Diagnosis of Processor Arrays

Laura Baldelli and Piero Maestrini

Dipartimento di Informatica, Università di Pisa, Pisa, Italy

Abstract

This paper introduces an approach to diagnosis of processor arrays, assuming that processors are horizontally, vertically and diagonally connected to their neighbors. The proposed algorithm subdivides the array into clusters containing nine processors and requires three steps. In the first step each cluster executes tests according to a rosace pattern, and clusters for which all test results were zero are classified as Z-C's. The remaining cluster, which contain at least one fault, are classified NZ-C's. In the second step, Z-C's are combined into aggregates (Z-AC's) and one Z-AC's is identified as a fault-free core of the array. The third step leads to identification of the state (of faulty or non faulty) of more nodes. The diagnosis is proved to be correct, although possibily incomplete, assuming that the number of faults is less that a bound T, which is an increasing function of the size of the array.

1: Introduction

The progress of VLSI technology leads to development of multiprocessor systems consisting of an increasingly large number of interconnected processors. One essential feature of very large systems is the regularity of the interprocessor communication pattern. Typical examples of regular architectures are arrays, trees and hypercubes. In such architectures, any processor is allowed to directly communicate with a small number of neighbors, arranged in a regular pattern.

As the system size increases, so does the probability that one or more processors become faulty. Although it can be expected that the number of faults increases less than proportionally with the number of processors, a very large system should be able to tolerate a relatively large number of faults. To ensure reliable computation, faulty processors should be identified and either isolated or replaced with spares.

Identification of faulty processors is the objective of system diagnosis. Models and algorithms for system diagnosis have been extensively investigated, using concentrated [1, 2, 3, 4] or distributed approaches [5, 6, 7, 8, 9]. Although different models are significantly different in many respects, all of them rely upon tests of processors performed by other processors. Performing the tests requires direct communication between testing and tested processors.

In the case of regularly interconnected systems, the limited interprocessor communication is a major obstacle to straightforward application of standard self-diagnosing algorithms. For example, in the case of PMC model [1], the self-diagnosing capability of any sistem is limited by its *one-step diagnosability* [1]. If the actual number of faults exceeds this figure, identification of faulty units may be impaired. In a regular system where each processor is directly connected to d neighbors, the one-step diagnosability is equal to d; i.e., a small constant. This contrasts with the requirement that the system should be able to tolerate a number of faults which is an incresasing function of its size. The same or similar limitations apply to other self-diagnosing models.

Extensive effort has been deployed to develop effective diagnosing algorithms for regularly interconnected systems [10, 11, 12]. The challenge is to find strategies leading to a diagnosis which is both *correct* and *complete*. If the faulty processors tend to concentrate into clusters, those with little or no comunication with good processors may be diagnosed as non-faulty, thus causing incorrect diagnosis. Further, the diagnosing algorithm may be unable to determine the state of some processors: in this hypothesis the diagnosis is said to be incomplete. Incorrect and incomplete diagnosis are almost unavoidable when processors are constrained to test a limited number of neighbors, unless appropriate but usually optimistic assumptions are made to limit the number and the possible distributions of faults. In the previous papers, these limitations have been overcome by using probabilistic analysis and simulation to verify that the proposed algorithms are very likely to produce correct and almost complete diagnosis.

This paper introduces a strategy to diagnose bidimensional processor arrays, leading to a diagnosis which is provably correct, although possibly incomplete. The diagnosis is correct provided the number of faults in the system is less than a threshold T. Contrary to one-step diagnosability and similar parameters, this threshold is an increasing function of the number of processors in the array, although the number of interprocessor connections is a small constant.

This strategy is based upon partitioning the array into clusters of processors and requires three steps. In the first step all clusters are diagnosed individually, using tests independently performed by nodes in each cluster. In the second step the information previously gained is augmented by performing intecluster tests between selected clusters. This step leads to identification of an aggregate of clusters which is a *fault-free core* of the array. Starting from the fault-free core, the third step incrementally defines a set of non-faulty processors, which in turn perform reliable tests of neighbors to identify more processors as faulty or non-faulty. Altough this strategy lends itself to different implementations, it is assumed that, in every step, syndrome decoding is done by a centralized, inherently reliable decoder.

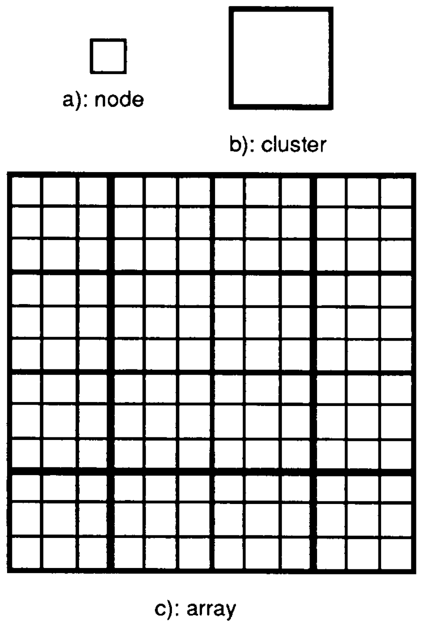


Fig. 1: Clusters in the processor array

2: Diagnosing strategy

Let A be an array of N processors and assume that every processor has horizontal, vertical and diagonal connections with its neighbors (i. e., it is connected to 8 other processors), unless it lies along the boundary of the array. Every connection provides bidirectional communication. Node n_i is said to be *adjacent* to node n_j if there exists a connection between n_i and n_j.

The array is considered to be partitioned into k *clusters*. Every cluster is a set of 9 nodes arranged in a square. One of the nodes is the *center* and the remaining nodes, called *peripheral* nodes, are adjacent to the center. For the sake of simplicity we assume a square array with $N= (3L)^2$, where L is an integer, although this limitation is not necessary and does not impair the generality of the results to be presented. This limitation implies that $k= L^2= N/9$, every node belongs to a unique cluster and the clusters are non overlapping (Fig. 1).

Clusters C_i and C_j, $i \neq j$, are said to be *adjacent* if there exists a connection between at least one node in C_i and at least one node in C_j. Depending upon their mutual arrangement, clusters may be adjacent horizontally, or vertically, or diagonally.

Any test of processor n_j is performed by some processor n_i which is adjacent to n_j. To perform the test, processor n_i sends a test sequence to n_j and receives a response from n_j; then n_i compares the response with the expected result to produce the test outcome. The test outcome is binary: 0 if the test passes, 1 if the test fails. The test invalidation rule is assumed to be the same as in the PMC model [1]. For the sake of simplicity, a fault in the bidirectional connection utilized to perform the test is assumed to be equivalent to a fault in the tested node, although different assumptions are possible.

The set of tests being executed to perform the diagnosis *(test pattern)* is defined by the diagnostic graph $D= (N, A)$, a directed graph where any node $n_i \in N$ corresponds to a processor and any arc $(n_i, n_j) \in A$ corresponds to a test of n_j executed by n_i. Upon execution of all tests in the test pattern, the arcs are labelled with the binary test outcomes. The set of all the test outcomes is called the *syndrome*. The proposed diagnosing algorithm is partly adaptive and proceeds through three steps. The different steps use different test patterns.

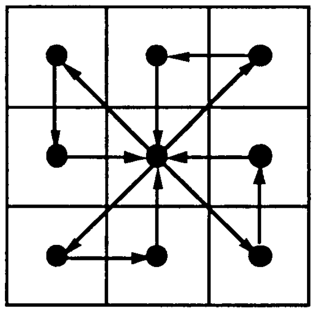


Fig. 2: Test pattern in cluster diagnosis

3: Cluster diagnosis

In the first step *(cluster diagnosis)*, nodes in each cluster test nodes in the same cluster using the test pattern shown in Fig. 2. The diagnostic graph of Fig. 2 is known as a *rosace*. A strong property of rosaces has already been exploited [13, 14] within strategies of sequential diagnosis. More simply, the cluster diagnosis exploits the following weak property, which can be easily verified: if performing all tests in the rosace yields the syndrome s_0, defined as the syndrome where all test outcomes are 0, then either all nodes in the rosace are non-faulty or all of them are faulty. Conversely, for any syndrome different from s_0, at least one node in the node set of the rosace must be faulty.

Testing of clusters proceeds independently: for each cluster, all tests in the rosace are executed to obtain the local syndrome. If the local syndrome is different from s_0, then the cluster must contain at least one faulty node and it is said to be a *non-zero-cluster (NZ-C)*. Clusters being identified as *NZ-C's* in the first step are called *first-step NZ-C's*. Otherwise the cluster is said to be a *zero-cluster (Z-C)*. In a *Z-C*, either all nodes are non-faulty *(true-zero-cluster*, or *TZ-C)*, or all of them are faulty *(false-zero-cluster*, or *FZ-C)*. *TZ-C's* and *FZ-C's* are indistinguishable by the local syndrome alone. Observe that there exists necessarily at least one *Z-C*, which is actually a *TZ-C*, provided the number of faults in the array is less than $k = N/9$.

4: Intercluster diagnosis

In the second step, called *intercluster diagnosis*, any two *Z-C's* which are adjacent test each other to identify more *NZ-C's*, called *second-step NZ-C's*, and to define aggregates of *Z-C's*. Intercluster diagnosis is performed by a peripheral node in the testing cluster, which tests a peripheral node in the tested cluster. The test pattern is a subgraph of the pattern shown in Fig. 3. Note that:

- if C_i is a *Z-C*, each peripheral node in cluster C_i tests a unique cluster C_j which is horizontally, vertically or diagonally adjacent to C_i, provided C_j exists and it is also a *Z-C;*

- for any two *Z-C's*, C_i and C_j, which are adjacent, there exist unique nodes $n_i \in C_i$ and $n_j \in C_j$ which test each other with tests (n_i, n_j) and (n_j, n_i). Tests (n_i, n_j) and (n_j, n_i) should be performed in sequence.

Let C_i and C_j be *Z-C's*, n_i be the testing node in C_i, n_j be the tested node in C_j and a_{ij} be the binary outcome of test (n_i, n_j). Similarly, let n_j be the testing node in C_j, n_i be the tested node in C_i and a_{ji} be the binary outcome of test (n_j, n_i). It is easily seen that:

- if both C_i and C_j are *TZ-C's*, the test outcomes will necessarily be $a_{ij} = 0$ and $a_{ji} = 0$. C_i and C_j retain their state of *Z-C's;*

- if C_i is a *TZ-C* and C_j is a *FZ-C*, then necessarily $a_{ij} = 1$ and C_j becomes a (second step) *NZ-C*. Instead, test (n_j, n_i) may yield both outcomes. If $a_{ji} = 0$, then C_i retains its state of *Z-C;* otherwise C_i is redefined as a (second step) *NZ-C*. Note that, in the latter hypothesis, the state of C_i is mistaken, since all of its nodes are non-faulty;

- the case where C_i is a *FZ-C* and C_j is a *TZ-C* is symmetric of the previous one;

- if both C_i and C_j are *FZ-C's*, then tests (n_i, n_j) and (n_j, n_i) may yield both outcomes. If $a_{ij} = 1$ (or $a_{ji} = 1$), then C_j (or C_i) is identified as a (second step) *NZ-C*. If $a_{ij} = 0$ (or $a_{ji} = 0$), then C_j (or C_i) retains its state of *Z-C*.

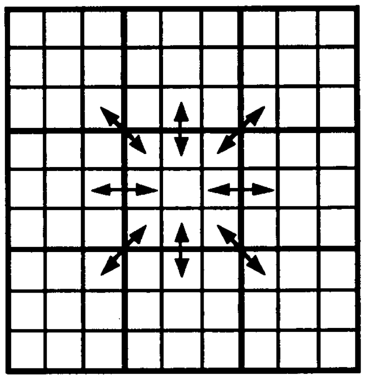


Fig. 3: Test pattern in intercluster diagnosis

Let nz_1 and nz_2 be the number of first-step *NZ-C's* and the number of second-step *NZ-C's*, respectively. Then the sum $nz_1 + nz_2$ is a lower bound to the number of faulty processors in the array. In fact, every first-step *NZ-C* contains at least one fault. Further, let C_i be any second-step *NZ-C* and recall that C_i was identified as a *Z-C* in the first step. If C_i is actually a *FZ-C*, it contains 9 faulty processors. If C_i is a *TZ-C*, it has necessarily been mistaken for a second step *NZ-C* by some adjacent cluster C_j. In turn, C_j was classified as a *Z-C* (actually, it is a *FZ-C)* in the first step and it was redefined as a *NZ-C* in the second-step because of test outcome $a_{ij} = 1$. In the worst case (Fig. 4), a single cluster C_j will erroneously classify 8 adjacent *TZ-C's* as second-step *NZ-C's*, and a total of 9 *NZ-C's* (including C_j itself) will account for the 9 faulty processors in C_j.

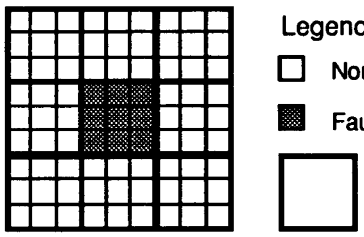


Fig. 4: Worst case for second-stage NZ-C's

If C_i and C_j are *Z-C* which test each other with $a_{ij}= 0$ and $a_{ji}= 0$, then they combine into one *zero-aggregate of clusters (Z-AC)*. By repeatedly applying this rule to combine adjacent *Z-C's*, *Z-AC's* are augmented up to their maximum size. A *Z-C* which fails to combine is considered to be a *Z-AC* of size 1.

The intercluster diagnosis eventually defines a number of *Z-AC's*, each consisting of one or more *Z-C's*. Every *Z-AC* is an aggregate of *Z-C's* all of which are either *TZ-C's* or *FZ-C's*: in the first case the *Z-AC* is said to be a *true-zero-aggregate of clusters (TZ-AC)*, otherwise it is called a *false-zero-aggregate of clusters (FZ-AC)*. Any *TZ-AC* is called a *fault-free core* of the array.

Unfortunately, *TZ-AC's* and *FZ-AC's* are indistinguishable by the test outcomes alone. However, it will be shown that at least one *Z-AC* can be identified as a fault-free core, provided the number of faults is less than a certain threshold *T*. It will also be shown that *T* is an increasing function of the number *N* of processors in the array.

5: Sparse diagnosis

Let AC_i be any *Z-AC* defined in the array upon completion of step 2 of the algorithm. The *area* of AC_i is defined as the number of *Z-C's* combining into AC_i. The *perimeter* of AC_i, denoted $p(AC_i)$, is defined as the set of clusters such that $AC_i \cap p(AC_i)= \varnothing$ and every cluster in $p(AC_i)$ is adjacent to at least one cluster in AC_i.

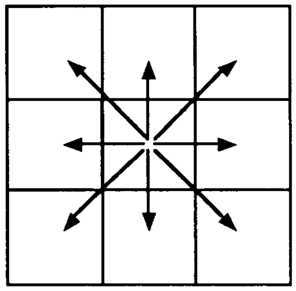


Fig. 5: Test pattern in sparse diagnosis

Since each *Z-AC* expands to its maximum size in step 2, it is completely enclosed whithin its perimeter, unless it is situated along the boundary of the array. Further, any C_j with $C_j \in p(AC_i)$ is necessarily a *NZ-C* (either a first-step *NZ-C* or second-step *NZ-C*). The circumstance that the *Z-AC's* are secluded within their perimeters does not prevent, in general, from extending the diagnosis beyond the perimeter itself. In fact, if there exist non-faulty nodes belonging to clusters in the perimeter, as it can be expected, reliable diagnosis may propagate through those nodes from the fault-free core.

Let AC_i be an arbitrary *Z-AC*, $\mathcal{AC}_i$ be the union of the node sets of clusters in AC_i, and $\mathcal{N}\text{-}\mathcal{AC}_i$ be the complement of $\mathcal{AC}_i$ in the node set of the array. Assuming AC_i to be a fault-free core, the objective of the sparse diagnosis is to augment $\mathcal{AC}_i$ with more non faulty nodes, up to its maximum cardinality. The sparse diagnosis is started by all nodes in $\mathcal{AC}_i$ which are adjacent to at least one node in $\mathcal{N}\text{-}\mathcal{AC}_i$. Any such $n_p \in \mathcal{AC}_i$ tests every adjacent node n_q, with $n_q \in \mathcal{N}\text{-}\mathcal{AC}_i$, using a test pattern which is a subgraph of the pattern shown in Fig. 5. Let A_p be the arc set of the actual test pattern (that is, the set of tests actually performed by n_p). The algoritm begins with node set $\mathcal{G}= \mathcal{AC}_i$. As the sparse diagnosis proceeds, set $\mathcal{G}$ is recursively redefined as follows:

$$\mathcal{G}= \mathcal{G} \cup \{n_q \,/\, n_q \in \mathcal{N}\text{-}\mathcal{G};\ n_p \in \mathcal{G};\ (n_p, n_q) \in A_p;\ a_{pq}= 0\}.$$

The sparse diagnosis can be speeded-up by assuming that $\mathcal{G}$ is augmented as $\mathcal{G}= \mathcal{G} \cup \mathcal{AC}_j$ whenever $n_p \in \mathcal{G}$ tests $n_q \in \mathcal{N}\text{-}\mathcal{G}$ with test outcome $a_{pq}= 0$ and $n_q \in \mathcal{AC}_j$, where $\mathcal{AC}_j$ is a *Z-AC*.

As $\mathcal{G}$ has been augmented to its maximun size, then every n_r such that $n_p \in \mathcal{G}$; $n_r \in \mathcal{N}\text{-}\mathcal{G}$; $(n_p, n_r) \in A_p$, is diagnosed as faulty. However, the correctness of the diagnosis depends upon the ability to select one *Z-AC* which is actually a fault-free core.

6: Existence of a fault-free core

The correctness of the diagnosis performed by the algorithm described in the preceding sections is based upon the assumption that at least one *Z-AC* can be identified as a fault-free core. It will be shown that this is actually the case, provided the number of faults is less than a threshold *T*, where *T* is an increasing function of the number of nodes in the array.

Once steps 1 and 2 of the algorithm have been completed, every cluster is classified as either a *Z-C* or a *NZ-C*. *Z-C's* are aggregated into *Z-AC's*, each consisting of one or more *Z-C's*, and *Z-AC's* are enclosed within their perimeters of *NZ-C's*. The number of faulty nodes in the array is limited below by the number of *NZ-C's*. In turn, the number of *NZ-C's* is limited below by the number of distinct clusters belonging to the perimeters of *Z-AC's*. If *z* is the actual number of *Z-AC's*, a lower bound to the number of faulty nodes is given by:

$$\#\,\mathcal{P}= \#\cup_{i=1,\,z}\, p(AC_i),$$

where $\mathcal{P}$ is called the *perimeter set* and $\#\,\mathcal{P}$ denotes its cardinality. This bound is actually reached if every *Z-C* is a *TZ-C* and every *NZ-C* is in the perimeter of some *Z-AC* and it contains exactly one faulty node.

However, from the point of view of the diagnosing algorithm, this corresponds to the favorable situation in which all *Z-AC's* are fault-free cores. The most adverse situation occurs if exactly one *Z-AC* is a *FZ-AC*, while any other *Z-AC* is a fault-free core (Fig. 6). In fact, this situation would lead to incorrect diagnosis in the unfortunate event that the third step of the diagnosing algorithm is started from the unique *FZ-AC*. Let AC_f, of area A, be the unique *FZ-AC:* then the number of faulty nodes in AC_f is $9A$ and a lower bound to the number of faults in the array is:

$$F_A = 9A + \#\mathcal{P}.$$

For any given A, the value of F_A depends upon the areas and shapes of all the *Z-AC's*. In fact, the area A of AC_f contributes to F_A with $9A$ faults, while every *Z-AC*, including AC_f itself, contributes to the perimeter set with a number of *NZ-C's* depending on its area and shape. Let F_{Amin} be the minimum of F_A over all possible areas and shapes, subject to the constraint that AC_f has area A and any other *Z-AC* has area no greater than A. If the number of faults is less than F_{Amin}, than the area of at least one *TZ-AC* must be greater than A. Define t as the minimum of F_{Amin} over all possible values of A. Thus, assuming a number of fault less than t implies that the *Z-AC* of maximum area is necessarily a fault-free core.

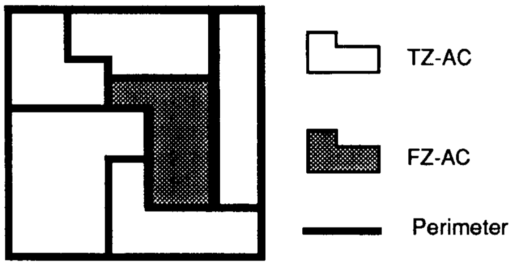
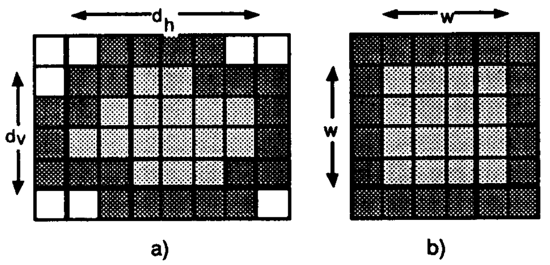


Fig. 6: Worst case for identification of a fault-free core

In order to derive F_{Amin}, consider an arbitrary *Z-AC*, denoted AC_i, which is not situated along the boundary of the array. Define the *horizontal diameter* (d_h) of AC_i as the maximum number of clusters in AC_i being intersected by any horizontal line, with the property that every intersected cluster either is in AC_i or is bracketed between at least one cluster in AC_i situated somewere on its left side and at least one cluster in AC_i situated somewere on its right side. Define similarly the *vertical diameter* (d_v) of AC_i. It is easily seen (Fig. 7a) that, for any shape of AC_i, the cardinality of $p(AC_i)$ is limited below by $2(d_h+1)+2(d_v+1)$. Further, with this cardinality of perimeter, the *Z-AC* of maximum area is the one completely filling the rectangle of sides d_h and d_v. On the other hand, the perimeter is an absolute minimum for given area if $d_h = d_v$ (Fig. 7b). It is concluded that, for any AC_i of area A, the cardinality of $p(AC_i)$ is limited below by $\#p_{min}(AC_i)$, defined as follows:

$$\#p_{min}(AC_i) = \left\lceil 4\left(\sqrt{A}+1\right)\right\rceil$$

This bound corresponds to the ideal situation in which AC_i fits a square of side $\sqrt{A}$. The ratio $A/\#\,p(ACi)$ is a non decreasing function of A. Similar results hold for *Z-AC's* which lie along the boundary.



▨ Cluster in AC_i ■ Cluster in $p(AC_i)$

Fig. 7: Z-AC's of minimum perimeter

In order to determine a bound to F_{Amin} for given A, consider a square array of N nodes. It is easily seen that the cardinality of the perimeter set is limited below by $\#\,p_{min}$, with:

$$\#\,p_{min} = 2\lambda\frac{\sqrt{N}}{3} - \lambda^2,$$

where:

$$\lambda = \left\lfloor \frac{\sqrt{N}/3 + \sqrt{A}}{\sqrt{A}+1} \right\rfloor$$

This bound corresponds to a perimeter set shaped as λ horizontal and λ vertical chains of *NZ-C's*, which are pairwise spaced apart by $\sqrt{A}$ clusters (Fig. 8). With this pattern, every *Z-AC* of area A is enclosed within a perimeter whose cardinality reaches the lower bound. Reducing to area A the remaining *Z-AC's* would increase the cardinality of the perimeter set. Assuming one *FZ-AC* of area A, a lower bound to the number of faults is determined as:

$$F_{A\,min} = 2\lambda\frac{\sqrt{N}}{3} - \lambda^2 + 9A.$$

For any given N, this function has a minimum for some positive A in the domain of reals. The minimum t in the domain of integers can be determined in a straigthforward way. It is seen that t is an increasing function of N.

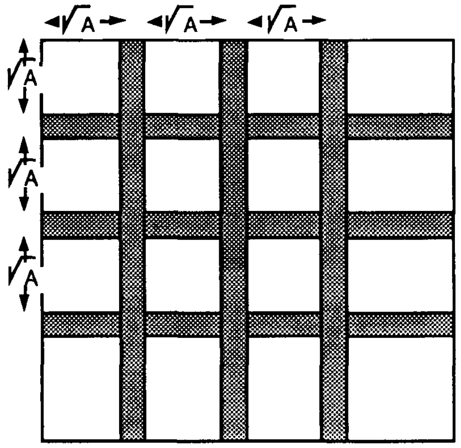


Fig. 7: Bound to the cardinality of the perimeter set

The threshold T is defined as $min(t, k)$, where $k= N/9$ is the number of faults below which there must exist at least one Z-C. Unless the array is relatively small, it is seen that $t< k$. In Tab. 1, the values of T are listed for some values of the size N of the array. If the array has N nodes and the number of faults is less than T, then any Z-AC of maximum area, denoted AC_{max}, must be a fault-free core. This does not exclude existence of one or more FZ-AC's each of which, however, must have an area less than the maximum. Selecting AC_{max} to start the third step of the diagnosing algoritm yields a correct diagnosis.

7: Final remarks

Assuming a sequential computational model (i. e., a centralized implementation of the diagnosing algorithm), the complexity of the algorithm which has been presented in this paper is clearly $O(N)$. In fact, since the number of tests performed by each processor is bounded above by a constant, the time to collect the test outcomes is $O(N)$, provided the central diagnoser has access to every node. For each cluster, the syndrome obtained in step 1 is decoded in constant time. The bookkeeping operations needed in step 2 to aggregate Z-C's into Z-AC's, and in step 3 to drive the sparse diagnosis starting from the fault-free core are easily performed in $O(N)$ time by appropriate sequential algorithms.

The ability to provide a diagnosis which is certainly correct in the assumption that the number of faults is less than T is the main advantage of the algorithm presented in this paper. Nevertheless, the diagnosis is not necessarily complete. In fact, the algorithm is unable to identify the state (of faulty or non-faulty) of those nodes which are made inaccessible to sparse diagnosis by an encircling set of faulty nodes. For those nodes which remain in a unidentified state and belong to the perimeter set, it is only known that at least one out of nine is faulty. However, an analysis similar to that developed in [11] shows that almost complete diagnosis is a very likely result. [15].

N	T	N	T
100	11	3500	199
200	17	4000	224
300	28	4500	243
400	31	5000	262
500	43	5500	280
600	46	6000	307
700	52	6500	324
800	60	7000	342
900	69	7500	358
1000	72	8000	381
1500	104	8500	399
2000	125	9000	416
2500	153	9500	432
3000	175	10000	452

Table 1: Threshold T as a function of the size N of the array

If the objective of the diagnosis is graceful degradation, the faulty or inaccessible nodes should be disconnected. After this operation, the array may continue to work with degraded performance, possibly as a collection of non communicating distributed subsystems. However, if the regularity of the connection pattern is a necessary feature of the processor array, graceful degradation seems to be of little interest and replacement of faulty processors is more suitable as the objective of diagnosis.

Techniques and strategies for replacement with spares are beyond the scope of this paper. Assuming that replacement is possible, one application of the diagnosing algorithm will lead to identification of a set $S_{\mathcal{G}}$ of good nodes, a set $S_{\mathcal{F}}$ of faulty nodes and possibly a set $S_{\mathcal{U}}$ of nodes which cannot be declared neither good nor faulty. The nodes identified as faulty are replaced. If set $S_{\mathcal{U}}$ is non-empty, then the diagnosing algorithm should be repeated one or more times, until all faulty nodes are identified and replaced. As a matter of fact, the algorithm presented in this paper is a heuristic to solve the problem of sequential diagnosis, as defined in [1], since all faults can be identified and repaired by repeated application of the diagnosing algorithm, provided their number is less than T. Thus, this parameter provides a lower bound to the *sequential diagnosability* of the given array.

It should be stated that this paper reports the current state of a continuing research, the scope of which extends tho other regular architectures. The reason why the rosace of 9 nodes was selected as the first choice for the test pattern in cluster diagnosis is that it provides an easier characterization of the perimeter set. However, preliminary results have also been obtained using different test patterns which do not imply diagonal connections.

References

[1] F.P Preparata, G. Metze, R.T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 848- 854, 1967.

[2] F. Barsi, F. Grandoni, P. Maestrini, "A theory of diagnosability of digital systems," *IEEE Trans. Comput.*, vol. C-25, pp. 585 - 593, 1976.

[3] J. Maeng, M. Malek, "A comparison connection assignment for self-diagnosis of multiprocessor systems," *IEEE Trans. Comput.*, vol. C-30, pp. 173 - 175, 1981.

[4] M. L. Blount, "Probabilistic treatment of diagnosis in digital systems," in *Proc. FTCS-7*, June 1977, pp. 72- 77.

[5] J. G. Kuhl, S. M. Reddy, "Distributed diagnosis in fully distributed systems," in *Proc. 7.th Symp. Comput. Architecture*, 1980, pp. 23 - 30.

[6] J. G. Kuhl, S. M. Reddy, "Fault diagnosis in fully distributed systems," in *Proc. FTCS-11*, June 1981, pp. 100-105.

[7] S. H. Hosseini, J. G. Kuhl, S. M. Reddy, "On self-diagnosis of the distributed systems," *IEEE Trans. Comput.*, vol. C-36, pp. 1378 - 1382, 1987.

[8] C. S. Holt, J. E. Smith, "Self-diagnosis in distributed systems," *IEEE Trans. Comput.*, vol. C-34, pp. 19 - 31, 1985.

[9] S. E. Kreutzer, S. L. Hakimi, "Distributed diagnosis and the system user, " *IEEE Trans. Comput.*, vol. C-37, pp. 71 - 78, 1988.

[10] A. K. Somani, V. K. Agarwal, "System-level diagnosis in systolic systems,". in *Proc. 1984 ICCD-84*, 1984, pp. 445 - 450.

[11] A. K. Somani, V. K. Agarwal, "Distributed diagnosis algorithms for regular interconnected structures," *IEEE Trans. Comput.*, vol. C-41, pp. 899 - 906, 1992.

[12] R. P. Bianchini, R. W. Somani, "Implementation of on-line distributed system-level diagnosis theory," *IEEE Trans. Comput.*, vol. C-41, pp. 616 - 626, 1992.

[13] U. Manber, "Diagnosability with repair," *IEEE Trans. Comput.*, vol. C-29, pp. 934 - 937, 1980.

[14] C. L. Liu, P. Maestrini, "On the sequential diagnosability of a class of digital systems," In *Proc. FTCS-11*, June 1981, pp 112 - 115.

[15] P. Maestrini, P. Santi, "Self-diagnosis of regularly interconnected systems based on a comparison model," TR-3/93, Dipartimento di Informatica, Università di Pisa, Jan. 1994.