A HIGHLY FAULT-TOLERANT CIRCUIT-SWITCHED HYPERCUBE

BABACK A. IZADI
Dept. of Elect. Eng. Tech.
DeVRY Inst. of Tech.
Columbus, Ohio 43209 USA
bai@devrycols.edu

FÜSUN ÖZGÜNER
Dept. of Elect. Eng.
The Ohio State University
Columbus, OH 43210 USA
ozguner@ee.eng.ohio-state.edu

ADNAN ACAN
Comp. Eng. Dept.
Eastern Mediterranean Univ.
Gazi Magosa, TRNC
Mersin 10, TURKEY

ABSTRACT

In this paper, we present a strongly fault-tolerant design for a d-dimensional hypercube multiprocessor and examine its reconfigurability. The augmented hypercube has a spare node connected to each node of an i-cube and the spare nodes are also connected as a $(d-i)$-dimensional hypercube. By utilizing the circuit-switched capabilities of the communication modules of the spare nodes, large number of 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 nodes and faulty links with a low overhead and no performance degradation.

KEYWORDS: Fault-tolerant multiprocessors, circuit-switched hypercube, reconfiguration.

1 INTRODUCTION

Multiprocessors based on the hypercube interconnection topology are being widely used for a range of scientific and real-time applications. As the size of the hypercube multicomputer grows, due to its complexity, the probability of node and/or link failures becomes high. Therefore, it is crucial that such systems be able to withstand a large number of faults for a reasonable amount of time.

Two main approaches to tolerate faults have been investigated. Some researchers have devised techniques where the healthy segment of the hypercube simulates the entire machine [12]; in practice however, these techniques normally result in a considerable performance degradation. To sustain the same level of performance, other researchers have investigated hardware schemes for the hypercube where spare nodes and/or links are used to replace the faulty ones. In the literature a reconfigurable system that retains the same service level, as well as keeping the same system topology after the occurrence of faults, is either referred to as a *strongly fault-tolerant* or a *structurally fault-tolerant* [7] system.

Two classes of hardware schemes have been proposed. Some researchers have examined local reconfiguration techniques where a spare node can only replace a faulty node within a given subset [5, 3, 2, 1]. A common drawback of these approaches is low utilization of spare nodes. Furthermore, they are not strongly fault tolerant. The second class of approaches uses global reconfiguration and is based on creating a supergraph of the hypercube such that if a node and its associated links are removed from the supergraph, the remaining graph would be isomorphic to the hypercube [8, 4, 7]. The disadvantage of these schemes is that the degree of each node becomes very large which makes their implementation impractical for any realistic k.

In this paper, we propose a global reconfiguration scheme that utilizes circuit-switched communication to make the hypercube strongly fault tolerant. In our scheme, a 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 an *enhanced cluster hypercube (ECH)*. Figure 1 depicts an ECH of dimension $d = 4$ with clusters of dimension $i = 2$. In the figure, the spare links are shown using lighter lines. By utilizing the circuit-switched communication modules of various spare nodes, edge-disjoint paths between multiple nodes of a cluster and multiple spare nodes can be made. For example in Figure 1, using the spare links, nodes 0100, 0110, and 0111 in cluster $01XX$ can be connected to the spare nodes $01SS$, $00SS$, and $11SS$ respectively without sharing any of the spare links. We use this property to show that multiple faulty nodes in a cluster can be tolerated. Faulty links are bypassed by establishing parallel paths using spare links. We present both theoretical and experimental results. Our

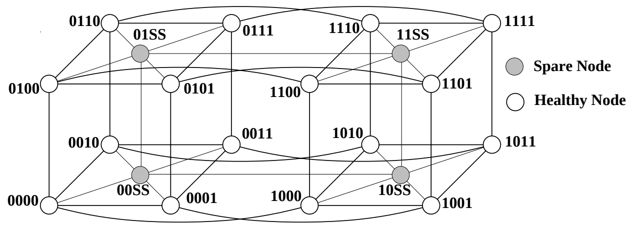

Figure 1. ENHANCED CLUSTER HYPERCUBE OF DIMENSION 4

theoretical results, representing the worst case scenario, indicate that the ECH can tolerate $2^{(d-i)}$ faulty nodes for up to 3 faulty nodes per cluster; proof of some of the theorems are omitted due to space limitation [9]. Our experimental results, based on random fault distribution, have yielded 100% fault coverage.

2 NOTATION AND DEFINITIONS

Each node of a d-dimensional hypercube is represented 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 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's assigned cluster. A link is specified uniquely by *d-tuple*$\{0, 1, \text{-}\}^d$ where "-" can be substituted by "0" and "1" to identify its connecting nodes. An intra-cluster spare link (a link connecting the spare node to a node within the cluster) 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 which the spare node is connected to. 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) is defined by a *d-tuple* $\{0, 1, S, \text{-}\}^d$ where "-" can be substituted by "0" and "1" to identify its connecting spare nodes. Hence, the inter-cluster spare link connecting spare nodes $01SS$ and $00SS$ is labeled 0-SS.

3 OVERVIEW OF THE ENHANCED CLUSTER APPROACH

In an ECH of dimension d and clusters of dimension i, the degree of each regular and spare node is $(d+1)$ and $2^i + (d-i)$ respectively. Note that for $i = 1$, the degree of both regular and spare nodes are $(d+1)$. The resultant topology has the advantage of using the same node architecture in both the regular and the spare nodes.

It is assumed that a given faulty node retains its communication capability. This is a common assumption since in hypercube multiprocessors such as the *iPSC/860* the computation and the communication modules of a node are separate. Furthermore, since the complexity of the computation module is much greater than that of the communication one, the probability of a failure in the computation module is much higher. This assumption may be avoided by duplicating the communication module in each node.

In hypercubes with circuit-switched communication modules, the cost of communication is nearly constant between any two given nodes. To facilitate reconfiguration, the routing channels described below are used at each node. The block diagrams of a regular node router and a spare node router for the 4-dimensional ECH of Figure 1 are depicted in Figure 2. The regular node router consists of $d + 1 = 5$ routing channels. The spare node router is made of $2^i + (d-i) = 6$ routing channels. Each routing channel consists of one channel in and one channel out. For

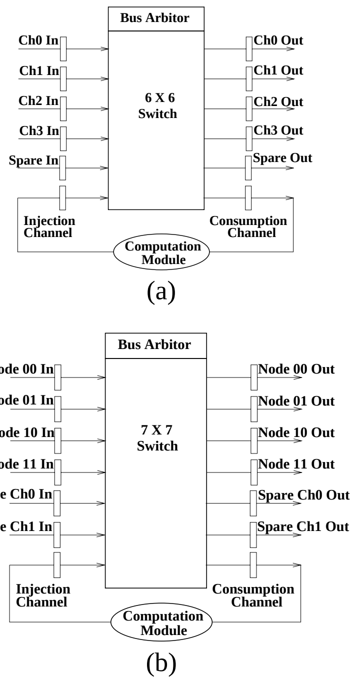

Figure 2. ARCHITECTURE OF A a) REGULAR NODE b) SPARE NODE

example in Figure 2(a), the Routing Channel 0 consists of Ch0 In and Ch0 Out allowing the regular node to be connected to its regular neighboring node in dimension 0. Similarly, the Spare Routing Channel consists of Spare In and Spare Out allowing the regular node to be linked to its local spare node. In Figure 2(b), the Spare Routing Channels connect the spare node to its neighboring spare nodes while the Node Routing Channels link the spare node to the regular nodes within its cluster.

To bypass a faulty link, it is necessary to connect the appropriate channels of the routing channels together. For example in Figure 2(b), connecting the routing channels of Node 00 and Node 01 (linking Node 00 In to Node 01 and Node 01 In to Node 00 Out) provides an alternate path to the intra-cluster link 010- in the subcube $01XX$ of Figure 1. Spare In, Spare Out (Figure 2(a)) of each router of nodes 0100 and 0101 are then connected to Node 00 Out, Node 00 In and Node 01 Out, Node 01 In of the spare node router. To provide an alternate path to an inter-cluster link, more than one spare node router is needed. For example, in Figure 1, the link 0-01 can be bypassed by the path $01S01 \rightarrow$ 0-$SS \rightarrow 00S01$ as follows. Node 0101 is connected to the spare node $01SS$ by linking node 0101's Spare In and Spare Out to the spare node $01SS$'s Node 01 Out and Node 01 In respectively. Similarly, node 0001 is connected to the spare node $00SS$. Also, the spare nodes $01SS$ and $00SS$ are interconnected by linking the spare node $01SS$'s Spare Ch0 Out and Spare Ch0 In to the spare node $00SS$'s Spare

Ch0 In and Spare Ch0 Out.

Connecting a spare node to a regular nodes is done to tolerate a node failure. If the spare node resides in the cluster of the faulty node, the Spare Routing Channel (Spare In and Spare Out) of the faulty node is connected to the appropriate Node Routing Channel of the spare node. If the assigned spare node and the faulty node belong to different clusters, a dedicated path to connect them needs to be established. Such a path can be constructed by linking the intermediate spare nodes of the spare hypercube. For example, in Figure 1, if the spare node $10SS$ is to replace the faulty node 0101, the appropriate channels of node 0101 to the spare node $01SS$, the spare node $01SS$ to the spare node $00SS$, and the spare node $00SS$ to the spare node $10SS$ needs to be interconnected. Once such a path is established, due to the capabilities of the circuit-switched routing modules, the physical location of the faulty node and its assigned spare node is irrelevant.

There are three cases which may involve routing data through a faulty node. The first one is when the message originates from the faulty node. Here, its spare replacement sends its data via its Injection Channel (Figure 2(b)) out to the appropriate Node XX Out channel. The message is then routed to the *communication module* (CM) of the faulty node via Spare In. Depending on the final destination node, any of Channel Outs may be selected. The second case is when the destination of the message is the faulty node. Once the data reaches the CM of the faulty node, it is automatically routed to the channel Spare Out instead of Consumption Channel. The spare node's CM would consequently receive the message using the appropriate channel (Node XX In). The message is then sent to the computation module of the spare node via Consumption Channel. The third case is when the faulty node is neither the source nor the destination of the message but is used merely as an intermediate switch connection. In this case, the spare node is not involved at all, and routing is done normally. Therefore, each spare node has dual functions. One is to be the logical replacement for a faulty node. The other is to be an intermediate switch connection. Both functions may be active at the same time. Note that the connection of the spare node router to the active node(s) is established during the reconfiguration and remains intact thereafter.

Upon detecting a node failure, the spare node within the respective cluster logically replaces the faulty node. If the local spare node is not available, an available spare node from a different cluster may be used. The spare node's CM then forwards messages bound for other spare nodes as well as forwarding/receiving its data to/from other nodes via the CM of the faulty node as discussed. Therefore, no modification to the available computation or communication algorithm is necessary.

Example: Figure 3 illustrates the reconfiguration of the ECH upon detection of faults in the links 0-00, 100- and nodes 1011, 1100, 1110, 1111. The spare nodes $10SS$, $01SS$, $11SS$, and $00SS$ are used to replace the faulty nodes 1011, 1100, 1110, and 1111 respectively. Note that by utilizing the intermediate spare nodes, in effect, 3 logical spare nodes are present in cluster $11XX$. Also, the intra-cluster faulty link 100- and the inter-cluster faulty link 0-00 are tolerated by establishing the indicated parallel paths.

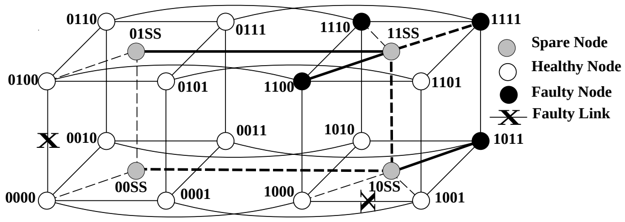

Figure 3. RECONFIGURATION OF AN ENHANCED CLUSTER HYPERCUBE

4 THEORETICAL RESULTS

Let's define a cluster with one or more faulty nodes as a faulty cluster. Since within a cluster, the local spare node is directly connected to every node, the number of edge-disjoint paths between the faulty nodes of a cluster and the unassigned spare nodes in other clusters is the same as the number of edge-disjoint paths between the local spare node of the faulty cluster and the unassigned spare nodes. The reconfigurability of the ECH is then a function of number of dedicated and edge-disjoint paths, within the spare hypercube, 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. We define the number of such edge-disjoint paths that must be constructed from a spare node as the *connection requirement* (C_R) of that spare node. For example in Figure 3, since 2 out of 3 logical spare nodes of cluster $11XX$ physically belong to other clusters, the C_R of the spare node $11SS$ is 2. Note that the C_R of a spare node is equal to the number of faulty nodes in its cluster minus one.

The availability of these edge-disjoint paths is a connectivity issue of the spare hypercube. The following theorems examine the connectivity of the spare cube and therefore the reconfigurability of the ECH. We first set the upper bound on the number of faulty nodes that can be tolerated in a cluster. We then examine the lower bound on the number of the faulty nodes that an ECH can tolerate for any fault distribution. Finally, we examine the conditions under which maximum number of faulty nodes can be tolerated.

Theorem 1 *The upper bound on the number of tolerated faulty nodes in a cluster that a d-dimensional ECH with cluster dimension of i can tolerate is $(d - i + 1)$.* ■

Theorem 2 *A d-dimensional ECH with cluster dimension of i can tolerate $(d - i + 1)$ faulty nodes regardless of the fault distribution.* ■

Let's group the spare nodes into three sets; S_S (set of source nodes), S_U (set of used nodes), and S_T (set of target nodes). A source node is a spare node in a cluster with multiple faulty nodes. The set S_S then represents the spare nodes with a C_R greater than 0. S_T is the set of unassigned spare nodes, and S_U consists of spare nodes that have been assigned to faulty nodes and have a C_R of 0. For example, considering only the faulty nodes in Figure 3, after assigning the local spare node to a local faulty node in each faulty cluster, $\mathrm{S}_S = \{11SS\}$, $\mathrm{S}_U = \{10SS\}$, and $\mathrm{S}_T = \{01SS, 00SS\}$. During the reconfiguration algorithm, which is discussed in Section 5, the spare nodes are dynamically assigned to the various sets. To illustrate this, suppose the C_R of a spare node $\alpha \in \mathrm{S}_S$ is greater than 0 and there is a dedicated path from α to $\beta \in \mathrm{S}_T$. Consequently, β replaces a faulty node in the cluster of α via the dedicated path. β is then called used and is assigned to S_U. Also, the C_R of α is reduced by one. If the C_R of α becomes zero, it is also marked as used and is assigned to S_U. The ECH is called reconfigured when S_S becomes an empty set.

As mentioned before, the reconfigurability of the ECH is a function of the number of dedicated and edge-disjoint paths, within the spare cube, that can be established between the local spare nodes (nodes in S_S) of the clusters with multiple faulty nodes and the available spare nodes (nodes in S_T) of the fault-free clusters. However, spare nodes do not have to be interconnected as a cube. Obviously, if the spare nodes are interconnected as a complete graph, the ECH can tolerate up to $2^{(d-i)}$ regardless of fault distribution. Hence, the reconfigurability of the ECH is a direct consequence of the connectivity of the topology which interconnects the spare nodes. Let's represent the topology of the graph connecting the spare nodes by $G = (V, E)$ where V=$\mathrm{S}_S \bigcup \mathrm{S}_U \bigcup \mathrm{S}_T$ and E consists of the appropriate spare links. Let the C_R of a node $n \in \mathrm{S}_S$ be represented by $\mathrm{C}_R(n)$ and denote sum of the C_R's of all nodes in a set P as $\sum_{n \in P} \mathrm{C}_R(n)$. Since the number of faulty nodes can't exceed the number of spare nodes, $|S_T| \geq \sum_{n \in S_S} C_R(n)$. The following theorem examines the connectivity of G as it pertains to the reconfigurability of the ECH.

Theorem 3 *Consider a graph $G(V, E)$ where $V = S_S \bigcup S_U \bigcup S_T$. The necessary and sufficient condition for every node $n \in S_S$ to have $C_R(n)$ edge-disjoint paths to $C_R(n)$ nodes in S_T, is that the minimum number of edges leaving any subset of nodes $P \subseteq V$ be greater than or equal to $\sum_{n \in (P \bigcap S_S)} C_R(n) - |P \bigcap S_T|$.*

Proof: We first prove the necessary condition: if from every node $n \in \mathrm{S}_S$, there exists $\mathrm{C}_R(n)$ edge-disjoint paths to $\mathrm{C}_R(n)$ nodes in S_T, then the minimum number of edges leaving any subset of nodes $P \subseteq V$ must be greater than or equal to $\sum_{n \in (P \bigcap S_S)} \mathrm{C}_R(n) - |P \cap \mathrm{S}_T|$ which is sum of the C_R's of S_S nodes within P minus the number of S_T nodes in P. Let's consider a subset $P_1 \subseteq \mathrm{S}_S$. Each of the edge-disjoint paths from a node in S_S to a node in S_T must be carried over at least one edge in the cutset (P_1,$V - P_1$). Therefore sum of the C_R's of the nodes in P_1, which represents the total required number of edge-disjoint paths from the nodes in P_1 to the nodes in S_T, must be smaller than or equal to the number of edges in the cutset (P_1,$V - P_1$). Now, let's consider a subset $P \subseteq V$ and denote the graph interconnecting the nodes of P as g. Obviously, g is a subgraph of G. Within g, there exists only $|P \bigcap \mathrm{S}_T|$ target nodes. Therefore, at most $|P \bigcap \mathrm{S}_T|$ of the edge-disjoint paths may exist in g. The rest of the paths must then be carried over the cutset $(P, V - P)$. Therefore, the necessary condition follows.

We next prove the sufficient condition: if the minimum number of edges leaving any subset of nodes $P \subseteq V$ is greater than or equal to $\sum_{n \in (P \bigcap S_S)} \mathrm{C}_R(n) - |P \bigcap \mathrm{S}_T|$, every node $n \in \mathrm{S}_S$ would have $\mathrm{C}_R(n)$ edge-disjoint paths to $\mathrm{C}_R(n)$ nodes in S_T. Let's create a new graph $G' = (V', E')$ by adding two nodes s and t to G as specified below and depicted by Figure 4. Each node in S_T is connected to t via a single edge. Each node $n \in \mathrm{S}_S$ is connected to s via $\mathrm{C}_R(n)$ parallel edges. Let sum of the C_R of all nodes in S_S be L. Number of edge-disjoint paths between s and t in G$'$, according to *Menger's* theorem [6], is equal to the size of the mincut in G'. We will show that there always exists an (s, t) mincut in G' whose size is equal to L. The mincut in G' may exist at s, t, G, or some combination of them. By construction, the size of the cut at s equals L. Similarly, the cutsize at t is greater than or equal to L since $|S_T| \geq \sum_{n \in S_S} \mathrm{C}_R(n) = \mathrm{L}$. Per stated condition, for $P = s \bigcup \mathrm{S}_S$ or $P = s \bigcup \mathrm{S}_S \bigcup \mathrm{S}_U$, the cut $(P, V' - P)$ must have a cutsize greater than or equal to L. Consider a general

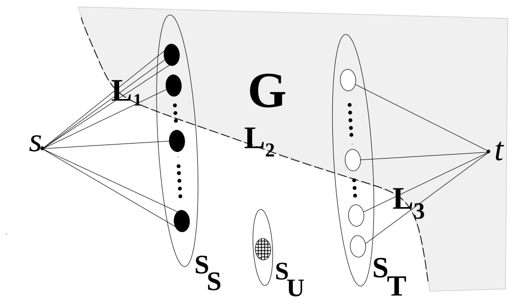

Figure 4. A CUT IN GRAPH G$'$

cut in G' crossing L_1 of the edges connecting s to S_S nodes, L_2 edges of G, and L_3 of the edges connecting S_T nodes to t (Figure 4). Number of S_T nodes on the unshaded side of the cut is L_3. Sum of the C_R's of S_S nodes within the same side of the cut is L$-$ L_1. Therefore, the stated condition can be formulated as $\mathrm{L}_2 \geq (\mathrm{L} - \mathrm{L}_1) - \mathrm{L}_3$ or $\mathrm{L}_1 + \mathrm{L}_2 + \mathrm{L}_3 \geq \mathrm{L}$. From this inequality, it follows that any cut in G' has a cutsize greater than or equal to L. Therefore, L is the size of the mincut. Hence, L edge-disjoint paths exist between nodes s and t. Then, each of the *s-t* edge-disjoint paths must pass through a unique node in S_T because each node in S_T is connected to t via a single edge. Since there only exists L edges from s (one per path), the number of edge-disjoint paths from s that passes through each node $n \in \mathrm{S}_S$ is equal to $\mathrm{C}_R(n)$. Therefore, each node $n \in \mathrm{S}_S$ can make $\mathrm{C}_R(n)$ edge-disjoint paths to $\mathrm{C}_R(n)$ distinct nodes in S_T. ■

We next apply Theorem 3 to the spare cube and examine the reconfigurability of the ECH. Theorems 4 and 5 set the bounds on the number of faulty nodes per cluster, under the maximum number of faulty nodes, that can be tolerated regardless of the fault distribution.

Theorem 4 *Clusters with 4 or more faulty nodes can cause the reconfiguration of a d-dimensional ECH to fail.*

Proof: We prove the theorem by showing that Theorem 3 does not hold for a distribution of 4 faulty nodes per cluster. Let S_S contain k_1 nodes with a C_R of 1, k_2 nodes with a C_R of 2, and k_3 nodes with a C_R of 3. Also, let S_T and S_U contain k_0 and k_u nodes respectively. Next consider a subset $P \subseteq V$ with $j_1 \leq k_1$ nodes with a C_R of 1, $j_2 \leq k_2$ nodes with a C_R of 2, $j_3 \leq k_3$ nodes with a C_R of 3, $j_0 \leq k_0$ unassigned spare nodes, and $j_u \leq k_u$ used nodes. Since the degree of each spare node within the spare cube is $(d-i)$, the number of links crossing P is $(d-i)j_3 + (d-i)j_2 + (d-i)j_1 + (d-i)j_0 + (d-i)j_u - l$ where l represents the number of internal links among the nodes in P. If an ECH is to tolerate up to 4 faulty nodes per cluster, by Theorem 3 it is necessary to show that $(d-i)(j_3+j_2+j_1+j_0+j_u) - l \geq 3j_3 + 2j_2 + j_1 - j_0$. Consider the case where $P = \mathrm{S}_S$ and every node has a C_R of 3, ($j_3 = k_3 = 2^{(d-i-2)}$, $j_2 = j_1 = j_0 = 0$). The above inequality can then be rewritten as $(d-i) * 2^{(d-i-2)} - l \geq 3 * 2^{(d-i-2)}$. The left side of the inequality represents the total number of links crossing P. When $2^{(d-i-2)}$ nodes form a subcube of dimension $(d-i-2)$, the number of links crossing P is $((d-i)-(d-i-2)) * 2^{(d-i-2)} = 2 * 2^{(d-i-2)}$. Therefore, $2 * 2^{(d-i-2)} \not\geq 3 * 2^{(d-i-2)}$ and hence by Theorem 3, the ECH may not be able to tolerate 4 faulty nodes per cluster. ■

Theorem 5 *The ECH can tolerate $2^{(d-i)}$ faulty nodes with up to 3 faulty nodes per cluster regardless of the fault distribution.* ■

5 EXPERIMENTAL RESULTS

As discussed above, some patterns of 4 faulty nodes per cluster can cause the reconfiguration of the ECH 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 ECH would be under random fault distributions.

We next examine the reconfiguration algorithm. An optimal reconfiguration algorithm can be developed by utilizing the maxflow/mincut algorithm [13]. Here, optimality is measured as the ability to assign a spare node to every faulty node whenever such an assignment is feasible vis-a-vis Theorem 3. The main drawback to reconfiguration using the maxflow/mincut algorithm is that a new digraph has to be constructed 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 which is called *Alloc-Spare*. The algorithm consists of three 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 $(2^{(d-i)})$, the reconfiguration fails. If the C_R of a spare node is greater than $(d-i)$, the reconfiguration fails due to Theorem 1. The reconfiguration also fails if sum of the C_R of any two neighboring spare nodes in the spare cube is greater than $2(d-i)-2$ (Theorem 3).
2. Local Assignment: The local spare node of every faulty cluster is assigned to a faulty node within the cluster. If all faulty nodes are covered, the ECH is reconfigured.
3. Non-Local Assignment: To find a set of candidate spare nodes that can be assigned to a faulty node, we utilize Lee's path-finding algorithm [11]. The algorithm begins by constructing a breadth-first search of minimum depth j ($1 \leq j \leq 2^{(d-i)} - 1$) in the spare cube from the local spare node of a faulty cluster with a non-zero C_R. If a free spare node is found, a path to the spare is formed. The algorithm guarantees that a path to a spare node will be found if there exits one and the path will be the shortest possible [11]. Therefore, all faulty nodes that are one link away from available spare nodes (at distance 1) are assigned first. Once a path is formed, the links associated with that path are deleted from the spare cube, resulting in a new structure. If there still remains some uncovered faulty nodes, a solvability test to check the C_R of neighboring spare nodes, similar to Early Abort, is performed on the new structure and Step 3 is repeated for a higher depth j. Reconfiguration fails if $j > (2^{(d-i)} - 1)$ which is the longest acyclic path in the spare cube. ■

We implemented algorithm Alloc-Spare for an ECH of dimension 20 with the spare cube of dimension 10. The simulation result for up to 1024 randomly placed faulty nodes is shown in Figure 5. 1000 simulation runs were performed for each given number of faulty nodes. The other plots in the figure pertain to the schemes proposed by [5, 10], [1], and [2] respectively. The result indicates 100% reconfigurability for the ECH under the random fault distribution.

To examine the limitation of the ECH under random fault distribution, we next assumed that the number of faulty nodes in the ECH is the maximum $(2^{(d-i)})$. We then assumed that each faulty cluster contains a fixed number of faulty nodes. Note that by Theorem 1, a faulty cluster may have up to $(d-i+1)$ faulty nodes. Under maximum number of faulty nodes, the number of faulty clusters is equal to the number of faulty nodes divided by the given number of faulty nodes per cluster. If the division results in a remainder, an additional cluster with the number of faulty nodes equal to the remainder needs to be allocated as well. The faulty clusters were then randomly allocated in the ECH and Alloc-Spare algorithm was applied to perform reconfiguration. The simulation results for an ECH of dimension 20 under different sizes of the spare cube are shown in Figure 6 where each point in the graph represents the average of 1000 simulation runs. Figure 6 indicates the percent number of cases where the ECH was reconfigured. The figure shows that under random fault distribution, an ECH with a large spare cube can nearly achieve 100% reconfiguration for up to $(d - i - 2)$ faulty nodes per cluster which is much

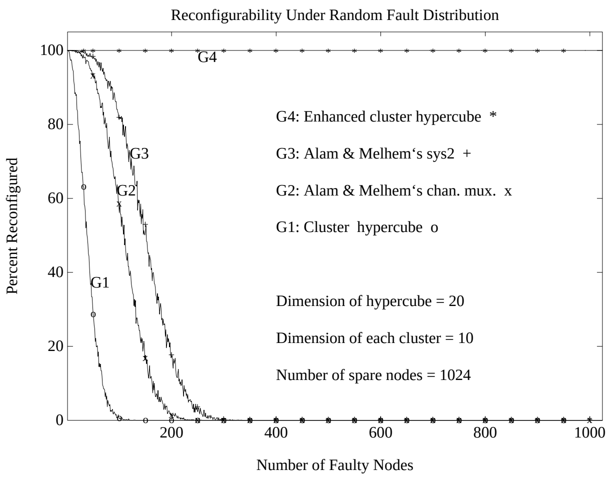

Figure 5. RECONFIGURABILITY UNDER RANDOM FAULT DISTRIBUTION

higher than the theoretical lower bound of 3 faulty nodes per cluster. Next we calculated the average number of

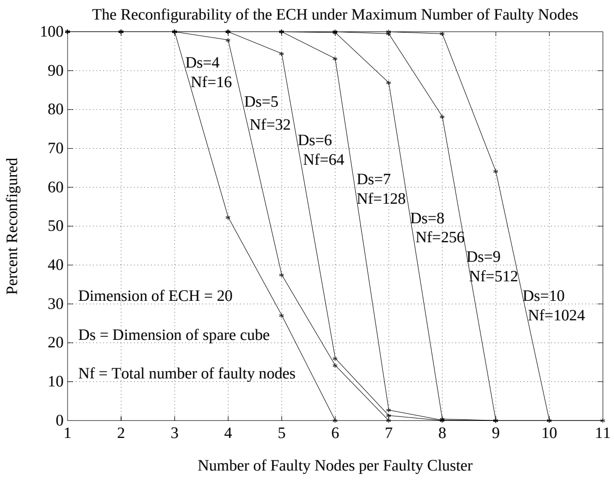

Figure 6. RECONFIGURATION OF THE ECH UNDER MAX. NUMBER OF FAULTY NODES

spare links of the spare cube per spare node that are left unused after the reconfiguration. Our results indicate that an ECH under the maximum number of faulty nodes and the highest number of reconfigurable faulty nodes per cluster, uses on the average less than 3 spare links per spare node to reconfigure. Therefore, the spare cube is a well connected graph via the unused spare links even after the reconfiguration. For the case depicted in Figure 5, the average number of utilized spare links per spare node is nearly 1. Hence, the following conclusion can be made. The ECH can tolerate a large number of faulty nodes with a very high probability. Since fast reconfiguration is very important in presence of faults, our near-optimal reconfiguration algorithm is more appropriate than the optimal one with negligible difference in the end result.

6 CONCLUSION

In this paper, we have presented a strongly fault-tolerant design for a d-dimensional hypercube multiprocessor and examined its reconfigurability. Theoretical results indicate that our scheme can always tolerate the maximum number of faulty nodes with up to 3 faulty nodes per cluster. Our experimental results suggest that the maximum number of faulty nodes can be tolerated. Although our scheme is global in the sense that any spare node can replace any faulty node, our reconfiguration algorithm can be implemented in a distributed manner[9]. By combining the reconfiguration algorithms for faulty nodes and faulty links, combinations of node and link failures are tolerated. Compared to other proposed schemes, the ECH can tolerate significantly more faults for the same overhead.

REFERENCES

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

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

[3] 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.

[4] 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*, pages 162–169, July 1992.

[5] 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*, pages 323–330, 1989.

[6] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.

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

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

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

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

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

[12] C. C. Li and W. K. Fuchs. Graceful degradation on hypercube multiprocessors using data redistribution. In *Proceedings of the Fifth Conference on Hypercube Concurrent Computers and Applications*, pages 1446–1454, April 1990.

[13] A. Tucker. *Applied Combinatorics 2nd ed.* Wiley, 1984.