A Multi-Mode Fault-Tolerant Hypercube Multiprocessor

Baback A. Izadi
Department of Electrical and Computer Engineering
State University of New York
75 South Manheim Blvd.
New Paltz, NY 12561
bai@engr.newpaltz.edu
Phone: (914) 257-3823
FAX: (914) 257-3730

Füsun Özgüner
Department of Electrical Engineering
Ohio State University
2015 Neil Ave.
Columbus, OH 43210-1277
ozguner@ee.eng.ohio-state.edu
Phone: (614)-292-3039
FAX: (614)-292-7596

Keywords: real time, fault tolerance, hypercube, augmented multiprocessor, wave switching

Abstract

In this paper, we present a real-time fault-tolerant design for a d-dimensional hypercube multiprocessor with two modes of operations and examine its reconfigurability. The augmented hypercube, at stage one, has a spare node connected to each node of a subcube of dimension i, and the spare nodes are also connected as a $(d-i)$-dimensional hypercube. At stage two, the process is repeated by assigning one spare node to each $(d-i-j)$-dimensional spare subcube of stage one. We consider two modes of operations, one under heavy computation or hard deadline and the other under light computation or soft deadline. By utilizing the capabilities of wave-switching communication modules of the spare nodes, faulty nodes and faulty links can be tolerated. Both theoretical and experimental results are presented. Compared with other proposed schemes, our approach can tolerate significantly more faulty components with a low overhead and no performance degradation.

I. Introduction

As the size of the hypercube multicomputer grows the probability of node and/or link failures becomes high. A common way to sustain the same level of performance in the presence of faults has been to augment the hypercube with spare nodes and/or links to replace the faulty ones [14, 4, 8, 3, 5, 2, 1, 11, 6, 7, 10]. In a real-time fault-tolerant system, faulty components have to be replaced with spares in a manner that also satisfies the required completion deadline. Two modes of operation is generally considered: the *strict mode* and the *relaxed mode*. The strict mode pertains to tasks whose computational requirements are heavy or have a hard completion deadline. The relaxed mode, on the other hand, consists of tasks with a soft completion deadline or a light computational load. A real-time fault-tolerant system needs to replace faulty components with spares in a manner that the required computational load and/or completion deadline are also satisfied. Therefore, in the strict mode of operation, in order to allow for fast reconfiguration, spare replacement of faulty components should result in very few changes in the system interconnections. A common approach to accommodate this mode of operation is to replace each faulty component with the local spare using a distributed reconfiguration algorithm [16]. On the other hand, in the relaxed mode of operation, a global reconfiguration algorithm is applied to maximize the probability that in the next strict mode of operation, there exists a local spare for every faulty component.

In this paper, we present a two-stage redundant scheme for the hypercube. The objectives of the scheme are two fold. First, facilitate real-time fault tolerance by allowing the system to operate in either the strict mode or the relaxed mode. Second, utilize the spare network to tolerate a large number of faculty nodes and faulty links.

The rest of the paper is organized as follows. In the next section, notation and definitions that are used throughout the paper are given. An overview of our approach is presented in Section III. In Section IV, we examine the reconfigurability of the scheme. Both theoretical and simulation results are presented. Finally, concluding remarks are discussed in Section V.

II. Definitions and Notation

Each regular node of a d-dimensional hypercube is denoted by *d-tuple* $(b_{d-1} \cdots b_i \cdots b_0)$, where $b_i \in \{0, 1\}$. A subcube is defined by a unique *d-tuple* $\{0, 1, X\}^d$ where "0" and "1" are the *bound* coordinates, and "X" represents the free coordinates. A $(d-k)$-dimensional subcube is represented by a *d-tuple* with *k* bound coordinates and $(d-k)$ free coordinates. Each spare node is uniquely defined by *d-tuple* $\{0, 1, S\}^d$ where "0" and "1" represent the bound coordinates and S corresponds to the free coordinates of the spare node's assigned cluster. A regular link is specified uniquely by *d-tuple* $\{0, 1,\text{-}\}^d$ where "-" can be substituted by "0" or "1" to identify its connecting nodes. An intra-cluster spare link (a link connecting the spare node of stage one to a regular node of its cluster, or a link connecting the spare node of stage two to a spare node of its cluster at stage one) is defined by a $(d+1)$*-tuple* $\{0, 1, S\}^{(d+1)}$ where S is inserted to the left of the $(i-1)$th bit of the regular node ID, to which the spare node is connected. For example, the intra-cluster spare link connecting the spare node $00SS$ and node 0001 is identified as $00S01$. An inter-cluster spare link (a link connecting two spare nodes at either stage one or stage two) is defined by a *d-tuple* $\{0, 1, S,\text{-}\}^d$ where "-" can be substituted by "0" or "1" to identify its connecting spare nodes. Hence, the inter-cluster spare link connecting spare nodes $01SS$ and $00SS$ is labeled 0-SS. Finally, we define the *connection requirement (C_R)* of a spare node in a cluster with multiple faulty nodes as the number of edge-disjoint paths that must be constructed within the spare cube from that spare node to other spare nodes in the fault-free clusters so that faulty nodes can be tolerated.

III. Overview of the TECH

In our scheme, at stage one, the d-dimensional hypercube is divided into $2^{(d-i)}$ subcubes of dimension i; we call each of these subcubes a cluster. One spare node is assigned to each cluster; the spare node is connected to every regular node of its cluster via an intra-cluster spare link. Spare nodes are also interconnected to form a $(d-i)$-dimensional spare cube using inter-cluster spare links. We call the resultant structure the *enhanced cluster hypercube (ECH)* [14]. At stage two, the process is repeated: the spare nodes of stage one are also divided into $2^{(d-i-j)}$ clusters of dimension j and one spare node from stage two is assigned to each of these clusters. Moreover, the spare nodes at stage two are interconnected as a $(d-i-j)$-dimensional spare cube. We call the resultant structure the two-stage enhanced cluster hypercube (*TECH*). Figure 1 depicts a TECH of dimension $d = 4$ with $i = 2$ and $j = 1$. In the figure, the regular links are shown with heavy lines. The spare links at stage one and stage two are shown with light and dashed lines, respectively. Each regular node in a TECH is connected to its d neighboring regular nodes and the local spare node at stage one. Each spare node at stage one is connected to 2^i regular nodes of its local cluster, its assigned spare node at stage two, and its $(d-i)$ neighboring spare nodes at stage one. Each spare node at stage two is connected to 2^j spare nodes of its local cluster at stage one and its $(d-i-j)$ neighboring spare nodes at stage two. Therefore, the degree of each regular node, each spare node at stage one, and each spare node at stage two are $(d+1)$, $(2^i + d - i + 1)$, and $(2^{i-j} + d - i - j)$, respectively.

We next describe how the TECH tolerates faulty nodes and faulty links. Each node is made of a computation module and a wave-switching communication module [9]. We

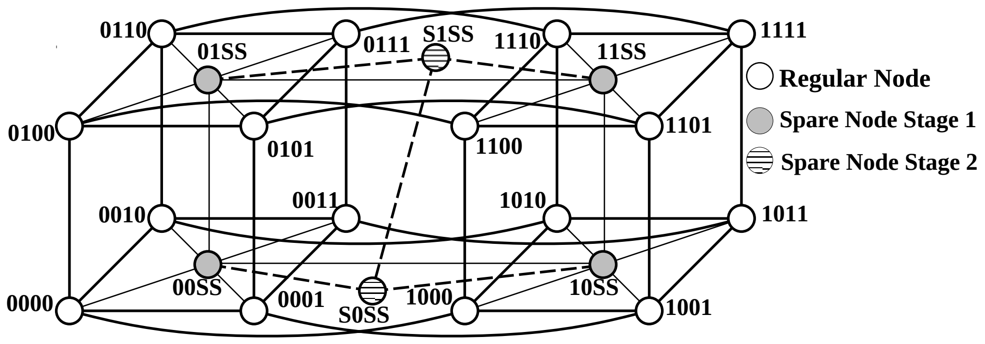

Figure 1: Two stage enhanced cluster hypercube of dimension 4

assume that faulty nodes retain their ability to communicate. This assumption may be avoided by duplicating the communication module in each node. To tolerate a faulty node, the computation module of the spare node logically replaces the computation module of the faulty node. In addition, if the spare node resides in the cluster of the faulty node, the new communication module consists of the functional communication module of the faulty node merged with the appropriate routine channel of the local spare node. If the assigned spare node and the faulty node belong to different clusters, a dedicated path is constructed by linking the appropriate routing channels of the intermediate spare nodes. Once such a path is established, due to the circuit-switched capability of the wave-switching communication modules, the physical location of the faulty node and its assigned spare node becomes irrelevant. Moreover, no modification of the available computation or communication algorithm is necessary. Similarly, faulty links are bypassed by establishing parallel paths using spare links. Figure 2 illustrates the reconfiguration of an TECH with $d = 4$, $i = 2$, and $j = 1$ in the presence of indicated faulty nodes and faulty links. Note that by utilizing spare nodes from other fault-free clusters, in effect, four logical spare nodes are present in cluster $11XX$. Figure 3 illustrates how spare nodes 01SS and 11SS replace faulty nodes 1100 and 1110, respectively.

IV. Reconfigurability of the TECH

To allow for fast reconfiguration in the strict mode of operation, the reconfiguration algorithm should result in minimum changes in system interconnections. Hence, under the strict mode of operation, each faulty node of a cluster is replaced by the local spare node at stage one. The algorithm is applied distributively, allowing each spare node at stage one to monitor the status of the regular nodes within its cluster, and replace the faculty node as outlined in the previous section. The reconfiguration algorithm under the strict mode of operation fails if more than one node becomes faculty in a cluster.

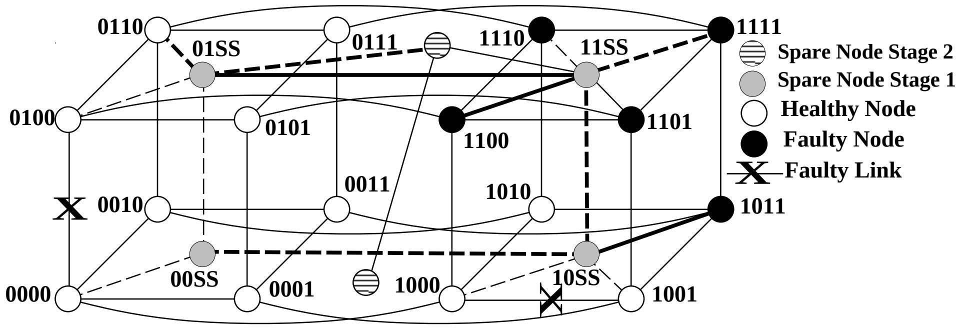

Figure 2: Reconfiguration of a TECH

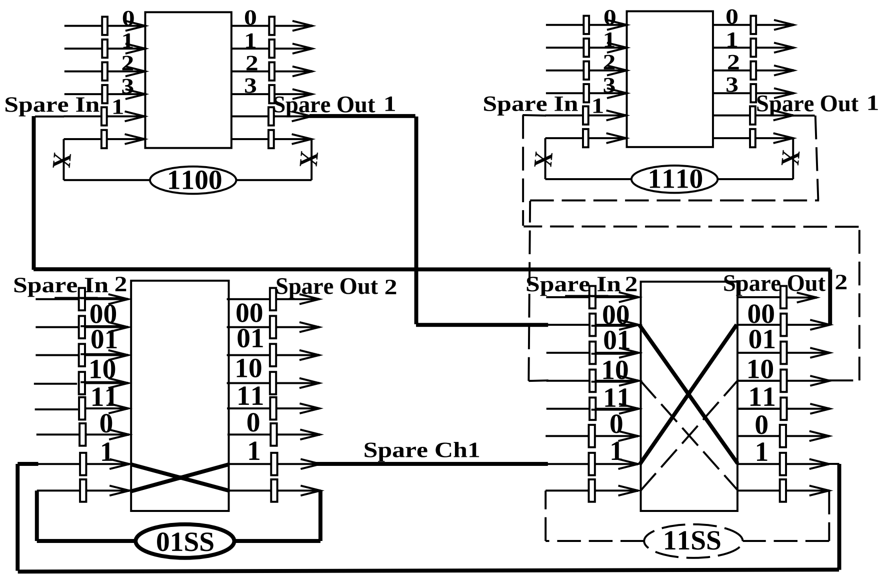

Figure 3: Replacing faulty nodes 1100 and 1110 with spare nodes 01SS and 11SS

The reconfiguration algorithm in the relaxed mode of operation assigns each detected faulty node to a spare node at stage two so that every healthy regular node would have an available spare node, at stage one, in its local cluster for possible reconfiguration in the next strict mode of operation. For example, in Figure 1, in the relaxed mode of operation, the task of faulty nodes 0110 and 0000 are assigned to spare nodes $S1SS$ and $S0SS$ while in the strict mode of operation, they would be assigned to the local spare nodes $01SS$ and $00SS$, respectively. Under the relaxed mode of operation, if there is no unassigned spare node at stage two, a spare node from stage one is assigned to the faulty node; details of reconfiguration algorithm under the relaxed mode of operation is discussed later in this section.

We next establish theoretical and simulation results pertaining to the relaxed mode of operation. From the previous section it follows that the reconfigurability of the TECH is a function of the number of dedicated and edge-disjoint paths, within the spare network at stages one and two, that can be established between the local spare node of a cluster with multiple faulty nodes and the available spare nodes in the fault-free clusters. The availability of these edge-disjoint paths is a connectivity issue within the spare network. The following theorems examine the connectivity of the spare network and establish bounds on the number of faulty nodes that a TECH can tolerate. Proof of theorems are omitted due to space limitation [12].

Theorem 1 *A d-dimensional TECH with cluster dimension of i at stage one can at most tolerate $(d - i + 2)$ faulty nodes in one cluster.*

■

Theorem 2 *A d-dimensional TECH with cluster dimension of i at stage on can tolerate $(d - i + 2)$ faulty nodes regardless of the fault distribution.*

■

Theorem 3 *A d-dimensional TECH with clusters of dimension i at the first stage and clusters of dimension j at the second stage can tolerate $2^{(d-i)} + 2^{(d-i-j)}$ faulty nodes regardless of the fault distribution provided that the maximum number of faulty nodes in each cluster of dimension i and $i + j$ is 4 and $3(2^j + 1)$, respectively.*

■

From Theorem 3 it follows that the reconfiguration of a TECH, under the maximum number of faulty nodes, is guaranteed provided that the number of edge-disjoint paths that must be initiated from each spare node at stages one and two be limited to three and two, respectively; the maximum C$_R$ of each spare node at stage one is three and at stage two is two. The result of Theorem 3 can be extended to a hypercube with multi-stage redundancies.

Theorem 4 *A d-dimensional enhanced cluster hypercube with k-stage redundancies can tolerate $\sum_{l=1}^{k} 2^{(d-\sum_{l=1}^{k} i_l)}$ faulty nodes provided that the maximum C_R of every spare node at stage k is 2 and every other spare node at other stages has a maximum C_R of 3.*

■

From our theoretical results, it follows that, in the relaxed mode of operation, some patterns of five faulty nodes per cluster can cause the reconfiguration of the TECH to fail. However, the probability that the faulty nodes can form such patterns is very low. Therefore, a more realistic measure of the reconfigurability of the TECH would be under random fault distributions. We next examine the simulation results based on the following reconfiguration algorithm. An optimal reconfiguration algorithm can be developed by utilizing the maxflow/mincut algorithm [14]. The main drawback to a reconfiguration using the above algorithm is that a digraph representation of the spare network has to be constructed [12] and the spare node assignment has to be done by the host processor. To overcome these deficiencies, we next present a near-optimal reconfiguration algorithm. The algorithm consists of four parts as specified below:

1. Early Abort: The following solvability checks are performed to determine whether the reconfiguration is feasible. If the total number of faulty nodes is greater than the number of spare nodes $(2^{(d-i)} + 2^{(d-i-j)})$, the reconfiguration fails. If the C$_R$ of a spare node at stage one is greater than $(d-i+1)$, the reconfiguration fails due to Theorem 1.

2. Assignment at Stage Two: We utilize Lee's path-finding algorithm [15] to find a set of candidate spare nodes at stage two that can be assigned to the faulty node. The algorithm begins by constructing a breadth-first search of minimum depth k $(1 \leq k \leq 2^{(d-i-j)} - 1)$ in the spare cube of stage two from the local spare node of a faulty cluster. If a free spare node is found, a path is formed to the faulty node. The algorithm guarantees that a path to a spare node will be found if one exits and the path will be the shortest possible [15]. Once a path is formed, the links associated with that path are deleted from the spare tree, resulting in a new structure. If there still remain some uncovered faulty nodes, a solvability test similar to step 1 is performed on the new structure and this step is repeated for a higher depth k in stage two of the spare cube.

3. Local Assignment: if all spare nodes at stage two are assigned and there still remain some faulty nodes, the local spare node of every faulty cluster is assigned to a faulty node within the cluster.

4. Assignment at Stage One: If there remain additional faulty nodes, we apply Lee's path-finding algorithm to both stages one and two from the local spare node of a faulty cluster with an unassigned faulty node. Reconfiguration fails if $k > 2^{(d-i)} + 2^{(d-i-j)}$, which is the longest acyclic path in the spare network. ■

We implemented the reconfiguration algorithm for a TECH with $d = 20$, $i = 10$, and $j = 3$. 1000 simulation runs were performed for each given number of faulty nodes. The result of our simulations, under the random fault distribution, indicate 100% reconfiguration in the presence of up to 1152 faulty nodes (the maximum). To compare the fault tolerant capability of the TECH with other schemes, we first simulated the reconfigurability of an

ECH (a TECH with spare nodes only at stage one) of $d = 20$ and $i = 10$; this is done since most fault-tolerant hypercube schemes in the literature have only one level of redundancy. Simulation result for up to 1024 randomly placed faulty nodes is shown in Figure 4 as plot G4. Plots G1, G2, and G3 in the figure pertain to the schemes proposed by [8, 13], [1], and [2], respectively; compared to other schemes in the literature, the selected ones tolerate more faulty nodes for similar hardware overhead. The result also indicates 100% reconfiguration of the ECH in the presence of up to maximum number of faulty nodes. We next

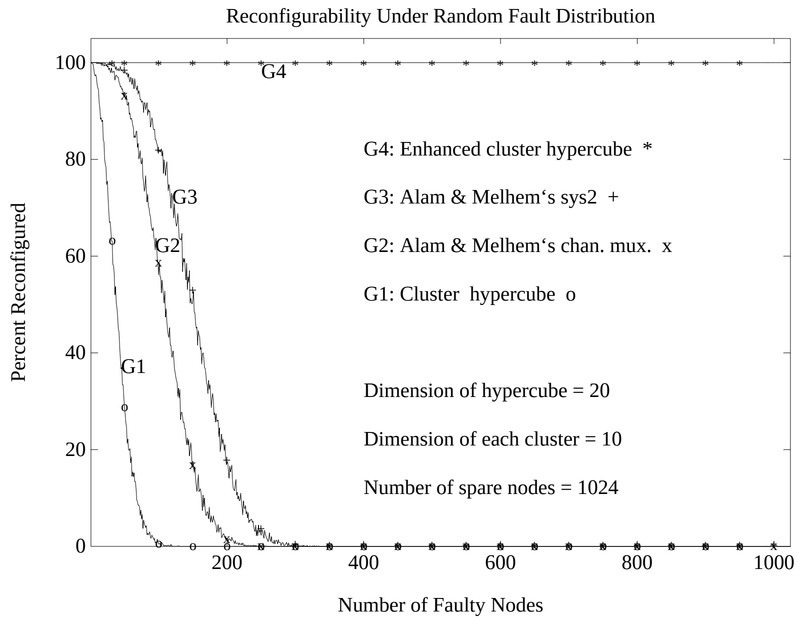

Figure 4: Reconfigurability of the ECH under random fault distribution

compared the reconfigurability of the TECH and the ECH under the maximum number of faulty nodes such that each cluster contains a fixed number of faulty nodes. Figure 5 depicts the simulation results for the hypercube of dimension 10; the solid and the dashed lines in the figure pertain to the result of the TECH under 1152 faulty nodes and the ECH under 1024 faulty nodes, respectively. The result indicates that, under the maximum number of faulty nodes, the TECH can handle one more faulty node per cluster than the ECH. The result is interesting since the degree of the spare node of the TECH at stage one is also higher than the ECH by one. To examine whether the same result would be attained under different dimension of spare cubes at stage one and stage two, simulation runs under the following spare cube dimensions were carried out. We chose the dimension of the spare cube for the ECH and the spare cube for the TECH at stage one to be 8. Furthermore, for the TECH, two different dimensions of the spare cubes at the second stage were ex-

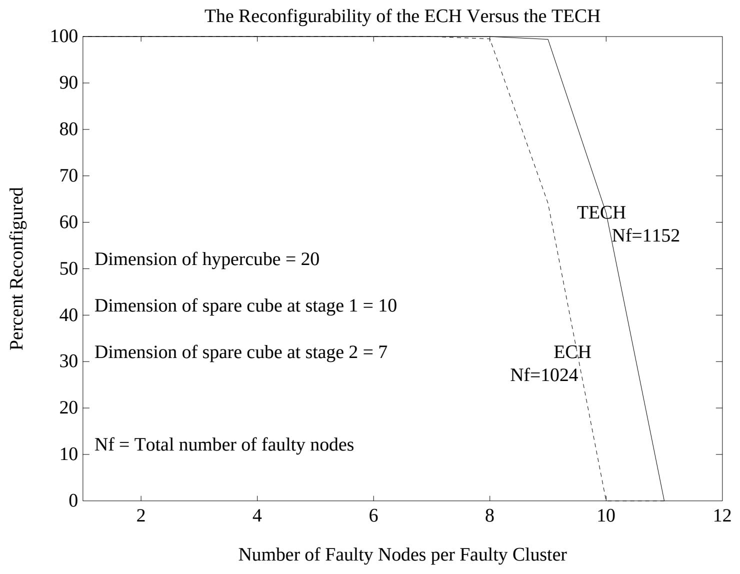

Figure 5: Reconfigurability of a TECH versus an ECH under random fault distribution

amined, one with a dimension 5 and the other with a dimension 4. The simulation results are shown in Figure 6. The results confirm that the TECH, under the maximum number of

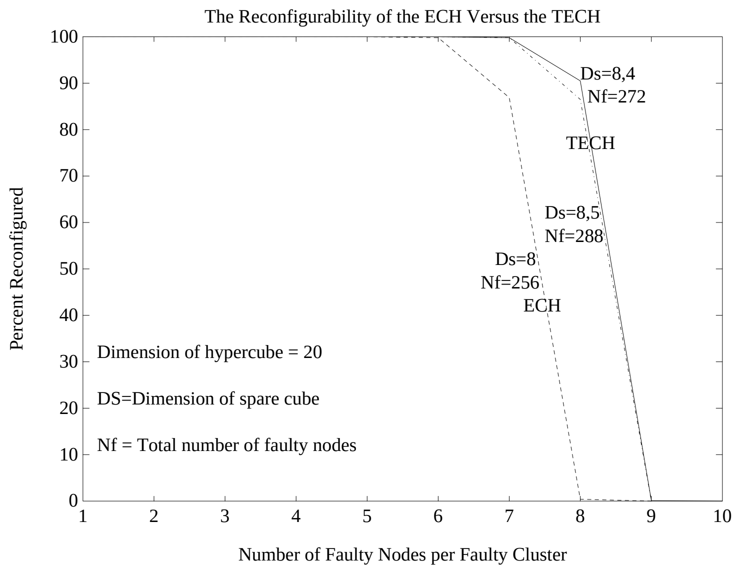

Figure 6: Reconfigurability of the TECH versus the ECH under random fault distribution

faulty nodes, can tolerate one more faulty node per cluster than the ECH, and therefore has a higher degree of reconfigurability. The results further illustrate that the reconfigurability of the TECH with a smaller spare cube at the second stage is slightly better. This is due to the fact that our simulation requires the TECH with a larger spare cube at the second stage to tolerate more faulty nodes. Simulation results of Figures 5 and 6 reveal that the existence of a second stage of redundancy is more critical to the higher reconfigurability of the TECH than the size of the spare cube at the second stage. Moreover, multi-stage redundancy should only marginally enhance the reconfigurability of the hypercube over the TECH since the degree of the spare node at stage one is ($d - i + 1$), regardless of the number of spare stages or their dimensions.

As indicated in the previous section, the TECH can also tolerate faulty links. However, no theoretical lower bound on the number of tolerated faulty links can be established, since two or more faulty links sharing a node in an TECH will cause the reconfiguration to fail. For example, in Figure 2, if the links 0-00 and 000- are faulty, the reconfiguration fails, since the spare link $00S00$ has to be used by two dedicated parallel paths to bypass the faulty links. Therefore, only simulation result can be examined. In addition, no distinction between the relaxed and the strict mode of operation can be made. In general, the simulation results based on random distribution of faulty links in the TECH is slightly better than in the ECH [14].

V. Conclusion

In this paper, we proposed a scheme to allow a hypercube multiprocessor tolerate faulty nodes in real time. During the strict mode of operation, the scheme uses local reconfiguration, which is the fastest and involves the fewest switch changes. Then, in the next relaxed mode of operation the tasks of local spare nodes, at stage one, are transferred to the spare nodes at the second stage by applying a global reconfiguration scheme. If a node becomes faulty during the relaxed mode of operation, the scheme tries to assign a spare node at stage two to replace it. This is done to maximize the probability that in the next strict mode of operation local spare nodes may be available to replace potential faulty nodes. Our theoretical results indicate that our scheme can always tolerate the maximum number of faulty nodes with up to four faulty nodes per cluster at stage one. Our experimental results suggest that, under random fault distribution, the maximum number of faulty nodes can be tolerated with a very high probability.

References

[1] M. Alam and R. Melhem. Channel multiplexing in modular fault tolerant multiprocessors. *Proceedings of the IEEE International Conference on Parallel Processing*, pp. I492–I496, 1991.

[2] M. Alam and R. Melhem. An efficient modular spare allocation scheme and its application to fault tolerant binary hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 2:117–126, January 1991.

[3] P. Banerjee. Strategies for reconfiguring hypercubes under faults. *Proceedings of the IEEE International Symposium on Fault Tolerant Computing*, pp. 210–217, 1990.

[4] P. Banerjee and M. Peercy. Design and evaluation of hardware strategies for reconfiguring hypercubes and meshes under faults. *IEEE Transactions on Computers,* 43:841–848, July 1994.

[5] P. Banerjee, J. Rahmeh, C. Stunkel, V. Nair, K. Roy, V. Balasubramanian, and J. Abraham. Algorithm-based fault tolerance on a hypercube multiprocessor. *IEEE Transactions on Computers*, 39:1132–1145, September 1990.

[6] J. Bruck, R. Cypher, and C. T. Ho. Efficient fault-tolerant mesh and hypercube architectures. *Proceedings of the 22nd Annual International Symposium on Fault Tolerant Computing*, pp. 162–169, July 1992.

[7] J. Bruck, R. Cypher, and C. T. Ho. Wildcard dimensions, coding theory and fault-tolerant meshes and hypercubes. *Proceedings of the 23nd Annual International Symposium on Fault Tolerant Computing*, pp. 260–267, July 1993.

[8] S. C. Chau and A. L. Liestman. A proposal for a fault-tolerant binary hypercubes architecture. *Proceedings of the IEEE International Symposium on Fault Tolerant Computing*, pp. 323–330, 1989.

[9] J. Duato, P. Lopez, and S. Yalamanchili. Deadlock- and livelock-free routing protocols for wave switching. In *Proceedings of the 11th International Parallel Processing Symposium*, pp. 570–577, April 1997.

[10] S. Dutt and J. P. Hayes. Designing fault-tolerant systems using automorphisms. *Journal of Parallel and Distributed Computing*, (12):249–268, 1991.

[11] J. P. Hayes. A graph model for fault-tolerant computing systems. *IEEE Transactions on Computers*, c-25(9):875–884, September 1976.

[12] B. Izadi. Design of fault-tolerant distributed memory multiprocessors. *Ph.D. thesis, the Ohio State University*, 1995.

[13] B. Izadi and F. Özgüner. Spare allocation and reconfiguration in a fault tolerant hypercube with direct connect capability. *Proceedings of the Sixth Conference on Distributed Memory Computing Conference*, pp. 711–714, April 1991.

[14] B. Izadi, F. Özgüner, and A. Acan. Highly fault-tolerant hypercube multicomputer. *IEE Proceedings on Computers and Digital Techniques,* 146(2):77–82, March 1999.

[15] C. Y. Lee. An algorithm for path connection and its applications. *IRE Transactions on Electronic Computers*, ec-10:346–365, 1961.

[16] R. Libeskind-Hadas, N. Shrivastava, R. Melhem, and C. Liu. Optimal reconfiguration algorithms for real-time fault-tolerant processor arrays. *IEEE Transactions on Parallel and Distributed Systems*, 6:498–510, May 1995.