{\mathrm{e}}_{\mathcal{M},\mathcal{S}}(i, j, X)$, in such a way that*

$$ini^{\mathrm{e}}_{\mathcal{M},\mathcal{S}}(i,j,X) = \begin{cases} (x,i,j,X) & \text{if } PE_{\mathcal{M}}[i,j] \text{ is a speaker sending a value } x \\ & \text{through port } X \text{ at the WRITE substep of } \mathcal{S}, \\ (\top,i,j,X) & \text{otherwise.} \end{cases}$$

These labels are called extended initial labels. ■

The total order $\preceq^{\mathrm{e}}$ is defined on the extended initial labels as a lexicographic order w.r.t. $\preceq$, $\leq$, and $\leq_{\mathrm{p}}$. $\leq_{\mathrm{p}}$ is the total order on {N,S,E,W} satisfying $\mathrm{N} \leq_{\mathrm{p}} \mathrm{S} \leq_{\mathrm{p}} \mathrm{E} \leq_{\mathrm{p}} \mathrm{W}$. The operator $\downarrow^{\mathrm{e}}$ is defined w.r.t. $\preceq^{\mathrm{e}}$ similarly as $\downarrow$ is to $\preceq$. In the following, we extend the LCC-PCG problem of $G = (V, E)$ in such a way that the component label of

$v \in V$ is defined as the value $\downarrow^{e}\{\text{extended initial label of } v' \mid v \overset{G}{\leftrightarrow} v'\}$. The definition of local component labels is modified similarly.

Let $G = (V, E)$ be $PCG\,(\mathcal{M}, \mathcal{S})$ where $\mathcal{M}$ is the RM of size $n \times n$ and $\mathcal{S}$ is a step of $\mathcal{M}$. Here, we show an algorithm that solves the LCC-PCG problem of G on $\mathcal{M}'$ where $\mathcal{M}'$ is the COMMON-COLLISION HV-RM of size $n \times n$. To simplify the exposition, we assume $n = 2^k$ for some positive integer k. To describe the algorithm, we use the following notations.

Definition 8 (Division of $\mathcal{M}'$) *For each $d \in \{0, \ldots, 2k\}$, a division of $\mathcal{M}'$ is defined as follows:*

d is even:

$\mathcal{M}'$ *is divided into* 2^{2k-d} *disjoint submeshes* $\mathcal{M}'^{d}_{p,q}$ *of size* $2^{d/2} \times 2^{d/2}$ $(0 \leq p, q < 2^{k-d/2})$. *Each* $\mathcal{M}'^{d}_{p,q}$ *consists of* $PE_{\mathcal{M}'}[i, j]$ $(p2^{d/2} \leq i < (p+1)2^{d/2}, q2^{d/2} \leq j < (q+1)2^{d/2})$.

d is odd:

$\mathcal{M}'$ *is divided into* 2^{2k-d} *disjoint submeshes* $\mathcal{M}'^{d}_{p,q}$ *of size* $2^{(d+1)/2} \times 2^{(d-1)/2}$ $(0 \leq p < 2^{k-(d+1)/2}, 0 \leq q < 2^{k-(d-1)/2})$. *Each* $\mathcal{M}'^{d}_{p,q}$ *consists of* $PE_{\mathcal{M}'}[i, j]$ $(p2^{(d+1)/2} \leq i < (p+1)2^{(d+1)/2}, q2^{(d-1)/2} \leq j < (q+1)2^{(d-1)/2})$.

■

See Figure 2.6 for an example. Note that each submesh $\mathcal{M}'^{d}_{p,q}$ consists of $\mathcal{M}'^{d-1}_{p,2q}$ and $\mathcal{M}'^{d-1}_{p,2q+1}$ if d is even, otherwise consists of $\mathcal{M}'^{d-1}_{2p,q}$ and $\mathcal{M}'^{d-1}_{2p+1,q}$ $(1 \leq d \leq 2k)$.

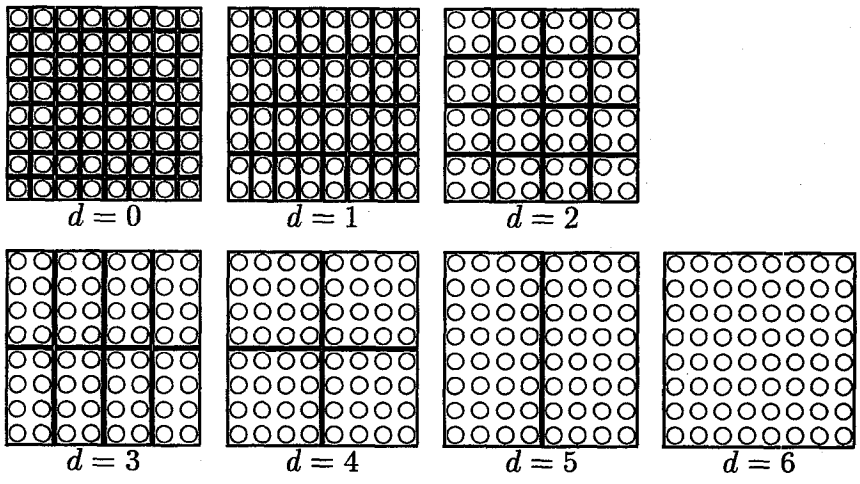


Figure 2.6: Divisions of an $n \times n$ HV-RM ($n = 2^3 = 8$).

Definition 9 (Subgraph of G) *For each submesh $\mathcal{M}'^{d}_{p,q}$, we associate a graph $G^d_{p,q} = (V^d_{p,q}, E^d_{p,q})$ such that $G^d_{p,q}$ is a reduced graph obtained from G by restricting the vertex set into $\{(i,j,X) \in V \mid PE_{\mathcal{M}'}[i,j]$ is in $\mathcal{M}'^{d}_{p,q}\}$. Let $G' = (V', E')$ be such a subgraph associated to a submesh $\mathcal{M}''$. Then, the boundary vertices of G' are defined as those vertices in V'_{Nbd}, V'_{Sbd}, V'_{Ebd}, and V'_{Wbd} where*

$V'_{\text{Nbd}} = \{(i,j,N) \in V' | PE_{\mathcal{M}'}[i,j]$ *is in the topmost row in* $\mathcal{M}''\}$,

$V'_{\text{Sbd}} = \{(i,j,S) \in V' | PE_{\mathcal{M}'}[i,j]$ *is in the bottom-most row in* $\mathcal{M}''\}$,

$V'_{\text{Ebd}} = \{(i,j,E) \in V' | PE_{\mathcal{M}'}[i,j]$ *is in the rightmost column in* $\mathcal{M}''\}$,

$V'_{\text{Wbd}} = \{(i,j,W) \in V' | PE_{\mathcal{M}'}[i,j]$ *is in the leftmost column in* $\mathcal{M}''\}$.

The vertices in each V'_{Nbd}, V'_{Sbd}, V'_{Ebd}, and V'_{Wbd} are respectively called the northern, southern, eastern, and western boundary vertices of G'.■

Then, in Figure 2.7, we show an algorithm that solves the LCC-PCG problem of G on $\mathcal{M}'$. The algorithm is carried out in the same

Algorithm RM_BY_HV-RM
{ *Labeling a pc-graph G on an $n \times n$* COMMON-COLLISION *HV-RM* $(\mathcal{M}')$. }

Stage 1: { *Label Combination* }
for d $\leftarrow$ 1 **to** $2k$ **do**
(d is even)
Each $\mathcal{M}'^{\mathrm{d}}_{p,q}$ computes the local component labels within $G^{\mathrm{d}}_{p,q}$ for the boundary vertices of $G^{\mathrm{d-1}}_{p,2q}$ and $G^{\mathrm{d-1}}_{p,2q+1}$.
(d is odd)
Each $\mathcal{M}'^{\mathrm{d}}_{p,q}$ computes the local component labels within $G^{\mathrm{d}}_{p,q}$ for the boundary vertices of $G^{\mathrm{d-1}}_{2p,q}$ and $G^{\mathrm{d-1}}_{2p+1,q}$.

Stage 2: { *Label Propagation* }
for d $\leftarrow$ $2k$ **to** 1 **do**
(d is even)
Each $\mathcal{M}'^{\mathrm{d}}_{p,q}$ computes the component labels of the boundary vertices of $G^{\mathrm{d-1}}_{p,2q}$ and $G^{\mathrm{d-1}}_{p,2q+1}$.
(d is odd)
Each $\mathcal{M}'^{\mathrm{d}}_{p,q}$ computes the component labels of the boundary vertices of $G^{\mathrm{d-1}}_{2p,q}$ and $G^{\mathrm{d-1}}_{2p+1,q}$.

end of RM_BY_HV-RM

Figure 2.7: Algorithm RM_BY_HV-RM.

fashion as the component labeling algorithm for a binary image [1, 11]. At each iteration of the for-loop of Stage 1, the newly obtained local component label of each vertex (i, j, X) is stored in $\mathrm{PE}_{\mathcal{M}'}[i, j]$.

The correctness of the algorithm is shown as follows.

Lemma 10 *RM_BY_HV-RM solves the LCC-PCG problem of G.*

Proof: Since

$$\bigcup_{p,q}(\text{the boundary vertices of } G^0_{p,q}) = V$$

holds, the component label of each vertex is obtained at the last iteration of the for-loop at Stage 2. ■

In the following, we show how to implement RM_BY_HV-RM on $\mathcal{M}'$.

Lemma 11 *Stage 1 of RM_BY_HV-RM can be performed in* $O(\log^2 n)$ *steps.*

Proof: Just before the l-th iteration of the for-loop at Stage 1, for each $G_{p,q}^{l-1}$, the local component label within $G_{p,q}^{l-1}$ is obtained for every boundary vertex of $G_{p,q}^{l-1}$. This is from the input condition (when $l = 1$), or from the result of the preceding iteration of the for-loop (when $l > 1$). Then, by Corollary 2, the execution of the for-loop body with $\mathrm{d} = l$ can be performed in

$$O(\log 2^{\lfloor l/2 \rfloor}) = O(l)$$

steps. Since

$$\sum_{l=1}^{2k} l = O(\log^2 n)$$

holds, the conclusion follows. ■

Lemma 12 *Stage 2 of RM_BY_HV-RM can be performed in* $O(\log n)$ *steps.*

Proof: First, we show that every iteration of the for-loop at Stage 2 can be done in a constant time. Let us consider only the case when d is even, the case when d is odd can be proved similarly.

Consider the execution of the for-loop body with $d = l$. During the execution, each $\mathcal{M}^{ll}_{p,q}$ computes the component labels of the boundary vertices of $G^{l-1}_{p,2q}$ and $G^{l-1}_{p,2q+1}$. Here, we consider the component labels of the boundary vertices of $G^{l-1}_{p,2q}$ only, those of $G^{l-1}_{p,2q+1}$ can be obtained similarly. Just before the execution of the for-loop body, the following conditions hold:

1. The component label is obtained for every boundary vertex of $G^{l}_{p,q}$.

2. The local component label within $G^{l}_{p,q}$ is obtained for every boundary vertex of $G^{l-1}_{p,2q}$ and $G^{l}_{p,q}$.

The condition 1 is from the last iteration of the for-loop at Stage 1 (when $l = 2k$), or from the result of the preceding iteration of the for-loop at Stage 2 (when $l < 2k$). The condition 2 is from Stage 1. Let V_{bd} be the set of all boundary vertices of $G^{l}_{p,q}$. Note that the component labels for the northern, southern, and western boundary vertices of $G^{l-1}_{p,2q}$ are already obtained because these vertices are also the boundary vertices of $G^{l}_{p,q}$. Thus, we have only to compute the component labels for the eastern boundary vertices of $G^{l-1}_{p,2q}$. Let V_{Ebd} be the set of the eastern boundary vertices of $G^{l-1}_{p,2q}$, and for each $v \in V_{\mathrm{bd}} \cup V_{\mathrm{Ebd}}$, let

$c'(v)$ and $c(v)$ denote the the local component label within $G^l_{p,q}$ and the component label, respectively. Then, the following property holds:

> For each $v \in V_{\mathrm{Ebd}}$, if there exists a vertex $v' \in V_{\mathrm{bd}}$ such that $c'(v') = c'(v)$, then $c(v) = c(v')$ holds for such v', otherwise $c(v) = c'(v)$.

The proof is similar to that of Property 3, and the details are omitted here. Because of this property, we has only to update $v \in V_{\mathrm{Ebd}}$ properly if there exists $v' \in V_{\mathrm{bd}}$ such that $c'(v') = c'(v)$, otherwise let $c(v) = c'(v)$. By the similar technique used for proving Lemma 9, this task can be done in a constant time (see [18] for the details). Now it is shown that every iteration of the for-loop can be performed in a constant time. Since

$$\sum_{l=1}^{2k} 1 = O(\log n)$$

holds, the conclusion follows. ■

Next, we consider the memory usage of RM_BY_HV-RM. To perform each one iteration of the for-loops at Stage 1 and Stage 2, just a constant number of temporary storage in each processor is sufficient. However, when the execution of the for-loop body at Stage 1 is completed with d = l, every processor of $\mathcal{M}'$ has to store the newly obtained local component label if it holds the boundary vertex of some $G^{l-1}_{p,q}$. Let us call these processors as *storage processors at* $d = l$ (Figure 2.8). Then, the memory usage of RM_BY_HV-RM in a processor is

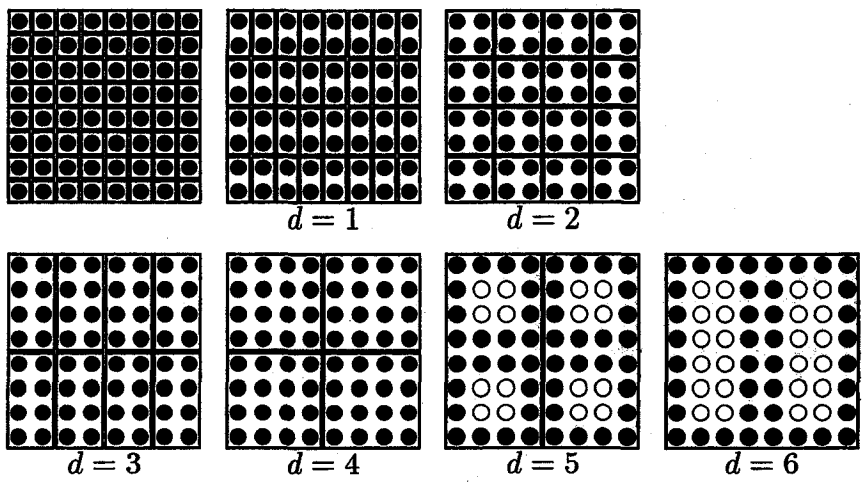


Figure 2.8: Storage processors of an $n \times n$ HV-RM for each d ($n = 2^3 = 8$).

proportional to how many times it becomes a storage processor during Stage 1. But as expected, there exists a processor which becomes a storage processor many times. For example, $\mathrm{PE}_{\mathcal{M}'}[0,0]$ becomes a storage processor every time, and it needs $\Omega(\log n)$ memory to store the local component labels. Because of this reason, the component labeling algorithm for a binary image in [1] uses $\Omega(\log n)$ storage in each processor. But as shown below, we can execute RM_BY_HV-RM with $O(1)$ memory in each processor.

Lemma 13 *RM-BY-HV-RM can be performed with using only a constant number of storage in each processor.*

Proof: We change the processors which maintain the newly obtained local component labels at each iteration of the for-loop of Stage 1, as

follows:

d is even:

The newly obtained local component labels for the boundary vertices of $G_{p,q}^{d-1}$ are stored in the processors located in the rightmost column of $\mathcal{M}_{p,q}'^{d-1}$ if q is even, otherwise in the processors in the leftmost column of $\mathcal{M}_{p,q}'^{d-1}$. The label of a western or eastern boundary vertex of $G_{p,q}^{d-1}$ obtained at a processor in row i of $\mathcal{M}_{p,q}'^{d-1}$ is stored in the processor in the same row of $\mathcal{M}_{p,q}'^{d-1}$. The label of a northern (resp. southern) boundary vertex of $G_{p,q}^{d-1}$ obtained at a processor in the topmost row (resp. bottom-most row) and column j of $\mathcal{M}_{p,q}'^{d-1}$ is stored in the processor in row j (resp. row $2^{d/2}-1-j$) of $\mathcal{M}_{p,q}'^{d-1}$.

d is odd:

The newly obtained local component labels for the boundary vertices of $G_{p,q}^{d-1}$ are stored in the processors located in the bottommost row of $\mathcal{M}_{p,q}'^{d-1}$ if p is even, otherwise in the processors in the topmost row of $\mathcal{M}_{p,q}'^{d-1}$. The label of a western or eastern boundary vertex of $G_{p,q}^{d-1}$ obtained at a processor in row i of $\mathcal{M}_{p,q}'^{d-1}$ is stored in the processor in column i of $\mathcal{M}_{p,q}'^{d-1}$. The label of a northern or southern boundary vertex of $G_{p,q}^{d-1}$ obtained at a processor in column j of $\mathcal{M}_{p,q}'^{d-1}$ is stored in the processor in the same column of $\mathcal{M}_{p,q}'^{d-1}$.

We call those processors as *folded storage processors at* $d = l$ (Figure 2.9). Then, every processor is ensured to become a folded storage processor at most 4 times during Stage 1, and each time it has only to store at most 4 labels in it. At each iteration of the for-loop of Stage

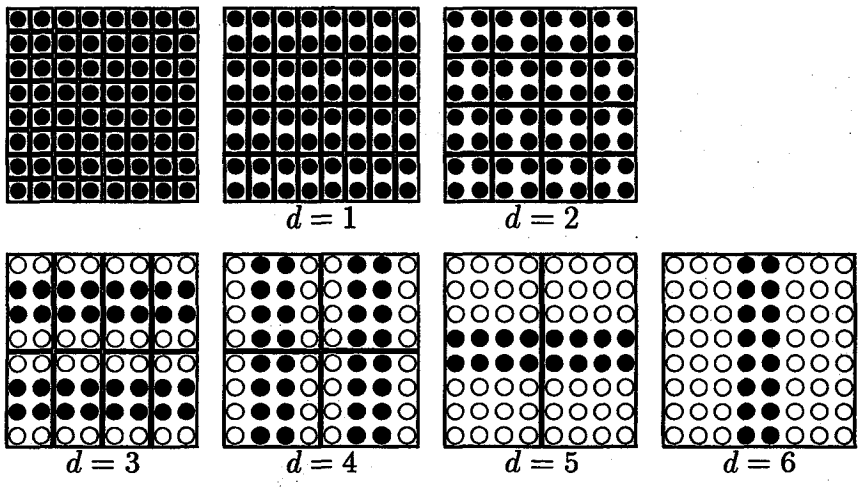


Figure 2.9: Folded storage processors of an $n \times n$ HV-RM for each d ($n = 2^3 = 8$).

1, the operation of moving the newly obtained local component labels to the proper folded storage processors can be performed in a constant time by using row and column broadcasts in each submesh. Similarly, the local component labels stored in the folded storage processors at d $= l$ can be moved back in a constant time to the processors in such a way that each local component label of (i, j, X) is moved to $\mathrm{PE}_{\mathcal{M}'}[i, j]$. Hence, this change of the label keeping strategy does not invalidate the proofs of Lemma 11 and 12. Thus, the conclusion follows. ■

Now, we can state the following theorem.

Theorem 2 *The* COMMON-COLLISION *HV-RM of size $n \times n$ can simulate a step of the RM of size $n \times n$ with slowdown $O(\log^2 n)$, using a constant number of storage in each processor.*

Proof: The first element of the component label of each $v \in V$ is equal to the component label of v defined in Section 2.2. Therefore, by Lemma 1, 10, 11, 12, 13, and Corollary 1, the conclusion follows. ■

2.7 Scaling Simulation

In this section, we discuss applications of our simulation algorithms. First, we present the following theorem.

Theorem 3 *The MWMB of size $n \times n$ can simulate the RM of size $n \times n$ with slowdown $O(\sqrt{n}\log^2 n)$.*

Proof: From Theorem 1 and 2. ■

Our algorithms can simulate an RM with the MIN bus model, and this may complement the slowdown in some cases. For example, some RM algorithms involve the operation of electing a leader processor among the processors interconnected by a bus, or of computing the minimum among the values held by such interconnected processors. And in some cases, the time cost for these operations becomes the crucial cost of the entire algorithm. But in the MIN bus model, such operations can be performed in a constant time. Therefore, we expect that the time cost of simulating such an RM algorithm on a COMMON-COLLISION HV-RM or on an MWMB can be reduced more in practice, by improving the time cost originally taken by the RM algorithm by rewriting it on the MIN RM model.

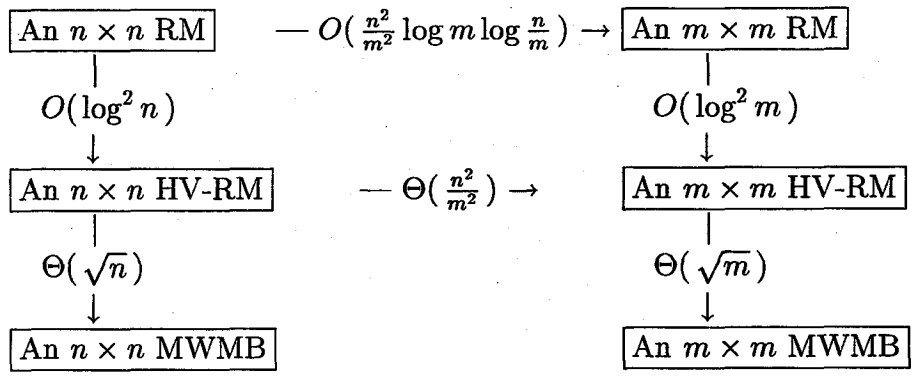


Figure 2.10: Costs of simulating a single step ($m < n$).

Next, we consider simulating an RM by a smaller enhanced mesh. In [3], Ben-Asher et al. proposed self-simulation algorithms on the RM, LRM, and HV-RM models (these models are of COMMON-COLLISION). In [9], Fernández-Zepeda et al. proposed more efficient RM self-simulation algorithm than that in [3]. We illustrate our results and their self-simulation results in Figure 2.10, concerning only the RM, HV-RM, and MWMB models. The horizontal arrows are by the self-simulation results in [3, 9].

Then, we have the following lemma.

Lemma 14 *The* COMMON-COLLISION *HV-RM of size $m \times m$ can simulate the RM of size $n \times n$ with slowdown $O((\frac{n}{m})^2 \log^2 n)$ $(m < n)$.*

Proof: From Theorem 2 and the self-simulation results on the HV-RM model in [3]. ■

In the HV-RM self-simulation algorithm in [3], each $\mathrm{PE}_{\mathcal{M}'}[i,j]$ simulates $\mathrm{PE}_{\mathcal{M}}[x,y]$ ($i(\frac{n}{m}) \leq x < (i+1)(\frac{n}{m})$, $j(\frac{n}{m}) \leq y < (j+1)(\frac{n}{m})$) where $\mathcal{M}$ is the simulated mesh of size $n \times n$ and $\mathcal{M}'$ is the simulating mesh of size $m \times m$ ($m < n$). Hence, the same processor mapping is taken in the simulation of Lemma 14. In the following, we show that we can improve the time cost given in Lemma 14.

Theorem 4 *The* COMMON-COLLISION *HV-RM of size $m \times m$ can simulate the RM of size $n \times n$ with slowdown $O(\,(\frac{n}{m})^2 \log n \log m\,)$ ($m < n$).*

Proof: Let $\mathcal{M}$ denote the RM of size $n \times n$. For simplify the exposition, we assume $n = 2^k$ and $m = 2^{k'}$ for some positive integers k and k' ($k' < k$). Let $2^{d'} = (\frac{n}{m})^2$. In Figure 2.11, we show the modified version of RM_BY_HV-RM.

Here, $\mathcal{M}'$ denotes the COMMON-COLLISION HV-RM of size $n \times n$, and $G = (V, E)$ is *PCG* $(\mathcal{M}, \mathcal{S})$ for any $\mathcal{S}$. RM_BY_HV-RM-2 solves the LCC-PCG problem of G on $\mathcal{M}'$. The correctness of RM_BY_HV-RM-2 is straightforward, and the details are omitted.

Next, we consider how to simulate RM_BY_HV-RM-2 on $\mathcal{M}''$ where $\mathcal{M}''$ is the COMMON-COLLISION HV-RM of size $m \times m$. The processor mapping from $\mathcal{M}'$ to $\mathcal{M}''$ is the same as the simulation of Lemma 14. Then, since each $\mathcal{M}'^{d'}_{i,j}$ is simulated by $\mathrm{PE}_{\mathcal{M}''}[i,j]$ alone, Stage 1.1 and Stage 2.2 can be performed in each processor by using a well-known sequential algorithm. Each $G^{d'}_{i,j}$ contains $4(\frac{n}{m})^2$ vertices and $\Theta(\,(\frac{n}{m})^2\,)$ edges, and hence this task can be completed in $O(\,(\frac{n}{m})^2\,)$ steps. As for

Stage 1.2 and Stage 2.1, they are simulated step-by-step by the HV-RM self-simulation algorithm in [3]. Since

$$
\begin{aligned}
& \sum_{l=d'+1}^{2k} \log 2^{\lfloor l/2 \rfloor} \\
\leq & \\
& \sum_{l=d'+1}^{2k} l \\
= & \\
& \frac{(2k-d')(2k+d'+1)}{2} \\
= & \quad \langle\, d' = 2k - 2k' \,\rangle \\
& \frac{2k'(4k-2k'+1)}{2} \\
\leq & \\
& 4kk' \\
= & \\
& O(\log n \log m)
\end{aligned}
$$

holds, Stage 1.2 can be performed in $O((\frac{n}{m})^2 \log n \log m)$ steps on $\mathcal{M}''$. As for Stage 2.1, since

$$
\begin{aligned}
& \sum_{l=d'+1}^{2k} 1 \\
= & \\
& 2k - d' \\
= & \quad \langle\, d' = 2k - 2k' \,\rangle \\
& 2k' \\
= & \\
& 2 \log m
\end{aligned}
$$

holds, it can be performed in $O((\frac{n}{m})^2 \log m)$ steps on $\mathcal{M}''$. As a whole, RM_BY_HV-RM-2 can be performed in $O((\frac{n}{m})^2 \log n \log m)$ steps on $\mathcal{M}''$. Then, by similar arguments of the proofs of Lemma 1 and Corollary 1, the conclusion follows. ■

Corollary 3 *The MWMB of size $m \times m$ can simulate the RM of size $n \times n$ with slowdown $O((\frac{n}{m})^2 \sqrt{m} \log n \log m)$ $(m < n)$.*

Proof: From Theorem 1 and 4. ■

Algorithm RM_BY_HV-RM-2
{ *Labeling a pc-graph G on an $n \times n$* COMMON-COLLISION *HV-RM* ($\mathcal{M}'$). }

Stage 1: { *Label Combination* }

Stage 1.1:
Each $\mathcal{M}'^{d'}_{p,q}$ computes the local component label within $G^{d'}_{p,q}$ for every vertex in $V^{d'}_{p,q}$.

Stage 1.2:
for d $\leftarrow$ $(d'+1)$ **to** $2k$ **do**
Execute the d-th iteration of the for-loop at Stage 1 of RM_BY_HV-RM.

Stage 2: { *Label Propagation* }

Stage 2.1:
for d $\leftarrow$ $2k$ **to** $d'+1$ **do**
Execute the $(2k - \mathrm{d} + 1)$-th iteration of the for-loop at Stage 2 of RM_BY_HV-RM.

Stage 2.2:
Each $\mathcal{M}'^{d'}_{p,q}$ computes the local component label within $G^{d'}_{p,q}$ for every vertex in $V^{d'}_{p,q}$, with treating the component labels of the boundary vertices of $G^{d'}_{p,q}$ as their initial labels.

end of RM_BY_HV-RM-2

Figure 2.11: Algorithm RM_BY_HV-RM-2.

Chapter 3

Simulation of Mesh with Separable Buses

In this chapter, we consider simulation problems among the *mesh with separable buses* (MSB), *mesh with restricted separable buses* (MRSB), and *mesh with partitioned buses* (MPB). The MSB is equal in computational power to the HV-RM defined in the Chapter 2, and is more powerful than the MRSB and MPB.

Here, we show that the MSB of size $n \times n$ can be simulated in $\Theta(n^{1/3})$ steps by the MPB of size $n \times n$ when $\ell = \Theta(n^{2/3})$, and in $\Theta(\ell)$ steps by the MRSB of size $n \times n$. These time costs are shown to be optimal in the worst case under the assumption on the processor mapping we take here.

We also consider the influences of propagation delays of broadcasting buses. We assume that the propagation delay of a broadcasting bus is introduced by the length of the bus (i.e., signal propagation delay) and the number of switch elements inserted to the bus (i.e., device propagation delay). Compared to the MSB, the propagation delay of buses in the MPB and MRSB can be small in practice, and thus we

consider that our simulation algorithms are useful when the mesh size becomes so large that we cannot neglect the delay.

This chapter is organized as follows. Section 3.1 describes the MSB, MPB, and MRSB models. Section 3.2 presents an algorithm that simulates the MSB on the MPB, and Section 3.3 gives an algorithm that simulates the MSB on the MRSB. And Section 3.4 discusses the influences of propagation delays.

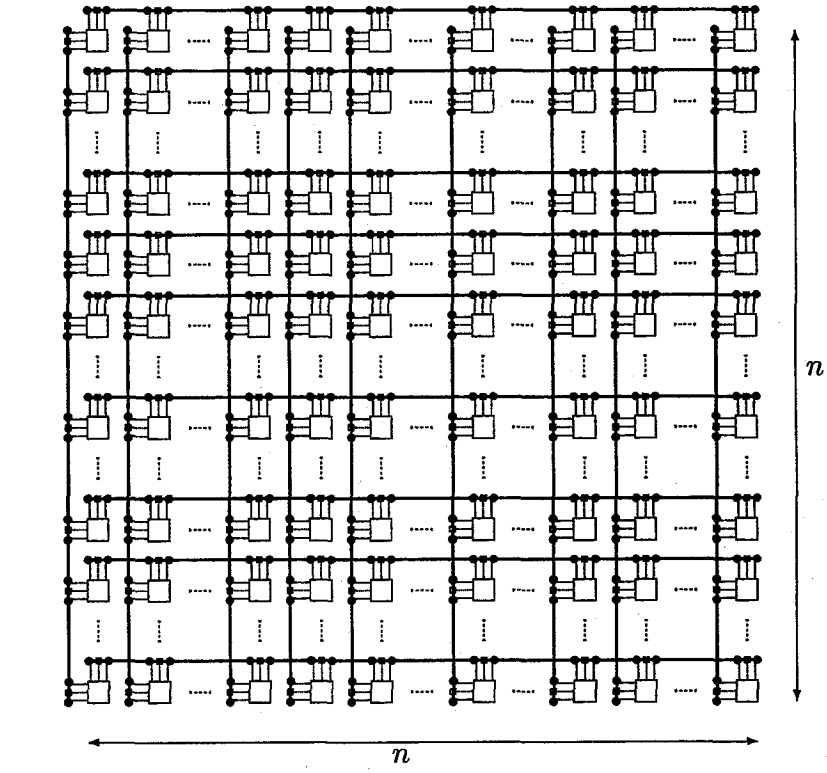
3.1 Models

An $n \times n$ mesh consists of n^2 identical SIMD processors or processing elements (PE's) arranged in a two-dimensional grid with n rows and n columns. The PE located at the grid point (i, j), denoted as PE$[i, j]$, is connected via bi-directional unit-time communication links to those PE's at $(i \pm 1, j)$ and $(i, j \pm 1)$, provided they exist $(0 \leq i, j < n)$. PE$[0, 0]$ is located in the top-left corner of the mesh. Each PE$[i, j]$ is assumed to know its coordinates (i, j).

An $n \times n$ *mesh with separable buses* (MSB) and an $n \times n$ *mesh with partitioned buses* (MPB) are the $n \times n$ meshes enhanced with broadcasting buses along each row and column (Figure 3.1[1] and 3.2). The broadcasting buses of the MSB, called *separable buses*, can be dynamically sectioned through the PE-controlled switches during the execution of programs, while those of the MPB are statically partitioned in advance by a fixed length ℓ. An $n \times n$ *mesh with restricted separable buses* (MRSB) is the $n \times n$ mesh enhanced with the separable broadcasting

[1] Here, the MSB is slightly different from the one proposed by Maeba et al [15]. Our model is closer (in fact, is equal in the computational power) to the HV-RM model defined in Chapter 2.

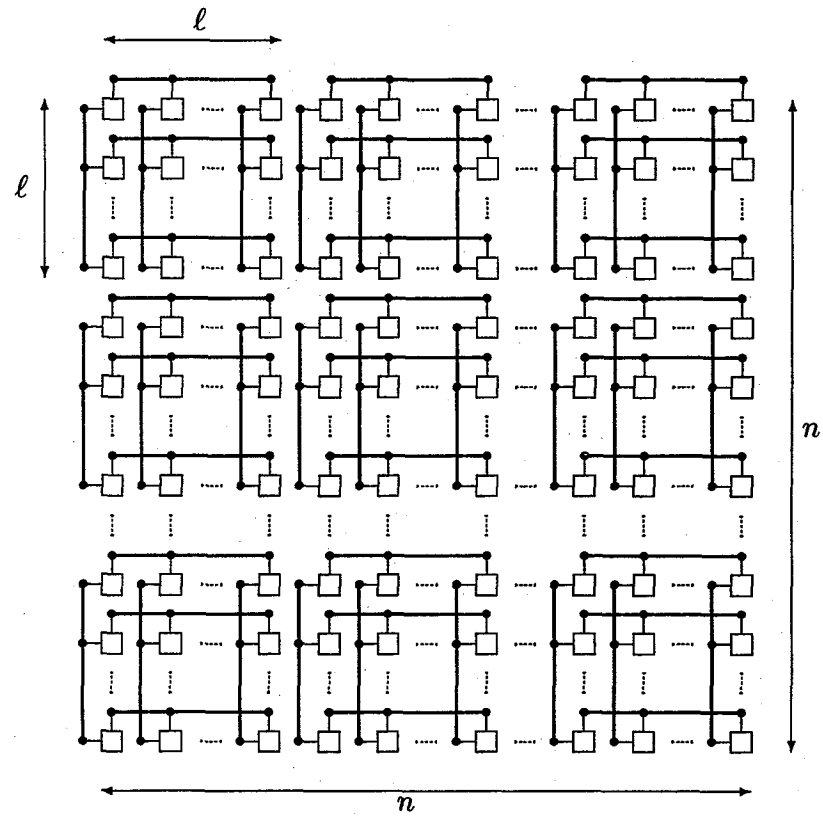
buses along every ℓ rows and ℓ columns for a fixed ℓ (Figure 3.3). In the MRSB, only PE$[i\ell, j\ell]$ can can gain access to the broadcasting buses $(0 \leq i, j < n/\ell)$.



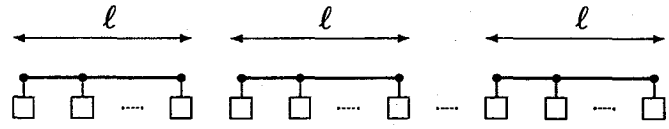
(a) An $n \times n$ MSB

(b) The broadcasting bus along a row of an $n \times n$ MSB

Figure 3.1: A mesh with separable buses (MSB). Local links are not shown.

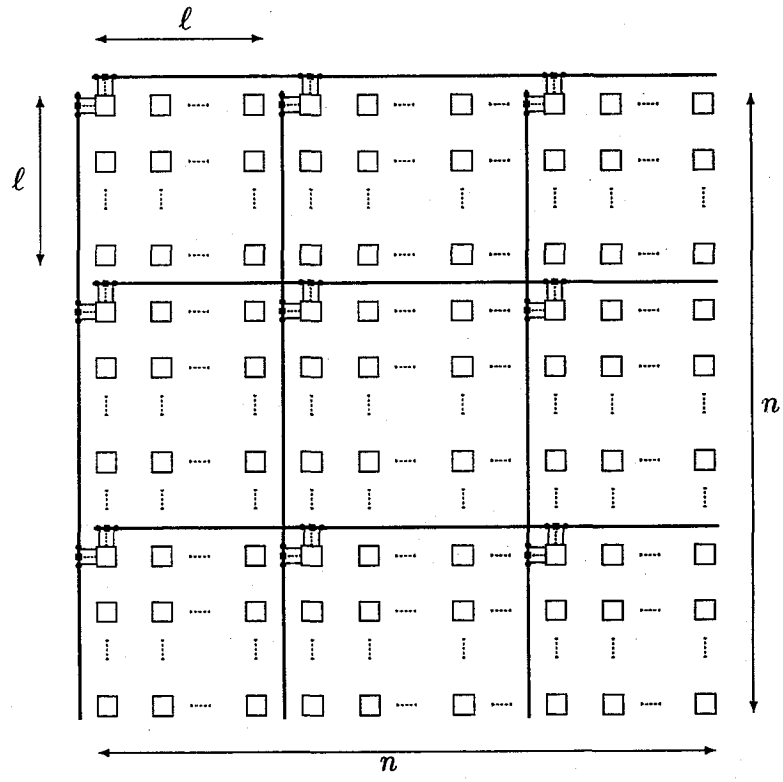


(a) An $n \times n$ MPB

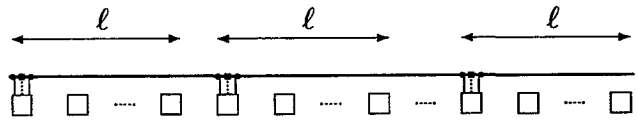


(b) The broadcasting buses along a row of an $n \times n$ MPB

Figure 3.2: A mesh with partitioned buses (MPB). Local links are not shown.



(a) An $n \times n$ MRSB



(b)The broadcasting bus in row $p\ell$ of an $n \times n$ MRSB

Figure 3.3: A mesh with restricted separable buses (MRSB). Local links are not shown.

A single time step of the MSB, MPB, and MRSB is composed of the following three substeps:

Local Comm. Substep : Every PE communicates with its adjacent PE's via local links.

Broadcast Substep : Every PE changes its switch configurations by local decision (this operation is only for the MSB and the MRSB). After that, along each broadcasting bus segment, several of the PE's connected to the bus send data to the bus, and several of the PE's on the bus receive the data transmitted on the bus.

Compute Substep : Every PE executes some local computation.

We assume that the propagation delay of the broadcasting buses is a constant time, and that each of the three substeps is executed in a constant time.

The bus accessing capability is of the COMMON-COLLISION CRCW model. If there is a write-conflict on a bus, then the PE's on the bus receive a special value $\perp$ (i.e., PE's can detect whether there is a write-conflict on a bus or not). If there is no data transmitted on a bus, then the PE's on the bus receive a special value ϕ (i.e., PE's can know whether there is data transmitted on a bus or not).

3.2 Simulation of MSB by MPB

In this section, we consider simulating a single step of the $n \times n$ MSB (denoted as $\mathcal{M}$) by the $n \times n$ MPB (denoted as $\mathcal{M}'$). To avoid confusion, let $\mathrm{PE}_{\mathcal{M}}[i,j]$ and $\mathrm{PE}_{\mathcal{M}'}[i,j]$ respectively denote $\mathrm{PE}[i,j]$ of $\mathcal{M}$ and $\mathrm{PE}[i,j]$ of $\mathcal{M}'$.

Given a single step of $\mathcal{M}$ in such a way that each $\mathrm{PE}_{\mathcal{M}'}[i,j]$ knows how $\mathrm{PE}_{\mathcal{M}}[i,j]$ behaves at this single step, we consider how to achieve the same computational task of the step using $\mathcal{M}'$. We assume that the computing power of PE's, the bandwidth of local links, and that of broadcasting buses are equivalent in both $\mathcal{M}$ and $\mathcal{M}'$.

To begin with, we prove the following lemma.

Lemma 15 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable bus in row i (resp. column i) of $\mathcal{M}$ can be simulated in row i (resp. column i) of $\mathcal{M}'$ in $O(\ell + n/\ell)$ steps $(0 \leq i < n)$.*

Proof: Take any single step $\mathcal{S}$ of $\mathcal{M}$ and $i \in \{0, 1, \ldots, n-1\}$. Let us consider simulating the broadcasts taken on the separable bus along row i of $\mathcal{M}$ only, those on the bus along column i of $\mathcal{M}$ can be simulated similarly.

First, we define some notations to describe the broadcasts to be simulated. Let P_j denote $\mathrm{PE}_{\mathcal{M}}[i,j]$ $(0 \leq j < n)$. To distinguish the two ports through which a PE has access to the row separable bus, we refer to the port on the left side of the sectioning switch as *port L* and the other as *port R*, as shown in Figure 3.1(b). Then, the broadcasts is carried out in the following way: (1) several of $\mathrm{P}_0, \mathrm{P}_1, \ldots, \mathrm{P}_{n-1}$ section the bus, (2) several of $\mathrm{P}_0, \mathrm{P}_1, \ldots, \mathrm{P}_{n-1}$ send data to the bus through port L and/or R, and (3) several of $\mathrm{P}_0, \mathrm{P}_1, \ldots, \mathrm{P}_{n-1}$ receive data from the bus through port L and/or R. W.r.t. these broadcasts performed in row i of $\mathcal{M}$, we define C_j^x, s_j^x, and r_j^x $(0 \leq j < n,\ x \in \{\mathrm{L}, \mathrm{R}\})$ as follows:

$$C_j^x = \{(k,y) \mid \text{port } x \text{ of } \mathrm{P}_j \text{ and port } y \text{ of } \mathrm{P}_k \text{ belong to the same bus segment after the broadcasting bus being sectioned}\},$$

$s_j^x = a$ if P_j sends data a to port x, otherwise $s_j^x = \phi$,

$r_j^x =$ (the data received by P_j from port x).

To describe each r_j^x using C_*^* and s_*^*, we define a binary commutative operator $\oplus$ in such a way that it satisfies the following equations for any x and y:

$$x \oplus \phi = \phi \oplus x = x,$$

$$x \oplus \perp = \perp \oplus x = \perp,$$

$$x \oplus y = y \oplus x = x \quad \text{if } x = y,$$

$$x \oplus y = y \oplus x = \perp \quad \text{if } x \neq y,\ x \neq \phi, \text{ and } y \neq \phi.$$

It is not difficult to confirm that $\oplus$ is well-defined and enjoys the associative law. Then, each r_j^x can be expressed as

$$r_j^x = \oplus_{(k,y) \in C_j^x} s_k^y. \tag{3.1}$$

Next, let P'_j denote $\mathrm{PE}_{\mathcal{M}'}[i,j]$ $(0 \leq j < n)$, and consider how to inform every P'_j of r_j^{L} and r_j^{R} when every P'_k is given s_k^{L}, s_k^{R}, and the switch configuration taken by P_k. We divide $\mathrm{P}'_0, \mathrm{P}'_1, \ldots, \mathrm{P}'_{n-1}$ into n/ℓ disjoint *blocks* $\mathcal{M}'_p$ $(0 \leq p < n/\ell)$. Each $\mathcal{M}'_p$ consists of P'_j $(p\ell \leq j < (p+1)\ell)$. Here, let $\mathrm{LP}'_{\mathcal{M}'p}$ (resp. $\mathrm{RP}'_{\mathcal{M}'p}$) denote the leftmost PE (resp. the rightmost PE) of $\mathcal{M}'_p$. It should be noted that $\mathrm{RP}'_{\mathcal{M}'p}$ and $\mathrm{LP}'_{\mathcal{M}'p+1}$ are adjacent PE's $(0 \leq p < n/\ell - 1)$ and that any PE in $\mathcal{M}'_p$ can communicate with the other PE's in $\mathcal{M}'_p$ in a single time step using the broadcasting bus $(0 \leq p < n/\ell)$. For each $j \in \{0, \ldots, n-1\}$ and $x \in \{\mathrm{L}, \mathrm{R}\}$, we let

$$r'^x_j = \oplus_{(k,y) \in C'^x_j} s_k^y \tag{3.2}$$

where $C'^{x}_{j} = C^{x}_{j} \cap \{(k,y) \mid \mathrm{P}_j$ and P_k are in the same block and $y \in \{\mathrm{L},\mathrm{R}\}\}$.

Then, in Figure 3.4, we show an algorithm that simulates the broadcasts taken along row i of $\mathcal{M}$. Each r^{x}_{j} is stored in variable $\mathtt{D}_x$ of P'_j when the algorithm terminates.

As for each r^{x}_{j} such that $C^{x}_{j} \subset \{(k,y) \mid \mathrm{P}'_k$ is in $\mathcal{M}'_p$ and $y \in \{\mathrm{L},\mathrm{R}\}\}$ for some $p \in \{0,\ldots,n/\ell-1\}$, it is obtained at $\mathtt{D}_x$ of P'_j at Phase 1 because $r^{x}_{j} = r'^{x}_{j}$ holds in such a case. As for the rest, they are computed at Phase 2. After the execution of the first for-loop, such r^{x}_{j} is obtained at $\mathtt{D}_y$ of P'_k for every $(k,y) \in C^{x}_{j}$ such that both $p\ell \leq k$ and $((p+1)\ell,\mathrm{L}) \notin C^{x}_{j}$ hold for some $p \in \{0,\ldots,n/\ell-1\}$, and at the second for-loop, the value is copied to $\mathtt{D}_y$ of P'_k for every $(k,y) \in C^{x}_{j}$.

Phase 1 can be performed in $O(\ell)$ steps, since each block consists of ℓ consecutive PE's. Note that this phase can be done similarly to the well-known algorithm on a linear processor array that performs a semigroup computation on values distributed one per processor by sequentially scanning those values. Phase 2 needs $O(n/\ell)$ steps. Thus, the conclusion follows. ■

Next, we consider improving the time cost shown in Lemma 15. For Lemma 15, we presented the algorithm SB-by-PB in which the first phase is performed in $O(\ell)$ steps by sequentially scanning data within each block. In the following, we reduce the time cost for this phase to $O(\ell^{1/2})$ steps, and as a result we obtain more efficient algorithm which runs in $O(\ell^{1/2}+n/\ell)$ steps.

As a corollary of Lemma 15, we state the following.

Algorithm SB-by-PB

{ *Simulating the broadcasts taken on the separable bus in row i of $\mathcal{M}$, using row i of $\mathcal{M}'$.* $\mathtt{D_L}$, $\mathtt{D_R}$, `t1`, `t2`, `t3`, *and* `t4` *are the local variables in each PE.* }

Phase 1: { *Local Simulation* }

In each $\mathcal{M}'_p$, by sequentially scanning those s_j^{L}, s_j^{R}, and the switch configuration taken by P_j $(p\ell \leq j < (p+1)\ell)$ stored in the PE's of $\mathcal{M}'_p$ from left to right and then from right to left, each P'_j obtains $r_j'^{\mathrm{L}}$ and $r_j'^{\mathrm{R}}$ in $\mathtt{D_L}$ and $\mathtt{D_R}$, and knows whether $(j,x) \in C_{l_p}^{\mathrm{L}}$ and $(j,x) \in C_{r_p}^{\mathrm{R}}$ hold for each $x \in \{\mathrm{L},\mathrm{R}\}$ where l_p and r_p are the column indexes of $\mathrm{LP}'_{\mathcal{M}'_p}$ and $\mathrm{RP}'_{\mathcal{M}'_p}$.

Phase 2: { *Global Simulation* }

for $p \leftarrow 0$ **to** $(n/\ell - 2)$ **do**

(1) $\mathrm{RP}'_{\mathcal{M}'_p}$ sends $\mathtt{D_R}$ to $\mathrm{LP}'_{\mathcal{M}'_{p+1}}$. The received data is stored in `t1`.

(2) $\mathrm{LP}'_{\mathcal{M}'_{p+1}}$ broadcasts `t1`$\oplus\mathtt{D_L}$ to all PE's in $\mathcal{M}'_{p+1}$. The received value is stored in `t2` of each PE.

(3) Each P'_j in $\mathcal{M}'_{p+1}$ does $\mathtt{D}_x \leftarrow$ `t2` if $(j,x) \in C_{(p+1)\ell}^{\mathrm{L}}$ $(x \in \{\mathrm{L},\mathrm{R}\})$.

for $p \leftarrow (n/\ell - 1)$ **to** 1 **do**

(1) $\mathrm{LP}'_{\mathcal{M}'_p}$ sends $\mathtt{D_L}$ to $\mathrm{RP}'_{\mathcal{M}'_{p-1}}$. The received data is stored in `t3`.

(2) $\mathrm{RP}'_{\mathcal{M}'_{p-1}}$ broadcasts `t3` to all PE's in $\mathcal{M}'_{p-1}$. The received data is stored in `t4` of each PE.

(3) Each P'_j in $\mathcal{M}'_{p-1}$ does $\mathtt{D}_x \leftarrow$ `t4` if $(j,x) \in C_{p\ell-1}^{\mathrm{R}}$ $(x \in \{\mathrm{L},\mathrm{R}\})$.

end of SB-by-PB

Figure 3.4: Algorithm SB-by-PB.

Corollary 4 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable bus in row i (resp. column i) of $\mathcal{M}$ can be simulated in row i (resp. column i) of $\mathcal{M}'$ in $O(n^{1/2})$ steps when $\ell = n^{1/2}$ $(0 \leq i < n)$.* ■

A close inspection of the algorithm used for proving Corollary 4 implies the following lemma[2] .

Lemma 16 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable bus in row i (resp. column i) of $\mathcal{M}$ can be simulated in row i (resp. column i) of $\mathcal{M}'$ in $O(n^{1/2})$ steps when $\ell = n$ $(0 \leq i < n)$.*

Proof: Consider the execution of SB-by-PB in a row of $\mathcal{M}'$ with $\ell = n^{1/2}$. As for Phase 1, since broadcasting buses are not used for this phase, the length of the broadcasting bus of $\mathcal{M}'$ does not matter. As for Phase 2, every time the broadcast occurs, there is only a single PE sending a value in the row, and thus this phase can be performed even if there is only one broadcasting bus covering the entire row of $\mathcal{M}'$. Hence, every operation in Phase 1 and 2 can be executed in the same time on the $n \times n$ MPB even when $\ell = n$. The proof for the simulation in each column is similar. Thus the conclusion follows. ■

Then, we can now improve the result of Lemma 15, as shown below.

Lemma 17 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable bus in row i (resp. column i) of $\mathcal{M}$ can be simulated in row i (resp. column i) of $\mathcal{M}'$ in $O(\ell^{1/2} + n/\ell)$ steps $(0 \leq i < n)$.*

Proof: Let us consider simulating the broadcasts taken on the separable bus in row i of $\mathcal{M}$ for a given step $\mathcal{S}$ of $\mathcal{M}$ only. To prove that the simulation can be performed in $O(\ell^{1/2} + n/\ell)$ steps, it suffices to show that the same task of Phase 1 of SB-by-PB can be achieved in $O(\ell^{1/2})$

[2] This lemma corresponds to Lemma 3 in Chapter 2.

steps. Here, we define P_j, P'_j, C^x_j, s^x_j, r^x_j, $\oplus$, $\mathcal{M}'_p$, $\mathrm{LP}'_{\mathcal{M}'p}$, $\mathrm{RP}'_{\mathcal{M}'p}$, and r'^x_j in the same way as in the proof of Lemma 15.

At Phase 1 of SB-by-PB, each $\mathcal{M}'_p$ locally simulates the broadcasts (i.e., computes r'^*_*). Since each $\mathcal{M}'_p$ can be seen as a linear array composed of ℓ consecutive PE's and a broadcasting bus, by executing the algorithm proving Lemma 16 within each $\mathcal{M}'_p$, every P'_j can know r'^{L}_j and r'^{R}_j in $O(\ell^{1/2})$ steps.

Next, consider letting each P'_j in $\mathcal{M}'_p$ know whether $(j,x) \in C^{\mathrm{L}}_{l_p}$ hold for each $x \in \{\mathrm{L},\mathrm{R}\}$ where l_p is the column index of $\mathrm{LP}'_{\mathcal{M}'p}$. Consider a broadcast operation in row i of $\mathcal{M}$ such that the bus configuration is the same as that of $\mathcal{S}$ and every P_j corresponding to $\mathrm{LP}'_{\mathcal{M}'p}$ for some p sends "1" to the port L. Then, by locally simulating this broadcast operation within each $\mathcal{M}'_p$, every P'_j in $\mathcal{M}'_p$ can know whether $(j,x) \in C^{\mathrm{L}}_{l_p}$ hold for each $x \in \{\mathrm{L},\mathrm{R}\}$ where l_p is the column index of $\mathrm{LP}'_{\mathcal{M}'p}$. (Here, note that if P'_j in $\mathcal{M}'_p$ obtains "1" for port X of P_j in this local simulation, it means that in the bus configuration of $\mathcal{S}$ the port X of P_j is connected to the port L of P_k corresponding to $\mathrm{LP}'_{\mathcal{M}'_p}$.) By the similar argument in the preceding paragraph, this local simulation can be performed in $O(\ell^{1/2})$ steps in each $\mathcal{M}'_p$. Similarly, in $O(\ell^{1/2})$ steps, every P'_j in $\mathcal{M}'_p$ can know whether $(j,x) \in C^{\mathrm{R}}_{r_p}$ hold for each $x \in \{\mathrm{L},\mathrm{R}\}$ where r_p is the column index of $\mathrm{RP}'_{\mathcal{M}'p}$.

Thus, the same computational task of Phase 1 of SB-by-PB can be done more efficiently in $O(\ell^{1/2})$ steps. Since Phase 2 of SB-by-PB needs $O(n/\ell)$ steps, the entire simulation can be completed in $O(\ell^{1/2} + n/\ell)$ steps. Thus, the conclusion follows. ■

In the proof of Lemma 17, we improved the time cost required for the Phase 1 of SB-by-PB by using the result of Lemma 16. Such an improvement is possible because the broadcasting buses of $\mathcal{M}'$ are partitioned by length ℓ and the algorithm proving Lemma 16 can be executed within each block in parallel. Because of this reason, we cannot apply the same speedup technique to the algorithm proving Lemma 3 in Chapter 2, though it is carried out in the same fashion as SB-by-PB.

Then, we obtain the following lemma immediately from Lemma 17.

Lemma 18 *$\mathcal{M}'$ can simulate any single step of $\mathcal{M}$ in $O(\ell^{1/2} + n/\ell)$ steps.*

Proof: $\mathcal{M}'$ can simulate the broadcast substep of a single step of $\mathcal{M}$, by first simulating the broadcasts taken along rows in parallel in each row, and then simulating those along columns similarly. This takes $O(\ell^{1/2} + n/\ell)$ steps from Lemma 17. As for the local comm. and compute substeps, $\mathcal{M}'$ can simulate them in a constant number of steps in each PE. Thus, the conclusion follows. ■

Next, we consider the lower bounds for simulating $\mathcal{M}$ by $\mathcal{M}'$.

Lemma 19 *There exists a single step of $\mathcal{M}$ that takes $\Omega(n/\ell)$ steps to be simulated on $\mathcal{M}'$.*

Proof: Consider the single step of $\mathcal{M}$ in which $\mathrm{PE}_{\mathcal{M}}[0,0]$ sends a value to $\mathrm{PE}_{\mathcal{M}}[0,n-1]$. It is obvious that this step must take $\Omega(n/\ell)$

steps to be simulated on $\mathcal{M}'$. ■

Lemma 20 *There exists a single step of $\mathcal{M}$ that takes $\Omega(\ell^{1/2})$ steps to be simulated on $\mathcal{M}'$.*

Proof: Consider the single step $\mathcal{S}$ of $\mathcal{M}$ whose broadcast substep consists of the following operations (here, L is some positive integer such that $L \leq n$ and $n \bmod L = 0$):

1. $\mathrm{PE}_{\mathcal{M}}[i,j]$ divides the row broadcasting bus if $(j \bmod L) = 0$ $(0 \leq i,j < n)$.

2. $\mathrm{PE}_{\mathcal{M}}[i,j]$ sends the content of variable a to the row broadcasting bus through port R if $(j \bmod L) = 0$ $(0 \leq i,j < n)$.

3. $\mathrm{PE}_{\mathcal{M}}[i,j]$ receives data from the row broadcasting bus through port L and stores it in variable b if $(j \bmod L) \neq 0$ $(0 \leq i,j < n)$.

Let us call the data broadcasted at this substep as *a-values*. Note that there are possibly n^2/L different a-values.

Take any algorithm $\mathcal{A}$ that correctly simulates $\mathcal{S}$ on $\mathcal{M}'$. Here, the simulation is carried out on the MPB model, and thus the simulating PE's can use only the local links and the statically partitioned broadcasting buses. Since each PE that initially holds an a-value is different from those PE's which will receive the value, every a-value must be transmitted through the local links and/or the broadcasting buses during the simulation. With these observations, we count the necessary steps for $\mathcal{A}$, by considering the following two cases:

Case 1: There exists an *a-value* transmitted only through local links.

Case 2: There is no *a-value* transmitted only through local links.

In Case 1, since the distance between the PE initially holding the a-value and the most distant PE which will receive it is $L-1$, $\mathcal{A}$ must take at least $L-1$ steps. On the other hand, in Case 2, since the total number of a-value is n^2/L, and the number of data which may be transmitted on the broadcasting buses is at most $2n^2/\ell$ in a single time step, $\mathcal{A}$ must take at least $\ell/(2L)$ steps. Thus, by letting $L = \ell^{1/2}$, in either case, $\mathcal{A}$ needs $\Omega(\ell^{1/2})$ steps.[3] Thus the conclusion follows. ■

Now, we can state the following theorem.

Theorem 5 *When $\ell = \Theta(n^{2/3})$, $\mathcal{M}'$ can simulate any single step of $\mathcal{M}$ in $\Theta(n^{1/3})$ steps. This time cost is optimal in the worst case.*

Proof: From Lemma 18, $\mathcal{M}'$ can simulate any single step of $\mathcal{M}$ in $O(n^{1/3})$ steps when $\ell = \Theta(n^{2/3})$. The optimality is from Lemma 19 and 20, since there exists a single step of $\mathcal{M}$ which cannot be simulated in $O(n^{1/3})$ steps by any algorithm if $\ell \neq \Theta(n^{2/3})$ and there exists a single step of $\mathcal{M}$ which must take $\Omega(n^{1/3})$ steps to be simulated when $\ell = \Theta(n^{2/3})$. ■

[3] The lower bound $\Omega(n^{1/2})$ for simulating the HV-RM by the MWMB presented in Chapter 2 is derived from this, since the $n \times n$ MWMB is the same as the $n \times n$ MPB with $\ell = n$.

3.3 Simulation of MSB by MRSB

In this section, we consider simulating the $n \times n$ MSB by the $n \times n$ MRSB. Let $\mathcal{M}$ denote the $n \times n$ MSB, and let $\mathcal{M}'$ the $n \times n$ MRSB. As in Section 3.2, we write $\mathrm{PE}_{\mathcal{M}}[i,j]$ and $\mathrm{PE}_{\mathcal{M}'}[i,j]$ for denoting $\mathrm{PE}[i,j]$ of $\mathcal{M}$ and $\mathrm{PE}[i,j]$ of $\mathcal{M}'$ respectively, and assume that the computing power of PE's, the bandwidth of local links, and that of broadcasting buses are equivalent in both $\mathcal{M}$ and $\mathcal{M}'$.

We begin by proving lemmas for simulating broadcasts of $\mathcal{M}$ by $\mathcal{M}'$.

Lemma 21 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable bus in row $i\ell$ (resp. column $i\ell$) of $\mathcal{M}$ can be simulated in row $i\ell$ (resp. column $i\ell$) of $\mathcal{M}'$ in $O(\ell)$ steps $(0 \leq i < n/\ell)$.*

Proof: Take any single step $\mathcal{S}$ of $\mathcal{M}$ and $i \in \{0, 1, \ldots, n/\ell - 1\}$. Let us consider simulating the broadcasts taken on the separable bus along row $i\ell$ of $\mathcal{M}$ only, those on the bus along column $i\ell$ of $\mathcal{M}$ can be simulated similarly.

Let P_j and P'_j denote $\mathrm{PE}_{\mathcal{M}}[i\ell, j]$ and $\mathrm{PE}_{\mathcal{M}'}[i\ell, j]$ respectively $(0 \leq j < n)$. C_j^x, s_j^x, r_j^x, $\oplus$, $\mathcal{M}'_p$, $\mathrm{LP}'_{\mathcal{M}'p}$, $\mathrm{RP}'_{\mathcal{M}'p}$, and $r_j'^x$ are defined in the same way as in the proof of Lemma 15. Then, in Figure 3.5, we show an algorithm that simulates the broadcasts performed along row $i\ell$ of $\mathcal{M}$. Each r_j^x is stored in variable $\mathtt{D}_x$ of P'_j when the algorithm terminates.

After the execution of (2-2) of Phase 2, for each $p \in \{0, \ldots, n/\ell - 1\}$, r_{lp}^{L} and r_{rp}^{R} are obtained respectively in $\mathtt{D}_{\mathrm{L}}$ and $\mathtt{D}'_{\mathrm{R}}$ of $\mathrm{LP}'_{\mathcal{M}'p}$ where l_p is the column index of $\mathrm{LP}'_{\mathcal{M}'p}$ and r_p is that of $\mathrm{RP}'_{\mathcal{M}'p}$. Then, using these information, each PE can update its $\mathtt{D}_{\mathrm{L}}$ and $\mathtt{D}_{\mathrm{R}}$ appropriately at Phase

Algorithm SB-by-RSB
{ *Simulating the broadcasts taken on the separable bus in row $i\ell$ of $\mathcal{M}$, using row $i\ell$ of $\mathcal{M}'$.* $\mathtt{D}_{\mathrm{L}}$, $\mathtt{D}_{\mathrm{R}}$, *and* $\mathtt{D}'_{\mathrm{R}}$ *are the local variables in each PE.* }

Phase 1: { *Local Simulation* }

(1-1) Execute the Phase 1 of SB-by-PB.

(1-2) In each $\mathcal{M}'_p$, the content of $\mathtt{D}_{\mathrm{R}}$ of $\mathrm{RP}'_{\mathcal{M}'_p}$ is transferred to $\mathrm{LP}'_{\mathcal{M}'_p}$. This value is stored in $\mathtt{D}'_{\mathrm{R}}$ of $\mathrm{LP}'_{\mathcal{M}'_p}$.

Phase 2: { *Global Simulation* }

(2-1) Each $\mathrm{LP}'_{\mathcal{M}'_p}$ divides the row bus if $(p\ell, \mathrm{L}) \notin C^{\mathrm{R}}_{(p+1)\ell-1}$.

(2-2) Each $\mathrm{LP}'_{\mathcal{M}'_p}$ sends the content of $\mathtt{D}_{\mathrm{L}}$ and that of $\mathtt{D}'_{\mathrm{R}}$ to the bus through port L and R respectively. The values received from port L and R are stored in $\mathtt{D}_{\mathrm{L}}$ and $\mathtt{D}'_{\mathrm{R}}$.

Phase 3: { *Local Propagation* }
In each $\mathcal{M}'_p$, PE's update $\mathtt{D}_{\mathrm{L}}$ and $\mathtt{D}_{\mathrm{R}}$ appropriately using the information of $\mathtt{D}_{\mathrm{L}}$ and $\mathtt{D}'_{\mathrm{R}}$ of $\mathrm{LP}'_{\mathcal{M}'_p}$.

end of SB-by-RSB

Figure 3.5: Algorithm SB-by-RSB.

3. The correctness is straightforward and we omit the details. Phase 1 and 3 can be performed in $O(\ell)$ steps, since each block $\mathcal{M}'_p$ consists of ℓ consecutive PE's. Phase 2 takes $O(1)$ steps. Hence, the conclusion follows. ■

Lemma 22 *For any single step of $\mathcal{M}$, the broadcasts taken on the separable buses along rows (columns) of $\mathcal{M}$ can be simulated on $\mathcal{M}'$ in $O(\ell)$ steps.*

Proof: Take any single step $\mathcal{S}$ of $\mathcal{M}$. Let us consider simulating the

Algorithm SBs-by-RSBs
{ *Simulating the broadcasts taken along rows of* $\mathcal{M}$, *using* $\mathcal{M}'$. }

Stage 1: { *Local Simulation* }
Execute the Phase 1 of SB-by-RSB in each row in parallel.

Stage 2: { *Global Simulation* }
In each $\mathcal{B}'_p$, do the following:
for $i \leftarrow 0$ **to** $(\ell - 1)$ **do**

Execute the Phase 2 of SB-by-RSB for the row i of $\mathcal{B}'_p$.

Stage 3: { *Local Propagation* }
Execute the Phase 3 of SB-by-RSB in each row in parallel.

end of SBs-by-RSBs

Figure 3.6: Algorithm SBs-by-RSBs.

row broadcasts only, the column broadcasts can be simulated similarly.

We divide $\mathcal{M}'$ into n/ℓ disjoint *bands* $\mathcal{B}'_p$ $(0 \leq p < n/\ell)$. Each $\mathcal{B}'_p$ consists of $\mathrm{PE}_{\mathcal{M}'}[i,j]$ $(p\ell \leq i < (p+1)\ell,\ 0 \leq j < n)$, i.e., $\mathcal{B}'_p$ contains row i of $\mathcal{M}'$ $(p\ell \leq i < (p+1)\ell)$. The row i of $\mathcal{B}'_p$ is the row $p\ell + i$ of $\mathcal{M}'$ $(0 \leq i < \ell,\ 0 \leq p < n/\ell)$. Then, in Figure 3.6, we show an algorithm that simulates the broadcasts along rows of $\mathcal{M}$.

Stage 1 and Stage 3 can be performed in $O(\ell)$ steps from Lemma 21. As for Stage 2, it can be done in $O(\ell)$ steps in the following way. In each $\mathcal{B}'_p$, only the row 0 of $\mathcal{B}'_p$ has a broadcasting bus (restricted separable bus) whereby Phase 2 of SB-by-RSB is performed. Hence, for each row i $(\neq 0)$ of $\mathcal{B}'_p$, the data necessary for the execution of Phase 2 of SB-by-RSB must be moved to row 0 of $\mathcal{B}'_p$, and after the data being processed, the result must be moved back to the row. These operations can be done by just shifting the data synchronously with each iteration of the for-loop. Thus, Phase 2 can be completed in $O(\ell + \ell) = O(\ell)$

steps, and the conclusion follows. ■

Then, we have the following lemma.

Lemma 23 *$\mathcal{M}'$ can simulate any single step of $\mathcal{M}$ in $O(\ell)$ steps.*

Proof: $\mathcal{M}'$ can simulate the broadcast substep of a single step of $\mathcal{M}$, by first simulating the broadcasts taken along rows, and then simulating those along columns. This takes $O(\ell)$ steps from Lemma 22. As for the local comm. and compute substeps, $\mathcal{M}'$ can simulate them in a constant number of steps in each PE. Thus, the conclusion follows. ■

The lower bound for simulating $\mathcal{M}$ by $\mathcal{M}'$ is given in the following lemma.

Lemma 24 *There exists a single step of $\mathcal{M}$ that takes $\Omega(\ell)$ steps to be simulated on $\mathcal{M}'$.*

Proof: Consider the single step of $\mathcal{M}$ in which $\mathrm{PE}_{\mathcal{M}}[0,0]$ sends a value to $\mathrm{PE}_{\mathcal{M}}[0,\ell/2]$. It is obvious that this step must take $\Omega(\ell)$ steps to be simulated on $\mathcal{M}'$. ■

Now, we obtain the following theorem.

Theorem 6 *$\mathcal{M}'$ can simulate any single step of $\mathcal{M}$ in $O(\ell)$ steps. This time cost is optimal in the worst case.*

Proof: From Lemma 23 and 24. ■

3.4 Influence of Propagation Delays

Although we assumed that the propagation delay of the broadcasting buses was a constant time, we cannot neglect the influence of the delay when the mesh size becomes large. Here, as in [13], let us take the following assumptions:

- The propagation delay of a bus is proportional to the sum of its signal propagation delay and device propagation delay.
- The signal propagation delay of an x-length bus is $O(x^\alpha)$ for some $\alpha \geq 0$.
- The device propagation delay of a bus is neglectable (Case 1), or is proportional to the number of the switch elements inserted to the bus (Case 2).

Then, in Table 3.1, we show the necessary time to perform any single step of the $n \times n$ MSB.

Table 3.1: Time costs to perform a single step of the $n \times n$ MSB when the propagation delay cannot be neglected.
(In Case 1 we can neglect the device propagation delay, and in Case 2 we cannot. Each mesh is of size $n \times n$, and the broadcasting buses of the MPB are partitioned with $\ell = n^{2/3}$.)

models	time costs	
	Case 1	Case 2
MSB	$O(\ n^\alpha\ )$	$O(\ n^\alpha + n\ )$
MPB	$O(\ n^{1/3} \cdot n^{2\alpha/3}\ )$	$O(\ n^{1/3} \cdot n^{2\alpha/3}\ )$
MRSB	$O(\ \ell \cdot n^\alpha\ )$	$O(\ \ell \cdot (n^\alpha + n/\ell)\ )$

As for the Case 1, we can see that the MPB can perform a single step of the MSB as efficiently as the MSB if $\alpha = 1$, and that it is even superior to the MSB if $\alpha > 1$. As for the Case 2, the MPB is equal to (when $\alpha = 1$) or superior to (when $\alpha \neq 1$) the MSB, and the MRSB has the same efficiency as the MSB if $\ell \leq n^{1-\alpha}$. But from a theoretical point of view, in these cases (except the MPB in Case 2 with $\alpha < 1$) there is less advantage of augmenting ordinary meshes with broadcasting buses, for it takes $\Omega(n)$ time to perform a single step of the $n \times n$ MSB.

Chapter 4

Conclusions

In Chapter 2, we have presented an algorithm that simulates the HV-RM of size $n \times n$ in $\Theta(\sqrt{n})$ steps on the MWMB of size $n \times n$, and an algorithm that simulates the RM of size $n \times n$ in $O(\log^2 n)$ steps on the HV-RM of size $n \times n$. As for the former algorithm, we have proved that its time cost is optimal in the worst case under the assumption on the processor mapping. Also, we have shown that the RM of size $n \times n$ can be simulated in $O((\frac{n}{m})^2 \log n \log m)$ steps on the HV-RM of size $m \times m$, and in $O((\frac{n}{m})^2 \sqrt{m} \log n \log m)$ steps on the MWMB of size $m \times m$ $(m < n)$. Although the time cost $O((\frac{n}{m})^2 \log n \log m)$ is less efficient than $O((\frac{n}{m})^2 \log m \log \frac{n}{m})$ given in [9], our algorithm only needs the power of HV-RM, which is simpler and weaker model than that used in [9].

In Chapter 3, we have shown that the MSB of size $n \times n$ can be simulated in $\Theta(n^{1/3})$ steps by the MPB of size $n \times n$ when $\ell = \Theta(n^{2/3})$. Also, we have proved that the time cost is optimal in the worst case under the assumption on the processor mapping. Comparing it with the result that the MSB of size $n \times n$ (= the HV-RM of size $n \times n$) can be

simulated in $\Theta(n^{1/2})$ steps by the MWMB of size $n \times n$ (= the MPB of size $n \times n$ with $\ell = n$), we can say that the time cost for simulating the MSB of size $n \times n$ is reduced from $\Theta(n^{1/2})$ to $\Theta(n^{1/3})$ without increasing the number of switch elements. Also, we have proved that the MSB of size $n \times n$ can be simulated in $\Theta(\ell)$ steps by the MRSB of size $n \times n$. We have shown that the time cost is optimal in the worst case under the assumption on the processor mapping. Since the number of sectioning switches used in the MRSB of size $n \times n$ is $2n^2/\ell^2$, our result shows that we can reduce the number of switch elements used in the MSB by the factor of ℓ^2 with only paying the extra time cost proportional to ℓ.

By combining these simulation algorithms properly, we can obtain the time costs for simulating the RM on the other models. Since it was argued that the RM can be used as a universal chip capable of simulating any equivalent-area architecture (e.g. the pyramid, mesh of trees, hypercube, etc.) without loss in time, our combined simulation algorithms provide the upper bounds for the HV-RM, MWMB, MSB, MPB, and MRSB to simulate other equivalent-area architectures, respectively.

We proved that three of our simulation algorithms are optimal in the worst case under the processor mapping. However, we do not know if there exists more efficient algorithm for the simulation of the RM by the HV-RM. Also, the upper-bounds mentioned in the preceding paragraph are not shown to be tight, and there may be more efficient algorithms. We are currently studying these problems.

Bibliography

[1] P. Baglietto, M. Maresca, and M. Migliardi. Image Labeling by Transitive Closure on the Polymorphic Processor Array. Technical report, DIST University of Genoa, March 1994. ftp://rigel.dist.unige.it/pub/docs/smimp_reports/smimp-03.ps.gz.

[2] Y. Ben-Asher, D. Gordon, and A. Schuster. Optimal Simulations in Reconfigurable Arrays. Technical Report #716, Technion - Israel Institute of Technology, Computer Science Department, February 1992.

[3] Y. Ben-Asher, D. Gordon, and A. Schuster. Efficient Self-Simulation Algorithms for Reconfigurable Arrays. *J. of Parallel and Distributed Computing*, 30(1):1–22, October 1995.

[4] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster. The Power of Reconfiguration. *J. of Parallel and Distributed Computing*, 13:139–153, 1991.

[5] D. Bhagavathi, P. J. Looges, S. Olariu, J. L. Schwing, W. Shen, L. Wilson, and J. Zhang. Time- and VLSI-Optimal Sorting Algo-

rithms on Meshes with Multiple Broadcasting. In *Proc. Int'l Conf. Parallel Processing*, volume III, pages 196–201, 1993.

[6] V. Bokka, H. Gurla, S. Olariu, and J. L. Schwing. Time- and VLSI-optimal Convex Hull Computation on Meshes with Multiple Broadcasting. *Information Processing Letters*, pages 273–280, 1995.

[7] V. Bokka, H. Gurla, S. Olariu, and J. L. Schwing. Time-Optimal Domain Specific Querying on Enhanced Meshes. *IEEE Trans. on Parallel and Distributed Systems*, 8(1):13–24, 1997.

[8] K. L. Chung. Prefix Computations on a Generalized Mesh-Connected Computer with Multiple Buses. *IEEE Trans. on Parallel and Distributed Systems*, 6(2):196–199, February 1995.

[9] J. A. Fernández-Zepeda, R. Vaidyanathan, and J. L. Trahan. Scaling Simulation of the Fusing-Restricted Reconfigurable Mesh. *IEEE Trans. on Parallel and Distributed Systems*, 9(9):861–871, September 1998.

[10] S. Fujita and M. Yamashita. Fast Gossiping on Mesh-Bus Computers. *IEEE Trans. on Computers*, 45(11):1326–1330, 1996.

[11] H.Li and Q.F.Stout, editor. *Reconfigurable Massively Parallel Computers*, pages 50–52. Printice-Hall, Inc., 1991.

[12] R. Lin, S. Olariu, J. L. Schwing, and J. Zhang. Simulating Enhanced Meshes, with Applications. *Parallel Processing Letters*, 3:59–70, 1993.

[13] T. Maeba, M. Sugaya, S. Tatsumi, and K. Abe. An Influence of Propagation Delays on the Computing Performance in a Processor Array with Separable Buses. *IEICE Trans. A*, J78-A(4):523–526, 1995.

[14] T. Maeba, M. Sugaya, S. Tatsumi, and K. Abe. Semigroup Computations on a Processor Array with Partitioned Buses. *IEICE Trans. A*, J80-A(2):410–413, 1997.

[15] T. Maeba, S. Tatsumi, and M. Sugaya. Algorithms for Finding Maximum and Selecting Median on a Processor Array with Separable Global Buses. *IEICE Trans. A*, J72-A(6):950–958, 1989.

[16] M. Maresca. Polymorphic Processor Arrays. *IEEE Trans. on Parallel and Distributed Systems*, 4(5):490–506, May 1993.

[17] Y. Matias and A. Schuster. Fast, Efficient Mutual and Self Simulations for Shared Memory and Reconfigurable Mesh. In *Proc. of the 7th IEEE Symp. on Parallel and Distributed Processing*, pages 238–246, 1995.

[18] S. Matsumae and N. Tokura. Simulation Algorithms among Enhanced Mesh Models. Technical Report 99–ICS–1, Dept. of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University, 1999.

[19] R. Miller, V. K. Prasanna-Kumar, D. Reisis, and Q. F. Stout. Meshes with Reconfigurable Buses. In *Proc. of the fifth MIT Conference on Advanced Research in VLSI*, pages 163–178, Boston, 1988.

[20] R. Miller, V. K. Prasanna-Kumar, D. I. Reisis, and Q. F. Stout. Parallel Computations on Reconfigurable Meshes. *IEEE Trans. on Computers*, 42(6):678–692, June 1993.

[21] M. M. Murshed and R. P. Brent. Algorithms for Optimal Self-Simulation of Some Restricted Reconfigurable Meshes. Technical Report TR-CS-97-16, Joint Computer Science Technical Report Series, The Australian National University, July 1997. http://discus.anu.edu.au:80/~murshed/.

[22] K. Nakano. A Bibliography of Published Papers on Dynamically Reconfigurable Architectures. *Parallel Processing Letters*, 5(1):111–124, 1995.

[23] M. Nigam and S. Sahni. Sorting n Numbers on $n \times n$ Reconfigurable Meshes with Buses. *J. of Parallel Distributed Computing*, 23(1):37–48, October 1994.

[24] V. K. Prasanna-Kumar and C. S. Raghavendra. Array Processor with Multiple Broadcasting. *J. of Parallel and Distributed Computing*, 4:173–190, 1987.

[25] 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*, 4(10):1073–1080, October 1993.

[26] H. Shi, G. X. Ritter, and J. N. Wilson. Simulations between Two Reconfigurable Mesh Models. *Information Processing Letters*, 55:137–142, August 1995.

[27] Q. F. Stout. Meshes with Multiple Buses. In *Proc. of the 27th Annual Symposium of Foundations of Computer Science*, pages 264–273, October 1986.

[28] J. L. Trahan, R. Vaidyanathan, and R. K. Thiruchelvan. On the Power of Segmenting and Fusing Buses. *J. of Parallel and Distributed Computing*, 34:82–94, 1996.

[29] B. Wang and G. Chen. Constant Time Algorithms for the Transitive Closure and Some Related Graph Problems on Processor Arrays with Reconfigurable Bus System. *IEEE Trans. on Parallel and Distributed Systems*, 1(4):500–507, October 1990.