Simulation of Meshes and Tori in an asymmetric Faulty Incrementally Extensible Hypercube with Unbounded Expansion

*JEN-CHIH LIN[1]
[1]Department of Digital Technology Design,
National Taipei University of Education,
No.134, Sec. 2, Heping E. Rd., Da-an District, Taipei City 106,
TAIWAN
E-mail:*yachih@tea.ntue.edu.tw http://tea.ntue.edu.tw/~yachih

Abstract: - The paper considers the problem of finding meshes and tori in a Faulty Incrementally Extensible Hypercube if any. We develop novel algorithms to facilitate the embedding job when the Incrementally Extensible Hypercube (IEH) contains faulty nodes. We present strategies for reconfiguring a mesh or tori in an IEH with unbounded expansion. These simulation approach shows a mesh or torus can be embedded into a faulty IEH $G_n(N)$ with load *1*, congestion *1* and dilation *3* such that $O(n^2-\lfloor log_2 m \rfloor^2)$ faults can be tolerated. Furthermore, the technology can be apply in grid computing and cloud computing.

Key-Words: - Incrementally Extensible Hypercube (IEH), hypercube, mesh, torus, embedding, expansion

1 Introduction

Rapidly advancing technology has made it possible for a large number of processing elements to be interconnected in a variety of configurations. In the investigation of parallel computing, networks of processors are often organized into various configurations such as hypercubes, trees, rings, and meshes. These configurations can be presented as graphs. If the properties and structures of the underlying graph are used effectively, the computations and communication speeds can often be improved. Many parallel algorithms have been designed to solve different problems on various networks. It would be of interest to be able to execute these algorithms in other networks. Therefore, the problem of simulating one network by another is modeled as a graph embedding problem where nodes and edges in the graph represent the processors and communication links between processors. Organizing computations in a network of processors is also modeled as a graph embedding. So, graph embedding problems become the important applications in a wide variety of computational situations.

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

An embedding of a graph $G=(V_G,E_G)$ into a graph $H=(V_H,E_H)$ includes a mapping of nodes of V_G into nodes of V_H and a mapping each edge of E_G into a path of *H*. The graph *G* and *H* are referred as the guest and host graphs. Four common measures of quality of an embedding in parallel processing are *dilation*, *expansion*, *congestion* and *load*. The dilation measures the communication delay and is defined as the maximum length of paths mapping by edges of E_G. The processor utilization is measured by the expansion that is defined as the ratio of total number of nodes of *G* to total number of nodes of *H*. The congestion of an embedding is the maximum number of edges of the guest graph *G* that is mapped by a single edge of the host graph *H*. The congestion is a measurement of queuing delay of massages. To measure the processing time of tasks is referred as the load in an embedding. The load is the maximum number node of *G* that is embedded in a single node of *H*.

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. Hypercube has been the focus of researches in parallel computing because of its well-defined properties with the modularity, regularity, and low diameter, etc. These characteristics make it easy to design efficient parallel programs and share machines among users. Its important advantages are high data bandwidth and low message latency. Moreover, the hypercube may contain many other networks as its subgraphs such as rings, trees, meshes, etc. On the other hand, lots of interconnection networks can be mapped into the hypercube. It is apparent to demonstrate how all of the parallel algorithms, designed by those interconnection networks, can be directly implemented on the hypercube without significantly affecting the number of processors or the computation time.

A hypercube, also known as a binary *n*-cube or cosmic cube, contains 2^n processors (nodes), each of which is connected by fixed communication paths (links) to *n* other nodes. The value *n* is known as the dimension of the hypercube. In a hypercube, two nodes are connected if and only if their addresses differ by one and only one bit. Extensive research efforts have been focused on hypercube design aspects and hypercube applications such as data permutation and matrix operations. These have resulted in several commercial products, such as the Intel iPSC and the Connection Machine. However, the number of nodes of this network is restricted to be a power of *2*, that can be, in some situations, a significant drawback. In fact, to upgrade a hypercube it is necessary to double the number of processors, which can be unrealizable for budget limitations and for technical reasons. Several hypercube-like networks that can be constructed for any number of nodes have been proposed, such as *Incomplete Hypercube*[14], *Supercube*[28, 29], *Flexible Hypercube*[11], *Incrementally Extensible Hypercube*[31, 32], and so on.

The Incrementally Extensible Hypercube (IEH) graph is a new topology of interconnection networks and proposed recently. Unlike the hypercube, the IEH graph is incrementally extensible, that is, any number of nodes can construct it. Besides, it has the optimal ability of the fault tolerance and the diameter of logarithmic in the number of nodes. The difference between the maximum and the minimum degree of nodes in an IEH graph is at most 1, so it is enough to say that the IEH graph is almost regular. However, the IEH graph does not have the drawbacks of the above-mentioned generalizations of the hypercube. The characteristics of IEH graphs are shown in [31].

Among the static interconnection networks used for SIMD[26] computers with an array of processors[2], one of the oldest and very popular architectures is a two-dimensional-mesh. Many important algorithms for solving various problems, e.g., matrix operations, simultaneous linear equations, graph-theoretic and image processing problems, etc., have been efficiently mapped in this mesh architecture.

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.

Also, we developed the methods for finding meshes or tori in an IEH graph. As the result, we can transit the parallel algorithms developed under the structure of meshes or tori to the IEH graph. This simulation approach enables extremely high-speed parallel computation in IEH graphs. Although IEH graphs are not absolutely asymmetric, it has the same power as the hypercube in terms of meshes and tori.

The remainder of this paper is organized as follows. Section 2 is devoted to some notations and definitions. The construction of the mesh 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

This section briefly describes notations and definitions of the IEH graph. The IEH graph of *n*-dimension is the composition of some *m* different hypercubes of dimension *k*, where $0 \le k \le n$ and $1 \le m \le n$. Let $G_n(N)$ be a *n*-dimensional IEH graph with *N* nodes, and *N* can be expressed by the binary string $N = b_n b_{n-1} b_{n-2} \ldots b_1 b_0$, and $b_i \in \{0,1\}$. Suppose that hypercube H_i is a part of the IEH graph $G_n(N)$, it is certain that the i_{th} bit of *N*, b_i must be *1*. That is, an IEH graph $G_n(N)$ is composed of some different hypercubes which have lower dimension than $G_n(N)$ has. For example, $G_3(13)$ is an IEH graph contains *13* nodes, and it is composed by three different-sized hypercubes H_0, H_2,and H_3 because $13=1101$, and $b_0=b_2=b_3=1$.

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. 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 $N = b_n b_{n-1} b_{n-2} \ldots b_1 b_0$, 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 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 $11\ldots10b_{i-1}b_{i-2}\ldots b_1 b_0$. Obviously each hypercube of dimension *i* (having 2^i nodes) have *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 $b_i \neq 0$. Set $j=i$ and $G_j = H_i$.

Set $i=i+1$.

While $i \le n$ 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 $11\ldots10b_j b_{j-1}\ldots b_0$ of H_i.

else

each node *x* in G_j with label $11\ldots1b_j b_{j-1}\ldots b_0$ is connected to (*i-j*) different nodes of H_i chosen in the following way:

$$\begin{cases} \overbrace{11\ldots\ldots11}^{\text{n-i}}\overbrace{01\ldots\ldots11}^{i-j-1}b_j b_{j-1}\ldots\ldots b_0 \\ \overbrace{11\ldots\ldots11}^{\text{n-i}}\overbrace{001\ldots\ldots11}^{i-j-1}b_j b_{j-1}\ldots\ldots b_0 \\ \overbrace{11\ldots\ldots11}^{\text{n-i}}\overbrace{010\ldots\ldots11}^{i-j-1}b_j b_{j-1}\ldots\ldots b_0 \\ \vdots\ddots\vdots \\ \overbrace{11\ldots\ldots11}^{\text{n-i}}\overbrace{011\ldots\ldots01}^{i-j-1}b_j b_{j-1}\ldots\ldots b_0 \\ \overbrace{11\ldots\ldots11}^{\text{n-i}}\overbrace{011\ldots\ldots10}^{i-j-1}b_j b_{j-1}\ldots\ldots b_0 \end{cases}$$

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 shows the example of $G_3(13)$. $G_3(13)$ consists of three subcubes. The three subcubes are *0*-subcube(H_0), *2*-subcube(H_2), and *3*-subcube(H_3). Nodes *14* is the single node in H_0, 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*, *14*), (*10*, *14*) are IC edges connected between H_0 and H_2 such that H_0 and H_2 are connected to be an IEH graph containing *5* nodes($G_2(5)$). In addition, the H_3 connects to $G_2(5)$ with these IC edges *(0, 8), (1, 9), (2,10), (3,11),* and *(6,14)*.

Definition 1[19] 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[19] Let $x=x_{n-1}\ldots x_0$, $y=y_{n-1}\ldots y_0$, then $Dim(x, y)=\{i \text{ in } (0\ldots n-1) \mid x_i \neq y_i\}$

Definition 3[1] 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 4[16] $m_1 \times m_2$ mesh or torus, denoted by $M_{m_1 \times m_2}$, is a *2*-dimensional mesh or torus, where $m_1 = 2^r, m_2 = 2^s$.

Definition 5[16] Any $m_1 \times m_2 \times \cdots \times m_d$ mesh or torus, denoted by $M_{m_1 \times m_2 \times \cdots \times m_d}$, in the *d*-dimensional space R_d, where $m_i = 2^{p_i}$.

Definition 6[19] 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 6, 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 $C_3=0C_2 \cup 1(C_2)^R=\{000, 001, 011, 010, 110, 111, 101, 100\}$.

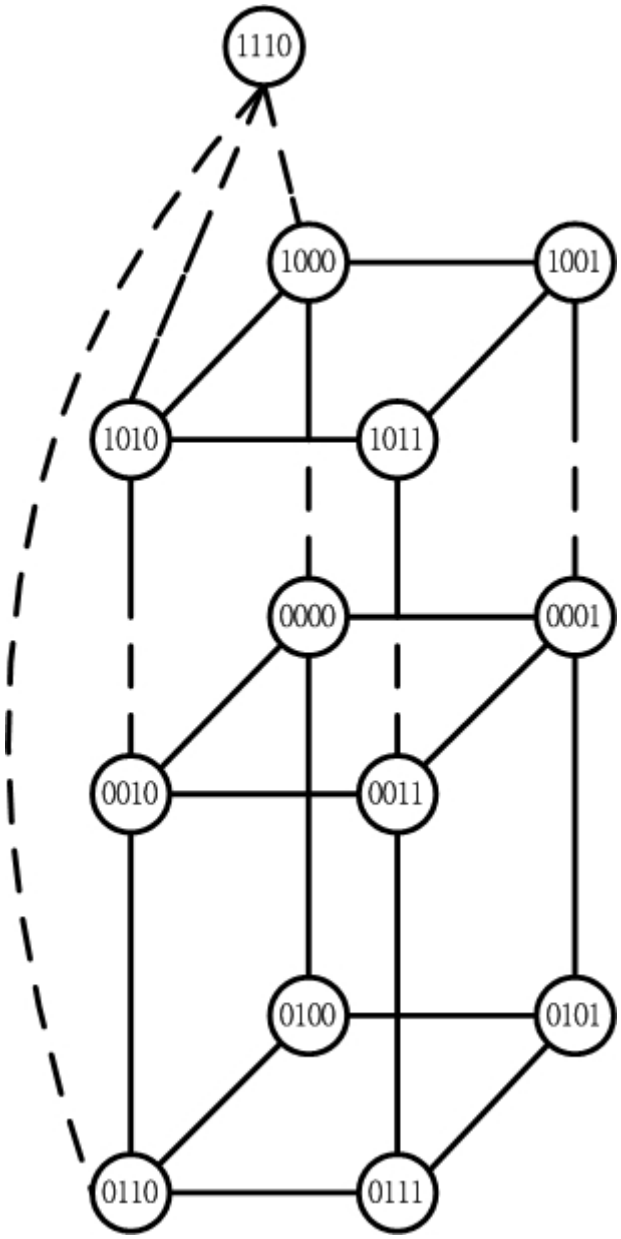

Fig. 1 The IEH graph contains 13 nodes

3 Meshes and Tori Embedding

The section describes the representation used to solve that embeds a mesh and torus in an IEH.

Lemma 1 $m_1 \times m_2$ mesh or torus, denoted by $M_{m_1 \times m_2}$, is a *2*-dimensional mesh or torus, where $m_1 = 2^r, m_2 = 2^s$ can be embedded in an *n*-dimensional hypercube where $n = r + s$.

Lemma 2 Any $m_1 \times m_2 \times \cdots \times m_d$ mesh or torus, denoted by $M_{m_1 \times m_2 \times \cdots \times m_d}$, in the *d*-dimensional space R_d, where $m_i = 2^{p_i}$ can be embedded in an *n*-dimensional hypercube where $n = p_1 + p_2 + \ldots + p_d$. The numbering of the mesh or torus nodes is any numbering such that its restriction to each i_{th} variable is a Gray sequence which is described in definition 2. Note that the assumption that all m_i*'s* be power of *2*.

Our proposition is best illustrated by an example. Consider a $M_{2\times 2}$ mesh or torus i.e., $d = 2$, $p_1 = 1$, $p_2 = 1$, $n = p_1 + p_2 = 2$. A binary number *H* of any node of the *2*-dimensional hypercube can be regarded as consisting of two parts: its first *1* bit and its last *1* bit, which we write in the form $H = X_1Y_1$, where X_i and Y_i are bits *0* or *1*. It is clear from the definition of an *n*-dimensional hypercube (with $n = 2$) that when the last *1* bit is fixed, then the resulting 2^{p_1} nodes form a p_1-dimensional hypercube (with $p_1 = 1$). Whenever we fix the first *1* bit we obtain a p_2-dimensional hypercube. The embedding then becomes clear. Choosing a *1*-bit *BRGC* for the *x*

direction and *1*-bit *BRGC* for the *y* direction, the point (x_i, y_i) of the mesh or torus is assigned to the node X_1Y_1 where X_1 is the *1*-bit *BRGC* for dimension of p_1 while Y_1 is the *1*-bit *BRGC* for dimension of p_2. Herein, we illustrate the result of the mesh or torus in Figure 2.

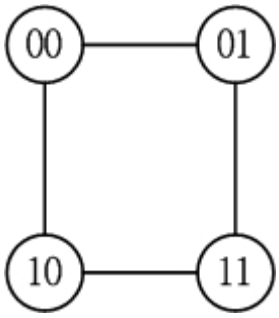

Fig. 2 A $M_{2\times2}$ mesh

The binary node number of any mesh and torus node is obtained by concatenation its binary *x* coordinate and its binary *y* coordinate. Therefore, if we call Gray sequence any subsequence of a *BRGC*, we observe that any column of mesh and torus nodes forms a Gray sequence and any row of mesh and torus nodes forms a Gray sequence. Thus, we will refer to the codes defined above as *2-D* Gray codes. Generalizations to higher dimensions are straightforward and one can state the above lemma 2.

Lemma 3 For any given *N*, a hypercube H_n must be a subgraph of an IEH $G_n(N)$, where $2^n \le N < 2^{n+1}$.

Proof. An IEH $G_n(N)$ must contain a hypercube H_n. That is trivially by the generation schema of an IEH $G_n(N)$ graph. It must contain the maximum hypercube H_n.

The simulation approach that a $M_{m_1\times m_2\times\cdots\times m_d}$ mesh or torus can be embedded in an IEH $G_n(N)$ *i*s as follows.

Simulation approach

$M_{m_1\times m_2\times\cdots\times m_d}$ ($m_i = 2^{p_i}$),

$G_n(N)$ ($2^n \le N < 2^{n+1}$),

$\forall p_1 + p_2 + \ldots + p_d = w, w \le n, p_1, p_2, \ldots, p_d \ge 1$

$G_n(N) = G(V,E)$ $M_{m_1\times m_2\times\cdots\times m_d} = G(V^{'}, E^{'})$,

$v \in V \quad v' \in V'$ (Denoted by unique binary string)

$v = X_n \ldots X_{w-1} X_{w-2} \ldots X_1 X_0$

$v' = X_{w-1} X_{w-2} \ldots X_1 X_0$

$v' \in V'$ can be embedded in V denote as $v = 0 \ldots 0 X_{w-1} X_{w-2} \ldots X_1 X_0$

Theorem 1 A $M_{2^r\times2^s}$ *2*-dimensional mesh or torus can be embedded in an IEH $G_n(N)$ where $r + s = \lfloor \log_2 N \rfloor$ with load *1*, dilation *1*, congestion *1*, and expansion *2*.

Proof: This is trivial by lemma 1and the above simulation approach.

Theorem 2 Any $M_{m_1\times m_2\times\cdots\times m_d}$ *d*-dimensional mesh or torus, where $m_i = 2^{p_i}$ can be embedded in an IEH $G_n(N)$, where $p_1 + p_2 + \ldots + p_d = \lfloor \log_2 N \rfloor$ with load *1*, dilation *1*, congestion *1* and expansion *2*.

Proof: It is trivial by lemma 2 and the above simulation approach.

This is the best illustrated by an example in Figure 3. That is a $M_{2\times2}$ mesh or torus can be embedded in an IEH $G_3(13)$.

Lemma 3 A mesh or tori contains any number of nodes can be embedded into an IEH graph with load *1*, congestion *1*, and dilation *1*.

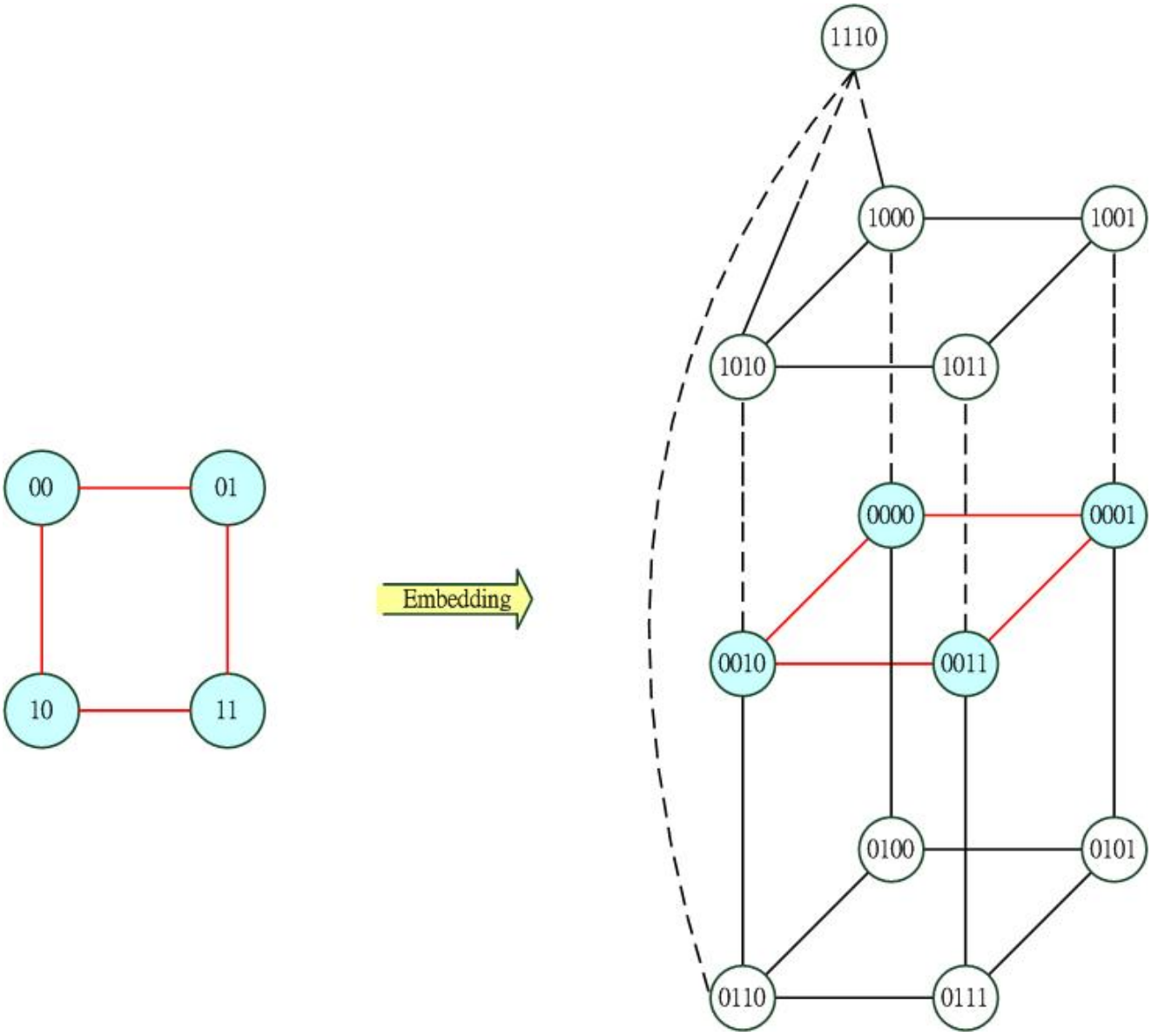

Fig. 3 Embedding of a $M_{2\times 2}$ mesh and torus in an IEH $G_3(13)$

4 Fault-Tolerant mapping with Unbounded Expansion

In the previous section, we have constructed a mesh and a torus in an IEH graph. In the section, we consider a faulty IEH with unbounded expansion embedding.

Theorem 3 A mesh or a tori can be mapped into an IEH graph with unbounded expansion.

Proof: It is trivial by lemma 3.

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 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 $0 \le m < n$.

Proof: Let $G_n(N)$ be an IEH graph contains $N=(a_{n-1}a_{n-2}...a_0)$ nodes. It is composed by hypercubes H_i if $a_i \neq 0$ 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 $H_0, H_1, ...,$ and H_{n-1} if it is greatest, so the maximal number of nodes in $G_m(N-|H_n|)$ is $|H_0|+|H_1|+|H_2|+...+|H_{n-1}|=2^0+2^1+...+2^{n-1}=2^n-1$. As the result, $2^n=|H_n| \ge (N-|H_n|)=2^n-1$.

Theorem 5 For an IEH $G_n(N)$, the nodes of the subgraph H_n has an IC edge at least.

Proof: By the construction of IEH and theorem 4, all

of nodes of $G_m(N)$, where $m < n$, has a unique IC edges connecting to H_n at least. Therefore, the nodes of the subgraph H_n of an IEH $G_n(N)$ have an IC edge at least.

Algorithm Mesh_Mapping(*x*)

Input: x /*the faulty node*/,
$M_{m_1 \times m_2 \times \cdots \times m_d}$ ($m_i = 2^{p_i}$),
$G_n(N)$ ($2^n \le N < 2^{n+1}$),
$\forall p_1 + p_2 + \ldots + p_d = w, w \le n,$
$p_1, p_2, \ldots, p_d \ge 1$

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 z
/* $HD(a, z)=1$, $Dim(a, z)=j$*/
25. if z is not a virtual node and it is free
26. then
27. return(z)
/*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 an 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 z
/* $HD(z, y)=1$, $Dim(z, y)=k$*/
41. if z is not a virtual node and it is free
42. then
43. return(z)
/*replace x with y*/
44. *exit()*
45. $k=k+1$
46. }
47. return("Failure")
48. end

Finding the replaceable node as follows:

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

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

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

$\vdots$

node $(n-\lfloor \log_2 m \rfloor) = 0X'_{n-1}X_{n-2}\ldots X_{\lfloor \log_2 m \rfloor}\ldots X_1\, X_0$

node $(n-\lfloor \log_2 m \rfloor +1) = 1X_{n-1}X_{n-2}\ldots X_{\lfloor \log_2 m \rfloor}\ldots X_1\ X_0$

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

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

$\vdots$

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

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

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

$\vdots$

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

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

$\vdots$

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

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

(The IC edge connects node *0* and node *$((n-\lfloor \log_2 m \rfloor+1)*(\lfloor \log_2 m \rfloor+1)+(1+2+\ldots+n)+1))$*

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

$\vdots$

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

For the IEH $G_3(13)$ as Figure 3, where the $M_{2\times2}$ has been mapped in it. We give a simple example in this section to explain the operations of the Mesh_Mapping algorithm when the faulty nodes exist. The procedure for handling this faulty node straightforward.

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, search the node *10*, to check whether it is free or not. If it is, it terminates.
7. If not, search the node *14*, to check whether it is free or not. If it is, it terminates.
8. If not, return("Failure").

Therefore, the whole searching path is listed as *{4(0100), 8(1000), 5(0101), 6(0110), 9(1001), 10(1010), 14(1110)}*.

In Figure 4, we assume that the node 0000 is faulty. We can find the six nodes in the sequence for replacing the faulty node.

We illustrate the search tree of finding a replaceable node in an IEH graph $G_3(13)$ as shown Figure 5.

Figure 6 shows the representation for Figure 5. In Figure 6, we have mapped these replaceable nodes into these idle nodes of the IEH graph $G_3(13)$.

Now, Figure 7 shows **Mesh_Mapping()** algorithm applied to the graph of Figure 4.

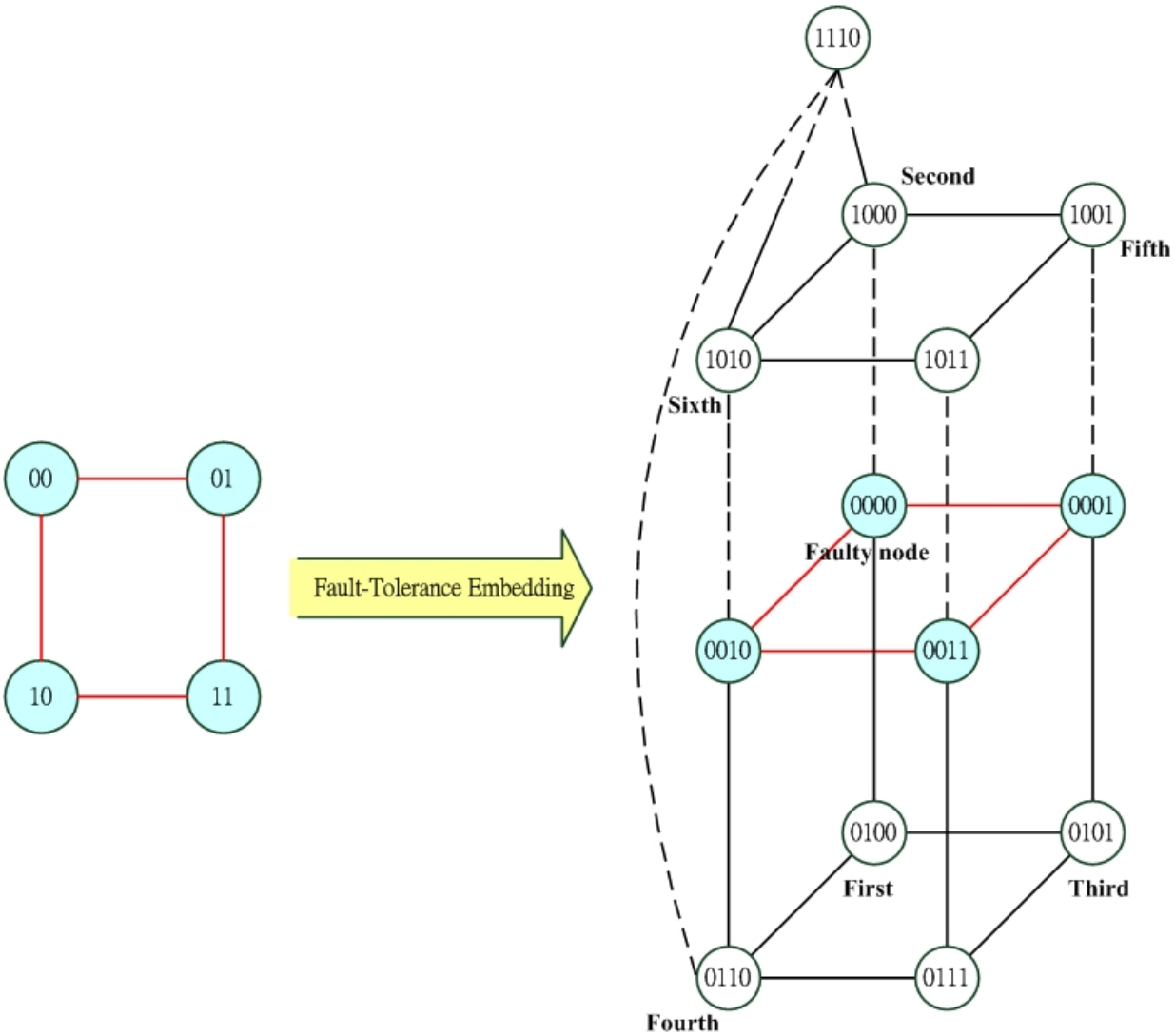

Fig. 4: Embedding of a $M_{2\times2}$ mesh and torus in a faulty IEH $G_3(13)$

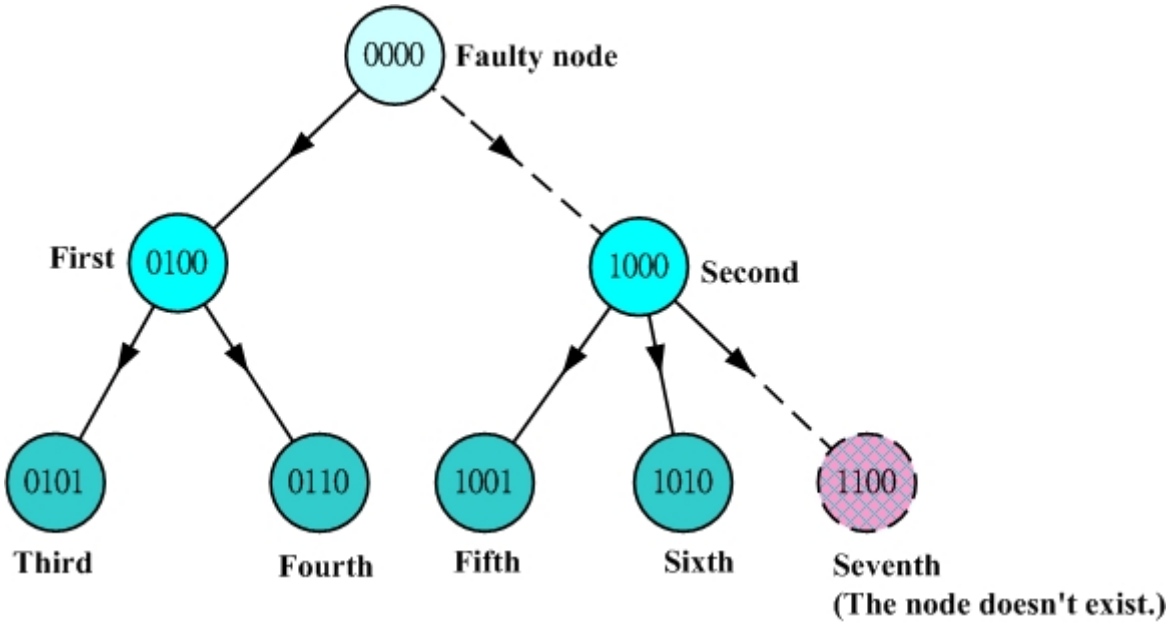

Fig. 5 The search tree for finding the searching path by Mesh_Mapping(*x*) algorithm

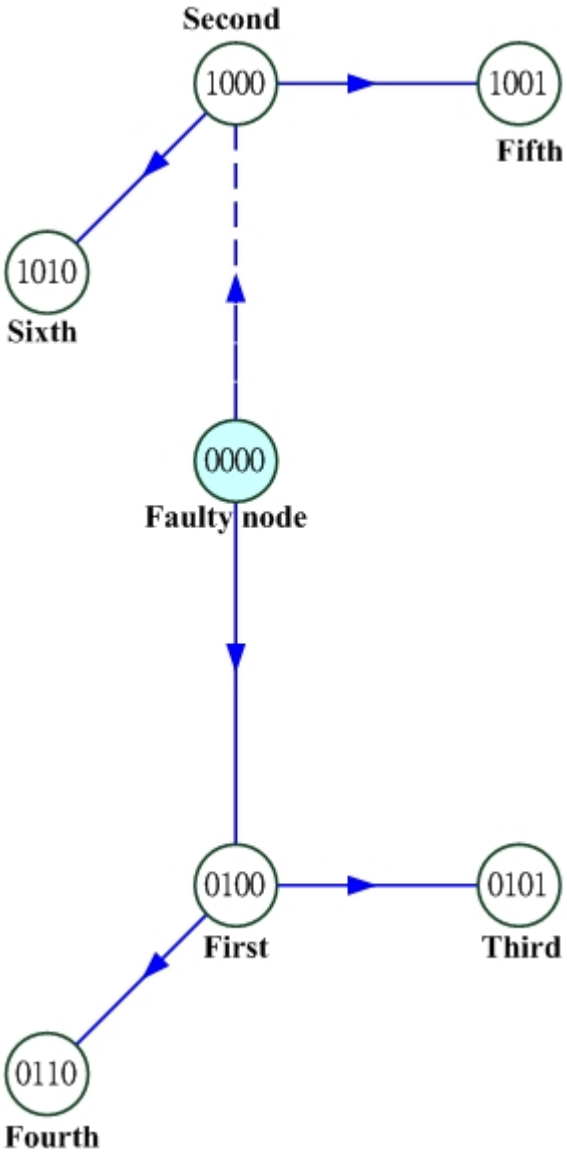

Fig. 6 Mapping Figure 5 into IEH $G_3(13)$

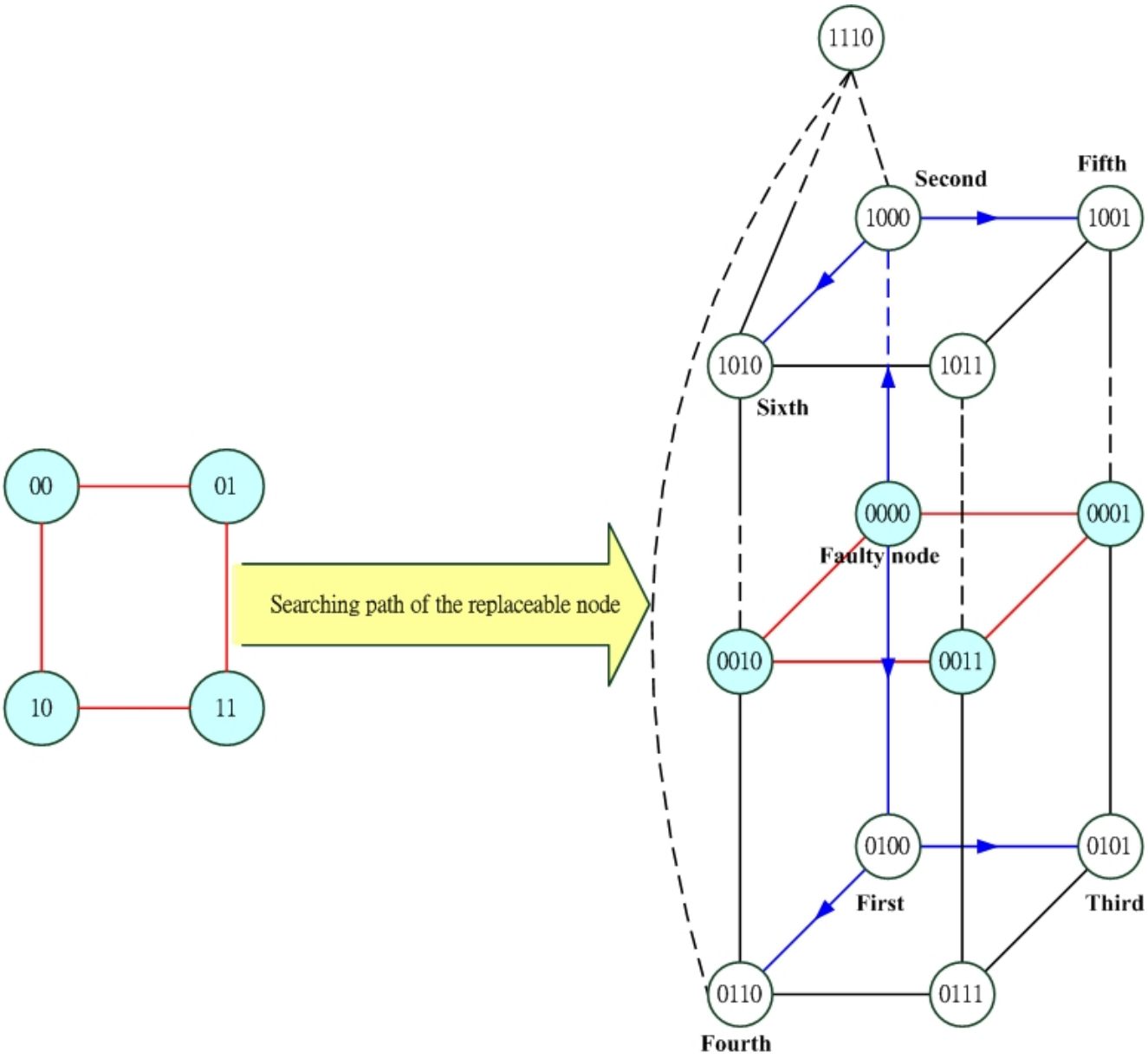

Fig. 7 Summary of Mesh_Mapping(*x*) algorithm applied to Figure 4

Theorem 6 A mesh or a torus $M_{m_1 \times m_2 \times \cdots \times m_d}$ can be mapped into a faulty IEH $G_n(N)$ graph with dilation *3*, congestion *1*, load *1*, and unbounded expansion.

Proof: Every searching path is only one path according to the algorithm Mesh_Mapping, 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, it is a worse case in which the dilation=*1+2=3* at most by algorithm Mesh_Mapping. Because these nodes and links of searching paths are not replicated from algorithm Mesh_Mapping, These costs associated with graph embedding are dilation *3*, congestion *1*, load *1*, and unbounded expansion.

Theorem 7 A searching path of algorithm Mesh_Mapping is including $1/2*n^2 + (n* \lfloor \log_2 m \rfloor) + 3/2*n - \lfloor \log_2 m \rfloor^2 + 1$ nodes.

Proof: We can embed $M_{m_1 \times m_2 \times \cdots \times m_d}$ into $G_n(N)$ by theorem 6. 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 $\lfloor \log_2 m \rfloor$ from the second node of the queue, and we can also get $(\lfloor \log_2 m \rfloor+1)$ different nodes. Until the queue is empty, the sum of all searched nodes is $(n - \lfloor \log_2 m \rfloor + 1) * (\lfloor \log_2 m \rfloor + 1)) + (1 + 2 + \ldots + n)$. The search path includes $(n - \lfloor \log_2 m \rfloor + 1) * (\lfloor \log_2 m \rfloor$

$+ 1)) + (1 + 2 + ... + n)$ nodes. We assume there is an IC edge connecting to the faulty node by theorem 4.3. 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 $(n - \lfloor log_2 m \rfloor + 1) * (\lfloor log_2 m \rfloor + 1)) + (1 + 2 + ... + n - \lfloor log_2 m \rfloor) + \lfloor log_2 m \rfloor = 1/2 * n^2 + (n * \lfloor log_2 m \rfloor) + 3/2 * n - \lfloor log_2 m \rfloor^2 + 1$ nodes.

Theorem 8 There are $O(n^2 - \lfloor log_2 m \rfloor^2)$ faults, which can be tolerated.

Proof: By theorem 7, the whole search path includes $1/2 * n^2 + 3/2 * n + ((n+1) * \lfloor log_2 m \rfloor) - \lfloor log_2 m \rfloor^2$ nodes. That is, $O (n^2 - \lfloor log_2 m \rfloor^2)$ faults can be tolerated.

5 Conclusion

In [20], we consider the problem of embedding rings in IEH graphs. After [20], we consider the problem of embedding meshes in IEH graphs. According the result, we not only can map these parallel programs of rings, but also can map these parallel programs of meshes in an IEH. In [21], we consider the problem of embedding meshes in Supercubes. Because the IEH is an asymmetric Incrementally Extensible Hypercube, it fit in with the distributed system or cloud computing system more than the Supercube. Obviously, the IEH is superior to Supercube in terms of embedding meshes or tori under faults.

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 meshes and tori in a faulty IEH. After a mesh is mapped in an IEH, we develop new algorithms to facilitate the embedding meshes and tori in a faulty IEH. Our results demonstrate that $O(n^2 - \lfloor log_2 m \rfloor^2)$ faults can be tolerated. Also, the methodology is proven and an algorithm is presented to solve them. These existent parallel algorithms in mesh architectures can be easily transformed to or implemented in IEH architectures with load *1*, congestion *1*, dilation *3*, and unbounded expansion. Although an IEH is asymmetric, it has the same power as the hypercube in terms of meshes. The technology can be apply in grid computing and cloud computing.

References:

[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] J. R. Armstromg and F. G. Gray, Fault-diagnosis in n-Cube array of microprocessor, *IEEE Trans. on Computers*, Vol. C-30, No. 4, 1992, pp. 587-590.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: numerical methods*, Prentice Hall, Englewood Ciffs, New Jersey, 1989.

[4] L. Bhuyan and D.P. Agrawal, Generalized Hypercubes and Hyperbus structure for a computer network, *IEEE Trans. on Computers*, Vol. 33, 1984, pp. 323-333.

[5] C. Chartand and O. R. Oellermann, *Applied and Algorithmic Graph Theory,* McGRAW-HILL Inc., 1993.

[6] K. Day and A. E. Al-Ayyoub, Fault Diameter of k-ary n-cube Networks, *IEEE Trans. on parallel and distributed systems*, Vol. 8, No. 9, 1997, pp. 903-907.

[7] Q. Dong, X. Yang, J. Zhao, and Y. Y. Tang, Embedding a family of disjoint 3D meshes into a crossed cube, *Information Sciences*, Vol. 178, No. 11, 2008, pp. 2396-2405.

[8] S. Dutt and J. P. Hayes, An automorphic approach to the design of fault-tolerance Multiprocessor, *Proc. 19th Inter. Symp. on Fault-Tolerant Computing*, 1989.

[9] M. J. Duff, CLIP4: A Large Scale Integrated Circuit Array Parallel Processor, *IEEE International Joint Conference on Pattern Recognition*, 1976, pp. 728-733.

[10] M. J. Duff, Real Applications on CLIP4, *in Integrated Technology for Parallel Image Processing*, Academic Press London, 1985, pp. 153-165.

[11] T. Hameenanttila, X.-L. Guan, J. D. Carothers, and J.-X. Chen, The Flexible Hypercube: A New Fault-Tolerant Architecture for Parallel Computing, *Journal of Parallel and Distributed Computing*, Vol. 37, 1996, pp. 213-220.

[12] J. Hastad, T. Leighton, and M. Newman, Reconfiguring a Hypercube in the Presence of Faults, *ACM Theory of Computing*, 1987, pp. 274-284.

[13] J. P. Hayes, and T.N. Mudge, Hypercube supercomputing, *Proc. IEEE*, Vol. 77, 1989, pp. 1829-1842.

[14] H.P. Katseff, "Incomplete Hypercubes," *IEEE Trans. on Computers*, Vol. 37, 1988, pp. 604-608.

[15] J. Kuskin, et al., The Stanford FLASH Multiprocessor, *Proceedings of the 21st Annual*

International Symposium on Computer Architecture, 1994, pp. 302-313.

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

[17] D. Lenoski, et al., The StanfordDASH Multiprocessor, *Computer*, Vol. 224, 1971, pp. 63-79.

[18] J.-C. Lin, "Fault-Tolerant Mapping of a Mesh in a Flexible Hypercube", *WSEAS Transactions on Computers*, Vol. 8, 2009, pp. 1587-1596.

[19] J.-C. Lin, "Simulation of Cycles in the IEH Graph," *International Journal of Hjgh Speed Computing*, Vo1. 10, No. 3, pp. 327-342 (1999).

[20] 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, pp. 121-128, 2006.

[21] J.-C. Lin, S.-J. Wu, H.-C. Keh, and L. Wang, "Fault-Tolerant Meshes and Tori Embedded in a Faulty Supercube", *WSEAS Transactions on Computers*, Vol. 9, No. 5, pp. 445-454, 2010.

[22] C. D. Park, and K.-Y. Chwa, Hamiltonian properties on the class of hypercube-like networks, *Information Processing Letters*, Vol. 91, 2004, pp. 11-17.

[23] F. P. Preparata, and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, Vol. 24, 1981, pp. 300-309.

[24] D. A. Rennels, On Implemanting Fault-tolerance in binary hypercubes, *Proc. 16th Inter . Symp. on Fault-tolerant Computing*, 1986, pp. 344-349.

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

[26] J. L. C. Sanz, *The SIMD Model of Parallel Computation*, Springer-Verlag New-York, Inc., 1994.

[27] C. Seitz, The Cosmic Cube, *Commun. ACM*, Vol. 28, 1985, pp. 22-33.

[28] A. Sen, Supercube: An Optimally Fault Tolerant Network Architecture, *Acta Informatica*, Vol. 26, 1989, pp. 741-748.

[29] A. Sen, A. Sengupta and S. Bandyopadhyay, Generalized Supercube: An incrementally expandable interconnection network, *Proceedings of the Third Symposium on Frontiers of Massively Parallel Computation-Frontiers'90*, 1990, pp. 384-387.

[30] H. Sullivan, T. Bashkow, A large scale, homogeneous, fully distributed parallel machine, I, *Proc. 4th Symp. Computer Architecture, ACM,* 1977, pp. 105-177.

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

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

[33] L. W. Tucker and G. G. Robertson, Architecture and applications of the connection machine, *IEEE Comput.*, Vol. 21, 1988, pp.26-38.

[34] N.-F. Tzeng and H.-L. Chen, Fast Compaction in Hypercubes, *IEEE Trans. on parallel and distributed systems*, Vol. 9, No. 1, 1998, pp. 50-55.

[35] S.-H. Wang, Y.-R. Leu, and S.-Y. Kuo, Distributed Fault-Tolerant Embedding of Several Topologies in Hypercubes, *Journal of Information Science and Engineering*, Vol. 20, No. 4, 2004, pp. 707-732.

[36] L. D. Wittie, Communications structures for largenetworks of microcomputers, *IEEE Trans. Comput.*, Vol. C-30, 1981, pp.264-273.

[37] C. Xu and F. C. M. Lau, *Load Balancing in Parallel Computers-Theory and Practice*, Kluwer Academic Publishers, Inc., 1997.

[38] P.-J. Yang, S.-B. Tien, and C.S. Raghavendra, Embedding of Rings and Meshes onto Faulty Hypercube Using Free Dimensions, *IEEE Trans. on Computers*, Vol. 43, No. 5, 1994, pp. 608-618.