Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses

Mauricio J. Serrano, *Student Member, IEEE,* and Behrooz Parhami, *Senior Member, IEEE*

***Abstract*— A two-dimensional mesh of processing elements (PE's) with separable row and column buses (i.e., broadcast mechanisms for rows and columns that can be logically divided into a number of local buses through the use of PE-controlled switches) has been shown to be quite effective for semigroup computation, prefix computation, and a wide class of other computations that do not require excessive communication or data routing. For meshes with separable row/column buses, we show how semigroup and prefix computations can be performed with the same asymptotic time complexity without the provision of buses for every row and every column and discuss the VLSI implications of this new architecture. We find that with our basic two-level arrangement, square meshes are not optimal for the above algorithms and that the time complexity can be reduced by using rectangular meshes. However, our method of building higher order meshes allows the construction of an optimal square mesh from rectangular submeshes. We further observe that through more extensive use of PE-controlled switches in the mesh, a hierarchy of buses can be realized efficiently in VLSI. The time-complexity results are shown to correspond to those previously published when certain parameters of our design are fixed at special values.**

***Index Terms*—Mesh-connected computer, mesh with multiple broadcasting, mesh with row/column buses, optimal aspect ratio, prefix computation, semigroup computation.**

I. Introduction

A two-dimensional mesh-connected computer consists of N processors or processing elements (PE's) arranged in a grid, where adjacent processors are connected via local communication links. The PE located at the grid point (i, j) is connected to those located at $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$. This structure can be easily realized in VLSI and allows a high degree of integration. Meshes are natural structures for solving many problems in matrix computations and image processing. Thus, researchers have designed numerous algorithms for solving such problems on them [2], [6], [7], [9], [12], [14], [18], [23], [24].

The main disadvantage of a mesh is that the number of steps, or time, required by an algorithm is lower-bounded by its diameter, i.e., $T = \Omega(N^{1/2})$. Many computations can be performed with lower latency compared to a mesh-connected computer if specialized architectures or networks are used [17]. For example, both semigroup and prefix computations are naturally suited to a tree of PE's or, equivalently, a tree-structured Boolean network. Bilardi and Preparata have characterized the complexity of N-element prefix computations, showing that a spectrum of designs is possible with computation time T in the range $[\Omega(\log N), O(N)]$ and network size $O((N/T)\log(N/T))$ in the worst case. In particular, this gives a size of $O(N^{1/2}\log N)$ for $T = \theta(N^{1/2})$, which is better than the $O(N)$ size of a mesh with the same latency. But generality or flexibility always has a cost.

As a result of the diameter-based lower bound of $\Omega(N^{1/2})$, many researchers have proposed meshes augmented with nonlocal communication mechanisms such as broadcasting and bypass connections. In broadcast schemes, only one processor is allowed to put data on any particular bus at a given time. Bokhari [5] showed that by using a global bus connected to all PE's, the maximum of N numbers, assigned one per PE, can be found in $O(N^{1/3})$ time. Stout [21] showed that using a global bus the time to perform semigroup computations (finding the maximum being a special case) can be reduced to $O(N^{1/3})$. Stout also showed that problems with higher degrees of global communication, exemplified by sorting, have a lower bound of $\Omega(N^{1/2})$, so broadcast schemes do not help to reduce their asymptotic complexity, although the constants associated with computation time are reduced. He also showed that the median of N values can be found in $O(N^{1/3}\log^{2/3} N)$ time.

The problem with a single global bus is that it becomes a bottleneck whenever multiple values have to be transferred and this places a limit on the attainable speedup. Thus, multiple broadcasting schemes have been proposed. Prasanna-Kumar and Raghavendra [19] considered a mesh of $N^{1/2} \times N^{1/2}$ PE's with broadcast buses in each row and each column (Fig. 1(a)) and developed parallel algorithms for many problems in linear algebra, image processing, computational geometry, and numerical computations. Using this mesh, semigroup computations take $O(N^{1/6})$, median finding $O(N^{1/6}\log^{2/3} N)$, convex polygon, and nearest neighbor $O(N^{1/6})$ time.

Chen *et al.* [10] showed that square versions of this architecture are not the best form for semigroup and prefix computations, but by using a rectangular $N^{5/8} \times N^{3/8}$ mesh, the time for performing these computations can be reduced to $O(N^{1/8})$. They also claim to have developed algorithms for the median row and the median problem with time complexities of $O(N^{1/8})$ and $O(N^{1/8}\log N)$, respectively, using

Manuscript received September 11, 1991; revised June 1, 1992. This work was supported in part by the U.S. National Science Foundation under Grant MIP-9001618.

The authors are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560.

IEEE Log Number 9213470.

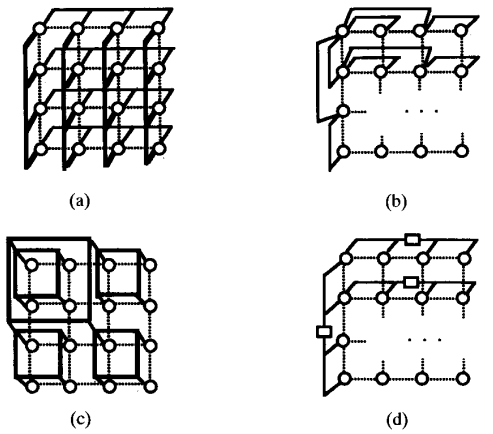

Fig. 1. Different interconnections schemes for a mesh: (a) mesh with row and column broadcast buses, (b) hypermesh, (c) mesh with multiple global buses, and (d) mesh with separable row and column buses.

this rectangular mesh.[1] Similar results about the optimality of $N^{5/8} \times N^{3/8}$ meshes for semigroup computations were obtained by Peleg and Bar-Noy [16], who also provide a formal proof that every algorithm for semigroup computation on a rectangular mesh with buses takes at least $\Omega(N^{1/8})$ steps and extend the results to d-dimensional meshes.

Ragavendra [20] proposed the hypermesh (HMESH) architecture which consists of an $N^{1/2} \times N^{1/2}$ mesh with a hierarchy of broadcast buses in each row and each column such that there are K PE's on each bus. In the first level, there are $N^{1/2}/K$ buses in each row or column, with a group of K PE's connected to each bus. One PE from each group is connected to a second-level bus in a similar manner. This process is repeated until there is only one group of K PE's which are then connected by a broadcast bus (Fig. 1(b)). Using this architecture, semigroup, median row, and shortest distance computations can be performed in $O(\log N)$ time. Carlson [9] proposed a mesh modified with a hierarchy of L global buses (Fig. 1(c)). This mesh can compute all terms of a linear recurrence system (a problem shown to be equivalent to a prefix computation) in $O(N^{1/(2+L)})$ time. Both of these structures, like the pyramid, are difficult to implement in VLSI.

Finally, Maeba *et al.* [13] considered a mesh with separable row and column buses, i.e., row/column buses that can be sectioned through PE-controlled switches (Fig. 1(d)). Unlike the hierarchy of global or row/column buses, this architecture is easy to implement in VLSI. Using this architecture with switches inserted after every other processor on a row or column bus, time complexity of $O(\log N)$ for semigroup computation and $O(N^{1/8} \log^{3/2} N)$ for median selection problem is claimed. However, this result is of theoretical interest only since it is achieved with an unrealistically large number of switches on each bus. Even if practically realizable, such a large number of switches would add to the bus complexity and delay.

[1]Note added in proof: See *J. Parallel Distributed Computing*, vol. 15, pp. 79–84, 1992.

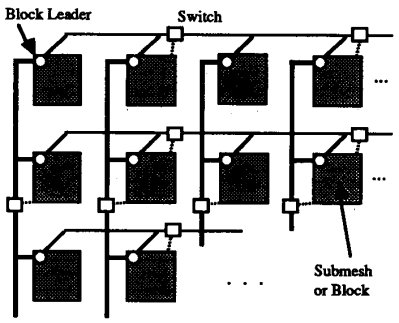

Fig. 2. The proposed scheme of mesh interconnection with separable row and column buses inserted between submeshes or PE blocks.

A recently proposed alternative to the use of buses, and one that is potentially more efficient for applications with high degrees of global communication, is Dally's "express channels" [11]. In the case of a two-dimensional mesh, express row and column channels simply connect the PE at (i, j) to the PE's at $(i \pm g, j)$ and $(i, j \pm h)$, for selected values of i and j, where g and h are design parameters.

II. Mesh with Fewer Separable Buses

We propose a modification to Maeba *et al.*'s architecture that can be implemented even more efficiently using VLSI technology. Like Maeba *et al.*'s scheme, our approach uses a standard mesh augmented with row and column broadcast buses which can be configured into local buses by PE-controlled switches. Unlike previous approaches, however, our scheme does not need all PE's to be connected to row and columns buses, but rather postulates one row (column) bus for every N^k rows (columns) of PE's where $k < 1/2$. This reduces the number of buses required, and thus the associated area cost, by a factor of N^k. As shown in Fig. 2, $N^k \times N^k$ blocks of PE's are interconnected using only local links as in a simple mesh. One PE in each block, designated as the block leader, is connected to the separable row and column buses to be described shortly (for examples, see Fig. 3).

This scheme allows the design of efficient parallel algorithms for problems with limited global communication. Examples include semigroup computation, prefix computation, and median computation. Algorithms proceed in time by first using the local links in each block and then using the structure of row/column buses, one hierarchy level at a time, until the buses connecting entire rows or columns are reached. Each level of the hierarchy corresponds to a particular pattern of open and closed switches in the row or column bus.

With regard to VLSI implementation, this structure has several advantages over previous approaches:

a) *Local links* are easier to build than global buses. This is because the proximity of the PE's makes it easy to provide a high-performance link interconnecting two neighboring PE's. Such a link can even be bit-serial because the distance is so small that high-bandwidth communication can still be achieved.

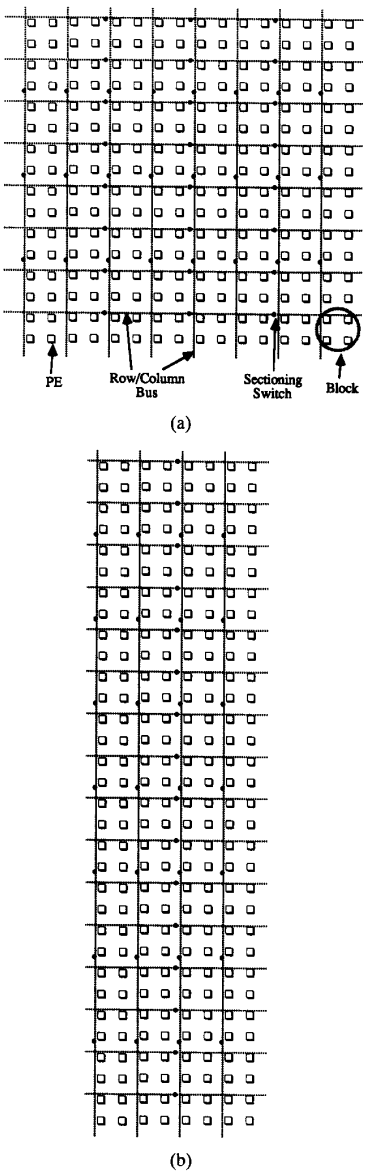

Fig. 3. Specific examples of the proposed mesh architecture with a reduced number of row/column buses: (a) square mesh with 256 PE's, and (b) optimal rectangular mesh of the same size. To avoid clutter, local mesh connections are not shown.

b) *Broadcast buses* are difficult to implement efficiently if they are to interconnect a large number of PE's over a long distance. Many authors have assumed a constant broadcast time, but in practice the broadcast time is proportional to log E, where E is the number of processors connected to the bus. With row and column buses for all PE's, the buses must have high performance to match the speed of local links.

c) *Scalability* in hardware is a concern too, because in building a large mesh composed of thousands of PE's, multiple VLSI chips have to be used. With row and column buses for all PE's, an unreasonable amount of interchip wiring will be required. Our approach overcomes this by using a scalable recursive method of mesh building, which will be explained in detail in Section IV.

The remainder of this paper is organized as follows. First, semigroup and prefix computation algorithms are given for our mesh (Section III). We find that for our basic two-level organization (one level corresponding to all the switches open and the other to all closed), a rectangular rather than square arrangement of PE's is the best form for these algorithms. Using this two-level sectioning of separable row and column buses, we achieve the same $O(N^{1/8})$ time complexity found by Chen *et al.* [10] for the mesh of Fig. 1(a), despite a significant reduction in the number of buses. The same result is achieved for prefix computation. Next, we generalize the method to hierarchical L-level sectioning of separable row and column buses achieving a time complexity of $O(N^{1/(3L+2)})$, which is better than the $O(N^{1/(L+2)})$ complexity found by Carlson [9] for a mesh with a hierarchy of L global buses. We then show that our method of building meshes is scalable by indicating how larger meshes (including certain square meshes) can be built from optimal-shaped rectangular submeshes. Finally, we discuss the VLSI implications of our designs.

III. Semigroup and Prefix Computations

Semigroup computations form an important class of computational problems. Examples include computing the sum, product, maximum, minimum, and Boolean parity over a set of values. A semigroup computation can be formally described by a pair $(\oplus, S)$, where $\oplus$ is an associative binary operator and $S = \{a_0, a_1, \ldots, a_{N-1}\}$ is a set of data items [10]. The problem is to compute $a_0 \oplus a_1 \oplus \ldots \oplus a_{N-1}$. Chen *et al.* [10] have proposed an $N^{5/8} \times N^{3/8}$ rectangular mesh with fully connected row and column buses which can perform a semigroup computation in $O(N^{1/8})$ time—down from $O(N^{1/6})$ resulting from a square mesh. Intuitively, the better performance of rectangular meshes compared to square ones results from the larger number of buses, and thus increased communication bandwidth, associated with the same number N of PE's. We will arrive at the same $O(N^{1/8})$ computation time with our simplified architecture.

A. The Basic Semigroup Algorithm

In this subsection, we describe the semigroup algorithm for a specific instance of our general mesh architecture, viz, an $N^{5/8} \times N^{3/8}$ mesh composed of $N^{1/8} \times N^{1/8}$ PE blocks. A *row group* is defined as $N^{1/8}$ blocks contiguous in a row while a *row band* consists of a group of $N^{1/8}$ consecutive *row groups* (Fig. 4). Two levels are assumed for the hierarchical sectioning of separable row and column buses. A row band is associated with an entire row bus (first hierarchy level) and a row group is associated with each one of the separable bus sections (second hierarchy level). Similarly, a *column group* consists of $N^{1/8}$ blocks contiguous in a column and a *column band* is formed by $N^{3/8}$ column groups. There are a total of $N^{1/4}$ column buses and $N^{1/2}$ row buses (refer to examples in Fig. 3). Initially there is one data item in each PE.

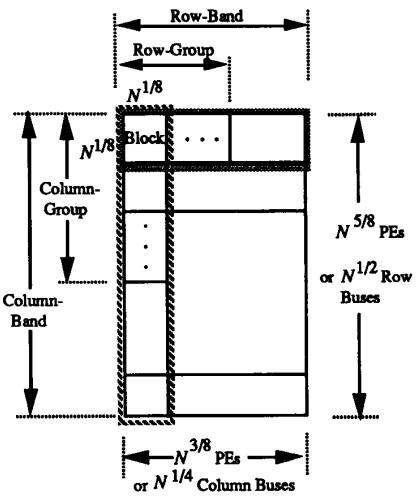

Fig. 4. Definition of terms for the semigroup algorithm.

Step 1 (Block Reduction): Perform the semigroup computation for each block, using local links. This step takes $O(N^{1/8})$ time. The result for each block is held in the upper left PE of the block, called the block leader. This step reduces the problem size from N to $N^{3/4}$.

Step 2a (Row-Group Reduction): Copy the partial result in each row group to the row-group leader which performs the semigroup computation as it receives the partial results. This is done by using the corresponding row bus sections and takes $O(N^{1/8})$ time. The problem size is now reduced to $N^{3/4}/N^{1/8} = N^{5/8}$.

Step 2b (Row-Band Reduction): Copy the partial results from row-group leaders to row-band leaders which perform the semigroup computation as they receive the partial results. This is done by using entire row buses and takes $O(N^{1/8})$ time. Now there are $N^{1/2}$ values in the leftmost column, one per row-band leader.

Step 3a (Column-Group Reduction]): Copy the partial results from the leftmost column groups to the leftmost column-group leaders which perform the semigroup computation as they receive the partial results. This is done by using column bus sections and takes $O(N^{1/8})$ time. The problem size is now $N^{3/8}$.

Step 3b (Replication of Values): Broadcast the partial result of each leftmost column-group leader to all PE's connected to a row bus, using $N^{3/8}$ of the $N^{1/2}$ row buses. This step takes constant time.

Step 3c (Column-to-Row Transposition): Partition the $N^{3/8}$ partial results into $N^{1/4}$ groups each having $N^{1/8}$ elements, so that each group can use a column bus. Copy the $N^{1/8}$ items in each group to the topmost PE (Row 0) which performs the semigroup computation. At the end of this $O(N^{1/8})$-time step, we have $N^{1/4}$ items in Row 0.

Steps 4a/b (Zeroth-Row Reduction): Steps 2a and 2b can be applied once again here to reduce the $N^{1/4}$ intermediate values to $N^{1/8}$ and then to $N^0 = 1$ final result in the upper leftmost PE. These two steps take $O(N^{1/8})$ time each.

An alternative approach to Step 4 is to perform the semigroup computation for the remaining $N^{1/4}$ values by using the simulation tree technique with $N^{1/4} \times N^{1/4}$ row and column buses, as described by Ragavendra [20] and Chen *et al.* [10]. This step would then take $O(\log N)$ time. However, since this does not change the overall time complexity of $O(N^{1/8})$ for our algorithm, we have chosen to use the conceptually simpler approach. One may note that Steps 2b and 3a can be interchanged without affecting the result. In this way the computation proceeds first by using local links within blocks, then using bus sections in row and column groups, and finally using entire row and column buses.

Compared to the mesh proposed by Chen *et al.* [10] which uses a total of $N^{5/8}+N^{3/8} = O(N^{5/8})$ row/column buses, our approach needs $N^{1/2} + N^{1/4} = O(N^{1/2})$ buses. Each of our column buses is connected to $N^{1/2}$ PE's compared to $N^{5/8}$ in the above scheme. The number of bus switches needed in our approach is $N^{1/8} \times N^{1/2} + N^{3/8} \times N^{1/4} = O(N^{5/8})$, but since switches are easier to build than buses, our mesh is simpler overall.

B. Prefix Computations

Semigroup computation can be viewed as a special case of prefix computation defined as the simultaneous computation of $S_i = a_0 \oplus a_1 \oplus \ldots \oplus a_i$ for all i in the range $0 \leq i \leq N-1$. Let $S_{i,j} = a_i \oplus a_{i+1} \oplus \ldots \oplus a_j, i \leq j$ be the local prefix from a_i to a_j. Clearly, $S_j = S_{0,j} = S_{0,i_1} \oplus S_{i_1+1,i_2} \oplus \ldots \oplus S_{i_M+1,j}$ where $0 < i_1 < i_2 < \ldots < i_M < j$. The algorithm for prefix computation is very similar to the one described in Section III-A, but we proceed from left to right instead of right to left. Also, intermediate results have to be kept in each node, because the algorithm has two phases: from local to global and from global to local.

To write the required algorithm, we need to specify an ordering for the PE's and their corresponding values. We assume that within a block, PE's are numbered in row-major order, starting from 0. PE blocks are also numbered in row-major order, starting from 0. The input data items $a_0, a_1, \ldots, a_{N-1}$ are stored in the mesh in increasing order of block numbers and in increasing order of PE numbers when they are within the same block. The algorithm is to produce S_i in the PE initially holding a_i.

Step 1 (Local Prefixes for each Block): Compute $S_{i_b,j}$ in parallel for each block, where a_j is held in the block and i_b is the minimum k such that a_k is held in the block.

Step 2 (Row Reduction): Generate row-group prefixes by broadcasting data using row bus sections. Then generate row-band prefixes by broadcasting data using entire row buses. At the end of this step the rightmost column contains the row-band prefixes. Note that we proceed from left to right.

Step 3 (Column Reduction): Generate the column-group prefixes by broadcasting data on column bus sections (actually we are using only the rightmost column bus). Broadcast the resulting $N^{3/8}$ prefixes within rows and do column-to-row transposition as in Steps 3b and 3c of the semigroup algorithm.

Step 4 (Zeroth-Row Reduction): Do a prefix computation for the items in Row 0.

Steps 5–8 (Backward Phase): These constitute Phase 2 of the algorithm. Instead of going from local to global, we go from global to local, keeping in mind that intermediate results have been stored in the PE's. Steps 1 through 4 are performed backwards to obtain the prefix in each PE.

Since any step in this algorithm takes $O(N^{1/8})$ time, the overall time complexity is $O(N^{1/8})$, as expected.

C. Extension to an Arbitrary Size Mesh

In this subsection, we will extend the algorithm of Section III-A (devised for a special case of our proposed architecture having $N^{5/8}$ rows, $N^{3/8}$ columns, $N^{1/8} \times N^{1/8}$ blocks, and two-level sectioning of buses) to an arbitrary size mesh with an arbitrary number of hierarchical sectioning levels for both row and column buses. The extension applies to the prefix algorithm as well. The parameters are defined as follows:

$N^r \times N^c$	Rectangular mesh with N^r rows and N^c columns $(r + c = 1, r \geq c)$.
$N^k \times N^k$	Size of a *block* of PEs connected only by local links $(k < 1/2)$
R	Number of hierarchical sectioning levels for row buses.
C	Number of hierarchical sectioning levels for column buses.

With the above notation, there are $N^r/N^k = N^{r-k}$ row and $N^c/N^k = N^{c-k}$ column buses.

Step 1: Perform the semigroup computation for each block, using local links. This step takes $O(N^k)$ time. The result of each block is held in the block leader. After this step, the problem size is reduced to $N/N^{2k} = N^{1-2k}$.

Step 2: The purpose of this step in the original algorithm was to reduce the problem size using the separable row buses. For two-level sectioning, two reductions are needed, each reducing the problem size by N^k. In general, for R-level sectioning, R reduction steps are needed, for a total reduction of N^{Rk}. At the end of this step, the problem size is reduced to $N^{1-(R+2)k}$, and the results are in the leftmost column. The time complexity of the reduction is $O(N^k)$. From here, we deduce that the number of columns in the mesh is $N^c = N^{(R+1)k}$, and the number of column buses is $N^{c-k} = N^{Rk}$.

Step 3a: Perform reduction on the leftmost column, reducing each time by N^k. With C-level hierarchical sectioning, $C - 1$ reductions are needed, for a total reduction of $N^{(C-1)k}$. At the end of this step, the problem size has been reduced to $N^{1-(R+C+1)k}$.

Step 3b: Broadcast the intermediate results of each leftmost column-group leader to all PE's connected to a row bus. This step takes a constant time.

Step 3c: Partition the $N^{1-(R+C+1)k}$ intermediate results into N^{Rk} groups with $N^{1-(2R+C+1)k}$ elements in each, so that each group can use one column bus. Copy the values of each group to the topmost PE (Row 0) which performs the semigroup computation. This step takes $O(N^{1-(2R+C+1)k})$ time. In order to minimize the time complexity, we require that $k = 1 - (2R + C + 1)k$, yielding $k = 1/(2R + C + 2)$. With this value for k, the current step takes $O(N^k)$ time, and at the end of the step, there are N^{Rk} results in row 0 of the mesh.

Step 4: Step 2 can be applied once again here to reduce the intermediate values in the topmost row by N^{Rk}. At the end, the final result can be found in the upper leftmost PE. This step takes $O(N^k)$ time.

From this discussion, we conclude that the optimal time complexity is $O(N^{1/(2R+C+2)})$ and that the optimal mesh has $N^r = N^{(R+C+1)k}$ rows and $N^c = N^{(R+1)k}$ columns and it is thus of size $N^{(R+C+1)/(2R+C+2)} \times N^{(R+1)/(2R+C+2)}$. We conclude that the optimal mesh is always rectangular because $R + C + 1 \neq R + 1$.

Let us examine some special cases. For $R = C = 1$, we obtain an $N^{3/5} \times N^{2/5}$ mesh having no bus switches and with a running time of $O(N^{1/5})$. This is a new and significant result in itself. By choosing $R = C = 2$ as in our previous algorithm, we end up with an $N^{5/8} \times N^{3/8}$ mesh with a running time of $O(N^{1/8})$, as expected. For $R = C = L$, we end up with a $N^{(2L+1)/(3L+2)} \times N^{(L+1)/(3L+2)}$ mesh, with a running time of $O(N^{1/(3L+2)})$. The number of buses needed is $N^{c-k} + N^{r-k} = O(N^{r-k})$, which in our case is $O(N^{2L/(3L+2)})$. Compare this result with those reported by Carlson [9] using a hierarchy of L global buses connecting all PE's, achieving a running time of $O(N^{1/(L+2)})$.

With L-level hierarchical sectioning of buses, the number of switches per row bus and column bus are approximately $N^{(L-1)/(3L+2)}$ and $N^{(2L-1)/(3L+2)}$, respectively. These expressions are obtained by dividing the number of PE's in a row (column) by $N^{2/(3L+2)}$ to account for the fact that one out of every $N^{1/(3L+2)}$ PE's is a block leader and a switch is inserted after every $N^{1/(3L+2)}$ block leaders. The total number of switches is derived by multiplying the number of row (column) buses by the number of switches per row (column) bus; viz $N^{2L/(3L+2)} \times N^{(L-1)/(3L+2)} + N^{L/(3L+2)} \times N^{(2L-1)/(3L+2)} = O(N^{(3L-1)/(3L+2)})$.

The running time with L-level hierarchical sectioning is actually $O(LN^{1/(3L+2)})$, but in the above discussion we ignored L since it is limited to be a small constant in practice. From a theoretical perspective, it is interesting to note that if L approaches $\log N$, then the running time becomes $O((\log N)(N^{1/(3 \log N+2)})) = O(\log N)$ since $N^{1/\log N} = e$. Thus, logarithmic time can be achieved asymptotically when L becomes large.

We can also perform a simple optimization, by assigning more than one data value to each processor, as pointed out by Stout [21]. We can assign N/P data items to each of P processors in a smaller mesh. The running time on each processor is linear so it takes $O(N/P)$ time to perform the semigroup computation. Once this is done, the mesh is used with a running time of $O(P^{1/(3L+2)})$. The combined execution time is $O(\max(N/P, P^{1/(3L+2)}))$. To minimize the time complexity, we set $N/P = O(P^{1/(3L+2)})$, resulting in $P = O(N^{(3L+2)/(3L+3)})$ which yields a complexity of $O(P^{1/(3L+2)}) = O(N^{1/(3L+3)})$. Setting $L = 2$, this expression reduces to $O(N^{1/9})$, the result found by Chen *et al.* [10].

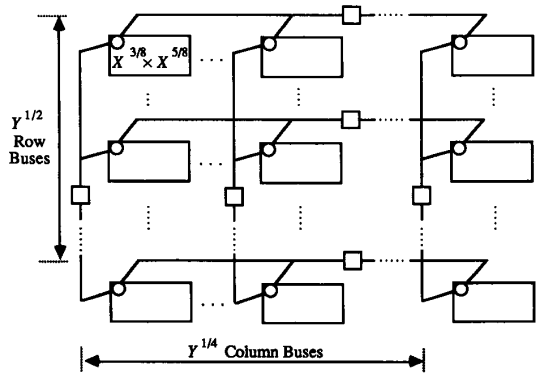

Fig. 5. Building a larger mesh from smaller rectangular meshes.

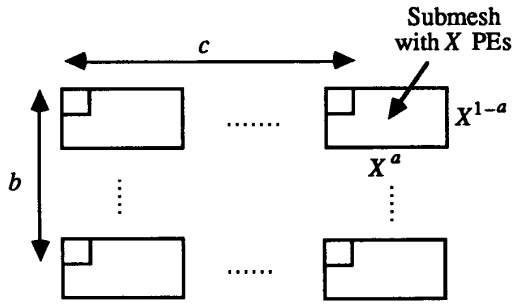

Fig. 6. Notation for a general iteration of the recursive procedure.

IV. Building Higher Order Meshes

In this section, we present a recursive procedure for combining small meshes to obtain larger ones that can perform semigroup computations with a lower complexity. We will also show that our method can be used to build square meshes from rectangular meshes. The process is first illustrated using the $N^{5/8} \times N^{3/8}$ mesh of Section II-A and then extended to arbitrary meshes.

A. Two-Level Mesh

We showed in Section II-A that the optimal mesh with two-level sectioning of buses has $N^{5/8} \times N^{3/8}$ PE's with $N^{1/2}$ row buses and $N^{1/4}$ column buses. Fig. 5 illustrates how we can build a higher order mesh using this rectangular mesh, connecting one PE from each rectangular mesh to higher level row and column buses.

We know that the optimal mesh with two level sectioning has $Y^{1/2}$ row buses and $Y^{1/4}$ column buses, for some Y. We use rectangular meshes of size $X^{5/8} \times X^{3/8}$, for some X, each giving a time complexity of $O(X^{1/8})$. Since the hierarchy has two levels, we conclude that $Y^{1/4} = (X^{1/8})^2$, or $Y = X$. The total number of PE's is $X^{5/8} \times X^{1/4} \times X^{3/8} \times X^{1/2} = N$, yielding $X = N^{4/7}$. The time complexity is $O(X^{1/8}) = O(N^{1/14})$, and the dimensions of the larger mesh are $N^{1/2} \times N^{1/2}$, so we end up with a square mesh! Note that this result should be the same as for four-level sectioning: $L = 4$ and $O(N^{1/(3L+2)}) = O(N^{1/14})$, which is consistent with our intuitive expectation.

B. Arbitrary Mesh

We will give the general proof for any arbitrary higher level mesh. As shown in Fig. 6, a higher order mesh is built from $X^a \times X^{1-a}$ meshes for some X, each guaranteeing a running time of $O(X^k)(k < 1/2)$. In our recursive procedure for building a larger mesh, we start first with a square mesh using only local links, so initially $a = k = 1/2$. We postulate b row buses and c column buses.

Each recursive iteration of the algorithm is as follows.

Step 1: Perform the semigroup computation for each component submesh. This step takes $O(X^k)$ time.

Step 2: The purpose of this step in the original algorithm was to reduce the problem size using the separable row buses. For two-level sectioning, two reductions are needed, each reducing the problem size by X^k. We thus conclude that the number of column buses is $c = X^{2k}$. At the end of this step, the leftmost column contains b intermediate results. Define $b = X^d$ for some d.

Step 3a: Perform reduction on the leftmost column by X^k. At the end of this step, the number of values in the leftmost column is X^{d-k}

Step 3b: Broadcast the partial results of each leftmost column-group leader to all PE's connected to a row bus. This step takes a constant time.

Step 3c: Partition the X^{d-k} partial results into X^{2k} groups with X^{d-3k} elements in each, so that each group can use a column bus. Copy the values of each group to the topmost PE (Row 0) which performs the semigroup computation. To minimize the time complexity, we set $k = d - 3k$ or $d = 4k$. Thus the number of row buses is $b = X^{4k}$.

Step 4: Step 2 can be applied once again here to reduce the intermediate values in the topmost row. At the end, the final result can be found in the upper leftmost PE.

If the higher order mesh is to consist of N PE's overall, we must have $X^a \times c \times X^{1-a} \times b = X^a \times X^{2k} \times X^{1-a} \times X^{4k} = N$, giving $X = N^{1/(6k+1)}$. Since the running time is $O(X^k)$, this higher order mesh gives us a running time of $O(N^{k/(6k+1)})$. From this we can derive the following recurrence relations where i is the iteration number:

$$k_{i+1} = k_i/(6k_i + 1), \quad \text{initially } k_0 = 1/2$$
$$a_{i+1} = (2k_i + a_i)/(6k_i + 1), \quad \text{initially } a_0 = 1/2.$$

The solution of the recurrence relation for k is $k_i = 1/(6i+2)$, giving a running time of $O(N^{1/(6i+2)})$. Intuitively, we can see that this result should correspond to a $2i$-level hierarchical sectioning scheme. Setting $L = 2i$ in the formula $O(N^{1/(3L+2)})$ yields $O(N^{1/(6i+2)})$ as expected. The solution of the recurrence relation for a is $a_i = (2i+1)/(6i+2)$ which again is consistent with the results presented in Section II-C.

By using this scheme as discussed, we always obtain a rectangular mesh. However, if we exchange the roles of rows and columns at each iteration step, we can get a square mesh. To prove this, we set $a_{i+1} = (2k_i + 1 - a_i)/(6k_i + 1)$; i.e., we replace a_i with $1 - a_i$ in the recurrence for a_{i+1}. It is then

easily verified by manipulating the recurrence expression for a_{i+2} that

$$a_{i+2} = (6k_i + a_i)/(12k_i + 1) = 1/2 + (a_i - 1/2)/(12k_i + 1).$$

Since the initial condition is $a_0 = 1/2$, we conclude that $a_{2j} = 1/2$ for all j; i.e., the mesh after iterations $0, 2, 4, 6, \ldots$ is square.

It is easy to extend this result to meshes that are built with L-level hierarchical sectioning at each iteration. In this case, $c = X^{Lk}$ (the number of column buses) and $b = X^{2Lk}$ (the number of row buses). It is then easily shown that for the ith iteration:

$$k_{i+1} = k_i/(3Lk_i + 1), \qquad \text{initially } k_0 = 1/2$$
$$a_{i+1} = (Lk_i + a_i)/(3Lk_i + 1), \qquad \text{initially } a_0 = 1/2.$$

The solution is $k_i = 1/(3iL + 2)$. In this general case, it can be similarly shown that the mesh after iterations $0, 2, 4, 6, \ldots$ can be square.

C. VLSI Implications

In the construction of a mesh composed of thousands of processors, many compromises have to be made. Given the state of the art in VLSI, it is impossible to embed the entire mesh into a single VLSI chip. Thus, several chips need to be built and interconnected. In building such a mesh, regularity of construction is important since it allows us to use several identical chips to build a large system. Also, the amount of interconnections, especially off-chip ones, is very important.

Our mesh design provides clear advantages in these regards. In a simple mesh, all PE's have four neighbors, except for the PE's on the first and last rows/columns. In our mesh, this does not have to be the case. Thus, it is possible to embed part of the mesh into a single VLSI chip. Then, multiple chips are connected via buses without having full connections as in the original mesh. Even though switches are easy to build, it may not be advisable to have many hierarchical sectioning levels by using more and more switches because of the delay that such switches introduce. Instead, the recursive procedure outlined in Section III-B can be employed to construct a larger mesh. Of course, this implies that some PE's have a higher degree (the leader PE's), but a compromise can be achieved between the number of switches and the number of levels in the iterative procedure.

V. Conclusion

In this paper, we have shown how semigroup and prefix computations can be performed with optimal time complexity on mesh-connected computers with separable row and column buses without the provision of buses for every row and every column. We have proven that with two-level sectioning of buses, square meshes are not optimal and presented a recursive procedure for building higher order meshes that can yield an optimal square mesh using more hierarchy levels. Our time-complexity results were shown to correspond to those previously published when certain parameters of our design are fixed at special values.

A possible next step is to develop algorithms for other computations using our mesh. For many problems with limited global communication, such as image processing, a significant improvement in time can be expected. Our results can also be extended to d-dimensional meshes, even though a mesh with $d > 2$ is of lesser practical importance.

References

[1] A. Aggarwal, "Optimal bounds for finding maximum on array of processors with k global buses," *IEEE Trans. Comput.*, vol. C-35, no. 1, pp. 62–64, Jan. 1986.

[2] M. J. Atallah and S. R. Kosaraju, "Graph problems on a mesh-connected processor array," *J. Ass. Comput. Mach.*, vol. 31, no. 3, pp. 644–667, July 1984.

[3] M. J. Atallah, "On multidimensional arrays of processors," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1306–1309, Oct. 1988.

[4] G. Bilardi and F. P. Preparata, "Size-time complexity of Boolean networks for prefix computation," *J. Ass. Comput. Mach.*, vol. 36, no. 2, pp. 362–382, Apr. 1989.

[5] S. H. Bokhari, "Finding maximum on an array processor with a global bus," *IEEE Trans. Comput.*, vol. C-33, no. 2, pp. 133–141, Feb. 1984.

[6] P. R. Cappello, "A mesh automaton for solving dense linear systems," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 418–425.

[7] D. A. Carlson, "Performing tree and prefix computations on modified mesh-connected parallel computers," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 715–718.

[8] D. A. Carlson, "Modified mesh-connected parallel computers," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1315–1321, Oct. 1988.

[9] D. A. Carlson, "Solving linear recurrence systems on mesh-connected computers with multiple global buses," *J. Parallel Distributed Computing,* vol. 8, pp. 89–95, 1990.

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

[11] W. J. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 40, no. 9, pp. 1016–1023, Sept. 1991.

[12] C. H. Lung, "Parallel algorithms on a mesh with multiple broadcasting," in *Proc. Symp. Appl. Computing,* Fayeteville, AR, 1990, pp. 92–95.

[13] T. Maeba, S. Tatsumi, and M. Sugaya, "Algorithms for finding maximum and selecting median on a processor array with separable global buses," *Electron. Commun. Japan,* pt. 3, vol. 73, no. 6, pp. 39–47, 1990.

[14] R. Miller and Q. F. Stout, "Geometric algorithms for digitized pictures on a mesh-connected computer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 2, pp. 216–227, Mar. 1985.

[15] R. F. Hobson and M. Rostam-Kafhesh, "A mesh-like array processor with fully connected rows and columns," *IEEE Pacific Rim Conf. Commun., Comput., Signal Processing,* June 1989, pp. 152–155.

[16] D. Peleg and A. Bar-Noy, "Square meshes are not always optimal," *IEEE Trans. Comput.*, vol. 40, no. 2, pp. 196–204, Feb. 1991.

[17] N. Pippenger, "The complexity of computations by networks," *IBM J. Res. Develop.*, vol. 31, no. 2, pp. 235–243, Mar. 1987.

[18] V. K. Prasanna-Kumar and D. I. Reisis, "Image computations on meshes with multiple broadcast," *IEEE Trans. Pattern Analy. Mach. Intell.*, vol. 11, pp. 1194–1201, Nov. 1989.

[19] V. K. Prasanna-Kumar and C. S. Raghavendra, "Array processor with multiple broadcasting," *J. Parallel Distributed Computing,* vol. 2, pp. 173–190, 1987.

[20] C. S. Raghavendra, "HMESH: A VLSI architecture for parallel processing," in *Proc. 2nd Conf. Algorithms and Hardware for Parallel Processing,* 1986, pp. 76–83.

[21] Q. F., Stout, "Mesh-connected computers with broadcasting," *IEEE Trans. Comput.*, vol. C-32, no. 9, pp. 826–830, Sept. 1983.

[22] Q. F. Stout and R. Miller, "Varying diameter and problem size in mesh-connected computers," in *Proc. Int. Conf. Parallel Processing,* Aug. 1985, pp. 697–699.

[23] B. F. Wang and G. H. Chen, "Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. Parallel Distributed Syst.*, vol. 1, no. 4, pp. 500–507, Oct. 1990.

[24] M. Willebeek-LeMair and A. P. Reeves, "Solving nonuniform problems on SIMD computers: Case study on region growing," *J. Parallel Distributed Computing,* vol. 8, pp. 135–149, 1990.

Mauricio J. Serrano (S'89) was born in Cali, Colombia, on September 10, 1958. He received the B.S. degree in electrical engineering from the Universidad Javeriana, Bogota, Colombia, in 1982, and the M.S. degree in computer engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1986. He is currently pursuing the Ph.D. degree in computer engineering at the University of California, Santa Barbara. His current research interests include parallel and distributed processing, computer architecture, performance modeling, and simulation.

Behrooz Parhami (S'70–M'73–SM'78) received the Ph.D. degree in computer science from the University of California, Los Angeles, in 1973.

From 1974 to 1988 he was with Sharif University of Technology in Tehran, Iran, where he was involved in the areas of educational planning, curriculum development, standardization efforts, technology transfer, and various editorial responsibilities. Presently, he is Professor in the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His current research deals with parallel architectures and algorithms, high-speed computer arithmetic, and dependable (fault-tolerant) computing. His technical publications include over 100 papers in journals and international conferences, two Farsi-language textbooks, and an English/Farsi glossary of computing terms.

Dr. Parhami is a member of the Association for Computing Machinery and the British Computer Society and a Distinguished Member of the Informatics Society of Iran for which he served as a founding member, Editor-in-Chief, and President during 1979-84. He also served as Chairman of IEEE Iran Section (1977-1986) and received the IEEE Centennial Medal in 1984.