Fault-Tolerant Meshes and Tori Embedded in a Faulty Supercube

*Jen-Chih Lin[1], [2]Shih-Jung Wu, Huan-Chao Keh[3], and Lu Wang[4]
[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, R.O.C.
E-mail:*yachih@tea.ntue.edu.tw
[2] Department of Innovative Information and Technology,
Tamkang University,
No. 180 Linwei Road., Chiao-hsi Shiang,
I-lan County 26247, Taiwan, R.O.C.
[3]Department of Computer Science and Information Engineering,
Tamkang University
No. 151 Ying-chuan Road, Tamsui,
Taipei 251, Taiwan, R.O.C.
[4]Graduate Institute of Management Science,
Tamkang University,
No. 151 Ying-chuan Road, Tamsui,
Taipei 251, Taiwan, R.O.C.

Abstract: - Hypercubes, meshes, and tori are well known interconnection networks for parallel computing. The Supercube network is a generalization of the hypercube. The main advantage of this network is that it has the same connectivity and diameter as that of the hypercube without the constraint that the number of nodes be a power of *2*. This paper proposes novel algorithms of fault-tolerant meshes and tori embedded in supercubes with node failures. The main results obtained (1) a replacing sequence of a supercube is including approximate to $(\lfloor \log_2 N \rfloor + 1)$ nodes. Therefore, there are $O(\lfloor \log_2 N \rfloor)$ faults, which can be tolerated. (2) The result implies that optimal simulation of mesh and torus in a faulty supercube for balancing the processor and communication link loads at present. According to the result, we can easily port the parallel or distributed algorithms developed for these structures to the supercubes. Therefore, these methods of reconfiguring enable extremely high-speed parallel computation.

Key-Words: - fault-tolerant, mesh, tori, graph embedding, supercube

1 Introduction

Selection of an appropriate interconnection network is the key to the design of any distributed/ multiprocessor system, because the speed of internode communication, rather than that of computation, is known to be the bottleneck in accomplishing speedup with multiple processors. Over the past two decades, an overwhelming number of interconnection networks have been reported in the literature. Examples include crossbars, multiple bused, multistage interconnection networks, and *hypercubes*[24], to name a few. Among these, the hypercube has received considerable attention due mainly to its rich topological properties. The hypercube is a regular structure, has a small diameter, and offers good connectivity with a relatively small node degree. Moreover, a number of other well-known topologies, such as rings, trees, meshes, and tori can be embedded in the hypercube. The hypercube is a popular architecture for parallel machines, but it cannot be a general architecture

widely used because hypercubes are not incrementally expandable. The *Supercube*[27, 42] is one of *hypercube-derived computers*[25] and does not have the drawbacks from the hypercube. Speaking of its architecture, a supercube has the same connectivity and diameter as the corresponding hypercube. It is shown to have the following desirable characteristics:

1. The nodes connectivity of an *N*-node supercube is at least $\lfloor \log_2 N \rfloor$.
2. The node degree of an *N*-node supercube is between *(k - 1)* and *(2k - 2)*, where $k=\lceil \log_2 N \rceil$.
3. Adding a new node to an existing network is easy; it does not need reorganization of existing edges.
4. The diameter of an *N*-node supercube is $\lfloor \log_2 N \rfloor$ at most.

A *mesh connected computer*[7, 11, 20, 25, 32] is easy to construct because it is regular, it has short connections, it requires only four connections per node, and it is possible to build in two dimensions without having any connections cross. Each node that is not on an edge of the array has a direct connection with its four nearest neighbors. At the same time, the top row is connected to the bottom row and leftmost column is connected to the rightmost column, so the interconnections logically form a *torus*[25]. The construction of such a machine in two dimensions requires that some connections cross.

The mesh and torus are two of the most important networks for parallel computers. A great deal of research has focused on the mesh and torus networks and several parallel computers have been built with 2- or 3-dimensional mesh or torus topologies. Examples include the CLIP4[9, 10], the GAPP25 (NCR Microelectronic Products Division), the MPP[25] (of Goodyear Aerospace), the MP-1[25] (sold by MASPAR Corporation), and the J-machine[25] is a project at MIT in a *3*-dimensional mesh topology. Mesh connected computers were shown to be efficient in performing many image and matrix operations. If a mesh connected computer can be simulated with a hypercube or a hypercube-derived computer, those same algorithms can be used on these other topologies. In the paper, one of the most important issues in the design of a system which contains many components is the system's performance in the presence faults.

In a multiprocessor system, we follow two fault models. 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*. 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. In this paper, we consider only the second type of fault-tolerant design in a supercube.

The power of a message-passing parallel computer depends on the topology chosen for underlying interconnection network, which can be modeled as undirected graph. Different graphs have been proposed as static interconnection topology for multiprocessors. Therefore, we model both the parallel algorithm and the parallel machine as graphs. *Graph embedding*[16] problem have application in a wide variety of computational situations. For example, the flow of information in a parallel algorithm defines a program graph, and embedding this in a network tells us how to organize the computation on the network. Other problems are laying out circuits on chips, representing data structures in computer memory, and finding efficient program control structures.

Given two graphs, *G(V, E)* and *G′(V′, E′)*, embedding the guest graph *G* in the host graph *G′* maps each vertex in the set *V* into a vertex (or a set of vertices) in the set *V′* and each edge in the set *E* into an edge(or a set of edge)in the *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. 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 an embedding is the maximum number of edges 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 paper develops novel algorithms to facilitate the embedding job when the supercube contains faulty nodes. Of particular concern are the network structures of the supercube that balance the load before as well as after faults starting to demote the performance of the supercube. To obtain replaceable nodes of faulty nodes, *2*-expansion is permitted such that up to $(\lfloor \log_2 N \rfloor + 1)$ faults can be tolerated with congestion *1*, dilation *2*, and load *1*, where $\lfloor \log_2 N$ is the dimension of a supercube. Results presented herein demonstrate that embedding methods are optimized.

The remaining part of this paper is organized as follows. Section 2 introduces the topological properties of the hypercube, the mesh, the torus, and the supercube. Notations and definitions of terms are also provided. In section 3, the paper presents the method for embedding meshes and tori in a supercube. Section 4 describes the novel fault-tolerant algorithm for embedding meshes and tori in a faulty supercube with *2*-expansion. Conclusions are finally made in section 5.

2 Preliminaries

This section formally introduces these definitions of these topologies of the hypercube, the mesh, the torus, and the supercube. For completeness, we begin with the list of useful properties of above interconnection networks.

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 embedded in hypercubes or hypercube-derived computers. One of these is mesh and torus. It is very popular network interconnection. One of the most attractive properties of the binary n-cube topology is that meshes and tori of arbitrary dimensions can be embedded in it. This is one of the main reasons for the success of hypercube architectures. Because of these, we consider the mesh and torus size in each direction is a power of *2*. Figure 1-1 and Figure 1-2 show us two examples. First example is a $2^2 \times 2^1$ *2*-dimensional mesh and second example is a $2^2 \times 2^2$ *2*-dimensional torus which are bi-directional connection between two nodes.

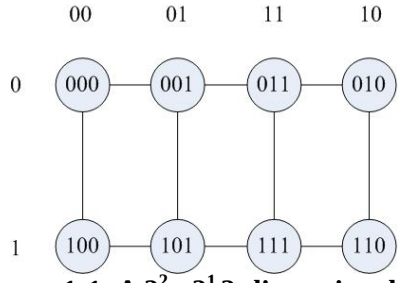


Figure 1-1: A $2^2 \times 2^1$ 2-dimensional mesh

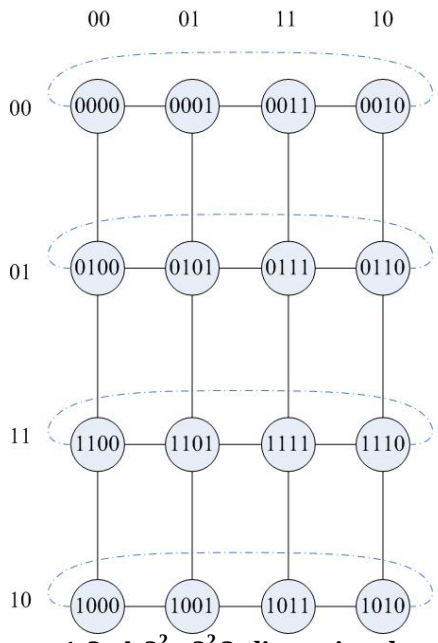


Figure 1-2: A $2^2 \times 2^2$ 2-dimensional torus

Definition 1[18, 38] 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 2[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 *2*, and insert a cipher in front of each codeword in C_1, then insert an one in front of each codeword in $(C_1)^R$. We get the code $C_2=\{00, 01, 11, 10\}$. Now, we can then repeat the procedure to built a *3*-bit Gray Code, and also get the code $C_3=0C_2 \cup 1(C_2)^R=\{000, 001, 011, 010, 110, 111, 101, 100\}$.

Definition 3[20] $m_1 \times m_2$ mesh or torus is a *2*-dimension mesh or torus that the mesh or torus size in each direction is a power of *2*. i.e., it is such that $m_1 = 2^r, m_2 = 2^s$.□

Definition 4[20] Higher dimension mesh or torus is a $m_1 \times m_2 \times \cdots \times m_d$ mesh or torus in *d*-dimension, and assume that the mesh or torus size in each direction is a power of *2*. □

Definition 5[19] For any two nodes *x* and *y*, let $x=x_n \ldots x_0$, $y=y_n \ldots y_0$, then $Dim(x ,y)=\{i \text{ in } (0 \ldots n) \mid x_i \neq y_i\}$.

A supercube is constructed by any number of nodes and based on hypercube. A supercube, denoted by S_N, is defined as an undirected graph $S_N = (V, E)$, where *V* is the set of processors (called nodes in our discussion) and *E* is the set of bidirectional communication links between the processors (called edges). Assume that *V* contains *N* nodes and each node can be numbered by an identical number in the range over *(0, N-1)*, in a $\lfloor \log_2 N \rfloor$-dimensional supercube, each node can be expressed by a $(\lfloor \log_2 N \rfloor +1)$-bit binary string, where *N* is a positive integer.

Definition 6[27, 42] Suppose $S_N = (V, E)$ is a $\lfloor \log_2 N \rfloor$-dimensional supercube, then the node set *V* can be divided into three subsets V_1, V_2, V_3, where

1. $V_3 = \{x \mid x \in V, x = 1u, \text{where } u \text{ is } \lfloor \log_2 N \rfloor\text{-bit sequences}\}$.
2. $V_2 = \{x \mid x \in V, x = 0u, 1u \text{ does not exist in } V, \text{where } u \text{ is } \lfloor \log_2 N \rfloor\text{-bit sequences}\}$,and
3. $V_1 = \{x \mid x \in V, x = 0u, 1u \in V, \text{where } u \text{ is } \lfloor \log_2 N \rfloor\text{-bit sequences}\}$.

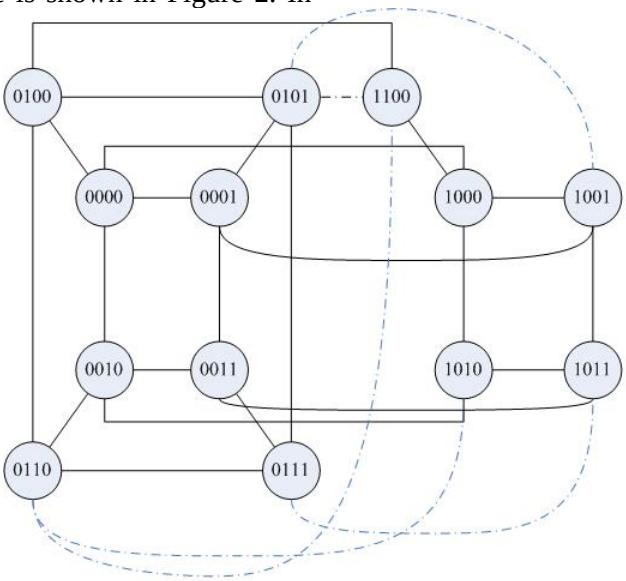
Definition 7[27, 42] Suppose $S_N = (V, E)$ is a $\lfloor \log_2 N \rfloor$-dimensional supercube, then the edge set *E* is the union of E_1, E_2, E_3 and E_4, where

1. $E_1 = \{(x, y) \mid x, y \in V, x = 0u, y = 0v, \text{where } u, v \text{ are } \lfloor \log_2 N \rfloor\text{-bit sequences and } HD(x, y)= 1\}$,
2. $E_2 = \{(x, y) \mid x, y \text{ in } V_3, x = 1u, y = 1v, \text{where } u, v \text{ are } \lfloor \log_2 N \rfloor\text{-bit sequences and } HD(x, y)= 1\}$,
3. $E_3 = \{(x, y) \mid x \text{ in } V_3, y \text{ in } V_2, x = 1u, y = 0 u, \text{where } u, v \text{ are } \lfloor \log_2 N \rfloor\text{-bit sequences and}$

4. *E_4 ={(x, y)| x in V_3, y in V_1, x = 1u, y = 0v, where u is n-bit sequences }.*

We illustrate the supercube with 13-node is shown in Figure 2.

An inevitable consequence of the flexible of construction and the fault-tolerant of a supercube is an uneven distribution of the utilized communication ports over system nodes. Although the supercube loses its property of regularity, more links help obtain the replacement nodes of the faulty nodes of the supercube. The supercube with *13*-node is shown in Figure 2. In the Figure 2, *V1 = {0000, 0001, 0010, 0011, 0100}, V2 = {0101, 0110, 0111}, V3 = {1000, 1001, 1010, 1011, 1100}, E1 = {(0000, 0001), (0000, 0010), (0000, 0100), (0001, 0011), (0001, 0101), (0010, 0011), (0010, 0110), (0011, 0111), (0100, 0101), (0100, 0110), (0101, 0111), (0110, 0111}, E2 = {(1000, 1001), (1000, 1010), (1000, 1100), (1001, 1011), (1010, 1011)}, E3={(0101, 1001), (0101, 1100), (0110, 1010), (0110, 1100), (0111, 1011)}, E4={(0000, 10000), (0001, 1001), (0010, 1010), (0011, 1011), (0100, 1100)}.*



V_1 = {0000, 0001, 0010, 0011, 0100},

V_2 = {0101, 0110, 0111},

V_3 = {1000, 1001, 1010, 1011, 1100},

E_1 = {(0000, 0001), (0000, 0010), (0000, 0100), (0001, 0011), (0001, 0101), (0010, 0011), (0010, 0110), (0011, 0111), (0100, 0101), (0100, 0110), (0101, 0111), (0110, 0111}.

E_2 = {(1000, 1001), (1000, 1010), (1000, 1100), (1001, 1011), (1010, 1011)},

E_3={(0101, 1001), (0101, 1100), (0110, 1010), (0110, 1100), (0111, 1011)},

E_4={(0000, 10000), (0001, 1001), (0010, 1010), (0011, 1011), (0100, 1100)}.

Figure 2: A supercube with 13-nodes

3 Embedding of meshes and tori

The section describes the representation used to solve that embeds a mesh and torus in a supercube with *2*-expansion.

Lemma 1[20] $m_1 \times m_2$ 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[20] 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 $M_{2^2\times 2^2}$ mesh or torus i.e., $d = 2$, $p_1 = 2$, $p_2 = 2$, $n = p_1 + p_2 = 4$. A binary number H of any node of the *4*-dimensional hypercube can be regarded as consisting of two parts: its first *2* bits and its last *2* bits, which we write in the form $H = X_1X_2Y_1Y_2$, where X_i and Y_i are bits *0* or *1*. It is clear from the definition of an *n*-dimensional hypercube (with $n = 4$) that when the last *2* bits are fixed, then the resulting 2^{p_1} nodes form a p_1-dimensional hypercube (with $p_1 = 2$). Whenever we fix the first *2* bits we obtain a p_2-dimensional hypercube. The embedding then becomes clear. Choosing a *2*-bit *BRGC* for the *x* direction and *2*-bit *BRGC* for the *y* direction, the point (x_i, y_i) of the mesh or torus is assigned to the node $X_1X_2Y_1Y_2$ where X_1X_2 is the *2*-bit *BRGC* for dimension of p_1 while Y_1Y_2 is the *2*-bit *BRGC* for dimension of p_2. Herein, we illustrate the result of the mesh or torus in Figure 1-2.

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 a supercube S_N, where $2^n \le N < 2^{n+1}$.

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

The embedding approach that a $M_{m_1\times m_2\times\cdots\times m_d}$ mesh or torus can be embedded in a S_N with *2*-expansion is as follows.

Embedding approach

$M_{m_1\times m_2\times\cdots\times m_d}$ ($m_i = 2^{p_i}$), S_N ($2^n \le N < 2^{n+1}$),

$\forall p_1 + p_2 + \ldots + p_d = \lfloor \log_2 N \rfloor$, $p_1, p_2, \ldots, p_d \ge 1$

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

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

$v = X_{\lfloor \log_2 N \rfloor} X_{\lfloor \log_2 N \rfloor - 1} X_{\lfloor \log_2 N \rfloor - 2} \cdots X_1 X_0$

$v' = X_{\lfloor \log_2 N \rfloor - 1} \cdots X_1 X_0$

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

$0_{\lfloor \log_2 N \rfloor} X_{\lfloor \log_2 N \rfloor - 1} X_{\lfloor \log_2 N \rfloor - 2} \cdots X_1 X_0$ □

Theorem 1 A $M_{2^r\times 2^s}$ *2*-dimensional mesh or torus can be embedded in a supercube S_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 1, lemma 2, and the above embedding 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 a S_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 the above embedding approach. □

This is the best illustrated by an example in Figure 3. That is a $M_{2^2\times 2^1}$ mesh or torus can be embedded in a S_{13} with *2*-expansion.

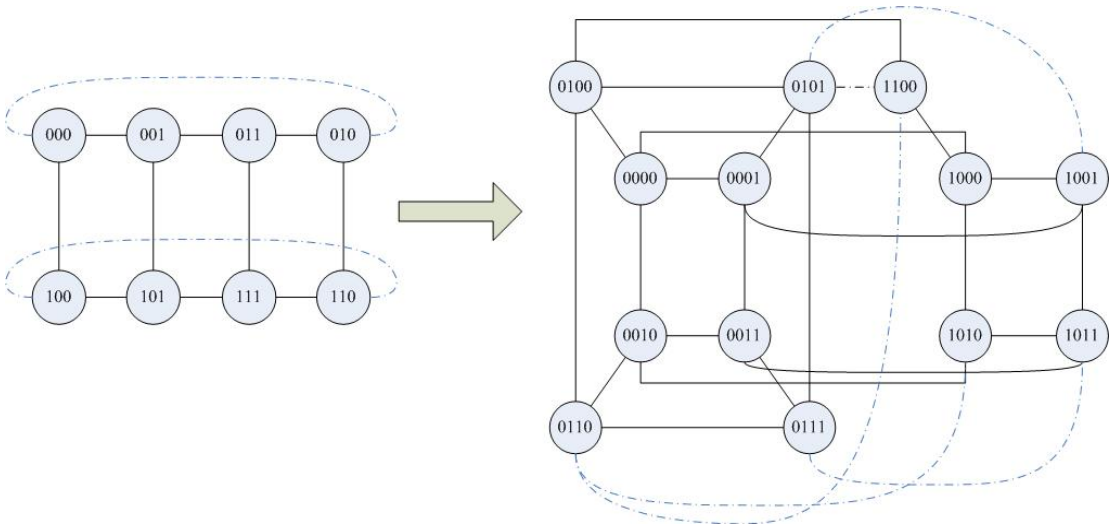


Figure 3: Embedding of a $M_{2^2\times 2^1}$ mesh and torus in a supercube S_{13} with *2*-expansion

4 Fault-Tolerant embedding of meshes and tori with node failures

The session 3 shows that a $M_{m_1\times m_2\times\cdots\times m_d}$ mesh and torus can be embedded in a S_N graph with expansion *2*, load *1*, congestion *1*, and dilation *1*. Hence, in this section, we consider a $M_{m_1\times m_2\times\cdots\times m_d}$ mesh and torus

can be embedded in a S_N with *2*-expansion graph which contains faulty node.

The algorithm design described in this section mainly accorded with the idea the search of bit sequence. We show that each node can be expressed by $(\lfloor \log_2 N \rfloor +1)$-bit binary string $X_{\lfloor \log_2 N \rfloor}X_{\lfloor \log_2 N \rfloor-1}X_{\lfloor \log_2 N \rfloor-2}\cdots X_1X_0$ where X_i is bit *0* or *1*. We search of highest dimension first. If the node is already used or fault, then we retain the most significant bit and change a bit at a time from X_0 to $X_{\lfloor \log_2 N \rfloor-1}$ sequentially until we find the replace node. Therefore, the replacing sequence can be found the replace node. Now, we propose a novel algorithm for embedding a $M_{m_1\times m_2\times\cdots\times m_d}$ mesh and torus in a faulty S_N with *2*-expansion as follows.

Algorithm MTtoS(x)

Input: *x* /*the faulty node*/,
$M_{m_1\times m_2\times\cdots\times m_d}$ ($m_i = 2^{p_i}$),
S_N ($2^n \le N < 2^{n+1}$),
$\forall p_1 + p_2 + \ldots + p_d = \lfloor \log_2 N \rfloor$,
$p_1, p_2, \ldots, p_d \ge 1$

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

1. *i=0*
2. if a node *x* is faulty
3. then
4. {
5. search the node f_1
/* $HD(x, f_1)=1$, $Dim(x, f_1)=\{\lfloor \log_2 N \rfloor\}$*/
6. if f_1 is a exist node and it is free
7. then
8. return(f_1) /*replace *x* with f_1*/
9. *exit()*
10. else
11. while $i < \lfloor \log_2 N \rfloor$ do
12. {
13. search the node f_2
/* $HD(x, f_2)=2$, $Dim(x, f_2)=\{\lfloor \log_2 N \rfloor, i\}$*/
14. if f_2 is a exist node and it is free
15. then
16. return(f_2) /*replace *x* with f_2*/
17. *exit()*
18. }
19. *i=i+1*
20. }
21. return("*Failure*")
22. end

By the algorithm *MTtoS ( )*, the replacing sequence of the faulty node is shown as follows.

$$node0 = 0X_{\lfloor \log_2 N \rfloor-1}X_{\lfloor \log_2 N \rfloor-2}\cdots X_1X_0$$

$node1 = 1X_{\lfloor \log_2 N \rfloor -1} X_{\lfloor \log_2 N \rfloor -2} \cdots X_1 X_0$

$node2 = 1X_{\lfloor \log_2 N \rfloor -1} X_{\lfloor \log_2 N \rfloor -2} \cdots X_1 X'_0$

$node3 = 1X_{\lfloor \log_2 N \rfloor -1} X_{\lfloor \log_2 N \rfloor -2} \cdots X'_1 X_0$

$\vdots$

$node(\lfloor \log_2 N \rfloor +1)$

$= 1X'_{\lfloor \log_2 N \rfloor -1} X_{\lfloor \log_2 N \rfloor -2} \cdots X_1 X_0$

We illustrate an example of embedding a $M_{2^2 \times 2^1}$ mesh or torus in a supercube S_{13} as shown in Figure 3. Furthermore, we illustrate an example of finding a replacing sequence in a *2*-expansion supercube S_{13} with node failures as shown in Figure 4.

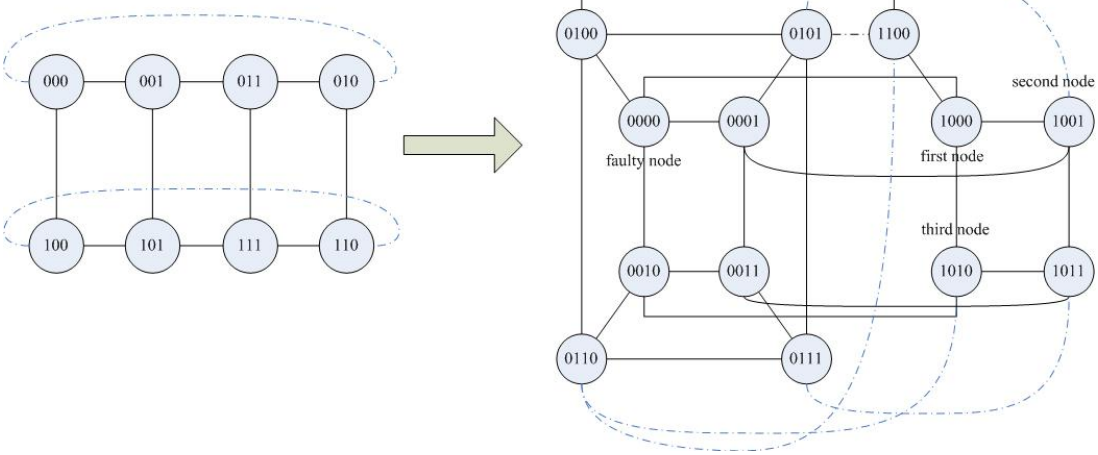


Figure 4: The replacing sequence of embedding a $M_{2^2 \times 2^1}$ Mesh or torus in a faulty supercube S_{13}

By Figure 4, when the faulty node (*0000*) exists, we execute the operations of the *MTtoS( )*. All node of the replacing sequence is listed as {*1000, 1001), 1010, 1100)*} by *MTtoS( )*.

Theorem 3 A $M_{2^r \times 2^s}$ *2*-dimensional mesh or torus can be embedded in a faulty S_N that $r+s = \lfloor \log_2 N \rfloor$ with load *1*, dilation *2*, congestion *1* and expansion *2*.

Proof. By executing the *MTtoS* method and the definition of the partial faulty model, 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 executing the *MTtoS* method in a worst case. Because these nodes and links of replacing sequences are not replicated from the *MTtoS* method, four costs associated with graph embedding are dilation *2*, expansion *2*, load *1* and congestion *1*. □

Theorem 4 A replacing sequence of the *MTtoS* method is including approximate to $(\lfloor \log_2 N \rfloor +1)$ nodes.

Proof. Every node can be represented by a $(\lfloor \log_2 N \rfloor +1)$-bit binary string $X_{\lfloor \log_2 N \rfloor} X_{\lfloor \log_2 N \rfloor -1} X_{\lfloor \log_2 N \rfloor -2} \cdots X_1 X_0$ where $i_p \in \{0,1\}$. First, we change the most significant bit from *0* to *1*. Then, a bit can be changed from X_0 to $X_{\lfloor \log_2 N \rfloor -1}$ sequentially by the *MTtoS* method. Because the supercube may be having a lack of some nodes, a replacing sequence of the *MTtoS* method is including approximate to $(\lfloor \log_2 N \rfloor +1)$ nodes. □

Theorem 5 There are $O(\lfloor \log_2 N \rfloor)$ faults can be tolerated in the faulty supercube with *2*-expansion.

Proof. It is trivial by theorem 4. □

Theorem 6 The result implies that optimal simulation of mesh and torus in a faulty supercube for balancing the processor and communication link loads.

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

5 Conclusions

Supercubes are superior to hypercubes in terms of embedding a mesh and torus under faults. Therefore,

this paper presented techniques to enhance the novel algorithm for fault-tolerant meshes and tori embedded in supercubes with node failures. The paper demonstrate that $O(\lfloor \log_2 N \rfloor)$ faults can be tolerated and the algorithm is optimized mainly for balancing the processor and communication link loads. Also, the methodology is proven and an algorithm is presented to solve them. These existent parallel algorithms on mesh or torus architectures to be easily transformed to or implemented on supercube architectures with load *1*, congestion *1*, dilation *2*, and expansion *2*. 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. Therefore, we can easily port the parallel or distributed algorithms developed for these structuring of mesh and torus to the supercubes.

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

[12] 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.

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

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

[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, Embedding Hamiltonian Cycles, Linear Arrays and Rings in a Faulty Supercube, *International Journal of High Speed Computing* , Vol. 11, 2000, pp. 189-201.

[19] J.-C. Lin and N.-C. Hsien, Reconfiguring Binary Tree Structures in a Faulty Supercube with Unbounded Expansion, *Parallel Computing*, Vol. 28, 2002, pp. 471-483.

[20] J.-C. Lin, Faulty-Avoiding Methods for Mapping Meshes in an IEH, *WSEAS Transactions on Computers*, Vol. 6, 2007, pp. 888-893.

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

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

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

[28] 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.

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

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

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

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

[33] 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.

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

[35] 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.

[36] 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.

[37] D. Wang, On Embedding Hamiltonian Cycles in Crossed Cubes, IEEE Transactions on Parallel and Distributed Systems, Vo1. 9, 2008, pp. 334-346.

[38] 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.

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

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

[41] 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.

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