Proc. 1988 International Conf. Parallel Processing

Data Movement Operations and Applications on Reconfigurable VLSI Arrays

Russ Miller
Dept. of Comp. Sci.
SUNY-Buffalo
Buffalo, NY 14260

V. K. Prasanna Kumar
Dept. of EE-Systems
USC
Los Angeles, CA 90089

Dionisios I. Reisis
Dept. of EE-Systems
USC
Los Angeles, CA 90089

Quentin F. Stout
Dept. of EECS
Univ. Michigan
Ann Arbor, MI, 48109

Abstract

This paper considers a *mesh with reconfigurable bus (reconfigurable mesh)*, that consists of a VLSI array of processors connected to a reconfigurable bus system. The N PEs are laid out as a square mesh in $O(N)$ VLSI area. The reconfiguration scheme can be used to dynamically obtain various interconnection patterns between the PEs. In fact, the array can be used as a universal chip capable of simulating any $O(N)$ area organization with a planar wiring layout without loss in time. The reconfiguration scheme also supports several parallel techniques developed for the CRCW PRAM. In this paper, we develop fundamental data movement operations for the reconfigurable mesh. These operations are used to give efficient solutions to a variety of problems involving graphs and digitized pictures. The running times of these algorithms are asymptotically superior to those developed for the mesh with multiple broadcasting, the mesh with multiple buses, the mesh-of-trees, and the pyramid computer.

1 Introduction

In this paper, we consider a reconfigurable VLSI array of processing elements that combines the advantages of a number of architectures including the mesh, pyramid, mesh-of-trees, and meshes with broadcast buses. Due to page limitations, this paper will only summarize a subset of the results that we have obtained for the reconfigurable mesh. The reader is referred to [7] for discussions and algorithms associated with the results given in this paper.

The *mesh with reconfigurable bus (reconfigurable mesh) of size N* consists of an $N^{1/2} \times N^{1/2}$ array of processors connected to a grid-shaped reconfigurable broadcast bus, where each processor has four locally controllable bus *switches*, as shown in Figure 1. Other than the buses and switches, the reconfigurable mesh is similar to the standard mesh in that it operates in SIMD mode and has $O(N)$ area, under the assumption that processors, switches, and individual links have constant size. In one unit of time each processor can perform standard arithmetic and boolean operations on its own data, can set any of its four switches, and can send and receive a piece of data from the bus.

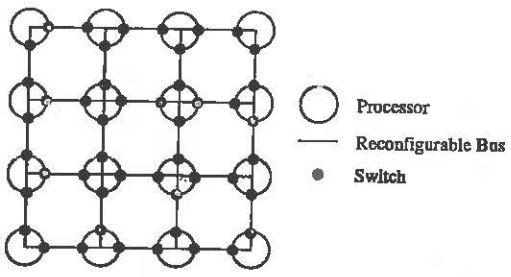

Figure 1: A reconfigurable mesh of size 16.

In each subbus shared by multiple processors, at any given time we assume that at most one processor may use the bus to broadcast a value, where a value consists of $O(\log N)$ bits. Notice that by setting the switches properly, sub-row (column) buses can be created within each row (column), sub-meshes with reconfigurable buses can be created, a global broadcast bus can be created, distinct buses can be created within distinct sets of contiguously labeled processors, and so forth.

Major advantages of the reconfigurable mesh are as follows.

1. Buses can be used to speed up parallel arithmetic and logic operations among data stored in different processors. The reconfiguration scheme supports several CRCW PRAM techniques. In fact, for some problems the reconfigurable mesh is superior to the PRAM.
2. The reconfigurable mesh provides an environment for efficient sparse data movement operations.
3. A significant asymptotic improvement can be achieved in the running times of algorithms that solve several problems on the reconfigurable mesh compared to efficient algorithms for the mesh-of-trees, pyramid, and mesh with static broadcast buses.
4. The reconfigurable mesh can act as a universal chip in that VLSI organizations with equivalent area and a planar wiring layout can be simulated without loss in time.

Many of the algorithms for the reconfigurable mesh (c.f., [7]) will continually reconfigure the system by setting the switches to give the desired substructures.

2 Related Architectures

It should be noted that although there are similarities, the reconfigurable mesh is very different from the CHiP project [15], the mesh augmented with broadcast buses[1, 3, 12, 16], and the *bus automaton* [4]. However, the reconfigurable mesh is similar to the *polymorphic-torus network* [6], with the major difference being that in the polymorphic-torus network there is an arbitrary crossbar in each processor to control connections between the north, south, east, and west bus ports. Finally, the reconfigurable mesh appears to be almost identical to the latest version of the *Content Addressable Array Parallel Processor (CAAPP)* [18], which was developed independently of the reconfigurable mesh.

3 Data Movement Operations

Data movement operations form the foundation of numerous algorithms for machines constructed as an interconnection of processors.

Proposition 3.1 *Given a set $S = \{a_i\}$ of N values, distributed one per processor on a reconfigurable mesh of size N so that processor P_i contains a_i, $0 \leq i \leq N-1$, and a unit-time binary associative operation $\otimes$, in $\Theta(\log N)$ time the parallel prefix problem can be solved so that each processor P_i knows $a_0 \otimes a_1 \otimes \ldots \otimes a_i$.*

We introduce a technique called *bus splitting*, in which processors exploit the ability to locally control the effective size of subbuses, to obtain the following Proposition.

Proposition 3.2 *Given a reconfigurable mesh of size N, in which each processor stores a bit of data, the logical OR of the data in each row (column), or the entire reconfigurable mesh, can be determined in $\Theta(1)$ time.*

The reconfigurable mesh can be superior to other parallel models for performing some computations. Consider, for example, computing the exclusive OR (EXOR) function of $N^{1/2}$ values stored in a row of the mesh. Note that [5] has shown that the exclusive OR function cannot be computed in $O(1)$ time on a PRAM using a polynomial number of processors. However, by exploiting the reconfigurability available, the EXOR function can be computed in $\Theta(1)$ time on the reconfigurable mesh.

Proposition 3.3 *Given a reconfigurable mesh of size N, in which each processor stores a bit of data, the exclusive OR (EXOR) of the $N^{1/2}$ data stored in a row (column) can be computed in $\Theta(1)$ time.*

Lemma 3.4 *Given a reconfigurable mesh of size N, suppose each processor in a row (column) stores a bit of data d_j, $0 \leq j \leq N^{1/2}-1$. Then, the computation of $\mathcal{F}_i$ given by $\mathcal{F}_i = \sum_{i=0}^{j} d_i$, $0 \leq j \leq N^{1/2}-1$, can be performed in $\Theta(1)$ time.*

Many parallel algorithms are designed to reduce data at intermediate stages of the algorithm. It is, therefore, often useful to be able to efficiently perform fundamental operations on reduced sets of data.

Proposition 3.5 *Given a reconfigurable mesh of size N, in which no more than one processor in each column stores a data value, the minimum (maximum) of these $O(N^{1/2})$ data items can be determined in $\Theta(1)$ time.*

By a somewhat more complicated sequence, Valiant's PRAM algorithm for finding the maximum [17] can be simulated on a reconfigurable mesh to find the maximum of all N values, assuming they are stored one value per processor.

Proposition 3.6 *Given a set of data items S of size N stored one per processor on a reconfigurable mesh of size N, the maximum value of S can be determined in $O(\log\log N)$ time.*

Proposition 3.7 *Given a set of bits S of size N stored one per processor on a reconfigurable mesh of size N, the EXOR of all items in S can be computed in $O(\log\log N)$ time.*

Proposition 3.8 *Suppose on a reconfigurable mesh of size N each processor has a label from a set of k distinct labels, $1 \leq k \leq N$. Further, suppose processors having the same label form arbitrary contiguous regions on the mesh. Then, each processor can know whether there is at least one tagged processor with its label in $\Theta(1)$ time. Also, given that there is at least one tagged processor for each label and all tagged processors with the same label store identical data, the tagged data can be broadcast to all other processors with the same label in the above time, for all labels in parallel.*

It is often desirable to model PRAM algorithms on other machines. In order to efficiently simulate the CRCW PRAM, one must be able to efficiently simulate the concurrent read and concurrent write properties. Define a *Random Access Read (RAR)* to be a data movement operation that models a concurrent read, in which each processor knows the index of another processor from which it wants to read data [11]. Similarly, a *Random Access Write (RAW)* will model a concurrent write in that each processor knows the index of a processor that it wishes to write to [11]. In case of multiple writes to the same processor, a tie-breaking scheme is used, such as minimum or maximum data value, or arbitrarily letting one value succeed.

Proposition 3.9 *Given a reconfigurable mesh of size N, in $O(k^{1/2}+\log N)$ time k data items may be moved in a RAR or RAW, where $k \leq N$.*

In fact, more efficient data movement can be performed if the distribution of the source processors, i.e., those processors sending data, as well as the destination processors, i.e., those processors receiving data, is uniform over the reconfigurable mesh.

Proposition 3.10 *Given a reconfigurable mesh of size N, if the number of source and destination processors within any block of size k^2 is $O(k)$, $1 \leq k \leq N^{1/2}$ then RAR and RAW can be performed in $O(\log N)$ time.*

Another fundamental operation that involves data movement is data reduction. Assume that each processor has at most one record having a *key* field and a *data* field. *Data reduction* will perform an associative binary operation on the data of records having the same key. At the end of the data reduction operation, each processor with key k will have the result of the binary operation performed over all data with key k.

Proposition 3.11 *Given a binary associative operator $\otimes$, data reduction can be performed on k distinct keys in $O(k^{1/2}+\log N)$ time on a reconfigurable mesh of size N, so that each processor knows the result of applying $\otimes$ over all data items with its key.*

Lemma 3.12 *Given a reconfigurable mesh of size N with k distinct keys randomly distributed one key per processor, the number of distinct keys can be determined in $O(k^{1/2}+\log N)$ time.*

4 Applications

In this section, we illustrate the performance of the reconfigurable mesh by giving simulations of other low wire area organizations, such as the mesh-of-trees and pyramid, discussing the use of the reconfigurable mesh as a universal chip, and by giving efficient parallel algorithms to solve problems involving graphs and images.

4.1 Simulations

Well known organizations such the mesh-of-trees and pyramid computer can be efficiently simulated by the reconfigurable mesh due to the numerous communications patterns that the reconfigurable mesh provides. In the first part of this section, we consider step by step simulation of the mesh-of-trees and pyramid.

A *mesh-of-trees (MOT) of base size N*, where N is an integral power of 4, has a total of $3N-2N^{1/2}$ processors. N of these are base processors arranged as a mesh of size N. Above each row and above each column of the mesh is a perfect binary tree of processors. Each row (column) tree has as its leaves an entire row (column) of base

processors. All row trees are disjoint, as are all column trees. Every row has exactly one leaf processor in common with each column tree. Each base processor is connected to 6 other processors (assuming they exist): 4 neighbors in the base, a parent in its row tree, and a parent in its column tree. Each processor in a row or column tree that is neither a leaf nor a root is connected to exactly 3 other processors in its tree: a parent and 2 children. Each root in a row or column tree is connected to its 2 children. Notice that in the MOT the processors in each row and in each column can be looked upon as placed at levels $0, 1, \ldots, k$ where $N^{1/2} = 2^k$.

Define a c-embedding of a hierarchical organization onto the reconfigurable mesh to have the following properties.

1. A constant number of processors of the hierarchical organization are mapped to each processor of the reconfigurable mesh.
2. The number of communication links between levels l and $l+1$, $0 \leq l \leq k-1$, incident on any row or column bus segment is $\leq c$.

Define a class of algorithms on a hierarchical organization to be *normalized algorithms* if the following hold.

1. During a computation step of a hierarchical algorithm, all data operated on are located at the same level of the hierarchical organization.
2. During a communication step of a hierarchical algorithm, communication is performed between at most two adjacent levels of the hierarchical organization.

[9] shows how to embed the mesh-of-trees into a mesh. This embedding is used to embed the mesh-of-trees into the reconfigurable mesh and obtain the following two propositions.

Proposition 4.1 *Any normalized algorithm running in $T(N)$ time on a mesh-of-trees of base size N can be simulated on a reconfigurable mesh of size N to finish in $O(T(N))$ time.*

Proposition 4.2 *Any algorithm running in $T(N)$ time on a mesh-of-trees of base size N, can be simulated on a reconfigurable mesh of size N to finish in $O(T(N)\log N)$ time. Further this time is optimal.*

We now turn our attention to the simulation of the reconfigurable mesh by the mesh-of-trees. During the execution of an algorithm on the reconfigurable mesh the buses are continuously configured. A configuration of the bus corresponds to partitioning the mesh into disjoint sets of contiguous processors. Reconfiguration of the bus can be simulated on the mesh-of-trees by identifying contiguous processors in the mesh. This reduces to the problem of identifying connected 1's in an $N^{1/2} \times N^{1/2}$ digitized image (see Section 5.2 for more details). On the mesh-of-trees this can be done in $O(\frac{\log^2 N}{\log\log N})$ time [9, 10].

Proposition 4.3 *A mesh-of-trees of base size N can simulate a reconfigurable mesh of size N in $O(\frac{T(N)\log^2 N}{\log\log N})$ time if the switch settings are dynamic, or $O(\frac{T(N)\log^2 N}{\log\log N} + \log^2 N)$ if the switch settings are static.*

We now turn our attention to relationships between the reconfigurable mesh and the pyramid computer. A *pyramid computer (pyramid) of size N* is a machine that can be viewed as a full, rooted, 4-ary tree of height $\log_4 N$, with additional horizontal links so that each horizontal *level* is a mesh. It is often convenient to view the pyramid as a tapering array of meshes. A pyramid of size N has at its base a mesh of size N, and a total of $\frac{4}{3}N - \frac{1}{3}$ processors. The levels are numbered so that the base is level 0 and the apex is level $\log_4 N$. A processor at level i is connected via bidirectional unit-time communication links to its 9 neighbors (assuming they exist): 4 siblings at level i, 4 children at level $i-1$, and a parent at level $i+1$.

An embedding of the pyramid into the reconfigurable mesh, similar to the mesh-of-trees embedding used in Propositions 4.1 and 4.2, is used to give the following.

Proposition 4.4 *Any algorithm running in time $T(N)$ on a pyramid of size N can be simulated on a reconfigurable mesh of size N in $O(T(N))$ time.*

The $\Theta(N^{1/4})$ time solution to the connected 1's problem on a pyramid of size N [8] is used to give the following.

Proposition 4.5 *Any algorithm running on the reconfigurable mesh of size N in time $T(N)$ can be simulated on a pyramid of size N in $O(T(N)N^{1/4})$ time. This simulation is optimal.*

Theorem 4.1 *Any architecture that can be laid out in an $N^{1/2} \times N^{1/2}$ grid and use a planar wiring (assuming wires have unit width) can be simulated by the reconfigurable mesh in constant time per unit time of the target architecture.*

4.2 Graph Problems

The first problem considered in this section is that of computing the connected components of an undirected graph with $N^{1/2}$ vertices, given as an adjacency matrix. The $(i,j)^{th}$ entry of the adjacency matrix of the graph is initially stored in processor $P_{i,j}$ of the reconfigurable mesh. The algorithm that we use is based on the $O(\log N)$ time algorithm presented for the CRCW PRAM [14].

Theorem 4.2 *Given the adjacency matrix of an undirected graph with $N^{1/2}$ vertices distributed so that the $(i,j)^{th}$ element of the matrix is stored in processor $P_{i,j}$ of a reconfigurable mesh of size N, the connected components of the graph can be determined in $O(\log N)$ time.*

The reconfigurable mesh can also be used to provide efficient solutions to some graph problems that assume unordered edges as input.

Theorem 4.3 *The connected components of a V vertex graph given in unordered edge input format, can be computed in $O(V^{1/2})$ time on the reconfigurable mesh of size N, where $N^{1/2} \leq V \leq N$.*

Corollary 4.6 *A minimal spanning forest of a V vertex graph given in unordered edge input format, can be computed in $O(V^{1/2})$ time on the reconfigurable mesh of size N, where $N^{1/2} \leq V \leq N$.*

Several graph properties can be deduced once a spanning tree of the graph is determined [2]. Using Corollary 4.6 and the data movement operations presented in Section 3, the following results can be obtained.

Corollary 4.7 *Given N edges of a graph G with V vertices distributed one vertex per processor in a reconfigurable mesh of size N, $N^{1/2} \leq V \leq N$, in $O(V^{1/2})$ time, one can*

a) check if G is bipartite,

b) compute the cyclic index of G, and

c) compute the articulation points of G.

4.3 Image Problems

Many problems involving digitized images can be solved efficiently on the reconfigurable mesh. The input to these problems is an $N^{1/2} \times N^{1/2}$ digitized image distributed one pixel per processor on a reconfigurable mesh of size N so that processor $P_{i,j}$ has pixel (i,j). The problems that we examine focus on labeling *figures (connected components)* and determining properties of the figures. The reconfigurable bus is used to isolate individual figures so as to be able to efficiently extract information concerning multiple figures in a digitized image. A subbus is created and dedicated to keep track of all deliberations with respect to each figure.

Theorem 4.4 *Given an $N^{1/2} \times N^{1/2}$ digitized image mapped one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, in $O(\log N)$ time the figures (connected components) can be labeled.*

Theorem 4.5 *Given an $N^{1/2} \times N^{1/2}$ digitized image mapped one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, in $O(\log N)$ time a closest figure to each figure can be determined.*

Theorem 4.6 *Given an $N^{1/2} \times N^{1/2}$ digitized image mapped one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, in $O(\log^2 N)$ time the extreme points of the convex hull can be enumerated for every figure.*

Theorem 4.7 *Given an $N^{1/2} \times N^{1/2}$ digitized image mapped one pixel per processor onto the processors of a reconfigurable mesh of size N in a natural fashion, in $\Theta(1)$ time several geometric properties of a set S of pixels can be determined. These properties include marking and enumerating the extreme points of the convex hull of the points, determining the diameter of the points, determining a smallest enclosing box of the points, and determining a smallest enclosing circle of the points.*

5 Conclusion

This paper considers the reconfigurable mesh as a viable alternative to a variety of processor organizations. We have presented efficient implementations of fundamental data movement operations for the reconfigurable mesh and have shown that it can be used as a universal chip, in that the reconfigurable mesh is capable of simulating any organization of processors occupying the same area and using a planar wiring layout without loss of time. We have also presented algorithms that show how the reconfigurable mesh can efficiently solve a number of graph and image problems using the fundamental data movement operations. The running times of these algorithms are asymptotically superior to running times of solutions for the mesh with multiple broadcasting, the mesh with multiple buses, the mesh-of-trees, and the pyramid computer. Further, we have shown that there are problems for which solutions on the reconfigurable mesh are more efficient than those possible for a PRAM.

6 Acknowledgments

The work of R. Miller was supported in part by NSF grant DCR-8608640. The work of V.K. Prasanna-Kumar was supported in part by NSF grant IRI-8710863. The work of D.I. Reisis was supported in part by DARPA contract F 33615-84-K-1404 monitored by the Air Force Wright Aeronautical Laboratory. The work of Q.F. Stout was supported in part by NSF grant DCR-8507851, and by an Incentives for Excellence Award from Digital Equipment Corporation.

References

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

[2] M. Atallah and R. Kosaraju, *Graph problems on a mesh connected processor array*, JACM, 1983.

[3] S. H. Bokhari, *Finding Maximum on an Array Processor with a Global Bus*, IEEE Transactions on Computers, Vol. C-33, No. 2, February 1984, pp 133-139.

[4] D. M. Champion and J. Rothstein, *Immediate parallel solution of the longest common subsequence problem*, 1987 International Conference on Parallel Processing, 70-77.

[5] M. Furst, J. Saxe and M. Sipser, *Parity, Circuits and Polynomial Time Hierarchy*, Proc. IEEE Foundations on Computer Science, pp. 260-270, 1981.

[6] H. Li and M. Maresca, *Polymorphic-Torus Network*, Proc. International Conference on Parallel Processing, 1987.

[7] R. Miller, V.K. Prasanna Kumar, D. Reisis, and Q.F. Stout, *Parallel computations on reconfigurable meshes*, Tech. Rept. 229, Dept. of EE-Systems and IRIS, USC, March, 1988.

[8] R. Miller and Q. F. Stout, *Data Movement Techniques for the Pyramid Computer*, SIAM Journal on Computing, Vol. 16, No. 1, pp. 38-60, February 1987.

[9] R. Miller and Q. F. Stout, *Some graph and image processing algorithms for the hypercube*, Hypercube Multiprocessors 1987, SIAM, pp. 418-425, 1987.

[10] R. Miller and Q. F. Stout, *Parallel Algorithms for Regular Architectures*, The MIT Press, 1988.

[11] D. Nassimi and S. Sahni, *Data Broadcasting in SIMD Computers*, IEEE Transactions on Computers 1981.

[12] V. K. Prasanna Kumar and C. S. Raghavendra, *Array Processor with Multiple Broadcasting*, Proceedings of the 1985 Annual Symposium on Computer Architecture, June 1985.

[13] V. K. Prasanna Kumar and M. Eshaghian, *Parallel Geometric algorithms for Digitized pictures on Mesh of Trees organization*, International Conference on Parallel Processing, 1986.

[14] Y. Shiloach and U. Vishkin, *A $O(\log N)$ Parallel Connectivity Algorithm*, Journal of Algorithms 3, 1982.

[15] L. Snyder, *Introduction to the Configurable, Highly Parallel Computer*, Computer 1S(1):47-56, January, 1982.

[16] Q. F. Stout, *Mesh Connected Computers with Broadcasting*, IEEE Trans. on Computers C-32, pp. 826-830, 1983.

[17] L. G. Valiant, *Parallelism in comparison problems*, SIAM J. on Computing 3, 1975.

[18] C.C. Weems, S.P. Levitan, A.R. Hanson, E.M. Riseman, J.G. Nash, D.B. Shu, *The image understanding architecture*, COINS Tech. Rept. 87-76, University of Massachusetts at Amherst.