Fault Tolerance Properties of Mesh-Connected Parallel Computers with Separable Row/Column Buses

Behrooz Parhami

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

Abstract — *A two-dimensional processor array with separable row/column buses that are logically divisible into a number of local buses has proven quite effective for a wide class of parallel computations. I show how these separable buses, originally proposed for improved performance, can also be used to achieve tolerance to processor and link failures with minimal overhead for certain applications.*

Keywords — Adaptive routing, Fault-tolerant algorithms, Parallel processing, Reconfiguration, Resilient computation.

I. INTRODUCTION

A two-dimensional $m \times n$ mesh-connected computer, or 2-D MCC, consists of $N = mn$ processing elements (PEs) arranged in a grid, where horizontally and vertically adjacent processors are connected via local communication links. Because of their potential for dense VLSI realization and natural fit to many problems in matrix computations, modeling of physical systems, and image processing, 2-D MCCs have become popular and numerous researchers have studied their properties with respect to algorithm design/implementation and fault tolerance.

With respect to algorithm design/implementation, it has been shown how basic building-block computations (semigroup and prefix computations, sorting, selection, data routing) can be programmed through efficient (pipelined) use of the nearest-neighbor connections. It has also been observed that the main disadvantage of a standard mesh is that the number of steps required by any non-trivial algorithm is lower-bounded by its fairly large diameter, $\Omega(N^{1/2})$.

Consequently, several researchers have proposed 2-D MCC architectures augmented with broadcast buses as global communication mechanisms or with express channels that provide shorter bypass paths. One such proposal, which is of interest in this paper is that of separable row/column buses whereby PEs in one row (respectively, column) are connected to a broadcast mechanism that can be logically divided into a number of local buses through the use of PE-controlled switches. Such an architecture has been shown to support $O(\log N)$-time semigroup/prefix computation and $O(N^{1/8}\log^{3/2}N)$-time median selection [MAEB90].

Studies dealing with fault tolerance of MCCs have concentrated on methods that can salvage a smaller working mesh from a larger one with faulty PEs or links. In other words, an $m \times n$ rectangular mesh is converted to an $m' \times n'$ mesh (where $m' \leq m$ and $n' \leq n$) using embedded switching mechanisms [CHEA90]. The main reason for this focus has been that both main application areas, viz defect tolerance and systolic numerical computation, require the availability of a complete working mesh.

In this paper, I take a different approach and investigate how much fault tolerance can be achieved through regular mechanisms in MCCs that do not have special embedded switching mechanisms for fault tolerance. As an example, in MCCs with separable row/column buses [MAEB90], [SERR92], [SERR93], originally added to improve their performance for certain global computations, the buses can be used in lieu of failed or otherwise unavailable near-neighbor connections to achieve fault tolerance. My aim is to characterize the extent of fault tolerance offered by such inherent redundancies and to investigate the corresponding cost/performance/dependability tradeoffs.

II. INCOMPLETE BYPASS MESHES

I consider applications that do not need the full 2-D connectivity of 2-D MCCs for efficient handling but rather view the mesh as a logical 1-D array with better communication latency characteristics than a physical 1-D array. For example, if data is to be routed or sorted, it is not essential that a complete $m \times n$ mesh be available, as long as the faulty PEs can be detected, disabled, and avoided along data movement paths. As another example, certain geometric problems on point sets fall in this class.

A. Model and Assumptions

Each PE fails with a small probability p, independent of all other PEs. In the following logical model for analysis and algorithm development, I assume that any faulty PE can be bypassed by setting certain routing switches. I will show in Section III how this logical model can be realized physically. An MCC in which one or more faulty PEs have been bypassed is referred to as an incomplete (bypass) MCC (Figure 1). Connectivity is of course assumed. Due to the bypass connections, this last assumption is not a serious one in that an unrealistically large number of PE faults would be required to partition the array.

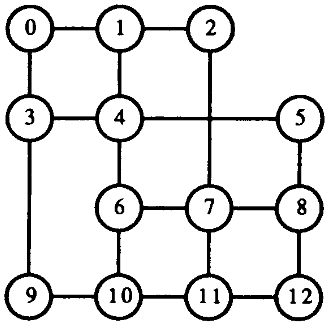

Figure 1. An incomplete bypass MCC.

Links do not fail in this logical model. This assumption is realistic in view of the implementations discussed in Section III. Furthermore, one can always incorporate the failure probabilities of 2 of the 4 nearest-neighbor links (e.g., the N and W links) into the PE failure probability and treat bypass failures as complete row/column failures. For the sake of generality (in case other implementations are proposed in future to support this logical model), row and column buses are not used in the algorithms. However, such buses can be used in the standard way to improve algorithm performance if desired.

B. Semigroup Computation

A semigroup computation is defined by $(\oplus, S)$, 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 $a_0 \oplus a_1 \oplus \ldots \oplus a_{N-1}$. Algorithms are available to perform such a semigroup computation on an $n \times n$ MCC in $O(n)$ steps.

For the sake of clarity of exposition, I assume that the upper-left PE is never faulty and that it is designated as the initial recipient of the semigroup computation result. This assumption can be easily relaxed.

I now show that an incomplete MCC can also perform a semigroup computation in $O(n)$ steps. Here is an outline of the algorithm. First, all PEs are set in the "horizontal" mode and the semigroup computation is perfomed in each row. Because the widths of various rows might differ (see Figure 1), each "row leader" waits until it receives a special endmarker generated by the rightmost boundary PE in that row. Then it enters the "vertical" mode. A PE in vertical mode sends its value to the north neighbor if it is in Column 0 and to some PE in Column 0 otherwise. The naive strategy of sending values up or to the left does not always work, as is clear for PE 2 in Figure 2.

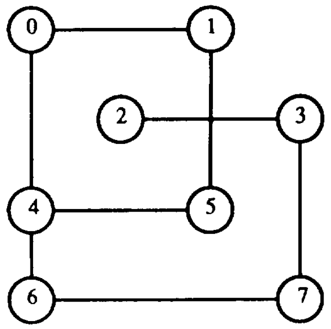

Figure 2. A degenerate incomplete bypass MCC.

One way around this problem is for each PE to have a tag showing the direction to move for getting to the closest PE in Column 0. For example, in Figure 2, the "Column- 0" tag for PEs 2 and 3 would be "E" and "S", respectively. This information is easily computed after each reconfiguration (via broadcast signals sent by all PEs in Column 0). It is easy to prove that the longest such path to Column 0 has length $O(n)$. Once the mesh leader has the final result, it broadcasts the value to all PEs. Each of the two phases is completed in $O(n)$ steps.

C. Parallel Prefix Computation

For parallel prefix computation, I use complete rows and columns, that exist in any large faulty MCC with near certainty, to derive an efficient $O(n)$-step algorithm.

Semigroup computation can be viewed as a special case of prefix computation defined as the simultaneous computation of $S_i = a_0 \oplus a_1 \oplus \ldots \oplus a_i$ for all i in the range $0 \le i \le N-1$. Let $S_{i,j} = a_i \oplus a_{i+1} \oplus \ldots \oplus a_j$, $i \le j$ be the partial 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 $0 < i_1 < i_2 < \ldots < i_M < j$.

To describe the algorithm, one needs to specify an ordering for the PEs and their associated values. My assumption is row-major ordering, starting from 0. The algorithm is to generate S_i in the PE initially holding a_i.

In the first phase of the algorithm, local prefixes for rows are computed. For example, in Figure 1, PEs 6, 7, and 8 would hold the values a_6, $a_6 \oplus a_7$, and $a_6 \oplus a_7 \oplus a_8$. In the second phase, each PE with no right neighbor must provide its value to all subsequent rows to be combined with the values held there. This is not as straightforward a step as in complete meshes. For example, PE 2 in Figure 1 has no direct connection to the next row. Also, step-by-step routing of values to the nearest PE in the next row is not viable unless the number of faulty PEs is bounded by a small constant. Figure 2 provides a good example where such routing would go from PE 1 to PE 3 ($d = 5$), from PE 3 to PE 4 ($d = 3$), and from PE 5 to PE 6 ($d = 2$), implying 10 steps overall.

With no topological restriction on the incomplete MCC (i.e. allowing degenerate cases such as the one depicted in Figure 2) and with PEs having $O(1)$ complexity, there appears to be no way for systematically sending the values in rightmost PEs of all rows downward while guaranteeing no duplicates and $O(n)$ running time. Thus the topology needs to be restricted.

A relatively minor condition that solves the above problem (and one that is satisfied in practice with near certainty) is the existence of at least one complete column having all of its n PEs fault-free.

prob{a given column is complete} = $(1-p)^n$
prob{a given column is incomplete} = $1-(1-p)^n$
prob{all n columns are incomplete} = $(1-(1-p)^n)^n$
prob{some column is complete} = $1-(1-(1-p)^n)^n$

For example with $p = 0.01$ and meshes of size 10×10 through 250×250, no complete column exists with probability of at most 10^{-9}. For $p = 0.001$, the probability of having no complete column is no greater than 10^{-16} for all practical mesh sizes (up to several billions of PEs). Thus the restriction is truly insignificant.

Now going back to the second phase of the algorithm, it is fairly easy to mark each PE in the rightmost complete column (the pivot column) as the "row pivot" PE. Then the following three subphases comprise Phase 2:

1. Sending the value in the rightmost PE of each row to the row pivot PE.
2. Sending the values down the pivot column while combining them.
3. Broadcasting combined values from each row pivot to all row elements.

Clearly the entire algorithm takes $O(n)$ steps. I observe that the algorithm could be modified to work with a pivot row and that the existence of a complete row or column also simplifies the semigroup algorithm. Whether or not efficient algorithms exist with milder restrictions is open.

D. Sorting and Selection

Sorting is an important operation in itself and also forms the basis for other data movement primitives. It is thus important that sorting be efficient. Sorting involves more extensive data movements compared to both semigroup and prefix computations. Thus, efficient sorting may require stronger topological constraints on the incomplete MCC.

A condition that allows us to sort efficiently is to have at least $n/2$ complete columns.

prob{any given column is complete} = $(1-p)^n = q$
prob{$\ge n/2$ columns complete} = $\sum_{j=n/2..n} \binom{n}{j} q^j (1-q)^{n-j}$

For example with $p = 0.01$ and a fairly small 8×8 mesh, four or more complete column exists with probability exceeding 0.9999. For $p = 0.001$ and meshes with 100 through 100,000 PEs, no complete column exists with probability of at most 10^{-9}.

A possible sorting algorithm works in three phases. First, all PEs in complete columns are marked and all row values are transferred to marked PEs. After this $O(n)$-step phase, each marked PE has one or two values. Second, sorting is done within the rectangular mesh consisting of the complete columns using any mesh sorting algorithm, suitably modified to deal with up to two values per PE and to guarantee that the number of values in each row does not change (this isn't as easy as it sounds). The third phase distributes the values to PEs in each row, taking into account the desired order (e.g., row-major or snakelike).

We have also adapted the shear-sort algorithm [SCHE89] to run directly on an incomplete MCC [PARH93]. Shear-sort works through alternating row and column sorts. The original proof of shear-sort is based on the zero-one principle and the fact that with each pair of row/column sorts, half of the remaining "dirty" rows (containing 0s *and* 1s) are "cleaned up" (contain only 0s *or* 1s). Clearly, this proof does not extend to an incomplete mesh in general.

We have shown that the number of dirty rows is reduced from D to $D/2 + 2X$ in each iteration, where X is the maximum number of faults among the odd- or even-numbered elements of a column [PARH93]. Hence, for reasonably small X, the number of dirty rows is reduced from n to some small quantity h after several iterations. Assuming the availability of end-around connections as in Figure 3, sorting can be completed through the application of hn steps of odd-even transposition along the overall snake. With X, and thus h, small, the $O(n \log n)$ time complexity of shear-sort is preserved.

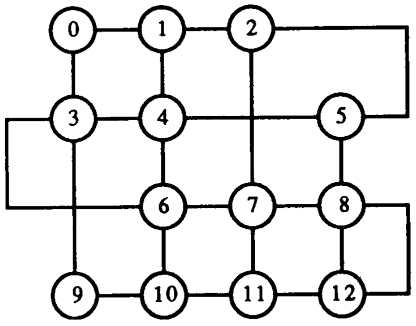

Figure 3. An incomplete MCC with end-around links.

Selection can be done either through sorting or by adapting existing selection algorithms for 2-D meshes to incomplete bypass MCCs. If complete columns are identified and used as in the case of sorting, adaptation of existing algorithms is straightforward. Direct implementation of selection algorithms for incomplete MCCs is now under study.

E. Data Routing

Data routing from a source PE at Row i, Column j, abbreviated as (i, j), to a destination PE at (i', j') can be accomplished in a variety of ways. Two issues must be considered: path length and path selection (routing). When faults render a mesh incomplete, some paths become shorter (e.g., 0 to 11 in Figure 1) while others become longer (e.g., 2 to 5 in Figure 1). This observation, along with a formal proof that the diameter of an incomplete $m \times n$ mesh has a tight upper bound of $m + n - 2$ [PARH93], suggest that path length is not a major issue.

As for routing, I have devised several strategies that are applicable to both packet routing and wormhole routing. One is based on complete rows and columns, of which many are likely to be available. At reconfiguration time, each PE will find out whether it is a member of complete row or column (2 flags) and if not, in which direction the closest complete row/column is located (2-bit tag). For example, in Figure 1, PE 10 will have both of its "complete" flags set and PE 8 will have its "nearest complete" tag point in the direction of PE 12. Routing is then accomplished by the following local decisions within each PE (choice of direction is implicit):

```
if data is already at the destination
then remove data
else if data is in the destination row/column
    then send along the row/column
    else if data is in a complete row/column
        then send along the row/column
        else send toward nearest complete row/column
        endif
    endif
endif
```

The above algorithm, like any adaptive routing algorithm [GAUG93], does not necessarily route data along a shortest path. However, the worst-case path length can be proven to be $2n - 2$ which is optimal. The actual performance is likely to be considerably better than the above upper bound. If δ is the worst-case distance to a closest complete row/column, then routing from (i, j) to (i', j') takes no more than $|i - i'| + |j - j'| + 2\delta$ steps.

III. REALIZING THE MODEL

In this section, I show how previously proposed separable row/column buses, with switches inserted between all adjacent PEs, can be used to physically realize the logical bypass model of Section II. I then propose a novel, more efficient, scheme for providing the needed bypass connections with separable buses.

A. Using Separable Buses

A separable row/column bus is a broadcast mechanism to which all PEs in a particular row/column are connected and which can be sectioned into several local buses through PE-controlled switches. If a switch is inserted after each PE, then any sectioning pattern is realizable. This is the ideal case for my fault tolerance scheme, although sparser switch placements are also possible.

Consider the column buses and assume that each PE can control the two switches to its north and south. Switches are assumed to be normally closed and failures are assumed to be benign, so that a faulty PE leaves its switch closed (note that this is a much less stringent requirement compared to the requirement that a faulty PE do something to establish the bypass paths). Finally, PE faults are assumed to be detectable by neighboring PEs.

To establish vertical bypass connections going north, a fault-free PE will open the switch to its south if its north neighbor is faulty. Similarly, the north switch is opened if the south neighbor is faulty. An obvious limitation is that both neighbors should not be faulty at the same time, since it is impossible for a PE to send to the north and receive from the south on the same bus. This isn't a serious limitation, since a PE surrounded by two faulty PEs can be assumed to have failed. After this, there is a vertical path on the column bus from any PE to the first fault-free PE to the north. Bypass connections in the other three directions are established similarly.

When a mesh algorithm calls for communication with the north neighbor, a sender whose north neighbor is faulty, puts the data on the column bus. Similarly, a receiver whose south neighbor is faulty, gets the requisite data from the column bus. Since, in algorithms devised with separable row/column buses, local links and buses are not used simultaneously, the above scheme imposes no performance penalty for previously proposed algorithms except for switch set-up times which are relatively small. Thus the performance and fault tolerance benefits of separable buses are simultaneously realized.

One issue of concern is whether the bypass connections realized through row/column buses can be operated at the same speed as the local, nearest-neighbor connections (otherwise, an additional slowdown factor will be introduced). Unlike row/column broadcasts, which may be slowed down by the large number of switches on any given bus, the delay through a bypass connection depends only on the number of switches between the source and destination PEs (same as the grid distance between the two). Even with highly pessimistic failure probabilities, it is virtually impossible to have such large failure clusters as to render the bypass delay unacceptable. Furthermore, in the extremely unlikely event that a row/column is affected by too many consecutive failures, the entire row/column can be bypassed to reduce the bypass delay (this can also have a positive effect on the performance of certain algorithms such as shear-sort).

B. A Novel Separable Bus Scheme

The scheme proposed above is somewhat inefficient in that it involves separate connections that are never used simultaneously under fault-free and fault conditions and requires each PE to control 4 switches (2 in each dimension). Figure 4 shows a novel architecture that merges the separable buses with local connections, thus reducing the number of ports per PE from 6 to 4 and the number of switches controlled by each PE from 4 to 2.

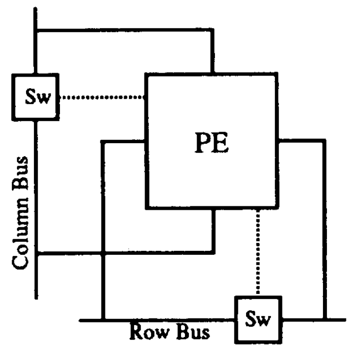

Figure 4. Proposed separable row/column buses.

When no fault is present, all switches are opened to allow for local communications. Switches are closed selectively to take advantage of the broadcast facility. With faults present, a scheme similar to that discussed earlier can be used to bypass faulty PEs horizontally or vertically. The main point is that no speed is lost due to the shared local and broadcast links because their usage is mutually exclusive. Switches are normally closed and must be opened temporarily to effect local communication. Thus, a PE that fails in a benign mode is permanently bypassed.

A distinct advantage of this new architecture over the previous one is that even an isolated PE, surrounded on all sides by faulty PEs, can be effectively utilized. This is due to the fact that each PE has two connections to the row and column buses and can thus send and receive on the same bus (with the switch open). A disadvantage of this scheme is that if a PE fails in such a way that one or both of its associated switches remain disconnected, an entire PE row and/or column may become unusable. It is possible, however, to use fail-safe design techniques to reduce the probability of such an event to an acceptably low level.

IV. CONCLUSION

I have presented a framework for the design of resilient algorithms that can run on complete as well as incomplete (bypass) MCCs and suggested two physical realizations for the assumed model. A key property of these algorithms is that they run with virtually no overhead on a fault-free MCC and with negligible penalty when the number of faults is small. Thus, both efficiency and graceful degradation are provided.

It has been shown that the use of buses has important performance implications for MCCs. Some of these carry over directly to my fault tolerance scheme. For example, both Subphase 1 and Subphase 3 of the second phase of the prefix algorithm discussed in Subsection II.C can be performed in 1 step instead of $O(n)$ steps if row/column buses can be utilized. It remains to be established if efficient fault-tolerant versions of algorithms that use the row/column buses for performance enhancement can be developed with this new architecture. However, one point is clear even now: the proposed scheme provides an architecture that can be used for high performance if there is no fault and for fault tolerance (perhaps at lower performance) in the presence of faults. The resulting performance dependability tradeoffs must be quantified.

The design of switches to establish the bypass connections is an important aspect of both of the proposed architectures. Similar switches have been successfully used in other architectures [MARE93]. Several alternatives are being examined in this respect.

Acknowledgement: This research was supported in part by the US National Science Foundation, Grant MIP-9001618.

REFERENCES

[CHEA90] Chean, M. and J.A.B. Fortes, "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays", *Computer*, Vol. 23, No. 1, pp. 55-69, Jan. 1990.

[GAUG93] Gaughan, P.T. & S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks", *Computer*, Vol. 26, pp. 12-23, May 1993.

[MAEB90] Maeba, T., S. Tatsumi, & M. Sugaya, "Algorithms for finding maximum and selecting median on a processor array with separable global buses", *Electronics and Communications in Japan*, Part 3, Vol. 73, No. 6, pp. 39-47, 1990.

[MARE93] Maresca, M., "Polymorphic Processor Arrays", *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, No. 5, pp. 490-506, May 1993.

[PARH93] Parhami, B. & C.Y. Hung, "Parallel Computation on Incomplete Meshes", Submitted for publication.

[SCHE89] Scherson, I.D. and S. Sen, "Parallel Sorting in Two-Dimensional VLSI Models of Computation", *IEEE Trans. Computers*, Vol. 38, No. 2, pp. 238-249, Feb. 1989.

[SERR92] Serrano, M.J. and B. Parhami, "Optimal Aspect Ratio and Number of Separable Row/Column Buses for Mesh-Connected Parallel Computers", *Proc. Int'l Parallel Processing Symp.*, Mar. 1992, pp. 343-347.

[SERR93] Serrano, M.J. and B. Parhami, "Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Global Buses", *IEEE Trans. Parallel and Distributed Systems*, to appear.