Efficient Algorithms for System Level Diagnosis of Multiprocessors Using Local Information

T. Bartha, E. Selényi

Technical University of Budapest
Department of Measurement and Instrument Engineering
Műegyetem rkp. 9. R/113, H-1521 Budapest, Hungary, bartha@mmt.bme.hu

Abstract. Massively parallel computers introduce new requirements for system-level fault diagnosis, like handling a huge number of processing elements in an inhomogeneous system. They also have specific attributes, such as regular topology and low local complexity. Traditional deterministic methods of system-level diagnosis did not consider these issues. This paper presents a new approach, called *local information diagnosis* that exploits the characteristics of massively parallel systems. The paper defines the diagnostic model, which is based on generalized test invalidation to handle inhomogeneity in multiprocessors. Five effective probabilistic diagnostic algorithms using the proposed method are also given, and their space and time complexity is estimated.

Keywords: massively parallel systems, regular interconnection structure, system-level diagnosis, inhomogeneous systems, generalized test invalidation

1 Introduction

The price to performance ratio of microprocessor components is decreasing rapidly. This tendency advances the development of *massively parallel* computers (MPCs), that deliver much larger processing power than single processor systems. MPCs are designed to operate even several thousand processing elements concurrently. Although modern electronic components have a very low permanent fault rate, the probability of an error in an MPC is significant due to the large number of components and long continuous operation. Fault tolerance provides means for keeping the system service uninterrupted by tolerating the effects of occurring errors.

Automatic fault diagnosis is an important part of multiprocessor fault tolerance. Its task is to locate the faulty units in the system. Once these units were identified, the system is reconfigured to logically remove them from operation. In *system-level diagnosis*, processors execute tests on other processors to detect possible errors according to a predefined schedule. When a test fails, the diagnostic algorithm classifies the fault state of processors by analyzing the collection of all test results [10].

2 Generalized theory of test invalidation

A processor can be either faulty or fault-free, and it may be testing another fault-free or faulty unit. The test results so obtained can be interpreted according to various test invalidation models. Many different test invalidation models exist in the literature. The most widely used ones are the *symmetric invalidation* model or PMC model [11], and the *asymmetric invalidation* model or BGM model [3]. In symmetric invalidation a test by a fault-free unit indicates the exact state of the tested unit (complete test assumption). On the other hand, a faulty tester produces an arbitrary test result, irrespective to the state of the tested unit. The asymmetric invalidation scheme is similar, but assumes that two units fail identically with a negligible probability so the test on a faulty unit by a faulty tester practically will always fail.

Lately *inhomogeneous systems*, consisting of different units and test invalidations have gained more practical importance. Algorithms based on *generalized invalidation models* can take this inhomogeneity into consideration. A complete description and analysis of generalized test invalidation can be found in [12]. The model can be outlined by Table 1.

According to the model, fault-free units always test other units correctly (i.e. tests are assumed to have 100 percent coverage). Test results for a faulty tester unit can have three values: always pass, always fail, or either

Table 1. Generalized test invalidation

Tester unit	Tested unit	Test result
fault-free	fault-free	pass
fault-free	faulty	fail
faulty	fault-free	$C \in$ {pass, fail, or non-deterministic}
faulty	faulty	$D \in$ {pass, fail, or non-deterministic}

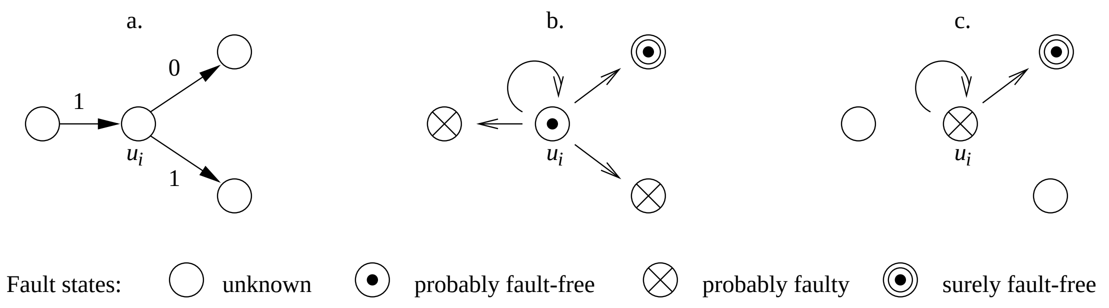

Figure 1. Example system with asymmetric invalidation: (a) Test outcomes, (b) Inferences, supposing that u_i is fault-free, (c) Inferences, supposing that u_i is faulty

pass or fail independent on the fault state of the tested unit. These results are denoted by the constants 0, 1, or X; respectively. Nine possible test invalidations can be derived from the model according to the actual C and D values, denoted by T_{CD}. Thus, symmetric invalidation corresponds to T_{XX} in the model, and asymmetric invalidation to $\mathrm{T}_{\mathrm{X}1}$ [8].

In an inhomogeneous system, the relationship between a tester and a tested unit is affected by three factors:

1. the test invalidation of both units,
2. the (supposed) fault state of both units, and
3. the actual test outcome.

This relationship can be expressed in the form of *one-step implication rules.* Four types of one-step implication rules exists: tautology, forward implication, backward implication, and contradiction. Two one-step implications involving the same unit can be combined using the transitive property: $a \to b \wedge b \to c \Rightarrow a \to c$. The set of all one-step and higher order implications obtained by repeated application of the transitive property is the *transitive closure.* A unit can be surely classified in two cases: when a contradiction exists between the supposed and implied fault states of the unit, or when an implication leads to the unit from an already surely classified unit. For units that cannot be surely classified additional information is required to make their diagnosis possible (e.g. an upper bound on the number of faulty units, exclusion of certain fault sets, etc.)

3 Diagnostic model

The assumptions on the fault model used in this paper are similar to those of the traditional approaches to system-level diagnosis [2]:

- the system consists of intelligent units, that test other units individually and completely,
- the test assignment is predefined,
- only normal interconnection facilities can be used for diagnostic purposes,
- faults are permanent,
- faults in the interconnection network are not considered,
- the test invalidation in the system conforms to the generalized model,
- the diagnosis is centralized.

The system includes a set of $u_i \in U$ units ($i = 1, 2, \ldots, n$), connected by a set of $v_j \in V$ interconnection facilities, or links ($j = 1, 2, \ldots, m$). The units and links form a graph $S = (U, V)$. The complete collection of test assignments is a digraph $T = (U, E)$, where $E \subseteq V$ contains the set of $t_{ij} = (u_i, u_j)$ tests between units u_i and u_j. Two sets can be associated with each u_i unit:

- the set of units tested by u_i: $\Gamma(u_i) = \{u_j | t_{ij} \in E\}$, and
- the set of testers of u_i: $\Gamma^{-1}(u_i) = \{u_j | t_{ji} \in E\}$.

The union of tested and tester units is the set of neighbors $N(u_i) = \Gamma(u_i) \cup \Gamma^{-1}(u_i)$. The set of units, that are reachable from u_i via directed edge sequences consisting of at most k edges are called the k-neighbors: $N_k(u_i)$. The cardinality of these sets are denoted by $\nu(u_i)$ and $\nu_k(u_i)$, respectively. Edges of the T digraph or *testing graph* are labeled by the $a_{ij} \in A$ test results. The A set of test results is called the *syndrome.*

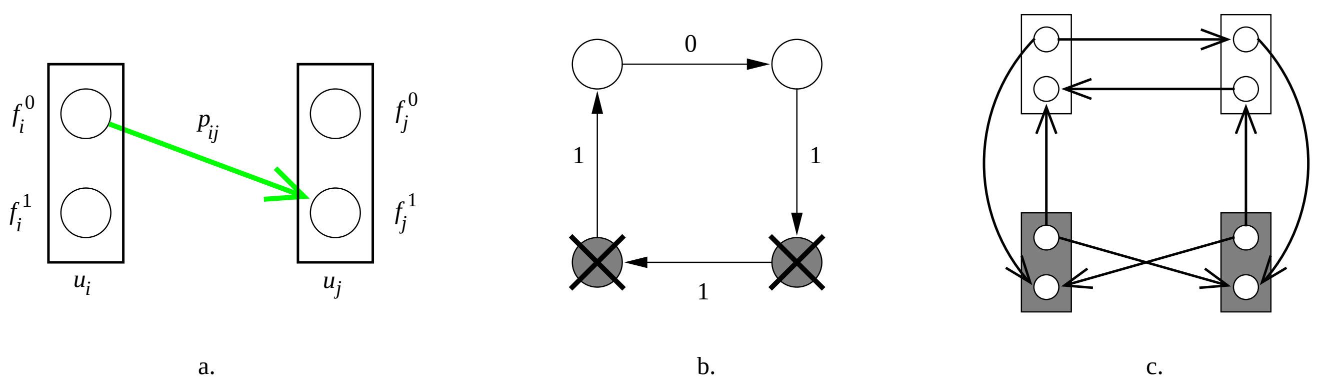

Figure 2. (a) Components of the inference graph, (b) Example test assignment (symmetric invalidation), (c) Corresponding inference graph

Diagnostic inferences also have a digraph form. The inference graph $I = (U, \mathcal{F}, \mathcal{P})$ is composed of the U set of units, the set of $f_i^{\{0,1\}} \in F$ possible fault states ($i = 1, 2, \ldots, n$), and the set of $p_{ij} \in \mathcal{P}$ one-step implications, derived from the actual syndrome. Each unit u_i is associated with two possible fault states: f_i^0 symbolizes the fault-free, f_i^1 the faulty state of the unit. In the graphical representation units, states and implications correspond to boxes, nodes and directed edges, respectively.

4 Concept of local information diagnosis

During the development or evaluation of a diagnostic algorithm the following main characteristics must be considered:

- What time and space complexity does the algorithm have?
- Does it constrain the number of faulty units?
- Can it take advantage of the small local complexity in regular topologies?
- Is it applicable to inhomogeneous systems as well?

Generally, the diagnostic procedure is intended to be executed in background using as little amount of resources as possible. At the same time it has to be fast and efficient to minimize fault latency. Therefore in large, complex systems like MPCs time and space complexity gains additional importance. Unfortunately, the problem of system level diagnosis in its most general form is NP-complete [9]. Better results could be produced only in restricted cases. For instance, in the case of t-diagnosable systems with symmetric invalidation the best existing method in terms of worst-case efficiency has $O(n^{2.5})$ time complexity [6].

Many traditional diagnostic algorithms employ a static t-limit on the number of tolerated faulty units. Yet, the t-limit is a very pessimistic restriction in large systems, as a significant amount of fault sets consisting of more than t faulty units are diagnosable [13]. Moreover, this amount increases proportionally with the total number of processors in the system. Two main situations are of interest:

1. the faults are scattered, separated from each other, or

2. the faults are close to each other in a group.

General fault sets can be considered as the combination of these two special cases. The first situation is simple from the diagnostic viewpoint: the faulty units are surrounded by fault-free testers, and hence there is enough information available in the syndrome to identify the faulty units. In the second situation, however, when diagnostic uncertainty is present no algorithm can assure the correct classification of the faults *inside* the group [12]. For this reason, a proper diagnostic method must (and can) guarantee the identification of the units only on the border of the fault group. Logically disconnecting these units the faulty units within the area bounded by the group are automatically isolated as well [1].

In most practical cases the above two situations can be handled using just a portion of the diagnostic information [5]. If faults are separated, the failed test results appear locally in the syndrome. The diagnosis of fault group borders is similar. It is enough to examine only a small environment of a unit to classify it, that is in many situations the implication chains do not need to be evaluated in full length. Limiting the calculation of inferences is the main idea of local information diagnosis. Naturally, this way only a partial diagnostic image can be obtained, therefore correct and complete diagnosis can be provided only with high probability. On the other hand, the approach has a small complexity and no restrictions need to be placed on the structure and size of the system and the possible fault sets.

5 Diagnostic algorithms

In this section we present five local information, system-level diagnostic algorithms. They are ranged in three categories.

5.1 Scalar algorithms

Scalar algorithms compute and utilize the quantity of implications ending in a given fault state, i.e. they do not keep a record about the source nodes of implication chains. This is advantageous from the efficiency viewpoint, but it obviously results in an additional loss of diagnostic information. The relationship of non-neighbor units cannot be determined, therefore some heuristics (like a weight function) must be employed in these methods.

Count Failed Test (CFT) algorithm

The CFT algorithm (see Figure 3) extends the idea of the diagnosis method presented by Dahbura et al. in [7]. It simply counts the number of test failed on each unit. Until there are failed tests, the algorithm selects the $u_m \in U$ unit with maximal $\mathrm{C_{FT}}[m]$ (failed tests count) value. If multiple units have maximal $\mathrm{C_{FT}}[m]$ values, then the one having a minimal $\mathrm{C_{FTT}}[p]$ (tester failed tests count) value will be selected. The $\mathrm{C_{FTT}}[p]$ value is the sum of all failed tests on the testers of the unit u_p. The u_m unit is then added to the Φ set of faulty units. The unit and its test results are removed from the $\mathrm{C_{FT}}$ and $\mathrm{C_{FTT}}$ arrays. When there are no more failed tests the remaining units are classified as fault-free.

Count Inference Paths (CIP) algorithm

The CIP algorithm uses a different approach [4]. Here the likelihood of a fault state is estimated by the number of implications supporting it. For this purpose the algorithm maintains two counters for each unit: $\Sigma^0[i]$ and $\Sigma^1[i]$, corresponding to the weighted number of paths leading to the fault-free state f_i^0 and the faulty state f_i^1 in the inference graph. The $\Sigma^0[i]$ and $\Sigma^1[i]$ numbers are calculated iteratively (see Figure 4). In the first step only the one-step implications are considered. In case of one-step sure inferences are available they are used to surely classify the respective units. Without sure inferences the number of the implications is counted. This number is multiplied by the W weight function, so additional information on the fault model can be included in diagnosis. Subsequently all paths of length $2, 3, \ldots, k$ are added to the calculation. The set of surely classified units is also extended using the implications drawn from already surely classified fault states. After the kth step the remaining unclassified units are diagnosed as faulty if $\Sigma^1[i] > \Sigma^0[i]$, otherwise they are assumed to be fault-free.

5.2 Limited inference algorithms

Limited inference algorithms do not lose diagnostic information at the representation of inferences, as they store every implication connecting different fault states. These methods compute implication chains only in a limited, predetermined length and classify the units on the basis of this incomplete diagnostic image.

Limited Multiplication of Inference Matrix (LMIM) algorithm

The LMIM algorithm is a simplified variant of the Selényi algorithm described in [12]. One-step implications are stored in an $2n \times 2n$ $\mathbf{M}$ hypermatrix. The $\mathbf{M}$ matrix consists of four $n \times n$ binary minormatrices: $\mathbf{M}^{00}$, $\mathbf{M}^{01}$, $\mathbf{M}^{10}$, and $\mathbf{M}^{11}$. The $\mathrm{m}^{xy}[i,j]$ element of the $\mathbf{M}^{xy}$ minormatrix ($x, y \in \{0,1\}$) is 1, if there exists an $f_i^x \to f_j^y$ one-step implication between units u_i and u_j.

Transitive closure can be computed by the logical closure of the $\mathbf{M}$ matrix. In the LMIM algorithm, transitive closure is estimated by raising the $\mathbf{M}$ matrix only to a small power. Thus, it will not contain all the inferences included in the syndrome. Units corresponding to "1"s in the main diagonal of $\mathbf{M}^{01}$ and $\mathbf{M}^{10}$ are involved in a contradiction, and so they are surely classified. For other u_i units, the sum of $f_j^0 \to f_i^0$ and $f_j^0 \to f_i^1$ implications is counted. If $\sum_j \mathrm{m}^{01}[j,i] > \sum_j \mathrm{m}^{00}[j,i]$ the unit is diagnosed as faulty, otherwise it is fault-free.

Distribution of Inference Lists (DIL) algorithm

The DIL algorithm extracts the same amount of diagnostic information from the syndrome as the LMIM method. However, it has a better time complexity, since it exploits the properties of regular topologies. The main structure of the algorithm is very similar to the CIP algorithm, only the stored data and the update method differ (see Figure 4). For every u_i unit four binary vectors: $\mathbf{m}^{00}[i]$, $\mathbf{m}^{01}[i]$, $\mathbf{m}^{10}[i]$, and $\mathbf{m}^{11}[i]$ contain the set of one-step implications on u_i (these vectors equal to the rows of the respective minormatrices in $\mathbf{M}$). In every step of the iteration the set of implications is expanded transitively. At the end of the process, the ith components of the $\mathbf{m}^{01}[i]$ and $\mathbf{m}^{10}[i]$ vectors indicate contradiction if sure classification is possible. Unclassified nodes are diagnosed as faulty, if $\sum_j \mathbf{m}^{01}[i][j] > \sum_j \mathbf{m}^{00}[i][j]$.

```
Algorithm CFT
{ initialization }
for each $u_i \in U$ do
    $\mathrm{C_{FT}}[i] \leftarrow \sum_j a_{ji}, u_j \in \Gamma^{-1}(u_i)$
    $\mathrm{C_{FTT}}[i] \leftarrow \sum_{j,k} a_{ji}, u_j \in \Gamma^{-1}(u_i), u_k \in \Gamma^{-1}(u_j)$
end for
{ classification }
while $\exists j, \mathrm{C_{FT}}[j] > 0$ do
    find $u_m$ with:
        maximal $\mathrm{C_{FT}}[m]$,
        minimal $\mathrm{C_{FTT}}[m]$
    $\Phi \leftarrow \Phi \cup u_m$
    remove $u_m$ and its test results
    from $\mathrm{C_{FT}}$ and $\mathrm{C_{FTT}}$
end while
```

```
Algorithm CIP
{ initialization }
for each $u_i \in U$ do
    for each $p_{ij}, u_j \in N(u_i)$ do
        if $p_{ij}$ is contradiction then
            surely classify $u_i$
        else $P[i] \leftarrow P[i] \cup p_{ij}$
    end for
end for
{ count inference paths }
for each iteration do
    for each $u_i \in U$ do
        for each $p_{ij} \in P[i]$ do
            if $u_i$ is surely classified then
                surely classify $u_j$
            else Update $\Sigma^0[i], \Sigma^1[i]$
        end for
    end for
end for
{ classification }
for each unclassified $u_i$ do
    if $\Sigma^1[i] > \Sigma^0[i]$ then
        $\Phi \leftarrow \Phi \cup u_i$
end for
```

Figure 3. The CFT and CIP algorithms

```
Algorithm Update $\Sigma^0[i], \Sigma^1[i]$
{ initialization } for each $u_i \in U$ do
    $\Sigma^0[i] \leftarrow 0, \mathrm{s}^0[i] \leftarrow 0, \mathrm{t}^0[i] \leftarrow -1$
    $\Sigma^1[i] \leftarrow 0, \mathrm{s}^1[i] \leftarrow 0, \mathrm{t}^1[i] \leftarrow -1$
end for
{ update $\Sigma^0[i], \Sigma^1[i]$ }
for each iteration do
    for each $u_i \in U$ do
        for each $p_{ij} \in P[i]$ do
            if $p_{ij} : f_i^0 \rightarrow f_j^0$ then
                $\Sigma^0[j] \leftarrow \Sigma^0[j] + W(\mathrm{s}^0[i] - \mathrm{t}^0[i])$
            else if $p_{ij} : f_i^0 \rightarrow f_j^1$ then
                $\Sigma^1[j] \leftarrow \Sigma^1[j] + W(\mathrm{s}^0[i] - \mathrm{t}^0[i])$
            else if $p_{ij} : f_i^1 \rightarrow f_j^0$ then
                $\Sigma^0[j] \leftarrow \Sigma^0[j] + W(\mathrm{s}^1[i] - \mathrm{t}^1[i])$
            else if $p_{ij} : f_i^1 \rightarrow f_j^1$ then
                $\Sigma^1[j] \leftarrow \Sigma^1[j] + W(\mathrm{s}^1[i] - \mathrm{t}^1[i])$
        end for
    end for
    for each $u_i \in U$ do
        $\mathrm{t}^0[i] \leftarrow \mathrm{s}^0[i], \mathrm{s}^0[i] \leftarrow \Sigma^0[i]$
        $\mathrm{t}^1[i] \leftarrow \mathrm{s}^1[i], \mathrm{s}^1[i] \leftarrow \Sigma^1[i]$
    end for
end for
```

```
Algorithm Update $\mathrm{m}^{00}, \mathrm{m}^{01}, \mathrm{m}^{10}, \mathrm{m}^{11}$
{ initialization }
for each $u_i \in U$ do
    $\mathrm{m}^{00}[i] \leftarrow u_j, \exists p_{ji} : f_j^0 \rightarrow f_i^0$
    $\mathrm{m}^{01}[i] \leftarrow u_j, \exists p_{ji} : f_j^0 \rightarrow f_i^1$
    $\mathrm{m}^{10}[i] \leftarrow u_j, \exists p_{ji} : f_j^1 \rightarrow f_i^0$
    $\mathrm{m}^{11}[i] \leftarrow u_j, \exists p_{ji} : f_j^1 \rightarrow f_i^1$
end for
{ update $\mathrm{m}^{00}, \mathrm{m}^{01}, \mathrm{m}^{10}, \mathrm{m}^{11}$ }
for each iteration do
    for each $u_i \in U$ do
        for each $p_{ij} \in P[i]$ do
            if $p_{ij} : f_i^0 \rightarrow f_j^0$ then
                $\mathrm{m}^{00}[j] \leftarrow \mathrm{m}^{00}[j] \cup \mathrm{m}^{00}[i]$
                $\mathrm{m}^{10}[j] \leftarrow \mathrm{m}^{10}[j] \cup \mathrm{m}^{10}[i]$
            else if $p_{ij} : f_i^0 \rightarrow f_j^1$ then
                $\mathrm{m}^{01}[j] \leftarrow \mathrm{m}^{01}[j] \cup \mathrm{m}^{00}[i]$
                $\mathrm{m}^{11}[j] \leftarrow \mathrm{m}^{11}[j] \cup \mathrm{m}^{10}[i]$
            else if $p_{ij} : f_i^1 \rightarrow f_j^0$ then
                $\mathrm{m}^{00}[j] \leftarrow \mathrm{m}^{00}[j] \cup \mathrm{m}^{01}[i]$
                $\mathrm{m}^{10}[j] \leftarrow \mathrm{m}^{10}[j] \cup \mathrm{m}^{11}[i]$
            else if $p_{ij} : f_i^1 \rightarrow f_j^1$ then
                $\mathrm{m}^{01}[j] \leftarrow \mathrm{m}^{01}[j] \cup \mathrm{m}^{01}[i]$
                $\mathrm{m}^{11}[j] \leftarrow \mathrm{m}^{11}[j] \cup \mathrm{m}^{11}[i]$
        end for
    end for
end for
```

Figure 4. The information update procedures used in the CIP and DIL algorithms

Table 2. Characteristics of the presented algorithms

Algorithm:	**CFT**	**CIP**	**LMIM**	**DIL**	**LTC**
Type:	scalar	scalar	limited inference	limited inference	limited information
Time complexity	$O(\nu\,(n+c\phi)\,)$	$O(n\nu)$	$O(n^3)$	$O(n\nu)$	$O(n\nu_k^3)$
Space complexity	$O(n)$	$O(n)$	$O(n^2)$	$O(n^2)$	$O(\nu_k^2)$
Diagnosable fault sets	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Invalidation model	symmetric	general	general	general	general
Topology	regular	general	general	general	general
Exploitation of regularity?	yes	yes	no	yes	yes
Inhomogeneous systems?	no	yes	yes	yes	yes

5.3 Limited information

The limited information approach calculates a complete transitive closure, but only in a small local environment of the diagnosed units.

Local Transitive Closure (LTC) algorithm

The LTC algorithm uses a different approach to utilize regularity. Unlike the previous methods, it limits the amount of diagnostic information involved in a decision, instead of the length of the considered inference chains. During the diagnosis of every u_i unit a $2\nu_k \times 2\nu_k$ $\mathbf{M}_k(U_i)$ hypermatrix is created. In the $\mathbf{M}_k(u_i)$ matrix only the k-neighbors are included. Then, the $\mathbf{M}_k(u_i)$ matrix is logically closed, resulting another estimation of the transitive closure from the viewpoint of the unit u_i. Therefore, the closed matrix is used exclusively for the diagnosis of u_i in the following way: sure classification is indicated by the corresponding elements of the $\mathbf{M}_k^{01}(u_i)$, $\mathbf{M}_k^{10}(u_i)$ minormatrices. If there is no contradiction, the u_i is diagnosed as faulty if $\sum_j \mathbf{M}_k^{10}(u_i)[j,i] > \sum_j \mathbf{M}_k^{00}(u_i)[j,i]$.

6 Performance

Table 2 summarizes the main characteristics of the presented algorithms:

Time complexity. The CIP and DIL algorithms exchange information locally, their complexity depend on the $\nu = \max_i \nu(u_i)$ number of neighboring units. The complexity of the CFT algorithm is additionally determined by the ϕ number of faulty units. The complexity of the LTC algorithm depends only on the $\nu_k = \max_i \nu_k(u_i)$ number of k-neighbors examined. Note, that the ν and ν_k values are constant in the function of system size, so the CFT, CIP, DIL, and LTC algorithms have essentially $O(n)$ time complexity. The only exception is the LMIM algorithm, its relatively low performance results from its topology independence.

Space complexity. The values reflect the amount of data stored for diagnostic purpose, not including the syndrome. For large systems, and in case of small number of considered k-neighbors the LTC algorithm has the best space complexity.

Number of diagnosable fault sets. An important feature of the algorithms, that they do not have a predefined t-limit on the number of faults. Thus, the number of fault sets correctly diagnosed increases proportionally with the system size, provided that the amount of faulty units is relatively low (less than 25 percent of the fault-free units).

Other characteristics. Most of the algorithms employ the generalized test invalidation model, and work with any system topology. However, the CFT method uses only the assumptions of symmetric invalidation, therefore its performance does not improve in the case of less restricted invalidation models. It also requires a regular topology for correct operation.

The presented methods are implemented in a simulation environment. We examined the correctness of the algorithms for the two specific fault configurations mentioned in Section 4. The measurements were performed on a 2-dimensional toroid mesh topology containing 8×8 processing elements. For analyzing group faults we used the fault pattern shown in Figure 5 (some interconnection links are omitted from the figure for simplicity). Scattered fault patterns were generated by randomly placing 4 faulty units in the system. The measurement results for all of the nonequivalent test invalidation models [8] are included in Table 3.

In the table, row (a) contains for each algorithm the number of misdiagnosed units on the fault group borders (the considered units are enclosed in a dashed rectangle on Figure 5) as the percentage of the amount of faults. None of the algorithms did misdiagnose any units in this area. However, inside the fault group there were misdiagnosed units in some cases due to diagnostic uncertainty. These results can be found in row (b). The percentage of

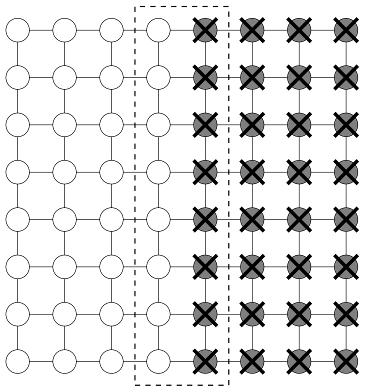

Figure 5. Fault pattern to examine diagnosis on the fault group borders

Table 3. Percentage of misdiagnosed units: (a) On fault group borders, (b) Inside fault groups, (c) Scattered faults

Algorithm	Case	**Invalidation model**						
		T_{00}	T_{01}	T_{0X}	T_{11}	T_{1X}	T_{X1}	T_{XX}
CFT	a.	0	0	0	0	0	0	0
	b.	50	25	19	19	19	21	20
	c.	0	0	0	0	0	0	0
CIP	a.	0	0	0	0	0	0	0
	b.	0	0	< 1	0	< 1	0	2
	c.	0	0	0	0	0	0	0
LMIM	a.	0	0	0	0	0	0	0
	b.	0	0	< 1	0	< 1	0	4
	c.	0	0	0	0	0	0	0
DIL	a.	0	0	0	0	0	0	0
	b.	0	0	< 1	0	< 1	0	4
	c.	0	0	0	0	0	0	0
LTC	a.	0	0	0	0	0	0	0
	b.	0	0	< 1	0	6	0	19
	c.	0	0	0	0	0	0	0

misdiagnosed units in scattered fault situations is shown in row (c). Every algorithm did produce a correct and complete diagnosis when the faults were separated from each other.

7 Conclusions

This paper introduced the concept of local information diagnosis, a novel approach to efficient system-level diagnosis of massively parallel systems. It accepts many of the assumptions of traditional methods, but uses a generalized test invalidation model to comply with inhomogeneous models as well. The approach estimates the transitive closure by evaluating the implication chains in the inference graph only in a limited length. The detected contradictions serve as a sure classification of the given unit. Other units without sure classification are diagnosed on the basis the number of inferences supporting the respective fault state hypotheses.

The paper also presented five algorithms to illustrate the main characteristics of local information diagnosis. Their time and space complexity, and the number of tolerated faults was included in the characteristics. Taking only local information into consideration the algorithms provide fast and efficient diagnosis even in large multiprocessor systems. We are currently working on making further measurements with the implemented algorithms to get more detailed information about their performance and limitations; as well as on the theoretical analysis of the practical and asymptotic behavior of the algorithms.

References

[1] J. Altmann, T. Bartha, and A. Pataricza. On integrating error detection into a fault diagnosis algorithm for massively parallel computers. In *IEEE Int. Computer Performance and Dependability Symp.*, pages 154–164. IEEE Computer Society, 1995.

[2] M. Barborak, M. Malek, and A. Dahbura. The consensus problem in fault-tolerant computing. *ACM Comput. Surveys*, 25(2):171–220, March 1993.

[3] F. Barsi, F. Grandolini, and P. Maestrini. A theory of diagnosability of digital systems. *IEEE Trans. Computers*, C-25(6):585–593, June 1976.

[4] T. Bartha. Effective approximate fault diagnosis of systems with inhomogeneous test invalidation. In *22nd Euromicro Conf.*, 1996. Accepted for publication.

[5] D. Blough, G. Sullivan, and G. Masson. Fault diagnosis for sparsely interconnected multiprocessor systems. In *19th Int. IEEE Symp. on fault-Tolerant Computing*, pages 62–69. IEEE Computer Society, 1989.

[6] A. Dahbura and G. Masson. An $O(n^{2.5})$ fault identification algorithm fo diagnosable systems. In *14th Int. IEEE Symp. on Fault-Tolerant Computing*, pages 428–433. IEEE Computer Society, 1984.

[7] A. Dahbura, K. Sabnani, and L. King. The comparison approach to multiprocessor fault diagnosis. *IEEE Trans. Computers*, C-36(3):373–378, March 1987.

[8] A. D. Friedman and L. Simoncini. System-level fault diagnosis. *IEEE Computer*, 13(3):47–53, March 1980.

[9] H. Fujiwara and K. Kinoshita. On the computational complexity of system diagnosis. *IEEE Trans. Computers*, C-27(10):881–885, October 1978.

[10] C. Kime. System diagnosis. In D. Pradhan, editor, *Fault-Tolerant Computing: Theory and Techniques*, volume 2, chapter 8, pages 577–623. Prentice-Hall, Englewood Cliffs, N. J., 1985.

[11] F. Preparata, G. Metze, and R. Chien. On the connection assignment problem of diagnosable systems. *IEEE Trans. Electronic Computers*, EC-16(6):848–854, December 1967.

[12] E. Selényi. *Generalization of System-Level Diagnosis.* D. Sc. thesis, Hungarian Academy of Sciences, Budapest, 1984.

[13] A. K. Somani, V. K. Agarwal, and D. Avis. Generalized theory for system level diagnosis. *IEEE Trans. Computers*, C-36(5):538–546, May 1987.