The Journal of Supercomputing, 20, 203–216, 2001

A Low-Cost Fault-Tolerant Structure for the Hypercube

DAJIN WANG e-mail: wang@pegasus.montclair.edu

State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing 210093, China; and Department of Computer Science, Montclair State University, Upper Montclair, New Jersey 07043

Abstract. We propose a new, low-cost fault-tolerant structure for the hypercube that employs spare processors and extra links. The target of the proposed structure is to fully tolerate the first faulty node, no matter where it occurs, and "almost fully" tolerate the second, meaning that the underlying hypercube topology can be resumed if the second faulty node occurs at most locations—expectantly 92% of locations. The unique features of our structure are that (1) it utilizes the unused extra link-ports in the processor nodes of the hypercube to obtain the proposed topology, so that minimum extra hardware is needed in constructing the fault-tolerant structure and (2) the structure's node-degrees are low as desired—the primary and spare nodes all have node-degrees of $n + 2$ for an n-dimensional hypercube. The number of spare nodes is one fourth of primary nodes. The reconfiguration algorithm in the presence of faults is elegant and efficient. The proposed structure also effectively enhances the diagnosability of the hypercube system. It is shown that the diagnosability of the structure is increased to $n + 2$, whereas an ordinary n-dimensional hypercube has diagnosability n.

Keywords: diagnosability, fault tolerance, hypercubes, interconnection networks, redundant systems

I. Introduction

The hypercube is one of the most investigated interconnection networks for multicomputer systems. It outperforms many of its counterparts in terms of regularity, computational power, communication ability measured by such factors as internode distance, diameter, traffic density and so on. An extensively studied topic about hypercube-structured system is its reliability—the ability to perform its intended tasks in the presence of faulty processors. When some processor nodes in the system become faulty, one of the two general approaches will be taken—without or with spare processors.

Many fault-tolerant schemes take the first approach, which is to manage to carry out the original task without using the faulty nodes and the communication links connected to these nodes. The reconfigured network using this approach will result in a different network, i.e., an "incomplete" hypercube. The original underlying topology is not preserved. While it may not affect the computational abilities for some applications, it can be expected that its performance will be discounted as the result of losing its original topology. Most work on hypercube tolerance concentrate on fault-tolerant routing, since routing is among the most basic issues for multicomputers. Recent work on fault-tolerant hypercube routing includes [9–11, 19]. In [9], a concept called *routing capability* was introduced, which is basically the information

a node maintains about how far it is from faulty nodes in the hypercube. A routing algorithm was presented that makes use of routing capability to facilitate efficient fault-avoiding routing of messages. Also for the purpose of fault-tolerant routing, Kaneko and Ito [11] proposed the notion of *full reachability*. A fully reachable node is a node that can be reached by all fault-free nodes of Hamming distance h via a path of length h. Based on nodes' full reachability, a fault-tolerant routing algorithm was developed, which improved the ones proposed earlier by Chiu and Wu in [10]. The *safety vector* notion proposed in [19] effectively includes faulty link information and provides more accurate information about the distribution of faults in a hypercube. Each node in a hypercube is associated with a bit vector, called a safety vector, calculated by information exchange among neighboring nodes. Routing algorithm using safety vector was presented, so that an optimal routing between two nodes can be guaranteed if the kth bit of the safety vector of the source node is set, where k is the Hamming distance between the source and destination nodes.

All preceding referenced schemes assume that the systems have no spare processors to replace faulty ones. The alternative fault-tolerance approach is to add spare nodes and links to the original system. When faulty nodes occur, the spares will be activated to replace the faulty nodes in such a way that the original topology can be completely preserved. Such a fault-tolerant system is said to be *strongly* fault-tolerant. Although sparing the system costs more, it is a preferable choice if the preservation of the original structure topology and computational power are a crucial consideration. In [20], Yang et al. presented an interesting strongly fault-tolerant hypercube architecture, which is constructed by interconnecting a set of basic fault-tolerant modules (FTM's). Each FTM is composed of a subhypercube with a fixed number of spare nodes, links, and some reconfiguration switches. The FTM's are interconnected in such a way that the spare nodes cannot only replace faults in their own FTM, but can also replace the faults in other FTM's. The system has been shown to be robust and outperform previously proposed modular fault-tolerant hypercube systems. However, the fault-tolerant architecture proposed in [20] needs specially fabricated FTM's. Earlier, a strong fault-tolerant scheme was proposed in [5]. But the work was mainly concentrating on fault-tolerant meshes.

In this paper, we propose a fault-tolerant hypercube that needs very little hardware support and achieves satisfactory strong fault-tolerance. The uniqueness of this structure is that it makes use of the unused extra link-ports in the processor nodes of the hypercube to obtain the proposed topology so that (1) minimum extra hardware is needed in constructing the fault-tolerant structure, and (2) the structure's node-degrees are very low: for an n-dimensional hypercube, the proposed topology requires a node-degree of $n + 2$. In real systems using hypercube topology, the processors are usually made with the maximum allowable link-ports for scalability. It is quite often that not all these ports will be used in the hypercube of the system. This motivates the idea of utilizing the unused ports to improve the system's performance. Extra links between nodes can be introduced. It has been shown that such *enhanced* hypercubes can achieve considerable improvements over regular hypercubes in many measurements such as mean internode distance, diameter, traffic density [17], and diagnosability [18]. Our proposed structure takes advantage of these "left-over" ports to achieve fault-tolerance. We will employ spare nodes

and add interconnection links via the unused extra link-ports. Since the required node-degree is $n + 2$, two extra link-ports will be used in each node for constructing the fault-tolerant hypercube. The hardware support for fault-tolerance is kept at minimum, since the extra link-ports are previously manufactured. If there is one faulty node, *no matter where it occurs*, the hypercube topology can be fully recovered. If two nodes become faulty, the second faulty node can be recovered with very high probability (around 0.92).

The rest of this paper is organized as follows. In Section II, we give the necessary background and define the terminology. In Section III, we first describe the proposed fault-tolerant hypercube, and then explicate the node replacement strategy and reconfiguration algorithm when faults occur. In Section III, we also compute the system's recoverability and do a brief cost analysis of our system, comparing its hardware requirement with other similarly conceived systems. In Section IV, we prove that the proposed system is $(n + 2)$-diagnosable. We give concluding remarks in Section V.

II. Preliminaries

The *n-hypercube* is one of the most popular interconnection models for multicomputer systems, where n is the *dimension* of the hypercube. Graphically speaking, an n-hypercube can be viewed as a graph $G = (V, E)$ such that V consists of 2^n nodes, numbered from $\underbrace{00\cdots0}_{n}$ to $\underbrace{11\cdots1}_{n}$. An edge (or link) $\{v_i, v_j\} \in E$, where $v_i, v_j \in V$, if and only if v_i and v_j have only one bit different. The nodes represent processors in the system, and the links represent communication channels among processors. Thus, every node has links with exactly n other nodes. There are altogether $n2^{n-1}$ links. Two nodes v_i, v_j of an n-hypercube that have d bits different are said to have *Hamming distance* d, denoted as $H(v_i, v_j) = d$. So in an n-hypercube, a link exists between v_i and v_j if and only if $H(v_i, v_j) = 1$. Notice that the nodes can be numbered differently as long as the above link regulation is obeyed. We will call n-hypercube simply n-cube for the sake of convenience. n-cube as a topology to interconnect processors has many attractive properties. Multicomputer systems built with hypercube structure have already been commercially available for a long time. Because of its importance for achieving high performance, the fault-tolerant computing for hypercube structures has been the interest of many researchers [3–7, 13, 15, 20].

An n-cube is composed of two $(n-1)$-cubes. We call the two $(n-1)$-cubes the *subcubes* of the n-cube. Recursively, an n-cube is composed of 2^{n-i} subcubes of dimension i. We represent a subcube of dimension i with notation $b_n \cdots b_{i+1} \underbrace{x\cdots x}_{i}$, which is composed of all nodes whose most significant $n - i$ bits are $b_n \cdots b_{i+1}$. An example showing an n-cube and some of its subcubes is given in Figure 1.

The fault-tolerant structure proposed in this paper employs extra nodes and links among nodes. We call the nodes (or links) of an original n-cube *primary nodes (or links)*, and call the extra nodes (or links) *spare nodes (or links)*. Sometimes we will just say primary or spare if the context makes it clear whether we mean nodes or

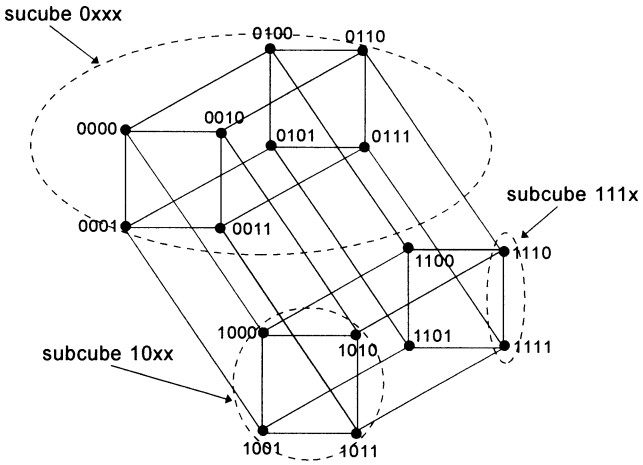

Figure 1. A 4-cube and its subcubes.

links. When a primary node becomes faulty, a spare will be chosen to replace it, and a reconfiguration, which may involve replacement of many spare nodes and links, will take place to restore the n-cube structure. Given the available spare nodes and links, if a faulty node can be replaced so that the original n-cube topology can be completely restored, then the faulty node is said to be *strongly* tolerated. In this paper we always seek strong tolerance of faulty nodes. Given the available spare nodes and links, if a faulty node can be tolerated no matter where it occurs, then the current system is said to be able to *fully* tolerate a fault. If a faulty node can be tolerated only when it occurs at some specific locations, then the current system is said to *partially* tolerate a fault. A node location is said to be a *recoverable location* if after the node's failure, the remaining nodes (with links among them) can still form a hypercube. Clearly, all spares are at recoverable locations because the original hypercube does not need any of them. We define the *recoverability* of a system, denoted α, by

$$\alpha = \frac{\text{number of nodes at recoverable locations}}{\text{total number of primaries and spares}}$$

For a fully tolerant system, $\alpha = 1$. When the system cannot tolerate any more fault, $\alpha = 0$. An ideal system should fully tolerate a large number of faulty nodes. But this cannot be done without very high hardware costs. The system proposed in this paper can fully tolerate the first faulty node, and tolerate the second faulty node with very high α (expectantly approaching 0.92 as n grows). It cannot tolerate the third faulty node.

It has been a long-standing approach for a multicomputer system to diagnose the faulty processors among themselves. The first paper to propose this approach by Preparata et al. dates back to 1967 [14]. In their model, the self-diagnosable system is represented by a directed graph $G = (V, A)$, or digraph for short, in which a node v_i can test all nodes v_j if arrow $(v_i \to v_j) \in A$. An undirected graph $G = (V, E)$ is a special case of a digraph $G = (V, A)$ in which $(v_i \to v_j) \in A \iff (v_j \to v_i) \in A$.

In our hypercube system, the two connected nodes are able to test each other directly, so the interconnection topology and testing assignment graph are the same. The test-result will be a conclusion that the tested node is "faulty" or "fault-free," denoted as label 1 or 0 on the corresponding arrow. A *syndrome* is defined as a function $s : A \to \{0, 1\}$. A subset $F \subseteq V$ is said to be *consistent* with a syndrome s if s can arise from the circumstance that all nodes in F are faulty and all nodes in $V - F$ are fault-free. For a given syndrome s, there maybe more than one subset of V that are consistent with s. If this happens, the system cannot diagnose for syndrome s, because the faulty-sets that can cause s are not unique. It is clear that for any system to perform self-diagnosis, there must be some (at least one) fault-free processors. The *diagnosability* is defined to be a positive integer t such that if $|F| > t$, the diagnosis cannot be carried out correctly. It is well-known that the diagnosability of an n-dimensional hypercube is n [1]. For the variants of hypercube, diagnosability has been a constant subject of research. The enhanced hypercube, for example, where there are 2^{n-1} more links than the plain hypercube, increases the diagnosability to $n + 1$ [18]. In this paper, we will show that our proposed fault-tolerant hypercube increases the system's diagnosability to $n + 2$.

III. The proposed fault-tolerant hypercube structure

A. The basic redundant cube

The basic "building block" of the proposed fault-tolerant hypercube is a redundant 3-cube. The 3-cube has 2 spare nodes, denoted as S_0 and S_1, respectively. Spare S_0 has 4 links to primary nodes 001, 010, 100 and 111. Spare S_1 is connected with the remaining nodes: 000, 011, 101 and 110. The two spares are connected by a link, too. Figure 2 shows the redundant 3-cube. It can be seen that the 4 primary nodes connected to the same spare all have Hamming distance of 2 between each other, and the structure is symmetrical. If any of the 8 primaries becomes faulty, it can be replaced by one of the spares. Without loss of generality we will assume 000 is the first node that goes bad. (Since the structure is symmetrical, if any other node goes bad first, the discussion is the same.) When node 000 goes faulty, spare

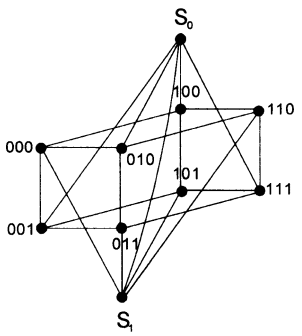

Figure 2. A basic redundant 3-cube.

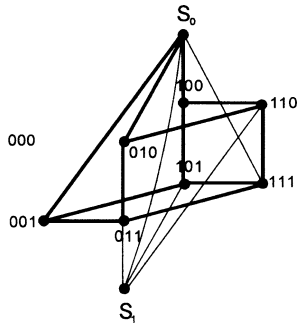

Figure 3. Reconfiguration after the first fault. The bold links represent the restored 3-cube after 000 becomes faulty.

node S_0 will take its place. The 3 (now failed) links from 000 to 001, 010 and 100 will be replaced by three corresponding links from S_0 and the original topology is completely restored. See Figure 3. For the remaining 7 primaries, not *every* node going faulty can be restored. However, 4 of them can be restored by the second spare S_1. Figure 4 shows the reconfigurations after 001 or 010 or 100 or 111 becomes second fault, respectively.

So for the redundant 3-cube, the first fault has the recoverability $\alpha_1 = 1$. For the second fault,

$$\alpha_2 = \frac{4 \text{ primaries } + 1 \text{ spare } (S_1)}{9} \approx 0.56$$

For $i \geq 3$, $\alpha_i = 0$, i.e., the third, fourth, ..., faulty nodes are unrecoverable.

B. The proposed fault-tolerant structure

Using this basic redundant 3-cube, we can build up a fault-tolerant hypercube of any size in the following way. The whole system consists of 2^{n-3} basic redundant 3-cubes, where $n \geq 3$. The basic cubes are addressed from $\underbrace{00\cdots0}_{n-3}xxx$ to $\underbrace{11\cdots1}_{n-3}xxx$. The two spares for $b_{n-3}b_{n-4}\cdots b_1xxx$ are denoted $b_{n-3}b_{n-4}\cdots b_1S_0$

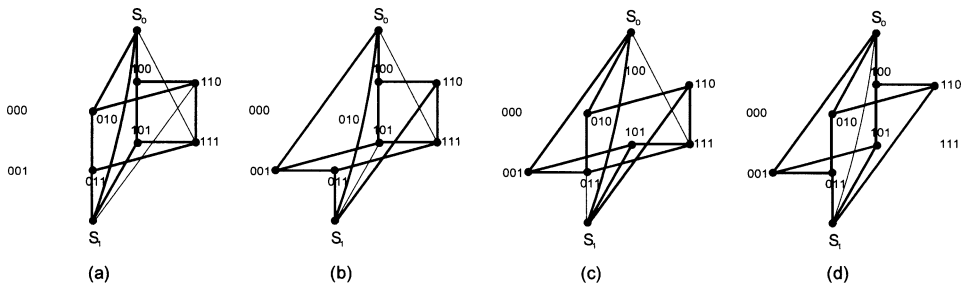

Figure 4. Reconfiguration after the second fault: (a), (b), (c) and (d) give the restoration if node 001 or 010 or 100 or 111 is the second faulty node, respectively.

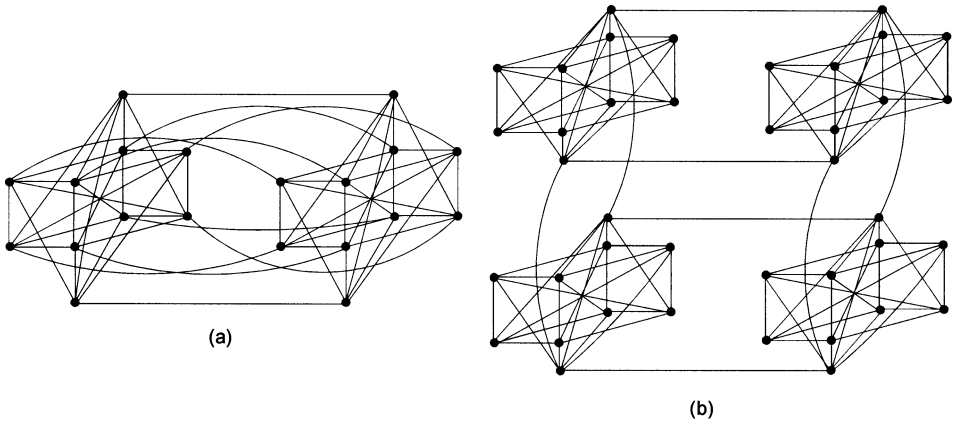

Figure 5. (a) A redundant 4-cube. (b) A redundant 5-cube in which the primary links between basic cubes are not shown.

and $b_{n-3}b_{n-4}\cdots b_1S_1$. The primary nodes between different basic cubes are connected in a regular n-cube manner. The spares are connected in a similar way: Two spares are linked if and only if their addresses have exactly one bit different. All spares themselves form a $(n-2)$-cube. For the purpose of further enhancing recoverability, we add 4 more extra links between a basic unit's primary nodes: Two nodes that have Hamming distance 3 are connected by an extra link. Figure 5 gives example redundant 4-cubes and 5-cubes.

Both a primary and a spare have the low node-degree of $n+2$, which is a desired property. A primary has the original n links, 1 link to a spare, and 1 extra link to a primary that is 3 Hamming distance away. A spare has 5 links within the basic cube it belongs to, and $n-3$ links to other cubes. All spares form an $(n-2)$-cube by themselves.

C. *The reconfiguration algorithm*

In the worst case, the proposed fault-tolerant hypercube can recover up to two faulty nodes. The pattern in which the two faults occur can be (1) both are primary nodes; (2) first primary, second spare; (3) first spare, second primary. We will separately discuss the reconfiguration strategy for each pattern. We do not discuss the trivial case, i.e., both faults are spare nodes, since no reconfiguration is needed if no primaries are failing.

C-1. Both faulty nodes are primary nodes. Without loss of generality we can assume that node $\underbrace{00\cdots0}_{n-3}000$ is the first to become faulty. Spare $\underbrace{00\cdots0}_{n-3}S_0$ will replace it with the corresponding 3 links. What is more, to easily and quickly facilitate a complete restore, all S_0 spares in other cubes replace the 000 nodes even though they are not faulty. The replaced good 000 nodes will now become spares for the second faulty node. We now analyze which nodes can be recovered as the second faulty, and how they are recovered.

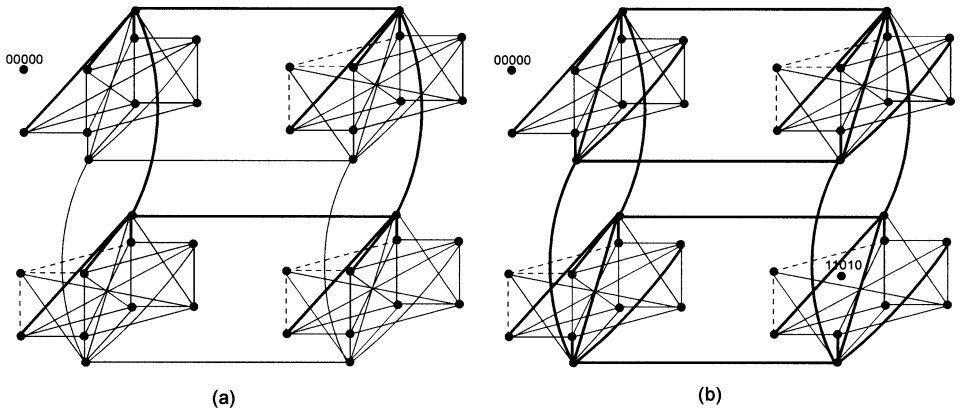

Figure 6. (a) Reconfiguration after 00000 becomes faulty first. (b) Reconfiguration after 11010 becomes faulty second. The bold links are activated spare links. The dashed links are good but inactivated.

First, any node whose least 3-bits are 001 or 010 or 100 or 111 can be recovered with spare S_1 in a way similar to what we showed for the basic cube. Just like in the case of first faulty, we replace the corresponding nodes in all cubes with their S_1's, so that the recovery process is easy and quick. Figure 6 shows an example reconfiguration after $00\cdots0000$ is the first faulty node and $11\cdots1010$ is the second. It can be seen that the original topology can be completely recovered, preserving all the computational power.

If the second faulty is a node whose least 3-bits are 011 or 101 or 110, and it does not fall in the same basic cube as the first faulty, then the hypercube topology still can be recovered, but in a weaker way, meaning that some direct links may have to be replaced by longer paths. Pick, say, $11\cdots1011$ as the second faulty. (For

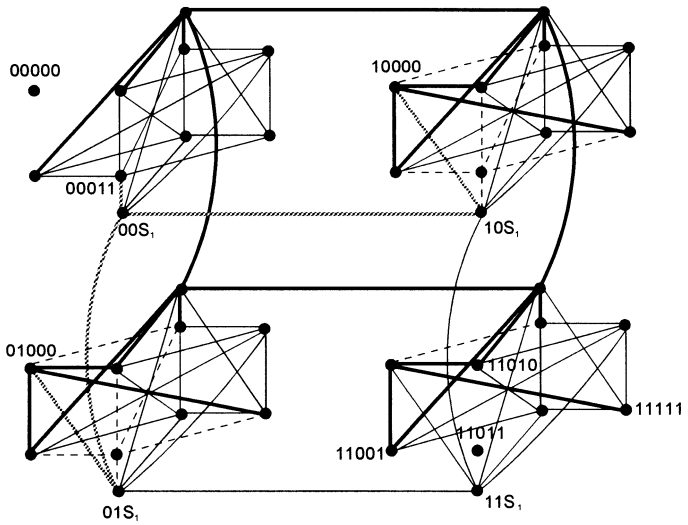

Figure 7. Reconfiguration 00000, 11011 become the first and second faulty node, respectively. The light bold links are paths replacing the originally direct links 01011–00011 and 10011–00011.

$11\cdots1101$ or $11\cdots1110$, the discussion is similar.) The good but replaced 000-node will be used as the spare in the following way. Refer to Figure 7. The 3 nodes that 011-node links to are 010-, 001- and 111-nodes. They are all linked to 000-node. Therefore 000-node can replace 011-node within the basic cube. For all the cubes except that of the first fault, the preceding replacement is carried out. In the cube of first fault, this replacement cannot be done because the 000-node in that cube is faulty. Thus, for inter-basic-cube connections, if the connection involves the cube of first fault, there should be a link between the 000-node in one cube (e.g., $01xxx$ in Figure 7) and the 011-node in the other ($00xxx$ in Figure 7). The length 3 path to replace such a link is as follows:

$$01000 \longrightarrow 01S_1 \longrightarrow 00S_1 \longrightarrow 00011.$$

Similarly, from 10000 to 00011 we have:

$$10000 \longrightarrow 10S_1 \longrightarrow 00S_1 \longrightarrow 00011.$$

It can be seen that there will be a communication delay due to the second fault occurring at 011- or 101- or 110-node.

If the second fault is a node whose least 3-bits are 000, then since its function already has been replaced by S_0 when recovering the first fault, nothing has to be done.

To summarize the preceding discussion: The first faulty node can be completely recovered no matter where it occurs, and without loss of generality it can be viewed as numbered $00\cdots0000$; if the second fault occurs at a node (in any basic cube) whose least 3-bits are 001 or 010 or 100 or 111, it can also be completely recovered; if the second fault occurs at a node whose least 3-bits are 011 or 101 or 110, and it does not fall in the same basic cube as the first fault, then it can still be recovered with some communication being delayed by a factor of 3; if the second fault is a node whose least 3-bits are 000, no recovery operation is needed. The nodes that cannot be recovered as the second fault include S_0 nodes in all basic cubes, and the three nodes in the basic cube of the first fault with least three bits 011 or 101 or 110.

The recoverability of the first fault $\alpha_1 = 1$. To calculate the recoverability for the second fault: After the first fault is out of consideration, the total number of nodes of the structure is $2^n + 2\cdot 2^{n-3} - 1$; the number of nodes that cannot be recovered is $3 + 2^{n-3}$. So, the recoverability of the second fault

$$\alpha_2 = \frac{(2^n + 2\cdot 2^{n-3} - 1) - (3 + 2^{n-3})}{2^n + 2\cdot 2^{n-3} - 1} = \frac{9\cdot 2^{n-3} - 4}{10\cdot 2^{n-3} - 1},$$

which approaches 0.9 even for a relatively small hypercube.

C-2. First fault primary, second fault spare. The operation to recover the first primary fault is the same as in C-1. We again assume that the first fault is $00\cdots000$. Then all S_0's have been used in recovering $00\cdots000$. So if the second (spare) fault is some S_0, it cannot be recovered. If instead, the second (spare) fault is some S_1,

then no recovery operation is needed, and the original hypercube topology is completely preserved.

The recoverability of the second fault α_2 in this case is also $(9 \cdot 2^{n-3} - 4)/(10 \cdot 2^{n-3} - 1)$.

C-3. First fault spare, second fault primary. Suppose the first fault is $\underbrace{00\cdots0}_{n-3}$ S_0. No reconfiguration is needed when this occurs. But we have lost the spare node for primary nodes $\underbrace{00\cdots0}_{n-3}$ 000, $\underbrace{00\cdots0}_{n-3}$ 011, $\underbrace{00\cdots0}_{n-3}$ 101 and $\underbrace{00\cdots0}_{n-3}$ 110. So when one of these four nodes becomes the second fault, it cannot be recovered.

For any 001-, 010-, 100-, or 111-node becoming the second fault, it can be recovered by replacing it with its S_1, and replacing the same node in all other blocks with S_1's. In any block except that of first fault, a 000-, or 011-, or 101-, or 110-node becoming the second fault can be recovered by replacing it with its S_0, and replacing the same node in all other blocks, except that of first block, with S_0's. Because $\underbrace{00\cdots0}_{n-3}$ S_0 has been faulty, the replacement cannot be effected in this block. This will result in link-delay in a handful of nodes. Suppose without loss of generality that a 000-node in some block other than that of first fault is the second fault. Then all 000-nodes are replaced with S_0's except $v_0 = \underbrace{00\cdots0}_{n-3}$ 000. So v_0 remains functioning. But its communication with $n-3$ neighboring S_0's will be delayed by a factor of 3. For instance, v_0's link to $\underbrace{00\cdots1}_{n-3}$ S_0 will now be via path

$$v_0 \longrightarrow \underbrace{00\cdots0}_{n-3} S_1 \longrightarrow \underbrace{00\cdots1}_{n-3} S_1 \longrightarrow \underbrace{00\cdots1}_{n-3} S_0.$$

The only nodes that cannot be recovered as the second fault are the 4 nodes in the block of first fault. Hence the second fault's recoverability

$$\alpha_2 = \frac{(2^n + 2 \cdot 2^{n-3} - 1) - 4}{2^n + 2 \cdot 2^{n-3} - 1} = \frac{10 \cdot 2^{n-3} - 5}{10 \cdot 2^{n-3} - 1},$$

which approaches 1 quickly as n grows.

We now calculate the expected recoverability $E(\alpha_2)$ for the second faulty node. Let $P\{A\}$ represent the probability for event A. Then

$$P_{pp} = P\{\text{first primary, second primary}\} = \frac{2^n}{2^n + 2^{n-2}} \cdot \frac{2^n - 1}{2^n + 2^{n-2} - 1},$$

$$P_{ps} = P\{\text{first primary, second spare}\} = \frac{2^n}{2^n + 2^{n-2}} \cdot \frac{2^{n-2}}{2^n + 2^{n-2} - 1},$$

$$P_{sp} = P\{\text{first spare, second primary}\} = \frac{2^{n-2}}{2^n + 2^{n-2}} \cdot \frac{2^n}{2^n + 2^{n-2} - 1},$$

$$P_{ss} = P\{\text{first spare, second spare}\} = \frac{2^{n-2}}{2^n + 2^{n-2}} \cdot \frac{2^{n-2} - 1}{2^n + 2^{n-2} - 1}.$$

We have

$$E(\alpha_2) = (P_{pp} + P_{ps}) \cdot \frac{9 \cdot 2^{n-3} - 4}{10 \cdot 2^{n-3} - 1} + (P_{sp} + P_{ss}) \cdot \frac{10 \cdot 2^{n-3} - 5}{10 \cdot 2^{n-3} - 1}$$
$$= 0.8 \cdot \frac{9 \cdot 2^{n-3} - 4}{10 \cdot 2^{n-3} - 1} + 0.2 \cdot \frac{10 \cdot 2^{n-3} - 5}{10 \cdot 2^{n-3} - 1}.$$

$E(\alpha_2)$ approaches 0.92 as n grows.

D. Cost

Besides the spare nodes, the only hardware needed to construct our proposed system are extra links since we are supposed to make use of the leftover link-ports previously fabricated. We will first calculate the total number of links needed for our spared n-cube and then compare with other fault-tolerant n-cubes. In our proposed system, all nodes are of degree $n + 2$, and there are $2^n + 2^{n-2}$ nodes in total. Thus $\sum_{v \in V} degree(v) = (n + 2)(2^n + 2^{n-2})$. By fundamental graph theory, for an undirected graph $G = (V, E)$, $\sum_{v \in V} degree(v) = 2|E|$. We get $|E| = (n + 2)(2^n + 2^{n-2})/2 = (5n + 10)2^{n-3}$. Among them, the original n-cube has $n2^{n-1}$ links; every basic 3-cube has another four "crossing" links, totaling $4 \cdot 2^{n-3}$; the 2^{n-2} spares form an $(n - 2)$-cube themselves, thus having $(n - 2)2^{n-3}$ links; finally, each pair of spares in a basic 3-cube has 8 links to primaries, totaling $8 \cdot 2^{n-3}$.

To see that our hardware requirement is a very modest one, let's see what other proposed systems need. First of all, our system needs no specially fabricated reconfiguration switches as were used by the systems proposed in [20] and [8]. In a comparative study in [20], it was shown that the system of [20] requires the least number of links among similar systems. A system proposed in [20] requires $[2n(2^m + p) + m2^m]2^{n-m}$ links, where n is the dimension of the hypercube, m is the dimension of FTM, and p is the number of spare nodes in each FTM. Thus, a system of [20] with $m = 3$ and $p = 2$ has exactly the same number of nodes as ours. The total number of links in such a system is $[2n(2^3 + 2) + 3 \cdot 2^3]2^{n-3} = (20n + 24)2^{n-3}$. The links our system takes are only about one fourth of that number.

IV. The diagnosability of the proposed structure

In this section we will show that, in addition to its fault-tolerating capability, the proposed structure also achieves better diagnosability than a plain hypercube. More specifically, we will prove that the whole system's diagnosability is increased to $n + 2$ for a fault-tolerant n-cube, whereas the diagnosability of an ordinary n-cube is given in the following lemma.

Lemma 1 [1, 12]. *A system of n-cube structure is n-diagnosable if* $n \geq 3$.

There are several different ways to characterize a t-diagnosable system. In our proof, we will use the characterization by Allan et al. [2] and Sullivan [16].

Definition 1 Let $G = (V, E)$. For a subset $V' \subset V$, the tester-set of V', denoted by $\mu^{-1}V'$, is defined as

$$\mu^{-1}V' = \{v | v \in V \text{ and } \{v, v'\} \in E \text{ for some } v' \in V'\} - V'.$$

Lemma 2 [16]. *A system $G = (V, A)$ is t-diagnosable if and only if*

$$\forall V' \subseteq V[V' \neq \phi \Rightarrow \frac{|V'|}{2} + |\mu^{-1}V'| > t].$$

Theorem 1 *The proposed fault-tolerant n-cube is $(n+2)$-diagnosable.*

Proof: We prove the theorem by showing that for any non-empty subset V' of V, $\frac{|V'|}{2} + |\mu^{-1}V'| > n + 2$ will be satisfied. Then by Lemma 2, the system is $(n+2)$-diagnosable.

Case 1. V' contains only primaries.

Without considering the V''s additional testers due to the spare nodes and extra links, we have $\frac{|V'|}{2} + |\mu^{-1}V'| > n$ by Lemmas 1 and 2. Without loss of generality we can always assume $v_0 = 0 \cdots 0000 \in V'$.

Case 1.1. $v_1 = 0 \cdots 0111 \in V'$. Then $\mu^{-1}V'$ will have 2 more nodes, $0 \cdots 0S_0$ (testing v_0) and $0 \cdots 0S_1$ (testing v_1). Therefore we have $\frac{|V'|}{2} + |\mu^{-1}V'| > n + 2$.

Case 1.2. $v_1 = 0 \cdots 0111 \notin V'$. Then $\mu^{-1}V'$ will also have 2 more nodes, $0 \cdots 0S_0$ and v_1 (both testing v_0). Thus we have $\frac{|V'|}{2} + |\mu^{-1}V'| > n + 2$.

Case 2. V' contains only spares.

All spare nodes form an $(n-2)$-cube by themselves. So by Lemmas 1 and 2, $\frac{|V'|}{2} + |\mu^{-1}V'| > n - 2$, where $\mu^{-1}V'$ only takes spares into account. Now pick any spare $v_0 \in V'$. v_0 will have four more primaries testing it. Therefore, we have $\frac{|V'|}{2} + |\mu^{-1}V'| > n - 2 + 4 = n + 2$.

Case 3. V' contains both primaries and spares.

Let $V' = V'_p \cup V'_s$, where V'_p, V'_s is a subset of primaries and spares, respectively. By Lemmas 1 and 2, we have $\frac{|V'_p|}{2} + |\mu^{-1}V'_p| > n$ and $\frac{|V'_s|}{2} + |\mu^{-1}V'_s| > n - 2$, where $\mu^{-1}V'_p$ only contains primaries and $\mu^{-1}V'_s$ only contains spares. Since $\mu^{-1}V'_p \cap \mu^{-1}V'_s = \phi$, it must follow that $\frac{|V'|}{2} + |\mu^{-1}V'| > 2n - 2 \geq n + 2$ for $n \geq 4$, which is a satisfied condition. □

V. Conclusion

We have proposed a strong fault-tolerant hypercube structure that uses spare nodes and links. It needs relatively very little hardware support and yet has been shown to achieve satisfactory results: The first faulty node can be fully recovered no matter where it occurs, the second faulty node can be recovered at over ninety percent

of locations. The advantage of our proposed structure is that it makes use of the unused extra link-ports in nodes of the hypercube to obtain the wanted topology. Since the extra link-ports are previously manufactured, the hardware support for obtaining fault-tolerance is minimal and the constructing process very easy. Besides the spare nodes, the only hardware needed for constructing the proposed system are extra links. A cost analysis shows that the total number of links of the system is $(5n + 10)2^{n-3}$, including both primary and spare links. This is about one fourth of the links needed by an equal sized fault-tolerant hypercube structure proposed earlier in [20].

We have also shown that the proposed structure enhances the diagnosability of the hypercube system. It has been shown that the diagnosability of the structure is increased to $n + 2$, while an ordinary n-dimensional hypercube has diagnosability n.

References

1. J. R. Armstrong and F. G. Gray. Fault diagnosis in a Boolean *n* cube array of microprocessors. *IEEE Transactions on Computing*, C-30(8):587–590, 1981.
2. F. J. Allan, T. Kameda, and S. Toida. An approach to the diagnosability analysis of a system. *IEEE Transactions on Computing*, 24(10):1040–1042, 1975.
3. M. S. Alam and R. G. Melhem. An efficient modular spare allocation scheme and its application to fault-tolerant binary hypercube. *IEEE Transactions on Parallel Distributed Systems*, 2:117–126, 1991.
4. P. Banerjee. Strategies for reconfiguring hypercube under faults. In *Proceedings of the 20th International Symposium on Fault-Tolerant Computing*, 1990.
5. J. Bruck, R. Cypher, and C.-T. Ho. Efficient fault-tolerant mesh and hypercube architectures. In *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*, July 1992, pp. 162–169.
6. J. Bruck, R. Cypher, and D. Soroker. Running algorithms efficiently on faulty hypercubes. *Computer Architecture News*, 19(1):89–96, 1991.
7. J. Bruck, R. Cypher, and D. Soroker. Embedding cube-connected cycles graphs into faulty hypercubes. *IEEE Transactions on Computing*, 43(10):1210–1220, 1994.
8. S. L. Chau and A. L. Liestman. A proposal for a fault-tolerant binary hypercube architecture. In *Proceedings of IEEE Fault Tolerant Computing*, 1989, pp. 323–330.
9. G.-M. Chiu and K.-S. Chen. Use of routing capability for fault-tolerant routing in hypercube multicomputers. *IEEE Transactions on Computing*, 46(8):953–958, 1997.
10. G.-M. Chiu and S.-P. Wu. A fault-tolerant routing strategy in hypercube multicomputers. *IEEE Transactions on Computing*, 45(2):143–155, 1996.
11. K. Kaneko and H. Ito. Fault-tolerant routing algorithms for hypercube networks. In *Proceedings of the 13th International Parallel Processing Symposium (IPPS) and 10th Symposium on Parallel and Distributed Processing (SPDP)*, April 1999, pp. 218–224.
12. J. Kuhl and S. Reddy. Distributed fault-tolerance for large multiprocessor systems. In *Proceedings of the 7th International Symposium Computing Architecture*, 1980, pp. 23–30.
13. T. C. Lee. Quick recovery of embedded structures in hypercube computers. In *Proceedings of the 5th Distributed Memory Computing Conference*, April 1990, pp. 1426–1435.
14. F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computing*, EC-16(12):848–854, 1967.
15. C. S. Raghavendra, P.-J. Yang, and S.-B. Tien. Free dimensions—an efficient approach to achieving fault tolerance in hypercubes. In *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*, July 1992, pp. 170–177.
16. G. F. Sullivan. A polynomial time algorithm for fault diagnosability. In *Proceedings of the 25th Annual Symposium on the Foundations of Computing Science*, pp. 148–156. IEEE Computer Society, 1984.

17. N.-F. Tzeng and S. Wei. Enhanced hypercubes. *IEEE Transactions on Computing*, C-40(3):284–294, 1991.
18. D. Wang. Diagnosability of enhanced hypercubes. *IEEE Transactions on Computing*, 43(9):1054–1061, 1994.
19. J. Wu. Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. *IEEE Transactions on Parallel and Distributed Systems*, 9(4):321–334, 1998.
20. C. S. Yang, L. P. Zu, and Y. N. Wu. A reconfigurable modular fault-tolerant hypercube architecture. *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1018–1032, 1994.