On Fault Tolerance of Two-Dimensional Mesh Networks

Soumen Maity[1], Amiya Nayak[2], and S. Ramsundar[3]

[1] Department of Mathematics, Indian Institute of Technology Guwahati
Guwahati 781 039, Assam, India
soumen@iitg.ernet.in
[2] School of Information Technology and Engineering (SITE)
University of Ottawa, 800 King Edward Avenue
Ottawa, Ontario, K1N 6N5, Canada
anayak@site.uottawa.ca
[3] Computer Science & Engineering, Indian Institute of Technology Guwahati
Guwahati 781 039, Assam, India
sundark@iitg.ernet.in

Abstract. The catastrophic fault pattern is a pattern of faults occurring at strategic locations that may render a system unusable regardless of its component redundancy and of its reconfiguration capabilities. In this paper, we characterize catastrophic fault patterns in mesh networks when the links are bidirectional or unidirectional. We determine the minimum number of faults required for a fault pattern to be catastrophic. We consider the problem of testing whether a set of faulty processors is catastrophic. In addition, when a fault pattern is not catastrophic we consider the problem of finding *optimal* reconfiguration strategies, where optimality is with respect to either the number of processing elements in the reconfigured network (the reconfiguration is optimal if such a number is maximized) or the number of bypass links to activate in order to reconfigure the array (the reconfiguration is optimal if such a number is minimized). The problem of finding a reconfiguration strategy that is optimal with respect to the size of the reconfigured network is NP-complete, when the links are bidirectional, while it can be solved in polynomial time, when the links are unidirectional. Considering optimality with respect to the number of bypass links to activate, we provide algorithms which efficiently find an optimal reconfiguration.

1 Introduction

Mesh architectures consist of a large number of identical and elementary processing elements locally connected in a regular fashion. Each element receives data from its neighbors, computes and sends the results again to its neighbors. Few particular elements located at the extremes of the systems (these extremes depend on the particular system) are allowed to communicate with the external world. In this paper, we will focus on mesh architectures.

S. Chaudhuri et al. (Eds.): ICDCN 2006, LNCS 4308, pp. 442–453, 2006.

Fault tolerant techniques are very important to mesh architectures [2]. Here we assume that only processors can fail. Indeed, since the number of processing elements is very large, the probability that a set of processing elements becomes faulty is fairly high. Without the provision of fault-tolerance capabilities, the yield of such an architecture would be so poor that it would be unacceptable. Thus, fault-tolerant mechanisms must be provided in order to avoid faulty processing elements taking part in the computation. A widely used technique to achieve fault tolerance consists of providing redundancy to the desired architecture [1,2]. In these systems the redundancy consists of additional processing elements, called spares, and additional connections, called bypass links. Bypass links are links that connect each processor with another processor at a fixed distance greater than 1. The redundant processing elements are used to replace any faulty processing element; the redundant links are used to bypass the faulty processing elements and reach others. The effectiveness of using redundancy to increase fault tolerance clearly depends on both the amount of redundancy and the reconfiguration capability of the system. It does however depend also on the distribution of faults in the system. There are sets of faulty processing elements for which no reconfiguration strategy is possible. Such sets are called catastrophic fault patterns (CFPs). From a network perspective, such fault patterns can cause network disconnection.

If we have to reconfigure a system when a fault pattern occurs, it is necessary to know if the fault pattern is catastrophic or not. Therefore it is important to study the properties of catastrophic fault patterns. Till today, the characterization of CFPs is known for linear arrays with the following results. The characterization has been used to obtain efficient testing algorithms both for unidirectional and bidirectional cases [13,14] with order of magnitude improvement over [3,4]. Efficient techniques has been obtained for constructing CFPs [12]. Using random walk as a tool, a closed form solution for the number of CFPs for uni- and bidirectional links has been provided in [8,10]. The knowledge of this number enables us to estimate the probability that the system operates correctly [11]. Recently, Maity, Nayak and Roy [9] characterize catastrophic fault patterns for two-dimensional arrays.

In this paper we completely characterize CFPs for mesh networks. We determine the minimum number of faults required for a fault pattern to be catastrophic. From a practical viewpoint, above result allow to prove some answers to the question about the guaranteed level of fault tolerance of a design. Guaranteed fault tolerance indicates positive answer to the question as: will the system withstand up to k faults always regardless of how and where they occur? We analyze catastrophic sets having the minimal number of faults. The paper also describes algorithm for testing whether a set of faults is catastrophic or not. In addition, when a fault pattern is not catastrophic, we consider the problem of finding optimal reconfiguration strategies for both unidirectional and bidirectional networks. Where the optimality is with respect to the number of processors in the reconfigured network or with respect to the number of bypass links. The

reconfiguration is optimal if number of processing elements is maximized in the former case, while the number of bypass links are to be minimum in the latter case.

2 Preliminaries

In this paper, we will focus on *mesh network*s. The basic components of such a network are the processing elements(PEs) indicated by circles in Figure 1. There are two kinds of links : *regular* and *bypass*. Regular links connect neighboring (either horizontal or vertical) PEs while bypass links connect non-neighbors. The bypass links are used strictly for reconfiguration purposes when a fault is detected, otherwise they are considered to be the redundant links. We now introduce the following definitions:

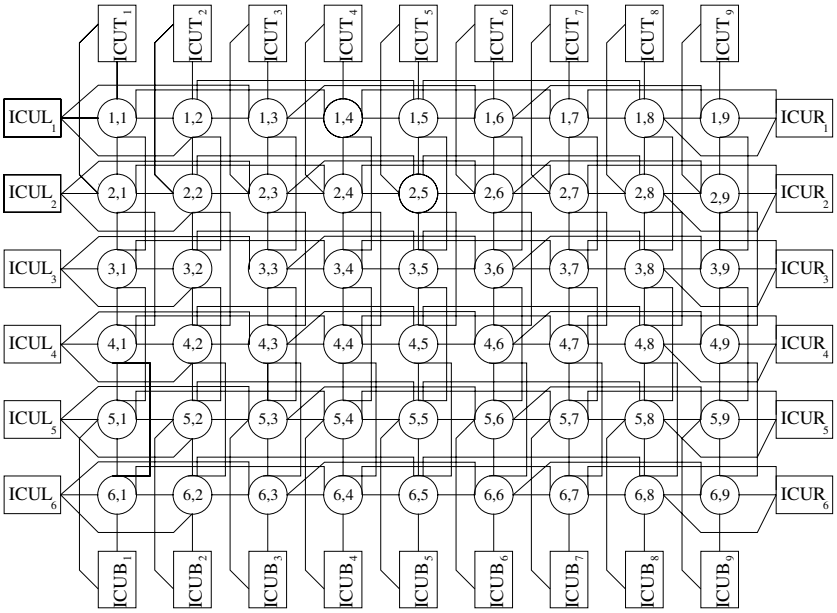

Fig. 1. Mesh network of 54 PEs

Definition 1. A mesh network $\mathcal{M} = (V, E)$ consists of a set V of PEs and a set E of links (where a link joins a pair of distinct PEs) satisfying the conditions listed below.

V is the union of five disjoint sets: the set $ICUL$=$\{ICUL_1, ICUL_2, \ldots, ICUL_{N_1}\}$ of left interface control units, the set $ICUR = \{ICUR_1,\ ICUR_2, \ldots,\ ICUR_{N_1}\}$ of right interface control units, the set $ICUT = \{ICUT_1,\ ICUT_2, \ldots,\ ICUT_{N_2}\}$ of top interface control units, the set $ICUB = \{ICUB_1,\ ICUB_2, \ldots,\ ICUB_{N_2}\}$

of bottom interface control units, and a two-dimensional array $A = \{p_{ij} : 1 \leq i \leq N_1,\ 1 \leq j \leq N_2\}$ of PEs. We sometimes refer to the processing element p_{ij} as $(i,\ j)$.

E consists of the links obtained as follows. Fix integers $1 = g_1 < g_2 < \ldots < g_k \leq N_2 - 1$ and $1 = v_1 < v_2 < \ldots < v_l \leq N_1 - 1$. Join p_{ij} to $p_{i'j'}$ by a link if and only if (i) $i = i'$ and $j' - j$ is one of $g_1,\ g_2, \ldots,\ g_k$ or (ii) $j = j'$ and $i' - i$ is one of $v_1,\ v_2, \ldots,\ v_l$. Also join $ICUL_i$ to $p_{i1},\ p_{i2}, \ldots,\ p_{ig_k}$ and join $p_{i,N_2-g_k+1},\ p_{i,N_2-g_k+2}, \ldots,\ p_{iN_2}$ to $ICUR_i$ by links, for $i = 1,\ 2, \ldots,\ N_1$. Similarly join $ICUT_j$ to $p_{1j},\ p_{2j}, \ldots,\ p_{v_l j}$ and join $p_{N_1-v_l+1,j},\ p_{N_1-v_l+2,j}, \ldots,\ p_{N_1 j}$ to $ICUB_j$ by links, for $j = 1,\ 2, \ldots,\ N_2$.

We assume that $N_2 > g_k$ and $N_1 > v_l$. We also assume N_1 and N_2 are multiple of v_l and g_k respectively.

Definition 2. We refer to $G = (g_1,\ g_2, \ldots,\ g_k \mid v_1,\ v_2, \ldots,\ v_l)$ as the *link redundancy* of $\mathcal{M}$. We call $g_1,\ g_2, \ldots,\ g_k$ the *horizontal link redundancies* of $\mathcal{M}$ and $v_1,\ v_2, \ldots,\ v_l$ the *vertical link redundancies* of $\mathcal{M}$.

Figure 1 shows a mesh network with $N_1 = 6$, $N_2 = 9$ and $G = (1,\ 3 \mid 1,\ 2)$. A link joining two PEs of the type p_{ij} and $p_{i,j+1}$ is called a *horizontal direct link* and a link joining two PEs of the type p_{ij} and $p_{i+1,j}$ is called a *vertical direct link*. Direct links are also called *regular links*. Links joining p_{ij} and $p_{i,\ j+g}$ with $g > 1$ are called *horizontal bypass links* and links joining p_{ij} and $p_{i+v,\ j}$ with $v > 1$ are called *vertical bypass links*.

The *length* of the horizontal bypass link joining p_{ij} to $p_{i,\ j+g}$ is g and the *length* of the vertical bypass link joining p_{ij} to $p_{i+v,\ j}$ is v.

Note that no links exist in the network $\mathcal{M}$ except the ones specified by G as in Definition 1. It is assumed that $ICUL$, $ICUR$, $ICUT$ and $ICUB$ always operate correctly and we are considering information flow from $ICUL \cup ICUT$ to $ICUR \cup ICUB$.

Definition 3. Given a two-dimensional array A, a *fault pattern* F for A is the set of faulty processors which can be any non-empty subset of A. An assignment of a fault pattern F to A means that every processing element belonging to F is faulty (and the others operate correctly).

Definition 4. A fault pattern is *catastrophic* for the mesh network $\mathcal{M}$ if $ICUL \cup ICUT$ is not connected to $ICUR \cup ICUB$ when the fault pattern F is assigned to A.

Definition 5. Let F be a fault pattern in a mesh network $\mathcal{M}$ with link redundancy $G = (1,\ g_2,\ \ldots,\ g_k \mid 1,\ v_2,\ \ldots,\ v_l)$. If we remove all faulty PEs, their adjacent links and all bypass links from $\mathcal{M}$ then a component of $\mathcal{M}$ will be called a *chunk*. Let $C_0,\ C_1,\ \ldots,\ C_n$ be the chunks of F where C_0 is connected with $ICUL \cup ICUT$, C_n is connected with $ICUR \cup ICUB$ and other C_i's are labelled arbitrarily.

Definition 6. A *path* from a working processor (i_0, j_0) to a possibly faulty processor (i_{s+1}, j_{s+1}) is a sequence of processors (i_0, j_0), (i_1, j_1), ..., (i_s, j_s), (i_{s+1}, j_{s+1}) such that, for each $k = 0, 1, \ldots, s$, processor (i_k, j_k) is a working processor connected by a link to processor (i_{k+1}, j_{k+1}) and a processor is used only once. The length of the path is $s + 1$. An *escape path* is a path form $ICUL \cup ICUT$ to $ICUR \cup ICUB$.

Our main contribution here is a complete characterization of catastrophic fault patterns for mesh networks. Let $\mathcal{M}$ be a mesh network with link redundancy $G = (g_1,\ g_2, \ldots,\ g_k \mid v_1,\ v_2, \ldots,\ v_l)$, and let F be a fault pattern. Then we prove that, F is catastrophic with respect to $\mathcal{M}$ implies that the cardinality of F, $|F| \geq \max\left\{\frac{N_1}{v_l}, \frac{N_2}{g_k}\right\} v_l g_k$. We provide an algorithm to test whether a given F is catastrophic with respect to link redundancy $G = (g_1,\ g_2, \ldots,\ g_k \mid v_1,\ v_2, \ldots,\ v_l)$.

When a fault pattern is not catastrophic, we are interested in finding escape paths. Depending on the fault pattern there can exist several escape paths. We are interested in finding those escape paths that are *optimal* with respect to the size of the reconfigured network or the number of redundant links to be activated to reconfigure the network. Here optimality is achieved when the size of the reconfigured network is maximized, that is, when the number of processors in the escape path that reconfigured the network is maximized. In this case, an optimal escape path is called a *maximum escape path*, and a reconfiguration set that achieves a maximum escape path is called a *maximum reconfiguration set*. For example, consider the fault pattern $F = \{\ (1,3), (1,4),\ (1,5),\ (2,1),\ (2,2),\ (2,3),\ (2,4),\ (2,5),\ (3,1),\ (3,4),\ (3,5),\ (4,1),\ (4,2),\ (4,3),\ (4,4)\}$ in a 4×5 mesh network $\mathcal{M}$ with link redundancy $G = (1, 2, 3 \mid 1, 2)$. The maximum escape path,

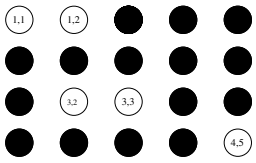

Fig. 2. A mesh with a fault pattern

having 4 processing elements, is given by $ICUL_1$ or $ICUT_1$, (1,1), (1,2), (3,2), (3,3) $ICUR_3$ or $ICUB_3$ for both bidirectional or unidirectional case. However maximum escape path is not unique always.

In the latter case, optimality is achieved when the number of redundant links that we have to activate in order to reconfigure the network is minimized. In this case, an optimal escape path is called *minimum escape path*, and a reconfiguration set that achieves a minimum escape path is called a *minimum reconfiguration set*. For example, consider the fault pattern in Figure 2. The escape path $ICUL_3$, (3,2), (3,3), $ICUR_3$ is a minimum escape path since it uses only two bypass links and there are no escape paths that use only 1 bypass link.

3 Characterization of Catastrophic Fault Patterns

In this section, we will characterize the catastrophic fault patterns for mesh networks and prove that the minimum number of faults in a catastrophic fault pattern is a function of N_1, N_2, the length of the longest horizontal bypass link and the length of the longest vertical bypass link.

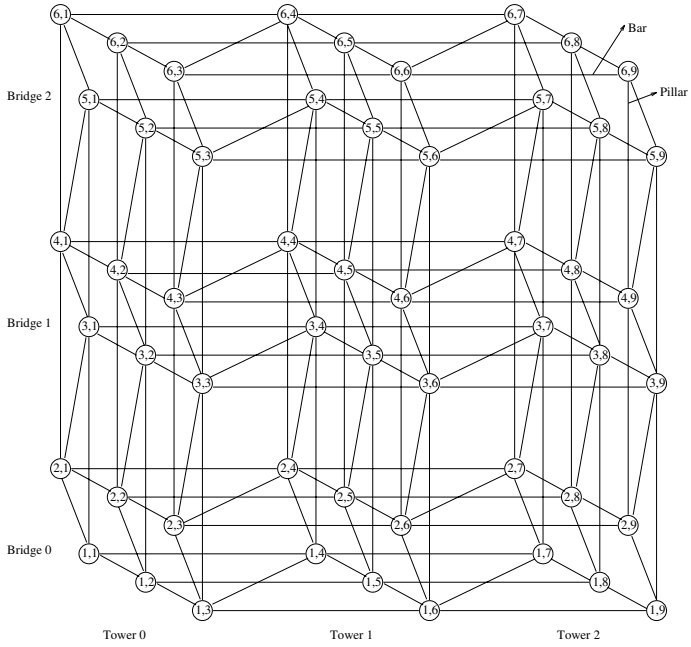

Fig. 3. Tower-Bridge representation of mesh networks of Figure 1

Theorem 1. *Suppose v_l divides N_1 and g_k divides N_2, then F is catastrophic with respect to $\mathcal{M}$ implies that the cardinality of F, $|F| \geq$ max $\left\{\frac{N_1}{v_l}, \frac{N_2}{g_k}\right\} v_l g_k$.*

Proof: Suppose to the contrary that $|F| <$ max $\left\{\frac{N_1}{v_l}, \frac{N_2}{g_k}\right\} v_l g_k$. Then partition the two-dimensional array A of PEs into blocks of v_l rows as $A = \begin{pmatrix} A_1 & A_2 & \dots & A_{\frac{N_1}{v_l}} \end{pmatrix}^T$ and again partition each block A_i into sub-blocks of g_k columns as $A_i = \begin{pmatrix} A_{i1} \vdots A_{i2} \vdots \dots \vdots A_{i\frac{N_2}{g_k}} \end{pmatrix}$. For example, sub-block $A_{12} = \{(1,4), (1,5), (1,6), (2,4), (2,5), (2,6)\}$ in Figure 3. We place the sub-blocks $A_{1j}, A_{2j}, \dots, A_{\frac{N_1}{v_l}j}$ as consecutive floors to form tower j as shown in Figure 3. On the other hand, the sub-blocks $A_{i1}, A_{i2}, \dots, A_{i\frac{N_2}{g_k}}$ form bridge i in between the towers. We will refer the sub-blocks as floors later. Observe that, each horizontal bypass link of the maximum length joins two consecutive elements in the same *bar* of a bridge. On the other hand, each vertical link of the

maximum length joins two consecutive elements in the same *pillar* of a tower. For example, Figure 3 shows the bar [(6, 3), (6, 6), (6, 9)] and the pillar [(2, 9),(4, 9), (6, 9)]. So, in this representation, going up along a pillar corresponds to using the longest vertical bypass links and going right along a bar corresponds to using the longest horizontal bypass links. Note that, there are $N_1 g_k$ bars and $N_2 v_l$ pillars. Now we consider two cases:

Case 1. $\frac{N_1}{v_l} > \frac{N_2}{g_k}$. That is $|F| < N_1 g_k$. Since the number of faulty elements $|F|$ is less than the number of bars, there must be a bar with no faulty element, regardless of the distribution of the fault pattern. Since the left and right of each bar are linked to ICUL and ICUR respectively, F cannot be catastrophic since we can use the horizontal bypass links of length g_k to avoid the faulty PEs, a contradiction.
Case 2. $\frac{N_1}{v_l} < \frac{N_2}{g_k}$. That is $|F| < N_2 v_l$. Since the number of faulty elements $|F|$ is less than the number of pillars, there must be a pillar with no faulty element, regardless of the distribution of the fault pattern. Since the top and bottom of each pillar are linked to ICUT and ICUB respectively, F cannot be catastrophic since we can use the vertical bypass links of length g_k to avoid the faulty PEs, a contradiction which proves the theorem. □

This theorem gives us a necessary condition on the minimum number of faults required for blocking a mesh network when $v_l | N_1$ and $g_k | N_2$. In general we have the following result:

Theorem 2. *F is catastrophic with respect to $\mathcal{M}$ implies that the cardinality of F, $|F| \geq$ max $\{N_1 g_k, N_2 v_l\}$.*

The proof of this theorem is similar to that of Theorem 1. This tells us that fewer than max $\{N_1 g_k, N_2 v_l\}$ faults occurring in A will not be catastrophic.

4 An Algorithm to Test Weather a Fault Pattern Is Catastrophic

4.1 Bidirectional Mesh

Let $\mathcal{M}$ be a bidirectional mesh network of $N_1 N_2$ processors with link redundancy $G = (1, g_2, \ldots, g_k \mid 1, v_2, \ldots, v_l)$, and let F be a fault pattern with m faults. A simple way to test if F is catastrophic for $\mathcal{M}$ is to consider a graph whose set of vertices is given by the chunks of working processors. More formally, we construct a graph $H = (V, E)$ as follows: The set V of vertices is $\{C_0, C_1, \ldots, C_n\}$, where C_i's represent chunks of F and $(C_i, C_j) \in E$ if and only if there are two processors, $p_{xy} \in C_i$ and $p_{x'y'} \in C_j$ such that $y = y'$ and $|x - x'| \in \{v_1, v_2, \ldots, v_l\}$ or $x = x'$ and $|y - y'| \in \{g_1, g_2, \ldots, g_k\}$, that is, such that these two processors are connected in $\mathcal{M}$ by a bypass link.

Fact 1. A fault pattern F is not catastrophic for a network $\mathcal{M}$, if and only if C_0 and C_n are connected in the graph H.

4.2 Unidirectional Mesh

Given a mesh network $\mathcal{M}$ and a fault pattern F, we construct a graph $G_0 = (V, E)$ as follows: The set V of vertices is the set of working processors and ICUs, and the set E of edges are the links between two working processors and the links between working processors and ICUs. In addition, there are two more vertices *source* and *sink*. We join *source* to each of the $ICUT$s and $ICUL$s and join each of the $ICUR$s and $ICUB$s to *sink*. If $\mathcal{M}$ is unidirectional then G_0 is directed and undirected otherwise. For the directed graph we preserve the direction of the link, in the edge representing it. We call the graph G_0 the *auxiliary graph* of $\mathcal{M}$ for the fault pattern F.

Example 1. Consider the fault pattern $F = \{ (1,3), (1,4), (2,1), (2,2), (2,3), (2,4), (3,2), (3,4), (3,5), (4,1), (4,2), (4,4)\}$ in a 4×5 unidirectional mesh network $\mathcal{M}$ with link redundancy $G = (1, 2, 3 \mid 1, 2)$. Figure 4 shows the auxiliary graph.

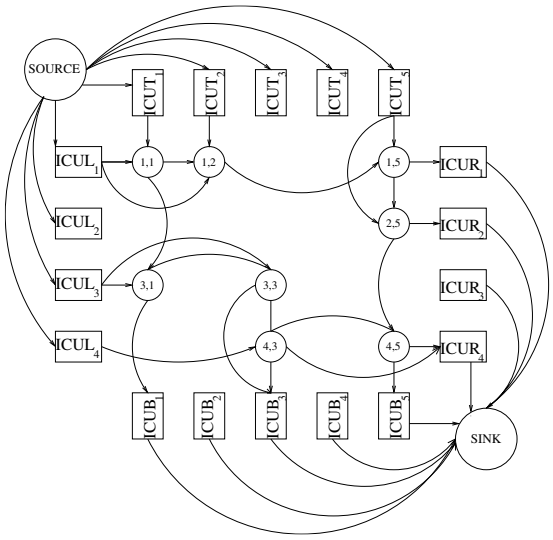

Fig. 4. Auxiliary graph G_0

Note that in the auxiliary graph $G_0 = (V, E)$, $|V| = 2N_1 + 2N_2 + w + 2$ and $|E| \leq 2N_1 + 2N_2 + w|G|$, where w is the number of working processing elements in the mesh. Also note that the auxiliary graph resulting from unidirectional mesh is directed acyclic in nature (See [5]). Time required to construct an auxiliary graph $G_0 = (V, E)$ is $O(2N_1 + 2N_2 + w|G|)$, that is, $O(N_1 + N_2 + w|G|)$.

To find out if the fault pattern is catastrophic or not, we assign weight 1 to all the edges in the auxiliary graph. Now, we can run the shortest path algorithm given in [5] for directed acyclic graphs and find out if there exists a path from *source* to *sink*. In a graph (V, E), the shortest path algorithm takes

$O(|V| + |E|)$ time. A fault pattern F is not catastrophic for mesh $\mathcal{M}$, if and only if *source* is connected with *sink* in the auxiliary graph. Thus the problem of testing whether a fault pattern is catastrophic for unidirectional mesh network requires $O(N_1 + N_2 + w|G|)$ time.

5 Maximum Escape Paths

In this section we consider the problem of finding maximum escape paths. We prove that the problem is NP-complete for a bidirectional mesh network, while for unidirectional mesh we provide an algorithm that finds a maximum escape path in $O(N_1 + N_2 + w|G|)$ time.

5.1 *MRL* Problem in the Case of Bidirectional Links

Consider the following *Maximuim Reconfiguration Length* (MRL for short) problem:

Definition 7. (MRL problem) Given a bidirectional redundant mesh $\mathcal{M}$, with link redundency G, a fault pattern F and a positive integer K, is there an escape path of length at least K?

Theorem 3. *The MRL problem is NP-complete.*

Proof: We reduce the problem of testing whether there exists a hamiltonian path between two given vertices of a graph (HP for short), known to be NP-complete (see [6]) , to MRL problem. Since it is easy to give a non deterministic polynomial time algorithm that solves MRL problem we conclude that MRL problem belongs to the class NP.

Let $H = (V, E)$ be the input graph of the HP problem. Without loss of generality, assume that $V = \{1, 2, ..., n\}$ and that 1 and n are the vertices to be tested.

Consider the following instance of our problem. The mesh $\mathcal{M}$ has $N_1 = 3$ rows and $N_2 = \frac{(6n^3-3n^2-9n+8)}{2}$ columns. For $i = 1, 2, \ldots, n$ define $a_i = (n + i - 2)n^2 + \frac{(n-1)(n-2)+(i-1)(i-2)}{2} + 1$. The horizontal link redundancies are for each edge $(i, j) \in E$, a horizontal bypass link of length $\mid a_i - a_j \mid$. Moreover there is an additional bypass link of length $g = a_1$ (it is easy to see that this is the longest horizontal link connecting $ICUL_2$ and PE $(2, a_1)$, and also PE $(2, a_n)$ and $ICUR_2$). The vertical link redundancy will be singleton $\{1\}$. The fault pattern F consists of all the processing elements in the first and the third rows and all the processing elements in the second row except those in the a_ith column, i.e., in the whole mesh only the processing elements $(2, a_i)$, $1 \leq i \leq n$, are working. Finally $K = n$.

Notice that the above MRL instance can be constructed in time polynomial in the size of the graph and all the integers occurring in the description of the instance are polynomially related to n.

We will prove that H has an hamiltonian path if and only if the above instance of the MRL problem admits a solution, i.e., if there is an escape path of size n. In order to prove this, we first need the following four facts.

a The only possible escape path connects $ICUL_2$ and $ICUR_2$. It is easy to notice that all other pairs of ICU's are already disconnected.

b Any escape path must traverse $(2, a_1)$ and $(2, a_n)$. Indeed, as the first and the last $g-1$ processing elements are faulty and the length of the longest link is g.

c If $(2, a_i)$, with $1 < i < n$, is traversed by an escape path, then it must be traversed after $(2, a_1)$ and before $(2, a_n)$. Indeed, let d_{ij}, $1 \leq i \neq j \leq n$, be the distance between $(2, a_i)$ and $(2, a_j)$, i.e., $d_{ij} =\mid a_i - a_j \mid = n^2 \mid i - j \mid + \mid \frac{(j-1)(j-2)}{2} - \frac{(i-1)(i-2)}{2} \mid$. Since for $(i \neq j)$ it holds that $n^2 \mid i - j \mid < d_{ij} < n^2 \mid i - j \mid +n^2$, then (for $1 \leq i \neq j, u \neq v \leq n$), we have $d_{ij} = d_{uv}$ if and only if $\{i, j\} = \{u, v\}$.

d Graph H is isomorphic to the graph consisting of the non faulty elements $(2, a_i)$, $i = 1, 2, \ldots, n$ and their incident horizontal links. Indeed, since $d_{ij} < d_{1n} < g$, $1 \leq i \neq j \leq n$, processors $(2, a_i)$ and $(2, a_j)$ are connected by a bypass link, if and only if vertices i and j are connected by an edge in graph H. Moreover, since no other two working processors are at a distance d_{ij}, this bypass link connects only $(2, a_i)$ and $(2, a_j)$.

Now we can prove that there is an escape path of length at least $K = n$ if and only if there is an hamiltonian path between vertices 1 and n in the graph H. Assume that there is an escape path of size $K = n$. Since in $\mathcal{M}$ there is exactly K working processors, each processor is involved in the escape path. Since all the working processors are traversed, by **a, b, c, d**, we conclude that there exists a hamiltonian path between vertices 1 and n in H (recall that by the definition of path each processor can be traversed at most once).

Conversely, given a hamiltonian path between vertices 1 and n in H, by **d**, it corresponds to a path from $(2, a_1)$ to $(2, a_n)$, which traverses once all the non faulty processing elements of $\mathcal{M}$. This path can be easily extended to an escape path of size $K = n$ connecting $ICUL_2$ to $(2, a_1)$ and $ICUR_2$ to $(2, a_n)$, respectively, by means of the longest bypass link.

Therefore we can test if there exists a hamiltonian path between vertices 1 and n in H by testing if there exists an escape path of size at least K for the array $\mathcal{M}$. □

5.2 *MRL* Problem in the Case of Unidirectional Links

When the mesh is unidirectional, the problem of finding a maximum escape path can be solved in $O(N_1 + N_2 + w|G|)$ time.

Definition 8. *An auxiliary escape path* is a path from *source* vertex to *sink* vertex in an auxiliary graph.

It is easy to note that if there are p edges in an auxiliary escape path then the corresponding escape path contains $p - 3$ processing elements.

Given a fault pattern F in an unidirectional mesh network, we first construct the corresponding auxiliary graph $G_0 = (V, E)$. To get the escape path with maximum processors, we assign weight -1 to all the edges in E. Now we run the

single source shortest path algorithm for directed acyclic graphs (DAG) given in [5] on G_0 with the source vertex as the *source*. The shortest auxiliary path, if any, from *source* to *sink* is obtained. By shortest auxiliary path we mean that the sum of the edge weights in the path is least. This algorithm takes $O(|V|+|E|)$ time. The construction of the graph takes $O(N_1+N_2+w|G|)$ time. It takes $O(|E|)$ time to assign the weight -1 to each edge. So the total time spent to find out an escape path with maximum processing elements, if any, is $O(N_1 + N_2 + w|G|)$.

6 Minimum Escape Paths

In this section we consider the problem of finding minimum escape paths. We prove that the problem can be solved in $O(w|G|+(N_1+N_2+w)\log(N_1+N_2+w))$ time if the bypass links are bidirectional, and in $O(N_1 + N_2 + w|G|)$ time if the bypass links are unidirectional. First we consider the case of bidirectional links.

Given a fault pattern F in a bidirectional mesh network, we construct the auxiliary graph $G_0 = (V, E)$ and assign weight 1 to all the edges representing the bypass links (horizontal or vertical) in G_0 and 0 to all the remaining edges. Now run the Dijkstra's single source shortest path algorithm on G_0 with source vertex as the *source*. The shortest auxiliary escape path, if any, is then obtained. This algorithm requires $O(|V|\log|V| + |E|)$ time (using Fibonacci heap data structure, see [5]). The construction of the auxiliary graph takes $O(N_1 + N_2 + w|G|)$ time and it takes $O(|E|)$ time to assign weight to each edge. Thus the problem of finding minimum escape path for bidirectional mesh network requires $O(w|G| + (N_1 + N_2 + w)\log(N_1 + N_2 + w))$ time.

Now we consider the case of unidirectional links. Given a fault pattern F in a unidirectional mesh network, we construct the auxiliary graph $G_0 = (V, E)$ and assign weight 1 to all the edges representing the bypass links (horizontal or vertical) in G_0 and 0 to all the remaining edges. Now, repeat the shortest path algorithm for DAG, as in the subsection 5.2. The shortest auxiliary escape path, if any, is obtained. Thus the problem of finding minimum escape path for unidirectional mesh network requires $O(N_1 + N_2 + w|G|)$ time.

7 Conclusions

In this paper we have completely characterized catastrophic fault pattern for mesh networks. Before attempting any reconfiguration it is important to test whether the set of faults is catastrophic. We have presented testing algorithms to test whether a given fault pattern is catastrophic. When a set of faults is not catastrophic it is important to provide efficient reconfiguration algorithms that provide optimal reconfigurations. Optimality is considered either with respect to the size of the reconfigured network or with the amount of bypass links need to reconfigure the network. We have proved that when the links are bidirectional, the

problem of finding optimal reconfiguration with respect to the size of the reconfigured network is NP-complete. In all the other three cases we give algorithms which efficiently find an optimal reconfiguration.

Acknowledgment. The authors like to thank the anonymous referees for important comments that improved the technical quality of the paper.

References

1. Balasubramanian, V. and Banerjee, P., A fault tolerant massively parallel processing architechture, *Journal of Parallel and Distributed Computing*, Vol. 4, 1987, pp. 363-383.
2. Bruck, J., Cypher, R. and Ho, C.T., Fault-tolerant meshes with minimal number of spares, *Proc. of 3rd IEEE Symposium on Parallel and Distributed Processing*, 1991, pp. 288-295.
3. De Prisco, R., Monti, A. and Pagli, L., "Efficient testing and reconfiguration of VLSI linear arrays", *Theoretical Computer Science*, Vol. 197, 1998, pp. 105-129.
4. De Prisco, R. and De Santis, A., Catastrophic faults in reconfigurable systolic linear arrays, *Discrete Applied Math.,* Vol. 75, 1997, pp. 105-123.
5. Cormen, T. H., Lierson, C. E, Rivest, R. L. and Stein, C., *Introduction to Algorithms*. MIT Press, Cambridge, MA.
6. Garey, M. and Johnson, D., *Computers and intractability*, Freeman, New York, 1979.
7. Kung, H. T., Why systolic architecture? *IEEE Computer*, Vol. 15, Jan. 1982, pp. 37-46.
8. Maity, S., Roy, B. and Nayak, A., Enumerating catastrophic fault patterns in VLSI arrays with both uni- and bidirectional links, *INTEGRATION: The VLSI Journal*, Vol. 30, 2001, pp. 157-168.
9. Maity, S., Nayak, A. and Roy, B., On Characterization of Catastrophic Faults in Two-Dimensional VLSI Arrays. *INTEGRATION, The VLSI Journal*, Vol. 38, 2004, pp. 267-281.
10. Maity, S., Roy, B. and Nayak, A., On Enumeration of Catastrophic Fault Patterns, *Information Processing Letters* Vol. 81, 2002, pp. 209-212.
11. Maity, S., Nayak, A., Roy, B., Reliability of VLSI Linear Arrays with Redundant Links. IWDC 2004, *Lecture Notes in Computer Science* Vol. 3326, pp. 326-337
12. Nayak, A., Pagli, L. and Santoro, N., Efficient construction of catastrophic patterns for VLSI reconfigurable arrays, *INTEGRATION: The VLSI Journal*, Vol. 15, 1993, pp. 133-150.
13. Nayak, A., Pagli, L. and Santoro, N., On testing of catastrophic faults in reconfigurable arrays with arbitrary link redundancy, *INTEGRATION: The VLSI Journal*, Vol. 20, 1996, pp. 327-342.
14. Nayak, A., Santoro, N., and Tan, R., Fault-Intolerance of reconfigurable systolic arrays, *Proc. of 20th Int. Symp. on Fault-Tolerant Computing*, 1990, pp. 202-209.
15. Leighton, T. and Leiserson, C. E., Wafer-scale integration of systolic arrays. *IEEE Trans. on Computers*, Vol. C-34, 1985, pp. 448-461.