DIAGNOSTIKA A ZABEZPEČENÍ ČÍSLICOVÝCH SYSTÉMŮ

1981

SELF-DIAGNOSIS IN MULTIMICROPROCESSOR SYSTEMS: THE MuTEAM APPROACH

P. Ciompi, F. Grandoni, L. Simoncini

Istituto di Elaborazione della Informazione
Via S.Maria 46, 56100 Pisa, Italy

Introduction

The requirement of high dependability is one of the most important issues which has been raised by the introduction of complex computing systems and by their use in real time application control.

In particular, attention has to be focused on the need of suitable and powerful diagnostic tools for enhancing the maintainability of the system.

These concepts represent the guideline of the MuTEAM project. MuTEAM is an experimental, distributed multimicroprocessor system developed for the National Computer Science Program of the Italian National Research Council. Two companion papers /1,2/ deal with the architectural aspects and operating system kernel of the MuTEAM while in this paper a proposal for embedding self-diagnostic facilities in a multimicroprocessor system and the underlying fault treatment strategy of the MuTEAM is presented.

The interest in a self-diagnostic approach is motivated by the fact that: a) it can be made modular and can be fitted to the actual resources in the system; b) it is functionally distributed, since no centralized control is strictly required; c) constitutes a valid solution for an efficient testing by conveniently exploiting the parallelism of multiprocessor systems; d) can be implemented at a sufficiently high level of abstraction and therefore can be considered independent by the used technologies; e) can be used for a periodic diagnosis as an easy maintenance aid, to pinpoint errors caused by latent faults to avoid a possible catastrophic failure of the system caused by the superimposition of many undetected faults; f) can be considered an aid to the online fault detection and diagnosis without the need of interrupting the normal computation of the system (concurrent diagnosis /3/); g) they can be fitted to degraded configurations of the system without a complete loss of diagnostic power.

Problems in the implementation of self diagnostic procedures

Several abstract models of self-diagnosis have been proposed and established theoretical results are present in the literature /4/. All the proposed models rely on the outline of the diagnostic characteristics of a system through a diagnostic graph. In such graph, in the models which are more investigated, the nodes represent the units u_i of the system and the arcs the testing relation among them. An arc from u_i

to u_j means that u_i is able to apply a diagnostic test to u_j and judge about its status (faulty or fault-free). Moreover the meaning of an arc from u_i to u_j is that the test result and judgment can be only influenced by malfunctions in u_i and u_j and not by the malfunction of another unit. By decoding the ordered set of test results (syndrome) the actual faulty units in the system can be identified.

These models, by their nature, rely on simple hypotheses which determine several problems on the applicability in real systems. In particular an actual self diagnostic procedure is organized in a temporal succession of phases and has to run in a potentially faulty environment.

The solution of dedicating a centralized unit, like a support processor /5/, in our opinion has to be avoided since it introduces a singular point which has to constitute the hard-core of the system /6/. Therefore we consider that the general organization of a diagnostic procedure is based on a set of asynchronous cooperating processes, all at the same level, without any master-slave relation among them, and without any centralized hard-core.

The problems, which arise for the actual implementation in a multiprocessor environment, are the following:

1 - During the test execution it is required that no unit u_i can act as a master for the tested unit u_j for avoiding that a malfunction of u_i makes definitely slave the fault-free u_j.

Therefore each unit has to maintain its autonomy even during the test execution by basing any interaction among units on explicit requests of actions and explicit acknowledgements of execution of the requested actions.

2 - Each unit, during the procedure, has to take both the statuses of "testing" and "under test". Since the diagnostic graph contains at least one closed loop, deadlock situations must be avoided in which the units in the diagnostic loop either are in the same status or oscillate between the two possible statuses for an indefinitely long time.

3 - The diagnostic process requires that the judgment of u_i about the status of u_j is not invalidated by any other unit. This determines the need of protected communication channels between units.

4 - As it is possible that either a faulty unit u_j does not makes available its test results to u_i or that, a fault-free unit u_k is not yet ready to make its results available to u_i, the problem of the correct decoding of an incomplete syndrome must be solved.

Implementation of diagnostic processes and influence on the architecture

Let us consider the flow diagram of a diagnostic process, as outlined in Fig. 1.

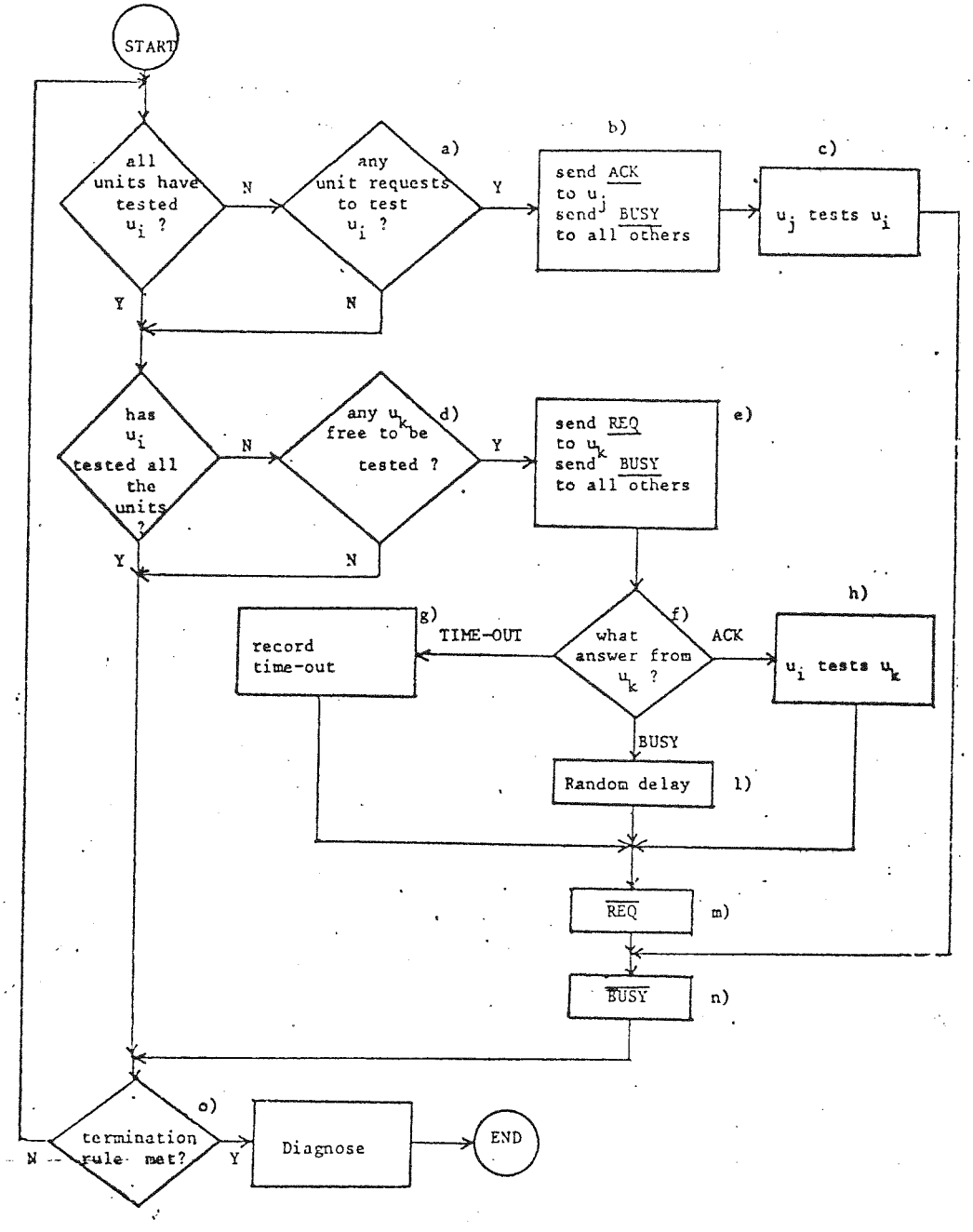

FIG. 1

Let u_k be one of the units which have to be tested by u_i and u_j be one of the units which have to test u_i.

If u_j request the test of u_i (block a), u_i sends an acknowledge to u_j and a busy to all the others (block b). The test from u_j on u_i is

executed (block c) and after that a not busy is sent (block n). If no request is pending, u_i tries to test u_k (block d). If u_k is not busy u_i send to it a request to u_k and a busy to u_j (block e). u_k must answer (block f) to the request from u_i with an acknowledge or a busy in a finite time which can be determined. This basilar conversation is used to avoid the master-slave relation among the units.

A time-out will evidence a malfunctioning of the units during this basilar conversation. If a time out is detected, this is recorderd by u_i (block g) and will be used for the termination rule. If u_k answers with an acknowledge, u_i executes the test of u_k (block h). On completion of the test u_i clears its request and sends a not busy (block m,n) to the other units and controls if the overall procedure is terminated (block o). A potential deadlock is present as pointed out in the previous section, if all the units in the loop of the diagnostic graph may have found out that the successors are available to be tested, they may have sent their request and all have got a busy answer. The solutions to this problem which are present in the literature /7,8/, are based on either a unique mutual exclusion semaphore or a unique control procedure. In any case the presence of such a singular point is not suitable in a potentially faulty environment, and therefore another solution is proposed: u_i introduces a random delay which has not to be correlated among the units, before continuing the process loop (block l).

This solution does not insure the deterministic avoidance of deadlock situations but statistics can be derived which relate the deadlock duration with the characteristics of the random delay.

The block o) of the termination rule needs more explanations. The use of a time-out as termination rule has been disregarded due to the difficulty in the evaluation of the maximal time.

A more efficient termination rule is based on the procedure outlined in Fig. 2.

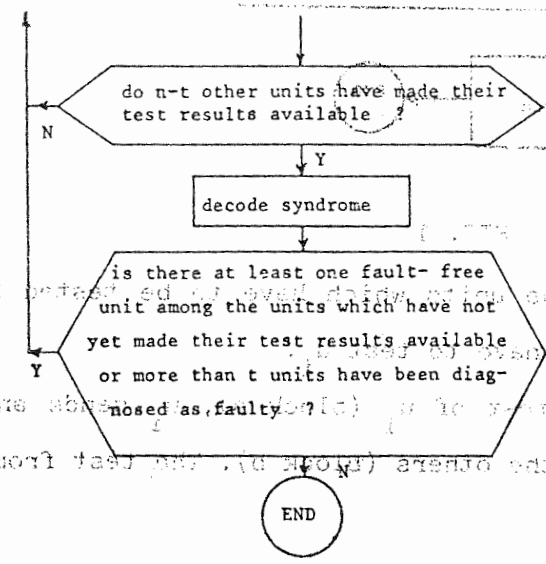

FIG. 2

The unit u_i starts the syndrome decoding as soon as n-t other units have made their results available. In other words, u_i tries to decode a partial syndrome to verify if fault-free units are present among those ones which have not made available their results. If no such unit exists, the syndrome is considered as complete and its decoding correct.

In /9/ it is formally shown that, it is always possible to recognize the existence (if present) of fault-free units among the set of units which have not made available their test results, and that the diagnostic process correctly terminates.

Communication channels must be provided for the diagnostic procedure.

As previously pointed out in Section 2, these channels among the units must be protected. A physical point to point connection among the units can provide the required communication channels as well as the necessary protection among the communication channels. However this solution is expensive and not sufficiently flexible. A more suitable approach may be based on the use of a common bus with a logically segmented shared memory. The access to the memory segments are explicitly enabled by a protection unit which controls the access rights which are assigned to the several units u_i. Possible solutions are shown in Fig. 3.

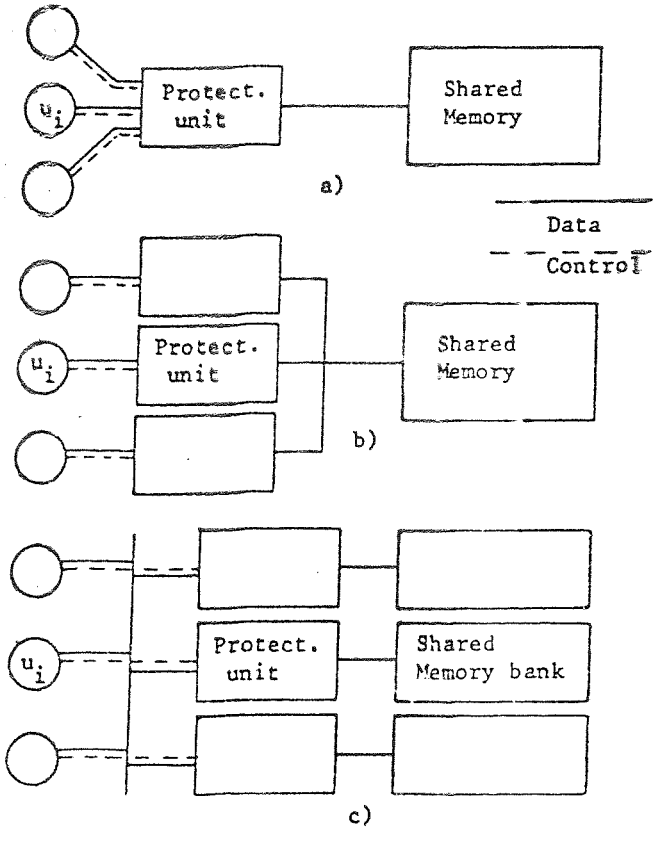

FIG. 3

In Fig. 3a) both the protection units and the shared memory are centralized; therefore this organization is not consistent with the overall approach we have tried to implement. A different solution is presented in Fig. 3b). Even in this case a fault in one of the protection units may allow the arbitrary access from a unit u_i to the shared memory. A more suitable solution is presented in Fig. 3c) where even the shared memory is partitioned in banks protected by their own protection unit. Each protection unit is controlled by the associated unit u_i. A fault in a protection unit can influence only the memory bank which is associated to it and cannot influence all the shared memory.

The dynamic management of the protection units is in charge of the unit u_i, which, if malfunctioning, can influence only the protection unit which is controlled by it.

Conclusion

In this paper an approach to the possible implementation of self-diagnostic techniques in multimicroprocessor systems has been presented. The aim was to incorporate in a system with high degree of dependability, the capability of identifying automatically the faulty units, minimizing the hard-core of the system dedicated to this function.

The diagnostic procedure will be implemented and experimented in the experimental MuTEAM prototype. It will be described using a CSP based language /3/, and by a suitable specification of the operating system kernel primitives which are necessary for the correct and efficient support of the diagnostic procedure.

The experimentation on MuTEAM will deal basically with the efficiency of such procedure implemented at a rather high level of abstraction and will not be focused on the aspects of completeness and complexity of the test of the units.

References

/1/ R. Cardini, M. La Manna, L. Lopriore, L. Strigini, "System Architecture and Protection Mechanisms of the MuTEAM Multimicroprocessor", this issue.

/2/ F. Baiardi, A. Fantechi, A. Tomasi, M. Vanneschi, "Protection and Error Confinement in a Message-Passing Environment: the MuTEAM Kernel", this issue.

/3/ L. Simoncini, F. Saheban, A.D. Friedman, "Design of Self-Diagnosable Multiprocessor Systems with Concurrent Computation and Diagnosis", IEEE Trans. on Computer, Vol. C-29, n° 6, June 1980, pp. 540-546.

/4/ A.D. Friedman, L. Simoncini, "System Level Fault Diagnosis", Computer, Vol. 13, n° 3, March 1980, pp. 47-53.

/5/ D.J. Kunshier, D.R. Mueller, "Support Processor Based System Fault Recovery", Proceedings FTCS-10, Kioto, Japan, Oct. 1-3, 1980, pp. 297-301.

/6/ A.K. Jones, P. Schwarz, "Experiences Using Multiprocessor Systems. A Status Report", ACM Comp. Surveys, 12, 2, June 1980, pp. 121-165.

/7/ E.W. Dijkstra, "Hierarchical Ordering of Sequential Processes", in Operating System Techniques, Academic Press, London 1972.

/8/ C.A.R. Hoare, "Communicating Sequential Processes", CACM, Vol. 21, n° 8, Aug. 1978, pp. 668-679.

/9/ P. Ciompi, F. Grandoni, L. Simoncini, "The MuTEAM Fault Treatment: a Proposal for Self-Diagnosis in Multimicroprocessor Systems", Techn. Rep., MUMICRO Series, June 1981.

This work has been supported by the National Computer Science Program of the Italian National Research Council.