Distributed Fault-Tolerant Embeddings of Rings in Incrementally Extensible Hypercubes with Unbounded Expansion

Jen-Chih Lin[1], Steven K.C. Lo[1], Shih-Jung Wu[2]* and Huan-Chao Keh[2]

[1]*Department of Information Management, Jin-Wen Institute of Technology, Taipei, Taiwan 231, R.O.C.*
[2]*Department of Information Science & Information Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.*

Abstract

The Incrementally Extensible Hypercube (IEH) is a generalization of interconnection network that is derived from the hypercube. Unlike the hypercube, the IEH can be constructed for any number of nodes. That is, the IEH is incrementally expandable. In this paper, the problem of embedding and reconfiguring ring structures is considered in an IEH with faulty nodes. There are a novel embedding algorithm proposed in this paper. The embedding algorithm enables us to obtain the good embedding of a ring into a faulty IEH with unbounded expansion, and such the result can be tolerated up to $O(n*\lfloor log_2 m \rfloor)$ faults with congestion *1*, load *1*, and dilation *4*. The presented embedding methods are optimized mainly for balancing the processor loads, while minimizing dilation and congestion as far as possible.

Key Words: Incrementally Extensible Hypercube (IEH), Fault-Tolerant, Embedding, Linear Array, Ring

1. Introduction

The *n*-dimensional hypercube (*n*-cube) is one of the most popular interconnection topologies for parallel computers. One of the main reasons for the popularity of hypercube architecture is its ability to efficiently simulate other architectures. If composite hypercubes are to be competitive as an architecture, we must demonstrate similar simulation capabilities. Hypercube-based parallel machines are built and sold commercially, and it is expected that they will continually play an important role in the future. One of the most important issues related to such parallel machines is how they can compute in the presence of faults. Hypercube popularity may be attributed to its regular structure and the requirements of its rich interconnection topology, and the number of nodes must be a power of *2*. In order to alleviate this shortcoming, several 'incomplete' hypercube-like architectures have been proposed.

The issue of computing with faulty hypercubes has been addressed in several recent papers [1−8]. And the results proposed by Hastad, Leighton and Newman [9] should be paid particular attention. They considered that in a faulty hypercube, every node is faulty with constant probability $p < 1$ and the faults are independently distributed. They proved that, with high probability, the faulty hypercube can simulate a fault-free hypercube with only a constant factor slowdown. Thus the hypercube is extremely tolerant of randomly distributed faults.

The problem of embedding an *n*-processor guest network *G* into an *n*-processor host network *H* is an important problem in distributed computing or parallel processing. Results on this problem not only demonstrate computational equivalence or non-equivalence) between networks of different topology, but also lead to efficient simulations

*Corresponding author. E-mail: 890190084@s90.tku.edu.tw

of algorithms originally designed for G on host H. Embedding and their implications to distributed computing or parallel processing have been studied extensively recently.

Graph embeddings have been used successfully to show simulation capabilities of a guest architecture by another host architecture [10,11]. In graph embedding techniques, host and guest architectures are viewed as graphs H and G, respectively, and then the graph G is embedded into the graph H. In the embedding of a graph G into H, we map the set of nodes of G into the set of noses of H and the edges of G to paths in H which connect the image of the nodes of G. In order to obtain efficient simulations of G by H, various cost measures of an embedding must be optimized. One such measure is the *dilation* of an embedding. The dilation of an edge of G is the length of the path onto which an edge of G is mapped. The dilation of the embedding is the maximum edge dilation of G. The *expansion* of the embedding is the ratio of the number of nodes in G to the number of nodes in H. The *congestion* is defined as the maximum number of paths over an edge in H, where every path represents an edge in G. The *load* is defined as the maximum number of nodes of G assigned to any node of H. We say that an embedding achieves a *balanced load* when *load = 1*.

In this paper, we study how algorithms that are designed for fault-tolerance *Incrementally Extensible Hypercube* can be implemented on *Incrementally Extensible Hypercubes* that contain faults. In the following discussion we will consider a parallel computer as a graph, in which the nodes correspond to processors and the edges correspond to communication links.

The remainder of this paper is organized as follows. Section 2 is devoted to some notations and definitions. The construction of the ring in an IEH is addressed in Section 3. Section 4 develops the embedding algorithm to a faulty IEH with unbounded expansion. Section 5 concludes this paper.

2. Preliminaries

We briefly describe notations and definitions of the IEH graph. The IEH graph is the composition of some m different hypercubes. Let $G_n(N)$ be an IEH graph with N nodes, and N can be expressed by the binary string $N = b_nb_{n-1}b_{n-2}\ldots b_1b_0$, and $b_i \in \{0,1\}$. An IEH graph $G_n(N)$ is composed of some different hypercubes which have lower dimension than $G_n(N)$ has. That is, $G_n(N)$ contains a hypercube, denoted by H_i, if and only if the i_{th} bit in the binary representation of N is *1*.

Accordingly, the IEH graph is composed of some hypercubes, so there is a new type of connections beside the usual connections in a hypercube. These edges (or links) are used for connecting two hypercubes are called *Inter-Cube* or *IC* edges. The basic philosophy in the design of the IEH graphs is to express N as a sum of several powers of *2*, i.e., to write N as a binary number, build the smaller hypercubes, and then to add appropriate inter-cube edges to connect those smaller hypercubes. For any given N, $2^n \le N < 2^{n+1}$, the steps of finding IEH graphs are as follows.

Step 1 Build subcube graphs. Express N as $(n+1)$ bits a binary number as $N = b_nb_{n-1}b_{n-2}\ldots b_1b_0$, where $b_i \in \{0,1\}$ and $b_n = 1$ since $N \ge 2^n$. For each b_i, $b_i \ne 0$, construct a hypercube graph H_i with 2^i nodes.

Step 2 Label the nodes. Note that each node has a $(n+1)$-bit binary label. Each hypercube H_i is labeled as $11\ldots10b_{i-1}b_{i-2}\ldots b_1b_0$. Obviously each hypercube of dimension i (having 2^i nodes) has i number of dashed and the individual nodes of the hypercube can be obtained by filling the dashes with *0* or *1* in all possible ways. In other words, the binary representation of each node in H_i has the same prefix of $(n-i)$*1*'s followed by a single zero.

Step 3 Construct the incremental hypercube in steps by providing the inter-cube edges. Find the minimum i such that $b_i \ne 0$. Set $j = i$ and $G_j = H_i$.

Set $i = i + 1$.

While $i \le n$ do

if $b_i \ne 0$ then

if $i - j = 1$ then

each node x in G_j with label $11\ldots b_jb_{j-1}\ldots b_0$ is connected to the node $11\ldots10b_jb_{j-1}\ldots b_0$ of H_i.

else

each node x in G_j with label $11\ldots1b_jb_{j-1}\ldots b_0$ is connected to $(i-j)$ different nodes of H_i chosen in the following way:

$$\begin{cases} \overbrace{11\ldots11}^{n-i}0\overbrace{11\ldots11}^{i-j-1}b_jb_{j-1}\ldots b_0 \\ \overbrace{11\ldots11}^{n-i}0\overbrace{01\ldots11}^{i-j-1}b_jb_{j-1}\ldots b_0 \\ \overbrace{11\ldots11}^{n-i}0\overbrace{10\ldots11}^{i-j-1}b_jb_{j-1}\ldots b_0 \\ \quad\vdots\quad\ \vdots\quad \ddots \qquad\qquad \vdots \\ \overbrace{11\ldots11}^{n-i}0\overbrace{11\ldots01}^{i-j-1}b_jb_{j-1}\ldots b_0 \\ \overbrace{11\ldots11}^{n-i}0\overbrace{11\ldots10}^{i-j-1}b_jb_{j-1}\ldots b_0 \end{cases}$$

Set $j = i$ and set G_j to be the composite graph generated in the previous steps. Note that G_j has now $\sum_{k=0}^{j} b_k 2^k$ nodes and the binary label of each node in G_j has a prefix of *(n–j) 1*'s.

$i = i + 1$

Return G_n as the desired incremental hypercube graph of N vertices.

Figure 1.1 shows the example of $G_3(14)$. $G_3(14)$ consists of three subcubes. The three subcubes are *1*-subcube (H_1), *2*-subcube (H_2), and *3*-subcube (H_3). Nodes *12* and *13* are composed as a *1*-subcube (H_1), Nodes *8*, *9*, *10*, and *11* are composed as a *2*-subcube (H_2), and nodes *0*, *1*, *2*, *3*, *4*, *5*, *6* and *7* are the elements of a *3*-subcube (H_3). The edges *(8, 12)*, *(9,13)* are IC edges connected between H_1 and H_2 such that H_1 and H_2 are connected to be an IEH graph containing *6* nodes ($G_2(6)$). In addition, the H_3 connects to $G_2(6)$ with these IC edges (*0, 8*), (*1, 9*), *(2,10)*, *(3,11)*, *(4,12)*, and *(5,13)*.

Definition 2.1 [3] The Hamming distance between two nodes with labels $x = x_{n-1}x_{n-2}...x_0$ and $y = y_{n-1}y_{n-2}...y_0$ is defined as

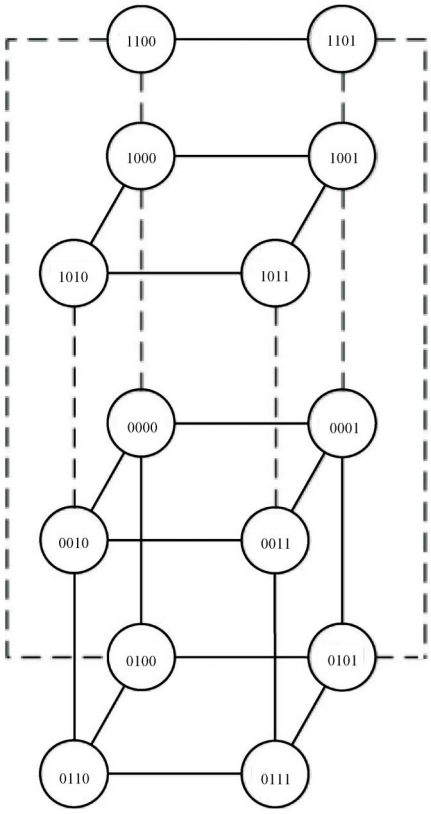


Figure 1.1. The IEH graph contains 14 nodes.

$$HD(x, y) = \sum_{i=0}^{n-1} hd(x_i, y_i), \text{ where } hd(x_i, y_i) = \begin{cases} 0, \text{if} x_i = y_i, \\ 1, \text{if} x_i \neq y_i. \end{cases}$$

Definition 2.2 [3] Let $x = x_{n-1}...x_0$, $y = y_{n-1}...y_0$, then $Dim(x, y) = \{i \text{ in } (0...n-1)x_i \neq y_i\}$

Definition 2.3 [3] The *Binary-Reflected Gray Code* (BRGC) is defined recursively as follows.

$C_{n+1} = \{0C_n, 1(C_n)^R\}$, where $C_1 = \{0, 1\}$ and $C_2 = \{0C_1, 1(C_1)^R\}$

For example, a *2*-bit Gray Code can be constructed by the sequence, defined in definition 2.3, and insert a cipher in front of each codeword in C_1, then insert an one in front of each codeword in $(C_1)^R$. We get the code $C_2 = \{00, 01, 11, 10\}$. Now, we can then repeat the procedure to built a *3*-bit Gray Code, and also get the code $C_3 = 0C_2 \cup 1(C_2)^R = \{000, 001, 011, 010, 110, 111, 101, 100\}$.

Definition 2.4 [4] If G is a graph, the vertex set of G is denoted by V and the edge set of G is denoted by E. A graph G' is said to be a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$.

Definition 2.5 [3] Suppose that an IEH graph contains N nodes, then H_i and H_j are two different-sized subcubes in the IEH graph, assume $i < j$. Let (p,q) is an edge in H_i, and (r,s) is an edge in H_j. If r and s are image nodes, in H_j, of p and q respectively, then the edge (r,s) is called an image edge, in H_j, of the edge (p,q).

Because (p,q) is an edge of H_i, where $HD(p,q) = 1$ and $Dim(p,q) = \{0\}$. According to the steps of finding IEH graphs $HD(r,s) = 1$ and $Dim(r,s) = \{0\}$, so (r,s) is an edge in H_j.

3. Rings Embedding

Almost all of IEH graphs, except for those with $N = 2^n - 1$ nodes, have a *Hamiltonian cycle*; if an IEH graph with $N = 2^n - 1$ nodes then it has only a *Hamiltonian path*, not cycle. The IEH graphs $G_0(1)$, $G_1(2)$, and $G_1(3)$ can obtain no cycle, because they are graphs with no cycle. Therefore, the number of nodes of an IEH graph contains a Hamiltonian cycle must be more than *3*. Besides, in the properties of the IEH graph, there is no Hamiltonian cycle in an IEH graph of $2^n - 1$ nodes.

Lemma 3.1 [12] Suppose that H_n is an n-dimensional hypercube, then the permutation of nodes in H_n as the sequence in a BRGC C_n is a Hamiltonian path. Consequently, a Hamiltonian cycle exists in a hypercube.

According to lemma 3.1, we know how to find a Hamiltonian cycle or path in a hypercube. It is simple that following the sequence of n-bit BRGC can find the Hamiltonian path and cycle in a hypercube H_n.

Lemma 3.2 [12] A ring R_r of length r can be mapped into the n-cube when r is even and $4 \le r \le 2^n$.

Lemma 3.3 [12] There is no cycles of odd length in a hypercube.

Lemma 3.4 [3] An IEH graph $G_{n-1}(N)$ only contains a Hamiltonian path and no Hamiltonian cycle for all $N \ge 3$ and $N = 2^n-1$.

Theorem 3.1 An IEH graph contains a Hamiltonian cycle for all $N \ge 4$ and $N \ne 2^n-1$.

Proof. Let $G_n(N)$ be an n-dimensional IEH graph with N nodes, and N can be expressed by the binary string $N = b_nb_{n-1}b_{n-2}\ldots b_1b_0$, where $b_i \in \{0,1\}$. We consider two cases.

Case1: N is even. Because N is even, $b_0 = 0$. In other words, H_0 does not exist. Let H_i and H_j be two adjacent subcubes in an IEH graph $G_n(N)$, where $i < j$. Let (p,q) is an edge in H_i, there exists an image edge (r,s) of (p,q) in H_j such that (p,r) and (q,s) is two IC edge. Note that r and s will differ in the same bit that p and q. We find a Hamiltonian cycle using the BRGC method in each subcube H_k, $0 < k \le n$, except subcube H_1. Without loss of generality, (p,q) and (r,s) are an edge of a Hamiltonian cycle of H_i and H_j. We can combine H_i and H_j by IC edges (p,r) and (q,s). We combine all of cycles by IC edges between different-sized subcubes to form a large cycle. We can also add these nodes of H_1 into the cycle by the above method if H_1 exists in the IEH graph. The final cycle will be a Hamiltonian cycle of the IEH graph.

Case 2: N is odd. Since $N \ne 2^n-1$, there exists $b_m = 0$, where $m \ne 0$ and $m \ne n$. In other words, H_0 has at least two IC edges connecting to some nodes in H_k with $b_k = 1$ for $m < k$. Let r and s are the neighbor nodes of H_0 in H_k connected by the IC edges. By the steps of finding IEH graphs, r and s are adjacent. Without loss of generality, because the subcube H_k is symmetric, we can revise the Hamiltonian cycle such that (r,s) is an edge of a Hamiltonian cycle of H_k. We can combine H_k and H_0 by IC edges. We combine all of cycles by IC edges between different-sized subcubes to form a large cycle. The final cycle will be a Hamiltonian cycle of the IEH graph.

Lemma 3.5 [3] A linear array or a ring contains any number of nodes can be embedded into an IEH graph with dilation 2.

4. Fault-Tolerant Embedding with Unbounded Expansion

In the previous section, we have constructed a linear array and a ring into an IEH graph. In the section, we consider a faulty IEH with unbounded expansion embedding.

Theorem 4.1 A linear array or a ring can be mapped into an IEH graph with unbounded expansion.

Proof. It is trivial by lemma 3.5.

The cardinality of H_i, denoted by $|H_i|$, is number of nodes in H_i. Similarly, $|G_n(N)|$ is number of nodes in the IEH graph $G_n(N)$.

Theorem 4.2 Suppose $G_n(N)$ is an IEH graph contains N nodes, H_n is the maximal hypercube exists in $G_n(N)$, then $|H_n| > (N-|H_n|)$. On the other hand, if $G_n(N)$ is divided into two parts, H_n and $G_m(N-|H_n|)$, H_n contains more nodes than $G_m(N-|H_n|)$ does, where $0 \ m < n$.

Proof. Let $G_n(N)$ be an IEH graph contains $N = (a_{n-1}\ a_{n-2}\ldots a_0)$ nodes. It is composed by hypercubes H_i if $a_i \ne 0$ for $0 \le i \le n$. It is necessary that the most significant bit a_{n-1} must be equal to 1, so H_n is a part of $G_n(N)$ Because H_n is an n-dimensional hypercube, $|H_n| = 2^n$. The rest part of $G_n(N)$ is $G_m(N-|H_n|)$ which is possibility composed by $H_0, H_1,\ldots,$ and H_{n-1} if it is greatest, so the maximal number of nodes in $G_m(N-|H_n|)$ is $|H_0| + |H_1| + |H_2| +\ldots+ |H_{n1}| = 2^0 + 2^1 +\ldots + 2^{n-1} = 2^n-1$. As the result, $2^n = |H_n| \ge (N-|H_n|) = 2^n-1$.

Theorem 4.3 For an IEH $G_n(N)$, the subgraph H_n has an IC edge at most.

Proof. By the construction of IEH and theorem 4.2, all of nodes of $G_m(N)$, where $m < n$, has an unique IC edges connecting to H_n. Therefore, the subgraph H_n of an IEH $G_n(N)$ has an IC edge at most.

Algorithm LB_Embedding(x)
Input: x /*the faulty node*/, $Gn(N)$, R_m
Output: y /*the replaceable node*/
1. $i = 0$; $j = 0$; $k = 0$
2. Create a Queue Q; $Q = \Phi$
3. if a node x is faulty

4. then
5. {
6. while $i < (n + 1 - \lfloor \log_2 m \rfloor)$ do
7. {
8. search the node y /*HD(x, y) = 1, Dim(x, y) = $\lfloor \log_2 m \rfloor + i$*/
9. if *y* is not a virtual node and it is free
10. then
11. return(*y*) /*replace *x* with *y**/
12. remove all nodes in *Q*
13. exit()
14. else
15. enqueue(y, $\lfloor \log_2 m \rfloor + i$)
16. $i = i + 1$
17. }
18. }
19. while *Q* is not empty do
20. {
21. dequeue(a,b)
22. while $j < b$ do
23. {
24. search the node *y* /*$HD(a, y) = 1, Dim(a, y) = j$*/
25. if *y* is not a virtual node and it is free
26. then
27. return(*y*) /*replace *x* with *y**/
28. remove all nodes in *Q*
29. exit()
30. $j = j + 1$
31. }
32. }
33. search the node y /*(x, y) is a IC edge*/
34. if *y* is not a virtual node and it is free
35. then
36. return(*y*) /*replace *x* with *y**/
37. exit()
38. while $k < \lceil \log_2 m \rceil$ do
39. {
40. search the node *y* /*$HD(q, y) = 1, Dim(x, y) = k$*/
41. if *y* is not a virtual node and it is free
42. then
43. return(*y*) /*replace *x* with *y**/
44. exit()
45. $k = k + 1$
46. }
47. return ("Failure")
48. end

$node\ 0 = 0X_{n-1}X_{n-2}...X_{\lfloor \log_2 m \rfloor}...X_1X_0$

$node\ 1 = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor}...X_1X_0$

$node\ 2 = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor+1}X_{\lfloor \log_2 m \rfloor}...X_1X_0$

$\vdots$

$node\ (n-\lfloor \log_2 m \rfloor) = 0X'_{n-1}X_{n-2}...X_{\lfloor \log_2 m \rfloor}...X_1X_0$

$node\ (n-\lfloor \log_2 m \rfloor + 1) = 1X_{n-1}X_{n-2}...X_{\lfloor \log_2 m \rfloor}...X_1X_0$

$node\ (n-\lfloor \log_2 m \rfloor + 2) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor}...X_1X'_0$

$node\ (n-\lfloor \log_2 m \rfloor + 3) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor}...X'_1X_0$

$\vdots$

$node\ (n-\lfloor \log_2 m \rfloor + 1 + \lfloor \log_2 m \rfloor) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor}X'_{\lfloor \log_2 m \rfloor-1}...X_1X_0$

$node\ (n-\lfloor \log_2 m \rfloor + 1 + \lfloor \log_2 m \rfloor + 1) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor+1}...X_1X'_0$

$node\ (n-\lfloor \log_2 m \rfloor + 1 + \lfloor \log_2 m \rfloor + 2) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor+1}...X_1X'_0$

$\vdots$

$node\ (n-\lfloor \log_2 m \rfloor + 1 + 2*\lfloor \log_2 m \rfloor) = 0X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor+1}X_{\lfloor \log_2 m \rfloor}X'_{\lfloor \log_2 m \rfloor-1}...X_1X_0$

$\vdots$

$node\ ((n-\lfloor \log_2 m \rfloor + 1)*(\lfloor \log_2 m \rfloor + 1)) = 1X_{n-1}X_{n-2}...X'_{\lfloor \log_2 m \rfloor-1}...X_1X_0$

$node\ ((n-\lfloor \log_2 m \rfloor + 1)*(\lfloor \log_2 m \rfloor + 1) + 1) = Y_nY_{n-1}Y_{n-2}...Y_{\lfloor \log_2 m \rfloor}...Y_1Y_0$

(The IC edge connect node *0* and node $((n-\lfloor \log_2 m \rfloor + 1)*(\lfloor \log_2 m \rfloor + 1) + 1))$

$node\ ((n-\lfloor \log_2 m \rfloor + 1)*(\lfloor \log_2 m \rfloor + 1) + \lfloor \log_2 m \rfloor) = Y_nY_{n-1}Y_{n-2}...Y_{\lfloor \log_2 m \rfloor}...Y_1Y'_0$

$\vdots$

$node((n-\lfloor \log_2 m \rfloor + 1)*(\lfloor \log_2 m \rfloor + 1) + \lfloor \log_2 m \rfloor) = Y_nY_{n-1}Y_{n-2}...Y'_{\lfloor \log_2 m \rfloor-1}...Y_1Y_0$

We give a simple example in this section to explain the operations of the *LB_Embedding* algorithm when the faulty nodes exist. For the IEH $G_3(11)$ as Figure 4.1, where the R_6 has been embedded in it. The sequence of R_6 is *{0, 4, 5, 1, 3, 2}*.

1. If the node *0* is faulty, it visits or signals the node *4*, to check whether it is free or not. If it is, it terminates.
2. If not, insert the node *4* to the queue, and search the node *8*, to check whether it is free or not. If it is, it terminates.
3. If not, insert the node *8* to the queue, and delete the node *4* from the queue, search the node *5*, to check whether it is free or not. If it is, it terminates.

4.If not, search the node *6*, to check whether it is free or not. If it is, it terminates.

5.If not, delete the node *4* from the queue, search the node *9*, to check whether it is free or not. If it is, it terminates.

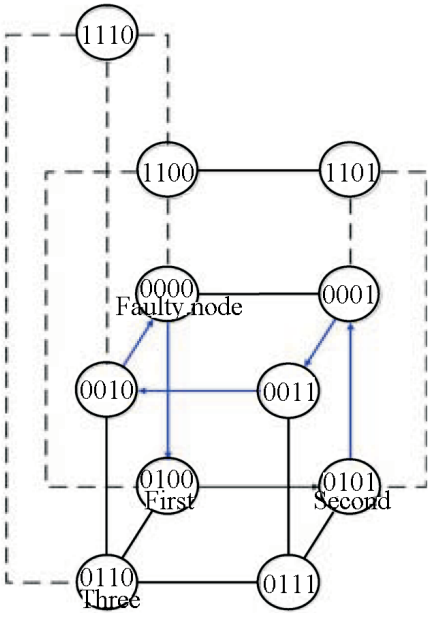
6.If not, return ("Failure").

Therefore, the whole searching path is listed as {4(0100), ~~8(1000)~~, 5(0101), 6(0110), ~~9(1001)~~, ~~10(1010)~~}.

We illustrate two examples of finding a replaceable node in an IEH graph $G_3(11)$ as shown Figure 4.1, Figure 4.2, and an examples of finding a replaceable node in an IEH graph $G_3(12)$ as shown Figure 4.3.

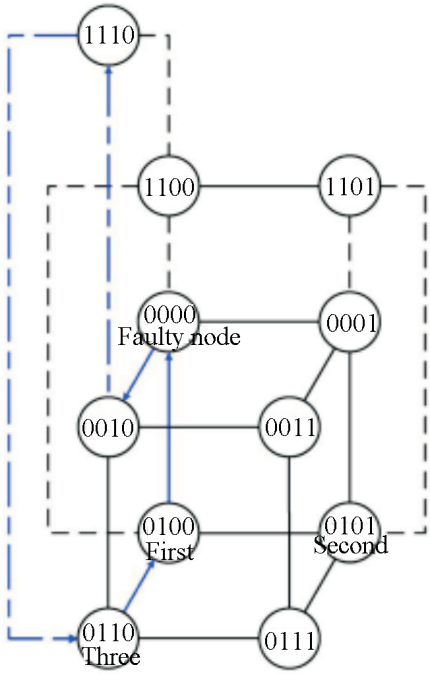
Theorem 4.3 A linear array or a ring R_m can be mapped into a faulty IEH $G_n(N)$ graph with dilation *4*, congestion *1*, and load *1*.

Proof. Every searching path is only one path according to the algorithm LB_Embedding(), allowing us to obtain congestion *1* and load *1*. Herein, we allow unbounded expansion to obtain the replaceable node of the faulty node. When a node is faulty and *m* is odd, it is a worse case in which the dilation = *2* + *2* = *4* at most by algorithm LB_Embedding() and lemma 3.5. Because these nodes and links of searching paths are not replicated from algorithm LB_Embedding(), These costs associated with graph embedding are dilation *4*, congestion *1*, and load *1*.



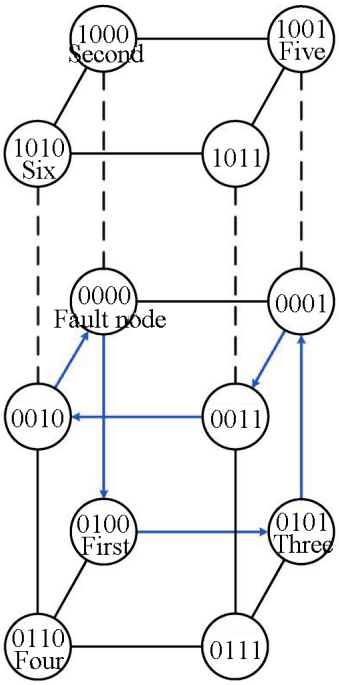
The searching path is 0100, ~~1000~~, 0101, 0110 ~~1001~~, ~~1010~~

Figure 4.1. Embedding R_6 into a H_3 of $G_3(11)$ with dilation *1*.



The searching path is 0100, ~~1000~~, 0101, 0110, ~~1001~~, ~~1010~~

Figure 4.2. Embedding R_5 into a H_3 of $G_3(11)$ with dilation *1*.



The searching path is 0100, 1000, 0101, 0110, 1001, 1010, ~~1100~~, ~~1101~~, ~~1110~~

Figure 4.3. Embedding R_5 into a H_3 of $G_3(12)$ with dilation *2*.

Theorem 4.4 A searching path of algorithm LB_Embedding() is including $((n-\lfloor log_2 m \rfloor + 1) + (n-\lfloor log_2 m \rfloor$

$+ 1)*\lfloor log_2 m \rfloor + \lfloor log_2 m \rfloor)$ nodes.

Proof. We can embed R_m into $G_n(N)$ by Theorem 4.3. If a node is faulty, we can change a bit in the binary string sequence from bit $\lfloor \log_2 m \rfloor$ to bit n and insert its corresponding node into the queue. In the worst case, we can get $(n-\lfloor log_2 m \rfloor + 1)$ different nodes. Then we delete the node from the queue. From the first node we can change a bit in the sequence from bit 0 to bit $(\lfloor log_2 m \rfloor - 1)$, and we can get $\lfloor log_2 m \rfloor$ different nodes. We can also change a bit in the sequence from bit 0 to bit $(\lfloor log_2 m \rfloor - 1)$ from the second node of the queue and we can also get from bit 0 to bit $(\lfloor log_2 m \rfloor - 1)$ different nodes. Until the queue is empty, the sum of all searched nodes is $(n-\lfloor log_2 m \rfloor + 1)*\lfloor log_2 m \rfloor$. The search path includes $(n-\lfloor log_2 m \rfloor + 1) + (n-\lfloor log_2 m \rfloor + 1)*\lfloor log_2 m \rfloor$ nodes. We assume there is an IC edge connecting to the faulty node. Therefore, we can search $\lfloor log_2 m \rfloor$ nodes in the worst case. We infer the edges of the replacing method exist and none of the nodes and the edges has a duplicate replacement. That is, the whole search path includes $(n-\lfloor log_2 m \rfloor + 1) + (n-\lfloor log_2 m \rfloor + 1)*\lfloor log_2 m \rfloor + \lfloor log_2 m \rfloor$ nodes.

Theorem 4.5 There are $O(n*\lfloor log_2 m \rfloor)$ faults, which can be tolerated.

Proof. By theorem 4.4, the whole search path includes $(n-\lfloor log_2 m \rfloor + 1) + (n-\lfloor log_2 m \rfloor + 1)*\lfloor log_2 m \rfloor + \lfloor log_2 m \rfloor = n*\lfloor log_2 m \rfloor - \lfloor log_2 m \rfloor^2 + \lfloor log_2 m \rfloor + n - \lfloor log_2 m \rfloor + 1 + \lfloor log_2 m \rfloor = n*\lfloor log_2 m \rfloor - \lfloor log_2 m \rfloor^2 + \lfloor log_2 m \rfloor + n + 1 nodes$. That is, $O(n*\lfloor log_2 m \rfloor)$ faults can be tolerated.

Theorem 4.6 Our results for the embedding methods are optimized mainly for balancing the processor and communication link loads.

Proof. Because these nodes and edges of searching paths are not replicated from the algorithm LB_Embedding() and load *1*, this observation implies that the primary optimization objective of mapping is to minimize the interprocessor communication cost and to balance the workload of processors having reached.

5. Conclusions

In this paper, we try to find the replaceable node of the faulty node. The main result of this paper is the fact that it is always possible to give solutions to the embedding of linear arrays and rings in a faulty IEH. After a ring is mapped into an IEH, we develop a novel algorithm to unbound expansion. And such costs are dilation *4*, congestion *1*, load *1*, and $O(n*\lfloor log_2 m \rfloor)$ faults can be tolerated. Furthermore, we can prove them and present some algorithms to solve them. According to the result, we can embed the parallel algorithms developed by the structure of a ring in an IEH. These methods of reconfiguring realize extremely high-speed parallel computation. This is an improvement of the results given in [8], when fault tolerance is of interest in a faulty hypercube. By the results, we can easily port the parallel or distributed algorithms developed for the structure of rings to the IEH graph.

References

[1] Armstrong, J. R. and Gray, F. G., "Fault Diagnosis in a Boolean n Cube Array of Microprocessors," *IEEE Trans. on Computers*, Vol. 30, pp. 587–590 (1981).

[2] Day, K. and Al-Ayyoub, A. E., "Fault Diameter of k-ary n-cube Networks," *IEEE Trans. on parallel and distributed systems*, Vol. 8, pp. 903–907 (1997).

[3] Jen-Chih Lin, "Simulation of Cycles in the IEH Graph," *International Journal of Hjgh Speed Computing*, Vo1. 10, pp. 327–342 (1999).

[4] Lin, J. C. and Keh, H. C, "Reconfiguration of Complete Binary Trees in Full IEH Graphs and Faulty Hypercubes," *International Journal of High performance computing Applications*, Vo1. 15, pp. 55–63 (2001).

[5] Lin, J. C and Hsien, N. C., "Reconfiguring Binary Tree Structures in a Faulty Supercube with Unbounded Expansion," *Parallel Computing*, Vo1. 8, pp. 471–483 (2002).

[6] Lin, J. C. and Lo, S. K. C., "Embedding Complete Binary Trees into Faulty Flexible Hypercubes with Unbounded Expansion" *Informatica*, Vol. 27, pp. 75–80 (2003).

[7] Rennels, D. A., "On Implemanting Fault-tolerance in Binary Hypercubes," *Proc. 16th Inter. Symp. on Fault-tolerant Computing*, pp. 344–349 (1986).

[8] Yang, P. J., Tien, S. B. and Raghavendra, C. S., "Embedding of Rings and Meshes onto Faulty Hypercube Using Free Dimensions," *IEEE Trans. on Computers*, Vol. 43, pp. 608–618 (1994).

[9] Hastad, J., Leighton, T. and Newman, M., "Reconfiguring a Hypercube in the Presence of Faults," *Proc. of 19th ACM Conf. on Theory of Computing*, pp. 274–284

(1987).

[10] Akers, S. B. and Krishnamurthy, B., "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Trans. on Computers*, Vol. 38, pp. 555–565 (1989).

[11] Hayes, J. P. and Mudge, T. N., "Hypercube Supercomputing," *Proc. IEEE*, Vol. 77, pp. 1829–1842 (1989).

[12] Saad, Y. and Schultz, M., "Topological Properties of Hypercube," *IEEE Trans. on Computers*, Vol. 37, pp. 867–871 (1988).

Manuscript Received: Jun. 10, 2005
Accepted: Sep. 30, 2005