Faulty-Tolerant Algorithm for Mapping a Complete Binary Tree in an IEH

Shih-Jung Wu[1], *Jen-Chih Lin[2], and Huan-Chao Keh[3]
Department of Computer Science and Information Engineering,
Nanya Institute of Technology,
No. 414, Sec. 3, Jhongshan E. Rd., Jhongli City,
Taoyuan County 32091, Taiwan, R.O.C.
[2]Department of Digital Content Design
National Taipei University of Education,
No.134, Sec. 2, Heping E. Rd., Da-an District,
Taipei City 106, Taiwan, R.O.C.
E-mail: *yachih@tea.ntue.edu.tw
[3]Department of Computer Science and Information Engineering,
Tamkang University
No. 151 Ying-chuan Road, Tamsui,
Taipei 251, Taiwan, R.O.C.

Abstract: - Different parallel architectures may require different algorithms to make the existent algorithms on one architecture be easily transformed to or implemented on another architecture. This paper proposes a novel algorithm for embedding complete binary trees in a faulty *Incrementally Extensible Hypercube* (IEH). Furthermore, to obtain the replaceable node of the faulty node, *2*-expansion is permitted such that up to *(n+1)* faults can be tolerated with dilation *3*, congestion *1* and load *1*. The presented embedding methods are optimized mainly for balancing the processor loads, while minimizing dilation and congestion as far as possible. According to the result, we can map the parallel algorithms developed by the structure of complete binary tree in an IEH. These methods of reconfiguring enable extremely high-speed parallel computation.

Key-Words: - Hypercube, Incrementally Extensible Hypercube, Complete binary tree, Fault-Tolerance, Embedding

1 Introduction

From the computational perspective, hypercube multiprocessors have recently offered a cost effective and feasible approach to supercomputing through parallelism at the processor level by directly connection a large number of low-cost processors with local memories which communicate by message-passing instead of shared variables. Therefore, hypercubes are widely used interconnection architectures in parallel machines.

Another reason for the popularity of the hypercube is that several other architectures can be embedded in it. Motivations for embedding architectures include:

(1) Efficient algorithms may exist for some architecture which suits the needs of these algorithms perfectly, and we may wish to implement these algorithms in the hypercube.
(2) Proof of embedding for an architecture is also proof of all of its algorithms to be implemented in the hypercube architecture, with a level of efficiency determined only by the cost associated with the embedding.
(3) The embedded architecture is usually easier to understand the visualize. It is often easier to design algorithms for the simpler architecture. In this sense, the embedded architecture can be considered an abstraction from the hypercube, where the irrelevant connections are masked out.
(4) Embedded architectures can be also considered as parallel data structures for parallel architectures. The embedding method shows the way to implement these data structures in the hypercube parallel computer.

But, there are some imperfections to be hypercube architectures for parallel computation, for example only 2^n nodes that a hypercube can be produced. In order to conquer the difficulties

associated with hypercube, several *hypercube-like*[5-9] Computers have been proposed during past years, which are discussed. Based on this observation, we present a new characterization of a fault-tolerant hypercube-like architecture, which is *Incrementally Extensible Hypercube* (IEH) that allows the performance degradation due to faults to be measured.

Tree is a basic network topology. A complete binary tree is special tree underlying divide-and-conquer algorithms. A complete binary tree arises in the solution of tridiagonal systems by even-odd cyclic reduction and solution of systems of equations that is noted in [4]. Suppose some process can be naturally decomposed into a collection of subprocesses that can be executed concurrently with certain communication between subprocesses by an edge between corresponding nodes. One obtains a complete binary tree by denoting each subprocess by a node and each communication between subprocesses by an edge between corresponding nodes. The problem of allocating those subprocesses, structured by a complete binary tree, to processors in a given interconnection networks will be reduced to the problem of embedding a complete binary tree[1, 4].

Graph mappings have been used successfully to show simulation capabilities of guest architecture by host architecture[1, 4]. In graph mapping techniques, host and guest architectures are viewed as graphs *H* and *G*, respectively, and then the graph *G* is mapped into the graph *H*. In the mapping of a graph *G* into *H*, we map the set of nodes of *G* into the set of nodes 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 a mapping must be optimized. The *load* is defined as the maximum number of nodes of *G* assigned to any node of *H*. We say that a mapping achieves a balanced load when *load = 1*. The *dilation* of an edge of *G* is the maximum distance between any pair of processes of *H* corresponding to a pair of neighbor processes of *G*. The *congestion* is defined as the maximum number of paths over an edge in *H*, where every path represents an edge in *G*. The *expansion* of the mapping is the ratio of the number of nodes in *G* to the number of nodes in *H*.

We develop some method to modify the IEH by the addition of spare nodes and communication links in order to obtain designs whose performance degrades gracefully in the presence of node failures.

The rest of this paper is organized as follows. In Section 2, definitions of these topologies are given. Notations and definitions of terms are also provided. Section 3 presents the method for mapping a complete binary tree. In Section 4, we describe the novel algorithm for mapping a complete binary tree in an IEH. Conclusions are finally made in section 5.

2 Preliminaries

We briefly describe notations and definitions of the hypercube and the IEH graph.

A hypercube Q_n of order n, is defined to be a symmetric graph $G= (V, E)$ where V is the set of 2^n vertices, each representing a distinct n-bit binary number and E is the set of symmetric edges such that two nodes are connected by an edge iff the number of positions where the bits differ in the binary labels of the two nodes is *1*.

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 $N= b_nb_{n-1}b_{n-2}...b_1b_0$, 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, $b_i \neq 0$, construct a hypercube graph Q_i with 2^i nodes.

Step 2 Label the nodes. Note that each node has a $(n+1)$-bit binary label. Each hypercube Q_i is labeled as $11...10b_{i-1}b_{i-2}\ldots b_1b_0$. 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 Q_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 $b_i \neq 0$. Set $j=i$ and $G_j=Q_i$.

Set $i=i+1$.

While $i \leq n$ do

if $b_i \neq 0$ then

if $i-j=1$ then

each node x in G_j with label $11\ldots b_j b_{j-1}\ldots b_0$ is connected to the node $11\ldots 10 b_j b_{j-1}\ldots b_0$ of Q_i.

else

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

$$\begin{cases} 11\ldots11011\ldots11 b_j b_{j-1}.....b_0 \\ 11\ldots11001\ldots11 b_j b_{j-1}.....b_0 \\ 11\ldots11010\ldots11 b_j b_{j-1}.....b_0 \\ \vdots \\ 11\ldots11011\ldots01 b_j b_{j-1}.....b_0 \\ 11\ldots11011\ldots10 b_j b_{j-1}.....b_0 \end{cases}$$

Set $j=i$ and set G_j to be the c osite 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. □

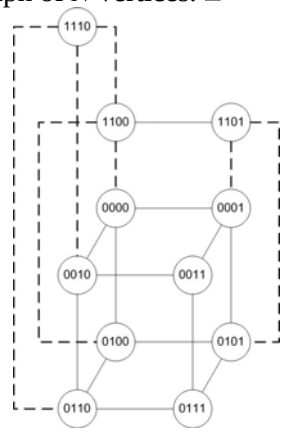

Fig. 1: The IEH graph contains 11 nodes

Figure 1 shows the example of $G_3(11)$. $G_3(11)$ consists of three subcubes. The three subcubes are *0*-subcube(H_0), *1*-subcube(H_1), and *3*-subcube(H_3). Node *14* is the single node in H_0. Nodes *12* and *13* are composed as a *1*-subcube(H_1), and nodes *0*, *1*, *2*, *3*, *4*, *5*, *6* and *7* are the elements of a *3*-subcube(H_3). The edge *(12, 14)* is an IC edge connected between H_0 and H_1 such that H_0 and H_1 are connected to be an IEH graph containing *3* nodes$(G_1(3))$. In addition, the H_3 connects to $G_1(3)$ with these IC edges (*2, 14*), (*6, 14*), *(0,12)*, *(4,12)*, *(1,13)*, and *(5,13)*.

Definition 1[5] 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

$HD(x, y)=\sum_{i=0}^{n-1} hd(x_i, y_i)$, 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[5] Let $x=x_{n-1}...x_0$, $y=y_{n-1}...y_0$, then $Dim(x, y)=\{i \text{ in } (0...n-1) \mid x_i \neq y_i\}$□

Definition 3[4] If a complete binary tree is a rooted binary tree and each internal nodes contains two offspring nodes, then a complete binary of height h denoted by T_h, contains 2^h-1 nodes. □

Definition 4[4] A double-rooted binary tree DT_h, where h is the height of the tree, is a complete binary tree with the root replaced by a path of length two is shown in figure 2. □

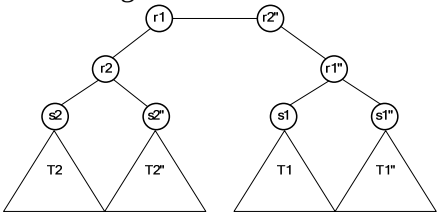

Fig. 2: A double-rooted complete binary tree

Lemma 7[4] A double-rooted complete binary tree can be embedded into a hypercube with dilation *2* and load *1*. □

Definition 5[6] An IEH graph is called a full IEH graph, denoted by F_n, if and only if it has $2^{n+1}-1$ nodes. Intuitively, a full IEH graph must contain hypercubes $H_0, H_1, \ldots, H_n$ as its subcubes. □

3 Mapping of Complete Binary Tree

Consider a complete binary tree T_h, where h is the number of levels. Assume that the levels of the tree are indexed as 0,…,($h-1$) starting from the root and each internal nodes contains two offspring nodes, then a complete binary tree T_h contains (2^h-1) nodes.

IEH graphs are provided with all properties of hypercubes because an IEH graph may contain some different-sized hypercubes as its subgraphs. Thus the IEH graphs are selected to be the host

graph of our embedding. Complete binary tree are usually used in a lot of algorithms and communications, so we proposed the method of embedding complete binary trees into IEH graphs.

We describe our approach that maps a complete binary tree in the full IEH or the IEH.

Lemma 1 A complete binary tree with height n contains the same number of nodes as an (n-1)-dimensional full IEH graph.

Proof. Suppose a tree T_n is said to be a complete binary tree with height n, then each node of T_n contains the same number of nodes in its right and left subtrees. Consequently, the number of nodes in a complete binary tree T_n has totally 2^n-1 nodes. A (n-1)-dimensional full IEH graph must, by the definition, contain n hypercubes with different dimensions. Therefore, a (n-1)-dimension IEH graph, simply denoted by F_{n-1}, will consist of hypercube with dimension with $0, 1, 2, \ldots$, and (n-1). The k_{th}-dimensional hypercube in a IEH is denoted by H_k. Obviously, an (n-1)-dimensional full IEH contains $2^0+2^1+2^2+\ldots+2^{n-1} = 2^n-1$ nodes because each k_{th}-dimensional hypercube in the IEH graph contains 2^k nodes. Hence, a complete binary tree with height n contains the same number of nodes as an (n-1)-dimensional full IEH graph.□

Lemma 2 An (n-1)-dimensional full IEH graph is a subgraph of an n-dimensional full IEH graph.

Proof. Let F_n be a full IEH graph, then two divided components H_n and F'_n exist in F_n where H_n is an n-dimensional hypercube and F'_n is an (n-1)-dimensional full IEH graph. According to the method of constructing an IEH graph, the vertex set of F_{n-1} is a subset of the vertex set of F_n and the edge set of F_{n-1} is a subset of the edge set F_n. Therefore, F_{n-1} is said to be a subgraph of F_n.□

Lemma 3 Assume that H_n is an n-dimensional hypercube and F_{n-1} is and (n-1)-dimensional full IEH graph, F_{n-1} is a subgraph of H_n.

Proof. According to the method of constructing an IEH graph, $H_0, H_1, \ldots, H_{n-1}$ construct F_{n-1}. These IC edges connecting between the n hypercubes are all of Hamming distance 1. Hence, these IC edges in F_{n-1} are normal edges in H_n. Consequently, the vertex set of F_{n-1} is a subset of the vertex set of H_n and the edge set of F_{n-1} is a subset of the edge set H_n. Therefore, F_{n-1} is said to be a subgraph of F_n. □

Lemma 4[6] A full IEH graph contains a complete binary tree.

Lemma 5[6] A complete binary tree can be embedded into a hypercube with one faulty node.

Lemma 6 Embedding the complete binary tree into a full IEH graph or a hypercube graph is dilation 2, expansion 1, load 1 and congestion 1.

Proof. In the embedding of T_n into F_{n-1}, there are some midway node must be passed to connect two nodes That is, some edges in the tree T_n are mapped in a path of the length 2 in the IEH graph F_n. As the consequence, the dilation of the embedding is 2 and congestion is 1. The embedding of T_n into F_{n-1} is one-to-one mapping for these nodes, so the expansion and load of our embedding is exactly equal to 1. Therefore, embedding the complete binary tree into a full IEH graph or a hypercube graph is dilation 2, expansion 1, load 1 and congestion 2.□

Theorem 1 A complete binary tree T_n can be embedded into $G_n(N)$ with expansion 2, dilation 2, congestion 1, and load 1.

Proof. It's trivial by lemma 4 and lemma 5. And an example is shown by figure 3. □

Figure 3 shows three steps of T_3 embedding to $G_3(11)$.

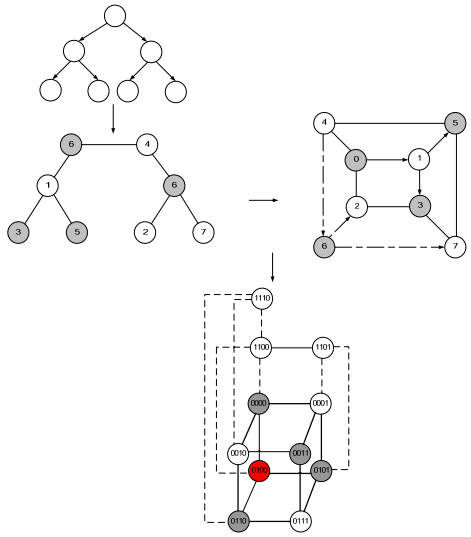

Fig. 3: Three steps of T_3 embedding to $G_3(11)$

4 Faulty-Tolerance Mapping

We present how to embed a complete binary tree to a faulty IEH graph. Hence, in this section, we consider a complete binary tree can be mapped in an IEH with 2-expansion graph which contains faulty node.

We show that a complete binary tree can be

mapped in an IEH graph with expansion *2*, load *1*, congestion *1*, and dilation *4*. By theorem *1*, T_n can be embedded into $G_n(N)$ with *2*-expansion. We propose an algorithm FT_Tree_Embedding( ) for mapping a complete tree in a faulty IEH as follows.

Algorithm FT_Tree_Embedding(*x*)

```
Input:  x        /*the faulty node*/, Gn(N)
Output: y        /*the replaceable node*/
1.    if the root r is faulty then
2.      search the other root node r'
3.      if the node r' is faulty then backtrack the root
        r
4.      else
5.          return(r') /* node r is replaced by node r'/
6.          exit()
7.          i=0
8.      if other node x is faulty
9.      then
10.       {
11.         search the node y
             /* HD(x, y )=1, Dim(x, y)={n}*/
12.         if y is a exist node and it is free
13.         then
14.             return(y) /*replace x with y*/
15.             exit()
16.         else
17.             while i < n do
18.                 search the node z
                     /* HD(y, z )=1, Dim(y, z)={i}*/
19.                 if z is a exist node and it is free
20.                 then
21.                     return(y) /*replace x with z*/
22.                     exit()
23.                                 i=i+1
24.   return("Failure")
25.   end
```

The searching path of the faulty node is shown as follows.

node0=$0X_{n-1}X_{n-2}\ldots X_1X_0$
node1=$1X_{n-1}X_{n-2}\ldots X_1X_0$
node2=$1X_{n-1}X_{n-2}\ldots X_1X'_0$
node3=$1X_{n-1}X_{n-2}\ldots X_1'X_0$
⋮
node(n+1)=$1X'_{n-1}X_{n-2}\ldots X_1X_0$

We illustrate an example of finding a replaceable node in $G_3(11)$ as shown figure *24* and figure *33*.

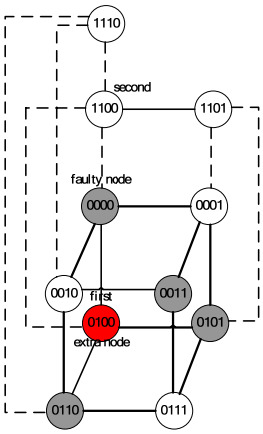

Fig. 4: T_3 can be embedded into faulty $G_3(11)$

We illustrate an example of finding a replaceable node in $G_3(11)$ as shown figure *5*.

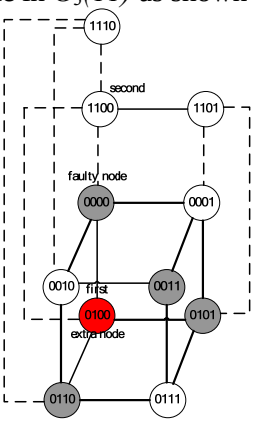

Fig.5: T_3 can be embedded into faulty $G_3(11)$

Theorem 2 A complete binary tree (T_h) can be embedded into faulty IEH $G_n(N)$ graph with dilation *4*, expansion *2*, congestion *1*, and load *1*.

Proof. By the algorithm FT_Tree_Embedding( ), every searching path is only on path. Allowing us to obtain congestion *1* and load *1*. Herein, we allow *2*-expansion to obtain the replaceable node of the faulty node. When a node is faulty, it is worst case in which the dilation=*2+2=4* at most by FT_Tree_Embedding( ). Because these nodes and links of searching path are not replicated from FT_Tree_Embedding( ), four costs associated with graph embedding are dilation *4*, expansion *2*, congestion *1*, and load *1*. □

Theorem 3 A searching path of FT_Tree_Embedding( ) is include approximate to (n+1) nodes.

Proof. Every node can be represented by a *(n+1)*-bit binary string $i_n \cdots i_0$ where $i_p \in \{0,1\}$. First, we change the most significant bit from *0* to *1*. Then, a bit can be changed from i_0 to i_{n-1} sequentially by

the FT_Tree_Embedding( ). But IEH graph may be have empty nodes. Hence, a searching path of FT_Tree_Embedding( ) is including approximate to *(n+1)* nodes. □

Theorem 4 There are $O(n)$ faults, which can be tolerated.

Proof. It's trivial by theorem *3*.□

5 Conclusions

In this paper, we consider the algorithm FT_Tree_Embedding for a complete binary tree can be mapped in an IEH. Considering embedding of complete binary tree into a faulty IEH, allowing *2*-expansion shows that up to *(n+1)* faults can be tolerated. The main result of this paper is that it is always possible to give solutions to the mapping of complete binary trees in a faulty IEH graph with *2*-expansion. The costs associated with graph embedding in our strategies of reconfiguration are dilation *3*, congestion *1* and load *1*. By the result, we can embed the parallel algorithms developed by the structure of complete binary tree in an IEH. These methods of reconfiguring enable extremely high-speed parallel computation. Therefore, we can easily port the parallel or distributed algorithms developed for these structures to the IEH graphs and hypercubes.

Reference

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

[2] M. Amiripour, H. Abachi, and K Dabke, Hardware Cost Analysis of Master-Slave Star Ring Super-Hypercube and Master-Slave Super-Super-Hypercube 4-Cube Architecture," *Proceedings of the 6th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '07)*, 2007, pp.115-120.

[3] C.-C. Chen, Dynamic Reconfiguration of Complete Binary Trees in Faulty Hypercubes, *Journal of Information Science and Engineering*, Vol. 21, No. 1, 2005, pp. 195-207.

[4] F. T. Leighton, *Introduction to parallel algorithms and architectures: Arrays, Trees, Hypercubes*, MORGAN KAUFMANN PUBLISHERS, Inc., 1992.

[5] J.-C. Lin, Simulation of Cycles in the IEH Graph, *International Journal of Hjgh Speed Computing*, Vo1. 10, 1999, pp. 327-342.

[6] J.-C. Lin, Load Balancing and Embedding Rings in Faulty Incrementally Extensible Hypercubes, *WSEAS Transactions on Computers*, Vol. 5, 2006, pp. 1867-1872.

[7] J.-C. Lin, Faulty-Avoiding Methods for Mapping Meshes in an IEH, *WSEAS Transactions on Computers*, Vol. 6, No.6, 2007, pp. 888-893.

[8] J.-C. Lin**,** S. K.C. Lo, S.-J. Wu, and H.-C. Keh, Distributed Fault-Tolerant embeddings of rings in Incrementally Extensible Hypercubes with Unbounded Expansion, *Tamkang Journal of Science and Engineering*, Vol. 9, No. 2, 2006, pp. 121-128.

[9] C.D. Park, and K.-Y. Chwa, Hamiltonian properties on the class of hypercube-like networks, *Information Processing Letters*, No. 91, 2004, pp. 11-17.

[10] J. T. Richard, and R. R. Yager, Hypercube Graph Representations and Fuzzy Measures of Graph Properties, *IEEE Trans. on Fuzzy Systems*, Vol. 15, No. 6, 2007, pp. 1278-1293.

[11] Y. Saad, and M. Schultz, Topological properties of Hypercube, *IEEE Trans. on Computers*, Vol. 37, 1988, pp. 867-871.

[12] S. Sur, and P. K. Srimani, Incrementally Extensible Hypercube Networks and Their Fault Tolerance, *Mathematical and Computer Modeling*, Vol. 23, 1996, pp. 1-15.

[13] S. Sur, and P. K. Srimani, IEH graphs: A novel generalization of hypercube graphs, *Acta Informatica*, Vol. 32, 1995, pp. 597-609.

[14] I. Zelina, P. Pop, C. P. Sitar, and I. Tascu, A parallel algorithm for interpolation in Pancake graph, *Proceedings of the 6th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '07)*, 2007, pp.98-101.