Evaluation of the SEE sensitivity and methodology for error rate prediction of applications implemented in Multi-core and Many-core processors

Pablo Francisco Ramos Vargas

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **NANO ELECTRONIQUE ET NANO TECHNOLOGIES**

Arrêté ministériel : 25 mai 2016

Présentée par

PABLO FRANCISCO RAMOS VARGAS

Thèse dirigée par **Raoul VELAZCO, Directeur de recherche, Laboratoire TIMA**, et
codirigée par **Nacer-Eddine ZERGAINOH, Maître de conférence, Université Grenoble Alpes**

préparée au sein du **Laboratoire Techniques de l'informatique et de la Microélectronique pour l'Architecture des Systèmes intégrées**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Evaluation de la sensibilité face aux SEE et méthodologie pour la prédiction de taux d'erreurs d'applications implémentées dans des processeurs Multi-cœur et Many-cœur

Thèse soutenue publiquement le **18 avril 2017**,
devant le jury composé de :

Monsieur Alain SYLVESTRE
Professeur, Université Grenoble Alpes, Président
Madame Lirida NAVINER
Professeur, Telecom ParisTech, Rapporteur
Monsieur Pascal BENOIT
Professeur, Université Montpellier 2, Rapporteur
Monsieur Raoul VELAZCO
Directeur de recherche, CNRS Délégation Alpes, Directeur de thèse

Dedication

To my beloved wife and sons…

To my parents…

“Always forward without looking back”

-Popular saying-

Acknowledgments

First and foremost, I would like to thank **God**, the almighty, for the miracle of the life, for my family and for providing me this wonderful opportunity of doing this Ph.D study and being part of a different culture.

I gratefully thank the Ecuadorian government through the Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) and the Universidad de las Fuerzas Armadas ESPE for the funding received towards this thesis.

I would like to thank Mrs. Dominique Borrione, Director of Laboratoty at the time, for welcoming me to TIMA.

I want to express my deep thanks and gratitude to my academic advisor, Dr. Raoul Velazco for his guidance and assistance throughout this research, for his motivation, optimism and of course for his good sense of humor.

I would like to express my thankfulness and appreciation to my co-advisor, Dr. Nacer-Eddine Zergainoh for his support and advices during the thesis.

I would like to thank Professor Lirida Naviner and Professor Pascal Benoit, for accepting to be my thesis examiners and for their helpful comments and suggestions.

I acknowledge all the staff of TIMA laboratory for their hospitality, gentle cooperation and availability.

My sincere thanks to Maud Baylac, Francesca Villa and Solenne Rey from the LPSC, for providing me the access to Genepi2 facility to perform the radiation experiments.

I would like to thank Stephan Gailhard, Vincent Ray, Camille Jalier and Renaud Stevens from Kalray Company for their support in the radiation experiments on the MPPA-256 many-core and for providing the required information about the device.

I would like to thank all the friends that I have found during my thesis in Grenoble, especially to Robert, Audrey, Martín, Jessica, Alejandro, Anaïs and Adrien.

Special thanks to my mother and my mother-in-law, for all of the sacrifices that they have made on my behalf and for their love and dedication to taking care of my children.

I want to thank my grand-mother Nina for her prays, my grand-parent Panchito for his encouragement, my grand-mother Olguita for her love and my grand-parent Telmo for showing me that one must fight till the end.

I want to thank my beloved wife Vanessa for loving me as I am, for these ten happy years of marriage, and for her support during this three and half years of thesis.

I want to thank my sons, Matías and Emmanuel for being my inspiration, for their love, tenderness, and for their smiles that decorated my life.

I gratefully thank my brother and my sister for being my childhood companions and for their constant support along the way.

Lastly, but not less important, I want to thank my parents for their love and unfailing support throughout my life, for always trusting in me and for encouraging me to continue studying.

Pablo Ramos

Contents

List of figures

List of Tables

List of Abbreviations

AMP	Asymmetric Multi-Processing
ATX	Advanced Technology eXtended
AVF	Architectural Vulnerability Factor
BSP	Board Support Package
CAPACITES	Calcul Parallèle pour Applications Critiques en Temps et Sureté
CAST	Certification Authorities Software Team
CC	Compute Cluster
CERN	Conseil Européen pour la Recherche Nucléaire
CEU	Code Emulating Upset
CMOS	Complementary Metal-Oxide Semiconductor
CMP	Chip Multi-Processor
CNES	Centre National d'Etudes Spatiales
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CR	Condition Register
CUDA	Compute Unified Device Architecture
DDR	Double Data Rate
DECC	Double ECC
DPAA	Data Path Accelerator Architecture
DUT	Device Under Test
DWC	Duplication with comparison
ECC	Error Correcting Code
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable Programmable Read Only Memory
ESRF	European Synchrotron Radiation Facility
FC-BGA	Flip-Chip Ball Grid Array
FD-SOI	Full Depleted Silicon On Insulator
FET	Field Effect Transistor
FIT	Failure in Time
FPGA	Field Programmable Gate Array

FPR	Floating Point Register
FT	Fault Tolerant
FTM	Fault Tolerant Mechanism
GCR	Galactic Cosmic Ray
GENEPI2	Generator of Neutron Pulsed and Intense
GPR	General Purpose Register
GPU	Graphic Processor Unit
HDL	Hardware Description Language
HPC	High Performance Computing
HWIFI	Hardware Based Fault Injection
ITC	Interim Test Chip
LANL	Los Alamos National Laboratory
LET	Linear Energy Transfer
LPSRAM	Low Power SRAM
LR	Link Register
MBU	Multiple Bit Upset
MCU	Multiple Cell Upset
MM	Matrix Multiplication
MPPA	Multi Purpose Processing Array
MTBF	Mean Time Between Failure
NASA	National Agency
NIEL	Non Ionizing Energy Loss
NoC	Network-On-Chip
OS	Operating System
PC	Program Counter
PE	Processing Engine
RBD	Reference Design Board
RHBD	Radiation Hardened By Design
RM	Resource Manager
RTOS	Real Time Operating System
SAA	South Atlantic Anomaly
SDK	Software Development Kit
SEB	Single Event Burn-Out
SECC	Single ECC
SEE	Single Event Effect
SEFI	Single Event Failure Interrupt
SEGR	Single Event Gate Rupture

LIST OF ABBREVIATIONS

SEL	Single Event Latch-up
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SFR	System Function Register
SMEM	Static Memory
SMP	Symmetric Multi-Processing
SOI	Silicon-On-Insulator
SP	Stack Pointer
SPR	Special Purpose Register
SRAM	Static Random Access Memory
SRR0	Save/Restore Register 0
SRR1	Save/Restore Register 1
SWIFI	Software Based Fault Injection
TMR	Triple Modular Redundancy
TSMC	Taiwan Semiconductor Manufacturing Company
U-boot	Universal bootloader
VLIW	Very Long Instruction Word
VR	Vector Register
XER	Exception register

Chapter 1 : Introduction

1.1 General context and Motivation

Multi-core and many-core processors are a promising solution for achieving reliability, high performance and low-power consumption. The use of these devices is becoming very attractive due to their huge processing capacity combined with their intrinsic redundancy capability, which make them ideal for the implementation of high-performance applications in scientific, safety-critical and commercial domains.

There are several international projects that work to validate the use of multi and many-core processors in critical-embedded system. In fact, spacecraft and avionic industries are interested in validating their usage for incorporating the functions of a whole system into a single chip, and for increasing the dependability of their applications (Villalpando, 2011), (CAST, 2016). For instance, the new trend in avionic systems architecture is to rely on IMA (Integrated Modular Avionic) instead of the classical federated architecture. The main difference between both architectures is that in the federated one, each system has private resources while in IMA the resources can be shared. One of the most important challenges for IMA architectures is the integration of Commercial-off-the-shelf (COTS) multi-core processors (Bieber, 2012). The use of COTS multi-core processors is convenient due to budget and availability issues. Nevertheless, the selection of these components creates complications for realistic cost, effort estimation, and certification against errors (Ye, 2004).

In fact, one of the main concerns of certification for critical-embedded systems is the radiation sensitivity of electronic components. This radiation may result in transient and permanent failures, called Single Event Effects (SEE). A representative form of SEE is the Single Event Upset (SEU), which deposited energy causes a single bit-flip. SEUs are critical since they may lead to the modification, randomly in time and location, of the content of a memory cell with unexpected consequences at the application level.

The occurrence of SEE mainly depends on the device technology and the radiation environment where the circuit is intended to operate. Indeed, having more dense and complex devices implies technology scaling which increase the radiation sensitivity. For this reason, manufacturers are continuously searching for new methods to improve technology process in order to reduce SEE consequences. Silicon-On-Insulator (SOI) is a clear example of these

improvements made to face traditional bulk CMOS drawbacks. Radiation Hardening by Design (RHBD) techniques are also used to mitigate SEU effects. For instance, the implementation of Error Correcting Codes (ECC) and parity to protect the internal memory of the processors is convenient but not enough in presence of Multiple Bit Upsets (MBUs). Another well-known RHBD technique is the Triple Modular Redundancy (TMR) which significantly improves the reliability of the system. However, implementing additional hardware components and protection mechanisms involves the introduction of an extra area and application overhead which leads to more power consumption and performance degradation. It is thus essential to add fault tolerance to the system with the minimal overhead.

In this context, multi-core and many-core processors are very suitable for implementing fault tolerant techniques based on their intrinsic redundancy capabilities. However, it is important to consider the following constraints. First, the high degree of miniaturization of these devices makes them more sensitive to radiation. Second, the huge number of memory cells included in the device increments its vulnerability to radiation. Lastly, the complexity of the architecture in terms of multiplicity of cores, inter-core communications, and memory and I/O management, affect the system reliability. Therefore, the evaluation of the impact of the radiation on the reliability of these devices is mandatory to validate their use in harsh radiation environments.

In order to evaluate the device reliability, its failure rate is required. The failure rate is obtained extrapolating the cross-section issued from static radiation tests to the desired radiation environment. Furthermore, since the failure rate of a system based on processors is application dependent, also dynamic radiation tests are needed to evaluate the system executing the target application. This dependency implies that any change in the application requires new tests. However, the cost of the radiation tests and the availability of the radiation facilities make them unfeasible. Consequently, it is mandatory to use an error-rate prediction approach to face these limitations.

There are few researches in the literature that study the radiation effects on multi/many-core processors. Among them, there are representative studies such as: reference (Stolt, 2012) establishes a dynamic cross-section model for a multi-core server based on a quad-core processor in 45nm CMOS technology. (Guertin, 2012) presents the SEE test results under 15 and 25 MeV ions of the 49-core Maestro ITC, which is a Radiation Hardened By Design processor. In (Oliveira, 2014), it is evaluated the radiation sensitivity of a Graphic Processing Unit (GPU) designed in 28nm technology. Even though, the mentioned works present important results, they aim at validating specific devices and do not provide a general approach to be applied to any multi/many-core processor.

1.2 Scientific context of the thesis

The present thesis has been developed in the context of CAPACITES project at Robust Integrated Systems (RIS) team of the TIMA (Techniques de l'Informatique et de la Microeléctronique pour l'Architecture des systèmes integrés) Laboratoire, and was supported in part by the French authorities through the "Investissement d'Avenir" program (CAPACITES project), by the Secretaría de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador (SENESCYT) grant *753-2012*, and by the Universidad de las Fuerzas Armadas ESPE grant *14-006-BP-DOC-ESPE-a2*.

The project "Calcul Parallèle pour Applications Critiques en Temps et Sureté" (CAPACITES) involves academic and industrial partners with the aim of developing a hardware and software platform based on the KALRAY MPPA many-core processor to accomplish the requirements of critical parallel applications in terms of time response and reliability. The application domains considered in this project are representative of the critical embedded systems implementing significant computing power whose deployment is currently limited or impossible due to the technological choices that have been made on traditional multi-core platforms. The application areas include space and avionics. Furthermore, among the intended project results one can cite: (1) capability to certify end-to-end applications of semantic layers by following the main precepts of the avionics standard DO178C, and (2) Audit of the capacity of the layers and execution models to meet the requirements of certification of the aeronautics, from MPPA multi-core processor.

1.3 Objectives and contributions

This thesis has two main objectives: the first one is to evaluate the SEE sensitivity of applications implemented in multi-core and many-core processors; the second one is to predict the error-rate of applications implemented in multi-core and many-core processors applying the principles of the CEU approach. Due to the complexity of the device's architecture, this work follows the hypothesis that the main contribution to the failure rate is provided by components not implementing protection mechanisms.

The Code Emulating Upset (CEU) approach (Velazco, 2000), was developed at TIMA Laboratory in past thesis for predicting error-rates in processor-based architectures by combining fault-injection campaigns with radiation experiments. The approach proposed in the present work is suitable for injecting faults in multi and many-core processors considering the complexity of their architecture and the implementation of several levels of cache memories as well as internal shared memories. In addition, this new proposal includes memory and exposure-time derating-factors. During this thesis, quantitative theory is applied to two

experiments which are combined to predict the soft error-rate. The first one is to perform radiation experiments with 14 *Mev* neutrons in particle accelerators to emulate a harsh radiation environment. The second one is to perform fault injection in order to simulate the consequences of SEUs in the program execution.

To validate the generality of this approach, different COTS devices were selected aiming at representing the most relevant technological and architectural aspects of multi and many-core processors. The first one was the Freescale P2041 processor manufactured in 45nm SOI technology which integrates four em500c processor cores. The second one was the Adapteva Epiphany E16G301 microprocessor manufactured in 65nm CMOS process which integrates 16 processor cores. The third one was the Kalray MPPA-256 many-core processor manufactured in 28nm TSMC CMOS technology which integrates 16 compute clusters, each one with 17 VLIW core processors. The effectiveness of the approach will be determined by comparing the predicted error-rate with the dynamic response obtained from dynamic radiation tests. This comparison allows validating the relevance of the approach for predicting the error rate of applications implemented in multi/many-core processors.

The main contributions to the state of art of this research are: (1) the proposal of a generic approach for determining the worst-case sensitivity of a multi-core processor implementing ECC and parity in their caches or shared memories that cannot be deactivated, (2) the evaluation of the dynamic response of the Freescale P2041 multi-core by implementing a memory-bound application, (3) the identification of the cache address tag as the source of erroneous results in the radiation test, (4) the worst-case sensitivity of the MPPA-256 many-core processor, (5) an evaluation of the dynamic response of the MPPA-256 demonstrating that by enabling the cache memories it is possible to gain in performance of the application without compromising the reliability of the device, (6) an approach for predicting the application error-rate of multi-core and many-core processors based on CEU principles, (7) the worst-case sensitivity of the Epiphany E16G301 multi-core processor, (8) the comparison of the reliability of the studied processors taking into account technological and architectural characteristics.

The first three contributions results were published in the IEEE Transactions on Nuclear Science (TNS) journal (Ramos, 2016). The second three contributions were published in the IEEE TNS journal (Vargas, 2017).

In the specific context of the CAPACITES project, this thesis contributes with the task related to the study of the reliability of the MPPA-256 many-core processor in a harsh radiation environment

1.4 Thesis outline

Regarding the contents of this thesis, the second chapter presents the background of the research describing the spatial and atmospheric radiation environments in order to introduce the effects of natural radiation on electronic circuits and systems. Several aspects of the characterization of integrated circuit to radiation are addressed. At the end of the chapter, the main issues related to the multi-core and many-core processors are described.

Chapter three summarizes the state of art of the topics covered in the present thesis: (1) SEE sensitivity of multi and many-core processors, (2) Software-based fault injection techniques, and (3) error-rate prediction of processor-based architectures. At the end of the chapter, it is discussed the selection of the CEU approach as a base for predicting the application error rate of multi-core processors.

Chapter four defines the methodology proposed by this research. It details the approach for evaluating the SEE sensitivity and predicting the error rate of applications implemented in multi and many-core processors, as well as the tools needed to carry-out radiation experiments.

Chapter five explains the selection of the platforms used for validating the approach and provides a description of the main characteristics of them. Two multi-core processors and one many-core processor are proposed. In addition, the programming model and the benchmark application are explained.

Chapter six presents the experimental results of the evaluation of the approach in the target devices. Fault injection campaigns are performed to obtain the sensitivity to SEE of the application. Static radiation test with 14 *Mev* neutrons provide the intrinsic sensitivity of the multi-core or many-core processors. Dynamic tests are performed to obtain the dynamic response of the device and for validating the prediction approach. At the end of the chapter, it is provided a comparison between the studied devices in terms of reliability.

Finally, chapter seven summarizes the research and provides general conclusions and some future perspectives.

Chapter 2 : Background

This chapter presents the radiation effects on electronic circuits by providing an overview of the spatial and atmospheric radiation environments where electronic devices and systems operate. The effects of the interaction of energetic particles on a semiconductor material are also explained with the aim of introducing the different types of single events that upset integrated circuits. Then, they are defined the issues related to the integrated circuit characterization starting by a technology overview, radiation tests, test conditions and real-life tests. Finally, a brief description of the generalities of the multi-core processors is provided.

2.1 Radiation environment

Electronic circuits and systems are exposed to natural and man-made radiation in spatial and atmospheric environments. During the sixties, many problems were observed in space electronics, although it was difficult to separate soft-failures from other forms of interference. In 1978, it appears the first evidences of malfunctions in electronic circuits embarked in spacecraft caused by the radiations of the space (May, 1979).

2.1.1 Spatial radiation environment

In the outer space, there are three main types of radiative sources that affect the Earth's atmosphere:

- Galactic and extragalactic cosmic rays
- Radiation coming from the sun such as solar wind and solar flares
- Earth's magnetic field which comprises the magnetosphere and the radiation belts.

Cosmic rays

A cosmic ray is a high-energy particle that travels throughout the galaxy including the solar system. The sun is the origin of some of these particles, but most of them come from galactic and extragalactic sources and are known as Galactic Cosmic Rays (GCRs) (Wulf, 2016). In 1912, the Austrian physicist Victor Hess made a historic balloon flight ascending up to 5300 meters and measuring the rate ionization in the atmosphere. He found that it increased

to some three times that at sea level and concluded that penetrating radiation was entering the atmosphere from above. He had discovered cosmic rays (CERN, 2016).

Cosmic-ray particles that reach the top of the Earth's atmosphere are termed primarily cosmic rays. About 85 percent of GCRs are protons (nuclei of hydrogen atoms), the lightest and most common element in the universe, approximately 12 percent consisting of alpha particles (helium nuclei), and the remaining are electrons and nuclei of heavier atoms that are typical nucleo-synthesis end products. The energy of primary cosmic rays ranges from around 1GeV to as much as 108 TeV. Figure 2.1 depicts the flux of the cosmic rays particles as function of their energy expressed in electron volts.

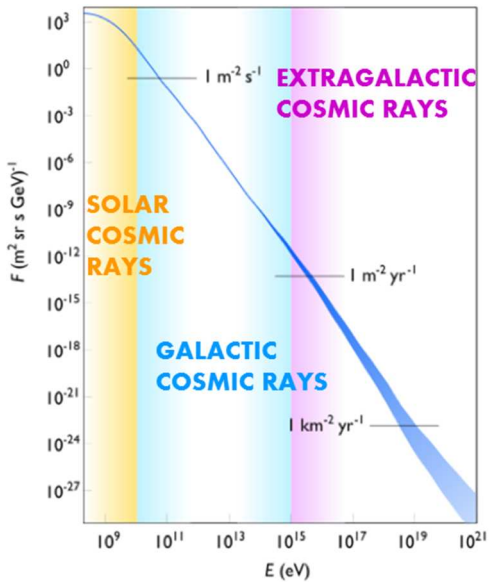

Figure 2.1: Energy Spectrum of Cosmic Rays (Lafebre)

Solar wind and solar flares

The Sun is mainly composed by hydrogen and helium that is a product of a nuclear fission reaction in the sun's core. The sun losses mass in form of high speed protons and electrons leaking away from its out layers in all directions at speeds about 400 Km/s. This flux of particles and plasma is called solar wind (McConnell, 2016).

A solar flare is defined as a sudden, rapid and intense variation in brightness observed on the Sun's surface due to a blast of hot gases. It releases a lot of energy in form of electromagnetic radiation that involves a very broad spectrum of emissions. According to NASA, the amount of energy released is the equivalent of a million of 100 megaton hydrogen bombs exploding at the same time (Taylor, 2015). The flare ejects clouds of electrons, ions, and atoms through the corona of the sun into space (Barth, 2003). These clouds can reach the

Earth one or two days after the event. They produce radiation across the electromagnetic spectrum at all wavelengths.

Radiation belts

The Van Allen radiation belts are two torus layers of energetic particles that are held in place around the magnetic field of the Earth. The main belt extends from an altitude about 10,000 and 60,000 kilometers above the surface. These radiation belts are mainly composed by energetic protons and electrons coming from solar wind and cosmic rays. The belts are situated in the inner region of the Earth's magnetosphere. The outer belt is formed by energetic electrons, and the inner belt is formed by a combination of protons and electrons (Barth, 2003). Other particles like alpha particles and heavy ions are present in less quantity. Satellites that orbit significant time in radiation belts must protect their electronic devices that may result damaged by the particles' effects.

Radiation belts presence had been conceived before space age but confirmed by the Explorer I and III missions on January 31, 1958 under Doctor James Van Allen. The inner Van Allen belt contains high concentration of electrons in the range of hundreds KeV and energetic protons with energies exceeding 100 MeV. It extends normally from an altitude of 1000 Km to 6000 Km excluding geographical areas such as the South Atlantic Anomaly (SAA) where the inner boundary may descend approximately 200 Km above Earth's surface. This leads to an increased flux of energetic particles that exposes orbiting satellites to higher levels of radiation.

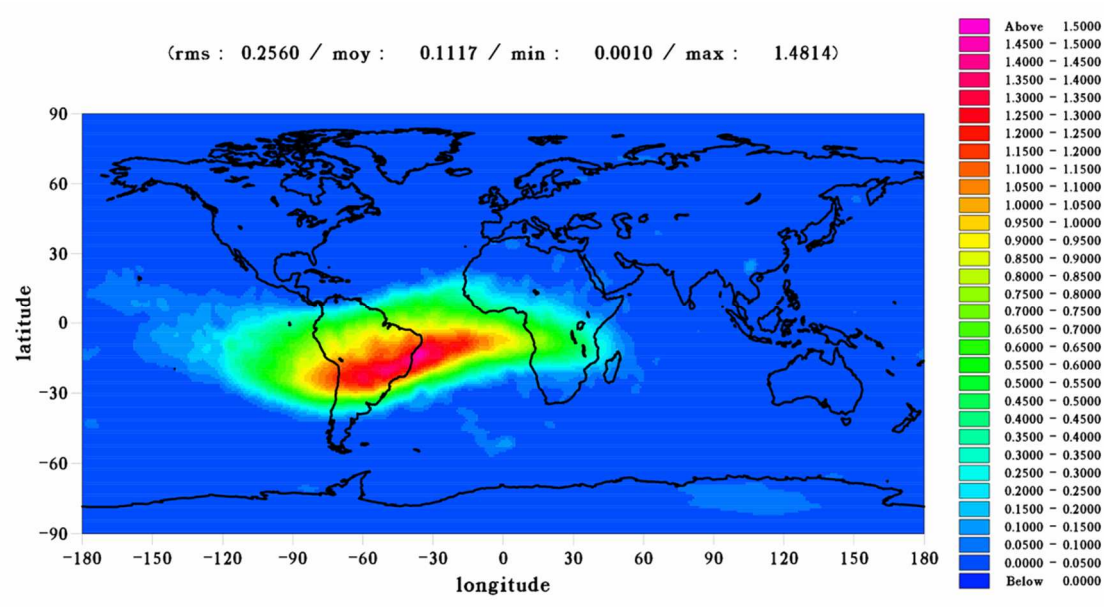

Figure 2.2: SSA measured by Jasson-1 (CNES/CLS)

The anomaly in the inner radiation belt results from the fact that the planet's magnetic field is not perfectly aligned with its geographic center and poles. The magnetic field is slightly stronger in the north and moves around the geographic poles leaving the south Atlantic area closer to the inner radiation belt. The effects are not relevant on the Earth's surface but they are very significant to orbiting satellites. Figure 2.2 shows the Jason-1 exposure to South Atlantic Anomaly effects measured on 2000-2004 period using the Doris ultra-stable oscillator (Lemoine, 2006).

2.1.2 Atmospheric radiation environment

Many forms of natural and artificial radiation are encountered in the Earth's atmosphere which can be beneficial or damaging to the environment. Special attention should be given to ionizing radiation since it can be harmful to microelectronics in excessive amounts.

Cosmic rays and solar particles are constantly bombarding the Earth's atmosphere and thus, colliding with atoms and molecules mainly nitrogen and oxygen (Barth, 2003). This interaction generates an air shower of secondary particles which products are protons, electrons, neutrons, heavy ions, muons and pions. Figure 2.3 illustrates the structure of an air shower in the atmosphere initiated by a high-energy proton.

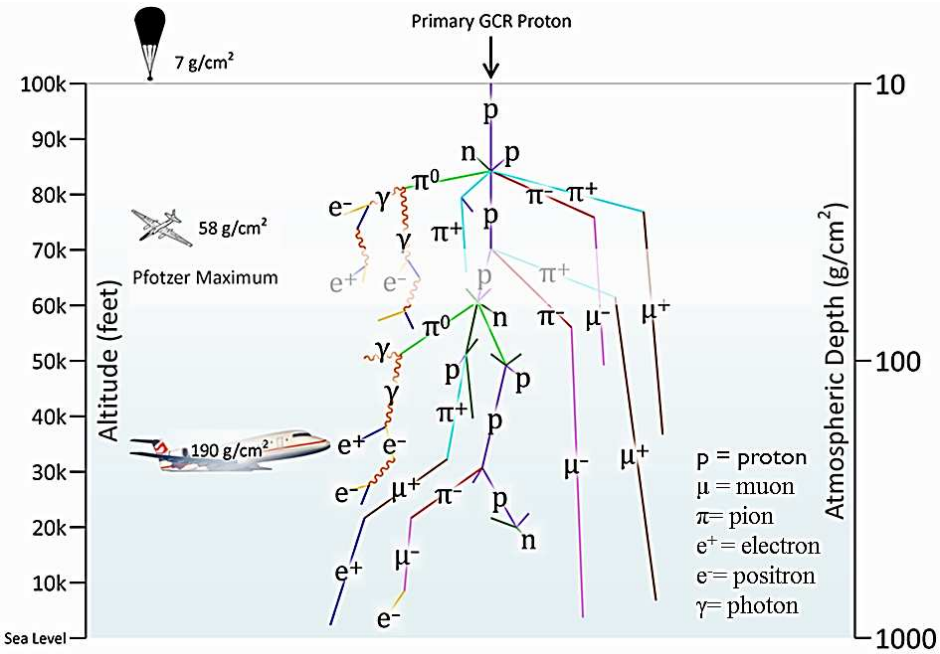

Figure 2.3: Particles interaction in the atmosphere (E.V. Benton)

In terms of radiation effects in the atmosphere, neutron particles are the most important followed by heavy ions and at the end the pions which effects are almost negligible. Neutrons are measurable at 330 Km altitude and their density increases until reach a peak at about 20 Km. However, below 20 Km altitude, the level of neutrons decreases, and at ground level the peek flux falls about 500 times. At avionic altitude neutrons are the dominant particles with energy above 100 MeV. Until 90's, only neutrons which energy exceeds 100 MeV were considered dangerous for electronic devices. Nevertheless, with the miniaturization of transistors, circuits become more sensitive to low energies. Although neutrons are not ionizing particles, they can hit with different atoms belonging to the physic layer of an integrated circuit generating ions that may produce faults. Figure 2.4 shows the variation of the different particle's fluxes in function of the altitude at 54º latitude. Note that near avionic altitudes the particle flux is maxima.

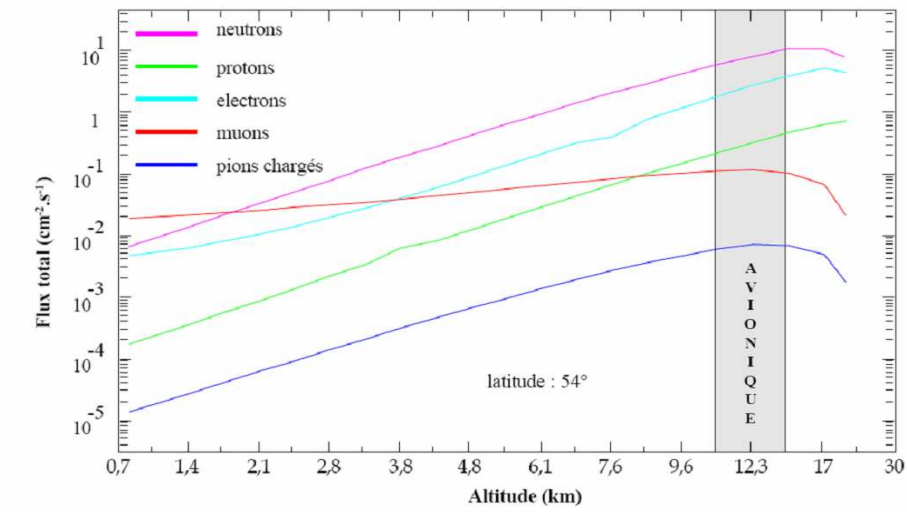

Figure 2.4: Particles Flux vs Altitude (O'Brien 1978)

2.2 Radiation effects on electronic circuits

Both natural and artificial radiation are present on the Earth's atmosphere, being the terrestrial and cosmic rays the most important sources that produce effects in microelectronics. The physic phenomenon can be explained as follows. When a particle passes through a piece of material, it has a certain probability of interacting with the nuclei or with electrons present in that material through the electromagnetic forces (Tavernier, 2010). Considering a very thin portion of matter, this probability is proportional to the thickness of the portion of matter and to the number of possible target particles per unit of volume in the material.

The effects of the particle interaction depend on the physic properties of the particle and the target, and can be nuclear or coulombic interaction. In the nuclear case, particles can

interact with the nucleus either by elastic and non-elastic manner giving a part of its kinetic energy to the recoils. In the coulombic case, it occurs an electrostatic interaction between electrically charged particles. If this interaction generates free carriers in the matter, this is called ionizing interaction. As the particle moves in the matter, its speed reduces until stopping when the entire energy is lost.

Stopping power is the average linear rate of energy loss of a heavy charged particle in a medium. This quantity has fundamental significance in radiation physics and is the result of two phenomena that slow down the progression of the incident particle. These two interaction phenomena are the electronic energy loss that produces ionization in the matter, and the nuclear energy loss that do not produce ionization.

The energy loss $S(E)$ of an incident particle interacting with matter can be considered as the sum of ionizing energy losses and non-ionizing energy losses. The term LET (Linear Energy Transfer) is used in dosimetry to describe the ionizing stopping power while the term NIEL (Non Ionizing Energy Loss) is used to describe the non-ionizing stopping power. The following equation expresses the total energy loss:

$$S(E) = \frac{dE}{dx} = \left(\frac{dE}{dx}\right)_{electronic} + \left(\frac{dE}{dx}\right)_{nuclear}$$

The radiation phenomena may cause transient, permanent and destructive effects in integrated circuits. The radiation effects can be grouped in two categories: cumulative effects and Single Event Effects (SEEs).

2.2.1 Cumulative effects

Also called dose effects result from the interaction between the particles and the insulating of electronic circuits. The absorbed dose is defined as the ionizing energy deposited to matter per unit mass and is expressed in Gray. A Gray is equivalent to the absorption of one Joule per kilogram of material (J/Kg) (Cleland, 2005). The Gray is the SI unit for absorbed dose. Another unit still used is the Rad (radiation absorbed dose). One Gray equals 100 Rad. Note that the dose shall always be referred to the absorbing material such as Si, SiO2, GaAs, etc.

Ionizing dose

Integrated circuits can suffer ionizing damage when energy deposited in a semiconductor or in an insulating layer frees charge carriers. These carriers diffuse or drift to other locations where they may get trapped, leading to unintended concentrations of charge and parasitic fields. It affects mainly devices based on surface conduction such as MOSFETs (Ratti,

2011). Depending on the material, electrons can reach the conduction band and thus free holes in the valence band. In the case of silicon slightly doped, it is assumed that the generation of electron-hole pairs is equivalent to the density of the charge carriers in equilibrium. The effect is temporary and then it will disappear.

Non-ionizing dose (Displacement Damage)

Incident particles displace atoms from their lattice site. The resulting defects alter the electronic properties of the crystal leading to circuit's malfunction. Significant effects of this non-ionizing dose include the increase of leakage current and the modification of semiconductor's doping. Displacement damage is the primary mechanism of device degradation for high energy neutrons irradiation, although a certain amount of atomic displacement may be determined by charged particles. This damage mainly affects devices on bulk conduction (e.g. BJTs, diodes, JFETs).

Total dose effects

The effects of the total dose are due to the progressive build-up of trapped charged in insulating layers or at Si/SiO2 interface as a consequence of ionization. Also, they are produced by the defects in the bulk of the devices originated from displacement events (Ratti, 2011).

2.2.2 Single event effects

In electronic devices, SEEs refer to all the possible effects induced by the interaction of a single energetic particle with a semiconductor material. This effect can be the result of a direct ionization of the material produced by a heavy ion or a proton, or an indirect ionization caused by neutrons. SEEs normally appear as transient pulses in logic circuitry or as a bit-flip in memory cells or registers. These effects are classified in *hard errors* that are non-recoverable errors and *soft errors* that may be recovered by a reset, by rewriting the information, or by a power cycle (Gaillard, 2011). SEEs are mainly the product of the deposition and collection of charge over a sensitive node or volume of the circuit. Figure 2.5 illustrates the mechanism for SEE production.

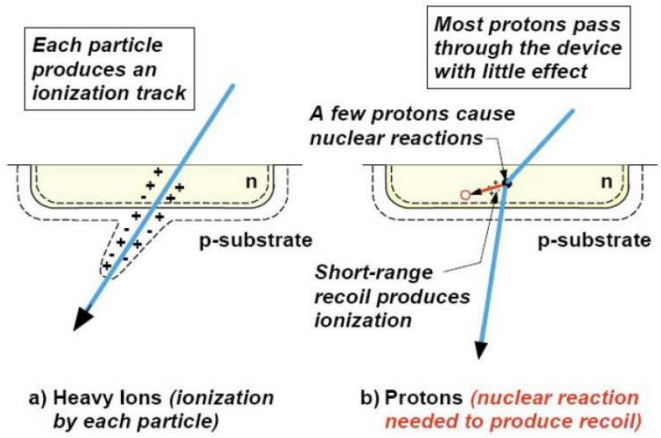

Figure 2.5: Mechanism for SEEs (Johnston. JPL)

Single Event Transient (SET)

The SETs, also called Analog Single Event Transients (ASETs) are principally transient pulses in analog circuits such as comparators, operational amplifiers, reference voltage circuits, etc. In combinational logic, SETs are transient pulses produced in a gate that can propagate in a combinatorial circuit path for being ultimately latched in a storage cell (Gaillard, 2011).

Single Event Upset (SEU)

A single event upset is a bit-flip in a memory element of a semiconductor device such as memories, latches and registers. These upsets are recoverable errors since they do not cause damage to the device. SEUs are random in nature and can be cleared with the next write operation in the corrupted memory location or by power cycling the device (Microsemi, 2011). When a single event perturbs many storage cells, a Multiple-Cell Upset (MCU) is obtained. If more than one bit in the same word is upset by a single event, a Multiple-Bit Upset (MBU) is obtained. The occurrence of this type of multiple events becomes more frequent with the device miniaturization.

Single Event Functional Interrupt (SEFI)

A Single Event Functional Interruption (SEFI) is the loss of functionality due to perturbation of control registers or clocks in a complex integrated circuit. The SEFI may give burst of errors or hangs. Functionality may be recovered by a power cycle, a reset, or a reload of the configuration register (Gaillard, 2011).

Single Event Latch-up (SEL)

A Single Event Latch-up may be triggered by a PNPN parasitic structures in bulk CMOS technology. A SEL is associated with a strong increase of power supply current. SELs can be destructive by overheating of the structure and localized metal fusion. This event needs a power cycle to be deactivated.

Single Event Burn-out (SEB)

Single Event Burn-out is the destruction of a power device such as IGBT or power MOS due to the thermal runaway resulting from the combination of a parasitic bipolar transistor and the avalanche mechanism.

Single Event Gate Rupture (SEGR)

Single Event Gate Rupture is a destructive event that results in the breakdown and subsequent conducting path through the oxide gate of an n-channel or p-channel power MOSFET transistor. An SEGR is manifested by an increase in gate leakage current and can lead either to the degradation or complete failure of the device (Jesd89A, 2006).

2.2.3 Technological advances issues

Regarding the consequences of SEEs in integrated circuits, the most significant issue is related to the technology scaling. In fact, size scaling means more than reducing the geometry feature size of the transistor. It also comprises other technology advances issues such as the use of new materials (e.g., alternative-k dielectrics), oxide changes, material resistance, new interconnection structures, etc.

Furthermore, changes in integrated circuit manufacturing lead to lower operating voltages, lower nodal capacitance and higher integration density. All of these factors increase SEU and SET sensitivity and also intensify the potential of MBU effects. Reducing the critical charge required to produce an upset through the variation in operating voltage and nodal capacitance, may produce an increase in the sensitivity of a specific circuit, since the deposited charge during ionization is invariant and the voltage transient produced is proportional to the deposited charge.

Another characteristic of advanced devices refers to the increasingly higher clock speeds which impacts in the generation of SET. Operating at high speeds may have two consequences: a further voltage transient propagation through a multi-stage circuit without attenuation, and more possibilities for a transient to be latched into a storage element.

Process technology

Process technology refers to the semiconductor device material and fabrication node (transistor's channel length) 90nm, 45nm, 28nm, etc. Bulk CMOS is the most popular semiconductor material and refers to a chip built on a standard silicon wafer.

Manufacturers scale transistors at each node by reducing the channel length in order to integrate more cores in a single die. As a result, the channel may suffer the short-channel-effect that degrades the sub-threshold slope or turn off the characteristics of a device. Another concern is the transistor variability. It occurs when a bulk CMOS transistor performs differently from its nominal behavior which may produce random differences in terms of threshold voltage. This phenomenon is called random dopant fluctuation (RDF) and is caused by the vibration of the dopant atoms in the channel (LaPedus, 2016).

For solving (RDF) and channel issues, manufacturers incorporated high-k/metal gate technology to 28 nm bulk CMOS that improves the transistor having a very robust node. However, problems remains with bulk CMOS since the channel region below the gate is depleted of mobile charge which leaves the dopant atoms ionized. There are two main solutions to face bulk CMOS drawbacks. One possibility is a fully depleted transistor technology such as FD-SOI, and another possibility is finFET.

Fully-depleted Silicon–On-Insulator (FD-SOI) is a planar technology which incorporates a thin insulating layer of Silicon dioxide (SiO2) within the substrate to suppress leakage. FD-SOI allows eliminating the doping and getting essentially the same electrostatic which means better mobility and less variability (LaPedus, 2016).

FinFET is a non-planar or 3D-like structure where the control of the current is achieved by implementing a gate on each one of the three sides of a fin. This technology also solves the bulk CMOS problem but is very expensive.

Despite of the advantages that this two technologies present over the conventional bulk CMOS to impulse device scaling, radiation effects were not well known until recent experimental studies that pointed out radiation behaviors specific to FD-SOI and FinFET architectures. Advanced SOI devices having a small sensitive volume (silicon island) should benefit against SEEs. In these devices, only the charges deposited in the silicon island by an ionizing particle may induce a single event. On the other side, in bulk devices charges deposited far from the active area may also be collected and produce a parasitic current pulse (Gaillardin, 2013).

The work presented in (Baggio, 2005) addresses the SEU sensitivity to protons and neutrons of FD-SOI devices. It was found that FD-SOI has a strong improvement to SEUs compared to bulk technologies. The use of FD-SOI devices instead of bulk CMOS in proton rich environments reduces the SEU sensitivity by more than one order of magnitude without any risk of latch-up. Advanced SOI technologies have valuable properties to operate in harsh radiation environments since they can resist both total ionizing dose and SEEs. Consequently, they are suitable to work in space and avionic applications.

2.3 Characterization of electronic devices to radiation

The technology advances affect the devices' sensitivity to radiation effects, and consequently its error rate and reliability. For evaluating the sensitivity to radiation of electronic devices, real-life tests and accelerated radiation ground tests are commonly used. Both types of tests are also used to obtain the soft-error rate and reliability. In addition fault-injection techniques are also used to evaluate the behavior of an application executing on a device exposed to harsh radiation environment.

2.3.1 Cross-section

The sensitivity of a device exposed to ionizing radiation is expressed in terms of the cross-section (σ). It is an effective area that quantifies the intrinsic probability that an ionizing particle crossing 1 cm^2 area produces an event type SEE. The following equation defines the cross-section as the average number of particles required to cause a SEE.

$$\sigma = \frac{\mathrm{Nev}}{\phi} \tag{2.1}$$

Where Nev is the number of detected events and ϕ is the particle fluence, that is the particle flux integrated over a certain period of time. For semiconductor memories where the capacity is known, σ can be expressed in cm^2/bit or cm^2/device.

The Device immunity to radiation is determined by its linear energy transfer threshold (LET_{th}). The LET_{th} is defined as the minimum LET to cause a SEE at a particle fluence of 10^7 ions/cm^2. The curve of the cross section versus LET specifies the characterization of an integrated circuit to SEE. Figure 2.6 illustrates a sample cross-section vs LET curve.

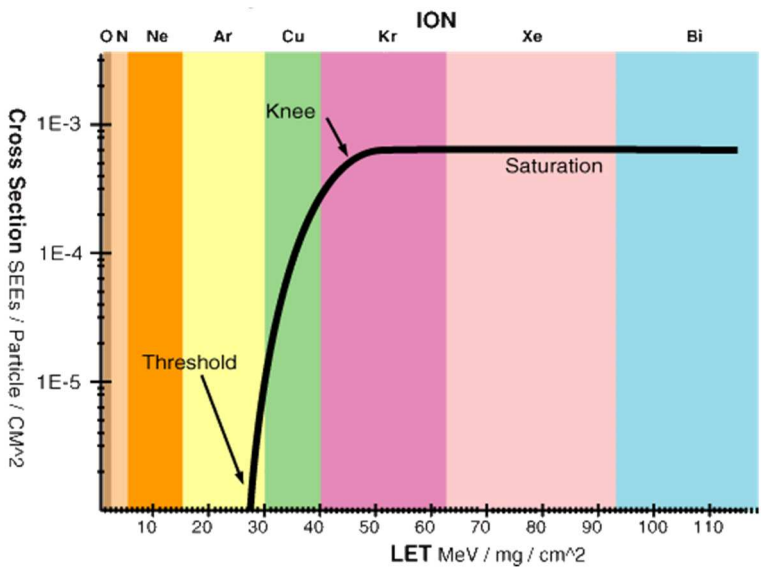

Figure 2.6: A sample Cross-section vs LET curve

2.3.2 Error-rate

In general, a soft error is a type of error in which a signal or datum is wrong. In a device memory, a soft error modifies data values or instructions. However, a soft error will not damage the hardware, just data that is being processed. In the spacecraft industry the SEUs are also called soft-errors.

The Soft Error-Rate (SER) is the rate at which soft errors appear in a device or system for a given environment (Gaillard, 2011). When the environment is known, the SER can be expressed in Failure in Time (FIT) or in Mean Time Between Failure (MTBF). One FIT is equal to a failure per billion hours. The sensitivity of semiconductor memories is often given in FIT/Mb or FIT/device. The FIT value can be predicted by simulation or is obtained experimentally in radiation facilities.

The cross-section of a device can be used to calculate the SER as follows:

$$\text{FIT value} = \sigma \, x \, \emptyset \, x \, 10^9 \tag{2.2}$$

Where σ is the device cross-section and $\emptyset$ is the particle flux in the real environment expressed in n/cm^2/h. For example, in New York City (NYC) where the neutron flux is about 14 neutrons per cm^2 per hour (Mukherjee, 2008), given a device which cross-section is 2x10^{-15} cm^2/bit, the number of FIT/Mb is:

$$\frac{\text{FIT}}{\text{Mb}} = 2\text{x}10^{-15} \, \frac{\text{cm}^2}{\text{bit}} \times 14 \, \frac{\text{n}}{\text{cm}^2.\text{h}} \times 1\text{x}10^9\text{h} \times 1\text{x}10^6\text{bit} = 28 \tag{2.3}$$

2.3.3 Device reliability

The reliability is the probability of having no failure in a semiconductor device within a given period of time (Shooman, 2002). The reliability is function of the failure rate. For electronic devices the failure rate is considered as a constant. When failure rate (λ) is constant, the following equation is applied.

$$R(t) = e^{-\lambda t} \tag{2.4}$$

Being $\lambda = \sigma * \emptyset$, where σ is the cross-section of the device and $\emptyset$ is the particle flux in a given environment.

2.3.4 Real-life tests

Real-life test is the most direct way to measure the Soft Error Rate (SER) in a device. It consists in exposing integrated circuits to natural radiation in different environments such as terrestrial atmosphere at different altitudes (avionic, mountains, stratospheric balloons, etc) or in the space. This is done with the aim of studying the effects of radiation on semiconductor circuits. Some interesting works are presented in (Chee, 2000), (Godhagen, 2000), (Sohn. 2000), (Taber, 1997).

The advantage of this method relies on the fact that devices are tested under standard operating conditions with normal ambient background radiation (Jesd89A, 2006). Consequently, they provide actual and trustworthy results. However, it is needed a large amount of devices that have to be exposed for a long period of time for obtaining statistically significant number of soft errors.

2.3.5 Radiation ground testing

At ground level it can be found several means to characterize integrated circuits to radiation. The radioactive sources, the particle accelerators and the laser beams can be used to obtain useful results in short periods of time compared with real-life test, since the radiation flux that they produce is several orders of magnitude greater than the one existing in the nature.

Radioactive source

A simple and inexpensive means to have a preliminary idea of the sensitivity to radiation of a component is to use a source of Californium 252 or Americium 242. In the case of Californium, alpha particles and two types of heavy ions are emitted giving a LET of 45 and 46 MeV.cm2/mg. The main limitation of this radioactive source is the penetration depth of ions (about 6 to 15μm) because of their low energy compared with those found in space environments and particle accelerators (Peronnard, 2009). If the DUT have considerable

surface layers, the ions will not reach the sensitive zones even if the device is thinned. However, these sources of radiation may be useful to validate the test platforms and thus validate the logistic before do further testing in particle accelerators.

Particle accelerators

A particle accelerator is an apparatus that uses electric fields to propel charged particles to nearly light speed while increasing their energy and magnetic fields to contain them in a well-defined beam. The goal of this machine is to provide energetic particles to investigate the structure of the atomic nucleus and many aspects of particle physics (CERN, 2015).

There are two main types of accelerators: straight line accelerators where the particle beam travels from one end to the other end, and circular accelerators where a beam of particles travels repeatedly round a loop. Linear type accelerator like Van de Graaf as well as circular type accelerators like cyclotrons, are very useful for integrated circuit characterization since they produce ionizing particles that allows evaluating the SEE sensitivity of an electronic circuit.

Laser beam

A laser is a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation. Laser beams are present in thousands of applications in daily life including electronics, medicine, industry, military, entertainment, and they are a key technology in fiber-optic communications (Wikipedia, 2016a).

Laser beams are an important means for the characterization of integrated circuits since they allow simulating SEEs effects. As described in (Buchner, 1987), this technique permits to test circuits rapidly for upsets sensitivity focusing in very tiny spots of a circuit. The main advantage of laser beams consists in allowing the mapping of the sensitive zones of a chip. This cannot be done by particle accelerators since the particles reach the entire surface. However, lasers have two main limitations: the beam reflection by the metallization layers, which is more problematic in complex components multi-layers, and the fact that deposited energy by photons during the test has no correlation with the LET of an energetic particle (Pouget, 2001).

2.3.6 Radiation evaluation issues

Ken LaBel in his document “Radiation Testing and Evaluation Issues for Modern Integrated Circuits” presents a useful list of attributes related to the increase of functionality and design complexity that can affect the utilization, testing and characterization of advanced integrated circuits (LaBel, 2005).

Intelligence

Modern integrated circuits include in their architecture embedded processors, microcontrollers, programmable fabric and other circuits that provide certain autonomous operation as well as a diversity of operating configurations that must be evaluated. This attribute makes difficult the selection of appropriate test and fault coverage. In addition, many of these devices implement protection mechanism such as error detection and correction (EDAC) that may affect testing and error-rate prediction.

Flexibility and programmability

Single events can perturb the programming capability of modern devices built-in embedded SRAMs or EEPROMs. When it occurs, the architecture configuration is lost or rearranged. This attribute can affect the approach of testing and characterization for SEEs depending on the storage mechanisms.

Complexity

Integrated circuits include a variety of different circuit types and technologies with different sensitivities and failure modes. This complexity affects error-rate predictions because an IC involves many cross-sections and LETs, test performance, test facility, test beam selection and other test considerations.

Integration density

Integrated circuits have millions of critical nodes, feature that makes fault coverage and test vector selection problematic.

Hidden circuit features

Circuits regularly have thousands of registers, built-in test elements, and other embedded circuits not identified by the manufacturer. These zones may not be accessible to the external user but can impact on the overall radiation response of the circuit.

Multi-layer construction

Circuits are often constructed using many levels of metal and complex packaging that make the SEE testing of critical nodes in radiation facilities very difficult due to beam energy limitations. The multi-layer construction makes impossible the use of diagnostic tools such as lasers or ion-micro beams. Additionally, the over-layers can contribute secondary particles that impact the radiation response of the device making difficult the error-rate prediction.

Power requirements

Circuits consume and dissipate significant amounts of power. It is important to consider this issue when testing in a vacuum chamber or reduced spaces.

Speed of operation

The high operating frequency of modern integrated circuits requires that SEE testing be compatible with this feature in order to obtain conservatives error-rate estimates. If a long cabling is required for testing, the high operating speeds can cause major problems.

2.3.7 Fault-injection

To validate systems operating under harsh radiation environment, the traditional real life and accelerated ground tests are used. However, their high cost[1] and their availability have boosted the use of fault-injection techniques. Fault injection is a useful technique for validating the dependability of devices or systems (Arlat, 1990). It provides a way to improve the coverage of hardware and software testing by introducing faults in a controlled manner into system's hardware or code paths with the aim of observing their behavior in presence of faults.

Numerous fault injection techniques and tools have been developed and tested. They can be classified into: Hardware-Based Fault Injection, Software-Based Fault Injection, Simulation-Based Fault Injection, Emulation- Based Fault Injection and Hybrid Fault Injection (Benso, 2003).

Hardware-Based Fault Injection

Also called Hardware-Implemented Fault Injection (HWIFI) is a technique used to induce faults at hardware level using external physical sources like the environment parameters, power supply disturbances, laser fault injections, or the modification of input pins of the circuit.

Software-Based Fault Injection

Also called Software-Implemented Fault Injection (SWIFI) is a technique used to reproduce at software level the errors than would have been produced when a fault target the hardware. It involves the modification of the program running on the target system to provide the ability to perform the fault injection.

Simulation-Based Fault Injection

It is a technique that consists in injecting the faults in high-level models (VHDL models) with the aim of evaluating the dependability of the system when the model is available.

[1] For instance the cost of radiating heavy-ions by particle accelerators is around 650 USD per hour.

The main advantages of this approach are the observability and controllability that allow accessing most of the sensitive zones of the device. Faults can be modeled and simulated with a fault simulator (Kooli, 2014).

Emulation-Based Fault Injection

It is a technique based on the use of Field Programmable Gate Arrays (FPGAs) for emulating the target system with the objective of reducing the time spent during simulation-based fault injection campaigns (Ziade, 2004).

Hybrid Fault Injection

It is a technique that combines the versatility of software-implemented fault injection with the accuracy of hardware monitoring. The aim of this approach is to totally exercise the system under analysis.

2.4 Multi and many-core processors

A multi-core processor is defined as a single computing component which contains two or more independent processing units (cores) with the aim of enhancing performance, reducing power consumption and providing simultaneous processing. The instruction set is similar to a conventional CPU instruction set, but multiple cores execute in parallel multiple instructions increasing the overall performance of the circuit. Multi-core processors integrate several cores packaged into a single integrated circuit die (Chip Multi-processor or CMP), or multiple dies in a single chip package.

The based principles of current multi-core processors were developed during 1975-2000 for parallel supercomputers, while the principles of current many-core processors, such as the KALRAY MPPA -256, are based on the parallel supercomputers developed during 2009-2015. For instance, the MPPA-256 is based on shared memory nodes of the multi-core type connected to each other by means of specialized networks and with explicit routing. A new age for the supercomputers has begun from June 2016, since the first Top500 supercomputer is based on a many-core processor with 260 cores.

The architecture issues comprise at least three main aspects: the memory and I/O management, the multiplicity of cores, and the inter-core communications (Vajda, 2011). Figure 2.7 illustrates a multi-core architecture with n cores, each core has two levels of private caches (L1, L2) and one L3 shared cache memory.

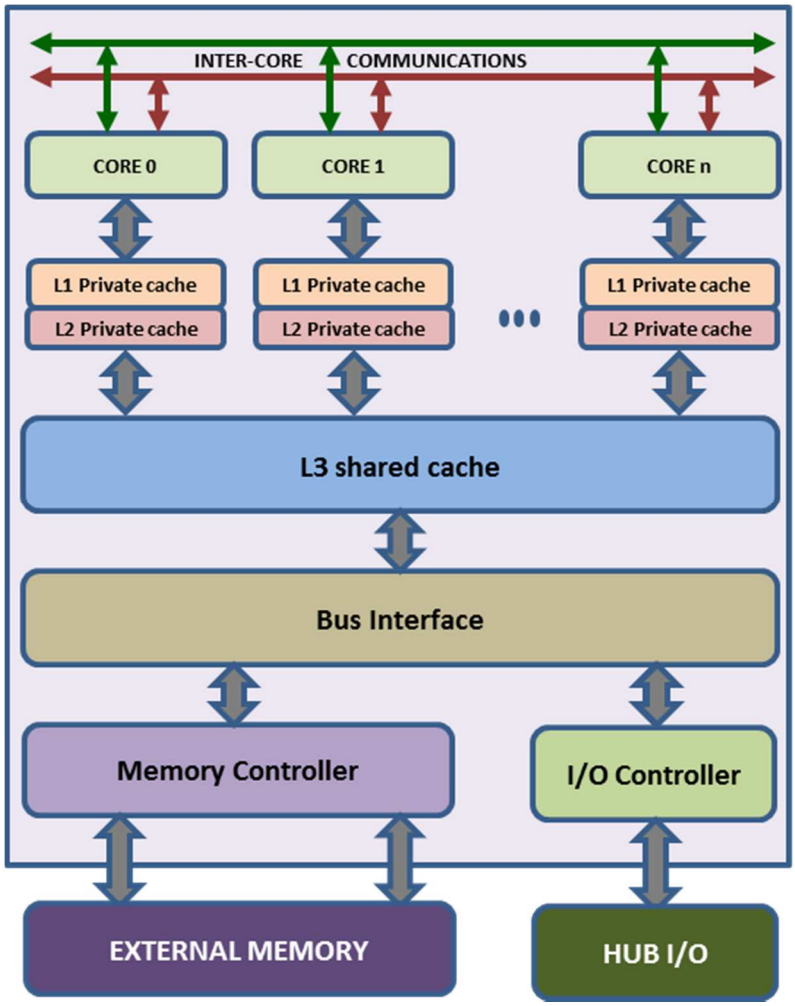

Figure 2.7: Architectural concept of a multi-core processor

2.4.1 Memory and I/O management

For accessing shared resources synchronization is needed. However, scaling synchronization in systems with large number of cores is very difficult. Therefore, the memory and I/O controllers have an important role to manage resources. Regarding the memory controller, it has to maintain consistency between the memories. In general, the memory hierarchy models propose the use of various levels of *private, shared* or *mixed* cache memories in multi-core processors. For many-core processors also a *distributed* memory is proposed.

Cache memory

A cache memory is a very fast and small-sized type of volatile memory integrated directly in the CPU chip used to reduce average cost in terms of time and energy to access data from the main memory. Cache memories hold frequently used data which can be easily retrieved by the processor instead of accessing the main memory. Most processors have independent instruction and data caches. Data cache is often organized as a hierarchy of cache

levels (L1, L2, L3, etc.). Most of multi-core processors have a split L1 cache for instructions and data, its own dedicated L2 cache, and a shared L3 or high-level cache (Wikipedia, 2016b).

When a processor needs to read from or write to a location in the main memory, it first checks weather a copy of that data is in the cache producing a cache hit or a cache miss.

Cache hit: It occurs when the processor finds the requested memory location in any cache line that might contain that address. In this case the processor immediately reads or writes the data in the cache lines which is much faster than accessing the main memory.

Cache miss: It occurs when the processor does not find the memory location in the cache. In this case, the processor accesses the main memory and transfers data in blocks of fixed size called cache lines. A cache entry is created when a cache line is copied from the memory into the cache. The cache entry includes the copied data as well as the requested memory location that is called tag.

Cache performance: The percentage of accesses resulting in cache hits is known as cache hit-rate. It is a measure of the efficiency of the cache for a given program or algorithm. When a read miss occurs, it delays the execution of the program since data have to be transferred from main memory. Write misses do not cause such penalty because the processor continues executing while data is copied to main memory.

Cache entry structure: Cache row entries usually have the following structure

Tag	***Data block***	***Flag bits***

The cache line (data block) contains the data fetched from the main memory. The tag contains part of the actual data address. The flag bits specify whether or not a cache line has been filled with valid data. The quantity of main memory data that the cache can hold is the cache size. This size can be estimated as the number of bytes stored in each cache line times the number of lines stored in the cache (Wikipedia, 2016b). An effective memory address is split into tag, index and block offset.

Tag	***Index***	***Block offset***

The index describes the cache row (line) in which the data has been put in. The index length is $log_2\ (r)$ bits for r cache rows. The block offset identifies the required data within the stored data block inside the cache row. The effective address is expressed in bytes, and then the block offset length is $log_2\ (b)$ bits, where b is the number of bytes per data block. The tag

comprises the most significant bits of the address, which is checked to see if it is the one needed (Wikipedia, 2016b).

For example, the Freescale e500mc processor implements separate 32 KB eight-way set associative L1 instruction and data cache with 64 bytes each line. Hence, there are 32 KB/64 Bytes = 512 cache blocks. The number of sets is equal to the number of cache lines divided by the number of ways of associativity, which leads to 512 blocks /8 way = 64 sets, and hence 2^6 different indices. As there are 64 bytes per cache line, then there are $2^6 = 64$ possible offsets. Since the processor physical address is 36 bits wide, this implies 24+6+6 = 36, being 24 bits for the tag field.

Associativity: The replacement policy decides where to copy a new entry of the main memory in the cache memory. If the cache is full associative, the replacement policy is free to choose any entry in the cache to put the copy of data. On the other side, the cache is direct mapped if each entry in the main memory has to be copied in one place in the cache. An N-way associative is a middle solution where the cache implements a compromise in which each entry of the main memory can be placed in N places in the cache.

The most important memory management issues are related to the bottleneck produced by the implementation of cache coherency protocols among large number of cores, the synchronization access to the same memory space, the memory bandwidth and the memory latency.

2.4.2 Multiplicity of cores

There are some scalability issues produced by the increasing number of cores in a chip, the most relevant comprise homogeneous and heterogeneous architectures. In homogeneous architectures all cores have the same instruction set architecture (ISA) and performance, while in heterogeneous architecture at least two cores differ on ISA and/or performance and/or functionality. Another difference is that in most of homogeneous architectures, there is implemented shared memory with full cache coherency. From a paralleling programming point of view, a homogeneous architecture is easier to program. On the contrast, heterogeneous architecture allows better partitioning of specific tasks.

Another important scalability issue is related to the manufacturing technology level. Due to the nano-metric size of transistors, the quantum effects should be considered since they decrease the hardware reliability.

2.4.3 Inter-core communications

Traditionally the *inter-core communications* have been achieved through a shared *common bus*. For avoiding bottlenecks in memory and I/O accesses, they have been implemented various levels of local cache memories. Some general purpose multi-core processors use cross-bar interconnections including levels of cache memories. Other technologies propose multiple ring buses and, for largely number of cores the use of network-on chip (NOC) is preferred.

2.4.4 Software abstraction layers

Due to the complexity of this kind of devices, the software abstraction layer model plays an important role to facilitate the development of programming by abstracting the underlying implementation and only revealing objects or actions that the developer requires. Figure 2.8 illustrates a classical software abstraction layer model for multi/many-core processors to access hardware resources.

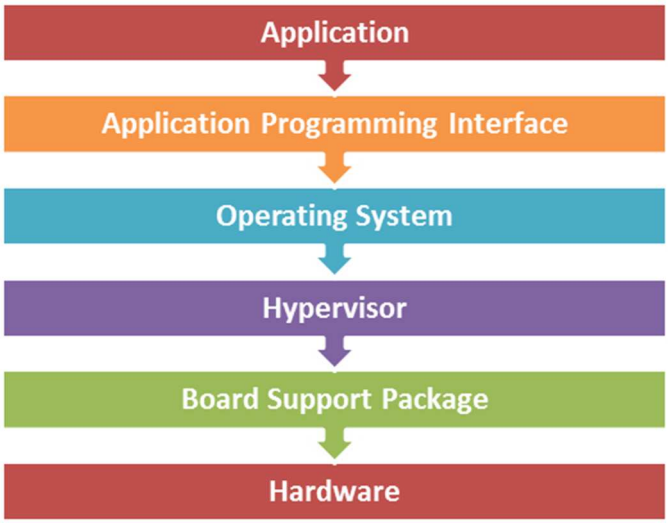

Figure 2.8: Software abstraction layer model

The Board Support Package (BSP) access directly to the hardware resources. There is no abstraction of hardware capabilities and components. BSP includes the libraries to manage all the functionalities provided by the device.

The Hypervisor functions serve to isolate the operating system (OS) from the hardware. It acts as an *OS host* that allows different OSs running on the same hardware. It provides partitioning and manages shared resources to avoid interfering.

The ***OS*** includes the Universal bootloader (u-boot) to start the system and the kernel functions to manage the resources and schedule the tasks on the cores.

The ***Application Programming Interface (API)*** is as set of definitions, protocols and tools used for build applications that run on top the OS. APIs are used to facilitate the programming model for the user. Common APIs used in multi/many-core processors are POSIX, OpenMP, CUDA, OpenCL. It is important to note that one API could be defined on the top of another. For instance OpenMP is defined on the top of POSIX.

During the design stage of a system, the programmer has to choose the programming model for its application. All the programming models include the BSP and application layer. The intermediate layers are optional and their utilization depends on the system design and the developer requirements.

When no intermediate layer is used, the programming model is called *bare-metal* or *bare-metal*. This model is the most flexible and complex one since the programmer has almost no restriction to hardware resources and has the control of each function. However, all the capabilities must be programmed: the startup of the cores, the task distribution among the cores, the synchronization between cores and/or tasks, the access to shared resources, the coherency, consistency etc.

On the other hand, when the programmer select a programming model based on an API such as OpenMP, developing the application is easier. Nevertheless, there are several restrictions in terms of hardware resources access such as read/write operations on special purpose registers that are reserved for OS or *hypervisor*.

2.4.5 Multi-processing mode

A multi-core processor is a flexible device that commonly is able to operate in different multi-processing modes. The main two modes are: Symmetric Multi-Processing mode (SMP) and Asymmetric Multi-Processing mode (AMP). In SMP mode, a single Operating System (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 as a common memory. This architecture provides fast performance since processes and threads are distributed among CPUs (Freescale, 2012).

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. All CPUs must cooperate to share the resources. Hence, AMP mode is very useful when working with embedded systems (Freescale, 2012).

2.5 Discussion

The effects of natural radiation have a considerable impact on electronic circuits especially at avionic altitudes, for space missions and other safety-critical applications. For this reason, aerospace and avionics require to mitigate these effects in order to guarantee the dependability in their applications. Mitigation techniques such as Radiation Hardening by Process (RHBP) and RHBD applied on devices are well known for accomplishing dependability requirements. Nevertheless, the price and availability of these dedicated devices become a problem for many avionic and spatial projects. Consequently, industrial and academics are interested in using COTS devices for their applications.

Multi-core and many-core processors offer a huge processing capacity and high performance by executing parallel computing. However, having more complex devices implies a high degree of miniaturization which makes the chip more sensitive to radiation. Despite of this manufacturing limitation, these devices provide a suitable solution to face reliability problems benefiting of the multiplicity of cores for implementing fault tolerance based on redundancy. Therefore, the evaluation of the impact of radiation in the reliability of these devices is mandatory to validate their use in harsh radiation environments.

In order to address this issue, this thesis proposes: 1) the evaluation of the intrinsic sensitivity to SEE of the multi and many-core processors by means of radiation ground testing, 2) the evaluation of the sensitivity to SEU of applications implemented in multi and many-core processors by fault injection campaigns, and 3) the prediction of the application error-rate by combining the results issued from radiation test and fault injection.

From the fault injection techniques presented in section 2.3.7, the SWIFI technique was selected as fault injection strategy since it allows running a large number of fault injection experiments. Moreover, it does not require dedicated complex hardware, gate-level netlist or RTL models that are described in hardware description languages. Consequently, it is suitable to evaluate COTS multi and many-cores. The next chapter will present the state of art of the sensitivity to SEE of multi-core and many-core processors. Several SWIFI techniques are also presented and compared in order to choose the fault injection approach for the evaluation of the application sensitivity to SEU. At the end of the chapter, they are described some approaches to predict the application error-rate of microprocessors.

Chapter 3 : State of the art

This chapter is devoted to present the state of the art of the sensitivity to SEE of multi-core and many-core processors obtained by radiation ground testing, the software-based fault-injection techniques applied to these devices, and the methods for predicting the application error-rate in mono-processors since to the author´s knowledge there are no prediction works dealing with multi-core processors.

3.1 Sensitivity to SEE of multi-core and many-core processors

In the literature, there are few works dealing with the sensitivity to single event effects of multi-core and many-core processors. The most relevant are presented below:

In (Stolt, 2012), it 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. The target device was an HP c7000 BladeSystem multi-core server designed for aircraft altitudes. The device was exposed to 14 MeV neutrons for simulating the effects of high energy neutrons present in the atmosphere at aircraft altitudes. The server is composed by six Intel X5570 based HP server blades and six interconnect modules. The operating conditions for the tests include the selection of the operating system, the BIOS setting, the processor utilization, and the input/output utilization. This work estimates that the cross-section per bit for 45nm CMOS technology at 14 MeV neutrons is $1x10^{-14}$ cm^2/bit. In addition, this work provides a fault handling comparison between Windows 5.2 and Linux 5.1 operating systems.

This work defines a reliability model for a server composed by six multi-core platforms operating in SMP mode under operating system. The model is very useful from a system point of view since the total error-rate is the sum of the partial error-rates of each component of the electronic board containing the multi-core. However, obtained results are hard to extrapolate to a multi-core processors based on different technologies and configured in different multi-processing mode.

In (Guertin, 2012) are presented the SEE test results of the 49-core Maestro Interim Test Chip (ITC) microprocessor. Maestro is a Radiation Hardened By Design (RHBD) processor for space applications based on the Tilera TILE64 processor. This 90nm many-core is produced by the Onboard Processing Expandable Reconfigurable Architecture (OPERA)

program and built by the Boeing Solid State Electronics Development (SSED). Experimental tests have been conducted at the Texas A/M University's (TAMU) cyclotron facility using 15 and 25 MeV ions. During the tests they were targeted the 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.

The results presented in this work are meaningful to evaluate the mitigation of errors provided by the design hardening techniques as well as the protection mechanisms. However, these results cannot be extrapolated to COTS devices which are not hardened.

The work presented in (Santini, 2014) proposes a generic metric (Mean Workload Between Failures) to evaluate the reliability of an embedded processor devoted to execute safety-critical applications. It considers both cross-section and exposure time, and It demonstrates that on modern embedded processors, enabling the caches memories may provide benefits to critical systems since the larger exposed sensitive area may be compensated by a shorter exposure time of the application which results in an overall improvement in terms of reliability. The proposed metric is experimentally validated through extensive radiation test campaigns targeting a 28nm Commercial-off-the-shelf ARM-based SoC. Radiation experiments were conducted at Los Alamos National Laboratory (LANL) and Los Alamos Neutron Science Center (LANSCE) with white neutrons source that emulates the energy spectrum of the atmospheric neutron flux. The failure probability of a bare-metal application is decreased when L1 cache is enabled. It is shown that it is not enough to rely only upon the cross-section to ponder reliability.

This approach for reliability evaluation of processors is based on the device cross-section and the execution time of the application. The study proves that for the COTS target device the best choice is to enable L1 caches but not L2 ones in order to improve the performance of the system without compromising its reliability.

Reference (Oliveira, 2014) presents the radiation sensitivity evaluation of cache memories and internal resources of a modern Graphic Processing Units (GPUs) designed in 28nm technology node. In addition, several hardening strategies based on Duplication With Comparison (DWC) to reduce GPU radiation sensitivity are presented and validated through radiation experiments. The device under test was the NVIDIA K20 that contains a compute-unified device architecture (CUDA)-based GPU, composed of streaming multiprocessor (SM) with the ability of executing several threads in parallel. The cross-section per bit of the L2 cache and shared memories were experimentally obtained in Los Alamos facility using 14 MeV energy neutrons. Three different DWC strategies were designed to mitigate radiation-induced effects on GPU's used in safety-critical and HPC applications. The efficiency of the proposed strategies was experimentally evaluated and compared with chip's ECC protection mechanism.

It was demonstrated that DWC strategies can be more effective than ECC when input data are duplicated.

The cross-section measurement of the 28nm technology node provides valuable information about the impact of the technology scaling in the sensitivity of the device. This sensitivity could be extrapolated to a multi-core processor case, but it is important to consider the nature of each device. The GPUs are co-processors implementing small caches intended to accelerate computations. In turn, the CPUs tend to have large portion of cache and internal memories which increase substantially its SEE sensitivity.

3.2 Software-based fault-injection techniques

SWIFI is the most convenient fault injection technique for evaluating applications running on COTS devices since it does not require dedicated complex hardware, gate-level netlist or RTL models that are described in hardware description languages. All types of faults can be injected in accessible memory cells such as registers and memories that represent the most sensitive zone of the chip. The main drawback of SWIFI techniques is their intrusiveness since they modify the program. This fact may affect the scheduling of tasks. If timing is not a concern, this type of fault- injection can be considered as non-intrusive. Otherwise, the timing involved during the injection can disrupt the system's operation (Ziade, 2012). Therefore, for critical-embedded systems a fault-injection technique with low intrusiveness is required. Some relevant SWIFI approaches are briefly described.

FIAT (Fault Injection-based Automated Testing): is an automated real-time distributed accelerated fault-injection environment developed at Carnegie-Mellon University, USA. It provides facilities for defining fault classes, which are the relationship between faults and error patterns, and for specifying the way long errors will strike and interact with the object (code or data) in execution. Initial version can inject faults in user application code, data, task and timers. Later versions capabilities include fault injection in operating systems (Segall, 1988).

FTAPE (Fault Tolerance and Performance Evaluator): is a tool developed at the University of Illinois for comparing fault-tolerant computers. It combines a system-wide fault injection with a controllable workload. This tool can inject faults as bit-flips to emulate errors in memory locations, user-accessible registers and disk accesses. This is done by inserting a special disk driver into the operating system (Tsai, 1994).

DOCTOR: is an integrated software fault-injection environment, developed at the University of Michigan, for evaluating system dependability by means of injecting various

types of faults: processor faults, memory faults and communications faults that can be permanent, transient or intermittent faults. DOCTOR can use three different trigger mechanisms: time-out triggered memory faults, traps for generating non-permanent faults, and program instruction changes during compilation for producing permanent faults (Han, 1995).

EXFI: is a low-cost fault-injection system for embedded microprocessors-based boards developed at the Politecnico di Torino, Italy. The Kernel of the EXFI system is based on *Trace Exception Mode* that exists in most microprocessors. The tool is capable to inject single bit-flip in memory images and user-registers of the processor. This method can be extended to support different fault models, specially spatial and temporal multiple bit-flip. It is a non-intrusive tool which does not require any change in the source code (Benso, 1998).

MAFALDA: (Microkernel Assessment by Fault injection Analysis and Design Aid): is a generic tool aiming at providing the characterization of the failure modes in presence of injected faults, and incorporating wrappers to improve these failure modes. This tool randomly performs fault injection in microkernel components and kernel-calls. It is performed by means of an interface to set-up and carry-out fault injections. The obtained results of experiments in two instances of commercial microkernels reveal their weak behavior (Rodriguez, 1999).

BOND: Is a fault-injection tool developed at Politecnico di Torino aiming at simulating faulty behavior in COTS software programs of a computer system running under Windows NT 4.0 OS. It exploits the idea of interposition agents in order to guarantee low impact in the execution of the program. This tool allows statistical and deterministic fault injection into different location such as code, data sections, heap, stack, processor register, system calls, without requiring any modification of either the OS or the target application (Baladini, 2000).

CEU (Code Emulating Upsets): is a fault injection approach developed at TIMA Labs for processor-based electronic boards. It contains a device capable of executing instruction sequences and supporting asynchronous interruptions. Bit-flips are injected by software means concurrently with the execution of the program in response to an interruption signal assertion. The interruption handler targets a memory cell or accessible register which is altered XOR-ing it content with an appropriate mask value (Velazco, 2000).

GOOFI (Generic Object-Oriented Fault Injection): is a fault injection tool developed at the Department of Computer Engineering at Chalmers University of Technology in Sweden. GOOFI is a tool designed to be adaptable to several target systems and different fault injection techniques. Its main feature is the portability between different host platforms since it relies on Java programming language and SQL compatible database (Aidemark, 2001).

JACA: is an open source fault injection tool developed at the State University of Campinas in Brazil. This tool allows injecting faults in object-oriented systems and can be adapted to any Java application. Using JACA, it is possible to perform low-level fault injection upsetting the assembly code, and high-level fault injection during runtime upsetting the attributes values, method parameters and return values in a Java program (Martins, 2002).

FERRARI (Fault and ERRor Automatic Real-time Injection): is a fault injector, developed at the Texas University, based on software traps that inject errors in the system. Software traps are triggered either by the program counter when pointing the desired program location, or by a timer. This tool is able to emulate transient errors and permanent faults to evaluate the dependability features of complex systems (Kanawati, 2002).

XCEPTION: is a commercial fault injection tool for dependability analysis developed at the University of Coimbra, Portugal. This tool benefits of the advanced debugging and performance monitoring resources available in modern processors to emulate realistic faults by means of software. It also uses the processor's exceptions to trigger the faults and monitor their impact on the behavior of the target system (Costa, 2003).

FAUMachine: is an open source virtual machine, developed at the Friedrich Alexander University of Erlangen-Nuremberg in Germany that permits to install a full operating system (Linux, Windows, DOS, Open and NetBSD) and run them as if they were independent computers. This is similar to other virtual machines but it supports fault injection capabilities for experimentation. FAUMachine allows fault injection in memory cells such as transient bit-flips, permanent stuck-at and coupling faults; faults in disk, CD/DVD such as transient and permanent block faults; and network faults such as transient, intermittent and permanent send or receive faults (Potyra, 2007).

LFI (Library Fault Injection): is a fault injection tool that automates the preparation of fault scenarios and their injection at the boundary between shared libraries and applications. LFI provides programmers a fast, easy and comprehensive method to test program robustness in presence of failures that are exposed at the interface between shared libraries and the applications under test (Marinescu, 2009).

FIES: is a fault injection framework for evaluating software-based Self-tests according to the safety standard IEC 61508. This virtual platform supports widely-used embedded COTS processors such as ARM cores. It provides feedback about the diagnostic coverage of self-test in early design stages. This approach also supports the simulation of faults in the control and execution path of an ARM processor and features an extended fault model to simulate memory coupling faults (Höller, 2014).

Table 3.1 summarizes the main characteristics of the described approaches. It can be seen that EXFI, BOND, CEU, JACA, XCEPTION, FAUMACHINE and FIES are low-intrusive tools. They are thus suitable for been used to evaluate critical-embedded applications.

Table 3.1: SWIFI tools summary

Fault Injection tools	***Technique***	***Software Level***	***Target area***	***Fault generation***	***Intrusiveness***
FIAT	Modify kernel	O S	Memory, registers, communications	Fault list	High
FTAPE	Memory/Register modification	O S	Memory, registers	Random	High
DOCTOR	Fault injection agent	O S	Memory, registers, communications	Probabilistic Past event	High
EXFI	Trace exception	O S	Memory image, code, registers	Fault list	Low
MAFALDA	Interception kernel calls	OS	Microkernel	Random	High
BOND	Interposition agents	OS	Code data sections, registers, function call parameters.	Fault list	Low
CEU	Interruptions	Bare-metal	Memory, registers	Random	Low
GOOFI	Pre-runtime Scan-chain	Bare-metal	Memory	Random	High
JACA	Computational reflection	OS	Attributes and methods Java application	Fault pattern	Low
FERRARI	System calls	OS	Memory process	Random	High
XCEPTION	Debugging	OS	Memory, data bus, address registers	Fault list	Low
FAUMACHINE	Virtualization kernel compilation	OS	Memory, disk, registers, network	Random	Low
LFI	Interception Library modif.	OS	Shared libraries	Fault profile	High
FIES	Dynamic translation	OS	Memory, registers	Fault defined XML	Low

Regarding the software level, it is relevant to consider that by definition a SWIFI technique could not target all the sensitive zones. In addition, if the fault-injection code runs on OS, there are some OS privileged functionalities that cannot be accessible by the fault-injector. Therefore, for achieving a better effectiveness, it is preferred a bare-metal level technique since it could access more chip resources. Indeed, when certification of an

application running on critical-embedded system is required, it is commonly tested in bare-metal (Girbal, 2015).

From the listed techniques, only CEU and GOOFI work at bare-metal level. The main difference between them is that CEU injects faults by means of interruptions at run-time, while GOOFI injects faults at compilation-time or by means of Scan-chain. The main disadvantage of GOOFI is that it does not target processor's registers. In addition, Scan-chain fault injection works only for devices compatible with this feature.

As the objective of this research is to provide an evaluation approach for multi-core and many-cores as much general as possible, CEU was selected as a base fault-injection approach.

3.3 Error-rate prediction

The estimation of the application error-rate is essential to validate the device and the application in terms of reliability. The objective of this estimation is to know if the processor is suitable to be used in harsh radiation environments. In the literature, there are significant works dealing with error-rate prediction of mono-processors. However, the selection of the appropriate approach for multi and many-core processors depends on several technological, architectural, and application aspects.

SER prediction based on probabilistic models

Reference (Wang, 2008) presents an environment dependent method to estimate the neutron induced soft error-rate (SER) by means of the propagation of single-event transient pulses through the affected logic circuit. The pulsed is modeled by two parameters: the probability of occurrence and the probability density function of the pulse width. In the analysis it was considered the entire neutron LET spectra of the terrestrial background. FIT rates were calculated for ISCAS85 benchmark circuits. In comparison with other SER analysis works, this method considers more factors such as the sensitivity region of a device, electrical masking and circuit technology, which influence the SER. In this probabilistic soft-error model the logic SER can be very responsive to aspects like circuit characterization, sensitive region calibration and process variation which make complex the soft-error estimation for logic circuits.

This probabilistic model is devoted to estimate the soft error-rate in logic circuits where SEEs exist as SETs. The experiments provide interesting results but authors make some technological and probabilistic assumptions that may affect the accuracy of the model.

SER prediction based on the sensitivity of constituent elements

Reference (Cabanas-Holmen, 2011) proposes an approach for predicting the single-event error-rate of a RHBD processor based on the sensitivity of the constituent circuits. This bottom-up approach for single-event error-rate analysis integrates the error rate caused by radiation events in different type of circuits including SEU/SET sensitivity. This analysis takes into account components like SET rate generated in the reset and clock trees that propagate to flip-flop inputs, the flip-flop SEU rate that corresponds to the intrinsic SEU sensitivity, the flip-flop SET rate generated in gates between flip-flops, the SRAM SEU rate generated by uncorrectable errors, analog circuits error rate (PLL rate). All of these components are affected by a derating and utilization factor. Heavy-ions radiation campaigns were performed at the Texas A&M University cyclotron to validate the approach on two chips: a RHBD version of the ARM Cortex™ R4 core and a radiation hardened single-core Tilera processor. Functional tests confirmed expected uncorrectable SRAM errors and give some perceptions of the sensitivity due to SET in data path logic and clock and reset trees. Data integrity error rates varied more than two orders of magnitude while the recoverable error rate differed by less than a factor of two. The recoverable error-rate prediction was within 35% of the rate calculated from measured results. Finally, heavy ion testing demonstrates that the SEE performance of 90nm RHBD tested microprocessors is very attractive for many space applications.

This bottom-up approach provides a good estimation of the error rate of the device since many internal components can be targeted. However, the application of this method is not feasible for COTS microprocessors which design or building blocks are not available for end users.

SER prediction based on the cache activity

In (Tang, 2011), it is presented a lifetime model for the private L1 cache in multiprocessors which is based on the activities and the states of cache lines. This model is applied to characterize and predict cache's vulnerability trend in multiprocessors. This experimental evaluation shows that cache vulnerable phases due to remote accesses increase dramatically as the number of cores increases. It is also proposed a protocol enhancement to prematurely invalidate cache lines in modified state for minimizing the vulnerability factor due to remote reads to modify cache lines as well as other phases starting with a write operation. The experimental results confirm the effectiveness of the proposed self-invalidation in improving the data cache reliability by reducing D-Dir phases due to protocol operations as well as reducing D-Repl. vulnerable local phases.

In this work, the proposed model is implemented using the MV5SIM simulator. In spite of the effective characterization of the vulnerability of the L1 cache, this model does not take

into account other sensitive components such as registers and shared memories; additionally, it needs to be complemented with hardware sensitivity for providing error-rate prediction.

Based on the worst-case sensitivity

In reference (Velazco, 2000) is presented an application error-rate prediction approach based on SWIFI. The prediction is achieved by combining the worst-case sensitivity of the technology, obtained by accelerated radiation tests, with the error-rate issued from fault injection $\tau_{SEU} = \tau_{inj} * \sigma_{STATIC}$. The Code Emulating Upset (CEU) code is responsible for the bit-flip injection in general purpose registers or addressable internal or external memory locations of microprocessor-based architectures. Bit-flips are injected randomly in location and time. For triggering the execution of the CEU code, it is necessary the assertion of an external interruption which interrupt handler pointed to the mentioned code. It is achieved by a dedicated test platform. Experimental results on two different boards built around the 80C51 microcontroller and the 320C50 digital signal processor showed the capabilities of this strategy since the predicted results were very close related with data obtained in radiation tests.

This approach provides an effective error-rate prediction for evaluating the sensitivity of an application implemented in microprocessor architectures. The main advantage of this approach is its applicability to any processor without a detailed architectural knowledge. Nevertheless, there are two limitations to be considered: 1) it is not possible to target all possible sensitive areas like internal flip-flops, control unit and latches inside, 2) upsets occurring during the instruction execution cannot be simulated.

In (Mukherjee, 2003) is described a method for generating accurate estimates of processor error-rate. This method defines a structure's architectural vulnerability factor (AVF) which is the probability that a fault in a particular structure will result in an error. The error rate of the structure is the product of its raw error-rate, determined by the process and circuit technology, and the AVF. For the experiments it was instrumented an Itanium2® -like IA64 processor simulator. This tool maps bit-level micro architectural state to some cases: 1) dynamically dead code, 2) pre-fetches, 3) wrong path instructions. In these cases, a fault will not affect correct execution, generating per structure and AVF estimations. Using an OS simulation front-end, Red Hat Linux 7.2 was modeled in detail. By tracing the subgroup of processor state bits necessary for architectural correct execution, the AVF is estimated. In absence of correction techniques, any fault in a memory cell that holds one of these bits would produce an observable error in the output of the program. Results show that per-structure AVF estimates should help microprocessors designers to assess the FIT rate of an entire processor in the design cycle. If the processor's FIT does not fulfill the application requirements, these

estimates can help designers to select the suitable error detection or correction schemes to make specific structures less vulnerable to SEUs.

This method provides an accurate estimation of the processor error-rate dedicated to system designers to make appropriate cost/reliability trade-offs. Nonetheless, the AVF is not easily applicable for the evaluation of COTS devices since it requires a detailed knowledge of the system architecture.

In (Housanny, 2012) it is proposed a methodology to evaluate the real cache sensitivity of an application aiming at calculating a more accurate failure rate. This method lies on the monitoring of cache accesses to evaluate cache sensitivity, and requires a microprocessor simulator. In this work, it was used as a target the LEON3 soft-core with several benchmarks. The soft error-rate (SER) is calculated by multiplying the worst-case sensitivity (SER_{RAW}), the architecture derating factor (Π_{arch}), and the software derating factor (Π_{SW}). The (SER_{RAW}) corresponds to the intrinsic sensitivity of the technology and is obtained by means of static tests in accelerator facilities. The (Π_{arch}) is deduced from the analysis of the cache architecture (protection mechanisms and cache management policy) based on the expected errors in the application environment. The execution of an application in a microprocessor simulator combined with the use of a dedicated cache analyzer tool is performed to obtain the (Π_{SW}). The validation of the results was done by means of fault injection on one implementation of the processor running the same programs. The identification of silent bits in the instruction cache gave a better estimation of the instruction cache sensitivity. The proposed approach has predicted all errors with a small overestimation. In addition, this methodology can be implemented in any cycle-accurate simulator providing cache performance information.

This approach is able to provide a good estimation of the error rate of cache memories. However, advanced multi-core and many core processors also implement shared memories which are the largest sensitive zone of the devices. Furthermore, this approach do not consider register's evaluation, which is a big limitation.

3.4 Conclusion

In the literature, it can be found very few studies regarding the sensitivity of multi-core and many-core processors. Due to the widespread use of these devices in embedded systems, it is necessary to evaluate the sensitivity of other COTS multi/many-core processors having different architectural models and technologies in order to give some guidelines in the selection of a device.

Regarding the error-rate prediction, significant approaches are validated for single processors. Among them, the CEU approach was selected to be extended to multi-core and many-core processors due to the following reasons:

- This approach can be applied to any COTS processor unlike other prediction approaches based on architecture and timing vulnerability factor which are not available in COTS components.
- Three works have demonstrated the effectiveness of the CEU approach (Velazco, 2000), (Rezgui, 2001), (Peronnard, 2008).
- CEU is able to target internal and external memories and accessible processor's registers of the device.
- The prediction is based on the real sensitivity of the application and not in probabilistic models or cache memory usage models based on simulators.

Concerning the disadvantages of the approach, the main drawback is the impossibility to target sensitive areas like internal flip-flops, control unit and latches. However, the contribution to the failure rate of these elements is very small.

In chapter four is described the methodology and tools for evaluating the SEE sensitivity of multi-core and many-core processors. An approach based on CEU principles is proposed for predicting the application error-rate of multi/many-core processors.

Chapter 4 : Methodology and tools

This chapter describes the methodology for evaluating the SEE sensitivity and predicting the error-rate of applications implemented on multi-core and many-core processors. At the beginning of the chapter a general overview of the proposed approach is provided. Then, the details of the approach are discussed. Finally, the description of the tools is presented.

4.1 Overview

The suitable metric for accomplish the evaluation of the SEE sensitivity of a system is the failure rate. Typically, for obtaining the failure rate of a system operating in a harsh radiation environment, dynamic radiation tests are performed in order to extrapolate the obtained results to the desired radiation environment. However, the cost and availability of radiation facilities are major constraints. Furthermore, due to the failure rate of a system based on multi/many-core processor is application dependent, it is required to apply a methodology to determine its effective sensitivity to radiation at lower cost. Certainly, a viable option is the prediction of the error rate based on the worst-case sensitivity of the device.

The purpose of the present thesis is to propose an approach based on the CEU principles for predicting the application error-rate of multi-core and many-core processors by combining fault-injection and radiation tests. The CEU principles can be implemented in any mono-core processor without a deep architectural knowledge. However, for multi/many-core processors there are several constraints that have to be overcome due to the complexity of the devices mainly related to the memory management and inter-core communications.

To validate the proposed method, it is implemented on three representative devices that have different manufacturing technologies, design concepts and architectural models. The effectiveness of the method will be determined by comparing the predicted error-rate with the measured one obtained from dynamic radiation tests.

Moreover, it is important to note that the evaluation of the reliability of an application is a compulsory step for devices intended to be used in safety-critical applications or to operate in harsh environments. Therefore, the obtained failure rate is used to compute the reliability of the systems.

Finally, the design of both, fault-injection and radiation experiments is addressed using quantitative theory. It is thus necessary to identify independent and dependent variables involved in the experiments. Independent variables are those affecting outcomes. Dependent variables are those that depend on independent variables: the outcomes or results of the influence of independent variables.

4.2 Code Emulating Upsets (CEU) approach

The CEU approach is a SWIFI fault injection and error-rate prediction approach developed at TIMA laboratory (Velazco, 2000) for studying the effects of upsets on the execution of microprocessor-based architectures. The prediction is done by combining two different strategies: fault injection and accelerated radiation test. Fault injection aims at providing the sensitivity of the application implemented in the processor, while radiation tests aims at obtaining the intrinsic sensitivity of the device.

Fault injection campaigns are performed in order to calculate the CEU rate (τ_{inj}), that is defined as the average number of injected faults needed to cause an error in the result of the application. It is required to execute the application a huge number of times in order to obtain enough quantity of samples for statistics.

$$\tau_{\mathrm{inj}} = \frac{\text{Number of errors}}{\text{Number of injected faults}} \tag{4.1}$$

On the other side, radiation tests allow to obtain the static cross-section (σ_{STATIC}) which provides the average number of particles needed to cause a bit-flip in the device memory cells. The σ_{STATIC} is the worst-case sensitivity of the device and is obtained by the following equation:

$$\sigma_{STATIC} = \frac{\text{Number of Upsets}}{\text{Fluence}} \qquad \text{Revisited (2.1)}$$

Combining the CEU rate and the static cross-section, it is possible to predict the error rate of an application implemented in a microprocessor as follows:

$$\tau_{SEU} = \tau_{inj} * \sigma_{STATIC} \tag{4.2}$$

This method lies on the injection of bit-flips randomly in location and time by using asynchronous interrupts during the execution time. When the asynchronous interrupt is assert, an interrupt handler (*CEU code)* is launched. This code is in charge of producing the selected error (SEU, MBU) in a randomly chosen memory cell (*CEU target*). It performs read and write operations in internal registers accessible via the instruction set, as well as in internal or external memory locations. The aim of the approach is reproducing the effects of SEU faults.

In order to inject a bit-flip in the *CEU target*, the following tasks are done by the *CEU code*:

- Reading the content of the target memory cell.
- Performing an XOR operation with an appropriate mask value that contains a “1” for the bits that are going to be flipped and “0” elsewhere.
- Writing the corrupted value to its original location.

Once the CEU has been injected, the processor restores the context from the stack and continues with the program execution. At the end of the execution, the results of the program are compared with a set of correct values obtained when the program executes without corruption in order to identify errors.

This approach has two limitations to be considered: (1) upsets occurring during the instruction execution cannot be simulated, and (2) it is not possible to target all possible sensitive areas such as internal flip-flops, control unit and latches inside processor’s architecture. In spite of these limitations, this approach was demonstrated to be very efficient and able to provide error rates results close to those obtained in radiation experiments (Velazco, 2010). This can be explained by the fact that for processors implementing different levels of cache memories as well as internal shared memory, the target memory area comprises most of the total sensitive area of the device, providing a significant validity to this approach.

When the CEU approach was designed, the goal was to simulate bit-flips in a non-intrusive way. Therefore, the interruption was asserted by an external device. A dedicated test board called THESIC (Testbed for Harsh Environment Studies on Integrated Circuits) developed by TIMA was used to control the experiment, managing the fault-injection and monitoring the DUT (Device Under Test). The THESIC board was monitored by a *host* computer via Ethernet communications. Later, an updated version of the THESIC platform called ASTERICS (Advanced System for the Test under Radiation of Integrated Circuits and Systems) was developed. The objective was to implement the required analog and digital environment for the operation of the DUT by means of a FPGA.

4.3 Error-rate prediction approach for multi-core and many–core processors based on CEU principles

Up to now, the CEU approach has been successfully applied and validated for mono-core processors. However, the complexity of the processors has significantly increased due to the manufacturing technology, device architecture, number of cores, interconnections,

functionalities, etc. Therefore, it is reasonable to validate a new approach for complex devices such as multi/many-core processors.

Due to the large number of functionalities and pins that complex processors implement, it is not further possible to use the ASTERICS platform for injecting fault in this kind of devices. It is thus convenient to extend the CEU approach to multi/many-core processors benefiting of the multiplicity of cores by using one of them as fault injector while the others execute the chosen application. In order to isolate the fault injector, the device has to be configured in AMP multi-processing mode. For performing the fault-injection, the inter-core interrupts[2] are used.

Considering that multi/many-core processors implement different types of memory cells (shared memories, cache memories, processors registers, etc), the total error rate must be expressed as the sum of the individual contribution of each component.

$$\tau_{SEU} = \tau_{SEU\_SHARED} + \tau_{SEU\_CACHE} + \tau_{SEU\_REG} + \cdots \quad (4.3)$$

This thesis also proposes the addition of derating factors to the contribution of shared and cache memories for improving the accuracy of the prediction. These factors depend on the memory used by the application and the exposure time to radiation of shared and cache memories.

Memory utilization factor (MF)

It is the amount of memory used by the application with respect to the total memory of the device. This factor is calculated considering the memory space occupied by code and data.

$$\mathrm{Mf} = \frac{\text{Used memory}}{\text{Available memory}} \quad (4.4)$$

Exposure time factor (Etf)

In some particular cases where the multi/many-core processor performs as a co-processor of a development board, it is possible to need synchronization between the co-processor and the *Host* processor in order to log results. This is required when the cores of the co-processor do not have direct access to *printf* functions. This synchronization is achieved by means of a master-slave scheme to guarantee every processor core reports detected errors. However, this communication model has an exposure-time loss penalty. While the communications are performed, it is possible to have SEUs affecting the internal memory of the DUT, which are not detected since at the beginning of each execution memory data are

[2] Also called inter-processor interrupts

reinitialized. Even if the probability of having this condition is very low, an exposure-time loss factor (Etf) should be added in this case. This factor is given by the following equation:

$$\mathrm{Etf} = 1 - \frac{St_{H-M} + St_{M-S}}{Texec} \tag{4.5}$$

where $\mathrm{St_{H-M}}$ is the synchronization time between the Host processor and the master core, $\mathrm{St_{M-S}}$ is the synchronization time between the master core and the slave cores, and Texec is the execution time of the application.

In order to measure the time spent by the communications and synchronization functions, some intrusive and non-intrusive methods are available. Profiling is a non-intrusive form of program analysis that allows measuring the duration of function calls (e.g. Valgrind). However, its use is not effective when the execution time of the functions is too small. In contrast, functions such as *gettimeofday(), clock(), clock_gettime()* are suitable for computing time in functions but they are intrusive. If none of these functions is available, the time can be measured in terms of clock cycles using the internal timers of the processor.

By adding these derating factors, the equation that defines the approach is the following:

$$\tau_{SEU} = \tau_{inj} * \sigma_{STATIC} * \mathrm{Mf} * \mathrm{Etf} \tag{4.6}$$

4.3.1 Fault Injection strategy

Fault injection campaigns are used for simulating the consequences of SEUs in applications running on the multi/many-core processors. Since the fault-injection campaigns are experiments that can be addressed using quantitative theory, the first task is to identify the variables involved in the experiment.

A previous step prior to design the experiments was to get familiar with fault-injection techniques in multi-core processors. On this behalf, it was performed fault-injection on systems configured with OS in SMP mode. The obtained results were published in (Mansour, 2014) and (Vargas, 2014).

Identification of the variables

The independent variables are divided in two groups: the variables depending on the system that are fixed during the experiment, and the variables to be manipulated during the fault injection campaign.

The dependent variable represents the errors produced during fault injection. The results can be classified in: erroneous results, exceptions and time-outs. The silent faults are

not errors but they are artifacts to determine the soft error-rate. Table 4.1 lists the independent and dependent variables for fault injection campaigns.

Table 4.1: Independent and dependent variables in fault injection

	Independent variables	***Dependent variables***	
Manipulated during the experiment	Location (Memory address/ Register)	Errors in results	Silent Faults (artifact)
	Injection Time		
	Bit to be altered		Erroneous results
System dependent	Implemented Application		Exceptions
	Multi-processing mode		Time-outs
	Programming Model		

- Silent fault: it occurs when the injected fault does not cause any consequence in the result of the program. (e.g., typical silent faults are those affecting data never used or data already used by the program).
- Erroneous result: the results of the program are not the expected ones.
- Exception: the program halts. It is primarily caused by faults injected on critical registers. A hang is a type of exception that crashes the system.
- Time-out: When the program does not respond after duration equal to the worst-case execution time.

Before starting the fault-injection campaigns, it is needed to determine the number of cycles required to execute the selected application. It is done in order to know the range of time in which the fault injection should be performed. Also, it serves to determine the time-out value. If the multi/many-core processor evaluated operates in stand-alone mode, the *monitor* functions of the application to determine *time-outs* and *hangs* are accomplished by the *master* core. For the other cases, when the multi/many-core processor works as a co-processor the *monitor* functions are accomplished by the *host* processor.

Fault injection in memory

In this strategy all the variables intended to be used by the application are placed, by software means, in a shared memory (cache or internal). In this way, the variables can be modified at any time by each one of the processor cores. The master core initializes the data that is going to be used by other cores. Once finished this step, it sends a message through an inter-processor interrupt, to indicate the slave cores to start the execution of the application.

Figure 4.1 presents a flow diagram of the memory fault-injection strategy for a quad-core processor.

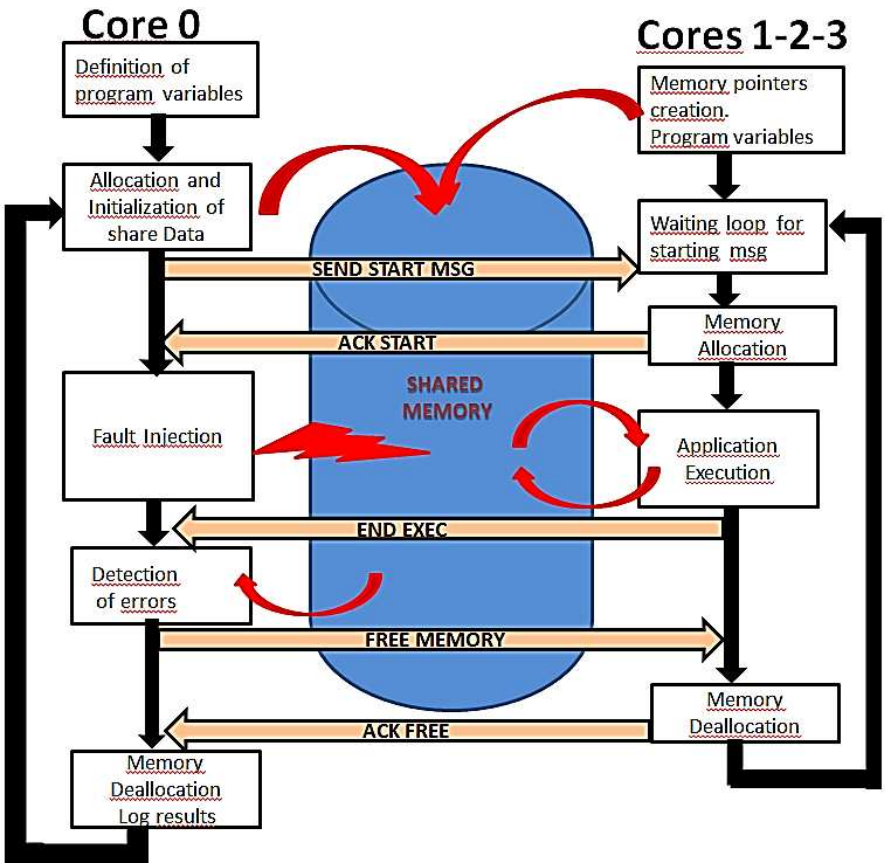

Figure 4.1: Memory fault-injection flow-diagram

Once the message is received by the slave cores, they confirm the reception of the message and start the execution of the application. While the application is running on the slave cores, the master core performs the fault injection. It randomly selects the target core, the injection instant (in terms of clock cycles), the address (global array index) and the bit to be altered. When a slave core finishes its calculation, it sends a message to the master core indicating that the execution was completed. The master core waits the arrival of the messages of each core, and then compares the obtained results with a set of correct results previously obtained.

Since the programmer cannot access directly to cache memories, for simulating SEU faults, the fault-injection is performed in the main memory. This strategy can be applied to any tested application. It only requires storing all the variables of the targeted application into a shared memory space.

Fault injection in processor registers

In this strategy, faults are injected in accessible registers belonging to the processors. Due to the fact that master core has no access to other cores' registers, it can execute an indirect fault injection via the instruction set. The fault injector performs an inter-core interruption to the selected core in which the interruption handler launches a code that targets accessible registers allowing emulating bit-flips in the selected core as previous described. Figure 4.2 presents a flow diagram of the register fault-injection strategy for a quad-core processor.

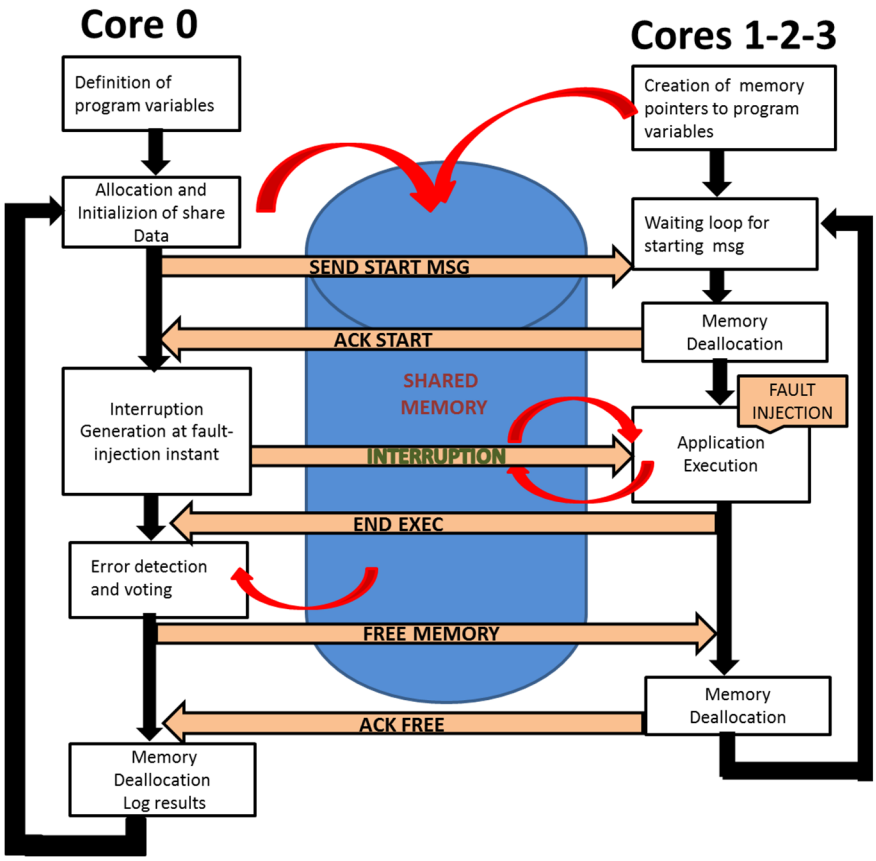

Figure 4.2: Register fault-injection flow-diagram

The target registers to be taken into account by this method are the General Purpose Registers (GPRs), the Floating Point Registers (FPRs), and accessible Special Function Registers (SFRs) or Special Purpose Registers (SPRs). It is important to note that modifying these registers may cause critical failures in the program execution.

Intrusiveness of the strategy

As the fault-injection strategy uses one core as fault injector, it is reasonable to think about the intrusiveness of the approach. For multi-cores having few cores, it is clear that a significant part of the sensitive area corresponding to the fault injector core is not targeted and

thus, should not be taken into account for the estimation of the error rate. On the other hand, when working with many-cores having hundreds of cores, the sensitive area corresponding to the fault injector is negligible. In addition, devices implementing shared memory concentrate most of the sensitive area outside the processor cores. Consequently, the presented fault injection strategy is valid to evaluate the sensitivity of a given application through the estimation of its error rate.

4.4 Radiation Ground Testing

Accelerated radiation ground tests are the fast way to obtain significant results concerning the sensitivity of a device in a short period of time, since the more particles hit the device the more SEE are observed. The reproducibility of the experiment is also another major advantage of this strategy. Two types of tests are considered for evaluating the sensitivity of the device: a static test in order to obtain the intrinsic sensitivity of the memory cells (cross-section), and a dynamic test for evaluating the dynamic response of the application. Furthermore, the static cross-section is used to predict the error-rate.

In this work, the experimental tests have been conducted with 14 MeV neutron radiation to simulate the effects of high energy neutrons present at avionic altitudes, since neutrons are the most representative particles in the Earth's atmosphere. Reference (Miller, 2013) discusses the relevance of using 14 MeV neutron test to characterize the SEU sensitivity of digital devices. Sections 3 and 6 of the JESD89A document of the *JEDEC STANDARD* were used as a base protocol for the experimental tests.

The test plan should define the required supply voltage for the radiation test. Due to the fact that soft error-rate of many devices is very sensitive to supply voltage, it is critically important that this parameter be accurately measured and controlled (Jesd89A, 2006). Carefully adjusting the supply voltage to match the test plan values, will lead to assure consistent results.

Radiation test familiarization

Before designing the experimental tests for multi/many-core processors, it was necessary to get familiar with the radiation tests. On this behalf, neutron radiation tests were performed on different memory devices. The obtained results from the radiation tests for a 90-nm CMOS SRAM memory (CYPRESS CY62167EV30LL) have contributed to the validation of GENEPI2 (GEnerator of NEutron Pulsed and Intense) facility for testing electronic circuits (Villa, 2014). In addition, the results obtained from the radiation tests of the Low Power SRAM memory (A-LPSRAM) manufactured by RENESAS in 150nm CMOS at low bias voltage were published in references (Clemente, 2015) (Clemente, 2016).

Confidence intervals

Due to the scarcity of experimental data issued from accelerated radiation tests because of the availability of the facility and the high cost of the experiments, it is compulsory to add uncertainty margins to the results. For numerous events (typically >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 (Autran, 2014), (Velazco, 2014). Therefore, the following equation has been applied:

$$\frac{1}{2}\chi^2\left(\frac{\alpha}{2}, 2\mathrm{Nerr}\right) < \mu < \frac{1}{2}\chi^2\left(1-\frac{\alpha}{2}, 2(\mathrm{Nerr}+1)\right) \tag{4.7}$$

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

4.4.1 Identification of the variables

Radiation tests are experiments that can be addressed using quantitative theory. Consequently, the first task is to identify the variables involved in the experiment. Table 4.2 lists the independent, intervening and dependent variables for the static and dynamic test. Note that the dynamic test also depends on the system configuration and the implemented application.

Table 4.2: Independent and dependent variables for radiation tests

	Independent variables	***Intervening variable***	***Dependent variables***	
Manipulated (static and dynamic tests)	Neutron flux	Neutron Energy	Errors	SEU
	Distance DUT to target			
	Exposure time			MBU
System dependent (dynamic test)	Implemented Application			MCU
	Multi-processing mode			SEFI
	Programming Model			

The independent variables are divided in two groups: variables depending on the system (exclusive for the dynamic test), and variables to be manipulated during the radiation experiments.

The dependent variable represents the errors observed during radiation tests. They can be classified in single errors, multiple errors, and sequence interruption errors. It is important to consider that depending on the memory architecture of the multi/many-core, single and double errors can be corrected and detected respectively by the protection mechanisms.

4.4.2 Static test

The static test aims at estimating the intrinsic sensitivity to radiation of the memory cells of a processor (technology). The static cross-section (σ_{STATIC}) can be obtained by means of radiation tests in an accelerator facility. The DUT is placed facing the center of the target perpendicularly to the beam axis at a distance depending on the required radiation flux. Typically, the method consists in writing a predefined pattern in the memory locations and accessible registers of the processor via the instruction set (Load and Store). Once finished the initialization, the DUT is irradiated and the program checks periodically the registers and memory locations along the radiation test to detect upset events. If an upset is detected, the program writes the correct pattern in the associated memory location and logs the results to an external host via serial or Ethernet ports. During the static test all the sensitive zones are exposed to radiation at the same time which do not represents the real behavior of the circuit, since not all the memory resources are used simultaneously when an application is executed. For this reason, the static cross-section provides a worst-case estimation of the device sensitivity.

If the target device implements protection mechanisms such as ECC or parity that cannot be deactivated, this method is not suitable as it is. It can be explained since 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 cache invalidation mechanisms (Ramos, 2015). It is 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.

Test pattern and exposure time

The elementary data pattern for memory circuits is a logical checkerboard, alternating by address and bit. A physical checkerboard is also useful if full layout information of the device is available (Jesd89A, 2006). All zeros and all ones is also a common pattern used during radiation test. However, some memories such as DRAMs usually have a favorite error failure,

either 0 -> 1 or 1 ->0. For this reason, for testing when there is no *a priori* information about the component, the test pattern have to balance the number of 0's and 1's. For this reason, the selected pattern for the static test was *0x55AA55AA*.

Regarding the exposure time of the device, it is important to consider that the probability of having an upset event during a given period of time is a stochastic process that follows a Poisson distribution. Thus, the waiting time between the read operations in the static test can be validated by analyzing the distribution of the number of events per unit of time. If the obtained distribution does not follow a Poisson law, the waiting time should be adjusted.

4.4.3 Dynamic test

The dynamic test is an approach aiming at estimating the dynamic response to radiation of an application running on processor. The dynamic cross section (σ_{DYN}) evaluates only the device memory cells used by the application. It is computed by applying equation (2.1). During the dynamic test, the DUT is exposed to neutron radiation for inducing SEE. It is aligned to the target and placed at a distance depending on the neutron flux desired. The method consists in the periodical execution of an application implemented in the processor while the device is being irradiated. In order to detect errors, at the end of each execution the results are compared with a set of correct values previously obtained. The experiment is launched and monitored using a *host* computer located outside the armored chamber. The communication between the host and the DUT is achieved by means of serial, JTAG or Ethernet communication protocols. When an error is detected, it is logged and transmitted to the host which stores the results.

4.5 Neutron Radiation Facility

The radiation ground tests were conducted at the GENEPI2 facility located at the LPSC (Laboratoire de Physique Subatomique et Cosmologie) in Grenoble, France. 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 by impinging a deuteron beam onto a Tritium (T) target. After acceleration at 220 keV, deuterons (d) produce neutrons (n) by the fusion reaction $d + T \rightarrow n + {}^{4}He$ (Villa, 2014).

From the target, neutrons are emitted in all directions with an average energy of 14 MeV. The Device Under Test (DUT) is set facing directly the target at a distance determined to adjust the neutron flux. For the radiation test campaigns, it was considered, to first approximation, that only neutrons emitted fully forward will impact the DUT. While the DUT is fully exposed to neutrons, a dedicated neutron shielding can protect the readout electronic platform. Figure 4.3 illustrates the Genepi2 particle accelerator layout.

Neutron production is monitored throughout the experiments to determine the neutron dose for each irradiation. An online *Si* detector, located within the beam pipe about 60 cm upstream of the target, collects the recoil particles backscattered from the target during the fusion reaction. In addition, a movable monitor characterizes the neutron emission forward, after their conversion into protons and detection in a 3-stage *Si* telescope. Aluminum foils are irradiated periodically and characterized by the LPSC low-activity laboratory LBA with germanium detectors thus providing an independent crosscheck under reference conditions.

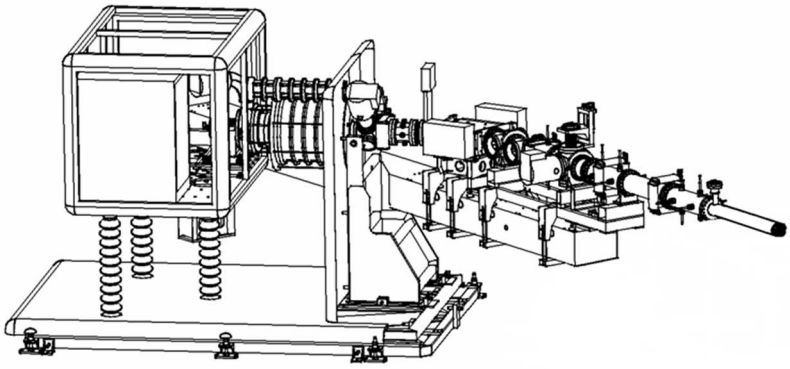

Figure 4.3: Genepi2 accelerator layout (Villa 2014)

Early 2015, a fresh T target was installed, generating a maximum neutron flux of $4.5\text{x}10^{7}(\text{n.cm}^{-2}.s^{-1})$. The main goals are to increase the neutron production and to improve the accelerator reliability. The major modification consists in replacing the current Deuterium ion source by a new one, based on the Electron Cyclotron Resonance (ECR) technique, delivering higher beam intensity. At the same time, a new monitor for the neutron production will be installed and commissioned. This will allow the precision on the dose measurement to be improved.

4.5.1 Radiation Facility Validation

GENEPI2 facility was validated through radiation ground testing with 15-MeV neutrons on a 90-nm CMOS SRAM memory (CYPRESS CY62167EV30LL). This memory was chosen since it was previously tested in other radiation facilities such as ASP, KVI and TRIUMF (Blackmore, 2003). In addition, this memory is well known thanks to reverse engineering and for being used in real-life SEU experimental platforms (Peronnard, 2009).

The SRAM was set facing the center of the target at a distance of 40 cm as shown in Figure 4.4. The repetition rate was restricted to 3000 Hz in order to limit the neutron flux. Under these conditions, the estimated neutron flux was around $3\,\text{x}10^{4}\ \text{n}.cm^{-2}.s^{-1}$ for obtaining a fluence of $1\ \text{x}10^{8}\ \text{n}.cm^{-2}$ within one hour of irradiation (Villa, 2014).

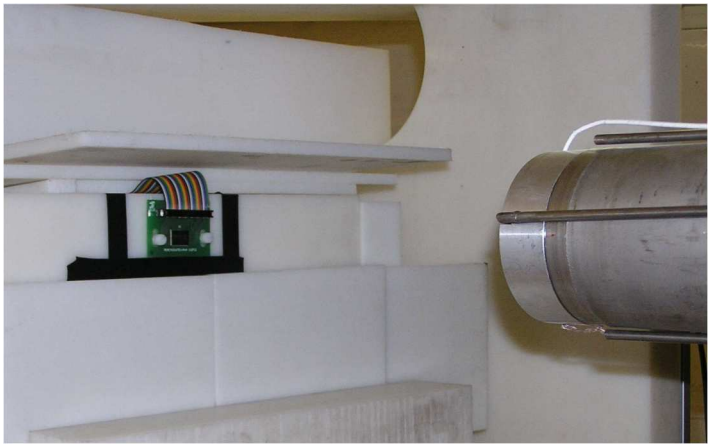

Figure 4.4: Genepi2 validation experiment (Villa 2014)

Detected SEUs were logged as long as the SRAM was exposed. Three different patterns were tested: all 1's, all 0's and alternated. During the radiation test, more than a hundred errors were identified including several MBUs and MCUs which is consistent with previous works (Hands, 2011), (Lambert, 2005). The resulting cross-section of the device was $1.2\ \mathrm{x}10^{-13}\ cm^2$/bit.

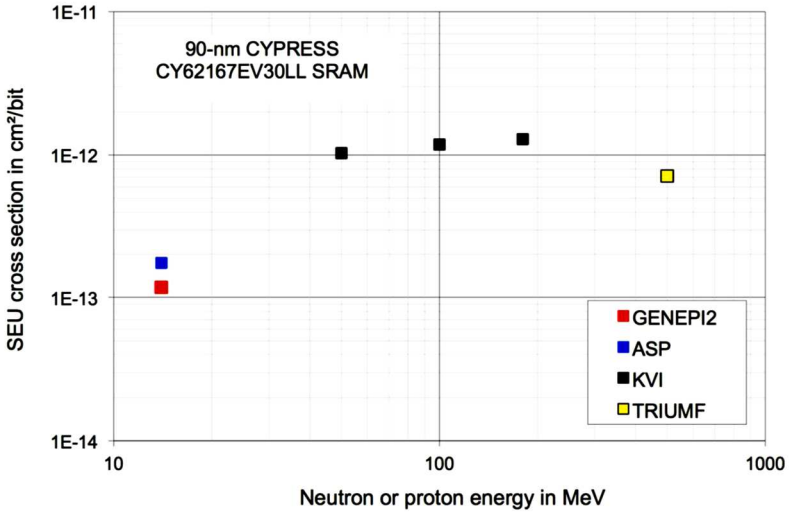

Figure 4.5: Neutron and proton SEU cross-section of 90-nm Cypress SRAM CY62167EV30LL (Villa 2014)

This result is shown in Figure 4.5 as well as the proton and neutron SEU cross-sections of 90-nm CYPRESS SRAMs issued from tests performed in other facilities. It can be seen a fairly good agreement between results from GENEPI2 and ASP. Radiation tests were performed on three different generations of SRAMs and no latch-up events were detected, thus validating the readout electronic shielding.

Other bulk CMOS SRAMs from another manufacturer, in 90 and 130nm technologies also were tested. Since the behavior of this family of devices under neutron radiation has been exhaustively studied and results agreed with those reported in literature (Hands, 2011) and (Hands, 2012), it is possible to conclude that the measurements are correct confirming the validity of the facility.

4.6 Conclusion

The methodology for studying the effects of the radiation on systems based on multi/many-core processors has been presented in this chapter. Two strategies for evaluating the SEE sensitivity of applications implemented on multi-core processors were considered. On one hand, an approach based on the CEU principles is proposed to predict the application error-rate. This approach uses fault-injection at application level and a physical evaluation of the device exposed to radiation. For that, the CEU fault-injection strategy has been extended to a multi/many-core processor benefiting of the multiplicity of cores. The details of the fault injection strategy using one core as fault injector as well as those related to the radiation experiments for obtaining the worst-case sensitivity of the device are presented. On the other hand, a physical evaluation of the device running the selected application is achieved by means of the dynamic radiation test. This strategy assesses directly the impact of SEE on the application.

The fact that these advanced architectures implement several levels of cache memories as well as internal shared memory increase significantly the sensitive area to be targeted. Thus, the aim of the present work is to validate the relevance of the proposed approach to predict the error-rate of an application implemented in the multi/many core processor combining the evaluation at device and application level.

The proposed approach provides the following advantages with respect to the CEU approach:

- Fault injection in processor's registers is accomplished using inter-core interrupts. In this way, the simulation time is significantly reduced.
- Fault injection in shared memories is performed directly by the fault-injector. In this case, it is a non-intrusive method.
- There is no restriction for target device. All architectures are suitable for the test.
- It is not further required the use of external platform (Thesic/Asterics) to trigger the fault injection, loading memory with corresponding data, compare the outputs to the expected results, and monitoring the execution time.
- It is not necessary to develop a daughter board containing the device to be tested.

Chapter 5 : Target platforms

This chapter presents the selected platforms to validate the proposed approach. At the beginning, the selection of the multi-processing mode and the hardware devices are discussed. Then, the development platform details and the main characteristics of the targeted device are described. Lastly, details about the benchmark application used for the evaluation are presented.

5.1 Programming model

The selected programming model for evaluating the proposed approach throughout this thesis is the *bare-metal* model. Despite the complexity of programming this model, it is preferred since it allows targeting more registers during the fault-injection campaigns. Moreover, the lack of OS allows the use of inter-core interrupts minimizing the injection time and the intrusiveness. Additionally, *bare-metal* provides the fault-injector the possibility to inject bit-flips directly in shared memory, in spite of the fact that in AMP mode each core has its own private memory space, since there is no *hypervisor* preventing this action. Consequently, in shared memory architectures the fault-injection is performed in an almost non-intrusive way, since the fault-injector reads and writes directly in the memory without interfering the execution of the other cores.

As it was explained in section 2.4.4, when the developer uses the *bare-metal* model, it has to program all the system functions: the startup of the cores, the task distribution among the cores, the synchronization between cores and/or tasks, the access to shared resources, the coherency consistency etc. To manage most of these functions, a *master* core is needed as well as a shared memory space for inter-processor communications. Therefore, a Master-Slave scheme was implemented. The *master* core is defined as the fault-injector core, whereas the slave cores execute in parallel the tested application.

In a general way, after a reset the *master* core is responsible for the booting operation and the configuration of the system. Then it starts up the other cores. A barrier was implemented to synchronize the initialization of the execution code in all the cores. In case of distributed algorithms, during the execution of the application the master distributes the tasks. Lastly, for

some devices where the processing cores do not have direct access to I/O, the master is also in charge of logging the results.

The main functions of the master are summarized in the following list:

- Booting the system.
- Starting up the other cores.
- Initializing global variables.
- Distributing the tasks.
- Synchronizing the communications by using inter-core communications.
- Injecting faults.
- Logging results (optional)

5.2 Device selection

The objective of this work is to provide a general approach for evaluating the SEE sensitivity and error-rate prediction of applications implemented in multi-core and many-cores. For accomplish this goal, it is required to target different devices aiming at representing the diversity of multi/many-core processors. The most relevant technological and architectural aspects to be considered for the selection of the device are:

- Manufacturing technology
- Memory architecture
- Number of cores
- Interconnections
- Memory protection mechanisms
- Performance
- Power consumption
- Reliability

The first selected platform is the Freescale P2041 RDB based on the QorIQ P2041 quad-core processor. This device was chosen due to:

- Implementation in 45nm SOI technology
- High performance
- High reliability: ECC and parity in its cache memories
- Based on the *PowerPC* architecture, validated in past works for aeronautics, single processor case (Peronnard, 2008).
- Freescale is one of the partners of the working group Multi-core for avionics (MCFA)

The second one is the Parallella computer which integrates the dual core ARM A9 processor used as host processor and the Epiphany E16G301 16-cores used as co-processor. The Epiphany multi-core was considered due to:

- High-performance
- Low power consumption
- Affordability, allowing the general public accessing to parallel computing.
- Co-processed architecture
- Interconnection (NoC)
- Open source
- The Parallella board was considered by the NASA for DragonEye UAS project

The third one uses the MPPA developer based on the Kalray MPPA-256 many-core processor which integrate 16 compute clusters having each one 17 VLIW cores. This many-core processor was selected due to:

- Implemented in CMOS 28nm technology
- High performance
- Huge number of cores
- Great configuration flexibility
- High reliability: ECC and interleaving in shared memories / parity in cache memories
- Exceptional power efficiency GFLOPS/W

For a better comprehension of the selected devices, Table 5.1 illustrates an overview of their main characteristics.

Table 5.1: Main features of the target devices

	P2041 multi-core	***Epiphany multi-core***	***MPPA-256 many-core***
Manufacturing technology	SOI 45 nm	CMOS 65nm	CMOS 28nm
Number of cores	4	16	256 + 32
Memory Hierarchy	L1-D, L1-I, L2 private L3 shared, DDR	SMEM DDR	L1 private SMEM, DDR
Core Interconnection	CCF	NoC	NoC
Protection mechanisms	ECC (L2/L3) Parity (L1)	None	ECC/Interleaving (Shared mem) Party (L1)
Power consumption	18 W	<2W	25W
Performance	15000 MIPS	32 Gflops	634 Gflops

Concerning the manufacturing technology, it is relevant to test a SOI device since it was demonstrated to be less sensitive to SEE than bulk CMOS (Gasiot, 2002). In addition, the degree of miniaturization is also an important issue to consider since the smaller the size of transistor channel, the more sensitive device. Although, these features have already been tested for devices such as SRAM memories and single processors, it is valuable to know their impact in the overall sensitivity of the multi/many-core processor.

From the memory architecture point of view, it is significant to compare the impact of SEE in a device implementing only cache memories, only shared memories or both of them. For the MPPA-256, two scenarios are considered: dynamic test cache enable and dynamic test cache disabled. It is done in order to know at what extent enabling the cache memories affect the reliability of the device.

Regarding the interconnections, the Epiphany implements a 2D mesh NoC for connecting 16 processor cores, while the MPPA-256 implements a 2D mesh NoC for connecting the 16 computer cluster. In contrast, the P2041 has a proprietary CoreNet coherency Fabric (CCF) to interconnect cores. An interesting issue would be to know the contribution of the interconnection to the error rate.

The number of cores of the device plays a preponderant role in its sensitivity, since the more cores, the more registers which typically are unprotected elements. However, having more cores may contribute to the reduction of the execution time when a parallel computing model is implemented. Less execution time leads to less exposure time to radiation of the application.

Memory protection mechanisms such as ECC and parity are essential for mitigating SEE. However, the way they are implemented in the memory hierarchy may impact its effectiveness. For that, radiation tests may provide a useful feedback of this implementation.

The power consumption is one of the most important issues in aerospace applications due to energy limitations. Devices with high power consumption are not good candidates for these applications. However, this limitation may be compensated with high performance and reliability.

Regarding the performance, sometimes it is difficult to compare the performance of two devices if it is not express in the same units. For example, for the MPPA-256 and the Epiphany it its expressed in GFLOPS (Giga floating point operations per second) while for the P2041 it is expressed in MIPS (millions instruction per second).

The energy efficiency, which is the relationship between performance and power consumption (GFLOPS/W), can provide a better overview of the device. Among the selected devices, the MPPA provides the best energy efficiency.

5.3 Platform description

The P2041 RDB, the Parallella computer, and the MPPA Developer platforms were used for evaluating the P2041 quad-core processor, Epiphany EG16301 multi-core processor, and MPPA-256 many-core processor respectively. The description of each platform as well as some details of each device, are presented in the following subsections.

5.3.1 P2041RDB

The Freescale P2041 RDB platform is a compact (micro-ATX) highly integrated reference design board featuring the quad-core P2041 device. Its maximum operating frequency is 1.5GHz and includes a rich input/output (I/O) mix. The board can serve as a reference for hardware development and its main applications are networking, control plane and mixed control plane in switches and routers, base station network interface, baseband cards, aerospace and defense, factory automation, etc. The P2014 RBD memory system supports 4 GB of DDR3 at 1333 MHz data rate. It has 128 MB of NOR flash, a 256 KB I^2C EEPROM and 16 MB of SPI memory. It also includes two USB 2.0 receptacles, two SATA ports and a SD card slot (Freescale, 2011). Figure 5.1 illustrates the components of the P2041RDB platform.

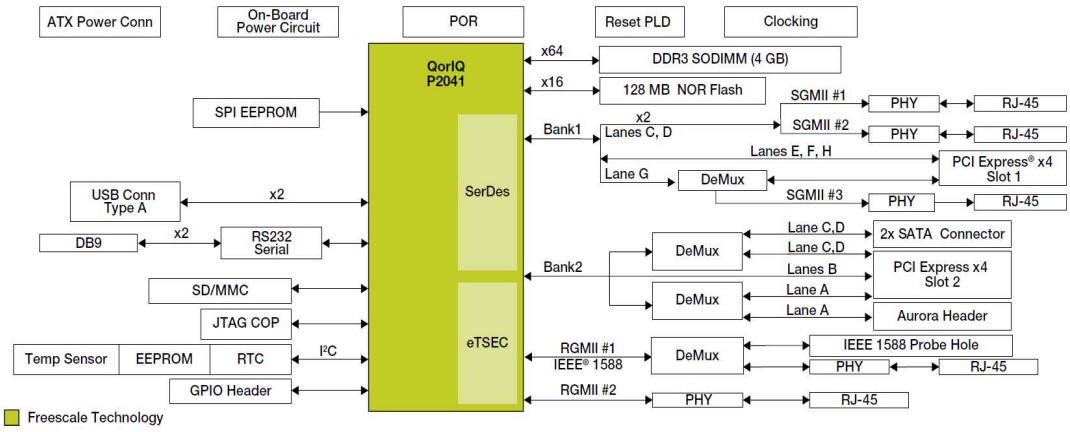

Figure 5.1: P2041 RDB platform (Freescale)

QorIQ P2041 processor

The QorIQ P2041communications processor is a multi-core processor based on four e500mc cores built in *Power Architectures* technology with high-performance data path acceleration architecture (DPAA), CoreNet fabric infrastructure, as well as network and peripheral bus interface required for networking. It is manufactured in 45nm SOI technology and includes a 10-Gigabit Ethernet Media Access controller (10GEC). This quad-core can operate up to 1.5 GHz and includes a three-level cache hierarchy. It has a 1 MB shared CoreNet Platform Cache (CPC) fronting the memory controller and a 64-bit DDR3 and DDR3L (low power) DRAM interface with 8-bit ECC and chip-select interleaving support. It also has five 1 Gbps Ethernet controllers, three PCI express 2.0 controllers running at up 5Gbps, four I^2C controllers, etc. Its SOI implementation makes this device immune to latch-up events. Figure 5.2 depicts the architecture of the P2041multi-core processor (Freescale, 2015a).

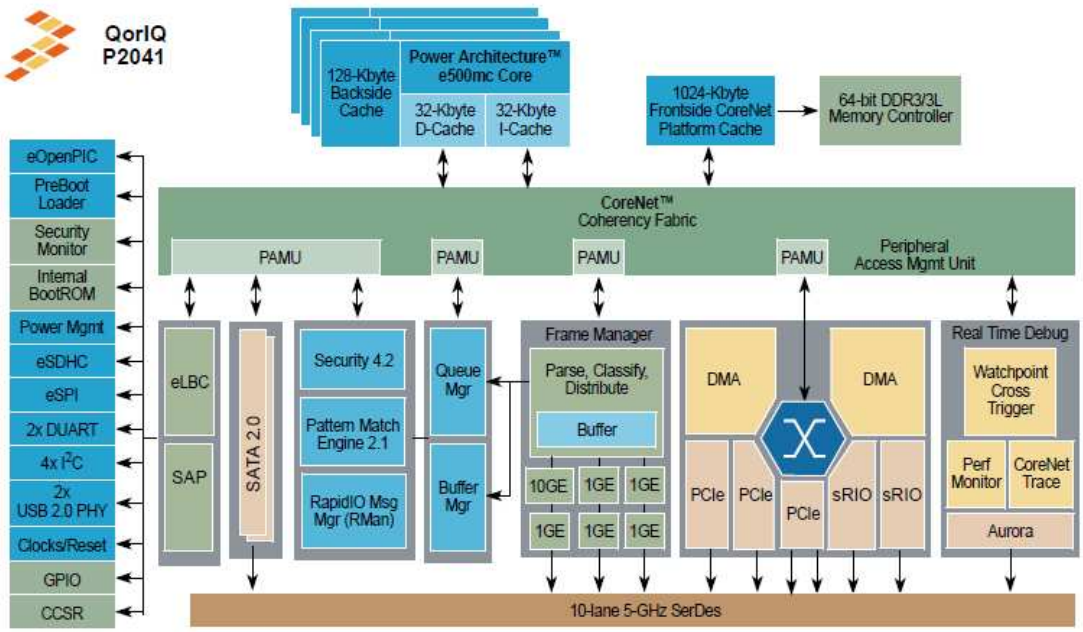

Figure 5.2: Architecture of the P2041multi-core processor (Freescale).

The e500mc core is a low power implementation of the resources for embedded processors defined by the Power ISA™. The core is a 32-bit implementation that implements 32 32-bit general purpose registers; however it support accesses to 36-bit physical addresses. The core is a superscalar processor that can issue two instructions and complete two instructions per clock cycle. The e500mc core implements independent on-chip 32 KB L1 caches for instruction and data with automatic cache invalidation when a parity error is detected, and a unified 128 KB backside L2 cache. L1 cache is protected with parity while L2 cache is 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. Figure 5.3 illustrates the L1 data cache organization (Freescale 2015b).

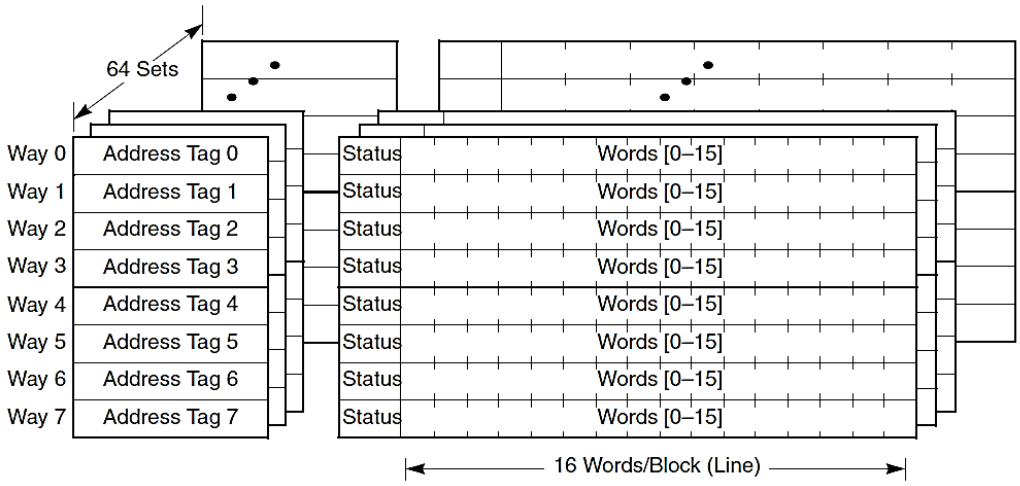

Figure 5.3: L1 data cache organization (Freescale)

The L1 instructions and data caches are organized as 64 sets of eight blocks with 64 bytes of data in each cache line. The data cache has 1 parity bit per byte and 1 parity bit per tag. Each block contains contiguous words from memory that are loaded from a 16-word boundary (that is, physical addresses bits 30 to 35 are zero). Cache blocks are also aligned on page boundaries. Physical address bits PA[24:29] provide the index to select a cache set. The tags consists of physical address bits PA[0:23].

5.3.2 Adapteva Parallella

The Parallella-16 board is a high performance computing platform credit card sized based on a dual core ARM A9 processor used as host and the Adapteva Epiphany 16-core used as co-processor for parallel computing. The board and the Epiphany chip are developed by Adapteva with the aim of providing an affordable super computer for speed up the transition from serial to parallel computing (Adapteva, 2009).

The central processor on the Parallella is the Zynq-7000 SoC that combines a dual-core CortexTM –A9 MPCoreTM processing system with Xilinx 28nm programmable logic. The Epiphany co-processor is the E16G301 device with 16 CPUs. The main memory is a 1GB 32-bit wide DDR3L SDRAM. In addition the board includes a flash memory of 128 Mb, a 10/100/1000 Ethernet port, 2 USB 2.0 connections, a Micro SD as a primary boot source and main Parallella storage medium, a serial port and a HDMI port. Figure 5.4 depicts the parallella architecture.

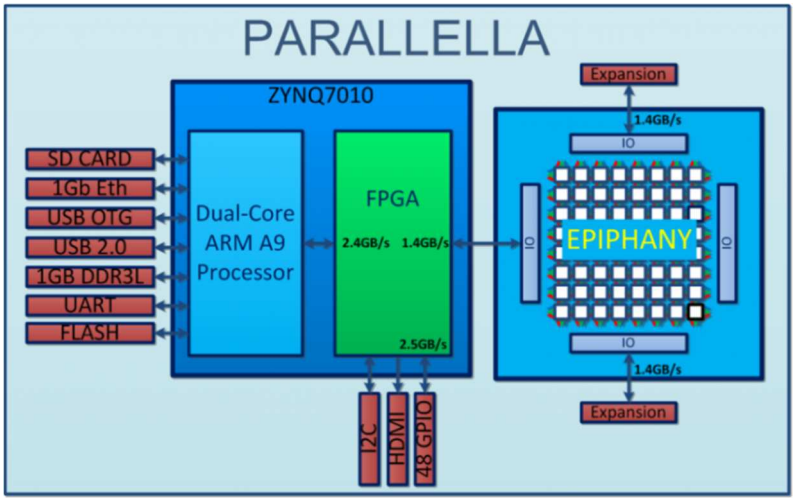

Figure 5.4: Parallella architecture (Adapteva)

Epiphany Multi-core processor description

The Epiphany is a scalable multi-core architecture with up to 4095 processors sharing a common 32 bits memory space. It defines a parallel computing fabric comprised of a 2D array of processors nodes connected by a low latency mesh network-on-chip. The E16G301 which is based on 3rd generation of the Epiphany multi-core IP, is a 16 core System-On-Chip implemented in a 65nm CMOS technology (Adapteva, 2011a). Each processor core is a 32-bit superscalar floating point RISC CPUs, capable of performing two floating point operations per clock cycle and one integer calculation per clock cycle. The device has a peak performance of 32 Gflops (2 Gflops per core). The maximum chip power consumption is less than 2 watt. Each CPU has an efficient general-purpose instruction set that excels at compute intensive applications while being efficiently programmable in C/C++. Figure 5.5 shows an implementation of the E16G301 architecture.

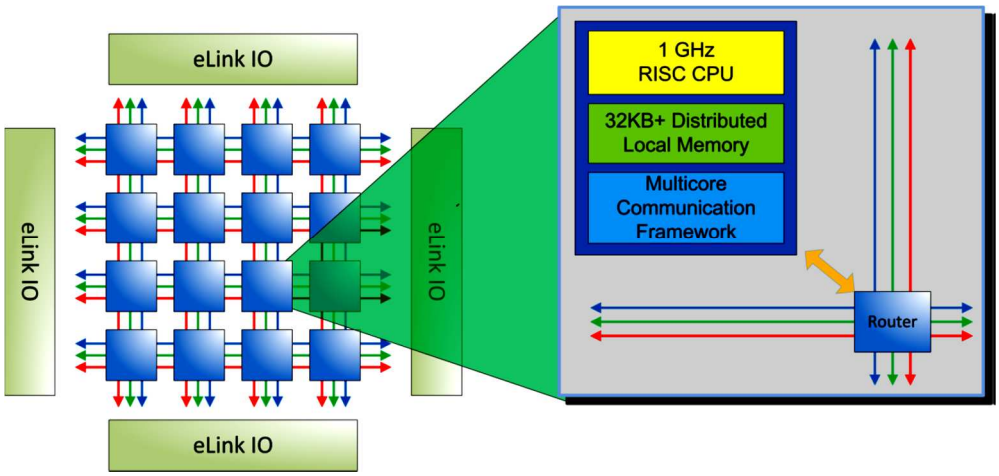

Figure 5.5: Implementation of the E16G301 Epiphany architecture (Adapteva)

The memory architecture of the Epiphany multi-core is based on a flat shared memory map. Each compute core has up to 1 MB of local memory as a unique addressable part of the total 32-bit address space. The core processor can access its own local memory as well as other processors' memory by means of standard load/store instructions. The local memory is comprised of 4 independent banks, each one of 8KB for a total of 32 KB for each CPU core as depicted in Figure 5.6. For the particular case of the Epiphany E16G301 that implements 16-cores, the chip has a 512 KB distributed shared memory (Adapteva, 2011b).

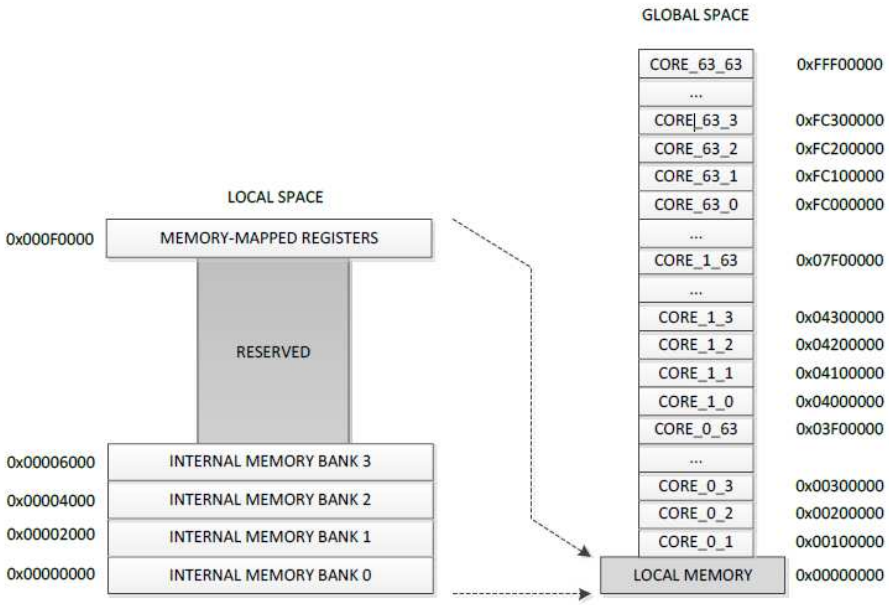

Figure 5.6: Epiphany global address map (Adapteva)

The Epiphany Network-On-Chip (eMesh) is a 2D mesh network for high speed inter-processor communication that connects the on-chip processor nodes. The mesh network efficiently handles all on-chip and off-chip communication in high throughput real-time applications. Each routing link can transfer up to 8 bytes of data on every clock cycle supporting an effective bandwidth of 64 GB/s at a mesh operating frequency of 1 Ghz. The network comprises three separated and orthogonal mesh structures: two networks for allocating on-chip and off-chip write traffic, and one network for all read traffic. Figure 5.7 shows an overview of the network-on-chip.

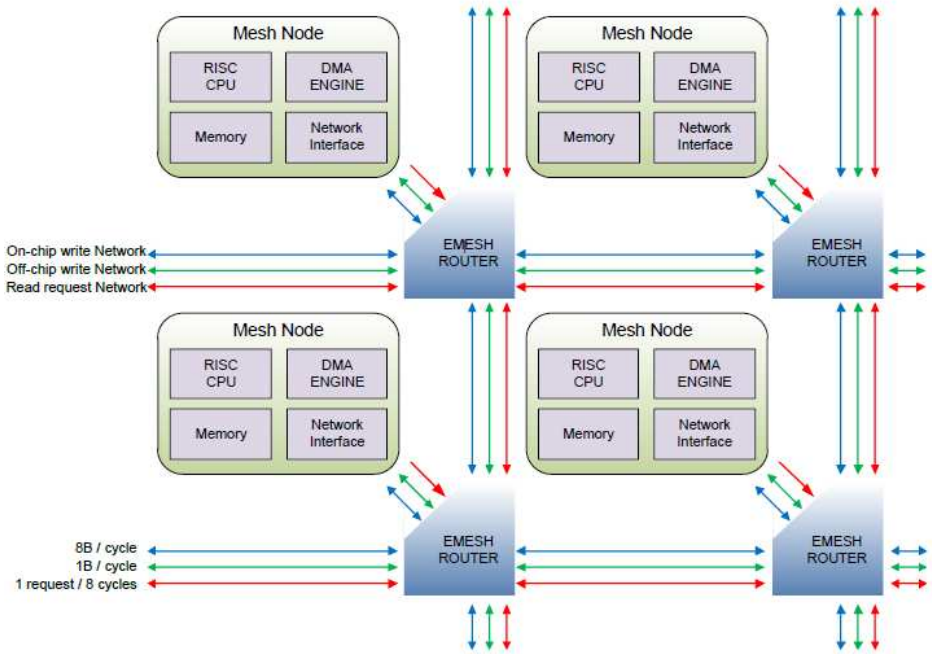

Figure 5.7: eMeshTM Network-On-Chip overview (Adapteva)

Anti-latch-up Circuit

Since the Epiphany multi-core is implemented in 65nm CMOS that does not provide a Latch-up protection, it was necessary to add in cascade to the board an anti-latch-up circuit for limiting the current supply to 1 A. The implemented circuit illustrated in Figure 5.8 works as follows: A pre-set potential coming from a voltage divider is applied to the positive terminal of comparator with the aim of determining the maximum output current of the circuit. The drop across the current sense resistor of 0.01 Ω is amplified by a gain of 10 using a differential amplifier, and connected to the negative of the comparator. Under normal working conditions, the current is within the permissible limits and the potential at the negative terminal is less than the potential at the positive one, then the comparator output is at high level which activates a relay through a transistor connecting the positive terminal of the power supply with the output of the anti-latch-up circuit.

As the desired output current must be limited to 1A, the maximum drop across the resistor is 0.01V. This voltage is amplified by a factor of 10 for obtaining 100mV at the input of the negative terminal of the comparator. Any excess of current will cause a potential drop greater than 100mV after amplification. For this reason, the voltage divider output should be set to 100mV. This is achieved by placing a combination of 1KΩ and 49KΩ resistors.

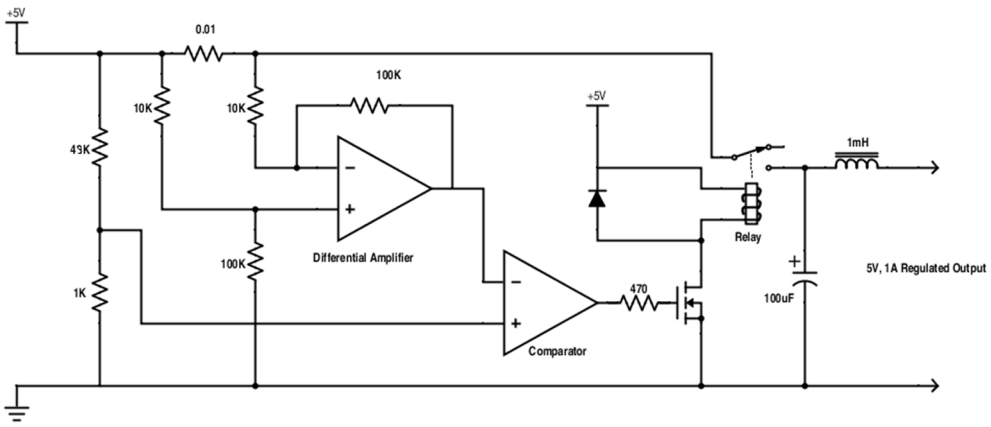

Figure 5.8: Anti-latch-up circuit

When the current drawn by the external circuit is higher than 1A, the comparator drives its output to low which switch off the transistor for deactivating the relay. Once the current peak falls down, the comparator will switch on the transistor again. The circuit thus acts like a switching regulator when the output requirement is more than 1A. To reduce the ripples, a parallel capacitor of 100uF and a series inductance of 10mH are connected at the output of the circuit.

5.3.3 MPPA Developer

The MPPA Developer is a development platform based on an Intel core I7 CPU operating at 3.6 GHz and running a Linux OS. The MPPA many-core is available within the Developer as an accelerator of the X86 Host CPU connected through 16 PCIe Gen3 lanes. In addition to the PCIe board and the MPPA-256 Processor, the platform includes a PCIe board for debug and probe.

The MPPA Developer is delivered with a user environment and configuration containing Linux CentOS 7 x86 64, Eclipse 4.3 and MPPA ACCESSCORE SDK v2.5 for developing, optimizing and evaluating applications. The latter includes three programming models for developing an application: POSIX, Kalray OpenCL and Lowlevel.

MPPA-256 Many-core processor description

The MPPA-256 (Multi-Purpose Processing Array) is a many-core processor manufactured in TSMC CMOS 28HP technology. It integrates 256 Processing Engine (PE) cores and 32 Resource Management (RM) cores, all based on the same VLIW 32-bit/64-bit architecture. The processor operates between 400 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 (Kalray, 2016).

The second version of this processor, called Bostan, is considered in this work. The global processor architecture is clustered with 16 compute clusters (CCs) and 2 input/output clusters (I/O) per device, where each cluster is built around a multi-banked local static memory (SMEM) of 2MB shared by the 16(PE) + 1(RM) cores in the case of the compute cluster, or by the 4(PE) + 4(RM) cores in the case of the I/O clusters. A wormhole switching network-on-chip (NoC) with 32 nodes and a 2D torus topology connects the compute clusters and the I/O clusters. An overview of the many-core processor is illustrated in Figure 5.9.

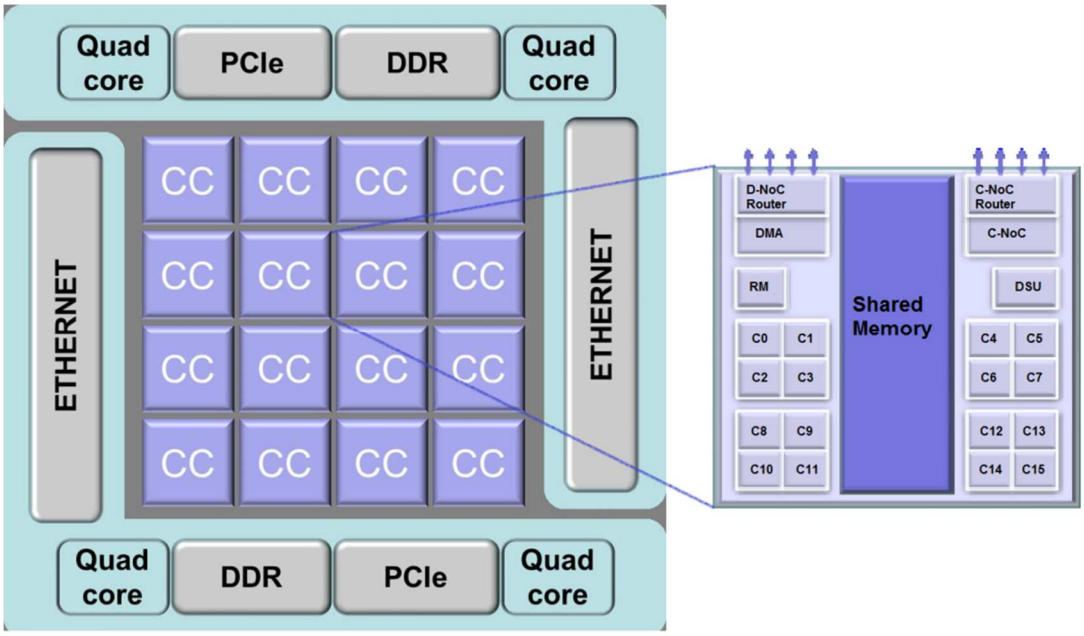

Figure 5.9: Many-core processor components (Kalray)

The SMEM is composed of 16 independent memory banks of 16384 64-bit words, for a total capacity of 2MB. Each memory bank is associated with a dedicated request arbiter that serves 20 bus masters: the D-NoC Rx (receive) interface, the D-NoC Tx (transmit) DMA engine, the DSU, the resource manager (RM), and 16 PE cores. Figure 5.10 illustrates the compute cluster buses.

The 16 memory banks are arranged in two sides of 8 banks, called left side and right side. The connections between the memory bus masters are replicated in order to provide independent access to the two sides. The private paths of the 16 PE cores are connected to the 16 memory bank arbiters. Other bus masters (D-NoC Rx, D-NoC Tx, DSU, RM) have their own private path also connected to the 16 memory bank arbiters (Kalray, 2016).

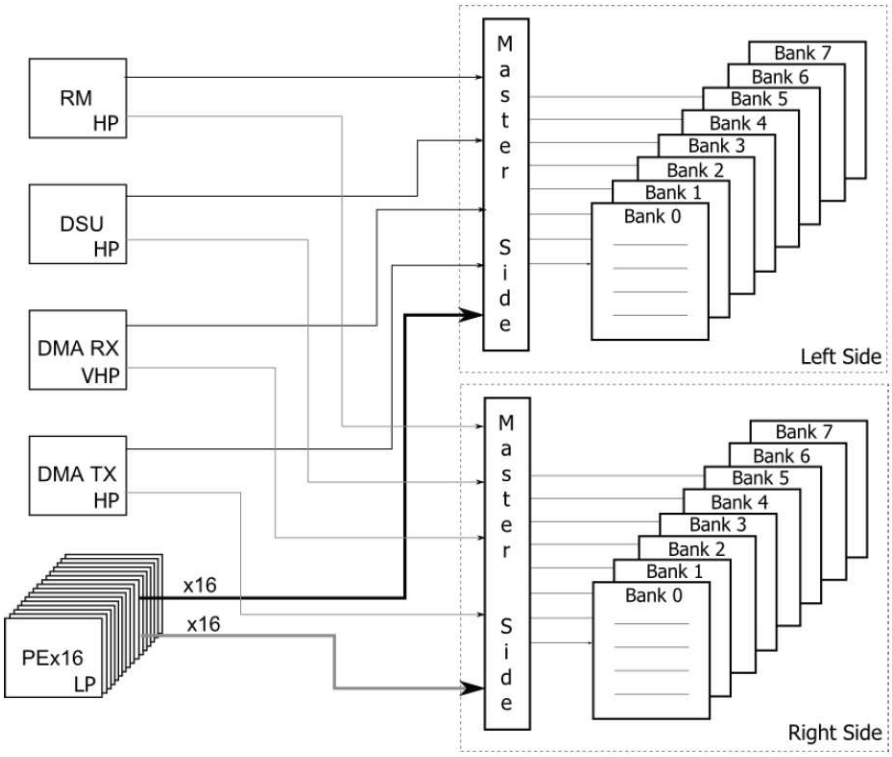

Figure 5.10: Compute cluster bus masters.

The main components of the many-core processor are covered by error protection mechanisms except the instruction 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. SECC errors 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 detected and signaled. The VLIW core implements separate instruction and data cache. There is no hardware cache coherency mechanism between cores nor between data and instruction cache. However, to enforce memory coherency, several software mechanisms are available to programmers.

The MPPA-256 many-core is embedded in a development platform containing the MPPA ACCESSCORE SDK version 2.5 for developing, optimizing and evaluating applications. The platform is based on an Intel core I7 CPU operating at 3.6 GHz and running a Linux OS. It includes a PCIe board MPPA-256 Bostan version and a PCIe board for debug and probe (Kalray, 2016). This board implements a module for controlling current and voltage aiming at mitigating latch-up events.

5.4 Benchmark application

A standard *n x n* matrix multiplication (MM), which is a memory-bound application, was selected to be tested throughout this thesis. It was considered since the matrix multiplication is one of the most essential algorithms in numerical algebra as well as in distributed, scientific and high-performance computing (Ballard, 2012). Concerning avionic applications, MM is used for image processing, filtering, adaptive control, and navigation and tracking. For the evaluation of the application, two scenarios are proposed.

In the first scenario, the sequential algorithm of the matrix multiplication was used. This scenario was implemented in the P2041 and the Epiphany multi-cores using one core of the device as the *master* core. 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 n of the matrix was selected depending on the memory capacity of each device in order to fill as much as possible the cache and shared memories, and 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. All the elements of the matrix A were filled up with the same value a. Similarly matrix B was filled up with b, thus the expected result was a x b x n 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 radiation experiments are totally independent of the input values, no matter the particle produces a bit flip in a fixed or random value.

For the second scenario, each compute cluster of the MPPA-256 many-core executes independently a parallel algorithm of the matrix multiplication. The source code is an assembler optimized version of a collaborative 256×256 matrix multiplication, distributed among the 16 processing elements (PE). Inside each cluster, the resource manager (RM) core is the *master* of the system. The computation run repeatedly to guarantee that each cluster computes enough time so that all the clusters work in parallel. A, B and C are single precision floating-point matrices. The size of the matrix was chosen so that data and code remain in the local SMEM memory.

5.5 Concluding remarks

In this chapter, the system configuration for the evaluation of the proposed approach was detailed. All the targeted devices were configured in AMP mode to isolate the fault-injector device. In addition, a memory-bound application was chosen to be implemented in

bare-metal mode. This application was selected since it allows using extensively the memory resources.

Three different devices grouping representative technological and architectural aspects of multi/many-core processor were selected to validate the approach. Details about the development platforms as well as the processors were described. Furthermore, the design of an anti-latchup circuit for protecting the Parallella board was also presented. This design is not specific for this board, so it can be used for protecting any circuit.

The evaluation of the selected platforms based on the Freescale P2041 quad-core processor, Epiphany EG16301 multi-core processor, and Kalray MPPA-256 many-core processor through accelerated ground testing and fault injection is presented in the next chapter.

Chapter 6 : Experimental results and evaluation

This chapter presents the evaluation of the P2041 quad-core processor, Epiphany multi-core processor, and MPPA-256 many-core processor. This is accomplished by means of fault-injection campaigns and radiation ground testing with 14 MeV neutrons. Radiations campaigns were performed to obtain the static cross-section of the device as well as the dynamic response. The results obtained during fault injection and static radiation experiments allow predicting the error rate of an application implemented in the concerning device. The dynamic cross-section of the device is used for validating the prediction approach and for assessing the reliability of the device executing an application while it is exposed to radiation.

6.1 Evaluation of the Freescale P2041 multi-core processor

For evaluating the dynamic response of the P2041 quad-core, the sequential algorithm of the *80 x 80* matrix multiplication was implemented on each core. The input matrix A was filled up with 1's and matrix B was filled up with 2's, thus the expected result was *160* for all the elements of matrix C.

6.1.1 Neutron radiation campaigns

Due to the P2041 multi-core is implemented in SOI technology, it is no necessary to add an anti-latch-up circuit. Since the cache memories of the target device implement protection mechanisms that cannot be deactivated, for evaluating this device a complementary technique based on *machine-check error report* was used. This allows logging data that have been corrupted and corrected during the radiation experiments.

For the experiments, the cores were configured in *write shadow mode* where all modified data in the L1 cache is written through into the L2 cache. This ensures that, if data or parity tags are corrupted in the L1 cache, it can be invalidated and repopulated with the valid data from the rest of the memory hierarchy (Freescale, 2015b). In addition, the L1, L2 and L3 caches, as well as the *machine-check error report* of each core were enabled. For logging all the SEEs occurred during the radiation experiments, the *machine-check error interrupt* and the *Cache Error Checking* bits were also enabled. Errors detected by the application and by the *machine-check-error report* were considered in order to evaluate the sensitivity of the 45nm

SOI technology of the target device. Table 6.1 shows the sensitive zones of the multicore-processor targeted in the radiation experiments.

Table 6.1: Targeted sensitive zones of the P2041 multi-core processor

Sensitive Zone	***Location***	***Capacity***		***Description***
L1	Core 0,1,2,3	32 KB/ D per core	32 KB / I per core	Data/Instruction Cache
L2	Core 0,1,2,3	128 KB per core		Backside Unified Cache
L3	Multi-core	1024 KB per chip		Frontside Cache
GPR	Core 0,1,2,3	32 registers of 32 bits		General Purpose Register
FPR	Core 0,1,2,3	32 registers of 64 bits		Floating Point Register

Experimental Setup

Static and dynamic radiation experiments were performed on the P2041 RDB development board which integrates the studied multi-core processor. The device under test was placed facing the center of the target perpendicularly to the beam axis at a distance of 19.1 ± 0.5 cm. The neutron beam energy was 14 MeV with an estimated flux of $1.96\,\text{x}10^5\ \text{n}.cm^{-2}.s^{-1}$ at 500 Hz frequency with an error of $0.1\,\text{x}10^5\ \text{n}.cm^{-2}.s^{-1}$. For protecting the rest of the platform, a 5 cm thickness polypropylene block was used.

Static Sensitivity

The first radiation campaign was carried out for obtaining the static cross-section of the device. 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. In this campaign were observed 58 SEEs within 2 hours of exposure time. Among them, 46 were SEUs and 12 Single Event Functional Interrupts (SEFIs). There were no detected errors in GPRs and FPRs. Latch-up events were not present due to the immunity of SOI technology process. Table 6.2 summarizes the results of this campaign.

Table 6.2: Results of the static radiation campaign

SEE Type	*Type of error*	*Occurrences*	*Consequences*
SEU	L1 Data cache parity	9	None
SEU	L2 Single-bit ECC	29	None
SEFI	L2 Tag parity	5	Hang
SEU	L2 Multiple-bit Tag Parity	1	None
SEU	L3 Single-bit ECC	7	None
SEFI	L3 Multiple-bit ECC	6	Hang
SEFI	Other errors	1	Hang
Total		58	

For obtaining the static cross-section of the SOI technology ($\sigma_{\mathrm{STATIC\_SOI}}$) the equation (2.1) was applied. This experiment considers as errors all the observed SEEs no matter they were detected by the machine-check error report, or by the self-testing application.

$$Fluence = Flux * exposure_{time} = 1.96\ \mathrm{x}10^{5}\ \mathrm{n}.cm^{-2}.s^{-1} * 7200\ \mathrm{s}$$

$$\sigma_{\mathrm{STATIC\_SOI}} = \frac{\text{Number of upsets}}{\text{Fluence}} = \frac{58}{1.41\mathrm{x}10^{9}} = 4.11\mathrm{x}10^{-8}\ \frac{cm^2}{device}$$

Due to the scarcity of experimental data (58 SEEs), it is compulsory to add uncertainty margins to these results. For a 95% confidence interval ($\alpha = 0.05$), the lower and upper limits for the dynamic cross-section are:

$$3.12\mathrm{x}10^{-8}\frac{\mathrm{cm}^2}{\mathrm{dev}} < \sigma_{\mathrm{STATIC\_SOI}} < 5.32\mathrm{x}10^{-8}\frac{\mathrm{cm}^2}{\mathrm{dev}}$$

The targeted registers and memory cells of the multicore processor expressed in bits represent the sum of: (1 x L3[1MB] + 4 x L2[128KB] + 4 x L1_data[32KB] + 4 x L1_inst[32KB] + 32 x GPR[32 bit] + 32 x FPR[64 bits]) resulting $1.47\ \mathrm{x}10^{7}$bits. Then, the confidence interval for the static cross-section per bit is estimated as $[2.12 - 3.62]\ \mathrm{x}10^{-15}\ cm^2/bit$. Reference (Stolt, 2012) provides the estimation of the bit cross-section of a 45nm CMOS technology processor ($1\mathrm{x}10^{-14}\ cm^2/bit$) 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.

On the other side, the static cross section of the device (σ_{STATIC}) is calculated only based on the errors produced during the static test. In this case, there were 12 SEFI that caused hangs. Then, the σ_{STATIC} of the device under test is:

$$\sigma_{\mathrm{STATIC}} = \frac{12}{1.41\mathrm{x}10^{9}} = 8.51\mathrm{x}10^{-9}\ \frac{cm^2}{device}$$

which expressed with a 95% confidence interval is:

$$4.40\text{x}10^{-9}\frac{\text{cm}^2}{\text{dev}} < \sigma_{\text{STATIC}} < 14.9\text{x}10^{-9}\frac{\text{cm}^2}{\text{dev}}$$

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 4.6 as "Other errors") that produced a system hang simultaneously in all the cores. This event leads to several errors logged by the self-testing application running on the processors that showed data different from the original word (0x55AA55AA) written in the memory. From these errors, two types of patterns were identified. The patterns are summarized in Table 6.3 and Table 6.4.

Table 6.3: Main memory space and details of the first pattern of errors occurred during the static test

Core	***Start Address***	***End Address***	***Range 1***	***Range 2***	***Range 3***
0	0x10000	0x30000	0x16548 - 0x1657c	0x14cc8 - 0x14cfc	0x16ac8 - 0x16afc
1	0x40000	0x60000	0x46048 - 0x4607c	0x463c8 - 0x463fc	0x46908 - 0x4693c
2	0x70000	0x90000	0x76048 - 0x7607c	0x76388 - 0x763bc	0x76908 - 0x7693c
3	0x100000	0x120000	0x106048 - 0x10607c	0x1063c8 - x1063fc	0x106908 - 0x10693c

Table 6.3 presents the main memory space used by each core (columns 2 and 3) and the address ranges where the first pattern was replicated. It consists of a set of fourteen words with consecutive addresses containing *0xDEADBEEF* as data. The second pattern constitutes scattered clusters of errors of four words each. In each of them, the first word contained *0xDEADBEEF*, the second one *0x20200044*, the third one *0x00130000* and the last one *0x00006000*. Table 6.4 summarizes the replications of this pattern, as well as the involved addresses.

Table 6.4: Details of the second pattern of errors occurred during the static test

Core	***No. Ocurrence***	***1st Word Address***	***2nd Word Address***	***3rd Word Address***	***4th Word Address***
0	1	0x10000	0x10004	0x10008	0x10024
	2	0x16000	0x16004	0x16008	0x16024
	3	0x16200	0x16204	0x16208	0x16224
	4	0x16600	0x16604	0x16608	0x16624
1	5	0x42000	0x42004	0x42008	0x42024
	6	0x46380	0x46384	0x46388	0x463a4
	7	0x46440	0x46444	0x46448	0x46464
	8	0x72000	0x72004	0x72008	0x72024

Core	*No. Ocurrence*	*1st Word Address*	*2nd Word Address*	*3rd Word Address*	*4th Word Address*
2	9	0x763c0	0x763c4	0x763c8	0x763e4
	10	0x76440	0x76444	0x76448	0x76464
	11	0x7c000	0x7c004	0x7c008	0x7c024
3	12	0x102000	0x102004	0x102008	0x102024
	13	0x106380	0x106384	0x106388	0x1063a4
	14	0x106440	0x106444	0x106448	0x106464

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 processor.

Dynamic Response

A second radiation test campaign was carried out with the device operating in AMP mode without OS to obtain the dynamic cross section. Two tests, each one lasting 2 hours were performed. Table 6.5 shows detected errors in L1, L2 and L3 cache memories. The Load Instruction and Instruction Fetch errors are the most critical ones since they produced processor hangs. Half of the observed L2 Tag parity errors lead to processor hangs. L1 Data cache parity errors are not critical since L1 cache is invalidated when parity fails.

Table 6.5: Results of the dynamic radiation test of the P2041 multi-core

SEE Type	*Type of error*	*Test 1*	*Test 2*	*Consequences*
SEFI	Load Instruction	1	0	Hang
SEU	L1 Data parity	19	17	None
SEU	L2 Single-bit ECC	9	20	None
SEFI	L2 Tag parity	0	4	Hang
SEU		3	1	None
SEU	Multiple L2 errors	3	1	None
SEU	L3 Single-bit ECC	3	2	None
SEFI	Instruction Fetch	0	1	Hang
MBU	Other errors	6	0	Erroneous result
Total		44	46	

L2 and L3 Single-bit errors are not critical as the ECC corrects them. Summarizing, there were one SEFI in test 1 and five SEFIs in test 2 that caused system hangs. In addition, six

events in test 1 caused 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 allowed identifying 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 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.
- 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 can 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. 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 6.1 (a), any line tag comprised in the interval (*0x403D6* - *0x403DC*) (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 *b1* and *b2* due to their physical adjacency. For the three cases the tags had to be changed (from *0x403DB* to *0x403DD* and from *0x403D8* to *0x403DE*). 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 *0x1A*, *0x1E* and *0x20* 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. Figure 6.1 depicts

the clusters of errors observed in Core 2 assuming that the particle affected the L1 cache. Two of them had line tag *0x403DD* and the other one *0x403DE*. Note that in Figure 6.1 (a), the main memory is illustrated by blocks of 64 Bytes. One memory address (36 bits) comprises cache tag (24 bits), set (6 bits) and word position (6 bits). Also note that matrices A, B and C do not begin exactly at the initial part of the address line that the cache refers to.

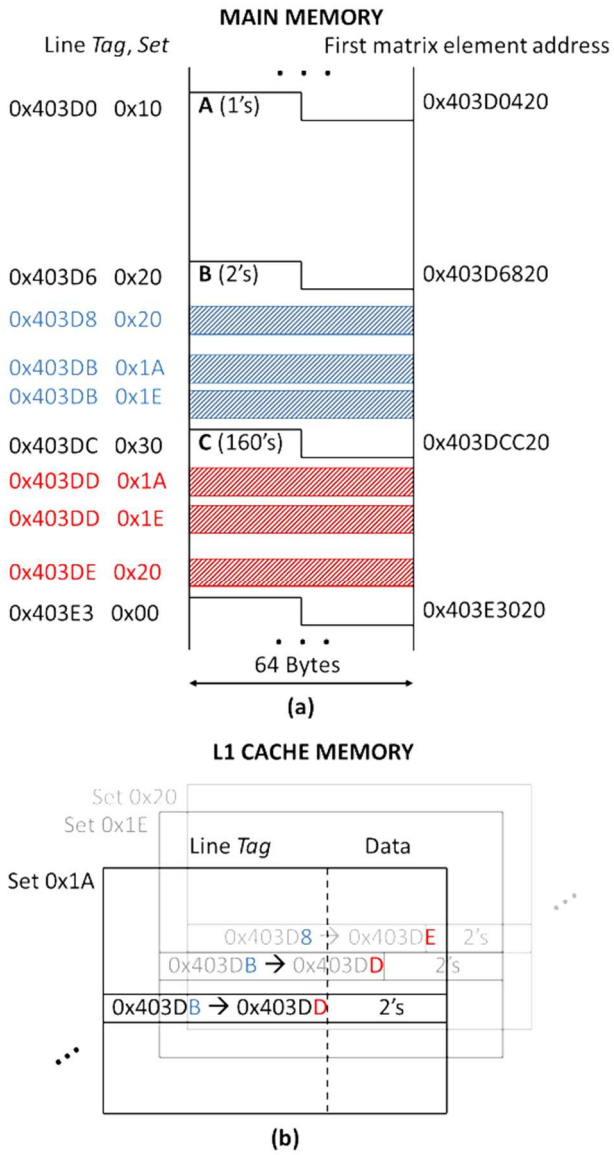

Figure 6.1: Clusters of errors caused by undetected tag errors.

Finally, the cluster of errors observed in Core 1 appears in the line tag *0x203DD* set *0x1B*. Following the previous analysis it is possible to verify that the particle has also changed the bits *b1* and *b2* of the line tag *0x203DB* set *0x1B* becoming the line tag *0x203DD*. 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 (Ramos, 2016).

2) Single Errors: Two separated matrix-result data were corrupted from "160" to "162" in Core 2, at addresses *0x403DF380* and *0x403DF480* respectively. Since the same bit *b1* was corrupted in both addresses and the difference between them is *0x100*, 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.

On the other side, the errors obtained during tests 1 and 2 described in Table 4.9 were added in order to have the total number of errors occurred within 4 hours of irradiation with a fluence of $2.82x10^9 n.cm^{-2}$. The total number of detected SEEs was 90, and among them 12 produced erroneous results and hangs. Then, applying equation (2.1) the dynamic cross-section is:

$$\sigma_{\mathrm{DYN}} = \frac{12}{2.82\mathrm{x}10^9} = 4.25\mathrm{x}10^{-9}\ \frac{cm^2}{device}$$

Applying equation (4.7) for confidence level of 0.95, the confidence interval of the dynamic cross-section in AMP scenario without OS is:

$$2.17\mathrm{x}10^{-9}\frac{\mathrm{cm}^2}{\mathrm{dev}} < \sigma_{\mathrm{DYNAMIC}} < 7.33\mathrm{x}10^{-9}\frac{\mathrm{cm}^2}{\mathrm{dev}}$$

6.1.2 Fault injection campaigns

Regarding fault-injection campaigns, the standard matrix multiplication was independently implemented on three of the four cores (1-3) while core 0 performs as the monitor and fault-injector of the application. Two types of fault-injection were performed, the first one targets program variables to emulate errors in cache memories, and the second one targets processor's registers to simulate faults in register memory cells.

Fault injection in program variables

Two fault-injection campaigns were performed. The first one aims at simulating SEUs in L1 Data cache memory by targeting program variables. The second one simulates SEUs in L2 cache memory by also targeting program variables.

For the first case a matrix sized of 40x40 was used, while for the second case a 80x80. In both cases, the matrices A, B and C were located in consecutive memory arrays. Additionally to these variables, the algorithm uses some internal variables to manage the loops and to store some intermediate results. Each variable was implemented in 32 bits. Table 6.6 summarizes the variables involved in the fault injection campaign.

Table 6.6: Variables details for both scenarios

Matrix size n	*Input variables*	*Output variables*	*Loop variables*	*Sensitive area*
40	3200	1600	3	~ 19 KB
80	12800	6800	3	~75KB

Fault injection in program variables 40x40 MM

The first fault-injection campaign considers the injection of one SEU per execution in program variables. The consequences of the injected SEU are classified as silent fault, erroneous result, exception and timeout.

In this experiment, SEU faults were injected at instants selected within the nominal duration of the executed program which was 95605 clock cycles. Figure 6.2 shows a general overview of the first test campaign where 43104 faults were injected.

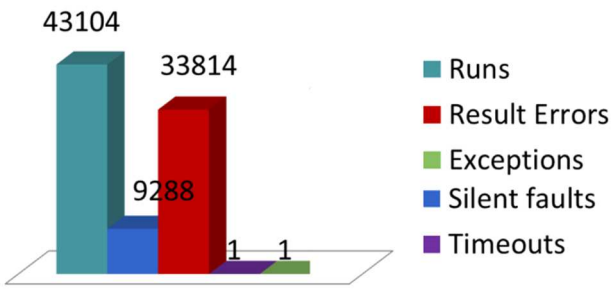

Figure 6.2: Results of the first fault-injection campaign

From the presented results, it can be calculated the application error-rate applying the equation (4.1) and considering as errors the result errors, time-outs and hangs.

$$\tau_{\mathrm{INJ}} = \frac{\text{Number of Errors}}{\text{Number of Fault Inj.}} = \frac{33816}{43104} = 78.45\text{x}10^{-2}$$

As shown in figure 6.2, the standard matrix multiplication algorithm is very sensitive to SEUs targeting the variables of the program, having an error rate about 78%. Very few time-outs and exceptions, caused by a fault injected into one of the loop variables, were detected during this campaign. It is explained since loop variables only represent 0.06% of the sensitive zone. Table 6.7 shows a distribution of errors among the variables for the first test campaign.

Table 6.7: Distribution of errors depending on the location of the targeted variable

Targeted Variable	***Runs***	***Silent fault***	***Result error***	***Time-out***	***Exception***
INPUT	28705	630	28075	0	0
OUTPUT	14370	8659	5711	0	0
INDEX	29	1	26	1	1

Table 6.7 identifies the critical zones of the matrix multiplication algorithm. Results show that fault injection in the input matrices have produced 83% of the errors, while faults injected in the output matrix produced only 17% of the errors. Note that the input area doubles in size the output area but the errors generated in the input area are almost 5 times the errors generated in the output area. Then, the most sensitive area of this application is the input area. This can be explained since input matrices have more exposure time than the output matrix because of math operations. The least sensitive area is the one corresponding to the internal variables since it has only three loop variables. However, in this zone were produced the time-outs and exceptions that can dramatically affect the sequence of the program.

Fault injection in program variables 80x80 MM

A second fault-injection campaign in program variables was performed increasing the matrix size to 80x80 which gives 75 KB of data. This was done in order to surpass the dimension of L1 data cache that is 32 KB. In this way, the use of L2 cache is guaranteed. It is important to note that the size of the matrix was selected in order to maintain a trade-off between the amount of memory used and the execution time. Table 6.8 shows an overview of the second test campaign where 99067 faults were injected with a nominal execution time of 758381 clock cycles.

Table 6.8: Results of the second fault injection campaign in program variables

SEUs per run	*Runs*	*Silent faults*	*Erroneous results*	*Timeouts*	*Exceptions*	*Error Rate (%)*
1	99069	34410	64657	1	1	65.27

As shown in Table 6.8, when the size of the matrix is increased, the application error-rate, calculated applying the equation (4.1) decreases. It shows that, even if the code of the application is the same, varying parameters as the matrix size may have a significant impact in the application's sensitivity. An extra experiment, out of the scope of this thesis, was performed running the same application under Linux operating system (Vargas, 2015). In this case, the fault injection campaign targeted not only program variables but also shared libraries and the operating system, providing an error rate about 40%. This result also shows that varying the configuration of the experiments may change significantly the error rate of the application. Figure 6.3 shows the total number of errors per execution versus the execution time.

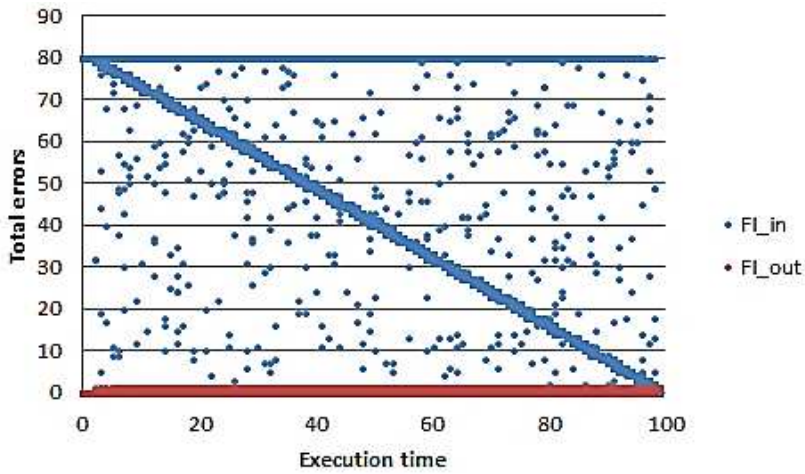

Figure 6.3: Number of errors vs execution time

In Figure 6.3, it can be seen that the total number of propagated errors in the application decreases linearly with the execution time. This happens since at the beginning of the execution the total area of the matrices is exposed. This area reduces as the indexes of the *for loops* of the algorithm increase until the end of the execution. The errors in blue come from fault injection in the input matrices, while errors in red come from fault injection in output matrix.

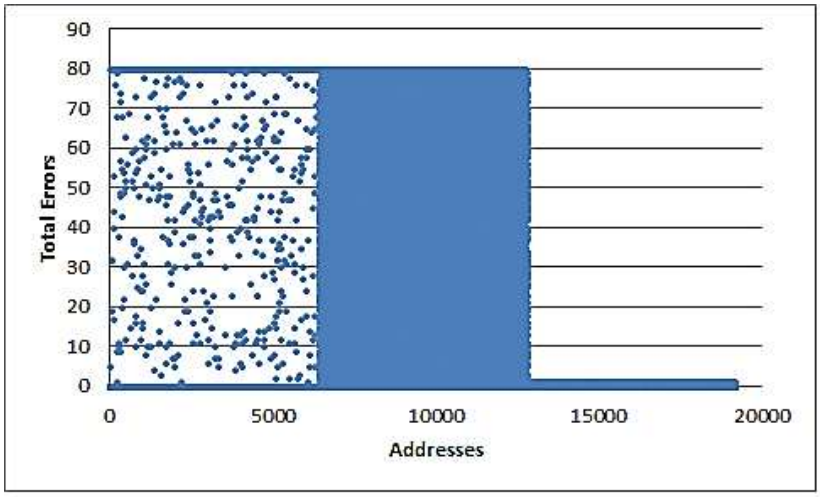

Figure 6.4: Number of errors vs affected address

Figure 6.4 shows the total number of errors per execution versus the addresses of the matrices. In this figure it is possible to see that the major density of errors belongs to the range of addresses of the second matrix. It can be explained since the algorithm uses nested *for* loops, where the index of the second loop is the one that repeats the size of the matrix times the number of the index of the first loop, which results in more exposure time of the second matrix.

The aberrant values observed in both figures are produced by faults injected in matrix A. In the case of figure 6.3, aberrant values also tend to form a diagonal as the main diagonal produced by faults injected in matrix B, but they are much more scarce since the exposure time of matrix A is much more shorter than the one of matrix B. In the same way, in figure 6.4, the errors produced by faults injected in memory addresses corresponding to the matrix A are scarce due to the short exposure time of the elements of matrix A.

The results presented in both figures evidence that fault injection is a very useful strategy for analyzing the behavior of an application in presence of SEU type events. In this way, programmers may have the possibility to modify the programing paradigm with the aim of reducing the impact of SEUs in the results of the application.

Fault injection in processor registers

The third fault injection campaign target 32 General Purpose Registers (GPRs) and 5 Special Purpose Registers (SPRs) which are accessible by software means. The registers are 32-bits sized, and thus the target sensitive area is about 1Kbits. For this campaign the number of injected faults was considerably lower than the fault injection in variables due to the criticality of certain registers such as SRR0 (program counter), SRR1 (machine state register)

in which faults may result in synchronization loss or exceptions. Table 6.9 shows a general overview of the second test campaign where 500 faults were injected.

Table 6.9: Results of fault injection in registers

Injected faults	***Silent faults***	***Erroneous results***	***Timeouts***	***Exceptions***	***SER (%)***
500	437	26	24	13	12.6

Results in table 6.9 show that the error-rate is about 12,6%. However, manipulating certain processor registers become critical because it causes several timeouts and exceptions. Hence, the performance of the whole system is dramatically affected. Nevertheless, the occurrence of a fault in a processor register has a very low probability since its physical area corresponds to 0.6% of the area occupied by program variables and data. Table 6.10 presents the error incidence classified by type of register.

Table 6.10: Detected errors distributed by registers

Targeted Registers	***Runs***	***Silent faults***	***Erroneous Results***	***Timeouts***	***Exceptions***	***SER(%)***
GPR 0 to 7; GPR9 , GPR 10, GPR12	145	128	17	0	0	12
GPR 8	19	12	4	0	3	21
GPR 11	16	9	4	0	3	25
GPR 13 to 31	235	235	0	0	0	0
SRR1	26	18	0	8	0	31
SRR0	27	3	1	16	7	89
LC/CRF/ XER	32	32	0	0	0	0

Table 6.10 shows that the most critical register is SRR0. This register saves the context of the program before the interruption subroutine. 59% of injected faults in the SRR0 register lead to a timeout and 26% produce exceptions. This can be explained due to the fact that the program counter contains the next instruction to be executed after the interruption, and perturbing its content will cause a sequence loss.

Another critical register is SRR1. It stores the content of the machine-state-register when an interruption occurs. A fault injection in this register may produce a time-out when processor state is performing an address translation for instruction and data memory access. On the other hand, the experiment proves that special registers LC (Load Context), CR (Condition Register) and XER (integer exception register) have no incidence in the program execution.

Regarding GPRs, they present different types of results depending on how often they are being used by the compiler in this application. In this case, it can be seen that GPR 8 and GPR 11 are frequently used to store data during computation. Figure 6.5 shows the distribution of the consequences of the fault injection campaign in accessible registers.

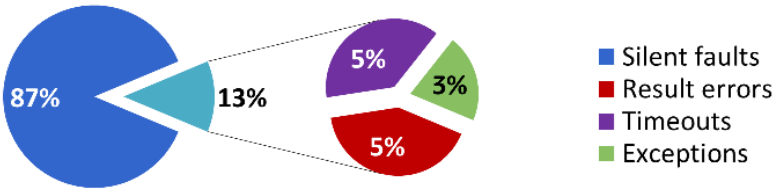

Figure 6.5: Consequences of fault injection in accessible registers

6.1.3 Error rate prediction

During the static radiation test, cache memories as well as accessible registers were targeted. Results show 12 errors in cache memories and no errors in tested registers. Therefore, the equation (4.3) is reduced to obtain the predicted error-rate as follows:

$$\tau_{SEU} = \tau_{SEU\_CACHE} = \tau_{\text{inj}} * \sigma_{\text{STATIC}} * Mf \qquad (6.1)$$

From the obtained results in sections 6.1.1 and 6.1.2, the confidence interval of the σ_{STATIC} was $[4.40 - 14.9]\text{x}10^{-9}\ \frac{cm^2}{device}$ and the τ_{inj} was 65%. In addition, since the 80x80 matrix multiplication occupies 75KB of data, and the size of the L2 cache is 128KB, the memory utilization factor (Mf) is:

$$\text{Mf} = \frac{75\ \text{KB}}{128\ \text{KB}} = 0.59$$

Then the predicted error-rate is:

$$\tau_{SEU} = 0.65 * 8.51x10^{-9} * 0.59 = 3.26x10^{-9}\ \frac{cm^2}{dev}$$

Being the confidence interval for the predicted error-rate:

$$\tau_{SEU} = 0.65 * [4.40 - 14.9]x10^{-9} * 0.59 = [1.68 - 5.71]x10^{-9}\ \frac{cm^2}{dev}$$

For validating the predicted error rate, it is necessary to compare this value with the measured one. The measured error-rate corresponds to the dynamic cross-section interval which is $[2.17 - 7.33]\ \mathrm{x}10^{-9}\,\frac{\mathrm{cm}^2}{\mathrm{dev}}$. Figure 6.6 shows the predicted and measured application error-rates for a 95%confidence interval.

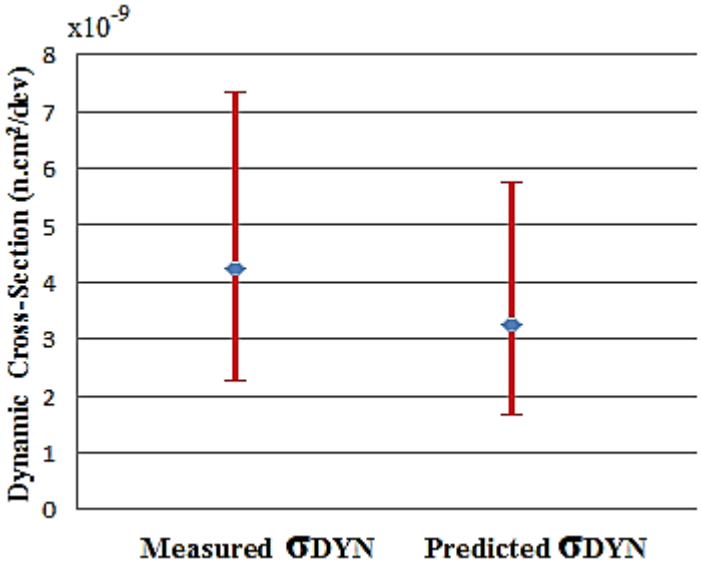

Figure 6.6: Predicted and measured application error-rates for the P2041

In Figure 6.6, it can be seen an underestimation of the predicted error rate that can be explained as follows: 1) most of SPR are non-writable and thus, they were not targeted in the static test and fault injection campaigns, 2) the device implements ECC and parity in their cache memories that correct most of the detected errors either by the ECC, or by cache invalidation. It may affect the accuracy of the prediction. However, the relationship between the two intervals suggests that the prediction approach provides an acceptable estimation of the error rate which can be very useful to evaluate any implemented application.

6.2 Evaluation of the Adapteva E16G301 Epiphany microprocessor

Unlike the other selected devices, the Epiphany multi-core does not implement protection mechanisms like ECC or parity in its internal memories. This fact is very interesting for analyzing the behavior of this device in presence of SEE.

As it was stated in the Parallella architecture description, the dual core ARM A9 is the host processor and the Epiphany E16G301 multi-core is the co-processor devoted to parallel computing. For inter-processor and multi-core to host communications, it was defined a shared memory space in the external DDR memory. Since the co-processor does not have direct access

to ***printf*** function for logging results, it has to write the information in the external DDR memory of the board, and then the *host* reads this information to log it.

For fault-injection purposes, all the variables used by the application are placed in the internal local memory of each processor. The variables of the program can be modified at any time by the fault injector core since the Epiphany architecture allows all processor cores accessing the whole internal memory of the chip.

In the Epiphany, it was implemented a master-slave scheme to guarantee the synchronization between core processors. In addition, the *master* core is also synchronized with the host processor. The *host* monitors the application and logs the results. Also during the test campaigns, at each execution the *host* initializes the input matrices with predefined values and the output matrix with 0's.

For initializing each execution, the *master* core is waiting for a *start* flag from the *host* processor through a writing operation in the external memory. In turn, *slave* cores are waiting for the start signal from the master core which writes a variable in the slave's local memory space. Once finished the calculation, each core compares the obtained results with a set of correct values previously obtained, and writes a flag value in a predefined location of its local memory for indicating to the master the end of the application. The *master* core waits until all the cores write the end value and finally, it orders the *host* to log the results.

The benchmark algorithm that was tested via the CEU fault-injection approach was a standard 45x45 matrix multiplication. The input matrix A was filled up with 5's while matrix B was filled up with 6's, thus the expected result was 1350 for all the elements of matrix C. The total number of variables used for the implementation of the matrix multiplication is 6078, distributed in 4050 input variables, 2025 output variables and 3 indexes for loop operations. Each variable was implemented in 32 bits, thus the targeted sensitive area was about 24 KB that perfectly fits in the 32 KB local memory. It is important to note that the local memory has to contain both code and data. The size of the matrix was selected so that data occupy as much local memory space as possible leaving enough space for the program's code.

6.2.1 Neutron radiation campaigns

In order to protect the device, it is required the use of an anti-latch-up circuit when the multi-core processor is exposed to neutron radiation. This circuit was detailed in the Parallella platform description. Table 6.11 summarizes the sensitive zones of the multi-core processor.

Table 6.11: Sensitive zones of the Epiphany E16G301 multi-core processor

Sensitive zone	***Location***	***Capacity***	***Description***
SRAM	Processor Core	32 KB per core	Local memory
GPR	Processor Core	64 registers of 32 bits per core	General Purpose Registers
SPR	Processor Core	42 registers of 32 bits per core	Special Purpose Registers

Experimental Setup

Two radiation test campaigns were performed on the multi-core processor: one static and one dynamic test. The DUT was placed facing the center of the target perpendicularly to the beam axis at a distance of 38.5 ± 0.5 cm. The neutron beam energy was 14 MeV with an estimated constant flux of $7.2 \text{ x}10^4 \text{ n}.cm^{-2}.s^{-1}$ with an error of $0.1 \text{ x}10^4 \text{ n}.cm^{-2}.s^{-1}$.

In the Parallella board the physical distance between the Epiphany multi-core and the *host* processor is less than one centimeter. Thus it was required to limit the neutron flux for avoiding particles affecting the host processor. In addition, special attention was required to protect the rest of the platform components from radiation. For that, the Epiphany multi-core was irradiated through a small hole in a 5 cm thickness polypropylene block intended to protect the platform. In the presented experiments, the voltage supply was controlled by using a camera, available in the casemate of the accelerator facility, for monitoring voltage and current parameters of the power supply.

Static Sensitivity

To evaluate the intrinsic sensitivity of the multi-core processor, a static radiation test targeting the internal memory of each core was performed. The *host* processor was in charge of filling the internal memory of the multi-core with a predefined pattern using the Epiphany SDK Utilities E-READ and E-WRITE (Adapteva, 2013). For this reason, the whole internal memory of the multi-core could be targeted. At the end of the experiment, the static cross-section σ_{STATIC} was estimated to obtain the intrinsic sensitivity of the device built-in 65nm CMOS technology.

The test was performed with an exposure time of 1 hour, providing a fluence around $2.59 \text{ x}10^8 n.cm^{-2}$. During the tests, 23 SEU and 1 MCU that produce bit-flips in the written pattern were detected. In addition, 2 SEFI that caused hangs in the *host* processor were observed

but not considered for the results since the DUT is only the Epiphany co-processor. Table 6.12 summarizes the results of the static radiation campaign. Note that the subscript number following MCU represents the multiplicity of the upset.

Table 6.12: Results of the static radiation test campaigns

SEE Type	***Test***	***Consequences***
SEU	23	Bit-flip
MCU (2)	1	Bit-flip
Total	24	

Table 6.13 shows a sample of data containing a bit-flips caused by SEU and MCU in the local memory, and logged during the experiment.

Table 6.13: Example of the obtained results in the static tests

PATTERN	***SEE TYPE***	***CORE***	***ADDRESS***	***DATA HEX***	***DATA BIN***
0x55AA55AA	SEU	3	0x7C84	0x55AA**4**5AA	0b0101 0101 1010 1010 010**0** 0101 1010 1010
	SEU	7	0x52B4	0x55AA55A**B**	0b0101 0101 1010 1010 0101 0101 1010 101**1**
	MCU(2)	2	0x7110	0x55AA5**D**AA	0b0101 0101 1011 1010 0101 **1**101 1010 1010
			0x7130	0x55AA5**D**AA	0b0101 0101 1011 1010 0101 **1**101 1010 1010

The results from the static test allow estimating the static cross-section of the CMOS 65nm shared memory as follows:

$$\sigma_{STATIC} = \frac{numb\ of\ SEE}{Fluence} = \frac{24}{2.59x10^{8}} = 9.27x10^{-8}\ \frac{cm^2}{dev}$$

A confidence interval must be applied to this result due to the scarcity of experimental data. Therefore, for a 95% confidence level, the lower and upper limits for the dynamic cross-section are:

$$0.59\text{x}10^{-7}\,\frac{\text{cm}^2}{\text{dev}} < \sigma_{\text{STATIC}} < 1.38\text{x}10^{-7}\,\frac{\text{cm}^2}{\text{dev}}$$

Since the tested memory area of the multi-core processor represents 4194304 bits, the cross section per bit is estimated at $2.21\text{x}10^{-14}\ cm^2/bit$ and with a 95% confidence interval $[1.4 - 3.29]\ \text{x}10^{-14}\ cm^2/bit$.

Dynamic Response

The second radiation test campaign was carried out to obtain the dynamic cross-section (σ_{DYN}) of an application running in the multi-core processor. The dynamic test was performed with an exposure time of 1 hour providing a fluence around $2.59x10^{8} n.cm^{-2}$. Table 6.14 summarizes the results of the dynamic radiation campaign.

Table 6.14: Results of the dynamic radiation test campaigns

SEE Type	***Test 1***	***Consequences***
SEU	7	Silent faults
SEU	11	Erroneous results
SEFI	1	Hangs
Total	19	

From the results presented in Table 6.14, only erroneous results and hangs were taken into account to calculate the dynamic cross-section as follows.

$$\sigma_{DYNAMIC} = \frac{numb\ of\ SEE}{Fluence} = \frac{12}{2.59x10^{8}} = 4.63x10^{-8}\ \frac{cm^{2}}{dev}$$

Since the application's logs provide information not only about errors but also about silent faults, the experimental application error-rate is about 63% which is coherent with the error rate obtained from fault injection campaigns which is 59%. Note that the value closest to the reality should be the one issued from fault injection due to the huge quantity of samples compared to the experimental one. As in the static case, it is imperative to add uncertainty margins to these results due to the lack of experimental data. Then, the lower and upper limits for the dynamic cross-section for a 95% confidence interval are:

$$2.39\text{x}10^{-8}\frac{\text{cm}^{2}}{\text{dev}} < \sigma_{\text{DYN}} < 8.09\text{x}10^{-8}\frac{\text{cm}^{2}}{\text{dev}}$$

Table 6.15: Example of erroneous result observed during the dynamic tests

Core	***Affected Zone***	***Written Datum***	***Bit altered***	***Modified Datum***	***Obtained Result***	***Expected Result***
8	Input Matrix A	0x1 / 1	12	0x1001 / 4097	0x205A / 8282	0x5A / 90
15	Input Matrix B	0x2 / 2	14	0x4002 / 1638	0x405A / 16474	0x5A / 90
15	Output Matrix C	N/A	0	N/A	0x5B / 91	0x5A / 90

In Table 6.15, it is shown a sample of some erroneous results produced by bit-flips affecting the input and output matrices. The bit that has flipped was easily detected since the initialization values of the matrices are known. Note that the erroneous results were propagated along the output matrix but the table shows just a sample of them. Since the device does not implement protection mechanisms, the erroneous results would be taken as valid results affecting dramatically the reliability of the multi-core.

6.2.2 Fault injection campaigns in local memories

Fault injection campaigns were performed in the internal memory of the device. Registers were not taken into account since during radiation experiments there were no events observed in registers. As in the P2041 case, the *master* plays the role of fault injector whereas the *slave* cores execute in parallel the tested application. While the application is running, the master core performs the fault injection. This fault injection campaign targets 512 KB of internal memory of the device.

The experiment considers the injection of one SEU per execution in the local memory of each processor. SEU faults were injected at instants chosen within the nominal duration of the executed program which was around 40376 clock cycles at 600 MHz frequency. Table 6.16 shows a general overview of the fault injection campaign where 119870 faults were injected in the local memory.

Table 6.16: Results of the fault injection campaign

Zone	***Silent Faults***	***Erroneous Results***	***Time-outs***	***Exceptions***
Local memory	48849	71009	3	9

From these results, it can be calculated the error rate of the internal memory applying the equation (4.1) and considering as errors the erroneous results, time-outs and hangs.

$$\tau_{\mathrm{INJ}} = \frac{\text{Number of Errors}}{\text{Number of Fault Inj.}} = \frac{71021}{119870} = 59.24\text{x}10^{-2}$$

This result shows that 59.24% of the injected SEUs cause errors in the application.

6.2.3 Error rate prediction for the Epiphany multi-core

During the static radiation test, shared memories as well as accessible registers were targeted. Results show errors only in shared memories and no errors in tested registers. In addition, due to the *host* processor logs the results, this leads a loss of the exposure time to

radiation of the memory cells of the Epiphany multi-core during the dynamic test. Therefore, the equation (4.3) to obtain the predicted error-rate is reduced and expressed as follows:

$$\tau_{SEU} = \tau_{SEU\_SHARED} = \tau_{inj} * \sigma_{STATIC} * Mf * Etf \quad (6.2)$$

The above equation can be applied recalling the τ_{inj} presented in sections 6.2.1 that was 59.24%, and the confidence interval of the σ_{STATIC} presented in section 6.2.2 that was $[0.59 - 1.38]\text{x}10^{-7}\ \frac{\text{cm}^2}{\text{device}}$.

The memory utilization factor (Mf) is calculated considering the local memory size of each core that is 32768 bytes and the memory occupied by the matrix multiplication that is 24303bytes, which gives:

$$Mf = \frac{24303\ \text{bytes}}{32768\ \text{bytes}} = 0.74$$

Since the duration of one execution is about 27597 µs, the exposure time loos is around 0.89% and the exposure-time loss factor (Etf) is 0.99. Table 6.17 shows the exposure time loss during the dynamic test.

Table 6.17: Exposure time loss during the dynamic test

Synchronization Host-Master core		***Synchronization Master–Slave cores***			
Host exceeding time	Waiting start signal from Host	Waiting start signal from Master	Waiting ACK from Slaves	Comparing Results time	Total (us)
225	13.48	3.55	1.1	1.44	244.57

Then the predicted error-rate is:

$$\tau_{SEU} = 59.24x10^{-2} * 9.27\ x10^{-8} * 0.74 * 0.99 = 4.02\ x10^{-8}\ \frac{cm^2}{dev}$$

Considering the previous value, the 95% confidence interval is:

$$\tau_{SEU} = 59.24x10^{-2} * [0.59 - 1.38]\ x10^{-7} * 0.74 * 0.99 = [2.50 - 5.98]\ x10^{-8}\ \frac{cm^2}{dev}$$

In order to validate the prediction approach for the studied multi-core processor, the predicted error-rate and the measured one, which is $[2.39 - 8.09]\ \text{x}10^{-8}\ cm^2/device$, are compared. Figure 6.7 depicts the predicted and measured error-rate intervals.

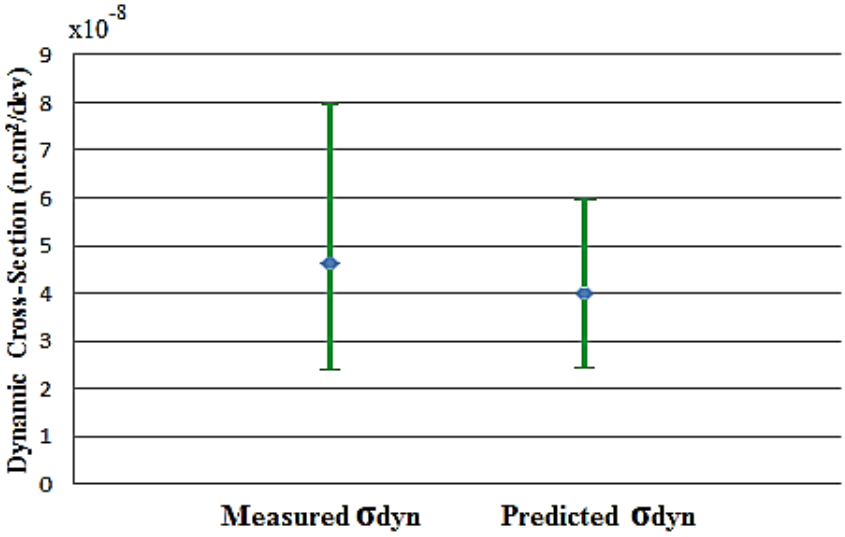

Figure 6.7: Predicted and measured application error-rates for the Epiphany

In Figure 6.7, it is shown that the predicted value is quite close to the measured one. The small underestimation of the predicted error rate may be explained due to the occurrence of a SEFI during the dynamic test. This SEFI probably have affected a sensitive zone that was not targeted during fault injection and static test. Despite of the lack of protection mechanisms, the Epiphany E16G301 is particularly well suited for several applications according to Table 6.18 taken from the E16G301 Datasheet (Adapteva, 2011a).

Table 6.18: Epiphany E16G301 main applications

Consumer	***Computing infrastructure***	***Mil/Aero***	***Medical***	***Communications***	***Embedded Vision***
Smart-phones	Super Computers	Radar/Sonar	Ultrasound	Communication test bed	Machine vision
Speech recognition	Big data analytics	Extremely large sensor Imaging	CT	Software defined radio	Autonomous Robots
Face detection	Software defined Networking	Military Radios		Adaptive Pre-distortion	Automotive safety

Furthermore, researchers at NASA Ames Research Center have published a report showing the use of Parallella platform, containing the E16G301, for on-board health management of the DragonEye Unmanned Aircraft System (UAS) (Rozier, 2015).

6.3 Evaluation of the Kalray MPPA-256 Many-core Processor

The dynamic response of the MPPA-256 many-core processor was evaluated through the execution of a distributed Matrix Multiplication application which runs independently the same algorithm on each cluster of the device. 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 is the master that distributes the tasks, logs the results, and communicates with the NOC resources. The operation of the MPPA-256 is monitored by the host of the MPPA Developer.

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.

The selected application is a collaborative 256×256 matrix multiplication, where each PE of a given cluster computes $\frac{1}{16}$ of the cluster result. This computation is running repeatedly as stated in equation (6.3)

$$C = \sum_{1}^{256} AxB \tag{6.3}$$

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 one cluster. Since clusters are spawned one after another, cluster 15 starts execution around 15M IO cycles after cluster 0.

Each compute cluster performs in AMP mode 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.

6.3.1 Neutron radiation campaigns

Four radiation test campaigns were performed on the Bostan PCIe board containing the MPPA256 many-core processor: static test, dynamic test cache enabled, dynamic test cache disabled and dynamic test varying operating parameters. For these experiments, there was no additional anti-latch-up circuit since current and voltage levels are automatically controlled by the Bostan PCIe board. Table 6.19 sums up the sensitive zones of the many-core processor.

Table 6.19: Sensitive zones 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	512 KB per cluster	Static Shared Memory
IC-CC	VLIW core	8 KB per core	Instruction Cache
DC-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 I/O	Shared Data cache
GPR	VLIW Core	64 registers of 32 bits per core	General Purpose Registers
SFR	VLIW Core	51 registers of 32 bits per core	System Function Registers

Experimental Setup

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 computer cluster frequency was set to 100MHz to reduce power consumption. The bias voltage of the device was set to 0.9V. Note that only for the last test campaign the bias voltage and the operating frequency were changed. The neutron beam energy was 14 MeV with an estimated flux of $1.2\,\text{x}10^5$ n.$cm^{-2}.s^{-1}$ at 500 Hz frequency with an error of $0.1\,\text{x}10^5$ n.$cm^{-2}.s^{-1}$. For protecting the rest of the platform, a 5 cm thickness polypropylene block was used. In the presented experiments, the power supply was controlled by means of a current-voltage controlling module implemented in the platform (PCIe board MPPA).

Static Sensitivity

To evaluate the intrinsic sensitivity of the many-core processor, a static test targeting the Static Shared Memories (SMEMs) of the compute clusters was performed. It was done

since SMEMs occupy most of the storage area of the device. To accomplish this task, an initialization program is loaded by means of the JTAG into all the SMEMs and is executed by the RM core of each compute cluster. This program writes 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 the predefined pattern along the radiation test. In the case the code finds any mismatch or detects a SECC or DECCS error, it is reported with a message sent to the host by the JTAG, specifying the nature of the error and the implicated SMEM address where it occurred. Then, the right value is written 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 a 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 (σ_{STATIC}) is obtained to estimate the intrinsic sensitivity of the memory cells belonging to the SMEMs built in CMOS 28nm TSMC HP technology.

A first radiation test campaign was devoted to obtain the static cross-section (σ_{Static}) 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 around 8.64 x$10^8 n.cm^{-2}$. During the test, 2720 Single-bit ECC (SECC) events were detected with no consequences to the self-test application since SMEMs implement ECC. In addition, one SEFI that caused a hang was observed. Note that no latch-up events were detected during the experiment which means that the Bostan PCIe board effectively manages the current and voltage levels.

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 6.8 shows the distribution of the observed bit-flips in the shared memories of the 16 clusters.

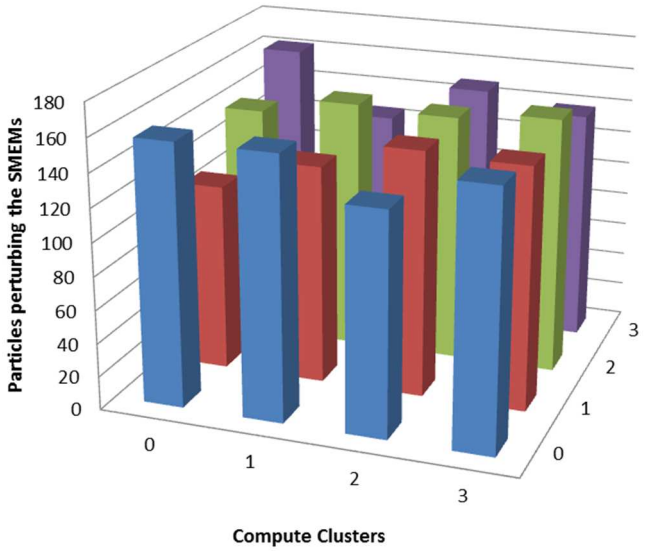

Figure 6.8: Distribution of the observed bit-flips in the SMEMs of the clusters. The axis x and z represent the cluster coordinates.

Figure 6.9 presents the SER count evolution of the MPPA during the 2 hours of test. It can be seen that an average of 100 bit-flips have been detected by the many-core each 5 minutes. Figure 6.8 and Figure 6.9 provide a temporal and spatial distribution of the bit-flips caused by neutron particles coming from an isotropic neutron beam. However, as temporal distribution is not homogeneous, the difference between the number of particles perturbing the device during static and dynamic tests may affect the error-rate prediction.

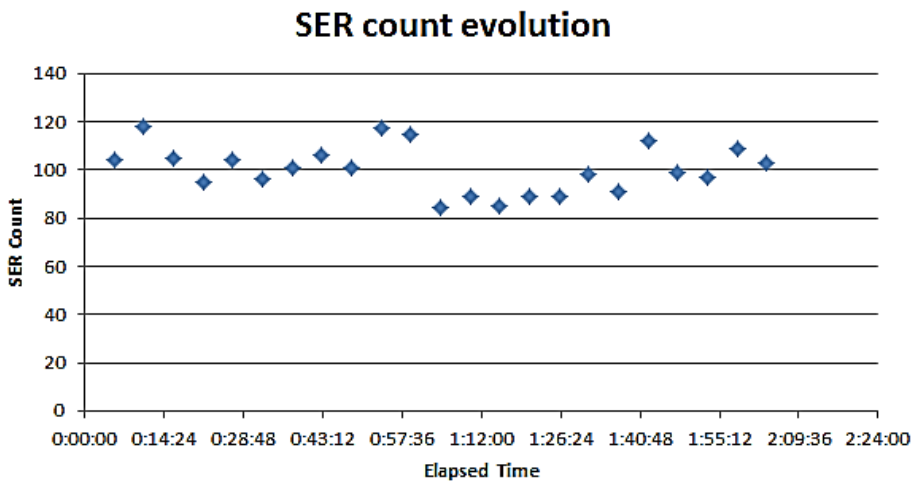

Figure 6.9: SER count evolution of the MPPA-256 during the radiation test

Table 6.20 summarizes the results of the static radiation campaign. Note that the subscript number following the *MCU* represents the multiplicity of the upset.

Table 6.20: Results of the static radiation campaign

Detected Error	*SEE Type*	*Occurrences*	*Bit-flip Cells*
SECC	SEU	1949	1949
SECC	MCU (2)	322	644
SECC	MCU (3)	24	72
SECC	MCU (4)	8	32
SECC	MCU (5)	2	10
SECC	MCU (6)	1	6
SECC	MCU (7)	1	7
Other error	SEFI	1	1
Total		2309	2721

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

$$\sigma_{STATIC\_CMOS} = \frac{number\ of\ SEE}{Fluence} = \frac{2721}{8.64x10^{8}} = 3.15x10^{-6}\ \frac{cm^2}{dev}$$

Since the tested SMEM area of the many-core processor represents $2.6\text{x}10^{8}$ bits, the static cross-section per bit of the TSMC CMOS technology is about $1.21\text{x}10^{-14}(\text{cm}^2/\text{bit})$. Assuming that the technology of the memory cells is similar for the different memory elements of the device, the cross-section of the processors' registers can be extrapolated from the cross-section per bit. Taking into account that there are [64 (GPRs) + 50 (SFRs)] registers containing 32 bits each one, and [256 (PE) + 32 (RM)] core processors, the sensitivity of the registers of the device can be expressed as:

$$\sigma_{STATIC} = 1050624\ \text{bit} * 1.21\ \text{x}\ 10^{-14}\frac{cm^2}{bit} = 12.71x10^{-9}\ \frac{cm^2}{dev}$$

Dynamic Response

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 σ_{DYN} and the average execution times were obtained. 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.

Dynamic Test with Cache Memories Enabled

The second radiation test campaign was carried out to obtain the dynamic cross-section cache enabled ($\sigma_{dyn_{ce}}$) of an application running in the many-core processor. For this test, the instruction and data cache memories of the compute cluster's cores were enabled and configured in *write-through mode.* The exposure time for this campaign was one hour providing a fluence about $4.32x10^{8}(n.cm^{-2})$. Table 6.21 summarizes the results of the dynamic radiation campaign.

Table 6.21: Results of the dynamic radiation test of the MPPA-256 many-core 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 comparison failed	SEU	2	Erroneous Result
Total		721	

From Table 6.21, 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 6.22 shows an example of an erroneous result produced during the radiation test.

Table 6.22: Example of an application error result occurred during the dynamic radiation 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 6.22, it can be seen that four float values belonging to cluster 14 were miscalculated (most significant part of the values). The 1KB distance between them and the analysis of the assembly code show that this result error was caused by a corruption of the GPR 41 when used in the main loop of the matrix multiplication. This register stores a float 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 was obtained during the fault injection campaign 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 (Vargas, 2017).

Due to the scarcity of experimental data, it is compulsory to add uncertainty margins to these results as explained in chapter 4 section 4.4.7. Then, for a 95% confidence interval, the lower and upper limits for the dynamic cross-section are:

$$1.43\text{x}10^{-9}\frac{\text{cm}^2}{\text{dev}} < \sigma_{\text{dyn\_ce}} < 20.29\text{x}10^{-9}\frac{\text{cm}^2}{\text{dev}}$$

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 around 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.

Dynamic Test with Cache Memories Disabled

The third radiation 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 second campaign. Table 6.23 summarizes the results of this dynamic radiation campaign. SECC errors produced in the SME's were corrected by the ECC while Register Trap and Memory Comp. Failed errors remain uncorrected. Only these uncorrected errors were taken into account to determine the dynamic cross-section.

Table 6.23: Results of the dynamic radiation test of the MPPA-256 many-core cache disabled

Type of error	*SEE Type*	*Occurrences*	*Consequences*
SECC	SEU	602	None
Register Trap	SEFI	1	Hang
Memory comparison failed	SEU	1	App. result error
Total		604	

Applying a similar analysis as in the previous case, the lower and upper limits for the dynamic cross-section are:

$$0.56x10^{-9}\frac{cm^2}{dev} < \sigma_{dyn\_cd} < 16.72x10^{-9}\frac{cm^2}{dev}$$

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 around 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 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.

Dynamic Test Varying Operating Parameters

A fourth radiation campaign was performed in order to observe the dynamic response of the many-core processor with its cache memories enabled when varying the device operating frequency and bias voltage. For both cases, the fluence was about 7.2 x$10^7(n.cm^{-2})$. Table 5.7 summarizes the results when varying the operating frequency with a bias voltage of 0.9V. Table 6.24 sums up the results when varying the bias voltage with a 100 MHz frequency.

Table 6.24: 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

In table Table 6.24 it can be seen that at 200MHz frequency the device is less sensitive to neutron radiation. On one hand, at 200MHz the execution time of the application is reduced compared to the one at 100MHz, which leads to a less exposure time. On the other hand, executing the application at 300MHz frequency leads to more power consumption which may affect the neutron sensitivity of the device.

Table 6.25: 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

Regarding the bias voltage, in Table 6.25 it can be seen that at 0.9V there are less detected errors. It confirms that the device is less sensitive when operating at its nominal bias voltage.

6.3.2 Fault injection campaigns in processor registers

For this campaign, only processor registers were taken into account since there were no errors in SMEMs and cache memories due to the implementation of ECC, interleaving, and parity protection that invalidates the cache in case of errors. This experiment considers one SEU injection per cluster since each CC performs the application independently of the others. One fault is also injected per execution in the GPRs or SFRs of the processor cores (PE or RM) of each cluster. In order to avoid the propagation of errors to the next execution, the HOST resets the platform and reloads the code to the MPPA processor after each run. Hence, the random variables required by the fault-injector are provided by the HOST, being the random instant, core, register and bit additional arguments of the main function.

While the application is running on the processing engine cores, at the injection time the resource manager core of each compute cluster performs the fault injection. If the target core is a PE, the RM core sends an inter-processor interrupt to the selected core and the latter performs the bit-flip in its register. The fault injection campaign targets 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 of the 51 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 contrary, other registers such as processing identification (PI), system reserved (SR0-SR5), performance monitor (PM0-PM3) and the hardwired registers were not targeted. The SPC and the SPS registers serve to emulate bit-flips in the PC and PS registers respectively. This is done due to the fact that in the context switching of the interruption routine, the value of the PC and PS are saved in the shadow registers (SPC and SPS) and when the program flow exits from this routine, it uses the values of these registers to be restored in the PC and PS to continue with the current program execution.

SEU faults were injected at instants chosen within the nominal execution time of the application which was around $5.3x10^8$ clock cycles. Table 6.26 shows a general overview of the fault-injection campaign where 94316 faults were injected in the GPRs and accessible SFRs.

Table 6.26: Results of the first fault injection campaign

Zone	***Silent Faults***	***Erroneous Results***	***Timeouts***	***Exceptions***
GPRs	36472	16387	6678	1996
SFRs	22745	2034	6365	1639
Total	59217	18421	13043	3635

From these results, it is calculated the error rate of the registers applying the equation (4.1) and considering as errors the erroneous results, timeouts and hangs.

$$\tau_{\mathrm{INJREG}} = \frac{\text{Number of Errors}}{\text{Number of Fault Inj.}} = \frac{35099}{94316} = 37.21\text{x}10^{-2}$$

This result shows that 37.21% of the injected SEUs in the accessible registers cause errors in the application.

6.3.3 Error rate prediction for the MPPA-256 many-core

Due to the MPPA-256 implement ECC and interleaving in its shared memories and parity in the processor core's caches memories, all the events observed during radiation tests were corrected. Then, the prediction of the error rate is based on the contribution of processor registers. On one hand, the error rate issued from the fault-injection in registers is 37.21%. On the other hand, the σ_{STATIC} from radiation experiments is $12.71\text{x}10^{-9}\ \frac{cm^2}{device}$. Then, applying the equation $\tau_{\mathrm{SEU}} = \tau_{\mathrm{inj}} * \sigma_{\mathrm{STATIC}}$ the predicted application error- rate is :

$$\tau_{SEU} = 37.21x10^{-2} * 12.71x10^{-9} = 4.73\ x10^{-9}\ \frac{cm^2}{dev}$$

In order to validate the prediction approach for this many-core processor, the predicted value should be compared with the measured one. In this case, the dynamic cross-section can be calculated gathering the results from both dynamic tests cache enabled and disabled. 3 SEU and 2 SEFI in general purpose registers that cause erroneous results or hang were added. The average fluence during the two hours of test was of $8.64\ x10^{8}(n.cm^{-2})$. Then, the dynamic cross-section is:

$$\sigma_{Dyn} = \frac{5}{8.64x10^{8}} = 5.78x10^{-9}\ \frac{cm^2}{dev}$$

For a 95% confidence interval, the measured application cross-section is:

$$\sigma_{Dyn} = [1.87 - 13.50\ x10^{-9}]\ \frac{cm^2}{dev}$$

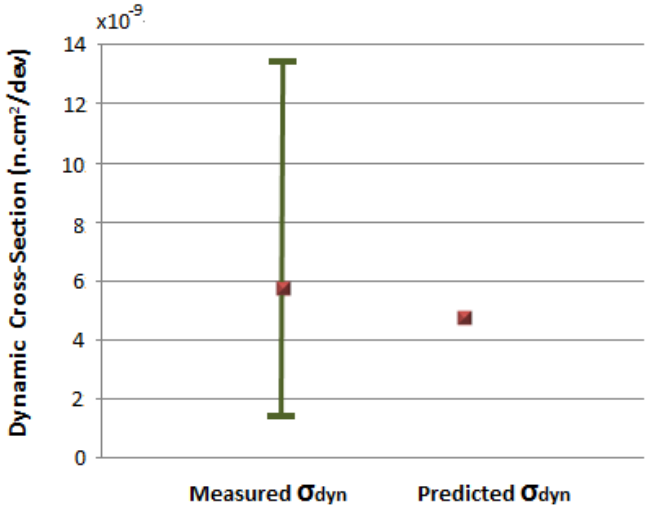

Figure 6.10: Predicted and measured application error-rates for the MPPA-256

Figure 6.10 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 during the fault injection campaigns.

6.4 Overall Comparison

This section is devoted to provide an overall comparison of the three multi/many-core processors evaluated in the context of this thesis. At this point, it is convenient recalling their main characteristics. The P2041 multi-core processor is manufactured in 45nm SOI technology which integrates four e500mc processor cores and implements ECC and parity in its cache memories. The second one is the Adapteva Epiphany E16G301 microprocessor manufactured in 65nm CMOS process which integrates 16 processor cores and does not implement any protection mechanism on its shared memories. The last one is the Kalray MPPA-256 many-core manufactured in 28nm TSMC CMOS technology which integrates 16 compute clusters each one with 17 VLIW core processors (1 resource manager and 16 processing elements) and implements ECC and parity in its shared and cache memories.

6.4.1 Failure-rate comparison

For assessing the failure-rate, equations (2.2) and (2.3) are used referenced to the neutron flux of NYC. Therefore, for obtaining the FIT and FIT/Mb values the following equations were applied:

$$\text{FIT} = \sigma \frac{cm^2}{n.device} x\, 14 \frac{n}{cm^2.h} \times 10^9 h$$

$$\frac{\text{FIT}}{\text{Mb}} = \sigma \frac{cm^2}{n.bit} \times 14 \frac{n}{cm^2.h} \times 10^9 h \times 10^6 bit$$

Table 6.27 summarizes the results obtained during the static tests for the three studied devices, and the calculated worst-case FIT.

Table 6.27: Worst case error-rate of the studied devices

DEVICE	σ_{STATIC} ***[cm²/device]***	σ_{STATIC} ***[cm²/bit]***	***SOFT ERROR RATE [FIT]***	***SOFT ERROR RATE [FIT/Mb]***
FREESCALE P2041 quad-core	$8.51x10^{-9}$	$5.79x10^{-16}$	119 ±40%	8 ±40%
ADAPTEVA E16G301 multi-core	$9.27x10^{-8}$	$2.21x10^{-14}$	1298 ±29%	309 ±29%
KALRAY MPPA-256 many-core	$12.71x10^{-9}$	$4.86x10^{-17}$	178 ±4%	0.68 ±4%

A similar calculation was done for obtaining the failure-rate of the application running on the selected devices. The results are presented in Table 6.28.

Table 6.28: Error rate comparison of the studied applications

DEVICE	σ_{DYN} ***[cm²/device]***	σ_{DYN} ***[cm²/bit]***	***SOFT ERROR RATE [FIT dyn]***	***SOFT ERROR RATE [FIT dyn /Mb]***
FREESCALE P2041 multi-core	$4.25x10^{-9}$	$2.89x10^{-16}$	60 ±40%	4 ±40%
ADAPTEVA E16G301 multi-core	$4.63x10^{-8}$	$11.03x10^{-15}$	648 ±29%	154 ±29%
KALRAY MPPA-256 many-core	$5.78x10^{-9}$	$2.14x10^{-17}$	81 ±4%	0.3 ±4%

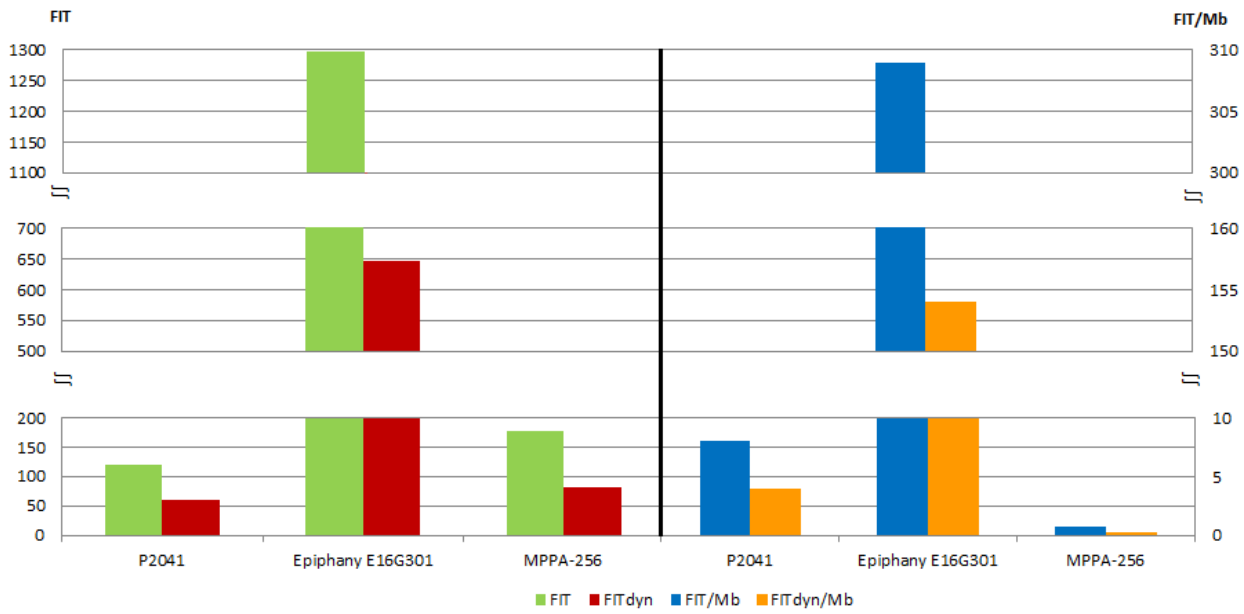

Figure 6.11: Comparison of the failure-error rates

A comparison of the failure rate for both scenarios is presented in Figure 6.11. On one hand, in the figure it can be seen that P2041 multi-core is the most reliable processor taking into account the failure rate per device. However, the difference with the MPPA-256 is not so ample as one could expect due to the difference in manufacturing technology (45nm SOI vs a 28nm CMOS). On the other hand, if the FIT/Mb is considered, the reliability of the MPPA-256

surpasses the P2041 one. This high difference in the reliability per Mb is mainly due to the effective implementation of ECC and interleaving in its shared memories, as well as parity in its cache memories, which have detected and corrected most of the observed errors during the radiation test. In turn, the failure rate of the Epiphany E16G301 is ten times greater than the one of the P2041. This is explained by the fact that Epiphany has no protection mechanisms.

6.4.2 Predicted and measured error-rate comparison

Figure 6.12, Figure 6.13, and Figure 6.14 show a comparison between the predicted and the extrapolated reliability of the application implemented in the P2041 multi-core, Epiphany multi-core and MPPA many-core processor respectively. The reliability curves, $R(t) = e^{-\lambda t}$ are plot from the predicted and the measured values of the dynamic cross-section of the devices and extrapolated at avionic altitude (35000 feet) where the neutron flux is about $2.99\ x10^{3}\ n.cm^{-2}.h^{-1}$. The considered period of time is 50000 hours which is the estimated average lifetime of a commercial aircraft.

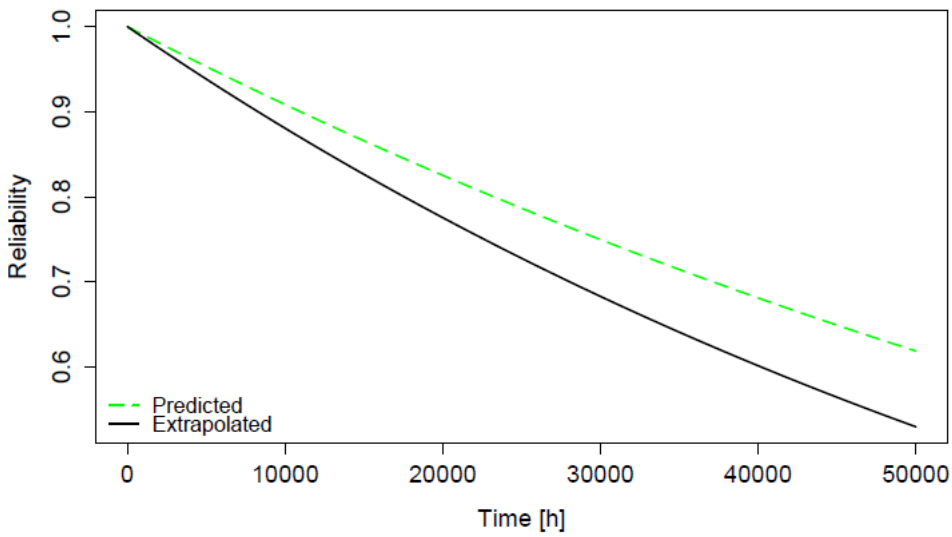

Figure 6.12: Predicted and extrapolated reliability of the P2041 multi-core processor

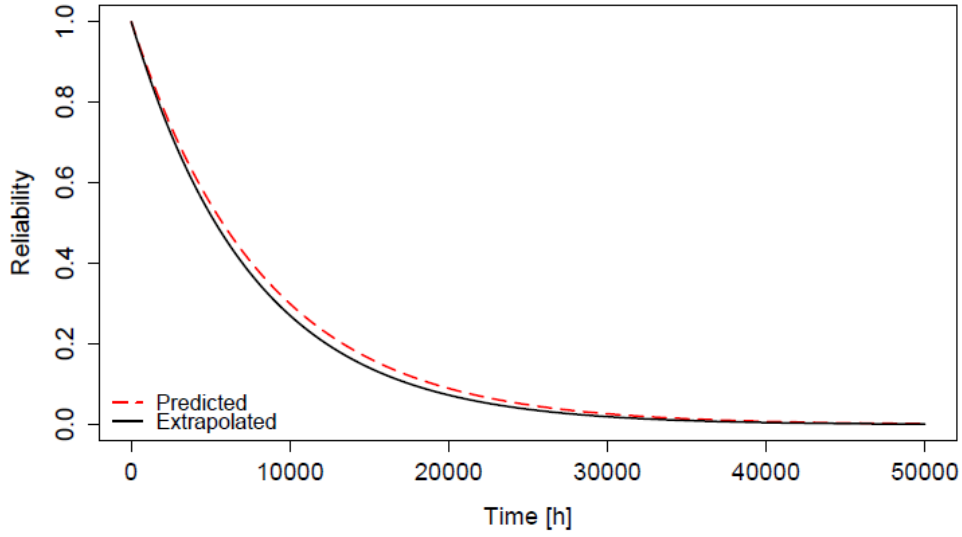

Figure 6.13: Predicted and extrapolated reliability of the Epiphany multi-core processor

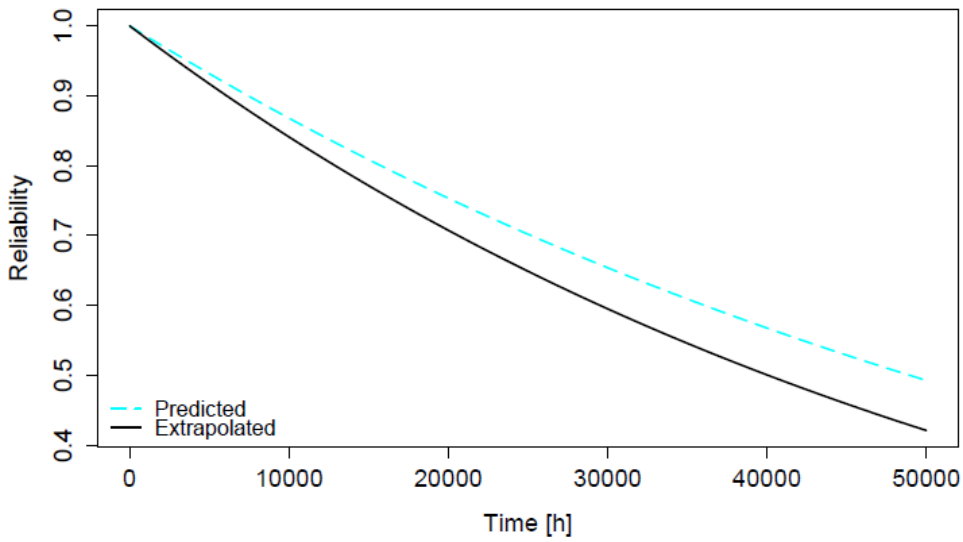

Figure 6.14: Predicted and extrapolated reliability of the MPPA many-core processor

From the figures, it can be seen an underestimation produced by the fact that not all sensitive-zones could be targeted during the static radiation test and the fault-injection campaigns. This is explained because there are some memory-cells that cannot be accessed by the user and other that cannot be written by software means. The most accurate prediction is the one achieved for the Epiphany multi-core since this device does not implement error detection and correction mechanism. The results corroborate that these mechanisms may affect testing and error-rate prediction (LaBel, 2005).

Moreover, a cross comparison between the reliability curves shows that the most reliable device is the P2041. It confirms that the process technology plays a preponderant role in the device reliability.

On the other side, as the reliability of multi/many-core processors strongly depends on the implemented application (software), the failure rate of the studied devices can be classified within the DO-178B Software Considerations in Airborne Systems and Equipment Certification. The DO-178B is a guideline used as *de facto* standard for developing avionic software systems (Ferrel, 2014). Table 6.29 shows the level of failure condition regarding the failure rate.

Table 6.29: Failure condition levels according DO-178B

Level	Failure Condition	Objectives	With independence	Failure rate
A	Catastrophic	66	25	10^{-9}/h
B	Hazardous	65	14	10^{-7}/h
C	Major	57	2	10^{-5}/h
D	Minor	28	2	10^{-3}/h
E	No Effect	2	0	N/A

The failure conditions for avionic systems are describe as follows:

Catastrophic: Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft.

Hazardous: Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers. (Safety-significant)

Major: Failure is significant, but has a lesser impact than a Hazardous failure (e.g. passenger discomfort) or significantly increases crew workload (safety related)

Minor: Failure is noticeable, but has a lesser impact than a Major failure (e.g. passenger inconvenience or a routine flight plan change)

No Effect: Failure has no impact on safety, aircraft operation, or crew workload.

Results show that the failure rate of the P2041 multi-core and MPPA-256 many-core executing a matrix multiplication as application, reaches level C of the DO-178B, being the

devices well suited for major failure conditions of avionic applications. On the other hand, the Epiphany multi-core processor reaches level D, being convenient for minor failure conditions.

6.5 Discussion

Radiation experiments performed with 14 MeV neutrons and fault injection campaigns are useful techniques for evaluating the intrinsic sensitivity, dynamic response, and application sensitivity of multi-core and many-core processor. The following is a summary of the analysis of the results presented in this chapter.

P2041 multi-core processor

Fault injection campaigns in program variables show that input matrix A and especially input B are much more sensitive to SEU than the output matrix due to a greater exposure time. Results shows the relevance of fault injection to analyze the behavior of an application in presence of SEUs, providing the possibility to modify the program code according to the obtained results for reducing the impact of faults in the results of the application.

Thanks of using the *machine check error report* during the radiation tests, it was possible to log all the detected SEEs, even those that were corrected by the protection mechanisms implemented in cache memories. It allowed assessing the sensitivity to neutron radiation of the 45nm SOI technology. The obtained results show that 45 nm SOI technology is between 3 and 5 times less sensitive to neutron radiation than its CMOS counterpart.

The analysis of the cluster of errors with the same pattern repeated simultaneously on all the cores during the static radiation test, suggest that a particle perturbed a shared resource belonging to the connectivity infrastructure (CoreNet Coherency Fabric). This fact supports the necessity of a deeper study of SEE consequences on inter-core communications, in spite of the small zone that it occupies with respect to the cache memories.

Dynamic tests have demonstrated that parity implemented in the L1 cache memories is not enough to protect *address tags* and data arrays. The clusters of errors produced in the same read cycle by an MBU affecting the *address tag* of the L1 caches of core 1 and core 2, puts in evidence a possible *3-D* implementation of the L1 cache memories. These results suggest that emerging technologies in cache implementation would potentially affect its sensitive to radiation.

Regarding the error-rate prediction, it is possible to observe an underestimation produced by the fact that not all sensitive zones were targeted during the static test and fault

injection campaigns. Moreover, the protection mechanisms implemented in their cache memories may influence the testing and error prediction.

Epiphany multi-core

Results issued from the static test allow estimating the worst-case sensitivity of the shared memory implemented in CMOS 65nm technology. For this experiment, the neutron flux was reduced almost three times compared to the flux used for the P2041 experiment. This was done to limit the perturbations produced by neutron particles in the *host* processor. However, the applied flux is about eight orders of magnitude greater than the one at avionic altitudes which corresponds to almost 10 years of exposure time of the device to the neutron radiation at 35000 feet. Despite of the efforts for protecting the rest of components of the platform, the SD card containing the Linux OS was corrupted in one of the experiments. This was solved by replacing the tainted SD by a new one.

Comparing the predicted error-rate with the measured one, it is possible to see that the proposed approach provides a good approximation. The small underestimation may be explained due to the fact that not all sensitive areas of the device were targeted during fault injection and static test.

MPPA-256 many-core processor

Due to the complexity of the communication architecture, it was proposed a fault-injection campaign at cluster level for avoiding the use of NOC services which may increase the latency during the fault-injection process. The use of bare-metal programming model allows injecting faults in both RM and PE cores. However, similarly to the other devices, it was not possible to target all registers which leads to an underestimation in the prediction-error rate. In addition, the accuracy of the prediction was affected by the protection mechanisms implement in the shared and cache memories of the MPPA-256.

The static radiation test shows that both the ECC and interleaving implemented in the SMEMs of the clusters are very effective to mitigate errors, since all the bit-flips were detected and corrected. An interesting issue from this test was the multiplicity of the detected MCUs ranging from 2 to 7. It gives some clues about how the organization of the memory may impact in the propagation of errors produced by a single particle.

Dynamic tests have demonstrated that by enabling the cache memories it is possible to increase the performance of the application without compromising the reliability of the device, since cache memories implement an effective parity protection. On the other side, non-

correctable errors were originated in GPRs since registers do not implement any protection mechanism.

A trade-off between the execution time and power consumption aiming at minimizing the impact of neutron radiation was achieved at 200MHz frequency. Regarding the bias voltage, it can be seen that the device is less sensitive to radiation when operating at its nominal voltage (0.9V). Consequently, decreasing the bias voltage for reducing the power consumption of the device is a critical issue to be considered.

Chapter 7 : Conclusions and perspectives

The use of multi-core and many-core processors ranging from safety-critical to commercial applications is rapidly increasing due to the growing demand of high performance, reliability and low power consumption. Moreover, their parallel computing and redundancy capabilities make them ideal candidates for avionics and space applications. However, the integration of several cores in a single integrated circuit leads to further miniaturization of transistors which increases their sensitivity to SEE. Although, technological and architectural improvements as well as software strategies have been developed to mitigate SEE, the implementation of additional hardware and software protections involves performance degradation and increase of power consumption. It is thus mandatory to evaluate the sensitivity to SEE of multi and many-core devices for validating their applicability in harsh environments, or for applications where the reliability is a concern.

There is a rising interest of using COTS multi and many-cores for avionics and safety-critical applications due to their low cost and time saving compared to dedicated complex solutions. Nevertheless, the choice of these components is still carried out in an ad hoc manner, which produces problems for accurate cost and reliability estimates. Hence, the selection of these COTS devices should be based on the evaluation of the impact of the radiation on their reliability by means of the estimation of their failure rate.

In this work it was proposed a general approach for evaluating the SEE sensitivity of applications implemented in multi-core and many-core processors. This was achieved by applying the principles of the CEU approach developed at TIMA laboratories in previous works, considering the technological and architectural evolution of multi and many-core architectures with respect to single processors. The soft error-rate was used as a metric for such evaluation. The prediction of the error rate was accomplished by combining the error rate of the application issued from fault injection campaigns, with the worst-case sensitivity of the device obtained from radiation ground testing. In order to validate the prediction approach, it was required to compare the predicted and measured error rates. The measured result was obtained by exposing the targeted device to neutron radiation while executing the desired application. Fault injection campaigns were devoted to inject faults in memory cells and

accessible registers benefiting of the multiplicity of cores for using one of them as a fault injector while the others execute the selected application. Radiation experiments were conducted with 14 *Mev* neutrons at GENEPI2 facility. Three different COTS devices have been evaluated in the context of this thesis. The first one was the Freescale P2041 quad-core processor manufactured in 45nm SOI technology. The second one was the Kalray MPPA-256 many-core manufactured in 28nm CMOS technology. The third one was the Adapteva Epiphany E16G301 microprocessor manufactured in 65nm CMOS technology.

7.1 Concluding remarks

Obtained results from the evaluation of the P2041 multi-core demonstrate that fault injection allows identifying vulnerabilities in the application, and improving the programming strategy for reducing the impact of faults in the results. From the static test, it was confirmed that SOI process technology is more robust than traditional bulk CMOS. On the other hand, dynamic tests have demonstrated that in spite of the parity and ECC protection mechanisms, there were errors in the result of the application caused by MBUs in the *address tags* and data array. Finally, results show an underestimation of the predicted error-rate, since not all sensitive zones were targeted during the static test and fault injection campaigns. Furthermore, the implementation of ECC and parity in the device's cache memories may affect the prediction.

From the evaluation of the Epiphany E16G301 multi-core processor, it can be seen that the proposed approach was effective for predicting the application error-rate. The fact that this device does not implement protection mechanisms has allowed a good estimation of the error-rate, confirming that protection mechanisms affect the testing and error-rate prediction (LaBel, 2005). During the dynamic radiation test, input matrices were also checked to identify silent faults. It was done in order to obtain the experimental error-rate of the application which has a good correlation with the error rate obtained from fault injection.

The evaluation of the MPPA-256 many-core processor shows that both, ECC and interleaving implemented in the SMEMs of the clusters are very effective to mitigate SEU type errors, since all the detected SEUs in the SMEMs were corrected during the static test. In addition, dynamic tests have demonstrated that by enabling the cache memories it is possible to gain in application performance without a reliability penalty, since cache memories implement an effective parity protection. Regarding the radiation experimental results, the prediction of the error-rate was based only on the registers' contribution since they do not implement any protection mechanism. Despite the complexity of this many-core processor, the prediction of the error-rate has a small underestimation that confirms the applicability of the approach to these devices. The possible reasons for this underestimation are: only accessible

registers were targeted, communication infrastructure was not targeted, protection mechanisms may affect the error-rate prediction,

The overall results presented in this thesis confirm that chip manufacturing process plays a preponderant role in the dependability of the device. A FIT comparison of the studied devices shows that the P2041 multi-core is the most reliable device since it is built in SOI technology. Nevertheless, the difference with the MPPA-256 built in 28nm CMOS is not so ample. In fact, if the FIT/Mb is considered, the MPPA-256 is the most reliable device. Both devices implement ECC and parity in their internal memories. However, the effectiveness of the MPPA-256 is greater than the efficiency of the P2041 due to the implementation of interleaving in its shared memories. Consequently, the selection of the appropriate device should be done based on the requirements of the application to be implemented (e.g., available memory space for code and data, number of cores working in parallel etc.), and a cost-benefit analysis.

Concerning the Epiphany multi-core, it has a failure rate about 11 and 7 times greater than the one of the P2041 and the MPPA-256 respectively. Due to the fact that better manufacturing technologies as well as hardware protections increase significantly the cost of the device, devices such as the Epiphany E16G306 multi-core could be considered for embedded systems performing non-critical avionic applications. In fact, the NASA's report: "Intelligent Hardware-Enable Sensor and Software Safety and Health Management for Autonomous UAS" states that the Epiphany multi-core can be used for several terrestrial and even avionic applications.

The fact that a many-core processor built on 28nm CMOS has a similar FIT than a multi-core 45nm SOI is very promising to widespread its use on embedded domain. First, because the mitigation techniques implemented on the device have decreased the reliability gap between both technologies related to miniaturization and manufacturing process (CMOS vs SOI). Second, it supports the possibility of using COTS devices in critical-embedded system due to the huge processing capability and the large internal memory capacity. Both characteristics allow overcoming the main problem of overhead caused by the implementation of hardware fault-tolerant techniques. Therefore, the use of this device makes feasible the implementation of software redundancy techniques to improve the system reliability by masking other SEEs consequences that were not mitigated by the protection mechanisms. Lastly, the high cost of SOI devices compared to CMOS devices boosts the use of the latter.

For selecting the appropriate device intended to be implemented in a system operating in harsh radiation environment, it is imperative to evaluate the SEE sensitivity of the candidates, and then to establish a trade-off between costs and reliability depending on the application and

the operating environment. However, the evaluation based on dynamic radiation tests is costly in terms of time and money. For this reason, a prediction approach is required. Furthermore, the widespread use of multi/many-core processors in embedded systems requires a prediction error-rate suitable for these devices. To the author's knowledge, this work presents for the first time an approach devoted to predict the error-rate of an application executing on multi/many-core processors. In spite of the underestimation of the predicted error-rates, this approach provides useful results that can be considered as a preliminary validation of COTS multi/many-cores.

There are several aspects that make difficult the SEE testing on multi/many-core processors and affect the error-rate prediction. One of them is the complexity of their architecture which integrates different components and functions on the same chip. In the approach developed in this thesis, the error-rate prediction of a system is presented as the sum of the individual contributions of error-rate prediction by each memory-cell component. To obtain the individual contribution is advisable to isolate functionalities. However, this is a not trivial task on these kinds of devices since isolation cannot be totally accomplished. This fact causes difficulties for identifying the zone where faults were produced. Moreover, to improve the effectiveness of the prediction, it is convenient to extend this approach to other device's components such as the communication infrastructure. Another aspect is the complex packaging and multi-layered construction of circuits which produce problems when testing in radiation facilities due to beam energy limitations. Finally, the implementation of error detection and correction mechanisms as well as the lack of experimental data (due to radiation facility costs and availability) also influence in the accuracy of the error-rate prediction approach.

7.2 Future directions

The current work has presented a first insight in the vast study of the sensitivity to radiation of multi-core and many-core processors. For continuing with this work, the following topics can be explored:

- Validation of the proposed approach using different programming models: The programming model may substantially affect the sensitivity of the device because of the used resources. For this reason, it is meaningful to validate the approach using Posix and OpenMP.
- Validation of a real space application: It can be done in the context of industrial and academic cooperation between TIMA and industrial partners such as Thales Alenia Space, or academics partners such as "Centre Spatial Universitaire de Grenoble".

- Evaluation of the communication infrastructure: Given the increasing number of cores in emerging many-cores processors, the communication network are changing implementing NoCs and combining them with buses for interconnecting processing cores and clusters. It implies having more control and data registers to be targeted by fault injection and radiation tests.
- Validation of the prediction approach using heavy-ions: For considering COTS multi/many-cores for space application it is mandatory to expose them to heavy-ions. However, it is important to consider equipment and hardware constraints such as testing in vacuum chamber and silicon thinning of the device under test.
- Application of redundancy techniques: This prediction approach can be combined with the implementation of redundant applications aiming at increasing the reliability of the device.

Publications

Journals

Ramos, P., Vargas, V., Baylac, M., Villa, F., Rey, S., Clemente, J. A., Zergainoh, N. E., Méhaut, J. F. and Velazco, R. (2016) "Evaluating the SEE Sensitivity of a 45 nm SOI Multi-Core Processor Due to 14 MeV Neutrons", IEEE Trans. Nucl. Sci., vol. 63, no 4, August, pp. 2193–2200.

Vargas, V., **Ramos, P**., Ray, V., Jalier, C., Stevens, R., Dupont de Dinechin, B., Baylac, M., Villa, F., Rey, S., Zergainoh, N. E., Mehaut, J. F. and Velazco, R. (2017) "Radiation Experiments on a 28nm Single-Chip Many-core Processor and SEU error-rate prediction", IEEE Trans. Nucl. Sci., vol. 64, no 1, pp. 483-490.

Clemente, J. A., Franco, F., Villa, F., Baylac, M., **Ramos, P**., Vargas, V., Mecha, H., Agapito, J. and Velazco, R. (2016) "Single Events in a COTS Soft-Error Free SRAM at Low Bias Voltage Induced by 15-MeV Neutrons", IEEE Trans. Nucl. Sci., vol. 63, no 4, August, pp. 2072–2079.

Conferences

Ramos, P., Vargas, V., Baylac, M., Villa, F., Rey, S., Clemente, J. A., Zergainoh, N. E., Méhaut, J. F. and Velazco R. (2015) "Sensitivity to Neutron Radiation of a 45 nm SOI Multi-Core Processor", In Proc. Radiation and its Effects on Components and Systems (RADECS), September, pp. 135-138.

Ramos, P., Vargas, V., Baylac, M., Villa, F., Rey, S., Zergainoh, N. E. and Velazco R. (2017) "Error-rate prediction for applications implemented in Multi-core and Many-core processors", Accepted for publication in Proc. Radiation and its Effects on Components and Systems (RADECS 2017), October 2-6 Geneva – Switzerland.

Vargas, V., **Ramos, P**., Velazco, R., Mehaut, J. F. and Zergainoh, N. E. (2015) "Evaluating SEU fault-injection on parallel applications implemented on multicore processors" In Proc. 6th Latin American Symposium on Circuits & Systems (LASCAS) February, DOI:10.1109/LASCAS.2015.7250449.

Vargas, V., **Ramos, P**., Mansour, W., Velazco, R., Zergainoh, N. E. and Mehaut, J. F. (2014) “Preliminary results of SEU fault-injection on multicore processors in AMP mode”, In Proc. 20th International On-Line Testing Symposium (IOLTS), September, pp. 194–197.

Mansour, W., **Ramos, P**., Ayoubi, R. and Velazco R. (2014) “SEU fault-injection at system level: method, tools and preliminary results”. 15th Latin American Test Workshop – LATW, pp. 1-5.

Clemente, J. A., Franco, F., Villa, F., Baylac, M., **Ramos, P**., Vargas, V., Mecha, H., Agapito, J. and Velazco, R. (2015) “Neutron-Induced Single Events in a COTS Soft-Error Free SRAM at Low Bias Voltage”, In Proceedings of Radiation and its Effects on Components and Systems (RADECS), September, pp 162-165.

Villa, F., Baylac, M., Rey, S., Rossetto, O., Mansour, W., **Ramos, P**., Velazco, R. and Hubert, G. (2014) “Accelerator-Based Neutron Irradiation of Integrated Circuits at GENEPI2 (France),” in Proc. Radiation Effects Data Workshop, July, pp. 1–5.

Kchaou, A., El Hadj, Y., Tourki, R., Bouesse, F., **Ramos, P**., and Velazco, R. (2016) “A deep analysis of SEU consequences in the internal memory of LEON3 processor”, in Proc. 17th Latin American Test Symposium (LATS), April, pp. 178.

Vargas, V., **Ramos, P**., Ray, V., Jalier, C., D. de Dinechin, B., Baylac, M., Villa, F., Rey, S., Minguez, B. and Velazco, R. (2016) “ First Results of Radiation Experiments on a 28nm Single-Chip Many-core Processor”, IEEE 53rd Nuclear and Space Radiation Effects Conference (NSREC), July 11-15, Portland, Oregon.

Bibliography

(Adapteva, 2009) ADAPTEVA. (2009) Parallella1.x Reference Manual Rev 14.09.09 [Online], Available: https://www.parallella.org/docs/parallella_manual.pdf [23 December 2016]

(Adapteva, 2011a) ADAPTEVA. (2011a) E16G301 Datasheet Rev 14.03.11. [Online], Available: http://www.adapteva.com/docs/e16g301_datasheet.pdf [23 December 2016]

(Adapteva, 2013) ADAPTEVA. (2013) Epiphany SDK Rev 5.13.09.10. [Online], Available: http://www.adapteva.com/docs/epiphany_sdk_ref.pdf [23 December 2016]

(Adapteva, 2011b) ADAPTEVA. (2011b) Epiphany Architecture Reference Rev 14.03.11. [Online], Available: http://www.adapteva.com/docs/epiphany_arch_ref.pdf [23 December 2016]

(Aidemark, 2001) Aidemark, J. et all. (2001) "GOOFI: Generic Object-Oriented Fault Injection tool", in proceedings of International Conference on Dependable Systems and Networks, Gothenburg, Sweden, July, pp. 83-88.

(Arlat, 1990) Arlat, J.et al. (1990) "Fault Injection for Dependability Validation: A methodology and Some Applications," IEEE Trans. On Soft. Eng. Vol. 16, No 2, pp. 166-182.

(Autran, 2014) Autran, J., Munteanu, P., Roche, P. and Gasiot, G. (2014). "Real-time Soft- Error Rate Measurements: A Review," Microelectronics Reliability, vol. 54, February, pp. 1455–1476.

(Baggio, 2005) Baggio, V. et all. (2005) "Neutron and Proton-induced single event upsets in advanced commercial fully depleted SOI SRAMs", IEEE Trans. Nucl. Sci., Vol 52, Issue 6, August, pp. 2319-2325.

(Ballard, 2012) Ballard, G., Demmel, J., Holtz, O., Lipshitz, B. and Schwartz, O. (2012) "Communication-Optimal Parallel Algorithm for Strassen's Matrix Multiplication", In Proc. Of 24th anual ACM symposium on Parallelism in algorithms and architectures, June, pp 193-204.

(Baldini, 2000) Baldini, A., Benso, A., Chiusano, S. and Prinetto, p. (2000) “BOND: An interposition agents based fault injector for Windows NT", *Proc.* IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Yamanashi, pp. 387-395.

(Barth, 2003) Barth, J.L. (2003) “Space, atmospheric, and terrestrial radiation environments”, NASA Goddard Space Flight Center, IEEE transactions on Nuclear Sciences, Vol. 50, No 3, June, pp 466-482.

(Benso, 1998) Benso, A.et all. (1998) “EXFI: A low cost Fault Injection System for embedded Microprocessors-based Boards”, ACM, Transactions on Design Automation of Electronic Systems, Vol 5, No 4, pp 626-634.

(Benso, 2003) Benso, A. and Prinetto, P. (2003) Fault Injection techniques and tools for embedded systems reliability evaluation, USA: Kluwer Academic.

(Bieber, 2012) Bieber P. et all. (2012) “New challenges for future avionic architectures”, AerospaceLab, 2012, pp 1-10.

(Blackmore, 2003) Blackmore, E. W., Dodd, P. E. and Shaneyfelt, M. R. (2003) “Improved capabilities for proton and neutrón irradiations at TRIUMF”, In Proc. IEEE Rad. Effects Data Workshop, pp. 149-155.

(Buchner, 1987) Buchner, S. P. et all. (1987) “Laser Simulation of Single Event Upsets”, IEEE Trans. Nucl. Sci., vol. 34, no. 6, pp. 1227-1233.

(Cabanas-Holmen, 2011) Cabanas-Holmen, M. et all. (2011). “Predicting the Single-Event Error Rate of a Radiation Hardened by design Microprocessor”, IEEE Trans. Nucl. Sci., Vol. 58, NO. 6, December, pp. 2726-2733.

(CAST, 2016) Certification Authorities Software Team (CAST). (2016) “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-32A.pdf

(CERN, 2015) Conseil Européen pour la Recherche Nucléaire. (2015) “Accelerator Complex” [Online], Available: https://home.cern/about/accelerators

(CERN, 2016) Conseil Européen pour la Recherche Nucléaire. (2016) “Cosmic Rays: Particles from outer space”, [Online], Available: http://home.cern/about/physics/cosmic-rays-particles-outer-space

(Chee, 2000) A. Chee et al. (2000). ‘’Potential doses to passengers and crew of supersonic transports’’, Health Phys., Vol. 79, pp. 547.

(Cleland, 2005) Cleland, M. R. (2005) Industrial applications of electron accelerators, USA: IBA Technology Group.

(Clemente, 2016) Clemente, J. A., Franco, F., Villa, F., Baylac, M., Ramos, P., Vargas, V., Mecha, H., Agapito, J. and Velazco, R. (2016) ”Single Events in a COTS Soft-Error Free SRAM at Low Bias Voltage Induced by 15-MeV Neutrons”, IEEE Trans. Nucl. Sci., vol. 63, no 4, August, pp. 2072–2079.

(Clemente, 2015) Clemente, J. A., Franco, F., Villa, F., Baylac, M., Ramos, P., Vargas, V., Mecha, H., Agapito, J. and Velazco, R. (2015) “Neutron-Induced Single Events in a COTS Soft-Error Free SRAM at Low Bias Voltage”, In Proceedings of Radiation and its Effects on Components and Systems (RADECS), September, pp 162-165.

(Costa, 2003) Costa, D. et all. (2003) “XCEPTION™: A Software Implemented Fault Injection Tool”, in Benzo, A. and Pinetto, P. (Ed.) Fault Injection Techniques and tools For Embedded Systems Reliability Evaluation, USA: Springer, pp 125-139.

(Ferrel, 2014) Ferrel T. and Ferrel D. (2014) ¨RTCA DO-178B/EUROCAE ED-12B¨, Digital Avionics Handbook, Third Edition, pp 195-206.

(Freescale, 2011) Freescale. (2011) QorIQ Multicore processor Development, “P2041 Reference Design Board”. Document number: P2041RDBFS/REV 1 [Online], Available:http://www.nxp.com/assets/documents/data/en/fact-sheets/P2041RDBFS.pdf [23 December 2016].

(Freescale, 2012) Freescale. (2012) QNX Software Systems “Running AMP, SMP or BMP Mode for Multicore Embedded Systems”, [Online], Available: http://cache.freescale.com/files/32bit/doc/brochure/PWRARBYNDBITSRAS.pdf [26 December 2016].

(Freescale, 2015a) Freescale. (2015a) “P2040/P2041 QorIQ Integrated Multicore Communication Processor Family Reference Manual Rev. 3.” [Online]. Available: http://cache.freescale.com/files/32bit/doc/ref_ manual/ P3041RM.pdf. [23 December 2016].

(Freescale, 2015b) Freescale. (2015b) “e500mc Core Reference Manual, Rev. 3.” [Online], Available:http://cache.freescale.com/files/32bit/doc/ref_ manual/E500MCRM.pdf [23 December 2016]

(Gaillard, 2011) Gaillard, R. (2011) “Single Event Effects: Mechanisms and Classification”, In Nicolaidis, M. (Ed.) Soft errors in modern electronic Systems, USA: Springer, chapter 2.

(Gaillardin, 2013) Gaillardin, M. et all. (2013) “Radiation Effects in Advanced SOI Devices: New Insights into Total Ionizing Dose and Single-Event-Effects”, SOI-3D- Subthreshold Microelectronics Technology Unified Conference (S3S).

(Gasiot, 2002) Gasiot, G., Ferlet-Cavrois, V., Baggio, J., Roche, P., Flatresse, P., Guyot, A., Morel, P., Bersillon, O. and du Port de Pontcharra, J. (2002) “SEU Sensitivity of Bulk and SOI Technologies to 14-MeV Neutrons”. IEEE Trans. Nucl. Sci., vol. 49, no 6, Dec., pp.3032–3037.

(Girbal, 2015) Girbal, S., Pérez, D. G., Le Rhun, J., Faugère, M., Pagetti, C. and Durrieu, G.(2015) "A complete toolchain for an interference-free deployment of avionic applications on multi-core systems," 2015 .IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), Prague, pp. 7A2-1-7A2-14.

(Guertin, 2012) Guertin, S. (2012) “Initial SEE Test of Maestro Microprocessor”, USA: Jet Propulsion Laboratory, California Institute of technology.

(Han, 1995) Han, S.et all. (1995) “DOCTOR: An Integrated Software Fault Injection Environment ”, International Computer Performance and Dependability Symposium, Erlangen, Germany, pp. 204-213.

(Hands, 2011) Hands, A., Morris, P., Dyer, C., Ryden, K. and Truscott, P. (2011) “Single Event Effects in Power MOSFETs and SRAMs Due to 3MeV, 14 MeV and Fission Neutrons”, IEEE Trans. Nucl. Sci., vol. 58, June, pp. 952–959.

(Hands, 2012) Hands, A., Morris, P., Ryden, K. and Dyer, C. (2012) “Large-Scale Multiple Cell Upsets in 90 nm Commercial SRAMs During Neutron Irradiation”, IEEE Trans. Nucl. Sci., vol. 59, pp. 2824–2830.

(Höller, 2014) Höller, A. et all. (2014) “FIES: A Fault Injection Framework for the Evaluation of Self-Tests for COTS-Based Safety-Critical Systems”, 15th International Microprocessor Test and Verification Workshop, pp. 105-110.

(Houssany, 2012) Houssany, S. et all. (2012) “Microprocessor Soft Error Rate Prediction Based on Cache Memory Analysis”, IEEE Trans. Nucl. Sci., vol 59, no 4, August, pp. 980-987.

(Jesd89A, 2006) Jesd89A. (2006) “Measurement and Reporting of Alpha Particle and terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices”, JEDEC Solid State Technology Association. [Online], Available: http://www.jedec.org/sites/default/files/docs/jesd89a.pdf [23 December 2016]

(Kalray, 2016) Kalray. (2016) MPPA-256 Boostan Cluster and I/O subsystem Architecture, France: Kalray.

(Kanawati, 2002) Kanawati, G. A.et all. (2002) “FERRARI: A Flexible sotware-based fault and error injection system”, IEEE Trans. On Comp., vol 44, no 2, August, pp. 248-260.

(Kooli, 2014) Kooli, M. and Di Natale, G. (2014) “A Survey on Simulation-based Fault Injection Tools for Complex Systems”, 9th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS).

(LaBel, 2005) LaBel, K. and Cohn, L. (2005) Radiation Testing and Evaluation Issues for Modern Integrated Circuits, NASA/GSFC Code 561.4.

(Lambert, 2005) Lambert, D. et all. (2005) “Neutron Induced SEU in SRAMs: Simulations with n-Si and n-O interactions”, IEEE Trans. Nucl. Sci., vol. 52, No 6, December, pp. 2332–2339.

(LaPedus, 2016) LaPedus, M. (2016) “Bulk CMOS VS FD-SOI”, Semiconductor Engineering, [Online], available: http://semiengineering.com/bulk-cmos-versus-fd-soi/ [23 December 2016].

(Lemoine, 2006) Lemoine, J. M. and Capdeville, H. (2006) “A corrective model for Jason-1 Doris Doppler data in relation to the south Atlantic Anomaly”, Journal Geod., vol. 80, no. 8, pp. 507-523.

(Mansour, 2014) Mansour, W., Ramos, P., Ayoubi, R. and Velazco R. (2014) “SEU fault-injection at system level: method, tools and preliminary results”. 15th Latin American Test Workshop – LATW, pp. 1-5.

(Marinescu, 2009) Marinescu, P. and Candea, G. (2009) “LFI: A Practical and General Library-Level Fault Injector”, in Proceedings on the International Conference on Dependable Systems and Networks, June, pp. 379 – 388.

(Martins, 2002) Martins, E. (2002) “Jaca: a reflective fault injection tool based on patterns”, in Proceedings of International Conference on Dependable Systems and Networks, June, pp. 438-487.

(May, 1979) May, T.C. and Woods, M.H. (1979) "Alpha-particle-induced soft errors in dynamic memories", IEEE Trans Electron Devices ED-26, pp 2-9.

(McConnell, 2016) McConnell, J. (2016) “Solar features”, NIAAS, [Online], Available: http://www.eaas.co.uk/cms/index.php?option=com_content&view=article&id=14:solar-features-by-john-mcconnell-fras&catid=5:learning-zone&Itemid=8, [23 December 2016]

(Microsemi, 2011) Microsemi (2011) Neutron-Induced Single Event Upset (SEU) FAQ, CA: Microsemi Corporation. [Online], Available: http://www.microsemi.com/document-portal/doc_view/130760-neutron-seu-faq. [26 December 2016].

(Miller, 2013) Miller, F., Weulersse, C., Carriere, T., Guibbaud, N., Morand, S. and Gaillard, R. (2013) “Investigation of 14 MeV Neutron Capabilities for SEU Hardness Evaluation,” IEEE Trans. Nucl. Sci., vol. 60, August, pp. 2789–2796.

(Mukherjee, 2003) Mukherjee S. et all. (2003) “A systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Mcroprocessor”, Proceedings of the 36th International Symposium on Microarchitectures (MICRO-36’03), pp. 29-40.

(Mukherjee, 2008) Mukherjee, S. (2008) Architecture Design for Soft Errors, USA: Elsevier, chapter 1, pp. 25-26.

(Oliveira, 2014) Oliveira, D. et all. (2014) “Modern GPUs Radiation Sensitivity Evaluation and Mitigation Through Duplication With Comparison”, IEEE Trans. Nucl. Sci., vol 61, no 6, December, pp. 3115-3122.

(Peronnard, 2008) Peronnard, P., Ecoffet, R., Pignol, M., Bellin, D. and Velazco, R. (2008) “Predicting the SEU Error Rate through Fault Injection for a Complex Microprocessor,” in Proc. 2008 IEEE International Symposium on Industrial Electronics, September, pp. 2288–2292.

(Peronnard, 2009a) Peronnard, P., Velazco, R. and Hubert, G. (2009a) “Real-Life SEU Experiments on 90 nm SRAMs in Atmospheric Environment: Measures Versus Predictions done by means of MUSCA SEP^3 platform” ,” IEEE Trans. Nucl. Sci., vol. 56, December, pp. 3450-3455.

(Peronnard, 2009b) Peronnard, P. (2009b) "Méthodes et outils pour l'évaluation de la sensibilité de circuits intégrés avancés face aux radiations naturelles", Micro et nanotechnologies/Microélectronique, Université Joseph-Fourier - Grenoble I.

(Potyra, 2007) Potyra, S.et all. (2007) “Evaluating fault tolerant systems designs using Faumachine”, in Proceedings of the 2007 Workshop on Engineering Fault Tolerant Systems EFTS, New York, NY, USA.

(Pouget, 2001) Pouget, V. et al. (2001) “Theorical Investigation on an Equivalent Laser LET”, Microelectronics Reliability, vol. 41, pp. 1513-1518.

(Ramos, 2015) Ramos, P., Vargas, V., Baylac, M., Villa, F., Rey, S., Clemente, J. A., Zergainoh, N. E., Méhaut, J. F. and Velazco, R. (2015) “Sensitivity to Neutron Radiation of a 45 nm SOI Multi-Core Processor”, In Proc. Radiation and its Effects on Components and Systems (RADECS), September, pp. 135-138.

(Ramos, 2016) Ramos, P., Vargas, V., Baylac, M., Villa, F., Rey, S., Clemente, J. A., Zergainoh, N. E., Méhaut, J. F. and Velazco, R. (2016) “Evaluating the SEE Sensitivity of a 45 nm SOI Multi-Core Processor Due to 14 MeV Neutrons”, IEEE Trans. Nucl. Sci., vol. 63, No 4, August, pp. 2193–2200.

(Ratti, 2011) Ratti, L. (2011) Total dose effects in electronic devices and circuits, Pavi, Italia: Universita degli studi di Pavia and INFN Pavia.

(Rezgui, 2001) Rezgui, S., Velazco, R., Ecoffet, R., Rodriguez, S. and Mingo, J. (2001)"Estimating Error Rates in Processor-Based Architectures," IEEE Trans. Nucl. Sci., vol. 48, December, pp. 1680–1687.

(Rodriguez, 1999) Rodríguez M., Salles F., Fabre JC., Arlat J. (1999) MAFALDA: Microkernel Assessment by Fault Injection and Design Aid. In: Hlavička J., Maehle E., Pataricza A. (eds) Dependable Computing EDCC-3. EDCC 1999.

(Rozier, 2015) Rozier, K., Schumman, j. and Ippolito, C. (2015) Intelligent Hardware-Enable Sensor and Software Safety and Health Management for Autonomous UAS, NASA/TM-2015-218817, [Online], Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150021506.pdf May, [26 December 2016]

(Saidi, 2015) Saidi, S., Ernst, R ., Uhrig, S., Theiling, H. and Dupont de Dinechin. B. (2015) "The shift to multicores in real-time and safety-critical systems". In Proc. Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Oct., pp 220–229.

(Santini, 2014) Santini, T. Rech, P. Nazar, G. Carro, L. and Rech Wagner, F. (2014) "Reducing Embedded Software Radiation-Induced Failures Through Cache Memories," in Proc. European Test Symposium, May, pp. 1-6.

(Segall, 1988) Segall, Z. et all. (1988) "FIAT- fault injection based automated testing environment", 118th International Symposium on Fault-Tolerant Computing, pp. 102-107.

(Shooman, 2002) Shooman, M. (2002). Reliability of Computer Systems and Networks: Fault Tolerance,Analysis,and Design. New York: John Wiley & Sons, Inc. pp. 11.

(Sohn, 2000) Sohn, J.H. (2000) The effects of single event upsets in avionics systems, Iowa, USA: M.S. thesis, lowa State Univ., May.

(Stolt, 2012) Stolt, S.S. and Normand, E. (2012) "A Multicore Server SEE Cross Section Model", IEEE Trans. Nucl. Sci., vol. 59, December, pp. 2803–2810.

(Taber, 1997) Taber, A. et al. (1997) "Single event upset in avionics", IEEE Trans. on Nuc. Sci, vol NS-40, no. 2, pp. 120-126.

(Tang, 2011) Tang, L. et all. (2011) "Characterizing the L1 Data Cache vulnerability to transient Errors in Chip-Multiprocessors", IEEE Computer Society Annual Symposium on VLSI, August, pp. 266-271.

(Tavernier, 2010) Tavernier, S. (2010) Experimental Techniques in Nuclear and Particle Physics, Berlin: Springer-Verlag.

(Taylor, 2015) Taylor, N. (2015) “What is Solar Wind?”, SPACE.COM, [Online], Available: http://www.space.com/11506-space-weather-sunspots-solar-flares-coronal-mass-ejections.html, [26 December 2016].

(Tsai, 1994) Tsai, T. and Ravishankar, I. (1994) “FTAPE: A fault injection tool to measure fault tolerance”, CA, USA: NASA Technical Documents.

(Vajda, 2011) Vajda, A.(2011) “Programming Many-Core Chips”, DOI 10.1007/978-1-4419-9739-5_2, Springer Science+Business Media, LLC ,pp. 9- 42.

(Vargas, 2014) Vargas, V., Ramos, P., Mansour, W., Velazco, R., Zergainoh, N. E. and Mehaut, J. F. (2014) “Preliminary results of SEU fault-injection on multicore processors in AMP mode”, In Proc. 20th International On-Line Testing Symposium (IOLTS), September, pp. 194–197.

(Vargas, 2015) Vargas, V., Ramos, P., Mansour, W., Velazco, R., Mehaut, J. F. and Zergainoh, N. E. (2015) “Evaluating SEU fault-injection on parallel applications implemented on multicore processors”, In Proc. 6th Latin American Symposium on Circuits & Systems (LASCAS) February.

(Vargas, 2017) Vargas, V., Ramos, P., Ray, V., Jalier, C., Stevens, R., Dupont de Dinechin, B., Baylac, M., Villa, F., Rey, S., Zergainoh, N. E., Mehaut, J. F. and Velazco, R. (2017) “Radiation Experiments on a 28nm Single-Chip Many-core Processor and SEU error-rate prediction”, IEEE Trans. Nucl. Sci., Early Access Articles.

(Velazco, 2000) Velazco, R., Rezgui, S. and Ecoffet, R. (2000) “Predicting Error Rate for Microprocessor-Based Digital Architectures through C.E.U. (Code Emulating Upsets) Injection,” IEEE Trans. Nucl. Sci., vol. 47, December, pp. 2405–2411.

(Velazco, 2010) Velazco, R., Foucard, G. and Peronnard, P. (2010) “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, December, pp. 3500–3505.

(Velazco, 2014) Velazco, R., Clemente, J.A., Hubert, G., Mansour, W., Palomar, C., Franco, F.J., Baylac, M., Rey, S., Rosetto, O. and Villa, F. (2014) “Evidence of the Robustness of a COTS Soft-Error Free SRAM to Neutron Radiation,” IEEE Trans. Nucl. Sci., vol. 61, December, pp. 3103–3108.

(Villa, 2014) Villa, F., Baylac, M., Rey, S., Rossetto, O., Mansour, W., Ramos, P., Velazco, R. and Hubert, G. (2014) “Accelerator-Based Neutron Irradiation of Integrated Circuits at GENEPI2 (France),” in Proc. Radiation Effects Data Workshop, July, pp. 1–5.

(Villalpando, 2011) Villalpando, C., Rennels, D., Some, R. and Cabanas-Holmen, M. (2011) "Reliable Multicore Processor for NASA Space Missions", Proc. Aerospace Conference, DOI:10.1109/AERO.2011.5747447.

(Wang, 2008) Wang, F. and Agrawal, V. (2008) "Probabilistic Soft Error Rate Estimations from Statistical SEU Parameters", in Proceeding of 17th IEEE North Atlantic Test Workshop, May, pp. 901-914.

(Wikipedia, 2016a) Wikipedia. (2016a) "Laser" [Online], Available: https://en.wikipedia.org/wiki/Laser [26 December 2016].

(Wikipedia, 2016b) Wikipedia. (2016b) "CPU_cache" [Online], Available: https://en.wikipedia.org/wiki/CPU_cache [26 December 2016].

(Wong, 2011) Wong, H., Vetz, V. and Rose, J. (2011) "Comparing FPGA vs. Custom CMOS and the Impact on Processor Microarchitecture", FPGA'11 Proc. of the 19th ACM/SIGDA international symposium on Field programmable gate arrays, February, pp 5-14.

(Wulf, 2016) Wulf, M. (2016) "Cosmic Ray", Encyclopaedia Britannica, [Online], Available: https://global.britannica.com/topic/cosmic-ray, [26 December 2016].

(Xilinx, 2015) XILINX, (2015) "Virtex-5 Family overview DS-100 V5.1" [Online], Available: https://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, [26 December 2016]

(Xilinx, 2016) XILINX, (2016) "Device Reliability Report UG116 V10.5" [Online], Available: https://www.xilinx.com/support/documentation/user_guides/ug116.pdf , [26 December 2016].

(Ye, 2004) Ye, F. and Kelly, T. (2004) "COTS Product Selection for Safety Critical Systems", Third International Conference, ICCBSS, Redondo Beach, CA. USA, February, pp 53-62.

(Ziade, 2004) Ziade, H., Ayobi, R. and Velazco, R. (2004) "A Survey on Fault Injection Techniques", The International Arab Journal of Information Technology, Vol 1, no 2, July, pp. 1-6.

Résumé de la thèse en français

i. Introduction

Contexte général et motivation

Les processeurs *multi-cœur* et *many-cœur* sont une solution prometteuse pour atteindre la fiabilité, la haute performance et la faible consommation d'énergie. L'utilisation de ces dispositifs devient très attrayante en raison de leur énorme capacité de traitement combinée à leur capacité de redondance intrinsèque, ce qui les rend idéales pour la mise en œuvre d'applications hautes performances dans des domaines scientifiques, commerciaux grand public, et de sûreté de fonctionnement. Par conséquent, il existe plusieurs projets internationaux qui travaillent à valider l'utilisation de processeurs multi et many-cœur dans des systèmes embarqués critiques. En fait, les industries spatiales et avioniques sont intéressées à valider l'utilisation de ces dispositifs pour intégrer les fonctions d'un système entier dans une seule puce et pour accroître la fiabilité de leurs applications (CAST, 2014). Par exemple, la nouvelle tendance de l'architecture des systèmes avioniques s'appuie sur l'IMA (Integrated Modular Avionics) au lieu de l'architecture fédérée classique. La principale différence entre les deux architectures est que dans le système fédéré, chaque système dispose de ressources privées alors que dans l'IMA, les ressources peuvent être partagées. L'un des défis les plus importants pour les architectures IMA est l'intégration de processeurs multi-cœurs commerciaux (Bieber, 2012). L'utilisation de processeurs multi-cœurs Commercial-Of-The-Shelf (COTS) est pratique en raison des problèmes de budget et de disponibilité. Néanmoins, la sélection de ces composantes crée des complications pour un coût réaliste, une estimation de l'effort, et une certification contre les erreurs (Ye, 2004).

En fait, l'une des principales préoccupations de la certification pour les systèmes critiques embarqués est la sensibilité au rayonnement des composants électroniques. Ce rayonnement peut entraîner des défaillances transitoires et permanentes, appelées Single Event Effects (SEE), qui sont les conséquences de l'impact d'une particule unique dans une zone sensible du circuit. Une forme représentative de SEE est le Single Event Upset (SEU), dont l'énergie déposée provoque le basculement d'un bit d'une cellule mémoire. Les SEU sont un problème critique à prendre en compte car ils peuvent conduire à la modification aléatoire en temps et en l'emplacement du contenu d'une cellule mémoire avec des conséquences inattendues au niveau de l'application.

L’apparition de SEE dépend principalement de la technologie du dispositif et de l'environnement radiatif dans lequel le circuit est destiné à fonctionner. En effet, avoir des dispositifs plus denses et complexes implique une mise à l'échelle de la technologie qui augmente leur vulnérabilité aux effets du rayonnement naturel. Pour cette raison, les fabricants sont constamment à la recherche de nouvelles méthodes pour améliorer le processus de la technologie afin de réduire les conséquences des SEEs. Silicon-On-Insulator (SOI) est un exemple clair de ces améliorations faites face aux inconvénients traditionnels du bulk CMOS. Les techniques de type Radiation Hardened By Design (RHBD) sont également utilisées pour atténuer les effets des SEUs. Par exemple, la mise en œuvre de codes de correction d'erreurs (ECC) et la parité pour protéger la mémoire interne des processeurs est pratique mais pas suffisante en présence de multiple bits (MBU). Une autre technique RHBD bien connue est la Triple Modular Redundancy (TMR) qui améliore significativement la fiabilité du système. Cependant, la mise en œuvre de composants matériels et de mécanismes de protection additionnels implique l'introduction d'une zone supplémentaire et d'un surcout dans l’application, ce qui peut entraîner une plus grande consommation d'énergie et une dégradation de la performance. Il est donc essentiel d'ajouter une tolérance aux fautes au système avec un minimum overhead.

Dans ce contexte, les processeurs multi- cœur et many-cœur conviennent parfaitement à la mise en œuvre de techniques de tolérance aux fautes basées sur leurs capacités de redondance intrinsèque. Cependant, il est important de tenir compte des contraintes suivantes. Tout d'abord, le degré élevé de miniaturisation de ces dispositifs les rend plus sensibles au rayonnement. Deuxièmement, le nombre énorme de cellules mémoire incluses dans le dispositif accroît sa vulnérabilité aux rayonnements. Enfin, la complexité de l'architecture en termes de multiplicité de cœurs, de communication inter- cœurs, de mémoire et d'E/S, affecte la fiabilité du système. Par conséquent, il est obligatoire d’évaluer la sensibilité face aux SEE des processeurs multi/many-cœurs afin de valider leur utilisation pour les applications où la fiabilité est cruciale.

Afin d'évaluer la fiabilité du dispositif, son taux d’erreur est nécessaire. Ce taux d’erreur peut être obtenu en extrapolant la section efficace issue des essais de radiation à l'environnement de rayonnement souhaité. En outre, étant donné que le taux d’erreur d'un système basé sur des processeurs dépend de l'application, des tests de radiation dynamique sont également nécessaires pour évaluer le système exécutant l'application cible. Cette dépendance implique que tout changement dans l'application nécessite de nouveaux tests. Cependant, le coût des tests de radiation et la disponibilité de l'installation de rayonnement les rend impraticables. Par conséquent, il est obligatoire d'utiliser une prédiction du taux d'erreur pour faire face à ces limitations.

Il y a peu de recherches dans la littérature qui étudient les effets de rayonnement sur les processeurs multi/many-cœur. Parmi eux, il y a des études représentatives telles que: référence

(Stolt, 2012) établit un modèle de section transversale dynamique pour un serveur multi-cœur basé sur un processeur quad-cœur en technologie 45nm CMOS. (Guertin, 2012) présente les résultats du test SEE au-dessous de 15 et 25 MeV ions de l'ITC Maestro de 49 noyaux, qui est un processeur Radiation Hardened By Design. In (Oliveira, 2014), on évalue la sensibilité au rayonnement d'une Graphic Processing Unit (GPU) conçue en technologie 28nm. Même si les travaux mentionnés présentent des résultats importants, ils visent à valider des dispositifs spécifiques et ne fournissent pas une approche générale à appliquer à n'importe quel processeur multi/many-cœur.

Contexte scientifique de la thèse

La présente thèse a été développée dans le cadre du projet CAPACITES au sein de l'équipe de Systèmes Intégrés Robustes (RIS) du laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés), et a été soutenue en partie par les autorités françaises par le biais du programme «Investissement d'Avenir» (projet CAPACITES) (CAPES), par le Secrétariat de l'Éducation Supérieure, la Science, la Technologie et l'Innovation de l'Équateur (SENESCYT) bourse 753-2012 et par l'Université des Forces Armées ESPE bourse 14-006-BP-DOC-ESPE-a2.

Le projet «Calcul Parallèle pour les applications critiques en temps et sécurité» (CAPACITES) implique des partenaires académiques et industriels dans le développement de la plateforme matérielle et logicielle basée sur le processeur many-cœur KALRAY MPPA pour répondre aux exigences des applications parallèles critiques sur le temps de réponse et la sûreté de fonctionnement. Les domaines d'application considérés dans ce projet sont représentatifs des systèmes embarqués critiques mettant en œuvre des puissances de calcul significatives et dont le déploiement est actuellement limité voire impossible en raison de choix technologiques qui ont été faits sur les plateformes multi-cœur traditionnelles. Les domaines d'application comprennent l'espace et l'avionique. De plus, parmi les résultats escomptés du projet, on peut citer: (1) la capacité de certification applicatives de bout en bout des couches sémantiques en suivant les principaux préceptes de la norme avionique DO178C, et (2) l'audit de la capacité des couches et de modelés d'exécution pour répondre aux exigences de certification aéronautique, à partir des processeurs many-cœur MPPA.

Objectifs et contributions de la thèse

L'objectif principal de la présente thèse est de fournir une approche de prédiction du taux d'erreur général et abordable pour évaluer la sensibilité des applications implémentées dans des processeurs multi et many-cœur exposés à des environnements radiatifs. En raison de la

complexité de l'architecture du dispositif, ce travail suit l'hypothèse que la principale contribution au taux d'erreur est fournie par des composants ne mettant pas en œuvre des mécanismes de protection.

Pour atteindre cet objectif, une extension de l'approche Code Emulating Upset (CEU) (Velazco, 2000) est proposée pour évaluer le taux d'erreur des applications implémentées dans les processeurs multi et many-cœur. L'approche CEU a été développée au laboratoire TIMA dans des travaux antérieurs pour prédire les taux d'erreur dans des architectures à processeur en combinant des campagnes d'injection de fautes et des essais sous radiation. L'extension proposée de l'approche CEU est appropriée pour injecter des fautes dans des processeurs multi et many-cœur en raison de la complexité de leur architecture et de la mise en œuvre de plusieurs niveaux de mémoires caches ainsi que des mémoires partagées. En outre, cette nouvelle proposition inclut des facteurs de déclassement de la mémoire et du temps d'exposition. Au cours de cette thèse, la théorie quantitative est appliquée à deux expériences qui sont combinées pour prédire le taux d'erreur. La première consiste à effectuer des essais sous radiation avec des neutrons de 14 Mev dans des accélérateurs de particules pour émuler un environnement radiatif agressif. Le second consiste à effectuer l'injection de fautes afin de simuler les conséquences des SEU dans l'exécution du programme.

Pour valider la généralité de cette approche, différents dispositifs COTS ont été sélectionnés en vue de représenter les aspects technologiques et architecturaux les plus pertinents des processeurs multi et many- cœur. Le premier était le processeur Freescale P2041, fabriqué en technologie SOI 45nm qui intègre quatre processeurs em500c. Le second a été le processeur Adapteva Epiphany E16G301 fabriqué en technologie CMOS 65nm qui intègre 16 processeurs. Le troisième était le processeur many-cœur Kalray MPPA-256 fabriqué en technologie CMOS 28nm qui intègre 16 clusters de calcul, chacun avec 17 processeurs VLIW. L'efficacité de l'approche sera déterminée en comparant le taux d'erreur prédit avec la réponse dynamique obtenue à partir des essais sous radiation. Cette comparaison permet de valider la pertinence de l'extension de l'approche CEU pour prédire le taux d'erreur des applications implémentées dans des processeurs multi/many-cœur.

Les principales contributions à l'état de l'art de cette recherche sont: (1) la proposition d'une approche générique pour déterminer la sensibilité dans le pire des cas d'un processeur multi-cœur implémentant ECC et parité dans leurs caches ou mémoires partagées qui ne peuvent pas être désactivées, (2) l'évaluation de la réponse dynamique du multi-cœur Freescale P2041 en implémentant une application *memory bound*, (3) l'identification de l'adresse cache comme source de résultats erronés dans le test dynamique de rayonnement, (4) la détermination de la sensibilité pire de cas du processeur many-cœur MPPA-256, (5) une évaluation de la réponse dynamique du MPPA-256 démontrant qu'en habilitant les mémoires caches, il est possible de

gagner en performance de l'application sans compromettre la fiabilité du processeur, (6) l'extension de l'approche CEU pour prédire le taux d'erreur d'applications implémentées dans des processeurs multi et many- cœur, (7) la sensibilité pire des cas du processeur multi-cœur Epiphany E16G301, (8) La fiabilité des processeurs étudiés en tenant compte des caractéristiques technologiques et architecturales.

Les résultats des trois premières contributions ont été publiés dans le journal IEEE Transactions on Nuclear Science (TNS) (Ramos, 2016). Les trois autres contributions ont été publiées dans le journal IEEE TNS (Vargas, 2017).

Dans le contexte spécifique du projet CAPACITES, cette thèse contribue à la tâche relative à l'étude de la fiabilité du processeur many- cœur MPPA-256 dans un environnement radiatif.

Organisation du manuscrit de thèse

Le manuscrit de la thèse est organisé en sept chapitres. Les trois premiers passent en revue toute la problématique des effets des radiations sur les circuits VLSI et donnent une synthèse des travaux existants.

Le premier chapitre est l'introduction de la thèse.

Le deuxième chapitre présente le contexte de la recherche décrivant les environnements de rayonnement spatial et atmosphérique afin d'introduire les effets du rayonnement naturel sur les circuits et systèmes électroniques. Plusieurs aspects de la caractérisation du circuit intégré au rayonnement sont abordés. À la fin du chapitre, les principaux problèmes liés aux processeurs multi-cœur et many-cœur sont décrits.

Le chapitre trois résume l'état de l'art des thèmes abordés dans la présente thèse: (1) la sensibilité des processeurs multi et many-cœur, (2) les techniques d'injection de fautes et (3) la prédiction du taux d'erreur des architectures basées sur le processeur. À la fin du chapitre, il est discuté la sélection de CEU comme une approche de base pour prédire le taux d'erreur d'applications implémentées dans des processeurs multi-cœur.

Le chapitre quatre définit la méthodologie proposée par cette recherche. Il détaille l'extension de l'approche CEU pour l'évaluation de la sensibilité face aux SEE et la prédiction du taux d'erreur des applications implémentées dans les processeurs multi et many-cœur, ainsi que les outils nécessaires pour la réalisation des expériences de rayonnement.

Le chapitre cinq explique la sélection des plates-formes utilisées pour valider l'approche et fournit une description des principales caractéristiques de celles-ci. Deux processeurs multi-

cœur et un processeur many-cœur sont proposés. En outre, le modèle de programmation et l'application de référence sont expliqués.

Le chapitre six présente les résultats expérimentaux de l'évaluation de l'approche dans les dispositifs cibles. Des campagnes d'injection de fautes sont effectuées pour obtenir la sensibilité face aux SEE de l'application. L'essai sous radiation statique avec des neutrons de 14 *Mev* fournit la sensibilité intrinsèque des processeurs multi-cœurs ou many-cœur. Des tests dynamiques sont effectués pour obtenir la réponse dynamique du dispositif et pour valider l'approche de prédiction. À la fin du chapitre, il est fourni une comparaison entre les dispositifs étudiés en termes de fiabilité.

Enfin, le chapitre sept résume la recherche et fournit des conclusions générales et des perspectives futures.

ii. Cadre théorique et travaux existants

Chapitre 2: Cadre théorique

Ce chapitre décrit les effets des radiations sur les circuits intégrés. Il présente un aperçu des environnements radiatifs spatiaux et atmosphériques où les circuits et systèmes électroniques fonctionnent. Les effets des particules énergétiques sur du matériau semi-conducteur sont également expliqués dans le but d'introduire les différents types d'événements singuliers qui perturbent les circuits et systèmes intégrés.

Les circuits et systèmes électroniques sont exposés aux rayonnements naturels et artificiels dans des environnements spatiaux et atmosphériques. Au cours des années soixante, de nombreux problèmes ont été observés dans l'électronique spatiale, bien qu'il ait été difficile de séparer les défaillances soft des autres formes d'interférences. En 1978, apparaissent les premières preuves de dysfonctionnements causés par les radiations présentes dans l'environnement spatial dans les circuits électroniques embarqués dans les engins spatiaux (May, 1979).

Dans l'espace extra-atmosphérique, il existe trois principaux types de sources radiatives qui affectent l'atmosphère terrestre:

- Les rayonnements cosmiques galactiques et extragalactiques
- Les rayonnements provenant du soleil tels que le vent solaire et les éruptions solaires
- Le champ magnétique terrestre qui comprend la magnétosphère et les ceintures de radiation

De nombreuses formes de rayonnement naturel et artificiel se rencontrent dans l'atmosphère terrestre, ce qui peut être bénéfique ou nuire à l'environnement. Une attention particulière doit être accordée aux rayonnements ionisants car ils peuvent être nocifs pour la

microélectronique. Les rayonnements cosmiques et les particules solaires bombardent constamment l'atmosphère terrestre et ainsi, entrant en collision avec les atomes d'azote et d'oxygène (Barth, 2003). Cette interaction génère dans l'atmosphère terrestre une « douche » de particules secondaires dont les produits sont des protons, des électrons, des neutrons, des ions lourds, des muons et des pions.

Dans les composants électroniques, les effets d'événements singuliers (Single Event Effect) se réfèrent à tous les effets possibles induits par l'interaction d'une unique particule énergétique avec un matériau semi-conducteur. Ces effets peuvent être le résultat d'une ionisation directe d'un matériau produit par un ion lourd ou un proton, ou d'une ionisation indirecte causée par des neutrons. Les SEEs sont la conséquence des impulsions transitoires de courant générées suite à l'ionisation provoquée par des particules incidentes. Ces impulsions peuvent provoquer des erreurs suite à leur propagation dans les circuits logiques ou causer un basculement d'état logique d'un bit dans les cellules mémoire ou les registres. Ils peuvent aussi produire des court-circuits masse-alimentation dans des circuits CMOS, provoquant ainsi la destruction du circuit (Single Event Latch-up, SEL). Les erreurs provoquées par les radiations sont classées en deux catégories (Gaillard, 2011) : des erreurs matérielles qui sont des erreurs non récupérables (destruction du circuit) et des erreurs logicielles qui peuvent être récupérées par réécriture de l'information ou par une réinitialisation.

Chapitre 3: État de l'art

Ce chapitre présente une revue de l'état de l'art des techniques SWIFI d'injection de fautes dans les circuits intégrés. Il décrit les principaux travaux concernant la sensibilité face aux SEEs des processeurs multi-cœur et many-cœur. Enfin, quelques méthodes pour prédire le taux d'erreur d'application des architectures à processeur sont présentées.

L'injection de fautes est une technique utile pour valider la fiabilité des dispositifs ou systèmes tolérants aux fautes (Arlat, 1990). Elle permet d'améliorer la couverture des tests matériel ou logiciel en introduisant des fautes de manière contrôlée dans le matériel ou les chemins de données du code du système dans le but d'observer leur comportement en présence de fautes.

Software-Implemented Fault Injection (SWIFI) est la technique d'injection de fautes la plus convenable pour évaluer les applications fonctionnant sur des dispositifs COTS car il ne nécessite pas de matériel complexe dédié, des netlists au niveau porte ou des modèles RTL qui sont décrits dans les langages de description matériel. Tous les types de fautes peuvent être injectés dans des cellules mémoires et registres accessibles qui représentent la zone la plus sensible de la puce. Le principal inconvénient des techniques SWIFI est leur intrusivité car ils modifient le programme. Cela peut affecter la planification des tâches. Si le timing n'est pas un

problème, ce type d'injection de fautes peut être considéré comme non intrusif. Sinon, le moment impliqué lors de l'injection peut perturber le fonctionnement du système (Ziade, 2012). Par conséquent, pour les systèmes embarqués critiques, une technique d'injection de fautes à faible intrusivité est nécessaire.

Concernant la sensibilité des processeurs multi/many-cœur, il existe trois études représentatives : (Stolt, 2012), (Guertin, 2012) et (Oliveira, 2014). Dans (Stolt, 2012), est établi un modèle de section efficace dynamique pour un serveur multi-cœur basé sur un processeur quadri-cœur en technologie 45nm CMOS. Dans (Guertin, 2012), sont présentés les résultats du test des SEE sous 15 et 25 MeV ions de l'ITC Maestro de 49 cœurs, qui est un processeur Radiation Hardened By Design (RHBD). Dans (Oliveira, 2014), est évaluée la sensibilité aux radiations d'une unité de traitement graphique (GPU) conçue en technologie 28 nm. Même si tous les travaux présentés ci-dessus sont intéressants, aucun d'eux ne donne des précisions sur le comportement dynamique et sur la fiabilité d'un multi-cœur en présence de rayonnement neutronique.

Au sujet de la prédiction du taux d'erreur, dans (Velazco, 2000) est présentée une approche de prédiction du taux d'erreur d'application basée sur SWIFI. Cette approche repose sur l'injection de bit-flips aléatoirement en emplacement et en temps en utilisant les capacités d'une carte d'application classique capable d'exécuter des séquences d'instructions et prenant en compte des interruptions asynchrones. La prédiction est faite en combinant la section efficace du composant, obtenue par des tests de radiation accélérés, avec le taux d'erreur issu de l'injection fautes. Dans (Mukherjee, 2003) est décrite une méthode pour générer des estimations précises du taux d'erreur des processeurs. Cette méthode définit le facteur de vulnérabilité architecturale (AVF) d'une structure, qui est la probabilité qu'une faute dans une structure particulière entraîne une erreur. Le taux d'erreur de la structure est le produit de son taux d'erreur brute (déterminé par la technologie des circuits) et l'AVF. (Wang, 2008) présente une méthode dépendant de l'environnement pour estimer le taux d'erreur induit par les neutrons, par la propagation de Single Event Transients (SET) à travers le circuit logique concerné. Le pulse est modélisé par deux paramètres: la probabilité d'occurrence et la fonction de densité de probabilité de la largeur d'impulsion. (Cabanas-Holmen, 2011) propose une approche pour prédire le taux d'erreur d'un processeur (RHBD) sur la base de la sensibilité des circuits constitutifs. L'approche *bottom-up* pour l'analyse intègre le taux d'erreur provoqué par les radiations dans différents types de circuits, y compris la sensibilité SEU/SET. Dans (Housanny, 2012), est proposé une méthodologie pour évaluer la sensibilité réelle de la mémoire cache d'une application visant à calculer un taux d'erreur plus précis. Cette méthode repose sur la surveillance des accès au cache pour évaluer la sensibilité du cache et nécessite un simulateur de microprocesseur.

Les approches mentionnées ont été validées pour des processeurs individuels. Parmi celles-ci, l'approche CEU a été choisie pour être étendue aux processeurs multi-cœurs et many-cœur pour les raisons suivantes:

- Cette approche peut être appliquée à tout processeur COTS contrairement à d'autres approches de prédiction basées sur le facteur de vulnérabilité architecturelle ou temporelle qui ne sont pas disponibles dans les composants COTS.
- Trois travaux ont démontré l'efficacité de l'approche CEU (Velazco, 2000), (Rezgui, 2001), (Peronnard, 2008).
- Il est capable de cibler les mémoires internes et externes et les registres du processeur accessibles du dispositif.
- La prédiction est basée sur la sensibilité réelle de l'application et non sur les modèles probabilistes ou des modèles d'utilisation de la mémoire cache.

En ce qui concerne l'approche CEU, l'inconvénient principal est l'impossibilité de cibler des zones sensibles comme les flip-flops, l'unité de commande et des latches. Cependant, la contribution au taux d'erreur de ces éléments est très faible.

iii. Méthodologie

Chapitre 4: Méthodologie et outils

Ce chapitre définit la méthodologie et les outils permettant l'évaluation de la sensibilité face aux SEE et de prédiction du taux d'erreur des applications implémentées sur les processeurs multi et many-cœur. Au début du chapitre, un aperçu général de l'approche proposée est fourni. Ensuite, les détails de l'approche sont discutés. Enfin, la description des outils est présentée.

Vue d'ensemble

La mesure appropriée pour effectuer l'évaluation de la sensibilité face aux SEE d'un système est le taux d'erreur. Typiquement, pour obtenir le taux d'erreur d'un système fonctionnant dans un environnement de rayonnement rude, des essais sous radiation dynamiques sont effectués pour extrapoler les résultats obtenus à l'environnement radiatif souhaité. Cependant, le coût et la disponibilité des installations de rayonnement constituent des contraintes majeures. En outre, en raison du taux d'erreur d'un système basé sur le processeur multi/many-cœur dépendant de l'application, il est nécessaire d'appliquer une méthodologie pour déterminer sa sensibilité effective au rayonnement à moindre coût. Il est certain qu'une option viable est la prédiction du taux d'erreur qui peut être basée sur les modèles probabilistes, les circuits constitutifs, l'analyse du cache et la sensibilité au pire des cas.

Le but de la présente thèse est l'extension de l'approche CEU à multi /many-cœur pour prédire le taux d'erreur d'application en combinant des campagnes d'injection de fautes et des essais sous radiation. Cette approche peut être implémentée dans n'importe quel processeur mono-cœur sans une profonde connaissance architecturale. Cependant, pour les processeurs multi / many-cœur, il existe plusieurs contraintes qui doivent être surmontées en raison de la complexité du dispositif, principalement liée à la gestion de la mémoire et aux communications inter-cœur. Pour valider la méthode CEU étendue, elle est implémentée sur trois dispositifs représentatifs qui ont des technologies de fabrication, des notions de conception et des modèles architecturaux différents. L'efficacité du procédé sera déterminée en comparant le taux d'erreur prédit avec le mesuré obtenu à partir de tests de radiation dynamique.

Enfin, la conception des essais à la fois d'injection de fautes et de rayonnement est traitée au moyen d'une théorie quantitative. La première tâche consiste à identifier les variables indépendantes et dépendantes impliquées dans les expériences.

Approche d'injection de fautes CEU

L'approche CEU considère une carte électronique à processeur qui comprend un dispositif capable d'exécuter un code séquentiel et de recevoir des interruptions asynchrones. Le programme effectue des opérations de lecture et d'écriture dans des registres internes accessibles via le jeu d'instructions, ainsi que dans des emplacements de mémoire interne ou externe. Le but de l'approche est de reproduire, de manière non intrusive, les effets des fautes de type Single Event Upset (SEU). Ceci est réalisé par l'assertion des signaux d'interruption asynchrones, pendant le temps d'exécution, afin de lancer un gestionnaire d'interruption (*CEU code*) qui produit l'erreur sélectionnée (SEU, MBU) dans une cellule de mémoire choisie aléatoirement (*CEU target*). Pour injecter un bit-flip dans une mémoire interne ou externe ainsi que dans un registre à usage général, les tâches suivantes doivent être effectuées par le code CEU :

- Lecture du contenu de la cellule mémoire.
- Exécution d'une opération XOR avec une valeur de masque appropriée qui contient un "1" pour les bits qui vont être inversés et "0" ailleurs.
- Ecriture de la valeur corrompue à son emplacement d'origine.

Ces tâches se produiront lorsqu'une interruption est obtenue à l'instant sélectionné. A ce stade, le processeur arrête l'exécution du programme, enregistre le contexte, exécute le code CEU responsable de l'injection de la faute et enfin restaure le contexte à partir de la pile afin de poursuivre l'exécution du programme. Une fois le CEU injecté, les résultats du programme doivent être comparés à un ensemble de valeurs de référence obtenues lorsque le programme s'exécute sans injection de fautes. Cependant, il y a deux limites à considérer:

- Les altérations qui se produisent pendant l'exécution ne peuvent pas être simulées.

- Il n'est pas possible de cibler toutes les zones sensibles possibles comme les flip-flops internes, l'unité de commande et les latchs à l'intérieur de l'architecture du processeur.

Extension de l'approche CEU a processeurs multi-cœur et many-cœur

Jusqu'à présent, l'approche CEU a été appliquée avec succès et validée pour les processeurs mono-cœur. Cependant, la complexité des processeurs a considérablement augmenté en raison de la technologie de fabrication, de l'architecture des dispositifs, du nombre de noyaux, des interconnexions, des fonctionnalités, etc. Il est donc raisonnable de valider l'utilisation de l'approche CEU pour des processeurs multi/many-cœur.

En raison du grand nombre de fonctionnalités et de pins, il n'est plus possible d'utiliser la plate-forme ASTERICS pour injecter des fautes dans ce type de dispositifs. Il est donc pratique d'étendre l'approche d'injection de fautes CEU pour des processeurs multi/many-cœur bénéficiant de la multiplicité de noyaux en utilisant l'un d'eux comme un injecteur de fautes tandis que les autres exécutent l'application choisie. Afin d'isoler l'injecteur de fautes, le dispositif doit être configuré en mode AMP. Pour effectuer l'injection de fautes, les interruptions inter-cœur sont utilisées.

Considérant que les processeurs multi/many-cœur mettent en œuvre différents types de cellules mémoires (mémoires partagées, mémoires caches, registres de processeurs, etc.), la prédiction du taux d'erreur total doit être exprimée comme la somme de la contribution individuelle de chaque composant.

$$\tau_{SEU} = \tau_{SEU\_SHARED} + \tau_{SEU\_CACHE} + \tau_{SEU\_REG} + \cdots \quad (0.1)$$

Cette thèse propose également l'ajout de facteurs de déclassement à la contribution des mémoires partagées et cache pour améliorer la précision de la prédiction. Ces facteurs dépendent de la mémoire utilisée par l'application et du temps d'exposition au rayonnement.

Facteur d'utilisation de la mémoire (MF)

C'est la quantité de mémoire utilisée par l'application par rapport à la mémoire totale du dispositif. Ce facteur est calculé compte tenu de l'espace mémoire occupé par le code et les données.

Facteur temps d'exposition (Etf)

Dans certains cas particuliers où le processeur multi/many-cœur fonctionne comme coprocesseur d'une carte de développement, il est possible d'avoir besoin de synchronisation entre

le coprocesseur et le processeur *host* pour enregistrer les résultats. Ceci est nécessaire lorsque les cœurs du coprocesseur n'ont pas d'accès direct aux fonctions *printf*. Cette synchronisation est réalisée au moyen d'un schéma maître-esclave pour garantir que chaque noyau de processeur signale les erreurs détectées. Toutefois, ce modèle de communication comporte une pénalité pour perte de temps d'exposition. Pendant la réalisation des communications, il est possible d'avoir des SEUs affectant la mémoire interne du DUT, qui ne sont pas détectés puisque au début de chaque phase d'exécution, les données de la mémoire sont réinitialisées. Même si la probabilité d'avoir cette condition est très faible, un facteur de perte de temps d'exposition (Etf) devrait être ajouté dans ce cas.

Modelé de programmation

Le modèle de programmation choisi pour évaluer cette approche est le modèle nu-métal. Malgré la complexité de la programmation de ce modèle, il est plus adapté car il permet de cibler plus de registres lors des campagnes d'injection de fautes. De plus, l'absence de système d'exploitation permet l'utilisation d'interruptions inter-cœur minimisant le temps d'injection et l'intrusivité. Par ailleurs, nu-metal offre à l'injecteur de fautes la possibilité d'injecter des bit-flips directement dans la mémoire partagée, bien que dans le mode AMP, chaque noyau possède son propre espace mémoire privé, puisqu'il n'y a aucun hyperviseur empêchant cette action. Par conséquent, dans des architectures de mémoire partagée, l'injection de fautes est réalisée de manière presque non intrusive, puisque l'injecteur de fautes lit et écrit directement dans la mémoire sans interférer avec l'exécution des autres noyaux.

Lorsque le développeur utilise le modèle nu-métal, il doit programmer toutes les fonctions du système: le démarrage des noyaux, la répartition des tâches entre les noyaux, la synchronisation entre les noyaux et/ou les tâches, l'accès aux ressources partagées, la cohérence, etc. Pour gérer la plupart de ces fonctions, un noyau maître est nécessaire ainsi qu'un espace mémoire partagé pour les communications entre processeurs. Par conséquent, un schéma maître-esclave a été mis en œuvre. Le noyau maître est défini comme le noyau injecteur de fautes, alors que les noyaux esclaves exécutent en parallèle l'application testée. D'une manière générale, après une réinitialisation, le noyau maître est responsable de l'amorçage et de la configuration du système. Ensuite, il démarre les autres noyaux. Une barrière a été mise en œuvre pour synchroniser l'initialisation du code d'exécution dans tous les noyaux. Dans le cas d'algorithmes distribués, pendant la réalisation de l'application, le maître distribue les tâches. Enfin, pour certains dispositifs où les cœurs de traitement n'ont pas d'accès direct aux E/S, le maître est également en charge de l'enregistrement des résultats. Les principales fonctions du maître sont résumées dans la liste suivante:

- Démarrage du système.
- Démarrage des autres cœurs.
- Initialisation des variables globales.
- Distribution des tâches.
- Synchronisation des communications en utilisant des communications inter-cœur.
- Injection de fautes.
- Enregistrement des résultats (facultatif)

iv. Campagnes d'injection de fautes

Des campagnes d'injection de fautes ont été effectuées dans des variables du programme pour simuler des SEUs perturbant les mémoires cache, les mémoires partagées et/ou dans les registres du processeur pour simuler des fautes dans des cellules de mémoire du registre. Les campagnes d'injection de fautes considèrent l'injection d'une SEU par exécution dans les variables du programme et registres. Les conséquences sont classées comme suit:

- **Fautes silencieuses**: elles se produisent lorsque la faute injectée n'entraîne aucune conséquence dans le résultat du programme. Les fautes silencieuses typiques sont celles affectant les données qui n'ont jamais été utilisées ou les données déjà utilisées par le programme.
- **Erreur de résultat**: les résultats du programme ne sont pas ceux attendus.
- **Exception**: le programme s'arrête. Il est principalement causé par des fautes injectées sur les variables de boucle ou les registres critiques.
- **Timeout**: lorsque le programme ne répond pas après une durée égale au temps d'exécution nominal.

Injection de fautes dans la mémoire

Dans cette approche, toutes les variables destinées à être utilisées par l'application sont placées, par logiciel, dans une mémoire partagée (cache ou interne). De cette façon, les variables peuvent être modifiées à tout moment par chacun des noyaux du processeur. Le noyau principal initialise les données qui seront utilisées par les autres noyaux. Une fois cette étape terminée, elle envoie un message via une interruption inter-processeur, pour indiquer les noyaux esclaves et pour démarrer l'exécution de l'application.

Une fois le message reçu par les nœuds esclaves, ils confirment la réception du message et démarrent l'exécution de l'application. Pendant que l'application s'exécute sur les noyaux esclaves, le noyau maître effectue l'injection de fautes. Il sélectionne de façon aléatoire le noyau

cible, l'instant d'injection (en termes de cycles d'horloge), l'adresse (index de matrice globale) et le bit à modifier. Lorsqu'un noyau esclave termine son calcul, il envoie un message au noyau maître indiquant que l'exécution a été terminée. Le noyau maître attend l'arrivée des messages de chaque noyau, puis compare les résultats obtenus avec un ensemble de résultats corrects précédemment obtenus.

Etant donné que le programmeur ne peut pas accéder directement aux mémoires cache, pour simuler des SEU, l'injection de fautes est effectuée dans la mémoire principale. Cette stratégie peut être appliquée à toute application testée. Il suffit de stocker toutes les variables de l'application ciblée dans un espace mémoire partagé.

Injection de fautes dans les registres du processeur

Les fautes sont injectées dans les registres accessibles du processeur. En raison du fait que le nœud principal n'a pas accès aux registres des autres cœurs, il peut exécuter une injection de fautes indirecte via le jeu d'instructions. L'injecteur de fautes effectue une interruption au nœud sélectionné dans lequel le gestionnaire d'interruption lance un code qui cible des registres accessibles permettant l'injection de fautes comme décrit précédemment. Les registres cibles à prendre en compte par cette approche sont les registres à usage général (GPR), les registres à points flottants (FPR) et les registres de fonctions spéciales (SFR) accessibles. Il est important de noter que la modification de ces registres peut provoquer des pannes critiques dans l'exécution du programme.

v. Tests sous radiations avec des neutrons

Les essais sous radiation sont le moyen le plus rapide d'obtenir des résultats significatifs sur la sensibilité d'un dispositif en une courte période de temps, car plus les particules frappent le dispositif, plus des SEEs sont observés. La reproductibilité de l'expérience est également un autre avantage majeur de cette stratégie. Deux types d'essais sont envisagés pour évaluer la sensibilité du dispositif: un test statique pour obtenir la sensibilité intrinsèque des cellules de mémoire (section efficace) et un test dynamique pour évaluer la réponse dynamique de l'application. De plus, la section statique est utilisée pour prédire le taux d'erreur.

Des campagnes de test sous neutrons ont été menées pour caractériser les processeurs multi/many-cœur, car les neutrons sont les particules les plus représentatives de l'atmosphère terrestre. Reference (Miller, 2013) discute de la pertinence de l'utilisation du test sous neutrons de 14 MeV pour caractériser la sensibilité face aux SEU des dispositifs numériques. Les sections

3 et 5 du document JESD89A du JEDEC STANDARD ont été utilisées comme protocole de base pour les essais expérimentaux.

Le plan d'essai doit définir la tension d'alimentation requise pour l'essai de radiation. Du fait que le taux d'erreur de nombreux dispositifs est très sensible à la tension d'alimentation, il est essentiel que ce paramètre soit mesuré et contrôlé avec précision (Jesd89A, 2006). Le réglage minutieux de la tension d'alimentation en fonction des valeurs du plan d'essai permet d'obtenir des résultats cohérents.

Logistique concernant les campagnes de test sous neutrons

Les essais sous neutrons ont été réalisés dans l'accélérateur GENEPI2 (Générateur de Neutrons Pulsé Intense) située au LPSC (Laboratoire de Physique Subatomique et Cosmologie) à Grenoble, France. Cet accélérateur a été développé à l'origine pour des expériences de physique nucléaire et, depuis 2014, il a été utilisé pour irradier des circuits intégrés issus de différentes technologies. GENEPI2 est un accélérateur électrostatique produisant des neutrons en impactant un faisceau de deuton sur une cible de Tritium (T). Après l'accélération à 220 keV, les deutons (d) produisent des neutrons (n) par la réaction de fusion d + T → n + 4He (Villa, 2014).

De la cible, les neutrons sont émis dans toutes les directions avec une énergie moyenne de 14 MeV. Le dispositif sous test (DUT) est placé face à la cible à une distance déterminée pour ajuster le flux neutronique. Pour les campagnes d'essai de rayonnement nous avons considéré, à première approximation, que seuls les neutrons émis complètement vers l'avant auront un impact sur le DUT. Alors que le DUT est entièrement exposé aux neutrons, un blindage neutronique dédié protège la plate-forme électronique de lecture.

Début 2015, une nouvelle cible T a été installée, générant un flux neutronique maximal de $4.5\text{x}10^{7}(\text{n.cm}^{-2}.s^{-1})$. Les objectifs principaux sont d'augmenter la production de neutrons et d'améliorer la fiabilité de l'accélérateur. La modification majeure consiste à remplacer la source actuelle d'ions deutérium par une nouvelle, basée sur la technique de résonance par cyclotron électronique (ECR), délivrant une intensité de faisceau plus élevée. Un nouveau moniteur pour la production de neutrons est en cours d'installation et sera mis en service. Ceci permettra d'améliorer la précision sur la mesure de la dose.

vi. Dispositifs sous test

Chapitre 5: Dispositifs ciblés

L'objectif de ce travail est de fournir une approche générale pour l'évaluation de la sensibilité face aux SEEs et la prédiction du taux d'erreur des applications implémentées en multi-cœur et many-cœur. Pour atteindre cet objectif, il est nécessaire de cibler différents dispositifs visant à représenter la diversité des processeurs multi/many-cœur. Les aspects technologiques et architecturaux les plus importants à considérer pour le choix du dispositif sont:

- Technologie de fabrication
- Architecture de la mémoire
- Nombre de cœurs
- Interconnexions
- Mécanismes de protection de la mémoire
- Performance
- Consommation d'énergie
- Fiabilité

La première plate-forme sélectionnée était la Freescale P2041 RDB basée sur le processeur quadri-cœur QorIQ P2041. Ce dispositif a été choisi en raison de:

- Technologie SOI 45 nm
- Haute performance
- Haute fiabilité: ECC et parité dans ses mémoires cache
- Basé sur l'architecture PowerPC, validé dans des travaux antérieurs pour l'aéronautique, cas de processeur unique (Peronnard, 2008)

Le deuxième était l'ordinateur Parallella qui intègre le processeur ARM A9 dual cœur utilisé comme *host* et le 16-cœur Epiphany E16G301 utilisé comme coprocesseur. L'épiphanie multi-cœur a été considéré en raison de:

- Haute performance
- Basse consommation d’énergie
- Abordabilité, permettant au grand public d'accéder au calcul parallèle.
- Architecture co-traitée
- Interconnexion (NoC)
- Open source

Le troisième était une carte de développement basée sur le processeur many-cœur Kalray MPPA-256 qui intègre 16 clusters de calcul ayant chacun 16 noyaux VLIW. Ce processeur à plusieurs noyaux a été sélectionné en raison de:

- Implémenté en technologie CMOS 28nm
- Haute performance
- Nombre significatif de cœurs
- Grande flexibilité de configuration
- Haute fiabilité: ECC et interleaving dans des mémoires partagées / parité dans des mémoires cache
- Efficacité énergétique exceptionnelle GFLOPS/W

Application testée

Une multiplication de matrice standard *n x n* (MM), qui est une application memory bound, a été sélectionnée pour être testée tout au long de cette recherche. Elle a été choisie car la multiplication matricielle est l'un des algorithmes les plus essentiels en algèbre numérique ainsi qu'en calcul distribué, scientifique et à haute performance (Ballard, 2012). Pour l'évaluation de l'application, deux scénarios sont proposés.

Dans le premier scénario, l'algorithme séquentiel de la multiplication matricielle a été utilisé. Ce scénario a été implémenté dans les multi-cœurs P2041 et l'Epiphany en utilisant un noyau du dispositif comme noyau principal. Chaque noyau exécute indépendamment la même multiplication matricielle (C = AxB) et compare ses résultats avec une valeur prédéfinie afin d'identifier des erreurs. La taille n de la matrice a été choisie en fonction de la capacité de mémoire de chaque dispositif afin de remplir le plus possible le cache et les mémoires partagées et de maintenir un compromis entre la quantité de mémoire utilisée et le temps d'exécution. Les matrices A, B et C sont localisées dans des vecteurs de mémoire consécutifs. Tous les éléments de la matrice A ont été remplis avec la même valeur *a*. De même, la matrice B a été remplie de *b*, donc le résultat attendu était *a x b x n* pour tous les éléments de la matrice C. Les matrices ont été remplies de valeurs fixes afin de simplifier l'analyse des données car une valeur connue aide à identifier quel bit ou bits ont été modifiés au cours de l'essai. De cette façon, on peut détecter des Multiple-Bit Upsets (MBUs) et des Multiple-Cell Upsets (MCUs). Il est important de noter que les résultats des expériences de rayonnement sont totalement indépendants des valeurs d'entrée. Pour le second scénario, chaque cluster de calcul du many-cœur MPPA-256 exécute indépendamment un algorithme parallèle de multiplication matricielle. Le code source est une version optimisée assembleur d'une multiplication matricielle collaborative *256x256*, répartie

entre les 16 éléments de traitement (PE). À l'intérieur de chaque cluster, le noyau du gestionnaire de ressources (RM) est le maître du système. Le calcul s'exécute plusieurs fois pour garantir que chaque cluster calcule suffisamment de temps pour que tous les clusters fonctionnent en parallèle. A, B et C sont des matrices à virgule flottante à précision simple. La taille de la matrice a été choisie de sorte que les données et le code restent dans la mémoire SMEM locale.

vii. Résultats expérimentaux

Chapitre 6: Résultats expérimentaux et évaluation des dispositifs

Les essais sous radiation réalisés avec des neutrons à 14 MeV et les campagnes d'injection de fautes sont des techniques utiles pour évaluer la sensibilité intrinsèque, la réponse dynamique et la sensibilité d'une application implémentée dans des processeurs multi et many-cœur. Voici un résumé de l'analyse des résultats présentés dans le chapitre six.

Processeur multi-cœur P2041

Les campagnes d'injection de fautes dans les variables du programme montrent que la matrice d'entrée *A* et spécialement l'entrée *B* sont beaucoup plus sensibles à SEU que la matrice de sortie en raison d'un temps d'exposition plus important. Les résultats montrent la pertinence de l'injection de fautes pour analyser le comportement d'une application en présence de SEU, offrant la possibilité de modifier le code du programme en fonction des résultats obtenus pour réduire l'impact des fautes dans les résultats de l'application.

Grâce à l'utilisation de la *machine check error report* lors des tests de radiation, il a été possible d'enregistrer tous les SEEs détectés, même ceux qui ont été corrigés par les mécanismes de protection mis en œuvre dans les mémoires cache. Elle a permis d'évaluer la sensibilité au rayonnement neutronique de la technologie SOI de 45 nm. Les résultats obtenus montrent que la technologie SOI de 45 nm est entre 3 et 5 fois moins sensible au rayonnement neutronique que sa contrepartie CMOS.

L'analyse des clusters d'erreurs avec le même motif répété simultanément sur tous les noyaux pendant l'essai sous radiation statique suggère qu'une particule perturbait une ressource partagée appartenant à l'infrastructure de connectivité (CoreNet Coherency Fabric). Ce fait appuie la nécessité d'une étude plus approfondie des conséquences des SEEs sur les communications inter-cœur, en dépit de la petite zone qu'elle occupe par rapport aux mémoires cache.

Des tests dynamiques ont démontré que la parité implémentée dans les mémoires cache L1 ne suffit pas à protéger les adresses cache et les tableaux de données. Les clusters d'erreurs, produites dans le même cycle de lecture par un MBU affectant le *tag* de l'adresse des caches L1

du noyau 1 et du noyau 2, mettent en évidence une possible implémentation 3-D des mémoires cache L1. Ces résultats suggèrent que les technologies émergentes dans l'implémentation du cache pourraient affecter potentiellement sa sensibilité au rayonnement.

En ce qui concerne la prédiction du taux d'erreur, on peut observer une sous-estimation produite par le fait que toutes les zones sensibles n'ont pas été ciblées pendant le test statique et les campagnes d'injection de fautes. De plus, les mécanismes de protection mis en œuvre dans leurs antémémoires peuvent influencer les tests et la prédiction des erreurs.

Epiphanie multi-cœur

Les résultats issus du test statique permettent d'estimer la sensibilité au pire des cas de la mémoire partagée implémentée en technologie CMOS 65nm. Pour cette expérience, le flux de neutrons a été réduit presque trois fois par rapport au flux utilisé dans les essais sous radiation du P2041. Ceci a été fait pour limiter les perturbations produites par les particules de neutrons dans le processeur *host*. Cependant, malgré les efforts pour protéger le reste des composants de la plate-forme, la carte SD contenant le système d'exploitation Linux a été corrompue dans l'une des expériences. Cela a été résolu en remplaçant la carte SD endommagée par une nouvelle.

En comparant le taux d'erreur prédit avec celui mesuré, on peut voir que l'approche proposée fournit une bonne approximation. La petite sous-estimation peut être expliquée du fait que toutes les zones sensibles du dispositif n'ont pas été ciblées pendant l'injection de fautes et le test statique.

MPPA-256 processeur many-cœur

En raison de la complexité de l'architecture de communication, il a été proposé une campagne d'injection de fautes au niveau des clusters pour éviter l'utilisation de services NOC, ce qui peut augmenter la latence pendant le processus d'injection de fautes. L'utilisation d'un modèle de programmation en carte à nue permet d'injecter des fautes dans les noyaux RM et PE. Cependant, de la même manière que pour les autres dispositifs, il n'était pas possible de cibler tous les registres ce qui conduit à une sous-estimation du taux d'erreur de prédiction. De plus, la précision de la prédiction a été affectée par les mécanismes de protection mis en œuvre dans les mémoires partagées et cache du MPPA-256.

Le test de rayonnement statique montre que l’ECC et l'interleaving mis en œuvre dans les SMEM des clusters sont très efficaces pour atténuer les erreurs, puisque tous les évènements ont été détectés et corrigés. Une question intéressante de ce test a été la multiplicité des MCUs allant de 2 à 7. Il donne quelques indices sur la façon dont l'organisation de la mémoire peut avoir un impact sur la propagation des erreurs produites par une seule particule.

Des tests dynamiques ont démontré qu'en habilitant les mémoires cache, il est possible d'augmenter les performances de l'application sans compromettre la fiabilité du dispositif, puisque les mémoires caches mettent en œuvre une protection de parité effective. D'autre part, les erreurs non corrigibles proviennent des GPR puisque les registres n'implémentent pas des mécanismes de protection.

Un compromis entre le temps d'exécution et la consommation d'énergie visant à minimiser l'impact du rayonnement neutronique a été atteint à une fréquence de 200 MHz. En ce qui concerne la tension de polarisation, on peut voir que le dispositif est moins sensible au rayonnement lorsqu'il fonctionne à sa tension nominale (0,9V). Par conséquent, la diminution de la tension de polarisation pour réduire la consommation d'énergie du dispositif est une question critique à considérer.

viii. Conclusions et travaux futurs

L'utilisation de processeurs multi-cœur et many- cœur allant de la sécurité critique aux applications commerciales augmente rapidement en raison de la demande croissante de haute performance, de fiabilité et de faible consommation d'énergie. De plus, leurs capacités de calcul parallèles et leur redondance les rendent des candidats idéaux pour les applications avioniques et spatiales. Cependant, l'intégration de plusieurs noyaux dans un seul circuit intégré conduit à une miniaturisation supplémentaire des transistors qui augmente leur sensibilité face aux SEEs. Bien que des améliorations technologiques et architecturales ainsi que des stratégies logicielles ont été mises au point pour atténuer les SEEs, la mise en œuvre de protections matérielles et logicielles supplémentaires implique une dégradation des performances et une augmentation de la consommation d'énergie. Il est donc obligatoire d'évaluer la sensibilité face aux SEEs des dispositifs multi-cœur pour valider leur applicabilité dans des environnements rudes ou pour des applications où la fiabilité est requise.

Il y a un intérêt croissant d'utiliser COTS multi et many-cœur pour l'avionique et les applications critiques pour la sécurité en raison de leur faible coût et gain de temps par rapport aux solutions complexes dédiées. Néanmoins, le choix de ces composants est toujours effectué de manière ponctuelle, ce qui pose des problèmes pour des estimations précises du coût et de la fiabilité. Par conséquent, la sélection de ces dispositifs COTS doit être basée sur l'évaluation de l'impact du rayonnement sur leur fiabilité au moyen de l'estimation de leur taux d'erreur.

Dans ce travail, il a été proposé une approche générale pour l'évaluation de la sensibilité face aux SEEs des applications implémentées dans des processeurs multi. Ceci a été réalisé par l'extension de l'approche CEU développée dans les laboratoires TIMA dans des travaux antérieurs, en tenant compte de l'évolution technologique et architecturale des architectures multi et many-cœur par rapport aux monoprocesseurs. Le taux d'erreur a été utilisé comme mesure pour cette évaluation. La prédiction du taux d'erreur a été obtenue en combinant le taux d'erreur de l'application issue des campagnes d'injection de fautes et la sensibilité dans le pire des cas du dispositif obtenue à partir des essais sous radiations. Afin de valider l'approche de prédiction, il a été nécessaire de comparer les taux d'erreur prédits et mesurés. Le résultat mesuré a été obtenu en exposant le dispositif ciblé au rayonnement neutronique tout en exécutant l'application souhaitée. Des campagnes d'injection de fautes ont été consacrées à l'injection de fautes dans des cellules mémoire et des registres accessibles en bénéficiant de la multiplicité de noyaux pour utiliser l'un d'eux comme injecteur de fautes alors que les autres exécutent l'application sélectionnée. Des expériences de radiations ont été réalisées avec des neutrons de 14 *Mev* à l'installation GENEPI2. Trois dispositifs COTS différents ont été évalués dans le cadre de cette thèse. Le premier était le Freescale P2041 processeur quadri-cœur fabriqué en technologie 45nm SOI. Le second a été le Kalray MPPA-256 many-cœur fabriqué en technologie 28nm CMOS. Le troisième a été l'Adapteva Epiphany E16G301 microprocesseur fabriqué en technologie 65nm CMOS.

Conclusions finales

Les résultats obtenus lors de l'évaluation du multi-cœur P2041 démontrent que l'injection de fautes permet d'identifier les vulnérabilités dans l'application et d'améliorer la stratégie de programmation pour réduire l'impact des fautes dans les résultats. À partir du test statique, il a été confirmé que la technologie SOI est plus robuste que la bulk CMOS traditionnelle. D'autre part, des tests dynamiques ont démontré qu'en dépit des mécanismes de protection parité et ECC, il y avait des erreurs dans le résultat de l'application causées par les MBU dans les adresses cache et les tableaux de données. Enfin, les résultats montrent une sous-estimation du taux d'erreur prédit, puisque toutes les zones sensibles n'ont pas été ciblées pendant le test statique et les campagnes d'injection de fautes. En outre, la mise en œuvre de ECC et la parité dans les mémoires cache du dispositif peuvent affecter la prédiction.

D'après l'évaluation du processeur multi-cœur Epiphany E16G301, on peut voir que l'extension proposée de l'approche CEU était efficace pour prédire le taux d'erreur d'application. Le fait que ce dispositif n'implémente pas de mécanismes de protection a permis une bonne

estimation du taux d'erreur, confirmant que les mécanismes de protection affectent les essais sous radiation et la prédiction du taux d'erreur (LaBel, 2005). Pendant les essais sous radiation dynamique, les matrices d'entrée ont également été vérifiées pour identifier des fautes silencieuses. Cela a été fait pour obtenir le taux d'erreur expérimental de l'application qui a une bonne corrélation avec le taux d'erreur obtenu à partir de l'injection de fautes.

L'évaluation du processeur many-coeur MPPA-256 montre que l'ECC et l'interleaving mis en œuvre dans les SMEM des clusters sont très efficaces pour atténuer les erreurs de type SEU, puisque toutes les SEUs détectées dans les SMEM ont été corrigées au cours du test statique. En outre, des tests dynamiques ont démontré qu'en habilitant les mémoires cache, il est possible d'augmenter la performance de l'application sans pénalité de fiabilité, puisque les mémoires cache mettent en œuvre une protection de parité effective. En ce qui concerne les résultats expérimentaux des rayonnements, la prédiction du taux d'erreur ne repose que sur la contribution des registres puisqu'ils n'ont pas de mécanismes de protection. Malgré la complexité de ce processeur à plusieurs noyaux, la prédiction du taux d'erreur a une petite sous-estimation qui confirme l'applicabilité de l'extension de l'approche CEU à ces dispositifs. Les raisons possibles de cette sous-estimation sont que seulement les registres accessibles ont été ciblés ou que l'infrastructure de communication n'a pas été ciblée ou que les mécanismes de protection peuvent influer sur la prédiction du taux d'erreur.

Les résultats globaux présentés dans cette thèse confirment que le processus de fabrication de la puce joue un rôle prépondérant dans la fiabilité du dispositif. Une comparaison FIT des dispositifs étudiés montre que le P2041 multi-cœur est le dispositif le plus fiable puisqu'il est construit en technologie SOI. Néanmoins, la différence avec le MPPA-256 construit en 28nm CMOS n'est pas si ample. En fait, si le FIT / Mb est considéré, le MPPA-256 est le dispositif le plus fiable. Les deux dispositifs mettent en œuvre ECC et la parité dans leurs mémoires internes. Cependant, l'efficacité du MPPA-256 est supérieure à l'efficacité du P2041 du fait de l'implémentation de l'interleaving dans ses mémoires partagées. Par conséquent, la sélection du dispositif approprié doit être effectuée sur la base des exigences de l'application (par exemple, l'espace mémoire disponible pour le code et les données, le nombre de noyaux travaillant en parallèle, etc.) et une analyse coûts-bénéfice.

En ce qui concerne l'épiphanie multi-coeur, il a un taux d'erreur environ 11 et 7 fois plus grand que celui des P2041 et MPPA-256 respectivement. En raison du fait que les meilleures technologies de fabrication ainsi que les protections matérielles augmentent considérablement le coût de l'appareil, des dispositifs tels que le multi-cœur Epiphany E16G306 pourraient être pris en compte dans des systèmes embarqués en fonction de la criticité de l'application et de l'environnement de travail. En fait, le rapport de la NASA: “Intelligent Hardware-Enable Sensor

and Software Safety and Health Management for Autonomous UAS” affirme que l'Epiphany multi-cœur peut être utilisé pour plusieurs applications terrestres et même avioniques.

Le fait qu'un processeur many-cœur construit sur 28nm CMOS a un FIT semblable à un multi-cœur 45nm SOI est très prometteur pour généraliser son utilisation sur le domaine des systèmes embarquées. Premièrement, parce que les mécanismes de protections implémentés sur le dispositif many-cœur ont réduit l'écart de fiabilité entre les deux technologies liées à la miniaturisation et au processus de fabrication (CMOS vs SOI). Deuxièmement, il supporte la possibilité d'utiliser des dispositifs COTS dans un système embarqué critique en raison de la grande capacité de traitement et de la capacité de mémoire interne importante. Les deux caractéristiques permettent de surmonter le principal problème de surcharge causé par la mise en œuvre de techniques matérielles tolérantes aux fautes. Par conséquent, l'utilisation de ce dispositif rend possible la mise en œuvre de techniques de redondance software pour améliorer la fiabilité du système en masquant d'autres SEE, conséquences qui n'ont pas été atténuées par les mécanismes de protection. Enfin, le coût élevé des dispositifs SOI par rapport aux dispositifs CMOS augmente l'utilisation de ces derniers.

Pour sélectionner le dispositif approprié à implémenter dans un système fonctionnant dans un environnement radiatif agressif, il est impératif d'évaluer la sensibilité face aux SEE des candidats, puis d'établir un compromis entre les coûts et la fiabilité en fonction de l'application et de l'environnement de travail. Cependant, l'évaluation basée sur des tests dynamiques de radiation est coûteuse en termes de temps et d'argent. Pour cette raison, une approche de prédiction est nécessaire. En outre, l'utilisation généralisée de processeurs multi/many-cœur dans des systèmes embarqués nécessite une approche de prédiction du taux d'erreur ajusté à ces dispositifs. À la connaissance de l'auteur, ce travail présente pour la première fois une approche consacrée à la prédiction du taux d'erreur des applications s'exécutant sur des processeurs multi et many-cœur. En dépit de la sous-estimation du taux d'erreur prédit, cette extension de l'approche CEU fournit des résultats utiles qui peuvent être considérés comme une validation préliminaire de COTS multi/many-cœur.

Plusieurs aspects rendent difficile le test des SEEs sur les processeurs multi/many-cœur et affectent la prédiction du taux d'erreur. L'un d'eux est la complexité de leur architecture qui intègre différents composants et fonctions sur la même puce. Dans l'approche développée dans cette thèse, la prédiction du taux d'erreur d'un système est présentée comme la somme des contributions individuelles de la prédiction du taux d'erreur par chaque composante a cellule-mémoire. Pour obtenir la contribution individuelle il est conseillé d'isoler les fonctionnalités. Cependant, ce n'est pas une tâche banale sur ces sortes de dispositifs et l'isolement ne peut pas être totalement accompli. Cela crée des difficultés pour identifier la zone où les fautes ont été produites. De plus, pour améliorer l'efficacité de la prédiction, il est convenable d'étendre cette

approche à d’autres composants du dispositif tels que l'infrastructure de communication. Un autre aspect est l'emballage complexe et la construction multicouches de circuits qui produisent des problèmes lors d'essais dans des installations de rayonnement en raison de limitation d'énergie de faisceau. Enfin, la mise en œuvre de mécanismes de détection et de correction d'erreurs ainsi que la carence de données expérimentales (en raison des coûts et de la disponibilité des installations de rayonnement) influent également sur la précision de l'approche de prédiction du taux d'erreur.

Perspectives futures

Le présent travail a présenté un premier aperçu dans la vaste étude de la sensibilité au rayonnement des processeurs multi-cœur et many-cœur. Pour poursuivre ce travail, les thèmes suivants peuvent être explorés:

- Validation de l'approche proposée en utilisant différents modèles de programmation: le modèle de programmation peut affecter notablement la sensibilité du dispositif en raison des ressources utilisées. Pour cette raison, il est utile de valider l'approche à l'aide de Posix et OpenMP.
- Validation d'une application spatiale réelle: elle peut se faire dans le cadre de la coopération industrielle et académique entre TIMA et des partenaires industriels tels que Thales Alenia Space ou des partenaires universitaires tels que le Centre Spatial Universitaire de Grenoble.
- Évaluation de l'infrastructure de communication: compte tenu du nombre croissant de noyaux dans les processeurs many-cœur émergents, le réseau de communication est en train de changer en implémentant des NoCs qui peuvent être combinés avec des infrastructures de type bus, pour l'interconnexion des noyaux et des clusters de traitement. Cela implique d'avoir plus de registres de contrôle et de données à cibler par l'injection de fautes et les tests sous radiation.
- Validation de l'approche de prédiction à l'aide d'ions lourds: pour envisager des COTS multi/many-cœur pour des applications spatiales, il est obligatoire de les exposer à des ions lourds. Cependant, il est important de tenir compte des contraintes matérielles telles que l'essai dans la chambre à vide et l'amincissement du silicium du dispositif.
- Application de techniques de redondance: cette approche de prédiction peut être combinée à la mise en œuvre d'une application redondante visant à accroître la fiabilité du dispositif.

Evaluation de la sensibilité face aux SEE et méthodologie pour la prédiction du taux d'erreurs d'applications implémentées dans des processeurs Multi-cœur et Many-cœur.

Résumé - La présente thèse propose une approche de prédiction du taux d'erreur et l'évaluation de la sensibilité des applications implémentées dans des processeurs multi-cœur et many-cœur exposés à des environnements radiatifs. Pour valider la généralité de cette approche, différents dispositifs COTS ont été ciblé en vue de représenter les aspects technologiques et architecturaux les plus pertinents des processeurs multi et many-cœur. Le premier a été le processeur quadri-cœurs P2041 de Freescale fabriqué en technologie 45nm SOI qui met en œuvre ECC et la parité dans leurs mémoires cache. Le second a été le microprocesseur Adapteva E16G301 fabriqué en technologie 65nm CMOS qui intègre 16 cœurs et n'implémente pas des mécanismes de protection. Le troisième a été le processeur many-cœur Kalray MPPA-256 fabriqué en technologie CMOS 28nm TSMC qui intègre 16 clusters de calcul chacun avec 17 cœurs, et met en œuvre ECC dans ses mémoires statiques et parité dans ses mémoires caches. L'évaluation de la sensibilité face aux SEE ainsi que la prédiction du taux d'erreur ont été réalisée par des essais sous radiation avec des neutrons de 14 *Mev* dans des accélérateurs de particules pour émuler un environnement de rayonnement agressif, et par injection de fautes dans des mémoires cache, des mémoires partagées ou des registres de processeur pour simuler les conséquences des SEU dans l'exécution du programme. Une analyse approfondie des erreurs observées a été effectuée pour identifier les vulnérabilités dans les mécanismes de protection. Des zones critiques telles que les adresses *tag* et les registres à usage général ont été principalement affectés pendant les expériences de rayonnement.

Mot Clés: Fiabilité, Test sous radiation, Injection de fautes, Processeurs Many-core, Processeurs Multi-core, Single Event Effect, Single Event Upset, robuste.

Evaluation of the SEE sensitivity and methodology for error rate prediction of applications implemented in Multi-core and Many-core processors

Abstract - The present thesis proposes an error-rate prediction approach and the evaluation of the sensitivity of applications implemented in multi and many-core processors exposed to harsh radiation environments. To validate the generality of this approach, three different COTS devices were targeted aiming at representing the most relevant technological and architectural aspects of multi/many-core processors. The first one was the Freescale P2041 quad-core processor manufactured in 45nm SOI technology which implements ECC and parity in their cache memories. The second one was the Adapteva Epiphany E16G301 microprocessor manufactured in 65nm CMOS process which integrates 16 processor cores and do not implement any protection mechanism. The third one was the Kalray MPPA-256 many-core processor manufactured in 28nm TSMC CMOS technology which integrates 16 compute clusters each one with 17 processor cores, and implements ECC in its static memories and parity in its cache memories. The SEE sensitivity evaluation and the error-rate prediction was accomplished by combining radiation experiments with 14 *Mev* neutrons in particle accelerators to emulate a harsh radiation environment, and fault injection in cache memories, shared memories or processor registers, to simulate the consequences of SEUs in the execution of the program. A deep analysis of the observed errors was carried out to identify vulnerabilities in the protection mechanisms. Critical zones such as address *tag* and general purpose registers were mainly affected during the radiation experiments.

Keywords: Reliability, Radiation ground test, Fault Injection, Many-core processor, Multi-core Processor, Single Event Effect, Single Event Upset, Robust.

Thèse préparé au Laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés), 46 avenue Félix Viallet, 38031, Grenoble Cedex, France.

ISBN: 978-2-11-129226-0