Tolerating Faults in a Mesh with a Row of Spare Nodes

Jehoshua Bruck Robert Cypher Ching-Tien Ho
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA 95120
{bruck,cypher,ho}@almaden.ibm.com

Abstract

We present an efficient method for tolerating faults in a two-dimensional mesh architecture. Our approach is based on adding spare components (nodes) and extra links (edges) such that the resulting architecture can be reconfigured as a mesh in the presence of faults. We optimize the cost of the fault-tolerant mesh architecture by adding about one row of redundant nodes in addition to a set of k spare nodes (while tolerating up to k node faults) and minimizing the number of links per node. Our results are surprisingly efficient and seem to be practical for small values of k. The degree of the fault-tolerant architecture is $k+5$ for odd k, and $k+6$ for even k. Our results can be generalized to d-dimensional meshes such that the number of spare nodes is less than the length of the shortest axis plus k, and the degree of the fault-tolerant mesh is $(d-1)k+d+3$ when k is odd and $(d-1)k+2d+2$ when k is even.

1 Introduction

1.1 Background

The advent of microprocessor technology and large scale integration at affordable costs have allowed the design and fabrication of parallel machines hosting a large number of processors. As the number of the components in an architecture becomes larger it is essential to consider the issue of computing in the presence of faults.

Many existing parallel machines have a mesh topology. Examples of two-dimensional mesh computers include the MPP (from Goodyear Aerospace), the MP-1 (from MASPAR), VICTOR (from IBM), and DELTA and Paragon (from Intel). The J-Machine, which is under development at MIT, is a three-dimensional mesh. The mesh is also a popular architecture for connecting computing modules on a board or chip. In addition, memory chips are also organized in the form of a two-dimensional mesh [15].

A large amount of research has been devoted to creating fault-tolerant parallel architectures. The techniques used in this research can be divided into two main classes. The first class consists of techniques which *do not* add redundancy to the desired architecture. Instead, these techniques attempt to mask the effects of faults by using the healthy part of the architecture to simulate the entire machine [1, 7, 12, 14]. The hope with this approach is to obtain the same functionality with a reasonable slowdown factor. While this approach yields interesting theoretical results, even a constant factor slowdown in performance can be very significant in practice. Furthermore, this approach requires that some healthy processors simulate several processors. As a result, each simulated processor can have only a fraction of the memory present in a healthy processor.

The second class consists of techniques which *do* add redundancy to the desired architecture. These techniques attempt to isolate the faults, usually by disabling certain links or disallowing certain switch settings, while maintaining the complete desired architecture [2, 3, 4, 5, 6, 8, 9, 13, 15, 16, 17, 18, 19, 20].

1.2 Our Approach

Our approach is based on adding redundancy which includes both spare processors and extra links. We make the following assumptions:

- Both processors and links can fail.
- Faults are total, namely, a faulty processor cannot route or compute.
- Faults are static, namely, a faulty component remains faulty.

0-8186-3200-3/92 $03.00 © 1992 IEEE

- Faults have been identified, for example, by a diagnostic procedure.

Given these assumptions our goal is to create a fault-tolerant architecture that can tolerate up to k (a given parameter) faults with no slowdown in performance. The abstraction of our approach is based on a graph model. In this model a distributed memory parallel computer is viewed as being a graph in which the nodes represent the processors and the edges represent the communication links. A target graph with n nodes is first selected. Then a fault-tolerant graph with $n + m$ nodes is defined with the property that given any set of k or fewer faulty nodes, the remaining graph is guaranteed to contain the target graph as a subgraph. Note that $k \leq m$. This approach guarantees that any algorithm designed for the target graph will run *with no slowdown* in the presence of k or fewer node faults, regardless of their distribution. Hence, minimizing the cost in this model amounts to (i) constructing a fault-tolerant graph with a small number of extra spare nodes, namely minimizing $(m - k)$, and (ii) constructing a fault-tolerant graph with a small degree.

Clearly, there is a trade-off between the number of extra spares and the degree of the fault-tolerant graph. In this paper we will present an approach that addresses this trade-off. We note here that although our results are stated for node faults they can also be used to tolerate edge faults by viewing a node incident with each faulty edge as being faulty.

This graph model of fault tolerance has been used by several other researchers. Hayes [13] has used this model with target graphs of cycles, linear arrays and trees. The work by Wong and Wong [20] and Paoli, Wong and Wong [17] relates to cycles. The recent work by Dutt and Hayes uses trees [8], hypercubes [9] and arbitrary graphs [10] as target graphs. The recent work by us [6] uses de Bruijn and shuffle-exchange networks as target graphs. Also, we have described in [4, 5] a number of constructions of fault-tolerant meshes with minimal numbers of spares, namely with $m = k$. Our approach is based on ideas related to circulant graphs [4] and to diagonal graphs [5]. In particular, our best constructions for fault-tolerant two-dimensional meshes, with k spares, can tolerate k faults and have degree $2k + 2$ when k is odd and $2k + 4$ when k is even [5]. While this result is extremely useful for $k = 1$ (and maybe up to $k = 3$) it is clear that the degree of the fault-tolerant graph is growing too fast for it to be useful in practice.

In this paper we study the possible trade-off between minimizing the number of extra spares and minimizing the degree of the fault-tolerant graph. While in [4, 5] we assumed that the cost of a processor is very high and we designed a fault-tolerant mesh architecture with an exact number of spares, in this paper we present constructions that require more spare processors but have the advantage of requiring fewer links per node. Specifically, the constructions of fault-tolerant graphs given here have degree $k + 5$ when k is odd and $k + 6$ when k is even and use an extra row of spares in addition to the k spares. Hence, we provide a technique for trading between redundancy in number of spare nodes and redundancy in the number of links per node. The construction presented here is particularly well-suited to large mesh architectures, in which case the extra row of components represents a small fraction of the total number of components, and a relatively large number of faults can be expected. Whenever k is 4 or greater, the construction given here has a smaller degree than that presented in [5].

Our constructions are based on a combination of two ideas: (i) Labeling the mesh nodes such that it becomes a subgraph of a circulant graph with offsets that are next to each other, and (ii) creating a linear ordering on the nodes and utilizing the concept of diagonal graphs, presented in [5]. These two ideas help in reducing the degree of the fault-tolerant graph. It turns out, that a combination of these two ideas can be accomplished if we add an extra row of spares.

In the next section we present the concepts of circulant and diagonal graphs and describe their application in the creation of fault-tolerant meshes. The constructions of fault-tolerant two-dimensional meshes and the renaming (reconfiguration) algorithm for these constructions are given in Section 3. In Section 4 we describe how our approach can be generalized to d-dimensional meshes. Finally, conclusions are presented in Section 5.

2 Circulant and Diagonal Graphs

A useful concept for creating fault-tolerant graphs is the graph known as "circulant graph" [11], which is defined below.

Definition: Let n be a positive integer and let S be a set of integers in the range 1 through $n - 1$. The *n-node circulant graph with connection set S*, denoted $C_{n,S}$, consists of n nodes. Each node in $C_{n,S}$ has a unique label in the range 0 through $n - 1$. Each node i is connected to all nodes of the form $(i \pm s) \bmod n$ where $s \in S$. The values in the connection set S will be

referred to as "jumps" or "offsets". A simple example of a circulant graph is a cycle, where there is only one offset and the value of that offset is 1.

Definition: Let k be a nonnegative integer and let $G = (V, E)$ be a graph. We say that the graph $G' = (V', E')$ is (k, G)-tolerant if the subgraph of G' induced by any set of $|V'| - k$ nodes contains G as a subgraph.

Definition: Let S be a set of integers and let k be a nonnegative integer. The *expansion of S by k*, denoted $expand(S, k)$, is the set T where

$$T = \bigcup_{s \in S} \{s, s+1, \ldots, s+k\}.$$

Note that $|expand(S, k)| \leq (k+1)|S|$.

The following theorem is an immediate consequence of a result proven by Dutt and Hayes [9].

Theorem 2.1 [9] *Let n be a positive integer, let S be a set of integers in the range 1 through $n-1$, let k be a nonnegative integer, and let $T = expand(S, k)$. The circulant graph $C_{n+k,T}$ is $(k, C_{n,S})$-tolerant.*

Theorem 2.1 gives a general technique for creating a fault-tolerant graph when the target graph is circulant. In fact, in [4] we have presented a few constructions based on this idea. We mention here two of these constructions.

Theorem 2.2 *Let M be an $r \times c$ mesh with $n = rc$ nodes. Then*

1. *Let $S = \{1, c\}$ and let $T = expand(S, k)$. The circulant graph $C_{n+k,T}$ is (k, M)-tolerant and has degree at most $4k + 4$.*
2. *Let $S = \{c, c+1\}$ and let $T = expand(S, k)$. The circulant graph $C_{n+k,T}$ is (k, M)-tolerant and has degree at most $2k + 4$.*

Notice that the first construction in Theorem 2.2 follows from the row-major ordering of a mesh and results in a large degree $(4k + 4)$, since it consists of 2 offsets (1 and c) that are far apart. The trick in the second construction is in having the offsets be two consecutive integers (c and $c+1$), the *expand* operation on the offsets is then shared by both offsets and the result is a degree $2k + 4$ fault-tolerant graph. We get the consecutive offsets by what we call antidiagonal-major ordering of the nodes [4].

Another important class of target graphs consists of what we call "diagonal graphs" [5]. The definition of diagonal graphs and a general technique for adding fault-tolerance to diagonal graphs are given next.

Definition: Let n be a positive integer and let S be a set of integers in the range 1 through $n-1$. The *n-node diagonal graph with connection set S*, denoted $D_{n,S}$, consists of n nodes. Each node in $D_{n,S}$ has a unique label in the range 0 through $n-1$. Each node i is connected to all nodes of the form $i \pm s$ where $s \in S$. Thus diagonal graphs are similar to circulant graphs, except they do not have the "wraparound" connections from high numbered nodes to low numbered nodes. The name "diagonal graph" refers to the structure of the adjacency matrix of such a graph.

Given the target graph $D_{n,S}$ (with the restriction that $S \subseteq \{1, 2, \ldots, \lceil n/3 \rceil\}$), we will use the circulant graph $C_{n+k,T}$, where $T = expand(S, \lfloor k/2 \rfloor)$, as the fault-tolerant graph. The idea is similar to the technique for adding fault-tolerance to circulant graphs which was described above. Recall that given the circulant target graph $C_{n,S}$, the fault-tolerant graph (which tolerates k faults) has the connection set $T = expand(S, k)$. The reason that we have to expand S by k is that an edge in the target graph may have to "jump over" as many as k faults in the fault-tolerant graph. In contrast, given the diagonal target graph $D_{n,S}$, the fault-tolerant graph requires only the connection set $T = expand(S, \lfloor k/2 \rfloor)$. The reason that we can expand S by $\lfloor k/2 \rfloor$ rather than by k is that the lowest and highest numbered nodes in $D_{n,S}$ have smaller degree than the other nodes in $D_{n,S}$. Thus if the fault-tolerant graph has a cluster of faults which are near one another (and thus could require an edge to jump over a large number of faults), we can choose to map the lowest and highest numbered nodes in $D_{n,S}$ to the healthy nodes near that cluster of faults. Using this mapping none of the edges has to jump over the cluster of faults, and the expansion by $\lfloor k/2 \rfloor$ is sufficient. This argument is formalized below.

Theorem 2.3 [5] *Let n be a positive integer, let $y = \lceil n/3 \rceil$, let S be a set of integers in the range 1 through y, let k be a positive integer, and let $T = expand(S, \lfloor k/2 \rfloor)$. The circulant graph $C_{n+k,T}$ is $(k, D_{n,S})$-tolerant.*

Given an $r \times c$ mesh, say M, with $n = rc$ nodes. Label the nodes by row-major ordering. It is easy to see that M is a subgraph of $D_{n,S}$ where $S = \{1, c\}$. Hence, using Theorem 2.3 we proved in [5] that

Theorem 2.4 *Let r and c be integers where $r \geq c$ and $r \geq 3$. Let M be an $r \times c$ mesh with $n = rc$*

nodes. Let $S = \{1, c\}$ and let $T = expand(S, \lfloor k/2 \rfloor)$. The circulant graph $C_{n+k,T}$ is (k, M)-tolerant and has degree at most $2k+4$ if k is even, and at most $2k+2$ if k is odd.

Hence, both construction 2 in Theorem 2.2 and the diagonal graph idea give a reduction in about a factor of two (for large k) in the degree of the fault-tolerant mesh compared to the first construction based on row-major ordering presented in Theorem 2.2. The challenge here is to combine between the two techniques to reduce the degree by a factor of 4 (for large k). We are able to achieve that at the cost of adding more spares than the minimal number needed. Notice that all the constructions presented above have minimal numbers of spares, namely k.

3 The Construction for 2-Dimensional Meshes

In this section we present our main result which is a construction of a fault-tolerant two-dimensional mesh which has a degree smaller than the best known constructions by a factor of 2 (for large k). The key to our result is a combination of the two ideas presented in Section 2: the contiguous offsets and the diagonal graph.

We are interested in this section in a two-dimensional mesh, denoted by M, with $n = rc$ nodes, where r specifies the number of rows and c specifies the number of columns. Each node is labeled with a unique label of the form (x_1, x_2) where $0 \leq x_1 < r$ and $0 \leq x_2 < c$. Each node (x_1, x_2) is connected to at most 4 nodes of the form $(x_1 \pm 1, x_2 \pm 1)$. We first present the construction of a (k, M)-tolerant graph and then we describe an efficient algorithm for finding the good mesh which is present in the (k, M)-tolerant graph after it has suffered k node faults. We will call this algorithm the *renaming algorithm*.

3.1 Construction of Fault-Tolerant Meshes

We first describe the construction of the general case. We then present, as an example, the special case of a 7×6 mesh and consider the case of a single fault.

Main Construction: Let M be an $r \times c$ mesh. M consists of $n = rc$ nodes. Let $\hat{M}_k$ be the corresponding (k, M)-tolerant graph. $\hat{M}_k$ consists of $n+m$ nodes that are numbered from 0 to $n+m-1$. There are two cases for the value m. When c is odd $m = k + c - 1$ and when c is even $m = k + c - 2$. The graph $\hat{M}_k$ is a circulant graph. For the definition of the edges there are two cases.

- *Case 1:* k is odd. Every node has degree $k + 5$. The edges are defined by the following set of offsets:
$$\{c + j \mid -1 \leq j \leq \frac{k+1}{2}\}.$$
- *Case 2:* k is even. Every node has degree $k + 6$. The edges are defined by the following set of offsets:
$$\{c + j \mid -1 \leq j \leq \frac{k+2}{2}\}.$$

The correctness of the above construction follows from Theorem 2.3 and is formalized as follows.

Theorem 3.1 *Let M be an $r \times c$ mesh with $n = rc$ nodes. Let $S = \{c-1, c, c+1\}$. Let $T = expand(S, \lfloor k/2 \rfloor)$. Let $\hat{M}_k$ be the circulant graph $C_{n+m,T}$ with $m = k + c - 1$ when c is odd and $m = k + c - 2$ when c is even. Then $\hat{M}_k$ is a (k, M)-tolerant graph.*

Proof: Let $\tilde{M}$ be the diagonal graph with $n+m$ nodes and the set of offsets $S = \{c-1, c, c+1\}$. Namely, $\tilde{M}$ is the diagonal graph that corresponds to the circulant graph $\hat{M}_0$. The key in the proof is to show that the diagonal graph $\tilde{M}$ contains the $r \times c$ mesh M as a subgraph and then apply Theorem 2.3 to obtain the fault-tolerant graph.

We will define the mapping between the nodes of the mesh M and the nodes of the diagonal graph $\tilde{M}$ as follows: Each node (i, j) in M corresponds to node

$$\phi(i, j) = (i + ((j+1) \bmod 2))c + j - 1 \qquad (1)$$

in $\tilde{M}$.

We need to show that the mapping $\phi(i, j)$ defined in Eq. (1) is correct. First it can be shown that $0 \leq \phi(i, j) \leq rc + c - 2$ when c is odd and $0 \leq \phi(i, j) \leq rc + c - 3$ when c is even, where $0 \leq i \leq r - 1$ and $0 \leq j \leq c - 1$.

Second we need to show that given the mapping $\phi(i, j)$, the edges of M exist in $\tilde{M}$. This is true because

$$\begin{aligned}
|\phi(i,j) - \phi(i+1,j)| &= c \in S, \\
|\phi(i,j) - \phi(i-1,j)| &= c \in S, \\
|\phi(i,j) - \phi(i,j+1)| &= \text{either } c+1 \text{ or } c-1 \in S, \\
\text{and } |\phi(i,j) - \phi(i,j-1)| &= \text{either } c+1 \text{ or } c-1 \in S,
\end{aligned}$$

for all valid values of (i, j). We can also prove that we never use "wraparound" edges when we embed M using $\phi(i, j)$. Hence, M is a subgraph of the diagonal graph $\tilde{M}$.

Now, we can apply Theorem 2.3 and notice that the circulant graph $C_{n+m,T}$ with

$$m = \begin{cases} k + c - 1, & \text{when } c \text{ is odd,} \\ k + c - 2, & \text{when } c \text{ is even,} \end{cases}$$

is a $(k, \tilde{M})$-tolerant graph. □

As an example, consider the case of a 7×6 mesh. Figure 1 presents the labeling of the nodes in the 7×6 mesh given by Eq. 1. As can be seen from the figure, the labeling results in a traversal of nodes in a "see-saw" manner. Figure 2 shows the graph $\hat{M}_0$, which is a circulant graph with offsets $\{5, 6, 7\}$ and contains the 7×6 mesh M as a subgraph. For clarity, some image of nodes, which are represented by empty circles, are added for the wraparound connections. Note that the labeling of nodes of the mesh contained in $\hat{M}_0$ is the same as that in Figure 1. Figure 3 presents the $(1, M)$-tolerant mesh with degree 6 and 5 extra nodes. Note that a new node numbered 46 is added in the lower right corner from Figure 2.

Next, we present a systematic way to label the remaining graph and obtain the mesh.

3.2 The Renaming Algorithm

We present an efficient algorithm which, given k faults in $\hat{M}_k$, finds a healthy $r \times c$ mesh. The algorithm defines the healthy mesh by assigning new labels to the nodes. The algorithm is a particular application of the more general labeling algorithm for diagonal graphs, presented in [5], where a proof of correctness can be found.

The Renaming Algorithm: We are given a set of k nodes in $\hat{M}_k$ that are faulty. If there are x faulty nodes where $x < k$, we arbitrarily select any $k - x$ healthy nodes and consider them to be faulty. Recall that the nodes in the graph are numbered 0 through $n + m - 1$. These nodes will be viewed as being ordered cyclically, with nodes $n + m - 1$ and 0 being adjacent. Thus, when the nodes are traversed in ascending order, node 0 follows node $n + m - 1$ and when they are traversed in descending order, node $n + m - 1$ follows node 0. The renaming algorithm consists of three steps.

- The first step uses two counters, one to count faulty nodes and one to count non-faulty nodes. The following routine is performed for all values of i where $0 \leq i \leq n+m-1$. First, both counters are set to 0. Then the nodes are visited in a descending order, starting with node i. As each node is visited, the appropriate counter is incremented. That is, if the visited node is faulty, the counter for faulty nodes is incremented, and if the visited node is non-faulty, the counter for non-faulty nodes is incremented. Thus, node i is the first node to be visited, and the appropriate counter is set according whether or not node i is faulty. The counter for non-faulty nodes is checked after it is incremented. If this counter is equal to $c + 2$, the process of visiting the nodes in descending order is terminated and the counter for faulty nodes is checked. If the counter for faulty nodes is greater than $k/2$, node i is designated as being "marked", while if it is less than or equal to $k/2$, node i is designated as being "unmarked". The non-faulty marked nodes are ones which have a large number of faulty nodes preceding them, and as a result they must be assigned to the first rows of the non-faulty mesh.

- The second step figures out which non-faulty node should play the role of node 0 in the non-faulty mesh. The second step uses a single counter and it consists of two phases. Phase 1 begins by setting the counter to 0. Then the nodes are visited in descending order, starting with any arbitrarily selected node. As each node is visited, the node is checked to see whether or not it is faulty and whether or not it is marked. If the node is non-faulty and unmarked, the counter is incremented. If the node is non-faulty and marked, the counter is reset to 0. If the node is faulty, the counter is left unchanged. Then the counter is checked and Phase 1 is terminated if the counter is greater than or equal to $n/3$. We will call the node that is being visited when the counter reaches $\lceil n/3 \rceil$ node a. Phase 2 then visits the nodes in ascending order beginning with node a. It terminates when it encounters a non-faulty node which is marked. This non-faulty marked node will be called node b.

- The third step then assigns numbers to the non-faulty nodes. The nodes are visited in ascending order, starting with node b, and the non-faulty nodes are assigned the values $0, 1, \ldots n + m - 1$ in order. Thus node b is assigned 0, the next non-faulty node that is visited is assigned 1, and the last non-faulty node that is visited is assigned

$n + m - 1$. These numbers correspond to the numbering of $\hat{M}_0$. The correspondence of a label ℓ to the coordinates of the mesh, say (i, j), is as follows:

$$i = \frac{(\ell + 1 - j)}{c} - ((j + 1) \bmod 2),$$

$$j = (\ell + 1) \bmod c.$$

Clearly, nodes with label ℓ for which $i < 0$ or $i \geq r$ are omitted.

Notice that in the case of a single fault the above algorithm will result in a new labeling that starts immediately after the fault. For example, consider the 7×6 $(1,M)$-tolerant graph in Figure 3 and assume that node 18 is faulty. Figure 4 presents the new labeling of the mesh. Each row in the configured mesh is shown by a thick line (possibly with wraparound).

4 Generalization to d-dimensional Meshes

In this section, we generalize the fault-tolerant technique described before from two-dimensional meshes to d-dimensional meshes, while preserving the number of spare nodes to be approximately the length of the shortest axis plus k.

4.1 Construction of d-dimensional Fault-Tolerant Meshes

Let M be a d-dimensional mesh of form $n_1 \times n_2 \times \cdots \times n_d$, where it is assumed that $n_1 \leq n_2 \leq \cdots \leq n_d$ and $n_d \geq 3$. Thus, M consists of $n = \prod_{i=1}^{d} n_i$ nodes. Let $\hat{M}_k$ be the corresponding (k, M)-tolerant graph. The graph $\hat{M}_k$ consists of $n+m$ nodes that are numbered from 0 through $n + m - 1$. There are two cases for the value m. When n_1 is odd $m = k + n_1 - 1$ and when n_1 is even $m = k + n_1 - 2$. The graph $\hat{M}_k$ is a circulant graph. For the definition of the edges there are two cases.

- *Case 1:* k is odd. Every node has degree $(d - 1)k + d + 3$. The edges are defined by the union of the following sets of offsets:

$$\{n_1 + j \mid -1 \leq j \leq \frac{k+1}{2}\},$$

$$\{n_1 n_2 + j \mid 0 \leq j \leq \frac{k-1}{2}\},$$

$$\{n_1 n_2 n_3 + j \mid 0 \leq j \leq \frac{k-1}{2}\}, \cdots,$$

and up to

$$\{n_1 n_2 \cdots n_{d-1} + j \mid 0 \leq j \leq \frac{k-1}{2}\}.$$

- *Case 2:* k is even. Every node has degree $(d - 1)k + 2d + 2$. The edges are defined by the union of the following sets of offsets:

$$\{n_1 + j \mid -1 \leq j \leq \frac{k+2}{2}\},$$

$$\{n_1 n_2 + j \mid 0 \leq j \leq \frac{k}{2}\},$$

$$\{n_1 n_2 n_3 + j \mid 0 \leq j \leq \frac{k}{2}\}, \cdots,$$

and up to

$$\{n_1 n_2 \cdots n_{d-1} + j \mid 0 \leq j \leq \frac{k}{2}\}.$$

We now give a few examples for the sets of offsets just defined. When $d = 2$, it yields the same definition for the two-dimensional meshes given before. When $d = 3$ and $k = 1$, the set of offsets is $\{n_1 - 1, n_1, n_1 + 1, n_1 n_2\}$. When $d = 3$ and $k = 3$, the set of offsets is $\{n_1 - 1, n_1, n_1 + 1, n_1 + 2, n_1 n_2, n_1 n_2 + 1\}$, which has the same set of offsets as for the case $d = 3$ and $k = 2$. The degree of a fault-tolerant 3-dimensional mesh that can tolerate k faults is $2k+6$ when k is odd and is $2k + 8$ when k is even.

It can be shown that the $\hat{M}_k$ as defined is a (k, M)-tolerant graph with M being a d-dimensional mesh (as defined before).

The renaming algorithm for locating the healthy d-dimensional mesh M in the fault-tolerant mesh $\hat{M}_k$ after up to k node faults have occurred is similar to the renaming algorithm for locating the healthy two-dimensional mesh described before. Details are omitted due to space limitations.

5 Concluding Remarks

We have presented constructions of fault-tolerant mesh architectures. Our approach is based on adding spare components (nodes) and extra links (edges) such that the resulting architecture can be reconfigured as a mesh in the presence of k faults. The degree of the fault-tolerant mesh is $k+5$ for odd k, and $k+6$ for even k. The number of spare nodes is at most $k + c - 1$. Those are the best known constructions in terms of the degree of the fault-tolerant graph for $k > 3$. Note that in the case that the fault-tolerant graph has an exact number of spares, namely k, it is easy to prove

that the degree of the fault-tolerant graph is at least $\max\{4, k+2\}$. Hence, using our techniques we got close to the lower bound related to exact number of spares by paying an extra row of redundancy. It would be interesting to close the gap by proving better lower bounds or by finding better constructions.

References

[1] F. Annexstein, *Fault Tolerance in Hypercube-Derivative Networks*, Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 179–188, 1989.

[2] V. Balasubramanian and P. Banerjee, *A Fault Tolerant Massively Parallel Processing Architecture*, J. of Parallel and Distributed Computing, vol. 4, pp. 363–383, 1987.

[3] K. E. Batcher, *Design of a Massively Parallel Processor*, IEEE Trans. on Computers, vol. C-29, no. 9, pp. 836–840, September 1980.

[4] J. Bruck, R. Cypher and C.-T. Ho, *Fault-Tolerant Meshes with Minimal Numbers of Spares*, Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing, pp. 288–295, Dallas TX, December 1991.

[5] J. Bruck, R. Cypher and C.-T. Ho, *Fault-Tolerant Meshes and Hypercubes with Minimal Numbers of Spares*, IBM Research Report, RJ 8211, July 1991.

[6] J. Bruck, R. Cypher and C.-T. Ho, *Fault-Tolerant de Bruijn and Shuffle-Exchange Networks*, IBM Research Report, RJ 8547, December 1991. To appear in the Proceedings of the 1992 International Conference on Parallel Processing.

[7] J. Bruck, R. Cypher and D. Soroker, *Running Algorithms Efficiently on Faulty Hypercubes*, Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 37–44, 1990.

[8] S. Dutt and J. P. Hayes, *On Designing and Reconfiguring k-Fault-Tolerant Tree Architectures*, IEEE Trans. on Computers, vol. C-39, no. 4, pp. 490–503, April 1990.

[9] S. Dutt and J. P. Hayes, *Designing Fault-Tolerant Systems Using Automorphisms*, Journal of Parallel and Distributed Computing, vol. 12, pp. 249–268, 1991.

[10] S. Dutt and J. P. Hayes, *Some Practical Issues in the Design of Fault-Tolerant Multiprocessors*, Proceedings of the 21st International Symposium on Fault-Tolerant Computing, pp. 292–299, June 1991.

[11] B. Elspas and J. Turner, *Graphs with Circulant Adjacency Matrices*, J. of Combinatorial Theory, no. 9, pp. 297–307, 1970.

[12] J. Hastad, F. T. Leighton and M. Newman, *Fast Computations using Faulty Hypercubes*, Proceedings of 21st Annual ACM Symposium on Theory of Computing, pp. 251–284, 1989.

[13] J. P. Hayes, *A Graph Model for Fault-Tolerant Computing Systems*, IEEE Trans. on Computers, vol. C-25, no. 9, pp. 875–884, September 1976.

[14] C. Kaklamanis, A. R. Karlin, F. T. Leighton, V. Milenkovic, P. Raghavan, S. Rao, C. Thomborson and A. Tsantilas, *Asymptotically Tight Bounds for Computing with Faulty Arrays of Processors*, Proc. of 31st Annual IEEE Symp. on Foundations of Computer Science, pp. 285–296, October 1990.

[15] S.-Y. Kuo and W. K. Fuchs, *Efficient Spare Allocation for Reconfigurable Arrays*, IEEE Design and Test, pp. 24–31, February 1987.

[16] F. T. Leighton and C. E. Leiserson, *Wafer Scale Integration of Systolic Arrays*, IEEE Trans. on Computers, vol. C-34, no. 5, pp. 448–461, May 1985.

[17] M. Paoli, W. W. Wong and C. K. Wong, *Minimum k-Hamiltonian Graphs, II*, J. of Graph Theory, Vol. 10, pp. 79–95, 1986.

[18] A. L. Rosenberg, *The Diogenes Approach to Testable Fault-Tolerant VLSI Processor Arrays*, IEEE Trans. on Computers, Vol. C-32, no. 10, pp. 902–910, October 1983.

[19] V. P. Roychowdhury, J. Bruck and T. Kailath, *Efficient Algorithms for Reconfiguration in VLSI/WSI Arrays*, IEEE Trans. on Computers, vol. C-39, no. 4, pp. 480–489, April 1990.

[20] W. W. Wong and C. K. Wong, *Minimum k-Hamiltonian Graphs*, J. of Graph Theory, Vol. 8, pp. 155-165, 1984.

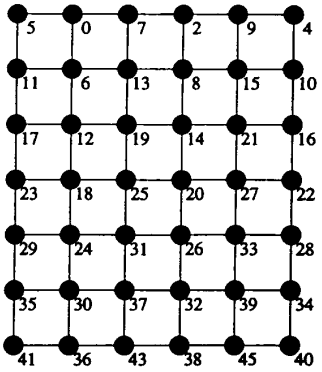

Figure 1: A labeling of a 7×6 mesh.

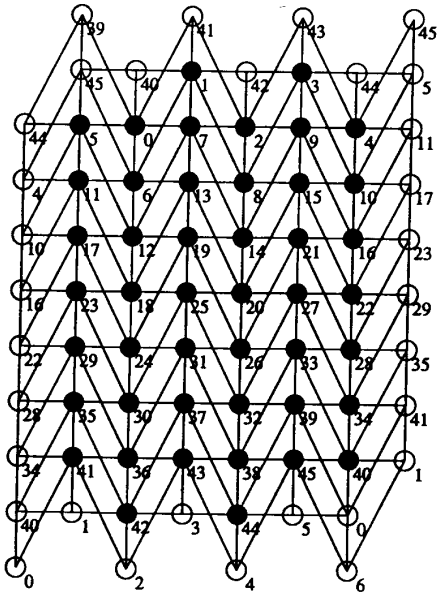

Figure 2: The circulant graph $\hat{M}_0$.

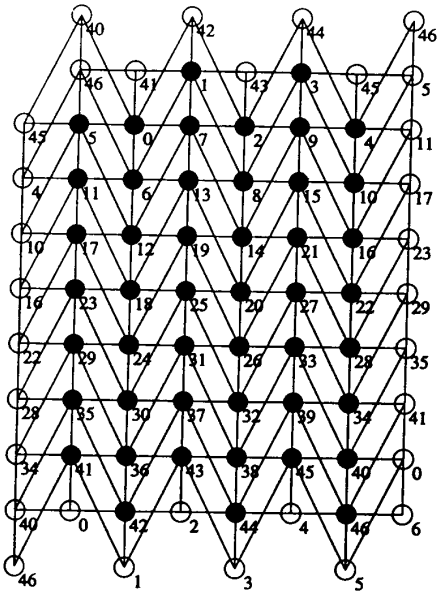

Figure 3: The $(1, M)$-tolerant graph.

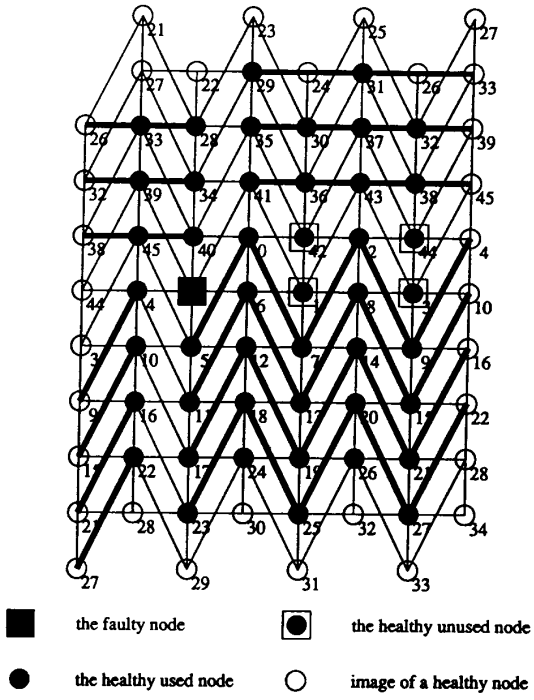

Figure 4: The reconfigured mesh.