CONCURRENT DIAGNOSTICS IN MULTIPROCESSOR SYSTEMS

J. HLAVIČKA

Department of Computers
Czech Technical University

Received March 5, 1987

Abstract

The paper presents a survey of diagnostic methods for multiprocessor systems. The diagnostic means known so far are first summarized and evaluated from the point of view of their applicability to systems with distributed control and specifically to the multiprocessor systems. A combination of different diagnostic means is then suggested in order to achieve the maximum diagnostic coverage with minimum overhead.

Introduction

Highly integrated microelectronic systems can be used in virtually any control function, especially there, where high performance, reliability or flexibility to changing conditions are required. All these qualities can be achieved above all by the use of multiprocessor systems which were made feasible especially through the miniaturization of electronic components. A multiprocessor system represents a new computing environment which still has to be studied from several points of view. The problems awaiting solution are above all the distribution of control functions, parallel programming, design of interconnection networks and, last but not least, the system diagnostics. Although a large amount of work has been invested into this field, there are still quite a lot of problems to be solved.

Many different methods and techniques of investigating the technical condition have been suggested and tried since the beginnings of the systematic diagnostics of digital systems. Before trying to apply them to the multiprocessor systems, let us first classify them in accordance to some criteria.

Main Types of Diagnostic Procedures

The diagnostic procedures can be classified in accordance with several criteria which are mutually independent. We may e.g. implement the diagnostics by some external means or by devices built directly into the unit-

under-test (UUT). On the other hand, the diagnostics can be based upon the tests performed periodically or upon the continuous evaluation of signals supervising the correct function of the system. These two properties can be interpreted as two independent criteria classifying the diagnostics procedures into four categories according to Table 1. For every cases we can see here also a typical implementation of diagnostic procedure.

Table 1. Classification of Diagnostic Functions

	External	Internal
Periodic	Tester	Maintenance proc.
Concurrent	Duplex, WDP	Error code, SIS

On the vertical edge we can see the classification of diagnostics into periodic and concurrent. The *periodic* diagnostics represent the classical approach to testing. The main goal of the periodic diagnostics is to determine the exact technical condition of the UUT in specified time intervals. These intervals can be specified rigorously in advance, or — which is more frequent — determined in accordance with the system's technical condition and its workload. In the latter case the diagnostic procedure is in fact not periodic, but the term which is due to the standard way of scheduling the maintenance is still being used. The traditional way of performing the periodic diagnostics is by some *external* means, e.g. by a tester. The diagnostics performed by a tester are usually denoted as "off-line diagnostics", because the UUT has to be disconnected from its functional environment and attached to the tester.

The periodic diagnostics can be performed also *internally*, i.e. by some device built directly into the UUT and constituting its integral part. In the context of automatic diagnostics, this role is typically played by a maintenance processor [9]. The maintenance processor continuously connected to the UUT, performs the so-called "on-line diagnostics". Fig. 1 shows the state diagram of on-line diagnostics running under OS Monitor and interleaved with application programs.

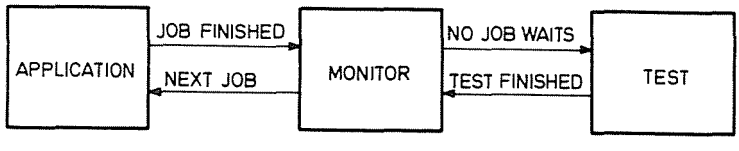

Fig. 1. State diagram of periodic diagnostics controlled by OS Monitor

If the normal function of UUT is not interrupted, the diagnostic procedure is denoted as "in-line". Logically, the in-line diagnostics can be performed at any moment only on some parts of UUT, namely on those which are momentarily not used. In smaller UUT's the function of a maintenance processor is fulfilled by relatively simple devices which are denoted by a global term "built-in self-test equipment" (BISTE). Until recently, the use of a built-in tester was a privilege of programmable systems. However, the recent results in the theory and practice of self-testing made BISTE available literally for any type of digital device, no matter how small or simple it is. A typical case is shown in Fig. 2 where the test generator is linear feedback shift register (LFSR) and the response evaluator is a similar register with parallel inputs (PLFSR).

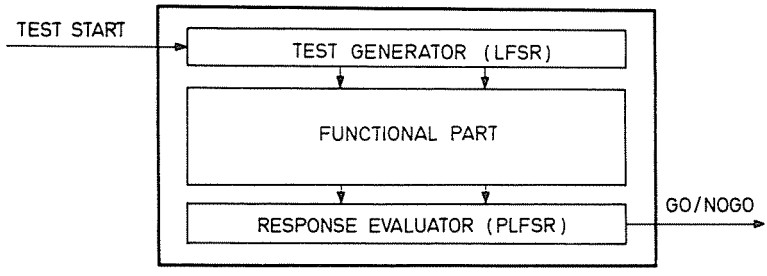

Fig. 2. BISTE for a small UUT

A special case of BISTE implementation which we are going to discuss later is the mutual testing of processors in a multiprocessor system. Here, the role of a built-in tester is cyclically assigned to the processors which otherwise perform the normal computing functions.

The *concurrent* diagnostics, as opposed to the periodic ones, cannot reveal the complete technical condition of the UUT. Instead, they observe the outputs generated by the system, signal any deviation from the normal function and thus improve the safety of the system. Although many existing faults in the system's hardware may go undetected for some time, this concept has another virtue: it never misses an erroneous output (which can hardly be achieved with the periodic diagnostics). The system having this property is called fail-safe or fault-secure. The most important advantage of the concurrent diagnostics is the capability to detect and/or to locate the soft (intermittent) faults.

The concurrent diagnostics can again be implemented externally or internally. Here the *internal* implementation, which represents a typical case, is based on the use of informational redundancy, i.e. error coding. However, the data formats are in many cases defined in advance (e.g. for microprocessors). Hence the error coding has a limited applicability and can be used practically

only in memory and I/O subsystems. Beyond this, the control signals which always have some inherent redundancy can be checked by some added checkers (e.g. for opcode and address validity, etc.).

The internal concurrent diagnostics can be implemented also by the algorithmic redundancy, i.e. by repetitive computation of the same task. To make this method sensitive also to hard failures (its sensitivity to soft failures is obvious), it is necessary to use different algorithms (N-version programming) or at least different copies of the same program for every repetition, e.g. by software triple modular redundancy (STMR). A simplified version of the algorithmic checking is the verification of assertions, (e.g. by substituting the roots back into the equation, checking the numerical bounds of the solution, checking whether the file is sorted, etc.).

Still another form of concurrent internal diagnostics is the detection of control flow errors by signatured instruction streams (SIS) [2], being a combination of hardware and software means. Each linear (i.e. not branching) segment of the program is provided with a signature verified at the run time. The original signature is computed as a cyclic code character by the compiler generating the object code. The verification is done by a PLFSR attached to the data bus. Fig. 3 shows the implementation of SIS in a typical microcomputer. This method detects all errors in the control flow of the program, but the errors in data paths remain mostly undetected. Thus the error coverage in this case is relatively low but the efficiency measured by the amount of information obtained for the invested means, is very good.

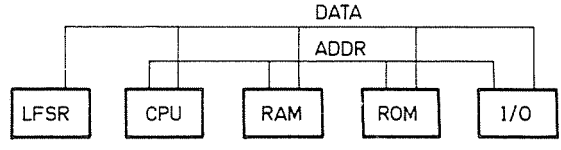

Fig. 3. Error checking by signatured instruction streams

The *external* implementation of the concurrent diagnostics must be based on an existence of some device which continuously verifies the correct function of UUT. Conceptually simplest is the duplication of processors, memories and other subsystems. As this approach leads to a relatively big overhead, a simpler processor called watchdog processor (WDP) can be used to perform the same function [3]. The basic configuration of a system with WDP is shown in Fig. 4. The main processor does the computation, whereas WDP verifies only the assertions, i.e. some properties of the result — similarly as in the case of algorithmic checking within one processor. As the verification of assertions is computationally less demanding, the processor used as WDP can be slower and with less memory. The verification is overlapped with the computation of the next task.

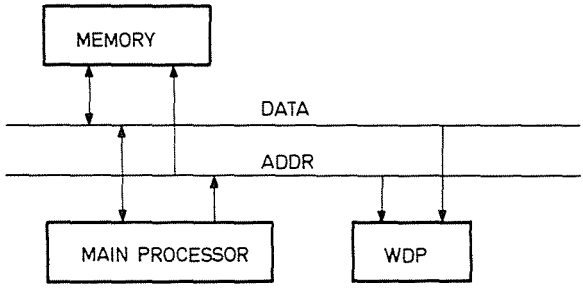

Fig. 4. Watchdog processor for a single-processor system

Choice of Method for Multiprocessor System Diagnostics

The characteristics structure of multiprocessor systems initiated an intense study of specific diagnostic methods for these systems, although any of the above-mentioned methods could be used for the system as a whole. But due to the distribution of the control functions we can treat every processor independently of the rest of the system. Hence it would be appropriate to introduce an additional criterion to those shown in Tab. 1, namely the level of application of diagnostics procedures. The interpretation of the terms introduced in Tab. 1 thus depends on our point of view, e.g. the external diagnostics of one processor can be seen as system's internal if the testing is done by another processor belonging to the system. In the multiprocessor system it is usual to apply the tests at the level of individual processors because the presence of several independent program-controlled units makes this system ideally amenable to the internal diagnostics during which every processor alternativelly plays the role of tester or UUT. Therefore the external diagnostics at the system level are usually not considered.

Periodic Diagnostics

The periodic diagnostics in multiprocessor systems have been solved above all on the basis of the Preparata-Metze-Chien (PMC) model defined in [5]. An overview of methods for multiprocessor system diagnostics can be found in [6].

This approach, although conceptually clear and straightforward, has some very serious drawbacks. One of them is a relatively complex interconnection network necessary for the transmission of diagnostic data. A still more important problem of fundamental importance is the question where the partial test results (i.e. the syndrome) will be evaluated. Concentrating this

function in one of the processors would create a bottleneck of the system, therefore some form of distributed evaluation of syndrome should be considered.

If we want to simplify the diagnostic graph of a multiprocessor system, we must give some more autonomy to the diagnostic functions in the individual processors. In the original PMC model no unit is testing itself, i.e. the self-test is applied only at the system level. A slight modification of this approach is the asymmetric test result invalidation, introduced in the BGM model by Barsi et al [7]. The asymmetric invalidation model assumes that every unit, even though faulty, can recognize and signal a fault in any other unit. This assumption is justified by the fact that an incorrect result of a processor test is so obvious that even a faulty unit can easily recognize it. However, there is always some danger that the tester unit, although it recognized the error, cannot signal it (e.g. because its output is stuck to "correct"). To avoid this possibility, we must design the units as self-checking. An error in the evaluation and signalling circuitry would be then signalled in the same way as an incorrect test result, i.e. as "faulty".

The diagnosability of a system with asymmetric invalidation is much higher than that for the PMC model, namely n-2 units can simultaneously be faulty whereas for PMC the upper limit is $(n-1)/2$ (the system is considered to be t-diagnosable if we can identify t faulty units at the same time). This leads us logically to the question how far the use of complete self-test (ST) in some or all of the system units would simplify the system diagnostics. This problem was studied e.g. in [8], however without respect to the structural properties of the system (assuming the existence of complete graph).

It can be expected that using one or more ST units will simplify the system-level diagnostics in several ways:

— simplifying the diagnostic algorithms
— speeding up the diagnostics
— reducing the diagnostic graph
— increasing the diagnosability.

In the extreme case where all units are ST, there are no diagnostic links, there is no algorithm necessary for system-level diagnostics (the evaluation of individual units is independent) and the exact diagnosis can be found even if all units are faulty. All units can be tested simultaneously, hence the whole system test is finished after a single unit test time.

It can be shown [10] that using one ST unit saves two lines if $t=2$. If we use more than one half of ST units, the value of t is equal to their number (e.g. a system with three ST units and six links has $t=3$). In the systems with large number of ST units it is advantageous to link every ST unit with all non-ST units, i.e. to design the diagnostic graph as a superposition of stars with centers in every ST unit.

Concurrent diagnostics

If the system is to be used for a real-time control, we must above all avoid the generation of an invalid control output, i.e. increase the safety of the system using the concurrent diagnostics. The simplest way of doing this would be assigning the same task always to two different processors and comparing the results before their output. If the computational capacity of the system is not big enough to let us sacrifice one half of the processors (or of the computing time) just for diagnostic purposes, we may use some multiprocessor modification of SIS or watchdog processor.

The use of SIS in a multiprocessor system was suggested in [4]. One processor called roving monitoring processor RMP is used for checking the program signatures stored in the signature queue SQ of every processor — see Fig. 5. RMP cyclically compares the signatures found in the queues with those found in the programs. This method, as well as in the case of single-processor system, detects above all the errors in program flow. If, on the other hand, the errors in data flow represent our primary concern, we must use some multiprocessor version of data checking.

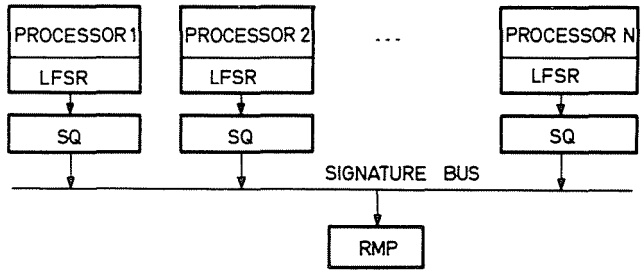

Fig. 5. Roving monitoring processor checking the signatured instruction streams in several processors

One of the possibilities is the generalization of the watchdog processor to the multiprocessor system, as shown in Fig. 6. One WDP is connected to a group of functional processors whose results are to be checked. WDP checks again only the assertions, which is much faster than the computation itself. Assuming that the WDP is of the same type as the functional processors, the number of processors served by one WDP is given by the relationship between the duration of one computation task and the verification of assertion.

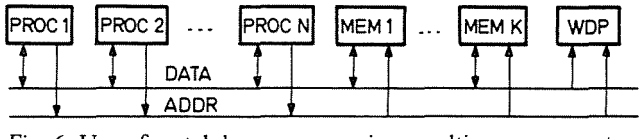

Fig. 6. Use of watchdog processor in a multiprocessor system

Conclusion

This short survey could present only some of the many alternative solutions of the problem of multiprocessor system diagnostics. Its main goal was to show that mutual tests of the processors are not always the best and by far not the only solution. There are still some more approaches to fault handling which were not mentioned, although they are very important too (e.g. the automatic fault masking, error recovery, etc).

The diagnostic methods for digital systems develop with the same speed as the technology and architecture, and on all levels of logical design the general trend towards self-testing and self-checking can be observed. The choice of a suitable diagnostic method for a multiprocessor system is a complex optimization task in which many input values are to be taken into consideration. We try in fact to reach the maximum fault coverage for minimum overall cost, but at the same time we should achieve the required level of reliability and security without decreasing the performance of the system. None of the methods presented above can guarantee the fulfillment of all these requirements, therefore it is necessary to combine them. Verification of the quality of a solution is possible practically only through simulation with fault injection or on a hardware model. First commercially available products with built-in diagnostic functions indicate that the acceptance of diagnosability criteria will soon become an industrial standard.

References

1. SIEWIOREK, D. P.–SWARZ, R. S.: The Theory and Practice of Reliable System Design. Digital Press, 1982.
2. SHEN, J. P.–SCHUETTE, M. P.: On-line Monitoring Using Signatured Instruction Streams. Proc. International Test Conference, Philadelphia, Oct. 1983. pp. 275–282.
3. MAHMOOD, A.–MCCLUSKEY, E. J.–LU, D. J.: Concurrent Fault Detection Using a Watchdog Processor and Assertions. Proc. International test Conference, Philadelphia, Oct. 1983. pp. 622–628.
4. TOMAS, S. O.–SHEN, J. P.: A Roving Monitoring Processor for Detection of Control Flow Errors in Multiple Processor Systems. Proceedings ICCD, New York. pp. 531–539 (1985).
5. PREPARATA, F. P.–METZE, G.–CHIEN, R. T.: On the Connection Assignment Problem of Diagnosable Systems. IEEE Trans. Comp., vol. *EC-16*, pp. 848–854 (Dec. 1967)
6. FRIEDMAN, A. D.–SIMONCINI, L.: System-Level Fault Diagnosis. IEEE Computer, No. 3, pp. 47–53 (1980)
7. BARSI, F.–GRANDONI, F.–MAESTRINI, P.: A Theory of Diagnosability of Digital Systems. IEEE Trans. Comp., vol. *C-25*, pp. 585–593 (June 1976)
8. ZRELOVA, T. I.: Self-Diagnosis of Digital Systems which Contain Parts with Self-Checking Built-in Circuits. Automation and Remote Control, No. 2, pp. 123–132 (1984)

9. LIU, T-S.: Maintenance processors for Mainframe Computers. IEEE Spectrum, No. 2, pp. 36–42 (Feb. 1984)
10. HLAVIČKA, J.: Diagnostics of Multiprocessor Systems with unit Self-Tests. Submitted for publication in Proc. 11th World Congress of IMEKO, Houston, USA (1988)

Doc. Ing. Jan HLAVIČKA, DrSc
CVUT-FEL, Dept. of Computers
Karlovo nam. 13, CS-12135 Prague 2.