Journal of Parallel and Distributed Computing **61**, 545–564 (2001)
doi:10.1006/jpdc.2000.1681, available online at http://www.idealibrary.com on IDEAL®

Embedding Hamiltonian Cycles into Folded Hypercubes with Faulty Links

Dajin Wang

Department of Computer Science, Montclair State University, Upper Montclair, New Jersey 07043
E-mail: wang@pegasus.montclair.edu

Received June 3, 1999; revised May 16, 2000; accepted August 23, 2000

It has been known that an n-dimensional hypercube (n-cube for short) can always embed a Hamiltonian cycle when the n-cube has no more than $n-2$ faulty links. In this paper, we study the link-fault tolerant embedding of a Hamiltonian cycle into the *folded hypercube*, which is a variant of the hypercube, obtained by adding a link to every pair of nodes with complementary addresses. We will show that a folded n-cube can tolerate up to $n-1$ faulty links when embedding a Hamiltonian cycle. We present an algorithm, *FT_HAMIL*, that finds a Hamiltonian cycle while avoiding any set of faulty links F provided that $|F| \leqslant n-1$. An operation, called *bit-flip*, on links of hypercube is introduced. Simple yet elegant, bit-flip will be employed by *FT_HAMIL* as a basic operation to generate a new Hamiltonian cycle from an old one (that contains faulty links). It is worth pointing out that the algorithm is optimal in the sense that for a folded n-cube, $n-1$ is the maximum number for $|F|$ that can be tolerated, F being an arbitrary set of faulty links.

Key Words: embedding; fault tolerance; folded hypercube; Hamiltonian cycle; link fault.

1. INTRODUCTION

The *hypercube* structure is a well-known interconnection model. As a topology for the interconnection of a multiprocessor system, it has been proven to possess many attractive properties, and multiprocessor computers built with hypercube structure have been in existence for a long time [3–5, 8]. Because of its importance, fault-tolerant computing for hypercube structures has been the focus of extensive research. A very important property of a hypercube is its ability to embed, or emulate, many commonly used structures such as rings, arrays, and trees. It has been known that a regular n-dimensional hypercube can embed a Hamiltonian cycle in the presence of up to $n-2$ faulty links [6]. That is, for an n-dimensional hypercube (which has 2^n nodes), we can always find a cycle traversing each and every node exactly once, using only nonfaulty links no matter how the faulty links are distributed, provided that there are no more than $n-2$ faulty links.

0743-7315/01 $35.00

Since its introduction, many variants of the hypercube have been proposed [1, 2, 7]. One variant that has aroused the interest of many researchers is the *folded hypercube*, which is an extension of the hypercube, constructed by adding a link to every pair of nodes that are the farthest apart, i.e., two nodes with complementary addresses. The folded hypercube has been proven to be able to improve the system's performance over a regular hypercube in many measurements [1]. Later, a more generalized kind of folded hypercube, *enhanced hypercube*, was proposed [9]. It has been shown that these extra links noticeably increase the hypercube's performance in many aspects [9, 10].

This paper studies the link-fault tolerance of the folded hypercube with respect to Hamiltonian cycle embedding. Being able to embed a Hamiltonian cycle in an interconnection network is very important. Many communication tasks (such as multicasting and broadcasting) rely on the establishment of a path that traverses every node. Since a folded hypercube shows performance improvement over a regular hypercube in many aspects, we are interested in knowing whether it can tolerate more faulty links when embedding a Hamiltonian cycle. As will be shown in the paper, a folded n-cube can tolerate up to $n-1$ faulty links when it embeds a Hamiltonian cycle. A succinct algorithm will be presented that finds a Hamiltonian cycle while avoiding any set F of faulty links provided that $|F| \leqslant n-1$. The algorithm is optimal in the sense that for a folded n-cube, $n-1$ is the maximum number for $|F|$ that can be tolerated—since every node has $n+1$ links connected to it, if n faulty links were allowed and all of them emanated from one node, there would be no Hamiltonian cycles.

It is worth pointing out that the added links in the folded hypercube were not introduced only for the purpose of increasing fault tolerability. Rather, many improvements are observed due to these added links [1, 9]. The result of this paper just reveals another good feature (and the algorithm to get it) of the folded hypercube, in addition to all the known ones. The rest of this paper is organized as follows. In Section 2, we give formal descriptions of regular and folded hypercubes. Section 3 has five subsections. Section 3.1 presents algorithm *HAMIL* that generates an initial Hamiltonian cycle. In Section 3.2, we introduce an operation called *bit-flip* that generates a new Hamiltonian cycle from an old one (that contains faulty links). Bit-flip will be employed by *FT_HAMIL* as a basic operation. Sections 3.3 and 3.4 describe algorithm *FT_HAMIL* that produces a Hamiltonian cycle in a folded n-cube, tolerating up to $n-1$ faulty links. Section 3.5 is a brief summary of 3.3 and 3.4. Section 4 gives some concluding remarks. Finally, a correctness proof for algorithm *FT_HAMIL* is provided in the Appendix.

2. PRELIMINARIES

An n-dimensional hypercube, n-cube for short, can be represented as an undirected graph $G(V, E)$ such that V consists of 2^n nodes, addressed (numbered) from $\underbrace{00\cdots0}_{n}$ to $\underbrace{11\cdots1}_{n}$, and a link (or edge) $e=\{v_i, v_j\} \in E$ iff v_i and v_j have exactly one bit different. Thus, each node has immediate links with exactly n other nodes. It can easily be shown that $|E| = n2^{n-1}$.

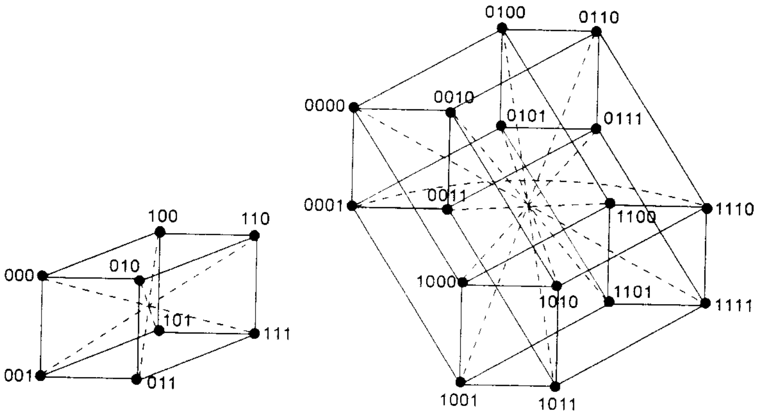

FIG. 1. A folded 3-cube and a folded 4-cube, in which links in skip set S are represented with dashed lines.

To uniquely represent a link e, note that the two nodes linked by e have exactly one bit different. Therefore e can be denoted using the two nodes it links: If $v_i = b_n..b_k..b_1$, $v_j = b_n..\bar{b}_k..b_1$ (where $b_l \in \{0, 1\}$, $l = 1, ..., n$), and $e = \{v_i, v_j\}$, then we denote e as $b_n..b_{k+1}xb_{k-1}..b_1$. We call $b_n..b_{k+1}xb_{k-1}..b_1$ a link of dimension-k, or dim-k for short. There are 2^{n-1} links in each dimension.

A folded hypercube is a regular hypercube augmented by adding more links among its nodes. More specifically, a folded n-cube is obtained by adding a link between two nodes whose addresses are complementary to each other; i.e., for a node whose address is $b_1 b_2 \cdots b_n$, it now has one more link to node $\bar{b}_1 \bar{b}_2 \cdots \bar{b}_n$, in addition to its original n links. So a folded n-cube has 2^{n-1} more links than a regular n-cube. We call these augmented links *skips*, to distinguish them from regular links, and use S to denote the set of skips. So the complete link set of a folded hypercube can be expressed as $E \cup S$. Figure 1 shows a 3-dimensional and a 4-dimensional folded hypercube.

In this paper, we study the problem of link-fault tolerant embedding of Hamiltonian cycles into folded hypercubes. We will show that in terms of Hamiltonian cycle embedding, the folded hypercube also outperforms its regular counterpart, in addition to many known improvements. It will be established that a folded n-cube can tolerate up to $n-1$ faulty links when it embeds a Hamiltonian cycle. We will propose an elegant algorithm that finds a Hamiltonian cycle while avoiding any set F of faulty links provided that $|F| \leqslant n-1$. The algorithm is also optimal because for a folded n-cube, $n-1$ is the maximum number for $|F|$ so that F can be tolerated.

3. FINDING A HAMILTONIAN CYCLE IN FOLDED n-CUBES WITH FAULTY LINKS

3.1. *Generating Initial Hamiltonian Cycle*

For the purpose of our algorithm, we list the $n2^{n-1}$ links of an n-cube in a specific pattern; i.e., the $n2^{n-1}$ are listed in n columns, with column i containing all links of dim-i. The first column lists the 2^{n-1} links in the following "increasing" order:

$$\underbrace{\begin{array}{c} 000\cdots 00x \\ 000\cdots 01x \\ \cdots\cdots\cdots \\ 111\cdots 11x \end{array}}_{n}$$

The ith column, $2 \leqslant i \leqslant n$, is obtained by left-rotating one bit, including the x-bit, for all links in the $(i-1)$th column. For example, the listing of all links of a 5-cube is

dim-5	dim-4	dim-3	dim-2	dim-1
$x0000$	$0x000$	$00x00$	$000x0$	$0000x$
$x0001$	$1x000$	$01x00$	$001x0$	$0001x$
$x0010$	$0x001$	$10x00$	$010x0$	$0010x$
$x0011$	$1x001$	$11x00$	$011x0$	$0011x$
$x0100$	$0x010$	$00x01$	$100x0$	$0100x$
$x0101$	$1x010$	$01x01$	$101x0$	$0101x$
$x0110$	$0x011$	$10x01$	$110x0$	$0110x$
$x0111$	$1x011$	$11x01$	$111x0$	$0111x$
$x1000$	$0x100$	$00x10$	$000x1$	$1000x$
$x1001$	$1x100$	$01x10$	$001x1$	$1001x$
$x1010$	$0x101$	$10x10$	$010x1$	$1010x$
$x1011$	$1x101$	$11x10$	$011x1$	$1011x$
$x1100$	$0x110$	$00x11$	$100x1$	$1100x$
$x1101$	$1x110$	$01x11$	$101x1$	$1101x$
$x1110$	$0x111$	$10x11$	$110x1$	$1110x$
$x1111$	$1x111$	$11x11$	$111x1$	$1111x$

To construct a Hamiltonian cycle in a fault-free n-cube, we make use of an algorithm called *REMOVAL*(k), initially introduced in [11]. *REMOVAL* selectively deletes $k2^{n-1}$ $(k \leqslant n-2)$ links from an n-cube, so that the remaining $(n-k)\,2^{n-1}$ links induce a subgraph of the n-cube that is symmetrically structured and $(n-k)$-connected. It turns out that when $k=n-2$, the remaining $2 \cdot 2^{n-1} = 2^n$ links construct a Hamiltonian cycle. Since we use *REMOVAL* exclusively with $k=n-2$ to induce a Hamiltonian cycle, we rename it as *HAMIL* in this paper. *HAMIL* takes as input the complete link set, listed in n columns as shown above. The 2^{n-1} links in a column are referred as the first link, second link, etc., in top-down order.

ALGORITHM *HAMIL*

```
{Purpose: Remove (n − 2) 2^(n−1) links from an n-cube
  to induce a Hamiltonian cycle}
Input: The complete link set of n-cube listed in n columns
Output: The remaining 2^n links forming a Hamiltonian cycle

for (j = 1 to n − 2)
{
        at column j,
                for i = 1 to 2^(n−j−1)
                { remove the i th link }

        at column j + 1,
                for i = 2^(n−j−2) + 1 to 2^(n−2)
                { remove the i th link }

        at column j + 1,
                for i = 2^(n−2) + 1 + 2^(n−j−2) to 2^(n−1)
                { remove the i th link }
}                                                /* end_for */
```

Applying *HAMIL* to a 4-cube and a 5-cube, respectively, the Hamiltonian link sets generated by *HAMIL* are shown below. Figure 2 shows the Hamiltonian cycle *HAMIL* generated for the 4-cube. A proof that this algorithm works correctly for an n-cube of any n can be found in [11].

4-cube

dim-4	dim-3	dim-2	dim-1
$x000$	$0x00$		
$x001$			
$x010$			
$x011$			
$x100$	$0x10$	$00x1$	$100x$
$x101$		$01x1$	$101x$
$x110$			$110x$
$x111$			$111x$

5-cube

dim-5	dim-4	dim-3	dim-2	dim-1
$x0000$	$0x000$			
$x0001$				
$x0010$				
$x0011$				
$x0100$				
$x0101$				
$x0110$				
$x0111$				
$x1000$	$0x100$	$00x10$	$000x1$	$1000x$
$x1001$		$01x10$	$001x1$	$1001x$
$x1010$			$010x1$	$1010x$
$x1011$			$011x1$	$1011x$
$x1100$				$1100x$
$x1101$				$1101x$
$x1110$				$1110x$
$x1111$				$1111x$

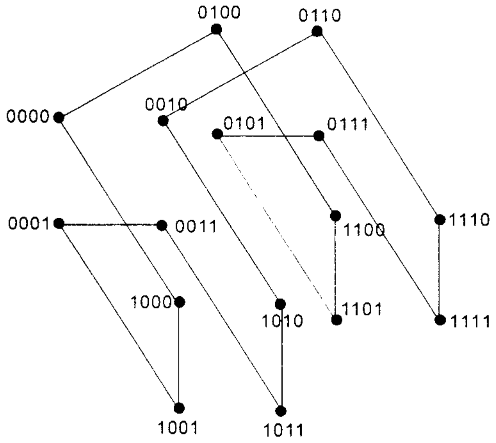

FIG. 2. Hamiltonian cycle induced from a 4-cube.

Algorithm *HAMIL* gives one specific Hamiltonian cycle embedding for a regular n-cube. It is the starting point of our procedure to find a Hamiltonian cycle in a folded n-cube, avoiding any set F of faulty links provided that $|F| \leqslant n-1$. Let this "starting" link set forming a Hamiltonian cycle be denoted H_0. If $H_0 \cap F = \emptyset$, then our job is done.

However, if $H_0 \cap F \neq \emptyset$, we need to find another Hamiltonian link set that avoids all links in F. The rest of this section describes an algorithm, *FT_HAMIL*, that finds a Hamiltonian link set using only good regular links if F contains at most $n-2$ regular links, or using good regular and skip links if F contains $n-1$ regular links.

3.2. Obtaining New Hamiltonian Cycle via Bit-flip

We show in this subsection that by a simple operation, which we call *bit-flip*, on all the links, we can readily get a new Hamiltonian link set from an old one. Bit-flip will serve as the basic operation in our algorithm.

If we take the complement of a specific bit over all *node addresses*, we get a new addressing for an n-cube. (Exchanging any two bits over all nodes also gives a new addressing. By a simple induction on n we can show that there are altogether $n!\,2^n$ different ways to address an n-cube.) We call this operation bit-flip.

A similar operation can be defined on the *links* of an n-cube, in which we take the complement at a specific bit-position i for links of all dimensions except dim-i (links of dim-i have x at bit-i). In this paper, bit-flip is always performed on links unless pointed out otherwise. We use $BF(i)$ to denote a bit-flip at bit-i.

It turns out that bit-flip can effectively specify a new Hamiltonian link set from an old one. To illustrate the idea, observe the result of bit-flipping on H_0. In Fig. 3, 3a shows H_0, 3b is the result of $BF(1)$ on H_0, and 3c is the result of $BF(4)$ on 3b.

It can be seen from Fig. 3b that the effect of $BF(1)$ is to move up the two dim-2 links, and move down the two dim-3 links of H_0 ("switch" the links along dim-1), so that a new Hamiltonian cycle is produced. If H_0 is known to contain faulty links in dim-2 or dim-3 (i.e., links $00x1, 01x1, 0x00$, and $0x10$), the new Hamiltonian

H_0

dim-4	*dim*-3	*dim*-2	*dim*-1
$x000$	$0x00$		
$x001$			
$x010$			
$x011$			
$x100$	$0x10$	$00x1$	$100x$
$x101$		$01x1$	$101x$
$x110$			$110x$
$x111$			$111x$

(a)

After $BF(1)$ on (a)

dim-4	*dim*-3	*dim*-2	*dim*-1
$x000$		$00x0$	
$x001$		$01x0$	
$x010$	$0x01$		
$x011$			
$x100$			$100x$
$x101$			$101x$
$x110$	$0x11$		$110x$
$x111$			$111x$

(b)

After $BF(4)$ on (b)

dim-4	*dim*-3	*dim*-2	*dim*-1
$x000$			$000x$
$x001$			$001x$
$x010$		$10x0$	$010x$
$x011$	$1x01$	$11x0$	$011x$
$x100$			
$x101$			
$x110$			
$x111$	$1x11$		

(c)

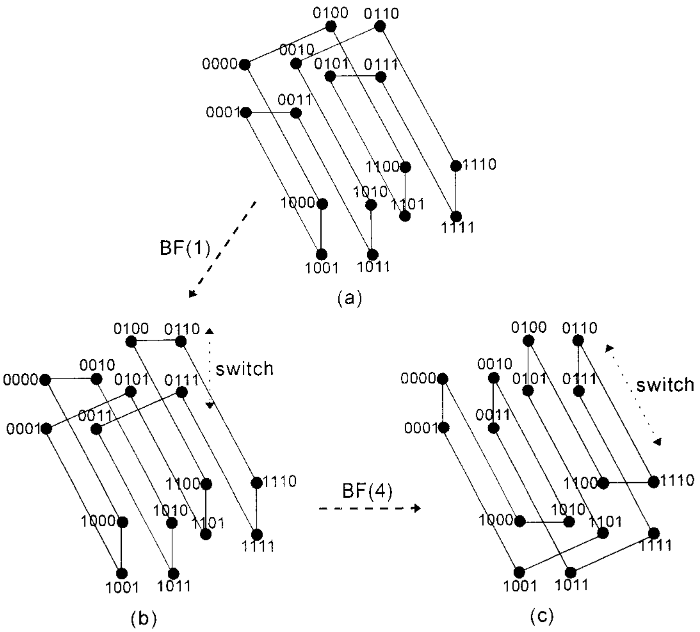

FIG. 3. (a) H_0 from a 4-cube. (b) The effect of *BF*(1) on (a). (c) The effect of *BF*(4) on (b).

cycle apparently avoids them. Likewise, in Fig. 3c, the effect of $BF(4)$ is as if to switch the links along dim-4, so that a new Hamiltonian cycle is generated.

Bit-flip will be used as the basic operation in our algorithm to find a Hamiltonian cycle while avoiding faulty link set F in a folded n-cube, provided $|F| \leqslant n-1$. As mentioned earlier, embedding a Hamiltonian cycle into a link-faulty regular n-cube was considered before. In [6], an algorithm using the Gray Code concept was proposed to find a Hamiltonian cycle avoiding up to $n-2$ faulty links. However, the algorithm we will present in the following subsections, using operation bit-flip, is much simpler. Moreover, with only minor modification, it can be adapted to find a Hamiltonian cycle avoiding up to $n-1$ faulty links in a folded n-cube.

We classify F into two cases and treat them in separate subsections. In Section 3.3, we consider the case of $|F \cap E| \leqslant n-2$; i.e., the regular link set contains at most $n-2$ faulty links, and if $|F| > n-2$, the remaining faulty links must belong to the skip set S. In this case, we can always find a fault-avoiding Hamiltonian cycle so that the cycle uses only regular links. In Section 3.4, we consider the case of $|F \cap E| = n-1$; i.e., all faulty links fall into the regular link set. In this case, we can find a fault-avoiding Hamiltonian cycle that uses both regular links and skips.

3.3. $|F \cap E| \leqslant n-2$

We can always assume that $|F \cap E| = n-2$, and the algorithm we present in this subsection will find a Hamiltonian cycle avoiding all links in F. Furthermore, the fault-avoiding Hamiltonian cycle uses only regular links. For an F such that $|F \cap E| < n-2$, this algorithm obviously also works.

Let $F_i = \{\text{faulty links in dim-}i\}$, $1 \leqslant i \leqslant n$, and let $f_i = |F_i|$. Then clearly

$$F_i \cap F_j = \varnothing \quad \text{if } i \neq j \qquad \text{and} \qquad \sum_{i=1}^{n} f_i = n-2.$$

It is known that there are $n!\,2^n$ different ways to address an n-cube. Figure 4 shows the eight different addressings of 2-cube. It can be seen that the dimension assignment of links varies as the node addressing varies.

Before generating Hamiltonian cycle H_0 using *HAMIL*, we first reassign dimensions to all links. Since there are $n-2$ faulty (regular) links, there will be at least

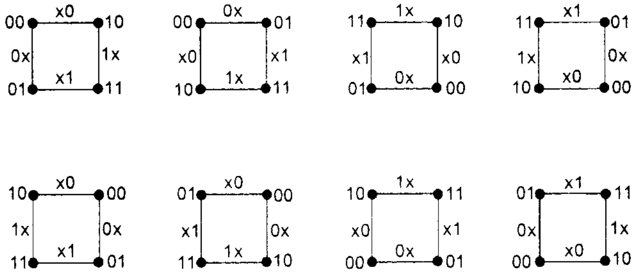

FIG. 4. All eight addressings of 2-cube.

two dimensions that do not contain any faulty links. So we can always address the folded n-cube, i.e., assign dimensions to links, in such a way that

$$f_1 = f_n = 0; \tag{3.3-1}$$

$$0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant n-2. \tag{3.3-2}$$

What (3.3-1)–(3.3-2) mean is that we want dimensions 1 and n to contain no faulty links, dimension $(n-1)$ to contain the largest number of faulty links, and dimension 2 the smallest number of faulty links. The purpose for this dimension reassignment will be explained in the proof of the algorithm.

After the preceding dimension reassignment, we apply algorithm *HAMIL* to generate Hamiltonian cycle H_0. If H_0 is already free of faulty links, then we are done. However, if H_0 contains some (up to $n-2$) faulty links, we start the procedure described below to find a fault-free Hamiltonian cycle.

We name the algorithm *FT_HAMIL*, standing for *FaultTolerant_HAMILtonian.* The procedure repeatedly generates a new Hamiltonian cycle H_{i+1} from an old one H_i, so that some (at least one) faulty links present in H_i are avoided in H_{i+1}. The procedure terminates the first time a fault-free Hamiltonian cycle is attained. Termination of the procedure is guaranteed because the number of repetitions is limited by the number of faults.

Algorithm *FT_HAMIL*

{Purpose: Find a Hamiltonian cycle in a folded n-cube avoiding all (up to $n-2$) regular faulty links, the resulting Hamiltonian cycle consists of only regular good links}

Input: Hamiltonian cycle H_0 with faulty links present
Output: A fault-free Hamiltonian cycle consisting of only regular links

bf[1..*n*]: An array of flags. A bit-flip can be performed at bit-*i* only if *bf*[*i*] = "*unflipped*"

```
begin:  for (h = n, n-3, ..., 2, 1)              /* Initialization of array bf */        (1)
            bf[h] ← "unflipped"
        for (h = n-1, n-2)                                                              (2)
            bf[h] ← "flipped"
        i ← 0                                                 /* Starts with H0 */       (3)

while:  while (Hi contains faulty links)                                                 (4)
        {
            for (j = n-1, n-2, ..., 3, 2)                                               (5)
            {
                if (Hi contains faulty links in dim-j)                                  (6)
                {
                    k ← j-1                                                             (7)
                    while (bf[k] ≠ "unflipped")       /* Find an "unflipped" bit k */   (8)
                        if (k-1) > 0){k ← k-1}
                        else {k ← n}
                    Hi+1 ← BF(k) on Hi          /* New Hamiltonian cycle generated */   (9)
                    bf[k] ← "flipped"      /* BF will not be performed at this bit again */ (10)
                    i ← i+1                                                             (11)
                    goto while                                                          (12)
                }
            }                                                         /* end_for */
        }                                                           /* end_while */
        return Hi     /* Hi a fault-free Hamiltonian cycle consisting of only regular links */ (13)
```

We will go over the algorithm for an example of a folded 6-cube with five faults, where four of those faults are on regular links. A correctness proof for *FT_HAMIL* is provided in the Appendix.

EXAMPLE. Suppose the four faulty links are 010x10, 00x010, 0x1000, and 0x1111. So $f_2 = 0$, $f_3 = 1$, $f_4 = 1$, and $f_5 = 2$. A faulty link will be represented by placing a box around it. Initially,

$bf =$

6	5	4	3	2	1
unflipped	*flipped*	*flipped*	*unflipped*	*unflipped*	*unflipped*

and H_0 contains faulty links 0x1000 and 010x10:

$H_0 =$

dim-6	*dim*-5	*dim*-4	*dim*-3	*dim*-2	*dim*-1
x00000	0x0000				
x00001					
x00010					
x00011					
x00100					
x00101					
x00110					
x00111					
x01000					
x01001					
x01010					
x01011					
x01100					
x01101					
x01110					
x01111					
x10000	0x1000	00x100	000x10	0000x1	10000x
x10001		01x100	001x10	0001x1	10001x
x10010			010x10	0010x1	10010x
x10011			011x10	0011x1	10011x
x10100				0100x1	10100x
x10101				0101x1	10101x
x10110				0110x1	10110x
x10111				0111x1	10111x
x11000					11000x
x11001					11001x
x11010					11010x
x11011					11011x
x11100					11100x
x11101					11101x
x11110					11110x
x11111					11111x

.

According to the algorithm, the first fault encountered is in dim-5 ($0x1000$), and the first "unflipped" $k \leqslant 4$ is 3, so a $BF(3)$ is performed on H_0, and the result sent to H_1. The updated *bf* and H_1 are shown below:

bf =

6	5	4	3	2	1
unflipped	*flipped*	*flipped*	*flipped*	*unflipped*	*unflipped*

H_1 =

dim-6	*dim*-5	*dim*-4	*dim*-3	*dim*-2	*dim*-1
*x*00000		00*x*000			
*x*00001		01*x*000			
*x*00010					
*x*00011					
*x*00100					
*x*00101					
*x*00110					
*x*00111					
*x*01000	0*x*0100				
*x*01001					
*x*01010					
*x*01011					
*x*01100					
*x*01101					
*x*01110					
*x*01111					
*x*10000			000*x*10	0000*x*1	10000*x*
*x*10001			001*x*10	0001*x*1	10001*x*
*x*10010			010*x*10	0010*x*1	10010*x*
*x*10011			011*x*10	0011*x*1	10011*x*
*x*10100				0100*x*1	10100*x*
*x*10101				0101*x*1	10101*x*
*x*10110				0110*x*1	10110*x*
*x*10111				0111*x*1	10111*x*
*x*11000	0*x*1100				11000*x*
*x*11001					11001*x*
*x*11010					11010*x*
*x*11011					11011*x*
*x*11100					11100*x*
*x*11101					11101*x*
*x*11110					11110*x*
*x*11111					11111*x*

.

H_1 contains faulty link $010x10$ in dim-3. The first "unflipped" $k \leqslant 2$ is 2. A $BF(2)$ is performed on H_1, and the result sent to H_2. The updated *bf* and H_2 are shown below:

bf =

6	5	4	3	2	1
unflipped	*flipped*	*flipped*	*flipped*	*flipped*	*unflipped*

H_2 =

dim-6	*dim*-5	*dim*-4	*dim*-3	*dim*-2	*dim*-1
x00000			000x00		
x00001			001x00		
x00010			010x00		
x00011			011x00		
x00100					
x00101					
x00110					
x00111					
x01000		00x010			
x01001		01x010			
x01010					
x01011					
x01100	0x0110				
x01101					
x01110					
x01111					
x10000				0000x1	10000x
x10001				0001x1	10001x
x10010				0010x1	10010x
x10011				0011x1	10011x
x10100				0100x1	10100x
x10101				0101x1	10101x
x10110				0110x1	10110x
x10111				0111x1	10111x
x11000					11000x
x11001					11001x
x11010					11010x
x11011					11011x
x11100	0x1110				11100x
x11101					11101x
x11110					11110x
x11111					11111x

.

H_2 contains faulty link $00x010$ in dim-4. The first "unflipped" $k \leqslant 3$ is 1. A $BF(1)$ is performed on H_2, and the result sent to H_3. The updated *bf* and H_3 are shown below:

bf =

6	5	4	3	2	1
unflipped	*flipped*	*flipped*	*flipped*	*flipped*	*flipped*

H_3 =

dim-6	*dim*-5	*dim*-4	*dim*-3	*dim*-2	*dim*-1
$x00000$				$0000x0$	
$x00001$				$0001x0$	
$x00010$				$0010x0$	
$x00011$				$0011x0$	
$x00100$				$0100x0$	
$x00101$				$0101x0$	
$x00110$				$0110x0$	
$x00111$				$0111x0$	
$x01000$			$000x01$		
$x01001$			$001x01$		
$x01010$			$010x01$		
$x01011$			$011x01$		
$x01100$		$00x011$			
$x01101$		$01x011$			
$x01110$	$0x0111$				
$x01111$					
$x10000$					$10000x$
$x10001$					$10001x$
$x10010$					$10010x$
$x10011$					$10011x$
$x10100$					$10100x$
$x10101$					$10101x$
$x10110$					$10110x$
$x10111$					$10111x$
$x11000$					$11000x$
$x11001$					$11001x$
$x11010$					$11010x$
$x11011$					$11011x$
$x11100$					$11100x$
$x11101$					$11101x$
$x11110$	$\boxed{0x1111}$				$11110x$
$x11111$					$11111x$

.

Finally, H_3 contains faulty link $0x1111$ in dim-5. The only "unflipped" k left is 6. $BF(6)$ is performed on H_3. We arrive at the fault-tree H_4:

$$H_4 = \begin{bmatrix}
\underline{dim\text{-}6} & \underline{dim\text{-}5} & \underline{dim\text{-}4} & \underline{dim\text{-}3} & \underline{dim\text{-}2} & \underline{dim\text{-}1} \\
x00000 & & & & & 00000x \\
x00001 & & & & & 00001x \\
x00010 & & & & & 00010x \\
x00011 & & & & & 00011x \\
x00100 & & & & & 00100x \\
x00101 & & & & & 00101x \\
x00110 & & & & & 00110x \\
x00111 & & & & & 00111x \\
x01000 & & & & 1000x0 & 01000x \\
x01001 & & & & 1001x0 & 01001x \\
x01010 & & & & 1010x0 & 01010x \\
x01011 & & & & 1011x0 & 01011x \\
x01100 & & & 100x01 & 1100x0 & 01100x \\
x01101 & & & 101x01 & 1101x0 & 01101x \\
x01110 & & 10x011 & 110x01 & 1110x0 & 01110x \\
x01111 & 1x0111 & 11x011 & 111x01 & 1111x0 & 01111x \\
x10000 & & & & & \\
x10001 & & & & & \\
x10010 & & & & & \\
x10011 & & & & & \\
x10100 & & & & & \\
x10101 & & & & & \\
x10110 & & & & & \\
x10111 & & & & & \\
x11000 & & & & & \\
x11001 & & & & & \\
x11010 & & & & & \\
x11011 & & & & & \\
x11100 & & & & & \\
x11101 & & & & & \\
x11110 & & & & & \\
x11111 & 1x1111 & & & &
\end{bmatrix}.$$

3.4. $|F \cap E| = n-1$

If all $n-1$ faulty links are regular links, i.e., $|F \cap E| = n-1$, then *FT_HAMIL* as described in Section 3.3 may not work. For example, if H_4 in the example of Section 3.3 still contains a faulty link, then *BF* at any bit will not produce another *totally* new fault-free cycle—some links that were "switched out" in a previous *BF* operation may be "switched back."

Since all $n-1$ faulty links are regular links, we have $F \cap S = \emptyset$; i.e., all skips are fault-free. With a slight change in dimension assignment, we can still make use of *FT_HAMIL* to produce a fault-free Hamiltonian cycle that uses all skips, and avoids all faulty links in E.

Now that there are $n-1$ faulty regular links (and no faulty skips), there will be one dimension that contains no faulty links at all. We can always assign dimensions to links with the following distribution of faults:

$$f_1 = 0; \tag{3.4-1}$$

$$0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant f_n \leqslant n-1. \tag{3.4-2}$$

Note that because dim-n contains the largest number of faulty links,

$$0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant n-2 \tag{3.4-3}$$

still holds.

With this dimension assignment, we apply algorithm *HAMIL*, followed by *FT_HAMIL*. Because of (3.4-3), the result H_j, $1 \leqslant j \leqslant (n-2)$, will be a Hamiltonian cycle that contains no faulty links in dim-2 through dim-$(n-1)$, but still contains f_n faulty links in dim-n. For example, H_2 may look like the following:

$$H_2 = \left[\begin{array}{cccccc}
\underline{dim\text{-}6} & \underline{dim\text{-}5} & \underline{dim\text{-}4} & \underline{dim\text{-}3} & \underline{dim\text{-}2} & \underline{dim\text{-}1} \\
x00000 & & & 000x00 & & \\
x00001 & & & 001x00 & & \\
x00010 & & & 010x00 & & \\
x00011 & & & 011x00 & & \\
x00100 & & & & & \\
\boxed{x00101} & & & & & \\
x00110 & & & & & \\
x00111 & & & & & \\
x01000 & & 00x010 & & & \\
x01001 & & 01x010 & & & \\
x01010 & & & & & \\
x01011 & & & & & \\
\boxed{x01100} & 0x0110 & & & & \\
x01101 & & & & & \\
x01110 & & & & & \\
x01111 & & & & & \\
x10000 & & & & 0000x1 & 10000x \\
x10001 & & & & 0001x1 & 10001x \\
x10010 & & & & 0010x1 & 10010x \\
x10011 & & & & 0011x1 & 10011x \\
x10100 & & & & 0100x1 & 10100x \\
x10101 & & & & 0101x1 & 10101x \\
x10110 & & & & 0110x1 & 10110x \\
x10111 & & & & 0111x1 & 10111x \\
x11000 & & & & & 11000x \\
x11001 & & & & & 11001x \\
x11010 & & & & & 11010x \\
x11011 & & & & & 11011x \\
x11100 & 0x1110 & & & & 11100x \\
x11101 & & & & & 11101x \\
x11110 & & & & & 11110x \\
x11111 & & & & & 11111x
\end{array}\right].$$

In Fig. 5, we redraw the three 4-cube-induced Hamiltonian cycles from Fig. 3, with dim-4 links drawn with dashed lines. It can be observed that dim-4 links play a special role: In all Hamiltonian cycles, all dim-4 links are used, and they appear alternatively in a cycle (i.e., all non-dim-4 links are connected only to dim-4 links). This observation is also true for a general n-cube. That is also the reason why *FT_HAMIL* cannot "switch out" any faulty links of dim-n—all of them are needed in forming a Hamiltonian cycle.

PROPOSITION 3.1. *In all Hamiltonian cycles generated with HAMIL and then FT_HAMIL, all dim-n links will be used, and they appear alternatively in the cycles.*

Proof. We only have to show that in all Hamiltonian cycles generated by *HAMIL* and *FT_HAMIL*, (1) every link in dim-i, $1 \leqslant i \leqslant n-1$, is connected with two links in dim-n; and (2) for any two links a of dim-i and b of dim-j, $1 \leqslant i < j \leqslant n-1$, a and b will not be connected.

(1) A link generated by *HAMIL* and then *FT_HAMIL* in dim-i has the general format $b_n..b_{i+1}xb_{i-1}..b_1$. The two nodes linked by it are then $b_n..b_{i+1}0b_{i-1}..b_1$ and $b_n..b_{i+1}1b_{i-1}..b_1$. However, node $b_n..b_{i+1}0b_{i-1}b_1$ is connected with dim-n link $x..b_{i+1}0b_{i-1}..b_1$ and node $b_n..b_{i+1}1b_{i-1}..b_1$ is connected with dim-n link $x..b_{i+1}1b_{i-1}..b_1$.

When *FT_HAMIL* uses bit-flip to generate H_1, H_2, etc., the links in dim-n remain unchanged. The above argument still holds.

(2) Observe H_0, generated by *HAMIL*.

$i=1$: A link a of dim-1 has "1" at dimension n, but all links b in dim-j, $2 \leqslant j \leqslant n-1$, have "0" (the complementary value of "1") at dimension n. Therefore a and b will not connect to a common node.

$2 \leqslant i \leqslant n-2$: A link a of dim-i has "1" at dimension $i-1$, but all links b in dim-j, $i+1 \leqslant j \leqslant n-1$, have "0" at dimension $i-1$. Therefore a and b will not connect to a common node.

Since algorithm *FT_HAMIL* uses bit-flip, the complementary property at each dimension (bit position) will be preserved. Therefore, for H_1, H_2, etc., (2) still holds. ∎

Now, if we remove all dim-4 links in Fig. 5, and add all skips, we can still get Hamiltonian cycles, in which all skips appear alternatively in the cycles. The three Hamiltonian cycles that use no dim-4 links but all of the skips are shown in Fig. 6.

Since $|F \cap E| = n-1$, there are no faulty skips. *HAMIL* and *FT_HAMIL* can generate a fault-free Hamiltonian cycle using good links in dim-1 through dim-$(n-1)$, and all skips.

3.5. *Summary*

From the discussion of 3.3 and 3.4, the final algorithm to find a link-fault-free Hamiltonian cycle in a folded n-cube can be summarized as follows:

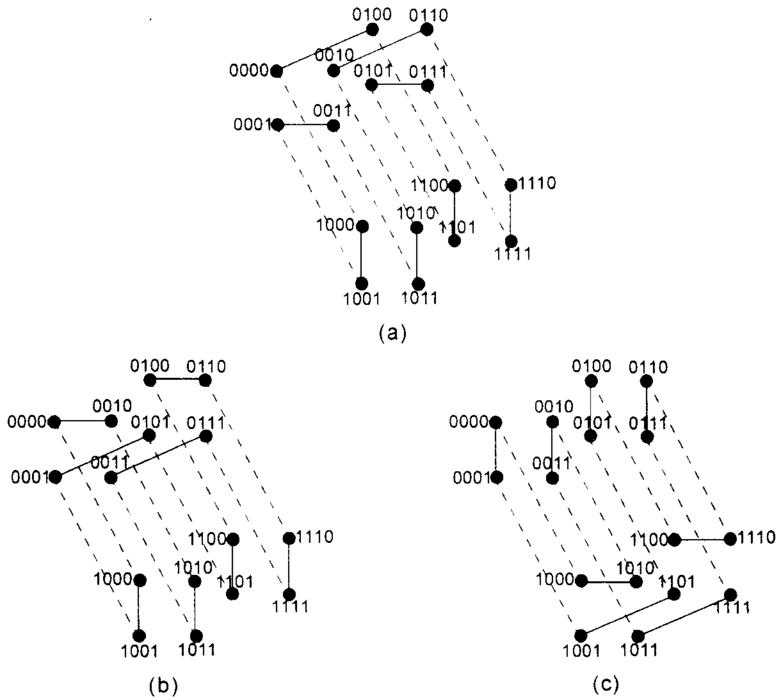

FIG. 5. Three Hamiltonian cycles induced from a 4-cube.

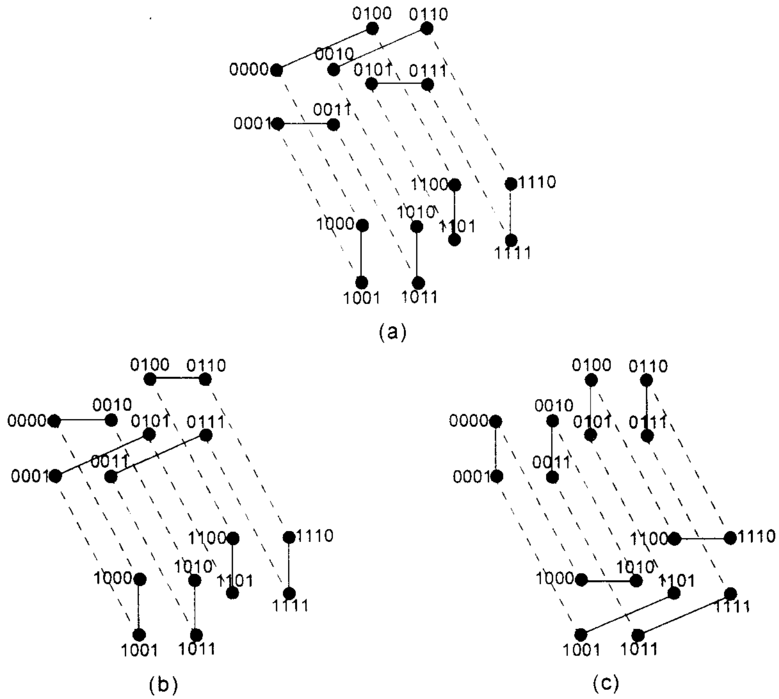

FIG. 6. Three Hamiltonian cycles that use no dim-4 links but all skips.

If ($|F \cap E| \leqslant n-2$) /* less than $n-1$ regular links are faulty */
{
  1. assign dimensions to links so that
     $f_1 = f_n = 0$, and $0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant n-2$
  2. perform *HAMIL*
  3. perform *FT_HAMIL*
  { producing a Hamiltonian cycle using only good regular links }
}
else /* ($|F \cap E| = n-1$), $n-1$ regular links are faulty */
{
  1. assign dimensions to links so that
     $f_1 = 0$, and $0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant f_n \leqslant n-1$
  2. perform *HAMIL*
  3. perform *FT_HAMIL*
  4. remove all dim-n links and add all skip links
  { producing a Hamiltonian cycle, using good regular links in dim-1 through dim-($n-1$) and using all skips }
}

4. CONCLUSION

This paper studied the link-fault tolerability of the folded hypercube with respect to Hamiltonian cycle embedding. It has been shown that a folded n-cube can tolerate $n-1$ faulty links when embedding a Hamiltonian cycle, whereas its regular counterpart is known to be able to tolerate $n-2$ faulty links. A succinct algorithm *FT_HAMIL* was presented that finds a Hamiltonian cycle in a folded n-cube in the presence of up to $n-1$ faulty links regardless of their distribution. For a folded n-cube, $n-1$ is the maximum number of faulty links that can be tolerated—if n faulty links were allowed, and all of them were incident on the same node, there would be no Hamiltonian cycles.

APPENDIX

Proof for FT_HAMIL. We prove by explaining the purpose of each statement of *FT_HAMIL*.

Statements (1) and (2). Observe H_0 from the example in Section 3.3. It can be generalized that $BF(n-1)$ and $BF(n-2)$ do not produce any new link set, because a new set will be produced by $BF(i)$ *only if* at some dimension, bit-i are all "1" or all "0." (For example, the links in dim-1 are all "1" at bit-6. Therefore $BF(6)$ produces a new link set, but at bit-5 and bit-4, links in all columns have both "1" and "0." Therefore $BF(5)$ and $BF(4)$ will not produce a new link set.) So we exclude bit-($n-1$) and bit-($n-2$) for BF operations. That is the purpose of line (2).

Statement (3) is self-evident.

Statement (4). Start of the while loop. In each round of the loop, at least one faulty link of H_i is avoided by $BF(k)$ at some bit-k. If there is no faulty link in H_i, *FT_HAMIL* terminates.

Statement (5). Check, in each dimension of links, to see if there are any faulty links. If there are any faulty links in a dim-j, statements (7) through (12) are carried out.

Note that the number of dimensions affected by $BF(i)$ varies as i varies. For example, for H_0, $BF(1)$ will switch links in dim-2, 3, 4, and 5; while $BF(3)$ will switch links only in dim-4 and 5. To generalize,

$$BF(n) \text{ will switch links in dim-1, 2,..., } (n-2), (n-1); \tag{A-1}$$

$$BF(n-1) \text{ and } BF(n-2) \text{ do not switch links in any dimension:} \tag{A-2}$$

$$\text{For all } i \leqslant (n-3), BF(i) \text{ will switch links in dim-}(i+1), (i+2), ..., (n-1). \tag{A-3}$$

That is to say, if a faulty link e_f is found in dim-j, then to “switch out” e_f, a BF must be performed either at a bit-k such that $k \leqslant j-1$ or at bit-n.

Statements (7) and (8). Based on above reasoning, if dim-j contains a faulty link, find an unflipped bit-k, so that $BF(k)$ will switch the links in dim-$(k+1)$, ..., j, ..., $(n-1)$.

Statement (9). The $BF(k)$ operation. A new Hamiltonian cycle is generated that does not use the faulty link found in dim-j.

Statement (10). Bit-k is marked as “flipped” so that any future BF will not be performed at bit-k. This guarantees that for any k, $BF(k)$ will be performed *at most once*. This way, every BF performed will produce a really new link set over all—a “switched out” faulty link in a previous BF will never be “switched back” in any future BF. Since there are $n-2$ bits marked as “unflipped” initially, *FT_HAMIL* will perform at most $n-2$ BF’s.

Statements (11) and (12). Statements (6) through (10) produced a new Hamiltonian cycle H_{i+1}, avoiding *at least* one faulty link found in H_i. Statements (11) and (12) prepare for the next round of loop.

Finally, we explain the purpose of dimension reassignment, i.e., why we want to readdress the folded n-cube in such a way that

$$f_1 = f_n = 0; \tag{A-4}$$

$$0 \leqslant f_2 \leqslant f_3 \leqslant \cdots \leqslant f_{n-1} \leqslant n-2. \tag{A-5}$$

From (A-1)–(A-3), we can see that

$$\text{Links of dim-1 will get switched with only } BF(n); \tag{A-6}$$

$$\text{Links of dim-2 will get switched with only } BF(1) \text{ and } BF(n); \tag{A-7}$$

$$\text{Links of dim-3 will get switched with } BF(1), BF(2), \text{ and } BF(n); \tag{A-8}$$

$$\cdots$$

$$\text{Links of dim-}(n-3) \text{ will get switched with } BF(1), ..., BF(n-4), \text{ and } BF(n); \tag{A-9}$$

Links of dim-$(n-2)$will get switched with $BF(1)$, ..., $BF(n-3)$, and $BF(n)$; i.e., every BF performed will switch dim-$(n-2)$ links; (A-10)

Links of dim-$(n-1)$ will also get switched with every BF performed: $BF(1)$, $BF(2)$, ..., $BF(n-3)$, $BF(n)$. (A-11)

What (A-7) implies is that if there are more than two faulty links in dim-2, we may not be able to switch them all out. In the worst case, $BF(1)$ will switch one faulty link out, $BF(n)$ will switch another faulty link out. The third faulty link will not be switched out by any other BFs, so dim-2 should contain at most 2 faulty links. Similarly, by (A-8), dim-3 should contain at most 3 faulty links, etc. To generalize,

dim-i should contain at most i faulty links, $2 \leqslant i \leqslant (n-2)$; (A-12)

dim-$(n-1)$ should contain at most $(n-2)$ faulty links. (A-13)

If faulty links are distributed so that (A-4)–(A-5) are satisfied, then dim-i will contain at most $\lfloor (n-2)/(n-i) \rfloor$ faulty links, $1 \leqslant i \leqslant (n-1)$, but

$$\left\lfloor \frac{n-2}{n-i} \right\rfloor < i$$

always holds when $1 \leqslant i \leqslant (n-2)$, satisfying (A-12). When $i = n-1$,

$$\left\lfloor \frac{n-2}{n-i} \right\rfloor = (n-2),$$

satisfying (A-13). ∎

REFERENCES

1. A. El-Amawy and S. Latifi, Properties and performance of folded hypercubes, *IEEE Trans. Parallel and Distrib. Syst.* **2** (Jan. 1991), 31–42.
2. A. H. Esfahanian, L. M. Ni, and B. E. Sagan, The twisted *N*-cube with application to multiprocessing, *IEEE Trans. Comput.* **40** (Jan. 1991), 88.
3. W. D. Hillis, "The Connection Machine," MIT Press, Cambridge, MA, 1985.
4. J. P. Hayes, T. N. Mudge, and Q. F. Stout, Architecture of a hypercube supercomputer, *in* "Proc. 1986 International Conference on Parallel Processing," pp. 653–660, Aug. 1986.
5. Intel Corporation, "iPSC System Overview" (Jan. 1986).
6. S. Latifi, S. Q. Zheng, and N. Bagherzadeh, Optimal ring embedding in hypercubes with faulty links, *in* "Proc. 22th IEEE International Symposium Fault-Tolerant Computing" pp. 178–184, July 1992.
7. F. P. Preparata and J. Vuillemin, The cube-connected cycles: A versatile network for parallel computation, *Commun. ACM* **24** (1981), 300–309.
8. C. L. Seitz, The cosmic cube, *Commun. ACM* **28** (1985), 22–23.
9. N. F. Tzeng and S. Wei, Enhanced hypercubes, *IEEE Trans Comput.* **40** (1991), 284–294.
10. D. Wang, Diagnosability of enhanced hypercubes, *IEEE Trans Comput.* **43** (Sept. 1994), 1054–1061.
11. D. Wang and Z. Wang, Minimum assignment of test links for hypercubes with lower fault bounds, *J. Parallel Distrib. Comput.* **40** (Feb. 1997), 185–193.