Optimal Aspect Ratio and Number of Separable Row/Column Buses for Mesh-Connected Parallel Computers

Mauricio J. Serrano and *Behrooz Parhami*

Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, USA

ABSTRACT

A two-dimensional mesh of PEs with separable row and column buses (broadcast mechanisms that can be logically divided into a number of local row/column buses through the use of PE-controlled switches) has been shown to be quite effective for semigroup, prefix, and a wide class of other parallel computations. We show how semigroup and prefix computations can be performed with the same asymptotic time complexity on meshes having separable buses for a subset of rows and columns. We find that with our basic arrangement, square grids are not optimal but that a hierarchical method of synthesizing large meshes builds optimal square meshes from rectangular submeshes. The time-complexity results are shown to correspond to those previously published when certain parameters of our design are fixed at special values.

I. BACKGROUND

A two-dimensional mesh-connected computer consists of N processors or processing elements (PEs) arranged in a grid, where horizontally and vertically adjacent processors are connected via local communication links. The main disadvantage of a standard mesh is that the number of steps required by any non-trivial algorithm is lower-bounded by its diameter, $\Omega(N^{1/2})$. Consequently, many authors have proposed meshes augmented with broadcast buses as global communication mechanisms. Due to lack of space, the following discussion contains reference citations only for the most essential and directly relevant work.

Bokhari showed that by using a bus connected to all PEs, the maximum of N numbers, assigned one per PE, can be found in $O(N^{1/3})$ time. Stout showed that using a global bus, the time to perform semigroup computations can be reduced to $O(N^{1/3})$ and that problems with higher degrees of global communication, exemplified by sorting, have a lower bound of $\Omega(N^{1/2})$, with or without broadcasting. He also showed that with broadcasting, 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. Hence the need for multiple broadcasting. Prasanna-Kumar and Raghavendra [5] considered a mesh of $N^{1/2}\times N^{1/2}$ PEs with broadcast buses in each row and column 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 takes $O(N^{1/6}\log^{2/3}N)$, and convex polygon and nearest neighbor take $O(N^{1/6})$ time.

Chen et al [2] showed that square versions of meshes with row/column buses are not optimal for semigroup and prefix computations and that by using a rectangular $N^{5/8}\times N^{3/8}$ mesh, time complexity of these computations reduces 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 this rectangular mesh. Similar results were obtained by Peleg and Bar-Noy [4] who also provide a formal proof that semigroup computation on a rectangular mesh with buses takes at least $\Omega(N^{1/8})$ steps and extend the results to d-dimensional meshes.

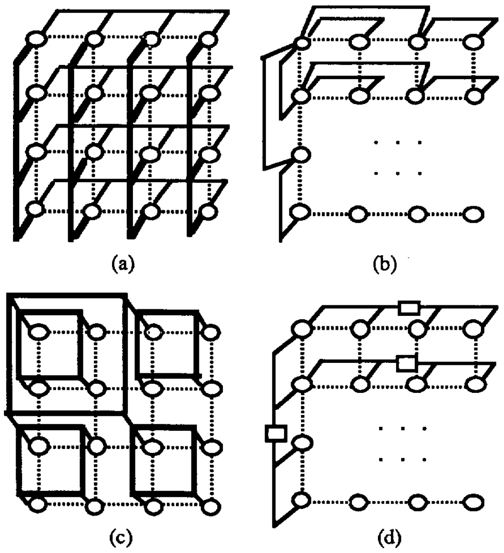

Figure 1. Different broadcasting schemes for a mesh: (a) Row & column buses, (b) Hypermesh, (c) Multiple global buses, and (d) Separable row/column buses.

Raghavendra proposed the hypermesh 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 PEs on each bus. In the first level, there are $N^{1/2}/K$ buses in each row or column, with a group of K PEs 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 PEs on the final bus. With this architecture, semigroup, median row,

0-8186-2672-0/92 $03.00 © 1992 IEEE

and shortest distance computations need $O(\log N)$ time. Carlson [1] proposed a mesh modified with a hierarchy of L global buses. This mesh can compute all terms of a linear recurrence system (a problem which is equivalent to a prefix computation) in $O(N^{1/(2+L)})$ time. Like pyramid, these structures are both difficult to implement in VLSI.

Finally, Maeba et al [3] consider a mesh with separable row and column buses; i.e., row/column buses that can be sectioned through PE-controlled switches. Unlike the hierarchy of global or row/column buses, this architecture is easy to implement in VLSI. Using this scheme with switches inserted after every other processor on each 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, for large N, 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.

II. FEWER SEPARABLE BUSES

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

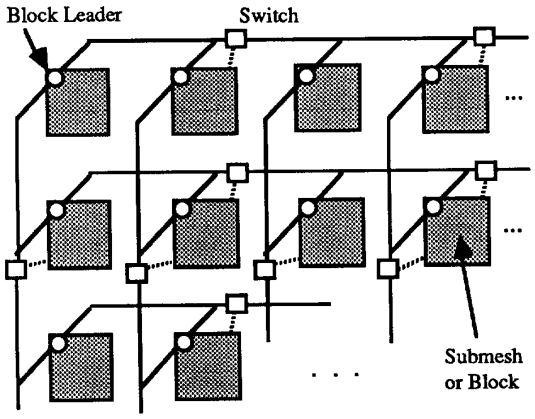

Figure 2. Proposed scheme of mesh interconnection with separable row/column buses inserted between submeshes or PE Blocks.

Our scheme allows the design of highly efficient parallel algorithms for problems with limited global communication. Examples include semigroup computations, prefix computations, and median finding. Algorithms proceed in time by first using the local links in each block and then using the structure of broadcast buses, with growing sections, until full row or column broadcasts are reached.

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

a) *Local links* are easier to build than broadcast buses.

b) *Broadcast buses* are easier to implement efficiently if they are to interconnect a relatively small set of PEs.

c) *Scalability* is improved when the number of buses is reduced (due to fewer inter-chip connections required).

The remainder of this paper is organized as follows. In Section III, we provide semigroup and prefix computation algorithms for our mesh. We find that for our basic organization, a rectangular rather than square arrangement of PEs is the best form for these algorithms. With two-level sectioning of separable row and column buses, we achieve the same $O(N^{1/8})$ time complexity found by Chen et al [2], despite a significant reduction in the number of buses. We then generalize the method to a hierarchy of L 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 of Carlson [1] for a mesh with a hierarchy of L global buses. In Section IV, we show that our architecture is scalable by indicating how meshes of higher degrees (including certain square meshes) can be built from optimal-shaped rectangular meshes.

III. SEMIGROUP AND PREFIX COMPUTATIONS

A semigroup computation can be formally defined 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. The problem is to compute $a_0 \oplus a_1 \oplus \ldots \oplus a_{N-1}$. Chen et al [2] 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 PEs. 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 will show 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. Two levels are assumed for the sectioning of separable row and column buses. A *row-group* is defined as $N^{1/8}$ horizontally contiguous blocks while a *row-band* consists of $N^{1/8}$ horizontally contiguous *row groups*. A row-band is associated with one separable 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 (Figure 3). Initially each PE holds a data item.

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 intermediate results in each row-group to the row-group leader which performs the semigroup computation. 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 intermediate results from row-group leaders to row-band leaders which perform the semigroup computation. This is done by using entire row buses and takes $O(N^{1/8})$ time. Now we have $N^{1/2}$ values in the leftmost column, one per row-band leader.

Step 3a [Column-Group Reduction]: Copy intermediate results from the leftmost column-groups to the leftmost column-group leaders which perform the semigroup computation. 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 intermediate results of each leftmost column-group leader to all PEs 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 $N^{3/8}$ intermediate 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.

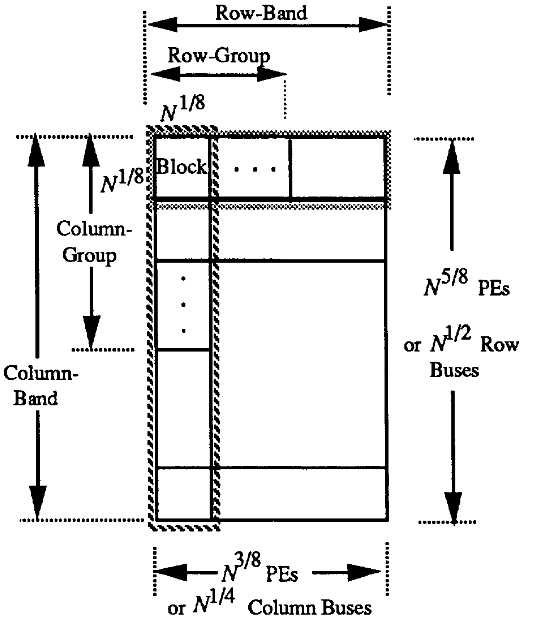
Row-Band
Row-Group
$N^{1/8}$
$N^{1/8}$
Block
Column-Group
Column-Band
$N^{5/8}$ PEs or $N^{1/2}$ Row Buses
$N^{3/8}$ PEs or $N^{1/4}$ Column Buses

Figure 3. Terminology for the semigroup algorithm.

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 (see, e.g., [2]). 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 a conceptually simpler approach.

Compared to the mesh proposed by Chen et al [2] which uses a total of $N^{5/8} + N^{3/8} = O(N^{5/8})$ buses, our approach needs $N^{1/2} + N^{1/4} = O(N^{1/2})$ buses. 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 computations can be viewed as special cases 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 $0 \le i \le N-1$. Let $S_{i,j} = a_i \oplus a_{i+1} \oplus \ldots \oplus a_j$, $i \le j$ 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 Subsection 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: local-to-global and global-to-local.

To write the required algorithm, we need to specify an ordering for the PEs and their corresponding values. We assume that within a block, PEs 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 an increasing order of PE numbers when they are within the same block. The algorithm is to generate 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 using 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 PEs. 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 section, we will extend the semigroup computation algorithm of Subsection A to an arbitrary size mesh with an arbitrary number of sectioning levels for both row and column buses. The extension applies to prefix algorithm as well. The parameters are defined as follows:

$N^r \times N^c$	N^r-row, N^c-column mesh ($r + c = 1, r \geq c$)
$N^k \times N^k$	PE *block* with only local links ($k < 1/2$)
R	Sectioning levels for row buses
C	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 row buses. For two-level sectioning, two reductions are needed, each reducing the problem size by N^k. In general, for an R-level hierarchy, 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 $O(N^k)$. 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. For C-level sectioning, $C - 1$ reductions are needed, for a total reduction of $N^{(C-1)k}$. The problem size has now been reduced to $N^{1-(R+C+1)k}$.

Step 3b: Broadcast the intermediate results of each leftmost column-group leader to all PEs 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 a column bus. Copy the group values 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 $k = 1 - (2R + C + 1)k$ or $k = 1/(2R + C + 2)$. With this value for k, this 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.

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 $N^{(R+C+1)/(2R+C+2)} \times N^{(R+1)/(2R+C+2)}$. Since $R + C + 1 > R + 1$, the optimal mesh is always rectangular.

Let us examine some special cases. For $R = C = 1$, we obtain an $N^{3/5} \times N^{2/5}$ mesh with no bus switches and with 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 have 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 [1] using a hierarchy of L global buses connecting all PEs, achieving a running time of $O(N^{1/(L+2)})$.

With an L-level hierarchy 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 are obtained by dividing the number of PEs in a row (column) by $N^{2/(3L+2)}$ to account for the fact that one out of every $N^{1/(3L+2)}$ PEs 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 sectioning is actually $O(L.N^{1/(3L+2)})$, but in the above discussion we ignored L since in practice it is a small constant. From a theoretical perspective, it is interesting to note that if L approaches $\log N$, 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. 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 $O(\max(N/P, P^{1/(3L+2)}))$. In order to minimize the time complexity, we require $N/P = O(P^{1/(3L+2)})$, or $P = O(N^{(3L+2)/(3L+3)})$ which gives us a complexity of $O(P^{1/(3L+2)}) = O(N^{1/(3L+3)})$. If we choose $L = 2$, this expression reduces to $O(N^{1/9})$, the result found by Chen et al [2].

IV. BUILDING LARGER 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 build square meshes from rectangular submeshes. The process is first illustrated using the $N^{5/8} \times N^{3/8}$ mesh of Subsection II.A and then extended to arbitrary meshes.

A. Two-Level Mesh

We showed in Subsection II.A that the optimal mesh with two-level sectioning has $N^{5/8} \times N^{3/8}$ PEs with $N^{1/2}$ row buses and $N^{1/4}$ column buses. We can build a larger mesh using this rectangular mesh by connecting one PE from each submesh 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 sectioning has two levels, we conclude that $Y^{1/4} = (X^{1/8})^2$, or $Y = X$. The total number of PEs 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! Interestingly this result is the same as for four-level sectioning: $L = 4$ and $O(N^{1/(3L+2)}) = O(N^{1/14})$.

B. Arbitrary Mesh

We now give the general proof for an arbitrary higher-level mesh built from $X^a \times X^{1-a}$ submeshes 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 mesh. 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 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 intermediate results of each leftmost column-group leader to all PEs connected to a row bus. This step takes a constant time.

Step 3c: Partition the X^{d-k} intermediate 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 there are $b = X^{4k}$ row buses.

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-level mesh is to have N PEs, we must have $X^a \times c \times X^{1-a} \times b = N$ or $X^a \times X^{2k} \times X^{1-a} \times X^{4k} = N$, giving $X = N^{1/(6k+1)}$. Since the running time is $O(X^k)$, this higher-level mesh gives us a running time of $O(N^{k/(6k+1)})$. From this we can derive the following recurrence relations:

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

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

By using this scheme directly, we always obtain a rectangular mesh. However, if we exchange the roles of rows and columns at each iteration step, we end up with a square mesh. To show 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 have $a_{2j} = 1/2$ for all j. Thus the mesh after iterations 0, 2, 4, 6, . . . is square.

It is easy to extend this result to meshes that are built with L-level sectioning at each iteration. In this case, $c = X^{Lk}$ (the number of column buses) and $b = X^{2Lk}$ (the number of row buses). Thus 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 $k_i = 1/(3Li + 2)$. As before, it can be shown that after iterations 0, 2, 4, 6, . . . the mesh is square in this general case.

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 each containing a few hundred PEs 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 density of interconnections, especially off-chip ones, is important.

Our mesh design provides clear advantages in these regards. In a simple mesh, all PEs have four neighbors, except for the PEs on the first and last rows/columns. In our mesh, that does not happen. 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 sectioning levels by using more and more switches because of the delay that such switches introduce. Instead, the recursive procedure outlined in Subsection III.B can be employed to construct a larger mesh. Of course, this implies that some PEs have a higher degree (the leaders), but a compromise can be achieved between the number of switches and the number of levels in the iterative procedure.

V. CONCLUSION

We have shown how semigroup and prefix computations can be performed with optimal time complexity on processor meshes with separable row and column buses without the provision of buses for every row and every column. We showed that with this new architecture, square meshes are not optimal and presented a recursive procedure for building higher order meshes that can yield an optimal square mesh. 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 not of practical importance.

REFERENCES

[1] D A Carlson, "Solving linear recurrence systems on mesh-connected computers with multiple global buses", *J. Parallel & Distrib. Comput.*, vol 8, pp 89-95, 1990.

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

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

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

[5] V K Prasanna-Kumar & C S Raghavendra, "Array processor with multiple broadcasting", *J. Parallel & Distrib. Comput.*, vol 2, pp 173-190, 1987.