VLSI DESIGN
1999, Vol. 9, No. 1, pp. 29–54
Reprints available directly from the publisher
Photocopying permitted by license only

Investigation of Various Mesh Architectures with Broadcast Buses for High-Performance Computing

SOTIRIOS G. ZIAVRAS*

Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights, Newark, New Jersey 07102

(Received 5 May 1997)

Extensive comparative analysis is carried out of various mesh-connected architectures that contain sparse broadcast buses for low-cost, high-performance parallel computing. The two basic architectures differ in the implementation of bus intersections. The first architecture simply allows row/column bus crossovers, whereas the second architecture implements such intersections with switches that introduce further flexibility. Both architectures have lower cost than the mesh with multiple broadcast, which has buses spanning each row and each column, but the former architectures maintain to high extent the powerful properties of the latter mesh. The architecture that employs switches for the creation of separable buses is even shown to often perform better than the higher-cost mesh with multiple broadcast. Architectures with separable buses that employ store-and-forward routing often perform better than architectures with contiguous buses that employ the high-cost wormhole routing technique. These architectures are evaluated in reference to cost, and efficiency in implementing several important operations and application algorithms. The results prove that these architectures are very promising alternatives to the mesh with multiple broadcast while their implementation is cost-effective and feasible.

Keywords: Enhanced mesh, mesh with multiple broadcast, parallel algorithms, parallel computer architectures, store-and-forward routing, wormhole routing

I. INTRODUCTION

The mesh architecture is used frequently in parallel processing because of its low VLSI complexity and its support for scalability. However, its main drawbacks, namely large diameter and large average internode distance, affect dramatically its communication capabilities. Although other popular interconnection networks, such as the direct binary hypercube, have smaller values for the latter pair of parameters, their major drawback is that they do not permit the application of incremental growth techniques [7] and their VLSI implementation becomes a Herculean task for

*Tel.: (973) 596-5651, Fax: (973) 596-5680. e-mail: ziavras@megahertz.njit.edu

massively parallel systems [2, 8]. To allow the efficient implementation of distant data transfers, several enhancements have been proposed for the mesh. The addition of a single global bus is such an enhancement [12, 13]. Although it is often assumed that the propagation time of messages on the global bus is independent of the size of the mesh, this justification may not be acceptable for practical systems [3].

To avoid bottlenecks caused by the single global bus, the mesh parallel computer can instead be augmented by adding multiple broadcast buses, where each bus connects a subset of PEs (processing elements) in the mesh. Such a popular architecture is the mesh with multiple broadcast [1, 21, 23]. It is a mesh-connected parallel computer where all PEs on each row and each column are connected to a shared row and shared column bus, respectively. The performance of this architecture is comparable to that of the pyramid computer for several image processing problems. Rectangular meshes with multiple broadcast may perform better than square ones with the same number of PEs, when row and column buses are considered, as the former systems contain more buses [3, 15, 29]. The mesh with multiple broadcast is the most relevant architecture in this paper. The rest of this section describes briefly other important variations of the mesh architecture.

The CHiP parallel computer [4] consists of a mesh (grid) of PEs with programmable switches interposed between neighboring PEs. The local memory in switches stores interconnection patterns to be implemented at run time. The reconfigurable mesh, or mesh with reconfigurable bus, is a square mesh of PEs where all PEs are connected to a global broadcast bus that spans all rows and columns [14, 22]. Switches are located at all column/row bus intersections and PEs control their neighboring switches that can divide the global bus into subbuses. The cost of this architecture may be prohibitively high because of its large number of switches, whereas the assumption of fixed-time data transfers may be unrealistic. Many algorithms have good theoretical performance on the reconfigurable mesh [17]. In the non-cross-over model, the four communication ports of a PE can be connected together to form only planar connections [27]. In the higher-cost cross-over model, non-planar connections can be formed as a PE may connect together its north–south and east–west ports independently.

The PEs in the polymorphic torus are located at the vertices of a two-dimensional torus network [5, 16, 25]. Switches are distributed over the nodes of the torus network, as for the reconfigurable mesh. Each switch can implement a complete graph of a PE's four ports. It supports high communication bandwidth, downplaying any hardware repercussions. Reconfiguration of switches is needed to match each time the structure of the program graph. This architecture has higher hardware cost than the reconfigurable mesh because crossbar switches are used to control all possible connections among the four NEWS (North–East–West–South) directions.

An X-shaped grid interconnects the PEs within each chip of the BLITZEN massively parallel processor array and extends across chip boundaries [18]. The network is dynamically reconfigured to support either NEWS or diagonal connections. The custom chip of BLITZEN contains 128 one-bit PEs arranged in an 8×16 array [18]. The standard configuration of this system contains 16,384 PEs arranged in an 128×128 array. The mesh of trees is constructed from a grid of PEs by adding additional PEs and wires to form on top of it a complete binary tree on each row and each column [9]. The cost of these additional PEs may be a drawback in the implementation of this architecture. Several slight variations of the mesh of trees have also been introduced [11].

Other mesh-connected parallel computers are obtained by superimposing one or more global meshes on an underlying mesh of PEs [6]. The mesh with a single global mesh is constructed starting with several regular meshes and connecting together with a global mesh the lead PEs at the top leftmost corners of these meshes. Additional links between PEs in the original meshes can be

used to form a single underlying mesh that contains all PEs. The creation of a mesh with l global meshes is also possible. The first global mesh connects the lead PEs of a set of regular meshes. The second global mesh connects the lead PEs of a subset of meshes with a single global mesh. This process is repeated, and finally, the lth global mesh connects a subset of meshes containing $l-1$ global meshes.

To improve the performance of the mesh with multiple broadcast, processor-controlled switches can be used to partition the row and column buses for reconfiguration purposes [26]. For example, a switch can be inserted after every other PE on each row or each column bus in order to produce a mesh with separable buses. A lower-cost modification of this architecture does not require that all PEs be connected to row and column broadcast buses [24]. The latter architecture employs one row/column bus for every n^k rows/columns of PEs in the $n \times n$ mesh, where $k < 1/2$. More specifically, PEs in blocks of size $n^k \times n^k$ are interconnected using only local links as in the mesh. The PE in the upper left corner of each block is connected to a pair of separable row and column buses. Data broadcasting algorithms first use local block links, then broadcast buses until full row or full column broadcasts are completed, and finally local block links again. Algorithms with limited global communication requirements, such as semigroup and prefix computations, can be implemented efficiently on these architectures. It has been shown that rectangular meshes are optimal for semigroup, prefix, and convex hull computations [24, 29]. However, we assume here only square meshes because our objective is to evaluate architectural differences of systems as they are related to data transfer operations.

The mesh with multiple broadcast and its aforementioned variation with separable buses achieve very good performance at the expense of very high hardware cost. Their existing cost-performance comparisons with alternative architectures are very limited. The objective of this paper is to show that families of low-cost mesh-connected architectures can achieve performance comparable to that of the higher-cost mesh with multiple broadcast. More specifically, this paper investigates in detail two families of mesh-connected architectures with sparse broadcast buses (*i.e.*, buses that do not cover every row and every column of the mesh). One of the architectures employs switches for the implementation of separable buses. In contrast to the work in [24] that assumes hierarchical sectioning of broadcast buses, these switches are located at all bus intersections and also connect to both row and column buses. Thus, they reduce the hardware cost further while improving the system's flexibility for many operations. Also, we assume that the underlying structure is a single/complete mesh, while [24] assumes independent submeshes that can communicate only *via* broadcast buses. Another innovation of this research is that extensive analysis is carried out for two classes of systems, namely those that employ the lower-cost store-and-forward routing technique and those employing the higher-cost wormhole routing technique.

The paper is organized as follows. Section II introduces the two families of mesh-connected architectures for the study. The development for these architectures of important operations and algorithms, mainly for comparison with the mesh with multiple broadcast, is the main objective in the succeeding sections. Algorithms for global broadcasting are presented in Section III. The implementation of some other fundamental data movement operations is presented in Section IV. Section V presents advanced prefix computation and graph component-labeling algorithms. Finally, Section VI contains conclusions.

II. MORB MESHES WITH MULTIPLE ORTHOGONAL BROADCAST BUSES

This section presents promising models(families) of low-cost *meshes with multiple orthogonal broadcast buses or multiple-orthogonal-broadcasts*

(MORB) *meshes* for the construction of low-cost, high-performance parallel computers. A MORB mesh contains sparse row and column broadcast buses. Therefore, it contains only a subset of the broadcast buses found in the mesh with multiple broadcast. Another innovation of this work is that both store-and-forward and wormhole-routing switching systems are investigated in detail. Two families of MORB meshes are studied here. The first family contains meshes with contiguous row/column buses, whereas the second family contains meshes with separable buses implemented with switches located at all intersections of broadcast buses.

Additional flexibility is introduced in the latter case (*i.e.*, MORB meshes with separable buses) due to the switches that can divide broadcast buses into subbuses for a much larger number of small-range broadcast operations. The properties of two-dimensional SIMD meshes are studied here. In the rest of the paper, MORB(n,p) represents an $n \times n$ grid of PEs (*i.e.*, a regular mesh), with p row and p column broadcast buses superimposed on it; $(r/2)$ is always chosen to be a natural number, where $r = ((n-1)/(p-1))$. The notation for the corresponding system with separable buses is MORBR^R (n,p). Throughout this paper we use the assignments $r = ((n-1)/(p-1))$, $R = (n/p)$, $N = n-1$, and $P = p-1$. Several sets of word-wide buses are embedded recursively into the regular $n \times n$ mesh in order to produce the MORB(n,p). The first set contains four distinct buses that connect together the PEs on the rows and columns at the boundaries of the mesh. That is, the $4N$ PEs on row 0, row N, column 0, and column N arc connected to this first set of four buses; each PE in a corner of the mesh is connected to both row and column buses. The second set of two buses coincide with the lines that divide the mesh into four equal-sized quadrants. More sets of buses are introduced by dividing each quadrant of PEs into four equal-sized subquadrants, and this procedure is repeated for the introduction of more buses. Figure 1 shows the structure of the MORB(17,5).

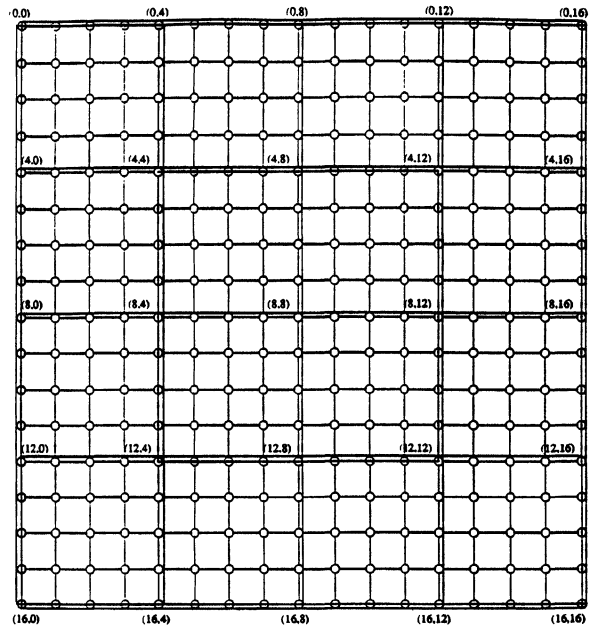

FIGURE 1 Structure of the MORB(17,5); double lines represent broadcast buses.

The term *bus segment* is used here to denote any contiguous part of a broadcast bus that falls between two consecutive bus crossovers. It is assumed that PEs connected to two (*i.e.*, row and column) buses can receive data on both buses simultaneously while they can transmit data on only one bus at a time. More than one PE can transmit simultaneously on the same bus only if all of them send the same value. The $n \times n$ mesh with multiple broadcast or multiple-broadcast mesh [21], denoted here by MB(n), is identical to the MORB(n,n). The broadcasting structure of the MORB mesh is similar to a special instance of the mesh with separable buses that has only one sectioning level for row and column broadcast buses [24].

A simple programmable switch is placed at each intersection of row and column buses in the MORB^R(n,p). These switches can partition row/column buses into subbuses for the simultaneous implementation of many broadcasts of limited range. This capability is not present in the mesh with multiple broadcast. A realistic approach is

taken here for the implementation on the MORB$^R(n,p)$ of broadcast operations with short communication (clock) cycle. More specifically, only the following types of PE coverage by bus broadcasts are allowed. (1) An entire row or column of PEs attached to the same broadcast bus. (2) An entire row or column of PEs attached to the same broadcast bus as well as any number of PEs attached to single column or row bus segments, respectively, that touch the former bus. (3) A contiguous part of a row or column of PEs attached to the same broadcast bus as well as any number of PEs attached to any number of single column or row bus segments, respectively, that touch any of the former PEs. These limited broadcast types guarantee low hardware complexity for the switches (otherwise, high-cost precharged circuits are needed to facilitate distant data transfers) and small clock cycle comparable to that of the mesh with multiple broadcast, thus supporting system scalability. In further comparison, the reconfigurable mesh [10] contains a much larger number of switches with significantly higher complexity because $O(n^2)$ switches may be traversed by a packet during a single data transfer (therefore, making absolutely essential the incorporation of precharged circuits). Additionally, long clock cycles may become inevitable in the implementation of distant bus transfers on the reconfigurable mesh. The mesh with fewer separable buses [24] cannot implement directly the PE coverages numbered (2) and (3). It can implement only the first type of PE coverage. Also, for the same fixed distance between consecutive switches the latter system requires double the number of switches found in the MORB^R mesh, because it uses distinct switches for row and column broadcast buses. Although the implementation on the MORB^R mesh of the PE coverages numbered (2) and (3) may require switches of slightly higher complexity, the reduced number of switches results in significant cost savings. Additionally, another major innovation of this paper is that both store-and-forward and wormhole-routing switching systems are studied extensively.

Hardware cost analysis for these families of meshes and relevant comparison with the mesh with multiple broadcast, the most relevant existing system, follow. In the rest of this section only, MORB mesh stands for both the MORB and MORB^R meshes. Also, throughout the paper *bus intersection* stands for *bus crossover*.

PROPOSITION 1 *The m-th set of broadcast buses in the MORB(n,p) contains* 2^{m-1} *(full-length) buses for* $m > 1$, *and four buses for* $m = 1$. *Each bus covers n PEs.*

Proof The first set of four broadcast buses (*i.e.*, for $m = 1$) cover the PEs on the boundaries of the mesh. For $m = 2$, the number of additional broadcast buses is equal to two, *i.e.*, 2^1, because the original mesh is divided into four quadrants by two buses that intersect in the center of the mesh. The proof for $m > 2$ is based on mathematical induction. Assume that the kth set contains 2^{k-1} buses, where $k > 2$. For the $(k+1)$th set, the number of row(column) buses introduced is double the number of row(column) buses in the kth set. Therefore, the number of buses in the $(k+1)$th set is 2^k. Hence, the result. ■

COROLLARY 1 *Ignoring duplication of PEs at bus intersections, the m-th set of broadcast buses cover* $2^{m-1} \times n$ *PEs for* $m > 1$, *and* $4n$ *PEs for* $m = 1$.

Proof Each row or column bus covers n PEs. From Proposition 1, the mth set contains 2^{m-1} buses for $m > 1$, therefore the total number of PEs covered by the mth set of buses is $2^{m-1} \times n$. It is $4n$ PEs for $m = 1$. PEs attached to two broadcast buses are considered twice in the calculations here, once for each bus. ■

PROPOSITION 2 *Ignoring duplication of PEs at bus intersections, the total number of PEs directly connected to broadcast buses in the MORB(n,p) is* $2pn$.

Proof Each row or column bus covers n PEs. There exist $2p$ buses, therefore the total number of PEs directly connected to broadcast buses is $2pn$. ■

The total number of switches in the $MORB^R(n,p)$ is equal to p^2. This is also the total number of bus intersections in the $MORB^R(n,p)$. In contrast, the total number of bus intersections in the MB(n) mesh with multiple broadcast is n^2 and results in much higher VLSI complexity for $n > p$ because of the implementation of more wire crossovers.

COROLLARY 2 *Assume that each PE needs a single communication port for its connection to a broadcast bus. The total number of ports in the MORB(n,p) for connections to broadcast buses is equal to* $2pn$.

Proof According to Proposition 2, the total number of PEs connected to broadcast buses is $2pn$ if bus intersections are not accounted for. Therefore, the total number of communication ports is equal to $2pn$. ■

The ratio of the total numbers of ports used for connections to broadcast buses in the MB(n) and MORB(n,p) meshes, respectively, is $((2n^2)/(2pn)) = R > 1$. Obviously, for fixed mesh size the value of this ratio increases with a decrease in p (*i.e.*, the number of broadcast buses in the MORB mesh). Considering also the cost of missing bus segments, the cost of the MORB mesh may be dramatically lower than that of the MB mesh. For fair hardware cost comparison with the $MORB^R(n,p)$ mesh, the cost of simple switches at bus intersections in the latter system must be accounted for. There exist p^2 such switches. Assuming that the cost of a broadcast bus segment between two neighboring PEs is c_1, the cost of a switch is c_2, the cost of a port is c_3, and the cost of implementing a bus wire crossover is c_4, then the ratio of the cost overheads for the provision of hardwired broadcast capabilities on the MB(n) and $MORB^R(n,p)$ meshes, respectively, is $(2nNc_1 + 2n^2c_3 + (n^2 - 4n + 4)c_4)/(2pNc_1 + p^2c_2 + 2pnc_3)$. The value of this cost ratio should be expected to be greater than 1 in all practical cases (note: it is always true that $p < n$). For $c_1 = c_2 = c_3 = c_4$ which could be an assumption very close to reality, the value of the above ratio is $(5n^2 - 6n + 4)/(p(p + 4\ n - 2))$; this value is approximately equal to (5/4) R (*i.e.*, it is always greater than 1) for $n > p \gg 1$. This analysis proves that even the robust $MORB^R$ mesh architecture, which has higher cost than the basic MORB mesh architecture, results in very significant cost savings when compared to the MB mesh architecture. Nevertheless, MORB meshes can become better alternatives than the mesh with multiple broadcast only if they can achieve performance comparable or better than that of the latter for frequently used operations and algorithms. Relevant comparisons follow in the rest of the paper.

III. GLOBAL BROADCASTING

A significant innovation of the analysis in this paper is that it is carried out for both the store-and-forward and wormhole-routing switching techniques. Before we dwell on the presentation and analysis of algorithms, let us describe timing models for these techniques. In store-and-forward switching, a packet is always stored temporarily in the packet buffer of any encountered intermediate processor. The network latency is given by $(M/B)L$, where M is the packet length (in number of bits, including the header), B is the bandwidth of a channel (in bits/sec), and L is the length (in number of hops) of the path travelled by the packet. In wormhole routing, each packet is divided into a number of flits (flow control digits) for transmission. As the header flit opens the selected path, the other flits of the packet are pipelined along the same path. If the header flit is blocked because of an unavailable channel, the remaining flits are also blocked along the same path; they are stored temporarily in flit buffers. The network latency is approximately equal to $(M_f/B)L + (M - M_f)/B$, where M_f is the flit length (we assume that the header is one flit long). For practical cases with $M_f \ll M$, the path length L does not have any significant effect on the network latency (assuming that L is not much larger than M).

The following conventions are used in this paper for the sake of simplicity. All network latencies are expressed here in numbers of communication cycles; a communication cycle is assumed to have the same duration as a clock cycle and is consumed for the transmission on a word-wide channel (all channels are assumed to be word-wide) of a single word between any two neighboring processors. For the sake of simplicity, the length of packets in our calculation of network latencies is expressed in number of communication cycles by assuming that $B = 1$ word/cycle; as a consequence, M and M_f represent numbers of communication cycles. The network latency is then approximated by $M\,L$ and $L + M$ for the store-and-forward and wormhole-routing techniques, respectively. For the sake of simplicity, a convention used here is that whenever L appears in an equation as an additive term because of wormhole routing, it represents communication cycles.

The problem of global broadcasting on MORB meshes, where a single PE copies a value into all other PEs in the system, is investigated here. This section is devoted exclusively to global broadcasting because this is an operation encountered very frequently in application algorithms. The primary goal is to illustrate that in practical cases the reduced number of broadcast buses does not seriously affect the broadcasting capability of the MORB mesh, when compared to the MB mesh. Since the algorithm of global broadcasting presented here is common to the MORB and MORB^R meshes, MORB mesh in this section stands for both the MORB and MORB^R meshes.

THEOREM 1 *The worst case time for a global broadcast on the MORB(n,p) is r NEWS communication steps and three bus broadcasting steps, for $p < n$. It is just two bus broadcasting steps for $p = n$ (i.e., similarly to the MB mesh).*

Proof Each side of the smallest-sized squares formed by broadcast buses covers $1+r$ PEs. Therefore, the maximum of the shortest distances of PEs from their nearest PE which is attached to a broadcast bus is $(r/2)$ (this is an integer number for $p < n$, as noted in Section II). This is also the largest possible number of NEWS communication steps required to reach the nearest broadcast bus. One broadcasting step is then needed to broadcast the value on the full bus to which the receiving PE is attached. Two additional bus broadcasting steps (*i.e.*, for row and column broadcast buses) are needed in the worst case for all PEs attached to broadcast buses to receive the value. The same number of NEWS communication steps as in the beginning (*i.e.*, $(r/2)$) are then required to copy the value from the upper and left boundaries of the aforementioned squares into the PEs within the squares. It is easy to see that only two bus broadcasting steps are required for $p = n$. Hence, the result. ■

This paper focuses on detailed time analysis of fundamental operations and algorithms for the MORB and MB meshes. Two delay models for bus broadcasts are used. These models are similar to those assumed for the reconfigurable mesh in asymptotic time analysis [22]. The *unit-time delay* (UTD) model assumes that a bus broadcast takes $O(1)$ time. The *log-time delay* (LTD) *model* assumes that a bus broadcast takes $O(\log l)$ time, where l is the number of switches or bus crossovers traversed. All logarithms here are to the base 2 and the emphasis is on detailed worst-case time analysis. It is assumed that each arithmetic or logic ALU operation requires t_{ALU} clock cycles, whereas each data transfer operation with a neighbor in the mesh requires t_{NEWS} clock cycles; therefore, $t_{\mathrm{NEWS}} = M$. Finally, a single bus broadcast takes t_{bus} clock cycles under the UTD model and $t_{\mathrm{bus}} \log l$ clock cycles under the LTD model, where l is the number of switches or bus crossovers traversed.

The store-and-forward and wormhole-routing switching techniques are chosen throughout this paper. The analysis of broadcasting first assumes the store-and-forward technique. Based on Theorem 1, the worst-case time for a global broadcast on the MORB(n,p) is $T_{\mathrm{global}}(n,p) = rt_{\mathrm{NEWS}} + 3\,t_{\mathrm{bus}}$

and $T'_{\text{global}}(n,p) = rt_{\text{NEWS}} + 3\log pt_{\text{bus}}$ under the UTD and LTD models, respectively, for $p < n$. For $p = n$, the values are $T_{\text{global}}(n,n) = 2t_{\text{bus}}$ and $T'_{\text{global}}(n,n) = 2\log pt_{\text{bus}}$, respectively, for the two models. Let us define as the cost of implementing global broadcasting on the MORB(n,p) the product $\text{cost}_{\text{global}}(n,p) = \text{PE\_buses}(n,p) \times T_{\text{global}}(n,p)$, where PE_buses stands for the total number of PEs attached to broadcast buses and is given in Proposition 2. This cost is in reference to the regular $n \times n$ mesh without broadcast buses, and can be used to compare the MORB and MB meshes for this operation; the MB(n) mesh has the same PE_buses as the MORB(n,n), that is $2n^2$. If $q = t_{\text{bus}}/t_{\text{NEWS}}$, the cost ratio $\text{cost}_{\text{global}}(n,p)/\text{cost}_{\text{global}}(n,n)$ becomes $p(r + 3q)/(2nq)$ and $p(r + 3q\log p)/(2nq\log p)$ under the UTD and LTD models, respectively. The smaller the value of this ratio, the better the MORB(n,p) system is for global broadcasting (*i.e.*, it has a better balance between hardware cost and performance). Under the UTD model, this ratio is approximately equal to $(r + 3q)/(2rq)$. Similarly, the time ratio $T_{\text{global}}(n, P)/T_{\text{global}}(n,n)$ is approximated by $(r + 3q)/(2q)$ for the UTD model.

Figure 2 shows these cost and time ratios as functions of the parameters q and r. The first graph shows that in practical cases (*i.e.*, $r \geq 4$) the cost ratio is much less than 1, while the second graph shows that the (execution) time ratio (*i.e.*, speedup) is less than 5. It can be concluded that the MORB mesh results in very significant cost savings, while it achieves good performance in global broadcasting. Since global broadcasting is the operation for which the MB mesh achieves its most impressive result, the results above are very encouraging for the viability of MORB meshes.

Assuming wormhole routing for the MORB (n,p) mesh, we have $T^W_{\text{global}}(n,p) = r + t_{\text{NEWS}} + 3t_{\text{bus}}$ and $T^W_{\text{global}}(n,p) = r + t_{\text{NEWS}} + 3\log pt_{\text{bus}}$, for $p \leq n$, under the UTD and LTD models, respectively. We have $(\text{cost}^W_{\text{global}}(n,p))/(\text{cost}^W_{\text{global}}(n,n)) = p(r + t_{\text{NEWS}} + 3t_{\text{bus}})$ under the UTD model. This ratio is much less than 1 in all practical cases. Wormhole routing results in dramatic performance improvement because it corresponds to a much lower cost ratio.

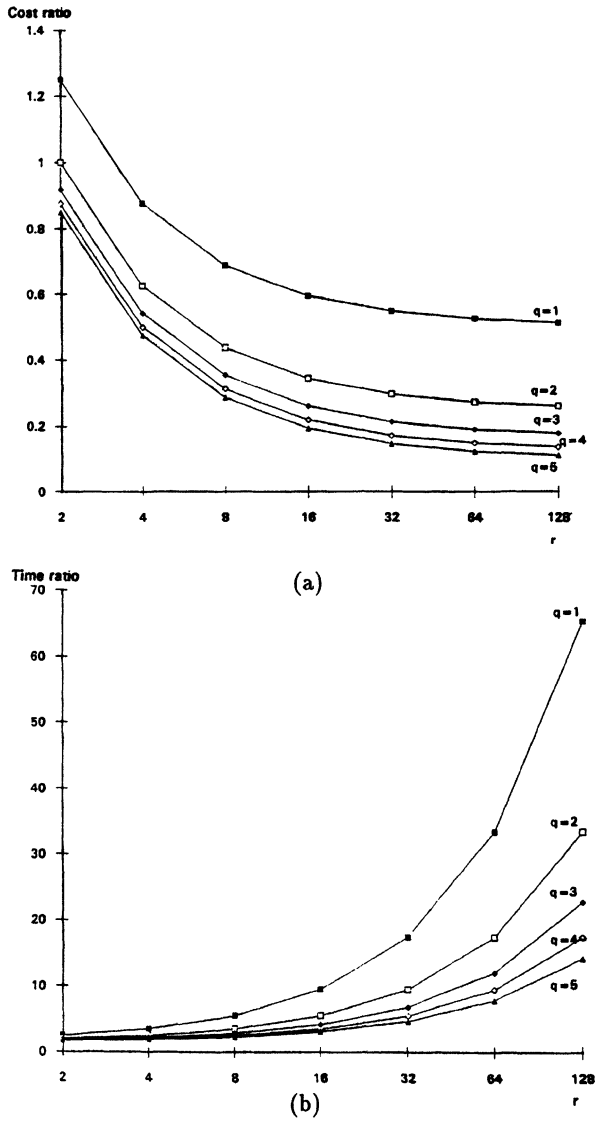

FIGURE 2 Approximation of the (a) cost ratio $\text{cost}_{\text{global}}(n,p)/\text{cost}_{\text{global}}(n,n)$ and (b) time ratio $T_{\text{global}}(n,p)/T_{\text{global}}(n,n)$ under the UTD model, where $r = (n-1)/(p-1)$ and $q = t_{\text{bus}}/t_{\text{NEWS}}$.

IV. OTHER DATA MOVEMENT OPERATIONS

The efficient implementation of popular data movement operations that involve on-the-fly

computations is critical for the successful employment of parallel computers [10]. For example, semi-group computations appear quite often in parallel algorithms [15]. The semigroup computation that employs the associative operator $\otimes$ calculates $a_1 \otimes a_2 \otimes a_3 \otimes \ldots \otimes a_N$ for a set of N data items. The operation can be add, multiply, maximum, minimum, *etc*. Without loss of generality, the example chosen here is the computation of the logical OR of n data items, stored one datum per $PE_{i,j}$ on the ith row of the $n \times n$ mesh, with the result stored in the leftmost $PE_{i,0}$, for all $0 \leq i, j \leq N$. The next theorem is pertinent for systems without wormhole routing.

THEOREM 2 *Given the MORB(n,p) where each PE stores a datum, the logical OR of the data on individual rows/columns can be found in parallel in time*

$$T_{rows\_OR}(n, p) = (r - 1 + 2(n - p))t_{NEWS} + (rp - 1)t_{ALU} + N t_{bus}$$

under the UTD model. The t_{bus} term for the LTD model is r log (p!) t_{bus}. p! represents the p factorial.

Proof Let the operation be applied to row data. In the first phase the OR operation is applied sequentially from right to left within row segments that fall between column broadcast buses. More specifically, at the end of the first phase each PE with Cartesian coordinates $(i, r\ j_1)$ contains the result of the OR operation among the data stored in the PEs with coordinates $(i, r\ j_1 + j_2)$, for all $0 \leq j_1 \leq p - 2$ and $0 \leq j_2 \leq r - 1$. This phase requires time $t_1 = (r-1)\ (t_{\text{NEWS}} + t_{\text{ALU}})$, in number of clock cycles. The second phase is composed of P subphases, where each subphase k, for $k = 1, 2, \ldots, P$, is composed of r cycles that transfer all values from the kth column broadcast bus to the PEs on column 0 (attached to the 0th broadcast bus) on the same row with the corresponding senders. The OR operation is also applied to the received data. For the implementation of data transfers, the NEWS network as well as broadcast buses are used. This phase takes time $t_2 = P(2(r-1)\ \ t_{\text{NEWS}} + r(t_{\text{bus}} + t_{\text{ALU}}))$ or $t_2 = (n-p)t_{\text{NEWS}} + N(t_{\text{bus}} + t_{\text{ALU}})$ under the UTD model. For the LTD model the time is $t_2' = 2P(r-1)t_{\text{NEWS}} + \sum_{k=1}^{p} r(\log(k+1)t_{\text{bus}} + t_{\text{ALU}})$ or $t_2' = 2(n-p)t_{\text{NEWS}} + Nt_{\text{ALU}} + r\log(\prod_{k=2}^{p} k)t_{\text{bus}}$. Hence, the result. ■

Therefore, this algorithm consumes time $O(n)$ under the UTD model. The time is $O(n + (n/p)\log(p!))$ under the LTD model. *In the rest of this paper, asymptotic time analysis is presented only for the UTD model because the latter provides very realistic performance while it is simple. Also, we always assume the UTD model for systems with wormhole routing.*

For a system with wormhole routing, $t_1^W = (r-1)t_{\text{ALU}} + r - 1 + M$, where r and M now represent numbers of communication cycles. Wormhole routing reduces dramatically the amount of time consumed in the second phase to $t_2^w = P(2(r - 1 + M) + r(t_{\text{bus}} + t_{\text{ALU}}))$. Thus,

$$T^W_{\text{rows\_OR}}(n, p) = (r - 1 + M)(1 + 2P) + N t_{\text{bus}} + (rp - 1)t_{\text{ALU}}.$$

Therefore, this algorithm consumes time $O(n)$ with wormhole routing as well. For $t_{\text{NEWS}} = M$, the ratio of NEWS communication overheads of the store-and-forward and wormhole-routing techniques is $[r - 1 + 2(n-p)]M/[(r - 1 + M)(1 + 2P)]$ or $M/[1 + (M/(r-1))]$. This ratio is shown in Figure 3 as a function of the parameters r and M. It can be concluded that wormhole routing results in dramatic performance improvement.

A word of caution is in order here. This problem may consume less time if data from more PEs than those corresponding to a single row segment are combined first. However, the purpose of this paper is not necessarily to always devise the best possible algorithm to solve a given problem, because of its comparative nature that emphasizes architectural differences. For the most efficient implementation of this operation, the technique devised for the second phase of the algorithm in Corollary 3 can

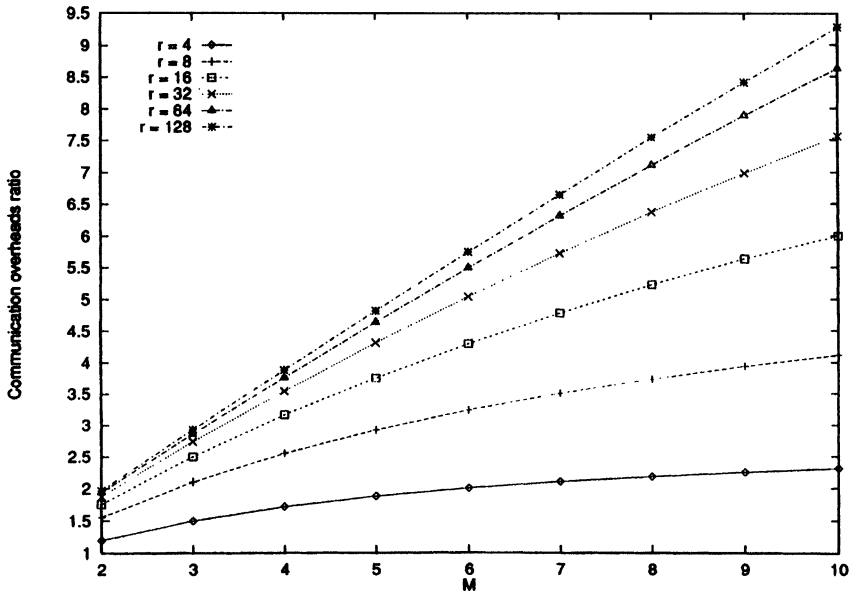

FIGURE 3 The ratio of NEWS communication overheads of the store-and-forward and wormhole-routing techniques for finding the logical OR of data on individual rows/columns, under the UTD model.

be used. Depending on the values of n and p, it may be that using the naive solution that does not incorporate broadcast buses can result in better performance; the latter solution has execution time $(n-1)t_{\text{NEWS}} + (n-1)t_{\text{ALU}}$.

The next corollary expands further by investigating the case of producing a single global result on systems without wormhole routing. The corresponding algorithm emphasizes optimality and its exact implementation depends on the hardware characteristics of the system under consideration.

COROLLARY 3 *Given the MORB(n,p) where each PE stores a datum, the logical OR among all data in the system can be found and stored in the* $PE_{0,0}$ *in time* $T_{all\_OR}(n,p) = \tau + t_c$ *and* $T'_{all\_OR}(n,p) = \tau + t'_c$ *under the UTD and LTD models, respectively, where*

- $\tau = (r/2)t_{NEWS} + ((r/2) + p)t_{ALU} + Pt_{bus}$,
- $t_c = (m-1)t_{NEWS} + (\lceil (n/m) \rceil - 1)t_{bus} + (m + \lceil (n/m) \rceil - 2)t_{ALU}$, *and*
- $t'_c = (m'-1)t_{NEWS} + [\lceil (n/m') \rceil - 1 + \log(\prod_{l=1}^{\lceil (n/m') \rceil - 1} \times \lfloor ((lm'+1)/p) \rfloor]t_{bus} + (m' + \lceil (n/m') \rceil - 2)t_{ALU}$.

Assuming a single broadcast bus that covers n PEs, the values of m and m′ are such that if the partial OR results are first found for contiguous segments of m or m′ PEs, respectively, and these partial results are then combined using the broadcast bus, the value of t_c, t'_c *is minimized under the UTD or LTD model, respectively.*

Proof The algorithm for this reduction operation comprises three phases. In the first phase, every PE which is not attached to a row broadcast bus sends its value to its closest PE which is attached to a row broadcast bus. The OR operation is applied to contained and received values in all intermediate steps. This phase consumes time $t_1 = (r/2)(t_{\text{NEWS}} + t_{\text{ALU}}) + t_{\text{ALU}}$ under both delay models. At this point, the OR reduction operator must be applied only by PEs attached to a row broadcast bus. In the second phase, the OR results are found in parallel for all rows of PEs which are attached to a row broadcast bus. This is accomplished in optimal time as follows. First, assume the UTD model. This phase is carried out by first finding the

partial OR results for contiguous segments of size m on each broadcast row and then using the broadcast bus to collect these partial results in the corresponding PEs of column 0; the optimal size m is determined later in this proof. Actually, there exist $\lfloor (n/m) \rfloor$ segments of size m and one segment of size $n - \lfloor (n/m) \rfloor$ m on each row. The total number of segments on a row is $\lceil (n/m) \rceil$. Because of the SIMD execution mode, this phase requires time $t_c = (m-1)$ $(t_{\text{NEWS}} + t_{\text{ALU}}) + (\lceil (n,m) \rceil - 1)$ $(t_{\text{bus}} + t_{\text{ALU}})$ or $t_c = (m-1)t_{\text{NEWS}} + (\lceil (n/m) \rceil - 1)$ $t_{\text{bus}} + (m + \lceil (n/m) \rceil - 2)t_{\text{ALU}}$. The value of m that minimizes t_c is found by $(dt_c/dm) = 0$ and is given by

$$m = \left\lfloor \left(n \frac{t_{\text{bus}} + t_{\text{ALU}}}{t_{\text{NEWS}} + t_{\text{ALU}}} \right)^{1/2} \right\rfloor.$$

For the LTD model the value of t'_c for contiguous segments of size m' is given by $t'_c = (m'-1)$ $(t_{\text{NEWS}} + t_{\text{ALU}}) + \sum_{l=1}^{\lceil (n/m') \rceil - 1} [(\log \lfloor ((l\,m' + 1)/p) \rfloor$ $+1)t_{\text{bus}} + t_{\text{ALU}}]$, where 0 is used instead of the logarithm if $\lfloor ((lm' + 1)/p) \rfloor = 0$. Thus, $t' = (m'-1)$ $t_{\text{NEWS}} + [\lceil (n/m') \rceil - 1 + \log(\prod_{l=1}^{\lceil (n/m') \rceil - 1} \lfloor ((lm' + 1)$ $/p) \rfloor)]t_{\text{bus}} + (m' + \lceil (n/m') \rceil - 2)t_{\text{ALU}}$. The third phase applies the OR operator to data stored in the PEs on column 0 which are attached to a row broadcast bus. Data are combined sequentially in pairs using the broadcast bus and starting with the PE that has the largest row number. This phase takes time $t_3 = P(t_{\text{bus}} + t_{\text{ALU}})$ under both delay models. The total time is given by $T_{\text{all\_OR}}(n,p)$ $= t_1 + t_c + t_3$ and $T'_{\text{all\_OR}}(n,p) = t'_1 + t'_c + t_3$ under the UTD and LTD models, respectively. Hence, the result. ■

Therefore, this algorithm consumes time $O((n/p) + p + n^{1/2})$ under the UTD model. It is $O(p)$ for $O(p) \geq O(n^{1/2})$, and $O(n/p)$ for $O(p) < O(n^{1/2})$. Assuming a system with wormhole routing, we substitute $(r/2) + M$ for $(r/2)t_{\text{NEWS}}$ in t_1, and $m - 1 + M$ for $(m-1)t_{\text{NEWS}}$ in t_c. The new value of m is now found to be

$$m^W = \left\lfloor \left(n \frac{t_{\text{bus}} + t_{\text{ALU}}}{1 + t_{\text{ALU}}} \right)^{1/2} \right\rfloor.$$

For $t_{\text{NEWS}} = M$, the difference between the communication overheads of the store-and-forward and wormhole-routing techniques is $((r/2) + m^w - 1)(M-1) - 2M$. This difference is shown in Figure 4 as a function of the parameters n, p, and M, for $t_{\text{bus}} = t_{\text{ALU}} = 1$. The pure computation times (*i.e.*, iguoring communication overheads) for the curves from the bottom to the top of the graph are 56, 56, 77, 85, 112, and 136, respectively. Thus, wormhole routing may result in dramatic performance improvement.

The semigroup computation can be performed in time $O(n^{2/5})$ on a rectangular $n^{6/5} \times n^{4/5}$ MORB mesh with $r - n^{2/5}$ [24]. However, we consider only square meshes in this paper (as noted in the Introduction). The MORB^R mesh with separable buses is treated now for the same operations as above.

THEOREM 3 *Given the MORB^R (n,p) where each PE stores a datum, the logical OR of the data on individual rows/columns can be found in parallel in time*

$$\begin{aligned} T^R_{row\_OR}(n,p) = &(r-1)t_{NEWS} \\ &+ (r - 1 + \lceil \log P \rceil r)t_{ALU} \\ &+ \lceil \log P \rceil r\, t_{bus} \end{aligned}$$

under the UTD model. The t_{bus} term for the LTD model is $r \sum_{i=1}^{\lceil \log P \rceil} \log(2^{i-1} + 1)t_{bus}$.

Proof The first phase is identical to that for the MORB (n,p) mesh, requiring time $t_1 = (r-1)$ $(t_{\text{NEWS}} + t_{\text{ALU}})$. At this point, only PEs attached to a vertical broadcast bus contain partial results to be combined in the next stage. Binary trees are then emulated to repeatedly combine pairs of values within each row until a single result is produced for the row. This emulation is easy to implement because of the separability of buses. More specifically, $\lceil \log P \rceil$ iterations are needed to combine the partial results on each row. The ith iteration, where $i = 1, 2, \ldots \lceil \log P \rceil$, combines values, in pairs, of segments whose indices differ by 2^{i-1}; segment indices run from 0 to $p-1$. Because of the sparse broadcast buses, each

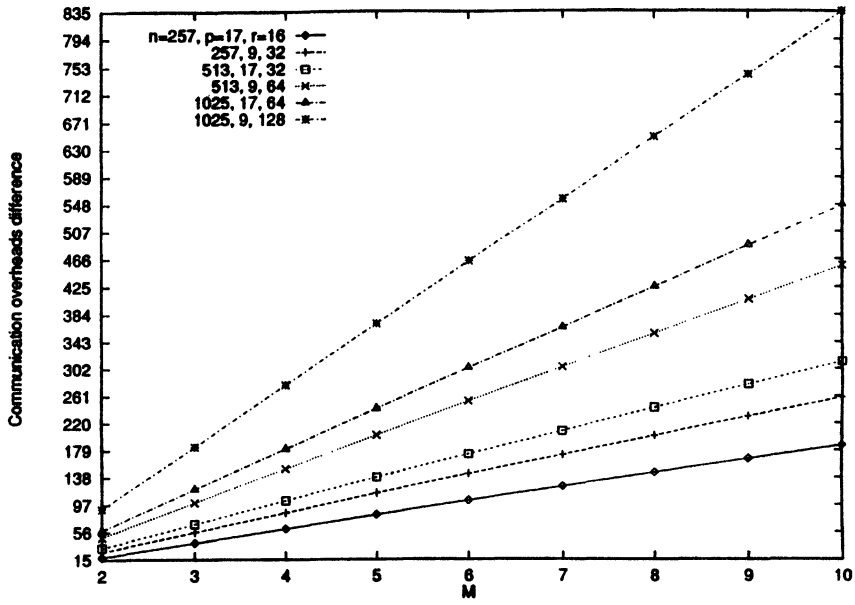

FIGURE 4 The difference between the communication overheads (in number of cycles) of the store-and-forward and wormholerouting techniques for finding the logical OR among all data in the MORB (n, p), under the UTD model.

iteration consumes r cycles, the same as the number of PEs within a (column) bus segment. Therefore, this phase requires time $t_2 = \lceil \log P \rceil r(t_{\text{bus}} + t_{\text{ALU}})$ under the UTD model. This phase for the LTD model requires time $t_2' = r\sum_{i=1}^{\lceil \log P \rceil} (\log(2^{i-1}+1)t_{\text{bus}} + t_{\text{ALU}})$ or $t_2' = r(\lceil \log P \rceil t_{\text{ALU}} + \sum_{i=1}^{\lceil \log P \rceil} \log(2^{i-1}+1)t_{\text{bus}}$. Hence, the result. ■

Therefore, this operation consumes time $O((n/p) \log p)$ under the UTD model. The separability of buses on the MORB$^R(n, p)$ reduces dramatically the time complexity of the algorithm in comparison to the MORB(n, p) mesh that requires time $O(n)$ for this operation (see Theorem 2). For the MORB$^R(n, p)$ with wormhole routing, we substitute $r - 1 + M$ for $(r-1)t_{\text{NEWS}}$. The communication overhead is thus reduced by $r(M-1) - 2M + 1$; this reduction is very significant for large values of r and M. Figure 5 shows the speedup $(T^W_{\text{rows\_OR}}(n, p)/T^R_{\text{rows\_OR}}(n, p))$ as a function of n, p, and M, for $t_{\text{bus}} = 1$ (*i.e.*, word-wide buses) and $t_{\text{bus}} = M$. The MORB(n, p) and MORB$^R(n, p)$ use wormhole routing and store-and-forward, respectively. This figure illustrates the versatility of the lower-cost mesh with separable buses that does not employ the higher-cost wormhole-routing technique.

The next corollary expands further by investigating the problem of producing a single global result on the MORB^R mesh with separable buses.

COROLLARY 4 *Given the MORB^R (n,p) where each PE stores a datum, the logical OR among all data in the system can be found and stored in the $PE_{0,0}$ in time*

$$T^R_{all\_OR}(n, p) = \left(\frac{3}{2}r - 1\right)t_{NEWS} + \left(\frac{3}{2}r + \lceil logP \rceil + \lceil logP \rceil\right)t_{ALU} + (\lceil logP \rceil + \lceil \log p \rceil)t_{bus}$$

under the UTD model. The t_{bus} term for the LTD model is

$$\left[2\sum_{i=1}^{\lceil logP \rceil} log(2^{i-1}+1) + log(2^{\lceil logp \rceil - 1}+1)\right]t_{bus}.$$

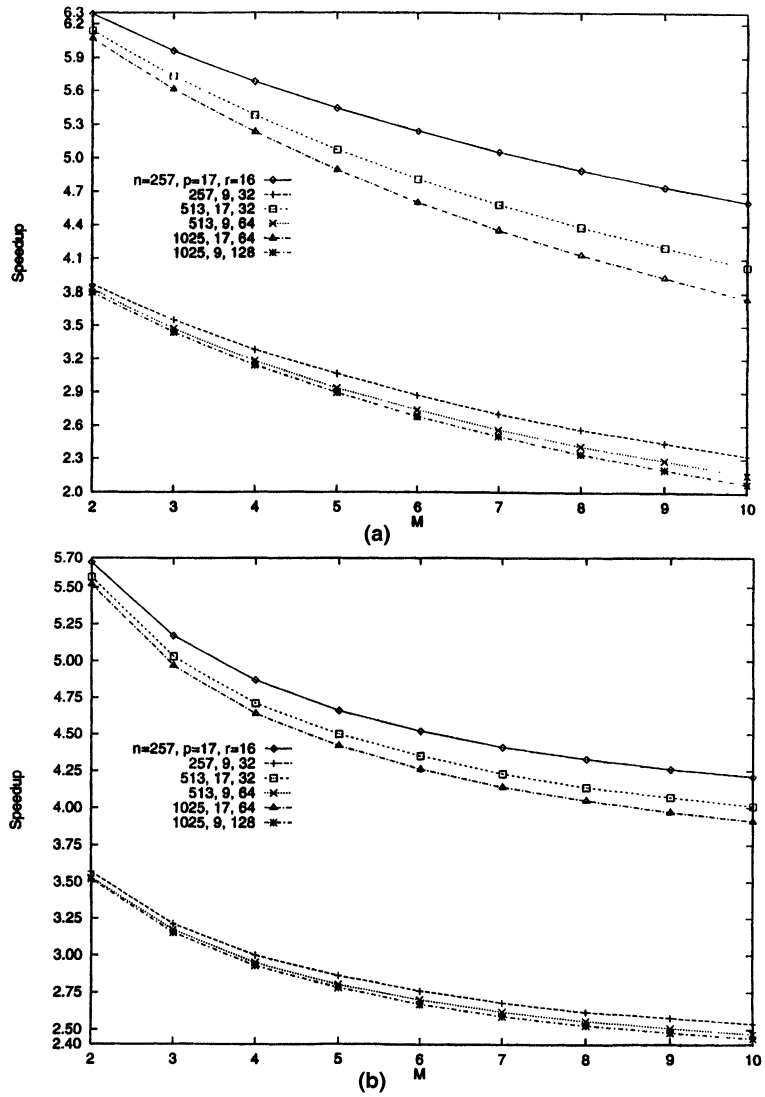

FIGURE 5 The speedup $(T^{W}_{\text{rows\_OR}}(n, p)/T^{R}_{\text{rows\_OR}}(n, p))$. (a)$t_{\text{bus}} = 1$. (b)$t_{\text{bus}} = M$.

Proof The algorithm for this reduction operation comprises three phases. In the first phase, every PE which is not attached to a row broadcast bus sends its value to its closest PE which is attached to a row broadcast bus. The OR operation is applied to contained and received values in all intermediate steps. This phase consumes time $t_1 = (r/2)\ (t_{\text{NEWS}} + t_{\text{ALU}}) + t_{\text{ALU}}$ under both delay models. At this point, the OR reduction operation must be applied only for PEs attached to a row broadcast bus. In the second phase, the OR results are found in parallel for all rows of PEs which are attached to a row broadcast bus. This is accomplished by first combining sequentially the data within row segments attached to broadcast buses and then reducing the values further by emulating binary trees, as in the proof of Theorem 3. This phase takes time $t_2 = (r-1)\ (t_{\text{NEWS}} + t_{\text{ALU}}) + \lceil \log P \rceil\ (t_{\text{bus}} + t_{\text{ALU}})$ under the UTD model. For the LTD model this phase takes time $t'_2 = (r-1)$

$t_{NEWS} + (r - 1 + \lceil \log P \rceil)t_{ALU} + \sum_{i=1}^{\lceil \log P \rceil} \log(2^{i-1} + 1)\, t_{bus}$. At the end of the second phase the partial results are stored in those PEs of column 0 which are located at bus intersections. These p values are combined in the third phase by emulating a binary tree on column 0. This phase takes time $t_3 = \lceil \log p \rceil (t_{bus} + t_{ALU})$ under the UTD model. For the LTD model this phase takes time $t'_3 = \lceil \log p \rceil t_{ALU} + \sum_{i=1}^{\lceil \log p \rceil} \log(2^{i-1} + 1) t_{bus}$. Hence, the result. ■

Therefore, this operation consumes time $O((n/p) + \log p)$ under the UTD model. In comparison to the MORB(n, p), the separability of buses permitted on the MORB$^R(n, p)$ reduces the time complexity of this operation since $O((n/p) + p + n^{1/2}) \geq O((n/p) + \log p)$. With wormhole routing, the communication overhead is reduced by $((3/2)r - 2)M - ((3/2)r - 1)$; this reduction is very significant for large values of r and M. The next corollary is introduced for the purpose of comparison with the higher-cost MB(n) mesh.

COROLLARY 5 *Given the MB(n) mesh with multiple broadcast where each PE stores a datum, the logical OR among all data in the system can be found and stored in the* $PE_{0,0}$ *in time* $T_{all\_OR}(n, n) = 2\ t_c$ and $T'_{all\_OR}(n, n) = 2t'_c$ *under the UTD and LTD models, respectively, where* t_c *and* t'_c *are as in Corollary* 3.

Proof The process described for the second phase of the algorithm in the proof of Corollary 3 is applied twice. This process consumes time t_c and t'_c under the UTD and LTD models, respectively. The first time it finds in parallel the logical OR results for individual rows and stores them on the leftmost column. The second time this process is applied to the latter column in order to store the final result in the $PE_{0,0}$. ■

Therefore, this algorithm consumes time $O(n^{1/2})$ under the UTD model. The asymptotic time required on the MORB(n, p) is larger for $O(n^{1/2}) > O(p)$. At lower hardware cost the MORB$^R(n, p)$ may perform better than the MB(n) for $O(n^{1/2}) < O(p)$. Also, it is important to notice that the asymptotic performance of the lower-cost MORB mesh is similar to that of the MB mesh for practical cases where $O(p) = O(n^{1/2})$.

Let us define as the cost of implementing this operation on the MORB(n, p), MORB$^R(n, p)$, and MB(n) meshes the products $\text{cost}_{\text{all\_OR}}(n, p) = \text{PE\_buses}(n,p) \times T_{\text{all-OR}}(n,p)$, $\text{cost}^R_{\text{all\_OR}}(n,p) = \text{PE\_buses}(n,p) \times T^R_{\text{all\_OR}}(n,p)$, and $\text{cost}^R_{\text{all\_OR}}(n, p) = \text{PE\_buses}(n, n) \times T^R_{\text{all\_OR}}(n, n)$, respectively. As earlier, PE_buses represents the total number of PEs attached to broadcast buses and is given by Proposition 2. This cost is in reference to the regular $n \times n$ mesh without broadcast buses, and can be used to compare the MORB, MORB^R, and MB meshes for this operation; the MB(n) mesh has the same PE_buses as the MORB(n, n), that is $2n^2$. If $t_{bus} = t_{NEWS} = t_{ALU}$, the cost ratios $(\text{cost}_{\text{all\_OR}}(n, p)/\text{cost}_{\text{all\_OR}}(n, n))$ and $(\text{cost}^R_{\text{all\_OR}}(n, p)/\text{cost}_{\text{all\_OR}}(n, n))$ under the UTD model are shown in Figure 6 for practical values of n and p. Figure 6 shows that the cost ratio is very small for both families of MORB meshes. It is even smaller for the MORB$^R(n, p)$ mesh because its separable buses support more efficient implementation of this global OR operation at a very small additional hardware cost.

For systems with wormhole routing, the additional communication overhead of the versatile MORB$^R(n, p)$ mesh when compared to the higher-cost MB(n) mesh is $O(r + \log p - n^{1/2})$. In practical cases this difference in communication overheads is either insignificant or negative, thus supporting our hypothesis that MORB^R meshes can often perform better than higher-cost meshes with multiple broadcast. Figure 7 that shows the difference $T_{\text{all\_OR}}(n, n) - T^{R,W}_{\text{all\_OR}}(n, p)$ supports our claim that MORB^R meshes with wormhole routing achieve much better performance in practical cases than MB meshes with multiple broadcast.

The last algorithm also can be used to find in the same amount of time the maximum or minimum of n^2 values, stored one value per PE. This is a very common operation. An alternative algorithm that implements this operation on the reconfigurable

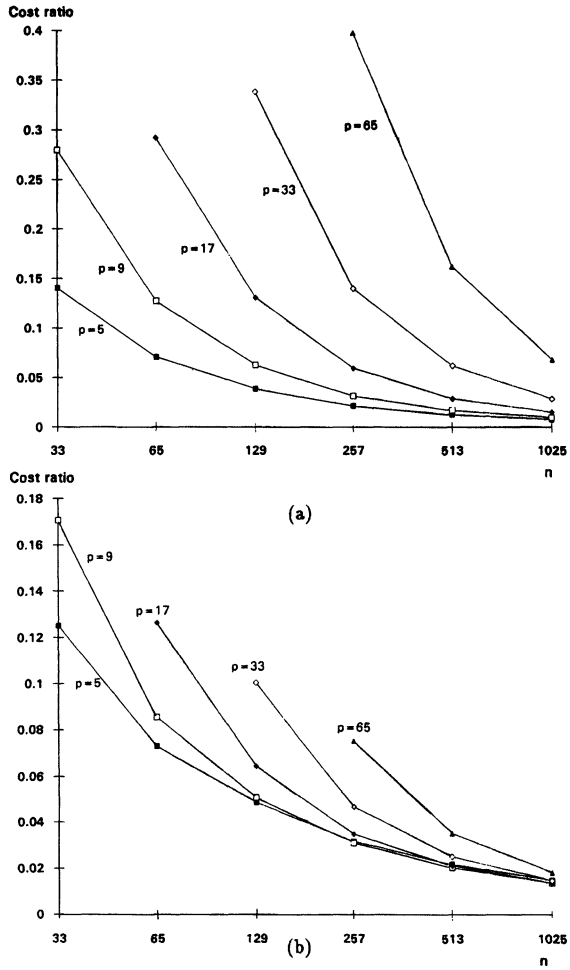

FIGURE 6 (a) The cost ratio $((\text{cost}_{\text{all\_OR}}(n,p)/\text{cost}_{\text{all-OR}}(n,n))$. (b) The cost ratio $(\text{cost}^{R}_{\text{all\_OR}}(n,p)/\text{cost}_{\text{all-OR}}(n,n))$. The UTD model and $t_{\text{bus}} = t_{\text{NEWS}} = t_{\text{ALU}}$ are assumed.

mesh was presented in [10]. The latter algorithm is adapted here for its application to MORB meshes. The proofs in the next two theorems show the modifications needed in the algorithm for its application to MORB and MORB^R meshes. The execution times stated in Corollaries 3 and 4 are then used in a comparative analysis involving the two algorithms that can solve the same problem.

THEOREM 4 *If data items are stored one per PE in the MORB(n, p), the maximum or minimum among the n^2 values can be found and stored in the $PE_{0,0}$ in time*

$$T_{max/min} = rt_{NEWS} + (p^2 + 3p + 2)t_{bus} + (r + p^2 + p + 4)t_{ALU}$$

under the UTD model. The t_{bus} term for the LTD model is

$$(4log(p!) - 4\log((\lceil p/2 \rceil - 1)!) + (p+1)\log p + p^2 + 1)t_{bus}.$$

Proof Without loss of generality, maximum in this proof stands for maximum or minimum. An algorithm developed for the reconfigurable mesh is modified here for the MORB mesh. The maximum among p values $x_0, x_1, \ldots, x_P$ initially stored on a single row of the reconfigurable $p \times p$ mesh is determined as follows [10]. Every $\text{PE}_{i,j}$ on row i first receives x_j, for all $0 \le i, j \le P$; a broadcast operation that involves all p columns is applied for this purpose. Then, every $\text{PE}_{i,i}$ uses a row broadcast to send its value x_i to all $\text{PE}_{i,j}$s on the same row, for all $0 \le i, j \le P$. Every $\text{PE}_{i,j}$ then compares the values x_i and x_j it contains and produces '1' if $x_j < x_i$, otherwise it produces '0'. If the result of the OR operation among all these values on column j is 0, then x_j is the maximum. The OR results for all columns are produced in parallel and stored in different PEs of row 0. These OR results can be found by bus splitting so that values from pairs of PEs are combined each time in a binary tree fashion. As more than one PE on row 0 may then contain the maximum value, a bus splitting technique is used on row 0 to send this value to the $\text{PE}_{0,0}$. More specifically, every PE on row 0 that contains the maximum value disconnects the bus on row 0 from its neighbor to the right and only PEs that contain the maximum value are allowed to broadcast. This way, the $\text{PE}_{0,0}$ receives the maximum value.

The same algorithm is applied here for data on each broadcast row of the MORB(n, p), after the smallest possible number of local maxima are

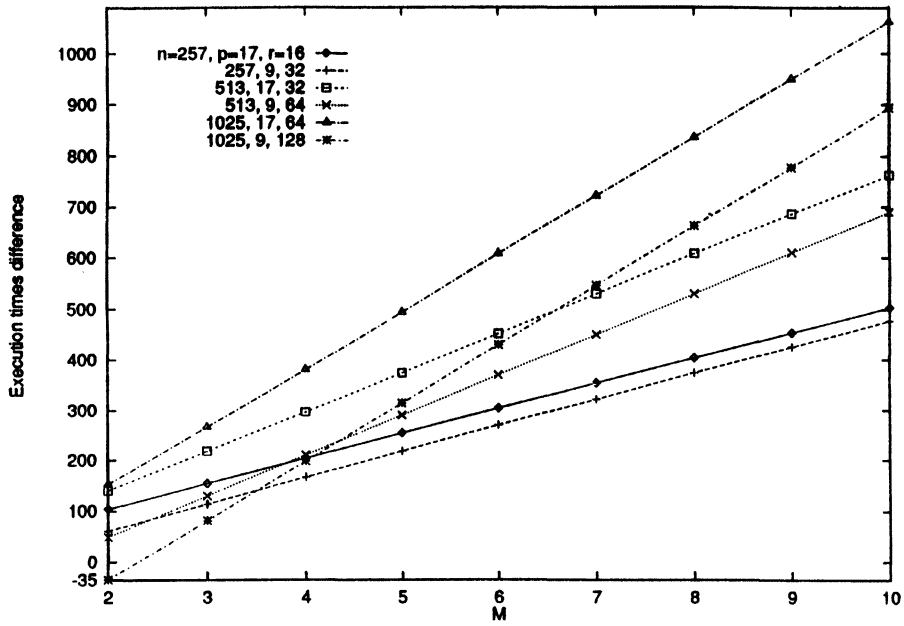

FIGURE 7 The difference $T_{\text{all-OR}}(n,n) - T^{R,W}_{\text{all-OR}}(n,p)$, in number of cycles, under the UTD model.

found for blocks of the mesh, with these local maxima being stored in PEs attached to broadcast buses. Therefore, the algorithm begins with every PE which is not attached to a broadcast bus sending its value to its closest PE which is attached to a broadcast bus. After every data transmission, the maximum value is found of the already contained value and any value received. This phase takes time $t_1 = (r/2)\ (t_{\text{NEWS}} + t_{\text{ALU}}) + t_{\text{ALU}}$; the last addition is due to PEs attached to broadcast buses that receive two values simultaneously. At this point, only the PEs attached to broadcast buses contain values which will be involved in the next phase of the algorithm. In the second phase, the values stored in the PEs attached to broadcast buses are combined further using NEWS connections. At the end of this phase, only the PEs at the intersections of broadcast buses contain values that remain to be combined. This second phase takes time $t_2(r/2)\ (t_{\text{NEWS}} + t_{\text{ALU}}) + 3t_{\text{ALU}}$; the last term is due to the simultaneous arrival of four values at some bus intersections.

Therefore, in the third phase the algorithm described in the beginning of this proof for the reconfigurable mesh is used p times, once for each set of values stored on a row broadcast bus (*i.e.*, similarly to data on a single row of the reconfigurable mesh). The following analysis assumes that in the kth execution of the third phase the maximum among the values on the $(k-1)$th row is found and stored in the $\text{PE}_{0,(k-1)r}$, where $k = 1, 2, \ldots, p$. Thus, at the end of the third phase the PEs of row 0 located at bus intersections are involved to find the maximum and store it in the $\text{PE}_{0,0}$, in time τ. Each application of the iterative process in the third phase requires time $\tau = 3\ t_{\text{bus}} + t_{\text{ALU}} + P\ (t_{\text{bus}} + t_{\text{ALU}})$ under the UTD model; each iteration involves a column broadcast, a row broadcast, an ALU comparison, a logical OR operation on columns, and the collection of the maximum value on row 0 using a single row broadcast (it was assumed in the Introduction that many PEs are allowed to send simultaneously the same value on the same broadcast bus). Therefore, the third

phase of the algorithm requires time $t_3 = (p + 1)\tau$ under the UTD model.

For the LTD model, the iterative process requires time $(2\sum_{i=\lceil (p/2)\rceil}^{p} \log i - \log(p/2))t_{\text{bus}}$ or $(2\log(p!) - 2\log((\lceil p/2\rceil - 1)!) - \log p + 1)t_{\text{bus}}$ for the column broadcasts. It takes time p log $p\,t_{\text{bus}} + p\,t_{\text{ALU}}$ for the row broadcasts and the subsequent comparisons. The OR reduction on columns consumes time $pP(t_{\text{bus}} + t_{\text{ALU}})$. The emulation of the bus splitting technique consumes the same amount of time as the column broadcasts. Finally, the end of the third phase that finds the maximum among the PEs on row 0 consumes time 3 $\log p\ t_{\text{bus}} + t_{\text{ALU}} + P(t_{\text{bus}} + t_{\text{ALU}})$. Therefore, the third phase of the algorithm requires time under the LTD model equal to $t_3' = (4\log(p!) - 4\log((\lceil (p/2)\rceil - 1)!) + (p+1)\log p + p^2 + 1)t_{\text{bus}} + p(p+1)t_{\text{ALU}}$. Hence, the result. ■

Therefore, the asymptotic time of this algorithm is $O((n/p) + p^2)$ under the UTD model. The algorithm presented in Corollary 3 can solve the same problem in time $O((n/p) + p + n^{1/2})$. It can be concluded that in practical cases, where $p^2 > n^{1/2}$, the latter algorithm has lower time complexity. It can be seen that this is also true for systems with wormhole routing.

The next theorem investigates the same problem for the MORB$^R(n, p)$ with separable buses. The switches present in the MORB$^R(n, p)$ can be taken advantage of for a closer match with the algorithm developed in Theorem 4 for the reconfigurable mesh.

THEOREM 5 *If data items are stored one per PE in the MORB^R (n, p), the maximum or minimum among the n^2 values can be found and stored in the $PE_{0,0}$ in time*

$$T^R_{max/min} = r\,t_{NEWS} + (r + 4 + (p+1)(1 + \lceil \log p \rceil))t_{ALU} + (p+1)(3 + \lceil log p \rceil)t_{bus}$$

under the UTD model. The t_{bus} term for the LTD model is

$$\left[4\,log(p!) - 4\,log\left(\left(\left\lceil \frac{p}{2} \right\rceil - 1\right)!\right) + (p+1)log\,p + (p+1)\sum_{i=1}^{\lceil logp \rceil} log(2^{i-1} + 1) + 2\right] t_{bus}.$$

Proof Without loss of generality, maximum in this proof stands for maximum or minimum. As in the proof of Theorem 4, first the smallest possible number of local maxima are found in the fastest possible way for blocks of the mesh, with these maxima finally stored in PEs attached to broadcast buses. The time for this phase is $t_1 = (r/2)$ $(t_{\text{NEWS}} + t_{\text{ALU}}) + t_{\text{ALU}}$. At this point, only the PEs attached to broadcast buses contain values which will be involved in the next phase of the algorithm. In the second phase, the values stored in the PEs attached to broadcast buses are combined by finding the maxima for subsets of PEs as in Theorem 4; NEWS connections are used for this purpose. At the end of this phase, only the PEs located at the intersections of broadcast buses contain values that remain to be combined. This phase takes time $t_2 = (r/2)$ $(t_{\text{NEWS}} + t_{\text{ALU}}) + 3t_{\text{ALU}}$. Therefore, in the third phase the algorithm for the reconfigurable mesh is used p times, once for each set of values stored on a row broadcast bus. The following analysis assumes that in the kth execution of the third phase the maximum of the values on the $(k-1)$th broadcast row is found and stored in the $\text{PE}_{0,\ (k-1)r}$, where $k = 1, 2, \ldots, p$. Thus, at the end of the third phase the PEs on row 0 are involved to find the maximum and store it in the $\text{PE}_{0,0}$, by executing the same algorithm once more. The iterative process in the third phase includes a (parallel) column broadcast, a row broadcast, an ALU comparison, an OR reduction operation on columns using binary tree emulations, and bus splitting on row 0. The third phase requires time $t_3 = (p + 1)$ $(3\,t_{\text{bus}} + t_{\text{ALU}} + \lceil \log p \rceil$ $(t_{\text{bus}} + t_{\text{ALU}>}))$ under the UTD model.

For the LTD model, the iterative process in the third phase requires time $(2\sum_{i=\lceil (p/2)\rceil}^{p} \log i -$

$[\log(p/2))t_{\text{bus}}$ or $(2\log(p!) - 2\log((\lceil(p/2)\rceil - 1)!) - \log p + 1)t_{\text{bus}}$ for the column broadcasts. It takes time $p\log p t_{\text{bus}} + p\ t_{\text{ALU}}$ for the row broadcasts and the subsequent comparisons. With binary tree emulation, the OR reduction on columns consumes time $p\lceil\log p\rceil t_{\text{ALU}} + p\sum_{i=1}^{\lceil\log p\rceil}\log(2^{i-1}+1)t_{\text{bus}}$. The bus splitting technique consumes the same amount of time as the column broadcasts. Finally, the end of the third phase that finds the maximum among the PEs on row 0 consumes time $3\log p t_{\text{bus}} + t_{\text{ALU}} + \lceil\log p\rceil t_{\text{ALU}} + \sum_{i=1}^{\lceil\log p\rceil}\log(2^{i-1}+1)t_{\text{bus}}$. Therefore, the time for the third phase under the LTD model is $t'_3 = [4\log(p!) - 4\log((\lceil p/2\rceil - 1)!) + (p+1)\log p + (p+1)\sum_{i=1}^{\lceil\log p\rceil}\log(2^{i-1}+1) + 2]t_{\text{bus}} + (p+1)(1+\lceil\log p\rceil)t_{\text{ALU}}$. Hence, the result. ■

Therefore, this operation consumes time $O((n/p) + p\log p)$ on the MORB$^R(n, p)$ under the UTD model. This performance is better than that of the MORB(n, p) in Theorem 4, which is $O((n/p) + p^2)$. It is the result of higher utilization of bus segments and increased number of parallel ALU operations. However, this new algorithm does not perform better in practical cases than the algorithm for the MORB$^R(n, p)$ in Corollary 4 that can solve the same problem in time $O((n/p) + \log p)$. The MORB$^R(n, p)$ without wormhole routing even performs better than the higher-cost MORB(n, p) with wormhole routing for this algorithm. The additional execution time of the latter is $(p + 1)(p - 1 - \lceil\log p\rceil)(t_{\text{ALU}} + t_{\text{bus}}) - M - r(M - 1)$, which is much larger than 0 in all practical cases (*e.g.*, for $p = 65$ and $t_{\text{ALU}} = t_{\text{bus}} = 1$, this difference is positive for all $M \leq 114$ packets). This result also shows that MORB^R meshes with separable buses do not necessarily require the higher-cost wormhole-routing technique to achieve very good performance.

V. ADVANCED ALGORITHMS

The effectiveness of MORB and MORB^R meshes in implementing efficiently prefix computations and graph connected-component labeling is investigated in this section. Both operations are very important in parallel processing.

A. Prefix Computation

For a prefix computation we are given n^2 inputs $a_1, a_2, \ldots, a_{n^2}$ and a binary associative operator $\otimes$. The n^2 outputs $b_1, b_2, \ldots, b_{n^2}$ are such that $b_k = a_1 \otimes a_2 \otimes \cdots \otimes a_k$, for any $1 \leq k \leq n^2$. This problem is fundamental in parallel processing and has everal applications [20, 28]. Hardwired implementation of prefix operations is very common for massively parallel computers.

An algorithm developed for the regular mesh without broadcast buses is modified here to make it suitable for MORB and MORB^R meshes. The next lemma is relevant.

LEMMA 1 *Given a set of n^2 values $a_1, a_2, \ldots, a_{n^2}$, distributed one per PE in the row-major order, and a binary associative operator $\otimes$, the parallel prefix problem for this operator can be solved on the regular mesh without broadcast buses in time $T_{mesh,\ prefix}(n) = 3Nt_{NEWS} + (2N + 1)t_{ALU}$.*

Proof The prefix computation problem is solved on the regular mesh without broadcast buses in three phases. In the first phase, the prefix computation is carried out sequentially from left to right on every row. This phase takes time $t_1 = N(t_{\text{NEWS}} + t_{\text{ALU}})$. In the second phase, the prefix computation is applied to the rightmost column sequentially, from top to bottom. This phase takes the same amount of time as the first phase (*i.e.*, $t_2 = t_1$). In the third phase, the content of the rightmost PE on every row $k - 1$ is sent to all PEs on row k, except for the rightmost PE, for $k = 1, 2, \ldots, N$, and the operator is applied to the contained and received values. The most efficient implementation of this phase consumes time $t_3 = N\ t_{\text{NEWS}} + t_{\text{ALU}}$. Hence, the result. ■

Therefore, the prefix computation problem consumes time $O(n)$ on the regular mesh. Wormhole routing improves the timing for the third step to $N + M + t_{\text{ALU}}$. The total time then becomes

$t^W_{\text{mesh,prefix}}(n) = (2N+1)(M + t_{\text{ALU}}) + N$. The next theorem deals with the MORB mesh.

THEOREM 6 *Given a set of n^2 values $a_1, a_2, \ldots, a_{n^2}$, distributed one per PE in the row-major order, and a binary associative operator $\otimes$, the parallel prefix problem for this operator can be solved on the MORB(n, p) in time*

$$\begin{aligned} T_{prefix}(n,p) =& 5(r-1)t_{NEWS} \\ &+ (3r+n+P)t_{ALU} \\ &+ (2r+N+P-1)t_{bus} \end{aligned}$$

under the UTD model. The t_{bus} term for LTD model is $(r(1 + \log P) + N + p-1)t_{bus}$.

Proof In the first phase, the prefix computation is carried out in parallel for all of the row segments that fall between any two consecutive column buses. That is, this phase groups together all PEs, on the same row i, with Cartesian coordinates $(i, r j_1 + j_2)$, where $0 \le i \le N$, $0 \le j_1 \le p-2$ and $1 \le j_2 \le r$. This phase involves left-to-right shifts within these row segments and requires time $t_1 = (r-1)(t_{\text{NEWS}} + t_{\text{ALU}})$ under both models. In the second phase, the following process is repeated sequentially in parallel from the leftmost to the rightmost smallest-sized squares formed by broadcast buses. The switches are set so that the r previously calculated partial prefix values which are located on the left side of such squares are transferred one-by-one to the PE each time that has the same x coordinate with the initial sender but lies on the opposite side of the corresponding square. The $\otimes$ operator is then applied each time to the received value and the contained partial prefix value. Therefore, this phase requires time $t_2 = P\, r(t_{\text{bus}} + t_{\text{ALU}})$ or $t_2 = N(t_{\text{bus}} + t_{\text{ALU}})$ under both models; the first term P represents the total number of iterations whereas r represents the total number of separate parallel data transfers in each iteration. In the third phase, using NEWS connections the partial prefix values stored in the PEs with addresses (i, rj_1) are sent sequentially to the PEs with addresses $(i, rj_1 + j_2)$, for all $0 \le i \le j_1 \le p-2$ and $1 \le j_2 \le r-1$ and the $\otimes$ operator is applied to the received value and the previous prefix value stored in each PE. The most efficient implementation consumes time $t_3 = (r-1)t_{\text{NEWS}} + t_{\text{ALU}}$. In the fourth phase, the prefix computation is carried out on the rightmost column in a top-down fashion. In a way similar to that for the first phase, the prefix computation is carried out in parallel for all of the segments that fall between two consecutive row buses. Then, partial prefix values are sent in a sequential top-down manner between PEs on the rightmost column attached to consecutive row buses and the $\otimes$ operator is applied. The fourth phase takes time $t_4 = t_1 P(t_{\text{bus}} + t_{\text{ALU}})$ or $t_4 = (r-1)(t_{\text{NEWS}} + t_{\text{ALU}}) + P(t_{\text{bus}} + t_{\text{ALU}})$, under both delay models. Time t_5 equal to t_3 is then needed in the fifth phase to perform prefix computations within the segments used in the fourth phase. In the sixth phase, the r prefix values at the rightmost side of each smallest-sized rightmost square of buses are broadcast one-by-one to all PEs at bus intersections which are also attached to the upper bus of each such referenced square. The column buses are then used to send these prefix values for rows to PEs attached to column buses so that the PE on row k receives the value coming from the rightmost PE. on row $k-1$. Then, the $\otimes$ operator is applied. This sixth phase takes time $t_6 = r\,(2 t_{\text{bus}} + t_{\text{ALU}}) - t_{\text{bus}}$ and $t_6^1 = r[(1+\log P)t_{\text{bus}} + t_{\text{ALU}}] - t_{\text{bus}}$ under the UTD and LTD models, respectively, assuming column-bus splits just above their intersections with row buses. The seventh phase is carried out similarly to the third phase, the only difference being that the value transmitted is the one received in the sixth phase. Therefore, this phase takes time t_7 equal to t_3. By summing up the times for the seven phases, we get the times stated in this theorem. ■

Therefore, the parallel prefix algorithm requires time $O(n)$ on the MORB(n, p), under the UTD model. This is also the asymptotic time for the regular mesh without broadcast buses. However, the constants in $T_{\text{prefix}}(n, p)$ are smaller. For

example, assume that $n = 513$, $p = 65$, $t_{\text{NEWS}} = 1$, $t_{\text{bus}} = 1$, and $t_{\text{ALU}} = 1$. Then, the speedup $(T_{\text{mesh,prefix}}(n)/T_{\text{prefix}}(n,p) = (3072/1227) = 2.5037$ is significant. Wormhole routing reduces the time for each of the steps 3, 5, and 7 from $(r-1)$ $t_{\text{NEWS}} + t_{\text{ALU}}$ to $r - 1 + M + t_{\text{ALU}}$. To derive the best performance of the MB(n) for this problem, we find the value of p that minimizes the execution time of the MORB(n, p). This value is $P_{\text{optimal}} = [1 + [(n-1)(5t_{\text{NEWS}} + 3t_{\text{ALU}} + 2t_{\text{bus}})/(t_{\text{ALU}} + t_{\text{bus}})]^{1/2}]$. Other types of data ordering may result in lower time complexity for prefix computations [24]. However, most often data are stored in the row-major order.

The algorithm for the MORB^R(n, p) follows.

THEOREM 7 *Given a set of n^2 values $a_1, a_2, \ldots, a_{n^2}$, distributed one per PE in the row-major order, and a binary associative operator $\otimes$, the parallel prefix problem for this operator can be solved on the MORB^R (n, p) in time*

$$\begin{aligned} T^R_{prefix}(n,p) =& 3(r-1)t_{NEWS} \\ &+ (r+2+(r+1)\lceil logp\rceil)t_{ALU} \\ &+ [(r+1)\lceil logp\rceil + r]t_{bus} \end{aligned}$$

under the UTD model. The t_{bus} term for the LTD model is $[r \log p + (r+1)\sum_{i=1}^{\lceil \log p\rceil} log(2^{i-1}+1)]t_{bus}$.

Proof In the first phase, the prefix computation is carried out in parallel for all of the row segments that fall between any two consecutive column buses. That is, this phase groups together all PEs on the same row i, with Cartesian coordinates $(i, rj_1 + j_2)$, where $0 \le i \le N$, $0 \le j_1 \le p-2$, and $1 \le j_2 \le r$. This phase involves left-to-right shifts within these row segments and requires time $t_1 = (r-1)$ $(t_{\text{NEWS}} + t_{\text{ALU}})$ under both models.

In the second phase, an algorithm that employs binary tree emulation is used [11]. Its implementation on the MORB$^{\text{R}}$(n, p) is carried out very efficiently due to the robustness supported by the switches. Let us focus on the p PEs which are attached to the same row broadcast bus. The first iteration combines pairs of partial prefix values coming from consecutive such PEs and stores for each such pair the result into the PE on the right side. The $\otimes$ operator is then applied to received and contained values. In the second iteration, every other PE that received a value in the previous iteration, starting with the leftmost such PE, sends its value to the two consecutive such PEs that follow on its right side. The $\otimes$ operator is then applied to received and contained values. Generally, in the ith iteration, every other PE which is the rightmost PE of a group of PEs that received the same value in the preceding iteration, starting with the leftmost such PE, sends its value to the 2^{i-1} consecutive PEs that follow on its right side. The $\otimes$ operator is then applied to received and contained values. A total of $\lceil \log p \rceil$ iterations are needed. Actually each iteration carries out the above process for all of the r PEs on each column bus segment. This phase consumes time $t_2 = r \lceil \log p \rceil (t_{\text{bus}} + t_{\text{ALU}})$ under the UTD model. For the LTD model the time is $t'_2 = r(t_{\text{ALU}} + \sum_{i=1}^{\lceil \log p \rceil} \log(2^{i-1}+1)t_{bus})$.

In the third phase, the values stored in PEs attached to column broadcast buses are shifted into the $r-1$ PEs on their right side and the $\otimes$ operator is applied each time. The most efficient implementation of this phase takes time $t_3 = (r-1)t_{\text{NEWS}} + t_{\text{ALU}}$. At this point the prefix computation is complete for all rows in the mesh. In the fourth phase, the prefix computation is carried out within the rightmost column. This phase takes time $t_4 = (t_2/r)$ and $t'_4 = (t'_2 = /r)$ under the UTD and LTD models, respectively. In the fifth phase, values are broadcast from the rightmost column within all rows and the $\otimes$ operator is then applied. This phase takes time $t_5 = r\, t_{\text{bus}} + t_{\text{ALU}} + t_3$ under the UTD model, and $t'_5 = r(\log p\, t_{bus} + t_{\text{ALU}}) + t_3$ under the LTD model. Hence, the result. ■

Therefore, this operation consumes time $O((n/p) \log p)$ on the MORB$^{\text{R}}$(n, p) under the UTD model, and the value of the speedup $(T_{\text{prefix}}(n,p)/T^R_{\text{prefix}}(n,p)) = O(p/\log p)$ shows the superiority of the MORB$^{\text{R}}$ mesh with separable buses.

Assume as earlier that $n = 513, p = 65$, $t_{\text{NEWS}} = 1$, $t_{\text{bus}} = 1$, and $t_{\text{ALU}} = 1$. Then, the values of the speedups $(T_{\text{mesh, prefix}}(n)/T^R_{\text{prefix}}(n,p)) = (3072/165) = 18.6181$ and $(T_{\text{prefix}}(n,p)/T^R_{\text{prefix}}(n,p)) = (1245/165) = 7.5455$ are extremely impressive. Figure 8 shows that the MORB$^R(n, p)$ without wormhole routing performs much better than the higher-cost MORB (n, p) with wormhole routing, for this algorithm; it is assumed that $t_{\text{bus}} = M$, $t_{\text{NEWS}} = M$, and $t_{\text{ALU}} = 1$.

Figure 9 shows the speedup $T^R_{\text{prefix}}(n,n)/T^R_{\text{prefix}}(n,p)$ for the comparison of the MORB$^R(n, p)$ with the higher-cost MB(n) mesh with broadcast buses; we assume that $t_{\text{ALU}} = 1$. The results show that the MORB$^R(n, p)$ performs better than the mesh with multiple broadcast MB(n) in all practical cases.

B. Graph Component Labeling

Algorithms for graph component labeling on the MORB(n, p) and MORB$^R(n, p)$ meshes are presented here. Assume an undirected graph $G(V, E)$, where V represents the set of vertices, with $|V| \leq n$ and E represents the set of edges. The graph is represented by its adjacency matrix A where the element a_{ij} is 1 if the vertices i and j are directly connected; otherwise, $a_{ij} = 0$. The connected component labeling algorithm assigns a label l_i to each vertex i, where l_i is the minimum of the vertex indices belonging to the same connected component with i. Its special case of 2D image connected component labeling is very popular [19]. Any two vertices in a connected component are connected through a path that employs one or more graph edges.

The graph component labeling algorithms presented here are based on an algorithm for the polymorphic torus [16]. A brief description of the latter algorithm for the $n \times n$ polymorphic torus follows. The algorithm uses two parallel variables, namely vrtx_num and label. The initialization sets $\text{vrtx\_num}_i 0 = i$ and $\text{vrtx\_num}_{0i} = i$ on the first row and the first column, and these values do not change during the execution of the algorithm. $\text{label}_{i0} = \text{label}_{0i}$ contain the label assigned to vertex i and are updated in each iteration. The initialization sets $\text{label}_{i0} = i$ and $\text{label}_{0i} = i$.

The main body of the algorithm implements a loop of $1 + \lceil \log n \rceil$ iterations. Each iteration executes two functions, namely *shortcut* and *hook*.

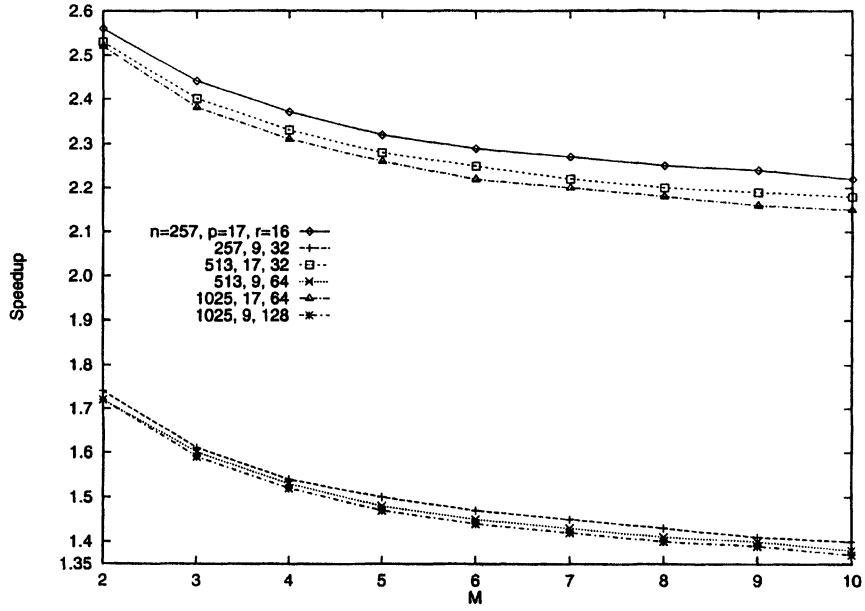

FIGURE 8 The speedup $T^W_{\text{prefix}}(n,p)/T^R_{\text{prefix}}(n,p)$, under the UTD model.

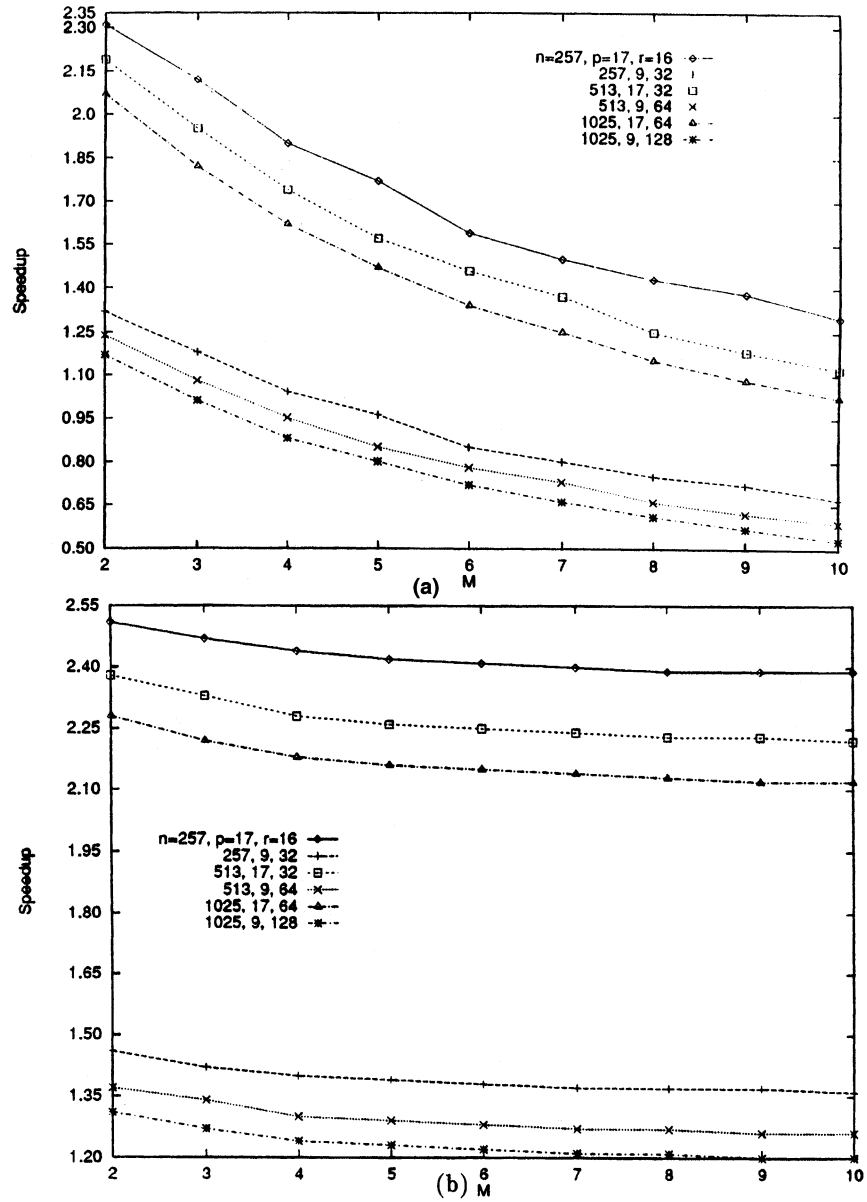

FIGURE 9 The speedup $T_{\text{prefix}}(n,p)/T^R_{\text{prefix}}(n,p)$, under the UTD model. (a) $t_{\text{bus}}=1$. (b) $t_{\text{bus}}=M$.

shortcut assigns as a label to each vertex i the current label of the vertex with index equal to i's label. *hook* assigns a new label to each vertex which is the minimum, over the set of vertices with the same label as vertex i, among the minima of the labels of the vertices connected to at least one such vertex. The implementation of these two functions is as follows. *Shortcut* broadcasts the labels from the first column to the east. Both the labels and the vertex indices are broadcast from the first row to the south. Labels broadcast from the first row are then sent to the west and copied into the variable *label* of PEs on the first column the label chosen for this copy operation comes from the leftmost

PE of each row that receives the same value for the vertex index broadcast from the first row and for the label broadcast from the first column. Finally, the labels are copied from the first column into the first row.

Each implementation of *hook* requires (label) column broadcasting from row 0, (label) row broadcasting from column 0, finding the minima within columns among labels received from the west by employing only the PEs that contain 1 for the adjacency matrix element (these minima are then stored in the PEs of the first row), broadcasting these minima from row 0 within the columns, finding the minima among just broadcast values within rows by choosing only the PEs that received in the beginning the same label from the north and west (these minima are then stored as labels in the PEs of column 0), and, finally, copying these labels from each PE_{i0} into the PE_{0i} (as in the last step of *shortcut*, the labels stored on the first column are copied into the first row). This graph component labeling algorithm requires time $O(\log^2 n)$ on the polymorphic torus.

THEOREM 8 *The graph component labeling problem can be solved on the MORB* (n, p) *in time*

$$T_{labeling}(n,p) = (1+\lceil \log n \rceil [3(r-1+2(n-p))t_{NEWS} + (3rp-1)t_{ALU} + 3(3r+N)t_{bus}]$$

under the UTD model. The t_{bus} *term for the LTD model is* $3\ r\ log(p^3 p^!)t_{bus}$.

Proof Assume a graph $G(V, E)$ with up to n vertices. Let row 0 and column 0 contain in the beginning the indices of the assigned vertices, while the adjacency matrix A is already stored in the $n \times n$ mesh, such that the PE_{ij} contains a_{ij}. Thus, there is no need for initialization. A loop of n iterations is carried out, similarly to the one described previously for the polymorphic torus.

Each implementation of *shortcut* requires a (parallel) column broadcast, a row broadcast (since the vertex index does not change, there is no need to broadcast it), a comparison in each PE of two values, an operation that finds the PE with the smallest index on each row for which the comparison produces affirmative result, an operation that copies the label from the latter PE of each row into the PE on the first column, and an operation that copies these labels from each PE_{i0} into the PE_{0i}. The column and row broadcasts, and the subsequent comparison are implemented in total time $t_1 = 2r\,t_{\text{bus}} + t_{\text{ALU}}$ and $t'_1 = 2r \log p\, t_{\text{bus}} + t_{\text{ALU}}$, under the UTD and LTD models, respectively. The operation that locates the PE with the smallest index on each row for which the comparison is affirmative, and copies the label from the latter PE into the corresponding PE on the first column consumes time $t_2 = T_{\text{rows,OR}}(n,p)$ and $t'_2 = T'_{\text{rows-OR}}(n,p)$, under the UTD and LTD models, respectively; we assume that the index of the source PE is attached to the transmitted label. In the last phase of *shortcut*, the labels stored on the first column are copied into the first row; the PE_{0i}, receives the label of the PE_{i0}, for $i = 1, 2, \ldots, N$. This phase is implemented as follows. Using broadcast buses, each PE_{i0} sends its label to the PE_{jj}, where $j = \lceil (i/r) \rceil \times r$. Then, broadcast buses are employed by each PE_{jj} in order to send the r values it received to the r distinct PEs $PE_{0,j-s}$, for $s = 0, 1, 2, r-1$. This process requires time $t_3 = 2\ r\ t_{\text{bus}}$ and $t'_3 = 2r \log p\, t_{\text{bus}}$, under the UTD and LTD models, respectively.

Ignoring temporarily the last column-to-row copy operation, there exist three broadcast operations on rows/columns, one internal PE comparison, and two operations to find the minimum within rows/columns. The three broadcast operations and the comparison require total time $t_4 = 3r\,t_{\text{bus}} + t_{\text{ALU}}$ and $t'_4 = 3r \log p\, t_{\text{bus}} = t_{\text{ALU}}$, under the UTD and LTD models, respectively. The two operations that find the minimum within individual rows/columns require total time $t_5 = 2\,T_{\text{rows\_OR}}(n,p)$ and $t'_5 = 2T'_{\text{rows-OR}}(n,p)$, under the UTD and LTD models, respectively. The last copy operation requires time $t_6 = t_3$ and $t'_6 = t'_3$, under the UTD and LTD models, respectively. Hence, the result. ■

Therefore, this algorithm consumes time $O(n \log n)$ on the MORB(n, p) mesh under the UTD model. With wormhole routing the term representing NEWS data transfers is replaced by 3 $(r-1+M)(1+2P)$.

THEOREM 9 *The graph component labeling problem can be solved on the MORB^R (n, p) in time*

$$T^R_{labeling}(n,p) = (1+\lceil \log n \rceil)[3(r-1)t_{NEWS} + (3r(1+\lceil log\ P \rceil)-1)t_{ALU} + 3r(3+\lceil logP \rceil)t_{bus}]$$

under the UTD model. The t_{bus} term for the LTD model is $3r(3\log p + \sum_{i=1}^{\lceil \log P \rceil} log(2^{i-1}+1))t_{bus}$.

Proof The same algorithm as in the proof of Theorem 8 is considered here. However, some steps in its implementation are altered to take advantage of the switches on separable buses. The implementation of *shortcut* requires two row/column (parallel) broadcast operations and a comparison that consume total time $t_1 = 2r\ t_{\text{bus}} + t_{\text{ALU}}$ and $t'_1 = 2r \log p\, t_{\text{bus}} + t_{\text{ALU}}$, under the UTD and LTD models, respectively. Also, the operation that copies into column 0 the labels of PEs with the smallest index on each row for which the comparison produces affirmative result consumes time $t_2 = T^R_{\text{rows-OR}}(n,p)$ and $t'_2 = T'^R_{\text{rows-OR}}(n,p)$, under the UTD and LTD models, respectively. Finally, the operation that copies the labels from column 0 into row 0 consumes time similar to that in the proof of the preceding theorem. More specifically, this process requires time $t_3 = 2\ r\ t_{\text{bus}}$ and $t'_3 = 2r \log p\, t_{bus}$, under the UTD and LTD models, respectively.

The implementation of *hook* requires three row/column broadcast operations and a comparison that consume total time $t_4 = 3\ r\ t_{\text{bus}} + t_{\text{ALU}}$ and $t'_4 = 3r \log p\, t_{\text{bus}} + t_{\text{ALU}}$, under the UTD and LTD models, respectively. The two operations that find the minimum among labels on each row/column consume total time $t_5 = 2T^R_{\text{rows-OR}}(n,p)$ and $t'_5 = 2T^R_{\text{rows-OR}}(n,p)$, under the UTD and LTD models, respectively. The last operation in *hook* consumes time $t_6 = t_3$ and $t'_6 = t'_3$, under the UTD and LTD models, respectively. Hence, the result. ■

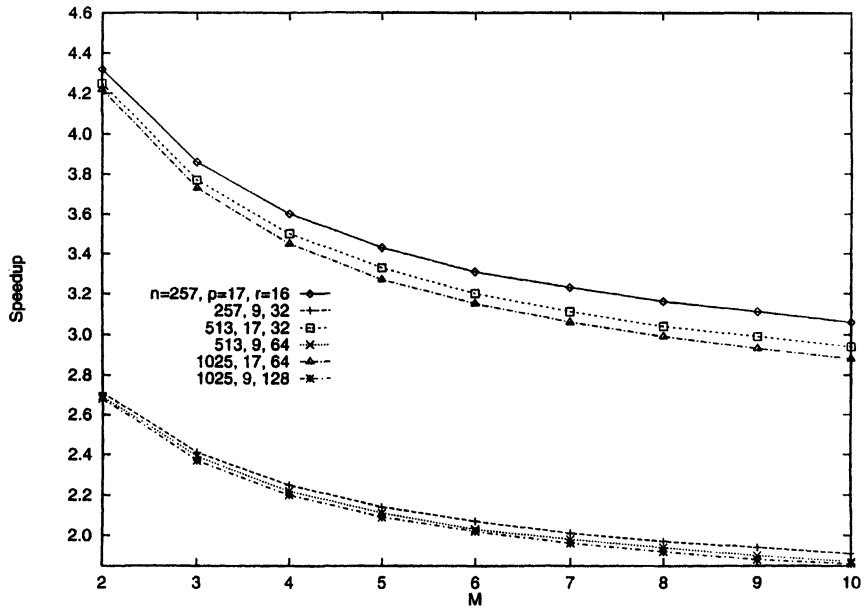

FIGURE 10 The speedup $T^W_{\text{labeling}}(n,p)/T^R_{\text{labeling}}(n,p)$, under the UTD model.

Therefore, this algorithm consumes time $O((n/p) \log p \log n)$ on the MORB$^R(n, p)$ under the UTD model. This results in a speedup of $O(p/\log p)$ when compared to the MORB(n, p) mesh for the same algorithm. For $n = 513$, $p = 65$, $t_{\text{NEWS}} = 1$, $t_{\text{bus}} = 1$, and $t_{\text{ALU}} = 1$, the value of the ratio $(T_{\text{labeling}}(n, p)/T^R_{\text{labeling}}(n, p)) = (64636/4444) = 14.5446$ illustrates the outstanding performance of the MORB^R mesh with separable buses. With wormhole routing the term representing NEWS data transfers is replaced by 3 $(r - 1 + M)$. Figure 10 shows that the MORB$^R(n, p)$ without wormhole routing performs much better than the higher-cost MORB(n, p) with wormhole routing, for this algorithm; it is assumed that $t_{\text{bus}} = M$, $t_{\text{NEWS}} = M$, and $t_{\text{ALU}} = 1$.

VI. CONCLUSIONS

Various mesh-connected architectures were investigated extensively in this paper. It was shown that low-cost architectures with sparse broadcast buses perform comparably to the high-cost mesh with multiple broadcast. In fact, such a low-cost architecture with separable buses was shown to often perform better than the latter mesh. This is a very important result because it shows that the viability and robustness of architectures with sparse broadcast buses can lead to successful parallel processing approaches. The comparisons were carried out for systems employing various routing switching techniques. Architectures with separable buses were shown to perform very well even without the high-cost wormhole-routing technique. To facilitate such comparisons, several operations and important algorithms were developed.

Acknowledgements

The work presented in this research was supported in part by the National Science Foundation under Grants ASC-9634775 (jointly with DARPA; co-sponsored by NASA) and DMI-9500260.

References

[1] Bhagavathi, D., Looges, P. J., Olariu, S., Schwing, J. L. and Zhang, J. (1994). "A Fast Selection Algorithm on Meshes with Multiple Broadcasting", *IEEE Trans. Paral. Distr. Systems*, **5**, 772–778.

[2] Ziavras, S. G. (1994). "RH: A Versatile Family of Reduced Hypercube Interconnection Networks", *IEEE Trans. Parallel Distrib. Systems*, **5**(11), 1210–1220.

[3] Bar-Noy, A. and Peleg, D. (1991). "Square Meshes are not always Optimal", *IEEE Trans. Comput.*, **40**(2), 196–204.

[4] Cuny, J. E. and Snyder, L. (1984). "Testing the Coordination Predicate", *IEEE Trans. Comput.*, **C-33**(3), 201–208.

[5] Li, H. and Maresca, M. (1989). "Polymorphic-Torus Architecture for Computer Vision", *IEEE Trans. Comput.*, **11**(3), 233–265.

[6] Carison, D. A. (1988). "Modified Mesh-Connected Parallel Computers", *IEEE Trans. Comput.*, **37**(10), 1315–1321.

[7] Ziavras, S. G. (1992). "On the Problem of Expanding Hypercube-Based Systems", *Journ. Paral. Distrib. Computing*, **16**(1), 41–53.

[8] Ziavras, S. G. (1994). "A Class of Scalable Architectures for High-Performance, Cost-Effective Parallel Computing", in *Proc. 6th IEEE Symp. Parallel Distr. Processing*, Dallas, TX, Oct. 26–29, pp. 162–169.

[9] Leighton, T. (1984). "Parallel Computation Using Meshes of Trees", in *Proc. 1983 Workshop on Graph-Theoretic Concepts in Computer Science*, Springer-Verlag, pp. 200–218.

[10] Miller, R., Prasanna-Kumar, V. K., Resis, D. I. and Stout, Q. F. (1993). "Parallel Computations on Reconfigurable Meshes", *IEEE Trans. Comput.*, **42**(6), 678–692.

[11] Leighton, F. T. (1992). *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*, Morgan Kaufmann Publ., San Mateo, CA.

[12] Bokhari, S. H. (1984). "Finding Maximum on an Array Processor with a Global Bus", *IEEE Trans. Comput.*, **C-33**, 133–139.

[13] Stout, Q. F. (1983). "Mesh-Connected Computers with Broadcasting", *IEEE Trans. Comput.*, **C-32**, 826–830.

[14] Weems, C. C., Levitan, S. P., Hanson, A. R., Riseman, E. M., Nash, J. G. and Shu, D. B. (1987). "The Image Understanding Architecture", *COINS Tech. Rep.*, 87–76, Univ. of MA, Amherst.

[15] Chen, Y. C., Chen, W. T., Chen, G. H. and Sheu, J. P. (1990). "Designing Efficient Parallel Algorithms on Mesh-Connected Computers with Multiple Broadcasting", *IEEE Trans. Paral. Distr. Systems*, **1**(2), 241–246.

[16] Maresca, M. (1993). "Polymorphic Processor Arrays", *IEEE Trans. Paral. Distr. Systems*, **4**(5), 490–506.

[17] Wang, B. F. and Chen, G. C. (1990). "Constant Time Algorithms for the Transitive Closure and Some Related Graph Problems on Processor Arrays with Reconfigurable Bus System", *IEEE Trans. Paral. Distr. Systems*, **1**(4), 500–507.

[18] Blevins, D. W., Davis, E. W., Heaton, R. A. and Reif, J. H. (1990). "BLITZEN: A Highly Integrated Massively Parallel Machine", *Journ. Paral. Distrib. Computing*, **8**(2), 150–160.

[19] Ziavras, S. G. (1993). "Connected Component Labeling on the BLITZEN Massively Parallel Processor", *Image Vision Computing Journ.*, **11**(10), 665–668.

[20] Reif, J. H. (Ed.), (1993). *Synthesis of Parallel Algorithms*, Morgan Kaufmann Publ., San Mateo, CA.

[21] Prasanna-Kumar, V. K. and Raghavendra, C. S. (1987). "Array Processor with Multiple Broadcasting", *Bourn. Paral. Distrib. Computing*, **4**(2), 173–190.
[22] Miller, R., Prasanna-Kumar, V. K., Reisis, D. I. and Stout, Q. F. (1988). "Meshes with Reconfigurable Buses", *MIT Conf. on Advanced Research in VLSI*, pp. 163– 178.
[23] Bhagavathi, D., Bokka, V. V., Gurla, H., Olariu, S., Schwing, J. L., Stojmenovic, I. and Zhang, J. (1995). "Time-Optimal Visibility-Related Algorithms on Meshes with Multiple Broadcasting", *IEEE Trans. Paral. Distr. Systems*, **6**(7), 687–703.
[24] Serrano, M. J. and Parhami, B. (1993). "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses", *IEEE Trans. Paral. Distr. Systems*, **4**(10), 1073–1080.
[25] Li, H. and Maresca, M. (1989). "Polymorphic-Torus Network", *IEEE Trans. Comput.*, **38**(9), 1345–1351.
[26] Maeba, T., Tatsumi, S. and Sugaya, M. (1990). "Algorithms for Finding Maximum and Selecting Median on a Processor Array with Separable Global Buses", *Electr. Commun. Japan*, Part 3, **73**(6), 39–47.
[27] MacKenzie, P. D. (1993). "A Separation Between Reconfigurable Mesh Models", in *Proc. Intern. Paral. Proces. Symp.*, pp. 84–88.
[28] Ziavras, S. G. and Mukherjee, A. (1996). "Data Broadcasting and Reduction, Prefix Computation, and Sorting on Reduced Hypercube Parallel Computers", *Parallel Computing*, **22**, 595–606.
[29] Bhagavathi, D., Gurla, H., Olariu, S., Schwing, J. L. and Zhang, J. (1996). "Square Meshes Are Not Optimal for Convex Hull Computation", *IEEE Trans. Paral. Distr. Systems*, **7**, 545–554.

Author's Biography

Sotirios G. Ziavras received the Diploma in EE from the National Technical University of Athens/Greece and the D.Sc. degree in Computer Science from George Washington University. He is currently an Associate Professor of Electrical and Computer Engineering, and Computer and Information Science at NJIT. In 1996 he was the principal investigator for an NSF/DARPA/NASA-funded New Millennium Computing Point Design project for PetaFLOPS computing. He is an Associate Editor of the Pattern Recognition journal. He is listed, among others, in Who's Who in Science and Engineering, Who's Who in America, Who's Who in the World, and Who's Who in the East. His research interests are parallel computer architectures and algorithms, advanced computer architecture, and computer vision. He is a member of the IEEE (Senior), Pattern Recognition Society, ACM, Greek Chamber of Engineers, and Eta Kappa Nu.