S.A. Chekmarev¹, V.Kh. Khanov²
Siberian State Aerospace University named after Academician M.F. Reshetnev (SibSAU)
31 “Krasnoyarskiy Rabochiy” prospect, Krasnoyarsk, 660037, Russia.
¹ Engineers of SibSAU
² Ph.D., assistant professor of SibSAU
E-mail: khvkh@mail.ru

Abstract. The paper presents an on-chip debugging method for the injection of single faults in the processor cores of systems-on-chip. The method consists in the placement of faults injection infrastructure in a system-on-chip as an intellectual property core. This simplifies the fault injection environment, reduces delays injection and improves the performance, as well as allows doing long autonomous campaign for injection of faults without the use of external devices.

1. Introduction

Fault Tolerance (FT) is an important requirement for digital equipment working in hard conditions, such as in space. Today the majority of such equipment is based on microprocessors. The greatest danger for microprocessors working as a part of equipment systems on board spacecraft constitute Single Event Upset (SEU), caused by cosmic ionizing radiation. These events can change the logical values in the elements of the internal memory of the microprocessor, such as registers and cache [1]. SEU is the most frequent cause of faults, so efforts for the FT design of architecture microprocessors mainly focus on protection from these events.

Fault Injection (FI) is a widely accepted solution to measure the SEU sensitivity of microprocessors and other digital devices. There are several different FI solutions which have their benefits as well as disadvantages. Some solutions are universal; others are dependent on the capabilities of the System Under Test (SUT). The majority of modern microprocessors include circuitry On-Chip Debugger (OCD), which allows access to internal resources. This feature can be used to modify the registers and memory, providing a useful mechanism for FI.

Existing approaches to inject faults using OCD [2, 4, 5, 6, 8] have essential limitation: most of them use host PC and external to the SUT debugger associated with host PC for the FI campaigns, i.e. FI source is out of the SUT. This solution complicates the fault injection environment and limits the performance of FI campaigns. However microprocessors type system-on-chip (SoC) may contain the necessary infrastructure for the FI, so the injection will be generated inside the microprocessor. This simplifies the FI-system and reduces the time for performing a single injection and increases the productivity of FI-companies.

This paper presents a new fault injection approach to inject faults in SoC-microprocessor that supports an OCD. Hardware module FI which autonomously produces FI experiments without external devices is embedded in the SoC-microprocessor as an IP-core. This simplifies the implementation of fault injection environment, avoids the communication between the target system and the software debugging tool which operates in the host PC. The feasibility of the proposed mechanism has been proved injecting faults on a LEON3 processor. The analysis of the results shows that this solution is an efficient, fast and little intrusive method to evaluate SEU sensitivity in SoC-processors.

* The research was done with the financial support of the Ministry of Education and Science of the Russian Federation in accordance with the agreement № 14.574.21.0041, unique identifier RFMEF57414X0041

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Published under licence by IOP Publishing Ltd
2. Fault Injection methodologies via on-chip debugging

Embedded debugging capabilities, included in most current microprocessors, allow the access to their internal resources for reading and writing. This feature can be used for fault injection. Injecting a fault via a circuitry OCD involves the following steps: setting a breakpoint via the circuitry OCD and waiting for the program to reach the breakpoint; reading the value of the target location (a register or a memory word) via the circuitry OCD; manipulating this value and then writing a new, faulty value back to the target location; resuming the program execution via a command sent to the circuitry OCD.

In most cases, OCD -approach uses the external debugger related with OCD a JTAG, Nexus or BDM interfaces. In [2] the fault injection environment architecture using the standard IEEE JTAG 1149.1 [3] is presented. It consists of a computer, a specialized FI device and SUT having circuitry OCD. The FI device includes a fault injection controller, a fault list, and golden results, debug commands module and JTAG controller. This solution allows carrying out the functions of controlling injections, preparation of the planned events and storage of results of fault injections not in the external host computer but with the help of the specialized FI device. The advantage is the ability to conduct long-term autonomous multiple FI experiments without a host computer, for example, a thousand or more events. The disadvantage of this architecture, like the previous one, is the complexity of the FI campaign, as it requires an external specialized FI device.

In [4, 5, 6] presents several FI-system using a standardized approach to debugging NEXUS [7]. The microprocessor of SUT contains circuitry OCD compliant with Nexus. Nexus standard defines the requirements for the debugging device and the interface that connects the debugging device and circuitry OCD. The disadvantage of the system is that the debugging device compatible with the standard Nexus is not designed to make faults. Is why it is equipped with additional modules for FI. In another example, a part of the expanded functional typical for [4], namely, FI controller, is located directly in the circuitry OCD, which increases productivity. The disadvantage of these infrastructures FI is the complexity, because it includes an external debugging device. In addition, the last solution has high intrusive, which reduces its practical significance.

In [7] simulated SEU error injection via OCD in the internal memory of the processor LEON3 [8], which is a SoC, is presented. OCD of LEON 3, in this case called Debug Support Unit (DSU), has its own direct interface with an on-chip bus, and, consequently, an external host PC has direct access to it. To improve the performance of FI the host PC is connected to the SUT via a high speed channel, for example, JTAG or Ethernet. FI campaigns are fully managed by software of a host PC, which is a disadvantage.

The objective of the work is to develop a simple and productive infrastructure FI, which allows carrying out long-term stand-alone experiments to test fault tolerant microprocessors such as system on chip to single faults via OCD-method.

2. Proposed solution

The goal is achieved by integrating FI infrastructure in SoC-microprocessor system as a specialized IP-core. This infrastructure includes Fault Injector, containing Fault Injection Controller, Fault Generator Drive and Golden Results. The environment architecture is presented in Fig. 1 and is intended for FPGA implementation.

The process of faults injection in the internal memory of the microprocessor core occurs as follows. Host PC initiates the start of the experiment by injection into the internal memory of the processor core. Fault Generator determines the time, the address and the position of the internal memory cell for making faults. This action can be performed automatically by a uniform random law or can be read from Fault List, which is pre-loaded into the internal memory of the FPGA from host PC. Fault Injection Controller, using the information received from the Fault Generator, via OCD writes an error to the required bit and the desired address of internal memory by way of inverting its content. To make a fault the OCD stops the processor core, reads the desired internal memory cell, inverts the required bit and writes the result back. Then Fault Injection Controller and OCD, interacting by an on-chip bus, repeat the same steps similarly, when Fault generator determines the next injection. If the mechanism of error detection and correction detects a fault during the performing of the current running program by the processor core of internal memory, then depending on the result the signs «fault is detected and corrected» or «fault is detected and not corrected» are saved in internal memory status register.

Simultaneously the Fault Injection
Controller polls the internal memory status register for finding facts of faults detection. To do this, it gives appropriate instructions to OCD, which reads the status register of on-chip memory, and analyzes the impact of faults. If signs of the impact of faults are established, this information is recorded in the Golden Results. FI process continues until the end of Fault List, which has been pre-loaded, or until the host PC forcibly stops the Fault Generator and thereby stops FI-campaign.

![Fault injection environment architecture](image)

**Figure 1.** Fault injection environment architecture.

Upon completion of the experiment the results of injection of Golden Results are transferred to the host PC. If the injection is performed automatically by the Fault Generator, then the information on the number of faults introduced during the experiment is also transmitted to the host PC. The results obtained are processed in the host PC to assess the effectiveness of the mechanism of error detection and correction of internal memory of the processor.

The proposed solution allows for long-term experiments on the injection of faults of almost any number (hundreds, thousands, tens of thousands, hundreds of thousands events) due to the use of a Faults Generator. It allows for the realization of not only the random uniform law of injection, but also any other law, including that simulating the degradation of the system under test during its operation due to accumulation of doses of radiation from outer space.

Fault Injector is an independent IP-core not affecting the remaining parts of the microprocessor system. Therefore, after all the experiments Fault Injector can be removed, and thus a complete microprocessor system will not have it. If Fault Injector is left as a part of a system composed of microprocessors, it can be used to diagnose system fault tolerant in microprocessor system exploitation as, for example, an onboard computer of a spacecraft.

### 3. Experimental results

The effectiveness and applicability of the proposed approach was evaluated in an open processor LEON3, equipped with a fault tolerant mechanism based on Hsiao code [9]. The main experimental studies were carried out on the board of a processor module [10] based on A3PE3000 FPGA of MicroSemi company [11], developed for small spacecraft TabletSat-Avrora [12]. Implementation of the system of FI for LEON3 is presented in Fig. 2. OSD in LEON3 called DSU (Debug Support Unit) as previously noted. DSU as well as all the IP-core has access to the on-chip bus AMBA, but also directly related to the core.
processor. DSU described above introduces faults in registers and cache. Registers and caches are EDAC-module for error correction, and status registers to store the results of the current FI. Fault Injector refers to the processor core through the AMBA, and thereby ensured its weak connection with it and, therefore low intrusiveness. After finishing the all FI-companies needed to test a processor the Fault Injector is removed from the system leaving no trace of it.

![An experimental model of processing module](image)

**Figure 2.** Implementation of the system of FI for LEON3.

FI campaigns were carried out with the help of automatically generated random uniform sequence of injections and injection of pre-formed in the host PC and loaded into FPGA memory of fault list separately for the register file and for the processor cache. Ongoing campaigns limited total number of events (100, 500, 1000 and 2000) for limiting the intensity of faults 1, 10, 100 or more events per minute. Thus, to confirm the functionality and feasibility of the proposed approach has been a large number of experiments. The article provides only part of the results.

Table 1 shows the results of injection in the ALU register file of the processor. Fault Generator injection produced by a random uniform law. During the tests, the processor performing the test program that in the loop consistently appealed to the registers with read/ write operations. In the first register recorded value of 0x11111111. Next is his reading. When mismatch of the result is displayed error information. Then, in the next register writes the value of the previous register plus 0x11111111 and then the process was repeated.
In the present case detection of single faults has been corrected fault tolerance mechanism. However, not all injections were discovered - the percentage of detected faults is indicated in parentheses. This is due to that some of them occurred in the memory area, which did not contain actual data at the time of entering of failure or data were subsequently overwritten of processor, before he turned to these areas with the read operation. Thus, the experimental results depend on the generated fault and on the specifics of the software testing.

The evaluation of the developed architecture performance was made in comparison with the approach presented in [8], when the injection performing is fully guaranteed by the host PC software. The same configuration of LEON3 processor with the same EDAC on the basis of Hsiao code and the same experimental conditions were used for the injection performing.

Analysis of injection procedures in the internal memory of the processor LEON3 showed that the delay at the time of injection is.

$$t_d = \frac{N_0 + N_b}{F},$$

were $N_0 = 54$ – the number of cycles required to make one injections, $N_b = 7$ – the maximum number of clock cycles required for the release of the bus AMBA, when her employment by another device, $F$ – system frequency, i.e. $54 \pm 7$ system clock cycles. When the system frequency is 25 MHz, the delay will be $2160 \pm 280$ ns. Floating value of the time delay associated with the feature of on-chip bus (at the start of injection the bus may be occupied by another master, which is part of a system-on-chip). However, you can ensure that the injection is performed for 61 clock cycle, allowing you to accurately predict the moments of the injection. Thus, compared with the approach shown [8] when the injection is completely provided software of host PC, proposed method allows to control the course of the FI experiment up to 280 nsec. This is achieved due to the lack of dependence on the host PC software and there is no need to exchange...
information on the external interface. These arguments allow us to classify proposed method of fault injection as real-time.

To evaluate the redundancy considered architecture has been implemented for a different number of faults and for FPGA produced by the companies Microsemi and ALTERA. The results presented in Table 3 show that the proposed approach is not resource-consuming. With the increasing size of the list of events the number of logic cells necessary for the implementation of the method remained virtually unchanged. The need for on-chip memory for multiple programmable and single programmable flash-FPGA of Microsemi company increases faster. This is primarily due to the characteristics of FPGA data. However, the application of the function of FI automatic generation in the proposed method allows reducing the required amount of on-chip memory FPGA. Fault Generator can be automatically initialized for entering new portion of injections. Thereby the need for memory blocks reduces.

### TABLE 3. Results of the synthesis of a faults injector in FPGAs

| Size of Fault List | Microsemi A3PE3000L | Microsemi RTAX 1000S | ALTERA Cyclone IV |
|-------------------|---------------------|---------------------|------------------|
|                   | Cells | FFs | Cells | FFs | Cells | FFs |
| 100               | 1285  (1.7%) | 2 (1.7%) | 301 (1.5%) | 2 (5%) | 370 (<1%) | 2.176 (<1%) |
| 500               | 1301  (1.7%) | 3 (2.6%) | 386 (2.1%) | 3 (8%) | 369 (<1%) | 8 (<1%) |
| 2000              | 1337  (1.7%) | 9 (8.0%) | 448 (2.4%) | 9 (25%) | 379 (<1%) | 34 (<1%) |

### 4. Conclusion

This paper presents a new fault injection method aimed at measurement of SEU sensitivity in SoC-microprocessors. The proposed solution is to integrate FI-infrastructure in SoC-microprocessors as a specialized IP-core. This enables to make FI-campaigns which do not need the debugging software completely autonomous from the host PC that in its turn makes the fault injection process faster. This solution is suitable for SoC-microprocessors in which OCD circuitry has its own direct interface with the on-chip bus. IP-core of FI-infrastructure is removed from the SUT at the end of the debugging.

Experimental results on an LEON3 processor have been presented. They illustrate such capabilities of the approach as long-term experiments conducted with any number of failures, small time delay of injection, and low resources consumption of SUT.

### References
1. S. Mukherjee, “Architecture Design for Soft Errors”, Elsevier, 2008 – 337 p.
2. M. Portela-Garcia, C. Lopez-Ongil, M. Garcia-Valderas, L. Entrena, “A Rapid Fault Injection Approach for Measuring SEU Sensitivity in Complex Processors”, IOLTS 2007, 13th IEEE International On-Line Testing Symposium – Heraklion, Crete, Greece – July 8 to July 11, 2007– pp.101-106.
3. “IEEE Standard Test Access Port and Boundary-Scan Architecture”, IEEE Std 1149.1, 2001 – 212p.
4. A. Fidalgo, G. Alves, J. Ferreira, OCD-FI: on-chip debug and fault injection, in: International Conference on Dependable Systems and Networks, June 2006 – pp. 214-219.
5. A. Fidalgo, J. Ferreira, “Using NEXUS Compliant Debuggers for Real Time Fault Injection on Microprocessors”, SBCC’06, 19th Symposium on Integrated Circuits and System Design, Minas Gerais, Brazil – August 28 - September 01, 2006 – pp. 214-219.
6. A. Fidalgo, M. Gerigota, G. Alves, J. Ferreira. “Real-time fault injection using enhanced on-chip debug infrastructures”, Microprocessors and Microsystems № 25, 2011– pp. 441-452
7. “The Nexus 5001 Forum Standard for a Global Embedded Processor Interface version 2.0”, IEEE-ISTO 5001, 2003 – 166p.
8. J. Gaisler, “LEON3-FT-RTAX SEU test results Gaisler Research”, Issue 1, December 7, 2005 – 8p.
9. “SPARC V8 32-bit Processor LEON3/ LEON3-FT Companion Core Data Sheet Template Design”, Gaisler Research, 2010 – 41 p.
10. M. Y. Hsiao, “A class of optimal minimum odd-weight column SEC-DED codes”, IBM J. Res. Develop., vol. 14, no. 4, Jul. 1970 – pp. 395-401.
11. V.Kh. Khanov, S.A. Chekmarev, V.A. Shakhmatov, M.Y. Vergasov, F.A. Lukin, “Development of equipment information exchange system of onboard control complex small spacecraft” [“Razrabotka apparatuty sistemy informatsionnogo obmena malogo kosmicheskogo apparata”], Vestnik SBGUA. no. 5 (49), 2013, pp. 149-153 (In Russ.).
12. “ProASIC3 FPGA Overview”, available at: http://www.microsemi.com/products/fpga-soc/fpga/proasic3-overview, (last visited: January 2014).
13. “Sputnix”, available at: http://www.sputnix.ru/ru, (last visited: March 2014).