Evaluation of a Dependable Architecture

Amir Rajabzadeh
rajabzad@ce.sharif.edu

Mirzad Mohandespour
mohandes@ce.sharif.edu

Ghassem Miremadi
miremadi@sharif.edu

Department of Computer Engineering
Sharif University of Technology

Abstract

Functionality of Master/Checker (M/C) mode is based on the redundancy of the processors supported by most modern processors. This paper presents an experimental evaluation of the M/C mode in a 32-bit Pentium® processor system using power-supply disturbance (PSD). The results of PSD show that the M/C mode has only 67.13% error detection coverage. The low coverage is caused by the crashes in the Master processor as the result of voltage fluctuations. Use of a watchdog timer may raise the coverage up to 99.73%. Moreover, the correctness of the results produced by the Master processor has been checked. In many cases, the results were correct while the Checker processor announces an error.

Keywords: Master/Checker Mode, Fault Injection, Experimental Evaluation, Watchdog Processor.

1. Introduction

In design of dependable microprocessor based systems, using hardware redundancy has always been a safe solution. Hardware redundancy is widely used in embedded, industrial, and real-time systems, here reliability [1], safety, security [2] and availability [1], [3] are of important concerns. Hardware redundancy can be used in deferent levels such as system level, chip level, or inside the chip. Redundancy in system level, despite its high reliability, is expensive due to system repetition. Redundancy inside the processor needs ASIC design. Fortunately, redundancy over chips and particularly processors is reasonably reliable and cost-effective. In general, two or more concurrent processors synchronously execute a single application, and an external comparator compares their outputs. In case of mismatch appearance, the comparator announces an error [4], [5].

In the last decade, the growth in computer architecture complexity and subsequently bus complexity, higher working frequency and higher number of processor pins, made manufacturing comparators very difficult, if not impossible. Therefore, the designers migrated from designing external comparators to the comparators inside the processor. The M/C mode allows the comparators to be used inside the processors. The M/C mode is supported in many modern processors such as Pentium Family [6], AMD *K5™* [7], MIPS R4000 [8].

In M/C mode, two processors synchronously execute a single program. One processor is in Master mode, and the other one in Checker mode. In the case of mismatch appearance in output pins, Checker announces an error. M/C mode is expandable to more than one Checker.

This paper discusses experimental evaluation of error detection coverage in the M/C mode. The processor used in these experiments, is a 32-bit Pentium® processor. To evaluate the M/C, the PSD fault injector is used and 3000 experiments are performed.

M/C mode in Pentium® processor has been discussed in the section 2 and will be followed by a thorough description of experimental system and its components in section 3. Section 4 presents the design of the fault injector. Afterward the results of the experiments are explained, and finally a conclusion is presented.

2. M/C mode

The M/C mode functionality is based on the duplication of processors. Both processors run the same program and process the same data stream, fully clock synchronous. In the Intel Pentium family, such a duplication structure can be set without external components, as the necessary logic is integrated inside the chip and is called Functional Redundancy Checking (FRC). Two processors are required to support FRC. The processor configured as the Master, operates according to the bus protocols. The outputs of the Checker are tri-stated (except few pins) so the outputs of the Master can be sampled. If the sampled value differs from the value computed internally by the Checker, the Checker asserts an output pin (IERR#) to indicate an error.

3. Experimental system

The architecture of the experimental system consists of five main parts as are shown in figure 1:

1) a Pentium® M/C board, 2) a watchdog processor (WDP), 3) fault injector, 4) an interface logic and 5) a Host computer.

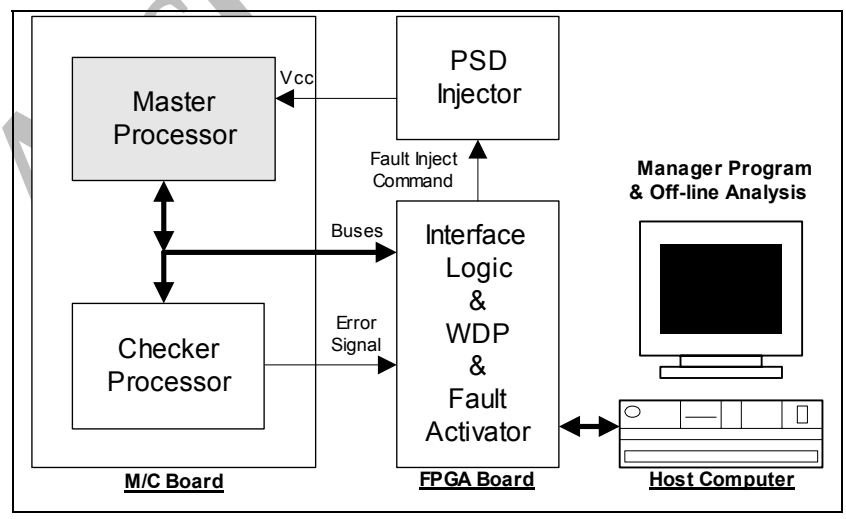

Figure 1 - Organization of the Experimental System

3.1 M/C board

The M/C board was equipped with two 100 MHz Intel Pentium® processors. The Pentium® processor was selected because of the following three reasons:
1) The processor supports M/C mode. Note that M/C mode is functionally equivalent to the features of the next generation of Pentium® processor family, 2) the implementation is simple and 3) the results can be obtained and evaluated reasonably fast. Notice that a custom board is developed to make processor pins accessible.

3.2 FPGA Board

The WDP, the Fault activator logic and the Interface logic were integrated in a board, called FPGA board equipped with an Altera Flex10k30 FPGA.

The WDP decodes the processor cycles for validity of Checker processor error signal. In addition, a workload timer (WL-timer) was implemented in the WDP. The WL-timer sets an upper limit for workload execution time. At the beginning of the workload, a few commands will be added to set the WL-timer initial values. With every clock pulse, WL-timer value decreases one unit. At the end of the workload, a signal stops the WL-timer. If the WL-timer reaches zero before the end of the workload, an error will be sent to Manager Program running in the Host computer.

The Fault activator gets a command from the Manager Program and activates the PSD fault injector circuit.

The Interface logic establishes communication between the Host computer and the WDP as well as the Fault activator. It is used for boards with different frequencies. It receives signals from Manager Program and sends them to the boards from one side, and from the other side, it reports the system condition to the Manager Program. It contains buffers to store data and results temporarily.

3.3 Host Computer

The task of the Host computer is to manage and control the whole experiment. For each fault, the Host computer performs several operations in the following order:

1) Resetting the Pentium® processors and all registers of the WDP.
2) Waiting for the start of workload program execution.
3) Determining a random time for fault injection.
4) Issuing the command for fault injection.
5) Reading coverage information from the WDP.
6) Analyzing raw data.

The information is stored in a Microsoft Access bank, which will be analyzed at the end of the experiments. This is done by the Manager Program in the Host computer.

4. Fault Injection Method

In this experiment, power-supply disturbance (PSD) fault injection method is used. The Manager Program controls a PSD circuit through the fault activator to inject faults into the Master processor. The PSD fault injection occurs with voltage drop in power supply for a few

milliseconds (Figure 2). In these experiments, when Pentium® power voltage drops to 2.5 V no severe problems occur, however every time the Pentium® power voltage drops under 2.5 V the processor does not work properly. Note that if the voltage drops to 2.2 V the crash probability increases dramatically, and at 2 V it crashes almost all the time.

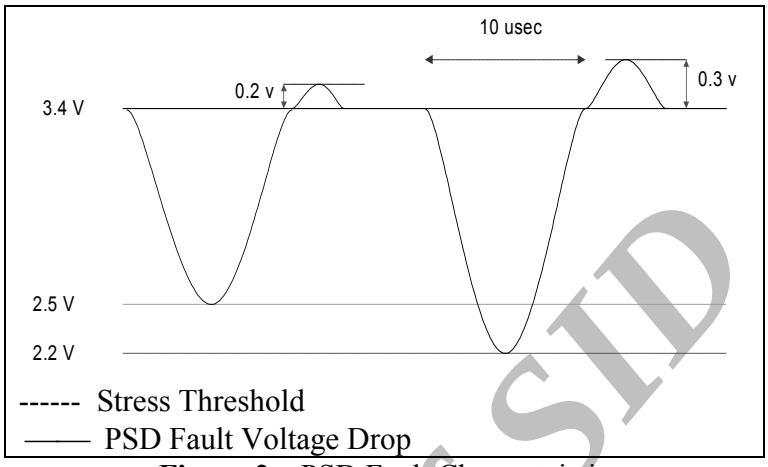

Figure 2 – PSD Fault Characteristic

Modern processors work with very high frequency. Therefore, the desired current should be applied in a short interval. Hence, the bulk storage capacitors with a low ESR (Effective Series Resistance) are required. In order to reduce the ESR, it may be necessary to place several bulk storage capacitors in parallel. These capacitors should be placed near the Pentium processor on the power plane(s) to ensure that the supply voltage stays within specified limits during changes in the supply current during operation.

It is very difficult to design a PSD fault injector circuit for modern processors with bulk storage capacitors around its power supply pins.

Assume the substitute capacitor of Pentium processor is C, then Q of the capacitor will be C×V or in other words C×3.5, and fault injector circuit should flush part of the charge quickly. As until the Pentium power supply voltage is above 2.5 V, processor operates normally, therefore the fault injector circuit should at least drop the voltage to 2.5 V and the circuit should flush C× (3.5-2.5) = C×1 during the injection time. According to this approximation, if the flushed current is assumed constant, it can be concluded that: (T is the duration of fault injection)

$Q = C \times V = I \times T \Rightarrow I = C \times V / T$

Assume the clock frequency inside the Pentium® processor is 100 MHz, the fault injection duration is half a clock (0.05 nsec) and capacitor is 1μF, then for 1V drop in processor power supply voltage the following current should be tolerated.

Duration $T = 0.05$ nsec $\Rightarrow I = C \times V / T = 1\mu F \times 1V / 0.05$ nsec = 20,000 A = 20 KA!

This calculation implies that, only high-energy pulses in power supply can produce a fault in Pentium® processor. In other words, any attack from fast pulses of the power supply will not make fault in the processor, unless pulses have huge duration. For instance, if pulse duration was several thousand cycles, it is possible to drop the voltage, and produce a fault. However if fault occurs in the processor in many cycles, then the crash can be simply predicted. Therefore, transition pulses in power supply take the processor in one of the two

following conditions: either the system continues normally or, it crashes. In other words, based on the experiments, there may be no middle condition.

5. Experiment Results

The main attention is on M/C mode, which has been tested with PSD fault injection method. A quick sort (Qsort) program is used as the workload. Another program verifies the correctness of the Qsort and sends the result to the Manager Program at the end of the workload. This is used as the reasonableness checking mechanism to check M/C results. The results are based on 3000 faults.

5.1. Evaluation of M/C Mode

This section presents the experimental results on error detection coverage of the Checker processor in the result of PSD fault injection in the Master processor. Results of the error detection coverage are shown in Table 1.

Table 1 – Error Detection Coverage of M/C mode

	PSD
Number of tests	3000
Number of errors detected by Checker processor	1984(66.13%)

According to the results, the M/C mode is not effective (66.13%) with PSD fault injection method. In this method processor power supply disconnects for a while. At this time, processor, as a huge sequential circuit with certain predefined states, changes states. Since the predefined states are less than undefined states in the huge sequential circuit, with high probability the new state is not a predefined state, therefore the processor crashes with high probability. To make the results confident and detect the crash state, a WL-Timer has been implemented. This timer checks an upper bound for the workload execution time in a way that it only detects faults at the crash time. As shown in Table 2, considering the small intersection between Checker and WL-Timer as the result of PSD fault injection, and the fact that WL-Timer does not announce any fault except at the crash time, the ineffectiveness of the M/C mode at hard crash is concluded. In addition, Table 2 shows that combination of the M/C mode and the WL-Timer can improve error detection coverage up to 99.73%.

Table 2 – Checker processor Error Detection Coverage

	PSD
Number of tests	3000
Number of errors detected by WL-Timer (crash states)	1486(49.53%)
Number of errors detected by Checker and WL-Timer	441(14.7%)
Number of errors detected by Checker or WL-Timer	2992 (99.73%)

5.2 Reasonableness Checking

At the end of Qsort, there is another program to examine the correctness of the results. This program sends a Sort-OK message to the Manager Program if the Qsort program sorted

array correctly. At this time, the results of the M/C mode and reasonableness checking mechanism will be investigated. Fault detection coverage in M/C mode and reasonableness checking mechanism (Sort-OK) is shown in table 3.

Table 3 – M/C Mode and Reasonableness Checking Mechanism

	PSD
Number of errors detected by Checker processor	1984
Checker detected and Sort-OK	1505(75.86%)

The M/C mode in 75.86% of the cases announces fault despite Sort-OK. The reasons are:

1- Some faults are overwritten and does not affect the results.

2- The Checker processor detects some faults mistakenly as the result of voltage drop at external bus of the Master processor.

It is very important that PSD in the Master processor can distort the bus voltage and causes the Checker processor to detect a fault mistakenly. Table 4 shows the voltage margin of Pentium® processor from the manufacturer manual [6]. From the Table 4 it is obvious that as the effect of fault injection in PSD if V_{High} drops under 2 V, there is a possibility that the Checker processor consider it as zero, since it is less than V_{IH-Min}.

Table 4 – Voltage Margin of Pentium® Processor

	V_{IH-Min}	V_{OH-Min}	V_{IL-Max}	V_{OL-Max}
Pentium® processor	2	2.4	0.8	0.4

In such case, the Checker announces one fault caused by voltage fluctuation. The Checker fault announcement shows its sensitivity and accuracy on one hand. On the other hand, based on the result checking (Sort-OK), no fault has appeared and Pentium® processor continues its operation without any fault. From this point of view, Checker has detected a wrong executive cycle due to voltage drop.

6. Conclusion

The results of the experiments are quite different from that of previous works on the subject [9]. There are few notes to pay attention to:

1. M/C mode is not strong to hard crash. Hard crash occurs in PSD due to change of state in M/C to some anonymous undefined states and as the result, faults are not recognizable by the Checker.

2. In PSD, part of recognized faults is because of voltage drop in bus. That is not an actual fault since in most cases reasonableness checking mechanism approves the correctness of the program execution.

3. In modern processors, huge capacitors and high frequency make a remarkable difference in PSD in comparison to previous generation of the processors. The results illustrate that sudden pulses in power supply should contain high energy to generate a fault. In other words, PSD faults occur as the result of pulses with long duration. Note that, if duration of fault pulses is long, processor is very probable to crash. Therefore, with high probability, pulses in Pentium®

power supply, take the processor in one of the following two states: Either the system continues to work normally, or it crashes. The occurrence of middle states is rare.
4. The results of this experiment are quite different from the results of the previous experiments with PSD. In modern processors, PSD does not necessarily generate control flow errors.
5. It mentioned that the PSD often takes processor to the hard crash. This makes the mechanisms based on time-out (e.g. WL-Timer) more important. As the results confirm, WL-Timer mechanism is capable of increasing fault detection coverage up to 99.73%.

References

[1] N. Oh, P.P. Shirvani and E.J. McCluskey, "Error Detection by Duplicated Instructions In Super-scalar Processors", *IEEE Trans. on Reliability*, Mar. 2002, pp. 63-75.

[2] P. Croll and P. Nixon, "Developing safety-critical software within a CASE environment", *IEE Colloquium on Computer Aided Software Engineering Tools for Real-Time Control*, Apr 1991, pp. 8.

[3] A. Benso, S. Di Carlo, G. Di Natale, P. Prinetto and L. Tagliaferri, "Control-flow checking via regular expressions", *Proceedings of 10th Asian Test Symposium*, Nov. 2001, pp. 229-303.

[4] G. Miremadi, J. Ohlsson, M. Rimen, and J. Karlsson, "Use of Time, Location and Instruction Signatures for Control Flow Checking", *Proceedings of the DCCA-6 International Conference*, Urbana Champaign, IEEE Computer Society Press, ISBN 0-8186-7803-8, 1998.

[5] G. Miremadi and J. Torin, "Effects of physical injection of transient faults on control flow and evaluation of some software-implemented error detection techniques", *Dependable computing and fault-tolerant systems*, springer, ISBN 3-211-82649, 1995. pp. 435-455.

[6] Intel Corp., *Pentium ® Processor Family Developer's Manual*, 1997.

[7] Advance Micro Devices, Inc., *AMD K5 ™ Processor* Data sheet, 1997.

[8] MIPS Technologies Inc., *MIPS R4000 Microprocessor User's Manual*, Second Edition, 1994.

[9] Advanced Micro Devices, Inc., *AMD x86-64 Architecture Programmer's Manual, Volume 2: System Programming*, September 2002.