Microprocessors and Microsystems 35 (2011) 441–452

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Real-time fault injection using enhanced on-chip debug infrastructures

André V. Fidalgo [a,*], Manuel G. Gericota [a], Gustavo R. Alves [a], José M. Ferreira [b]

[a] Instituto Superior de Engenharia do Porto, Polytechnic Institute of Porto, Portugal
[b] Faculdade de Engenharia da Universidade do Porto, University of Porto, Portugal

ARTICLE INFO

Article history:
Available online 28 October 2010

Keywords:
Dependability
Fault injection
On-chip debug
Real-time systems
Microprocessors

ABSTRACT

The rapid increase in the use of microprocessor-based systems in critical areas, where failures imply risks to human lives, to the environment or to expensive equipment, significantly increased the need for dependable systems, able to detect, tolerate and eventually correct faults. The verification and validation of such systems is frequently performed via fault injection, using various forms and techniques. However, as electronic devices get smaller and more complex, controllability and observability issues, and sometimes real time constraints, make it harder to apply most conventional fault injection techniques. This paper proposes a fault injection environment and a scalable methodology to assist the execution of real-time fault injection campaigns, providing enhanced performance and capabilities. Our proposed solutions are based on the use of common and customized on-chip debug (OCD) mechanisms, present in many modern electronic devices, with the main objective of enabling the insertion of faults in microprocessor memory elements with minimum delay and intrusiveness. Different configurations were implemented starting from basic Components Off-The-Shelf (COTS) microprocessors, equipped with real-time OCD infrastructures, to improved solutions based on modified interfaces, and dedicated OCD circuitry that enhance fault injection capabilities and performance. All methodologies and configurations were evaluated and compared concerning performance gain and silicon overhead.

1. Introduction

Most of today's safety–critical applications require some type of computer-based device, broadening the application range of microprocessor systems. As electronic systems increase in complexity and decrease in size, their correct behavior is becoming harder to guarantee [1]. The higher sensitiveness to noise and other factors increases the probability of errors, even for devices used in non-hostile environments. The most frequent hazard affecting microprocessor systems is usually referred as a Single Event Upset (SEU) and consists of a change of state of a flip-flop, induced by an ionizing particle such as a cosmic ray or proton. This event may change the logical value of memory elements, such as registers or memory cells [2].

The verification and validation of dependable systems requires the study of failures and errors in order to evaluate their probability of occurrence and subsequent effects. The possibly destructive nature of a failure and the long error latencies make it difficult to identify their causes in the operational environment, and recommend the organization of experiments under precisely controlled conditions. Depending on the system function and architecture, hardware [3] and software [4] fault tolerance techniques can be used to minimize the effects of SEUs, enabling the system to provide acceptable service in their presence. All vulnerable critical systems should be verified to ensure operation within acceptable limits in the presence of such events, and validated to check if they accomplish their intended objectives. Fault injection can be used both to evaluate fault tolerance implementations and to estimate fault consequences on non-tolerant systems.

When dealing with microprocessors, the main limitations imposed on fault injection are control, internal access, intrusiveness and performance. Ideally a fault injection methodology should allow precise control of fault insertion, both in time and space, complete replicability of experiments, and access to all microprocessor resources. Simultaneously it should require no modifications to the target software or hardware, and should execute in real time. As this is not technically feasible, all fault injection environments are based on acceptable (or possible) trade-offs. Access to the area where faults are to be inserted is a major problem, often requiring either ad hoc [5], intrusive [6], or low-controllability [7] approaches. The first and second solutions require special hardware or modifications to running software, offer restricted coverage, and may be difficult to execute in real-time. The third solution is usually based on contactless fault injection techniques, making fault synchronization and replication hard or impossible to guarantee. OCD infrastructures have been used as an efficient alternative

* Corresponding author. Tel.: +351 228340500; fax: +351 228321159.
E-mail addresses: anf@isep.ipp.pt (A.V. Fidalgo), mgg@isep.ipp.pt (M.G. Gericota), gca@isep.ipp.pt (G.R. Alves), jmf@fe.up.pt (J.M. Ferreira).

0141-9331/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.10.002

to handle such problems [8] and the addition of circuitry to evaluate the vulnerability to SEU effects is increasingly accepted at the design stage [9].

This paper proposes a set of fault injection solutions enabled by debug features that are now present in recent microprocessor devices. The proposed fault injection environment was designed to be non-intrusive and to allow real time emulation of SEU effects in the microprocessor memory. Real time operation requirements may indeed justify the use of modified OCD infrastructures in order to provide better fault injection capabilities and/or performance. The rationale behind the proposed solutions is that microprocessor systems dependability would benefit from enhancements aimed at improving fault injection operations, making them viable from both economical and technical viewpoints. The modified OCDs proposed in this paper are based on the use of wider data link with an external debugger, or on the use of a dedicated fault injection module, with low overhead and higher autonomy. More intrusive fault modules were also considered as a way to increase fault coverage on safety–critical devices, enabling the insertion of precisely controlled faults on internal registers or protected memory.

The next section summarizes the state of the art and preliminary research. Section 3 presents our proposed solutions, including the experimental environment and application methodology. Section 4 presents the experimental results obtained during the course of this work. Finally, Section 5 presents the main conclusions, and suggests directions for future research.

2. State of the art

2.1. Real-time fault injection in microprocessors

Real time usually designates systems that must provide adequate response within a specified time window. In this case, dependability is harder to implement and more troublesome to evaluate. The correctness of the results must be checked and accurate meeting of deadlines is mandatory, without modifying or stopping the target system.

Real-time fault injection must be executed with the target system running at full speed, with minimum intrusiveness and delays. Most traditional fault injection approaches cannot be adequately used under these constraints. Simulation based fault injection can be useful on early stages of development, but it is often time-consuming and intrinsically dependent on the quality of the available model [10,11]. Additionally, it is very difficult to implement a model that accurately represents all the delays and other timing aspects, and a different technique must be used once a prototype (or production model) is available. Software fault injection adds fault insertion routines, causing extra delays and limiting the fault targets to those areas accessible by the application code. Although work on this area has shown that it can be used for some real-time systems [12], it presents considerable limitations in terms of intrusiveness and coverage. The need to slow down or stop the running application also makes it inconvenient to apply most contact fault injection techniques, since they degrade system performance. Most technical solutions to this problem rely on contactless fault injection [7] or on special dedicated infrastructures [13], both of which are complex and expensive. Contactless techniques present controllability and replicability problems, concerning precise control of the instant and location of a fault. Dedicated fault injection infrastructures come together with silicon overhead and often require special prototype versions of the target system, hardly or even not adaptable to the final product. Additionally, access to internal blocks where faults are more probable, generally the memory elements and communication buses, is also problematic, particularly without disturbing the running applications.

Recent approaches to real-time fault injection include improved software techniques [14], halting the target with minimal delay for near real-time fault injection [15] or taking advantage of recent FPGA capabilities [16,17]. As many of today's microprocessors incorporate dedicated OCD circuitry, designed to operate independently of the target system resources, their use for fault injection purposes is becoming increasingly popular.

2.2. Fault injection via OCD

The OCD implementations present in different families of microprocessors share common characteristics that form a core feature set, usually including run control, breakpoint support, and memory and register access. Some devices offer more advanced features such as watchpoints, program trace and real time debug capabilities. In general terms, an OCD is a combination of hardware and software embedded onto the microprocessor chip, accessible through an interface port, and usually requiring an external debugger.

OCD infrastructures provide access to internal resources during system operation, being an excellent mechanism for modifying register and/or memory values, i.e. for *inserting faults*, and subsequently retrieving the data necessary to assess the effect of those faults. In most cases, OCD fault injection techniques rely on halting the processor, via control signals or breakpoints [18].

The major problem of on-chip debugging is the lack of a consistent set of capabilities and a standard communication interface across processor architectures. Standard ports (RS232, JTAG) are commonly used for the physical connection [19,20], but their capabilities vary widely. Several standardization efforts for OCD infrastructures and interfaces were initiated on recent years [21–23]. IEEE-ISTO 5001, *The Nexus 5001 Forum Standard for a Global Embedded Processor Debug Interface* [24], was the first of these efforts and is currently well documented and stable.

To better evaluate the advantages and limitations of real-time fault injection on NEXUS compliant microprocessors, preliminary work was performed using COTS devices. This approach was similar to other research works [8,25], and used a commercial target microprocessor and a debugger.

The obtained results confirmed most of the expected benefits and simultaneously identified some shortcomings, both in fault triggering and performance. It proved that it is possible to insert faults in memory without affecting the running application and to use the trace information as an effective means of analyzing the program flow, before and after fault activation. However, as the fault injection campaigns must be run on the host machine, the operating system (Windows or Unix) and physical connection to the NEXUS compliant debugger (Ethernet or USB) lead to long and non-deterministic memory access times. The consequence is the occurrence of experiments with inconclusive results, since in such cases the fault actually inserted does not emulate a single bit-flip as intended. Depending on the targeted memory area, the actual percentage of inconclusive fault insertions could be as high as 50%, requiring additional debugging and result analysis for validating each experiment.

The triggering source represents an additional source of problems. The use of trace data proved unreliable due to variable communication delays, making it necessary to use an external trigger signal. As a consequence, it was impossible to synchronize the fault insertion and the events of the running application.

To overcome the identified problems, three solutions were developed to enhance real-time fault injection capabilities: (1) a debugger customized for fault injection, (2) higher bandwidth between the debugger and the OCD, and (3) the migration of some capabilities into the OCD infrastructure itself.