PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 37, NO. 2, PP. 97-109 (1993)

FAULT INJECTION BASED DEPENDABILITY ANALYSIS[1]

Balázs BENYÓ* and András PATARICZA**

*Department of Measurement and Instrument Engineering
Technical University of Budapest
H-1521 Budapest, Hungary
Fax: +36-1-166-6808
E-mail: benyo@mmt.bme.hu

**Friedrich-Alexander Universität Erlangen-Nürnberg
Institut für Mathematische Maschinen und Datenverarbeitung III
D-91058 Erlangen, Martensstrasse 3

Received: Nov. 22, 1993

Abstract

In more recent years there has been a rapid increase in the use of fault tolerant systems. The majority of computer systems, even those which are not labeled as fault tolerant have some built-in fault tolerant features. Accordingly, the need for dependability evaluation tools is increasing. These tools may help the system designer in the validation of the fault tolerance specification of their systems.

A portable, general purpose evaluation environment (called DEEP, *Dependability Evaluation Experimental Package*) was developed for the dependability analysis of fault tolerant systems. Our objective was to design a general purpose tool both in the sense of the target machine type and fault conditions as well. A special emphasis was given to a realistic fault injection scheme.

The test environment was implemented for the dependability analysis of the *Modular Expandable Multiprocessor SYstem* MEMSY, developed at the Friedrich-Alexander University of Erlangen-Nuremberg.

In the paper the developed dependability environment (DEEP) is treated. The system structure and the detailed description of the modules are introduced.

The paper contains the description of the reimplementation work of the developed portable system for the master-checker simulation as well. Experimental results of the evaluation of the MEMSY system are presented.

Keywords: fault injection, dependability analysis, master-checker simulation, fault tolerance, experimental validation.

[1]This research is part of Project 70 of the Hungarian-German Scientific Cooperation Agreement with additional support from the following grants: SFB 182 (DFG), OTKA-760 and T-3394 (Hungarian National Scientific Found).

1. Introduction

The estimation of the level of dependability becomes more and more a general requirement not only for fault-tolerant but even for general purpose systems as well. Accordingly, the need for dependability evaluation tools is increasing.

Several dependability evaluation systems exist. The systems can be grouped into the following main classes according to the applied algorithm:

- System examination in *natural (real) environment.*
- System examination in *artificial environment.*
- *Simulation* based estimation.

The dependability evaluation in the real operational environment is a very tedious and time consuming process. In simulation huge efforts are needed to build a sufficiently detailed model of the system properly reflecting the effects of the faults as well. Accordingly, the majority of the dependability evaluation tools examine the target computer system in an artificial environment using fault injection experiments. However, despite the similarities in the evaluation methodology for different computer systems the strong hardware dependency of the usual systems inhibits portability.

Our primary goal was to develop a portable, general purpose evaluation environment for the dependability analysis of fault tolerant systems. The objective was to design a general purpose tool both in the sense of the target machine type and fault conditions as well. Special emphasis was given to a realistic fault injection scheme.

As a pilot project for the verification of the new tool an algorithm was developed evaluating the main dependability parameters of a computer system in *master-checker* mode. This mode is the simplest form of a multiplicated modular redundant system. The processor is duplicated and the corresponding input pins of both CPUs are interconnected, accordingly they process the same input stream. One of them, the master drives the output signals. The master and checker outputs are pairwise interconnected, too, but the checker output pins are used only as inputs to the internal comparator, comparing the signals driven by the master and the internal signal of the checker. The main problem in the evaluation of the system dependability in master-checker mode results from the improper observability and controllability of the system, as in this mode the checker processor is absolutely hidden to the user. This means that this part of the system is neither observable nor controllable.

Accordingly, for the evaluation of the system dependability in master-checker mode traditionally a hardware fault injection method is used. The advantage of this solution is its realistic fault injection methodology but the system realization is time consuming, expensive and unportable.

With the aid of the developed dependability evaluation environment (called DEEP, *Dependability Evaluation Experimental Package*) the algorithm was implemented for the dependability analysis in the master-checker mode working *Modular Expandable Multiprocessor SYstem* MEMSY[2] (developed at the Friedrich-Alexander University of Erlangen-Nuremberg, see [4], [5]). The paper contains the evaluation algorithm and the experiments of the implementation as well.

2. Dependability Evaluation Environment

2.1. Dependability Evaluation Arguments

Dependability evaluation is the process of generating the description of the behaviour of the system related to error occurrence [1]. According to this standard, the main characteristics of a dependable system are defined as follows:

- *Fault coverage*: The subset of faults that are discovered by the system. The fault coverage measure can be estimated simply by collecting the error reports from the system under test, and comparing the number of non-error-free test runs with the total number of test runs.
- *Fault latency*: The average time between the fault occurrence and its manifestation to the outer world.

The DEEP dependability evaluation system investigates both the fault coverage and latency time as well.

2.2. Development Guidelines

The primary goal in the development was the assurance of a high *portability* in order to support its use in the dependability evaluation of several computer systems. Other important factors influencing the development are the following:

- The limitation of the number of necessary experiments for the analysis to an *acceptable level*. The growing complexity of the systems to be examined results in any realistic situation in an enormous number of possible faults. In order to create a measure of system dependability all of these components must be considered.
- The analysis process should be *repeatable* and *comparable*, e.g. in the cases of the evaluation of the several system configurations.

[2]The MEMSY project is supported by the DFG (Deutsche Forschungsgemeinschaft) as part of the 'Sonderforschungsbereich' SFB 182.

– The possibility for application specific dependability checks was to be assured by the open interface for integration of user specified additions e.g. for the examination of the system behavior in the case of special fault injection.

2.3. Fault Injection Method

The dependability analysis method, adapted for the DEEP system is based on fault injection. The main idea of this principle is as follows:

Two systems of identical structure are modelled: the *reference* system and the *probe* system. The environment contains all of the factors influencing the system operation and output (e.g. system stimuli). The only difference between these systems is that during each experiment at a predefined time one fault from the fault list is injected into the probe system by the experiment supervisor.

In this way two system outputs are generated: The error-free response from the reference system which becomes the reference output and the probably faulty output from the probe system. By comparing these outputs and the internal states of the systems, we can evaluate the effect of the fault to the system operation. (The analysis of the internal states supports the detection of those injected faults which are masked or remain latent).

The basic scheme of the fault injection methodology, essentially identical with the classical model of testing is shown in *Fig. 1.*

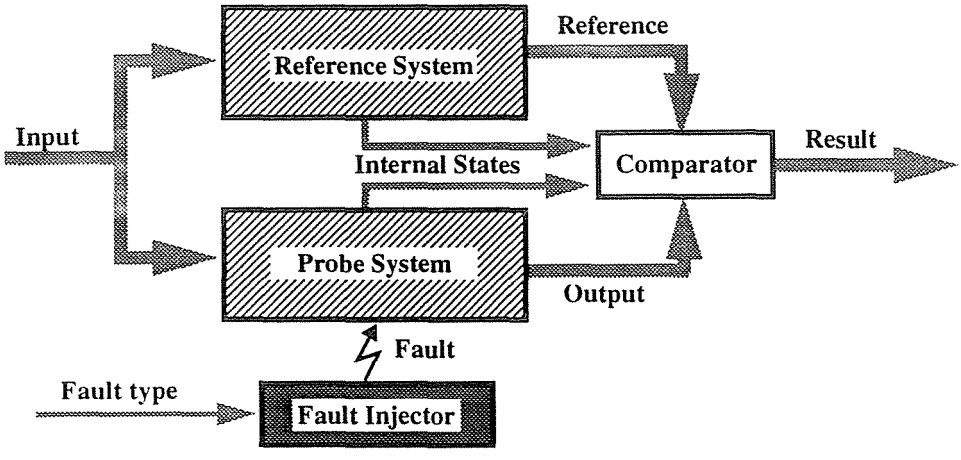

Fig. 1. The scheme of dependability evaluation with fault injection

The main difference between this scheme and the classical test scheme is only the fault injector inserting faults into the probe system.

3. System Development

In order to assure a high degree of portability the hardware dependent and hardware independent functions are implemented in strictly separated software modules, similarly to the architecture of the UNIX operating system. The evaluation environment contains the important hardware independent parts of the above described system, and provides the interfaces to the user for the integration of the hardware dependent parts.

3.1. System Structure

The system is divided into four basic functional modules:

- *Fault injector*

The fault injector supports the generation of different fault types. The faults are injected into the memory directly supporting the simulation of the effects of storage, bus transfer, etc. errors. As the basic functions of this module are based on memory accesses, only the setting of a few configuration parameters, such as memory word width, memory area borders, etc. is required to tailor this module to a new target system.

This principle can be adapted for the error simulation of such other devices as the CPU as well by deducing their faults to a memory image error. The main idea, e.g. for the simulation of a register fault in a CPU can be realized by the following method:

  - Copy all internal registers of the CPU into the main memory, into the work area of the experimental environment.
  - Inject a fault into the copy of the register.
  - Reload copies into internal registers.

- *Observer*

The observer module generates the description of the observed systems state in order to compare it with the reference systems description.

Some experimental setups are required to store the system state description. In order to decrease the huge storage place requirement of the state description the observer ensures the possibility to perform some information compaction methods, e.g. checksum or signature analysis.

- *Evaluator*

The evaluation module analyses the occurrence of the several faults and the average latency time of the error. As a service to the supervisor module it accounts the required experiment number based on the defined

consistency interval. The applied algorithm of the experiment number estimation can be found in [2].

- *Supervisor module*

  The supervisor module has three main functions:

  - Organizing and synchronizing the working modules during the experiments.
  - Handling the observed system.
  - Providing some basic services to the other modules.

The hardware dependent part of the fault injection system contains the *description of hardware environment* and some simple service routines (e.g. a pseudo random number generator, based on [2], controlling the fault injection into the defined memory area or a time-out watch-dog, etc.). Due to the hardware dependency of parts of the system, these routines are located in separate source files in the supervisor module and they are linked to the system after a recompilation. At the same time the supervisor module assures a unified interface of the hardware dependent functions logically belonging to other modules. A separate hardware description file contains the definitions of data types and constants used by all of the modules.

The structure and information flow of DEEP are shown in *Fig. 2.* Information flow in this sense means not only data flow but also several services provided by modules to each other as well. A more detailed technical description of the system can be found in [7].

4. Dependability Evaluation of the Computer in Master-checker Mode

For the conceptual validation of the DEEP project an application was selected, which assured not only the checking of correctness of the implementation but also characterized the efforts needed to model building, too. As a pilot project, the simulation of the master-checker mode was selected. The main attraction of this selection is that the execution of a whole model design process was required, while the results of the measurement experiments were well-predictable and verifiable due to the simplicity of the principle to be modeled.

4.1. The Master-checker Principle

Duplicated modular redundancy is a well known principle in fault tolerant system design [4], [5], [6], which means that the function to be made fault tolerant is executed simultaneously in two identical units processing the

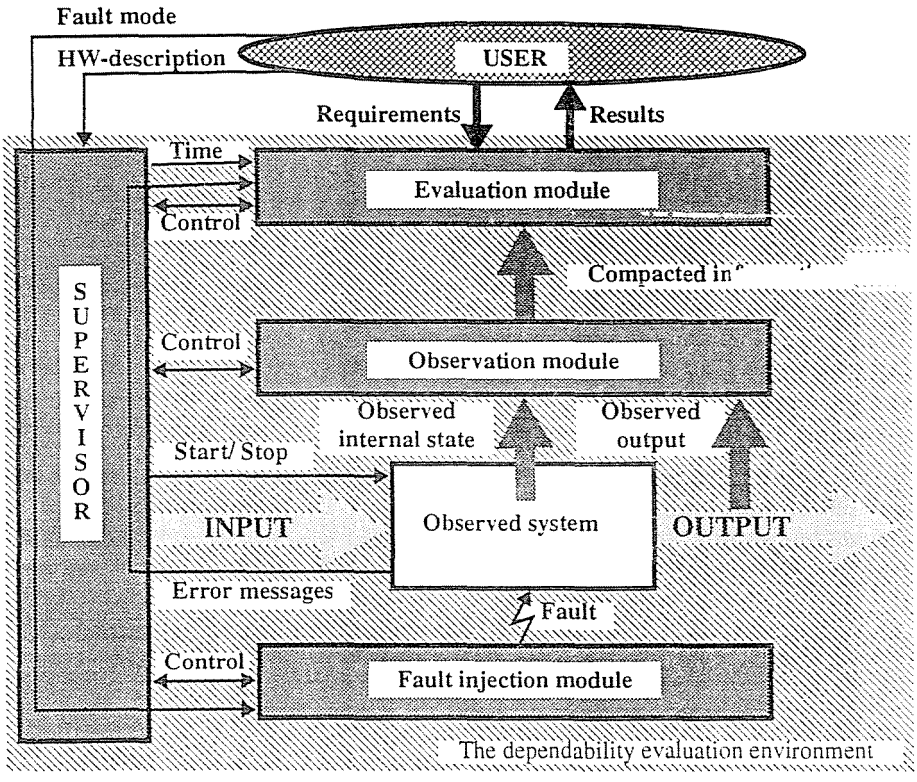

Fig. 2. The environment structure and the internal information flow

same input. As far as the system is fault-free, each unit generates the very same output. In the case of a fault in any unit a discrepancy appears on the system outputs which can be detected by a comparator.

In master checker mode the system setup, as depicted in *Fig. 3* is based upon this duplication of the system. Both the master and checker units receive the same inputs. One of the units, the so-called master generates the output signals of the system. The checker unit performs the same operations as the master and the outputs of both units are compared by the built-in comparators of the checker. In the case of a mismatch an error is detected and signalled to the 'outer world'. It should be pointed out that the application of a combinatorial comparator assumes a strict synchronism of both units.

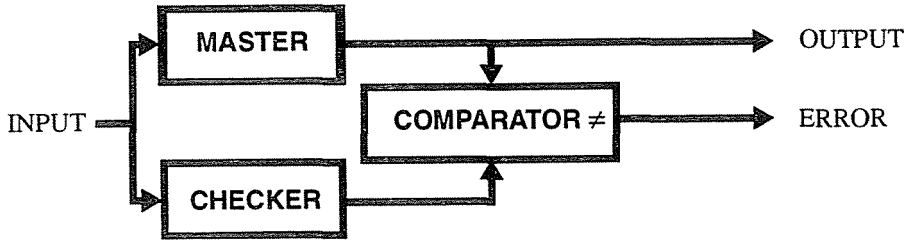

Fig. 3. Master-checker mode

As in all duplicated systems, this approach supports only fault detection. In the case of a discrepancy it cannot be decided which unit produced a faulty output, thus neither unit level diagnostics, nor error correction can be performed.

Another restriction on the fault detecting capabilities results from the processing of the same input signal stream by both processors. In the case of an input error both units generate identical (and from the user's point of view faulty) outputs without any error signal. For detailed information on the master-checker principle see [3] and [6].

4.2. The Problem with the Realization of the Fault Injection Based Evaluation

The master-checker configuration detects any fault in the duplicated part of the system. In order to evaluate the dependability measures of a system in master-checker mode faults are to be injected into one of the duplicated units of the system. The problem is caused by the improper controllability of these units, as in master-checker mode both of the duplicated units (the master and checker) receive the very same input. In this way a software injected fault had an identical effect on both units. Because of this reason in the master-checker mode a software fault generator without hardware modification cannot be applied to the evaluation of the effects of faults in the computing core.

Potential solutions to this problem are the use of a hardware fault injector or the evaluation of a fully software-based simulation model, which assures higher controllability than the real system itself. Because of the unportability of the hardware fault injection based system evaluation we selected the second solution.

4.3. The Method of Master-checker Simulation

In the simulation of the master-checker mode a special type of simulation was applied: The *real system* has been utilized in normal configuration running a strictly deterministic, and therefore repeatable process as benchmark.

At the beginning of the dependability analysis a reference run was made. Storing the information consisting of the system states descriptions during this reference run a reference data base describing the error-free system operation was generated. In the forthcoming experiments the comparison functions of the master-checker mode were simulated by subsequently performing step by step comparisons between the actual and stored system states. In the case when the comparison results in a discrepancy of the states a master-checker fault event was detected.

The main idea in this modelling approach is the approximation of the immediate error detection of the master-checker mode by the sequence of checks of sampled system states. It should be pointed out that a sufficiently high frequency of the checks of the system state assures a statistically reasonable approximation even for the error latency measurement.

4.4. The Realization of Master-checker Simulation

MEMSY is based on the MVME188 system containing one or two MC88100 central processor units and some MC88200 cache/memory management units depending on the system configuration [3], [7], [9]. The basic structure of the system is shown in *Fig. 4.* In case of a MVME188 system the duplicated part of the system is the computation kernel: the CPU and the corresponding MMUs.

In the real master-checker mode the corresponding M-bus signals of this units are compared at the beginning of each clock-cycle. Based on practical considerations some simplifications were applied in the simulation:

- The memory contents were compared rather than M-bus signals.
- The check frequency was lower than in master-checker mode, as described above. (In the actual implementation one checkpoint was inserted after each C instruction.)
- Only the data lines of the system were compared.

Considering the first and third simplifications the simulation model based estimator of the fault coverage is lower than in the real master-checker mode. Because of the second simplification the error latency measured by the simulation model will be longer than in the real case.

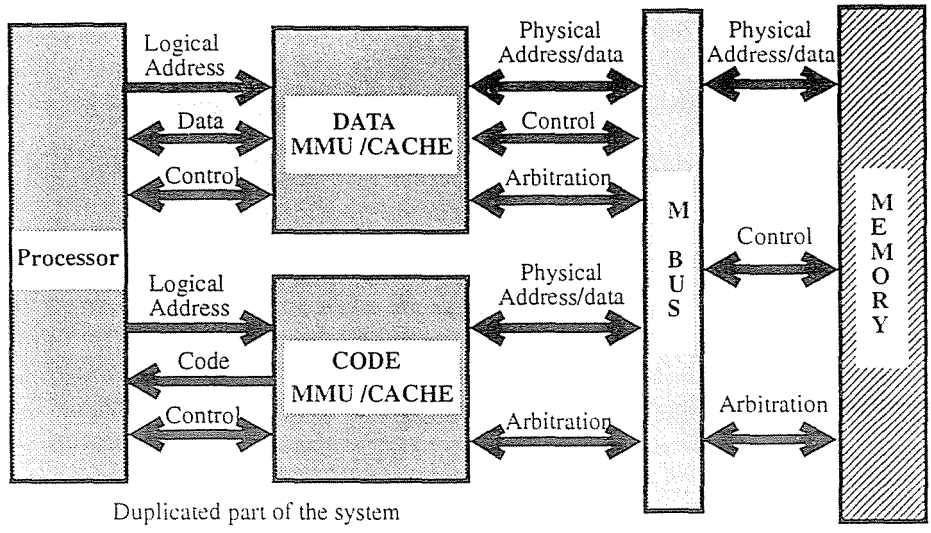

Fig. 4. MC88200 system configuration

In summary: the result of the dependability evaluation utilizing the implemented simulation model of the master-checker mode is a pessimistic estimation of the real system dependability.

In the first reference run of the benchmark in every check-point the current memory state (or the compacted memory content) was saved. After this in the sample runs at each checkpoint the current memory state was compared with the saved reference. In case of a discrepancy a master-checker error was registered.

The method is illustrated in *Fig. 5.*

5. Experimental Results

As described earlier, in this phase of the development the primary target of the experiments was the validation of the simulation methodology of the master-checker configuration rather than the evaluation of dependability in all details.

In order to prepare a comparison with other fault-tolerance techniques such as watch-dog processors performing a control-flow check with a high level instruction resolution, the checkpoint number was utilized in the evaluation of the master-checker mode rather than real time of the checkpoint

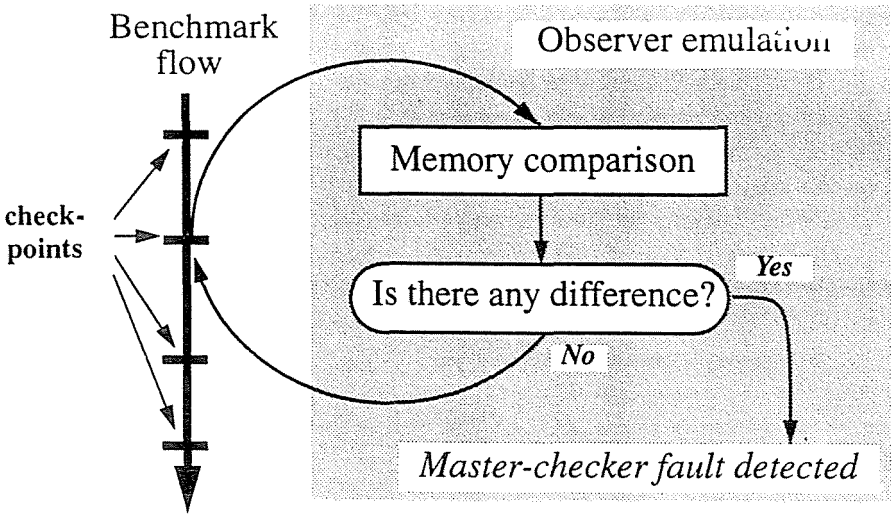

Fig. 5. Master-checker simulation method

as a time stamp. In this way a rough estimation of the error latency is produced.

In the experiments a copyback memory update policy was applied. (For the description of the MC88200 CMMU see [10]). In a different case of injected type of fault, 2000 repetitions of the experiment were done. The effected area was the whole cache: the address tags, the status information and the data area as well.

The results are summarized in *Table 1.*

Table 1
Experimental results

Injected fault	Masked	Detected	Latency*
5 single stuck at 0 fault	62%	38%	38%
1 double stuck at 0 fault	54%	46%	42%
5 double stuck at 0 fault	18%	82%	40%

* In portion of the total run time.

Even this rather simplified experiment shows the value of the approach in the qualitative evaluation of fault-tolerant systems.

The value of fault coverage approximately equals its expected value. The majority of the faults are masked by the later operations. These experimental results nearly coincide with the corresponding ones in the literature [8]. The number of masked errors decreases below 50% only in the case of serious, non-realistic fault model.

The long fault latency compared with the total run time shows that the memory area used by the benchmark fits into the cache and the errors were detected only near to the finishing phase of the benchmark run, during the copyback operation of the computed results into the main memory. Accordingly, the injected fault set does not affect significantly the fault latency.

6. Summary

6.1. Results

A portable dependability evaluation system called DEEP, *Dependability Evaluation Experimental Package* is developed containing the hardware independent parts of a fault injection based dependability analysis system and provides well-defined interfaces for the implementation of hardware dependent parts. The system contains approximately of 1000 lines of C source code.

A new, simulation based dependability evaluation method was created for the master-checker mode working computer systems. The simulation algorithm was validated by implementing it in the MEMSY system.

6.2. Further Work

The continuation of the work, on the one hand, should be the full dependability evaluation of the MEMSY system in master-checker mode. We should make experiments in the case of various fault models. By experiments in different cache policies different system configurations can be compared with respect to dependability.

On the other hand, an important addition to DEEP will be the integration of statistical routines in order to adaptively control the number of the experiments in order to assure a proper confidence level of the measurement results, and a perspective goal will be the integration of this simulated data acquisition system to an intelligent model generation and verification system.

A current diploma project aims the implementation of DEEP for educational purposes on PC.

References

1. LAPRIE, J.C. (ed.) (1991): Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese, IFIP WG 10.4 Dependable Computing and Fault Tolerance, Springer-Verlag, Wien, New York.
2. JAIN, R. (1991): The Art of Computer System Performance Analysis (Techniques for Experimental Design, Measurement, Simulation and Modelling), John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore, Wiley cop. 1991.
3. PATARICZA, A. (1991): Hardware Testing Concept for the MEMSY Computing Nodes, Friedrich-Alexander University of Erlangen-Nürnberg, Inst. of Computer Science, Internal Report No. 6/91.
4. DAL CIN, M. – GRYGIER, A. – HESSENAUER, H. – HILDEBRAND, U. – HÖNIG, J. – HOHL, W. – MICHEL, E. – PATARICZA, A. (1993): Fault Tolerance in Distributed Shared Memory Multiprocessors. To appear in Springer LNCS, 1993.
5. DAL CIN, M. – HOHL, W. – MICHEL, E.– PATARICZA, A.(1993): Error Detection Mechanisms for Massively Parallel Multiprocessors. *Proc. Euromicro Workshop on Parallel and Distributed Processing*, Gran Canaria, 27 - 29. Jan. 1993, pp. 401-408.
6. HOHL, W. – MICHEL, E. – PATARICZA A. (1993): Hardware Support for Error Detection in Multiprocessor Systems - A Case Study, *Microprocessors and Microsystems*, Vol. 17, No 4, May 1993, pp. 201-206.
7. BENYÓ, B. (1991): Fault Injection Based Dependability Evaluation, Diploma Thesis, Erlangen-Budapest, Jun. 1991.
8. CHOI, G. S. – IER, R.K. – CARRENO, V.A. (1990): Simulated Fault Injection: A methodology to Evaluate Fault Tolerant Microprocessor Architectures, *IEEE Transaction on Reliability*, Vol. 39, No. 4, October 1990.
9. MOTOROLA (1989a): MVME188 VMEmodule RISC Microcomputer User's Manual, Preliminary; September 1989.
10. MOTOROLA (1989b): MC88200 Cache/Memory Management Unit Users Manual, Prentice Hall, Englewood Cliffs, New Jersey 07632.