Effective Partitioning of Static Global Buses for Small Processor Arrays

Susumu Matsumae*

Abstract—This paper shows an effective partitioning of static global row/column buses for tightly coupled 2D mesh-connected small processor arrays ("mesh", for short). With additional O(n/m (n/m + log m)) time slowdown, it enables the mesh of size $m \times m$ with static row/column buses to simulate the mesh of the larger size $n \times n$ with reconfigurable row/column buses ($m \le n$). This means that if a problem can be solved in $O(T)$ time by the mesh of size $n \times n$ with reconfigurable buses, then the same problem can be solved in $O(T$ n/m (n/m + log m)) time on the mesh of a smaller size $m \times m$ without a reconfigurable function. This time-cost is optimal when the relation $n \ge m$ log m holds (e.g., $m = n^{1-\varepsilon}$ for $\varepsilon > 0$).

Keywords—Processor Array, Dynamically Reconfigurable Bus, Statically Partitioned Bus, Scaling-Simulation, Polylogarithmic Time Simulation

1. INTRODUCTION

The mesh-connected processor array ("mesh", for short) is one of fundamental parallel computational models. Its architecture is suitable for VLSI implementation and allows for a high degree of integration. However, the mesh has a crucial drawback in that its communication diameter is quite large due to the lack of a broadcasting mechanism. To overcome this problem, many researchers have considered adding broadcasting buses to the mesh [1-6].

Consider a linear processor array of n processing elements (PEs) where each adjacent PEs can communicate with each other via the local communication link. If it has a global bus spanning the entire array, it would take only one step to perform a data broadcast operation, while it would take $O(n)$ steps if it has no global bus. Also, if it has a global bus spanning the entire array, it would take $O(n^{1/2})$ steps to perform a fundamental prefix semi-group computation, while it would take $O(n)$ steps again if it has no global bus. Thus, the power of broadcasting capability is inevitable for the mesh architecture to be an efficient computational model.

To avoid write-conflicts on a global bus and to make effective use of it, the global bus may be partitioned into smaller bus segments. It is an interesting problem that decides the effective/optimal length used for the partitioning of a bus. As for the simulation problem discussed in

※ This work is a revised version of the paper, "Efficient Partitioning of Static Buses for Processor Arrays of Small Size" presented at the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Busan, Korea, 2010. The author thanks the anonymous referees for their many helpful comments in improving the quality and readability of this paper. The MEXT Grant-in-Aid for Young Scientists (B) (20700014) partly supported this work.

Manuscript received October 1, 2010; accepted February 22, 2011.

Corresponding Author: Susumu Matsumae

* Dept. of Information Science, Graduate School of Science and Engineering, Saga University, Saga, Japan (matsumae@is.saga-u.ac.jp)

[7], we successfully proved that the problem can be solved optimally in $O(n^{1/3})$ steps if we partition every row/column buses into sub-buses of length $n^{2/3}$. However, in general, the effective/optimal bus length depends on the problem to be solved, and hence cannot be fixed in advance.

Recently, the dynamically reconfigurable bus systems gained much attention due to its powerful computational power [8]. It can dynamically change its bus configuration during the execution of programs. The reconfigurable function enables the models to make efficient use of broadcast buses, and to solve many important, fundamental problems efficiently, mostly in a constant or polylogarithmic time. Such reconfigurability, however, makes the bus systems complex and causes negative effects on the communication latency of global buses [9].

In this paper, we investigate an effective partitioning of static row/column buses for the mesh that is of a small size. In [10], we have shown that the mesh of size $n \times n$ with static row/column buses can simulate the mesh of the same size $n \times n$ with reconfigurable row/column buses with additional $O(\log n)$ step slowdown. This means that if a problem is solved in $O(T)$ time by the mesh of size $n \times n$ with reconfigurable row/column buses, then the same problem can be solved as well in only $O(T \log n)$ time on the mesh of the same size $n \times n$ without reconfigurable function.

Here in this paper, we extend the result to the case where the simulating mesh is of smaller size. Such simulation is called *scaling-simulation*. In general, massively parallel computing assumes a huge number of processors, but we cannot always prepare the required number of processors at hand. Hence, the scaling-simulation technique is practically important. We prove that the mesh of size $m \times m$ with static row/column buses can simulate the mesh of larger size $n \times n$ with reconfigurable row/column buses with additional $O(n/m\ (n/m + \log m))$ slowdown ($m \leq n$). This time-cost is optimal when the relation $n \geq m \log m$ holds (e.g., $m = n^{1-\varepsilon}$ for $\varepsilon > 0$), because the time-cost becomes $O(n^2/m^2)$ which matches the trivial lower bound derived from the ratio of the number of simulated processors to that of simulating ones.

This paper is organized as follows: Section 2 explains several models of mesh-connected parallel computers with global buses. Section 3 defines the problem formally and introduces the basic approach to solve it. Section 4 describes our algorithm. Lastly, Section 5 offers concluding remarks.

2. MODELS

An $n \times n$ mesh consists of n^2 identical SIMD processors or processing elements (PEs) 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 PEs at $(i \pm 1, j)$ and $(i, j \pm 1)$, provided that 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 the addition of broadcasting buses along every row and column. 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 (Fig. 1 and 2).

A single time step of the MSB and the MPB is composed of the following three sub-steps:

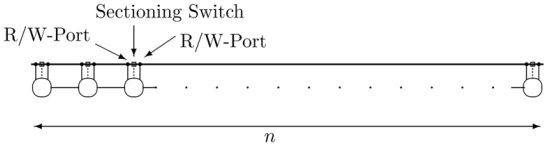


Fig. 1. A separable bus along a row of the $n \times n$ MSB. Each PE has access to the bus via the two ports beside the sectioning switch

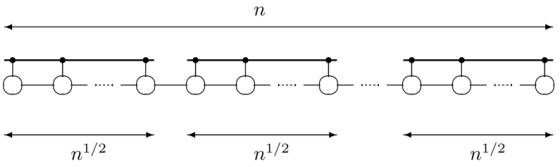


Fig. 2. A partitioned bus along a row of the $n \times n$ MPB. In this example, the bus is equally partitioned by length $n^{1/2}$

Local communication sub-step:
Every PE communicates with its adjacent PEs via local links.

Broadcast sub-step:
Every PE changes its switch configurations by local decision (this operation is only for the MSB). Then, along each broadcasting bus segment, several of the PEs connected to the bus send data to the bus, and several of the PEs on the bus receive the data transmitted on the bus.

Compute sub-step:
Every PE executes some local computation.

The bus accessing capability is similar to that of the Common-CRCW PRAM model. If there is a write-conflict on a bus, the PEs on the bus receive a special value $\perp$ (i.e., PEs can detect whether there is a write-conflict on a bus or not). If there is no data transmitted on a bus, the PEs on the bus receive a special value ø (i.e., PEs can know whether there is data transmitted on a bus or not).

3. PROBLEM

3.1 Scaling-Simulation

In this paper, we consider the (*scaling*) *simulation* of the $n \times n$ MSB by the $m \times m$ MPB ($m \leq n$). To simplify the exposition, we assume that $n \bmod m = 0$. We define the processor mapping as follows: each PE$[i, j]$ of the $m \times m$ MPB simulates PE$[x, y]$ of the $n \times n$ MSB ($i\ n/m \leq x < (i+1)\ n/m$, $j\ n/m \leq y < (j+1)\ n/m$). It should be noted that, in this mapping, each PE of the $m \times m$ MPB simulates the corresponding $(n/m) \times (n/m)$ sub-mesh of the $n \times n$ MSB. We assume that the computing power of PEs, the bandwidth of local links, and that of broadcasting buses are equivalent

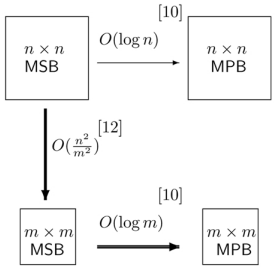


Fig. 3. The simulation costs among the MSB and MPB with different sizes ($m \leq n$)

in both simulated and simulating meshes. Throughout the paper, we assume that the simulation is done step-by-step, that is, we consider how to simulate any single step of the MSB by using the MPB. The trivial lower bound for the scaling-simulation problem is $\Omega(n^2/m^2)$, which is the ratio of the number of simulated processors to that of simulating ones.

Fig. 3 shows the relation among MSB and MPB for different sizes $n \times n$ and $m \times m$ ($m \leq n$). By combining the following 2 algorithms (thick arrows in Fig. 3):

1) the $O(n^2/m^2)$ step algorithm [12] for simulating the $n \times n$ MSB by the $m \times m$ MPB, and
2) the $O(\log m)$ step algorithm [10] for simulating the $m \times m$ MSB by the $m \times m$ MPB,

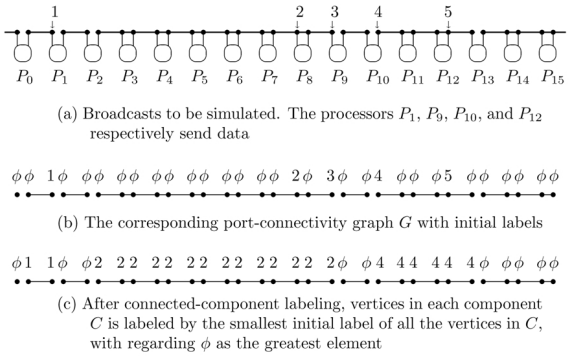
we can immediately obtain that the $n \times n$ MSB can be simulated in $O(n^2/m^2 \log m)$ steps by the $m \times m$ MPB. However, this does not match the lower bound unless we assume that m is a fixed constant. In [11], we have improved the simulation cost and proved that any one step of the $n \times n$ MSB can be simulated in $O(n^2/m^2 + n^{1/3})$ steps by the MPB. Since n^2/m^2 becomes dominant when m is less than $O(n^{5/6})$, we can say that our scaling-simulation algorithm [11] is optimal when $m = O(n^{1-\varepsilon})$ for $\varepsilon > 1/6$. Here in this paper, we further improve the cost toward $O(n/m\ (n/m + \log m))$. If the relation $n \geq m \log m$ holds (e.g. $m = n^{1-\varepsilon}$ for $\varepsilon > 0$), the time-cost becomes optimal.

3.2 Scaling-Simulation by Connected-Component Labeling

In what follows, we focus on how to mimic the broadcast substep of the MSB using the MPB, because the local communication and the compute sub-steps of the MSB can be easily simulated in $O(n^2/m^2)$ steps by the MPB.

Here, the problem of simulating the broadcast sub-step of the MSB is explained by connected-component labeling (CC-labeling) for a *port-connectivity graph* (pc-graph). See Fig. 4 for an example. Vertices of the pc-graph correspond to the read/write-ports of PEs, and edges stand for the port-to-port connections. Each vertex is initially labeled by the value, which is sent through the corresponding port by the PE at the broadcast sub-step. If there is no data sent through the port, the vertex is labeled by ø. The CC-labeling is done in such a way that vertices in each component C is labeled by the smallest initial label of all the vertices in C, with regarding ø as the greatest value. These labels are called *component labels*. Obviously, the simulation of the broadcast substep of the MSB can be achieved in $O(T)$ steps on the MPB if the CC-labeling of the corresponding port-connectivity graph can be executed in $O(T)$ steps by the MPB.

In this paper, we solve the CC-labeling problem by a divide-and-conquer strategy composed



(a) Broadcasts to be simulated. The processors P_1, P_9, P_{10}, and P_{12} respectively send data

(b) The corresponding port-connectivity graph G with initial labels

(c) After connected-component labeling, vertices in each component C is labeled by the smallest initial label of all the vertices in C, with regarding ϕ as the greatest element

Fig. 4. Broadcasts on a separable bus along a row of the $n \times n$ MSB are simulated by connected-component labeling of the port-connectivity graph. Here, n=16

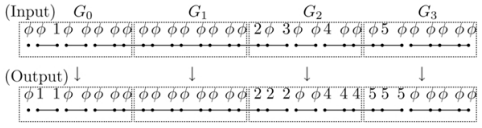
of the following three phases:

Phase 1: { *local labeling*}

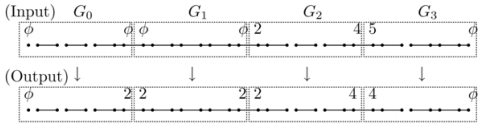
Divide the pc-graph into subgraphs, and label vertices locally within each of the subgraphs. The labels are called *local component labels*. In each subgraph, check whether the two vertices located at the boundary of the subgraph is connected to each other or not.

Phase 2: { *global labeling of boundary vertices*}

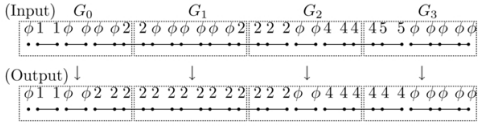
Label those vertices located at the boundary of each subgraph with component labels.



Phase 1: Connected-component labeling is locally executed within each subgraph G_i. Vertices are labeled by local component labels



Phase 2: Every boundary vertices is labeled by component label



Phase 3: Connected-component labeling is executed within each subgraph G_i for the consistency with Phase 2. Every vertex is labeled by component label

Fig. 5. An example for the connected-component labeling algorithm. The algorithm consists of three phases

Phase 3: { *local labeling for adjustment*}
Update vertex labels with component labels within each of the subgraphs for consistency with Phase 2

See Fig. 5 for an example.

4. SCALING-SIMULATION OF THE MSB BY THE MPB

In this section, we prove that any one step of the $n \times n$ MSB can be simulated in $O(n/m\ (n/m + \log m))$ steps by the $m \times m$ MPB ($m \leq n$).

To begin with, we formally introduce the following theorem:

Theorem 1 *[10] Any step of the* $n \times n$ MSB *can be simulated in* $O(\log n)$ *steps by the* $n \times n$ MPB.

Each row separable bus of the $n \times n$ MSB is simulated by the $m \times m$ MPB as follows (the case for a column separable bus is similar). To simulate the broadcasts taken along a row separable bus of the simulated $n \times n$ MSB, the CC-labeling problem of the corresponding pc-graph G is solved by the simulating $m \times m$ MPB. Here, we divide the pc-graph G into m disjoint subgraphs G_i, $0 \leq i < m$, of width n/m so that each G_i is locally stored in a single PE of the MPB. (Here, we say that a subgraph of pc-graph is of width w if it has $2w$ vertices corresponding to the R/W-ports of w consecutive PEs.) Then, each phase of the CC-labeling algorithm in the preceding section can be carried out as follows:

Phase 1: Since each sub-graph G_i is locally stored in a single PE of the simulating MPB, each PE of the MPB sequentially checks the vertex information from left to right and then right to left. This phase takes $O(n/m)$ steps because it simply scans $O(n/m)$ vertices and edges.

Phase 2: This phase is essentially the same as the problem of labeling a pc-graph G' of width m by using a row of the $m \times m$ MPB. The pc-graph G' is defined as follows:

- The graph G' contains only $2m$ vertices, which correspond to those vertices located at the boundaries of G_i ($0 \leq i < m$).
- The connectivity among the vertices of G' is the same as that of the original pc-graph G. Note that the connectivity information between two boundary vertices of each G_i is checked in $O(n/m)$ steps at Phase 1.

Thus, this phase takes $O(\log m)$ steps.

Phase 3: The operation is essentially the same as that of Phase 1. Hence, this phase takes $O(n/m)$ steps as well.

Thus, it takes $O(n/m + \log m)$ steps for a row of the $m \times m$ MPB to simulate the broadcasts taken along a separable row of the $n \times n$ MSB. Since each row of the MPB has to simulate the assigned n/m row separable buses of the MSB, as a whole, the time cost becomes $O(n/m\ (n/m + \log m))$ steps. Hence, we have the following lemma:

Lemma 1 *Any broadcasts taken along rows/columns of the* $n \times n$ MSB *can be simulated in* $O(n/m\ (n/m + \log m))$ *steps by the* $m \times m$ MPB ($m \leq n$).

Local communication and compute sub-steps can be simulated in $O(n^2/m^2)$ steps locally in each PE of the $m \times m$ MPB. The broadcast sub-step is simulated by first simulating broadcasts along rows and then simulating those along columns. As a whole, the simulation completes in $O(n^2/m^2 + n/m\ (n/m + \log m))$ steps (Lemma 1). Now, we can obtain the main theorem of this paper:

Theorem 2 *Any step of the* $n \times n$ MSB *can be simulated in* $O(n/m\ (n/m + \log m))$ *steps by the* $m \times m$ MPB ($m \leq n$).

5. CONCLUDING REMARKS

We have presented an algorithm that simulates the $n \times n$ MSB on the $m \times m$ MPB in $O(n/m\ (n/m + \log m))$ steps ($m \leq n$). If the relation $n \geq m \log m$ holds (e.g., $m = n^{1-\varepsilon}$ for $\varepsilon > 0$), the time-cost is optimal because it matches the lower bound $\Omega(n^2/m^2)$ derived from the ratio of the number of simulated PEs and that of simulating ones.

From a practical viewpoint, we expect that the communication latency of the broadcasting buses of the MPB model is much smaller than that of the MSB model. Each broadcasting bus of the $n \times n$ MSB can form the broadcasting bus whose length is n, and such a bus contains $O(n)$ sectioning switch elements in it. As for the $n \times n$ MPB, though the bus length is also at most n, but no switch element is inserted to the bus because it has no sectioning switch. Hence, compared to the MSB model, the MPB model has an advantage since each broadcasting bus has smaller propagation delay introduced by the switch elements inserted into the bus (i.e., device propagation delay), and thus our scaling-simulation algorithm is useful when the mesh size becomes so large that we cannot neglect the delay.

Furthermore, our scaling simulation algorithm has an advantage in that it can simulate the MSB model in which the concurrent write is resolved by the MIN rule [13] where the minimum among the sent values is received when a write-conflict occurs. This is because our algorithm simulates the broadcast operation of the MSB by connected-component labeling of the corresponding *port-connectivity graph* [13]. This fact may make up for the slowdown required for the simulation because the MSB with MIN-bus model is very powerful. For example, it can find the minimum among the values distributed over the mesh in a constant time.

REFERENCES

[1] V. K. Prasanna-Kumar and C. S. Raghavendra. "Array processor with multiple broadcasting," *J. of Parallel Distributed Computing*, 4:173-190, 1987.

[2] 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.

[3] 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.

[4] B. Wang and G. Chen. "Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. Parallel and Distributed Systems*, 1(4):500-507, October, 1990.

[5] 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.

[6] 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.

[7] S. Matsumae and N. Tokura. "Simulating a mesh with separable buses," *Transactions of Information Processing Society of Japan*, 40(10):3706-3714, 1999.

[8] R. Vaidyanathan and J. L. Trahan. *Dynamic Reconfiguration*. Kluwer Academic/Plenum Publishers, 2004.

[9] 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.

[10] S. Matsumae. "Polylogarithmic time simulation of reconfigurable row/column buses by static buses," In *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS2010)*, 2010.

[11] S. Matsumae. "Tight bounds on the simulation of meshes with dynamically reconfigurable row/column buses by meshes with statically partitioned buses," *J. of Parallel and Distributed Computing*, 66(10):1338-1346, 2006.

[12] 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.

[13] S. Matsumae and N. Tokura. "Simulation algorithms among enhanced mesh models," *IEICE Transactions on Information and Systems*, E82-D(10):1324-1337, 1999.

Susumu Matsumae

He received his M.E. and PhD degrees in computer science from Osaka University, Japan, in 1996 and 2000, respectively. From 2000 to 2001, he was a research associate at the Graduate School of Engineering Science, Osaka University. From 2001 to 2007, he was with the Department of Information Systems at Tottori University of Environmental Studies, Japan. In 2007, he joined the Department of Information Science at Saga University, Japan. His research interests include parallel algorithms and architectures, logic, and computational complexity.