Evaluating the SEE sensitivity of a 45nm SOI Multi-core Processor due to 14 MeV Neutrons
Pabo Ramos, Vanessa Vargas, M. Baylac, F. Villa, S. Rey, Juan Antonio Clemente, Nacer-Eddine Zergainoh, Jean-François Méhaut, Raoul Velazco

To cite this version:

Pabo Ramos, Vanessa Vargas, M. Baylac, F. Villa, S. Rey, et al.. Evaluating the SEE sensitivity of a 45nm SOI Multi-core Processor due to 14 MeV Neutrons. IEEE Transactions on Nuclear Science, Institute of Electrical and Electronics Engineers, 2016, 63 (4), pp.2193 - 2200. <10.1109/TNS.2016.2537643>. <hal-01280648>
Evaluating the SEE sensitivity of a 45nm SOI Multi-core Processor due to 14 MeV Neutrons

Pablo Ramos, Vanessa Vargas, Maud Baylac, Francesca Villa, Solenne Rey, Juan Antonio Clemente, Nacer-Eddine Zergainoh, Jean-François Méhaut and Raoul Velazco

Abstract—The aim of this work is to evaluate the SEE sensitivity of a multi-core processor having implemented ECC and parity in their cache memories. Two different application scenarios are studied. The first one configures the multi-core in Asymmetric Multi-Processing mode running a memory-bound application, whereas the second one uses the Symmetric Multi-Processing mode running a CPU-bound application. The experiments were validated through radiation ground testing performed with 14 MeV neutrons on the Freescale P2041 multi-core manufactured in 45nm SOI technology. A deep analysis of the observed errors in cache memories was carried-out in order to reveal vulnerabilities in the cache protection mechanisms. Critical zones like tag addresses were affected during the experiments. In addition, the results show that the sensitivity strongly depends on the application and the multi-processing mode used.

Index Terms—Accelerated testing, AMP, Multi-core, SEE, SEFI, SEU, Soft Error, SOI, SMP

I. INTRODUCTION

A VIONICS and spacecraft applications require determinism and robustness in their reactive embedded systems. The current technological trend in embedded systems is the use of multi-core processors in order to satisfy the growing demand of performance and reliability without a critical increase of power consumption. The inherent redundancy capability of multi-core architectures makes them ideal for implementing fault-tolerant mechanisms [1]. Moreover, these devices provide a great flexibility because they allow implementing different multi-processing modes and programming paradigms. Hence, avionics and spacecraft industries are interested in validating the use of multi-core and many-core devices for their applications [2], [3].

This work was supported in part by e2v, a Freescale partner, http://www.e2v.com, by the Secretaría de Educación Superior Ciencia Tecnología e Innovación del Ecuador (SENESCYT), by the Spanish Ministry of Education, Culture and Sports project TIN2013-40968-P, and by the "José Castillejo» mobility grant for professors and researchers.

P. Ramos and V. Vargas are with the Université Grenoble-Alpes & TIMA Labs, Grenoble (France), and with the Universidad de las Fuerzas Armadas ESPE, DEEE, Sangolquí, Ecuador, e-mail: {pframos, vcvargas}@espe.edu.ec.

M. Baylac, F. Villa and S. Rey are with Laboratoire de Physique Subatomique et de Cosmologie LPSC, Université Grenoble-Alpes & CNRS/IN2P3, Grenoble, France, e-mail: {maud.baylac, francesca.villa, solenne.rey}@lpsc.in2p3.fr.

J. A. Clemente is with the Computer Architecture Department, Facultad de Informática, Universidad Complutense de Madrid (UCM), Spain, e-mail: ja.clemente@fidi.ucm.es.

N.E. Zergainoh and R. Velazco are with TIMA Labs,(Université Grenoble-Alpes & CNRS respectively), Grenoble (France), e-mail: {nacer-eddine.zergainoh, raoul.velazco}@imag.fr.

J.F. Méhaut is with LIG Labs., Université Grenoble-Alpes & CNRS, Grenoble (France), e-mail: jean-francois.mehaut@imag.fr.

The continuous technology scaling in integrated circuits makes them more sensitive to the effects of natural radiation such as Single Event Effects (SEEs) [4]. For this reason, physical designers are continuously searching for new methods to improve manufacturing technologies to reduce SEE consequences. For instance, Silicon-On-Insulator (SOI) technology has been proved to be less sensitive than CMOS bulk technology [5].

On the other side, additional hardware implementations have been added to multi-core architectures for improving their reliability. Examples of these protection mechanisms are the implementation of Error Correcting Code (ECC) and parity in cache memories. Hamming codes are very useful to mitigate Single Event Upsets (SEUs) since they can detect double errors and correct single ones. Nevertheless, new hardware introduces an extra area with the corresponding increase in power consumption and performance degradation [6].

The use of cache memories reduces the memory access time, thereby increasing substantially the performance of the system. However, enabling caches implies the increase of sensitive area and thus, the reduction of system reliability. Therefore, a trade-off between performance and reliability is needed depending on the application.

Given the significant importance that represents the use of multi-core and many-core processors for avionics and safety-critical applications, it is mandatory to evaluate their sensitivity to SEEs, and particularly to SEUs. The present work aims at evaluating the sensitivity to neutron radiation of a 45nm SOI multi-core processor. Two main contributions are presented: The first one is the determination of the static cross section of a device implementing Error Correcting Code (ECC) and parity (that cannot be deactivated) in their caches. The second one is the evaluation of the neutron radiation sensitivity of the studied multi-core working in two different multi-processing modes running parallel applications.

For achieving the mentioned contributions, static and dynamic tests were carried out with a neutron beam of 14 MeV to obtain the corresponding cross-sections. Concerning the dynamic tests, two operating modes were explored. The Asymmetric Multi-Processing Mode (AMP) without operating system (OS) was used in order to test specific sensitive hardware resources, such as cache memories and registers. On the other hand, the Symmetric Multi-Processing Mode (SMP) was implemented to test the resources used by the embedded Linux SDK V1.6 OS. Experimental results show the relationship between the system reliability and the multi-processing mode used.
A preliminary version of this work was presented in [7]. The present work provides a detailed description of the adopted approach, the results regarding the errors not detected by the studied multi-core architecture, and a deep analysis of the experimental results including the probable causes of the observed errors.

The remainder of the paper is organized as follows: Section II presents the related work. Section III describes the adopted approach that has been used to evaluate the intrinsic sensitivity of the multi-core and its dynamic response. Section IV details the experimental setup. Section V presents and analyzes the results issued from neutron ground testing. Finally, Section VI concludes the paper and provides some directions for future work.

II. RELATED WORK

Several interesting works dealing with the sensitivity of electronic components can be found in the literature. Reference [8] summarizes the sensitivity to SEE induced by neutrons of different integrated circuits (i.e., SRAMs, microprocessors and FPGAs) applicable to avionics. However, there are very few works available regarding multi-core and many-core processors sensitivity.

In [9] is presented a significant work that establishes a dynamic cross-section model for a multi-core server based on quad-core processors in 45nm bulk CMOS technology. Also, it provides a fault handling comparison between Windows 5.2 and Linux 5.1 operating systems.

Reference [10] presents the radiation sensitivity evaluation of a modern Graphic Processing Units (GPUs) designed in 28nm technology node, and composed by an array of streaming multi-processors which share the L2 cache memory. It also provides a hardening strategy based on Duplication with Comparison.

Authors of [11] propose to disable the cache memories of high-end processors in safety-critical applications in order to gain in reliability in spite of the increase of the execution time. An accurate analysis of the effects of soft errors in the instruction and data caches is also presented.

The work presented in [12] demonstrates that, by enabling L1 cache, it is possible to improve the performance of the system without compromising the reliability. A generic metric (Mean Workload Between Failures) taking into account both cross section and exposure time was introduced to evaluate the reliability of a embedded processor devoted to execute safety-critical applications.

III. ADOPTED APPROACH

A. Static Sensitivity

To estimate the intrinsic sensitivity of the accessible memory cells of the multi-core processor, the neutron static cross-section $\sigma_{STATIC}$ was obtained. Typically, the method used to obtain $\sigma_{STATIC}$ consists in writing a predefined pattern in the memory locations and registers, and checking it along the radiation experiments to detect errors.

However, since the cache memories on the target device implement protection mechanisms that cannot be deactivated, this method is not suitable as it is. This can be explained due to the fact that single-bit errors in a word are not visible while reading memory locations because whenever they occur, they are corrected by either the ECC or the cache invalidation mechanisms. It was thus necessary to use a complementary technique based on machine-check error report for logging data that have been corrupted during the radiation experiments.

In processors including machine-check error report, it is possible to enable an interrupt routine for reporting errors. The information about the errors is saved in some special-purpose registers of the device. By reading these registers, one can know the type of error occurred, address, data, as well as the obtained and calculated ECCs.

For this test, the multi-core processor was configured in AMP mode without OS in order to have independence in the execution of each core when performing the self-testing of their cache memories. In addition, the L1, L2 and L3 caches, as well as the machine-check error report of each core must be enabled. In order to simplify the interpretation of the results due to cache coherence mechanisms, the self-testing application was configured so that each core reads from and writes to different sections of the main memory. Each section has the same size as the L2 cache. In the particular case of the L3 cache, only the core 0 was configured to use it, preventing other cores to access it.

B. Dynamic Response

For evaluating the reliability of the target device when an application is running, the dynamic cross section ($\sigma_{DY N}$) has been obtained. The motivation is to observe to what extent the dynamic chip response depends on the application and the multi-processing paradigm implemented. Thus, the SMP and AMP multi-processing modes were both adopted in this experiment.

In SMP mode, a single OS that runs on all the cores is responsible for achieving parallelism in the application. It dynamically distributes the tasks among the cores, manages the organization of task completion, and controls the shared resources. In AMP mode, the cores run independently of each other, with or without OS. Also, they have their own private memory space, although there is a common infrastructure for inter-core communications. Hence, AMP mode is very useful.
when working with embedded systems [13]. Figure 1 depicts these two Multi-Processing modes.

Concerning dynamic tests, two different scenarios were considered. On the one hand, a memory-bound application was implemented when the processor operates in AMP mode without OS to evaluate the sensitivity of memory resources. On the other hand, a CPU-bound application was implemented when the processor operates in SMP mode in order to maximize the use of CPU resources and scheduling. In both cases, errors detected by the application and by the machine check-error report were considered in order to evaluate the sensitivity of the target device. In the SMP scenario, it was necessary to modify the original traps code in the kernel and in the u-boot of the Linux OS. The traps code is the code that is executed by the system when an exception or a fault occurs. In this approach, this section of code was modified to log all the events detected by the machine-check error report. Also, in the case of a L2 cache error detection, the L2 error registers values were logged to obtain more details about the error. It is important to note that, the original traps code logs the recoverable and unrecoverable conditions that cause a machine-check exception. If the condition is recoverable by the machine check, then it returns to the previous state and its operation is resumed.

IV. EXPERIMENTAL SETUP

A. Neutron Radiation Facility

The radiation ground tests were conducted at the GENEPI2 (GEnerator of NEutron Pulsed and Intense) facility located at the LPSC (Laboratoire de Physique Subatomique et Cosmologie) in Grenoble, France [14]. This accelerator was originally developed for nuclear physics experiments, and recently it has been used to irradiate integrated circuits from different technologies.

From the target, neutrons are emitted in all directions. The Device Under Test (DUT) is set facing directly the target at a distance determined to adjust the neutron flux. While the DUT is fully exposed to neutrons, a dedicated neutron shielding can be used to protect the readout electronic platform.

Neutrons are produced with an average energy of 14 MeV. For the radiation campaigns, it was considered, to first approximation, that only neutrons emitted fully forward will impact the DUT. In this case, the neutron energy is maximal at 15 MeV. Reference [15] discusses the relevance of using 14 MeV neutron test to characterize the SEU sensitivity of digital devices.

Neutron production is monitored throughout the experiments to determine the neutron dose for each irradiation. An online Si detector, located within the beam pipe 1 meter upstream of the target, collects the recoil particles backscattered from the target during the fusion reaction.

Early 2015, a fresh T target was installed, generating a maximum neutron flux of $4.5 \times 10^7 \text{ n.cm}^{-2} \cdot \text{s}^{-1}$. For the radiation tests presented in this work, the flux was limited to $2 \times 10^5 \text{ n.cm}^{-2} \cdot \text{s}^{-1}$.

B. Device Under Test

The target device was a Freescale P2041 multi-core processor which is inside the P2041RDB design board [16]. The multi-core is based on four e500mc cores built on Power Architectures technology and manufactured in 45nm SOI technology. This quad-core can operate up to 1.5 GHz and includes a three-level cache hierarchy. The e500mc core is a 32-bit superscalar processor that includes independent on-chip 32 KB L1 caches for instruction and data, and a unified 128 KB backside L2 cache. Additionally, the P2041 includes a 1024 KB L3 cache shared among the four cores. Table I gives details about the sensitive areas of the multi-core processor that were targeted during the radiation campaigns.

The e500mc processor implements an L1 instruction and data cache with automatic cache invalidation when a parity error is detected. Both the L2 backside cache and the L3 shared frontside cache are protected with configurable ECC or parity for the data array, and parity for the tag array. This architecture corrects single bit errors and detects multiple-bit ones [17]. Figure 2 depicts the memory architecture of the studied multi-core processor.

Table I

<table>
<thead>
<tr>
<th>Sensitive zone</th>
<th>Location</th>
<th>Capacity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Cores 0, 1, 2, 3</td>
<td>32 KB / D and 32 KB / I per core</td>
<td>Data / Instruction Cache</td>
</tr>
<tr>
<td>L2</td>
<td>Cores 0, 1, 2, 3</td>
<td>128 KB per core</td>
<td>Backside Unified Cache</td>
</tr>
<tr>
<td>L3</td>
<td>Multi-core</td>
<td>1024 KB per chip</td>
<td>Frontside cache</td>
</tr>
<tr>
<td>GPR</td>
<td>Cores 0, 1, 2, 3</td>
<td>32 registers of 32 bits</td>
<td>General purpose register</td>
</tr>
<tr>
<td>FPR</td>
<td>Cores 0, 1, 2, 3</td>
<td>32 registers of 64 bits</td>
<td>Floating point register</td>
</tr>
</tbody>
</table>
can be invalidated and repopulated with the valid data from the rest of the memory hierarchy [17]. For logging all the SEEs occurred during the radiation experiments, the machine-check error interrupt and the Cache Error Checking bits were enabled.

C. Tested Applications

In the first scenario, for evaluating the dynamic response of the studied multi-core processor, a memory-bound 80x80 Matrix Multiplication (MM) algorithm was implemented. In this case, the device was configured in AMP mode, where each core executes independently the same matrix multiplication (C = AxB) and compares its results with a predefined value in order to identify errors. The size of the matrix was selected in order to maintain a trade-off between the amount of memory used and the execution time. The matrices A, B and C were located in consecutive memory vectors. Matrix A was filled up with 1’s and B was filled up with 2’s, thus the expected result was 160 for all the elements of matrix C. The matrices were filled up with fixed values in order to simplify the data analysis since a known value helps to identify which bit or bits have been changed during the test. In this way, MBUs (Multiple Bit Upsets) and MCUs (Multiple Cell Upsets) can be easily detected. It is important to note that the results of the experiment are totally independent of the input values, no matter the particle produces a bit flip in a fixed or random value.

In the second scenario, the CPU-bound application Traveling Salesman Problem (TSP) for 16 cities was implemented. Its execution was distributed among all the cores. The multi-core processor was configured in SMP mode in which the OS manages the resources in order to maximize the processing capacity of the cores. The parallel implementation of the TSP application makes this benchmark intrinsically fault-tolerant since, if one core is stopped by any reason, another core could find the correct result. The source code of this parallel implementation was an adapted version of the one used in [18]. An application based on Linux PTRACE-Process trace functions was implemented to monitor parallel applications and their related processes.

V. EXPERIMENTAL RESULTS

A. Static Cross Section

A first radiation campaign was carried out for obtaining the multi-core static cross-section. The device under test was placed facing the center of the target perpendicularly to the beam axis at a distance of \( \sim 19.1 \text{ cm} \). The neutron beam energy was \( 14MeV \) with a flux of \( \sim 1.96 \times 10^5 \text{ n} \cdot \text{cm}^{-2} \cdot \text{s}^{-1} \) at a \( 500 \text{ Hz} \) frequency. The cross section is defined as:

\[
\sigma = \frac{\text{Number of Upsets}}{\text{Fluence}}
\]

(1)

58 SEEs were detected within 2 hours of exposure time. Among them, 46 were SEUs and 12 Single Event Functional Interrupts (SEFIs). There were no errors in general and floating point registers. Table II summarizes the results of this campaign.

In this experiment, all the SEEs were considered errors no matter they were detected by the machine-check error report or by the self-testing application. Then, the obtained static cross-section is:

\[
\sigma_{\text{STATIC}} = \frac{58}{1.41 \times 10^6} = 4.11 \times 10^{-8} \text{ cm}^2 \text{ device}
\]

(2)

Due to the scarcity of experimental data (58 SEEs), it is compulsory to add uncertainty margins to these results. For numerous events (typically \( \geq 100 \)), the Poisson distribution can be used to calculate such margins. However, in this situation the most accurate and universal way to calculate the uncertainty margins consists in using the relationship between the cumulative distribution functions of the Poisson and chi-squared distributions as described in [19], [20]. Therefore, the following equation has been applied:

\[
\frac{1}{2} \chi^2(\frac{\alpha}{2}, 2N_{err}) < \mu < \frac{1}{2} \chi^2(1 - \frac{\alpha}{2}, 2(N_{err} + 1))
\]

(3)

where \( \chi^2(p, n) \) is the quantile function of the chi-square distribution with \( n \) degrees of freedom, \( \alpha \) is a parameter that defines the \( 100(1-\alpha) \) percent confidence interval, and \( N_{err} \) is the number of errors detected.

For a 95 \% confidence interval (\( \alpha = 0.05 \)), the lower and upper limits for the static cross section are:

\[
3.12 \times 10^{-8} \text{ cm}^2 \text{ device} < \sigma_{\text{STATIC}} < 5.32 \times 10^{-8} \text{ cm}^2 \text{ device}
\]

(4)

Since the accessible registers and memory cells of the multi-core processor represent about \( 1.47 \times 10^7 \) bits, the confidence interval for the static cross section per bit is estimated as \( [2.12 - 3.62] \times 10^{-15} \text{ cm}^2 / \text{bit} \). Reference [9] provides the estimation of the bit cross section for a 45nm CMOS technology processor \( (1 \times 10^{-14} \text{ cm}^2 / \text{bit}) \) for neutrons with the same energy. From these results, it can be seen that 45nm SOI technology is between three and five times less sensitive to SEEs than its CMOS counterpart.

Errors in L1, L2 and L3 caches, both in data arrays and cache tags were detected by the machine-check error report. In addition, it was observed a SEFI (depicted in Table II as "Other errors") that provoked a system hang simultaneously in all the cores. This event lead to several errors logged by the self-testing application running on the processors that showed data different from the original word \( (0 \times 55.AA55.AA) \) written in the memory. From these errors, two types of patterns were identified.

<table>
<thead>
<tr>
<th>SEE Type</th>
<th>Type of error</th>
<th>Occurrences</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEU</td>
<td>L1 Data parity</td>
<td>9</td>
<td>None</td>
</tr>
<tr>
<td>SEU</td>
<td>L2 Single-bit ECC</td>
<td>29</td>
<td>None</td>
</tr>
<tr>
<td>SEFI</td>
<td>L2 Tag parity</td>
<td>5</td>
<td>Hang</td>
</tr>
<tr>
<td>SEU</td>
<td>Multiple L2 errors</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>SEU</td>
<td>L3 Single-bit ECC</td>
<td>7</td>
<td>None</td>
</tr>
<tr>
<td>SEFI</td>
<td>L3 Multiple-bit ECC</td>
<td>6</td>
<td>Hang</td>
</tr>
<tr>
<td>SEFI</td>
<td>Other errors</td>
<td>1</td>
<td>Hang</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>58</td>
<td></td>
</tr>
</tbody>
</table>

Table II: Results of the Static Radiation Campaign
The first one consists of a set of fourteen words with consecutive addresses containing 0xDEADBEEF as data. Table III presents the main memory space used by each core (Columns 2 and 3) and the address ranges where this pattern was replicated. The second pattern constitutes scattered clusters of errors of four words each. In each of them, the first word contained 0xDEADBEEF, the second one 0x200000, the third one 0x01300000 and the last one 0x00006000. Table IV summarizes the replications of this pattern, as well as the involved addresses.

### Table III
**Main Memory Space and Details of the First Pattern of Errors During the Static Test**

<table>
<thead>
<tr>
<th>Core</th>
<th>Start Addr</th>
<th>End Addr</th>
<th>Range 1</th>
<th>Range 2</th>
<th>Range 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x10000</td>
<td>0x30000</td>
<td>0x16548 - 0x16570c</td>
<td>0x14c8 - 0x14fc</td>
<td>0x16a8 - 0x16af</td>
</tr>
<tr>
<td>1</td>
<td>0x40000</td>
<td>0x60000</td>
<td>0x46048 - 0x46070c</td>
<td>0x46c8 - 0x46fc</td>
<td>0x46908 - 0x4693c</td>
</tr>
<tr>
<td>2</td>
<td>0x70000</td>
<td>0x90000</td>
<td>0x76048 - 0x76070c</td>
<td>0x76c8 - 0x76fc</td>
<td>0x76908 - 0x7693c</td>
</tr>
<tr>
<td>3</td>
<td>0x100000</td>
<td>0x120000</td>
<td>0x106048 - 0x106070c</td>
<td>0x106c8 - 0x106fc</td>
<td>0x106908 - 0x10693c</td>
</tr>
</tbody>
</table>

Due to the fact that errors have occurred simultaneously and the observed pattern is repeated among the cores, it is presumed that a particle perturbed a shared resource of the chip. Because of the nature of these errors, it is suggested that the affected resource was a register belonging to the CoreNet Coherency Fabric (CCF), which is the connectivity infrastructure of the multi-core system.

### B. Sensitivity of the P2041 in AMP Scenario

A second radiation test campaign was carried out to obtain the dynamic cross section in AMP scenario (\(\sigma_{\text{DY}_N,\text{AMP}}\)). The device was again set at a distance of \(\sim 19.1\ cm\) from the target. The neutron energy was 14 Mev with a flux of \(\sim 1.96 \times 10^{5} \text{ n} \cdot \text{cm}^{-2} \cdot \text{s}^{-1}\) at a 500 Hz frequency. Two tests, each one lasting 2 hours were performed.

Table V shows that L1, L2 and L3 caches were all perturbed by neutrons. The Load Instruction and Instruction fetch errors are the most critical ones since they provoked processor hang.

### Table IV
**Details of Second Pattern of Errors During the Static Test**

<table>
<thead>
<tr>
<th>Core</th>
<th>No. Occurrence</th>
<th>1st Word Addr</th>
<th>2nd Word Addr</th>
<th>3rd Word Addr</th>
<th>4th Word Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0x10000</td>
<td>0x10004</td>
<td>0x10008</td>
<td>0x10024</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0x16000</td>
<td>0x16004</td>
<td>0x16008</td>
<td>0x16024</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0x16200</td>
<td>0x16204</td>
<td>0x16208</td>
<td>0x16224</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0x16600</td>
<td>0x16604</td>
<td>0x16608</td>
<td>0x16624</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0x42000</td>
<td>0x42004</td>
<td>0x42008</td>
<td>0x42024</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0x46380</td>
<td>0x46384</td>
<td>0x46388</td>
<td>0x463a4</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0x46440</td>
<td>0x46444</td>
<td>0x46448</td>
<td>0x46464</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0x72000</td>
<td>0x72004</td>
<td>0x72008</td>
<td>0x72024</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0x763c0</td>
<td>0x763c4</td>
<td>0x763c8</td>
<td>0x763e4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0x76440</td>
<td>0x76444</td>
<td>0x76448</td>
<td>0x76464</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0x7c000</td>
<td>0x7c004</td>
<td>0x7c008</td>
<td>0x7c024</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>0x1020000</td>
<td>0x1020004</td>
<td>0x1020008</td>
<td>0x102024</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>0x1063800</td>
<td>0x1063840</td>
<td>0x1063880</td>
<td>0x1063a40</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0x1064400</td>
<td>0x1064440</td>
<td>0x1064480</td>
<td>0x1064640</td>
</tr>
</tbody>
</table>

Half of the observed L2 Tag parity errors lead to processor hang. L1 Data cache parity errors are not critical since L1 cache is invalidated when parity fails. Finally, L2 and L3 Single-bit errors are not critical as the ECC corrects them.

Briefly, there were one SEFI in Test 1 and five SEFIs in Test 2 that caused system hangs. In addition, six events in Test 1 provoked errors in the results of the application, but they were not detected by the multi-core machine-check error report. This puts in evidence that errors were produced by Multiple Bit Upsets (MBUs) involving not only data, but also parity information. A deeper analysis has allowed to identify the origin and multiplicity of these events. Four of them were clusters of errors whereas the other two were single data errors.

1) Clusters of Errors: Three clusters of errors occurred in Core 2, and one in Core 1. All of them were very closely related and they were detected in the same read cycle. Each cluster involves exactly 16 consecutive positions of the resulting matrix. Each matrix element was an integer value (4 bytes). In all cases, an incorrect result of "2" was observed instead of the expected "160".

Considering that:
- The e500mc processor features a set associative L1 cache memory organized as 64 sets of 8 blocks with 64 bytes in each cache line.
- The L2 cache memory is organized as 256 sets of 8 blocks of 64-byte cache lines [16].
- The number of consecutive corrupted addresses exactly matches the size of the cache line in the processor architecture.
- The physical addresses involved in each cluster correspond to a cache block.

Then, it is clear that the cluster of errors was produced by an upset affecting the cache address tag. It could be explained as follows: Upon reading the involved addresses which have Line Tag T stored in Set S, the cache hardware retrieves incorrect data instead of fetching the correct values from the main memory because a tag belonging to this set S was corrupted and became that precise tag T. Figure 3 depicts the clusters of errors observed in Core 2 assuming that the particle affected the L1 cache. Two of them had line tag 0 × 403DD and the other one 0 × 403DE.

The persistence of 2’s in these errors indicates that the
cache had already been filled up with the contents of matrix $B$. Taking into account the data address mapping shown in Figure 3 (a), any line tag comprised in the interval $(0 \times 403D6 - 0 \times 403DC)$ (matrix $B$) could have become the cluster error line tag. Comparing the tags of the clusters of errors with each one of the tags in the previous interval, it was possible to detect a MBU affecting bits $b_1$ and $b_2$ due to their physical adjacency. For the three cases the tags had to be changed (from $0 \times 403DB$ to $0 \times 403DD$ and from $0 \times 403DB$ to $0 \times 403DE$). These errors were not detected by the parity protection mechanisms since parity bit remains the same. Note that the L1 cache implements only one parity bit per tag.

Thus, in the authors’ opinion, a particle modified two consecutive bits (MBU) belonging to three different tags (Multiple Cell Upset with multiplicity of three). Moreover, when decoding the corrupted addresses, it was possible to determine that the cache lines in Sets $0 \times 1A$, $0 \times 1E$ and $0 \times 20$ were affected. The fact that even and quasi-consecutive sets in cache were involved, gives clues about the possible 3-D implementation of the caches.

Finally, the cluster of errors observed in Core 1 appears in the line tag $0 \times 203DD$ set $0 \times 1B$. Following the previous analysis it is possible to verify that the particle has also changed the bits $b_1$ and $b_2$ of the line tag $0 \times 203DB$ set $0 \times 1B$ becoming the line tag $0 \times 203DD$. This perturbation in the cache was not detected since the parity remains the same. This cluster of errors may have been produced by a MBU, or it was probably related to the clusters of errors occurred in Core 2 due to their similarities, in which case the mentioned MCU would have multiplicity of 4.

2) Single Errors: Two separated matrix-result data were corrupted from “160” to “162” in Core 2 at addresses $0 \times 403DF380$ and $0 \times 403DF480$ respectively. Since the same bit ($b_3$) was corrupted in both addresses and the difference between them is $0 \times 100$, it is very likely that they constitute an MCU. Also, this distance suggests that memory interleaving probably involves memory blocks of 256 addresses. These events were not detected by the parity protection which indicates that the parity bit was corrupted as well. Note that the L1 data cache implements one-bit-per-byte parity checking.

To conclude, the occurrences of application errors and hangs are evidences that the ECC and Tag parity mechanisms are not enough to guarantee the immunity of the cache memories.

The errors obtained during tests 1 and 2 described in Table V were added in order to have the total number of errors occurred within 4 hours of irradiation. The device was exposed to a fluence of $2.82 \times 10^{9} \text{cm}^{-2}$. In this experiment the total number of SEE was 90 and, among them 12 produced errors (application errors and hangs). Applying Equations (1) and (3) for a confidence level of 0.95, the dynamic cross-section in AMP scenario without OS is:

$$2.17 \times 10^{-9} \text{cm}^{2} < \sigma_{DYN, AMP} < 7.33 \times 10^{-9} \text{cm}^{2}$$

C. Sensitivity of the P2041 in SMP Scenario

Three additional radiation campaigns were carried out to calculate the dynamic cross section in SMP scenario ($\sigma_{DYN,SMP}$). In the first campaign the code was loaded from the NOR flash memory provided in the design board. During the test, the application definitely stopped due to a fatal crash in the OS after only 22 minutes. When a reboot of the system was performed, the image of the OS could not be loaded since the NOR flash memory was corrupted as well. Once, the OS image was restored, the test continues but a fatal crash in the OS occurred again after 28 minutes, giving a total test time of 50 minutes. That is the reason why for the other two campaigns the OS image was loaded from a hard disk. The second and the third campaign lasted one and four hours respectively. Table VI shows the characteristics of the radiation test campaigns.

Table VII summarizes the obtained results for the three tests. The fault classification was done based on the OS fault-handling messages and the monitor application. Faults with multiple indications were scored at the most critical level. The
order of the rows in this table depends on the criticality of the fault, being the last one the most critical. It is important to note that the results include the messages obtained during the execution of the application and during the idle time.

### Table VI

**Test Campaigns Characteristics for SMP Scenario**

<table>
<thead>
<tr>
<th>Test Campaign</th>
<th>Flux ([n \cdot \text{cm}^{-2} \cdot \text{s}^{-1}])</th>
<th>Time ([\text{min}])</th>
<th>Fluence ([n \cdot \text{cm}^{-2}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>(\sim 1.96 \times 10^5)</td>
<td>50</td>
<td>(6.00 \times 10^5)</td>
</tr>
<tr>
<td>Test 2</td>
<td>(\sim 1.62 \times 10^5)</td>
<td>60</td>
<td>(5.83 \times 10^5)</td>
</tr>
<tr>
<td>Test 3</td>
<td>(\sim 1.45 \times 10^5)</td>
<td>240</td>
<td>(20.88 \times 10^5)</td>
</tr>
</tbody>
</table>

### Table VII

**Results of Radiation Experiments for the P2041 Working in SMP Scenario**

<table>
<thead>
<tr>
<th>SEE Type</th>
<th>Type of OS Fault</th>
<th>Test1</th>
<th>Test2</th>
<th>Test3</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEU</td>
<td>Machine check exception - Cache</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>SEU</td>
<td>Machine check exception - Code lost</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>None Timeout</td>
</tr>
<tr>
<td>SEU</td>
<td>Other error messages</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Timeout</td>
</tr>
<tr>
<td>SEU</td>
<td>Abnormal process termination</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>Timeout</td>
</tr>
<tr>
<td>SEU</td>
<td>System hang</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>System crash</td>
</tr>
<tr>
<td>SEU</td>
<td>Automatic system restart</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>System crash</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>20</td>
<td>22</td>
<td>114</td>
<td></td>
</tr>
</tbody>
</table>

Most of the **Machine Check Exceptions (MCEs)** were produced by errors affecting the cache memories. When the condition exception was recoverable by the system, there were no consequences neither in the application nor in the system. However, there was one case in which a system process of the scheduler was affected. A **Machine check exception - Code lost** occurs when the **MCE routine** has lost the raised error code that has provoked the exception; consequently there is no possibility to determine the source of the error.

An **Abnormal process termination** occurs when the monitor detects a timeout in the application, or when the **MCE** logs an exception in kernel code which causes an unreliable system condition. A **System hang** is produced when the system shell do not respond to any command, or when the **MCE** logs a message showing that a rebooting is needed. In the most critical level, the **MCE** logs an **Automatic system restart** message. Finally, there were errors that did not come either from the **MCE** or the monitor application. They were classified as **Other error messages** and have caused, in one case an application timeout, and in the other case a killing of a system process.

To estimate the dynamic cross-section, only faults that led to **unreliable system condition, application timeouts and system crashes** were considered. The total number of events for the three tests was 156, and among them 86 produced errors. The total fluence was \(3.27 \times 10^9 n \cdot \text{cm}^{-2}\). Applying Equations (1) and (3) for a confidence level of 0.95, the dynamic cross section in SMP scenario is:

\[2.10 \times 10^{-8} \text{ cm}^2_{\text{device}} < \sigma_{DYN,SMP} < 3.25 \times 10^{-8} \text{ cm}^2_{\text{device}}\]

The **machine-check-error report** and the **traps** implemented in the OS allowed determining the source of the errors of these faults. Figure 4 illustrates the relationship between the OS faults and the hardware source of the error. It is important to note that the dynamic cross section is strongly dependent on the characteristics of the tested scenario. The obtained results show that around of 70% of the errors affects the system while the other 30% affects the application itself. It can be explained, since this scenario maximizes the use of CPU resources and scheduling, and the TSP implementation has an intrinsic fault-tolerant capability.

### D. Comparison of SEEs consequences in the two dynamic scenarios

The SEEs consequences issued from the two dynamic scenarios are shown in Figure 5. In the AMP-memory-bound 86.67% of the events had no consequences, 6.67% provoked errors in the user application, and 6.67% caused system hangs. On the other hand, in the SMP-CPU-bound scenario 44.87% of the observed events had no consequences on the system or application, 13.46 % of the events provoked timeouts of the user application, 20.51% of the events caused an unreliable condition in system and 21.15% crashed the system. Table VIII summarizes the uncertainty margins of the SEEs consequences for the two dynamic scenarios with a 0.95 confidence level.

A comparison of both dynamic cross sections \(\sigma_{DYN,AMP}\) and \(\sigma_{DYN,SMP}\) shows that the dynamic response of the device depends not only on the application but also on the adopted multi-processing mode. Moreover, the obtained results revealed that errors may occur in SMP mode, even if the OS is in idle mode.

In the literature, there is a work that compares the performance of the SMP and AMP modes both with operating systems for a dual-core giving as a conclusion that SMP...
outperforms the AMP mode [21]. Inferring this affirmation to the present work, it is possible to suggest the existence of a trade-off between reliability and performance according to the multi-processing mode selected.

VI. CONCLUSIONS AND FUTURE WORK

This work has evaluated the sensitivity to 14 MeV neutrons of a 45nm SOI P2041 multi-core processor. From the static test results, it can be seen that 45 nm SOI technology is between 3 and 5 times less sensitive to neutron radiation than its CMOS counterpart.

The dynamic AMP tests have demonstrated that in spite of the parity and ECC protection mechanisms, errors have been occurred in the application results. A deeper analysis has allowed to determine that errors were caused by MBUs in the address tags and data.

Results presented in section V-D suggest that the studied multi-core working in AMP scenario without OS is at least five times less sensitive to SEEs than working in SMP scenario. The main reason is that AMP scenario exploits fewer chip resources than SMP scenario.

In future work, a similar approach will be applied for other multi-core and many-core circuits with different memory architectures.

ACKNOWLEDGMENT

Authors thank to F. J. Franco, our college from UCM, for his valuable collaboration in the analysis of the results.

Table VIII

<table>
<thead>
<tr>
<th>Consequence Type</th>
<th>AMP-MM</th>
<th>SMP-TSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>[62 - 97]</td>
<td>[55 - 88]</td>
</tr>
<tr>
<td>Timeout</td>
<td>[0 - 4]</td>
<td>[13 - 32]</td>
</tr>
<tr>
<td>Unreliable system</td>
<td>N/A</td>
<td>[22 - 45]</td>
</tr>
<tr>
<td>System crash</td>
<td>[2 - 13]</td>
<td>[23 - 46]</td>
</tr>
<tr>
<td>App. result error</td>
<td>[2 - 6]</td>
<td>[0 - 4]</td>
</tr>
<tr>
<td>Total</td>
<td>[72 - 110]</td>
<td>[132 - 182]</td>
</tr>
</tbody>
</table>

REFERENCES


Figure 5. SEE consequences according to the scenario implemented. The confidence intervals are shown by means of the red lines.