Fault-Tolerant Mapping of a Mesh Network in a Flexible Hypercube

*Jen-Chih Lin

Department of Digital Technology Design,
National Taipei University of Education,
No.134, Sec. 2, Heping E. Rd., Da-an District,
＜Taiwan, R.O.C.
E-mail:*yachih@tea.ntue.edu.tw

Abstract: - The Flexible Hypercube FH_N is an important variant of the Hypercube H_n and possesses many desirable properties for interconnection networks. This paper proposes a novel algorithm of fault-tolerant method for mapping a mesh network in a Flexible Hypercube. The main results obtained (1) a searching path of a FH_N is including approximate to $(n+1)$ nodes, where $n=\lfloor \log_2 N \rfloor$. Therefore, there are $O(\lfloor \log_2 N \rfloor)$ faults, which can be tolerated. (2) Our results for the mapping methods are optimized mainly for balancing the processor and communication link loads. These results mean that the parallel algorithms developed by the structure of mesh network can be executed in a faulty FH_N. The useful properties revealed and the algorithm proposed in this paper can find their way when the system designers evaluate a candidate network's competence and suitability, balancing regularity and other performance criteria, in choosing an interconnection network.

Key-Words: - Hypercube, Flexible Hypercube, Mesh Network, Fault-Tolerant, embedding

1 Introduction

In the study of parallel computing, networks of processors are often organized into various configurations such as trees, rings, linear arrays, *meshes* and *hypercubes*[16]. These configurations can be represented as graphs. If the properties and structures of underlying graph used effectively, the computation and communication speeds can often improved.

Hypercube multiprocessor has been the focus of many researchers over the past few years. Several prototypes and parallel systems have been constructed, such as Intel's Paragon, Dash[17], Flash[15], and Alewife[1]. At the same time, Hypercube has been used as the interconnection network in a wide variety of commercial and experimental distributed memory multiprocessors such as the Cosmic Cube[25], the Intel "hypercube" systems (iPSC, iPSC/2), the Ametek/Symult S-series, the NCUBE and the Connection Machines (CM-1, CM-2). The popularity of the hypercube multiprocessor or multicomputer systems is due to their tempting properties such as logarithmic diameter and node degree, high bisection width, ease to embed other common structures, and many known efficient data communication schemes[24]. Although hypercubes possess many advantages for parallel and distributed computing, there are some limitations for constructing hypercubes. However, due to the power-of-2 size and logarithmic degree, hypercubes suffer two major disadvantages, namely, high cost extensibility and large internal fragmentation in partitioning. Limitations of the hypercube include its nonplanarity (which complicates the layout of hypercubes implemented within VLSI chips), and its inability to grow incrementally. The incremental extensibility is a very essential and desirable property in real world applications for designing interconnection networks. At the same time, the *mesh*[8, 10, 31, 39] interconnection network is very popular. And it substitutes the hypercube interconnection in parallel machine.

Since hypercube architectures are very regular and scalable, programs for these computers are frequently designed with the dimension n of the host hypercube H_n treated as an input variable, so that the same program can be run without modification on hypercube of different sizes. This scalability property of hypercube architecture makes it possible to tolerate faults gracefully by confining a program to a fault-free subcube of the host hypercube. The reduction of the effective cube size due to faults is a tough indication of the performance degradation. In

order to conquer the difficulties associated with hypercubes and these generalizations of the hypercubes, the *Flexible Hypercube*[11] has been proposed during past years. The Flexible Hypercube unlike the hypercube, may be expanded (or designed) in a number of possible configurations while guaranteeing the same basic fault-tolerant properties and without a change in the communication. The existence of hypercube subgraphs in the Flexible Hypercube ensures that hypercube embedding algorithms developed for the hypercube may also be utilized in the Flexible Hypercube. The flexibility in node placement may possibly be utilized to aid in supporting a specific embedding. The Flexible Hypercube, while maintaining the fault-tolerant of the other topologies and the ease of communication, allows the placement of new nodes at any currently unused addresses in the system.

In order to execute a parallel program, tasks are to be mapped in a processor of parallel machine or interconnection network. It is possible to model this kind of problem in graph-theoretical terms of graph embedding[2, 20]. We model both the parallel algorithm and the parallel machine as graphs. Given two graphs, *G(V,E)* and *G'(V',E')*, embedding the guest graph *G* into the host graph *G'* maps each vertex in the set *V* in a vertex(or a set of vertices) in the set *V'* and each edge in the set *E* in an edge(or a set of edges) in the set *E'*. Let these nodes in a graph correspond to processors and edges to communication links in an interconnection network. Embedding one graph into another is important because an algorithm may have been designed for a specific interconnection network, and it may be necessary to adapt it to another network. Four costs associated with graph embedding are *dilation*, *expansion*, *load*, and *congestion*. The maximum amount that we must stretch any edge to achieve an embedding is called the dilation of the embedding. By expansion, we mean the ratio of the number of nodes in the host graph to the number of nodes in the graph that is being embedded. The congestion of the embedding is the maximum number of edges of the of the guest graph that are embedded using any single edge of the host graph. The load of an embedding is the maximum number of nodes of the guest graph that are embedded in any single node of the host graph. An efficient simulation of one network on another network requires that these four costs be as small as possible. However, for most embedding problems, it is impossible to obtain an embedding that minimizes these costs simultaneously. Therefore, some tradeoffs among these costs must be made.

The issue of computing with faulty hypercubes has been addressed in several recent papers [3, 7, 12, 13, 23, 32, 35, 36, 39]. Particularly notable is the result by Hastad, Leighton and Newman [12]. They considered a faulty hypercube in which 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.

Load Balancing, communication locality, communication congestion, and node utility in process graphs can be abstractly studies as the problem of embedding[2]. In a process graph, the nodes represent processes comprising a distributed program or a parallel program and the edges represent communications between processes. In a multiprocessor system, we follow two fault models defined in [16]. The first model assumes that, in a faulty node, the computational function of the node is lost while the communication function remains intact; this is the partial faulty model. The second model assumes that, in a faulty node, the communication function is lost too; this is the total faulty model. Conceptually, the network interface hardware operates independent of the computer's processor. In this paper, our model is the partial faulty model. That is, when the computation nodes are faulty, the communication links are well and only the faulty nodes are remapped.

One approach to achieve fault-tolerant in hypercubes is to introduce spare nodes or links[36], so that hypercube structure can still be maintained when nodes fail. This approach can be expensive and it is difficult to make hardware modifications on those machines already in the market place. Another approach exploits the inherent redundant nodes or links in hypercube to achieve fault-tolerant[16]; that is no extra nodes or links are unused nodes as spares. In this paper, we consider only the second type of fault-tolerant design in a Flexible Hypercube.

The paper presents novel algorithms to facilitate the embedding job when the Flexible Hypercube contains faulty nodes. Of particular concern are the network structures of the Flexible Hypercube that balance the load before as well as after faults starting to degrade the performance of the Flexible Hypercube. To obtain replaceable nodes of faulty nodes, *2*-expansion is permitted such that up to *(n+1)* faults can be tolerated with congestion *1*, dilation *2*, and load *1*, where $n=\lfloor \log_2 N \rfloor$ is the dimension of a Flexible Hypercube FH_N. Results presented herein demonstrate that embedding methods are optimized.

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 mesh network. In Section 4, we describe the novel fault-tolerant method for mapping a mesh network in a Flexible Hypercube with *2*-expansion. Conclusions are finally made in section 5.

2 Preliminaries

We briefly describe these definitions of these topologies of the hypercube, the mesh network, and the flexible hypercube.

A hypercube H_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*.

There are many topologies can be mapped in hypercubes or hypercube-like computers. One of these is mesh network. It is very popular network interconnection. One of the most attractive properties of the binary n-cube topology is that meshes of arbitrary dimensions can be mapped in it. This is one of the main reasons for the success of hypercube architectures. Because of these, we consider the mesh size in each direction is a power of *2*. The figure 1 and the figure 2 show us two examples. First example, a 2×2 *2*-dimensional mesh has *4* nodes which are bi-directional connection between two nodes. Second example, A $2^2 \times 2^1$ *2*-dimensional mesh has *8* nodes which are bi-directional connection between two nodes.

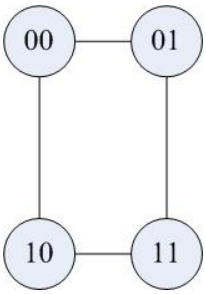

Fig. 1: A 2×2 *2*-dimensional mesh

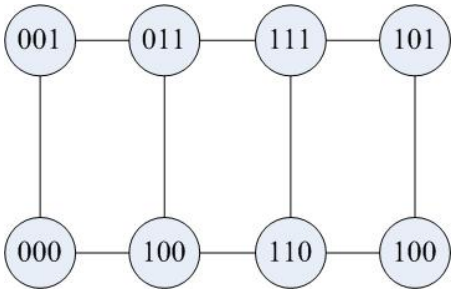

Fig. 2: A $2^2\times 2^1$ *2*-dimensional mesh

Definition 1[16] $m_1 \times m_2$ mesh is a *2*-dimension mesh that the mesh size in each direction is a power of *2*. i.e., it is such that $m_1 = 2^r, m_2 = 2^s$.□

Definition 2[16] Higher dimension mesh is a $m_1 \times m_2 \times \cdots \times m_d$ mesh in d-dimension, and assume that the mesh size in each direction is a power of *2*. □

Definition 3[18] 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 *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 $C_3=0C_2 \cup 1(C_2)^R=\{000, 001, 011, 010, 110, 111, 101, 100\}$.

The Flexible Hypercube is constructed by any number of nodes and based on a hypercube. A Flexible Hypercube, denoted by FH_N, is defined as an undirected graph $FH_N=(V,E)$,where V is the set of processors (called nodes) and E is the set of

bidirectional communication links between the processors (called edges). In an n-dimensional Flexible Hypercube with N nodes where $2^n \le N < 2^{n+1}$ (n is a positive integer), each node can be expressed by an $(n+1)$-bit binary string $i_n.....i_0$ where $i_p \in \{0, 1\}$ and $0 \le p \le n$

Definition 4[19] A $(2^{n+1}-t)$-node Flexible Hypercube is a lack of t nodes, which are referred to herein as virtual nodes. For any virtual node y, denoted as $I(x)$ where x is any node of Flexible Hypercube, if the function $I(x)$ exists, then $x_n = \overline{y_n}$ and $x_i=y_i$ for $0 \le i \le n-1$

Definition 5[18, 21] The Hamming distance of two nodes x and y, denoted by $HD(x, y)$, is the number of *1*'s in the bit set of resulting sequence of the bitwise *XOR* of x and y.

Definition 6[19] For any two nodes x and y in a Flexible Hypercube, let $x=x_n \ldots x_0$, $y=y_n \ldots y_0$, then $Dim(x, y)=\{i \text{ in } (0 \ldots n) \mid x_i \neq y_i\}$.

Definition 7[11] Suppose $FH_N=(V, E)$ is an n-dimensional Flexible Hypercube ,then the node sets H_1, H_2, V_1, V_2, V_3 are defined as follows

1. $H_1 = \{x \mid x \in V \text{ and } x_n = 0\}$,
2. $H_2 = \{x \mid x \in V \text{ and } (x_n = 1 \text{ or } I(x) \notin V)\}$,
3. $V_1 = H_1 - H_2$
4. $V_2 = H_1 \cap H_2$
5. $V_3 = H_2 - H_1$

Definition 8[11] Suppose $FH_N=(V, E)$ is an n-dimensional Flexible Hypercube ,then the edge set E is the union of E_1, E_2, E_3, and E_4 ,where

1. $E_1 = \{(x, y) \mid x, y \in H_1 \text{ and } HD(x, y) = 1\}$,
2. $E_2 = \{(x, y) \mid x, y \in V_3 \text{ and } HD(x, y) = 1\}$,
3. $E_3 = \{(x, y) \mid x \in V_3, y \in V_1 \text{ and } HD(x, y) = 1\}$,
4. $E_4 = \{(x, y) \mid x \in V_3, y \in V_2 \text{ and } HD(x, y) = 2\}$.

Addressing of nodes in a Flexible Hypercube is constructed as follows. As discussed above, addresses consist of binary strings of $(n+1)$-bits. The first 2^n-1 addresses correspond to nodes in H_1 and must be the binary representations of *0* through 2^n-1. Each of the remaining nodes (up to 2^n-1 nodes) in the set $V_3=H_2-H_1$ may be placed adjacent to any node x in H_1 and is given the addressing $I(x)$. Any node in H_1 is a hamming distance of *1* from at most one node in V_3. This method of node addressing effectively relaxes the constraint that all nodes in the network must be numbered consecutively. This is unique among the hypercube topologies mentions above. Notably, hypercubes are special cases of a Flexible Hypercube; it can also be expanded flexibly with respect to the placement of new nodes in the system while maintaining fault-tolerant. When a new node is added to a Flexible Hypercube system, $(n+1)$ new connections should be added and at most n existing edges must be removed.

An inevitable consequence of the flexible of construction and the fault-tolerant of a Flexible Hypercube is an uneven distribution of the utilized communication ports over system nodes. Although the Flexible Hypercube loses its property of regularity, more links help obtain the replacement nodes of the faulty nodes of the Flexible Hypercube. The Flexible Hypercube with *14*-node is shown in the figure 3. In the figure 3, $H_1 = \{0000, 0001, 00010, 0011, 0100, 0101, 0110, 0111\}$, $H_2 = \{0001, 0011, 1000, 1010, 1100, 1101, 1110, 1111\}$, $V_1 = \{0000, 0010, 0100, 0101, 0110, 0111\}$, $V_2 = \{0001, 0011\}$, and $V_3 = \{1000, 1010, 1100, 1101, 1110, 1111\}$.

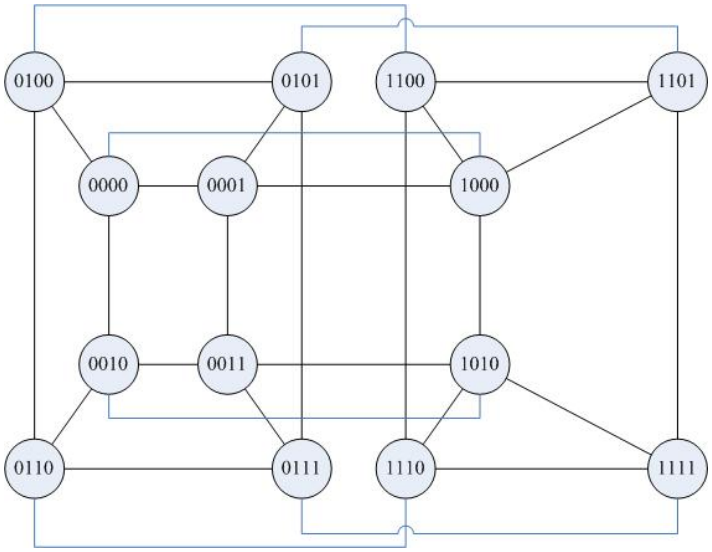

Fig. 3: A Flexible Hypercube contains 14-nodes

3 Mapping of Meshes

We describe our approach that maps a mesh network in a Flexible Hypercube with *2*-expansion in this section.

Lemma 1[16] $m_1 \times m_2$ mesh is a *2*-dimensional mesh, where $m_1 = 2^r, m_2 = 2^s$ can be mapped in an *n*-dimensional hypercube where $n = r+ s$. □

Lemma 2[16] Any $m_1 \times m_2 \times \cdots \times m_d$ mesh in the *d*-dimensional space R_d, where $m_i = 2^{p_i}$ can be mapped in an *n*-dimensional hypercube where $n = p_1 + p_2 + \ldots + p_d$. The numbering of the mesh nodes is any numbering such that its restriction to each i_{th} variable is a Gray sequence which is described in definition 3. Note that the assumption that all m_i's be power of *2*. □

Our proposition is best illustrated by an example. Consider a *2*-dimensional 8×4 mesh i.e., $d = 2$, $p_1 = 3$, $p_2 = 2$, $n = p_1 + p_2 = 5$. A binary number *M* of any node of the *3*-dimensional hypercube can be regarded as consisting of two parts: its first *3* bits and its last *2* bits, which we write in the form

$M = \alpha_1\alpha_2\alpha_3\beta_1\beta_2$, where α_i and β_i are bits *0* or *1*. It is clear from the definition of *n*-dimensional hypercube that when the last *2* bits are fixed, then the resulting 2^{p_1} nodes form a p_1-dimensional hypercube (with $p_1 = 3$). Whenever we fix the first *3* bits we obtain a p_2-dimensional hypercube (with $p_2 = 2$). The mapping then becomes clear. Choosing a *3*-bit *BRGC* for the *x* direction and *2*-bit *BRGC* for the *y* direction, the point (x_i, y_i) of the mesh is assigned to the node $\alpha_1\alpha_2\alpha_3\beta_1\beta_2$ where $\alpha_1\alpha_2\alpha_3$ is the *3*-bit *BRGC* for dimension of p_1 while $\beta_1\beta_2$ is the *2*-bit *BRGC* for dimension of p_2.

The binary node number of any mesh 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 nodes forms a Gray sequence and any row of mesh 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 a Flexible Hypercube FH_N, where $2^n \le N < 2^{n+1}$.

Proof. A FH_N must contain a hypercube H_n. That is trivially by the generation schema of a FH_N graph. It must contain the maximum hypercube H_n. □

The mapping approach that a mesh can be mapped in a FH_N with *2*-expansion is as follows.

Mapping approach

M, FH_N, $\forall r+s = \lfloor \log_2 N \rfloor$, $r, s \ge 1$

$FH_N = G(V, E)$, $M_{2^r * 2^s} = G(V', E')$,

$v \in V \quad v' \in V'$

(Denoted by unique binary string)

$v = X_{r+s}X_{r+s-1}X_{r+s-2}\cdots X_1X_0$

$v' = X_{r+s-1}\cdots X_1X_0$

$v' \in V'$ can be mapped in V denote as

$0_{r+s}X_{r+s-1}X_{r+s-2}\cdots X_1X_0$ □

Theorem 1 A $2^r \times 2^s$ *2*-dimensional mesh can be mapped in FH_N where $2^n \le N < 2^{n+1}$ that $n = r + s$ with load *1*, dilation *1*, congestion *1* and expansion *2*.

Proof. This is trivial by lemma 2, lemma 3 and the above mapping approach. □

Theorem 2 Any $m_1 \times m_2 \times \cdots \times m_d$ *d*-dimensional mesh, where $m_i = 2^{p_i}$ can be mapped in a FH_N, where $2^n \le N < 2^{n+1}$, that $n = p_1 + p_2 + \ldots + p_d$ with load *1*, dilation *1*, congestion *1* and expansion *2*.

Proof. It is similarly with above approach. □

This is the best illustrated by two examples in the figure 4 and the figure 5. That is a 2×2 mesh (with *4* nodes) can be mapped in a FH_7 with *2*-expansion.

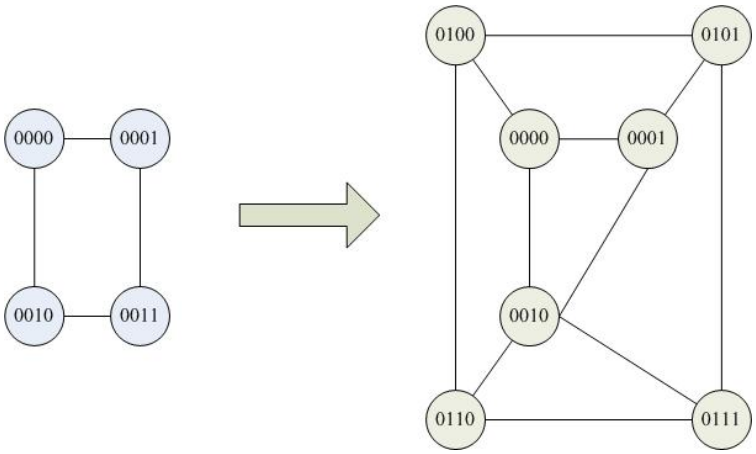

Fig. 4: A *2×2* mesh can be mapped in FH_7 with *2*-expansion

The second example is a $2^2 \times 2^1$ mesh (with *8* nodes) can be mapped in a FH_{14} with *2*-expansion.

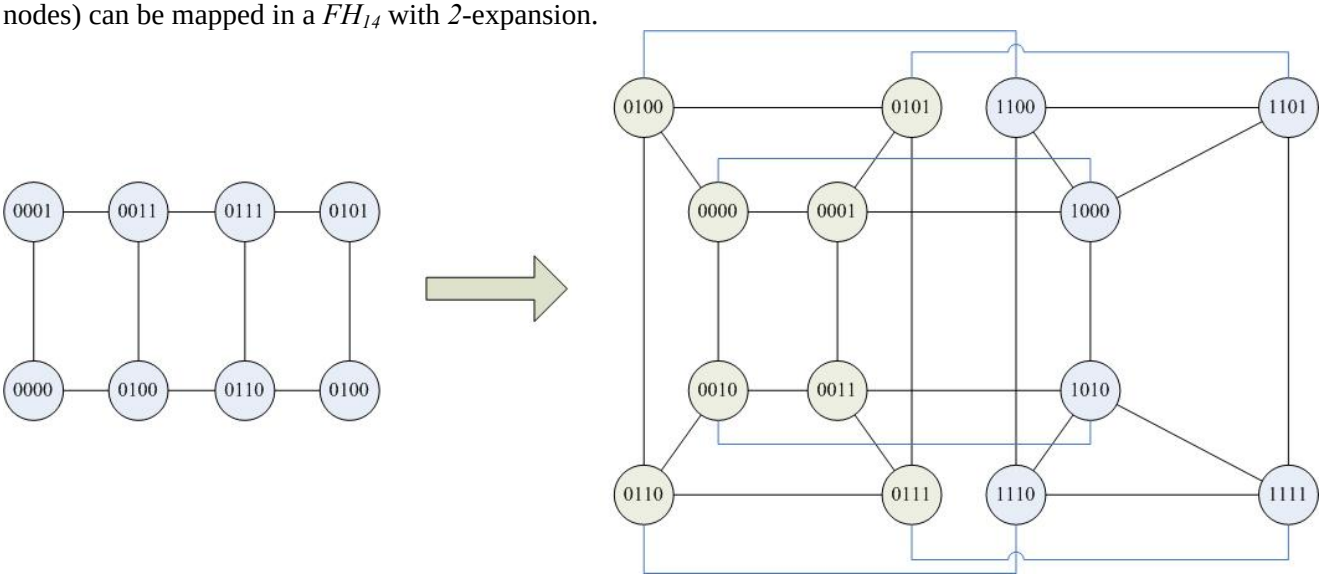

Fig. 5: A $2^2 \times 2^1$ mesh can be mapped in FH_{14} with *2*-expansion

4 Fault-Tolerant mapping of meshes

In session 3, we show that a mesh network can be mapped in a FH_N graph with expansion *2*, load *1*, congestion *1*, and dilation *1*. Hence, in this section, we consider a mesh network can be mapped in a FH_N with *2*-expansion graph which contains faulty node.

The algorithm design described in this section mainly accorded with the idea the search of highest dimension first. By applying the bit-flip in the leading bit in order from left to right, the searching path can reach the higher half node immediately and then expand out to its neighbors in any *(n-1)* directions till we find the replaceable node, where $n = \lfloor \log_2 N \rfloor$ is the dimension of the graph. This kind of strategy is called the strategy in higher dimension of precedence in this session. We also propose a novel algorithm for mapping a mesh network in a faulty FH_N with *2*-expansion as follows.

Algorithm MtoFH(x)

Input: *x* /*the faulty node*/,
FH_N,
Mesh $2^r \times 2^s$, where
$r + s = \lfloor \log_2 N \rfloor$

Output: *y* /*the replaceable node*/

1. *i=0*
2. if a node *x* is faulty
3. then
4. {
5. search the node *p*
/* *HD(x, p)=1*, *Dim(x, p)={n}*/

```
6.      if p is a exist node and it is free
7.      then
8.          return(p) /*replace x with p*/
9.          exit()
10.     else
11.         while i < n do
12.         {
13.         search the node q
            /* HD(x, q)=2, Dim(x, q)={n, i}*/
14.         if q is a exist node and it is free
15.         then
16.         return(q) /*replace x with q*/
17.         exit()
18.         }
19.         i=i+1
20.   }
21. return("Failure")
22. end
```

By the algorithm *MtoFH( )*, the searching path of the faulty node is shown as follows.

$node0=0X_{n-1}X_{n-2}\ldots X_1X_0$
$node1=1X_{n-1}X_{n-2}\ldots X_1X_0$
$node2=1X_{n-1}X_{n-2}\ldots X_1 X'_0$
$node3=1X_{n-1}X_{n-2}\ldots X'_1 X_0$
$\vdots$
$node(n+1)=1X'_{n-1}X_{n-2}\ldots X_1 X_0$

We illustrate two examples of finding a replaceable node in a FH_{14} as shown the figure 6 and the figure 7. The first example is a 2×2 mesh (with *8* nodes) can be mapped in a FH_{14} with *2*-expansion.

When the faulty node (*000*) exists, we execute the operations of the *MtoFH( )*.

All node of the searching path is listed as *{(100}, (101), (110)}*.

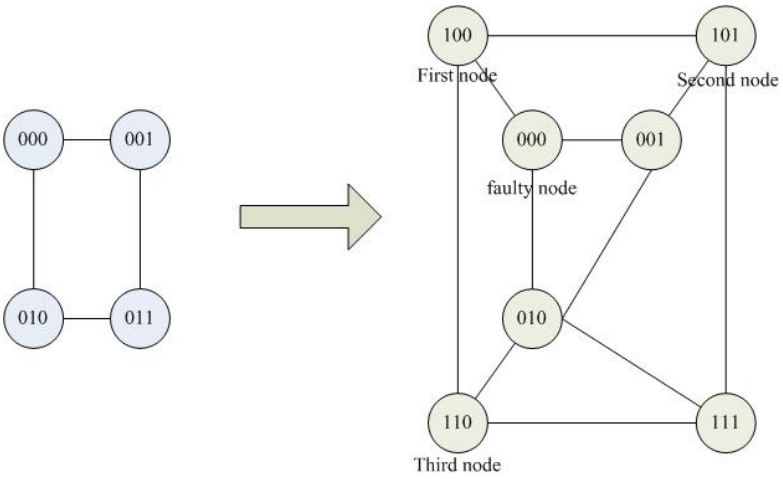

Fig. 6: Mapping a $2^2\times2^1$ mesh in a faulty FH_{14}

The second example is a 4×2 mesh (with *8* nodes) can be mapped in a FH_{14} with *2*-expansion.

When the faulty node (*0100*) exists, we explain the operations of the *MtoFH( )*. All node of the searching path is listed as *{(1100, (1101), (1110), (1000)}*.

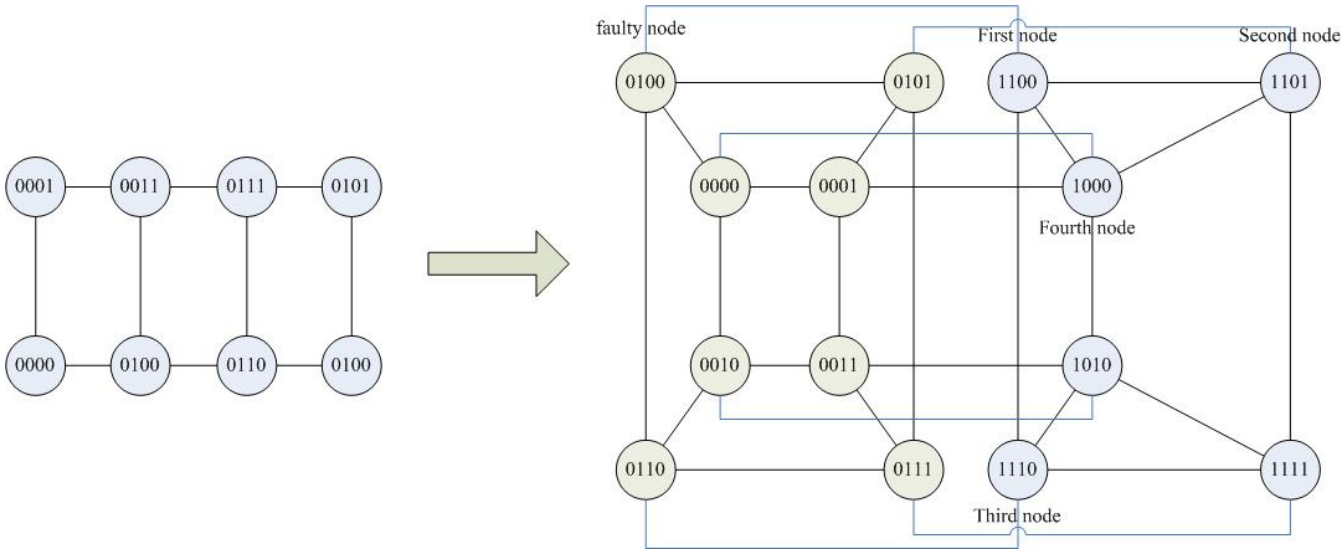

Fig. 7: Mapping a $2^2 \times 2^1$ mesh in a faulty FH_{14}

Theorem 3 A $2^r \times 2^s$ *2*-dimensional mesh can be mapped in a faulty FH_N that $r+s=\lfloor \log_2 N \rfloor$ with load *1*, dilation *2*, congestion *1* and expansion *2*.

Proof. By executing the *MtoFH* method, allowing us to get congestion *1* and load *1*. And we allow *2*-expansion to obtain the replace node of faulty node. When a node is faulty, the dilation maybe become *1+1=2* at most by the *MtoFH* method in a worst case. Because these nodes and links of searching paths are not replicated from the *MtoFH* method, four costs associated with graph mapping are dilation *2*, expansion *2*, load *1* and congestion *1*. □

Theorem 4 A searching path of the *MtoFH* method is including approximate to *(n+1)* nodes, where $n=\lfloor \log_2 N \rfloor$.

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 *MtoFH* method. But FH_N graph may be having virtual nodes. Hence, a searching path of the *MtoFH* method is including approximate to *(n+1)* nodes. □

Theorem 5 There are $O(\lfloor \log_2 N \rfloor)$ faults, which can be tolerated.

Proof. It is trivial by theorem 4. □

Theorem 6 Our results for the mapping methods are optimized mainly for balancing the processor and communication link loads.

Proof. Our results demonstrate that a mesh network can be mapped in a faulty flexible hypercube with load *1*, dilation *2*, congestion *1* and expansion *2*. These nodes and links of these replacing nodes are not replicated from the algorithm *MtoFH( )*. This observation implies that the primary optimization objective of mapping is minimizing the interprocessor communication cost and to balance the workload of processors have reached.□

5 Conclusions

This paper develops a new algorithm to facilitate the mapping(embedding) job when the Flexible Hypercube contains faulty nodes. The replaceable node of the faulty node is obtained, allowing us *2*-expansion. Our results demonstrate that $O(\lfloor \log_2 N \rfloor)$ faults can be tolerated. Also, the methodology is proven and an algorithm is presented to solve them. These existent parallel algorithms on mesh architectures to be easily transformed to or implemented on Flexible Hypercube architectures with load *1*, congestion *1* and dilation *2*.

After any mesh networks can be reconfiguring in a Flexible Hypercube with faulty nodes, we are also interested in the mapping of arbitrary multi-dimensional mesh networks in a faulty flexible hypercube with unbounded expansion.

According to the result, we can embed the parallel algorithms developed by the structure of mesh in a FH_N. These methods of reconfiguring enable extremely high-speed parallel computation and internet computing. Therefore, we can easily port the parallel or distributed algorithms developed for these structures to the FH_N graphs.

References:

[1] A. Agrawal, et al., The MIT Alewife Machine: Architecture and Performance, *Proceedings of the 22nd Annual International Symposium on*

Computer Architecture, 1995. pp. 2-13.

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

[3] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] J. Fan, and X. Jia, Embedding meshes into crossed cubes, *Information Sciences*, Vol. 177, No. 15, 2007, pp. 3151-3160.

[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, No. 5, 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, Simulation of Cycles in the IEH Graph, *International Journal of High Speed Computing*, Vol. 10, 1999, pp. 327-342.

[19] J.-C. Lin, T.-H. Chi, H.-C. Keh and A.-H. A. Liou, Embedding of Complete Binary Tree with 2-expansion in a Faulty Flexible Hypercube, *Journal of Systems Architecture*, Vol. 47, No. 6 , 2001, pp. 543-548.

[20] J.-C. Lin, Load-Balance and Fault-Tolerance for embedding a Complete Binary Tree in an IEH with N-expansion, *WSEAS Transactions on Computers*, Vol. 7, No. 7, 2008, pp. 919-928.

[21] 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.

[22] F. P. Preparata and J. Vuillemin, The cube-connected cycles: A versatile network for parallel computation, *Commun. ACM*, Vol. 24, No. 5, , 1981, pp. 300-309.

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

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

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

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

[27] 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.

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

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

[30] S. Sur, and P. K. Srimani, IEH graphs: A novel

generalization of hypercube graphs, *Acta Informatica*, Volume 32, 1995, pp 597-609.

[31] C.-H. Tsai, Embedding of meshes in Möbius cubes, *Theoretical Computer Science*, Vol. 401, No. 1, 2008, pp. 181-190.

[32] Y.-C. Tseng and T.-H. Lai, On the Embedding of a class of Regular Graphs in a Faulty Hypercube, *J. Parallel and Distrib. Comput.*, Vol. 37, 1996, pp. 200-206.

[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, An Effective Approach to the Enhancement of Incomplete Hypercube Computers, *J. Parallel and Distrib. Comput.,* Vol. 14, 1992, pp. 163-174.

[35] 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.

[36] 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.

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

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

[39] 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.

[40] S.-M. Yuan, Topological properties of supercube, *Information Processing Letters*, Vol. 37, 1991, pp. 241-245.

[41] 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.