Radiation Experiments on a 28 nm Single-Chip Many-Core Processor and SEU Error-Rate Prediction

Vanessa Vargas, Pablo Ramos, Vincent Ray, Camille Jalier, Renaud Stevens, Benoît Dupont De Dinechin, Maud Baylac, Francesca Villa, Solenne Rey, Nacer-Eddine Zergainoh, Jean-Francois Méhaut, and Raoul Velazco

***Abstract*— This work evaluates the SEE static and dynamic sensitivity of a single-chip many-core processor having implemented 16 compute clusters, each one with 16 processing cores. The SEU error-rate of an application implemented in the device is predicted by combining experimental results with those issued from fault injection campaigns applying the CEU (Code Emulating Upsets) approach. In addition, a comparison of the dynamic tests when processing-cores cache memories are enabled and disabled is presented. The experiments were validated through radiation ground testing performed with 14 MeV neutrons on the MPPA-256 many-core processor manufactured in TSMC CMOS 28HP technology. An analysis of the erroneous results in processor GPRs was carried-out in order to explain their possible causes.**

***Index Terms*— Accelerated testing, fault injection, many-core, parallel processing, SEE, SEFI, SEU, soft error.**

I. Introduction

MANY-CORE processors design is becoming a new challenge since manufacturers have to face critical factors such as performance, reliability and power consumption. The exceptional computational capabilities of these devices make them very attractive for the implementation of high-performance applications in scientific and commercial fields. For instance, the *Sunway TaihuLight Supercomputer*, which was ranked in the first position of the TOP500 list on June 2016, is based on the many-core processor ShenWei SW26010 (260 cores) [1]. Aerospace and avionics industries have also special interest in multi-core and many-core processors since they are capable of dealing with the simultaneous requirements of high performance computing and low power consumption [2], [3].

However, the continuous technology shrink combined with the design complexity increase their vulnerability to natural radiation, especially to Single Event Effects (SEEs). A significant advantage of many-core processors to face this vulnerability concern consists in their inherent redundancy capability which makes them ideal for implementing fault tolerant techniques such as N-modular redundancy, which applies majority-voting [4]. In addition, for improving device reliability, complementary protection mechanisms are implemented in memory cells. Error Correcting Codes (ECC) and Parity are commonly used to protect processor's cache memories and device's shared memories. ECC typically allows double error detection and single error correction while parity allows only single error detection. Nevertheless, implementing additional protections involves the introduction of an extra area which leads to more power consumption and performance degradation.

The dependability of many-core processors is a crucial issue to consider, especially if the devices are intended to be used for safety-critical applications [5]. It is thus mandatory to evaluate the SEE sensitivity of such devices.

Radiation ground testing and real-life tests are widely used to evaluate processor-based architectures in terms of SEE cross-section. This quantity characterizes the sensitive zones but is not enough to forecast the behavior of an application implemented on such devices exposed to radiation environments. Ideally, a dynamic test should be performed to evaluate the behavior of an application implemented in processor-based devices. It consists in exposing to radiation the device under test while it executes the selected application. Nevertheless, the high cost of the use of particle accelerator facilities makes infeasible testing several applications. Therefore, it is compulsory to use a method to predict the application error-rate of multi-core and many-core processors.

Manuscript received July 8, 2016; revised October 30, 2016 and December 6, 2016; accepted December 7, 2016. Date of publication December 9, 2016; date of current version February 28, 2017. This work was supported in part by the French authorities through the "Investissements d'Avenir" program (CAPACITES project), by the Universidad de las Fuerzas Armadas ESPE and by the Secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador (SENESCYT) STIC-AmSud-EnergySFE Project PIC-16-ESPE-STIC-001.

V. Vargas and P. Ramos are with the Universidad de las Fuerzas Armadas ESPE, DEEE, Sangolqui, Ecuador and also with Université Grenoble-Alpes & TIMA Labs, Grenoble 38000, France (e-mail: vcvargas@espe.edu.ec; pframos@espe.edu.ec).

V. Ray, C. Jalier, R. Stevens, and B. Dupont De Dinechin are with Societé Kalray, Montbonnot-Saint-Martin 38330, France (e-mail: vincent.ray@kalray.eu; camille.jalier@kalray.eu; renaud.stevens@kalray.eu; benoit.dupont de dinechin@kalray.eu).

M. Baylac, F. Villa, and S. Rey are with the Laboratoire de Physique Subatomique et de Cosmologie LPSC, Université Grenoble-Alpes &CNRS/IN2P3, Grenoble 38000, France (e-mail: maud.baylac@lpsc.in2p3.fr; francesca.villa@lpsc.in2p3.fr; solenne.rey@lpsc.in2p3.fr).

N.-E. Zergainoh and R. Velazco are with TIMA Labs. (Université Grenoble-Alpes & CNRS), Grenoble 38000, France (e-mail: nacer-eddine.zergainoh@imag.fr; raoul.velazco@imag.fr).

J.-F. Méhaut is with LIG Labs., Université Grenoble-Alpes & CNRS, Grenoble 38000, France (e-mail: jean-francois.mehaut@imag.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2016.2638081

The present work aims at evaluating the intrinsic sensitivity to 14 MeV neutrons of a CMOS 28 nm single-chip many-core processor with 256 processing cores. The prediction of the application error-rate combining the static cross-section of the device, obtained from radiation experiments, and the SEU error-rate issued from fault injection campaigns is also established. Results obtained from dynamic tests were compared with those predicted in order to validate the approach.

The remainder of the paper is organized as follows: Section II presents the related work. Section III describes the device under test. Section IV defines the adopted approach to evaluate the intrinsic sensitivity of the device and the prediction of the application error-rate. Section V details the experimental setup. Section VI presents and analyzes the results issued from neutron ground testing. Finally, Section VII concludes the paper and provides some directions for future work.

II. Related Work

In the literature, it is possible to find several works related to the evaluation of the radiation sensitivity of electronic components, such as memories, microprocessors and FPGAs. For instance, there is an interesting work [6] that summarizes the sensitivity to SEEs induced by neutrons of different integrated circuits applicable to avionics. However, there are very few works available regarding multi-core and many-core processors sensitivity.

Reference [7] presents the SEE test results under 15 and 25 MeV ions of the 49-core Maestro ITC microprocessor. Maestro is a radiation hardened by design (RHBD) processor based on the Tilera TILE64 processor intended to be used in space applications. Radiation tests targeted L1 and L2 cache memories as well as registers of the tile core. The main observed SEE mechanism was upsets in the L1 and L2 caches which were handled by an effective EDAC included in the Maestro design.

Authors of [8] establish a dynamic cross-section model for a multi-core server based on quad-core processors in 45 nm bulk CMOS technology. They also provide a fault handling comparison between Windows 5.2 and Linux 5.1 operating systems.

Reference [9] illustrates that a 45nm Silicon-On-Insulator (SOI) Quad-core processor is about four times less sensitive to SEE that its CMOS counterpart. In addition, it can be seen that the dynamic sensitivity of the device strongly depends on the multi-processing mode used.

Reference [10] evaluates the radiation sensitivity 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. A hardening strategy based on Duplication with Comparison is proposed and validated.

In [11], it is proposed to disable the cache memories of high-end processors in safety-critical applications to gain in reliability in spite of the increase of the execution time. This work also presents an analysis of the soft-error effects in data and instruction cache memories.

Reference [12] demonstrates that by enabling L1 cache it is possible to improve the performance of the system without compromising the reliability. They introduce a generic metric (Mean Workload Between Failures) for evaluating the reliability of a embedded processor devoted to execute safety-critical applications. This metric takes into account both cross section and exposure time.

III. Many-Core Processor

The MPPA-256 (Multi Purpose Processing Array) is a many-core processor manufactured in TSMC CMOS 28HP technology. The processor operates between 100 MHz and 600 MHz, for a typical power ranging between 15 W and 25 W. Its peak floating-point performances at 600 MHz are 634 GFLOPS and 316 GFLOPS for single and double-precision respectively. It integrates 256 Processing Engine (PE) cores and 32 Resource Management (RM) cores. The MPPA-256 considered in this work is the second version called Bostan.

A. Development Platform

The MPPA Developer is a development platform containing the MPPA ACCESSCORE SDK version 2.5 for developing, optimizing and evaluating applications. It is based on an Intel core I7 CPU operating at 3.6 GHz and running a Linux OS. The platform includes a PCIe board MPPA-256 Processor and a PCIe board for debug and probe.

The MPPA ACCESSCORE SDK includes three programming models for developing an application: POSIX, Kalray OpenCL and Lowlevel. The Kalray software stack includes several abstraction layers.

Since the objective of this work is to evaluate the chip sensitivity to SEE, the many-core was configured in bareboard minimizing the use of libraries and its impact.

B. Many-Core Processor Architecture

The global processor architecture is clustered with 16 compute clusters (CCs) and 2 input/output (I/O) clusters per device. Each compute cluster is built around a multi-banked local static memory (SMEM) of 2MB shared by the 16 (PE) + 1(RM) cores. Each I/O cluster has two quad-cores and two main banks of 2 MB. Each quad-core is a Symmetric Multi-Processing system.

A wormhole switching network-on-chip (NoC) with 32 nodes and a 2D torus topology connects the compute clusters and the I/O clusters. In Figure 1 it is illustrated an overview of the many-core processor [13].

The cores comprising the MPPA-256 are based on the same VLIW 32-bit/64-bit architecture. The VLIW core implements separate instruction and data cache. There is no hardware cache coherency mechanism between cores, nor between data cache and instruction cache. However, to enforce memory coherency, several software mechanisms are available to programmers.

The main components of the many-core processor are covered by error protection mechanisms except the instruction

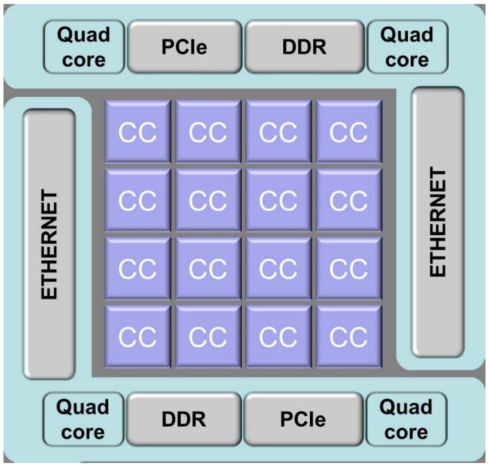

Fig. 1. Many-core processor components.

and data cache memories of the VLIW core that are protected by parity. The SMEM implementation interleaves bits of 8 adjacent 64-bit words which allows localized errors spread as multiple single ECC (SECC) errors. They are detected and corrected on the fly. The NoC router queues (512 of 32-bit flits each) are also protected by ECC. Note that SECC errors are silently corrected while Double ECC (DECC) errors are signaled.

IV. Adopted Approach

In this work, the Code Emulating Upset (CEU) approach [14] is adapted to the MPPA many-core processor with the aim of reproducing, in a non-intrusive way, the effects of SEU faults. This approach combines the static cross-section of the device (σ_{Static}) with the error-rate issued from fault injection campaigns (τ_{Inj}) to estimate the error rate (τ_{SEU}) of an application implemented in a processor as described in (1). This approach was demonstrated to be very efficient to predict the application error-rate of processor-based architectures as shown in [15], [16].

$$\tau_{SEU} = \tau_{Inj} \times \sigma_{Static} \tag{1}$$

On one hand, the error-rate of an application is derived from SEU fault injection campaigns. This quantity is defined as the average number of injected faults needed to provoke an error in the result of the application.

$$\tau_{Inj} = \frac{Number\ of\ Errors}{Number\ of\ Injected\ Faults} \tag{2}$$

On the other hand, the static cross-section of the device (σ_{Static}) is obtained from radiation experiments. This value provides the average number of particles needed to cause a bit-flip in a memory cell, and is defined as follows:

$$\sigma_{Static} = \frac{Number\ of\ Upsets}{Fluence} \tag{3}$$

Lastly, for validating the prediction approach, it is necessary to expose to radiation the many-core processor while running the selected application (Dynamic test).

As it is desired to evaluate the internal computing-cluster resources, the many-core processor is configured in bare-board where each cluster executes independently the same application. There are no shared resources between compute clusters. However, the network-on-chip (NOC) resources are used for intercluster communications when the IO cluster spawns the executable code to the compute clusters, and when the clusters log the results.

The details of each stage of the approach are detailed in the following subsections.

A. Fault Injection

Up to now, CEU approach was done by asserting asynchronous external-interrupt signals where the execution of an interrupt handler in the processor emulates the selected error (SEU) in a randomly chosen memory cell. However, in the case of the MPPA many-core processor, it is possible to benefit of the multiplicity of cores for using one of them as a fault injector while the others run the chosen application. In a previous work, it was presented a similar strategy using inter-processor interrupts to inject faults in a quad-core processor [17].

Before starting the fault-injection campaign, it is necessary to determine the number of cycles required to execute the selected application. This is done in order to know the range of time in which the fault injection should be performed. While the application is running on the processing engine cores, the resource manager core of each compute cluster performs the fault injection. It randomly selects the core (any of the processing engine or itself), the register, the injection instant (in terms of clock cycles), and the bit to be altered. At the injection instant, the RM sends an inter-processor interrupt to the selected core. The latter performs the bit-flip in its register.

This work only considers SEU emulation where one SEU per cluster and per run is injected. The fault-injection procedure is repeated several times in order to obtain enough quantity of samples to calculate the injection error-rate τ_{inj}.

The fault injection campaign is devoted to inject faults in General Purpose Registers (GPRs) and System Function Registers (SFRs) of the compute cluster's cores (PEs or RM). Since some SFRs are non-writable by software means, only 34 SFRs of 50 SFRs are targeted. Among the targeted SFRs, the most critical ones are the 8 registers saved during context switching: shadow program counter (spc), shadow program status (sps), return address (ra), compute status (cs), processing status (ps), loop counter (lc), loop start address (ls) and loop exit address (le). On the contrast, other registers such as processing identification, system reserved and performance monitor are not targeted.

B. Static Sensitivity

To evaluate the intrinsic sensitivity of the many-core processor, it was performed a static test targeting the SMEMs of the compute clusters, since they occupy most of the storage area of the many-core device. To accomplish this task, an initialization program is loaded by means of the JTAG into all the SMEMs and is executed on the RM core of each compute cluster.

TABLE I
MEMORY AREA OF THE MPPA-256 MANY-CORE PROCESSOR

Sensitive zone	*Location*	*Capacity*	*Description*
SMEM	Computing Cluster	2 MB per cluster	Static Shared Memory
SMEM	I/O Cluster	4 MB per cluster	Static Shared Memory
IC-CC	CC VLIW core	8 KB per core	Instruction Cache
DC-CC	CC VLIW core	8 KB per core	Separated Data cache
IC-IO	IO VLIW core	32 KB per core	Instruction Cache
DC-IO	I/O cluster	128 KB per master core	Shared/Separated Data cache
GPR	VLIW Core	64 registers of 32 bits per core	General Purpose Registers
SFR	VLIW Core	50 registers of 32 bits per core	System Function Registers

This program initializes the SMEMs by writing a predefined 64-bits pattern into all the memory locations of the SMEMs except 3.3% of the memory that is devoted to the code itself.

Once finished the initialization, a checking program is periodically loaded by means of the JTAG into all the SMEMs and is executed on the RM core of each compute cluster. This program reads each double-word of the SMEM and compares it with a predefined pattern along the radiation test. In the case the code finds any mismatch or detects a SECC or DECC error, it sends a message to the host via JTAG, specifying the nature of the error and the implicated SMEM address where it occurred. The next step is to write the correct value in the corresponding memory location.

Note that detected SECC errors are auto-corrected in the SMEMs of the clusters and signaled to the processor that have performed the memory access. As a consequence, each time the program finishes its execution, all the SMEMs start with a fresh state that is cleared out of any SECC error. At the end of the experiment, the static cross-section ($\sigma_{Stat_{SMEM}}$) is obtained to estimate the intrinsic sensitivity of a memory cell belonging to the SMEMs built-in CMOS 28nm TSMC HP technology. Table I summarizes the memory area of the many-core processor.

C. Dynamic Response

The dynamic response of the device is evaluated through the execution of a testing application of the MPPA that must accomplish the following characteristics: intensive use of the cluster resources, code and data size maximum of 2 MB, evenly load distribution among PEs, and enough execution time to ensure all PEs running in parallel.

The code is loaded by means of the JTAG in the SMEM of the I/O cluster 0. This cluster then spawns the same executable into the 16 compute clusters and orders them to start the execution of the program. Within each cluster, the RM core wakes-up the 16 PE cores, and each one of them starts the execution of the application.

As in the static test case, SECC and other errors such as data parity (DPAR) and instruction parity (IPAR) are reported with a message. The goal of this test is to evaluate the dynamic behavior of the many-core processor when no operating system is used. To do this, two different scenarios are considered.

In the first scenario, the cache memories of the cores are all enabled and configured in write-through mode. In the second one, the cache memories are all disabled. In both cases, the application cross-section (σ_{DYN}) and the average execution time are obtained.

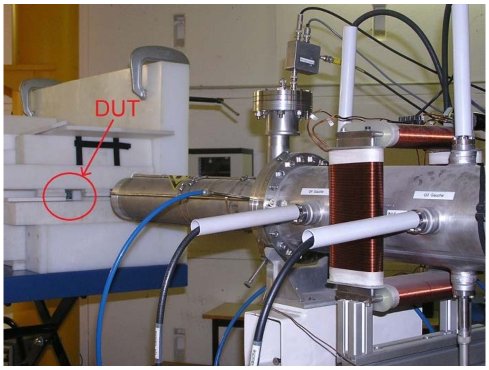

Fig. 2. Experiment at GENEPI2 facility.

Furthermore, for the cache enabled scenario, additional radiation campaigns are conducted varying the device operating frequency and the bias voltage to do a rough but rapid characterization of voltage/frequency conditions in which the circuit is able to operate.

V. 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 [18]. This accelerator was originally developed for nuclear physics experiments, and since 2014 it has been used to irradiate integrated circuits from different technologies.

GENEPI2 is an electrostatic accelerator producing neutrons with an average energy of 14 MeV generating a maximum neutron flux of $4.5 \times 10^7\ n \cdot cm^{-2} \cdot s^{-1}$. For the radiation tests presented in this paper, the flux was intentionally limited to $1.2 \times 10^5\ n \cdot cm^{-2} \cdot s^{-1}$. It is considered, to a first approximation, that only neutrons emitted fully forward will impact the DUT while a dedicated shielding protects the readout electronic platform as depicted in Figure 2.

B. Test Parameters

Four radiation test campaigns were performed on the many-core processor: static test, dynamic test cache enabled, dynamic test cache disabled and dynamic test varying operating parameters. The device under test was decapsulated and placed facing the center of the target perpendicularly to the beam axis at a distance of 15.2 ± 0.5 cm. The DUT fan was placed laterally to cool-down the device, rather than being placed on the device. Consequently, the computing cluster frequency was set to 100 MHz to reduce power consumption. The bias voltage of the device was set to 0.9V. Note that only for the last radiation test campaign the bias voltage and the operating frequency were changed.

The neutron beam energy was 14 MeV with an estimated flux of $1.2 \times 10^5\ n \cdot cm^{-2} \cdot s^{-1}$ at 500 Hz frequency with an error of $\pm\ 0.1 \times 10^5\ n \cdot cm^{-2} \cdot s^{-1}$.

C. Benchmark Application

The selected application is an assembler optimized version of a collaborative 256×256 matrix multiplication, where each PE of a given cluster computes $^1/_{16}$ of the cluster result. This computation is running repeatedly being $n = 256$ as stated in (4).

$$C = \sum_{1}^{n} A \times B \tag{4}$$

A, B and C are single precision floating-point matrices. The size of the matrix was chosen so that data remain in the local SMEM memory. The iteration of the matrix operation is done to guarantee that each cluster computes enough time so that all the clusters work in parallel during a considerable time slice. For a 256 matrix size, it takes around 1M IO cycles to spawn 1 cluster. Since clusters are spawned one after another, cluster 15 starts execution around 15M IO cycles after cluster 0.

Each compute cluster performs in Asymmetric Multi-Processing mode (AMP)and the computational work is distributed evenly among the processing cores. The synchronization of the computation is done by events between the RM and the PE cores. The RM wakes up the 16 PEs and sends a notification to each one to start the computation. Then, it waits for a notification from each PE indicating the work was done.

Once all PEs computations have finished, the RM core compares the result matrix with a static expected result-matrix E, and reports any mismatch including the associated addresses and values. Then, the matrix C is filled up with zeros and the PEs start again the computation. The program executes continuously the same algorithm in each cluster along the radiation test.

Since there are no hardware memory coherency in the compute cluster, each PE ensures memory coherency by software means, by updating shared data before (read coherency) and after the computation (write coherency). In addition, the RM calls the memory coherency functions when using the shared data.

VI. Experimental Results

A. Fault-Injection Results

The experiment considers the injection of one SEU per execution in the GPRs or SFRs of the (PE or RM) cores belonging to each cluster as described in section IV *A*. The consequences of a bit-flip in a memory cell were classified as: *silent faults*, *erroneous results*, *timeouts* and *exceptions*.

TABLE II

Results of the Fault Injection Campaigns

Zone	*Silent faults*	*Erroneous results*	*Timeouts*	*Exceptions*
GPRs	36472	16387	6678	1996
SFRs	22745	2034	6365	1639
TOTAL	59217	18421	13043	3635

In these experiments, the SEU faults were injected at a random instant within the nominal duration of the executed program which was around 5.3×10^8 clock cycles.

Table II shows a general overview of the fault injection campaign where 94316 faults were injected in the GPRs and accessible SFRs.

From these results, it was calculated the injection error-rate of the registers applying (2), and considering as errors the erroneous results, timeouts and exceptions.

$$\tau_{Inj} = \frac{Nb\ of\ Errors}{Nb\ of\ Fault\ Inj} = \frac{35099}{94316} = 37.21 \times 10^{-2}$$

This result shows that 37.21% of the injected SEUs in the accessible registers cause errors in the application. Since registers have no protection mechanisms, this campaign is very useful to emulate the behavior of the application in presence of SEUs.

B. Static Test

The first radiation test campaign was devoted to obtain the static cross-section ($\sigma_{Stat_{SMEM}}$) of the SMEMs belonging to the compute clusters. In this campaign the cache memories of the device were disabled and the exposure time was two hours providing a fluence of around $8.64 \times 10^8\ n \cdot cm^{-2} \cdot s^{-1}$. During the test, 2720 Single-bit ECC (SECC) events were detected with no consequences to the self-test application since clusters' SMEMs implement ECC. In addition, one SEFI that caused a hang was observed.

Analyzing the SECCs addresses and considering the tri-dimensional structure of the device, it was possible to identify several Multiple Cell Upsets (MCU). This analysis allows determining that 2309 neutron particles perturbed the SMEMs of the many-core processor. Figure 3 shows the distribution of the observed bit-flips in the shared memories of the 16 clusters. Table III summarizes the results of the static test radiation campaign. Note that the subscript number following the *MCU* represents the multiplicity of the upset.

The results from the static test allow estimating the static cross-section of a CMOS 28nm memory cell as follows:

$$\sigma_{Stat_{SMEM}} = \frac{2721}{8.64 \times 10^8} = 3.15 \times 10^{-6} \frac{cm^2}{device}$$

Since the tested memory area of the many-core processor represents 2.60×10^8 bits, the static cross-section per bit of the SMEMs is about $1.21 \times 10^{-14} \frac{cm^2}{bit}$.

Assuming that the technology of the memory cells is similar for the different memory areas of the device, the cross-section of the GPRs and SFRs can be extrapolated from

TABLE III

RESULTS OF THE STATIC RADIATION TEST

Detected Error	*SEE Type*	*Occurrences*	*Bit-flip Cells*
SECC	SEU	1949	1949
SECC	$\text{MCU}_{(2)}$	322	644
SECC	$\text{MCU}_{(3)}$	24	72
SECC	$\text{MCU}_{(4)}$	8	32
SECC	$\text{MCU}_{(5)}$	2	10
SECC	$\text{MCU}_{(6)}$	1	6
SECC	$\text{MCU}_{(7)}$	1	7
N/A	SEFI	1	1
Total		2309	2721

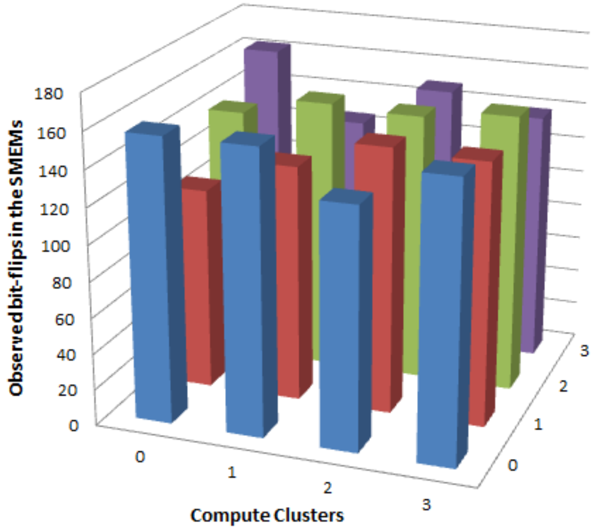

Fig. 3. Distribution of the observed bit-flips in the SMEMs of the clusters.

the cross-section per bit. Taking into account that there are [64 (GPRs) + 50 (SFRs)] ×32 (bits) × [256 (PE) +32 (RM)], the registers' cross-section for the device can be expressed as:

$$\sigma_{Stat_{Reg}} = 1050624\ bit \times 1.21 \times 10^{-14} \frac{cm^2}{bit}$$
$$= 12.71 \times 10^{-9} \frac{cm^2}{device}$$

C. Dynamic Test With Cache Memories Enabled

The second radiation test campaign was carried out to obtain the dynamic cross-section ($\sigma_{Dyn_{ce}}$) of an application running in the many-core processor. For this test, the instruction and data cache memories were enabled in the compute cluster' cores. The exposure time was one hour providing a fluence of about $4.32 \times 10^8\ n \cdot cm^{-2}$. Table IV summarizes the results of the dynamic radiation campaign.

From table IV, it is possible to identify five different types of errors. SECC and Instruction and Data Cache Parity errors were corrected by the ECC and parity protections. On the contrary, Register Trap and Memory-Comparison Failed errors are non-correctable errors since processor registers do not implement protection mechanisms. The Memory Comparison Failed error was detected by the RM core when it identifies differences between the results of the application and the expected values. Table V shows an example of an erroneous result produced during the radiation test.

TABLE IV

RESULTS OF THE DYNAMIC RADIATION TEST WITH CACHE ENABLED

Detected Error	*SEE Type*	*Occurrences*	*Consequences*
SECC	SEU	676	None
Data Cache Parity	SEU	36	None
Inst. Cache Parity	SEU	6	None
Register Trap	SEFI	1	Hang
Memory Comp. Failed	SEU	2	Erroneous Result
Total		721	

TABLE V

ERRONEOUS RESULT OBSERVED DURING DYNAMIC TEST

Cluster	*Address*	*Read Value*	*Expected Value*
14	0xed168	0x4747b37c422f1751	0x44a4f298422f1751
14	0xed568	0xc580e9b8451cd5cc	0xc552f801451cd5cc
14	0xed968	0xc72862ee452874c2	0xc5583ffd452874c2
14	0xedd68	0x46b51afd452bc9da	0x453c182a452bc9da

In table V, it can be seen that four float values belonging to cluster 14 were miscalculated. The 1KB distance between them and the analysis of the assembly code show that this erroneous result was caused by a corruption of the GPR 41 when used in the main loop of the matrix multiplication. This register stores a floating point value coming from the matrix B that will be multiplied and accumulated with 4 float values coming from matrix A, during the current iteration to produce the points of the result matrix C located in $C[i][j]$, $C[i+1][j]$, $C[i+2][j]$ and $C[i+3][j]$. The 1KB distance between corrupted values can be explained due to the fact that a row of A, B or C matrix contains 256 float values (4 bytes each one). Note that the same type of error occurred during the fault injection campaigns in GPRs.

To determine the dynamic cross-section, only the 3 non correctable errors presented in table IV (1 Register trap and 2 Memory Comp. Failed) were taken into account. Due to the scarcity of experimental data, it is compulsory to add uncertainty margins to these results. The more accurate 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:

$$\frac{1}{2}\chi^2\left(\frac{\alpha}{2}, 2N_{err}\right) < \mu < \frac{1}{2}\chi^2\left(1-\frac{\alpha}{2}, 2(N_{err}+1)\right) \quad (5)$$

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

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

$$1.43 \times 10^{-9} \frac{cm^2}{device} < \sigma_{Dyn_{ce}} < 20.29 \times 10^{-9} \frac{cm^2}{device}$$

TABLE VI
RESULTS OF THE DYNAMIC RADIATION CAMPAIGN CACHE DISABLED

Detected Error	*SEE Type*	*Occurrences*	*Consequences*
SECC	SEU	602	None
Register Trap	SEFI	1	Hang
Memory Comp. Failed	SEU	1	Erroneous Result
Total		604	

Regarding the performance of the application when both instruction and data cache memories are enabled, 679 computations of the matrix multiplication were completed per cluster in one hour. The average computation time was ~5.30 seconds at 100 MHz frequency. There were no cache misses in the instruction caches, since the code is quite small and occupies around 400 bytes that easily fit in the PEs' instruction caches. Concerning data caches, the miss rate was roughly 2.5%. Even though the 256 PEs are working all the time fully loaded, 30% of the execution time is spent waiting for missing data to arrive from the SMEM.

D. Dynamic Test With Cache Memories Disabled

The third radiation test campaign was carried out to obtain the dynamic cross-section ($\sigma_{Dyn_{cd}}$) with cache memories disabled. The test parameters and exposure time were the same as those of the previous case. Table VI summarizes the results of this dynamic radiation campaign.

SECC errors produced in the SMEMs were corrected by the ECC while Register Trap and Memory Comp. Failed errors remain uncorrected. Only uncorrected errors were taken into account to determine the dynamic cross-section. Applying a similar analysis as in the previous case, the lower and upper limits for the dynamic cross-section are:

$$0.56 \times 10^{-9}\frac{cm^2}{device} < \sigma_{Dyn_{ce}} < 16.72 \times 10^{-9}\frac{cm^2}{device}$$

Regarding the performance of the application when data cache memories are disabled, 348 computations of the matrix multiplication were completed per cluster in one hour. The average computation time was ~ 10.34 seconds at 100 MHz frequency.

Comparing the results of the two dynamic test campaigns, it can be seen that the matrix multiplication algorithm performs twice as fast as when cache memories were enabled without reliability penalty, since the detected errors were corrected by the parity protection. Consequently, for this many-core processor it is convenient to enable caches memories even for critical applications.

E. Dynamic Test Varying Operating Parameters

A fourth radiation test campaign was performed in order to observe the dynamic response of the many-core processor with cache memories enabled when varying the device operating frequency and bias voltage. For each case, the fluence was about $7.2 \times 10^7\ n \cdot cm^{-2}$. Table VII summarizes the results when varying the operating frequency with a constant bias voltage of 0.9. Table VIII sums up the results when varying the bias voltage with a constant 100 MHz frequency.

TABLE VII
RESULTS VARYING THE DEVICE OPERATING FREQUENCY

Detected Error	*100 MHz*	*200MHz*	*300MHz*
SECC	115	93	115
Data Cache Parity	6	3	7
Inst. Cache Parity	1	0	1
Register Trap	0	0	0
Memory Comp. Failed	0	0	1
Total	122	96	124

TABLE VIII
RESULTS VARYING THE DEVICE BIAS VOLTAGE

Detected Error	*0.8 V*	*0.9 V*	*1.0 V*
SECC	120	115	124
Data Cache Parity	7	6	2
Inst. Cache Parity	3	1	2
Register Trap	0	0	0
Memory Comp. Failed	1	0	1
Total	131	122	129

The fact that there are no significant differences between the results presented in each column of tables VII and VIII, suggests that the device sensitivity is not voltage and frequency dependent for the presented ranges. The slight difference between the totals can be explained by the random distribution of the impinging neutron particles over the chip during the tests. However, note that at 0.9V there are less detected errors maybe due to the fact that this is the nominal bias voltage of the device. As in the previous cases, SECC, Data Cache Parity and Instruction Cache Parity errors were corrected. Only Memory Comp. Failed errors remain uncorrected.

F. SEU Error-Rate Prediction

As described in section IV, the predicted application error-rate is calculated combining the results of fault injection campaigns in GPR and SFR registers, with the static cross-section of registers as follows:

$$\begin{aligned}\tau_{SEU_{REG}} &= \tau_{Inj} \times \sigma_{Static_{REG}}\\ \tau_{SEU_{REG}} &= 37.21 \times 10^{-2} \times 12.71 \times 10^{-9}\\ &= 4.73 \times 10^{-9}\frac{cm^2}{device}\end{aligned}$$

In order to validate the prediction approach for the many-core processor, the predicted value is compared with the measured one. In this case, the dynamic cross-section is calculated gathering the results from both dynamic tests cache enabled and disabled. 3 SEU and 2 SEFI in general purpose registers that caused erroneous results or hangs were added. The average fluence during the two hours of test was of $8.64 \times 10^8\ n \cdot cm^{-2}$. Then, applying (3) the dynamic cross-section is:

$$\sigma_{Dyn} = \frac{5}{8.64 \times 10^8} = 5.78 \times 10^{-9}\frac{cm^2}{device}$$

Due to the scarcity of experimental data, the 95% confidence interval of the result was calculated, being the cross-section of the application $\sigma_{Dyn} = [1.87 - 13.50] \times 10^{-9} cm^2/device$.

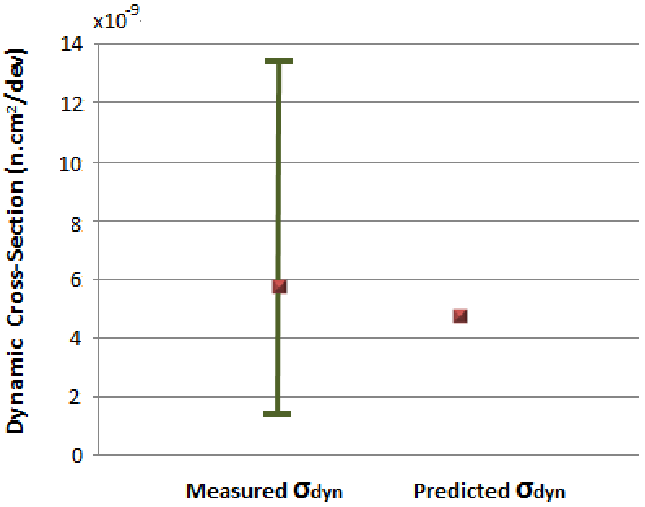

Fig. 4. Confidence intervals of the measured dynamic cross-section vs the predicted dynamic cross-section.

Figure 4 depicts the confidence intervals of the measured dynamic cross-section compared with the predicted one.

Comparing the predicted and the measured error rates, it can be seen that the predicted value is quite close to the experimental one within the confidence interval. The underestimation of the predicted value can be explained since 16 SFRs, corresponding to the 14% of the sensitive area of the many-core, are not accessible by software means and thus, they were not targeted in the fault injection campaigns.

VII. Conclusions and Future Work

This paper presents the 14 MeV neutron sensitivity of the MPPA-256 many-core processor. The cross-section per bit of the 28nm TSMC CMOS technology obtained from the static test is about $1.21 \times 10^{-14} \frac{cm^2}{bit}$. Results suggest that ECC and interleaving implemented in the SMEMs of the clusters are very effective to mitigate SEUs as all detected events of this type were corrected.

The evaluation of the device dynamic response shows that by enabling the cache memories, it is possible to gain in performance of the application without compromising reliability, as all the detected errors produced in data and instruction cache memories were corrected by the parity protection. The non-correctable errors that occurred in the different dynamic tests were produced by bit-flips in GPRs since registers do not implement any protection mechanism.

Results presented in section VI-F show that the predicted application error-rate is reasonably close to the measured one. Consequently, despite the complexity of the many-core processor, this work support the relevance of the use of the CEU approach to predict the error-rate of applications implemented in such devices.

In future work, the sensitivity of the MPPA-256 will be evaluated when executing applications using inter-cluster communication, as well as running an operating system. Furthermore, the CEU approach will be applied to other multi-core and many-core processors executing different applications.

Acknowledgment

The authors thank Benjamin Minguez from the Societé Kalray for his contribution to the radiation experiments presented in this work.

References

[1] J. Dongarra. (Jun. 2016). *Report on the Sunway Taihulight System*. [Online]. Available: http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf

[2] Certification Authorities Software Team (CAST). (May 2014). *Position Paper Cast-32 Multi-Core Processors*. [Online]. Available: https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-32.pdf

[3] C. Villalpando, D. Rennels, R. Some, and M. Cabanas-Holmen, "Reliable multicore processors for NASA space missions," in *Proc. IEEE Aerosp. Conf.*, Mar. 2011, pp. 1–12, doi: 10.1109/AERO.2011.5747447.

[4] S. Mukherjee, *Architecture Design for Soft Errors*. San Francisco, CA, USA: Morgan Kaufmann, 2008.

[5] G. H. Asadi, S. Vilas, M. B. Tahoori, and D. Kaeli, "Balancing performance and reliability in the memory hierarchy," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Mar. 2005, pp. 269–279.

[6] E. Normand and L. Dominik, "Cross comparison guide for results of neutron see testing of microelectronics applicable to avionics," in *Proc. Radiat. Effects Data Workshop*, Jul. 2010, pp. 50–57.

[7] S. Guertin. (Jul. 2012). Initial SEE test of maestro. National Aeronautics and Space Administration, Jet Propulsion Laboratory, Pasadena, CA, USA. [Online]. Available: http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/42682/1/12-2145_A1b.pdf

[8] S. S. Stolt and E. Normand, "A multicore server SEE cross section model," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 6, pp. 2803–2810, Dec. 2012.

[9] P. Ramos *et al.*, "Evaluating the SEE sensitivity of a 45 nm SOI multicore processor due to 14 MeV neutrons," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 4, pp. 2193–2200, Aug. 2016.

[10] D. A. G. Oliveira *et al.*, "Modern GPUs radiation sensitivity evaluation and mitigation through duplication with comparison," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 3115–3122, Dec. 2014.

[11] M. Rebaudengo, M. Reorda, and M. Violante, "An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor," in *Proc. Conf. Design, Autom. Test Eur.*, 2003, pp. 602–607.

[12] T. Santini, P. Rech, G. Nazar, L. Carro, and F. R. Wagner, "Reducing embedded software radiation-induced failures through cache memories," in *Proc. Eur. Test Symp.*, May 2014, pp. 1–6, doi: 10.1109/ETS.2014.6847793.

[13] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. de Dinechin, "The shift to multicores in real-time and safety-critical systems," in *Proc. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Oct. 2015, pp. 220–229.

[14] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through C.E.U. (code emulating upsets) injection," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 6, pp. 2405–2411, Dec. 2000.

[15] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodriguez, and J. R. Mingo, "Estimating error rates in processor-based architectures," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 5, pp. 1680–1687, Oct. 2001.

[16] R. Velazco, G. Foucard, and P. Peronnard, "Combining results of accelerated radiation tests and fault injections to predict the error rate of an application implemented in SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 6, pp. 3500–3505, Dec. 2010.

[17] V. Vargas, P. Ramos, W. Mansour, R. Velazco, N. Zergainoh, and J. Mehaut, "Preliminary results of SEU fault-injection on multicore processors in AMP mode," in *Proc. IEEE 20th Int. On-Line Test. Symp. (IOLTS)*, Sep. 2014, pp. 194–197.

[18] F. Villa *et al.*, "Accelerator-based neutron irradiation of integrated circuits at GENEPI2 (France)," in *Proc. Radiat. Effects Data Workshop*, Jul. 2014, pp. 1–5, doi: 10.1109/REDW.2014.7004511.

[19] J. Autran, P. Munteanu, P. Roche, and G. Gasiot, "Real-time soft-error rate measurements: A review," *Microelectron. Rel.*, vol. 54, pp. 1455–1476, Feb. 2014.

[20] R. Velazco *et al.*, "Evidence of the robustness of a COTS soft-error free SRAM to neutron radiation," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 6, pp. 3103–3108, Dec. 2014.